ACTIONSCRIPT™ 2.0
リファレンスガイド

Adobe

ActionScript 2.0 リファレンスガイド

目 次

第 1 章

ActionScript 言語エレメント

1

このセクションでは、グローバル関数とグローバルプロパティ (これらの言語エレメントは、ActionScript™ クラスには属していません)、コンパイラディレクティブ、および ActionScript™ で使用され ECMAScript (ECMA-262) Edition 4 の言語仕様草案で定義されている定数、演算子、ステートメント、およびキーワードに関して、シンタックス、使用法、およびコードサンプルをそれぞれ示します。

コンパイラディレクティブ

このセクションでは、コンパイラで特定の命令を前処理するために ActionScript ファイルで使用するディレクティブについて説明します。コンパイラディレクティブを含む行の最後にはセミコロン (;) を記述しません。

コンパイラディレクティブ一覧

ディレクティブ	説明
#endinitclip	初期化アクションのブロックの終わりを示します。
#include	指定したファイル内のコマンドを呼び出し元のスクリプトにインクルードし、そのスクリプトの一部であるかのように扱います。
#initclip	初期化アクションのブロックの始まりを示します。

#endinitclip ディレクティブ

```
#endinitclip
```

初期化アクションのブロックの終わりを示します。

#endinitclip ディレクティブを含む行の最後にはセミコロン (;) を記述しません。

対応バージョン： ActionScript 1.0、Flash Player 6.0

例

```
#initclip
...initialization actions go here...
#endinitclip
```

#include ディレクティブ

```
#include "[path]filename.as":String
```

指定したファイル内のコマンドを呼び出し元のスクリプトにインクルードし、そのスクリプトの一部であるかのように扱います。#include ディレクティブは、コンパイル時に起動されます。このため、外部ファイルに何らかの変更を行った場合には、ファイルを保存し、その外部ファイルを使用している FLA ファイルを再コンパイルする必要があります。

#include ステートメントを含むスクリプトに対して [シンタックスチェック] ボタンを使用すると、インクルードされるファイルのシンタックスもチェックされます。

#include アクションは、FLA ファイルおよび外部スクリプトファイルで使用できますが、ActionScript 2.0 クラスファイルでは使用できません。

インクルードするファイルのパスには相対パスも絶対パスも指定できますが、パスを省略することもできます。パスを省略する場合は、AS ファイルが次のいずれかの場所に存在する必要があります。

- FLA ファイルと同じディレクトリ。#include ステートメントを含むスクリプトと同じディレクトリ。
- 次のいずれかのグローバルな Include ディレクトリ。--Windows 2000 または Windows XP : C:¥Documents and Settings¥‹ ユーザー名 ›¥LocalSettings¥ Application Data¥Adobe¥Flash CS3¥‹ 言語 ›¥Configuration¥Include --Macintosh OS X: Hard Drive/Users/Library/Application Support/Adobe/FlashCS3/‹ 言語 ›/Configuration/Include
- Flash CS3 プログラム ¥‹ 言語 ›¥First Run¥Include ディレクトリこのディレクトリにファイルを保存した場合、Flash を次回起動したときに、ファイルはグローバル Include ディレクトリにコピーされます。

AS ファイルの相対パスを指定するには、現在のディレクトリを表す 1 つのドット (.)、親ディレクトリを表す 2 つのドット (..)、およびサブディレクトリを表すスラッシュ (/) を使用します。次の例を参照してください。

AS ファイルの絶対パスを指定するには、そのプラットフォーム (Macintosh または Windows) でサポートされている形式を使用します。次の例を参照してください。ただし、この方法では、スクリプトをコンパイルするのに使用するコンピュータで同じディレクトリ構造を維持する必要があるので、推奨されません。

ファイルを First Run/Include ディレクトリまたはグローバル Include ディレクトリに保存する場合は、これらのファイルをバックアップしてください。Flash をアンインストールした後で再びインストールする必要性が生じた場合、これらのディレクトリは削除された後に上書きされてしまいます。

#include ディレクティブを含む行の最後にはセミコロン (;) を記述しません。

対応バージョン： ActionScript 1.0、Flash Player 4.0

パラメータ

`[path]filename.as:String` - *filename.as* [アクション] パネルまたは現在のスクリプトに追加するスクリプトのファイル名とパス (オプション)。推奨されるファイル拡張子は *.as* です。

例

次の例は、スクリプトにインクルードするファイルのパスを指定するさまざまな方法を示しています。

```
// Note that #include statements do not end with a semicolon (;)
// AS file is in same directory as FLA file or script
// or is in the global Include directory or the First Run/Include directory
#include "init_script.as"

// AS file is in a subdirectory of one of the above directories
// The subdirectory is named "FLA_includes"
#include "FLA_includes/init_script.as"
// AS file is in a subdirectory of the script file directory
// The subdirectory is named "SCRIPT_includes"
#include "SCRIPT_includes/init_script.as"
// AS file is in a directory at the same level as one of the above directories
// AS file is in a directory at the same level as the directory
// that contains the script file
// The directory is named "ALL_includes"
#include "../ALL_includes/init_script.as"

// AS file is specified by an absolute path in Windows
// Note use of forward slashes, not backslashes
#include "C:/Flash_scripts/init_script.as"

// AS file is specified by an absolute path on Macintosh
#include "Mac HD:Flash_scripts:init_script.as"
```

#initclip ディレクティブ

#initclip [order:Number]

初期化アクションのブロックの始まりを示します。複数のクリップを同時に初期化する場合は、最初に初期化するクリップを order パラメータによって指定できます。ムービークリップシンボルを定義すると、初期化アクションが実行されます。ムービークリップが書き出されたシンボルである場合は、SWF ファイルのフレーム 1 にあるアクションの前に初期化アクションが実行されます。それ以外の場合は、関連するムービークリップシンボルの最初のインスタンスを含むフレームのアクションの直前に実行されます。

初期化アクションは、SWF ファイルの再生時に 1 回だけ実行されます。初期化アクションは、クラス定義や登録など、1 回だけ実行する初期化処理に使用してください。

#initclip ディレクティブを含む行の最後にはセミコロン (;) を記述しません。

対応バージョン： ActionScript 1.0、Flash Player 6.0

パラメータ

order:Number (オプション) - #initclip コードのブロックの実行順を指定する負以外の整数。このパラメータは省略可能です。指定する場合は変数ではなく整数リテラルを使用し、値は 16 進法ではなく 10 進法で表す必要があります。1 つのムービークリップシンボルに複数の #initclip ブロックがある場合、そのムービークリップシンボルで指定された最後の order 値が、シンボル内のすべての #initclip ブロックに使用されます。

例

次の例では、ムービークリップインスタンス内のフレーム 1 に ActionScript を配置します。variables.txt テキストファイルは同じディレクトリにあるものとします。

```
#initclip

trace("initializing app");

var variables:LoadVars = new LoadVars();

variables.load("variables.txt");

variables.onLoad = function(success:Boolean) {

 trace("variables loaded:"+success);

 if (success) {
 for (i in variables) {
 trace("variables."+i+" = "+variables[i]);
 }
 }
};

#endinitclip
```

定数

定数とは、値が変化しないプロパティを表すのに使用する変数のことです。このセクションでは、どのスクリプトでも使用できるグローバル定数について説明します。

定数一覧

オプション	定数	説明
	false	true の逆を表す一意のブール値です。
	Infinity	正の無限大を表す IEEE-754 値を指定します。
	-Infinity	負の無限大を表す IEEE-754 値を指定します。
	NaN	NaN (非数) を表す IEEE-754 値を保持する定義済みの変数です。
	newline	キャリッジリターン文字 (\r) を挿入します。これにより、コードが生成するテキスト出力に空白行ができます。
	null	変数に代入できる特別な値、またはデータがない場合に関数から返される特別な値です。
	true	false の逆を表す一意のブール値です。
	undefined	通常、変数に値がまだ割り当てられていないことを示すために使用される特別な値。

false 定数

true の逆を表す一意のブール値です。

自動的な型指定により false を数値に変換すると、その結果は 0 となります。false をストリングに変換すると、その結果は "false" となります。

対応バージョン： ActionScript 1.0、Flash Player 5

例

この例では、自動的な型指定によって false がどのような数値やストリングに変換されるかを示します。

```
var bool1:Boolean = Boolean(false);

// converts it to the number 0
trace(1 + bool1); // outputs 1

// converts it to a string
trace("String: " + bool1); // outputs String: false
```

Infinity 定数

正の無限大を表す IEEE-754 値を指定します。この定数の値は、Number.POSITIVE_INFINITY と同じです。

対応バージョン： ActionScript 1.0、Flash Player 5

関連項目

POSITIVE_INFINITY（Number.POSITIVE_INFINITY プロパティ）

-Infinity 定数

負の無限大を表す IEEE-754 値を指定します。この定数の値は、Number.NEGATIVE_INFINITY と同じです。

対応バージョン： ActionScript 1.0、Flash Player 5

関連項目

NEGATIVE_INFINITY（Number.NEGATIVE_INFINITY プロパティ）

NaN 定数

NaN（非数）を表す IEEE-754 値を保持する定義済みの変数です。数値が NaN かどうかを判別するには、isNaN() を使用します。

対応バージョン： ActionScript 1.0、Flash Player 5

関連項目

isNaN 関数, NaN（Number.NaN プロパティ）

newline 定数

キャリッジリターン文字 (\r) を挿入します。これにより、コードが生成するテキスト出力に空白行ができます。コード内の関数またはステートメントで検索される情報のためのスペースを作成するには、newline を使用します。

対応バージョン： ActionScript 1.0、Flash Player 4

例

次の例では、newline を使って、trace() ステートメントの出力を複数行に表示しています。

```
var myName:String = "Lisa", myAge:Number = 30;
trace(myName+myAge);
trace("-----");
trace(myName+newline+myAge);
// output:
Lisa30
-----
Lisa
30
```

関連項目

trace 関数

null 定数

変数に代入できる特別な値、またはデータがない場合に関数から返される特別な値です。null は、存在しない、または定義されていないデータ型を表す値として使用されます。

対応バージョン： ActionScript 1.0、Flash Player 5

例

次の例では、インデックスの付いた配列の最初の 6 つの値をチェックし、値が設定されていない場合 (値 == null) にメッセージを出力します。

```
var testArray:Array = new Array();
testArray[0] = "fee";
testArray[1] = "fi";
testArray[4] = "foo";

for (i = 0; i < 6; i++) {
 if (testArray[i] == null) {
 trace("testArray[" + i + "] == null");
 }
}
```

出力は次のようになります。

```
testArray[2] == null
testArray[3] == null
testArray[5] == null
```

true 定数

false の逆を表す一意のブール値です。自動的な型指定により true を数値に変換すると、その結果は 1 となります。true をストリングに変換すると、その結果は "true" となります。

対応バージョン： ActionScript 1.0、Flash Player 5

例

次の例では、if ステートメントで true を使用しています。

```
var shouldExecute:Boolean;
// ...
// code that sets shouldExecute to either true or false goes here
// shouldExecute is set to true for this example:

shouldExecute = true;

if (shouldExecute == true) {
 trace("your statements here");
}

// true is also implied, so the if statement could also be written:
// if (shouldExecute) {
// trace("your statements here");
// }
```

次の例では、自動的な型指定によって true を数値 1 に変換する方法を示しています。

```
var myNum:Number;
myNum = 1 + true;
trace(myNum); // output: 2
```

関連項目

false 定数, Boolean

undefined 定数

通常、変数に値がまだ割り当てられていないことを示すために使用される特別な値。未定義の値を参照すると、特別な値 undefined が返されます。ActionScript コード typeof(undefined) は、ストリング "undefined" を返します。タイプ undefined の唯一の値は undefined です。

Flash Player 6 以前用にパブリッシュされたファイルでは、String(undefined) は "" (空のストリング) になります。Flash Player 7 以降で使用するためにパブリッシュされたファイルでは、String(undefined) の値は "undefined" になります (undefined がストリングに変換されます)。

Flash Player 6 以前用にパブリッシュされたファイルでは、Number(undefined) は 0 になります。Flash Player 7 以降用にパブリッシュされたファイルでは、Number(undefined) は NaN になります。

値 undefined は特別な値 null に似ています。null と undefined を等価演算子 (==) で比較すると、結果は true になります。ただし、null と undefined を厳密な等価演算子 (===) で比較すると、結果は false になります。

対応バージョン： ActionScript 1.0、Flash Player 5

例

次の例では、変数 x が宣言されていないので、値は undefined になります。

コードの第 1 セクションでは、等価演算子 (==) を使って値 x を値 undefined と比較し、その結果を [出力] パネルに表示します。

コードの第 2 セクションでは、等価 (==) 演算子を使って値 null と undefined を比較します。

```
// x has not been declared
trace("The value of x is "+x);

if (x == undefined) {
 trace("x is undefined");
} else {
 trace("x is not undefined");
}

trace("typeof (x) is "+typeof (x));

if (null == undefined) {
 trace("null and undefined are equal");
} else {
 trace("null and undefined are not equal");
}
```

次の結果が [出力] パネルに表示されます。

```
The value of x is undefined
x is undefined
typeof (x) is undefined
null and undefined are equal
```

グローバル関数

ここでは、ActionScript を使用している SWF ファイルで使用できるビルトイン関数について説明します。グローバル関数は、データ型の処理 (たとえば `Boolean()`、`int()`)、デバッグ情報の作成 (`trace()`)、Flash Player やブラウザとの通信 (`fscommand()`) など、各種の一般的なプログラミングタスクを扱います。

グローバル関数一覧

オプション	シグネチャ	説明
	`Array([numElements:Number], [elementN:Object]) : Array`	新しい空の配列を作成します。また、指定されたエレメントを配列に変換します。
	`asfunction(function:String, parameter:String)`	HTML テキストフィールドの URL 専用プロトコルであり、HREF リンクから ActionScript 関数を呼び出すことができます。
	`Boolean(expression:Object) : Boolean`	*expression* パラメータをブール値に変換し、`true` または `false` を返します。
	`call(frame:Object)`	**非推奨** Flash Player 5 以降では使用しないでください。このアクションの代わりに `function` ステートメントを使用します。 呼び出されたフレームで、そのフレームに再生ヘッドを移動せずに、スクリプトを実行します。
	`chr(number:Number) : String`	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに `String.fromCharCode()` を使用します。 ASCII コード番号を文字に変換します。
	`clearInterval (intervalID:Number)`	`setInterval()` の呼び出しを停止します。
	`clearTimeout (id:Number)`	指定した `setTimeout()` 呼び出しをキャンセルします。
	`duplicateMovieClip (target:Object, newname:String, depth:Number)`	SWF ファイルの再生中にムービークリップのインスタンスを作成します。
	`escape(expression:String) : String`	パラメータをストリングに変換し、URL エンコードします。この場合、英数字以外のすべての文字は % が付いた 16 進シーケンスで置き換えられます。
	`eval(expression:Object) : Object`	変数、プロパティ、オブジェクト、ムービークリップに名前でアクセスします。

オプション	シグネチャ	説明
	`fscommand(command:String, parameters:String)`	SWF ファイルが、Flash Player または Flash Player のホストプログラム (Web ブラウザなど) と通信できるようになります。
	`getProperty(my_mc:Object, property) : Object`	ムービークリップ *my_mc* の指定プロパティの値を返します。
	`getTimer() : Number`	SWF ファイルを再生し始めてからの経過時間をミリ秒単位で返します。
	`getURL(url:String, [window:String], [method:String])`	特定の URL からウィンドウにドキュメントをロードしたり、定義済みの URL に存在する別のアプリケーションに変数を渡したりします。
	`getVersion() : String`	Flash Player のバージョンとプラットフォーム情報を含むストリングを返します。
	`gotoAndPlay([scene:String], frame:Object)`	シーン内の指定されたフレームに再生ヘッドを送り、そのフレームから再生します。
	`gotoAndStop([scene:String], frame:Object)`	シーン内の指定されたフレームに再生ヘッドを送り、停止します。
	`ifFrameLoaded([scene:String], frame:Object)`	**非推奨** Flash Player 5 以降では使用しないでください。この関数は使用されなくなりました。`MovieClip._framesloaded` プロパティを使用することをお勧めします。 特定のフレームの内容がローカルに使用できるかどうかを確認します。
	`int(value:Number) : Number`	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに、正の値には `Math.floor()` を使用し、負の値には `Math.ceil` を使用します。 小数値を切り捨てることによって、10 進数の整数部を取り出します。したがって、負数の扱いは Math.floor() メソッドと異なります。
	`isFinite(expression:Object) : Boolean`	*expression* を評価し、有限大である場合は `true` を、無限大または負の無限大である場合は `false` を返します。
	`isNaN(expression:Object) : Boolean`	パラメータを評価し、値が `NaN` (非数) である場合は `true` を返します。

オプション	シグネチャ	説明
	length(expression:String, variable:Object) : Number	**非推奨** Flash Player 5 以降では使用しないでください。この関数およびすべてのストリング関数は使用されなくなりました。String クラスのメソッドと String.length プロパティを使用して同じ処理を行うことをお勧めします。 指定されたストリングまたは変数の長さを返します。
	loadMovie(url:String, target:Object, [method:String])	元の SWF ファイルの再生中に SWF ファイルまたは JPEG ファイルを Flash Player 内にロードします。
	loadMovieNum(url:String, level:Number, [method:String])	ロードした元の SWF ファイルの再生中に SWF ファイルまたは JPEG ファイルを Flash Player のレベル内にロードします。
	loadVariables(url:String, target:Object, [method:String])	テキストファイルや、ColdFusion、CGI スクリプト、ASP (Active Server Pages)、パーソナルホームページ (PHP)、または Perl スクリプトで作成されたテキストなどの外部ファイルからデータを読み取り、ターゲットムービークリップの変数に値を設定します。
	loadVariablesNum (url:String, level:Number, [method:String])	テキストファイルや、ColdFusion、CGI スクリプト、ASP、PHP、または Perl スクリプトで作成されたテキストなどの外部ファイルからデータを読み取り、Flash Player の特定のレベルの変数に値を設定します。
	mbchr(number:Number)	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに String.fromCharCode() メソッドを使用します。 ASCII コード番号をマルチバイト文字に変換します。
	mblength(string:String) : Number	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに String クラスのメソッドとプロパティを使用します。 マルチバイト文字のストリング長を返します。
	mbord(character:String) : Number	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに String.charCodeAt() を使用します。 指定された文字をマルチバイト文字コード番号に変換します。
	mbsubstring(value:String, index:Number, count:Number) : String	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに String.substr() を使用します。 マルチバイト文字ストリングから、マルチバイト文字ストリングを抽出します。
	MMExecute(command:String) : String	ActionScript から Flash JavaScript API (JSAPI) のコマンドを発行できます。

オプション	シグネチャ	説明
	nextFrame()	次のフレームに再生ヘッドを送ります。
	nextScene()	次のシーンのフレーム 1 に再生ヘッドを送ります。
	Number(expression: Object) : Number	*expression* パラメータを数値に変換します。
	Object([value: Object]) : Object	空のオブジェクトを新規作成するか、指定された数値、ストリング、またはブール値をオブジェクトに変換します。
	on(mouseEvent:Object)	アクションをトリガするマウスイベントまたはキー押下を指定します。
	onClipEvent (movieEvent:Object)	ムービークリップの特定のインスタンスに対して定義されたアクションを起動します。
	ord(character: String) : Number	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに String クラスのメソッドとプロパティを使用します。 文字を ASCII コード番号に変換します。
	parseFloat(string: String) : Number	ストリングを浮動小数に変換します。
	parseInt(expression: String, [radix:Number]) : Number	ストリングを整数に変換します。
	play()	タイムライン内で再生ヘッドを前へ進めます。
	prevFrame()	前のフレームに再生ヘッドを送ります。
	prevScene()	前のシーンのフレーム 1 に再生ヘッドを送ります。
	print(target:Object, boundingBox:String)	パラメータ (bmovie、bmax、または bframe) で指定された境界に従って、target ムービークリップを印刷します。
	printAsBitmap (target:Object, boundingBox:String)	パラメータ (bmovie、bmax、または bframe) で指定された境界に従って、target ムービークリップをビットマップとして印刷します。
	printAsBitmapNum (level:Number, boundingBox:String)	パラメータ (bmovie、bmax、または bframe) で指定された境界に従って、Flash Player 内のレベルをビットマップとして印刷します。
	printNum(level: Number, boundingBox:String)	boundingBox パラメータ (bmovie、bmax、bframe) で指定された境界に従って、Flash Player 内のレベルを印刷します。

オプション	シグネチャ	説明
	random(value: Number) : Number	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに Math.random( ) を使用します。 0 から <code>value</code> パラメータで指定された整数まで (この整数は含まない) の間のランダムな整数を返します。
	removeMovieClip (target:Object)	指定されたムービークリップを削除します。
	setInterval (functionReference: Function, interval:Number, [param:Object], objectReference: Object, methodName:String) : Number	SWF ファイルの再生時に一定の間隔で関数、またはオブジェクトのメソッドを呼び出します。
	setProperty(target: Object, property:Object, expression:Object)	ムービークリップの再生時にムービークリップのプロパティ値を変更します。
	setTimeout (functionReference: Object, delay:Number, args:Object) : Number	ミリ秒単位で指定した遅延時間の経過後に、指定した関数を実行します。
	showRedrawRegions (enable:Boolean, [color:Number])	デバッガプレーヤーに、再描画されるスクリーン領域をアウトラインで囲む機能を提供します。
	startDrag(target: Object, [lock:Boolean], [left,top,right, bottom:Number])	ムービーの再生中に *target* ムービークリップをドラッグ可能にします。
	stop()	再生中の SWF ファイルを停止します。
	stopAllSounds()	再生ヘッドを停止せずに、SWF ファイルで再生中のサウンドをすべて停止します。
	stopDrag()	現在実行中のドラッグ操作を停止します。
	String(expression: Object) : String	指定されたパラメータのストリング表現を返します。

オプション	シグネチャ	説明
	substring(string: String, index:Number, count:Number) : String	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに String.substr() を使用します。 ストリングの一部を抽出します。
	targetPath (targetObject: Object) : String	*movieClipObject* のターゲットパスをストリングとして返します。
	tellTarget(target: String, statement(s))	**非推奨** Flash Player 5 以降では使用しないでください。ドット (.) 表記と with ステートメントを使用することをお勧めします。 *statements* パラメータで指定された指示を、*target* パラメータで指定されたタイムラインに適用します。
	toggleHighQuality()	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに _quality を使用します。 Flash Player でアンチエイリアス処理のオンとオフを切り替えます。
	trace(expression: Object)	式を評価し、結果を出力します。
	unescape(string: String) : String	パラメータ *x* をストリングとして評価し、URL エンコードされた形式からストリングをデコード (すべての 16 進シーケンスを ASCII 文字に変換) して、ストリングを返します。
	unloadMovie(target: Object)	loadMovie() を使ってロードしたムービークリップを Flash Player から削除します。
	unloadMovieNum(level: Number)	loadMovieNum() を使ってロードした SWF ファイルやイメージを Flash Player から削除します。
	updateAfterEvent()	ハンドラ内で呼び出した場合、または setInterval() を使って呼び出した場合に、表示を更新します。

Array 関数

```
Array() : Array
Array(numElements:Number) : Array
Array(element0:Object, [element1, element2, ...elementN]) : Array
```

長さが 0 以上の新しい配列、または指定したエレメントのリスト (各種データ型など) により格納される配列を作成します。

Array() を使用して、次のいずれかを作成します。

- 空の配列
- 長さのみが指定されておりエレメント値が定義されていない配列

■ エレメントが特定の値を持つ配列

この関数は、Array コンストラクタを使用して配列を作成するのに似ています。Array クラスのコンストラクタを参照してください。

エレメントの数 (numElements)、または異なる種類を含むエレメントリスト (element0, element1, ... elementN) を渡すことができます。

複数のデータ型を指定できるパラメータは、シグネチャで Object としてリストされます。

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

numElements:Number (オプション) - 配列内のエレメント数を指定する正の整数。numElements またはエレメントリストのいずれかを指定できます。両方を指定することはできません。

elementN:Object (オプション) - パラメータ (element0, element1, ... , elementN)。任意のデータ型の値を指定できます。複数のデータ型を指定できるパラメータは、Object としてリストされます。numElements またはエレメントリストのいずれかを指定できます。両方を指定することはできません。

戻り値

Array - 配列。

例

```
var myArray:Array = Array();
myArray.push(12);
trace(myArray); //traces 12
myArray[4] = 7;
trace(myArray); //traces 12,undefined,undefined,undefined,7
```

使用法 2 - 次の例では、エレメントを定義せずに、長さ 4 の配列を作成します。

```
var myArray:Array = Array(4);
trace(myArray.length); // 4
trace(myArray); // undefined,undefined,undefined,undefined
```

使用法 3 - 次の例では、3 つのエレメントが定義された配列を作成します。

```
var myArray:Array = Array("firstElement", "secondElement", "thirdElement");
trace(myArray); // firstElement,secondElement,thirdElement
Unlike the Array class constructor, the Array() function does not use the keyword
  new .
```

関連項目

Array

asfunction プロトコル

```
asfunction:function:Function, parameter:String
```

HTML テキストフィールドの URL 専用プロトコルであり、HREF リンクから ActionScript 関数を呼び出すことができます。HTML テキストフィールド内で、HTML の A タグを使用してリンクを作成できます。A タグの HREF 属性に含まれる URL では通常、HTTP、HTTPS、FTP などの標準プロトコルを使用します。asfunction プロトコルは Flash 専用の追加プロトコルであり、リンクと連携して ActionScript 関数を呼び出します。

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

`function:String` - 関数の識別子。

`parameter:String` - *function* パラメータで指定された関数に渡されるストリング。

例

次の例では、playMP3() 関数を定義します。TextField オブジェクト list_txt を作成し、HTML テキストをレンダリングできるように設定します。テキスト Track 1 と Track 2 は、テキストフィールド内のリンクです。ユーザーがいずれかのリンクをクリックすると playMP3() 関数が呼び出され、asfunction 呼び出しパラメータとして渡される MP3 が再生されます。

```
var myMP3:Sound = new Sound();
function playMP3(mp3:String) {
 myMP3.loadSound(mp3, true);
 myMP3.onLoad = function(success) {
 if (!success) {
 // code to handle errors here
 }
 };
}
this.createTextField("list_txt", this.getNextHighestDepth(), 0, 0, 200, 100);
list_txt.autoSize = true;
list_txt.html = true;
list_txt.multiline = true;
list_txt.htmlText = "<a href=\"asfunction:playMP3, track1.mp3\">Track 1</
  a><br>";
list_txt.htmlText += "<a href=\"asfunction:playMP3, track2.mp3\">Track 2</
  a><br>";
```

リンクをクリックすると、MP3 サウンドファイルが Flash Player に送られます。

関連項目

htmlText（TextField.htmlText プロパティ）

Boolean 関数

Boolean(expression:Object) : Boolean

次に示すように、パラメータ expression をブール値に変換して、値を返します。

- expression がブール値である場合、戻り値は expression です。
- expression が数値あれば、0 でない場合の戻り値は true となり、0 の場合は false となります。

expression がストリングである場合は、戻り値は次のようになります。

- Flash Player 6 以前用にパブリッシュされたファイルでは、ストリングはまず数値に変換されます。数値が 0 でない場合の値は true となり、数値が 0 の場合の戻り値は false となります。
- Flash Player 7 以降用にファイルをパブリッシュした場合の結果は、ストリングの長さがゼロより長ければ true、空のストリングであれば false になります。

expression がストリングである場合の結果は、ストリングの長さがゼロより長ければ true、空のストリングであれば false になります。

- expression が undefined または NaN (非数) であれば、戻り値は false です。
- expression がムービークリップまたはオブジェクトであれば、戻り値は true です。

Boolean クラスのコンストラクタとは異なり、Boolean( ) 関数はキーワード new を使用しません。さらに、Boolean クラスのコンストラクタは、パラメータが指定されない場合に Boolean オブジェクトを false に初期化しますが、Boolean( ) 関数は、パラメータが指定されない場合に undefined を返します。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 7 ではビヘイビアが変更されました。

パラメータ

expression: Object - ブール値に変換される式。

戻り値

Boolean - ブール値。

例

```
trace(Boolean(-1)); // true
trace(Boolean(0)); // false
trace(Boolean(1)); // true


trace(Boolean(true)); // true
trace(Boolean(false)); // false


trace(Boolean("true")); // true
trace(Boolean("false")); // true
```

```
trace(Boolean("Craiggers")); // true
trace(Boolean("")); // false
```

ファイルを Flash Player 6 以前用にパブリッシュした場合、上記の 3 つの例は次のように異なる結果となります。

```
trace(Boolean("true")); // false
trace(Boolean("false")); // false
trace(Boolean("Craiggers")); // false
```

この例では、Boolean() 関数と Boolean クラスの使い方の大きな違いを示します。Boolean() 関数はブール値を作成し、Boolean クラスは Boolean オブジェクトを作成します。ブール値は値で比較され、Boolean オブジェクトは参照で比較されます。

```
// Variables representing Boolean values are compared by value
var a:Boolean = Boolean("a"); // a is true
var b:Boolean = Boolean(1); // b is true
trace(a==b); // true

// Variables representing Boolean objects are compared by reference
var a:Boolean = new Boolean("a"); // a is true
var b:Boolean = new Boolean(1); // b is true
trace(a == b); // false
```

関連項目

Boolean

call 関数

```
call(frame)
```

非推奨 Flash Player 5 以降では使用しないでください。このアクションの代わりに function ステートメントを使用します。

呼び出されたフレームで、そのフレームに再生ヘッドを移動せずに、スクリプトを実行します。スクリプトの実行後にローカル変数は存在しません。

- ブロック内 ({ }) で変数を宣言しないで、アクションリストを call() アクションから実行した場合、変数はローカルであり、現在のリストの終わりで終了します。
- ブロック内で変数を宣言しないで、現在のアクションリストを call() アクションから実行しない場合、変数はタイムライン変数として解釈されます。

対応バージョン：ActionScript 1.0、Flash Player 4

パラメータ

frame:Object - タイムラインのフレームのラベルまたは番号。

関連項目

function ステートメント , call（Function.call メソッド）

chr 関数

```
chr(number:Number) : String
```

非推奨 Flash Player 5 以降では使用しないでください。この関数の代わりに String.fromCharCode() を使用します。

ASCII コード番号を文字に変換します。

対応バージョン： ActionScript 1.0、Flash Player 4

パラメータ

number:Number - ASCII コード番号。

戻り値

String - 指定した ASCII コードの文字値。

例

次の例では、数値 65 を文字 A に変換し、それを変数 myVar: myVar = chr(65); に代入します。

関連項目

fromCharCode（String.fromCharCode メソッド）

clearInterval 関数

```
clearInterval(intervalID:Number) : Void
```

setInterval() の呼び出しを停止します。

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

intervalID:Number - setInterval() 呼び出しから返される数値 (整数) の識別子。

例

次の例では、最初に間隔への呼び出しを設定し、次にその呼び出しをクリアします。

```
function callback() {
 trace("interval called: "+getTimer()+" ms.");
}

var intervalID:Number = setInterval(callback, 1000);
```

関数の使用を終了したときは間隔をクリアする必要があります。clearInt_btn という名前のボタンを作成し、次の ActionScript を使用して setInterval() をクリアします。

```
clearInt_btn.onRelease = function(){
 clearInterval( intervalID );
 trace("cleared interval");
};
```

関連項目

setInterval 関数

clearTimeout 関数

```
clearTimeout(id:Number) : Void
```

指定した setTimeout() 呼び出しをキャンセルします。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

id:Number - キャンセル対象の setTimeout() 呼び出しの識別番号。

例

さらに、setTimeout() 関数の例も参照してください。

関連項目

setTimeout 関数, setInterval 関数

duplicateMovieClip 関数

```
duplicateMovieClip(target:String, newname:String, depth:Number) : Void
duplicateMovieClip(target:MovieClip, newname:String, depth:Number) : Void
```

SWF ファイルの再生中にムービークリップのインスタンスを作成します。複製ムービークリップの再生ヘッドは、元のムービークリップでの再生ヘッドの位置に関係なく、常にフレーム 1 から始まります。元のムービークリップ内の変数は、複製されたムービークリップにコピーされません。duplicateMovieClip() で作成されたムービークリップインスタンスを削除するには、removeMovieClip() 関数またはメソッドを使用します。

対応バージョン： ActionScript 1.0、Flash Player 4

パラメータ

target:Object - 複製するムービークリップのターゲットパス。このパラメータには、String ("my_mc" など)、またはムービークリップインスタンス (my_mc など) への直接参照のいずれかを指定できます。複数のデータ型を指定できるパラメータは、Object としてリストされます。

newname:String - 複製したムービークリップのインスタンス名。

depth:Number - 複製したムービークリップ固有の深度。深度は、複製したムービークリップの重ね順です。この重ね順は、タイムラインの重ね順に似ています。深度の低いムービークリップは、より高い重ね順のクリップの下に隠されます。既に占有されている深度の SWF ファイルが置き換えられるのを避けるため、複製したムービークリップには、それぞれ固有の深度を割り当てます。

例

次の例では、新しいムービークリップインスタンス img_mc を作成します。イメージをムービークリップにロードした後、img_mc クリップを複製します。複製されたクリップの名前は newImg_mc です。この新しいクリップを元のクリップと重ならないようにステージに移動し、同じイメージを 2 番目のクリップにロードします。

```
this.createEmptyMovieClip("img_mc", this.getNextHighestDepth());
img_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
duplicateMovieClip(img_mc, "newImg_mc", this.getNextHighestDepth());
newImg_mc._x = 200;
newImg_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
```

複製されたムービークリップを削除するには、ボタン myButton_btn に次のコードを追加します。

```
this.myButton_btn.onRelease = function(){
 removeMovieClip(newImg_mc);
};
```

関連項目

removeMovieClip 関数,duplicateMovieClip (MovieClip.duplicateMovieClip メソッド),removeMovieClip (MovieClip.removeMovieClip メソッド)

escape 関数

```
escape(expression:String) : String
```

パラメータをストリングに変換し、URL エンコードします。この場合、英数字以外のすべての文字は % が付いた 16 進シーケンスで置き換えられます。URL エンコードされたストリング内のパーセント記号 (%) は、エスケープ文字の開始を表すもので、剰余演算子 (%) ではありません。

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

`expression:String` - ストリングに変換し、URL エンコードする対象の式。

戻り値

`String` - URL エンコードされたストリング。

例

次のコードを作成すると、"someuser%40somedomain%2Ecom" が生成されます。

```
var email:String = "someuser@somedomain.com";
trace(escape(email));
```

この例では、アットマーク (@) が %40 に、ピリオド (.) が %2E に置き換えられます。この関数は、次のコードに示すように、リモートサーバーに送信する情報の中に特殊文字 (たとえば & や ?) が含まれている場合に便利です。

```
var redirectUrl:String = "http://www.somedomain.com?loggedin=true&username=Gus";
getURL("http://www.myothersite.com?returnurl="+ escape(redirectUrl));
```

関連項目

`unescape` 関数

eval 関数

```
eval(expression:Object) : Object
eval(expression:String) : Object
```

変数、プロパティ、オブジェクト、ムービークリップに名前でアクセスします。expression が変数またはプロパティである場合は、変数またはプロパティの値が返されます。expression がオブジェクトまたはムービークリップである場合は、オブジェクトまたはムービークリップへの参照が返されます。expression に指定したエレメントが見つからない場合は、undefined が返されます。

Flash 4 では、配列をシミュレートするために `eval()` 関数を使用していましたが、Flash 5 以降では Array クラスを使用する必要があります。

Flash 4 では、eval() を使って変数の値またはインスタンスの名前を動的に設定および取得することもできます。ただし、同じことは配列アクセス演算子 ([]) を使用しても可能です。

Flash 5 以降では、eval() を使用して変数の値またはインスタンス名を動的に設定および取得することはできません。これは、等式の左辺に eval() を使用できないためです。たとえば、次のようなコードは、

```
eval ("var" + i) = "first";
```

次のコードに置き換える必要があります。

```
this["var"+i] = "first"
```

または、次のようにもできます。

```
set ("var" + i, "first");
```

対応バージョン： ActionScript 1.0、Flash Player 5 - すべての機能を使用するには Flash Player 5 以降が必要です。Flash Player 4 に書き出す場合は eval() 関数を使用できますが、スラッシュ表記を使用する必要があります。また、変数だけにアクセス可能で、プロパティやオブジェクトにはアクセスできません。

パラメータ

expression:Object - 取得する変数、プロパティ、オブジェクト、またはムービークリップの名前。このパラメータには、String またはオブジェクトインスタンスへの直接参照のいずれかを指定できます (引用符 (" ") は使用しなくてもかまいません)。

戻り値

Object - オブジェクトまたはムービークリップへの参照、undefined、または値。

例

次の例では、eval() を使用して、動的に指定されるムービークリップのプロパティを設定します。この ActionScript は、square1_mc、square2_mc、square3_mc という 3 つのムービークリップの _rotation プロパティを設定します。

```
for (var i = 1; i <= 3; i++) {
 setProperty(eval("square"+i+"_mc"), _rotation, 5);
}
```

次の ActionScript を使用することもできます。

```
for (var i = 1; i <= 3; i++) {
 this["square"+i+"_mc"]._rotation = -5;
}
```

関連項目

Array, set variable ステートメント

fscommand 関数

```
fscommand(command:String, parameters:String) : Void
```

SWF ファイルが、Flash Player または Flash Player をホスティングするプログラム (Web ブラウザなど) と通信できるようになります。fscommand( ) 関数を使用して、Macromedia Director、または ActiveX コントロールに対応している Visual Basic (VB) や Visual C++ などのプログラムにメッセージを渡すこともできます。

fscommand( ) 関数を使用すると、SWF ファイルが Web ページ内のスクリプトと通信できるようになります。ただし、スクリプトへのアクセスは、Web ページの allowScriptAccess 属性の設定により制御されます。この属性は、SWF ファイルが埋め込まれる HTML コード内で設定します。たとえば、Internet Explorer の PARAM タグ内や、Netscape の EMBED タグ内で設定します。allowScriptAccess が "never" に設定されていると、SWF ファイルは Web ページスクリプトにアクセスできません。Flash Player 7 以降では、allowScriptAccess が "always" に設定されていると、SWF ファイルは Web ページスクリプトに常にアクセスできます。allowScriptAccess が "sameDomain" に設定されていると、Web ページと同じドメインに存在する SWF ファイルからのスクリプト実行のみが許可されます。以前のバージョンの Flash Player では、スクリプトの実行が常に許可されていました。HTML ページで allowScriptAccess が指定されていない場合、この属性のデフォルト値は、バージョン 8 以降の SWF ファイルでは "sameDomain" になり、バージョン 7 以前の SWF ファイルでは "always" になります。

シンタックス 1: fscommand( ) を使用して Flash Player にメッセージを送るには、定義済みのコマンドとパラメータを使用します。次の表に、fscommand( ) 関数の command パラメータと parameters パラメータに指定できる値を示します。これらの値は、プロジェクタを含め、Flash Player で再生する SWF ファイルを制御します。プロジェクタは、Flash Player がなくてもスタンドアローンアプリケーションとして実行可能な形式で保存される SWF ファイルです。

コマンド	パラメータ	用途
quit	なし	プロジェクタを終了します。
fullscreen	true または false	true を指定すると、Flash Player はフルスクリーンモードに設定されます。false を指定すると、プレーヤーはノーマルメニュービューになります。
allowscale	true または false	false を指定すると、SWF ファイルは常に元のサイズで表示され、拡大・縮小されないようにプレーヤーが設定されます。true を指定すると、SWF ファイルは強制的にプレーヤーのウィンドウサイズに対して 100 %に拡大・縮小されます。
showmenu	true または false	true を指定すると、すべてのコンテキストメニューアイテムが有効になります。false を指定すると、[設定] および [Flash Player について] 以外のすべてのコンテキストメニュー項目が非表示になります。

コマンド	パラメータ	用途
exec	アプリケーションへのパス	プロジェクタの内部からアプリケーションを実行します。
trapallkeys	true または false	true を指定すると、アクセラレータキーを含むすべてのキーイベントが Flash Player の onClipEvent(keyDown/keyUp) ハンドラに送られます。

利用可能なコマンド :

- Web プレーヤーでは、上記の表で示したコマンドはいずれも利用できません。
- プロジェクタなどのスタンドアローンアプリケーションでは、上記のすべてのコマンドを利用できます。
- ムービープレビュープレーヤでは、allowscale および exec のみを利用できます。

exec コマンドで使用できる文字は、A ～ Z、a ～ z、0 ～ 9、ピリオド (.)、アンダースコア (_) だけです。exec コマンドは、fscommand サブディレクトリでのみ実行されます。つまり、exec コマンドを使ってアプリケーションを呼び出す場合、アプリケーションは fscommand という名前のサブディレクトリ内に存在する必要があります。exec コマンドは、Flash プロジェクタファイル内だけから実行できます。

シンタックス 2 : fscommand( ) を使用して、Web ブラウザ内の JavaScript などのスクリプト言語にメッセージを送るには、パラメータ command および parameters に任意の 2 つのパラメータを渡すことができます。これらのパラメータはストリングと式のいずれでもよく、fscommand( ) 関数をキャッチする、つまり処理する JavaScript 関数の側で使用されます。

Web ブラウザでは、fscommand( ) は、SWF ファイルを含む Web ページ内に存在する JavaScript 関数 moviename_DoFScommand を呼び出します。moviename には、EMBED タグの NAME 属性または OBJECT タグの ID プロパティに使用した Flash オブジェクトの名前を指定します。SWF ファイルに myMovie という名前を付けた場合、JavaScript 関数 myMovie_DoFScommand が呼び出されます。

SWF ファイルを含む Web ページでは、allowScriptAccess 属性を設定することで、SWF ファイルによるその Web ページへのアクセスを許可または拒否します。この属性は、SWF ファイルが埋め込まれる HTML コード内で設定します。たとえば、Internet Explorer の PARAM タグ内や、Netscape の EMBED タグ内で設定します。allowScriptAccess が "never" に設定されている場合、送信スクリプトは常に失敗します。allowScriptAccess が "always" に設定されている場合、送信スクリプトは常に成功します。"sameDomain" に設定されている場合は、Web ページと同じドメインに存在する SWF ファイルからのスクリプト実行のみが許可されます。allowScriptAccess が Web ページで指定されていない場合、Flash Player 8 では "sameDomain" がデフォルト値になり、それより前のバージョンでは "always" がデフォルト値になります。

この関数を使用するときは、Flash Player セキュリティモデルを考慮してください。Flash Player 8 では、fscommand() 関数は、呼び出し元 SWF ファイルがローカルファイルシステムのサンドボックスまたはネットワーク接続したローカルのサンドボックスに置かれ、さらに、含まれている HTML ページが信頼されないページとしてサンドボックスに置かれている場合には、使用できません。詳細については、以下を参照してください。

- 『ActionScript 2.0 の学習』の第 17 章「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

シンタックス 3 : fscommand() 関数は、メッセージを Macromedia Director に送ることができます。これらのメッセージは、Lingo (Director のスクリプト言語) により、ストリング、イベント、実行可能 Lingo コードのいずれかと解釈されます。メッセージがストリングまたはイベントである場合、fscommand() 関数からメッセージを受信するための Lingo コードを書き、Director 内でアクションを実行する必要があります。詳細については、www.adobe.com/support/director の Director サポートセンターを参照してください。

シンタックス 4 : ActiveX コントロールに対応した VisualBasic や Visual C++ などのプログラムでは、fscommand() を使って、その環境のプログラミング言語で処理できる 2 つのストリングを含む VB イベントを送信できます。詳細については、www.adobe.com/support/flash/ の Flash サポートセンターで "Flash メソッド " というキーワードを使用して検索してください。

注意 : Flash Player 8 以降用にパブリッシュする場合、ExternalInterface クラスを使用すると、JavaScript と ActionScript (シンタックス 2) との間、または ActionScript と、VisualBasic や Visual C++ などの ActiveX コントロールに対応した他のプログラム (シンタックス 4) との間の通信機能が向上します。fscommand() は、メッセージを Flash Player に送信するため (シンタックス 1) および Macromedia Director に送信するために (シンタックス 3) 引き続き使用する必要があります。

対応バージョン : ActionScript 1.0、Flash Player 3

パラメータ

`command:String` - ホストアプリケーションに任意の用途で渡されるストリング、または Flash Player に渡されるコマンド。

`parameters:String` - ホストアプリケーションに任意の用途で渡されるストリング、または Flash Player に渡される値。

例

次の例に示す fscommand() は、fullscreen_btn または unfullscreen_btn ボタンを離したときに SWF ファイルをフルスクリーンに拡大するように Flash Player を設定します。

```
this.fullscreen_btn.onRelease = function() {
 fscommand("fullscreen", true);
};

this.unfullscreen_btn.onRelease = function() {
 fscommand("fullscreen", false);
};
```

次の例では、JavaScript のメッセージボックスを HTML ページに表示するために、Flash 内のボタンに fscommand() を設定しています。メッセージ自体は、fscommand パラメータとして JavaScript に送られます。

SWF ファイルを含む Web ページに関数を追加します。この、myDocument _DoFSCommand() 関数は、fscommand() の呼び出しを待ちます。Flash で fscommand() がトリガされると (ユーザーがボタンを押すなど)、command と parameter がストリングとして myDocument _DoFSCommand() 関数に渡されます。渡されたストリングは、JavaScript または VBScript のコードで任意の用途に使用できます。次の例では、関数内にある if ステートメントで、コマンドストリングが "messagebox" であるかどうかを確認します。そうである場合は、JavaScript アラートボックスに fscommand() 関数の parameters ストリングの内容が表示されます。

```
function myDocument_DoFSCommand(command, args) {
 if (command == "messagebox") {
 alert(args);
 }
}
```

Flash ドキュメントで、ボタンに fscommand() を追加します。

```
fscommand("messagebox", "This is a message box called from within Flash.")
```

次の例に示すように、fscommand() 関数のパラメータでは式を使用することもできます。

```
fscommand("messagebox", "Hello, " + name + ", welcome to our website!")
```

SWF ファイルをテストするには、[ファイル]-[パブリッシュプレビュー]-[HTML] を選択します。Flash を使用する SWF ファイルを、[パブリッシュ設定] ダイアログボックスの [HTML] タグを選択して表示される FSCommand テンプレートを使ってパブリッシュすると、myDocument_DoFSCommand() 関数が自動的に挿入されます。SWF ファイルの NAME 属性と ID 属性には、ファイル名が使用されます。たとえば、myDocument.fla というファイルの場合は、属性値として myDocument が設定されます。

関連項目

ExternalInterface (flash.external.ExternalInterface)

getProperty 関数

```
getProperty(my_mc:Object, property:Object) : Object
```

ムービークリップ *my_mc* の指定プロパティの値を返します。

対応バージョン： ActionScript 1.0、Flash Player 4

パラメータ

my_mc:Object - String オブジェクト (プロパティ取得対象のムービークリップのインスタンス名)、または MovieClip オブジェクト (プロパティ取得対象のムービークリップの参照) のいずれか。

property - ムービークリップのプロパティ。

戻り値

Object - 指定したプロパティの値。

例

次の例では、新しいムービークリップ someClip_mc を作成し、ムービークリップ someClip_mc のアルファ値 (_alpha) を [出力] パネルに表示します。

```
this.createEmptyMovieClip("someClip_mc", 999);
trace("The alpha of "+getProperty(someClip_mc, _name)+" is:
   "+getProperty(someClip_mc, _alpha));
```

getTimer 関数

```
getTimer() : Number
```

SWF ファイルを再生し始めてからの経過時間をミリ秒単位で返します。

対応バージョン： ActionScript 1.0、Flash Player 4

戻り値

Number - SWF ファイルの再生を開始してから経過したミリ秒数。

例

次の例では、getTimer() 関数と setInterval() 関数を使用して、単純なタイマーを作成します。

```
this.createTextField("timer_txt", this.getNextHighestDepth(), 0, 0, 100, 22);
function updateTimer():Void {
 timer_txt.text = getTimer();
}

var intervalID:Number = setInterval(updateTimer, 100);
```

getURL 関数

```
getURL(url:String, [window:String, [method:String]]) : Void
```

特定の URL からウィンドウにドキュメントをロードしたり、定義済みの URL に存在する別のアプリケーションに変数を渡したりします。この関数をテストするには、ロードするファイルが指定した場所にあることを確認します。絶対 URL (*http://www.myserver.com* など) を使用するには、ネットワーク接続が確立されている必要があります。

セキュリティ上の注意 : Flash Player 8 以降では、ブラウザ内で実行されるローカルコンテンツの場合、"javascript:" 疑似プロトコルを指定する getURL() 呼び出し (たとえば getURL("javascript:someFunction()")) は、ローカルに信頼されたセキュリティサンドボックス内に SWF ファイルおよびそれを含む Web ページ (存在する場合) がある場合にのみ可能です。

セキュリティ上の注意 : Flash Player 9 以降では、ローカルファイルシステムのサンドボックスで実行される SWF ファイル内のコードが getURL() 関数を呼び出し、window パラメータにカスタムウィンドウ名を指定する場合、ウィンドウ名はランダムな名前に変換されます。名前の形式は "_flashXXXXXXXX" で、それぞれの X はランダムな 16 進数です。それを含むブラウザウィンドウを閉じるまでの同じセッション内では、関数を再び呼び出して window パラメータに同じ名前を指定した場合、同じランダムストリングが使用されます。

対応バージョン : ActionScript 1.0、Flash Player 2 - GET オプションと POST オプションは、Flash Player 4 以降のバージョンでのみ使用できます。

パラメータ

url:String - ドキュメントを取得するための URL。

window:String (オプション) - ドキュメントのロード先のウィンドウまたは HTML フレームを指定します。特定のウィンドウの名前を入力するか、次の予約されたターゲット名から選択します。

- _self は、現在のウィンドウ内の現在のフレームを指定します。
- _blank は、新規ウィンドウを指定します。
- _parent は、現在のフレームの親を指定します。
- _top 現在のウィンドウ内の最上位のフレームを指定します。

method:String (オプション) - 変数を送るための GET メソッドまたは POST メソッド。変数がない場合は、このパラメータを省略します。GET メソッドは、変数を URL の最後に追加します。このメソッドは、変数のデータ量が少ないときに使用します。POST メソッドは、別の HTTP ヘッダで変数を送信します。このメソッドは、変数のデータ量が多いときに使用します。

例

次の例では、イメージをムービークリップにロードします。イメージがクリックされると、新しいブラウザウィンドウに新しい URL がロードされます。

```
var listenerObject:Object = new Object();
listenerObject.onLoadInit = function(target_mc:MovieClip) {
 target_mc.onRelease = function() {
 getURL("http://www.adobe.com/software/flash/flashpro/", "_blank");
 };
};
var logo:MovieClipLoader = new MovieClipLoader();
logo.addListener(listenerObject);
logo.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
 this.createEmptyMovieClip("adobe_mc", this.getNextHighestDepth()));
```

次の例では、getURL() を使用して、電子メールメッセージを送信します。

```
myBtn_btn.onRelease = function(){
 getURL("mailto:you@somedomain.com");
};
```

次の ActionScript では、SWF ファイルがブラウザウィンドウに埋め込まれているときに JavaScript を使用してアラートウィンドウを開きます。getURL() を使って JavaScript を呼び出す場合、url パラメータの長さは最大 508 文字に制限されます。

```
myBtn_btn.onRelease = function(){
 getURL("javascript:alert('you clicked me')");
};
```

GET や POST を使用して変数を送信することもできます。次の例では、GET を使用して、変数を URL に追加します。

```
var firstName:String = "Gus";
var lastName:String = "Richardson";
var age:Number = 92;
myBtn_btn.onRelease = function() {
 getURL("http://www.adobe.com", "_blank", "GET");
};
```

次の ActionScript では、POST を使用して、HTTP ヘッダ内で変数を送信します。ブラウザウィンドウでドキュメントをテストしてください。そうでないと、変数は GET を使って送信されます。

```
var firstName:String = "Gus";
var lastName:String = "Richardson";
var age:Number = 92;
getURL("http://www.adobe.com", "_blank", "POST");
```

関連項目

loadVariables 関数,send (XML.send メソッド),sendAndLoad (XML.sendAndLoad メソッド)

getVersion 関数

```
getVersion() : String
```

Flash Player のバージョンとプラットフォーム情報を含むストリングを返します。getVersion 関数は、Flash Player 5 以降のバージョン情報のみを返します。

対応バージョン： ActionScript 1.0、Flash Player 5

戻り値

String - Flash Player のバージョンとプラットフォーム情報を含むストリング。

例

次の例では、SWF ファイルを再生している Flash Player のバージョン番号をトレースします。

```
var flashVersion:String = getVersion();
trace(flashVersion); // WIN 8,0,1,0
trace($version); // WIN 8,0,1,0
trace(System.capabilities.version); // WIN 8,0,1,0
```

次のストリングが getVersion 関数から返されます。

```
WIN 8,0,1,0
```

この返されたストリングは、プラットフォームが Microsoft Windows であり、Flash Player がメジャーバージョン 8、マイナーバージョン 1 (8.1) であることを示します。

関連項目

os (capabilities.os プロパティ), version (capabilities.version プロパティ)

gotoAndPlay 関数

```
gotoAndPlay([scene:String], frame:Object) : Void
```

シーン内の指定されたフレームに再生ヘッドを送り、そのフレームから再生します。シーンを指定しないと、再生ヘッドは現在のシーン内の指定されたフレームに進みます。*scene* パラメータはルートタイムラインでのみ使用でき、ムービークリップやドキュメント内のその他のオブジェクトのタイムラインでは使用できません。

対応バージョン： ActionScript 1.0、Flash Player 2

パラメータ

scene:String (オプション) - 再生ヘッドの送り先となるシーンの名前を示すストリング。

frame:Object - 再生ヘッドの送り先となるフレーム番号を表す数値、または再生ヘッドの送り先となるフレームのラベルを表すストリング。

例

次の例では、2 つのシーン sceneOne と sceneTwo がドキュメントに含まれています。シーン 1 には、フレーム 10 の newFrame というフレームラベルと、2 つのボタン myBtn_btn および myOtherBtn_btn があります。次の ActionScript をメインタイムラインのシーン 1 のフレーム 1 に配置します。

```
stop();
myBtn_btn.onRelease = function(){
 gotoAndPlay("newFrame");
};

myOtherBtn_btn.onRelease = function(){
 gotoAndPlay("sceneTwo", 1);
};
```

ユーザーがボタンをクリックすると、指定された場所に再生ヘッドが移動した後、再生が続行されます。

関連項目

gotoAndPlay (MovieClip.gotoAndPlay メソッド), nextFrame 関数, play 関数, prevFrame 関数

gotoAndStop 関数

```
gotoAndStop([scene:String], frame:Object) : Void
```

シーン内の指定されたフレームに再生ヘッドを送り、停止します。再生ヘッドは、シーンが指定されていない場合には現在のシーン内のフレームに送られます。*scene* パラメータはルートタイムラインでのみ使用でき、ムービークリップやドキュメント内のその他のオブジェクトのタイムラインでは使用できません。

対応バージョン : ActionScript 1.0、Flash Player 2

パラメータ

scene:String (オプション) - 再生ヘッドの送り先となるシーンの名前を示すストリング。

frame:Object - 再生ヘッドの送り先となるフレーム番号を表す数値、または再生ヘッドの送り先となるフレームのラベルを表すストリング。

例

次の例では、2 つのシーン sceneOne と sceneTwo がドキュメントに含まれています。シーン 1 には、フレーム 10 の newFrame というフレームラベルと、2 つのボタン myBtn_btn および myOtherBtn_btn があります。次の ActionScript をメインタイムラインのシーン 1 のフレーム 1 に配置します。

```
stop();

myBtn_btn.onRelease = function(){
 gotoAndStop("newFrame");
};

myOtherBtn_btn.onRelease = function(){
 gotoAndStop("sceneTwo", 1);
};
```

ユーザーがボタンをクリックすると、指定された場所に再生ヘッドが移動した後、停止します。

関連項目

gotoAndStop (MovieClip.gotoAndStop メソッド), stop 関数, play 関数, gotoAndPlay 関数

ifFrameLoaded 関数

```
ifFrameLoaded([scene:String], frame) {
statement(s);
}
```

非推奨 Flash Player 5 以降では使用しないでください。この関数は使用されなくなりました。MovieClip._framesloaded プロパティを使用することをお勧めします。

特定のフレームの内容がローカルに使用できるかどうかを確認します。SWF ファイル全体がローカルコンピュータにダウンロードされるのを待つ間に単純なアニメーションの再生を開始する場合は、ifFrameLoaded を使用します。_framesloaded が ifFrameLoaded と異なる点は、_framesloaded では、独自の if ステートメントや else ステートメントを追加できることです。

対応バージョン： ActionScript 1.0、Flash Player 4

パラメータ

scene:String (オプション) - ロードする必要のあるシーンの名前を示すストリング。

frame:Object - 次のステートメントを実行する前にロードする必要のあるフレーム番号またはフレームラベル。

関連項目

addListener (MovieClipLoader.addListener メソッド)

int 関数

```
int(value:Number) : Number
```

非推奨 Flash Player 5 以降では使用しないでください。この関数の代わりに `Math.floor()`(正の値の場合) および `Math.ceil`(負の値の場合) を使用します。

小数値を切り捨てることによって、10 進数の整数部を取り出します。したがって、負数の扱いは Math.floor() メソッドと異なります。この関数は `Math.floor()` (*value* パラメータが正の場合) および `Math.ceil()` (*value* パラメータが負の場合) と同じです。

対応バージョン : ActionScript 1.0、Flash Player 4

パラメータ

`value:Number` - 整数部を取り出す数値。

戻り値

`Number` - 取り出された整数値。

関連項目

`round (Math.round メソッド )`, `floor (Math.floor メソッド )`, `ceil (Math.ceil メソッド )`

isFinite 関数

```
isFinite(expression:Object) : Boolean
```

expression を評価し、有限大である場合は `true` を、無限大または負の無限大である場合は `false` を返します。無限大または負の無限大は、0 による除算などの数学的なエラーの可能性を示します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`expression:Object` - 評価するブール値、変数、または式。

戻り値

`Boolean` - ブール値。

例

`isFinite` の戻り値の例を次に示します。

```
isFinite(56)
// returns true
```

```
isFinite(Number.POSITIVE_INFINITY)
//returns false
```

isNaN 関数

```
isNaN(expression:Object) : Boolean
```

パラメータを評価し、値が NaN (非数) である場合は true を返します。この関数は、数式が正常に評価されて数値になるかどうかをチェックする場合に便利です。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

expression:Object - 評価されるブール値、変数、または式。

戻り値

Boolean - ブール値。

例

次のコードは、isNaN( ) 関数の戻り値を示します。

```
trace( isNaN("Tree") );
// returns true

trace( isNaN(56) );
// returns false

trace( isNaN(Number.POSITIVE_INFINITY) )
// returns false
```

次の例では、isNAN( ) を使用して、数式にエラーが含まれるかどうかをチェックする方法を示しています。

```
var dividend:Number;
var divisor:Number;
divisor = 1;
trace( isNaN(dividend/divisor) );
// output: true
// The output is true because the variable dividend is undefined.
// Do not use isNAN() to check for division by 0 because it will return false.
// A positive number divided by 0 equals Infinity (Number.POSITIVE_INFINITY).
// A negative number divided by 0 equals -Infinity (Number.NEGATIVE_INFINITY).
```

関連項目

NaN 定数, NaN (Number.NaN プロパティ)

length 関数

```
length(expression:String)length(variable)
```

非推奨 Flash Player 5 以降では使用しないでください。この関数およびすべてのストリング関数は使用されなくなりました。String クラスのメソッドと `String.length` プロパティを使用して同じ処理を行うことをお勧めします。

指定されたストリングまたは変数の長さを返します。

対応バージョン： ActionScript 1.0、Flash Player 4

パラメータ

`expression:String` - ストリング。

`variable:Object` - 変数の名前。

戻り値

`Number` - 指定したストリングまたは変数の長さ。

例

次の例では、ストリング "Hello" の長さを返します。結果は 5 になります。`length("Hello");`

関連項目

" ストリング区切り記号演算子 , `String`, `length （String.length プロパティ ）`

loadMovie 関数

```
loadMovie(url:String, target:Object, [method:String]) : Void
loadMovie(url:String, target:String, [method:String]) : Void
```

元の SWF ファイルの再生中に SWF、JPEG、GIF、または PNG の各ファイルを Flash Player のムービークリップ内にロードします。非アニメーション GIF ファイル、PNG ファイル、およびプログレッシブ JPEG ファイルのサポートが Flash Player 8 で追加されました。アニメーション GIF を読み込むと、先頭のフレームのみ表示されます。非プログレッシブ JPEG ファイルは、Flash Player 6 以降でサポートされます。

ヒント : ダウンロードの進捗状況を監視するには、この関数の代わりに `MovieClipLoader.loadClip()` を使用してください。

`loadMovie()` 関数を使用すると、複数の SWF ファイルを同時に表示し、別の HTML ドキュメントをロードせずに SWF ファイルを切り替えることができます。`loadMovie()` 関数を使用しない場合、Flash Player は 1 つの SWF ファイルを表示します。

SWF ファイルまたは JPEG ファイルを特定のレベルにロードするには、`loadMovie()` の代わりに `loadMovieNum()` を使用します。

SWF ファイルをターゲットムービークリップにロードすると、そのムービークリップのターゲットパスを使用して、ロードした SWF ファイルをターゲットとすることができます。ターゲットにロードした SWF ファイルまたはイメージは、ターゲットムービークリップの位置、回転、および拡大・縮小プロパティを継承します。ロードしたイメージまたは SWF ファイルの左上隅は、ターゲットムービークリップの基準点に位置合わせされます。ターゲットがルートタイムラインである場合、イメージまたは SWF ファイルの左上隅はステージの左上隅に位置合わせされます。

`loadMovie()` を使ってロードした SWF ファイルを削除するには、`unloadMovie()` を使用します。

この関数を使用するときは、Flash Player セキュリティモデルを考慮してください。

Flash Player 8 :

- 呼び出し元のムービークリップがローカルファイルシステムのサンドボックスにあり、ロードするムービークリップがネットワーク上のサンドボックスにある場合、ロードできません。
- 呼び出し元 SWF ファイルがネットワーク上のサンドボックスにあり、ロードするムービークリップがローカルにある場合、ロードできません。
- 信頼できるローカルのサンドボックスまたはネットワーク接続したローカルのサンドボックスからネットワーク上のサンドボックスにアクセスするには、クロスドメインポリシーファイルを使用して Web サイトで許可する必要があります。
- ローカルファイルシステムのサンドボックスにあるムービークリップでは、ネットワーク接続したローカルのサンドボックスにあるムービークリップをスクリプト処理できません。その逆も同様です。

Flash Player 7 以降 :

- Web サイトでクロスドメインポリシーファイルを使用して、リソースへのクロスドメインアクセスを許可できます。
- SWF ファイル間のスクリプト処理は、SWF ファイルが置かれているドメインに基づいて制限されます。これらの制限を調整するには、`System.security.allowDomain()` メソッドを使用します。

詳細については、次の参照先を参照してください。

- 『ActionScript 2.0 の学習 』の第 17 章「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

対応バージョン : ActionScript 1.0、Flash Player 3 - JPEG ファイルのロードは、Flash Player 6 で使用可能になった機能です。非アニメーション GIF ファイル、PNG ファイル、またはプログレッシブ JPEG ファイルのロードは、Flash Player 8 で使用可能になった機能です。

パラメータ

`url:String` - ロードする SWF ファイルまたは JPEG ファイルの絶対 URL または相対 URL。相対パスは、レベル 0 の SWF ファイルが埋め込まれた HTML ファイルを基準にする必要があります。絶対 URL の場合は http:// や file:// などのプロトコル参照を含めて指定します。

`target:Object` - ターゲットムービークリップへのパスを表すムービークリップオブジェクトまたはストリングへの参照。ターゲットムービークリップは、ロードした SWF ファイルまたはイメージに置き換えられます。

`method:String` (オプション) - 変数を送信するための HTTP メソッドを指定します。パラメータはストリング `GET` または `POST` でなければなりません。送る変数がない場合は、このパラメータを省略します。`GET` メソッドは、変数を URL の最後に追加します。このメソッドは、変数のデータ量が少ないときに使用します。`POST` メソッドは、別の HTTP ヘッダで変数を送信します。このメソッドは、変数のデータ量が多いときに使用します。

例

シンタックス 1: 次の例では、ステージに存在する `mySquare` というムービークリップを、同じディレクトリからロードした SWF ファイル circle.swf と置き換えます。

```
loadMovie("circle.swf", mySquare);
// equivalent statement (Usage 1): loadMovie("circle.swf", _level0.mySquare);
// equivalent statement (Usage 2): loadMovie("circle.swf", "mySquare");
```

次の例では、SWF ファイル circle.swf を同じディレクトリからロードしますが、`mySquare` ムービークリップではなくメインムービークリップを置き換えます。

```
loadMovie("circle.swf", this);
// Note that using "this" as a string for the target parameter will not work
// equivalent statement (Usage 2): loadMovie("circle.swf", "_level0");
```

次の `loadMovie()` ステートメントは、`createEmptyMovieClip()` を使って作成された新しいムービークリップ `logo_mc` に SWF ファイル sub.swf を同じディレクトリからロードします。

```
this.createEmptyMovieClip("logo_mc", 999);
loadMovie("sub.swf", logo_mc);
```

次のコードを追加すると、sub.swf をロードするのと同じディレクトリから JPEG イメージ image1.jpg を SWF ファイルにロードできます。JPEG イメージは、ボタン `myBtn_btn` をクリックするとロードされます。このコードは、JPEG イメージを `logo_mc` にロードします。したがって、sub.swf は JPEG イメージで置き換えられます。

```
myBtn_btn.onRelease = function(){
 loadMovie("image1.jpg", logo_mc);
};
```

シンタックス 2 : 次の例では、ステージに存在する mySquare というムービークリップを、同じディレクトリからロードした SWF ファイル circle.swf と置き換えます。

```
loadMovie("circle.swf", "mySquare");
```

関連項目

_level プロパティ , loadMovieNum 関数 , loadMovie (MovieClip.loadMovie メソッド), loadClip (MovieClipLoader.loadClip メソッド), unloadMovie 関数

loadMovieNum 関数

```
loadMovieNum(url:String, level:Number, [method:String]) : Void
```

元の SWF ファイルの再生中に SWF、JPEG、GIF、または PNG の各ファイルを任意のレベルにロードします。非アニメーション GIF ファイル、PNG ファイル、およびプログレッシブ JPEG ファイルのサポートが Flash Player 8 で追加されました。アニメーション GIF を読み込むと、先頭のフレームのみ表示されます。非プログレッシブ JPEG ファイルは、Flash Player 6 以降でサポートされます。

ヒント : ダウンロードの進捗状況を監視するには、この関数の代わりに MovieClipLoader.loadClip() を使用してください。

通常、Flash Player は 1 つの SWF ファイルを表示した後に閉じます。loadMovieNum() アクションを使用すると、複数の SWF ファイルを同時に表示し、別の HTML ドキュメントをロードせずに SWF ファイルを切り替えることができます。

レベルではなくターゲットを指定するには、loadMovieNum() の代わりに loadMovie() を使います。

Flash Player では、レベル 0 からレベルが積み重ねられます。これらのレベルは各レベルのオブジェクト以外は透明であり、フィルムのレイヤーに似ています。loadMovieNum() を使用する場合は、SWF ファイルをロードする先の Flash Player のレベルを指定します。SWF ファイルをレベル内にロードすると、シンタックス _level *N* を使用できます。*N* は SWF ファイルのターゲットとなるレベル番号です。

SWF ファイルをロードするときは、任意のレベル番号を指定できます。SWF ファイルが既にロードされているレベルに SWF ファイルをロードすることもできます。その場合は、新しい SWF ファイルによって既存の SWF ファイルが置き換えられます。SWF ファイルをレベル 0 内にロードすると、Flash Player の各レベルはアンロードされ、レベル 0 は新しいファイルに置き換えられます。レベル 0 の SWF ファイルによって、ロードした他のすべての SWF ファイルのフレームレート、背景色、およびフレームサイズが設定されます。

loadMovieNum() アクションを使用すると、JPEG ファイルを再生中の SWF にロードすることもできます。イメージと SWF ファイルのいずれの場合でも、ファイルのロード時にイメージの左上隅がステージの左上隅に位置合わせされます。また、どちらの場合でも、ロードしたファイルは回転と拡大・縮小を継承し、元の内容が指定されたレベルで上書きされます。

注意 : プログレッシブ形式で保存された JPEG ファイルはサポートされていません。

loadMovieNum() を使ってロードした SWF ファイルやイメージを削除するには、unloadMovieNum() を使用します。

このメソッドを使用するときは、Flash Player セキュリティモデルを考慮してください。

Flash Player 8 :

- 呼び出し元のムービークリップがローカルファイルシステムのサンドボックスにあり、ロードするムービークリップがネットワーク上のサンドボックスにある場合、ロードできません。
- 呼び出し元 SWF ファイルがネットワーク上のサンドボックスにあり、ロードするムービークリップがローカルにある場合、ロードできません。
- 信頼できるローカルのサンドボックスまたはネットワーク接続したローカルのサンドボックスからネットワーク上のサンドボックスにアクセスするには、クロスドメインポリシーファイルを使用して Web サイトで許可する必要があります。
- ローカルファイルシステムのサンドボックスにあるムービークリップでは、ネットワーク接続したローカルのサンドボックスにあるムービークリップをスクリプト処理できません。その逆も同様です。

Flash Player 7 以降 :

- Web サイトでクロスドメインポリシーファイルを使用して、リソースへのクロスドメインアクセスを許可できます。
- SWF ファイル間のスクリプト処理は、SWF ファイルが置かれているドメインに基づいて制限されます。これらの制限を調整するには、System.security.allowDomain() メソッドを使用します。

詳細については、次の参照先を参照してください。

- 『ActionScript 2.0 の学習 』の第 17 章「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

対応バージョン : ActionScript 1.0、Flash Player 4 - Flash 5 以降で開いた Flash 4 ファイルは、正しいシンタックスを使用するように変換されます。JPEG ファイルのロードは、Flash Player 6 で使用可能になった機能です。非アニメーション GIF ファイル、PNG ファイル、またはプログレッシブ JPEG ファイルのロードは、Flash Player 8 で使用可能になった機能です。

パラメータ

`url:String` - ロードする SWF ファイルまたは JPEG ファイルの絶対 URL または相対 URL。相対パスの場合は、レベル 0 の SWF ファイルを基準にする必要があります。スタンドアローンの Flash Player 内で使用する際、または Flash オーサリングアプリケーション内のプレビューモードでテストする際には、すべての SWF ファイルを同じフォルダに置く必要があり、ファイル名にフォルダやディスクドライブの指定を含めることはできません。

`level:Number` - SWF ファイルをロードする先の Flash Player のレベルを指定する整数。

`method:String` (オプション) - 変数を送信するための HTTP メソッドを指定します。パラメータはストリング `GET` または `POST` でなければなりません。送る変数がない場合は、このパラメータを省略します。`GET` メソッドは、変数を URL の最後に追加します。このメソッドは、変数のデータ量が少ないときに使用します。`POST` メソッドは、別の HTTP ヘッダで変数を送信します。このメソッドは、変数のデータ量が多いときに使用します。

例

次の例では、JPEG イメージ tim.jpg を Flash Player のレベル 2 内にロードします。

```
loadMovieNum("http://www.helpexamples.com/flash/images/image1.jpg", 2);
```

関連項目

unloadMovieNum 関数,loadMovie 関数,loadClip (MovieClipLoader.loadClip メソッド),_level プロパティ

loadVariables 関数

```
loadVariables(url:String, target:Object, [method:String]) : Void
```

テキストファイルや、ColdFusion、CGI スクリプト、ASP (Active Server Pages)、パーソナルホームページ (PHP)、または Perl スクリプトで作成されたテキストなどの外部ファイルからデータを読み取り、ターゲットムービークリップの変数に値を設定します。このアクションを使用して、現在の SWF ファイルの変数を新しい値で更新することもできます。

指定された URL にあるテキストは、標準の MIME 形式 *application/x-www-form-urlencoded* (CGI スクリプトで使用される標準形式) でなければなりません。変数はいくつでも指定できます。たとえば、次の例では複数の変数が定義されています。

```
company=Adobe&address=601+Townsend&city=San+Francisco&zip=94103
```

Flash Player 7 より前のバージョンの Player で SWF ファイルを実行している場合、*url* は、呼び出し元の SWF ファイルと同じスーパードメインに属している必要があります。スーパードメインは、ファイルの URL の左端の要素を削除することで求められます。たとえば、www.someDomain.com に存在する SWF ファイルは、store.someDomain.com のデータソースからデータをロードできます。どちらのファイルも同じスーパードメイン someDomain.com に属しているからです。

Flash Player 7 以降のバージョンで動作する SWF ファイルでは、*url* は呼び出し元の SWF ファイルと完全に同じドメインに属している必要があります (『ActionScript 2.0 の学習』の「Flash Player のセキュリティ機能」を参照)。たとえば www.someDomain.com に置かれている SWF ファイルは、www.someDomain.com に置かれているソースからのみデータをロードできます。異なるドメインからデータをロードするには、アクセス対象の SWF ファイルをホストするサーバーにクロスドメインポリシーファイルを配置できます。詳細については、『ActionScript 2.0 の学習』の「ドメイン間のデータロード許可について」を参照してください。

変数を特定のレベルにロードするには、`loadVariables()` の代わりに `loadVariablesNum()` を使用します。

対応バージョン : ActionScript 1.0、Flash Player 4 - Flash Player 7 ではビヘイビアが変更されました。

パラメータ

`url:String` - 変数が存在する絶対 URL または相対 URL。呼び出し元の SWF ファイルが Web ブラウザで実行されている場合、*url* は SWF ファイルと同じドメインに属している必要があります。詳細については、「説明」を参照してください。

`target:Object` - ロードした変数を受け取るムービークリップへのターゲットパス。

`method:String` (オプション) - 変数を送信するための HTTP メソッドを指定します。パラメータはストリング `GET` または `POST` でなければなりません。送る変数がない場合は、このパラメータを省略します。`GET` メソッドは、変数を URL の最後に追加します。このメソッドは、変数のデータ量が少ないときに使用します。`POST` メソッドは、別の HTTP ヘッダで変数を送信します。このメソッドは、変数のデータ量が多いときに使用します。

例

次の例では、テキストファイル "params.txt" の情報を、`createEmptyMovieClip()` を使用して作成されたムービークリップ `target_mc` にロードします。`setInterval()` 関数は、ロードの進捗状況をチェックする場合に使用します。このスクリプトは、params.txt ファイル内の `done` という名前の変数をチェックします。

```
this.createEmptyMovieClip("target_mc", this.getNextHighestDepth());
loadVariables("params.txt", target_mc);
function checkParamsLoaded() {
 if (target_mc.done == undefined) {
 trace("not yet.");
 } else {
 trace("finished loading. killing interval.");
 trace("-------------");
 for (i in target_mc) {
 trace(i+": "+target_mc[i]);
 }
```

```
 trace("-------------");
 clearInterval(param_interval);
 }
}
var param_interval:Number = setInterval(checkParamsLoaded, 100);
```

外部ファイル params.txt には、次のテキストが含まれています。

```
var1="hello"&var2="goodbye"&done="done"
```

関連項目

loadVariablesNum 関数,loadMovie 関数,loadMovieNum 関数,getURL 関数,loadMovie (MovieClip.loadMovie メソッド),loadVariables (MovieClip.loadVariables メソッド),load (LoadVars.load メソッド)

loadVariablesNum 関数

```
loadVariablesNum(url:String, level:Number, [method:String]) : Void
```

テキストファイルや、ColdFusion、CGI スクリプト、ASP、PHP、または Perl スクリプトで作成されたテキストなどの外部ファイルからデータを読み取り、Flash Player の特定のレベルの変数に値を設定します。この関数を使用して、現在の SWF ファイルの変数を新しい値で更新することもできます。

指定された URL にあるテキストは、標準の MIME 形式 *application/x-www-form-urlencoded* (CGI スクリプトで使用される標準形式) になっている必要があります。変数はいくつでも指定できます。たとえば、次の例では複数の変数が定義されています。

```
company=Adobe&address=601+Townsend&city=San+Francisco&zip=94103
```

Flash Player 7 より前のバージョンの Player で SWF ファイルを実行している場合、*url* は、呼び出し元の SWF ファイルと同じスーパードメインに属している必要があります。スーパードメインは、ファイルの URL の左端の要素を削除することで求められます。たとえば、www.someDomain.com に存在する SWF ファイルは、store.someDomain.com に存在するソースからデータをロードできます。これは、どちらのファイルも同じスーパードメイン someDomain.com に属しているからです。

Flash Player 7 以降のバージョンで動作する SWF ファイルでは、*url* は呼び出し元の SWF ファイルと完全に同じドメインに属している必要があります (『ActionScript 2.0 の学習』の「Flash Player のセキュリティ機能」を参照)。たとえば www.someDomain.com に置かれている SWF ファイルは、www.someDomain.com に置かれているソースからのみデータをロードできます。異なるドメインからデータをロードするには、SWF ファイルをホストするサーバーにクロスドメインポリシーファイルを配置できます。詳細については、『ActionScript 2.0 の学習』の「ドメイン間のデータロード許可について」を参照してください。

変数をターゲットムービークリップにロードするには、loadVariablesNum() の代わりに loadVariables() を使います。

対応バージョン： ActionScript 1.0、Flash Player 4 - Flash Player 7 ではビヘイビアが変更されました。Flash 5 以降で開いた Flash 4 ファイルは、正しいシンタックスを使用するように変換されます。

パラメータ

`url:String` - 変数が存在する絶対 URL または相対 URL。呼び出し元の SWF ファイルが Web ブラウザで実行されている場合、*url* は SWF ファイルと同じドメインに属している必要があります。詳細については、「説明」を参照してください。

`level:Number` - 変数を受け取る Flash Player のレベルを指定する整数。

`method:String` (オプション) - 変数を送信するための HTTP メソッドを指定します。パラメータはストリング `GET` または `POST` でなければなりません。送る変数がない場合は、このパラメータを省略します。`GET` メソッドは、変数を URL の最後に追加します。このメソッドは、変数のデータ量が少ないときに使用します。`POST` メソッドは、別の HTTP ヘッダで変数を送信します。このメソッドは、変数のデータ量が多いときに使用します。

例

次の例では、テキストファイル params.txt の情報を、Flash Player のレベル 2 にある SWF ファイルのメインタイムラインにロードします。テキストフィールドの変数名は、params.txt ファイルで宣言されている変数名と同じでなければなりません。`setInterval()` 関数は、SWF へのデータのロードの進行状況をチェックする場合に使用します。このスクリプトは、params.txt ファイル内の `done` という名前の変数をチェックします。

```
loadVariablesNum("params.txt", 2);
function checkParamsLoaded() {
 if (_level2.done == undefined) {
 trace("not yet.");
 } else {
 trace("finished loading. killing interval.");
 trace("-------------");
 for (i in _level2) {
 trace(i+": "+_level2[i]);
 }
 trace("-------------");
 clearInterval(param_interval);
 }
}
var param_interval:Number = setInterval(checkParamsLoaded, 100);

// Params.txt includes the following text
var1="hello"&var2="goodbye"&done="done"
```

関連項目

getURL 関数,loadMovie 関数,loadMovieNum 関数,loadVariables 関数,loadMovie (MovieClip.loadMovie メソッド),loadVariables (MovieClip.loadVariables メソッド),load (LoadVars.load メソッド)

mbchr 関数

```
mbchr(number:Number)
```

非推奨 Flash Player 5 以降では使用しないでください。この関数の代わりに String.fromCharCode() メソッドを使用します。

ASCII コード番号をマルチバイト文字に変換します。

対応バージョン: ActionScript 1.0、Flash Player 4

パラメータ

`number:Number` - マルチバイト文字に変換する数値。

関連項目

fromCharCode (String.fromCharCode メソッド)

mblength 関数

```
mblength(string:String) : Number
```

非推奨 Flash Player 5 以降では使用しないでください。この関数の代わりに String クラスのメソッドとプロパティを使用します。

マルチバイト文字のストリング長を返します。

対応バージョン: ActionScript 1.0、Flash Player 4

パラメータ

`string:String` - 長さを調べるストリング。

戻り値

Number - マルチバイト文字のストリング長。

関連項目

String,length (String.length プロパティ)

mbord 関数

```
mbord(character:String) : Number
```

非推奨 Flash Player 5 以降では使用しないでください。この関数の代わりに `String.charCodeAt()` を使用します。

指定された文字をマルチバイト文字コード番号に変換します。

対応バージョン： ActionScript 1.0、Flash Player 4

パラメータ

`character:String` - マルチバイト数値に変換する文字。

戻り値

`Number` - 変換された文字。

関連項目

`charCodeAt（String.charCodeAt メソッド）`

mbsubstring 関数

```
mbsubstring(value:String, index:Number, count:Number) : String
```

非推奨 Flash Player 5 以降では使用しないでください。この関数の代わりに `String.substr()` を使用します。

マルチバイト文字ストリングから、マルチバイト文字ストリングを抽出します。

対応バージョン： ActionScript 1.0、Flash Player 4

パラメータ

`value:String` - 抽出元のマルチバイト文字ストリング。

`index:Number` - 抽出する最初の文字の位置を表す数値。

`count:Number`- 抽出されるストリングに含まれる文字数。インデックス文字は除きます。

戻り値

`String` - マルチバイト文字ストリングから抽出されたストリング。

関連項目

`substr（String.substr メソッド）`

MMExecute 関数

`MMExecute(`*"Flash JavaScript API command;":String*`) : String`

ActionScript から Flash JavaScript API (JSAPI) のコマンドを発行できます。Flash MX2004 では、`MMExecute` 関数は、Flash パネルとして使用されるムービー (ファイルは WindowSWF ディレクトリに格納)、XMLtoUI ダイアログボックス、またはコンポーネントのカスタム UI によってのみ呼び出されます。JSAPI コマンドは、Flash Player、テストムービーモード、またはオーサリング環境の外部では無効になります。

Flash JSAPI には、ユーザーがオーサリング環境で入力可能なコマンドを複製 (エミュレート) するためのいくつかのオブジェクト、メソッド、およびプロパティが備わっています。JSAPI では、メニューにコマンドを追加する、ステージ上のオブジェクトを操作する、一連のコマンドを繰り返し実行するといった方法をスクリプトに取り入れることによって、Flash の機能を拡張できます。

通常、ユーザーは [コマンド]-[コマンドの実行] を選択して JSAPI スクリプトを実行します。ただし、この関数を ActionScript スクリプトの中で使用して JSAPI コマンドを直接呼び出すこともできます。ファイルのフレーム 1 上のスクリプトで `MMExecute()` を使用した場合、このコマンドは SWF ファイルのロード時に実行されます。

JSAPI の詳細については、www.adobe.com/go/jsapi_info_jp を参照してください。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

`command:String` - Flash JavaScript (JSFL) ファイル内で使用できる任意のコマンド。

戻り値

`String` - JavaScript ステートメントによって返される結果のストリング表現 (存在する場合)。

例

次のコマンドでは、現在のドキュメントのライブラリ内のアイテム数をトレースウィンドウに出力します。この例は Flash パネルとして実行する必要があります。Flash ファイルをムービープレビューやブラウザで実行すると `MMExecute` を呼び出せないためです。

- 次のコードを、空の Flash ドキュメントのメインタイムラインのフレーム 1 に配置します。`var numLibItems = MMExecute("fl.getDocumentDOM().library.items.length");var message = numLibItems + " items in library";MMExecute('fl.trace("' + message + '");');`
- FLA ファイルを設定ディレクトリ内にある WindowSWF ディレクトリに保存して、[ファイル]-[パブリッシュ] を選択します (または、別の場所に保存して、SWF ファイルをそのディレクトリに直接パブリッシュするか、SWF ファイルをそのディレクトリに移動させます)。

■ アプリケーションを終了し、再起動します (WindowSWF ディレクトリに初めてファイルを追加するときに必要)。

次に、[ウィンドウ]-[他のパネル] メニューの下部でファイルを選択します。

ActionScript の trace 関数は Flash パネルから動作しません。この例では JavaScript `fl.trace` バージョンを使用して出力を取得します。`MMExecute` の結果を Flash パネルファイルの一部であるテキストフィールドにコピーする方が簡単な場合もあります。

nextFrame 関数

```
nextFrame() : Void
```

次のフレームに再生ヘッドを送ります。

対応バージョン： ActionScript 1.0、Flash Player 2

例

次の例では、ユーザーが右矢印キーまたは下矢印キーを押すと、再生ヘッドが次のフレームに進んで停止します。ユーザーが左矢印キーまたは上矢印キーを押すと、再生ヘッドは前のフレームに戻って停止します。リスナーは初期化されます。矢印キーが押されるまで待機し、`init` 変数を使用して、再生ヘッドがフレーム 1 に移動したときにリスナーが再定義されるのを防ぎます。

```
stop();

if (init == undefined) {
 someListener = new Object();
 someListener.onKeyDown = function() {
 if (Key.isDown(Key.LEFT) || Key.isDown(Key.UP)) {
 _level0.prevFrame();
 } else if (Key.isDown(Key.RIGHT) || Key.isDown(Key.DOWN)) {
 _level0.nextFrame();
 }
 };
 Key.addListener(someListener);
 init = 1;
}
```

関連項目

`prevFrame` 関数

nextScene 関数

```
nextScene() : Void
```

次のシーンのフレーム 1 に再生ヘッドを送ります。

対応バージョン： ActionScript 1.0、Flash Player 2

例

次の例では、ユーザーが実行時に作成されたボタンをクリックすると、再生ヘッドが次のシーンのフレーム 1 に送られます。2 つのシーンを作成し、次の ActionScript をシーン 1 のフレーム 1 に入力します。

```
stop();

if (init == undefined) {
 this.createEmptyMovieClip("nextscene_mc", this.getNextHighestDepth());
 nextscene_mc.createTextField("nextscene_txt", this.getNextHighestDepth(), 200,
  0, 100, 22);
 nextscene_mc.nextscene_txt.autoSize = true;
 nextscene_mc.nextscene_txt.border = true;
 nextscene_mc.nextscene_txt.text = "Next Scene";
 this.createEmptyMovieClip("prevscene_mc", this.getNextHighestDepth());
 prevscene_mc.createTextField("prevscene_txt", this.getNextHighestDepth(), 00,
  0, 100, 22);
 prevscene_mc.prevscene_txt.autoSize = true;
 prevscene_mc.prevscene_txt.border = true;
 prevscene_mc.prevscene_txt.text = "Prev Scene";
 nextscene_mc.onRelease = function() {
 nextScene();
 };

 prevscene_mc.onRelease = function() {
 prevScene();
 };

 init = true;
}
```

stop() アクションをシーン 2 のフレーム 1 に配置したことを確認してください。

関連項目

prevScene 関数

Number 関数

```
Number(expression) : Number
```

次に示すように、パラメータ *expression* を数値に変換し、値を返します。

- *expression* が数値の場合は、戻り値は *expression* です。
- *expression* がブール値の場合は、*expression* が true であれば戻り値は 1 となり、*expression* が false であれば戻り値は 0 となります。
- *expression* がストリングである場合、関数は、*expression* を 1.57505e-3 のような指数表現を伴う 10 進数として解析しようとします。
- *expression* が NaN の場合は、戻り値は NaN です。
- *expression* が undefined の場合の戻り値は次のようになります。- Flash Player 6 以前用にパブリッシュされたファイルでは、結果は 0 になります。- Flash Player 7 以降用にパブリッシュされたファイルでは、結果は NaN になります。

対応バージョン : ActionScript 1.0、Flash Player 4 - Flash Player 7 ではビヘイビアが変更されました。

パラメータ

expression: Object - 数値に変換される式。0x で始まる数値やストリングは 16 進数として解釈されます。0 で始まる数値やストリングは 8 進数として解釈されます。

戻り値

Number - 数値または NaN (非数)。

例

次の例では、実行時にステージでテキストフィールドが作成されます。

```
this.createTextField("counter_txt", this.getNextHighestDepth(), 0, 0, 100, 22);
counter_txt.autoSize = true;
counter_txt.text = 0;
function incrementInterval():Void {
 var counter:Number = counter_txt.text;
 // Without the Number() function, Flash would concatenate the value instead
 // of adding values. You could also use "counter_txt.text++;"
 counter_txt.text = Number(counter) + 1;
}
var intervalID:Number = setInterval(incrementInterval, 1000);
```

関連項目

NaN 定数, 数値 (Number), parseInt 関数, parseFloat 関数

Object 関数

```
Object([value:Object]) : Object
```

空のオブジェクトを新規作成するか、指定された数値、ストリング、またはブール値をオブジェクトに変換します。このコマンドは、Object コンストラクタを使用してオブジェクトを作成することと同じです。「Object クラスのコンストラクタ」を参照してください。

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

`value:Object` (オプション) - 数値、ストリング、またはブール値。

戻り値

`Object` - オブジェクト。

例

次の例では、新しい空のオブジェクトが作成され、そのオブジェクトに値が格納されます。

```
var company:Object = new Object();
company.name = "Adobe";
company.address = "601 Townsend Street";
company.city = "San Francisco";
company.state = "CA";
company.postal = "94103";
for (var i in company) {
 trace("company."+i+" = "+company[i]);
}
```

関連項目

`Object`

on ハンドラ

```
on(mouseEvent:Object) {
// your statements here
}
```

アクションをトリガするマウスイベントまたはキー押下を指定します。

対応バージョン： ActionScript 1.0、Flash Player 2 - Flash 2 ではサポートされないイベントもあります。

パラメータ

mouseEvent: Object - *mouseEvent* は、イベントと呼ばれるトリガです。イベントが発生すると、中カッコ ({}) 内でそれに続くステートメントが実行されます。次の値のすべてを、*mouseEvent* パラメータに指定できます。

- press - ポインタがボタン上にあるときにマウスボタンを押した場合。
- release - ポインタがボタン上にあるときにマウスボタンを離した場合。
- releaseOutside - ポインタがボタン上にあるときにマウスボタンを押し、そのままボタン領域の外側に移動してからボタンを離した場合。press イベントと dragOut イベントは両方とも、常に releaseOutside イベントより前に配置します。
- rollOut - ポインタがボタン領域の外側に移動した場合。
- rollOver - マウスポインタがボタン上に移動した場合。
- dragOut - ポインタがボタン上にあるときにマウスボタンを押し、そのままボタン領域の外側に移動した場合。
- dragOver - ポインタがボタン上にあるときにマウスボタンを押し、そのままボタンの外側に移動し、またボタン上に戻った場合。
- keyPress "<*key*>" 指定されたキーを押した場合。パラメータの *key* の部分には、[アクション] パネルのコードヒントで示されているように、キー定数を指定します。このパラメータを使用してキー入力を取得できます。つまり、特定のキーの組み込みビヘイビアを上書きできます。ボタンは、アプリケーションの任意の場所 (ステージ上またはステージ外) に配置できます。この技術には制約が 1 つあります。on() ハンドラを実行時に適用できないため、オーサリング時に適用する必要があります。必ず [制御]-[キーボードショートカットを無効] を選択してください。[制御]-[ムービープレビュー] を使用してアプリケーションをテストすると、ビルトインビヘイビアを持つ特定のキーは上書きされません。

キー定数の一覧については、Key クラスを参照してください。

例

次のスクリプトでは、マウスを押したときに startDrag() 関数が実行され、マウスを離してオブジェクトをドロップしたときに条件付きスクリプトが実行されます。

```
on (press) {
 startDrag(this);
}
on (release) {
 trace("X:"+this._x);
 trace("Y:"+this._y);
 stopDrag();
}
```

関連項目

onClipEvent ハンドラ , キー

onClipEvent ハンドラ

```
onClipEvent(movieEvent:Object) {
// your statements here
}
```

ムービークリップの特定のインスタンスに対して定義されたアクションを起動します。

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

movieEvent:Object - *movieEvent* は、イベントと呼ばれるトリガです。イベントが発生すると、その後ろの中カッコ ({}) 内のステートメントが実行されます。次の値のすべてを、*movieEvent* パラメータに指定できます。

- load - ムービークリップがインスタンス化され、タイムラインに表示されると、アクションが即座に開始されます。
- unload - ムービークリップがタイムラインから削除された後、最初のフレームでアクションが開始します。Unload ムービークリップイベントに関連するアクションは、該当するフレームにアクションが割り当てられる前に処理されます。
- enterFrame - アクションはムービークリップのフレームレートで継続的にトリガされます。enterFrame クリップイベントに関連するアクションは、該当するフレームに割り当てられているフレームアクションの前に処理されます。
- mouseMove - マウスを動かすたびに、アクションが開始されます。現在のマウスの位置を判別するには、_xmouse プロパティと _ymouse プロパティを使用します。
- mouseDown - 左マウスボタンを押すと、アクションが開始されます。
- mouseUp - 左マウスボタンを離すと、アクションが開始します。
- keyDown - キーを押すと、アクションが開始します。最後に押されたキーについての情報を取得するには、Key.getCode() を使用します。
- keyUp - キーを離すと、アクションが開始します。最後に押されたキーについての情報を取得するには、Key.getCode() メソッドを使用します。
- data - loadVariables() アクションまたは loadMovie() アクションによってデータを取得すると、アクションが開始されます。loadVariables() アクションと共に指定すると、data イベントは最後の変数がロードされたときの 1 回だけ発生します。loadMovie() アクションで指定すると、各データセクションが読み込まれるたびに、data イベントが繰り返し発生します。

例

次の例は、keyDown ムービーイベントで onClipEvent() を使用しており、ムービークリップまたはボタンにアタッチされるように設計されています。通常、keyDown ムービーイベントは Key オブジェクトのメソッドやプロパティと共に使用します。次のスクリプトでは、Key.getCode() を使用して、ユーザーが押したキーを調べます。押されたキーが Key.RIGHT プロパティと一致する場合は、再生ヘッドを次のフレームに送ります。押されたキーが Key.LEFT プロパティと一致する場合は、再生ヘッドを前のフレームに送ります。

```
onClipEvent (keyDown) {
 if (Key.getCode() == Key.RIGHT) {
 this._parent.nextFrame();
 } else if (Key.getCode() == Key.LEFT) {
 this._parent.prevFrame();
 }
}
```

次の例では、load および mouseMove ムービーイベントで onClipEvent() を使用します。_xmouse プロパティと _ymouse プロパティは、マウスが移動するたびにマウスの位置を追跡します。位置は、実行時に作成されるテキストフィールドに表示されます。

```
onClipEvent (load) {
 this.createTextField("coords_txt", this.getNextHighestDepth(), 0, 0, 100, 22);
 coords_txt.autoSize = true;
 coords_txt.selectable = false;
}
onClipEvent (mouseMove) {
 coords_txt.text = "X:"+_root._xmouse+",Y:"+_root._ymouse;
}
```

関連項目

キー , _xmouse (MovieClip._xmouse プロパティ), _ymouse (MovieClip._ymouse プロパティ), on ハンドラ , updateAfterEvent 関数

ord 関数

```
ord(character:String) : Number
```

非推奨 Flash Player 5 以降では使用しないでください。この関数の代わりに String クラスのメソッドとプロパティを使用します。

文字を ASCII コード番号に変換します。

対応バージョン： ActionScript 1.0、Flash Player 4

パラメータ

`character:String` - ASCII コード番号に変換する文字。

戻り値

`Number` - 指定した文字の ASCII コード番号。

関連項目

`String`, `charCodeAt (String.charCodeAt メソッド )`

parseFloat 関数

```
parseFloat(string:String) : Number
```

ストリングを浮動小数に変換します。この関数は、先頭から始まる数値に含まれない文字に達するまで、ストリング内の数値を読み取って (つまり解析して)、結果を返します。ストリングが解析できる数値で始まっていない場合、`parseFloat()` は `NaN` を返します。有効な整数の前の空白は、後続の非数値文字と同様に無視されます。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`string:String` - 読み込まれて浮動小数に変換されるストリング。

戻り値

`Number` - 数値または `NaN` (非数)。

例

次の例では、`parseFloat()` 関数を使用して各種数値を評価します。

```
trace(parseFloat("-2")); // -2
trace(parseFloat("2.5")); // 2.5
trace(parseFloat(" 2.5")); // 2.5
trace(parseFloat("3.5e6")); // 3500000
trace(parseFloat("foobar")); // NaN
trace(parseFloat("3.75math")); // 3.75
trace(parseFloat("0garbage")); // 0
```

関連項目

`NaN 定数`, `parseInt 関数`

parseInt 関数

```
parseInt(expression:String, [radix:Number]) : Number
```

ストリングを整数に変換します。パラメータで指定されたストリングを数値に変換できない場合は NaN を返します。0x から始まる整数は、16 進数と解釈されます。0 から始まる整数、または基数として 8 を指定する整数は 8 進数と解釈されます。有効な整数の前の空白は、後続の非数値文字と同様に無視されます。

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

`expression:String` - 整数に変換されるストリング。

`radix:Number` (オプション) - 解析する数値の基数を表す整数。有効な値は、2 ～ 36 です。

戻り値

`Number` - 数値または `NaN` (非数)。

例

次の例では、`parseInt()` 関数を使用して各種の数値を評価します。

この例では 3 を返します。

```
parseInt("3.5")
```

次の例では NaN を返します。

```
parseInt("bar")
```

次の例では 4 を返します。

```
parseInt("4foo")
```

次の例は 16 進数の変換例で、1016 を返します。

```
parseInt("0x3F8")
```

次の例は、オプションの *radix* パラメータを使用した 16 進数の変換例で、1000 を返します。

```
parseInt("3E8", 16)
```

次は 2 進数の変換例で、10 (2 進数の 1010 の 10 進表現) を返します。

```
parseInt("1010", 2)
```

次は 8 進数の解析例で、511 (8 進数 777 の 10 進表現) を返します。

```
parseInt("0777")
parseInt("777", 8)
```

関連項目

, `parseFloat 関数`

play 関数

```
play() : Void
```

タイムライン内で再生ヘッドを前へ進めます。

対応バージョン： ActionScript 1.0、Flash Player 2

例

次の例では、stop_mc および play_mc というインスタンス名の 2 つのムービークリップインスタンスがステージにあります。ActionScript は、stop_mc ムービークリップインスタンスがクリックされると、SWF ファイルの再生を停止します。play_mc インスタンスがクリックされると、再生を再開します。

```
this.stop_mc.onRelease = function() {
 stop();
};
this.play_mc.onRelease = function() {
 play();
};
trace("frame 1");
```

関連項目

gotoAndPlay 関数, gotoAndPlay (MovieClip.gotoAndPlay メソッド)

prevFrame 関数

```
prevFrame() : Void
```

前のフレームに再生ヘッドを送ります。現在のフレームが 1 である場合、再生ヘッドは移動しません。

対応バージョン： ActionScript 1.0、Flash Player 2

例

次の ActionScript が myBtn_btn というボタンのタイムラインのフレームに配置されていると、ユーザーがこのボタンをクリックしたときに、再生ヘッドが前のフレームに送られます。

```
stop();
this.myBtn_btn.onRelease = function(){
 prevFrame();
};
```

関連項目

nextFrame 関数, prevFrame (MovieClip.prevFrame メソッド)

prevScene 関数

```
prevScene() : Void
```

前のシーンのフレーム 1 に再生ヘッドを送ります。

対応バージョン： ActionScript 1.0、Flash Player 2

関連項目

nextScene 関数

print 関数

```
print(target:Object, boundingBox:String) : Void
```

パラメータ (bmovie、bmax、または bframe) で指定された境界に従って、target ムービークリップを印刷します。ターゲットムービー内のフレームを指定して印刷する場合は、指定するフレームに #p フレームラベルを割り当てます。print() は、printAsBitmap() よりも高い品質の印刷を生成できますが、アルファ透明度や特殊カラー効果を使用するムービーを印刷する際には使うことができません。

boundingBox パラメータに bmovie を指定し、フレームに #b ラベルを割り当てなかった場合、印刷領域はロードされたムービーのステージサイズによって決まります。ロードされたムービーには、メインのムービークリップのステージサイズは継承されません。

ムービークリップ内の印刷可能エレメントをすべて完全にロードしないと、印刷を開始できません。

Flash Player の印刷機能は、PostScript プリンタと非 PostScript プリンタをサポートしています。非 PostScript プリンタでは、ベクターはビットマップに変換されます。

対応バージョン： ActionScript 1.0、Flash Player 4 - (4.0.20.0) オーサリングの対象が Flash Player 7 以降の場合、PrintJob オブジェクトを作成すれば、印刷処理を制御しやすくなります。詳細については、PrintJob クラスを参照してください。

パラメータ

target:Object - 印刷するムービークリップのインスタンス名。デフォルトでは、ターゲットインスタンス内のすべてのフレームを印刷できます。ムービークリップ内のフレームを指定して印刷する場合は、指定するフレームに #p フレームラベルを割り当てます。

boundingBox:String - ムービークリップの印刷領域を設定する修飾子。このパラメータは引用符 (" または ') で囲み、次のいずれかの値を指定します。

- bmovie - ムービー内の特定のフレームの境界ボックスを、ムービー内のすべての印刷可能なフレーム用の印刷領域として指定します。印刷領域として使用する境界ボックスのフレームに #b フレームラベルを割り当てます。
- bmax - 印刷領域として、印刷可能なフレームすべての全境界ボックスから成るコンポジットを指定します。ムービークリップ内の印刷可能なフレームのサイズが異なるときは、bmax パラメータを指定します。
- bframe - 各印刷可能フレームの境界ボックスをそのフレームの印刷領域として使用することを指定します。これにより各フレームの印刷領域が変化し、印刷領域に収まるようにオブジェクトが拡大 / 縮小されます。各フレーム内にサイズの異なるオブジェクトがある場合、各オブジェクトを印刷領域に収めるには、bframe を使用します。

例

次の例では、各フレームの境界ボックスで定義された印刷領域を使って holder_mc 内の印刷可能フレームをすべて印刷します。

```
this.createEmptyMovieClip("holder_mc", 999);
holder_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");

this.myBtn_btn.onRelease = function() {
 print(this._parent.holder_mc, "bframe");
};
```

この ActionScript では、bframe を bmovie で置き換えることで、#b というフレームラベルが付けられたフレームの境界ボックスによって印刷領域が定義されるようにできます。

関連項目

printAsBitmap 関数, printAsBitmapNum 関数, PrintJob, printNum 関数

printAsBitmap 関数

```
printAsBitmap(target:Object, boundingBox:String) : Void
```

パラメータ (bmovie、bmax、または bframe) で指定された境界に従って、target ムービークリップをビットマップとして印刷します。透明度またはカラー効果を利用したオブジェクトによるフレームを含むムービークリップを印刷するには、printAsBitmap() を使用します。printAsBitmap() アクションは、できるだけ多くの定義と高品質を維持するために、そのプリンタで利用できる最高解像度で印刷を行います。

ムービークリップにアルファ透明度やカラー効果が含まれていない場合は、印刷品質を上げるために、print() を使うことをお勧めします。

boundingBox パラメータに bmovie を指定し、フレームに #b ラベルを割り当てなかった場合、印刷領域はロードされたムービーのステージサイズによって決まります。ロードされたムービーには、メインのムービークリップのステージサイズは継承されません。

ムービークリップ内の印刷可能エレメントをすべて完全にロードしないと、印刷を開始できません。

Flash Player の印刷機能は、PostScript プリンタと非 PostScript プリンタをサポートしています。非 PostScript プリンタでは、ベクターはビットマップに変換されます。

対応バージョン： ActionScript 1.0、Flash Player 4 - (4.0.20.0) オーサリングの対象が Flash Player 7 以降の場合、PrintJob オブジェクトを作成すれば、印刷処理を制御しやすくなります。詳細については、PrintJob クラスを参照してください。

パラメータ

target:Object - 印刷するムービークリップのインスタンス名。デフォルトでは、ムービークリップ内のすべてのフレームが印刷されます。ムービークリップ内のフレームを指定して印刷する場合は、指定するフレームに #p フレームラベルを割り当てます。

boundingBox:String - ムービークリップの印刷領域を設定する修飾子。このパラメータは引用符 (" または ') で囲み、次のいずれかの値を指定します。

- bmovie - ムービー内の特定のフレームの境界ボックスを、ムービー内のすべての印刷可能なフレーム用の印刷領域として指定します。印刷領域として使用する境界ボックスのフレームに #b フレームラベルを割り当てます。
- bmax - 印刷領域として、印刷可能なフレームすべての全境界ボックスから成るコンポジットを指定します。ムービー内の印刷可能なフレームのサイズが異なるときは、bmax パラメータを指定します。
- bframe - 各印刷可能フレームの境界ボックスをそのフレームの印刷領域として使用することを指定します。これにより各フレームの印刷領域が変化し、印刷領域に収まるようにオブジェクトが拡大 / 縮小されます。各フレーム内にサイズの異なるオブジェクトがある場合、各オブジェクトを印刷領域に収めるには、bframe を使用します。

例

次の例では、フレームの境界ボックスで定義された印刷領域を使って holder_mc 内の印刷可能フレームをすべて印刷します。

```
this.createEmptyMovieClip("holder_mc", 999);
holder_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");

this.myBtn_btn.onRelease = function() {
 printAsBitmap(this._parent.holder_mc, "bframe");
};
```

関連項目

print 関数, printAsBitmapNum 関数, printNum 関数, PrintJob

printAsBitmapNum 関数

```
printAsBitmapNum(level:Number, boundingBox:String) : Void
```

パラメータ (bmovie、bmax、または bframe) で指定された境界に従って、Flash Player 内のレベルをビットマップとして印刷します。透明度またはカラー効果を利用したオブジェクトによるフレームを含むムービークリップを印刷するには、printAsBitmapNum() を使用します。printAsBitmapNum() アクションは、できるだけ高度な定義と品質を維持するために、最高のプリンタ解像度で印刷を行います。ビットマップとして印刷するように指定されたフレームの印刷可能なファイルサイズを計算するには、ピクセル幅にピクセル高さとプリンタ解像度を乗じます。

ムービーにアルファ透明度やカラー効果が含まれていない場合は、印刷品質を上げるために、printNum() を使うことをお勧めします。

boundingBox パラメータに bmovie を指定し、フレームに #b ラベルを割り当てなかった場合、印刷領域はロードされたムービーのステージサイズによって決まります。ロードされたムービーには、メインのムービーのステージサイズは継承されません。

ムービークリップ内の印刷可能エレメントをすべて完全にロードしないと、印刷を開始できません。

Flash Player の印刷機能は、PostScript プリンタと非 PostScript プリンタをサポートしています。非 PostScript プリンタでは、ベクターはビットマップに変換されます。

対応バージョン： ActionScript 1.0、Flash Player 5 - オーサリングの対象が Flash Player 7 以降の場合、PrintJob オブジェクトを作成すれば、印刷処理を制御しやすくなります。詳細については、PrintJob クラスを参照してください。

パラメータ

level:Number - 印刷する Flash Player のレベル。デフォルトでは、レベル内のすべてのフレームが印刷されます。レベル内のフレームを指定して印刷する場合は、指定するフレームに #p フレームラベルを割り当てます。

boundingBox:String - ムービークリップの印刷領域を設定する修飾子。このパラメータは引用符 (" または ') で囲み、次のいずれかの値を指定します。

- bmovie - ムービー内の特定のフレームの境界ボックスを、ムービー内のすべての印刷可能なフレーム用の印刷領域として指定します。印刷領域として使用する境界ボックスのフレームに #b フレームラベルを割り当てます。
- bmax - 印刷領域として、印刷可能なフレームすべての全境界ボックスから成るコンポジットを指定します。ムービー内の印刷可能なフレームのサイズが異なるときは、bmax パラメータを指定します。

- bframe - 各印刷可能フレームの境界ボックスをそのフレームの印刷領域として使用することを指定します。これにより各フレームの印刷領域が変化し、印刷領域に収まるようにオブジェクトが拡大 / 縮小されます。各フレーム内にサイズの異なるオブジェクトがある場合、各オブジェクトを印刷領域に収めるには、bframe を使用します。

例

次の例では、ユーザーが myBtn_btn ボタンをクリックすると、ステージの内容が印刷されます。印刷する領域は、フレームの境界ボックスによって定義されます。

```
myBtn_btn.onRelease = function(){
 printAsBitmapNum(0, "bframe")
};
```

関連項目

print 関数, printAsBitmap 関数, PrintJob, printNum 関数

printNum 関数

```
printNum(level:Number, boundingBox:String) : Void
```

boundingBox パラメータ (bmovie、bmax、bframe) で指定された境界に従って、Flash Player 内のレベルを印刷します。ターゲットムービー内のフレームを指定して印刷する場合は、指定するフレームに #p フレームラベルを割り当てます。printNum() を使うと、printAsBitmapNum() を使う場合よりも高品質で印刷できますが、アルファ透明度や特殊カラー効果を含むムービーを printNum() で印刷することはできません。

boundingBox パラメータに bmovie を指定し、フレームに #b ラベルを割り当てなかった場合、印刷領域はロードされたムービーのステージサイズによって決まります。ロードされたムービーには、メインのムービーのステージサイズは継承されません。

ムービークリップ内の印刷可能エレメントをすべて完全にロードしないと、印刷を開始できません。

Flash Player の印刷機能は、PostScript プリンタと非 PostScript プリンタをサポートしています。非 PostScript プリンタでは、ベクターはビットマップに変換されます。

対応バージョン： ActionScript 1.0、Flash Player 5 - オーサリングの対象が Flash Player 7 以降の場合、PrintJob オブジェクトを作成すれば、印刷処理を制御しやすくなります。詳細については、PrintJob クラスを参照してください。

パラメータ

`level:Number` - 印刷する Flash Player のレベル。デフォルトでは、レベル内のすべてのフレームが印刷されます。レベル内のフレームを指定して印刷する場合は、指定するフレームに #p フレームラベルを割り当てます。

`boundingBox:String` - ムービークリップの印刷領域を設定する修飾子。このパラメータは引用符 (" または ') で囲み、次のいずれかの値を指定します。

- `bmovie` - ムービー内の特定のフレームの境界ボックスを、ムービー内のすべての印刷可能なフレーム用の印刷領域として指定します。印刷領域として使用する境界ボックスのフレームに #b フレームラベルを割り当てます。
- `bmax` - 印刷領域として、印刷可能なフレームすべての全境界ボックスから成るコンポジットを指定します。ムービー内の印刷可能なフレームのサイズが異なるときは、`bmax` パラメータを指定します。
- `bframe` - 各印刷可能フレームの境界ボックスをそのフレームの印刷領域として使用することを指定します。これにより各フレームの印刷領域が変化し、印刷領域に収まるようにオブジェクトが拡大 / 縮小されます。各フレーム内にサイズの異なるオブジェクトがある場合、各オブジェクトを印刷領域に収めるには、`bframe` を使用します。

関連項目

`print 関数`, `printAsBitmap 関数`, `printAsBitmapNum 関数`, `PrintJob`

random 関数

```
random(value:Number) : Number
```

非推奨 Flash Player 5 以降では使用しないでください。この関数の代わりに `Math.random()` を使用します。

0 から *value* パラメータで指定された整数まで (この整数は含まない) の間のランダムな整数を返します。

対応バージョン : ActionScript 1.0、Flash Player 4

パラメータ

`value:Number` - 整数。

戻り値

`Number` - ランダムな整数。

例

次のように `random()` を使用すると、値 0、1、2、3、または 4 が返されます。`random(5);`

関連項目

random (Math.random メソッド)

removeMovieClip 関数

removeMovieClip(target:Object)

指定されたムービークリップを削除します。

対応バージョン : ActionScript 1.0、Flash Player 4

パラメータ

target:Object - duplicateMovieClip() で作成したムービークリップインスタンスのターゲットパスか、MovieClip.attachMovie()、MovieClip.duplicateMovieClip()、または MovieClip.createEmptyMovieClip() で作成したムービークリップのインスタンス名。

例

次の例では、myClip_mc という新しいムービークリップを作成し、そのムービークリップを複製します。2 番目のムービークリップは newClip_mc という名前です。両方のムービークリップにイメージがロードされます。button_mc ボタンがクリックされると、複製されたムービークリップがステージから削除されます。

```
this.createEmptyMovieClip("myClip_mc", this.getNextHighestDepth());
myClip_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
duplicateMovieClip(this.myClip_mc, "newClip_mc", this.getNextHighestDepth());
newClip_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
newClip_mc._x = 200;
this.button_mc.onRelease = function() {
 removeMovieClip(this._parent.newClip_mc);
};
```

関連項目

duplicateMovieClip 関数, duplicateMovieClip (MovieClip.duplicateMovieClip メソッド), attachMovie (MovieClip.attachMovie メソッド), removeMovieClip (MovieClip.removeMovieClip メソッド), createEmptyMovieClip (MovieClip.createEmptyMovieClip メソッド)

setInterval 関数

```
setInterval(functionReference:Function, interval:Number, [param1:Object, param2,
  ..., paramN]) : Number
setInterval(objectReference:Object, methodName:String, interval:Number,
  [param1:Object, param2, ..., paramN]) : Number
```

SWF ファイルの再生時に一定の間隔で関数、またはオブジェクトのメソッドを呼び出します。`setInterval()` を使用すると、任意の関数を一定の間隔で繰り返し実行できます。

`setInterval()` を使用するためのヒントを次に示します。

- 呼び出されている関数のスコープを識別します。
- インターバル ID (`setInterval()` の戻り値) が設定されたスコープを識別します。
- 前に設定した間隔をクリアしてから、新しい間隔を設定します。

これらのヒントについて、以下で詳しく説明します。

呼び出されている関数のスコープを識別します。呼び出される関数のスコープを識別するには、`setInterval()` メソッドが実行できるオブジェクト (オブジェクトスコープ) を最初のパラメータとして渡し、実行するメソッドの名前を 2 番目のパラメータとして渡します (2 つ目のシグネチャで示す)。これによって、目的のメソッドが、渡されたオブジェクト参照スコープから実行されます。メソッドは、このような方法で実行されると、オブジェクト上のメンバー変数を `this` キーワードを使用して参照できます。

インターバル ID が設定されたスコープを識別します。インターバル ID (`intervalId`) が設定されたスコープを識別するには、`setInterval()` に渡すオブジェクトスコープ上のメンバー変数にこの ID を代入します。これによって、呼び出された関数が `this.intervalId` でインターバル ID を見つけることができます。

前に設定した間隔をクリアします。前に設定した間隔をクリアしてから新しい間隔を設定するには、通常、`setInterval()` を呼び出す " 前 " に、`clearInterval()` を必ず呼び出す必要があります。これによって、現在実行されている間隔への唯一の参照である `intervalId` 変数の上書きや破棄を回避できます。`setInterval()` を呼び出す前に `clearInterval()` を呼び出すには、例に示しているように、開始スクリプトと実行スクリプトの両方が `intervalId` にアクセスする必要があります。

メモ : スクリプトでループ処理を停止するには、必ず `clearInterval()` を呼び出してください。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

`functionReference:Function` - 呼び出される関数への参照。

`interval:Number` - `functionReference` または `methodName` 関数に対する呼び出しの間隔 (ミリ秒)。

interval が SWF ファイルのフレームレート (たとえば 10 フレーム / 秒 [fps] の場合は 100 ミリ秒の間隔) より短い場合、インターバル関数は interval にできるだけ近い間隔で呼び出されます。呼び出す間隔にメモリに負荷のかかる長いスクリプトを実行すると、呼び出しの遅延が発生します。呼び出された関数によりビジュアルエレメントに変更が加えられる場合は、updateAfterEvent() 関数を使用して、十分な画面更新回数を確保する必要があります。interval が SWF ファイルのフレームレートより長い場合は、画面の更新に伴う影響を最小限に抑えるために、interval が経過し、しかも再生ヘッドが次のフレームに入った後にのみ、インターバル関数が呼び出されます。

param:Object (オプション) - functionReference または methodName に送られた関数に渡すパラメータ。複数のパラメータは、次のようにカンマで区切ります。*param1* , *param2* , ..., *paramN*

objectReference:Object - methodName で指定したメソッドを含むオブジェクト。

methodName:String - objectReference で指定したオブジェクトのスコープ内に存在するメソッド。

戻り値

Number - 間隔を識別するための整数 (インターバル ID)。これを clearInterval() に渡して間隔をクリアします。

例

例 1: 次の例では、メッセージを 20 ミリ秒間隔で最大 10 回出力した後、この間隔をクリアします。オブジェクトスコープ this が最初のパラメータとして渡され、メソッド名 executeCallback が 2 番目のパラメータとして渡されます。これによって、executeCallback() は、呼び出し元のスクリプトと同じスコープから実行されるようになります。

```
var intervalId:Number;
var count:Number = 0;
var maxCount:Number = 10;
var duration:Number = 20;

function executeCallback():Void {
 trace("executeCallback intervalId: " + intervalId + " count: " + count);
 if(count >= maxCount) {
 clearInterval(intervalId);
 }
 count++;
}

intervalId = setInterval(this, "executeCallback", duration);
```

例 2 : 次の例は最初の例と似ていますが、setInterval() の前に clearInterval() を呼び出している点が異なります。これによって、不要なループを避けることができます。このことは、特定の間隔がクリアされるまで開始スクリプトを何度でも実行できる、イベントベースのシステムでは特に重要となります。

```
var intervalId:Number;
var count:Number = 0;
var maxCount:Number = 10;
var duration:Number = 20;

function executeCallback():Void {
 trace("executeCallback intervalId: " + intervalId + " count: " + count);
 if(count >= maxCount) {
 clearInterval(intervalId);
 }
 count++;
}

function beginInterval():Void {
 if(intervalId != null) {
 trace("clearInterval");
 clearInterval(intervalId);
 }
 intervalId = setInterval(this, "executeCallback", duration);
}

beginInterval();
beginInterval();
beginInterval();
```

例 3 : 次の例では、呼び出される関数にカスタム引数を渡す方法を示します。

```
var intervalId:Number;
var count:Number = 0;
var maxCount:Number = 10;
var duration:Number = 20;
var colors:Array = new Array("red",
 "blue",
 "yellow",
 "purple",
 "green",
 "orange",
 "salmon",
 "pink",
 "lilac",
 "powder blue",
 "mint");
```

```
function executeCallback(param:String) {
 trace("executeCallback intervalId: " + intervalId + " count: " + count + "
  param: " + param);
 clearInterval(intervalId);
 if(count < maxCount) {
 count++;
 intervalId = setInterval(this, "executeCallback", duration, colors[count]);
 }
}

if(intervalId != null) {
 clearInterval(intervalId);
}

intervalId = setInterval(this, "executeCallback", duration, colors[count]);
```

例 4 : 次の例では、setInterval() を ActionScript 2.0 カスタムクラスから適切に使用する方法を示します。前の例と同様、this を setInterval() 関数に渡して、呼び出されるメソッドが適切なスコープ内で確実に実行されるようにしています。

```
class CustomClass {
 private var intervalId:Number;
 private var count:Number = 0;
 private var maxCount:Number = 10;
 private var duration:Number = 20;

 public function CustomClass():Void {
 beginInterval();
 }

 private function beginInterval():Void {
 if(intervalId != null) {
 trace("clearInterval");
 clearInterval(intervalId);
 }
 intervalId = setInterval(this, "executeCallback", duration);
 }

 public function executeCallback():Void {
 trace("executeCallback intervalId: " + intervalId + " count: " + count);
 if(count >= maxCount) {
 clearInterval(intervalId);
 }
 count++;
 }
}
```

新規ドキュメントで、新しいクラスの新しいインスタンスをインスタンス化します。

```
var custom:CustomClass = new CustomClass();
```

関連項目

clearInterval 関数,updateAfterEvent 関数,class ステートメント

setProperty 関数

```
setProperty(target:Object, property:Object, expression:Object) : Void
```

ムービークリップの再生時にムービークリップのプロパティ値を変更します。

対応バージョン： ActionScript 1.0、Flash Player 4

パラメータ

target:Object - 設定対象のプロパティがあるムービークリップのインスタンス名へのパス。

property:Object - 設定するプロパティ。

expression:Object - プロパティの新しいリテラル値またはプロパティの新しい値として評価される等式のいずれか一方。

例

次の ActionScript は、新しいムービークリップを作成し、イメージをロードします。setProperty() を使用して、そのクリップの _x 座標と _y 座標を設定します。right_btn というボタンをクリックすると、params_mc というムービークリップの _x 座標が 20 ピクセル増加します。

```
this.createEmptyMovieClip("params_mc", 999);
params_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
setProperty(this.params_mc, _y, 20);
setProperty(this.params_mc, _x, 20);
this.right_btn.onRelease = function() {
 setProperty(params_mc, _x, getProperty(params_mc, _x)+20);
};
```

関連項目

getProperty 関数

setTimeout 関数

setTimeout() : Number

ミリ秒単位で指定した遅延時間の経過後に、指定した関数を実行します。setTimeout() 関数は setInterval() 関数に似ていますが、setTimeout() は関数を一度呼び出した後、自動的に削除される点が異なります。

setTimeout() が関数を呼び出すのを防ぐために clearTimeout() メソッドを引き続き使用できるようにするには、setTimeout() 呼び出しの戻り値を必ず変数に代入してください。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

functionReference:Object - 実行する関数の名前。引用符や括弧を使用しないでください。また、呼び出す関数にパラメータを指定しないでください。たとえば、functionName を使用します。functionName() や functionName(param) は使用しないでください。

delay:Number - 関数が実行されるまでの遅延時間 (ミリ秒単位)。

args:Object - 関数に渡すゼロ個以上の引数 (複数の引数はコンマで区切ります)。

戻り値

Number - 時間のプロセスに対する一意の数値識別子。

例

次の例では、setTimeout() を使用して 2 秒の遅延の後に my_delayedFunction という関数を呼び出します。ユーザーが Escape キーを押した場合には、戻り値を使って clearTimeout() を呼び出します。この例では、my_delayedFunction 呼び出し前にユーザーが Escape キーを押さない限り、2 秒の経過後に "two second delay" というストリングが出力されます。

```
var my_timedProcess:Number = setTimeout(my_delayedFunction, 2000, "two second
  delay");

function my_delayedFunction (arg1) {
 trace(arg1);
}

var escListener:Object = new Object();
escListener.onKeyDown = function() {
 if (Key.isDown(Key.ESCAPE)) {
 clearTimeout(my_timedProcess);
 }
};
Key.addListener(escListener);
```

この例を使用する場合、テスト環境では [制御]-[キーボードショートカットを無効] を選択してください。

関連項目

clearTimeout 関数, setInterval 関数

showRedrawRegions 関数

showRedrawRegions(enable:Boolean, [color:Number]) : Void

デバッガプレーヤーに、再描画されるスクリーン領域 (つまり、問題が発生した更新対象の領域) をアウトラインで囲む機能を提供します。このアウトラインは、[再描画領域を表示] メニューオプションで有効にすることもできます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

enable:Boolean - 領域の再描画を有効にするか (true)、または無効にするか (false) を指定します。true に設定すると、再描画用の矩形が表示されます。false に設定すると、再描画用の矩形がクリアされます。

color:Number (オプション) - 描画に使用する色。デフォルト値は赤 (0xFF0000) です。

例

次の例では、showRedrawRegions 関数を説明します。

```
var w:Number = 100;
var h:Number = 100;

var shape1:MovieClip = createShape("shape1");
shape1.onEnterFrame = function():Void {
 this._x += 5;
 this._y += 5;
}

var shape2:MovieClip = createShape("shape2");
shape2.onEnterFrame = function():Void {
 this._y += 5;
}

_global.showRedrawRegions(true);
```

```
function createShape(name:String):MovieClip {
 var mc:MovieClip = this.createEmptyMovieClip(name, this.getNextHighestDepth());
 mc.beginFill(0xFFCC00);
 mc.moveTo(200, 200);
 mc.curveTo(300, 200, 300, 100);
 mc.curveTo(300, 0, 200, 0);
 mc.curveTo(100, 0, 100, 100);
 mc.curveTo(100, 200, 200, 200);
 mc.endFill();
 return mc;
}
```

startDrag 関数

```
startDrag(target:Object, [lock:Boolean, left:Number, top:Number, right:Number,
bottom:Number]) : Void
```

ムービーの再生中に *target* ムービークリップをドラッグ可能にします。一度に 1 つのムービークリップしかドラッグできません。startDrag() を実行した後は、stopDrag() によって明示的に停止するか、他のムービークリップに対して startDrag() アクションが呼び出されるまで、ムービークリップはドラッグ可能なままになります。

対応バージョン： ActionScript 1.0、Flash Player 4

パラメータ

target: Object - ドラッグするムービークリップのターゲットパス。

lock: Boolean (オプション) - ドラッグ可能なムービークリップがマウス位置の中心にロックされるか (true)、それともユーザーがムービークリップ上で最初にクリックした点にロックされるか (false) を指定するブール値。

left,top,right,bottom: Number (オプション) - ムービークリップの制限矩形を指定するムービークリップの親の座標を基準にした相対値。

例

次の例では、ムービークリップ pic_mc を作成し、そのムービークリップに startDrag() アクションと stopDrag() アクションを追加することで、実行時にユーザーが任意の位置にドラッグできるようにします。イメージは、MovieClipLoader クラスを使用して pic_mc にロードされます。

```
var pic_mcl:MovieClipLoader = new MovieClipLoader();
pic_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
 this.createEmptyMovieClip("pic_mc", this.getNextHighestDepth()));
var listenerObject:Object = new Object();
listenerObject.onLoadInit = function(target_mc) {
 target_mc.onPress = function() {
 startDrag(this);
 };
```

```
 target_mc.onRelease = function() {
 stopDrag();
 };
};
pic_mcl.addListener(listenerObject);
```

関連項目

stopDrag 関数, _droptarget (MovieClip._droptarget プロパティ), startDrag (MovieClip.startDrag メソッド)

stop 関数

```
stop() : Void
```

再生中の SWF ファイルを停止します。このアクションは、ボタンでムービークリップを制御する場合に最もよく使用します。

対応バージョン： ActionScript 1.0、Flash Player 2

関連項目

gotoAndStop 関数, gotoAndStop (MovieClip.gotoAndStop メソッド)

stopAllSounds 関数

```
stopAllSounds() : Void
```

再生ヘッドを停止せずに、SWF ファイルで再生中のサウンドをすべて停止します。ストリーミングするために設定されたサウンドは、そのサウンドが置かれているフレームに再生ヘッドが移動すると再生を再開します。

対応バージョン： ActionScript 1.0、Flash Player 3

例

次のコードを実行するとテキストフィールドが作成され、そこに音楽の ID3 情報が表示されます。新しい Sound オブジェクトインスタンスが作成され、SWF ファイルに MP3 がロードされます。サウンドファイルから ID3 情報が抽出されます。ユーザーが stop_mc, をクリックすると、サウンドが一時停止します。ユーザーが play_mc, をクリックすると、一時停止されていた位置から再生が再開します。

```
this.createTextField("songinfo_txt", this.getNextHighestDepth, 0, 0,
  Stage.width, 22);
var bg_sound:Sound = new Sound();
bg_sound.loadSound("yourSong.mp3", true);
bg_sound.onID3 = function() {
 songinfo_txt.text = "(" + this.id3.artist + ") " + this.id3.album + " - " +
  this.id3.track + " - "
 + this.id3.songname;
 for (prop in this.id3) {
 trace(prop+" = "+this.id3[prop]);
 }
 trace("ID3 loaded.");
};
this.play_mc.onRelease = function() {
 /* get the current offset. if you stop all sounds and click the play button, the
  MP3 continues from
 where it was stopped, instead of restarting from the beginning. */
 var numSecondsOffset:Number = (bg_sound.position/1000);
 bg_sound.start(numSecondsOffset);
};
this.stop_mc.onRelease = function() {
 stopAllSounds();
};
```

関連項目

Sound

stopDrag 関数

```
stopDrag() : Void
```

現在実行中のドラッグ操作を停止します。

対応バージョン： ActionScript 1.0、Flash Player 4

例

次のコードをメインタイムラインに置くと、ユーザーがマウスボタンを離したときにムービークリップインスタンス my_mc でのドラッグアクションが停止します。

```
my_mc.onPress = function () {
 startDrag(this);
}

my_mc.onRelease = function() {
 stopDrag();
}
```

関連項目

startDrag 関数, _droptarget (MovieClip._droptarget プロパティ), startDrag (MovieClip.startDrag メソッド), stopDrag (MovieClip.stopDrag メソッド)

String 関数

String(expression:Object) : String

指定されたパラメータのストリング表現を返します。具体的には次の処理が行われます。

- *expression* が数値である場合、返されるストリングは数値のテキスト表現です。
- *expression* がストリングの場合は、返されるストリングは *expression* です。
- *expression* がオブジェクトである場合、戻り値は、オブジェクトのストリングプロパティを呼び出した結果として生成されるストリング表現です。このようなプロパティが存在しない場合は、Object.toString() を呼び出した結果として生成されるストリング表現です。
- *expression* がブール値の場合、返されるストリングは "true" または "false" です。
- *expression* がムービークリップである場合、戻り値はスラッシュ (/) 表記のムービークリップのターゲットパスです。

expression が undefined である場合は、戻り値は次のようになります。

- Flash Player 6 以前用にパブリッシュされたファイルでは、結果は空のストリング ("") になります。
- Flash Player 7 以降用にパブリッシュされたファイルでは、結果は undefined になります。

メモ : スラッシュ表記は、ActionScript 2.0 ではサポートされていません。

対応バージョン : ActionScript 1.0、Flash Player 4 - Flash Player 7 ではビヘイビアが変更されました。

パラメータ

expression:Object - ストリングに変換される式。

戻り値

String - ストリング。

例

次の例では ActionScript を使用して、指定された式をストリングに変換します。

```
var string1:String = String("3");
var string2:String = String("9");
trace(string1+string2); // 39
```

どちらのパラメータもストリングの場合、値は加算されるのではなく連結されます。

関連項目

toString (Number.toString メソッド), toString (Object.toString メソッド), String, " ストリング区切り記号演算子

substring 関数

```
substring(string:String, index:Number, count:Number) : String
```

非推奨 Flash Player 5 以降では使用しないでください。この関数の代わりに String.substr( ) を使用します。

ストリングの一部を抽出します。この関数は 1 から始まります。一方、String オブジェクトのメソッドは 0 から始まります。

対応バージョン： ActionScript 1.0、Flash Player 4

パラメータ

string:String - 新しいストリングを抽出する元のストリング。

index:Number - 抽出する最初の文字の位置を表す数値。

count:Number- 抽出されるストリングに含まれる文字数。インデックス文字は除きます。

戻り値

String - 抽出されたサブストリング。

関連項目

substr (String.substr メソッド)

targetPath 関数

```
targetPath(targetObject:Object) : String
```

MovieClip、Button、TextField、または Video オブジェクトのターゲットパスをストリングとして返します。ターゲットパスは、ドット (.) 表記で返されます。スラッシュ (/) 表記でターゲットパスを検索するには、_target プロパティを使用します。

対応バージョン： ActionScript 1.0、Flash Player 5 - Flash Player 6 で、Button オブジェクト、TextField オブジェクト、および Video オブジェクトのサポートが追加されました。

パラメータ

targetObject:Object - 検索するターゲットパスのオブジェクトへの参照 (_root や _parent など)。MovieClip、Button、または TextField オブジェクトなどを指定できます。

戻り値

String - 指定したオブジェクトのターゲットパスを含むストリング。

例

次の例では、ムービークリップをロードすると同時にそのターゲットパスを表示します。

```
this.createEmptyMovieClip("myClip_mc", this.getNextHighestDepth());
trace(targetPath(myClip_mc)); // _level0.myClip_mc
```

関連項目

eval 関数

tellTarget 関数

```
tellTarget(target:String) {
statement(s);
}
```

Flash Player 5 以降では、**使用しないでください**。ドット (.) 表記と with ステートメントを使用することをお勧めします。

statements パラメータで指定された指示を、*target* パラメータで指定されたタイムラインに適用します。tellTarget アクションは、ナビゲーションコントロールに使用すると便利です。ステージ上のムービークリップを停止または開始するボタンに、tellTarget を割り当てます。ムービークリップを、そのクリップ内の特定のフレームにジャンプさせることもできます。たとえば、ボタンに tellTarget を指定して、ステージ上のムービークリップを停止または開始したり、特定のフレームにジャンプしたりするように指示することができます。

Flash 5 以降では、tellTarget アクションの代わりにドット (.) 表記を使用できます。with アクションを使用して同じタイムラインに対して複数のアクションを発行できます。with アクションのターゲットは任意のオブジェクトですが、tellTarget アクションのターゲットはムービークリップのみです。

対応バージョン： ActionScript 1.0、Flash Player 3

パラメータ

target:String - 制御するタイムラインのターゲットパスを指定するストリング。

statement(s) - 条件が true である場合に実行される指示。

例

次の tellTarget ステートメントでは、メインタイムライン上のムービークリップインスタンス ball を制御します。ball インスタンスのフレーム 1 は空白であり、stop() アクションがあるので、ステージ上では見えません。次のアクションを備えたボタンがクリックされると、tellTarget は ball 内の再生ヘッドに対して、アニメーションが始まるフレーム 2 に進むように指示します。

```
on(release) {
tellTarget("_parent.ball") {
gotoAndPlay(2);
}
}
```

次の例では、ドット (.) 表記を使用して同じ結果を得ます。

```
on(release) {
_parent.ball.gotoAndPlay(2);
}
```

ball インスタンスに対して複数のコマンドを発行する場合は、次のステートメントに示すように with アクションを使用できます。

```
on(release) {
with(_parent.ball) {
gotoAndPlay(2);
_alpha = 15;
_xscale = 50;
_yscale = 50;
}
}
```

関連項目

with ステートメント

toggleHighQuality 関数

```
toggleHighQuality()
```

非推奨 Flash Player 5 以降では使用しないでください。この関数の代わりに _quality を使用します。

Flash Player でアンチエイリアス処理のオンとオフを切り替えます。アンチエイリアス処理を行うとオブジェクトのエッジが滑らかになりますが、ムービーの再生は遅くなります。これは、Flash Player のすべての SWF ファイルに影響します。

対応バージョン： ActionScript 1.0、Flash Player 2

例

次のコードは、アンチエイリアス処理のオンとオフをクリックで切り替えるボタンに適用できます。

```
on(release) {
toggleHighQuality();
}
```

関連項目

_highquality プロパティ, _quality プロパティ

trace 関数

```
trace(expression:Object)
```

Flash Debug Player を使用すると、trace() 関数の出力を取得し、その結果を表示できます。trace ステートメント内の引数に String 以外のデータ型が含まれる場合、trace 関数はそのデータ型に関連した toString() メソッドを呼び出します。たとえば、引数がブール値の場合、trace 関数は Boolean.toString() を呼び出して戻り値を表示します。

このステートメントは、ムービーのプレビュー中にプログラミングのメモを記録したり [出力] パネルにメッセージを表示する場合に使用します。条件の有無を確認したり、値を [出力] パネルに表示したりするには、*expression* パラメータを使用します。trace() ステートメントは、JavaScript の alert 関数に似ています。

[パブリッシュ設定] ダイアログボックスで [Trace アクションを省略] コマンドを使用すると、書き出す SWF ファイルから trace() アクションを削除することができます。

対応バージョン : ActionScript 1.0、Flash Player 4

パラメータ

expression:Object - 評価する式。[ムービープレビュー] コマンドを使用して SWF ファイルを Flash オーサリングツールで開くと、*expression* パラメータの値が [出力] パネルに表示されます。

例

次の例では、trace() ステートメントを使用して、動的に作成されるテキストフィールド error_txt のメソッドとプロパティを [出力] パネルに表示します。

```
this.createTextField("error_txt", this.getNextHighestDepth(), 0, 0, 100, 22);
for (var i in error_txt) {
trace("error_txt."+i+" = "+error_txt[i]);
}
/*
error_txt.styleSheet = undefined
error_txt.mouseWheelEnabled = true
error_txt.condenseWhite = false
```

```
...
error_txt.maxscroll = 1
error_txt.scroll = 1
*/
```

unescape 関数

```
unescape(string:String) : String
```

パラメータ *x* をストリングとして評価し、URL エンコードされた形式からストリングをデコード (すべての 16 進シーケンスを ASCII 文字に変換) して、ストリングを返します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`string:String` - アンエスケープする 16 進シーケンスのストリング。

戻り値

`String` - URL エンコードされたパラメータからデコードしたストリング。

例

次の例では、エスケープからアンエスケープへの変換プロセスを示します。

```
var email:String = "user@somedomain.com";
trace(email);
var escapedEmail:String = escape(email);
trace(escapedEmail);
var unescapedEmail:String = unescape(escapedEmail);
trace(unescapedEmail);
```

次の結果が [出力] パネルに表示されます。

```
user@somedomain.com
user%40somedomain%2Ecom
user@somedomain.com
```

unloadMovie 関数

```
unloadMovie(target:MovieClip) : Void
unloadMovie(target:String) : Void
```

`loadMovie()` を使ってロードしたムービークリップを Flash Player から削除します。`loadMovieNum()` を使ってロードしたムービーをアンロードするには、`unloadMovie()` ではなく `unloadMovieNum()` を使用します。

対応バージョン : ActionScript 1.0、Flash Player 3

パラメータ

`target:Object` - ムービークリップのターゲットパス。このパラメータには、String ("my_mc" など)、またはムービークリップインスタンス (my_mc など) への直接参照のいずれかを指定できます。複数のデータ型を指定できるパラメータは、`Object` としてリストされます。

例

次の例では、`pic_mc` という名前で新しいムービークリップを作成し、そのクリップにイメージをロードします。イメージは MovieClipLoader クラスを使用してロードします。イメージをクリックすると、ムービークリップが SWF ファイルからアンロードされます。

```
var pic_mcl:MovieClipLoader = new MovieClipLoader();
pic_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
 this.createEmptyMovieClip("pic_mc", this.getNextHighestDepth()));
var listenerObject:Object = new Object();
listenerObject.onLoadInit = function(target_mc) {
 target_mc.onRelease = function() {
 unloadMovie(pic_mc);
 /* or you could use the following, which refers to the movie clip referenced by
   'target_mc'. */
 //unloadMovie(this);
 };
};
pic_mcl.addListener(listenerObject);
```

関連項目

`loadMovie (MovieClip.loadMovie メソッド )`,
`unloadClip (MovieClipLoader.unloadClip メソッド )`

unloadMovieNum 関数

`unloadMovieNum(level:Number) : Void`

loadMovieNum() を使ってロードした SWF ファイルやイメージを Flash Player から削除します。`MovieClip.loadMovie()` を使用してロードした SWF またはイメージをアンロードするには、`unloadMovieNum()` ではなく `unloadMovie()` を使用します。

対応バージョン : ActionScript 1.0、Flash Player 3

パラメータ

`level:Number` - ロードされたムービーのレベル (`_level N`)。

例

次の例では、SWF ファイルにイメージをロードします。unload_btn をクリックすると、ロードされたコンテンツが削除されます。

```
loadMovieNum("yourimage.jpg", 1);
unload_btn.onRelease = function() {
 unloadMovieNum(1);
}
```

関連項目

loadMovieNum 関数, unloadMovie 関数, loadMovie (MovieClip.loadMovie メソッド)

updateAfterEvent 関数

```
updateAfterEvent() : Void
```

onClipEvent() ハンドラ内で呼び出すか、setInterval() に渡す関数またはメソッドの一部としてこれを呼び出すと、ムービーに設定された FPS 値とは関係なく表示が更新されます。updateAfterEvent の呼び出しを onClipEvent() ハンドラ内から、または setInterval() に渡す関数またはメソッドの一部として行わなかった場合、呼び出しは無視されます。この関数は、特定の Mouse ハンドラおよび MovieClip ハンドラでのみ機能します。Mouse クラスでは mouseDown、mouseUp、mouseMove、keyDown、keyUp ハンドラ、MovieClip クラスでは onMouseMove、onMouseDown、onMouseUp、onKeyDown、onKeyUp ハンドラです。Key クラスでは機能しません。

対応バージョン： ActionScript 1.0、Flash Player 5

例

次の例では、cursor_mc という名前のカスタムカーソルを作成する方法を示します。ActionScript を使用してマウスカーソルを cursor_mc に置き換えます。次に updateAfterEvent() を使用してステージを連続的に更新し、カーソルの動きが滑らかに表示されるようにします。

```
Mouse.hide();
cursor_mc.onMouseMove = function() {
 this._x = this._parent._xmouse;
 this._y = this._parent._ymouse;
 updateAfterEvent();
};
```

関連項目

onClipEvent ハンドラ, setInterval 関数

グローバルプロパティ

グローバルプロパティは、どのスクリプトでも使用でき、ドキュメント内のどのタイムラインやスコープでも参照できます。たとえば、グローバルプロパティを使用して、ロードされた他のムービークリップのタイムラインに相対アクセス (_parent) または絶対アクセス (_root) できます。また、スコープを制限 (this) または拡張 (super) することもできます。さらに、グローバルプロパティを使用して、スクリーンリーダーのアクセシビリティ、再生品質、サウンドバッファサイズなどの実行時設定を調整ができます。

グローバルプロパティー覧

オプション	プロパティ	説明
	_accProps	SWF ファイル、ムービークリップ、ボタン、ダイナミックテキストフィールド、およびテキスト入力フィールドについて、スクリーンリーダーのアクセシビリティオプションを実行時に制御できます。
	_focusrect	プロパティ (グローバル)。キーボードフォーカスがあるボタンまたはムービークリップの周囲に黄色の矩形を表示するかどうかを指定します。
	_global	String、Object、Math、Array などのコア ActionScript クラスを保持するグローバルオブジェクトへの参照です。
	_highquality	**非推奨** Flash Player 5 以降では使用しないでください。このプロパティの代わりに _quality を使用します。現在の SWF ファイルに適用されるアンチエイリアスのレベルを指定します。
	_level	_level *N* のルートタイムラインへの参照です。
	maxscroll	**非推奨** Flash Player 5 以降では使用しないでください。このプロパティの代わりに TextField.maxscroll を使用します。 テキストフィールド内の最終行も表示されている場合に、そのフィールドの先頭に表示されるテキストの行番号を示します。
	_parent	現在のムービークリップまたはオブジェクトを含むムービークリップまたはオブジェクトへの参照を指定するか返します。
	_quality	ムービークリップに使用するレンダリング品質を設定または取得します。
	_root	ルートムービークリップのタイムラインへの参照を指定するか返します。

オプション	プロパティ	説明
	scroll	**非推奨** Flash Player 5 以降では使用しないでください。このプロパティの代わりに TextField.scroll を使用します。 変数に関連するテキストフィールドの情報表示を制御します。
	_soundbuftime	ストリーミングサウンドをバッファする秒数を設定します。
	this	オブジェクトインスタンスまたはムービークリップインスタンスを参照します。

_accProps プロパティ

_accProps.*propertyName*
instanceName._accProps.*propertyName*

SWF ファイル、ムービークリップ、ボタン、ダイナミックテキストフィールド、およびテキスト入力フィールドについて、スクリーンリーダーのアクセシビリティオプションを実行時に制御できます。オーサリング時に、これらのプロパティは [アクセシビリティ] パネルの対応する設定よりも優先されます。これらのプロパティに対する変更を有効にするには、Accessibility.updateProperties() を呼び出す必要があります。

[アクセシビリティ] パネルの詳細については、『Flash ユーザーガイド』の「Flash の [アクセシビリティ] パネル」を参照してください。

アクセシビリティ補助をサポートする環境で Player が実行されるかどうかを確認するには、System.capabilities.hasAccessibility プロパティを使用します。

次の表に、各 _accProps プロパティの名前とデータ型、[アクセシビリティ] パネルの対応する設定、およびプロパティを適用できるオブジェクトの種類を示します。"逆ロジック" という用語は、プロパティ設定が [アクセシビリティ] パネルの対応する設定とは逆になることを示します。たとえば、silent プロパティを true に設定することは、[ムービーをアクセス可能にする] オプションまたは [オブジェクトをアクセス可能にする] オプションの選択を解除することに相当します。

プロパティ	データ型	[アクセシビリティ] パネルの対応する設定	適用対象
silent	Boolean	ムービーをアクセス可能にする / オブジェクトをアクセス可能にする (逆ロジック)	SWF ファイル全体、ムービークリップ、ボタン、ダイナミックテキスト、テキスト入力
forceSimple	ブール値	子オブジェクトをアクセス可能にする (逆ロジック)	SWF ファイル全体およびムービークリップ

プロパティ	データ型	[アクセシビリティ] パネルの対応する設定	適用対象
name	String	名前	SWF ファイル全体、ムービークリップ、ボタン、テキスト入力
description	String	説明	SWF ファイル全体、ムービークリップ、ボタン、ダイナミックテキスト、テキスト入力
shortcut	String	ショートカット	ムービークリップ、ボタン、テキスト入力

[ショートカット] フィールドでは Control+A という形式の名前を使用します。[アクセシビリティ] パネルにキーボードショートカットを追加しても、スクリーンリーダーに既存のショートカットが通知されるだけで、キーボードショートカットは作成されません。アクセス可能なオブジェクトにキーボードショートカットを割り当てる方法の詳細については、Key.addListener() を参照してください。

[アクセシビリティ] パネルのタブインデックスに対応する設定を指定するには、Button.tabIndex、MovieClip.tabIndex、または TextField.tabIndex のいずれかのプロパティを使用します。

実行時に [自動ラベル] 設定を指定することはできません。

Flash ドキュメント全体を表す _accProps オブジェクトを参照するには、instanceName パラメータを省略します。_accProps の値はオブジェクトであることが必要です。つまり、_accProps オブジェクトが存在しない場合は、_accProps オブジェクトのプロパティに値を代入する前に、オブジェクトを作成する必要があります。次にその例を示します。

```
if ( _accProps == undefined )
{
 _accProps = new Object();
}
_accProps.name = "My SWF file";
```

_accProps に *instanceName* パラメータを指定しなかった場合、_accProps プロパティに加えた変更は SWF ファイル全体に適用されます。たとえば、次のコードでは、SWF ファイル全体に対し、Accessibility の name プロパティに "Pet Store" というストリングを設定し、Accessibility.updateProperties() を呼び出してこの変更内容を有効にしています。

```
_accProps.name = "Pet Store";
Accessibility.updateProperties();
```

これとは対照的に、次のコードでは、インスタンス名が price_mc のムービークリップについて、name プロパティに "Price" というストリングを設定しています。

```
price_mc._accProps.name = "Price";
Accessibility.updateProperties();
```

複数のアクセシビリティプロパティを指定する場合、Accessibility.updateProperties() は最後に 1 回だけ呼び出すだけでよく、各プロパティステートメントごとに呼び出す必要はありません。次に例を示します。

```
_accProps.name = "Pet Store";

animal_mc._accProps.name = "Animal";
animal_mc._accProps.description = "Cat, dog, fish, etc.";

price_mc._accProps.name = "Price";
price_mc._accProps.description = "Cost of a single item";

Accessibility.updateProperties();
```

ドキュメントまたはオブジェクトのアクセシビリティプロパティを指定しない場合、[アクセシビリティ] パネルで設定されたすべての値が適用されます。

アクセシビリティプロパティを指定した後で、その値を [アクセシビリティ] パネルで設定された値に戻すことはできません。ただし、_accProps オブジェクトからプロパティを削除することにより、プロパティをデフォルト値 (ブール値の場合は false、ストリング値の場合は空ストリング) に設定できます。次にその例を示します。

```
my_mc._accProps.silent = true; // set a property
// other code here
delete my_mc._accProps.silent; // revert to default value
```

対応バージョン： ActionScript 1.0、Flash Player 6,0,65,0

パラメータ

propertyName:Boolean or String - アクセシビリティプロパティ名 (有効な名前については次の説明を参照)。*instanceName*

instanceName:String - ムービークリップ、ボタン、ダイナミックテキストフィールド、またはテキスト入力フィールドのインスタンスに割り当てられたインスタンス名。Flash ドキュメント全体を表す _accProps オブジェクトを参照するには、*instanceName* を省略します。

例

イメージを変更して、アクセシビリティオブジェクトの説明を更新する場合は、次のような ActionScript コードを使用します。

```
my_mc.gotoAndStop(2);

if (my_mc._accProps == undefined ) {
 my_mc._accProps = new Object();
}

my_mc._accProps.name = "Photo of Mount Rushmore";
Accessibility.updateProperties();
```

関連項目

isActive (Accessibility.isActive メソッド), updateProperties (Accessibility.updateProperties メソッド), hasAccessibility (capabilities.hasAccessibility プロパティ)

_focusrect プロパティ

```
_focusrect = Boolean;
```

キーボードフォーカスがあるボタンまたはムービークリップの周囲に黄色の矩形を表示するかどうかを指定します。_focusrect がデフォルト値 true に設定されている場合、ユーザーが Tab キーを押して SWF ファイル内のオブジェクト間を移動するときに、現在フォーカスのあるボタンまたはムービークリップの周囲に黄色の矩形が表示されます。黄色の矩形を表示しない場合は、false を指定します。これはグローバルプロパティであり、特定のインスタンスに対して上書きできます。

グローバル _focusrect プロパティが false に設定されている場合、すべてのボタンとムービークリップのデフォルトのビヘイビアでは、キーボードナビゲーションが Tab キーに限定されます。Enter キーや矢印キーを含め、他のキーはすべて無視されます。すべてのキーナビゲーションを回復させるには、_focusrect を true に設定する必要があります。特定のボタンやムービークリップのキー操作機能をすべて回復させる場合は、Button._focusrect と MovieClip._focusrect のいずれを使用しても、このグローバルプロパティを上書きできます。

メモ： コンポーネントを使用する場合、FocusManager は、このグローバルプロパティの使用を含め、Flash Player のフォーカス処理をオーバーライドします。

対応バージョン： ActionScript 1.0、Flash Player 4

例

次の例では、ブラウザウィンドウでフォーカスを受け取ったときに、SWF ファイル内のインスタンスの周囲の黄色の矩形を非表示にします。ボタンまたはムービークリップをいくつか作成し、次の ActionScript をタイムラインのフレーム 1 に追加します。

```
_focusrect = false;
```

パブリッシュ設定を Flash Player 6 に変更し、[ファイル]-[パブリッシュプレビュー]-[HTML] を選択してブラウザウィンドウで SWF ファイルをテストします。ブラウザウィンドウ内で SWF をクリックしてフォーカスを与え、Tab キーを使ってそれぞれのインスタンスにフォーカスを移動します。_focusrect が無効になっている場合は、Enter キーやスペースキーを押しても onRelease イベントハンドラは呼び出されません。これは、_focusrect が有効になっているか true のときに実行されます。

関連項目

_focusrect (Button._focusrect プロパティ), _focusrect (MovieClip._focusrect プロパティ)

_global プロパティ

```
_global.identifier
```

String、Object、Math、Array などのコア ActionScript クラスを保持するグローバルオブジェクトへの参照です。たとえば、Math オブジェクトや Date オブジェクトのように、グローバル ActionScript オブジェクトとして公開されるライブラリを作成できます。タイムライン宣言またはローカル宣言した変数および関数とは異なり、グローバル変数および関数は SWF ファイルのすべてのタイムラインおよびスコープに公開されます。ただし、ネストされているスコープで同じ名前の識別子により隠蔽される場合を除きます。

メモ： グローバル変数の値を設定するときには、変数の完全修飾名 (たとえば _global.variableName) を使用する必要があります。そうしない場合、同じ名前のローカル変数が作成されて、設定対象のグローバル変数の識別が曖昧になります。

戻り値 String、Object、Math、Array などのコア ActionScript クラスを保持するグローバルオブジェクトへの参照です。

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の例では、SWF ファイルのすべてのタイムラインとスコープで使用できるトップレベル関数 `factorial()`, を作成します。

```
_global.factorial = function(n:Number) {
 if(n <= 1) {
 return 1;
 }
 else {
 return n * factorial(n - 1);
 }
}

trace(factorial(1)); // 1
trace(factorial(2)); // 2
trace(factorial(3)); // 6
trace(factorial(4)); // 24
```

次の例では、グローバル変数の値を設定するときに完全修飾変数名を誤って使用すると、予期しない結果がどのように発生するかを示します。

```
_global.myVar = "globalVariable";
trace(_global.myVar); // globalVariable
trace(myVar); // globalVariable

myVar = "localVariable";
trace(_global.myVar); // globalVariable
trace(myVar); // localVariable
```

関連項目

var ステートメント , set variable ステートメント

_highquality プロパティ

```
_highquality
```

非推奨 Flash Player 5 以降では使用しないでください。このプロパティの代わりに _quality を使用します。

現在の SWF ファイルに適用されるアンチエイリアスのレベルを指定します。ビットマップスムージングを常にオンにして最高品質を適用するには、2 (最高品質) を指定します。アンチエイリアス処理を適用するには、1(高品質) を指定します。SWF ファイルにアニメーションが含まれない場合は、ビットマップは滑らかになります。アンチエイリアス処理を避けるには、0 (低品質) を指定します。

対応バージョン : ActionScript 1.0、Flash Player 4

例

次の ActionScript はメインタイムラインに指定します。アニメーション以外のファイルに対してビットマップスムージングが適用されるように、グローバルな quality プロパティを設定します。

```
_highquality = 1;
```

関連項目

_quality プロパティ

_level プロパティ

_level*N*

_level *N* のルートタイムラインへの参照です。_level プロパティを使用してターゲットとする SWF ファイルは、loadMovieNum() を使用して Flash Player 内に事前にロードしておく必要があります。_level *N* を使用すると、ロードされている SWF ファイルを *N* によって割り当てられたレベルのターゲットにすることもできます。

Flash Player のインスタンス内にロードした最初の SWF ファイルは、自動的に _level0 内にロードされます。_level0 の SWF ファイルによって、その後にロードするすべての SWF ファイルのフレームレート、背景色、およびフレームサイズが設定されます。次に、SWF ファイルは _level0 の SWF ファイルより高い番号のレベルに積み重ねられます。

Flash Player 内にロードする SWF ファイルごとに、loadMovieNum() アクションを使用してレベルを割り当てる必要があります。レベルは任意の順番で割り当てることができます。SWF ファイルが既に含まれているレベル (_level0 を含む) を割り当てると、そのレベルの SWF ファイルはアンロードされ、新しい SWF ファイルに置き換えられます。

対応バージョン： ActionScript 1.0、Flash Player 4

例

次の例では、_level9 にロードされた SWF ファイル sub.swf のメインタイムラインで再生ヘッドを停止します。sub.swf ファイルにはアニメーションが含まれています。このファイルは、次の ActionScript を含むドキュメントと同じディレクトリにあります。

```
loadMovieNum("sub.swf", 9);
myBtn_btn.onRelease = function() {
 _level9.stop();
};
```

上記の例の _level9.stop() は、次のコードで置き換えることができます。

```
_level9.gotoAndStop(5);
```

このアクションは、再生ヘッドを停止する代わりに、_level9 の SWF ファイルのメインタイムラインからフレーム 5 に再生ヘッドを送ります。

関連項目

loadMovie 関数, swapDepths (MovieClip.swapDepths メソッド)

maxscroll プロパティ

variable_name.maxscroll

非推奨 Flash Player 5 以降では使用しないでください。このプロパティの代わりに TextField.maxscroll を使用します。

テキストフィールド内の最終行も表示されている場合に、そのフィールドの先頭に表示されるテキストの行番号を示します。maxscroll プロパティは、scroll プロパティと連携してテキストフィールド内の情報表示を制御します。このプロパティは読み取り専用であり、変更できません。

対応バージョン： ActionScript 1.0、Flash Player 4

関連項目

maxscroll（TextField.maxscroll プロパティ）, scroll（TextField.scroll プロパティ）

_parent プロパティ

_parent.*property*
_parent._parent.*property*

現在のムービークリップまたはオブジェクトを含むムービークリップまたはオブジェクトへの参照を指定するか返します。現在のオブジェクトとは、_parent を参照する ActionScript コードがあるオブジェクトです。現在のムービークリップまたはオブジェクトの上位のムービークリップまたはオブジェクトへの相対パスを指定するには、_parent を使用します。

対応バージョン： ActionScript 1.0、Flash Player 5

例

次の例では、square_mc というインスタンス名のムービークリップがステージにあります。このムービークリップ内には、circle_mc というインスタンス名の別のムービークリップがあります。次の ActionScript では、円をクリックすると circle_mc 親インスタンス (square_mc) を変更できます。相対アドレスを使用する場合 (_root の代わりに _parent を使用) は、最初に [アクション] パネルの [ターゲットパスを挿入] ボタンを使用する方が簡単です。

```
this.square_mc.circle_mc.onRelease = function() {
 this._parent._alpha -= 5;
};
```

関連項目

_root プロパティ , targetPath 関数

_quality プロパティ

_quality:*String*

ムービークリップに使用するレンダリング品質を設定または取得します。デバイスフォントは常にエイリアス処理されるため、_quality プロパティには影響されません。

_quality プロパティは、次の表に示す値に設定できます。

値	説明	グラフィックのアンチエイリアス	ビットマップスムージング
"LOW"	低いレンダリング品質。	グラフィックスはアンチエイリアス処理されていません。	ビットマップはスムージングされません。
"MEDIUM"	普通のレンダリング品質。この設定は、テキストを含まないムービーに適しています。	グラフィックスは 2 x2 ピクセルグリッドを使用してアンチエイリアス処理されます。	Flash Player 8 では、ビットマップは、MovieClip.attachBitmap() 呼び出しおよび MovieClip.beginBitmapFill() 呼び出しで使用される smoothing パラメータに基づいてスムージングされます。 Flash Player 6 および 7 では、ビットマップはスムージングされません。
"HIGH"	高いレンダリング品質。デフォルトのレンダリング品質設定です。	グラフィックスは 4 x4 ピクセルグリッドを使用してアンチエイリアス処理されます。	Flash Player 8 では、ビットマップは、MovieClip.attachBitmap() 呼び出しおよび MovieClip.beginBitmapFill() 呼び出しで使用される smoothing パラメータに基づいてスムージングされます。 Flash Player 6 および 7 では、ムービークリップが静的なものである場合、ビットマップはスムージングされます。
"BEST"	非常に高いレンダリング品質。	グラフィックスは 4 x4 ピクセルグリッドを使用してアンチエイリアス処理されます。	Flash Player 8 では、ビットマップは、MovieClip.attachBitmap() 呼び出しおよび MovieClip.beginBitmapFill() 呼び出しで使用される smoothing パラメータに基づいてスムージングされます。 smoothing を "BEST" に設定した場合、平均アルゴリズムを使ってムービークリップを縮小すると、より高い品質でレンダリングされます。この場合、レンダリング処理に時間はかかりますが、大きなイメージのサムネールなどを高品質で表示できます。 Flash Player 6 および 7 では、ビットマップは常にスムージングされます。

対応バージョン： ActionScript 1.0、Flash Player 5

例

次の例では、レンダリング品質を LOW に設定します。

```
_quality = "LOW";
```

_root プロパティ

```
_root.movieClip
_root.action
_root.property
```

ルートムービークリップのタイムラインへの参照を指定するか返します。ムービークリップに複数のレベルがある場合、ルートムービークリップのタイムラインは現在実行中のスクリプトを含むレベルにあります。たとえば、レベル 1 のスクリプトの評価が _root であれば、_level1 が返されます。

_root を指定することは、使用が推奨されないスラッシュ表記 (/) を使って現在のレベルでの絶対パスを指定することと同じです。

メモ： _root を含むムービークリップを他のムービークリップにロードすると、_root は _root を含むムービークリップのタイムラインではなく、ロードする側のムービークリップのタイムラインを指すようになります。他のムービークリップにロードする場合でも、_root がロードされるムービークリップ自身のタイムラインを指すようにするには、MovieClip._lockroot を使用します。

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

`movieClip:String` - ムービークリップのインスタンス名。

`action:String` - アクションまたはメソッド。

`property:String` - MovieClip オブジェクトのプロパティ。

例

次の例では、現在実行中のスクリプトを含むレベルのタイムラインを停止します。

```
_root.stop();
```

次の例では、_root のスコープ内の変数とインスタンスを出力します。

```
for (prop in _root) {
 trace("_root."+prop+" = "+_root[prop]);
}
```

関連項目

_lockroot (MovieClip._lockroot プロパティ), _parent プロパティ , targetPath 関数

scroll プロパティ

textFieldVariableName.scroll = *x*

非推奨 Flash Player 5 以降では使用しないでください。このプロパティの代わりに TextField.scroll を使用します。

変数に関連するテキストフィールドの情報表示を制御します。scroll プロパティは、テキストフィールドにおける内容の表示開始位置を定義します。設定後、Flash Player はユーザーがテキストフィールドをスクロールするたびに、内容を更新します。scroll プロパティは、長い文節内の特定の段落にユーザーを誘導したり、スクロールテキストフィールドを作成したりする場合に便利です。このプロパティは、取得および変更が可能です。

対応バージョン： ActionScript 1.0、Flash Player 4

例

次の例は、テキストフィールド myText をスクロールする UP ボタンに割り当てるコードです。

```
on (release) {
myText.scroll = myText.scroll + 1;
}
```

関連項目

maxscroll (TextField.maxscroll プロパティ),scroll (TextField.scroll プロパティ)

_soundbuftime プロパティ

_soundbuftime:*Number* = *integer*

ストリーミングサウンドをバッファする秒数を設定します。デフォルト値は 5 秒です。

対応バージョン： ActionScript 1.0、Flash Player 4

パラメータ

integer:Number - SWF ファイルのストリーミングが開始するまでの秒数。

例

次の例では、mp3 ファイルをストリーミングし、サウンドをバッファした後でユーザーに対して再生します。実行時にテキストフィールドが 2 つ作成され、タイマー情報とデバッグ情報が収められます。_soundbuftime プロパティは、MP3 を 10 秒間バッファするように設定されます。MP3 用に新しい Sound オブジェクトインスタンスが作成されます。

```
// create text fields to hold debug information.
this.createTextField("counter_txt", this.getNextHighestDepth(), 0, 0, 100, 22);
this.createTextField("debug_txt", this.getNextHighestDepth(), 0, 20, 100, 22);
// set the sound buffer to 10 seconds.
_soundbuftime = 10;
// create the new sound object instance.
var bg_sound:Sound = new Sound();
// load the MP3 sound file and set streaming to true.
bg_sound.loadSound("yourSound.mp3", true);
// function is triggered when the song finishes loading.
bg_sound.onLoad = function() {
 debug_txt.text = "sound loaded";
};
debug_txt.text = "sound init";
function updateCounter() {
 counter_txt.text++;
}
counter_txt.text = 0;
setInterval(updateCounter, 1000);
```

this プロパティ

```
this
```

オブジェクトインスタンスまたはムービークリップインスタンスを参照します。スクリプトの実行時には、this はそのスクリプトを含むムービークリップインスタンスを参照します。メソッドの実行時には、this には呼び出されたメソッドを含むオブジェクトへの参照が入ります。

ボタンに割り当てられた on() イベントハンドラ内では、this はそのボタンを含むタイムラインを参照します。ムービークリップに割り当てられている onClipEvent() イベントハンドラ内では、this はそのムービークリップ自体のタイムラインを参照します。

this は、それが指定されたスクリプトのコンテキスト内で評価されます。したがって、クラスファイルで定義された変数については、スクリプト内で this を指定して参照することはできません。

対応バージョン： ActionScript 1.0、Flash Player 5

例

"ApplyThis.as" という名前の ActionsScript ファイルを作成し、次のコードを入力します。

```
class ApplyThis {
 var str:String = "Defined in ApplyThis.as";
 function conctStr(x:String):String {
 return x+x;
 }
 function addStr():String {
 return str;
 }
}
```

次に、FLA ファイルまたは別の ActionScript ファイル内に、次のコードを入力します。

```
var obj:ApplyThis = new ApplyThis();
var abj:ApplyThis = new ApplyThis();
abj.str = "defined in FLA or AS";
trace(obj.addStr.call(abj, null)); //output: defined in FLA or AS
trace(obj.addStr.call(this, null)); //output: undefined
trace(obj.addStr.call(obj, null)); //output: Defined in applyThis.as
```

同様に、ダイナミッククラスで定義された関数を呼び出すには、this を使用して適切なスコープで関数を呼び出す必要があります。

```
// incorrect version of Simple.as
/*
dynamic class Simple {
 function callfunc() {
 trace(func());
 }
}
*/
// correct version of Simple.as
dynamic class simple {
 function callfunc() {
 trace(this.func());
 }
}
```

FLA ファイルまたは別の ActionScript ファイル内に、次のコードを入力します。

```
var obj:Simple = new Simple();
obj.num = 0;
obj.func = function() {
 return true;
};
obj.callfunc();
// output: true
```

上記のコードは、this を callfunc() メソッドで使用した場合に機能します。ただし、不適切なバージョンの Simple.as を使用するとシンタックスエラーが発生し、上記の例ではコメントアウトされます。

次の例では、キーワード this は Circle オブジェクトを参照します。

```
function Circle(radius:Number):Void {
 this.radius = radius;
 this.area = Math.PI*Math.pow(radius, 2);
}
var myCircle = new Circle(4);
trace(myCircle.area);
```

ムービークリップ内のフレームに割り当てられた次のステートメントでは、キーワード this は現在のムービークリップを参照します。

```
// sets the alpha property of the current movie clip to 20
this._alpha = 20;
```

MovieClip.onPress ハンドラ内の次のステートメントでは、キーワード this は現在のムービークリップを参照します。

```
this.square_mc.onPress = function() {
 startDrag(this);
};
this.square_mc.onRelease = function() {
 stopDrag();
};
```

関連項目

on ハンドラ, onClipEvent ハンドラ

演算子

記号演算子は式の値の組み合わせ、比較、または修正の方法を指定する文字です。

演算子一覧

演算子	説明
+ (addition)	数値式を加算するか、ストリングを連結 (結合) します。
+= (addition assignment)	*expression1* に *expression1* + *expression2* の値を代入します。
[] (array access)	指定されたエレメント (*a0* など) を使って新しい配列または多次元配列を初期化するか、配列内のエレメントにアクセスします。
= (assignment)	*expression2* の値 (右側のパラメータ) を *expression1* の変数、配列エレメント、またはプロパティに代入します。
& (bitwise AND)	*expression1* と *expression2* を 32 ビット符号なし整数に変換し、整数パラメータをビット単位で論理積 (AND) 演算します。

演算子	説明
&= (bitwise AND assignment)	*expression1* に *expression1* & *expression2* の値を代入します。
<< (bitwise left shift)	*expression1* と *expression2* を 32 ビット整数に変換し、*expression2* の変換により生成された整数で指定される桁数だけ *expression1* 内のすべてのビットを左にシフトします。
<<= (bitwise left shift and assignment)	この演算子はビット単位の左シフト (<<=) 演算を行い、その内容を結果として *expression1* に格納します。
~ (bitwise NOT)	1 の補数演算子またはビット単位の補数演算子とも呼ばれます。
| (bitwise OR)	*expression1* と *expression2* を 32 ビット符号なし整数に変換し、*expression1* と *expression2* の対応ビットの少なくとも一方が 1 である各ビット位置に 1 を返します。
|= (bitwise OR assignment)	*expression1* に *expression1* | *expression2* の値を代入します。
>> (bitwise right shift)	*expression1* と *expression2* を 32 ビット整数に変換し、*expression2* の変換により生成された整数で指定される桁数だけ *expression1* 内のすべてのビットを右にシフトします。
>>= (bitwise right shift and assignment)	この演算子はビット単位の右シフト演算を行い、その内容を結果として *expression1* に格納します。
>>> (bitwise unsigned right shift)	左側のビットには常に 0 が詰められるために元の *expression* の符号が保持されないという点を除いて、ビット単位の右シフト (>>) 演算子と同じです。浮動小数点数は、小数点以下が切り捨てられて整数に変換されます。
>>>= (bitwise unsigned right shift and assignment)	ビット単位の符号なし右シフト演算を行い、その内容を結果として *expression1* に格納します。
^ (bitwise XOR)	*expression1* と *expression2* を 32 ビット符号なし整数に変換し、*expression1* と *expression2* の対応ビットのいずれか一方のみが 1 である各ビット位置に 1 を返します。
^= (bitwise XOR assignment)	*expression1* に *expression1* ^ *expression2* の値を代入します。
/*..*/ (block comment delimiter)	スクリプトコメントのブロックを示します。
, (comma)	式 *expression1*、式 *expression2*、... の順に評価します。

演算子	説明
add (concatenation (strings))	**非推奨** Flash Player 5 以降では使用しないでください。Flash Player 5 以降用のコンテンツを作成する場合は、加算 (+) 演算子を使用することをお勧めします。この演算子は Flash Player 8 以降ではサポートされていません。複数のストリングを連結します。
?: (conditional)	*expression1* を評価し、*expression1* の値が true である場合は *expression2* の値を返します。それ以外の場合は *expression3* の値を返します。
-- (decrement)	*expression* から 1 を引くプリデクリメント単項演算子またはポストデクリメント単項演算子です。
/ (division)	*expression1* を *expression2* で除算します。
/= (division assignment)	*expression1* に *expression1 / expression2* の値を代入します。
. (dot)	ネストされた子のムービークリップ、変数、またはプロパティにアクセスするためにムービークリップの階層をナビゲートする場合に使用します。
== (equality)	2 つの式の等価性をテストします。
eq (equality (strings))	**非推奨** Flash Player 5 以降では使用しないでください。この演算子の代わりに == (equality) 演算子を使用します。 *expression1* のストリング表現が *expression2* のストリング表現と等しい場合は true を返します。それ以外の場合は false を返します。
> (greater than)	2 つの式を比較し、*expression1* が *expression2* より大きいかどうかを判定します。大きい場合は true を返します。
gt (greater than (strings))	**非推奨** Flash Player 5 以降では使用しないでください。この演算子の代わりに > (より大きい) 演算子を使用します。 *expression1* のストリング表現を *expression2* のストリング表現と比較し、*expression1* が *expression2* より大きい場合は true を返します。それ以外の場合は false を返します。
>= (greater than or equal to)	2 つの式を比較し、*expression1* が *expression2* より大きいか等しい場合は true、*expression1* が *expression2* より小さい場合は false と判定します。
ge (greater than or equal to (strings))	**非推奨** Flash Player 5 以降では使用しないでください。この演算子の代わりに >= (より大きいか等しい) 演算子を使用します。 *expression1* が *expression2* より大きいか等しい場合は true を返します。それ以外の場合は false を返します。
++ (increment)	*expression* に 1 を加えるプリインクリメント単項演算子またはポストインクリメント単項演算子です。
!= (inequality)	等価 (==) 演算子の正反対が真であるかどうかをテストします。

演算子	説明
<> (inequality)	**非推奨** Flash Player 5 以降では使用しないでください。この演算子は使用されなくなりました。!= (inequality) 演算子を使用することをお勧めします。等価 (==) 演算子の正反対が真であるかどうかをテストします。
instanceof	object が classConstructor のインスタンスまたは classConstructor のサブクラスであるかどうかをテストします。
< (less than)	2 つの式を比較し、*expression1* が *expression2* より小さいかどうかを判定します。小さい場合は true を返します。
lt (less than (strings))	**非推奨** Flash Player 5 以降では使用しないでください。この演算子の代わりに < (より小さい) 演算子を使用します。 expression1 が expression2 よりも小さい場合は true を返します。それ以外の場合は false を返します。
<= (less than or equal to)	2 つの式を比較し、*expression1* が *expression2* より小さいまたは等しいかどうかを判定します。小さいか等しい場合は true を返します。
le (less than or equal to (strings))	**非推奨** Flash Player 5 以降では使用しないでください。この演算子の代わりに <= (より小さいか等しい) 演算子を使用します。 *expression1* が *expression2* より小さいか等しい場合は true を返します。それ以外の場合は false を返します。
// (line comment delimiter)	スクリプトコメントの先頭を示します。
&& (logical AND)	オペランドをブール (論理) として評価し、論理演算を行います。
and (logical AND)	**非推奨** Flash Player 5 以降では使用しないでください。論理積 (AND) (&&) 演算子を使用することをお勧めします。 Flash Player 4 で論理積 (AND) (&&) 演算をします。
! (logical NOT)	変数や式のブール値を反転します。
not (logical NOT)	**非推奨** Flash Player 5 以降では使用しないでください。この演算子の代わりに ! (logical NOT) (等価) 演算子を使用します。 Flash Player 4 で論理否定 (NOT) (!) 演算をします。
|| (logical OR)	オペランドをブール (論理) 値として評価し、論理演算を行います。オペランドがブール値を返す論理式 (等価や非等価、比較などを調べる式) であれば、両式が共に false の場合のみ false を返します。それ以外の場合、つまりどちらか一方の式または両式が true であれば、戻り値は true になります。
or (logical OR)	**非推奨** Flash Player 5 以降では使用しないでください。この演算子の代わりに || (logical OR) 演算子を使用します。 *condition1* と *condition2* を評価し、いずれかの式が true であれば、式全体が true になります。
% (modulo)	*expression1* を *expression2* で割ったときの剰余を計算します。

演算子	説明
%= (modulo assignment)	*expression1* に *expression1* % *expression2* の値を代入します。
* (multiplication)	2 つの数値や式を乗算します。
*= (multiplication assignment)	*expression1* に *expression1* * *expression2* の値を代入します。
new	新しい匿名のオブジェクトを作成し、constructor パラメータで指定された関数を呼び出します。
ne (not equal (strings))	**非推奨** Flash Player 5 以降では使用しないでください。この演算子の代わりに != (inequality) 演算子を使用します。 *expression1* が *expression2* と等しくない場合は true を返します。それ以外の場合は false を返します。
{} (object initializer)	新しいオブジェクトを作成し、指定された *name* と *value* プロパティペアで初期化します。
() (parentheses)	パラメータに対してグループ化演算を実行するか、複数の式を順番に評価します。または、パラメータを囲み、結果をパラメータとしてカッコの外側にある関数に渡します。
=== (strict equality)	2 つの式が等しいかどうかをテストします。厳密な等価 (===) 演算子は、データ型が変換されない点を除いては、等価 (==) 演算子と同じです。
!== (strict inequality)	厳密な等価 (===) 演算子の正反対が真であるかどうかをテストします。
" (string delimiter)	引用符 (") で前後を囲んだ文字はリテラル値を表します。変数、数値、その他の ActionScript エレメントではなく、ストリングと見なされます。
- (subtraction)	符号反転や減算に使用します。
-= (subtraction assignment)	*expression1* に *expression1* - *expression2* の値を代入します。
: (type)	厳密な型指定を行う際に使用します。この演算子では、変数のタイプ、関数の戻り値のタイプ、または関数パラメータのタイプを指定します。
typeof	typeof 演算子は expression を評価し、その式が String、MovieClip、Object、Function、Number、Boolean のいずれの値であるかを示すストリングを返します。
void	void 演算子は式を評価した後、その値を破棄して、undefined を返します。

+ 加算演算子

expression1 + expression2

数値式を加算するか、ストリングを連結 (結合) します。ある式がストリングの場合、他の式はすべてストリングに変換され、連結されます。両方の式が整数の場合、合計は整数になります。いずれかの式または両方の式が浮動小数の場合、合計は浮動小数になります。

対応バージョン : ActionScript 1.0、Flash Player 4 - Flash 4 では、+ は数値演算子としてのみ使用します。Flash Player 5 以降では、+ は、パラメータのデータ型によって、数値演算子またはストリング連結子として使用します。Flash 4 ファイルを Flash 5 以降のオーサリング環境に読み込むと、変換処理が実行され、データ型の整合性が保たれます。次の例では、数値等価比較が格納された Flash 4 ファイルの変換を示します。

Flash 4 ファイル : `x + y`

変換された Flash 5 以降のファイル : `Number(x) + Number(y)`

オペランド

`expression1` - 数値またはストリング。

`expression2 : Number` - 数値またはストリング。

戻り値

`Object` - ストリング、整数、または浮動小数。

例

シンタックス 1: 次の例では、2 つのストリングを連結し、その結果を [出力] パネルに表示します。

```
var name:String = "Cola";
var instrument:String = "Drums";
trace(name + " plays " + instrument); // output: Cola plays Drums
```

シンタックス 2 : このステートメントでは、整数 2 と 3 を加算し、結果の整数 5 を [出力] パネルに表示します。

```
trace(2 + 3); // output: 5
```

このステートメントでは、浮動小数 2.5 と 3.25 を加算し、結果の浮動小数 5.75 を [出力] パネルに表示します。

```
trace(2.5 + 3.25); // output: 5.75
```

シンタックス 3 : 動的フィールドおよびテキスト入力フィールドに関連付けられた変数のデータ型はストリングです。次の例では、変数 `deposit` はステージのテキスト入力フィールドです。ユーザーが預金額を入力すると、スクリプトは `deposit` を `oldBalance` に加算しようとします。ただし、`deposit` は String データ型であるため、スクリプトは変数の値を合計せずに値を連結 (結合して 1 つのストリングを構成) します。

```
var oldBalance:Number = 1345.23;
var currentBalance = deposit_txt.text + oldBalance;
trace(currentBalance);
```

たとえば、ユーザーが預金テキストフィールドに「475」と入力すると、trace() ステートメントは値 4751345.23 を [出力] パネルに表示します。これを修正するには、次のように Number() 関数を使用してストリングを数値に変換します。

```
var oldBalance:Number = 1345.23;
var currentBalance:Number = Number(deposit_txt.text) + oldBalance;
trace(currentBalance);
```

次の例から、文字列式の右側にある数値の合計が計算されないことがわかります。

```
var a:String = 3 + 10 + "asdf";
trace(a); // 13asdf
var b:String = "asdf" + 3 + 10;
trace(b); // asdf310
```

+= 加算後代入演算子

expression1 += *expression2*

expression1 に *expression1* + *expression2* の値を代入します。たとえば、次の 2 つのステートメントは同じ結果になります。

```
x += y;
x = x + y;
```

この演算子は、ストリングの連結も行います。加算 (+) 演算子のすべての規則が、加算後代入 (+=) 演算子に適用されます。

対応バージョン : ActionScript 1.0、Flash Player 4

オペランド

expression1 : Number - 数値またはストリング。

expression2 : Number - 数値またはストリング。

戻り値

Number - 加算した結果。

例

シンタックス 1: 次の例では、+= 演算子をストリング式で使用します。"My name is Gilbert" が [出力] パネルに表示されます。

```
var x1:String = "My name is ";
x1 += "Gilbert";
trace(x1); // output: My name is Gilbert
```

シンタックス 2: 次に、数値に対して加算後代入 (+=) 演算子を使用する例を示します。

```
var x:Number = 5;
var y:Number = 10;
x += y;
trace(x); // output: 15
```

関連項目

+ 加算演算子

[] 配列アクセス演算子

myArray = [*a0, a1,...aN*]
myArray[*i*] = *value*
myObject [*propertyName*]

指定されたエレメント (*a0* など) を使って新しい配列または多次元配列を初期化するか、配列内のエレメントにアクセスします。配列アクセス演算子を使用すると、インスタンス、変数、およびオブジェクト名を動的に設定および取得できます。この演算子を使用してオブジェクトのプロパティにアクセスすることもできます。

シンタックス 1: 配列は、エレメントと呼ばれるプロパティを持つオブジェクトです。各エレメントは、インデックスと呼ばれる番号で識別されます。配列を作成する場合は、エレメントを配列アクセス演算子 ([]) (角括弧) で囲みます。配列には、さまざまなタイプのエレメントを格納できます。たとえば、次の配列 employee には 3 つのエレメントがあります。最初のエレメントは数値、2 番目と 3 番目のエレメントはストリング (引用符内) です。

```
var employee:Array = [15, "Barbara", "Jay"];
```

カッコをネストすると、多次元配列をシミュレートできます。ネスト可能な配列の深さは、256 レベルまでです。次のコードでは、3 つのエレメントで構成される配列 ticTacToe を作成します。各エレメントはそれぞれが 3 つのエレメントで構成される配列になっています。

```
var ticTacToe:Array = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]; // Select Debug > List
  Variables in test mode
// to see a list of the array elements.
```

シンタックス 2 : 各エレメントに直接アクセスするには、エレメントのインデックスを [] (角括弧) で囲みます。また、配列に新しいエレメントを追加したり、既存のエレメントの値を変更または取得することもできます。次の例に示すように、配列の最初のインデックスは常に 0 です。

```
var my_array:Array = new Array();
my_array[0] = 15;
my_array[1] = "Hello";
my_array[2] = true;
```

次の例のように角カッコ ([]) を使用して 4 番目のエレメントを追加できます。

```
my_array[3] = "George";
```

角カッコ ([]) を使用して多次元配列のエレメントにアクセスできます。最初の [] (角カッコ) のセットは元の配列のエレメントを示します。2 番目のセットはネストされた配列のエレメントを示します。次のコード行は、数値 6 を [出力] パネルに表示します。

```
var ticTacToe:Array = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];
trace(ticTacToe[1][2]);// output: 6
```

シンタックス 3 : `eval()` 関数の代わりに配列アクセス演算子 ([]) を使用して、ムービークリップ名の値またはオブジェクトのプロパティを動的に設定および取得できます。次のコード行は、数値 6 を [出力] パネルに表示します。

```
name["mc" + i] = "left_corner";
```

対応バージョン : ActionScript 1.0、Flash Player 4

オペランド

`myArray : Object` - *myArray* 配列の名前。

`a0, a1,...aN : Object` - *a0,a1,...aN* 配列のエレメント。ネストされた配列を含む、すべてのネイティブタイプまたはオブジェクトインスタンスです。

`i : Number` - *i* 0 以上の整数インデックス。

`myObject : Object` - *myObject* オブジェクトの名前。

`propertyName : String` - *propertyName* オブジェクトのプロパティを指定するストリング。

戻り値

`Object` -

シンタックス 1 : 配列への参照。

シンタックス 2 : 配列の値。ネイティブタイプまたはオブジェクトインスタンス (Array インスタンスを含む)。

シンタックス 3 : オブジェクトのいずれかのプロパティ。ネイティブタイプまたはオブジェクトインスタンス (Array インスタンスを含む)。

例

次の例では、新しい空の Array オブジェクトを作成する 2 つの方法を示しています。最初の行で角カッコ ([]) を使用しています。

```
var my_array:Array = [];
var my_array:Array = new Array();
```

次の例では、配列 employee_array を作成し、trace() ステートメントを使用して、エレメントを [出力] パネルに表示します。4 行目で配列内のエレメントを変更し、5 行目では変更後の新しい配列を [出力] パネルに表示しています。

```
var employee_array = ["Barbara", "George", "Mary"];
trace(employee_array); // output: Barbara,George,Mary
employee_array[2] = "Sam";
trace(employee_array); // output: Barbara,George,Sam
```

次の例では、[] (角括弧) 内の式 ("piece" + i) を評価し、その結果を、ムービークリップ my_mc から取得する変数名として使用します。この例では、変数 i はボタンと同じタイムライン上にある必要があります。たとえば、変数 i の値が 5 である場合は、ムービークリップ my_mc の変数 piece5 の値が [出力] パネルに表示されます。

```
myBtn_btn.onRelease = function() {
 x = my_mc["piece"+i];
 trace(x);
};
```

次のコードでは、[] (角カッコ) 内の式を評価し、その結果をムービークリップ name_mc から取得する変数名として使用します。

```
name_mc["A" + i];
```

Flash 4 の ActionScript スラッシュシンタックスの使用法に詳しい場合は、eval() 関数を使用して同じ結果を得ることができます。

```
eval("name_mc.A" & i);
```

次の ActionScript を使用すると、_root スコープ内のすべてのオブジェクトを繰り返し処理できます。デバッグ時に効果的な方法です。

```
for (i in _root) {
 trace(i+": "+_root[i]);
}
```

代入ステートメントの左側で配列アクセス ([]) 演算子を使用して、インスタンス、変数、およびオブジェクト名を動的に設定することもできます。

```
employee_array[2] = "Sam";
```

関連項目

Array, Object, eval 関数

= 代入演算子

expression1 = *expression2*

expression2 の値 (右側のパラメータ) を *expression1* の変数、配列エレメント、またはプロパティに代入します。値による代入と、参照による代入があります。値による代入では、*expression2* の実際の値がコピーされ、*expression1* に格納されます。値による代入を使用した場合、変数には数値またはストリングのリテラルが代入されます。参照による代入では、*expression2* への参照が *expression1* に格納されます。一般に、参照による代入は、new 演算子と組み合わせて使用されます。new 演算子を使用した場合、メモリ内にオブジェクトが作成され、そのオブジェクトが格納されたメモリ位置への参照が変数に代入されます。

対応バージョン : ActionScript 1.0、Flash Player 4 - Flash 4 では、= は数値等価演算子として使用します。Flash 5 以降では、= は代入演算子であり、等価を評価するには == 演算子を使用します。Flash 4 ファイルを Flash 5 以降のオーサリング環境に読み込むと、変換処理が実行され、データ型の整合性が保たれます。

Flash 4 ファイル : `x = y`

変換された Flash 5 以降のファイル : `Number(x) == Number(y)`

オペランド

`expression1` : `Object` - 変数、配列のエレメント、またはオブジェクトのプロパティ。

`expression2` : `Object` - 任意のタイプの値。

戻り値

`Object` - 代入された値、*expression2*。

例

次の例では、値による代入を使用して、値 5 を変数 x に代入しています。

```
var x:Number = 5;
```

次の例では、値による代入を使用して、値 "hello" を変数 x に代入しています。

```
var x:String;
x = " hello ";
```

次の例では、参照による代入を使用して、moonsOfJupiter 変数を作成しています。この変数には、新たに作成された Array オブジェクトへの参照が格納されます。次に値による代入を使用し、"Callisto" という値を、変数 moonsOfJupiter で参照される配列の最初のエレメントにコピーしています。

```
var moonsOfJupiter:Array = new Array();
moonsOfJupiter[0] = "Callisto";
```

次の例では、参照による代入を使用して新しいオブジェクトを作成し、変数 mercury にこのオブジェクトへの参照を格納しています。次に、値による代入を使用して、mercury オブジェクトの diameter プロパティに値 3030 を代入しています。

```
var mercury:Object = new Object(); mercury.diameter = 3030; // in miles
trace (mercury.diameter); // output: 3030
```

次の例は、上記の例の続きです。merkur (mercury のドイツ語) という変数を作成し、その変数に mercury の値を代入しています。これにより、メモリ内の同じオブジェクトを参照する 2 つの変数が作成されました。同じオブジェクトを参照しているので、どちらの変数を使用しても、このオブジェクトのプロパティにアクセスできます。次に、マイルの代わりにキロメートルを使用するように、diameter プロパティを変更することもできます。

```
var merkur:Object = mercury;
merkur.diameter = 4878; // in kilometers
trace (mercury.diameter); // output: 4878
```

関連項目

== 等価演算子

& ビット単位の論理積 (AND) 演算子

expression1 & *expression2*

expression1 と *expression2* を 32 ビット符号なし整数に変換し、整数パラメータをビット単位で論理積 (AND) 演算します。浮動小数点数は、小数点以下が切り捨てられて整数に変換されます。結果は、新しい 32 ビット整数です。

正の整数は 4294967295 (0xFFFFFFFF) を最大値とする符号なし 16 進値に変換されます。最大値より大きい値は、32 ビットを超えないように、変換時に最上位の桁が切り捨てられます。負の値は、2 の補数表現を使用して、-2147483648 (0x800000000) を最小値とする符号なし 16 進値に変換されます。最小値よりも小さい値は、さらに高い精度で 2 の補数に変換されたうえで、最上位の桁が切り捨てられます。

戻り値は符号付きの 2 の補数として解釈されるため、戻り値は -2147483648 から 2147483647 の整数になります。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash 4 では、AND (&) 演算子は、ストリングの結合に使用されました。Flash 5 以降では、AND (&) 演算子は、ビット単位の AND 演算子です。このため、ストリングを連結するには、加算 (+) 演算子を使用する必要があります。AND (&) 演算子が使用された Flash 4 ファイルを Flash 5 以降のオーサリング環境に読み込むと、加算 (+) 演算子を使用するように自動的に更新されます。

オペランド

`expression1 : Number` - 数値。

`expression2 : Number` - 数値。

戻り値

`Number` - ビット演算の結果。

例

次の例では、数値のビット表現を比較し、同じ位置のビットが共に 1 であった場合にのみ、1 を返します。この ActionScript では、13 (2 進数 1101) と 11 (2 進数 1011) を加算し、両方とも 1 の桁について 1 を返します。

```
var insert:Number = 13;
var update:Number = 11;
trace(insert & update); // output : 9 (or 1001 binary)
```

2 つの数値のうち、共に 1 が立っているのは先頭と末尾の桁だけであるため、13 と 11 の演算結果は 9 になります。

次の例は、戻り値の変換の動作を示しています。

```
trace(0xFFFFFFFF); // 4294967295
trace(0xFFFFFFFF & 0xFFFFFFFF); // -1
trace(0xFFFFFFFF & -1); // -1
trace(4294967295 & -1); // -1
trace(4294967295 & 4294967295); // -1
```

関連項目

`&=` ビット単位の論理積 （AND） 代入演算子 , `^` ビット単位の排他的論理和 （XOR） 演算子 , `^=` ビット単位の排他的論理和 （XOR） 代入演算子 , `|` ビット単位の論理和 （OR） 演算子 , `|=` ビット単位の排他的論理和 （OR） 代入演算子 , `~` ビット単位の否定 （NOT） 演算子

&= ビット単位の論理積 (AND) 代入演算子

expression1 `&=` *expression2*

expression1 に *expression1* `&` *expression2* の値を代入します。たとえば、次の 2 つの式は同じです。

```
x &= y;
x = x & y;
```

対応バージョン： ActionScript 1.0、Flash Player 5

オペランド

expression1 : Number - 数値。

expression2 : Number - 数値。

戻り値

Number - *expression1* & *expression2* の値です。

例

次の例では、x に 値 9 が代入されます。

```
var x:Number = 15;
var y:Number = 9;
trace(x &= y); // output: 9
```

関連項目

& ビット単位の論理積（AND）演算子 , ^ ビット単位の排他的論理和（XOR）演算子 , ^= ビット単位の排他的論理和（XOR）代入演算子 , | ビット単位の論理和（OR）演算子 , |= ビット単位の排他的論理和（OR）代入演算子 , ~ ビット単位の否定（NOT）演算子

<< ビット単位の左シフト演算子

expression1 << *expression2*

expression1 と expression2 を 32 ビット整数値に変換します (それぞれを V1 および V2. と呼ぶとします)。V1 の値のすべてのビットを、V2 位置だけ左にシフトします。この演算により、シフト後に左端からあふれた V1 のビットは破棄され、空になった右側のビット位置には 0 が挿入されます。値を 1 桁左にシフトすることは、2 を掛けることと同じ意味になります。

浮動小数点数は、小数点以下が切り捨てられて整数に変換されます。正の整数は 4294967295 (0xFFFFFFFF) を最大値とする符号なし 16 進値に変換されます。最大値より大きい値は、32 ビットを超えないように、変換時に最上位の桁が切り捨てられます。負の値は、2 の補数表現を使用して、-2147483648 (0x800000000) を最小値とする符号なし 16 進値に変換されます。最小値よりも小さい値は、さらに高い精度で 2 の補数に変換されたうえで、最上位の桁が切り捨てられます。

戻り値は符号付きの 2 の補数として解釈されるため、戻り値は -2147483648 から 2147483647 の整数になります。

対応バージョン : ActionScript 1.0、Flash Player 5

オペランド

expression1 : Number - 左にシフトされる数値または式。

expression2 : Number - 0 ～ 31 の整数に変換される数値または式。

戻り値

Number - ビット演算の結果。

例

次の例では、整数 1 を 10 ビット左にシフトします。x = 1 << 10 この演算の結果は x = 1024 です。10 進の 1 は 2 進の 1 です。10 左にシフトした 2 進の 1 は 10000000000 になります。それを 10 進に戻すと 1024 になります。次の例では、整数 7 を 8 ビット左にシフトします。x = 7 << 8 この演算の結果は x = 1792 です。10 進の 7 は 2 進の 111 です。8 ビット左にシフトした 2 進の 111 は 11100000000 になります。それを 10 進に戻すと 1792 になります。次の例の出力結果を見ると、ビットが 2 つ分、左にシフトされていることがわかります。

```
// 2 binary == 0010
// 8 binary == 1000
trace(2 << 2); // output: 8
```

関連項目

>>= ビット単位の右シフト後代入演算子 , >> ビット単位の右シフト演算子 ,
<<= ビット単位の左シフト後代入演算子 , >>> ビット単位の符号なし右シフト演算子 ,
>>>= ビット単位の符号なし右シフト後代入演算子

<<= ビット単位の左シフト後代入演算子

expression1 <<= *expression2*

この演算子はビット単位の左シフト (<<=) 演算を行い、その内容を結果として *expression1* に格納します。次の 2 つの式は等価です。

```
A <<= B;
A = (A << B)
```

対応バージョン： ActionScript 1.0、Flash Player 5

オペランド

expression1 : Number - 左にシフトされる数値または式。

expression2 : Number - 0 ～ 31 の整数に変換される数値または式。

戻り値

Number - ビット演算の結果。

例

次の例では、ビット単位での左シフト後代入 (<<=) 演算子を使用して、すべてのビットを 1 桁ずつ左にシフトしています。

```
var x:Number = 4;
// shift all bits one slot to the left.
x <<= 1;
trace(x); // output: 8
// 4 decimal = 0100 binary
// 8 decimal = 1000 binary
```

関連項目

<< ビット単位の左シフト演算子 , >>= ビット単位の右シフト後代入演算子 , >> ビット単位の右シフト演算子

~ ビット単位の否定 (NOT) 演算子

~expression

1 の補数演算子またはビット単位の補数演算子とも呼ばれます。*expression* を 32 ビット符号付き整数に変更し、ビット単位で 1 の補数を適用します。つまり、0 を保持するすべてのビットは 1 に設定され、1 を保持するすべてのビットは 0 に設定されます。結果は、符号付き 32 ビット整数です。

たとえば、0x7777 という 16 進数値は、2 進数では 0111011101110111 と表現されます。

この 16 進数値をビット単位で符号反転 (-0x7777) すると、1000100010001000 という 2 進数になります。

16 進数では、これは 0x8888 になります。したがって -0x7777 は 0x8888 です。

ビット単位演算子は、フラグビットを表現する場合に最もよく使用されます (ブール値をそれぞれ 1 ビットにパックすることができます)。

浮動小数点数は、小数点以下が切り捨てられて整数に変換されます。正の整数は 4294967295 (0xFFFFFFFF) を最大値とする符号なし 16 進値に変換されます。最大値より大きい値は、32 ビットを超えないように、変換時に最上位の桁が切り捨てられます。負の値は、2 の補数表現を使用して、-2147483648 (0x800000000) を最小値とする符号なし 16 進値に変換されます。最小値よりも小さい値は、さらに高い精度で 2 の補数に変換されたうえで、最上位の桁が切り捨てられます。

戻り値は符号付きの 2 の補数として解釈されるため、戻り値は -2147483648 から 2147483647 の整数になります。

対応バージョン : ActionScript 1.0、Flash Player 5

オペランド

`expression : Number` - 数値。

戻り値

Number - ビット演算の結果。

例

次に、フラグビットでビット単位の否定 (NOT) (-) 演算子を使用する例を示します。

```
var ReadOnlyFlag:Number = 0x0001; // defines bit 0 as the read-only flag
var flags:Number = 0;
trace(flags);
/* To set the read-only flag in the flags variable,
 the following code uses the bitwise OR:
*/
flags |= ReadOnlyFlag;
trace(flags);
/* To clear the read-only flag in the flags variable,
 first construct a mask by using bitwise NOT on ReadOnlyFlag.
 In the mask, every bit is a 1 except for the read-only flag.
 Then, use bitwise AND with the mask to clear the read-only flag.
 The following code constructs the mask and performs the bitwise AND:
*/
flags &= ~ReadOnlyFlag;
trace(flags);
// output: 0 1 0
```

関連項目

& ビット単位の論理積（AND）演算子, &= ビット単位の論理積（AND）代入演算子, ^ ビット単位の排他的論理和（XOR）演算子, ^= ビット単位の排他的論理和（XOR）代入演算子, | ビット単位の論理和（OR）演算子, |= ビット単位の排他的論理和（OR）代入演算子

| ビット単位の論理和 (OR) 演算子

expression1 | *expression2*

expression1 と *expression2* を 32 ビット符号なし整数に変換し、*expression1* と *expression2* の対応ビットの少なくとも一方が 1 である各ビット位置に 1 を返します。浮動小数点数は、小数点以下が切り捨てられて整数に変換されます。結果は、新しい 32 ビット整数です。

正の整数は 4294967295 (0xFFFFFFFF) を最大値とする符号なし 16 進値に変換されます。最大値より大きい値は、32 ビットを超えないように、変換時に最上位の桁が切り捨てられます。負の値は、2 の補数表現を使用して、-2147483648 (0x800000000) を最小値とする符号なし 16 進値に変換されます。最小値よりも小さい値は、さらに高い精度で 2 の補数に変換されたうえで、最上位の桁が切り捨てられます。

戻り値は符号付きの 2 の補数として解釈されるため、戻り値は -2147483648 から 2147483647 の整数になります。

対応バージョン： ActionScript 1.0、Flash Player 5

オペランド

`expression1` : `Number` - 数値。

`expression2` : `Number` - 数値。

戻り値

`Number` - ビット演算の結果。

例

次に、ビット単位の論理和 (OR) (|) 演算の例を示します。

```
// 15 decimal = 1111 binary
var x:Number = 15;
// 9 decimal = 1001 binary
var y:Number = 9;
// 1111 | 1001 = 1111
trace(x | y); // returns 15 decimal (1111 binary)
```

ビット単位の論理和 (|) と通常の論理和 (||) を混同しないようにしてください。

関連項目

`&` ビット単位の論理積（AND）演算子 , `&=` ビット単位の論理積（AND）代入演算子 , `^` ビット単位の排他的論理和（XOR）演算子 , `^=` ビット単位の排他的論理和（XOR）代入演算子 , `|=` ビット単位の排他的論理和（OR）代入演算子 , `~` ビット単位の否定（NOT）演算子

|= ビット単位の排他的論理和 (OR) 代入演算子

expression1 `|=` *expression2*

expression1 に *expression1* `|` *expression2* の値を代入します。たとえば、次の 2 つのステートメントは同じです。

```
x |= y;
x = x | y;
```

対応バージョン： ActionScript 1.0、Flash Player 5

オペランド

`expression1` : `Number` - 数値または変数。

`expression2` : `Number` - 数値または変数。

戻り値

Number - ビット演算の結果。

例

次にビット単位の排他的論理和 (OR) 代入 (|=)) 演算子の使用例を示します。

```
// 15 decimal = 1111 binary
var x:Number = 15;
// 9 decimal = 1001 binary
var y:Number = 9;
// 1111 |= 1001 = 1111
trace(x |= y); // returns 15 decimal (1111 binary)
```

関連項目

& ビット単位の論理積（AND）演算子 , &= ビット単位の論理積（AND）代入演算子 , ^ ビット単位の排他的論理和（XOR）演算子 , ^= ビット単位の排他的論理和（XOR）代入演算子 , | ビット単位の論理和（OR）演算子 , |= ビット単位の排他的論理和（OR）代入演算子 ,
~ ビット単位の否定（NOT）演算子

>> ビット単位の右シフト演算子

expression1 >> *expression2*

expression1 と *expression2* を 32 ビット整数に変換し、*expression2* の変換により生成された整数で指定される桁数だけ *expression1* 内のすべてのビットを右にシフトします。シフト後、右端のビットは破棄されます。元の *expression* の符号を維持するには、*expression1* の最上位ビット (左端のビット) が 0 である場合は左側のビットに 0 を置き、最上位ビットが 1 である場合には 1 を置きます。値を右に 1 つシフトすることは、2 で割って剰余を切り捨てることと同じです。

浮動小数点数は、小数点以下が切り捨てられて整数に変換されます。正の整数は 4294967295 (0xFFFFFFFF) を最大値とする符号なし 16 進値に変換されます。最大値より大きい値は、32 ビットを超えないように、変換時に最上位の桁が切り捨てられます。負の値は、2 の補数表現を使用して、-2147483648 (0x800000000) を最小値とする符号なし 16 進値に変換されます。最小値よりも小さい値は、さらに高い精度で 2 の補数に変換されたうえで、最上位の桁が切り捨てられます。

戻り値は符号付きの 2 の補数として解釈されるため、戻り値は -2147483648 から 2147483647 の整数になります。

対応バージョン : ActionScript 1.0、Flash Player 5

オペランド

expression1 : Number - 右にシフトされる数値または式。

expression2 : Number - 0 ～ 31 の整数に変換される数値または式。

戻り値

Number - ビット演算の結果。

例

次の例では、65535 を 32 ビット整数に変換し、右側に 8 ビットシフトします。

```
var x:Number = 65535 >> 8;
trace(x); // outputs 255
```

次の例は、前の例の結果を示しています。

```
var x:Number = 255;
```

これは、10 進数の 65535 が 2 進数の 1111111111111111 (1 が 16 個) であり、8 ビットだけ右にシフトすると 2 進数の 11111111 になるためです。2 進数の 11111111 は 10 進数の 255 です。整数は 32 ビットであるため、最上位ビットの 0 で残りのビットを埋めます。

次の例では、-1 を 32 ビット整数に変換し、右側に 1 ビットシフトします。

```
var x:Number = -1 >> 1;
trace(x); // outputs -1
```

次の例は、前の例の結果を示しています。

```
var x:Number = -1;
```

これは、10 進数の -1 が 2 進数の 11111111111111111111111111111111 (1 が 32 個) に等しく、右に 1 ビットシフトすると最下位ビット (右端のビット) が切り捨てられ、最上位ビットに 1 が詰められるためです。その結果は 2 進数の 11111111111111111111111111111111 (1 が 32 個) となり、これは 32 ビットの整数 -1 を表します。

関連項目

>>= ビット単位の右シフト後代入演算子

>>= ビット単位の右シフト後代入演算子

expression1 >>= *expression2*

この演算子はビット単位の右シフト演算を行い、その内容を結果として *expression1* に格納します。

次の 2 つのステートメントは等価です。

```
A >>= B;
A = (A >> B);
```

対応バージョン : ActionScript 1.0、Flash Player 5

オペランド

expression1 : Number - 右にシフトされる数値または式。

expression2 : Number - 0 ～ 31 の整数に変換される数値または式。

戻り値

Number - ビット演算の結果。

例

次のコメント付きコードは、ビット単位の右シフト後代入 (>>=) 演算子の使用例です。

```
function convertToBinary(numberToConvert:Number):String {
 var result:String = "";
 for (var i = 0; i<32; i++) {
 // Extract least significant bit using bitwise AND
 var lsb:Number = numberToConvert & 1;
 // Add this bit to the result
 string result = (lsb ? "1" : "0")+result;
 // Shift numberToConvert right by one bit, to see next bit
 numberToConvert >>= 1;
 }
 return result;
}
trace(convertToBinary(479));
// Returns the string 00000000000000000000000111011111
// This string is the binary representation of the decimal
// number 479
```

関連項目

>> ビット単位の右シフト演算子

>>> ビット単位の符号なし右シフト演算子

expression1 >>> *expression2*

左側のビットには常に 0 が詰められるために元の *expression* の符号が保持されないという点を除いて、ビット単位の右シフト (>>) 演算子と同じです。

浮動小数点数は、小数点以下が切り捨てられて整数に変換されます。正の整数は 4294967295 (0xFFFFFFFF) を最大値とする符号なし 16 進値に変換されます。最大値より大きい値は、32 ビットを超えないように、変換時に最上位の桁が切り捨てられます。負の値は、2 の補数表現を使用して、-2147483648 (0x800000000) を最小値とする符号なし 16 進値に変換されます。最小値よりも小さい値は、さらに高い精度で 2 の補数に変換されたうえで、最上位の桁が切り捨てられます。

対応バージョン : ActionScript 1.0、Flash Player 5

オペランド

expression1 : Number - 右にシフトされる数値または式。

expression2 : Number - 0 ～ 31 の整数に変換される数値または式。

戻り値

Number - ビット演算の結果。

例

次の例では、-1 を 32 ビット整数に変換し、右側に 1 ビットシフトします。

```
var x:Number = -1 >>> 1;
trace(x); // output: 2147483647
```

これは、10 進数の -1 が 2 進数の 11111111111111111111111111111111 (1 が 32 個) であり、右に符号なしで 1 ビットシフトする場合は最下位 (右端) ビットが切り捨てられ、最上位 (左端) ビットに 0 が詰められるためです。その結果は 2 進数の 01111111111111111111111111111111 となります。これは、32 ビット整数の 2147483647 を表します。

関連項目

>>= ビット単位の右シフト後代入演算子

>>>= ビット単位の符号なし右シフト後代入演算子

expression1 >>>= *expression2*

ビット単位の符号なし右シフト演算を行い、その内容を結果として *expression1* に格納します。次の 2 つのステートメントは等価です。

```
A >>>= B;
A = (A >>> B);
```

対応バージョン : ActionScript 1.0、Flash Player 5

オペランド

expression1 : Number - 右にシフトされる数値または式。

expression2 : Number - 0 ～ 31 の整数に変換される数値または式。

戻り値

Number - ビット演算の結果。

例

関連項目

>>> ビット単位の符号なし右シフト演算子 , >>= ビット単位の右シフト後代入演算子

^ ビット単位の排他的論理和 (XOR) 演算子

expression1 ^ *expression2*

expression1 と *expression2* を 32 ビット符号なし整数に変換し、*expression1* と *expression2* の対応ビットのいずれか一方のみが 1 である各ビット位置に 1 を返します。浮動小数点数は、小数点以下が切り捨てられて整数に変換されます。結果は、新しい 32 ビット整数です。

正の整数は 4294967295 (0xFFFFFFFF) を最大値とする符号なし 16 進値に変換されます。最大値より大きい値は、32 ビットを超えないように、変換時に最上位の桁が切り捨てられます。負の値は、2 の補数表現を使用して、-2147483648 (0x800000000) を最小値とする符号なし 16 進値に変換されます。最小値よりも小さい値は、さらに高い精度で 2 の補数に変換されたうえで、最上位の桁が切り捨てられます。

戻り値は符号付きの 2 の補数として解釈されるため、戻り値は -2147483648 から 2147483647 の整数になります。

対応バージョン : ActionScript 1.0、Flash Player 5

オペランド

expression1 : Number - 数値。

expression2 : Number - 数値。

戻り値

Number - ビット演算の結果。

例

次の例では、ビット単位の排他的論理和 (XOR) 演算子を 10 進数である 15 および 9 に適用し、その結果を変数 a に割り当てます。

```
// 15 decimal = 1111 binary
// 9 decimal = 1001 binary
var x:Number = 15 ^ 9;
trace(x);
// 1111 ^ 1001 = 0110
// returns 6 decimal (0110 binary)
```

関連項目

& ビット単位の論理積 （AND） 演算子 , &= ビット単位の論理積 （AND） 代入演算子 ,
^= ビット単位の排他的論理和 （XOR） 代入演算子 , | ビット単位の論理和 （OR） 演算子 ,
|= ビット単位の排他的論理和 （OR） 代入演算子 , ~ ビット単位の否定 （NOT） 演算子

^= ビット単位の排他的論理和 (XOR) 代入演算子

expression1 ^= *expression2*

expression1 に *expression1* ^ *expression2* の値を代入します。たとえば、次の 2 つのステートメントは同じです。

```
x ^= y;
x = x ^ y;
```

対応バージョン : ActionScript 1.0、Flash Player 5

オペランド

expression1 : Number - 整数および変数。

expression2 : Number - 整数および変数。

戻り値

Number - ビット演算の結果。

例

次の例は、ビット単位の排他的論理和 (XOR) (^=) 演算です。

```
// 15 decimal = 1111 binary
var x:Number = 15;
// 9 decimal = 1001 binary
var y:Number = 9;
trace(x ^= y); // returns 6 decimal (0110 binary)
```

関連項目

& ビット単位の論理積 （AND） 演算子 , &= ビット単位の論理積 （AND） 代入演算子 , ^ ビット単位の排他的論理和 （XOR） 演算子 , | ビット単位の論理和 （OR） 演算子 , |= ビット単位の排他的論理和 （OR） 代入演算子 , ~ ビット単位の否定 （NOT） 演算子

/*..*/ コメントブロック区切り記号演算子

```
/* comment */
/* comment
comment */
```

スクリプトコメントのブロックを示します。開始コメントタグ (/*) と終了コメントタグ (*/) の間の文字はすべてコメントと解釈され、ActionScript インタプリタでは無視されます。単一行のコメントブロックを指定するには、// (コメント行区切り記号) を使用します。連続した複数行のコメントブロックを指定するには、/* (コメントブロック区切り記号) を使用します。この形式のコメントブロック区切り記号を使用する際に閉じるタグ (*/) を省略すると、エラーメッセージが返されます。コメントをネストしようとすると、エラーメッセージが返されます。コメントの終わりは、開始コメントタグ (/*) の後、最初に出現する終了コメントタグ (*/) によって示されます。開始コメントタグ (/*) がその間にいくつ指定されていても結果は同じです。

対応バージョン : ActionScript 1.0、Flash Player 5

オペランド

`comment` - 任意の文字。

例

次のスクリプトでは、スクリプトの先頭にコメント区切り記号を使用します。

```
/* records the X and Y positions of
the ball and bat movie clips */
var ballX:Number = ball_mc._x;
var ballY:Number = ball_mc._y;
var batX:Number = bat_mc._x;
var batY:Number = bat_mc._y;
```

次のように、コメントをネストしようとすると、エラーメッセージが表示されます。

```
/* this is an attempt to nest comments.
/* But the first closing tag will be paired
with the first opening tag */
and this text will not be interpreted as a comment */
```

関連項目

// コメント行区切り記号演算子

, カンマ演算子

(expression1 , expression2 [, expressionN...])

式 *expression1*、式 *expression2*、... の順に評価します。通常、この演算子は for ループステートメントで使用します。カッコ () 演算子と組み合わせて使用することもあります。

対応バージョン： ActionScript 1.0、Flash Player 4

オペランド

expression1 : Number - 評価される式。

expression2 : Number - 評価される式。

expressionN : Number - 上記に加えて評価される任意の個数の式。

戻り値

Object - *expression1*、*expression2* などの値。

例

次の例では、for ループでカンマ (,) 演算子を使用しています。

```
for (i = 0, j = 0; i < 3 && j < 3; i++, j+=2) {
 trace("i = " + i + ", j = " + j);
}
// Output:
// i = 0, j = 0
// i = 1, j = 2
```

次の例では、カンマ (,) 演算子を括弧 () 演算子なしで使用し、カンマ演算子が代入 (=) 演算子より優先順位が低いことを示しています。

```
var v:Number = 0;
v = 4, 5, 6;
trace(v); // output: 4
```

次の例では、カンマ (,) 演算子を括弧 () 演算子と共に使用し、カンマ演算子が最後の式の値を返すことを示しています。

```
var v:Number = 0;
v = (4, 5, 6);
trace(v); // output: 6
```

次の例では、カンマ (,) 演算子を括弧 () 演算子なしで使用し、カンマ演算子がすべての式を順番に評価することを示しています。代入 (=) 演算子はカンマ演算子より優先順位が高いため、最初の式 v + 4 が変数 v に割り当てられます。2 番目の式 z++ では、z に 1 を加えています。

```
var v:Number = 0;
var z:Number = 0;
v = v + 4 , z++, v + 6;
trace(v); // output: 4
trace(z); // output: 1
```

次の例は、括弧 () 演算子が追加されていることを除いて前の例と同じです。括弧の追加により演算の順序が変わり、カンマ演算子が代入 (=) 演算子より先に評価されます。

```
var v:Number = 0;
var z:Number = 0;
v = (v + 4, z++, v + 6);
trace(v); // output: 6
trace(z); // output: 1
```

関連項目

( ) カッコ演算子

加算結合 (ストリング) 演算子

string1 add *string2*

非推奨 Flash Player 5 以降では使用しないでください。Flash Player 5 以降用のコンテンツを作成する場合は、加算 (+) 演算子を使用することをお勧めします。この演算子は Flash Player 8 以降ではサポートされていません。

複数のストリングを連結します。加算 (+) 演算子は Flash 4 の連結 (&) 演算子に代わるものです。& 演算子を使用する Flash Player 4 ファイルを Flash 5 以降のオーサリング環境に読み込むと、ストリング連結に加算 (+) 演算子を使用するように自動変換されます。Flash Player 4 以前で使用するためのコンテンツを作成する場合には、ストリング連結に加算 (+) 演算子を使用してください。

対応バージョン : ActionScript 1.0、Flash Player 4

オペランド

string1 : String - ストリング。

string2 : String - ストリング。

戻り値

String - 結合されたストリング。

関連項目

+ 加算演算子

? 条件演算子

expression1 ? *expression2* : *expression3*

expression1 を評価し、*expression1* の値が true である場合は *expression2* の値を返します。それ以外の場合は *expression3* の値を返します。

対応バージョン：ActionScript 1.0、Flash Player 4

オペランド

expression1 : Object - 評価結果がブール値になる式。通常は x < 5 などの比較式。

expression2 : Object - 任意のタイプの値。

expression3 : Object - 任意のタイプの値。

戻り値

Object - *expression2* または *expression3* の値。

例

次のステートメントでは、expression1 の評価結果が true なので、変数 x の値が変数 z に代入されます。

```
var x:Number = 5;
var y:Number = 10;
var z = (x < 6) ? x: y;
trace (z); // returns 5
```

次に、簡単な条件ステートメントの例を示します。

```
var timecode:String = (new Date().getHours() < 11) ? "AM" : "PM";
trace(timecode);
```

次のように、同じ条件ステートメントを、もう少し長く記述することもできます。

```
if (new Date().getHours() < 11) {
 var timecode:String = "AM";
} else {
 var timecode:String = "PM";
} trace(timecode);
```

-- デクリメント演算子

--expression
expression--

expression から 1 を引くプリデクリメント単項演算子またはポストデクリメント単項演算子です。*expression* は、変数、配列のエレメント、またはオブジェクトのプロパティです。プリデクリメント形式の演算子 (*--expression*) は、*expression* から 1 を減算し、結果を返します。ポストデクリメント形式の演算子 (*expression--*) は、*expression* から 1 を減算し、*expression* の初期値 (減算前の値) を返します。

対応バージョン : ActionScript 1.0、Flash Player 4

オペランド

`expression` : `Number` - 数値、または評価結果が数値になる変数。

戻り値

`Number` - デクリメントされた値の結果。

例

プリデクリメント形式の演算子は、x を 2 にデクリメント (x - 1 = 2) して、結果を y として返します。

```
var x:Number = 3;
var y:Number = --x; //y is equal to 2
```

ポストデクリメント形式の演算子は、x を 2 にデクリメント (x - 1 = 2) して、x の元の値を結果 y として返します。

```
var x:Number = 3;
var y:Number = x--; //y is equal to 3
```

次の例は、10 から 1 までループし、各ループでカウンタ変数 i を 1 ずつ減らしています。

```
for (var i = 10; i>0; i--) {
 trace(i);
}
```

/ 除算演算子

expression1 / *expression2*

expression1 を *expression2* で除算します。除算演算の結果は倍精度の浮動小数です。

対応バージョン： ActionScript 1.0、Flash Player 4

オペランド

expression ： Number - 数値、または評価結果が数値になる変数。

戻り値

Number - 浮動小数の演算結果。

例

次のステートメントでは、Stage の現在の幅と高さを除算します。結果は [出力] パネルに表示されます。

```
trace(Stage.width/2);
trace(Stage.height/2);
```

Stage の幅と高さが 550 x 400 (デフォルト) の場合、275 と 150 が出力されます。

関連項目

% 剰余演算子

/= 除算後代入演算子

expression1 /= *expression2*

expression1 に *expression1* / *expression2* の値を代入します。たとえば、次の 2 つのステートメントは同じです。

```
x /= y;
x = x / y;
```

対応バージョン： ActionScript 1.0、Flash Player 4

オペランド

expression1 ： Number - 数値、または評価結果が数値になる変数。

expression2 ： Number - 数値、または評価結果が数値になる変数。

戻り値

Number - 数値。

例

次のコードは、変数と数値を伴う除算後代入 (/=)) 演算子の使用例です。

```
var x:Number = 10;
var y:Number = 2;
x /= y; trace(x); // output: 5
```

関連項目

/ 除算演算子

. ドット演算子

object.property_or_methodinstancename.variable
instancename.childinstanceinstancename.childinstance.variable

ネストされた子のムービークリップ、変数、またはプロパティにアクセスするためにムービークリップの階層をナビゲートする場合に使用します。ドット演算子は、オブジェクトまたはトップレベルクラスのプロパティを調べたり、プロパティを設定するときに使用します。また、オブジェクトまたはトップレベルクラスのメソッドを実行したり、データ構造体を作成する場合にも使用されます。

対応バージョン： ActionScript 1.0、Flash Player 4

オペランド

`object` : `Object` - クラスのインスタンス。任意のビルトインクラスまたはカスタムクラスのインスタンスを指定できます。このパラメータは、常にドット (.) 演算子の左側で使用します。

`property_or_method` - オブジェクトに関連するプロパティまたはメソッドの名前。ビルトインオブジェクトの有効なメソッドとプロパティは、そのクラスのメソッドとプロパティの一覧表に示されています。このパラメータは、常にドット (.) 演算子の右側で使用します。

`instancename` : `MovieClip` - ムービークリップのインスタンス名。`variable` – また、ドット (.) 演算子の左側にあるインスタンス名は、ムービークリップのタイムラインの変数を表すこともできます。

`childinstance` : `MovieClip` - 別のムービークリップの子になっている (ネストされている) ムービークリップインスタンス。

戻り値

`Object` - ドットの右側に指定されたメソッド、プロパティ、またはムービークリップ。

例

次の例では、ムービークリップ person_mc 内の変数 hairColor の現在の値を調べます。

```
person_mc.hairColor
```

Flash 4 のオーサリング環境では、ドットシンタックスはサポートされません。ただし、Flash Player 4 用にパブリッシュされた Flash MX 2004 ファイルでは、ドット演算子を使用できます。前の例は、次に示す従来の Flash 4 シンタックスと同じです。

```
/person_mc:hairColor
```

次の例では、_root スコープ内に新しいムービークリップを作成します。次に、container_mc というムービークリップ内にテキストフィールドを作成しています。テキストフィールドの autoSize プロパティを true に設定し、現在の日付を設定しています。

```
this.createEmptyMovieClip("container_mc", this.getNextHighestDepth());
this.container_mc.createTextField("date_txt", this.getNextHighestDepth(), 0, 0,
  100, 22);
this.container_mc.date_txt.autoSize = true;
this.container_mc.date_txt.text = new Date();
```

ドット (.) 演算子は、SWF ファイル内のインスタンスを指定する場合、およびインスタンスのプロパティや値を設定する場合に使用します。

== 等価演算子

expression1 == expression2

2 つの式の等価性をテストします。式が等しい場合、結果は true です。

「等価」の定義は、パラメータのデータ型により異なります。

- 数値とブール値は値により比較され、それらの値が同じであれば、等しいと見なされます。
- ストリング式は、文字数が同じで、同じ文字で構成されている場合に、等しいと見なされます。
- オブジェクト、配列、および関数を表す変数は、参照により比較されます。2 つの変数が同じオブジェクト、配列、または関数を参照する場合、それらの変数は等価です。2 つの別個の配列は、エレメント数が同じである場合でも、等しいとは見なされません。

値による比較では、*expression1* と *expression2* のデータ型が異なる場合、ActionScript は *expression2* のデータ型を *expression1* と同じデータ型に変換することを試みます。

対応バージョン： ActionScript 1.0、Flash Player 5

オペランド

expression1 : Object - 数値、ストリング、ブール値、変数、オブジェクト、配列、または関数。

expression2 : Object - 数値、ストリング、ブール値、変数、オブジェクト、配列、または関数。

戻り値

Boolean - 比較結果を表すブール値。

例

次の例では、if ステートメントで等価 (==)) 演算子を使用します。

```
var a:String = "David", b:String = "David";
if (a == b) {
 trace("David is David");
}
```

以下の例では、混在するデータ型の比較演算の結果を示します。

```
var x:Number = 5;
var y:String = "5";
trace(x == y); // output: true
var x:String = "5";
var y:String = "66";
trace(x == y); // output: false
var x:String = "chris";
var y:String = "steve";
trace(x == y); // output: false
```

次に参照による比較の例を示します。1 番目の例では、長さとエレメントの同じ 2 つの配列を比較しています。この 2 つの配列に対し、等価演算子は false を返します。これらの配列は一見、同じように見えますが、参照による比較で等価と見なされるためには、同じ配列を参照していることが必要です。2 番目の例では、変数 firstArray と同じ配列を指し示す thirdArray という変数を作成しています。この 2 つの配列に対しては、等価演算子が true を返します。2 つの変数が同じ配列を参照しているからです。

```
var firstArray:Array = new Array("one", "two", "three");
var secondArray:Array = new Array("one", "two", "three");
trace(firstArray == secondArray);
// will output false
// Arrays are only considered equal
// if the variables refer to the same array.
var thirdArray:Array = firstArray;
trace(firstArray == thirdArray); // will output true
```

関連項目

! 論理否定 （NOT） 演算子 , != 不等価演算子 , !== 厳密な不等価演算子 , && 論理積 （AND） 演算子 , || 論理和 （OR） 演算子 , === 厳密な等価演算子

eq 等価 (ストリング) 演算子

expression1 eq *expression2*

非推奨 Flash Player 5 以降では使用しないでください。この演算子の代わりに == (equality) 演算子を使用します。

2 つの式が等しいかどうかを比較して、*expression1* のストリング表現が *expression2* のストリング表現と等しければ値 true を返し、そうでなければ false を返します。

対応バージョン : ActionScript 1.0、Flash Player 4

オペランド

expression1 : Object - 数値、ストリング、または変数。

expression2 : Object - 数値、ストリング、または変数。

戻り値

Boolean - 比較した結果。

関連項目

== 等価演算子

> より大きい演算子

expression1 > *expression2*

2 つの式を比較し、*expression1* が *expression2* より大きいかどうかを判定します。大きい場合は true を返します。*expression1* が *expression2* より小さいか等しい場合は、false を返します。ストリング式はアルファベット順で評価されます。すべての大文字は小文字に先行します。

対応バージョン : ActionScript 1.0、Flash Player 4 - Flash 4 では、> は数値演算子として使用します。Flash 5 以降では、より大きい (>) 演算子は、各種データ型を処理できる比較演算子です。Flash 4 ファイルを Flash 5 以降のオーサリング環境に読み込むと、変換処理が実行され、データ型の整合性が保たれます。次に、数値等価比較が格納された Flash 4 ファイルの変換を示します。Flash 4 ファイル : x > y 変換された Flash 5 以降のファイル : Number(x) > Number(y)

オペランド

expression1 : Object - 数値またはストリング。

expression2 : Object - 数値またはストリング。

戻り値

Boolean - 比較結果を表すブール値。

例

次の例では、より大きい (>) 演算子を使用して、テキストフィールド score_txt の値が 90 より大きいかどうかを判定します。

```
if (score_txt.text>90) {
 trace("Congratulations, you win!");
} else {
 trace("sorry, try again");
}
```

gt より大きい (ストリング) 演算子

expression1 gt *expression2*

非推奨 Flash Player 5 以降では使用しないでください。この演算子の代わりに > (より大きい) 演算子を使用します。

expression1 のストリング表現を *expression2* のストリング表現と比較し、*expression1* が *expression2* より大きい場合は true を返します。それ以外の場合は false を返します。

対応バージョン : ActionScript 1.0、Flash Player 4

オペランド

expression1 : Object - 数値、ストリング、または変数。

expression2 : Object - 数値、ストリング、または変数。

戻り値

Boolean - 比較結果を表すブール値。

関連項目

> より大きい演算子

>= より大きいか等しい演算子

expression1 >= *expression2*

2 つの式を比較し、*expression1* が *expression2* より大きいか等しい場合は true、*expression1* が *expression2* より小さい場合は false と判定します。

対応バージョン : ActionScript 1.0、Flash Player 4 - Flash 4 では、>= は数値演算子として使用します。Flash 5 以降では、より大きいか等しい (>=) 演算子は、各種データ型を処理できる比較演算子です。Flash 4 ファイルを Flash 5 以降のオーサリング環境に読み込むと、変換処理が実行され、データ型の整合性が保たれます。次に、数値等価比較が格納された Flash 4 ファイルの変換を示します。Flash 4 ファイル : x >= y 変換された Flash 5 以降のファイル : Number(x) >= Number(y)

オペランド

expression1 : Object - ストリング、整数、または浮動小数。

expression2 : Object - ストリング、整数、または浮動小数。

戻り値

Boolean - 比較結果を表すブール値。

例

次の例では、より大きいか等しい (>=) 演算子を使用して、現在の時刻 (時) が 12 以上であるかどうかを判定します。

```
if (new Date().getHours() >= 12) {
 trace("good afternoon");
} else {
 trace("good morning");
}
```

ge 大きいか等しい (ストリング) 演算子

expression1 ge *expression2*

非推奨 Flash Player 5 以降では使用しないでください。この演算子の代わりに >= (より大きいか等しい) 演算子を使用します。

expression1 のストリング表現を *expression2* のストリング表現と比較し、*expression1* が *expression2* より大きいか等しい場合は true を返します。それ以外の場合は false を返します。

対応バージョン : ActionScript 1.0、Flash Player 4

オペランド

expression1 : Object - 数値、ストリング、または変数。

expression2 : Object - 数値、ストリング、または変数。

戻り値

Boolean - 比較した結果。

関連項目

>= より大きいか等しい演算子

++ インクリメント演算子

++expression
expression++

expression に 1 を加えるプリインクリメント単項演算子またはポストインクリメント単項演算子です。*expression* は、変数、配列のエレメント、またはオブジェクトのプロパティです。プリインクリメント形式の演算子 (*++expression*) は、*expression* に 1 を加算し、結果を返します。ポストインクリメント形式の演算子 (*expression++*) は、*expression* に 1 を加算し、*expression* の初期値 (加算前の値) を返します。

プリインクリメント形式の演算子は、x を 2 にインクリメント (x + 1 = 2) して、結果を y として返します。

```
var x:Number = 1;
var y:Number = ++x;
trace("x:"+x); //traces x:2
trace("y:"+y); //traces y:2
```

ポストインクリメント形式の演算子は、x を 2 にインクリメント (x + 1 = 2) して、x の元の値を結果 y として返します。

```
var x:Number = 1;
var y:Number = x++;
trace("x:"+x); //traces x:2
trace("y:"+y); //traces y:1
```

対応バージョン : ActionScript 1.0、Flash Player 4

オペランド

expression : Number - 数値、または評価結果が数値になる変数。

戻り値

Number - インクリメントした結果。

例

次の例では、++ をポストインクリメント演算子として使用し、while ループを 5 回実行しています。

```
var i:Number = 0;
while (i++ < 5) {
 trace("this is execution " + i);
}
/* output:
 this is execution 1
 this is execution 2
 this is execution 3
 this is execution 4
 this is execution 5
*/
```

次の例では、++ をプリインクリメント演算子として使用します。

```
var a:Array = new Array();
var i:Number = 0;
while (i < 10) {
 a.push(++i);
}
trace(a.toString()); //traces: 1,2,3,4,5,6,7,8,9,10
```

次の例では、++ をプリインクリメント演算子として使用します。

```
var a:Array = [];
for (var i = 1; i <= 10; ++i) {
 a.push(i);
}
trace(a.toString()); //traces: 1,2,3,4,5,6,7,8,9,10
```

このスクリプトによって、1,2,3,4,5,6,7,8,9,10 という結果が [出力] パネルに表示されます。

次の例では、++ をポストインクリメント演算子として while ループ内で使用します。

```
// using a while loop
var a:Array = new Array();
var i:Number = 0;
while (i < 10) {
 a.push(i++);
}
trace(a.toString()); //traces 0,1,2,3,4,5,6,7,8,9
```

次の例では、for ループで ++ をポストインクリメント演算子として使用しています。

```
// using a for loop
var a:Array = new Array();
for (var i = 0; i < 10; i++) {
 a.push(i);
}
trace(a.toString()); //traces 0,1,2,3,4,5,6,7,8,9
```

このスクリプトを実行すると、次の結果が [出力] パネルに表示されます。

```
0,1,2,3,4,5,6,7,8,9
```

!= 不等価演算子

expression1 != *expression2*

等価 (==) 演算子の正反対が真であるかどうかをテストします。*expression1* が *expression2* に等しい場合、結果は false になります。等価 (==) 演算子と同様、等価の定義は比較対象のデータ型によって次のように異なります。

- 数値、ストリング、およびブール値は、値により比較されます。
- オブジェクト、配列、および関数は、参照で比較されます。
- 変数はその型に応じて、値または参照で比較されます。

値による比較は、2 つの式が同じ値を持つという、ごく一般的な視点から見た等価のことです。たとえば、値で比較した場合、(2 + 3) という式と (1 + 4) という式は等価になります。

参照による比較の場合、2 つの式が等価と見なされるのは、両者が同じオブジェクト、配列、または関数を参照しているときだけです。オブジェクト、配列、または関数によって保持された値は比較の対象になりません。

値による比較では、*expression1* と *expression2* のデータ型が異なる場合、ActionScript は *expression2* のデータ型を *expression1* と同じデータ型に変換することを試みます。

対応バージョン： ActionScript 1.0、Flash Player 5

オペランド

`expression1 : Object` - 数値、ストリング、ブール値、変数、オブジェクト、配列、または関数。

`expression2 : Object` - 数値、ストリング、ブール値、変数、オブジェクト、配列、または関数。

戻り値

`Boolean` - 比較結果を表すブール値。

例

次の例では、不等価 (!=) 演算子の結果を示します。

```
trace(5 != 8); // returns true
trace(5 != 5) //returns false
```

次の例では、if ステートメントでの不等価 (!=) 演算子の使い方を示します。

```
var a:String = "David";
var b:String = "Fool";
if (a != b) {
 trace("David is not a fool");
}
```

次の例では、2 つの関数の参照による比較を示します。

```
var a:Function = function() { trace("foo"); };
var b:Function = function() { trace("foo"); };
a(); // foo
b(); // foo
trace(a != b); // true
a = b;
a(); // foo
b(); // foo
trace(a != b); // false
// trace statement output: foo foo true foo foo false
```

次の例では、2 つの配列の参照による比較を示します。

```
var a:Array = [ 1, 2, 3 ];
var b:Array = [ 1, 2, 3 ];
trace(a); // 1, 2, 3
trace(b); // 1, 2, 3
trace(a!=b); // true
a = b;
trace(a); // 1, 2, 3
trace(b); // 1, 2, 3
trace(a != b); // false
// trace statement output: 1,2,3 1,2,3 true 1,2,3 1,2,3 false
```

関連項目

! 論理否定（NOT）演算子, !== 厳密な不等価演算子, && 論理積（AND）演算子, || 論理和（OR）演算子, == 等価演算子, === 厳密な等価演算子

<> 不等価演算子

expression1 <> *expression2*

非推奨 Flash Player 5 以降では使用しないでください。この演算子は使用されなくなりました。!= (inequality) 演算子を使用することをお勧めします。

等価 (==) 演算子の正反対が真であるかどうかをテストします。*expression1* が *expression2* に等しい場合、結果は false になります。等価 (==) 演算子の場合と同様に、等価の定義は比較対象のデータ型によって次のように異なります。

- 数値、ストリング、およびブール値は、値により比較されます。
- オブジェクト、配列、および関数は、参照で比較されます。
- 変数はその型に応じて、値または参照で比較されます。

対応バージョン： ActionScript 1.0、Flash Player 2

オペランド

expression1 : Object - 数値、ストリング、ブール値、変数、オブジェクト、配列、または関数。

expression2 : Object - 数値、ストリング、ブール値、変数、オブジェクト、配列、または関数。

戻り値

Boolean - 比較結果を表すブール値。

関連項目

!= 不等価演算子

instanceof 演算子

object instanceof *classConstructor*

object が classConstructor のインスタンスまたは classConstructor のサブクラスであるかどうかをテストします。instanceof 演算子は、プリミティブタイプをラッパーオブジェクトに変換しません。たとえば、次のコードは true を返します。

```
new String("Hello") instanceof String;
```

一方、次のコードは false を返します。

```
"Hello" instanceof String;
```

対応バージョン： ActionScript 1.0、Flash Player 6

オペランド

object ： Object - ActionScript オブジェクト。

classConstructor ： Function - String や Date などの ActionScript コンストラクタ関数への参照。

戻り値

Boolean - object が classConstructor のインスタンスまたはサブクラスである場合、instanceof は true を返し、そうでない場合は false を返します。また、_global instanceof Object は false を返します。

関連項目

typeof 演算子

< より小さい演算子

expression1 < *expression2*

2 つの式を比較し、*expression1* が *expression2* より小さいかどうかを判定します。小さい場合は true を返します。*expression1* が *expression2* よりも大きいか等しい場合は、false を返します。ストリング式はアルファベット順で評価されます。すべての大文字は小文字に先行します。

対応バージョン： ActionScript 1.0、Flash Player 4 - Flash 4 では、< は数値演算子として使用します。Flash 5 以降では、(<)(より小さい) 演算子は、各種データ型を処理できる比較演算子です。Flash 4 ファイルを Flash 5 以降のオーサリング環境に読み込むと、変換処理が実行され、データ型の整合性が保たれます。次に、数値等価比較が格納された Flash 4 ファイルの変換を示します。

Flash 4 ファイル：x < y

変換された Flash 5 以降のファイル：Number(x) < Number(y)

オペランド

expression1 : Number - 数値またはストリング。

expression2 : Number - 数値またはストリング。

戻り値

Boolean - 比較結果を表すブール値。

例

次の例では、数値とストリングの両方について true と false が返される場合を示します。

```
trace(3 < 10); // true
trace(10 < 3); // false
trace("Allen" < "Jack"); // true
trace("Jack" < "Allen"); //false
trace("11" < "3"); // true
trace("11" < 3); // false (numeric comparison)
trace("C" < "abc"); // true
trace("A" < "a"); // true
```

lt より小さい (ストリング) 演算子

expression1 lt *expression2*

非推奨 Flash Player 5 以降では使用しないでください。この演算子の代わりに < (より小さい) 演算子を使用します。

expression1 と *expression2* を比較して、*expression1* が *expression2* よりも小さい場合は true を返し、そうでない場合は false を返します。

対応バージョン : ActionScript 1.0、Flash Player 4

オペランド

expression1 : Object - 数値、ストリング、または変数。

expression2 : Object - 数値、ストリング、または変数。

戻り値

Boolean - 比較した結果。

関連項目

< より小さい演算子

<= より小さいか等しい演算子

expression1 <= *expression2*

2 つの式を比較し、*expression1* が *expression2* より小さいまたは等しいかどうかを判定します。小さいか等しい場合は true を返します。 *expression1* が *expression2* よりも大きい場合は、false を返します。ストリング式はアルファベット順で評価されます。すべての大文字は小文字に先行します。

対応バージョン： ActionScript 1.0、Flash Player 4 - Flash 4 では、<= は数値演算子として使用します。Flash 5 以降では、より小さいか等しい (<=) 演算子は、各種データ型を処理できる比較演算子です。Flash 4 ファイルを Flash 5 以降のオーサリング環境に読み込むと、変換処理が実行され、データ型の整合性が保たれます。次に、数値等価比較が格納された Flash 4 ファイルの変換を示します。

Flash 4 ファイル : `x <= y`

変換された Flash 5 以降のファイル : `Number(x) <= Number(y)`

オペランド

`expression1 : Object` - 数値またはストリング。

`expression2 : Object` - 数値またはストリング。

戻り値

`Boolean` - 比較結果を表すブール値。

例

次に、数値とストリングの両方について true と false が返される場合の例を示します。

```
trace(5 <= 10); // true
trace(2 <= 2); // true
trace(10 <= 3); // false
trace("Allen" <= "Jack"); // true
trace("Jack" <= "Allen"); // false
trace("11" <= "3"); // true
trace("11" <= 3); // false (numeric comparison)
trace("C" <= "abc"); // true
trace("A" <= a); // true
```

le 小さいか等しい (ストリング) 演算子

expression1 le *expression2*

非推奨 Flash Player 5 以降では使用しないでください。この演算子の代わりに <= (より小さいか等しい) 演算子を使用します。

expression1 を *expression2* と比較し、*expression1* が *expression2* 以下の場合は true を返し、それ以外の場合は false を返します。

対応バージョン : ActionScript 1.0、Flash Player 4

オペランド

expression1 : Object - 数値、ストリング、または変数。

expression2 : Object - 数値、ストリング、または変数。

戻り値

Boolean - 比較した結果。

関連項目

<= より小さいか等しい演算子

// コメント行区切り記号演算子

// *comment*

スクリプトコメントの先頭を示します。コメント行区切り記号 (//) と行末の間に表示される文字はすべてコメントと解釈され、ActionScript インタプリタによって無視されます。

対応バージョン : ActionScript 1.0、Flash Player 1.0

オペランド

comment - 任意の文字。

例

次のスクリプトは、コメント行区切り記号を使用して 1 行目、3 行目、5 行目、7 行目をコメントとして識別します。

```
// record the X position of the ball movie clip
var ballX:Number = ball_mc._x;
// record the Y position of the ball movie clip
var ballY:Number = ball_mc._y;
// record the X position of the bat movie clip
var batX:Number = bat_mc._x;
// record the Y position of the bat movie clip
var batY:Number = bat_mc._y;
```

関連項目

/*..*/ コメントブロック区切り記号演算子

&& 論理積 (AND) 演算子

expression1 && *expression2*

オペランドをブール (論理) として評価し、論理演算を行います。*expression1* と *expression2* が両方とも true である場合は、true が返されます。それ以外の場合は false が返されます。

式	評価
true&&true	true
true&&false	false
false&&false	false
false&&true	false

対応バージョン : ActionScript 1.0、Flash Player 4

オペランド

expression1 : Number - ブール値、またはブール値に変換される式。

expression2 : Number - ブール値、またはブール値に変換される式。

戻り値

Boolean - 論理演算結果を表すブール値。

例

次の例では、論理積 AND (&&) 演算子を使用してゲームの勝敗判定をテストします。turns 変数と score 変数は、ゲームの回数または得点に応じて更新されます。次のスクリプトでは、3 回以内に 75 点以上を得点すると、[出力] パネルに "You Win the Game!" と表示されます。

```
var turns:Number = 2;
var score:Number = 77;
if ((turns <= 3) && (score >= 75)) {
 trace("You Win the Game!");
} else {
 trace("Try Again!");
}
// output: You Win the Game!
```

関連項目

! 論理否定 （NOT） 演算子 , != 不等価演算子 , !== 厳密な不等価演算子 , || 論理和 （OR） 演算子 , == 等価演算子 , === 厳密な等価演算子

and 論理積 (AND) 演算子

condition1 and *condition2*

非推奨 Flash Player 5 以降では使用しないでください。論理積 (AND) (&&) 演算子を使用することをお勧めします。

Flash Player 4 で論理積 (AND) (&&) 演算を実行します。両式の評価値が true である場合に、式全体が true になります。

対応バージョン： ActionScript 1.0、Flash Player 4

オペランド

condition1 : Boolean - true または false に評価される条件または式。

condition2 : Boolean - true または false に評価される条件または式。

戻り値

Boolean - 論理演算結果を表すブール値。

関連項目

&& 論理積 （AND） 演算子

! 論理否定 (NOT) 演算子

```
! expression
```

変数や式のブール値を反転します。*expression* が変数で、その絶対値または変換された値が true である場合、! *expression* の値は false になります。式 x && y の評価が false である場合、式 !(x && y) の評価は true です。

次の式は、論理否定 (!) 演算子を使用した結果を示します。

! *true* は false を返し、! *false* は true を返す

対応バージョン： ActionScript 1.0、Flash Player 4

オペランド

expression : Boolean - 評価結果がブール値になる式または変数。

戻り値

Boolean - 論理演算結果を表すブール値。

例

次の例では、変数 happy が false に設定されています。if ステートメントの条件部 !happy を評価し、その結果が true である場合、trace() ステートメントによりストリングが [出力] パネルに表示されます。

```
var happy:Boolean = false;
if (!happy) {
 trace("don't worry, be happy"); //traces don't worry, be happy
}
```

!false は true と等価であるため trace が実行されます。

関連項目

!= 不等価演算子 , !== 厳密な不等価演算子 , && 論理積 （AND） 演算子 , || 論理和 （OR） 演算子 , == 等価演算子 , === 厳密な等価演算子

not 論理否定 (NOT) 演算子

`not expression`

非推奨 Flash Player 5 以降では**使用しないでください**。この演算子の代わりに ! (logical NOT) (等価) 演算子を使用します。

Flash Player 4 で論理否定 (NOT) (!) 演算をします。

対応バージョン : ActionScript 1.0、Flash Player 4

オペランド

`expression : Object` - ブール値に変換される変数または式。

戻り値

`Boolean` - 論理演算の結果。

関連項目

! 論理否定 (NOT) 演算子

|| 論理和 (OR) 演算子

expression1 || *expression2*

オペランドをブール (論理) 値として評価し、論理演算を行います。オペランドがブール値を返す論理式 (等価や非等価、比較などを調べる式) であれば、両式が共に false の場合のみ false を返します。それ以外の場合、つまりどちらか一方の式または両式が true であれば、戻り値は true になります。*expression1* の評価が false であれば、*expression2* (演算子の右側の式) が評価されます。*expression2* の評価が false であれば、最終結果は false です。そうでなければ、true です。

expression1 の評価が true の場合、*expression2* に関数呼び出しが指定されていても、その関数は実行されません。

一方の式または両式の評価が true である場合、結果は true になります。両式の評価が false である場合のみ、結果は false になります。論理和 (OR) 演算子は任意の数のオペランドに適用できます。いずれかのオペランドの評価が true であれば、結果は true です。

対応バージョン : ActionScript 1.0、Flash Player 4

オペランド

`expression1 : Number` - ブール値、またはブール値に変換される式。

`expression2 : Number` - ブール値、またはブール値に変換される式。

戻り値

Boolean - 論理演算の結果。

例

次の例では、if ステートメントで論理和 (OR) (| |) 演算子を使用しています。2 番目の式の評価結果が true なので、最終結果は true になります。

```
var x:Number = 10;
var y:Number = 250;
var start:Boolean = false;
if ((x > 25) || (y > 200) || (start)) {
 trace("the logical OR test passed"); // output: the logical OR test passed
}
```

if ステートメントの条件の 1 つが true (y>200) なので、"the logical OR test passed" というメッセージが表示されます。それ以外の 2 つの式が false と評価されても、true と評価される条件が 1 つでもあれば、if ブロックが実行されます。

次のように、*expression2* に関数呼び出しを指定すると、予期しない結果が生じる場合があります。演算子の左側の式が true に評価された場合、右側の式は評価されずに (関数 fx2() は呼び出されないで)、左側の式の結果だけが返されます。

```
function fx1():Boolean {
 trace("fx1 called");
 return true;
}
function fx2():Boolean {
 trace("fx2 called");
 return true;
}
if (fx1() || fx2()) {
 trace("IF statement entered");
}
```

次の内容が [出力] パネルに表示されます :fx1 called IF statement entered

関連項目

! 論理否定 （NOT） 演算子 , != 不等価演算子 , !== 厳密な不等価演算子 , && 論理積 （AND） 演算子 , == 等価演算子 , === 厳密な等価演算子

or 論理和 (OR) 演算子

condition1 or *condition2*

非推奨 Flash Player 5 以降では使用しないでください。この演算子の代わりに || （logical OR）演算子を使用します。

condition1 と *condition2* を評価し、いずれかの式が true であれば、式全体が true になります。

対応バージョン： ActionScript 1.0、Flash Player 4

オペランド

condition1 : Boolean - 評価結果が true または false になる式。

condition2 : Boolean - 評価結果が true または false になる式。

戻り値

Boolean - 論理演算の結果。

関連項目

|| 論理和 （OR） 演算子 , | ビット単位の論理和 （OR） 演算子

% 剰余演算子

expression1 % *expression2*

expression1 を *expression2* で割ったときの剰余を計算します。いずれかの *expression* パラメータが非数値である場合、剰余 (%) 演算子はそれを数値に変換しようと試みます。*expression* は数値、または数値に変換されるストリングです。

剰余演算結果の符号は、被除数 (最初の数値) の符号と一致します。たとえば、-4 % 3 と -4 % -3 の評価結果は共に -1 になります。

対応バージョン： ActionScript 1.0、Flash Player 4 - Flash 4 ファイルでは、% 演算子は、SWF ファイルで x - int(x/y) * y に拡張されるため、これより後の Flash Player のバージョンと比較して、速度や正確さが劣る場合があります。

オペランド

expression1 : Number - 数値、または評価結果が数値になる式。

expression2 : Number - 数値、または評価結果が数値になる式。

戻り値

Number - 算術演算の結果。

例

次の例では、数値に対して剰余 (%) 演算子を使用します。

```
trace(12%5); // traces 2
trace(4.3%2.1); // traces 0.0999999999999996
trace(4%4); // traces 0
```

剰余 (%) 演算子は余りだけを返すため、最初の trace ステートメントでは 12/5 や 2.4 ではなく 2 が返されます。ところが、2 番目の trace ステートメントでは 0.1 ではなく、0.0999999999999996 が返されます。これは 2 進数計算では浮動小数点の精度に限度があるからです。

関連項目

/ 除算演算子 , round （Math.round メソッド ）

%= 剰余代入演算子

expression1 %= expression2

expression1 に *expression1 % expression2* の値を代入します。次の 2 つのステートメントは等価です。

```
x %= y;
x = x % y;
```

対応バージョン： ActionScript 1.0、Flash Player 4 - Flash 4 ファイルでは、% 演算子は、SWF ファイルで `x - int(x/y) * y` に拡張されるため、これより後の Flash Player のバージョンと比較して、速度や正確さが劣る場合があります。

オペランド

`expression1` : `Number` - 数値、または評価結果が数値になる式。

`expression2` : `Number` - 数値、または評価結果が数値になる式。

戻り値

`Number` - 算術演算の結果。

例

次の例では、値 4 を変数 x に代入します。

```
var x:Number = 14;
var y:Number = 5;
trace(x = y); // output: 4
```

関連項目

% 剰余演算子

* 乗算演算子

*expression1 * expression2*

2 つの数値や式を乗算します。両方の式が整数であれば、その積は整数です。いずれかの式または両方の式が浮動小数であれば、その積は浮動小数になります。

対応バージョン： ActionScript 1.0、Flash Player 4

オペランド

expression1 : Number - 数値、または評価結果が数値になる式。

expression2 : Number - 数値、または評価結果が数値になる式。

戻り値

Number - 整数または浮動小数。

例

シンタックス 1: 次のステートメントでは、整数 2 と 3 を乗算します。

```
trace(2*3); // output: 6
```

結果は、整数の 6 です。シンタックス 2: 次のステートメントは、浮動小数 2.0 と 3.1416 を乗算します。

```
trace(2.0 * 3.1416); // output: 6.2832
```

結果は、浮動小数の 6.2832 です。

*= 乗算後代入演算子

*expression1 *= expression2*

expression1 に *expression1 * expression2* の値を代入します。たとえば、次の 2 つの式は同じです。

```
x *= y;
x = x * y
```

対応バージョン： ActionScript 1.0、Flash Player 4

オペランド

expression1 : Number - 数値、または評価結果が数値になる式。

expression2 : Number - 数値、または評価結果が数値になる式。

戻り値

Number - *expression1 * expression2* の値です。式を数値に変換できない場合は、NaN (非数) を返します。

例

シンタックス 1: 次の例では、値 50 を変数 x に代入します。

```
var x:Number = 5;
var y:Number = 10;
trace(x *= y); // output: 50
```

シンタックス 2: 次の例の 2 行目と 3 行目は、等号の右側の式を計算し、その結果を x と y にそれぞれ代入します。

```
var i:Number = 5;
var x:Number = 4 - 6;
var y:Number = i + 2;
trace(x *= y); // output: -14
```

関連項目

* 乗算演算子

new 演算子

new *constructor()*

新しい匿名のオブジェクトを作成し、constructor パラメータで指定された関数を呼び出します。new 演算子は、カッコ内のオプションのパラメータと、キーワード this で参照される新しく作成されたオブジェクトを関数に渡します。constructor 関数は this を使用してオブジェクトの変数を設定できます。

対応バージョン : ActionScript 1.0、Flash Player 5

オペランド

constructor : Object - カッコで囲まれたオプションのパラメータが後に続く関数。この関数は通常、作成するオブジェクトのタイプの名前 (Array、Number、Object など) です。

例

次の例では、Book( ) 関数を作成し、new 演算子を使用して book1 オブジェクトと book2 オブジェクトを作成します。

```
function Book(name, price){
 this.name = name;
 this.price = price;
}

book1 = new Book("Confederacy of Dunces", 19.95);
book2 = new Book("The Floating Opera", 10.95);
```

次の例では、new 演算子を使用して、18 個のエレメントを含む Array オブジェクトを作成します。

```
golfCourse_array = new Array(18);
```

関連項目

[] 配列アクセス演算子 , {} オブジェクト初期化演算子

ne 不等価 (ストリング) 演算子

expression1 ne *expression2*

非推奨 Flash Player 5 以降では使用しないでください。この演算子の代わりに != （inequality）演算子を使用します。

expression1 を *expression2* と比較して、*expression1* が *expression2* と等しくない場合は true を返します。それ以外の場合は false を返します。

対応バージョン： ActionScript 1.0、Flash Player 4

オペランド

expression1 : Object - 数値、ストリング、または変数。

expression2 : Object - 数値、ストリング、または変数。

戻り値

Boolean - expression1 が expression2 と等しくない場合は true を返します。それ以外の場合は false を返します。

関連項目

!= 不等価演算子

{} オブジェクト初期化演算子

object = { *name1 : value1 , name2 : value2 ,... nameN : valueN* }
{*expression1; [...expressionN]*}

新しいオブジェクトを作成し、指定された *name* と *value* プロパティペアで初期化します。この演算子を使用することは、new Object シンタックスを使用して代入演算子でプロパティペアを設定するのと同じです。新しく作成されるオブジェクトのプロトタイプとして、汎用の Object オブジェクトがあります。

また、フローを制御するステートメント (for、while、if、else、switch) や関数で、ひとかたまりのコードブロックであることを示すときにも、この演算子が使用されます。

対応バージョン： ActionScript 1.0、Flash Player 5

オペランド

`object : Object` - 作成するオブジェクト。*name1,2,...N* プロパティの名前。 *value1,2,...N* 各 *name* プロパティに対応する値。

戻り値

`Object` -

シンタックス 1: Object オブジェクト。

シンタックス 2: なし。ただし、関数で明示的に `return` ステートメントが指定されている場合は、その関数で戻り値の型が指定されます。

例

次のコードの 1 行目はオブジェクト初期化 ({}) 演算子を使用して空のオブジェクトを作成し、2 行目はコンストラクタ関数を使用して新しいオブジェクトを作成します。

```
var object:Object = {};
var object:Object = new Object();
```

次の例では、オブジェクト account を作成し、name、address、city、state、zip、balance の各プロパティを対応する値で初期化します。

```
var account:Object = {name:"Adobe", address:"601 Townsend Street", city:"San
  Francisco", state:"California", zip:"94103", balance:"1000"};
for (i in account) {
 trace("account." + i + " = " + account[i]);
}
```

次の例では、配列とオブジェクトの初期化演算子を相互にネストする方法を示します。

```
var person:Object = {name:"Gina Vechio", children:["Ruby", "Chickie", "Puppa"]};
```

次の例では、前の例と同じ内容を、コンストラクタ関数で生成します。

```
var person:Object = new Object();
person.name = "Gina Vechio";
person.children = new Array();
person.children[0] = "Ruby";
person.children[1] = "Chickie";
person.children[2] = "Puppa";
```

上の例で示した ActionScript は、次のように記述することもできます。

```
var person:Object = new Object();
person.name = "Gina Vechio";
person.children = new Array("Ruby", "Chickie", "Puppa");
```

関連項目

`Object`

() カッコ演算子

(expression1 [, expression2])
(expression1, expression2)
function (parameter1,..., parameterN)

パラメータに対してグループ化演算を実行するか、複数の式を順番に評価します。または、パラメータを囲み、結果をパラメータとしてカッコの外側にある関数に渡します。

シンタックス 1: 式内での演算子の実行順序を制御します。カッコは通常の優先順位を無効にし、カッコ内の式を最初に評価します。カッコがネストされている場合は、最も内側のカッコから順に外側のカッコへと内容が評価されます。

シンタックス 2: カンマ区切りの一連の式を順番に評価し、最後に実行された式の結果を返します。

シンタックス 3: パラメータを囲み、パラメータとして括弧の外側にある関数に渡します。

対応バージョン: ActionScript 1.0、Flash Player 4

オペランド

`expression1 : Object` - 数値、ストリング、変数、またはテキスト。

`expression2 : Object` - 数値、ストリング、変数、またはテキスト。

`function : Function` - カッコの内容に対して実行される関数。

`parameter1...parameterN : Object` - これらのパラメータの実行結果がパラメータとしてカッコの外側の関数に渡されます。

例

シンタックス 1: 次のステートメントは、括弧を使用して式の実行順序を制御します (各式の値が [出力] パネルに表示されます)。

```
trace((2 + 3)*(4 + 5)); // Output: 45
trace((2 + 3) * (4 + 5)); // Output: 45trace(2 + (3 * (4 + 5))); // // writes 29
trace(2 + (3 * (4 + 5))); // Output: 29trace(2+(3*4)+5); // writes 19
trace(2 + (3 * 4) + 5); // Output: 19
```

シンタックス 2: 次の例は、関数 `foo()` を評価した後に関数 `bar()` を評価し、さらに式 `a + b` の結果を返します。

```
var a:Number = 1;
var b:Number = 2;
function foo() { a += b; }
function bar() { b *= 10; }
trace((foo(), bar(), a + b)); // outputs 23
```

シンタックス 3: 次の例は、関数で括弧を使用する場合を示しています。

```
var today:Date = new Date();
trace(today.getFullYear()); // traces current year
function traceParameter(param):Void { trace(param); }
traceParameter(2 * 2); //traces 4
```

関連項目

with ステートメント

=== 厳密な等価演算子

expression1 === expression2

2 つの式が等しいかどうかをテストします。厳密な等価 (===) 演算子は、データ型が変換されない点を除いては、等価 (==) 演算子と同じです。両方の式が、そのデータ型も含めて等しい場合、結果は true です。

「等価」の定義は、パラメータのデータ型により異なります。

- 数値とブール値は値により比較され、それらの値が同じであれば、等しいと見なされます。
- ストリング式は、文字数が同じで、同じ文字で構成されている場合に、等しいと見なされます。
- オブジェクト、配列、および関数を表す変数は、参照により比較されます。2 つの変数が同じオブジェクト、配列、または関数を参照する場合、それらの変数は等価です。2 つの別個の配列は、エレメント数が同じである場合でも、等しいとは見なされません。

対応バージョン : ActionScript 1.0、Flash Player 6

オペランド

expression1 : Object - 数値、ストリング、ブール値、変数、オブジェクト、配列、または関数。

expression2 : Object - 数値、ストリング、ブール値、変数、オブジェクト、配列、または関数。

戻り値

Boolean - 比較結果を表すブール値。

例

次のコードのコメントは、等価演算子および厳密な等価演算子を使用した演算の戻り値を示しています。

```
// Both return true because no conversion is done
var string1:String = "5";
var string2:String = "5";
trace(string1 == string2); // true
trace(string1 === string2); // true
// Automatic data typing in this example converts 5 to "5"
var string1:String = "5";
var num:Number = 5;
trace(string1 == num); // true
trace(string1 === num); // false
// Automatic data typing in this example converts true to "1"
var string1:String = "1";
var bool1:Boolean = true;
trace(string1 == bool1); // true
```

```
trace(string1 === bool1); // false
// Automatic data typing in this example converts false to "0"
var string1:String = "0";
var bool2:Boolean = false;
trace(string1 == bool2); // true
trace(string1 === bool2); // false
```

次の例では、厳密な等価で比較する場合に、参照を格納する変数とリテラル値を格納する変数の扱いがどのように違うかを示しています。この結果を見ると、String クラスでは new 演算子の使用を避け、ストリングリテラルを使用する理由が明らかになります。

```
// Create a string variable using a literal value
var str:String = "asdf";
// Create a variable that is a reference
var stringRef:String = new String("asdf");
// The equality operator does not distinguish among literals, variables,
// and references
trace(stringRef == "asdf"); // true
trace(stringRef == str); // true
trace("asdf" == str); // true
// The strict equality operator considers variables that are references
// distinct from literals and variables
trace(stringRef === "asdf"); // false
trace(stringRef === str); // false
```

関連項目

! 論理否定（NOT）演算子, != 不等価演算子, !== 厳密な不等価演算子, && 論理積（AND）演算子, || 論理和（OR）演算子, == 等価演算子

!== 厳密な不等価演算子

expression1 !== *expression2*

厳密な等価 (===) 演算子の正反対が真であるかどうかをテストします。厳密な不等価演算子の動作は、データ型が変換されない点を除いて、不等価演算子と同じです。

expression1 が *expression2* と等しく、両者のデータ型が同じである場合、結果は false になります。厳密な等価 (===) 演算子と同様、等価の定義は比較対象のデータ型によって次のように異なります。

- 数値、ストリング、およびブール値は、値により比較されます。
- オブジェクト、配列、および関数は、参照で比較されます。
- 変数はその型に応じて、値または参照で比較されます。

対応バージョン： ActionScript 1.0、Flash Player 6

オペランド

`expression1 : Object` - 数値、ストリング、ブール値、変数、オブジェクト、配列、または関数。

`expression2 : Object` - 数値、ストリング、ブール値、変数、オブジェクト、配列、または関数。

戻り値

`Boolean` - 比較結果を表すブール値。

例

次のコードのコメントは、等価演算子 (==)、厳密な等価演算子 (===)、厳密な不等価演算子 (!==) を使用した演算の戻り値を示しています。

```
var s1:String = "5";
var s2:String = "5";
var s3:String = "Hello";
var n:Number = 5;
var b:Boolean = true;
trace(s1 == s2); // true
trace(s1 == s3); // false
trace(s1 == n); // true
trace(s1 == b); // false
trace(s1 === s2); // true
trace(s1 === s3); // false
trace(s1 === n); // false
trace(s1 === b); // false
trace(s1 !== s2); // false
trace(s1 !== s3); // true
trace(s1 !== n); // true
trace(s1 !== b); // true
```

関連項目

`! 論理否定 （NOT） 演算子`, `!= 不等価演算子`, `&& 論理積 （AND） 演算子`, `|| 論理和 （OR） 演算子`, `== 等価演算子`, `=== 厳密な等価演算子`

" ストリング区切り記号演算子

"*text*"

引用符 (") で前後を囲んだ文字はリテラル値を表します。変数、数値、その他の ActionScript エレメントではなく、ストリング と見なされます。

対応バージョン： ActionScript 1.0、Flash Player 4

オペランド

`text : String` - 0 個以上の文字のシーケンス。

例

次の例では、引用符 (") を使用して、変数 *yourGuess* の値がリテラルストリング "Prince Edward Island" であり、変数名ではないことを指定します。province の値は変数であり、リテラルではありません。province の値を決定するには、*yourGuess* の値を特定する必要があります。

```
var yourGuess:String = "Prince Edward Island";
submit_btn.onRelease = function() { trace(yourGuess); };
// displays Prince Edward Island
```

関連項目

String, String 関数

- 減算演算子

```
(Negation) -expression
(Subtraction) expression1 - expression2
```

符号反転や減算に使用します。

シンタックス 1: 符号反転に使用する場合、数値 *expression* の符号を逆にします。

シンタックス 2: 減算に使用する場合、2 つの式に対して算術的な減算を行い、*expression1* から *expression2* を減算します。両方の式が整数であれば、その差は整数です。いずれかの式または両方の式が浮動小数であれば、その差は浮動小数です。

対応バージョン : ActionScript 1.0、Flash Player 4

オペランド

expression1 : Number - 数値、または評価結果が数値になる式。

expression2 : Number - 数値、または評価結果が数値になる式。

戻り値

Number - 整数または浮動小数。

例

シンタックス 1: 次のステートメントは、式 2 + 3 の符号を逆にします。

```
trace(-(2+3)); // output: -5
```

シンタックス 2: 次のステートメントは、整数 5 から整数 2 を減算します。

```
trace(5-2); // output: 3
```

結果は、整数の 3 です。シンタックス 3: 次のステートメントは、浮動小数 3.25 から浮動小数 1.5 を減算します。

```
trace(3.25-1.5); // output: 1.75
```

結果は、浮動小数の 1.75 です。

-= 減算後代入演算子

expression1 `-=` *expression2*

expression1 に *expression1* `-` *expression2* の値を代入します。たとえば、次の 2 つのステートメントは同じです。`x -= y ;x = x - y;`

ストリング式は数値に変換される必要があります。変換されない場合は、`NaN` (非数) が返されます。

対応バージョン : ActionScript 1.0、Flash Player 4

オペランド

`expression1 : Number` - 数値、または評価結果が数値になる式。

`expression2 : Number` - 数値、または評価結果が数値になる式。

戻り値

`Number` - 算術演算の結果。

例

次の例では、減算後代入 (`-=`) 演算子を使用して、5 から 10 を減算し、その結果を変数 x に割り当てます。

```
var x:Number = 5;
var y:Number = 10;
x -= y; trace(x); // output: -5
```

次の例では、ストリングがどのように数値に変換されるかを示します。

```
var x:String = "5";
var y:String = "10";
x -= y; trace(x); // output: -5
```

関連項目

- 減算演算子

: type 演算子

```
[ modifiers ] var variableName : type
function functionName () : type { ... }
function functionName ( parameter1:type , ... , parameterN:type ) [ :type ]{ ... }
```

厳密な型指定を行う際に使用します。この演算子では、変数のタイプ、関数の戻り値のタイプ、または関数パラメータのタイプを指定します。変数の宣言または代入で使用する場合、この演算子は変数のタイプを指定します。関数の宣言または定義で使用する場合、この演算子は関数の戻り値のタイプを指定します。関数定義の関数パラメータで使用する場合、この演算子はそのパラメータに指定する変数のタイプを指定します。

タイプはコンパイル時のみの機能です。すべてのタイプはコンパイル時にチェックされ、不一致が見つかった場合はエラーが報告されます。型の不一致は、代入操作、関数の呼び出し、ドット (.) 演算子を使用したクラスメンバーの逆参照の際に発生する可能性があります。タイプの不一致エラーを防ぐには、厳密な型指定を使用します。

使用できるタイプとしては、すべてのネイティブオブジェクトタイプ、自分で定義したクラスとインターフェイス、Function および Void があります。認識されるネイティブタイプには、ブール (Boolean)、数値 (Number)、ストリング (String) があります。また、すべてのビルトインクラスは、ネイティブタイプとしてサポートされます。

対応バージョン： ActionScript 1.0、Flash Player 6

オペランド

`variableName : Object` - 変数の識別子。*type* ネイティブのデータ型、定義済みのクラス名、またはインターフェイス名。*functionName* 関数の識別子。*parameter* 関数の識別子。

例

シンタックス 1: 次の例では、userName という名前のパブリック変数を宣言します。この変数は String 型です。この変数に空のストリングを代入します。

```
var userName:String = "";
```

シンタックス 2: 次の例は、関数パラメータの型を指定する方法を示しています。ここでは、integer という Number 型のパラメータを受け取る randomInt() という関数を定義しています。

```
function randomInt(integer:Number):Number {
 return Math.round(Math.random()*integer);
}
trace(randomInt(8));
```

シンタックス 3: 次の例は、squareRoot() という名前の関数を定義します。この関数は val という Number 型のパラメータを受け取り、val の平方根を返します。この戻り値も Number 型です。

```
function squareRoot(val:Number):Number {
 return Math.sqrt(val);
}
trace(squareRoot(121));
```

関連項目

var ステートメント , function ステートメント

typeof 演算子

typeof(*expression*)

expression を評価し、その式の値が String, MovieClip, Object, Function, Number, Boolean のいずれであるかを示すストリングを返します。

対応バージョン： ActionScript 1.0、Flash Player 5

オペランド

expression : Object - ストリング、ムービークリップ、ボタン、オブジェクト、または関数。

戻り値

String - 式の型を示すストリング表現。次の表は、typeof 演算子を使用した結果を式の型別に示しています。

式のタイプ	結果
ストリング	string
ムービークリップ	movieclip
ボタン	object
テキストフィールド	object
数値	number
ブール値	boolean
オブジェクト	object
関数	function

関連項目

instanceof 演算子

void 演算子

`void` *expression*

`void` 演算子は式を評価した後、その値を破棄して、`undefined` を返します。`void` 演算子は、多くの場合、== 演算子を使用して undefined 値をテストする比較において使用します。

対応バージョン： ActionScript 1.0、Flash Player 5

オペランド

`expression` : `Object` - 評価される式。

ステートメント

ステートメントとは、特定のアクションを実行または指定する言語エレメントです。たとえば、return ステートメントは、関数の実行結果を値として返します。if ステートメントは、次に行うべき処理を判定するための条件を評価します。switch ステートメントは ActionScript ステートメントの分岐構造を作成します。

ステートメント一覧

ステートメント	説明
`break`	ループ (`for`、`for..in`、`do..while`、または `while`) 内で使用します。または、`switch` ステートメント内の特定のケースと関連するステートメントのブロック内でも使用します。
`case`	`switch` ステートメントの条件を定義します。
`class`	カスタムクラスを定義します。自分で定義したメソッドとプロパティを共有するオブジェクトをインスタンス化できます。
`continue`	ループの終わりまで制御が通過したかのように、最も内側のループ内の残りのステートメントをすべてスキップして、ループの次の反復を開始します。
`default`	`switch` ステートメントのデフォルトケースを定義します。
`delete`	*reference* パラメータで指定されたオブジェクト参照を破棄し、参照が正常に削除された場合は `true` を返します。それ以外の場合は `false` を返します。
`do..while`	`while` ループに似ていますが、条件の最初の評価に先立ってステートメントが実行される点が異なります。
`dynamic`	指定したクラスに基づくオブジェクトが、実行時にダイナミックプロパティを追加したり、ダイナミックプロパティにアクセスできるようにします。
`else`	`if` ステートメントの条件が `false` を返したときに実行されるステートメントを指定します。

ステートメント	説明
else if	条件を評価し、最初の if ステートメントの条件から false を返された場合に実行するステートメントを指定します。
extends	他のクラス (スーパークラス) のサブクラスであるクラスを定義します。
for	init (初期化) 式を 1 回だけ評価してから、ループシーケンスを開始します。
for..in	オブジェクトのプロパティまたは配列のエレメントに対して反復処理を行うもので、各プロパティまたはエレメントに対して *statement* を実行します。
function	何らかのタスクを実行するために定義する一連のステートメントで構成されます。
get	外部クラスファイルで定義したクラスに基づき、オブジェクトに関連付けられたプロパティを暗黙的に取得できるようにします。
if	条件を評価して、SWF ファイル内の次のアクションを決定します。
implements	実装するインターフェイスで宣言されているメソッドをクラスですべて定義する必要があることを指定します。
import	完全修飾名を指定しなくてもクラスにアクセスできるようにします。
interface	インターフェイスを定義します。
intrinsic	定義済みのクラスをコンパイル時にタイプチェックできるようにします。
private	変数や関数を宣言または定義しているクラス、またはそのクラスのサブクラスだけからその変数や関数にアクセスできるように指定します。
public	すべての呼び出し元から変数や関数にアクセスできるように指定します。
return	関数から返される値を指定します。
set	外部クラスファイルで定義したクラスに基づき、オブジェクトに関連付けられたプロパティを暗黙的に設定できるようにします。
set variable	変数に値を代入します。
static	変数や関数が、1 つのクラスを基に生成されるすべてのオブジェクトで生成されるのではなく、そのクラスで 1 回だけ生成されるようにします。
super	メソッドやコンストラクタのスーパークラスバージョンを呼び出します。
switch	ActionScript ステートメントの分岐構造を作成します。
throw	catch{} コードブロックによって処理 (キャッチ) できるエラーを生成 (スロー) します。
try..catch..finally	エラーが発生する可能性のあるコードブロックを囲み、そのエラーに対処します。
var	ローカル変数やタイムライン変数を宣言する場合に使用します。
while	条件を評価して、条件の評価結果が true になる場合はステートメントを実行します。その後、ループの先頭に戻り、再び条件を評価します。
with	*object* パラメータでムービークリップなどのオブジェクトを指定し、そのオブジェクト内の式やアクションを *statement(s)* パラメータで評価できるようにします。

break ステートメント

```
break
```

ループ (for、for..in、do..while、または while) 内で使用します。または、switch ステートメント内の特定のケースと関連するステートメントのブロック内でも使用します。break ステートメントをループ内で使用すると、ループ本体の残りの部分をスキップし、繰り返し処理を停止して、ループステートメントの次のステートメントを実行します。break ステートメントを switch 内で使用すると、case ブロック内の残りのステートメントをスキップし、囲んでいる switch ステートメントに続く最初のステートメントにジャンプします。

ネストされているループ内では、break ステートメントは、そのループの残りの部分をスキップするだけで、ネストされている一連のループは終了しません。一連のループを終了する方法については、try..catch..finally を参照してください。

対応バージョン： ActionScript 1.0、Flash Player 4

例

次の例では、break ステートメントを使用して無限ループから抜け出します。

```
var i:Number = 0;
while (true) {
 trace(i);
 if (i >= 10) {
 break; // this will terminate/exit the loop
 }
 i++;
}
```

このコードは、次の結果を表示します。

```
0
1
2
3
4
5
6
7
8
9
10
```

関連項目

for ステートメント

case ステートメント

```
case expression : statement(s)
```

switch ステートメントの条件を定義します。*expression* パラメータと、厳密な等価 (===) を使用している switch ステートメントの *expression* パラメータが等しい場合、break ステートメントが見つかるか、または switch ステートメントの終わりに到達するまで、*statement(s)* パラメータ内のステートメントが実行されます。

case ステートメントを switch ステートメントの外側で使用すると、エラーが発生し、スクリプトはコンパイルされません。

メモ：*statement(s)* パラメータは、必ず break ステートメントで終了する必要があります。*statement(s)* パラメータの break statement ステートメントを省略すると、switch ステートメントが終了せずに、次の case ステートメントが実行されます。

対応バージョン：ActionScript 1.0、Flash Player 4

パラメータ

expression:String - 任意の式。

例

次の例では、switch ステートメント thisMonth の条件を定義します。thisMonth が case ステートメント内の式と等しい場合は、ステートメントが実行されます。

```
var thisMonth:Number = new Date().getMonth();
switch (thisMonth) {
 case 0 :
 trace("January");
 break;
 case 1 :
 trace("February");
 break;
 case 5 :
 case 6 :
 case 7 :
 trace("Some summer month");
 break;
 case 8 :
 trace("September");
 break;
 default :
 trace("some other month");
}
```

関連項目

break ステートメント

class ステートメント

```
[dynamic] class className [ extends superClass ] [ implements interfaceName
  [, interfaceName... ] ] {
 // class definition here
}
```

カスタムクラスを定義します。自分で定義したメソッドとプロパティを共有するオブジェクトをインスタンス化できます。たとえば、送り状追跡システムを作成する場合に、送り状クラスを定義して、各送り状で必要になるすべてのメソッドとプロパティを定義します。その後、new invoice() コマンドを使用して、送り状オブジェクトを作成します。

クラスの名前は、そのクラスが格納された外部ファイルの名前と一致している必要があります。外部ファイルの名前は、クラス名にファイル拡張子 .as が付けられた名前である必要があります。たとえば、あるクラスに Student という名前を付ける場合、このクラスを定義するファイルの名前は Student.as になります。

クラスがパッケージ化されている場合、クラス宣言には base.sub1.sub2.MyClass という形式で完全修飾されたクラス名を使用する必要があります。さらに、パッケージ構造を反映するディレクトリ構造のパス内にクラスの AS ファイルを保存する必要があります (たとえば base/sub1/sub2/MyClass.as)。クラス定義の形式が "class MyClass" の場合、デフォルトパッケージに含まれ、パス内の所定のディレクトリの最上位レベルに MyClass.as ファイルを置く必要があります。

そのため、クラスを作成する前に、どのようなディレクトリ構造にするかあらかじめ決めておくことをお勧めします。そうしないと、クラスファイルを作成した後に移動する場合、新しい場所を反映させるためにクラス宣言ステートメントを修正する必要が生じます。

クラス定義はネストできません。つまり、クラス定義内に別のクラスを定義することはできません。

実行時にオブジェクトがダイナミックプロパティを追加したり、ダイナミックプロパティにアクセスしたりできるようにする場合は、クラスステートメントの前に dynamic キーワードを付加します。インターフェイスを実装するクラスを宣言するには、implements キーワードを使用します。クラスのサブクラスを作成するには、extends キーワードを使用します。クラスは 1 つしか拡張できませんが、インターフェイスは複数実装することができます。implements キーワードと extends キーワードは、1 つのステートメントで一緒に使うことができます。次の例に、implements キーワードと extends キーワードの一般的な使い方を示します。

```
class C implements Interface_i, Interface_j // OK
class C extends Class_d implements Interface_i, Interface_j // OK
class C extends Class_d, Class_e // not OK
```

対応バージョン： ActionScript 2.0、Flash Player 6

パラメータ

className:String - クラスの完全修飾名。

例

次の例では、Plant という名前のクラスを作成しています。Plant コンストラクタは 2 つのパラメータを受け取ります。

```
// Filename Plant.as
class Plant {
 // Define property names and types
 var leafType:String;
 var bloomSeason:String;
 // Following line is constructor
 // because it has the same name as the class
 function Plant(param_leafType:String, param_bloomSeason:String) {
 // Assign passed values to properties when new Plant object is created
 this.leafType = param_leafType;
 this.bloomSeason = param_bloomSeason;
 }
 // Create methods to return property values, because best practice
 // recommends against directly referencing a property of a class
 function getLeafType():String {
 return leafType;
 }
 function getBloomSeason():String {
 return bloomSeason;
 }
}
```

外部のスクリプトファイルまたは [アクション] パネルで、new 演算子を使用して Plant オブジェクトを作成します。

```
var pineTree:Plant = new Plant("Evergreen", "N/A");
// Confirm parameters were passed correctly
trace(pineTree.getLeafType());
trace(pineTree.getBloomSeason());
```

次の例では、ImageLoader というクラスを作成しています。ImageLoader コンストラクタは 3 つのパラメータを受け取ります。

```
// Filename ImageLoader.as
class ImageLoader extends MovieClip {
 function ImageLoader(image:String, target_mc:MovieClip, init:Object) {
 var listenerObject:Object = new Object();
 listenerObject.onLoadInit = function(target) {
 for (var i in init) {
 target[i] = init[i];
 }
 };
 var JPEG_mcl:MovieClipLoader = new MovieClipLoader();
 JPEG_mcl.addListener(listenerObject);
 JPEG_mcl.loadClip(image, target_mc);
 }
}
```

外部のスクリプトファイルまたは [アクション] パネルで、new 演算子を使用して ImageLoader オブジェクトを作成します。

```
var jakob_mc:MovieClip = this.createEmptyMovieClip("jakob_mc",
  this.getNextHighestDepth());
var jakob:ImageLoader = new ImageLoader("http://www.helpexamples.com/flash/
  images/image1.jpg", jakob_mc, {_x:10, _y:10, _alpha:70, _rotation:-5});
```

関連項目

dynamic ステートメント

continue ステートメント

```
continue
```

ループの終わりまで制御が通過したかのように、最も内側のループ内の残りのステートメントをすべてスキップして、ループの次の反復を開始します。ループの外部では作用しません。

対応バージョン： ActionScript 1.0、Flash Player 4

例

次の while ループで continue を使用すると、Flash インタプリタはループ本体の残りの部分をスキップし、ループの上端にジャンプします。そこで、条件が再度評価されます。

```
trace("example 1");
var i:Number = 0;
while (i < 10) {
 if (i % 3 == 0) {
 i++;
 continue;
 }
 trace(i);
 i++;
}
```

次の do..while ループで continue を使用すると、Flash インタプリタはループ本体の残りの部分をスキップし、ループの下端にジャンプします。そこで、条件が評価されます。

```
trace("example 2");
var i:Number = 0;
do {
 if (i % 3 == 0) {
 i++;
 continue;
 }
 trace(i);
 i++;
}
while (i < 10);
```

for ループで continue を使用すると、Flash インタプリタはループ本体の残りの部分をスキップします。次の例では、3 を法とする i が 0 に等しい場合、trace(i) ステートメントはスキップされます。

```
trace("example 3");
for (var i = 0; i < 10; i++) {
 if (i % 3 == 0) {
 continue;
 }
 trace(i);
}
```

次の for..in ループで continue を使用すると、Flash インタプリタはループ本体の残りの部分をスキップし、ループの上端にジャンプし、そこに列挙されている次の値が処理されます。

```
for (i in _root) {
 if (i == "$version") {
 continue;
 }
 trace(i);
}
```

関連項目

do..while ステートメント

default ステートメント

default: *statements*

switch ステートメントのデフォルトケースを定義します。このステートメントが実行されるのは、switch ステートメントの *expression* パラメータが、所定の switch ステートメントの case キーワードに続く *expression* パラメータと等しくない場合です (厳密な等価 [===] を使用)。

switch ステートメントにおいて default ケースステートメントは必須ではありません。default ケースステートメントは、リストの最後に置く必要はありません。default ステートメントを switch ステートメントの外側で使用すると、エラーが発生し、スクリプトはコンパイルされません。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

statements: String - 任意のステートメント。

例

次の例では、式 A は式 B とも式 D とも等しくありません。したがって、default キーワードに続くステートメントが実行され、trace() ステートメントが [出力] パネルに送られます。

```
var dayOfWeek:Number = new Date().getDay();
switch (dayOfWeek) {
 case 1 :
 trace("Monday");
 break;
 case 2 :
 trace("Tuesday");
 break;
 case 3 :
 trace("Wednesday");
 break;
 case 4 :
 trace("Thursday");
 break;
 case 5 :
 trace("Friday");
 break;
 default :
 trace("Weekend");
}
```

関連項目

switch ステートメント

delete ステートメント

```
delete reference
```

reference パラメータで指定されたオブジェクト参照を破棄し、参照が正常に削除された場合は true を返します。それ以外の場合は false を返します。この演算子はスクリプトが使用しているメモリを解放するときに役立ちます。delete 演算子を使用してオブジェクトへの参照を削除できます。オブジェクトへの参照がすべて削除されると、オブジェクトの削除と、そのオブジェクトが使用していたメモリの解放が行われます。

delete は演算子ですが、通常は次のようにステートメントとして使用します。

```
delete x;
```

reference パラメータが存在しない場合、または削除できない場合には、delete 演算子は処理ができないので false を返します。定義済みオブジェクトとプロパティは削除できません。また、var ステートメントを使って関数内で宣言した変数も削除できません。delete 演算子を使用してムービークリップを削除することはできません。

対応バージョン： ActionScript 1.0、Flash Player 5

戻り値

Boolean - ブール値。

パラメータ

reference:Object - 消去する変数またはオブジェクトの名前。

例

シンタックス 1: 次の例では、オブジェクトを作成し、使用してから、不要になったそのオブジェクトを削除します。

```
var account:Object = new Object();
account.name = "Jon";
account.balance = 10000;
trace(account.name); //output: Jon
delete account;
trace(account.name); //output: undefined
```

シンタックス 2: 次の例では、オブジェクトのプロパティを削除します。

```
// create the new object "account"
var account:Object = new Object();
// assign property name to the account
account.name = "Jon";
// delete the property
delete account.name;
```

シンタックス 3: 次の例では、オブジェクトのプロパティを削除します。

```
var my_array:Array = new Array();
my_array[0] = "abc"; // my_array.length == 1
my_array[1] = "def"; // my_array.length == 2
my_array[2] = "ghi"; // my_array.length == 3
// my_array[2] is deleted, but Array.length is not changed
delete my_array[2];
trace(my_array.length); // output: 3
trace(my_array); // output: abc,def,undefined
```

シンタックス 4: 次の例では、オブジェクト参照を delete で削除します。

```
var ref1:Object = new Object();
ref1.name = "Jody";
// copy the reference variable into a new variable
// and delete ref1
ref2 = ref1;
delete ref1;
trace("ref1.name "+ref1.name); //output: ref1.name undefined
trace("ref2.name "+ref2.name); //output: ref2.name Jody
```

ref1 を ref2 にコピーしていなければ、ref1 を削除したときにオブジェクトへの参照が存在しなくなるので、オブジェクトは削除されることになります。ref2 を削除すると、そのオブジェクトへの参照が存在しなくなるのでオブジェクトが破棄され、オブジェクトによって使用されていたメモリが使用可能になります。

シンタックス 5：次の例では、delete から返されるブール値をその後のコード実行の条件として使用する方法を示します。アイテムが既に削除済みであれば、そのアイテムに対して再び delete を呼び出すと false が返されます。

```
var my_array:Array = [ "abc", "def", "ghi" ];
var deleteWasSuccessful:Boolean

deleteWasSuccessful = delete my_array[0];
if(deleteWasSuccessful) delete my_array[1];
deleteWasSuccessful = delete my_array[0];
if(deleteWasSuccessful) delete my_array[2];

trace(my_array) // outputs: undefined,undefined,ghi
```

関連項目

var ステートメント

do..while ステートメント

do { *statement(s)* } while (*condition*)

while ループに似ていますが、条件の最初の評価に先立ってステートメントが実行される点が異なります。その後、ステートメントは、条件が true と評価された場合だけ実行されます。

do..while ループでは、ループ内のコードが少なくとも 1 回は必ず実行されます。while ループを使って、実行するステートメントのコピーを while ループの開始前に配置することで同じ操作を実現できますが、多くのプログラマは do..while ループの方が読みやすいと考えています。

条件が常に true と評価されると、do..while ループは無限ループになります。無限ループに陥ると、Flash Player に問題が発生し、警告メッセージが出力されたり、プレーヤーがクラッシュすることがあります。ループの回数が分かっている場合は、できる限り for ループを使用してください。for ループは読みやすくデバッグも簡単ですが、あらゆる状況で do..while に代えて使用できるわけではありません。

対応バージョン： ActionScript 1.0、Flash Player 4

パラメータ

condition:Boolean - 評価する条件。コードの do ブロック内の *statement(s)* は、*condition* パラメータの評価が true である限り実行されます。

例

次の例では、do..while ループを使用して条件が true, かどうかを評価し、myVar が 5 よりも大きくなるまで myVar をトレースします。myVar が 5 よりも大きくなると、ループは終了します。

```
var myVar:Number = 0;
do {
 trace(myVar);
 myVar++;
}
while (myVar < 5);
/* output:
0
1
2
3
4
*/
```

関連項目

break ステートメント

dynamic ステートメント

```
dynamic class className [ extends superClass ] [ implements interfaceName
  [, interfaceName... ] ] {
 // class definition here
}
```

指定したクラスに基づくオブジェクトが、実行時にダイナミックプロパティを追加したり、ダイナミックプロパティにアクセスできるようにします。

クラス定義内およびクラスインスタンス内部でアクセスされるメンバーは、クラススコープ内のメンバーと比較されないので、ダイナミッククラスに対するタイプチェックは、非ダイナミッククラスの場合よりも緩くになります。ただし、その場合でもクラスメンバー関数の戻り値とパラメータのタイプに対しては、タイプチェックが実行されます。この動作は、MovieClip オブジェクトを操作する場合に特に便利です。ムービークリップにプロパティやオブジェクトを動的に追加する方法が数多くあります。具体的には、MovieClip.createEmptyMovieClip() や MovieClip.createTextField() などです。

ダイナミッククラスのサブクラスもダイナミックになります。

次に示すように、オブジェクトを宣言するときには必ずタイプを指定します。

```
var x:MyClass = new MyClass();
```

次に示すように、オブジェクト宣言時に型を指定しない場合、そのオブジェクトはダイナミックであると見なされます。

```
var x = new MyClass();
```

対応バージョン： ActionScript 2.0、Flash Player 6

例

次の例では、クラス Person2 が dynamic キーワードで宣言されていないので、このクラスで宣言されていない関数を呼び出すと、コンパイルエラーが発生します。

```
class Person2 {
 var name:String;
 var age:Number;
 function Person2(param_name:String, param_age:Number) {
 trace ("anything");
 this.name = param_name;
 this.age = param_age;
 }
}
```

同じディレクトリ内にある FLA または AS ファイル内で、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
// Before dynamic is added
var craig:Person2 = new Person2("Craiggers", 32);
for (i in craig) {
 trace("craig." + i + " = " + craig[i]);
}
/* output:
craig.age = 32
craig.name = Craiggers */
```

宣言されていない関数 dance を追加すると、次に示すように、エラーが生成されます。

```
trace("");
craig.dance = true;
for (i in craig) {
 trace("craig." + i + " = " + craig[i]);
}
/* output: **Error** Scene=Scene 1, layer=Layer 1, frame=1:Line 14: There is no
   property with the name 'dance'. craig.dance = true; Total ActionScript Errors:
   1 Reported Errors: 1 */
```

dynamic キーワードを Person2 クラスに追加します。最初の行は次のようになります。

```
dynamic class Person2 {
```

コードを再びテストします。出力は次のようになります。

```
craig.dance = true craig.age = 32 craig.name = Craiggers
```

関連項目

class ステートメント

else ステートメント

```
if (condition){
statement(s);
} else {
statement(s);
}
```

if ステートメントの条件が false を返したときに実行されるステートメントを指定します。else ステートメントによって実行するステートメントブロックを囲む中括弧 ({}) は、実行するステートメントが 1 つしかない場合は不要です。

対応バージョン： ActionScript 1.0、Flash Player 4

パラメータ

condition:Boolean - 評価結果が true または false になる式。

例

次の例では、else 条件を使用して、age_txt 変数が 18 より大きいかまたは小さいかを判定しています。

```
if (age_txt.text>=18) {
 trace("welcome, user");
}
else {
 trace("sorry, junior");
 userObject.minor = true;
 userObject.accessAllowed = false;
}
```

次の例では、else ステートメントに続くステートメントが 1 つだけなので、中括弧 ({}) は必要ありません。

```
if (age_txt.text>18) { trace("welcome, user"); } else trace("sorry, junior");
```

関連項目

if ステートメント

else if ステートメント

```
if(condition) {
statement(s);
} else if(condition) {
statement(s);
}
```

条件を評価し、最初の if ステートメントの条件から false を返された場合に実行するステートメントを指定します。else if 条件から true が返されると、Flash インタプリタは中括弧 ({ }) 内の条件の後にあるステートメントを実行します。else if 条件が false の場合は、中カッコ内のステートメントはスキップされ、中カッコの後のステートメントが実行されます。

スクリプト内に分岐処理を作成するには else if ステートメントを使用します。分岐が複数ある場合は、switch ステートメントの使用を検討してください。

対応バージョン： ActionScript 1.0、Flash Player 4

パラメータ

condition:Boolean - 評価結果が true または false になる式。

例

次の例では、else if ステートメントを使用して、score_txt を指定された値と比較しています。

```
if (score_txt.text>90) {
 trace("A");
}
else if (score_txt.text>75) {
 trace("B");
}
else if (score_txt.text>60) {
 trace("C");
}
else {
 trace("F");
}
```

関連項目

if ステートメント

extends ステートメント

```
class className extends otherClassName {}
interface interfaceName extends otherInterfaceName {}
```

他のクラス (スーパークラス) のサブクラスであるクラスを定義します。サブクラスは、スーパークラスで定義されているメソッド、プロパティ、関数などをすべて継承します。

インターフェイスは extends キーワードを使用して拡張することもできます。他のインターフェイスを拡張したインターフェイスには、元のインターフェイスのメソッド宣言がすべて含まれます。

対応バージョン : ActionScript 2.0、Flash Player 6

パラメータ

className:String - 定義するクラスの名前。

例

次の例の Car クラスは Vehicle クラスを拡張しているので、Vehicle クラスのすべてのメソッド、プロパティ、および関数を継承します。スクリプトで Car オブジェクトをインスタンス化すると、Car クラスのメソッドと Vehicle クラスのメソッドの両方を使用できます。

次の例では、Vehicle クラスを定義している Vehicle.as ファイルの内容を示します。

```
class Vehicle {
 var numDoors:Number;
 var color:String;
 function Vehicle(param_numDoors:Number, param_color:String) {
 this.numDoors = param_numDoors;
 this.color = param_color;
 }
 function start():Void {
 trace("[Vehicle] start");
 }
 function stop():Void {
 trace("[Vehicle] stop");
 }
 function reverse():Void {
 trace("[Vehicle] reverse");
 }
}
```

次の例では、同じディレクトリ内にあるもう 1 つの AS ファイル Car.as の内容を示します。このクラスは Vehicle クラスを拡張したクラスで、次のような 3 つの変更を行っています。Car クラスでは、最初に、車オブジェクトにフルサイズのスペアタイヤが装着されているかどうかを追跡するための変数 fullSizeSpare を追加しています。2 番目に、車の盗難防止アラームをアクティブにする activateCarAlarm() を、車に固有の新しいメソッドとして追加します。3 番目に、Car クラスでは停車の際にアンチロックブレーキシステムを使用することを示すために、stop() 関数が上書きされています。

```
class Car extends Vehicle {
 var fullSizeSpare:Boolean;
 function Car(param_numDoors:Number, param_color:String,
  param_fullSizeSpare:Boolean) {
 this.numDoors = param_numDoors;
 this.color = param_color;
 this.fullSizeSpare = param_fullSizeSpare;
 }
 function activateCarAlarm():Void {
 trace("[Car] activateCarAlarm");
 }
 function stop():Void {
 trace("[Car] stop with anti-lock brakes");
 }
}
```

次の例では、Car オブジェクトをインスタンス化し、Vehicle クラスに定義されているメソッド (`start()`) を呼び出します。次に、Car クラスによってオーバーライドされるメソッド (`stop()`) を呼び出し、最後に、Car クラスのメソッド (`activateCarAlarm()`) を呼び出します。

```
var myNewCar:Car = new Car(2, "Red", true);
myNewCar.start(); // output: [Vehicle] start
myNewCar.stop(); // output: [Car] stop with anti-lock brakes
myNewCar.activateCarAlarm(); // output: [Car] activateCarAlarm
```

Vehicle クラスのサブクラスは、`super` キーワードを使用して記述することもできます。サブクラスは、このキーワードを使用することでスーパークラスのプロパティやメソッドにアクセスできます。次の例では、同じディレクトリ内にある 3 番目の AS ファイル Truck.as の内容を示します。Truck クラスでは、コンストラクタ内と、上書きされた `reverse()` 関数内で `super` キーワードを使用しています。

```
class Truck extends Vehicle {
 var numWheels:Number;
 function Truck(param_numDoors:Number, param_color:String,
  param_numWheels:Number) {
 super(param_numDoors, param_color);
 this.numWheels = param_numWheels;
 }
 function reverse():Void {
 beep();
 super.reverse();
 }
 function beep():Void {
 trace("[Truck] make beeping sound");
 }
}
```

次の例では、Truck オブジェクトをインスタンス化し、Truck クラスによってオーバーライドされるメソッド (reverse()) を呼び出してから、Vehicle クラスで定義されているメソッド (stop()) を呼び出します。

```
var myTruck:Truck = new Truck(2, "White", 18);
myTruck.reverse(); // output: [Truck] make beeping sound [Vehicle] reverse
myTruck.stop(); // output: [Vehicle] stop
```

関連項目

class ステートメント

for ステートメント

```
for(init; condition; next) {
statement(s);
}
```

init (初期化) 式を 1 回だけ評価してから、ループシーケンスを開始します。ループシーケンスは、*condition* 式を評価することで開始されます。*condition* 式の評価結果が true の場合は、*statement* が実行され、*next* 式が評価されます。その後、*condition* 式の評価からループシーケンスが再び開始されます。

for ステートメントによって実行するステートメントブロックを囲む中括弧 ({}) は、実行するステートメントが 1 つしかない場合は不要です。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

init - ループの開始前に評価される式。通常は代入式です。このパラメータに対しては、var ステートメントも実行できます。

例

次の例では、for を使用して配列のエレメントを追加します。

```
var my_array:Array = new Array();
for (var i:Number = 0; i < 10; i++) {
 my_array[i] = (i + 5) * 10;
}
trace(my_array); // output: 50,60,70,80,90,100,110,120,130,140
```

次の例では、for を使って同じアクションを繰り返し実行します。次のコードでは、for ループにより 1 から 100 の数値を加算します。

```
var sum:Number = 0;
for (var i:Number = 1; i <= 100; i++) {
 sum += i;
}
trace(sum); // output: 5050
```

次の例では、実行するステートメントが1つしかない場合は中カッコを付ける必要がないことを示します。

```
var sum:Number = 0;
for (var i:Number = 1; i <= 100; i++)
 sum += i;
trace(sum); // output: 5050
```

関連項目

++ インクリメント演算子

for..in ステートメント

```
for (variableIterant in object) {
statement(s);
}
```

オブジェクトのプロパティまたは配列のエレメントに対して反復処理を行うもので、各プロパティまたはエレメントに対して *statement* を実行します。for..in アクションでは、オブジェクトのメソッドは列挙されません。

プロパティの中には、for..in アクションで列挙できないものがあります。たとえば、_x や _y のようなムービークリッププロパティは列挙されません。外部クラスファイルでは、インスタンスメンバーとは異なり、静的メンバーは列挙されません。

for..in ステートメントは、反復処理されるオブジェクトのプロトタイプチェーン内のオブジェクトのプロパティに対して反復します。オブジェクトのプロパティが最初に列挙され、次にそのプロトタイプのプロパティが列挙され、さらにそのプロトタイプのプロトタイプのプロパティが列挙されます。以下同様です。逆インデックス順序でプロパティが反復処理されます。for..in ステートメントは、同じプロパティ名を繰り返して列挙しません。オブジェクト child にプロトタイプ parent が存在し、両方に prop プロパティがある場合、child の for..in ステートメントは、child の prop プロパティは列挙しますが parent の prop プロパティは無視します。

for..in ステートメントによって実行するステートメントブロックを囲む中括弧 ({}) は、実行するステートメントが1つしかない場合は不要です。

for..in ループをクラスファイル (外部 AS ファイル) 内に記述した場合、インスタンスメンバーをループで使用することはできませんが、静的メンバーはループで使用できます。ただし、クラスのインスタンスの FLA ファイル内に for..in ループを記述した場合、インスタンスメンバーは使用できますが、静的メンバーは使用できません。

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

variableIterant:String - 反復子の役割を果たし、オブジェクトのプロパティまたは配列内のエレメントを参照する変数の名前。

例

次の例では、for..in を使用してオブジェクトのプロパティに対して反復処理を行います。

```
var myObject:Object = {firstName:"Tara", age:27, city:"San Francisco"};
for (var prop in myObject) {
 trace("myObject."+prop+" = "+myObject[prop]);
}
//output
myObject.firstName = Tara
myObject.age = 27
myObject.city = San Francisco
```

次の例では、for..in を使用して配列のエレメントに対して反復処理を行います。

```
var myArray:Array = new Array("one", "two", "three");
for (var index in myArray)
 trace("myArray["+index+"] = " + myArray[index]);
// output:
myArray[2] = three
myArray[1] = two
myArray[0] = one
```

次の例では、for..in で typeof 演算子を使用して特定のタイプの子に対して反復処理を行います。

```
for (var name in this) {
 if (typeof (this[name]) == "movieclip") {
 trace("I have a movie clip child named "+name);
 }
}
```

メモ：ムービークリップが複数ある場合、その出力はそれらのクリップのインスタンス名で構成されます。

次の例では、ムービークリップの子を列挙し、それぞれを対応するタイムラインのフレーム 2 に進めます。RadioButtonGroup ムービークリップは親であり、_RedRadioButton_、_GreenRadioButton_、および _BlueRadioButton_ という 3 つの子を持っています。

```
for (var name in RadioButtonGroup) { RadioButtonGroup[name].gotoAndStop(2); }
```

function ステートメント

```
Usage 1: (Declares a named function.)
function functionname([parameter0, parameter1,...parameterN]){
statement(s)
}
Usage 2: (Declares an anonymous function and returns a reference to it.)
function ([parameter0, parameter1,...parameterN]){
statement(s)
}
```

何らかのタスクを実行するために定義する一連のステートメントで構成されます。ある位置で関数を定義し、SWF ファイル内の異なるスクリプトからその関数を呼び出すことができます。関数を定義する場合、その関数のパラメータも指定できます。パラメータは、関数が処理する値のプレースホルダーです。関数を呼び出すたびに異なるパラメータを渡すことができます。これにより、1つの関数を多くの異なる状況で再利用できます。

関数に値を生成させる、つまり関数が値を "返す" ようにするには、関数の *statement(s)* で `return` ステートメントを使用します。

このステートメントを使用して、*functionname*、*parameters*、および *statement(s)* が指定された `function` を定義します。スクリプトによって関数が呼び出されると、関数定義内のステートメントが実行されます。関数は前方参照が許されます。つまり、同じスクリプト内では、関数を呼び出す箇所よりも後に関数を宣言できます。関数定義は、同じ関数の以前の定義と置き換わります。ステートメントが許されている場所であればどこでも、このシンタックスを使用できます。

この関数ステートメントを使用して匿名関数を作成し、その関数への参照を返すことができます。このシンタックスは式の中で使用され、特にオブジェクト内にメソッドを組み込む場合に便利です。

さらに、関数定義内で `arguments` オブジェクトを使用することができます。一般に、`arguments` オブジェクトは、可変数のパラメータを受け取る関数や再帰的な匿名関数を作成する場合に使用します。

対応バージョン： ActionScript 1.0、Flash Player 5

戻り値

`String` - シンタックス 1: この宣言形式では何も返されません。シンタックス 2: 匿名関数への参照。

パラメータ

`functionname:String` - 宣言した関数の名前。

例

次の例では、1つのパラメータを受け取る関数 sqr を定義し、そのパラメータの Math.pow(x, 2) を返します。

```
function sqr(x:Number) {
 return Math.pow(x, 2);
}
var y:Number = sqr(3);
trace(y); // output: 9
```

同じスクリプト内で関数を定義および使用する場合は、関数の使用箇所の後に関数定義を記述することもできます。

```
var y:Number = sqr(3);
trace(y); // output: 9
function sqr(x:Number) {
 return Math.pow(x, 2);
}
```

次の関数は、LoadVars オブジェクトを作成し、params.txt を SWF ファイルにロードします。ファイルが正常にロードされると、"variables loaded" と表示されます。

```
var myLV:LoadVars = new LoadVars();
myLV.load("params.txt");
myLV.onLoad = function(success:Boolean) {
 trace("variables loaded");
}
```

get ステートメント

```
function get property () {
 // your statements here
}
```

外部クラスファイルで定義したクラスに基づき、オブジェクトに関連付けられたプロパティを暗黙的に取得できるようにします。暗黙的な get メソッドを使用すると、プロパティに直接アクセスすることなく、オブジェクトのプロパティにアクセスできます。暗黙的な get メソッドと set メソッドは、ActionScript 1.0 の Object.addProperty() メソッドの簡易版です。

対応バージョン： ActionScript 2.0、Flash Player 6

パラメータ

property:String - get でアクセスするプロパティの参照名。この値は対応する set コマンドに使用した値と一致させる必要があります。

例

次の例では、Team クラスを定義します。Team クラスには、クラス内のプロパティを取得および設定するための get/set メソッドがあります。

```
class Team {
 var teamName:String;
 var teamCode:String;
 var teamPlayers:Array = new Array();
 function Team(param_name:String, param_code:String) {
 this.teamName = param_name;
 this.teamCode = param_code;
 }
 function get name():String {
 return this.teamName;
 }
 function set name(param_name:String):Void {
 this.teamName = param_name;
 }
}
```

次の ActionScript をタイムラインのフレームに入力します。

```
var giants:Team = new Team("San Fran", "SFO");
trace(giants.name);
giants.name = "San Francisco";
trace(giants.name);
/* output:
San Fran San Francisco */
```

giants.name をトレースするときは、get メソッドを使用してプロパティの値を返します。

関連項目

addProperty (Object.addProperty メソッド)

if ステートメント

```
if(condition) {
statement(s);
}
```

条件を評価して、SWF ファイル内の次のアクションを決定します。条件が true の場合は、条件の後にある中括弧 ({ }) 内のステートメントが実行されます。条件が false の場合は、中カッコ内のステートメントはスキップされ、中カッコの後のステートメントが実行されます。スクリプト内に分岐処理を作成するには、if ステートメントを else ステートメントや else if ステートメントと組み合わせます。

if ステートメントによって実行するステートメントブロックを囲む中括弧 ({ }) は、実行するステートメントが 1 つしかない場合は不要です。

対応バージョン： ActionScript 1.0、Flash Player 4

パラメータ

condition: Boolean - 評価結果が true または false になる式。

例

次の例では、カッコ内の条件で変数 name を評価し、リテラル値 "Erica" が含まれているかどうかを確認します。"Erica" が含まれている場合は、中カッコ内の play() 関数が実行されます。

```
if(name == "Erica"){
 play();
}
```

次の例では、if ステートメントを使用して、ユーザーが SWF ファイル内の submit_btn インスタンスをクリックするまでに要した時間を評価します。SWF ファイルの再生開始後 10 秒を超えてからユーザーがボタンをクリックした場合、条件は true と評価され、(createTextField() を使用して) 実行時に作成されたテキストフィールドに、中括弧 ({ }) 内のメッセージが表示されます。SWF ファイルの再生開始後 10 秒以内にユーザーがボタンをクリックした場合、条件は false と評価され、別のメッセージが表示されます。

```
this.createTextField("message_txt", this.getNextHighestDepth, 0, 0, 100, 22);
message_txt.autoSize = true;
var startTime:Number = getTimer();
this.submit_btn.onRelease = function() {
 var difference:Number = (getTimer() - startTime) / 1000;
 if (difference > 10) {
 this._parent.message_txt.text = "Not very speedy, you took "+difference+"
  seconds.";
 }
 else {
 this._parent.message_txt.text = "Very good, you hit the button in "+difference+"
  seconds.";
 }
};
```

関連項目

else ステートメント

implements ステートメント

myClass implements *interface01* [, *interface02* , ...]

実装するインターフェイスで宣言されているメソッドをクラスですべて定義する必要があることを指定します。

対応バージョン： ActionScript 2.0、Flash Player 6

例

interface を参照してください。

関連項目

class ステートメント

import ステートメント

```
import className
import packageName.*
```

完全修飾名を指定しなくてもクラスにアクセスできるようにします。たとえば、スクリプト内でカスタムクラス macr.util.users.UserClass を使用する場合、このクラスを完全修飾名で参照するか、import で読み込む必要があります。import で読み込んでおくと、クラス名での参照が可能になります。

```
// before importing
var myUser:macr.util.users.UserClass = new macr.util.users.UserClass();
// after importing
import macr.util.users.UserClass;
var myUser:UserClass = new UserClass();
```

アクセス対象のクラスファイルがパッケージ (*working_directory* /macr/utils/users) 内に複数存在する場合は、次の例に示すように、1 つのステートメントですべてのクラスファイルを読み込むことができます。

```
import macr.util.users.*;
```

import ステートメントは最初に指定しておく必要があります。そうすると、このステートメントで読み込まれたクラスは以降、完全修飾名を使わずにアクセスできます。

読み込んだクラスがスクリプト内で使用されなかった場合、そのクラスは SWF ファイルには出力されません。したがって、SWF ファイルのサイズを気にすることなく、大きなパッケージを読み込むことができます。クラスに関連付けられたバイトコードは、それが実際に使用された場合にのみ、SWF ファイルに含まれます。

import ステートメントは、それを呼び出している現在のスクリプト (フレームまたはオブジェクト) にのみ適用されます。たとえば、macr.util パッケージのすべてのクラスを Flash ドキュメントのフレーム 1 に読み込んだと仮定します。そのフレームでは、そのパッケージ内のクラスを簡単な名前で参照できます。

```
// On Frame 1 of a FLA:
import macr.util.*;
var myFoo:foo = new foo();
```

ただし、他のフレームのスクリプトでは、そのパッケージのクラスを完全修飾名 (`var myFoo:foo = new macr.util.foo();`) で参照するか、そのパッケージのクラスを読み込む import ステートメントを追加する必要があります。

対応バージョン : ActionScript 2.0、Flash Player 6

パラメータ

`className:String` - 外部クラスファイルで定義したクラスの完全修飾名。

例

interface ステートメント

`interface` *`InterfaceName`* `[extends` *`InterfaceName`* `] {}`

インターフェイスを定義します。インターフェイスはクラスに似ていますが、次に示す重要な違いがあります。

- インターフェイスにはメソッドの宣言だけが含まれます。メソッド実装は含まれません。つまり、インターフェイスを実装するすべてのクラスは、インターフェイスで宣言されている各メソッドの実装を定義する必要があります。
- インターフェイス定義では、パブリックメンバーしか使用できず、インスタンスおよびクラスのメンバーは使用できません。
- get ステートメントおよび set ステートメントは、インターフェイス定義では使用できません。

対応バージョン : ActionScript 2.0、Flash Player 6

例

次の例では、インターフェイスを定義および実装する方法をいくつか示しています。

```
(in top-level package .as files Ia, B, C, Ib, D, Ic, E)
// filename Ia.as
interface Ia {
 function k():Number; // method declaration only
 function n(x:Number):Number; // without implementation
}
```

```
// filename B.as
class B implements Ia {
 function k():Number {
 return 25;
 }
 function n(x:Number):Number {
 return x + 5;
 }
} // external script or Actions panel // script file
var mvar:B = new B();
trace(mvar.k()); // 25
trace(mvar.n(7)); // 12
// filename c.as
class C implements Ia {
 function k():Number {
 return 25;
 }
} // error: class must implement all interface methods
// filename Ib.as
interface Ib {
 function o():Void;
}
class D implements Ia, Ib {
 function k():Number {
 return 15;
 }
 function n(x:Number):Number {
 return x * x;
 }
 function o():Void {
 trace("o");
 }
} // external script or Actions panel // script file
mvar = new D();
trace(mvar.k()); // 15
trace(mvar.n(7)); // 49
trace(mvar.o()); // "o"
interface Ic extends Ia {
 function p():Void;
}
class E implements Ib, Ic {
 function k():Number {
 return 25;
 }
 function n(x:Number):Number {
 return x + 5;
 }
 function o():Void {
 trace("o");
 }
```

```
 function p():Void {
 trace("p");
 }
}
```

関連項目

class ステートメント

intrinsic ステートメント

```
intrinsic class className [extends superClass] [implements interfaceName [,
   interfaceName...] ] {
 //class definition here
}
```

定義済みのクラスをコンパイル時にタイプチェックできるようにします。Flash では、intrinsic クラス宣言を使用することで、Array、Object、および String といったビルトインクラスのタイプチェックをコンパイル時に行えます。このキーワードは、関数の実装が必要でないこと、およびバイトコードを生成しないことをコンパイラに示します。

intrinsic キーワードは、変数や関数の宣言に使用することもできます。Flash では、このキーワードを使用することで、グローバル関数やグローバルプロパティのタイプチェックをコンパイル時に行えます。

intrinsic キーワードは、ビルトインクラス、ビルドインオブジェクト、グローバル変数、グローバル関数のタイプチェックをコンパイル時に行えるようにする目的で作成されたものです。このキーワードは汎用的に使用するというものではなく、定義済みのクラス (特に ActionScript 1.0 を使用して定義されているクラス) をコンパイル時にタイプチェックする方法を模索しているデベロッパーにとって有用なものです。

このキーワードは、[アクション] パネルに記述するスクリプトではありません。外部スクリプトファイルで使用する場合にのみサポートされます。

対応バージョン : ActionScript 2.0、Flash Player 6

例

次の例では、定義済みの ActionScript 1.0 クラスのコンパイル時タイプチェックを有効にしています。このコードでは、myCircle.setRadius() 呼び出しにより Number 値ではなく String 値がパラメータとして送られるので、コンパイル時エラーが発生します。パラメータを Number 値に (たとえば "10" を 10 に) 変換することで、このエラーを回避できます。

```
// The following code must be placed in a file named Circle.as
// that resides within your classpath:
intrinsic class Circle {
 var radius:Number;
 function Circle(radius:Number);
 function getArea():Number;
 function getDiameter():Number;
 function setRadius(param_radius:Number):Number;
}

// This ActionScript 1.0 class definition may be placed in your FLA file.
// Circle class is defined using ActionScript 1.0
function Circle(radius) {
 this.radius = radius;
 this.getArea = function(){
 return Math.PI*this.radius*this.radius;
 };
 this.getDiameter = function() {
 return 2*this.radius;
 };
 this.setRadius = function(param_radius) {
 this.radius = param_radius;
 }
}

// ActionScript 2.0 code that uses the Circle class
var myCircle:Circle = new Circle(5);
trace(myCircle.getArea());
trace(myCircle.getDiameter());
myCircle.setRadius("10");
trace(myCircle.radius);
trace(myCircle.getArea());
trace(myCircle.getDiameter());
```

関連項目

class ステートメント

private ステートメント

```
class someClassName{
 private var name;
 private function name() {
 // your statements here
 }
}
```

変数や関数を宣言または定義しているクラス、またはそのクラスのサブクラスだけからその変数や関数にアクセスできるように指定します。デフォルトでは、すべての呼び出し元から変数や関数にアクセスできます。このキーワードは、変数や関数へのアクセスを制限する場合に使用します。このキーワードの目的は、ソフトウェア開発を支援して最適なコーディング方法を促進する (カプセル化など) ことであり、セキュリティメカニズムとして機密データのセキュリティを確保することではありません。実行時に変数へのアクセスを常に制限するとは限りません。

このキーワードは、クラス定義でのみ使用できます。インターフェイス定義では使用できません。

対応バージョン： ActionScript 2.0、Flash Player 6

パラメータ

`name:String` - private として指定する変数または関数の名前。

例

次の例では、`private` キーワードを使用して、変数や関数へのアクセスを制限する方法を示します。Alpha.as という新しい AS ファイルを作成します。

```
class Alpha {
 private var privateProperty = "visible only within class and subclasses";
 public var publicProperty = "visible everywhere";
}
```

Alpha.as と同じディレクトリに、次のコードを含む Beta.as という名前の新しい AS ファイルを作成します。

```
class Beta extends Alpha {
 function Beta() {
 trace("privateProperty is " + privateProperty);
 }
}
```

次のコードに示すように、この Beta クラスのコンストラクタは、Alpha クラスから継承された `privateProperty` プロパティにアクセスできます。

```
var myBeta:Beta = new Beta(); // Output: privateProperty is visible only within
  class and subclasses
```

`privateProperty` 変数に、Alpha クラスの外部または Alpha クラスから継承されたクラスの外部からアクセスしようとすると、エラーになります。次のコードは、あらゆるクラスの外部にあるため、エラーになります。

```
trace(myBeta.privateProperty); // Error
```

関連項目

`public` ステートメント

public ステートメント

```
class someClassName{
 public var name;
 public function name() {
 // your statements here
 }
}
```

すべての呼び出し元から変数や関数にアクセスできるように指定します。デフォルトでは変数および関数が public として扱われるため、このキーワードは主に形式上の目的で使用します。たとえば、private または static の変数が含まれるコードブロックで public を明確に区別したい場合などが該当します。

対応バージョン： ActionScript 2.0、Flash Player 6

パラメータ

`name:String` - public として指定する変数または関数の名前。

例

次の例では、クラスファイルでパブリック変数を使用する方法を示します。"User.as" という名前で新しいクラスファイルを作成し、次のコードを入力します。

```
class User {
 public var age:Number;
 public var name:String;
}
```

次に、同じディレクトリに新しい FLA または AS ファイルを作成し、タイムラインのフレーム 1 に次の ActionScript を入力します。

```
import User;
var jimmy:User = new User();
jimmy.age = 27;
jimmy.name = "jimmy";
```

User クラスのパブリック変数の 1 つをプライベート変数に変更した場合は、プロパティにアクセスしようとするとエラーが生成されます。

関連項目

private ステートメント

return ステートメント

`return[expression]`

関数から返される値を指定します。return ステートメントは、*expression* を評価し、その結果をステートメントが実行される関数の値として返します。return ステートメントにより、実行は即座に呼び出し元の関数に返されます。return ステートメントを単独で使用したときの戻り値は undefined. です。

複数の値を取得することはできません。複数の値を取得しようとした場合は、最後の値だけが返されます。たとえば、次の例では c が返されます。

```
return a, b, c ;
```

複数の値を取得する必要がある場合は、代わりに配列またはオブジェクトを使用します。

対応バージョン： ActionScript 1.0、Flash Player 5

戻り値

String - *expression* パラメータ (指定されている場合) の評価。

パラメータ

expression - 関数の値として評価して返すストリング、数値、ブール値、配列、またはオブジェクト。このパラメータはオプションです。

例

次の例では、sum( ) 関数の本体内で return ステートメントを使用し、3 つのパラメータの加算結果を返します。コードの 2 行目で sum( ) 関数を呼び出し、戻り値を変数 newValue に割り当てます。

```
function sum(a:Number, b:Number, c:Number):Number {
 return (a + b + c);
}
var newValue:Number = sum(4, 32, 78);
trace(newValue); // output: 114
```

関連項目

function ステートメント

set ステートメント

```
function set property(varName) {
 // your statements here
}
```

外部クラスファイルで定義したクラスに基づき、オブジェクトに関連付けられたプロパティを暗黙的に設定できるようにします。set メソッドを暗黙的に使用することにより、オブジェクトのプロパティに直接アクセスせずにプロパティの値を変更できます。暗黙的な get メソッドと set メソッドは、ActionScript 1.0 の `Object.addProperty()` メソッドの簡易版です。

対応バージョン： ActionScript 2.0、Flash Player 6

パラメータ

`property:String` - set でアクセスするプロパティの参照名。この値は対応する get コマンドで使用する値と一致させる必要があります。

例

次の例では、Login クラスを作成し、set キーワードを使用してプライベート変数を設定する方法を示します。

```
class Login {
 private var loginUserName:String;
 private var loginPassword:String;
 public function Login(param_username:String, param_password:String) {
 this.loginUserName = param_username;
 this.loginPassword = param_password;
 }
 public function get username():String {
 return this.loginUserName;
 }
 public function set username(param_username:String):Void {
 this.loginUserName = param_username;
 }
 public function set password(param_password:String):Void {
 this.loginPassword = param_password;
 }
}
```

Login.as と同じディレクトリにある FLA ファイルまたは AS ファイルで、タイムラインのフレーム 1 に次の ActionScript を入力します。

```
var gus:Login = new Login("Gus", "Smith");
trace(gus.username); // output: Gus
gus.username = "Rupert";
trace(gus.username); // output: Rupert
```

この例では、値がトレースされると get 関数が実行されます。set 関数は、次のように値が渡される場合にのみトリガされます。

```
gus.username = "Rupert";
```

関連項目

get ステートメント

set variable ステートメント

set("*variableString*",*expression*)

変数に値を代入します。変数とは、データを格納する容器のようなものです。容器そのものは変化しませんが、その内容は変化します。SWF ファイルの再生時に変数の値を変更すると、ユーザーが実行した処理に関する情報を記録および保存できます。また、SWF ファイルの再生時に変化した値を記録し、条件が true (真) なのか false (偽) なのかを評価することもできます。

変数には、ストリング、数値、ブール値、オブジェクト、ムービークリップなどのすべてのデータ型を保持できます。各 SWF ファイルおよびムービークリップのタイムラインには、それぞれ独自の変数のセットがあり、各変数の値は他のタイムラインの変数には影響されません。

set ステートメント内では厳密な型指定はサポートされません。このステートメントを使用して、クラスファイル内の変数に関連付けられたデータ型と異なるデータ型を持つ値を変数に設定した場合、コンパイラエラーは生成されません。

variableString パラメータは、変数名ではなくストリングです。この違いは重要なので注意が必要です。最初のパラメータとして既存の変数名を set( ) に渡すときに引用符 ("") で囲まないと、変数の評価結果が *expression* の値として代入されます。たとえば、myVariable という名前のストリング変数を作成し、"Tuesday" という値を代入するとします。このときに不注意で引用符を使用しないと、新しい変数 Tuesday が作成され、myVariable に代入しようとした値がその変数に代入されることになります。

```
var myVariable:String = "Tuesday";
set (myVariable, "Saturday");
trace(myVariable); // outputs Tuesday
trace(Tuesday); // outputs Saturday
```

引用符 ("") を使用すれば、この問題を回避できます。

```
set ("myVariable", "Saturday");
trace(myVariable); //outputs Saturday
```

対応バージョン： ActionScript 1.0、Flash Player 4

パラメータ

variableString:String - *expression* パラメータの値を保持する変数を指定するストリング。

例

次の例では、変数に値を代入します。"Jakob" という値を name 変数に割り当てます。

```
set("name", "Jakob");
trace(name);
```

次のコードは 3 回ループして、3 つの新しい変数 (caption0、caption1、caption2) を作成します。

```
for (var i = 0; i < 3; i++) {
 set("caption" + i, "this is caption " + i);
}
trace(caption0);
trace(caption1);
trace(caption2);
```

関連項目

var ステートメント

static ステートメント

```
class someClassName{
 static var name;
 static function name() {
 // your statements here
 }
}
```

変数や関数が、1 つのクラスを基に生成されるすべてのオブジェクトで生成されるのではなく、そのクラスで 1 回だけ生成されようにします。

シンタックス someClassName.name を使用すると、クラスのインスタンスを作成せずに静的クラスメンバーにアクセスできます。クラスのインスタンスを作成する場合は、そのインスタンスを使用して静的メンバーにアクセスすることもできます。ただし、その静的メンバーにアクセスする非静的関数を使用する場合のみに限定されます。

このキーワードは、クラス定義でのみ使用できます。インターフェイス定義では使用できません。

対応バージョン： ActionScript 2.0、Flash Player 6

パラメータ

name:String - static として指定する変数または関数の名前。

例

次の例では、作成されたクラスのインスタンス数を追跡するカウンタを、static キーワードを使用して作成する方法を示します。numInstances は静的変数なので、個々のインスタンスごとに作成されるのではなく、クラス全体で 1 回のみ作成されます。"Users.as" という新しい AS ファイルを作成し、次のコードを入力します。

```
class Users {
 private static var numInstances:Number = 0;
 function Users() {
 numInstances++;
 }
 static function get instances():Number {
 return numInstances;
 }
}
```

同じディレクトリに FLA ドキュメントまたは AS ドキュメントを作成し、タイムラインのフレーム 1 に次の ActionScript を入力します。

```
trace(Users.instances);
var user1:Users = new Users();
trace(Users.instances);
var user2:Users = new Users();
trace(Users.instances);
```

関連項目

private ステートメント

super ステートメント

```
super.method([arg1, ..., argN])
super([arg1, ..., argN])
```

最初のシンタックススタイルは、オブジェクトメソッドの本体内で使用して、メソッドのスーパークラスバージョンを呼び出すことができます。必要に応じて、スーパークラスメソッドにパラメータ (arg1 ... argN) を渡すこともできます。このスタイルは、スーパークラスのメソッドにビヘイビアを追加するだけでなく、そのメソッドを使用して元のビヘイビアを実行するようなサブクラスメソッドを作成する場合に便利です。

2 番目のシンタックススタイルは、コンストラクタ関数の本体内で使用して、コンストラクタ関数のスーパークラスバージョンを呼び出すことができます。必要に応じて、関数にパラメータを渡すこともできます。このスタイルは、補足的な初期化を行うだけでなく、スーパークラスのコンストラクタを呼び出してスーパークラスの初期化を行うようなサブクラスを作成する場合に便利です。

対応バージョン： ActionScript 1.0、Flash Player 6

戻り値

両方の形式も関数を呼び出します。関数は不特定の値を返します。

パラメータ

`method:Function` - スーパークラスで呼び出すメソッド。

`argN` - メソッドのスーパークラスバージョン (シンタックス 1) またはスーパークラスのコンストラクタ関数 (シンタックス 2) に渡すオプションのパラメータ。

switch ステートメント

```
switch (expression){
caseClause:
 [defaultClause:]
}
```

ActionScript ステートメントの分岐構造を作成します。if ステートメントと同様に、switch ステートメントは条件をテストし、条件が true を返した場合にステートメントを実行します。すべての switch ステートメントには、default ケースを入れるようにしてください。default ケースには break ステートメントを入れて、後で別の case を追加した場合にフォールスルーエラーが発生しないようにしてください。case がフォールスルーエラーを起こすのは、break ステートメントが含まれていない場合です。

対応バージョン : ActionScript 1.0、Flash Player 4

パラメータ

expression - 任意の式。

例

次の例では、String.fromCharCode(Key.getAscii()) パラメータが A と評価された場合、case "A" の後ろにある trace() ステートメントが実行されます。このパラメータが a と評価された場合、case "a" の後ろにある trace() ステートメントが実行されます。以降、同様に処理されます。いずれの case 式も String.fromCharCode(Key.getAscii()) パラメータと一致しない場合は、default キーワードの後にある trace() ステートメントが実行されます。

```
var listenerObj:Object = new Object();
listenerObj.onKeyDown = function() {
 switch (String.fromCharCode(Key.getAscii())) {
 case "A" :
 trace("you pressed A");
 break;
 case "a" :
 trace("you pressed a");
 break;
```

```
 case "E" :
 case "e" :
 trace("you pressed E or e");
 break;
 case "I" :
 case "i" :
 trace("you pressed I or i");
 break;
 default :
 trace("you pressed some other key");
 break;
 }
};
Key.addListener(listenerObj);
```

関連項目

=== 厳密な等価演算子

throw ステートメント

`throw` *`expression`*

`catch{}` コードブロックによって処理 (キャッチ) できるエラーを生成 (スロー) します。`catch` ブロックが例外をキャッチしない場合は、スローされた値のストリング表現が [出力] パネルに表示されます。

一般には、Error クラスまたはそのサブクラスのインスタンスをスローします。「例」を参照してください。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

`expression:Object` - ActionScript の式またはオブジェクト。

例

この例の `checkEmail()` 関数は、受け取ったストリングが正しいフォーマットの電子メールアドレスかどうかを確認します。ストリングに "@" 記号が含まれていない場合は、エラーをスローします。

```
function checkEmail(email:String) {
 if (email.indexOf("@") == -1) {
 throw new Error("Invalid email address");
 }
}
checkEmail("someuser_theirdomain.com");
```

次に、以下のコードで try コードブロック内の checkEmail() 関数を呼び出します。email_txt ストリングに有効な電子メールアドレスが含まれない場合は、テキストフィールド error_txt にエラーメッセージが表示されます。

```
try {
 checkEmail("Joe Smith");
}
catch (e) {
 error_txt.text = e.toString();
}
```

次の例では、Error クラスのサブクラスをスローします。checkEmail() 関数を修正して、そのサブクラスのインスタンスをスローします。

```
// Define Error subclass InvalidEmailError // In InvalidEmailError.as: class
   InvalidEmailAddress extends Error { var message = "Invalid email address."; }
```

FLA ファイルまたは AS ファイルで、タイムラインのフレーム 1 に次の ActionScript を入力します。

```
import InvalidEmailAddress;
function checkEmail(email:String) {
 if (email.indexOf("@") == -1) {
 throw new InvalidEmailAddress();
 }
}
try {
 checkEmail("Joe Smith");
}
catch (e) {
 this.createTextField("error_txt", this.getNextHighestDepth(), 0, 0, 100, 22);
 error_txt.autoSize = true;
 error_txt.text = e.toString();
}
```

関連項目

Error

try..catch..finally ステートメント

```
try {
// ... try block ...
} finally {
// ... finally block ...
}
```

```
try {
// ... try block ...
} catch(error [:ErrorType1]) {
// ... catch block ...
} [catch(error[:ErrorTypeN]) {
// ... catch block ...
}] [finally {
// ... finally block ...
}]
```

エラーが発生する可能性のあるコードブロックを囲み、そのエラーに対処します。try コードブロック内のコードで throw ステートメントを使用してエラーをスローすると、catch ブロックがある場合は、そのブロックに制御が移ります。さらに finally ブロックがある場合は、そのブロックが実行されます。finally ブロックは、エラーがスローされたかどうかに関係なく、必ず実行されます。try ブロック内のコードがエラーをスローしない場合 (try ブロックの実行が正常に終了した場合) でも、finally ブロックのコードは実行されます。return ステートメントを使用して try ブロックを終了した場合でも、finally ブロックは実行されます。

try ブロックの後には、catch ブロックまたは finally ブロック、またはその両方を続ける必要があります。1 つの try ブロックに対して複数の catch ブロックを指定できますが、finally ブロックは 1 つしか記述できません。try ブロックは、必要なだけ何レベルでもネストできます。

catch ハンドラで指定する *error* パラメータには、e、theException、または x のような単純な識別子を使用します。catch ハンドラの変数は、タイプ付きにすることもできます。複数の catch ブロックを使用する場合は、型付きエラーを使用すれば、1 つの try ブロックからスローされる複数の型のエラーをキャッチできます。

スローした例外がオブジェクトである場合、スローしたオブジェクトが指定したタイプのサブクラスであれば、タイプが一致します。特定のタイプのエラーをスローした場合は、対応するエラーを処理する catch ブロックが実行されます。指定したタイプではない例外がスローされた場合は、catch ブロックは実行されず、try ブロックからそのエラーに一致する外部の catch ハンドラに対して例外が自動的にスローされます。

関数の内部でエラーをスローし、その関数に catch ハンドラが含まれていない場合、その関数は終了します。catch ブロックが見つからない限り、呼び出し元の関数もすべて終了します。このプロセスの間に、すべてのレベルの finally ハンドラが呼び出されます。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

error:Object - throw ステートメントでスローされる式。通常は Error クラスまたはそのサブクラスのインスタンスです。

例

次の例では、try..finally ステートメントの作成方法を示します。finally ブロックのコードは必ず実行されるので、通常、try ブロックの実行後に必要な " 後処理 " を実行するために使用されます。次の例では、setInterval() を使用して 1000 ミリ秒 (1 秒) ごとに関数を呼び出します。エラーが発生すると、エラーがスローされて catch ブロックでキャッチされます。エラーの有無に関係なく、finally ブロックは必ず実行されます。setInterval() が使用されているので、finally ブロックに clearInterval() を配置して、メモリから間隔を消去する必要があります。

```
myFunction = function () {
 trace("this is myFunction");
};
try {
 myInterval = setInterval(this, "myFunction", 1000);
 throw new Error("my error");
}
catch (myError:Error) {
 trace("error caught: "+myError);
}
finally {
 clearInterval(myInterval);
 trace("error is cleared");
}
```

次の例では、エラーの有無にかかわらず、finally ブロックを使って ActionScript オブジェクトを削除します。"Account.as" という名前で新しい AS ファイルを作成します。

```
class Account {
 var balance:Number = 1000;
 function getAccountInfo():Number {
 return (Math.round(Math.random() * 10) % 2);
 }
}
```

"Account.as" と同じディレクトリに新しい AS ドキュメントまたは FLA ドキュメントを作成し、タイムラインのフレーム 1 に次の ActionScript を入力します。

```
import Account;
var account:Account = new Account();
try {
 var returnVal = account.getAccountInfo();
 if (returnVal != 0) {
 throw new Error("Error getting account information.");
 }
}
finally {
 if (account != null) {
 delete account;
 }
}
```

これは try..catch ステートメントの使用例です。try ブロック内のコードが実行されます。try ブロック内のコードで例外がスローされると、制御が catch ブロックに移ります。それにより、Error.toString() メソッドを使ってテキストフィールドにエラーメッセージが表示されます。

"Account.as" と同じディレクトリに新しい FLA ドキュメントを作成し、タイムラインのフレーム 1 に次の ActionScript を入力します。

```
import Account;
var account:Account = new Account();
try {
 var returnVal = account.getAccountInfo();
 if (returnVal != 0) {
 throw new Error("Error getting account information.");
 }
 trace("success");
}
catch (e) {
 this.createTextField("status_txt", this.getNextHighestDepth(), 0, 0, 100, 22);
 status_txt.autoSize = true;
 status_txt.text = e.toString();
}
```

次の例では、1 つの try コードブロックに対して複数の型付き catch コードブロックを使用します。try ブロックでは、発生したエラーのタイプに応じてさまざまなタイプのオブジェクトをスローします。この例の myRecordSet は、RecordSet という (架空の) クラスのインスタンスです。このクラスの sortRows() メソッドは RecordSetException および MalformedRecord の 2 種類のエラーをスローする可能性があります。

次の例において、RecordSetException オブジェクトと MalformedRecord オブジェクトは Error クラスのサブクラスです。それぞれが独自の AS クラスファイルで定義されています。

```
// In RecordSetException.as:
class RecordSetException extends Error {
 var message = "Record set exception occurred.";
}
// In MalformedRecord.as:
class MalformedRecord extends Error {
 var message = "Malformed record exception occurred.";
}
```

RecordSet クラスの sortRows() メソッドでは、発生した例外のタイプに応じて、これらの定義済みオブジェクトのいずれかをスローします。次に例を示します。

```
class RecordSet {
 function sortRows() {
 var returnVal:Number = randomNum();
 if (returnVal == 1) {
 throw new RecordSetException();
 }
 else if (returnVal == 2) {
 throw new MalformedRecord();
```

```
 }
 }
 function randomNum():Number {
 return Math.round(Math.random() * 10) % 3;
 }
}
```

最後に、他の AS ファイルまたは FLA スクリプトでは、次のコードを使って、RecordSet クラスのインスタンスの sortRows() メソッドを呼び出します。このコードでは、sortRows() でスローされるエラーのタイプごとに、catch ブロックを定義します。

```
import RecordSet;
var myRecordSet:RecordSet = new RecordSet();
try {
 myRecordSet.sortRows();
 trace("everything is fine");
}
catch (e:RecordSetException) {
 trace(e.toString());
}
catch (e:MalformedRecord) {
 trace(e.toString());
}
```

関連項目

Error

var ステートメント

var *variableName* [= *value1*][...,*variableNameN*[=*valueN*]]

ローカル変数を宣言する場合に使用します。関数内で変数を宣言した場合、その変数はローカルです。変数はその関数用に定義され、関数呼び出しの終了時にスコープから外れます。具体的には、var を使用して定義された変数は、その変数が含まれるコードブロックでのみ有効です。コードブロックは中カッコ ({}) で囲まれます。

関数外で変数を宣言した場合、その変数は、そのステートメントが含まれるタイムライン全体で使用できます。

別のオブジェクトにスコープがまたがる変数をローカル変数で宣言することはできません。

```
my_array.length = 25; // ok
var my_array.length = 25; // syntax error
```

var を使用すると、変数を厳密に型指定することができます。

カンマで宣言を区切ることにより、1 つのステートメントで複数の変数を宣言できます。ただし、このシンタックスを使用すると、コードの可読性が悪くなる場合があります。

```
var first:String = "Bart", middle:String = "J.", last:String = "Bartleby";
```

メモ：外部スクリプトのクラス定義の中でプロパティを宣言する際にも、var を使用する必要があります。クラスファイルでは、変数のスコープとしてパブリック、プライベート、スタティックがサポートされます。

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

`variableName:String` - 識別子。

例

次の ActionScript は、製品名からなる新しい配列を作成します。Array.push を使用して、配列の末尾にエレメントを追加します。厳密な型指定を使用する場合は、var キーワードを使用する必要があります。product_array の前に var を使用しないで厳密な型指定を使おうとすると、エラーが発生します。

```
var product_array:Array = new Array("MX 2004", "Studio", "Dreamweaver", "Flash",
  "ColdFusion", "Contribute", "Breeze");
product_array.push("Flex");
trace(product_array);
// output: MX 2004,Studio,Dreamweaver,Flash,ColdFusion,Contribute,Breeze,Flex
```

while ステートメント

```
while(condition) {
statement(s);
}
```

条件を評価して、条件の評価結果が true になる場合はステートメントを実行します。その後、ループの先頭に戻り、再び条件を評価します。条件が false に評価された場合、ステートメントはスキップされ、ループが終了します。

while ステートメントは、次の手順を実行します。手順1～4の各繰り返しは、ループの反復と呼ばれます。*condition* は、次の手順に示すように、各反復の始めに再テストされます。

- 式 *condition* が評価されます。
- *condition* の評価が true であるか、ブール値 true に変換される値 (ゼロ以外の数値など) である場合は、手順3に進みます。それ以外の場合は、while ステートメントが完了し、while ループの直後のステートメントから実行が再開されます。
- ステートメントブロック *statement(s)* を実行します。
- 手順1に進みます。

一般にループ処理は、カウンタ変数が指定値より小さい間はアクションを実行するという場合に使用します。各ループの最後で、指定された値に達するまでカウンタがインクリメントされます。指定された値に達すると、*condition* は true でなくなり、ループは終了します。

while ステートメントによって実行するステートメントブロックを囲む中括弧 ({}) は、実行するステートメントが 1 つしかない場合は不要です。

対応バージョン： ActionScript 1.0、Flash Player 4

パラメータ

condition:Boolean - 評価結果が true または false になる式。

例

次の例では、while ステートメントを使用して式をテストします。i の値が 20 未満の場合に、i の値がトレースされます。条件が true でなくなると、ループは終了します。

```
var i:Number = 0;
while (i < 20) {
 trace(i);
 i += 3;
}
```

次の結果が [出力] パネルに表示されます。

```
0
3
6
9
12
15
18
```

関連項目

continue ステートメント

with ステートメント

```
with (object:Object) {
statement(s);
}
```

object パラメータでムービークリップなどのオブジェクトを指定し、そのオブジェクト内の式やアクションを *statement(s)* パラメータで評価できるようにします。これにより、オブジェクトの名前やパスをオブジェクトに繰り返し書き込む必要がなくなります。

object パラメータは、*statement(s)* パラメータのプロパティ、変数、および関数を読み取るときのコンテキストになります。たとえば、*object* が my_array であり、指定されたプロパティのうちの 2 つが length と concat である場合、これらのプロパティは my_array として自動的に読み取られます。length および my_array.concat です。別の例で、*object* が state.california である場合、with ステートメント内のアクションまたはステートメントは california インスタンス内から呼び出されます。

statement(s) パラメータで識別子の値を検索する場合、ActionScript は *object* で指定されたスコープチェーンの先頭から開始し、特定の順序でスコープチェーンの各レベルで識別子を検索します。

識別子を解決するために with ステートメントで使用されるスコープチェーンは、次のようにリストの最初の項目から始まり、最後の項目まで続きます。

- 最も内側の with ステートメントで、*object* パラメータで指定されたオブジェクト。
- 最も外側の with ステートメント内の *object* パラメータで指定されたオブジェクト。
- Activation オブジェクト (ローカル変数を保持する関数が関数内で呼び出されたときに自動的に作成されるテンポラリオブジェクト)。
- 実行中のスクリプトを含むムービークリップ。
- Global オブジェクト (Math や String などの定義済みオブジェクト)。

with ステートメント内に変数を設定するには、with ステートメントの外側で変数を宣言しておくか、変数を設定するタイムラインへのフルパスを入力する必要があります。変数を宣言せずに with ステートメントに設定すると、with ステートメントはスコープチェーンに従って値を検索します。変数がまだ存在しない場合、with ステートメントが呼び出されたタイムライン上に新しい値が設定されます。

with() を使用する代わりに、直接パスを使用することができます。パスが長く、入力が面倒な場合は、ローカル変数を作成してパスを格納しておき、コード内でその変数を再利用することができます。次の ActionScript に例を示します。

```
var shortcut = this._parent._parent.name_txt; shortcut.text = "Hank";
  shortcut.autoSize = true;
```

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

object:Object - ActionScript オブジェクトまたはムービークリップのインスタンス。

例

次の例では、someOther_mc インスタンスの _x プロパティと _y プロパティを設定してから、フレーム 3 に進んで停止するように someOther_mc に指示します。

```
with (someOther_mc) {
 _x = 50;
 _y = 100;
 gotoAndStop(3);
}
```

次のコードの抜粋では、with ステートメントを使用しないで上記コードを記述する方法を示します。

```
someOther_mc._x = 50;
someOther_mc._y = 100;
someOther_mc.gotoAndStop(3);
```

with ステートメントは、スコープチェーンリスト内の複数の項目に同時にアクセスする場合に便利です。次の例では、ビルトインの Math オブジェクトをスコープチェーンの前に設定します。Math をデフォルトオブジェクトとして設定すると、cos、sin、PI の各識別子がそれぞれ Math.cos、Math.sin、Math.PI というように解決されます。a、x、y、r の各識別子は Math オブジェクトのメソッドやプロパティではありませんが、関数 polar() のオブジェクトアクティベーションスコープ内に存在するので、それぞれ対応するローカル変数として処理されます。

```
function polar(r:Number):Void {
 var a:Number, x:Number, y:Number;
 with (Math) {
 a = PI * pow(r, 2);
 x = r * cos(PI);
 y = r * sin(PI / 2);
 }
 trace("area = " + a);
 trace("x = " + x);
 trace("y = " + y);
} polar(3);
```

次の結果が [出力] パネルに表示されます。

```
area = 28.2743338823081
x = -3
y = 3
```

第 2 章

ActionScript クラス

2

ActionScript™ クラスのドキュメントには、シンタックスや使用方法に関する情報が含まれているほか、ActionScript™ の特定のクラスに属するメソッド、プロパティ、イベントハンドラとリスナーのコードサンプルも含まれています (グローバル関数やグローバルプロパティは除く)。こうしたクラスは、アルファベット順に一覧表示され、flash.* パッケージで見られる Flash Player 8 の新しいクラスも含まれています。特定のメソッドまたはプロパティがどのクラスに属しているのかわからない場合、インデックスで調べることができます。

Accessibility

```
Object
  |
  +-Accessibility
```

```
public class Accessibility
extends Object
```

Accessibility クラスは、スクリーンリーダーとの通信を管理します。スクリーンリーダーは、視覚障害のあるユーザー向けに、画面の内容を音声で出力する補助技術です。Accessibility クラスのメソッドは静的です。つまり、クラスのインスタンスを作成しなくても、このクラスのメソッドを使うことができます。

ボタンやムービークリップ、テキストフィールドなど、特定のオブジェクトのアクセシビリティプロパティを取得または設定するには、_accProps プロパティを使用します。Player がアクセシビリティ補助をサポートする環境で実行されているかどうかを確認するには、System.capabilities.hasAccessibility を使用します。

対応バージョン : ActionScript 1.0、Flash Player 6

関連項目

hasAccessibility (capabilities.hasAccessibility プロパティ), _accProps プロパティ

プロパティ一覧

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

メソッド一覧

オプション	署名	説明
static	isActive() : Boolean	アクセシビリティ補助が現在アクティブになっており、Player がアクセシビリティ補助と通信しているかどうかを示します。
static	updateProperties() : Void	_accProps (アクセシビリティプロパティ) オブジェクトに対するすべての変更を有効にします。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

isActive (Accessibility.isActive メソッド)

```
public static isActive() : Boolean
```

アクセシビリティ補助が現在アクティブになっており、Player がアクセシビリティ補助と通信しているかどうかを示します。スクリーンリーダーなどのアクセシビリティ補助がある場合にアプリケーションの動作を変更するには、このメソッドを使用します。

メモ : ドキュメントを再生する Flash ウィンドウが最初に表示された後、このメソッドを 1、2 秒以内に呼び出すと、アクティブ状態のアクセシビリティクライアントが存在しても、false 値が返されることがあります。これは、Flash とアクセシビリティクライアントとの間の通信が非同期であるために発生します。この問題を回避するには、ドキュメントを読み込んだ後、1、2 秒経過してから、このメソッドを呼び出します。

対応バージョン : ActionScript 1.0、Flash Player 6

戻り値

Boolean - ブール値。Flash Player がアクセシビリティ補助 (通常はスクリーンリーダー) と通信している場合は true、それ以外の場合は false を返します。

例

次の例は、アクセシビリティ補助が現在アクティブであるかどうかをチェックします。

```
if (Accessibility.isActive()) {
  trace ("An accessibility aid is currently active");
} else {
  trace ("There is currently no active accessibility aid");
}
```

関連項目

updateProperties (Accessibility.updateProperties メソッド), _accProps プロパティ , hasAccessibility (capabilities.hasAccessibility プロパティ)

updateProperties (Accessibility.updateProperties メソッド)

```
public static updateProperties() : Void
```

_accProps (アクセシビリティプロパティ) オブジェクトに対するすべての変更を有効にします。アクセシビリティプロパティの詳細については、_accProps を参照してください。

複数のオブジェクトのアクセシビリティプロパティを変更する場合、必要な Accessibility.updateProperties() 呼び出しは 1 回だけです。何度も呼び出すと、パフォーマンスが低下するばかりか、スクリーンリーダーの結果がわかりにくくなる可能性があります。

対応バージョン : ActionScript 1.0、Flash Player 6,0,65,0

例

イメージを変更して、アクセシビリティオブジェクトの説明を更新する場合は、次のような ActionScript コードを使用します。

```
my_mc.gotoAndStop(2);

if (my_mc._accProps == undefined ) {
  my_mc._accProps = new Object();
}
my_mc._accProps.name = "Photo of Mount Rushmore";
Accessibility.updateProperties();
```

関連項目

isActive (Accessibility.isActive メソッド), _accProps プロパティ , hasAccessibility (capabilities.hasAccessibility プロパティ)

引数

```
Object
  |
  +-arguments
```

```
public class arguments
extends Object
```

arguments オブジェクトは、関数の引数を保存したり、引数にアクセスしたりするのに使用されます。関数の本体に含まれる場合、ローカルの arguments 変数を使ってアクセスできます。

引数は配列エレメントとして保存され、最初の引数は arguments[0]、2 番目の引数は arguments[1] ... としてアクセスされます。arguments.length プロパティは、関数に渡される引数の数を示します。関数で宣言された数と異なる数の引数が渡される場合があることに注意してください。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 6 以降、arguments オブジェクトは Array クラスのすべてのメソッドとプロパティをサポートしています。

関連項目

Function

プロパティ一覧

オプション	プロパティ	説明
	callee:Object	現在実行中の関数への参照。
	caller:Object	現在実行中の関数を呼び出した関数への参照。別の関数から呼び出された関数でない場合は null を返します。
	length:Number	関数に渡される引数の数。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

メソッド一覧

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

callee (arguments.callee プロパティ)

public callee : Object

現在実行中の関数への参照。

対応バージョン : ActionScript 1.0、Flash Player 5

関連項目

caller (arguments.caller プロパティ)

caller (arguments.caller プロパティ)

public caller : Object

現在実行中の関数を呼び出した関数への参照。別の関数から呼び出された関数でない場合は null を返します。

対応バージョン : ActionScript 1.0、Flash Player 6

関連項目

callee (arguments.callee プロパティ)

length (arguments.length プロパティ)

public length : Number

関数に渡される引数の数。関数で宣言された数よりも増減する場合があります。

対応バージョン : ActionScript 1.0、Flash Player 5

Array

```
Object
  |
  +-Array
```

```
public dynamic class Array
extends Object
```

Array クラスを使用すると、インデックス付き配列にアクセスして操作できます。インデックス付き配列とは、配列内での位置を表す番号でプロパティを特定するオブジェクトのことです。この番号のことをインデックスといいます。すべてのインデックス付き配列はゼロから始まります。つまり、配列内の第 1 のエレメントは [0] であり、第 2 のエレメントは [1] です (以下同様)。Array オブジェクトを作成するには、コンストラクタ new `Array()` を使用します。配列の各エレメントにアクセスするには、配列アクセス演算子 (`[]`) を使用します。

配列エレメントには、数値、ストリング、オブジェクト、他の配列など、各種データ型を保存できます。インデックス付き配列を 1 つ作成し、その各エレメントにそれぞれ異なるインデックス配列を代入することによって、多次元 配列を作成できます。このような配列は、表内のデータを表現する場合に使用できるので多次元であると見なされます。

配列の代入は、値ではなく参照によって行われます。ある配列変数を別の配列変数に代入すると、両方とも同じ配列を参照するようになります。

```
var oneArray:Array = new Array("a", "b", "c");
var twoArray:Array = oneArray; // Both array variables refer to the same array.
twoArray[0] = "z";
trace(oneArray); // Output: z,b,c.
```

Array クラスは、結合配列を作成する目的では使用しないでください。結合配列は異なるデータ構造を持ち、数値エレメントではなく、名前エレメントがあります。結合配列 (ハッシュともいう) を作成するには、Object クラスを使用してください。ActionScript では Array クラスを使用して結合配列を作成できますが、Array クラスのメソッドやプロパティは使用できません。基本的に、結合配列は Object クラスのインスタンスであり、キー値ペアはプロパティとその値という形式で表現されます。Object データタイプを使用して結合配列を宣言する別の理由は、そうすると、オブジェクトリテラルを使用して結合配列にデータを設定できるからです (ただし、宣言時のみ)。次の例では、オブジェクトリテラルを使用して結合配列を作成し、ドット演算子と配列アクセス演算子を両方とも使用して項目にアクセスします。その後、新しいプロパティを作成することで新しいキー値ペアを追加します。

```
var myAssocArray:Object = {fname:"John", lname:"Public"};
trace(myAssocArray.fname); // Output: John
trace(myAssocArray["lname"]); // Output: Public
myAssocArray.initial = "Q";
trace(myAssocArray.initial); // Output: Q
```

対応バージョン： ActionScript 1.0、Flash Player 5 - Flash Player 6 ではネイティブオブジェクトになり、パフォーマンスが大幅に向上しました。

例

次の例では、my_array を 4 つの月の名前で構成します。

```
var my_array:Array = new Array();
my_array[0] = "January";
my_array[1] = "February";
my_array[2] = "March";
my_array[3] = "April";
```

プロパティー覧

オプション	プロパティ	説明
static	CASEINSENSITIVE:Number	この定数は、ソート方法において、大文字と小文字を区別しないソートを指定します。
static	DESCENDING:Number	この定数は、ソート方法において、降順でのソートを指定します。
	length:Number	配列内のエレメント数を示す負でない整数。
static	NUMERIC:Number	この定数は、ソート方法において、(文字ストリングではなく) 数値によるソートを指定します。
static	RETURNINDEXEDARRAY:Number	sort() メソッドまたは sortOn() メソッドの呼び出し結果としてインデックス付き配列を返すソートを指定します。
static	UNIQUESORT:Number	この定数は、ソート方法において、一意性ソート要件を指定します。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター覧

署名	説明
Array([value:Object])	配列を作成できます。

メソッド一覧

オプション	署名	説明
	concat([value: Object]) : Array	パラメータで指定されたエレメントを配列内のエレメントと連結して、新しい配列を作成します。
	join([delimiter: String]) : String	配列内のエレメントをストリングに変換し、指定されたセパレータをエレメント間に挿入し、エレメントを連結して、その結果をストリングとして返します。
	pop() : Object	配列の最後のエレメントを削除して、そのエレメントの値を返します。
	push(value:Object) : Number	エレメントを配列の最後に追加して、配列の新しい長さを返します。
	reverse() : Void	配列の並びを反転させます。
	shift() : Object	配列の最初のエレメントを削除して、そのエレメントを返します。
	slice([startIndex: Number], [endIndex:Number]) : Array	元の配列から一連のエレメントを取り出して、新しい配列を返します。元の配列は変更されません。
	sort ([compareFunction: Object], [options:Number]) : Array	配列内のエレメントをソートします。
	sortOn(fieldName: Object, [options:Object]) : Array	配列内のフィールド (複数のフィールドも可能) に基づいて、配列内のエレメントをソートします。
	splice(startIndex: Number, [deleteCount:Number], [value:Object]) : Array	配列のエレメントを追加および削除します。
	toString() : String	指定された Array オブジェクト内のエレメントを表すストリングを返します。
	unshift(value: Object) : Number	エレメントを配列の先頭に追加して、配列の新しい長さを返します。

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

Array コンストラクタ

```
public Array([value:Object])
```

配列を作成できます。コンストラクタを使用して、さまざまなタイプの配列を作成できます。たとえば、空の配列、長さだけが指定されてエレメント値が定義されない配列、またはエレメントが特定の値を持つ配列などが可能です。

シンタックス 1: パラメータを 1 つも指定しない場合は、長さが 0 の配列が作成されます。

シンタックス 2: 長さだけを指定した場合、length 個のエレメントを持つ配列が作成されます。各エレメントの値は undefined に設定されます。

シンタックス 3: element パラメータを使って値を指定した場合は、特定の値を持つ配列が作成されます。

対応バージョン: ActionScript 1.0、Flash Player 5

パラメータ

value:Object (オプション) - 次のいずれか :

- 配列内のエレメント数を指定する整数
- 任意の 1 つまたは複数の値で構成されるリスト。値として使用できる型は、ブール値、数値、ストリング、オブジェクト、配列です。配列内の第 1 のエレメントのインデックス番号は常に 0 です。

メモ: Array コンストラクタに数値パラメータを 1 つだけ渡した場合、そのパラメータは length と見なされ、Integer() 関数を使って整数に変換されます。

例

シンタックス 1: 次の例では、初期の長さが 0 である新しい Array オブジェクトを作成します。

```
var my_array:Array = new Array();
trace(my_array.length); // Traces 0.
```

シンタックス 2: 次の例では、初期の長さが 4 である新しい Array オブジェクトを作成します。

```
var my_array:Array = new Array(4);
trace(my_array.length); // Returns 4.
trace(my_array[0]); // Returns undefined.
if (my_array[0] == undefined) { // No quotation marks around undefined.
  trace("undefined is a special value, not a string");
} // Traces: undefined is a special value, not a string.
```

シンタックス 3: 次の例では、初期の長さが 5 である新しい Array オブジェクト go_gos_array を作成します。

```
var go_gos_array:Array = new Array("Belinda", "Gina", "Kathy", "Charlotte",
  "Jane");
trace(go_gos_array.length); // Returns 5.
trace(go_gos_array.join(", ")); // Displays elements.
```

go_gos_array 配列の初期エレメントを次のように指定します。

```
go_gos_array[0] = "Belinda";
go_gos_array[1] = "Gina";
go_gos_array[2] = "Kathy";
go_gos_array[3] = "Charlotte";
go_gos_array[4] = "Jane";
```

次のコードでは、go_gos_array 配列に 6 番目のエレメントを追加し、2 番目のエレメントを変更します。

```
go_gos_array[5] = "Donna";
go_gos_array[1] = "Nina"
trace(go_gos_array.join(" + "));
// Returns Belinda + Nina + Kathy + Charlotte + Jane + Donna.
```

関連項目

[] 配列アクセス演算子 , length （Array.length プロパティ ）

CASEINSENSITIVE (Array.CASEINSENSITIVE プロパティ)

public static CASEINSENSITIVE : Number

この定数は、ソート方法において、大文字と小文字を区別しないソートを指定します。この定数は、sort() メソッドまたは sortOn() メソッドの options パラメータに使用できます。

この定数の値は 1 です。

対応バージョン : ActionScript 1.0、Flash Player 7

関連項目

sort (Array.sort メソッド), sortOn (Array.sortOn メソッド)

concat (Array.concat メソッド)

public concat([value:Object]) : Array

パラメータで指定されたエレメントを配列内のエレメントと連結して、新しい配列を作成します。value パラメータで配列が指定されている場合は、配列自体でなく、その配列のエレメントが連結されます。配列 my_array は変更されません。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

value:Object (オプション) - 新しい配列内で連結される数値、エレメント、またはストリング。値を渡さない場合には、my_array の複製が作成されます。

戻り値

Array - この配列のエレメントの後にパラメータのエレメントが続く配列。

例

次のコードは、2 つの配列を連結します。

```
var alpha_array:Array = new Array("a","b","c");
var numeric_array:Array = new Array(1,2,3);
var alphaNumeric_array:Array =alpha_array.concat(numeric_array);
trace(alphaNumeric_array);
// Creates array [a,b,c,1,2,3].
```

次のコードは、3 つの配列を連結します。

```
var num1_array:Array = [1,3,5];
var num2_array:Array = [2,4,6];
var num3_array:Array = [7,8,9];
var nums_array:Array=num1_array.concat(num2_array,num3_array)
trace(nums_array);
// Creates array [1,3,5,2,4,6,7,8,9].
```

ネストされた配列は、通常の配列と同じように処理されます。ネストされた配列内のエレメントは、配列 x_array 内で個別のエレメントとして分解されません。次に例を示します。

```
var a_array:Array = new Array ("a","b","c");

// 2 and 3 are elements in a nested array.
var n_array:Array = new Array(1, [2, 3], 4);

var x_array:Array = a_array.concat(n_array);
trace(x_array[0]); // a
trace(x_array[1]); // b
trace(x_array[2]); // c
trace(x_array[3]); // 1
trace(x_array[4]); // 2, 3
trace(x_array[5]); // 4
```

DESCENDING (Array.DESCENDING プロパティ)

```
public static DESCENDING : Number
```

この定数は、ソート方法において、降順でのソートを指定します。この定数は、sort() メソッドまたは sortOn() メソッドの options パラメータに使用できます。

この定数の値は 2 です。

対応バージョン : ActionScript 1.0、Flash Player 7

関連項目

sort (Array.sort メソッド),sortOn (Array.sortOn メソッド)

join (Array.join メソッド)

```
public join([delimiter:String]) : String
```

配列内のエレメントをストリングに変換し、指定されたセパレータをエレメント間に挿入し、エレメントを連結して、その結果をストリングとして返します。ネストされた配列は、join() メソッドに渡されるセパレータで区切るのではなく、常にカンマ (,) で区切ります。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

delimiter:String (オプション) - 返されるストリング内の配列エレメントを区切る文字またはストリング。このパラメータを省略すると、デフォルトのセパレータとしてカンマ (,) が使用されます。

戻り値

String - ストリング。

例

次の例では、3 つのエレメント Earth、Moon、Sun を含む配列を作成した後、配列を 3 回結合します。1 回目はデフォルトのセパレータ (カンマ (,) とスペース) を使い、2 回目はダッシュ (-) を使い、最後はプラス記号 (+) を使います。

```
var a_array:Array = new Array("Earth","Moon","Sun")
trace(a_array.join());
// Displays Earth,Moon,Sun.
trace(a_array.join(" - "));
// Displays Earth - Moon - Sun.
trace(a_array.join(" + "));
// Displays Earth + Moon + Sun.
```

次の例では、2 つの配列がネストされた配列を作成します。1 番目の配列には、Europa、Io、および Callisto という 3 つのエレメントを格納します。2 番目の配列には、Titan および Rhea という 2 つのエレメントを格納します。配列をプラス記号 (+) で結合しても、ネストされた配列内の各エレメントはカンマ (,) で区切られたままになります。

```
var a_nested_array:Array = new Array(["Europa", "Io", "Callisto"], ["Titan",
  "Rhea"]);
trace(a_nested_array.join(" + "));
// Returns Europa,Io,Callisto + Titan,Rhea.
```

関連項目

split (String.split メソッド)

length (Array.length プロパティ)

```
public length : Number
```

配列内のエレメント数を示す負でない整数。このプロパティは、新しいエレメントが配列に追加されると自動更新されます。配列エレメントに値を代入するとき (`my_array[index] = value` など)、`index` が数値でかつ `index+1` が `length` プロパティよりも大きい場合、`length` プロパティが `index+1` に更新されます。

メモ : `length` プロパティに既存の長さよりも短い値を代入した場合、配列は切り詰められます。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次のコードでは、`length` プロパティがどのように更新されるかを示します。最初の長さが 0 で、1、2、および 10 に更新されます。`length` プロパティに既存の長さよりも短い値を代入した場合、配列は切り詰められます。

```
var my_array:Array = new Array();
trace(my_array.length); // initial length is 0
my_array[0] = "a";
trace(my_array.length); // my_array.length is updated to 1
my_array[1] = "b";
trace(my_array.length); // my_array.length is updated to 2
my_array[9] = "c";
trace(my_array.length); // my_array.length is updated to 10
trace(my_array);
// displays:
// a,b,undefined,undefined,undefined,undefined,undefined,undefined,undefined,c

// if the length property is now set to 5, the array will be truncated
my_array.length = 5;
trace(my_array.length); // my_array.length is updated to 5
trace(my_array); // outputs: a,b,undefined,undefined,undefined
```

NUMERIC (Array.NUMERIC プロパティ)

```
public static NUMERIC : Number
```

この定数は、ソート方法において、(文字ストリングではなく) 数値によるソートを指定します。この定数を options パラメータに設定すると、sort() メソッドと sortOn() メソッドは、数字を文字ストリングとしてではなく数値としてソートします。NUMERIC 定数が設定されていない場合、ソートを実行すると、各配列エレメントは文字ストリングとして処理され、Unicode 順でソートされます。

たとえば、値が [2005, 7, 35] である配列があるとします。options パラメータに NUMERIC 定数が含まれて**いない**場合、ソート後の配列は [2005, 35, 7] となります。NUMERIC 定数が含まれて**いる**場合は、ソート後の配列は [7, 35, 2005] になります。

この定数が配列内の数値に対してのみ適用されることに注意してください。数値データを含むストリング (["23", "5"] など) には適用されません。

この定数の値は 16 です。

対応バージョン : ActionScript 1.0、Flash Player 7

関連項目

sort (Array.sort メソッド), sortOn (Array.sortOn メソッド)

pop (Array.pop メソッド)

```
public pop() : Object
```

配列の最後のエレメントを削除して、そのエレメントの値を返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Object - 指定された配列の最後のエレメントの値。

例

次のコードでは、エレメントが 4 つある配列 myPets_array を作成した後、その最後のエレメントを削除します。

```
var myPets_array:Array = new Array("cat", "dog", "bird", "fish");
var popped:Object = myPets_array.pop();
trace(popped); // Displays fish.
trace(myPets_array); // Displays cat,dog,bird.
```

関連項目

push (Array.push メソッド), shift (Array.shift メソッド), unshift (Array.unshift メソッド)

push (Array.push メソッド)

```
public push(value:Object) : Number
```

エレメントを配列の最後に追加して、配列の新しい長さを返します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`value:Object` - 配列に付加される値。

戻り値

`Number` - 新しい配列の長さを表す整数。

例

次の例では、cat と dog の 2 つのエレメントを持つ配列 myPets_array を作成します。2 行目で配列に 2 つのエレメントを追加します。

push() メソッドから配列の新しい長さが返されるので、最終行の trace() ステートメントは myPets_array の新しい長さ (4) を [出力] パネルに表示します。

```
var myPets_array:Array = new Array("cat", "dog");
var pushed:Number = myPets_array.push("bird", "fish");
trace(pushed); // Displays 4.
```

関連項目

pop (Array.pop メソッド), shift (Array.shift メソッド), unshift (Array.unshift メソッド)

RETURNINDEXEDARRAY (Array.RETURNINDEXEDARRAY プロパティ)

```
public static RETURNINDEXEDARRAY : Number
```

sort() メソッドまたは sortOn() メソッドの呼び出し結果としてインデックス付き配列を返すソートを指定します。この定数は、sort() メソッドまたは sortOn() メソッドの options パラメータで使用できます。この定数を使用すると、ソートの結果を表す配列が返されますが、元の配列は変更されないままになるので、プレビュー機能やコピー機能になります。

この定数の値は 8 です。

対応バージョン : ActionScript 1.0、Flash Player 7

関連項目

sort (Array.sort メソッド), sortOn (Array.sortOn メソッド)

reverse (Array.reverse メソッド)

```
public reverse() : Void
```

配列の並びを反転させます。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、このメソッドを使用して、配列 numbers_array の並びを反転させます。

```
var numbers_array:Array = new Array(1, 2, 3, 4, 5, 6);
trace(numbers_array); // Displays 1,2,3,4,5,6.
numbers_array.reverse();
trace(numbers_array); // Displays 6,5,4,3,2,1.
```

shift (Array.shift メソッド)

```
public shift() : Object
```

配列の最初のエレメントを削除して、そのエレメントを返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Object - 配列の最初のエレメント。

例

次のコードでは、配列 myPets_array を作成し、その配列から最初のエレメントを削除して変数 shifted に代入します。

```
var myPets_array:Array = new Array("cat", "dog", "bird", "fish");
var shifted:Object = myPets_array.shift();
trace(shifted); // Displays "cat".
trace(myPets_array); // Displays dog,bird,fish.
```

関連項目

pop (Array.pop メソッド), push (Array.push メソッド), unshift (Array.unshift メソッド)

slice (Array.slice メソッド)

```
public slice([startIndex:Number], [endIndex:Number]) : Array
```

元の配列から一連のエレメントを取り出して、新しい配列を返します。元の配列は変更されません。返される配列には、startIndex エレメントから endIndex エレメントまでのエレメント (endIndex エレメント自体は除く) がすべて含まれます。

パラメータを何も渡さないと、元の配列の複製が作成されます。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

startIndex:Number (オプション) - スライスの始点のインデックスを指定する数値。*start* が負の数値の場合、始点は配列の末尾から指定されます。つまり、-1 が最後のエレメントです。

endIndex:Number (オプション) - スライスの終点のインデックスを指定する数値。このパラメータを省略すると、スライスには配列の最初から最後までのすべてのエレメントが取り込まれます。*end* が負の数値の場合、終点は配列の末尾から指定されます。つまり、-1 が最後のエレメントです。

戻り値

Array - 元の配列から一連のエレメントを取り出した配列。

例

次の例では、まず 5 種類のペットで構成される配列を作成し、次に slice() を使用して、4 足動物のみで構成される新しい配列を作成します。

```
var myPets_array:Array = new Array("cat", "dog", "fish", "canary", "parrot");
var myFourLeggedPets_array:Array = new Array();
var myFourLeggedPets_array = myPets_array.slice(0, 2);
trace(myFourLeggedPets_array); // Returns cat,dog.
trace(myPets_array); // Returns cat,dog,fish,canary,parrot.
```

次の例では、5 種類のペットで構成される配列を作成し、次に slice() に負の start パラメータを渡して、配列の最後の 2 つのエレメントをコピーします。

```
var myPets_array:Array = new Array("cat", "dog", "fish", "canary", "parrot");
var myFlyingPets_array:Array = myPets_array.slice(-2);
trace(myFlyingPets_array); // Traces canary,parrot.
```

次の例では、5 種類のペットで構成される配列を作成した後、slice() に負の end パラメータを渡して、配列の中間のエレメントをコピーします。

```
var myPets_array:Array = new Array("cat", "dog", "fish", "canary", "parrot");
var myAquaticPets_array:Array = myPets_array.slice(2,-2);
trace(myAquaticPets_array); // Returns fish.
```

sort (Array.sort メソッド)

```
public sort([compareFunction:Object], [options:Number]) : Array
```

配列内のエレメントをソートします。ソートは Unicode 値に基づいて実行されます (ASCII は Unicode のサブセットです)。

`Array.sort()` のデフォルト動作は、次のように動なります。

- ソートでは大文字と小文字が区別されます。*Z* が *a* よりも先になります。
- 昇順にソートされます。*a* が *b* よりも先になります。
- 配列はソート順を反映するように変更されます。同じソートフィールドを持つ複数のエレメントは、ソート済みの配列の中で連続的かつランダムに格納されます。
- 数値フィールドは、ストリングとしてソートされます。たとえば、"1" は "9" よりも小さいストリング値であるため、100 は 99 よりも先になります。

デフォルト設定と異なる設定を使用して配列のソートを行う場合は、`options` パラメータのエントリで説明されているソートオプションのいずれかを使用することも、ソート処理を行う独自のカスタム関数を作成することもできます。カスタム関数を作成する場合は、カスタム関数の名前を最初のパラメータ (`compareFunction`) として使用して `sort()` メソッドを呼び出すことができます。

対応バージョン : ActionScript 1.0、Flash Player 5 - 配列ソートオプションが Flash Player 7 で追加されました。

パラメータ

`compareFunction:Object` (オプション) - 配列内のエレメントのソート順を決定する比較関数。エレメント A と B がある場合、`compareFunction` は次の 3 つの値のいずれかを返します。

- A が B の前に表示されるソート順の場合は -1
- A = B の場合は 0
- A が B の後に表示されるソート順の場合は 1

`options:Number` (オプション) - デフォルトのソート動作を変更する数値または定義済み定数の名前。複数指定する場合は、ビット単位の論理和 (OR) | 演算子で区切ります。`options` パラメータには次の値を指定できます。

- `Array.CASEINSENSITIVE` または 1
- `Array.DESCENDING` または 2
- `Array.UNIQUESORT` または 4
- `Array.RETURNINDEXEDARRAY` または 8
- `Array.NUMERIC` または 16

このパラメータの詳細については、「`Array.sortOn()` メソッド」を参照してください。

メモ : `Array.sort()` は ECMA-262 仕様を拡張した Flash 固有の機能です。ECMA-262 でも定義されていますが、配列ソートオプションは Flash Player 7 で導入されたものです。

戻り値

Array - 戻り値は、次に示すように、指定されたパラメータによって異なります。

- options パラメータに値 4 または Array.UNIQUESORT を指定し、ソート対象の複数のエレメントにまったく同じソートフィールドがある場合は 0 が返されます。配列は変更されません。
- options パラメータに値 8 または Array.RETURNINDEXEDARRAY を指定すると、ソート結果を反映した配列が返されます。配列は変更されません。
- それ以外の場合、値は返されません。ソート順を反映するように配列が変更されます。

例

シンタックス 1: 次に、options に値が渡される場合と渡されない場合の Array.sort() の例を示します。

```
var fruits_array:Array = new Array("oranges", "apples", "strawberries",
  "pineapples", "cherries");
trace(fruits_array); // Displays
  oranges,apples,strawberries,pineapples,cherries.
fruits_array.sort();
trace(fruits_array); // Displays
  apples,cherries,oranges,pineapples,strawberries.
trace(fruits_array); // Writes apples,cherries,oranges,pineapples,strawberries.
fruits_array.sort(Array.DESCENDING);
trace(fruits_array); // Displays
  strawberries,pineapples,oranges,cherries,apples.
trace(fruits_array); // Writes strawberries,pineapples,oranges,cherries,apples.
```

シンタックス 2: 次に、Array.sort() と比較関数を組み合わせた例を示します。エントリは「名前:パスワード」でソートされます。エントリの名前の部分だけをソート条件として使用します。

```
var passwords_array:Array = new Array("mom:glam", "ana:ring", "jay:mag",
  "anne:home", "regina:silly");
function order(a, b):Number {
  var name1:String = a.split(":")[0];
  var name2:String = b.split(":")[0];
  if (name1<name2) {
  return -1;
  } else if (name1>name2) {
  return 1;
  } else {
  return 0;
  }
}
trace("Unsorted:");
//Displays Unsorted:
trace(passwords_array);
//Displays mom:glam,ana:ring,jay:mag,anne:home,regina:silly.
//Writes mom:glam,ana:ring,jay:mag,anne:home,regina:silly
passwords_array.sort(order);
```

```
trace("Sorted:");
//Displays Sorted:
trace(passwords_array);
//Displays ana:ring,anne:home,jay:mag,mom:glam,regina:silly.
//Writes ana:ring,anne:home,jay:mag,mom:glam,regina:silly.
```

関連項目

| ビット単位の論理和（OR）演算子 , sortOn（Array.sortOn メソッド）

sortOn (Array.sortOn メソッド)

```
public sortOn(fieldName:Object, [options:Object]) : Array
```

配列内のフィールド (複数のフィールドも可能) に基づいて、配列内のエレメントをソートします。配列は、次に示す特性を備えている必要があります。

- インデックス付き配列を対象とします。結合配列は対象外です。
- 配列の各エレメントは、プロパティがあるオブジェクトを保持するものとします。
- すべてのオブジェクトには共通のプロパティが少なくとも 1 つあるものとします。このようなプロパティをフィールドといいます。

複数の fieldName パラメータを渡す場合、最初のフィールドが第 1 ソートフィールド、2 番目のフィールドが第 2 ソートフィールドを表します (3 番目以降も同様)。ソートは Unicode 値に基づいて実行されます (ASCII は Unicode のサブセットです)。fieldName パラメータで指定されたフィールドが、比較対象のいずれのエレメントにも含まれていない場合、そのフィールドは undefined と見なされます。ソート済みの配列では、エレメントが連続的かつランダムに格納されます。

デフォルトでは、Array.sortOn() は次のように動作します。

- ソートでは大文字と小文字が区別されます。*Z* が *a* よりも先になります。
- 昇順にソートされます。*a* が *b* よりも先になります。
- 配列はソート順を反映するように変更されます。同じソートフィールドを持つ複数のエレメントは、ソート済みの配列の中で連続的かつランダムに格納されます。
- 数値フィールドは、ストリングとしてソートされます。たとえば、"1" は "9" よりも小さいストリング値であるため、100 は 99 よりも先になります。

Flash Player 7 では options パラメータが追加されました。このパラメータを使用すると、デフォルトのソート動作をオーバーライドすることができます。単純な配列 (たとえば 1 つのフィールドだけを持つ配列) をソートする場合や、options パラメータによってサポートされないソート順序を指定する場合には、Array.sort() を使用します。

複数のフラグを渡すには、ビット単位の論理和 OR (|) 演算子で区切ります。

```
my_array.sortOn(someFieldName, Array.DESCENDING | Array.NUMERIC);
```

Flash Player 8 では、複数のフィールドでソートを行う場合に、各フィールドに対して異なるソートオプションを指定する機能が追加されました。Flash Player 8 では、options パラメータはさまざまなソートオプションを受け入れます。各ソートオプションは fieldName パラメータのソートフィールドに対応します。次の例では、第 1 ソートフィールド a を降順で、第 2 ソートフィールド b を数値ソートで、第 3 ソートフィールド c を大文字と小文字を区別しないでソートします。

```
Array.sortOn (["a", "b", "c"], [Array.DESCENDING, Array.NUMERIC,
  Array.CASEINSENSITIVE]);
```

メモ : fieldName 配列と options 配列のエレメントは同数になっている必要があります。同数でない場合、options 配列は無視されます。また、Array.UNIQUESORT オプションと Array.RETURNINDEXEDARRAY オプションは、配列内の最初のエレメントとしてのみ使用できます。そうしない場合、これらのオプションは無視されます。

対応バージョン : ActionScript 1.0、Flash Player 6 - options パラメータが Flash Player 7 で追加されました。複数のソートフィールドで各種 options パラメータを使用する機能が Flash Player 8 で追加されました。

パラメータ

fieldName: Object - ソート値として使用するフィールドを指定するストリング、または最初のエレメントが第 1 ソートフィールド、2 番目が第 2 ソートフィールド (3 番目以降も同様) を表す配列。

options: Object (オプション) - ソートビヘイビアを変更する数値または定義済み定数の名前。ビット単位の論理和 OR (|) 演算子によって区切ります。options パラメータには次の値を指定できます。

- Array.CASEINSENSITIVE または 1
- Array.DESCENDING または 2
- Array.UNIQUESORT または 4
- Array.RETURNINDEXEDARRAY または 8
- Array.NUMERIC または 16

数値形式 (2) ではなく、ストリング形式のフラグ (DESCENDING など) を使用すると、コードヒントが有効になります。

戻り値

Array - 戻り値は、パラメータを渡したかどうかにより異なります。

- options パラメータの値として 4 (Array.UNIQUESORT) を指定し、ソート対象の複数のエレメントにまったく同じソートフィールドがある場合は値 0 が返されます。配列は変更されません。
- options パラメータの値として 8 (Array.RETURNINDEXEDARRAY) を指定した場合、そのソート結果を反映する配列が返されます。配列は変更されません。
- これ以外の場合、何も返されず、ソート順を反映するよう配列が変更されます。

例

次の例では、新しい配列を作成し、その配列を name フィールドと city フィールドに基づいてソートします。最初のソートでは、第1ソート値に name を、第2ソート値に city を使用します。2番目のソートでは、第1ソート値に city を、第2ソート値に name を使用します。

```
var rec_array:Array = new Array();
rec_array.push({name: "john", city: "omaha", zip: 68144});
rec_array.push({name: "john", city: "kansas city", zip: 72345});
rec_array.push({name: "bob", city: "omaha", zip: 94010});
for(i=0; i<rec_array.length; i++){
  trace(rec_array[i].name + ", " + rec_array[i].city);
}
// Results:
// john, omaha
// john, kansas city
// bob, omaha

rec_array.sortOn(["name", "city"]);
for(i=0; i<rec_array.length; i++){
  trace(rec_array[i].name + ", " + rec_array[i].city);
}
// Results:
// bob, omaha
// john, kansas city
// john, omaha

rec_array.sortOn(["city", "name" ]);
for(i=0; i<rec_array.length; i++){
  trace(rec_array[i].name + ", " + rec_array[i].city);
}
// Results:
// john, kansas city
// bob, omaha
// john, omaha
```

次の例では、オブジェクト配列を使用して、options パラメータの使用法を示します。

```
var my_array:Array = new Array();
my_array.push({password: "Bob", age:29});
my_array.push({password: "abcd", age:3});
my_array.push({password: "barb", age:35});
my_array.push({password: "catchy", age:4});
```

password フィールドに対してデフォルトのソートを実行すると、次の結果が生成されます。

```
my_array.sortOn("password");
// Bob
// abcd
// barb
// catchy
```

password フィールドに対して大文字と小文字を区別しないソートを実行すると、次の結果が生成されます。

```
my_array.sortOn("password", Array.CASEINSENSITIVE);
// abcd
// barb
// Bob
// catchy
```

password フィールドに対して、大文字と小文字を区別しない降順ソートを実行すると、次の結果が生成されます。

```
my_array.sortOn("password", Array.CASEINSENSITIVE | Array.DESCENDING);
// catchy
// Bob
// barb
// abcd
```

age フィールドに対してデフォルトのソートを実行すると、次の結果が生成されます。

```
my_array.sortOn("age");
// 29
// 3
// 35
// 4
```

age フィールドに対して数値ソートを実行すると、次の結果が生成されます。

```
my_array.sortOn("age", Array.NUMERIC);
// my_array[0].age = 3
// my_array[1].age = 4
// my_array[2].age = 29
// my_array[3].age = 35
```

age フィールドに対して数値の降順ソートを実行すると、次の結果が生成されます。

```
my_array.sortOn("age", Array.DESCENDING | Array.NUMERIC);
// my_array[0].age = 35
// my_array[1].age = 29
// my_array[2].age = 4
// my_array[3].age = 3
```

Array.RETURNEDINDEXARRAY ソートオプションを使用する場合は、別の配列に戻り値を代入する必要があります。元の配列は変更されません。

```
var indexArray:Array = my_array.sortOn("age", Array.RETURNINDEXEDARRAY);
```

関連項目

| ビット単位の論理和（OR）演算子, sort（Array.sort メソッド）

splice (Array.splice メソッド)

```
public splice(startIndex:Number, [deleteCount:Number], [value:Object]) : Array
```

配列のエレメントを追加および削除します。このメソッドは、コピーを作成しないで、配列を変更します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`startIndex:Number` - 挿入または削除を開始する配列エレメントのインデックスを示す整数。負の整数を指定すると、配列の末尾を基準として位置を指定できます (たとえば、-1 は配列の最後のエレメントです)。

`deleteCount:Number` (オプション) - 削除するエレメント数を示す整数。この数には、`startIndex` パラメータで指定するエレメントが含まれます。`deleteCount` パラメータに値を指定しないと、`startIndex` パラメータで指定した配列エレメントから最後の配列エレメントまでの値がすべて削除されます。値として 0 を指定すると、エレメントは削除されません。

`value:Object` (オプション) - `startIndex` パラメータで指定した挿入箇所に挿入する値を指定します。

戻り値

`Array` - 元の配列から削除したエレメントが含まれる配列。

例

次の例では、配列を 1 つ作成し、splice メソッドの `startIndex` パラメータにエレメントインデックス 1 を使用してスプライスします。これにより、2 番目以降のすべてのエレメントが配列から削除され、元の配列にはインデックス 0 だけが残ります。

```
var myPets_array:Array = new Array("cat", "dog", "bird", "fish");
trace( myPets_array.splice(1) ); // Displays dog,bird,fish.
trace( myPets_array ); // cat
```

次の例では、配列を 1 つ作成し、splice メソッドの `startIndex` パラメータにエレメントインデックス 1 を、`deleteCount` パラメータに数値 2 を使用してスプライスします。これにより、2 番目以降の 2 つのエレメントが配列から削除され、元の配列には最初と最後のエレメントだけが残ります。

```
var myFlowers_array:Array = new Array("roses", "tulips", "lilies", "orchids");
trace( myFlowers_array.splice(1,2 ) ); // Displays tulips,lilies.
trace( myFlowers_array ); // roses,orchids
```

次の例では、配列を 1 つ作成し、splice メソッドの startIndex パラメータにエレメントインデックス 1 を、deleteCount パラメータに数値 0 を、value パラメータにストリング chair 使用してスプライスします。この場合、元の配列からは何も削除されず、ストリング chair がインデックス 1 に追加されます。

```
var myFurniture_array:Array = new Array("couch", "bed", "desk", "lamp");
trace( myFurniture_array.splice(1,0, "chair" ) ); // Displays empty array.
trace( myFurniture_array ); // displays couch,chair,bed,desk,lamp
```

toString (Array.toString メソッド)

```
public toString() : String
```

指定された Array オブジェクト内のエレメントを表すストリングを返します。インデックス 0 から最大インデックスまでの配列内のすべてのエレメントを、カンマで区切られた連結ストリングに変換して返します。カスタムセパレータを指定するには、Array.join() メソッドを使用します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

String - ストリング。

例

次の例では、my_array を作成し、それをストリングに変換します。

```
var my_array:Array = new Array();
my_array[0] = 1;
my_array[1] = 2;
my_array[2] = 3;
my_array[3] = 4;
my_array[4] = 5;
trace(my_array.toString()); // Displays 1,2,3,4,5.
```

上記の例では、trace ステートメントの結果として、1,2,3,4,5 が出力されます。

関連項目

split (String.split メソッド),join (Array.join メソッド)

UNIQUESORT (Array.UNIQUESORT プロパティ)

public static UNIQUESORT : Number

この定数は、ソートメソッドにおいて、一意性ソート要件を指定します。この定数は、sort( ) メソッドまたは sortOn( ) メソッドの options パラメータで使用できます。一意性ソートオプションを指定すると、ソート対象に含まれる任意の 2 つのエレメントまたはフィールドの値が同じである場合にソートが中止されます。

この定数の値は 4 です。

対応バージョン : ActionScript 1.0、Flash Player 7

関連項目

sort (Array.sort メソッド), sortOn (Array.sortOn メソッド)

unshift (Array.unshift メソッド)

public unshift(value:Object) : Number

エレメントを配列の先頭に追加して、配列の新しい長さを返します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

value:Object - 配列の先頭に挿入される数値、エレメント、または変数。

戻り値

Number - 新しい配列の長さを表す整数。

例

次の例では、Array.unshift( ) メソッドの使用法を示します。

```
var pets_array:Array = new Array("dog", "cat", "fish");
trace( pets_array ); // Displays dog,cat,fish.
pets_array.unshift("ferrets", "gophers", "engineers");
trace( pets_array ); // Displays ferrets,gophers,engineers,dog,cat,fish.
```

関連項目

pop (Array.pop メソッド), push (Array.push メソッド), shift (Array.shift メソッド)

AsBroadcaster

```
Object
  |
  +-AsBroadcaster
```

```
public class AsBroadcaster
extends Object
```

このクラスには、ユーザー定義オブジェクトに追加可能なイベント通知機能とリスナー管理機能があります。このクラスは、独自のイベント処理メカニズムを作成する上級ユーザー向けです。このクラスを使用すると、任意のオブジェクトをイベントブロードキャスターにすることができます。また、ブロードキャストオブジェクトが broadcastMessage() メソッドを呼び出したときにいつでも通知を受信するリスナーオブジェクトを作成できます。

AsBroadcaster クラスにはコンストラクタ関数がありません。このクラスを使用するには、次に示すプロセスに従います。

- イベントブロードキャスターとして機能するオブジェクトを選択または作成します。
- 静的な AsBroadcaster.initialize(obj:Object) メソッドを呼び出すことでオブジェクトをイベントブロードキャスターにします。ここで、obj パラメータはブロードキャスターにするオブジェクトの名前です。
- リスナーオブジェクトを選択または作成します。リスナーオブジェクトは、ブロードキャストオブジェクトがメッセージをブロードキャストするといつでも通知を受信します。
- リスナーオブジェクトごとにリスナーメソッドを定義します。リスナーメソッドは、イベント通知に応答して ActionScript を実行します。メソッドの名前は、ブロードキャストオブジェクトからブロードキャストされるイベントの名前と一致させる必要があります。
- myBroadcaster.addListener(myListener) を呼び出すことで各リスナーオブジェクトをイベントブロードキャスターに登録します。ここで、myBroadcaster はイベントブロードキャスターオブジェクトの名前であり、myListener はリスナーオブジェクトの名前です。各イベントブロードキャスターは、メッセージをブロードキャストするときの通知先リスナーオブジェクトのリストを格納します。リスナーをリストに追加するには addListener() メソッドを使用し、リストからリスナーを削除するには removeListener() メソッドを使用します。
- 最後に、メッセージをブロードキャストするために、myBroadcaster.broadcastMessage(eventName:String) メソッドを呼び出します。ここで、myBroadcaster はイベントブロードキャスターの名前であり、eventName は、リスナーメソッドの名前に一致するイベントの名前です。

メモ : *AsBroadcaster* の 2 番目の文字を大文字にするというミスがよく見られます。AsBroadcaster.initialize() メソッドを呼び出すときには、2 番目の文字を必ず小文字にしてください。*AsBroadcaster* のスペルを間違えると、警告なしに失敗します。

対応バージョン : ActionScript 1.0、Flash Player 6

プロパティー覧

オプション	プロパティ	説明
	_listeners:Array (読み取り専用)	すべての登録済みリスナーオブジェクトへの参照リストです。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

メソッド一覧

オプション	署名	説明
	addListener(listenerObj:Object) : Boolean	イベント通知メッセージを受信するオブジェクトを登録します。
	broadcastMessage (eventName:String) : Void	リスナーリストに含まれる各オブジェクトにイベントメッセージを送信します。
static	initialize(obj:Object) : Void	指定されたオブジェクトにイベント通知機能とリスナー管理機能を追加します。
	removeListener (listenerObj:Object) : Boolean	イベント通知メッセージを受信するオブジェクトのリストからオブジェクトを削除します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

addListener (AsBroadcaster.addListener メソッド)

```
public addListener(listenerObj:Object) : Boolean
```

イベント通知メッセージを受信するオブジェクトを登録します。このメソッドは、ブロードキャストオブジェクトで呼び出され、引数としてリスナーオブジェクトが渡されます。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

`listenerObj:Object` - イベント通知を受信するリスナーオブジェクトの名前。

戻り値

`Boolean` - このメソッドの戻り値の型は技術的には Boolean ですが、必ず値 `true` を返すので、実質的には `Void` です。

例

次の例は、`AsBroadcaster.initialize()` メソッドのエントリに記載されている完全な例から抜粋したものです。

```
someObject.addListener(myListener1); // Register myListener1 as listener.
someObject.addListener(myListener2); // Register myListener2 as listener.
```

関連項目

`initialize (AsBroadcaster.initialize メソッド )`,
`removeListener (AsBroadcaster.removeListener メソッド )`

broadcastMessage (AsBroadcaster.broadcastMessage メソッド)

```
public broadcastMessage(eventName:String) : Void
```

リスナーリストに含まれる各オブジェクトにイベントメッセージを送信します。リスナーオブジェクトがメッセージを受信するとき、Flash Player はリスナーオブジェクトにある同じ名前の関数を呼び出します。オブジェクトが次のようにイベントメッセージをブロードキャストするとします。

```
obj.broadcastMessage("onAlert");
```

このメッセージの受信時に、Flash Player は受信するリスナーオブジェクトの `onAlert()` というメソッドを呼び出します。

メモ : `broadcastMessage()` メソッドに追加の引数を指定することにより、リスナー関数に引数を渡すことができます。`eventName` パラメータの後に追加した引数はすべて、リスナーメソッドが引数として受け取ります。

このメソッドは、AsBroadcaster.initialize() メソッドを使用して初期化されたオブジェクトだけから呼び出すことができます。

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

eventName:String - ブロードキャストするイベントの名前。リスナーメソッドの名前は、ブロードキャストイベントを受信するためにこのパラメータと一致させる必要があります。eventName の後に追加引数を入れることで、リスナーメソッドに引数を渡すことができます。

例

次の例は、AsBroadcaster.initialize() メソッドのエントリで最初に記載されている完全な例から抜粋したものです。

```
someObject.broadcastMessage("someEvent"); // Broadcast the "someEvent" message.
```

次の例は、AsBroadcaster.initialize() メソッドのエントリで 2 番目に記載されている完全な例から抜粋したものです。引数をリスナーメソッドに送る方法を示しています。

```
someObject.broadcastMessage("someEvent", 3, "arbitrary string");
```

関連項目

initialize (AsBroadcaster.initialize メソッド),
removeListener (AsBroadcaster.removeListener メソッド)

initialize (AsBroadcaster.initialize メソッド)

```
public static initialize(obj:Object) : Void
```

指定されたオブジェクトにイベント通知機能とリスナー管理機能を追加します。これは静的メソッドなので、AsBroadcaster クラスを使って呼び出す必要があります。ここで、someObject はイベントブロードキャスターとして初期化するオブジェクトの名前です。

```
AsBroadcaster.initialize(someObject);
```

メモ： *AsBroadcaster* の 2 番目の文字を大文字にするというミスがよく見られます。AsBroadcaster.initialize() メソッドを呼び出すときには、2 番目の文字を必ず小文字にしてください。*AsBroadcaster* のスペルを間違えると、警告なしに失敗します。

このメソッドは、obj パラメータで指定されるオブジェクトに、_listeners プロパティおよび次の 3 つのメソッドを追加します。

- obj.addListener()
- obj.removeListener()
- obj.broadcastMessage()

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

obj:Object - ブロードキャストオブジェクトとして機能させるオブジェクト

例

次の例では、汎用オブジェクト someObject を作成して、それをイベントブロードキャスターにします。出力は、2 つの trace() ステートメントで示されるストリングになります。

```
var someObject:Object = new Object(); // Creates broadcast object.

var myListener1:Object = new Object(); // Creates listener object.
var myListener2:Object = new Object(); // Creates listener object.

myListener1.someEvent = function() { // Creates listener method.
  trace("myListener1 received someEvent");
}
myListener2.someEvent = function() { // Creates listener method.
  trace("myListener2 received someEvent");
}

AsBroadcaster.initialize(someObject); // Makes someObject an event broadcaster.
someObject.addListener(myListener1); // Registers myListener1 as listener.
someObject.addListener(myListener2); // Registers myListener2 as listener.
someObject.broadcastMessage("someEvent"); // Broadcasts the "someEvent" message.
```

次の例では、broadcastMessage() メソッドを使用して、リスナーメソッドに引数を渡す方法を示します。出力は、3 つの trace() ステートメントに示された 3 つのストリングになります。それには、broadcastMessage() メソッドを通じて渡される引数も含まれます。

```
var someObject:Object = new Object();

var myListener:Object = new Object();
myListener.someEvent = function(param1:Number, param2:String) {
  trace("myListener received someEvent");
  trace("param1: " + param1);
  trace("param2: " + param2);
}

AsBroadcaster.initialize(someObject);
someObject.addListener(myListener);
someObject.broadcastMessage("someEvent", 3, "arbitrary string");
```

_listeners (AsBroadcaster._listeners プロパティ)

public _listeners : Array [読み取り専用]

すべての登録済みリスナーオブジェクトへの参照リストです。このプロパティは内部処理のために設けたものであり、直接操作することを目的としたものではありません。addListener() メソッドと removelistener() メソッドを呼び出すことで、この配列にオブジェクトを追加したり、この配列からオブジェクトを削除したりします。

このプロパティは、AsBroadcaster.initialize() メソッドを使用して初期化されたオブジェクトだけから呼び出すことができます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、length プロパティを使用して、イベントブロードキャスターに現時点で登録されているリスナーオブジェクト数を確認する方法を示します。次のコードは、AsBroadcaster.initialize() エントリの「例」セクションで最初に記載されている完全な例に追加すると機能します。

```
trace(someObject._listeners.length); // Output: 2
```

上級ユーザー向けとして、次の例では、_listeners プロパティを使用して、イベントブロードキャスターに登録されているすべてのリスナーと、各リスナーオブジェクトのすべてのプロパティを列挙する方法を示します。次の例では、最初のリスナーオブジェクトに 2 つの異なるリスナーメソッドを作成します。

```
var someObject:Object = new Object(); // create broadcast object

var myListener1:Object = new Object(); // create listener object
var myListener2:Object = new Object(); // create listener object

myListener1.someEvent = function() { // create listener method
  trace("myListener1 received someEvent");
}
myListener1.anotherEvent = function() { // create another listener method
  trace("myListener1 received anotherEvent");
}
myListener2.someEvent = function() { // create listener method
  trace("myListener2 received someEvent");
}

AsBroadcaster.initialize(someObject); // make someObject an event broadcaster
someObject.addListener(myListener1); // register myListener1 as listener
someObject.addListener(myListener2); // register myListener2 as listener

var numListeners:Number = someObject._listeners.length; // get number of
  registered listeners
```

```
// cycle through all listener objects, listing all properties of each listener
  object
for (var i:Number = 0; i < numListeners; i++) {
  trace("Listener " + i + " listens for these events:");
  for (item in someObject._listeners[i]) {
    trace (" " + item + ": " + someObject._listeners[i][item]);
  }
}
```

関連項目

initialize (AsBroadcaster.initialize メソッド)

removeListener (AsBroadcaster.removeListener メソッド)

public removeListener(listenerObj:Object) : Boolean

イベント通知メッセージを受信するオブジェクトのリストからオブジェクトを削除します。

このメソッドは、AsBroadcaster.initialize() メソッドを使用して初期化されたオブジェクトだけから呼び出すことができます。

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

listenerObj:Object - ブロードキャストオブジェクトからイベント通知を受信するように登録されているリスナーオブジェクトの名前。

戻り値

Boolean - リスナーオブジェクトが削除されると true を返します。それ以外の場合は false を返します。

例

次の例では、登録済みリスナーのリストからリスナーを削除する方法を示します。次のコードは、AsBroadcaster.initialize() エントリの「例」セクションで最初に記載されている完全な例に追加すると機能します。trace() ステートメントの目的は、あくまで、removeListener() メソッドの呼び出し後に登録済みリスナー数が 1 つ減少していることを確認することです。

```
trace(someObject._listeners.length); // Output: 2
someObject.removeListener(myListener1);
trace(someObject._listeners.length); // Output: 1
```

関連項目

addListener (AsBroadcaster.addListener メソッド),
initialize (AsBroadcaster.initialize メソッド)

BevelFilter (flash.filters.BevelFilter)

```
Object
  |
  +-BitmapFilter
     |
     +-flash.filters.BevelFilter
```

```
public class BevelFilter
extends BitmapFilter
```

BevelFilter クラスを使用すると、Flash の各種オブジェクトにベベル効果を追加できます。ボタンなどのオブジェクトにベベル効果を適用すると 3 次元的に表現されます。異なるハイライトカラー、シャドウカラー、ベベルのぼかし量、ベベルの角度、ベベルの配置、ノックアウト効果を使って、ベベルの外観をカスタマイズできます。

フィルタの使い方は、フィルタの適用先オブジェクトによって異なります。

- 実行時にムービークリップ、テキストフィールド、ボタンにフィルタを適用する場合は、filters プロパティを使用します。オブジェクトの filters プロパティを設定しても、そのオブジェクトは変更されません。filters プロパティをクリアすれば、設定を取り消すことができます。
- BitmapData インスタンスにフィルタを適用するには、BitmapData.applyFilter() メソッドを使用します。BitmapData オブジェクトで applyFilter を呼び出すことによって、ソース BitmapData オブジェクトとフィルタオブジェクトが取得され、フィルタを適用して得られるイメージが生成されます。

イメージやビデオにもオーサリング時にフィルタ効果を適用できます。詳細については、オーサリングのマニュアルを参照してください。

ムービークリップやボタンにフィルタを適用する場合は、ムービークリップやボタンの cacheAsBitmap プロパティを true に設定します。すべてのフィルタをクリアすると、cacheAsBitmap は元の値に戻ります。

このフィルタはステージの拡大・縮小に対応していますが、通常の拡大・縮小、回転、傾斜には対応していません。オブジェクト自体が拡大・縮小される場合 (_xscale と _yscale が 100% ではない場合)、フィルタは拡大・縮小されません。拡大・縮小されるのは、ステージをズームインした場合のみです。

結果として得られるイメージの幅または高さが 2880 ピクセルを超える場合、フィルタは適用されません。たとえば、フィルタが適用されたサイズの大きいムービークリップをズームインするとき、結果として得られるイメージが 2880 ピクセルの制限を超える場合は、フィルタがオフになります。

対応バージョン： ActionScript 1.0、Flash Player 8

関連項目

filters (MovieClip.filters プロパティ), cacheAsBitmap (MovieClip.cacheAsBitmap プロパティ), filters (Button.filters プロパティ), cacheAsBitmap (Button.cacheAsBitmap プロパティ), filters (TextField.filters プロパティ), applyFilter (BitmapData.applyFilter メソッド), MovieClip

プロパティ一覧

オプション	プロパティ	説明
	angle:Number	ベベルの角度です。
	blurX:Number	水平方向のぼかし量 (ピクセル単位) です。
	blurY:Number	垂直方向のぼかし量 (ピクセル単位) です。
	distance:Number	ベベルのオフセット距離です。
	highlightAlpha: Number	ハイライトカラーのアルファ透明度の値です。
	highlightColor: Number	ベベルのハイライトカラーです。
	knockout:Boolean	true の場合は、ノックアウト効果を適用します。その結果、オブジェクトは完全に透明になり、ドキュメントの背景色で表示されます。
	quality:Number	フィルタを適用する回数です。
	shadowAlpha:Number	シャドウカラーのアルファ透明度の値です。
	shadowColor:Number	ベベルのシャドウカラーです。
	strength:Number	インプリントやスプレッドの長さです。
	type:String	ベベルの種類です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター一覧

署名	説明
BevelFilter ([distance:Number], [angle:Number], [highlightColor: Number], [highlightAlpha: Number], [shadowColor: Number], [shadowAlpha: Number], [blurX:Number], [blurY:Number], [strength:Number], [quality:Number], [type:String], [knockout:Boolean])	指定されたパラメータで新しい BevelFilter インスタンスを初期化します。

メソッド一覧

オプション	署名	説明
	clone() : BevelFilter	このフィルタオブジェクトのコピーを返します。

BitmapFilter クラスから継承されるメソッド

clone (BitmapFilter.clone メソッド)

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

angle (BevelFilter.angle プロパティ)

```
public angle : Number
```

ベベルの角度です。指定できる値は 0 ～ 360 度です。デフォルト値は 45 です。

角度の値は、オブジェクトに対する架空の光源の角度を表し、オブジェクトに対する効果の相対位置を決定します。距離が 0 に設定された場合、効果がオブジェクトからオフセットされないため、角度プロパティは適用されません。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip インスタンス (rect) がクリックされたときに、その angle プロパティを変更します。

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelDistance");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.angle = 225;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, 0.8, 0x0000FF, 0.8,
  20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

BevelFilter コンストラクタ

```
public BevelFilter([distance:Number], [angle:Number], [highlightColor:Number],
  [highlightAlpha:Number], [shadowColor:Number], [shadowAlpha:Number],
  [blurX:Number], [blurY:Number], [strength:Number], [quality:Number],
  [type:String], [knockout:Boolean])
```

指定されたパラメータで新しい BevelFilter インスタンスを初期化します。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

`distance:Number` (オプション) - ベベルのオフセット距離 (ピクセル単位) です (浮動小数値)。デフォルト値は 4 です。

`angle:Number` (オプション) - ベベルの角度 (0 ～ 360 度) です。デフォルト値は 45 です。

`highlightColor:Number` (オプション) - ベベルのハイライトカラー (*0xRRGGBB*)。デフォルト値は 0xFFFFFF. です。

`highlightAlpha:Number` (オプション) - ハイライトカラーのアルファ透明度の値。0 ～ 1 の範囲の値を指定できます。たとえば .25 と指定すると、透明度は 25% になります。デフォルト値は 1 です。

`shadowColor:Number` (オプション) - ベベルのシャドウカラー (*0xRRGGBB*)。デフォルト値は 0x000000 です。

`shadowAlpha:Number` (オプション) - シャドウカラーのアルファ透明度の値。0 ～ 1 の範囲の値を指定できます。たとえば 0.25 と指定すると、透明度は 25% になります。デフォルト値は 1 です。

`blurX:Number` (オプション) - 水平方向のぼかし量 (ピクセル単位) です。指定できる値は 0 ～ 255 (浮動小数) です。デフォルト値は 4 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

`blurY:Number` (オプション) - 垂直方向のぼかし量です。指定できる値は 0 ～ 255 (浮動小数) です。デフォルト値は 4 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

`strength:Number` (オプション) - インプリントやスプレッドの長さです。値が大きいほど、濃い色がインプリントされるので、ベベルと背景との間のコントラストが強くなります。有効な値は 0 ～ 255 で、デフォルト値は 1 です。

`quality:Number` (オプション) - フィルタを適用する回数。デフォルト値は 1 (低品質) です。値 2 は標準の品質であり、値 3 は高品質です。

`type:String` (オプション) - ベベルの種類。有効な値は `"inner"`、`"outer"`、および `"full"` です。デフォルト値は `"inner"` です。

`knockout:Boolean` (オプション) - `true` の場合は、ノックアウト効果を適用します。その結果、オブジェクトは完全に透明になり、ドキュメントの背景色で表示されます。デフォルト値は `false` (ノックアウトなし) です。

例

次の例では、新しい BevelFilter をインスタンス化して、MovieClip インスタンス (rect) に適用します。

```
import flash.filters.BevelFilter;

var distance:Number = 5;
var angleInDegrees:Number = 45;
var highlightColor:Number = 0xFFFF00;
var highlightAlpha:Number = 0.8;
var shadowColor:Number = 0x0000FF;
var shadowAlpha:Number = 0.8;
var blurX:Number = 5;
var blurY:Number = 5;
var strength:Number = 5;
var quality:Number = 3;
var type:String = "inner";
var knockout:Boolean = false;

var filter:BevelFilter = new BevelFilter(distance,
                           angleInDegrees,
                           highlightColor,
                           highlightAlpha,
                           shadowColor,
                           shadowAlpha,
                           blurX,
                           blurY,
                           strength,
                           quality,
                           type,
                           knockout);

var rect:MovieClip = createRectangle(100, 100, 0x00CC00, "bevelFilterExample");
rect.filters = new Array(filter);

function createRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;
  return rect;
}
```

blurX (BevelFilter.blurX プロパティ)

```
public blurX : Number
```

水平方向のぼかし量 (ピクセル単位) です。指定できる値は 0 ～ 255 (浮動小数) です。デフォルト値は 4 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip インスタンス (rect) がクリックされたときに、その blurX プロパティを変更します。

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelBlurX");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.blurX = 10;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, 0.8, 0x0000FF, 0.8,
  20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

blurY (BevelFilter.blurY プロパティ)

```
public blurY : Number
```

垂直方向のぼかし量 (ピクセル単位) です。指定できる値は 0 ～ 255 (浮動小数) です。デフォルト値は 4 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip インスタンス (rect) がクリックされたときに、その blurY プロパティを変更します。

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelBlurY");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.blurY = 10;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, 0.8, 0x0000FF, 0.8,
  20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

clone (BevelFilter.clone メソッド)

```
public clone() : BevelFilter
```

このフィルタオブジェクトのコピーを返します。

対応バージョン : ActionScript 1.0、Flash Player 8

戻り値

`BevelFilter` - 元の BevelFilter インスタンスとプロパティがすべて同じである新しい BevelFilter インスタンス。

例

次の例では、3 つの BevelFilter オブジェクトを作成し、それらを比較します。BevelFilter コンストラクタを使用することにより、filter_1 オブジェクトを作成できます。filter_1 と等しくなるよう filter_2 オブジェクトを作成します。filter_1 のクローンを作成して、clonedFilter を作成します。filter_2 が filter_1 と等しいと評価された場合に、filter_1 と同じ値を含んでいたとしても、clonedFilter は等しいと評価されないことに注意してください。

```
import flash.filters.BevelFilter;

var filter_1:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, 0.8, 0x0000FF, 0.8,
  20, 20, 1, 3, "inner", false);
var filter_2:BevelFilter = filter_1;
var clonedFilter:BevelFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false

for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> type: inner
  // >> blurY: 20
  // >> blurX: 20
  // >> knockout: false
  // >> strength: 1
  // >> quality: 3
    // >> shadowAlpha: 0.8
  // >> shadowColor: 255
  // >> highlightAlpha: 0.8
  // >> highlightColor: 16776960
  // >> angle: 45
  // >> distance: 5
}
```

```
for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
  // >> type: inner
  // >> blurY: 20
  // >> blurX: 20
  // >> knockout: false
  // >> strength: 1
  // >> quality: 3
  // >> shadowAlpha: 0.8
  // >> shadowColor: 255
  // >> highlightAlpha: 0.8
  // >> highlightColor: 16776960
  // >> angle: 45
  // >> distance: 5
}
```

filter_1、filter_2、および clonedFilter の関係をさらに詳しく示すために、次の例では、filter_1 の knockout プロパティを変更します。knockout を変更すると、clone() メソッドが、値を参照する代わりに、filter_1 の値に基づいてインスタンスを作成します。

```
import flash.filters.BevelFilter;

var filter_1:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, 0.8, 0x0000FF, 0.8,
  20, 20, 1, 3, "inner", false);
var filter_2:BevelFilter = filter_1;
var clonedFilter:BevelFilter = filter_1.clone();

trace(filter_1.knockout); // false
trace(filter_2.knockout); // false
trace(clonedFilter.knockout); // false

filter_1.knockout = true;

trace(filter_1.knockout); // true
trace(filter_2.knockout); // true
trace(clonedFilter.knockout); // false
```

distance (BevelFilter.distance プロパティ)

```
public distance : Number
```

ベベルのオフセット距離です。この値はピクセル単位で指定します (浮動小数値)。デフォルト値は 4 です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip インスタンス (rect) がクリックされたときに、その distance プロパティを変更します。

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelDistance");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.distance = 3;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, 0.8, 0x0000FF, 0.8,
  20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

highlightAlpha (BevelFilter.highlightAlpha プロパティ)

```
public highlightAlpha : Number
```

ハイライトカラーのアルファ透明度の値です。値には、0 ～ 1 の正規化した値を指定します。たとえば 0.25 を指定すると、透明値 25% が設定されます。デフォルト値は 1 です。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip インスタンス (rect) がクリックされたときに、その highlightAlpha プロパティを変更します。

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelHighlightAlpha");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.highlightAlpha = 0.2;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, 0.8, 0x0000FF, 0.8,
  20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

highlightColor (BevelFilter.highlightColor プロパティ)

```
public highlightColor : Number
```

ベベルのハイライトカラーです。有効な値は 16 進数形式 (*0xRRGGBB*) です。デフォルト値は 0xFFFFFF. です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip インスタンス (`rect`) がクリックされたときに、その `highlightColor` プロパティを変更します。

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelHighlightColor");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.highlightColor = 0x0000FF;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, 0.8, 0x0000FF, 0.8,
  20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

knockout (BevelFilter.knockout プロパティ)

```
public knockout : Boolean
```

true の場合は、ノックアウト効果を適用します。その結果、オブジェクトは完全に透明になり、ドキュメントの背景色で表示されます。デフォルト値は false (ノックアウトなし) です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip インスタンス (rect) がクリックされたときに、その knockout プロパティを変更します。

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelKnockout");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.knockout = true;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, 0.8, 0x0000FF, 0.8,
  20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

quality (BevelFilter.quality プロパティ)

```
public quality : Number
```

フィルタを適用する回数。デフォルト値は 1(低品質) です。値 2 は標準の品質であり、値 3 は高品質です。フィルタに設定された値が小さいほど、速くレンダリングできます。

多くのアプリケーションでは、quality 値 1、2、または 3 で十分です。最大 15 までの値を使用してさまざまな効果を出すことができますが、値が大きくなるほどレンダリング速度が低下します。多くの場合、quality の値を大きくする代わりに blurX と blurY の値を大きくするだけで、同様の効果が得られます。この方法を実行すると、より高速にレンダリングされます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip インスタンス (rect) がクリックされたときに、その quality プロパティを変更します。

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelQuality");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.quality = 1;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, 0.8, 0x0000FF, 0.8,
  20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

shadowAlpha (BevelFilter.shadowAlpha プロパティ)

```
public shadowAlpha : Number
```

シャドウカラーのアルファ透明度の値です。この値には、0 ～ 1 の正規化した値を指定します。たとえば 0.25 を指定すると、透明値 25% が設定されます。デフォルト値は 1 です。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip インスタンス (rect) がクリックされたときに、その shadowAlpha プロパティを変更します。

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelShadowAlpha");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.shadowAlpha = 0.2;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, 0.8, 0x0000FF, 0.8,
  20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

shadowColor (BevelFilter.shadowColor プロパティ)

```
public shadowColor : Number
```

ベベルのシャドウカラーです。有効な値は 16 進数形式 (*0xRRGGBB*) です。デフォルト値は 0x000000 です。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip インスタンス (`rect`) がクリックされたときに、その `shadowColor` プロパティを変更します。

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelShadowColor");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.shadowColor = 0xFFFF00;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, 0.8, 0x0000FF, 0.8,
  20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

strength (BevelFilter.strength プロパティ)

```
public strength : Number
```

インプリントの強さまたは広がり。指定できる値は 0 ～ 255 です。値が大きいほど、濃い色がインプリントされるので、ベベルと背景との間のコントラストが強くなります。デフォルト値は 1 です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip インスタンス (rect) がクリックされたときに、その strength プロパティを変更します。

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelStrength");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.strength = 10;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, 0.8, 0x0000FF, 0.8,
  20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

type (BevelFilter.type プロパティ)

```
public type : String
```

ベベルの種類です。有効な値は "inner"、"outer"、および "full" です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip インスタンス (rect) がクリックされたときに、その type プロパティを変更します。

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelType");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.type = "outer";
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, 0.8, 0x0000FF, 0.8,
  20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

BitmapData (flash.display.BitmapData)

```
Object
  |
  +-flash.display.BitmapData
```

```
public class BitmapData
extends Object
```

BitmapData クラスでは、任意のサイズの透明または不透明のビットマップイメージを作成し、実行時にさまざまな方法で操作できます。

このクラスを使用すると、ビットマップのレンダリング処理を Flash Player 内部の表示更新ルーチンから分離できます。BitmapData オブジェクトを直接操作することで複雑なイメージを作成できるので、ベクターデータのコンテンツを連続的に再描画するフレーム単位のオーバーヘッドを避けることができます。

BitmapData クラスのメソッドは、汎用のフィルタインターフェイスを通じて使用できない各種の効果に対応しています。

BitmapData オブジェクトには、ピクセルデータの配列が含まれています。このデータは、完全に不透明なビットマップも、アルファチャンネルデータを含む透明なビットマップも表現できます。いずれの種類の BitmapData オブジェクトも 32 ビット整数のバッファとして保存されます。各 32 ビット整数は、ビットマップ内の 1 つのピクセルのプロパティを決定します。

各 32 ビット整数は、アルファ透明度とピクセルの赤緑青 (ARGB) の値を表す 4 つの 8 ビットチャンネル値 (0 ～ 255) の組み合わせです。

`BitmapData.copyChannel()` メソッド、または `DisplacementMapFilter.componentX` プロパティと `DisplacementMapFilter.componentY` プロパティで 4 つのチャネル (赤、緑、青、およびアルファ) を使用する場合、これらのチャネルは、次のように数値として表現されます。

- `1` (赤)
- `2` (緑)
- `4` (青)
- `8` (アルファ)

`MovieClip.attachBitmap()` メソッドにより、BitmapData オブジェクトを MovieClip オブジェクトに関連付けることができます。

`MovieClip.beginBitmapFill()` メソッドにより、BitmapData オブジェクトを使用してムービークリップ内の領域を塗りつぶすことができます。

BitmapData オブジェクトの最大の幅と高さは 2880 ピクセルです。

BitmapData オブジェクトのいずれかのメソッドやプロパティを呼び出したとき、BitmapData オブジェクトが無効であれば (たとえば `height == 0` かつ `width == 0` の場合)、呼び出しは失敗して、数値を返すプロパティとメソッドは -1 を返します。

対応バージョン : ActionScript 1.0、Flash Player 8

関連項目

attachBitmap (MovieClip.attachBitmap メソッド),
beginBitmapFill (MovieClip.beginBitmapFill メソッド)

プロパティ一覧

オプション	プロパティ	説明
	height:Number (読み取り専用)	ビットマップイメージの高さ (ピクセル単位) です。
	rectangle:Rectangle (読み取り専用)	ビットマップイメージのサイズと位置を定義する矩形です。
	transparent:Boolean (読み取り専用)	ビットマップイメージがピクセル単位の透明度をサポートするかどうかを定義します。
	width:Number (読み取り専用)	ビットマップイメージの幅 (ピクセル単位) です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター一覧

署名	説明
BitmapData(width:Number, height:Number, [transparent:Boolean], [fillColor:Number])	指定された幅と高さで BitmapData オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	applyFilter (sourceBitmap: BitmapData, sourceRect:Rectangle, destPoint:Point, filter:BitmapFilter) : Number	ソースイメージとフィルタオブジェクトを受け取り、フィルタを適用して得られるイメージを生成します。
	clone() : BitmapData	新しい BitmapData オブジェクトとして、元のインスタンスのクローンを返します。含まれるビットマップはまったく同じコピーになります。
	colorTransform(rect: Rectangle, colorTransform: ColorTransform) : Void	ColorTransform オブジェクトを使用して、ビットマップイメージの特定領域のカラー値を調整します。
	compare (otherBitmapData: BitmapData) : Object	2 つの BitmapData オブジェクトを比較します。
	copyChannel (sourceBitmap: BitmapData, sourceRect:Rectangle, destPoint:Point, sourceChannel:Number, destChannel:Number) : Void	別の BitmapData オブジェクトまたは現在の BitmapData オブジェクトの 1 つのチャンネルのデータを現在の BitmapData オブジェクトのチャンネルにデータを転送します。
	copyPixels (sourceBitmap: BitmapData, sourceRect:Rectangle, destPoint:Point, [alphaBitmap: BitmapData], [alphaPoint:Point], [mergeAlpha:Boolean]) : Void	イメージ間のピクセル操作 (伸長、回転、カラー効果なし) を高速に実行するルーチンを提供します。
	dispose() : Void	BitmapData オブジェクトを格納するために使用するメモリを解放します。

オプション	署名	説明
	draw(source:Object, [matrix:Matrix], [colorTransform: ColorTransform], [blendMode:Object], [clipRect:Rectangle], [smooth:Boolean]) : Void	Flash Player のベクターレンダーを使用して、ソースイメージやソースムービークリップをターゲットイメージ上に描画します。
	fillRect(rect: Rectangle, color:Number) : Void	指定された ARGB カラーで矩形領域のピクセルを塗りつぶします。
	floodFill(x:Number, y:Number, color:Number) : Void	(*x*, *y*) 座標を始点として所定の色で塗りつぶすことにより、イメージの塗りつぶし処理を実行します。
	generateFilterRect (sourceRect:Rectangle, filter:BitmapFilter) : Rectangle	BitmapData オブジェクト、ソース矩形、フィルタオブジェクトを指定して、applyFilter() 呼び出しによって影響を受ける矩形領域を決定します。
	getColorBoundsRect (mask:Number, color:Number, [findColor:Boolean]) : Rectangle	ビットマップイメージ内のピクセルのうち、指定された色のすべてのピクセルを完全に含む矩形領域を決定します。
	getPixel(x:Number, y:Number) : Number	BitmapData オブジェクトの特定ポイント (*x*, *y*) の RGB ピクセル値を表す整数を返します。
	getPixel32(x:Number, y:Number) : Number	アルファチャンネルデータと RGB データを含む ARGB カラー値を返します。
	hitTest(firstPoint: Point, firstAlphaThreshold: Number, secondObject:Object, [secondBitmapPoint: Point], [secondAlphaThreshold: Number]) : Boolean	1 つのビットマップイメージと、ポイント、矩形、または他のビットマップイメージとの間でピクセルレベルのヒット検出を実行します。
static	loadBitmap (id:String) : BitmapData	ライブラリ内の指定されたリンケージ識別子で識別されるシンボルのビットマップイメージ表現を含む、新しい BitmapData オブジェクトを返します。

オプション	署名	説明
	merge(sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point, redMult:Number, greenMult:Number, blueMult:Number, alphaMult:Number) : Void	ソースイメージとターゲットイメージをチャンネルごとにブレンドします。
	noise(randomSeed:Number, [low:Number], [high:Number], [channelOptions:Number], [grayScale:Boolean]) : Void	ランダムノイズを表すピクセルでイメージを塗ります。
	paletteMap (sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point, [redArray:Array], [greenArray:Array], [blueArray:Array], [alphaArray:Array]) : Void	指定された最大 4 つのカラーパレットデータ配列 (チャンネルごとに 1 つの配列) を使用して、イメージ内のカラーチャンネル値をマッピングし直します。
	perlinNoise(baseX:Number, baseY:Number, numOctaves:Number, randomSeed:Number, stitch:Boolean, fractalNoise:Boolean, [channelOptions:Number], [grayScale:Boolean], [offsets:Object]) : Void	Perlin ノイズイメージを生成します。

オプション	署名	説明
	pixelDissolve (sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point, [randomSeed:Number], [numberOfPixels:Number], [fillColor:Number]) : Number	ソースイメージからターゲットイメージへのピクセルディゾルブ、または同じイメージを使用することによるピクセルディゾルブを実行します。
	scroll(x:Number, y:Number) : Void	所定の (*x*, *y*) ピクセル量だけイメージをスクロールします。
	setPixel(x:Number, y:Number, color:Number) : Void	BitmapData オブジェクトの 1 つのピクセルに色を設定します。
	setPixel32(x:Number, y:Number, color:Number) : Void	BitmapData オブジェクトの 1 つのピクセルにカラー値とアルファ透明度を設定します。
	threshold (sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point, operation:String, threshold:Number, [color:Number], [mask:Number], [copySource:Boolean]) : Number	指定されたしきい値に照らしてイメージのピクセル値をテストし、テストに合格したピクセルに新しいカラー値を設定します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

applyFilter (BitmapData.applyFilter メソッド)

```
public applyFilter(sourceBitmap:BitmapData, sourceRect:Rectangle,
  destPoint:Point, filter:BitmapFilter) : Number
```

ソースイメージとフィルタオブジェクトを受け取り、フィルタを適用して得られるイメージを生成します。

このメソッドはビルトインフィルタオブジェクトの動作によって変わります。そのオブジェクトには、入力ソース矩形によって影響を受けるターゲット矩形を決定するコードがあります。

フィルタを適用した後、結果として得られるイメージが入力イメージよりも大きくなることがあります。たとえば、BlurFilter クラスを使用してソース矩形 (50,50,100,100) とターゲットポイント (10,10) をぼかすと、ターゲットイメージで変更される領域は、ぼかしのために、(10,10,60,60) よりも大きくなります。このことは、`applyFilter()` 呼び出し中に内部で発生します。

`sourceBitmapData` パラメータの `sourceRect` パラメータが内側領域 (200 x 200 のイメージ内の (50,50,100,100) など) である場合、フィルタは、`sourceRect` パラメータの外側にあるソースピクセルを使用して、ターゲット矩形を生成します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`sourceBitmap:BitmapData` - 使用する入力ビットマップイメージ。ソースイメージは、別の BitmapData オブジェクトにすることも、現在の BitmapData インスタンスを参照することもできます。

`sourceRect:Rectangle` - 入力として使用するソースイメージの領域を定義する矩形。

`destPoint:Point` - ソース矩形の左上隅に対応するターゲットイメージ (現在の BitmapData インスタンス) 内のポイント。

`filter:BitmapFilter` - フィルタ適用処理を実行する場合に使用するフィルタオブジェクト。各種類のフィルタは、次のように所定の要件を備えています。

- BlurFilter – このフィルタは、透明または不透明なソースイメージとターゲットイメージを使用できます。イメージのフォーマットが一致しない場合、フィルタ適用中に作成されるソースイメージのコピーがターゲットイメージのフォーマットに合わせられます。
- BevelFilter, DropShadowFilter, GlowFilter – これらのフィルタのターゲットイメージは透明イメージでなければなりません。DropShadowFilter または GlowFilter を呼び出すと、ドロップシャドウまたはグローのアルファチャンネルデータを含むイメージが作成されます。ドロップシャドウはターゲットイメージ上に作成されません。これらのフィルタを不透明なターゲットイメージに対して使用すると、エラーコード値 -6 が返されます。
- ConvolutionFilter – このフィルタは、透明または不透明なソースイメージとターゲットイメージを使用できます。
- ColorMatrixFilter – このフィルタは、透明または不透明なソースイメージとターゲットイメージを使用できます。

■ DisplacementMapFilter – このフィルタは、透明または不透明なソースイメージとターゲットイメージを使用できますが、イメージの形式はソースとターゲットで同じでなければなりません。

戻り値

Number - フィルタが正常に適用されたかどうかを示す数値。フィルタが正常に適用されると、ゼロが返されます。フィルタの適用中にエラーが発生すると、負の数値が返されます。

例

次の例では、BitmapData インスタンスにベベルフィルタを適用する方法について説明します。

```
import flash.display.BitmapData;
import flash.filters.BevelFilter;
import flash.geom.Point;

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xCCCCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, 0.8, 0x0000FF, 0.8, 20,
  20, 1, 3, "inner", false);

mc.onPress = function() {
  myBitmapData.applyFilter(myBitmapData, myBitmapData.rectangle, new Point(0,
  0), filter);
}
```

関連項目

BevelFilter (flash.filters.BevelFilter),
BlurFilter (flash.filters.BlurFilter),
ColorMatrixFilter (flash.filters.ColorMatrixFilter),
ConvolutionFilter (flash.filters.ConvolutionFilter),
DisplacementMapFilter (flash.filters.DisplacementMapFilter),
DropShadowFilter (flash.filters.DropShadowFilter),
GlowFilter (flash.filters.GlowFilter),filters (MovieClip.filters プロパティ)

BitmapData コンストラクタ

```
public BitmapData(width:Number, height:Number, [transparent:Boolean],
  [fillColor:Number])
```

指定された幅と高さで BitmapData オブジェクトを作成します。fillColor パラメータに値を指定した場合、ビットマップのすべてのピクセルにその色が設定されます。

transparent パラメータに false を渡さない限り、デフォルトではビットマップが透明として作成されます。不透明のビットマップを作成した後、それを透明のビットマップに変更することはできません。不透明のビットマップに含まれるすべてのピクセルは、24 ビットのカラーチャンネル情報だけを使用します。ビットマップを transparent と定義した場合、すべてのピクセルは、アルファ透明チャンネルを含む 32 ビットのカラーチャンネル情報を使用します。

BitmapData オブジェクトの最大の幅と高さは 2880 ピクセルです。width または height に 2880 よりも大きい値を指定すると、新しいインスタンスは作成されません。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

width:Number - ビットマップイメージの幅 (ピクセル単位) です。

height:Number - ビットマップイメージの高さ (ピクセル単位) です。

transparent:Boolean (オプション) - ビットマップイメージがピクセル単位の透明度をサポートするかどうかを定義します。デフォルト値は true です (透明)。完全に透明なビットマップを作成するには、transparent パラメータの値を true に、fillColor パラメータの値を 0x00000000 (または 0) に設定します。

fillColor:Number (オプション) - ビットマップイメージ領域を塗りつぶすのに使用する 32 ビット ARGB カラー値です。デフォルト値は 0xFFFFFFFF (白) です。

例

次の例では、新しい BitmapData オブジェクトを作成します。この例で使用する値は、transparent パラメータと fillColor パラメータのデフォルト値です。こうしたパラメータがなくても、コンストラクタを呼び出して、同じ結果を得ることができます。

```
import flash.display.BitmapData;

var width:Number = 100;
var height:Number = 80;
var transparent:Boolean = true;
var fillColor:Number = 0xFFFFFFFF;

var bitmap_1:BitmapData = new BitmapData(width, height, transparent, fillColor);

trace(bitmap_1.width); // 100
trace(bitmap_1.height); // 80
```

```
trace(bitmap_1.transparent); // true

var bitmap_2:BitmapData = new BitmapData(width, height);

trace(bitmap_2.width); // 100
trace(bitmap_2.height); // 80
trace(bitmap_2.transparent); // true
```

clone (BitmapData.clone メソッド)

```
public clone() : BitmapData
```

新しい BitmapData オブジェクトとして、元のインスタンスのクローンを返します。含まれるビットマップはまったく同じコピーになります。

対応バージョン : ActionScript 1.0、Flash Player 8

戻り値

`BitmapData` - 元のオブジェクトと同一の新しい BitmapData オブジェクト。

例

次の例では、3 つの BitmapData オブジェクトを作成し、それらを比較します。BitmapData コンストラクタを使用することにより、bitmap_1 インスタンスを作成します。bitmap_1 と等しくなるよう bitmap_2 インスタンスを作成します。bitmap_1 のクローンを作成することにより、clonedBitmap インスタンスを作成します。bitmap_2 が bitmap_1 と等しいと評価される場合に、たとえ bitmap_1 と同じ値を含んでいても clonedBitmap は等しいと評価されないことに注意してください。

```
import flash.display.BitmapData;

var bitmap_1:BitmapData = new BitmapData(100, 80, false, 0x000000);
var bitmap_2:BitmapData = bitmap_1;
var clonedBitmap:BitmapData = bitmap_1.clone();

trace(bitmap_1 == bitmap_2); // true
trace(bitmap_1 == clonedBitmap); // false

for(var i in bitmap_1) {
  trace(">> " + i + ": " + bitmap_1[i]);
  // >> generateFilterRect: [type Function]
  // >> dispose: [type Function]
  // >> clone: [type Function]
  // >> copyChannel: [type Function]
  // >> noise: [type Function]
  // >> merge: [type Function]
  // >> paletteMap: [type Function]
  // >> hitTest: [type Function]
  // >> colorTransform: [type Function]
```

```
  // >> perlinNoise: [type Function]
  // >> getColorBoundsRect: [type Function]
  // >> floodFill: [type Function]
  // >> setPixel32: [type Function]
  // >> getPixel32: [type Function]
  // >> pixelDissolve: [type Function]
  // >> draw: [type Function]
  // >> threshold: [type Function]
  // >> scroll: [type Function]
  // >> applyFilter: [type Function]
  // >> copyPixels: [type Function]
  // >> fillRect: [type Function]
  // >> setPixel: [type Function]
  // >> getPixel: [type Function]
  // >> transparent: false
  // >> rectangle: (x=0, y=0, w=100, h=80)
  // >> height: 80
  // >> width: 100
}

for(var i in clonedBitmap) {
  trace(">> " + i + ": " + clonedBitmap[i]);
  // >> generateFilterRect: [type Function]
  // >> dispose: [type Function]
  // >> clone: [type Function]
  // >> copyChannel: [type Function]
  // >> noise: [type Function]
  // >> merge: [type Function]
  // >> paletteMap: [type Function]
  // >> hitTest: [type Function]
  // >> colorTransform: [type Function]
  // >> perlinNoise: [type Function]
  // >> getColorBoundsRect: [type Function]
  // >> floodFill: [type Function]
  // >> setPixel32: [type Function]
  // >> getPixel32: [type Function]
  // >> pixelDissolve: [type Function]
  // >> draw: [type Function]
  // >> threshold: [type Function]
  // >> scroll: [type Function]
  // >> applyFilter: [type Function]
  // >> copyPixels: [type Function]
  // >> fillRect: [type Function]
  // >> setPixel: [type Function]
  // >> getPixel: [type Function]
  // >> transparent: false
  // >> rectangle: (x=0, y=0, w=100, h=80)
  // >> height: 80
  // >> width: 100
}
```

bitmap_1、bitmap_2、および clonedBitmap 間の関係をさらに詳しく示すため、次の例では bitmap_1 の (1, 1) のピクセル値を変更します。(1, 1) のピクセル値を変更すると、clone() メソッドは値を参照する代わりに、bitmap_1 インスタンスの値に基づいてインスタンスを作成します。

```
import flash.display.BitmapData;

var bitmap_1:BitmapData = new BitmapData(100, 80, false, 0x000000);
var bitmap_2:BitmapData = bitmap_1;
var clonedBitmap:BitmapData = bitmap_1.clone();

trace(bitmap_1.getPixel32(1, 1)); // -16777216
trace(bitmap_2.getPixel32(1, 1)); // -16777216
trace(clonedBitmap.getPixel32(1, 1)); // -16777216

bitmap_1.setPixel32(1, 1, 0xFFFFFF);

trace(bitmap_1.getPixel32(1, 1)); // -1
trace(bitmap_2.getPixel32(1, 1)); // -1
trace(clonedBitmap.getPixel32(1, 1)); // -16777216
```

colorTransform (BitmapData.colorTransform メソッド)

```
public colorTransform(rect:Rectangle, colorTransform:ColorTransform) : Void
```

ColorTransform オブジェクトを使用して、ビットマップイメージの特定領域のカラー値を調整します。矩形がビットマップイメージの境界と一致する場合、このメソッドはイメージ全体のカラー値を変換します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`rect:Rectangle` - ColorTransform オブジェクトが適用されるイメージの領域を定義する Rectangle オブジェクト。

`colorTransform:ColorTransform` - 適用するカラー変換値が記述されている ColorTransform オブジェクト。

例

次の例では、BitmapData インスタンスにカラー変換処理を適用する方法について説明します。

```
import flash.display.BitmapData;
import flash.geom.ColorTransform;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
```

```
mc.onPress = function() {
  myBitmapData.colorTransform(myBitmapData.rectangle, new ColorTransform(1, 0,
  0, 1, 255, 0, 0, 0));
}
```

関連項目

ColorTransform (flash.geom.ColorTransform), Rectangle (flash.geom.Rectangle)

compare (BitmapData.compare メソッド)

```
public compare(otherBitmapData:BitmapData) : Object
```

2 つの BitmapData オブジェクトを比較します。2 つの BitmapData オブジェクトのサイズ (幅と高さ) が同じであれば、メソッドは新しい BitmapData オブジェクトを返します。この新しいオブジェクトの各ピクセルは、2 つのソースオブジェクトのピクセル間の「差分」です。

- 2 つのピクセルが等しい場合、差分ピクセルは 0x00000000 です。
- 2 つのピクセルの RGB 値が異なる場合 (アルファ値は無視)、差分ピクセルは 0xFFRRGGBB です。ここで RR/GG/BB はそれぞれ赤、緑、青チャンネル間の個々の差分値を表します。この場合、アルファチャンネルの差分は無視されます。
- アルファチャンネル値だけが異なる場合、ピクセル値は 0x *ZZ* FFFFFF です (*ZZ* はアルファ値の差分)。

たとえば、次のような 2 つの BitmapData オブジェクトがあるとします。

```
var bmd1:BitmapData = new BitmapData(50, 50, true, 0xFFFF0000);
var bmd2:BitmapData = new BitmapData(50, 50, true, 0xCCFFAA00);
var diffBmpData:BitmapData = bmd1.compare(bmd2);
```

メモ : 2 つの BitmapData オブジェクトを塗りつぶすために使用されるそれぞれの色の RGB 値はわずかに異なります (0xFF0000 と 0xFFAA00)。compare() メソッドの結果として新しい BitmapData オブジェクトが生成され、その各ピクセルは 2 つのビットマップ間の RGB 値の差分を示します。

次のような 2 つの BitmapData オブジェクトがあるとします。両者の RGB カラーは同じですが、アルファ値は異なります。

```
var bmd1:BitmapData = new BitmapData(50, 50, true, 0xFFFFAA00);
var bmd2:BitmapData = new BitmapData(50, 50, true, 0xCCFFAA00);
var diffBmpData:BitmapData = bmd1.compare(bmd2);
```

compare() メソッドの結果として新しい BitmapData オブジェクトが生成され、その各ピクセルは 2 つのビットマップ間のアルファ値の差分を示します。

BitmapData オブジェクトが等しい (幅と高さ、およびピクセル値が同じ) 場合、このメソッドは数値 0 を返します。

引数が渡されない場合、または引数が BitmapData オブジェクトでない場合、このメソッドは -1 を返します。

いずれかの BitmapData オブジェクトが既に破棄されている場合、このメソッドは -2 を返します。

BitmapData オブジェクトの幅が等しくない場合、高さが同じであれば、このメソッドは数値 -3 を返します。

BitmapData オブジェクトの高さが等しくない場合、幅が同じであれば、このメソッドは数値 -4 を返します。

次の例では、幅の異なる 2 つの Bitmap オブジェクトを比較します (それぞれの幅は 50 と 60)。

```
var bmd1:BitmapData = new BitmapData(100, 50, false, 0xFFFF0000);
var bmd2:BitmapData = new BitmapData(100, 60, false, 0xFFFFAA00);
trace(bmd1.compare(bmd2)); // -3
```

対応バージョン : ActionScript 1.0、Flash Player 9

パラメータ

`otherBitmapData:BitmapData` - ソース BitmapData オブジェクトと比較される BitmapData オブジェクト。

戻り値

`Object` - 2 つの BitmapData オブジェクトのサイズ (幅と高さ) が同じであれば、このメソッドは、2 つのオブジェクト間の差分を示す新しい BitmapData オブジェクトを返します (主な説明の項を参照してください)。2 つの BitmapData オブジェクトが等しい場合、このメソッドは数値 0 を返します。引数が渡されない場合、または引数が BitmapData オブジェクトでない場合には、メソッドは -1 を返します。いずれかの BitmapData オブジェクトが既に破棄されている場合、メソッドは -2 を返します。BitmapData オブジェクトの幅が等しくない場合、メソッドは数値 -3 を返します。BitmapData オブジェクトの高さが等しくない場合、メソッドは数値 -4 を返します。

copyChannel (BitmapData.copyChannel メソッド)

```
public copyChannel(sourceBitmap:BitmapData, sourceRect:Rectangle,
  destPoint:Point, sourceChannel:Number, destChannel:Number) : Void
```

別の BitmapData オブジェクトまたは現在の BitmapData オブジェクトの 1 つのチャンネルのデータを現在の BitmapData オブジェクトのチャンネルにデータを転送します。ターゲット BitmapData オブジェクトの他のチャンネルのデータはすべて保たれます。

ソースチャンネルの値とターゲットチャンネルの値は、次のいずれかの値 (またはいずれか複数の値の合計) です。

- 1 (赤)
- 2 (緑)
- 4 (青)
- 8 (アルファ)

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`sourceBitmap:BitmapData` - 使用する入力ビットマップイメージ。ソースイメージは、別の BitmapData オブジェクトにすることも、現在の BitmapData オブジェクトを参照することもできます。

`sourceRect:Rectangle` - ソース側の Rectangle オブジェクト。ビットマップ内のより小さな領域からチャンネルデータをコピーするだけの場合、BitmapData オブジェクトのサイズ全体よりも小さいソース矩形を指定します。

`destPoint:Point` - 新しいチャンネルデータを配置する矩形領域の左上隅を表すターゲット側の Point オブジェクト。ある領域のチャンネルデータをターゲットイメージ内の別の領域にコピーする場合は、(0,0) 以外のポイントを指定します。

`sourceChannel:Number` - ソースチャネルです。(1,2,4,8) というセットからの 1 つの値、またはそのうちの複数の値の合計を使用します。セットは、それぞれ赤、緑、青、アルファの各チャンネルを表します。

`destChannel:Number` - ターゲットチャンネル。(1,2,4,8) というセットからの 1 つの値、またはそのうちの複数の値の合計を使用します。セットは、それぞれ赤、緑、青、アルファの各チャンネルを表します。

例

次の例では、BitmapData オブジェクトのソース ARGB チャンネルを、別の場所にあるソース ARGB チャンネル自体にコピーする方法を示します。

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

mc.onPress = function() {
  myBitmapData.copyChannel(myBitmapData, new Rectangle(0, 0, 50, 80), new
  Point(51, 0), 3, 1);
}
```

関連項目

Rectangle (flash.geom.Rectangle)

copyPixels (BitmapData.copyPixels メソッド)

```
public copyPixels(sourceBitmap:BitmapData, sourceRect:Rectangle,
  destPoint:Point, [alphaBitmap:BitmapData], [alphaPoint:Point],
  [mergeAlpha:Boolean]) : Void
```

イメージ間のピクセル操作 (伸長、回転、カラー効果なし) を高速に実行するルーチンを提供します。このメソッドは、ソースイメージの矩形領域を、ターゲット BitmapData オブジェクトのターゲットポイントにある同じサイズの矩形領域にコピーします。

alphaBitmap および alphaPoint パラメータを指定すると、2 番目のイメージをソースイメージのアルファソースとして使用できます。ソースイメージにアルファデータがある場合、両方のアルファデータセットを使用して、ソースイメージのピクセルがソースイメージ上に合成されます。alphaPoint パラメータは、アルファイメージ内のポイントであり、ソース側矩形の左上隅に対応するものです。ソースイメージとアルファイメージが交わらない部分のピクセルは、ターゲットイメージにコピーされません。

mergeAlpha プロパティは、透明なイメージを別の透明なイメージにコピーするときにアルファチャンネルを使用するかどうかを制御します。アルファを使用しないで単純にピクセルをコピーする場合は、mergeAlpha プロパティを false に設定します。そうすると、ソースからターゲットにすべてのピクセルがコピーされます。mergeAlpha プロパティのデフォルト値は false です。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

sourceBitmap:BitmapData - ピクセルのコピー元になる入力ビットマップイメージ。ソースイメージは、別の BitmapData インスタンスにすることも、現在の BitmapData インスタンスを参照することもできます。

sourceRect:Rectangle - 入力として使用するソースイメージの領域を定義する矩形。

destPoint:Point - ターゲットポイント。新しいピクセルを配置する矩形領域の左上隅を表します。

alphaBitmap:BitmapData (オプション) - 第 2 のアルファ BitmapData オブジェクトソース。

alphaPoint:Point (オプション) - sourceRect パラメータの左上隅に対応するアルファ BitmapData オブジェクトソース内のポイント。

mergeAlpha:Boolean (オプション) - ブール値。アルファチャンネルを使用するには、値を true に設定します。アルファチャンネルを使用せずにピクセルをコピーするには、値を false に設定します。

例

次の例では、ある BitmapData インスタンスから別の BitmapData インスタンスにピクセルをコピーする方法を示します。

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var bitmapData_1:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);
var bitmapData_2:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc_1:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_1.attachBitmap(bitmapData_1, this.getNextHighestDepth());

var mc_2:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_2.attachBitmap(bitmapData_2, this.getNextHighestDepth());
mc_2._x = 101;

mc_1.onPress = function() {
  bitmapData_2.copyPixels(bitmapData_1, new Rectangle(0, 0, 50, 80), new
  Point(51, 0));
}

mc_2.onPress = function() {
  bitmapData_1.copyPixels(bitmapData_2, new Rectangle(0, 0, 50, 80), new
  Point(51, 0));
}
```

dispose (BitmapData.dispose メソッド)

```
public dispose() : Void
```

BitmapData オブジェクトを格納するために使用するメモリを解放します。

このメソッドをイメージに対して呼び出すと、イメージの幅と高さがゼロに設定されます。BitmapData オブジェクトのメモリを解放した後にその BitmapData インスタンスのメソッドやプロパティを呼び出すと、失敗して値 -1 が返されます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、BitmapData インスタンスのメモリを解放する方法を示します。これにより、インスタンスはクリアな状態になります。

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

mc.onPress = function() {
  myBitmapData.dispose();

  trace(myBitmapData.width); // -1
  trace(myBitmapData.height); // -1
  trace(myBitmapData.transparent); // -1
}
```

draw (BitmapData.draw メソッド)

```
public draw(source:Object, [matrix:Matrix], [colorTransform:ColorTransform],
  [blendMode:Object], [clipRect:Rectangle], [smooth:Boolean]) : Void
```

Flash Player のベクターレンダーを使用して、ソースイメージやソースムービークリップをターゲットイメージ上に描画します。変換マトリックス、ColorTransform オブジェクト、ブレンドモード設定、およびターゲット Rectangle オブジェクトを指定することにより、レンダリングの実行方法を制御できます。必要に応じて、拡大・縮小するときにビットマップのスムージングを行うかどうかを指定することもできます。これは、ソースオブジェクトが BitmapData オブジェクトである場合にのみ機能します。

このメソッドは、オーサリングツールインターフェイス内のオブジェクトに対して標準ベクターレンダーを使用してオブジェクトを描画する方法に対応します。

ソース MovieClip オブジェクトは、この呼び出しでオンステージ変換を使用しません。ライブラリやファイル内に存在するように処理され、マトリックス変換、カラー変換、ブレンドモードはありません。独自の変換プロパティを使用してムービークリップを描画する場合は、Transform オブジェクトを使用して、各種の変換プロパティを渡すことができます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`source:Object` - 描画する BitmapData オブジェクト。

matrix:Matrix (オプション) - ビットマップの座標を拡大・縮小、回転、または変換する場合に使用する Matrix オブジェクト。オブジェクトが指定されていない場合、ビットマップイメージは変換されません。このパラメータを渡す必要があるが、イメージを変換したくない場合、このパラメータを、デフォルトの new Matrix() コンストラクタを使って作成される単位マトリックスに設定します。

colorTransform:ColorTransform (オプション) - ビットマップのカラー値を調整する場合に使用する ColorTransform オブジェクト。オブジェクトが指定されていない場合、ビットマップイメージのカラーは変換されません。このパラメータを渡す必要があるが、イメージを変換したくない場合、このパラメータを、デフォルトの new ColorTransform() コンストラクタを使って作成される ColorTransform オブジェクトに設定します。

blendMode:Object (オプション) - 変換のブレンドモード設定。このパラメータは 1 から 14 までの整数、またはストリング (たとえば "normal"、"darken") です。有効な blendMode の値については、MovieClip クラスの blendMode プロパティを参照してください。

clipRect:Rectangle (オプション) - Rectangle オブジェクト。この値を指定しない場合、切り取りは発生しません。

smooth:Boolean (オプション) - matrix パラメータに従って BitmapData オブジェクトを伸縮・回転するときにスムージング (補間) を行うかどうかを指定するブール値。デフォルト値は false です。smoothing パラメータは、source パラメータが BitmapData オブジェクトの場合にのみ適用されます。smoothing を false に指定した場合、回転・伸縮後の BitmapData イメージの形状がギザギザになる可能性があります。たとえば、次の 2 つのイメージは source パラメータとして同じ BitmapData オブジェクトを使用しますが、左側では smoothing パラメータが true に、右側では false に設定されます。

smoothing を true に設定してビットマップを描画する場合、smoothing を false に設定した場合よりも長い時間がかかります。

例

次の例では、ソース MovieClip インスタンスから BitmapData オブジェクトに描画する方法を示します。

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc_1:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_1.attachBitmap(myBitmapData, this.getNextHighestDepth());

var mc_2:MovieClip = createRectangle(50, 40, 0xFF0000);
mc_2._x = 101;
```

```
var myMatrix:Matrix = new Matrix();
myMatrix.rotate(Math.PI/2);

var translateMatrix:Matrix = new Matrix();
translateMatrix.translate(70, 15);

myMatrix.concat(translateMatrix);

var myColorTransform:ColorTransform = new ColorTransform(0, 0, 1, 1, 0, 0,
  255, 0);
var blendMode:String = "normal";

var myRectangle:Rectangle = new Rectangle(0, 0, 100, 80);
var smooth:Boolean = true;

mc_1.onPress = function() {
  myBitmapData.draw(mc_2, myMatrix, myColorTransform, blendMode, myRectangle,
  smooth);
}

function createRectangle(width:Number, height:Number, color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

関連項目

blendMode (MovieClip.blendMode プロパティ),
ColorTransform (flash.geom.ColorTransform),Matrix (flash.geom.Matrix),
Rectangle (flash.geom.Rectangle)

fillRect (BitmapData.fillRect メソッド)

```
public fillRect(rect:Rectangle, color:Number) : Void
```

指定された ARGB カラーで矩形領域のピクセルを塗りつぶします。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`rect:Rectangle` - 塗りつぶす矩形領域。

`color:Number` - 領域を塗りつぶす場合に使用する ARGB カラー値。ARGB カラー値は通常、16 進数形式 (たとえば、0xFF336699) で指定します。

例

次の例では、BitmapData の Rectangle によって定義された領域をカラーで塗りつぶす方法を示します。

```
import flash.display.BitmapData;
import flash.geom.Rectangle;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

mc.onPress = function() {
  myBitmapData.fillRect(new Rectangle(0, 0, 50, 40), 0x00FF0000);
}
```

関連項目

Rectangle (flash.geom.Rectangle)

floodFill (BitmapData.floodFill メソッド)

```
public floodFill(x:Number, y:Number, color:Number) : Void
```

(*x* , *y*) 座標を始点として所定の色で塗りつぶすことにより、イメージの塗りつぶし処理を実行します。floodFill() メソッドは、各種のペイントプログラムのバケツツールのようなものです。color は、アルファ情報とカラー情報を含む ARGB カラーです。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`x:Number` - イメージの x 座標。

`y:Number` - イメージの y 座標。

`color:Number` - 塗りとして使用する ARGB カラー。ARGB カラー値は通常、16 進数形式 (たとえば、0xFF336699) で指定します。

例

次の例では、BitmapData オブジェクト内のマウスをクリックした地点を始点として、イメージへのカラーの塗りつぶしを適用する方法を示します。

```
import flash.display.BitmapData;
import flash.geom.Rectangle;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

myBitmapData.fillRect(new Rectangle(0, 0, 50, 40), 0x00FF0000);

mc.onPress = function() {
  myBitmapData.floodFill(_xmouse, _ymouse, 0x000000FF);
}
```

generateFilterRect (BitmapData.generateFilterRect メソッド)

`public generateFilterRect(sourceRect:Rectangle, filter:BitmapFilter) : Rectangle`

BitmapData オブジェクト、ソース矩形、フィルタオブジェクトを指定して、`applyFilter()` メソッド呼び出しによって影響を受ける矩形領域を決定します。

たとえば、ぼかしフィルタは通常、元のイメージのサイズよりも大きい領域に影響します。デフォルトの BlurFilter インスタンス (`blurX = blurY = 4`) によってフィルタが適用される 100 x 200 ピクセルのイメージは `(-2,-2,104,204)` というターゲット矩形を生成します。`generateFilterRect()` メソッドを使用すると、このターゲット矩形のサイズを前もって知ることができるので、フィルタ処理の前にターゲットイメージを適切なサイズにすることができます。

一部のフィルタでは、ソースイメージのサイズに基づいてターゲット矩形がクリッピングされる場合があります。たとえば、内側の `DropShadow` は、ソースイメージよりも大きい結果を生成しません。この API では、ソース `rect` パラメータではなく、BitmapData オブジェクトをソースの境界として使用します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`sourceRect:Rectangle` - 入力として使用するソースイメージの領域を定義する矩形。

`filter:BitmapFilter` - ターゲット矩形を算出するために使用するフィルタオブジェクト。

戻り値

`Rectangle` - イメージ、sourceRect パラメータ、およびフィルタを使用して算出されるターゲット矩形。

例

次の例では、applyfilter() メソッドが影響を及ぼすターゲット矩形を決定する方法を示します。

```
import flash.display.BitmapData;
import flash.filters.BevelFilter;
import flash.geom.Rectangle;

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xCCCCCCCC);

var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, 0.8, 0x0000FF, 0.8, 20,
  20, 1, 3, "outter", false);

var filterRect:Rectangle =
  myBitmapData.generateFilterRect(myBitmapData.rectangle, filter);

trace(filterRect); // (x=-31, y=-31, w=162, h=142)
```

getColorBoundsRect (BitmapData.getColorBoundsRect メソッド)

```
public getColorBoundsRect(mask:Number, color:Number, [findColor:Boolean]) :
  Rectangle
```

ビットマップイメージ内のピクセルのうち、指定された色のすべてのピクセルを完全に含む矩形領域を決定します。

たとえば、ソースイメージがあり、0 以外のアルファチャネルを含むイメージの矩形を決定する場合には、パラメータとして {mask: 0xFF000000, color: 0x00000000} を渡します。(value & mask) != color であるピクセルの境界が、イメージ全体で検索されます。イメージの周囲に存在する空白を調べるには、{mask: 0xFFFFFFFF, color: 0xFFFFFFFF} を渡して、空白以外のピクセルの境界を検索します。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

`mask:Number` - 16 進数のカラー値。

`color:Number` - 16 進数のカラー値。

`findColor:Boolean` (オプション) - 値が `true` に設定された場合、イメージ内のカラー値の境界を返します。値が `false` に設定された場合、指定されたカラーがイメージ内に存在しない場所の境界を返します。デフォルト値は `true` です。

戻り値

`Rectangle` - 指定された色であるイメージの領域。

例

次の例では、ビットマップイメージ内の指定した色のピクセルすべてを完全に囲む矩形領域を決定する方法を示します。

```
import flash.display.BitmapData;
import flash.geom.Rectangle;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.fillRect(new Rectangle(0, 0, 50, 40), 0x00FF0000);

mc.onPress = function() {
  var colorBoundsRect:Rectangle = myBitmapData.getColorBoundsRect(0x00FFFFFF,
  0x00FF0000, true);
  trace(colorBoundsRect); // (x=0, y=0, w=50, h=40)
}
```

getPixel (BitmapData.getPixel メソッド)

```
public getPixel(x:Number, y:Number) : Number
```

BitmapData オブジェクトの特定ポイント (*x* , *y*) の RGB ピクセル値を表す整数を返します。`getPixel()` メソッドは、乗算されていないピクセル値を返します。アルファ情報は返しません。

BitmapData オブジェクト内のピクセルはすべて、乗算済みカラー値として保存されます。乗算済みイメージピクセルは、アルファデータが既に乗算された赤、緑、青の各カラーチャンネル値を保持します。たとえば、アルファ値が 0 の場合、乗算されていない値に関わらず、RGB チャンネルの値も 0 になります。

このようにデータが失われると、処理の実行時に問題が生じることがあります。Flash Player のメソッドはすべて、乗算されていない値を受け取ったり返したりします。ピクセルの内部表現は、値として返される前、乗算されていません。設定処理の際は、ピクセル値が事前に乗算されてから、生のイメージピクセルが設定されます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

x:Number - ピクセルの *x* 座標。

y:Number - ピクセルの *y* 座標。

戻り値

Number - RGB ピクセル値を表す数値。(*x*, *y*) 座標がイメージの境界外である場合は、0 を返します。

例

次の例では、getPixel() メソッドを使用して、特定の *x* 座標と *y* 座標にあるピクセルの RGB 値を取得します。

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
trace("0x" + myBitmapData.getPixel(0, 0).toString(16)); // 0xcccccc
```

関連項目

getPixel32 (BitmapData.getPixel32 メソッド)

getPixel32 (BitmapData.getPixel32 メソッド)

```
public getPixel32(x:Number, y:Number) : Number
```

アルファチャンネルデータと RGB データを含む ARGB カラー値を返します。このメソッドは getPixel() メソッドと似ていますが、getPixel() メソッドはアルファチャンネルデータがない RGB カラーを返します。

対応バージョン: ActionScript 1.0、Flash Player 8

パラメータ

x:Number - ピクセルの *x* 座標。

y:Number - ピクセルの *y* 座標。

戻り値

Number - ARGB ピクセル値を表す数値。(*x*, *y*) 座標がイメージの境界外である場合は、0 を返します。ビットマップが、透明ではなく不透明として作成された場合、このメソッドはエラーコード -1 を返します。

例

次の例では、getPixel32() メソッドを使用して、特定の *x* 座標と *y* 座標にあるピクセルの ARGB 値を取得します。

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xFFAACCEE);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

var alpha:String = (myBitmapData.getPixel32(0, 0) >> 24 & 0xFF).toString(16);
trace(">> alpha: " + alpha); // ff

var red:String = (myBitmapData.getPixel32(0, 0) >> 16 & 0xFF).toString(16);
trace(">> red: " + red); // aa

var green:String = (myBitmapData.getPixel32(0, 0) >> 8 & 0xFF).toString(16);
trace(">> green: " + green); // cc

var blue:String = (myBitmapData.getPixel32(0, 0) & 0xFF).toString(16);
trace(">> blue: " + blue); // ee

trace("0x" + alpha + red + green + blue); // 0xffaaccee
```

関連項目

getPixel (BitmapData.getPixel メソッド)

height (BitmapData.height プロパティ)

public height : Number [読み取り専用]

ビットマップイメージの高さ (ピクセル単位) です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、BitmapData インスタンスの height プロパティの設定を試みて失敗することによって、このプロパティが読み取り専用であることを確認します。

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
trace(myBitmapData.height); // 80
```

```
myBitmapData.height = 999;
trace(myBitmapData.height); // 80
```

hitTest (BitmapData.hitTest メソッド)

```
public hitTest(firstPoint:Point, firstAlphaThreshold:Number,
  secondObject:Object, [secondBitmapPoint:Point],
  [secondAlphaThreshold:Number]) : Boolean
```

1 つのビットマップイメージと、ポイント、矩形、または他のビットマップイメージとの間でピクセルレベルのヒット検出を実行します。ヒットテストを行うとき、どのオブジェクトも伸縮や回転といった変換は考慮されません。

イメージが不透明である場合、このメソッドでは完全に不透明な矩形とみなされます。透過性を考慮するピクセルレベルのヒットテストを実施する場合は、両方のイメージとも透明である必要があります。2 つの透明なイメージをテストする場合、アルファしきい値パラメータによって、アルファチャンネル値 (0 ～ 255) がいくつであれば不透明とみなすかを制御します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`firstPoint:Point` - 現在の BitmapData インスタンスのピクセル位置を定義するポイント。

`firstAlphaThreshold:Number` - このヒットテストで不透明とみなすアルファチャンネルの最大値。

`secondObject:Object` - Rectangle、Point、または BitmapData オブジェクト。

`secondBitmapPoint:Point` (オプション) - 2 番目の BitmapData オブジェクト内のピクセル位置を定義するポイント。このパラメータは、`secondObject` の値が BitmapData オブジェクトである場合にのみ使用します。

`secondAlphaThreshold:Number` (オプション) - 2 番目の BitmapData オブジェクト内で不透明であると見なされるアルファチャネルの最大値。このパラメータは、`secondObject` の値が BitmapData オブジェクトであり、両方の BitmapData オブジェクトが透明である場合にのみ使用します。

戻り値

`Boolean` - ブール値。ヒットした場合は `true` を返します。それ以外の場合は `false` を返します。

例

次の例では、BitmapData オブジェクトが MovieClip と衝突しているかどうかを調べる方法を示します。

```
import flash.display.BitmapData;
import flash.geom.Point;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc_1:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_1.attachBitmap(myBitmapData, this.getNextHighestDepth());

var mc_2:MovieClip = createRectangle(20, 20, 0xFF0000);

var destPoint:Point = new Point(myBitmapData.rectangle.x,
  myBitmapData.rectangle.y);
var currPoint:Point = new Point();

mc_1.onEnterFrame = function() {
  currPoint.x = mc_2._x;
  currPoint.y = mc_2._y;
  if(myBitmapData.hitTest(destPoint, 255, currPoint)) {
    trace(">> Collision at x:" + currPoint.x + " and y:" + currPoint.y);
  }
}

mc_2.startDrag(true);

function createRectangle(width:Number, height:Number, color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

loadBitmap (BitmapData.loadBitmap メソッド)

```
public static loadBitmap(id:String) : BitmapData
```

ライブラリ内の指定されたリンケージ識別子で識別されるシンボルのビットマップイメージ表現を含む、新しい BitmapData オブジェクトを返します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`id:String` - ライブラリ内のシンボルのリンケージ識別子。

戻り値

BitmapData - シンボルのビットマップイメージ表現。

例

次の例では、linkageId libraryBitmap が設定されたビットマップをライブラリからロードします。このビットマップを MovieClip に関連付けて、ビットマップに視覚的表現を適用します。

```
import flash.display.BitmapData;

var linkageId:String = "libraryBitmap";
var myBitmapData:BitmapData = BitmapData.loadBitmap(linkageId);
trace(myBitmapData instanceof BitmapData); // true

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
```

merge (BitmapData.merge メソッド)

```
public merge(sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point,
  redMult:Number, greenMult:Number, blueMult:Number, alphaMult:Number) : Void
```

ソースイメージとターゲットイメージをチャンネルごとにブレンドします。チャンネルごとに次の式を使用します。

```
new red dest = (red source * redMult) + (red dest * (256 - redMult) / 256;
```

redMult 値、greenMult 値、blueMult 値、および alphaMult 値は、カラーチャンネルごとに使用する乗数になります。指定できる範囲は 0 ～ 256 です。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

sourceBitmap:BitmapData - 使用する入力ビットマップイメージ。ソースイメージは、別の BitmapData オブジェクトにすることも、現在の BitmapData オブジェクトを参照することもできます。

sourceRect:Rectangle - 入力として使用するソースイメージの領域を定義する矩形。

destPoint:Point - ソース矩形の左上隅に対応するターゲットイメージ (現在の BitmapData インスタンス) 内のポイント。

redMult:Number - 赤チャンネル値と乗算する数値。

greenMult:Number - 緑チャンネル値と乗算する数値。

blueMult:Number - 青チャンネル値と乗算する数値。

alphaMult:Number - アルファ透明度の値と乗算する数値。

例

次の例では、ある BitmapData の部分と他の BitmapData を結合する方法を示します。

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var bitmapData_1:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);
var bitmapData_2:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc_1:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_1.attachBitmap(bitmapData_1, this.getNextHighestDepth());

var mc_2:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_2.attachBitmap(bitmapData_2, this.getNextHighestDepth());
mc_2._x = 101;

mc_1.onPress = function() {
  bitmapData_1.merge(bitmapData_2, new Rectangle(0, 0, 50, 40), new Point(25,
  20), 128, 0, 0, 0);
}
```

noise (BitmapData.noise メソッド)

```
public noise(randomSeed:Number, [low:Number], [high:Number],
  [channelOptions:Number], [grayScale:Boolean]) : Void
```

ランダムノイズを表すピクセルでイメージを塗ります。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

`randomSeed:Number` - 使用するランダムシード (乱数の種)。

`low:Number` (オプション) - チャンネルごとに生成する最小値 (0 ～ 255)。デフォルトは 0 です。

`high:Number` (オプション) - チャンネルごとに生成する最大値 (0 ～ 255)。デフォルトは 255 です。

`channelOptions:Number` (オプション) - 任意の 4 つのカラーチャネル値 (1 (赤)、2 (緑)、4 (青)、および 8(アルファ)) の組み合わせである数値。論理和 (OR) 演算子 (|) を使用して、チャンネル値を組み合わせることもできます。デフォルト値は (1 | 2 | 4) です。

`grayScale:Boolean` (オプション) - ブール値です。true の場合、すべてのカラーチャンネルに同じ値を設定することでグレースケールのイメージが作成されます。このパラメータを true に設定しても、アルファチャンネル選択には影響しません。デフォルト値は false です。

例

次の例では、カラーとモノクロの両方のビットマップについて、BitmapData オブジェクトにピクセルノイズを適用する方法を示します。

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var bitmapData_1:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);
var bitmapData_2:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc_1:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_1.attachBitmap(bitmapData_1, this.getNextHighestDepth());

var mc_2:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_2.attachBitmap(bitmapData_2, this.getNextHighestDepth());
mc_2._x = 101;

mc_1.onPress = function() {
     bitmapData_1.merge(bitmapData_2, new Rectangle(0, 0, 50, 40), new Point(25,
  20), 128, 0, 0, 0);
}

mc_1.onPress = function() {
     bitmapData_1.noise(128, 0, 255, 1, true);
}

mc_2.onPress = function() {
     bitmapData_2.noise(128);
}
```

paletteMap (BitmapData.paletteMap メソッド)

```
public paletteMap(sourceBitmap:BitmapData, sourceRect:Rectangle,
  destPoint:Point, [redArray:Array], [greenArray:Array], [blueArray:Array],
  [alphaArray:Array]) : Void
```

指定された 4 つのカラーパレットデータ配列 (チャンネルごとに 1 つの配列) を使用して、イメージ内のカラーチャンネル値をマッピングし直します。

Flash Player は以下の手順に従って、結果として得られるイメージを生成します。

赤、緑、青、アルファの各値を算出したら、これらの値を標準の 32 ビット整数算術演算を使用して足し合わせます。各ピクセルの赤、緑、青、アルファのチャンネル値を抽出して、別個の 0 ～ 255 の値にします。これらの値を使用して、該当する配列 redArray、greenArray、blueArray、および alphaArray 内の新しいカラー値を調べます。これら 4 つの配列にはそれぞれ 256 個の値が含まれている必要があります。新しいチャンネル値を 4 つともすべて取得した後、それらの値を組み合わせて、ピクセルに適用される標準の ARGB 値にします。

このメソッドではクロスチャンネル効果をサポートできます。値を足し合わせるとき、各入力配列は完全な 32 ビット値を含むことができます。このルーチンは、チャンネル単位のクランピングに対応していません。

チャンネルに関して配列が指定されない場合は、ソースイメージからターゲットイメージにカラーチャンネルを単にコピーします。

このメソッドは各種効果で使用できます。たとえば、通常のパレットマッピング (1 つのチャンネルを選択して疑似色イメージに変換する) などです。このメソッドは、ガンマ、曲線、平準化、量子化といった各種のカラー操作アルゴリズムにも使用できます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

sourceBitmap:BitmapData - 使用する入力ビットマップイメージ。ソースイメージは、別の BitmapData オブジェクトにすることも、現在の BitmapData オブジェクトを参照することもできます。

sourceRect:Rectangle - 入力として使用するソースイメージの領域を定義する矩形。

destPoint:Point - ソース矩形の左上隅に対応するターゲットイメージ (現在の BitmapData オブジェクト) 内のポイント。

redArray:Array (オプション) - redArray が null でない場合、red = redArray[source red value] else red = source rect value です。

greenArray:Array (オプション) - greenArray が null でない場合、green = greenArray[source green value] else green = source green value. です。

blueArray:Array (オプション) - blueArray が null でない場合、blue = blueArray[source blue value] else blue = source blue value です。

alphaArray:Array (オプション) - alphaArray が null でない場合、alpha = alphaArray[source alpha value] else alpha = source alpha value です。

例

次の例では、単一の BitmapData オブジェクトでパレットマップを使用して、赤の塗りを緑に、緑の塗りを赤に変換する方法を示します。

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
  mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

myBitmapData.fillRect(new Rectangle(51, 0, 50, 80), 0x0000FF00);

mc.onPress = function() {
  var redArray:Array = new Array(256);
  var greenArray:Array = new Array(256);

  for(var i = 0; i < 255; i++) {
    redArray[i] = 0x00000000;
    greenArray[i] = 0x00000000;
  }

  redArray[0xFF] = 0x0000FF00;
  greenArray[0xFF] = 0x00FF0000;

  myBitmapData.paletteMap(myBitmapData, new Rectangle(0, 0, 100, 40), new
  Point(0, 0), redArray, greenArray, null, null);
}
```

perlinNoise (BitmapData.perlinNoise メソッド)

```
public perlinNoise(baseX:Number, baseY:Number, numOctaves:Number,
  randomSeed:Number, stitch:Boolean, fractalNoise:Boolean,
  [channelOptions:Number], [grayScale:Boolean], [offsets:Object]) : Void
```

Perlin ノイズイメージを生成します。

Perlin ノイズ生成アルゴリズムでは、個々のランダムノイズ関数 (オクターブといいます) を補間および組み合わせることで、より自然に見えるランダムノイズを生成する単一の関数にします。音楽のオクターブと同様、各オクターブ関数の周波数は、その前のオクターブ関数の周波数の 2 倍になります。Perlin ノイズは、複数のノイズデータセットをさまざまな詳細レベルで組み合わせるので、「フラクタルノイズの和」として説明されてきました。

Perlin ノイズ関数は、木目、雲、山脈など、自然現象や風景をシミュレートする場合に使用できます。ほとんどの場合、Perlin ノイズ関数の出力はそのまま表示しないで、他のイメージを強調する場合や、擬似ランダムバリエーションを与える場合に使用します。

単純なデジタルランダムノイズ関数は、コントラストのきついポイントが含まれるイメージを生成する場合が多いです。このようにきついコントラストは自然界にはあまりありません。Perlin ノイズアルゴリズムは、さまざまな詳細レベルで実行する複数のノイズ関数を混ぜ合わせます。このアルゴリズムで、隣接するピクセルの値が大きく異なることはありません。

メモ : Perlin ノイズアルゴリズムは、Ken Perlin 氏にちなんで命名されました。Ken Perlin 氏は、1982 年の映画「トロン」のコンピュータグラフィックを生成した後に、このアルゴリズムを開発しました。Perlin 氏は 1997 年に、Perlin ノイズ関数に関する技術的功績によりアカデミー賞を受賞しています。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`baseX:Number` - *x* 方向で使用する周波数。たとえば、64 x 128 のイメージに見合うサイズのノイズを生成するには、`baseX` 値として 64 を指定します。

`baseY:Number` - *y* 方向で使用する周波数。たとえば、64 x 128 のイメージに見合うサイズのノイズを生成するには、`baseY` 値として 128 を指定します。

`numOctaves:Number` - このノイズを作成するときに組み合わせるオクターブ (つまり個々のノイズ関数) の数。オクターブ数を多くすると、よりきめ細かいイメージを作成できます。オクターブ数を増やすと、処理時間も長くなります。

`randomSeed:Number` - ランダムシード (乱数の種) として使用する数値。他のすべてのパラメータを同じままにした場合、ランダムシードの値を変更することでさまざまな疑似乱数を生成できます。Perlin ノイズ関数は、マッピング関数であり真の乱数生成関数ではありません。このため、同じランダムシードから毎回同じ結果が作成されます。

`stitch:Boolean` - ブール値。`true` の場合、このメソッドは、イメージのトランジションエッジをスムーズにして、ビットマップ塗りとしてタイリング用のシームレスなテクスチャの作成を試みます。

fractalNoise:Boolean - ブール値。true の場合、このメソッドはフラクタルノイズを生成します。それ以外の場合は、乱流を生成します。乱流があるイメージにはグラデーションに視覚的な不連続性があるので、炎や海の波のようなシャープな視覚効果に適していることがあります。

channelOptions:Number (オプション) - カラーチャネルを表す数値。この値を作成するには、4 つのカラーチャネル値 1 (赤)、2 (緑)、4 (青)、および 8 (アルファ) を単独で、または任意に組み合わせて使用できます。チャネル値を組み合わせるには、論理和 OR 演算子を使用できます。たとえば、1 | 2というコードを使用すると、赤と緑のチャネルを組み合わせることができます。

grayScale:Boolean (オプション) - ブール値です。true の場合、赤、緑、および青の各カラーチャンネルに同じ値を設定して、グレースケールイメージが作成されます。この値が true に設定されても、アルファチャンネルの値に影響はありません。デフォルト値は false です。

offsets:Object (オプション) - オクターブごとの *x* オフセットと *y* オフセットに対応するポイントからなる配列。オフセット値を操作することで、perlinNoise イメージのレイヤーをスムーズにスクロールできます。オフセット配列内の各ポイントは、特定のオクターブノイズ関数に影響を与えます。

例

次の例では、Perlin ノイズを BitmapData オブジェクトに適用する方法を示します。

```
import flash.display.BitmapData;

var bitmapData_1:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);
var bitmapData_2:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc_1:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_1.attachBitmap(bitmapData_1, this.getNextHighestDepth());

var mc_2:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_2.attachBitmap(bitmapData_2, this.getNextHighestDepth());
mc_2._x = 101;

mc_1.onPress = function() {
  var randomNum:Number = Math.floor(Math.random() * 10);
    bitmapData_1.perlinNoise(100, 80, 6, randomNum, false, true, 1, true, null);
}

mc_2.onPress = function() {
  var randomNum:Number = Math.floor(Math.random() * 10);
    bitmapData_2.perlinNoise(100, 80, 4, randomNum, false, false, 15, false,
  null);
}
```

pixelDissolve (BitmapData.pixelDissolve メソッド)

```
public pixelDissolve(sourceBitmap:BitmapData, sourceRect:Rectangle,
  destPoint:Point, [randomSeed:Number], [numberOfPixels:Number],
  [fillColor:Number]) : Number
```

ソースイメージからターゲットイメージへのピクセルディゾルブ、または同じイメージを使用することによるピクセルディゾルブを実行します。Flash Player は randomSeed 値を使用して、ランダムなピクセルディゾルブを生成します。この関数の戻り値は後続の呼び出しに渡して、終了するまでピクセルディゾルブを続ける必要があります。

ソースイメージとターゲットイメージが等しくない場合は、同じプロパティを使用して、ソースからターゲットにピクセルがコピーされます。これによって、空白イメージから完全に設定されたイメージへとディゾルブできます。

ソースイメージとターゲットイメージが等しい場合は、color パラメータを使ってピクセルが塗られます。これによって、完全に設定されたイメージを消去するようにディゾルブできます。このモードでは、ターゲット point パラメータが無視されます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

sourceBitmap:BitmapData - 使用する入力ビットマップイメージ。ソースイメージは、別の BitmapData オブジェクトにすることも、現在の BitmapData インスタンスを参照することもできます。

sourceRect:Rectangle - 入力として使用するソースイメージの領域を定義する矩形。

destPoint:Point - ソース矩形の左上隅に対応するターゲットイメージ (現在の BitmapData インスタンス) 内のポイント。

randomSeed:Number (オプション) - ピクセルディゾルブを開始するときに使用するランダムシード (乱数の種)。デフォルト値は 0 です。

numberOfPixels:Number (オプション) - デフォルトは、ソース領域 (幅 x 高さ) の 1/30 です。

fillColor:Number (オプション) - ソース値とターゲット値が等しいピクセルの塗りつぶしに使用する ARGB カラー値。デフォルト値は 0 です。

戻り値

Number - 後続の呼び出しで使用する新しいランダムシード (乱数の種)。

例

次の例では、pixelDissolve() を使用して、8000 個のピクセルすべてのカラーが変更されるまで一度に 40 個のピクセルをディゾルブすることにより、グレーの BitmapData オブジェクトを赤に変換します。

```
import flash.display.BitmapData;
import flash.geom.Point;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

mc.onPress = function() {
  var randomNum:Number = Math.floor(Math.random() * 10);
  dissolve(randomNum);
}

var intervalId:Number;
var totalDissolved:Number = 0;
var totalPixels:Number = 8000;

function dissolve(randomNum:Number) {
  var newNum:Number = myBitmapData.pixelDissolve(myBitmapData,
  myBitmapData.rectangle, new Point(0, 0), randomNum, 40, 0x00FF0000);
  clearInterval(intervalId);
  if(totalDissolved < totalPixels) {
    intervalId = setInterval(dissolve, 10, newNum);
  }
  totalDissolved += 40;
}
```

rectangle (BitmapData.rectangle プロパティ)

public rectangle : Rectangle (読み取り専用)

ビットマップイメージのサイズと位置を定義する矩形です。矩形の上端と左端が 0 になります。幅と高さは、BitmapData オブジェクトのピクセルの幅および高さと等しくなります。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、Bitmap インスタンスの rectangle プロパティの設定を試みて失敗することによって、このプロパティが読み取り専用であることを確認します。

```
import flash.display.BitmapData;
import flash.geom.Rectangle;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);
```

```
var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
trace(myBitmapData.rectangle); // (x=0, y=0, w=100, h=80)

myBitmapData.rectangle = new Rectangle(1, 2, 4, 8);
trace(myBitmapData.rectangle); // (x=0, y=0, w=100, h=80)
```

scroll (BitmapData.scroll メソッド)

```
public scroll(x:Number, y:Number) : Void
```

所定の (*x*, *y*) ピクセル量だけイメージをスクロールします。スクロール領域外のエッジ領域は変わらずにそのままになります。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

x:Number - 水平方向のスクロール量。

y:Number - 垂直方向のスクロール量。

例

次の例では、BitmapData オブジェクトのスクロール方法を示します。

```
import flash.display.BitmapData;
import flash.geom.Rectangle;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

myBitmapData.fillRect(new Rectangle(0, 0, 25, 80), 0x00FF0000);

mc.onPress = function() {
  myBitmapData.scroll(25, 0);
}
```

setPixel (BitmapData.setPixel メソッド)

```
public setPixel(x:Number, y:Number, color:Number) : Void
```

BitmapData オブジェクトの 1 つのピクセルに色を設定します。この処理中、イメージピクセルのアルファチャンネルは現在の値が保持されます。RGB カラーパラメータの値は、乗算されていないカラー値として処理されます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`x:Number` - 値を変更するピクセルの *x* 座標。

`y:Number` - 値を変更するピクセルの *y* 座標。

`color:Number` - ピクセルに設定する RGB カラー。

例

次の例では、`setPixel()` メソッドを使用して、特定の *x* 座標と *y* 座標にあるピクセルに RGB 値を割り当てます。ドラッグすることにより、作成済みのビットマップに対し 0x000000 で描画できます。

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

mc.onPress = function() {
  this.onEnterFrame = sketch;
}

mc.onRelease = function() {
  delete this.onEnterFrame;
}

function sketch() {
  myBitmapData.setPixel(_xmouse, _ymouse, 0x000000);
}
```

関連項目

getPixel (BitmapData.getPixel メソッド), setPixel32 (BitmapData.setPixel32 メソッド)

setPixel32 (BitmapData.setPixel32 メソッド)

```
public setPixel32(x:Number, y:Number, color:Number) : Void
```

BitmapData オブジェクトの 1 つのピクセルにカラー値とアルファ透明度を設定します。このメソッドは setPixel( ) メソッドと似ています。主な違いは、setPixel32( ) メソッドではアルファチャンネル情報が含まれる ARGB カラー値を引数として取ることです。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

x:Number - 値を変更するピクセルの x 座標。

y:Number - 値を変更するピクセルの y 座標。

color:Number - ピクセルに設定する ARGB カラー。不透明な (透明でない) ビットマップを作成した場合、このカラー値のアルファ透明度部分は無視されます。

例

次の例では、setPixel32( ) メソッドを使用して、特定の x 座標と y 座標にあるピクセルに ARGB 値を割り当てます。マウスボタンを押してドラッグすることにより、作成済みのビットマップをアルファ値なしの 0x000000 で描画できます。

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xFFCCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

mc.onPress = function() {
  this.onEnterFrame = sketch;
}

mc.onRelease = function() {
  delete this.onEnterFrame;
}

function sketch() {
  myBitmapData.setPixel32(_xmouse, _ymouse, 0x00000000);
}
```

関連項目

getPixel32 (BitmapData.getPixel32 メソッド), setPixel (BitmapData.setPixel メソッド)

threshold (BitmapData.threshold メソッド)

```
public threshold(sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point,
  operation:String, threshold:Number, [color:Number], [mask:Number],
  [copySource:Boolean]) : Number
```

指定されたしきい値に照らしてイメージのピクセル値をテストし、テストに合格したピクセルに新しいカラー値を設定します。threshold() メソッドを使用すると、イメージ内のカラー範囲を分離して置換し、イメージピクセルに対しその他の論理演算を実行することができます。

しきい値テストのロジックは次のとおりです。

```
if ((pixelValue & mask) operation (threshold & mask)) then

    set pixel to color

 else

    if (copySource) then

       set pixel to corresponding pixel value from sourceBitmap
```

operation パラメータには、しきい値テストで使用する比較演算子を指定します。たとえば、"==" を使用することにより、イメージ内の特定のカラー値を分離できます。または、{operation: "<", mask: 0xFF000000, threshold: 0x7F000000, color: 0x00000000} を使用することにより、ソースイメージピクセルのアルファが 0x7F 未満の場合に、すべてのターゲットピクセルが完全に透明になるように設定できます。アニメーション化されたトランジションやその他の効果に対して、この技法を使用できます。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

sourceBitmap:BitmapData - 使用する入力ビットマップイメージ。ソースイメージは、別の BitmapData オブジェクトにすることも、現在の BitmapData インスタンスを参照することもできます。

sourceRect:Rectangle - 入力として使用するソースイメージの領域を定義する矩形。

destPoint:Point - ソース矩形の左上隅に対応するターゲットイメージ (現在の BitmapData インスタンス) 内のポイント。

operation:String - 比較演算子 "<"、"<="、">"、">="、"=="、"!=" の 1 つ。ストリングとして渡されます。

threshold:Number - 各ピクセルがしきい値に収まっているか、しきい値を超えているかどうかを確認するための基準となる値。

color:Number (オプション) - しきい値テストが成功した場合にピクセルに設定されるカラー値。デフォルトは 0x00000000 です。

mask:Number (オプション) - カラーコンポーネントを分離するときに使用するマスク。デフォルト値は 0xFFFFFFFF です。

copySource:Boolean (オプション) - ブール値です。true の場合、しきい値テストが失敗したときに、ソースイメージのピクセル値がターゲットイメージにコピーされます。false の場合、しきい値テストが失敗したときにソースイメージはコピーされません。デフォルト値は false です。

戻り値

Number - 変更されたピクセル数。

例

次の例では、カラー値が特定のしきい値以上であるピクセルのカラー値を変更する方法を示します。

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

myBitmapData.fillRect(new Rectangle(0, 0, 50, 80), 0x00FF0000);

mc.onPress = function() {
  myBitmapData.threshold(myBitmapData, new Rectangle(0, 0, 100, 40), new
  Point(0, 0), "==", 0x00CCCCCC, 0x000000FF, 0x00FF0000, false);
}
```

transparent (BitmapData.transparent プロパティ)

public transparent : Boolean (読み取り専用)

ビットマップイメージがピクセル単位の透明度をサポートするかどうかを定義します。transparent パラメータに対して true を渡すことにより、BitmapData オブジェクトを作成する場合にのみ、この値を設定できます。BitmapData オブジェクトを作成した後、transparent プロパティの値が true であるかどうかを確認することにより、このオブジェクトがピクセルごとの透明度をサポートするかどうかを確認できます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、Bitmap インスタンスの transparent プロパティの設定を試みて失敗することによって、このプロパティが読み取り専用であることを確認します。

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
trace(myBitmapData.transparent); // false

myBitmapData.transparent = true;
trace(myBitmapData.transparent); // false
```

width (BitmapData.width プロパティ)

public width : Number (読み取り専用)

ビットマップイメージの幅 (ピクセル単位) です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、Bitmap インスタンスの width プロパティの設定を試みて失敗することによって、このプロパティが読み取り専用であることを確認します。

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
trace(myBitmapData.width); // 100

myBitmapData.width = 999;
trace(myBitmapData.width); // 100
```

BitmapFilter (flash.filters.BitmapFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
```

```
public class BitmapFilter
extends Object
```

すべてのイメージフィルタ効果用の BitmapFilter 基本クラスです。

BevelFilter、BlurFilter、ColorMatrixFilter、ConvolutionFilter、DisplacementMapFilter、DropShadowFilter、GlowFilter、GradientBevelFilter、および GradientGlowFilter クラスはすべて、BitmapFilter クラスを継承します。このフィルタ効果は、ビットマップや MovieClip のインスタンスに適用できます。

BitmapFilter クラスの先行するサブクラスに対してのみサブクラスを作成できます。

対応バージョン： ActionScript 1.0、Flash Player 8

プロパティ一覧

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

メソッド一覧

オプション	署名	説明
	clone() : BitmapFilter	元の BitmapFilter オブジェクトとまったく同じコピーである BitmapFilter オブジェクトを返します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

clone (BitmapFilter.clone メソッド)

```
public clone() : BitmapFilter
```

元の BitmapFilter オブジェクトとまったく同じコピーである BitmapFilter オブジェクトを返します。

対応バージョン： ActionScript 1.0、Flash Player 8

戻り値

`BitmapFilter` - BitmapFilter オブジェクト。

BlurFilter (flash.filters.BlurFilter)

```
Object
  |
  +-BitmapFilter
     |
     +-flash.filters.BlurFilter
```

```
public class BlurFilter
extends BitmapFilter
```

BlurFilter クラスを使用すると、Flash の各種オブジェクトにぼかし視覚効果を適用できます。ぼかし効果は、イメージの細部をぼかします。ソフトフォーカスがかかっているように見えるぼかしから、半透明ガラスを通してイメージを見るようにかすんで見えるガウスぼかしまで作成できます。このフィルタの `quality` プロパティを 1 に設定すると、ソフトフォーカスがかかっているように見えるぼかしになります。`quality` プロパティを 3 に設定すると、ガウスぼかしフィルタに近似したものになります。

フィルタの使い方は、フィルタの適用先オブジェクトによって異なります。

- 実行時にムービークリップ、テキストフィールド、ボタンにフィルタを適用する場合は、`filters` プロパティを使用します。オブジェクトの `filters` プロパティを設定しても、そのオブジェクトは変更されません。`filters` プロパティをクリアすれば、設定を取り消すことができます。
- BitmapData インスタンスにフィルタを適用するには、`BitmapData.applyFilter()` メソッドを使用します。BitmapData オブジェクトで `applyFilter` を呼び出すことによって、ソース BitmapData オブジェクトとフィルタオブジェクトが取得され、フィルタを適用した結果として得られるイメージが生成されます。

イメージやビデオにもオーサリング時にフィルタ効果を適用できます。詳細については、オーサリングのマニュアルを参照してください。

ムービークリップやボタンにフィルタを適用する場合は、ムービークリップやボタンの cacheAsBitmap プロパティを true に設定します。すべてのフィルタをクリアすると、cacheAsBitmap は元の値に戻ります。

このフィルタはステージの拡大・縮小に対応していますが、通常の拡大・縮小、回転、傾斜には対応していません。オブジェクト自体が拡大・縮小される場合 (_xscale と _yscale が 100% ではない場合)、フィルタ効果は拡大・縮小されません。拡大・縮小されるのは、ステージをズームインした場合のみです。

結果として得られるイメージの幅または高さが 2880 ピクセルを超える場合、フィルタは適用されません。たとえば、フィルタが適用されたサイズの大きいムービークリップをズームインするとき、結果として得られるイメージが 2880 ピクセルの制限を超える場合は、フィルタがオフになります。

対応バージョン： ActionScript 1.0、Flash Player 8

関連項目

filters (MovieClip.filters プロパティ), cacheAsBitmap (MovieClip.cacheAsBitmap プロパティ), filters (Button.filters プロパティ), cacheAsBitmap (Button.cacheAsBitmap プロパティ), filters (TextField.filters プロパティ), applyFilter (BitmapData.applyFilter メソッド)

プロパティ一覧

オプション	プロパティ	説明
	blurX:Number	水平方向のぼかし量です。
	blurY:Number	垂直方向のぼかし量です。
	quality:Number	ぼかしの実行回数です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター一覧

署名	説明
BlurFilter([blurX:Number], [blurY:Number], [quality:Number])	指定されたパラメータでフィルタを初期化します。

メソッド一覧

オプション	署名	説明
	clone() : BlurFilter	このフィルタオブジェクトのコピーを返します。

BitmapFilter クラスから継承されるメソッド

clone (BitmapFilter.clone メソッド)

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

BlurFilter コンストラクタ

```
public BlurFilter([blurX:Number], [blurY:Number], [quality:Number])
```

指定されたパラメータでフィルタを初期化します。デフォルト値では、ソフトフォーカスのかかったイメージが作成されます。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

blurX:Number (オプション) - 水平方向のぼかし量です。指定できる値は 0 ～ 255 (浮動小数値) です。デフォルト値は 4 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

blurY:Number (オプション) - 垂直方向のぼかし量です。指定できる値は 0 ～ 255 (浮動小数値) です。デフォルト値は 4 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

quality:Number (オプション) - フィルタを適用する回数。デフォルト値は 1 (低品質) です。値 2 は普通の品質です。さらに、値 3 は高い品質で、ガウスぼかしに近似したものになります。

例

次の例では、新しい BlurFilter コンストラクタをインスタンス化して、それを平らな矩形シェイプに適用します。

```
import flash.filters.BlurFilter;
var rect:MovieClip = createRectangle(100, 100, 0x003366, "BlurFilterExample");

var blurX:Number = 30;
var blurY:Number = 30;
var quality:Number = 3;

var filter:BlurFilter = new BlurFilter(blurX, blurY, quality);
var filterArray:Array = new Array();
filterArray.push(filter);
rect.filters = filterArray;

function createRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  mc.beginFill(bgColor);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc._x = 20;
  mc._y = 20;
  return mc;
}
```

blurX (BlurFilter.blurX プロパティ)

```
public blurX : Number
```

水平方向のぼかし量。指定できる値は 0 ～ 255 (浮動小数) です。デフォルト値は 4 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip インスタンスがクリックされたときに、その blurX プロパティを変更します。

```
import flash.filters.BlurFilter;
var mc:MovieClip = createBlurFilterRectangle("BlurFilterBlurX");
mc.onRelease = function() {
   var filter:BlurFilter = this.filters[0];
   filter.blurX = 200;
   this.filters = new Array(filter);
}

function createBlurFilterRectangle(name:String):MovieClip {
   var rect:MovieClip = this.createEmptyMovieClip(name,
   this.getNextHighestDepth());
   var w:Number = 100;
   var h:Number = 100;
   rect.beginFill(0x003366);
   rect.lineTo(w, 0);
   rect.lineTo(w, h);
   rect.lineTo(0, h);
   rect.lineTo(0, 0);
   rect._x = 20;
   rect._y = 20;

   var filter:BlurFilter = new BlurFilter(30, 30, 2);
   var filterArray:Array = new Array();
   filterArray.push(filter);
   rect.filters = filterArray;
   return rect;
}
```

blurY (BlurFilter.blurY プロパティ)

```
public blurY : Number
```

垂直方向のぼかし量。指定できる値は 0 ～ 255 (浮動小数) です。デフォルト値は 4 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip インスタンスがクリックされたときに、その blurY プロパティを変更します。

```
import flash.filters.BlurFilter;
var mc:MovieClip = createBlurFilterRectangle("BlurFilterBlurY");
mc.onRelease = function() {
   var filter:BlurFilter = this.filters[0];
   filter.blurY = 200;
   this.filters = new Array(filter);
}

function createBlurFilterRectangle(name:String):MovieClip {
   var rect:MovieClip = this.createEmptyMovieClip(name,
   this.getNextHighestDepth());
   var w:Number = 100;
   var h:Number = 100;
   rect.beginFill(0x003366);
   rect.lineTo(w, 0);
   rect.lineTo(w, h);
   rect.lineTo(0, h);
   rect.lineTo(0, 0);
   rect._x = 20;
   rect._y = 20;

   var filter:BlurFilter = new BlurFilter(30, 30, 2);
   var filterArray:Array = new Array();
   filterArray.push(filter);
   rect.filters = filterArray;
   return rect;
}
```

clone (BlurFilter.clone メソッド)

```
public clone() : BlurFilter
```

このフィルタオブジェクトのコピーを返します。

対応バージョン： ActionScript 1.0、Flash Player 8

戻り値

`BlurFilter` - 元の BlurFilter インスタンスとプロパティがすべて同じである新しい BlurFilter インスタンス。

例

次の例では、3 つの BlurFilter オブジェクトを作成し、それらを比較します。`BlurFilter` コンストラクタを使用することにより、`filter_1` オブジェクトを作成します。`filter_1` と等しくなるよう `filter_2` オブジェクトを作成します。`filter_1` のクローンを作成することによって `clonedFilter` オブジェクトを作成できます。`filter_2` が `filter_1` と等しいと評価される場合に、たとえ `filter_1` と同じ値を含んでいても `clonedFilter` は等しいと評価されないことに注意してください。

```
import flash.filters.BlurFilter;

var filter_1:BlurFilter = new BlurFilter(30, 30, 2);
var filter_2:BlurFilter = filter_1;
var clonedFilter:BlurFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false

for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> quality: 2
  // >> blurY: 30
  // >> blurX: 30
}

for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
  // >> quality: 2
  // >> blurY: 30
  // >> blurX: 30
}
```

filter_1、filter_2、および clonedFilter の関係をさらに詳しく示すために、次の例では、filter_1 の quality プロパティを変更します。quality を変更すると、clone() メソッドが、filter_1 の値を参照する代わりに、この値に基づいて新しいインスタンスが作成されます。

```
import flash.filters.BlurFilter;

var filter_1:BlurFilter = new BlurFilter(30, 30, 2);
var filter_2:BlurFilter = filter_1;
var clonedFilter:BlurFilter = filter_1.clone();

trace(filter_1.quality); // 2
trace(filter_2.quality); // 2
trace(clonedFilter.quality); // 2

filter_1.quality = 1;

trace(filter_1.quality); // 1
trace(filter_2.quality); // 1
trace(clonedFilter.quality); // 2
```

quality (BlurFilter.quality プロパティ)

```
public quality : Number
```

ぼかしの実行回数です。有効な値は 0 ～ 15 です。デフォルト値は 1 で、低品質と等価です。値 2 は普通の品質です。値 3 は高品質であり、ガウスぼかしに近似したものになります。

多くのアプリケーションでは、quality 値 1、2、または 3 で十分です。さらに、15 までの数値を使用すれば、ぼかしの適用回数を増やしてぼかし効果を増大できますが、値が大きいほどレンダリングが遅くなることに注意してください。多くの場合、quality の値を大きくする代わりに blurX と blurY の値を大きくするだけで、同様の効果が得られます。この方法を実行すると、より高速にレンダリングされます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、矩形を作成して、quality 値が 1 のぼかしフィルタを適用します。矩形をクリックすると、quality は 3 に増分され、矩形のぼかしが濃くなります。

```
import flash.filters.BlurFilter;
var mc:MovieClip = createBlurFilterRectangle("BlurFilterQuality");
mc.onRelease = function() {
  var filter:BlurFilter = this.filters[0];
  filter.quality = 3;
  this.filters = new Array(filter);
}
```

```
function createBlurFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BlurFilter = new BlurFilter(30, 30, 1);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

Boolean

```
Object
  |
  +-Boolean
```

```
public class Boolean
extends Object
```

Boolean クラスは、標準 JavaScript の Boolean オブジェクトと同じ機能を持つラッパーオブジェクトです。Boolean クラスを使用して、Boolean オブジェクトのプリミティブなデータ型またはストリング表現を調べることができます。

Boolean オブジェクトのメソッドを呼び出すには、コンストラクタの新しい `Boolean()` を使用してこのオブジェクトを作成する必要があります。

対応バージョン： ActionScript 1.0、Flash Player 5 - Flash Player 6 ではネイティブオブジェクトになり、パフォーマンスが大幅に向上しました。

プロパティー覧

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター一覧

署名	説明
Boolean([value:Object])	Boolean オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	toString() : String	Boolean オブジェクトのストリング表現 ("true" または "false") を返します。
	valueOf() : Boolean	指定された Boolean オブジェクトのプリミティブな値のタイプが true の場合は true を、それ以外の場合は false を返します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

Boolean コンストラクタ

public Boolean([value:Object])

Boolean オブジェクトを作成します。value パラメータを省略すると、Boolean オブジェクトは値 false で初期化されます。value パラメータの値を指定すると、メソッドによって評価され、評価結果はグローバル Boolean() 関数の規則に従ってブール値として返されます。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

value:Object (オプション) - 任意の式。デフォルト値は false です。

例

次のコードでは、新しい空の Boolean オブジェクトとして myBoolean を作成します。

```
var myBoolean:Boolean = new Boolean();
```

toString (Boolean.toString メソッド)

```
public toString() : String
```

Boolean オブジェクトのストリング表現 ("true" または "false") を返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

String - ストリング。"true" または "false"。

例

この例では、Boolean 型の変数を作成した後、ストリングからなる配列で使用するために、toString() を使って値をストリングに変換します。

```
var myStringArray:Array = new Array("yes", "could be");
var myBool:Boolean = 0;
myBool.toString();
myStringArray.push(myBool);
```

valueOf (Boolean.valueOf メソッド)

```
public valueOf() : Boolean
```

指定された Boolean オブジェクトのプリミティブな値のタイプが true の場合は true を、それ以外の場合は false を返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Boolean - ブール値。

例

次の例では、このメソッドがどのように動作するか、および新しい Boolean オブジェクトのプリミティブな値のタイプが false であることを示します。

```
var x:Boolean = new Boolean();
trace(x.valueOf());   // false
x = (6==3+3);
trace(x.valueOf());   // true
```

Button

```
Object
  |
  +-Button
```

```
public class Button
extends Object
```

SWF ファイル内のすべてのボタンシンボルは、Button オブジェクトのインスタンスです。プロパティインスペクタを使用して、ボタンにインスタンス名を付けることができます。また Button クラスのメソッドとプロパティを使用して、ActionScript でボタンを操作できます。ボタンのインスタンス名は、ムービーエクスプローラに表示されます。[アクション] パネルの [ターゲットパスの挿入] ダイアログボックスにも表示されます。

Button クラスは Object クラスから継承します。

対応バージョン： ActionScript 1.0、Flash Player 6

関連項目

Object

プロパティ一覧

オプション	プロパティ	説明
	_alpha:Number	my_btn で指定されているボタンのアルファ透明度の値です。
	blendMode:Object	ボタンのブレンドモードです。
	cacheAsBitmap: Boolean	true に設定されていると、ボタンのビットマップ内部表現がキャッシュされます。
	enabled:Boolean	ボタンが有効であるか無効であるかを指定するブール値です。
	filters:Array	ボタンに関連付けられている各フィルタオブジェクトを格納するインデックス付き配列です。
	_focusrect:Boolean	キーボードフォーカスがあるボタンの周囲に黄色の矩形を表示するかどうかを指定するブール値です。
	_height:Number	ボタンの高さ (ピクセル単位) です。
	_highquality:Number	**非推奨** Flash Player 7 以降では使用しないでください。このプロパティの代わりに Button._quality を使用します。現在の SWF ファイルに適用されるアンチエイリアスのレベルを指定します。
	menu:ContextMenu	ContextMenu オブジェクト contextMenu を Button オブジェクト my_button に関連付けます。

オプション	プロパティ	説明
	_name:String	my_btn で指定されているボタンのインスタンス名です。
	_parent:MovieClip	現在のムービークリップまたはオブジェクトを含むムービークリップまたはオブジェクトを指す参照です。
	_quality:String	プロパティ (グローバル)。SWF ファイルに使用するレンダリング品質を設定または取得します。
	_rotation:Number	ボタンの元の位置からの回転角 (度単位) です。
	scale9Grid:Rectangle	ボタンの 9 つの拡大・縮小領域を定義する矩形領域です。
	_soundbuftime:Number	サウンドのストリーミングを開始するまでにサウンドをプリバッファする秒数を指定するプロパティです。
	tabEnabled:Boolean	my_btn が自動タブ順に含まれているかどうかを指定します。
	tabIndex:Number	SWF ファイル内のオブジェクトのタブ順をカスタマイズできます。
	_target:String (読み取り専用)	my_btn で指定されているボタンインスタンスのターゲットパスを返します。
	trackAsMenu:Boolean	他のボタンまたはムービークリップがマウスの解放イベントを受け取ることができるかどうかを示すブール値です。
	_url:String (読み取り専用)	ボタンを作成した SWF ファイルの URL を取得します。
	useHandCursor: Boolean	マウスがボタン上に移動したときに、指差しハンドポインタ (ハンドカーソル) を表示するかどうかを示すブール値。デフォルト値は true です。
	_visible:Boolean	my_btn で指定されているボタンを表示するかどうかを示すブール値です。
	_width:Number	ボタンの幅 (ピクセル単位) です。
	_x:Number	親ムービークリップのローカル座標を基準にしたボタンの x 座標を設定する整数です。
	_xmouse:Number (読み取り専用)	ボタンを基準にした相対的なマウス位置の *x* 座標を返します。
	_xscale:Number	ボタンの基準点から適用した場合のボタンの水平方向の拡大・縮小率 (パーセント単位) です。
	_y:Number	親ムービークリップのローカル座標を基準にしたボタンの *y* 座標です。

オプション	プロパティ	説明
	_ymouse:Number (読み取り専用)	ボタンを基準にした相対的なマウス位置の y 座標を示します。
	_yscale:Number	ボタンの基準点から適用した場合のボタンの垂直方向の拡大・縮小率 (パーセント単位) です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベントの一覧

イベント	説明
onDragOut = function() {}	マウスボタンをボタン上でクリックした後、ポインタをボタンの外側にドラッグすると、呼び出されます。
onDragOver = function() {}	マウスボタンを押したまま、一度ボタンの外側に移動し、次にボタン上に戻ると、呼び出されます。
onKeyDown = function() {}	ボタンにキーボードフォーカスがあるときにキーを押すと、呼び出されます。
onKeyUp = function() {}	ボタンにフォーカスがあるときにキーを離すと、呼び出されます。
onKillFocus = function(newFocus: Object) {}	ボタンがキーボードフォーカスを失うと、呼び出されます。
onPress = function() {}	ボタンを押すと呼び出されます。
onRelease = function() {}	ボタンを離すと呼び出されます。
onReleaseOutside = function() {}	ポインタがボタン内にあるときにボタンを押した後で、ポインタがボタン外にあるときにマウスを離すと、呼び出されます。
onRollOut = function() {}	ポインタがボタン領域の外側に移動すると、呼び出されます。
onRollOver = function() {}	ポインタがボタン領域の上に移動すると、呼び出されます。
onSetFocus = function(oldFocus: Object) {}	ボタンがキーボードフォーカスを受け取ると、呼び出されます。

メソッド一覧

オプション	署名	説明
	getDepth() : Number	ボタンインスタンスの深度を返します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

_alpha (Button._alpha プロパティ)

public _alpha : Number

my_btn で指定されているボタンのアルファ透明度の値です。有効な値は 0 (完全な透明) ～ 100 (完全な不透明) です。デフォルト値は 100 です。_alpha が 0 に設定されているボタン内のオブジェクトは表示されませんが、その状態でもアクティブです。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次のコードは、ユーザーがボタンをクリックすると、ボタン myBtn_btn の _alpha プロパティを 50% に設定します。まず、Button インスタンスをステージに追加します。次に、インスタンスに myBtn_btn という名前を付けます。最後に、フレーム 1 を選択し、次のコードを [アクション] パネル内に置きます。

```
myBtn_btn.onRelease = function(){
  this._alpha = 50;
};
```

関連項目

_alpha (MovieClip._alpha プロパティ), _alpha (TextField._alpha プロパティ)

blendMode (Button.blendMode プロパティ)

```
public blendMode : Object
```

ボタンのブレンドモードです。ブレンドモードは、画面上の別のオブジェクトの上のレイヤー内でのボタンの外観に影響します。

Flash Player は blendMode プロパティをボタンの各ピクセルに適用します。各ピクセルは、3 つの要素カラー (赤、緑、青) で構成されており、各要素カラーは 0x00 ～ 0xFF の値を持ちます。Flash Player は、ボタンのあるピクセルの各要素カラーと背景のピクセルの対応するカラーとを比較します。たとえば、blendMode が "lighten" に設定されている場合、Flash Player はボタンの赤の値と背景の赤の値とを比較して明るい方を表示色の赤の成分として使用します。

次の表で、blendMode の設定について説明します。blendMode プロパティを設定するには、1 ～ 14 の整数またはストリングを使用できます。表の中の図は、ボタン (2) が画面上の別のオブジェクト (1) に重なったときにボタン (2) に適用される blendMode を示しています。

整数値	ストリング値	図	説明
1	"normal"		ボタンは、背景の前に表示されます。ボタンのピクセル値により、背景のピクセル値が無効になります。ボタンが透明な部分では、背景が表示されます。
2	"layer"		ボタンを事前構成するための一時バッファの作成を強制します。この処理は、ボタン内に子のオブジェクトが複数存在し、子の blendMode が "normal" 以外に設定されている場合に自動的に行われます。
3	"multiply"		ボタンの要素カラーの値と背景色の値とを乗算した後、0xFF で除算して正規化して、色を暗くします。これは、シャドウや深度効果によく使用されます。 たとえば、ボタン内のあるピクセルの要素カラー (たとえば赤) と背景のピクセルの対応するカラーの値がともに 0x88 である場合、乗算した結果は 0x4840 です。0xFF で除算すると、その要素カラーの値が 0x48 となり、ボタンや背景よりも濃い色になります。

整数値	ストリング値	図	説明
4	"screen"		ボタンの色の補数 (逆) と背景色の補数を乗算して、ブリーチ効果を得ます。この設定は、ハイライトや、ボタンの黒い部分の削除によく使用されます。
5	"lighten"		ボタンの要素カラーと背景の要素カラーのうち明るい方 (値が大きい方) を選択します。この設定は、重ね合わせタイプによく使用されます。 たとえば、ボタンのピクセルの RGB 値が 0xFFCC33 で、背景のピクセルの RGB 値が 0xDDF800 の場合、表示されるピクセルの RGB 値は、0xFF > 0xDD、0xCC < 0xF8、および 0x33 > 0x00 のそれぞれ大きい方が採用され、0xFFF833 になります。
6	"darken"		ボタンの要素カラーと背景の要素カラーのうち暗い方 (値が小さい方) を選択します。この設定は、重ね合わせタイプによく使用されます。 たとえば、ボタンのピクセルの RGB 値が 0xFFCC33 で、背景のピクセルの RGB 値が 0xDDF800 の場合、表示されるピクセルの RGB 値は、0xFF > 0xDD、0xCC < 0xF8、および 0x33 > 0x00 のそれぞれの小さい方が採用され、0xDDCC00 になります。
7	"difference"		ボタンの要素カラーと背景の要素カラーを比較し、2 つの要素カラーのうち明るい方の値から暗い方の値を差し引きます。この設定は、重ね合わせタイプによく使用されます。 たとえば、ボタンのピクセルの RGB 値が 0xFFCC33 で、背景のピクセルの RGB 値が 0xDDF800 の場合、0xFF - 0xDD = 0x22、0xF8 - 0xCC = 0x2C、0x33 - 0x00 = 0x33 のため、表示されるピクセルの RGB 値は 0x222C33 になります。

整数値	ストリング値	図	説明
8	"add"		ボタンの要素カラーの値を背景の要素カラーに加算し、上限 0xFF を適用します。この設定は、2 つのオブジェクト間で色を明るくするディゾルブをアニメーションにするときによく使用されます。 たとえば、ボタンのピクセルの RGB 値が 0xAAA633 で、背景のピクセルの RGB 値が 0xDD2200 の場合、0xAA + 0xDD › 0xFF、0xA6 + 0x22 = 0xC8、0x33 + 0x00 = 0x33 のため、表示されるピクセルの RGB 値は 0xFFC833 になります。
9	"subtract"		結果の下限を 0 として、ボタンの要素カラーの値をその背景の要素カラーの値から減算します。この設定は、2 つのオブジェクト間で色を暗くするディゾルブをアニメーションにするときによく使用されます。 たとえば、ボタンのピクセルの RGB 値が 0xAA2233 で、背景のピクセルの RGB 値が 0xDDA600 の場合、0xDD - 0xAA = 0x33、0xA6 - 0x22 = 0x84、0x00 - 0x33 ‹ 0x00 のため、表示されるピクセルの RGB 値は 0x338400 になります。
10	"invert"		背景を反転します。
11	"alpha"		ボタンの各ピクセルのアルファ値を背景に適用します。そのためには、"layer" blendMode を親ボタンに適用する必要があります。たとえば、図の親ボタン (白い背景) は、blendMode = "layer" に設定されています。
12	"erase"		ボタンのアルファ値に基づいて背景を消去します。そのためには、"layer" blendMode 設定を親ボタンに適用する必要があります。たとえば、図の親ボタン (白い背景) は、blendMode = "layer" に設定されています。

整数値	ストリング値	図	説明
13	"overlay"		背景の暗さに基づいて、各ビットマップのカラーを調整します。背景が 50% グレーよりも明るい場合、ボタンと背景の色が網がけされ、より明るくなります。背景が 50% グレーよりも暗い場合、2 つの色が乗算されて、より暗くなります。この設定は、シャドウ効果によく使用されます。
14	"hardlight"		ボタンの暗さに基づいて、各ビットマップのカラーを調整します。ボタンが 50% のグレーよりも暗い場合、ボタンと背景のカラーが網がけされ、明るくなります。ボタンが 50% グレーよりも暗い場合、2 つの色が乗算されて、より暗くなります。この設定は、シャドウ効果によく使用されます。

blendMode プロパティを他の値に設定しようとすると、プロパティは "normal" に設定されます。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、このプロパティを整数に設定すると、その値が対応するストリングに直ちに変換されることがわかります。

```
my_button.blendMode = 8;
trace (my_button.blendMode) // add
```

関連する例については、MovieClip クラスの blendMode プロパティの説明を参照してください。

関連項目

blendMode (MovieClip.blendMode プロパティ)

cacheAsBitmap (Button.cacheAsBitmap プロパティ)

```
public cacheAsBitmap : Boolean
```

true に設定されていると、ボタンのビットマップ内部表現がキャッシュされます。これにより、複雑なベクターコンテンツが含まれるボタンのパフォーマンスを向上できます。

ボタンの cacheAsBitmap が true に設定されている場合、ボタンの 4 つの状態のそれぞれのビットマップ表現が保存されます。

ビットマップがキャッシュされているボタンのすべてのベクターデータは、メインステージでなくビットマップに描画されます。その後、ビットマップは、最も近いピクセル境界に吸着された非伸縮、非回転のピクセルとして、メインステージにコピーされます。ピクセルは、親オブジェクトと 1 対 1 でマップされます。ビットマップの境界が変更されると、ビットマップは伸縮されずに再作成されます。

内部ビットマップが作成されるのは、cacheAsBitmap プロパティが true に設定されている場合だけです。

ボタンの cacheAsBitmap プロパティを true に設定した場合、レンダリングはそのままで、ボタンでピクセルへの吸着が自動的に実行されます。アニメーションの速度は、ベクターコンテンツの複雑さに応じて大幅に速くなる可能性があります。

ボタンにフィルタを適用するとき、そのフィルタ配列が空でなければ、cacheAsBitmap プロパティが常に true に設定されます。また、ボタンにフィルタが適用されている場合、そのボタンの cacheAsBitmap プロパティを false に設定しても、その値は true と報告されます。ボタンのすべてのフィルタをクリアすると、cacheAsBitmap の設定が直前の設定に戻ります。

以下の場合は、cacheAsBitmap プロパティが true に設定されていても、ビットマップは使用されず、ボタンがベクターデータからレンダリングされます。

- ビットマップが大きすぎる場合、つまり、幅または高さが 2880 ピクセルより大きい場合
- メモリ不足によりビットマップにメモリを割り当てることができない場合

cacheAsBitmap プロパティは、その内容がほぼ静的で、拡大や縮小、回転が頻繁に行われないボタンに最適です。そのようなボタンでは、cacheAsBitmap プロパティにより、ボタンの変換時 (その x 位置と y 位置の変更時) にパフォーマンスが向上します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、myButton という名前の既存の Button インスタンスにドロップシャドウを適用します。その後、フィルタの適用時に true に設定される cacheAsBitmap の値をトレースします。

```
import flash.filters.DropShadowFilter;
trace(myButton.cacheAsBitmap); // false
var dropShadow:DropShadowFilter = new DropShadowFilter(6, 45, 0x000000, 50, 5,
  5, 1, 2, false, false, false);
myButton.filters = new Array(dropShadow);
trace(myButton.cacheAsBitmap); // true
```

enabled (Button.enabled プロパティ)

```
public enabled : Boolean
```

ボタンが有効であるか無効であるかを指定するブール値です。ボタンが無効になっているとき (enabled プロパティが false に設定されているとき)、ボタンは表示されますが、クリックできません。デフォルト値は true です。このプロパティは、ナビゲーションの一部を無効にする場合に便利です。たとえば、現在表示されているページのボタンを無効にしてクリックできないようにすることで、ページのリロードを禁止できます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ボタンのクリックを無効および有効にする方法を示します。myBtn1_btn と myBtn2_btn, の 2 つのボタンがステージ上にあります。次のような ActionScript を追加して、myBtn2_btn ボタンをクリックできないようにします。最初に、2 つのボタンインスタンスをステージに追加します。次に、それぞれのインスタンスに myBtn1_btn および myBtn2_btn という名前を付けます。最後に、フレーム 1 に次のコードを追加して、ボタンのクリックを有効または無効にします。

```
myBtn1_btn.enabled = true;
myBtn2_btn.enabled = false;

//button code
// the following function will not get called
// because myBtn2_btn.enabled was set to false
myBtn1_btn.onRelease = function() {
  trace( "you clicked : " + this._name );
};
myBtn2_btn.onRelease = function() {
  trace( "you clicked : " + this._name );
};
```

filters (Button.filters プロパティ)

```
public filters : Array
```

ボタンに関連付けられている各フィルタオブジェクトを格納するインデックス付き配列です。flash.filters パッケージには、使用できる特定のフィルタを定義する複数のクラスが含まれています。

フィルタは、ActionScript コードを使用して、設計時または実行時に Flash オーサリングツールで適用できます。ActionScript を使用してフィルタを適用するには、Button.filters 配列全体の一時コピーを作成してその一時配列を変更した後、一時配列の値を Button.filters 配列に代入し直す必要があります。新しいフィルタオブジェクトを Button.filters 配列に直接追加することはできません。次のコードは、myButton という名前のターゲットのボタンに影響を与えません。

```
myButton.filters[0].push(myDropShadow);
```

ActionScript を使用してフィルタを追加するには、以下の手順に従う必要があります。このとき、ターゲットのボタンの名前は myButton とします。

- 選択したフィルタクラスのコンストラクタ関数を使用して、新しいフィルタオブジェクトを作成します。
- myButton.filters 配列の値を、myFilters などの名前の一時配列に割り当てます。
- 新しいフィルタオブジェクトを一時配列 myFilters に追加します。
- 一時配列の値を myButton.filters 配列に代入します。

filters 配列が空の場合、一時配列を使用する必要はありません。代わりに、作成したフィルタオブジェクトを格納する配列リテラルを直接代入することができます。

設計時または実行時に作成されたかどうかに関わらず、既存のフィルタオブジェクトを変更するには、以下の方法で filters 配列のコピーを変更する必要があります。

- myButton.filters 配列の値を、myFilters などの名前の一時配列に割り当てます。
- 一時配列 myFilters を使用してプロパティを変更します。たとえば、配列内の最初のフィルタの quality プロパティを設定するには myList[0].quality = 1; というコードを使用できます。
- 一時配列の値を myButton.filters 配列に代入します。

ボタンのフィルタをクリアするには、filters を空の配列 ([]) に設定します。

フィルタが関連付けられているボタンは、ロード時に透明なビットマップとしてボタン自身をキャッシュするようにマークされます。これ以降、ボタンに有効なフィルタリストがある限り、ボタンはビットマップとしてキャッシュされます。このビットマップは、フィルタ効果のソースイメージとして使用されます。通常、各ボタンにはビットマップのセットが 2 つずつあります。1 つはフィルタが適用されていない元のソース、もう 1 つはフィルタ適用後の最終イメージ (ボタンの 4 つの状態のそれぞれのイメージ) です。最終イメージはレンダリング時に使用されます。ボタンが変更されていない限り、最終イメージを更新する必要はありません。

複数のフィルタを格納する filters 配列を使用していて、各配列インデックスに割り当てられているフィルタの種類を追跡する必要がある場合、独自の filters 配列を維持し、別のデータ構造を使用して、各配列インデックスに関連付けられているフィルタの種類を追跡できます。filters 配列の各インデックスに関連付けられているフィルタの種類を簡単に判別する方法はありません。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、ボタン myButton にドロップシャドウフィルタを追加します。

```
import flash.filters.DropShadowFilter;
var myDropFilter:DropShadowFilter = new DropShadowFilter(6, 45, 0x000000, 50, 5,
  5, 1, 2, false, false, false);
var myFilters:Array = myButton.filters;
myFilters.push(myDropFilter);
myButton.filters = myFilters;
```

次の例では、配列内の最初のフィルタの quality 設定を 15 に変更します。この例は、myButton テキストフィールドに少なくとも 1 つのフィルタオブジェクトが関連付けられている場合にのみ機能します。

```
var myList:Array = myButton.filters;
myList[0].quality = 15;
myButton.filters = myList;
```

関連項目

cacheAsBitmap (Button.cacheAsBitmap プロパティ)

_focusrect (Button._focusrect プロパティ)

```
public _focusrect : Boolean
```

キーボードフォーカスがあるボタンの周囲に黄色の矩形を表示するかどうかを指定するブール値です。このプロパティでグローバル _focusrect プロパティの設定を上書きできます。デフォルトでは、ボタンインスタンスの _focusrect プロパティは null です。つまり、ボタンインスタンスはグローバル _focusrect プロパティを上書きしません。あるボタンインスタンスの _focusrect プロパティを true または false に設定した場合、そのボタンインスタンスのグローバル _focusrect プロパティの設定は上書きされます。

Flash Player 4 または Flash Player 5 の SWF ファイルでは、_focusrect プロパティはグローバル _focusrect プロパティを制御します。これはブール値です。Flash Player 6 以降では、この動作は、個々のムービークリップで _focusrect プロパティをカスタマイズできるように変更されています。

_focusrect プロパティが false である場合、そのボタンのキーボードナビゲーションは Tab キーに限定されます。Enter キーや矢印キーを含め、他のキーはすべて無視されます。すべてのキーナビゲーションを元に戻すには、_focusrect を true に設定する必要があります。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ブラウザウィンドウでフォーカスを受け取ったときに、SWF ファイル内の指定されたボタンインスタンスの周囲の黄色の矩形を非表示にします。3 つのボタン myBtn1_btn、myBtn2_btn、myBtn3_btn を作成し、タイムラインのフレーム 1 に次のような ActionScript を追加します。

```
myBtn2_btn._focusrect = false;
```

パブリッシュ設定を Flash Player 6 に変更し、[ファイル]-[パブリッシュプレビュー]-[HTML] を選択してブラウザウィンドウで SWF ファイルをテストします。ブラウザウィンドウ内で SWF をクリックしてフォーカスを与え、Tab キーを使ってそれぞれのインスタンスにフォーカスを移動します。_focusrect が無効になっている場合は、Enter キーまたはスペースキーを押してもこのボタンのコードは実行されません。

getDepth (Button.getDepth メソッド)

```
public getDepth() : Number
```

ボタンインスタンスの深度を返します。

各ムービークリップ、ボタン、およびテキストフィールドには関連付けられた一意の深度があり、どのように他のオブジェクトの手前や背後に表示されるかが決まります。深度の高い方のオブジェクトが手前に表示されます。

対応バージョン : ActionScript 1.0、Flash Player 6

戻り値

Number - ボタンインスタンスの深度。

例

myBtn1_btn と myBtn2_btn をステージに作成した場合、次の ActionScript を使用して深度をトレースできます。

```
trace(myBtn1_btn.getDepth());
trace(myBtn2_btn.getDepth());
```

buttonMovie.swf という名前の SWF ファイルをこのドキュメントにロードする場合、メイン SWF ファイルにある別のボタンを使用して、その SWF ファイル内のボタン myBtn4_btn の深度をトレースできます。

```
this.createEmptyMovieClip("myClip_mc", 999);
myClip_mc.loadMovie("buttonMovie.swf");
myBtn3_btn.onRelease = function(){
  trace(myClip_mc.myBtn4_btn.getDepth());
};
```

これらのボタンのうち、メイン SWF ファイル内のボタンとロードされた SWF ファイル内のボタンの深度は同じです。このように紛らわしいのは、buttonMovie.swf が深度 999 でロードされたため、そこに含まれるボタンの深度も、メイン SWF ファイル内のボタンを基準にした 999 になるためです。それぞれのムービークリップは独自の内部 z 順序を持つこと、つまり、それぞれのムービークリップには独自の深度値のセットがあることに注意してください。2 つのボタンは同じ深度値を持つことができますが、その値は、同じ z 順序内の他のオブジェクトとの関連においてのみ意味を持ちます。このケースでは、ボタンは同じ深度値を持ちますが、値はそれぞれ異なるムービークリップに関連付けられています。たとえば、メイン SWF ファイル内のボタンの深度値はメインタイムラインの z 順序に関連付けられており、ロードされた SWF ファイル内のボタンの深度値はムービークリップ myClip_mc の内部 z 順序に関連付けられています。

関連項目

getDepth (MovieClip.getDepth メソッド), getDepth (TextField.getDepth メソッド), getInstanceAtDepth (MovieClip.getInstanceAtDepth メソッド)

_height (Button._height プロパティ)

```
public _height : Number
```

ボタンの高さ (ピクセル単位) です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、my_btn という名前のボタンの高さと幅を、特定の高さと幅に設定します。

```
my_btn._width = 500;
my_btn._height = 200;
```

_highquality (Button._highquality プロパティ)

```
public _highquality : Number
```

非推奨 Flash Player 7 以降では使用しないでください。このプロパティの代わりに Button._quality を使用します。

現在の SWF ファイルに適用されるアンチエイリアスのレベルを指定します。ビットマップスムージングを常にオンにして最高品質を適用するには、2 (最高品質) を指定します。アンチエイリアス処理を適用するには、1(高品質) を指定します。SWF ファイルにアニメーションが含まれない場合は、ビットマップは滑らかになります。これはデフォルト値です。アンチエイリアス処理を避けるには、0 (低品質) を指定します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

ボタンインスタンスをステージに追加し、myBtn_btn という名前を付けます。楕円ツールを使用して、線と塗りのカラーを持つ楕円をステージ上に描きます。フレーム 1 を選択し、[アクション] パネルを使用して次の ActionScript を追加します。

```
myBtn_btn.onRelease = function() {
 myBtn_btn._highquality = 0;
};
```

myBtn_btn をクリックすると、円の線がギザギザになります。次の ActionScript を追加して、この効果を SWF にグローバルに適用することもできます。

```
_quality = 0;
```

関連項目

_quality (Button._quality プロパティ), _quality プロパティ

menu (Button.menu プロパティ)

```
public menu : ContextMenu
```

ContextMenu オブジェクト contextMenu を Button オブジェクト my_button に関連付けます。ContextMenu クラスを使用すると、ユーザーが Flash Player 内を右クリック (Windows) または Control キーを押しながらクリック (Macintosh) したときに表示されるコンテキストメニューを修正することができます。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例では、ContextMenu オブジェクトを myBtn_btn という名前のボタンインスタンスに割り当てます。ContextMenu オブジェクトには、"Save..." というラベルのメニューアイテムが 1 つあり、このメニューアイテムには doSave というコールバックハンドラ関数が関連付けられます。

ボタンインスタンスをステージに追加し、myBtn_btn という名前を付けます。

```
var menu_cm:ContextMenu = new ContextMenu();
menu_cm.customItems.push(new ContextMenuItem("Save...", doSave));
function doSave(menu:Object, obj:Object):Void {
  trace( " You selected the 'Save...' menu item ");
}
myBtn_btn.menu = menu_cm;
```

[制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。マウスカーソルを myBtn_btn 上に移動し、右クリック (Windows の場合) または Control キーを押したままクリック (Macintosh の場合) します。[Save] メニューアイテムを含むコンテキストメニューが表示されます。メニューから [Save] を選択すると、[出力] パネルが表示されます。

関連項目

ContextMenu, ContextMenuItem, menu (MovieClip.menu プロパティ), menu (TextField.menu プロパティ)

_name (Button._name プロパティ)

```
public _name : String
```

my_btn で指定されているボタンのインスタンス名です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、SWF ファイルの現在のタイムライン内に存在する Button インスタンスのインスタンス名をすべてトレースします。

```
for (i in this) {
  if (this[i] instanceof Button) {
  trace(this[i]._name);
  }
}
```

onDragOut (Button.onDragOut ハンドラ)

```
onDragOut = function() {}
```

マウスボタンをボタン上でクリックした後、ポインタをボタンの外側にドラッグすると、呼び出されます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ポインタをボタンの外にドラッグしたときにステートメントを実行する方法を示します。my_btn という名前のボタンをステージ上に作成し、タイムラインのフレームに次の ActionScript を追加します。

```
my_btn.onDragOut = function() {
  trace("onDragOut: "+this._name);
};
my_btn.onDragOver = function() {
  trace("onDragOver: "+this._name);
};
```

onDragOver (Button.onDragOver ハンドラ)

```
onDragOver = function() {}
```

マウスボタンを押したまま、一度ボタンの外側に移動し、次にボタン上に戻ると、呼び出されます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、[出力] パネルに `trace()` ステートメントを送る onDragOver ハンドラの関数を定義します。`my_btn` という名前のボタンをステージ上に作成し、タイムラインに次の ActionScript を追加します。

```
my_btn.onDragOut = function() {
  trace("onDragOut: "+this._name);
};
my_btn.onDragOver = function() {
  trace("onDragOver: "+this._name);
};
```

SWF ファイルをテストするときは、ポインタをいったんボタンインスタンスの外にドラッグします。次に、マウスボタンを押したまま、再びポインタをボタンインスタンス上にドラッグします。[出力] パネルにその操作が記録されます。

関連項目

`onDragOut (Button.onDragOut ハンドラ )`

onKeyDown (Button.onKeyDown ハンドラ)

```
onKeyDown = function() {}
```

ボタンにキーボードフォーカスがあるときにキーを押すと、呼び出されます。`onKeyDown` イベントハンドラにはパラメータは渡されません。`Key.getAscii()` メソッドと `Key.getCode()` メソッドを使用して、どのキーが押されたかを調べることができます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、`onKeyDown` ハンドラが呼び出されたときに [出力] パネルにテキストを送る関数を定義します。`my_btn` という名前のボタンをステージ上に作成し、タイムラインのフレームに次の ActionScript を追加します。

```
my_btn.onKeyDown = function() {
  trace("onKeyDown: "+this._name+" (Key: "+getKeyPressed()+")");
};
```

```
function getKeyPressed():String {
  var theKey:String;
  switch (Key.getAscii()) {
  case Key.BACKSPACE :
    theKey = "BACKSPACE";
    break;
  case Key.SPACE :
    theKey = "SPACE";
    break;
  default :
    theKey = chr(Key.getAscii());
  }
  return theKey;
}
```

[制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。テスト環境では、[制御]-[キーボードショートカットを無効] を選択してください。次に、Tab キーを押してボタンにフォーカスを移動し (my_btn インスタンスの周囲に黄色の矩形が表示されます)、キーボードのキーを押してテキストを入力します。キーを押して入力したテキストが [出力] パネルに表示されます。

関連項目

onKeyUp (Button.onKeyUp ハンドラ), getAscii (Key.getAscii メソッド), getCode (Key.getCode メソッド)

onKeyUp (Button.onKeyUp ハンドラ)

```
onKeyUp = function() {}
```

ボタンにフォーカスがあるときにキーを離すと、呼び出されます。onKeyUp イベントハンドラにはパラメータは渡されません。Key.getAscii() メソッドと Key.getCode() メソッドを使用して、どのキーが押されたかを調べることができます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、onKeyDown handler ハンドラが呼び出されたときに [出力] パネルにテキストを送信する関数を定義します。my_btn という名前のボタンをステージ上に作成し、タイムラインのフレームに次の ActionScript を追加します。

```
my_btn.onKeyDown = function() {
  trace("onKeyDown: "+this._name+" (Key: "+getKeyPressed()+")");
};
my_btn.onKeyUp = function() {
  trace("onKeyUp: "+this._name+" (Key: "+getKeyPressed()+")");
};
```

```
function getKeyPressed():String {
  var theKey:String;
  switch (Key.getAscii()) {
  case Key.BACKSPACE :
    theKey = "BACKSPACE";
    break;
  case Key.SPACE :
    theKey = "SPACE";
    break;
  default :
    theKey = chr(Key.getAscii());
  }
  return theKey;
}
```

Ctrl + Enter を押して SWF ファイルをテストします。テスト環境では、[制御]-[キーボードショートカットを無効] を選択してください。次に、Tab キーを押してボタンにフォーカスを移動し (my_btn インスタンスの周囲に黄色の矩形が表示されます)、キーボードのキーを押してテキストを入力します。キーを押して入力したテキストが [出力] パネルに表示されます。

関連項目

onKeyDown (Button.onKeyDown ハンドラ), getAscii (Key.getAscii メソッド), getCode (Key.getCode メソッド)

onKillFocus (Button.onKillFocus ハンドラ)

```
onKillFocus = function(newFocus:Object) {}
```

ボタンがキーボードフォーカスを失うと、呼び出されます。onKillFocus ハンドラは、1 つのパラメータ newFocus を受け取ります。このパラメータは、フォーカスを受け取る新しいオブジェクトを表すオブジェクトです。フォーカスを受け取るオブジェクトがない場合、newFocus の値は null です。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

newFocus:Object - フォーカスを受け取るオブジェクト。

例

次の例では、ボタンがフォーカスを失ったときにステートメントを実行する方法を示します。my_btn という名前のボタンインスタンスをステージ上に作成し、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
this.createTextField("output_txt", this.getNextHighestDepth(), 0, 0, 300, 200);
output_txt.wordWrap = true;
output_txt.multiline = true;
output_txt.border = true;
my_btn.onKillFocus = function() {
  output_txt.text = "onKillFocus: "+this._name+newline+output_txt.text;
};
```

SWF ファイルをブラウザウィンドウでテストするために、Tab キーを使用してウィンドウ内のエレメント間でフォーカスを移動します。ボタンインスタンスがフォーカスを失うと、テキストが output_txt テキストフィールドに送られます。

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

onPress (Button.onPress ハンドラ)

```
onPress = function() {}
```

ボタンを押すと呼び出されます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の例では、onPress ハンドラが呼び出されたときに trace() ステートメントを [出力] パネルに送る関数を定義します。

```
my_btn.onPress = function () {
  trace ("onPress called");
};
```

onRelease (Button.onRelease ハンドラ)

```
onRelease = function() {}
```

ボタンを離すと呼び出されます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、onRelease ハンドラが呼び出されたときに trace() ステートメントを [出力] パネルに送る関数を定義します。

```
my_btn.onRelease = function () {
   trace ("onRelease called");
};
```

onReleaseOutside (Button.onReleaseOutside ハンドラ)

```
onReleaseOutside = function() {}
```

ポインタがボタン内にあるときにボタンを押した後で、ポインタがボタン外にあるときにマウスを離すと、呼び出されます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、onReleaseOutside ハンドラが呼び出されたときに trace() ステートメントを [出力] パネルに送る関数を定義します。

```
my_btn.onReleaseOutside = function () {
   trace ("onReleaseOutside called");
};
```

onRollOut (Button.onRollOut ハンドラ)

```
onRollOut = function() {}
```

ポインタがボタン領域の外側に移動すると、呼び出されます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、onRollOut ハンドラが呼び出されたときに trace() ステートメントを [出力] パネルに送る関数を定義します。

```
my_btn.onRollOut = function () {
  trace ("onRollOut called");
};
```

onRollOver (Button.onRollOver ハンドラ)

```
onRollOver = function() {}
```

ポインタがボタン領域の上に移動すると、呼び出されます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、onRollOver ハンドラが呼び出されたときに trace() ステートメントを [出力] パネルに送る関数を定義します。

```
my_btn.onRollOver = function () {
  trace ("onRollOver called");
};
```

onSetFocus (Button.onSetFocus ハンドラ)

```
onSetFocus = function(oldFocus: Object) {}
```

ボタンがキーボードフォーカスを受け取ると、呼び出されます。oldFocus パラメータは、フォーカスを失うオブジェクトです。たとえば、Tab キーを押して入力フォーカスをテキストフィールドからボタンに移動すると、oldFocus にテキストフィールドのインスタンスが入ります。

前にフォーカスがあったオブジェクトが存在しない場合、oldFocus には null 値が入ります。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

`oldFocus:Object` - キーボードフォーカスを失うオブジェクト。

例

次の例では、SWF ファイルのユーザーがあるボタンから別のボタンにフォーカスを移動したときにステートメントを実行する方法を示します。btn1_btn と btn2_btn の 2 つのボタンを作成し、タイムラインのフレーム 1 に次のような ActionScript を追加します。

```
Selection.setFocus(btn1_btn);
trace(Selection.getFocus());
btn2_btn.onSetFocus = function(oldFocus) {
  trace(oldFocus._name + "lost focus");
};
```

Ctrl + Enter を押して SWF ファイルをテストします。[制御]-[キーボードショートカットを無効] を選択していない場合は選択してください。まず、btn1_btn がフォーカスを得ます。次に、btn1_btn がフォーカスを失い、btn2_btn がフォーカスを得ます。情報が [出力] パネルに表示されます。

_parent (Button._parent プロパティ)

```
public _parent : MovieClip
```

現在のムービークリップまたはオブジェクトを含むムービークリップまたはオブジェクトを指す参照です。現在のオブジェクトとは、_parent を参照する ActionScript コードがあるオブジェクトです。

現在のムービークリップまたはオブジェクトの上位のムービークリップまたはオブジェクトへの相対パスを指定するには、_parent を使用します。_parent を使用して表示リストの複数上のレベルに移動するには、次のようにします。

```
this._parent._parent._alpha = 20;
```

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例で、ボタン my_btn はムービークリップ my_mc 内に配置されています。次のコードでは、_parent プロパティを使用してムービークリップ my_mc への参照を取得しています。

```
trace(my_mc.my_btn._parent);
```

[出力] パネルには次のように表示されます。

```
_level0.my_mc
```

関連項目

_parent (MovieClip._parent プロパティ), _target (MovieClip._target プロパティ), _root プロパティ

_quality (Button._quality プロパティ)

```
public _quality : String
```

プロパティ (グローバル)。SWF ファイルに使用するレンダリング品質を設定または取得します。デバイスフォントは常にエイリアス処理されるため、_quality プロパティには影響されません。

_quality プロパティは、次の値に設定できます。

- "LOW" 低いレンダリング品質。グラフィックスはアンチエイリアス処理されず、ビットマップはスムージングされません。
- "MEDIUM" 普通のレンダリング品質。グラフィックスは 2 x 2 ピクセルグリッドを使用してアンチエイリアス処理されますが、ビットマップはスムージングされません。これは、テキストを含まないムービーに適しています。
- "HIGH" 高いレンダリング品質。グラフィックスは 4x4 ピクセルグリッドを使用してアンチエイリアス処理されます。ビットマップは、ムービーが静的なものである場合は、スムージングされます。デフォルトのレンダリング品質設定です。
- "BEST" 非常に高いレンダリング品質。グラフィックスは 4 x 4 ピクセルグリッドを使用してアンチエイリアス処理され、ビットマップは常にスムージングされます。

メモ：このプロパティを Button オブジェクトに対して指定することもできますが、実際にはグローバルプロパティであるため、単に _quality という形で値を指定することもできます。

対応バージョン：ActionScript 1.0、Flash Player 6

例

この例では、my_btn というボタンのレンダリング品質を LOW に設定します。

```
my_btn._quality = "LOW";
```

_rotation (Button._rotation プロパティ)

```
public _rotation : Number
```

ボタンの元の位置からの回転角 (度単位) です。時計回りに回転させる場合は 0 ～ 180 の値を指定します。反時計回りに回転させる場合は 0 ～ -180 の値を指定します。この範囲を超える値は、360 に加算または 360 から減算され、範囲内に収まる値になるように調整されます。たとえば、`my_btn._rotation = 450` というステートメントは `my_btn._rotation = 90` と同義です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ステージ上の 2 つのボタンを回転させます。ステージ上に `control_btn` と `my_btn` の 2 つのボタンを作成します。`my_btn` は、回転している状態を判別できるように、完全な円にしないでください。次に、タイムラインのフレーム 1 に次の ActionScript を入力します。

```
var control_btn:Button;
var my_btn:Button;
control_btn.onRelease = function() {
  my_btn._rotation += 10;
};
```

ステージ上にもう 1 つのボタン `myOther_btn` を作成します。このボタンも、(回転している状態を判別できるように) 完全な円にしないでください。タイムラインのフレーム 1 に次の ActionScript を入力します。

```
var myOther_btn:Button;
this.createEmptyMovieClip("rotater_mc", this.getNextHighestDepth());
rotater_mc.onEnterFrame = function() {
  myOther_btn._rotation += 2;
};
```

この例で使用している `MovieClip.getNextHighestDepth()` メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、`MovieClip.getNextHighestDepth()` メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

_rotation (MovieClip._rotation プロパティ), _rotation (TextField._rotation プロパティ)

scale9Grid (Button.scale9Grid プロパティ)

public scale9Grid : Rectangle

ボタンの 9 つの拡大・縮小領域を定義する矩形領域です。このプロパティが null に設定されている場合、拡大・縮小変換が適用されるときにボタン全体が通常どおりに拡大・縮小します。

ボタンの scale9Grid プロパティを定義すると、グリッドの中央領域を定義する scale9Grid 矩形をベースにして、ボタンは 9 つの領域を持つグリッドに分割されます。グリッドには、中央の領域の他に次の 8 つの領域があります。

- 中央の矩形の左上にある領域
- 矩形の上にある領域
- 矩形の右上にある領域
- 矩形の左側の領域
- 矩形の右側の領域
- 矩形の外側にある左下隅の領域
- 矩形の下側にある領域
- 矩形の外側にある右下隅の領域

矩形で定義される中心以外の 8 つの領域は、ボタンの拡大・縮小時に特別な規則が適用される額縁と考えることができます。

scale9Grid プロパティが設定されているときにボタンを拡大・縮小すると、テキストとグラデーションはすべて通常どおり拡大・縮小します。ただし、それ以外の種類のオブジェクトには次の規則が適用されます。

- 中央領域のコンテンツは通常どおり拡大・縮小します。
- 左上、右上、左下、右下のコンテンツは拡大・縮小しません。
- 上側領域と下側領域にあるコンテンツは水平方向にのみ拡大・縮小します。左側領域と右側領域にあるコンテンツは垂直方向にのみ拡大・縮小します。

ボタンが回転されると、それ以降の拡大・縮小はすべて通常どおりになり、scale9Grid プロパティは無視されます。

通常、scale9Grid プロパティは、ボタンを拡大・縮小しても境界線の太さが変化しないボタンを設定するときに使用します。

図および関連する例を含め、詳細については、MovieClip.scale9Grid を参照してください。

対応バージョン : ActionScript 1.0、Flash Player 8

関連項目

Rectangle (flash.geom.Rectangle), scale9Grid (MovieClip.scale9Grid プロパティ)

_soundbuftime (Button._soundbuftime プロパティ)

```
public _soundbuftime : Number
```

サウンドのストリーミングを開始するまでにサウンドをプリバッファする秒数を指定するプロパティです。

メモ：このプロパティを Button オブジェクトに対して指定することもできますが、実際にはロードしたすべてのサウンドに適用されるグローバルプロパティであるため、単に _soundbuftime という形で値を指定することもできます。このプロパティを Button オブジェクトに対して設定すると、実際にグローバルプロパティに設定されます。

詳細および使用例については、グローバルプロパティ _soundbuftime を参照してください。

対応バージョン： ActionScript 1.0、Flash Player 6

関連項目

_soundbuftime プロパティ

tabEnabled (Button.tabEnabled プロパティ)

```
public tabEnabled : Boolean
```

my_btn が自動タブ順に含まれているかどうかを指定します。デフォルト値は undefined です。

tabEnabled プロパティが undefined または true である場合、オブジェクトは自動タブ順に含まれます。tabIndex プロパティにも値を設定すると、オブジェクトはカスタムタブ順にも含まれます。tabEnabled が false の場合は、tabIndex プロパティが設定されていても、オブジェクトは自動タブ順またはカスタムタブ順に含まれません。

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の ActionScript を使用して、4 つのボタンのいずれかの tabEnabled プロパティを false に設定します。ただし、4 つのボタン (one_btn、two_btn、three_btn、および four_btn) はすべて、tabIndex を使用してカスタムタブ順で配置されます。three_btn の tabIndex が設定されていても、そのインスタンスの tabEnabled が false に設定されているので、three_btn はカスタムタブ順にも自動タブ順にも含まれません。4 つのボタンのタブ順を設定するには、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
three_btn.tabEnabled = false;
two_btn.tabIndex = 1;
four_btn.tabIndex = 2;
three_btn.tabIndex = 3;
one_btn.tabIndex = 4;
```

テスト環境で SWF ファイルをテストする場合は、[制御]-[キーボードショートカットを無効] を選択してキーボードショートカットを無効にしてください。

関連項目

tabIndex (Button.tabIndex プロパティ),tabEnabled (MovieClip.tabEnabled プロパティ),tabEnabled (TextField.tabEnabled プロパティ)

tabIndex (Button.tabIndex プロパティ)

public tabIndex : Number

SWF ファイル内のオブジェクトのタブ順をカスタマイズできます。ボタン、ムービークリップ、またはテキストフィールドインスタンスの tabIndex プロパティを設定できます。デフォルトの値は undefined です。

SWF ファイルに現在表示されているオブジェクトに tabIndex プロパティがある場合は、自動タブ順序が無効になり、SWF ファイルのオブジェクトの tabIndex プロパティからタブ順序が計算されます。カスタムタブ順には、tabIndex プロパティを持つオブジェクトのみが含まれます。

tabIndex プロパティは、通常、負以外の整数です。オブジェクトのタブ順は、その tabIndex プロパティに従って昇順に決定されます。tabIndex の値が 1 であるオブジェクトは、tabIndex の値が 2 であるオブジェクトよりも前になります。2 つのオブジェクトの tabIndex が同じ値である場合、オブジェクトのタブ順は undefined になります。

tabIndex プロパティによって定義されるカスタムタブ順は *flat* です。つまり、SWF ファイル内のオブジェクトの階層関係は無視されます。SWF ファイルで tabIndex プロパティを持つすべてのオブジェクトは、タブ順序に従って配置されます。タブ順序は tabIndex の値の順番に従います。tabIndex の値が同じである 2 つのオブジェクト間では、優先順が undefined になります。複数のオブジェクトの tabIndex に同じ値を使用しないでください。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の ActionScript を使用して、4 つのボタンのいずれかの tabEnabled プロパティを false に設定します。ただし、4 つのボタン (one_btn、two_btn、three_btn、および four_btn) はすべて、tabIndex を使用してカスタムタブ順で配置されます。three_btn の tabIndex が設定されていても、そのインスタンスの tabEnabled が false に設定されているので、three_btn はカスタムタブ順にも自動タブ順にも含まれません。4 つのボタンのタブ順を設定するには、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
three_btn.tabEnabled = false;
two_btn.tabIndex = 1;
four_btn.tabIndex = 2;
three_btn.tabIndex = 3;
one_btn.tabIndex = 4;
```

テスト環境で SWF ファイルをテストする場合は、[制御]-[キーボードショートカットを無効] を選択してキーボードショートカットを無効にしてください。

関連項目

tabEnabled (Button.tabEnabled プロパティ), tabChildren (MovieClip.tabChildren プロパティ), tabEnabled (MovieClip.tabEnabled プロパティ), tabIndex (MovieClip.tabIndex プロパティ), tabIndex (TextField.tabIndex プロパティ)

_target (Button._target プロパティ)

public _target : String (読み取り専用)

my_btn で指定されているボタンインスタンスのターゲットパスを返します。

対応バージョン: ActionScript 1.0、Flash Player 6

例

my_btn という名前のボタンインスタンスをステージに追加し、タイムラインのフレーム 1 に次のコードを追加します。

```
trace(my_btn._target); //displays /my_btn
```

my_btn を選択し、ムービークリップに変換します。新しいムービークリップに my_mc というインスタンス名を付けます。タイムラインのフレーム 1 の既存の ActionScript を削除し、次のコードで置き換えます。

```
my_mc.my_btn.onRelease = function(){
  trace(this._target); //displays /my_mc/my_btn
};
```

スラッシュ表記をドット表記に変換するには、上記のコードを次のように変更します。

```
my_mc.my_btn.onRelease = function(){
  trace(eval(this._target)); //displays _level0.my_mc.my_btn
};
```

これにより、次のようにターゲットオブジェクトのメソッドやパラメータにアクセスできるようになります。

```
my_mc.my_btn.onRelease = function(){
  var target_btn:Button = eval(this._target);
trace(target_btn._name); //displays my_btn
};
```

関連項目

_target (MovieClip._target プロパティ)

trackAsMenu (Button.trackAsMenu プロパティ)

```
public trackAsMenu : Boolean
```

他のボタンまたはムービークリップがマウスの解放イベントを受け取ることができるかどうかを示すブール値です。ボタンをドラッグし、2 番目のボタン上で離すと、onRelease イベントが 2 番目のボタンに対して登録されます。このプロパティを使用してメニューを作成できます。trackAsMenu プロパティは、任意のボタンオブジェクトまたはムービークリップオブジェクトで設定できます。trackAsMenu プロパティが定義されていない場合、デフォルトのビヘイビアは false です。

trackAsMenu プロパティは必要に応じていつでも変更できます。変更は即座に反映されます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、2 つのボタンをメニューとして追跡する方法を示します。ステージ上に one_btn と two_btn の 2 つのボタンインスタンスを配置します。タイムラインに次の ActionScript を配置します。

```
var one_btn:Button;
var two_btn:Button;
one_btn.trackAsMenu = true;
two_btn.trackAsMenu = true
one_btn.onRelease = function() {
  trace("clicked one_btn");
};
two_btn.onRelease = function() {
  trace("clicked two_btn");
};
```

SWF ファイルをテストします。ステージの one_btn 上でクリックし、マウスボタンを押したまま two_btn 上に移動した後でマウスボタンを離します。次に、ActionScript の中で trackAsMenu を含む 2 行をコメントアウトします。再び SWF ファイルをテストして、ボタンのビヘイビアの違いを確認します。

関連項目

trackAsMenu (MovieClip.trackAsMenu プロパティ)

_url (Button._url プロパティ)

public _url : String (読み取り専用)

ボタンを作成した SWF ファイルの URL を取得します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

ステージ上に one_btn と two_btn の 2 つのボタンインスタンスを作成します。タイムラインのフレーム 1 に次の ActionScript を入力します。

```
var one_btn:Button;
var two_btn:Button;
this.createTextField("output_txt", 999, 0, 0, 100, 22);
output_txt.autoSize = true;
one_btn.onRelease = function() {
  trace("clicked one_btn");
  trace(this._url);
};
two_btn.onRelease = function() {
  trace("clicked "+this._name);
  var url_array:Array = this._url.split("/");
  var my_str:String = String(url_array.pop());
  output_txt.text = unescape(my_str);
};
```

それぞれのボタンをクリックすると、ボタンが含まれる SWF のファイル名が [出力] パネルに表示されます。

useHandCursor (Button.useHandCursor プロパティ)

public useHandCursor : Boolean

マウスがボタン上に移動したときに、指差しハンドポインタ (ハンドカーソル) を表示するかどうかを示すブール値。デフォルト値は true です。このプロパティを false に設定すると、代わりに矢印ポインタが使われます。

useHandCursor プロパティは必要に応じていつでも変更できます。変更は即座に反映されます。useHandCursor プロパティは、プロトタイプオブジェクトから読み取ることができます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

ステージ上に 2 つのボタンを作成し、それぞれに myBtn1_btn と myBtn2_btn というインスタンス名を付けます。タイムラインのフレーム 1 に次の ActionScript を入力します。

```
myBtn1_btn.useHandCursor = false;
myBtn1_btn.onRelease = buttonClick;
myBtn2_btn.onRelease = buttonClick;
function buttonClick() {
  trace(this._name);
}
```

マウスを myBtn1_btn 上に移動してクリックしても、指差しハンドポインタは表示されません。一方、マウスを myBtn2_btn 上に移動してクリックすると、指差しハンドポインタが表示されます。

_visible (Button._visible プロパティ)

```
public _visible : Boolean
```

my_btn で指定されているボタンを表示するかどうかを示すブール値です。(_visible プロパティが false に設定されている) 非表示ボタンは、使用できません。

対応バージョン : ActionScript 1.0、Flash Player 6

例

ステージ上に 2 つのボタンを作成し、それぞれに myBtn1_btn と myBtn2_btn というインスタンス名を付けます。タイムラインのフレーム 1 に次の ActionScript を入力します。

```
myBtn1_btn.onRelease = function() {
  this._visible = false;
  trace("clicked "+this._name);
};
myBtn2_btn.onRelease = function() {
  this._alpha = 0;
  trace("clicked "+this._name);
};
```

アルファを 0 に設定した後でも myBtn2_btn をクリックできます。

関連項目

_visible (MovieClip._visible プロパティ),_visible (TextField._visible プロパティ)

_width (Button._width プロパティ)

```
public _width : Number
```

ボタンの幅 (ピクセル単位) です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ボタン my_btn の width プロパティの値を大きくして、幅を [出力] パネルに表示します。タイムラインのフレーム 1 に次の ActionScript を入力します。

```
my_btn.onRelease = function() {
  trace(this._width);
  this._width *= 1.1;
};
```

関連項目

_width (MovieClip._width プロパティ)

_x (Button._x プロパティ)

```
public _x : Number
```

親ムービークリップのローカル座標を基準にしたボタンの x 座標を設定する整数です。メインのタイムラインにあるムービークリップの座標系は、ステージの左上隅を (0,0) として参照します。変形されているムービークリップ内にあるボタンの座標系は、ボタンを囲むムービークリップのローカル座標系になります。したがって、反時計回りに 90 度回転したムービークリップ内のボタンは、反時計回りに 90 度回転した座標系を継承します。ボタンの座標は、基準点の位置を参照します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ステージ上の my_btn の座標を 0 に設定します。ボタン my_btn を作成し、タイムラインのフレーム 1 に次の ActionScript を入力します。

```
my_btn._x = 0;
my_btn._y = 0;
```

関連項目

_xscale (Button._xscale プロパティ), _y (Button._y プロパティ),
_yscale (Button._yscale プロパティ)

_xmouse (Button._xmouse プロパティ)

public _xmouse : Number (読み取り専用)

ボタンを基準にした相対的なマウス位置の *x* 座標を返します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ステージの xmouse 位置と、ステージ上に配置されたボタン my_btn を表示します。タイムラインのフレーム 1 に次の ActionScript を入力します。

```
this.createTextField("mouse_txt", 999, 5, 5, 150, 40);
mouse_txt.html = true;
mouse_txt.wordWrap = true;
mouse_txt.border = true;
mouse_txt.autoSize = true;
mouse_txt.selectable = false;
//
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
   var table_str:String = "<textformat tabstops='[50,100]'>";
   table_str += "<b>Stage</b>\t"+"x:"+_xmouse+"\t"+"y:"+_ymouse+newline;
   table_str += "<b>Button</
   b>\t"+"x:"+my_btn._xmouse+"\t"+"y:"+my_btn._ymouse+newline;
   table_str += "</textformat>";
   mouse_txt.htmlText = table_str;
};
Mouse.addListener(mouseListener);
```

関連項目

_ymouse (Button._ymouse プロパティ)

_xscale (Button._xscale プロパティ)

public _xscale : Number

ボタンの基準点から適用した場合のボタンの水平方向の拡大・縮小率 (パーセント単位) です。デフォルトの基準点は (0,0) です。

ローカル座標系を拡大または縮小すると、_x プロパティと _y プロパティの設定に影響します。この設定はピクセル単位で表されます。たとえば、親ムービークリップを 50% に縮小して、_x プロパティを設定すると、ボタン内のオブジェクトの移動距離は、拡大・縮小率が 100% だったときの半分のピクセル数になります。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ボタン my_btn を拡大します。ボタンをクリックして離すと、ボタンは *x* 軸および *y* 軸方向に 10% ずつ拡大されます。タイムラインのフレーム 1 に次の ActionScript を入力します。

```
my_btn.onRelease = function(){
  this._xscale *= 1.1;
  this._yscale *= 1.1;
};
```

関連項目

_x (Button._x プロパティ), _y (Button._y プロパティ), _yscale (Button._yscale プロパティ)

_y (Button._y プロパティ)

```
public _y : Number
```

親ムービークリップのローカル座標を基準にしたボタンの y 座標です。メインのタイムラインにあるボタンの座標系は、ステージの左上隅を (0,0) として参照します。変形されている別のムービークリップ内にあるボタンの座標系は、ボタンを囲むムービークリップのローカル座標系になります。したがって、反時計回りに 90 度回転したムービークリップ内のボタンは、反時計回りに 90 度回転した座標系を継承します。ボタンの座標は、基準点の位置を参照します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ステージ上の my_btn の座標を 0 に設定します。ボタン my_btn を作成し、タイムラインのフレーム 1 に次の ActionScript を入力します。

```
my_btn._x = 0;
my_btn._y = 0;
```

関連項目

_x (Button._x プロパティ), _xscale (Button._xscale プロパティ), _yscale (Button._yscale プロパティ)

_ymouse (Button._ymouse プロパティ)

public _ymouse : Number (読み取り専用)

ボタンを基準にした相対的なマウス位置の y 座標を示します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ステージの ymouse 位置と、ステージ上に配置されたボタン my_btn を表示します。タイムラインのフレーム 1 に次の ActionScript を入力します。

```
this.createTextField("mouse_txt", 999, 5, 5, 150, 40);
mouse_txt.html = true;
mouse_txt.wordWrap = true;
mouse_txt.border = true;
mouse_txt.autoSize = true;
mouse_txt.selectable = false;
//
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
   var table_str:String = "<textformat tabstops='[50,100]'>";
   table_str += "<b>Stage</b>\t"+"x:"+_xmouse+"\t"+"y:"+_ymouse+newline;
   table_str += "<b>Button</
   b>\t"+"x:"+my_btn._xmouse+"\t"+"y:"+my_btn._ymouse+newline;
   table_str += "</textformat>";
   mouse_txt.htmlText = table_str;
};
Mouse.addListener(mouseListener);
```

関連項目

_xmouse (Button._xmouse プロパティ)

_yscale (Button._yscale プロパティ)

public _yscale : Number

ボタンの基準点から適用した場合のボタンの垂直方向の拡大・縮小率 (パーセント単位) です。デフォルトの基準点は (0,0) です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ボタン my_btn を拡大します。ボタンをクリックして離すと、ボタンは *x* 軸および *y* 軸方向に 10% ずつ拡大されます。タイムラインのフレーム 1 に次の ActionScript を入力します。

```
my_btn.onRelease = function(){
   this._xscale *= 1.1;
   this._yscale *= 1.1;
};
```

関連項目

_y（Button._y プロパティ）, _x（Button._x プロパティ）, _xscale（Button._xscale プロパティ）

カメラ

```
Object
  |
  +-Camera
```

```
public class Camera
extends Object
```

Camera クラスは主に Flash Media Server で使われますが、サーバーなしでも限定的な方法で使用できます。

Camera クラスを使用すれば、Flash Player を実行するコンピュータに接続されたビデオカメラからビデオをキャプチャできます。たとえば、ローカルシステムに接続した Web カメラから送られてくるビデオを監視できます。Flash にはこれと似たオーディオ機能も用意されています。詳細については、Microphone クラスを参照してください。

警告： SWF ファイルが Camera.get() から返されたカメラにアクセスしようとすると、そのカメラへのアクセスを許可するかどうかをユーザーが選択するための [プライバシー] ダイアログボックスが表示されます。Camera クラスの例として使用するステージのサイズは 215 x 138 ピクセル以上にしてください。これは、ダイアログボックスを表示するために必要な Flash の最小サイズです。エンドユーザーおよび管理ユーザーがカメラへのアクセスをサイト単位で、またはグローバルに無効にすることもできます。

Camera オブジェクトを作成または参照するには、Camera.get() メソッドを使います。

対応バージョン： ActionScript 1.0、Flash Player 6

プロパティー覧

オプション	プロパティ	説明
	activityLevel: Number (読み取り専用)	カメラが検知しているモーション量を示す数値です。
	bandwidth:Number (読み取り専用)	現在の送信ビデオフィードで使用できる最大帯域幅 (バイト単位) を指定する整数です。
	currentFps:Number (読み取り専用)	現在のデータキャプチャレート (1 秒あたりのフレーム数) です。
	fps:Number (読み取り専用)	希望する最大データキャプチャレート (1 秒あたりのフレーム数) です。
	height:Number (読み取り専用)	現在のキャプチャの高さ (ピクセル単位) です。
	index:Number (読み取り専用)	カメラのインデックスを指定するゼロから始まる整数です。これは、Camera.names から返される配列のインデックスと同じです。
	motionLevel:Number (読み取り専用)	Camera.onActivity(true) を呼び出すのに必要なモーション量を指定する数値です。
	motionTimeOut:Number (読み取り専用)	カメラがモーション検知を停止してから、Camera.onActivity (false) が呼び出されるまでの時間 (ミリ秒単位) です。
	muted:Boolean (読み取り専用)	ユーザーが [Macromedia Flash Player 設定] パネルの [プライバシー] でカメラへのアクセスを禁止したか (true)、または許可したか (false) を示すブール値です。
	name:String (読み取り専用)	現在のカメラの名前を示すストリングです。カメラのハードウェアから返されるものです。
static	names:Array (読み取り専用)	[Macromedia Flash Player 設定] パネルの [プライバシー] を表示せずに、使用できるすべてのカメラの名前が含まれるストリング配列を取得します。
	quality:Number (読み取り専用)	必要な画質レベルを指定する整数です。各ビデオフレームに適用される圧縮率によって決まります。
	width:Number (読み取り専用)	現在のキャプチャの幅 (ピクセル単位) です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベントの一覧

イベント	説明
onActivity = function(active: Boolean) {}	カメラがモーションの検知を開始または停止すると、呼び出されます。
onStatus = function(infoObject: Object) {}	ユーザーがカメラへのアクセスを許可または禁止すると、呼び出されます。

メソッド一覧

オプション	署名	説明
static	get([index:Number]) : Camera	ビデオをキャプチャする Camera オブジェクトへの参照を返します。
	setMode([width:Number], [height:Number], [fps:Number], [favorArea:Boolean]) : Void	カメラのキャプチャモードを指定の要件に最も近いネイティブモードに設定します。
	setMotionLevel ([motionLevel:Number], [timeOut:Number]) : Void	Camera.onActivity(true) を呼び出すのに必要なモーション量を指定します。
	setQuality ([bandwidth:Number], [quality:Number]) : Void	現在の送信ビデオフィードの1秒あたりの最大帯域幅、または必要な画質を設定します。

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

activityLevel (Camera.activityLevel プロパティ)

public activityLevel : Number (読み取り専用)

カメラが検知しているモーション量を示す数値です。0 (モーションは検知されていない) から 100 (大量のモーションが検知されている) までの値が設定されます。このプロパティの値は、Camera.setMotionLevel( ) に設定を渡す必要があるかどうかを判断するのに役立ちます。

Video.attachVideo( ) が呼び出されていないため、使用可能なカメラがまだ使用されていない場合は、このプロパティが -1 に設定されます。

圧縮されていないローカルビデオだけをストリーミングする場合、このプロパティは Camera.onActivity イベントハンドラに関数を割り当て済みであるときにかぎり設定されます。それ以外の場合は undefined になります。

対応バージョン : ActionScript 1.0、Flash Player 6

例

この例では、activityLevel プロパティと ProgressBar インスタンスを使用して、カメラが検知するモーション量を検知します。[ライブラリ] オプションメニューから [新規ビデオ] を選択して新しいビデオインスタンスを作成します。インスタンスをステージに追加し、そのインスタンスに my_video という名前を付けます。ProgressBar コンポーネントのインスタンスをステージに追加し、そのインスタンスに activity_pb という名前を付けます。次に、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
// video instance on the Stage.
var my_video:Video;
var activity_pb:mx.controls.ProgressBar;
var my_cam:Camera = Camera.get();
my_video.attachVideo(my_cam);
activity_pb.mode = "manual";
activity_pb.label = "Activity %3%%";

this.onEnterFrame = function() {
  activity_pb.setProgress(my_cam.activityLevel, 100);
};
```

```
my_cam.onActivity = function(isActive:Boolean) {
   var themeColor:String = (isActive) ? "haloGreen" : "haloOrange";
   activity_pb.setStyle("themeColor", themeColor);
};
```

関連項目

motionLevel (Camera.motionLevel プロパティ),
setMotionLevel (Camera.setMotionLevel メソッド)

bandwidth (Camera.bandwidth プロパティ)

public bandwidth : Number (読み取り専用)

現在の送信ビデオフィードで使用できる最大帯域幅 (バイト単位) を指定する整数です。0 は、ビデオのフレーム品質を維持するために必要な帯域幅を使用できることを示します。

このプロパティの値を設定するには、Camera.setQuality() を使います。

対応バージョン: ActionScript 1.0、Flash Player 6

例

次の例では、カメラフィードで使用する最大帯域幅を変更します。[ライブラリ] オプションメニューから [新規ビデオ] を選択して新しいビデオインスタンスを作成します。インスタンスをステージに追加し、そのインスタンスに my_video という名前を付けます。NumericStepper コンポーネントのインスタンスをステージに追加し、そのインスタンスに bandwidth_nstep という名前を付けます。次に、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var bandwidth_nstep:mx.controls.NumericStepper;
var my_video:Video;
var my_cam:Camera = Camera.get();
my_video.attachVideo(my_cam);
this.createTextField("bandwidth_txt", this.getNextHighestDepth(), 0, 0, 100,
   22);
bandwidth_txt.autoSize = true;
this.onEnterFrame = function() {
   bandwidth_txt.text = "Camera is currently using "+my_cam.bandwidth+" bytes
   ("+Math.round(my_cam.bandwidth/1024)+" KB) bandwidth.";
};
//
bandwidth_nstep.minimum = 0;
bandwidth_nstep.maximum = 128;
bandwidth_nstep.stepSize = 16;
bandwidth_nstep.value = my_cam.bandwidth/1024;
function changeBandwidth(evt:Object) {
   my_cam.setQuality(evt.target.value 1024, 0);
}
bandwidth_nstep.addEventListener("change", changeBandwidth);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

setQuality (Camera.setQuality メソッド)

currentFps (Camera.currentFps プロパティ)

public currentFps : Number (読み取り専用)

現在のデータキャプチャレート (1 秒あたりのフレーム数) です。このプロパティを設定することはできません。ただし、Camera.setMode() メソッドを使うと、これに関係する Camera.fps プロパティを設定できます。このプロパティは、データをキャプチャする際の最大フレームレートを指定します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、currentFps プロパティと ProgressBar インスタンスを使用して、カメラのデータキャプチャレート (1 秒あたりのフレーム数) を検知します。[ライブラリ] オプションメニューから [新規ビデオ] を選択して新しいビデオインスタンスを作成します。インスタンスをステージに追加し、そのインスタンスに my_video という名前を付けます。ProgressBar コンポーネントのインスタンスをステージに追加し、そのインスタンスに fps_pb という名前を付けます。次に、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var my_video:Video;
var fps_pb:mx.controls.ProgressBar;
var my_cam:Camera = Camera.get();
my_video.attachVideo(my_cam);
this.onEnterFrame = function() {
  fps_pb.setProgress(my_cam.fps-my_cam.currentFps, my_cam.fps);
};

fps_pb.setStyle("fontSize", 10);
fps_pb.setStyle("themeColor", "haloOrange");
fps_pb.labelPlacement = "top";
fps_pb.mode = "manual";
fps_pb.label = "FPS: %2 (%3%% dropped)";
```

関連項目

setMode (Camera.setMode メソッド), fps (Camera.fps プロパティ)

fps (Camera.fps プロパティ)

public fps : Number (読み取り専用)

希望する最大データキャプチャレート (1 秒あたりのフレーム数) です。使用できる最大レートは、カメラの性能によって異なります。このプロパティに設定した値をカメラがサポートしていない場合は、このフレームレートは達成されません。

- このプロパティに希望の値を設定するには、Camera.setMode( ) メソッドを使用します。
- 現在のデータキャプチャレートを調べるには、Camera.currentFps プロパティを使用します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、currentFps プロパティと ProgressBar インスタンスを使用して、カメラのデータキャプチャレート (1 秒あたりのフレーム数) を検知します。[ライブラリ] オプションメニューから [新規ビデオ] を選択して新しいビデオインスタンスを作成します。インスタンスをステージに追加し、そのインスタンスに my_video という名前を付けます。ProgressBar コンポーネントのインスタンスをステージに追加し、そのインスタンスに fps_pb という名前を付けます。次に、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var my_video:Video;
var fps_pb:mx.controls.ProgressBar;
var my_cam:Camera = Camera.get();
my_video.attachVideo(my_cam);
this.onEnterFrame = function() {
  fps_pb.setProgress(my_cam.fps-my_cam.currentFps, my_cam.fps);
};

fps_pb.setStyle("fontSize", 10);
fps_pb.setStyle("themeColor", "haloOrange");
fps_pb.labelPlacement = "top";
fps_pb.mode = "manual";
fps_pb.label = "FPS: %2 (%3%% dropped)";
```

メモ : setMode( ) 関数を使用しても、要求した fps が設定されない場合があります。要求した fps、または可能な限り最大の fps のいずれか一方が設定されます。

関連項目

currentFps (Camera.currentFps プロパティ), setMode (Camera.setMode メソッド)

get (Camera.get メソッド)

```
public static get([index: Number]) : Camera
```

ビデオをキャプチャする Camera オブジェクトへの参照を返します。ビデオのキャプチャを実際に開始するには、Camera オブジェクトを Video オブジェクトに関連付ける必要があります。詳細については、`Video.attachVideo()` を参照してください。

`new` コンストラクタを使用して作成するオブジェクトとは異なり、`Camera.get()` を複数回呼び出した場合は、同じカメラへの参照が返されます。したがって、スクリプトに `first_cam = Camera.get()` という行と `second_cam = Camera.get()` という行が含まれている場合、`first_cam` と `second_cam` はどちらも同じデフォルトのカメラを参照します。

一般には、*index* の値は指定せず、単に `Camera.get()` を使用してデフォルトのカメラへの参照を取得します。ユーザーは、このセクションで後述する [カメラ] ボックスで、Flash で使用するデフォルトのカメラを指定できます。*index* の値を指定すると、ユーザーの希望と異なるカメラへの参照を取得する可能性があります。*index* を使用するのは、特殊な場合に限られます (たとえば、アプリケーションで 2 つのカメラから同時にビデオをキャプチャする場合)。

SWF ファイルで `Camera.get()` から返されたカメラにアクセスしようとすると、そのカメラへのアクセスを許可するかどうかをユーザーが選択するための [プライバシー] ダイアログボックスが表示されます。ステージのサイズは必ず 215 x 138 ピクセル以上に設定してください。これは、ダイアログボックスを表示するために必要な最小サイズです。

ユーザーがこのダイアログボックスに応答すると、`Camera.onStatus` イベントハンドラがユーザーの応答を示す情報オブジェクトを返します。このイベントハンドラを使わずに、ユーザーがカメラへのアクセスを許可したかどうかを判断するには、`Camera.muted` プロパティを使用します。

ユーザーは、再生中の SWF ファイルを右クリック (Windows) または Control キーを押しながらクリック (Macintosh) し、[設定] を選択して [プライバシー] パネルを開き、[後で確認] を選択すると、特定のドメインに対する永続的なプライバシー設定を指定できます。

ActionScript からユーザーに対して許可または拒否を設定することはできません。ただし、`System.showSettings(0)` を使用することで、ユーザーの [プライバシー] パネルを表示できます。ユーザーが [後で確認] を選択すると、このドメインの SWF ファイルでは [プライバシー] ダイアログボックスが表示されなくなります。

`Camera.get` が `null` を返した場合、カメラは他のアプリケーションによって使用されているか、そのシステムにはカメラがインストールされていません。カメラがインストールされているかどうかを調べるには、`Camera.names.length` を使用します。`Camera.get()` が参照するカメラをユーザーが選択するための [カメラ] ボックスを表示するには、`System.showSettings(3)` を使用します。

カメラのハードウェアスキャンには時間がかかります。カメラが 1 つでも見つかると、その Flash Player インスタンスの存続中は、ハードウェアが再びスキャンされることはありません。しかし、カメラが見つからなかった場合は、Camera.get が呼び出されるたびにハードウェアがスキャンされます。この動作は、ユーザーがカメラを接続するのを忘れた場合に便利です。Camera.get を呼び出すための [再試行] ボタンを SWF ファイルに用意すれば、ユーザーが SWF ファイルを再起動しなくてもカメラを見つけることができます。

メモ：正しいシンタックスは Camera.get() です。Camera オブジェクトを変数に代入するには、active_cam = Camera.get() のようなシンタックスを使用します。

対応バージョン：ActionScript 1.0、Flash Player 6

パラメータ

index:Number (オプション) - 取得するカメラを指定するゼロから始まる整数。このパラメータ値は、Camera.names プロパティから返される配列で決まります。デフォルトのカメラを取得するには、このパラメータを省略します。ほとんどのアプリケーションでは、デフォルトのカメラを使用することをお勧めします。

戻り値

Camera - index を指定しない場合は、デフォルトのカメラへの参照を返します。デフォルトのカメラが他のアプリケーションで使用されている場合は、使用できる最初のカメラへの参照を返します。複数のカメラがインストールされている場合、ユーザーは [Macromedia Flash Player 設定] パネルの [カメラ] でデフォルトのカメラを指定できます。利用できるカメラがない場合、またはカメラがインストールされていない場合は、null を返します。index を指定した場合は、指定したカメラへの参照を返します。そのカメラが利用できない場合は null を返します。

例

次の例では、使用するアクティブなカメラを ComboBox インスタンスから選択できるようにします。現在アクティブであるカメラは Label インスタンスに表示されます。[ライブラリ] オプションメニューから [新規ビデオ] を選択して新しいビデオインスタンスを作成します。インスタンスをステージに追加し、そのインスタンスに my_video という名前を付けます。Label コンポーネントのインスタンスをステージに追加し、そのインスタンスに camera_lbl という名前を付けます。ComboBox コンポーネントのインスタンスをステージに追加し、そのインスタンスに cameras_cb という名前を付けます。次に、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);
var camera_lbl:mx.controls.Label;
var cameras_cb:mx.controls.ComboBox;
camera_lbl.text = my_cam.name;
cameras_cb.dataProvider = Camera.names;
```

```
function changeCamera():Void {
  my_cam = Camera.get(cameras_cb.selectedIndex);
  my_video.attachVideo(my_cam);
  camera_lbl.text = my_cam.name;
}
cameras_cb.addEventListener("change", changeCamera);
camera_lbl.setStyle("fontSize", 9);
cameras_cb.setStyle("fontSize", 9);
```

関連項目

index (Camera.index プロパティ), muted (Camera.muted プロパティ),
names (Camera.names プロパティ), onStatus (Camera.onStatus ハンドラ),
showSettings (System.showSettings メソッド),
attachVideo (Video.attachVideo メソッド)

height (Camera.height プロパティ)

public height : Number [読み取り専用]

現在のキャプチャの高さ (ピクセル単位) です。このプロパティの値を設定するには、Camera.setMode( ) を使用します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次のコードは、ビデオインスタンスの現在の幅、高さ、および FPS をステージ上の Label コンポーネントインスタンスに表示します。[ライブラリ] オプションメニューから [新規ビデオ] を選択して新しいビデオインスタンスを作成します。インスタンスをステージに追加し、そのインスタンスに my_video という名前を付けます。Label コンポーネントのインスタンスをステージに追加し、そのインスタンスに dimensions_lbl という名前を付けます。次に、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);
var dimensions_lbl:mx.controls.Label;
dimensions_lbl.setStyle("fontSize", 9);
dimensions_lbl.setStyle("fontWeight", "bold");
dimensions_lbl.setStyle("textAlign", "center");
dimensions_lbl.text = "width: "+my_cam.width+", height: "+my_cam.height+", FPS:
  "+my_cam.fps;
```

Camera.setMode( ) の例も参照してください。

関連項目

width (Camera.width プロパティ), setMode (Camera.setMode メソッド)

index (Camera.index プロパティ)

public index : Number (読み取り専用)

カメラのインデックスを指定するゼロから始まる整数です。これは、Camera.names から返される配列のインデックスと同じです。

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の例では、実行時に作成されるテキストフィールドにカメラの配列を表示して、現在使用されているカメラを示します。[ライブラリ] オプションメニューから [新規ビデオ] を選択して新しいビデオインスタンスを作成します。インスタンスをステージに追加し、そのインスタンスに my_video という名前を付けます。Label コンポーネントのインスタンスをステージに追加し、そのインスタンスに camera_lbl という名前を付けます。次に、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var camera_lbl:mx.controls.Label;
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);

camera_lbl.text = my_cam.index+". "+my_cam.name;
this.createTextField("cameras_txt", this.getNextHighestDepth(), 25, 160, 160,
  80);
cameras_txt.html = true;
cameras_txt.border = true;
cameras_txt.wordWrap = true;
cameras_txt.multiline = true;
for (var i = 0; i<Camera.names.length; i++) {
  cameras_txt.htmlText += "<li><u><a
  href=\"asfunction:changeCamera,"+i+"\">"+Camera.names[i]+"</a></u></li>";
}
function changeCamera(index:Number) {
  my_cam = Camera.get(index);
  my_video.attachVideo(my_cam);
  camera_lbl.text = my_cam.index+". "+my_cam.name;
}
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

names （Camera.names プロパティ), get （Camera.get メソッド)

motionLevel (Camera.motionLevel プロパティ)

public motionLevel : Number (読み取り専用)

Camera.onActivity(true) を呼び出すのに必要なモーション量を指定する数値です。指定できる値は 0 ～ 100 です。デフォルト値は 50 です。

ビデオは motionLevel プロパティの値に関係なく表示できます。詳細については、Camera.setMotionLevel() を参照してください。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、カメラフィードのモーションレベルを連続的に検知します。[ライブラリ] オプションメニューから [新規ビデオ] を選択して新しいビデオインスタンスを作成します。インスタンスをステージに追加し、そのインスタンスに my_video という名前を付けます。Label コンポーネント、NumericStepper コンポーネント、および ProgressBar コンポーネントのインスタンスを 1 つずつステージに追加し、それぞれのインスタンスに motionLevel_lbl、motionLevel_nstep、および motion_pb という名前を付けます。次に、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);

// configure the ProgressBar component instance
var motion_pb:mx.controls.ProgressBar;
motion_pb.mode = "manual";
motion_pb.label = "Motion: %3%%";

var motionLevel_lbl:mx.controls.Label;
// configure the NumericStepper component instance
var motionLevel_nstep:mx.controls.NumericStepper;
motionLevel_nstep.minimum = 0;
motionLevel_nstep.maximum = 100;
motionLevel_nstep.stepSize = 5;
motionLevel_nstep.value = my_cam.motionLevel;

// Continuously update the progress of the ProgressBar component instance to the
   activityLevel
// of the current Camera instance, which is defined in my_cam
this.onEnterFrame = function() {
   motion_pb.setProgress(my_cam.activityLevel, 100);
};

// When the level of activity goes above or below the number defined in
   Camera.motionLevel,
// trigger the onActivity event handler.
my_cam.onActivity = function(isActive:Boolean) {
```

```
  // If isActive equals true, set the themeColor variable to "haloGreen".
  // Otherwise set the themeColor to "haloOrange".
  var themeColor:String = (isActive) ? "haloGreen" : "haloOrange";
  motion_pb.setStyle("themeColor", themeColor);
};

function changeMotionLevel() {
  // Set the motionLevel property for my_cam Camera instance to the value of the
  NumericStepper
  // component instance. Maintain the current motionTimeOut value of the my_cam
  Camera instance.
  my_cam.setMotionLevel(motionLevel_nstep.value, my_cam.motionTimeOut);
}
motionLevel_nstep.addEventListener("change", changeMotionLevel);
```

関連項目

onActivity (Camera.onActivity ハンドラ), onStatus (Camera.onStatus ハンドラ), setMotionLevel (Camera.setMotionLevel メソッド), activityLevel (Camera.activityLevel プロパティ)

motionTimeOut (Camera.motionTimeOut プロパティ)

public motionTimeOut : Number (読み取り専用)

カメラがモーション検知を停止してから、Camera.onActivity (false) が呼び出されるまでの時間 (ミリ秒単位) です。デフォルト値は 2000 (2 秒) です。

この値を設定するには、Camera.setMotionLevel() を使います。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、アクティビティレベルがモーションレベルより小さくなったときに ProgressBar インスタンスの Halo テーマの色を変更します。NumericStepper インスタンスを使用して motionTimeOut プロパティに秒数を設定します。[ライブラリ] オプションメニューから [新規ビデオ] を選択して新しいビデオインスタンスを作成します。インスタンスをステージに追加し、そのインスタンスに my_video という名前を付けます。Label コンポーネント、NumericStepper コンポーネント、および ProgressBar コンポーネントのインスタンスを 1 つずつステージに追加し、それぞれのインスタンスに motionLevel_lbl、motionTimeOut_nstep、および motion_pb という名前を付けます。次に、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var motionLevel_lbl:mx.controls.Label;
var motion_pb:mx.controls.ProgressBar;
var motionTimeOut_nstep:mx.controls.NumericStepper;
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);
```

```
this.onEnterFrame = function() {
  motionLevel_lbl.text = "activityLevel: "+my_cam.activityLevel;
};

motion_pb.indeterminate = true;
my_cam.onActivity = function(isActive:Boolean) {
  if (isActive) {
  motion_pb.setStyle("themeColor", "haloGreen");
  motion_pb.label = "Motion is above "+my_cam.motionLevel;
  } else {
  motion_pb.setStyle("themeColor", "haloOrange");
  motion_pb.label = "Motion is below "+my_cam.motionLevel;
  }
};
function changeMotionTimeOut() {
  my_cam.setMotionLevel(my_cam.motionLevel, motionTimeOut_nstep.value 1000);
}
motionTimeOut_nstep.addEventListener("change", changeMotionTimeOut);
motionTimeOut_nstep.value = my_cam.motionTimeOut/1000;
```

関連項目

setMotionLevel (Camera.setMotionLevel メソッド),
onActivity (Camera.onActivity ハンドラ)

muted (Camera.muted プロパティ)

public muted : Boolean (読み取り専用)

ユーザーが [Macromedia Flash Player 設定] パネルの [プライバシー] でカメラへのアクセスを禁止したか (true)、または許可したか (false) を示すブール値です。この値が変わると、Camera.onStatus が呼び出されます。詳細については、Camera.get() を参照してください。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、my_cam.muted が true に評価された場合にエラーメッセージを表示します。[ライブラリ] オプションメニューから [新規ビデオ] を選択して新しいビデオインスタンスを作成します。インスタンスをステージに追加し、そのインスタンスに my_video という名前を付けます。次に、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);
my_cam.onStatus = function(infoObj:Object) {
```

```
  if (my_cam.muted) {
  // If user is denied access to their Camera, you can display an error message
  here. You can display the user's Camera/Privacy settings again using
  System.showSettings(0);
  trace("User denied access to Camera");
  System.showSettings(0);
  }
};
```

関連項目

get (Camera.get メソッド),onStatus (Camera.onStatus ハンドラ)

name (Camera.name プロパティ)

public name : String (読み取り専用)

現在のカメラの名前を示すストリングです。カメラのハードウェアから返されるものです。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、デフォルトのカメラの名前をテキストフィールドに表示します。Windows の場合、この名前は [スキャナとカメラ] コントロールパネルに表示されるデバイス名と同じです。[ライブラリ] オプションメニューから [新規ビデオ] を選択して新しいビデオインスタンスを作成します。インスタンスをステージに追加し、そのインスタンスに my_video という名前を付けます。次に、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);

this.createTextField("name_txt", this.getNextHighestDepth(), 0, 0, 100, 22);
name_txt.autoSize = true;
name_txt.text = my_cam.name;
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

get (Camera.get メソッド),names (Camera.names プロパティ)

names (Camera.names プロパティ)

`public static names : Array` (読み取り専用)

[Macromedia Flash Player 設定] パネルの [プライバシー] を表示せずに、使用できるすべてのカメラの名前が含まれるストリング配列を取得します。この配列は ActionScript の他の配列と同じように動作します。この配列を使用して、それぞれのカメラのゼロから始まるインデックスと、システム上のカメラの数 (`Camera.names.length`) を調べることができます。詳細については、Array クラスの `Camera.names` を参照してください。

`Camera.names` プロパティが呼び出されると、ハードウェアを広範囲にわたって調べる必要があります。このため、配列を作成するまでに数秒間かかることがあります。ほとんどの場合は、デフォルトのカメラを使用できます。

メモ : 正しいシンタックスは `Camera.names` です。戻り値を変数に代入するには、*`cam_array`* `= Camera.names` のようなシンタックスを使用します。現在のカメラの名前を調べるには、*`active_cam`*`.name` を使用します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、デフォルトのカメラを使用します。ただし、利用できるカメラが複数ある場合は、デフォルトとして設定するカメラをユーザーが選択できます。カメラが 1 台しかない場合は、デフォルトのカメラが使用されます。[ライブラリ] オプションメニューから [新規ビデオ] を選択して新しいビデオインスタンスを作成します。インスタンスをステージに追加し、そのインスタンスに `my_video` という名前を付けます。次に、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var my_video:Video;
var cam_array:Array = Camera.names;
if (cam_array.length>1) {
  System.showSettings(3);
}
var my_cam:Camera = Camera.get();
my_video.attachVideo(my_cam);
```

関連項目

`get` (Camera.get メソッド), `index` (Camera.index プロパティ), `name` (Camera.name プロパティ)

onActivity (Camera.onActivity ハンドラ)

```
onActivity = function(active:Boolean) {}
```

カメラがモーションの検知を開始または停止すると、呼び出されます。このイベントハンドラに応答するには、アクティビティ値を処理する関数を作成する必要があります。

`Camera.onActivity(true)` を呼び出すのに必要なモーションの量、および `Camera.onActivity(false)` を呼び出すまでに経過する必要のある無活動時間の長さを指定するには、`Camera.setMotionLevel()` を使用します。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

`active:Boolean` - ブール値。カメラがモーションの検知を開始すると true、停止すると false に設定されます。

例

次の例では、カメラがモーション検知を開始または停止すると、[出力] パネルに true または false を表示します。

```
// Assumes a Video object named "myVideoObject" is on the Stage
active_cam = Camera.get();
myVideoObject.attachVideo(active_cam);
active_cam.setMotionLevel(10, 500);
active_cam.onActivity = function(mode)
{
  trace(mode);
}
```

関連項目

setMotionLevel (Camera.setMotionLevel メソッド)

onStatus (Camera.onStatus ハンドラ)

```
onStatus = function(infoObject: Object) {}
```

ユーザーがカメラへのアクセスを許可または禁止すると、呼び出されます。このイベントハンドラに応答するには、カメラによって生成される情報オブジェクトを処理する関数を作成する必要があります。

SWF ファイルがカメラにアクセスしようとすると、[プライバシー] ダイアログボックスが表示され、ユーザーはアクセスを許可するか禁止するかを選択できます。

- ユーザーがアクセスを許可すると、Camera.muted プロパティが false に設定されます。さらに、code プロパティが "Camera.Unmuted"、level プロパティが "Status" に設定された情報オブジェクトを使用して、このイベントハンドラが呼び出されます。
- ユーザーがアクセスを拒否すると、Camera.muted プロパティが true に設定されます。さらに、code プロパティが "Camera.Muted"、level プロパティが "Status" に設定された情報オブジェクトを使用して、このイベントハンドラが呼び出されます。

このイベントハンドラを使用せずに、ユーザーがカメラへのアクセスを許可したかどうかを判断するには、Camera.muted プロパティを使用します。

メモ: ユーザーが特定のドメインのすべての SWF ファイルに対するアクセスを永続的に許可または禁止した場合は、ユーザーが後でプライバシー設定を変更しない限り、そのドメインの SWF ファイルに対して、このハンドラは呼び出されません。詳細については、Camera.get() を参照してください。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

infoObject:Object - ステータスメッセージに従って定義されるパラメータ。

例

次の ActionScript は、ユーザーがカメラへのアクセスを許可または禁止したときに、メッセージを表示します。

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);
my_cam.onStatus = function(infoObj:Object) {
  switch (infoObj.code) {
  case 'Camera.Muted' :
  trace("Camera access is denied");
  break;
  case 'Camera.Unmuted' :
  trace("Camera access granted");
  break;
  }
}
```

関連項目

get (Camera.get メソッド), muted (Camera.muted プロパティ), showSettings (System.showSettings メソッド), onStatus (System.onStatus ハンドラ)

quality (Camera.quality プロパティ)

public quality : Number (読み取り専用)

必要な画質レベルを指定する整数です。各ビデオフレームに適用される圧縮率によって決まります。指定できる品質値は 1 (最低品質、最大圧縮率) から 100 (最高品質、圧縮なし) までです。デフォルト値は 0 です。これは、使用できる帯域幅を超えることがないように、画質が必要に応じて変更されることを示します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、NumericStepper インスタンスを使用して、カメラフィードに適用される圧縮率を指定します。[ライブラリ] オプションメニューから [新規ビデオ] を選択して新しいビデオインスタンスを作成します。インスタンスをステージに追加し、そのインスタンスに my_video という名前を付けます。NumericStepper インスタンスを追加し、そのインスタンスに quality_nstep という名前を付けます。次に、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var quality_nstep:mx.controls.NumericStepper;

var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);

quality_nstep.minimum = 0;
quality_nstep.maximum = 100;
quality_nstep.stepSize = 5;
quality_nstep.value = my_cam.quality;

function changeQuality() {
  my_cam.setQuality(my_cam.bandwidth, quality_nstep.value);
}
quality_nstep.addEventListener("change", changeQuality);
```

関連項目

setQuality (Camera.setQuality メソッド)

setMode (Camera.setMode メソッド)

```
public setMode([width:Number], [height:Number], [fps:Number],
  [favorArea:Boolean]) : Void
```

カメラのキャプチャモードを指定の要件に最も近いネイティブモードに設定します。指定したすべてのパラメータに一致するネイティブモードがカメラにない場合は、要求したモードに最も近いキャプチャモードが選択されます。これにより、画像の一部が切り取られたり、フレームが削除される可能性があります。

デフォルトでは、画像のサイズを維持するために、必要に応じてフレームが削除されます。削除されるフレームの数を最小限に抑えるには、favorArea に false を指定します。ただし、これによって画像のサイズは小さくなります。

ネイティブモードを選択した場合は、指定した縦横比ができる限り維持されます。たとえば、*active_cam*.setMode(400, 400, 30) コマンドを実行し、そのカメラで利用できる最大の幅と高さが 320 および 288 である場合は、幅と高さがともに 288 に設定されます。幅と高さを同じ値に設定することで、要求された 1:1 の縦横比が維持されます。

要求した値に最も近いモードが選択された後に、割り当てられた幅、高さ、ビデオキャプチャレートを調べるには、Camera.width、Camera.height、および Camera.fps を使用します。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

width:Number (オプション) - 必要なキャプチャの幅 (ピクセル単位)。デフォルト値は 160 です。

height:Number (オプション) - 必要なキャプチャの高さ (ピクセル単位)。デフォルト値は 120 です。

fps:Number (オプション) - 必要なデータキャプチャレート (1 秒あたりのフレーム数)。デフォルト値は 15 です。

favorArea:Boolean (オプション) - 指定要件に合うネイティブモードがカメラにない場合に、幅、高さ、フレームレートをどのように操作するかを指定するブール値。デフォルト値は true で、キャプチャサイズを維持することを示します。このパラメータを使用すると、*width* 値と *height* 値に最も近いモードが選択されます。ただし、これによりフレームレートが低下し、パフォーマンスに悪影響を及ぼすことがあります。カメラの高さと幅よりも最大フレームレートを優先するには、favorArea パラメータに false を指定します。

例

次の例では、カメラのキャプチャモードを設定します。TextInput インスタンスにフレームレートを入力し、Enter キーまたは Return キーを押すと、フレームレートを適用できます。[ライブラリ] オプションメニューから [新規ビデオ] を選択して新しいビデオインスタンスを作成します。インスタンスをステージに追加し、そのインスタンスに my_video という名前を付けます。TextInput コンポーネントのインスタンスを追加し、そのインスタンスに fps_ti という名前を付けます。次に、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);

fps_ti.maxChars = 2;
fps_ti.restrict = [0-9];
fps_lbl.text = "Current: "+my_cam.fps+" fps";

function changeFps():Void {
  my_cam.setMode(my_cam.width, my_cam.height, fps_ti.text);
  fps_lbl.text = "Current: "+my_cam.fps+" fps";
  fps_ti.text = my_cam.fps;
  Selection.setSelection(0,2);
}
fps_ti.addEventListener("enter", changeFps);
```

関連項目

fps (Camera.fps プロパティ), height (Camera.height プロパティ), width (Camera.width プロパティ), currentFps (Camera.currentFps プロパティ)

setMotionLevel (Camera.setMotionLevel メソッド)

```
public setMotionLevel([motionLevel:Number], [timeOut:Number]) : Void
```

Camera.onActivity(true) を呼び出すのに必要なモーション量を指定します。最後のモーションから、モーションが停止したと見なして Camera.onActivity(false) を呼び出すまでの時間をミリ秒単位で設定することもできます。

メモ : sensitivity パラメータの値に関係なく、ビデオを表示できます。このパラメータは、ビデオを実際にキャプチャまたは表示するかどうかではなく、Camera.onActivity を呼び出すタイミングと状況のみを決定します。

- カメラがモーションをまったく検知しないようにするには、*sensitivity* に 100 を指定します。この場合、Camera.onActivity は呼び出されません。この値は通常、テスト目的にのみ使用します。たとえば、Camera.onActivity が呼び出されたときに発生するように設定したすべてのアクションを一時的に無効にする場合などに使用します。

- カメラが現在検知しているモーションの量を調べるには、Camera.activityLevel プロパティを使います。

  モーションの精度 (sensitivity) の値がアクティビティの値に直接対応します。モーションがまったく存在しない場合、アクティビティ値は 0 です。継続的にモーションが発生している場合、アクティビティ値は 100 です。移動していない場合、アクティビティ値はモーション精度値よりも低くなります。移動している場合、アクティビティ値は頻繁にモーション精度値を超えます。

  このメソッドの目的は Microphone.setSilenceLevel() に似ています。どちらのメソッドも、onActivity イベントハンドラを呼び出すタイミングを指定するために使用します。ただし、パブリッシュするストリームに対する影響という点では、この 2 つのメソッドは大きく異なります。
- Microphone.setSilenceLevel() は帯域幅を最適化するように設計されています。オーディオストリームが無音と考えられる場合には、オーディオデータは送信されません。代わりに、無音状態が始まったことを示すメッセージが送信されます。
- Camera.setMotionLevel() はモーションを検知し、使用する帯域幅には影響しないように設計されています。ビデオストリームでモーションが検知されない間も、ビデオは送信されます。

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

motionLevel: Number (オプション) - Camera.onActivity(true) を呼び出すのに必要なモーション量を指定する数値。指定できる値は 0 ～ 100 です。デフォルト値は 50 です。

timeOut: Number (オプション) - 最後のモーションから、モーションが停止したと判断して Camera.onActivity(false) イベントハンドラを呼び出すまでの時間をミリ秒単位で指定する数値パラメータ。デフォルト値は 2000 (2 秒) です。

例

次の例では、ビデオアクティビティが開始または停止したときに、[出力] パネルにメッセージを送信します。ここではモーション精度値として 30 を使用していますが、これを別の値に変更することで、値の変化がモーション検知に与える影響を確認できます。

```
// Assumes a Video object named "myVideoObject" is on the Stage
active_cam = Camera.get();
x = 0;
function motion(mode) {
  trace(x + ": " + mode);
  x++;
}
active_cam.onActivity = function(mode) {
  motion(mode);
}
active_cam.setMotionLevel(30, 500);
myVideoObject.attachVideo(active_cam);
```

関連項目

motionLevel (Camera.motionLevel プロパティ), motionTimeOut (Camera.motionTimeOut プロパティ), onActivity (Camera.onActivity ハンドラ), activityLevel (Camera.activityLevel プロパティ)

setQuality (Camera.setQuality メソッド)

```
public setQuality([bandwidth:Number], [quality:Number]) : Void
```

現在の送信ビデオフィードの 1 秒あたりの最大帯域幅、または必要な画質を設定します。一般に、このメソッドは、Flash Media Server を使ってビデオを送信している場合にだけ使用します。

このメソッドを使用して、送信ビデオフィードのエレメントとして帯域幅と画質のどちらがそのアプリケーションにとって重要かを指定します。

- 帯域幅の使用量を優先する場合は、*bandwidth* に数値を、*frameQuality* に 0 を指定します。こうすると、指定した帯域幅内で最高品質のビデオが送信されます。必要であれば、指定の帯域幅を超えるのを防ぐために、画質が下げられます。一般には、モーションが増えるほど、画質は下がります。
- 画質を優先する場合は、*bandwidth* に 0 を、*frameQuality* に数値を指定します。こうすると、指定した画質を維持するために必要なだけの帯域幅が使われます。必要であれば、画質を維持するためにフレームレートが下げられます。一般には、モーションが増えるほど、使用する帯域幅も増加します。
- 帯域幅と品質の両方の重要性が同程度である場合は、両方のパラメータに数値を指定します。指定の品質を達成でき、指定の帯域幅を超えない範囲のビデオが送信されます。必要であれば、指定の帯域幅を超えることなく画質を維持するために、フレームレートが下げられます。

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

`bandwidth:Number` (オプション) - 現在の送信ビデオフィードが使用できる最大帯域幅 (1 秒あたりのバイト数) を指定する整数。*frameQuality* の値を維持するために必要な帯域幅を Flash ビデオで使用するには、*bandwidth* に 0 を指定します。デフォルト値は 16384 です。

`quality:Number` (オプション) - 必要な画質レベルを指定する整数。各ビデオフレームに適用される圧縮率で決まります。指定できる値は 1 (最低品質、最大圧縮率) から 100 (最高品質、圧縮なし) までです。帯域幅を超過するのを避けるために必要に応じて画質を変更するには、*quality* に 0 を指定します。デフォルト値は 0 です。

例

次の例では、このメソッドを使用して帯域幅と画質を制御する方法を示します。

```
// Ensure that no more than 8192 (8K/second) is used to send video
active_cam.setQuality(8192,0);

// Ensure that no more than 8192 (8K/second) is used to send video
// with a minimum quality of 50
active_cam.setQuality(8192,50);

// Ensure a minimum quality of 50, no matter how much bandwidth it takes
active_cam.setQuality(0,50);
```

関連項目

get (Camera.get メソッド), quality (Camera.quality プロパティ),
bandwidth (Camera.bandwidth プロパティ)

width (Camera.width プロパティ)

public width : Number (読み取り専用)

現在のキャプチャの幅 (ピクセル単位)。このプロパティに希望の値を設定するには、Camera.setMode() メソッドを使用します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次のコードは、ビデオインスタンスの現在の幅、高さ、および FPS をステージ上の Label コンポーネントインスタンスに表示します。[ライブラリ] オプションメニューから [新規ビデオ] を選択して新しいビデオインスタンスを作成します。インスタンスをステージに追加し、そのインスタンスに my_video という名前を付けます。Label コンポーネントのインスタンスをステージに追加し、そのインスタンスに dimensions_lbl という名前を付けます。次に、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);
var dimensions_lbl:mx.controls.Label;
dimensions_lbl.setStyle("fontSize", 9);
dimensions_lbl.setStyle("fontWeight", "bold");
dimensions_lbl.setStyle("textAlign", "center");
dimensions_lbl.text = "width: "+my_cam.width+", height: "+my_cam.height+", FPS:
  "+my_cam.fps;
```

Camera.setMode() の例も参照してください。

関連項目

height (Camera.height プロパティ),setMode (Camera.setMode メソッド)

capabilities (System.capabilities)

```
Object
  |
  +-System.capabilities
```

```
public class capabilities
extends Object
```

Capabilities クラスを使用すると、SWF ファイルのホストであるシステムと Flash Player の機能を確認できるので、コンテンツをさまざまな形式に合わせることができます。たとえば、携帯電話の画面 (モノクロ、100 x 100 ピクセル) はコンピュータの画面 (カラー、1000 x 1000 ピクセル) とは異なります。できるだけ多くのユーザーに適切なコンテンツを提供するため、System.capabilities オブジェクトを使用して、各ユーザーが使用しているデバイスのタイプを確認できます。デバイスの機能に応じて異なる SWF ファイルを送るようにサーバーに指示したり、デバイスの機能に応じて表示形式を変更するように SWF ファイルに指示したりできます。

機能情報は、HTTP の GET メソッドまたは POST メソッドを使用して送信できます。次の例は、MP3 対応で解像度が 1600 x 1200 ピクセルであるコンピュータにおいて Windows XP と Flash Player 8 (8.0.0.0) が動作している場合のサーバーストリングです。

```
A=t&SA=t&SV=t&EV=t&MP3=t&AE=t&VE=t&ACC=f&PR=t&SP=t&

SB=f&DEB=t&V=WIN%208%2C0%2C0%2C0&M=Macromedia%20Windows&

R=1600x1200&DP=72&COL=color&AR=1.0&OS=Windows%20XP&

L=en&PT=External&AVD=f&LFD=f&WD=f"
```

System.capabilities オブジェクトのプロパティはすべて読み取り専用です。

対応バージョン : ActionScript 1.0、Flash Player 6

プロパティ一覧

オプション	プロパティ	説明
static	avHardwareDisable: Boolean (読み取り専用)	ユーザーのカメラとマイクへのアクセスが管理上禁止されているか (true)、あるいは許可されているか (false) を示すブール値です。
static	hasAccessibility: Boolean (読み取り専用)	Flash Player が実行されている環境が、Flash Player とアクセシビリティ補助との間の通信をサポートしている場合は true、サポートしていない場合は false になるブール値です。
static	hasAudio:Boolean (読み取り専用)	Flash Player が実行されているシステムにオーディオ機能があるかどうかを指定します。
static	hasAudioEncoder: Boolean (読み取り専用)	Flash Player がオーディオストリームをエンコードできるかどうかを示します。
static	hasEmbeddedVideo: Boolean (読み取り専用)	Flash Player が実行されているシステムが埋め込みビデオをサポートしている場合は true、サポートしない場合は false になるブール値です。
static	hasIME:Boolean (読み取り専用)	システムに IME (Input Method Editor : 入力方式エディタ) がインストールされているかどうかを示します。
static	hasMP3:Boolean (読み取り専用)	システムに MP3 デコーダがあるかどうかを示します。
static	hasPrinting:Boolean (読み取り専用)	Flash Player が実行されているシステムが印刷機能をサポートしている場合は true、サポートしない場合は false になるブール値です。
static	hasScreenBroadcast: Boolean (読み取り専用)	Flash Media Server を通じて実行されるスクリーンブロードキャストアプリケーションの開発を Flash Player がサポートしている場合は true、サポートしていない場合は false になるブール値です。
static	hasScreenPlayback: Boolean (読み取り専用)	Flash Media Server を通じて実行されるスクリーンブロードキャストアプリケーションの再生を Flash Player がサポートしている場合は true、サポートしていない場合は false になるブール値です。
static	hasStreamingAudio: Boolean (読み取り専用)	プレーヤーがストリーミングオーディオを再生できる場合は true、再生できない場合は false になるブール値です。
static	hasStreamingVideo: Boolean (読み取り専用)	プレーヤーがストリーミングビデオを再生できる場合は true、再生できない場合は false になるブール値です。

オプション	プロパティ	説明
static	hasVideoEncoder: Boolean (読み取り専用)	Flash Player がビデオストリームをエンコードできるかどうかを示します。
static	isDebugger:Boolean (読み取り専用)	Flash Player が正式にリリースされたバージョンか (false)、それともデバッグ用の特別なバージョンか (true) を示すブール値です。
static	language:String (読み取り専用)	Flash Player が実行されているシステムの言語を示します。
static	localFileReadDisable: Boolean (読み取り専用)	ユーザーのハードディスクへの読み取りアクセスが管理上禁止されているか (true)、許可されているか (false) を示すブール値です。
static	manufacturer: String (読み取り専用)	Flash Player の製造元を示すストリング。" Adobe *OSName*" という形式で、*OSName* には " Windows"、" Macintosh"、 "Linux"、または " Other OS Name" が入ります。
static	os:String (読み取り専用)	現在のオペレーティングシステムを示すストリングです。
static	pixelAspectRatio: Number (読み取り専用)	画面のピクセル縦横比を示す整数です。
static	playerType:String (読み取り専用)	Flash Player のタイプを示すストリングです。
static	screenColor:String (読み取り専用)	画面の色を示すストリングです。
static	screenDPI:Number (読み取り専用)	画面の dpi (1 インチあたりのドット数) 解像度をピクセル単位で示した数値です。
static	screenResolutionX: Number (読み取り専用)	画面の最大水平解像度を示す整数です。
static	screenResolutionY: Number (読み取り専用)	画面の最大垂直解像度を示す整数です。
static	serverString:String (読み取り専用)	それぞれの System.capabilities プロパティの値を示す URL エンコードされたストリングです。
static	version:String (読み取り専用)	Flash Player のプラットフォームとバージョンの情報を含むストリング。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

メソッド一覧

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

avHardwareDisable (capabilities.avHardwareDisable プロパティ)

public static avHardwareDisable : Boolean (読み取り専用)

ユーザーのカメラとマイクへのアクセスが管理上禁止されているか (true)、あるいは許可されているか (false) を示すブール値です。サーバーストリングは AVD です。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.avHardwareDisable);
```

関連項目

get (Camera.get メソッド), get (Microphone.get メソッド), showSettings (System.showSettings メソッド)

hasAccessibility (capabilities.hasAccessibility プロパティ)

public static hasAccessibility : Boolean (読み取り専用)

Flash Player が実行されている環境が、Flash Player とアクセシビリティ補助との間の通信をサポートしている場合は true、サポートしていない場合は false になるブール値です。サーバーストリングは ACC です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.hasAccessibility);
```

関連項目

isActive (Accessibility.isActive メソッド), updateProperties (Accessibility.updateProperties メソッド), _accProps プロパティ

hasAudio (capabilities.hasAudio プロパティ)

public static hasAudio : Boolean (読み取り専用)

Flash Player が実行されているシステムにオーディオ機能があるかどうかを指定します。Flash Player では、このプロパティは常に true です。サーバーストリングは A です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.hasAudio);
```

hasAudioEncoder (capabilities.hasAudioEncoder プロパティ)

`public static hasAudioEncoder : Boolean ( 読み取り専用 )`

Flash Player がオーディオストリームをエンコードできるかどうかを示します。Flash Player がオーディオストリーム (マイクからのストリームなど) をエンコードできる場合は `true`、エンコードできない場合は `false` になるブール値です。サーバーストリングは `AE` です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.hasAudioEncoder);
```

hasEmbeddedVideo (capabilities.hasEmbeddedVideo プロパティ)

`public static hasEmbeddedVideo : Boolean ( 読み取り専用 )`

Flash Player が実行されているシステムが埋め込みビデオをサポートしている場合は `true`、サポートしない場合は `false` になるブール値です。サーバーストリングは `EV` です。

対応バージョン : ActionScript 1.0、Flash Player 6,0,65,0

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.hasEmbeddedVideo);
```

hasIME (capabilities.hasIME プロパティ)

`public static hasIME : Boolean ( 読み取り専用 )`

システムに IME (Input Method Editor : 入力方式エディタ) がインストールされているかどうかを示します。Flash Player が実行されているシステムに IME がインストールされている場合は `true` 値、IME がインストールされていない場合は `false` 値になります。サーバーストリングは `IME` です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、Flash Player が実行されているシステムに IME がインストールされている場合に、IME を ALPHANUMERIC_FULL に設定します。

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());
  System.IME.setConversionMode(System.IME.ALPHANUMERIC_FULL);
  trace(System.IME.getConversionMode());
}
```

hasMP3 (capabilities.hasMP3 プロパティ)

```
public static hasMP3 : Boolean ( 読み取り専用 )
```

システムに MP3 デコーダがあるかどうかを示します。Flash Player が実行されているシステムに MP3 デコーダがある場合は true、MP3 デコーダがない場合は false になるブール値です。サーバーストリングは MP3 です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.hasMP3);
```

hasPrinting (capabilities.hasPrinting プロパティ)

```
public static hasPrinting : Boolean ( 読み取り専用 )
```

Flash Player が実行されているシステムが印刷機能をサポートしている場合は true、サポートしない場合は false になるブール値です。サーバーストリングは PR です。

対応バージョン : ActionScript 1.0、Flash Player 6,0,65,0

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.hasPrinting);
```

hasScreenBroadcast (capabilities.hasScreenBroadcast プロパティ)

`public static hasScreenBroadcast : Boolean ( 読み取り専用 )`

Flash Media Server を通じて実行されるスクリーンブロードキャストアプリケーションの開発を Flash Player がサポートしている場合は `true`、サポートしていない場合は `false` になるブール値です。サーバーストリングは `SB` です。

対応バージョン : ActionScript 1.0、Flash Player 6,0,79,0

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.hasScreenBroadcast);
```

hasScreenPlayback (capabilities.hasScreenPlayback プロパティ)

`public static hasScreenPlayback : Boolean ( 読み取り専用 )`

Flash Media Server を通じて実行されるスクリーンブロードキャストアプリケーションの再生を Flash Player がサポートしている場合は `true`、サポートしていない場合は `false` になるブール値です。サーバーストリングは `SP` です。

対応バージョン : ActionScript 1.0、Flash Player 6,0,79,0

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.hasScreenPlayback);
```

hasStreamingAudio (capabilities.hasStreamingAudio プロパティ)

`public static hasStreamingAudio : Boolean ( 読み取り専用 )`

プレーヤーがストリーミングオーディオを再生できる場合は `true`、再生できない場合は `false` になるブール値です。サーバーストリングは `SA` です。

対応バージョン : ActionScript 1.0、Flash Player 6,0,65,0

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.hasStreamingAudio);
```

hasStreamingVideo (capabilities.hasStreamingVideo プロパティ)

`public static hasStreamingVideo : Boolean ( 読み取り専用 )`

プレーヤーがストリーミングビデオを再生できる場合は `true`、再生できない場合は `false` になるブール値です。サーバーストリングは `SV` です。

対応バージョン : ActionScript 1.0、Flash Player 6,0,65,0

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.hasStreamingVideo);
```

hasVideoEncoder (capabilities.hasVideoEncoder プロパティ)

`public static hasVideoEncoder : Boolean ( 読み取り専用 )`

Flash Player がビデオストリームをエンコードできるかどうかを示します。Flash Player がビデオストリーム (Web カメラからのストリームなど) をエンコードできる場合は `true`、エンコードできない場合は `false` になるブール値です。サーバーストリングは `VE` です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.hasVideoEncoder);
```

isDebugger (capabilities.isDebugger プロパティ)

`public static isDebugger : Boolean ( 読み取り専用 )`

Flash Player が正式にリリースされたバージョンか (`false`)、それともデバッグ用の特別なバージョンか (`true`) を示すブール値です。サーバーストリングは `DEB` です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.isDebugger);
```

language (capabilities.language プロパティ)

`public static language : String` (読み取り専用)

Flash Player が実行されているシステムの言語を示します。このプロパティは、ISO 639-1 による小文字 2 文字の言語コードで指定されます。中国語については、簡体字と繁体字を識別するために ISO 3166 による大文字 2 文字の国コードサブタグが追加されます。言語自体の名前には英語タグが使用されます。たとえば、`fr` はフランス語を示します。

このプロパティは、Flash Player 7 で 2 とおりの変更が加えられました。まず、英語版のシステムで言語コードに国コードが含まれなくなりました。Flash Player 6 の場合、すべての英語のシステムでは、言語コードと 2 文字の国コード (サブタグ) の組み合わせ `en-US` が返されます。Flash Player 7 の場合、英語のシステムでは、言語コード `en` だけが返されます。次に、Microsoft Windows システムの場合、このプロパティはユーザーインターフェイス (UI) 言語を返します。Microsoft Windows プラットフォーム上で Flash Player 6 を使用した場合、`System.capabilities.language` は、ユーザーロケールを返します。ユーザーロケールは、日付、時間、通貨などをフォーマットするための設定を制御します。Microsoft Windows プラットフォーム上で Flash Player 7 を使用した場合、このプロパティは UI 言語を返します。UI 言語は、すべてのメニュー、ダイアログボックス、エラーメッセージ、およびヘルプファイルなどで使用される言語を参照します。次の表に、指定できる値を示します。

言語	タグ
チェコ語	cs
デンマーク語	da
オランダ語	nl
英語	en
フィンランド語	fi
フランス語	fr
ドイツ語	de
ハンガリー語	hu
イタリア語	it
日本語	ja
韓国語	ko
ノルウェー語	no
その他 / 不明	xu
ポーランド語	pl
ポルトガル語	pt

言語	タグ
ロシア語	ru
簡体字中国語	zh-CN
スペイン語	es
スウェーデン語	sv
繁体字中国語	zh-TW
トルコ語	tr

対応バージョン： ActionScript 1.0、Flash Player 6 - Flash Player 7 ではビヘイビアが変更されました。

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.language);
```

localFileReadDisable (capabilities.localFileReadDisable プロパティ)

```
public static localFileReadDisable : Boolean ( 読み取り専用 )
```

ユーザーのハードディスクへの読み取りアクセスが管理上禁止されているか (`true`)、許可されているか (`false`) を示すブール値です。これが `true` に設定されている場合、Flash Player は、ファイル (Flash Player 起動時の最初の SWF ファイルを含む) をユーザーのハードディスクから読み取ることができません。たとえば、`XML.load()`、`LoadMovie()`、または `LoadVars.load()` を使用してユーザーのハードディスク上にあるファイルを読み取ろうとしても、このプロパティが `true` の場合は、読み取りに失敗します。

このプロパティが `true` に設定されていると、ランタイム共有ライブラリも読み取ることができません。ただし、ローカル共有オブジェクトは、このプロパティの値に関係なく読み取ることができます。サーバーストリングは `LFD` です。

対応バージョン： ActionScript 1.0、Flash Player 7

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.localFileReadDisable);
```

manufacturer (capabilities.manufacturer プロパティ)

public static manufacturer : String (読み取り専用)

Flash Player の製造元を示すストリング。"Adobe *OSName*" という形式で、*OSName* には "Windows"、"Macintosh"、"Linux"、または "Other OS Name" が入ります。サーバーストリングは M です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.manufacturer);
```

os (capabilities.os プロパティ)

public static os : String (読み取り専用)

現在のオペレーティングシステムを示すストリングです。os プロパティは、ストリング "Windows XP"、"Windows 2000"、"Windows NT"、"Windows 98/ME"、"Windows 95"、"Windows CE" (デスクトップバージョンの Flash Player ではなく SDK のみ)、"Linux"、および "MacOS" を返します。サーバーストリングは OS です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.os);
```

pixelAspectRatio (capabilities.pixelAspectRatio プロパティ)

public static pixelAspectRatio : Number (読み取り専用)

画面のピクセル縦横比を示す整数です。サーバーストリングは AR です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.pixelAspectRatio);
```

playerType (capabilities.playerType プロパティ)

```
public static playerType : String ( 読み取り専用 )
```

Flash Player のタイプを示すストリングです。このプロパティに指定できる値は、次のうちのいずれかです。

- "StandAlone" - スタンドアローン Flash Player
- "External" - 外部の Player またはムービープレビューモードで使用される Flash Player
- "PlugIn" - Flash Player ブラウザプラグイン
- "ActiveX" - Microsoft Internet Explorer で使用される Flash Player ActiveX コントロール

サーバーストリングは PT です。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.playerType);
```

screenColor (capabilities.screenColor プロパティ)

```
public static screenColor : String ( 読み取り専用 )
```

画面の色を示すストリングです。このプロパティの値は、"color"、"gray"、"bw" のいずれかになります (それぞれカラー、グレースケール、モノクロを表します)。サーバーストリングは COL です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.screenColor);
```

screenDPI (capabilities.screenDPI プロパティ)

```
public static screenDPI : Number ( 読み取り専用 )
```

画面の dpi (1 インチあたりのドット数) 解像度をピクセル単位で示した数値です。サーバーストリングは DP です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.screenDPI);
```

screenResolutionX (capabilities.screenResolutionX プロパティ)

```
public static screenResolutionX : Number ( 読み取り専用 )
```

画面の最大水平解像度を示す整数です。サーバーストリングは R で、画面の幅と高さの両方を返します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.screenResolutionX);
```

screenResolutionY (capabilities.screenResolutionY プロパティ)

```
public static screenResolutionY : Number ( 読み取り専用 )
```

画面の最大垂直解像度を示す整数です。サーバーストリングは R で、画面の幅と高さの両方を返します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.screenResolutionY);
```

serverString (capabilities.serverString プロパティ)

```
public static serverString : String ( 読み取り専用 )
```

それぞれの System.capabilities プロパティの値を示す URL エンコードされたストリングです。

URL エンコードストリングの例を次に示します。

```
A=t&SA=t&SV=t&EV=t&MP3=t&AE=t&VE=t&ACC=f&PR=t&SP=t&

 SB=f&DEB=t&V=WIN%208%2C0%2C0%2C0&M=Macromedia%20Windows&

 R=1600x1200&DP=72&COL=color&AR=1.0&OS=Windows%20XP&

 L=en&PT=External&AVD=f&LFD=f&WD=f
```

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.serverString);
```

version (capabilities.version プロパティ)

```
public static version : String ( 読み取り専用 )
```

Flash Player のプラットフォームとバージョンの情報を含むストリング。サーバーストリングは V です。バージョン番号の形式は "platform majorVersion, minorVersion, buildNumber, internalBuildNumber" です。platform の値は、"WIN"、"MAC"、または "UNIX" です。たとえば、次のようになります。

```
WIN 8,0,22,0 // Flash Player 8 for Windows
MAC 7,0,25,0 // Flash Player 7 for Macintosh
UNIX 5,0,55,0 // Flash Player 5 for UNIX
```

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、この読み取り専用プロパティの値をトレースします。

```
trace(System.capabilities.version);
```

Color

```
Object
  |
  +-Color
```

```
public class Color
extends Object
```

非推奨 Flash Player 8 以降では使用しないでください。Color クラスの代わりに flash.geom.ColorTransform クラスを使用します。

Color クラスを使用すると、ムービークリップの RGB カラー値とカラー変換を設定したり、設定した値を取得したりできます。

Color オブジェクトのメソッドを呼び出すには、コンストラクタ `new Color()` を使用して Color オブジェクトを作成する必要があります。

対応バージョン : ActionScript 1.0、Flash Player 5

プロパティ一覧

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター覧

署名	説明
Color(target:Object)	**このクラスは使用しないでください。** *target_mc* パラメータで指定されたムービークリップ用の Color オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	getRGB() : Number	**このクラスは使用しないでください。** Color オブジェクトで使用中の R+G+B 組み合わせを返します。
	getTransform() : Object	**このクラスは使用しないでください。** Color.setTransform() の前回の呼び出しで設定されたカラー変換情報を返します。
	setRGB(offset: Number) : Void	**このクラスは使用しないでください。** Color オブジェクトの RGB カラーを指定します。
	setTransform (transformObject: Object) : Void	**このクラスは使用しないでください。** Color オブジェクトのカラー変換情報を設定します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

Color コンストラクタ

public Color(target: Object)

非推奨 Flash Player 8 以降では使用しないでください。Color クラスの代わりに flash.geom.ColorTransform クラスを使用します。

target_mc パラメータで指定されたムービークリップ用の Color オブジェクトを作成します。作成した Color オブジェクトのメソッドを使用して、ターゲットムービークリップ全体の色を変更できます。

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

`target:Object` - ムービークリップのインスタンス名。

例

次の例では、ムービークリップ my_mc 用に my_color という Color オブジェクトを作成し、その RGB 値をオレンジ色に設定します。

```
var my_color:Color = new Color(my_mc);
my_color.setRGB(0xff9933);
```

getRGB (Color.getRGB メソッド)

```
public getRGB() : Number
```

非推奨 Flash Player 8 以降では使用しないでください。Color クラスの代わりに flash.geom.ColorTransform クラスを使用します。

Color オブジェクトで使用中の R+G+B 組み合わせを返します。

対応バージョン： ActionScript 1.0、Flash Player 5

戻り値

`Number` - 指定された色の RGB 値を表す数値。

例

次のコードでは、Color オブジェクト my_color の RGB 値を取得し、その値を 16 進ストリングに変換して、変数 myValue に代入します。このコードの動作を確認するには、ムービークリップのインスタンスをステージに追加し、そのインスタンスに my_mc という名前を付けます。

```
var my_color:Color = new Color(my_mc);
// set the color
my_color.setRGB(0xff9933);
var myValue:String = my_color.getRGB().toString(16);
// trace the color value
trace(myValue); // traces ff9933
```

関連項目

setRGB (Color.setRGB メソッド),rgb (ColorTransform.rgb プロパティ)

getTransform (Color.getTransform メソッド)

```
public getTransform() : Object
```

非推奨 Flash Player 8 以降では使用しないでください。Color クラスの代わりに flash.geom.ColorTransform クラスを使用します。

`Color.setTransform()` の前回の呼び出しで設定されたカラー変換情報を返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

`Object` - 指定された色に関する現在のオフセット値とパーセント値をプロパティとして持つオブジェクト。

例

次の例では、変換オブジェクトを取得し、現在の値を基準に my_mc のカラーおよびアルファの新しいパーセントを設定します。このコードの動作を確認するには、my_mc というインスタンス名のマルチカラーのムービークリップをステージに配置します。さらに、次のコードをメインタイムラインのフレーム 1 に追加し、[制御]-[ムービープレビュー] を選択します。

```
var my_color:Color = new Color(my_mc);
var myTransform:Object = my_color.getTransform();
myTransform = { ra: 50, ba: 50, aa: 30};
my_color.setTransform(myTransform);
```

カラー変換オブジェクトのパラメータの詳細については、`Color.setTransform()` を参照してください。

関連項目

`setTransform (Color.setTransform メソッド )`

setRGB (Color.setRGB メソッド)

```
public setRGB(offset:Number) : Void
```

非推奨 Flash Player 8 以降では使用しないでください。Color クラスの代わりに flash.geom.ColorTransform クラスを使用します。

Color オブジェクトの RGB カラーを指定します。このメソッドを呼び出すと、`Color.setTransform()` のそれまでの設定が上書きされます。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`offset:Number` - `0x` *RRGGBB* 設定される 16 進カラーまたは RGB カラー。*RR*、*GG*、および *BB* はそれぞれ、各カラー成分のオフセットを指定する 2 桁の 16 進数で構成されます。`0x` は、数値が 16 進数であることを ActionScript コンパイラに伝えるものです。

例

次の例では、ムービークリップ `my_mc` に RGB カラー値を設定します。このコードの動作を確認するには、`my_mc` というインスタンス名のムービークリップをステージに配置します。さらに、次のコードをメインタイムラインのフレーム 1 に追加し、[制御]-[ムービープレビュー] を選択します。

```
var my_color:Color = new Color(my_mc);
my_color.setRGB(0xFF0000); // my_mc turns red
```

関連項目

setTransform (Color.setTransform メソッド), rgb (ColorTransform.rgb プロパティ)

setTransform (Color.setTransform メソッド)

```
public setTransform(transformObject:Object) : Void
```

非推奨 Flash Player 8 以降では使用しないでください。Color クラスの代わりに flash.geom.ColorTransform クラスを使用します。

Color オブジェクトのカラー変換情報を設定します。*colorTransformObject* パラメータは、`new Object` コンストラクタで作成する汎用オブジェクトです。このオブジェクトには、カラーの赤、緑、青、およびアルファ (透明度) の各成分のパーセント値およびオフセット値を指定するパラメータがあります。値の入力形式は 0xRRGGBBAA です。

カラー変換オブジェクトのパラメータは、[拡張効果] ダイアログボックスの設定に対応しており、次のように定義します。

- *ra* は、赤の成分のパーセント (-100 ～ 100) です。
- *rb* は、赤の成分のオフセット (-255 ～ 255) です。
- *ga* は、緑の成分のパーセント (-100 ～ 100) です。
- *gb* は、緑の成分のオフセット (-255 ～ 255) です。
- *ba* は、青の成分のパーセント (-100 ～ 100) です。
- *bb* は、青の成分のオフセット (-255 ～ 255) です。
- *aa* は、アルファのパーセント (-100 ～ 100) です。
- *ab* は、アルファのオフセット (-255 ～ 255) です。

colorTransformObject パラメータは次のように作成します。

```
var myColorTransform:Object = new Object();
myColorTransform.ra = 50;
myColorTransform.rb = 244;
myColorTransform.ga = 40;
myColorTransform.gb = 112;
myColorTransform.ba = 12;
myColorTransform.bb = 90;
myColorTransform.aa = 40;
myColorTransform.ab = 70;
```

次のシンタックスを使用して、*colorTransformObject* パラメータを作成することもできます。

```
var myColorTransform:Object = { ra: 50, rb: 244, ga: 40, gb: 112, ba: 12, bb: 90,
  aa: 40, ab: 70}
```

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

transformObject:Object - new Object コンストラクタで作成したオブジェクト。Object クラスのこのインスタンスには、カラー変換値を指定するプロパティ ra、rb、ga、gb、ba、bb、aa、ab が必要です。これらのプロパティについては、上記の setTransform() メソッドの要約で説明されています。

例

次の例では、ターゲット SWF ファイル用に新しい Color オブジェクトを作成し、上で定義したプロパティを使用して汎用オブジェクト myColorTransform を作成します。次に、setTransform() メソッドを使用して *colorTransformObject* を Color オブジェクトに渡します。このコードを Flash (FLA) ドキュメントで使用するには、コードをメインタイムラインのフレーム 1 に追加し、my_mc というインスタンス名のムービークリップをステージに配置します。

```
// Create a color object called my_color for the target my_mc
var my_color:Color = new Color(my_mc);
// Create a color transform object called myColorTransform using
// Set the values for myColorTransform
var myColorTransform:Object = { ra: 50, rb: 244, ga: 40, gb: 112, ba: 12, bb: 90,
  aa: 40, ab: 70};
// Associate the color transform object with the Color object
// created for my_mc
my_color.setTransform(myColorTransform);
```

関連項目

Object

ColorMatrixFilter (flash.filters.ColorMatrixFilter)

```
Object
  |
  +-BitmapFilter
     |
     +-flash.filters.ColorMatrixFilter
```

```
public class ColorMatrixFilter
extends BitmapFilter
```

ColorMatrixFilter クラスを使用すると、入力イメージの各ピクセルの RGBA カラー値とアルファ値に 4 x 5 マトリックス変換を適用することで、新しい RGBA カラー値とアルファ値からなる結果を作成できます。これにより、彩度変更、色相回転、輝度アルファ変換など、各種効果を行うことができます。このフィルタは、ビットマップと MovieClip のインスタンスに適用できます。

フィルタの使い方は、フィルタの適用先オブジェクトによって異なります。

- 実行時にムービークリップにフィルタを適用する場合は、`filters` プロパティを使用します。オブジェクトの `filters` プロパティを設定してもオブジェクトは変更されません。また、`filters` プロパティをクリアすることで元に戻すことができます。
- BitmapData インスタンスにフィルタを適用するには、`BitmapData.applyFilter()` メソッドを使用します。BitmapData オブジェクトで `applyFilter()` を呼び出すことによって、ソース BitmapData オブジェクトとフィルタオブジェクトが取得され、フィルタを適用した結果として得られるイメージが生成されます。

イメージやビデオにもオーサリング時にフィルタ効果を適用できます。詳細については、オーサリングのマニュアルを参照してください。

ムービークリップやボタンにフィルタを適用する場合は、ムービークリップやボタンの `cacheAsBitmap` プロパティを `true` に設定します。すべてのフィルタをクリアすると、`cacheAsBitmap` は元の値に戻ります。

次の式が使用されます。ここで、a[0] ～ a[19] は 20 個の要素を持つ配列プロパティマトリックスのエントリ 0 ～ 19 に対応します。

```
redResult = a[0] * srcR + a[1] * srcG + a[2] * srcB + a[3] * srcA + a[4]
greenResult = a[5] * srcR + a[6] * srcG + a[7] * srcB + a[8] * srcA + a[9]
blueResult = a[10] * srcR + a[11] * srcG + a[12] * srcB + a[13] * srcA + a[14]
alphaResult = a[15] * srcR + a[16] * srcG + a[17] * srcB + a[18] * srcA + a[19]
```

このフィルタにより、元の各ピクセルが赤、緑、青、アルファの各成分 (srcR、srcG、srcB、srcA) に分解されます。最後の手順で、カラー成分が 1 つのピクセルに再び結合され、結果が出力されます。

この計算は、乗算されていないカラー値に対して実行します。入力グラフィックが乗算済みカラー値で構成される場合は、この処理のために、乗算済みのカラー値が乗算されていないカラー値に自動的に変換されます。

次の 2 つの最適化モードを使用できます。

アルファのみ。ここに示すようにアルファ成分のみを調整するマトリックスをフィルタに渡すと、フィルタによりそのパフォーマンスが最適化されます。

```
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 N 0 (where N is between 0.0 and 1.0)
```

高速バージョン。Pentium 3 以降、Apple G4 以降など、SSE/Altivec アクセラレータ対応のプロセッサでのみ使用できます。アクセラレータは、乗数項の範囲が -15.99 ～ 15.99 で、加算項 [4]、a[9]、a[14]、および a[19] の範囲が -8000 ～ 8000 の場合に使用されます。

結果として得られるイメージの幅または高さが 2880 ピクセルを超える場合、フィルタは適用されません。たとえば、フィルタが適用されたサイズの大きいムービークリップをズームインするとき、結果として得られるイメージが 2880 ピクセルの制限に達する場合は、フィルタがオフになります。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、BitmapFilter を使用して、マウスポインタの位置を基にイメージのカラーのサイズを操作します。ポインタを左上隅 (0,0) に置いた場合、イメージは変更されません。ポインタを右に移動するに従って、緑と青のチャンネルがともにイメージから削除されます。ポインタを下に移動するに従って、赤のチャンネルが削除されます。ポインタをステージの右下に置くと、イメージは完全な黒になります。この例では、リンケージ識別子が "YourImageLinkage" に設定されたイメージがライブラリにあると仮定しています。

```
import flash.filters.BitmapFilter;
import flash.filters.ColorMatrixFilter;

var image:MovieClip = this.attachMovie("YourImageLinkage", "YourImage",
  this.getNextHighestDepth());
image.cacheAsBitmap = true;

var listener:Object = new Object();
listener.image = image;
listener.onMouseMove = function() {
  var xPercent:Number = 1 - (_xmouse/Stage.width);
  var yPercent:Number = 1 - (_ymouse/Stage.height);
  var matrix:Array = new Array();
  matrix = matrix.concat([yPercent, 0, 0, 0, 0]); // red
  matrix = matrix.concat([0, xPercent, 0, 0, 0]); // green
  matrix = matrix.concat([0, 0, xPercent, 0, 0]); // blue
```

```
   matrix = matrix.concat([0, 0, 0, 1, 0]); // alpha

   var filter:BitmapFilter = new ColorMatrixFilter(matrix);
   image.filters = new Array(filter);
}

Mouse.addListener(listener);
listener.onMouseMove();
```

関連項目

getPixel (BitmapData.getPixel メソッド), applyFilter (BitmapData.applyFilter メソッド), filters (MovieClip.filters プロパティ), cacheAsBitmap (MovieClip.cacheAsBitmap プロパティ)

プロパティ一覧

オプション	プロパティ	説明
	matrix:Array	4 x 5 カラー変換用の 20 エレメントの配列です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター一覧

署名	説明
ColorMatrixFilter (matrix:Array)	指定されたパラメータで新しい ColorMatrixFilter インスタンスを初期化します。

メソッド一覧

オプション	署名	説明
	clone() : ColorMatrixFilter	このフィルタオブジェクトのコピーを返します。

BitmapFilter クラスから継承されるメソッド

clone (BitmapFilter.clone メソッド)

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

clone (ColorMatrixFilter.clone メソッド)

```
public clone() : ColorMatrixFilter
```

このフィルタオブジェクトのコピーを返します。

対応バージョン： ActionScript 1.0、Flash Player 8

戻り値

ColorMatrixFilter - 元のインスタンスと同じプロパティをすべて備えた新しい ColorMatrixFilter インスタンス。

例

次の例では、ColorMatrixFilter の新しいインスタンスを作成した後、clone メソッドでその複製を作成します。matrix プロパティは、たとえば clonedFilter.matrix[2] = 1; のようにして直接変更することはできません。このプロパティを変更するには、配列への参照を取得して変更を加えた後、clonedFilter.matrix = changedMatrix を使用して値をリセットする必要があります。

```
import flash.filters.ColorMatrixFilter;

var matrix:Array = new Array();
matrix = matrix.concat([1, 0, 0, 0, 0]); // red
matrix = matrix.concat([0, 1, 0, 0, 0]); // green
matrix = matrix.concat([0, 0, 1, 0, 0]); // blue
matrix = matrix.concat([0, 0, 0, 1, 0]); // alpha

var filter:ColorMatrixFilter = new ColorMatrixFilter(matrix);
trace("filter: " + filter.matrix);

var clonedFilter:ColorMatrixFilter = filter.clone();
matrix = clonedFilter.matrix;
matrix[2] = 1;
clonedFilter.matrix = matrix;
trace("clonedFilter: " + clonedFilter.matrix);
```

ColorMatrixFilter コンストラクタ

```
public ColorMatrixFilter(matrix:Array)
```

指定されたパラメータで新しい ColorMatrixFilter インスタンスを初期化します。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

`matrix:Array` - 4 x 5 のマトリックス構成を持つ 20 エレメントの配列です。

matrix (ColorMatrixFilter.matrix プロパティ)

```
public matrix : Array
```

4 x 5 カラー変換用の 20 エレメントの配列です。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、ColorMatrixFilter の新しいインスタンスを作成した後、その `matrix` プロパティを変更します。`matrix` プロパティは、たとえば `clonedFilter.matrix[2] = 1;` のようにして直接変更することはできません。このプロパティを変更するには、配列への参照を取得し、その参照を変更し、`clonedFilter.matrix = changedMatrix` を使用して値をリセットする必要があります。

```
import flash.filters.ColorMatrixFilter;

var matrix:Array = new Array();
matrix = matrix.concat([1, 0, 0, 0, 0]); // red
matrix = matrix.concat([0, 1, 0, 0, 0]); // green
matrix = matrix.concat([0, 0, 1, 0, 0]); // blue
matrix = matrix.concat([0, 0, 0, 1, 0]); // alpha

var filter:ColorMatrixFilter = new ColorMatrixFilter(matrix);
trace("filter: " + filter.matrix);
var changedMatrix:Array = filter.matrix;
changedMatrix[2] = 1;
filter.matrix = changedMatrix;
trace("filter: " + filter.matrix);
```

ColorTransform (flash.geom.ColorTransform)

```
Object
  |
  +-flash.geom.ColorTransform

public class ColorTransform
extends Object
```

ColorTransform クラスを使用すると、ムービークリップ内のすべてのカラー値を数学的に調整できます。カラー調整関数、つまりカラー変換は、赤、緑、青、アルファ透明度の 4 つのチャンネルすべてに適用できます。

ColorTransform オブジェクトをムービークリップに適用するときに、各カラーチャンネルの新しい値を算出する方法は次のとおりです。

- 新しい red 値 = (古い red 値 * `redMultiplier`)) + `redOffset`
- 新しい green 値 = (古い green 値 * `greenMultiplier`)) + `greenOffset`
- 新しい blue 値 = (古い blue 値 * `blueMultiplier`)) + `blueOffset`
- 新しい alpha 値 = (古い alpha 値 * `alphaMultiplier`)) + `alphaOffset`

算出後、カラーチャンネル値が 255 よりも大きい場合は 255 に設定されます。0 より小さい場合は 0 に設定されます。

ColorTransform オブジェクトのメソッドを呼び出すには、コンストラクタ `new ColorTransform()` を使用して ColorTransform オブジェクトを作成する必要があります。

カラー変換は、ムービークリップ (ロードした SWF オブジェクトなど) の背景色には適用されません。ムービークリップに割り当てられているグラフィックとシンボルにのみ適用されます。

対応バージョン： ActionScript 1.0、Flash Player 8

関連項目

colorTransform (Transform.colorTransform プロパティ)

プロパティー覧

オプション	プロパティ	説明
	alphaMultiplier: Number	アルファ透明度チャンネル値に乗算する 10 進数値です。
	alphaOffset: Number	アルファ透明度チャンネル値に alphaMultiplier 値を乗算した後に加算する数値です。数値の範囲は -255 ～ 255 です。
	blueMultiplier: Number	青チャンネル値に乗算する 10 進数値です。
	blueOffset:Number	青チャンネル値に blueMultiplier 値を乗算した後に加算する数値です。数値の範囲は -255 ～ 255 です。
	greenMultiplier: Number	緑チャンネル値に乗算する 10 進数値です。
	greenOffset:Number	緑チャンネル値に greenMultiplier 値を乗算した後に加算する数値です。数値の範囲は -255 ～ 255 です。
	redMultiplier:Number	赤チャンネル値に乗算する 10 進数値です。
	redOffset:Number	赤チャンネル値に redMultiplier 値を乗算した後に加算する数値です。数値の範囲は -255 ～ 255 です。
	rgb:Number	ColorTransform オブジェクトの RGB カラー値です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター覧

署名	説明
ColorTransform([redMultiplier: Number], [greenMultiplier:Number], [blueMultiplier:Number], [alphaMultiplier:Number], [redOffset:Number], [greenOffset:Number], [blueOffset:Number], [alphaOffset:Number])	指定されたカラーチャンネル値とアルファ値を持つ ColorTransform オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	concat(second: ColorTransform) : Void	第 2 の付加的なカラー変換をムービークリップに適用します。
	toString() : String	ColorTransform オブジェクトのすべてのプロパティが列挙されたストリングを書式設定して返します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

alphaMultiplier (ColorTransform.alphaMultiplier プロパティ)

public alphaMultiplier : Number

アルファ透明度チャンネル値に乗算する 10 進数値です。

MovieClip._alpha プロパティを使用してムービークリップのアルファ透明度の値を直接設定すると、そのムービークリップの ColorTransform オブジェクトの alphaMultiplier の値に影響します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、ColorTransform オブジェクト colorTrans を作成し、その alphaMultiplier 値を 1 から 0.5 に調整します。

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.alphaMultiplier); // 1

colorTrans.alphaMultiplier = 0.5;
trace(colorTrans.alphaMultiplier); // 0.5

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;
```

```
function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

関連項目

_alpha (MovieClip._alpha プロパティ)

alphaOffset (ColorTransform.alphaOffset プロパティ)

```
public alphaOffset : Number
```

アルファ透明度チャンネル値に alphaMultiplier 値を乗算した後に加算する数値です。数値の範囲は -255 ～ 255 です。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、ColorTransform オブジェクト colorTrans を作成し、その alphaOffset 値を 0 から -128 に調整します。

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.alphaOffset); // 0

colorTrans.alphaOffset = -128;
trace(colorTrans.alphaOffset); // -128

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
```

```
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

blueMultiplier (ColorTransform.blueMultiplier プロパティ)

```
public blueMultiplier : Number
```

青チャンネル値に乗算する 10 進数値です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、ColorTransform オブジェクト `colorTrans` を作成し、その `blueMultiplier` 値を 1 から 0.5 に調整します。

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.blueMultiplier); // 1

colorTrans.blueMultiplier = 0.5;
trace(colorTrans.blueMultiplier); // 0.5

var rect:MovieClip = createRectangle(20, 80, 0x0000FF);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

blueOffset (ColorTransform.blueOffset プロパティ)

```
public blueOffset : Number
```

青チャンネル値に blueMultiplier 値を乗算した後に加算する数値です。数値の範囲は -255 ～ 255 です。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、ColorTransform オブジェクト colorTrans を作成し、その blueOffset 値を 0 から 255 に調整します。

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.blueOffset); // 0

colorTrans.blueOffset = 255;
trace(colorTrans.blueOffset); // 255

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

ColorTransform コンストラクタ

```
public ColorTransform([redMultiplier:Number], [greenMultiplier:Number],
  [blueMultiplier:Number], [alphaMultiplier:Number], [redOffset:Number],
  [greenOffset:Number], [blueOffset:Number], [alphaOffset:Number])
```

指定されたカラーチャンネル値とアルファ値を持つ ColorTransform オブジェクトを作成します。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

`redMultiplier:Number` (オプション) - 赤の乗数の値 (0 ～ 1)。デフォルト値は 100 です。

`greenMultiplier:Number` (オプション) - 緑の乗数の値 (0 ～ 1)。デフォルト値は 100 です。

`blueMultiplier:Number` (オプション) - 青の乗数の値 (0 ～ 1)。デフォルト値は 100 です。

`alphaMultiplier:Number` (オプション) - アルファ透明度の乗数の値 (0 ～ 1)。デフォルト値は 100 です。

`redOffset:Number` (オプション) - 赤のカラーチャンネル値のオフセット (-255 ～ 255)。デフォルト値は 0 です。

`greenOffset:Number` (オプション) - 緑のカラーチャンネル値のオフセット (-255 ～ 255)。デフォルト値は 0 です。

`blueOffset:Number` (オプション) - 青のカラーチャンネル値のオフセット (-255 ～ 255)。デフォルト値は 0 です。

`alphaOffset:Number` (オプション) - アルファ透明度のカラーチャンネル値のオフセット (-255 ～ 255)。デフォルト値は 0 です。

例

次の例では、`greenTransform` という ColorTransform オブジェクトを作成します。

```
var greenTransform:flash.geom.ColorTransform = new
  flash.geom.ColorTransform(0.5, 1.0, 0.5, 0.5, 10, 10, 10, 0);
```

次の例では、デフォルトのコンストラクタ値で ColorTransfrom オブジェクト `colorTrans_1` を作成します。`colorTrans_1` と `colorTrans_2` で同じ値がトレースされれば、デフォルトのコンストラクタ値が使用されていることになります。

```
import flash.geom.ColorTransform;

var colorTrans_1:ColorTransform = new ColorTransform(1, 1, 1, 1, 0, 0, 0, 0);
trace(colorTrans_1);
//(redMultiplier=1, greenMultiplier=1, blueMultiplier=1, alphaMultiplier=1,
  redOffset=0, greenOffset=0, blueOffset=0, alphaOffset=0)
```

```
var colorTrans_2:ColorTransform = new ColorTransform();
trace(colorTrans_2);
//(redMultiplier=1, greenMultiplier=1, blueMultiplier=1, alphaMultiplier=1,
  redOffset=0, greenOffset=0, blueOffset=0, alphaOffset=0)
```

concat (ColorTransform.concat メソッド)

```
public concat(second:ColorTransform) : Void
```

第 2 の付加的なカラー変換をムービークリップに適用します。2 番目の変換パラメータのセットは、最初の変換の完了後、ムービークリップの色に適用されます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`second:ColorTransform` - 現在の ColorTransform オブジェクトと結合される 2 番目の ColorTransform オブジェクト。

例

次の例では、ColorTransform オブジェクト `colorTrans_2` を `colorTrans_1` に連結します。その結果、赤の成分のフルオフセットとアルファ透明度の乗数 0.5 が結合されます。

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans_1:ColorTransform = new ColorTransform(1, 1, 1, 1, 255, 0, 0, 0);
trace(colorTrans_1);
// (redMultiplier=1, greenMultiplier=1, blueMultiplier=1, alphaMultiplier=1,
  redOffset=255, greenOffset=0, blueOffset=0, alphaOffset=0)

var colorTrans_2:ColorTransform = new ColorTransform(1, 1, 1, 0.5, 0, 0, 0, 0);
trace(colorTrans_2);
// (redMultiplier=1, greenMultiplier=1, blueMultiplier=1, alphaMultiplier=0.5,
  redOffset=0, greenOffset=0, blueOffset=0, alphaOffset=0)

colorTrans_1.concat(colorTrans_2);
trace(colorTrans_1);
// (redMultiplier=1, greenMultiplier=1, blueMultiplier=1, alphaMultiplier=0.5,
  redOffset=255, greenOffset=0, blueOffset=0, alphaOffset=0)

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans_1;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
```

```
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

greenMultiplier (ColorTransform.greenMultiplier プロパティ)

```
public greenMultiplier : Number
```

緑チャンネル値に乗算する 10 進数値です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、ColorTransform オブジェクト colorTrans を作成し、その greenMultiplier 値を 1 から 0.5 に調整します。

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.greenMultiplier); // 1

colorTrans.greenMultiplier = 0.5;
trace(colorTrans.greenMultiplier); // 0.5

var rect:MovieClip = createRectangle(20, 80, 0x00FF00);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

greenOffset (ColorTransform.greenOffset プロパティ)

```
public greenOffset : Number
```

緑チャンネル値に greenMultiplier 値を乗算した後に加算する数値です。数値の範囲は -255 ～ 255 です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、ColorTransform オブジェクト colorTrans を作成し、その greenOffset 値を 0 から 255 に調整します。

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.greenOffset); // 0

colorTrans.greenOffset = 255;
trace(colorTrans.greenOffset); // 255

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

redMultiplier (ColorTransform.redMultiplier プロパティ)

```
public redMultiplier : Number
```

赤チャンネル値に乗算する 10 進数値です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、ColorTransform オブジェクト colorTrans を作成し、その redMultiplier 値を 1 から 0.5 に調整します。

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.redMultiplier); // 1

colorTrans.redMultiplier = 0.5;
trace(colorTrans.redMultiplier); // 0.5

var rect:MovieClip = createRectangle(20, 80, 0xFF0000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

redOffset (ColorTransform.redOffset プロパティ)

```
public redOffset : Number
```

赤チャンネル値に redMultiplier 値を乗算した後に加算する数値です。数値の範囲は -255 ～ 255 です。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、ColorTransform オブジェクト colorTrans を作成し、その redOffset 値を 0 から 255 に調整します。

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.redOffset); // 0

colorTrans.redOffset = 255;
trace(colorTrans.redOffset); // 255

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

rgb (ColorTransform.rgb プロパティ)

```
public rgb : Number
```

ColorTransform オブジェクトの RGB カラー値です。

このプロパティを設定すると、それに応じて 3 つのカラーオフセット値 (redOffset、greenOffset、blueOffset) が設定され、3 つのカラー乗数値 (redMultiplier、greenMultiplier、blueMultiplier) が 0 に設定されます。アルファ透明度の乗数値とオフセット値は変わりません。

このプロパティには、0x *RRGGBB* という形式の値を渡します。*RR* 、*GG* 、*BB* はそれぞれ、各カラー成分のオフセットを指定する 2 桁の 16 進数で構成されます。0x は、数値が 16 進数であることを ActionScript コンパイラに伝えるものです。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、ColorTransform オブジェクト colorTrans を作成し、その rgb 値を 0xFF0000. に調整します。

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.rgb); // 0

colorTrans.rgb = 0xFF0000;
trace(colorTrans.rgb); // 16711680
trace("0x" + colorTrans.rgb.toString(16)); // 0xff0000

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

toString (ColorTransform.toString メソッド)

```
public toString() : String
```

ColorTransform オブジェクトのすべてのプロパティが列挙されたストリングを書式設定して返します。

対応バージョン： ActionScript 1.0、Flash Player 8

戻り値

`String` - ColorTransform オブジェクトのすべてのプロパティを列挙するストリング。

例

次の例では、ColorTransform オブジェクト `colorTrans` を作成し、その `toSting()` メソッドを呼び出します。このメソッドにより、(redMultiplier=RM, greenMultiplier=GM, blueMultiplier=BM, alphaMultiplier=AM, redOffset=RO, greenOffset=GO, blueOffset=BO, alphaOffset=AO) の形式のストリングが得られます。

```
import flash.geom.ColorTransform;

var colorTrans:ColorTransform = new ColorTransform(1, 2, 3, 4, -255, -128, 128,
  255);
trace(colorTrans.toString());
// (redMultiplier=1, greenMultiplier=2, blueMultiplier=3, alphaMultiplier=4,
  redOffset=-255, greenOffset=-128, blueOffset=128, alphaOffset=255)
```

ContextMenu

```
Object
  |
  +-ContextMenu

public dynamic class ContextMenu
extends Object
```

ContextMenu クラスを使用すると、Flash Player のコンテキストメニューのアイテムを実行時に制御できます。このメニューは、ユーザーが Flash Player 内を右クリック (Windows) または Control キーを押しながらクリック (Macintosh) したときに表示されます。ContextMenu クラスのメソッドとプロパティを使用すると、カスタムメニューアイテムの追加、ビルトインコンテキストメニューアイテムの表示制御 ([ズームイン]、[プリント] など)、新しいメニューの作成を行うことができます。

ContextMenu オブジェクトは、特定のボタンやムービークリップ、テキストフィールドオブジェクト、またはムービー全体に関連付けることができます。その際、Button クラス、MovieClip クラス、TextField クラスの menu プロパティを使用します。`menu` プロパティの詳細については、`Button.menu`、`MovieClip.menu`、および `TextField.menu` を参照してください。

ContextMenu オブジェクトに新しいアイテムを追加するには、ContextMenuItem オブジェクトを作成して ContextMenu.customItems 配列に追加します。コンテキストメニューアイテムの作成方法については、ContextMenuItem クラスを参照してください。

Flash Player には 3 種類のコンテキストメニューがあります。Flash Player を右クリックしたときに表示される標準メニュー、選択可能テキストフィールドまたは編集可能テキストフィールドを右クリックしたときに表示される編集メニュー、Flash Player への SWF ファイルのロードが失敗したときに表示されるエラーメニューです。ContextMenu クラスで修正できるのは、標準メニューと編集メニューだけです。

カスタムメニューアイテムは、常に Flash Player コンテキストメニューの一番上に、つまり、どの可視ビルトインメニューアイテムよりも上に表示されます。ビルトインメニューアイテムとカスタムメニューアイテムの間にはセパレータが表示されます。コンテキストメニューに追加できるカスタムアイテムは 15 アイテム以内です。コンテキストメニューから [設定] メニューアイテムを削除することはできません。[設定] メニューアイテムは、ユーザーがプライバシーやコンピュータの記憶領域に作用する設定にアクセスできるようにするための、Flash に必須のメニューアイテムです。また、使用している Flash Player のバージョンをユーザーが確認するために必要な [Macromedia Flash Player 7 について] メニューアイテムもコンテキストメニューから削除できません。

ContextMenu オブジェクトのメソッドを呼び出す前に、コンストラクタ new ContextMenu() を使用して ContextMenu オブジェクトを作成する必要があります。

対応バージョン： ActionScript 1.0、Flash Player 7

関連項目

ContextMenuItem, menu (Button.menu プロパティ), menu (MovieClip.menu プロパティ), menu (TextField.menu プロパティ)

プロパティ一覧

オプション	プロパティ	説明
	builtInItems:Object	zoom、quality、play、loop、rewind、forward_back、print というブール値プロパティを持つオブジェクト。
	customItems:Array	ContextMenuItem オブジェクトの配列です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベントの一覧

イベント	説明
onSelect = function(item: Object, item_menu:Object) {}	ユーザーが Flash Player コンテキストメニューを呼び出したときに、メニューが実際に表示される前に呼び出されます。

コンストラクター覧

署名	説明
ContextMenu ([callbackFunction: Function])	新しい ContextMenu オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	copy() : ContextMenu	指定された ContextMenu オブジェクトのコピーを作成します。
	hideBuiltInItems() : Void	[設定] を除き、指定された ContextMenu オブジェクト内のすべてのビルトインメニューアイテムを非表示にします。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

builtInItems (ContextMenu.builtInItems プロパティ)

```
public builtInItems : Object
```

zoom、quality、play、loop、rewind、forward_back、print というブール値プロパティを持つオブジェクト。これらのプロパティを false に設定すると、対応するメニューが、指定した ContextMenu オブジェクトから削除されます。これらのプロパティは列挙することができ、デフォルトでは true に設定されています。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例では、SWF ファイルの現在のタイムラインにある ContextMenu オブジェクト my_cm のビルトインメニューアイテムのうち、[画質] と [プリント] を削除します。

```
var my_cm:ContextMenu = new ContextMenu ();
my_cm.builtInItems.quality=false;
my_cm.builtInItems.print=false;
this.menu = my_cm;
```

メモ : コンテキストメニューから [設定] または [Macromedia Flash Player 7 について] メニューアイテムを削除することはできません。

次の例では、for..in ループを使用して、ContextMenu オブジェクト my_cm のすべてのビルトインメニューアイテムの名前と値を順番に列挙しています。

```
var my_cm:ContextMenu = new ContextMenu();
for(eachProp in my_cm.builtInItems) {
  var propName = eachProp;
  var propValue = my_cm.builtInItems[propName];
  trace(propName + ": " + propValue);
}
```

ContextMenu コンストラクタ

```
public ContextMenu([callbackFunction:Function])
```

新しい ContextMenu オブジェクトを作成します。オブジェクトを作成する際に、イベントハンドラの識別子を指定することもできます。指定した関数は、ユーザーがコンテキストメニューを表示したときに、メニューが実際に表示されるよりも前に呼び出されます。これは、アプリケーションの状態や、ユーザーが右クリック (Windows) または Control キーを押しながらクリック (Macintosh) したオブジェクトの種類 (ムービークリップ、テキストフィールド、ボタンなど) またはタイムラインに応じて、メニューの内容をカスタマイズする場合に便利です。イベントハンドラの作成例については、ContextMenu.onSelect を参照してください。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

`callbackFunction:Function` (オプション) - ユーザーが右クリック (Windows の場合) または Control キーを押したままクリック (Macintosh の場合) したときに、メニューが表示される前に呼び出される関数への参照。

例

次の例では、コンテキストメニュー内のすべてのビルトインオブジェクトを非表示にします。ただし、[設定] と [Macromedia Flash Player 7 について] は無効にできません。

```
var newMenu:ContextMenu = new ContextMenu();
newMenu.hideBuiltInItems();
this.menu = newMenu;
```

次の例では、指定されたイベントハンドラ `menuHandler` がブール型変数 `showItem` の値に応じてカスタムメニューアイテム (`ContextMenu.customItems` 配列を使用) を有効または無効にします。この変数の値が `false` の場合は、カスタムメニューアイテムを無効にします。それ以外の場合は有効にします。

```
var showItem = true; // Change this to false to remove
var my_cm:ContextMenu = new ContextMenu(menuHandler);
my_cm.customItems.push(new ContextMenuItem("Hello", itemHandler));
function menuHandler(obj, menuObj) {
  if (showItem == false) {
  menuObj.customItems[0].enabled = false;
  } else {
  menuObj.customItems[0].enabled = true;
  }
}
function itemHandler(obj, item) {
  //...put code here...
  trace("selected!");
}
this.menu = my_cm;
```

ユーザーがステージ上で右クリック (Windows の場合) または Control キーを押したままクリック (Macintosh の場合) すると、カスタムメニューが表示されます。

関連項目

menu (Button.menu プロパティ), onSelect (ContextMenu.onSelect ハンドラ), customItems (ContextMenu.customItems プロパティ), hideBuiltInItems (ContextMenu.hideBuiltInItems メソッド), menu (MovieClip.menu プロパティ), menu (TextField.menu プロパティ)

copy (ContextMenu.copy メソッド)

```
public copy() : ContextMenu
```

指定された ContextMenu オブジェクトのコピーを作成します。コピーとして作成されたオブジェクトは、元のオブジェクトのすべてのプロパティを継承します。

対応バージョン : ActionScript 1.0、Flash Player 7

戻り値

`ContextMenu` - ContextMenu オブジェクト。

例

次の例では、ContextMenu オブジェクト my_cm を作成し、そのビルトインメニューアイテムを非表示にし、"Save..." というメニューアイテムを追加します。その後、my_cm のコピーを作成して変数 clone_cm に代入します。このコピーは、元のメニューのすべてのプロパティを継承します。

```
var my_cm:ContextMenu = new ContextMenu();
my_cm.hideBuiltInItems();
var menuItem_cmi:ContextMenuItem = new ContextMenuItem("Save...", saveHandler);
my_cm.customItems.push(menuItem_cmi);
function saveHandler(obj, menuItem) {
  // saveDocument();
  // custom function (not shown)
  trace("something");
}
clone_cm = my_cm.copy();
this.menu = my_cm;
for (var i in clone_cm.customItems) {
  trace("clone_cm-> "+clone_cm.customItems[i].caption);
}
for (var i in my_cm.customItems) {
  trace("my_cm-> "+my_cm.customItems[i].caption);
}
```

customItems (ContextMenu.customItems プロパティ)

```
public customItems : Array
```

ContextMenuItem オブジェクトの配列です。配列内の各オブジェクトは、定義したカスタムコンテキストメニューアイテムを表します。このプロパティを使用して、これらのカスタムメニューアイテムを追加、削除、変更することができます。

新しいメニューアイテムを作成するには、まず新しい ContextMenuItem オブジェクトを作成します。次に Array.push() などを使用して、作成したオブジェクトを *menu_mc* .customItems 配列に追加します。新しいメニューアイテムの作成方法については、ContextMenuItem クラスを参照してください。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例では、menuItem_cmi という新しいカスタムメニューアイテムを作成します。そのキャプションは "Send e-mail" で、コールバックハンドラの名前は emailHandler です。customItems 配列を使用して、この新しいメニューアイテムを ContextMenu オブジェクト my_cm に追加します。最後に、新しいメニューをムービークリップ email_mc に関連付けます。この例の動作を確認するには、ムービークリップインスタンスをステージに作成し、プロパティインスペクタを使用して、そのインスタンスに email_mc という名前を付けます。ムービープレビューモードで、マウスカーソルを email_mc クリップ上に移動し、コンテキストメニューを表示すると、この新しいコンテキストメニューアイテムが表示されます。

```
var my_cm:ContextMenu = new ContextMenu();
var menuItem_cmi:ContextMenuItem = new ContextMenuItem("Send e-mail",
  emailHandler);
my_cm.customItems.push(menuItem_cmi);
email_mc.menu = my_cm;
function emailHandler() {
  trace("sending email");
}
```

関連項目

menu (Button.menu プロパティ), menu (MovieClip.menu プロパティ),
menu (TextField.menu プロパティ), push (Array.push メソッド)

hideBuiltInItems (ContextMenu.hideBuiltInItems メソッド)

```
public hideBuiltInItems() : Void
```

[設定] を除き、指定された ContextMenu オブジェクト内のすべてのビルトインメニューアイテムを非表示にします。Flash Debug Player が実行されている場合、デバッグのメニューアイテムは表示されますが、リモートデバッグが有効化されていない SWF ファイルについてはグレー表示されます。

このメソッドは、標準コンテキストメニューに表示されるメニューアイテムだけを非表示にします。編集メニューやエラーメニューに表示されるメニューアイテムには影響しません。

このメソッドは、*my_cm* .builtInItems のすべてのブール型プロパティを false に設定することによって機能します。ビルトインアイテムを個別に表示するには、この後の例に示すように、*my_cm* .builtInItems のそのアイテムに対応するメンバーを true に設定します。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例では、新しい ContextMenu オブジェクト my_cm を作成します。[プリント] を除くビルトインメニューアイテムが非表示になります。このメニューオブジェクトは、現在のタイムラインに関連付けられています。

```
var my_cm:ContextMenu = new ContextMenu();
my_cm.hideBuiltInItems();
my_cm.builtInItems.print = true;
this.menu = my_cm;
```

onSelect (ContextMenu.onSelect ハンドラ)

```
onSelect = function(item:Object, item_menu:Object) {}
```

ユーザーが Flash Player コンテキストメニューを呼び出したときに、メニューが実際に表示される前に呼び出されます。このイベントハンドラを使用すると、現在のアプリケーションの状態に基づいて、コンテキストメニューの内容をカスタマイズできます。

新しい ContextMenu オブジェクトを作成する際に、ContextMenu オブジェクトのコールバックハンドラを指定することもできます。詳細については、ContextMenuItem の onSelect を参照してください。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

item:Object - コンテキストメニューが呼び出されたときにマウスの下にあったオブジェクト (ムービークリップ、ボタン、または選択可能テキストフィールド) への参照。このオブジェクトの menu プロパティは、有効な ContextMenu オブジェクトに設定されている必要があります。

item_menu:Object - object の menu プロパティに代入されている ContextMenu オブジェクトへの参照。

例

次の例では、コンテキストメニューが呼び出されたオブジェクトの種類を判定します。

```
my_cm:ContextMenu = new ContextMenu();
function menuHandler(obj:Object, menu:ContextMenu) {
  if(obj instanceof MovieClip) {
    trace("Movie clip: " + obj);
  }
  if(obj instanceof TextField) {
    trace("Text field: " + obj);
  }
  if(obj instanceof Button) {
    trace("Button: " + obj);
  }
}
my_cm.onSelect = menuHandler;
my_mc.menu = my_cm;
my_btn.menu = my_cm;
```

ContextMenuItem

```
Object
  |
  +-ContextMenuItem
```

```
public dynamic class ContextMenuItem
extends Object
```

ContextMenuItem クラスを使用すると、コンテキストメニューに表示するカスタムメニューアイテムを作成できます。各 ContextMenuItem オブジェクトには、コンテキストメニュー内に表示されるキャプション (テキスト) と、そのメニューアイテムが選択されたときに呼び出されるコールバックハンドラ (関数) があります。コンテキストメニューに新しいアイテムを追加するには、ContextMenu オブジェクトの customItems 配列にアイテムを追加します。

特定のメニューアイテムを有効または無効にすることや、表示または非表示にしたり、メニューアイテムに関連付けられているキャプションやコールバックハンドラを変更したりできます。

カスタムメニューアイテムは、コンテキストメニューの一番上に、つまり、どのビルトインアイテムよりも上に表示されます。カスタムメニューアイテムとビルトインアイテムとの間には常にセパレータが表示されます。コンテキストメニューに追加できるカスタムアイテムは 15 アイテム以内です。各アイテムには、少なくとも 1 つの表示可能な文字が含まれている必要があります。制御文字、改行、その他の空白文字は無視されます。アイテムの長さは 100 文字以内でなければなりません。ビルトインメニューアイテムまたは他のカスタムメニューアイテムと同じアイテムは、一致するアイテムが表示されるかどうかにかかわらず、無視されます。メニューアイテムを比較する際、大文字と小文字、カンマとピリオド、空白文字は無視されます。

以下の単語は、単独でも他の単語と組み合わせてもカスタムアイテムには使用できません。*Macromedia* 、*Flash Player* 、*Settings*。

対応バージョン： ActionScript 1.0、Flash Player 7

プロパティ一覧

オプション	プロパティ	説明
	caption:String	コンテキストメニューに表示するメニューアイテムのキャプション (テキスト) を示すストリングです。
	enabled:Boolean	指定されたメニューアイテムが有効か無効かを示すブール値です。
	separatorBefore: Boolean	指定されたメニューアイテムの上にセパレータを表示するかどうかを示すブール値です。
	visible:Boolean	Flash Player のコンテキストメニューを表示するときに、指定されたメニューアイテムを表示するかどうかを示すブール値です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベントの一覧

イベント	説明
onSelect = function(obj: Object, menuItem:Object) {}	指定したメニューアイテムが Flash Player のコンテキストメニューで選択されると、呼び出されます。

コンストラクター一覧

署名	説明
ContextMenuItem (caption:String, callbackFunction: Function, [separatorBefore: Boolean], [enabled:Boolean], [visible:Boolean])	ContextMenu.customItems 配列に追加できる新しい ContextMenuItem オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	copy() : ContextMenuItem	指定された ContextMenuItem オブジェクトのコピーを作成して返します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

caption (ContextMenuItem.caption プロパティ)

public caption : String

コンテキストメニューに表示するメニューアイテムのキャプション (テキスト) を示すストリングです。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例では、選択されたメニューアイテムのキャプション "Pause Game" を [出力] パネルに表示します。

```
var my_cm:ContextMenu = new ContextMenu();
var menuItem_cmi:ContextMenuItem = new ContextMenuItem("Pause Game", onPause);
my_cm.customItems.push(menuItem_cmi);
function onPause(obj, menuItem) {
  trace("You chose: " + menuItem.caption);
}
this.menu = my_cm;
```

ContextMenuItem コンストラクタ

```
public ContextMenuItem(caption:String, callbackFunction:Function,
  [separatorBefore:Boolean], [enabled:Boolean], [visible:Boolean])
```

ContextMenu.customItems 配列に追加できる新しい ContextMenuItem オブジェクトを作成します。

対応バージョン： ActionScript 1.0、Flash Player 7

パラメータ

caption:String - メニューアイテムに関連付けるテキストを指定するストリング。

callbackFunction:Function - メニューアイテムが選択されたときに呼び出す定義済みの関数。

separatorBefore:Boolean（オプション）- コンテキストメニュー内でメニューアイテムの上にセパレータを表示するかどうかを示すブール値。デフォルト値は false です。

enabled:Boolean（オプション）- コンテキストメニュー内でメニューアイテムを有効にするか無効にするかを示すブール値。デフォルト値は true です。

visible:Boolean（オプション）- メニューアイテムを表示するかどうかを示すブール値。デフォルト値は true です。

例

次の例では、セパレータで区切られた [Start] および [Stop] というメニューアイテムを ContextMenu オブジェクト my_cm に追加します。コンテキストメニューで [Start] が選択されると startHandler() 関数が呼び出され、[Stop] が選択されると stopHandler() が呼び出されます。ContextMenu オブジェクトは現在のタイムラインに適用されます。

```
var my_cm:ContextMenu = new ContextMenu();
my_cm.customItems.push(new ContextMenuItem("Start", startHandler));
my_cm.customItems.push(new ContextMenuItem("Stop", stopHandler, true));
function stopHandler(obj, item) {
  trace("Stopping...");
}
function startHandler(obj, item) {
  trace("Starting...");
}
this.menu = my_cm;
```

copy (ContextMenuItem.copy メソッド)

```
public copy() : ContextMenuItem
```

指定された ContextMenuItem オブジェクトのコピーを作成して返します。コピーには、元のオブジェクトのすべてのプロパティが含まれます。

対応バージョン : ActionScript 1.0、Flash Player 7

戻り値

`ContextMenuItem` - ContextMenuItem オブジェクト。

例

次の例では、新しい ContextMenuItem オブジェクト `original_cmi` を作成します。そのキャプションは "Pause"、コールバックハンドラは `onPause` 関数です。次に、ContextMenuItem オブジェクトのコピーを作成し、変数 `copy_cmi` に代入します。

```
var original_cmi:ContextMenuItem = new ContextMenuItem("Pause", onPause);
function onPause(obj:Object, menu:ContextMenu) {
  trace("pause me");
}

var copy_cmi:ContextMenuItem = original_cmi.copy();

var my_cm:ContextMenu = new ContextMenu();
my_cm.customItems.push(original_cmi);
my_cm.customItems.push(copy_cmi);

my_mc.menu = my_cm;
```

enabled (ContextMenuItem.enabled プロパティ)

```
public enabled : Boolean
```

指定されたメニューアイテムが有効か無効かを示すブール値です。このプロパティのデフォルト値は `true` です。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例では、2 つの新しいコンテキストメニューアイテム [Start] と [Stop] を作成します。ユーザーが [Start] を選択すると、SWF ファイルが起動されたときからの経過時間がミリ秒単位でトレースされます。次に、メニューの [Start] が無効になります。[Stop] が選択されると、SWF ファイルが起動されたときからの経過時間がミリ秒単位でトレースされます。[Start] メニューアイテムが再び有効になり、[Stop] メニューアイテムが無効になります。

```
var my_cm:ContextMenu = new ContextMenu();
var startMenuItem:ContextMenuItem = new ContextMenuItem("Start", startHandler);
startMenuItem.enabled = true;
my_cm.customItems.push(startMenuItem);
var stopMenuItem:ContextMenuItem = new ContextMenuItem("Stop", stopHandler,
  true);
stopMenuItem.enabled = false;
my_cm.customItems.push(stopMenuItem);
function stopHandler(obj, item) {
  trace("Stopping... "+getTimer()+"ms");
  startMenuItem.enabled = true;
  stopMenuItem.enabled = false;
}
function startHandler(obj, item) {
  trace("Starting... "+getTimer()+"ms");
  startMenuItem.enabled = false;
  stopMenuItem.enabled = true;
}
this.menu = my_cm;
```

onSelect (ContextMenuItem.onSelect ハンドラ)

```
onSelect = function(obj:Object, menuItem:Object) {}
```

指定したメニューアイテムが Flash Player のコンテキストメニューで選択されると、呼び出されます。指定したコールバックハンドラは、`obj` (ユーザーがコンテキストメニューを表示したときにマウスの下にあったオブジェクトの参照)、および `item` (選択されたメニューアイテムを表す ContextMenuItem オブジェクトの参照) の 2 つのパラメータを受け取ります。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

`obj:Object` - ユーザーが右クリック (Windows の場合) または Control キーを押したままクリック (Macintosh の場合) したオブジェクト (ムービークリップ、タイムライン、ボタン、または選択可能テキストフィールド) への参照。

`menuItem:Object` - 選択された ContextMenuItem オブジェクトへの参照。

例

次の例では、コンテキストメニューが呼び出されたオブジェクトの種類を判定します。

```
var my_cmi:ContextMenu = new ContextMenu();
var start_cmi:ContextMenuItem = new ContextMenuItem("Start");
start_cmi.onSelect = function(obj, item) {
  trace("You chose: "+item.caption);
};
my_cmi.customItems.push(start_cmi);
my_cmi.customItems.push(new ContextMenuItem("Stop", stopHandler, true));
function stopHandler(obj, item) {
  trace("Stopping...");
}
this.menu = my_cmi;
```

関連項目

onSelect (ContextMenu.onSelect ハンドラ)

separatorBefore (ContextMenuItem.separatorBefore プロパティ)

```
public separatorBefore : Boolean
```

指定されたメニューアイテムの上にセパレータを表示するかどうかを示すブール値です。このプロパティのデフォルト値は false です。

メモ：カスタムメニューアイテムとビルトインメニューアイテムの間には常にセパレータが表示されます。

対応バージョン： ActionScript 1.0、Flash Player 7

例

次の例では、[Open]、[Save]、[Print] という 3 つのメニューアイテムを作成します。[Save] と [Print] の間にはセパレータを表示します。次に、これらのメニューアイテムを ContextMenu オブジェクトの customItems 配列に追加します。最後に、メニューを SWF ファイルの現在のタイムラインに割り当てます。

```
var my_cm:ContextMenu = new ContextMenu();
var open_cmi:ContextMenuItem = new ContextMenuItem("Open", itemHandler);
var save_cmi:ContextMenuItem = new ContextMenuItem("Save", itemHandler);
var print_cmi:ContextMenuItem = new ContextMenuItem("Print", itemHandler);
print_cmi.separatorBefore = true;
my_cm.customItems.push(open_cmi, save_cmi, print_cmi);
function itemHandler(obj, menuItem) {
  trace("You chose: " + menuItem.caption);
};
this.menu = my_cm;
```

関連項目

onSelect (ContextMenu.onSelect ハンドラ)

visible (ContextMenuItem.visible プロパティ)

public visible : Boolean

Flash Player のコンテキストメニューを表示するときに、指定されたメニューアイテムを表示するかどうかを示すブール値です。このプロパティのデフォルト値は true です。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例では、2 つの新しいコンテキストメニューアイテム [Start] と [Stop] を作成します。ユーザーが [Start] を選択すると、SWF ファイルが起動されたときからの経過時間がミリ秒単位で表示されます。次に、メニューの [Start] が非表示になります。[Stop] が選択されると、SWF ファイルが起動されたときからの経過時間がミリ秒単位で表示されます。[Start] メニューアイテムが再び表示され、[Stop] メニューアイテムが非表示になります。

```
var my_cm:ContextMenu = new ContextMenu();
var startMenuItem:ContextMenuItem = new ContextMenuItem("Start", startHandler);
startMenuItem.visible = true;
my_cm.customItems.push(startMenuItem);
var stopMenuItem:ContextMenuItem = new ContextMenuItem("Stop", stopHandler,
  true);
stopMenuItem.visible = false;
my_cm.customItems.push(stopMenuItem);
function stopHandler(obj, item) {
  trace("Stopping... "+getTimer()+"ms");
  startMenuItem.visible = true;
  stopMenuItem.visible = false;
}
function startHandler(obj, item) {
  trace("Starting... "+getTimer()+"ms");
  startMenuItem.visible = false;
  stopMenuItem.visible = true;
}
this.menu = my_cm;
```

ConvolutionFilter (flash.filters.ConvolutionFilter)

```
Object
  |
  +-BitmapFilter
    |
    +-flash.filters.ConvolutionFilter
```

```
public class ConvolutionFilter
extends BitmapFilter
```

ConvolutionFilter クラスを使用すると、マトリックス畳み込みフィルタ効果を適用できます。畳み込みでは、入力イメージ内のピクセルを、隣接するピクセルと組み合わせて、イメージを作成します。畳み込みを使用すると、ぼかし、エッジ検出、シャープ、エンボス、ベベルなど、幅広いイメージ処理を実現できます。この効果は、ビットマップと MovieClip のインスタンスに適用できます。

フィルタの使い方は、フィルタの適用先オブジェクトによって異なります。

- 実行時にムービークリップにフィルタを適用する場合は、`filters` プロパティを使用します。オブジェクトの `filters` プロパティを設定してもオブジェクトは変更されません。また、`filters` プロパティをクリアすることで元に戻すことができます。
- BitmapData インスタンスにフィルタを適用するには、`BitmapData.applyFilter()` メソッドを使用します。BitmapData オブジェクトで `applyFilter()` を呼び出すことによって、ソース BitmapData オブジェクトとフィルタオブジェクトが取得され、フィルタを適用した結果として得られるイメージが生成されます。

イメージやビデオにもオーサリング時にフィルタ効果を適用できます。詳細については、オーサリングのマニュアルを参照してください。

ムービークリップやボタンにフィルタを適用する場合は、ムービークリップやボタンの `cacheAsBitmap` プロパティを `true` に設定します。すべてのフィルタをクリアすると、`cacheAsBitmap` は元の値に戻ります。

マトリックスの畳み込みは、*n* X *m* マトリックスに基づいています。入力イメージ内の特定のピクセル値を隣接するピクセル値と組み合わせることによって、新しいピクセル値を生成します。結果として得られるピクセルは、対応するソースピクセルおよび連接するピクセルにマトリックスを適用することで算出されます。

3 x 3 のマトリックス畳み込みの場合、独立するカラーチャンネルごとに次の式を使用します。

```
dst (x, y) = ((src (x-1, y-1) * a0 + src(x, y-1) * a1....
src(x, y+1) * a7 + src (x+1,y+1) * a8) / divisor) + bias
```

SSE (Streaming SIMD Extensions : ストリーミング SIMD 拡張) を提供するプロセッサで実行するときに処理が高速化されるフィルタ仕様もあります。

- フィルタは 3 x 3 フィルタである必要があります。
- フィルタ項はすべて -127 ～ +127 の整数である必要があります。
- すべてのフィルタ項の合計の絶対値は 127 を超えられません。
- いずれかのフィルタ項が負である場合、divisor は 2.00001 ～ 256 である必要があります。
- フィルタ項がすべて正である場合、divisor の許容範囲は 1.1 ～ 256 です。
- bias は整数である必要があります。

結果として得られるイメージの幅または高さが 2880 ピクセルを超える場合、フィルタは適用されません。たとえば、フィルタが適用されたサイズの大きいムービークリップをズームインするとき、結果として得られるイメージが 2880 ピクセルの制限に達する場合は、フィルタがオフになります。

対応バージョン : ActionScript 1.0、Flash Player 8

関連項目

applyFilter (BitmapData.applyFilter メソッド), filters (MovieClip.filters プロパティ), cacheAsBitmap (MovieClip.cacheAsBitmap プロパティ)

プロパティ一覧

オプション	プロパティ	説明
	alpha:Number	代替カラーのアルファ透明度の値です。
	bias:Number	マトリックス変換の結果に加算するバイアスです。
	clamp:Boolean	イメージをクランプする必要があるかどうかを示します。
	color:Number	ソースイメージの外にあるピクセルを置換する 16 進数のカラー値です。
	divisor:Number	マトリックス変換中に使用する除数です。
	matrix:Array	マトリックス変換に使用する値の配列です。コピーを返します。
	matrixX:Number	マトリックスの *x* 次元 (マトリックスの列数) です。
	matrixY:Number	マトリックスの *y* 次元 (マトリックスの行数) です。
	preserveAlpha: Boolean	畳み込みの適用先が何であるかを示します。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター覧

署名	説明
ConvolutionFilter (matrixX:Number, matrixY:Number, matrix:Array, [divisor:Number], [bias:Number], [preserveAlpha: Boolean], [clamp:Boolean], [color:Number], [alpha:Number])	指定されたパラメータで ConvolutionFilter インスタンスを初期化します。

メソッド一覧

オプション	署名	説明
	clone() : ConvolutionFilter	このフィルタオブジェクトのコピーを返します。

BitmapFilter クラスから継承されるメソッド

clone (BitmapFilter.clone メソッド)

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

alpha (ConvolutionFilter.alpha プロパティ)

```
public alpha : Number
```

代替カラーのアルファ透明度の値です。有効な値は 0 ～ 1.0 で、デフォルトは 0 です。たとえば値を 0.25 に設定すると、透明度は 25 パーセントになります。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、filter の alpha プロパティを、デフォルト値である 1 から 0.35 に変更します。

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var alpha:Number = 0.35;
var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1, 1, 1,
  1, 1], 9, 0, true, false, 0x0000FF, alpha);

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xCCFF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128, 0, 255, 1 | 2 | 4 | 8, false);

mc.onPress = function() {
  myBitmapData.applyFilter(myBitmapData, new Rectangle(0, 0, 98, 78), new
  Point(2, 2), filter);
}
```

bias (ConvolutionFilter.bias プロパティ)

public bias : Number

マトリックス変換の結果に加算するバイアスです。デフォルトは 0 です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、filter の bias プロパティを、デフォルト値である 0 から 50 に変更します。

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;

var bias:Number = 50;
var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1, 1, 1,
  1, 1], 9, bias);

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128);

mc.onPress = function() {
  myBitmapData.applyFilter(myBitmapData, myBitmapData.rectangle, new Point(0,
  0), filter);
}
```

clamp (ConvolutionFilter.clamp プロパティ)

public clamp : Boolean

イメージをクランプする必要があるかどうかを示します。true の場合、ソースイメージの外にあるピクセルに対して、入力イメージの所定のエッジのカラー値を複製するという方法で、必要に応じて境界に沿って入力イメージを拡張します。false の場合は、別の色を使用します。その色は color プロパティと alpha プロパティで指定します。デフォルト値は true です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、BitmapData クラスを使用して 2 つのボックスを作成します。そのうち 1 つはもう 1 つの半分のサイズです。最初の例がロードされるとき、attachBitmap() を使ってより大きなボックスが mc 内部に描画されます。mc がクリックされて applyFilter() メソッドが呼び出されると、smallBox をソースビットマップとして BitmapData の largeBox インスタンスが再描画されます。applyFilter() は、largeBox と同じ幅と高さに指定された Rectangle の上に smallBox を描画するため、ソースビットマップは描画領域より小さくなります。この場合、ConvolutionFilter の clamp プロパティは false に設定され、ソースビットマップ smallBox が重ならない領域は、変数 clampColor および clampAlpha によって指定されるとおり赤になります。

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

// Variables that affect clamping:
var clamp:Boolean = false;
var clampColor:Number = 0xFF0000;
var clampAlpha:Number = 1;

// For illustration, keep other ConvolutionFilter variables neutral:
var bias:Number = 0;
var preserveAlpha:Boolean = true;
// Construct a neutral matrix
var matrixCols:Number = 3;
var matrixRows:Number = 3;
var matrix:Array = [ 1,1,1,
                     1,1,1,
                     1,1,1 ];

var filter:ConvolutionFilter = new ConvolutionFilter(matrixCols, matrixRows,
  matrix, matrix.length, bias, preserveAlpha, clamp, clampColor, clampAlpha);

var largeBoxWidth:Number = 100;
var largeBoxHeight:Number = 100;
var largeBox:BitmapData = new BitmapData(largeBoxWidth, largeBoxWidth, true,
  0xCC00FF00);
var smallBoxWidth:Number = largeBoxWidth / 2;
var smallBoxHeight:Number = largeBoxHeight / 2;
var smallBox:BitmapData = new BitmapData(smallBoxWidth, smallBoxWidth, true,
  0xCC0000FF);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(largeBox, this.getNextHighestDepth());

mc.onPress = function() {
  largeBox.applyFilter(smallBox, new Rectangle(0,0, largeBoxWidth,
  largeBoxHeight), new Point(0,0), filter);
}
```

clone (ConvolutionFilter.clone メソッド)

```
public clone() : ConvolutionFilter
```

このフィルタオブジェクトのコピーを返します。

対応バージョン： ActionScript 1.0、Flash Player 8

戻り値

`ConvolutionFilter` - 元のインスタンスと同じプロパティをすべて備えた新しい ConvolutionFilter インスタンス。

例

次の例では、3 つの ConvolutionFilter オブジェクトを作成して比較します。`filter_1` は `ConvolutionFilter` コンストラクタを使って作成されます。`filter_2` は `filter_1` と等しい値に設定することにより作成されます。`clonedFilter` は `filter_1` のクローンを作成することにより作成されます。`filter_2` が `filter_1` に等しいと評価されても、`filter_1` と同じ値を含んでいる `clonedFilter` が等しいと評価されないことに注意してください。

```
import flash.filters.ConvolutionFilter;

var filter_1:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1, 1,
  1, 1, 1], 9);
var filter_2:ConvolutionFilter = filter_1;
var clonedFilter:ConvolutionFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false

for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> alpha: 0
  // >> color: 0
  // >> clamp: true
  // >> preserveAlpha: true
  // >> bias: 0
  // >> divisor: 9
  // >> matrix: 1,1,1,1,1,1,1,1,1
  // >> matrixY: 3
  // >> matrixX: 3
}
```

```
for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
  // >> alpha: 0
  // >> color: 0
  // >> clamp: true
  // >> preserveAlpha: true
  // >> bias: 0
  // >> divisor: 9
  // >> matrix: 1,1,1,1,1,1,1,1,1
  // >> matrixY: 3
  // >> matrixX: 3
}
```

filter_1、filter_2、および clonedFilter の関係をさらに詳しく示すために、次の例では、filter_1 の bias プロパティを変更します。bias を変更すると、clone() メソッドは filter_1 の値を参照する代わりに、それらの値に基づいて新しいインスタンスを作成します。

```
import flash.filters.ConvolutionFilter;

var filter_1:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1, 1,
  1, 1, 1], 9);
var filter_2:ConvolutionFilter = filter_1;
var clonedFilter:ConvolutionFilter = filter_1.clone();
trace(filter_1.bias); // 0
trace(filter_2.bias); // 0
trace(clonedFilter.bias); // 0

filter_1.bias = 20;

trace(filter_1.bias); // 20
trace(filter_2.bias); // 20
trace(clonedFilter.bias); // 0
```

color (ConvolutionFilter.color プロパティ)

```
public color : Number
```

ソースイメージの外にあるピクセルを置換する 16 進数のカラー値です。これはアルファ成分なしの RGB 値です。デフォルトは 0 です。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、filter の color プロパティを、デフォルト値である 0 から 0xFF0000. に変更します。

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;
```

```
var color:Number = 0x0000FF;
var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1, 1, 1,
  1, 1], 9, 0, true, false, color, 1);

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xCCFF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128, 0, 255, 1 | 2 | 4 | 8, false);

var height:Number = 100;
var width:Number = 80;
mc.onPress = function() {
  height -= 2;
  width -= 2;
  myBitmapData.applyFilter(myBitmapData, new Rectangle(0, 0, height, width), new
  Point(2, 2), filter);
}
```

ConvolutionFilter コンストラクタ

```
public ConvolutionFilter(matrixX:Number, matrixY:Number, matrix:Array,
  [divisor:Number], [bias:Number], [preserveAlpha:Boolean], [clamp:Boolean],
  [color:Number], [alpha:Number])
```

指定されたパラメータで ConvolutionFilter インスタンスを初期化します。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

matrixX:Number - マトリックスの *x* 次元 (マトリックスの列数)。デフォルト値は 0 です。

matrixY:Number - マトリックスの *y* 次元 (マトリックスの行数)。デフォルト値は 0 です。

matrix:Array - マトリックス変換に使用する値の配列です。コピーを返します。この配列に含まれる項目数は必ず matrixX*matrixY に等しくなります。

divisor:Number (オプション) - マトリックス変換中に使用される除数。デフォルト値は 1 です。除数がすべてのマトリックス値の合計と等しい場合は、結果全体のカラー強度が均等化されます。値 0 は無視し、代わりにデフォルト値を使用します。

bias:Number (オプション) - マトリックス変換の結果に加算するバイアスです。デフォルト値は 0 です。

preserveAlpha:Boolean (オプション) - false である場合は、アルファチャンネルを含め、すべてのチャンネルに畳み込みを適用します。true である場合は、畳み込みをカラーチャンネルだけに適用します。デフォルト値は true です。

clamp:Boolean (オプション) - true の場合、ソースイメージの外にあるピクセルに対して、入力イメージの所定のエッジのカラー値を複製するという方法で、必要に応じて境界に沿って入力イメージを拡張します。false の場合は、別の色を使用します。その色は color プロパティと alpha プロパティで指定します。デフォルト値は true です。

color:Number (オプション) - ソースイメージの外にあるピクセルを置換するための 16 進数のカラー値。

alpha:Number (オプション) - 代替カラーのアルファです。

例

次のコードは、除数が 9 である 3 x 3 畳み込みフィルタを作成します。このフィルタは、イメージをぼかします。

```
var myArray:Array = [1, 1, 1, 1, 1, 1, 1, 1, 1];
var myFilter:ConvolutionFilter = new flash.filters.ConvolutionFilter (3, 3,
myArray, 9);
```

次の例では、4 つの必須パラメータ matrixX、matrixY、matrix、divisor を使用して ConvolutionFilter オブジェクトを作成します。

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;

var matrixX:Number = 3;
var matrixY:Number = 3;
var matrix:Array = [1, 1, 1, 1, 1, 1, 1, 1, 1];
var divisor:Number = 9;

var filter:ConvolutionFilter = new ConvolutionFilter(matrixX, matrixY, matrix,
  divisor);

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128);

mc.onPress = function() {
  myBitmapData.applyFilter(myBitmapData, myBitmapData.rectangle, new Point(0,
  0), filter);
}
```

divisor (ConvolutionFilter.divisor プロパティ)

public divisor : Number

マトリックス変換中に使用する除数です。デフォルト値は 1 です。除数がすべてのマトリックス値の合計と等しい場合は、結果全体のカラー強度が均等化されます。値 0 は無視し、代わりにデフォルト値を使用します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、filter の divisor プロパティを 6 に変更します。

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;

var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1, 1, 1,
  1, 1], 9);

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128);

mc.onPress = function() {
  var newDivisor:Number = 6;
  filter.divisor = newDivisor;
    myBitmapData.applyFilter(myBitmapData, myBitmapData.rectangle, new Point(0,
  0), filter);
}
```

matrix (ConvolutionFilter.matrix プロパティ)

public matrix : Array

マトリックス変換に使用する値の配列です。コピーを返します。この配列に含まれる項目数は必ず matrixX*matrixY に等しくなります。

matrix プロパティは、たとえば myFilter.matrix[2] = 1; のようにして、値を直接変更することはできません。このプロパティを変更するには、次の例で示したように、配列への参照を取得し、その参照を変更し、filter.matrix = newMatrix; を使用して値をリセットする必要があります。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、filter の matrix プロパティを、ビットマップをぼかすものからシャープな効果を与えるものに変更します。

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;

var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1, 1, 1,
  1, 1], 9);

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128);

mc.onPress = function() {
  var newMatrix:Array = [0, -1, 0, -1, 8, -1, 0, -1, 0];
  filter.matrix = newMatrix;
  myBitmapData.applyFilter(myBitmapData, myBitmapData.rectangle, new Point(0,
  0), filter);
}
```

matrixX (ConvolutionFilter.matrixX プロパティ)

```
public matrixX : Number
```

マトリックスの x 次元 (マトリックスの列数) です。デフォルト値は 0 です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、filter の matrixX プロパティを表示します。

```
import flash.filters.ConvolutionFilter;

var filter:ConvolutionFilter = new ConvolutionFilter(2, 3, [1, 0, 0, 1, 0, 0],
  6);
trace(filter.matrixX); // 2
```

matrixY (ConvolutionFilter.matrixY プロパティ)

```
public matrixY : Number
```

マトリックスの y 次元 (マトリックスの行数) です。デフォルト値は 0 です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、filter の matrixY プロパティを表示します。

```
import flash.filters.ConvolutionFilter;

var filter:ConvolutionFilter = new ConvolutionFilter(2, 3, [1, 0, 0, 1, 0, 0],
  6);
trace(filter.matrixY); // 3
```

preserveAlpha (ConvolutionFilter.preserveAlpha プロパティ)

```
public preserveAlpha : Boolean
```

畳み込みの適用先が何であるかを示します。false である場合は、アルファチャンネルを含め、すべてのチャンネルに畳み込みを適用します。true である場合は、畳み込みをカラーチャンネルだけに適用します。デフォルト値は true です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、filter の preserveAlpha プロパティを、デフォルト値 true から false に変更します。

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;

var preserveAlpha:Boolean = false;
var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1, 1, 1,
  1, 1], 9, 0, preserveAlpha);

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xCCFF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128, 0, 255, 1 | 2 | 4 | 8, false);

mc.onPress = function() {
  myBitmapData.applyFilter(myBitmapData, myBitmapData.rectangle, new Point(0,
  0), filter);
}
```

CustomActions

```
Object
  |
  +-CustomActions
```

```
public class CustomActions
extends Object
```

CustomActions クラスのメソッドを使用すると、Flash オーサリングツールで再生している SWF ファイルで、オーサリングツールに登録されている任意のカスタムアクションを管理できます。SWF ファイルにより、カスタムアクションのインストールとアンインストール、カスタムアクションの XML 定義の取得、および登録されているカスタムアクションのリストの取得を行うことができます。

これらのメソッドを使用して、Flash オーサリングツールの拡張として SWF ファイルをビルドできます。このような拡張では、Flash Application Protocol を使用して UDDI リポジトリをナビゲートしたり、Web サービスを [アクション] ツールボックスにダウンロードしたりできます。

対応バージョン： ActionScript 1.0、Flash Player 6

プロパティ一覧

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

メソッド一覧

オプション	署名	説明
static	get(name:String) : String	カスタムアクション XML 定義ファイル name の内容を読み込みます。
static	install(name:String, data:String) : Boolean	name パラメータで指定された新しいカスタムアクション XML 定義ファイルをインストールします。
static	list() : Array	Flash オーサリングツールに登録されているすべてのカスタムアクションの名前で構成される Array オブジェクトを返します。
static	uninstall(name: String) : Boolean	カスタムアクション XML 定義ファイル name を削除します。

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

get (CustomActions.get メソッド)

```
public static get(name:String) : String
```

カスタムアクション XML 定義ファイル name の内容を読み込みます。

定義ファイルの名前は単純なファイル名である必要があります。.xml ファイル拡張子を付けず、またディレクトリ区切り記号 (':'、'/'、または '\') も使用しません。

name で指定された定義ファイルが見つからない場合は、undefined が返されます。name パラメータで指定されたカスタムアクション XML 定義が見つかると、その全体が読み取られて、文字列として返されます。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

name:String - 取得するカスタムアクション定義の名前。

戻り値

String - カスタムアクションの XML 定義が見つかった場合はストリング、それ以外の場合は undefined を返します。

例

次の例では、ComboBox インスタンス内のカスタムアクションを一覧表示し、ボタンインスタンスがクリックされたときにカスタムアクションを取得します。ComboBox、Button、および TextArea のインスタンスをステージにドラッグします。ComboBox のインスタンス名は customActionName_cb、TextArea のインスタンス名は customActionXml_ta、Button のインスタンス名は view_button とします。タイムラインのフレーム 1 に次の ActionScript を入力します。

```
import mx.controls.*;

var customActionName_cb:ComboBox;
var customActionXml_ta:TextArea;
var view_button:Button;
```

```
customActionName_cb.dataProvider = CustomActions.list();

customActionXml_ta.editable = false;

var viewListener:Object = new Object();
viewListener.click = function(evt:Object) {
  var caName:String = String(customActionName_cb.selectedItem);
  customActionXml_ta.text = CustomActions.get(caName);
};
view_button.addEventListener("click", viewListener);
```

install (CustomActions.install メソッド)

```
public static install(name:String, data:String) : Boolean
```

name パラメータで指定された新しいカスタムアクション XML 定義ファイルをインストールします。このファイルの内容は、ストリング *customXML* で指定されます。

定義ファイルの名前は単純なファイル名である必要があります。.xml ファイル拡張子を付けず、またディレクトリ区切り記号 (':'、'/'、または '\') も使用しません。

名前 name で既存のカスタムアクションファイルがある場合は、上書きされます。

このメソッドが呼び出されたときに Configuration/ActionsPanel/CustomActions ディレクトリが存在しない場合は、そのディレクトリが作成されます。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

name:String - インストールするカスタムアクション定義の名前。

data:String - インストールする XML 定義のテキスト。

戻り値

Boolean - インストール中にエラーが発生すると、ブール値 false が返されます。それ以外は、カスタムアクションが正常にインストールされたことを示す true が返されます。

例

次の例では、XML ファイルの情報を [アクション] パネルにインストールします。テキストエディタを開き、新しいドキュメントを dogclass.xml という名前で保存します。次のコードを入力します。

```
<?xml version="1.0"?>
<customactions>
  <actionspanel>
  <folder version="7" id="DogClass" index="true" name="Dog" tiptext="Dog Class">
    <string version="7" id="getFleas" name="getFleas" tiptext="gets number of
  fleas" text=".getFleas(% fleas %)" />
  </folder>
  </actionspanel>
  <colorsyntax>
    <identifier text=".getFleas" />
  </colorsyntax>
  <codehints>
    <typeinfo pattern=" _dog" object="Dog"/>
  </codehints>
</customactions>
```

次に、同じディレクトリ内で新しい FLA ファイルを開き、タイムラインのフレーム 1 を選択します。[アクション] パネルで、次のコードを入力します。

```
var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean) {
  trace(success);
  CustomActions.install("dogclass", this.firstChild);
  trace(CustomActions.list());
};
my_xml.load("dogclass.xml");
```

[制御]-[ムービープレビュー] を選択します。XML が正常にロードされると `true` となり、Flash オーサリングツールに登録されているすべてのカスタムアクションの名前を含む配列が [出力] パネルに表示されます。SWF ファイルを閉じ、[アクション] パネルを開きます。[アクション] ツールボックスに "Dog" という新しいフォルダが増え、その中に "getFleas" という項目が表示されます。

list (CustomActions.list メソッド)

```
public static list() : Array
```

Flash オーサリングツールに登録されているすべてのカスタムアクションの名前で構成される Array オブジェクトを返します。配列のエレメントは単純な名前です。.xml ファイル拡張子もディレクトリ区切り記号 (":"、"/"、"\" など) も使用しません。登録済みのカスタムアクションがない場合は、`list()` により長さゼロの配列が返されます。エラーが発生した場合、`list()` は値 `undefined` を返します。

対応バージョン : ActionScript 1.0、Flash Player 6

戻り値

`Array` - 配列。

例

次の例では、ComboBox インスタンス内のカスタムアクションを一覧表示し、ボタンインスタンスがクリックされたときにカスタムアクションを取得します。ComboBox、Button、および TextArea のインスタンスをステージにドラッグします。ComboBox のインスタンス名は `customActionName_cb`、TextArea のインスタンス名は `customActionXml_ta`、Button のインスタンス名は `view_button` とします。タイムラインのフレーム 1 に次の ActionScript を入力します。

```
import mx.controls.*;

var customActionName_cb:ComboBox;
var customActionXml_ta:TextArea;
var view_button:Button;

customActionName_cb.dataProvider = CustomActions.list();

customActionXml_ta.editable = false;

var viewListener:Object = new Object();
viewListener.click = function(evt:Object) {
   var caName:String = String(customActionName_cb.selectedItem);
   customActionXml_ta.text = CustomActions.get(caName);
};
view_button.addEventListener("click", viewListener);
```

uninstall (CustomActions.uninstall メソッド)

```
public static uninstall(name:String) : Boolean
```

カスタムアクション XML 定義ファイル name を削除します。

定義ファイルの名前は単純なファイル名である必要があります。.xml ファイル拡張子を付けず、またディレクトリ区切り記号 (':'、'/'、または '\') も使用しません。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

`name:String` - アンインストールするカスタムアクション定義の名前。

戻り値

`Boolean` - name に対応するカスタムアクションが見つからない場合は、ブール値 false を返します。カスタムアクションを正常に削除した場合は、true を返します。

例

次の例では、新しいカスタムアクションをインストールし、Flash オーサリングツールに登録されているすべてのカスタムアクションの名前を含む配列を [出力] パネルに表示します。uninstall_btn をクリックすると、カスタムアクションはアンインストールされます。インストールされているカスタムアクションの名前を含む配列が表示され、dogclass が配列から削除されます。ボタン uninstall_btn を作成し、タイムラインのフレーム 1 に次の ActionScript を入力します。

```
var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean) {
  trace(success);
  CustomActions.install("dogclass", this.firstChild);
  trace(CustomActions.list());
};
my_xml.load("dogclass.xml");

uninstall_btn.onRelease = function() {
  CustomActions.uninstall("dogclass");
  trace(CustomActions.list());
};
```

dogclass.xml の作成方法の詳細については、CustomActions.install() を参照してください。

関連項目

install (CustomActions.install メソッド)

Date

```
Object
  |
  +-Date
```

```
public class Date
extends Object
```

Date クラスを使用すると、世界時 (グリニッジ標準時、現在の呼称は世界標準時または UTC) を基準にするか、Flash Player を実行しているオペレーティングシステムを基準にして、日時の値を調べることができます。Date クラスのメソッドは静的ではありません。メソッドを呼び出すときに指定した個々の Date オブジェクトにのみ適用されます。`Date.UTC()` メソッドは例外で、これは静的なメソッドです。

Date クラスによる夏時間の処理方法は、オペレーティングシステムおよび Flash Player のバージョンに応じて異なります。Flash Player 6 以降では、夏時間は各オペレーティングシステムで次のように処理されます。

- Windows - Date オブジェクトの出力は夏時間に合わせて自動的に調整されます。Date オブジェクトは、夏時間が現在の地域で採用されているかどうかを確認します。夏時間が採用されている場合は、夏時間に移行する日付と時刻の基準を確認します。ただし、現在有効な夏時間の日付が過去と未来に適用されるために、夏時間の日付が異なる地域では、計算される夏時間の日付が過去の日付と食い違う場合があります。
- Mac OS X - Date オブジェクトの出力は夏時間に合わせて自動的に調整されます。Mac OS X では、タイムゾーン情報データベースを使用して、現在または過去の日付または時刻に夏時間の時差を適用する必要があるかどうかを決定します。
- Mac OS 9 - 現在の日付および時刻に夏時間の時差を適用する必要があるかどうかを判定できるだけの情報しか提供されません。したがって、Date オブジェクトは、現在の夏時間の時差が過去および将来のすべての日時に適用されると想定します。

Flash Player 5 によるオペレーティングシステム別の夏時間の対処方法は次のとおりです。

- Windows - 夏時間に関する米国の規則が常に適用されるために、米国とは夏時間に移行する時期が異なるヨーロッパおよび他の地域では移行期間が食い違います。Flash は、現在の地域で採用されている夏時間を正しく検出します。

Date クラスのメソッドを呼び出すには、まず後述の Date クラスのコンストラクタを使用して Date オブジェクトを作成する必要があります。

対応バージョン : ActionScript 1.0、Flash Player 5

プロパティー覧

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター覧

署名	説明
Date([yearOrTimevalue:Number], [month:Number], [date:Number], [hour:Number], [minute:Number], [second:Number], [millisecond:Number])	指定された日時を保持する新しい Date オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	getDate() : Number	指定された Date オブジェクトの日付 (1 ～ 31 の整数) をローカル時間で返します。
	getDay() : Number	指定された Date オブジェクトの曜日 (日曜日は 0、月曜日は 1 など) をローカル時間で返します。
	getFullYear() : Number	指定された Date オブジェクトの年 (2000 など、4 桁の数字) をローカル時間で返します。
	getHours() : Number	指定された Date オブジェクトの時 (0 ～ 23 の整数) をローカル時間で返します。
	getMilliseconds() : Number	指定された Date オブジェクトのミリ秒 (0 ～ 999 の整数) をローカル時間で返します。
	getMinutes() : Number	指定された Date オブジェクトの分 (0 ～ 59 の整数) をローカル時間で返します。
	getMonth() : Number	指定された Date オブジェクトの月 (1 月は 0、2 月は 1 など) をローカル時間で返します。
	getSeconds() : Number	指定された Date オブジェクトの秒 (0 ～ 59 の整数) をローカル時間で返します。
	getTime() : Number	指定された Date オブジェクトに関して、1970 年 1 月 1 日 0 時 (世界時) からのミリ秒数を返します。

オプション	署名	説明
	getTimezoneOffset() : Number	コンピュータのローカル時間と世界時の差 (分単位) を返します。
	getUTCDate() : Number	指定された Date オブジェクトの日付 (1 ～ 31 の整数) を世界時で返します。
	getUTCDay() : Number	指定された Date オブジェクトの曜日 (日曜日は 0、月曜日は 1 など) を世界時で返します。
	getUTCFullYear() : Number	指定された Date オブジェクトの年 (4 桁表記) を世界時で返します。
	getUTCHours() : Number	指定された Date オブジェクトの時 (0 ～ 23 の整数) を世界時で返します。
	getUTCMilliseconds() : Number	指定された Date オブジェクトのミリ秒 (0 ～ 999 の整数) を世界時で返します。
	getUTCMinutes() : Number	指定された Date オブジェクトの分 (0 ～ 59 の整数) を世界時で返します。
	getUTCMonth() : Number	指定された Date オブジェクトの月 (0 [1 月] ～ 11 [12 月]) を世界時で返します。
	getUTCSeconds() : Number	指定された Date オブジェクトの秒 (0 ～ 59 の整数) を世界時で返します。
	getUTCYear() : Number	この Date の年を世界時 (UTC) で返します。
	getYear() : Number	指定された Date オブジェクトの年をローカル時間で返します。
	setDate(date:Number) : Number	指定された Date オブジェクトの日付をローカル時間で設定し、新しい時間をミリ秒で返します。
	setFullYear(year:Number, [month:Number], [date:Number]) : Number	指定された Date オブジェクトの年をローカル時間で設定し、新しい時間をミリ秒で返します。
	setHours(hour:Number) : Number	指定された Date オブジェクトの時をローカル時間で設定し、新しい時間をミリ秒で返します。
	setMilliseconds (millisecond:Number) : Number	指定された Date オブジェクトのミリ秒をローカル時間で設定し、新しい時間をミリ秒で返します。
	setMinutes(minute:Number) : Number	指定された Date オブジェクトの分をローカル時間で設定し、新しい時間をミリ秒で返します。

オプション	署名	説明
	setMonth(month:Number, [date:Number]) : Number	指定された Date オブジェクトの月をローカル時間で設定し、新しい時間をミリ秒で返します。
	setSeconds(second: Number) : Number	指定された Date オブジェクトの秒をローカル時間で設定し、新しい時間をミリ秒で返します。
	setTime(millisecond: Number) : Number	指定された Date オブジェクトの日付を 1970 年 1 月 1 日 0 時からのミリ秒数で設定し、新しい時間をミリ秒で返します。
	setUTCDate(date: Number) : Number	指定された Date オブジェクトの日付を世界時で設定し、新しい時間をミリ秒で返します。
	setUTCFullYear(year: Number, [month:Number], [date:Number]) : Number	指定された Date オブジェクト (*my_date*) の年を世界時で設定し、新しい時間をミリ秒で返します。
	setUTCHours(hour: Number, [minute:Number], [second:Number], [millisecond: Number]) : Number	指定された Date オブジェクトの時を世界時で設定し、新しい時間をミリ秒で返します。
	setUTCMilliseconds (millisecond:Number) : Number	指定された Date オブジェクトのミリ秒を世界時で設定し、新しい時間をミリ秒で返します。
	setUTCMinutes(minute: Number, [second:Number], [millisecond: Number]) : Number	指定された Date オブジェクトの分を世界時で設定し、新しい時間をミリ秒で返します。
	setUTCMonth(month: Number, [date:Number]) : Number	指定された Date オブジェクトの月を世界時で設定し (必要であれば日付も設定可能)、新しい時間をミリ秒で返します。
	setUTCSeconds(second: Number, [millisecond: Number]) : Number	指定された Date オブジェクトの秒を世界時で設定し、新しい時間をミリ秒で返します。
	setYear(year:Number) : Number	指定された Date オブジェクトの年をローカル時間で設定し、新しい時間をミリ秒で返します。

オプション	署名	説明
	toString() : String	指定された Date オブジェクトのストリング値を読み取り可能な形式で返します。
static	UTC(year:Number, month:Number, [date:Number], [hour:Number], [minute:Number], [second:Number], [millisecond: Number]) : Number	1970 年 1 月 1 日 0 時 (世界時) からパラメータで指定された時刻までのミリ秒数を返します。
	valueOf() : Number	この Date に関して、1970 年 1 月 1 日 0 時 (世界時) からのミリ秒数を返します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

Date コンストラクタ

```
public Date([yearOrTimevalue:Number], [month:Number], [date:Number],
  [hour:Number], [minute:Number], [second:Number], [millisecond:Number])
```

指定された日時を保持する新しい Date オブジェクトを作成します。

Date() コンストラクタは、日付と、ミリ秒までの時刻を指定するために、最大 7 つまでパラメータ (year、month、...、millisecond) を取ります。別の方法として、1970 年 1 月 1 日 0:00:00 (世界時) を基準とする経過時間をミリ秒単位で表す 1 つの値を Date() コンストラクタに渡すこともできます。また、パラメータをまったく指定しないこともできます。その場合は、現在の日時が代入された Date() オブジェクトが作成されます。

Date オブジェクトの作成方法をいくつか次に示します。

```
var d1:Date = new Date();
var d3:Date = new Date(2000, 0, 1);
var d4:Date = new Date(65, 2, 6, 9, 30, 15, 0);
var d5:Date = new Date(-14159025000);
```

先頭行のコードでは、代入ステートメントが実行されたときの時刻が Date オブジェクトに設定されます。

2 行目では、year、month、date パラメータを使って Date オブジェクトを作成しています。その結果、2000 年 1 月 1 日 0:00:00 (世界時) になります。

3 行目では、year、month、date パラメータを使って Date オブジェクトを作成しています。その結果、1965 年 3 月 6 日 09:30:15 (世界時) (+0 ミリ秒) になります。ここでは、年が 2 桁の整数で指定されているので 1965 年として解釈されます。

4 行目では、パラメータを 1 つだけ使用しています。このパラメータは 1970 年 1 月 1 日 0:00:00 (世界時) を基準とする経過時間をミリ秒単位で表す時間値です。この値が負の場合は 1970 年 1 月 1 日 0:00:00 (世界時) より前の時刻を表すので、この例では、1969 年 7 月 21 日 02:56:15 (世界時) になります。

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

`yearOrTimevalue:Number` (オプション) - 他にもパラメータが指定されている場合、この数値は年 (1965 など) を表します。他にパラメータが設定されていない場合は、時間値を表します。数値が年を表す場合、0 ～ 99 の値は 1900 ～ 1999 を表します。それ以外の場合は年を 4 桁で指定する必要があります。数値が時間値を表す (他にパラメータが指定されない) 場合は、1970 年 1 月 1 日 0:00:00 を基準とする経過時間をミリ秒単位で表す値になります。負の値は 1970 年 1 月 1 日 0:00:00 (世界時) よりも前の時刻を表し、正の値はそれより後の時刻を表します。

`month:Number` (オプション) - 0 (1 月) ～ 11 (12 月) の整数です。

`date:Number` (オプション) - 1 ～ 31 の整数です。

`hour:Number` (オプション) - 0 (0 時) ～ 23 (午後 11 時) の整数です。

`minute:Number` (オプション) - 0 ～ 59 の整数です。

`second:Number` (オプション) - 0 ～ 59 の整数です。

`millisecond:Number` (オプション) - ミリ秒を表す 0 ～ 999 の整数です。

例

次の例では、現在の日時を調べます。

```
var now_date:Date = new Date();
```

次の例では、Mary の誕生日である 1974 年 8 月 12 日を表す新しい Date オブジェクトを作成します。月のパラメータはゼロから始まるので、この例では 8 ではなく 7 を使用します。

```
var maryBirthday:Date = new Date (74, 7, 12);
```

次の例では、新しい Date オブジェクトを作成し、Date.getMonth()、Date.getDate()、および Date.getFullYear() の戻り値を連結します。

```
var today_date:Date = new Date();
var date_str:String = ((today_date.getMonth()+1)+"/"+today_date.getDate()+"/
  "+today_date.getFullYear());
trace(date_str); // displays current date in United States date format
```

関連項目

getMonth (Date.getMonth メソッド), getDate (Date.getDate メソッド), getFullYear (Date.getFullYear メソッド)

getDate (Date.getDate メソッド)

```
public getDate() : Number
```

指定された Date オブジェクトの日付 (1 ～ 31 の整数) をローカル時間で返します。ローカル時間は、Flash Player を実行しているオペレーティングシステムによって決まります。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Number - 整数。

例

次の例では、新しい Date オブジェクトを作成し、Date.getMonth()、Date.getDate()、および Date.getFullYear() の戻り値を連結します。:

```
var today_date:Date = new Date();
var date_str:String = (today_date.getDate()+"/"+(today_date.getMonth()+1)+"/
  "+today_date.getFullYear());
trace(date_str); // displays current date in United States date format
```

関連項目

getMonth (Date.getMonth メソッド), getFullYear (Date.getFullYear メソッド)

getDay (Date.getDay メソッド)

```
public getDay() : Number
```

指定された Date オブジェクトの曜日 (日曜日は 0、月曜日は 1 など) をローカル時間で返します。ローカル時間は、Flash Player を実行しているオペレーティングシステムによって決まります。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Number - 曜日を表す整数。

例

次の例では、新しい Date オブジェクトを作成し、getDay() を使用して現在の曜日を取得します。:

```
var dayOfWeek_array:Array = new Array("Sunday", "Monday", "Tuesday",
  "Wednesday", "Thursday", "Friday", "Saturday");
var today_date:Date = new Date();
var day_str:String = dayOfWeek_array[today_date.getDay()];
trace("Today is "+day_str);
```

getFullYear (Date.getFullYear メソッド)

```
public getFullYear() : Number
```

指定された Date オブジェクトの年 (2000 など、4 桁の数字) をローカル時間で返します。ローカル時間は、Flash Player を実行しているオペレーティングシステムによって決まります。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Number - 年を表す整数。

例

次の例では、コンストラクタを使って Date オブジェクトを作成します。trace ステートメントは、getFullYear() メソッドの戻り値を表示します。

```
var my_date:Date = new Date();
trace(my_date.getYear()); // displays 104
trace(my_date.getFullYear()); // displays current year
```

getHours (Date.getHours メソッド)

```
public getHours() : Number
```

指定された Date オブジェクトの時 (0 ～ 23 の整数) をローカル時間で返します。ローカル時間は、Flash Player を実行しているオペレーティングシステムによって決まります。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

`Number` - 整数。

例

次の例では、コンストラクタを使って現在の時刻に基づいて Date オブジェクトを作成し、`getHours()` メソッドを使ってオブジェクトの時の値を表示します。

```
var my_date:Date = new Date();
trace(my_date.getHours());

var my_date:Date = new Date();
var hourObj:Object = getHoursAmPm(my_date.getHours());
trace(hourObj.hours);
trace(hourObj.ampm);

function getHoursAmPm(hour24:Number):Object {
  var returnObj:Object = new Object();
  returnObj.ampm = (hour24<12) ? "AM" : "PM";
  var hour12:Number = hour24%12;
  if (hour12 == 0) {
  hour12 = 12;
  }
  returnObj.hours = hour12;
  return returnObj;
}
```

getMilliseconds (Date.getMilliseconds メソッド)

```
public getMilliseconds() : Number
```

指定された Date オブジェクトのミリ秒 (0 ～ 999 の整数) をローカル時間で返します。ローカル時間は、Flash Player を実行しているオペレーティングシステムによって決まります。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

`Number` - 整数。

例

次の例では、コンストラクタを使って現在の時刻に基づいて Date オブジェクトを作成し、getMilliseconds() メソッドを使ってオブジェクトのミリ秒の値を返します。

```
var my_date:Date = new Date();
trace(my_date.getMilliseconds());
```

getMinutes (Date.getMinutes メソッド)

```
public getMinutes() : Number
```

指定された Date オブジェクトの分 (0 ～ 59 の整数) をローカル時間で返します。ローカル時間は、Flash Player を実行しているオペレーティングシステムによって決まります。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Number - 整数。

例

次の例では、コンストラクタを使って現在の時刻に基づいて Date オブジェクトを作成し、getMinutes() メソッドを使ってオブジェクトの分の値を返します。

```
var my_date:Date = new Date();
trace(my_date.getMinutes());
```

getMonth (Date.getMonth メソッド)

```
public getMonth() : Number
```

指定された Date オブジェクトの月 (1 月は 0、2 月は 1 など) をローカル時間で返します。ローカル時間は、Flash Player を実行しているオペレーティングシステムによって決まります。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Number - 整数。

例

次の例では、コンストラクタを使って現在の時刻に基づいて Date オブジェクトを作成し、getMonth() メソッドを使ってオブジェクトの月の値を返します。

```
var my_date:Date = new Date();
trace(my_date.getMonth());
```

次の例では、コンストラクタを使って現在の時刻に基づいて Date オブジェクトを作成し、getMonth() メソッドを使って現在の月を数値として表示し、さらに月の名前を表示します。

```
var my_date:Date = new Date();
trace(my_date.getMonth());
trace(getMonthAsString(my_date.getMonth()));
function getMonthAsString(month:Number):String {
   var monthNames_array:Array = new Array("January", "February", "March",
   "April", "May", "June", "July", "August", "September", "October", "November",
   "December");
   return monthNames_array[month];
}
```

getSeconds (Date.getSeconds メソッド)

```
public getSeconds() : Number
```

指定された Date オブジェクトの秒 (0 ～ 59 の整数) をローカル時間で返します。ローカル時間は、Flash Player を実行しているオペレーティングシステムによって決まります。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Number - 整数。

例

次の例では、コンストラクタを使って現在の時刻に基づいて Date オブジェクトを作成し、getSeconds() メソッドを使ってオブジェクトの秒の値を返します。

```
var my_date:Date = new Date();
trace(my_date.getSeconds());
```

getTime (Date.getTime メソッド)

```
public getTime() : Number
```

指定された Date オブジェクトに関して、1970 年 1 月 1 日 0 時 (世界時) からのミリ秒数を返します。このメソッドは、複数の Date オブジェクトを比較する際に特定の時間を表すのに使用します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

`Number` - 整数。

例

次の例では、コンストラクタを使って現在の時刻に基づく Date オブジェクトを作成し、`getTime()` メソッドを使って 1970 年 1 月 1 日 0 時からのミリ秒数を返します。

```
var my_date:Date = new Date();
trace(my_date.getTime());
```

getTimezoneOffset (Date.getTimezoneOffset メソッド)

```
public getTimezoneOffset() : Number
```

コンピュータのローカル時間と世界時の差 (分単位) を返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

`Number` - 整数。

例

次の例では、サンフランシスコの現地夏時間と世界時の差を返します。夏時間は、Date オブジェクトで定義された日付が夏時間の期間内である場合だけ、返される結果に反映されます。この例の出力は 420 分であり、[出力] パネルに表示されます (7 時間 * 60 分 / 時 = 420 分)。この例は太平洋標準時の夏時間 (PDT, GMT-0700) です。結果は地域と時期に応じて異なります。

```
var my_date:Date = new Date();
trace(my_date.getTimezoneOffset());
```

getUTCDate (Date.getUTCDate メソッド)

```
public getUTCDate() : Number
```

指定された Date オブジェクトの日付 (1 ～ 31 の整数) を世界時で返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

`Number` - 整数。

例

次の例では、新しい Date オブジェクトを作成し、`Date.getUTCDate()` および `Date.getDate()` を使用します。`Date.getUTCDate()` の戻り値は、現地のタイムゾーンと世界時の関係によっては、`Date.getDate()`, の戻り値と異なる場合があります。

```
var my_date:Date = new Date(2004,8,25);
trace(my_date.getUTCDate()); // output: 25
```

関連項目

`getDate (Date.getDate メソッド )`

getUTCDay (Date.getUTCDay メソッド)

```
public getUTCDay() : Number
```

指定された Date オブジェクトの曜日 (日曜日は 0、月曜日は 1 など) を世界時で返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

`Number` - 整数。

例

次の例では、新しい Date オブジェクトを作成し、`Date.getUTCDay()` および `Date.getDay()` を使用します。`Date.getUTCDay()` の戻り値は、現地のタイムゾーンと世界時の関係によっては、`Date.getDay()`, の戻り値と異なる場合があります。

```
var today_date:Date = new Date();
trace(today_date.getDay()); // output will be based on local timezone
trace(today_date.getUTCDay()); // output will equal getDay() plus or minus one
```

関連項目

`getDay (Date.getDay メソッド )`

getUTCFullYear (Date.getUTCFullYear メソッド)

```
public getUTCFullYear() : Number
```

指定された Date オブジェクトの年 (4 桁表記) を世界時で返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

`Number` - 整数。

例

次の例では、新しい Date オブジェクトを作成し、`Date.getUTCFullYear()` および `Date.getFullYear()` を使用します。今日の日付が 12 月 31 日または 1 月 1 日の場合、現地のタイムゾーンと世界時の関係によっては、`Date.getUTCFullYear()` の戻り値が `Date.getFullYear()` の戻り値と異なる場合があります。

```
var today_date:Date = new Date();
trace(today_date.getFullYear()); // display based on local timezone
trace(today_date.getUTCFullYear()); // displays getYear() plus or minus 1
```

関連項目

`getFullYear (Date.getFullYear メソッド )`

getUTCHours (Date.getUTCHours メソッド)

```
public getUTCHours() : Number
```

指定された Date オブジェクトの時 (0 ～ 23 の整数) を世界時で返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

`Number` - 整数。

例

次の例では、新しい Date オブジェクトを作成し、`Date.getUTCHours()` および `Date.getHours()` を使用します。`Date.getUTCHours()` の戻り値は、現地のタイムゾーンと世界時の関係によっては、`Date.getHours()`, の戻り値と異なる場合があります。

```
var today_date:Date = new Date();
trace(today_date.getHours()); // display based on local timezone
trace(today_date.getUTCHours()); // display equals getHours() plus or minus 12
```

関連項目

getHours (Date.getHours メソッド)

getUTCMilliseconds (Date.getUTCMilliseconds メソッド)

```
public getUTCMilliseconds() : Number
```

指定された Date オブジェクトのミリ秒 (0 ～ 999 の整数) を世界時で返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Number - 整数。

例

次の例では、新しい Date オブジェクトを作成し、getUTCMilliseconds() を使って Date オブジェクトのミリ秒の値を返します。

```
var today_date:Date = new Date();
trace(today_date.getUTCMilliseconds());
```

getUTCMinutes (Date.getUTCMinutes メソッド)

```
public getUTCMinutes() : Number
```

指定された Date オブジェクトの分 (0 ～ 59 の整数) を世界時で返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Number - 整数。

例

次の例では、新しい Date オブジェクトを作成し、getUTCMinutes() を使って Date オブジェクトの分の値を返します。

```
var today_date:Date = new Date();
trace(today_date.getUTCMinutes());
```

getUTCMonth (Date.getUTCMonth メソッド)

```
public getUTCMonth() : Number
```

指定された Date オブジェクトの月 (0 [1 月] ～ 11 [12 月]) を世界時で返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

`Number` - 整数。

例

次の例では、新しい Date オブジェクトを作成し、`Date.getUTCMonth()` および `Date.getMonth()` を使用します。今日の日付が月の最初または最後の日の場合、`Date.getUTCMonth()` の戻り値は、現地のタイムゾーンと世界時の関係によっては、`Date.getMonth()` の戻り値と異なる場合があります。

```
var today_date:Date = new Date();
trace(today_date.getMonth()); // output based on local timezone
trace(today_date.getUTCMonth()); // output equals getMonth() plus or minus 1
```

関連項目

`getMonth (Date.getMonth メソッド )`

getUTCSeconds (Date.getUTCSeconds メソッド)

```
public getUTCSeconds() : Number
```

指定された Date オブジェクトの秒 (0 ～ 59 の整数) を世界時で返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

`Number` - 整数。

例

次の例では、新しい Date オブジェクトを作成し、`getUTCSeconds()` を使って Date オブジェクトの秒の値を返します。

```
var today_date:Date = new Date();
trace(today_date.getUTCSeconds());
```

getUTCYear (Date.getUTCYear メソッド)

```
public getUTCYear() : Number
```

この Date の年を世界時 (UTC) で返します。年は、4 桁の年から 1900 を引いて示されます。たとえば、2000 年は 100 と表されます。

対応バージョン : ActionScript 1.0、Flash Player 8

戻り値

Number - 整数。

例

次の例では、新しい Date オブジェクトを作成し、Date.getUTCFullYear() および Date.getFullYear() を使用します。今日の日付が 12 月 31 日または 1 月 1 日の場合、現地のタイムゾーンと世界時の関係によっては、Date.getUTCFullYear() の戻り値が Date.getFullYear() の戻り値と異なる場合があります。

```
var today_date:Date = new Date();

trace(today_date.getFullYear()); // display based on local timezone

trace(today_date.getUTCFullYear()); // displays getYear() plus or minus 1
```

getYear (Date.getYear メソッド)

```
public getYear() : Number
```

指定された Date オブジェクトの年をローカル時間で返します。ローカル時間は、Flash Player を実行しているオペレーティングシステムによって決まります。年は、4 桁の年から 1900 を引いて示されます。たとえば、2000 年は 100 と表されます。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Number - 整数。

例

次の例では、年月が 2004 年 5 月に設定された Date オブジェクトを作成します。この場合、Date.getYear() メソッドは 104 を返し、Date.getFullYear() メソッドは 2004 を返します。

```
var today_date:Date = new Date(2004,4);
trace(today_date.getYear()); // output: 104
trace(today_date.getFullYear()); // output: 2004
```

関連項目

getFullYear (Date.getFullYear メソッド)

setDate (Date.setDate メソッド)

```
public setDate(date:Number) : Number
```

指定された Date オブジェクトの日付をローカル時間で設定し、新しい時間をミリ秒で返します。ローカル時間は、Flash Player を実行しているオペレーティングシステムによって決まります。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`date:Number` - 1 ～ 31 の整数。

戻り値

`Number` - 整数。

例

次の例では、日付を 2004 年 5 月 15 日に設定した新しい Date オブジェクトをまず作成した後、`Date.setDate()` を使って日付を 2004 年 5 月 25 日に変更します。

```
var today_date:Date = new Date(2004,4,15);
trace(today_date.getDate()); //displays 15
today_date.setDate(25);
trace(today_date.getDate()); //displays 25
```

setFullYear (Date.setFullYear メソッド)

```
public setFullYear(year:Number, [month:Number], [date:Number]) : Number
```

指定された Date オブジェクトの年をローカル時間で設定し、新しい時間をミリ秒で返します。`month` パラメータと `date` パラメータを指定すると、両方は同時にローカル時間に設定されます。ローカル時間は、Flash Player を実行しているオペレーティングシステムによって決まります。

このメソッドを呼び出しても、指定した Date オブジェクトの他のフィールドは修正されません。ただし、このメソッドを呼び出した結果として曜日が変わった場合には、`Date.getUTCDay()` と `Date.getDay()` は新しい値を返します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`year:Number` - 年を指定する 4 桁の数値です。2 桁の数値は 4 桁の年の省略形を表しません。たとえば、99 は 1999 年ではなく、99 年です。

`month:Number` (オプション) - 0 (1 月) ～ 11 (12 月) の整数です。このパラメータを省略した場合、指定した Date オブジェクトの month フィールドは変更されません。

`date:Number` (オプション) - 1 ～ 31 の数値です。このパラメータを省略した場合、指定した Date オブジェクトの date フィールドは変更されません。

戻り値

`Number` - 整数。

例

次の例では、日付を 2004 年 5 月 15 日に設定した新しい Date オブジェクトをまず作成した後、`Date.setFullYear()` を使って日付を 2002 年 5 月 15 日に変更します。

```
var my_date:Date = new Date(2004,4,15);
trace(my_date.getFullYear()); //output: 2004
my_date.setFullYear(2002);
trace(my_date.getFullYear()); //output: 2002
```

関連項目

`getUTCDay (Date.getUTCDay メソッド ), getDay (Date.getDay メソッド )`

setHours (Date.setHours メソッド)

```
public setHours(hour:Number) : Number
```

指定された Date オブジェクトの時をローカル時間で設定し、新しい時間をミリ秒で返します。ローカル時間は、Flash Player を実行しているオペレーティングシステムによって決まります。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`hour:Number` - 0 (0 時) ～ 23 (午後 11 時) の整数です。

戻り値

`Number` - 整数。

例

次の例では、日時を 2004 年 5 月 15 日午前 8:00 に設定した新しい Date オブジェクトをまず作成した後、Date.setHours() を使って時刻を午後 4:00 に変更します。

```
var my_date:Date = new Date(2004,4,15,8);
trace(my_date.getHours()); // output: 8
my_date.setHours(16);
trace(my_date.getHours()); // output: 16
```

setMilliseconds (Date.setMilliseconds メソッド)

```
public setMilliseconds(millisecond:Number) : Number
```

指定された Date オブジェクトのミリ秒をローカル時間で設定し、新しい時間をミリ秒で返します。ローカル時間は、Flash Player を実行しているオペレーティングシステムによって決まります。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`millisecond:Number` - 0 ～ 999 の整数。

戻り値

`Number` - 整数。

例

次の例では、日時を 2004 年 5 月 15 日午前 8:30 分 250 ミリ秒に設定した新しい Date オブジェクトをまず作成した後、Date.setMilliseconds() を使ってミリ秒の値を 575 に変更します。

```
var my_date:Date = new Date(2004,4,15,8,30,0,250);
trace(my_date.getMilliseconds()); // output: 250
my_date.setMilliseconds(575);
trace(my_date.getMilliseconds()); // output: 575
```

setMinutes (Date.setMinutes メソッド)

```
public setMinutes(minute:Number) : Number
```

指定された Date オブジェクトの分をローカル時間で設定し、新しい時間をミリ秒で返します。ローカル時間は、Flash Player を実行しているオペレーティングシステムによって決まります。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`minute:Number` - 0 ～ 59 の整数。

戻り値

Number - 整数。

例

次の例では、日時を 2004 年 5 月 15 日午前 8:00 に設定した新しい Date オブジェクトをまず作成した後、Date.setMinutes() を使って時刻を午前 8:30 に変更します。

```
var my_date:Date = new Date(2004,4,15,8,0);
trace(my_date.getMinutes()); // output: 0
my_date.setMinutes(30);
trace(my_date.getMinutes()); // output: 30
```

setMonth (Date.setMonth メソッド)

```
public setMonth(month:Number, [date:Number]) : Number
```

指定された Date オブジェクトの月をローカル時間で設定し、新しい時間をミリ秒で返します。ローカル時間は、Flash Player を実行しているオペレーティングシステムによって決まります。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`month:Number` - 0 (1 月) ～ 11 (12 月) の整数です。

`date:Number` (オプション) - 1 ～ 31 の整数です。このパラメータを省略した場合、指定した Date オブジェクトの date フィールドは変更されません。

戻り値

Number - 整数。

例

次の例では、日付を 2004 年 5 月 15 日に設定した新しい Date オブジェクトをまず作成した後、Date.setMonth() を使って日付を 2004 年 6 月 15 日に変更します。

```
var my_date:Date = new Date(2004,4,15);
trace(my_date.getMonth()); //output: 4
my_date.setMonth(5);
trace(my_date.getMonth()); //output: 5
```

setSeconds (Date.setSeconds メソッド)

```
public setSeconds(second:Number) : Number
```

指定された Date オブジェクトの秒をローカル時間で設定し、新しい時間をミリ秒で返します。ローカル時間は、Flash Player を実行しているオペレーティングシステムによって決まります。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`second:Number` - 0 ～ 59 の整数。

戻り値

`Number` - 整数。

例

次の例では、日時を 2004 年 5 月 15 日午前 8:00:00 に設定した新しい Date オブジェクトをまず作成した後、`Date.setSeconds()` を使って時刻を午前 8:00:45 に変更します。

```
var my_date:Date = new Date(2004,4,15,8,0,0);
trace(my_date.getSeconds()); // output: 0
my_date.setSeconds(45);
trace(my_date.getSeconds()); // output: 45
```

setTime (Date.setTime メソッド)

```
public setTime(millisecond:Number) : Number
```

指定された Date オブジェクトの日付を 1970 年 1 月 1 日 0 時からのミリ秒数で設定し、新しい時間をミリ秒で返します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`millisecond:Number` - 数値。1 月 1 日 0 時 (世界時) を 0 とする整数値です。

戻り値

`Number` - 整数。

例

次の例では、日時を 2004 年 5 月 15 日午前 8:00 に設定した新しい Date オブジェクトをまず作成した後、Date.setTime() を使って時刻を午前 8:30 に変更します。

```
var my_date:Date = new Date(2004,4,15,8,0,0);
var myDate_num:Number = my_date.getTime(); // convert my_date to milliseconds
myDate_num += 30 * 60 * 1000; // add 30 minutes in milliseconds
my_date.setTime(myDate_num); // set my_date Date object 30 minutes forward
trace(my_date.getFullYear()); // output: 2004
trace(my_date.getMonth()); // output: 4
trace(my_date.getDate()); // output: 15
trace(my_date.getHours()); // output: 8
trace(my_date.getMinutes()); // output: 30
```

setUTCDate (Date.setUTCDate メソッド)

```
public setUTCDate(date:Number) : Number
```

指定された Date オブジェクトの日付を世界時で設定し、新しい時間をミリ秒で返します。このメソッドを呼び出しても、指定した Date オブジェクトの他のフィールドは変更されません。ただし、このメソッドを呼び出した結果として曜日が変わった場合には、Date.getUTCDay() と Date.getDay() は新しい値を返すことがあります。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`date:Number` - 数値。1 ～ 31 の整数です。

戻り値

`Number` - 整数。

例

次の例では、今日の日付に設定した新しい Date オブジェクトを作成した後、Date.setUTCDate() を使って日にちの値を 10 に変更し、さらに 25 に変更します。

```
var my_date:Date = new Date();
my_date.setUTCDate(10);
trace(my_date.getUTCDate()); // output: 10
my_date.setUTCDate(25);
trace(my_date.getUTCDate()); // output: 25
```

関連項目

getUTCDay (Date.getUTCDay メソッド), getDay (Date.getDay メソッド)

setUTCFullYear (Date.setUTCFullYear メソッド)

`public setUTCFullYear(year:Number, [month:Number], [date:Number]) : Number`

指定された Date オブジェクト (*my_date*) の年を世界時で設定し、新しい時間をミリ秒で返します。

オプションとして、このメソッドは指定された Date オブジェクトが表す月日も設定できます。このメソッドを呼び出しても、指定した Date オブジェクトの他のフィールドは変更されません。ただし、このメソッドを呼び出した結果として曜日が変わった場合には、`Date.getUTCDay()` と `Date.getDay()` は新しい値を返すことがあります。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`year:Number` - 4 桁の年 (2000 など) を表す整数です。

`month:Number` (オプション) - 0 (1 月) ～ 11 (12 月) の整数です。このパラメータを省略した場合、指定した Date オブジェクトの month フィールドは変更されません。

`date:Number` (オプション) - 1 ～ 31 の整数です。このパラメータを省略した場合、指定した Date オブジェクトの date フィールドは変更されません。

戻り値

`Number` - 整数。

例

次の例では、今日の日付に設定した新しい Date オブジェクトをまず作成した後、`Date.setUTCFullYear()` を使って年の値を 2001 に変更し、さらに日付を 1995 年 5 月 25 日に変更します。

```
var my_date:Date = new Date();
my_date.setUTCFullYear(2001);
trace(my_date.getUTCFullYear()); // output: 2001
my_date.setUTCFullYear(1995, 4, 25);
trace(my_date.getUTCFullYear()); // output: 1995
trace(my_date.getUTCMonth()); // output: 4
trace(my_date.getUTCDate()); // output: 25
```

関連項目

getUTCDay (Date.getUTCDay メソッド), getDay (Date.getDay メソッド)

setUTCHours (Date.setUTCHours メソッド)

```
public setUTCHours(hour:Number, [minute:Number], [second:Number],
  [millisecond:Number]) : Number
```

指定された Date オブジェクトの時を世界時で設定し、新しい時間をミリ秒で返します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`hour:Number` - 数値。0 (0 時) ～ 23 (午後 11 時) の整数です。

`minute:Number` (オプション) - 数値。0 ～ 59 の整数です。このパラメータを省略した場合、指定した Date オブジェクトの minutes フィールドは変更されません。

`second:Number` (オプション) - 数値。0 ～ 59 の整数です。このパラメータを省略した場合、指定した Date オブジェクトの seconds フィールドは変更されません。

`millisecond:Number` (オプション) - 数値。0 ～ 999 の整数です。このパラメータを省略した場合、指定した Date オブジェクトの milliseconds フィールドは変更されません。

戻り値

`Number` - 整数。

例

次の例では、今日の日付に設定した新しい Date オブジェクトを作成した後、`Date.setUTCHours()` を使って時刻を午前 8:30 に変更し、さらに午後 5:30:47 に変更します。

```
var my_date:Date = new Date();
my_date.setUTCHours(8,30);
trace(my_date.getUTCHours()); // output: 8
trace(my_date.getUTCMinutes()); // output: 30
my_date.setUTCHours(17,30,47);
trace(my_date.getUTCHours()); // output: 17
trace(my_date.getUTCMinutes()); // output: 30
trace(my_date.getUTCSeconds()); // output: 47
```

setUTCMilliseconds (Date.setUTCMilliseconds メソッド)

```
public setUTCMilliseconds(millisecond:Number) : Number
```

指定された Date オブジェクトのミリ秒を世界時で設定し、新しい時間をミリ秒で返します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`millisecond:Number` - 0 ～ 999 の整数。

戻り値

`Number` - 整数。

例

次の例では、日時を 2004 年 5 月 15 日午前 8:30 分 250 ミリ秒に設定した新しい Date オブジェクトをまず作成した後、`Date.setUTCMilliseconds()` を使ってミリ秒の値を 575 に変更します。

```
var my_date:Date = new Date(2004,4,15,8,30,0,250);
trace(my_date.getUTCMilliseconds()); // output: 250
my_date.setUTCMilliseconds(575);
trace(my_date.getUTCMilliseconds()); // output: 575
```

setUTCMinutes (Date.setUTCMinutes メソッド)

```
public setUTCMinutes(minute:Number, [second:Number], [millisecond:Number]) :
  Number
```

指定された Date オブジェクトの分を世界時で設定し、新しい時間をミリ秒で返します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`minute:Number` - 0 ～ 59 の整数。

`second:Number` (オプション) - 0 ～ 59 の整数です。このパラメータを省略した場合、指定した Date オブジェクトの seconds フィールドは変更されません。

`millisecond:Number` (オプション) - 0 ～ 999 の整数です。このパラメータを省略した場合、指定した Date オブジェクトの milliseconds フィールドは変更されません。

戻り値

`Number` - 整数。

例

次の例では、日時を 2004 年 5 月 15 日午前 8:00 に設定した新しい Date オブジェクトをまず作成した後、Date.setUTCMinutes() を使って時刻を午前 8:30 に変更します。

```
var my_date:Date = new Date(2004,4,15,8,0);
trace(my_date.getUTCMinutes()); // output: 0
my_date.setUTCMinutes(30);
trace(my_date.getUTCMinutes()); // output: 30
```

setUTCMonth (Date.setUTCMonth メソッド)

```
public setUTCMonth(month:Number, [date:Number]) : Number
```

指定された Date オブジェクトの月を世界時で設定し (必要であれば日付も設定可能)、新しい時間をミリ秒で返します。このメソッドを呼び出しても、指定した Date オブジェクトの他のフィールドは変更されません。ただし、date パラメータに値を指定した結果として曜日が変わった場合は、Date.getUTCDay() と Date.getDay() は新しい値を返すことがあります。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

month:Number - 0 (1 月) ～ 11 (12 月) の整数です。

date:Number (オプション) - 1 ～ 31 の整数です。このパラメータを省略した場合、指定した Date オブジェクトの date フィールドは変更されません。

戻り値

Number - 整数。

例

次の例では、日付を 2004 年 5 月 15 日に設定した新しい Date オブジェクトをまず作成した後、Date.setMonth() を使って日付を 2004 年 6 月 15 日に変更します。

```
var today_date:Date = new Date(2004,4,15);
trace(today_date.getUTCMonth()); // output: 4
today_date.setUTCMonth(5);
trace(today_date.getUTCMonth()); // output: 5
```

関連項目

getUTCDay (Date.getUTCDay メソッド), getDay (Date.getDay メソッド)

setUTCSeconds (Date.setUTCSeconds メソッド)

```
public setUTCSeconds(second:Number, [millisecond:Number]) : Number
```

指定された Date オブジェクトの秒を世界時で設定し、新しい時間をミリ秒で返します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`second:Number` - 0 ～ 59 の整数。

`millisecond:Number` (オプション) - 0 ～ 999 の整数です。このパラメータを省略した場合、指定した Date オブジェクトの milliseconds フィールドは変更されません。

戻り値

`Number` - 整数。

例

次の例では、日時を 2004 年 5 月 15 日午前 8:00:00 に設定した新しい Date オブジェクトをまず作成した後、`Date.setSeconds()` を使って時刻を午前 8:30:45 に変更します。

```
var my_date:Date = new Date(2004,4,15,8,0,0);
trace(my_date.getUTCSeconds()); // output: 0
my_date.setUTCSeconds(45);
trace(my_date.getUTCSeconds()); // output: 45
```

setYear (Date.setYear メソッド)

```
public setYear(year:Number) : Number
```

指定された Date オブジェクトの年をローカル時間で設定し、新しい時間をミリ秒で返します。ローカル時間は、Flash Player を実行しているオペレーティングシステムによって決まります。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`year:Number` - 年を表す数値です。`year` が 0 ～ 99 の整数である場合、`setYear` は、1900 + `year` の値を年として設定します。それ以外の場合は、`year` パラメータの値を年として設定します。

戻り値

`Number` - 整数。

例

次の例では、日付を 2004 年 5 月 25 日に設定した新しい Date オブジェクトを作成した後、setYear() を使って年を 1999 に変更し、さらに 2003 に変更します。

```
var my_date:Date = new Date(2004,4,25);
trace(my_date.getYear()); // output: 104
trace(my_date.getFullYear()); // output: 2004
my_date.setYear(99);
trace(my_date.getYear()); // output: 99
trace(my_date.getFullYear()); // output: 1999
my_date.setYear(2003);
trace(my_date.getYear()); // output: 103
trace(my_date.getFullYear()); // output: 2003
```

toString (Date.toString メソッド)

```
public toString() : String
```

指定された Date オブジェクトのストリング値を読み取り可能な形式で返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

String - ストリング。

例

次の例では、dateOfBirth_date という Date オブジェクト内の情報をストリングとして返します。trace ステートメントからの出力はローカル時間に基づいて変化します。太平洋標準時の夏時間の場合、出力は世界時よりも 7 時間前の値になります (Mon Aug 12 18:15:00 GMT-0700 1974)。

```
var dateOfBirth_date:Date = new Date(74, 7, 12, 18, 15);
trace (dateOfBirth_date);
trace (dateOfBirth_date.toString());
```

UTC (Date.UTC メソッド)

```
public static UTC(year:Number, month:Number, [date:Number], [hour:Number],
  [minute:Number], [second:Number], [millisecond:Number]) : Number
```

1970 年 1 月 1 日 0 時 (世界時) からパラメータで指定された時刻までのミリ秒数を返します。このメソッドは、特定の Date オブジェクトからでなく Date オブジェクトコンストラクタから呼び出される静的なメソッドです。このメソッドを使用すると、世界時を採用する Date オブジェクトを作成できます。一方、Date コンストラクタはローカル時間を採用します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`year:Number` - 年を表す 4 桁の数値 (2000 など) です。

`month:Number` - 0 (1 月) ～ 11 (12 月) の整数です。

`date:Number` (オプション) - 1 ～ 31 の整数です。

`hour:Number` (オプション) - 0 (0 時) ～ 23 (午後 11 時) の整数です。

`minute:Number` (オプション) - 0 ～ 59 の整数です。

`second:Number` (オプション) - 0 ～ 59 の整数です。

`millisecond:Number` (オプション) - 0 ～ 999 の整数です。

戻り値

`Number` - 整数。

例

次の例では、世界時で定義された新しい Date オブジェクト `maryBirthday_date` を作成します。これは、コンストラクタメソッド `new Date` を使用して、世界時に対応した Date オブジェクトを作成する例です。出力は、ローカル時間に基づいて変化します。太平洋標準時の夏時間の場合、出力は UTC よりも 7 時間前の値になります (Sun Aug 11 17:00:00 GMT-0700 1974)。

```
var maryBirthday_date:Date = new Date(Date.UTC(1974, 7, 12));
trace(maryBirthday_date);
```

valueOf (Date.valueOf メソッド)

```
public valueOf() : Number
```

この `Date` に関して、1970 年 1 月 1 日 0 時 (世界時) からのミリ秒数を返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

`Number` - ミリ秒数。

DisplacementMapFilter (flash.filters.DisplacementMapFilter)

```
Object
  |
  +-BitmapFilter
     |
     +-flash.filters.DisplacementMapFilter
```

```
public class DisplacementMapFilter
extends BitmapFilter
```

DisplacementMapFilter クラスは、指定された BitmapData オブジェクト (置き換えマップイメージという) のピクセル値を使用して、MovieClip インスタンスなど、ステージ上のオブジェクトの置き換え (変位) を実行します。このフィルタを使用すると、BitmapData や MovieClip のインスタンスでワープ効果や斑点効果を実現できます。

フィルタの使い方は、フィルタの適用先オブジェクトによって異なります。

実行時にムービークリップにフィルタを適用する場合は、`filters` プロパティを使用します。オブジェクトの `filters` プロパティを設定してもオブジェクトは変更されません。また、`filters` プロパティをクリアすることで元に戻すことができます。

BitmapData インスタンスにフィルタを適用するには、`BitmapData.applyFilter()` メソッドを使用します。BitmapData オブジェクトの `applyFilter()` を呼び出すと、その BitmapData オブジェクトは変更されて元に戻せなくなります。

イメージやビデオにもオーサリング時にフィルタ効果を適用できます。詳細については、オーサリングのマニュアルを参照してください。

ムービークリップやボタンにフィルタを適用する場合は、ムービークリップやボタンの `cacheAsBitmap` プロパティを `true` に設定します。すべてのフィルタをクリアすると、`cacheAsBitmap` は元の値に戻ります。

このフィルタでは次の式を使用します。

```
dstPixel[x, y] = srcPixel[x + ((componentX(x, y) - 128) * scaleX) / 256, y +
  ((componentY(x, y) - 128) * scaleY) / 256]
```

`componentX(x, y)` は、`(x - mapPoint.x ,y - mapPoint.y)` の `mapBitmap` プロパティから `componentX` カラー値を取得します。

フィルタで使用するマップイメージは、ステージの拡大・縮小率に一致するように拡大・縮小されます。オブジェクト自体が拡大・縮小される場合には拡大・縮小されません。

このフィルタは、ステージの拡大・縮小をサポートしていますが、通常の拡大・縮小、回転、傾斜はサポートしていません。オブジェクト自体を拡大・縮小する場合 (*x* 方向の尺度と *y* 方向の尺度が 100% でない場合)、フィルタ効果は拡大・縮小されません。フィルタ効果が拡大・縮小するのは、ステージをズームインする場合のみです。

これが DisplacementMapFilter クラスの動作方法です。ターゲットビットマップ内の各ピクセル (*x*,*y*) に対し、DisplacementMapFilter クラスは次のように動作します。

- マップビットマップ内の (*x*,*y*) からカラーを取得します。
- このカラーに基づいてオフセットを計算します。
- ソースビットマップ内で、このオフセット位置 (*x* + *dx* , *y* + *dy*) を調べます。
- 境界に関する条件が満たされる場合、ターゲット (*x*,*y*) にこのピクセルを記述します。

結果として得られるイメージの幅または高さが 2880 ピクセルを超える場合、フィルタは適用されません。たとえば、フィルタが適用されたサイズの大きいムービークリップをズームインするとき、結果として得られるイメージが 2880 ピクセルの制限に達する場合は、フィルタがオフになります。

対応バージョン： ActionScript 1.0、Flash Player 8

関連項目

applyFilter (BitmapData.applyFilter メソッド), filters (MovieClip.filters プロパティ), cacheAsBitmap (MovieClip.cacheAsBitmap プロパティ)

プロパティ一覧

オプション	プロパティ	説明
	alpha:Number	範囲外置き換えの場合に使用するアルファ透明度値を指定します。
	color:Number	範囲外置き換えの場合に使用する色を指定します。
	componentX:Number	*x* の結果を変位させるために、どのカラーチャンネルをマップイメージで使用するかを指定します。
	componentY:Number	*y* の結果を変位させるために、どのカラーチャンネルをマップイメージで使用するかを指定します。
	mapBitmap:BitmapData	置き換えマップデータが含まれる BitmapData オブジェクトです。
	mapPoint:Point	マップイメージの左上隅を基準としたターゲットムービークリップの左上隅のオフセットが含まれる flash.geom.Point 値です。
	mode:String	フィルタのモードです。
	scaleX:Number	マップ計算の *x* 置き換え結果を拡大・縮小する場合に使用する乗数です。
	scaleY:Number	マップ計算の *y* 置き換え結果を拡大・縮小する場合に使用する乗数です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター覧

署名	説明
DisplacementMapFilter (mapBitmap:BitmapData, mapPoint:Point, componentX:Number, componentY:Number, scaleX:Number, scaleY:Number, [mode:String], [color:Number], [alpha:Number])	指定されたパラメータで DisplacementMapFilter インスタンスを初期化します。

メソッド一覧

オプション	署名	説明
	clone() : DisplacementMapFilter	このフィルタオブジェクトのコピーを返します。

BitmapFilter クラスから継承されるメソッド

clone (BitmapFilter.clone メソッド)

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

alpha (DisplacementMapFilter.alpha プロパティ)

```
public alpha : Number
```

範囲外置き換えの場合に使用するアルファ透明度値を指定します。この値には、0.0 ～ 1.0 の正規化した値を指定します。たとえば 0.25 を指定すると、透明値 25% が設定されます。デフォルト値は 0 です。このプロパティは、mode プロパティが 3，COLOR に設定された場合に使用します。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip オブジェクト filteredMc がクリックされたとき、範囲外の alpha プロパティを 0x00FF00 に変更します。

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.scaleY = 25;
  filter.mode = "color";
  filter.alpha = 0.25;
  this.filters = new Array(filter);
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap, new
  Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number, type:String,
  hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);
```

```
  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55, 0x99],
  matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal", bmp.rectangle,
  true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300, 80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

clone (DisplacementMapFilter.clone メソッド)

```
public clone() : DisplacementMapFilter
```

このフィルタオブジェクトのコピーを返します。

対応バージョン : ActionScript 1.0、Flash Player 8

戻り値

`DisplacementMapFilter` - 元のインスタンスと同じプロパティをすべて備えた新しい DisplacementMapFilter インスタンス。

例

次の例では、3 つの DisplacementMapFilter オブジェクトを作成して比較します。`filter_1` は、`DisplacementMapFilter` コンストラクタを使用して作成されます。`filter_2` は、`filter_1` と等しい値に設定することにより作成されます。`clonedFilter` は、`filter_1` のクローンを作成することによって作成されます。`filter_2` が `filter_1` に等しいと評価されても、`filter_1` と同じ値を含んでいる `clonedFilter` が等しいと評価されないことに注意してください。

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000, "radial",
  true);

var filter_1:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap, new
  Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);
var filter_2:DisplacementMapFilter = filter_1;
var clonedFilter:DisplacementMapFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false

for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> alpha: 0
  // >> color: 0
  // >> mode: wrap
  // >> scaleY: 10
  // >> scaleX: 10
  // >> componentY: 1
  // >> componentX: 1
  // >> mapPoint: (-30, -30)
  // >> mapBitmap: [object Object]
}

for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
  // >> alpha: 0
  // >> color: 0
  // >> mode: wrap
  // >> scaleY: 10
  // >> scaleX: 10
  // >> componentY: 1
  // >> componentX: 1
  // >> mapPoint: (-30, -30)
  // >> mapBitmap: [object Object]
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number, type:String,
  hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);
```

```
  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55, 0x99],
  matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal", bmp.rectangle,
  true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}
```

filter_1、filter_2、および clonedFilter の関係をさらに詳しく示すために、次の例では、filter_1 の mode プロパティを変更します。mode を変更すると、clone() メソッドは filter_1 の値を参照する代わりに、それらの値に基づいて新しいインスタンスを作成します。

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000, "radial",
  true);

var filter_1:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap, new
  Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);
var filter_2:DisplacementMapFilter = filter_1;
var clonedFilter:DisplacementMapFilter = filter_1.clone();

trace(filter_1.mode); // wrap
trace(filter_2.mode); // wrap
trace(clonedFilter.mode); // wrap

filter_1.mode = "ignore";

trace(filter_1.mode); // ignore
trace(filter_2.mode); // ignore
trace(clonedFilter.mode); // wrap

function createGradientBitmap(w:Number, h:Number, bgColor:Number, type:String,
  hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);
```

```
  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55, 0x99],
  matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal", bmp.rectangle,
  true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}
```

color (DisplacementMapFilter.color プロパティ)

```
public color : Number
```

範囲外置き換えの場合に使用する色を指定します。置き換えの有効範囲は 0.0 ～ 1.0 で、値は 16 進数形式です。color のデフォルト値は 0 です。このプロパティは、mode プロパティが 3，COLOR に設定された場合に使用します。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip オブジェクト filteredMc がクリックされたとき、範囲外の color プロパティを 0x00FF00 に変更します。

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.scaleY = 25;
  filter.mode = "color";
  filter.alpha = .25;
  filter.color = 0x00FF00;
  this.filters = new Array(filter);
}
```

```
function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap, new
  Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number, type:String,
  hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55, 0x99],
  matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal", bmp.rectangle,
  true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300, 80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

componentX (DisplacementMapFilter.componentX プロパティ)

```
public componentX : Number
```

x の結果を変位させるために、どのカラーチャンネルをマップイメージで使用するかを指定します。可能な値は、1 (赤)、2 (緑)、4 (青)、8 (アルファ) です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip `filteredMc` がクリックされたときに、その `componentX` プロパティを変更します。値が 1 から 4 に変更され、それに応じてカラーチャンネルが赤から青に変わります。

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.componentX = 4;
  this.filters = new Array(filter);
}


function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap, new
  Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number, type:String,
  hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);
```

```
  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55, 0x99],
  matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal", bmp.rectangle,
  true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300, 80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

関連項目

BitmapData (flash.display.BitmapData)

componentY (DisplacementMapFilter.componentY プロパティ)

```
public componentY : Number
```

y の結果を変位させるために、どのカラーチャンネルをマップイメージで使用するかを指定します。可能な値は、1 (赤)、2 (緑)、4 (青)、8 (アルファ) です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip `filteredMc` がクリックされたときに、その `componentY` プロパティを変更します。値が 1 から 4 に変更され、それに応じてカラーチャンネルが赤から青に変わります。

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;
```

```
var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.componentY = 4;
  this.filters = new Array(filter);
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap, new
  Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number, type:String,
  hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55, 0x99],
  matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal", bmp.rectangle,
  true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}
```

```
function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300, 80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

関連項目

BitmapData (flash.display.BitmapData)

DisplacementMapFilter コンストラクタ

```
public DisplacementMapFilter(mapBitmap:BitmapData, mapPoint:Point ,
  componentX:Number, componentY:Number, scaleX:Number, scaleY:Number,
  [mode:String], [color:Number], [alpha:Number])
```

指定されたパラメータで DisplacementMapFilter インスタンスを初期化します。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

`mapBitmap:BitmapData` - 置き換えマップデータが含まれる BitmapData オブジェクトです。

`mapPoint:Point` - マップイメージの左上隅を基準としたターゲットムービークリップの左上隅のオフセットが含まれる `flash.geom.Point` 値です。

`componentX:Number` - *x* の結果を変位させるために、どのカラーチャンネルをマップイメージで使用するかを指定します。可能な値は次のとおりです。

- 1（赤）
- 2（緑）
- 4（青）
- 8（アルファ）

`componentY:Number` - *y* の結果を変位させるために、どのカラーチャンネルをマップイメージで使用するかを指定します。可能な値は次のとおりです。

- 1（赤）
- 2（緑）
- 4（青）
- 8（アルファ）

`scaleX:Number` - マップ計算の *x* 置き換え結果を拡大・縮小する場合に使用する乗数。

`scaleY:Number` - マップ計算の *y* 置き換え結果を拡大・縮小する場合に使用する乗数。

`mode:String` (オプション) - フィルタのモード。可能な値は次のとおりです。

- `"wrap"` – 置き換え値をソースイメージの反対側で折り返します。
- `"clamp"` – 置き換え値をソースイメージのエッジにクランプします。
- `"ignore"` – 置き換え値が範囲外である場合、その置き換えを無視して、ソースピクセルを使用します。
- `"color"` – 置き換え値がイメージの外にある場合、フィルタの `alpha` プロパティと `color` プロパティで構成されるピクセル値を置き換えます。

`color:Number` (オプション) - 範囲外置き換えの場合に使用する色を指定します。置き換えの有効範囲は 0.0 ～ 1.0 です。`mode` が `"color"` に設定されている場合にこのパラメータを使用します。

`alpha:Number` (オプション) - 範囲外置き換えの場合に使用するアルファ値を指定します。この値には、0.0 ～ 1.0 の正規化した値を指定します。たとえば .25 を指定すると、透明値 25% が設定されます。デフォルトは 0 です。このパラメータは、`mode` が `"color"` に設定されている場合に使用します。

例

次のコンストラクタ関数は、フィルタの新しいインスタンスを作成します。

```
myFilter = new flash.filters.DisplacementMapFilter (mapBitmap, mapPoint,
  componentX, componentY, scale, [mode], [color], [alpha])
```

次の例では、放射状グラデーションで新しい DisplacementMapFilter をインスタンス化し、MovieClip オブジェクト `txtBlock` を含むテキストに適用します。

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000, "radial");

var mapPoint:Point = new Point(-30, -30);
var componentX:Number = 1;
var componentY:Number = 1;
var scaleX:Number = 10;
var scaleY:Number = 10;
var mode:String = "wrap";
var color:Number = 0x000000;
var alpha:Number = 0x000000;

var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  mapPoint, componentX, componentY, scaleX, scaleY, mode, color, alpha);

var txtBlock:MovieClip = createTextBlock();
txtBlock._x = 30;
txtBlock._y = 30;
```

```
txtBlock.filters = new Array(filter);

function createGradientBitmap(w:Number, h:Number, bgColor:Number, type:String,
  hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55, 0x99],
  matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal", bmp.rectangle,
  true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
    txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300, 80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

mapBitmap (DisplacementMapFilter.mapBitmap プロパティ)

```
public mapBitmap : BitmapData
```

置き換えマップデータが含まれる BitmapData オブジェクトです。

mapBitmap プロパティの値を直接変更することはできません。このプロパティを変更するには、mapBitmap への参照を取得し、その参照を変更した後、mapBitmap をその参照に設定する必要があります。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip `filteredMc` がクリックされたときに、その `mapBitmap` プロパティを変更します。

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();
var scope:Object = this;

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.mapBitmap = scope.createGradientBitmap(300, 80, 0xFF000000, "linear");
  this.filters = new Array(filter);
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap, new
  Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number, type:String,
  hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55, 0x99],
  matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;
```

```
  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal", bmp.rectangle,
  true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300, 80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

関連項目

BitmapData (flash.display.BitmapData)

mapPoint (DisplacementMapFilter.mapPoint プロパティ)

public mapPoint : Point

マップイメージの左上隅を基準としたターゲットムービークリップの左上隅のオフセットが含まれる flash.geom.Point 値です。

mapPoint プロパティの値を直接変更することはできません。このプロパティを変更するには、mapPoint への参照を取得し、その参照を変更した後、mapPoint をその参照に設定する必要があります。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip filteredMc がクリックされたときに、その mapPoint プロパティを変更します。

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();
```

```
filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.mapPoint = new Point(-30, -40);
  this.filters = new Array(filter);
  this._x = 30;
  this._y = 40;
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap, new
  Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number, type:String,
  hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55, 0x99],
  matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal", bmp.rectangle,
  true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}
```

```
function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300, 80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

関連項目

Point (flash.geom.Point)

mode (DisplacementMapFilter.mode プロパティ)

```
public mode : String
```

フィルタのモードです。可能な値は次のとおりです。

- "wrap" – 置き換え値をソースイメージの反対側で折り返します。これがデフォルト値です。
- "clamp" – 置き換え値をソースイメージのエッジにクランプします。
- "ignore" – 置き換え値が範囲外である場合、その置き換えを無視して、ソースピクセルを使用します。
- "color" – 置き換え値がイメージの外にある場合、フィルタの alpha プロパティと color プロパティで構成されるピクセル値を置き換えます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、範囲外にある置き換え値を作成するのに scaleY を変更した後、既存の MovieClip filteredMc がクリックされたときに、その mode プロパティを ignore に変更します。

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.scaleY = 25;
  filter.mode = "ignore";
  this.filters = new Array(filter);
}
```

```
function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap, new
  Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number, type:String,
  hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55, 0x99],
  matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal", bmp.rectangle,
  true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300, 80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

scaleX (DisplacementMapFilter.scaleX プロパティ)

public scaleX : Number

マップ計算の *x* 置き換え結果を拡大・縮小する場合に使用する乗数です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip filteredMc がクリックされたときに、その scaleX プロパティを変更します。

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.scaleX = 5;
  this.filters = new Array(filter);
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap, new
  Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number, type:String,
  hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55, 0x99],
  matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
```

```
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal", bmp.rectangle,
  true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300, 80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

scaleY (DisplacementMapFilter.scaleY プロパティ)

```
public scaleY : Number
```

マップ計算の *y* 置き換え結果を拡大・縮小する場合に使用する乗数です。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、既存の MovieClip `filteredMc` がクリックされたときに、その `scaleY` プロパティを変更します。

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.scaleY = 5;
  this.filters = new Array(filter);
}
```

```
function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap, new
  Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number, type:String,
  hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55, 0x99],
  matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal", bmp.rectangle,
  true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300, 80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

DropShadowFilter (flash.filters.DropShadowFilter)

```
Object
  |
  +-BitmapFilter
     |
     +-flash.filters.DropShadowFilter
```

```
public class DropShadowFilter
extends BitmapFilter
```

DropShadowFilter クラスを使用すると、Flash の各種オブジェクトにドロップシャドウ (陰影) を追加できます。ドロップシャドウのスタイルには複数のオプションがあり、内側シャドウ、外側シャドウ、ノックアウトモードなどがあります。

フィルタの使い方は、フィルタの適用先オブジェクトによって異なります。

- 実行時にムービークリップ、テキストフィールド、ボタンにフィルタを適用する場合は、filters プロパティを使用します。オブジェクトの filters プロパティを設定してもオブジェクトは変更されません。また、filters プロパティをクリアすることで元に設定を戻すことができます。
- BitmapData インスタンスにフィルタを適用するには、BitmapData.applyFilter() メソッドを使用します。BitmapData オブジェクトで applyFilter() を呼び出すことによって、ソース BitmapData オブジェクトとフィルタオブジェクトが取得され、フィルタを適用した結果として得られるイメージが生成されます。

イメージやビデオにもオーサリング時にフィルタ効果を適用できます。詳細については、オーサリングのマニュアルを参照してください。

ムービークリップやボタンにフィルタを適用する場合は、ムービークリップやボタンの cacheAsBitmap プロパティを true に設定します。すべてのフィルタをクリアすると、cacheAsBitmap は元の値に戻ります。

このフィルタはステージの拡大・縮小に対応していますが、通常の拡大・縮小、回転、傾斜には対応していません。オブジェクト自体が拡大・縮小される場合 (_xscale と _yscale が 100% ではない場合)、フィルタ効果は拡大・縮小されません。フィルタ効果が拡大・縮小するのは、ステージをズームインする場合のみです。

結果として得られるイメージの幅または高さが 2880 ピクセルを超える場合、フィルタは適用されません。たとえば、フィルタが適用されたサイズの大きいムービークリップをズームインするとき、結果として得られるイメージが 2880 ピクセルの制限を超える場合は、フィルタがオフになります。

対応バージョン： ActionScript 1.0、Flash Player 8

関連項目

filters (MovieClip.filters プロパティ), cacheAsBitmap (MovieClip.cacheAsBitmap プロパティ), filters (Button.filters プロパティ), cacheAsBitmap (Button.cacheAsBitmap プロパティ), filters (TextField.filters プロパティ), applyFilter (BitmapData.applyFilter メソッド)

プロパティ一覧

オプション	プロパティ	説明
	alpha:Number	シャドウカラーのアルファ透明度の値です。
	angle:Number	シャドウの角度です。
	blurX:Number	水平方向のぼかし量です。
	blurY:Number	垂直方向のぼかし量です。
	color:Number	シャドウの色です。
	distance:Number	影のオフセット距離 (ピクセル単位) です。
	hideObject:Boolean	オブジェクトが非表示であるかどうかを示します。
	inner:Boolean	影が内側の影であるかどうかを指定します。
	knockout:Boolean	true の場合は、ノックアウト効果を適用します。その結果、オブジェクトは完全に透明になり、ドキュメントの背景色で表示されます。
	quality:Number	フィルタを適用する回数です。
	strength:Number	インプリントやスプレッドの長さです。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター一覧

署名	説明
DropShadowFilter ([distance:Number], [angle:Number], [color:Number], [alpha:Number], [blurX:Number], [blurY:Number], [strength:Number], [quality:Number], [inner:Boolean], [knockout:Boolean], [hideObject:Boolean])	指定されたパラメータで新しい DropShadowFilter インスタンスを作成します。

メソッド一覧

オプション	署名	説明
	clone() : DropShadowFilter	このフィルタオブジェクトのコピーを返します。

BitmapFilter クラスから継承されるメソッド

clone (BitmapFilter.clone メソッド)

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

alpha (DropShadowFilter.alpha プロパティ)

```
public alpha : Number
```

シャドウカラーのアルファ透明度の値です。0 ～ 1 の範囲の値を指定できます。たとえば 0.25 と指定すると、透明度は 25% になります。デフォルト値は 1 です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その alpha プロパティを変更します。

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowAlpha");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.alpha = 0.4;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, 0.8, 16,
  16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

angle (DropShadowFilter.angle プロパティ)

```
public angle : Number
```

シャドウの角度です。指定できる値は 0 ～ 360° (浮動小数) です。デフォルト値は 45 です。

角度の値は、オブジェクトに対する架空の光源の角度を表し、オブジェクトに対する効果の相対位置を決定します。距離が 0 に設定された場合、効果がオブジェクトからオフセットされないため、角度プロパティは適用されません。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その angle プロパティを変更します。

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowAngle");
mc.onRelease = function() {
   var filter:DropShadowFilter = this.filters[0];
   filter.angle = 135;
   this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
   var art:MovieClip = this.createEmptyMovieClip(name,
   this.getNextHighestDepth());
   var w:Number = 100;
   var h:Number = 100;
   art.beginFill(0x003366);
   art.lineTo(w, 0);
   art.lineTo(w, h);
   art.lineTo(0, h);
   art.lineTo(0, 0);
   art._x = 20;
   art._y = 20;

   var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, 0.8, 16,
   16, 1, 3, false, false, false);
   var filterArray:Array = new Array();
   filterArray.push(filter);
   art.filters = filterArray;
   return art;
}
```

blurX (DropShadowFilter.blurX プロパティ)

```
public blurX : Number
```

水平方向のぼかし量。指定できる値は 0 ～ 255 (浮動小数) です。デフォルト値は 4 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その blurX プロパティを変更します。

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowBlurX");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.blurX = 40;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, 0.8, 16,
  16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

blurY (DropShadowFilter.blurY プロパティ)

```
public blurY : Number
```

垂直方向のぼかし量。指定できる値は 0 ～ 255 (浮動小数) です。デフォルト値は 4 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その blurY プロパティを変更します。

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowBlurY");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.blurY = 40;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, 0.8, 16,
  16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

clone (DropShadowFilter.clone メソッド)

```
public clone() : DropShadowFilter
```

このフィルタオブジェクトのコピーを返します。

対応バージョン： ActionScript 1.0、Flash Player 8

戻り値

`DropShadowFilter` - 元のインスタンスのプロパティをすべて備えた新しい DropShadowFilter インスタンス。

例

次の例では、3 つの DropShadowFilter オブジェクトを作成して比較します。`filter_1` は、DropShadowFilter コンストラクタを使用して作成されます。`filter_2` は、`filter_1` と等しい値に設定することにより作成されます。`clonedFilter` は、`filter_1` のクローンを作成することにより作成されます。`filter_2` が `filter_1` に等しいと評価されても、`filter_1` と同じ値を含んでいる `clonedFilter` が等しいと評価されないことに注意してください。

```
import flash.filters.DropShadowFilter;

var filter_1:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8, 16,
  16, 1, 3, false, false, false);
var filter_2:DropShadowFilter = filter_1;
var clonedFilter:DropShadowFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false

for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> hideObject: false
  // >> strength: 1
  // >> blurY: 16
  // >> blurX: 16
  // >> knockout: false
  // >> inner: false
  // >> quality: 3
  // >> alpha: 0.8
  // >> color: 0
  // >> angle: 45
  // >> distance: 15
}
```

```
for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
  // >> hideObject: false
  // >> strength: 1
  // >> blurY: 16
  // >> blurX: 16
  // >> knockout: false
  // >> inner: false
  // >> quality: 3
  // >> alpha: 0.8
  // >> color: 0
  // >> angle: 45
  // >> distance: 15
}
```

filter_1、filter_2、および clonedFilter の関係をさらに詳しく示すために、次の例では、filter_1 の knockout プロパティを変更します。knockout を変更することは、clone() メソッドによって、filter_1 を参照する代わりにその値に基づいて新しいインスタンスが作成されることを示します。

```
import flash.filters.DropShadowFilter;

var filter_1:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, 0.8, 16,
  16, 1, 3, false, false, false);
var filter_2:DropShadowFilter = filter_1;
var clonedFilter:DropShadowFilter = filter_1.clone();

trace(filter_1.knockout); // false
trace(filter_2.knockout); // false
trace(clonedFilter.knockout); // false

filter_1.knockout = true;

trace(filter_1.knockout); // true
trace(filter_2.knockout); // true
trace(clonedFilter.knockout); // false
```

color (DropShadowFilter.color プロパティ)

public color : Number

シャドウの色です。有効な値は 16 進数形式 *0xRRGGBB* です。デフォルト値は 0x000000 です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その color プロパティを変更します。

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowColor");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.color = 0xFF0000;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, 0.8, 16,
  16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

distance (DropShadowFilter.distance プロパティ)

```
public distance : Number
```

影のオフセット距離 (ピクセル単位) です。デフォルト値は 4 (浮動小数) です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その distance プロパティを変更します。

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowDistance");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.distance = 40;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, 0.8, 16,
  16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

DropShadowFilter コンストラクタ

```
public DropShadowFilter([distance:Number], [angle:Number], [color:Number],
  [alpha:Number], [blurX:Number], [blurY:Number], [strength:Number],
  [quality:Number], [inner:Boolean], [knockout:Boolean], [hideObject:Boolean])
```

指定されたパラメータで新しい DropShadowFilter インスタンスを作成します。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

`distance:Number` (オプション) - シャドウのオフセット距離 (ピクセル単位)。デフォルト値は 4 (浮動小数) です。

`angle:Number` (オプション) - 0 ～ 360° で表されるシャドウの角度 (浮動小数値)。デフォルト値は 45 です。

`color:Number` (オプション) - シャドウのカラー (16 進数形式 *0xRRGGBB*)。デフォルト値は 0x000000 です。

`alpha:Number` (オプション) - シャドウカラーのアルファ透明度の値。0 ～ 1 の範囲の値を指定できます。たとえば 0.25 と指定すると、透明度は 25% になります。デフォルト値は 1 です。

`blurX:Number` (オプション) - 水平方向のぼかし量です。指定できる値は 0 ～ 255 (浮動小数) です。デフォルト値は 4 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

`blurY:Number` (オプション) - 垂直方向のぼかし量です。指定できる値は 0 ～ 255 (浮動小数) です。デフォルト値は 4 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

`strength:Number` (オプション) - インプリントやスプレッドの長さです。値が大きいほど、濃い色がインプリントされるので、シャドウと背景との間のコントラストが強くなります。指定できる値は 0 ～ 255 で、デフォルトは 1 です。

`quality:Number` (オプション) - フィルタを適用する回数。有効な値は 0 ～ 15 です。デフォルト値は 1 (低品質) です。値 2 は標準の品質であり、値 3 は高品質です。

`inner:Boolean` (オプション) - 影が内側の影であるかどうかを指定します。`true` の場合は、内部シャドウを示します。デフォルトは `false` (外側シャドウ) で、オブジェクトの外周にあるシャドウを示します。

`knockout:Boolean` (オプション) - `true` の場合は、ノックアウト効果を適用します。その結果、オブジェクトは完全に透明になり、ドキュメントの背景色で表示されます。デフォルトは `false` (ノックアウトなし) です。

`hideObject:Boolean` (オプション) - オブジェクトが非表示であるかどうかを示します。`true` を指定すると、オブジェクト自体は描画されず、シャドウだけが表示されます。デフォルトは `false` で、オブジェクトが表示されます。

例

次の例では、DropShadowFilter の新しいインスタンスを作成し、そのインスタンスにフラットな矩形シェイプを適用します。

```
import flash.filters.DropShadowFilter;
var art:MovieClip = createRectangle(100, 100, 0x003366,
  "gradientGlowFilterExample");
var distance:Number = 20;
var angleInDegrees:Number = 45;
var color:Number = 0x000000;
var alpha:Number = 0.8;
var blurX:Number = 16;
var blurY:Number = 16;
var strength:Number = 1;
var quality:Number = 3;
var inner:Boolean = false;
var knockout:Boolean = false;
var hideObject:Boolean = false;

var filter:DropShadowFilter = new DropShadowFilter(distance,
                              angleInDegrees,
                              color,
                              alpha,
                              blurX,
                              blurY,
                              strength,
                              quality,
                              inner,
                              knockout,
                              hideObject);
var filterArray:Array = new Array();
filterArray.push(filter);
art.filters = filterArray;

function createRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  mc.beginFill(bgColor);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc._x = 20;
  mc._y = 20;
  return mc;
}
```

hideObject (DropShadowFilter.hideObject プロパティ)

```
public hideObject : Boolean
```

オブジェクトが非表示であるかどうかを示します。true を指定すると、オブジェクト自体は描画されず、シャドウだけが表示されます。デフォルトは false で、オブジェクトが表示されます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その hideObject プロパティを変更します。

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowHideObject");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.hideObject = true;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, 0.8, 16,
  16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

inner (DropShadowFilter.inner プロパティ)

```
public inner : Boolean
```

影が内側の影であるかどうかを指定します。true の場合は、内側シャドウであることを示します。デフォルトは false (外側シャドウ) で、オブジェクトの外周にあるシャドウを示します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その inner プロパティを変更します。

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowInner");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.inner = true;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, 0.8, 16,
  16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

knockout (DropShadowFilter.knockout プロパティ)

```
public knockout : Boolean
```

true の場合は、ノックアウト効果を適用します。その結果、オブジェクトは完全に透明になり、ドキュメントの背景色で表示されます。デフォルトは false (ノックアウトなし) です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その knockout プロパティを変更します。

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowKnockout");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.knockout = true;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, 0.8, 16,
  16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

quality (DropShadowFilter.quality プロパティ)

```
public quality : Number
```

フィルタを適用する回数。有効な値は 0 ～ 15 です。デフォルト値は 1 で、低品質と等価です。値 2 は標準の品質であり、値 3 は高品質です。フィルタに設定された値が小さいほど、速くレンダリングできます。

多くのアプリケーションでは、quality 値 1、2、または 3 で十分です。最大 15 までの値を使用してさまざまな効果を出すことができますが、値が大きくなるほどレンダリング速度が低下します。多くの場合、quality の値を大きくする代わりに blurX と blurY の値を大きくするだけで、同様の効果が得られます。この方法を実行すると、より高速にレンダリングされます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その quality プロパティを変更します。

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowQuality");
mc.onRelease = function() {
   var filter:DropShadowFilter = this.filters[0];
   filter.quality = 0;
   this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
   var art:MovieClip = this.createEmptyMovieClip(name,
   this.getNextHighestDepth());
   var w:Number = 100;
   var h:Number = 100;
   art.beginFill(0x003366);
   art.lineTo(w, 0);
   art.lineTo(w, h);
   art.lineTo(0, h);
   art.lineTo(0, 0);
   art._x = 20;
   art._y = 20;

   var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, 0.8, 16,
   16, 1, 3, false, false, false);
   var filterArray:Array = new Array();
   filterArray.push(filter);
   art.filters = filterArray;
   return art;
}
```

strength (DropShadowFilter.strength プロパティ)

```
public strength : Number
```

インプリントの強さまたは広がり。値が大きいほど、濃い色がインプリントされるので、シャドウと背景との間のコントラストが強くなります。指定できる値は 0 ～ 255 で、デフォルト値は 1 です。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その strength プロパティを変更します。

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowStrength");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.strength = 0.6;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, 0.8, 16,
  16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

Error

```
Object
  |
  +-Error
```

```
public class Error
extends Object
```

スクリプト内で発生したエラーについての情報を含みます。Error オブジェクトを作成するには、Error コンストラクタ関数を使用します。一般には、try コードブロック内から新しい Error オブジェクトの throw を実行します。そして、catch コードブロックまたは finally コードブロックでこれをキャッチします。

Error クラスのサブクラスを作成し、そのサブクラスのインスタンスをスローすることもできます。

対応バージョン： ActionScript 1.0、Flash Player 7

プロパティー覧

オプション	プロパティ	説明
	message:String	Error オブジェクトに関連付けられたメッセージです。
	name:String	Error オブジェクトの名前です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター覧

署名	説明
Error([message: String])	新しい Error オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	toString() : String	デフォルトでは "Error" というストリングを返します。Error.message が定義されている場合は、その値を返します。

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

Error コンストラクタ

```
public Error([message:String])
```

新しい Error オブジェクトを作成します。*message* を指定した場合は、その値がオブジェクトの Error.message プロパティに割り当てられます。

対応バージョン： ActionScript 1.0、Flash Player 7

パラメータ

message:String (オプション) - Error オブジェクトに関連付けられたストリング。

例

次の例の関数は、受け取った 2 つのストリングが同じでない場合に、指定のメッセージを使用してエラーをスローします。

```
function compareStrings(str1_str:String, str2_str:String):Void {
  if (str1_str != str2_str) {
  throw new Error("Strings do not match.");
  }
}
try {
  compareStrings("Dog", "dog");
  // output: Strings do not match.
} catch (e_err:Error) {
  trace(e_err.toString());
}
```

関連項目

throw ステートメント, try..catch..finally ステートメント

message (Error.message プロパティ)

```
public message : String
```

Error オブジェクトに関連付けられたメッセージです。デフォルトでは、このプロパティの値は "Error" です。Error オブジェクトを作成する際に、Error コンストラクタ関数にエラーストリングを渡すことで message プロパティを指定できます。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例の関数は、theNum に入力されたパラメータに応じて、指定されたメッセージをスローします。2 つの数値を使って除算できる場合は、SUCCESS というストリングとその結果が表示されます。0 による除算を行おうとした場合、または 1 つしかパラメータが入力されていない場合は、それに対応するエラーが表示されます。

```
function divideNum(num1:Number, num2:Number):Number {
   if (isNaN(num1) || isNaN(num2)) {
   throw new Error("divideNum function requires two numeric parameters.");
   } else if (num2 == 0) {
   throw new Error("cannot divide by zero.");
   }
   return num1/num2;
}
try {
   var theNum:Number = divideNum(1, 0);
   trace("SUCCESS! "+theNum);
} catch (e_err:Error) {
   trace("ERROR! "+e_err.message);
   trace("\t"+e_err.name);
}
```

この ActionScript の除算する数値を変えずにテストすると、0 による除算を行うことになり、[出力] パネルにエラーが表示されます。

関連項目

throw ステートメント , try..catch..finally ステートメント

name (Error.name プロパティ)

```
public name : String
```

Error オブジェクトの名前です。デフォルトでは、このプロパティの値は "Error" です。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例の関数は、除算する 2 つの数値に応じて、指定されたエラーをスローします。タイムラインのフレーム 1 に次の ActionScript を追加します。

```
function divideNumber(numerator:Number, denominator:Number):Number {
   if (isNaN(numerator) || isNaN(denominator)) {
   throw new Error("divideNum function requires two numeric parameters.");
   } else if (denominator == 0) {
   throw new DivideByZeroError();
   }
   return numerator/denominator;
}
try {
   var theNum:Number = divideNumber(1, 0);
   trace("SUCCESS! "+theNum);
   // output: DivideByZeroError -> Unable to divide by zero.
} catch (e_err:DivideByZeroError) {
   // divide by zero error occurred
   trace(e_err.name+" -> "+e_err.toString());
} catch (e_err:Error) {
   // generic error occurred
   trace(e_err.name+" -> "+e_err.toString());
}
```

カスタムエラーを追加するには、次のコードを DivideByZeroError.as ファイルに追加し、FLA ドキュメントと同じディレクトリにクラスファイルを保存します。

```
class DivideByZeroError extends Error {
   var name:String = "DivideByZeroError";
   var message:String = "Unable to divide by zero.";
}
```

関連項目

throw ステートメント , try..catch..finally ステートメント

toString (Error.toString メソッド)

```
public toString() : String
```

デフォルトでは "Error" というストリングを返します。Error.message が定義されている場合は、その値を返します。

対応バージョン : ActionScript 1.0、Flash Player 7

戻り値

`String` - ストリング。

例

次の例の関数は、受け取った 2 つのストリングが同じでない場合に、指定のメッセージを使用してエラーをスローします。

```
function compareStrings(str1_str:String, str2_str:String):Void {
if (str1_str != str2_str) {
  throw new Error("Strings do not match.");
  }
}
try {
  compareStrings("Dog", "dog");
  // output: Strings do not match.
} catch (e_err:Error) {
  trace(e_err.toString());
}
```

関連項目

message (Error.message プロパティ),throw ステートメント ,try..catch..finally ステートメント

ExternalInterface (flash.external.ExternalInterface)

```
Object
  |
  +-flash.external.ExternalInterface
```

```
public class ExternalInterface
extends Object
```

ExternalInterface クラスは外部 API であり、ActionScript と Flash Player のコンテナとの間で直接通信できるようにするアプリケーションプログラミングインターフェイスです。コンテナの例としては、JavaScript が含まれる HTML ページや、Flash Player が埋め込まれたデスクトップアプリケーションなどがあります。

ExternalInterface の機能は `fscommand()`、`CallFrame()`、および `CallLabel()` メソッドと似ていますが、柔軟性に優れ、一般的に、より広い範囲に適用できます。JavaScript と ActionScript の通信には ExternalInterface を使用することをお勧めします。

ActionScript から、HTML ページ上の JavaScript 関数を呼び出し、任意のデータ型の引数の数を渡して、呼び出しから戻り値を受け取ることができます。

HTML ページの JavaScript から、Flash Player の ActionScript 関数を呼び出すことができます。ActionScript 関数は値を返すことができ、JavaScript は、呼び出しの戻り値として即座にこの値を受け取ります。

ExternalInterface は、次のブラウザとオペレーティングシステムの組み合わせに対応しています。

ブラウザ	オペレーティングシステム	
Internet Explorer 5.0 以降	Windows	
Netscape 8.0 以降	Windows	Macintosh
Mozilla 1.7.5 以降	Windows	Macintosh
Firefox 1.0 以降	Windows	Macintosh
Safari 1.3 以降		Macintosh

ExternalInterface を利用するには、ユーザーの Web ブラウザが、一部のブラウザによってプラグインスクリプトとして公開されている ActiveX または NPRuntime API のいずれかをサポートしている必要があります。http://www.mozilla-japan.org/projects/plugins/npruntime.html を参照してください。

対応バージョン： ActionScript 1.0、Flash Player 8

プロパティ一覧

オプション	プロパティ	説明
static	available:Boolean (読み取り専用)	この Player が外部インターフェイスを備えたコンテナに含まれているかどうかを示します。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

メソッド一覧

オプション	署名	説明
static	addCallback (methodName:String, instance:Object, method:Function) : Boolean	ActionScript メソッドをコンテナから呼び出し可能なものとして登録します。
static	call(methodName: String, [parameter1: Object]) : Object	Flash Player コンテナで公開されている関数を呼び出し、必要に応じて引数を渡します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

addCallback (ExternalInterface.addCallback メソッド)

```
public static addCallback(methodName:String, instance:Object, method:Function) :
  Boolean
```

ActionScript メソッドをコンテナから呼び出し可能なものとして登録します。addCallBack( ) の呼び出しが成功すると、Flash Player 内に登録されている関数をコンテナ内の JavaScript や ActiveX から呼び出すことができます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

methodName:String - JavaScript から呼び出すことができる ActionScript 関数の名前です。この名前と実際の ActionScript メソッドの名前が一致する必要はありません。

instance:Object - メソッドで this が解決するオブジェクト。このオブジェクトは、メソッドの検索元となるオブジェクトである必要はありません。ここには、任意のオブジェクト (null も可能) を指定できます。

method:Function - JavaScript から呼び出される ActionScript メソッド。

戻り値

Boolean - 呼び出しが成功した場合は、true を返します。失敗した場合には false を返します。失敗原因としては、インスタンスが利用できない場合、セキュリティ上の制限に抵触した場合、該当する関数オブジェクトが存在しない場合、再帰呼び出しが発生した場合などがあります。

戻り値 false は、コンテナ環境がセキュリティサンドボックスに属しているために、呼び出し側のコードにアクセス権がないことを示す場合もあります。次のようにすれば、この問題を回避できます。

- 含まれている HTML ページ内の SWF ファイルの <object> タグで、次のパラメータを設定します。

  ```
  <param name = "allowScriptAccess" value = "always" />
  ```

- SWF ファイルで、次の ActionScript を追加します。

  ```
  System.security.allowDomain( sourceDomain )
  ```

例

次の例では、コンテナから呼び出し可能な関数として goToAdobe( ) 関数を goHome と登録します。

```
import flash.external.*;

var methodName:String = "goHome";
var instance:Object = null;
var method:Function = goToAdobe;
var wasSuccessful:Boolean = ExternalInterface.addCallback(methodName, instance,
  method);
```

```
var txtField:TextField = this.createTextField("txtField",
  this.getNextHighestDepth(), 0, 0, 200, 50);
txtField.border = true;
txtField.text = wasSuccessful.toString();

function goToAdobe() {
  txtField.text = "http://www.adobe.com";
  getURL("http://www.adobe.com", "_self");
}
```

前述の例が正常に動作するには、次のコードをコンテナ HTML ページにコピーしてペーストする必要があります。このコードは、値 externalInterfaceExample を持つために OBJECT タグの id 属性と EMBED タグの name 属性に依存します。Internet Explorer と Netscape ではムービーオブジェクトの参照方法が異なるため、関数 thisMovie はブラウザに応じて適切なシンタックスを返します。HTML ページがホストサーバー上に置かれていない場合、ブラウザはセキュリティ上の警告を出す可能性があります。

メモ: プラグインオブジェクトにアクセスする他のメソッド (たとえば document.getElementById("pluginName") や document.all.pluginName) は使用しないでください。これは、一部のブラウザでは、このようなメソッドの動作が異なるためです。

```
<form>
  <input type="button" onclick="callExternalInterface()" value="Call
  ExternalInterface" />
</form>
<script>
function callExternalInterface() {
  thisMovie("externalInterfaceExample").goHome();
}

function thisMovie(movieName) {
  if (navigator.appName.indexOf("Microsoft") != -1) {
    return window[movieName]
  }
  else {
    return document[movieName]
  }
}
</script>
```

関連項目

allowDomain (security.allowDomain メソッド)

available (ExternalInterface.available プロパティ)

```
public static available : Boolean ( 読み取り専用 )
```

この Player が外部インターフェイスを備えたコンテナに含まれているかどうかを示します。外部インターフェイスが利用できる場合、このプロパティは true になります。利用できない場合には false になります。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、ExternalInterface.available を使用して、外部インターフェイスを備えたコンテナ内に Player が含まれているかどうかを決定します。

```
import flash.external.*;

var isAvailable:Boolean = ExternalInterface.available;
trace(isAvailable);
```

call (ExternalInterface.call メソッド)

```
public static call(methodName:String, [parameter1:Object]) : Object
```

Flash Player コンテナで公開されている関数を呼び出し、必要に応じて引数を渡します。指定された関数が利用できない場合は null を返します。それ以外の場合は、関数の戻り値を返します。再帰呼び出しはできません。再帰呼び出しの場合は null を返します。

コンテナが HTML ページである場合、このメソッドは <script> エレメントに囲まれた JavaScript 関数を呼び出します。

コンテナが何か他の ActiveX コンテナの場合、このメソッドは、指定された名前でイベントをブロードキャストし、そのイベントはコンテナによって処理されます。

コンテナが Netscape プラグインをホストしている場合、新しい NPRuntime インターフェイス用のカスタムサポートを記述するか、HTML コントロールを埋め込んだ後にそのコントロール内に Flash Player を埋め込むことができます。HTML コントロールを埋め込んだ場合、ネイティブコンテナアプリケーションとやり取りする JavaScript インターフェイスを通じて Flash Playerと通信できます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

methodName:String - コンテナ内にある呼び出し先関数の名前です。関数がパラメータを受け取る場合は、methodName パラメータの後に記述する必要があります。

parameter1:Object (オプション) - 関数に渡す任意のパラメータ。任意のパラメータを指定することができ、複数のパラメータを指定する場合はカンマで区切ります。パラメータには任意の ActionScript データ型を使用できます。呼び出し先が JavaScript 関数である場合、ActionScript のデータ型は JavaScript のデータ型に自動的にマーシャリングされます。呼び出し先が何か他の ActiveX コンテナである場合、パラメータは要求メッセージの中にエンコードされます。

戻り値

Object - コンテナから受け取った応答です。呼び出しに失敗した場合は、null を返します。失敗原因としては、コンテナに該当する関数が存在しない場合、インターフェイスが利用できない場合、再帰が発生した場合、セキュリティ上の問題がある場合などがあります。

例

次の例では、SWF を含む HTML ページ内の JavaScript 関数 sayHello() を呼び出します。この呼び出しは、ExternalInterface.call() メソッドを使って実行されます。

```
import flash.external.*;

var greeting:String;
var btn:MovieClip = createButton(100, 30, 0xCCCCCC);
btn.onPress = function() {
  greeting = String(ExternalInterface.call("sayHello", "browser"));
  this.mcTxt.text = greeting; // >> Hi Flash.
}

function createButton(width:Number, height:Number, color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  var mcFmt:TextFormat;

  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);

  mcFmt = new TextFormat();
  mcFmt.align = "center";
  mcFmt.bold = true;

  mc.createTextField("mcTxt", depth, 0, 0, width, height);
  mc.mcTxt.text = "Call JS Function";
  mc.mcTxt.setTextFormat(mcFmt);

  return mc;
}
```

前述の例が正常に動作するには、次のコードをコンテナ HTML ページにコピーしてペーストする必要があります。HTML ページがホストサーバー上に置かれていない場合、ブラウザはセキュリティ上の警告を出す可能性があります。

```
<script>
  function sayHello(name) {
    alert(">> Hello " + name + ".");
    return ">> Hi Flash.";
  }
</script>
```

FileReference (flash.net.FileReference)

```
Object
  |
  +-flash.net.FileReference
```

```
public class FileReference
extends Object
```

FileReference クラスには、ユーザーのコンピュータとサーバーとの間でファイルをアップロードおよびダウンロードするための手段があります。オペレーティングシステムのダイアログボックスを使用して、アップロードするファイルや、ダウンロード先の場所をユーザーが選択できるようにします。各 FileReference オブジェクトはユーザーのハードディスク上にある 1 つのファイルを参照し、ファイルのサイズ、タイプ、名前、作成日、変更日、クリエータタイプ (Macintosh のみ) に関する情報を保持するプロパティを備えています。

FileReference インスタンスは、次の 2 とおりの方法で作成できます。

- FileReference コンストラクタで new 演算子を使用する方法 : `var myFileReference = new FileReference();`。
- `FileReferenceList.browse()` を呼び出す方法。これによって FileReference オブジェクトの配列が作成されます。

アップロード処理の実行中、FileReference オブジェクトのすべてのプロパティの値が、`FileReference.browse()` または `FileReferenceList.browse()` への呼び出しによって設定されます。ダウンロード処理の実行中、name プロパティの値は `onSelect` が呼び出されたときに設定されます。これ以外のすべてのプロパティの値は、`onComplete` が呼び出されたときに設定されます。

`browse()` メソッドはオペレーティングシステムのダイアログボックスを開いて、ユーザーがアップロード対象のローカルファイルを選択できるようにします。`FileReference.browse()` メソッドを使用すると、ユーザーは単一のファイルを選択できます。`FileReferenceList.browse()` メソッドを使用すると、複数のファイルを選択できます。`browse()` メソッドの呼び出しが正常に終了したら、`FileReference.upload()` メソッドを呼び出して一度に 1 つのファイルをアップロードします。`FileReference.download()` メソッドは、ファイルの保存先をユーザーに指定させ、リモート URL からのダウンロードを開始します。

browse() および download() 呼び出しによって生成されるダイアログボックスのデフォルトのファイルの場所を、FileReference クラスおよび FileReferenceList クラスを使って設定することはできません。ダイアログボックスに表示されるデフォルトの場所は、最後に参照されたフォルダ (その場所を決定できる場合) またはデスクトップです。これらのクラスを使って、転送ファイルを読み込んだり、転送ファイルに書き込んだりすることはできません。これらのクラスを使用しても、アップロードまたはダウンロードを開始した SWF ファイルが、アップロードファイルやダウンロードファイル、またはユーザーのディスクのファイルの場所にアクセスすることはできません。

また FileReference と FileReferenceList クラスは認証方法も提供しません。認証が必要なサーバーでは、Flash Player ブラウザプラグインを使ってファイルをダウンロードできますが、すべての Player でのアップロード、およびスタンドアローンまたは外部 Player でのダウンロードは失敗します。処理が正常に終了したかどうかを確認して、エラー処理を行う場合は、FileReference のイベントリスナーを使用します。

アップロード処理およびダウンロード処理の場合、SWF ファイルは、それ自体のドメイン内に含まれるファイルだけにアクセスできます。こうしたドメインには、クロスドメインポリシーファイルで指定されたドメインが含まれます。アップロードまたはダウンロードを開始している SWF が、ファイルサーバーと同じドメインに属していない場合、ファイルサーバーにポリシーファイルを配置する必要があります。

FileReference.browse()、FileReferenceList.browse()、または FileReference.download() メソッドの呼び出し中、SWF ファイルの再生が一時停止する場合があります。一時停止が発生するプラットフォームは、Linux 対応 Flash Player、Mac OS X の Flash Player プラグイン、Macintosh の外部 Flash Player、および Mac OS X 10.1 以前のスタンドアローン Flash Player です。すべての Windows 用 Flash Palyer と Mac OS X 10.2 以降で実行される Macintosh 用スタンドアローン Flash Player で実行する場合、SWF ファイルの再生は続行します。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、ユーザーがアップロード対象のイメージまたはテキストファイルを選択できるようにする FileReference オブジェクトを作成します。また、使用可能なイベントの待機も実行します。

```
import flash.net.FileReference;

var allTypes:Array = new Array();
var imageTypes:Object = new Object();
imageTypes.description = "Images (*.jpg, *.jpeg, *.gif, *.png)";
imageTypes.extension = "*.jpg; *.jpeg; *.gif; *.png";
allTypes.push(imageTypes);

var textTypes:Object = new Object();
textTypes.description = "Text Files (*.txt, *.rtf)";
textTypes.extension = "*.txt;*.rtf";
allTypes.push(textTypes);
```

```
var listener:Object = new Object();

listener.onSelect = function(file:FileReference):Void {
  trace("onSelect: " + file.name);
  if(!file.upload("http://www.yourdomain.com/yourUploadHandlerScript.cfm")) {
    trace("Upload dialog failed to open.");
  }
}

listener.onCancel = function(file:FileReference):Void {
  trace("onCancel");
}

listener.onOpen = function(file:FileReference):Void {
  trace("onOpen: " + file.name);
}

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

listener.onHTTPError = function(file:FileReference):Void {
  trace("onHTTPError: " + file.name);
}

listener.onIOError = function(file:FileReference):Void {
  trace("onIOError: " + file.name);
}

listener.onSecurityError = function(file:FileReference, errorString:String):Void
  {
  trace("onSecurityError: " + file.name + " errorString: " + errorString);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse(allTypes);
```

関連項目

FileReferenceList (flash.net.FileReferenceList)

プロパティー覧

オプション	プロパティ	説明
	creationDate:Date (読み取り専用)	ローカルディスク上に存在するファイルの作成日です。
	creator:String (読み取り専用)	ファイルの Macintosh クリエータタイプです。
	modificationDate: Date (読み取り専用)	ローカルディスク上に存在するファイルの最終変更日です。
	name:String (読み取り専用)	ローカルディスク上に存在するファイルの名前です。
	postData:String	アップロードまたはダウンロード時に送信する POST パラメータ
	size:Number (読み取り専用)	ローカルディスク上に存在するファイルのサイズ (バイト単位) です。
	type:String (読み取り専用)	ファイル形式です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベントの一覧

イベント	説明
onCancel = function(fileRef: FileReference) {}	ユーザーがファイル参照ダイアログボックスを閉じると、呼び出されます。
onComplete = function(fileRef: FileReference) {}	アップロード処理またはダウンロード処理が正常に完了したときに呼び出されます。
onHTTPError = function(fileRef: FileReference, httpError:Number) {}	HTTP エラーのためにアップロードが失敗したときに呼び出されます。
onIOError = function(fileRef: FileReference) {}	入出力エラーが発生したときに呼び出されます。
onOpen = function(fileRef: FileReference) {}	アップロード処理またはダウンロード処理が開始するときに呼び出されます。

イベント	説明
`onProgress = function(fileRef: FileReference, bytesLoaded:Number, bytesTotal:Number) {}`	ファイルのアップロード処理中またはダウンロード処理中に定期的に呼び出されます。
`onSecurityError = function(fileRef: FileReference, errorString:String) {}`	セキュリティエラーのためにアップロードまたはダウンロードが失敗したときに呼び出されます。
`onSelect = function(fileRef: FileReference) {}`	ユーザーがアップロードするファイルまたはダウンロードするファイルをファイル参照ダイアログボックスから選択したときに、呼び出されます。
`onUploadCompleteData = function(fileRef: FileReference, data:String) {}`	アップロードが正常に終了した後にサーバーからデータを受信すると呼び出されます。

コンストラクター一覧

署名	説明
`FileReference()`	新しい FileReference オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	`addListener(listener: Object) : Void`	FileReference イベントリスナーが呼び出されたときに通知を受けるオブジェクトを登録します。
	`browse([typelist: Array]) : Boolean`	アップロードするローカルファイルを選択するためのファイル参照ダイアログボックスを表示します。
	`cancel() : Void`	この FileReference オブジェクトで進行中のアップロード処理またはダウンロード処理を取り消します。
	`download(url:String, [defaultFileName: String]) : Boolean`	リモートサーバーからファイルをダウンロードするためのダイアログボックスを表示します。

オプション	署名	説明
	removeListener (listener:Object) : Boolean	イベント通知メッセージを受信するオブジェクトのリストからオブジェクトを削除します。
	upload(url:String, uploadDataFieldName: String, testUpload:Boolean) : Boolean	ユーザーが選択したファイルをリモートサーバーにアップロードする処理を開始します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

addListener (FileReference.addListener メソッド)

```
public addListener(listener:Object) : Void
```

FileReference イベントリスナーが呼び出されたときに通知を受けるオブジェクトを登録します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`listener:Object` - FileReference イベントリスナーからのコールバック通知を待機するオブジェクト。

例

次の例は、リスナーを FileReference のインスタンスに追加します。

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
}
```

```
listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.adobe.com/platform/whitepapers/
  platform_overview.pdf";
fileRef.download(url, "FlashPlatform.pdf");
```

browse (FileReference.browse メソッド)

```
public browse([typelist:Array]) : Boolean
```

アップロードするローカルファイルを選択するためのファイル参照ダイアログボックスを表示します。このダイアログボックスは、オペレーティングシステムのネイティブのダイアログボックスです。このメソッドを呼び出して、ユーザーが正常にファイルを選択すると、この FileReference オブジェクトのプロパティにそのファイルのプロパティが設定されます。これ以降 `FileReference.browse()` が呼び出されるたびに、FileReference オブジェクトのプロパティは、ダイアログボックスでユーザーが選択したファイルにリセットされます。

一度に 1 つの `browse()` セッションまたは `download()` セッションだけを実行できます。これは、一度に 1 つのダイアログボックスしか表示できないためです。

どのファイルをダイアログボックスに表示するかを決定するために、ファイルタイプの配列を渡すことができます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`typelist:Array` (オプション) - ダイアログボックスに表示するファイルをフィルタにかける場合に使用するファイルタイプの配列。このパラメータを省略すると、すべてのファイルが表示されます。このパラメータを設定する場合は、エレメントを中カッコ ({ }) で囲んで配列に入れる必要があります。配列の形式として、次の 2 つのいずれかを使用できます。

- Windows ファイル拡張子だけが含まれるファイル形式の記述リスト。配列の各エレメントには、ファイル形式を説明するストリングと、Windows ファイル拡張子をセミコロンで区切ったリストを入れる必要があります。各拡張子の前にはワイルドカード文字 (*) を付けます。各エレメントのシンタックスは次のとおりです。`[{description: "`*`string describing the first set of file types`*`", extension: "`*`semicolon-delimited list of file extensions`*`"}]` 例 : `[{description: "Images", extension: "*.jpg;*.gif;*.png"}, {description: "SWF files", extension: "*.swf"}, {description: "Documents", extension: "*.doc;*.pdf"}]`

- Windows ファイル拡張子と Macintosh ファイル形式が含まれるファイル形式記述リスト。配列の各エレメントには、ファイル形式を説明するストリングを含む必要があります。つまり、各拡張子の前にワイルドカード文字 (*) を付けた状態で Windows ファイル拡張子をセミコロンで区切ったリスト、および各ファイル形式の前にワイルドカード文字 (*) を付けた状態で Macintosh ファイル形式をセミコロンで区切ったリストを含む必要があります。各エレメントのシンタックスは次のとおりです。`[{description: "string describing the first set of file types", extension: "semicolon-delimited list of Windows file extensions", macType: "semicolon-delimited list of Macintosh file types"}]` 例:
  `[{description: "Image files", extension: "*.jpg;*.gif;*.png", macType: "JPEG;jp2_;GIFF"}, {description: "SWF files", extension: "*.swf", macType: "SWFL"}]`

この 2 つの形式は、1 つの `browse()` 呼び出しの中で混在できません。どちらか一方を使用する必要があります。

拡張子のリストは、ユーザーが選択したファイルに応じて、Windows のファイルをフィルタにかけるために使用されます。ダイアログボックスに実際に表示されるわけではありません。ファイル形式をユーザーに表示するには、拡張子リストのほか、説明用ストリングにもファイル形式をリストする必要があります。説明用ストリングは、Windows のダイアログボックスに表示されます。Macintosh では使用されません。Macintosh では、Macintosh のファイル形式リストを指定すると、このリストがファイルのフィルタリングに使用されます。Macintosh のファイルタイプのリストを指定しない場合、Windows 拡張子のリストが使用されます。

戻り値

`Boolean` - パラメータが有効で、ファイル参照ダイアログボックスが表示された場合に、`true` を返します。ダイアログボックスが表示されない場合、既に他のブラウザセッションが進行中である場合、または `typelist` パラメータが使用されたが、配列内のエレメントに説明用文字列または拡張子ストリングが指定されていない場合は、`false` を返します。

例

次の例は、ユーザーがアップロード対象のイメージファイルを選択できるダイアログボックスを表示します。

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("Opened " + file.name);
}

listener.onCancel = function(file:FileReference):Void {
  trace("User cancelled");
}
```

```
var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

関連項目

onSelect (FileReferenceList.onSelect イベントリスナー), onCancel (FileReference.onCancel イベントリスナー), download (FileReference.download メソッド), browse (FileReferenceList.browse メソッド)

cancel (FileReference.cancel メソッド)

```
public cancel() : Void
```

この FileReference オブジェクトで進行中のアップロード処理またはダウンロード処理を取り消します。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、要求されたファイルのおよそ半分をダウンロードした後、そのダウンロードをキャンセルします。これは、明らかに通常の操作ではありません。このメソッドを使用するケースとしては、ユーザーがダウンロードステータスダイアログボックスで [キャンセル] をクリックできるようにする場合が想定されます。

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
  if(bytesLoaded >= (bytesTotal / 2)) {
    file.cancel();
  }
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.adobe.com/platform/whitepapers/
  platform_overview.pdf";
fileRef.download(url, "FlashPlatform.pdf");
```

creationDate (FileReference.creationDate プロパティ)

public creationDate : Date (読み取り専用)

ローカルディスク上に存在するファイルの作成日です。FileReference オブジェクトに値が設定されていない場合に、このプロパティの値を取得する呼び出しが行われると null を返します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例は、ユーザーによって選択されたファイルの作成日を取得します。

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("creationDate: " + file.creationDate);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

関連項目

browse (FileReference.browse メソッド)

creator (FileReference.creator プロパティ)

public creator : String (読み取り専用)

ファイルの Macintosh クリエータタイプです。Windows の場合、このプロパティは null になります。FileReference オブジェクトに値が設定されていない場合に、このプロパティの値を取得する呼び出しが行われると null を返します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例は、ユーザーによって選択されたファイルの Macintosh クリエータタイプを取得します。

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("creator: " + file.creator);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

関連項目

browse (FileReference.browse メソッド)

download (FileReference.download メソッド)

```
public download(url:String, [defaultFileName:String]) : Boolean
```

リモートサーバーからファイルをダウンロードするためのダイアログボックスを表示します。Flash Player ではアップロードまたはダウンロードできるファイルのサイズに制限はありませんが、このプレーヤーが正式にサポートしているアップロードまたはダウンロードのサイズは最大 100 MB です。

このメソッドは、まず、オペレーティングシステムのダイアログボックスを表示して、ユーザーにファイル名を入力してもらった後、ファイル名の保存先となるローカルコンピュータ上の場所を選択してもらいます。ユーザーが保存場所を選択し、[保存] などをクリックして、ファイルをローカルに保存することを確認すると、リモートサーバーからのダウンロードが開始します。リスナーは、ダウンロードが進行中なのか、成功したのか、失敗したのかを示すイベントを受け取ります。download() を呼び出した後のダイアログボックスやダウンロード処理の状態を確認するには、ActionScript で onCancel、onOpen、onProgress、および onComplete などのイベントリスナーを使用してイベントを待機する必要があります。

FileReference.upload() 関数と FileReference.download() 関数はノンブロッキング処理を行います。これらの関数は呼び出された後、ファイル転送が完了する前に返されます。さらに、FileReference オブジェクトがスコープ外に移動した場合、そのオブジェクトに対して完了していないアップロードまたはダウンロードは、スコープから離れた時点でキャンセルされます。アップロードまたはダウンロードの続行を期待できる間は、FileReference オブジェクトがスコープ内に残ることを確認してください。

ファイルが正常にダウンロードされると、FileReference オブジェクトのプロパティにローカルファイルのプロパティが設定され、onComplete リスナーが呼び出されます。

一度に 1 つの browse() セッションまたは download() セッションだけを実行できます。これは、一度に 1 つのダイアログボックスしか表示できないためです。

このメソッドは、どのファイルタイプのダウンロードにも対応しており、HTTP と HTTPS のいずれも使用できます。

POST パラメータをサーバーに送信するには、FileReference.postData の値に任意のパラメータを設定します。また、解析するサーバースクリプトに関して、URL にパラメータを付加することにより、download() 呼び出しで GET パラメータをサーバーに送ることもできます。

メモ: サーバーでユーザー認証が必要な場合、ブラウザ内で実行される SWF ファイル、つまり、ブラウザプラグインまたは ActiveX コントロールを使用する SWF ファイルでのみ、認証用のユーザー名とパスワードをユーザーが入力できるダイアログボックスを表示できます。ただし、それはダウンロードの場合のみです。プラグインまたは ActiveX コントロールを使用するアップロードの場合、および、スタンドアローンまたは外部 Player を使用するアップロードとダウンロードの場合、ファイル転送は失敗します。

このメソッドを使用するときは、Flash Player セキュリティモデルを考慮してください。

- 呼び出し元 SWF ファイルが信頼されないコードとしてローカルのサンドボックスに置かれている場合、このメソッドは使用できません。
- デフォルトのセキュリティ設定では、サンドボックス間のアクセスは拒否されます。クロスドメインポリシーファイルを追加することによって、Web サイトでリソースにアクセスできるようになります。

詳細については、次の参照先を参照してください。

- 『ActionScript 2.0 の学習』の「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

対応バージョン: ActionScript 1.0、Flash Player 8

パラメータ

`url:String` - ローカルコンピュータにダウンロードするファイルの URL。解析するサーバースクリプトに関して、URL にパラメータを付加することにより、`download()` 呼び出で GET パラメータをサーバーに送ることができます。(たとえば `http://www.myserver.com/picture.jpg?userID=jdoe`)

いくつかのブラウザでは、URL ストリングの長さに制限があります。長さが 256 文字を超える場合、一部のブラウザまたはサーバーでは失敗する場合があります。

`defaultFileName:String`(オプション) - ダウンロードするファイルとしてダイアログボックスに表示するデフォルトファイル名です。このストリングに、/\:*?という文字を含めることはできません。"<>|%。

このパラメータを省略すると、リモート URL のファイル名が構文解析されて、デフォルトとして使用されます。

戻り値

Boolean - ユーザーがファイルを選択できるダイアログボックスが表示される場合は true。ダイアログボックスが表示されない場合、このメソッドは false を返します。ダイアログボックスを表示できない原因は次のとおりです。

- url パラメータの値を渡さなかった場合。
- 渡されたパラメータの形式が正しくなかった場合。
- url パラメータの長さが 0 の場合。
- セキュリティ侵害が発生した場合。つまり、SWF ファイルが、そのセキュリティサンドボックス外にあるサーバーのファイルにアクセスしようとした場合。
- 既に他のブラウズセッションが進行中であった場合。ブラウズセッションは、FileReference.browse()、FileReferenceList.browse()、または FileReference.download() によって開始できます。
- プロトコルが HTTP または HTTPS でなかった場合。

例

次の例では、download メソッドを使ってファイルのダウンロードを実行します。すべてのイベントについてリスナーが存在します。

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onSelect = function(file:FileReference):Void {
  trace("onSelect: " + file.name);
}

listener.onCancel = function(file:FileReference):Void {
  trace("onCancel");
}

listener.onOpen = function(file:FileReference):Void {
  trace("onOpen: " + file.name);
}

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}
```

```
listener.onIOError = function(file:FileReference):Void {
  trace("onIOError: " + file.name);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.adobe.com/platform/whitepapers/
  platform_overview.pdf";
if(!fileRef.download(url, "FlashPlatform.pdf")) {
  trace("dialog box failed to open.");
}
```

関連項目

browse (FileReference.browse メソッド), browse (FileReferenceList.browse メソッド), upload (FileReference.upload メソッド)

FileReference コンストラクタ

```
public FileReference()
```

新しい FileReference オブジェクトを作成します。設定されると、FileReference オブジェクトはユーザーのローカルディスク上のファイルを表します。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、新しい FileReference オブジェクトを作成して、PDF ファイルのダウンロードを開始します。

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onComplete = function(file:FileReference) {
  trace("onComplete : " + file.name);
}

var url:String = "http://www.adobe.com/platform/whitepapers/
  platform_overview.pdf";
var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.download(url, "FlashPlatform.pdf");
```

関連項目

browse (FileReference.browse メソッド)

modificationDate (FileReference.modificationDate プロパティ)

public modificationDate : Date (読み取り専用)

ローカルディスク上に存在するファイルの最終変更日です。FileReference オブジェクトに値が設定されていない場合に、このプロパティの値を取得する呼び出しが行われると null を返します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例は、ユーザーによって選択されたファイルの modificationDate を取得します。

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("modificationDate: " + file.modificationDate);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

関連項目

browse (FileReference.browse メソッド)

name (FileReference.name プロパティ)

public name : String (読み取り専用)

ローカルディスク上に存在するファイルの名前です。FileReference オブジェクトに値が設定されていない場合に、このプロパティの値を取得する呼び出しが行われると null を返します。

FileReference オブジェクトのすべてのプロパティの値は、browse() を呼び出すことにより設定されます。download() を呼び出した場合、他の FileReference のプロパティと異なり、name プロパティの値は onSelect が呼び出されたときに設定されます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例は、ユーザーによって選択されたファイルの名前を取得します。

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("name: " + file.name);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

関連項目

browse (FileReference.browse メソッド)

onCancel (FileReference.onCancel イベントリスナー)

```
onCancel = function(fileRef:FileReference) {}
```

ユーザーがファイル参照ダイアログボックスを閉じると、呼び出されます。このダイアログボックスは、FileReference.browse() メソッド、FileReferenceList.browse() メソッド、または FileReference.download() メソッドを呼び出したときに表示されます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`fileRef:FileReference` - 処理を開始した FileReference オブジェクト。

例

次の例では、ユーザーがファイル参照ダイアログボックスを閉じた場合のメッセージをトレースします。このメソッドは、ダイアログボックスが表示された後に、ユーザーが [キャンセル] をクリックするか、Esc キーを押した場合にだけトリガされます。

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onCancel = function(file:FileReference):Void {
  trace("onCancel");
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
```

```
var url:String = "http://www.adobe.com/platform/whitepapers/
  platform_overview.pdf";
if(!fileRef.download(url, "FlashPlatform.pdf")) {
  trace("dialog box failed to open.");
}
```

onComplete (FileReference.onComplete イベントリスナー)

```
onComplete = function(fileRef:FileReference) {}
```

アップロード処理またはダウンロード処理が正常に完了したときに呼び出されます。正常に完了したということは、ファイル全体がアップロードまたはダウンロードされたことを意味します。ファイルのダウンロードの場合、Flash Player がファイル全体をディスクにダウンロードし終わると、このイベントリスナーが呼び出されます。ファイルのアップロードの場合、Flash Player がデータ伝送先サーバーから HTTP ステータスコード 200 を受け取った後、このイベントリスナーが呼び出されます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`fileRef:FileReference` - 処理を開始した FileReference オブジェクト。

例

次の例では、onComplete イベントがトリガされた場合のメッセージをトレースします。

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.adobe.com/platform/whitepapers/
  platform_overview.pdf";
fileRef.download(url, "FlashPlatform.pdf");
```

onHTTPError (FileReference.onHTTPError イベントリスナー)

```
onHTTPError = function(fileRef:FileReference, httpError:Number) {}
```

HTTP エラーのためにアップロードが失敗したときに呼び出されます。

ファイルをダウンロードする際、Flash Player はブラウザのスタックを頼りにするので、このエラーはダウンロードエラーでは利用されません。HTTP エラーのためにダウンロードが失敗した場合は、I/O エラーとして通知されます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`fileRef:FileReference` - 処理を開始した FileReference オブジェクト。

`httpError:Number` - このアップロードの失敗原因である HTTP エラー。たとえば、`httpError` が 404 である場合はページが見つからなかったことを表します。HTTP エラーの値は、ftp://ftp.isi.edu/in-notes/rfc2616.txt にある HTTP 仕様書のセクション 10.4 と 10.5 に記載されています。

例

次の例では、`onHttpError` など、想定されるイベント用のリスナーを含む FileReference オブジェクトを作成します。このリスナーは、HTTP エラーが原因でアップロードが失敗した場合にのみトリガされます。

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onSelect = function(file:FileReference):Void {
  trace("onSelect: " + file.name);
  if(!file.upload("http://www.yourdomain.com/yourUploadHandlerScript.cfm")) {
    trace("Upload dialog failed to open.");
  }
}

listener.onCancel = function(file:FileReference):Void {
  trace("onCancel");
}

listener.onOpen = function(file:FileReference):Void {
  trace("onOpen: " + file.name);
}
```

```
listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

listener.onHTTPError = function(file:FileReference):Void {
  trace("onHTTPError: " + file.name);
}

listener.onIOError = function(file:FileReference):Void {
  trace("onIOError: " + file.name);
}

listener.onSecurityError = function(file:FileReference, errorString:String):
  Void {
  trace("onSecurityError: " + file.name + " errorString: " + errorString);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

onIOError (FileReference.onIOError イベントリスナー)

```
onIOError = function(fileRef:FileReference) {}
```

入出力エラーが発生したときに呼び出されます。

このリスナーは、次のいずれかの原因でアップロードまたはダウンロードが失敗したときに呼び出されます。

- Player でファイルの読み込み中、書き込み中、または転送中に入出力エラーが発生した場合。
- ユーザー名とパスワードなど、認証が必要なサーバーに SWF がファイルをアップロードしようとした場合。アップロードする際、Flash Player には、ユーザーがパスワードを入力する手段がありません。認証が必要なサーバーに対して SWF がファイルをアップロードしようとすると、アップロードは失敗します。

- スタンドアローンまたは外部 Player で、認証が必要なサーバーから SWF ファイルがファイルをダウンロードしようとした場合。ダウンロードする際、スタンドアローンまたは外部 Player には、ユーザーがパスワードを入力する手段がありません。認証が必要なサーバーに対して、これらの Player 内の SWF がファイルをダウンロードしようとすると、ダウンロードは失敗します。ファイルのダウンロードは、ActiveX コントロール Player やブラウザプラグイン Player でのみ成功する可能性があります。
- `upload()` の `url` パラメータに渡された値に、無効なプロトコルが含まれている場合。有効なプロトコルは HTTP と HTTPS です。

重要 : ブラウザ内で実行される、つまり、ブラウザプラグインまたは ActiveX コントロールを使用する Flash アプリケーションでのみ、認証用のユーザー名とパスワードをユーザーが入力できるダイアログボックスを表示できます。ただし、それはダウンロードの場合のみです。プラグインまたは ActiveX コントロールを使用してアップロードする場合や、スタンドアローンまたは外部 Player を使用してアップロードまたはダウンロードする場合、ファイル転送は失敗します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`fileRef:FileReference` - 処理を開始した FileReference オブジェクト。

例

次の例では、`onIOError` イベントがトリガされたときのメッセージをトレースします。わかりやすくするため、この例に他のイベントリスナーは含まれていません。

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onIOError = function(file:FileReference):Void {
  trace("onIOError");
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.download("http://www.adobe.com/NonExistentFile.pdf",
  "NonExistentFile.pdf");
```

onOpen (FileReference.onOpen イベントリスナー)

```
onOpen = function(fileRef:FileReference) {}
```

アップロード処理またはダウンロード処理が開始するときに呼び出されます。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

`fileRef:FileReference` - 処理を開始した FileReference オブジェクト。

例

次の例では、onOpen イベントがトリガされたときのメッセージをトレースします。

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onOpen = function(file:FileReference):Void {
  trace("onOpen: " + file.name);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.adobe.com/platform/whitepapers/
  platform_overview.pdf";
fileRef.download(url, "FlashPlatform.pdf");
```

onProgress (FileReference.onProgress イベントリスナー)

```
onProgress = function(fileRef:FileReference, bytesLoaded:Number,
  bytesTotal:Number) {}
```

ファイルのアップロード処理中またはダウンロード処理中に定期的に呼び出されます。onProgress リスナーは、Flash Player がバイトをサーバーに転送しているときに呼び出され、最終的に転送が成功しなくても、転送の実行中は定期的に呼び出されます。ファイルの転送が成功して完了したかどうか、およびそのタイミングを確認するには、onComplete を使用します。

場合によっては、onProgress リスナーが呼び出されないこともあります。たとえば、転送対象ファイルが非常に小さい場合や、アップロードやダウンロードが非常に短時間に終わる場合などです。

ファイルアップロードの進捗状況は、OS X 10.3 より前の Macintosh プラットフォームでは確認できません。onProgress イベントはアップロード処理中に呼び出されますが、bytesLoaded パラメータの値は、進捗状況を確認できないことを示す -1 です。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`fileRef:FileReference` - 処理を開始した FileReference オブジェクト。

`bytesLoaded:Number` - これまで転送されたバイト数。

`bytesTotal:Number` - 転送するファイルの全体サイズ (バイト単位)。サイズが決定できない場合、この値は -1 になります。

例

次の例では、onProgress イベントリスナーを使用してダウンロードの進捗状況をトレースします。

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress: " + file.name + " with bytesLoaded: " + bytesLoaded + "
  bytesTotal: " + bytesTotal);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.adobe.com/platform/whitepapers/
  platform_overview.pdf";
fileRef.download(url, "FlashPlatform.pdf");
```

関連項目

onComplete (FileReference.onComplete イベントリスナー)

onSecurityError (FileReference.onSecurityError イベントリスナー)

```
onSecurityError = function(fileRef:FileReference, errorString:String) {}
```

セキュリティエラーのためにアップロードまたはダウンロードが失敗したときに呼び出されます。呼び出し側の SWF ファイルが自分のドメインの外にある SWF ファイルにアクセスしようとして、アクセス権限がないと発生します。クロスドメインポリシーファイルを使用することで、このエラーに対処できます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

fileRef:FileReference - 処理を開始した FileReference オブジェクト。

errorString:String - onSecurityError が呼び出される原因となったエラーを示します。この値は "securitySandboxError" です。

例

次の例では、onSecurityError など、想定されるイベント用のリスナーを含む FileReference オブジェクトを作成します。この onSecurityError リスナーは、セキュリティエラーが原因でアップロードが失敗した場合にのみトリガされます。

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onSelect = function(file:FileReference):Void {
  trace("onSelect: " + file.name);
  if(!file.upload("http://www.yourdomain.com/yourUploadHandlerScript.cfm")) {
    trace("Upload dialog failed to open.");
  }
}

listener.onCancel = function(file:FileReference):Void {
  trace("onCancel");
}

listener.onOpen = function(file:FileReference):Void {
  trace("onOpen: " + file.name);
}
```

```
listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

listener.onHTTPError = function(file:FileReference):Void {
  trace("onHTTPError: " + file.name);
}

listener.onIOError = function(file:FileReference):Void {
  trace("onIOError: " + file.name);
}

listener.onSecurityError = function(file:FileReference, errorString:String):Void
  {
  trace("onSecurityError: " + file.name + " errorString: " + errorString);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

onSelect (FileReference.onSelect イベントリスナー)

```
onSelect = function(fileRef:FileReference) {}
```

ユーザーがアップロードするファイルまたはダウンロードするファイルをファイル参照ダイアログボックスから選択したときに、呼び出されます。このダイアログボックスは、FileReference.browse() メソッド、FileReferenceList.browse() メソッド、または FileReference.download() メソッドを呼び出したときに表示されます。ユーザーがファイルを選択し、[OK] などをクリックして操作を確認すると、FileReference オブジェクトのプロパティに値が設定されます。

onSelect リスナーは、どのメソッドによって呼び出されたかに応じて、少し異なる動作を実行します。browse() 呼び出しの後に onSelect が呼び出された場合、Flash Player は FileReference オブジェクトのすべてのプロパティを読み取ることができます。これは、ユーザーが選択したファイルが、ローカルファイルシステムに存在するためです。download() 呼び出しの後に onSelect が呼び出された場合、Flash Player は name プロパティのみを読み取ることができます。これは、onSelect が呼び出された時点では、ファイルがまだローカルファイルシステムにダウンロードされていないためです。ファイルがダウンロードされた後、onComplete が呼び出された時点で、Flash Player は FileReference オブジェクトのすべてのプロパティを読み取ることができます。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

`fileRef:FileReference` - 処理を開始した FileReference オブジェクト。

例

次の例は、onSelect イベントリスナー内のメッセージをトレースします。

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("onSelect: " + file.name);
  if(!file.upload("http://www.yourdomain.com/yourUploadHandlerScript.cfm")) {
    trace("Upload dialog failed to open.");
  }
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

onUploadCompleteData (FileReference.onUploadCompleteData イベントリスナー)

```
onUploadCompleteData = function(fileRef:FileReference, data:String) {}
```

アップロードが正常に終了した後にサーバーからデータを受信すると呼び出されます。サーバーからデータが返されない場合、このリスナーは呼び出されません。

対応バージョン : ActionScript 2.0、Flash Player 9.0.28.0

パラメータ

`fileRef:FileReference` - 処理を開始した FileReference オブジェクト。

`data:String` - アップロードが正常に終了するとサーバーから返される未処理のデータ。

postData (FileReference.postData プロパティ)

```
public postData : String
```

アップロードまたはダウンロード時に送信する POST パラメータ

使用できるバージョン : ActionScript 1.0、Flash Player 8

関連項目

upload (FileReference.upload メソッド),download (FileReference.download メソッド)

removeListener (FileReference.removeListener メソッド)

```
public removeListener(listener:Object) :Boolean
```

イベント通知メッセージを受信するオブジェクトのリストからオブジェクトを削除します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`listener:Object` - FileReference イベントリスナーからのコールバック通知を待機するオブジェクト。

戻り値

`Boolean` - `listener` パラメータで指定したオブジェクトが正常に削除された場合に `true` を返します。それ以外の場合は `false` を返します。

例

次の例では、`removeListener` メソッドを使用してイベントリスナーを削除します。ユーザーがダウンロードをキャンセルした場合に、今後その FileReference オブジェクトからイベントを受け取ることがないよう、リスナーが削除されます。

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onCancel = function(file:FileReference):Void {
  trace(file.removeListener(this)); // true
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.adobe.com/platform/whitepapers/
  platform_overview.pdf";
fileRef.download(url, "FlashPlatform.pdf");
```

size (FileReference.size プロパティ)

public size : Number (読み取り専用)

ローカルディスク上に存在するファイルのサイズ (バイト単位) です。FileReference オブジェクトに値が設定されていない場合に、このプロパティの値を取得する呼び出しが行われると null を返します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例は、ユーザーによって選択されたファイルのサイズを取得します。

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("size: " + file.size + " bytes");
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

関連項目

browse (FileReference.browse メソッド)

type (FileReference.type プロパティ)

public type : String (読み取り専用)

ファイル形式です。Window の場合、このプロパティはファイル拡張子になります。Macintosh の場合、このプロパティは 4 文字のファイルタイプになります。FileReference オブジェクトに値が設定されていない場合に、このプロパティの値を取得する呼び出しが行われると null を返します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例は、ユーザーによって選択されたファイルの形式を取得します。

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("type: " + file.type);
}
```

```
var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

関連項目

browse (FileReference.browse メソッド)

upload (FileReference.upload メソッド)

```
public upload(url:String, uploadDataFieldName:String, testUpload:Boolean) :
  Boolean
```

ユーザーが選択したファイルをリモートサーバーにアップロードする処理を開始します。Flash Player ではアップロードまたはダウンロードできるファイルのサイズに制限はありませんが、このプレーヤーが正式にサポートしているアップロードまたはダウンロードのサイズは最大 100 MB です。このメソッドを呼び出す前に、FileReference.browse() または FileReferenceList.browse() を呼び出す必要があります。

リスナーは、アップロードが進行中なのか、成功したのか、失敗したのかを示すイベントを受け取ります。FileReferenceList オブジェクトを使用すると、ユーザーが複数のファイルを選択してアップロードすることが可能になりますが、ファイルは 1 つずつアップロードする必要があります。この操作を実行するには、FileReference オブジェクトの FileReferenceList.fileList 配列で繰り返し処理を実行します。

FileReference.upload() 関数と FileReference.download() 関数はノンブロッキング処理を行います。これらの関数は呼び出された後、ファイル転送が完了する前に返されます。さらに、FileReference オブジェクトがスコープ外に移動した場合、そのオブジェクトに対して完了していないアップロードまたはダウンロードは、スコープから離れた時点でキャンセルされます。アップロードまたはダウンロードの続行を期待できる間は、FileReference オブジェクトがスコープ内に残ることを確認してください。

ファイルは、url パラメータに渡された URL にアップロードされます。URL は、アップロードを許可するよう設定されたサーバースクリプトである必要があります。Flash Player は、HTTP POST メソッドを使用してファイルをアップロードします。アップロードを処理するサーバースクリプトは、次のエレメントを持つ POST リクエストを想定しています。

- multipart/form-data の Content-Type エレメント
- name 属性がデフォルトで "Filedata" に、filename 属性が元のファイル名に設定された Content-Disposition エレメント
- ファイルのバイナリコンテンツ

POST リクエストの例を次に示します。

```
Content-Type: multipart/form-data; boundary=AaB03x
--AaB03x
Content-Disposition: form-data; name="Filedata"; filename="example.jpg"
Content-Type: application/octet-stream
... contents of example.jpg ...
--AaB03x--
```

POST パラメータをサーバーに送信するには、FileReference.postData の値に任意のパラメータを設定します。URL にパラメータを追加することにより、upload() 呼び出しを使ってサーバーに GET パラメータを送信できます。

アップロード対象のファイルが約 10 KB を超える場合、Windows 用の Flash Player は、転送が成功するかどうか検証するために、実際のファイルをアップロードする前にテストアップロードとして中身がゼロの POST を送信します。実際のファイル内容は、2 番目の POST に含まれます。ファイルのサイズが小さい場合、Flash Player は、アップロード対象のファイルを含む POST のアップロードのみを実行します。Macintosh 用 Flash Player は現在、POST テストアップロードを行っていません。

メモ: サーバーでユーザー認証が必要な場合、ブラウザ内で実行される、つまり、ブラウザプラグインまたは ActiveX コントロールを使用する SWF ファイルでのみ、認証用のユーザー名とパスワードをユーザーが入力できるダイアログボックスを表示できます。ただし、それはダウンロードの場合のみです。プラグインまたは ActiveX コントロールを使用するアップロードの場合、および、スタンドアローンまたは外部 Player を使用するアップロードとダウンロードの場合、ファイル転送は失敗します。

このメソッドを使用するときは、Flash Player セキュリティモデルを考慮してください。

- 呼び出し元 SWF ファイルが信頼されないコードとしてローカルのサンドボックスに置かれている場合、このメソッドは使用できません。
- デフォルトのセキュリティ設定では、サンドボックス間のアクセスは拒否されます。クロスドメインポリシーファイルを追加することによって、Web サイトでリソースにアクセスできるようになります。

詳細については、次の参照先を参照してください。

- 『ActionScript 2.0 の学習』の「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

対応バージョン: ActionScript 1.0、Flash Player 8

パラメータ

url:String - HTTP POST 呼び出しを使ってアップロードを処理するよう設定されたサーバースクリプトの URL です。この URL では、HTTP または HTTPS (セキュアアップロード) を使用できます。

uploadDataFieldName:String - アップロード POST のファイルデータに先行するフィールド名です。このパラメータは、Flash Player 9.0.28.0 以降でのみサポートされています。

uploadDataFieldName 値は、null 以外、空白以外のストリングである必要があります。

uploadDataFieldName のデフォルト値は "Filedata" です。

```
Content-Type: multipart/form-data; boundary=AaB03x
--AaB03x
Content-Disposition: form-data; name="Filedata"; filename="example.jpg"
Content-Type: application/octet-stream
... contents of example.jpg ...
--AaB03x--
```

testUpload:Boolean - テストファイルアップロードを要求するための設定です。testUpload が true でファイルの大きさが 10 KB を超える場合、Flash Player は、コンテンツ長 0 でテストファイルアップロードの POST を試行します。テストアップロードの目的は、実際のファイルのアップロードが成功するかどうかと、サーバー認証が必要な場合はそれが成功するかどうかをチェックすることです。testUpload のデフォルト値は false です。現時点では、テストアップロードは Windows 版でのみ行われます。

URL にパラメータを追加することにより、upload() 呼び出しを使ってサーバーに GET パラメータを送信できます (たとえば、http://www.myserver.com/upload.cgi?userID=jdoe)。

一部のブラウザでは、URL ストリングの長さに制限があるものがあります。長さが 256 文字を超える場合、一部のブラウザまたはサーバーでは失敗する場合があります。

戻り値

Boolean - 次のいずれかの状況に合致する場合、false を返します。

- このオブジェクトで FileReference.browse() がまだ正常に呼び出されていない場合、またはこのオブジェクトを使って FileReferenceList.browse() がまだその filelist 配列内で正常に呼び出されていない場合。
- プロトコルが HTTP または HTTPS でなかった場合。
- セキュリティ侵害が発生した場合。つまり、SWF ファイルが、そのセキュリティサンドボックス外にあるサーバーのファイルにアクセスしようとした場合。
- url パラメータの形式が正しくなかった場合。
- 呼び出しで、正しいパラメータ数が設定されていない場合。

例

次の例は、まずユーザーに対してアップロードするファイルを選択するよう促し、onSelect リスナーと onCancel リスナーを処理した後で、最終的に実際のファイルアップロードの結果を処理する upload() メソッドの実行形態を示します。

```
import flash.net.FileReference;

var allTypes:Array = new Array();
var imageTypes:Object = new Object();
imageTypes.description = "Images (*.jpg, *.jpeg, *.gif, *.png)";
imageTypes.extension = "*.jpg; *.jpeg; *.gif; *.png";
allTypes.push(imageTypes);

var listener:Object = new Object();

listener.onSelect = function(file:FileReference):Void {
  trace("onSelect: " + file.name);
  if(!file.upload("http://www.yourdomain.com/yourUploadHandlerScript.cfm")) {
    trace("Upload dialog failed to open.");
  }
}

listener.onCancel = function(file:FileReference):Void {
  trace("onCancel");
}

listener.onOpen = function(file:FileReference):Void {
  trace("onOpen: " + file.name);
}

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

listener.onHTTPError = function(file:FileReference):Void {
  trace("onHTTPError: " + file.name);
}

listener.onIOError = function(file:FileReference):Void {
  trace("onIOError: " + file.name);
}
```

```
listener.onSecurityError = function(file:FileReference, errorString:String):Void {
  trace("onSecurityError: " + file.name + " errorString: " + errorString);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse(allTypes);
```

関連項目

browse (FileReference.browse メソッド), browse (FileReferenceList.browse メソッド), download (FileReference.download メソッド), fileList (FileReferenceList.fileList プロパティ)

FileReferenceList (flash.net.FileReferenceList)

```
Object
  |
  +-flash.net.FileReferenceList
```

```
public class FileReferenceList
extends Object
```

FileReferenceList クラスには、ユーザーがアップロードするファイルを選択する手段 (複数選択可能) があります。FileReferenceList オブジェクトは、ユーザーのディスク上にあるローカルファイルを FileReference オブジェクトの配列として表現します。FileReference オブジェクトと FileReference クラスの詳細および重要な考慮事項については、「FileReference クラス」を参照してください。これらは FileReferenceList で使用します。

FileReferenceList クラスを使用するには

- クラス `var myFileRef = new FileReferenceList();` のインスタンスを作成します。
- `FileReferenceList.browse()` を呼び出して、ユーザーがアップロードするファイルを選択できるダイアログボックス (`myFileRef.browse();`) を表示します。
- `browse()` が正常に呼び出された後、FileReferenceList オブジェクトの `fileList` プロパティに FileReference オブジェクトの配列が設定されます。
- `fileList` 配列内の各エレメントについて `FileReference.upload()` を呼び出します。

FileReferenceList クラスには、`browse()` メソッドや、複数のファイルを使用するための `fileList` プロパティなどがあります。`FileReferenceList.browse()` の呼び出し中、Linux および Mac OS X 10.1 以前のスタンドアローンの外部 Player では SWF ファイルの再生が一時停止します。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、ユーザーが複数のファイルを選択して、サーバーにアップロードできます。

```
import flash.net.FileReferenceList;
import flash.net.FileReference;

var listener:Object = new Object();

listener.onSelect = function(fileRefList:FileReferenceList) {
  trace("onSelect");
  var list:Array = fileRefList.fileList;
  var item:FileReference;
  for(var i:Number = 0; i < list.length; i++) {
    item = list[i];
    trace("name: " + item.name);
    trace(item.addListener(this));
    item.upload("http://www.yourdomain.com/");
  }
}

listener.onCancel = function():Void {
  trace("onCancel");
}

listener.onOpen = function(file:FileReference):Void {
  trace("onOpen: " + file.name);
}

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

listener.onHTTPError = function(file:FileReference, httpError:Number):Void {
  trace("onHTTPError: " + file.name + " httpError: " + httpError);
}

listener.onIOError = function(file:FileReference):Void {
  trace("onIOError: " + file.name);
}

listener.onSecurityError = function(file:FileReference, errorString:String):Void
  {
  trace("onSecurityError: " + file.name + " errorString: " + errorString);
}
```

```
var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

関連項目

FileReference (flash.net.FileReference)

プロパティー覧

オプション	プロパティ	説明
	fileList:Array	FileReference オブジェクトの配列です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベントの一覧

イベント	説明
onCancel = function(fileRefList: FileReferenceList) {}	ユーザーがファイル参照ダイアログボックスを閉じると、呼び出されます。
onSelect = function(fileRefList: FileReferenceList) {}	ユーザーがアップロードするファイルをファイル参照ダイアログボックスから 1 つ以上選択したときに呼び出されます。

コンストラクター覧

署名	説明
FileReferenceList()	新しい FileReferenceList オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	addListener (listener:Object) : Void	FileReferenceList イベントリスナーが呼び出されたときに通知を受けるオブジェクトを登録します。
	browse([typelist: Array]) : Boolean	アップロードするローカルファイルを 1 つ以上選択できるファイル参照ダイアログボックスを表示します。
	removeListener (listener:Object) : Boolean	イベント通知メッセージを受信するオブジェクトのリストからオブジェクトを削除します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

addListener (FileReferenceList.addListener メソッド)

public addListener(listener:Object) : Void

FileReferenceList イベントリスナーが呼び出されたときに通知を受けるオブジェクトを登録します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

listener:Object - FileReferenceList イベントリスナーからのコールバック通知を待機するオブジェクト。

例

次の例では、addListener() メソッドを説明します。

```
import flash.net.FileReferenceList;

var listener:Object = new Object();
listener.onCancel = function(fileRefList:FileReferenceList) {
  trace("onCancel");
}
```

```
listener.onSelect = function(fileRefList:FileReferenceList) {
  trace("onSelect: " + fileRefList.fileList.length);
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

browse (FileReferenceList.browse メソッド)

```
public browse([typelist:Array]) : Boolean
```

アップロードするローカルファイルを 1 つ以上選択できるファイル参照ダイアログボックスを表示します。このダイアログボックスは、オペレーティングシステムのネイティブのダイアログボックスです。このメソッドを呼び出して、ユーザーが正常にファイルを選択すると、この FileReferenceList オブジェクトの `fileList` プロパティに FileReference オブジェクトの配列が設定されます。FileReference オブジェクトはユーザーが選択したファイルごとに作成されます。これ以降 `FileReferenceList.browse()` が呼び出されるたびに、FileReferenceList.fileList プロパティは、ダイアログボックスでユーザーが選択したファイルにリセットされます。

どのファイルをダイアログボックスに表示するかを決定するために、ファイルタイプの配列を渡すことができます。

FileReferenceList オブジェクトに対し、一度に 1 つの `browse()` セッションまたは `download()` セッションだけを実行できます。これは、一度に 1 つのダイアログボックスしか表示できないからです。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`typelist:Array` (オプション) - ダイアログボックスに表示するファイルをフィルタにかける場合に使用するファイルタイプの配列です。このパラメータを省略すると、すべてのファイルが表示されます。このパラメータを設定する場合は、エレメントを中カッコ ({ }) で囲んで配列に入れる必要があります。配列の形式として、次の 2 つのいずれかを使用できます。

- Windows ファイル拡張子だけが含まれるファイル形式の記述リスト。配列の各エレメントには、ファイル形式を説明するストリングと、Windows ファイル拡張子をセミコロンで区切ったリストを入れる必要があります。各拡張子の前にはワイルドカード (*) を付けます。各エレメントのシンタックスは次のとおりです。`[{description: "`*string describing the first set of file types*`", extension: "`*semicolon-delimited list of file extensions*`"}]`

  例 : `[{description: "Images", extension: "*.jpg;*.gif;*.png"}, {description: "SWF files", extension: "*.swf"}, {description: "Documents", extension: "*.doc;*.pdf"}]`

- Windows ファイル拡張子と Macintosh ファイル形式が含まれるファイル形式の記述リスト。配列の各エレメントには、ファイル形式を説明するストリングを含む必要があります。つまり、各拡張子の前にワイルドカード (*) を付けた状態で Windows ファイル拡張子をセミコロンで区切ったリスト、および各ファイル形式の前にワイルドカード (*) を付けた状態で Macintosh ファイル形式をセミコロンで区切ったリストを含む必要があります。各エレメントのシンタックスは次のとおりです。`[{description: "string describing the first set of file types", extension: "semicolon-delimited list of Windows file extensions", macType: "semicolon-delimited list of Macintosh file types"}]` 例:
  `[{description: "Image files", extension: "*.jpg;*.gif;*.png", macType: "JPEG;jp2_;GIFf;PNGf"}, {description: "SWF files", extension: "*.swf", macType: "SWFL"}]`

この 2 つの形式は、1 つの `browse()` 呼び出しの中で混在できません。どちらか一方を使用する必要があります。

拡張子のリストは、ユーザーが選択したファイル形式に応じて、Windows のファイルをフィルタにかけるために使用します。ダイアログボックスに実際に表示されるわけではありません。ファイル形式をユーザーに表示するには、拡張子リストのほか、説明用ストリングにもファイル形式をリストする必要があります。説明用ストリングは、Windows のダイアログボックスに表示されます。Macintosh では使用されません。Macintosh では、Macintosh のファイル形式リストを指定すると、このリストがファイルのフィルタリングに使用されます。Macintosh のファイルタイプのリストを指定しない場合、Windows 拡張子のリストが使用されます。

戻り値

`Boolean` - パラメータが有効で、ファイル参照ダイアログボックスが表示された場合に、`true` を返します。ダイアログボックスが表示されない場合、既に他のブラウザセッションが進行中である場合、または `typelist` パラメータが使用されたが、配列内のエレメントに説明用文字列または拡張子ストリングが指定されていない場合は、`false` を返します。

例

次の例では、`browse()` メソッドを説明します。

```
import flash.net.FileReferenceList;

var allTypes:Array = new Array();
var imageTypes:Object = new Object();
imageTypes.description = "Images (*.JPG;*.JPEG;*.JPE;*.GIF;*.PNG;)";
imageTypes.extension = "*.jpg; *.jpeg; *.jpe; *.gif; *.png;";
allTypes.push(imageTypes);

var textTypes:Object = new Object();
textTypes.description = "Text Files (*.TXT;*.RTF;)";
textTypes.extension = "*.txt; *.rtf";
```

```
allTypes.push(textTypes);

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.browse(allTypes);
```

関連項目

browse (FileReference.browse メソッド),
FileReference (flash.net.FileReference)

fileList (FileReferenceList.fileList プロパティ)

public fileList : Array

FileReference オブジェクトの配列です。

FileReferenceList.browse() メソッドが呼び出され、ユーザーが browse() によって開いたダイアログボックスからファイルを選択すると、このプロパティに FileReference オブジェクトの配列が設定されます。このオブジェクトは、ユーザーが選択したファイルを表します。その後、この配列を使用して、FileReference.upload() で各ファイルをアップロードできます。一度に 1 つのファイルをアップロードする必要があります。

fileList プロパティは、browse() が FileReferenceList オブジェクトで呼び出されるたびに設定されます。

FileReference オブジェクトのプロパティは、FileReference クラスに記載されています。

対応バージョン: ActionScript 1.0、Flash Player 8

例

次の例では、fileList プロパティを説明します。

```
import flash.net.FileReferenceList;
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(fileRefList:FileReferenceList) {
  trace("onSelect");
  var list:Array = fileRefList.fileList;
  var item:FileReference;
  for(var i:Number = 0; i < list.length; i++) {
    item = list[i];
    trace("name: " + item.name);
  }
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

関連項目

FileReference (flash.net.FileReference), upload (FileReference.upload メソッド), browse (FileReferenceList.browse メソッド)

FileReferenceList コンストラクタ

```
public FileReferenceList()
```

新しい FileReferenceList オブジェクトを作成します。このオブジェクトには、browse() が呼び出されるまで何も設定されません。FileReference オブジェクトで browse() を呼び出すと、このオブジェクトの fileList プロパティに FileReference オブジェクトの配列が設定されます。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、新しい FileReferenceList オブジェクトを作成し、選択されたファイルに繰り返し処理を実行して、名前を出力します。

```
import flash.net.FileReferenceList;

var listener:Object = new Object();
listener.onSelect = function(fileRefList:FileReferenceList) {
  trace("onSelect");
  var arr:Array = fileRefList.fileList;
  for(var i:Number = 0; i < arr.length; i++) {
    trace("name: " + arr[i].name);
  }
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

関連項目

FileReference (flash.net.FileReference), browse (FileReferenceList.browse メソッド)

onCancel (FileReferenceList.onCancel イベントリスナー)

```
onCancel = function(fileRefList:FileReferenceList) {}
```

ユーザーがファイル参照ダイアログボックスを閉じると、呼び出されます。(このダイアログボックスは、FileReferenceList.browse() メソッド、FileReference.browse() メソッド、または FileReference.download() メソッドが呼び出されたときに表示されます)。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`fileRefList:FileReferenceList` - 処理を開始した FileReferenceList オブジェクトです。

例

次の例では、onCancel リスナーを説明します。

```
import flash.net.FileReferenceList;

var listener:Object = new Object();
listener.onCancel = function(fileRefList:FileReferenceList) {
  trace("onCancel");
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

関連項目

browse (FileReferenceList.browse メソッド)

onSelect (FileReferenceList.onSelect イベントリスナー)

```
onSelect = function(fileRefList:FileReferenceList) {}
```

ユーザーがアップロードするファイルをファイル参照ダイアログボックスから 1 つ以上選択したときに呼び出されます。(このダイアログボックスは、FileReferenceList.browse() メソッド、FileReference.browse() メソッド、または FileReference.download() メソッドが呼び出されたときに表示されます)。ユーザーがファイルを選択し、[保存] などをクリックして操作を確認すると、ユーザーが選択したファイルを表す FileReference オブジェクトが FileReferenceList オブジェクトに設定されます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

fileRefList:FileReferenceList - 処理を開始した FileReferenceList オブジェクトです。

例

次の例では、onSelect リスナーを説明します。

```
import flash.net.FileReferenceList;
import flash.net.FileReference;

var listener:Object = new Object();

listener.onSelect = function(fileRefList:FileReferenceList) {
  trace("onSelect");
  var list:Array = fileRefList.fileList;
  var item:FileReference;
  for(var i:Number = 0; i < list.length; i++) {
    item = list[i];
    trace("name: " + item.name);
    trace(item.addListener(this));
    item.upload("http://www.yourdomain.com/");
  }
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

関連項目

browse (FileReferenceList.browse メソッド)

removeListener (FileReferenceList.removeListener メソッド)

```
public removeListener(listener:Object) : Boolean
```

イベント通知メッセージを受信するオブジェクトのリストからオブジェクトを削除します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`listener:Object` - FileReferenceList イベントリスナーからのコールバック通知を待機するオブジェクト。

戻り値

`Boolean` - オブジェクトが削除された場合は `true` を返します。それ以外の場合は `false` を返します。

例

次の例では、`removeListener` メソッドを説明します。

```
import flash.net.FileReferenceList;

var listener:Object = new Object();
listener.onCancel = function(fileRefList:FileReferenceList) {
  trace("onCancel");
  trace(fileRefList.removeListener(this)); // true
}

listener.onSelect = function(fileRefList:FileReferenceList) {
  trace("onSelect: " + fileRefList.fileList.length);
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

Function

```
Object
  |
  +-Function
```

```
public dynamic class Function
extends Object
```

ActionScript のユーザー定義関数とビルトイン関数は、どちらも Function クラスのインスタンスである Function オブジェクトで表されます。

対応バージョン： ActionScript 1.0、Flash Player 6

プロパティー覧

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

メソッド一覧

オプション	署名	説明
	apply(thisObject: Object, [argArray:Array]) : Void	ActionScript が呼び出す関数内で使用される thisObject の値を指定します。
	call(thisObject: Object, [parameter1:Object]) : Object	Function オブジェクトが表す関数を呼び出します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

apply (Function.apply メソッド)

```
public apply(thisObject:Object, [argArray:Array]) : Void
```

ActionScript が呼び出す関数内で使用される thisObject の値を指定します。このメソッドは、呼び出される関数に渡されるパラメータも指定します。apply() は Function クラスのメソッドなので、ActionScript 内のすべての Function オブジェクトのメソッドとしても使用できます。

パラメータは、カンマ区切りリストとしてパラメータを指定する Function.call() とは異なり、Array オブジェクトとして指定します。これは、スクリプトが実際に実行されるまで、渡されるパラメータ数が不明である場合にも便利です。

呼び出された関数が戻り値として指定する値を返します。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

thisObject:Object - myFunction の適用先のオブジェクト。

argArray:Array (オプション) - エレメントをパラメータとして myFunction に渡す配列。

例

次の 2 つの関数呼び出しは同じです。

```
Math.atan2(1, 0)
Math.atan2.apply(null, [1, 0])
```

次の簡単な例は、apply() を使用してパラメータの配列を渡す方法を示しています。

```
function theFunction() {
  trace(arguments);
}

// create a new array to pass as a parameter to apply()
var firstArray:Array = new Array(1,2,3);
theFunction.apply(null,firstArray);
// outputs: 1,2,3

// create a second array to pass as a parameter to apply()
var secondArray:Array = new Array("a", "b", "c");
theFunction.apply(null,secondArray);
// outputs a,b,c
```

次の例では、apply() を使用してパラメータの配列を渡し、this の値を指定します。

```
// define a function
function theFunction() {
  trace("this == myObj? " + (this == myObj));
  trace("arguments: " + arguments);
}
```

```
// instantiate an object
var myObj:Object = new Object();

// create arrays to pass as a parameter to apply()
var firstArray:Array = new Array(1,2,3);
var secondArray:Array = new Array("a", "b", "c");

// use apply() to set the value of this to be myObj and send firstArray
theFunction.apply(myObj,firstArray);
// output:
// this == myObj? true
// arguments: 1,2,3

// use apply() to set the value of this to be myObj and send secondArray
theFunction.apply(myObj,secondArray);
// output:
// this == myObj? true
// arguments: a,b,c
```

関連項目

call (Function.call メソッド)

call (Function.call メソッド)

```
public call(thisObject:Object, [parameter1:Object]) : Object
```

Function オブジェクトが表す関数を呼び出します。ActionScript のすべての関数は Function オブジェクトによって表されます。したがって、すべての関数は、このメソッドをサポートしています。

ほとんどの場合、このメソッドの代わりに関数呼び出し演算子 (( )) を使用できます。関数呼び出し演算子を使うと、コードが簡潔になり読みやすくなります。このメソッドは、主に関数呼び出しの thisObject パラメータを明示的に制御する必要がある場合に役立ちます。通常、関数をオブジェクトのメソッドとして、関数の本体内で呼び出すと、次のように thisObject が myObject に設定されます。

```
myObject.myMethod(1, 2, 3);
```

thisObject が他の異なる場所をポイントするように設定する場合もあります。たとえば、オブジェクトのメソッドとして呼び出す関数が、実際には、そのオブジェクトのメソッドとして格納されていない場合などです。

```
myObject.myMethod.call(myOtherObject, 1, 2, 3);
```

関数をオブジェクトのメソッドとして呼び出さずに通常の関数として呼び出すには、thisObject パラメータに値 null を渡します。たとえば、次の 2 つの関数呼び出しは同じです。

```
Math.sin(Math.PI / 4)
Math.sin.call(null, Math.PI / 4)
```

呼び出された関数が戻り値として指定する値を返します。

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

thisObject:Object - 関数の本体内で thisObject の値を指定するオブジェクト。

parameter1:Object (オプション) - myFunction に渡すパラメータ。指定できるパラメータの数は 0 個以上です。

戻り値

Object -

例

次の例では、Function.call() メソッドを使用することで、関数をオブジェクトに格納しないまま、別のオブジェクトのメソッドとして動作させます。

```
function myObject() {
}
function myMethod(obj) {
  trace("this == obj? " + (this == obj));
}
var obj:Object = new myObject();
myMethod.call(obj, obj);
```

trace() ステートメントの出力は次のようになります。

```
this == obj? true
```

関連項目

apply (Function.apply メソッド)

GlowFilter (flash.filters.GlowFilter)

```
Object
  |
  +-BitmapFilter
     |
     +-flash.filters.GlowFilter
```

```
public class GlowFilter
extends BitmapFilter
```

GlowFilter クラスを使用すると、Flash の各種オブジェクトにグロー効果を適用できます。グローのスタイルには複数のオプションがあり、内側グロー、外側グロー、ノックアウトモードなどがあります。グローフィルタは、ドロップシャドウの distance プロパティと angle プロパティとをゼロに設定したドロップシャドウフィルタによく似ています。

フィルタの使い方は、フィルタの適用先オブジェクトによって異なります。

- 実行時にムービークリップ、テキストフィールド、ボタンにフィルタを適用する場合は、filters プロパティを使用します。オブジェクトの filters プロパティを設定してもオブジェクトは変更されません。また、filters プロパティをクリアすることで元に設定を戻すことができます。
- BitmapData インスタンスにフィルタを適用するには、BitmapData.applyFilter() メソッドを使用します。BitmapData オブジェクトで applyFilter() を呼び出すことによって、ソース BitmapData オブジェクトとフィルタオブジェクトが取得され、フィルタを適用した結果として得られるイメージが生成されます。

イメージやビデオにもオーサリング時にフィルタ効果を適用できます。詳細については、オーサリングのマニュアルを参照してください。

ムービークリップやボタンにフィルタを適用する場合は、ムービークリップやボタンの cacheAsBitmap プロパティを true に設定します。すべてのフィルタをクリアすると、cacheAsBitmap は元の値に戻ります。

このフィルタはステージの拡大・縮小に対応していますが、通常の拡大・縮小、回転、傾斜には対応していません。オブジェクト自体を拡大・縮小する場合 (_xscale と _yscale が 100% でない場合)、フィルタ効果は拡大・縮小されません。フィルタ効果が拡大・縮小するのは、ステージをズームする場合のみです。

結果として得られるイメージの幅または高さが 2880 ピクセルを超える場合、フィルタは適用されません。たとえば、フィルタが適用されたサイズの大きいムービークリップをズームインするとき、結果として得られるイメージが 2880 ピクセルの制限を超える場合は、フィルタがオフになります。

対応バージョン : ActionScript 1.0、Flash Player 8

関連項目

applyFilter (BitmapData.applyFilter メソッド), cacheAsBitmap (Button.cacheAsBitmap プロパティ), filters (Button.filters プロパティ), DropShadowFilter (flash.filters.DropShadowFilter), cacheAsBitmap (MovieClip.cacheAsBitmap プロパティ), filters (MovieClip.filters プロパティ), filters (TextField.filters プロパティ)

プロパティー覧

オプション	プロパティ	説明
	alpha:Number	カラーのアルファ透明度の値。
	blurX:Number	水平方向のぼかし量。
	blurY:Number	垂直方向のぼかし量。
	color:Number	グローのカラー。
	inner:Boolean	グローが内側グローであるかどうかを示します。
	knockout:Boolean	オブジェクトにノックアウト効果を適用するかどうかを指定します。
	quality:Number	フィルタを適用する回数。
	strength:Number	インプリントやスプレッドの長さ。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター覧

署名	説明
GlowFilter([color:Number], [alpha:Number], [blurX:Number], [blurY:Number], [strength:Number], [quality:Number], [inner:Boolean], [knockout:Boolean])	指定されたパラメータで新しい GlowFilter インスタンスを初期化します。

メソッド一覧

オプション	署名	説明
	clone() : GlowFilter	このフィルタオブジェクトのコピーを返します。

BitmapFilter クラスから継承されるメソッド

clone (BitmapFilter.clone メソッド)

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

alpha (GlowFilter.alpha プロパティ)

public alpha : Number

カラーのアルファ透明度の値。0 ～ 1 の範囲の値を指定できます。たとえば 0.25 と指定すると、透明度は 25% になります。デフォルト値は 1 です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その alpha プロパティを変更します。

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterAlpha");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.alpha = 0.4;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;
```

```
  var filter:GlowFilter = new GlowFilter(0x000000, 0.8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

blurX (GlowFilter.blurX プロパティ)

```
public blurX : Number
```

水平方向のぼかし量。指定できる値は 0 ～ 255 (浮動小数) です。デフォルト値は 6 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その blurX プロパティを変更します。

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterBlurX");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.blurX = 20;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, 0.8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

blurY (GlowFilter.blurY プロパティ)

```
public blurY : Number
```

垂直方向のぼかし量。指定できる値は 0 ～ 255 (浮動小数) です。デフォルト値は 6 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その blurY プロパティを変更します。

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterBlurY");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.blurY = 20;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, 0.8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

clone (GlowFilter.clone メソッド)

```
public clone() : GlowFilter
```

このフィルタオブジェクトのコピーを返します。

対応バージョン： ActionScript 1.0、Flash Player 8

戻り値

`GlowFilter` - 元の GlowFilter インスタンスのプロパティをすべて備えた新しい GlowFilter インスタンス。

例

次の例では、3 つの GlowFilter オブジェクトを作成して比較します。`filter_1` は、GlowFilter コンストラクタを使用して作成されます。`filter_2` は、`filter_1` と等しい値に設定することにより作成されます。`clonedFilter` は、`filter_1` のクローンを作成することにより作成されます。`filter_2` が `filter_1` に等しいと評価されても、`filter_1` と同じ値を含んでいる `clonedFilter` が等しいと評価されないことに注意してください。

```
import flash.filters.GlowFilter;

var filter_1:GlowFilter = new GlowFilter(0x33CCFF, 0.8, 35, 35, 2, 3, false,
  false);
var filter_2:GlowFilter = filter_1;
var clonedFilter:GlowFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false

for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> strength: 2
  // >> blurY: 35
  // >> blurX: 35
  // >> knockout: false
  // >> inner: false
  // >> quality: 3
  // >> alpha: 0.8
  // >> color: 3394815
}
```

```
for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
  // >> strength: 2
  // >> blurY: 35
  // >> blurX: 35
  // >> knockout: false
  // >> inner: false
  // >> quality: 3
  // >> alpha: 0.8
  // >> color: 3394815
}
```

filter_1、filter_2、および clonedFilter の関係をさらに詳しく示すために、次の例では、filter_1 の knockout プロパティを変更します。knockout を変更することは、clone() メソッドによって、filter_1 を参照する代わりにその値に基づいて新しいインスタンスが作成されることを示します。

```
import flash.filters.GlowFilter;

var filter_1:GlowFilter = new GlowFilter(0x33CCFF, 0.8, 35, 35, 2, 3, false,
  false);
var filter_2:GlowFilter = filter_1;
var clonedFilter:GlowFilter = filter_1.clone();

trace(filter_1.knockout); // false
trace(filter_2.knockout); // false
trace(clonedFilter.knockout); // false

filter_1.knockout = true;

trace(filter_1.knockout); // true
trace(filter_2.knockout); // true
trace(clonedFilter.knockout); // false
```

color (GlowFilter.color プロパティ)

```
public color : Number
```

グローのカラー。指定できる値は、16 進数形式 (0x *RRGGBB*) です。デフォルト値は 0xFF0000 です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その color プロパティを変更します。

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterColor");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.color = 0x00FF33;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, .8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

GlowFilter コンストラクタ

```
public GlowFilter([color:Number], [alpha:Number], [blurX:Number],
  [blurY:Number], [strength:Number], [quality:Number], [inner:Boolean],
  [knockout:Boolean])
```

指定されたパラメータで新しい GlowFilter インスタンスを初期化します。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

`color:Number` (オプション) - グローの色を 16 進数形式 (0x *RRGGBB*) で指定します。デフォルト値は 0xFF0000 です。

`alpha:Number` (オプション) - カラーのアルファ透明度の値です。0 ～ 1 の範囲の値を指定できます。たとえば .25 と指定すると、透明度は 25% になります。デフォルト値は 1 です。

`blurX:Number` (オプション) - 水平方向のぼかし量です。指定できる値は 0 ～ 255 (浮動小数) です。デフォルト値は 6 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

`blurY:Number` (オプション) - 垂直方向のぼかし量です。指定できる値は 0 ～ 255 (浮動小数) です。デフォルト値は 6 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

`strength:Number` (オプション) - インプリントやスプレッドの長さです。値が大きいほど、濃い色がインプリントされるので、グローと背景との間のコントラストが強くなります。指定できる値は 0 ～ 255 で、デフォルトは 2 です。

`quality:Number` (オプション) - フィルタを適用する回数。有効な値は 0 ～ 15 です。デフォルト値は 1 で、低品質と等価です。値 2 は標準の品質であり、値 3 は高品質です。

`inner:Boolean` (オプション) - グローが内側グローであるかどうかを示します。`true` の場合は、内側グローであることを示します。デフォルトは `false` (外側グロー) で、オブジェクトの外側のエッジにあるグローを示します。

`knockout:Boolean` (オプション) - オブジェクトにノックアウト効果を適用するかどうかを指定します。`true` を指定すると、オブジェクトの塗りが透明になり、ドキュメントの背景色が表示されます。デフォルトは `false` (ノックアウトなし) です。

例

次の例では、GlowFilter の新しいインスタンスを作成し、そのインスタンスにフラットな矩形シェイプを適用します。

```
import flash.filters.GlowFilter;

var rect:MovieClip = createRectangle(100, 100, 0x003366,
  "gradientGlowFilterExample");

var color:Number = 0x33CCFF;
var alpha:Number = 0.8;
var blurX:Number = 35;
var blurY:Number = 35;
var strength:Number = 2;
var quality:Number = 3;
var inner:Boolean = false;
var knockout:Boolean = false;

var filter:GlowFilter = new GlowFilter(color,
                        alpha,
                        blurX,
                        blurY,
                        strength,
                        quality,
                        inner,
                        knockout);
var filterArray:Array = new Array();
filterArray.push(filter);
rect.filters = filterArray;

function createRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  mc.beginFill(bgColor);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc._x = 20;
  mc._y = 20;
  return mc;
}
```

inner (GlowFilter.inner プロパティ)

```
public inner : Boolean
```

グローが内側グローであるかどうかを示します。true の場合は、内側グローであることを示します。デフォルトは false (外側グロー) で、オブジェクトの外側のエッジにあるグローを示します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その inner プロパティを変更します。

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterInner");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.inner = true;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, 0.8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

knockout (GlowFilter.knockout プロパティ)

```
public knockout : Boolean
```

オブジェクトにノックアウト効果を適用するかどうかを指定します。true を指定すると、オブジェクトの塗りが透明になり、ドキュメントの背景色が表示されます。デフォルトは false (ノックアウトなし) です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その knockout プロパティを変更します。

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterKnockout");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.knockout = true;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, 0.8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

quality (GlowFilter.quality プロパティ)

```
public quality : Number
```

フィルタを適用する回数。有効な値は 0 ～ 15 です。デフォルト値は 1 で、低品質と等価です。値 2 は標準の品質であり、値 3 は高品質です。フィルタに設定された値が小さいほど、速くレンダリングできます。

多くのアプリケーションでは、quality 値 1、2、または 3 で十分です。最大 15 までの値を使用してさまざまな効果を出すことができますが、値が大きくなるほどレンダリング速度が低下します。多くの場合、quality の値を大きくする代わりに blurX と blurY の値を大きくするだけで、同様の効果が得られます。この方法を実行すると、より高速にレンダリングされます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その quality プロパティを変更します。

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterQuality");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.quality = 1;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, 0.8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

strength (GlowFilter.strength プロパティ)

```
public strength : Number
```

インプリントの強さまたは広がり。値が大きいほど、濃い色がインプリントされるので、グローと背景との間のコントラストが強くなります。指定できる値は 0 ～ 255 で、デフォルトは 2 です。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その strength プロパティを変更します。

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterStrength");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.strength = 0.8;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, 0.8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

GradientBevelFilter (flash.filters.GradientBevelFilter)

```
Object
  |
  +-BitmapFilter
     |
     +-flash.filters.GradientBevelFilter
```

```
public class GradientBevelFilter
extends BitmapFilter
```

GradientBevelFilter クラスを使用すると、Flash の各種オブジェクトにグラデーションベベル効果を適用できます。グラデーションベベルは、オブジェクトの外側、内側、または上側が斜めになったエッジであり、グラデーションカラーで強調されます。斜めのエッジによってオブジェクトが 3 次元に見えます。

フィルタの使用方法は、フィルタの適用先オブジェクトによって異なります。

- 実行時にムービークリップ、テキストフィールド、ボタンにフィルタを適用する場合は、filters プロパティを使用します。オブジェクトの filters プロパティを設定しても、オブジェクトは変更されません。filters プロパティをクリアすることにより、フィルタを取り消すことができます。
- BitmapData インスタンスにフィルタを適用するには、BitmapData.applyFilter() メソッドを使用します。BitmapData オブジェクトで applyFilter() を呼び出すことによって、ソース BitmapData オブジェクトとフィルタオブジェクトが取得され、フィルタを適用した結果として得られるイメージが生成されます。

イメージやビデオにもオーサリング時にフィルタ効果を適用できます。詳細については、オーサリングのマニュアルを参照してください。

ムービークリップやボタンにフィルタを適用する場合は、ムービークリップやボタンの cacheAsBitmap プロパティを true に設定します。すべてのフィルタをクリアすると、cacheAsBitmap は元の値に戻ります。

このフィルタはステージの拡大・縮小に対応していますが、通常の拡大・縮小、回転、傾斜には対応していません。オブジェクト自体を拡大・縮小する場合 (_xscale と _yscale が 100% でない場合)、フィルタ効果は拡大・縮小されません。フィルタ効果が拡大・縮小するのは、ステージをズームする場合のみです。

結果として得られるイメージの幅または高さが 2880 ピクセルを超える場合、フィルタは適用されません。たとえば、フィルタが適用されたサイズの大きいムービークリップをズームインするとき、結果として得られるイメージが 2880 ピクセルの制限を超える場合は、フィルタがオフになります。

対応バージョン： ActionScript 1.0、Flash Player 8

関連項目

ratios (GradientBevelFilter.ratios プロパティ),applyFilter (BitmapData.applyFilter メソッド),BevelFilter (flash.filters.BevelFilter),filters (Button.filters プロパティ),cacheAsBitmap (Button.cacheAsBitmap プロパティ),cacheAsBitmap (MovieClip.cacheAsBitmap プロパティ),filters (MovieClip.filters プロパティ),filters (TextField.filters プロパティ)

プロパティ一覧

オプション	プロパティ	説明
	alphas:Array	colors 配列内の各色に対応するアルファ透明度の値の配列。
	angle:Number	角度 (度数)。
	blurX:Number	水平方向のぼかし量。
	blurY:Number	垂直方向のぼかし量。
	colors:Array	グラデーションで使用する RGB 16 進数カラー値の配列。
	distance:Number	オフセット距離。
	knockout:Boolean	オブジェクトにノックアウト効果を適用するかどうかを指定します。
	quality:Number	フィルタを適用する回数。
	ratios:Array	colors 配列内の対応するカラーの色分布比率の配列。
	strength:Number	インプリントやスプレッドの長さ。
	type:String	ベベル効果の配置。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ),__proto__ (Object.__proto__ プロパティ),prototype (Object.prototype プロパティ),__resolve (Object.__resolve プロパティ)

コンストラクター一覧

署名	説明
GradientBevelFilter ([distance:Number], [angle:Number], [colors:Array], [alphas:Array], [ratios:Array], [blurX:Number], [blurY:Number], [strength:Number], [quality:Number], [type:String], [knockout:Boolean])	指定されたパラメータでフィルタを初期化します。

メソッド一覧

オプション	署名	説明
	clone() : GradientBevelFilter	このフィルタオブジェクトのコピーを返します。

BitmapFilter クラスから継承されるメソッド

clone (BitmapFilter.clone メソッド)

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

alphas (GradientBevelFilter.alphas プロパティ)

public alphas : Array

colors 配列内の各色に対応するアルファ透明度の値の配列。配列内の各エレメントに指定できる値は 0 ～ 1 です。たとえば 0.25 と指定すると、透明度の値は 25% に設定されます。

alphas プロパティの値を直接変更することはできません。このプロパティを変更するには、alphas への参照を取得し、その参照を変更した後、alphas をその参照に設定する必要があります。

colors、alphas、ratios の 3 つのプロパティはすべて関係しています。colors 配列の先頭のエレメントが alphas 配列と ratios 配列の先頭のエレメントに対応するなど、3 つの配列の同じインデックスのエレメントが互いに対応しています。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例は、既存のエンティティで alphas プロパティを設定する方法を示しています。

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("alphasExample");
mc.onPress = function() {
   var arr:Array = this.filters;
   var alphas:Array = [0.2, 0, 0.2];
   arr[0].alphas = alphas;
   this.filters = arr;
}
mc.onRelease = function() {
   var arr:Array = this.filters;
   var alphas:Array = [1, 0, 1];
   arr[0].alphas = alphas;
   this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
   var art:MovieClip = this.createEmptyMovieClip(name,
   this.getNextHighestDepth());
   var w:Number = 150;
   var h:Number = 150;
   art.beginFill(0xCCCCCC);
   art.lineTo(w, 0);
   art.lineTo(w, h);
   art.lineTo(0, h);
   art.lineTo(0, 0);

   var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
   var alphas:Array = [1, 0, 1];
   var ratios:Array = [0, 128, 255];
   var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
   alphas, ratios, 5, 5, 5, 2, "inner", false);

   art.filters = new Array(filter);
   return art;
}
```

関連項目

colors (GradientBevelFilter.colors プロパティ),
ratios (GradientBevelFilter.ratios プロパティ)

angle (GradientBevelFilter.angle プロパティ)

```
public angle : Number
```

角度 (度数)。指定できる値は 0 ～ 360 で、デフォルトは 45 です。

角度の値は、オブジェクトに向けられる架空の光源の角度を表します。この値により、グラデーションの色がオブジェクトに適用される角度、つまり、ハイライトとシャドウの表示場所や、配列の先頭の色の表示場所が決定します。先頭の色を適用する場所が決まると、他の色が配列内に格納されている順に適用されます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例は、既存のオブジェクトで angle プロパティを設定する方法を示しています。

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("angleExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].angle = 45;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

関連項目

ratios (GradientBevelFilter.ratios プロパティ)

blurX (GradientBevelFilter.blurX プロパティ)

```
public blurX : Number
```

水平方向のぼかし量。指定できる値は 0 ～ 255 です。1 以下の値を指定すると、元のイメージがそのままコピーされます。デフォルト値は 4 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例は、既存のオブジェクトで blurX プロパティを設定する方法を示しています。

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("blurXExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].blurX = 16;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

blurY (GradientBevelFilter.blurY プロパティ)

```
public blurY : Number
```

垂直方向のぼかし量。指定できる値は 0 ～ 255 です。1 以下の値を指定すると、元のイメージがそのままコピーされます。デフォルト値は 4 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例は、既存のオブジェクトで blurY プロパティを設定する方法を示しています。

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("blurYExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].blurY = 16;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

clone (GradientBevelFilter.clone メソッド)

```
public clone() : GradientBevelFilter
```

このフィルタオブジェクトのコピーを返します。

対応バージョン： ActionScript 1.0、Flash Player 8

戻り値

`GradientBevelFilter` - 元の GradientBevelFilter インスタンスとプロパティがすべて同じである新しい GradientBevelFilter インスタンス。

例

次の例では、2 つの矩形シェイプを作成します。最初のシェイプ `sourceClip` にはベベル効果が適用されています。2 番目のシェイプ `resultClip` は、クリックされるまで効果が適用されません。

```
import flash.filters.GradientBevelFilter;

var sourceClip:MovieClip = setUpFlatRectangle(150, 150, 0xCCCCCC,
  "cloneSourceClip");
var resultClip:MovieClip = setUpFlatRectangle(150, 150, 0xCCCCCC,
  "cloneResultClip");

resultClip.source = sourceClip;

var sourceFilter:GradientBevelFilter = getNewFilter();
sourceClip.filters = new Array(sourceFilter);

resultClip._x = 180;
resultClip.onRelease = function() {
  this.filters = new Array(this.source.filters[0].clone());
}

function setUpFlatRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  mc.beginFill(bgColor);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  return mc;
}

function getNewFilter():GradientBevelFilter {
  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  return new GradientBevelFilter(5, 225, colors, alphas, ratios, 5, 5, 5, 2,
  "inner", false);
}
```

colors (GradientBevelFilter.colors プロパティ)

public colors : Array

グラデーションで使用する RGB 16 進数カラー値の配列。たとえば、赤は 0xFF0000、青は 0x0000FF などです。

colors プロパティの値を直接変更することはできません。このプロパティを変更するには、colors への参照を取得し、その参照を変更した後、colors をその参照に設定する必要があります。

colors、alphas、ratios の 3 つのプロパティはすべて関係しています。colors 配列の先頭のエレメントが alphas 配列と ratios 配列の先頭のエレメントに対応するなど、3 つの配列の同じインデックスのエレメントが互いに対応しています。

対応バージョン：ActionScript 1.0、Flash Player 8

例

次の例は、既存のエンティティで colors プロパティを設定する方法を示しています。

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("colorsExample");
mc.onPress = function() {
  var arr:Array = this.filters;
  var colors:Array = [0x000000, 0xCCCCCC, 0xFFFFFF];
  arr[0].colors = colors;
  this.filters = arr;
}
mc.onRelease = function() {
  var arr:Array = this.filters;
  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  arr[0].colors = colors;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
```

```
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 2, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

関連項目

alphas (GradientBevelFilter.alphas プロパティ),
ratios (GradientBevelFilter.ratios プロパティ)

distance (GradientBevelFilter.distance プロパティ)

public distance : Number

オフセット距離。デフォルト値は 4 です。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例は、既存のオブジェクトで distance プロパティを設定する方法を示しています。

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("distanceExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].distance = 1;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

GradientBevelFilter コンストラクタ

```
public GradientBevelFilter([distance:Number], [angle:Number], [colors:Array],
  [alphas:Array], [ratios:Array], [blurX:Number], [blurY:Number],
  [strength:Number], [quality:Number], [type:String], [knockout:Boolean])
```

指定されたパラメータでフィルタを初期化します。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

`distance:Number` (オプション) - オフセット距離。有効な値は 0 ～ 8 で、デフォルト値は 4 です。

`angle:Number` (オプション) - 角度 (度数)。指定できる値は 0 ～ 360 で、デフォルトは 45 です。

`colors:Array` (オプション) - グラデーションで使用する RGB 16 進数カラー値の配列。たとえば、赤は 0xFF0000、青は 0x0000FF などです。

`alphas:Array` (オプション) - `colors` 配列の色に対応するアルファ透明度の値の配列。配列内の各エレメントに指定できる値は 0 ～ 1 です。たとえば .25 と指定すると、透明度の値は 25% に設定されます。

`ratios:Array` (オプション) - 色分布比率の配列。0 ～ 255 の範囲の値を指定できます。

`blurX:Number` (オプション) - 水平方向のぼかし量です。指定できる値は 0 ～ 255 です。1 以下の値を指定すると、元のイメージがそのままコピーされます。デフォルト値は 4 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

`blurY:Number` (オプション) - 垂直方向のぼかし量です。指定できる値は 0 ～ 255 です。1 以下の値を指定すると、元のイメージがそのままコピーされます。デフォルト値は 4 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

`strength:Number` (オプション) - インプリントやスプレッドの長さです。値が大きいほど、濃い色がインプリントされるので、ベベルと背景との間のコントラストが強くなります。指定できる値は 0 ～ 255 です。0 を指定すると、フィルタは適用されません。デフォルト値は 1 です。

`quality:Number` (オプション) - フィルタの品質。0 ～ 15 の範囲の値を指定できます。デフォルト値は 1 です。ほとんどすべての場合に、1 (低品質)、2 (普通の品質)、または 3 (高品質) で十分です。フィルタに設定された値が小さいほど、速くレンダリングできます。

`type:String` (オプション) - ベベル効果の配置です。有効な値は次のとおりです。

- `"outer"` : ベベルがオブジェクトの外側エッジに配置されます。
- `"inner"` : ベベルがオブジェクトの内側エッジに配置されます。
- `"full"` : ベベルがオブジェクトの上に配置されます。

デフォルト値は `"inner"` です。

knockout:Boolean (オプション) - ノックアウト効果を適用するかどうかを指定します。true を指定すると、オブジェクトの塗りが透明になり、ドキュメントの背景色が表示されます。デフォルトは false (ノックアウトなし) です。

例

次の例では、GradientBevelFilter の新しいインスタンスを作成して値を割り当て、そのインスタンスをフラットな矩形イメージに適用します。

```
import flash.filters.GradientBevelFilter;
import flash.filters.BitmapFilter;
var art:MovieClip = setUpFlatRectangle(150, 150, 0xCCCCCC,
  "gradientBevelFilterExample");
var distance:Number = 5;
var angleInDegrees:Number = 225; // opposite 45 degrees
var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
var alphas:Array = [1, 0, 1];
var ratios:Array = [0, 128, 255];
var blurX:Number = 8;
var blurY:Number = 8;
var strength:Number = 2;
var quality:Number = 3;
var type:String = "inner";
var knockout:Boolean = true;

var filter:GradientBevelFilter = new GradientBevelFilter(distance,
                                 angleInDegrees,
                                 colors,
                                 alphas,
                                 ratios,
                                 blurX,
                                 blurY,
                                 strength,
                                 quality,
                                 type,
                                 knockout);
var filterArray:Array = new Array();
filterArray.push(filter);
art.filters = filterArray;

function setUpFlatRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  mc.beginFill(bgColor);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  return mc;
}
```

関連項目

ratios (GradientBevelFilter.ratios プロパティ)

knockout (GradientBevelFilter.knockout プロパティ)

public knockout : Boolean

オブジェクトにノックアウト効果を適用するかどうかを指定します。ノックアウト効果を適用すると、オブジェクトの塗りが透明になり、ドキュメントの背景色が表示されます。true を指定すると、ノックアウト効果が適用されます。デフォルトは false で、ノックアウト効果は適用されません。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例は、既存のオブジェクトで knockout プロパティを設定する方法を示しています。

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("knockoutExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].knockout = true;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

quality (GradientBevelFilter.quality プロパティ)

```
public quality : Number
```

フィルタを適用する回数。有効な値は 0 ～ 15 です。デフォルト値は 1 で、低品質と等価です。値 2 は標準の品質であり、値 3 は高品質です。フィルタに設定された値が小さいほど、速くレンダリングできます。

多くのアプリケーションでは、quality 値 1、2、または 3 で十分です。最大 15 までの値を使用してさまざまな効果を出すことができますが、値が大きくなるほどレンダリング速度が低下します。多くの場合、quality の値を大きくする代わりに blurX と blurY の値を大きくするだけで、同様の効果が得られます。この方法を実行すると、より高速にレンダリングされます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例は、既存のオブジェクトで quality プロパティを設定する方法を示しています。

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("qualityExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].quality = 1; // low quality
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

関連項目

ratios (GradientBevelFilter.ratios プロパティ)

ratios (GradientBevelFilter.ratios プロパティ)

public ratios : Array

colors 配列内の対応する色の色分布比率の配列。配列のエレメントに指定できる値は、0 ～ 255 です。

ratios プロパティの値を直接変更することはできません。このプロパティを変更するには、ratios への参照を取得し、その参照を変更した後、ratios をその参照に設定する必要があります。

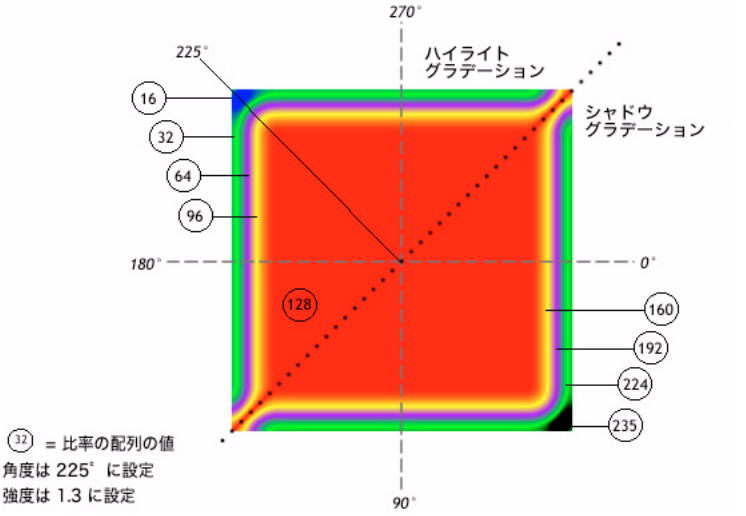
colors、alphas、ratios の 3 つのプロパティはすべて関係しています。colors 配列の先頭のエレメントが alphas 配列と ratios 配列の先頭のエレメントに対応するなど、3 つの配列の同じインデックスのエレメントが互いに対応しています。

グラデーションベベルの色をどのよう分布するかを理解するために、まず、グラデーションベベルで使用する色を考えます。ここでは、ハイライト 1 色とシャドウ 1 色から構成される単純なベベルを考えます。グラデーションベベルには 1 つのハイライトグレデーションと 1 つのシャドウグラデーションがあります。ハイライトを左上隅に表示し、シャドウを右下隅に表示するとします。ここでは、一例として、ハイライトに 4 色、シャドウに 4 色を使用します。ハイライトとシャドウに加え、ハイライトとシャドウのエッジが交差する部分にベース塗りの色を使用します。この場合、合計 9 色を使用するので、色分布比率の配列のエレメント数は 9 です。

グラデーションは互いに混ざり合ったさまざまな色のストライプで構成されていると考えた場合、色分布比率の値は、その色のグラデーションの範囲内における位置を表します。この位置は、0 がグラデーションの最も外側の点を表し、255 がグラデーションの最も内側の点を表します。通常、中間値は 128 で、これがベース塗りの値になります。下のイメージのベベル効果を得るには、9 色の例を使用して色分布比率の値を次のように設定します。

- 最初の 4 色は 0 ～ 127 の範囲で、それぞれの値が前の値と等しいか、それより大きくなるように設定します。これがハイライトのベベルエッジです。
- 5 番目の色 (中間色) はベース塗りで、128 に設定します。ピクセル値の 128 によりベース塗りが設定されます。このベース塗りは、type が outer に設定されている場合はシェイプの外側 (およびベベルエッジの周囲) に現れ、type が inner に設定されている場合はシェイプの内側に現れてオブジェクト自体の塗りを効果的に隠します。
- 最後の 4 色は 129 ～ 255 の範囲で、それぞれの値が前の値と等しいか、それより大きくなるように設定します。これがシャドウのベベルエッジです。

各エッジの色分布を同じにする場合は、中間色がベース塗りになるように、使用する色数を奇数にします。0 ～ 127 と 129 ～ 255 の範囲で色を均等に分布した後、値を調整してグラデーションの各色のストライプの幅を変更します。9 色のグラデーションベベルの場合、[16, 32, 64, 96, 128, 160, 192, 224, 235] のような配列が考えられます。次の図は、ここまで説明したグラデーションベベルを表しています。



グラデーションの色の広がりは、blurX、blurY、strength、quality プロパティの値、および ratios 値に基づいて変化します。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例は、既存のエンティティで ratios プロパティを設定する方法を示しています。

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("ratiosExample");
mc.onPress = function() {
  var arr:Array = this.filters;
  var ratios:Array = [127, 128, 129];
  arr[0].ratios = ratios;
  this.filters = arr;
}
mc.onRelease = function() {
  var arr:Array = this.filters;
  var ratios:Array = [0, 128, 255];
  arr[0].ratios = ratios;
  this.filters = arr;
}
```

```
function setUpFilter(name:String):MovieClip {
   var art:MovieClip = this.createEmptyMovieClip(name,
   this.getNextHighestDepth());
   var w:Number = 150;
   var h:Number = 150;
   art.beginFill(0xCCCCCC);
   art.lineTo(w, 0);
   art.lineTo(w, h);
   art.lineTo(0, h);
   art.lineTo(0, 0);

   var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
   var alphas:Array = [1, 0, 1];
   var ratios:Array = [0, 128, 255];
   var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
   alphas, ratios, 5, 5, 5, 2, "inner", false);

   art.filters = new Array(filter);
   return art;
}
```

関連項目

alphas (GradientBevelFilter.alphas プロパティ),colors (GradientBevelFilter.colors プロパティ),beginGradientFill (MovieClip.beginGradientFill メソッド)

strength (GradientBevelFilter.strength プロパティ)

```
public strength : Number
```

インプリントの強さまたは広がり。値が大きいほど、濃い色がインプリントされるので、ベベルと背景との間のコントラストが強くなります。指定できる値は 0 ～ 255 です。0 を指定すると、フィルタは適用されません。デフォルト値は 1 です。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例は、既存のオブジェクトで strength プロパティを設定する方法を示しています。

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("strengthExample");
mc.onRelease = function() {
   var arr:Array = this.filters;
   arr[0].strength = 1;
   this.filters = arr;
}
```

```
function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

関連項目

ratios (GradientBevelFilter.ratios プロパティ)

type (GradientBevelFilter.type プロパティ)

public type : String

ベベル効果の配置。有効な値は次のとおりです。

- "outer" : ベベルがオブジェクトの外側エッジに配置されます。
- "inner" : ベベルがオブジェクトの内側エッジに配置されます。
- "full" : ベベルがオブジェクトの上に配置されます。

デフォルト値は "inner" です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例は、既存のオブジェクトで type プロパティを設定する方法を示しています。

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("typeExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].type = "outer";
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

GradientGlowFilter (flash.filters.GradientGlowFilter)

```
Object
  |
  +-BitmapFilter
     |
     +-flash.filters.GradientGlowFilter
```

```
public class GradientGlowFilter
extends BitmapFilter
```

GradientGlowFilter クラスを使用すると、グラデーショングロー効果を Flash の各種オブジェクトに適用できます。グラデーショングローは、制御可能なカラーグラデーションを持ったリアルな輝きです。グラデーショングローは、オブジェクトの内側エッジや外側エッジの周囲、またはオブジェクトの上に適用できます。

フィルタの使用方法は、フィルタの適用先オブジェクトによって異なります。

- 実行時にムービークリップ、テキストフィールド、ボタンにフィルタを適用する場合は、filters プロパティを使用します。オブジェクトの filters プロパティを設定しても、オブジェクトは変更されません。filters プロパティをクリアすることにより、フィルタを取り消すことができます。
- BitmapData インスタンスにフィルタを適用するには、BitmapData.applyFilter() メソッドを使用します。BitmapData オブジェクトで applyFilter() を呼び出すことによって、ソース BitmapData オブジェクトとフィルタオブジェクトが取得され、フィルタを適用した結果として得られるイメージが生成されます。

イメージやビデオにもオーサリング時にフィルタ効果を適用できます。詳細については、オーサリングのマニュアルを参照してください。

ムービークリップやボタンにフィルタを適用する場合は、ムービークリップやボタンの cacheAsBitmap プロパティを true に設定します。すべてのフィルタをクリアすると、cacheAsBitmap は元の値に戻ります。

このフィルタはステージの拡大・縮小に対応していますが、通常の拡大・縮小、回転、傾斜には対応していません。オブジェクト自体を拡大・縮小する場合 (_xscale と _yscale が 100% でない場合)、フィルタ効果は拡大・縮小されません。フィルタ効果が拡大・縮小するのは、ステージをズームする場合のみです。

結果として得られるイメージの幅または高さが 2880 ピクセルを超える場合、フィルタは適用されません。たとえば、フィルタが適用されたサイズの大きいムービークリップをズームインするとき、結果として得られるイメージが 2880 ピクセルの制限を超える場合は、フィルタがオフになります。

対応バージョン： ActionScript 1.0、Flash Player 8

関連項目

ratios (GradientGlowFilter.ratios プロパティ), applyFilter (BitmapData.applyFilter メソッド), cacheAsBitmap (Button.cacheAsBitmap プロパティ), filters (Button.filters プロパティ), GlowFilter (flash.filters.GlowFilter), cacheAsBitmap (MovieClip.cacheAsBitmap プロパティ), filters (MovieClip.filters プロパティ), filters (TextField.filters プロパティ)

プロパティ一覧

オプション	プロパティ	説明
	alphas:Array	colors 配列内の各色に対応するアルファ透明度の値の配列。
	angle:Number	角度 (度数)。
	blurX:Number	水平方向のぼかし量。
	blurY:Number	垂直方向のぼかし量。
	colors:Array	グラデーションを定義する色の配列。
	distance:Number	グローのオフセット。
	knockout:Boolean	オブジェクトにノックアウト効果を適用するかどうかを指定します。
	quality:Number	フィルタを適用する回数。
	ratios:Array	colors 配列内の対応する色の色分布比率の配列。
	strength:Number	インプリントやスプレッドの長さ。
	type:String	フィルタ効果の配置。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター一覧

署名	説明
GradientGlowFilter ([distance:Number], [angle:Number], [colors:Array], [alphas:Array], [ratios:Array], [blurX:Number], [blurY:Number], [strength:Number], [quality:Number], [type:String], [knockout:Boolean])	指定されたパラメータでフィルタを初期化します。

メソッド一覧

オプション	署名	説明
	clone() : GradientGlowFilter	このフィルタオブジェクトのコピーを返します。

BitmapFilter クラスから継承されるメソッド

clone (BitmapFilter.clone メソッド)

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

alphas (GradientGlowFilter.alphas プロパティ)

```
public alphas : Array
```

colors 配列内の各色に対応するアルファ透明度の値の配列。配列内の各エレメントに指定できる値は 0 ～ 1 です。たとえば .25 と指定すると、アルファ透明度は 25 パーセントとなります。

alphas プロパティの値を直接変更することはできません。このプロパティを変更するには、alphas への参照を取得し、その参照を変更した後、alphas をその参照に設定する必要があります。

colors、alphas、ratios の 3 つのプロパティはすべて関係しています。colors 配列の先頭のエレメントが alphas 配列と ratios 配列の先頭のエレメントに対応するなど、3 つの配列の同じインデックスのエレメントが互いに対応しています。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その alphas プロパティを変更します。

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowAlphas");
mc.onRelease = function() {
   var filter:GradientGlowFilter = this.filters[0];
   var alphas:Array = filter.alphas;
   alphas.pop();
   alphas.pop();
   alphas.push(.3);
   alphas.push(1);
   filter.alphas = alphas;
   this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
   var art:MovieClip = this.createEmptyMovieClip(name,
   this.getNextHighestDepth());
   var w:Number = 100;
   var h:Number = 100;
   art.beginFill(0x003366);
   art.lineTo(w, 0);
   art.lineTo(w, h);
   art.lineTo(0, h);
   art.lineTo(0, 0);
   art._x = 20;
   art._y = 20;

   var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
   var alphas:Array = [0, 1, 1, 1];
   var ratios:Array = [0, 63, 126, 255];
   var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors, alphas,
   ratios, 55, 55, 2.5, 2, "outer", false);
```

```
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

関連項目

colors (GradientGlowFilter.colors プロパティ),
ratios (GradientGlowFilter.ratios プロパティ)

angle (GradientGlowFilter.angle プロパティ)

public angle : Number

角度 (度数)。指定できる値は 0 ～ 360 で、デフォルトは 45 です。

角度の値は、オブジェクトに対する架空の光源の角度を表し、オブジェクトに対する効果の相対位置を決定します。distance が 0 に設定されている場合、効果がオブジェクトからオフセットされないので、angle プロパティはオブジェクトに影響しません。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その angle プロパティを変更します。

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowAngle");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.distance = 50;
  filter.angle = 90;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;
```

```
  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors, alphas,
  ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

blurX (GradientGlowFilter.blurX プロパティ)

```
public blurX : Number
```

水平方向のぼかし量。指定できる値は 0 ～ 255 です。1 以下の値を指定すると、元のイメージがそのままコピーされます。デフォルト値は 4 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その blurX プロパティを変更します。

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowBlurX");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.blurX = 255;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;
```

```
  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors, alphas,
  ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

blurY (GradientGlowFilter.blurY プロパティ)

```
public blurY : Number
```

垂直方向のぼかし量。指定できる値は 0 ～ 255 です。1 以下の値を指定すると、元のイメージがそのままコピーされます。デフォルト値は 4 です。2 のべき乗 (2、4、8、16、32 など) は、他の値と比べて速くレンダリングできるよう最適化されます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その blurY プロパティを変更します。

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowBlurY");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.blurY = 255;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;
```

```
  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors, alphas,
  ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

clone (GradientGlowFilter.clone メソッド)

```
public clone() : GradientGlowFilter
```

このフィルタオブジェクトのコピーを返します。

対応バージョン : ActionScript 1.0、Flash Player 8

戻り値

`GradientGlowFilter` - 元の GradientGlowFilter インスタンスとプロパティがすべて同じである新しい GradientGlowFilter インスタンス。

例

次の例では、3 つの GradientGlowFilter オブジェクトを作成して比較します。`filter_1` は、GradientGlowFilter コンストラクタを使用して作成されます。`filter_2` は、`filter_1` と等しい値に設定することにより作成されます。`clonedFilter` は、`filter_1` のクローンを作成することにより作成されます。`filter_2` が `filter_1` と等しいと評価された場合に、`filter_1` と同じ値を含んでいても `clonedFilter` は等しいと評価されないことに注意してください。

```
import flash.filters.GradientGlowFilter;

var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
var alphas:Array = [0, 1, 1, 1];
var ratios:Array = [0, 63, 126, 255];
var filter_1:GradientGlowFilter = new GradientGlowFilter(0, 45, colors, alphas,
  ratios, 55, 55, 2.5, 2, "outer", false);
var filter_2:GradientGlowFilter = filter_1;
var clonedFilter:GradientGlowFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false
```

```
for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> type: outer
  // >> knockout: false
  // >> strength: 2.5
  // >> quality: 2
  // >> blurY: 55
  // >> blurX: 55
  // >> ratios: 0,63,126,255
  // >> alphas: 0,1,1,1
  // >> colors: 16777215,16711680,16776960,52479
  // >> angle: 45
  // >> distance: 0
}

for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
  // >> type: outer
  // >> knockout: false
  // >> strength: 2.5
  // >> quality: 2
  // >> blurY: 55
  // >> blurX: 55
  // >> ratios: 0,63,126,255
  // >> alphas: 0,1,1,1
  // >> colors: 16777215,16711680,16776960,52479
  // >> angle: 45
  // >> distance: 0
}
```

filter_1、filter_2、および clonedFilter の関係をさらに詳しく示すために、次の例では、filter_1 の knockout プロパティを変更します。knockout を変更することは、clone() メソッドによって、filter_1 を参照する代わりにその値に基づいて新しいインスタンスが作成されることを示します。

```
import flash.filters.GradientGlowFilter;

var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
var alphas:Array = [0, 1, 1, 1];
var ratios:Array = [0, 63, 126, 255];
var filter_1:GradientGlowFilter = new GradientGlowFilter(0, 45, colors, alphas,
  ratios, 55, 55, 2.5, 2, "outer", false);
var filter_2:GradientGlowFilter = filter_1;
var clonedFilter:GradientGlowFilter = filter_1.clone();

trace(filter_1.knockout); // false
trace(filter_2.knockout); // false
trace(clonedFilter.knockout); // false
```

```
filter_1.knockout = true;

trace(filter_1.knockout); // true
trace(filter_2.knockout); // true
trace(clonedFilter.knockout); // false
```

colors (GradientGlowFilter.colors プロパティ)

```
public colors : Array
```

グラデーションを定義する色の配列。たとえば、赤は 0xFF0000、青は 0x0000FF などです。

colors プロパティの値を直接変更することはできません。このプロパティを変更するには、colors への参照を取得し、その参照を変更した後、colors をその参照に設定する必要があります。

colors、alphas、ratios の 3 つのプロパティはすべて関係しています。colors 配列の先頭のエレメントが alphas 配列と ratios 配列の先頭のエレメントに対応するなど、3 つの配列の同じインデックスのエレメントが互いに対応しています。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その colors プロパティを変更します。

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowColors");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  var colors:Array = filter.colors;
  colors.pop();
  colors.push(0xFF00FF);
  filter.colors = colors;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;
```

```
  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors, alphas,
  ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

関連項目

alphas (GradientGlowFilter.alphas プロパティ),
ratios (GradientGlowFilter.ratios プロパティ)

distance (GradientGlowFilter.distance プロパティ)

public distance : Number

グローのオフセット。デフォルト値は 4 です。

対応バージョン: ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その distance プロパティを変更します。

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowDistance");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.distance = 20;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;
```

```
  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors, alphas,
  ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

GradientGlowFilter コンストラクタ

```
public GradientGlowFilter([distance:Number], [angle:Number], [colors:Array],
  [alphas:Array], [ratios:Array], [blurX:Number], [blurY:Number],
  [strength:Number], [quality:Number], [type:String], [knockout:Boolean])
```

指定されたパラメータでフィルタを初期化します。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

`distance:Number`（オプション）- グローのオフセット。デフォルトは 4 です。

`angle:Number`（オプション）- 角度（度数）。指定できる値は 0 ～ 360 で、デフォルトは 45 です。

`colors:Array`（オプション）- グラデーションを定義する色の配列。たとえば、赤は 0xFF0000、青は 0x0000FF などです。

`alphas:Array`（オプション）- `colors` 配列の色に対応するアルファ透明度の値の配列。配列内の各エレメントに指定できる値は 0 ～ 1 です。たとえば .25 と指定すると、アルファ透明度は 25 パーセントとなります。

`ratios:Array`（オプション）- 色分布比率の配列。指定できる値は 0 ～ 255 です。この値は、100% でサンプリングされる色の幅の割合をパーセントで定義します。

`blurX:Number`（オプション）- 水平方向のぼかし量です。指定できる値は 0 ～ 255 です。1 以下の値を指定すると、元のイメージがそのままコピーされます。デフォルト値は 4 です。2 のべき乗（2、4、8、16、32 など）は、他の値と比べて速くレンダリングできるよう最適化されます。

`blurY:Number`（オプション）- 垂直方向のぼかし量です。指定できる値は 0 ～ 255 です。1 以下の値を指定すると、元のイメージがそのままコピーされます。デフォルト値は 4 です。2 のべき乗（2、4、8、16、32 など）は、他の値と比べて速くレンダリングできるよう最適化されます。

`strength:Number`（オプション）- インプリントやスプレッドの長さです。値が大きいほど、濃い色がインプリントされるので、グローと背景との間のコントラストが強くなります。0 ～ 255 の範囲の値を指定できます。値が大きいほど、インプリントが濃くなります。値を 0 にすると、フィルタが適用されなくなります。デフォルト値は 1 です。

quality:Number (オプション) - フィルタを適用する回数。有効な値は 0 ～ 15 です。デフォルト値は 1 で、低品質と等価です。値 2 は標準の品質であり、値 3 は高品質です。

type:String (オプション) - フィルタ効果の配置です。有効な値は次のとおりです。

- "outer" : グローがオブジェクトの外側エッジに配置されます。
- "inner" : グローがオブジェクトの内側エッジに配置されます。これがデフォルトです。
- "full" : グローがオブジェクトの上に配置されます。

デフォルト値は "inner" です。

knockout:Boolean (オプション) - オブジェクトにノックアウト効果を適用するかどうかを指定します。ノックアウト効果を適用すると、オブジェクトの塗りが透明になり、ドキュメントの背景色が表示されます。true を指定すると、ノックアウト効果が適用されます。デフォルトは false で、ノックアウト効果は適用されません。

例

次の例では、グラデーショングローフィルタの新しいインスタンスを作成して値を割り当て、そのインスタンスをフラットな矩形イメージに適用します。

```
import flash.filters.GradientGlowFilter;
var art:MovieClip = createRectangle(100, 100, 0x003366,
  "gradientGlowFilterExample");
var distance:Number = 0;
var angleInDegrees:Number = 45;
var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
var alphas:Array = [0, 1, 1, 1];
var ratios:Array = [0, 63, 126, 255];
var blurX:Number = 50;
var blurY:Number = 50;
var strength:Number = 2.5;
var quality:Number = 3;
var type:String = "outer";
var knockout:Boolean = false;

var filter:GradientGlowFilter = new GradientGlowFilter(distance,
                                angleInDegrees,
                                colors,
                                alphas,
                                ratios,
                                blurX,
                                blurY,
                                strength,
                                quality,
                                type,
                                knockout);
var filterArray:Array = new Array();
filterArray.push(filter);
art.filters = filterArray;
```

```
function createRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  mc.beginFill(bgColor);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc._x = 20;
  mc._y = 20;
  return mc;
}
```

knockout (GradientGlowFilter.knockout プロパティ)

```
public knockout : Boolean
```

オブジェクトにノックアウト効果を適用するかどうかを指定します。ノックアウト効果を適用すると、オブジェクトの塗りが透明になり、ドキュメントの背景色が表示されます。true を指定すると、ノックアウト効果が適用されます。デフォルトは false で、ノックアウト効果は適用されません。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その knockout プロパティを変更します。

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowKnockout");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.knockout = true;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;
```

```
  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors, alphas,
  ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

quality (GradientGlowFilter.quality プロパティ)

```
public quality : Number
```

フィルタを適用する回数。有効な値は 0 ～ 15 です。デフォルト値は 1 で、低品質と等価です。値 2 は標準の品質であり、値 3 は高品質です。フィルタに設定された値が小さいほど、速くレンダリングできます。

多くのアプリケーションでは、quality 値 1、2、または 3 で十分です。最大 15 までの値を使用してさまざまな効果を出すことができますが、値が大きくなるほどレンダリング速度が低下します。多くの場合、quality の値を大きくする代わりに blurX と blurY の値を大きくするだけで、同様の効果が得られます。この方法を実行すると、より高速にレンダリングされます。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その quality プロパティを変更します。

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowQuality");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.quality = 3;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
```

```
    art._x = 20;
    art._y = 20;

    var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
    var alphas:Array = [0, 1, 1, 1];
    var ratios:Array = [0, 63, 126, 255];
    var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors, alphas,
    ratios, 55, 55, 2.5, 2, "outer", false);
    var filterArray:Array = new Array();
    filterArray.push(filter);
    art.filters = filterArray;
    return art;
}
```

ratios (GradientGlowFilter.ratios プロパティ)

```
public ratios : Array
```

colors 配列内の対応する色の色分布比率の配列。指定できる値は 0 ～ 255 です。

ratios プロパティの値を直接変更することはできません。このプロパティを変更するには、ratios への参照を取得し、その参照を変更した後、ratios をその参照に設定する必要があります。

colors、alphas、ratios の 3 つのプロパティはすべて関係しています。colors 配列の先頭のエレメントが alphas 配列と ratios 配列の先頭のエレメントに対応するなど、3 つの配列の同じインデックスのエレメントが互いに対応しています。

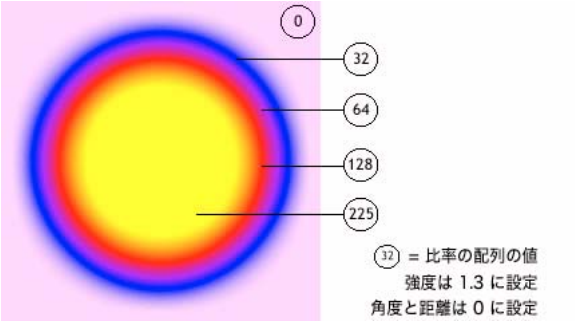
グラデーショングローフィルタは、オブジェクトの中心 (distance が 0 に設定されている場合) から発せられる輝きであり、互いに混ざり合った色のストライプから構成されるグラデーションを持っていると考えます。colors 配列の先頭の色がグローの最も外側の色です。最後の色がグローの最も内側の色です。

ratios 配列の各値は、その色のグラデーションの範囲における位置を表します。0 はグラデーションの最も外側の点を表し、255 はグラデーションの最も内側の点を表します。色分布比率の値は、0 ～ 255 ピクセルの範囲で指定できます。たとえば、[0, 64, 128, 200, 255] のように値を増やしていきます。0 ～ 128 の値は、グローの外側のエッジに表示されます。129 ～ 255 の値は、グローの内側の領域に表示されます。色の分布比率の値とフィルタの type の値によっては、フィルタの適用先のオブジェクトによってフィルタの色が見えなくなることがあります。

次のコードとイメージでは、フィルタを黒い円のムービークリップに適用し、type を "full" に設定しています。説明のため、colors 配列の先頭の色であるピンクの alpha 値を 1 にして、ドキュメントの白い背景の中で目立つようにしていますが、実際に先頭の色をこのように表示することはおそらくありません。配列の最後の色である黄色により、フィルタの適用先の黒い円が見えなくなっています。

```
var colors = [0xFFCCFF, 0x0000FF, 0x9900FF, 0xFF0000, 0xFFFF00];
 var alphas = [1, 1, 1, 1, 1];
 var ratios = [0, 32, 64, 128, 225];
```

```
var myGGF = new GradientGlowFilter(0, 0, colors, alphas, ratios, 50, 50, 1, 2,
  "full", false);
```



type を "outer" または "full" に設定したときに、ドキュメントの背景と一体になった効果を得るには、配列の先頭の色をドキュメントの背景と同じに色に設定するか、先頭の色のアルファ値を 0 に設定します。どちらの方法を使用しても、フィルタが背景に溶け込みます。

コードに小さな変更を 2 つ加えると、同じ ratios 配列と colors 配列でもグローの効果が大きく異なります。配列の先頭の色のアルファ値を 0 に設定してフィルタがドキュメントの白い背景に溶け込むようにし、type プロパティを "outer" または "inner" に設定します。次のイメージで結果を比較してください。


外側グロー


内側グロー

グラデーションの色の広がりは、blurX、blurY、strength、quality プロパティの値、および ratios 値に基づいて変化します。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その ratios プロパティを変更します。

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowRatios");
mc.onRelease = function() {
   var filter:GradientGlowFilter = this.filters[0];
   var ratios:Array = filter.ratios;
   ratios.shift();
   ratios.unshift(40);
   filter.ratios = ratios;
   this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
   var art:MovieClip = this.createEmptyMovieClip(name,
   this.getNextHighestDepth());
   var w:Number = 100;
   var h:Number = 100;
   art.beginFill(0x003366);
   art.lineTo(w, 0);
   art.lineTo(w, h);
   art.lineTo(0, h);
   art.lineTo(0, 0);
   art._x = 20;
   art._y = 20;

   var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
   var alphas:Array = [0, 1, 1, 1];
   var ratios:Array = [0, 63, 126, 255];
   var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors, alphas,
   ratios, 55, 55, 2.5, 2, "outer", false);
   var filterArray:Array = new Array();
   filterArray.push(filter);
   art.filters = filterArray;
   return art;
}
```

関連項目

colors (GradientGlowFilter.colors プロパティ),alphas (GradientGlowFilter.alphas プロパティ),beginGradientFill (MovieClip.beginGradientFill メソッド)

strength (GradientGlowFilter.strength プロパティ)

```
public strength : Number
```

インプリントの強さまたは広がり。値が大きいほど、濃い色がインプリントされるので、グローと背景との間のコントラストが強くなります。指定できる値は 0 ～ 255 です。0 を指定すると、フィルタは適用されません。デフォルト値は 1 です。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その strength プロパティを変更します。

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowStrength");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.strength = 1;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors, alphas,
  ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

type (GradientGlowFilter.type プロパティ)

```
public type : String
```

フィルタ効果の配置。有効な値は次のとおりです。

- "outer" : グローがオブジェクトの外側エッジに配置されます。
- "inner" : グローがオブジェクトの内側エッジに配置されます。これがデフォルトです。
- "full" : グローがオブジェクトの上に配置されます。

デフォルト値は "inner" です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、既存のムービークリップがクリックされたときに、その type プロパティを変更します。

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowType");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.type = "inner";
  filter.strength = 1;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors, alphas,
  ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

IME (System.IME)

```
Object
  |
  +-System.IME

public class IME
extends Object
```

IME クラスを使用すると、クライアントコンピュータで実行されている Flash Player アプリケーションで、オペレーティングシステムの IME (Input Method Editor : 入力方式エディタ) を直接操作できます。IME がインストールされているかどうか、IME が有効になっているかどうか、およびどの IME が有効になっているかを調べることができます。Flash Player アプリケーションで IME を無効または有効にすることができます。また、オペレーティングシステムによっては、その他の限られた機能を使用できます。

IME を使用すると、中国語、日本語、韓国語といったアジアの言語で ASCII 以外の文字を入力できるようになります。IME の詳細については、開発対象であるアプリケーションのプラットフォームとして使用するオペレーティングシステムのマニュアルを参照してください。入力方式に関する他の情報源を次に示します。

- http://www.microsoft.com/globaldev/default.mspx
- http://developer.apple.com/documentation/
- http://java.sun.com

Macintosh の場合、IME を有効にしておかないと、IME アプリケーションプログラミングインターフェイス (API) の呼び出しが失敗します。IME を手動で有効にすると、それ以降の呼び出しは、意図したとおりに動作します。たとえば、システムで日本語 IME を使用する場合、使用可能な 4 つのモード (ひらがな、全角カタカナ、全角英数字、半角カタカナ) のうちのいずれかのモードで IME を有効にしておいてから、この API を呼び出す必要があります。

次の表に、このクラスのプラットフォーム別の対応状況を示します。

機能	Windows	Macintosh OS X	Macintosh Classic	Linux / Solaris XIM
IME がインストールされているかどうかの確認 `System.capabilities.hasIME`	○	○	○	○
IME のオン / オフ `System.IME.setEnabled()`	○	○	○ ***	○
IME のオン / オフの確認 `System.IME.getEnabled()`	○	○	○	○
IME 変換モードの設定 `System.IME.setConversionMode()`	○	○ **	X	○
IME 変換モードの取得 `System.IME.getConversionMode()`	○	○ **	X	○
変換対象ストリングの IME への送信 `System.IME.setCompositionString()`	○ *	X	X	○

IME からの変換前のストリングの取得
`System.IME.addListener()`
`listener.onIMEComposition()`
`System.IME.removeListener()`
	○ *	X	X	○

IME への変換要求の送信 `System.IME.doConversion()`	○ *	X	X	○

* 一部の Windows IME では、上記のうち一部サポートしていない操作があります。現時点において、すべての操作をサポートしている IME は日本語 IME のみです。オペレーティングシステムコールのサポートは、IME ごとに異なります。

** Macintosh の場合、これらのメソッドは日本語に対してのみサポートされており、サードパーティ製の IME ではサポートされていません。

*** この API を使用する前に、いずれかの日本語モード (ひらがな、全角および半角カタカナ、全角英数字) を有効にしておく必要があります。

対応バージョン： ActionScript 1.0、Flash Player 8

プロパティ一覧

オプション	プロパティ	説明
static	ALPHANUMERIC_FULL: String	"ALPHANUMERIC_FULL" という値のストリングで、setConversionMode() および getConversionMode() で使用します。
static	ALPHANUMERIC_HALF: String	"ALPHANUMERIC_HALF" という値のストリングで、setConversionMode() および getConversionMode() で使用します。
static	CHINESE:String	"CHINESE" という値のストリングで、setConversionMode() および getConversionMode() で使用します。
static	JAPANESE_HIRAGANA :String	"JAPANESE_HIRAGANA" という値のストリングで、setConversionMode() および getConversionMode() で使用します。
static	JAPANESE_KATAKANA_FULL: String	"JAPANESE_KATAKANA_FULL" という値のストリングで、setConversionMode() および getConversionMode() で使用します。
static	JAPANESE_KATAKANA_HALF: String	"JAPANESE_KATAKANA_HALF" という値のストリングで、setConversionMode() および getConversionMode() で使用します。
static	KOREAN:String	"KOREAN" という値のストリングで、setConversionMode() および getConversionMode() で使用します。
static	UNKNOWN:String	"UNKNOWN" という値のストリングで、getConversionMode() で使用します。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベントの一覧

イベント	説明
onIMEComposition = function ([readingString: String]) {}	IME への入力ストリングが設定されたときに通知されます。

メソッド一覧

オプション	署名	説明
static	addListener(listener: Object) : Void	onIMEComposition イベントによって IME イベントハンドラが呼び出されたときに通知を受け取るオブジェクトを登録します。
static	doConversion() : Boolean	現在の入力ストリングの第1候補を選択するように IME に指示します。
static	getConversionMode() : String	現在の IME の変換モードを示します。
static	getEnabled() : Boolean	システム IME が有効になっているかどうかを示します。
static	removeListener (listener:Object) : Boolean	IME.addListener() を使用して IME インスタンスに以前に登録したリスナーオブジェクトを削除します。
static	setCompositionString (composition:String) : Boolean	IME への入力ストリングを設定します。
static	setConversionMode (mode:String) : Boolean	現在の IME の変換モードを設定します。
static	setEnabled(enabled: Boolean) : Boolean	システム IME を有効または無効にします。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

addListener (IME.addListener メソッド)

```
public static addListener(listener:Object) : Void
```

onIMEComposition イベントによって IME イベントハンドラが呼び出されたときに通知を受け取るオブジェクトを登録します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

listener:Object - onIMEComposition (readingString) メソッドがあり、IME イベントハンドラからのコールバック通知を待機するオブジェクト。readingString でこのメソッドに渡される読み取りストリングは、IME の入力モードでのストリングです。たとえば、ユーザーがひらがなで入力してから漢字候補を選択する場合、読み取り文字列は元のひらがなになります。

例

次の例では、ユーザーがテキストフィールドをクリックして入力ストリングを設定したときに通知を出力するリスナーオブジェクトを System.IME に追加しています。

```
var IMEListener:Object = new Object();
IMEListener.onIMEComposition = function(str:String) {
  trace(">> onIMEComposition: " + str);
}
System.IME.addListener(IMEListener);
trace(System.IME.length);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.createTextField("txt", this.getNextHighestDepth(), 0, 0, 0, 0);
mc.txt.border = true;
mc.txt.background = true;
mc.txt.autoSize = "left";
mc.txt.text = "Click this text to add a listener.";

mc.onPress = function() {
  if(System.capabilities.hasIME) {
    Selection.setFocus(mc.txt);
    System.IME.setCompositionString(mc.txt.text);
  }
}
```

ALPHANUMERIC_FULL (IME.ALPHANUMERIC_FULL プロパティ)

public static ALPHANUMERIC_FULL : String

"ALPHANUMERIC_FULL" という値のストリングで、setConversionMode() および getConversionMode() で使用します。この定数は、すべての IME で使用します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、システムに IME (Input Method Editor : 入力方式エディタ) がインストールされている場合 (System.capabilities.hasIME) に、IME を ALPHANUMERIC_FULL に設定します。

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.ALPHANUMERIC_FULL);
  trace(System.IME.getConversionMode());
}
```

関連項目

setConversionMode (IME.setConversionMode メソッド), getConversionMode (IME.getConversionMode メソッド), hasIME (capabilities.hasIME プロパティ)

ALPHANUMERIC_HALF (IME.ALPHANUMERIC_HALF プロパティ)

public static ALPHANUMERIC_HALF : String

"ALPHANUMERIC_HALF" という値のストリングで、setConversionMode() および getConversionMode() で使用します。この定数は、すべての IME で使用します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、システムに IME (Input Method Editor : 入力方式エディタ) がインストールされている場合 (System.capabilities.hasIME) に、IME を ALPHANUMERIC_HALF に設定します。

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.ALPHANUMERIC_HALF);
  trace(System.IME.getConversionMode());
}
```

関連項目

setConversionMode (IME.setConversionMode メソッド), getConversionMode (IME.getConversionMode メソッド)

CHINESE (IME.CHINESE プロパティ)

public static CHINESE : String

"CHINESE" という値のストリングで、setConversionMode( ) および getConversionMode( ) で使用します。この定数は、簡体字中国語 IME と繁体字中国語 IME で使用します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、システムに IME (Input Method Editor : 入力方式エディタ) がインストールされている場合 (System.capabilities.hasIME) に、IME を CHINESE に設定します。

```
if(System.capabilities.hasIME) {
   trace(System.IME.getConversionMode());

   System.IME.setConversionMode(System.IME.CHINESE);
   trace(System.IME.getConversionMode());
}
```

関連項目

setConversionMode (IME.setConversionMode メソッド), getConversionMode (IME.getConversionMode メソッド)

doConversion (IME.doConversion メソッド)

public static doConversion( ) : Boolean

現在の入力ストリングの第 1 候補を選択するように IME に指示します。

対応バージョン : ActionScript 1.0、Flash Player 8

戻り値

Boolean - 呼び出しに成功した場合は true、それ外の場合は false を返します。

例

次の例は、IME 入力文字列の最初の候補を選択する方法を示しています。システムに IME がインストールされている場合、ユーザーがテキストフィールドをクリックすると、候補が選択されます。

```
var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.createTextField("txt", this.getNextHighestDepth(), 0, 0, 0, 0);
mc.txt.border = true;
mc.txt.background = true;
mc.txt.autoSize = "left";
mc.txt.text = "Set this text as the composition string and convert it.";

mc.onPress = function() {
  if(System.capabilities.hasIME) {
    Selection.setFocus(mc.txt);
    System.IME.setCompositionString(mc.txt.text);
    trace(System.IME.doConversion());
  }
}
```

getConversionMode (IME.getConversionMode メソッド)

```
public static getConversionMode() : String
```

現在の IME の変換モードを示します。

対応バージョン : ActionScript 1.0、Flash Player 8

戻り値

`String` - 変換モード。変換モードを示す、次の IME モードストリング定数を指定できます。

- `ALPHANUMERIC_FULL`
- `ALPHANUMERIC_HALF`
- `CHINESE`
- `JAPANESE_HIRAGANA`
- `JAPANESE_KATAKANA_FULL`
- `JAPANESE_KATAKANA_HALF`
- `KOREAN`
- `UNKNOWN`

例

次の例では、システムに IME (Input Method Editor : 入力方式エディタ) がインストールされている場合 (System.capabilities.hasIME) に、その IME を取得します。

```
var mode:String = System.IME.UNKNOWN;
if(System.capabilities.hasIME) {
  mode = System.IME.getConversionMode();
}
trace(mode);
```

関連項目

ALPHANUMERIC_FULL (IME.ALPHANUMERIC_FULL プロパティ), ALPHANUMERIC_HALF (IME.ALPHANUMERIC_HALF プロパティ), CHINESE (IME.CHINESE プロパティ), JAPANESE_HIRAGANA (IME.JAPANESE_HIRAGANA プロパティ), JAPANESE_KATAKANA_FULL (IME.JAPANESE_KATAKANA_FULL プロパティ), JAPANESE_KATAKANA_HALF (IME.JAPANESE_KATAKANA_HALF プロパティ), KOREAN (IME.KOREAN プロパティ), UNKNOWN (IME.UNKNOWN プロパティ)

getEnabled (IME.getEnabled メソッド)

```
public static getEnabled() : Boolean
```

システム IME が有効になっているかどうかを示します。IME が有効であるときはマルチバイト入力になり、IME が無効であるときは英数入力になります。

対応バージョン : ActionScript 1.0、Flash Player 8

戻り値

Boolean - システム IME が有効な場合は true、無効な場合は false を返します。

例

以下の例では、isEnabled() を呼び出して IME が有効になっているかどうかを確認します。

```
if(System.capabilities.hasIME) {
  var isImeEnabled:Boolean = System.IME.getEnabled();
  trace(isImeEnabled);
}
```

JAPANESE_HIRAGANA (IME.JAPANESE_HIRAGANA プロパティ)

public static JAPANESE_HIRAGANA : String

"JAPANESE_HIRAGANA" という値のストリングで、setConversionMode( ) および getConversionMode( ) で使用します。この定数は、日本語 IME で使用します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、システムに IME (Input Method Editor : 入力方式エディタ) がインストールされている場合 (System.capabilities.hasIME) に、IME を JAPANESE_HIRAGANA に設定します。

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.JAPANESE_HIRAGANA);
  trace(System.IME.getConversionMode());
}
```

関連項目

setConversionMode (IME.setConversionMode メソッド),getConversionMode (IME.getConversionMode メソッド)

JAPANESE_KATAKANA_FULL (IME.JAPANESE_KATAKANA_FULL プロパティ)

public static JAPANESE_KATAKANA_FULL : String

"JAPANESE_KATAKANA_FULL" という値のストリングで、setConversionMode( ) および getConversionMode( ) で使用します。この定数は、日本語 IME で使用します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、システムに IME (Input Method Editor : 入力方式エディタ) がインストールされている場合 (System.capabilities.hasIME) に、IME を JAPANESE_KATAKANA_FULL に設定します。

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.JAPANESE_KATAKANA_FULL);
  trace(System.IME.getConversionMode());
}
```

関連項目

setConversionMode (IME.setConversionMode メソッド), getConversionMode (IME.getConversionMode メソッド)

JAPANESE_KATAKANA_HALF (IME.JAPANESE_KATAKANA_HALF プロパティ)

public static JAPANESE_KATAKANA_HALF : String

"JAPANESE_KATAKANA_HALF" という値のストリングで、setConversionMode() および getConversionMode() で使用します。この定数は、日本語 IME で使用します。

対応バージョン: ActionScript 1.0、Flash Player 8

例

次の例では、システムに IME (Input Method Editor : 入力方式エディタ) がインストールされている場合 (System.capabilities.hasIME) に、IME を JAPANESE_KATAKANA_HALF に設定します。

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.JAPANESE_KATAKANA_HALF);
  trace(System.IME.getConversionMode());
}
```

関連項目

setConversionMode (IME.setConversionMode メソッド), getConversionMode (IME.getConversionMode メソッド)

KOREAN (IME.KOREAN プロパティ)

public static KOREAN : String

"KOREAN" という値のストリングで、setConversionMode() および getConversionMode() で使用します。この定数は、韓国語 IME で使用します。

対応バージョン: ActionScript 1.0、Flash Player 8

例

次の例では、システムに IME (Input Method Editor : 入力方式エディタ) がインストールされている場合 (System.capabilities.hasIME) に、IME を KOREAN に設定します。

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.KOREAN);
  trace(System.IME.getConversionMode());
}
```

関連項目

setConversionMode (IME.setConversionMode メソッド),getConversionMode (IME.getConversionMode メソッド)

onIMEComposition (IME.onIMEComposition イベントリスナー)

```
onIMEComposition = function([readingString:String]) {}
```

IME への入力ストリングが設定されたときに通知されます。このリスナーを使用するには、リスナーオブジェクトを作成します。次にこのリスナーの関数を定義し、IME.addListener( ) を使用してリスナーを IME オブジェクトに登録します。次に例を示します。

```
var someListener:Object = new Object();
someListener.onIMEComposition = function () {
  // statements
}
System.IME.addListener(someListener);
```

1 つのイベントに関する通知を複数のリスナーで受け取ることができるので、リスナーごとに異なる複数のコードを作成して連携させることもできます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

readingString:String (オプション) - ユーザーが候補の選択を開始する前に IME に入力された元のテキスト。

例

次の例では、コールバックメソッド onIMEComposition() を持つリスナーオブジェクトを System.IME に追加します。このコールバックメソッドは、ユーザーがテキストフィールドをクリックして入力ストリングを設定したときに通知を出力します。

```
var IMEListener:Object = new Object();
IMEListener.onIMEComposition = function(str:String) {
  trace(">> onIMEComposition: " + str);
}
System.IME.addListener(IMEListener);
trace(System.IME.length);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.createTextField("txt", this.getNextHighestDepth(), 0, 0, 0, 0);
mc.txt.border = true;
mc.txt.background = true;
mc.txt.autoSize = "left";
mc.txt.text = "Click this text to add a listener.";

mc.onPress = function() {
  if(System.capabilities.hasIME) {
    Selection.setFocus(mc.txt);
    System.IME.setCompositionString(mc.txt.text);
  }
}
```

関連項目

addListener (IME.addListener メソッド),setCompositionString (IME.setCompositionString メソッド)

removeListener (IME.removeListener メソッド)

public static removeListener(listener:Object) : Boolean

IME.addListener() を使用して IME インスタンスに以前に登録したリスナーオブジェクトを削除します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

listener:Object - IME イベントハンドラからのコールバック通知を受け取らないようにするオブジェクト。

戻り値

Boolean - リスナーオブジェクトが削除された場合は true、それ以外の場合は false を返します。

例

次の例では、ユーザーがテキストフィールドをクリックして入力ストリングを設定したときに System.IME からリスナーオブジェクトを削除します。

```
var IMEListener:Object = new Object();
IMEListener.onIMEComposition = function(str:String) {
  trace(">> onIMEComposition: " + str);

  System.IME.removeListener(this);
  trace(System.IME.length); // 0
}
System.IME.addListener(IMEListener);
trace(System.IME.length); // 1

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.createTextField("txt", this.getNextHighestDepth(), 0, 0, 0, 0);
mc.txt.border = true;
mc.txt.background = true;
mc.txt.autoSize = "left";
mc.txt.text = "Click this text to add and remove a listener.";

mc.onPress = function() {
  if(System.capabilities.hasIME) {
    Selection.setFocus(mc.txt);
    System.IME.setCompositionString(mc.txt.text);
  }
}
```

setCompositionString (IME.setCompositionString メソッド)

```
public static setCompositionString(composition:String) : Boolean
```

IME への入力ストリングを設定します。このストリングが設定されると、ユーザーは IME 候補を選択してから、現在フォーカスのあるテキストフィールドの結果を確定できます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`composition:String` - IME に送るストリング。

戻り値

`Boolean` - IME 入力文字列が正常に設定された場合は `true` を返します。どのテキストフィールドにもフォーカスがない場合、このメソッドは失敗し、`false` を返します。

例

次の例は、IME 入力文字列の設定方法を示しています。ユーザーのシステムに IME がインストールされている場合、ユーザーがテキストフィールドをクリックすると、IME の変換候補が表示されます。

```
var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.createTextField("txt", this.getNextHighestDepth(), 0, 0, 0, 0);
mc.txt.border = true;
mc.txt.background = true;
mc.txt.autoSize = "left";
mc.txt.text = "Set this text as the composition string.";

mc.onPress = function() {
  if(System.capabilities.hasIME) {
    Selection.setFocus(mc.txt);
    trace(System.IME.setCompositionString(mc.txt.text));
  }
}
```

setConversionMode (IME.setConversionMode メソッド)

```
public static setConversionMode(mode:String) : Boolean
```

現在の IME の変換モードを設定します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`mode:String` - 変換モード。次の IME モードストリング定数を指定できます。

- `ALPHANUMERIC_FULL`
- `ALPHANUMERIC_HALF`
- `CHINESE`
- `JAPANESE_HIRAGANA`
- `JAPANESE_KATAKANA_FULL`
- `JAPANESE_KATAKANA_HALF`
- `KOREAN`

戻り値

`Boolean` - 変換モードが正常に設定された場合は `true`、それ以外の場合は `false` を返します。

例

次の例では、システムに IME (Input Method Editor : 入力方式エディタ) がインストールされている場合 (System.capabilities.hasIME) に、その IME 変換モードを取得して変数 mode に代入します。

```
var mode:String = System.IME.UNKNOWN;
if(System.capabilities.hasIME) {
  mode = System.IME.getConversionMode();
}
System.IME.setConversionMode(mode);
trace(System.IME.getConversionMode());
```

関連項目

ALPHANUMERIC_FULL (IME.ALPHANUMERIC_FULL プロパティ), ALPHANUMERIC_HALF (IME.ALPHANUMERIC_HALF プロパティ), CHINESE (IME.CHINESE プロパティ), JAPANESE_HIRAGANA (IME.JAPANESE_HIRAGANA プロパティ), JAPANESE_KATAKANA_FULL (IME.JAPANESE_KATAKANA_FULL プロパティ), JAPANESE_KATAKANA_HALF (IME.JAPANESE_KATAKANA_HALF プロパティ), KOREAN (IME.KOREAN プロパティ)

setEnabled (IME.setEnabled メソッド)

```
public static setEnabled(enabled:Boolean) : Boolean
```

システム IME を有効または無効にします。IME が有効であるときはマルチバイト入力になり、IME が無効であるときは英数入力になります。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

enabled:Boolean - システム IME を有効にする場合は true、無効にする場合は false を指定します。

戻り値

Boolean - システム IME を有効にできた場合は true、それ以外の場合は false を返します。

例

次の例では、isEnabled() メソッドを呼び出して IME が有効になっているかどうかを確認した後、setEnabled() メソッドを呼び出して現在と逆の状態に変更します。

```
if(System.capabilities.hasIME) {
  var isImeEnabled:Boolean = System.IME.getEnabled();
  trace(isImeEnabled);

  if(isImeEnabled) {
    System.IME.setEnabled(false);
```

```
  }
  else {
    System.IME.setEnabled(true);
  }

  var isImeEnabled:Boolean = System.IME.getEnabled();
  trace(isImeEnabled);
}
```

UNKNOWN (IME.UNKNOWN プロパティ)

```
public static UNKNOWN : String
```

"UNKNOWN" という値のストリングで、getConversionMode() で使用します。この定数は、すべての IME で使用します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、システムに IME (Input Method Editor : 入力方式エディタ) がインストールされている場合 (System.capabilities.hasIME) に、IME を UNKNOWN に設定します。

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.UNKNOWN);
  trace(System.IME.getConversionMode());
}
```

関連項目

getConversionMode (IME.getConversionMode メソッド)

キー

```
Object
  |
  +-Key

public class Key
extends Object
```

Key クラスはトップレベルのクラスで、コンストラクタを実行しなくても、そのメソッドやプロパティを使用できます。Key クラスのメソッドを使用すると、標準キーボードで制御できるインターフェイスを作成できます。Key クラスのプロパティは、矢印キー、PageUp キー、PageDown キーなど、アプリケーションの制御に最も一般的に使用されるキーを表す定数です。

Flash アプリケーションが監視できるのは、そのフォーカス内で発生するキーボードイベントのみです。Flash アプリケーションは、別のアプリケーションでのキーボードイベントを検出できません。

対応バージョン： ActionScript 1.0、Flash Player 6

プロパティ一覧

オプション	プロパティ	説明
static	BACKSPACE:Number	Backspace キーのキーコード値 (8)。
static	CAPSLOCK:Number	CapsLock キーのキーコード値 (20)。
static	CONTROL:Number	Control キーのキーコード値 (17)。
static	DELETEKEY:Number	Delete キーのキーコード値 (46)。
static	DOWN:Number	下矢印キーのキーコード値 (40)。
static	END:Number	End キーのキーコード値 (35)。
static	ENTER:Number	Enter キーのキーコード値 (13)。
static	ESCAPE:Number	Escape キーのキーコード値 (27)。
static	HOME:Number	Home キーのキーコード値 (36)。
static	INSERT:Number	Insert キーのキーコード値 (45)。
static	LEFT:Number	左矢印キーのキーコード値 (37)。
static	_listeners:Array (読み取り専用)	Key オブジェクトに登録したすべてのリスナーオブジェクトへの参照のリスト。
static	PGDN:Number	PageDown キーのキーコード値 (34)。
static	PGUP:Number	PageUp キーのキーコード値 (33)。
static	RIGHT:Number	右矢印キーのキーコード値 (39)。
static	SHIFT:Number	Shift キーのキーコード値 (16)。
static	SPACE:Number	スペースバーのキーコード値 (32)。
static	TAB:Number	Tab キーのキーコード値 (9)。
static	UP:Number	上矢印キーのキーコード値 (38)。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベントの一覧

イベント	説明
onKeyDown = function() {}	キーを押すと通知されます。
onKeyUp = function() {}	キーを離すと通知されます。

メソッド一覧

オプション	署名	説明
static	addListener(listener: Object) : Void	onKeyDown 通知と onKeyUp 通知を受け取るオブジェクトを登録します。
static	getAscii() : Number	最後に押されたか離されたキーの ASCII コードを返します。
static	getCode() : Number	最後に押されたキーのキーコード値を返します。
static	isAccessible() : Boolean	セキュリティ制限に基づいて、最後に押されたキーに他の SWF ファイルがアクセスできるかどうかを示すブール値を返します。
static	isDown(code:Number) : Boolean	*keycode* に指定されたキーが押された場合は true、それ以外の場合は false を返します。
static	isToggled(code: Number) : Boolean	CapsLock キーまたは NumLock キーがアクティブになった場合 (アクティブ状態になった場合) は true、それ以外の場合は false を返します。
static	removeListener (listener:Object) : Boolean	Key.addListener() を使用して以前に登録したオブジェクトを削除します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

addListener (Key.addListener メソッド)

```
public static addListener(listener:Object) : Void
```

onKeyDown 通知と onKeyUp 通知を受け取るオブジェクトを登録します。キーを押すか離すと、入力フォーカスに関係なく、addListener() に登録されているすべてのリスナーオブジェクトの onKeyDown メソッドまたは onKeyUp メソッドが呼び出されます。複数のオブジェクトでキーボード通知を受け取ることができます。リスナー *newListener* が登録済みの場合は何も変化しません。

Flash アプリケーションが監視できるのは、そのフォーカス内で発生するキーボードイベントのみです。Flash アプリケーションは、別のアプリケーションでのキーボードイベントを検出できません。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

`listener:Object` - onKeyDown メソッドと onKeyUp メソッドを持つオブジェクト。

例

次の例では、新しいリスナーオブジェクトを作成し、onKeyDown と onKeyUp の関数を定義します。最後の行では、addListener() を使用してリスナーを Key オブジェクトに登録し、キーが押されるか、離されたときにイベント通知を受け取ることができるようにします。

```
var myListener:Object = new Object();
myListener.onKeyDown = function () {
  trace ("You pressed a key.");
}
myListener.onKeyUp = function () {
  trace ("You released a key.");
}
Key.addListener(myListener);
```

次の例では、my_btn というインスタンス名のボタンにキーボードショートカット Ctrl + 7 を割り当て、ショートカットに関する情報をスクリーンリーダーで利用できるようにします (_accProps を参照)。Ctrl + 7 が押されると、myOnPress 関数が "hello" というテキストを [出力] パネルに表示します。

```
function myOnPress() {
  trace("hello");
}
function myOnKeyDown() {
  // 55 is key code for 7
  if (Key.isDown(Key.CONTROL) && Key.getCode() == 55) {
  Selection.setFocus(my_btn);
  my_btn.onPress();
  }
}
var myListener:Object = new Object();
myListener.onKeyDown = myOnKeyDown;
Key.addListener(myListener);
my_btn.onPress = myOnPress;
my_btn._accProps.shortcut = "Ctrl+7";
Accessibility.updateProperties();
```

関連項目

getCode (Key.getCode メソッド), isDown (Key.isDown メソッド), onKeyDown (Key.onKeyDown イベントリスナー), onKeyUp (Key.onKeyUp イベントリスナー), removeListener (Key.removeListener メソッド)

BACKSPACE (Key.BACKSPACE プロパティ)

```
public static BACKSPACE : Number
```

Backspace キーのキーコード値 (8)。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、新しいリスナーオブジェクトを作成し、onKeyDown の関数を定義します。最後の行では、addListener() を使用してリスナーを Key オブジェクトに登録し、キーが押されたときにイベント通知を受け取ることができるようにします。

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.BACKSPACE)) {
  trace("you pressed the Backspace key.");
  } else {
  trace("you DIDN'T press the Backspace key.");
  }
};
Key.addListener(keyListener);
```

この例を使用する場合、テスト環境では [制御]-[キーボードショートカットを無効] を選択してください。

CAPSLOCK (Key.CAPSLOCK プロパティ)

```
public static CAPSLOCK : Number
```

CapsLock キーのキーコード値 (20)。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、新しいリスナーオブジェクトを作成し、onKeyDown の関数を定義します。最後の行では、addListener() を使用してリスナーを Key オブジェクトに登録し、キーが押されたときにイベント通知を受け取ることができるようにします。

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.CAPSLOCK)) {
  trace("you pressed the Caps Lock key.");
  trace("\tCaps Lock == "+Key.isToggled(Key.CAPSLOCK));
  }
};
Key.addListener(keyListener);
```

CapsLock キーが押されると、情報を [出力] パネルに表示します。isToggled メソッドを使用して CapsLock キーがアクティブになっているかどうかを調べ、その結果に応じて、[出力] パネルに true または false を表示します。

CONTROL (Key.CONTROL プロパティ)

```
public static CONTROL : Number
```

Control キーのキーコード値 (17)。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、my_btn というインスタンス名のボタンにキーボードショートカット Ctrl + 7 を割り当て、ショートカットに関する情報をスクリーンリーダーで利用できるようにします (_accProps を参照)。Ctrl + 7 が押されると、myOnPress 関数が "hello" というテキストを [出力] パネルに表示します。

```
function myOnPress() {
  trace("hello");
}
function myOnKeyDown() {
  // 55 is key code for 7
  if (Key.isDown(Key.CONTROL) && Key.getCode() == 55) {
  Selection.setFocus(my_btn);
  my_btn.onPress();
  }
}
var myListener:Object = new Object();
myListener.onKeyDown = myOnKeyDown;
Key.addListener(myListener);
my_btn.onPress = myOnPress;
my_btn._accProps.shortcut = "Ctrl+7";
Accessibility.updateProperties();
```

DELETEKEY (Key.DELETEKEY プロパティ)

```
public static DELETEKEY : Number
```

Delete キーのキーコード値 (46)。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、Drawing API とリスナーオブジェクトを使用して、マウスカーソルで線を描画できるようにします。描画した線を削除するには、BackSpace キーまたは Delete キーを押します。

```
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.drawing = true;
  canvas_mc.moveTo(_xmouse, _ymouse);
  canvas_mc.lineStyle(3, 0x99CC00, 100);
};
mouseListener.onMouseUp = function() {
  this.drawing = false;
};
mouseListener.onMouseMove = function() {
  if (this.drawing) {
  canvas_mc.lineTo(_xmouse, _ymouse);
  }
  updateAfterEvent();
};
Mouse.addListener(mouseListener);
//
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.DELETEKEY) || Key.isDown(Key.BACKSPACE)) {
  canvas_mc.clear();
  }
};
Key.addListener(keyListener);
```

この例を使用する場合、テスト環境では [制御]-[キーボードショートカットを無効] を選択してください。

この例で使用している `MovieClip.getNextHighestDepth()` メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、`MovieClip.getNextHighestDepth()` メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

DOWN (Key.DOWN プロパティ)

```
public static DOWN : Number
```

下矢印キーのキーコード値 (40)。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、矢印キーが押されたときにムービークリップ car_mc を定数の距離 (10) だけ移動します。スペースバーが押されたときはサウンドを再生します。この例で使用するライブラリ内のサウンドに対して、リンケージ識別子 horn_id を割り当ててください。

```
var DISTANCE:Number = 10;
var horn_sound:Sound = new Sound();
horn_sound.attachSound("horn_id");
var keyListener_obj:Object = new Object();
keyListener_obj.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.SPACE :
  horn_sound.start();
  break;
  case Key.LEFT :
  car_mc._x -= DISTANCE;
  break;
  case Key.UP :
  car_mc._y -= DISTANCE;
  break;
  case Key.RIGHT :
  car_mc._x += DISTANCE;
  break;
  case Key.DOWN :
  car_mc._y += DISTANCE;
  break;
  }
};
Key.addListener(keyListener_obj);
```

END (Key.END プロパティ)

```
public static END : Number
```

End キーのキーコード値 (35)。

対応バージョン : ActionScript 1.0、Flash Player 5

ENTER (Key.ENTER プロパティ)

```
public static ENTER : Number
```

Enter キーのキーコード値 (13)。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、矢印キーが押されたときにムービークリップ car_mc を定数の距離 (10) だけ移動します。Enter キーが押されると、インスタンス car_mc が停止し、onEnterFrame イベントが削除されます。

```
var DISTANCE:Number = 5;
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.LEFT :
  car_mc.onEnterFrame = function() {
    this._x -= DISTANCE;
  };
  break;
  case Key.UP :
  car_mc.onEnterFrame = function() {
    this._y -= DISTANCE;
  };
  break;
  case Key.RIGHT :
  car_mc.onEnterFrame = function() {
    this._x += DISTANCE;
  };
  break;
  case Key.DOWN :
  car_mc.onEnterFrame = function() {
    this._y += DISTANCE;
  };
  break;
  case Key.ENTER :
  delete car_mc.onEnterFrame;
  break;
  }
};
Key.addListener(keyListener);
```

この例を使用する場合、テスト環境では [制御]-[キーボードショートカットを無効] を選択してください。

ESCAPE (Key.ESCAPE プロパティ)

```
public static ESCAPE : Number
```

Escape キーのキーコード値 (27)。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、タイマーを設定します。Esc キーが押されると、キーを押してから離すまでにかかった時間を含む情報を [出力] パネルに表示します。

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.ESCAPE)) {
  // Get the current timer, convert the value to seconds and round it to two
  decimal places.
  var timer:Number = Math.round(getTimer()/10)/100;
  trace("you pressed the Esc key: "+getTimer()+" ms ("+timer+" s)");
  }
};
Key.addListener(keyListener);
```

この例を使用する場合、テスト環境では [制御]-[キーボードショートカットを無効] を選択してください。

getAscii (Key.getAscii メソッド)

```
public static getAscii() : Number
```

最後に押されたか離されたキーの ASCII コードを返します。ASCII の戻り値は英語のキーボード値です。たとえば、Shift + 2 が押されると、`Key.getAscii()` は日本語のキーボードでも英語のキーボードと同じように @ を返します。

Flash アプリケーションが監視できるのは、そのフォーカス内で発生するキーボードイベントのみです。Flash アプリケーションは、別のアプリケーションでのキーボードイベントを検出できません。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

`Number` - 最後に押されたキーの ASCII 値。このメソッドは、押されたり、離されたりしたキーがなかった場合、またはセキュリティ上の理由でキーコードにアクセスできない場合は、0 を返します。

例

次の例では、キーが押されるたびに getAscii() メソッドを呼び出します。リスナーオブジェクト keyListener を作成し、onKeyDown イベントの発生時に Key.getAscii() を呼び出す関数を定義します。次に、keyListener オブジェクトを Key オブジェクトに登録します。keyListener オブジェクトは、SWF ファイルの再生中にキーが押されるたびに onKeyDown メッセージをブロードキャストします。

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  trace("The ASCII code for the last key typed is: "+Key.getAscii());
};
Key.addListener(keyListener);
```

この例を使用する場合、テスト環境では [制御]-[キーボードショートカットを無効] を選択してください。

次の例では、Key.getAscii() の呼び出しを追加して、2 つのメソッドの相違点を示します。最も大きな違いは、Key.getAscii() では大文字と小文字が区別されるのに対し、Key.getCode() では大文字と小文字が区別されないことです。

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  trace("For the last key typed:");
  trace("\tThe Key code is: "+Key.getCode());
  trace("\tThe ASCII value is: "+Key.getAscii());
  trace("");
};
Key.addListener(keyListener);
```

この例を使用する場合、テスト環境では [制御]-[キーボードショートカットを無効] を選択してください。

関連項目

isAccessible (Key.isAccessible メソッド)

getCode (Key.getCode メソッド)

```
public static getCode() : Number
```

最後に押されたキーのキーコード値を返します。

メモ : このメソッドの Flash Lite 版は、プラットフォームによって渡されるキーコードに応じて、ストリングまたは数字を返します。有効なキーコードは、このクラスが受け付ける標準キーコードと ExtendedKey クラスのプロパティとしてリストアップされている特殊キーのコードだけです。

Flash アプリケーションが監視できるのは、そのフォーカス内で発生するキーボードイベントのみです。Flash アプリケーションは、別のアプリケーションでのキーボードイベントを検出できません。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Number - 最後に押されたキーのキーコード。このメソッドは、押されたり、離されたりしたキーがなかった場合、またはセキュリティ上の理由でキーコードにアクセスできない場合は、0 を返します。

例

次の例では、キーが押されるたびに getCode() メソッドを呼び出します。リスナーオブジェクト keyListener を作成し、onKeyDown イベントの発生時に Key.getCode() を呼び出す関数を定義します。次に、keyListener オブジェクトを Key オブジェクトに登録します。keyListener は、SWF ファイルの再生中にキーが押されるたびに onKeyDown メッセージをブロードキャストします。

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  // compare return value of getCode() to constant
  if (Key.getCode() == Key.ENTER) {
    trace ("Virtual key code: "+Key.getCode()+" (ENTER key)");
  }
  else {
    trace("Virtual key code: "+Key.getCode());
  }
};
Key.addListener(keyListener);
```

この例を使用する場合、テスト環境では [制御]-[キーボードショートカットを無効] を選択してください。

次の例では、Key.getAscii() の呼び出しを追加して、2 つのメソッドの相違点を示します。最も大きな違いは、Key.getAscii() では大文字と小文字が区別されるのに対し、Key.getCode() では大文字と小文字が区別されないことです。

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  trace("For the last key typed:");
  trace("\tThe Key code is: "+Key.getCode());
  trace("\tThe ASCII value is: "+Key.getAscii());
  trace("");
};
Key.addListener(keyListener);
```

この例を使用する場合、テスト環境では [制御]-[キーボードショートカットを無効] を選択してください。

関連項目

getAscii (Key.getAscii メソッド),isAccessible (Key.isAccessible メソッド)

HOME (Key.HOME プロパティ)

```
public static HOME : Number
```

Home キーのキーコード値 (36)。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、ドラッグ可能なムービークリップ car_mc を *x y* 座標 (0,0) に割り当てます。Home キーが押されると、car_mc は (0,0) に戻ります。この例を使用する場合は、リンケージ識別子が car_id のムービークリップを作成し、タイムラインのフレーム 1 に次の ActionScript を追加してください。

```
this.attachMovie("car_id", "car_mc", this.getNextHighestDepth(), {_x:0, _y:0});
car_mc.onPress = function() {
  this.startDrag();
};
car_mc.onRelease = function() {
  this.stopDrag();
};
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.HOME)) {
  car_mc._x = 0;
  car_mc._y = 0;
  }
};
Key.addListener(keyListener);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

INSERT (Key.INSERT プロパティ)

```
public static INSERT : Number
```

Insert キーのキーコード値 (45)。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、新しいリスナーオブジェクトを作成し、onKeyDown の関数を定義します。最後の行では、addListener() を使用してリスナーを Key オブジェクトに登録し、キーが押されたときにイベント通知を受け取り、[出力] パネルに情報を表示できるようにします。

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.INSERT)) {
  trace("You pressed the Insert key.");
  }
};
Key.addListener(keyListener);
```

isAccessible (Key.isAccessible メソッド)

```
public static isAccessible() : Boolean
```

セキュリティ制限に基づいて、最後に押されたキーに他の SWF ファイルがアクセスできるかどうかを示すブール値を返します。デフォルトでは、特定のドメインにある SWF ファイルのコードから、別のドメインにある SWF ファイルによって生成されるキーストロークにアクセスすることはできません。ドメイン間のセキュリティの詳細については、『ActionScript 2.0 の学習』の「セキュリティについて」を参照してください。

対応バージョン : ActionScript 1.0、Flash Player 8

戻り値

Boolean - 最後に押されたキーにアクセスできる場合は true を返します。アクセスが許可されていない場合は false を返します。

isDown (Key.isDown メソッド)

```
public static isDown(code:Number) : Boolean
```

keycode に指定されたキーが押された場合は true、それ以外の場合は false を返します。Macintosh では、CapsLock キーと NumLock キーのキーコード値は同じです。

Flash アプリケーションが監視できるのは、そのフォーカス内で発生するキーボードイベントのみです。Flash アプリケーションは、別のアプリケーションでのキーボードイベントを検出できません。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

code:Number - 特定のキーに割り当てられたキーコード値、または特定のキーに関連付けられた Key クラスプロパティ。

戻り値

Boolean - *keycode* に指定されたキーが押された場合は true、それ以外の場合は false を返します。

例

次のスクリプトは、ユーザーがムービークリップ car_mc の位置を制御できるようにします。

```
car_mc.onEnterFrame = function() {
 if (Key.isDown(Key.RIGHT)) {
 this._x += 10;
 } else if (Key.isDown(Key.LEFT)) {
 this._x -= 10;
 }
};
```

isToggled (Key.isToggled メソッド)

```
public static isToggled(code:Number) : Boolean
```

CapsLock キーまたは NumLock キーがアクティブになった場合 (アクティブ状態になった場合) は true、それ以外の場合は false を返します。通常、"toggled" という用語は、2 つの選択肢の間で何かが切り替わったことを表しますが、Key.isToggled() メソッドは、キーがアクティブ状態になった場合にのみ true を返します。Macintosh では、CapsLock キーと NumLock キーのキーコード値は同じです。

Flash アプリケーションが監視できるのは、そのフォーカス内で発生するキーボードイベントのみです。Flash アプリケーションは、別のアプリケーションでのキーボードイベントを検出できません。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

code:Number - CapsLock キー (20) または NumLock キー (144) のキーコード。

戻り値

Boolean - CapsLock キーまたは NumLock キーがアクティブになった場合 (アクティブ状態になった場合) は true、それ以外の場合は false を返します。

例

次の例では、キーが押されるたびに isToggled() メソッドを呼び出し、CapsLock キーがアクティブ状態に切り替わるたびに trace ステートメントを実行します。リスナーオブジェクト keyListener を作成し、onKeyDown イベントの発生時に Key.isToggled() を呼び出す関数を定義します。次に、keyListener オブジェクトを Key オブジェクトに登録します。keyListener は、SWF ファイルの再生中にキーが押されるたびに onKeyDown メッセージをブロードキャストします。

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.CAPSLOCK)) {
  trace("you pressed the Caps Lock key.");
  trace("\tCaps Lock == "+Key.isToggled(Key.CAPSLOCK));
  }
};
Key.addListener(keyListener);
```

CapsLock キーが押されると、情報を [出力] パネルに表示します。isToggled メソッドを使用して CapsLock キーがアクティブになっているかどうかを調べ、その結果に応じて、[出力] パネルに true または false を表示します。

次の例では、CapsLock キーおよび NumLock キーが切り替えられたときに更新される 2 つのテキストフィールドを作成します。それぞれのテキストフィールドでは、対応するキーがアクティブになると true が表示され、キーが非アクティブになると false が表示されます。

```
this.createTextField("capsLock_txt", this.getNextHighestDepth(), 0, 0, 100, 22);
capsLock_txt.autoSize = true;
capsLock_txt.html = true;
this.createTextField("numLock_txt", this.getNextHighestDepth(), 0, 22, 100, 22);
numLock_txt.autoSize = true;
numLock_txt.html = true;
//
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  capsLock_txt.htmlText = "<b>Caps Lock:</b> "+Key.isToggled(Key.CAPSLOCK);
  numLock_txt.htmlText = "<b>Num Lock:</b> "+Key.isToggled(144);
};
Key.addListener(keyListener);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

LEFT (Key.LEFT プロパティ)

```
public static LEFT : Number
```

左矢印キーのキーコード値 (37)。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、矢印キーが押されたときにムービークリップ car_mc を定数の距離 (10) だけ移動します。スペースバーが押されたときはサウンドを再生します。この例で使用するライブラリ内のサウンドに対して、リンケージ識別子 horn_id を割り当ててください。

```
var DISTANCE:Number = 10;
var horn_sound:Sound = new Sound();
horn_sound.attachSound("horn_id");
var keyListener_obj:Object = new Object();
keyListener_obj.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.SPACE :
  horn_sound.start();
  break;
  case Key.LEFT :
  car_mc._x -= DISTANCE;
  break;
  case Key.UP :
  car_mc._y -= DISTANCE;
  break;
  case Key.RIGHT :
  car_mc._x += DISTANCE;
  break;
  case Key.DOWN :
  car_mc._y += DISTANCE;
  break;
  }
};
Key.addListener(keyListener_obj);
```

_listeners (Key._listeners プロパティ)

```
public static _listeners : Array ( 読み取り専用 )
```

Key オブジェクトに登録したすべてのリスナーオブジェクトへの参照のリスト。このプロパティは内部処理のためのものですが、Key オブジェクトに現時点で登録されているリスナー数を確認する場合に便利なことがあります。addListener( ) メソッドと removelistener( ) メソッドを呼び出すことで、この配列にオブジェクトを追加したり、この配列からオブジェクトを削除したりします。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、length プロパティを使用して、Key オブジェクトに現時点で登録されているリスナーオブジェクト数を確認します。

```
var myListener:Object = new Object();
myListener.onKeyDown = function () {
trace ("You pressed a key.");
}
Key.addListener(myListener);

trace(Key._listeners.length); // Output: 1
```

onKeyDown (Key.onKeyDown イベントリスナー)

```
onKeyDown = function() {}
```

キーを押すと通知されます。onKeyDown を使用するには、リスナーオブジェクトを作成する必要があります。その後、onKeyDown 用の関数を定義し、addListener( ) を使用してリスナーを Key オブジェクトに登録します。次に例を示します。

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  trace("DOWN -> Code: "+Key.getCode()+"\tASCII: "+Key.getAscii()+"\tKey:
  "+chr(Key.getAscii()));
};
keyListener.onKeyUp = function() {
  trace("UP -> Code: "+Key.getCode()+"\tASCII: "+Key.getAscii()+"\tKey:
  "+chr(Key.getAscii()));
};
Key.addListener(keyListener);
```

1 つのイベントに関する通知を複数のリスナーで受け取ることができるので、リスナーごとに異なる複数のコードを作成して連携させることもできます。

Flash アプリケーションが監視できるのは、そのフォーカス内で発生するキーボードイベントのみです。Flash アプリケーションは、別のアプリケーションでのキーボードイベントを検出できません。

対応バージョン： ActionScript 1.0、Flash Player 6

関連項目

addListener (Key.addListener メソッド)

onKeyUp (Key.onKeyUp イベントリスナー)

```
onKeyUp = function() {}
```

キーを離すと通知されます。onKeyUp を使用するには、リスナーオブジェクトを作成する必要があります。その後、onKeyUp 用の関数を定義し、addListener() を使用してリスナーを Key オブジェクトに登録します。次に例を示します。

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  trace("DOWN -> Code: "+Key.getCode()+"\tASCII: "+Key.getAscii()+"\tKey:
  "+chr(Key.getAscii()));
};
keyListener.onKeyUp = function() {
  trace("UP -> Code: "+Key.getCode()+"\tASCII: "+Key.getAscii()+"\tKey:
  "+chr(Key.getAscii()));
};
Key.addListener(keyListener);
```

1 つのイベントに関する通知を複数のリスナーで受け取ることができるので、リスナーごとに異なる複数のコードを作成して連携させることもできます。

Flash アプリケーションが監視できるのは、そのフォーカス内で発生するキーボードイベントのみです。Flash アプリケーションは、別のアプリケーションでのキーボードイベントを検出できません。

対応バージョン： ActionScript 1.0、Flash Player 6

関連項目

addListener (Key.addListener メソッド)

PGDN (Key.PGDN プロパティ)

```
public static PGDN : Number
```

PageDown キーのキーコード値 (34)。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、PageDown キーまたは PageUp キーが押されたときにムービークリップ car_mc を回転させます。

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.PGDN)) {
  car_mc._rotation += 5;
  } else if (Key.isDown(Key.PGUP)) {
  car_mc._rotation -= 5;
  }
};
Key.addListener(keyListener);
```

PGUP (Key.PGUP プロパティ)

```
public static PGUP : Number
```

PageUp キーのキーコード値 (33)。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、PageDown キーまたは PageUp キーが押されたときにムービークリップ car_mc を回転させます。

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.PGDN)) {
  car_mc._rotation += 5;
  } else if (Key.isDown(Key.PGUP)) {
  car_mc._rotation -= 5;
  }
};
Key.addListener(keyListener);
```

removeListener (Key.removeListener メソッド)

```
public static removeListener(listener:Object) : Boolean
```

Key.addListener() を使用して以前に登録したオブジェクトを削除します。

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

`listener:Object` - オブジェクト。

戻り値

`Boolean` - *`listener`* が正常に削除された場合は `true` を返します。*`listener`* が Key オブジェクトのリスナーリストになかった場合など、*`listener`* が正常に削除されなかった場合は `false` を返します。

例

次の例では、左矢印キーと右矢印キーを使用して、ムービークリップ `car_mc` を移動します。Esc キーを押すとリスナーが削除されるので、`car_mc` は移動しなくなります。

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.LEFT :
  car_mc._x -= 10;
  break;
  case Key.RIGHT :
  car_mc._x += 10;
  break;
  case Key.ESCAPE :
  Key.removeListener(keyListener);
  }
};
Key.addListener(keyListener);
```

RIGHT (Key.RIGHT プロパティ)

```
public static RIGHT : Number
```

右矢印キーのキーコード値 (39)。

対応バージョン： ActionScript 1.0、Flash Player 5

例

次の例では、矢印キーが押されたときにムービークリップ car_mc を定数の距離 (10) だけ移動します。スペースバーが押されたときはサウンドを再生します。この例では、ライブラリ内のサウンドにリンケージ識別子 horn_id を割り当てます。

```
var DISTANCE:Number = 10;
var horn_sound:Sound = new Sound();
horn_sound.attachSound("horn_id");
var keyListener_obj:Object = new Object();
keyListener_obj.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.SPACE :
  horn_sound.start();
  break;
  case Key.LEFT :
  car_mc._x -= DISTANCE;
  break;
  case Key.UP :
  car_mc._y -= DISTANCE;
  break;
  case Key.RIGHT :
  car_mc._x += DISTANCE;
  break;
  case Key.DOWN :
  car_mc._y += DISTANCE;
  break;
  }
};
Key.addListener(keyListener_obj);
```

SHIFT (Key.SHIFT プロパティ)

```
public static SHIFT : Number
```

Shift キーのキーコード値 (16)。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、Shift キーが押されたときに car_mc を拡大します。

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.SHIFT)) {
  car_mc._xscale = 2;
  car_mc._yscale = 2;
  } else if (Key.isDown(Key.CONTROL)) {
  car_mc._xscale /= 2;
  car_mc._yscale /= 2;
  }
};
Key.addListener(keyListener);
```

SPACE (Key.SPACE プロパティ)

```
public static SPACE : Number
```

スペースバーのキーコード値 (32)。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、矢印キーが押されたときにムービークリップ car_mc を定数の距離 (10) だけ移動します。スペースバーが押されたときはサウンドを再生します。この例では、ライブラリ内のサウンドにリンケージ識別子 horn_id を割り当てます。

```
var DISTANCE:Number = 10;
var horn_sound:Sound = new Sound();
horn_sound.attachSound("horn_id");
var keyListener_obj:Object = new Object();
keyListener_obj.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.SPACE :
  horn_sound.start();
  break;
  case Key.LEFT :
  car_mc._x -= DISTANCE;
  break;
  case Key.UP :
  car_mc._y -= DISTANCE;
```

```
  break;
  case Key.RIGHT :
  car_mc._x += DISTANCE;
  break;
  case Key.DOWN :
  car_mc._y += DISTANCE;
  break;
  }
};
Key.addListener(keyListener_obj);
```

TAB (Key.TAB プロパティ)

```
public static TAB : Number
```

Tab キーのキーコード値 (9)。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、テキストフィールドを作成し、Tab キーが押されたときに日付をテキストフィールドに表示します。

```
this.createTextField("date_txt", this.getNextHighestDepth(), 0, 0, 100, 22);
date_txt.autoSize = true;
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.TAB)) {
  var today_date:Date = new Date();
  date_txt.text = today_date.toString();
  }
};
Key.addListener(keyListener);
```

この例を使用する場合、テスト環境では [制御]-[キーボードショートカットを無効] を選択してください。

この例で使用している `MovieClip.getNextHighestDepth()` メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、`MovieClip.getNextHighestDepth()` メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

UP (Key.UP プロパティ)

```
public static UP : Number
```

上矢印キーのキーコード値 (38)。

対応バージョン： ActionScript 1.0、Flash Player 5

例

次の例では、矢印キーが押されたときにムービークリップ car_mc を定数の距離 (10) だけ移動します。スペースバーが押されたときはサウンドを再生します。この例では、ライブラリ内のサウンドにリンケージ識別子 horn_id を割り当てます。

```
var DISTANCE:Number = 10;
var horn_sound:Sound = new Sound();
horn_sound.attachSound("horn_id");
var keyListener_obj:Object = new Object();
keyListener_obj.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.SPACE :
  horn_sound.start();
  break;
  case Key.LEFT :
  car_mc._x -= DISTANCE;
  break;
  case Key.UP :
  car_mc._y -= DISTANCE;
  break;
  case Key.RIGHT :
  car_mc._x += DISTANCE;
  break;
  case Key.DOWN :
  car_mc._y += DISTANCE;
  break;
  }
};
Key.addListener(keyListener_obj);
```

LoadVars

```
Object
  |
  +-LoadVars
```

```
public dynamic class LoadVars
extends Object
```

LoadVars クラスを使用すると、データのロードが成功したかどうかを確認することや、ダウンロードの進行状況を監視できます。loadVariables() 関数の代わりに LoadVars クラスを使用して、Flash アプリケーションとサーバーの間で変数を転送できます。

具体的には、LoadVars クラスはオブジェクト内のすべての変数を指定の URL に送ったり、指定された URL にあるすべての変数をオブジェクトにロードしたりできます。また、すべての変数ではなく特定の変数を送信することもできるので、アプリケーションの効率が向上します。LoadVars.onLoad ハンドラを使用すると、(データがロードされる前ではなく) データがロードされるときにアプリケーションが実行されるようにすることができます。

LoadVars クラスの動作は XML クラスとよく似ています。このクラスは、load() メソッド、send() メソッド、および sendAndLoad() メソッドを使用してサーバーと通信します。LoadVars クラスと XML クラスの大きな違いは、LoadVars クラスが ActionScript の名前と値のペアを転送するのに対して、XML クラスは XML オブジェクトに格納されている XML DOM (ドキュメントオブジェクトモデル : Document Object Model) ツリーを転送するということです。LoadVars クラスは、XML クラスと同じセキュリティ制限に従います。

対応バージョン : ActionScript 1.0、Flash Player 6

関連項目

loadVariables 関数, onLoad (LoadVars.onLoad ハンドラ), XML

プロパティー覧

オプション	プロパティ	説明
	contentType:String	LoadVars.send() または LoadVars.sendAndLoad() を呼び出したときにサーバーに送られる MIME タイプ。
	loaded:Boolean	load 処理または sendAndLoad 処理が完了したかどうかを示すブール値です。デフォルトは undefined です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベントの一覧

イベント	説明
onData = function(src: String) {}	サーバーからのデータのダウンロードが完了したとき、またはサーバーからデータをダウンロード中にエラーが発生したときに呼び出されます。
onHTTPStatus = function(httpStatus: Number) {}	Flash Player がサーバーから HTTP ステータスコードを受け取ると、呼び出されます。
onLoad = function(success: Boolean) {}	LoadVars.load() 処理または LoadVars.sendAndLoad() 処理が完了したときに呼び出されます。

コンストラクター覧

署名	説明
LoadVars()	LoadVars オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	addRequestHeader (header:Object, headerValue: String) : Void	POST アクションによって送信される HTTP リクエストヘッダ (Content-Type や SOAPAction など) を追加または変更します。
	decode(queryString: String) : Void	変数ストリングを、指定された LoadVars オブジェクトのプロパティに変換します。
	getBytesLoaded() : Number	LoadVars.load() または LoadVars.sendAndLoad() によってダウンロードされたバイト数を返します。
	getBytesTotal() : Number	LoadVars.load() または LoadVars.sendAndLoad() によってダウンロードされた総バイト数を返します。
	load(url:String) : Boolean	指定された URL から変数をダウンロードし、変数データを解析し、結果として得られた変数を my_lv に代入します。
	send(url:String, target:String, [method:String]) : Boolean	*my_lv* オブジェクト内の変数を、指定された URL に送信します。

オプション	署名	説明
	sendAndLoad(url:String, target:Object, [method:String]) : Boolean	my_lv オブジェクト内の変数を、指定された URL に送信 (Post) します。
	toString() : String	*my_lv* 内の列挙可能な変数をすべて含むストリングを、MIME コンテンツエンコード *application/x-www-form-urlencoded* で返します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

addRequestHeader (LoadVars.addRequestHeader メソッド)

public addRequestHeader(header:Object, headerValue:String) : Void

POST アクションによって送信される HTTP リクエストヘッダ (Content-Type や SOAPAction など) を追加または変更します。シンタックス 1 では、2 つのストリング (header および headerValue) をメソッドに渡します。シンタックス 2 では、ヘッダー名とヘッダー値を交互に含むストリングの配列を渡します。

同じヘッダ名に対して複数の呼び出しを実行すると、呼び出しのたびに前の呼び出しで設定された値が上書きされます。

次に示すリクエストヘッダーは使用できません。また、制限されたこれらのストリングは、大文字と小文字が区別されません。したがって、たとえば Get、get、GET はすべて使用できません。

Accept-Charset, Accept-Encoding, Accept-Ranges, Age, Allow, Allowed, Connection, Content-Length, Content-Location, Content-Range, Date, Delete, ETag, Expect, Get, Host, Keep-Alive, Last-Modified, Location, Max-Forwards, Options, Post, Proxy-Authenticate, Proxy-Authorization, Public, Put, Range, Referer, Retry-After, Server, TE, Trace, Trailer, Transfer-Encoding, Upgrade, URI, User-Agent, Vary, Via, Warning, WWW-Authenticate, x-flash-version.

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

`header:Object` - HTTP リクエストヘッダー名を表すストリング、またはストリングの配列。

`headerValue:String` - `header` に関連付けられた値を表すストリング。

例

次の例では、値が `Foo` である `SOAPAction` というカスタム HTTP ヘッダーを `my_lv` オブジェクトに追加します。

```
my_lv.addRequestHeader("SOAPAction", "'Foo'");
```

次の例では、`headers` という配列を作成します。この配列には、HTTP ヘッダーとその値が交互に格納されます。この配列を `addRequestHeader()` に引数として渡します。

```
var headers = ["Content-Type", "text/plain", "X-ClientAppVersion", "2.0"];
my_lv.addRequestHeader(headers);
```

次の例では、`FLASH-UUID` というリクエストヘッダーを追加する新しい LoadVars オブジェクトを作成します。ヘッダーには変数が格納され、サーバー側でチェックできます。

```
var my_lv:LoadVars = new LoadVars();
my_lv.addRequestHeader("FLASH-UUID", "41472");
my_lv.name = "Mort";
my_lv.age = 26;
my_lv.send("http://flash-mx.com/mm/cgivars.cfm", "_blank", "POST");
```

関連項目

addRequestHeader (XML.addRequestHeader メソッド)

contentType (LoadVars.contentType プロパティ)

```
public contentType : String
```

`LoadVars.send()` または `LoadVars.sendAndLoad()` を呼び出したときにサーバーに送られる MIME タイプ。デフォルトは *application/x-www-form-urlencoded* です。

対応バージョン: ActionScript 1.0、Flash Player 6

例

次の例では、LoadVars オブジェクトを作成し、サーバーに送信されるデータのデフォルトのコンテンツタイプを表示します。

```
var my_lv:LoadVars = new LoadVars();
trace(my_lv.contentType); // output: application/x-www-form-urlencoded
```

関連項目

send (LoadVars.send メソッド), sendAndLoad (LoadVars.sendAndLoad メソッド)

decode (LoadVars.decode メソッド)

```
public decode(queryString:String) : Void
```

変数ストリングを、指定された LoadVars オブジェクトのプロパティに変換します。

このメソッドは、LoadVars.onData イベントハンドラによって内部的に使用されます。ほとんどの場合、ユーザーが直接このメソッドを呼び出す必要はありません。LoadVars.onData イベントハンドラを上書きした場合は、LoadVars.decode( ) を明示的に呼び出して変数のストリングを解析できます。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

`queryString:String` - 名前と値のペアを含む、URL エンコードされたクエリーストリング。

例

次の例では、3 つの変数をトレースしています。

```
// Create a new LoadVars object
var my_lv:LoadVars = new LoadVars();
//Convert the variable string to properties
my_lv.decode("name=Mort&score=250000");
trace(my_lv.toString());
// Iterate over properties in my_lv
for (var prop in my_lv) {
  trace(prop+" -> "+my_lv[prop]);
}
```

関連項目

onData (LoadVars.onData ハンドラ), parseXML (XML.parseXML メソッド)

getBytesLoaded (LoadVars.getBytesLoaded メソッド)

```
public getBytesLoaded() : Number
```

LoadVars.load( ) または LoadVars.sendAndLoad( ) によってダウンロードされたバイト数を返します。ロード処理が実行中ではない場合、またはまだ始まっていない場合は、undefined を返します。

対応バージョン : ActionScript 1.0、Flash Player 6

戻り値

Number - 整数。

例

次の例では、ProgressBar インスタンスおよび LoadVars オブジェクトを使用して、テキストファイルをダウンロードします。ファイルをテストすると、ファイルのロードの成否と SWF ファイルにロードされたデータ量が [出力] パネルに表示されます。LoadVars.load() コマンドの URL パラメータは、HTTP を使用して有効なテキストファイルを指定するように値を置き換える必要があります。この例を使用してハードディスク上にあるローカルファイルをロードしようとしても、ムービープレビューモードの Flash Player ではローカルファイル全体がロードされるので、正常に機能しません。このコードの動作を確認するには、ProgressBar インスタンス loadvars_pb をステージに追加します。次に、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var loadvars_pb:mx.controls.ProgressBar;
var my_lv:LoadVars = new LoadVars();
loadvars_pb.mode = "manual";
this.createEmptyMovieClip("timer_mc", 999);
timer_mc.onEnterFrame = function() {
  var lvBytesLoaded:Number = my_lv.getBytesLoaded();
  var lvBytesTotal:Number = my_lv.getBytesTotal();
  if (lvBytesTotal != undefined) {
  trace("Loaded "+lvBytesLoaded+" of "+lvBytesTotal+" bytes.");
  loadvars_pb.setProgress(lvBytesLoaded, lvBytesTotal);
  }
};
my_lv.onLoad = function(success:Boolean) {
  loadvars_pb.setProgress(my_lv.getBytesLoaded(), my_lv.getBytesTotal());
  delete timer_mc.onEnterFrame;
  if (success) {
  trace("LoadVars loaded successfully.");
  } else {
  trace("An error occurred while loading variables.");
  }
};
my_lv.load("[place a valid URL pointing to a text file here]");
```

関連項目

load (LoadVars.load メソッド),sendAndLoad (LoadVars.sendAndLoad メソッド)

getBytesTotal (LoadVars.getBytesTotal メソッド)

```
public getBytesTotal() : Number
```

LoadVars.load() または LoadVars.sendAndLoad() によってダウンロードされた総バイト数を返します。ロード処理が実行中ではない場合、またはまだ始まっていない場合には、undefined を返します。総バイト数を判別できない場合 (ダウンロードは開始したが、サーバーが HTTP のコンテンツ長を送信しなかった場合など) にも undefined を返します。

対応バージョン : ActionScript 1.0、Flash Player 6

戻り値

Number - 整数。

例

次の例では、ProgressBar インスタンスおよび LoadVars オブジェクトを使用して、テキストファイルをダウンロードします。ファイルをテストすると、ファイルのロードの成否と SWF ファイルにロードされたデータ量が [出力] パネルに表示されます。LoadVars.load() コマンドの URL パラメータは、HTTP を使用して有効なテキストファイルを指定するように値を置き換える必要があります。この例を使用してハードディスク上にあるローカルファイルをロードしようとしても、ムービープレビューモードの Flash Player ではローカルファイル全体がロードされるので、正常に機能しません。このコードの動作を確認するには、ProgressBar インスタンス loadvars_pb をステージに追加します。次に、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var loadvars_pb:mx.controls.ProgressBar;
var my_lv:LoadVars = new LoadVars();
loadvars_pb.mode = "manual";
this.createEmptyMovieClip("timer_mc", 999);
timer_mc.onEnterFrame = function() {
  var lvBytesLoaded:Number = my_lv.getBytesLoaded();
  var lvBytesTotal:Number = my_lv.getBytesTotal();
  if (lvBytesTotal != undefined) {
  trace("Loaded "+lvBytesLoaded+" of "+lvBytesTotal+" bytes.");
  loadvars_pb.setProgress(lvBytesLoaded, lvBytesTotal);
  }
};
my_lv.onLoad = function(success:Boolean) {
  loadvars_pb.setProgress(my_lv.getBytesLoaded(), my_lv.getBytesTotal());
  delete timer_mc.onEnterFrame;
  if (success) {
  trace("LoadVars loaded successfully.");
  } else {
  trace("An error occurred while loading variables.");
  }
};
my_lv.load("[place a valid URL pointing to a text file here]");
```

関連項目

load (LoadVars.load メソッド), sendAndLoad (LoadVars.sendAndLoad メソッド)

load (LoadVars.load メソッド)

```
public load(url:String) : Boolean
```

指定された URL から変数をダウンロードし、変数データを解析し、結果として得られた変数を my_lv に代入します。my_lv のプロパティのうち、ダウンロードされた変数と名前が同じものは上書きされます。my_lv のプロパティのうち、ダウンロードされた変数と名前が異なるものは削除されません。これは非同期アクションです。

ダウンロードされたデータは、MIME コンテンツタイプ *application/x-www-form-urlencoded* である必要があります。

これは loadVariables() で使用されるのと同じ形式です。

また、Flash Player 7 用にパブリッシュされたファイルでは、LoadVars.load() でロードされる外部変数の大文字と小文字の区別に対応しています。

このメソッドは XML.load() と似ています。

メモ : ロード中のファイルに ASCII 文字以外の文字 (英語以外の多くの言語に存在する) が含まれている場合は、ASCII のような非 Unicode 形式ではなく UTF-8 または UTF-16 エンコーディング形式でファイルを保存することをお勧めします。

このメソッドを使用するときは、Flash Player セキュリティモデルを考慮してください。

Flash Player 8 :

- 呼び出し元 SWF ファイルが ローカルファイルシステムのサンドボックスにあり、ターゲットリソースがネットワーク上のサンドボックスにある場合、データをロードできません。
- 呼び出し元 SWF ファイルがネットワーク上のサンドボックスにあり、ターゲットリソースがローカルにある場合、データをロードできません。

詳細については、次の参照先を参照してください。

- 『ActionScript 2.0 の学習』の「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

Flash Player 7 以降では、クロスドメインポリシーファイルによって、Web サイトでのリソースへのクロスドメインアクセスを許可することができます。Flash Player 7 以降で SWF ファイルを実行している場合、url はまったく同じドメインに属している必要があります。たとえば、www.someDomain.com に存在する SWF ファイルは、store.someDomain.com に存在するソースからデータをロードできません。これは、両方のファイルが同じドメインに属していないためです。

Flash Player 7 より前のバージョンの Player で SWF ファイルを実行している場合、url は、呼び出し元の SWF ファイルと同じスーパードメインに属している必要があります。スーパードメインは、ファイルの URL の左端の要素を削除することで求められます。たとえば、www.someDomain.com に存在する SWF ファイルは、store.someDomain.com に存在するソースからデータをロードできます。これは、どちらのファイルも同じスーパードメイン someDomain.com に属しているためです。

対応バージョン : ActionScript 1.0、Flash Player 6 - Flash Player 7 ではビヘイビアが変更されました。

パラメータ

url:String - ストリング。変数のダウンロード元の URL。呼び出し元の SWF ファイルが Web ブラウザ内で実行している場合、host は SWF ファイルと同じドメインに属している必要があります。

戻り値

Boolean - パラメータが渡されなかった場合 (パラメータが null の場合) は false、それ以外の場合は true を返します。データのロードに成功したかどうかを確認するには、onLoad() イベントハンドラを使用します。

例

次のコードでは、サーバー側の PHP スクリプトから Flash アプリケーションにデータが返されたことを通知する onLoad ハンドラ関数を定義した後、データを passvars.php にロードします。

```
var my_lv:LoadVars = new LoadVars();
my_lv.onLoad = function(success:Boolean) {
  if (success) {
  trace(this.toString());
  } else {
  trace("Error loading/parsing LoadVars.");
  }
};
my_lv.load("http://www.helpexamples.com/flash/params.txt");
```

別の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/DataIntegration\Guestbook" フォルダに移動して guestbook.fla ファイルにアクセスします。

関連項目

load (XML.load メソッド), loaded (LoadVars.loaded プロパティ), onLoad (LoadVars.onLoad ハンドラ), useCodepage (System.useCodepage プロパティ)

loaded (LoadVars.loaded プロパティ)

```
public loaded : Boolean
```

load 処理または sendAndLoad 処理が完了したかどうかを示すブール値です。デフォルトは undefined です。LoadVars.load( ) または LoadVars.sendAndLoad( ) の処理が始まると、loaded プロパティは false に設定されます。処理が完了すると、loaded プロパティは true に設定されます。処理が完了していない場合、またはエラーが発生して失敗した場合は、loaded プロパティは false に設定されたままです。

このプロパティは XML.loaded プロパティに似ています。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、テキストファイルをロードし、処理が完了したときに情報を [出力] パネルに表示します。

```
var my_lv:LoadVars = new LoadVars();
my_lv.onLoad = function(success:Boolean) {
  trace("LoadVars loaded successfully: "+this.loaded);
};
my_lv.load("http://www.helpexamples.com/flash/params.txt");
```

関連項目

load (LoadVars.load メソッド), sendAndLoad (LoadVars.sendAndLoad メソッド), load (XML.load メソッド)

LoadVars コンストラクタ

```
public LoadVars()
```

LoadVars オブジェクトを作成します。その LoadVars オブジェクトのメソッドを使用してデータを送信およびロードできます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、my_lv という LoadVars オブジェクトを作成します。

```
var my_lv:LoadVars = new LoadVars();
```

onData (LoadVars.onData ハンドラ)

```
onData = function(src:String) {}
```

サーバーからのデータのダウンロードが完了したとき、またはサーバーからデータをダウンロード中にエラーが発生したときに呼び出されます。このハンドラはデータが解析される前に呼び出されるので、Flash Player に組み込まれている解析ルーチンではなく、独自の解析ルーチンを呼び出す場合に利用できます。LoadVars.onData に割り当てられた関数の src パラメータの値は undefined であるか、またはサーバーからダウンロードされた URL エンコード形式の名前と値のペアを含むストリングです。src パラメータが undefined である場合は、サーバーからのデータのダウンロード時にエラーが発生したことを示しています。

デフォルトの実装では、LoadVars.onData は LoadVars.onLoad を呼び出します。LoadVars.onData にカスタム関数を割り当ててデフォルトの実装を上書きすることはできますが、独自に実装した LoadVars.onData の中で呼び出さない限り、LoadVars.onLoad は呼び出されません。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

src:String - ストリングまたは undefined。LoadVars.load( ) メソッドまたは LoadVars.sendAndLoad( ) メソッドの呼び出しから返される生 (未解析) のデータ。

例

次の例では、テキストファイルをロードし、ロード処理の完了時に内容を TextArea インスタンス content_ta に表示します。エラーが発生した場合は、その情報を [出力] パネルに表示します。

```
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
  if (src == undefined) {
    trace("Error loading content.");
    return;
  }
  content_ta.text = src;
};
my_lv.load("content.txt", my_lv, "GET");
```

関連項目

onLoad (LoadVars.onLoad ハンドラ), onLoad (LoadVars.onLoad ハンドラ), load (LoadVars.load メソッド), sendAndLoad (LoadVars.sendAndLoad メソッド)

onHTTPStatus (LoadVars.onHTTPStatus ハンドラ)

```
onHTTPStatus = function(httpStatus:Number) {}
```

Flash Player がサーバーから HTTP ステータスコードを受け取ると、呼び出されます。このハンドラを使用することにより、HTTP ステータスコードを取得してそれに基づいて処理を行えます。

onHTTPStatus ハンドラは onData ハンドラの前に呼び出され、ロードが失敗した場合には undefined 値を返して onLoad の呼び出しを実行します。onHTTPStatus を上書きしているかどうかにかかわらず、onHTTPStatus がトリガされた後、onData が常にトリガされます。onHTTPStatus ハンドラを最も効果的に使用するには、onHTTPStatus 呼び出しの結果をキャッチするための関数を作成する必要があります。その結果を onData ハンドラと onLoad ハンドラで使用します。onHTTPStatus が呼び出されない場合は、URL リクエストが試みられなかったことを示します。これは、その要求が SWF ファイルのセキュリティサンドボックス規則に違反している場合に起こる可能性があります。

Flash Player がサーバーからステータスコードを取得できなかった場合、またはサーバーと通信できなかった場合、記述した ActionScript のコードにデフォルト値の 0 が渡されます。値 0 は、(たとえば正しくない形式の URL が要求された場合などに) どのプレーヤーでも生成される可能性があります。また、サーバーからの HTTP ステータスコードをプレーヤーに渡すことができないブラウザで実行される Flash Player プラグインでは、常に値 0 が生成されます。該当するブラウザには、Netscape、Mozilla、Safari、Opera、および Internet Explorer for Macintosh があります。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

httpStatus : Number - サーバーから返された HTTP ステータスコード。たとえば、値 404 は、要求された URI と一致する URI が見つからなかったことを示します。HTTP ステータスコードは、ftp://ftp.isi.edu/in-notes/rfc2616.txt にある HTTP 仕様書のセクション 10.4 と 10.5 に記載されています。

例

次の例では、onHTTPStatus( ) をデバッグに役立てる方法を示します。この例では、HTTP ステータスコードを収集し、その値とタイプを LoadVars オブジェクトのインスタンスに代入します。この例では、インスタンスのメンバー this.httpStatus と this.httpStatusType を実行時に作成しています。onData メソッドでこれらのメンバーを使用して、デバッグに役立てることができる HTTP 応答に関する情報をトレースします。

```
var myLoadVars:LoadVars = new LoadVars();

myLoadVars.onHTTPStatus = function(httpStatus:Number) {
  this.httpStatus = httpStatus;
  if(httpStatus < 100) {
    this.httpStatusType = "flashError";
  }
  else if(httpStatus < 200) {
    this.httpStatusType = "informational";
  }
  else if(httpStatus < 300) {
    this.httpStatusType = "successful";
  }
  else if(httpStatus < 400) {
    this.httpStatusType = "redirection";
  }
  else if(httpStatus < 500) {
    this.httpStatusType = "clientError";
  }
  else if(httpStatus < 600) {
    this.httpStatusType = "serverError";
  }
}

myLoadVars.onData = function(src:String) {
  trace(">> " + this.httpStatusType + ": " + this.httpStatus);
  if(src != undefined) {
    this.decode(src);
    this.loaded = true;
    this.onLoad(true);
  }
  else {
    this.onLoad(false);
  }
}

myLoadVars.onLoad = function(success:Boolean) {
}

myLoadVars.load("http://blogs.adobe.com/mxna/flashservices/
  getMostRecentPosts.cfm");
```

関連項目

onHTTPStatus (XML.onHTTPStatus ハンドラ), load (LoadVars.load メソッド), sendAndLoad (LoadVars.sendAndLoad メソッド)

onLoad (LoadVars.onLoad ハンドラ)

```
onLoad = function(success:Boolean) {}
```

LoadVars.load() 処理または LoadVars.sendAndLoad() 処理が完了したときに呼び出されます。操作が成功した場合は、その操作によってダウンロードされた変数が my_lv に格納されています。これらの変数は、このハンドラが呼び出されると使用できるようになります。

このハンドラはデフォルトでは定義されていません。

このイベントハンドラは XML.onLoad と似ています。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

success:Boolean - ブール値。ロード処理が正常に完了した場合は true、正常に完了しなかった場合は false に設定されます。

例

次の例では、TextInput インスタンス name_ti、TextArea インスタンス result_ta、Button インスタンス submit_button をステージに追加します。ユーザーが Login ボタンのインスタンスをクリックすると、send_lv と result_lv の 2 つの LoadVars オブジェクトが作成されます。send_lv オブジェクトは name_ti インスタンスから名前をコピーし、データを greeting.cfm に送信します。このスクリプトの結果が result_lv オブジェクトにロードされ、サーバーの応答が TextArea インスタンス (result_ta) に表示されます。タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var submitListener:Object = new Object();
submitListener.click = function(evt:Object) {
  var result_lv:LoadVars = new LoadVars();
  result_lv.onLoad = function(success:Boolean) {
    if (success) {
      result_ta.text = result_lv.welcomeMessage;
    } else {
      result_ta.text = "Error connecting to server.";
    }
  };
  var send_lv:LoadVars = new LoadVars();
  send_lv.name = name_ti.text;
  send_lv.sendAndLoad("http://www.flash-mx.com/mm/greeting.cfm", result_lv,
  "POST");
};
submit_button.addEventListener("click", submitListener);
```

別の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Login" フォルダに移動して login.fla ファイルにアクセスします。

関連項目

onLoad (XML.onLoad ハンドラ), loaded (LoadVars.loaded プロパティ), load (LoadVars.load メソッド), sendAndLoad (LoadVars.sendAndLoad メソッド)

send (LoadVars.send メソッド)

```
public send(url:String, target:String, [method:String]) : Boolean
```

my_lv オブジェクト内の変数を、指定された URL に送信します。*my_lv* 内の列挙可能なすべての変数がデフォルトで *application/x-www-form-urlencoded* 形式のストリングとして 1 つに連結され、HTTP の POST メソッドを使用して URL に送信されます。これは、loadVariables() で使用されるものと同じ形式です。HTTP リクエストヘッダで送信される MIME コンテンツタイプは、my_lv.contentType の値、またはデフォルトの *application/x-www-form-urlencoded* です。GET を指定しなければ、POST メソッドが使用されます。

指定した URL のスクリプトまたはアプリケーションが確実に実行されるように、target パラメータを指定する必要があります。target パラメータを省略すると、true が返されますが、スクリプトまたはアプリケーションは実行されません。

次の場合は、send() メソッドが便利です。

- サーバー応答で SWF の内容を置き換える (target パラメータとして "_self" を使用)
- サーバー応答を新しいウィンドウに表示する (target パラメータとして "_blank" を使用)
- サーバー応答を親フレームまたは最上位のフレームに表示する (target パラメータとして "_parent" または "_top" を使用)
- サーバー応答を指定したフレームに表示する (フレーム名を target パラメータのストリングとして使用)

send() メソッドの呼び出しが成功すると常に、新しいブラウザウィンドウが開くか、既存のウィンドウまたはフレームの内容が置き換わります。情報をサーバーに送り、新しいウィンドウを開いたり、ウィンドウやフレームの内容を置き換えたりせずに、SWF ファイルの再生を続行する場合は、LoadVars.sendAndLoad() を使用してください。

このメソッドは XML.send() と似ています。

Flash のテスト環境では、常に GET メソッドを使用します。POST メソッドを使ってテストする場合は、ブラウザ内で使用してください。

このメソッドを使用するときは、Flash Player セキュリティモデルを考慮してください。

- Flash Player 8 では、呼び出し元 SWF ファイルが信頼されないコードとしてローカルのサンドボックスに置かれている場合、このメソッドは使用できません。
- Flash Player 7 以降では、呼び出し元 SWF ファイルがローカルファイルである場合、このメソッドは使用できません。

詳細については、次の参照先を参照してください。

- 『ActionScript 2.0 の学習』の「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

`url:String` - ストリング。変数のアップロード先の URL。

`target:String` - ストリング。応答を表示するブラウザのウィンドウまたはフレーム。特定のウィンドウの名前を入力するか、次の予約されたターゲット名から選択します。

- `"_self"` は、現在のウィンドウ内の現在のフレームを指定します。
- `"_blank"` は、新規ウィンドウを指定します。
- `"_parent"` は、現在のフレームの親を指定します。
- `"_top"` 現在のウィンドウ内の最上位のフレームを指定します。

`method:String ( オプション )` - ストリング。HTTP プロトコルの `GET` メソッドまたは `POST` メソッド。デフォルト値は `POST` です。

戻り値

`Boolean` - ブール値。パラメータが指定されない場合は `false`、それ以外の場合は `true` を返します。

例

次の例では、テキストフィールドから 2 つの値をコピーし、その情報を処理する CFM スクリプトに送信します。スクリプトでは、ユーザーがハイスコアを出したかどうかをチェックして、そのデータをデータベーステーブルに挿入するなどの処理を実行します。

```
var my_lv:LoadVars = new LoadVars();
my_lv.playerName = playerName_txt.text;
my_lv.playerScore = playerScore_txt.text;
my_lv.send("setscore.cfm", "_blank", "POST");
```

関連項目

`sendAndLoad ( LoadVars.sendAndLoad メソッド ), send ( XML.send メソッド )`

sendAndLoad (LoadVars.sendAndLoad メソッド)

```
public sendAndLoad(url:String, target:Object, [method:String]) : Boolean
```

my_lv オブジェクト内の変数を、指定された URL に送信 (Post) します。サーバーの応答がダウンロードされ、変数データとして解析され、結果の変数が target オブジェクトに挿入されます。

変数の送信方法は、LoadVars.send() の場合と同じです。変数を target にダウンロードする方法は LoadVars.load() と同じです。

このメソッドを使用するときは、Flash Player セキュリティモデルを考慮してください。

Flash Player 8 :

- 呼び出し元 SWF ファイルが ローカルファイルシステムのサンドボックスにあり、ターゲットリソースがネットワーク上のサンドボックスにある場合、データをロードできません。
- 呼び出し元 SWF ファイルがネットワーク上のサンドボックスにあり、ターゲットリソースがローカルにある場合、データをロードできません。

詳細については、次の参照先を参照してください。

- 『ActionScript 2.0 の学習』の「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

Flash Player 7 以降 :

- Web サイトでクロスドメインポリシーファイルを使用して、リソースへのクロスドメインアクセスを許可できます。
- Flash Player 7 以降で SWF ファイルを実行している場合、url はまったく同じドメインに属している必要があります。たとえば www.someDomain.com に置かれている SWF ファイルは、www.someDomain.com に置かれているソースからのみデータをロードできます。

Flash Player 7 より前のバージョンの Player で SWF ファイルを実行している場合、url は、呼び出し元の SWF ファイルと同じスーパードメインに属している必要があります。スーパードメインは、ファイルの URL の一番左の要素を削除することによって派生されます。たとえば、www.someDomain.com に存在する SWF ファイルは、store.someDomain.com に存在するソースからデータをロードできます。これは、どちらのファイルも同じスーパードメイン someDomain.com に属しているためです。

このメソッドは XML.sendAndLoad() と似ています。

対応バージョン : ActionScript 1.0、Flash Player 6 - Flash Player 7 ではビヘイビアが変更されました。

パラメータ

`url:String` - ストリング。変数のアップロード先の URL。呼び出し元の SWF ファイルが Web ブラウザ内で実行している場合、`host` は SWF ファイルと同じドメインに属している必要があります。

`target:Object` - ダウンロードされる変数を受け取る LoadVars オブジェクトまたは XML オブジェクト。

`method:String` (オプション) - ストリング。HTTP プロトコルの `GET` メソッドまたは `POST` メソッド。デフォルト値は `POST` です。

戻り値

`Boolean` - ブール値。

例

次の例では、TextInput インスタンス `name_ti`、TextArea インスタンス `result_ta`、Button インスタンス `submit_button` をステージに追加します。ユーザーがこの例の Login ボタンインスタンスをクリックすると、`send_lv` および `result_lv` の 2 つの LoadVars オブジェクトが作成されます。`send_lv` オブジェクトは `name_ti` インスタンスから名前をコピーし、データを greeting.cfm に送信します。このスクリプトの結果が `result_lv` オブジェクトにロードされ、サーバーの応答が TextArea インスタンス (`result_ta`) に表示されます。この例を確認するには、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var submitListener:Object = new Object();
submitListener.click = function(evt:Object) {
  var result_lv:LoadVars = new LoadVars();
  result_lv.onLoad = function(success:Boolean) {
  if (success) {
    result_ta.text = result_lv.welcomeMessage;
  } else {
    result_ta.text = "Error connecting to server.";
  }
  };
  var send_lv:LoadVars = new LoadVars();
  send_lv.name = name_ti.text;
  send_lv.sendAndLoad("http://www.flash-mx.com/mm/greeting.cfm", result_lv,
  "POST");
};
submit_button.addEventListener("click", submitListener);
```

別の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Login" フォルダに移動して login.fla ファイルにアクセスします。

関連項目

send (LoadVars.send メソッド), load (LoadVars.load メソッド), sendAndLoad (XML.sendAndLoad メソッド)

toString (LoadVars.toString メソッド)

```
public toString() : String
```

my_lv 内の列挙可能な変数をすべて含むストリングを、MIME コンテンツエンコード *application/x-www-form-urlencoded* で返します。

対応バージョン : ActionScript 1.0、Flash Player 6

戻り値

String - ストリング。

例

次の例では、新しい LoadVars() オブジェクトをインスタンス化し、2 つのプロパティを作成します。次に、toString() を使用して、両方のプロパティを含む URL エンコード形式のストリングを返します。

```
var my_lv:LoadVars = new LoadVars();
my_lv.name = "Gary";
my_lv.age = 26;
trace (my_lv.toString()); //output: age=26&name=Gary
```

LocalConnection

```
Object
  |
  +-LocalConnection
```

```
public dynamic class LocalConnection
extends Object
```

LocalConnection クラスを使用すると、fscommand() または JavaScript を使用しなくても相互に指示を送ることができる SWF ファイルを作成できます。LocalConnection オブジェクトを使って通信できるのは、同じクライアントコンピュータ上で実行中の SWF ファイルだけです。ただし、これらは異なるアプリケーションで実行されていてもかまいません。たとえば、ブラウザで実行されている SWF ファイルと、プロジェクタで実行されている SWF ファイルとの間で通信することもできます。LocalConnection オブジェクトを使って 1 つの SWF ファイル内でデータを送受信することもできますが、標準的な方法ではありません。このセクションの例では、異なる SWF ファイル間で通信する方法を示します。

データを送受信するには、主に LocalConnection.send() メソッドと LocalConnection.connect() メソッドを使用します。基本的に、コード内では次のコマンドを実装します。ここで示しているように、LocalConnection.send() コマンドと LocalConnection.connect() コマンドの両方で同じ接続名 lc_name を指定します。

```
// Code in the receiving SWF file
this.createTextField("result_txt", 1, 10, 10, 100, 22);
result_txt.border = true;
var receiving_lc:LocalConnection = new LocalConnection();
receiving_lc.methodToExecute = function(param1:Number, param2:Number) {
  result_txt.text = param1+param2;
};
receiving_lc.connect("lc_name");

// Code in the sending SWF file
var sending_lc:LocalConnection = new LocalConnection();
sending_lc.send("lc_name", "methodToExecute", 5, 7);
```

LocalConnection オブジェクトの最も簡単な使用方法は、同じドメイン内の LocalConnection オブジェクト間だけで通信することです。この場合、セキュリティに関する問題への対処は不要です。しかし、異なるドメイン間で通信を行う必要がある場合は、セキュリティ対策を実施する必要があります。いくつかの方法があります。詳細については、LocalConnection.send() の connectionName パラメータの説明、LocalConnection.allowDomain、および LocalConnection.domain() を参照してください。

対応バージョン： ActionScript 1.0、Flash Player 6

プロパティ一覧

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベントの一覧

イベント	説明
allowDomain = function ([sendingDomain: String]) {}	receiving_lc が送信側 LocalConnection オブジェクトからメソッドを呼び出す要求を受け取ったときに呼び出されます。
allowInsecureDomain = function ([sendingDomain: String]) {}	セキュアなプロトコル (HTTPS) を使用するドメインにホストされている SWF ファイルに存在する receiving_lc が、セキュアでないプロトコルでホストされている SWF ファイルに存在する送信側 LocalConnection オブジェクトからメソッドの呼び出し要求を受け取ったときに呼び出されます。
onStatus = function (infoObject:Object) {}	送信側 LocalConnection オブジェクトが、受信側 LocalConnection オブジェクトにコマンドを送信しようとした後に呼び出されます。

コンストラクター覧

署名	説明
LocalConnection()	LocalConnection オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	close() : Void	LocalConnection オブジェクトを閉じます (切断します)。
	connect (connectionName: String) : Boolean	LocalConnection.send() コマンド (送信側 LocalConnection オブジェクト) からのコマンドを受け取るように LocalConnection オブジェクトを準備します。
	domain() : String	現在の SWF ファイルが存在するドメインを表すストリングを返します。
	send(connectionName: String, methodName:String, [args:Object]) : Boolean	LocalConnection.connect(connectionName) コマンド (受信側 LocalConnection オブジェクト) で確立した接続を使用して、method で指定されたメソッドを呼び出します。

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

allowDomain (LocalConnection.allowDomain ハンドラ)

```
allowDomain = function([sendingDomain:String]) {}
```

receiving_lc が送信側 LocalConnection オブジェクトからメソッドを呼び出す要求を受け取ったときに呼び出されます。このハンドラに記述するコードからは、ブール値の true または false を返す必要があります。このハンドラから true を返さない場合、送信側オブジェクトからの要求は無視され、メソッドは呼び出されません。

このイベントハンドラがない場合、Flash Player は次のコードと同じデフォルトのセキュリティポリシーを適用します。

```
my_lc.allowDomain = function (sendingDomain)
{
  return (sendingDomain == this.domain());
}
```

LocalConnection.allowDomain を使用することにより、特定のドメインまたはすべてのドメインからの LocalConnection オブジェクトに対して、受信側 LocalConnection オブジェクトのメソッドを実行することを明示的に許可できます。sendingDomain パラメータを宣言しない場合、通常はすべてのドメインからのコマンドを受け入れることになります。その場合、このハンドラのコードは return true のみとなります。sendingDomain パラメータを宣言した場合は、sendingDomain の値と、コマンドを受け取るドメインとを比較します。次の例では、両方の実装を示します。

ファイルが Flash Player 6 向けに作成されている場合、sendingDomain パラメータには呼び出し元のスーパードメインが格納されます。ファイルが Flash Player 7 以降向けに作成されている場合、sendingDomain パラメータには呼び出し元のドメインが格納されます。後者の場合、www.domain.com または store.domain.com のいずれかでホストされた SWF ファイルからのアクセスを許可するには、両方のドメインからのアクセスを明示的に許可する必要があります。

```
// For Flash Player 6
receiving_lc.allowDomain = function(sendingDomain) {
  return(sendingDomain=="domain.com");
}
// For Flash Player 7 or later
receiving_lc.allowDomain = function(sendingDomain) {
  return(sendingDomain=="www.domain.com" ||
    sendingDomain=="store.domain.com");
}
```

また、ファイルが Flash Player 7 以降用向けに作成されている場合、このメソッドを使用して、セキュアでないプロトコルでホストされている SWF ファイルから、セキュアなプロトコル (HTTPS) を使用してホストされている SWF ファイルへのアクセスを許可することはできません。この場合は、`LocalConnection.allowInsecureDomain` イベントハンドラを使用してください。

場合によっては、次の状況が発生することがあります。子 SWF ファイルを別のドメインからロードしたとします。子 SWF ファイルから親 SWF ファイルへ LocalConnection 呼び出しを実行できるように、このメソッドを実装しようと思っていますが、子 SWF ファイルを派生する最終的なドメインが不明なことがあります。このような状況は、たとえば、ロードバランシングリダイレクトやサードパーティ製サーバーを使用する場合に発生します。

この状況では、このメソッドの実装に `MovieClip._url` プロパティを使用できます。たとえば、SWF ファイルを my_mc にロードした場合は、ドメインパラメータが `my_mc._url` のドメインに一致するかどうかをチェックすることにより、このメソッドを実装できます。`my_mc._url` に含まれている完全な URL からドメインを解析する必要があります。

この場合、`my_mc` の SWF ファイルがロードされるまで待つようにしてください。ファイルが完全にロードされるまで、_url プロパティが最終的な正しい値に設定されないためです。子 SWF ファイルのロードが完了したかどうか確認するには、`MovieClipLoader.onLoadComplete` を使用するのが最もよいでしょう。

逆の状況もあり得ます。親からの LocalConnection 呼び出しを受け入れる子 SWF ファイルを作成しようとする場合に、親のドメインが不明であることがあります。この場合は、ドメインパラメータが _parent._url のドメインに一致するかどうかをチェックすることにより、このメソッドを実装できます。この場合も、_parent._url に含まれている完全な URL からドメインを解析する必要があります。この状況では、親 SWF ファイルがロードされるまで待つ必要はありません。親 SWF ファイルは子 SWF ファイルがロードされた時点で既にロードされているからです。

対応バージョン： ActionScript 1.0、Flash Player 7

パラメータ

`sendingDomain:String` (オプション) - 送信側 LocalConnection オブジェクトが格納されている SWF ファイルのドメインを指定するストリングです。

例

次の例では、受信側 SWF ファイルの LocalConnection オブジェクトが、すべてのドメインの SWF ファイルに対してメソッドの呼び出しを許可します。この例を LocalConnection.connect() の例と比較してください。後者では、同じドメイン内の SWF ファイルだけが受信側 SWF ファイルの trace() メソッドを呼び出すことができます。接続名でのアンダースコア (_) の使用方法については、LocalConnection.send() を参照してください。

```
this.createTextField("welcome_txt", this.getNextHighestDepth(), 10, 10, 100,
  20);
var my_lc:LocalConnection = new LocalConnection();
my_lc.allowDomain = function(sendingDomain:String) {
  domain_txt.text = sendingDomain;
  return true;
};
my_lc.allowInsecureDomain = function(sendingDomain:String) {
  return (sendingDomain == this.domain());
}
my_lc.sayHello = function(name:String) {
  welcome_txt.text = "Hello, "+name;
};
my_lc.connect("_mylc");
```

次の例では、前の SWF ファイルにストリングを送り、ローカル接続がそのファイルに接続できたかどうかについてステータスメッセージを表示します。name_ti という TextInput コンポーネント、status_ta という TextArea インスタンス、および send_button という Button インスタンスを使用してコンテンツを表示します。

```
var sending_lc:LocalConnection;
var sendListener:Object = new Object();
sendListener.click = function(evt:Object) {
  sending_lc = new LocalConnection();
  sending_lc.onStatus = function(infoObject:Object) {
  switch (infoObject.level) {
  case 'status' :
    status_ta.text = "LocalConnection connected successfully.";
    break;
  case 'error' :
    status_ta.text = "LocalConnection encountered an error.";
    break;
  }
  };
  sending_lc.send("_mylc", "sayHello", name_ti.text);
};
send_button.addEventListener("click", sendListener);
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

次の例では、受信側 SWF ファイルは thisDomain.com にあり、thisDomain.com または thatDomain.com 内の SWF ファイルからのコマンドだけを受け入れます。

```
var aLocalConn:LocalConnection = new LocalConnection();
aLocalConn.Trace = function(aString) {
  aTextField += aString+newline;
};
aLocalConn.allowDomain = function(sendingDomain) {
  return (sendingDomain == this.domain() || sendingDomain == "www.adobe.com");
};
aLocalConn.connect("_mylc");
```

Flash Player 7 以降用にパブリッシュされた場合、ドメイン完全一致が使用されます。つまり、この例では、SWF ファイルが www.thatDomain.com にある場合は失敗し、thatDomain.com にある場合は動作することになります。

関連項目

connect (LocalConnection.connect メソッド),domain (LocalConnection.domain メソッド),send (LocalConnection.send メソッド),_url (MovieClip._url プロパティ), onLoadComplete (MovieClipLoader.onLoadComplete イベントリスナー),_parent プロパティ

allowInsecureDomain (LocalConnection.allowInsecureDomain ハンドラ)

```
allowInsecureDomain = function([sendingDomain:String ]) {}
```

セキュアなプロトコル (HTTPS) を使用するドメインにホストされている SWF ファイルに存在する receiving_lc が、セキュアでないプロトコルでホストされている SWF ファイルに存在する送信側 LocalConnection オブジェクトからメソッドの呼び出し要求を受け取ったときに呼び出されます。このハンドラに記述するコードからは、ブール値の true または false を返す必要があります。このハンドラから true を返さない場合、送信側オブジェクトからの要求は無視され、メソッドは呼び出されません。

デフォルトでは、HTTPS プロトコルを使用してホストされている SWF ファイルは、HTTPS プロトコルを使用してホストされている SWF ファイルにのみアクセスできます。この実装方法により、HTTPS プロトコルが提供する整合性が保たれます。

このメソッドでデフォルトのビヘイビアを変更することはお勧めできません。デフォルトのビヘイビアを変更すると、HTTPS のセキュリティが損なわれます。ただし、デフォルトのビヘイビアを変更する必要がある場合もあります。たとえば、Flash Player 7 以降用にパブリッシュされた HTTPS ファイルに対して、Flash Player 6 用にパブリッシュされた HTTP ファイルからのアクセスを許可する必要がある場合などです。

Flash Player 6 用にパブリッシュされた SWF ファイルの場合、LocalConnection.allowDomain イベントハンドラを使用して HTTP から HTTPS へのアクセスを許可できます。ただし、Flash Player 7 の場合はセキュリティの実装方法が異なります。Flash Player 7 以降用にパブリッシュされた SWF ファイルでこのようなアクセスを許可するためには、LocalConnection.allowInsecureDomain() を使用する必要があります。

対応バージョン： ActionScript 1.0、Flash Player 7

パラメータ

sendingDomain:String (オプション) - 送信側 LocalConnection オブジェクトが格納されている SWF ファイルのドメインを指定するストリングです。

例

次の例では、現在のドメインまたは www.adobe.com からの接続を許可するか、セキュアでない接続を現在のドメインからのみ許可します。

```
this.createTextField("welcome_txt", this.getNextHighestDepth(), 10, 10, 100,
  20);
var my_lc:LocalConnection = new LocalConnection();
my_lc.allowDomain = function(sendingDomain:String) {
  domain_txt.text = sendingDomain;
  return (sendingDomain == this.domain() || sendingDomain == "www.adobe.com");
};
my_lc.allowInsecureDomain = function(sendingDomain:String) {
  return (sendingDomain == this.domain());
}
my_lc.sayHello = function(name:String) {
  welcome_txt.text = "Hello, "+name;
};
my_lc.connect("lc_name");
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

allowDomain (LocalConnection.allowDomain ハンドラ),
connect (LocalConnection.connect メソッド)

close (LocalConnection.close メソッド)

```
public close() : Void
```

LocalConnection オブジェクトを閉じます (切断します)。このコマンドは、LocalConnection オブジェクトでそれ以上コマンドを受け入れない場合に実行します。たとえば、他の SWF ファイルで同じ connectionName パラメータを使用して LocalConnection.connect() コマンドを実行する場合などです。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、close_button という Button コンポーネントインスタンスがクリックされると、receiving_lc という接続を閉じます。

```
this.createTextField("welcome_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
this.createTextField("status_txt", this.getNextHighestDepth(), 10, 42, 100,44);

var receiving_lc:LocalConnection = new LocalConnection();
receiving_lc.sayHello = function(name:String) {
  welcome_txt.text = "Hello, "+name;
};
receiving_lc.connect("lc_name");
var closeListener:Object = new Object();
closeListener.click = function(evt:Object) {
  receiving_lc.close();
  status_txt.text = "connection closed";
};
close_button.addEventListener("click", closeListener);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

connect (LocalConnection.connect メソッド)

connect (LocalConnection.connect メソッド)

```
public connect(connectionName:String) : Boolean
```

LocalConnection.send() コマンド (送信側 LocalConnection オブジェクト) からのコマンドを受け取るように LocalConnection オブジェクトを準備します。このコマンドを使用するオブジェクトは " 受信側 LocalConnection オブジェクト " と呼ばれます。受信側と送信側のオブジェクトは、同じクライアントコンピュータ上で実行する必要があります。

このセクションのすべての例に示すように、このメソッドを呼び出す前に、*receiving_lc* に関連付けられたメソッドを定義する必要があります。

デフォルトでは、Flash Player は connectionName を解決して " *superdomain* :connectionName" にします。ここで、*superdomain* は、LocalConnection.connect() コマンドが格納された SWF ファイルのスーパードメインを表します。たとえば、受信側 LocalConnection オブジェクトを含む SWF ファイルが www.someDomain.com に配置されている場合、connectionName は "someDomain.com:connectionName" に解決されます。SWF ファイルがクライアントコンピュータに配置されている場合、superdomain には "localhost" が割り当てられます。

またデフォルトでは、受信側 LocalConnection オブジェクトは、接続名が " *superdomain* : connectionName" の値に解決される送信側 LocalConnection オブジェクトからのコマンドだけを受け入れます。このため、同じドメインに配置されている SWF ファイル間の通信は簡単です。

同じドメインの SWF ファイル間でのみ通信を実装する場合は、connectionName に対して、先頭がアンダースコア (_) 以外で、ドメイン名が指定されていないストリングを指定します ("myDomain:connectionName" など)。LocalConnection.connect(connectionName) コマンドと同じストリングを使用してください。

異なるドメイン内の SWF ファイル間の通信を実装する場合は、*connectionName* にアンダースコア (_) で始まるストリングを指定すると、受信側 LocalConnection オブジェクトを含む SWF のドメイン間でのポータビリティが高まります。考えられる 2 つの状況を次に示します。

- *connectionName* のストリングがアンダースコア (_) で始まっていない場合、Flash Player により、"myDomain:connectionName" のように、スーパードメインとコロンの接頭辞が追加されます。これにより、他のドメインの同じ名前を持つ接続とのコンフリクトは回避できますが、すべての送信側 LocalConnection オブジェクトは、"myDomain:connectionName" のようにこのスーパードメインを指定する必要があります。受信側 LocalConnection オブジェクトを含む SWF が別のドメインに移動された場合は、"anotherDomain:connectionName" のように、新しいスーパードメインに応じて接頭辞が変更されます。すべての送信側 LocalConnection オブジェクトは、新しいスーパードメインを参照するように手動で編集する必要があります。

- *connectionName* のストリングが、"_connectionName" のようにアンダースコアで始まっている場合、ストリングに接頭辞は追加されません。つまり、受信側および送信側 LocalConnection オブジェクトは、connectionName にまったく同じストリングを使用します。受信側オブジェクトが LocalConnection.allowDomain を使用し、すべてのドメインからの通信を受け入れるように指定した場合は、送信側 LocalConnection オブジェクトを変更せずに、受信側 LocalConnection オブジェクトを含む SWF を別のドメインに移動させることができます。

詳細については、LocalConnection.send() の connectionName の説明、LocalConnection.allowDomain、および LocalConnection.domain() を参照してください。

メモ：コロンは、connectionName のストリングとスーパードメインを区切る特殊文字として使用されます。connectionName にコロンを含むストリングを指定するのは無効です。

対応バージョン：ActionScript 1.0、Flash Player 6

パラメータ

connectionName: String - *receiving_lc* と通信する LocalConnection.send() コマンドで指定した接続名に対応するストリング。

戻り値

Boolean - ブール値。同じクライアントコンピュータ上に、*connectionName* パラメータに同じ値を使用して既にこのコマンドを実行しているプロセスがない場合は true、それ以外の場合は false を返します。

例

次の例では、特定のドメイン内の SWF ファイルから、同じドメイン内の受信側 SWF ファイルのメソッド printOut を呼び出します。

まず、次のコードが含まれる SWF ファイルを作成します。

```
this.createTextField("tf", this.getNextHighestDepth(), 10, 10, 300, 100);
var aLocalConnection:LocalConnection = new LocalConnection();
aLocalConnection.connect("demoConnection");
aLocalConnection.printOut = function(aString:String):Void{
  tf.text += aString;
}
```

さらに、次のコードが含まれる 2 番目の SWF ファイルを作成します。

```
var sending_lc:LocalConnection = new LocalConnection();
sending_lc.send("demoConnection", "printOut", "This is a message from file B.
  Hello.");
```

この例をテストするには、1 番目の SWF ファイルを実行してから、2 番目の SWF ファイルを実行します。

別の例を示します。SWF 1 には次のコードが含まれます。このコードは、実行時に MP3 ファイルを再生する新しい Sound オブジェクトを作成します。playback_pb という ProgressBar に MP3 ファイルの再生状況を表示します。song_lbl という Label コンポーネントインスタンスに MP3 ファイルの名前を表示します。異なる SWF ファイルのボタンを使用して、LocalConnection オブジェクトによる再生を制御します。

```
var playback_pb:mx.controls.ProgressBar;
var my_sound:Sound;
playback_pb.setStyle("themeColor", "haloBlue");
this.createEmptyMovieClip("timer_mc", this.getNextHighestDepth());
var receiving_lc:LocalConnection = new LocalConnection();
receiving_lc.playMP3 = function(mp3Path:String, mp3Name:String) {
  song_lbl.text = mp3Name;
  playback_pb.indeterminate = true;
  my_sound = new Sound();
  my_sound.onLoad = function(success:Boolean) {
  playback_pb.indeterminate = false;
  };
  my_sound.onSoundComplete = function() {
  delete timer_mc.onEnterFrame;
  };
  timer_mc.onEnterFrame = function() {
  playback_pb.setProgress(my_sound.position, my_sound.duration);
  };
  my_sound.loadSound(mp3Path, true);
};
receiving_lc.connect("lc_name");
```

SWF 2 には、play_btn というボタンがあります。ボタンがクリックされると、SWF 1 に接続して 2 つの変数を渡します。最初の変数にはストリーミングする MP3 ファイルが格納されます。2 番目の変数は SWF 1 の Label コンポーネントインスタンスに表示されるファイル名です。

```
play_btn.onRelease = function() {
  var sending_lc:LocalConnection = new LocalConnection();
  sending_lc.send("lc_name", "playMP3", "song1.mp3", "Album - 01 - Song");
};
```

SWF 3 には、play_btn というボタンがあります。ボタンがクリックされると、SWF 1 に接続して 2 つの変数を渡します。最初の変数にはストリーミングする MP3 ファイルが格納されます。2 番目の変数は SWF 1 の Label コンポーネントインスタンスに表示されるファイル名です。

```
play_btn.onRelease = function() {
  var sending_lc:LocalConnection = new LocalConnection();
  sending_lc.send("lc_name", "playMP3", "song2.mp3", "Album - 02 - Another
  Song");
};
```

これらの例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

send (LocalConnection.send メソッド), allowDomain (LocalConnection.allowDomain ハンドラ), domain (LocalConnection.domain メソッド)

domain (LocalConnection.domain メソッド)

```
public domain() : String
```

現在の SWF ファイルが存在するドメインを表すストリングを返します。

Flash Player 6 用にパブリッシュされた SWF では、現在の SWF ファイルのスーパードメインを示すストリングを返します。たとえば、現在の SWF ファイルが www.adobe.com にある場合は、"adobe.com" を返します。

Flash Player 7 以降用にパブリッシュされた SWF では、現在の SWF ファイルのドメインを示すストリングを返します。たとえば、現在の SWF ファイルが www.adobe.com にある場合は、"www.adobe.com" を返します。

現在の SWF ファイルがクライアントコンピュータ上のローカルファイルである場合は、"localhost" を返します。

このコマンドの最も一般的な使用方法は、呼び出す受信側 LocalConnection オブジェクトのメソッドのパラメータとして、送信側 LocalConnection オブジェクトのドメイン名を渡すか、LocalConnection.allowDomain と組み合わせて、特定のドメインのコマンドを受け入れるというものです。同じドメイン内にある LocalConnection オブジェクト間でのみ通信する場合は、通常、このコマンドを使用する必要はありません。

対応バージョン : ActionScript 1.0、Flash Player 6 - Flash Player 7 ではビヘイビアが変更されました。

戻り値

String - 現在の SWF ファイルが配置されているドメインを示すストリング。詳細については、「説明」を参照してください。

例

次の例では、受信側 SWF ファイルは同じドメインまたは adobe.com に配置されている SWF ファイルからのコマンドのみを受け付けます。

```
// If both the sending and receiving SWF files are Flash Player 6,
// then use the superdomain
var my_lc:LocalConnection = new LocalConnection();
my_lc.allowDomain = function(sendingDomain):String{
  return (sendingDomain==this.domain() || sendingDomain=="adobe.com");
}

// If either the sending or receiving SWF file is Flash Player 7 or later,
// then use the exact domain. In this case, commands from a SWF file posted
// at www.adobe.com will be accepted, but those from one posted at
// a different subdomain, e.g. blogs.adobe.com, will not.
var my_lc:LocalConnection = new LocalConnection();
my_lc.allowDomain = function(sendingDomain):String{
  return (sendingDomain==this.domain() || sendingDomain=="www.adobe.com");
}
```

次の例では、www.yourdomain.com にある送信側 SWF ファイルから、www.mydomain.com にある受信側 SWF ファイルのメソッドを呼び出します。送信側 SWF ファイルは、呼び出すメソッドのパラメータとして自分のドメイン名を含めるので、受信側 SWF ファイルは正しいドメインの LocalConnection オブジェクトに応答を返すことができます。また、送信側 SWF ファイルは、mydomain.com にある SWF ファイルからのコマンドのみを受け入れるように指定します。

ここでは、参考のために行番号を示します。イベントのシーケンスは次のとおりです。

- 受信側 SWF ファイルが "sum" という名前の接続でコマンドを受け取るように準備します (11 行目)。この接続の名前は "mydomain.com:sum" に解決されます。LocalConnection.connect() を参照してください。
- 送信側 SWF ファイルが "result" という LocalConnection オブジェクトで応答を受け取るように準備します (67 行目)。また、mydomain.com に配置されている SWF ファイルからのコマンドだけを受け入れるように指定します (51 ～ 53 行目)。
- 送信側 SWF ファイルで、"mydomain.com:sum" という名前の接続の aSum メソッドを呼び出し (68 行目)、パラメータとして、スーパードメイン、応答を受け取る接続の名前 ("result")、aSum で使用される値 (123 および 456) を渡します。
- aSum メソッド (6 行目) が、sender = "mydomain.com:result"、replyMethod = "aResult"、n1 = 123、n2 = 456 で呼び出されます。その結果、次のコードを実行します。

  ```
  this.send("mydomain.com:result", "aResult", (123 + 456));
  ```

■ aResult メソッド (54 行目) で、aSum から返された値 (579) を表示します。

```
// The receiving SWF at http://www.mydomain.com/folder/movie.swf
// contains the following code

1 var aLocalConnection:LocalConnection = new LocalConnection();
2 aLocalConnection.allowDomain = function()
3 {
  // Allow connections from any domain
4 return true;
5 }
6 aLocalConnection.aSum = function(sender, replyMethod, n1, n2)
7 {
8 this.send(sender, replyMethod, (n1 + n2));
9 }
10
11 aLocalConnection.connect("sum");

// The sending SWF at http://www.yourdomain.com/folder/movie.swf
// contains the following code

50 var lc:LocalConnection = new LocalConnection();
51 lc.allowDomain = function(aDomain) {
     // Allow connections only from mydomain.com
52 return (aDomain == "mydomain.com");
53 }
54 lc.aResult = function(aParam) {
55 trace("The sum is " + aParam);
56 }
  // determine our domain and see if we need to truncate it
57 var channelDomain:String = lc.domain();
58 if (getVersion() >= 7 && this.getSWFVersion() >= 7)
59 {
    // split domain name into elements
60 var domainArray:Array = channelDomain.split(".");

    // if more than two elements are found,
    // chop off first element to create superdomain
61 if (domainArray.length > 2)
62 {
63 domainArray.shift();
64 channelDomain = domainArray.join(".");
65 }
66 }

67 lc.connect("result");
68 lc.send("mydomain.com:sum", "aSum", channelDomain + ':' + "result",
"aResult", 123, 456);
```

関連項目

allowDomain (LocalConnection.allowDomain ハンドラ),connect (LocalConnection.connect メソッド)

LocalConnection コンストラクタ

```
public LocalConnection()
```

LocalConnection オブジェクトを作成します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、受信側と送信側の SWF ファイルで LocalConnnection オブジェクトを作成する方法を示します。2 つの SWF ファイルでは、それぞれの LocalConnection オブジェクトに対して同じ名前を使うことも、異なる名前を使うこともできます。この例では、異なる名前を使用します。

```
// Code in the receiving SWF file
this.createTextField("result_txt", 1, 10, 10, 100, 22);
result_txt.border = true;
var receiving_lc:LocalConnection = new LocalConnection();
receiving_lc.methodToExecute = function(param1:Number, param2:Number) {
  result_txt.text = param1+param2;
};
receiving_lc.connect("lc_name");
```

次の SWF ファイルは、要求を最初の SWF ファイルに送ります。

```
// Code in the sending SWF file
var sending_lc:LocalConnection = new LocalConnection();
sending_lc.send("lc_name", "methodToExecute", 5, 7);
```

関連項目

connect (LocalConnection.connect メソッド),send (LocalConnection.send メソッド)

onStatus (LocalConnection.onStatus ハンドラ)

```
onStatus = function(infoObject:Object) {}
```

送信側 LocalConnection オブジェクトが、受信側 LocalConnection オブジェクトにコマンドを送信しようとした後に呼び出されます。このイベントハンドラに応答するには、LocalConnection オブジェクトから送られる情報オブジェクトを処理する関数を作成する必要があります。

このイベントハンドラから返される情報オブジェクトの level 値として status が含まれる場合は、受信側 LocalConnection オブジェクトにコマンドが正常に送信されたことになります。これは、受信側 LocalConnection オブジェクトの指定のメソッドが正常に呼び出されたということではありません。単に、コマンドを送信できたというだけです。たとえば、受信側 LocalConnection オブジェクトが送信側ドメインからの接続を受け入れない場合、またはそのメソッドが存在しない場合には、メソッドは呼び出されません。メソッドが呼び出されたかどうかを確認するには、受信側オブジェクトが送信側オブジェクトに応答を送信する必要があります。

このイベントハンドラから返される情報オブジェクトに level 値として error が含まれている場合は、受信側 LocalConnection オブジェクトにコマンドを送信できません。このハンドラを呼び出した `sending_lc.send()` コマンドで指定した名前に対応する受信側 LocalConnection オブジェクトが接続されていないからです。

この `onStatus` ハンドラ以外に、Flash には `System.onStatus` という " スーパー " 関数もあります。特定のオブジェクトに対して `onStatus` が呼び出され、それに応答する関数が割り当てられていない場合、`System.onStatus` に割り当てられた関数があれば、その関数が実行されます。

ほとんどの場合、このハンドラは次の例に示すように、エラー状態にのみ対処するように実装します。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

`infoObject:Object` - ステータスメッセージに従って定義されるパラメータ。このパラメータの詳細については、「説明」を参照してください。

例

次の例では、SWF ファイルが `lc_name` という別のローカル接続オブジェクトに接続したどうかに関するステータスメッセージを表示します。`name_ti` という TextInput コンポーネント、`status_ta` という TextArea インスタンス、および `send_button` という Button インスタンスを使用してコンテンツを表示します。

```
var sending_lc:LocalConnection;
var sendListener:Object = new Object();
sendListener.click = function(evt:Object) {
  sending_lc = new LocalConnection();
  sending_lc.onStatus = function(infoObject:Object) {
    switch (infoObject.level) {
    case 'status' :
```

```
        status_ta.text = "LocalConnection connected successfully.";
          break;
      case 'error' :
        status_ta.text = "LocalConnection encountered an error.";
        break;
      }
  };
  sending_lc.send("lc_name", "sayHello", name_ti.text);
};
send_button.addEventListener("click", sendListener);
```

関連項目

send (LocalConnection.send メソッド),onStatus (System.onStatus ハンドラ)

send (LocalConnection.send メソッド)

```
public send(connectionName:String, methodName:String, [args:Object]) : Boolean
```

LocalConnection.connect(connectionName) コマンド (受信側 LocalConnection オブジェクト) で確立した接続を使用して、method で指定されたメソッドを呼び出します。このコマンドを使用するオブジェクトを " 送信側 LocalConnection オブジェクト " といいます。送信側オブジェクトと受信側オブジェクトを含む SWF ファイルは、同じクライアントコンピュータ上で実行する必要があります。

このコマンドにパラメータとして渡すことができるデータの量は 40 KB に制限されます。シンタックスが正しいにもかかわらず、コマンドから false が返される場合は、LocalConnection.send() 要求を複数に分割してそれぞれを 40 KB より小さくしてみてください。

LocalConnection.connect() で説明されているように、デフォルトでは connectionName に現在のスーパードメインが追加されます。異なるドメイン間で通信を行う場合は、送信側と受信側の両方の LocalConnection オブジェクトで、現在のスーパードメインが追加されないように connectionName を定義する必要があります。これには、次の 2 つの方法があります。

- 送信側と受信側の両方の LocalConnection オブジェクトで、connectionName の先頭にアンダースコア (_) を使用します。受信側オブジェクトが格納されている SWF ファイルでは LocalConnection.allowDomain を使用して、すべてのドメインからの接続を受け入れるように指定します。この方法では、送信側と受信側の SWF ファイルを任意のドメインに配置できます。
- 送信側 LocalConnection オブジェクトの connectionName にスーパードメインを含めます。たとえば、myDomain.com:myConnectionName と指定します。受信側オブジェクトでは、LocalConnection.allowDomain を使用して、指定したスーパードメイン (この例では myDomain.com) からの接続を受け入れるか、またはすべてのドメインからの接続を受け入れるように指定します。

メモ : 受信側 LocalConnection オブジェクトの connectionName ではスーパードメインを指定できません。スーパードメインを指定できるのは送信側 LocalConnection オブジェクトだけです。

このメソッドを使用するときは、Flash Player セキュリティモデルを考慮してください。デフォルトでは、LocalConnection オブジェクトはそれを作成した SWF ファイルの Sandbox に関連づけられ、LocalConnection オブジェクトのクロスドメインでの呼び出しは、`LocalConnection.allowDomain()` メソッドが呼び出されていない限り許可されません。

詳細については、次の参照先を参照してください。

- 『ActionScript 2.0 の学習』の「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

`connectionName:String` - *`sending_lc`* と通信する `LocalConnection.connect()` コマンドで指定した接続名に対応するストリング。

`methodName:String` - 受信側 LocalConnection オブジェクト内で呼び出すメソッドの名前を指定するストリング。メソッド `send`、`connect`、`close`、`domain`、`onStatus`、または `allowDomain` を使用すると、このコマンドは失敗します。

`args:Object` (オプション) - 指定されたメソッドに渡される引数。

戻り値

`Boolean` - ブール値。要求を実行できる場合は `true`、それ以外の場合は `false` を返します。

メモ：`true` が返された場合でも、単にコマンドのシンタックスが正しいというだけで、受信側 LocalConnection オブジェクトへの接続に成功したとは限りません。接続が成功したかどうかを調べる方法については、`LocalConnection.onStatus` を参照してください。

例

同じドメインに置かれている LocalConnection オブジェクト間で通信する例については、`LocalConnection.connect()` を参照してください。すべてのドメインの LocalConnection オブジェクト間で通信する例については、`LocalConnection.allowDomain` を参照してください。指定したドメインに置かれている LocalConnection オブジェクト間で通信する例については、`LocalConnection.allowDomain` と `LocalConnection.domain()` を参照してください。

関連項目

`allowDomain (LocalConnection.allowDomain ハンドラ)`, `connect (LocalConnection.connect メソッド)`, `domain (LocalConnection.domain メソッド)`, `onStatus (LocalConnection.onStatus ハンドラ)`

Locale (mx.lang.Locale)

```
Object
  |
  +-mx.lang.Locale

public class Locale
extends Object
```

mx.lang.Locale クラスを使用すると、SWF ファイルで表示する多言語テキストを制御できます。Flash の [ストリング] パネルを使用すると、ダイナミックテキストフィールドでストリングリテラルの代わりにストリング ID を使用できます。これにより、特定言語の XML ファイルからロードしたテキストを表示する SWF ファイルを作成できます。この XML ファイルでは、XLIFF (XML Localization Interchange File Format : XML ローカリゼーション交換ファイル形式) を使用する必要があります。XLIFF ファイルに含まれる特定言語のストリングを表示する方法は 3 とおりあります。

- [実行時に自動で] - Flash Player によって、ストリング ID が、System.capabilities.language から返されるデフォルトのシステム言語と一致する XML ファイルのストリングと置き換えられます。
- [ステージ言語を使用して手動で] - コンパイル時にストリング ID がストリングに置き換えられるので、Flash Player では変更できません。
- [実行時に ActionScript を使用して] - ストリング ID の置き換えを、実行時に ActionScript を使用して制御します。このオプションでは、ストリング ID を置き換えるタイミングと言語を両方とも制御できます。

このクラスのプロパティとメソッドは、[実行時に ActionScript を使用して] という方法でストリング ID を置き換える場合に使用できます。

使用可能なプロパティとメソッドはすべて静的です。つまり、mx.lang.Locale クラスのインスタンスではなく、クラス自体を使ってアクセスします。

メモ : Locale クラスは、『ActionScript 2.0 リファレンスガイド』で説明している他のクラスとは異なり、Flash Player には含まれていません。このクラスは Flash Authoring クラスパスにインストールされており、SWF ファイルに自動的にコンパイルされます。Locale クラスを使用すると、クラスが SWF ファイルにコンパイルされるので、SWF のサイズがわずかに大きくなります。

対応バージョン : ActionScript 2.0、Flash Player 7

プロパティ一覧

オプション	プロパティ	説明
static	autoReplace:Boolean	XML ファイルをロードした後に自動的にストリングを置き換えるかどうかを示します。
static	languageCodeArray:Array (読み取り専用)	指定された言語または FLA ファイルにロードされた言語の言語コードが含まれる配列です。
static	stringIDArray:Array (読み取り専用)	FLA ファイル内のストリング ID がすべて含まれる配列です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

メソッド一覧

オプション	署名	説明
static	addDelayedInstance (instance:Object, stringID:String) : Void	インスタンスとストリング ID のペアを内部配列に追加して、後で使用できるようにします。
static	addXMLPath(langCode: String, path:String) : Void	言語コードと言語パスのペアを内部配列に追加して、後で使用できるようにします。
static	checkXMLStatus() : Boolean	XML ファイルがロードされた場合は true を、それ以外の場合は false を返します。
static	getDefaultLang() : String	[ストリング] パネルのダイアログボックスで設定されたか、setDefaultLang() メソッドを呼び出すことで設定されたデフォルトの言語コード。
static	initialize() : Void	使用言語を自動的に判別し、その言語の XML ファイルをロードします。
static	loadLanguageXML (xmlLanguageCode:String, customXmlCompleteCallback: Function) : Void	指定された言語の XML ファイルをロードします。
static	loadString(id:String) : String	指定されたストリング ID に関連付けられたストリング値を現在の言語で返します。

オプション	署名	説明
static	loadStringEx(stringID:String, languageCode:String) : String	指定されたストリング ID と言語コードに関連付けられたストリング値を返します。
static	setDefaultLang(langCode:String) : Void	デフォルトの言語コードを設定します。
static	setLoadCallback (loadCallback:Function) : Void	XML ファイルがロードされた後に呼び出されるコールバック関数を設定します。
static	setString(stringID:String, languageCode:String, stringValue:String) : Void	指定されたストリング ID と言語コードに関連付ける新しいストリング値を設定します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

addDelayedInstance (Locale.addDelayedInstance メソッド)

public static addDelayedInstance(instance:Object, stringID:String) : Void

インスタンスとストリング ID のペアを内部配列に追加して、後で使用できるようにします。このメソッドは主に、ストリングの置き換え方法が [実行時に自動で] である場合に Flash によって使用されます。

対応バージョン : ActionScript 2.0、Flash Player 7

パラメータ

instance:Object - 設定するテキストフィールドのインスタンス名。

stringID:String - 言語のストリング ID。

例

次の例では、autoReplace プロパティと addDelayedInstance() メソッドを使用して、ステージ上のテキストフィールドに、英語の XML ファイル内の IDS_GREETING ストリングを入力します。

```
import mx.lang.Locale;
greeting_txt.autoSize = "left";
Locale.autoReplace = true;
Locale.addDelayedInstance(greeting_txt, "IDS_GREETING");
Locale.loadLanguageXML("en");
```

addXMLPath (Locale.addXMLPath メソッド)

```
public static addXMLPath(langCode:String, path:String) : Void
```

言語コードと言語パスのペアを内部配列に追加して、後で使用できるようにします。このメソッドは主に、ストリングの置き換え方法が [実行時に自動で] または [実行時に ActionScript を使用して] である場合に Flash によって使用されます。

対応バージョン : ActionScript 2.0、Flash Player 7

パラメータ

`langCode:String` - 言語コード。

`path:String` - 追加する XML パス。

例

次の例では、setInterval() を使用して、特定言語の XML ファイルが正常にロードされたかどうかをチェックします。

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("en");
// create interval to check if language XML file is loaded
var locale_int:Number = setInterval(checkLocaleStatus, 10);
function checkLocaleStatus():Void {
  if (Locale.checkXMLStatus()) {
    clearInterval(locale_int);
    trace("clearing interval @ " + getTimer() + " ms");
  }
}
// callback function for Locale.setLoadCallback()
function localeCallback(success:Boolean):Void {
  greeting_txt.text = Locale.loadString("IDS_GREETING");
}
```

autoReplace (Locale.autoReplace プロパティ)

```
public static autoReplace : Boolean
```

XML ファイルをロードした後に自動的にストリングを置き換えるかどうかを示します。true に設定されている場合、テキスト置き換え方法は [ストリング] パネルの設定 [実行時に自動で] と同じになります。つまり、Flash Player によってホスト環境のデフォルト言語が決定され、自動的にその言語でテキストが表示されます。false に設定されている場合、テキスト置き換え方法は [ストリング] パネルの設定 [実行時に ActionScript を使用して] と同じになります。つまり、開発者が、該当する XML ファイルをロードしてテキストを表示する必要があります。

このプロパティのデフォルト値には、[ストリング] パネルのダイアログボックスの [ストリングの置き換え] の設定が反映されます。[実行時に自動で] (デフォルト設定) の場合は true が設定され、[実行時に ActionScript を使用して] の場合は false が設定されます。

対応バージョン : ActionScript 2.0、Flash Player 8

例

次の例では、Locale.autoReplace プロパティを使用して、ステージ上で動的に作成された greeting_txt テキストフィールドに、英語の XML ファイル内の IDS_GREETING ストリングの内容を入力します。[ストリング] パネルで、[設定] ボタンをクリックして、[設定] ダイアログボックスを表示します。[設定] ダイアログボックスを使用して、2 つのアクティブな言語として英語 (en) およびフランス語 (fr) を追加します。[ストリングの置き換え] ラジオコントロールを [実行時に ActionScript を使用して] に設定し、[OK] をクリックします。最後に、[ストリング] パネルで、IDS_GREETING のストリング ID を入力し、アクティブな言語ごとにテキストを追加します。

```
import mx.lang.Locale;
this.createTextField("greeting_txt", 10, 40, 40, 200, 20);
greeting_txt.autoSize = "left";
Locale.autoReplace = true;
Locale.addDelayedInstance(greeting_txt, "IDS_GREETING");
Locale.loadLanguageXML("en");
```

checkXMLStatus (Locale.checkXMLStatus メソッド)

```
public static checkXMLStatus() : Boolean
```

XML ファイルがロードされた場合は true を、それ以外の場合は false を返します。

対応バージョン : ActionScript 2.0、Flash Player 7

戻り値

Boolean - XML ファイルがロードされた場合は true を、それ以外の場合は false を返します。

例

次の例では、間隔を使用して、特定言語の XML ファイルが正常にロードされたかどうかを 10 ミリ秒ごとにチェックします。XML ファイルがロードされると、ステージ上の greeting_txt テキストフィールドに、その XML ファイル内の IDS_GREETING ストリングが入力されます。

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("en");
// create interval to check if language XML file is loaded
var locale_int:Number = setInterval(checkLocaleStatus, 10);
function checkLocaleStatus():Void {
  if (Locale.checkXMLStatus()) {
    clearInterval(locale_int);
    trace("clearing interval @ " + getTimer() + " ms");
  }
}
// callback function for Locale.setLoadCallback()
function localeCallback(success:Boolean):Void {
  greeting_txt.text = Locale.loadString("IDS_GREETING");
}
```

getDefaultLang (Locale.getDefaultLang メソッド)

```
public static getDefaultLang() : String
```

[ストリング] パネルのダイアログボックスで設定されたか、setDefaultLang() メソッドを呼び出すことで設定されたデフォルトの言語コード。

対応バージョン : ActionScript 2.0、Flash Player 8

戻り値

String - デフォルトの言語コードを返します。

例

次の例では、Flash ドキュメントのデフォルトの初期言語を保持しておくために使用される defLang という変数を作成します。[ストリング] パネルで、[設定] ボタンをクリックして、[設定] ダイアログボックスを表示します。次に、2 つのアクティブな言語として英語 (en) およびフランス語 (fr) を追加します。[ストリングの置き換え] ラジオコントロールを [実行時に ActionScript を使用して] に設定し、[OK] をクリックします。[ストリング] パネルで、ストリング ID IDS_GREETING を追加し、アクティブな言語ごとにテキストを追加します。

```
import mx.lang.Locale;
var defLang:String = "fr";
Locale.setDefaultLang(defLang);
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML(Locale.getDefaultLang());
function localeCallback(success:Boolean) {
  if (success) {
    trace(Locale.stringIDArray); // IDS_GREETING
    trace(Locale.loadString("IDS_GREETING"));
  } else {
    trace("unable to load XML");
  }
}
```

関連項目

setDefaultLang (Locale.setDefaultLang メソッド)

initialize (Locale.initialize メソッド)

```
public static initialize() : Void
```

使用言語を自動的に判別し、その言語の XML ファイルをロードします。このメソッドは主に、ストリングの置き換え方法が [実行時に自動で] である場合に Flash によって使用されます。

対応バージョン : ActionScript 2.0、Flash Player 7

例

次の例では、initialize() メソッドを使用して、ステージ上の greeting_txt テキストフィールドに、ユーザーの現在の OS 言語を入力します。initialize() メソッドを直接使用する代わりに、ストリング置き換え方法である [実行時に自動で] を使用します。

```
import mx.lang.Locale;
trace(System.capabilities.language);
Locale.autoReplace = true;
Locale.addDelayedInstance(greeting_txt, "IDS_GREETING");
Locale.initialize();
```

languageCodeArray (Locale.languageCodeArray プロパティ)

```
public static languageCodeArray : Array ( 読み取り専用 )
```

指定された言語または FLA ファイルにロードされた言語の言語コードが含まれる配列です。言語コードは、アルファベット順にソートされません。

対応バージョン : ActionScript 2.0、Flash Player 8

例

次の例では、ComboBox コンポーネントの現在値に基づいて、特定言語の XML ファイルをロードします。ComboBox コンポーネントをステージまでドラッグし、インスタンス名を lang_cb にします。テキストツールを使用して、ダイナミックテキストフィールドを作成し、インスタンス名を greeting_txt にします。[ストリング] パネルでは、少なくとも 2 つのアクティブな言語を追加し、[ストリングの置き換え] ラジオコントロールを [実行時に ActionScript を使用して] に設定して、[OK] をクリックします。次に、ストリング ID IDS_GREETING を追加し、アクティブな言語ごとにテキストを追加します。最後に、次の ActionScript コードをメインタイムラインのフレーム 1 に追加します。

```
import mx.lang.Locale;
Locale.setLoadCallback(localeListener);
lang_cb.dataProvider = Locale.languageCodeArray.sort();
lang_cb.addEventListener("change", langListener);

function langListener(eventObj:Object):Void {
  Locale.loadLanguageXML(eventObj.target.value);
}
function localeListener(success:Boolean):Void {
  if (success) {
    greeting_txt.text = Locale.loadString("IDS_GREETING");
  } else {
    greeting_txt.text = "unable to load language XML file.";
  }
}
```

loadLanguageXML (Locale.loadLanguageXML メソッド)

```
public static loadLanguageXML(xmlLanguageCode:String,
  customXmlCompleteCallback:Function) : Void
```

指定された言語の XML ファイルをロードします。

対応バージョン : ActionScript 2.0、Flash Player 8

パラメータ

`xmlLanguageCode:String` - ロードする特定言語の XML ファイルの言語コード。

`customXmlCompleteCallback:Function` - 特定言語の XML ファイルのロード時に呼び出されるコールバック関数。

例

次の例では、`loadLanguageXML()` を使用して、英語の XML ファイルをロードします。英語の XML ファイルがロードされると、`localeCallback()` メソッドが呼び出され、`greeting_txt` テキストフィールドに、その XML ファイル内の `IDS_GREETING` ストリングの内容が入力されます。

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("en");
// create interval to check if language XML file is loaded
var locale_int:Number = setInterval(checkLocaleStatus, 10);
function checkLocaleStatus():Void {
  if (Locale.checkXMLStatus()) {
    clearInterval(locale_int);
    trace("clearing interval @ " + getTimer() + " ms");
  }
}
// callback function for Locale.setLoadCallback()
function localeCallback(success:Boolean):Void {
  greeting_txt.text = Locale.loadString("IDS_GREETING");
}
```

loadString (Locale.loadString メソッド)

```
public static loadString(id:String) : String
```

指定されたストリング ID に関連付けられたストリング値を現在の言語で返します。

対応バージョン : ActionScript 2.0、Flash Player 7

パラメータ

`id:String` - ロードするストリングの識別 (ID) 番号。

戻り値

`String` - 指定されたストリング ID に関連付けられたストリング値を現在の言語で返します。

例

次の例では、間隔を使用して、特定言語の XML ファイルが正常にロードされたかどうかを 10 ミリ秒ごとにチェックします。XML ファイルがロードされると、ステージ上の `greeting_txt` テキストフィールドに、その XML ファイル内の `IDS_GREETING` ストリングが入力されます。

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("en");
// create interval to check if language XML file is loaded
var locale_int:Number = setInterval(checkLocaleStatus, 10);
function checkLocaleStatus():Void {
  if (Locale.checkXMLStatus()) {
    clearInterval(locale_int);
    trace("clearing interval @ " + getTimer() + " ms");
  }
}
// callback function for Locale.setLoadCallback()
function localeCallback(success:Boolean):Void {
  greeting_txt.text = Locale.loadString("IDS_GREETING");
}
```

関連項目

`loadStringEx (Locale.loadStringEx メソッド )`

loadStringEx (Locale.loadStringEx メソッド)

```
public static loadStringEx(stringID:String, languageCode:String) : String
```

指定されたストリング ID と言語コードに関連付けられたストリング値を返します。予期しない XML ファイルのローディングを防止するため、loadStringEx() は、特定言語の XML ファイルがロードされていない場合にその XML ファイルをロードしません。特定言語の XML ファイルをロードする場合は、loadLanguageXML() を呼び出す適切なタイミングを決定してください。

対応バージョン : ActionScript 2.0、Flash Player 8

パラメータ

stringID:String - ロードするストリングの識別 (ID) 番号。

languageCode:String - 言語コード。

戻り値

String - languageCode パラメータで指定された言語と指定されたストリング ID に関連付けられたストリング値。

例

次の例では、loadStringEx() メソッドを使用して、現在ロードされているフランス語の XML ファイルに対応する IDS_GREETING ストリングの値を出力します。

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("fr");
function localeCallback(success:Boolean) {
  trace(success);
  trace(Locale.stringIDArray); // IDS_GREETING
  trace(Locale.loadStringEx("IDS_GREETING", "fr")); // bonjour
}
```

関連項目

loadString (Locale.loadString メソッド)

setDefaultLang (Locale.setDefaultLang メソッド)

```
public static setDefaultLang(langCode:String) : Void
```

デフォルトの言語コードを設定します。

対応バージョン : ActionScript 2.0、Flash Player 7

パラメータ

`langCode:String` - 言語コードを表すストリング。

例

次の例では、Flash ドキュメントのデフォルトの初期言語を保持しておくために使用される defLang という変数を作成します。[ストリング] パネルで、[設定] ボタンをクリックして、[設定] ダイアログボックスを表示します。次に、2 つのアクティブな言語として英語 (en) およびフランス語 (fr) を追加します。[ストリングの置き換え] ラジオコントロールを [実行時に ActionScript を使用して] に設定し、[OK] をクリックします。[ストリング] パネルで、ストリング ID IDS_GREETING を追加し、アクティブな言語ごとにテキストを追加します。

```
import mx.lang.Locale;
var defLang:String = "fr";
Locale.setDefaultLang(defLang);
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML(Locale.getDefaultLang());
function localeCallback(success:Boolean) {
  if (success) {
    trace(Locale.stringIDArray); // IDS_GREETING
    trace(Locale.loadString("IDS_GREETING"));
  } else {
    trace("unable to load XML");
  }
}
```

関連項目

getDefaultLang (Locale.getDefaultLang メソッド)

setLoadCallback (Locale.setLoadCallback メソッド)

```
public static setLoadCallback(loadCallback:Function) : Void
```

XML ファイルがロードされた後に呼び出されるコールバック関数を設定します。

対応バージョン : ActionScript 2.0、Flash Player 7

パラメータ

`loadCallback:Function` - 特定言語の XML ファイルのロード時に呼び出されるコールバック関数。

例

次の例では、間隔を使用して、特定言語の XML ファイルが正常にロードされたかどうかを 10 ミリ秒ごとにチェックします。XML ファイルがロードされると、ステージ上の `greeting_txt` テキストフィールドに、その XML ファイル内の `IDS_GREETING` ストリングが入力されます。

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("en");
// create interval to check if language XML file is loaded
var locale_int:Number = setInterval(checkLocaleStatus, 10);
function checkLocaleStatus():Void {
  if (Locale.checkXMLStatus()) {
    clearInterval(locale_int);
    trace("clearing interval @ " + getTimer() + " ms");
  }
}
// callback function for Locale.setLoadCallback()
function localeCallback(success:Boolean):Void {
  greeting_txt.text = Locale.loadString("IDS_GREETING");
}
```

setString (Locale.setString メソッド)

```
public static setString(stringID:String, languageCode:String,
  stringValue:String) : Void
```

指定されたストリング ID と言語コードに関連付ける新しいストリング値を設定します。

対応バージョン : ActionScript 2.0、Flash Player 8

パラメータ

`stringID:String` - 設定するストリングの識別 (ID) 番号。

`languageCode:String` - 言語コード。

`stringValue:String` - ストリング値。

例

次の例では、setString() メソッドを使用して、英語 (en) とフランス語 (fr) の両方の値を IDS_WELCOME ストリングに設定します。

```
import mx.lang.Locale;
Locale.setString("IDS_WELCOME", "en", "hello");
Locale.setString("IDS_WELCOME", "fr", "bonjour");
trace(Locale.loadStringEx("IDS_WELCOME", "en")); // hello
```

stringIDArray (Locale.stringIDArray プロパティ)

public static stringIDArray : Array (読み取り専用)

FLA ファイル内のストリング ID がすべて含まれる配列です。ストリング ID は、アルファベット順にソートされません。

対応バージョン: ActionScript 2.0、Flash Player 8

例

次の例では、現在ロードされている特定言語の XML ファイルの Locale.stringIDArray プロパティを出力します。[ストリング] パネルで、[設定] ボタンをクリックして、[設定] ダイアログボックスを表示します。次に、2 つのアクティブな言語として英語 (en) およびフランス語 (fr) を追加します。[ストリングの置き換え] ラジオコントロールを [実行時に ActionScript を使用して] に設定し、[OK] をクリックします。[ストリング] パネルで、ストリング ID IDS_GREETING を追加し、アクティブな言語ごとにテキストを追加します。

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("fr");
function localeCallback(success:Boolean) {
  trace(success);
  trace(Locale.stringIDArray); // IDS_GREETING
  trace(Locale.loadStringEx("IDS_GREETING", "fr")); // bonjour
}
```

Math

```
Object
  |
  +-Math
```

```
public class Math
extends Object
```

Math クラスはトップレベルのクラスで、コンストラクタを実行しなくても、そのメソッドやプロパティを使用できます。

数学定数および関数にアクセスして処理するには、このクラスのメソッドとプロパティを使用します。Math クラスのプロパティとメソッドはすべて静的であり、`Math.method(parameter)` または `Math.constant` というシンタックスを使用して呼び出す必要があります。ActionScript では、定数は倍精度の IEEE-754 浮動小数の最大精度で定義されます。

Math クラスのいくつかのメソッドは、ラジアン単位の角度をパラメータとして使用します。メソッドを呼び出す前に次の式を使用してラジアン値を計算し、計算した値をパラメータとして指定できます。また、式の右辺全体 (`degrees` には度数で角度を代入) をラジアンパラメータとして指定することもできます。

ラジアン値を計算するには、次の式を使用します。

```
radians = degrees * Math.PI/180
```

次の例では、角度 45 度のサインを計算する式をパラメータとして渡します。

`Math.sin(45 * Math.PI/180)` は `Math.sin(.7854)` と同じです。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

プロパティー覧

オプション	プロパティ	説明
static	E:Number	自然対数の底を表す数学定数で *e* と表記されるものです。
static	LN10:Number	10 の自然対数を表す数学定数で loge10 と表記されるものです。近似値は 2.302585092994046 です。
static	LN2:Number	2 の自然対数を表す数学定数で loge2 と表記されるものです。近似値は 0.6931471805599453 です。
static	LOG10E:Number	10 を底とする定数 e (`Math.E`) の対数を表す数学定数で log10e と表記されるものです。近似値は 0.4342944819032518 です。
static	LOG2E:Number	2 を底とする定数 e (`Math.E`) の対数を表す数学定数で log2e と表記されるものです。近似値は 1.4426950408889633 87 です。

オプション	プロパティ	説明
static	PI:Number	円周と円の直径の比を表す数学定数で pi と表記されるものです。近似値は 3.141592653589793 です。
static	SQRT1_2:Number	1/2 の平方根を表す数学定数です。近似値は 0.7071067811865476 です。
static	SQRT2:Number	2 の平方根を表す数学定数です。近似値は 1.4142135623730951 です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

メソッド一覧

オプション	署名	説明
static	abs(x:Number) : Number	パラメータ x で指定された数値の絶対値を計算して返します。
static	acos(x:Number) : Number	パラメータ x で指定された数値のアークコサイン (逆余弦) を計算してラジアン単位で返します。
static	asin(x:Number) : Number	パラメータ x で指定された数値のアークサイン (逆正弦) を計算してラジアン単位で返します。
static	atan(tangent: Number) : Number	パラメータ tangent で指定された値がタンジェント (正接) の値になる角度を計算してラジアン単位で返します。
static	atan2(y:Number, x:Number) : Number	円の x 軸 (0,0 は円の中心を示します) から反時計回りに測定した場合に、y/x 座標の角度をラジアン単位で計算して返します。
static	ceil(x:Number) : Number	指定された数値または式を切り上げた値を返します。
static	cos(x:Number) : Number	ラジアン単位で指定された角度のコサイン (余弦) を計算して返します。
static	exp(x:Number) : Number	自然対数の底 (*e*) を、パラメータ x で指定された指数で累乗した値を返します。
static	floor(x:Number) : Number	パラメータ x で指定された数値または式を切り捨てた値を返します。
static	log(x:Number) : Number	パラメータ x の自然対数を返します。

オプション	署名	説明
static	max(x:Number, y:Number) : Number	x と y を評価し、大きいほうの値を返します。
static	min(x:Number, y:Number) : Number	x と y を評価し、小さいほうの値を返します。
static	pow(x:Number, y:Number) : Number	x の y 乗を計算して返します。
static	random() : Number	疑似乱数 n を返します (0 <= n < 1)。
static	round(x:Number) : Number	パラメータ x の値を最も近い整数に四捨五入し、値を返します。
static	sin(x:Number) : Number	ラジアン単位で指定された角度のサイン (正弦) を計算して返します。
static	sqrt(x:Number) : Number	指定された数値の平方根を計算して返します。
static	tan(x:Number) : Number	指定された角度のタンジェント (正接) を計算して返します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

abs (Math.abs メソッド)

```
public static abs(x:Number) : Number
```

パラメータ x で指定された数値の絶対値を計算して返します。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

パラメータ

x:Number - 数値。

戻り値

Number - 数値。

例

次の例では、Math.abs( ) を使用して数値の絶対値を返します。x パラメータ (この例では num) の値は変わりません。

```
var num:Number = -12;
var numAbsolute:Number = Math.abs(num);
trace(num); // output: -12
trace(numAbsolute); // output: 12
```

acos (Math.acos メソッド)

```
public static acos(x:Number) : Number
```

パラメータ x で指定された数値のアークコサイン (逆余弦) を計算してラジアン単位で返します。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

パラメータ

x : Number - -1.0 ～ 1.0 の数値。

戻り値

Number - 数値。パラメータ x のアークコサインです。

例

次の例では、いくつかの値のアークコサインを表を呼び出すと、結果として得られる示します。

```
trace(Math.acos(-1)); // output: 3.14159265358979
trace(Math.acos(0)); // output: 1.5707963267949
trace(Math.acos(1)); // output: 0
```

関連項目

asin (Math.asin メソッド), atan (Math.atan メソッド), atan2 (Math.atan2 メソッド), cos (Math.cos メソッド), sin (Math.sin メソッド), tan (Math.tan メソッド)

asin (Math.asin メソッド)

```
public static asin(x:Number) : Number
```

パラメータ x で指定された数値のアークサイン (逆正弦) を計算してラジアン単位で返します。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

パラメータ

x:Number - -1.0 ～ 1.0 の数値。

戻り値

Number - 2 で割った負の π と 2 で割った正の π の間 (-π/2 ～ π/2) の数値。

例

次の例では、いくつかの値のアークサインを表示します。

```
trace(Math.asin(-1)); // output: -1.5707963267949
trace(Math.asin(0)); // output: 0
trace(Math.asin(1)); // output: 1.5707963267949
```

関連項目

acos (Math.acos メソッド),atan (Math.atan メソッド),atan2 (Math.atan2 メソッド),cos (Math.cos メソッド),sin (Math.sin メソッド),tan (Math.tan メソッド)

atan (Math.atan メソッド)

```
public static atan(tangent:Number) : Number
```

パラメータ tangent で指定された値がタンジェント (正接) の値になる角度を計算してラジアン単位で返します。戻り値は、2 で割った負の π と 2 で割った正の π の間 (-π/2 ～ π/2) の値になります。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

パラメータ

tangent:Number - 角度のタンジェントを表す数値。

戻り値

Number - 2 で割った負の π と 2 で割った正の π の間 (-π/2 ～ π/2) の数値。

例

次の例では、いくつかのタンジェントの角度の値を表示します。

```
trace(Math.atan(-1)); // output: -0.785398163397448
trace(Math.atan(0)); // output: 0
trace(Math.atan(1)); // output: 0.785398163397448
```

関連項目

acos (Math.acos メソッド), asin (Math.asin メソッド), atan2 (Math.atan2 メソッド), cos (Math.cos メソッド), sin (Math.sin メソッド), tan (Math.tan メソッド)

atan2 (Math.atan2 メソッド)

```
public static atan2(y:Number, x:Number) : Number
```

円の *x* 軸 (0,0 は円の中心を示します) から反時計回りに測定した場合に、y/x 座標の角度をラジアン単位で計算して返します。戻り値は、正の π と負の π の間の値になります。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

パラメータ

y : Number - ポイントの *y* 座標を指定する数値。

x : Number - ポイントの *x* 座標を指定する数値。

戻り値

Number - 数値。

例

次の例では、座標 (0, 10) によって指定されたポイント (たとえば x = 0 と y = 10) の角度をラジアンで返します。atan2 への最初のパラメータは常に y 座標です。

```
trace(Math.atan2(10, 0)); // output: 1.5707963267949
```

関連項目

acos (Math.acos メソッド), asin (Math.asin メソッド), atan (Math.atan メソッド), cos (Math.cos メソッド), sin (Math.sin メソッド), tan (Math.tan メソッド)

ceil (Math.ceil メソッド)

```
public static ceil(x:Number) : Number
```

指定された数値または式を切り上げた値を返します。数値の切り上げとは、その数値以上の最も近い整数にすることです。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

パラメータ

x:Number - 数値または式。

戻り値

Number - パラメータ x の値以上の最も近い整数。

例

次の例では、値 13 を返します。

```
Math.ceil(12.5);
```

関連項目

floor (Math.floor メソッド),round (Math.round メソッド)

cos (Math.cos メソッド)

```
public static cos(x:Number) : Number
```

ラジアン単位で指定された角度のコサイン (余弦) を計算して返します。ラジアンを計算するには、Math クラスの説明を参照してください。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

パラメータ

x:Number - 角度をラジアンで表した数値。

戻り値

Number - -1.0 ～ 1.0 の数値。

例

次の例では、いくつかの角度のコサインを表示します。

```
trace (Math.cos(0)); // 0 degree angle. Output: 1
trace (Math.cos(Math.PI/2)); // 90 degree angle. Output: 6.12303176911189e-17
trace (Math.cos(Math.PI)); // 180 degree angle. Output: -1
trace (Math.cos(Math.PI*2)); // 360 degree angle. Output: 1
```

メモ : 90 度のコサインはゼロですが、2 進数を使用する 10 進数計算に伴う誤差により、Flash Player では、ゼロに極めて近い値ですが、ゼロではない数値になります。

関連項目

acos (Math.acos メソッド), asin (Math.asin メソッド), atan (Math.atan メソッド), atan2 (Math.atan2 メソッド), sin (Math.sin メソッド), tan (Math.tan メソッド)

E (Math.E プロパティ)

```
public static E : Number
```

自然対数の底を表す数学定数で *e* と表記されるものです。*e* のおよその値は 2.71828182845905 です。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

例

この例では、Math.E を使用して、1 年間に 100 % の利子が付く場合の複利を計算する方法を示します。

```
var principal:Number = 100;
var simpleInterest:Number = 100;
var continuouslyCompoundedInterest:Number = (100 * Math.E) - principal;

trace ("Beginning principal: $" + principal);
trace ("Simple interest after one year: $" + simpleInterest);
trace ("Continuously compounded interest after one year: $" +
  continuouslyCompoundedInterest);

//
Output:
Beginning principal: $100
Simple interest after one year: $100
Continuously compounded interest after one year: $171.828182845905
```

exp (Math.exp メソッド)

```
public static exp(x:Number) : Number
```

自然対数の底 (e) を、パラメータ x で指定された指数で累乗した値を返します。定数 Math.E を使用して、*e* の値を指定できます。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

パラメータ

x:Number - 指数。数値または式です。

戻り値

Number - 数値。

例

次の例では、2 つの値の対数を表示します。

```
trace(Math.exp(1)); // output: 2.71828182845905
trace(Math.exp(2)); // output: 7.38905609893065
```

関連項目

E (Math.E プロパティ)

floor (Math.floor メソッド)

```
public static floor(x:Number) : Number
```

パラメータ x で指定された数値または式を切り捨てた値を返します。切り捨てとは、指定された数値または式以下の最も近い整数にすることです。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

パラメータ

x:Number - 数値または式。

戻り値

Number - パラメータ x の値以下の最も近い整数。

例

次の例では、値 12 を返します。

```
Math.floor(12.5);
```

次の例では、値 -7 を返します。

```
Math.floor(-6.5);
```

LN10 (Math.LN10 プロパティ)

```
public static LN10 : Number
```

10 の自然対数を表す数学定数で loge10 と表記されるものです。近似値は 2.302585092994046 です。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

例

この例では、Math.LN10 の値を出力します。

```
trace(Math.LN10);
// output: 2.30258509299405
```

LN2 (Math.LN2 プロパティ)

```
public static LN2 : Number
```

2 の自然対数を表す数学定数で loge2 と表記されるものです。近似値は 0.6931471805599453 です。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

log (Math.log メソッド)

```
public static log(x:Number) : Number
```

パラメータ x の自然対数を返します。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

パラメータ

x:Number - 値が 0 よりも大きい数値または式。

戻り値

Number - パラメータ x の自然対数。

例

次の例では、3 つの数値の対数を表示します。

```
trace(Math.log(0)); // output: -Infinity
trace(Math.log(1)); // output: 0
trace(Math.log(2)); // output: 0.693147180559945
trace(Math.log(Math.E)); // output: 1
```

LOG10E (Math.LOG10E プロパティ)

```
public static LOG10E : Number
```

10 を底とする定数 e (Math.E) の対数を表す数学定数で log10e と表記されるものです。近似値は 0.4342944819032518 です。

Math.log( ) メソッドは、数値の自然対数を計算します。Math.log( ) の結果に Math.LOG10E を乗算すると、10 を底とする対数を得ることができます。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

例

次の例では、10 を底とする対数の計算方法を示します。

```
trace(Math.log(1000) * Math.LOG10E);
// Output: 3
```

LOG2E (Math.LOG2E プロパティ)

```
public static LOG2E : Number
```

2 を底とする定数 e (Math.E) の対数を表す数学定数で log2e と表記されるものです。近似値は 1.4426950408889633 87 です。

Math.log メソッドは、数値の自然対数を計算します。Math.log( ) の結果に Math.LOG2E を乗算すると、2 を底とする対数を得ることができます。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

例

次の例では、2 を底とする対数の計算方法を示します。

```
trace(Math.log(16) * Math.LOG2E);
// Output: 4
```

max (Math.max メソッド)

```
public static max(x:Number, y:Number) : Number
```

x と y を評価し、大きいほうの値を返します。

対応バージョン： ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

パラメータ

x:Number - 数値または式。

y:Number - 数値または式。

戻り値

Number - 数値。

例

次の例では、評価された式のうち大きい方である Thu Dec 30 00:00:00 GMT-0700 2004 を表示します。

```
var date1:Date = new Date(2004, 11, 25);
var date2:Date = new Date(2004, 11, 30);
var maxDate:Number = Math.max(date1.getTime(), date2.getTime());
trace(new Date(maxDate).toString());
```

関連項目

min (Math.min メソッド), Date

min (Math.min メソッド)

```
public static min(x:Number, y:Number) : Number
```

x と y を評価し、小さいほうの値を返します。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

パラメータ

x:Number - 数値または式。

y:Number - 数値または式。

戻り値

Number - 数値。

例

次の例では、評価された式のうち小さい方である Sat Dec 25 00:00:00 GMT-0700 2004 を表示します。

```
var date1:Date = new Date(2004, 11, 25);
var date2:Date = new Date(2004, 11, 30);
var minDate:Number = Math.min(date1.getTime(), date2.getTime());
trace(new Date(minDate).toString());
```

関連項目

max (Math.max メソッド)

PI (Math.PI プロパティ)

```
public static PI : Number
```

円周と円の直径の比を表す数学定数で pi と表記されるものです。近似値は 3.141592653589793 です。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

例

次の例では、数学定数 pi と Drawing API を使用して円を描きます。

```
drawCircle(this, 100, 100, 50);
//
function drawCircle(mc:MovieClip, x:Number, y:Number, r:Number):Void {
  mc.lineStyle(2, 0xFF0000, 100);
  mc.moveTo(x+r, y);
  mc.curveTo(r+x, Math.tan(Math.PI/8)*r+y, Math.sin(Math.PI/4)*r+x,
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, r+y, x, r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, r+y, -Math.sin(Math.PI/4)*r+x,
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-r+x, Math.tan(Math.PI/8)*r+y, -r+x, y);
  mc.curveTo(-r+x, -Math.tan(Math.PI/8)*r+y, -Math.sin(Math.PI/4)*r+x, -
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, -r+y, x, -r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, -r+y, Math.sin(Math.PI/4)*r+x, -
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(r+x, -Math.tan(Math.PI/8)*r+y, r+x, y);
}
```

pow (Math.pow メソッド)

```
public static pow(x:Number, y:Number) : Number
```

x の y 乗を計算して返します。

対応バージョン： ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

パラメータ

x:Number - 累乗される数値。底ともいいます。

y:Number - パラメータ x を累乗する指数。

戻り値

Number - 数値。

例

この例では、Math.pow と Math.sqrt を使用して線の長さを計算します。

```
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.origX = _xmouse;
  this.origY = _ymouse;
};
mouseListener.onMouseUp = function() {
  this.newX = _xmouse;
  this.newY = _ymouse;
  var minY = Math.min(this.origY, this.newY);
  var nextDepth:Number = canvas_mc.getNextHighestDepth();
  var line_mc:MovieClip =
  canvas_mc.createEmptyMovieClip("line"+nextDepth+"_mc", nextDepth);
  line_mc.moveTo(this.origX, this.origY);
  line_mc.lineStyle(2, 0x000000, 100);
  line_mc.lineTo(this.newX, this.newY);
  var hypLen:Number = Math.sqrt(Math.pow(line_mc._width,
  2)+Math.pow(line_mc._height, 2));
  line_mc.createTextField("length"+nextDepth+"_txt",
  canvas_mc.getNextHighestDepth(), this.origX, this.origY-22, 100, 22);
  line_mc['length'+nextDepth+'_txt'].text = Math.round(hypLen) +" pixels";
};
Mouse.addListener(mouseListener);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

random (Math.random メソッド)

```
public static random() : Number
```

疑似乱数 n を返します (0 <= n < 1)。返される数値は技術的に非公開の方式で計算されたもので、「擬似」乱数になります。

対応バージョン： ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

戻り値

Number - 数値。

例

次の例では、4 ～ 11 (4 と 11 を含みます) の整数を 100 個出力します。

```
function randRange(min:Number, max:Number):Number {
   var randomNum:Number = Math.floor(Math.random() * (max - min + 1)) + min;
   return randomNum;
}
for (var i = 0; i < 100; i++) {
   var n:Number = randRange(4, 11)
   trace(n);
}
```

round (Math.round メソッド)

```
public static round(x:Number) : Number
```

パラメータ x の値を最も近い整数に四捨五入し、値を返します。パラメータ x が 2 つの最も近い整数から等距離である場合 (.5 で終わる数値など)、値は次に大きな整数に切り上げられます。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

パラメータ

x:Number - 数値。

戻り値

Number - 数値。整数です。

例

次の例では、指定された 2 つの整数の間にあるランダムな数値を返します。

```
function randRange(min:Number, max:Number):Number {
   var randomNum:Number = Math.round(Math.random() * (max-min+1) + (min-.5));
   return randomNum;
}
for (var i = 0; i<25; i++) {
   trace(randRange(4, 11));
}
```

関連項目

ceil (Math.ceil メソッド),floor (Math.floor メソッド)

sin (Math.sin メソッド)

```
public static sin(x:Number) : Number
```

ラジアン単位で指定された角度のサイン (正弦) を計算して返します。ラジアンを計算するには、Math クラスの説明を参照してください。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

パラメータ

x:Number - 角度をラジアンで表した数値。

戻り値

Number - 数値。指定された角度のサイン (-1.0 ～ 1.0) です。

例

次の例では、数学定数 pi、角度のサイン、および Drawing API を使用して円を描きます。

```
drawCircle(this, 100, 100, 50);
//
function drawCircle(mc:MovieClip, x:Number, y:Number, r:Number):Void {
  mc.lineStyle(2, 0xFF0000, 100);
  mc.moveTo(x+r, y);
  mc.curveTo(r+x, Math.tan(Math.PI/8)*r+y, Math.sin(Math.PI/4)*r+x,
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, r+y, x, r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, r+y, -Math.sin(Math.PI/4)*r+x,
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-r+x, Math.tan(Math.PI/8)*r+y, -r+x, y);
  mc.curveTo(-r+x, -Math.tan(Math.PI/8)*r+y, -Math.sin(Math.PI/4)*r+x, -
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, -r+y, x, -r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, -r+y, Math.sin(Math.PI/4)*r+x, -
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(r+x, -Math.tan(Math.PI/8)*r+y, r+x, y);
}
```

関連項目

acos (Math.acos メソッド), asin (Math.asin メソッド), atan (Math.atan メソッド), atan2 (Math.atan2 メソッド), cos (Math.cos メソッド), tan (Math.tan メソッド)

sqrt (Math.sqrt メソッド)

```
public static sqrt(x:Number) : Number
```

指定された数値の平方根を計算して返します。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

パラメータ

x:Number - 0 以上の数値または式。

戻り値

Number - パラメータ *x* が 0 以上の場合は数値、それ以外は NaN (非数) を返します。

例

この例では、Math.pow と Math.sqrt を使用して線の長さを計算します。

```
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.origX = _xmouse;
  this.origY = _ymouse;
};
mouseListener.onMouseUp = function() {
  this.newX = _xmouse;
  this.newY = _ymouse;
  var minY = Math.min(this.origY, this.newY);
  var nextDepth:Number = canvas_mc.getNextHighestDepth();
  var line_mc:MovieClip =
  canvas_mc.createEmptyMovieClip("line"+nextDepth+"_mc", nextDepth);
  line_mc.moveTo(this.origX, this.origY);
  line_mc.lineStyle(2, 0x000000, 100);
  line_mc.lineTo(this.newX, this.newY);
  var hypLen:Number = Math.sqrt(Math.pow(line_mc._width,
  2)+Math.pow(line_mc._height, 2));
  line_mc.createTextField("length"+nextDepth+"_txt",
  canvas_mc.getNextHighestDepth(), this.origX, this.origY-22, 100, 22);
  line_mc['length'+nextDepth+'_txt'].text = Math.round(hypLen) +" pixels";
};
Mouse.addListener(mouseListener);
```

SQRT1_2 (Math.SQRT1_2 プロパティ)

```
public static SQRT1_2 : Number
```

1/2 の平方根を表す数学定数です。近似値は 0.7071067811865476 です。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

例

この例では、Math.SQRT1_2. の値を出力します。

```
trace(Math.SQRT1_2);
// Output: 0.707106781186548
```

SQRT2 (Math.SQRT2 プロパティ)

```
public static SQRT2 : Number
```

2 の平方根を表す数学定数です。近似値は 1.4142135623730951 です。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

例

この例では、Math.SQRT2. の値を出力します。

```
trace(Math.SQRT2);
// Output: 1.4142135623731
```

tan (Math.tan メソッド)

```
public static tan(x:Number) : Number
```

指定された角度のタンジェント (正接) を計算して返します。ラジアンを計算するには、Math クラスの概説を参照してください。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 4 では、Math クラスのメソッドとプロパティは、近似を使用してシミュレートされるため、Flash Player 5 でサポートされている非シミュレート Math 関数ほど正確ではありません。

パラメータ

x:Number - 角度をラジアンで表した数値。

戻り値

Number - 数値。パラメータ x のタンジェントを返します。

例

次の例では、数学定数 pi、角度のタンジェント、および Drawing API を使用して円を描きます。

```
drawCircle(this, 100, 100, 50);
//
function drawCircle(mc:MovieClip, x:Number, y:Number, r:Number):Void {
  mc.lineStyle(2, 0xFF0000, 100);
  mc.moveTo(x+r, y);
  mc.curveTo(r+x, Math.tan(Math.PI/8)*r+y, Math.sin(Math.PI/4)*r+x,
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, r+y, x, r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, r+y, -Math.sin(Math.PI/4)*r+x,
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-r+x, Math.tan(Math.PI/8)*r+y, -r+x, y);
  mc.curveTo(-r+x, -Math.tan(Math.PI/8)*r+y, -Math.sin(Math.PI/4)*r+x, -
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, -r+y, x, -r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, -r+y, Math.sin(Math.PI/4)*r+x, -
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(r+x, -Math.tan(Math.PI/8)*r+y, r+x, y);
}
```

関連項目

acos (Math.acos メソッド), asin (Math.asin メソッド), atan (Math.atan メソッド), atan2 (Math.atan2 メソッド), cos (Math.cos メソッド), sin (Math.sin メソッド)

Matrix (flash.geom.Matrix)

```
Object
  |
  +-flash.geom.Matrix
```

```
public class Matrix
extends Object
```

flash.geom.Matrix クラスは、座標空間の間でポイントをマッピングする方法を決定する変換マトリックス (変換行列) を表します。Matrix オブジェクトのプロパティを設定して、MovieClip オブジェクトや BitmapData オブジェクトに適用することで、オブジェクトに対する各種グラフィック変換を実行できます。これらの変換機能には、平行移動 (*x* と *y* の位置変更)、回転、拡大・縮小、傾斜などが含まれます。

このような変換をアフィン変換と言います。アフィン変換では、変換中に線分の直線性が保たれると共に、平行線は平行のままになります。

変換マトリックスをムービークリップに適用するには、flash.geom.Transform オブジェクトを作成して、そのオブジェクトの matrix プロパティを変換マトリックスに設定します。Matrix オブジェクトは、flash.display.BitmapData クラスの `draw()` メソッドなど、いくつかのメソッドのパラメータとしても使用します。

変換マトリックスオブジェクトは、次の内容を備えた 3 x 3 のマトリックスであるとみなします。

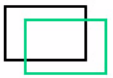
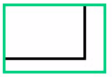
$$\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ u & v & w \end{bmatrix}$$

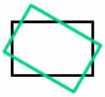
従来の変換マトリックスでは、`u`、`v`、および `w` プロパティによって特別な機能を提供します。Matrix クラスは 2 次元空間でのみ演算できるので、`u` プロパティと `v` プロパティの値は 0.0 で、`w` プロパティの値は 1.0 です。言い換えると、このマトリックスの有効値は次のとおりです。

$$\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Matrix オブジェクトの他の 6 つのプロパティ (`a`、`b`、`c`、`d`、`tx`、`ty`) のすべてについて、値を取得および設定できます。

Matrix クラスは、主要な 4 種類の変換機能 (平行移動、拡大・縮小、回転、傾斜) に対応しています。これらのうち 3 つの機能には専用メソッドがあります。以下の表に示します。

変換	メソッド	マトリックスの値	表示結果	説明
平行移動 (変位)	`translate (tx, ty)`	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$		イメージを `tx` ピクセルだけ右に、`ty` ピクセルだけ下に移動します。
拡大・縮小	`scale(sx, sy)`	$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$		各ピクセルの位置を *x* 軸方向に `sx` 倍、*y* 軸方向に `sy` 倍することで、イメージの大きさを変更します。

変換	メソッド	マトリックスの値	表示結果	説明
回転	rotate(q)	$\begin{bmatrix} \cos(q) & \sin(q) & 0 \\ -\sin(q) & \cos(q) & 0 \\ 0 & 0 & 1 \end{bmatrix}$		イメージを角度 q (単位はラジアン) だけ回転します。
傾斜または変形	なし。プロパティ b と c に設定する必要があります。	$\begin{bmatrix} 0 & \tan(skew_y) & 0 \\ \tan(skew_x) & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$		x 軸または y 軸に平行な方向にイメージをスライドします。Matrix オブジェクトの b プロパティは、y 軸に沿った傾斜角度のタンジェントを表し、Matrix オブジェクトの c プロパティは、x 軸に沿った傾斜角度のタンジェントを表します。

効果的に複数の変換を組み合わせることができるように、変換関数はそれぞれ現在のマトリックスプロパティを変更します。これを行うには、そのムービークリップまたはビットマップターゲットにマトリックスを適用する前に、複数の変形関数を呼び出します。

対応バージョン： ActionScript 1.0、Flash Player 8

関連項目

transform (MovieClip.transform プロパティ), Transform (flash.geom.Transform), draw (BitmapData.draw メソッド), a (Matrix.a プロパティ), b (Matrix.b プロパティ), c (Matrix.c プロパティ), d (Matrix.d プロパティ), tx (Matrix.tx プロパティ), ty (Matrix.ty プロパティ), translate (Matrix.translate メソッド), scale (Matrix.scale メソッド), rotate (Matrix.rotate メソッド)

プロパティー覧

オプション	プロパティ	説明
	a:Number	Matrix オブジェクトの 1 行 1 列目の値です。イメージの拡大・縮小や回転を行う際に、*x* 軸方向のピクセルの位置に影響を与えます。
	b:Number	Matrix オブジェクトの 1 行 2 列目の値です。イメージの回転や傾斜を行う際に、*y* 軸方向のピクセルの位置に影響を与えます。
	c:Number	Matrix オブジェクトの 2 行 1 列目の値です。イメージの回転や傾斜を行う際に、*x* 軸方向のピクセルの位置に影響を与えます。
	d:Number	Matrix オブジェクトの 2 行 2 列目の値です。イメージの拡大・縮小や回転を行う際に、*y* 軸方向のピクセルの位置に影響を与えます。
	tx:Number	*x* 軸方向に各ポイントを平行移動する距離です。
	ty:Number	*y* 軸方向に各ポイントを平行移動する距離です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター覧

署名	説明
Matrix([a:Number], [b:Number], [c:Number], [d:Number], [tx:Number], [ty:Number])	指定されたパラメータで新しい Matrix オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	clone() : Matrix	新しい Matrix オブジェクトとして、このマトリックスのクローンを返します。含まれるオブジェクトはまったく同じコピーになります。
	concat(m:Matrix) : Void	マトリックスを現在のマトリックスと連結して、2 つのマトリックスの図形効果を効果的に組み合わせます。
	createBox(scaleX: Number, scaleY:Number, [rotation:Number], [tx:Number], [ty:Number]) : Void	拡大・縮小、回転、平行移動に関するパラメータなどがあります。
	createGradientBox (width:Number, height:Number, [rotation:Number], [tx:Number], [ty:Number]) : Void	MovieClip.beginGradientFill() メソッドで使用する特定のスタイルを作成します。
	deltaTransformPoint(pt:Point) : Point	変換前の座標空間内のポイントが指定されると、そのポイントの変換後の座標を返します。
	identity() : Void	変換されたムービークリップや図形が元と同一になる値を各マトリックスプロパティに設定します。
	invert() : Void	元のマトリックスの逆の変形を実行します。
	rotate(angle: Number) : Void	回転変換を適用する場合に現在のマトリックスを使用できるように、そのマトリックスに値を設定します。
	scale(sx:Number, sy:Number) : Void	適用時の効果としてイメージのサイズが変更されるようにマトリックスを変更します。
	toString() : String	Matrix オブジェクトのプロパティのリストを表すテキスト値を返します。
	transformPoint (pt:Point) : Point	Matrix オブジェクトで表現される図形変換を、指定されたポイントに適用します。
	translate(tx:Number, ty:Number) : Void	変換の効果としてオブジェクトが *x* 軸方向と *y* 軸方向に移動するように Matrix オブジェクトを変更します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド),hasOwnProperty (Object.hasOwnProperty メソッド),isPropertyEnumerable (Object.isPropertyEnumerable メソッド),isPrototypeOf (Object.isPrototypeOf メソッド),registerClass (Object.registerClass メソッド),toString (Object.toString メソッド),unwatch (Object.unwatch メソッド),valueOf (Object.valueOf メソッド),watch (Object.watch メソッド)

a (Matrix.a プロパティ)

public a : Number

Matrix オブジェクトの 1 行 1 列目の値です。イメージの拡大・縮小や回転を行う際に、*x* 軸方向のピクセルの位置に影響を与えます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、Matrix オブジェクト myMatrix を作成し、a の値を設定します。

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.a); // 1

myMatrix.a = 2;
trace(myMatrix.a); // 2
```

b (Matrix.b プロパティ)

public b : Number

Matrix オブジェクトの 1 行 2 列目の値です。イメージの回転や傾斜を行う際に、*y* 軸方向のピクセルの位置に影響を与えます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、Matrix オブジェクト myMatrix を作成し、b の値を設定します。

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.b); // 0

var degrees:Number = 30;
var radians:Number = (degrees/180) * Math.PI;
myMatrix.b = Math.tan(radians);
trace(myMatrix.b); // 0.577350269189626
```

c (Matrix.c プロパティ)

```
public c : Number
```

Matrix オブジェクトの 2 行 1 列目の値です。イメージの回転や傾斜を行う際に、*x* 軸方向のピクセルの位置に影響を与えます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、Matrix オブジェクト myMatrix を作成し、c の値を設定します。

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.c); // 0

var degrees:Number = 30;
var radians:Number = (degrees/180) * Math.PI;
myMatrix.c = Math.tan(radians);
trace(myMatrix.c); // 0.577350269189626
```

clone (Matrix.clone メソッド)

```
public clone() : Matrix
```

新しい Matrix オブジェクトとして、このマトリックスのクローンを返します。含まれるオブジェクトはまったく同じコピーになります。

対応バージョン : ActionScript 1.0、Flash Player 8

戻り値

Matrix - Matrix オブジェクト。

例

次の例では、myMatrix から clonedMatrix 変数を作成します。Matrix クラスには equals メソッドがないので、次の例では、2 つのマトリックスの等価性をテストするために独自に作成した関数を使用します。

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix(2, 0, 0, 2, 0, 0);
var clonedMatrix:Matrix = new Matrix();

trace(myMatrix); // (a=2, b=0, c=0, d=2, tx=0, ty=0)
trace(clonedMatrix); // (a=1, b=0, c=0, d=1, tx=0, ty=0)
trace(equals(myMatrix, clonedMatrix)); // false
```

```
clonedMatrix = myMatrix.clone();

trace(myMatrix); // (a=2, b=0, c=0, d=2, tx=0, ty=0)
trace(clonedMatrix); // (a=2, b=0, c=0, d=2, tx=0, ty=0)
trace(equals(myMatrix, clonedMatrix)); // true

function equals(m1:Matrix, m2:Matrix):Boolean {
  return m1.toString() == m2.toString();
}
```

concat (Matrix.concat メソッド)

```
public concat(m:Matrix) : Void
```

マトリックスを現在のマトリックスと連結して、2 つのマトリックスの図形効果を効果的に組み合わせます。数学的に言うと、2 つのマトリックスを連結することは、マトリックスの乗算を使って組み合わせることと同じです。

たとえば、マトリックス m1 はオブジェクトの大きさを 4 倍にし、マトリックス m2 はオブジェクトを 1.5707963267949 (Math.PI/2) ラジアン回転する場合、m1.concat(m2) は、オブジェクトの大きさを 4 倍にし、そのオブジェクトを Math.PI/2 ラジアン回転するマトリックスに m1 を変換します。

このメソッドは、連結されたマトリックスでソースマトリックスを置き換えます。2 つのソースマトリックスをどちらも変更しないで 2 つのマトリックスを連結する場合は、例に示されているように、まず clone() メソッドを使用して、ソースマトリックスをコピーします。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

m:Matrix - ソースマトリックスに連結するマトリックス。

例

次の例では、3 つのマトリックスを作成して、3 つの矩形のムービークリップの変換を定義します。最初の 2 つのマトリックス rotate45Matrix と doubleScaleMatrix は 2 つの矩形 rectangleMc_1 と rectangleMc_2 に適用されます。次に、rotate45Matrix と doubleScaleMatrix の concat() メソッドを使用して、3 つ目のマトリックス scaleAndRotateMatrix を作成します。その後、このマトリックスを rectangleMc_3 に適用して拡大・縮小と回転を行います。

```
import flash.geom.Matrix;
import flash.geom.Transform;

var rectangleMc_0:MovieClip = createRectangle(20, 80, 0x000000);
var rectangleMc_1:MovieClip = createRectangle(20, 80, 0xFF0000);
var rectangleMc_2:MovieClip = createRectangle(20, 80, 0x00FF00);
var rectangleMc_3:MovieClip = createRectangle(20, 80, 0x0000FF);

var rectangleTrans_1:Transform = new Transform(rectangleMc_1);
var rectangleTrans_2:Transform = new Transform(rectangleMc_2);
var rectangleTrans_3:Transform = new Transform(rectangleMc_3);

var rotate45Matrix:Matrix = new Matrix();
rotate45Matrix.rotate(Math.PI/4);
rectangleTrans_1.matrix = rotate45Matrix;
rectangleMc_1._x = 100;
trace(rotate45Matrix.toString()); // (a=0.707106781186548, b=0.707106781186547,
  c=-0.707106781186547, d=0.707106781186548, tx=0, ty=0)

var doubleScaleMatrix:Matrix = new Matrix();
doubleScaleMatrix.scale(2, 2);
rectangleTrans_2.matrix = doubleScaleMatrix;
rectangleMc_2._x = 200;
trace(doubleScaleMatrix.toString()); // (a=2, b=0, c=0, d=2, tx=0, ty=0)

var scaleAndRotateMatrix:Matrix = doubleScaleMatrix.clone();
scaleAndRotateMatrix.concat(rotate45Matrix);
rectangleTrans_3.matrix = scaleAndRotateMatrix;
rectangleMc_3._x = 300;
trace(scaleAndRotateMatrix.toString()); // (a=1.4142135623731,
  b=1.41421356237309, c=-1.41421356237309, d=1.4142135623731, tx=0, ty=0)

function createRectangle(width:Number, height:Number, color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

createBox (Matrix.createBox メソッド)

```
public createBox(scaleX:Number, scaleY:Number, [rotation:Number], [tx:Number],
  [ty:Number]) : Void
```

拡大・縮小、回転、平行移動に関するパラメータなどがあります。マトリックスに適用する際に、これらのパラメータに基づいて、マトリックスの値を設定します。

createBox() を使用すると、identity()、rotate()、scale()、translate() の各メソッドを続けて適用するのと効果が同じになるマトリックスを取得できます。たとえば、mat1.createBox(2,2,Math.PI/5, 100, 100) の効果は次の 2 つのステートメントのものと同じです。

```
import flash.geom.Matrix;

var mat1:Matrix = new Matrix();
mat1.identity();
mat1.rotate(Math.PI/4);
mat1.scale(2,2);
mat1.translate(10,20);
```

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

scaleX:Number - 水平方向の拡大・縮小倍率。

scaleY:Number - 垂直方向の拡大・縮小倍率。

rotation:Number (オプション) - 回転量 (ラジアン単位)。デフォルト値は 0 です。

tx:Number (オプション) - *x* 軸に沿って右方向に平行移動 (移動) するピクセル数。デフォルト値は 0 です。

ty:Number (オプション) - *y* 軸に沿って下方向に平行移動 (移動) するピクセル数。デフォルト値は 0 です。

例

次の例では、createBox() メソッドを呼び出すことで、myMatrix の scaleX の倍率、scaleY の倍率、回転、*x* の位置、*y* の位置を設定します。

```
import flash.geom.Matrix;
import flash.geom.Transform;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

myMatrix.createBox(1, 2, Math.PI/4, 100, 200);
trace(myMatrix.toString()); // (a=0.707106781186548, b=1.41421356237309,
  c=-0.707106781186547, d=1.4142135623731, tx=100, ty=200)
```

```
var rectangleMc:MovieClip = createRectangle(20, 80, 0xFF0000);
var rectangleTrans:Transform = new Transform(rectangleMc);
rectangleTrans.matrix = myMatrix;
```

関連項目

beginBitmapFill (MovieClip.beginBitmapFill メソッド)

createGradientBox (Matrix.createGradientBox メソッド)

```
public createGradientBox(width:Number, height:Number, [rotation:Number],
  [tx:Number], [ty:Number]) : Void
```

MovieClip.beginGradientFill() メソッドで使用する特定のスタイルを作成します。width と height は scaleX と scaleY のペアで拡大・縮小されます。tx と ty の値は、width と height の半分だけオフセットされます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

width:Number - グラデーションボックスの幅。

height:Number - グラデーションボックスの高さ。

rotation:Number (オプション) - 回転量 (ラジアン単位)。デフォルト値は 0 です。

tx:Number (オプション) - *x* 軸に沿って右方向に平行移動 (移動) する距離 (ピクセル単位)。この値は、width パラメータの半分だけオフセットされます。デフォルト値は 0 です。

ty:Number (オプション) - *y* 軸に沿って下方向に平行移動 (移動) する距離 (ピクセル単位)。この値は、height パラメータの半分だけオフセットされます。デフォルト値は 0 です。

例

次の例では、MovieClip オブジェクトの beginGradientFill() メソッドのパラメータとして myMatrix を使用します。

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

myMatrix.createGradientBox(200, 200, 0, 50, 50);
trace(myMatrix.toString()); // (a=0.1220703125, b=0, c=0, d=0.1220703125,
  tx=150, ty=150)
```

```
var depth:Number = this.getNextHighestDepth();
var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
var colors:Array = [0xFF0000, 0x0000FF];
var alphas:Array = [100, 100];
var ratios:Array = [0, 0xFF];
mc.beginGradientFill("linear", colors, alphas, ratios, myMatrix);
mc.lineTo(0, 300);
mc.lineTo(300, 300);
mc.lineTo(300, 0);
mc.lineTo(0, 0);
```

関連項目

beginGradientFill (MovieClip.beginGradientFill メソッド)

d (Matrix.d プロパティ)

```
public d : Number
```

Matrix オブジェクトの 2 行 2 列目の値です。イメージの拡大・縮小や回転を行う際に、*y* 軸方向のピクセルの位置に影響を与えます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、Matrix オブジェクト myMatrix を作成し、d の値を設定します。

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.d); // 1

myMatrix.d = 2;
trace(myMatrix.d); // 2
```

deltaTransformPoint (Matrix.deltaTransformPoint メソッド)

```
public deltaTransformPoint(pt:Point) : Point
```

変換前の座標空間内のポイントが指定されると、そのポイントの変換後の座標を返します。transformPoint() メソッドを使用して適用される通常の変換とは異なり、deltaTransformPoint() メソッドの変換では平行移動パラメータ tx と ty が考慮されません。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

pt:Point - Point オブジェクト。

戻り値

Point - 新しい Point オブジェクト。

例

次の例では、deltaTransformPoint() メソッドを使用して myPoint から deltaTransformedPoint を作成する方法を示します。この例では、translate() メソッドはポイント deltaTransformedPoint の位置を変更しません。ただし、scale() メソッドはそのポイントの位置を変更します。ポイントの x 値が 3 倍されて、50 から 150 に増加します。

```
import flash.geom.Matrix;
import flash.geom.Point;

var myMatrix:Matrix = new Matrix();
trace(myMatrix); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

myMatrix.translate(100, 0);
trace(myMatrix); // (a=1, b=0, c=0, d=1, tx=100, ty=0)

myMatrix.scale(3, 3);
trace(myMatrix); // (a=3, b=0, c=0, d=3, tx=300, ty=0)

var myPoint:Point = new Point(50,0);
trace(myPoint); // (50, 0)

var deltaTransformedPoint:Point = myMatrix.deltaTransformPoint(myPoint);
trace(deltaTransformedPoint); // (150, 0)

var pointMc_0:MovieClip = createRectangle(10, 10, 0xFF0000);
pointMc_0._x = myPoint.x;

var pointMc_1:MovieClip = createRectangle(10, 10, 0x00FF00);
pointMc_1._x = deltaTransformedPoint.x;

function createRectangle(width:Number, height:Number, color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

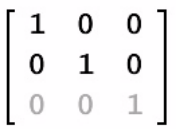
identity (Matrix.identity メソッド)

```
public identity() : Void
```

変換されたムービークリップや図形が元と同一になる値を各マトリックスプロパティに設定します。

identity() メソッドを呼び出すと、結果として得られる行列のプロパティは、a=1, b=0, c=0, d=1, tx=0, ty=0 になります。

マトリックス表記の場合、単位マトリックス (単位行列) は次のようになります。



対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、identity() メソッドを呼び出すことで、呼び出し先の Matrix オブジェクトを単位 Matrix オブジェクトに変換します。元の Matrix オブジェクトに事前に適用する変換の回数や種類に関係なく、identity() を呼び出すと、マトリックスの値が (a=1, b=0, c=0, d=1, tx=0, ty=0) に変換されます。

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix(2, 0, 0, 2, 0 ,0);
trace(myMatrix.toString()); // (a=2, b=0, c=0, d=2, tx=0, ty=0)

myMatrix.rotate(Math.atan(3/4));
trace(myMatrix.toString()); // (a=1.6, b=1.2, c=-1.2, d=1.6, tx=0, ty=0)

myMatrix.translate(100,200);
trace(myMatrix.toString()); // (a=1.6, b=1.2, c=-1.2, d=1.6, tx=100, ty=200)

myMatrix.scale(2, 2);
trace(myMatrix.toString()); // (a=3.2, b=2.4, c=-2.4, d=3.2, tx=200, ty=400)

myMatrix.identity();
trace(myMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)
```

invert (Matrix.invert メソッド)

```
public invert() : Void
```

元のマトリックスの逆の変形を実行します。逆マトリックスをオブジェクトに適用して、元のマトリックスの適用時に実行された変換を取り消すことができます。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、doubleScaleMatrix の invert() メソッドを呼び出すことで、halfScaleMatrix を作成し、この 2 つのマトリックスが他方のマトリックスの逆マトリックス、つまり、他方のマトリックスで実行された変換を取り消すマトリックスであることを示します。この例では、noScaleMatrix と同等の originalAndInverseMatrix を作成することでこの変換を示します。

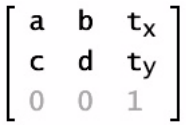
```
import flash.geom.Matrix;
import flash.geom.Transform;

var rectangleMc_0:MovieClip = createRectangle(20, 80, 0xFF0000);
var rectangleMc_1:MovieClip = createRectangle(20, 80, 0x00FF00);
var rectangleMc_2:MovieClip = createRectangle(20, 80, 0x0000FF);
var rectangleMc_3:MovieClip = createRectangle(20, 80, 0x000000);

var rectangleTrans_0:Transform = new Transform(rectangleMc_0);
var rectangleTrans_1:Transform = new Transform(rectangleMc_1);
var rectangleTrans_2:Transform = new Transform(rectangleMc_2);
var rectangleTrans_3:Transform = new Transform(rectangleMc_3);

var doubleScaleMatrix:Matrix = new Matrix(2, 0, 0, 2, 0, 0);
rectangleTrans_0.matrix = doubleScaleMatrix;
trace(doubleScaleMatrix.toString()); // (a=2, b=0, c=0, d=2, tx=0, ty=0)

var noScaleMatrix:Matrix = new Matrix(1, 0, 0, 1, 0, 0);
rectangleTrans_1.matrix = noScaleMatrix;
rectangleMc_1._x = 100;
trace(noScaleMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

var halfScaleMatrix:Matrix = doubleScaleMatrix.clone();
halfScaleMatrix.invert();
rectangleTrans_2.matrix = halfScaleMatrix;
rectangleMc_2._x = 200;
trace(halfScaleMatrix.toString()); // (a=0.5, b=0, c=0, d=0.5, tx=0, ty=0)

var originalAndInverseMatrix:Matrix = doubleScaleMatrix.clone();
originalAndInverseMatrix.concat(halfScaleMatrix);
rectangleTrans_3.matrix = originalAndInverseMatrix;
rectangleMc_3._x = 300;
trace(originalAndInverseMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)
```

```
function createRectangle(width:Number, height:Number, color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

Matrix() コンストラクタ

```
public Matrix([a:Number], [b:Number], [c:Number], [d:Number], [tx:Number],
  [ty:Number])
```

指定されたパラメータで新しい Matrix オブジェクトを作成します。マトリックス (行列) 表記の場合、プロパティは次のようになります。



新しい `Matrix()` コンストラクタにパラメータが指定されない場合は、次の値で構成される「単位マトリックス (単位行列)」を作成します。

`a = 1`	`b = 0`
`c = 0`	`d = 1`
`tx = 0`	`ty = 0`

マトリックス表記の場合、単位マトリックス (単位行列) は次のようになります。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

a:Number (オプション) - 新しい Matrix オブジェクトの 1 行 1 列目の値。

b:Number (オプション) - 新しい Matrix オブジェクトの 1 行 2 列目の値。

c:Number (オプション) - 新しい Matrix オブジェクトの 2 行 1 列目の値。

d:Number (オプション) - 新しい Matrix オブジェクトの 2 行 2 列目の値。

tx:Number (オプション) - 新しい Matrix オブジェクトの 1 行 3 列目の値。

ty:Number (オプション) - 新しい Matrix オブジェクトの 2 行 3 列目の値。

例

次の例では、Matrix コンストラクタにパラメータを渡さないで matrix_1 を作成し、このコンストラクタにパラメータを渡して matrix_2 を作成します。パラメータを渡さないで作成した Matrix オブジェクト matrix_1 は、値 (a=1, b=0, c=0, d=1, tx=0, ty=0) を持つ単位マトリックスです。

```
import flash.geom.Matrix;

var matrix_1:Matrix = new Matrix();
trace(matrix_1); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

var matrix_2:Matrix = new Matrix(1, 2, 3, 4, 5, 6);
trace(matrix_2); // (a=1, b=2, c=3, d=4, tx=5, ty=6)
```

rotate (Matrix.rotate メソッド)

```
public rotate(angle:Number) : Void
```

回転変換を適用する場合に現在のマトリックスを使用できるように、そのマトリックスに値を設定します。

rotate() メソッドは、Matrix オブジェクトの a、b、c および d の各プロパティを変更します。マトリックス表記の場合、これは次のようになります。

$$\begin{bmatrix} \cos(q) & \sin(q) & 0 \\ -\sin(q) & \cos(q) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`angle:Number` - 回転角度 (ラジアン単位)。

例

次の例では、rotate() メソッドを使って rectangleMc を時計方向に 30 度回転します。myMatrix を rectangleMc に適用すると、*_x* 値がリセットされるので、手動で 100 に設定し直します。

```
import flash.geom.Matrix;
import flash.geom.Transform;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

var degrees:Number = 30;
var radians:Number = (degrees/180) * Math.PI;
myMatrix.rotate(radians);
trace(myMatrix.toString()); // (a=0.866025403784439, b=0.5, c=-0.5,
  d=0.866025403784439, tx=0, ty=0)

var rectangleMc:MovieClip = createRectangle(20, 80, 0xFF0000);
trace(rectangleMc._x); // 0
rectangleMc._x = 100;
trace(rectangleMc._x); // 100

var rectangleTrans:Transform = new Transform(rectangleMc);
rectangleTrans.matrix = myMatrix;
trace(rectangleMc._x); // 0
rectangleMc._x = 100;
trace(rectangleMc._x); // 100

function createRectangle(width:Number, height:Number, color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

上の例では、MovieClip オブジェクトの _x プロパティを使用して rectangleMc の位置を変更しています。通常、マトリックスポジショニングを行う場合、ポジショニング手法を混在させることは悪いシンタックスであると考えられています。上の例を良いシンタックスで記述する場合は、平行移動マトリックスを myMatrix に連結することで rectangleMc の水平方向の位置を変更します。次の例で、それを示します。

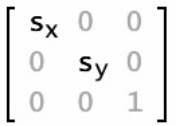
```
import flash.geom.Matrix;
import flash.geom.Transform;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

var degrees:Number = 30;
var radians:Number = (degrees/180) * Math.PI;
myMatrix.rotate(radians);
trace(myMatrix.toString()); // (a=0.866025403784439, b=0.5, c=-0.5,
  d=0.866025403784439, tx=0, ty=0)

var translateMatrix:Matrix = new Matrix();
translateMatrix.translate(100, 0);
myMatrix.concat(translateMatrix);
trace(myMatrix.toString()); // (a=0.866025403784439, b=0.5, c=-0.5,
  d=0.866025403784439, tx=100, ty=0)

var rectangleMc:MovieClip = createRectangle(20, 80, 0xFF0000);
trace(rectangleMc._x); // 0
rectangleMc._x = 100;
trace(rectangleMc._x); // 100

var rectangleTrans:Transform = new Transform(rectangleMc);
rectangleTrans.matrix = myMatrix;
trace(rectangleMc._x); // 100

function createRectangle(width:Number, height:Number, color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

scale (Matrix.scale メソッド)

```
public scale(sx:Number, sy:Number) : Void
```

適用時の効果としてイメージのサイズが変更されるようにマトリックスを変更します。サイズが変更されるイメージでは、各ピクセルの位置が *x* 軸方向に sx 倍に変更され、*y* 軸方向に sy 倍に変更されます。

scale() メソッドは、Matrix オブジェクトの a プロパティと d プロパティを変更します。マトリックス表記の場合、これは次のようになります。



対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

sx:Number - オブジェクトを *x* 軸方向に拡大・縮小するために使用される乗数。

sy:Number - オブジェクトを *y* 軸方向に拡大・縮小するために使用される乗数。

例

次の例では、scale() メソッドを使用して、myMatrix を水平方向に 3 倍、垂直方向に 4 倍します。

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix(2, 0, 0, 2, 100, 100);
trace(myMatrix.toString()); // (a=2, b=0, c=0, d=2, tx=100, ty=100)

myMatrix.scale(3, 4);
trace(myMatrix.toString()); // (a=6, b=0, c=0, d=8, tx=300, ty=400)
```

toString (Matrix.toString メソッド)

```
public toString() : String
```

Matrix オブジェクトのプロパティのリストを表すテキスト値を返します。

対応バージョン : ActionScript 1.0、Flash Player 8

戻り値

String - Matrix オブジェクトのプロパティ (a、b、c、d、tx、ty) の値が含まれるストリング。

例

次の例では、myMatrix を作成し、その値を (a=A, b=B, c=C, d=D, tx=TX, ty=TY) という形式のストリングに変換します。

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace("myMatrix: " + myMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)
```

transformPoint (Matrix.transformPoint メソッド)

```
public transformPoint(pt:Point) : Point
```

Matrix オブジェクトで表現される図形変換を、指定されたポイントに適用します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

pt:Point - 変換するポイント (x,y)。

戻り値

Point - 新しい Point オブジェクト。

例

次の例では、transformPoint() メソッドを使用して myPoint から transformedPoint を作成する方法を示します。translate() メソッドは transformedPoint の位置を変更します。この例では、scale() で元の x 値を 50 から 150 に 3 倍に増加し、translate() で x を 300 増加して、合計で 450 にします。

```
import flash.geom.Matrix;
import flash.geom.Point;

var myMatrix:Matrix = new Matrix();
trace(myMatrix); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

myMatrix.translate(100, 0);
trace(myMatrix); // (a=1, b=0, c=0, d=1, tx=100, ty=0)

myMatrix.scale(3, 3);
trace(myMatrix); // (a=3, b=0, c=0, d=3, tx=300, ty=0)

var myPoint:Point = new Point(50,0);
trace(myPoint); // (50, 0)

var transformedPoint:Point = myMatrix.transformPoint(myPoint);
trace(transformedPoint); // (450, 0)
```

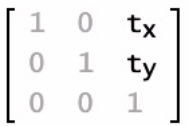
```
var pointMc_0:MovieClip = createRectangle(10, 10, 0xFF0000);
pointMc_0._x = myPoint.x;

var pointMc_1:MovieClip = createRectangle(10, 10, 0x00FF00);
pointMc_1._x = transformedPoint.x;

function createRectangle(width:Number, height:Number, color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

translate (Matrix.translate メソッド)

```
public translate(tx:Number, ty:Number) : Void
```

変換の効果としてオブジェクトが *x* 軸方向と *y* 軸方向に移動するように Matrix オブジェクトを変更します。

translate() メソッドは、Matrix オブジェクトの tx プロパティと ty プロパティを変更します。マトリックス表記の場合、これは次のようになります。



対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

tx:Number - *x* 軸に沿って右方向に移動する量 (ピクセル単位)。

ty:Number - *y* 軸に沿って下方向に移動する量 (ピクセル単位)。

例

次の例では、translate() メソッドを使用して rectangleMc の位置を x 方向に 100、y 方向に 50 変更します。translate() は平行移動の値 tx と ty を変更しますが、a、b、c、d は変更しません。

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix(2, 0, 0, 2, 100, 100);
trace(myMatrix.toString()); // (a=2, b=0, c=0, d=2, tx=100, ty=100)
```

```
myMatrix.translate(100, 50);
trace(myMatrix.toString()); // (a=2, b=0, c=0, d=2, tx=200, ty=150)
```

tx (Matrix.tx プロパティ)

```
public tx : Number
```

x 軸方向に各ポイントを平行移動する距離です。これは、Matrix オブジェクトの 1 行 3 列目の値に相当します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、Matrix オブジェクト myMatrix を作成し、tx の値を設定します。

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.tx); // 0

myMatrix.tx = 50; // 50
trace(myMatrix.tx);
```

ty (Matrix.ty プロパティ)

```
public ty : Number
```

y 軸方向に各ポイントを平行移動する距離です。これは、Matrix オブジェクトの 2 行 3 列目の値に相当します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、Matrix オブジェクト myMatrix を作成し、ty の値を設定します。

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.ty); // 0

myMatrix.ty = 50;
trace(myMatrix.ty); // 50
```

マイク

```
Object
  |
  +-Microphone
```

```
public class Microphone
extends Object
```

Microphone クラスを使用すると、Flash Player を実行するコンピュータに接続されたマイクからオーディオをキャプチャできます。

Microphone クラスは主に Flash Media Server で使用しますが、サーバーなしでも限定された方法で使用できます。たとえば、マイクからのサウンドをローカルシステムのスピーカーを通して出力できます。

注意 : Flash Player に [プライバシー] ダイアログボックスが表示され、ユーザーはマイクへのアクセスを許可するか拒否するかを選択できます。ステージのサイズは必ず 215 x 138 ピクセル以上に設定してください。これは、ダイアログボックスを表示するために必要な最小サイズです。

ユーザーおよび管理者は、サイトごとにまたはグローバルにマイクへのアクセスを無効にする場合もあります。

Microphone オブジェクトを作成または参照するには、`Microphone.get()` メソッドを使います。

対応バージョン : ActionScript 1.0、Flash Player 6

プロパティ一覧

オプション	プロパティ	説明
	`activityLevel:Number` (読み取り専用)	マイクで検知されているサウンド量を示す数値です。
	`gain:Number` (読み取り専用)	マイクによるシグナルのゲイン (増幅率) です。
	`index:Number` (読み取り専用)	マイクのインデックスを指定するゼロから始まる整数です。これは、`Microphone.names` から返される配列のインデックスと同じです。
	`muted:Boolean` (読み取り専用)	ユーザーがマイクへのアクセスを拒否しているか (`true`) 許可しているか (`false`) を示すブール値です。
	`name:String` (読み取り専用)	現在のサウンドキャプチャデバイスの名前を示すストリングです。サウンドキャプチャハードウェアから返されるものです。

オプション	プロパティ	説明
static	names:Array (読み取り専用)	[Macromedia Flash Player 設定] パネルの [プライバシー] を表示しないで、使用できるすべてのサウンドキャプチャデバイスの名前が含まれるストリング配列を取得します。
	rate:Number (読み取り専用)	マイクのサウンドキャプチャレート (kHz) です。
	silenceLevel:Number (読み取り専用)	マイクを有効にして Microphone.onActivity(true) を呼び出すのに必要な音量を指定する整数です。
	silenceTimeOut:Number (読み取り専用)	マイクによるサウンド検知が停止してから Microphone.onActivity(false) を呼び出すまでの経過時間をミリ秒単位で表した数値です。
	useEchoSuppression: Boolean (読み取り専用)	読み取り専用プロパティ。エコー抑制が有効になっている場合は true、それ以外の場合は false を示すブール値です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベント一覧

イベント	説明
onActivity = function(active: Boolean) {}	マイクがサウンドの検知を開始または停止したときに呼び出されます。
onStatus = function(infoObject: Object) {}	ユーザーがマイクへのアクセスを許可または拒否したときに呼び出されます。

メソッド一覧

オプション	署名	説明
static	get([index:Number]) : Microphone	オーディオをキャプチャする Microphone オブジェクトの参照を返します。
	setGain(gain:Number) : Void	マイクによるサウンドのゲイン (増幅率) を指定します。これは、シグナルが転送前にマイクで増幅される量を表します。
	setRate(rate:Number) : Void	マイクでサウンドをキャプチャするレート (kHz) を設定します。
	setSilenceLevel (silenceLevel:Number, [timeOut:Number]) : Void	サウンドと見なす最小入力レベルと、実際に無音状態が始まったと見なすまでの無音時間の長さを設定します。後者は省略可能です。
	setUseEchoSuppression(useEchoSuppression: Boolean) : Void	オーディオコーデックのエコー抑制機能を使用するかどうかを指定します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

activityLevel (Microphone.activityLevel プロパティ)

public activityLevel : Number (読み取り専用)

マイクで検知されているサウンド量を示す数値です。値の範囲は 0 (サウンドが検知されていない) ～ 100 (非常に大音量が検知されている) です。このプロパティ値に基づいて、Microphone.setSilenceLevel() メソッドに渡す適切な値を判断できます。

マイクを利用できる状態であっても、Microphone.get() がまだ呼び出されていないためにまだ使用されていない場合は、このプロパティは -1 に設定されます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、現在のマイクのアクティビティレベルを activityLevel_pb という ProgressBar インスタンスに表示します。

```
var activityLevel_pb:mx.controls.ProgressBar;
activityLevel_pb.mode = "manual";
activityLevel_pb.label = "Activity Level: %3%%";
activityLevel_pb.setStyle("themeColor", "0xFF0000");
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
this.onEnterFrame = function() {
  activityLevel_pb.setProgress(active_mic.activityLevel, 100);
};
active_mic.onActivity = function(active:Boolean) {
  if (active) {
  var haloTheme_str:String = "haloGreen";
  } else {
  var haloTheme_str:String = "0xFF0000";
  }
  activityLevel_pb.setStyle("themeColor", haloTheme_str);
};
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

get (Microphone.get メソッド), setSilenceLevel (Microphone.setSilenceLevel メソッド), setGain (Microphone.setGain メソッド)

gain (Microphone.gain プロパティ)

public gain : Number (読み取り専用)

マイクによるシグナルのゲイン (増幅率) です。有効な値は 0 ～ 100 で、デフォルト値は 50 です。

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の例では、gain_pb という ProgressBar インスタンスを使用してマイクのゲインを表示し、gain_nstep という NumericStepper インスタンスを使用してマイクのゲインを設定します。

```
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
```

```
gain_pb.label = "Gain: %3";
gain_pb.mode = "manual";
gain_pb.setProgress(active_mic.gain, 100);
gain_nstep.value = active_mic.gain;

function changeGain() {
  active_mic.setGain(gain_nstep.value);
  gain_pb.setProgress(active_mic.gain, 100);
}
gain_nstep.addEventListener("change", changeGain);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

setGain (Microphone.setGain メソッド)

get (Microphone.get メソッド)

```
public static get([index:Number]) : Microphone
```

オーディオをキャプチャする Microphone オブジェクトの参照を返します。実際にオーディオのキャプチャを開始するには、Microphone オブジェクトを MovieClip オブジェクトに接続する必要があります。MovieClip.attachAudio() を参照してください。

new コンストラクタを使用して作成するオブジェクトとは異なり、Microphone.get() を複数回呼び出すと、同じマイクへの参照が返されます。したがって、スクリプトに mic1 = Microphone.get() という行と mic2 = Microphone.get() という行が含まれている場合、mic1 と mic2 は両方とも同じデフォルトのマイクを参照します。

一般には、*index* の値を指定しないで、単に Microphone.get() メソッドを使用してデフォルトのマイクの参照を取得します。このセクションで後述する [マイク] ボックスで、Flash で使用するデフォルトのマイクをユーザーが指定できます。*index* 値を指定すると、ユーザーの指定とは異なるマイクの参照が取得される可能性があります。*index* を使用するのは特殊な場合に限られます。たとえば、アプリケーションで 2 つのマイクから同時にオーディオをキャプチャする場合などに、このパラメータを使用します。

SWF ファイルで Microphone.get() メソッドから返されたマイクにアクセスしようとすると (MovieClip.attachAudio() を使用した場合など)、[プライバシー] ダイアログボックスが表示されます。ユーザーは、そのマイクへのアクセスを許可するか拒否するかを選択できます。ステージのサイズは必ず 215 x 138 ピクセル以上に設定してください。これは、ダイアログボックスを表示するために必要な最小サイズです。

ユーザーがこのダイアログボックスに応答すると、Microphone.onStatus イベントハンドラがユーザーの応答を示す情報オブジェクトを返します。このイベントハンドラを使わずに、ユーザーがマイクへのアクセスを許可したかどうかを判断するには、Microphone.muted プロパティを使用します。

ユーザーは、再生中の SWF ファイルを右クリック (Windows) または Control キーを押しながらクリック (Macintosh) し、[設定] を選択し、[プライバシー] パネルを開き、[後で確認] を選択することによって、特定のドメインに対して永続的なプライバシー設定を指定できます。

ActionScript からユーザーに対して許可または拒否を設定することはできません。ただし、System.showSettings(0) を使用することで、ユーザーの [プライバシー] パネルを表示できます。ユーザーが [後で確認] を選択すると、このドメインの SWF ファイルでは [プライバシー] ダイアログボックスが表示されなくなります。

Microphone.get() が null を返した場合は、マイクが他のアプリケーションで使用されているか、そのシステムにマイクがインストールされていません。マイクがインストールされているかどうかを調べるには、Microphones.names.length を使用します。[Macromedia Flash Player 設定] パネルの [マイク] ボックスを表示して、Microphone.get() が参照するマイクをユーザーが選択できるようにするには、System.showSettings(2) を使用します。

対応バージョン : ActionScript 1.0、Flash Player 6 - メモ : 正しいシンタックスは Microphone.get() です。Microphone オブジェクトを変数に代入するには、active_mic = Microphone.get() のようなシンタックスを使用します。

パラメータ

index: Number **(任意) -** 取得するマイクを示すゼロから始まる整数で、Microphone.names に格納されている配列から決定します。デフォルトのマイクを取得する場合は、このパラメータを省略します。ほとんどのアプリケーションでは、デフォルトのマイクを使用するようにお勧めします。

戻り値

Microphone -

- *index* を指定しない場合は、デフォルトのマイクへの参照が返されます。デフォルトのマイクが利用できない場合は、利用できる最初のマイクへの参照が返されます。利用できるマイクがない場合、またはマイクがインストールされていない場合は、null が返されます。
- *index* を指定した場合は、要求したマイクへの参照が返されます。そのマイクが利用できない場合は、null が返されます。

例

次の例では、ユーザーにデフォルトのマイクを指定させた後、オーディオをキャプチャしてローカルに再生します。ハウリングを避けるために、このコードはヘッドフォンを付けた状態でテストすることをお勧めします。

```
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
System.showSettings(2);
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

get (Microphone.get メソッド), index (Microphone.index プロパティ), muted (Microphone.muted プロパティ), names (Microphone.names プロパティ), onStatus (Microphone.onStatus ハンドラ), attachAudio (MovieClip.attachAudio メソッド), showSettings (System.showSettings メソッド)

index (Microphone.index プロパティ)

public index : Number (読み取り専用)

マイクのインデックスを指定するゼロから始まる整数です。これは、Microphone.names から返される配列のインデックスと同じです。

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の例では、ユーザーのマシンで使用できるサウンドキャプチャデバイスの名前を mic_cb という ComboBox インスタンスに表示します。Label コンポーネントのインスタンス mic_lbl にマイクのインデックスを表示します。ComboBox を使用してデバイスを切り替えることができます。

```
var mic_lbl:mx.controls.Label;
var mic_cb:mx.controls.ComboBox;

this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
mic_lbl.text = "["+active_mic.index+"] "+active_mic.name;
mic_cb.dataProvider = Microphone.names;
mic_cb.selectedIndex = active_mic.index;
```

```
var cbListener:Object = new Object();
cbListener.change = function(evt:Object) {
   active_mic = Microphone.get(evt.target.selectedIndex);
   sound_mc.attachAudio(active_mic);
   mic_lbl.text = "["+active_mic.index+"] "+active_mic.name;
};
mic_cb.addEventListener("change", cbListener);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

get (Microphone.get メソッド),names (Microphone.names プロパティ)

muted (Microphone.muted プロパティ)

public muted : Boolean (読み取り専用)

ユーザーがマイクへのアクセスを拒否しているか (true) 許可しているか (false) を示すブール値です。この値が変わると、Microphone.onStatus が呼び出されます。詳細については、Microphone.get() を参照してください。

対応バージョン： ActionScript 1.0、Flash Player 6

例

この例では、デフォルトのマイクを取得して、ミュートになっているかどうかをチェックします。

```
var active_mic:Microphone = Microphone.get();
trace(active_mic.muted);
```

関連項目

get (Microphone.get メソッド),onStatus (Microphone.onStatus ハンドラ)

name (Microphone.name プロパティ)

public name : String (読み取り専用)

現在のサウンドキャプチャデバイスの名前を示すストリングです。サウンドキャプチャハードウェアから返されるものです。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、名前の配列やデフォルトのデバイスなど、マシンのサウンドキャプチャデバイスに関する情報を表示します。

```
var status_ta:mx.controls.TextArea;
status_ta.html = false;
status_ta.setStyle("fontSize", 9);
var microphone_array:Array = Microphone.names;
var active_mic:Microphone = Microphone.get();
status_ta.text = "The default device is: "+active_mic.name+newline+newline;
status_ta.text += "You have "+microphone_array.length+" device(s)
  installed."+newline+newline;
for (var i = 0; i<microphone_array.length; i++) {
  status_ta.text += "["+i+"] "+microphone_array[i]+newline;
}
```

関連項目

get (Microphone.get メソッド),names (Microphone.names プロパティ)

names (Microphone.names プロパティ)

public static names : Array (読み取り専用)

[Macromedia Flash Player 設定] パネルの [プライバシー] を表示しないで、使用できるすべてのサウンドキャプチャデバイスの名前が含まれるストリング配列を取得します。この配列は他のすべての ActionScript 配列と同様に動作します。この配列を使って、各サウンドキャプチャデバイスのゼロから始まるインデックスと、システム上の各サウンドキャプチャデバイスの数 (Microphone.names.length) を調べることができます。詳細については、Microphone.names Array クラスを参照してください。

Microphone.names プロパティを呼び出すと、ハードウェアを広範囲にわたって調べる必要があり、配列を作成するまでに数秒間かかることがあります。ほとんどの場合は、デフォルトのマイクをそのまま使用できます。

対応バージョン : ActionScript 1.0、Flash Player 6 - メモ : 正しいシンタックスは Microphone.names です。戻り値を変数に代入するには、mic_array=Microphone.names のようなシンタックスを使用します。現在のマイクの名前を調べるには、active_mic.name を使用します。

例

次の例では、名前の配列やデフォルトのデバイスなど、マシンのサウンドキャプチャデバイスに関する情報を表示します。

```
var status_ta:mx.controls.TextArea;
status_ta.html = false;
status_ta.setStyle("fontSize", 9);
var microphone_array:Array = Microphone.names;
var active_mic:Microphone = Microphone.get();
status_ta.text = "The default device is: "+active_mic.name+newline+newline;
status_ta.text += "You have "+microphone_array.length+" device(s)
  installed."+newline+newline;
for (var i = 0; i<microphone_array.length; i++) {
  status_ta.text += "["+i+"] "+microphone_array[i]+newline;
}
```

たとえば、次のような情報が表示されます。

```
The default device is: Logitech USB Headset
You have 2 device(s) installed.
[0] Logitech USB Headset
[1] YAMAHA AC-XG WDM Audio
```

関連項目

name（Microphone.name プロパティ）,get（Microphone.get メソッド）

onActivity (Microphone.onActivity ハンドラ)

```
onActivity = function(active: Boolean) {}
```

マイクがサウンドの検知を開始または停止したときに呼び出されます。このイベントハンドラに応答するには、アクティビティ値を処理する関数を作成する必要があります。

`Microphone.onActivity(true)` を呼び出すのに必要な音量と、サウンドのない時間がどれだけ続くと `Microphone.onActivity(false)` が呼び出されるかを指定するには、`Microphone.setSilenceLevel()` を使用します。

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

`active:Boolean` - ブール値。マイクがサウンドの検知を開始した場合には true、停止した場合には false に設定されます。

例

次の例では、アクティビティレベルを activityLevel_pb という ProgressBar インスタンスに表示します。マイクでサウンドが検知されると、onActivity 関数が呼び出されます。この関数は、ProgressBar インスタンスを変更します。

```
var activityLevel_pb:mx.controls.ProgressBar;
activityLevel_pb.mode = "manual";
activityLevel_pb.label = "Activity Level: %3%%";
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
active_mic.onActivity = function(active:Boolean) {
  if (active) {
    activityLevel_pb.indeterminate = false;
    activityLevel_pb.label = "Activity Level: %3%%";
  } else {
    activityLevel_pb.indeterminate = true;
    activityLevel_pb.label = "Activity Level: (inactive)";
  }
};
this.onEnterFrame = function() {
  activityLevel_pb.setProgress(active_mic.activityLevel, 100);
};
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

setSilenceLevel (Microphone.setSilenceLevel メソッド)

onStatus (Microphone.onStatus ハンドラ)

```
onStatus = function(infoObject:Object) {}
```

ユーザーがマイクへのアクセスを許可または拒否したときに呼び出されます。このイベントハンドラに応答するには、マイクによって生成される情報オブジェクトを処理する関数を作成する必要があります。

SWF ファイルがマイクにアクセスしようとすると、[プライバシー] ダイアログボックスが表示され、ユーザーはアクセスを許可するか拒否するかを選択できます。

- ユーザーがアクセスを許可した場合は、Microphone.muted プロパティが false に設定されます。次に、code プロパティ値が "Microphone.Unmuted"、level プロパティの値が "Status" に設定された情報オブジェクトを使用して、このイベントハンドラが呼び出されます。

■ ユーザーがアクセスを拒否した場合は、Microphone.muted プロパティが true に設定されます。次に、code プロパティ値が "Microphone.Muted"、level プロパティの値が "Status" に設定された情報オブジェクトを使用して、このイベントハンドラが呼び出されます。

このイベントハンドラを使わずに、ユーザーがマイクへのアクセスを許可したかどうかを判断するには、Microphone.muted プロパティを使用します。

メモ : ユーザーが特定のドメインのすべての SWF ファイルに対してアクセスを永続的に許可または拒否した場合は、ユーザーが後からプライバシー設定を変更しない限り、そのドメインの SWF ファイルに対してはこのメソッドは呼び出されません。

詳細については、Microphone.get() を参照してください。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

`infoObject:Object` - ステータスメッセージに従って定義されるパラメータ。

例

次の例では、[プライバシー] ダイアログボックスを表示します。ユーザーはこのダイアログボックスで、ハイパーリンクをクリックしたときにマイクへのアクセスを許可または拒否するかどうかを選択できます。アクセスを拒否すると、赤色の大きな文字で "*muted*" と表示されます。マイクへのアクセスを許可すると、この文字は表示されません。

```
this.createTextField("muted_txt", this.getNextHighestDepth(), 10, 10, 100, 22);
muted_txt.autoSize = true;
muted_txt.html = true;
muted_txt.selectable = false;
muted_txt.htmlText = "<a href=\"asfunction:System.showSettings\"><u>Click Here</
  u></a> to Allow/Deny access.";
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
active_mic.onStatus = function(infoObj:Object) {
  status_txt._visible = active_mic.muted;
  muted_txt.htmlText = "Status: <a
  href=\"asfunction:System.showSettings\"><u>"+infoObj.code+"</u></a>";
};
this.createTextField("status_txt", this.getNextHighestDepth(), 0, 0, 100, 22);
status_txt.html = true;
status_txt.autoSize = true;
status_txt.htmlText = "<font size='72' color='#FF0000'>muted</font>";
status_txt._x = (Stage.width-status_txt._width)/2;
status_txt._y = (Stage.height-status_txt._height)/2;
status_txt._visible = active_mic.muted;
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

get (Microphone.get メソッド), muted (Microphone.muted プロパティ), showSettings (System.showSettings メソッド), onStatus (System.onStatus ハンドラ)

rate (Microphone.rate プロパティ)

public rate : Number (読み取り専用)

マイクのサウンドキャプチャレート (kHz) です。デフォルト値は 8 kHz ですが、サウンドキャプチャデバイスがこの値に対応している必要があります。対応していない場合、デフォルト値はそのサウンドキャプチャデバイスが対応している 8 kHz よりも高いレートのうち、8 kHz に最も近い値になります。通常は 11 kHz です。

この値を設定するには、Microphone.setRate() を使います。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次のコードは、rate_cb という ComboBox インスタンスを使用して、マイクでサウンドをキャプチャするレートをユーザーが変更できるようにします。現在のレートは、rate_lbl という Label インスタンスに表示します。

```
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
var rate_array:Array = new Array(5, 8, 11, 22, 44);
rate_cb.dataProvider = rate_array;
rate_cb.labelFunction = function(item:Object) {
  return (item+" kHz");
};
for (var i = 0; i<rate_array.length; i++) {
  if (rate_cb.getItemAt(i) == active_mic.rate) {
  rate_cb.selectedIndex = i;
  break;
  }
}
function changeRate() {
  active_mic.setRate(rate_cb.selectedItem);
  rate_lbl.text = "Current rate: "+active_mic.rate+" kHz";
}
```

```
rate_cb.addEventListener("change", changeRate);
rate_lbl.text = "Current rate: "+active_mic.rate+" kHz";
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

setRate (Microphone.setRate メソッド)

setGain (Microphone.setGain メソッド)

```
public setGain(gain: Number) : Void
```

マイクによるサウンドのゲイン (増幅率) を指定します。これは、シグナルが転送前にマイクで増幅される量を表します。値 0 は 0 倍、つまりサウンドが転送されないことを示します。

この設定はステレオのボリュームつまみのようなものと考えることができます。0 は消音、50 は通常のボリューム、50 より小さい値は通常よりも低いボリューム、50 より大きい値は通常よりも大きいボリュームを表します。

対応バージョン: ActionScript 1.0、Flash Player 6

パラメータ

gain:Number - マイクによるシグナルのゲイン (増幅率) を指定する整数。指定できる値は 0 ～ 100 です。デフォルト値は 50 です。ただし、ユーザーは [Macromedia Flash Player 設定] パネルの [マイク] でこの値を変更できます。

例

次の例では、gain_pb という ProgressBar インスタンスを使用してマイクのゲインを表示し、gain_nstep という NumericStepper インスタンスを使用してマイクのゲインを設定します。

```
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);

gain_pb.label = "Gain: %3";
gain_pb.mode = "manual";
gain_pb.setProgress(active_mic.gain, 100);
gain_nstep.value = active_mic.gain;

function changeGain() {
  active_mic.setGain(gain_nstep.value);
  gain_pb.setProgress(active_mic.gain, 100);
}
```

```
gain_nstep.addEventListener("change", changeGain);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

gain (Microphone.gain プロパティ), setUseEchoSuppression (Microphone.setUseEchoSuppression メソッド)

setRate (Microphone.setRate メソッド)

```
public setRate(rate:Number) : Void
```

マイクでサウンドをキャプチャするレート (kHz) を設定します。

対応バージョン: ActionScript 1.0、Flash Player 6

パラメータ

`rate:Number` - マイクのサウンドキャプチャレート (kHz)。指定できる値は 5、8、11、22、44 です。デフォルト値は 8 kHz です。ただし、サウンドキャプチャデバイスがこの値に対応している必要があります。対応していない場合、デフォルト値はそのサウンドキャプチャデバイスが対応している 8 kHz よりも高いレートのうち、8 kHz に最も近い値になります。通常は 11 kHz です。

例

次の例では、ユーザーが選択したレート (userRate に設定した値) が 5、8、11、22、44 のいずれかである場合に、その値をマイクのレートとして設定します。それ以外の場合は、サウンドキャプチャデバイスが対応しているレートの中で最も近い有効な値に丸めます。

```
active_mic.setRate(userRate);
```

次の例では、rate_cb という ComboBox インスタンスを使用して、マイクでサウンドをキャプチャするレートをユーザーが変更できるようにします。現在のレートは、rate_lbl という Label インスタンスに表示します。

```
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
var rate_array:Array = new Array(5, 8, 11, 22, 44);
rate_cb.dataProvider = rate_array;
rate_cb.labelFunction = function(item:Object) {
  return (item+" kHz");
};
```

```
for (var i = 0; i<rate_array.length; i++) {
   if (rate_cb.getItemAt(i) == active_mic.rate) {
   rate_cb.selectedIndex = i;
   break;
   }
}
function changeRate() {
   active_mic.setRate(rate_cb.selectedItem);
   rate_lbl.text = "Current rate: "+active_mic.rate+" kHz";
}
rate_cb.addEventListener("change", changeRate);
rate_lbl.text = "Current rate: "+active_mic.rate+" kHz";
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

rate (Microphone.rate プロパティ)

setSilenceLevel (Microphone.setSilenceLevel メソッド)

```
public setSilenceLevel(silenceLevel:Number, [timeOut:Number]) : Void
```

サウンドと見なす最小入力レベルと、実際に無音状態が始まったと見なすまでの無音時間の長さを設定します。後者は省略可能です。

- マイクがサウンドをまったく検知しないようにするには、*level* に 100 を指定します。こうすると、Microphone.onActivity が呼び出されることはありません。
- マイクが現在検知している音量を調べるには、Microphone.activityLevel を使います。

アクティビティ検知とは、人が話していることをオーディオレベルに基づいて検知する機能のことです。誰も話していないときは、関連付けられているオーディオストリームを送信する必要がないので、帯域幅を節約できます。この情報を利用して、ユーザー (または他の人) が話していないことを視覚的なフィードバックとして示すこともできます。

サイレンス値はアクティビティ値に直接対応します。完全なサイレンスのアクティビティ値は 0 です。継続的な大音量 (現在のゲインに基づいて設定できる大音量) のアクティビティ値は 100 です。ゲイン (増幅率) を適切に調整すると、話していないときにはアクティビティ値がサイレンス値よりも小さくなります。話しているときには、アクティビティ値はサイレンス値よりも大きくなります。

このメソッドの目的は Camera.setMotionLevel() に似ています。どちらのメソッドも、onActivity イベントハンドラを呼び出すタイミングを指定するために使用します。ただし、パブリッシュするストリームに対する影響という点では、この 2 つのメソッドは大きく異なります。

- Camera.setMotionLevel() はモーションを検知し、使用する帯域幅には影響しないように設計されています。ビデオストリームでモーションが検知されない間も、ビデオは送信されます。
- Microphone.setSilenceLevel() は帯域幅を最適化するように設計されています。オーディオストリームが無音と考えられる場合には、オーディオデータは送信されません。代わりに、無音状態が始まったことを示すメッセージが送信されます。

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

silenceLevel:Number - マイクを有効にして Microphone.onActivity(true) を呼び出すのに必要な音量を指定する整数。指定できる値は 0 ～ 100 です。デフォルト値は 10 です。

timeOut:Number (オプション) - アクティビティがない場合に、サウンドが停止したと判定して Microphone.onActivity(false) イベントハンドラを呼び出すまでの経過時間をミリ秒単位で指定する整数。デフォルト値は 2000 (2 秒) です。

例

次の例では、silenceLevel_nstep という NumericStepper インスタンスへのユーザー入力に基づいてサイレンスレベルを変更します。オーディオストリームが無音であるかどうかの判定に応じて、silenceLevel_pb という ProgressBar インスタンスの外観を変更します。無音でない場合は、オーディオストリームのアクティビティレベルを表示します。

```
var silenceLevel_pb:mx.controls.ProgressBar;
var silenceLevel_nstep:mx.controls.NumericStepper;

this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);

silenceLevel_pb.label = "Activity level: %3";
silenceLevel_pb.mode = "manual";
silenceLevel_nstep.minimum = 0;
silenceLevel_nstep.maximum = 100;
silenceLevel_nstep.value = active_mic.silenceLevel;

var nstepListener:Object = new Object();
nstepListener.change = function(evt:Object) {
  active_mic.setSilenceLevel(evt.target.value, active_mic.silenceTimeOut);
};
silenceLevel_nstep.addEventListener("change", nstepListener);
```

```
this.onEnterFrame = function() {
  silenceLevel_pb.setProgress(active_mic.activityLevel, 100);
};
active_mic.onActivity = function(active:Boolean) {
  if (active) {
  silenceLevel_pb.indeterminate = false;
  silenceLevel_pb.setStyle("themeColor", "haloGreen");
  silenceLevel_pb.label = "Activity level: %3";
  } else {
  silenceLevel_pb.indeterminate = true;
  silenceLevel_pb.setStyle("themeColor", "0xFF0000");
  silenceLevel_pb.label = "Activity level: (inactive)";
  }
};
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

setMotionLevel (Camera.setMotionLevel メソッド), activityLevel (Microphone.activityLevel プロパティ), onActivity (Microphone.onActivity ハンドラ), setGain (Microphone.setGain メソッド), silenceLevel (Microphone.silenceLevel プロパティ), silenceTimeOut (Microphone.silenceTimeOut プロパティ)

setUseEchoSuppression (Microphone.setUseEchoSuppression メソッド)

```
public setUseEchoSuppression(useEchoSuppression:Boolean) : Void
```

オーディオコーデックのエコー抑制機能を使用するかどうかを指定します。[Macromedia Flash Player 設定] パネルの [マイク] でユーザーが [エコーを減らす] を選択していない限り、デフォルト値は false です。

エコー抑制とは、スピーカーから出たサウンドが同じコンピュータのマイクによって拾われるオーディオフィードバックの影響を減らす機能のことです。これは、フィードバックを完全に除去するエコー除去機能とは異なります。

通常、キャプチャするサウンドをヘッドセットではなく同じコンピュータのスピーカーで再生する場合には、エコー抑制機能の使用をお勧めします。サウンド出力デバイスをユーザーが選択できるようにした SWF ファイルでは、ユーザーがスピーカーを使用し、マイクも使用する場合に Microphone.setUseEchoSuppression(true) を呼び出すことをお勧めします。

ユーザーが、[Macromedia Flash Player 設定] パネルの [マイク] でこれらの設定を調整することもできます。

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

`useEchoSuppression:Boolean` - エコー抑制を使用するか (`true`) 使用しないか (`false`) を示すブール値。

例

次の例では、ユーザーが `useEchoSuppression_ch` という CheckBox インスタンスを選択したときに、エコー抑制機能をオンにします。`activityLevel_pb` という ProgressBar インスタンスに、現在のオーディオストリームのアクティビティレベルを表示します。

```
var useEchoSuppression_ch:mx.controls.CheckBox;
var activityLevel_pb:mx.controls.ProgressBar;

this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);

activityLevel_pb.mode = "manual";
activityLevel_pb.label = "Activity Level: %3";
useEchoSuppression_ch.selected = active_mic.useEchoSuppression;
this.onEnterFrame = function() {
  activityLevel_pb.setProgress(active_mic.activityLevel, 100);
};
var chListener:Object = new Object();
chListener.click = function(evt:Object) {
  active_mic.setUseEchoSuppression(evt.target.selected);
};
useEchoSuppression_ch.addEventListener("click", chListener);
```

この例で使用している `MovieClip.getNextHighestDepth()` メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、`MovieClip.getNextHighestDepth()` メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

`setUseEchoSuppression (Microphone.setUseEchoSuppression メソッド )`, `useEchoSuppression (Microphone.useEchoSuppression プロパティ )`

silenceLevel (Microphone.silenceLevel プロパティ)

public silenceLevel : Number (読み取り専用)

マイクを有効にして Microphone.onActivity(true) を呼び出すのに必要な音量を指定する整数です。デフォルト値は 10 です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、silenceLevel_nstep という NumericStepper インスタンスへのユーザー入力に基づいてサイレンスレベルを変更します。オーディオストリームが無音であるかどうかの判定に応じて、silenceLevel_pb という ProgressBar インスタンスの外観を変更します。無音でない場合は、オーディオストリームのアクティビティレベルを表示します。

```
var silenceLevel_pb:mx.controls.ProgressBar;
var silenceLevel_nstep:mx.controls.NumericStepper;

this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);

silenceLevel_pb.label = "Activity level: %3";
silenceLevel_pb.mode = "manual";
silenceLevel_nstep.minimum = 0;
silenceLevel_nstep.maximum = 100;
silenceLevel_nstep.value = active_mic.silenceLevel;

var nstepListener:Object = new Object();
nstepListener.change = function(evt:Object) {
  active_mic.setSilenceLevel(evt.target.value, active_mic.silenceTimeOut);
};
silenceLevel_nstep.addEventListener("change", nstepListener);

this.onEnterFrame = function() {
  silenceLevel_pb.setProgress(active_mic.activityLevel, 100);
};
active_mic.onActivity = function(active:Boolean) {
  if (active) {
  silenceLevel_pb.indeterminate = false;
  silenceLevel_pb.setStyle("themeColor", "haloGreen");
  silenceLevel_pb.label = "Activity level: %3";
  } else {
  silenceLevel_pb.indeterminate = true;
  silenceLevel_pb.setStyle("themeColor", "0xFF0000");
  silenceLevel_pb.label = "Activity level: (inactive)";
  }
};
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

gain (Microphone.gain プロパティ), setSilenceLevel (Microphone.setSilenceLevel メソッド)

silenceTimeOut (Microphone.silenceTimeOut プロパティ)

public silenceTimeOut : Number (読み取り専用)

マイクによるサウンド検知が停止してから Microphone.onActivity(false) を呼び出すまでの経過時間をミリ秒単位で表した数値です。デフォルト値は 2000 (2 秒) です。

この値を設定するには、Microphone.setSilenceLevel() を使います。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、マイクによるサウンド検知が停止してから、Microphone.onActivity(false) を呼び出すまでの時間をユーザーが制御できるようにします。ユーザーは、silenceTimeOut_nstep という NumericStepper インスタンスを使用してこの値を制御します。オーディオストリームが無音であるかどうかの判定に応じて、silenceLevel_pb という ProgressBar インスタンスの外観を変更します。無音でない場合は、オーディオストリームのアクティビティレベルを表示します。

```
var silenceLevel_pb:mx.controls.ProgressBar;
var silenceTimeOut_nstep:mx.controls.NumericStepper;

this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);

silenceLevel_pb.label = "Activity level: %3";
silenceLevel_pb.mode = "manual";
silenceTimeOut_nstep.minimum = 0;
silenceTimeOut_nstep.maximum = 10;
silenceTimeOut_nstep.value = active_mic.silenceTimeOut/1000;

var nstepListener:Object = new Object();
nstepListener.change = function(evt:Object) {
  active_mic.setSilenceLevel(active_mic.silenceLevel, evt.target.value 1000);
};
silenceTimeOut_nstep.addEventListener("change", nstepListener);
```

```
this.onEnterFrame = function() {
  silenceLevel_pb.setProgress(active_mic.activityLevel, 100);
};
active_mic.onActivity = function(active:Boolean) {
  if (active) {
  silenceLevel_pb.indeterminate = false;
  silenceLevel_pb.setStyle("themeColor", "haloGreen");
  silenceLevel_pb.label = "Activity level: %3";
  } else {
  silenceLevel_pb.indeterminate = true;
  silenceLevel_pb.setStyle("themeColor", "0xFF0000");
  silenceLevel_pb.label = "Activity level: (inactive)";
  }
};
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

setSilenceLevel (Microphone.setSilenceLevel メソッド)

useEchoSuppression (Microphone.useEchoSuppression プロパティ)

public useEchoSuppression : Boolean (読み取り専用)

読み取り専用プロパティ。エコー抑制が有効になっている場合は true、それ以外の場合は false を示すブール値です。[Macromedia Flash Player 設定] パネルの [マイク] でユーザーが [エコーを減らす] を選択していない限り、デフォルト値は false です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ユーザーが useEchoSuppression_ch という CheckBox インスタンスを選択したときに、エコー抑制機能をオンにします。activityLevel_pb という ProgressBar インスタンスに、現在のオーディオストリームのアクティビティレベルを表示します。

```
var useEchoSuppression_ch:mx.controls.CheckBox;
var activityLevel_pb:mx.controls.ProgressBar;

this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
```

```
activityLevel_pb.mode = "manual";
activityLevel_pb.label = "Activity Level: %3";
useEchoSuppression_ch.selected = active_mic.useEchoSuppression;
this.onEnterFrame = function() {
  activityLevel_pb.setProgress(active_mic.activityLevel, 100);
};
var chListener:Object = new Object();
chListener.click = function(evt:Object) {
  active_mic.setUseEchoSuppression(evt.target.selected);
};
useEchoSuppression_ch.addEventListener("click", chListener);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

setUseEchoSuppression (Microphone.setUseEchoSuppression メソッド)

Mouse

```
Object
  |
  +-Mouse
```

```
public class Mouse
extends Object
```

Mouse クラスはトップレベルのクラスで、コンストラクタを実行しなくても、そのメソッドやプロパティを使用できます。Mouse クラスのメソッドを使って、SWF ファイルでマウスポインタ (カーソル) を非表示にしたり表示したりできます。デフォルトではマウスポインタが表示されますが、ポインタを非表示にし、ムービークリップで作成したカスタムポインタを使用することもできます。

Flash アプリケーションが監視できるのは、そのフォーカス内で発生するマウスイベントのみです。Flash アプリケーションは、別のアプリケーションでのマウスイベントを検出できません。

対応バージョン : ActionScript 1.0、Flash Player 5

プロパティ一覧

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベント一覧

イベント	説明
onMouseDown = function() {}	マウスボタンが押されると通知されます。
onMouseMove = function() {}	マウスポインタが移動すると通知されます。
onMouseUp = function() {}	マウスボタンが離されると通知されます。
onMouseWheel = function([delta: Number], [scrollTarget: Object]) {}	マウスホイールを回転させると通知されます。

メソッド一覧

オプション	署名	説明
static	addListener (listener:Object) : Void	onMouseDown、onMouseMove、onMouseUp、onMouseWheel の各リスナーの通知を受け取るオブジェクトを登録します。
static	hide() : Number	SWF ファイル内でマウスポインタを非表示にします。
static	removeListener (listener:Object) : Boolean	以前に addListener() を使用して登録したオブジェクトを削除します。
static	show() : Number	SWF ファイル内でマウスポインタを表示します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

addListener (Mouse.addListener メソッド)

```
public static addListener(listener:Object) : Void
```

onMouseDown、onMouseMove、onMouseUp、onMouseWheel の各リスナーの通知を受け取るオブジェクトを登録します。onMouseWheel リスナーは Windows でのみサポートされます。

listener パラメータには、少なくとも 1 つのリスナーに対して定義済みメソッドを持つオブジェクトを含む必要があります。

マウスを押すか、移動するか、離すか、またはマウスでスクロールすると、フォーカスの有無に関係なく、このメソッドで登録したすべてのリスナーオブジェクトの onMouseDown メソッド、onMouseMove メソッド、onMouseUp メソッド、および onMouseWheel メソッドが呼び出されます。複数のオブジェクトがマウス通知を受け取ることができます。listener が登録済みである場合、変化は起きません。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

`listener:Object` - オブジェクト。

例

この例は、"ActionScript" サンプルフォルダにある "*animation.fla*" ファイルから抜粋したものです。

```
// Create a mouse listener object
var mouseListener:Object = new Object();

// Every time the mouse cursor moves within the SWF file,
   update the position of the crosshair movie clip
   instance on the Stage.
mouseListener.onMouseMove = function() {
   crosshair_mc._x = _xmouse;
   crosshair_mc._y = _ymouse;
};

// When you click the mouse, check to see if the cursor is within the boundaries
   of the Stage. If so, increment the number of shots.
mouseListener.onMouseDown = function() {
   if (bg_mc.hitTest(_xmouse, _ymouse, false)) {
   _global.shots++;
   }
};
Mouse.addListener(mouseListener);
```

スクリプト全体を表示するには、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Animation" フォルダに移動して animation.fla ファイルにアクセスします。

関連項目

onMouseDown (Mouse.onMouseDown イベントリスナー), onMouseMove (Mouse.onMouseMove イベントリスナー), onMouseUp (Mouse.onMouseUp イベントリスナー), onMouseWheel (Mouse.onMouseWheel イベントリスナー)

hide (Mouse.hide メソッド)

```
public static hide() : Number
```

SWF ファイル内でマウスポインタを非表示にします。デフォルトでは、ポインタは表示されます。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Number - 0 または 1 の整数値。Mouse.hide(), の呼び出し前にマウスポインタが非表示にされていた場合は 0 を返します。Mouse.hide() の呼び出し前にマウスポインタが表示されていた場合は 1 を返します。

例

次のコードは、標準のマウスポインタを非表示にし、pointer_mc ムービークリップインスタンスの x 位置および y 位置を、ポインタ位置 x および y に設定します。ムービークリップを作成し、そのリンケージ識別子を pointer_id に設定します。タイムラインのフレーム 1 に次の ActionScript を追加します。

```
// to use this script you need a symbol
// in your library with a Linkage Identifier of "pointer_id".
this.attachMovie("pointer_id", "pointer_mc", this.getNextHighestDepth());
Mouse.hide();
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  pointer_mc._x = _xmouse;
  pointer_mc._y = _ymouse;
  updateAfterEvent();
};
Mouse.addListener(mouseListener);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

show (Mouse.show メソッド), _xmouse (MovieClip._xmouse プロパティ), _ymouse (MovieClip._ymouse プロパティ)

onMouseDown (Mouse.onMouseDown イベントリスナー)

```
onMouseDown = function() {}
```

マウスボタンが押されると通知されます。onMouseDown リスナーを使用するには、リスナーオブジェクトを作成します。その後、onMouseDown の関数を定義し、addListener() メソッドを使用してリスナーを Mouse オブジェクトに登録します。次に例を示します。

```
var someListener:Object = new Object();
someListener.onMouseDown = function () { ... };
Mouse.addListener(someListener);
```

1 つのイベントに関する通知を複数のリスナーで受け取ることができるので、リスナーごとに異なる複数のコードを作成して連携させることもできます。

Flash アプリケーションが監視できるのは、そのフォーカス内で発生するマウスイベントのみです。Flash アプリケーションは、別のアプリケーションでのマウスイベントを検出できません。

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の例では、Drawing API を使用して、ユーザーが実行時にマウスをクリック、ドラッグ、解放するたびに矩形を描画します。

```
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.isDrawing = true;
  this.orig_x = _xmouse;
  this.orig_y = _ymouse;
  this.target_mc = canvas_mc.createEmptyMovieClip("",
  canvas_mc.getNextHighestDepth());
};
mouseListener.onMouseMove = function() {
  if (this.isDrawing) {
    this.target_mc.clear();
    this.target_mc.lineStyle(1, 0xFF0000, 100);
    this.target_mc.moveTo(this.orig_x, this.orig_y);
    this.target_mc.lineTo(_xmouse, this.orig_y);
    this.target_mc.lineTo(_xmouse, _ymouse);
    this.target_mc.lineTo(this.orig_x, _ymouse);
    this.target_mc.lineTo(this.orig_x, this.orig_y);
  }
  updateAfterEvent();
};
mouseListener.onMouseUp = function() {
  this.isDrawing = false;
};
Mouse.addListener(mouseListener);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

addListener (Mouse.addListener メソッド)

onMouseMove (Mouse.onMouseMove イベントリスナー)

```
onMouseMove = function() {}
```

マウスポインタが移動すると通知されます。onMouseMove リスナーを使用するには、リスナーオブジェクトを作成します。その後、onMouseMove の関数を定義し、addListener() メソッドを使用してリスナーを Mouse オブジェクトに登録します。次に例を示します。

```
var someListener:Object = new Object();
someListener.onMouseMove = function () { ... };
Mouse.addListener(someListener);
```

1 つのイベントに関する通知を複数のリスナーで受け取ることができるので、リスナーごとに異なる複数のコードを作成して連携させることもできます。

Flash アプリケーションが監視できるのは、そのフォーカス内で発生するマウスイベントのみです。Flash アプリケーションは、別のアプリケーションでのマウスイベントを検出できません。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、マウスポインタをツールとして使用し、onMouseMove と Drawing API を使用して線を描画します。ユーザーがマウスカーソルをドラッグすると、線が描かれます。

```
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.isDrawing = true;
  canvas_mc.lineStyle(2, 0xFF0000, 100);
  canvas_mc.moveTo(_xmouse, _ymouse);
};
```

```
mouseListener.onMouseMove = function() {
  if (this.isDrawing) {
    canvas_mc.lineTo(_xmouse, _ymouse);
  }
  updateAfterEvent();
};
mouseListener.onMouseUp = function() {
  this.isDrawing = false;
};
Mouse.addListener(mouseListener);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

次の例では、標準のマウスポインタを非表示にし、pointer_mc ムービークリップインスタンスの *x* 位置および *y* 位置を、ポインタ位置 *x* および *y* に設定します。ムービークリップを作成し、そのリンケージ識別子を pointer_id に設定します。タイムラインのフレーム 1 に次の ActionScript を追加します。

```
// to use this script you need a symbol
// in your library with a Linkage Identifier of "pointer_id".
this.attachMovie("pointer_id", "pointer_mc", this.getNextHighestDepth());
Mouse.hide();
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  pointer_mc._x = _xmouse;
  pointer_mc._y = _ymouse;
  updateAfterEvent();
};
Mouse.addListener(mouseListener);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

addListener (Mouse.addListener メソッド)

onMouseUp (Mouse.onMouseUp イベントリスナー)

```
onMouseUp = function() {}
```

マウスボタンが離されると通知されます。onMouseUp リスナーを使用するには、リスナーオブジェクトを作成します。その後、onMouseUp の関数を定義し、addListener() メソッドを使用してリスナーを Mouse オブジェクトに登録します。次に例を示します。

```
var someListener:Object = new Object();
someListener.onMouseUp = function () { ... };
Mouse.addListener(someListener);
```

1 つのイベントに関する通知を複数のリスナーで受け取ることができるので、リスナーごとに異なる複数のコードを作成して連携させることもできます。

Flash アプリケーションが監視できるのは、そのフォーカス内で発生するマウスイベントのみです。Flash アプリケーションは、別のアプリケーションでのマウスイベントを検出できません。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、マウスポインタをツールとして使用し、onMouseMove と Drawing API を使用して線を描画します。ユーザーがマウスカーソルをドラッグすると、線が描かれます。ユーザーがマウスボタンを離すと、線の描画が停止します。

```
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.isDrawing = true;
  canvas_mc.lineStyle(2, 0xFF0000, 100);
  canvas_mc.moveTo(_xmouse, _ymouse);
};
mouseListener.onMouseMove = function() {
  if (this.isDrawing) {
    canvas_mc.lineTo(_xmouse, _ymouse);
  }
  updateAfterEvent();
};
mouseListener.onMouseUp = function() {
  this.isDrawing = false;
};
Mouse.addListener(mouseListener);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

addListener (Mouse.addListener メソッド)

onMouseWheel (Mouse.onMouseWheel イベントリスナー)

```
onMouseWheel = function([delta:Number], [scrollTarget:Object]) {}
```

マウスホイールを回転させると通知されます。onMouseWheel リスナーを使用するには、リスナーオブジェクトを作成します。その後、onMouseWheel の関数を定義し、addListener( ) を使用してリスナーを Mouse オブジェクトに登録します。

メモ : マウスホイールイベントリスナーは、Windows バージョンの Flash Player でのみ利用できます。

Flash アプリケーションが監視できるのは、そのフォーカス内で発生するマウスイベントのみです。Flash アプリケーションは、別のアプリケーションでのマウスイベントを検出できません。

対応バージョン : ActionScript 1.0、Flash Player 6 - (Windows のみ)。

パラメータ

delta:Number (オプション) - ユーザーがマウスホイールを 1 目盛り回すごとにスクロールする行数を示す数値。正の delta 値は上方向へのスクロールを表します。負の値は下方向へのスクロールを表します。通常の値は 1 ～ 3 の範囲ですが、ホイールの回転が速くなると、delta の値は大きくなります。

scrollTarget:Object (オプション) - マウスホイールを回転させたときに、マウスポインタが置かれた位置の一番上にあるムービークリップまたはオブジェクトインスタンスを示すパラメータ。delta の値を指定しないで、scrollTarget の値を指定する場合は、delta に null を指定します。

例

次の例では、マウスホイールイベントに応答するリスナーオブジェクトの作成方法を示します。この例では、ユーザーがマウスホイールを回転するたびに、clip_mc というムービークリップオブジェクトの *x* 座標が変更されます。

```
var mouseListener:Object = new Object();
mouseListener.onMouseWheel = function(delta) {
  clip_mc._x += delta;
}
Mouse.addListener(mouseListener);
```

次の例では、マウスホイールを回転すると回転する線を描画します。実行時に SWF ファイルをクリックしてからマウスホイールを回転することでムービークリップの動きを確認します。

```
this.createEmptyMovieClip("line_mc", this.getNextHighestDepth());
line_mc.lineStyle(2, 0xFF0000, 100);
line_mc.moveTo(0, 100);
line_mc.lineTo(0, 0);
line_mc._x = 200;
line_mc._y = 200;

var mouseListener:Object = new Object();
mouseListener.onMouseWheel = function(delta:Number) {
  line_mc._rotation += delta;
};
mouseListener.onMouseDown = function() {
  trace("Down");
};
Mouse.addListener(mouseListener);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

addListener (Mouse.addListener メソッド), mouseWheelEnabled (TextField.mouseWheelEnabled プロパティ)

removeListener (Mouse.removeListener メソッド)

```
public static removeListener(listener:Object) : Boolean
```

以前に addListener() を使用して登録したオブジェクトを削除します。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

listener:Object - オブジェクト。

戻り値

Boolean - listener オブジェクトが正常に削除されると、true を返します。listener が正常に削除されない場合、たとえば、listener が Mouse オブジェクトのリスナーリストに登録されていない場合などは、false を返します。

例

次の例では、ステージに 3 つのボタンを割り当てて、実行時にユーザーがマウスポインタを使用して SWF ファイルで線を描くことができるようにします。ボタンの 1 つは、SWF ファイルからすべての線を消去します。2 番目のボタンは、マウスリスナーを削除して、ユーザーが線を描けないようにします。3 番目のボタンは、削除したマウスリスナーを追加して、ユーザーが再び線を描けるようにします。タイムラインのフレーム 1 に次の ActionScript を追加します。

```
this.createClassObject(mx.controls.Button, "clear_button",
  this.getNextHighestDepth(), {_x:10, _y:10, label:'clear'});
this.createClassObject(mx.controls.Button, "stopDrawing_button",
  this.getNextHighestDepth(), {_x:120, _y:10, label:'stop drawing'});
this.createClassObject(mx.controls.Button, "startDrawing_button",
  this.getNextHighestDepth(), {_x:230, _y:10, label:'start drawing'});
startDrawing_button.enabled = false;
//
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.isDrawing = true;
  canvas_mc.lineStyle(2, 0xFF0000, 100);
  canvas_mc.moveTo(_xmouse, _ymouse);
};
mouseListener.onMouseMove = function() {
  if (this.isDrawing) {
  canvas_mc.lineTo(_xmouse, _ymouse);
  }
  updateAfterEvent();
};
mouseListener.onMouseUp = function() {
  this.isDrawing = false;
};
Mouse.addListener(mouseListener);
var clearListener:Object = new Object();
clearListener.click = function() {
  canvas_mc.clear();
};
clear_button.addEventListener("click", clearListener);
//
var stopDrawingListener:Object = new Object();
stopDrawingListener.click = function(evt:Object) {
  Mouse.removeListener(mouseListener);
  evt.target.enabled = false;
  startDrawing_button.enabled = true;
};
stopDrawing_button.addEventListener("click", stopDrawingListener);
var startDrawingListener:Object = new Object();
startDrawingListener.click = function(evt:Object) {
  Mouse.addListener(mouseListener);
  evt.target.enabled = false;
```

```
  stopDrawing_button.enabled = true;
};
startDrawing_button.addEventListener("click", startDrawingListener);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

show (Mouse.show メソッド)

```
public static show() : Number
```

SWF ファイル内でマウスポインタを表示します。デフォルトでは、ポインタは表示されます。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Number - 0 または 1 の整数値。Mouse.show(), の呼び出し前にマウスポインタが非表示にされていた場合は 0 を返します。Mouse.show() の呼び出し前にマウスポインタが表示されていた場合は 1 を返します。

例

次の例では、ムービークリップ my_mc の上にマウスカーソルが移動されたときに、ライブラリからカスタムカーソルを割り当てます。ライブラリ内のムービークリップにリンケージ識別子 cursor_help_id を設定し、次の ActionScript をタイムラインのフレーム 1 に追加します。

```
my_mc.onRollOver = function() {
  Mouse.hide();
  this.attachMovie("cursor_help_id", "cursor_mc", this.getNextHighestDepth(),
  {_x:this._xmouse, _y:this._ymouse});
};
my_mc.onMouseMove = function() {
  this.cursor_mc._x = this._xmouse;
  this.cursor_mc._y = this._ymouse;
};
my_mc.onRollOut = function() {
  Mouse.show();
  this.cursor_mc.removeMovieClip();
};
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

hide (Mouse.hide メソッド), _xmouse (MovieClip._xmouse プロパティ), _ymouse (MovieClip._ymouse プロパティ)

MovieClip

```
Object
  |
  +-MovieClip
```

```
public dynamic class MovieClip
extends Object
```

MovieClip クラスのメソッドは、ムービークリップをターゲットとするアクションと同じ機能を提供します。[アクション] パネルの [アクション] ツールボックスに同等のアクションを持たない他のメソッドもあります。

新しいムービークリップを作成する場合は、コンストラクタメソッドを使用しません。ムービークリップインスタンスは、次の 3 つのいずれかのメソッドを使用して作成します。

- attachMovie() メソッドを使用すると、ライブラリ内に存在するムービークリップシンボルに基づいて、ムービークリップインスタンスを作成できます。
- createEmptyMovieClip() メソッドを使用すると、別のムービークリップに基づいて、空のムービークリップインスタンスを子として作成できます。
- duplicateMovieClip() メソッドを使用すると、別のムービークリップに基づいて、ムービークリップインスタンスを作成できます。

MovieClip クラスのメソッドを呼び出すには、次のシンタックスを使用して、ムービークリップインスタンスを名前で参照します。このシンタックスでは、*my_mc* がムービークリップインスタンスです。

```
my_mc.play();
my_mc.gotoAndPlay(3);
```

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

モーショントゥイーンが含まれている MovieClip オブジェクトの次のいずれかのプロパティを変更した場合、Flash Player によって、その MovieClip オブジェクトの再生ヘッドが停止されます。該当するプロパティは、_alpha、blendMode、filters、_height、opaqueBackground、_rotation、scale9Grid、scrollRect、transform、_visible、_width、_x、_xscale、_y、または _yscale です。ただし、その MovieClip オブジェクトの子 MovieClip オブジェクトの再生ヘッドは停止しません。

対応バージョン： ActionScript 1.0、Flash Player 3

プロパティ一覧

オプション	プロパティ	説明
	_alpha:Number	ムービークリップのアルファ透明度。
	blendMode:Object	このムービークリップのブレンドモード。
	cacheAsBitmap: Boolean	true に設定されている場合、ムービークリップの内部ビットマップ表現がキャッシュされます。
	_currentframe: Number (読み取り専用)	ムービークリップのタイムライン内で再生ヘッドがあるフレーム番号を返します。
	_droptarget:String (読み取り専用)	このムービークリップがドロップされたムービークリップインスタンスの絶対パスを、スラッシュシンタックス表記で返します。
	enabled:Boolean	ムービークリップの有効 / 無効を示すブール値。
	filters:Array	ムービークリップに現在関連付けられている各フィルタオブジェクトが格納されている、インデックスの配列。
	focusEnabled:Boolean	プログラムで Selection.setFocus() を使用してムービークリップにフォーカスを割り当てることができるかどうかを指定します。
	_focusrect:Boolean	ムービークリップにフォーカスがあるときに、その周囲に黄色の矩形を表示するかどうかを指定するブール値。
	forceSmoothing: Boolean	loadMovie() メソッドを使用して追加されたイメージのうち、ムービークリップと同じ階層レベルにあるイメージが拡大・縮小されるときに、そのイメージをスムージングするかどうかを決定するブール値。
	_framesloaded: Number (読み取り専用)	ストリーミング SWF ファイルからロードされたフレーム数。
	_height:Number	ピクセル単位で示したムービークリップの高さ。
	_highquality:Number	**非推奨** Flash Player 7 以降では使用しないでください。このプロパティの代わりに MovieClip._quality を使用します。 現在の SWF ファイルに適用されるアンチエイリアスのレベルを指定します。
	hitArea:Object	ムービークリップのヒット領域となる別のムービークリップを指定します。
	_lockroot:Boolean	SWF ファイルがムービークリップにロードされたときに、_root で参照する内容を指定するブール値。

オプション	プロパティ	説明
	menu:ContextMenu	指定された ContextMenu オブジェクトをムービークリップに関連付けます。
	_name:String	ムービークリップのインスタンス名。
	opaqueBackground:Number	数値 (16 進数の RGB 値) で指定されたムービークリップの不透明 (透明でない) な背景色。
	_parent:MovieClip	現在のムービークリップまたはオブジェクトを含むムービークリップまたはオブジェクトを指す参照。
	_quality:String	SWF ファイルに使用するレンダリング品質を設定または取得します。
	_rotation:Number	ムービークリップの元の位置からの回転角度を度単位で指定します。
	scale9Grid:Rectangle	ムービークリップの 9 つの拡大・縮小領域を定義する矩形の領域。
	scrollRect:Object	scrollRect プロパティを使用すると、ムービークリップのコンテンツをすばやくスクロールして、より多くのコンテンツを表示できるウィンドウを利用できるようになります。
	_soundbuftime:Number	サウンドのストリーミングを開始するまでにサウンドをプリバッファする秒数を指定します。
	tabChildren:Boolean	ムービークリップの子が自動タブ順に含まれているかどうかを判別します。
	tabEnabled:Boolean	ムービークリップが自動タブ順に含まれるかどうかを示します。
	tabIndex:Number	ムービー内のオブジェクトのタブ順をカスタマイズできます。
	_target:String (読み取り専用)	ムービークリップインスタンスのターゲットパスをスラッシュ表記で返します。
	_totalframes:Number (読み取り専用)	ムービークリップインスタンス内のフレーム総数です。
	trackAsMenu:Boolean	他のボタンまたはムービークリップがマウスの解放イベントを受け取ることができるかどうかを示すブール値です。
	transform:Transform	ムービークリップのマトリックス、カラー変換、ピクセル境界に関係するプロパティを持つオブジェクト。

オプション	プロパティ	説明
	_url:String (読み取り専用)	ムービークリップのダウンロード元である SWF、JPEG、GIF、または PNG の各ファイルの URL を取得します。
	useHandCursor: Boolean	ムービークリップ上にマウスが移動したときに、指差しハンドポインタ (ハンドカーソル) を表示するかどうかを示すブール値。
	_visible:Boolean	ムービークリップを表示するかどうかを示すブール値。
	_width:Number	ピクセル単位で示したムービークリップの幅。
	_x:Number	親ムービークリップのローカル座標を基準にしてムービークリップの *x* 座標を設定する整数。
	_xmouse:Number (読み取り専用)	マウス位置の *x* 座標を返します。
	_xscale:Number	ムービークリップの基準点から適用するムービークリップの水平スケール (*percentage*) を決定します。
	_y:Number	親ムービークリップのローカル座標を基準にしてムービークリップの *y* 座標を設定します。
	_ymouse:Number (読み取り専用)	マウス位置の *y* 座標を示します。
	_yscale:Number	ムービークリップの基準点から適用するムービークリップの垂直スケール (*percentage*) を設定します。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベントの一覧

イベント	説明
onData = function() {}	ムービークリップが MovieClip.loadVariables() の呼び出しからデータを受け取ったときに呼び出されます。
onDragOut = function() {}	マウスボタンが押された状態で、ポインタがオブジェクトの外に移動したときに呼び出されます。
onDragOver = function() {}	ポインタがムービークリップ外にドラッグされた後で、ムービークリップ上に移動したときに呼び出されます。

イベント	説明
`onEnterFrame = function() {}`	SWF ファイルのフレームレートで繰り返し呼び出されます。
`onKeyDown = function() {}`	ムービークリップにフォーカスがあるときにキーを押すと、呼び出されます。
`onKeyUp = function() {}`	キーを離すと呼び出されます。
`onKillFocus = function(newFocus: Object) {}`	ムービークリップがフォーカスを失うと、呼び出されます。
`onLoad = function() {}`	ムービークリップのインスタンスが作成されてタイムラインに読み込まれると、呼び出されます。
`onMouseDown = function() {}`	マウスボタンが押されると、呼び出されます。
`onMouseMove = function() {}`	マウスポインタが移動すると、呼び出されます。
`onMouseUp = function() {}`	マウスボタンが離されると、呼び出されます。
`onPress = function() {}`	ポインタがムービークリップ上にあるときにマウスをクリックすると、呼び出されます。
`onRelease = function() {}`	ムービークリップの上でマウスボタンを離すと、呼び出されます。
`onReleaseOutside = function() {}`	マウスボタンをムービークリップ領域内で押して、ムービークリップ領域の外側で離したときに呼び出されます。
`onRollOut = function() {}`	ポインタをムービークリップ領域の外側に移動すると、呼び出されます。
`onRollOver = function() {}`	ポインタをムービークリップ領域の上に移動すると、呼び出されます。
`onSetFocus = function(oldFocus:Object) {}`	ムービークリップがキーボードフォーカスを受け取ると、呼び出されます。
`onUnload = function() {}`	ムービークリップをタイムラインから削除した後に表示される最初のフレームで呼び出されます。

メソッド一覧

オプション	署名	説明
	attachAudio(id: Object) : Void	再生するオーディオソースを指定します。
	attachBitmap (bmp:BitmapData, depth:Number, [pixelSnapping: String], [smoothing:Boolean]) : Void	ビットマップイメージをムービークリップに割り当てます。
	attachMovie(id: String, name:String, depth:Number, [initObject:Object]) : MovieClip	ライブラリからシンボルを取得し、ムービークリップに割り当てます。
	beginBitmapFill (bmp:BitmapData, [matrix:Matrix], [repeat:Boolean], [smoothing:Boolean]) : Void	描画領域をビットマップイメージで塗りつぶします。
	beginFill(rgb:Number, [alpha:Number]) : Void	新しい描画パスの始点を示します。
	beginGradientFill (fillType:String, colors:Array, alphas:Array, ratios:Array, matrix:Object, [spreadMethod:String], [interpolationMethod: String], [focalPointRatio: Number]) : Void	新しい描画パスの始点を示します。
	clear() : Void	MovieClip.lineStyle() で指定された線スタイルを含め、ムービークリップの描画メソッドを使用して実行時に作成されたすべてのグラフィックを削除します。

オプション	署名	説明
	createEmptyMovieClip (name:String, depth:Number) : MovieClip	既存のムービークリップの子として空のムービークリップを作成します。
	createTextField (instanceName:String, depth:Number, x:Number, y:Number, width:Number, height:Number) : TextField	このメソッドを呼び出したムービークリップの子となる空のテキストフィールドを新しく作成します。
	curveTo(controlX: Number, controlY:Number, anchorX:Number, anchorY:Number) : Void	(controlX, controlY) で指定されたコントロールポイントを使用し、現在の描画位置から (anchorX, anchorY) まで、現在の線のスタイルで曲線を描画します。
	duplicateMovieClip (name:String, depth:Number, [initObject:Object]) : MovieClip	SWF ファイルの再生中に、指定されたムービークリップのインスタンスを作成します。
	endFill() : Void	beginFill() メソッドまたは beginGradientFill() メソッドへの最後の呼び出し以降に追加された線と曲線に塗りを適用します。
	getBounds(bounds: Object) : Object	bounds パラメータに基づいて、ムービークリップの最小および最大の *x* 座標と *y* 座標を示すプロパティを返します。
	getBytesLoaded() : Number	ムービークリップについて、既にロード (ストリーミング) されたバイト数を返します。
	getBytesTotal() : Number	ムービークリップのサイズをバイト単位で返します。
	getDepth() : Number	ムービークリップインスタンスの深度を返します。
	getInstanceAtDepth (depth:Number) : MovieClip	特定の深度にムービークリップが既に配置されているかどうかを判別します。

オプション	署名	説明
	getNextHighestDepth() : Number	現在のムービークリップ内の同一レベル、同一レイヤー上の他のすべてのオブジェクトよりもムービークリップを前面に表示する場合に、MovieClip.attachMovie() メソッド、MovieClip.duplicateMovieClip() メソッド、MovieClip.createEmptyMovieClip() メソッドに渡す深度値を調べることができます。
	getRect(bounds: Object) : Object	シェイプ上の線を除き、bounds パラメータに基づいて、ムービークリップの最小および最大の *x* 座標と *y* 座標を示すプロパティを返します。
	getSWFVersion() : Number	ムービークリップがパブリッシュされた時の対象の Flash Player のバージョンを示す整数値を返します。
	getTextSnapshot() : TextSnapshot	指定されたムービークリップのすべての静止テキストフィールド内のテキストを含む TextSnapshot オブジェクトを返します。ただし、子のムービークリップに存在するテキストは含まれません。
	getURL(url:String, [window:String], [method:String]) : Void	指定された URL から、指定されたウィンドウにドキュメントを読み込みます。
	globalToLocal (pt:Object) : Void	pt オブジェクトをステージ (グローバル) 座標からムービークリップ内 (ローカル) の座標に変換します。
	gotoAndPlay(frame: Object) : Void	指定されたフレームで SWF ファイルの再生を開始します。
	gotoAndStop(frame: Object) : Void	このムービークリップの指定されたフレームに再生ヘッドを送り、そこで停止させます。
	hitTest() : Boolean	ムービークリップを評価して、それが target または x 座標と y 座標のパラメータで示されるヒット領域と重なるかまたは交差するかを確認します。

オプション	署名	説明
	lineGradientStyle (fillType:String, colors:Array, alphas:Array, ratios:Array, matrix:Object, [spreadMethod: String], [interpolationMethod: String], [focalPointRatio: Number]) : Void	lineTo() メソッドおよび curveTo() メソッドへの今後の呼び出しに使用する線のスタイルを指定します。このスタイルは、次に lineStyle() メソッドまたは lineGradientStyle() メソッドを異なるパラメータで呼び出すまで使用されます。
	lineStyle(thickness: Number, rgb:Number, alpha:Number, pixelHinting:Boolean, noScale:String, capsStyle:String, jointStyle:String, miterLimit:Number) : Void	lineTo() メソッドおよび curveTo() メソッドで今後の呼び出しに使用する線のスタイルを指定します。このスタイルは、次に lineStyle() メソッドを異なるパラメータで呼び出すまで使用されます。
	lineTo(x:Number, y:Number) : Void	現在の描画位置から (x, y) まで、現在の線のスタイルを使用して線を描画します。その後で、現在の描画位置は (x, y) に設定されます。
	loadMovie(url:String, [method:String]) : Void	元の SWF ファイルの再生中に SWF ファイル、JPEG ファイル、GIF ファイル、または PNG ファイルを Flash Player のムービークリップ内にロードします。
	loadVariables(url: String, [method:String]) : Void	外部ファイルからデータを読み取り、ムービークリップの変数の値を設定します。
	localToGlobal (pt:Object) : Void	pt オブジェクトをムービークリップ内 (ローカル) の座標からステージ (グローバル) の座標に変換します。
	moveTo(x:Number, y:Number) : Void	現在の描画位置を (x, y) に移動します。
	nextFrame() : Void	次のフレームに再生ヘッドを送り、停止します。

オプション	署名	説明
	play() : Void	ムービークリップのタイムラインで再生ヘッドを移動します。
	prevFrame() : Void	直前のフレームに再生ヘッドを戻し、停止します。
	removeMovieClip() : Void	duplicateMovieClip()、MovieClip.duplicateMovieClip() MovieClip.createEmptyMovieClip() または MovieClip.attachMovie() で作成したムービークリップインスタンスを削除します。
	setMask(mc:Object) : Void	mc パラメータのムービークリップをマスクにして、呼び出し元のムービークリップを表示します。
	startDrag([lockCenter: Boolean], [left:Number], [top:Number], [right:Number], [bottom:Number]) : Void	指定されたムービークリップをユーザーがドラッグできるようにします。
	stop() : Void	再生中のムービークリップを停止します。
	stopDrag() : Void	MovieClip.startDrag() メソッドを終了します。
	swapDepths(target: Object) : Void	ムービークリップのスタッキング順序、つまり深度 (z 順序) を、target パラメータに指定したムービーと入れ替えます。または、target パラメータに指定した深度に現在置かれているムービーと入れ替えます。
	unloadMovie() : Void	ムービークリップインスタンスの内容を削除します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

_alpha (MovieClip._alpha プロパティ)

```
public _alpha : Number
```

ムービークリップのアルファ透明度。有効な値は 0 (完全な透明) ～ 100 (完全な不透明) です。デフォルト値は 100 です。ムービークリップ内で _alpha が 0 に設定されているオブジェクトは、非表示の場合でもアクティブです。たとえば、_alpha プロパティが 0 に設定されているムービークリップ内のボタンをクリックできます。ボタンを完全に無効にするには、ムービークリップの _visible プロパティを false に設定します。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 4

例

次のコードは、マウスがムービークリップ上を移動するときに動的に作成されるムービークリップ triangle の _alpha プロパティを 50% に設定します。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
this.createEmptyMovieClip("triangle", this.getNextHighestDepth());

triangle.beginFill(0x0000FF, 100);
triangle.moveTo(10, 10);
triangle.lineTo(10, 100);
triangle.lineTo(100, 10);
triangle.lineTo(10, 10);

triangle.onRollOver = function() {
  this._alpha = 50;
};
triangle.onRollOut = function() {
  this._alpha = 100;
};
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

_alpha (Button._alpha プロパティ), _alpha (TextField._alpha プロパティ), _visible (MovieClip._visible プロパティ)

attachAudio (MovieClip.attachAudio メソッド)

```
public attachAudio(id:Object) : Void
```

再生するオーディオソースを指定します。オーディオソースの再生を停止するには、id パラメータに false を指定します。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン： ActionScript 1.0、Flash Player 6 - Flash Video (FLV) ファイルからオーディオを割り当てる機能が Flash Player 7 で追加されました。

パラメータ

id:Object - 再生するオーディオを含むオブジェクト。有効な値は、Microphone オブジェクト、FLV ファイルを再生する NetStream オブジェクト、および false です。false を指定すると、オーディオの再生が停止します。

例

次の例では、新しい NetStream 接続を作成します。[ライブラリ] パネルを開き、[ライブラリ] パネルのオプションメニューから [新規ビデオ] を選択し、新しいビデオシンボルを追加します。シンボルのインスタンス名を my_video にします。実行時に FLV ビデオを動的にロードします。attachAudio() メソッドは、FLV ファイルからステージ上のムービークリップにオーディオを割り当てるときに使用します。オーディオを割り当てた後は、Sound クラス、および volUp_btn と volDown_btn の 2 つのボタンを使用してムービークリップ内でオーディオを制御できます。

```
var my_nc:NetConnection = new NetConnection();
my_nc.connect(null);
var my_ns:NetStream = new NetStream(my_nc);
my_video.attachVideo(my_ns);
my_ns.play("yourVideo.flv");
this.createEmptyMovieClip("flv_mc", this.getNextHighestDepth());
flv_mc.attachAudio(my_ns);
var audio_sound:Sound = new Sound(flv_mc);

// Add volume buttons.
volUp_btn.onRelease = function() {
  if (audio_sound.getVolume()<100) {
  audio_sound.setVolume(audio_sound.getVolume()+10);
  updateVolume();
  }
};
```

```
volDown_btn.onRelease = function() {
   if (audio_sound.getVolume()>0) {
   audio_sound.setVolume(audio_sound.getVolume()-10);
   updateVolume();
   }
};

// Updates the volume.
this.createTextField("volume_txt", this.getNextHighestDepth(), 0, 0, 100, 22);
updateVolume();
function updateVolume() {
   volume_txt.text = "Volume: "+audio_sound.getVolume();
}
```

次の例では、動的に作成されたムービークリップインスタンス audio_mc のオーディオソースとして、マイクを指定します。

```
var active_mic:Microphone = Microphone.get();
this.createEmptyMovieClip("audio_mc", this.getNextHighestDepth());
audio_mc.attachAudio(active_mic);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

マイク , play (NetStream.play メソッド), Sound, attachVideo (Video.attachVideo メソッド)

attachBitmap (MovieClip.attachBitmap メソッド)

```
public attachBitmap(bmp:BitmapData, depth:Number, [pixelSnapping:String],
   [smoothing:Boolean]) : Void
```

ビットマップイメージをムービークリップに割り当てます。

ビットマップをムービークリップに割り当てた後、ムービークリップからそのビットマップオブジェクトへの参照が作成されます。ビットマップを割り当てるとき、ビットマップの外見に影響を与える pixelSnapping パラメータおよび smoothing パラメータを指定できます。

オブジェクトをムービークリップに追加すると、そのオブジェクトにはアクセスできなくなります。depth パラメータ、pixelSnapping パラメータ、および smoothing パラメータは、attachBitmap() メソッド呼び出し中にのみ設定できますが、その後は変更できません。

最初に createEmptyMovieClip() メソッドを使用して空のムービークリップを作成し、次に attachBitmap() メソッドを使用します。これにより、たとえばムービークリップの matrix プロパティを使用して、ムービークリップに変形を適用し、ビットマップを変形できます。

ピクセルへの吸着では、ビットマップが配置される位置は、部分ピクセルの上ではなく、最も近い整数ピクセル値の位置になります。ピクセルへの吸着モードには、次の 3 つのモードがあります。

- auto モード。このモードでは、ビットマップを伸縮または回転しない限り、ピクセルへの吸着が行われます。
- always モード。このモードでは、ビットマップの伸縮または回転とは無関係に、常にピクセルへの吸着が行われます。
- never モード。このモードでは、ムービークリップのピクセルへの吸着がオフになります。

スムージングモードは、拡大・縮小時のイメージの外観に影響を与えます。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

`bmp:BitmapData` - 透明または不透明なビットマップイメージ。

`depth:Number` - ビットマップイメージを配置するムービークリップ内の深度を指定する整数。

`pixelSnapping:String ( オプション )` - ピクセルへの吸着モード。`auto`、`always`、`never` のいずれかを指定できます。デフォルトのモードは `auto` です。

`smoothing:Boolean ( オプション )` - スムージングモードを有効にする場合は `true`、無効にする場合は `false` を指定します。スムージングモードはデフォルトでは有効です。

例

次の例では、非常に基本的なビットマップをムービークリップに割り当てます。

```
import flash.display.*;

this.createEmptyMovieClip("bmp1", this.getNextHighestDepth());
var bmpData1:BitmapData = new BitmapData(200, 200, false, 0xaa3344);
bmp1.attachBitmap(bmpData1, 2, "auto", true);
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している `MovieClip.getNextHighestDepth()` メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

attachMovie (MovieClip.attachMovie メソッド)

```
public attachMovie(id:String, name:String, depth:Number,
  [initObject:Object]) : MovieClip
```

ライブラリからシンボルを取得し、ムービークリップに割り当てます。attachMovie( ) メソッドで割り当てた SWF ファイルを削除するには、MovieClip.removeMovieClip( ) または MovieClip.unloadMovie( ) を使用します。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`id:String` - ステージ上のムービークリップに割り当てるライブラリ内のムービークリップシンボルのリンケージ名。これは、[リンケージプロパティ] ダイアログボックス内の [識別子] フィールドに入力した名前です。

`name:String` - ムービークリップに割り当てられる一意のインスタンス名。

`depth:Number` - SWF ファイルが配置される深度を指定する整数。

`initObject:Object` (オプション) - 新しく割り当てられたムービークリップに設定するプロパティを含むオブジェクト。このパラメータは、Flash Player 6 以降でのみサポートされています。このパラメータを使用すると、動的に作成したムービークリップは、クリップのパラメータを受け取ることができます。オブジェクトではない initObject は無視されます。initObject のすべてのプロパティが新しいインスタンスにコピーされます。initObject で指定されたプロパティは、コンストラクタ関数で使用できます。

戻り値

`MovieClip` - 新しく作成したインスタンスへの参照。

例

次の例では、リンケージ識別子が linkageName であるシンボルを SWF ファイルのステージ上にあるムービークリップインスタンスに割り当てます。

```
this.attachMovie("linkageName", "instance1", this.getNextHighestDepth());
this.attachMovie("linkageName", "instance2", this.getNextHighestDepth(),
  {_x:100, _y:100});
```

この例で使用している MovieClip.getNextHighestDepth( ) メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth( ) メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

removeMovieClip (MovieClip.removeMovieClip メソッド), unloadMovie (MovieClip.unloadMovie メソッド), removeMovieClip 関数

beginBitmapFill (MovieClip.beginBitmapFill メソッド)

```
public beginBitmapFill(bmp:BitmapData, [matrix:Matrix], [repeat:Boolean],
  [smoothing:Boolean]) : Void
```

描画領域をビットマップイメージで塗りつぶします。ビットマップを繰り返すか、タイリング表示して、領域を塗りつぶすことができます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

bmp:BitmapData - 透明または不透明なビットマップイメージ。

matrix:Matrix (オプション) - flash.geom.Matrix クラスのマトリックスオブジェクト。これを使用してビットマップ上に変形を定義できます。たとえば、次のマトリックスを使用すると、ビットマップを 45 度 (π/4 ラジアン) 回転できます。

```
var matrix = new flash.geom.Matrix();
matrix.rotate(Math.PI/4);
```

repeat:Boolean (オプション) - true の場合、ビットマップイメージが一定のパターンでタイリング表示されます。false の場合、ビットマップイメージは繰り返されません。ビットマップからはみ出す塗り領域にはビットマップのエッジが使用されます。

たとえば、次のビットマップ (20 x 20 ピクセルのチェッカーボードのパターン) を考えます。

次の例のように repeat が true に設定されている場合、ビットマップの塗りでビットマップが繰り返されます。

repeat が false に設定されている場合、エッジのピクセルがビットマップの外側の塗り領域に使用されます。

smoothing:Boolean(オプション)- false の場合、最近傍アルゴリズムを使用して拡大されたビットマップイメージがレンダリングされ、ピクセル化されたように見えます。true の場合、双線形アルゴリズムを使用して拡大されたビットマップイメージがレンダリングされます。最近傍アルゴリズムを使用したレンダリングは、通常はかなり高速です。このパラメータのデフォルト値は false です。

例

次のコードでは、単純なビットマップを定義した後、beginBitmapFill() メソッドを使用して、このビットマップをタイリング表示してムービークリップを塗りつぶします。

```
import flash.display.*;
import flash.geom.*;

var bmpd:BitmapData = new BitmapData(20,20);
var rect1:Rectangle = new Rectangle(0,0,10,10);
var rect2:Rectangle = new Rectangle(0, 10, 10, 20);
var rect3:Rectangle = new Rectangle(10, 0, 20, 10);
var rect4:Rectangle = new Rectangle(10, 10, 20, 20);
bmpd.fillRect(rect1, 0xAA0000FF);
bmpd.fillRect(rect2, 0xAA00FF00);
bmpd.fillRect(rect3, 0xAAFF0000);
bmpd.fillRect(rect4, 0xAA999999);

this.createEmptyMovieClip("bmp_fill_mc", this.getNextHighestDepth());
with (bmp_fill_mc) {
  matrix = new Matrix();
  matrix.rotate(Math.PI/8);
  repeat = true;
  smoothing = true;
  beginBitmapFill(bmpd, matrix, repeat, smoothing);
  moveTo(0, 0);
  lineTo(0, 60);
  lineTo(60, 60);
  lineTo(60, 0);
  lineTo(0, 0);
  endFill();
}

bmp_fill_mc._xscale = 200;
bmp_fill_mc._yscale = 200;
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

beginFill (MovieClip.beginFill メソッド)

```
public beginFill(rgb:Number, [alpha:Number]) : Void
```

新しい描画パスの始点を示します。開いたパスがあり (現在の描画位置が MovieClip.moveTo() メソッドで指定した前の座標と等しくない場合)、そのパスに関連付けられた塗りがあるときは、パスが線で閉じられ、塗りが適用されます。これは、MovieClip.endFill() メソッドを呼び出した場合の結果と似ています。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

`rgb:Number` - 16 進カラー値。たとえば、赤は 0xFF0000、青は 0x0000FF で表します。この値を指定しないか、値が undefined である場合、塗りは作成されません。

`alpha:Number` (オプション) - 塗りのアルファ値を指定する 0 ～ 100 の整数。この値を指定しないと、100 (不透明) が使用されます。0 未満の値を指定しても 0 が適用されます。100 を超える値を指定しても 100 が適用されます。

例

次の例では、ステージ上に赤の塗りで四角形を作成します。

```
this.createEmptyMovieClip("square_mc", this.getNextHighestDepth());
square_mc.beginFill(0xFF0000);
square_mc.moveTo(10, 10);
square_mc.lineTo(100, 10);
square_mc.lineTo(100, 100);
square_mc.lineTo(10, 100);
square_mc.lineTo(10, 10);
square_mc.endFill();
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

別の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/DrawingAPI" フォルダに移動して drawingapi.fla ファイルにアクセスします。

関連項目

moveTo (MovieClip.moveTo メソッド), endFill (MovieClip.endFill メソッド), beginGradientFill (MovieClip.beginGradientFill メソッド)

beginGradientFill (MovieClip.beginGradientFill メソッド)

```
public beginGradientFill(fillType:String, colors:Array, alphas:Array,
  ratios:Array, matrix:Object, [spreadMethod:String],
  [interpolationMethod:String], [focalPointRatio:Number]) : Void
```

新しい描画パスの始点を示します。最初のパラメータが undefined, であるか、いずれのパラメータも渡されない場合は、パスに関連する塗りがありません。開いたパスがある場合 (現在の描画位置が MovieClip.moveTo() メソッドで指定された前の座標と等しくない場合) で、パスに関連する塗りがあるときは、そのパスが線で閉じられた後で塗りが適用されます。この動作は MovieClip.endFill() メソッドを呼び出したときと似ています。

このメソッドは、次のいずれかの条件が存在する場合は失敗します。

- colors、alphas、および ratios の各パラメータの項目数が等しくない。
- fillType パラメータが "linear" でも "radial" でもない。
- matrix パラメータのオブジェクトのフィールドのいずれかが無いか、無効である。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 6 - spreadMethod、interpolationMethod、および focalPointRatio の各パラメータが Flash Player 8 で追加されました。

パラメータ

fillType:String - "linear" または "radial" のいずれかのストリングを指定できます。

colors:Array - グラデーションで使用できる RGB 16 進カラー値の配列 (赤 0xFF0000、青 0x0000FF など)。最大 15 色まで指定できます。各色について、alphas パラメータと ratios パラメータで対応する値を必ず指定してください。

alphas:Array - colors 配列内の各色に対応するアルファ値の配列。有効な値は 0 ～ 100 です。0 未満の値の場合は 0 が、100 を超える値の場合は 100 が適用されます。

ratios:Array - 色分布の比率の配列。有効な値は 0 ～ 255 です。この値は、100% でサンプリングされる色の幅の割合をパーセントで定義します。colors パラメータの値ごとに、値を指定してください。

たとえば、青と緑の 2 色を含む線状グラデーションの場合、次の図は、ratios 配列のさまざまな値に基づいて、グラデーションで配置される色を示します。

ratios	グラデーション
[0, 127]	
[0, 255]	
[127, 255]	

配列内の値は、[0, 63, 127, 190, 255] のように順に増やしていく必要があります。

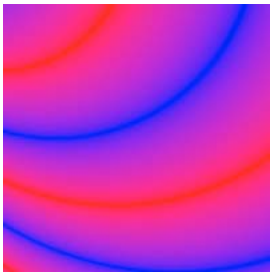
matrix:Object - 次の 3 つの形式のいずれかの変換マトリックス。

- flash.geom.Matrix クラスで定義されているマトリックスオブジェクト。このパラメータは、Flash Player 8 以降でのみサポートされています。flash.geom.Matrix クラスには、createGradientBox() メソッドがあります。このメソッドを使用すると、MovieClip クラスの beginGradientFill() メソッドで使用できるマトリックスを容易に設定できます。Flash Player 8 以降では、この形式のマトリックスを使用することをお勧めします。

  次の例では、このタイプの matrix パラメータを指定した beginGradientFill() メソッドを使用します。

```
import flash.geom.*

this.createEmptyMovieClip("gradient_mc", this.getNextHighestDepth());
with (gradient_mc)
{
  colors = [0xFF0000, 0x0000FF];
  fillType = "radial"
  alphas = [100, 100];
  ratios = [0, 0xFF];
  spreadMethod = "reflect";
  interpolationMethod = "linearRGB";
  focalPointRatio = 0.9;
  matrix = new Matrix();
  matrix.createGradientBox(100, 100, Math.PI, 0, 0);
  beginGradientFill(fillType, colors, alphas, ratios, matrix,
    spreadMethod, interpolationMethod, focalPointRatio);
  moveTo(100, 100);
  lineTo(100, 300);
  lineTo(300, 300);
  lineTo(300, 100);
  lineTo(100, 100);
  endFill();
}
```

  このコードでは、画面上に次のイメージが描画されます。



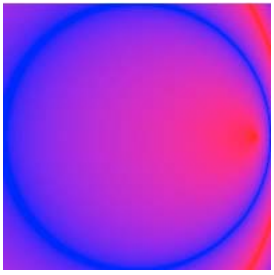
■ プロパティ a、b、c、d、e、f、g、h および i を使用できます。これらのプロパティを使用して、3 x 3 のマトリックスを次の形式で記述できます。

```
a b c
d e f
g h i
```

メモ : Flash Player 8 以降の場合、このリストの最初の箇条書き項目で示したように、flash.geom.Matrix オブジェクトの形式で matrix パラメータを定義することをお勧めします。次の例では、このタイプの matrix パラメータを指定した beginGradientFill() メソッドを使用します。

```
this.createEmptyMovieClip("gradient_mc", this.getNextHighestDepth());
with (gradient_mc)
{
  colors = [0xFF0000, 0x0000FF];
  fillType = "radial"
  alphas = [100, 100];
  ratios = [0, 0xFF];
  spreadMethod = "reflect";
  interpolationMethod = "linearRGB";
  focalPointRatio = 0.9;
  matrix = {a:200, b:0, c:0, d:0, e:200, f:0, g:200, h:200, i:1};
  beginGradientFill(fillType, colors, alphas, ratios, matrix, spreadMethod,
interpolationMethod, focalPointRatio);
  moveTo(100, 100);
  lineTo(100, 300);
  lineTo(300, 300);
  lineTo(300, 100);
  lineTo(100, 100);
  endFill();
}
```

このコードでは、画面上に次のイメージが描画されます。



■ matrixType、x, y、w、h、r の各プロパティを持つオブジェクト。

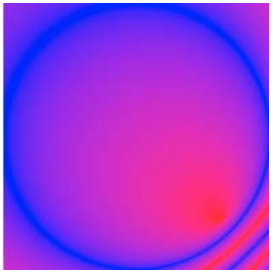
各プロパティの内容は次のとおりです。matrixType はストリング "box" です。x はグラデーションの左上隅の、親クリップの基準点からの相対的な水平座標、y はグラデーションの左上隅の、親クリップの基準点からの相対的な垂直座標を示します。また、w はグラデーションの幅、h はグラデーションの高さ、r はグラデーションのラジアン単位の回転です。

メモ : Flash Player 8 以降の場合、このリストの最初の箇条書き項目で示したように、flash.geom.Matrix オブジェクトの形式で matrix パラメータを定義することをお勧めします。

次の例では、このタイプの matrix パラメータを指定した beginGradientFill() メソッドを使用します。

```
this.createEmptyMovieClip("gradient_mc", this.getNextHighestDepth());
with (gradient_mc)
{
  colors = [0xFF0000, 0x0000FF];
  fillType = "radial"
  alphas = [100, 100];
  ratios = [0, 0xFF];
  spreadMethod = "reflect";
  interpolationMethod = "linearRGB";
  focalPointRatio = 0.9;
  matrix = {matrixType:"box", x:100, y:100, w:200, h:200, r:(45/
  180)*Math.PI};
  beginGradientFill(fillType, colors, alphas, ratios, matrix,
    spreadMethod, interpolationMethod, focalPointRatio);
  moveTo(100, 100);
  lineTo(100, 300);
  lineTo(300, 300);
  lineTo(300, 100);
  lineTo(100, 100);
  endFill();
}
```

このコードでは、画面上に次のイメージが描画されます。

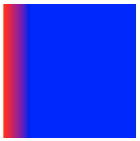


`spreadMethod:String` (オプション) - Flash Player 8 で追加されました。"pad"、"reflect"、"repeat" のいずれかで、グラデーションの塗りのモードを調整します。デフォルト値は "pad" です。

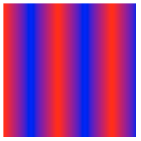
たとえば、2 つの色の間にシンプルな線状グラデーションがあるとします。

```
import flash.geom.*;
var fillType:String = "linear"
var colors:Array = [0xFF0000, 0x0000FF];
var alphas:Array = [100, 100];
var ratios:Array = [0x00, 0xFF];
var matrix:Matrix = new Matrix();
matrix.createGradientBox(20, 20, 0, 0, 0);
var spreadMethod:String = "pad";
this.beginGradientFill(fillType, colors, alphas, ratios, matrix, spreadMethod);
this.moveTo(0, 0);
this.lineTo(0, 100);
this.lineTo(100, 100);
this.lineTo(100, 0);
this.lineTo(0, 0);
this.endFill();
```

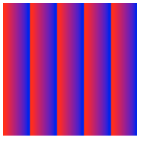
この例では spread メソッドに `"pad"` を使用しているため、グラデーションの塗りは次のようになります。



spread メソッドに `"reflect"` を使用する場合、グラデーションの塗りは次のようになります。

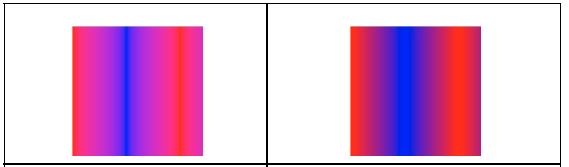


spread メソッドに `"repeat"` を使用する場合、グラデーションの塗りは次のようになります。



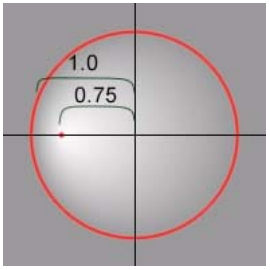
`interpolationMethod:String` (オプション) - Flash Player 8 で追加されました。"RGB" または "linearRGB" のいずれかです。"linearRGB" では、グラデーションカラーは線状に散布されます。デフォルト値は "RGB" です。

たとえば、2 つの色の間に、spreadMethod パラメータが "reflect" に設定されたシンプルな線状グラデーションがあるとします。それぞれの補間方法で、外観に次のような影響があります。



"linearRGB"	"RGB"

focalPointRatio:Number (オプション) - Flash Player 8 で追加されました。グラデーションの焦点の位置を調整する数値です。0 は焦点が中央にあること、1 は焦点がグラデーション円の一方の境界にあること、-1 は焦点がグラデーション円のもう一方の境界にあることを示します。-1 未満または 1 より大きい値は、-1 または 1 に丸められます。たとえば、次のイメージでは、focalPointRatio が 0.75 に設定されています。



例

次のコードでは、球状の陰影の効果を作成します。

```
import flash.geom.*
this.createEmptyMovieClip("gradient_mc", this.getNextHighestDepth());
with (gradient_mc)
  {
    fillType = "radial"
  colors = [0x000000, 0xFFFFFF];
    alphas = [50, 90];
    ratios = [0, 0xFF];
    spreadMethod = "pad";
    interpolationMethod = "RGB";
  focalPointRatio = 0.3;
  matrix = new Matrix();
  matrix.createGradientBox(100, 100, 0, 0, 0);
  beginGradientFill(fillType, colors, alphas, ratios, matrix,
    spreadMethod, interpolationMethod, focalPointRatio);
  moveTo(0, 0);
  lineTo(0, 100);
  lineTo(100, 100);
```

```
    lineTo(100, 0);
    lineTo(0, 0);
    endFill();
}
```

次のように描画されます。このイメージは 50% 縮小されています。

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

関連項目

createGradientBox (Matrix.createGradientBox メソッド), beginFill (MovieClip.beginFill メソッド), endFill (MovieClip.endFill メソッド), lineStyle (MovieClip.lineStyle メソッド), lineTo (MovieClip.lineTo メソッド), moveTo (MovieClip.moveTo メソッド)

blendMode (MovieClip.blendMode プロパティ)

public blendMode : Object

このムービークリップのブレンドモード。ブレンドモードは、画面上の別のオブジェクトの上のレイヤー内におけるムービークリップの外観に影響します。

Flash Player は blendMode プロパティをムービークリップの各ピクセルに適用します。各ピクセルは、3 つの要素カラー (赤、緑、青) で構成されており、各要素カラーは 0x00 ～ 0xFF の値を持ちます。Flash Player は、ムービークリップの 1 つのピクセルの各要素カラーを、背景のピクセルの対応するカラーと比較します。たとえば、blendMode が "lighten" に設定されている場合、Flash Player はムービークリップの赤の値と背景の赤の値とを比較し、明るい方を表示色の赤成分として使用します。

次の表で、blendMode の設定について説明します。blendMode プロパティを設定するには、1 ～ 14 の整数またはストリングを使用できます。表の中の図は、画面上の別のオブジェクト (1) に重なった円のムービークリップ (2) に適用される blendMode の値を示しています。

整数値	ストリング値	図	説明
1	"normal"		ムービークリップは、背景の前に表示されます。ムービークリップのピクセル値により、背景のピクセル値が無効になります。ムービークリップが透明な部分では、背景が表示されます。
2	"layer"		ムービークリップを事前構成するための一時バッファの作成を強制します。この処理は、ムービークリップに子のオブジェクトが複数存在し、子の blendMode が "normal" 以外に設定されている場合に自動的に行われます。
3	"multiply"		ムービークリップの要素カラーの値と背景色の値とを乗算した後、0xFF で除算して正規化して、色を暗くします。これは、シャドウや深度効果によく使用されます。たとえば、ムービークリップ内のあるピクセルの要素カラー (たとえば赤) と背景のピクセルの対応するカラーの値がともに 0x88 である場合、乗算した結果は 0x4840 です。0xFF で除算すると、その要素カラーの値が 0x48 となり、ムービークリップや背景よりも濃い色になります。
4	"screen"		ムービークリップの色の補数 (逆) と背景色の補数を乗算して、ブリーチ効果を得ます。この設定は、ハイライトや、ムービークリップの黒い領域の削除によく使用されます。
5	"lighten"		ムービークリップの要素カラーと背景の要素カラーのうち明るい方 (値が大きい方) を選択します。この設定は、重ね合わせタイプによく使用されます。 たとえば、ムービークリップのピクセルの 1 つの RGB 値が 0xFFCC33 で、背景のピクセルの RGB 値が 0xDDF800 の場合、表示されるピクセルの RGB 値は 0xFFF833 になります (0xFF > 0xDD、0xCC < 0xF8、および 0x33 > 0x00 のそれぞれの大きい方が採用されます)。
6	"darken"		ムービークリップの要素カラーと背景の要素カラーのうち暗い方 (値が小さい方) を選択します。この設定は、重ね合わせタイプによく使用されます。 たとえば、ムービークリップのピクセルの 1 つの RGB 値が 0xFFCC33 で、背景のピクセルの RGB 値が 0xDDF800 の場合、表示されるピクセルの RGB 値は 0xDDCC00 になります (0xFF > 0xDD、0xCC < 0xF8、および 0x33 > 0x00 のそれぞれの小さい方が採用されます)。

整数値	ストリング値	図	説明
7	"difference"		ムービークリップの要素カラーと背景の要素カラーを比較し、2つの要素カラーのうち明るい方の値から暗い方の値を差し引きます。この設定は、鮮やかな色にする場合によく使用されます。 たとえば、ムービークリップのピクセルの1つのRGB値が0xFFCC33で、背景のピクセルのRGB値が0xDDF800の場合、0xFF - 0xDD = 0x22、0xF8 - 0xCC = 0x2C、0x33 - 0x00 = 0x33のため、表示されるピクセルのRGB値は0x222C33になります。
8	"add"		ムービークリップの要素カラーの値を背景の要素カラーに加算し、上限0xFFを適用します。この設定は、2つのオブジェクト間で色を明るくするディゾルブをアニメーションにするときによく使用されます。 たとえば、ムービークリップのピクセルの1つのRGB値が0xAAA633で、背景のピクセルのRGB値が0xDD2200の場合、0xAA + 0xDD > 0xFF、0xA6 + 0x22 = 0xC8、0x33 + 0x00 = 0x33のため、表示されるピクセルのRGB値は0xFFC833になります。
9	"subtract"		結果の下限を0として、ムービークリップの要素カラーの値をその背景の要素カラーの値から減算します。この設定は、2つのオブジェクト間で色を暗くするディゾルブをアニメーションにするときによく使用されます。 たとえば、ムービークリップのピクセルの1つのRGB値が0xAA2233で、背景のピクセルのRGB値が0xDDA600の場合、0xDD - 0xAA = 0x33、0xA6 - 0x22 = 0x84、0x00 - 0x33 < 0x00のため、表示されるピクセルのRGB値は0x338400になります。
10	"invert"		背景を反転します。
11	"alpha"		ムービークリップの各ピクセルのアルファ値を背景に適用します。そのためには、blendModeの設定"layer"を親ムービークリップに適用する必要があります。たとえば、図の親ムービークリップ(白い背景)は、blendMode = "layer"に設定されています。

整数値	ストリング値	図	説明
12	"erase"		ムービークリップのアルファ値に基づいて背景を消去します。そのためには、blendMode の設定 "layer" を親ムービークリップに適用する必要があります。たとえば、図の親ムービークリップ (白い背景) は、blendMode = "layer" に設定されています。
13	"overlay"		背景の暗さに基づいて、各ビットマップの色を調整します。背景が 50% グレーよりも明るい場合、ムービークリップと背景の色が網がけされ、明るくなります。背景が 50% グレーよりも暗い場合、2 つの色が乗算されて、より暗くなります。この設定は、シャドウ効果によく使用されます。
14	"hardlight"		ムービーの暗さに応じて、各ビットマップの色を調整します。ムービークリップが 50% のグレーよりも暗い場合、ムービークリップと背景の色が網がけされ、明るくなります。ムービークリップが 50% グレーよりも暗い場合、2 つの色が乗算されて、より暗くなります。この設定は、シャドウ効果によく使用されます。

blendMode プロパティを他の値に設定しようとすると、プロパティは "normal" に設定されます。

ただし、このプロパティを整数に設定すると、その値は対応するストリングバージョンに直ちに変換されます。

```
this.createEmptyMovieClip("mclip", this.getNextHighestDepth());
mclip.blendMode = 8;
trace(mclip.blendMode) // add
```

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、グラデーションの塗りで 2 つのムービークリップを設定し、前景にある片方のムービークリップのブレンドモードを 1 秒ごとに変更します。効果に "alpha" ブレンドモードが現れるように、ムービークリップ mc2 のグラデーションにアルファ比の範囲が含まれ、親のムービークリップに "layer" ブレンドモードが適用されます (this.blendMode="layer")。

```
this.createEmptyMovieClip("mc1", this.getNextHighestDepth());
this.createEmptyMovieClip("mc2", this.getNextHighestDepth());
this.blendMode="layer";
this.createTextField("blendLabel", this.getNextHighestDepth(), 50, 150, 100,
100)

fillClip(mc1, 0x00AA00, 0x22FFFF, 100, 100)
fillClip(mc2, 0xFF0000, 0x2211FF, 100, 50)
mc2._x = 33;
mc2._y = 33;
```

```
  var blendModeIndex = 0;

  setInterval(changeBlendMode, 1000);
  function changeBlendMode()
  {
    mc2.blendMode = blendModeIndex % 14 + 1 ;
      // values 1 - 14
  blendLabel.text = (blendModeIndex% 14 + 1) + ": " + mc2.blendMode;
    blendModeIndex++;
  }

  function fillClip(mc:MovieClip, color1:Number, color2:Number,
          alpha1:Number, alpha2: Number)
{
    matrix = {a:100, b:0, c:0, d:0, e:100, f:0, g:50, h:20, i:1};
    mc.beginGradientFill("linear", [color1, color2], [alpha1, alpha2], [0,
  0xFF], matrix);
    mc.lineStyle(8,0x888888,100)
    mc.moveTo(0, 0);
    mc.lineTo(0, 100);
    mc.lineTo(100, 100);
    mc.lineTo(100, 0);
    mc.lineTo(0, 0);
    mc.endFill();
  }
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドの代わりに、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

cacheAsBitmap (MovieClip.cacheAsBitmap プロパティ)

public cacheAsBitmap : Boolean

true に設定されている場合、ムービークリップの内部ビットマップ表現がキャッシュされます。これで、複雑なベクターコンテンツを格納しているムービークリップのパフォーマンスを向上させることができます。

ビットマップがキャッシュされているムービークリップのベクターデータはすべて、メインステージでなくビットマップに描画されます。その後、ビットマップは、最も近いピクセル境界に吸着された非伸縮、非回転のピクセルとして、メインステージにコピーされます。ピクセルは、親オブジェクトと 1 対 1 でマップされます。ビットマップの境界が変更されると、ビットマップは伸縮されずに再作成されます。

内部ビットマップが作成されるのは、cacheAsBitmap プロパティが true に設定されている場合だけです。

ムービークリップの cacheAsBitmap プロパティを true に設定しても、レンダリングは変化しませんが、ムービークリップでピクセルへの吸着が自動的に実行されます。ベクターコンテンツの複雑さによっては、アニメーションの速度が大幅に向上する可能性があります。

ムービークリップにフィルタを適用するたびに、そのムービークリップの filter 配列が空でなければ、cacheAsBitmap プロパティが自動的に true に設定されます。ムービークリップにフィルタが適用されている場合、そのムービークリップの cacheAsBitmap は、false に設定されている場合でも、true と報告されます。ムービークリップのすべてのフィルタをクリアすると、cacheAsBitmap は直前の設定に戻ります。

次の場合は、cacheAsBitmap プロパティが true に設定されていても、ムービークリップはビットマップを使用せずにベクターデータからレンダリングされます。

- ビットマップが大きすぎる (幅または高さが 2,880 ピクセルを超える) 場合。
- メモリ不足によりビットマップにメモリを割り当てることができない場合。

cacheAsBitmap プロパティは、その内容がほぼ静的で、拡大や縮小、回転が頻繁に行われないムービークリップに最適です。そのようなムービークリップでは、cacheAsBitmap プロパティによって、ムービークリップの変換時 (その *x* 位置と *y* 位置の変更時) にパフォーマンスが向上します。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、ムービークリップインスタンスにドロップシャドウを適用します。その後、フィルタの適用時に true に設定される cacheAsBitmap プロパティの値をトレースします。

```
import flash.filters.DropShadowFilter;

var container:MovieClip = setUpShape();
trace(container.cacheAsBitmap); // false
var dropShadow:DropShadowFilter = new DropShadowFilter(6, 45, 0x000000, 50, 5,
  5, 1, 2, false, false, false);
container.filters = new Array(dropShadow);
trace(container.cacheAsBitmap); // true

function setUpShape():MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip("container",
  this.getNextHighestDepth());
  mc._x = 10;
  mc._y = 10;
  var w:Number = 50;
  var h:Number = 50;
  mc.beginFill(0xFFCC00);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
```

```
  return mc;
}
```

関連項目

opaqueBackground (MovieClip.opaqueBackground プロパティ),cacheAsBitmap (MovieClip.cacheAsBitmap プロパティ)

clear (MovieClip.clear メソッド)

```
public clear() : Void
```

MovieClip.lineStyle() で指定された線スタイルを含め、ムービークリップの描画メソッドを使用して実行時に作成されたすべてのグラフィックを削除します。Flash の描画ツールを使用してオーサリング時に手動で作成したシェイプや線は削除されません。

対応バージョン: ActionScript 1.0、Flash Player 6

例

次の例では、ステージ上にボックスを描画します。ユーザーがボックスのグラフィックをクリックすると、ステージ上からグラフィックが削除されます。

```
this.createEmptyMovieClip("box_mc", this.getNextHighestDepth());
box_mc.onRelease = function() {
  this.clear();
};
drawBox(box_mc, 10, 10, 320, 240);
function drawBox(mc:MovieClip, x:Number, y:Number, w:Number, h:Number):Void {
  mc.lineStyle(0);
  mc.beginFill(0xEEEEEE);
  mc.moveTo(x, y);
  mc.lineTo(x+w, y);
  mc.lineTo(x+w, y+h);
  mc.lineTo(x, y+h);
  mc.lineTo(x, y);
  mc.endFill();
}
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

別の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/DrawingAPI" フォルダに移動して drawingapi.fla ファイルにアクセスします。

関連項目

lineStyle (MovieClip.lineStyle メソッド)

createEmptyMovieClip (MovieClip.createEmptyMovieClip メソッド)

public createEmptyMovieClip(name:String, depth:Number) : MovieClip

既存のムービークリップの子として空のムービークリップを作成します。このメソッドの動作は attachMovie() メソッドに似ていますが、新しいムービークリップのリンケージ識別子を指定する必要がありません。新しく作成された空のムービークリップの基準点は左上隅です。このメソッドは、いずれかのパラメータを省略すると失敗します。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン: ActionScript 1.0、Flash Player 6

パラメータ

name:String - 新しいムービークリップのインスタンス名を指定するストリング。

depth:Number - 新しいムービークリップの深度を指定する整数。

戻り値

MovieClip - 新しく生成されたムービークリップへの参照。

例

次の例では、container という名前で空の MovieClip を作成します。その中に新しい TextField を作成し、新しい TextField.text プロパティを設定します。

```
var container:MovieClip = this.createEmptyMovieClip("container",
  this.getNextHighestDepth());
var label:TextField = container.createTextField("label", 1, 0, 0, 150, 20);
label.text = "Hello World";
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

attachMovie (MovieClip.attachMovie メソッド)

createTextField (MovieClip.createTextField メソッド)

```
public createTextField(instanceName:String, depth:Number, x:Number, y:Number,
  width:Number, height:Number) : TextField
```

このメソッドを呼び出したムービークリップの子となる空のテキストフィールドを新しく作成します。createTextField( ) メソッドを使用すると、ムービーの再生中にテキストフィールドを作成できます。depth パラメータは、ムービークリップ内の新しいテキストフィールドの深度 (z 順序の位置) を決定します。それぞれの深度にオブジェクトを 1 つだけ含むことができます。既にテキストフィールドが存在する深度に新しいテキストフィールドを作成すると、既存のテキストフィールドが新しいテキストフィールドで置き換えられます。既存のテキストフィールドが上書きされないようにするには、MovieClip.getInstanceAtDepth( ) を使用して特定の深度が既に占有されているかどうかを確認するか、MovieClip.getNextHighestDepth( ) を使用して占有されていない最も上の深度を調べます。テキストフィールドの位置は (x、y)、幅は width、高さは *height* になります。x パラメータと y パラメータは、コンテナのムービークリップを基準とします。これらのパラメータは、テキストフィールドの _x プロパティと _y プロパティに対応します。width パラメータと height パラメータは、テキストフィールドの _width プロパティと _height プロパティにそれぞれ対応しいます。

テキストフィールドのデフォルトのプロパティは次のとおりです。

```
type = "dynamic"
border = false
background = false
password = false
multiline = false
html = false
embedFonts = false
selectable = true
wordWrap = false
mouseWheelEnabled = true
condenseWhite = false
restrict = null
variable = null
maxChars = null
styleSheet = undefined
tabInded = undefined
```

createTextField( ) で作成したテキストフィールドは、次のデフォルトの TextFormat オブジェクトの設定を受け取ります。

```
font = "Times New Roman" // "Times" on Mac OS
size = 12
color = 0x000000
bold = false
italic = false
underline = false
url = ""
target = ""
```

```
align = "left"
leftMargin = 0
rightMargin = 0
indent = 0
leading = 0
blockIndent = 0
bullet = false
display = block
tabStops = [] // (empty array)
```

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン： ActionScript 1.0、Flash Player 6 - Flash Player 8 では、void ではなく、作成した TextField オブジェクトの参照を返します。

パラメータ

`instanceName:String` - 新しいテキストフィールドのインスタンス名を識別するストリング。

`depth:Number` - 新しいテキストフィールドの深度を指定する正の整数。

`x:Number` - 新しいテキストフィールドの *x* 座標を指定する整数。

`y:Number` - 新しいテキストフィールドの *y* 座標を指定する整数。

`width:Number` - 新しいテキストフィールドの幅を指定する正の整数。

`height:Number` - 新しいテキストフィールドの高さを指定する正の整数。

戻り値

`TextField` - Flash Player 8 では、TextField オブジェクトへの参照を返します。Flash Player 8 より前のバージョンの Player では、`void` を返します。

例

次の例では、幅 300、高さ 100、*x* 座標 100、*y* 座標 100、境界なし、赤、および下線付きのテキストフィールドを作成します。

```
this.createTextField("my_txt", 1, 100, 100, 300, 100);
my_txt.multiline = true;
my_txt.wordWrap = true;
var my_fmt:TextFormat = new TextFormat();
my_fmt.color = 0xFF0000;
my_fmt.underline = true;
my_txt.text = "This is my first test field object text.";
my_txt.setTextFormat(my_fmt);
```

別の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Animation" フォルダに移動して animation.fla ファイルにアクセスします。

関連項目

getInstanceAtDepth (MovieClip.getInstanceAtDepth メソッド), getNextHighestDepth (MovieClip.getNextHighestDepth メソッド), TextFormat

_currentframe (MovieClip._currentframe プロパティ)

```
public _currentframe : Number ( 読み取り専用 )
```

ムービークリップのタイムライン内で再生ヘッドがあるフレーム番号を返します。

対応バージョン : ActionScript 1.0、Flash Player 4

例

次の例では、_currentframe プロパティを使用して、ムービークリップ actionClip_mc の再生ヘッドを現在の位置から 5 つ先のフレームに進めます。

```
actionClip_mc.gotoAndStop(actionClip_mc._currentframe + 5);
```

curveTo (MovieClip.curveTo メソッド)

```
public curveTo(controlX:Number, controlY:Number, anchorX:Number,
  anchorY:Number) : Void
```

(controlX, controlY) で指定されたコントロールポイントを使用し、現在の描画位置から (anchorX, anchorY) まで、現在の線のスタイルで曲線を描画します。その後、現在の描画位置は (anchorX, anchorY) に設定されます。描画先のムービークリップに Flash の描画ツールで作成したコンテンツが含まれている場合、curveTo() メソッドへの呼び出しの結果がそのコンテンツの下に描画されます。moveTo() メソッドを呼び出す前に curveTo() を呼び出すと、現在の描画位置がデフォルトの (0,0) に設定されます。いずれかのパラメータを省略すると、このメソッドは失敗し、現在の描画位置は変更されません。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

controlX:Number - コントロールポイントの、親ムービークリップの基準点からの相対的な水平座標を指定する整数。

controlY:Number - コントロールポイントの、親ムービークリップの基準点からの相対的な垂直座標を指定する整数。

anchorX:Number - 次のアンカーポイントの、親ムービークリップの基準点からの相対的な水平座標を指定する整数。

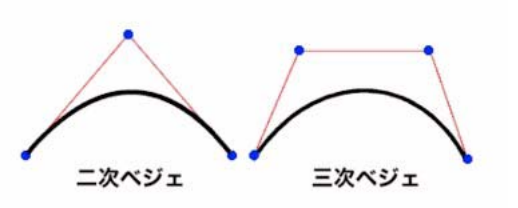
anchorY:Number - 次のアンカーポイントの、親ムービークリップの基準点からの相対的な垂直座標を指定する整数。

例

次の例では、極細の青の実線と赤の塗りで環状に近い曲線を描画します。

```
this.createEmptyMovieClip("circle_mc", 1);
with (circle_mc) {
  lineStyle(0, 0x0000FF, 100);
  beginFill(0xFF0000);
  moveTo(0, 100);
  curveTo(0,200,100,200);
  curveTo(200,200,200,100);
  curveTo(200,0,100,0);
  curveTo(0,0,0,100);
  endFill();
}
```

この例で描画される曲線は、二次ベジェ曲線です。二次ベジェ曲線は、2 つのアンカーポイントと 1 つのコントロールポイントで構成されています。曲線は、2 つのアンカーポイントを補間し、コントロールポイントに向かいます。



次のスクリプトでは、curveTo() メソッドと Math クラスを使用して円を作成します。

```
this.createEmptyMovieClip("circle2_mc", 2);
circle2_mc.lineStyle(0, 0x000000);
drawCircle(circle2_mc, 100, 100, 100);
function drawCircle(mc:MovieClip, x:Number, y:Number, r:Number):Void {
  mc.moveTo(x+r, y);
  mc.curveTo(r+x, Math.tan(Math.PI/8)*r+y, Math.sin(Math.PI/4)*r+x,
Math.sin(Math.PI/4)*r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, r+y, x, r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, r+y, -Math.sin(Math.PI/4)*r+x,
Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-r+x, Math.tan(Math.PI/8)*r+y, -r+x, y);
  mc.curveTo(-r+x, -Math.tan(Math.PI/8)*r+y, -Math.sin(Math.PI/4)*r+x,
-Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, -r+y, x, -r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, -r+y, Math.sin(Math.PI/4)*r+x,
-Math.sin(Math.PI/4)*r+y);
  mc.curveTo(r+x, -Math.tan(Math.PI/8)*r+y, r+x, y);
}
```

別の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/DrawingAPI" フォルダに移動して drawingapi.fla ファイルにアクセスします。

関連項目

beginFill (MovieClip.beginFill メソッド), createEmptyMovieClip (MovieClip.createEmptyMovieClip メソッド), endFill (MovieClip.endFill メソッド), lineStyle (MovieClip.lineStyle メソッド), lineTo (MovieClip.lineTo メソッド), moveTo (MovieClip.moveTo メソッド), Math

_droptarget (MovieClip._droptarget プロパティ)

public _droptarget : String (読み取り専用)

このムービークリップがドロップされたムービークリップインスタンスの絶対パスを、スラッシュシンタックス表記で返します。_droptarget プロパティはスラッシュ (/) で始まるパスを常に返します。インスタンスの _droptarget プロパティと参照とを比較するには、eval() 関数を使用して戻り値をスラッシュシンタックスからドットシンタックス参照に変換します。

メモ : スラッシュシンタックスをサポートしない ActionScript 2.0 を使用する場合は、ドットシンタックス参照への変換が必要です。

対応バージョン : ActionScript 1.0、Flash Player 4

例

次の例では、ムービークリップインスタンス garbage_mc の _droptarget プロパティを評価し、eval() を使用して、評価結果をスラッシュシンタックスからドットシンタックス参照に変換します。次に、garbage_mc 参照とムービークリップインスタンス trashcan_mc への参照とを比較します。この 2 つの参照が等しい場合、garbage_mc の可視性を false に設定します。これらの参照が等しくない場合は、garbage インスタンスが元の位置にリセットされます。

```
origX = garbage_mc._x;
origY = garbage_mc._y;
garbage_mc.onPress = function() {
  this.startDrag();
};
garbage_mc.onRelease = function() {
  this.stopDrag();
  if (eval(this._droptarget) == trashcan_mc) {
  this._visible = false;
  } else {
  this._x = origX;
  this._y = origY;
  }
};
```

関連項目

startDrag (MovieClip.startDrag メソッド),stopDrag (MovieClip.stopDrag メソッド),eval 関数

duplicateMovieClip (MovieClip.duplicateMovieClip メソッド)

```
public duplicateMovieClip(name:String, depth:Number, [initObject:Object]) :
  MovieClip
```

SWF ファイルの再生中に、指定されたムービークリップのインスタンスを作成します。duplicateMovieClip() メソッドを呼び出したときに元のムービークリップがどのフレーム上にあっても、複製されたムービークリップは常にフレーム 1 から始まります。複製元のムービークリップ内の変数は、複製されたムービークリップにコピーされません。duplicateMovieClip() メソッドを使用して作成されたムービークリップは、その親で duplicateMovieClip() メソッドを呼び出した場合は複製されません。親のムービークリップが削除されると、複製されたムービークリップも削除されます。MovieClip.loadMovie() または MovieClipLoader クラスを使用してムービークリップをロードした場合、SWF ファイルの内容は複製されません。つまり、JPEG ファイル、GIF ファイル、PNG ファイル、または SWF ファイルをロードした後でムービークリップを複製して帯域幅を節約することはできません。

このメソッドを duplicateMovieClip() のグローバル関数バージョンと比較してみます。このメソッドのグローバル関数バージョンでは、複製するターゲットムービークリップを指定するパラメータが必要になります。MovieClip クラスのバージョンでは、このようなパラメータは不要です。メソッドのターゲットが、メソッドが呼び出されるムービークリップインスタンスであるためです。さらに、duplicateMovieClip() のグローバルバージョンでは、initobject パラメータも、新しく作成された MovieClip インスタンスへの参照の戻り値もサポートされていません。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

name:String - 複製したムービークリップの一意の識別子。

depth:Number - 新しいムービークリップを配置する深度を指定する一意の整数。新しいムービークリップインスタンスをオーサリング環境で作成されたすべてのコンテンツの下に配置するには、深度 -16384 を使用します。-16383 以上 -1 以下の値は、オーサリング環境での使用のために予約されているので、このメソッドでは使用しないでください。深度の有効値の範囲は 0 以上 1048575 以下です。

initObject:Object (オプション) - Flash Player 6 以降でのみサポートされます。複製ムービークリップに設定するプロパティを持つオブジェクトを指定します。このパラメータを使用すると、動的に作成したムービークリップは、クリップのパラメータを受け取ることができます。オブジェクトではない initObject は無視されます。initObject のすべてのプロパティが新しいインスタンスにコピーされます。initObject で指定されたプロパティは、コンストラクタ関数で使用できます。

戻り値

MovieClip - 複製されたムービークリップへの参照。Flash Player 6 以降でのみサポートされます。

例

次の例では、新しく作成されたムービークリップを何回か複製し、複製のたびにターゲットをトレースします。

```
var container:MovieClip = setUpContainer();
var ln:Number = 10;
var spacer:Number = 1;
var duplicate:MovieClip;
for(var i:Number = 1; i < ln; i++) {
  var newY:Number = i * (container._height + spacer);
  duplicate = container.duplicateMovieClip("clip-" + i, i, {_y:newY});
  trace(duplicate); // _level0.clip-[number]
}

function setUpContainer():MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip("container",
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 20;
  mc.beginFill(0x333333);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  return mc;
}
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

loadMovie (MovieClip.loadMovie メソッド), removeMovieClip (MovieClip.removeMovieClip メソッド), duplicateMovieClip 関数

enabled (MovieClip.enabled プロパティ)

```
public enabled : Boolean
```

ムービークリップの有効 / 無効を示すブール値。enabled のデフォルト値は true です。enabled を false に設定すると、ムービークリップのコールバックメソッドと on *action* イベントハンドラは呼び出されなくなり、Over、Down、Up の各フレームは無効になります。enabled プロパティは、ムービークリップのタイムラインには影響しません。ムービークリップが再生中である場合は、再生が継続されます。ムービークリップは、ムービークリップイベント (mouseDown、mouseUp、keyDown、keyUp など) を引き続き受け取ります。

enabled プロパティは、ムービークリップのボタンに似たプロパティのみを制御します。enabled プロパティはいつでも変更できます。このプロパティの変更後、ムービークリップはすぐに有効 / 無効になります。enabled プロパティは、プロトタイプオブジェクトから読み取ることができます。enabled プロパティが false に設定されている場合、オブジェクトは自動タブ順に含まれません。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ユーザーがムービークリップ circle_mc をクリックすると、このムービークリップが無効になります。

```
circle_mc.onRelease = function() {
  trace("disabling the "+this._name+" movie clip.");
  this.enabled = false;
};
```

endFill (MovieClip.endFill メソッド)

```
public endFill() : Void
```

beginFill() メソッドまたは beginGradientFill() メソッドの最後の呼び出し以降に追加された線と曲線に塗りを適用します。適用される塗りは、beginFill() または beginGradientFill() の前回の呼び出しで指定されたものです。現在の描画位置が moveTo() メソッドの直前の呼び出しで指定された座標と等しくない場合、塗りが定義されていれば、パスが線で閉じられた後、塗りが適用されます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ステージ上に赤の塗りで四角形を作成します。

```
this.createEmptyMovieClip("square_mc", this.getNextHighestDepth());
square_mc.beginFill(0xFF0000);
square_mc.moveTo(10, 10);
square_mc.lineTo(100, 10);
square_mc.lineTo(100, 100);
square_mc.lineTo(10, 100);
square_mc.lineTo(10, 10);
square_mc.endFill();
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

別の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/DrawingAPI" フォルダに移動して drawingapi.fla ファイルにアクセスします。

関連項目

beginFill (MovieClip.beginFill メソッド), beginGradientFill (MovieClip.beginGradientFill メソッド), moveTo (MovieClip.moveTo メソッド)

filters (MovieClip.filters プロパティ)

```
public filters : Array
```

ムービークリップに現在関連付けられている各フィルタオブジェクトが格納されている、インデックスの配列。flash.filters パッケージには、使用できる特定のフィルタを定義する複数のクラスが含まれています。

フィルタは、ActionScript コードを使用して、設計時または実行時に Flash オーサリングツールで適用できます。ActionScript を使用してフィルタを適用するには、MovieClip.filters 配列全体の一時コピーを作成してその一時配列を変更した後、一時配列の値を MovieClip.filters 配列に代入し直す必要があります。新しいフィルタオブジェクトを MovieClip.filters 配列に直接追加することはできません。

ActionScript を使用してフィルタを追加するには、次の手順を実行する必要があります。ターゲットムービークリップの名前は myMC とします。

- 選択したフィルタクラスのコンストラクタ関数を使用して、新しいフィルタオブジェクトを作成します。
- myMC.filters 配列の値を、myFilters などの名前の一時配列に割り当てます。
- 新しいフィルタオブジェクトを一時配列 myFilters に追加します。
- 一時配列の値を myMC.filters 配列に代入します。

filters 配列が空の場合、一時配列を使用する必要はありません。代わりに、作成したフィルタオブジェクトを格納する配列リテラルを直接代入することができます。

設計時または実行時に作成されたかどうかに関わらず、既存のフィルタオブジェクトを変更するには、以下の方法で filters 配列のコピーを変更する必要があります。

- myMC.filters 配列の値を、myFilters などの名前の一時配列に割り当てます。
- 一時配列 myFilters を使用してプロパティを変更します。たとえば、配列内の最初のフィルタの quality プロパティを設定する場合は、コードとして myList[0].quality = 1; を使用できます。
- 一時配列の値を myMC.filters 配列に代入します。

ムービークリップのフィルタをクリアするには、filters に空の配列 ([]) を代入します。

ロード時に、ムービークリップにフィルタが関連付けられている場合、透明なビットマップとしてムービークリップ自身をキャッシュするようにマークされます。これ以降、ムービークリップに有効なフィルタリストがある限り、ムービークリップはビットマップとしてキャッシュされます。このソースビットマップは、フィルタ効果のソースイメージとして使用されます。各ムービークリップは通常、フィルタが適用されていない元のムービークリップのビットマップと、フィルタ適用後の最終イメージのビットマップの 2 つのビットマップを持ちます。最終イメージはレンダリング時に使用されます。ムービークリップが変更されていない限り、最終イメージを更新する必要はありません。

複数のフィルタを格納する filters 配列を使用していて、各配列インデックスに割り当てられているフィルタの種類を追跡する必要がある場合、独自の filters 配列を維持し、別のデータ構造を使用して、各配列インデックスに関連付けられているフィルタの種類を追跡できます。filters 配列の各インデックスに関連付けられているフィルタの種類を簡単に判別する方法はありません。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、ドロップシャドウフィルタを myMC という名前のムービークリップに追加します。

```
var myDropFilter = new flash.filters.DropShadowFilter();
var myFilters:Array = myMC.filters;
myFilters.push(myDropFilter);
myMC.filters = myFilters;
```

次の例では、配列内の最初のフィルタの quality 設定を 15 に変更します。この例は、ムービークリップ myMC に少なくとも 1 つのフィルタオブジェクトが関連付けられている場合にのみ機能します。

```
var myList:Array = myMC.filters;
myList[0].quality = 15;
myMC.filters = myList;
```

メモ： 新しいフィルタオブジェクトを MovieClip.filters 配列に直接追加することはできないため、次のコードは、myMC という名前のターゲットムービークリップには影響しません。

```
myMC.filters.push(myDropShadow);
```

focusEnabled (MovieClip.focusEnabled プロパティ)

```
public focusEnabled : Boolean
```

プログラムで Selection.setFocus() を使用してムービークリップにフォーカスを割り当てることができるかどうかを指定します。focusEnabled プロパティが false に設定されている場合、または undefined (デフォルト) の場合は、イベントハンドラの onDragOut、onDragOver、onPress、onRelease、onReleaseOutside、onRollOut、onRollOver のいずれかがムービークリップに定義されていなければ、Selection.setFocus() を使用してムービークリップにフォーカスを割り当てることはできません。

focusEnabled プロパティが true に設定されている場合は、これらのイベントハンドラがどれも定義されていなくても、Selection.setFocus() を使用してムービークリップにフォーカスを割り当てることができます。

focusEnabled プロパティと tabEnabled プロパティを混同しないようにしてください。tabEnabled プロパティは、Tab キーを使用してムービークリップにフォーカスを割り当てることができるかどうかを制御するものであり、Selection.setFocus() メソッドには影響がありません。一方 focusEnabled プロパティは、キーボードナビゲーションには影響がありません。

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の例では、ムービークリップ my_mc の focusEnabled プロパティを false に設定します。

```
my_mc.focusEnabled = false;
```

関連項目

setFocus (Selection.setFocus メソッド)

_focusrect (MovieClip._focusrect プロパティ)

```
public _focusrect : Boolean
```

ムービークリップにフォーカスがあるときに、その周囲に黄色の矩形を表示するかどうかを指定するブール値。このプロパティは、グローバル _focusrect プロパティの設定を上書きできます。ムービークリップインスタンスの _focusrect プロパティのデフォルト値は null です。つまり、ムービークリップインスタンスは、グローバル _focusrect プロパティを上書きしません。ムービークリップインスタンスの _focusrect プロパティを true または false に設定した場合、その 1 つのムービークリップインスタンスのグローバル _focusrect プロパティの設定が上書きされます。

Flash Player 4 または Flash Player 5 の SWF ファイルでは、_focusrect プロパティはグローバル _focusrect プロパティを制御します。これはブール値です。Flash Player 6 以降では、この動作が変更され、ムービークリップごとに _focusrect をカスタマイズできます。

_focusrect プロパティが false の場合、そのムービークリップのキーボードナビゲーションは、Tab キーに限定されます。Enter キーや矢印キーを含め、他のキーはすべて無視されます。すべてのキーナビゲーションを元に戻すには、_focusrect を true に設定する必要があります。

対応バージョン : ActionScript 1.0、Flash Player 6

例

この例では、ブラウザウィンドウで、SWF ファイル内の指定されたムービークリップインスタンスにフォーカスがある場合に、その周囲に表示される黄色い矩形を非表示にする方法を示します。mc1_mc、mc2_mc、mc3_mc の 3 つのムービークリップを作成し、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
mc1_mc._focusrect = true;
mc2_mc._focusrect = false;
mc3_mc._focusrect = true;

mc1_mc.onRelease = traceOnRelease;
mc3_mc.onRelease = traceOnRelease;

function traceOnRelease() {
  trace(this._name);
}
```

[ファイル]-[パブリッシュプレビュー]-[HTML] を選択して、ブラウザウィンドウで SWF ファイルをテストします。ブラウザウィンドウで SWF をクリックしてフォーカスを与えたら、Tab キーを押して各インスタンスにフォーカスを与えます。_focusrect が無効になっている場合は、Enter キーまたはスペースバーを押してブラウザのムービークリップのコードを実行することはできません。

また、SWF ファイルのテストはテスト環境で行うことができます。テスト環境でメインメニューから [制御]-[キーボードショートカットを無効] を選択します。これにより、SWF ファイルのインスタンスの周囲にあるフォーカスの矩形を表示できます。

関連項目

_focusrect プロパティ , _focusrect (Button._focusrect プロパティ)

forceSmoothing (MovieClip.forceSmoothing プロパティ)

public forceSmoothing : Boolean

loadMovie() メソッドを使用して追加されたイメージのうち、ムービークリップと同じ階層レベルにあるイメージが拡大・縮小されるときに、そのイメージをスムージングするかどうかを決定するブール値。forceSmoothing を true に設定すると、レンダリングのパフォーマンスが低下します。デフォルト値は false です。

対応バージョン : ActionScript 1.0、Flash Player 9

_framesloaded (MovieClip._framesloaded プロパティ)

public _framesloaded : Number (読み取り専用)

ストリーミング SWF ファイルからロードされたフレーム数。このプロパティは、特定のフレームとその前のすべてのフレームの内容がロードされており、ユーザーのブラウザでローカルに使用できるかどうかを判別する場合に使用できます。また、大きい SWF ファイルのダウンロードを監視する場合にも便利です。たとえば、SWF ファイルの指定されたフレームがロードを完了するまで、その SWF ファイルがロード中であることを示すメッセージをユーザーに表示する場合に使用できます。

対応バージョン : ActionScript 1.0、Flash Player 4

例

次の例では、_framesloaded プロパティを使用して、すべてのフレームがロードされた時点で SWF ファイルを開始します。すべてのフレームのロードが完了するまでは、ムービークリップインスタンス bar_mc の _xscale プロパティは比例的に増加し、ロードの進行状況を示すプログレスバーを作成します。

タイムラインのフレーム 1 に次の ActionScript を入力します。

```
var pctLoaded:Number = Math.round(this.getBytesLoaded()/
  this.getBytesTotal()*100);
bar_mc._xscale = pctLoaded;
```

フレーム 2 に次のコードを追加します。

```
if (this._framesloaded < this._totalframes) {
  this.gotoAndPlay(1);
} else {
  this.gotoAndStop(3);
}
```

フレーム 3 以後にコンテンツを配置し、フレーム 3 に次のコードを追加します。

```
stop();
```

関連項目

MovieClipLoader

getBounds (MovieClip.getBounds メソッド)

```
public getBounds(bounds:Object) : Object
```

bounds パラメータに基づいて、ムービークリップの最小および最大の *x* 座標と *y* 座標を示すプロパティを返します。

メモ : ムービークリップのローカル座標をステージ座標に変換するには、MovieClip.localToGlobal() メソッドを使用します。ステージ座標をローカル座標に変換するには、MovieClip.globalToLocal() メソッドを使用します。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

bounds:Object - その座標系を基準点として使用するタイムラインのターゲットパス。

戻り値

Object - xMin、xMax、yMin、yMax. の各プロパティを持つオブジェクトです。

例

次の例では、ムービークリップ square_mc を作成します。このコードはこのムービークリップ用に四角形を描画し、MovieClip.getBounds() を使用してインスタンスの座標値を [出力] パネルに表示します。

```
this.createEmptyMovieClip("square_mc", 1);
square_mc._x = 10;
square_mc._y = 10;
square_mc.beginFill(0xFF0000);
square_mc.moveTo(0, 0);
square_mc.lineTo(100, 0);
square_mc.lineTo(100, 100);
square_mc.lineTo(0, 100);
square_mc.lineTo(0, 0);
square_mc.endFill();
```

```
var bounds_obj:Object = square_mc.getBounds(this);
for (var i in bounds_obj) {
  trace(i+" --> "+bounds_obj[i]);
}
```

[出力] パネルに次の情報が表示されます。

```
yMax --> 110
yMin --> 10
xMax --> 110
xMin --> 10
```

関連項目

getRect (MovieClip.getRect メソッド), globalToLocal (MovieClip.globalToLocal メソッド), localToGlobal (MovieClip.localToGlobal メソッド)

getBytesLoaded (MovieClip.getBytesLoaded メソッド)

```
public getBytesLoaded() : Number
```

ムービークリップについて、既にロード (ストリーミング) されたバイト数を返します。この値と、MovieClip.getBytesTotal() から返される値を比較して、ロード済みのムービークリップの割合を判別することができます。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Number - ロードされたバイト数を示す整数。

例

次の例では、_framesloaded プロパティを使用して、すべてのフレームがロードされた時点で SWF ファイルを開始します。すべてのフレームのロードが完了するまでは、ムービークリップインスタンス loader の _xscale プロパティは比例的に増加し、ロードの進行状況を示すプログレスバーを作成します。

タイムラインのフレーム 1 に次の ActionScript を入力します。

```
var pctLoaded:Number = Math.round(this.getBytesLoaded()/this.getBytesTotal() *
  100);
bar_mc._xscale = pctLoaded;
```

フレーム 2 に次のコードを追加します。

```
if (this._framesloaded<this._totalframes) {
  this.gotoAndPlay(1);
} else {
  this.gotoAndStop(3);
}
```

フレーム 3 以後にコンテンツを配置し、フレーム 3 に次のコードを追加します。

```
stop();
```

関連項目

getBytesTotal (MovieClip.getBytesTotal メソッド)

getBytesTotal (MovieClip.getBytesTotal メソッド)

```
public getBytesTotal() : Number
```

ムービークリップのサイズをバイト単位で返します。外部にあるムービークリップ (ターゲットまたはレベルにロードされるルート SWF ファイルまたはムービークリップ) の場合、戻り値は SWF ファイルの圧縮されていないサイズです。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Number - ムービークリップの総バイト数を示す整数。

例

次の例では、_framesloaded プロパティを使用して、すべてのフレームがロードされた時点で SWF ファイルを開始します。すべてのフレームのロードが完了するまでは、ムービークリップインスタンス loader の _xscale プロパティは比例的に増加し、ロードの進行状況を示すプログレスバーを作成します。

タイムラインのフレーム 1 に次の ActionScript を入力します。

```
var pctLoaded:Number = Math.round(this.getBytesLoaded()/
  this.getBytesTotal()*100);
bar_mc._xscale = pctLoaded;
```

フレーム 2 に次のコードを追加します。

```
if (this._framesloaded<this._totalframes) {
  this.gotoAndPlay(1);
} else {
  this.gotoAndStop(3);
}
```

フレーム 3 以後にコンテンツを配置し、フレーム 3 に次のコードを追加します。

```
stop();
```

関連項目

getBytesLoaded (MovieClip.getBytesLoaded メソッド)

getDepth (MovieClip.getDepth メソッド)

```
public getDepth() : Number
```

ムービークリップインスタンスの深度を返します。

各ムービークリップ、ボタン、およびテキストフィールドには関連付けられた一意の深度があり、どのように他のオブジェクトの手前や背後に表示されるかが決まります。深度の値が大きいオブジェクトが手前に表示されます。デザイン時に (オーサリングツールで) 作成されるコンテンツは、最初に深度 -16383 が関連付けられます。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 6

戻り値

Number - ムービークリップの深度。

例

次のコードは、ステージ上のすべてのムービークリップインスタンスの深度をトレースします。

```
for (var i in this) {
  if (typeof (this[i]) == "movieclip") {
  trace("movie clip '"+this[i]._name+"' is at depth "+this[i].getDepth());
  }
}
```

関連項目

getInstanceAtDepth (MovieClip.getInstanceAtDepth メソッド), getNextHighestDepth (MovieClip.getNextHighestDepth メソッド), swapDepths (MovieClip.swapDepths メソッド), getDepth (TextField.getDepth メソッド), getDepth (Button.getDepth メソッド)

getInstanceAtDepth (MovieClip.getInstanceAtDepth メソッド)

```
public getInstanceAtDepth(depth:Number) : MovieClip
```

特定の深度にムービークリップが既に配置されているかどうかを判別します。MovieClip.attachMovie() メソッド、MovieClip.duplicateMovieClip() メソッド、MovieClip.createEmptyMovieClip() メソッドを使用する前にこのメソッドを使用して、これらのメソッドに渡す深度パラメータに既にムービークリップが置かれているかどうかを調べることができます。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

`depth:Number` - 照会する深度レベルを指定する整数。

戻り値

`MovieClip` - 指定した深度に位置する MovieClip インスタンスへの参照。その深度にムービークリップが存在しない場合は `undefined`。

例

次の例では、ムービークリップインスタンス `triangle` が配置されている深度を [出力] パネルに表示します。

```
this.createEmptyMovieClip("triangle", 1);

triangle.beginFill(0x0000FF, 100);
triangle.moveTo(100, 100);
triangle.lineTo(100, 150);
triangle.lineTo(150, 100);
triangle.lineTo(100, 100);

trace(this.getInstanceAtDepth(1)); // _level0.triangle
```

関連項目

attachMovie (MovieClip.attachMovie メソッド), duplicateMovieClip (MovieClip.duplicateMovieClip メソッド), createEmptyMovieClip (MovieClip.createEmptyMovieClip メソッド), getDepth (MovieClip.getDepth メソッド), getNextHighestDepth (MovieClip.getNextHighestDepth メソッド), swapDepths (MovieClip.swapDepths メソッド)

getNextHighestDepth (MovieClip.getNextHighestDepth メソッド)

```
public getNextHighestDepth() : Number
```

現在のムービークリップ内の同一レベル、同一レイヤー上の他のすべてのオブジェクトよりもムービークリップを前面に表示する場合に、MovieClip.attachMovie() メソッド、MovieClip.duplicateMovieClip() メソッド、MovieClip.createEmptyMovieClip() メソッドに渡す深度値を調べることができます。返される値は 0 以上です。負の値は返されません。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

メモ : バージョン 2 のコンポーネントを使用している場合は、このメソッドを使用しないでください。ステージまたはライブラリにバージョン 2 のコンポーネントがあると、getNextHighestDepth() メソッドの戻り値が 1048676 になることがあります。この深度は、有効な値の範囲外です。バージョン 2 のコンポーネントを使用している場合は、バージョン 2 のコンポーネントの DepthManager クラスを必ず使用してください。

対応バージョン : ActionScript 1.0、Flash Player 7

戻り値

Number - ムービークリップ内で、同一レベル、同一レイヤー上に存在する他のすべてのムービークリップの前面に表示できる最初の深度インデックスを示す整数。

例

次の例では、createEmptyMovieClip() メソッドの depth パラメータとして getNextHighestDepth() メソッドを使用して、3 つのムービークリップインスタンスを描画します。各ムービークリップに、その深度のラベルを付けます。

```
for (i = 0; i < 3; i++) {
  drawClip(i);
}

function drawClip(n:Number):Void {
  this.createEmptyMovieClip("triangle" + n, this.getNextHighestDepth());
  var mc:MovieClip = eval("triangle" + n);
  mc.beginFill(0x00aaFF, 100);
  mc.lineStyle(4, 0xFF0000, 100);
  mc.moveTo(0, 0);
  mc.lineTo(100, 100);
  mc.lineTo(0, 100);
  mc.lineTo(0, 0);
  mc._x = n * 30;
  mc._y = n * 50
  mc.createTextField("label", this.getNextHighestDepth(), 20, 50, 200, 200)
  mc.label.text = mc.getDepth();
}
```

関連項目

getDepth (MovieClip.getDepth メソッド), getInstanceAtDepth (MovieClip.getInstanceAtDepth メソッド), swapDepths (MovieClip.swapDepths メソッド), attachMovie (MovieClip.attachMovie メソッド), duplicateMovieClip (MovieClip.duplicateMovieClip メソッド), createEmptyMovieClip (MovieClip.createEmptyMovieClip メソッド)

getRect (MovieClip.getRect メソッド)

```
public getRect(bounds:Object) : Object
```

シェイプ上の線を除き、bounds パラメータに基づいて、ムービークリップの最小および最大の *x* 座標と *y* 座標を示すプロパティを返します。getRect() から返される値は、MovieClip.getBounds() から返される値と同じか、それより小さくなります。

メモ : ムービークリップのローカル座標をステージ座標に変換するには、MovieClip.localToGlobal() メソッドを使用します。ステージ座標をローカル座標に変換するには、MovieClip.globalToLocal() メソッドを使用します。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

bounds:Object - タイムラインのターゲットパス。このタイムラインの座標系が基準点として使用されます。

戻り値

Object - xMin、xMax、yMin、yMax. の各プロパティを持つオブジェクトです。

例

次の例では、ムービークリップを作成し、その中に線の太さが 4 ピクセルの正方形を描画します。次に、MovieClip.getBounds() メソッドと MovieClip.getRect() メソッドを呼び出して、2 つの差を示します。getBounds() メソッドは、正方形の線の太さを含めた、ムービークリップ全体の座標の最小値と最大値を返します。getRect() メソッドは、正方形の線の太さの 4 ピクセルを除く、座標の最大値と最小値を返します。

```
this.createEmptyMovieClip("square_mc", 1);
square_mc._x = 10;
square_mc._y = 10;
square_mc.beginFill(0xFF0000);
square_mc.lineStyle(4, 0xFF00FF, 100, true, "none", "round", "miter", 1);
square_mc.moveTo(0, 0);
square_mc.lineTo(100, 0);
square_mc.lineTo(100, 100);
square_mc.lineTo(0, 100);
square_mc.lineTo(0, 0);
square_mc.endFill();

var bounds_obj:Object = square_mc.getBounds(this);
trace("getBounds() output:");
for (var i in bounds_obj) {
  trace(i+" --> "+bounds_obj[i]);
}

var rect_obj:Object = square_mc.getRect(this);
trace("getRect() output:");
for (var i in rect_obj) {
  trace(i+" --> "+rect_obj[i]);
}
```

trace() ステートメントの出力は次のようになります。

```
getBounds() output:
yMax --> 112
yMin --> 8
xMax --> 112
xMin --> 8
getRect() output:
yMax --> 110
yMin --> 10
xMax --> 110
xMin --> 10
```

関連項目

getBounds (MovieClip.getBounds メソッド),globalToLocal (MovieClip.globalToLocal メソッド),localToGlobal (MovieClip.localToGlobal メソッド)

getSWFVersion (MovieClip.getSWFVersion メソッド)

```
public getSWFVersion() : Number
```

ムービークリップがパブリックされたときの対象の Flash Player のバージョンを示す整数値を返します。ムービークリップが JPEG ファイル、GIF ファイル、PNG ファイルのいずれかの場合、またはエラーが発生してムービークリップ の SWF バージョンを判別できなかった場合は、-1 を返します。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 7

戻り値

`Number` - ムービークリップにロードされている SWF ファイルがパブリッシュされたときに対象となった Flash Player のバージョンを示す整数。

例

次の例では、新しいコンテナを作成し、`getSWFVersion()` の値を出力します。次に、MovieClipLoader を使用して、Flash Player 7 にパブリッシュされた外部 SWF ファイルをロードし、`onLoadInit` ハンドラが呼び出された後に `getSWFVersion()` の値を出力します。

```
var container:MovieClip = this.createEmptyMovieClip("container",
  this.getUpperEmptyDepth());
var listener:Object = new Object();
listener.onLoadInit = function(target:MovieClip):Void {
  trace("target: " + target.getSWFVersion()); // target: 7
}
var mcLoader:MovieClipLoader = new MovieClipLoader();
mcLoader.addListener(listener);
trace("container: " + container.getSWFVersion()); // container: 8
mcLoader.loadClip("FlashPlayer7.swf", container);
```

getTextSnapshot (MovieClip.getTextSnapshot メソッド)

```
public getTextSnapshot() : TextSnapshot
```

指定されたムービークリップのすべての静止テキストフィールド内のテキストを含む TextSnapshot オブジェクトを返します。ただし、子のムービークリップに存在するテキストは含まれません。このメソッドは、常に TextSnapshot オブジェクトを返します。

Flash はテキストを連結し、それを TextSnapshot オブジェクトに格納します。格納される順序には、ムービークリップ内の静止テキストフィールドのタブインデックスの序列が反映されます。タブインデックス値が存在しないテキストフィールドは、オブジェクト内にランダムな順序で、タブインデックス値が存在するテキストフィールドよりも前に格納されます。フィールドの終端と次のフィールドの開始位置を示す改行やフォーマット文字はありません。

メモ : Flash では、静止テキストに対してタブインデックス値を指定することはできません。ただし、Macromedia FlashPaper など、タブインデックス値を指定できる製品もあります。

TextSnapshot オブジェクトの内容は動的には変化しません。ムービークリップが別のフレームに移動した場合、または何らかの形で変更された場合 (ムービークリップが追加または削除された場合など)、TextSnapshot オブジェクトにムービークリップの最新のテキストが保持されていないことがあります。オブジェクトの内容を最新に保つには、必要に応じてこのコマンドを再発行する必要があります。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 7 - Flash Player 6 以降用にパブリッシュされた SWF ファイル。Flash Player 7 以降で再生されます。

戻り値

`TextSnapshot` - ムービークリップからの静止テキストを含む TextSnapshot オブジェクト。

例

次の例は、このメソッドの使用方法を示しています。このコードを使用するには、"TextSnapshot Example" というテキストを含む静止テキストフィールドをステージに配置してください。

```
var textSnap:TextSnapshot = this.getTextSnapshot();
trace(textSnap.getText(0, textSnap.getCount(), false));
```

関連項目

`TextSnapshot`

getURL (MovieClip.getURL メソッド)

```
public getURL(url:String, [window:String], [method:String]) : Void
```

指定された URL から、指定されたウィンドウにドキュメントを読み込みます。getURL() メソッドでは、GET メソッドまたは POST メソッドを使用して、URL で定義されている別のアプリケーションに変数を渡すこともできます。

Flash コンテンツをホストする Web ページでは、allowScriptAccess 属性を明示的に設定して、Flash Player のスクリプトを HTML コードから許可または拒否する必要があります。この HTML コードは、Internet Explorer では PARAM タグに、Netscape Navigator では EMBED タグにあります。

- allowScriptAccess が "never" の場合、送信スクリプトは常に失敗します。
- allowScriptAccess が "always" の場合、送信スクリプトは常に成功します。
- allowScriptAccess が "sameDomain" (バージョン 8 以降の SWF ファイルでサポート) の場合、SWF ファイルがホスト側 Web ページと同じドメインに存在すれば、送信スクリプトが許可されます。
- HTML ページで allowScriptAccess が指定されていない場合、デフォルト値は、バージョン 8 の SWF ファイルでは "sameDomain"、バージョン 8 より前の SWF ファイルでは "always" です。

このメソッドを使用するときは、Flash Player セキュリティモデルを考慮してください。Flash Player 8 では、呼び出し元 SWF ファイルがローカルファイルシステムのサンドボックスにあり、リソースがローカルでない場合、このメソッドは許可されません。

詳細については、次の参照先を参照してください。

- 『ActionScript 2.0 の学習』の「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

url:String - ドキュメントを取得するための URL。

window:String (オプション) - ドキュメントのロード先のウィンドウまたは HTML フレームを指定する名前、フレーム、または式。次の予約済みターゲット名のいずれかを使用することもできます。_self は現在のウィンドウ内の現在のフレームを指定します。_blank は新しいウィンドウを指定します。_parent は現在のフレームの親を指定します。_top は現在のウィンドウ内のトップレベルのフレームを指定します。

method:String(オプション)- ロード対象の SWF ファイルに関連付けられた変数を送信するためのメソッドを指定するストリング。"GET" または "POST" のいずれかを指定します。変数が存在しない場合は、このパラメータを省略します。それ以外の場合は、変数のロードに GET メソッドと POST メソッドのどちらを使用するかを指定してください。GET を指定すると、変数が URL の最後に付加されます。このメソッドは、変数の数が少ない場合に使用します。POST を指定すると、変数が別の HTTP ヘッダーで送信されます。このメソッドは、個々の変数のストリングが長い場合に使用します。

例

次の ActionScript は、ムービークリップインスタンスを作成し、新しいブラウザウィンドウでアドビシステムズ社の Web サイトを開きます。

```
this.createEmptyMovieClip("loader_mc", this.getNextHighestDepth());
loader_mc.getURL("http://www.adobe.com", "_blank");
```

getURL() メソッドでは、次のようにリモートのサーバー側のスクリプトに変数を送信することもできます。

```
this.createEmptyMovieClip("loader_mc", this.getNextHighestDepth());
loader_mc.username = "some user input";
loader_mc.password = "random string";
loader_mc.getURL("http://www.flash-mx.com/mm/viewscope.cfm", "_blank", "GET");
```

これらの例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

このメソッドを使用するときは、Flash Player セキュリティモデルを考慮してください。

- Flash Player 8 では、呼び出し元 SWF ファイルがローカルファイルシステムのサンドボックスにあり、リソースがローカルでない場合、MovieClip.getURL() メソッドは許可されません。

詳細については、次の参照先を参照してください。

- 『ActionScript 2.0 の学習』の「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

関連項目

getURL 関数, sendAndLoad (LoadVars.sendAndLoad メソッド), send (LoadVars.send メソッド)

globalToLocal (MovieClip.globalToLocal メソッド)

```
public globalToLocal(pt:Object) : Void
```

pt オブジェクトをステージ (グローバル) 座標からムービークリップ内 (ローカル) の座標に変換します。

MovieClip.localToGlobal() メソッドを使用すると、指定された *x* 座標と *y* 座標を、ステージの左上隅を基準にした相対値から、特定のムービークリップの左上隅を基準にした相対値に変換できます。

最初に x と y の 2 つのプロパティを持つ汎用オブジェクトを作成する必要があります。これらの x 値と y 値 (x および y と呼ぶ必要があります) は、ステージの左上隅を基準にしているため、グローバル座標と呼ばれます。x プロパティは、左上隅からの水平オフセットを表します。つまり、そのポイントが基準点からどれだけ右に配置されているかを示します。たとえば、x = 50 の場合、そのポイントは左上隅から 50 ピクセル右に配置されていることになります。y プロパティは、左上隅からの垂直オフセットを表します。つまり、そのポイントが基準点からどれだけ下に配置されているかを示します。たとえば、y = 20 の場合、そのポイントは左上隅から 20 ピクセル下に配置されていることになります。次のコードでは、これらの座標を持つ汎用オブジェクトを作成します。

```
var myPoint:Object = new Object();
myPoint.x = 50;
myPoint.y = 20;
```

代わりに、オブジェクトを作成し、同時にリテラル Object 値を使用して値を割り当てることもできます。

```
var myPoint:Object = {x:50, y:20};
```

グローバル座標を使用して point オブジェクトを作成したら、その座標をローカル座標に変換できます。globalToLocal() メソッドは、パラメータとして受け取る汎用オブジェクトの x 値と y 値を変更するため、値を返しません。このメソッドは、ステージ (グローバル座標) を基準にした値から、特定のムービークリップ (ローカル座標) を基準にした値に変更します。

たとえば、ムービークリップを作成してポイント (_x:100, _y:100) に配置し、ステージの左上隅 (x:0, y:0) を表すグローバルポイントを globalToLocal() メソッドに渡すと、このメソッドはその x 値と y 値をローカル座標、この場合 (x:-100, y:-100) に変換します。この変換が行われるのは、*x* 座標と *y* 座標がこの時点で、ステージの左上隅でなくムービークリップの左上隅を基準にして表現されているためです。値が負になるのは、ムービークリップの左上隅からステージの左上隅に到達するのに、100 ピクセル左 (負の *x*)、かつ 100 ピクセル上 (負の *y*) に移動する必要があるためです。

ムービークリップの座標は、MovieClip の *x* 値と *y* 値を設定する MovieClip のプロパティである _x と _y を使用して表現されていました。ただし、汎用オブジェクトでは、アンダースコアなしの x と y を使用します。次のコードでは、*x* 値と *y* 値をローカル座標に変換します。

```
var myPoint:Object = {x:0, y:0}; // Create your generic point object.
this.createEmptyMovieClip("myMovieClip", this.getNextHighestDepth());
myMovieClip._x = 100; // _x for movieclip x position
myMovieClip._y = 100; // _y for movieclip y position
```

```
myMovieClip.globalToLocal(myPoint);
trace ("x: " + myPoint.x); // -100
trace ("y: " + myPoint.y); // -100
```

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

`pt:Object` - 汎用の Object クラスを使用して作成したオブジェクトの名前または識別子。このオブジェクトのプロパティで *x* 座標と *y* 座標を指定します。

例

次の ActionScript を、イメージ photo1.jpg と同じディレクトリ内にある FLA ファイルまたは AS ファイルに追加します。

```
this.createTextField("coords_txt", this.getNextHighestDepth(), 10, 10, 100, 22);
coords_txt.html = true;
coords_txt.multiline = true;
coords_txt.autoSize = true;
this.createEmptyMovieClip("target_mc", this.getNextHighestDepth());
target_mc._x = 100;
target_mc._y = 100;
target_mc.loadMovie("photo1.jpg");

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
   var point:Object = {x:_xmouse, y:_ymouse};
   target_mc.globalToLocal(point);
   var rowHeaders = "<b>   \t</b><b>_x\t</b><b>_y</b>";
   var row_1 = "_root\t"+_xmouse+"\t"+_ymouse;
   var row_2 = "target_mc\t"+point.x+"\t"+point.y;
   coords_txt.htmlText = "<textformat tabstops='[100, 150]'>";
   coords_txt.htmlText += rowHeaders;
   coords_txt.htmlText += row_1;
   coords_txt.htmlText += row_2;
   coords_txt.htmlText += "</textformat>";
};
Mouse.addListener(mouseListener);
```

この例で使用している `MovieClip.getNextHighestDepth()` メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、`MovieClip.getNextHighestDepth()` メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

getBounds (MovieClip.getBounds メソッド),localToGlobal (MovieClip.localToGlobal メソッド),Object

gotoAndPlay (MovieClip.gotoAndPlay メソッド)

```
public gotoAndPlay(frame:Object) : Void
```

指定されたフレームで SWF ファイルの再生を開始します。フレーム以外にシーンも指定するには、gotoAndPlay() を使用します。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

frame:Object - 再生ヘッドの送り先となるフレーム番号を表す数値、または再生ヘッドの送り先となるフレームのラベルを表すストリング。

例

次の例では、_framesloaded プロパティを使用して、すべてのフレームがロードされた時点で SWF ファイルを開始します。すべてのフレームのロードが完了するまでは、ムービークリップインスタンス loader の _xscale プロパティは比例的に増加し、ロードの進行状況を示すプログレスバーを作成します。

タイムラインのフレーム 1 に次の ActionScript を入力します。

```
var pctLoaded:Number = Math.round(this.getBytesLoaded()/
  this.getBytesTotal()*100);
bar_mc._xscale = pctLoaded;
```

フレーム 2 に次のコードを追加します。

```
if (this._framesloaded<this._totalframes) {
  this.gotoAndPlay(1);
} else {
  this.gotoAndStop(3);
}
```

フレーム 3 以後にコンテンツを配置し、フレーム 3 に次のコードを追加します。

```
stop();
```

関連項目

gotoAndPlay 関数,play 関数

gotoAndStop (MovieClip.gotoAndStop メソッド)

```
public gotoAndStop(frame:Object) : Void
```

このムービークリップの指定されたフレームに再生ヘッドを送り、そこで停止させます。フレーム以外にシーンも指定するには、gotoAndStop() メソッドを使用します。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

frame:Object - 再生ヘッドを送る先のフレーム番号。

例

次の例では、_framesloaded プロパティを使用して、すべてのフレームがロードされた時点で SWF ファイルを開始します。すべてのフレームのロードが完了するまでは、ムービークリップインスタンス loader の _xscale プロパティは比例的に増加し、ロードの進行状況を示すプログレスバーを作成します。

タイムラインのフレーム 1 に次の ActionScript を入力します。

```
var pctLoaded:Number = Math.round(this.getBytesLoaded()/
  this.getBytesTotal()*100);
bar_mc._xscale = pctLoaded;
```

フレーム 2 に次のコードを追加します。

```
if (this._framesloaded<this._totalframes) {
  this.gotoAndPlay(1);
} else {
  this.gotoAndStop(3);
}
```

フレーム 3 以後にコンテンツを配置し、フレーム 3 に次のコードを追加します。

```
stop();
```

関連項目

gotoAndStop 関数 , stop 関数

_height (MovieClip._height プロパティ)

```
public _height : Number
```

ピクセル単位で示したムービークリップの高さです。

対応バージョン : ActionScript 1.0、Flash Player 4

例

次のコード例では、ムービークリップの高さと幅を [出力] パネルに表示します。

```
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var image_mcl:MovieClipLoader = new MovieClipLoader();
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  trace(target_mc._name+" = "+target_mc._width+" X "+target_mc._height+"
  pixels");
};
image_mcl.addListener(mclListener);

image_mcl.loadClip("example.jpg", image_mc);
```

この例で使用している `MovieClip.getNextHighestDepth()` メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、`MovieClip.getNextHighestDepth()` メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

この例で使用されている MovieClipLoader クラスでは、Flash Player 7 以降が必要です。

関連項目

`_width (MovieClip._width プロパティ )`

_highquality (MovieClip._highquality プロパティ)

```
public _highquality : Number
```

非推奨 Flash Player 7 以降では使用しないでください。このプロパティの代わりに `MovieClip._quality` を使用します。

現在の SWF ファイルに適用されるアンチエイリアスのレベルを指定します。ビットマップスムージングを常にオンにして最高品質を適用するには、2 (最高品質) を指定します。アンチエイリアス処理を適用するには、1(高品質) を指定します。SWF ファイルにアニメーションが含まれない場合は、ビットマップは滑らかになります。アンチエイリアス処理を避けるには、0 (低品質) を指定します。このプロパティでグローバル `_highquality` プロパティの設定を上書きできます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の ActionScript は、SWF ファイルに対して最高品質のアンチエイリアス処理の適用を指定します。

```
my_mc._highquality = 2;
```

関連項目

_quality (MovieClip._quality プロパティ), _quality プロパティ

hitArea (MovieClip.hitArea プロパティ)

```
public hitArea : Object
```

ムービークリップのヒット領域となる別のムービークリップを指定します。hitArea プロパティが存在しないか、このプロパティの値が null または undefined の場合は、ムービークリップ自体がヒット領域として使用されます。hitArea プロパティの値は、ムービークリップインスタンスへの参照である場合があります。

hitArea プロパティはいつでも変更できます。このプロパティを変更したムービークリップには新しいヒット領域の動作が直ちに反映されます。ヒット領域として指定したムービークリップは可視状態である必要はありません。不可視状態であっても、そのグラフィカルシェイプをヒット領域として検出できます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ムービークリップ circle_mc をムービークリップ square_mc のヒット領域として設定します。これらの 2 つのムービークリップをステージ上に配置し、ドキュメントをテストします。ユーザーが circle_mc をクリックすると、ムービークリップ square_mc は、クリックされたことをトレースします。

```
square_mc.hitArea = circle_mc;
square_mc.onRelease = function() {
  trace("hit! "+this._name);
};
```

ムービークリップ circle_mc の visible プロパティを false に設定して、square_mc のヒット領域を非表示にすることもできます。

```
circle_mc._visible = false;
```

関連項目

hitTest (MovieClip.hitTest メソッド)

hitTest (MovieClip.hitTest メソッド)

```
public hitTest() : Boolean
```

ムービークリップを評価して、それが target、または x 座標と y 座標のパラメータで示されるヒット領域と重なっている (または交差している) かどうかを確認します。

シンタックス 1: shapeFlag の設定に従って、*x* 座標と *y* 座標を、指定されたインスタンスのシェイプまたは境界ボックスと比較します。shapeFlag を true に設定すると、ステージ上でインスタンスが実際に占有する領域のみが評価されます。*x* と *y* が任意の点で重なる場合は、true が返されます。この評価は、ムービークリップが、指定されたヒット領域またはホットスポット領域内にあるかどうかを判別する場合に有効です。このシンタックスの場合、メソッドのシグネチャは次のようになります。

```
public hitTest(x:Number, y:Number, [shapeFlag:Boolean]):Boolean
```

シンタックス 2 : target の境界ボックスと指定されたインスタンスの境界ボックスを評価し、両者が任意の点で重なるか交差する場合に true を返します。このシンタックスの場合、メソッドのシグネチャは次のようになります。

```
public hitTest(target:Object):Boolean
```

パラメータ : x: Number はステージ上のヒット領域の *x* 座標、y: Number は、ステージ上のヒット領域の *y* 座標です。*x* 座標と *y* 座標は、グローバル座標空間で定義されます。また、shapeFlag: Boolean は、指定したインスタンスのシェイプ全体を評価するか (true)、境界ボックスだけを評価するか (false) を指定するブール値です。このパラメータは、x 座標と y 座標のパラメータでヒット領域を指定する場合にのみ指定できます。target: Object は、ムービークリップと交差するか重なるヒット領域のターゲットパスです。target パラメータは通常、ボタンまたはテキスト入力フィールドを表します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Boolean - ブール値。指定したヒット領域にムービークリップが重なっている (交差している) 場合は true 、それ以外の場合は false を返します。

例

次の例では、hitTest() を使用して、ユーザーがマウスボタンを離したときにムービークリップ circle_mc がムービークリップ square_mc と重なっている (交差している) かどうかを判別します。

```
square_mc.onPress = function() {
  this.startDrag();
};
square_mc.onRelease = function() {
  this.stopDrag();
  if (this.hitTest(circle_mc)) {
  trace("you hit the circle");
  }
};
```

次の例では、hitTest() メソッドを使用して、ユーザーがマウスボタンを離したときにムービークリップ triangle_mc がポイント (100, 75) (三角形の初期位置の中心から約 50 ピクセル右のポイント) と重なっているかどうかを判別します。

```
createEmptyMovieClip("triangle_mc", getNextHighestDepth());
triangle_mc.beginFill(0x006090);
triangle_mc.moveTo(50, 50);
triangle_mc.lineTo(100, 150);
triangle_mc.lineTo(0, 150);
triangle_mc.lineTo(50, 50);

var hit_X = 100;
var hit_Y = 75;
var shapeFlag = true;

triangle_mc.onPress = function() {
  this.startDrag();
};

triangle_mc.onRelease = function() {
  this.stopDrag();
  if (this.hitTest(hit_X, hit_Y, shapeFlag)) {
    trace("Hit.");
  }
};
```

関連項目

getBounds (MovieClip.getBounds メソッド),globalToLocal (MovieClip.globalToLocal メソッド),localToGlobal (MovieClip.localToGlobal メソッド),hitTest (BitmapData.hitTest メソッド)

lineGradientStyle (MovieClip.lineGradientStyle メソッド)

```
public lineGradientStyle(fillType:String, colors:Array, alphas:Array,
  ratios:Array, matrix:Object, [spreadMethod:String],
  [interpolationMethod:String], [focalPointRatio:Number]) : Void
```

lineTo() メソッドおよび curveTo() メソッドの今後の呼び出しに使用する線のスタイルを指定します。このスタイルは、次に lineStyle() メソッドまたは lineGradientStyle() メソッドを異なるパラメータで呼び出すまで使用されます。パスの描画中に lineGradientStyle() メソッドを呼び出し、パス内の線のセグメントごとに異なるスタイルを指定できます。

メモ : lineGradientStyle() を呼び出す前に lineStyle() を呼び出して線を有効にします。そうしないと、線のスタイルが undefined のままになります。

メモ : clear() を呼び出すと、線のスタイルの設定が undefined に戻ります。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

`fillType:String` - `"linear"` または `"radial"` を指定できます。

`colors:Array` - グラデーションで使用する RGB 16 進カラー値の配列 (赤 0xFF0000、青 0x0000FF など)。最大 15 色まで指定できます。各色について、`alphas` パラメータと `ratios` パラメータで対応する値を必ず指定してください。

`alphas:Array` - *`colors`* 配列内の各色に対応するアルファ値の配列。有効な値は 0 ～ 100 です。0 未満の値の場合は 0 が、100 を超える値の場合は 100 が適用されます。

`ratios:Array` - 色分布の比率の配列。有効な値は 0 ～ 255 です。この値は、100% でサンプリングされる色の幅の割合をパーセントで定義します。`colors` パラメータの値ごとに、値を指定してください。

たとえば、青と緑の 2 色を含む線状グラデーションの場合、次の図は、`ratios` 配列のさまざまな値に基づいて、グラデーションで配置される色を示します。

`ratios`	グラデーション
`[0, 127]`	
`[0, 255]`	
`[127, 255]`	

配列内の値は、`[0, 63, 127, 190, 255]` のように順に増やしていく必要があります。

`matrix:Object` - 次のいずれかの組み合わせのプロパティを持つ変換マトリックスのオブジェクト。

- プロパティ `a`、`b`、`c`、`d`、`e`、`f`、`g`、`h` および `i` を使用して、3 x 3 のマトリックスを次の形式で記述できます。

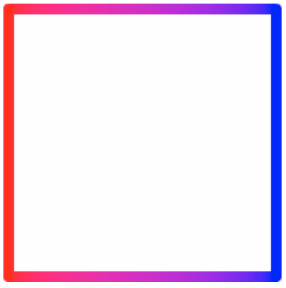
```
a b c
d e f
g h i
```

次の例では、これらのプロパティを持つオブジェクトを `matrix` パラメータに指定した `lineGradientFill()` メソッドを使用します。

```
this.createEmptyMovieClip("gradient_mc", 1);
with (gradient_mc) {
  colors = [0xFF0000, 0x0000FF];
    alphas = [100, 100];
    ratios = [0, 0xFF];
```

```
    matrix = {a:200, b:0, c:0, d:0, e:200, f:0, g:200, h:200, i:1};
    spreadMethod = "reflect";
    interpolationMethod = "linearRGB";
    focalPointRatio = 0.9;
  lineStyle(8);
    lineGradientStyle("linear", colors, alphas, ratios, matrix,
    spreadMethod, interpolationMethod, focalPointRatio);
    moveTo(100, 100);
    lineTo(100, 300);
    lineTo(300, 300);
    lineTo(300, 100);
    lineTo(100, 100);
    endFill();
}
```

このコードでは、画面上に次のように描画されます。



- matrixType, x, y, w, h, r.

  各プロパティの内容は次のとおりです。matrixType はストリング "box" です。x はグラデーションの左上隅の、親クリップの基準点からの相対的な水平座標、y はグラデーションの左上隅の、親クリップの基準点からの相対的な垂直座標を示します。また、w はグラデーションの幅、h はグラデーションの高さ、r はグラデーションのラジアン単位の回転です。

  次の例では、これらのパラメータを matrix パラメータに指定した lineGradientFill() を使用します。

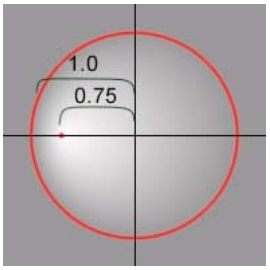
```
this.createEmptyMovieClip("gradient_mc", 1);
with (gradient_mc) {
  colors = [0xFF0000, 0x0000FF];
  alphas = [100, 100];
  ratios = [0, 0xFF];
  matrix = {matrixType:"box", x:100, y:100, w:200, h:200, r:(45/
  180)*Math.PI};
  spreadMethod = "reflect";
  interpolationMethod = "linearRGB";
  lineStyle(8);
  lineGradientStyle("linear", colors, alphas, ratios, matrix,
    spreadMethod, interpolationMethod);
  moveTo(100, 100);
```

```
        lineTo(100, 300);
        lineTo(300, 300);
        lineTo(300, 100);
        lineTo(100, 100);
        endFill();
    }
```

spreadMethod:String (オプション) - グラデーションの塗りのモードを表すストリング。有効な値は "pad"、"reflect"、または "repeat" です。

interpolationMethod:String (オプション) - 有効な値は "RGB" または "linearRGB" です。

focalPointRatio:Number (オプション) - グラデーションの焦点の位置を表す数値。0 は焦点が中央にあること、1 は焦点がグラデーション円のいずれかの境界にあること、-1 は焦点がグラデーション円のもう一方の境界にあることを示します。-1 未満の値は -1 に、1 より大きい値は 1 に丸められます。次のイメージは、focalPointRatio を -0.75 に設定したグラデーションを示しています。



例

次のコードでは、両方のメソッドを使用し、赤と青のグラデーションの塗りで 2 つの重なった矩形を描画します。

```
this.createEmptyMovieClip("gradient_mc", 1);
with (gradient_mc) {
  colors = [0xFF0000, 0x0000FF];
  alphas = [100, 100];
  ratios = [0, 0xFF];
  matrix = {a:500, b:0, c:0, d:0, e:200, f:0, g:350, h:200, i:1};
  lineStyle(16);
  lineGradientStyle("linear", colors, alphas, ratios, matrix);
  moveTo(100, 100);
  lineTo(100, 300);
  lineTo(600, 300);
  lineTo(600, 100);
  lineTo(100, 100);
  endFill();
  matrix2 = {matrixType:"box", x:100, y:310, w:500, h:200, r:(30/180)*Math.PI};
  lineGradientStyle("linear", colors, alphas, ratios, matrix2);
  moveTo(100, 320);
  lineTo(100, 520);
```

```
  lineTo(600, 520);
  lineTo(600, 320);
  lineTo(100, 320);
  endFill();
}
```

このコードにより、次のイメージが描画されます。このイメージは 50% 縮小されています。

関連項目

beginGradientFill (MovieClip.beginGradientFill メソッド), lineStyle (MovieClip.lineStyle メソッド), lineTo (MovieClip.lineTo メソッド), moveTo (MovieClip.moveTo メソッド)

lineStyle (MovieClip.lineStyle メソッド)

```
public lineStyle(thickness:Number, rgb:Number, alpha:Number,
  pixelHinting:Boolean, noScale:String, capsStyle:String, jointStyle:String,
  miterLimit:Number) : Void
```

lineTo() メソッドおよび curveTo() メソッドで今後の呼び出しに使用する線のスタイルを指定します。このスタイルは、次に lineStyle() メソッドを異なるパラメータで呼び出すまで使用されます。パスの描画中に lineStyle() メソッドを呼び出し、パス内の線のセグメントごとに異なるスタイルを指定できます。

メモ : clear() メソッドを呼び出すと、線のスタイルの設定値が undefined に戻ります。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 6 - pixelHinting、noScale、capsStyle、jointStyle、および miterLimit の各パラメータが Flash Player 8 で追加されました。

パラメータ

`thickness:Number` - 線の太さをポイント単位で示す整数。有効な値は 0 ～ 255 です。数値を指定しない場合、または `undefined` を指定した場合、線は描画されません。0 未満の値を指定した場合は 0 が使用されます。0 は極細線です。最大の太さは 255 です。255 を超える値を指定した場合は 255 が使用されます。

`rgb:Number` - 色を表す 16 進値。たとえば、赤は 0xFF0000、青は 0x0000FF で表します。値を指定しないと、0x000000 (黒) が使用されます。

`alpha:Number` - 線の色のアルファ値を示す整数。有効な値は 0 ～ 100 です。値を指定しないと 100 (ソリッド) が使用されます。0 未満の値を指定しても 0 が適用されます。100 を超える値を指定しても 100 が適用されます。

`pixelHinting:Boolean` - Flash Player 8 で追加されました。ピクセル全体に対して線をヒンティングするかどうかを指定するブール値です。この値は、曲線のアンカーの位置と線のサイズ自身の両方に影響を与えます。`pixelHinting` を `true` に設定すると、全ピクセル幅に線幅がヒンティングされます。`pixelHinting` を `false` に設定すると、曲線と直線で非連続が発生することがあります。たとえば、次の図は、`lineStyle()` メソッドで使用される `pixelHinting` パラメータの設定が異なるだけで他はまったく同じ 2 つの角丸矩形がどのようにレンダリングされるかを示したものです (違いが分かるようにイメージは 200% に拡大されています)。

値を指定しない場合、線でピクセルのヒンティングが使用されません。

`noScale:String` - Flash Player 8 で追加されました。線の拡大・縮小方法を指定するストリングです。有効な値は次のとおりです。

- `"normal"`– 常に太さを拡大・縮小します (デフォルト)。
- `"none"`– 太さを拡大・縮小しません。
- `"vertical"`– オブジェクトを垂直方向にのみ拡大・縮小する場合、太さを拡大・縮小しません。
- `"horizontal"`– オブジェクトを水平方向にのみ拡大･縮小する場合、太さを拡大･縮小しません。

`capsStyle:String` - Flash Player 8 で追加されました。線の終端のキャップの種類を指定するストリングです。有効な値は `"round"`、`"square"`、および `"none"` です。この値を指定しない場合は、丸いキャップが使用されます。

たとえば、次の図は capsStyle のさまざまな設定を示します。それぞれの設定で、図の青い線は太さ 30 で、capsStyle が適用されています。重ね合わせた黒い線は太さ 1 で、capsStyle は適用されていません。

jointStyle: String - Flash Player 8 で追加されました。角で使用する接合点の外観の種類を指定するストリングです。有効な値は "round"、"miter"、および "bevel" です。この値を指定しないと、角丸 ("round") の接合点が使用されます。

たとえば、次の図は capsStyle のさまざまな設定を示します。それぞれの設定で、角のある青い線は太さ 30 で、jointStyle が適用されています。重ね合わせた黒い線は太さ 1 で、jointStyle は適用されていません。

jointStyle を "miter" に設定すると、miterLimit パラメータでマイターの長さを制限できます。

miterLimit: Number - Flash Player 8 で追加されました。マイターが切り取られる限度を示す数値です。有効な値の範囲は 1 ～ 255 で、この範囲を超えた値は 1 または 255 に丸められます。この値は、jointStyle を "miter" に設定した場合にのみ有効です。値が指定されていない場合は 3 が使用されます。miterLimit の値により、線の接合点を越えてマイターを延長できる長さが決定します。マイターを延長できる長さは、この値と線の thickness が乗算されたものです。たとえば、miterLimit の値が 2.5 で、thickness が 10 ピクセルの場合、マイターは 25 ピクセルで切り取られます。

例として、次のような角のある線を考えます。どの線も thickness を 20 に設定して描画されていますが、miterLimit はそれぞれ 1、2、4 に設定されています。重ね合わされた黒の基準線は、接合点を示しています。

miterLimit のそれぞれの値には、マイターが切り取られる独自の最大角度が存在します。次の表に、いくつかの例を示します。

Value of miterLimit の値：	角度が以下より小さい場合は切り取られる
1.414	90 度
2	60 度
4	30 度
8	15 度

例

次の例では、5 ピクセルのマゼンタ色の実線を使用して、塗りなし、ピクセルヒンティングあり、線の拡大・縮小なし、キャップなし、miterLimit が 1 に設定されたマイタージョイント付きの三角形を描画します。

```
this.createEmptyMovieClip("triangle_mc", this.getNextHighestDepth());
triangle_mc.lineStyle(5, 0xff00ff, 100, true, "none", "round", "miter", 1);
triangle_mc.moveTo(200, 200);
triangle_mc.lineTo(300, 300);
triangle_mc.lineTo(100, 300);
triangle_mc.lineTo(200, 200);
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

関連項目

beginFill (MovieClip.beginFill メソッド), beginGradientFill (MovieClip.beginGradientFill メソッド), clear (MovieClip.clear メソッド), curveTo (MovieClip.curveTo メソッド), lineTo (MovieClip.lineTo メソッド), moveTo (MovieClip.moveTo メソッド)

lineTo (MovieClip.lineTo メソッド)

```
public lineTo(x:Number, y:Number) : Void
```

現在の描画位置から (x, y) まで、現在の線のスタイルを使用して線を描画します。その後で、現在の描画位置は (x, y) に設定されます。描画先のムービークリップに Flash の描画ツールで作成したコンテンツが含まれている場合、lineTo() の呼び出しの結果はこのコンテンツの下に描画されます。moveTo() メソッドを呼び出す前に lineTo() メソッドを呼び出すと、現在の描画位置はデフォルトの (0, 0) になります。いずれかのパラメータを省略すると、このメソッドは失敗し、現在の描画位置は変更されません。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

x:Number - 親ムービークリップの基準点からの相対的な水平座標を示す整数。

y:Number - 親ムービークリップの基準点からの相対的な垂直座標を示す整数。

例

次の例では、5 ピクセルのマゼンタ色の実線と、部分的に透明な青の塗りを使用して三角形を描画します。

```
this.createEmptyMovieClip("triangle_mc", 1);
triangle_mc.beginFill(0x0000FF, 30);
triangle_mc.lineStyle(5, 0xFF00FF, 100);
triangle_mc.moveTo(200, 200);
triangle_mc.lineTo(300, 300);
triangle_mc.lineTo(100, 300);
triangle_mc.lineTo(200, 200);
triangle_mc.endFill();
```

関連項目

beginFill (MovieClip.beginFill メソッド),createEmptyMovieClip (MovieClip.createEmptyMovieClip メソッド),endFill (MovieClip.endFill メソッド),lineStyle (MovieClip.lineStyle メソッド),moveTo (MovieClip.moveTo メソッド)

loadMovie (MovieClip.loadMovie メソッド)

```
public loadMovie(url:String, [method:String]) : Void
```

元の SWF ファイルの再生中に SWF、JPEG、GIF、または PNG の各ファイルを Flash Player のムービークリップ内にロードします。非アニメーション GIF ファイル、PNG ファイル、およびプログレッシブ JPEG ファイルのサポートが Flash Player 8 で追加されました。アニメーション GIF を読み込むと、先頭のフレームのみ表示されます。非プログレッシブ JPEG ファイルは、Flash Player 6 以降でサポートされます。

ヒント : ダウンロードの進捗状況を監視するには、`loadMovie()` メソッドではなく、`MovieClipLoader.loadClip()` メソッドを使用します。

`loadMovie()` メソッドを使用しない場合は、Flash Player が 1 つの SWF ファイルを表示して終了します。`loadMovie()` メソッドを使用すると、複数の SWF ファイルを同時に表示し、別の HTML ドキュメントをロードせずに SWF ファイルを切り替えることができます。

ムービークリップ内にロードした SWF ファイルまたはイメージは、そのムービークリップの位置、回転、および拡大・縮小の各プロパティを継承します。ムービークリップのターゲットパスを使用して、ロードした SWF ファイルをターゲットとして設定できます。

`loadMovie()` メソッドを呼び出すときは、次のサンプルコードに示すように、ロード先のムービー内で `MovieClip._lockroot` プロパティを `true` に設定します。ロード先のムービー内で `_lockroot` を `true` に設定しない場合、ロードされたムービー内の `_root` への参照は、ロードされたムービーの `_root` ではなく、ロード先の `_root` をポイントします。

```
myMovieClip._lockroot = true;
```

`loadMovie()` メソッドでロードした SWF ファイルまたはイメージを削除するには、`MovieClip.unloadMovie()` メソッドを使用します。

アクティブな SWF ファイルを保持して、新しいデータをその中にロードするには、`MovieClip.loadVariables()` メソッド、`XML` オブジェクト、Flash Remoting オブジェクトまたは Runtime Shared オブジェクトを使用します。

イベントハンドラと `MovieClip.loadMovie()` を併用すると、予期しない結果が生じる可能性があります。`on()` を使用してイベントハンドラをボタンに割り当てるか、`MovieClip.onPress()` のようなイベントハンドラメソッドを使用してダイナミックハンドラを作成した後、`loadMovie()` を呼び出すと、新しいコンテンツがロードされた後にイベントハンドラが維持されません。一方、`onClipEvent()` または `on()` を使用してイベントハンドラをムービークリップに割り当てた後、そのムービークリップで `loadMovie()` を呼び出すと、新しいコンテンツがロードされた後にイベントハンドラが維持されます。

このメソッドを使用するときは、Flash Player セキュリティモデルを考慮してください。

Flash Player 8 :

- 呼び出し元のムービークリップがローカルファイルシステムのサンドボックスにあり、ロードするムービークリップがネットワーク上のサンドボックスにある場合、ロードできません。
- 呼び出し元 SWF ファイルがネットワーク上のサンドボックスにあり、ロードするムービークリップがローカルにある場合、ロードできません。
- 信頼できるローカルのサンドボックスまたはネットワーク接続したローカルのサンドボックスからネットワーク上のサンドボックスにアクセスするには、クロスドメインポリシーファイルを使用して Web サイトで許可する必要があります。
- ローカルファイルシステムのサンドボックスにあるムービークリップでは、ネットワーク接続したローカルのサンドボックスにあるムービークリップをスクリプト処理できません。その逆も同様です。

Flash Player 7 以降 :

- Web サイトでクロスドメインポリシーファイルを使用して、リソースへのクロスドメインアクセスを許可できます。
- SWF ファイル間のスクリプト処理は、SWF ファイルが置かれているドメインに基づいて制限されます。これらの制限を調整するには、`System.security.allowDomain()` メソッドを使用します。

詳細については、次の参照先を参照してください。

- 『ActionScript 2.0 の学習』の「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 5 - JPEG ファイルのロードは、Flash Player 6 で使用可能になった機能です。非アニメーション GIF ファイル、PNG ファイル、またはプログレッシブ JPEG ファイルのロードは、Flash Player 8 で使用可能になった機能です。

パラメータ

`url:String` - ロードする単一の SWF ファイル、JPEG ファイル、GIF ファイル、または PNG ファイルの絶対 URL または相対 URL。相対パスは、レベル 0 の SWF ファイルが埋め込まれた HTML ファイルを基準にする必要があります。絶対 URL の場合は `http://` や `file:///` などのプロトコル参照を含めて指定します。SWF、JPEG、GIF、または PNG ファイルが複数ある場合は、`loadMovie()` を続けて呼び出します。

`method:String` (オプション) - 変数を送信するための HTTP メソッドを指定します。パラメータはストリング `GET` または `POST` でなければなりません。送信する変数がない場合は、このパラメータを省略します。`GET` メソッドは、変数を URL の最後に追加します。このメソッドは、変数のデータ量が少ないときに使用します。`POST` メソッドは、別の HTTP ヘッダで変数を送信します。このメソッドは、変数のデータ量が多いときに使用します。

例

次の例では、新しいムービークリップを作成した後、その中に子を作成し、子に PNG イメージをロードします。この場合、`loadMovie` の呼び出し前に親に割り当てられていたインスタンスの値がすべて保持されます。

```
var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mc.onRelease = function():Void {
  trace(this.image._url); // http://www.w3.org/Icons/w3c_main.png
}
var image:MovieClip = mc.createEmptyMovieClip("image",
  mc.getNextHighestDepth());
image.loadMovie("http://www.w3.org/Icons/w3c_main.png");
```

この例で使用している `MovieClip.getNextHighestDepth()` メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、`MovieClip.getNextHighestDepth()` メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

`_lockroot (MovieClip._lockroot プロパティ )`, `unloadMovie (MovieClip.unloadMovie メソッド )`, `loadVariables (MovieClip.loadVariables メソッド )`, `loadMovie (MovieClip.loadMovie メソッド )`, `onPress (MovieClip.onPress ハンドラ )`, `MovieClipLoader`, `onClipEvent ハンドラ`, `on ハンドラ`, `loadMovieNum 関数`, `unloadMovie 関数`, `unloadMovieNum 関数`

loadVariables (MovieClip.loadVariables メソッド)

```
public loadVariables(url:String, [method:String]) : Void
```

外部ファイルからデータを読み取り、ムービークリップの変数の値を設定します。外部ファイルとして、Macromedia ColdFusion によって作成されたテキストファイル、CGI スクリプト、ASP (Active Server Page: アクティブサーバーページ)、PHP スクリプトのほか、適切に書式化されたテキストファイルを使用できます。ファイルには任意の数の変数を含むことができます。

また、`loadVariables()` メソッドを使用して、アクティブなムービークリップ内の変数の値を更新できます。

`loadVariables()` メソッドでは、URL のテキストが標準の MIME 形式 *application/x-www-form-urlencoded* (CGI スクリプト形式) である必要があります。

Flash Player 7 より前のバージョンの Player で SWF ファイルを実行している場合、`url` は、呼び出し元の SWF ファイルと同じスーパードメインに属している必要があります。スーパードメインは、ファイルの URL の一番左の要素を削除することによって派生されます。たとえば、www.someDomain.com に存在する SWF ファイルは、store.someDomain.com のデータソースからデータをロードできます。どちらのファイルも同じスーパードメイン someDomain.com に属しているからです。

Flash Player 7 以降で動作する SWF ファイルでは、`url` は、呼び出し元の SWF ファイルとまったく同じドメインに属している必要があります。たとえば www.someDomain.com に置かれている SWF ファイルは、www.someDomain.com に置かれているソースからのみデータをロードできます。異なるドメインからデータをロードするには、アクセス対象のデータソースをホストするサーバーにクロスドメインポリシーファイルを置く必要があります。

変数を特定のレベルにロードするには、`loadVariables()` の代わりに `loadVariablesNum()` を使用します。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 7 ではビヘイビアが変更されました。

パラメータ

`url:String` - ロードする変数がある外部ファイルの絶対 URL または相対 URL。呼び出し元の SWF ファイルが Web ブラウザで実行されている場合、`url` は SWF ファイルと同じドメインに属している必要があります。詳細については、次の説明を参照してください。

`method:String` (オプション) - 変数を送信するための HTTP メソッドを指定します。パラメータはストリング `GET` または `POST` でなければなりません。送信する変数がない場合は、このパラメータを省略します。`GET` メソッドは、変数を URL の最後に追加します。このメソッドは、変数のデータ量が少ないときに使用します。`POST` メソッドは、別の HTTP ヘッダで変数を送信します。このメソッドは、変数のデータ量が多いときに使用します。

例

次の例では、テキストファイル params.txt の情報を、createEmptyMovieClip() を使用して作成されたムービークリップ target_mc にロードします。setInterval() 関数は、ロードの進捗状況をチェックする場合に使用します。このスクリプトは、params.txt ファイル内の done という名前の変数をチェックします。

```
this.createEmptyMovieClip("target_mc", this.getNextHighestDepth());
target_mc.loadVariables("params.txt");
function checkParamsLoaded() {
  if (target_mc.done == undefined) {
  trace("not yet.");
  } else {
  trace("finished loading. killing interval.");
  trace("-------------");
  for (i in target_mc) {
    trace(i+": "+target_mc[i]);
  }
  trace("-------------");
  clearInterval(param_interval);
  }
}
var param_interval = setInterval(checkParamsLoaded, 100);
```

params.txt ファイルには、次のテキストが含まれています。

```
var1=hello&var2=goodbye&done=done
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

loadMovie (MovieClip.loadMovie メソッド), loadVariablesNum 関数, unloadMovie (MovieClip.unloadMovie メソッド)

localToGlobal (MovieClip.localToGlobal メソッド)

```
public localToGlobal(pt:Object) : Void
```

pt オブジェクトをムービークリップ内 (ローカル) の座標からステージ (グローバル) の座標に変換します。

MovieClip.localToGlobal() メソッドを使用すると、指定された *x* 座標と *y* 座標を、特定のムービークリップの左上隅を基準にした相対値から、ステージの左上隅を基準にした相対値に変換できます。

最初に、x と y の 2 つのプロパティを持つ汎用オブジェクトを作成する必要があります。これらの x と y の値は (これら x、y と呼ぶ必要があります) はムービークリップの左上隅を基準としているためローカル座標と呼ばれます。x プロパティは、ムービークリップの左上隅からの水平オフセットを表します。つまり、そのポイントが基準点からどれだけ右に配置されているかを示します。たとえば、x = 50 の場合、そのポイントは左上隅から 50 ピクセル右に配置されていることになります。y プロパティは、ムービークリップの左上隅からの垂直オフセットを表します。つまり、そのポイントが基準点からどれだけ下に配置されているかを示します。たとえば、y = 20 の場合、そのポイントは左上隅から 20 ピクセル下に配置されていることになります。次のコードでは、これらの座標を持つ汎用オブジェクトを作成します。

```
var myPoint:Object = new Object();
myPoint.x = 50;
myPoint.y = 20;
```

代わりに、オブジェクトを作成し、同時にリテラル Object 値を使用して値を割り当てることもできます。

```
var myPoint:Object = {x:50, y:20};
```

ローカル座標を使用して point オブジェクトを作成したら、その座標をグローバル座標に変換できます。localToGlobal() メソッドは、パラメータとして受け取る汎用オブジェクトの x 値と y 値を変更するため、値を返しません。このメソッドは、特定のムービークリップ (ローカル座標) を基準にした値から、ステージ (グローバル座標) を基準にした値に変更します。

たとえば、ムービークリップを作成してポイント (_x:100, _y:100) に配置し、ムービークリップの左上隅に近いポイント (x:10, y:10) を表すローカルポイントを localToGlobal() メソッドに渡す場合、このメソッドでは、その *x* 値と *y* 値をグローバル座標、この場合 (x:110, y:110) に変換する必要があります。この変換が行われるのは、*x* 座標と *y* 座標がこの時点で、ムービークリップの左上隅でなくステージの左上隅を基準にして表現されているためです。

ムービークリップの座標は、MovieClip の *x* 値と *y* 値を設定する MovieClip のプロパティである _x と _y を使用して表現されていました。ただし、汎用オブジェクトでは、アンダースコアなしの x と y を使用します。次のコードでは、*x* 座標と *y* 座標をグローバル座標に変換します。

```
var myPoint:Object = {x:10, y:10}; // create your generic point object
this.createEmptyMovieClip("myMovieClip", this.getNextHighestDepth());
myMovieClip._x = 100; // _x for movieclip x position
myMovieClip._y = 100; // _y for movieclip y position
```

```
myMovieClip.localToGlobal(myPoint);
trace ("x: " + myPoint.x); // 110
trace ("y: " + myPoint.y); // 110
```

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

`pt:Object` - *x* 座標と *y* 座標をプロパティとして指定し、Object クラスを使用して作成したオブジェクトの名前または識別子。

例

次の例では、オブジェクト `my_mc` の *x* 座標と *y* 座標を、ムービークリップ内の座標 (ローカル) からステージ座標 (グローバル) に変換します。インスタンスをクリックしてドラッグした後に、ムービークリップの中心点が反映されます。

```
this.createTextField("point_txt", this.getNextHighestDepth(), 0, 0, 100, 22);
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
 var point:Object = {x:my_mc._width/2, y:my_mc._height/2};
 my_mc.localToGlobal(point);
 point_txt.text = "x:"+point.x+", y:"+point.y;
};
Mouse.addListener(mouseListener);
my_mc.onPress = function() {
 this.startDrag();
};
my_mc.onRelease = function() {
 this.stopDrag();
};
```

この例で使用している `MovieClip.getNextHighestDepth()` メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、`MovieClip.getNextHighestDepth()` メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

`globalToLocal (MovieClip.globalToLocal メソッド )`

_lockroot (MovieClip._lockroot プロパティ)

```
public _lockroot : Boolean
```

SWF ファイルがムービークリップにロードされたときに、_root で参照する内容を指定するブール値。_lockroot プロパティのデフォルト値は undefined です。このプロパティは、ロードする SWF ファイルで設定することも、ムービークリップをロードするハンドラで設定することもできます。

たとえば、ユーザーがプレイするゲームを選択したら、そのゲーム (Chess.swf など) をムービークリップ game_mc にロードする、Games.fla という名前のドキュメントがあるとします。Games.swf にロードされた後も、Chess.swf で使用されている _root がすべて、Games.swf の _root ではなく、Games.swf の _root を参照する必要があります。Chess.fla にアクセスでき、それを Flash Player 7 以降にパブリッシュする場合、メインタイムラインでこのステートメントを Chess.fla に追加できます。

```
this._lockroot = true;
```

Chess.fla にアクセスできない場合 (たとえば、Chess.swf を他の人のサイトから chess_mc にロードしている場合)、ロード時に Chess.swf の _lockroot プロパティを設定できます。次の ActionScript を lockroot.fla のメインタイムラインに配置します。

```
chess_mc._lockroot = true;
```

この例では、Games.swf が Flash Player 7 以降用にパブリッシュされていれば、Chess.swf はどのバージョンの Flash Player 用にでもパブリッシュすることができます。

loadMovie() の呼び出しでは、次のコードで示すように、ロード先のムービーで MovieClip._lockroot プロパティを true に設定します。ロード先のムービー内で _lockroot を true に設定しない場合、ロードされたムービー内の _root への参照は、ロードされたムービーの _root ではなく、ロード先の _root をポイントします。

```
myMovieClip._lockroot = true;
```

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例では、"lockroot.fla" の _lockroot がメインの SWF ファイルに適用されています。SWF ファイルが別の FLA ドキュメントにロードされる場合、_root は常に "lockroot.swf" のスコープを参照します。これにより、コンフリクトを回避できます。次の ActionScript を lockroot.fla のメインタイムラインに配置します。

```
this._lockroot = true;
_root.myVar = 1;
_root.myOtherVar = 2;
trace("from lockroot.swf");
for (i in _root) {
  trace(" "+i+" -> "+_root[i]);
}
trace("");
```

次の情報が表示されます。

```
from lockroot.swf
myOtherVar -> 2
myVar -> 1
_lockroot -> true
$version -> WIN 7,0,19,0
```

次の例では、"lockroot.swf" および "nolockroot.swf" の 2 つの SWF ファイルをロードします。"lockroot.fla" ドキュメントには前の例の ActionScript が含まれています。nolockroot.fla ファイルでは、タイムラインのフレーム 1 に次のコードが追加されています。

```
_root.myVar = 1;
_root.myOtherVar = 2;
trace("from nolockroot.swf");
for (i in _root) {
  trace(" "+i+" -> "+_root[i]);
}
trace("");
```

"lockroot.swf" ファイルには `lockroot` が適用されていますが、"nolockroot.swf" ファイルには適用されていません。ファイルがロードされた後、各ファイルがそれぞれの `_root` スコープから変数を出力します。次の ActionScript を FLA ドキュメントのメインタイムフレームに配置します。

```
this.createEmptyMovieClip("lockroot_mc", this.getNextHighestDepth());
lockroot_mc.loadMovie("lockroot.swf");
this.createEmptyMovieClip("nolockroot_mc", this.getNextHighestDepth());
nolockroot_mc.loadMovie("nolockroot.swf");
function dumpRoot() {
  trace("from current SWF file");
  for (i in _root) {
  trace(" "+i+" -> "+_root[i]);
  }
  trace("");
}
dumpRoot();
```

次の情報が表示されます。

```
 from current SWF file
 dumpRoot -> [type Function]
 $version -> WIN 7,0,19,0
 nolockroot_mc -> _level0.nolockroot_mc
 lockroot_mc -> _level0.lockroot_mc
```

```
from nolockroot.swf
myVar -> 1
i -> lockroot_mc
dumpRoot -> [type Function]
$version -> WIN 7,0,19,0
nolockroot_mc -> _level0.nolockroot_mc
lockroot_mc -> _level0.lockroot_mc

from lockroot.swf
myOtherVar -> 2
myVar -> 1
```

_lockroot が適用されていないファイルには、ルート SWF 内の他の変数もすべて含まれます。"nolockroot.fla" にアクセスできない場合、メインタイムラインに追加した次の ActionScript を使用して、前のメインの FLA ドキュメントの _lockroot を変更できます。

```
this.createEmptyMovieClip("nolockroot_mc", this.getNextHighestDepth());
nolockroot_mc._lockroot = true;
nolockroot_mc.loadMovie("nolockroot.swf");
```

次のように表示されます。

```
from current SWF file
dumpRoot -> [type Function]
$version -> WIN 7,0,19,0
nolockroot_mc -> _level0.nolockroot_mc
lockroot_mc -> _level0.lockroot_mc

from nolockroot.swf
myOtherVar -> 2
myVar -> 1

from lockroot.swf
myOtherVar -> 2
myVar -> 1
```

関連項目

_root プロパティ, _lockroot (MovieClip._lockroot プロパティ), attachMovie (MovieClip.attachMovie メソッド), loadMovie (MovieClip.loadMovie メソッド), onLoadInit (MovieClipLoader.onLoadInit イベントリスナー)

menu (MovieClip.menu プロパティ)

public menu : ContextMenu

指定された ContextMenu オブジェクトをムービークリップに関連付けます。ContextMenu クラスを使用すると、ユーザーが Flash Player 内を右クリック (Windows) または Control キーを押しながらクリック (Macintosh) したときに表示されるコンテキストメニューを修正することができます。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例では、ContextMenu オブジェクト menu_cm をムービークリップ image_mc に割り当てます。ContextMenu オブジェクトには [View Image in Browser] というラベルのカスタムメニュー項目があり、関数 viewImage() が関連付けられています。

```
var menu_cm:ContextMenu = new ContextMenu();
menu_cm.customItems.push(new ContextMenuItem("View Image in Browser...",
  viewImage));
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  target_mc.menu = menu_cm;
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("photo1.jpg", image_mc);

function viewImage(target_mc:MovieClip, obj:Object) {
  getURL(target_mc._url, "_blank");
}
```

実行時にイメージを右クリック (Windows の場合) または Control キーを押したままクリック (Macintosh の場合) し、コンテキストメニューの [View Image in Browser] を選択すると、ブラウザウィンドウでイメージが開きます。

関連項目

menu (Button.menu プロパティ), ContextMenu, ContextMenuItem, menu (TextField.menu プロパティ)

moveTo (MovieClip.moveTo メソッド)

```
public moveTo(x:Number, y:Number) : Void
```

現在の描画位置を (x, y) に移動します。いずれかのパラメータを省略すると、このメソッドは失敗し、現在の描画位置は変更されません。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

x:Number - 親ムービークリップの基準点からの相対的な水平座標を示す整数。

y:Number - 親ムービークリップの基準点からの相対的な垂直座標を示す整数。

例

次の例では、5 ピクセルのマゼンタ色の実線と、部分的に透明な青の塗りを使用して三角形を描画します。

```
this.createEmptyMovieClip("triangle_mc", 1);
triangle_mc.beginFill(0x0000FF, 30);
triangle_mc.lineStyle(5, 0xFF00FF, 100);
triangle_mc.moveTo(200, 200);
triangle_mc.lineTo(300, 300);
triangle_mc.lineTo(100, 300);
triangle_mc.lineTo(200, 200);
triangle_mc.endFill();
```

関連項目

createEmptyMovieClip (MovieClip.createEmptyMovieClip メソッド),lineStyle (MovieClip.lineStyle メソッド),lineTo (MovieClip.lineTo メソッド)

_name (MovieClip._name プロパティ)

```
public _name : String
```

ムービークリップのインスタンス名

対応バージョン : ActionScript 1.0、Flash Player 4

例

次の例では、ステージ上のムービークリップを右クリック (Windows の場合) または Control キーを押しながらクリック (Macintosh の場合) し、コンテキストメニューの [Info] を選択して、そのインスタンスに関する情報を表示できます。いくつかのムービークリップをインスタンス名を付けて追加し、次の ActionScript を AS ファイルまたは FLA ファイルに追加します。

```
var menu_cm:ContextMenu = new ContextMenu();
menu_cm.customItems.push(new ContextMenuItem("Info...", getMCInfo));
function getMCInfo(target_mc:MovieClip, obj:Object) {
  trace("You clicked on the movie clip '"+target_mc._name+"'.");
  trace("\t width:"+target_mc._width+", height:"+target_mc._height);
  trace("");
}
for (var i in this) {
  if (typeof (this[i]) == 'movieclip') {
  this[i].menu = menu_cm;
  }
}
```

関連項目

_name (Button._name プロパティ)

nextFrame (MovieClip.nextFrame メソッド)

```
public nextFrame() : Void
```

次のフレームに再生ヘッドを送り、停止します。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、_framesloaded と nextFrame() を使用して、SWF ファイルにコンテンツをロードします。タイムラインのフレーム 1 にはコードを追加せず、フレーム 2 に次の ActionScript を追加します。

```
if (this._framesloaded >= 3) {
  this.nextFrame();
} else {
  this.gotoAndPlay(1);
}
```

次に、次のコード (およびロードする必要のあるコンテンツ) をフレーム 3 に追加します。

```
stop();
```

関連項目

nextFrame 関数, prevFrame 関数, prevFrame (MovieClip.prevFrame メソッド)

onData (MovieClip.onData ハンドラ)

```
onData = function() {}
```

ムービークリップが MovieClip.loadVariables() の呼び出しからデータを受け取ったときに呼び出されます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。関数はタイムラインに定義できます。また、MovieClip クラスを拡張するクラスファイル内、またはライブラリ内のシンボルにリンクされるクラスファイル内に定義できます。

このハンドラは、MovieClip.loadVariables() メソッドまたは loadVariables() グローバル関数でのみ使用できます。MovieClip.loadMovie() メソッドまたは loadMovie() 関数でイベントハンドラを呼び出す場合は、このハンドラの代わりに onClipEvent(*data*) を使用します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例は、MovieClip.onData() の正しい使用方法を示しています。FLA と同じディレクトリからファイル *OnData.txt* をロードします。ファイルのデータが MovieClip オブジェクトにロードされ、onData() が実行されたら、データをトレースします。

```
var mc:MovieClip = this.createEmptyMovieClip("my_mc",
  this.getNextHighestDepth());

mc.onData = function() {
  for(var i in this) {
    trace(">> " + i + ": " + this[i]);
  }
}

mc.loadVariables("OnData.txt");
```

関連項目

onClipEvent ハンドラ , loadVariables (MovieClip.loadVariables メソッド)

onDragOut (MovieClip.onDragOut ハンドラ)

```
onDragOut = function() {}
```

マウスボタンが押された状態で、ポインタがオブジェクトの外に移動したときに呼び出されます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。関数はタイムラインに定義できます。また、MovieClip クラスを拡張するクラスファイル内、またはライブラリ内のシンボルにリンクされるクラスファイル内に定義できます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、onDragOut イベントハンドラの関数を定義します。この関数は、[出力] パネルに trace() アクションを送ります。

```
my_mc.onDragOut = function () {
  trace("onDragOut called");
}
```

関連項目

onDragOver (MovieClip.onDragOver ハンドラ)

onDragOver (MovieClip.onDragOver ハンドラ)

```
onDragOver = function() {}
```

ポインタがムービークリップ外にドラッグされた後で、ムービークリップ上に移動したときに呼び出されます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。関数はタイムラインに定義できます。また、MovieClip クラスを拡張するクラスファイル内、またはライブラリ内のシンボルにリンクされるクラスファイル内に定義できます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、onDragOver イベントハンドラの関数を定義します。この関数は、[出力] パネルに trace() アクションを送ります。

```
my_mc.onDragOver = function () {
  trace("onDragOver called");
}
```

関連項目

onDragOut (MovieClip.onDragOut ハンドラ)

onEnterFrame (MovieClip.onEnterFrame ハンドラ)

```
onEnterFrame = function() {}
```

SWF ファイルのフレームレートで繰り返し呼び出されます。onEnterFrame イベントハンドラに割り当てる関数は、影響を受けるフレームに追加されている他のすべての ActionScript コードの前に処理されます。

このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。関数はタイムラインに定義できます。また、MovieClip クラスを拡張するクラスファイル内、またはライブラリ内のシンボルにリンクされるクラスファイル内に定義できます。

定義済み関数を Flash Player が呼び出すのを停止させる条件が揃ったら、onEnterFrame イベントハンドラの値を null に設定します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、onEnterFrame イベントハンドラの関数を定義します。この関数は、[出力] パネルに trace() アクションを送ります。

```
my_mc.onEnterFrame = function () {
  trace("onEnterFrame called");
}
```

onKeyDown (MovieClip.onKeyDown ハンドラ)

```
onKeyDown = function() {}
```

ムービークリップにフォーカスがあるときにキーを押すと、呼び出されます。onKeyDown イベントハンドラにはパラメータは渡されません。Key.getAscii() メソッドと Key.getCode() メソッドを使用して、どのキーが押されたかを調べることができます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。関数はタイムラインに定義できます。また、MovieClip クラスを拡張するクラスファイル内、またはライブラリ内のシンボルにリンクされるクラスファイル内に定義できます。

onKeyDown イベントハンドラは、ムービークリップの入力フォーカスが有効で、設定されている場合にのみ動作します。まず、ムービークリップの MovieClip.focusEnabled プロパティを true に設定します。次に、ムービークリップにフォーカスを与えます。そのためには、Selection.setFocus() を使用するか、Tab キーでムービークリップに移動するように設定します。

Selection.setFocus() を使用する場合は、ムービークリップのパスを Selection.setFocus() に渡す必要があります。フォーカスは、ユーザーがマウスを動かすことによって他のエレメントに簡単に移動します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、onKeyDown() イベントハンドラの関数を定義します。この関数は、[出力] パネルに trace() アクションを送ります。ムービークリップ my_mc を作成し、次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
my_mc.onKeyDown = function () {
  trace("key was pressed");
}
```

onKeyDown イベントハンドラを実行するには、ムービークリップにフォーカスがあることが必要です。次の ActionScript を追加して入力フォーカスを設定します。

```
my_mc.tabEnabled = true;
my_mc.focusEnabled = true;
Selection.setFocus(my_mc);
```

ユーザーがキーを押すと、[出力] パネルに key was pressed と表示されます。しかし、マウスを動かした後は、ムービークリップからフォーカスが移動されてしまうため、key was pressed は表示されません。このため、ほとんどの場合は Key.onKeyDown を使用する必要があります。

関連項目

getAscii (Key.getAscii メソッド), getCode (Key.getCode メソッド), onKeyDown (Key.onKeyDown イベントリスナー), focusEnabled (MovieClip.focusEnabled プロパティ), onKeyUp (MovieClip.onKeyUp ハンドラ), setFocus (Selection.setFocus メソッド)

onKeyUp (MovieClip.onKeyUp ハンドラ)

```
onKeyUp = function() {}
```

キーを離すと呼び出されます。onKeyUp イベントハンドラにはパラメータは渡されません。Key.getAscii() メソッドと Key.getCode() メソッドを使用して、どのキーが押されたかを調べることができます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。関数はタイムラインに定義できます。また、MovieClip クラスを拡張するクラスファイル内、またはライブラリ内のシンボルにリンクされるクラスファイル内に定義できます。

onKeyUp イベントハンドラは、ムービークリップの入力フォーカスが有効で、設定されている場合にのみ動作します。まず、ムービークリップの MovieClip.focusEnabled プロパティを true に設定します。次に、ムービークリップにフォーカスを与えます。そのためには、Selection.setFocus() を使用するか、Tab キーでムービークリップに移動するように設定します。

Selection.setFocus() を使用する場合は、ムービークリップのパスを Selection.setFocus() に渡す必要があります。フォーカスは、ユーザーがマウスを動かすことによって他のエレメントに簡単に移動します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、onKeyUp イベントハンドラの関数を定義します。この関数は、[出力] パネルに trace() アクションを送ります。

```
my_mc.onKeyUp = function () {
  trace("onKey called");
}
```

次の例では、入力フォーカスを設定します。

```
my_mc.focusEnabled = true;
Selection.setFocus(my_mc);
```

関連項目

getAscii (Key.getAscii メソッド), getCode (Key.getCode メソッド), onKeyDown (Key.onKeyDown イベントリスナー), focusEnabled (MovieClip.focusEnabled プロパティ), onKeyDown (MovieClip.onKeyDown ハンドラ), setFocus (Selection.setFocus メソッド)

onKillFocus (MovieClip.onKillFocus ハンドラ)

```
onKillFocus = function(newFocus:Object) {}
```

ムービークリップがフォーカスを失うと、呼び出されます。onKillFocus メソッドは、1 つのパラメータ newFocus を受け取ります。このパラメータは、フォーカスを受け取る新しいオブジェクトを表すオブジェクトです。フォーカスを受け取るオブジェクトがない場合、newFocus の値は null です。

このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。関数はタイムラインに定義できます。また、MovieClip クラスを拡張するクラスファイル内、またはライブラリ内のシンボルにリンクされるクラスファイル内に定義できます。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

newFocus:Object - キーボードフォーカスを受け取るオブジェクト。

例

次の例では、フォーカスを失うムービークリップに関する情報と現在フォーカスがあるインスタンスを表示します。ステージ上には、my_mc と other_mc の 2 つのムービークリップがあります。次の ActionScript を AS ドキュメントまたは FLA ドキュメントに追加します。

```
my_mc.onRelease = Void;
other_mc.onRelease = Void;
my_mc.onKillFocus = function(newFocus) {
  trace("onKillFocus called, new focus is: "+newFocus);
};
```

2 つのインスタンス間をタブ移動すると、情報が [出力] パネルに表示されます。

関連項目

onSetFocus (MovieClip.onSetFocus ハンドラ)

onLoad (MovieClip.onLoad ハンドラ)

```
onLoad = function() {}
```

ムービークリップのインスタンスが作成されてタイムラインに読み込まれると、呼び出されます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。関数はタイムラインに定義できます。また、MovieClip クラスを拡張するクラスファイル内、またはライブラリ内のシンボルにリンクされるクラスファイル内に定義できます。

このハンドラは、クラスに関連付けられたシンボルがライブラリに存在するムービークリップでのみ使用します。たとえば、`MovieClip.loadMovie()` を使用して SWF ファイルを動的にロードする場合など、特定のムービークリップがロードされたときにイベントハンドラを呼び出す場合は、このハンドラの代わりに、`onClipEvent(load)` または MovieClipLoader クラスを使用します。他のハンドラは、`MovieClip.onLoad` とは異なり、どのムービークリップがロードされたときにも呼び出されます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

この例は、MovieClip クラスを拡張する ActionScript 2.0 クラス定義における `onLoad` イベントハンドラの使用方法を示しています。まず、Oval.as という名前のクラスファイルを作成し、`onLoad()` という名前のクラスメソッドを定義します。次の例のように、クラスファイルは適切なクラスファイルに配置します。

```
// contents of Oval.as
class Oval extends MovieClip{
  public function onLoad () {
    trace("onLoad called");
  }
}
```

次に、ライブラリ内にムービークリップシンボルを作成し、Oval という名前を付けます。[ライブラリ] パネルでシンボルをコンテキストクリック (通常右クリック) し、ポップアップメニューから [リンケージ] を選択します。[ActionScript に書き出し] をクリックし、[識別子] フィールドと [ActionScript 2.0 クラス] フィールドに `Oval` と入力します。[ActionScript に書き出し] をオンのままにして、[OK] をクリックします。

次に、ファイルの最初のフレームに移動して、[アクション] パネルで次のコードを入力します。

```
var myOval:Oval = Oval(attachMovie("Oval","Oval_1",1));
```

最後に、ムービープレビューを実行します。"onLoad called" という出力テキストが表示されます。

関連項目

loadMovie (MovieClip.loadMovie メソッド),onClipEvent ハンドラ ,MovieClipLoader

onMouseDown (MovieClip.onMouseDown ハンドラ)

```
onMouseDown = function() {}
```

マウスボタンが押されると、呼び出されます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。関数はタイムラインに定義できます。また、MovieClip クラスを拡張するクラスファイル内、またはライブラリ内のシンボルにリンクされるクラスファイル内に定義できます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、onMouseDown() イベントハンドラの関数を定義します。この関数は、[出力] パネルに trace() アクションを送ります。

```
my_mc.onMouseDown = function () {
  trace("onMouseDown called");
}
```

onMouseMove (MovieClip.onMouseMove ハンドラ)

```
onMouseMove = function() {}
```

マウスポインタが移動すると、呼び出されます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。関数はタイムラインに定義できます。また、MovieClip クラスを拡張するクラスファイル内、またはライブラリ内のシンボルにリンクされるクラスファイル内に定義できます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、onMouseMove() イベントハンドラの関数を定義します。この関数は、[出力] パネルに trace() アクションを送ります。

```
my_mc.onMouseMove = function () {
  trace("onMouseMove called");
}
```

onMouseUp (MovieClip.onMouseUp ハンドラ)

```
onMouseUp = function() {}
```

マウスボタンが離されると、呼び出されます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。関数はタイムラインに定義できます。また、MovieClip クラスを拡張するクラスファイル内、またはライブラリ内のシンボルにリンクされるクラスファイル内に定義できます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、onMouseUp() イベントハンドラの関数を定義します。この関数は、[出力] パネルに trace() アクションを送ります。

```
my_mc.onMouseUp = function () {
  trace("onMouseUp called");
}
```

onPress (MovieClip.onPress ハンドラ)

```
onPress = function() {}
```

ポインタがムービークリップ上にあるときにマウスをクリックすると、呼び出されます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。関数はライブラリに定義できます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、onPress() イベントハンドラの関数を定義します。この関数は、[出力] パネルに trace() アクションを送ります。

```
my_mc.onPress = function () {
  trace("onPress called");
}
```

onRelease (MovieClip.onRelease ハンドラ)

```
onRelease = function() {}
```

ムービークリップの上でマウスボタンを離すと、呼び出されます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。関数はタイムラインに定義できます。また、MovieClip クラスを拡張するクラスファイル内、またはライブラリ内のシンボルにリンクされるクラスファイル内に定義できます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、onRelease( ) イベントハンドラの関数を定義します。この関数は、[出力] パネルに trace( ) アクションを送ります。

```
my_mc.onRelease = function () {
  trace("onRelease called");
}
```

onReleaseOutside (MovieClip.onReleaseOutside ハンドラ)

```
onReleaseOutside = function() {}
```

マウスボタンをムービークリップ領域内で押して、ムービークリップ領域の外側で離したときに呼び出されます。

このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。関数はタイムラインに定義できます。また、MovieClip クラスを拡張するクラスファイル内、またはライブラリ内のシンボルにリンクされるクラスファイル内に定義できます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、onReleaseOutside( ) イベントハンドラの関数を定義します。この関数は、[出力] パネルに trace( ) アクションを送ります。

```
my_mc.onReleaseOutside = function () {
  trace("onReleaseOutside called");
}
```

onRollOut (MovieClip.onRollOut ハンドラ)

```
onRollOut = function() {}
```

ポインタをムービークリップ領域の外側に移動すると、呼び出されます。

このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。関数はタイムラインに定義できます。また、MovieClip クラスを拡張するクラスファイル内、またはライブラリ内のシンボルにリンクされるクラスファイル内に定義できます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、onRollOut() イベントハンドラの関数を定義します。この関数は、[出力] パネルに trace() アクションを送ります。

```
my_mc.onRollOut = function () {
  trace("onRollOut called");
}
```

onRollOver (MovieClip.onRollOver ハンドラ)

```
onRollOver = function() {}
```

ポインタをムービークリップ領域の上に移動すると、呼び出されます。

このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。関数はタイムラインに定義できます。また、MovieClip クラスを拡張するクラスファイル内、またはライブラリ内のシンボルにリンクされるクラスファイル内に定義できます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、onRollOver() イベントハンドラの関数を定義します。この関数は、[出力] パネルに trace() アクションを送ります。

```
my_mc.onRollOver = function () {
  trace("onRollOver called");
}
```

onSetFocus (MovieClip.onSetFocus ハンドラ)

```
onSetFocus = function(oldFocus:Object) {}
```

ムービークリップがキーボードフォーカスを受け取ると、呼び出されます。oldFocus パラメータは、フォーカスを失うオブジェクトです。たとえば、ユーザーが Tab キーを押して入力フォーカスをムービークリップからテキストフィールドに移動すると、oldFocus にムービークリップインスタンスが入ります。

前にフォーカスがあったオブジェクトが存在しない場合、oldFocus には null 値が入ります。

このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。関数はタイムラインで定義できます。また、MovieClip クラスを拡張するクラスファイル内、またはライブラリ内のシンボルにリンクされるクラスファイル内に定義できます。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

`oldFocus:Object` - キーボードフォーカスを失うオブジェクト。

例

次の例では、キーボードフォーカスを受け取るムービークリップに関する情報と、前にフォーカスがあったインスタンスが表示されます。ステージ上には、my_mc と other_mc の 2 つのムービークリップがあります。次の ActionScript を AS ドキュメントまたは FLA ドキュメントに追加します。

```
my_mc.onRelease = Void;
other_mc.onRelease = Void;
my_mc.onSetFocus = function(oldFocus) {
  trace("onSetFocus called, previous focus was: "+oldFocus);
}
```

2 つのインスタンス間をタブ移動すると、情報が [出力] パネルに表示されます。

関連項目

onKillFocus (MovieClip.onKillFocus ハンドラ)

onUnload (MovieClip.onUnload ハンドラ)

```
onUnload = function() {}
```

ムービークリップをタイムラインから削除した後に表示される最初のフレームで呼び出されます。onUnload イベントハンドラに関連付けられているアクションは、影響を受けるフレームにアクションが割り当てられる前に処理されます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。関数はタイムラインに定義できます。また、MovieClip クラスを拡張するクラスファイル内、またはライブラリ内のシンボルにリンクされるクラスファイル内に定義できます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、MovieClip.onUnload() イベントハンドラの関数を定義します。この関数は、[出力] パネルに trace() アクションを送ります。

```
my_mc.onUnload = function () {
  trace("onUnload called");
}
```

opaqueBackground (MovieClip.opaqueBackground プロパティ)

```
public opaqueBackground : Number
```

数値 (16 進数の RGB 値) で指定されたムービークリップの不透明 (透明でない) な背景色。値が null または undefined の場合、不透明な背景は存在しません。cacheAsBitmap プロパティが true に設定されているムービークリップでは、opaqueBackground を設定すると、レンダリングのパフォーマンスが向上します。

opaqueBackground を未設定にしたために透明な領域が多数あるムービークリップでは、パフォーマンス上の大きなメリットを認識できます。

メモ : shapeFlag パラメータが true に設定されている hitTest() メソッドでは、不透明な背景領域は照合されません。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、三角形のアウトラインを作成して opaqueBackground プロパティを特定の色に設定します。

```
var triangle:MovieClip = this.createEmptyMovieClip("triangle",
  this.getNextHighestDepth());
triangle._x = triangle._y = 50;
triangle.lineStyle(3, 0xFFCC00);
triangle.lineTo(0, 30);
```

```
triangle.lineTo(50, 0);
triangle.lineTo(0, 0);
triangle.endFill();
triangle.opaqueBackground = 0xCCCCCC;
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

関連項目

cacheAsBitmap (MovieClip.cacheAsBitmap プロパティ), hitTest (MovieClip.hitTest メソッド)

_parent (MovieClip._parent プロパティ)

```
public _parent : MovieClip
```

現在のムービークリップまたはオブジェクトを含むムービークリップまたはオブジェクトを指す参照です。現在のオブジェクトは、_parent プロパティを参照するオブジェクトです。現在のムービークリップまたはオブジェクトの上位にあるムービークリップまたはオブジェクトへの相対パスを指定するには、_parent プロパティを使用します。

次のコードのように、_parent を使用して表示リストの複数上のレベルに移動できます。

```
this._parent._parent._alpha = 20;
```

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、ムービークリップとその親のタイムラインへの参照をトレースします。my_mc というインスタンス名でムービークリップを作成し、それをメインタイムラインに追加します。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
my_mc.onRelease = function() {
  trace("You clicked the movie clip: "+this);
  trace("The parent of "+this._name+" is: "+this._parent);
}
```

ムービークリップをクリックすると、[出力] パネルに次の情報が表示されます。

```
You clicked the movie clip: _level0.my_mc
The parent of my_mc is: _level0
```

関連項目

_parent (Button._parent プロパティ), _root プロパティ , targetPath 関数 , _parent (TextField._parent プロパティ)

play (MovieClip.play メソッド)

```
public play() : Void
```

ムービークリップのタイムラインで再生ヘッドを移動します。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン： ActionScript 1.0、Flash Player 5

例

次の ActionScript を使用して SWF ファイルのメインタイムラインを再生します。これは、メインタイムライン上のムービークリップボタン my_mc 用の ActionScript です。

```
stop();
my_mc.onRelease = function() {
  this._parent.play();
};
```

次の ActionScript を使用して、SWF ファイル内のムービークリップのタイムラインを再生します。これは、ムービークリップ animation_mc を再生する、メインタイムライン上のボタン my_btn の ActionScript です。

```
animation_mc.stop();
my_btn.onRelease = function(){
  animation_mc.play();
};
```

関連項目

play 関数, gotoAndPlay (MovieClip.gotoAndPlay メソッド), gotoAndPlay 関数

prevFrame (MovieClip.prevFrame メソッド)

```
public prevFrame() : Void
```

直前のフレームに再生ヘッドを戻し、停止します。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン： ActionScript 1.0、Flash Player 5

例

次の例では、2 つのムービークリップボタンがタイムラインを制御します。prev_mc ボタンは再生ヘッドを前のフレームに移動し、next_mc ボタンは再生ヘッドを次のフレームに移動します。タイムライン上の一連のフレームにコンテンツを追加し、次の ActionScript をタイムラインのフレーム 1 に追加します。

```
stop();
prev_mc.onRelease = function() {
   var parent_mc:MovieClip = this._parent;
   if (parent_mc._currentframe>1) {
   parent_mc.prevFrame();
   } else {
   parent_mc.gotoAndStop(parent_mc._totalframes);
   }
};
next_mc.onRelease = function() {
   var parent_mc:MovieClip = this._parent;
   if (parent_mc._currentframe<parent_mc._totalframes) {
   parent_mc.nextFrame();
   } else {
   parent_mc.gotoAndStop(1);
   }
};
```

関連項目

prevFrame 関数

_quality (MovieClip._quality プロパティ)

```
public _quality : String
```

SWF ファイルに使用するレンダリング品質を設定または取得します。デバイスフォントは常にエイリアス処理されるため、_quality プロパティには影響されません。

_quality は、次の値に設定できます。

値	説明	グラフィックのアンチエイリアス	ビットマップスムージング
"LOW"	低いレンダリング品質。	グラフィックスはアンチエイリアス処理されていません。	ビットマップはスムージングされません。

値	説明	グラフィックのアンチエイリアス	ビットマップスムージング
"MEDIUM"	普通のレンダリング品質。この設定は、テキストを含まないムービーに適しています。	グラフィックスは2x2ピクセルグリッドを使用してアンチエイリアス処理されます。	Flash Player 8 では、ビットマップは、MovieClip.attachBitmap() 呼び出しおよび MovieClip.beginBitmapFill() 呼び出しで使用される smoothing パラメータに基づいてスムージングされます。 Flash Player 6 および 7 では、ビットマップはスムージングされません。
"HIGH"	高いレンダリング品質。デフォルトのレンダリング品質設定です。	グラフィックスは4x4ピクセルグリッドを使用してアンチエイリアス処理されます。	Flash Player 8 では、ビットマップは、MovieClip.attachBitmap() 呼び出しおよび MovieClip.beginBitmapFill() 呼び出しで使用される smoothing パラメータに基づいてスムージングされます。 Flash Player 6 および 7 では、ムービークリップが静的なものである場合、ビットマップはスムージングされます。
"BEST"	非常に高いレンダリング品質。	グラフィックスは4x4ピクセルグリッドを使用してアンチエイリアス処理されます。	Flash Player 8 では、ビットマップは、MovieClip.attachBitmap() 呼び出しおよび MovieClip.beginBitmapFill() 呼び出しで使用される smoothing パラメータに基づいてスムージングされます。smoothing を設定すると、平均化アルゴリズムにより、ムービークリップを縮小したときのレンダリング品質が向上します。レンダリング処理に時間はかかりますが、この設定は大きなイメージの高品質のサムネールを表示する場合などに利用できます。 Flash Player 6 および 7 では、ビットマップは常にスムージングされます。

メモ : このプロパティを MovieClip インスタンスに対して指定することもできますが、実際にはグローバルプロパティであるので、単に _quality という形で値を指定することもできます。MovieClip インスタンスに _quality を設定すると、その品質が SWF 全体に設定されます。

対応バージョン： ActionScript 1.0、Flash Player 6

例

この例では、ムービークリップ my_mc (さらにはこのムービークリップが含まれる SWF 全体) のレンダリング品質を、LOW に設定します。

```
my_mc._quality = "LOW";
```

関連項目

_quality プロパティ

removeMovieClip (MovieClip.removeMovieClip メソッド)

```
public removeMovieClip() : Void
```

duplicateMovieClip()、MovieClip.duplicateMovieClip()、MovieClip.createEmptyMovieClip() または MovieClip.attachMovie() で作成したムービークリップインスタンスを削除します。

このメソッドは、負の深度値に割り当てられているムービークリップを削除しません。オーサリングツールで作成したムービークリップには、デフォルトで負の深度値が割り当てられます。負の深度値に割り当てられているムービークリップを削除するには、最初に MovieClip.swapDepths() を使用して、ムービークリップを正の深度値に移動します。

メモ : バージョン 2 のコンポーネントを使用している場合は、このメソッドを使用しないでください。ステージまたはライブラリにバージョン 2 のコンポーネントがあると、getNextHighestDepth() メソッドの戻り値が 1048676 になることがあります。この深度は、有効な値の範囲外です。バージョン 2 のコンポーネントを使用している場合は、バージョン 2 のコンポーネントの DepthManager クラスを必ず使用してください。

メモ : バージョン 2 のコンポーネントを使用している場合、バージョン 2 のコンポーネントの DepthManager クラスでなく MovieClip.getNextHighestDepth() を使用して深度値を割り当てると、removeMovieClip() はエラー通知なしで失敗します。何らかの バージョン 2 コンポーネントを使用している場合、DepthManager クラスでは、カーソルとツールヒントに対して使用可能な最高 (1048575) および最低 (-16383) の深度を自動的に予約します。その後 getNextHighestDepth() を呼び出すと、有効範囲外にある 1048576 が返されます。removeMovieClip() メソッドは、有効範囲外にある深度値を見つけると、エラー通知なしで失敗します。getNextHighestDepth() をバージョン 2 のコンポーネントと併用する必要がある場合、有効な深度値を割り当てるには swapDepths()、ムービークリップのコンテンツを削除するには MovieClip.unloadMovie() を使用できます。代わりに、DepthManager クラスを使用して、有効範囲内の深度値を割り当てることができます。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン： ActionScript 1.0、Flash Player 5

例

次の例では、ボタンをクリックするたびに、ムービークリップインスタンスをステージに random 位置で割り当てます。ムービークリップインスタンスをクリックすると、そのインスタンスが SWF ファイルから削除されます。

```
function randRange(min:Number, max:Number):Number {
  var randNum:Number = Math.round(Math.random()*(max-min))+min;
  return randNum;
}
var bugNum:Number = 0;
addBug_btn.onRelease = addBug;
function addBug() {
  var thisBug:MovieClip = this._parent.attachMovie("bug_id", "bug"+bugNum+"_mc",
  bugNum,
{_x:randRange(50, 500), _y:randRange(50, 350)});
  thisBug.onRelease = function() {
  this.removeMovieClip();
  };
  bugNum++;
}
```

関連項目

duplicateMovieClip 関数, createEmptyMovieClip (MovieClip.createEmptyMovieClip メソッド), duplicateMovieClip (MovieClip.duplicateMovieClip メソッド), attachMovie (MovieClip.attachMovie メソッド), swapDepths (MovieClip.swapDepths メソッド)

_rotation (MovieClip._rotation プロパティ)

```
public _rotation : Number
```

ムービークリップの元の位置からの回転角度を度数で指定します。時計回りに回転させる場合は 0 ～ 180 の値を指定します。反時計回りに回転させる場合は 0 ～ -180 の値を指定します。この範囲を超える値は、360 に加算または 360 から減算され、範囲内に収まる値が取得されます。たとえば、`my_mc._rotation = 450` というステートメントは `my_mc._rotation = 90` と同義です。+/- 720 よりも大きい値には、360 の倍数が使用されます。

対応バージョン： ActionScript 1.0、Flash Player 4

例

次の例では、ムービークリップインスタンス triangle を動的に作成します。SWF ファイルを実行し、ムービークリップをクリックすると、ムービークリップが回転します。

```
this.createEmptyMovieClip("triangle", this.getNextHighestDepth());

triangle.beginFill(0x0000FF, 100);
triangle.moveTo(100, 100);
triangle.lineTo(100, 150);
triangle.lineTo(150, 100);
triangle.lineTo(100, 100);

triangle.onMouseUp= function() {
  this._rotation += 15;
};
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

_rotation (Button._rotation プロパティ), _rotation (TextField._rotation プロパティ)

scale9Grid (MovieClip.scale9Grid プロパティ)

public scale9Grid : Rectangle

ムービークリップの 9 つの拡大・縮小領域を定義する矩形の領域。null に設定すると、任意の拡大・縮小変形を適用した際、ムービークリップ全体が通常どおり拡大・縮小します。

ムービークリップの scale9Grid プロパティを定義すると、グリッドの中央領域を定義する scale9Grid 矩形に基づき、ムービークリップが 9 つの領域を持つグリッドに分割されます。グリッドには、他に次の 8 つの領域があります。

- 矩形の外側にある左上隅の領域
- 矩形の上側の領域
- 矩形の外側にある右上隅の領域
- 矩形の左側の領域
- 矩形の右側の領域
- 矩形の外側にある左下隅の領域
- 矩形の下側にある領域
- 矩形の外側にある右下隅の領域

矩形で定義される中心以外の 8 つの領域は、ムービークリップの拡大・縮小時に特別な規則が適用される額縁のようなものと考えることができます。

scale9Grid プロパティが設定されている場合にムービークリップを拡大・縮小すると、すべてのテキストと子のムービークリップは、配置されている scale9Grid の領域に関係なく、通常どおり拡大・縮小します。ただし、他の種類のオブジェクトでは、次の規則が適用されます。

- 中心領域のすべてのコンテンツは、通常どおり拡大・縮小します。
- 隅のコンテンツが拡大・縮小されるのは、中心領域が 0 に縮小されたときだけです。
- 上下の領域のコンテンツは、水平方向にのみ拡大・縮小されます。左右の領域のコンテンツは、垂直方向にのみ拡大・縮小されます。
- すべての塗り (ビットマップ、ビデオ、グラデーションを含む) は、形状に収まるように伸縮されます。

ムービークリップを回転すると、その後の拡大・縮小は通常どおりになります。また、scale9Grid プロパティは無視されます。

たとえば、次のムービークリップと、そのムービークリップの scale9Grid プロパティとして適用される矩形があるとします。

 ムービークリップ。	 赤の矩形は scale9Grid を表します。

ムービークリップを拡大・縮小または伸縮すると、矩形内のオブジェクトは通常どおり拡大・縮小しますが、矩形外のオブジェクトは scale9Grid の規則に従って拡大・縮小します。

75% に縮小 :	
50% に縮小 :	
25% に縮小 :	

水平方向に 150% 伸長 :	

通常、scale9Grid は、コンポーネントの拡大・縮小時にエッジの線を同じ幅に保つようコンポーネントを設定するときに使用します。

Adobe Flash のオーサリング環境では、ライブラリ内にあるムービークリップシンボルの 9 つのスライスの拡大・縮小のガイドを有効にできます。このガイドにより、オブジェクトの scale9grid をグラフィックで確認しながら決定できます。シンボルの 9 つのスライスの拡大・縮小を設定すると、そのシンボルのすべてのインスタンスの scale9grid プロパティが自動的に設定されます。9 つのスライスの拡大・縮小が有効なシンボルでは、SWF ファイルを作成するとき、9 つのスライスの拡大・縮小グリッドの複数の領域にまたがる曲線は、領域ごとに別の曲線に分割されます。たとえば、9 つのスライスの拡大・縮小が有効なムービークリップシンボルにある曲線と、9 つのスライスの拡大・縮小が有効でないムービークリップシンボルにある同じ曲線とを比較してみます。

シンボルの 9 つのスライスの拡大・縮小が有効になっている場合	
シンボルの 9 つのスライスの拡大・縮小が有効になっていない場合	

Flash で SWF ファイルが作成されるとき、図の最初のムービークリップの曲線は 3 つの曲線に分割されます。これは、9 つのスライスの拡大・縮小が有効になっていない 2 番目のムービークリップには該当しません。2 番目のムービークリップの scale9Grid を、最初のムービークリップの scale9Grid と一致する矩形に設定しても、Flash が最初のムービークリップの曲線を分割する方法により、2 つのムービークリップを拡大・縮小した結果は異なります。

9 つのスライスの拡大・縮小が有効になっているシンボルを 150% に拡大した場合	
9 つのスライスの拡大・縮小が有効になっていないシンボルを 150% に拡大した場合	

対応バージョン : ActionScript 2.0、Flash Player 8

例

次の例では、20 ピクセルの線 (境界を形成) とグラデーションの塗りを含むムービークリップを作成します。ムービークリップは、マウスの位置に基づいて拡大・縮小します。またムービークリップで設定されている scale9Grid により、20 ピクセルの線の太さは、ムービークリップの拡大・縮小時に変化しません。ただし、ムービークリップ内のグラデーションは拡大・縮小します。

```
import flash.geom.Rectangle;
import flash.geom.Matrix;

this.createEmptyMovieClip("my_mc", this.getNextHighestDepth());

var grid:Rectangle = new Rectangle(20, 20, 260, 260);
my_mc.scale9Grid = grid ;

my_mc._x = 50;
my_mc._y = 50;

function onMouseMove()
{
  my_mc._width = _xmouse;
  my_mc._height = _ymouse;
}

my_mc.lineStyle(20, 0xff3333, 100);
var gradient_matrix:Matrix = new Matrix();
gradient_matrix.createGradientBox(15, 15, Math.PI, 10, 10);
my_mc.beginGradientFill("radial", [0xffff00, 0x0000ff],
       [100, 100], [0, 0xFF], gradient_matrix,
       "reflect", "RGB", 0.9);
my_mc.moveTo(0, 0);
my_mc.lineTo(0, 300);
my_mc.lineTo(300, 300);
my_mc.lineTo(300, 0);
my_mc.lineTo(0, 0);
my_mc.endFill();
```

関連項目

Rectangle (flash.geom.Rectangle)

scrollRect (MovieClip.scrollRect プロパティ)

public scrollRect : Object

scrollRect プロパティを使用すると、ムービークリップのコンテンツをすばやくスクロールして、より多くのコンテンツを表示できるウィンドウを利用できるようになります。テキストフィールドと複雑なコンテンツは、より速くスクロールします。データのスクロールでは、ベクターデータからムービークリップ全体を生成し直す代わりに、ピクセルレベルのコピーが使用されているためです。パフォーマンスを向上させるには、scrollRect と cacheAsBitmap を true に設定したムービークリップとを組み合わせて使用します。

ムービークリップは、特定の幅、高さ、スクロールオフセットを使用して、トリミングされスクロールされます。scrollRect プロパティは、ムービークリップの座標空間に格納され、ムービークリップ全体と同様に拡大･縮小されます。スクロールしているムービークリップ上のトリミングされたウィンドウの隅の境界は、ムービークリップの原点 (0, 0) とポイント (scrollWidth, scrollHeight) です。これらのポイントは原点を中心に配置されず、左上隅の原点を使用します。スクロール対象のムービークリップは、常にピクセル単位でスクロールします。ムービークリップを 90 度回転し、scrollRect.x プロパティを設定して左右にスクロールすると、このムービークリップは上下にスクロールします。

このプロパティを flash.geom.Rectangle オブジェクトに設定すると、ムービークリップが特定のサイズにトリミングされ、スクロールされます。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、setUpContainer() 関数を呼び出して MovieClip 階層を設定した後、新しい矩形を scrollRect プロパティとして設定します。

```
import flash.geom.Rectangle;
var container:MovieClip = setUpContainer();
var window:Rectangle = new Rectangle(0, 0, 100, 40);
container.scrollRect = window;

function setUpContainer():MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip("container",
  this.getNextHighestDepth());
  mc._x = 50;
  mc._y = 50;
  mc.opaqueBackground = 0xCCCCCC;

  var content:MovieClip = mc.createEmptyMovieClip("content",
  mc.getNextHighestDepth());
  var colors:Array = [0xFF0000, 0x0000FF];
  var alphas:Array = [100, 100];
  var ratios:Array = [0, 0xFF];
  var matrix:Object = {a:150, b:0, c:0, d:0, e:150, f:0, g:150, h:150, i:1};
```

```
    content.beginGradientFill("linear", colors, alphas, ratios, matrix);
    content.lineTo(300, 0);
    content.lineTo(300, 300);
    content.lineTo(0, 300);
    content.lineTo(0, 0);
    content.endFill();
    content._rotation = -90;

    mc.onEnterFrame = function() {
      this.content._y += 1;
    }

    return mc;
}
```

setUpContainer() 関数は、次の手順を実行します。

- container という名前の MovieClip を作成します。
- container 内に content という名前の MovieClip を作成します。
- MovieClip content 内にグラデーションシェイプを描画します。
- MovieClip container への参照を返します。

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

setMask (MovieClip.setMask メソッド)

```
public setMask(mc:Object) : Void
```

mc パラメータのムービークリップをマスクにして、呼び出し元のムービークリップを表示します。

setMask() メソッドを使用すると、複雑な多重レイヤーのコンテンツを持つ複数フレームのムービークリップをマスクとして設定できます (マスクレイヤーを使用すれば可能です)。マスクされたムービークリップにあるデバイスフォントは、描画されますがマスクされません。ムービークリップをそれ自身のマスクとして指定することはできません。たとえば、my_mc.setMask(my_mc) とは記述できません。

setMask() で複数のオブジェクトをマスクするには、コンテナとする 1 つのムービークリップ内にそれらのオブジェクトを配置し、次にそのコンテナムービークリップをマスクします。この操作には、グローバルプロパティ this を使用します。たとえば、リンケージ識別子がそれぞれ "theMaskee1_mc"、"theMaskee2_mc"、"circleMask_mc" の 3 つのムービークリップがライブラリにある場合は、次の ActionScript を使用できます (インスタンスの深度は別々である必要があります)。

```
this.attachMovie("theMaskee1_mc", "theMaskee1_instance", 10,{_x:200, _y:100});
this.attachMovie("theMaskee2_mc", "theMaskee2_instance", 20,{_x:200, _y:200});
this.attachMovie("circleMask_mc", "circleMask_instance", 30 ,{_x:200, _y:150});
this.setMask(circleMask_instance);
```

ムービークリップを含むマスクレイヤーを作成してから、それに setMask() メソッドを適用すると、setMask() の呼び出しが優先され、元に戻すことはできません。たとえば、マスクレイヤー内のムービークリップ UIMaskee をマスクする別のレイヤーの別のムービークリップ UIMask があるとします。SWF ファイルの再生中に UIMask.setMask(UIMaskee) を呼び出すと、その時点から UIMask が UIMaskee によってマスクされます。

ActionScript で作成したマスクをキャンセルするには、setMask() に null を渡します。次のコードでは、タイムラインのマスクレイヤーに影響を与えずにマスクをキャンセルします。

```
UIMask.setMask(null);
```

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

mc:Object - マスクとして使用するムービークリップのインスタンス名。ストリングまたは MovieClip を使用できます。

例

次の例では、ムービークリップ circleMask_mc をムービークリップ theMaskee_mc のマスクとして使用します。

```
theMaskee_mc.setMask(circleMask_mc);
```

_soundbuftime (MovieClip._soundbuftime プロパティ)

```
public _soundbuftime : Number
```

サウンドのストリーミングを開始するまでにサウンドをプリバッファする秒数を指定します。

メモ： このプロパティは、MovieClip オブジェクトに対して指定することもできますが、実際にはロードされたすべてのサウンドに適用されるグローバルプロパティであるので、単に _soundbuftime として値を指定することもできます。MovieClip オブジェクトにこのプロパティを設定すると、実際にはグローバルプロパティが設定されます。

対応バージョン： ActionScript 1.0、Flash Player 6

関連項目

_soundbuftime プロパティ

startDrag (MovieClip.startDrag メソッド)

```
public startDrag([lockCenter:Boolean], [left:Number], [top:Number],
  [right:Number], [bottom:Number]) : Void
```

指定されたムービークリップをユーザーがドラッグできるようにします。MovieClip.stopDrag() を呼び出して明示的に停止するか、他のムービークリップをドラッグ可能にするまでの間、ムービーはドラッグ可能なままになります。一度に 1 つのムービークリップのみドラッグ可能です。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

lockCenter:Boolean (オプション) - ドラッグ可能なムービークリップがマウス位置の中心にロックされるか (true)、ユーザーがムービークリップ上で最初にクリックした点にロックされるか (false) を指定するブール値。

left:Number (オプション) - ムービークリップの制限矩形を指定するムービークリップの親の座標を基準にした相対値。

top:Number (オプション) - ムービークリップの制限矩形を指定するムービークリップの親の座標を基準にした相対値。

right:Number (オプション) - ムービークリップの制限矩形を指定するムービークリップの親の座標を基準にした相対値。

bottom:Number (オプション) - ムービークリップの制限矩形を指定するムービークリップの親の座標を基準にした相対値。

例

次の例では、ドラッグ可能なムービークリップインスタンス mc_1 を作成します。

```
this.createEmptyMovieClip("mc_1", 1);

with (mc_1) {
  lineStyle(1, 0xCCCCCC);
  beginFill(0x4827CF);
  moveTo(0, 0);
  lineTo(80, 0);
  lineTo(80, 60);
  lineTo(0, 60);
  lineTo(0, 0);
  endFill();
}
```

```
mc_1.onPress = function() {
  this.startDrag();
};
mc_1.onRelease = function() {
  this.stopDrag();
};
```

関連項目

_droptarget (MovieClip._droptarget プロパティ),startDrag 関数,stopDrag (MovieClip.stopDrag メソッド)

stop (MovieClip.stop メソッド)

```
public stop() : Void
```

再生中のムービークリップを停止します。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、ムービークリップ aMovieClip を停止します。

```
aMovieClip.stop();
```

関連項目

stop 関数

stopDrag (MovieClip.stopDrag メソッド)

```
public stopDrag() : Void
```

MovieClip.startDrag() メソッドを終了します。このメソッドによってドラッグ可能になったムービークリップは、stopDrag() メソッドを実行するか、他のムービークリップがドラッグ可能になるまで、ドラッグ可能のままです。一度に 1 つのムービークリップのみドラッグ可能です。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、ドラッグ可能なムービークリップインスタンス mc_1 を作成します。

```
this.createEmptyMovieClip("mc_1", 1);

with (mc_1) {
  lineStyle(1, 0xCCCCCC);
  beginFill(0x4827CF);
  moveTo(0, 0);
  lineTo(80, 0);
  lineTo(80, 60);
  lineTo(0, 60);
  lineTo(0, 0);
  endFill();
}

mc_1.onPress = function() {
  this.startDrag();
};
mc_1.onRelease = function() {
  this.stopDrag();
};
```

関連項目

_droptarget (MovieClip._droptarget プロパティ), startDrag (MovieClip.startDrag メソッド), stopDrag 関数

swapDepths (MovieClip.swapDepths メソッド)

```
public swapDepths(target:Object) : Void
```

ムービークリップのスタッキング順序、つまり深度 (z 順序) を、target パラメータに指定したムービーと入れ替えます。または、target パラメータに指定した深度に現在置かれているムービーと入れ替えます。両方のムービークリップは、同じ親ムービークリップに属している必要があります。ムービークリップの深度を入れ替えると、あるムービークリップを他のムービークリップの前面または背面に移動させるという効果が得られます。このメソッドを呼び出すときにムービークリップがトゥイーンしている場合、トゥイーンは停止します。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`target:Object` - このパラメータは、次のいずれかの形式で指定できます。

- ムービークリップを配置する深度を指定する数値。
- 別のムービークリップインスタンスを指定するインスタンス名。指定したムービークリップインスタンスとこのメソッドの適用先ムービークリップの深度が入れ替わります。両方のムービークリップは、同じ親ムービークリップに属している必要があります。

例

次の例では、2 つのムービークリップインスタンスの重ね順を入れ替えます。myMC1_mc と myMC2_mc の 2 つのムービークリップインスタンスをステージ上で重ねた後、親のタイムラインに次のスクリプトを追加します。

```
myMC1_mc.onRelease = function() {
  this.swapDepths(myMC2_mc);
};
myMC2_mc.onRelease = function() {
  this.swapDepths(myMC1_mc);
};
```

関連項目

_level プロパティ, getDepth (MovieClip.getDepth メソッド), getInstanceAtDepth (MovieClip.getInstanceAtDepth メソッド), getNextHighestDepth (MovieClip.getNextHighestDepth メソッド)

tabChildren (MovieClip.tabChildren プロパティ)

`public tabChildren : Boolean`

ムービークリップの子が自動タブ順に含まれているかどうかを判別します。tabChildren プロパティが undefined または true の場合、そのムービークリップの子は自動タブ順に含まれます。tabChildren の値が false の場合、ムービークリップの子は自動タブ順に含まれません。デフォルト値は undefined です。

対応バージョン： ActionScript 1.0、Flash Player 6

例

ムービークリップとして、複数の項目を含むリストボックスユーザーインターフェースウィジェットを作成するとします。ユーザーが各項目をクリックして選択できるように、各項目をボタンとして実装します。ただし、Tab キーで移動できるのはリストボックス自体のみにします。リストボックス内の項目はタブ順から除外します。項目をタブ順から除外するには、リストボックスの tabChildren プロパティを false に設定します。

tabIndex プロパティが使用されている場合、tabChildren プロパティは意味を持ちません。tabChildren プロパティは自動タブ順にのみ影響します。

次の例では、親ムービークリップ menu_mc 内のすべての子ムービークリップのタブ順が無効になります。

```
menu_mc.onRelease = function(){};
menu_mc.menu1_mc.onRelease = function(){};
menu_mc.menu2_mc.onRelease = function(){};
menu_mc.menu3_mc.onRelease = function(){};
menu_mc.menu4_mc.onRelease = function(){};

menu_mc.tabChildren = false;
```

menu_mc の子のムービークリップインスタンスを自動タブ順に含めるには、コードの最後の行を次のように変更します。

```
menu_mc.tabChildren = true;
```

関連項目

tabIndex (Button.tabIndex プロパティ),tabEnabled (MovieClip.tabEnabled プロパティ),tabIndex (MovieClip.tabIndex プロパティ),tabIndex (TextField.tabIndex プロパティ)

tabEnabled (MovieClip.tabEnabled プロパティ)

```
public tabEnabled : Boolean
```

ムービークリップが自動タブ順に含まれるかどうかを示します。デフォルト値は undefined です。

tabEnabled プロパティが undefined の場合、オブジェクトは、そのオブジェクトで少なくとも 1 つのムービークリップハンドラ (MovieClip.onRelease など) が定義されている場合にのみ、自動タブ順に含まれます。tabEnabled が true の場合、オブジェクトは自動タブ順に含まれます。tabIndex プロパティにも値を代入すると、オブジェクトはカスタムタブ順にも含まれます。

tabEnabled が false の場合は、tabIndex プロパティが設定されていても、オブジェクトは自動タブ順またはカスタムタブ順に含まれません。ただし、MovieClip.tabChildren が true の場合、tabEnabled が false に設定されている子を含め、ムービークリップの子を自動タブ順に含むことができます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、myMC2_mc が自動タブ順に含まれません。

```
myMC1_mc.onRelease = function() {};
myMC2_mc.onRelease = function() {};
myMC3_mc.onRelease = function() {};
myMC2_mc.tabEnabled = false;
```

関連項目

onRelease (MovieClip.onRelease ハンドラ),tabEnabled (Button.tabEnabled プロパティ),tabChildren (MovieClip.tabChildren プロパティ),tabIndex (MovieClip.tabIndex プロパティ),tabEnabled (TextField.tabEnabled プロパティ)

tabIndex (MovieClip.tabIndex プロパティ)

```
public tabIndex : Number
```

ムービー内のオブジェクトのタブ順をカスタマイズできます。tabIndex プロパティのデフォルト値は undefined です。tabIndex プロパティは、ボタン、ムービークリップ、またはテキストフィールドの各インスタンスで設定できます。

SWF ファイルのオブジェクトに tabIndex プロパティがある場合、自動タブ順は無効になり、タブ順は SWF ファイルのオブジェクトの tabIndex プロパティから計算されます。カスタムタブ順には、tabIndex プロパティを持つオブジェクトのみが含まれます。

tabIndex プロパティは正の整数である必要があります。オブジェクトのタブ順は、その tabIndex プロパティに従って昇順に決定されます。tabIndex の値が 1 であるオブジェクトは、tabIndex の値が 2 であるオブジェクトよりも前になります。カスタムタブ順は、SWF ファイルのオブジェクトの階層関係を無視します。tabIndex プロパティがある SWF ファイルのすべてのオブジェクトは、タブ順に挿入されます。複数のオブジェクトの tabIndex に同じ値を使用しないでください。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の ActionScript は 3 つのムービークリップインスタンスのカスタムタブ順を設定します。

```
myMC1_mc.onRelease = function() {};
myMC2_mc.onRelease = function() {};
myMC3_mc.onRelease = function() {};
myMC1_mc.tabIndex = 2;
myMC2_mc.tabIndex = 1;
myMC3_mc.tabIndex = 3;
```

関連項目

tabIndex (Button.tabIndex プロパティ),tabIndex (TextField.tabIndex プロパティ)

_target (MovieClip._target プロパティ)

```
public _target : String ( 読み取り専用 )
```

ムービークリップインスタンスのターゲットパスをスラッシュ表記で返します。ターゲットパスをドット表記に変換するには、eval() 関数を使用します。

対応バージョン : ActionScript 1.0、Flash Player 4

例

次の例では、SWF ファイル内のムービークリップインスタンスのターゲットパスを、スラッシュ表記とドット表記で表示します。

```
for (var i in this) {
  if (typeof (this[i]) == "movieclip") {
  trace("name: " + this[i]._name + ",\t target: " + this[i]._target + ",\t
  target(2):"
       + eval(this[i]._target));
  }
}
```

_totalframes (MovieClip._totalframes プロパティ)

```
public _totalframes : Number ( 読み取り専用 )
```

ムービークリップインスタンス内のフレーム総数です。

対応バージョン : ActionScript 1.0、Flash Player 4

例

次の例では、2 つのムービークリップボタンがタイムラインを制御します。prev_mc ボタンは再生ヘッドを前のフレームに移動し、next_mc ボタンは再生ヘッドを次のフレームに移動します。タイムライン上の一連のフレームにコンテンツを追加し、次の ActionScript をタイムラインのフレーム 1 に追加します。

```
stop();
prev_mc.onRelease = function() {
  var parent_mc:MovieClip = this._parent;
  if (parent_mc._currentframe>1) {
  parent_mc.prevFrame();
  } else {
  parent_mc.gotoAndStop(parent_mc._totalframes);
  }
};
next_mc.onRelease = function() {
  var parent_mc:MovieClip = this._parent;
  if (parent_mc._currentframe<parent_mc._totalframes) {
  parent_mc.nextFrame();
  } else {
  parent_mc.gotoAndStop(1);
  }
};
```

trackAsMenu (MovieClip.trackAsMenu プロパティ)

```
public trackAsMenu : Boolean
```

他のボタンまたはムービークリップがマウスの解放イベントを受け取ることができるかどうかを示すブール値です。trackAsMenu プロパティを使用してメニューを作成できます。trackAsMenu プロパティは、任意のボタンまたはムービークリップオブジェクトで設定できます。trackAsMenu プロパティがない場合、デフォルトの動作は false です。

trackAsMenu プロパティはいつでも変更できます。このプロパティを変更したムービークリップには、新しい動作が直ちに反映されます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ステージ上の 3 つのムービークリップの trackAsMenu プロパティを設定します。1 つのムービークリップをクリックし、2 番目のムービークリップでマウスボタンを離すと、イベントを受け取るインスタンスを確認できます。

```
myMC1_mc.trackAsMenu = true;
myMC2_mc.trackAsMenu = true;
myMC3_mc.trackAsMenu = false;

myMC1_mc.onRelease = clickMC;
myMC2_mc.onRelease = clickMC;
myMC3_mc.onRelease = clickMC;

function clickMC() {
  trace("you clicked the "+this._name+" movie clip.");
};
```

関連項目

trackAsMenu (Button.trackAsMenu プロパティ)

transform (MovieClip.transform プロパティ)

```
public transform : Transform
```

ムービークリップのマトリックス、カラー変換、ピクセル境界に関係するプロパティを持つオブジェクト。特定のプロパティ matrix、colorTransform、および 3 つの読み取り専用プロパティ (concatenatedMatrix、concatenatedColorTransform、および pixelBounds) については、Transform クラスで説明します。

変形オブジェクトの各プロパティは、それ自身がオブジェクトです。これは重要なことです。matrix オブジェクトまたは colorTransform オブジェクトの新しい値を設定する方法として、オブジェクトを作成し、そのオブジェクトを transform.matrix プロパティまたは transform.colorTransform プロパティにコピーすることしかないためです。

たとえば、ムービークリップのマトリックスの tx 値を増やすには、matrix オブジェクト全体のコピーを作成して新しいオブジェクトの tx プロパティを変更した後、新しいオブジェクトを変形オブジェクトの matrix プロパティにコピーする必要があります。

```
var myMatrix:Object = myDisplayObject.transform.matrix;
myMatrix.tx += 10;
myDisplayObject.transform.matrix = myMatrix;
```

tx プロパティを直接設定することはできません。コード

`myDisplayObject.transform.matrix.tx += 10;` は、myDisplayObject に影響を与えません。

変形オブジェクト全体をコピーし、それを別のムービークリップの transform プロパティに割り当てることもできます。たとえば、次のコードでは、変形オブジェクト全体を myOldDisplayObj から myNewDisplayObj にコピーします。

```
myNewDisplayObj.transform = myOldDisplayObj.transform;
```

これで、新しいムービークリップ myNewDisplayObj のマトリックス、カラー変換、ピクセル境界は、古いムービークリップ myOldDisplayObj と同じ値になります。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例は、ムービークリップの transform プロパティと Matrix ポジショニングを使用してムービークリップにアクセスし、その位置を変更する方法を示しています。

```
import flash.geom.Matrix;

var rect:MovieClip = createRectangle(20, 80, 0xFF0000);

var translateMatrix:Matrix = new Matrix();
translateMatrix.translate(10, 0);

rect.onPress = function() {
  var tmpMatrix:Matrix = this.transform.matrix;
  tmpMatrix.concat(translateMatrix);
  this.transform.matrix = tmpMatrix;
}

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

関連項目

Transform (flash.geom.Transform)

unloadMovie (MovieClip.unloadMovie メソッド)

```
public unloadMovie() : Void
```

ムービークリップインスタンスの内容を削除します。インスタンスプロパティとクリップハンドラは残ります。

プロパティとクリップハンドラも含めてインスタンスを削除するには、MovieClip.removeMovieClip() を使用します。

サブクラスを作成することにより、MovieClip クラスのメソッドおよびイベントハンドラを拡張できます。

対応バージョン： ActionScript 1.0、Flash Player 5

例

次の例では、ユーザーがムービークリップ box をクリックしたときに、ムービークリップインスタンス box をアンロードします。

```
this.createEmptyMovieClip("box", 1);

with (box) {
  lineStyle(1, 0xCCCCCC);
  beginFill(0x4827CF);
  moveTo(0, 0);
  lineTo(80, 0);
  lineTo(80, 60);
  lineTo(0, 60);
  lineTo(0, 0);
  endFill();
}

box.onRelease = function() {
    box.unloadMovie();
};
```

関連項目

removeMovieClip (MovieClip.removeMovieClip メソッド),attachMovie (MovieClip.attachMovie メソッド),loadMovie (MovieClip.loadMovie メソッド), unloadMovie 関数,unloadMovieNum 関数

_url (MovieClip._url プロパティ)

public _url : String (読み取り専用)

ムービークリップのダウンロード元である SWF、JPEG、GIF、または PNG の各ファイルの URL を取得します。

対応バージョン： ActionScript 1.0、Flash Player 4 - ムービークリップのダウンロード元である JPEG ファイルの URL を取得できるようになったのは Flash Player 6 からです。ムービークリップのダウンロード元である GIF ファイルと PNG ファイルの URL を取得できるようになったのは Flash Player 8 からです。

例

次の例では、インスタンス image_mc にロードされたイメージの URL を [出力] パネルに表示します。

```
this.createEmptyMovieClip("image_mc", 1);
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  trace("_url: "+target_mc._url);
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("http://www.adobe.com/images/shared/product_boxes/112x112/
  box_studio_112x112.jpg", image_mc);
```

次の例では、ContextMenu オブジェクト menu_cm をムービークリップ image_mc に割り当てます。オブジェクト menu_cm には [View Image in Browser] というラベルのカスタムメニュー項目があり、この項目には関数 viewImage() が関連付けられています。

```
var menu_cm:ContextMenu = new ContextMenu();
menu_cm.customItems.push(new ContextMenuItem("View Image in Browser...",
  viewImage));
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  target_mc.menu = menu_cm;
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("photo1.jpg", image_mc);

function viewImage(target_mc:MovieClip, obj:Object) {
  getURL(target_mc._url, "_blank");
}
```

実行時にイメージを右クリック (Windows の場合) または Control キーを押したままクリック (Macintosh の場合) し、コンテキストメニューの [View Image in Browser] を選択すると、ブラウザウィンドウでイメージが開きます。

これらの例で使用している MovieClipLoader クラスには、Flash Player 7 以降が必要です。これらの例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

useHandCursor (MovieClip.useHandCursor プロパティ)

```
public useHandCursor : Boolean
```

ムービークリップ上にマウスが移動したときに、指差しハンドポインタ (ハンドカーソル) カーソルを表示するかどうかを示すブール値。useHandCursor プロパティのデフォルト値は true です。useHandCursor プロパティを有効にするには、ムービークリップに onRelease イベントハンドラが定義されている必要があります。useHandCursor を true に設定した場合、onRelease イベントハンドラが定義されたムービークリップ上にマウスを移動すると、指差し状態のボタン用ハンドカーソルが表示されます。useHandCursor を false に設定すると、代わりに矢印のポインタが使用されます。

useHandCursor プロパティはいつでも変更できます。このプロパティを変更したムービークリップには、新しい動作が直ちに反映されます。useHandCursor プロパティは、プロトタイプオブジェクトから読み取ることができます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、2 つのムービークリップ、myMC1_mc および myMC2_mc の useHandCursor プロパティを設定します。一方のインスタンスのプロパティを true に設定し、もう一方のインスタンスのプロパティを false に設定します。両方のインスタンスとも、イベントを受け取ることができます。

```
myMC1_mc.onRelease = traceMC;
myMC2_mc.onRelease = traceMC;
myMC2_mc.useHandCursor = false;

function traceMC() {
  trace("you clicked: "+this._name);
};
```

関連項目

onRelease (MovieClip.onRelease ハンドラ)

_visible (MovieClip._visible プロパティ)

```
public _visible : Boolean
```

ムービークリップを表示するかどうかを示すブール値です。_visible プロパティが false に設定されている非表示のムービークリップは、使用できません。たとえば、_visible プロパティが false に設定されたムービークリップ内のボタンはクリックできません。

対応バージョン : ActionScript 1.0、Flash Player 4

例

次の例では、2 つのムービークリップ、myMC1_mc および myMC2_mc の _visible プロパティを設定します。一方のインスタンスのプロパティを true に設定し、もう一方のインスタンスのプロパティを false に設定します。インスタンス myMC1_mc は、_visible プロパティが false に設定された後はクリックできなくなります。

```
myMC1_mc.onRelease = function() {
  trace(this._name+"._visible = false");
  this._visible = false;
};
myMC2_mc.onRelease = function() {
  trace(this._name+"._alpha = 0");
  this._alpha = 0;
};
```

関連項目

_visible (Button._visible プロパティ),_visible (TextField._visible プロパティ)

_width (MovieClip._width プロパティ)

```
public _width : Number
```

ピクセル単位で示したムービークリップの幅。

対応バージョン : ActionScript 1.0、Flash Player 4

例

次のコード例では、ムービークリップの高さと幅を [出力] パネルに表示します。

```
this.createEmptyMovieClip("triangle", this.getNextHighestDepth());

triangle.beginFill(0x0000FF, 100);
triangle.moveTo(100, 100);
triangle.lineTo(100, 150);
triangle.lineTo(150, 100);
triangle.lineTo(100, 100);
```

```
trace(triangle._name + " = " + triangle._width + " X " + triangle._height + "
  pixels");
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

_height (MovieClip._height プロパティ)

_x (MovieClip._x プロパティ)

```
public _x : Number
```

親ムービークリップのローカル座標を基準にしてムービークリップの *x* 座標を設定する整数。ムービークリップがメインタイムラインにある場合、その座標系はステージの左上隅を (0,0) として参照します。ムービークリップが、変形されている別のムービークリップの内部にある場合、そのムービークリップの座標系は、それを囲む親ムービークリップのローカル座標系になります。したがって、反時計回りに 90 度回転したムービークリップの場合、そのムービークリップの子は、反時計回りに 90 度回転した座標系を継承します。ムービークリップの座標は、基準点の位置を参照します。

対応バージョン : ActionScript 1.0、Flash Player 3

例

次の例では、リンケージ識別子 cursor_id のムービークリップを SWF ファイルに割り当てます。このムービークリップの名前は cursor_mc, で、デフォルトのマウスポインタの置換に使用されます。次の ActionScript は、ムービークリップインスタンスの現在の座標を、マウスポインタの位置に設定します。

```
this.attachMovie("cursor_id", "cursor_mc", this.getNextHighestDepth(),
  {_x:_xmouse, _y:_ymouse});
Mouse.hide();
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  cursor_mc._x = _xmouse;
  cursor_mc._y = _ymouse;
  updateAfterEvent();
};
Mouse.addListener(mouseListener);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

_xscale (MovieClip._xscale プロパティ), _y (MovieClip._y プロパティ), _yscale (MovieClip._yscale プロパティ)

_xmouse (MovieClip._xmouse プロパティ)

public _xmouse : Number (読み取り専用)

マウス位置の *x* 座標を返します。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、ステージ (_level0) 上のマウスの現在の *x* 座標と *y* 座標と、そのステージ上のムービークリップ my_mc を基準とする相対位置を返します。

```
this.createTextField("mouse_txt", this.getNextHighestDepth(), 0, 0, 150, 66);
mouse_txt.html = true;
mouse_txt.multiline = true;
var row1_str:String = " \t<b>_xmouse\t</b><b>_ymouse</b>";
my_mc.onMouseMove = function() {
  mouse_txt.htmlText = "<textformat tabStops='[50,100]'>";
  mouse_txt.htmlText += row1_str;
  mouse_txt.htmlText += "<b>_level0</b>\t"+_xmouse+"\t"+_ymouse;
  mouse_txt.htmlText += "<b>my_mc</b>\t"+this._xmouse+"\t"+this._ymouse;
  mouse_txt.htmlText += "</textformat>";
};
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

Mouse, _ymouse (MovieClip._ymouse プロパティ)

_xscale (MovieClip._xscale プロパティ)

```
public _xscale : Number
```

ムービークリップの基準点から適用するムービークリップの水平スケール (*percentage*) を決定します。デフォルトの基準点は (0,0) です。

ローカル座標系を拡大または縮小すると、ピクセル全体で定義されている _x プロパティと _y プロパティの設定に影響します。たとえば、親ムービークリップを 50% に縮小して、_x プロパティを設定すると、ムービークリップ内のオブジェクトの移動距離は、ムービーの拡大・縮小率が 100% だったときの半分のピクセル数になります。

_xscale プロパティを負の値に設定すると、ムービークリップは水平方向に拡大・縮小および反転します。

対応バージョン : ActionScript 1.0、Flash Player 4

例

次の例では、実行時にムービークリップ box_mc を作成します。描画 API を使用してこのインスタンス内にボックスを描画し、ボックスの上にマウスが移動すると、ムービークリップに水平方向および垂直方向の拡大・縮小が適用されます。マウスがインスタンスから離れると、元の拡大・縮小率に戻ります。

```
this.createEmptyMovieClip("box_mc", 1);
box_mc._x = 100;
box_mc._y = 100;
with (box_mc) {
  lineStyle(1, 0xCCCCCC);
  beginFill(0xEEEEEE);
  moveTo(0, 0);
  lineTo(80, 0);
  lineTo(80, 60);
  lineTo(0, 60);
  lineTo(0, 0);
  endFill();
};
box_mc.onRollOver = function() {
  this._x -= this._width/2;
  this._y -= this._height/2;
  this._xscale = 200;
  this._yscale = 200;
};
box_mc.onRollOut = function() {
  this._xscale = 100;
  this._yscale = 100;
  this._x += this._width/2;
  this._y += this._height/2;
};
```

関連項目

_width (MovieClip._width プロパティ), _x (MovieClip._x プロパティ), _y (MovieClip._y プロパティ), _yscale (MovieClip._yscale プロパティ)

_y (MovieClip._y プロパティ)

public _y : Number

親ムービークリップのローカル座標を基準にしてムービークリップの y 座標を設定します。ムービークリップがメインタイムラインにある場合、その座標系はステージの左上隅を (0,0) として参照します。変形を含んでいる別のムービークリップの内部にムービークリップがある場合、そのムービークリップの座標系は、それを囲む親ムービークリップのローカル座標系になります。したがって、反時計回りに 90 度回転したムービークリップの場合、そのムービークリップの子は、反時計回りに 90 度回転した座標系を継承します。ムービークリップの座標は、基準点の位置を参照します。

対応バージョン : ActionScript 1.0、Flash Player 3

例

次の例では、リンケージ識別子 cursor_id のムービークリップを SWF ファイルに割り当てます。このムービークリップの名前は cursor_mc で、デフォルトのマウスポインタの置換に使用されます。次の ActionScript は、ムービークリップインスタンスの現在の座標を、マウスポインタの位置に設定します。

```
this.attachMovie("cursor_id", "cursor_mc", this.getNextHighestDepth(),
  {_x:_xmouse, _y:_ymouse});
Mouse.hide();
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  cursor_mc._x = _xmouse;
  cursor_mc._y = _ymouse;
  updateAfterEvent();
};
Mouse.addListener(mouseListener);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

_x (MovieClip._x プロパティ), _xscale (MovieClip._xscale プロパティ), _yscale (MovieClip._yscale プロパティ)

_ymouse (MovieClip._ymouse プロパティ)

public _ymouse : Number (読み取り専用)

マウス位置の y 座標を示します。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、ステージ (_level0) 上のマウスの現在の *x* 座標と *y* 座標と、そのステージ上のムービークリップ my_mc を基準とする相対位置を返します。

```
this.createTextField("mouse_txt", this.getNextHighestDepth(), 0, 0, 150, 66);
mouse_txt.html = true;
mouse_txt.multiline = true;
var row1_str:String = " \t<b>_xmouse\t</b><b>_ymouse</b>";
my_mc.onMouseMove = function() {
  mouse_txt.htmlText = "<textformat tabStops='[50,100]'>";
  mouse_txt.htmlText += row1_str;
  mouse_txt.htmlText += "<b>_level0</b>\t"+_xmouse+"\t"+_ymouse;
  mouse_txt.htmlText += "<b>my_mc</b>\t"+this._xmouse+"\t"+this._ymouse;
  mouse_txt.htmlText += "</textformat>";
};
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

Mouse, _xmouse (MovieClip._xmouse プロパティ)

_yscale (MovieClip._yscale プロパティ)

public _yscale : Number

ムービークリップの基準点から適用するムービークリップの垂直スケール (*percentage*) を設定します。デフォルトの基準点は (0,0) です。

ローカル座標系を拡大または縮小すると、ピクセル全体で定義されている _x プロパティと _y プロパティの設定に影響します。たとえば、親ムービークリップを 50% に縮小して、_x プロパティを設定すると、ムービークリップ内のオブジェクトの移動距離は、ムービーの拡大・縮小率が 100% だったときの半分のピクセル数になります。

対応バージョン : ActionScript 1.0、Flash Player 4

例

次の例では、実行時にムービークリップ box_mc を作成します。描画 API を使用してこのインスタンス内にボックスを描画し、ボックスの上にマウスが移動すると、ムービークリップに水平方向および垂直方向の拡大・縮小が適用されます。マウスがインスタンスから離れると、元の拡大・縮小率に戻ります。

```
this.createEmptyMovieClip("box_mc", 1);
box_mc._x = 100;
box_mc._y = 100;
with (box_mc) {
  lineStyle(1, 0xCCCCCC);
  beginFill(0xEEEEEE);
  moveTo(0, 0);
  lineTo(80, 0);
  lineTo(80, 60);
  lineTo(0, 60);
  lineTo(0, 0);
  endFill();
};
box_mc.onRollOver = function() {
  this._x -= this._width/2;
  this._y -= this._height/2;
  this._xscale = 200;
  this._yscale = 200;
};
box_mc.onRollOut = function() {
  this._xscale = 100;
  this._yscale = 100;
  this._x += this._width/2;
  this._y += this._height/2;
};
```

関連項目

_height (MovieClip._height プロパティ),_x (MovieClip._x プロパティ),_xscale (MovieClip._xscale プロパティ),_y (MovieClip._y プロパティ)

MovieClipLoader

```
Object
  |
  +-MovieClipLoader
```

```
public class MovieClipLoader
extends Object
```

このクラスを使用すると、SWF、JPEG、GIF、および PNG の各ファイルのムービークリップへのロード時にステータス情報を提供するリスナーコールバックを実装できます。MovieClipLoader の機能を使用するには、`loadMovie()` や `MovieClip.loadMovie()` ではなく、`MovieClipLoader.loadClip()` を使用して SWF ファイルをロードします。

メモ : MovieClipLoader の GIF および PNG ファイル形式のサポートは、Flash 8 の新機能です。これよりも前のバージョンのツールで作成されたプロジェクトを読み込む場合は、パブリッシュ設定を更新して Flash 8 以降をターゲットにする必要があります。この操作をしないと、SWF と JPEG ファイル形式しか有効になりません。

`MovieClipLoader.loadClip()` コマンドを発行した後、次に列挙した順にイベントが発生します。

- ダウンロード対象ファイルの先頭バイトがハードディスクに書き込まれると、`MovieClipLoader.onLoadStart` リスナーが呼び出されます。
- `MovieClipLoader.onLoadProgress` リスナーを実装している場合は、このリスナーがロードプロセス中に呼び出されます。**メモ :** `MovieClipLoader.getProgress()` は、ロードプロセス中にいつでも呼び出すことができます。
- ダウンロード対象ファイル全体がハードディスクに書き込まれると、`MovieClipLoader.onLoadComplete` リスナーが呼び出されます。
- ダウンロード対象ファイルの最初のフレームアクションが実行されると、`MovieClipLoader.onLoadInit` リスナーが呼び出されます。

`MovieClipLoader.onLoadInit` が呼び出されると、プロパティを設定したりメソッドを使用するなどの方法で、ロード済みのムービーを操作することができます。

ファイルが完全にロードされなかった場合は、`MovieClipLoader.onLoadError` リスナーが呼び出されます。

メモ : ブラウザによっては、ファイルのローカルキャッシュ動作が MovieClipLoader のイベントに影響する場合があります。SWF ファイルの開発中に MovieClipLoader のイベントをテストする場合は、あらかじめブラウザのキャッシュをクリアしておいてください。

対応バージョン : ActionScript 1.0、Flash Player 7

プロパティー覧

オプション	プロパティ	説明
	checkPolicyFile: Boolean	オブジェクト自体のロードを開始する前に、Flash Player が、ロードされるオブジェクトのサーバーからクロスドメインポリシーファイルをダウンロードしようとするかどうかを指定します。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベント一覧

イベント	説明
onLoadComplete = function([target_mc: MovieClip], [httpStatus: Number]) {}	MovieClipLoader.loadClip() でロードされたファイルが完全にダウンロードされたときに呼び出されます。
onLoadError = function(target_mc: MovieClip, errorCode:String, [httpStatus: Number]) {}	MovieClipLoader.loadClip() でロードされたファイルがロードに失敗したときに呼び出されます。
onLoadInit = function([target_mc: MovieClip]) {}	ロード対象クリップの先頭フレーム上のアクションが実行されたときに呼び出されます。
onLoadProgress = function([target_mc: MovieClip], loadedBytes:Number, totalBytes: Number) {}	ロードプロセス中 (つまり MovieClipLoader.onLoadStart から MovieClipLoader.onLoadComplete までの間)、ロード対象のコンテンツがディスクに書き込まれるたびに呼び出されます。
onLoadStart = function([target_mc: MovieClip]) {}	MovieClipLoader.loadClip() の呼び出しでファイルのダウンロードが開始されたときに呼び出されます。

コンストラクター覧

署名	説明
MovieClipLoader()	SWF、JPEG、GIF、または PNG の各ファイルのダウンロード時、イベントに応答するリスナーを実装するための MovieClipLoader オブジェクトを生成します。

メソッド一覧

オプション	署名	説明
	addListener (listener:Object) : Boolean	MovieClipLoader イベントハンドラが呼び出されたときに通知を受けるオブジェクトを登録します。
	getProgress(target:Object) : Object	MovieClipLoader.loadClip() でロードされているファイルについて、ロード済みのバイト数および総バイト数を返します。ムービーが圧縮されている場合、圧縮された状態のバイト数を返します。
	loadClip(url:String, target:Object) : Boolean	元のムービーの再生中に、SWF、JPEG、プログレッシブ JPEG、非アニメーション GIF、または PNG の各ファイルを Flash Player のムービークリップ内にロードします。
	removeListener (listener:Object) : Boolean	MovieClipLoader イベントハンドラが呼び出されたときの通知を待機するリスナーを削除します。
	unloadClip(target:Object) : Boolean	MovieClipLoader.loadClip() でロードされたムービークリップを削除します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

addListener (MovieClipLoader.addListener メソッド)

```
public addListener(listener:Object) : Boolean
```

MovieClipLoader イベントハンドラが呼び出されたときに通知を受けるオブジェクトを登録します。

対応バージョン： ActionScript 1.0、Flash Player 7

パラメータ

listener:Object - MovieClipLoader イベントハンドラからのコールバック通知を待機するオブジェクト。

戻り値

Boolean - ブール値。リスナーが正常に確立された場合は true、それ以外の場合は false を返します。

例

次の例では、image_mc というムービークリップにイメージをロードします。ムービークリップインスタンスは回転してステージ上に配置され、ステージおよびムービークリップの周囲に線が描かれます。

```
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  target_mc._x = Stage.width/2-target_mc._width/2;
  target_mc._y = Stage.height/2-target_mc._height/2;
  var w:Number = target_mc._width;
  var h:Number = target_mc._height;
  target_mc.lineStyle(4, 0x000000);
  target_mc.moveTo(0, 0);
  target_mc.lineTo(w, 0);
  target_mc.lineTo(w, h);
  target_mc.lineTo(0, h);
  target_mc.lineTo(0, 0);
  target_mc._rotation = 3;
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  image_mc);
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

onLoadComplete (MovieClipLoader.onLoadComplete イベントリスナー),onLoadError (MovieClipLoader.onLoadError イベントリスナー),onLoadInit (MovieClipLoader.onLoadInit イベントリスナー),onLoadProgress (MovieClipLoader.onLoadProgress イベントリスナー),onLoadStart (MovieClipLoader.onLoadStart イベントリスナー),removeListener (MovieClipLoader.removeListener メソッド)

checkPolicyFile (MovieClipLoader.checkPolicyFile プロパティ)

```
public checkPolicyFile : Boolean
```

オブジェクト自体のロードを開始する前に、Flash Player が、ロードされるオブジェクトのサーバーからクロスドメインポリシーファイルをダウンロードしようとするかどうかを指定します。

イメージ (JPG、GIF、または PNG) を呼び出し元 SWF ファイル自体のドメイン外からロードし、そのイメージのコンテンツに BitmapData.draw() を使用してアクセスする予定の場合に、このフラグを true に設定します。ロード時に checkPolicyFile を指定せずにこの操作を試行すると、必要なポリシーファイルがまだダウンロードされていないために、セキュリティエラーになる場合があります。

checkPolicyFile を true に設定して MovieClipLoader.loadClip() メソッドを呼び出した場合、関連するクロスドメインポリシーファイルを正常にダウンロードするか、そのようなポリシーファイルが存在しないことがわかるまで、Flash Player は url で指定されたオブジェクトのダウンロードを開始しません。Flash Player では、既にダウンロードされているポリシーファイルが最初に考慮され、次に System.ecurity.loadPolicyFile() の呼び出しで指定された保留されているポリシーファイルのダウンロードが試行されます。次に、url に対応するデフォルトの場所 (url と同じサーバーの /crossdomain.xml) からのポリシーファイルのダウンロードが試行されます。どの場合でも、該当するポリシーファイルがそのサーバー上に存在していることが必要です。また、そのポリシーファイルでは、ポリシーファイルの場所の url にあるオブジェクトにアクセスできるようになっており、呼び出し元 SWF ファイルのドメインからのアクセスが <allow-access-from> タグで許可されていることが必要です。

checkPolicyFile を true に設定した場合、Flash Player ではポリシーファイルが完了するまで、MovieClipLoader.loadClip() で指定する主要なダウンロードの開始が待機されます。したがって、必要なポリシーファイルが存在していれば、MovieClipLoader からイベント通知を受け取ると直ちにポリシーファイルのダウンロードが完了し、ポリシーファイルを必要とする操作の実行を安全に開始できます。

checkPolicyFile を true に設定した場合に、関連するポリシーファイルが見つからなくても、セキュリティエラーになる操作を試行するまでエラーは通知されません。

ロードするイメージへのピクセルレベルのアクセスを必要としない場合は、checkPolicyFile を true に設定しないようにしてください。この場合、ポリシーファイルの確認は時間の浪費になります。ダウンロードの開始が遅れ、ネットワーク帯域幅を不必要に消費する場合があるためです。

また、MovieClipLoader.loadClip() を使用して SWF ファイルをダウンロードする場合も、checkPolicyFile を true に設定しないようにしてください。これは、SWF から SWF へのアクセス許可はポリシーファイルではなく Security.allowDomain() によって制御されるので、SWF ファイルをロードする場合は checkPolicyFile の効果がないためです。この場合、ポリシーファイルの確認は時間の浪費になります。SWF のダウンロードの開始が遅れ、ネットワーク帯域幅を不必要に消費する場合があるためです。ポリシーファイルのダウンロードは主要なダウンロードの前に発生するため、Flash Player では主要なダウンロードが SWF ファイルまたはイメージのいずれであるかはわかりません。

サーバーサイド HTTP リダイレクトを使用する可能性がある URL からオブジェクトをダウンロードする場合は、checkPolicyFile に注意してください。Flash Player は、常に loadClip で指定する初期 URL に対応するポリシーファイルを取得しようとします。最終的なオブジェクトが HTTP リダイレクトによって別の URL から取得される場合、最初にダウンロードされたポリシーファイルはオブジェクトの最終的な URL に適用できないことがあります。この URL はセキュリティ判定において重要です。

対応バージョン： ActionScript 2.0、Flash Player 9

getProgress (MovieClipLoader.getProgress メソッド)

```
public getProgress(target:Object) : Object
```

MovieClipLoader.loadClip() でロードされているファイルについて、ロード済みのバイト数および総バイト数を返します。ムービーが圧縮されている場合、圧縮された状態のバイト数を返します。getProgress メソッドを使用すると、MovieClipLoader.onLoadProgress リスナー関数を作成または追加しなくても、こういった情報を明示的に要求できます。

対応バージョン： ActionScript 1.0、Flash Player 7

パラメータ

target:Object - MovieClipLoader.loadClip() を使用してロードされる SWF、JPEG、GIF、または PNG の各ファイル。

戻り値

Object - bytesLoaded および bytesTotal の 2 つの整数プロパティを持つオブジェクト。

例

次の例では、getProgress() メソッドの使用方法を示します。通常は、このメソッドを使用するのではなく、リスナーオブジェクトを作成して onLoadProgress イベントを待機します。また、getProgress() の同期呼び出しにより、コンテナについてロード済みのバイト数と合計バイト数を返すことができますが、外部で要求されたオブジェクトの値は対象外であることに注意してください。

```
var container:MovieClip = this.createEmptyMovieClip("container",
  this.getNextHighestDepth());
var image:MovieClip = container.createEmptyMovieClip("image",
  container.getNextHighestDepth());

var mcLoader:MovieClipLoader = new MovieClipLoader();
var listener:Object = new Object();
listener.onLoadProgress = function(target:MovieClip, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace(target + ".onLoadProgress with " + bytesLoaded + " bytes of " +
  bytesTotal);
}
mcLoader.addListener(listener);
mcLoader.loadClip("http://www.w3.org/Icons/w3c_main.png", image);

var interval:Object = new Object();
interval.id = setInterval(checkProgress, 100, mcLoader, image, interval);

function checkProgress(mcLoader:MovieClipLoader, image:MovieClip,
  interval:Object):Void {
  trace(">> checking progress now with : " + interval.id);
  var progress:Object = mcLoader.getProgress(image);
  trace("bytesLoaded: " + progress.bytesLoaded + " bytesTotal: " +
  progress.bytesTotal);
  if(progress.bytesLoaded == progress.bytesTotal) {
    clearInterval(interval.id);
  }
}
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

loadClip (MovieClipLoader.loadClip メソッド), onLoadProgress (MovieClipLoader.onLoadProgress イベントリスナー)

loadClip (MovieClipLoader.loadClip メソッド)

```
public loadClip(url:String, target:Object) : Boolean
```

元のムービーの再生中に、SWF、JPEG、プログレッシブ JPEG、非アニメーション GIF、または PNG の各ファイルを Flash Player のムービークリップ内にロードします。アニメーション GIF を読み込むと、先頭のフレームのみ表示されます。このメソッドを使用すると、複数の SWF ファイルを同時に表示し、別の HTML ドキュメントをロードせずに SWF ファイルを切り替えることができます。

loadClip() メソッドを loadMovie() や MovieClip.loadMovie() の代わりに使用することには有利な点がいくつかあります。次のハンドラは、リスナーオブジェクトの使用によって実装されます。リスナーをアクティブにするには、MovieClipLoader.addListener(listenerObject) を使用して、MovieClipLoader クラスでリスナーを登録します。

- ロードが開始されたときに MovieClipLoader.onLoadStart ハンドラが呼び出される。
- クリップをロードできなかった場合に MovieClipLoader.onLoadError ハンドラが呼び出される。
- ロードプロセスの進行時にリアルタイムに MovieClipLoader.onLoadProgress ハンドラが呼び出される。
- ファイルのダウンロードが終了し、ロードされたムービークリップのメソッドとプロパティが使用可能になる前に、MovieClipLoader.onLoadComplete ハンドラが呼び出される。このハンドラは、onLoadInit ハンドラの前に呼び出されます。
- クリップの先頭フレームのアクションが実行された後に MovieClipLoader.onLoadInit ハンドラが呼び出される。そのため、ロード済みのクリップに対して各種の操作を開始できます。このハンドラは、onLoadComplete ハンドラの後に呼び出されます。ほとんどの場合、onLoadInit ハンドラを使用します。

ムービークリップ内にロードした SWF ファイルまたはイメージは、そのムービークリップの位置、回転、および拡大・縮小の各プロパティを継承します。ムービークリップのターゲットパスを使用して、ロードしたムービーをターゲットとして設定できます。

loadClip() メソッドは、単一のムービークリップまたはレベルにファイル (複数のファイルが可能です) をロードするときに使用できます。MovieClipLoader リスナーオブジェクトは、ロード中のターゲットムービークリップのインスタンスにパラメータとして渡されます。代わりに、ロードするファイルごとに異なる MovieClipLoader オブジェクトを作成することもできます。

このメソッドを使用してロードされたムービーまたはイメージを削除したり、進行中のロード処理をキャンセルするには、MovieClipLoader.unloadClip() を使用します。

MovieClipLoader.getProgress() および MovieClipLoaderListener.onLoadProgress は、ファイルがローカルである場合、オーサリングプレーヤーの実際の bytesLoaded および bytesTotal の値を報告しません。オーサリング環境でプロファイラ機能を使用すると、MovieClipLoader.getProgress() および MovieClipLoaderListener.onLoadProgress は、プロファイラで提供される減少した帯域幅レートではなく、実際のダウンロードレートでダウンロードを報告します。

このメソッドを使用するときは、Flash Player セキュリティモデルを考慮してください。

Flash Player 8 以降 :

- 呼び出し元のムービークリップがローカルファイルシステムのサンドボックスにあり、ロードするムービークリップがネットワーク上のサンドボックスにある場合、ロードできません。
- 呼び出し元 SWF ファイルがネットワーク上のサンドボックスにあり、ロードするムービークリップがローカルにある場合、ロードできません。
- 信頼できるローカルのサンドボックスまたはネットワーク接続したローカルのサンドボックスからネットワーク上のサンドボックスにアクセスするには、クロスドメインポリシーファイルを使用して Web サイトで許可する必要があります。
- ローカルファイルシステムのサンドボックスにあるムービークリップでは、ネットワーク接続したローカルのサンドボックスにあるムービークリップをスクリプト処理できません。その逆も同様です。

Flash Player 7 :

- Web サイトでクロスドメインポリシーファイルを使用して、リソースへのクロスドメインアクセスを許可できます。
- SWF ファイル間のスクリプト処理は、SWF ファイルが置かれているドメインに基づいて制限されます。これらの制限を調整するには、System.security.allowDomain() メソッドを使用します。

詳細については、次の参照先を参照してください。

- 『ActionScript 2.0 の学習』の「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

対応バージョン : ActionScript 1.0、Flash Player 7 - 非アニメーション GIF ファイル、PNG ファイル、またはプログレッシブ JPEG ファイルが Flash Player 8 でサポートされました。

パラメータ

`url:String` - ロードする SWF、JPEG、GIF、または PNG の各ファイルの絶対 URL または相対 URL。相対パスは、レベル 0 の SWF ファイルが埋め込まれた HTML ファイルを基準にする必要があります。絶対 URL の場合は http:// や file:// などのプロトコル参照を含めて指定します。ファイル名には、ドライブ指定を含めることはできません。

`target:Object` - ムービークリップのターゲットパス、またはムービーのロード先となる、Flash Player のレベルを指定する整数。ターゲットムービークリップは、ロードした SWF ファイルまたはイメージに置き換えられます。

戻り値

`Boolean` - ブール値。URL リクエストが正常に送信された場合は `true`、それ以外の場合は `false` を返します。

例

次の例では、`onLoadInit` イベント用のハンドラを作成し要求を行うことで、`MovieClipLoader.loadClip()` メソッドの使用方法を示します。

次のコードを、タイムラインのフレームアクションに直接配置するか、MovieClip を拡張するクラスに貼り付ける必要があります。また、このコードでは、コンパイル済みの SWF ファイルと同じディレクトリに YourImage.jpg という名前のイメージがあると想定します。

```
var container:MovieClip = createEmptyMovieClip("container",
  getNextHighestDepth());
var mcLoader:MovieClipLoader = new MovieClipLoader();
mcLoader.addListener(this);
mcLoader.loadClip("YourImage.jpg", container);

function onLoadInit(mc:MovieClip) {
  trace("onLoadInit: " + mc);
}
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している `MovieClip.getNextHighestDepth()` メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

`onLoadInit (MovieClipLoader.onLoadInit イベントリスナー )`

MovieClipLoader コンストラクタ

```
public MovieClipLoader()
```

SWF、JPEG、GIF、または PNG の各ファイルのダウンロード時、イベントに応答するリスナーを実装するための MovieClipLoader オブジェクトを生成します。

対応バージョン： ActionScript 1.0、Flash Player 7

例

`MovieClipLoader.loadClip()` を参照してください。

関連項目

`addListener (MovieClipLoader.addListener メソッド )`, `loadClip (MovieClipLoader.loadClip メソッド )`

onLoadComplete (MovieClipLoader.onLoadComplete イベントリスナー)

```
onLoadComplete = function([target_mc:MovieClip], [httpStatus:Number]) {}
```

`MovieClipLoader.loadClip()` でロードされたファイルが完全にダウンロードされたときに呼び出されます。`MovieClipLoader.addListener()` を使用して追加するリスナーオブジェクトでこのリスナーを呼び出します。`onLoadComplete` イベントリスナーは、Flash Player によってコードに渡されますが、リスナー関数にすべてのパラメータを実装する必要はありません。`target_mc` の値は、この呼び出しが実行されるムービークリップを識別します。この識別は、複数のファイルが同じリスナーセットでロードされる場合に有用です。

Flash Player 8 以降では、このリスナーは HTTP ステータスコードを返すことができます。Flash Player がサーバーからステータスコードを取得できなかった場合、またはサーバーと通信できなかった場合、記述した ActionScript のコードにデフォルト値の 0 が渡されます。値 0 は、(たとえば正しくない形式の URL が要求された場合などに) どのプレーヤーでも生成される可能性があります。また、サーバーからの HTTP ステータスコードを Flash Player に渡すことができないブラウザで実行される Flash Player プラグインでは、常に値 0 が生成されます。該当するブラウザには、Netscape、Mozilla、Safari、Opera、および Internet Explorer for Macintosh があります。

`MovieClipLoader.onLoadComplete` と `MovieClipLoader.onLoadInit` の違いを理解することが重要です。`onLoadComplete` イベントは、SWF、JPEG、GIF、または PNG の各ファイルがロードされた後、アプリケーションが初期化される前に呼び出されます。この時点で、ロードされたムービークリップのメソッドおよびプロパティにアクセスすることはできません。そのため、関数の呼び出しや、特定のフレームの移動などを行うことはできません。ほとんどの場合、代わりに `onLoadInit` イベントを使用することをお勧めします。このイベントは、コンテンツがロードされて完全に初期化された後で呼び出されます。

対応バージョン： ActionScript 1.0、Flash Player 7 - 非アニメーション GIF ファイル、PNG ファイル、またはプログレッシブ JPEG ファイルが Flash Player 8 でサポートされました。

パラメータ

`target_mc:MovieClip` (オプション) - `MovieClipLoader.loadClip()` メソッドでロードされるムービークリップ。

`httpStatus:Number` (オプション) - (Flash Player 8 以降のみ) サーバーから返された HTTP ステータスコード。たとえば、ステータスコード 404 は、要求された URI と一致するものがサーバーで見つからなかったことを示します。HTTP ステータスコードの詳細については、ftp://ftp.isi.edu/in-notes/rfc2616.txt にある HTTP 仕様書のセクション 10.4 と 10.5 を参照してください。

例

次の例では、ムービークリップ、新しい `MovieClipLoader` インスタンス、および匿名のイベントリスナーを作成します。これは、`onLoadComplete` イベントのリスナーですが、ロードされたエレメントのプロパティを使用するために `onLoadInit` イベントを待機します。

```
var loadListener:Object = new Object();

loadListener.onLoadComplete = function(target_mc:MovieClip,
  httpStatus:Number):Void {
  trace(">> loadListener.onLoadComplete()");
  trace(">> ============================");
  trace(">> target_mc._width: " + target_mc._width); // 0
  trace(">> httpStatus: " + httpStatus);
}

loadListener.onLoadInit = function(target_mc:MovieClip):Void {
  trace(">> loadListener.onLoadInit()");
  trace(">> ============================");
  trace(">> target_mc._width: " + target_mc._width); // 315
}

var mcLoader:MovieClipLoader = new MovieClipLoader();
mcLoader.addListener(loadListener);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mcLoader.loadClip("http://www.w3.org/Icons/w3c_main.png", mc);
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している `MovieClip.getNextHighestDepth()` メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

addListener (MovieClipLoader.addListener メソッド), loadClip (MovieClipLoader.loadClip メソッド), onLoadStart (MovieClipLoader.onLoadStart イベントリスナー), onLoadError (MovieClipLoader.onLoadError イベントリスナー), onLoadInit (MovieClipLoader.onLoadInit イベントリスナー)

onLoadError (MovieClipLoader.onLoadError イベントリスナー)

```
onLoadError = function(target_mc:MovieClip, errorCode:String,
  [httpStatus:Number]) {}
```

MovieClipLoader.loadClip() でロードされたファイルがロードに失敗したときに呼び出されます。このリスナーが呼び出される理由はさまざまです。たとえば、サーバーがダウンした場合、ファイルが見つからなかったり、セキュリティ侵害が発生します。

MovieClipLoader.addListener() を使用して追加するリスナーオブジェクトでこのリスナーを呼び出します。

target_mc の値は、この呼び出しが実行されるムービークリップを識別します。このパラメータは、複数のファイルが同じリスナーセットでロードされる場合に有用です。

errorCode パラメータについては、MovieClipLoader.onLoadStart または MovieClipLoader.onLoadComplete のいずれのリスナーも呼び出されなかった場合、ストリング "URLNotFound" が返されます。たとえば、サーバーがダウンしている場合やファイルが見つからなかった場合などです。MovieClipLoader.onLoadStart だけが呼び出され、MovieClipLoader.onLoadComplete は呼び出されなかった場合、ストリング "LoadNeverCompleted" が返されます。たとえば、サーバーに大きな負荷がかかったことによりダウンロードが中断された場合や、サーバーがクラッシュした場合などです。

Flash Player 8 以降では、このリスナーは httpStatus パラメータで HTTP ステータスコードを返すことができます。Flash Player がサーバーからステータスコードを取得できない場合、または Flash Player がサーバーと通信できない場合、デフォルト値 0 が ActionScript コードに渡されます。値 0 は、(たとえば正しくない形式の URL が要求された場合などに) どのプレーヤーでも生成される可能性があります。また、サーバーからの HTTP ステータスコードを Flash Player に渡すことができないブラウザで実行される Flash Player プラグインでは、常に値 0 が生成されます。該当するブラウザには、Netscape、Mozilla、Safari、Opera、および Internet Explorer for Macintosh があります。Flash Player が URL リクエストに基づくロード操作を実行しようとしなかった場合も、値 0 が生成されます。これは、その要求が SWF ファイルのセキュリティサンドボックス規則に違反している場合に起こる可能性があります。

対応バージョン： ActionScript 1.0、Flash Player 7

パラメータ

target_mc:MovieClip - MovieClipLoader.loadClip() メソッドでロードされるムービークリップ。

errorCode:String - "URLNotFound" または "LoadNeverCompleted" のエラーの理由を示すストリング。

httpStatus:Number (オプション) - (Flash Player 8 以降のみ) サーバーから返された HTTP ステータスコード。たとえば、ステータスコード 404 は、要求された URI と一致するものがサーバーで見つからなかったことを示します。HTTP ステータスコードの詳細については、ftp://ftp.isi.edu/in-notes/rfc2616.txt にある HTTP 仕様書のセクション 10.4 と 10.5 を参照してください。

例

次の例では、イメージのロードが失敗すると [出力] パネルに情報を表示します。この例で使用される URL はデモンストレーション専用ですので、有効な URL に置き換えてください。

```
var loadListener:Object = new Object();

loadListener.onLoadError = function(target_mc:MovieClip, errorCode:String,
  httpStatus:Number) {
  trace(">> loadListener.onLoadError()");
  trace(">> ==========================");
  trace(">> errorCode: " + errorCode);
  trace(">> httpStatus: " + httpStatus);
}

var mcLoader:MovieClipLoader = new MovieClipLoader();
mcLoader.addListener(loadListener);

var mc:MovieClip = this.createEmptyMovieClip("mc", this.getNextHighestDepth());
mcLoader.loadClip("http://www.fakedomain.com/images/bad_hair_day.jpg", mc);
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

addListener (MovieClipLoader.addListener メソッド), loadClip (MovieClipLoader.loadClip メソッド), onLoadStart (MovieClipLoader.onLoadStart イベントリスナー), onLoadComplete (MovieClipLoader.onLoadComplete イベントリスナー)

onLoadInit (MovieClipLoader.onLoadInit イベントリスナー)

```
onLoadInit = function([target_mc:MovieClip]) {}
```

ロード対象クリップの先頭フレーム上のアクションが実行されたときに呼び出されます。このリスナーが呼び出されると、プロパティを設定したりメソッドを使用するなどの方法で、ロード済みのムービーを操作することができます。MovieClipLoader.addListener() を使用して追加するリスナーオブジェクトでこのリスナーを呼び出します。

target_mc の値は、この呼び出しが実行されるムービークリップを識別します。このパラメータは、複数のファイルが同じリスナーセットでロードされる場合に有用です。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

target_mc:MovieClip (オプション) - MovieClipLoader.loadClip() メソッドでロードされるムービークリップ。

例

次の例では、image_mc というムービークリップインスタンスにイメージをロードします。

onLoadInit および onLoadComplete イベントを使用して、イメージのロードにどのくらい時間がかかるかを調べます。その情報は、timer_txt というテキストフィールドに表示されます。

```
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadStart = function(target_mc:MovieClip) {
  target_mc.startTimer = getTimer();
};
mclListener.onLoadComplete = function(target_mc:MovieClip) {
  target_mc.completeTimer = getTimer();
};
mclListener.onLoadInit = function(target_mc:MovieClip) {
  var timerMS:Number = target_mc.completeTimer-target_mc.startTimer;
  target_mc.createTextField("timer_txt", target_mc.getNextHighestDepth(), 0,
  target_mc._height,
target_mc._width, 22);
  target_mc.timer_txt.text = "loaded in "+timerMS+" ms.";
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  image_mc);
```

次の例では、実行時に作成されたムービークリップにムービーがロードされたかどうかをチェックします。この例で使用される URL はデモンストレーション専用ですので、有効な URL に置き換えてください。

```
this.createEmptyMovieClip("tester_mc", 1);
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  trace("movie loaded");
}
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("http://www.yourserver.com/your_movie.swf", tester_mc);
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

addListener (MovieClipLoader.addListener メソッド), loadClip (MovieClipLoader.loadClip メソッド), onLoadStart (MovieClipLoader.onLoadStart イベントリスナー)

onLoadProgress (MovieClipLoader.onLoadProgress イベントリスナー)

```
onLoadProgress = function([target_mc:MovieClip], loadedBytes:Number,
  totalBytes:Number) {}
```

ロードプロセス中 (つまり MovieClipLoader.onLoadStart から MovieClipLoader.onLoadComplete までの間)、ロード対象のコンテンツがディスクに書き込まれるたびに呼び出されます。MovieClipLoader.addListener() を使用して追加するリスナーオブジェクトでこのリスナーを呼び出します。このメソッドの loadedBytes パラメータおよび totalBytes パラメータを使用することにより、ダウンロードの進捗情報を表示できます。

target_mc の値は、この呼び出しが実行されるムービークリップを識別します。これは、複数のファイルが同じリスナーセットでロードされる場合に有用です。

メモ : onLoadProgress をプレビューモードで使用してハードディスク上にあるローカルファイルをロードしようとしても、プレビューモードではローカルファイル全体がロードされるため、正常に機能しません。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

target_mc:MovieClip (オプション) - MovieClipLoader.loadClip() メソッドでロードされるムービークリップ。

loadedBytes:Number - リスナーが呼び出された時点で既にロードされていたバイト数。

totalBytes:Number - ロード対象ファイルの総バイト数。

例

次の例では、ムービークリップ、新しい MovieClipLoader インスタンス、および匿名イベントリスナーを作成します。ロードの進行状況を定期的に出力し、ロードが完了して ActionScript でアセットを使用できるようになると最終的な通知を発行します。

```
var container:MovieClip = this.createEmptyMovieClip("container",
  this.getNextHighestDepth());
var mcLoader:MovieClipLoader = new MovieClipLoader();
var listener:Object = new Object();
listener.onLoadProgress = function(target:MovieClip, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace(target + ".onLoadProgress with " + bytesLoaded + " bytes of " +
  bytesTotal);
}
listener.onLoadInit = function(target:MovieClip):Void {
  trace(target + ".onLoadInit");
}
mcLoader.addListener(listener);
mcLoader.loadClip("http://www.w3.org/Icons/w3c_main.png", container);
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

addListener (MovieClipLoader.addListener メソッド), loadClip (MovieClipLoader.loadClip メソッド), getProgress (MovieClipLoader.getProgress メソッド)

onLoadStart (MovieClipLoader.onLoadStart イベントリスナー)

```
onLoadStart = function([target_mc:MovieClip]) {}
```

MovieClipLoader.loadClip() の呼び出しでファイルのダウンロードが開始されたときに呼び出されます。MovieClipLoader.addListener() を使用して追加するリスナーオブジェクトでこのリスナーを呼び出します。

target_mc の値は、この呼び出しが実行されるムービークリップを識別します。このパラメータは、複数のファイルが同じリスナーセットでロードされる場合に有用です。

対応バージョン： ActionScript 1.0、Flash Player 7

パラメータ

target_mc:MovieClip (オプション) - MovieClipLoader.loadClip() メソッドでロードされるムービークリップ。

例

次の例では、image_mc というムービークリップインスタンスにイメージをロードします。onLoadInit および onLoadComplete イベントを使用して、イメージのロードにどのくらい時間がかかるかを調べます。その情報は、timer_txt というテキストフィールドに表示されます。

```
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadStart = function(target_mc:MovieClip) {
  target_mc.startTimer = getTimer();
};
mclListener.onLoadComplete = function(target_mc:MovieClip) {
  target_mc.completeTimer = getTimer();
};
mclListener.onLoadInit = function(target_mc:MovieClip) {
  var timerMS:Number = target_mc.completeTimer-target_mc.startTimer;
  target_mc.createTextField("timer_txt", target_mc.getNextHighestDepth(), 0,
  target_mc._height,
target_mc._width, 22);
  target_mc.timer_txt.text = "loaded in "+timerMS+" ms.";
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  image_mc);
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

addListener (MovieClipLoader.addListener メソッド), loadClip (MovieClipLoader.loadClip メソッド), onLoadError (MovieClipLoader.onLoadError イベントリスナー), onLoadInit (MovieClipLoader.onLoadInit イベントリスナー), onLoadComplete (MovieClipLoader.onLoadComplete イベントリスナー)

removeListener (MovieClipLoader.removeListener メソッド)

public removeListener(listener:Object) : Boolean

MovieClipLoader イベントハンドラが呼び出されたときの通知を待機するリスナーを削除します。以降のロード画面は表示されません。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

listener:Object - MovieClipLoader.addListener() を使用して追加されるリスナーオブジェクト。

戻り値

Boolean - ブール値。リスナーが正常に削除された場合は true、それ以外の場合は false を返します。

例

次の例では、ムービークリップにイメージをロードし、ユーザーが start_button および stop_button という 2 つのボタンを使用してロードを開始および停止できるようにします。ロードを開始または停止すると、情報が [出力] パネルに表示されます。

```
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadStart = function(target_mc:MovieClip) {
  trace("\t onLoadStart");
};
mclListener.onLoadComplete = function(target_mc:MovieClip) {
  trace("\t onLoadComplete");
};
mclListener.onLoadError = function(target_mc:MovieClip, errorCode:String) {
  trace("\t onLoadError: "+errorCode);
};
mclListener.onLoadInit = function(target_mc:MovieClip) {
  trace("\t onLoadInit");
  start_button.enabled = true;
  stop_button.enabled = false;
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
//
start_button.clickHandler = function() {
  trace("Starting...");
  start_button.enabled = false;
  stop_button.enabled = true;
  //
  image_mcl.addListener(mclListener);
```

```
  image_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  image_mc);
};
stop_button.clickHandler = function() {
  trace("Stopping...");
  start_button.enabled = true;
  stop_button.enabled = false;
  //
  image_mcl.removeListener(mclListener);
};
stop_button.enabled = false;
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

addListener (MovieClipLoader.addListener メソッド)

unloadClip (MovieClipLoader.unloadClip メソッド)

```
public unloadClip(target:Object) : Boolean
```

MovieClipLoader.loadClip() でロードされたムービークリップを削除します。このコマンドをムービーのロード中に発行すると、MovieClipLoader.onLoadError が呼び出されます。

対応バージョン： ActionScript 1.0、Flash Player 7

パラメータ

target:Object - my_mcl.loadClip() に対応する呼び出しに渡されるストリングまたは整数。

戻り値

Boolean - ブール値。ムービークリップが正常に削除された場合は true、それ以外の場合は false を返します。

例

次の例では、image_mc というムービークリップにイメージをロードします。ムービークリップをクリックすると、そのムービークリップが削除され、情報が [出力] パネルに表示されます。

```
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  target_mc._x = 100;
  target_mc._y = 100;
  target_mc.onRelease = function() {
  trace("Unloading clip...");
```

```
  trace("\t name: "+target_mc._name);
  trace("\t url: "+target_mc._url);
  image_mcl.unloadClip(target_mc);
  };
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  image_mc);
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

loadClip (MovieClipLoader.loadClip メソッド), onLoadError (MovieClipLoader.onLoadError イベントリスナー)

NetConnection

```
Object
  |
  +-NetConnection
```

```
public dynamic class NetConnection
extends Object
```

NetConnection クラスを使用すると、ストリーミング FLV ファイルをローカルドライブまたは HTTP アドレスから再生できます。

メモ : このクラスは、Flash Media Server と組み合わせた場合には Flash Player 6 でもサポートされます。詳細については、Flash Media Server のマニュアルを参照してください。

対応バージョン : ActionScript 1.0、Flash Player 7

プロパティ一覧

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター一覧

署名	説明
NetConnection()	ローカルのストリーミングビデオファイル (FLV) を再生する際、NetStream オブジェクトと組み合わせて使用することが可能な NetConnection オブジェクトを生成します。

メソッド一覧

オプション	署名	説明
	connect(targetURI: String) : Boolean	ビデオファイル (FLV) の再生に使用する、HTTP アドレスまたはローカルファイルシステムとのローカル接続を開きます。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

connect (NetConnection.connect メソッド)

```
public connect(targetURI:String) : Boolean
```

ビデオファイル (FLV) の再生に使用する、HTTP アドレスまたはローカルファイルシステムとのローカル接続を開きます。

このメソッドを使用する場合は、Flash Player セキュリティモデルおよび次のセキュリティについての考慮事項を検討してください。

- デフォルトでは、サンドボックス間のアクセスを拒否します。クロスドメインポリシーファイルを使用することによって、Web サイトでリソースにアクセスできるようになります。
- Web サイトでは、Flash Media Server にサーバーサイド ActionScript アプリケーションロジックを追加することにより、リソースへのアクセスを拒否できます。
- Flash Player 8 では、呼び出し元 SWF ファイルがローカルファイルシステムのサンドボックスにある場合、NetConnection.connect() は使用できません。

詳細については、次の参照先を参照してください。

- 『ActionScript 2.0 の学習』の「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

対応バージョン： ActionScript 1.0、Flash Player 7 - メモ : このメソッドは、Flash Media Server と組み合わせた場合には Flash Player 6 でもサポートされます。詳細については、Flash Media Server のマニュアルを参照してください。

パラメータ

`targetURI:String` - このパラメータには `null` を渡す必要があります。

戻り値

`Boolean` - `false` の場合、接続は失敗しており使用できません。`true` の場合、`connect()` メソッドの呼び出し時に接続は失敗していませんが、成功が保証されるわけではありません。

例

次の例では、video2.flv ファイルを再生するために接続を開きます。[ライブラリ] パネルから [新規ビデオ] を選択して新しいビデオオブジェクトを作成し、`my_video` というインスタンス名を付けます。

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video2.flv");
```

関連項目

`NetStream`

NetConnection コンストラクタ

```
public NetConnection()
```

ローカルのストリーミングビデオファイル (FLV) を再生する際、NetStream オブジェクトと組み合わせて使用することが可能な NetConnection オブジェクトを生成します。NetConnection オブジェクトを生成した後に、`NetConnection.connect()` を使用して実際の接続を確立します。

Flash ドキュメント内にビデオを埋め込むよりも、外部の FLV ファイルを再生した方が、パフォーマンスとメモリ管理の効率がよくなる、ビデオと Flash のフレームレートを独立化できる、などのメリットがあります。NetConnection クラスを使用すると、ストリーミング FLV ファイルをローカルドライブまたは HTTP アドレスから再生できます。

対応バージョン： ActionScript 1.0、Flash Player 7 - メモ : このクラスは、Flash Media Server と組み合わせた場合には Flash Player 6 でもサポートされます。詳細については、Flash Media Server のマニュアルを参照してください。

例

NetConnection.connect() の例を参照してください。

関連項目

connect (NetConnection.connect メソッド), attachVideo (Video.attachVideo メソッド), NetStream

NetStream

```
Object
  |
  +-NetStream
```

```
public dynamic class NetStream
extends Object
```

NetStream クラスには、ローカルファイルシステムまたは HTTP アドレスから Flash Video (FLV) ファイルを再生するためのメソッドとプロパティがあります。NetConnection オブジェクトを使用してビデオのストリーミングを実行するには、NetStream オブジェクトを使用します。Flash ドキュメント内にビデオを埋め込むよりも、外部の FLV ファイルを再生した方が、パフォーマンスとメモリ管理の効率がよくなる、ビデオと Flash のフレームレートを独立化できる、などのメリットがあります。このクラスには、ファイルのロードおよび再生時にその進捗を追跡し、ユーザーが再生を制御 (停止、一時停止など) できるようにするための数多くのメソッドおよびプロパティが用意されています。

対応バージョン： ActionScript 1.0、Flash Player 7

プロパティ一覧

オプション	プロパティ	説明
	bufferLength: Number (読み取り専用)	バッファにデータが格納されてからの経過秒数。
	bufferTime:Number (読み取り専用)	NetStream.setBufferTime() によってバッファに割り当てられた秒数。
	bytesLoaded:Number (読み取り専用)	既に Player にロードされているデータのバイト数。

オプション	プロパティ	説明
	bytesTotal:Number (読み取り専用)	Player にロードされるファイルの総バイト数。
	checkPolicyFile: Boolean	FLV ファイル自体のロードを開始する前に、Flash Player が、ロードされる FLV ファイルのサーバーからクロスドメインポリシーファイルをダウンロードしようとするかどうかを指定します。
	currentFps: Number (読み取り専用)	1秒あたりの表示フレーム数。
	time:Number (読み取り専用)	再生ヘッドの位置 (秒単位)。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベント一覧

イベント	説明
onCuePoint = function(infoObject: Object) {}	FLV ファイルの再生中に、埋め込みキューポイントに達すると呼び出されます。
onMetaData = function(infoObject: Object) {}	再生中の FLV ファイルに埋め込まれている説明情報を Flash Player が受け取ったときに呼び出されます。
onStatus = function(infoObject: Object) {}	NetStream オブジェクトについて、ステータスが変化するたび、またはエラーが通知されるたびに呼び出されます。

コンストラクター覧

署名	説明
NetStream(connection: NetConnection)	指定された NetConnection オブジェクトを使用して、FLV ファイルを再生するためのストリームを生成します。

メソッド一覧

オプション	署名	説明
	close() : Void	ストリーム上のすべてのデータの再生を停止し、NetStream.time プロパティを 0 に設定して、他のユーザーがストリームにアクセスできるようにします。
	pause([flag: Boolean]) : Void	ストリームの再生を一時停止または再開します。
	play(name:Object, start:Number, len:Number, reset:Object) : Void	外部ビデオファイル (FLV) の再生を開始します。
	seek(offset: Number) : Void	ストリームの先頭から、指定された秒数に最も近いキーフレームをシークします。
	setBufferTime (bufferTime: Number) : Void	ストリームの表示を開始するまでにメッセージをどの程度の時間バッファリングしておくかを指定します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

bufferLength (NetStream.bufferLength プロパティ)

public bufferLength : Number (読み取り専用)

バッファにデータが格納されてからの経過秒数。このプロパティを NetStream.bufferTime と組み合わせることにより、バッファが満たされるまでの推定時間を見積もることができます。たとえば、バッファにデータがロードされるのを待つユーザーに対してフィードバック情報を提供することが可能になります。

対応バージョン : ActionScript 1.0、Flash Player 7 - メモ : このプロパティは、Flash Media Server と組み合わせた場合には Flash Player 6 でもサポートされます。詳細については、Flash Media Server のマニュアルを参照してください。

例

次の例では、テキストフィールドをダイナミックに作成し、バッファにデータが格納されてからの経過秒数に関する情報を表示します。このテキストフィールドは、ビデオが設定されているバッファの長さ、バッファが満たされている割合も表示します。

```
this.createTextField("buffer_txt", this.getNextHighestDepth(), 10, 10, 300, 22);
buffer_txt.html = true;

var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
stream_ns.setBufferTime(3);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");

var buffer_interval:Number = setInterval(checkBufferTime, 100, stream_ns);
function checkBufferTime(my_ns:NetStream):Void {
  var bufferPct:Number = Math.min(Math.round(my_ns.bufferLength/
  my_ns.bufferTime*100), 100);
  var output_str:String = "<textformat tabStops='[100,200]'>";
  output_str += "Length: "+my_ns.bufferLength+"\t"+"Time:
  "+my_ns.bufferTime+"\t"+"Buffer:"+bufferPct+"%";
  output_str += "</textformat>";
  buffer_txt.htmlText = output_str;
}
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

関連項目

bufferTime (NetStream.bufferTime プロパティ),bytesLoaded (NetStream.bytesLoaded プロパティ)

bufferTime (NetStream.bufferTime プロパティ)

public bufferTime : Number (読み取り専用)

NetStream.setBufferTime( ) によってバッファに割り当てられた秒数。デフォルト値は 0.1 (1/10 秒) です。バッファ内のデータの経過秒数を調べるには、NetStream.bufferLength を使用します。

対応バージョン : ActionScript 1.0、Flash Player 7 - メモ : このプロパティは、Flash Media Server と組み合わせた場合には Flash Player 6 でもサポートされます。詳細については、Flash Media Server のマニュアルを参照してください。

例

次の例では、テキストフィールドをダイナミックに作成し、バッファにデータが格納されてからの経過秒数に関する情報を表示します。このテキストフィールドは、ビデオが設定されているバッファの長さ、バッファが満たされている割合も表示します。

```
this.createTextField("buffer_txt", this.getNextHighestDepth(), 10, 10, 300, 22);
buffer_txt.html = true;

var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
stream_ns.setBufferTime(3);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");

var buffer_interval:Number = setInterval(checkBufferTime, 100, stream_ns);
function checkBufferTime(my_ns:NetStream):Void {
  var bufferPct:Number = Math.min(Math.round(my_ns.bufferLength/
  my_ns.bufferTime*100), 100);
  var output_str:String = "<textformat tabStops='[100,200]'>";
  output_str += "Length: "+my_ns.bufferLength+"\t"+"Time:
  "+my_ns.bufferTime+"\t"+"Buffer:"+bufferPct+"%";
  output_str += "</textformat>";
  buffer_txt.htmlText = output_str;
}
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth( ) メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

関連項目

setBufferTime (NetStream.setBufferTime メソッド), time (NetStream.time プロパティ), bufferLength (NetStream.bufferLength プロパティ)

bytesLoaded (NetStream.bytesLoaded プロパティ)

public bytesLoaded : Number (読み取り専用)

既に Player にロードされているデータのバイト数。このメソッドを NetStream.bytesTotal と組み合わせることにより、バッファが満たされるまでの推定時間を見積もることができます。たとえば、バッファにデータがロードされるのを待つユーザーに対してフィードバック情報を提供することが可能になります。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例では、Drawing API と bytesLoaded および bytesTotal プロパティを使用してプログレスバーを作成します。これには、"video1.flv" が my_video というビデオオブジェクトインスタンスにロードされる進行状況を表示します。loaded_txt というテキストフィールドがダイナミックに作成され、ロードの進行状況に関する情報を表示します。

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");

this.createTextField("loaded_txt", this.getNextHighestDepth(), 10, 10, 160, 22);
this.createEmptyMovieClip("progressBar_mc", this.getNextHighestDepth());
progressBar_mc.createEmptyMovieClip("bar_mc",
  progressBar_mc.getNextHighestDepth());
with (progressBar_mc.bar_mc) {
  beginFill(0xFF0000);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 10);
  lineTo(0, 10);
  lineTo(0, 0);
  endFill();
  _xscale = 0;
}
progressBar_mc.createEmptyMovieClip("stroke_mc",
  progressBar_mc.getNextHighestDepth());
with (progressBar_mc.stroke_mc) {
  lineStyle(0, 0x000000);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 10);
  lineTo(0, 10);
  lineTo(0, 0);
}
```

```
var loaded_interval:Number = setInterval(checkBytesLoaded, 500, stream_ns);
function checkBytesLoaded(my_ns:NetStream) {
  var pctLoaded:Number = Math.round(my_ns.bytesLoaded/my_ns.bytesTotal*100);
  loaded_txt.text = Math.round(my_ns.bytesLoaded/1000)+" of
  "+Math.round(my_ns.bytesTotal/1000)+" KB loaded ("+pctLoaded+"%)";
  progressBar_mc.bar_mc._xscale = pctLoaded;
  if (pctLoaded>=100) {
  clearInterval(loaded_interval);
  }
}
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

関連項目

bytesTotal (NetStream.bytesTotal プロパティ), bufferLength (NetStream.bufferLength プロパティ)

bytesTotal (NetStream.bytesTotal プロパティ)

public bytesTotal : Number (読み取り専用)

Player にロードされるファイルの総バイト数。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例では、Drawing API と bytesLoaded および bytesTotal プロパティを使用してプログレスバーを作成します。これには、"video1.flv" が my_video というビデオオブジェクトインスタンスにロードされる進行状況を表示します。loaded_txt というテキストフィールドがダイナミックに作成され、ロードの進行状況に関する情報を表示します。

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");

this.createTextField("loaded_txt", this.getNextHighestDepth(), 10, 10, 160, 22);
this.createEmptyMovieClip("progressBar_mc", this.getNextHighestDepth());
progressBar_mc.createEmptyMovieClip("bar_mc",
  progressBar_mc.getNextHighestDepth());
```

```
with (progressBar_mc.bar_mc) {
  beginFill(0xFF0000);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 10);
  lineTo(0, 10);
  lineTo(0, 0);
  endFill();
  _xscale = 0;
}
progressBar_mc.createEmptyMovieClip("stroke_mc",
  progressBar_mc.getNextHighestDepth());
with (progressBar_mc.stroke_mc) {
  lineStyle(0, 0x000000);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 10);
  lineTo(0, 10);
  lineTo(0, 0);
}

var loaded_interval:Number = setInterval(checkBytesLoaded, 500, stream_ns);
function checkBytesLoaded(my_ns:NetStream) {
  var pctLoaded:Number = Math.round(my_ns.bytesLoaded/my_ns.bytesTotal*100);
  loaded_txt.text = Math.round(my_ns.bytesLoaded/1000)+" of
  "+Math.round(my_ns.bytesTotal/1000)+" KB loaded ("+pctLoaded+"%)";
  progressBar_mc.bar_mc._xscale = pctLoaded;
  if (pctLoaded>=100) {
  clearInterval(loaded_interval);
  }
}
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

関連項目

bytesLoaded (NetStream.bytesLoaded プロパティ), bufferTime (NetStream.bufferTime プロパティ)

checkPolicyFile (NetStream.checkPolicyFile プロパティ)

public checkPolicyFile : Boolean

FLV ファイル自体のロードを開始する前に、Flash Player が、ロードされる FLV ファイルのサーバーからクロスドメインポリシーファイルをダウンロードしようとするかどうかを指定します。このフラグは、プログレッシブビデオダウンロード (スタンドアローン FLV ファイル) に NetStream を使用する場合に適用され、NetStream を使用して RTMP アセットを取得する場合は適用されません。

FLV ビデオファイルを呼び出し元 SWF ファイル自体のドメイン外からロードし、そのイメージのコンテンツに BitmapData.draw() を使用してアクセスする予定の場合に、このフラグを true に設定します。ロード時に checkPolicyFile を指定せずにこの操作を試行すると、必要なポリシーファイルがまだダウンロードされていないために、セキュリティエラーになる場合があります。

checkPolicyFile を true に設定して NetStream.play() を呼び出す場合、関連するクロスドメインポリシーファイルを正常にダウンロードするか、そのようなポリシーファイルが存在しないことがわかるまで、Flash Player は play() の呼び出しで指定されたオブジェクトのダウンロードを開始しません。Flash Player では、最初に既にダウンロードされているポリシーファイルが考慮され、次に System.security.loadPolicyFile() の呼び出しで指定された保留されているポリシーファイルのダウンロードが試行されます。次に、play() に渡した URL に対応するデフォルトの場所 (その URL と同じサーバーの /crossdomain.xml) からのポリシーファイルのダウンロードが試行されます。どの場合でも、該当するポリシーファイルがそのサーバー上に存在していることが必要です。また、そのポリシーファイルでは、ポリシーファイルの場所として play() に渡した URL にあるオブジェクトにアクセスできるようになっており、呼び出し元 SWF のドメインからのアクセスが <allow-access-from> タグで許可されていることが必要です。

checkPolicyFile を true に設定した場合、Flash Player ではポリシーファイルが完了するまで、play() で指定する主要なダウンロードの開始が待機されます。したがって、必要なポリシーファイルが存在していれば、NetStream から onMetaData または onStatus イベントを受け取ると直ちにポリシーファイルのダウンロードが完了し、ポリシーファイルを必要とする操作の実行を安全に開始できます。

checkPolicyFile を true に設定し、関連するポリシーファイルが見つからない場合、SecurityError 例外がスローされる操作を試行するまで、エラーは通知されません。

ロードするビデオへのピクセルレベルのアクセスを必要としない場合は、checkPolicyFile を true に設定しないようにしてください。この場合、ポリシーファイルの確認は時間の浪費になります。ダウンロードの開始が遅れ、ネットワーク帯域幅を不必要に消費する場合があるためです。

サーバーサイド HTTP リダイレクトを使用する可能性がある URL から FLV ファイルをダウンロードする場合は、checkPolicyFile に注意してください。Flash Player は、常に NetStream.play() で指定する初期 URL に対応するポリシーファイルを取得しようとします。最終的な FLV ファイルが HTTP リダイレクトによって別の URL から取得される場合は、最初にダウンロードされたポリシーファイルは FLV ファイルの最終的な URL に適用できないことがあります。この URL はセキュリティ判定において重要です。

対応バージョン： ActionScript 2.0、Flash Player 9

close (NetStream.close メソッド)

```
public close() : Void
```

ストリーム上のすべてのデータの再生を停止し、NetStream.time プロパティを 0 に設定して、他のユーザーがストリームにアクセスできるようにします。また、このコマンドは、HTTP を使用してダウンロードされた FLV ファイルのローカルコピーを削除します。Flash Player では、Flash Player で作成した FLV ファイルのローカルコピーを削除しますが、ビデオのコピーがブラウザのキャッシュディレクトリに残る場合があります。FLV ファイルをキャッシュやローカル記憶域に一切残さないようにする必要がある場合には、Flash Media Server を使用してください。

対応バージョン： ActionScript 1.0、Flash Player 7 - メモ : このメソッドは、Flash Media Server と組み合わせた場合には Flash Player 6 でもサポートされます。詳細については、Flash Media Server のマニュアルを参照してください。

例

次の onDisconnect() 関数は接続を閉じ、close_btn というボタンをクリックすると、ローカルディスクに格納された video1.flv の一時コピーを削除します。

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");

close_btn.onRelease = function(){
  stream_ns.close();
};
```

関連項目

pause (NetStream.pause メソッド),play (NetStream.play メソッド)

currentFps (NetStream.currentFps プロパティ)

`public currentFps : Number` (読み取り専用)

1 秒あたりの表示フレーム数。複数のシステム上で再生できるように FLV ファイルを書き出す場合、テスト中にこの値をチェックすることで、ファイルの書き出し時にどの程度の圧縮が必要かを見極めることができます。

対応バージョン : ActionScript 1.0、Flash Player 7 - メモ : このプロパティは、Flash Media Server と組み合わせた場合には Flash Player 6 でもサポートされます。詳細については、Flash Media Server のマニュアルを参照してください。

例

次の例では、テキストフィールドをダイナミックに作成し、video1.flv が 1 秒あたりに表示する現在のフレーム数を表示します。

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");

this.createTextField("fps_txt", this.getNextHighestDepth(), 10, 10, 50, 22);
fps_txt.autoSize = true;
var fps_interval:Number = setInterval(displayFPS, 500, stream_ns);
function displayFPS(my_ns:NetStream) {
  fps_txt.text = "currentFps (frames per second):
  "+Math.floor(my_ns.currentFps);
}
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している `MovieClip.getNextHighestDepth()` メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

NetStream コンストラクタ

`public NetStream(connection:NetConnection)`

指定された NetConnection オブジェクトを使用して、FLV ファイルを再生するためのストリームを生成します。

対応バージョン : ActionScript 1.0、Flash Player 7 - メモ : このクラスは、Flash Media Server と組み合わせた場合には Flash Player 6 でもサポートされます。詳細については、Flash Media Server のマニュアルを参照してください。

パラメータ

`connection:NetConnection` - NetConnection オブジェクト。

例

次のコードは、まず新しい NetConnection オブジェクト connection_nc を生成し、さらに、connection_nc を使用して stream_ns という新しい NetStream オブジェクトを生成しています。[ライブラリ] オプションメニューから [新規ビデオ] を選択して新しいビデオオブジェクトインスタンスを作成し、my_video というインスタンス名を付けます。

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");
```

関連項目

NetConnection, attachVideo (Video.attachVideo メソッド)

onCuePoint (NetStream.onCuePoint ハンドラ)

```
onCuePoint = function(infoObject:Object) {}
```

FLV ファイルの再生中に、埋め込みキューポイントに達すると呼び出されます。このハンドラを使用して、ビデオが特定のキューポイントに達したときにコード内のアクションをトリガすることができます。これにより、アプリケーションの他のアクションとビデオ再生イベントを同期させることができます。

FLV ファイルに埋め込み可能なキューポイントには、2 つのタイプがあります。

- " ナビゲーション " キューポイントは FLV ファイル内のキーフレームと、その厳密に一致するキーフレームに対応するキューポイントの time プロパティを指定します。ナビゲーションキューポイントは、ユーザーをビデオファイルに移動させるブックマークやエントリポイントとしてよく使われます。
- " イベント " キューポイントは、時間で指定します。その時間が特定のキーフレームに対応しているかどうかは関係ありません。イベントキューポイントは、通常ビデオ内で何か処理が行われるときの時間を表し、他のアプリケーションイベントのトリガに使用できます。

onCuePoint() イベントハンドラは、次のプロパティを持つオブジェクトを受け取ります。

プロパティ	説明
name	キューポイントが FLV ファイル内に埋め込まれたときに、キューポイントに指定された名前です。
time	ビデオファイルの再生時にキューポイントが発生した時間 (秒数) です。
type	到達したキューポイントの種類です。" ナビゲーション " または " イベント " のいずれかです。
パラメータ	このキューポイントに指定された名前と値のペアのストリングの結合配列です。パラメータ名または値には、任意の有効なストリングを使用できます。

最初に FLV ファイルをエンコードするとき、または Flash オーサリングツールでビデオの読み込みウィザードを使用してビデオクリップを読み込むときに、FLV ファイルにキューポイントを定義できます。

onMetaData() イベントハンドラも、ビデオファイル内のキューポイントに関する情報を取得します。ただし、onMetaData() イベントハンドラは、ビデオの再生が開始される前に、すべてのキューポイントに関する情報を取得します。onCuePoint() イベントハンドラは、再生時にそのキューポイントに指定された時間に、1つのキューポイントに関する情報を取得します。

一般に、コードで特定のキューポイントの発生時に応答するには、onCuePoint() イベントハンドラを使用して、コード内の特定のアクションをトリガします。

onMetaData() イベントハンドラに指定されたキューポイントのリストを使用すると、ビデオストリームの事前に定義されたポイントでユーザーにビデオの再生を開始させることができます。キューポイントの time プロパティの値を NetStream.seek() メソッドに渡して、そのキューポイントからビデオを再生します。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

infoObject:Object - キューポイントの name、time、type、および parameters を含むオブジェクトです。

例

この例で使用するコードは、新しい NetConnection オブジェクトおよび NetStream オブジェクトを作成することにより開始します。次に、コードは NetStream オブジェクトの onCuePoint() ハンドラを定義します。このハンドラは、infoObject オブジェクト内の名前の付いた各プロパティを循環し、プロパティの名前と値を出力します。parameters という名前のプロパティが見つかると、リストの各パラメータ名を循環してパラメータ名と値を出力します。

```
var nc:NetConnection = new NetConnection();
nc.connect(null);
var ns:NetStream = new NetStream(nc);

ns.onCuePoint = function(infoObject:Object)
{
  trace("onCuePoint:");
  for (var propName:String in infoObject) {
    if (propName != "parameters")
    {
      trace(propName + " = " + infoObject[propName]);
    }
    else
    {
      trace("parameters =");
      if (infoObject.parameters != undefined) {
        for (var paramName:String in infoObject.parameters)
        {
          trace(" " + paramName + ": " + infoObject.parameters[paramName]);
        }
      }
      else
      {
        trace("undefined");
      }
    }
  }
  trace("---------");
}

ns.play("http://www.helpexamples.com/flash/video/cuepoints.flv");
```

これにより、次の情報が表示されます。

```
onCuePoint:
parameters =
lights: beginning
type = navigation
time = 0.418
name = point1
---------
onCuePoint:
parameters =
```

```
lights: middle
type = navigation
time = 7.748
name = point2
---------
onCuePoint:
parameters =
lights: end
type = navigation
time = 16.02
name = point3
---------
```

パラメータ名 "lights" は、この例のビデオの作成者が付けた名前です。キューポイントのパラメータには、任意の名前を付けることができます。

関連項目

onMetaData (NetStream.onMetaData ハンドラ)

onMetaData (NetStream.onMetaData ハンドラ)

```
onMetaData = function(infoObject:Object) {}
```

再生中の FLV ファイルに埋め込まれている説明情報を Flash Player が受け取ったときに呼び出されます。

Flash Video Exporter ユーティリティ (バージョン 1.1 以降) は、ビデオの継続時間、作成日付、データレート、その他の情報をビデオファイルに埋め込みます。各ビデオエンコーダは、それぞれ異なるメタデータのセットを埋め込みます。

このハンドラは、NetStream.play() メソッドの呼び出し後、ただしビデオ再生ヘッドが進むよりは前にトリガされます。

多くの場合、FLV メタデータに埋め込まれた継続期間の値は実際の継続時間に近似したものになりますが、正確な値ではありません。つまり、再生ヘッドがビデオストリームの末尾にある場合、FLV メタデータに埋め込まれた継続期間の値は NetStream.time プロパティの値と必ずしも一致するわけではありません。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

infoObject:Object - メタデータアイテムごとに 1 つのプロパティを含むオブジェクト。

例

この例で使用するコードは、新しい NetConnection オブジェクトおよび NetStream オブジェクトを作成することにより開始します。次に、コードは NetStream オブジェクトの onMetaData() ハンドラを定義します。このハンドラは、infoObject オブジェクト内の名前の付いた各プロパティを循環し、プロパティの名前と値を出力します。

```
var nc:NetConnection = new NetConnection();
nc.connect(null);
var ns:NetStream = new NetStream(nc);

ns.onMetaData = function(infoObject:Object) {
  for (var propName:String in infoObject) {
    trace(propName + " = " + infoObject[propName]);
  }
};

ns.play("http://www.helpexamples.com/flash/video/water.flv");
```

これにより、次の情報が表示されます。

```
canSeekToEnd = true
 videocodecid = 4
 framerate = 15
 videodatarate = 400
 height = 215
 width = 320
 duration = 7.347
```

プロパティの一覧は、FLV ファイルをエンコードするために使用したソフトウェアによって異なります。

関連項目

time (NetStream.time プロパティ), play (NetStream.play メソッド), NetConnection

onStatus (NetStream.onStatus ハンドラ)

```
onStatus = function(infoObject:Object) {}
```

NetStream オブジェクトについて、ステータスが変化するたび、またはエラーが通知されるたびに呼び出されます。このイベントハンドラに応答するには、情報オブジェクトを処理する関数を作成する必要があります。

情報オブジェクトには、onStatus ハンドラの結果ストリングを含む code プロパティと、status または error のいずれかのストリングを含む level プロパティがあります。

この onStatus ハンドラ以外に、Flash には System.onStatus という " スーパー " 関数もあります。特定のオブジェクトに対して onStatus が呼び出され、それに応答する関数が割り当てられていない場合、System.onStatus に割り当てられた関数があれば、その関数が実行されます。

特定の NetStream アクティビティが発生した場合、次のイベントによって通知されます。

code プロパティ	level プロパティ	説明
NetStream.Buffer.Empty	status	データを十分な速度で受信していないので、バッファを満たしていません。バッファが再び満たされるまでデータのフローは中断されます。バッファが満たされた時点で NetStream.Buffer.Full メッセージが送信され、再びストリームの再生が開始されます。
NetStream.Buffer.Full	status	バッファが満たされ、ストリームの再生が開始されます。
NetStream.Buffer.Flush	status	データのストリームが終了しました。残りのバッファは空になります。
NetStream.Play.Start	status	再生が開始されました。
NetStream.Play.Stop	status	再生が停止されました。
NetStream.Play.StreamNotFound	error	play() メソッドに渡した FLV が見つかりません。
NetStream.Seek.InvalidTime	error	プログレッシブダウンロードでダウンロードされたビデオに対して、ユーザーが現時点でダウンロード済みのビデオデータの末尾を超えて再生またはシークしようとしました。または、ファイル全体のダウンロード後にビデオの末尾を超えて再生またはシークしようとしました。Error.message.details プロパティには、ユーザーがシークできる有効な末尾を示す時間コードが含まれています。Error.message プロパティを参照してください。
NetStream.Seek.Notify	status	シーク操作が完了しました。

バッファに関するエラーが繰り返し表示される場合は、NetStream.setBufferTime() メソッドを使用してバッファを変更してみてください。

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

infoObject:Object - ステータスメッセージまたはエラーメッセージに従って定義されるパラメータ。

例

次の例では、ストリームに関するデータを [出力] パネルに表示します。

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");
stream_ns.onStatus = function(infoObject:Object) {
     trace("NetStream.onStatus called: ("+getTimer()+" ms)");
     for (var prop in infoObject) {
        trace("\t"+prop+":\t"+infoObject[prop]);
     }
     trace("");
};
```

関連項目

setBufferTime (NetStream.setBufferTime メソッド),onStatus (System.onStatus ハンドラ)

pause (NetStream.pause メソッド)

```
public pause([flag:Boolean]) : Void
```

ストリームの再生を一時停止または再開します。

このメソッドを (パラメータを送信せずに) 呼び出すと最初は再生を一時停止し、次に呼び出したときには再生を再開します。このメソッドをボタンに追加して、ユーザーがボタン操作で再生を一時停止または再生できるようにしたい場合があります。

対応バージョン : ActionScript 1.0、Flash Player 7 - メモ : このメソッドは、Flash Media Server と組み合わせた場合には Flash Player 6 でもサポートされます。詳細については、Flash Media Server のマニュアルを参照してください。

パラメータ

`flag:Boolean` (オプション) - 再生を一時停止するか (`true`)、再生を再開するか (`false`) を指定するブール値。このパラメータを省略すると、`NetStream.pause()` によって動作が交互に切り替わります。指定されたストリームに対して最初に呼び出されたときには再生を一時停止し、次に呼び出されたときには再生を再開します。

例

このメソッドの使用例を次に示します。

```
my_ns.pause(); // pauses play first time issued
my_ns.pause(); // resumes play
my_ns.pause(false); // no effect, play continues
my_ns.pause(); // pauses play
```

関連項目

close (NetStream.close メソッド),play (NetStream.play メソッド)

play (NetStream.play メソッド)

```
public play(name:Object, start:Number, len:Number, reset:Object) : Void
```

外部ビデオファイル (FLV) の再生を開始します。ビデオデータを表示するには、Video.attachVideo() メソッドを呼び出します。ビデオと一緒にストリームされるオーディオ、またはオーディオだけが含まれた FLV ファイルは自動的に再生されます。

FLV ファイルに関連付けられたオーディオを制御するには、MovieClip.attachAudio() を使用して、ムービークリップにオーディオをアタッチします。その後、Sound オブジェクトを作成することにより、オーディオのいくつかの特性を制御できます。詳細については、MovieClip.attachAudio() を参照してください。

FLV ファイルが見つからなかった場合は、NetStream.onStatus イベントハンドラが呼び出されます。再生中のストリームを停止させるには、NetStream.close() を使用します。

SWF ファイルと同じディレクトリまたはサブディレクトリに格納されたローカルの FLV ファイルを再生できますが、上位のディレクトリを参照することはできません。たとえば、SWF ファイルが "/training" というディレクトリに格納されており、"/training/videos" ディレクトリに格納されたビデオを再生する場合は、次のシンタックスを使用します。

```
my_ns.play("videos/videoName.flv");
```

"/training" ディレクトリに格納されたビデオを再生するには、次のシンタックスを使用します。

```
my_ns.play("videoName.flv");
```

このメソッドを使用するときは、Flash Player セキュリティモデルを考慮してください。

Flash Player 8 :

- 呼び出し元 SWF ファイルが ローカルファイルシステムのサンドボックスにあり、ターゲットリソースがローカル以外のサンドボックスにある場合、NetStream.play() は使用できません。
- 信頼できるローカルのサンドボックスまたはネットワーク接続したローカルのサンドボックスからネットワーク上のサンドボックスにアクセスするには、クロスドメインポリシーファイルを使用して Web サイトで許可する必要があります。

詳細については、次の参照先を参照してください。

- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

対応バージョン： ActionScript 1.0、Flash Player 7 - メモ : このメソッドは、Flash Media Server と組み合わせた場合には Flash Player 6 でもサポートされます。詳細については、Flash Media Server のマニュアルを参照してください。

パラメータ

`name:Object` - 再生する FLV ファイルの名前 (二重引用符で囲む)。http:// および file:// の両方の形式を使用できます。file:// のパスは SWF ファイルの相対パスにする必要があります。

`start:Number` -

`len:Number` -

`reset:Object` -

例

次に、`NetStream.play()` コマンドの使用例をいくつか紹介します。ユーザーのコンピュータにあるファイルを再生できます。"joe_user" ディレクトリは、SWF が格納されているディレクトリのサブディレクトリです。ファイルをサーバー上で再生できます。

```
// Play a file that is on the user's computer.
my_ns.play("file://joe_user/flash/videos/lectureJune26.flv");

// Play a file on a server.
my_ns.play("http://someServer.someDomain.com/flash/video/orientation.flv");
```

関連項目

`attachAudio (MovieClip.attachAudio メソッド)`, `close (NetStream.close メソッド)`, `onStatus (NetStream.onStatus ハンドラ)`, `pause (NetStream.pause メソッド)`, `attachVideo (Video.attachVideo メソッド)`

seek (NetStream.seek メソッド)

```
public seek(offset:Number) : Void
```

ストリームの先頭から、指定された秒数に最も近いキーフレームをシークします。ストリームの指定位置に到達すると再生が再開されます。

対応バージョン : ActionScript 1.0、Flash Player 7 - メモ : このメソッドは、Flash Media Server と組み合わせた場合には Flash Player 6 でもサポートされます。詳細については、Flash Media Server のマニュアルを参照してください。

パラメータ

`offset:Number` - FLV ファイルにおける、再生ヘッドのおおよその移動時間 (秒単位)。`offset` に最も近いキーフレームに再生ヘッドを移動します。

- ストリームの先頭に移動するには、`offset` に 0 を指定します。
- ストリームの先頭から前方にシークするには、進めたい秒数をパラメータに指定します。たとえば、*my_ns*`.seek(15)` と指定すると、再生ヘッドが先頭から 15 秒の位置に移動します。
- 現在位置を基準にしてシークするには、*my_ns* `.time + n` または *my_ns* `.time - n` を指定すると、現在位置を基準にしてそれぞれ *n* 秒分、前方または後方にシークされます。たとえば、現在位置から 20 秒巻き戻すには、`my_ns.seek(my_ns.time - 20)`. のように指定します。

ビデオがシークする正確な位置は、書き出し時の 1 秒あたりのフレーム数の設定によって異なります。したがって、同じビデオが 6 fps および 30 fps で書き出された場合に、両方のビデオオブジェクトに `my_ns.seek(15)` を使用すると、2 つの異なる位置がシークされます。

例

次に、`NetStream.seek()` コマンドの使用例をいくつか紹介します。ストリームの先頭に戻ること、ストリームの先頭から 30 秒の位置に移動すること、および現在の位置から 3 分前に巻き戻すことができます。

```
// Return to the beginning of the stream
my_ns.seek(0);

// Move to a location 30 seconds from the beginning of the stream
my_ns.seek(30);

// Move backwards three minutes from current location
my_ns.seek(my_ns.time - 180);
```

関連項目

play (NetStream.play メソッド),time (NetStream.time プロパティ)

setBufferTime (NetStream.setBufferTime メソッド)

public setBufferTime(bufferTime:Number) : Void

ストリームの表示を開始するまでにメッセージをどの程度の時間バッファリングしておくかを指定します。たとえば、ストリームの最初の 15 秒間を中断することなく再生するには、bufferTime を 15 に設定します。この場合、Flash はデータを 15 秒間バッファリングしてからストリームの再生を開始します。

対応バージョン : ActionScript 1.0、Flash Player 7 - メモ : このメソッドは、Flash Media Server と組み合わせた場合には Flash Player 6 でもサポートされます。詳細については、Flash Media Server のマニュアルを参照してください。

パラメータ

bufferTime:Number - Flash がデータの表示を開始するまでにデータをバッファリングする秒数。デフォルト値は 0.1 (1/10 秒) です。

例

NetStream.bufferLength の例を参照してください。

関連項目

bufferLength (NetStream.bufferLength プロパティ), bufferTime (NetStream.bufferTime プロパティ)

time (NetStream.time プロパティ)

public time : Number (読み取り専用)

再生ヘッドの位置 (秒単位)。

対応バージョン : ActionScript 1.0、Flash Player 7 - メモ : このプロパティは、Flash Media Server と組み合わせた場合には Flash Player 6 でもサポートされます。詳細については、Flash Media Server のマニュアルを参照してください。

例

次の例では、ダイナミックに作成された time_txt というテキストフィールドに、再生ヘッドの現在の位置を表示します。[ライブラリ] オプションメニューから [新規ビデオ] を選択して新しいビデオオブジェクトインスタンスを作成し、my_video というインスタンス名を付けます。my_video という新しいビデオオブジェクトを作成します。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");
//
stream_ns.onStatus = function(infoObject:Object) {
  statusCode_txt.text = infoObject.code;
};

this.createTextField("time_txt", this.getNextHighestDepth(), 10, 10, 100, 22);
time_txt.text = "LOADING";

var time_interval:Number = setInterval(checkTime, 500, stream_ns);
function checkTime(my_ns:NetStream) {
  var ns_seconds:Number = my_ns.time;
  var minutes:Number = Math.floor(ns_seconds/60);
  var seconds = Math.floor(ns_seconds%60);
  if (seconds<10) {
  seconds = "0"+seconds;
  }
  time_txt.text = minutes+":"+seconds;
}
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

関連項目

bufferLength (NetStream.bufferLength プロパティ), bytesLoaded (NetStream.bytesLoaded プロパティ)

数値 (Number)

```
Object
  |
  +-Number

public class Number
extends Object
```

Number クラスは、Number データ型の単純なラッパーオブジェクトです。Number クラスに関連するメソッドとプロパティを使用してプリミティブな数値を処理することができます。このクラスは、JavaScript の Number クラスと同じです。

Number クラスのプロパティは静的であるため、プロパティを使用するためのオブジェクトは不要で、コンストラクタを使用する必要はありません。

次の例では、Number クラスの toString() メソッドを呼び出します。このメソッドはストリング 1234 を返します。

```
var myNumber:Number = new Number(1234);
myNumber.toString();
```

次の例では、コンストラクタを使用せずに、MIN_VALUE プロパティの値を宣言された変数に割り当てます。

```
var smallest:Number = Number.MIN_VALUE;
```

対応バージョン： ActionScript 1.0、Flash Player 5 - Flash Player 6 ではネイティブオブジェクトになり、パフォーマンスが大幅に向上しました。

プロパティー覧

オプション	プロパティ	説明
static	MAX_VALUE:Number	表現可能な最大の数値 (倍精度 IEEE-754)。
static	MIN_VALUE:Number	表現可能な負以外の最小数値 (倍精度 IEEE-754) です。
static	NaN:Number	非数 (NaN) を表す IEEE-754 の値。
static	NEGATIVE_INFINITY: Number	負の無限大を表す IEEE-754 値を指定します。
static	POSITIVE_INFINITY: Number	正の無限大を表す IEEE-754 値を指定します。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター覧

署名	説明
Number(num:Object)	新しい Number オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	toString(radix: Number) : String	指定された Number オブジェクト (*myNumber*) のストリング表現を返します。
	valueOf() : Number	指定された Number オブジェクトのプリミティブな値のタイプを返します。

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

MAX_VALUE (Number.MAX_VALUE プロパティ)

public static MAX_VALUE : Number

表現可能な最大の数値 (倍精度 IEEE-754)。この数値は、約 1.79e+308 です。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の ActionScript は、表現可能な最大の数値および最小の数値を [出力] パネルに表示します。

```
trace("Number.MIN_VALUE = "+Number.MIN_VALUE);
trace("Number.MAX_VALUE = "+Number.MAX_VALUE);
```

このコードは、次の値を表示します。

```
Number.MIN_VALUE = 4.94065645841247e-324
Number.MAX_VALUE = 1.79769313486232e+308
```

MIN_VALUE (Number.MIN_VALUE プロパティ)

public static MIN_VALUE : Number

表現可能な負以外の最小数値 (倍精度 IEEE-754) です。この数値は、約 5e-324 です。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の ActionScript は、表現可能な最大の数値および最小の数値を [出力] パネルに表示します。

```
trace("Number.MIN_VALUE = "+Number.MIN_VALUE);
trace("Number.MAX_VALUE = "+Number.MAX_VALUE);
```

このコードは、次の値を表示します。

```
Number.MIN_VALUE = 4.94065645841247e-324
Number.MAX_VALUE = 1.79769313486232e+308
```

NaN (Number.NaN プロパティ)

```
public static NaN : Number
```

非数 (NaN) を表す IEEE-754 の値。

対応バージョン : ActionScript 1.0、Flash Player 5

関連項目

isNaN 関数

NEGATIVE_INFINITY (Number.NEGATIVE_INFINITY プロパティ)

```
public static NEGATIVE_INFINITY : Number
```

負の無限大を表す IEEE-754 値を指定します。このプロパティの値は、-Infinity 定数の値と同じです。

負の無限大は、数学演算または関数が表現できる下限を超える負の値を返すときに返される特別な数値です。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、以下の値の除算結果を比較します。

```
var posResult:Number = 1/0;
if (posResult == Number.POSITIVE_INFINITY) {
  trace("posResult = "+posResult); // output: posResult = Infinity
}
var negResult:Number = -1/0;
if (negResult == Number.NEGATIVE_INFINITY) {
  trace("negResult = "+negResult); // output: negResult = -Infinity
```

Number コンストラクタ

```
public Number(num:Object)
```

新しい Number オブジェクトを作成します。new Number コンストラクタは、主にプレースホルダーとして使用します。Number オブジェクトは、パラメータをプリミティブ値に変換する Number() 関数とは異なります。

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

num:Object - 作成される Number オブジェクトの数値、または数値に変換される値。value が指定されなかった場合のデフォルト値は 0 です。

例

次のコードは、新しい Number オブジェクトを作成します。

```
var n1:Number = new Number(3.4);
var n2:Number = new Number(-10);
```

関連項目

toString（Number.toString メソッド）, valueOf（Number.valueOf メソッド）

POSITIVE_INFINITY (Number.POSITIVE_INFINITY プロパティ)

```
public static POSITIVE_INFINITY : Number
```

正の無限大を表す IEEE-754 値を指定します。このプロパティの値は、Infinity 定数の値と同じです。

正の無限大は、数学演算または関数が表現できる上限を超える正の値を返すときに返される特別な数値です。

対応バージョン： ActionScript 1.0、Flash Player 5

例

次の例では、以下の値の除算結果を比較します。

```
var posResult:Number = 1/0;
if (posResult == Number.POSITIVE_INFINITY) {
  trace("posResult = "+posResult); // output: posResult = Infinity
}
var negResult:Number = -1/0;
if (negResult == Number.NEGATIVE_INFINITY) {
  trace("negResult = "+negResult); // output: negResult = -Infinity
```

toString (Number.toString メソッド)

```
public toString(radix:Number) : String
```

指定された Number オブジェクト (*myNumber*) のストリング表現を返します。Number オブジェクトの値が先行ゼロを持たない小数 (.4 など) の場合、Number.toString() は先行ゼロを追加します。結果は 0.4 となります。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

radix:Number - 数値からストリングへの変換に使用する基数 (2 ～ 36) を指定します。radix パラメータを指定しない場合、デフォルト値は 10 です。

戻り値

String - ストリング。

例

次の例では、radix パラメータに 2 および 8 を使用し、数値 9 に対応する表現を含むストリングを返します。

```
var myNumber:Number = new Number(9);
trace(myNumber.toString(2)); // output: 1001
trace(myNumber.toString(8)); // output: 11
```

次の例では、結果が 16 進数値になります。

```
var r:Number = new Number(250);
var g:Number = new Number(128);
var b:Number = new Number(114);
var rgb:String = "0x"+ r.toString(16)+g.toString(16)+b.toString(16);
trace(rgb);
// output: rgb:0xFA8072 (Hexadecimal equivalent of the color 'salmon')
```

valueOf (Number.valueOf メソッド)

```
public valueOf() : Number
```

指定された Number オブジェクトのプリミティブな値のタイプを返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Number - 指定された Number オブジェクト対応する数値です。

例

次の例では、numSocks オブジェクトのプリミティブな値が結果として返されます。

```
var numSocks = new Number(2);
trace(numSocks.valueOf()); // output: 2
```

Object

Object

public class **Object**

Object クラスは、ActionScript クラス階層のルートにあります。このクラスは、JavaScript Object クラスで提供される機能の小さいサブセットから構成されます。

対応バージョン： ActionScript 1.0、Flash Player 5 - Flash Player 6 ではネイティブオブジェクトになり、パフォーマンスが大幅に向上しました。

プロパティー覧

オプション	プロパティ	説明
	constructor:Object	特定のオブジェクトインスタンスのコンストラクタ関数を参照します。
	__proto__:Object	オブジェクトの作成に使用された、クラスの prototype プロパティ (ActionScript 2.0) またはコンストラクタ関数 (ActionScript 1.0) を参照します。
static	prototype:Object	クラスまたは関数オブジェクトのスーパークラスを参照します。
	__resolve:Object	ActionScript コードが未定義のメソッドまたはプロパティを参照する場合に自動的に呼び出されるユーザー定義関数への参照です。

コンストラクター覧

署名	説明
Object()	Object オブジェクトを作成し、そのオブジェクトのコンストラクタメソッドへの参照をオブジェクトの constructor プロパティに格納します。

メソッド一覧

オプション	署名	説明
	addProperty(name:String, getter:Function, setter:Function) : Boolean	getter/setter プロパティを作成します。
	hasOwnProperty(name:String) : Boolean	オブジェクトに指定されたプロパティが定義されているかどうかを示します。
	isPropertyEnumerable(name:String) : Boolean	指定されたプロパティが存在し列挙できるかどうかを示します。
	isPrototypeOf(theClass:Object) : Boolean	Object クラスのインスタンスが、パラメータとして指定されたオブジェクトのプロトタイプチェーン内にあるかどうかを示します。
static	registerClass(name:String, theClass:Function) : Boolean	ムービークリップシンボルと ActionScript オブジェクトクラスを関連付けます。
	toString() : String	指定されたオブジェクトをストリングに変換し、返します。
	unwatch(name:String) : Boolean	Object.watch() で作成した監視ポイントを削除します。
	valueOf() : Object	指定されたオブジェクトのプリミティブな値を返します。
	watch(name:String, callback:Function, [userData:Object]) : Boolean	ActionScript オブジェクトの指定されたプロパティの変更に応じて呼び出されるイベントハンドラを登録します。

addProperty (Object.addProperty メソッド)

```
public addProperty(name:String, getter:Function, setter:Function) : Boolean
```

getter/setter プロパティを作成します。Flash は getter/setter プロパティを読み込むと、get 関数を呼び出します。この関数の戻り値は name の値になります。Flash は getter/setter プロパティを書き込むと、set 関数を呼び出し、それに新しい値をパラメータとして渡します。指定された名前のプロパティが既に存在する場合は、新しいプロパティによって上書きされます。

get 関数はパラメータがない関数です。戻り値のタイプは不特定です。戻り値のタイプは呼び出しごとに変わることがあります。戻り値は、プロパティの現在の値として扱われます。

set 関数のパラメータは 1 つであり、プロパティの新しい値です。たとえば、プロパティ x がステートメント x = 1 によって割り当てられた場合、set 関数には Nubmer 型のパラメータ 1 が渡されます。set 関数の戻り値は無視されます。

getter/setter プロパティはプロトタイプオブジェクトに追加できます。getter/setter プロパティをプロトタイプオブジェクトに追加すると、プロトタイプオブジェクトを継承するすべてのオブジェクトインスタンスは getter/setter プロパティを継承します。つまり、1 つのプロトタイプオブジェクトに getter/setter プロパティを追加するだけで、クラスのすべてのインスタンスに同じプロパティを追加できます (プロトタイプオブジェクトにメソッドを追加するのと似ています)。継承先のプロトタイプオブジェクトの getter/setter プロパティに対して呼び出される get/set 関数には、このプロトタイプオブジェクトへの参照ではなく、継承元のオブジェクトへの参照が渡されます。

正常に呼び出されなかった場合は、Object.addProperty() が失敗し、エラーが発生することがあります。次の表は、発生する可能性があるエラーの一覧です。

エラー状態	結果
name が有効なプロパティ名ではありません。たとえば、空のストリングです。	false が返され、プロパティは追加されません。
getter は有効な関数オブジェクトではありません。	false が返され、プロパティは追加されません。
setter は有効な関数オブジェクトではありません。	false が返され、プロパティは追加されません。

対応バージョン : ActionScript 1.0、Flash Player 6 - ActionScript 2.0 クラスでは、このメソッドの代わりに get または set を使用できます。

パラメータ

name:String - 作成対象のオブジェクトプロパティの名前を示すストリング。

getter:Function - プロパティの値を取得するために呼び出される関数。このパラメータは Function オブジェクトです。

setter:Function - プロパティの値を設定するために呼び出される関数。このパラメータは Function オブジェクトです。このパラメータに値 null を渡すと、プロパティは読み取り専用になります。

戻り値

Boolean - ブール値：プロパティが正常に作成された場合は true を返します。それ以外の場合は false を返します。

例

次の例では、オブジェクトに 2 つの内部メソッド setQuantity() および getQuantity() があります。bookcount というプロパティの値を設定または取得するときに、これらのメソッドが呼び出されます。もう 1 つの getTitle() という内部メソッドは、bookname プロパティに設定されている読み取り専用の値を返します。スクリプトが myBook.bookcount の値を取得すると、ActionScript インタプリタは myBook.getQuantity() を自動的に呼び出します。スクリプトで myBook.bookcount の値を修正すると、インタプリタは myObject.setQuantity() を呼び出します。bookname プロパティでは set 関数を指定していません。したがって、bookname プロパティ値を変更する試みは無視されます。

```
function Book() {
  this.setQuantity = function(numBooks:Number):Void {
  this.books = numBooks;
  };
  this.getQuantity = function():Number {
  return this.books;
  };
  this.getTitle = function():String {
  return "Catcher in the Rye";
  };
  this.addProperty("bookcount", this.getQuantity, this.setQuantity);
  this.addProperty("bookname", this.getTitle, null);
}
var myBook = new Book();
myBook.bookcount = 5;
trace("You ordered "+myBook.bookcount+" copies of "+myBook.bookname);
// output: You ordered 5 copies of Catcher in the Rye
```

前の例は有効ですが、bookcount および bookname プロパティが Book オブジェクトの各インスタンスに追加されます。オブジェクトの各インスタンスで 2 つのプロパティが必須となります。クラスに bookcount や bookname, のようなプロパティが多数ある場合は、多大なメモリを消費します。代わりの方法として、bookcount および bookname プロパティを 1 つの場所にだけ存在させるために、プロパティを Book.prototype に追加します。このようにしても、結果は bookcount プロパティと bookname プロパティをすべてのインスタンスに直接追加したシンタックス例のコードと同じです。Book インスタンスのいずれかのプロパティにアクセスした場合に、プロパティが存在しなければ、プロトタイプチェーンを上にたどることによって Book.prototype で定義されたバージョンが見つかります。次の例では、プロパティを Book.prototype に追加する方法を示します。

```
function Book() {}
Book.prototype.setQuantity = function(numBooks:Number):Void {
  this.books = numBooks;
};
Book.prototype.getQuantity = function():Number {
  return this.books;
};
Book.prototype.getTitle = function():String {
  return "Catcher in the Rye";
};
Book.prototype.addProperty("bookcount", Book.prototype.getQuantity,
  Book.prototype.setQuantity);
Book.prototype.addProperty("bookname", Book.prototype.getTitle, null);
var myBook = new Book();
myBook.bookcount = 5;
trace("You ordered "+myBook.bookcount+" copies of "+myBook.bookname);
```

次の例では、ActionScript 2.0 で利用可能な暗黙的な getter および setter 関数を使用する方法を示しています。Book 関数を定義して Book.prototype を編集する代わりに、"Book.as" という名前の外部ファイルに Book クラスを定義します。次のコードは、"Book.as" という名前の別の外部ファイルに配置する必要があります。このファイルは、Flash アプリケーションのクラスパスに含まれ、このクラス定義のみを格納します。

```
class Book {
  var books:Number;
  function set bookcount(numBooks:Number):Void {
  this.books = numBooks;
  }
  function get bookcount():Number {
  return this.books;
  }
  function get bookname():String {
  return "Catcher in the Rye";
  }
}
```

次のコードは、FLA ファイルに配置することができ、前の例の場合と同じ機能を果します。

```
var myBook:Book = new Book();
myBook.bookcount = 5;
trace("You ordered "+myBook.bookcount+" copies of "+myBook.bookname);
```

関連項目

get ステートメント , set ステートメント

constructor (Object.constructor プロパティ)

```
public constructor : Object
```

特定のオブジェクトインスタンスのコンストラクタ関数を参照します。constructor プロパティは、Object クラスのコンストラクタを使用して作成されると、すべてのオブジェクトに自動的に割り当てられます。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例は、myObject オブジェクトのコンストラクタ関数への参照です。

```
var my_str:String = new String("sven");
trace(my_str.constructor == String); //output: true
```

instanceof オペレータを使用すると、オブジェクトが指定したクラスに属するかどうかを調べることができます。

```
var my_str:String = new String("sven");
trace(my_str instanceof String); //output: true
```

しかし、次の例では Object.constructor プロパティがプリミティブデータ型 (次のようなストリングリテラルなど) をラッパーオブジェクトに変換します。次の例のように、instanceof オペレータは変換を行いません。

```
var my_str:String = "sven";
trace(my_str.constructor == String); //output: true
trace(my_str instanceof String); //output: false
```

関連項目

instanceof 演算子

hasOwnProperty (Object.hasOwnProperty メソッド)

```
public hasOwnProperty(name:String) : Boolean
```

オブジェクトに指定されたプロパティが定義されているかどうかを示します。ターゲットオブジェクトに name パラメータで指定されたストリングに一致するプロパティがある場合は true を返します。それ以外の場合は false を返します。このメソッドは、オブジェクトのプロトタイプチェーンをチェックせず、オブジェクト自身にプロパティが存在する場合にのみ true を返します。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

name:String -

戻り値

Boolean - ブール値 : ターゲットオブジェクトに name パラメータで指定されたプロパティがある場合、この値は true になり、それ以外の場合は false になります。

isPropertyEnumerable (Object.isPropertyEnumerable メソッド)

```
public isPropertyEnumerable(name:String) : Boolean
```

指定されたプロパティが存在し列挙できるかどうかを示します。true の場合、このプロパティが存在し、for..in ループで列挙できます。このメソッドではターゲットオブジェクトのプロトタイプチェーンをチェックしないため、プロパティがターゲットオブジェクト上に存在している必要があります。

作成するプロパティは列挙できますが、ビルトインプロパティは通常列挙できません。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

name:String -

戻り値

Boolean - ブール値 : name パラメータで指定されたプロパティが列挙可能な場合、この値は true になります。

例

次の例では、汎用オブジェクトを作成し、プロパティをオブジェクトに追加してから、オブジェクトが列挙可能かどうかを確認します。この例では、対比のために、Array.length プロパティというビルトインプロパティが列挙できないことも示します。

```
var myObj:Object = new Object();
myObj.prop1 = "hello";
trace(myObj.isPropertyEnumerable("prop1")); // Output: true

var myArray = new Array();
trace(myArray.isPropertyEnumerable("length")); // Output: false
```

関連項目

for..in ステートメント

isPrototypeOf (Object.isPrototypeOf メソッド)

```
public isPrototypeOf(theClass:Object) : Boolean
```

Object クラスのインスタンスが、パラメータとして指定されたオブジェクトのプロトタイプチェーン内にあるかどうかを示します。このメソッドは、オブジェクトが theClass パラメータで指定されたオブジェクトのプロトタイプチェーン内にある場合に true を返します。ターゲットオブジェクトが theClass オブジェクトのプロトタイプチェーン内にない場合はもちろん、theClass パラメータがオブジェクトでない場合も、false を返します。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

theClass:Object -

戻り値

Boolean - ブール値 : オブジェクトが theClass パラメータで指定されたオブジェクトのプロトタイプチェーン内にある場合、この値は true になり、それ以外の場合は false になります。

Object コンストラクタ

```
public Object()
```

Object オブジェクトを作成し、そのオブジェクトのコンストラクタメソッドへの参照をオブジェクトの constructor プロパティに格納します。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、myObject という名前の汎用オブジェクトを作成します。

```
var myObject:Object = new Object();
```

__proto__ (Object.__proto__ プロパティ)

```
public __proto__ : Object
```

オブジェクトの作成に使用された、クラスの prototype プロパティ (ActionScript 2.0) またはコンストラクタ関数 (ActionScript 1.0) を参照します。__proto__ プロパティは、作成したすべてのオブジェクトに自動的に割り当てられます。ActionScript インタプリタは、__proto__ プロパティを使用してオブジェクトのクラスまたはコンストラクタ関数の prototype プロパティにアクセスし、オブジェクトがそのスーパークラスから継承しているプロパティとメソッドを確認します。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、Shape という名前のクラスと Circle という名前の Shape のスーパークラスを作成します。

```
// Shape class defined in external file named Shape.as
class Shape {
  function Shape() {}
}

// Circle class defined in external file named Circle.as
class Circle extends Shape{
  function Circle() {}
}
```

Circle クラスを使用して、Circle の 2 つのインスタンスを作成できます。

```
var oneCircle:Circle = new Circle();
var twoCircle:Circle = new Circle();
```

次の trace ステートメントでは、両方のインスタンスの __proto_ プロパティが Circle クラスの prototype プロパティを参照していることを示します。

```
trace(Circle.prototype == oneCircle.__proto__); // Output: true
trace(Circle.prototype == twoCircle.__proto__); // Output: true
```

関連項目

prototype (Object.prototype プロパティ)

prototype (Object.prototype プロパティ)

```
public static prototype : Object
```

クラスまたは関数オブジェクトのスーパークラスを参照します。prototype プロパティは自動的に作成され、作成したクラスまたは関数オブジェクトに割り当てられます。このプロパティは、作成したクラスまたは関数に固有であるという点で静的です。たとえば、カスタムクラスを作成すると、prototype プロパティの値は、クラスのすべてのインスタンスで共有され、クラスプロパティとしてのみアクセスできます。カスタムクラスのインスタンスは、prototype プロパティに直接アクセスできません。アクセスするには、__proto__ プロパティ経由で行います。

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の例では、Shape という名前のクラスと Circle という名前の Shape のスーパークラスを作成します。

```
// Shape class defined in external file named Shape.as
class Shape {
  function Shape() {}
}

// Circle class defined in external file named Circle.as
class Circle extends Shape{
  function Circle() {}
}
```

Circle クラスを使用して、Circle の 2 つのインスタンスを作成できます。

```
var oneCircle:Circle = new Circle();
var twoCircle:Circle = new Circle();
```

次の trace ステートメントでは、Circle クラスの prototype プロパティがそのスーパークラス Shape をポイントしていることを示します。識別子 Shape は、Shape クラスのコンストラクタ関数を参照しています。

```
trace(Circle.prototype.constructor == Shape); // Output: true
```

次の trace ステートメントでは、prototype プロパティと __proto__ プロパティを同時に使用して、継承階層 (またはプロトタイプチェーン) を 2 レベル上に移動する方法を示します。Circle.prototype.__proto__ プロパティには、Shape クラスのスーパークラスへの参照が格納されています。

```
trace(Circle.prototype.__proto__ == Shape.prototype); // Output: true
```

関連項目

__proto__ (Object.__proto__ プロパティ)

registerClass (Object.registerClass メソッド)

```
public static registerClass(name:String, theClass:Function) : Boolean
```

ムービークリップシンボルと ActionScript オブジェクトクラスを関連付けます。シンボルが存在しない場合は、ストリング識別子とオブジェクトクラスが関連付けられます。

指定されたムービークリップシンボルのインスタンスがタイムラインで挿入された場合、そのシンボルは MovieClip クラスではなく、theClass パラメータで指定されたクラスに登録されます。

指定したムービークリップシンボルのインスタンスが MovieClip.attachMovie() または MovieClip.duplicateMovieClip() によって作成された場合は、MovieClip クラスではなく、theClass で指定したクラスに登録されます。theClass が null である場合、このメソッドは指定されたムービークリップシンボルまたはクラス識別子に関連する ActionScript クラス定義を削除します。ムービークリップシンボルの場合、ムービークリップの既存のインスタンスは変更されませんが、シンボルの新しいインスタンスはデフォルトクラス MovieClip に関連付けられます。

シンボルが既にクラスに登録されている場合は、それを新しい登録に置き換えます。

ムービークリップインスタンスがタイムラインで挿入されたか、attachMovie() または duplicateMovieClip() で作成された場合、ActionScript はそのオブジェクトを指すキーワード this を使用して対応するクラスのコンストラクタを呼び出します。コンストラクタ関数の呼び出しにはパラメータを使用しません。

このメソッドを使って、MovieClip 以外の ActionScript クラスにムービークリップを登録すると、新しいクラスのプロトタイプチェーンに MovieClip クラスを含めない限り、ムービークリップシンボルは、ビルトイン MovieClip クラスのメソッド、プロパティ、およびイベントを継承しません。次のコードでは、MovieClip クラスのプロパティを継承する theClass という新しい ActionScript クラスを作成します。

```
theClass.prototype = new MovieClip();
```

対応バージョン： ActionScript 1.0、Flash Player 6 - ActionScript 2.0 でクラスを使用している場合には、このメソッドを使用するのではなく、[リンケージプロパティ] ダイアログボックスまたは [シンボルプロパティ] ダイアログボックスの [ActionScript 2.0 クラス] フィールドを使用して、オブジェクトとクラスを関連付けることができます。

パラメータ

name:String - ムービークリップシンボルのリンケージ識別子、または ActionScript クラスのストリング識別子。

theClass:Function - ActionScript クラスのコンストラクタ関数への参照、またはシンボルを登録解除する null。

戻り値

Boolean - ブール値 : クラスが正常に登録されると、true が返されます。それ以外は、false が返されます。

関連項目

attachMovie (MovieClip.attachMovie メソッド), duplicateMovieClip (MovieClip.duplicateMovieClip メソッド)

__resolve (Object.__resolve プロパティ)

```
public __resolve : Object
```

ActionScript コードが未定義のメソッドまたはプロパティを参照する場合に自動的に呼び出されるユーザー定義関数への参照です。ActionScript コードがオブジェクトの未定義のプロパティまたはメソッドを参照する場合、Flash Player はオブジェクトの __resolve プロパティが定義されているかどうかを判断します。__resolve が定義されていると、参照先の関数が実行され、未定義のプロパティまたはメソッドの名前が渡されます。これにより、プロパティまたはメソッドが実際に定義された場合と同じように、未定義のプロパティの値および未定義のメソッドのステートメントをプログラムによって提供できます。このプロパティは、クライアント / サーバー間で非常に透過的なやり取りが行われるようにする場合に有用であり、サーバーサイドメソッドを呼び出す方法として推奨されています。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例は最初の例から段階的に構築されており、__resolve プロパティの 5 とおりのシンタックスを示しています。わかりやすくするために、前のシンタックスと異なるキーステートメントは太字になっています。

シンタックス 1: 次の例では、__resolve を使用し、未定義のプロパティがそれぞれ "Hello, world!" という値を返すオブジェクトを構築しています。

```
// instantiate a new object
var myObject:Object = new Object();

// define the __resolve function
myObject.__resolve = function (name) {
  return "Hello, world!";
};
trace (myObject.property1); // output: Hello, world!
trace (myObject.property2); // output: Hello, world!
```

シンタックス 2: 次の例では、__resolve を ファンクター (関数を生成する関数) として使用しています。__resolve を使用することで、未定義のメソッドの呼び出しが、myFunction という名前の汎用関数にリダイレクトされます。

```
// instantiate a new object
var myObject:Object = new Object();

// define a function for __resolve to call
myObject.myFunction = function (name) {
  trace("Method " + name + " was called");
};
```

```
// define the __resolve function
myObject.__resolve = function (name) {
    return function () { this.myFunction(name); };
};

// test __resolve using undefined method names
myObject.someMethod(); // output: Method someMethod was called
myObject.someOtherMethod(); //output: Method someOtherMethod was called
```

シンタックス 3 : 次の例は、前の例を基にして構築されており、解決されたメソッドをキャッシュに保存する機能を追加しています。メソッドをキャッシュに保存すると、__resolve が各メソッドに対して呼び出されるのは 1 回だけです。したがって、オブジェクトメソッドの遅延構築が可能です。遅延構築は、メソッドが最初に使用されるときまで、メソッドの作成 (構築) を延期する最適化技術です。

```
// instantiate a new object
var myObject:Object = new Object();
// define a function for __resolve to call
myObject.myFunction = function(name) {
  trace("Method "+name+" was called");
};
// define the __resolve function
myObject.__resolve = function(name) {
  trace("Resolve called for "+name); // to check when __resolve is called
  // Not only call the function, but also save a reference to it
  var f:Function = function () {
    this.myFunction(name);
  };
  // create a new object method and assign it the reference
  this[name] = f;
  // return the reference
  return f;
};
// test __resolve using undefined method names
// __resolve will only be called once for each method name
myObject.someMethod(); // calls __resolve
myObject.someMethod(); // does not call __resolve because it is now defined
myObject.someOtherMethod(); // calls __resolve
myObject.someOtherMethod(); // does not call __resolve, no longer undefined
```

シンタックス 4 : 次の例は前の例を基にして構築されており、メソッド名 onStatus() をローカルで使用するために予約し、他の未定義のプロパティと同じ方法で解決されないようにしています。追加されたコードは太字になっています。

```
// instantiate a new object
var myObject:Object = new Object();
// define a function for __resolve to call
myObject.myFunction = function(name) {
  trace("Method "+name+" was called");
};
// define the __resolve function
myObject.__resolve = function(name) {
  // reserve the name "onStatus" for local use
  if (name == "onStatus") {
    return undefined;
  }
  trace("Resolve called for "+name); // to check when __resolve is called
  // Not only call the function, but also save a reference to it
  var f:Function = function () {
    this.myFunction(name);
  };
  // create a new object method and assign it the reference
  this[name] = f;
  // return the reference
  return f;
};
// test __resolve using the method name "onStatus"
trace(myObject.onStatus("hello"));
// output: undefined
```

シンタックス 5 : 次の例は前の例を基にして構築されており、パラメータを受け入れるファンクターを作成しています。この例では、arguments オブジェクトのシンタックスを拡張し、Array クラスのメソッドをいくつか使用しています。

```
// instantiate a new object
var myObject:Object = new Object();

// define a generic function for __resolve to call
myObject.myFunction = function (name) {
  arguments.shift();
  trace("Method " + name + " was called with arguments: " +
  arguments.join(','));
};

// define the __resolve function
myObject.__resolve = function (name) {
  // reserve the name "onStatus" for local use
  if (name == "onStatus") {
    return undefined;
  }
  var f:Function = function () {
    arguments.unshift(name);
    this.myFunction.apply(this, arguments);
  };
```

```
  // create a new object method and assign it the reference
  this[name] = f;
  // return the reference to the function
  return f;
};

// test __resolve using undefined method names with parameters
myObject.someMethod("hello");
// output: Method someMethod was called with arguments: hello

myObject.someOtherMethod("hello","world");
// output: Method someOtherMethod was called with arguments: hello,world
```

関連項目

引数, Array

toString (Object.toString メソッド)

```
public toString() : String
```

指定されたオブジェクトをストリングに変換し、返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

String - ストリング。

例

この例では、汎用オブジェクトの toString() に対する戻り値を示しています。

```
var myObject:Object = new Object();
trace(myObject.toString()); // output: [object Object]
```

このメソッドを上書きし、さらに意味のある値を返すことができます。次の例では、このメソッドがビルトインクラスの Date、Array、および Number に対して上書きされたことを示しています。

```
// Date.toString() returns the current date and time
var myDate:Date = new Date();
trace(myDate.toString()); // output: [current date and time]

// Array.toString() returns the array contents as a comma-delimited string
var myArray:Array = new Array("one", "two");
trace(myArray.toString()); // output: one,two
```

```
// Number.toString() returns the number value as a string
// Because trace() won't tell us whether the value is a string or number
// we will also use typeof() to test whether toString() works.
var myNumber:Number = 5;
trace(typeof (myNumber)); // output: number
trace(myNumber.toString()); // output: 5
trace(typeof (myNumber.toString())); // output: string
```

次の例では、カスタムクラスの toString() を上書きする方法を示しています。最初に、"*Vehicle.as*" という名前のテキストファイルを作成し、そのファイルに Vehicle クラス定義のみを含めて、設定フォルダ内の "Classes" フォルダに格納します。

```
// contents of Vehicle.as
class Vehicle {
   var numDoors:Number;
   var color:String;
   function Vehicle(param_numDoors:Number, param_color:String) {
   this.numDoors = param_numDoors;
   this.color = param_color;
   }
   function toString():String {
   var doors:String = "door";
   if (this.numDoors > 1) {
      doors += "s";
   }
   return ("A vehicle that is " + this.color + " and has " + this.numDoors + " "
   + doors);
   }
}

// code to place into a FLA file
var myVehicle:Vehicle = new Vehicle(2, "red");
trace(myVehicle.toString());
// output: A vehicle that is red and has 2 doors

// for comparison purposes, this is a call to valueOf()
// there is no primitive value of myVehicle, so the object is returned
// giving the same output as toString().
trace(myVehicle.valueOf());
// output: A vehicle that is red and has 2 doors
```

unwatch (Object.unwatch メソッド)

```
public unwatch(name:String) : Boolean
```

Object.watch() で作成した監視ポイントを削除します。このメソッドは、監視ポイントが正常に削除されると、true を返します。それ以外は、false を返します。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

name:String - 監視対象から外すオブジェクトプロパティの名前を示すストリング。

戻り値

Boolean - ブール値 : 監視ポイントが正常に削除された場合は true を返します。それ以外の場合は false を返します。

例

Object.watch() の例を参照してください。

関連項目

watch (Object.watch メソッド), addProperty (Object.addProperty メソッド)

valueOf (Object.valueOf メソッド)

```
public valueOf() : Object
```

指定されたオブジェクトのプリミティブな値を返します。オブジェクトにプリミティブな値がない場合、オブジェクトが返されます。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Object - 指定したオブジェクトのプリミティブ値、またはオブジェクトそのもの。

例

次の例では、汎用オブジェクト (プリミティブな値を持たない) の valueOf() の戻り値を示し、toString() の戻り値と比較しています。最初に、汎用オブジェクトを作成します。// 次に、新しい Date オブジェクトを作成し、February 1, 2004, 8:15 AM に設定する // toString() メソッドは人間が理解できる形式で現在の時間を返す valueOf() メソッドは、プリミティブ値をミリ秒単位で返します。次に、2 つの単純なエレメントを含む新しい Array オブジェクトを作成します。toString() と valueOf() は同じ値 one,two を返します。

```
// Create a generic object
var myObject:Object = new Object();
trace(myObject.valueOf()); // output: [object Object]
trace(myObject.toString()); // output: [object Object]
```

次の例では、ビルトインクラス Date および Array の戻り値を示し、Object.toString() の戻り値と比較しています。

```
// Create a new Date object set to February 1, 2004, 8:15 AM
// The toString() method returns the current time in human-readable form
// The valueOf() method returns the primitive value in milliseconds
var myDate:Date = new Date(2004,01,01,8,15);
trace(myDate.toString()); // output: Sun Feb 1 08:15:00 GMT-0800 2004
trace(myDate.valueOf()); // output: 1075652100000

// Create a new Array object containing two simple elements
// In this case both toString() and valueOf() return the same value: one,two
var myArray:Array = new Array("one", "two");
trace(myArray.toString()); // output: one,two
trace(myArray.valueOf()); // output: one,two
```

toString() を上書きするカスタム クラスの Object.valueOf() の戻り値の例については、Object.toString() を参照してください。

関連項目

toString (Object.toString メソッド)

watch (Object.watch メソッド)

```
public watch(name:String, callback:Function, [userData:Object]) : Boolean
```

ActionScript オブジェクトの指定されたプロパティの変更に応じて呼び出されるイベントハンドラを登録します。プロパティが変更されると、イベントハンドラはそれを含むオブジェクト myObject と共に呼び出されます。

callback メソッド定義で return ステートメントを使用して、監視しているプロパティの値に影響を与えることができます。callback メソッドによって返される値は、監視対象のオブジェクトプロパティに割り当てられます。プロパティへの変更を監視するか、変更するか、または禁止するかによって、戻り値の選択は異なります。

- 単にプロパティを監視する場合は、newVal パラメータを返します。
- プロパティの値を変更する場合は、固有の値を返します。
- プロパティを変更しないようにするには、oldVal パラメータを返します。

定義する callback メソッドに return ステートメントがない場合は、監視対象のオブジェクトプロパティに undefined. の値が割り当てられます。

監視ポイントは、変更された newval (または oldval) を返すことにより、値の代入をフィルタリング (または無効化) できます。監視ポイントが設定されているプロパティを削除しても、その監視ポイントは消えません。後でプロパティを再作成するときに、監視ポイントは依然として有効です。監視ポイントを削除するには、Object.unwatch メソッドを使用します。

1つのプロパティに登録できる監視ポイントは1つのみです。同じプロパティに対して Object.watch() を続けて呼び出すと、元の監視ポイントが置き換えられます。

Object.watch() メソッドの動作は、JavaScript 1.2 以降の Object.watch() 関数に似ています。主な相違点は、userData パラメータです。これは Flash で Object.watch() に追加されたパラメータであり、Netscape Navigator ではサポートされていません。userData パラメータは、イベントハンドラに渡し、イベントハンドラ内で使用できます。

Object.watch() メソッドでは getter/setter プロパティを監視できません。getter/setter プロパティは遅延評価に基づいて処理されます。つまり、プロパティの値は、プロパティが実際に検索されるまで決定されません。遅延評価では、必要になるまで評価が遅延されるため、プロパティの更新頻度が低い場合には便利です。ただし、Object.watch() はプロパティを評価し、callback 関数を呼び出すかどうかを決定する必要があります。getter/setter プロパティを使用すると、Object.watch() はプロパティを常に評価する必要があり、非効率的です。

通常、ActionScript の定義済みプロパティである _x、_y、_width、_height などは getter/setter プロパティであり、Object.watch() では監視できません。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

name: String - 監視対象であるオブジェクトプロパティの名前を示すストリング。

callback: Function - 監視対象のプロパティの変更に応じて呼び出す関数。このパラメータは関数オブジェクトです。ストリングとしての関数名ではありません。callback の形式は callback(prop, oldVal, newVal, userData) です。

userData: Object (オプション) - callback メソッドに渡す任意の ActionScript データ。userData パラメータを省略すると、undefined が callback メソッドに渡されます。

戻り値

Boolean - ブール値 : 監視ポイントが正常に作成された場合は true を返します。それ以外の場合は false を返します。

例

次の例では、watch() を使用し、speed プロパティが速度制限を超えるかどうかを監視しています。

```
// Create a new object
var myObject:Object = new Object();

// Add a property that tracks speed
myObject.speed = 0;

// Write the callback function to be executed if the speed property changes
var speedWatcher:Function = function(prop, oldVal, newVal, speedLimit) {
  // Check whether speed is above the limit
  if (newVal > speedLimit) {
  trace ("You are speeding.");
  }
  else {
  trace ("You are not speeding.");
  }

  // Return the value of newVal.
  return newVal;
}
// Use watch() to register the event handler, passing as parameters:
// - the name of the property to watch: "speed"
// - a reference to the callback function speedWatcher
// - the speedLimit of 55 as the userData parameter
myObject.watch("speed", speedWatcher, 55);

// set the speed property to 54, then to 57
myObject.speed = 54; // output: You are not speeding
myObject.speed = 57; // output: You are speeding

// unwatch the object
myObject.unwatch("speed");
myObject.speed = 54; // there should be no output
```

関連項目

addProperty (Object.addProperty メソッド),unwatch (Object.unwatch メソッド)

Point (flash.geom.Point)

```
Object
  |
  +-flash.geom.Point
```

```
public class Point
extends Object
```

Point クラスは、*x* が水平軸、*y* が垂直軸を表す、2 次元の座標系内の位置を表します。

次のコードでは、位置 (0,0) に配置されるポイントを作成します。

```
var myPoint:Point = new Point();
```

対応バージョン： ActionScript 1.0、Flash Player 8

プロパティ一覧

オプション	プロパティ	説明
	length:Number	(0,0) からこのポイントまでの線のセグメントの長さです。
	x:Number	ポイントの水平座標です。
	y:Number	ポイントの垂直座標です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター一覧

署名	説明
Point(x:Number, y:Number)	新しいポイントを作成します。

メソッド一覧

オプション	署名	説明
	add(v:Point) : Point	このポイントの座標に他のポイントの座標を加算して、新しいポイントを作成します。
	clone() : Point	この Point オブジェクトのコピーを作成します。
static	distance(pt1:Point, pt2:Point) : Number	pt1 と pt2 との距離を返します。
	equals(toCompare: Object) : Boolean	2 つのポイントが等しいかどうかを判別します。
static	interpolate(pt1: Point, pt2:Point, f:Number) : Point	2 つの指定されたポイント間にあるポイントを判別します。
	normalize(length: Number) : Void	(0,0) と現在のポイント間の線のセグメントを設定された長さに拡大・縮小します。
	offset(dx:Number, dy:Number) : Void	Point オブジェクトを指定された量だけオフセットします。
static	polar(len:Number, angle:Number) : Point	極座標ペアを直交点座標に変換します。
	subtract(v:Point) : Point	このポイントの座標から他のポイントの座標を減算して、新しいポイントを作成します。
	toString() : String	*x* 座標の値と *y* 座標の値を格納するストリングを返します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

add (Point.add メソッド)

```
public add(v:Point) : Point
```

このポイントの座標に他のポイントの座標を加算して、新しいポイントを作成します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

v:Point - 追加するポイント。

戻り値

Point - 新しいポイント。

例

次の例では、point_2 を point_1 に加算して Point オブジェクト resultPoint を作成します。

```
import flash.geom.Point;
var point_1:Point = new Point(4, 8);
var point_2:Point = new Point(1, 2);
var resultPoint:Point = point_1.add(point_2);
trace(resultPoint.toString()); // (x=5, y=10)
```

clone (Point.clone メソッド)

```
public clone() : Point
```

この Point オブジェクトのコピーを作成します。

対応バージョン : ActionScript 1.0、Flash Player 8

戻り値

Point - 新しい Point オブジェクト。

例

次の例では、myPoint オブジェクトで検出された値を使用して clonedPoint という Point オブジェクトのコピーを作成します。clonedPoint オブジェクトには、myPoint の値がすべて含まれていますが、同じオブジェクトではありません。

```
import flash.geom.Point;
var myPoint:Point = new Point(1, 2);
var clonedPoint:Point = myPoint.clone();
trace(clonedPoint.x); // 1
trace(clonedPoint.y); // 2
trace(myPoint.equals(clonedPoint)); // true
trace(myPoint === clonedPoint); // false
```

distance (Point.distance メソッド)

```
public static distance(pt1:Point, pt2:Point) : Number
```

pt1 と pt2 との距離を返します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`pt1:Point` - 最初のポイント。

`pt2:Point` - 2 番目のポイント。

戻り値

`Number` - 最初のポイントと 2 番目のポイント間の距離。

例

次の例では、`point_1` と `point_2` を作成し、その間の距離 (`distanceBetween`) を判別します。

```
import flash.geom.Point;
var point_1:Point = new Point(-5, 0);
var point_2:Point = new Point(5, 0);
var distanceBetween:Number = Point.distance(point_1, point_2);
trace(distanceBetween); // 10
```

equals (Point.equals メソッド)

```
public equals(toCompare:Object) : Boolean
```

2 つのポイントが等しいかどうかを判別します。*x* 値と *y* 値が同じ場合、2 つのポイントは等しいことになります。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`toCompare:Object` - 比較するポイント。

戻り値

`Boolean` - オブジェクトがこの Point オブジェクトと等しい場合は `true` を返します。等しくない場合は `false` を返します。

例

次の例では、一方のポイントの値が他方のポイントの値と等しいかどうかを判別します。2 つのオブジェクトが等しい場合、equals() は厳密な等価演算子 (===) と同じ結果は返しません。

```
import flash.geom.Point;
var point_1:Point = new Point(1, 2);
var point_2:Point = new Point(1, 2);
var point_3:Point = new Point(4, 8);
trace(point_1.equals(point_2)); // true
trace(point_1.equals(point_3)); // false
trace(point_1 === point_2); // false
trace(point_1 === point_3); // false
```

interpolate (Point.interpolate メソッド)

```
public static interpolate(pt1:Point, pt2:Point, f:Number) : Point
```

2 つの指定されたポイント間にあるポイントを判別します。パラメータ f はパラメータ pt1 および pt2 で指定された 2 つの端点に対する、新しい補間ポイントの場所を決定します。パラメータ f が 1.0 に近づくほど、補間ポイントは最初のポイント (パラメータ pt1) に近づきます。パラメータ f が 0 に近づくほど、補間ポイントは 2 番目のポイント (パラメータ pt2) に近づきます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

pt1:Point - 最初のポイント。

pt2:Point - 2 番目のポイント。

f:Number - 2 つのポイント間の補間のレベル。pt1 と pt2 間の線に沿って新しいポイントがある位置を示します。f=1 の場合は pt1 が返されます。f=0 の場合は pt2 が返されます。

戻り値

Point - 新しい補間ポイント。

例

次の例では、point_1 と point_2 の真ん中 (50%) にある補間ポイント (interpolatedPoint) を見つけます。

```
import flash.geom.Point;
var point_1:Point = new Point(-100, -100);
var point_2:Point = new Point(50, 50);
var interpolatedPoint:Point = Point.interpolate(point_1, point_2, 0.5);
trace(interpolatedPoint.toString()); // (x=-25, y=-25)
```

length (Point.length プロパティ)

```
public length : Number
```

(0,0) からこのポイントまでの線のセグメントの長さです。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、Point オブジェクト myPoint を作成し、(0, 0) からそのポイントまでの長さを判別します。

```
import flash.geom.Point;
var myPoint:Point = new Point(3,4);
trace(myPoint.length); // 5
```

関連項目

polar (Point.polar メソッド)

normalize (Point.normalize メソッド)

```
public normalize(length:Number) : Void
```

(0,0) と現在のポイント間の線のセグメントを設定された長さに拡大・縮小します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

length:Number - 拡大・縮小値。たとえば、現在のポイントが (0,5) で、1 に正規化すると、返されるポイントは (0,1) になります。

例

次の例では、normalizedPoint オブジェクトの長さを 5 から 10 に拡張します。

```
import flash.geom.Point;
var normalizedPoint:Point = new Point(3, 4);
trace(normalizedPoint.length); // 5
trace(normalizedPoint.toString()); // (x=3, y=4)
normalizedPoint.normalize(10);
trace(normalizedPoint.length); // 10
trace(normalizedPoint.toString()); // (x=6, y=8)
```

関連項目

length (Point.length プロパティ)

offset (Point.offset メソッド)

```
public offset(dx:Number, dy:Number) : Void
```

Point オブジェクトを指定された量だけオフセットします。dx の値を *x* の元の値に加算して、新しい *x* 値を作成します。dy の値を *y* の元の値に加算して、新しい *y* 値を作成します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

dx:Number - 水平座標 *x* をオフセットする量。

dy:Number - 垂直座標 *y* をオフセットする量。

例

次の例では、ポイントの位置を指定された *x* と *y* の量だけオフセットします。

```
import flash.geom.Point;
var myPoint:Point = new Point(1, 2);
trace(myPoint.toString()); // (x=1, y=2)
myPoint.offset(4, 8);
trace(myPoint.toString()); // (x=5, y=10)
```

関連項目

add (Point.add メソッド)

Point コンストラクタ

```
public Point(x:Number, y:Number)
```

新しいポイントを作成します。このメソッドのパラメータを渡さなければ、(0,0) にポイントが作成されます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

x:Number - 水平座標。デフォルト値は 0 です。

y:Number - 垂直座標。デフォルト値は 0 です。

例

最初の例では、デフォルトのコンストラクタを使用して Point オブジェクト point_1 を作成します。

```
import flash.geom.Point;
var point_1:Point = new Point();
trace(point_1.x); // 0
trace(point_1.y); // 0
```

2 番目の例では、座標 *x* = 1 および *y* = 2 を使用して Point オブジェクト point_2 を作成します。

```
import flash.geom.Point;
var point_2:Point = new Point(1, 2);
trace(point_2.x); // 1
trace(point_2.y); // 2
```

polar (Point.polar メソッド)

```
public static polar(len:Number, angle:Number) : Point
```

極座標ペアを直交点座標に変換します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`len:Number` - 極座標ペアの長さ座標。

`angle:Number` - 極座標ペアの角度 (ラジアン単位)。

戻り値

`Point` - 直交ポイント。

例

次の例では、angleInRadians と線の長さ 5 から cartesianPoint を作成します。辺の長さの比が 3:4:5 の直角三角形の特性から、Math.atan(3/4) と等しい angleInRadians が使用されています。

```
import flash.geom.Point;
var len:Number = 5;
var angleInRadians:Number = Math.atan(3/4);
var cartesianPoint:Point = Point.polar(len, angleInRadians);
trace(cartesianPoint.toString()); // (x=4, y=3)
```

コンピュータで pi などの超越数を処理する場合、浮動小数の演算では正確さに限度があるので、数値の丸めに関するエラーが発生することがあります。Math.PI を使用する場合、次の例に示すように、Math.round() 関数の使用を検討してください。

```
import flash.geom.Point;
var len:Number = 10;
var angleInRadians:Number = Math.PI;
var cartesianPoint:Point = Point.polar(len, angleInRadians);
trace(cartesianPoint.toString()); // should be (x=-10, y=0), but is (x=-10,
  y=1.22460635382238e-15)
trace(Math.round(cartesianPoint.y)); // 0
```

関連項目

length (Point.length プロパティ),round (Math.round メソッド)

subtract (Point.subtract メソッド)

```
public subtract(v:Point) : Point
```

このポイントの座標から他のポイントの座標を減算して、新しいポイントを作成します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

v:Point - 減算するポイント。

戻り値

Point - 新しいポイント。

例

次の例では、point_1 から point_2 を減算して point_3 を作成します。

```
import flash.geom.Point;
var point_1:Point = new Point(4, 8);
var point_2:Point = new Point(1, 2);
var resultPoint:Point = point_1.subtract(point_2);
trace(resultPoint.toString()); // (x=3, y=6)
```

toString (Point.toString メソッド)

```
public toString() : String
```

x 座標の値と *y* 座標の値を格納するストリングを返します。形式は (*x*, *y*) であるため、23,17 にあるポイントでは、"(*x* =23, *y* =17)" が返されます。

対応バージョン : ActionScript 1.0、Flash Player 8

戻り値

String - ストリング。

例

次の例では、ポイントを作成し、その値を形式 (x=x, y=y) のストリングに変換します。

```
import flash.geom.Point;
var myPoint:Point = new Point(1, 2);
trace("myPoint: " + myPoint.toString()); // (x=1, y=2)
```

x (Point.x プロパティ)

```
public x : Number
```

ポイントの水平座標。デフォルト値は 0 です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、Point オブジェクト myPoint を作成し、*x* 座標値を設定します。

```
import flash.geom.Point;
var myPoint:Point = new Point();
trace(myPoint.x); // 0
myPoint.x = 5;
trace(myPoint.x); // 5
```

y (Point.y プロパティ)

```
public y : Number
```

ポイントの垂直座標。デフォルト値は 0 です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、Point オブジェクト myPoint を作成し、*y* 座標値を設定します。

```
import flash.geom.Point;
var myPoint:Point = new Point();
trace(myPoint.y); // 0
myPoint.y = 5;
trace(myPoint.y); // 5
```

PrintJob

```
Object
  |
  +-PrintJob
```

```
public class PrintJob
extends Object
```

PrintJob クラスを使用すると、コンテンツを作成して 1 ページまたは複数のページに印刷できます。このクラスは、print() メソッドで利用できる印刷機能よりも優れているだけでなく、実行時にオフスクリーンで動的なコンテンツを描画し、1 つの [印刷] ダイアログボックスでユーザーからの入力を受け付け、コンテンツの比率のまま拡大・縮小なしでドキュメントを印刷することができます。この機能は、データベースコンテンツやダイナミックテキストなどダイナミックコンテンツをレンダリングおよびプリントするときに特に便利です。

さらに、PrintJob.start() によって得られるプロパティを使用すると、ドキュメントからユーザーのプリンタ設定 (用紙の高さ、幅、方向など) にアクセスすることができ、プリンタの設定に合わせて Flash のコンテンツを動的にフォーマットするようにドキュメントを構成することができます。これらのレイアウトプロパティは読み取り専用で、Flash Player では変更できません。

対応バージョン : ActionScript 1.0、Flash Player 7

プロパティ一覧

オプション	プロパティ	説明
	orientation:String (読み取り専用)	印刷するイメージの向きです。
	pageHeight:Number (読み取り専用)	ページ上で実際に印刷可能な領域の高さ (ポイント単位) です。
	pageWidth:Number (読み取り専用)	ページ上で実際に印刷可能な領域の幅 (ポイント単位) です。
	paperHeight:Number (読み取り専用)	用紙全体の高さ (ポイント単位) です。
	paperWidth:Number (読み取り専用)	用紙全体の幅 (ポイント単位) です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター覧

署名	説明
PrintJob()	ページを印刷できる PrintJob オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	addPage(target:Object, [printArea:Object], [options:Object], [frameNum:Number]) : Boolean	指定のレベルまたはムービークリップを 1 ページとして印刷スプーラに送ります。
	send() : Void	PrintJob.start() メソッドおよび PrintJob.addPage() メソッドの後に、スプールしたページをプリンタに送るために使用します。
	start() : Boolean	オペレーティングシステムの [印刷] ダイアログボックスを表示し、スプールを開始します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

addPage (PrintJob.addPage メソッド)

```
public addPage(target:Object, [printArea:Object], [options:Object],
  [frameNum:Number]) : Boolean
```

指定のレベルまたはムービークリップを 1 ページとして印刷スプーラに送ります。このメソッドを使う前に、PrintJob.start( ) を使用する必要があります。印刷ジョブに対して PrintJob.addPage( ) を 1 回または複数回使用した後、PrintJob.send( ) を使用して、スプールしたページをプリンタに送る必要があります。

このメソッドから false が返された場合 (PrintJob.start( ) を呼び出していない場合や、ユーザーが印刷ジョブを取り消した場合など) は、その後で PrintJob.addPage( ) を呼び出すと失敗します。ただし、それまでに PrintJob.addPage( ) の呼び出しが成功していれば、最後に PrintJob.send( ) コマンドを実行することで、スプールに成功したすべてのページがプリンタに送られます。

printArea の値を指定した場合、xMin 座標と yMin 座標がページの印刷可能領域の左上隅 (0,0 座標) に対応します。印刷可能領域は、PrintJob.start( ) で設定した読み取り専用の pageHeight プロパティと pageWidth プロパティで表されます。印刷結果はページ上の印刷可能領域の左上隅を基準に整列されるので、printArea で定義した領域が用紙上の印刷可能領域よりも大きい場合は、印刷結果の右側と下側の部分は切り取られることがあります。printArea の値を指定しない場合にステージが印刷可能領域より大きいときも、同様の切り取りが発生します。

印刷前にムービークリップを拡大 / 縮小するには、このメソッドを呼び出す前に MovieClip._xscale プロパティと MovieClip._yscale プロパティを設定します。その後、これらのプロパティを元の値に戻します。ムービークリップの拡大・縮小は、printArea とは無関係です。つまり、サイズが 50 x 50 ピクセルの領域を印刷するよう指定した場合、2500 ピクセルが印刷されます。ムービークリップを拡大・縮小した場合も同様に 2500 ピクセルで印刷されますが、拡大・縮小サイズは維持されます。

Flash Player の印刷機能は、PostScript プリンタと非 PostScript プリンタをサポートしています。非 PostScript プリンタでは、ベクターはビットマップに変換されます。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

target:Object - 数値またはストリング。印刷するムービークリップのレベルまたはインスタンス名です。レベルを示す数値 (_root ムービーの場合は 0)、またはムービークリップのインスタンス名を表す引用符 [""] で囲んだストリングを指定します。

printArea:Object (オプション) - 印刷する領域を示すオブジェクト。次の形式で指定します。

```
{xMin:topLeft, xMax:topRight, yMin:bottomLeft, yMax:bottomRight}
```

printArea に指定する座標は、_root ムービークリップ (target = 0 の場合)、または target に指定したレベルまたはムービークリップの基準点からの相対的な画面ピクセル数を表します。4 つすべての座標を指定する必要があります。幅 (xMax-xMin) および高さ (yMax-yMin) は 0 より大きい値にする必要があります。

ポイントとは印刷用の測定単位で、ピクセルとは画面用の測定単位です。ポイントは物理的な固定サイズ (1/72 インチ) ですが、ピクセル深度のサイズは画面の解像度によって異なります。次に、インチやセンチメートルと、twip (ピクセルの 1/20) やポイントとの対応を示します。

- 1 point = 1/72 inch = 20 twips
- 1 inch = 72 points = 1440 twips
- 1 cm = 567 twips

ピクセルとポイントの間の変換レートは、プリンタの設定、およびムービークリップが伸縮されているかどうかによって決まります。72 ピクセル幅の伸縮されていないムービークリップは、用紙上には 1 インチ幅でプリントされます。画面の解像度とは関係なく、1 ポイントが 1 ピクセルに対応します。

メモ : `print()`、`printAsBitmap()`、`printAsBitmapNum()`、または `printNum()` を使って印刷する場合には、#p フレームラベルを使って印刷領域を指定するという方法がありました。`addPage()` メソッドを使用する場合は、`printArea` パラメータを使用して印刷領域を指定します。#p フレームラベルは無視されます。

`printArea` パラメータを省略した場合、または間違った値を指定した場合は、`target` の全ステージ領域が印刷されます。`printArea` の値を指定せずに、`options` または `frameNumber` の値を指定するには、`printArea` に `null` を指定します。

`options:Object` (オプション) - ベクターとビットマップのどちらを印刷するかを指定するパラメータ。次の形式を使用します。

`{printAsBitmap:`*`Boolean`*`}`

デフォルト値は `false` で、これはベクター形式を示します。`target` をビットマップとして印刷するには、printAsBitmap に `true` を指定します。使用する値を決めるときには、次の項目に注意してください。

- 印刷するコンテンツにビットマップイメージが含まれる場合は、透明効果とカラー効果を含めるために `{printAsBitmap:true}` を使用します。
- コンテンツにビットマップイメージが含まれない場合は、このパラメータを省略するか `{printAsBitmap:false}` を使用して、より品質の高いベクター形式で印刷します。

`options` を省略した場合、または間違った値を渡した場合は、ベクター形式が使用されます。`options` の値を指定せずに、`frameNumber` の値を指定するには、`options` に `null` を指定します。

`frameNum:Number` (オプション) - 印刷するフレームを指定できるオプションの数値。`frameNumber` を渡しても、そのフレームの ActionScript は呼び出されません。このパラメータを省略すると、`target` 内の現在のフレームが印刷されます。

メモ : `print()`、`printAsBitmap()`、`printAsBitmapNum()`、または `printNum()` を使用して印刷する場合、複数のフレームを印刷する場合には、印刷するページを指定するために #p フレームラベルを使用するという方法がありました。`PrintJob.addPage()` を使用して複数のフレームを印刷するには、フレームごとに `PrintJob.addPage()` コマンドを使う必要があります。#p フレームラベルは無視されます。プログラムで行う方法については、「例」のセクションを参照してください。

戻り値

Boolean - ブール値 : ページが正常に印刷スプーラに送られた場合は true、それ以外の場合は false を返します。

例

次に、addPage() コマンドのいくつかの使用例を示します。

```
my_btn.onRelease = function()
{
  var pageCount:Number = 0;

  var my_pj:PrintJob = new PrintJob();

  if (my_pj.start())
  {
  // Print entire current frame of the _root movie in vector format
  if (my_pj.addPage(0)){
    pageCount++;

    // Starting at 0,0, print an area 400 pixels wide and 500 pixels high
    // of the current frame of the _root movie in vector format
    if (my_pj.addPage(0, {xMin:0,xMax:400,yMin:0,yMax:500})){
    pageCount++;

    // Starting at 0,0, print an area 400 pixels wide and 500 pixels high
    // of frame 1 of the _root movie in bitmap format
    if (my_pj.addPage(0, {xMin:0,xMax:400,yMin:0,yMax:500},
      {printAsBitmap:true}, 1)){
      pageCount++;

      // Starting 50 pixels to the right of 0,0 and 70 pixels down,
      // print an area 500 pixels wide and 600 pixels high
      // of frame 4 of level 5 in vector format
      if (my_pj.addPage(5, {xMin:50,xMax:550,yMin:70,yMax:670},null, 4)){
      pageCount++;

      // Starting at 0,0, print an area 400 pixels wide
      // and 400 pixels high of frame 3 of the "dance_mc" movie clip
      // in bitmap format
      if (my_pj.addPage("dance_mc",
        {xMin:0,xMax:400,yMin:0,yMax:400},{printAsBitmap:true}, 3)){
        pageCount++;

        // Starting at 0,0, print an area 400 pixels wide
        // and 600 pixels high of frame 3 of the "dance_mc" movie clip
        // in vector format at 50% of its actual size
        var x:Number = dance_mc._xscale;
        var y:Number = dance_mc._yscale;
        dance_mc._xscale = 50;
```

```
            dance_mc._yscale = 50;

            if (my_pj.addPage("dance_mc",
               {xMin:0,xMax:400,yMin:0,yMax:600},null, 3)){
            pageCount++;
            }
            dance_mc._xscale = x;
            dance_mc._yscale = y;
          }
          }
       }
       }
    }
    }

    // If addPage() was successful at least once, print the spooled pages.
    if (pageCount > 0){
    my_pj.send();
    }
    delete my_pj;
}
```

関連項目

send (PrintJob.send メソッド),start (PrintJob.start メソッド)

orientation (PrintJob.orientation プロパティ)

public orientation : String (読み取り専用)

印刷するイメージの向きです。このプロパティは、"landscape" または "portrait" のいずれかに設定できます。このプロパティは、PrintJob.start() メソッドの呼び出し後にのみ使用可能になります。

対応バージョン : ActionScript 1.0、Flash Player 7

pageHeight (PrintJob.pageHeight プロパティ)

public pageHeight : Number (読み取り専用)

ページ上で実際に印刷可能な領域の高さ (ポイント単位) です。ユーザーが設定した余白は含みません。このプロパティは、PrintJob.start() メソッドの呼び出し後にのみ使用可能になります。

対応バージョン : ActionScript 1.0、Flash Player 7

pageWidth (PrintJob.pageWidth プロパティ)

public pageWidth : Number (読み取り専用)

ページ上で実際に印刷可能な領域の幅 (ポイント単位) です。ユーザーが設定した余白は含みません。このプロパティは、PrintJob.start() メソッドの呼び出し後にのみ使用可能になります。

対応バージョン : ActionScript 1.0、Flash Player 7

paperHeight (PrintJob.paperHeight プロパティ)

public paperHeight : Number (読み取り専用)

用紙全体の高さ (ポイント単位) です。このプロパティは、PrintJob.start() メソッドの呼び出し後にのみ使用可能になります。

対応バージョン : ActionScript 1.0、Flash Player 7

paperWidth (PrintJob.paperWidth プロパティ)

public paperWidth : Number (読み取り専用)

用紙全体の幅 (ポイント単位) です。このプロパティは、PrintJob.start() メソッドの呼び出し後にのみ使用可能になります。

対応バージョン : ActionScript 1.0、Flash Player 7

PrintJob コンストラクタ

public PrintJob()

ページを印刷できる PrintJob オブジェクトを作成します。

印刷ジョブを実装するには、次のメソッドを順に使用します。コンストラクタから PrintJob.send()、および削除まで、特定の印刷ジョブに関連するすべてのコマンドを同じフレーム内に配置する必要があります。my_pj.addPage() メソッド呼び出しの [params] をカスタムパラメータで置き換えます。

```
// create PrintJob object
var my_pj:PrintJob = new PrintJob();

// display print dialog box, but only initiate the print job
// if start returns successfully.
if (my_pj.start()) {

   // use a variable to track successful calls to addPage
   var pagesToPrint:Number = 0;

   // add specified area to print job
   // repeat once for each page to be printed
   if (my_pj.addPage([params])) {
```

```
  pagesToPrint++;
  }
  if (my_pj.addPage([params])) {
  pagesToPrint++;
  }
  if (my_pj.addPage([params])) {
  pagesToPrint++;
  }

  // send pages from the spooler to the printer, but only if one or more
  // calls to addPage() was successful. You should always check for successful
  // calls to start() and addPage() before calling send().
  if (pagesToPrint > 0) {
  my_pj.send();  // print page(s)
  }
}

// clean up
delete my_pj;  // delete object
```

最初の PrintJob オブジェクトがアクティブである間は、2 つ目の PrintJob オブジェクトを作成できません。1 つ目の PrintJob オブジェクトがアクティブであるときに new PrintJob() を呼び出して 2 つ目の PrintJob オブジェクトを作成しようとしても、2 つ目の PrintJob オブジェクトは生成されません。

対応バージョン： ActionScript 1.0、Flash Player 7

例

PrintJob.addPage() を参照してください。

関連項目

addPage (PrintJob.addPage メソッド),send (PrintJob.send メソッド),start (PrintJob.start メソッド)

send (PrintJob.send メソッド)

```
public send() : Void
```

PrintJob.start() メソッドおよび PrintJob.addPage() メソッドの後に、スプールしたページをプリンタに送るために使用します。PrintJob.send() への呼び出しは、関連する PrintJob.start() および PrintJob.addpage() への呼び出しが失敗した場合は失敗するので、PrintJob.send() を呼び出す前に、PrintJob.addpage() および PrintJob.start() への呼び出しが成功したかどうかをチェックする必要があります。

```
var my_pj:PrintJob = new PrintJob();
if (my_pj.start()) {
   if (my_pj.addPage(this)) {
   my_pj.send();
   }
}
delete my_pj;
```

対応バージョン： ActionScript 1.0、Flash Player 7

例

詳細については、PrintJob.addPage() および PrintJob.start() を参照してください。

関連項目

addPage (PrintJob.addPage メソッド), start (PrintJob.start メソッド)

start (PrintJob.start メソッド)

```
public start() : Boolean
```

オペレーティングシステムの [印刷] ダイアログボックスを表示し、スプールを開始します。ユーザーは [印刷] ダイアログボックスで印刷設定を変更できます。PrintJob.start() メソッドが正常に返された場合、次の読み取り専用プロパティに値が渡され、ユーザーの印刷設定を表します。

プロパティ	タイプ	単位	メモ
PrintJob.paperHeight	Number	ポイント	用紙全体の高さ。
PrintJob.paperWidth	Number	ポイント	用紙全体の幅。
PrintJob.pageHeight	Number	ポイント	用紙上の実際の印刷可能領域の高さ。ユーザーが設定した余白は含みません。
PrintJob.pageWidth	Number	ポイント	用紙上の実際の印刷可能領域の幅。ユーザーが設定した余白は含みません。
PrintJob.orientation	String		"portrait" または "landscape"。

ユーザーが [印刷] ダイアログボックスで [OK] をクリックすると、オペレーティングシステムへの印刷ジョブのスプールが開始されます。スプーラにページを送信するには、印刷に関する ActionScript のコマンドを発行し、PrintJob.addPage() コマンドを発行する必要があります。このメソッドに値を渡す読み取り専用の高さ、幅、方向のプロパティを使用し、印刷結果をフォーマットできます。

ユーザーが [OK] をクリックするとすぐに " ページ 1 を印刷しています " などのメッセージが表示されるので、PrintJob.addPage() コマンドと PrintJob.send() コマンドをすぐに実行することをお勧めします。

このメソッドが false を返した場合 (ユーザーがオペレーティングシステムの [印刷] ダイアログボックスで [OK] ではなく [キャンセル] をクリックした場合など) は、その後で PrintJob.addPage() や PrintJob.send() を呼び出すと失敗します。ただし、この戻り値を調べて、その結果として PrintJob.addPage() コマンドを送らなかった場合は、印刷スプールを空にするために、次のように PrintJob オブジェクトを削除する必要があります。

```
var my_pj:PrintJob = new PrintJob();
var myResult:Boolean = my_pj.start();
  if(myResult) {
  // addPage() and send() statements here
  }
delete my_pj;
```

対応バージョン : ActionScript 1.0、Flash Player 7

戻り値

Boolean - ブール値 : [印刷] ダイアログボックスでユーザーが [OK] をクリックした場合は true、ユーザーが [キャンセル] をクリックした場合、またはエラーが発生した場合は false を返します。

例

次の例では、方向のプロパティの値を使用して印刷結果を調整する方法を示します。

```
// create PrintJob object
var my_pj:PrintJob = new PrintJob();

// display print dialog box
if (my_pj.start()) {
  // boolean to track whether addPage succeeded, change this to a counter
  // if more than one call to addPage is possible
  var pageAdded:Boolean = false;

  // check the user's printer orientation setting
  // and add appropriate print area to print job
  if (my_pj.orientation == "portrait") {
  // Here, the printArea measurements are appropriate for an 8.5" x 11"
  // portrait page.
  pageAdded = my_pj.addPage(this,{xMin:0,xMax:600,yMin:0,yMax:800});
  }
  else {
  // my_pj.orientation is "landscape".
  // Now, the printArea measurements are appropriate for an 11" x 8.5"
  // landscape page.
  pageAdded = my_pj.addPage(this,{xMin:0,xMax:750,yMin:0,yMax:600});
  }
```

```
  // send pages from the spooler to the printer
  if (pageAdded) {
  my_pj.send();
  }
}

// clean up
delete my_pj;
```

関連項目

addPage (PrintJob.addPage メソッド), send (PrintJob.send メソッド)

Rectangle (flash.geom.Rectangle)

```
Object
  |
  +-flash.geom.Rectangle

public class Rectangle
extends Object
```

Rectangle クラスは、Rectangle オブジェクトの作成および変更に使用されます。Rectangle オブジェクトは、その位置 (左上隅のポイント (*x* , *y*) で示される)、および幅と高さで定義される領域です。これらの領域をデザインする際には、注意が必要です。矩形の左上隅が 0,0 で、高さが 10、幅が 20 の場合、右下隅は 9,19 となります。幅と高さのカウントが 0,0 から始まるためです。

Rectangle クラスの x、y、width、および height の各プロパティは、お互いに独立しているため、あるプロパティの値を変更しても、他のプロパティに影響はありません。ただし、right プロパティと bottom プロパティはこれら 4 つのプロパティと不可分に関連しています。right を変更すると、width も変更されます。bottom を変更すると、height も変更されます。また、width プロパティまたは right プロパティを設定する前に、left プロパティまたは *x* プロパティを設定する必要があります。

Rectangle オブジェクトは、BitmapData クラスのフィルタをサポートするために使用されます。これらのオブジェクトは、MovieClip.scrollRect プロパティでも使用され、特定の幅、高さ、およびスクロールオフセットを使用して MovieClip インスタンスをトリミングしスクロールできるようにします。

対応バージョン : ActionScript 1.0、Flash Player 8

関連項目

scrollRect (MovieClip.scrollRect プロパティ)

プロパティー覧

オプション	プロパティ	説明
	bottom:Number	y プロパティと height プロパティの合計です。
	bottomRight:Point	Rectangle オブジェクトの右下隅の位置で、そのポイントの x プロパティと y プロパティの値で決まります。
	height:Number	矩形の高さをピクセル単位で表します。
	left:Number	矩形の左上隅の *x* 座標。
	right:Number	x プロパティと width プロパティの合計です。
	size:Point	Rectangle オブジェクトのサイズで、width プロパティと height プロパティの値を持つ Point オブジェクトとして表現されます。
	top:Number	矩形の左上隅の *y* 座標。
	topLeft:Point	Rectangle オブジェクトの左上隅の位置で、そのポイントの *x* 値と *y* 値で決まります。
	width:Number	矩形の幅をピクセル単位で表します。
	x:Number	矩形の左上隅の *x* 座標。
	y:Number	矩形の左上隅の *y* 座標。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター覧

署名	説明
Rectangle(x:Number, y:Number, width:Number, height:Number)	左上隅が x パラメータと y パラメータで指定される新しい Rectangle オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	clone() : Rectangle	元の Rectangle オブジェクトと x、y、width、および height の各プロパティの値が同じである、新しい Rectangle オブジェクトを返します。
	contains(x:Number, y:Number) : Boolean	指定されたポイントがこの Rectangle オブジェクトで定義される矩形領域内にあるかどうかを判別します。
	containsPoint(pt: Point) : Boolean	指定されたポイントがこの Rectangle オブジェクトで定義される矩形領域内にあるかどうかを判別します。
	containsRectangle (rect:Rectangle) : Boolean	rect パラメータで指定された Rectangle オブジェクトがこの Rectangle オブジェクト内にあるかどうかを判別します。
	equals(toCompare: Object) : Boolean	toCompare パラメータで指定されたオブジェクトがこの Rectangle オブジェクトと等しいかどうかを判別します。
	inflate(dx:Number, dy:Number) : Void	Rectangle オブジェクトのサイズを、指定された量だけ大きくします。
	inflatePoint(pt: Point) : Void	Rectangle オブジェクトのサイズを大きくします。
	intersection (toIntersect: Rectangle) : Rectangle	toIntersect パラメータで指定された Rectangle オブジェクトがこの Rectangle オブジェクトと交差する場合に、intersection() メソッドは、交差領域を Rectangle オブジェクトとして返します。
	intersects (toIntersect: Rectangle) : Boolean	toIntersect パラメータで指定されたオブジェクトがこの Rectangle オブジェクトと交差するかどうかを判別します。
	isEmpty() : Boolean	この Rectangle オブジェクトが空かどうかを判別します。
	offset(dx:Number, dy:Number) : Void	左上隅で決まる Rectangle オブジェクトの位置を、指定された量だけ調整します。
	offsetPoint (pt:Point) : Void	Point オブジェクトをパラメータとして使用して、Rectangle オブジェクトの位置を調整します。
	setEmpty() : Void	Rectangle オブジェクトのすべてのプロパティを 0 に設定します。
	toString() : String	Rectangle オブジェクトの水平位置と垂直位置、および幅と高さをリストするストリングを作成して返します。
	union(toUnion: Rectangle) : Rectangle	2 つの矩形間の水平と垂直の空間を塗りつぶすことにより、2 つの矩形を加算して新しい Rectangle オブジェクトを作成します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

bottom (Rectangle.bottom プロパティ)

public bottom : Number

y プロパティと height プロパティの合計です。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、Rectangle オブジェクトを作成し、その bottom プロパティの値を 15 から 30 に変更します。rect.height の値も 10 から 25 に変わることに注意してください。

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(5, 5, 10, 10);
trace(rect.height); // 10
trace(rect.bottom); // 15

rect.bottom = 30;
trace(rect.height); // 25
trace(rect.bottom); // 30
```

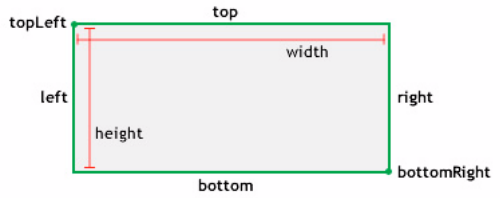
関連項目

y (Rectangle.y プロパティ), height (Rectangle.height プロパティ)

bottomRight (Rectangle.bottomRight プロパティ)

```
public bottomRight : Point
```

Rectangle オブジェクトの右下隅の位置で、そのポイントの x プロパティと y プロパティの値で決まります。



対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、Point オブジェクトの値を使用して、Rectangle オブジェクトの `bottomRight` プロパティを設定します。`rect.width` と `rect.height` も変わることに注意してください。

```
import flash.geom.Rectangle;
import flash.geom.Point;

var rect:Rectangle = new Rectangle(1, 2, 4, 8);
trace(rect.bottom); // 10
trace(rect.right); // 5
trace(rect.height); // 8
trace(rect.width); // 4

var myBottomRight:Point = new Point(16, 32);
rect.bottomRight = myBottomRight;
trace(rect.bottom); // 32
trace(rect.right); // 16
trace(rect.height); // 30
trace(rect.width); // 15
```

関連項目

Point (flash.geom.Point)

clone (Rectangle.clone メソッド)

```
public clone() : Rectangle
```

元の Rectangle オブジェクトと x、y、width、および height の各プロパティの値が同じである、新しい Rectangle オブジェクトを返します。

対応バージョン : ActionScript 1.0、Flash Player 8

戻り値

Rectangle - 元の Rectangle オブジェクトと x、y、width、および height の各プロパティの値が同じである、新しい Rectangle オブジェクト

例

次の例では、3 つの Rectangle オブジェクトを作成し、それらを比較します。rect_1 は、Rectangle コンストラクタを使用して作成されます。rect_2 は、rect_1 と等しい値に設定することにより作成されます。clonedRect は、rect_1 のクローンを作成することにより作成されます。rect_2 が rect_1 と等しいと評価された場合に、rect_1 と同じ値を含んでいても clonedRect は等しいと評価されないことに注意してください。

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(1, 2, 4, 8);
var rect_2:Rectangle = rect_1;
var clonedRect:Rectangle = rect_1.clone();

trace(rect_1 == rect_2); // true
trace(rect_1 == clonedFilter); // false

for(var i in rect_1) {
  trace(">> " + i + ": " + rect_1[i]);
  >> toString: [type Function]
  >> equals: [type Function]
  >> union: [type Function]
  >> intersects: [type Function]
  >> intersection: [type Function]
  >> containsRectangle: [type Function]
  >> containsPoint: [type Function]
  >> contains: [type Function]
  >> offsetPoint: [type Function]
  >> offset: [type Function]
  >> inflatePoint: [type Function]
  >> inflate: [type Function]
  >> size: (x=4, y=8)
  >> bottomRight: (x=5, y=10)
  >> topLeft: (x=1, y=2)
  >> bottom: 10
  >> top: 2
```

```
  >> right: 5
  >> left: 1
  >> isEmpty: [type Function]
  >> setEmpty: [type Function]
  >> clone: [type Function]
  >> height: 8
  >> width: 4
  >> y: 2
  >> x: 1
}

for(var i in clonedRect) {
  trace(">> " + i + ": " + clonedRect[i]);
  >> toString: [type Function]
  >> equals: [type Function]
  >> union: [type Function]
  >> intersects: [type Function]
  >> intersection: [type Function]
  >> containsRectangle: [type Function]
  >> containsPoint: [type Function]
  >> contains: [type Function]
  >> offsetPoint: [type Function]
  >> offset: [type Function]
  >> inflatePoint: [type Function]
  >> inflate: [type Function]
  >> size: (x=4, y=8)
  >> bottomRight: (x=5, y=10)
  >> topLeft: (x=1, y=2)
  >> bottom: 10
  >> top: 2
  >> right: 5
  >> left: 1
  >> isEmpty: [type Function]
  >> setEmpty: [type Function]
  >> clone: [type Function]
  >> height: 8
  >> width: 4
  >> y: 2
  >> x: 1
}
```

rect_1、rect_2、および clonedRect の関係をさらに詳しく示すために、次の例では、rect_1 の x プロパティを変更します。x を変更することは、clone() メソッドによって、rect_1 を参照する代わりにその値に基づいて新しいインスタンスが作成されることを示します。

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(1, 2, 4, 8);
var rect_2:Rectangle = rect_1;
var clonedRect:Rectangle = rect_1.clone();
```

```
trace(rect_1.x); // 1
trace(rect_2.x); // 1
trace(clonedRect.x); // 1

rect_1.x = 10;

trace(rect_1.x); // 10
trace(rect_2.x); // 10
trace(clonedRect.x); // 1
```

関連項目

x（Rectangle.x プロパティ）, y（Rectangle.y プロパティ）, width（Rectangle.width プロパティ）, height（Rectangle.height プロパティ）

contains (Rectangle.contains メソッド)

```
public contains(x:Number, y:Number) : Boolean
```

指定されたポイントがこの Rectangle オブジェクトで定義される矩形領域内にあるかどうかを判別します。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

x:Number - ポイントの *x* 値 (水平位置)。

y:Number - ポイントの *y* 値 (垂直位置)。

戻り値

Boolean - 指定されたポイントが Rectangle オブジェクト内にある場合は `true` を、Rectangle オブジェクト内にない場合は `false` を返します。

例

次の例では、Rectangle オブジェクトを作成し、3 つの座標ペアのそれぞれが境界内に収まるかどうかをテストします。

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(10, 10, 50, 50);
trace(rect.contains(59, 59)); // true
trace(rect.contains(10, 10)); // true
trace(rect.contains(60, 60)); // false
```

関連項目

Point (flash.geom.Point)

containsPoint (Rectangle.containsPoint メソッド)

```
public containsPoint(pt:Point) : Boolean
```

指定されたポイントがこの Rectangle オブジェクトで定義される矩形領域内にあるかどうかを判別します。このメソッドは、Point オブジェクトをパラメータとして使用することを除けば、`Rectangle.contains()` メソッドと似ています。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`pt:Point` - *x* および *y* の値で表されるポイント。

戻り値

`Boolean` - 指定されたポイントが Rectangle オブジェクト内にある場合は `true` を、Rectangle オブジェクト内にない場合は `false` を返します。

例

次の例では、1 つの Rectangle オブジェクトおよび 3 つの Point オブジェクトを作成し、各ポイントが矩形の境界内に収まるかどうかをテストします。

```
import flash.geom.Rectangle;
import flash.geom.Point;

var rect:Rectangle = new Rectangle(10, 10, 50, 50);
trace(rect.containsPoint(new Point(10, 10))); // true
trace(rect.containsPoint(new Point(59, 59))); // true
trace(rect.containsPoint(new Point(60, 60))); // false
```

関連項目

`contains (Rectangle.contains メソッド )`, `Point (flash.geom.Point)`

containsRectangle (Rectangle.containsRectangle メソッド)

```
public containsRectangle(rect:Rectangle) : Boolean
```

`rect` パラメータで指定された Rectangle オブジェクトがこの Rectangle オブジェクト内にあるかどうかを判別します。2 番目の Rectangle オブジェクトが最初の Rectangle オブジェクトの境界内に完全に収まる場合、最初の Rectangle オブジェクトは 2 番目の Rectangle オブジェクトを包含しているといいます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`rect:Rectangle` - チェック対象の Rectangle オブジェクト。

戻り値

`Boolean` - 指定する Rectangle オブジェクトがこの Rectangle オブジェクトに含まれている場合は `true` を返します。それ以外の場合は `false` を返します。

例

次の例では、4 つの新しい Rectangle オブジェクトを作成し、矩形 A が矩形 B、C、D を含んでいるかどうかを判別します。

```
import flash.geom.Rectangle;

var rectA:Rectangle = new Rectangle(10, 10, 50, 50);
var rectB:Rectangle = new Rectangle(10, 10, 50, 50);
var rectC:Rectangle = new Rectangle(10, 10, 51, 51);
var rectD:Rectangle = new Rectangle(15, 15, 45, 45);

trace(rectA.containsRectangle(rectB)); // true
trace(rectA.containsRectangle(rectC)); // false
trace(rectA.containsRectangle(rectD)); // true
```

equals (Rectangle.equals メソッド)

```
public equals(toCompare:Object) : Boolean
```

`toCompare` パラメータで指定されたオブジェクトがこの Rectangle オブジェクトと等しいかどうかを判別します。このメソッドは、オブジェクトの `x`、`y`、`width`、および `height` の各プロパティを、この Rectangle オブジェクトの同じプロパティと比較します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`toCompare:Object` - この Rectangle オブジェクトと比較する矩形

戻り値

`Boolean` - オブジェクトの `x`、`y`、`width`、および `height` の各プロパティの値がこの Rectangle オブジェクトと等しい場合は `true` を返します。それ以外の場合は `false` を返します。

例

次の例では、rect_1 と rect_2 は等しいですが、rect_3 はこれら 2 つのオブジェクトと等しくありません。これは、rect_3 の x、y、width、および height の各プロパティが rect_1 と rect_2 のそれぞれのプロパティと等しくないからです。

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(0, 0, 50, 100);
var rect_2:Rectangle = new Rectangle(0, 0, 50, 100);
var rect_3:Rectangle = new Rectangle(10, 10, 60, 110);

trace(rect_1.equals(rect_2)); // true;
trace(rect_1.equals(rect_3)); // false;
```

このメソッドの署名では抽象的なオブジェクトのみを対象とし、他の Rectangle インスタンスだけが等しいものとして処理されます。

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(0, 0, 50, 100);
var nonRect:Object = new Object();
nonRect.x = 0;
nonRect.y = 0;
nonRect.width = 50;
nonRect.height = 100;
trace(rect_1.equals(nonRect));
```

関連項目

x (Rectangle.x プロパティ), y (Rectangle.y プロパティ), width (Rectangle.width プロパティ), height (Rectangle.height プロパティ)

height (Rectangle.height プロパティ)

```
public height : Number
```

矩形の高さをピクセル単位で表します。Rectangle オブジェクトの height の値を変更しても、x、y、および width の各プロパティに影響はありません。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、Rectangle オブジェクトを作成し、その height プロパティを 10 から 20 に変更します。rect.bottom も変わることに注意してください。

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(5, 5, 10, 10);
trace(rect.height); // 10
trace(rect.bottom); // 15

rect.height = 20;
trace(rect.height); // 20
trace(rect.bottom); // 25
```

関連項目

x (Rectangle.x プロパティ), y (Rectangle.y プロパティ), width (Rectangle.width プロパティ)

inflate (Rectangle.inflate メソッド)

```
public inflate(dx:Number, dy:Number) : Void
```

Rectangle オブジェクトのサイズを、指定された量だけ大きくします。Rectangle オブジェクトの中心点は変わりませんが、サイズは dx 値に応じて左右に大きくなり、dy 値に応じて上下に大きくなります。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

dx:Number - Rectangle オブジェクトの左右に加わる値。次の等式を使用して、矩形の新しい幅と位置が計算されます。

```
x -= dx;
width += 2 * dx;
```

dy:Number - Rectangle オブジェクトの上下に加わる値。次の等式を使用して、矩形の新しい高さと位置が計算されます。

```
y -= dy;
height += 2 * dy;
```

例

次の例では、Rectangle オブジェクトを作成し、その width プロパティの値を 16 * 2 (32) だけ大きくし、height プロパティの値を 32 * 2 (64) だけ大きくします。

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(1, 2, 4, 8);
trace(rect.toString()); // (x=1, y=2, w=4, h=8)

rect.inflate(16, 32);
trace(rect.toString()); // (x=-15, y=-30, w=36, h=72)
```

関連項目

x (Rectangle.x プロパティ),y (Rectangle.y プロパティ)

inflatePoint (Rectangle.inflatePoint メソッド)

```
public inflatePoint(pt:Point) : Void
```

Rectangle オブジェクトのサイズを大きくします。このメソッドは、Point オブジェクトをパラメータとして使用することを除けば、Rectangle.inflate() メソッドと似ています。

次の 2 つのコード例の結果は同じになります。

```
rect1 = new flash.geom.Rectangle(0,0,2,5);
rect1.inflate(2,2)
rect1 = new flash.geom.Rectangle(0,0,2,5);
pt1 = new flash.geom.Point(2,2);
rect1.inflatePoint(pt1)
```

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

pt:Point - 矩形をポイントの *x* 座標値と *y* 座標値だけ大きくします。

例

次の例では、Rectangle オブジェクトを作成し、ポイントで見つかった *x* (水平) と *y* (垂直) の量だけ膨張させます。

```
import flash.geom.Rectangle;
import flash.geom.Point;

var rect:Rectangle = new Rectangle(0, 0, 2, 5);
trace(rect.toString()); // (x=0, y=0, w=2, h=5

var myPoint:Point = new Point(2, 2);
rect.inflatePoint(myPoint);
trace(rect.toString()); // (x=-2, y=-2, w=6, h=9)
```

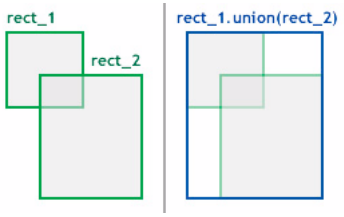
関連項目

Point (flash.geom.Point)

intersection (Rectangle.intersection メソッド)

```
public intersection(toIntersect:Rectangle) : Rectangle
```

toIntersect パラメータで指定された Rectangle オブジェクトがこの Rectangle オブジェクトと交差する場合に、intersection() メソッドは、交差領域を Rectangle オブジェクトとして返します。矩形が交差しない場合、このメソッドは、プロパティが 0 に設定された空の Rectangle オブジェクトを返します。



対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

toIntersect:Rectangle - この Rectangle オブジェクトと交差するかどうかを調べる対象の Rectangle オブジェクト

戻り値

Rectangle - 交差領域と等しい Rectangle オブジェクト。矩形が交差しない場合、このメソッドは x、y、width、および height の各プロパティが 0 に設定された空の Rectangle オブジェクトを返します。

例

次の例では、rect_1 が rect_2 と交差する領域を判別します。

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(0, 0, 50, 50);
var rect_2:Rectangle = new Rectangle(25, 25, 100, 100);
var intersectingArea:Rectangle = rect_1.intersection(rect_2);
trace(intersectingArea.toString()); // (x=25, y=25, w=25, h=25)
```

intersects (Rectangle.intersects メソッド)

```
public intersects(toIntersect:Rectangle) : Boolean
```

toIntersect パラメータで指定されたオブジェクトがこの Rectangle オブジェクトと交差するかどうかを判別します。このメソッドは、指定された Rectangle オブジェクトの x、y、width、および height の各プロパティをチェックして、この Rectangle オブジェクトと交差するかどうかを調べます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

toIntersect:Rectangle - この Rectangle オブジェクトと比較する Rectangle オブジェクト。

戻り値

Boolean - 指定されたオブジェクトがこの Rectangle オブジェクトと交差する場合は true を返します。交差しない場合は false を返します。

例

次の例では、rectA が rectB または rectC と交差するかどうかを判別します。

```
import flash.geom.Rectangle;
var rectA:Rectangle = new Rectangle(10, 10, 50, 50);
var rectB:Rectangle = new Rectangle(59, 59, 50, 50);
var rectC:Rectangle = new Rectangle(60, 60, 50, 50);
var rectAIntersectsB:Boolean = rectA.intersects(rectB);
var rectAIntersectsC:Boolean = rectA.intersects(rectC);
trace(rectAIntersectsB); // true
trace(rectAIntersectsC); // false

var firstPixel:Rectangle = new Rectangle(0, 0, 1, 1);
var adjacentPixel:Rectangle = new Rectangle(1, 1, 1, 1);
var pixelsIntersect:Boolean = firstPixel.intersects(adjacentPixel);
trace(pixelsIntersect); // false
```

関連項目

x (Rectangle.x プロパティ), y (Rectangle.y プロパティ), width (Rectangle.width プロパティ), height (Rectangle.height プロパティ)

isEmpty (Rectangle.isEmpty メソッド)

```
public isEmpty() : Boolean
```

この Rectangle オブジェクトが空かどうかを判別します。

対応バージョン： ActionScript 1.0、Flash Player 8

戻り値

Boolean - Rectangle オブジェクトの幅と高さが 0 以下の場合は true を返します。それ以外の場合は false を返します。

例

次の例では、空の Rectangle オブジェクトを作成し、空であることを確認します。

```
import flash.geom.*;
var rect:Rectangle = new Rectangle(1, 2, 0, 0);
trace(rect.toString()); // (x=1, y=2, w=0, h=0)
trace(rect.isEmpty()); // true
```

次の例では、空ではない Rectangle を作成し、空になるようにします。

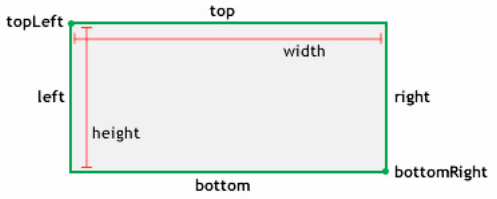
```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(1, 2, 4, 8);
trace(rect.isEmpty()); // false
rect.width = 0;
trace(rect.isEmpty()); // true
rect.width = 4;
trace(rect.isEmpty()); // false
rect.height = 0;
trace(rect.isEmpty()); // true
```

left (Rectangle.left プロパティ)

```
public left : Number
```

矩形の左上隅の *x* 座標。Rectangle オブジェクトの left プロパティ値を変更しても、y および height の各プロパティに影響はありません。ただし、これは width プロパティに影響しますが、x 値の変更は width プロパティに影響しません。

left プロパティの値は、x プロパティの値と等価です。



対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、left プロパティを 0 から 10 に変更します。rect.x と同様、rect.width も変わることに注意してください。

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(0, 0, 100, 100);
trace(rect.left); // 0
trace(rect.x); // 0
trace(rect.width); // 100

rect.left = 10;
trace(rect.left); // 10
trace(rect.x); // 10
trace(rect.width); // 90
```

関連項目

x （Rectangle.x プロパティ）, y （Rectangle.y プロパティ）, width （Rectangle.width プロパティ）, height （Rectangle.height プロパティ）

offset (Rectangle.offset メソッド)

```
public offset(dx:Number, dy:Number) : Void
```

左上隅で決まる Rectangle オブジェクトの位置を、指定された量だけ調整します。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

dx:Number - Rectangle オブジェクトの *x* 値をこの量だけ移動します。

dy:Number - Rectangle オブジェクトの *y* 値をこの量だけ移動します。

例

次の例では、Rectangle オブジェクトを作成し、その *x* 値と *y* 値をそれぞれ 5 と 10 ずつオフセットします。

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(1, 2, 4, 8);
trace(rect.toString()); // (x=1, y=2, w=4, h=8)

rect.offset(16, 32);
trace(rect.toString()); // (x=17, y=34, w=4, h=8)
```

offsetPoint (Rectangle.offsetPoint メソッド)

```
public offsetPoint(pt:Point) : Void
```

Point オブジェクトをパラメータとして使用して、Rectangle オブジェクトの位置を調整します。このメソッドは、Point オブジェクトをパラメータとして使用することを除けば、Rectangle.offset() メソッドと似ています。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

pt:Point - この Rectangle オブジェクトをオフセットするための Point オブジェクト。

例

次の例では、ポイントで見つかった値を使用して、Rectangle をオフセットします。

```
import flash.geom.Rectangle;
import flash.geom.Point;

var rect:Rectangle = new Rectangle(1, 2, 4, 8);
trace(rect.toString()); // (x=1, y=2, w=4, h=8)

var myPoint:Point = new Point(16, 32);
rect.offsetPoint(myPoint);
trace(rect.toString()); // (x=17, y=34, w=4, h=8)
```

関連項目

Point (flash.geom.Point)

Rectangle コンストラクタ

```
public Rectangle(x:Number, y:Number, width:Number, height:Number)
```

左上隅が x パラメータと y パラメータで指定される新しい Rectangle オブジェクトを作成します。パラメータなしでこのコンストラクタ関数を呼び出すと、x、y、width、および height の各プロパティが 0 に設定された矩形が作成されます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

x:Number - 矩形の左上隅の *x* 座標。

y:Number - 矩形の左上隅の *y* 座標。

width:Number - 矩形の幅 (ピクセル単位)。

height:Number - 矩形の高さ (ピクセル単位)。

例

次の例では、指定されたパラメータで Rectangle オブジェクトを作成します。

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(5, 10, 50, 100);
trace(rect.toString()); // (x=5, y=10, w=50, h=100)
```

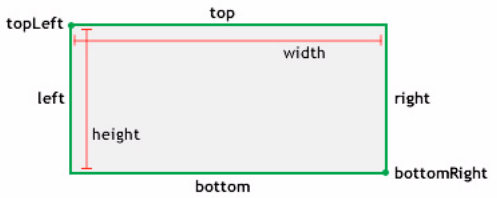
関連項目

x (Rectangle.x プロパティ),y (Rectangle.y プロパティ),width (Rectangle.width プロパティ),height (Rectangle.height プロパティ)

right (Rectangle.right プロパティ)

```
public right : Number
```

x プロパティと width プロパティの合計です。



対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、Rectangle オブジェクトを作成し、その right プロパティを 15 から 30 に変更します。rect.width も変わることに注意してください。

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(5, 5, 10, 10);
trace(rect.width); // 10
trace(rect.right); // 15

rect.right = 30;
trace(rect.width); // 25
trace(rect.right); // 30
```

関連項目

x (Rectangle.x プロパティ),width (Rectangle.width プロパティ)

setEmpty (Rectangle.setEmpty メソッド)

```
public setEmpty() : Void
```

Rectangle オブジェクトのすべてのプロパティを 0 に設定します。その幅または高さが 0 以下の場合、Rectangle オブジェクトは空です。

このメソッドは、x、y、width、および height の各プロパティの値を 0 に設定します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、空ではない Rectangle オブジェクトを作成し、空になるようにします。

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(5, 10, 50, 100);
trace(rect.isEmpty()); // false
rect.setEmpty();
trace(rect.isEmpty()); // true
```

関連項目

x （Rectangle.x プロパティ),y （Rectangle.y プロパティ),width （Rectangle.width プロパティ),height （Rectangle.height プロパティ ）

size (Rectangle.size プロパティ)

```
public size : Point
```

Rectangle オブジェクトのサイズで、width プロパティと height プロパティの値を持つ Point オブジェクトとして表現されます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、Rectangle オブジェクトを作成し、そのサイズ (size) を取得し、そのサイズ (size) を変更し、Rectangle オブジェクトに新しい値を設定します。size プロパティで使用される Point オブジェクトは、*x* 値と *y* 値を使用して、Rectangle オブジェクトの width プロパティと height プロパティを表します。

```
import flash.geom.Rectangle;
import flash.geom.Point;

var rect:Rectangle = new Rectangle(1, 2, 4, 8);
var size:Point = rect.size;
trace(size.x); // 4;
trace(size.y); // 8;
```

```
size.x = 16;
size.y = 32;
rect.size = size;
trace(rect.x); // 1
trace(rect.y); // 2
trace(rect.width); // 16
trace(rect.height); // 32
```

関連項目

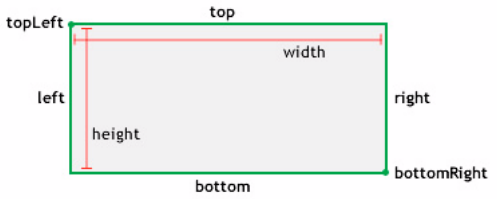
Point (flash.geom.Point)

top (Rectangle.top プロパティ)

public top : Number

矩形の左上隅の y 座標。Rectangle オブジェクトの top プロパティを変更しても、x および width の各プロパティに影響はありません。ただし、これは height プロパティに影響しますが、y 値の変更は height プロパティに影響しません。

top プロパティの値は、y プロパティの値と等価です。



対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、top プロパティの値を 0 から 10 に変更します。rect.height と同様、rect.y も変わることに注意してください。

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(0, 0, 100, 100);
trace(rect.top); // 0
trace(rect.y); // 0
trace(rect.height); // 100


rect.top = 10;
trace(rect.top); // 10
trace(rect.y); // 10
trace(rect.height); // 90
```

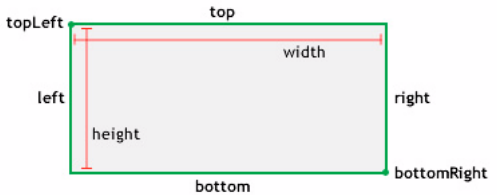
関連項目

x (Rectangle.x プロパティ), y (Rectangle.y プロパティ), width (Rectangle.width プロパティ), height (Rectangle.height プロパティ)

topLeft (Rectangle.topLeft プロパティ)

public topLeft : Point

Rectangle オブジェクトの左上隅の位置で、そのポイントの x 値と y 値で決まります。



対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、Point オブジェクトの値を使用して、Rectangle オブジェクトの topLeft プロパティを設定します。rect.x と rect.y も変わることに注意してください。

```
import flash.geom.Rectangle;
import flash.geom.Point;

var rect:Rectangle = new Rectangle();
trace(rect.left); // 0
trace(rect.top); // 0
trace(rect.x); // 0
trace(rect.y); // 0

var myTopLeft:Point = new Point(5, 15);
rect.topLeft = myTopLeft;
trace(rect.left); // 5
trace(rect.top); // 15
trace(rect.x); // 5
trace(rect.y); // 15
```

関連項目

Point (flash.geom.Point), x (Rectangle.x プロパティ), y (Rectangle.y プロパティ)

toString (Rectangle.toString メソッド)

```
public toString() : String
```

Rectangle オブジェクトの水平位置と垂直位置、および幅と高さをリストするストリングを作成して返します。

対応バージョン : ActionScript 1.0、Flash Player 8

戻り値

`String` - Rectangle オブジェクトの x、y、width、および height の各プロパティの値を列挙するストリング。

例

次の例では、rect_1 のストリング表現を役に立つデバッグ用のテキストと連結します。

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(0, 0, 50, 100);
trace("Rectangle 1 : " + rect_1.toString()); // Rectangle 1 : (x=0, y=0, w=50,
  h=100)
```

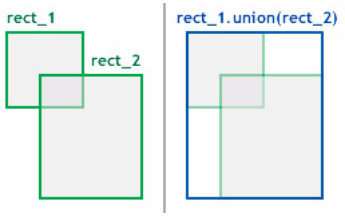
関連項目

x （Rectangle.x プロパティ),y （Rectangle.y プロパティ),width （Rectangle.width プロパティ),height （Rectangle.height プロパティ)

union (Rectangle.union メソッド)

```
public union(toUnion:Rectangle) : Rectangle
```

2 つの矩形間の水平と垂直の空間を塗りつぶすことにより、2 つの矩形を加算して新しい Rectangle オブジェクトを作成します。



対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`toUnion:Rectangle` - この Rectangle オブジェクトに追加する Rectangle オブジェクト。

戻り値

Rectangle - 2 つの矩形の和集合である新しい Rectangle オブジェクト。

例

次の例では、他の 2 つの矩形の和集合から Rectangle オブジェクトを作成します。

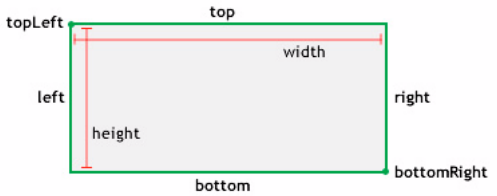
たとえば、プロパティ x=20、y=50、width=60、および height=30 (20, 50, 60, 30) の矩形と、プロパティ (150, 130, 50, 30) の別の矩形があるとします。この 2 つの矩形の和集合は、2 つの矩形を包含する、プロパティ (20, 50, 180, 110) のより大きな矩形になります。

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(20, 50, 60, 30);
var rect_2:Rectangle = new Rectangle(150, 130, 50, 30);
var combined:Rectangle = rect_1.union(rect_2);
trace(combined.toString()); // (x=20, y=50, w=180, h=110)
```

width (Rectangle.width プロパティ)

```
public width : Number
```

矩形の幅をピクセル単位で表します。Rectangle オブジェクトの width プロパティ値を変更しても、x、y、および height の各プロパティに影響はありません。



対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、Rectangle オブジェクトを作成し、その width プロパティを 10 から 20 に変更します。rect.right も変わることに注意してください。

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(5, 5, 10, 10);
trace(rect.width); // 10
trace(rect.right); // 15

rect.width = 20;
trace(rect.width); // 20
trace(rect.right); // 25
```

関連項目

x（Rectangle.x プロパティ）, y（Rectangle.y プロパティ）, height（Rectangle.height プロパティ）

x (Rectangle.x プロパティ)

public x : Number

矩形の左上隅の *x* 座標。Rectangle オブジェクトの *x* プロパティ値を変更しても、y、width、および height の各プロパティに影響はありません。矩形の左上隅の *x* 座標。Rectangle オブジェクトの left プロパティを変更しても、y および height の各プロパティに影響はありません。ただし、x 値の変更が width プロパティに影響しないのとは異なり、これは width プロパティに影響します。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、空の Rectangle オブジェクトを作成し、その x プロパティを 10 に設定します。rect.left も変わることに注意してください。

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle();
trace(rect.x); // 0
trace(rect.left); // 0

rect.x = 10;
trace(rect.x); // 10
trace(rect.left); // 10
```

関連項目

left（Rectangle.left プロパティ）

y (Rectangle.y プロパティ)

public y : Number

矩形の左上隅の y 座標。Rectangle オブジェクトの y プロパティ値を変更しても、x、width、および height の各プロパティに影響はありません。

y プロパティの値は、top プロパティの値と等価です。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、空の Rectangle オブジェクトを作成し、その y プロパティを 10 に設定します。rect.top も変わることに注意してください。

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle();
trace(rect.y); // 0
trace(rect.top); // 0

rect.y = 10;
trace(rect.y); // 10
trace(rect.top); // 10
```

関連項目

x (Rectangle.x プロパティ), width (Rectangle.width プロパティ), height (Rectangle.height プロパティ), top (Rectangle.top プロパティ)

security (System.security)

```
Object
  |
  +-System.security

public class security
extends Object
```

System.security クラスには、異なるドメインに属する複数の SWF ファイルが互いにどのようにやりとりするかを指定するためのメソッドがあります。

詳細については、次の参照先を参照してください。

- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

対応バージョン : ActionScript 1.0、Flash Player 6

プロパティ一覧

オプション	プロパティ	説明
static	sandboxType:String (読み取り専用)	呼び出し元の SWF ファイルが動作しているセキュリティサンドボックスのタイプを示します。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

メソッド一覧

オプション	署名	説明
static	allowDomain(domain1: String) : Void	指定されたドメインの SWF ファイルおよび HTML ファイルが、allowDomain() 呼び出しを含む SWF ファイルのオブジェクトおよび変数にアクセスすることを許可します。
static	allowInsecureDomain (domain:String) : Void	指定したドメイン内の SWF ファイルおよび HTML ファイルが、HTTPS プロトコルでホストされた呼び出し元 SWF ファイルのオブジェクトと変数にアクセスすることを許可します。
static	loadPolicyFile(url: String) : Void	url パラメータで指定された場所からドメイン間ポリシーファイルをロードします。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

allowDomain (security.allowDomain メソッド)

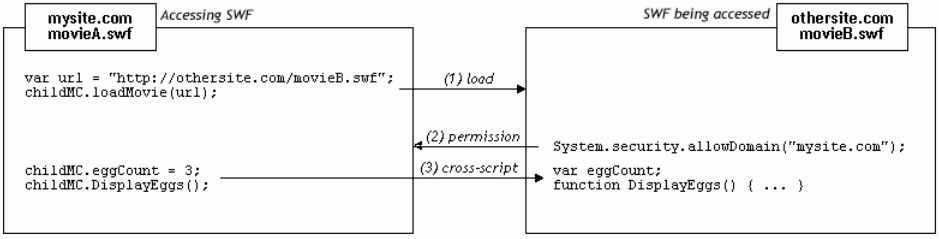
public static allowDomain(domain1:String) : Void

指定されたドメインの SWF ファイルおよび HTML ファイルが、allowDomain() 呼び出しを含む SWF ファイルのオブジェクトおよび変数にアクセスすることを許可します。

たとえば http://mysite.com/movieA.swf と http://mysite.com/movieB.swf のように、2 つの SWF ファイルが同じドメインからサービスを受けている場合、movieA.swf で movieB.swf 内の変数、オブジェクト、プロパティ、メソッドなどを調査、変更でき、movieB.swf でも movieA.swf に対して同様のことを実行できます。これは、クロスムービースクリプトまたは単にクロススクリプトと呼ばれます。

たとえば http://mysite.com/movieA.swf と http://othersite.com/movieB.swf のように、2 つの SWF ファイルが異なるドメインからサービスを受けている場合、デフォルトでは、movieA.swf で movieB.swf をスクリプトすることも、movieB.swf で movieA.swf をスクリプトすることも許可されません。SWF ファイルでは、他のドメインの SWF ファイルに対して、`System.security.allowDomain()` を呼び出すことで、自分自身をスクリプトする許可を与えます。これをクロスドメインスクリプトと呼びます。movieB.swf では `System.security.allowDomain("mysite.com")` を呼び出すことで、movieA.swf に movieB.swf をスクリプトする許可を与えます。

クロスドメインの場合は、2 つのドメインが関与するため、どちらの側を対象にしているかを明確にすることが重要です。説明のため、ここでは、クロススクリプトを実行する側をアクセス元 (通常、アクセスする SWF) と呼び、他をアクセス先 (通常、アクセスされる SWF) と呼びます。例の説明を続けます。movieA.swf が movieB.swf をスクリプトする場合、movieA.swf がアクセス元で、movieB.swf がアクセス先となります。



`System.security.allowDomain()` を使用して確立されるクロスドメイン許可は、非対称です。前の例では、movieA.swf は movieB.swf をスクリプトできますが、movieB.swf は movieA.swf をスクリプトできません。movieA.swf で、movieA.swf をスクリプトする許可を othersite.com に与える `System.security.allowDomain()` を呼び出していないためです。対称的な許可を設定するには、両方の SWF ファイルで System.security.allowDomain() を呼び出すようにする必要があります。

Flash Player では、SWF ファイルを他の SWF ファイルによるクロスドメインスクリプトから保護するだけでなく、HTML ファイル によるクロスドメインスクリプトからも保護します。HTML から SWF へのスクリプトは、`SetVariable` などの古い Flash ブラウザを使用するか、`ExternalInterface.addCallback()` で確立したコールバックを使用することで実行できます。HTML から SWF へのスクリプトがドメインの境界を横切る場合、アクセス先 SWF ファイルは、アクセス元が SWF ファイルの場合と同様に、`System.security.allowDomain()` を呼び出す必要があります。さもなければ、操作は失敗します。

`System.security.allowDomain()` にパラメータとして IP アドレスを指定しても、指定された IP アドレスに存在するすべてのアクセス元からのアクセスが許可されるわけではありません。許可されるのは、その IP アドレスにマッピングされるドメイン名ではなく、URL にその IP アドレスを明示的に指定してロードされたアクセス元からのアクセスだけです。

バージョン固有の相違点 Flash Player のクロスドメインセキュリティ規則は、バージョンを追うごとに発展してきました。次の表は、相違点をまとめたものです。

クロススクリプト操作に関係する最新の SWF のバージョン	allowDomain() の必要性	allowInsecureDomain() の必要性	allowDomain() または allowInsecureDomain() を呼び出す必要がある SWF	allowDomain() または allowInsecureDomain() に指定できる項目
5 以前	いいえ	いいえ	該当なし	該当なし
6	はい (スーパードメインが一致しない場合)	いいえ	アクセス先 SWF ファイル、またはアクセス先 SWF ファイルと同じスーパードメインにある任意の SWF ファイル	■ テキストベースのドメイン (mysite.com) ■ IP アドレス (192.168.1.1)
7	はい (ドメインが完全に一致しない場合)	はい (HTTP から HTTPS へのアクセスを実行する場合 (ドメインが完全に一致する場合でも))	アクセス先 SWF ファイル、またはアクセス先 SWF ファイルとまったく同じドメインにある任意の SWF ファイル	■ テキストベースのドメイン (mysite.com) ■ IP アドレス (192.168.1.1)
8 以降	はい (ドメインが完全に一致しない場合)	はい (HTTP から HTTPS へのアクセスを実行する場合 (ドメインが完全に一致する場合でも))	アクセス先 SWF	■ テキストベースのドメイン (mysite.com) ■ IP アドレス (192.168.1.1) ■ ワイルドカード (*)

Flash Player の動作を制御するバージョンは、Flash Player 自身のバージョンでなく、SWF バージョン (SWF ファイルのパブリッシュバージョン) です。たとえば、Flash Player 8 でバージョン 7 用にパブリッシュされた SWF ファイルを再生する場合、バージョン 7 と一貫した動作が適用されます。これによって、アップグレードしても、展開された SWF ファイル内の `System.security.allowDomain()` の動作は変更されなくなります。

前の表のバージョン列は、クロススクリプト操作に関係する最新の SWF のバージョンを示しています。Flash Player の動作は、アクセス元 SWF ファイルまたはアクセス先 SWF ファイルのバージョンのいずれか新しい方のバージョンによって決まります。

次の段落では、System.security.allowDomain() に関係する Flash Player のセキュリティ機能の変更について詳細に説明します。

バージョン 5: クロスドメインスクリプトの制限はありません。

バージョン 6: クロスドメインスクリプトのセキュリティが導入されました。デフォルトでは、クロスドメインスクリプトは禁止されています。System.security.allowDomain() で許可できます。2 つのファイルが同じドメインに属しているかどうかを判別するため、各ファイルのスーパードメインが使用されます。このスーパードメインは、ファイルの URL の完全なホスト名から最初のセグメントを除き、最低 2 セグメントにまでしたものです。たとえば、www.mysite.com のスーパードメインは、mysite.com となります。この場合、www.mysite.com の SWF ファイルと store.mysite.com の SWF ファイルは、System.security.allowDomain() を呼び出さずに、お互いにスクリプトを実行できるようになります。

バージョン 7: スーパードメイン一致が、完全なドメイン一致に変更されました。2 つのファイルでお互いにスクリプトの実行が許可されるのは、それらの URL のホスト名が同じ場合だけです。それ以外の場合は、System.security.allowDomain() を呼び出す必要があります。デフォルトでは、HTTPS 以外の URL からロードしたファイルは、まったく同じドメインからファイルをロードしたとしても、HTTPS の URL からロードしたファイルをスクリプトできません。この制限は HTTPS ファイルの保護に役立ちます。HTTPS 以外のファイルは、ダウンロード中に変更されやすく、HTTPS 以外のファイルが悪意を持って変更されると、このような不正操作を受けにくい HTTPS ファイルを損傷する可能性があります。アクセス先の HTTPS SWF ファイルが自発的にこの制限を無効にできるように、System.security.allowInsecureDomain() を導入しましたが、System.security.allowInsecureDomain() を使用しないことをお勧めします。

バージョン 8:2 つの主要な変更点は次のとおりです。

- System.security.allowDomain() を呼び出した場合、アクセス先 SWF ファイルが System.security.allowDomain() を呼び出した SWF ファイルであるときにのみ、クロススクリプト操作が許可されるようになりました。つまり、System.security.allowDomain() を呼び出した SWF ファイルのみが、自身へのアクセスを許可できるようになりました。前のバージョンでは、System.security.allowDomain() を呼び出した場合に、アクセス先 SWF ファイルが System.security.allowDomain() を呼び出した SWF ファイルと同じドメインにある SWF ファイルであれば、クロススクリプト操作が許可されていました。前のバージョンでは、System.security.allowDomain() を呼び出すと、呼び出し元 SWF ファイルのドメイン全体が開きました。

- System.security.allowDomain("*") および System.security.allowInsecureDomain("*") ではワイルドカード値のサポートが追加されました。ワイルドカード (*) 値を使用すると、アクセス元ファイルがどこからロードされたかを問わず任意のファイルでクロススクリプト操作を許可できます。ワイルドカードは、グローバル許可と考えてください。ワイルドカード許可は一般的に有用ですが、Flash Player 8 の新しいローカルファイルセキュリティ規則の下で特定の操作を有効にする場合、特に必要となります。具体的には、ローカル SWF ファイルで、その SWF ファイルにインターネット上の SWF ファイルをスクリプトするネットワークアクセス許可がある場合、アクセス先のインターネット SWF ファイルは、ローカル SWF ファイルの出所が不明であることを反映して、System.security.allowDomain("*") を呼び出す必要があります。アクセス先のインターネット SWF ファイルが HTTPS URL からロードされる場合は、インターネット SWF ファイルでは、代わりに System.security.allowInsecureDomain("*") を呼び出す必要があります。

場合によっては、他のドメインから子 SWF ファイルをロードし、その子 SWF ファイルで親 SWF ファイルをスクリプトしたいものの、子 SWF ファイルの最終的なロード元となるドメインが判らないという状況になることがあります。このような状況は、たとえばロードバランシングリダイレクトやサードパーティ製サーバーを使用する場合に発生します。

この状況では、MovieClip._url プロパティをパラメータとしてこのメソッドに使用できます。たとえば、SWF ファイルをムービークリップ my_mc にロードした場合は、System.security.allowDomain(my_mc._url) を呼び出します。この操作をした場合は、my_mc の SWF ファイルのロードが開始されるまで待つようにしてください。これは、ロードが開始されるまで、_url プロパティには最終的な正しい値が収められないためです。子 SWF ファイルのロードが開始されたかどうか確認するには、MovieClipLoader.onLoadStart を使用するのが最もよいでしょう。

この反対の状況が発生する場合もあります。つまり、親 SWF ファイルで子 SWF ファイルをスクリプトしたいが、親 SWF ファイルのドメインがわからない場合です。この状況では、子 SWF ファイルから System.security.allowDomain(_parent._url) を呼び出してください。親 SWF ファイルがロードされるまで待つ必要はありません。親 SWF ファイルは子 SWF ファイルがロードされた時点で既にロードされています。

Flash Player 8 用にパブリッシュする場合、System.security.allowDomain("*") を呼び出すことによってこのような状況を処理することもできます。ただし、これは危険な近道になりかねません。任意のドメインの任意の他の SWF ファイルから呼び出し元 SWF ファイルにアクセスできるようになるためです。通常、_url プロパティを使用した方が安全です。

詳細については、次の参照先を参照してください。

- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

対応バージョン： ActionScript 1.0、Flash Player 6 - Flash Player 7 ではビヘイビアが変更されました。Flash Player 8 ではビヘイビアが変更されました。

パラメータ

`domain1:String` - `System.Security.allowDomain()` 呼び出しを含む SWF ファイル内のオブジェクトと変数にアクセスできるドメインを指定するストリング。ドメインは、次の形式で指定します。

- `"domain.com"`
- `"http://domain.com"`
- `"http://IPaddress"`
- (Flash Player 8 のみ) `"*"` ワイルドカード (`"*"`) を `System.security.allowDomain()` に渡すと、ローカルホストを含むすべてのドメインが呼び出し元 SWF ファイルにアクセスできるようになります。ワイルドカードを使用する前に、呼び出し元 SWF ファイルに対してこのような広範なアクセスを提供したいのかどうかを確認してください。このメソッドの主な説明の項を参照してください。

例

www.adobe.com/MovieA.swf の SWF ファイルに次の行が含まれています。

```
System.security.allowDomain("www.shockwave.com");
loadMovie("http://www.shockwave.com/MovieB.swf", my_mc);
```

MovieA には `allowDomain()` 呼び出しが含まれているので、MovieB は MovieA のオブジェクトと変数にアクセスすることができます。MovieA にこの呼び出しが含まれていない場合は、Flash Player のセキュリティ実装により、MovieB は MovieA のオブジェクトと変数にアクセスできません。

関連項目

`addCallback (ExternalInterface.addCallback メソッド)`, `onLoadComplete (MovieClipLoader.onLoadComplete イベントリスナー)`, `_parent (MovieClip._parent プロパティ)`, `_url (MovieClip._url プロパティ)`, `allowInsecureDomain (security.allowInsecureDomain メソッド)`

allowInsecureDomain (security.allowInsecureDomain メソッド)

```
public static allowInsecureDomain(domain:String) : Void
```

指定したドメイン内の SWF ファイルおよび HTML ファイルが、HTTPS プロトコルでホストされた呼び出し元 SWF ファイルのオブジェクトと変数にアクセスすることを許可します。このメソッドを使用することはお勧めできません。後の「セキュリティについての考慮事項」を参照してください。

このメソッドは、`System.security.allowDomain()` と同様に機能します。ただし、アクセス元が HTTPS 以外のプロトコルでロードされ、アクセス先が HTTPS でロードされる操作も許可します。Flash Player 7 以降では、HTTPS 以外のファイルで、HTTPS ファイルをスクリプトすることは許可されません。アクセス先 HTTPS SWF で `allowInsecureDomain()` メソッドを使用すると、この制限が解除されます。

HTTPS 以外のファイルから HTTPS ファイルへのスクリプトを有効にする場合にのみ、`allowInsecureDomain()` を使用してください。アクセス元の HTTPS 以外のファイルとアクセス先の HTTPS ファイルが、同じドメインに所属しているときのスクリプト (たとえば、http://mysite.com にある SWF ファイルで https://mysite.com にある SWF ファイルをスクリプトする場合) を有効にするために使用します。HTTPS 以外のファイル間でのスクリプト、HTTPS ファイル間でのスクリプト、または HTTPS ファイルから HTTPS 以外のファイルへのスクリプトを有効にするためには使用しないでください。このような状況では、代わりに `allowDomain()` を使用します。

セキュリティについての考慮事項 : Flash Player では柔軟性を最大化するために `allowInsecureDomain()` メソッドが用意されていますが、このメソッドを呼び出さないことをお勧めします。HTTPS を介してファイルを提供すると、開発者やユーザーに対して複数の保護機能を提供できますが、`allowInsecureDomain` の呼び出しによって、これらの保護機能は脆弱になります。次のシナリオでは、`allowInsecureDomain()` を十分な考慮なしで使用した場合にセキュリティが危険にさらされる状況を説明します。

メモ : 次の情報は、考えられるシナリオのうちの 1 つで、クロススクリプトの実際例を通じて `allowInsecureDomain()` を理解できるように設計されています。セキュリティアーキテクチャに関する問題をすべて取り上げているわけではなく、背景情報としてのみ使用する必要があります。Adobe デベロッパーセンターには、Flash Player およびセキュリティに関する広範な情報があります。詳細については、http://www.adobe.com/devnet/security/ を参照してください。

次の 2 つのコンポーネントから構成される e- コマースサイトを構築するシナリオを考えてみましょう。1 つはカタログで、公開情報しか含まれていないため、保護する必要はありません。もう 1 つは、ショッピングカート / チェックアウトコンポーネントで、ユーザーの財務情報と個人情報を保護するためにセキュリティで保護する必要があります。ここで、カタログのサービスは http://mysite.com/catalog.swf から、カートのサービスは https://mysite.com/cart.swf から提供するものとします。このサイトの 1 つの要件として、第三者がセキュリティアーキテクチャの脆弱性を利用して、ユーザーのクレジットカード番号を盗むことができないことが挙げられます。

ここで、中間当事者である攻撃者がサーバーとユーザーの間に介入して、ユーザーがショッピングカートアプリケーションに入力したクレジットカード番号を盗もうとするとします。中間当事者としては、一部のユーザーが利用している悪質な ISP、ユーザーの職場にいる悪意がある管理者など、パブリックインターネット経由で送信されるネットワークパケットをユーザーとサーバー間で表示または変更できるものが考えられます。この状況は珍しいことではありません。

cart.swf で HTTPS を使用してクレジットカード情報をサーバーに送信している場合、中間当事者の攻撃者は、HTTPS 送信が暗号化されているため、ネットワークパケットから直接この情報を盗むことができません。ただし、攻撃者は別の手法を使用できます。それは、いずれかの SWF ファイルの内容をユーザーへの配信時に変更し、その SWF ファイルを改変バージョン (ユーザー情報を攻撃者が所有する別のサーバーに送信する) と置き換える手法です。

HTTPS プロトコルは、何よりもまずこの " 変更 " 攻撃を防止します。暗号化されているだけでなく、HTTPS 送信に不正操作防止機能が付いているためです。中間当事者の攻撃者がパケットを変更すると、受信側はその変更を検出し、パケットを廃棄します。このため、この状況で攻撃者は cart.swf を変更できません。このファイルが HTTPS 経由で配信されるからです。

ところで、HTTP 経由で提供される catalog.swf 内のボタンで、HTTP 経由で提供される cart.swf 内のショッピングカートにアイテムを追加できるようにしたいとします。この機能を実現するために、cart.swf で `allowInsecureDomain()` を呼び出します。これで、catalog.swf は cart.swf をスクリプトできます。ただし、このアクションでは、予期しない結果が発生します。この場合、仮想の攻撃者は、catalog.swf をユーザーが最初にダウンロードする際、その内容を変更できます。catalog.swf は HTTP で配信され、不正操作防止機能が付いていないためです。攻撃者が変更した catalog.swf は、これで cart.swf をスクリプトできるようになります。cart.swf に `allowInsecureDomain()` への呼び出しが含まれているためです。変更された catalog.swf ファイルは、ActionScript を使用して、cart.swf 内の変数にアクセスできるため、ユーザーのクレジットカード情報やその他の機密データを読み取ることができます。その後、変更された catalog.swf は、このデータを攻撃者のサーバーに送信できます。

この実装は明らかに好ましいものではありませんが、サイト上の 2 つの SWF ファイル間でクロススクリプトを許可したい場合があります。次に、この仮想の e- コマースサイトを再設計して `allowInsecureDomain()` を除外する方法として、考えられる方法を 2 つ示します。

- アプリケーション内のすべての SWF ファイルを HTTPS 経由で提供します。これは最も単純で信頼性が高いソリューションです。説明したシナリオでは、catalog.swf と cart.swf の両方を HTTPS 経由で提供します。catalog.swf などのファイルを HTTP から HTTPS に切り替える際、帯域幅の使用量とサーバーの CPU 負荷が少し上がり、ユーザー側では、アプリケーションのロード時間がやや長くなる可能性があります。実際のサーバーで実験して、これらの影響の重大度を判別する必要があります。通常は、それぞれ 10 ～ 20% 程度で、まったく影響がないこともあります。サーバーにある HTTPS 加速ハードウェアやソフトウェアを使用すると、通常は結果を改善できます。関係するすべての SWF ファイルを HTTPS 経由で提供する主要なメリットは、ブラウザから内容が混在した警告を生成せずに、HTTPS URL をユーザーのブラウザ内の主要な URL として使用できることです。また、ブラウザのカギのアイコンが表示されるため、セキュリティに関する共通の信頼できるインジケータがユーザーに提供されます。
- HTTP から HTTPS へのスクリプトでなく、HTTPS から HTTP へのスクリプトを使用します。説明したシナリオでは、ユーザーのショッピングカートの内容を catalog.swf に保存し、cart.swf でチェックアウトプロセスのみ管理させることができます。チェックアウト時に、cart.swf で、catalog.swf 内の ActionScript 変数からカートの内容を取得できます。HTTP から HTTPS へのスクリプトに関する制限は非対称的なものです。HTTP 経由で提供される catalog.swf ファイルが HTTPS 経由で提供される cart.swf ファイルをスクリプトすることは安全上許可されませんが、HTTPS 経由で提供される cart.swf ファイルは HTTP 経由で提供される catalog.swf ファイルをスクリプトできます。このアプローチは、すべて HTTPS 経由のアプローチよりも微妙な点があります。不正操作を受けやすいため、HTTP 経由で提供される SWF ファイルは信頼しないように注意する必要があります。たとえば、cart.swf でカートの内容を記述した ActionScript 変数を取得する際、cart.swf 内の ActionScript コードで、この変数の値が期待した形式になっていると信頼することはできません。カートの内容を慎重に確認して、cart.swf に好ましくないアクションをとらせる無効なデータが含まれていないことを調べる必要があります。また、中間当事者が catalog.swf を変更して、たとえば、ユーザーのカートにアイテムを入れることにより、有効だが不正確なデータを cart.swf に提供する危険性を甘受する必要があります。通常のチェックアウトプロセスでは、カートの内容と合計金額を表示してユーザーに最後の承認を求めることで、この危険性をいくぶん緩和していますが、それでも危険性は残っています。

Web ブラウザでは、長年にわたり HTTPS ファイルと HTTPS 以外のファイルの分離を強制してきました。説明したシナリオでは、この制限が必要であることの 1 つの理由が明確に示されています。Flash Player では、絶対必要な場合、このセキュリティの制限を回避できますが、それを実行する前に結果を慎重に検討する必要があります。

詳細については、次の参照先を参照してください。

- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

対応バージョン： ActionScript 1.0、Flash Player 7

パラメータ

`domain:String` - www.myDomainName.com や store.myDomainName.com などの完全なドメイン名。Flash Player 8 では、ワイルドカード ("*") を `System.security.allowInsecureDomain()` に渡すと、ローカルホストを含むすべてのドメインが、呼び出し元 SWF ファイルにアクセスできます。ローカルホストを含むすべてのドメインに HTTPS SWF ファイルへのアクセスを許可する場合を除き、ワイルドカードは使用しないでください。

例

次の例では、登録済みの学生だけがアクセスできるよう、セキュアなドメイン上で数学のテストをホストしています。また、特定の概念を説明する複数の SWF ファイルを作成し、これをセキュアではないドメインでホストしているとします。ここで、学生が、概念情報を含む SWF ファイルからテストにアクセスできるようにしてみましょう。

```
// This SWF file is at https://myEducationSite.somewhere.com/mathTest.swf
// Concept files are at http://myEducationSite.somewhere.com
System.security.allowInsecureDomain("myEducationSite.somewhere.com");
```

関連項目

allowDomain (security.allowDomain メソッド), exactSettings (System.exactSettings プロパティ)

loadPolicyFile (security.loadPolicyFile メソッド)

```
public static loadPolicyFile(url:String) : Void
```

`url` パラメータで指定された場所からドメイン間ポリシーファイルをロードします。Flash Player では、SWF ファイルが置かれているサーバー以外のサーバーからデータをロードすることを許可する許可メカニズムとして、ポリシーファイルが使用されます。

Flash Player 7.0.14.0 では、ポリシーファイルの検索が行われるのは、データロード要求が行われたサーバーの /crossdomain.xml に限られます。XMLSocket 接続試行の場合、XMLSocket 接続の試行が行われたサブドメインのポート 80 上にある HTTP サーバーで、/crossdomain.xml に検索が行われます。さらに Flash Player 7.0.14.0 (およびそれ以前のバージョン) では、XMLSocket 接続がポート 1024 以上に限られています。

`System.security.loadPolicyFile()` を追加することで、Flash Player 7.0.19.0 では任意の場所からポリシーファイルをロードできるようになります。次に例を示します。

```
System.security.loadPolicyFile("http://foo.com/sub/dir/pf.xml");
```

これにより、Flash Player は指定された URL からポリシーファイルを取得できるようになります。この場所に置かれているポリシーファイルによって得られる許可は、サーバーの仮想ディレクトリ階層で同レベル以下のコンテンツすべてに適用されます。次に示すコードは、前の例の続きです。

```
loadVariables("http://foo.com/sub/dir/vars.txt") // allowed
loadVariables("http://foo.com/sub/dir/deep/vars2.txt") // allowed
loadVariables("http://foo.com/elsewhere/vars3.txt") // not allowed
```

`loadPolicyFile()` を使用して、任意の数のポリシーファイルをロードできます。ポリシーファイルを必要とする要求がある場合、Flash Player はポリシーファイルのダウンロードがすべて完了するまで必ず待機します。その間に要求が拒否されることはありません。`loadPolicyFile()` で指定されたポリシーファイルによって要求が許可されなかった場合は、最終的にデフォルトの場所である /crossdomain.xml が参照されます。

特定のポート番号で `xmlsocket` プロトコルを使用することで、直接 XMLSocket サーバーからポリシーファイルを取得することができます。次に例を示します。

```
System.security.loadPolicyFile("xmlsocket://foo.com:414");
```

このコードを使用すると、Flash Player は指定されたホストとポートからポリシーファイルを取得しようとします。1024 以上のポートだけでなく、任意のポートを使用できます。指定されたポートを使用して接続が確立されると、Flash Player は `null` バイトで終了する `<policy-file-request />` を送信します。XMLSocket サーバーを設定することで、ポリシーファイルと通常の XMLSocket 接続に同じポートを使用することができます。この場合、サーバーは `<policy-file-request />` が受信されるまで待機し、その後でポリシーファイルを送信する必要があります。標準の接続とは異なるポートを使用して、ポリシーファイルを提供するようにサーバーを設定することもできます。この場合は、ポリシーファイル専用のポートで接続が確立されるとすぐにポリシーファイルを送信できます。ポリシーファイルを終了するためにサーバーから `null` バイトを送信し、それ以降の接続を閉じる必要があります。サーバー側で接続を閉じなければ、最後の null バイトが受信されるとすぐに Flash Player 側で接続が閉じられます。

XMLSocket サーバーで提供するポリシーファイルのシンタックスは、他のポリシーファイルとほぼ同じですが、アクセスを許可するポートも指定する必要がある点が異なります。ポリシーファイルが 1024 未満のポートから提供される場合、任意のポートへのアクセスが許可されます。ポリシーファイルが 1024 以上のポートから提供される場合、他の 1024 以上のポートにのみアクセスを許可できます。許可するポートは、`<allow-access-from>` タグの `"to-ports"` 属性で指定します。単一のポート番号、ポート範囲、ワイルドカードを使用できます。次に、XMLSocket ポリシーファイルの例を示します。

```
<cross-domain-policy>
<allow-access-from domain="*" to-ports="507" />
<allow-access-from domain="*.foo.com" to-ports="507,516" />
<allow-access-from domain="*.bar.com" to-ports="516-523" />
<allow-access-from domain="www.foo.com" to-ports="507,516-523" />
<allow-access-from domain="www.bar.com" to-ports="*" />
</cross-domain-policy>
```

元のデフォルトの場所、つまり HTTP サーバーのポート 80 の /crossdomain.xml から取得されたポリシーファイルでは、1024 以上の全ポートへのアクセスが暗黙的に許可されています。HTTP サーバーの他の場所から、XMLSocket 操作を許可するポリシーファイルを取得することはできません。XMLSocket ポリシーファイルを独自の場所に置く場合、XMLSocket サーバー上に置く必要があります。

1024 未満のポートに接続する機能は新しいため、ムービークリップが自分自身のサブドメインに接続する場合でも、`loadPolicyFile()` によってロードするポリシーファイルを使用して常にこの接続の許可を行う必要があります。

詳細については、次の参照先を参照してください。

- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

対応バージョン： ActionScript 1.0、Flash Player 7,0,19,0

パラメータ

`url:String` - ロードするドメイン間ポリシーファイルが置かれている URL を表すストリング。

sandboxType (security.sandboxType プロパティ)

`public static sandboxType : String` (読み取り専用)

呼び出し元の SWF ファイルが動作しているセキュリティサンドボックスのタイプを示します。

`System.security.sandboxType` は、次のいずれかの値になります。

- `remote` : この SWF ファイルはインターネット URL からのものであり、ドメインベースのサンドボックス規則に従って機能します。
- `localWithFile` : この SWF ファイルはローカルファイルであり、信頼性はなく、ネットワークを指定してパブリッシュされているわけではありません。ローカルのデータソースから読み取ることができますが、インターネットでのやり取りはできません。
- `localWithNetwork` : この SWF ファイルはローカルファイルであり、信頼性はありませんが、ネットワークを指定してパブリッシュされています。インターネットでやり取りできますが、ローカルのデータソースから読み取ることはできません。

- `localTrusted`: この SWF ファイルはローカルファイルであり、設定マネージャまたは "FlashPlayerTrust" 構成ファイルを使用して信頼性があります。ローカルのデータソースから読み取ることも、インターネットでやり取りすることもできます。

このプロパティは、任意のバージョンの SWF ファイルから確認できますが、サポートされるのは Flash Player 8 以降のみです。つまり、たとえば Flash Player 8 で再生しているバージョン 7 の SWF ファイルから、このプロパティを確認することができます。このサポートでは、8 より古いバージョンについてパブリッシュする場合、このプロパティが再生時にサポートされるかどうかがパブリッシュ時にはわかりません。したがって、バージョン 7 以前の SWF ファイルでは、このプロパティが未定義の値であることがわかる場合があります。これは、Flash Player のバージョン (`System.capabilities.version` で指定) が 8 よりも古いときにのみ生じる状況ですが、この場合は SWF ファイルの URL がローカルファイルであるかどうかによってサンドボックスのタイプを判別できます。SWF ファイルの URL がローカルファイルである場合は、SWF ファイルが `"localTrusted"` として分類されると想定できます。Flash Player 8 より古いバージョンでは、これがすべてのローカルコンテンツの処理方法でした。SWF ファイルの URL がローカルファイルでない場合は、SWF ファイルが `"remote".` として分類されると想定できます。

詳細については、次の参照先を参照してください。

- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

対応バージョン: ActionScript 1.0、Flash Player 8 - バージョン固有の詳細説明を参照してください。

選択

```
Object
  |
  +-Selection
```

```
public class Selection
extends Object
```

Selection クラスを使用すると、ムービー内でカーソルを置くテキストフィールド、つまりフォーカスが置かれているフィールドを設定および制御することができます。選択範囲のインデックスはゼロから始まります。つまり、第 1 の位置は 0、第 2 の位置は 1、以下同様です。

フォーカスのあるフィールドは一度に 1 つしか存在しないため、Selection クラスにはコンストラクタ関数がありません。

対応バージョン: ActionScript 1.0、Flash Player 5

プロパティー覧

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベントの一覧

イベント	説明
onSetFocus = function([oldfocus:Object], [newfocus:Object]) {}	フォーカスが変更されると、通知されます。

メソッド一覧

オプション	署名	説明
static	addListener(listener:Object) : Void	フォーカスの変更通知を受け取るオブジェクトを登録します。
static	getBeginIndex() : Number	選択範囲の先頭を示すインデックスを返します。
static	getCaretIndex() : Number	点滅するカーソル (キャレット) 位置のインデックスを返します。
static	getEndIndex() : Number	フォーカスされている選択範囲の終わりのインデックスを返します。
static	getFocus() : String	フォーカスのあるオブジェクトのターゲットパスを指定するストリングを返します。
static	removeListener(listener:Object) : Boolean	Selection.addListener() を使用して以前に登録したオブジェクトを削除します。
static	setFocus(newFocus:Object) : Boolean	newFocus パラメータで指定する選択可能 (編集可能) なテキストフィールド、ボタン、またはムービークリップにフォーカスを設定します。
static	setSelection(beginIndex:Number, endIndex:Number) : Void	現在フォーカスのあるテキストフィールドの選択範囲を設定します。

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

addListener (Selection.addListener メソッド)

public static addListener(listener:Object) : Void

フォーカスの変更通知を受け取るオブジェクトを登録します。フォーカスが変わると (Selection.setFocus() が呼び出された場合など)、addListener() で登録されたすべてのリスナーオブジェクトの onSetFocus メソッドが呼び出されます。複数のオブジェクトがフォーカスの変更通知をリッスンできます。指定したリスナーが登録済みである場合、変化は起きません。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

listener:Object - onSetFocus メソッドを持つ新しいオブジェクト。

例

次の例では、実行時に 2 つのテキスト入力フィールドを作成し、各テキストフィールドの境界線を true に設定します。このコードでは、focusListener という名前の ActionScript オブジェクト (汎用オブジェクト) を新しく作成しています。そして、このオブジェクト自体の onSetFocus プロパティを定義し、関数を割り当てています。この関数は 2 つのパラメータを取ります。フォーカスを失ったテキストフィールドへの参照と、フォーカスを受け取ったテキストフィールドへの参照です。関数では、フォーカスを失ったテキストフィールドの border プロパティを false に、フォーカスを受け取ったテキストフィールドの border プロパティを true に、それぞれ設定しています。

```
this.createTextField("one_txt", 99, 10, 10, 200, 20);
this.createTextField("two_txt", 100, 10, 50, 200, 20);
one_txt.border = true;
one_txt.type = "input";
two_txt.border = true;
two_txt.type = "input";

var focusListener:Object = new Object();
focusListener.onSetFocus = function(oldFocus_txt, newFocus_txt) {
  oldFocus_txt.border = false;
  newFocus_txt.border = true;
};
Selection.addListener(focusListener);
```

SWF ファイルをテストするときに、2 つのテキストフィールドの間をタブで移動してみてください。Tab キーを使用して 2 つのフィールド間でフォーカスを移動できるように、[制御]-[キーボードショートカットを無効] を選択します。

関連項目

setFocus (Selection.setFocus メソッド)

getBeginIndex (Selection.getBeginIndex メソッド)

```
public static getBeginIndex() : Number
```

選択範囲の先頭を示すインデックスを返します。インデックスがないか、フォーカスを持つテキストフィールドがない場合は -1 を返します。選択範囲インデックスはゼロから始まります。つまり、第 1 の位置は 0、第 2 の位置 1、以下同様です。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Number - 整数。

例

次の例では、実行時にテキストフィールドを作成し、プロパティを設定します。現在選択されているテキストを大文字に変更するためのコンテキストメニューアイテムが追加されます。

```
this.createTextField("output_txt", this.getNextHighestDepth(), 0, 0, 300, 200);
output_txt.multiline = true;
output_txt.wordWrap = true;
output_txt.border = true;
output_txt.type = "input";
output_txt.text = "Enter your text here";
var my_cm:ContextMenu = new ContextMenu();
my_cm.customItems.push(new ContextMenuItem("Uppercase...", doUppercase));
function doUppercase():Void {
  var startIndex:Number = Selection.getBeginIndex();
  var endIndex:Number = Selection.getEndIndex();
  var stringToUppercase:String = output_txt.text.substring(startIndex,
  endIndex);
  output_txt.replaceText(startIndex, endIndex,
  stringToUppercase.toUpperCase());
}
output_txt.menu = my_cm;
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

別の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Strings" フォルダに移動して Strings.fla ファイルにアクセスします。

関連項目

getEndIndex (Selection.getEndIndex メソッド)

getCaretIndex (Selection.getCaretIndex メソッド)

```
public static getCaretIndex() : Number
```

点滅するカーソル (キャレット) 位置のインデックスを返します。点滅するカーソルが表示されていない場合は -1 を返します。選択範囲インデックスはゼロから始まります。つまり、第 1 の位置は 0 であり、第 2 の位置は 1 です。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Number - 整数。

例

次の例では、実行時にテキストフィールドのプロパティを作成し、設定しています。getCaretIndex() メソッドは、キャレットのインデックスを返し、その値を別のテキストフィールドに表示するために使用されます。

```
this.createTextField("pos_txt", this.getNextHighestDepth(), 50, 20, 100, 22);
this.createTextField("content_txt", this.getNextHighestDepth(), 50, 50, 400,
  300);
content_txt.border = true;
content_txt.type = "input";
content_txt.wordWrap = true;
content_txt.multiline = true;
content_txt.onChanged = getCaretPos;

var keyListener:Object = new Object();
keyListener.onKeyUp = getCaretPos;
Key.addListener(keyListener);

var mouseListener:Object = new Object();
mouseListener.onMouseUp = getCaretPos;
Mouse.addListener(mouseListener);

function getCaretPos() {
  pos_txt.text = Selection.getCaretIndex();
}
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

別の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Strings" フォルダに移動して Strings.fla ファイルにアクセスします。

getEndIndex (Selection.getEndIndex メソッド)

```
public static getEndIndex() : Number
```

フォーカスされている選択範囲の終わりのインデックスを返します。インデックスがないか、現在フォーカスされている選択範囲がない場合は -1 を返します。選択範囲インデックスはゼロから始まります。つまり、第 1 の位置は 0、第 2 の位置は 1 です。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Number - 整数。

例

この例は、ActionScript サンプルのフォルダの "Strings.fla" ファイルから実行されます。

```
// define the function which converts the selected text in an instance,
// and convert the string to upper or lower case.
function convertCase(target, menuItem) {
  var beginIndex:Number = Selection.getBeginIndex();
  var endIndex:Number = Selection.getEndIndex();
  var tempString:String;
  // make sure that text is actually selected.
  if (beginIndex>-1 && endIndex>-1) {
  // set the temporary string to the text before the selected text.
  tempString = target.text.slice(0, beginIndex);
  switch (menuItem.caption) {
  case 'Uppercase...' :
    // if the user selects the "Uppercase..." context menu item,
    // convert the selected text to upper case.
    tempString += target.text.substring(beginIndex, endIndex).toUpperCase();
    break;
  case 'Lowercase...' :
    tempString += target.text.substring(beginIndex, endIndex).toLowerCase();
    break;
  }
  // append the text after the selected text to the temporary string.
  tempString += target.text.slice(endIndex);
```

```
  // set the text in the target text field to the contents of the temporary
  string.
  target.text = tempString;
  }
}
```

スクリプト全体については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Strings" フォルダに移動して Strings.fla ファイルにアクセスします。

関連項目

getBeginIndex (Selection.getBeginIndex メソッド)

getFocus (Selection.getFocus メソッド)

```
public static getFocus() : String
```

フォーカスのあるオブジェクトのターゲットパスを指定するストリングを返します。

- TextField オブジェクトにフォーカスがあり、インスタンス名が付いている場合、このメソッドは TextField オブジェクトのターゲットパスを返します。それ以外は、TextField の変数名を返します。
- Button オブジェクトまたはボタンムービークリップにフォーカスがある場合、このメソッドはそれらのターゲットパスを返します。
- TextField オブジェクト、Button オブジェクト、コンポーネントインスタンス、ボタンムービークリップのいずれにもフォーカスがない場合は、`null` が返されます。

対応バージョン : ActionScript 1.0、Flash Player 5 - ボタンとテキストフィールドのインスタンス名は、Flash Player 6 以降で使用できます。

戻り値

`String` - ストリングまたは `null`。

例

次の例では、現在フォーカスがある選択内容のターゲットパスを TextArea コンポーネントインスタンスに表示します。いくつかのコンポーネントインスタンスやボタン、テキストフィールド、ムービークリップインスタンスをステージに追加します。いくつかのコンポーネントインスタンスやボタン、テキストフィールド、ムービークリップインスタンスを SWF ファイルに追加します。次の ActionScript を AS ファイルまたは FLA ファイルに追加します。

```
var focus_ta:mx.controls.TextArea;
my_mc.onRelease = function() {};
my_btn.onRelease = function() {};

var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.SPACE)) {
  focus_ta.text = Selection.getFocus()+newline+focus_ta.text;
  }
};
Key.addListener(keyListener);
```

SWF ファイルをテストし、Tab キーを使用してステージ上のインスタンス間を移動します。テスト環境では、[制御]-[キーボードショートカットを無効] を選択してください。

関連項目

onSetFocus (Selection.onSetFocus イベントリスナー), setFocus (Selection.setFocus メソッド)

onSetFocus (Selection.onSetFocus イベントリスナー)

```
onSetFocus = function([oldfocus:Object], [newfocus:Object]) {}
```

フォーカスが変更されると、通知されます。このリスナーを使用するには、リスナーオブジェクトを作成します。次にこのリスナーの関数を定義し、Selection.addListener() を使用してリスナーを Selection オブジェクトに登録します。次に例を示します。

```
var someListener:Object = new Object();
someListener.onSetFocus = function () {
  // statements
}
Selection.addListener(someListener);
```

1 つのイベントに関する通知を複数のリスナーで受け取ることができるので、リスナーごとに異なる複数のコードを作成して連携させることもできます。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

`oldfocus:Object` (オプション) - フォーカスを失うオブジェクト。

`newfocus:Object` (オプション) - フォーカスを受け取るオブジェクト。

例

次の例では、ダイナミックに作成された複数のテキストフィールドの間で、SWF ファイルの入力フォーカスの移動のタイミングを判断する方法について説明します。次の ActionScript を FLA または AS ファイルに入力し、ドキュメントをテストします。

```
this.createTextField("one_txt", 1, 0, 0, 100, 22);
this.createTextField("two_txt", 2, 0, 25, 100, 22);
this.createTextField("three_txt", 3, 0, 50, 100, 22);
this.createTextField("four_txt", 4, 0, 75, 100, 22);

for (var i in this) {
  if (this[i] instanceof TextField) {
  this[i].border = true;
  this[i].type = "input";
  }
}

this.createTextField("status_txt", this.getNextHighestDepth(), 200, 10, 300,
  100);
status_txt.html = true;
status_txt.multiline = true;

var someListener:Object = new Object();
someListener.onSetFocus = function(oldFocus, newFocus) {
  status_txt.htmlText = "<b>setFocus triggered</b>";
  status_txt.htmlText += "<textformat tabStops='[20,80]'>";
  status_txt.htmlText += " \toldFocus:\t"+oldFocus;
  status_txt.htmlText += " \tnewFocus:\t"+newFocus;
  status_txt.htmlText += " \tgetFocus:\t"+Selection.getFocus();
  status_txt.htmlText += "</textformat>";
};
Selection.addListener(someListener);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

addListener (Selection.addListener メソッド), setFocus (Selection.setFocus メソッド)

removeListener (Selection.removeListener メソッド)

`public static removeListener(listener:Object) : Boolean`

`Selection.addListener()` を使用して以前に登録したオブジェクトを削除します。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

`listener:Object` - フォーカスの通知受信を中止するオブジェクト。

戻り値

`Boolean` - `listener` が正常に削除された場合は `true` を返します。`listener` が正常に削除されなかった場合 (`listener` が Selection オブジェクトのリスナーリストにない場合など) は `false` を返します。

例

次の ActionScript は、いくつかのテキストフィールドインスタンスをダイナミックに作成します。テキストフィールドを選択すると、[出力] パネルに情報が表示されます。`remove_btn` インスタンスをクリックすると、リスナーが削除され、[出力] パネルに情報が表示されなくなります。

```
this.createTextField("one_txt", 1, 0, 0, 100, 22);
this.createTextField("two_txt", 2, 0, 25, 100, 22);
this.createTextField("three_txt", 3, 0, 50, 100, 22);
this.createTextField("four_txt", 4, 0, 75, 100, 22);

for (var i in this) {
  if (this[i] instanceof TextField) {
  this[i].border = true;
  this[i].type = "input";
  }
}

var selectionListener:Object = new Object();
selectionListener.onSetFocus = function(oldFocus, newFocus) {
  trace("Focus shifted from "+oldFocus+" to "+newFocus);
};
Selection.addListener(selectionListener);

remove_btn.onRelease = function() {
  trace("removeListener invoked");
  Selection.removeListener(selectionListener);
};
```

関連項目

`addListener (Selection.addListener メソッド )`

setFocus (Selection.setFocus メソッド)

```
public static setFocus(newFocus:Object) : Boolean
```

newFocus パラメータで指定する選択可能 (編集可能) なテキストフィールド、ボタン、またはムービークリップにフォーカスを設定します。null または undefined を渡すと、現在のフォーカスは削除されます。

対応バージョン : ActionScript 1.0、Flash Player 5 - ボタンとムービークリップのインスタンス名は、Flash Player 6 以降でのみ使用できます。

パラメータ

newFocus:Object - ボタン、ムービークリップ、テキストフィールドの各インスタンスなどのオブジェクト、またはボタン、ムービークリップ、テキストフィールドの各インスタンスへのパスを指定するストリング。パスを指定するストリングリテラルを渡す場合は、そのパスを引用符 (" ") で囲む必要があります。ドットまたはスラッシュ表記を使用してパスを指定できます。ActionScript 2.0 の場合は、ドット表記を使用する必要があります。相対パスと絶対パスのいずれでも使用できます。

戻り値

Boolean - ブール値。フォーカスの試行が成功した場合は true、失敗した場合は false を返します。

例

次の例では、ブラウザウィンドウで実行中に、テキストフィールドは username_txt テキストフィールドにフォーカスを設定します。必要なテキストフィールド (username_txt と password_txt) のいずれかが未入力である場合、カーソルはデータが未入力のテキストフィールドに自動的にフォーカスを与えます。たとえば、ユーザーが username_txt テキストフィールドに何も入力せずに送信ボタンを押すと、エラーメッセージが表示され、カーソルは username_txt テキストフィールドにフォーカスを与えます。

```
this.createTextField("status_txt", this.getNextHighestDepth(), 100, 70, 100,
  22);
this.createTextField("username_txt", this.getNextHighestDepth(), 100, 100, 100,
  22);
this.createTextField("password_txt", this.getNextHighestDepth(), 100, 130, 100,
  22);
this.createEmptyMovieClip("submit_mc", this.getNextHighestDepth());
submit_mc.createTextField("submit_txt", this.getNextHighestDepth(), 100, 160,
  100, 22);
submit_mc.submit_txt.autoSize = "center";
submit_mc.submit_txt.text = "Submit";
submit_mc.submit_txt.border = true;
submit_mc.onRelease = checkForm;
username_txt.border = true;
password_txt.border = true;
username_txt.type = "input";
```

```
password_txt.type = "input";
password_txt.password = true;
Selection.setFocus("username_txt");
//
function checkForm():Boolean {
  if (username_txt.text.length == 0) {
  status_txt.text = "fill in username";
  Selection.setFocus("username_txt");
  return false;
  }
  if (password_txt.text.length == 0) {
  status_txt.text = "fill in password";
  Selection.setFocus("password_txt");
  return false;
  }
  status_txt.text = "success!";
  Selection.setFocus(null);
  return true;
}
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

getFocus (Selection.getFocus メソッド)

setSelection (Selection.setSelection メソッド)

```
public static setSelection(beginIndex:Number, endIndex:Number) : Void
```

現在フォーカスのあるテキストフィールドの選択範囲を設定します。新しい選択範囲は、beginIndex パラメータで指定されたインデックスから始まり、endIndex パラメータで指定されたインデックスで終わります。選択範囲のインデックスはゼロから始まります。つまり、第 1 の位置は 0、第 2 の位置は 1、以下同様です。現在フォーカスされているテキストフィールドがない場合は、このメソッドは何も実行しません。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

beginIndex:Number - 選択範囲の始めのインデックス。

endIndex:Number - 選択範囲の終わりのインデックス。

例

次の ActionScript では、実行時にテキストフィールドを作成し、ストリングを追加します。テキストフィールドにフォーカスを与え、フォーカスのあるテキストフィールドの文字数のスパンを選択します。

```
this.createTextField("myText_txt", 99, 10, 10, 200, 30);
myText_txt.text = "this is my text";
this.onEnterFrame = function () {
  Selection.setFocus("myText_txt");
  Selection.setSelection(0, 3);
  delete this.onEnterFrame;
}
```

次の例では、endIndex パラメータを指定しない方法を示します。最初の文字を選択するには、値が 0 でなく 1 の endIndex を使用する必要があります。endIndex パラメータを 0 に設定すると、何も選択されません。

```
this.createTextField("myText_txt", 99, 10, 10, 200, 30);
myText_txt.text = "this is my text";
this.onEnterFrame = function () {
  Selection.setFocus("myText_txt");
  Selection.setSelection(0, 1);
  delete this.onEnterFrame;
}
```

SharedObject

```
Object
  |
  +-SharedObject

public dynamic class SharedObject
extends Object
```

SharedObject クラスは、ユーザーのコンピュータ上で限定された量のデータを読み込みおよび格納するために使用されます。共有オブジェクトを使用すると、ユーザーのコンピュータ上に永続化されているオブジェクト間で、リアルタイムでデータを共有することができます。ローカル共有オブジェクトは、ブラウザのクッキーに似ています。

共有オブジェクトの 3 つの考えられる使用方法を次に示します。

- ユーザーの高得点を保存するゲーム。このゲームでは、サーバー上の専用記憶域なしで、ユーザー名や高得点など、ユーザーのパーソナライズされたデータを提供できます。
- オンライン、オフラインのいずれでも機能する電話帳アプリケーション。プロジェクタアプリケーションとして提供される電話帳には、ユーザーが入力した名前と電話番号のリストが付いたローカルデータキャッシュを格納できます。インターネット接続が使用可能な場合、このアプリケーションでは、サーバーから最新情報を取得します。接続を使用できない場合は、共有オブジェクトに保存されている最新のデータが使用されます。

- 新しいサイト上でユーザーが閲覧した記事のレコードなど、複雑な Web サイトのユーザー環境設定またはトラッキングデータ。この情報をトラッキングすると、新規で未読の記事でなく、既に閲覧された記事を表示できます。この情報をユーザーのコンピュータに保存すると、サーバーの負荷を削減できます。

ローカル共有オブジェクトは、ローカルの永続性を管理します。たとえば、SharedObject.getLocal() を呼び出して、ゲームの高得点を格納する共有オブジェクトを作成できます。共有オブジェクトはローカルに永続化されるので、そのデータ属性はゲームを閉じたときにユーザーのコンピュータに保存されます。次回セッションを開くと、前のセッションの高得点が表示されます。代わりに、ゲームを閉じる前に、共有オブジェクトのプロパティを null に設定することもできます。次回 SWF ファイルを実行すると、前の高得点なしでゲームが開きます。

ローカル共有オブジェクトを作成するには、次のシンタックスを使用します。

```
var so:SharedObject = SharedObject.getLocal("userHighScore");
so.data.highScore = new Number();
so.flush();
```

この例では、共有オブジェクトがディスクに明示的に保存または書き込まれています。アプリケーションが閉じると、共有オブジェクトは自動的に保存されますが、データをディスクに書き込む手順を示すため、ここに表示しています。

ローカルディスク領域についての考慮事項：ローカル共有オブジェクトは非常に役に立ちますが、アプリケーションの設計時に考慮する必要があるいくつかの制限があります。SWF ファイルでローカル共有オブジェクトの書き込みが許可されない場合があります。ローカル共有オブジェクトに格納されているデータが、わからないうちに削除される場合もあります。Flash Player のユーザーは、個々のドメインまたはすべてのドメインで使用できるディスク領域を管理できます。ユーザーが使用可能なディスク領域の量を減らすと、一部のローカル共有オブジェクトが削除される可能性があります。Flash Player のユーザーには、サードパーティドメイン (現在のブラウザのアドレスバーにあるドメイン以外のドメイン) によるローカル共有オブジェクトの読み取りまたは書き込みを防止できるプライバシーコントロール機能もあります。

メモ：サードパーティによるディスクへの共有オブジェクトの書き込みが許可されない場合でも、ローカルコンテンツではサードパーティの共有オブジェクトをディスクに書き込むことができます。

使用可能なディスク領域の量とユーザーのプライバシーコントロールに関する失敗についてチェックすることをお勧めします。getLocal() および flush() の呼び出し時にこれらのチェックを実行します。

SharedObject.getLocal() – ユーザーがサードパーティの共有オブジェクトを無効にし、SWF ファイルのドメインがブラウザのアドレスバーのドメインと一致しない場合、このメソッドは null を返します。

SharedObject.flush() – ユーザーがあなたのドメインまたはすべてのドメインの共有オブジェクトを無効にした場合、このメソッドは false を返します。追加の記憶域が必要な場合、"pending" を返します。ユーザーは、インタラクティブに増加を許可するかどうかを決定する必要があります。

SWF ファイルでローカル共有オブジェクトを作成または変更しようとする場合、SWF ファイルの幅が最低でも 215 ピクセル、高さが最低でも 138 ピクセルあることを確認してください。このサイズは、ダイアログボックス (ローカル共有オブジェクトの記憶域制限を増やすかどうかをユーザーに確認する) を表示するための最小の大きさです。SWF ファイルがこの大きさよりも小さく、記憶域制限を増やす必要がある場合、SharedObject.flush() は失敗し、"pending" を返します。ただし、その後、"SharedObject.Flush.Failed" の結果を使用して SharedObject.onStatus ハンドラを呼び出します。

対応バージョン : ActionScript 1.0、Flash Player 6

関連項目

getLocal (SharedObject.getLocal メソッド), flush (SharedObject.flush メソッド), onStatus (SharedObject.onStatus ハンドラ)

プロパティ一覧

オプション	プロパティ	説明
	data:Object	オブジェクトの data プロパティに割り当てられた属性のコレクション。これらの属性は共有および保存することができます。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベントの一覧

イベント	説明
onStatus = function(infoObject: Object) {}	共有オブジェクトに対してエラー、警告、情報通知が送信されたときに呼び出されます。

メソッド一覧

オプション	署名	説明
	clear() : Void	共有オブジェクトのすべてのデータを消去し、ディスクから共有オブジェクトを削除します。
	flush([minDiskSpace:Number]) : Object	ローカル永続共有オブジェクトをすぐにローカルファイルに書き込みます。
static	getLocal(name:String, [localPath:String], [secure:Boolean]) : SharedObject	現在のクライアントだけが利用できるローカル永続共有オブジェクトへの参照を返します。
	getSize() : Number	共有オブジェクトの現在のサイズ (バイト数) を取得します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

clear (SharedObject.clear メソッド)

public clear() : Void

共有オブジェクトのすべてのデータを消去し、ディスクから共有オブジェクトを削除します。my_so への参照はアクティブなままで、my_so は空になります。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例では、共有オブジェクトにデータを設定してから、その共有オブジェクトのすべてのデータを空にします。

```
var my_so:SharedObject = SharedObject.getLocal("superfoo");
my_so.data.name = "Hector";
trace("before my_so.clear():");
for (var prop in my_so.data) {
  trace("\t"+prop);
}
```

```
trace("");
my_so.clear();
trace("after my_so.clear():");
for (var prop in my_so.data) {
  trace("\t"+prop);
}
```

この ActionScript は、[出力] パネルに次のメッセージを表示します。

```
before my_so.clear():
  name

after my_so.clear():
```

data (SharedObject.data プロパティ)

```
public data : Object
```

オブジェクトの data プロパティに割り当てられた属性のコレクション。これらの属性は共有および保存することができます。それぞれの属性は、ActionScript または JavaScript の基本タイプのオブジェクトです。たとえば、Array、Number、Boolean などです。次のコードでは、共有オブジェクトにさまざまな値を割り当てています。

```
var items_array:Array = new Array(101, 346, 483);
var currentUserIsAdmin:Boolean = true;
var currentUserName:String = "Ramona";

var my_so:SharedObject = SharedObject.getLocal("superfoo");
my_so.data.itemNumbers = items_array;
my_so.data.adminPrivileges = currentUserIsAdmin;
my_so.data.userName = currentUserName;

for (var prop in my_so.data) {
  trace(prop+": "+my_so.data[prop]);
}
```

永続的なオブジェクトの場合は、共有オブジェクトの data プロパティのすべての属性が保存されます。また共有オブジェクトには次の情報が含まれています。

```
userName: Ramona
adminPrivileges: true
itemNumbers: 101,346,483
```

メモ : so.data = someValue のように、共有オブジェクトの data プロパティに値を直接割り当てるのはやめてください。このような割り当ては無視されます。

ローカル共有オブジェクトの属性を削除するには、delete so.data.attributeName というコードを使用します。ローカル共有オブジェクトの属性を null または undefined に設定しても属性は削除されません。

共有オブジェクトのプライベート値 (オブジェクトの使用中にそのクライアントインスタンスでのみ利用でき、閉じるときにオブジェクトと共に保存されない値) を作成するには、data 以外の名前のプロパティを作成して、その値を保存します。次に例を示します。

```
var my_so:SharedObject = SharedObject.getLocal("superfoo");
my_so.favoriteColor = "blue";
my_so.favoriteNightClub = "The Bluenote Tavern";
my_so.favoriteSong = "My World is Blue";

for (var prop in my_so) {
  trace(prop+": "+my_so[prop]);
}
```

共有オブジェクトには次のデータが含まれます。

```
favoriteSong: My World is Blue
favoriteNightClub: The Bluenote Tavern
favoriteColor: blue
data: [object Object]
```

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の例では、TextInput コンポーネントインスタンスから共有オブジェクト my_so にテキストを保存します (完全な例については、SharedObject.getLocal() を参照)。

```
// Create a listener object and function for the <enter> event.
var textListener:Object = new Object();
textListener.enter = function(eventObj:Object) {
  my_so.data.myTextSaved = eventObj.target.text;
  my_so.flush();
};
```

関連項目

getLocal (SharedObject.getLocal メソッド)

flush (SharedObject.flush メソッド)

```
public flush([minDiskSpace:Number]) : Object
```

ローカル永続共有オブジェクトをすぐにローカルファイルに書き込みます。このメソッドを使用しない場合、共有オブジェクトがファイルに書き込まれるのは、その共有オブジェクトのセッションが終了した時点となります。つまり、その SWF ファイルを閉じた時点、その共有オブジェクトに対する参照がなくなってガベージコレクションされた時点、または SharedObject.clear( ) が呼び出された時点のいずれかです。

このメソッドが "pending" を返した場合は、Flash Player では、このドメインからのオブジェクトを保存するためのディスク領域を増やすように求めるダイアログボックスが表示されます。将来的に共有オブジェクトを保存する際に、領域が自動的に拡張されるようにする ("pending" が返されないようにする) には、minimumDiskSpace に値を指定します。ファイルにオブジェクトを書き込む際には、現在のサイズの共有オブジェクトを保存するために十分な領域だけではなく、minimumDiskSpace に指定したバイト数が確認されます。

たとえば、共有オブジェクトが最初は小さくても後から最大で 500 バイトまで増えると予想される場合には、minimumDiskSpace に 500 を指定します。ユーザーに対して共有オブジェクトへのディスク領域の割り当てを求める際には、500 バイトの領域が要求されます。要求されたディスク領域をユーザーが割り当てた場合、それ以降、オブジェクトのサイズが 500 バイトを超えない限り、オブジェクトを保存する際に追加のディスク領域を要求されることはありません。

ユーザーがこのダイアログボックスに応答した後、このメソッドがもう一度呼び出され、true または false が返されます。また、SharedObject.onStatus が SharedObject.Flush.Success または SharedObject.Flush.Failed の code プロパティを使用して呼び出されます。

詳細については、SharedObject クラス概要の「ローカルディスク領域についての考慮事項」を参照してください。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

minDiskSpace:Number (オプション) - このオブジェクトに割り当てる必要のあるバイト数を示す整数。デフォルト値は 0 です。

戻り値

Object - ブール値 : 次のように、true または false、あるいは "pending" のストリング値を返します。

- ユーザーがこのドメインからのオブジェクトに対してローカル情報記憶域を許可し、オブジェクトを保存するのに十分な領域が割り当てられた場合は、このメソッドは true を返します。minimumDiskSpace に値を指定した場合は、この値以上の領域が割り当てられないと、true は返されません。
- ユーザーがこのドメインからのオブジェクトに対してローカル情報記憶域を許可した場合でも、割り当てられた領域がオブジェクトを保存するのに十分でないと、このメソッドは "pending" を返します。
- ユーザーがこのドメインからのオブジェクトに対してローカル情報記憶域を永続的に禁止した場合、または何かの理由で Flash がオブジェクトを保存できない場合は、このメソッドは false を返します。

メモ : ディスクへのサードパーティの共有オブジェクトの書き込みを許可していない場合でも、ローカルコンテンツでは、常にサードパーティドメイン (現在のブラウザのアドレスバーにあるドメイン以外のドメイン) からの共有オブジェクトをディスクに書き込むことができます。

例

次の関数は、共有オブジェクト my_so, を取得し、設定可能なプロパティにユーザーが指定した値を設定します。最後に flush() を呼び出して設定を保存し、最小 1000 バイトのディスク領域を割り当てます。

```
this.syncSettingsCore = function(soName:String, override:Boolean,
  settings:Object) {
  var my_so:SharedObject = SharedObject.getLocal(soName, "http://
  www.mydomain.com/app/sys");
  // settings list index
  var i;
  // For each specified value in settings:
  // If override is true, set the persistent setting to the provided value.
  // If override is false, fetch the persistent setting, unless there
  // isn't one, in which case, set it to the provided value.
  for (i in settings) {
  if (override || (my_so.data[i] == null)) {
    my_so.data[i] = settings[i];
  } else {
    settings[i] = my_so.data[i];
  }
  }
  my_so.flush(1000);
};
```

関連項目

clear (SharedObject.clear メソッド), onStatus (SharedObject.onStatus ハンドラ)

getLocal (SharedObject.getLocal メソッド)

```
public static getLocal(name:String, [localPath:String], [secure:Boolean]) :
  SharedObject
```

現在のクライアントだけが利用できるローカル永続共有オブジェクトへの参照を返します。共有オブジェクトがまだ存在しない場合は、このメソッドにより作成されます。このメソッドは、SharedObject クラスの静的メソッドです。オブジェクトを変数に代入するには、次のようなシンタックスを使用します。

```
var so:SharedObject = SharedObject.getLocal("savedData")
```

メモ : ユーザーがこのドメインに対してローカル記憶域を許可しない場合は、localPath の値が指定されていても、オブジェクトはローカルに保存されません。ただし、ローカルコンテンツは例外です。ディスクへのサードパーティの共有オブジェクトの書き込みを許可していない場合でも、ローカルコンテンツでは、常にサードパーティドメイン (現在のブラウザのアドレスバーにあるドメイン以外のドメイン) からの共有オブジェクトをディスクに書き込むことができます。

名前の競合を避けるために、共有オブジェクトを作成している SWF ファイルの位置が考慮されます。たとえば、www.myCompany.com/apps/stockwatcher.swf にある SWF ファイルが portfolio という名前の共有オブジェクトを作成した場合、この共有オブジェクトは、www.yourCompany.com/photoshoot.swf にある SWF ファイルが作成した portfolio という別のオブジェクトとは競合しません。これは、この 2 つの SWF ファイルが異なるディレクトリに置かれているからです。

localPath パラメータはオプションですが、慎重に使用する必要があります。特に、他の SWF ファイルが共有オブジェクトにアクセスしなければならない場合には注意してください。共有オブジェクトのデータが、別の場所に移動されない 1 つの SWF ファイルに固有のものである場合は、デフォルト値を使用することをお勧めします。他の SWF ファイルが共有オブジェクトにアクセスする必要がある場合、または共有オブジェクトを作成する SWF ファイルを後で移動する場合には、このパラメータの値により、SWF ファイルから共有オブジェクトにアクセスできるかどうかに影響が生じます。たとえば、localPath を SWF ファイルへの完全パスのデフォルト値に設定し、共有オブジェクトを作成すると、他の SWF ファイルは共有オブジェクトにアクセスできません。元の SWF ファイルを後で別の場所に移動すると、その SWF ファイルからも、共有オブジェクトに格納されているデータにアクセスできなくなります。

localpath パラメータを使用することにより、共有オブジェクトへのアクセスを誤って制限してしまう可能性を低減できます。最も制限の緩やかなオプションは、localPath パラメータを "/" に設定する方法です。ドメインのすべての SWF ファイルが共有オブジェクトを利用できますが、ドメイン内の他の共有オブジェクトとの名前の競合が起こりやすくなります。最も制限の厳しいオプションを設定する場合は、localPath パラメータに、SWF ファイルへの完全パスに含まれるフォルダ名を付加します。たとえば、www.myCompany.com/apps/stockwatcher.swf で SWF ファイルによって作成された portfolio 共有オブジェクトの localPath パラメータオプションは、"/"、"/apps"、および "/apps/stockwatcher.swf" です。アプリケーションに最適な柔軟性を提供できるオプションがどれかを判断する必要があります。

このメソッドを使用するときは、Flash Player セキュリティモデルを考慮してください。

- サンドボックスの境界を越えて共有オブジェクトにアクセスすることはできません。
- [Flash Player の設定] ダイアログボックスまたは設定マネージャを使用して、共有オブジェクトへのアクセスを制限できます。デフォルトでは、ドメインごとに最大 100K のデータサイズまでの共有オブジェクトを作成できます。管理ユーザーおよび一般ユーザーは、ファイルシステムへの書き込み機能に制限を適用することもできます。

ローカルファイル (ローカルにインストールされた SWF ファイルまたはプロジェクタ (EXE)) として再生する SWF ファイルコンテンツをパブリッシュし、複数のローカル SWF ファイルから特定の共有オブジェクトにアクセスする必要がある場合、ローカルファイルでは、共有オブジェクトの保存に 2 つの異なる場所を使用できることに注意してください。使用されるドメインは、共有オブジェクトを作成したローカルファイルに付与されるセキュリティ許可によって変わります。ローカルファイルには、次の 3 つの異なる許可レベルを設定できます。1 つ目はローカルファイルシステム専用アクセス、2 つ目はネットワーク専用アクセス、3 つ目はネットワークとローカルファイルシステム両方のアクセスです。ローカルファイルシステムへのアクセス権を持つローカルファイル (1 または 3) は、共有オブジェクトを 1 つの場所に格納します。ローカルファイルシステムへのアクセス権を持たないローカルファイル (2) は、共有オブジェクトを別の場所に格納します。詳細については、次の参照先を参照してください。

- 『ActionScript 2.0 の学習』の「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

`name:String` - オブジェクトの名前を表すストリング。名前にはスラッシュ (/) を指定できます。たとえば、`work/addresses` は有効な名前です。共有オブジェクト名にスペース、および以下の文字を含めることはできません。

~ % & \ ; : " ' , < > ? #

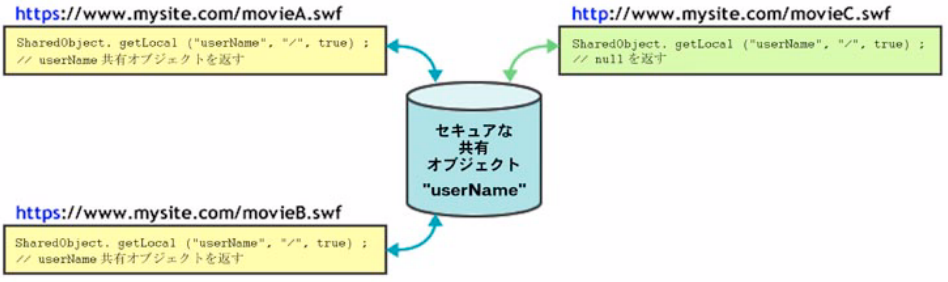
`localPath:String` (オプション) - 共有オブジェクトを作成した SWF ファイルの完全パスまたは部分パスを示すストリング。これによって、共有オブジェクトをローカルに保存する位置が決まります。デフォルト値は完全パスです。

`secure:Boolean` (オプション) - (Flash Player 8 のみ) この共有オブジェクトへのアクセスが HTTPS 接続経由で配信される SWF ファイルに制限されるかどうかを判別します。SWF ファイルが HTTPS 経由で配信されるとした場合 :

- このパラメータを `true` に設定すると、Flash Player は新しいセキュアな共有オブジェクトを作成するか、既存のセキュアな共有オブジェクトの参照を取得します。このセキュアな共有オブジェクトは、HTTPS 経由で配信される SWF ファイルからのみ読み取るか、またはこの SWF ファイルにのみ書き込むことができ、`true` に設定された `secure` パラメータを持つ `SharedObject.getLocal()` を呼び出します。
- このパラメータを `false` に設定すると、Flash Player は新しい共有オブジェクトを作成するか、既存の共有オブジェクトの参照を取得します。この共有オブジェクトは、HTTPS 以外の接続経由で配信される SWF ファイルからのみ読み取るか、またはこの SWF ファイルにのみ書き込むことができます。

SWF ファイルが HTTPS 以外の接続経由で配信される場合、このパラメータを `true` に設定しようとすると、新しい共有オブジェクトの作成 (または作成済みのセキュアな共有オブジェクトへのアクセス) は失敗し、`null` が返されます。このパラメータの値に関係なく、作成された共有オブジェクトはドメインで使用可能なディスク領域の総容量を対象とします。デフォルト値は `false` です。

次の図に、`secure` パラメータの使用方法を示します。



戻り値

SharedObject - ローカルに永続化され、現在のクライアントでのみ利用できる共有オブジェクトへの参照が返されます。共有オブジェクトを作成または見つけられない場合 (localPath に指定したディレクトリが存在しない場合や secure パラメータが正しく使用されていない場合など) は、null が返されます。

サードパーティの Flash コンテンツによる永続共有オブジェクトの作成と保存が禁止されている (ローカルコンテンツに適用されない) 場合、このメソッドは失敗し、null を返します。ユーザーは、http://www.adobe.com/support/documentation/en/flashplayer/help/settings_manager03.html にある設定マネージャの [グローバルストレージ設定] パネルでサードパーティの永続共有オブジェクトを禁止できます。

例

次の例では、TextInput コンポーネントインスタンスに入力されるテキストを格納する共有オブジェクトを作成します。結果の SWF ファイルは、再生を開始するときに、保存されたテキストを共有オブジェクトからロードします。ユーザーが Enter キーを押すたびに、テキストフィールドのテキストが共有オブジェクトに書き込まれます。この例を使用するには、TextInput コンポーネントをステージにドラッグし、このインスタンスに myText_ti という名前をつけます。次のコードをメインタイムラインにコピーします (ステージの空の領域をクリックするか、Esc キーを押して、コンポーネントからフォーカスを削除します)。

```
// Create the shared object and set localpath to server root.
var my_so:SharedObject = SharedObject.getLocal("savedText", "/");
// Load saved text from the shared object into the myText_ti TextInput component.
myText_ti.text = my_so.data.myTextSaved;
// Assign an empty string to myText_ti if the shared object is undefined
// to prevent the text input box from displaying "undefined" when
// this script is first run.
if (myText_ti.text == undefined) {
  myText_ti.text = "";
}
// Create a listener object and function for <enter> event
var textListener:Object = new Object();
textListener.enter = function(eventObj:Object) {
  my_so.data.myTextSaved = eventObj.target.text;
  my_so.flush();
};
// Register the listener with the TextInput component instance
myText_ti.addEventListener("enter", textListener);
```

次の例では、再生した最後のフレームをローカル共有オブジェクト kookie に保存しています。

```
// Get the kookie
var my_so:SharedObject = SharedObject.getLocal("kookie");

// Get the user of the kookie and go to the frame number saved for this user.
if (my_so.data.user != undefined) {
  this.user = my_so.data.user;
  this.gotoAndStop(my_so.data.frame);
}
```

SWF ファイルの各フレームに次のコードを挿入します。

```
// On each frame, call the rememberme function to save the frame number.
function rememberme() {
  my_so.data.frame=this._currentframe;
  my_so.data.user="John";
}
```

getSize (SharedObject.getSize メソッド)

```
public getSize() : Number
```

共有オブジェクトの現在のサイズ (バイト数) を取得します。

すべてのデータプロパティを順に確認することによって、共有オブジェクトのサイズが計算されます。オブジェクトが持つデータプロパティが多いほど、サイズの計算に時間がかかります。オブジェクトのサイズを調べる処理は、非常に時間がかかる場合があり、特に必要がない限り、このメソッドの使用を避けることができます。

対応バージョン : ActionScript 1.0、Flash Player 6

戻り値

Number - 共有オブジェクトのサイズ (バイト数) を示す数値。

例

次の例では、共有オブジェクト my_so のサイズを取得します。

```
var items_array:Array = new Array(101, 346, 483);
var currentUserIsAdmin:Boolean = true;
var currentUserName:String = "Ramona";

var my_so:SharedObject = SharedObject.getLocal("superfoo");
my_so.data.itemNumbers = items_array;
my_so.data.adminPrivileges = currentUserIsAdmin;
my_so.data.userName = currentUserName;

var soSize:Number = my_so.getSize();
trace(soSize);
```

onStatus (SharedObject.onStatus ハンドラ)

```
onStatus = function(infoObject:Object) {}
```

共有オブジェクトに対してエラー、警告、情報通知が送信されたときに呼び出されます。このイベントハンドラに応答するには、共有オブジェクトによって生成される情報オブジェクトを処理する関数を作成する必要があります。

情報オブジェクトには、onStatus ハンドラの結果ストリングを含む code プロパティと、"Status" または "Error" のいずれかのストリングを含む level プロパティがあります。

onStatus ハンドラ以外に、Flash には System.onStatus というスーパー関数もあります。特定のオブジェクトに対して onStatus が呼び出され、それに応答する関数が割り当てられていない場合、System.onStatus に割り当てられた関数があれば、この関数が実行されます。

特定の SharedObject アクティビティが発生した場合、次のイベントによって通知されます。

code プロパティ	level プロパティ	説明
SharedObject.Flush.Failed	Error	"pending" を返した SharedObject.flush() コマンドが失敗しました。[ローカル記憶領域] ダイアログボックスが表示が表示されたときに、ユーザーが、共有オブジェクトに対して追加のディスク領域を割り当てませんでした。
SharedObject.Flush.Success	Status	"pending" を返した SharedObject.flush() コマンドが正常に完了しました。ユーザーは、共有オブジェクトに対して追加のディスク領域を割り当てました。

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

infoObject:Object - ステータスメッセージに従って定義されるパラメータ。

例

次の例では、ユーザーが SharedObject オブジェクトインスタンスのディスクへの書き込みを許可したかどうかに応じて、異なるメッセージを表示します。

```
var message_str:String;
this.createTextField("message_txt", this.getNextHighestDepth(), 0, 0, 300, 22);
message_txt.html = true;
this.createTextField("status_txt", this.getNextHighestDepth(), 10, 30, 300,
  100);
status_txt.multiline = true;
status_txt.html = true;
```

```
var items_array:Array = new Array(101, 346, 483);
var currentUserIsAdmin:Boolean = true;
var currentUserName:String = "Ramona";
var my_so:SharedObject = SharedObject.getLocal("superfoo");
my_so.data.itemNumbers = items_array;
my_so.data.adminPrivileges = currentUserIsAdmin;
my_so.data.userName = currentUserName;

my_so.onStatus = function(infoObject:Object) {
  status_txt.htmlText = "<textformat tabStops='[50]'>";
  for (var i in infoObject) {
    status_txt.htmlText += "<b>"+i+"</b>"+"\t"+infoObject[i];
  }
  status_txt.htmlText += "</textformat>";
};

var flushResult = my_so.flush(1000001);
switch (flushResult) {
case 'pending' :
  message_str = "flush is pending, waiting on user interaction.";
  break;
case true :
  message_str = "flush was successful. Requested storage space approved.";
  break;
case false :
  message_str = "flush failed. User denied request for additional storage.";
  break;
}
message_txt.htmlText = "<a href=\"asfunction:System.showSettings,1\
"><u>"+message_str+"</u></a>";
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

getLocal (SharedObject.getLocal メソッド),onStatus (System.onStatus ハンドラ)

Sound

```
Object
  |
  +-Sound
```

```
public class Sound
extends Object
```

Sound クラスを使用すると、ムービー内のサウンドを制御することができます。ムービーの再生中にライブラリからムービークリップにサウンドを追加し、追加したサウンドを制御することができます。新しい Sound オブジェクトの作成時に target を指定しない場合は、このクラスのメソッドを使用してムービー全体のサウンドを制御できます。

Sound クラスのメソッドを呼び出す前に、コンストラクタ `new Sound` を使用して Sound オブジェクトを作成する必要があります。

対応バージョン： ActionScript 1.0、Flash Player 5

プロパティー覧

オプション	プロパティ	説明
	`checkPolicyFile:Boolean`	サウンド自体のロードを開始する前に、Flash Player が、ロードされるサウンドのサーバーからクロスドメインポリシーファイルをダウンロードしようとするかどうかを指定します。
	`duration:Number` (読み取り専用)	ミリ秒数で示したサウンドの継続時間です。
	`id3:Object` (読み取り専用)	MP3 ファイルの一部であるメタデータに対するアクセスを提供します。
	`position:Number` (読み取り専用)	サウンドを再生しているミリ秒数です。

Object クラスから継承されるプロパティ

`constructor` (Object.constructor プロパティ), `__proto__` (Object.__proto__ プロパティ), `prototype` (Object.prototype プロパティ), `__resolve` (Object.__resolve プロパティ)

イベントの一覧

イベント	説明
`onID3 = function() {}`	`Sound.attachSound()` または `Sound.loadSound()` を使用してロードした MP3 ファイルについて、新しい ID3 データが利用できるようになるたびに呼び出されます。
`onLoad = function(success: Boolean) {}`	サウンドがロードされると、自動的に呼び出されます。
`onSoundComplete = function() {}`	サウンドの再生が終了すると、自動的に呼び出されます。

コンストラクター一覧

署名	説明
`Sound([target: Object])`	指定されたムービークリップの新しい Sound オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	`attachSound(id: String) : Void`	id パラメータで指定されたサウンドを、指定された Sound オブジェクトに割り当てます。
	`getBytesLoaded() : Number`	指定した Sound オブジェクトに対してロード (ストリーミング) されたバイト数を返します。
	`getBytesTotal() : Number`	指定した Sound オブジェクトのサイズをバイト単位で返します。
	`getPan() : Number`	最後の `setPan()` 呼び出しで設定されたパンレベルを返します。戻り値は -100 (左) ～ +100 (右) の整数です。
	`getTransform() : Object`	最後の `Sound.setTransform()` 呼び出しで設定された Sound オブジェクトのサウンド変換情報を返します。
	`getVolume() : Number`	サウンドボリュームレベルを 0 ～ 100 の整数で返します。ここで、0 はボリュームオフ、100 はフルボリュームです。
	`loadSound(url: String, isStreaming: Boolean) : Void`	MP3 ファイルを Sound オブジェクトにロードします。
	`setPan(value: Number) : Void`	左右のチャネル (スピーカー) でサウンドを再生する方法を決定します。

オプション	署名	説明
	setTransform (transformObject: Object) : Void	Sound オブジェクトにサウンド変換情報、つまり " バランス " 情報を設定します。
	setVolume(value: Number) : Void	Sound オブジェクトにボリュームを設定します。
	start([secondOffset: Number], [loops:Number]) : Void	最後に割り当てたサウンドの再生を開始します。再生を開始する位置は、パラメータを指定しない場合はサウンドの先頭、指定する場合は secondOffset パラメータで指定したサウンド内のポイントです。
	stop([linkageID: String]) : Void	idName パラメータで指定されたサウンドを停止します。パラメータを指定しないと、現在再生中のすべてのサウンドを停止します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

attachSound (Sound.attachSound メソッド)

```
public attachSound(id:String) : Void
```

id パラメータで指定されたサウンドを、指定された Sound オブジェクトに割り当てます。サウンドは、現在の SWF ファイルのライブラリ内に存在し、[リンケージプロパティ] ダイアログボックスで書き出しを指定する必要があります。サウンドの再生を開始するには、Sound.start() を呼び出します。

SWF ファイルのすべてのシーンからサウンドを制御できるようにするには、サウンドを SWF のメインタイムラインに置きます。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

id:String - ライブラリ内の書き出されたサウンドの識別子。識別子は、[リンケージプロパティ] ダイアログボックスで設定します。

例

次の例では、my_sound にサウンド logoff_id をアタッチします。ライブラリ内のサウンドはリンケージ識別子 logoff_id を持っています。

```
var my_sound:Sound = new Sound();
my_sound.attachSound("logoff_id");
my_sound.start();
```

checkPolicyFile (Sound.checkPolicyFile property)

public checkPolicyFile : Boolean

サウンド自体のロードを開始する前に、Flash Player が、ロードされるサウンドのサーバーからクロスドメインポリシーファイルをダウンロードしようとするかどうかを指定します。このフラグは、Sound.loadSound() には適用できますが、loadSound が呼び出されていない Sound オブジェクトには適用できません。

サウンドを呼び出し元 SWF ファイル自体のドメイン外からロードし、そのサウンドのコンテンツに ActionScript からアクセスする必要がある場合に、このフラグを true に設定します。サウンドコンテンツへのアクセスの例には、MP3 メタデータを取得するための Sound.id3 プロパティの参照があります。ロード時に、指定された checkPolicyFile を持たずにこれらの操作のいずれかを試行した場合、セキュリティエラーが発生する場合があります。必要なポリシーファイルがまだロードされていないためです。

checkPolicyFile を true に設定して Sound.loadSound() を呼び出す場合、関連するクロスドメインポリシーファイルを正常にダウンロードするか、そのようなポリシーファイルが存在しないことがわかるまで、Flash Player は loadSound() で指定されたサウンドのダウンロードを開始しません。Flash Player では、最初に、既にダウンロードされているポリシーファイルが考慮され、次に System.security.loadPolicyFile() の呼び出しで指定された保留されているポリシーファイルのダウンロードが試行されます。次に、loadSound() で指定された URL に対応するデフォルトの場所からポリシーファイルのダウンロードが試行されます。これは、loadSound() と同じサーバーの /crossdomain.xml です。どのような場合でも、Flash Player では、指定されたポリシーファイルがそのサーバー上に存在しており、ポリシーファイルの場所に応じてサウンドファイルへのアクセスが提供されており、ポリシーファイルにより、<allow-access-from> タグに応じて呼び出し元 SWF ファイルのドメインによるアクセスが許可されている必要があります。

checkPolicyFile を true に設定した場合、Flash Player ではポリシーファイルが完了するまで、loadSound() で指定する主要なダウンロードの開始が待機されます。したがって、必要なポリシーファイルが存在している場合には、Sound オブジェクトから onLoad イベントを受け取るとすぐにポリシーファイルのダウンロードが完了し、ポリシーファイルが必要な操作を安全に開始できます。

checkPolicyFile を true に設定し、関連するポリシーファイルが見つからない場合、セキュリティエラーの原因となる操作を試行するまで、エラーは表示されません。

ロードするサウンドの内容へのアクセスを必要としない場合は、checkPolicyFile を true に設定しないでください。この場合、ポリシーファイルの確認は時間の浪費になります。ダウンロードの開始が遅れ、ネットワーク帯域幅を不必要に消費する場合があるためです。

サーバーサイド HTTP リダイレクトを使用する可能性がある URL からサウンドをダウンロードする場合は、checkPolicyFile に注意してください。Flash Player は、常に loadSound() で指定する初期 URL に対応するポリシーファイルを取得しようとします。最終的なサウンドファイルが HTTP リダイレクトによって別の URL から取得される場合、最初にダウンロードされたポリシーファイルはサウンドの最終的な URL に適用できないことがあります。そのため、URL はセキュリティ判定において重要です。

対応バージョン： ActionScript 2.0、Flash Player 9

duration (Sound.duration プロパティ)

public duration : Number (読み取り専用)

ミリ秒数で示したサウンドの継続時間です。

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の例では、サウンドをロードし、[出力] パネルにサウンドファイルの継続時間を表示します。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
var my_sound:Sound = new Sound();
my_sound.onLoad = function(success:Boolean) {
  var totalSeconds:Number = this.duration/1000;
  trace(this.duration+" ms ("+Math.round(totalSeconds)+" seconds)");
  var minutes:Number = Math.floor(totalSeconds/60);
  var seconds = Math.floor(totalSeconds)%60;
  if (seconds<10) {
  seconds = "0"+seconds;
  }
  trace(minutes+":"+seconds);
};
my_sound.loadSound("song1.mp3", true);
```

次の例では、SWF ファイルにいくつかの音楽をロードします。Drawing API を使用して作成したプログレスバーは、ロードの進行状況を表示します。音楽が始まってロードが完了すると、[出力] パネルに情報が表示されます。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
var pb_height:Number = 10;
var pb_width:Number = 100;
var pb:MovieClip = this.createEmptyMovieClip("progressBar_mc",
  this.getNextHighestDepth());
pb.createEmptyMovieClip("bar_mc", pb.getNextHighestDepth());
pb.createEmptyMovieClip("vBar_mc", pb.getNextHighestDepth());
pb.createEmptyMovieClip("stroke_mc", pb.getNextHighestDepth());
pb.createTextField("pos_txt", pb.getNextHighestDepth(), 0, pb_height, pb_width,
  22);
```

```
pb._x = 100;
pb._y = 100;

with (pb.bar_mc) {
  beginFill(0x00FF00);
  moveTo(0, 0);
  lineTo(pb_width, 0);
  lineTo(pb_width, pb_height);
  lineTo(0, pb_height);
  lineTo(0, 0);
  endFill();
  _xscale = 0;
}
with (pb.vBar_mc) {
  lineStyle(1, 0x000000);
  moveTo(0, 0);
  lineTo(0, pb_height);
}
with (pb.stroke_mc) {
  lineStyle(3, 0x000000);
  moveTo(0, 0);
  lineTo(pb_width, 0);
  lineTo(pb_width, pb_height);
  lineTo(0, pb_height);
  lineTo(0, 0);
}

var my_interval:Number;
var my_sound:Sound = new Sound();
my_sound.onLoad = function(success:Boolean) {
  if (success) {
  trace("sound loaded");
  }
};
my_sound.onSoundComplete = function() {
  clearInterval(my_interval);
  trace("Cleared interval");
}
my_sound.loadSound("song3.mp3", true);
my_interval = setInterval(updateProgressBar, 100, my_sound);

function updateProgressBar(the_sound:Sound):Void {
  var pos:Number = Math.round(the_sound.position/the_sound.duration*100);
  pb.bar_mc._xscale = pos;
  pb.vBar_mc._x = pb.bar_mc._width;
  pb.pos_txt.text = pos+"%";
}
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

position (Sound.position プロパティ)

getBytesLoaded (Sound.getBytesLoaded メソッド)

public getBytesLoaded() : Number

指定した Sound オブジェクトに対してロード (ストリーミング) されたバイト数を返します。getBytesLoaded() の値と getBytesTotal() の値を比較すると、サウンドの何 % がロードされたかを判断できます。

対応バージョン: ActionScript 1.0、Flash Player 6

戻り値

Number - ロードされたバイト数を示す整数。

例

次の例では、SWF ファイルにロードするサウンドファイルの、ロードされたバイト数と合計バイト数を表示する 2 つのテキストフィールドをダイナミックに作成します。ファイルがロードを終了すると、テキストフィールドにはメッセージが表示されます。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
this.createTextField("message_txt", this.getNextHighestDepth(), 10,10,300,22)
this.createTextField("status_txt", this.getNextHighestDepth(), 10, 50, 300, 40);
status_txt.autoSize = true;
status_txt.multiline = true;
status_txt.border = false;

var my_sound:Sound = new Sound();
my_sound.onLoad = function(success:Boolean) {
  if (success) {
  this.start();
  message_txt.text = "Finished loading";
  }
};
my_sound.onSoundComplete = function() {
  message_txt.text = "Clearing interval";
  clearInterval(my_interval);
};
```

```
my_sound.loadSound("song2.mp3", true);
var my_interval:Number;
my_interval = setInterval(checkProgress, 100, my_sound);
function checkProgress(the_sound:Sound):Void {
  var pct:Number = Math.round(the_sound.getBytesLoaded()/
  the_sound.getBytesTotal()*100);
  var pos:Number = Math.round(the_sound.position/the_sound.duration*100);
  status_txt.text = the_sound.getBytesLoaded()+" of
  "+the_sound.getBytesTotal()+" bytes ("+pct+"%)"+newline;
  status_txt.text += the_sound.position+" of "+the_sound.duration+" milliseconds
  ("+pos+"%)"+newline;
}
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

getBytesTotal (Sound.getBytesTotal メソッド)

getBytesTotal (Sound.getBytesTotal メソッド)

```
public getBytesTotal() : Number
```

指定した Sound オブジェクトのサイズをバイト単位で返します。

対応バージョン : ActionScript 1.0、Flash Player 6

戻り値

Number - 指定した Sound オブジェクトの総サイズをバイト単位で示す整数。

例

このメソッドの使用例については Sound.getBytesLoaded() を参照してください。

関連項目

getBytesLoaded (Sound.getBytesLoaded メソッド)

getPan (Sound.getPan メソッド)

```
public getPan() : Number
```

最後の setPan( ) 呼び出しで設定されたパンレベルを返します。戻り値は -100 (左) ～ +100 (右) の整数です。0 は左右のチャネルを等しい値に設定します。パンの設定では、SWF ファイル内での現在と今後のサウンドの左右バランスを制御します。

このメソッドの効果は setVolume( ) または setTransform( ) の効果に追加されます。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Number - 整数。

例

次の例では、Drawing API を使用してスライダバーを作成します。ユーザーがスライダバーをドラッグすると、ロードされたサウンドのパンレベルが変更されます。ダイナミックに作成されたテキストフィールドに、現在のパンレベルが表示されます。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
var bar_width:Number = 200;
this.createEmptyMovieClip("bar_mc", this.getNextHighestDepth());
with (bar_mc) {
  lineStyle(4, 0x000000);
  moveTo(0, 0);
  lineTo(bar_width+4, 0);
  lineStyle(0, 0x000000);
  moveTo((bar_width/2)+2, -8);
  lineTo((bar_width/2)+2, 8);
}
bar_mc._x = 100;
bar_mc._y = 100;

this.createEmptyMovieClip("knob_mc", this.getNextHighestDepth());
with (knob_mc) {
  lineStyle(0, 0x000000);
  beginFill(0xCCCCCC);
  moveTo(0, 0);
  lineTo(4, 0);
  lineTo(4, 10);
  lineTo(0, 10);
  lineTo(0, 0);
  endFill();
}
knob_mc._x = bar_mc._x+(bar_width/2);
knob_mc._y = bar_mc._y-(knob_mc._height/2);
```

```
knob_mc.left = knob_mc._x-(bar_width/2);
knob_mc.right = knob_mc._x+(bar_width/2);
knob_mc.top = knob_mc._y;
knob_mc.bottom = knob_mc._y;

knob_mc.onPress = function() {
  this.startDrag(false, this.left, this.top, this.right, this.bottom);
};
knob_mc.onRelease = panSound;
knob_mc.onReleaseOutside = panSound;

function panSound():Void { {
  this.stopDrag();
  var multiplier:Number = 100/(this.right-this.left)*2;
  var pan:Number = (this._x-this.left-(bar_width/2))*multiplier;
  my_sound.setPan(pan);
  pan_txt.text = my_sound.getPan();
};

var my_sound:Sound = new Sound();
my_sound.loadSound("song2.mp3", true);
this.createTextField("pan_txt", this.getNextHighestDepth(), knob_mc._x,
  knob_mc._y+knob_mc._height, 20, 22);
pan_txt.selectable = false;
pan_txt.autoSize = "center";
pan_txt.text = my_sound.getPan();
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

setPan (Sound.setPan メソッド)

getTransform (Sound.getTransform メソッド)

public getTransform() : Object

最後の Sound.setTransform() 呼び出しで設定された Sound オブジェクトのサウンド変換情報を返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Object - 指定した Sound オブジェクトのチャネルパーセント値を含むプロパティを持つオブジェクト。

例

次の例では、ライブラリ内のシンボルから、4 つのムービークリップを割り当てます (リンケージ識別子 knob_id)。これらは、スライダ (または、つまみ) として使用され、SWF ファイルにロードされるサウンドファイルを制御します。これらのスライダは、変換オブジェクトを制御したり、サウンドファイルのバランスを調整します。詳細については、Sound.setTransform( ) を参照してください。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
var my_sound:Sound = new Sound();
my_sound.loadSound("song1.mp3", true);
var transform_obj:Object = my_sound.getTransform();

this.createEmptyMovieClip("transform_mc", this.getNextHighestDepth());
transform_mc.createTextField("transform_txt", transform_mc.getNextHighestDepth,
  0, 8, 120, 22);
transform_mc.transform_txt.html = true;

var knob_ll:MovieClip = transform_mc.attachMovie("knob_id", "ll_mc",
  transform_mc.getNextHighestDepth(), {_x:0, _y:30});
var knob_lr:MovieClip = transform_mc.attachMovie("knob_id", "lr_mc",
  transform_mc.getNextHighestDepth(), {_x:30, _y:30});
var knob_rl:MovieClip = transform_mc.attachMovie("knob_id", "rl_mc",
  transform_mc.getNextHighestDepth(), {_x:60, _y:30});
var knob_rr:MovieClip = transform_mc.attachMovie("knob_id", "rr_mc",
  transform_mc.getNextHighestDepth(), {_x:90, _y:30});

knob_ll.top = knob_ll._y;
knob_ll.bottom = knob_ll._y+100;
knob_ll.left = knob_ll._x;
knob_ll.right = knob_ll._x;
knob_ll._y = knob_ll._y+(100-transform_obj['ll']);
knob_ll.onPress = pressKnob;
knob_ll.onRelease = releaseKnob;
knob_ll.onReleaseOutside = releaseKnob;

knob_lr.top = knob_lr._y;
knob_lr.bottom = knob_lr._y+100;
knob_lr.left = knob_lr._x;
knob_lr.right = knob_lr._x;
knob_lr._y = knob_lr._y+(100-transform_obj['lr']);
knob_lr.onPress = pressKnob;
knob_lr.onRelease = releaseKnob;
knob_lr.onReleaseOutside = releaseKnob;

knob_rl.top = knob_rl._y;
knob_rl.bottom = knob_rl._y+100;
knob_rl.left = knob_rl._x;
knob_rl.right = knob_rl._x;
knob_rl._y = knob_rl._y+(100-transform_obj['rl']);
knob_rl.onPress = pressKnob;
```

```
knob_rl.onRelease = releaseKnob;
knob_rl.onReleaseOutside = releaseKnob;

knob_rr.top = knob_rr._y;
knob_rr.bottom = knob_rr._y+100;
knob_rr.left = knob_rr._x;
knob_rr.right = knob_rr._x;
knob_rr._y = knob_rr._y+(100-transform_obj['rr']);
knob_rr.onPress = pressKnob;
knob_rr.onRelease = releaseKnob;

knob_rr.onReleaseOutside = releaseKnob;

updateTransformTxt();

function pressKnob() {
  this.startDrag(false, this.left, this.top, this.right, this.bottom);
}
function releaseKnob() {
  this.stopDrag();
  updateTransformTxt();
}
function updateTransformTxt() {
  var ll_num:Number = 30+100-knob_ll._y;
  var lr_num:Number = 30+100-knob_lr._y;
  var rl_num:Number = 30+100-knob_rl._y;
  var rr_num:Number = 30+100-knob_rr._y;
  my_sound.setTransform({ll:ll_num, lr:lr_num, rl:rl_num, rr:rr_num});
  transform_mc.transform_txt.htmlText = "<textformat tabStops='[0,30,60,90]'>";
  transform_mc.transform_txt.htmlText +=
  ll_num+"\t"+lr_num+"\t"+rl_num+"\t"+rr_num;
  transform_mc.transform_txt.htmlText += "</textformat>";
}
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

setTransform（Sound.setTransform メソッド）

getVolume (Sound.getVolume メソッド)

```
public getVolume() : Number
```

サウンドボリュームレベルを 0 ～ 100 の整数で返します。ここで、0 はボリュームオフ、100 はフルボリュームです。デフォルト設定は 100 です。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

`Number` - 整数。

例

次の例では、Drawing API および実行時に作成されるムービークリップを使用してスライダを作成します。ダイナミックに作成されたテキストフィールドに、SWF ファイルで再生されているサウンドの現在のボリュームレベルが表示されます。次の ActionScript を AS ファイルまたは FLA ファイルに追加します。

```
var my_sound:Sound = new Sound();
my_sound.loadSound("song3.mp3", true);

this.createEmptyMovieClip("knob_mc", this.getNextHighestDepth());

knob_mc.left = knob_mc._x;
knob_mc.right = knob_mc.left+100;
knob_mc.top = knob_mc._y;
knob_mc.bottom = knob_mc._y;

knob_mc._x = my_sound.getVolume();

with (knob_mc) {
  lineStyle(0, 0x000000);
  beginFill(0xCCCCCC);
  moveTo(0, 0);
  lineTo(4, 0);
  lineTo(4, 18);
  lineTo(0, 18);
  lineTo(0, 0);
  endFill();
}

knob_mc.createTextField("volume_txt", knob_mc.getNextHighestDepth(),
  knob_mc._width+4, 0, 30, 22);
knob_mc.volume_txt.text = my_sound.getVolume();
```

```
knob_mc.onPress = function() {
  this.startDrag(false, this.left, this.top, this.right, this.bottom);
  this.isDragging = true;
};
knob_mc.onMouseMove = function() {
  if (this.isDragging) {
  this.volume_txt.text = this._x;
  }
}
knob_mc.onRelease = function() {
  this.stopDrag();
  this.isDragging = false;
  my_sound.setVolume(this._x);

};
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

setVolume (Sound.setVolume メソッド)

id3 (Sound.id3 プロパティ)

public id3 : Object (読み取り専用)

MP3 ファイルの一部であるメタデータに対するアクセスを提供します。

MP3 サウンドファイルには、ファイルについてのメタデータを示す ID3 タグを含めることができます。Sound.attachSound() または Sound.loadSound() を使用してロードした MP3 サウンドに ID3 タグがある場合は、これらのプロパティを調べることができます。サポートされているのは、UTF-8 文字セットを使用する ID3 タグだけです。

Flash Player 6 (6.0.40.0) 以降は、ID3 1.0 および ID3 1.1 のタグをサポートするために、Sound.id3 プロパティを使用しています。Flash Player 7 では、ID3 2.0 (厳密には 2.3 および 2.4) のタグのサポートが追加されました。次の表に、標準の ID3 2.0 タグと、そのタグが表すコンテンツタイプを示します。これらを調べるには、*my_sound*.id3.COMM、*my_sound*.id3.TIME などの形式を使用します。MP3 ファイルには、この表に示していないタグも含めることができます。Sound.id3 を使用すると、それらのタグにもアクセスできます。

プロパティ	説明
TFLT	ファイル形式
TIME	時刻
TIT1	内容の属するグループの説明
TIT2	タイトル / 曲名 / 内容の説明
TIT3	サブタイトル / 説明の追加情報
TKEY	最初の調
TLAN	言語
TLEN	長さ
TMED	メディアタイプ
TOAL	オリジナルのアルバム / ムービー / ショーのタイトル
TOFN	オリジナルのファイル名
TOLY	オリジナルの作詞家 / 文書作成者
TOPE	オリジナルのアーティスト / 演奏者
TORY	オリジナルのリリース年
TOWN	ファイルの所有者 / ライセンス保持者
TPE1	主な演奏者 / ソリスト
TPE2	バンド / オーケストラ / 伴奏
TPE3	指揮者 / 演奏者詳細情報
TPE4	翻訳、リミックス、その他の修正を行った人
TPOS	セット中の位置
TPUB	発行者
TRCK	トラック番号 / セット内の位置
TRDA	録音日
TRSN	インターネットラジオ局の名前
TRSO	インターネットラジオ局の所有者
TSIZ	サイズ
TSRC	ISRC (国際標準録音資料コード)
TSSE	エンコードに使用したソフトウェア / ハードウェアと設定
TYER	年
WXXX	URL link frame

Flash Player 6 は、いくつかの ID3 1.0 タグをサポートしていました。MP3 ファイル内にこれらのタグがなく、対応する ID3 2.0 タグがある場合は、ID3v2 タグが ID3v1 のプロパティにコピーされます。次の表は、ID3 2.0 タグから ID3 1.0 プロパティへの対応を示しています。このプロセスによって、ID3 1.0 のプロパティを読み取るように作成された既存のスクリプトとの後方互換性が維持されます。

ID3 2.0 タグ	対応する ID3 1.0 プロパティ
COMM	Sound.id3.comment
TALB	Sound.id3.album
TCON	Sound.id3.genre
TIT2	Sound.id3.songname
TPE1	Sound.id3.artist
TRCK	Sound.id3.track
TYER	Sound.id3.year

対応バージョン： ActionScript 1.0、Flash Player 6 - Flash Player 7 ではビヘイビアが更新されました。

例

次の例は、song.mp3 の ID3 プロパティを追跡して [出力] パネルに表示します。

```
var my_sound:Sound = new Sound();
my_sound.onID3 = function(){
   for( var prop in my_sound.id3 ){
   trace( prop + " : "+ my_sound.id3[prop] );
   }
}
my_sound.loadSound("song.mp3", false);
```

関連項目

attachSound (Sound.attachSound メソッド),loadSound (Sound.loadSound メソッド)

loadSound (Sound.loadSound メソッド)

```
public loadSound(url:String, isStreaming:Boolean) : Void
```

MP3 ファイルを Sound オブジェクトにロードします。isStreaming パラメータを使用して、サウンドがイベントサウンドかストリーミングサウンドかを指定できます。

イベントサウンドは、再生前に完全にロードされます。イベントサウンドは、ActionScript Sound クラスによって管理され、このオブジェクトのすべてのメソッドおよびプロパティに応答します。

ストリーミングサウンドは、ダウンロードと並行して再生されます。解凍処理を開始できるだけのデータが受信されると、再生が開始されます。

このメソッドでロードされたすべての MP3 (イベントとストリーミングの両方) は、ユーザーのシステム上のブラウザのファイルキャッシュに保存されます。

このメソッドを使用するときは、Flash Player セキュリティモデルを考慮してください。

Flash Player 8 :

- 呼び出し元 SWF ファイルが ローカルファイルシステムのサンドボックスにあり、サウンドがネットワーク上のサンドボックスにある場合、`Sound.loadSound()` は使用できません。
- 信頼できるローカルのサンドボックスまたはネットワーク接続したローカルのサンドボックスからアクセスするには、クロスドメインポリシーファイルを使用して Web サイトで許可する必要があります。

Flash Player 7 以降 :

- クロスドメインポリシーファイルを使用して、異なるドメイン内のリクエスタからリソースへのアクセスを Web サイトで許可できます。

詳細については、次の参照先を参照してください。

- 『ActionScript 2.0 の学習』の「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

`url:String` - サーバー上の MP3 サウンドファイルの位置。

`isStreaming:Boolean` - ブール値。サウンドがストリーミングサウンドか (`true`)、またはイベントサウンドか (`false`) を示します。

例

次の例では、イベントサウンドをロードします。イベントサウンドはロードが完了するまで再生できません。

```
var my_sound:Sound = new Sound();
my_sound.loadSound("song1.mp3", false);
```

次の例では、ストリーミングサウンドをロードします。

```
var my_sound:Sound = new Sound();
my_sound.loadSound("song1.mp3", true);
```

関連項目

onLoad (Sound.onLoad ハンドラ)

onID3 (Sound.onID3 ハンドラ)

```
onID3 = function() {}
```

Sound.attachSound() または Sound.loadSound() を使用してロードした MP3 ファイルについて、新しい ID3 データが利用できるようになるたびに呼び出されます。このハンドラを使用すると、ポーリングなしで ID3 データにアクセスできます。1 つのファイルに ID3 1.0 タグと ID3 2.0 タグの両方が存在する場合、このハンドラは 2 回呼び出されます。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例では、song1.mp3 ファイル (SWF ファイルと同じディレクトリにある) の ID3 プロパティを出力します。

```
var my_sound:Sound = new Sound();

my_sound.onID3 = function() {
  for (var prop in this.id3) {
    trace(prop + ": " + this.id3[prop])
  }
};

my_sound.loadSound("song1.mp3", true);
```

関連項目

attachSound (Sound.attachSound メソッド), id3 (Sound.id3 プロパティ), loadSound (Sound.loadSound メソッド)

onLoad (Sound.onLoad ハンドラ)

```
onLoad = function(success:Boolean) {}
```

サウンドがロードされると、自動的に呼び出されます。このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。匿名関数または名前付き関数を使用できます。それぞれの例については、Sound.onSoundComplete を参照してください。このハンドラは、mySound.loadSound() を呼び出す前に定義する必要があります。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

success:Boolean - ブール値。my_sound のロードが成功した場合は true、それ以外の場合は false を返します。

例

次の例では、新しい Sound オブジェクトを作成して、サウンドをロードします。サウンドのロードは onLoad ハンドラにより処理されます。音楽が正常にロードされた後に再生を開始できます。新しい FLA ファイルを作成し、次の ActionScript を FLA ファイルまたは AS ファイルに追加します。このコード例が動作するには、song1.mp3 という名前の MP3 ファイルを、FLA ファイルまたは AS ファイルと同じディレクトリに置く必要があります。

```
this.createTextField("status_txt", this.getNextHighestDepth(), 0,0,100,22);

// create a new Sound object
var my_sound:Sound = new Sound();
// if the sound loads, play it; if not, trace failure loading
my_sound.onLoad = function(success:Boolean) {
if (success) {
my_sound.start();
status_txt.text = "Sound loaded";
} else {
status_txt.text = "Sound failed";
}
};
// load the sound
my_sound.loadSound("song1.mp3", true);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

loadSound (Sound.loadSound メソッド)

onSoundComplete (Sound.onSoundComplete ハンドラ)

```
onSoundComplete = function() {}
```

サウンドの再生が終了すると、自動的に呼び出されます。このハンドラを使用して、サウンドの再生が終了したときに SWF ファイル内のイベントをトリガすることができます。

このイベントハンドラが呼び出されたときに実行される関数を定義する必要があります。匿名関数でも名前付き関数でも使用できます。

対応バージョン： ActionScript 1.0、Flash Player 6

例

シンタックス 1: 次の例では、匿名関数を使用します。

```
var my_sound:Sound = new Sound();
my_sound.attachSound("mySoundID");
my_sound.onSoundComplete = function() {
trace("mySoundID completed");
};
my_sound.start();
```

シンタックス 2: 次の例では、名前付き関数を使用します。

```
function callback1() {
trace("mySoundID completed");
}
var my_sound:Sound = new Sound();
my_sound.attachSound("mySoundID");
my_sound.onSoundComplete = callback1;
my_sound.start();
```

関連項目

onLoad（Sound.onLoad ハンドラ）

position (Sound.position プロパティ)

public position : Number（読み取り専用）

サウンドを再生しているミリ秒数。サウンドをループしている場合、サウンド位置は各ループの最初に 0 にリセットされます。

対応バージョン: ActionScript 1.0、Flash Player 6

例

このプロパティの使用例については Sound.duration を参照してください。

関連項目

duration（Sound.duration プロパティ）

setPan (Sound.setPan メソッド)

```
public setPan(value:Number) : Void
```

左右のチャネル (スピーカー) でサウンドを再生する方法を決定します。モノラルサウンドの場合、*pan* は左右いずれのスピーカーからサウンドを再生するかを決定します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`value:Number` - サウンドの左右バランスを指定する整数。有効値の範囲は -100 ～ 100 です。ここで、-100 は左のチャネルのみを使用し、100 は右のチャネルのみを使用します。0 は 2 つのチャネル間でサウンドのバランスを等しくします。

例

このメソッドの使用例については `Sound.getPan()` を参照してください。

関連項目

`attachSound (Sound.attachSound メソッド )`, `getPan (Sound.getPan メソッド )`, `setTransform (Sound.setTransform メソッド )`, `setVolume (Sound.setVolume メソッド )`, `start (Sound.start メソッド )`

setTransform (Sound.setTransform メソッド)

```
public setTransform(transformObject:Object) : Void
```

Sound オブジェクトにサウンド変換情報、つまり " バランス " 情報を設定します。

`soundTransformObject` パラメータは、汎用 Object オブジェクトのコンストラクタメソッドを使用して作成するオブジェクトです。このオブジェクトには、左右のチャンネル (スピーカー) 間でサウンドをどのように配分するかを指定するパラメータがあります。

サウンドは、大量のディスク領域およびメモリを必要とします。ステレオサウンドはモノラルサウンドの 2 倍のデータを使用するため、一般には 22 kHz、6 ビットのモノラルサウンドを使用するのが最善です。`setTransform()` メソッドを使用すると、モノラルサウンドをステレオサウンドとして再生したり、ステレオサウンドをモノラルサウンドのように再生したり、またサウンドに特殊効果を追加することができます。

`soundTransformObject` のプロパティは、次のとおりです。

`ll` - 左スピーカーで再生する左入力データの量を指定するパーセント値 (0 ～ 100)

`lr` - 左スピーカーで再生する右入力データの量を指定するパーセント値 (0 ～ 100)

`rr` - 右スピーカーで再生する右入力データの量を指定するパーセント値 (0 ～ 100)

`rl` - 右スピーカーで再生する左入力データの量を指定するパーセント値 (0 ～ 100)

パラメータの最終結果は、次の式で表されます。

```
leftOutput = left_input * ll + right_input * lr
rightOutput = right_input * rr + left_input * rl
```

left_input または right_input の値は、SWF ファイル内のサウンドのタイプ (ステレオまたはモノラル) によって決定されます。

ステレオサウンドのデフォルトでは、左スピーカーと右スピーカーの間でサウンド入力を等しく分割するように、次の変換設定になっています。

```
ll = 100
lr = 0
rr = 100
rl = 0
```

モノラルサウンドのデフォルトでは、左スピーカーですべてのサウンド入力を再生するように、次の変換設定になっています。

```
ll = 100
lr = 100
rr = 0
rl = 0
```

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

transformObject: Object - 汎用の Object オブジェクト用コンストラクタで作成したオブジェクト。

例

次の例では、setTransform() メソッドでは実現でき、setVolume() メソッドや setPan() メソッドでは両方を組み合わても実現できない設定を示します。

次のコードでは、新しい soundTransformObject オブジェクトを作成し、両方のチャネルのサウンドが左チャネルでのみ再生されるように、オブジェクトのプロパティを設定します。

```
var mySoundTransformObject:Object = new Object();
mySoundTransformObject.ll = 100;
mySoundTransformObject.lr = 100;
mySoundTransformObject.rr = 0;
mySoundTransformObject.rl = 0;
```

soundTransformObject オブジェクトを Sound オブジェクトに適用するには、次のように setTransform() メソッドを使用してオブジェクトを Sound オブジェクトに渡します。

```
my_sound.setTransform(mySoundTransformObject);
```

次の例では、ステレオサウンドをモノラルとして再生します。soundTransformObjectMono オブジェクトは以下のパラメータを使用します。

```
var mySoundTransformObjectMono:Object = new Object();
mySoundTransformObjectMono.ll = 50;
mySoundTransformObjectMono.lr = 50;
mySoundTransformObjectMono.rr = 50;
mySoundTransformObjectMono.rl = 50;
my_sound.setTransform(mySoundTransformObjectMono);
```

次の例では、半分のボリュームで左チャネルを再生し、右チャネルに左チャネルの残りの部分を追加します。soundTransformObjectHalf オブジェクトは以下のパラメータを使用します。

```
var mySoundTransformObjectHalf:Object = new Object();
mySoundTransformObjectHalf.ll = 50;
mySoundTransformObjectHalf.lr = 0;
mySoundTransformObjectHalf.rr = 100;
mySoundTransformObjectHalf.rl = 50;
my_sound.setTransform(mySoundTransformObjectHalf);

var mySoundTransformObjectHalf:Object = {ll:50, lr:0, rr:100, rl:50};
```

Sound.getTransform() の例も参照してください。

関連項目

Object, getTransform（Sound.getTransform メソッド）

setVolume (Sound.setVolume メソッド)

```
public setVolume(value:Number) : Void
```

Sound オブジェクトにボリュームを設定します。

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

`value:Number` - ボリュームレベルを表す 0 ～ 100 の数値。100 はフルボリュームであり、0 はボリュームなしです。デフォルト設定は 100 です。

例

このメソッドの使用例については Sound.getVolume() を参照してください。

関連項目

setPan（Sound.setPan メソッド）, setTransform（Sound.setTransform メソッド）

Sound コンストラクタ

```
public Sound([target:Object])
```

指定されたムービークリップの新しい Sound オブジェクトを作成します。ターゲットインスタンスを指定しないと、Sound オブジェクトはムービー内のすべてのサウンドを制御します。

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

`target:Object`（オプション）- Sound オブジェクトを適用する MovieClip または Button インスタンス。

例

次の例では、`global_sound` という新しい Sound オブジェクトを作成します。2 行目で、`setVolume()` メソッドを呼び出してムービー内のすべてのサウンドのボリュームを 50% に調整します。

```
var global_sound:Sound = new Sound();
global_sound.setVolume(50);
```

次の例では、新しい Sound オブジェクトを作成し、それにターゲットムービークリップ *my_mc* を渡し、`start` メソッドを呼び出して *my_mc* 内のサウンドを開始します。

```
var movie_sound:Sound = new Sound(my_mc);
movie_sound.start();
```

start (Sound.start メソッド)

```
public start([secondOffset:Number], [loops:Number]) : Void
```

最後に割り当てたサウンドの再生を開始します。再生を開始する位置は、パラメータを指定しない場合はサウンドの先頭、指定する場合は `secondOffset` パラメータで指定したサウンド内のポイントです。

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

`secondOffset:Number`（オプション）- 特定のポイントでサウンド再生を開始するためのパラメータ。たとえば、30 秒のサウンドがあり、ちょうど中間からサウンドの再生を開始するには、`secondOffset` パラメータに 15 を指定します。サウンドは再生開始が 15 秒遅延するのでなく、15 秒の時点から開始されます。

`loops:Number`（オプション）- サウンドを連続再生する回数を指定するためのパラメータ。サウンドがストリーミングサウンドの場合、このパラメータは使用できません。

例

次の例では、新しい Sound オブジェクトを作成して、サウンドをロードします。サウンドのロードは onLoad ハンドラにより処理されます。音楽が正常にロードされた後に再生を開始できます。start() メソッドを使用して、サウンドの再生が開始します。新しい FLA ファイルを作成し、次の ActionScript を FLA ファイルまたは AS ファイルに追加します。このコード例が動作するには、"song1.mp3" という名前の MP3 ファイルを、FLA ファイルまたは AS ファイルと同じディレクトリに置く必要があります。

```
this.createTextField("status_txt", this.getNextHighestDepth(), 0,0,100,22);

// create a new Sound object
var my_sound:Sound = new Sound();
// if the sound loads, play it; if not, trace failure loading
my_sound.onLoad = function(success:Boolean) {
   if (success) {
   my_sound.start();
   status_txt.text = "Sound loaded";
   } else {
   status_txt.text = "Sound failed";
   }
};
// load the sound
my_sound.loadSound("song1.mp3", true);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

stop (Sound.stop メソッド)

stop (Sound.stop メソッド)

```
public stop([linkageID:String]) : Void
```

idName パラメータで指定されたサウンドを停止します。パラメータを指定しないと、現在再生中のすべてのサウンドを停止します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

linkageID:String (オプション) - 再生を停止する特定のサウンドを指定するパラメータ。idName パラメータは、引用符 (" ") で囲む必要があります。

例

次の例では、stop_btn および play_btn の 2 つのボタンを使用し、SWF ファイルにロードされるサウンドの再生を制御します。ドキュメントに 2 つのボタンを追加し、次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
var my_sound:Sound = new Sound();
my_sound.loadSound("song1.mp3", true);

stop_btn.onRelease = function() {
  trace("sound stopped");
  my_sound.stop();
};
play_btn.onRelease = function() {
  trace("sound started");
  my_sound.start();
};
```

関連項目

start (Sound.start メソッド)

ステージ

```
Object
  |
  +-Stage
```

```
public class Stage
extends Object
```

Stage クラスはトップレベルのクラスで、コンストラクタを実行しなくてもそのメソッド、プロパティ、およびハンドラを使用できます。このクラスのメソッドとプロパティを使用して、SWF ファイルの境界に関する情報にアクセスし、操作できます。

対応バージョン： ActionScript 1.0、Flash Player 5 - Flash Player 6 ではネイティブオブジェクトになり、パフォーマンスが大幅に向上しました。

プロパティー覧

オプション	プロパティ	説明
static	align:String	Flash Player またはブラウザ内での SWF ファイルの整列設定を示します。
static	displayState:String	ムービーをフルスクリーンモードで再生するように、または Flash Player のフルスクリーンモードを解除するように Flash Player を設定します。
static	height:Number	読み取り専用プロパティ。ステージの現在の高さをピクセル単位で示します。
static	scaleMode:String	Flash Player 内での SWF ファイルの拡大 / 縮小に関する現在の設定を示します。
static	showMenu:Boolean	Flash Player のコンテキストメニューにデフォルトの項目を表示するかどうかを指定します。
static	width:Number	読み取り専用プロパティ。ステージの現在の幅をピクセル単位で示します。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベント一覧

イベント	説明
onFullScreen = function(bFull: Boolean) {}	ムービーがフルスクリーンモードになると、あるいはフルスクリーンモードから戻ると呼び出されます。
onResize = function() {}	Stage.scaleMode が noScale に設定されているとき、SWF ファイルのサイズが変更されると呼び出されます。

メソッド一覧

オプション	署名	説明
static	addListener(listener: Object) : Void	Stage.scaleMode = "noScale" である場合に、SWF ファイルのサイズ変更を検出します。
static	removeListener(listener: Object) : Boolean	addListener() で作成したオブジェクトを削除します。

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

addListener (Stage.addListener メソッド)

public static addListener(listener:Object) : Void

Stage.scaleMode = "noScale" である場合に、SWF ファイルのサイズ変更を検出します。addListener() メソッドは、ムービークリップのデフォルトの拡大 / 縮小設定 (showAll) や、その他の拡大 / 縮小設定 (exactFit および noBorder) とは併用できません。

addListener() を使用するには、まずリスナーオブジェクトを作成する必要があります。Stage オブジェクトのリスナーオブジェクトは、Stage.onResize から通知を受け取ります。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

listener:Object - Stage.onResize イベントハンドラからのコールバック通知を待機するオブジェクト。

例

次の例では、新しいリスナーオブジェクト stageListener を作成します。次に、stageListener を使用して onResize を呼び出し、さらに onResize の起動に応じて呼び出す関数を定義します。最後に、stageListener オブジェクトを Stage オブジェクトのコールバックリストに追加します。リスナーオブジェクトを使用すると、複数のオブジェクトがサイズ変更通知を取得できます。

```
this.createTextField("stageSize_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
var stageListener:Object = new Object();
stageListener.onResize = function() {
  stageSize_txt.text = "w:"+Stage.width+", h:"+Stage.height;
};
Stage.scaleMode = "noScale";
Stage.addListener(stageListener);
```

関連項目

onResize (Stage.onResize イベントリスナー), removeListener (Stage.removeListener メソッド)

align (Stage.align プロパティ)

```
public static align : String
```

Flash Player またはブラウザ内での SWF ファイルの整列設定を示します。

次の表は、align プロパティの値の一覧です。表にない値を使用すると、SWF ファイルはデフォルトの位置である Flash Player またはブラウザ領域の中央に配置されます。

値	垂直方向	水平方向
"T"	上	中央
"B"	下	中央
"L"	中央	左
"R"	中央	右
"TL"	上	左
"TR"	上	右
"BL"	下	左
"BR"	下	右

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の例では、SWF ファイルの異なる整列を示します。ComboBox インスタンスを stageAlign_cb のインスタンス名を持つドキュメントに追加します。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
var stageAlign_cb:mx.controls.ComboBox;
stageAlign_cb.dataProvider = ['T', 'B', 'L', 'R', 'TL', 'TR', 'BL', 'BR'];
var cbListener:Object = new Object();
cbListener.change = function(evt:Object) {
   var align:String = evt.target.selectedItem;
   Stage.align = align;
};
stageAlign_cb.addEventListener("change", cbListener);
Stage.scaleMode = "noScale";
```

ComboBox から異なる整列設定を選択します。

displayState (Stage.displayState プロパティ)

```
public static displayState : String
```

ムービーをフルスクリーンモードで再生するように、または Flash Player のフルスクリーンモードを解除するように Flash Player を設定します。このプロパティは、Flash Player の現在の状態を確認するのにも使用できます。

- `fullScreen`–Flash Player のステージをユーザーの画面全体に拡大するように設定します。
- `normal`–Flash Player が標準のステージ表示モードに戻るように設定します。

フルスクリーンモードでのムービーの拡大 / 縮小ビヘイビアは、`scaleMode` の設定で決まります (この設定をするには、`Stage.scaleMode` プロパティ、SWF ファイルの `param`、または HTML ファイルの `embed` のタグ設定を使います)。Flash Player がフルスクリーンモードに移行している間に `scaleMode` プロパティを `noScale` に設定すると、ステージの `width` プロパティと `height` プロパティが更新され、`Stage.onResize` イベントリスナーが呼び出されます。

HTML ページ内で再生する SWF ファイルには次の制限が適用されます (スタンドアローンの Flash Player を使用する SWF ファイルには適用されません)。

- フルスクリーンモードを有効にするには、次の例に示すように、SWF ファイルの参照を含む HTML ページで `object` タグおよび `embed` タグに `allowFullScreen` パラメータを追加し、`"true"` に設定します。

  ```
  <param name="allowFullScreen" value="true" />
        ...
  <embed src="example.swf" allowFullScreen="true" ... >
  ```

  HTML ページでは、SWF 埋め込みタグを生成するスクリプトも使用できます。適切な `allowFullScreen` 設定を挿入するようにスクリプトを変更する必要があります。Flash オーサリングで生成された HTML ページでは、`AC_FL_RunContent()` 関数を使用して SWF ファイルの参照を埋め込みます。また、次のように、`allowNetworking` パラメータ設定を追加する必要があります。

  ```
  AC_FL_RunContent( ... "allowFullScreen", "true", ... )
  ```

- フルスクリーンモードはユーザーのマウスクリックまたはキー入力で開始します。ムービーの `Stage.displayState` はユーザー入力によってのみ変更できます。Flash Player がフルスクリーンモードになっている間は、キーボード入力を行うことができません (唯一可能なのは、フルスクリーンモードを解除するキーボードショートカットだけです)。フルスクリーンモードになると、ムービーの上に Flash Player ダイアログボックスが表示されます。そのダイアログボックスでは、現在フルスクリーンモードになっていること、また Esc キーを押すとフルスクリーンモードを終了できることが通知されます。

対応バージョン : ActionScript 2.0、Flash Player 9.0.28.0

例

次の例では、SWF をフルスクリーンモードで表示する方法を示します。上記の制限にご注意ください。Button インスタンスを myButton のインスタンス名を持つドキュメントに追加します。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
var myButton:mx.controls.Button;
myButton.label = "Toggle Fullscreen";
myButton.setSize(150,22);
var buttonListener:Object = new Object();
buttonListener.click = function(evt:Object) {
  Stage.displayState = Stage.displayState == "normal" ? "fullScreen" : "normal";
};
myButton.addEventListener("click", buttonListener);
```

関連項目

scaleMode (Stage.scaleMode プロパティ),onFullScreen (Stage.onFullScreen ハンドラ),onResize (Stage.onResize イベントリスナー)

height (Stage.height プロパティ)

```
public static height : Number
```

読み取り専用プロパティ。ステージの現在の高さをピクセル単位で示します。Stage.scaleMode の値が noScale である場合、height プロパティは Flash Player の高さを表します。Stage.scaleMode の値が noScale 以外の場合、height プロパティは SWF ファイルの高さを表します。

対応バージョン: ActionScript 1.0、Flash Player 6

例

次の例では、新しいリスナーオブジェクト stageListener を作成します。次に、myListener を使用して onResize を呼び出し、さらに onResize の起動に応じて呼び出す関数を定義します。最後に、myListener オブジェクトを Stage オブジェクトのコールバックリストに追加します。リスナーオブジェクトを使用すると、複数のオブジェクトがサイズ変更通知を取得できます。

```
this.createTextField("stageSize_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
var stageListener:Object = new Object();
stageListener.onResize = function() {
  stageSize_txt.text = "w:"+Stage.width+", h:"+Stage.height;
};
Stage.scaleMode = "noScale";
Stage.addListener(stageListener);
```

関連項目

align (Stage.align プロパティ), scaleMode (Stage.scaleMode プロパティ), width (Stage.width プロパティ)

onFullScreen (Stage.onFullScreen ハンドラ)

```
onFullScreen = function(bFull:Boolean) {}
```

ムービーがフルスクリーンモードになると、あるいはフルスクリーンモードから戻ると呼び出されます。SWF ファイルのフルスクリーンの状態は、このハンドラーか、または Stage.displayState プロパティを使って確認できます。フルスクリーンモードを終了するには、ActionScript を使用するか、ユーザーがキーボードショートカットを呼び出すか、またはフルスクリーンウィンドウからフォーカスを動かします。

対応バージョン： ActionScript 2.0、Flash Player

パラメータ

bFull:Boolean -

関連項目

displayState (Stage.displayState プロパティ), addListener (Stage.addListener メソッド), removeListener (Stage.removeListener メソッド)

onResize (Stage.onResize イベントリスナー)

```
onResize = function() {}
```

Stage.scaleMode が noScale に設定されているとき、SWF ファイルのサイズが変更されると呼び出されます。このイベントハンドラを使用して、SWF ファイルのサイズ変更に応じてステージ上のオブジェクトを配置する関数を記述できます。

```
myListener.onResize = function()[
 // your statements here
 }
```

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の例では、ステージのサイズが変更されたときに、[出力] パネルにメッセージを表示します。

```
Stage.scaleMode = "noScale"
var myListener:Object = new Object();
myListener.onResize = function () {
  trace("Stage size is now " + Stage.width + " by " + Stage.height);
}
Stage.addListener(myListener);
// later, call Stage.removeListener(myListener)
```

関連項目

scaleMode (Stage.scaleMode プロパティ), addListener (Stage.addListener メソッド), removeListener (Stage.removeListener メソッド)

removeListener (Stage.removeListener メソッド)

```
public static removeListener(listener:Object) : Boolean
```

addListener() で作成したオブジェクトを削除します。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

`listener:Object` - addListener() を使用してオブジェクトのコールバックに追加されたオブジェクト。

戻り値

`Boolean` - ブール値。

例

次の例では、ダイナミックに作成されたテキストフィールドに、ステージのサイズを表示します。ステージのサイズを変更すると、テキストフィールドの値も更新されます。remove_btn というインスタンス名のボタンを作成します。タイムラインのフレーム 1 に次の ActionScript を追加します。

```
this.createTextField("stageSize_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
stageSize_txt.autoSize = true;
stageSize_txt.border = true;
var stageListener:Object = new Object();
stageListener.onResize = function() {
  stageSize_txt.text = "w:"+Stage.width+", h:"+Stage.height;
};
Stage.addListener(stageListener);
```

```
remove_btn.onRelease = function() {
  stageSize_txt.text = "Removing Stage listener...";
  Stage.removeListener(stageListener);
}
```

[制御]-[ムービープレビュー] を選択してこの例をテストします。テキストフィールドに表示される値は、テスト環境のサイズを変更すると更新されます。remove_btn をクリックすると、リスナーが削除され、テキストフィールドの値は更新されなくなります。

関連項目

addListener (Stage.addListener メソッド)

scaleMode (Stage.scaleMode プロパティ)

```
public static scaleMode : String
```

Flash Player 内での SWF ファイルの拡大 / 縮小に関する現在の設定を示します。scaleMode プロパティは、SWF ファイルに特定の拡大 / 縮小モードを適用します。デフォルトでは、[パブリッシュ設定] ダイアログボックスに設定されている HTML パラメータが SWF ファイルに適用されます。

scaleMode プロパティには、"exactFit"、"showAll"、"noBorder"、"noScale" の各値を使用することができます。他の値を使用すると、scaleMode プロパティはデフォルトの "showAll" に設定されます。

- デフォルトの showAll を指定すると、指定された領域内に Flash コンテンツ全体が元の縦横比を維持したまま、歪まずに表示されます。ただし、アプリケーションの両側に境界枠が表示されることがあります。
- noBorder を指定すると、指定された領域いっぱいに Flash コンテンツがサイズ調整されて、歪まずに表示されます。ただし、アプリケーションの元の縦横比を保つために、ある程度トリミングされることがあります。
- exactFit を指定すると、指定された領域にちょうど収まるように Flash コンテンツ全体が表示されますが、元の縦横比は保たれません。歪みが発生する場合もあります。
- noScale を指定すると、Flash コンテンツのサイズが固定され、プレーヤーウィンドウのサイズが変更された場合でも、サイズが維持されます。プレーヤーウィンドウが Flash コンテンツよりも小さい場合は、トリミングされることがあります。

メモ : デフォルト設定は showAll ですが、ムービープレビューモードでは、noScale がデフォルト設定になります。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、SWF ファイルのさまざまな拡大 / 縮小設定を示します。ComboBox インスタンスを scaleMode_cb のインスタンス名を持つドキュメントに追加します。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
var scaleMode_cb:mx.controls.ComboBox;
scaleMode_cb.dataProvider = ["showAll", "exactFit", "noBorder", "noScale"];
var cbListener:Object = new Object();
cbListener.change = function(evt:Object) {
  var scaleMode_str:String = evt.target.selectedItem;
  Stage.scaleMode = scaleMode_str;
};
scaleMode_cb.addEventListener("change", cbListener);
```

別の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/StageSize" フォルダに移動して stagesize.fla ファイルにアクセスします。

showMenu (Stage.showMenu プロパティ)

```
public static showMenu : Boolean
```

Flash Player のコンテキストメニューにデフォルトの項目を表示するかどうかを指定します。showMenu を true (デフォルト) に設定すると、すべてのコンテキストメニュー項目が表示されます。showMenu を false に設定すると、メニュー項目として [設定] および [Flash Player について] だけが表示されます。

対応バージョン: ActionScript 1.0、Flash Player 6

例

次の例では、Flash Player のコンテキストメニューを有効および無効にするクリック可能なテキストリンクを作成します。

```
this.createTextField("showMenu_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
showMenu_txt.html = true;
showMenu_txt.autoSize = true;
showMenu_txt.htmlText = "<a href=\"asfunction:toggleMenu\"><u>Stage.showMenu =
  "+Stage.showMenu+"</u></a>";
function toggleMenu() {
  Stage.showMenu = !Stage.showMenu;
  showMenu_txt.htmlText = "<a href=\"asfunction:toggleMenu\"><u>Stage.showMenu =
  "+Stage.showMenu+"</u></a>";
}
```

関連項目

ContextMenu, ContextMenuItem

width (Stage.width プロパティ)

```
public static width : Number
```

読み取り専用プロパティ。ステージの現在の幅をピクセル単位で示します。Stage.scaleMode の値が "noScale" である場合、width プロパティは Flash Player の幅を表します。つまり、Flash Player のウィンドウのサイズを変更すると、それに合わせて Stage.width が変化します。Stage.scaleMode の値が "noScale" 以外の場合、width はオーサリング時に [ドキュメントプロパティ] ダイアログボックスで設定した SWF ファイルの幅を表します。つまり、Flash Player のウィンドウのサイズを変更しても width の値は一定になります。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、新しいリスナーオブジェクト stageListener を作成します。次に、stageListener を使用して onResize を呼び出し、さらに onResize の起動に応じて呼び出す関数を定義します。最後に、stageListener オブジェクトを Stage オブジェクトのコールバックリストに追加します。リスナーオブジェクトを使用すると、複数のオブジェクトがサイズ変更通知を取得できます。

```
this.createTextField("stageSize_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
var stageListener:Object = new Object();
stageListener.onResize = function() {
  stageSize_txt.text = "w:"+Stage.width+", h:"+Stage.height;
};
Stage.scaleMode = "noScale";
Stage.addListener(stageListener);
```

関連項目

align (Stage.align プロパティ), height (Stage.height プロパティ), scaleMode (Stage.scaleMode プロパティ)

String

```
Object
  |
  +-String

public class String
extends Object
```

String クラスは、ストリングプリミティブデータ型のラッパーです。このメソッドとプロパティを使用して、プリミティブストリング値の型を操作できます。String() 関数を使用して、任意のオブジェクトの値をストリングに変換できます。

concat()、fromCharCode()、slice()、substr() を除く String クラスのすべてのメソッドは汎用メソッドです。つまり、メソッドが toString() を呼び出した後で、メソッドの操作が実行されます。これらのメソッドは String オブジェクト以外のオブジェクトでも使用できます。

すべてのストリングインデックスはゼロから始まるため、各ストリング x の最終文字のインデックスは x.length - 1 のようになります。

String クラスのメソッドを呼び出すには、コンストラクタメソッド new String を使用するか、ストリングリテラル値を使用します。ストリングリテラルを指定すると、ActionScript インタプリタはそれをテンポラリ String オブジェクトに自動変換し、その後、テンポラリ String オブジェクトを破棄します。ストリングリテラルで String.length プロパティを使用することもできます。

ストリングリテラルと String オブジェクトを混同しないように注意してください。次の例では、コードの 1 行目でストリングリテラル first_string を作成し、2 行目で String オブジェクト second_string を作成します。

```
var first_string:String = "foo"
var second_string:String = new String("foo")
```

String オブジェクトを特に使用する必要がない限り、ストリングリテラルを使用してください。

対応バージョン： ActionScript 1.0、Flash Player 5 - Flash Player 6 ではネイティブオブジェクトになり、パフォーマンスが大幅に向上しました。

プロパティー覧

オプション	プロパティ	説明
	length:Number	指定した String オブジェクト内にある文字数を表す整数です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター覧

署名	説明
String(value:String)	新しい String オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	charAt(index:Number) : String	パラメータ index で指定された位置にある文字を返します。
	charCodeAt(index:Number) : Number	16 で指定された文字を表す 0 ～ 65535 の 16 ビット整数を返します。
	concat(value:Object) : String	String オブジェクトの値とパラメータを連結し、新しく形成したストリングを返します。元の値 my_str は変更されません。
static	fromCharCode() : String	パラメータ内の Unicode 値に対応する文字をストリングとして返します。
	indexOf(value:String, [startIndex:Number]) : Number	ストリング内を検索し、ストリング内の startIndex 以降の位置で見つかった最初の value の位置を返します。
	lastIndexOf(value:String, [startIndex:Number]) : Number	ストリングを右から左へと探し、ストリング内で startIndex の前に見つかった最後の value のインデックスを返します。
	slice(start:Number, end:Number) : String	返されるストリングには、start 文字から end 文字の前までのすべての文字が含まれます。
	split(delimiter:String, [limit:Number]) : Array	指定された delimiter パラメータがある各位置で String オブジェクトをサブストリングに分割し、そのサブストリングを配列として返します。
	substr(start:Number, length:Number) : String	ストリング内で start パラメータで指定されたインデックスから length パラメータで指定された文字数までの文字を返します。
	substring(start:Number, end:Number) : String	start パラメータと end パラメータで指定された点の間の文字をストリングとして返します。
	toLowerCase() : String	このストリングのコピーを返します。すべての大文字が小文字に変換されます。
	toString() : String	プロパティがストリングかどうかに関係なく、オブジェクトのプロパティをストリングとして返します。
	toUpperCase() : String	このストリングのコピーを返します。すべての小文字が大文字に変換されます。
	valueOf() : String	String インスタンスのプリミティブ値を返します。

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

charAt (String.charAt メソッド)

public charAt(index:Number) : String

パラメータ index で指定された位置にある文字を返します。index に指定した値が 0 ～ (string.length - 1) の範囲内でない場合は、空のストリングを返します。

このメソッドは String.charCodeAt() に似ていますが、16 ビット整数の文字コードではなく文字が返される点が異なります。

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

index:Number - ストリング内の文字の位置を指定する整数。最初の文字の位置は 0 で、最後の文字の位置は my_str .length-1 です。

戻り値

String - 指定されたインデックス位置の文字。指定されたインデックスがこの String のインデックスの範囲外である場合は空の String が返されます。

例

次の例では、ストリング "Chris" の最初の文字に対してこのメソッドを呼び出します。

```
var my_str:String = "Chris";
var firstChar_str:String = my_str.charAt(0);
trace(firstChar_str); // output: C
```

関連項目

charCodeAt (String.charCodeAt メソッド)

charCodeAt (String.charCodeAt メソッド)

```
public charCodeAt(index:Number) : Number
```

16 で指定された文字を表す 0 ～ 65535 の 16 ビット整数を返します。index に指定した値が 0 ～ (string.length - 1) の範囲内でない場合は、NaN を返します。

このメソッドは、String.charAt() と似ていますが、文字ではなく 16 ビット整数文字コードを返す点が異なります。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

index:Number - ストリング内の文字の位置を指定する整数。最初の文字の位置は 0 で、最後の文字の位置は my_str.length -1 です。

戻り値

Number - index で指定された文字を表す整数。

例

次の例では、ストリング "Chris" の最初の文字に対してこのメソッドを呼び出します。

```
var my_str:String = "Chris";
var firstChar_num:Number = my_str.charCodeAt(0);
trace(firstChar_num); // output: 67
```

関連項目

charAt (String.charAt メソッド)

concat (String.concat メソッド)

```
public concat(value:Object) : String
```

String オブジェクトの値とパラメータを連結し、新しく形成したストリングを返します。元の値 my_str は変更されません。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

value:Object - value1[,...valueN] 連結される値。

戻り値

String - ストリング。

例

次の例では、ストリングを 2 つ作成し、String.concat() を使用して各ストリングを連結します。

```
var stringA:String = "Hello";
var stringB:String = "World";
var combinedAB:String = stringA.concat(" ", stringB);
trace(combinedAB); // output: Hello World
```

fromCharCode (String.fromCharCode メソッド)

```
public static fromCharCode() : String
```

パラメータ内の Unicode 値に対応する文字をストリングとして返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

String - 指定された Unicode 文字コードのストリング値。

例

次の例では、fromCharCode() メソッドを使用して、電子メールアドレスに @ 文字を挿入します。

```
var address_str:String = "dog"+String.fromCharCode(64)+"house.net";
trace(address_str); // output: dog@house.net
```

indexOf (String.indexOf メソッド)

```
public indexOf(value:String, [startIndex:Number]) : Number
```

ストリング内を検索し、ストリング内の startIndex 以降の位置で見つかった最初の value の位置を返します。このインデックスはゼロから始まります。つまりストリングの最初の文字は、インデックス 1 ではなくインデックス 0 にあると見なされます。value が見つからない場合、メソッドは -1 を返します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

value:String - ストリング。検索対象のサブストリングです。

startIndex:Number (オプション) - 検索の開始インデックスを指定する整数

戻り値

Number - 最初に見つかった指定のサブストリングの位置。見つからなかった場合は -1 を返します。

例

次の例では、indexOf() を使用して文字とサブストリングのインデックスを返します。

```
var searchString:String = "Lorem ipsum dolor sit amet.";
var index:Number;

index = searchString.indexOf("L");
trace(index); // output: 0

index = searchString.indexOf("l");
trace(index); // output: 14

index = searchString.indexOf("i");
trace(index); // output: 6

index = searchString.indexOf("ipsum");
trace(index); // output: 6

index = searchString.indexOf("i", 7);
trace(index); // output: 19

index = searchString.indexOf("z");
trace(index); // output: -1
```

関連項目

lastIndexOf (String.lastIndexOf メソッド)

lastIndexOf (String.lastIndexOf メソッド)

```
public lastIndexOf(value:String, [startIndex:Number]) : Number
```

ストリングを右から左へと探し、ストリング内で startIndex の前に見つかった最後の value のインデックスを返します。このインデックスはゼロから始まります。つまりストリングの最初の文字は、インデックス 1 ではなくインデックス 0 にあると見なされます。value が見つからない場合、メソッドは -1 を返します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

value:String - 検査対象のストリング。

startIndex:Number (オプション) - value を検索する開始位置を指定する整数。

戻り値

Number - 最後に見つかった指定のサブストリングの位置。見つからなかった場合は -1 を返します。

例

次の例では、lastIndexOf() を使用して特定の文字のインデックスを返す方法を示します。

```
var searchString:String = "Lorem ipsum dolor sit amet.";
var index:Number;

index = searchString.lastIndexOf("L");
trace(index); // output: 0

index = searchString.lastIndexOf("l");
trace(index); // output: 14

index = searchString.lastIndexOf("i");
trace(index); // output: 19

index = searchString.lastIndexOf("ipsum");
trace(index); // output: 6

index = searchString.lastIndexOf("i", 18);
trace(index); // output: 6

index = searchString.lastIndexOf("z");
trace(index); // output: -1
```

関連項目

indexOf (String.indexOf メソッド)

length (String.length プロパティ)

public length : Number

指定した String オブジェクト内にある文字数を表す整数です。

すべてのストリングインデックスはゼロから始まるため、各ストリング x の最終文字のインデックスは x.length - 1 のようになります。

対応バージョン: ActionScript 1.0、Flash Player 5

例

次の例では、新しい String オブジェクトを作成し、String.length を使用して文字数をカウントします。

```
var my_str:String = "Hello world!";
trace(my_str.length); // output: 12
```

次の例では、0 から my_str.length までループします。このコードでは、ストリング内の文字をチェックします。ストリングに @ 文字が含まれる場合には、[出力] パネルに true が表示されます。@ 文字がない場合には、[出力] パネルに false が表示されます。

```
function checkAtSymbol(my_str:String):Boolean {
  for (var i = 0; i<my_str.length; i++) {
  if (my_str.charAt(i) == "@") {
    return true;
  }
  }
  return false;
}

trace(checkAtSymbol("dog@house.net")); // output: true
trace(checkAtSymbol("Chris")); // output: false
```

別の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Strings" フォルダに移動して Strings.fla ファイルにアクセスします。

slice (String.slice メソッド)

```
public slice(start:Number, end:Number) : String
```

返されるストリングには、start 文字から end 文字の前までのすべての文字が含まれます。元の String オブジェクトは変更されません。end パラメータを指定しないと、サブストリングの終わりはストリングの終わりです。start で指定されたインデックスの文字が、end で指定されたインデックス文字と同じか、その右側にある場合、メソッドは空の文字列を返します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

start:Number - スライスの始点のゼロから始まるインデックス。start が負の数値の場合、始点はストリングの終わりから決定されます。このとき、-1 が最後の文字になります。

end:Number - スライスの終点のインデックスより大きい整数値。end パラメータで指定されたインデックス位置の文字は、取り出されるストリングには含まれません。このパラメータを省略すると、String.length が使用されます。end が負の数値の場合、終点はストリングの終わりからカウントされて決定されます。このとき、-1 が最後の文字になります。

戻り値

String - 指定されたストリングのサブストリング。

例

次の例では、変数 my_str , を作成し、それに String 値を割り当てた後、start パラメータと end パラメータにさまざまな値を使用して slice( ) メソッドを呼び出します。slice( ) の各呼び出しは、[出力] パネルに出力を表示する trace( ) ステートメントでラップされています。

```
// Index values for the string literal
// positive index: 0 1 2 3 4
// string: L o r e m
// negative index: -5 -4 -3 -2 -1

var my_str:String = "Lorem";

// slice the first character
trace("slice(0,1): "+my_str.slice(0, 1)); // output: slice(0,1): L
trace("slice(-5,1): "+my_str.slice(-5, 1)); // output: slice(-5,1): L

// slice the middle three characters
trace("slice(1,4): "+my_str.slice(1, 4)); // slice(1,4): ore
trace("slice(1,-1): "+my_str.slice(1, -1)); // slice(1,-1): ore

// slices that return empty strings because start is not to the left of end
trace("slice(1,1): "+my_str.slice(1, 1)); // slice(1,1):
trace("slice(3,2): "+my_str.slice(3, 2)); // slice(3,2):
trace("slice(-2,2): "+my_str.slice(-2, 2)); // slice(-2,2):

// slices that omit the end parameter use String.length, which equals 5
trace("slice(0): "+my_str.slice(0)); // slice(0): Lorem
trace("slice(3): "+my_str.slice(3)); // slice(3): em
```

別の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Strings" フォルダに移動して Strings.fla ファイルにアクセスします。

関連項目

substr (String.substr メソッド), substring (String.substring メソッド)

split (String.split メソッド)

```
public split(delimiter:String, [limit:Number]) : Array
```

指定された delimiter パラメータがある各位置で String オブジェクトをサブストリングに分割し、そのサブストリングを配列として返します。区切り記号として空のストリング ("") を使用すると、ストリング内の各文字がエレメントとして配列に挿入されます。

delimiter パラメータが未定義の場合は、ストリング全体が返される配列の最初のエレメントに挿入されます。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

delimiter: String - ストリング。my_str を分割する文字またはストリングです。

limit: Number (オプション) - 配列に挿入する項目数。

戻り値

Array - my_str のサブストリングを含む配列。

例

次の例では、5 つのエレメントで構成される配列を返します。

```
var my_str:String = "P,A,T,S,Y";
var my_array:Array = my_str.split(",");
for (var i = 0; i<my_array.length; i++) {
  trace(my_array[i]);
}
// output:
  P
  A
  T
  S
  Y
```

次の例では、2 つのエレメント "P" および "A" で構成される配列を返します。

```
var my_str:String = "P,A,T,S,Y";
var my_array:Array = my_str.split(",", 2);
trace(my_array); // output: P,A
```

次の例に示すように、delimiter パラメータに空のストリング ("") を使用すると、ストリング内の各文字がエレメントとして配列に挿入されます。

```
var my_str:String = new String("Joe");
var my_array:Array = my_str.split("");
for (var i = 0; i<my_array.length; i++) {
  trace(my_array[i]);
}
// output:
  J
  o
  e
```

別の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Strings" フォルダに移動して Strings.fla ファイルにアクセスします。

関連項目

join (Array.join メソッド)

String コンストラクタ

```
public String(value:String)
```

新しい String オブジェクトを作成します。

メモ : String オブジェクトよりもストリングリテラルを使用する方が負荷が少なく、一般的により簡単に使用できます。String オブジェクトを使用することに特別な理由がない場合以外は、String クラスのコンストラクタよりもストリングリテラルを使用することをお勧めします。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`value:String` - 新しい String オブジェクトの初期値。

substr (String.substr メソッド)

```
public substr(start:Number, length:Number) : String
```

ストリング内で `start` パラメータで指定されたインデックスから `length` パラメータで指定された文字数までの文字を返します。`substr` メソッドは、`my_str` に指定されたストリングを変更せずに、新しいストリングを返します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`start:Number` - サブストリングを作成するために使用する `my_str` 内の開始位置を示す整数。`start` が負の数値の場合、開始位置はストリングの終わりから決定されます。このとき、-1 が最後の文字です。

`length:Number` - 作成するサブストリングの文字数。`length` を指定しないと、サブストリングにはストリングの始めから終わりまでのすべての文字が含まれます。

戻り値

`String` - 指定されたストリングのサブストリング。

例

次の例では、新しいストリング my_str を作成し、substr() を使用してストリングの 2 つ目の文字を返します。まず正の start パラメータを使用し、次に負の start パラメータを使用します。

```
var my_str:String = new String("Hello world");
var mySubstring:String = new String();
mySubstring = my_str.substr(6,5);
trace(mySubstring); // output: world

mySubstring = my_str.substr(-5,5);
trace(mySubstring); // output: world
```

別の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Strings" フォルダに移動して Strings.fla ファイルにアクセスします。

substring (String.substring メソッド)

```
public substring(start:Number, end:Number) : String
```

start パラメータと end パラメータで指定された点の間の文字をストリングとして返します。end パラメータを指定しないと、サブストリングの終わりはストリングの終わりです。start が end と等しい場合、空白のストリングが返されます。start の値が end の値を超えると、2 つのパラメータは入れ替えられて関数が実行されます。元の値は変更されません。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

start:Number - サブストリングを作成するために使用する my_str の開始位置を示す整数。start パラメータに指定できる値は、0 ～ String.length - 1 です。start パラメータに負数を指定した場合は、0 が使われます。

end:Number - my_str から取り出す最後の文字のインデックスに 1 を加えた整数。end に指定できる値は、1 ～ String.length です。end パラメータで指定されたインデックス位置の文字は、取り出されるストリングには含まれません。このパラメータを省略すると、String.length が使用されます。このパラメータが負の値である場合は、0 が使用されます。

戻り値

String - 指定されたストリングのサブストリング。

例

次の例では、substring() の使用方法を示します。

```
var my_str:String = "Hello world";
var mySubstring:String = my_str.substring(6,11);
trace(mySubstring); // output: world
```

次に、負の start パラメータを使用した場合の例を示します。

```
var my_str:String = "Hello world";
var mySubstring:String = my_str.substring(-5,5);
trace(mySubstring); // output: Hello
```

別の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Strings" フォルダに移動して Strings.fla ファイルにアクセスします。

toLowerCase (String.toLowerCase メソッド)

```
public toLowerCase() : String
```

このストリングのコピーを返します。すべての大文字が小文字に変換されます。元のストリングは変更されません。

このメソッドは、対応する Unicode の小文字が存在するすべての文字 (単に A ～ Z ではない) を変換します。これらの大文字と小文字のマッピングは、Unicode Character Database で定義されているとおりに、UnicodeData.txt ファイルと SpecialCasings.txt ファイルで定義されています。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

String - ストリング。

例

次の例では、すべての文字が大文字になっているストリングを作成した後、toLowerCase() を使用して、そのストリングの大文字をすべて小文字に変換したコピーを作成します。

```
var upperCase:String = "LOREM IPSUM DOLOR";
var lowerCase:String = upperCase.toLowerCase();
trace("upperCase: " + upperCase); // output: upperCase: LOREM IPSUM DOLOR
trace("lowerCase: " + lowerCase); // output: lowerCase: lorem ipsum dolor
```

別の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Strings" フォルダに移動して Strings.fla ファイルにアクセスします。

関連項目

toUpperCase (String.toUpperCase メソッド)

toString (String.toString メソッド)

```
public toString() : String
```

プロパティがストリングかどうかに関係なく、オブジェクトのプロパティをストリングとして返します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

`String` - ストリング。

例

次の例では、プロパティがストリングであるかどうかに関係なく、オブジェクトのプロパティをすべてリストする大文字のストリングを出力します。

```
var employee:Object = new Object();
employee.name = "bob";
employee.salary = 60000;
employee.id = 284759021;

var employeeData:String = new String();
for (prop in employee)
{
  employeeData += employee[prop].toString().toUpperCase() + " ";
}
trace(employeeData);
```

`toString()` メソッドがこのコードに含まれず、`for` ループ内の行で `employee[prop].toUpperCase()` を使用した場合、出力は "`undefined undefined BOB`" になります。`toString()` メソッドを指定すると、目的の出力 "`284759021 60000 BOB`" が生成されます。

toUpperCase (String.toUpperCase メソッド)

```
public toUpperCase() : String
```

このストリングのコピーを返します。すべての小文字が大文字に変換されます。元のストリングは変更されません。

このメソッドは、対応する Unicode の大文字が存在するすべての文字 (単に a ～ z ではない) を変換します。これらの大文字と小文字のマッピングは、Unicode Character Database で定義されているとおりに、UnicodeData.txt ファイルと SpecialCasings.txt ファイルで定義されています。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

`String` - ストリング。

例

次の例では、すべての文字が小文字になっているストリングを作成した後、toUpperCase() を使用して、そのストリングのコピーを作成します。

```
var lowerCase:String = "lorem ipsum dolor";
var upperCase:String = lowerCase.toUpperCase();
trace("lowerCase: " + lowerCase); // output: lowerCase: lorem ipsum dolor
trace("upperCase: " + upperCase); // output: upperCase: LOREM IPSUM DOLOR
```

別の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Strings" フォルダに移動して Strings.fla ファイルにアクセスします。

関連項目

toLowerCase (String.toLowerCase メソッド)

valueOf (String.valueOf メソッド)

```
public valueOf() : String
```

String インスタンスのプリミティブ値を返します。このメソッドは、String オブジェクトをプリミティブなストリング値に変換するように設計されています。Flash Player は必要時に自動的に valueOf() を呼び出すようになっているため、このメソッドを明示的に呼び出すことが必要となることはほとんどありません。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

String - ストリングの値。

例

次の例では、String クラスの新しいインスタンスを作成し、valueOf メソッドが 新規インスタンスへの参照ではなく *primitive* 値を返すことを示します。

```
var str:String = new String("Hello World");
var value:String = str.valueOf();
trace(str instanceof String); // true
trace(value instanceof String); // false
trace(str === value); // false
```

StyleSheet (TextField.StyleSheet)

```
Object
  |
  +-TextField.StyleSheet

public class StyleSheet
extends Object
```

StyleSheet クラスを使用すると、フォントのサイズや色、その他のスタイルなどのテキストフォーマット規則を含む StyleSheet オブジェクトを作成することができます。スタイルシートによって定義したスタイルは、HTML 形式または XML 形式のテキストを含む TextField オブジェクトに適用することができます。TextField オブジェクトに含まれるテキストは、StyleSheet オブジェクトによって定義されるタグスタイルに従って、自動的にフォーマットされます。テキストスタイルを使用して、新しいフォーマットタグの定義、ビルトイン HTML タグの再定義、特定の HTML タグに適用できるスタイルクラスの作成を行うことができます。

スタイルを TextField オブジェクトに適用するには、StyleSheet オブジェクトを TextField オブジェクトの `styleSheet` プロパティに適用します。

メモ： このオブジェクトに適用するスタイルシートを使用するテキストフィールドは編集不可になっています。つまり、`type` プロパティが `"input"` に設定されているテキストフィールドは、スタイルシートのフォーマットがテキストフィールドのデフォルトテキストに適用されますが、ユーザーはその内容を編集できなくなります。TextFormat クラスを使ってスタイルをテキスト入力フィールドに割り当てることを検討してください。

Flash Player は、オリジナルの CSS1 仕様 (www.w3.org/TR/REC-CSS1) にあるプロパティのサブセットをサポートしています。次の表に、サポートされている CSS (カスケーディングスタイルシート) プロパティとその値、および対応する ActionScript プロパティの名前を示します。ActionScript プロパティ名は、どれも対応する CSS プロパティ名から派生しており、名前にハイフンが含まれる場合はハイフンを省略し、後ろの文字を大文字にしています。

CSS プロパティ	ActionScript プロパティ	用途とサポートされる値
`color`	`color`	16 進数のカラー値のみがサポートされます。`blue` などの名前付きカラーはサポートされません。カラーは、`#FF0000` のようなフォーマットで記述されます。
`display`	`display`	サポートされる値は `inline`、`block`、および `none` です。

CSS プロパティ	ActionScript プロパティ	用途とサポートされる値
font-family	fontFamily	使用するフォントをカンマ区切りリストで指定します。優先度の高い順に並べます。任意のフォントファミリ名を使用できます。汎用フォント名を指定した場合、適切なデバイスフォントに置換されます。次のようなフォント変換が行われます。mono は _typewriter に、sans-serif は _sans に、serif は _serif にそれぞれ変換されます。
font-size	fontSize	値の数字の部分だけを使用します。単位 (px、pt) は解析されません。ピクセルとポイントは同じ意味になります。
font-style	fontStyle	有効な値は normal と italic です。
font-weight	fontWeight	有効な値は normal と bold です。
kerning	kerning	有効な値は true と false です。カーニングは、埋め込みフォントに対してのみサポートされています。Courier New など特定のフォントでは、カーニングがサポートされていません。kerning プロパティは、Macintosh で作成された SWF ファイルではなく、Windows で作成された SWF ファイルでのみサポートされます。ただし、これらの SWF ファイルは Windows 以外のバージョンの Flash Player で表示することが可能であり、カーニングも適用されます。
leading	leading	行間に均等に配分されるスペースの量です。この値は、各行の後に追加されるピクセル数を示します。負の値を指定すると、行の間隔が狭くなります。値の数字の部分だけを使用します。単位 (px、pt) は解析されません。ピクセルとポイントは同じ意味になります。Flash Player 8 以降で使用できるもの。
letter-spacing	letterSpacing	文字間に均等に配分されるスペースの量です。この値は、各文字の後の送りに追加されるピクセル数を示します。負の値を指定すると、文字の間隔が狭くなります。値の数字の部分だけを使用します。単位 (px、pt) は解析されません。ピクセルとポイントは同じ意味になります。

CSS プロパティ	ActionScript プロパティ	用途とサポートされる値
margin-left	marginLeft	値の数字の部分だけを使用します。単位 (px、pt) は解析されません。ピクセルとポイントは同じ意味になります。
margin-right	marginRight	値の数字の部分だけを使用します。単位 (px、pt) は解析されません。ピクセルとポイントは同じ意味になります。
text-align	textAlign	有効な値は left、center、right、および justify です。
text-decoration	textDecoration	有効な値は none と underline です。
text-indent	textIndent	値の数字の部分だけを使用します。単位 (px、pt) は解析されません。ピクセルとポイントは同じ意味になります。

対応バージョン： ActionScript 1.0、Flash Player 7

関連項目

TextField

プロパティー覧

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベントの一覧

イベント	説明
onLoad = function(success: Boolean) {}	load() 処理が完了したときに呼び出されます。

コンストラクター覧

署名	説明
StyleSheet()	StyleSheet オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	clear() : Void	指定した StyleSheet オブジェクトのスタイルをすべて削除します。
	getStyle(name:String) : Object	指定したスタイル (name) に関連付けられているスタイルオブジェクトのコピーを返します。
	getStyleNames() : Array	このスタイルシートに登録されているすべてのスタイルの名前 (ストリング) を含む配列を返します。
	load(url:String) : Boolean	StyleSheet への CSS ファイルのロードを開始します。
	parseCSS(cssText: String) : Boolean	cssText の CSS を解析し、その内容を含む StyleSheet オブジェクトをロードします。
	setStyle(name:String, style:Object) : Void	指定された名前を使用して、新しいスタイルを StyleSheet オブジェクトに追加します。
	transform(style: Object) : TextFormat	CSS 解析機能を拡張します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

clear (StyleSheet.clear メソッド)

```
public clear() : Void
```

指定した StyleSheet オブジェクトのスタイルをすべて削除します。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例では、styles.css という StyleSheet を SWF ファイルにロードし、ロードされたスタイルを [出力] パネルに表示します。clear_btn をクリックすると、my_styleSheet オブジェクトからすべてのスタイルが削除されます。

```
// Create a new StyleSheet object
import TextField.StyleSheet;
var my_styleSheet:StyleSheet = new StyleSheet();

my_styleSheet.onLoad = function(success:Boolean) {
  if (success) {
    trace("Styles loaded.");
    var styles_array:Array = my_styleSheet.getStyleNames();
    for (var i = 0; I < styles_array.length; i++) {
      trace("\t"+styles_array[i]);
    }
    trace("");
  } else {
    trace("Error loading CSS");
  }
};

// Start the loading operation
my_styleSheet.load("styles.css");

clear_btn.onRelease = function() {
  my_styleSheet.clear();
  trace("Styles cleared.");
  var styles_array:Array = my_styleSheet.getStyleNames();
  for (var i = 0; i<styles_array.length; i++) {
    trace("\t"+styles_array[i]);
  }
  trace("");
};
```

getStyle (StyleSheet.getStyle メソッド)

```
public getStyle(name:String) : Object
```

指定したスタイル (name) に関連付けられているスタイルオブジェクトのコピーを返します。name にスタイルオブジェクトが関連付けられていない場合は、null を返します。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

`name:String` - 取得するスタイルの名前。

戻り値

Object - スタイルオブジェクト。スタイルオブジェクトがない場合は null を返します。

例

次の例では、CSS ファイルからスタイルをロードして StyleSheet を解析し、スタイル名とプロパティの値を [出力] パネルに表示します。StyleSheetTracer.as という名前で ActionScript ファイルを作成し、次のコードをファイルに入力します。

```
import TextField.StyleSheet;
class StyleSheetTracer {
  // StyleSheetTracer.displayFromURL
  // This method displays the CSS style sheet at
  // the specified URL in the Output panel.
  static function displayFromURL(url:String):Void {
    // Create a new StyleSheet object
    var my_styleSheet:StyleSheet = new StyleSheet();
    // The load operation is asynchronous, so set up
    // a callback function to display the loaded StyleSheet.
    my_styleSheet.onLoad = function(success:Boolean) {
      if (success) {
        StyleSheetTracer.display(this);
      } else {
        trace("Error loading style sheet "+url);
      }
    };
    // Start the loading operation.
    my_styleSheet.load(url);
  }
  static function display(my_styleSheet:StyleSheet):Void {
    var styleNames:Array = my_styleSheet.getStyleNames();
    if (!styleNames.length) {
      trace("This is an empty style sheet.");
    } else {
      for (var i = 0; i<styleNames.length; i++) {
        var styleName:String = styleNames[i];
        trace("Style "+styleName+":");
        var styleObject:Object = my_styleSheet.getStyle(styleName);
        for (var propName in styleObject) {
          var propValue = styleObject[propName];
          trace("\t"+propName+": "+propValue);
        }
        trace("");
      }
    }
  }
}
```

styles.css という名前で CSS ドキュメントを作成します。この CSS には .heading と .mainBody という 2 つのスタイルが含まれており、font-family、font-size、および font-weight の各プロパティが定義されます。CSS ドキュメントに次のコードを入力します。

```
.heading {
font-family: Arial, Helvetica, sans-serif;
font-size: 24px;
font-weight: bold;
}
.mainBody {
font-family: Arial, Helvetica, sans-serif;
font-size: 12px;
font-weight: normal;
}
```

最後に、FLA ファイルまたは ActionScript ファイルに、外部スタイルシート "styles.css" をロードする ActionScript を次のように入力します。

```
StyleSheetTracer.displayFromURL("styles.css");
```

この場合、[出力] パネルには次の情報が表示されます。

```
Style .heading:
fontWeight: bold
fontSize: 24px
fontFamily: Arial, Helvetica, sans-serif

Style .mainBody:
fontWeight: normal
fontSize: 12px
fontFamily: Arial, Helvetica, sans-serif
```

関連項目

setStyle (StyleSheet.setStyle メソッド),getStyleNames (StyleSheet.getStyleNames メソッド)

getStyleNames (StyleSheet.getStyleNames メソッド)

```
public getStyleNames() : Array
```

このスタイルシートに登録されているすべてのスタイルの名前 (ストリング) を含む配列を返します。

対応バージョン : ActionScript 1.0、Flash Player 7

戻り値

Array - スタイル名の配列 (ストリング)。

例

次の例では、styleSheet という名前の StyleSheet オブジェクトを作成します。このオブジェクトには、heading と bodyText の 2 つのスタイルが含まれています。その後、StyleSheet オブジェクトの getStyleNames() メソッドを呼び出し、結果を names_array 配列に代入し、配列の内容を [出力] パネルに表示します。

```
import TextField.StyleSheet;
var my_styleSheet:StyleSheet = new StyleSheet();
my_styleSheet.setStyle("heading", {fontsize:'24px'});
my_styleSheet.setStyle("bodyText", {fontsize:'12px'});
var names_array:Array = my_styleSheet.getStyleNames();
trace(names_array.join("\n"));
```

[出力] パネルに次の情報が表示されます。

```
bodyText
heading
```

関連項目

getStyle (StyleSheet.getStyle メソッド)

load (StyleSheet.load メソッド)

```
public load(url:String) : Boolean
```

StyleSheet への CSS ファイルのロードを開始します。ロード操作は非同期です。ファイルのロードが完了したことを検出するには、onLoad() コールバックハンドラを使用します。CSS ファイルは、それをロードする SWF ファイルと同じドメインに置かれている必要があります。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

url:String - ロードする CSS ファイルの URL。URL は、SWF ファイルが現在置かれている URL と同じドメイン内にある必要があります。

戻り値

Boolean - パラメータが渡されない場合 (null) は false。それ以外の場合は true を返します。StyleSheet が正常にロードされたかどうかをチェックするには、onLoad() コールバックハンドラを使用します。

例

ActionScript 2.0 を使用したスタイルシートの非同期ロードの例については、getStyle() の例を参照してください。

次の例では、styles.css という CSS ファイルを StyleSheet オブジェクト my_styleSheet にロードします。ファイルが正常にロードされたら、StyleSheet オブジェクトを TextField オブジェクト news_txt に適用します。

```
import TextField.StyleSheet;
this.createTextField("news_txt", 999, 10, 10, 320, 240);
news_txt.multiline = true;
news_txt.wordWrap = true;
news_txt.html = true;

var my_styleSheet:StyleSheet = new StyleSheet();
my_styleSheet.onLoad = function(success:Boolean) {
  if (success) {
  news_txt.styleSheet = my_styleSheet;
  news_txt.htmlText = "<p class=\"heading\">Heading goes here!</p>"
    + "<p class=\"mainBody\">Lorem ipsum dolor sit amet, consectetuer "
    + "adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet "
    + "dolore magna aliquam erat volutpat.</p>";
  }
};
my_styleSheet.load("styles.css");
```

styles.css のコード全体は、getStyle() の例を参照してください。

関連項目

onLoad (StyleSheet.onLoad ハンドラ), getStyle (StyleSheet.getStyle メソッド)

onLoad (StyleSheet.onLoad ハンドラ)

```
onLoad = function(success:Boolean) {}
```

load() 処理が完了したときに呼び出されます。StyleSheet が正常にロードされた場合、success パラメータは true です。ファイルが受信されなかったか、サーバーから応答を受信する際にエラーが発生した場合は、success パラメータには false が渡されます。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

success:Boolean - CSS ファイルのロードが正常に終了したかどうかを示すブール値。正常にロードされた場合は true、それ以外の場合は false を返します。

例

次の例では、styles.css という CSS ファイルを StyleSheet オブジェクト my_styleSheet にロードします。ファイルのロードが正常に終了したら、StyleSheet オブジェクトを TextField オブジェクト news_txt に適用します。

```
import TextField.StyleSheet;
this.createTextField("news_txt", 999, 10, 10, 320, 240);
news_txt.multiline = true;
news_txt.wordWrap = true;
news_txt.html = true;

var my_styleSheet:StyleSheet = new StyleSheet();
my_styleSheet.onLoad = function(success:Boolean) {
  if (success) {
  news_txt.styleSheet = my_styleSheet;
  news_txt.htmlText = "<p class=\"heading\">Heading goes here!"
    + "</p><p class=\"mainBody\">Lorem ipsum dolor "
    + "sit amet, consectetuer adipiscing elit, sed diam nonummy "
    + "nibh euismod tincidunt ut laoreet dolore magna aliquam "
    + "erat volutpat.</p>";
  }
};
my_styleSheet.load("styles.css");
```

styles.css のコード全体は、getStyle() の例を参照してください。ActionScript 2.0 を使用したスタイルシートの非同期ロードの例については、getStyle() の例を参照してください。

関連項目

load (StyleSheet.load メソッド), getStyle (StyleSheet.getStyle メソッド)

parseCSS (StyleSheet.parseCSS メソッド)

```
public parseCSS(cssText:String) : Boolean
```

cssText の CSS を解析し、その内容を含む StyleSheet オブジェクトをロードします。cssText 内のスタイルが StyleSheet に既に含まれている場合は、StyleSheet のプロパティは維持され、cssText だけに含まれるスタイルが追加または変更されます。

ネイティブ CSS 解析機能を拡張するには、StyleSheet クラスのサブクラスを作成して、このメソッドをオーバーライドします。

対応バージョン： ActionScript 1.0、Flash Player 7

パラメータ

cssText:String - 解析する CSS テキスト。

戻り値

Boolean - テキストが正常に解析されたかどうかを示すブール値。正常に解析された場合は true、それ以外の場合は false を返します。

例

次の例では、css_str 内の CSS を解析します。このスクリプトは、CSS の解析が成功したかどうかに関する情報を表示してから、解析した CSS を [出力] パネルに表示します。

```
import TextField.StyleSheet;
var css_str:String = ".heading {font-family: Arial, Helvetica, sans-serif;
  font-size: 24px; font-weight: bold; }";
var my_styleSheet:StyleSheet = new StyleSheet();
if (my_styleSheet.parseCSS(css_str)) {
  trace("parsed successfully");
  dumpStyles(my_styleSheet);
} else {
  trace("unable to parse CSS");
}
//
function dumpStyles(styles:StyleSheet):Void {
  var styleNames_array:Array = styles.getStyleNames();
  for (var i = 0; i<styleNames_array.length; i++) {
    var styleName_str:String = styleNames_array[i];
    var styleObject:Object = styles.getStyle(styleName_str);
    trace(styleName_str);
    for (var prop in styleObject) {
      trace("\t"+prop+": "+styleObject[prop]);
    }
    trace("");
  }
}
```

setStyle (StyleSheet.setStyle メソッド)

```
public setStyle(name:String, style:Object) : Void
```

指定された名前を使用して、新しいスタイルを StyleSheet オブジェクトに追加します。指定したスタイルが StyleSheet に存在しない場合は、追加されます。指定したスタイルが StyleSheet に既に存在する場合は、置き換えられます。style パラメータに null を指定した場合は、指定したスタイルが削除されます。

このメソッドに渡したスタイルオブジェクトのコピーが作成されます。

サポートされるスタイルの一覧については、StyleSheet クラスの説明に記載されている表を参照してください。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

`name:String` - StyleSheet に追加するスタイルの名前。

`style:Object` - スタイルを記述するオブジェクト、または `null`。

例

次の例では、`emphasized` という名前のスタイルを `myStyleSheet` という StyleSheet に追加します。このスタイルには、`color` と `fontWeight` の 2 つのスタイルプロパティが含まれています。スタイルオブジェクトは、`{}` 演算子を使って定義します。

```
myStyleSheet.setStyle("emphasized", {color:'#000000',fontWeight:'bold'});
```

Object クラスのインスタンスを使用してスタイルオブジェクトを作成し、そのオブジェクト (`styleObj`) を `style` パラメータとして渡すこともできます。次に例を示します。

```
import TextField.StyleSheet;
var my_styleSheet:StyleSheet = new StyleSheet();

var styleObj:Object = new Object();
styleObj.color = "#000000";
styleObj.fontWeight = "bold";
my_styleSheet.setStyle("emphasized", styleObj);
delete styleObj;

var styleNames_array:Array = my_styleSheet.getStyleNames();
for (var i=0;i<styleNames_array.length;i++) {
  var styleName:String = styleNames_array[i];
  var thisStyle:Object = my_styleSheet.getStyle(styleName);
  trace(styleName);
  for (var prop in thisStyle) {
    trace("\t"+prop+": "+thisStyle[prop]);
  }
  trace("");
}
```

[出力] パネルに次の情報が表示されます。

```
emphasized
fontWeight: bold
color: #000000
```

メモ : Flash Player では `setStyle()` に渡したスタイルオブジェクトのコピーが作成されるため、コード例の `delete styleObj` コマンドにより、`setStyle()` に渡した元のスタイルオブジェクトを削除することでメモリの使用量が削減されます。

関連項目

`{}` オブジェクト初期化演算子 , `StyleSheet (TextField.StyleSheet)`

StyleSheet コンストラクタ

```
public StyleSheet()
```

StyleSheet オブジェクトを作成します。

対応バージョン： ActionScript 1.0、Flash Player 7

例

次の例では、スタイルシートをロードし、ドキュメントにロードされるスタイルをトレースします。次の ActionScript を ActionScript ファイルまたは FLA ファイルに追加します。

```
import TextField.StyleSheet;
var my_styleSheet:StyleSheet = new StyleSheet();
my_styleSheet.onLoad = function(success:Boolean) {
  if (success) {
    trace("Styles loaded:");
    var styles_array:Array = my_styleSheet.getStyleNames();
    trace(styles_array.join(newline));
  } else {
    trace("Error loading CSS");
  }
};
my_styleSheet.load("styles.css");
```

この styles.css ファイルには 2 つのスタイル `.heading` および `.mainbody` が含まれているため、[出力] パネルに次の情報が表示されます。

```
Styles loaded:
.heading
.mainBody
```

styles.css のコード全体は、`getStyle()` の例に示されています。

関連項目

`getStyle (StyleSheet.getStyle メソッド )`

transform (StyleSheet.transform メソッド)

```
public transform(style:Object) : TextFormat
```

CSS 解析機能を拡張します。上級開発者であれば、StyleSheet クラスを拡張して、このメソッドをオーバーライドできます。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

`style:Object` - スタイルを記述するオブジェクト、または `null`。このスタイルには、オブジェクトのプロパティとしてスタイル規則が含まれます。

戻り値

`TextFormat` - CSS 規則をテキストフォーマットプロパティにマッピングした結果を含む TextFormat オブジェクト。

例

次の例では、`transform()` メソッドを拡張します。

```
import TextField.StyleSheet;
class AdvancedCSS extends StyleSheet {
  public function AdvancedCSS() {
    trace("AdvancedCSS instantiated");
  }

  public function transform(styleObject):TextFormat {
    trace("tranform called");
  }
}
```

System

```
Object
  |
  +-System
```

```
public class System
extends Object
```

System クラスには、ユーザーのコンピュータ上で行われる特定の操作 (共有オブジェクト、カメラやマイクのローカル設定、クリップボードの使用など) に関連するプロパティが含まれています。System パッケージ内の次に示す特定のクラスには、追加のプロパティとメソッドがあります。Capabilities クラス、Security クラス、および IME クラスです。

対応バージョン : ActionScript 1.0、Flash Player 6

関連項目

capabilities (System.capabilities), security (System.security), IME (System.IME)

プロパティ一覧

オプション	プロパティ	説明
static	exactSettings:Boolean	ローカル設定 (カメラやマイクのアクセス許可など) またはローカルに永続化されたデータ (共有オブジェクト) にアクセスする場合に、スーパードメイン一致規則 (false) を使用するか、ドメイン完全一致規則 (true) を使用するかを示すブール値です。
static	useCodepage:Boolean	外部テキストファイルを解析するときに、Unicode を使用するか、または Flash Player を実行するオペレーティングシステムの従来のコードページを使用するかを示すブール値です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベントの一覧

イベント	説明
onStatus = function(infoObject: Object) {}	イベントハンドラ。特定のオブジェクトのスーパーイベントハンドラを提供します。

メソッド一覧

オプション	署名	説明
static	setClipboard(text: String) : Void	クリップボードの内容を、指定されたテキストストリングで置き換えます。
static	showSettings ([tabID:Number]) : Void	指定した [Macromedia Flash Player 設定] パネルを表示します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

exactSettings (System.exactSettings プロパティ)

```
public static exactSettings : Boolean
```

ローカル設定 (カメラやマイクのアクセス許可など) またはローカルに永続化されたデータ (共有オブジェクト) にアクセスする場合に、スーパードメイン一致規則 (false) を使用するか、ドメイン完全一致規則 (true) を使用するかを示すブール値です。デフォルト値は、Flash Player 7 以降用にパブリッシュされたファイルの場合は true、Flash Player 6 用にパブリッシュされたファイルの場合は false になります。

この値が true の場合、here.xyz.com でホストされた SWF ファイルの設定やデータは here.xyz.com という名前のディレクトリに、there.xyz.com でホストされた SWF ファイルの設定やデータは there.xyz.com という名前のディレクトリに保存されます。この値が false の場合、here.xyz.com、there.xyz.com、および xyz.com でホストされた SWF ファイルの設定やデータは共有され、すべて xyz.com という名前のディレクトリに保存されます。

このプロパティが false に設定されたファイルと、true に設定されたファイルとが存在する場合、異なるサブドメインの SWF ファイルで設定およびデータが共有されます。たとえば、このプロパティの値が、here.xyz.com でホストされた SWF ファイルでは false、xyz.com でホストされた SWF ファイルでは true である場合、どちらのファイルでも xyz.com ディレクトリに置かれている同じ設定とデータが使用されます。設定とデータを共有したくない場合は、各ファイルで適切な設定とデータが収められている場所が正しく指定されるように、このプロパティを設定してください。

このプロパティの値をデフォルト値以外に変更する場合は、ドキュメントの先頭フレームで変更します。このプロパティの値をデフォルト値以外に変更する場合は、スクリプトの先頭近くで変更します。ローカル設定へのアクセスが必要なアクティビティ (System.showSettings() や SharedObject.getLocal() など) の後で、このプロパティを変更することはできません。

loadMovie()、MovieClip.loadMovie()、または MovieClipLoader.loadClip() を使用して、ある SWF ファイルを別の SWF ファイルにロードした場合、Flash Player 7 用にパブリッシュされたすべてのファイルに対して 1 つの System.exactSettings 値が共有され、さらに、Flash Player 6 用にパブリッシュされたすべてのファイルに対して 1 つの System.exactSettings 値が共有されます。MovieClip.loadMovie() または MovieClipLoader.loadClip() を使用して、ある SWF ファイルを別の SWF ファイルにロードした場合、すべてのファイルに対して 1 つの System.exactSettings 値が共有されます。したがって、特定のバージョンの Player 用にパブリッシュされたファイルで、このプロパティに値を指定した場合は、ロードしようとするすべてのファイルについても同じ作業を行う必要があります。複数のファイルをロードした場合、最後にロードされたファイルで指定された設定が、それ以前に指定された設定よりも優先されます。

通常、System.exactSettings の値はデフォルトのままで対応できます。よく必要となる処理は、SWF ファイルのあるセッションで共有オブジェクトを保存し、同じ SWF ファイルの後のセッションで同じ共有オブジェクトを取得することです。System.exactSettings の値にかかわらず、この状況は常に true になります。しかし、元は Flash Player 6 用にパブリッシュされた SWF ファイルによって作成された共有オブジェクトを、Flash Player 7 以降用にパブリッシュされた SWF ファイルで取得できるように、System.exactSettings の値をデフォルト値から変更したい場合があります。Flash Player 6 用の SWF ファイルによって作成された共有オブジェクトは、その SWF ファイルのスーパードメインに応じたフォルダに保存されるため、Flash Player 7 用の SWF ファイルで共有オブジェクトを取得するには、スーパードメイン規則を使用する必要があります。これを行うには、Flash Player 7 用の SWF ファイルで System.exactSettings = false を指定する必要があります。Flash Player 6 用にパブリッシュされた SWF ファイルと、同じ共有オブジェクトデータを共有する Flash Player 7 用の SWF ファイルを共存させることもできます。この場合は、単純に System.exactSettings の値 (true または false) を調べて、その値を Flash Player 6 SWF ファイルと Flash Player 7 SWF ファイルで共通して使用します。

対応バージョン： ActionScript 1.0、Flash Player 7

例

次の例は、スーパードメイン一致規則の指定方法を示しています。

関連項目

loadMovie (MovieClip.loadMovie メソッド), loadClip (MovieClipLoader.loadClip メソッド), getLocal (SharedObject.getLocal メソッド), exactSettings (System.exactSettings プロパティ)

onStatus (System.onStatus ハンドラ)

```
onStatus = function(infoObject:Object) {}
```

イベントハンドラ。特定のオブジェクトのスーパーイベントハンドラを提供します。

LocalConnection、NetStream、SharedObject の各クラスには、情報、ステータス、またはエラーメッセージを提供するために情報オブジェクトを使用する onStatus イベントハンドラが用意されています。このイベントハンドラに応答するには、情報オブジェクトを処理する関数を作成する必要があります。また、返される情報オブジェクトの形式と内容を知っておく必要があります。

このような特定の onStatus メソッドだけでなく、`System.onStatus` と呼ばれるスーパー関数も用意されています。この関数は第 2 のエラーメッセージハンドラとして機能します。LocalConnection、NetStream、または SharedObject クラスのインスタンスによって情報オブジェクトに "error" の level プロパティが渡された場合、そのインスタンスに onStatus 関数が定義されていなければ、代わりに `System.onStatus` に定義されている関数が使用されます。

メモ : Camera クラスおよび Microphone クラスにも、onStatus ハンドラが用意されていますが、情報オブジェクトには "error" の level プロパティは渡されません。したがって、これらのハンドラに対して関数を指定しなかった場合、`System.onStatus` は呼び出されません。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

`infoObject:Object` - ステータスメッセージに従って定義されるパラメータ。

例

次の例では、特定のクラスに対応する onStatus 関数が存在しない場合に情報オブジェクトを処理する System.onStatus 関数を作成する方法を示します。

```
// Create generic function
System.onStatus = function(genericError:Object){
  // Your script would do something more meaningful here
  trace("An error has occurred. Please try again.");
}
```

次の例では、NetStream クラスのインスタンス用の onStatus 関数を作成する方法を示します。

```
// Create function for NetStream object

videoStream_ns.onStatus = function(infoObject:Object) {
  if (infoObject.code == "NetStream.Play.StreamNotFound") {
    trace("Could not find video file.");
  }
}
```

関連項目

onStatus (Camera.onStatus ハンドラ),onStatus (LocalConnection.onStatus ハンドラ),onStatus (Microphone.onStatus ハンドラ),onStatus (NetStream.onStatus ハンドラ),onStatus (SharedObject.onStatus ハンドラ)

setClipboard (System.setClipboard メソッド)

```
public static setClipboard(text:String) : Void
```

クリップボードの内容を、指定されたテキストストリングで置き換えます。

メモ：セキュリティ上、システムクリップボードの内容を読み取ることはできません。つまり、対応する System.getClipboard() メソッドはありません。

対応バージョン： ActionScript 1.0、Flash Player 7 - Flash Player 6 以降用にパブリッシュされた SWF ファイル。Flash Player 7 以降で再生されます。

パラメータ

text:String - システムクリップボードの現在の内容 (存在する場合) を置き換えるプレーンテキストストリング文字。

例

次の例では、"Hello World" というテキストをシステムクリップボードに置きます。

```
System.setClipboard("Hello world");
```

次の例では、実行時に 2 つのテキストフィールド in_txt および out_txt を作成します。in_txt フィールド内のテキストを選択する際には、copy_btn をクリックしてデータをクリップボードにコピーできます。その後で、そのテキストを out_txt フィールドに貼り付けることができます。

```
this.createTextField("in_txt", this.getNextHighestDepth(), 10, 10, 160, 120);
in_txt.multiline = true;
in_txt.border = true;
in_txt.text = "lorum ipsum...";
this.createTextField("out_txt", this.getNextHighestDepth(), 10, 140, 160, 120);
out_txt.multiline = true;
out_txt.border = true;
out_txt.type = "input";

copy_btn.onRelease = function() {
  System.setClipboard(in_txt.text);
  Selection.setFocus("out_txt");
};
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

showSettings (System.showSettings メソッド)

```
public static showSettings([tabID:Number]) : Void
```

指定した [Macromedia Flash Player 設定] パネルを表示します。ユーザーはパネルを使用して、以下のアクションを実行できます。

- カメラとマイクへのアクセスの許可または禁止
- 共有オブジェクト用に使用できるローカルディスクの容量の指定
- デフォルトのカメラとマイクの選択
- マイクの録音レベルとエコー抑制設定の指定

たとえば、アプリケーションでカメラを使用する必要がある場合は、[Macromedia Flash Player 設定] パネルの [プライバシー] で [許可] を選択するようにユーザーに要求し、その後で `System.showSettings(0)` コマンドを実行することができます。ステージのサイズは必ず 215 x 138 ピクセル以上に設定してください。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

`tabID:Number` - 表示する [Macromedia Flash Player 設定] パネルを示す数値。次の値を指定できます。

`panel` の値	表示される設定パネル
なし (パラメータを省略) またはサポートしていない値	前回ユーザーが [Macromedia Flash Player 設定] パネルを閉じたときに開いていたパネル。
0	プライバシー
1	ローカル記憶領域
2	マイク
3	カメラ

例

次の例は、Flash Player 設定の [ローカル記憶領域] パネルを表示する方法を示しています。

```
System.showSettings(1);
```

関連項目

get (Camera.get メソッド), get (Microphone.get メソッド), getLocal (SharedObject.getLocal メソッド)

useCodepage (System.useCodepage プロパティ)

```
public static useCodepage : Boolean
```

外部テキストファイルを解析するときに、Unicode を使用するか、または Flash Player を実行するオペレーティングシステムの従来のコードページを使用するかを示すブール値です。System.useCodepage のデフォルト値は false です。

- このプロパティを false に設定すると、外部テキストファイルは Unicode と解釈されます。これらのファイルは、保存する際に Unicode でエンコードする必要があります。
- このプロパティを true に設定すると、Flash Player を実行するオペレーティングシステムの通常のコードページを使用して外部テキストファイルが解釈されます。

外部ファイルとしてロードしたテキスト (loadVariables() または getURL() ステートメント、LoadVars クラスまたは XML クラスを使用) を保存する際には、このテキストファイルを Flash Player で Unicode と認識できるように、Unicode でエンコードする必要があります。外部ファイルを Unicode でエンコードするには、Windows 2000 のメモ帳など、Unicode をサポートするアプリケーションでファイルを保存する必要があります。

Unicode でエンコードされていない外部テキストファイルをインクルードまたはロードする際には、System.useCodepage を true に設定してください。データをロードする SWF ファイルの最初のフレームの先頭行として、次のコードを追加します。

```
System.useCodepage = true;
```

このコードがあると、外部テキストは、Flash Player を実行するオペレーティングシステムの通常のコードページを使用して解釈されます。一般に、英語の Windows オペレーティングシステムの場合は CP1252、日本語のオペレーティングシステムの場合は Shift-JIS が使用されます。Flash Player 6 以降で System.useCodepage を true に設定すると、テキストは Flash Player 5 の場合と同様に扱われます。Flash Player 5 では、すべてのテキストは、Flash Player を実行するオペレーティングシステムの通常のコードページを使用して解釈されていました。

System.useCodepage を true に設定した場合、外部テキストファイル内で使用されている文字が Flash Player を実行するオペレーティングシステムの通常のコードページに含まれていないと、そのテキストは表示されないことに注意してください。たとえば、中国語を含む外部テキストファイルをロードする場合、CP1252 コードページを使用するシステムではこれらの文字を表示できません。CP1252 コードページには中国語が含まれていないためです。

SWF ファイルで使用する外部テキストファイルをすべてのプラットフォームのユーザーが表示できるようにするには、すべての外部テキストファイルを Unicode で保存し、System.useCodepage にはデフォルト値の false を使用します。これにより、Flash Player 6 以降ではテキストが Unicode として解釈されます。

対応バージョン : ActionScript 1.0、Flash Player 6

TextField

```
Object
  |
  +-TextField

public dynamic class TextField
extends Object
```

TextField クラスは、テキストの表示と入力用の領域を作成するために使用されます。SWF ファイルのすべてのダイナミックテキストフィールドおよびテキスト入力フィールドは、TextField クラスのインスタンスです。プロパティインスペクタを使用して、テキストフィールドにインスタンス名を付けることができます。また、TextField クラスのメソッドとプロパティを使用して、ActionScript でテキストフィールドを操作できます。TextField インスタンスの名前は、ムービーエクスプローラに表示されます。[アクション] パネルの [ターゲットパスの挿入] ダイアログボックスにも表示されます。

テキストフィールドを動的に作成する場合は、new 演算子を使用しません。`MovieClip.createTextField()` を使用します。テキストフィールドのデフォルトのサイズは 100 x 100 ピクセルです。

TextField クラスのメソッドを使用すると、オーサリング時または実行時に作成したダイナミックテキストフィールドやテキスト入力フィールドにテキストを設定、選択、および操作できます。

ActionScript には、テキストを実行時にフォーマットする方法がいくつも用意されています。TextFormat クラスでは、TextField オブジェクトの文字フォーマットと段落フォーマットを設定できます。Flash Player 7 以降では、TextField.styleSheet プロパティと StyleSheet クラスを使用して、テキストフィールドに CSS (Cascading Style Sheet) スタイルを適用できます。CSS を使用すると、ビルトイン HTML タグのスタイル設定、新しいフォーマットタグの定義、またはスタイルの適用を行うことができます。HTML 形式のテキスト (CSS スタイルを使用している場合も可) をテキストフィールドに直接割り当てることができます。Flash Player 7 以降では、テキストフィールドに割り当てる HTML テキストに埋め込みメディア (ムービークリップ、SWF ファイル、JPEG ファイル、GIF ファイル、PNG ファイル) を含めることができます。この場合テキストは、Web ブラウザ上で HTML ドキュメントの埋め込みメディアの周りをテキストが囲むのと同じように、埋め込みメディアの周りを囲みます。

Flash Player では、テキストのフォーマットに利用できる HTML タグのサブセットをサポートしています。

対応バージョン： ActionScript 1.0、Flash Player 6

関連項目

Object, createTextField (MovieClip.createTextField メソッド)

プロパティ一覧

オプション	プロパティ	説明
	_alpha:Number	テキストフィールドのアルファ透明度値を設定または取得します。
	antiAliasType:String	この TextField インスタンスに使用されるアンチエイリアス処理のタイプです。
	autoSize:Object	テキストフィールドの自動的な拡大・縮小および整列を制御します。
	background:Boolean	テキストフィールドに背景の塗りがあるかどうかを指定します。
	backgroundColor:Number	テキストフィールドの背景の色です。
	border:Boolean	テキストフィールドに境界線があるかどうかを指定します。
	borderColor:Number	テキストフィールドの境界線の色です。
	bottomScroll:Number (読み取り専用)	テキストフィールドの現在の表示範囲で最終行を示す整数 (1 から始まるインデックス) です。
	condenseWhite:Boolean	HTML テキストフィールド内の余分な空白 (スペース、改行など) を削除するかどうかを指定するブール値です。
	embedFonts:Boolean	埋め込みフォントのアウトラインを使用してレンダリングするかどうかを指定します。
	filters:Array	テキストフィールドに現在関連付けられている各フィルタオブジェクトが格納されている、インデックスの配列です。
	gridFitType:String	この TextField インスタンスに使用されるグリッド合わせのタイプです。
	_height:Number	ピクセル単位で示したテキストフィールドの高さです。
	_highquality:Number	**非推奨** Flash Player 7 以降では使用しないでください。このプロパティの代わりに TextField._quality を使用します。 現在の SWF ファイルに適用されるアンチエイリアスのレベルを指定します。
	hscroll:Number	現在の水平スクロール位置を示します。
	html:Boolean	テキストフィールドに HTML 表現があるかどうかを示すフラグです。

オプション	プロパティ	説明
	htmlText:String	テキストフィールドが HTML テキストフィールドである場合、このプロパティにはテキストフィールドの内容の HTML 表現が入ります。
	length:Number (読み取り専用)	テキストフィールド内の文字数を示します。
	maxChars:Number	テキストフィールドに入る最大の文字数を示します。
	maxhscroll:Number (読み取り専用)	TextField.hscroll の最大値を示します。
	maxscroll:Number (読み取り専用)	TextField.scroll の最大値を示します。
	menu:ContextMenu	*contextMenu* に指定した ContextMenu オブジェクトをテキストフィールド *my_txt* に関連付けます。
	mouseWheelEnabled: Boolean	複数行のテキストフィールドで、マウスポインタがテキストフィールドをクリックしユーザーがホイールを回転させると、自動的にスクロールするかどうかを示すブール値です。
	multiline:Boolean	テキストフィールドが複数行テキストフィールドであるかどうかを示します。
	_name:String	テキストフィールドのインスタンス名です。
	_parent:MovieClip	現在のテキストフィールドまたはオブジェクトを含むムービークリップまたはオブジェクトへの参照です。
	password:Boolean	テキストフィールドがパスワードテキストフィールドであるかどうかを指定します。
	_quality:String	SWF ファイルに使用するレンダリング品質です。
	restrict:String	ユーザーがテキストフィールドに入力できる文字のセットを示します。
	_rotation:Number	テキストフィールドの元の位置からの回転角を度単位で指定します。
	scroll:Number	テキストフィールドのテキストの垂直座標です。
	selectable:Boolean	テキストフィールドが選択可能であるかどうかを示すブール値です。
	sharpness:Number	この TextField インスタンス内の文字エッジのシャープネスです。
	_soundbuftime:Number	サウンドのストリーミングを開始するまでにサウンドをプリバッファする秒数です。

オプション	プロパティ	説明
	styleSheet:StyleSheet	テキストフィールドにスタイルシートを関連付けます。
	tabEnabled:Boolean	テキストフィールドが自動タブ順に含まれるかどうかを指定します。
	tabIndex:Number	SWF ファイル内のオブジェクトのタブ順をカスタマイズできます。
	_target:String (読み取り専用)	テキストフィールドインスタンスのターゲットパスです。
	text:String	テキストフィールドの現在のテキストを示します。
	textColor:Number	テキストフィールドのテキストの色を示します。
	textHeight:Number	テキストの高さをピクセル単位で示します。
	textWidth:Number	テキストの幅をピクセル単位で示します。
	thickness:Number	この TextField インスタンス内の文字エッジの太さです。
	type:String	テキストフィールドのタイプを指定します。
	_url:String (読み取り専用)	テキストフィールドを作成した SWF ファイルの URL を取得します。
	variable:String	テキストフィールドに関連付けられた変数の名前です。
	_visible:Boolean	テキストフィールド *my_txt* が表示されるかどうかを示すブール値です。
	_width:Number	ピクセル単位で示したテキストフィールドの幅です。
	wordWrap:Boolean	テキストフィールドのテキストを折り返すかどうかを示すブール値です。
	_x:Number	親ムービークリップのローカル座標を基準にしてテキストフィールドの x 座標を設定する整数です。
	_xmouse:Number (読み取り専用)	テキストフィールドを基準にしたポインタの x 座標を返します。
	_xscale:Number	テキストフィールドの基準点から適用する、テキストフィールドの水平方向の拡大・縮小率 (パーセンテージ) を決定します。
	_y:Number	親ムービークリップのローカル座標を基準にしたテキストフィールドの y 座標です。
	_ymouse:Number (読み取り専用)	テキストフィールドを基準にしたポインタの y 座標を示します。

オプション	プロパティ	説明
	_yscale:Number	テキストフィールドの基準点から適用する、テキストフィールドの垂直方向の拡大・縮小率 (パーセンテージ) です。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベントの一覧

イベント	説明
onChanged = function(changedField:TextField) {}	イベントハンドラ / リスナー。テキストフィールドの内容が変更されると、呼び出されます。
onKillFocus = function(newFocus:Object) {}	テキストフィールドがキーボードフォーカスを失うと、呼び出されます。
onScroller = function(scrolledField: TextField) {}	イベントハンドラ / リスナー。テキストフィールドのスクロールプロパティが変わると呼び出されます。
onSetFocus = function(oldFocus:Object) {}	テキストフィールドがキーボードフォーカスを受け取ると、呼び出されます。

メソッド一覧

オプション	署名	説明
	addListener(listener: Object) : Boolean	TextField イベント通知を受け取るオブジェクトを登録します。
	getDepth() : Number	テキストフィールドの深度を返します。
static	getFontList() : Array	Flash Player のホストシステム上のフォント名を配列として返します。
	getNewTextFormat() : TextFormat	テキストフィールドのテキストフォーマットオブジェクトのコピーを含む TextFormat オブジェクトを返します。
	getTextFormat ([beginIndex:Number], [endIndex:Number]) : TextFormat	文字、文字の範囲または TextField オブジェクト全体に対して TextFormat オブジェクトを返します。

オプション	署名	説明
	removeListener (listener:Object) : Boolean	TextField.addListener() でテキストフィールドインスタンスに以前に登録したリスナーオブジェクトを削除します。
	removeTextField() : Void	テキストフィールドを削除します。
	replaceSel(newText: String) : Void	現在の選択内容を newText パラメータの内容に置き換えます。
	replaceText(beginIndex: Number, endIndex:Number, newText:String) : Void	指定されたテキストフィールドの beginIndex パラメータと endIndex パラメータで指定した文字範囲を、newText パラメータの内容に置き換えます。
	setNewTextFormat(tf: TextFormat) : Void	テキストフィールドに新しいデフォルトのテキストフォーマットを設定します。
	setTextFormat ([beginIndex:Number], [endIndex:Number], textFormat:TextFormat) : Void	textFormat パラメータで指定したテキストフォーマットを、テキストフィールド内のテキストの一部またはすべてに適用します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

addListener (TextField.addListener メソッド)

```
public addListener(listener:Object) : Boolean
```

TextField イベント通知を受け取るオブジェクトを登録します。このオブジェクトは、onChanged イベントハンドラおよび onScroller イベントハンドラが呼び出されたときにイベント通知を受け取ります。テキストフィールドが変更またはスクロールされると、TextField.onChanged イベントハンドラと TextField.onScroller イベントハンドラが呼び出され、その後リスナーとして登録されているオブジェクトの onChanged イベントハンドラと onScroller イベントハンドラが呼び出されます。複数のオブジェクトをリスナーとして登録することができます。

テキストフィールドからリスナーオブジェクトを削除するには、TextField.removeListener() を呼び出します。

onScroller イベントハンドラと onChanged イベントハンドラには、テキストフィールドインスタンスへの参照がパラメータとして渡されます。イベントハンドラメソッドにパラメータを含めることによって、このデータを受け取ることができます。たとえば、次の例では、onScroller イベントハンドラで txt パラメータを受け取ります。その後、trace ステートメントでこのパラメータを使用して、テキストフィールドのインスタンス名を [出力] パネルに表示します。

```
my_txt.onScroller = function(textfield_txt:TextField) {
  trace(textfield_txt._name+" scrolled");
};
```

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

listener: Object - onChanged イベントハンドラまたは onScroller イベントハンドラを持つオブジェクト。

戻り値

Boolean -

例

次の例では、テキスト入力フィールド my_txt に対して onChanged ハンドラを定義します。その後、新しいリスナーオブジェクト txtListener を定義し、そのオブジェクト用に onChanged ハンドラを定義します。このハンドラは、テキストフィールド my_txt が変更されると呼び出されます。このコードの最終行では、TextField.addListener を呼び出し、リスナーオブジェクト txtListener をテキストフィールド my_txt に登録します。これにより、このリスナーは my_txt が変更されると通知されるようになります。

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 100, 22);
my_txt.border = true;
my_txt.type = "input";

my_txt.onChanged = function(textfield_txt:TextField) {
  trace(textfield_txt._name+" changed");
};
var txtListener:Object = new Object();
txtListener.onChanged = function(textfield_txt:TextField) {
  trace(textfield_txt._name+" changed and notified myListener");
};
my_txt.addListener(txtListener);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

onChanged (TextField.onChanged ハンドラ), onScroller (TextField.onScroller ハンドラ), removeListener (TextField.removeListener メソッド)

_alpha (TextField._alpha プロパティ)

```
public _alpha : Number
```

テキストフィールドのアルファ透明度値を設定または取得します。有効な値は 0 (完全な透明) ～ 100 (完全な不透明) です。デフォルト値は 100 です。透明度値は、デバイスフォントを使用するテキストフィールドではサポートされません。テキストフィールドで _alpha 透明度プロパティを使用するには、埋め込みフォントを使用する必要があります。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次のコードは、テキストフィールド my_txt の _alpha プロパティを 20% に設定します。[ライブラリ] オプションメニューの [新しいフォント] を選択して、ライブラリに新しいフォントシンボルを作成します。次に、新しいフォントのリンケージを my font に設定します。フォントシンボルのリンケージを my font に設定します。次の ActionScript コードを FLA ファイルまたは AS ファイルに追加します。

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.font = "my font";
// where 'my font' is the linkage name of a font in the Library
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 100, 22);
my_txt.border = true;
my_txt.embedFonts = true;
my_txt.text = "Hello World";
my_txt.setTextFormat(my_fmt);
my_txt._alpha = 20;
```

この例で使用されている MovieClip.getNextHighestDepth() メソッドでは、Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

_alpha (Button._alpha プロパティ), _alpha (MovieClip._alpha プロパティ)

antiAliasType (TextField.antiAliasType プロパティ)

```
public antiAliasType : String
```

この TextField インスタンスに使用されるアンチエイリアス処理のタイプ。高度なアンチエイリアスは、Flash Player 8 以降でのみ使用できます。この設定は、フォントが埋め込まれている (embedFonts プロパティが true に設定されている) 場合にのみ制御できます。Flash Player 8 では、デフォルト設定は "advanced" です。

このプロパティの値を設定するには、次のストリング値を使用します。

ストリング値	説明
"normal"	通常のテキストのアンチエイリアスを適用します。これは、Flash Player のバージョン 7 以前で使用されているアンチエイリアスのタイプと一致します。
"advanced"	文字を読みやすくする高度なアンチエイリアスを適用します。これは Flash Player 8 で使用可能になった機能です。高度なアンチエイリアスでは、小さいサイズのフォントフェイスを高品質でレンダリングすることができます。この機能は、小さいフォントのテキストが多いアプリケーションで使用する場合に最適です。48 ポイントより大きいフォントに対して高度なアンチエイリアスを使用することはお勧めしません。

対応バージョン : ActionScript 1.0、Flash Player 8

例

この例では、2 つのテキストフィールドを作成し、最初のテキストフィールドにのみ高度なアンチエイリアスを適用します。リンケージ識別子が "Times-12" に設定されたライブラリにフォントが埋め込まれていることが前提です。フォントを埋め込むには、次の手順を実行します。

- ライブラリを開きます。
- ライブラリの右上隅にある [ライブラリ] オプションメニューをクリックします。
- ドロップダウンリストから [新しいフォント] を選択します。
- フォントに "Times-12" という名前を付けます。
- ドロップダウンリストから "Times New Roman" を選択します。
- [OK] ボタンをクリックします。
- 新しく作成したフォントを右クリックし、[リンケージ] を選択します。
- [ActionScript に書き出し] チェックボックスをオンにします。
- [OK] ボタンをクリックして、デフォルトの識別子 "Times-12" をそのまま使用します。

```
var my_format:TextFormat = new TextFormat();
my_format.font = "Times-12";

var my_text1:TextField = this.createTextField("my_text1",
  this.getNextHighestDepth(), 10, 10, 300, 30);
my_text1.text = "This text uses advanced anti-aliasing.";
my_text1.antiAliasType = "advanced";
my_text1.border = true;
my_text1.embedFonts = true;
my_text1.setTextFormat(my_format);

var my_text2:TextField = this.createTextField("my_text2",
  this.getNextHighestDepth(), 10, 50, 300, 30);
my_text2.text = "This text uses normal anti-aliasing."
my_text2.antiAliasType = "normal";
my_text2.border = true;
my_text2.embedFonts = true;
my_text2.setTextFormat(my_format);
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

関連項目

TextRenderer (flash.text.TextRenderer), gridFitType (TextField.gridFitType プロパティ), thickness (TextField.thickness プロパティ), sharpness (TextField.sharpness プロパティ)

autoSize (TextField.autoSize プロパティ)

public autoSize : Object

テキストフィールドの自動的な拡大・縮小および整列を制御します。autoSize の有効な値は、"none" (デフォルト)、"left"、"right"、"center" の 4 つです。autoSize プロパティを設定する場合、true は "left" を指定するのと同じであり、false は "none" を指定するのと同じです。

autoSize と TextField.wordWrap に指定した値によって、テキストフィールドが左、右、または下に伸縮します。これらのプロパティのデフォルト値は false です。

autoSize が "none" (デフォルト値) または false に設定されていると、サイズ変更は行われません。

autoSize を "left" または true に設定すると、テキストは左揃えテキストとして扱われます。つまり、テキストフィールドの左側が固定され、単一行テキストフィールドの右側のみが伸縮します。テキストに改行 ("\n" または "\r" など) がある場合、テキストの次の行が収まるようにフィールドの下側が拡張されます。wordWrap も true に設定した場合、テキストフィールドの下側だけが伸縮し、右側は固定されたままになります。

autoSize を "right" に設定すると、テキストは右揃えテキストとして扱われます。つまり、テキストフィールドの右側が固定され、単一行テキストフィールドの左側のみが伸縮します。テキストに改行 ("\n" または "\r" など) がある場合、テキストの次の行が収まるようにフィールドの下側が拡張されます。wordWrap も true に設定した場合、テキストフィールドの下側だけが伸縮し、左側は固定されたままになります。

autoSize を "center" に設定すると、テキストは中央揃えテキストとして扱われます。つまり、単一行テキストフィールドのサイズ変更を行うと、左右両側が均等に伸縮されます。テキストに改行 ("\n" または "\r" など) がある場合、テキストの次の行が収まるようにフィールドの下側が拡張されます。wordWrap も true に設定した場合、テキストフィールドの下側だけが伸縮し、左右両側は固定されたままになります。

対応バージョン： ActionScript 1.0、Flash Player 6

例

次のコードで autoSize の値を変えてみてください。指定された値に応じてフィールドのサイズが変化するのがわかります。SWF ファイルの再生中にマウスでクリックすると、各テキストフィールドの "short text" ストリングが、autoSize に各種設定が適用された、より長いテキストに置き換わります。

```
this.createTextField("left_txt", 997, 10, 10, 70, 30);
this.createTextField("center_txt", 998, 10, 50, 70, 30);
this.createTextField("right_txt", 999, 10, 100, 70, 30);
this.createTextField("true_txt", 1000, 10, 150, 70, 30);
this.createTextField("false_txt", 1001, 10, 200, 70, 30);

left_txt.text = "short text";
left_txt.border = true;

center_txt.text = "short text";
center_txt.border = true;

right_txt.text = "short text";
right_txt.border = true;

true_txt.text = "short text";
true_txt.border = true;

false_txt.text = "short text";
false_txt.border = true;

// create a mouse listener object to detect mouse clicks
var myMouseListener:Object = new Object();
// define a function that executes when a user clicks the mouse
```

```
myMouseListener.onMouseDown = function() {
  left_txt.autoSize = "left";
  left_txt.text = "This is much longer text";
  center_txt.autoSize = "center";
  center_txt.text = "This is much longer text";
  right_txt.autoSize = "right";
  right_txt.text = "This is much longer text";
  true_txt.autoSize = true;
  true_txt.text = "This is much longer text";
  false_txt.autoSize = false;
  false_txt.text = "This is much longer text";
};
// register the listener object with the Mouse object
Mouse.addListener(myMouseListener);
```

background (TextField.background プロパティ)

```
public background : Boolean
```

テキストフィールドに背景の塗りがあるかどうかを指定します。true である場合、テキストフィールドに背景の塗りがあります。false である場合、テキストフィールドには背景の塗りがありません。

対応バージョン：ActionScript 1.0、Flash Player 6

例

次の例では、キーボード上の任意のキーを押したときに、オン、オフが切り替わる背景色を持つテキストフィールドを作成します。

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 320, 240);
my_txt.border = true;
my_txt.text = "Lorum ipsum";
my_txt.backgroundColor = 0xFF0000;

var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  my_txt.background = !my_txt.background;
};
Key.addListener(keyListener);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

backgroundColor (TextField.backgroundColor プロパティ)

public backgroundColor : Number

テキストフィールドの背景の色。デフォルト値は 0xFFFFFF (白) です。このプロパティは、現在背景がない場合でも取得または設定できます。ただし、背景の色が表示されるのはテキストフィールドに境界線がある場合のみです。

対応バージョン : ActionScript 1.0、Flash Player 6

例

TextField.background の例を参照してください。

関連項目

background (TextField.background プロパティ)

border (TextField.border プロパティ)

public border : Boolean

テキストフィールドに境界線があるかどうかを指定します。true である場合、テキストフィールドに境界線があります。false である場合、テキストフィールドに境界線がありません。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、テキストフィールド my_txt を作成し、境界線のプロパティを true に設定し、フィールド内にテキストを表示します。

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 320, 240);
my_txt.border = true;
my_txt.text = "Lorum ipsum";
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

borderColor (TextField.borderColor プロパティ)

public borderColor : Number

テキストフィールドの境界線の色。デフォルト値は 0x000000 (黒) です。このプロパティは、現在境界線がない場合でも取得または設定できます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、テキストフィールド my_txt を作成し、境界線のプロパティを true に設定し、フィールド内にテキストを表示します。

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 320, 240);
my_txt.border = true;
my_txt.borderColor = 0x00FF00;
my_txt.text = "Lorum ipsum";
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

border (TextField.border プロパティ)

bottomScroll (TextField.bottomScroll プロパティ)

public bottomScroll : Number (読み取り専用)

テキストフィールドの現在の表示範囲で最終行を示す整数 (1 から始まるインデックス)。テキストフィールドは、テキストのブロックにかぶせたウィンドウのようなものです。TextField.scroll プロパティは、そのウィンドウに表示されている先頭行を示すインデックス (1 から始まるインデックス) です。

TextField.scroll から TextField.bottomScroll までのすべての行が、テキストフィールドに現在表示されています。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、テキストフィールドを作成し、そこにテキストを流し込みます。"my_btn" というインスタンス名のボタンを挿入する必要があります。ボタンをクリックすると、テキストフィールドの comment_txt フィールドで scroll プロパティと bottomScroll プロパティがトレースされます。

```
this.createTextField("comment_txt", this.getNextHighestDepth(), 0, 0, 160, 120);
comment_txt.html = true;
comment_txt.selectable = true;
comment_txt.multiline = true;
comment_txt.wordWrap = true;
comment_txt.htmlText = "<b>What is hexadecimal?</b><br>"
  + "The hexadecimal color system uses six digits to represent color values. "
  + "Each digit has sixteen possible values or characters. The characters range"
  + " from 0 to 9 and then A to F. Black is represented by (#000000) and white, "
  + "at the opposite end of the color system, is (#FFFFFF).";
my_btn.onRelease = function() {
  trace("scroll: "+comment_txt.scroll);
  trace("bottomScroll: "+comment_txt.bottomScroll);
};
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

condenseWhite (TextField.condenseWhite プロパティ)

```
public condenseWhite : Boolean
```

HTML テキストフィールド内の余分な空白 (スペース、改行など) を削除するかどうかを指定するブール値です。デフォルト値は false です。

この値を true に設定した場合は、テキストフィールド内で改行を指定するときに
 や <P> などの標準の HTML コマンドを使用する必要があります。

condenseWhite プロパティは htmlText プロパティを設定する前に設定します。

テキストフィールドの html プロパティが false である場合、このプロパティは無視されます。

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の例では、2 つのテキストフィールド first_txt および second_txt を作成します。2 番目のテキストフィールドから空白が削除されます。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
var my_str:String = "Hello\tWorld\nHow are you?\t\t\tEnd";

this.createTextField("first_txt", this.getNextHighestDepth(), 10, 10, 160, 120);
first_txt.html = true;
first_txt.multiline = true;
first_txt.wordWrap = true;
first_txt.condenseWhite = false;
first_txt.border = true;
first_txt.htmlText = my_str;

this.createTextField("second_txt", this.getNextHighestDepth(), 180, 10, 160,
  120);
second_txt.html = true;
second_txt.multiline = true;
second_txt.wordWrap = true;
second_txt.condenseWhite = true;
second_txt.border = true;
second_txt.htmlText = my_str;
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

html (TextField.html プロパティ)

embedFonts (TextField.embedFonts プロパティ)

public embedFonts : Boolean

埋め込みフォントのアウトラインを使用してレンダリングするかどうかを指定します。true である場合に、埋め込みフォントアウトラインを使用してテキストフィールドをレンダリングすることを示すブール値です。false である場合は、デバイスフォントを使用してテキストフィールドをレンダリングします。

テキストフィールドの embedFonts を true に設定する場合、テキストフィールドに適用される TextFormat オブジェクトの font プロパティ経由で、そのテキストのフォントを指定する必要があります。指定されたフォントが対応するリンケージインスタンス名を持つライブラリに存在しない場合は、テキストは表示されません。

対応バージョン : ActionScript 1.0、Flash Player 6

例

この例では、ダイナミックテキストフィールド my_txt を作成してから、次の ActionScript を使用してフォントの埋め込みとテキストフィールドの回転を行う必要があります。my font への参照は、リンケージを my font に設定して、ライブラリ内のフォントシンボルを参照します。この例では、my font というライブラリ内に、リンケージプロパティで、[識別子] が my font に設定され、[ActionScript に書き出し] と [最初のフレームに書き出し] が選択されているフォントシンボルがあるものとします。

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.font = "my font";

this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 160, 120);
my_txt.wordWrap = true;
my_txt.embedFonts = true;
my_txt.text = "Hello world";
my_txt.setTextFormat(my_fmt);
my_txt._rotation = 45;
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

filters (TextField.filters プロパティ)

```
public filters : Array
```

テキストフィールドに現在関連付けられている各フィルタオブジェクトが格納されている、インデックスの配列。flash.filters パッケージには、使用できる特定のフィルタを定義する複数のクラスが含まれています。

フィルタは、ActionScript コードを使用して、設計時または実行時に Flash オーサリングツールで適用できます。ActionScript を使用してフィルタを適用するには、TextField.filters 配列全体の一時コピーを作成してその一時配列を変更した後、一時配列の値を TextField.filters 配列に代入し直す必要があります。新しいフィルタオブジェクトを TextField.filters 配列に直接追加することはできません。次のコードは、myTextField という名前のターゲットテキストフィールドには影響しません。

```
myTextField.filters[0].push(myDropShadow);
```

ActionScript を使用してフィルタを追加するには、次の手順を実行する必要があります。ターゲットムービークリップの名前は myTextField とします。

- 選択したフィルタクラスのコンストラクタ関数を使用して、新しいフィルタオブジェクトを作成します。
- myTextField.filters 配列の値を、myFilters などの名前の一時配列に割り当てます。
- 新しいフィルタオブジェクトを一時配列 myFilters に追加します。
- 一時配列の値を myTextField.filters 配列に代入します。

filters 配列が空の場合、一時配列を使用する必要はありません。代わりに、作成したフィルタオブジェクトを格納する配列リテラルを直接代入することができます。

設計時または実行時に作成されたかどうかに関わらず、既存のフィルタオブジェクトを変更するには、次の方法で filters 配列のコピーを変更する必要があります。

- myTextField.filters 配列の値を、myFilters などの名前の一時配列に割り当てます。
- 一時配列 myFilters を使用してプロパティを変更します。たとえば、配列内の最初のフィルタの quality プロパティを設定するには myList[0].quality = 1; というコードを使用できます。
- 一時配列の値を myTextField.filters 配列に代入します。

テキストフィールドのフィルタをクリアするには、filters を空の配列 ([]) に設定します。

複数のフィルタを格納する filters 配列を使用していて、各配列インデックスに割り当てられているフィルタの種類を追跡する必要がある場合、独自の filters 配列を維持し、別のデータ構造を使用して、各配列インデックスに関連付けられているフィルタの種類を追跡できます。filters 配列の各インデックスに関連付けられているフィルタの種類を簡単に判別する方法はありません。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、フィールド myTextField にドロップシャドウフィルタを追加します。

```
var myDropFilter = new flash.filters.DropShadowFilter();
var myFilters:Array = myTextField.filters;
myFilters.push(myDropFilter);
myTextField.filters = myFilters;
```

次の例では、配列内の最初のフィルタの quality 設定を 15 に変更します。この例は、myTextField テキストフィールドに少なくとも 1 つのフィルタオブジェクトが関連付けられている場合にのみ機能します。

```
var myList:Array = myTextField.filters;
myList[0].quality = 15;
myTextField.filters = myList;
```

getDepth (TextField.getDepth メソッド)

```
public getDepth() : Number
```

テキストフィールドの深度を返します。

対応バージョン : ActionScript 1.0、Flash Player 6

戻り値

`Number` - テキストフィールドの深度を表す整数。

例

次の例は、それぞれ異なる深度にあるテキストフィールドを示しています。ステージ上にダイナミックテキストフィールドを作成し、次の ActionScript コードを FLA ファイルまたは AS ファイルに追加します。このコードでは、実行時に 2 つのテキストフィールドが動的に作成され、それぞれの深度が出力されます。

```
this.createTextField("first_mc", this.getNextHighestDepth(), 10, 10, 100, 22);
this.createTextField("second_mc", this.getNextHighestDepth(), 10, 10, 100, 22);
for (var prop in this) {
   if (this[prop] instanceof TextField) {
   var this_txt:TextField = this[prop];
   trace(this_txt._name+" is a TextField at depth: "+this_txt.getDepth());
   }
}
```

この例で使用している `MovieClip.getNextHighestDepth()` メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、`MovieClip.getNextHighestDepth()` メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

getFontList (TextField.getFontList メソッド)

```
public static getFontList() : Array
```

Flash Player のホストシステム上のフォント名を配列として返します。現在ロードされている SWF ファイル内のすべてのフォント名を返すのではありません。これらの名前は `String` タイプです。このメソッドは、グローバル TextField クラスの静的メソッドです。このメソッドを呼び出す際には、テキストフィールドインスタンスを指定できません。

対応バージョン : ActionScript 1.0、Flash Player 6

戻り値

`Array` - フォント名の配列。

例

次のコードでは、getFontList() から返されるフォントの一覧を表示します。

```
var font_array:Array = TextField.getFontList();
font_array.sort();
trace("You have "+font_array.length+" fonts currently installed");
trace("---------------------------------------");
for (var i = 0; i<font_array.length; i++) {
  trace("Font #"+(i+1)+":\t"+font_array[i]);
}
```

getNewTextFormat (TextField.getNewTextFormat メソッド)

```
public getNewTextFormat() : TextFormat
```

テキストフィールドのテキストフォーマットオブジェクトのコピーを含む TextFormat オブジェクトを返します。テキストフォーマットオブジェクトは、新しく挿入するテキスト (replaceSel() メソッドで挿入したテキストまたはユーザーが入力したテキストなど) に適用されるフォーマットです。getNewTextFormat() を呼び出すと、すべてのプロパティが定義された TextFormat オブジェクトが返されます。null のプロパティはありません。

対応バージョン： ActionScript 1.0、Flash Player 6

戻り値

TextFormat - TextFormat オブジェクト。

例

次の例では、指定されたテキストフィールドの (my_txt) テキストフォーマットオブジェクトを表示します。

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 160, 120);
var my_fmt:TextFormat = my_txt.getNewTextFormat();
trace("TextFormat has the following properties:");
for (var prop in my_fmt) {
  trace(prop+": "+my_fmt[prop]);
}
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

getTextFormat (TextField.getTextFormat メソッド)

```
public getTextFormat([beginIndex:Number], [endIndex:Number]) : TextFormat
```

文字、文字の範囲または TextField オブジェクト全体に対して TextFormat オブジェクトを返します。

次の表に、使用できる 3 つのシンタックスを示します。

シンタックス	説明
`my_textField.getTextFormat()`	テキストフィールド内すべてのテキストに関するフォーマット情報を含む TextFormat オブジェクトを返します。テキストフィールド内のすべてのテキストに共通するプロパティのみが結果の TextFormat オブジェクトに設定されます。混在型のプロパティ (テキストに複数の異なる値が混在する場合) は、その値が `null`. に設定されます。
`my_textField.getTextFormat(beginIndex:Number)`	テキストフィールドで `beginIndex` の位置でのテキストフォーマットのコピーを含む TextFormat オブジェクトを返します。
`my_textField.getTextFormat(beginIndex:Number,endIndex:Number)`	`beginIndex` から `endIndex` までの範囲のテキストに関するフォーマット情報を含む TextFormat オブジェクトを返します。指定された範囲内のすべてのテキストに共通するプロパティのみが結果の TextFormat オブジェクトに設定されます。混在型のプロパティ (範囲内に複数の異なる値が混在する場合) は、その値が `null` に設定されます。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

`beginIndex:Number` - ストリング内の文字を指定する整数。`beginIndex` および `endIndex` を指定しない場合、返される TextFormat オブジェクトの対象は TextField 全体になります。

`endIndex:Number` (オプション) - テキスト範囲の終了位置を指定する整数。`beginIndex` を指定して `endIndex` を指定しない場合、返される TextFormat の対象は `beginIndex` で指定された 1 つの文字になります。

戻り値

`TextFormat` - 指定されたテキストのフォーマットプロパティを表す TextFormat オブジェクト。

例

次の ActionScript コードは、実行時に作成されたテキストフィールドの全フォーマット情報を表示します。

```
this.createTextField("dyn_txt", this.getNextHighestDepth(), 0, 0, 100, 200);
dyn_txt.text = "Frank";
dyn_txt.setTextFormat(new TextFormat());
var my_fmt:TextFormat = dyn_txt.getTextFormat();
for (var prop in my_fmt) {
  trace(prop+": "+my_fmt[prop]);
}
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

getNewTextFormat (TextField.getNewTextFormat メソッド), setNewTextFormat (TextField.setNewTextFormat メソッド), setTextFormat (TextField.setTextFormat メソッド)

gridFitType (TextField.gridFitType プロパティ)

public gridFitType : String

この TextField インスタンスに使用されるグリッド合わせのタイプです。このプロパティは、テキストフィールドの antiAliasType プロパティが "advanced" に設定されている場合にのみ適用されます。

gridFitType プロパティでは、次のストリング値を使用できます。

ストリング値	説明
"none"	グリッドフィッティングなしを指定します。文字の水平方向と垂直方向の線が、ピクセルグリッドに合わせられることはありません。通常、これはアニメーションや大きなフォントサイズに適した設定です。
"pixel"	太い水平線と垂直線がピクセルグリッドに合わされるように指定します。この設定は左揃えのテキストフィールドに対してのみ機能します。この設定を使用するには、テキストフィールドの antiAliasType プロパティを "advanced" に設定する必要があります。通常、この設定を使用すると、左揃えのテキストが最も読みやすくなります。
"subpixel"	太い水平線と垂直線が、LCD モニタのサブピクセルグリッドに合わされるよう指定します。この設定を使用するには、テキストフィールドの antiAliasType プロパティを "advanced" に設定する必要があります。"subpixel" 設定は、右揃えまたは中央揃えのダイナミックテキストに適した設定で、アニメーション品質とテキスト品質のバランスを取るのに便利です。

対応バージョン： ActionScript 1.0、Flash Player 8

例

この例では、それぞれ異なる gridFitType 設定を使用する 3 つのテキストフィールドを示します。リンケージ識別子が "Times-12" に設定されたライブラリにフォントが埋め込まれていることが前提です。フォントを埋め込むには、次の手順を実行します。

- ライブラリを開きます。
- ライブラリの右上隅にある [ライブラリ] オプションメニューをクリックします。
- ドロップダウンリストから [新しいフォント] を選択します。
- フォントに "Times-12" という名前を付けます。
- ドロップダウンリストから "Times New Roman" を選択します。
- [OK] ボタンをクリックします。
- 新しく作成したフォントを右クリックし、[リンケージ] を選択します。
- [ActionScript に書き出し] チェックボックスをオンにします。
- [OK] ボタンをクリックして、デフォルトの識別子 "Times-12" をそのまま使用します。

```
var my_format:TextFormat = new TextFormat();
my_format.font = "Times-12";

var my_text1:TextField = this.createTextField("my_text1",
  this.getNextHighestDepth(), 9.5, 10, 400, 100);
my_text1.text = "this.gridFitType = none";
my_text1.embedFonts = true;
my_text1.antiAliasType = "advanced";
my_text1.gridFitType = "none";
my_text1.setTextFormat(my_format);

var my_text2:TextField = this.createTextField("my_text2",
  this.getNextHighestDepth(), 9.5, 40, 400, 100);
my_text2.text = "this.gridFitType = advanced";
my_text2.embedFonts = true;
my_text2.antiAliasType = "advanced";
my_text2.gridFitType = "pixel";
my_text2.setTextFormat(my_format);

var my_text3:TextField = this.createTextField("my_text3",
  this.getNextHighestDepth(), 9.5, 70, 400, 100);
my_text3.text = "this.gridFitType = subpixel";
my_text3.embedFonts = true;
my_text3.antiAliasType = "advanced";
my_text3.gridFitType = "subpixel";
my_text3.setTextFormat(my_format);
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

関連項目

TextRenderer (flash.text.TextRenderer), antiAliasType (TextField.antiAliasType プロパティ), sharpness (TextField.sharpness プロパティ)

_height (TextField._height プロパティ)

public _height : Number

ピクセル単位で示したテキストフィールドの高さ。

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の例では、テキストフィールドの高さと幅のプロパティを設定します。

```
my_txt._width = 200;
my_txt._height = 200;
```

_highquality (TextField._highquality プロパティ)

public _highquality : Number

非推奨 Flash Player 7 以降では使用しないでください。このプロパティの代わりに TextField._quality を使用します。

現在の SWF ファイルに適用されるアンチエイリアスのレベルを指定します。ビットマップスムージングを常にオンにして最高品質を適用するには、2 (最高品質) を指定します。アンチエイリアス処理を適用するには、1 (高品質) を指定します。SWF ファイルにアニメーションが含まれない場合は、ビットマップは滑らかになります。これはデフォルト値です。アンチエイリアス処理を避けるには、0 (低品質) を指定します。

対応バージョン： ActionScript 1.0、Flash Player 6

関連項目

_quality (TextField._quality プロパティ)

hscroll (TextField.hscroll プロパティ)

public hscroll : Number

現在の水平スクロール位置を示します。hscroll プロパティが 0 である場合、テキストは水平にスクロールされません。

垂直スクロールの単位は行数ですが、水平スクロールの単位はピクセル数です。水平スクロールをピクセル単位で指定するのは、一般的に使用されるフォントのほとんどがプロポーショナルフォントであり、文字の幅が一定でないためです。垂直スクロールの場合は、通常、1 行のテキストの一部だけ表示されるよりも行全体が表示されることが好まれるため、行単位でスクロールします。1 行の中に複数のフォントが存在する場合でも、使用されている最大のフォントに合わせて行の高さが調整されます。

メモ : hscroll プロパティは、垂直スクロールプロパティ TextField.scroll のように 1 から始まるのではなく、0 から始まります。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、2 つのボタン scrollLeft_btn および scrollRight_btn を使用して、テキストフィールド my_txt を水平方向にスクロールします。スクロール量は、テキストフィールド scroll_txt に表示されます。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
this.createTextField("scroll_txt", this.getNextHighestDepth(), 10, 10, 160, 20);
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 30, 160, 22);
my_txt.border = true;
my_txt.multiline = false;
my_txt.wordWrap = false;
my_txt.text = "Lorem ipsum dolor sit amet, consectetuer adipiscing...";

scrollLeft_btn.onRelease = function() {
  my_txt.hscroll -= 10;
  scroll_txt.text = my_txt.hscroll+" of "+my_txt.maxhscroll;
};
scrollRight_btn.onRelease = function() {
  my_txt.hscroll += 10;
  scroll_txt.text = my_txt.hscroll+" of "+my_txt.maxhscroll;
};
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

maxhscroll (TextField.maxhscroll プロパティ),scroll (TextField.scroll プロパティ)

html (TextField.html プロパティ)

```
public html : Boolean
```

テキストフィールドに HTML 表現があるかどうかを示すフラグ。html プロパティが true である場合、テキストフィールドは HTML テキストフィールドです。html プロパティが false である場合、テキストフィールドは HTML 以外のテキストフィールドです。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、html プロパティを true に設定するテキストフィールドを作成します。このテキストフィールドには、HTML 形式のテキストが表示されます。

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 160, 22);
my_txt.html = true;
my_txt.htmlText = "<b> this is bold text </b>";
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

htmlText (TextField.htmlText プロパティ)

htmlText (TextField.htmlText プロパティ)

```
public htmlText : String
```

テキストフィールドが HTML テキストフィールドである場合、このプロパティにはテキストフィールドの内容の HTML 表現が入ります。テキストフィールドが HTML テキストフィールドではない場合、その動作は text プロパティと同じです。テキストフィールドを HTML テキストフィールドとして指定するには、プロパティインスペクタを使用するか、テキストフィールドの html プロパティを true に設定します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、html プロパティを true に設定するテキストフィールドを作成します。このテキストフィールドには、HTML 形式のテキストが表示されます。

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 160, 22);
my_txt.html = true;
my_txt.htmlText = "<b> this is bold text </b>";
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

html (TextField.html プロパティ), asfunction プロトコル

length (TextField.length プロパティ)

public length : Number (読み取り専用)

テキストフィールド内の文字数を示します。このプロパティは、text.length と同じ値をより速く返します。タブ (\t) などの文字も 1 文字としてカウントされます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、現在の日付を表示するテキストフィールド date_txt 内の文字数を出力します。

```
var today:Date = new Date();
this.createTextField("date_txt", this.getNextHighestDepth(), 10, 10, 100, 22);
date_txt.autoSize = true;
date_txt.text = today.toString();
trace(date_txt.length);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

maxChars (TextField.maxChars プロパティ)

public maxChars : Number

テキストフィールドに入る最大の文字数を示します。スクリプトは maxChars の許容数を超えるテキストを挿入できます。maxChars プロパティで指定されるのは、ユーザーが入力できるテキストの量だけです。このプロパティの値が null である場合、ユーザーが入力できるテキストの量には制限がありません。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、フィールド内にユーザーが 2 文字まで入力できるテキストフィールド age_txt を作成します。

```
this.createTextField("age_txt", this.getNextHighestDepth(), 10, 10, 30, 22);
age_txt.type = "input";
age_txt.border = true;
age_txt.maxChars = 2;
age_txt.restrict = "0-9";
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

maxhscroll (TextField.maxhscroll プロパティ)

public maxhscroll : Number (読み取り専用)

TextField.hscroll の最大値を示します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

TextField.hscroll の例を参照してください。

maxscroll (TextField.maxscroll プロパティ)

public maxscroll : Number (読み取り専用)

TextField.scroll の最大値を示します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、スクロールテキストフィールド my_txt の最大値を設定します。テキストフィールドをスクロールする 2 つのボタン、scrollUp_btn および scrollDown_btn を作成します。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
this.createTextField("scroll_txt", this.getNextHighestDepth(), 10, 10, 160, 20);
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 30, 320, 240);
my_txt.multiline = true;
my_txt.wordWrap = true;
for (var i = 0; i<10; i++) {
  my_txt.text += "Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed
  diam nonummy nibh "
       + "euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.";
}
```

```
scrollUp_btn.onRelease = function() {
  my_txt.scroll--;
  scroll_txt.text = my_txt.scroll+" of "+my_txt.maxscroll;
};
scrollDown_btn.onRelease = function() {
  my_txt.scroll++;
  scroll_txt.text = my_txt.scroll+" of "+my_txt.maxscroll;
};
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

menu (TextField.menu プロパティ)

```
public menu : ContextMenu
```

contextMenu に指定した ContextMenu オブジェクトをテキストフィールド *my_txt* に関連付けます。ContextMenu クラスを使用すると、ユーザーが Flash Player 内を右クリック (Windows) または Control キーを押しながらクリック (Macintosh) したときに表示されるコンテキストメニューを修正することができます。

このプロパティを使用できるのは、選択可能 (編集可能) なテキストフィールドだけです。選択不可能なテキストフィールドに対しては効果がありません。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例では、ContextMenu オブジェクト menu_cm をテキストフィールド news_txt に割り当てます。この ContextMenu オブジェクトには、"Resize" というラベルのカスタムメニュー項目があり、サイズ変更機能の追加に使用できる doResize() というコールバックハンドラが関連付けられています。ハンドラはここでは示していません。

```
this.createTextField("news_txt", this.getNextHighestDepth(), 10, 10, 320, 240);
news_txt.border = true;
news_txt.wordWrap = true;
news_txt.multiline = true;
news_txt.text = "To see the custom context menu item, right click (PC) or ";
news_txt.text += "control click (Mac) within the text field.";
var menu_cm:ContextMenu = new ContextMenu();
menu_cm.customItems.push(new ContextMenuItem("Resize", doResize));

function doResize(obj:TextField, item:ContextMenuItem):Void {
  // "Resize" code here
  trace("you selected: "+item.caption);
}
news_txt.menu = menu_cm;
```

テキストフィールド領域内を右クリック (Windows の場合) または Control キーを押しながらクリック (Macintosh の場合) すると、カスタムメニュー項目が表示されます。

メモ : Flash で既に使用されているメニューアイテムは使用できません。たとえば、`Print...` (ドットが 3 つ) は Flash で予約されているため、このメニューアイテムは使用できません。しかし、`Print..` (ドットが 2 つ) などのように、Flash で使用されていないメニューアイテムは使用できます。

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している `MovieClip.getNextHighestDepth()` メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

関連項目

`Button`, `ContextMenu`, `ContextMenuItem`, `MovieClip`

mouseWheelEnabled (TextField.mouseWheelEnabled プロパティ)

```
public mouseWheelEnabled : Boolean
```

複数行のテキストフィールドで、マウスポインタがテキストフィールドをクリックしユーザーがホイールを回転させると、自動的にスクロールするかどうかを示すブール値です。デフォルト値は `true` です。このプロパティは、マウスホイールでテキストフィールドをスクロールしない場合や、テキストフィールドのスクロールを独自に実装する場合に便利です。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例では、2 つのテキストフィールドを作成します。`scrollable_txt` フィールドの `mouseWheelEnabled` プロパティが true に設定され、フィールドをクリックしてマウスホイールを回転させると `scrollable_txt` がスクロールします。`nonscrollable_txt` フィールドは、フィールドをクリックしてマウスホイールを回転してもスクロールしません。

```
var font_array:Array = TextField.getFontList().sort();

this.createTextField("scrollable_txt", this.getNextHighestDepth(), 10, 10, 240,
  320);
scrollable_txt.border = true;
scrollable_txt.wordWrap = true;
scrollable_txt.multiline = true;
scrollable_txt.text = font_array.join("\n");

this.createTextField("nonscrollable_txt", this.getNextHighestDepth(), 260, 10,
  240, 320);
nonscrollable_txt.border = true;
nonscrollable_txt.wordWrap = true;
```

```
nonscrollable_txt.multiline = true;
nonscrollable_txt.mouseWheelEnabled = false;
nonscrollable_txt.text = font_array.join("\n");
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

関連項目

mouseWheelEnabled (TextField.mouseWheelEnabled プロパティ)

multiline (TextField.multiline プロパティ)

public multiline : Boolean

テキストフィールドが複数行テキストフィールドであるかどうかを示します。値が true である場合は複数行テキストフィールド、値が false である場合は単一行テキストフィールドです。

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の例では、2 つのテキストフィールドを作成します。最初のテキストフィールドの multiline プロパティを true に設定し、2 番目のテキストフィールドの multiline プロパティを false に設定します。ファイルを実行する際、最初のテキストフィールドで ENTER キーを押すのと 2 番目のテキストフィールドで ENTER キーを押すのでは違いがあります。

```
this.createTextField("input1", this.getNextHighestDepth(), 10, 10, 240, 320);
input1.border = true;
input1.type = "input";
input1.multiline = true;
input1.text = "Type some text and press the ENTER key: ";

this.createTextField("input2", this.getNextHighestDepth(), 10, 10, 240, 320);
input2.border = true;
input2._x = 275;
input2.type = "input";
input2.multiline = false;
input2.text = "Type some text and press the ENTER key: ";
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

_name (TextField._name プロパティ)

```
public _name : String
```

テキストフィールドのインスタンス名。

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の例は、それぞれ異なる深度にあるテキストフィールドを示しています。ステージ上にダイナミックテキストフィールドを作成します。次の ActionScript を FLA ファイルまたは AS ファイルに追加すると、実行時に 2 つのテキストフィールドが動的に作成され、それぞれの深度が [出力] パネルに表示されます。

```
this.createTextField("first_mc", this.getNextHighestDepth(), 10, 10, 100, 22);
this.createTextField("second_mc", this.getNextHighestDepth(), 10, 10, 100, 22);
for (var prop in this) {
  if (this[prop] instanceof TextField) {
  var this_txt:TextField = this[prop];
  trace(this_txt._name+" is a TextField at depth: "+this_txt.getDepth());
  }
}
```

ドキュメントをテストするときには、インスタンス名と深度が [出力] パネルに表示されます。

この例で使用している `MovieClip.getNextHighestDepth()` メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、`MovieClip.getNextHighestDepth()` メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

onChanged (TextField.onChanged ハンドラ)

```
onChanged = function(changedField:TextField) {}
```

イベントハンドラ / リスナー。テキストフィールドの内容が変更されると、呼び出されます。デフォルトは undefined です。スクリプトで定義できます。

`onChanged` ハンドラには、パラメータとしてテキストフィールドインスタンスへの参照が渡されます。イベントハンドラメソッドにパラメータを含めることによって、このデータを受け取ることができます。たとえば、次の例では、`onChanged` イベントハンドラで `textfield_txt` パラメータを受け取ります。その後、`trace()` ステートメントで、このパラメータを使用して、テキストフィールドのインスタンス名を [出力] パネルに表示します。

```
this.createTextField("myInputText_txt", 99, 10, 10, 300, 20);
myInputText_txt.border = true;
myInputText_txt.type = "input";
```

```
myInputText_txt.onChanged = function(textfield_txt:TextField) {
trace("the value of "+textfield_txt._name+" was changed. New value is:
   "+textfield_txt.text);
};
```

onChanged ハンドラは、ユーザーの操作によって変更が生じた場合にのみ呼び出されます。たとえば、ユーザーがキーボードで入力した場合や、マウスを使用してテキストフィールドの内容を変更した場合、メニューアイテムを選択した場合などです。プログラムによってテキストフィールドが変更されても、テキストフィールドに加えられる変更はコードから識別できるため、onChanged イベントはトリガされません。

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

changedField: TextField - イベントをトリガするフィールド。

関連項目

TextFormat, setNewTextFormat （TextField.setNewTextFormat メソッド）

onKillFocus (TextField.onKillFocus ハンドラ)

```
onKillFocus = function(newFocus:Object) {}
```

テキストフィールドがキーボードフォーカスを失うと、呼び出されます。onKillFocus メソッドは、1 つのパラメータ newFocus を受け取ります。このパラメータは、フォーカスを受け取る新しいオブジェクトを表すオブジェクトです。フォーカスを受け取るオブジェクトがない場合、newFocus の値は null です。

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

newFocus: Object - フォーカスを受け取るオブジェクト。

例

次の例では、2 つのテキストフィールド first_txt および second_txt を作成します。1 つのテキストフィールドにフォーカスを置くと、フォーカスを置かれたテキストフィールドに関する情報と、フォーカスを失ったテキストフィールドに関する情報が、[出力] パネルに表示されます。

```
this.createTextField("first_txt", 1, 10, 10, 300, 20);
first_txt.border = true;
first_txt.type = "input";
this.createTextField("second_txt", 2, 10, 40, 300, 20);
second_txt.border = true;
second_txt.type = "input";
```

```
first_txt.onKillFocus = function(newFocus:Object) {
  trace(this._name+" lost focus. New focus changed to: "+newFocus._name);
};
first_txt.onSetFocus = function(oldFocus:Object) {
  trace(this._name+" gained focus. Old focus changed from: "+oldFocus._name);
}
```

関連項目

onSetFocus (TextField.onSetFocus ハンドラ)

onScroller (TextField.onScroller ハンドラ)

```
onScroller = function(scrolledField:TextField) {}
```

イベントハンドラ / リスナー。テキストフィールドのスクロールプロパティが変わると呼び出されます。

onScroller ハンドラには、パラメータとしてテキストフィールドインスタンスへの参照が渡されます。イベントハンドラメソッドにパラメータを含めることによって、このデータを受け取ることができます。たとえば、次の例では、onScroller イベントハンドラで my_txt パラメータを受け取ります。その後、trace( ) ステートメントでこのパラメータを使用して、テキストフィールドのインスタンス名を [出力] パネルに表示します。

```
myTextField.onScroller = function (my_txt:TextField) {
  trace (my_txt._name + " scrolled");
};
```

TextField.onScroller イベントハンドラは、一般的にはスクロールバーを実装するために使用されます。通常スクロールバーには、テキストフィールドにおける現在の水平スクロール位置または垂直スクロール位置を示すサムなどのインジケータがあります。テキストフィールドはマウスとキーボードを使用して操作し、スクロール位置を変更することができます。このようなユーザー操作によってスクロール位置が変更されたかどうかを、スクロールバーのコードに通知する必要があります。そのために TextField.onScroller を使用します。

onScroller は、ユーザーによるテキストフィールドの操作、またはプログラムによって、スクロール位置が変更された場合に呼び出されます。onChanged ハンドラは、ユーザーの操作によって変更が生じた場合にのみ発行されます。コードの一部によってスクロール位置が変更されても、スクロールバーのコードに関連付けがない場合、通知がないとスクロールバーのコードではスクロール位置の変更が認識されないため、この 2 つのオプションが必要になります。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

scrolledField:TextField - スクロール位置が変更された TextField オブジェクトへの参照。

例

次の例では、テキストフィールド my_txt を作成し、2 つのボタン、scrollUp_btn および scrollDown_btn を使用して、このテキストフィールドのコンテンツをスクロールします。onScroller イベントハンドラが呼び出されると、trace ステートメントを使用して [出力] パネルに情報を表示します。インスタンス名 scrollUp_btn および scrollDown_btn を付けて 2 つのボタンを作成し、次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
this.createTextField("scroll_txt", this.getNextHighestDepth(), 10, 10, 160, 20);
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 30, 320, 240);
my_txt.multiline = true;
my_txt.wordWrap = true;

for (var i = 0; i<10; i++) {
  my_txt.text += "Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed
  diam "
    + "nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat
  volutpat.";
}
scrollUp_btn.onRelease = function() {
  my_txt.scroll--;
};
scrollDown_btn.onRelease = function() {
  my_txt.scroll++;
};
my_txt.onScroller = function() {
    trace("onScroller called");
  scroll_txt.text = my_txt.scroll+" of "+my_txt.maxscroll;
};
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

hscroll (TextField.hscroll プロパティ), maxhscroll (TextField.maxhscroll プロパティ), maxscroll (TextField.maxscroll プロパティ), scroll (TextField.scroll プロパティ)

onSetFocus (TextField.onSetFocus ハンドラ)

```
onSetFocus = function(oldFocus:Object) {}
```

テキストフィールドがキーボードフォーカスを受け取ると、呼び出されます。oldFocus パラメータは、フォーカスを失うオブジェクトです。たとえば、Tab キーを押して入力フォーカスをボタンからテキストフィールドに移動すると、oldFocus にボタンインスタンスが入ります。前にフォーカスがあったオブジェクトが存在しない場合、oldFocus には null 値が入ります。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

oldFocus:Object - キーボードフォーカスを失うオブジェクト。

例

TextField.onKillFocus の例を参照してください。

関連項目

onKillFocus (TextField.onKillFocus ハンドラ)

_parent (TextField._parent プロパティ)

```
public _parent : MovieClip
```

現在のテキストフィールドまたはオブジェクトを含むムービークリップまたはオブジェクトへの参照。現在のオブジェクトとは、_parent を参照する ActionScript コードがあるオブジェクトです。

現在のテキストフィールドよりも上位のムービークリップまたはオブジェクトへの相対パスを指定するには、_parent を使用します。_parent を使用して表示リストの複数のレベルを上位に移動するには、次のようにします。

```
_parent._parent._alpha = 20;
```

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の ActionScript は、2 つのテキストフィールドを作成し、各オブジェクトの _parent に関する情報を出力します。最初のテキストフィールド first_txt は、メインタイムラインに作成されます。2 番目のテキストフィールド second_txt は、ムービークリップ holder_mc 内に作成されます。

```
this.createTextField("first_txt", this.getNextHighestDepth(), 10, 10, 160, 22);
first_txt.border = true;
trace(first_txt._name+"'s _parent is: "+first_txt._parent);

this.createEmptyMovieClip("holder_mc", this.getNextHighestDepth());
```

```
holder_mc.createTextField("second_txt", holder_mc.getNextHighestDepth(), 10, 40,
  160, 22);
holder_mc.second_txt.border = true;
trace(holder_mc.second_txt._name+"'s _parent is:
  "+holder_mc.second_txt._parent);
```

次の情報が [出力] パネルに表示されます。

```
first_txt's _parent is: _level0
second_txt's _parent is: _level0.holder_mc
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

_parent (Button._parent プロパティ), _parent (MovieClip._parent プロパティ), _root プロパティ , targetPath 関数

password (TextField.password プロパティ)

```
public password : Boolean
```

テキストフィールドがパスワードテキストフィールドであるかどうかを指定します。password の値が true である場合、テキストフィールドはパスワードテキストフィールドであり、入力された文字はアスタリスクで隠されます。false である場合、テキストフィールドはパスワードテキストフィールドではありません。パスワードモードを有効にすると、[カット] コマンドと [コピー] コマンド、およびそれに対応するキーボードショートカットが機能しなくなります。このセキュリティ機能により、ユーザーの不在時にキーボードショートカットを使用して悪質なユーザーがパスワードが盗むことを防止できます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、2 つのテキストフィールド username_txt および password_txt を作成します。両方のテキストフィールドにテキストが入力されますが、password_txt では password プロパティが true に設定されています。このため、password_txt に入力した文字はアスタリスクで表示されます。

```
this.createTextField("username_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
username_txt.border = true;
username_txt.type = "input";
username_txt.maxChars = 16;
username_txt.text = "hello";
```

```
this.createTextField("password_txt", this.getNextHighestDepth(), 10, 40, 100,
  22);
password_txt.border = true;
password_txt.type = "input";
password_txt.maxChars = 16;
password_txt.password = true;
password_txt.text = "world";
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

_quality (TextField._quality プロパティ)

```
public _quality : String
```

SWF ファイルに使用するレンダリング品質。デバイスフォントは常にエイリアス処理されるため、_quality プロパティには影響されません。

メモ：このプロパティを TextField オブジェクトに対して指定することもできますが、実際にはグローバルプロパティであるので、単に _quality という形で値を指定することもできます。詳細については、_quality グローバルプロパティを参照してください。

_quality プロパティは、次の値に設定できます。

- "LOW" 低いレンダリング品質。グラフィックスはアンチエイリアス処理されず、ビットマップはスムージングされません。
- "MEDIUM" 普通のレンダリング品質。グラフィックスは 2 x 2 ピクセルグリッドを使用してアンチエイリアス処理されますが、ビットマップはスムージングされません。これは、テキストを含まないムービーに適しています。
- "HIGH" 高いレンダリング品質。グラフィックスは 4x4 ピクセルグリッドを使用してアンチエイリアス処理されます。ビットマップは、ムービーが静的なものである場合は、スムージングされます。デフォルトのレンダリング品質設定です。
- "BEST" 非常に高いレンダリング品質。グラフィックスは 4x4 ピクセルグリッドを使用してアンチエイリアス処理され、ビットマップは常にスムージングされます。

対応バージョン：ActionScript 1.0、Flash Player 6

例

次の例では、レンダリング品質を LOW に設定します。

```
my_txt._quality = "LOW";
```

関連項目

_quality プロパティ

removeListener (TextField.removeListener メソッド)

```
public removeListener(listener:Object) : Boolean
```

TextField.addListener() でテキストフィールドインスタンスに以前に登録したリスナーオブジェクトを削除します。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

listener:Object - TextField.onChanged または TextField.onScroller から通知を受け取らなくなるオブジェクト。

戻り値

Boolean - listener が正常に削除された場合は true を返します。listener が正常に削除されなかった場合 (listener が TextField オブジェクトのリスナーリストにない場合など) は false を返します。

例

次の例では、テキスト入力フィールド my_txt を作成します。ユーザーがこのテキストフィールドに入力すると、テキストフィールド内の文字数に関する情報が [出力] パネルに表示されます。ユーザーが removeListener_btn インスタンスをクリックした場合は、リスナーが削除され、情報が表示されなくなります。

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 160, 20);
my_txt.border = true;
my_txt.type = "input";

var txtListener:Object = new Object();
txtListener.onChanged = function(textfield_txt:TextField) {
  trace(textfield_txt+" changed. Current length is: "+textfield_txt.length);
};
my_txt.addListener(txtListener);

removeListener_btn.onRelease = function() {
  trace("Removing listener...");
  if (!my_txt.removeListener(txtListener)) {
  trace("Error! Unable to remove listener");
  }
};
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

removeTextField (TextField.removeTextField メソッド)

```
public removeTextField() : Void
```

テキストフィールドを削除します。この操作を実行できるのは、MovieClip.createTextField() で作成したテキストフィールドだけです。このメソッドを呼び出すと、テキストフィールドは削除されます。このメソッドは MovieClip.removeMovieClip() と似ています。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、remove_btn インスタンスをクリックしてステージから削除できるテキストフィールドを作成します。remove_btn という名前を付けてボタンを作成し、次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 300, 22);
my_txt.text = new Date().toString();
my_txt.border = true;

remove_btn.onRelease = function() {
  my_txt.removeTextField();
};
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

replaceSel (TextField.replaceSel メソッド)

```
public replaceSel(newText:String) : Void
```

現在の選択内容を newText パラメータの内容に置き換えます。テキストは、現在のデフォルトの文字フォーマットとデフォルトの段落フォーマットを使用して、現在の選択内容の位置に挿入されます。テキストフィールドが HTML テキストフィールドである場合でも、テキストは HTML として扱われません。

replaceSel() メソッドを使用すると、他の部分のテキストの文字フォーマットおよび段落フォーマットを損なわずにテキストを挿入および削除できます。

このコマンドを発行する前に、Selection.setFocus() を使用してフィールドにフォーカスを合わせる必要があります。

メモ : このメソッドは、スタイルシートをテキストフィールドに適用する場合には動作しません。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

`newText:String` - ストリング。

例

次のコード例では、テキストが設定された複数行のテキストフィールドをステージ上に作成します。テキストの一部を選択し、テキストフィールドの上で右クリック (Windows の場合) または Control キーを押しながらクリック (Macintosh の場合) すると、コンテキストメニューから "Enter current date" を選択できます。これにより、選択したテキストを現在の日付に置き換える関数が呼び出されます。

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 320, 240);
my_txt.border = true;
my_txt.wordWrap = true;
my_txt.multiline = true;
my_txt.type = "input";
my_txt.text = "Select some sample text from the text field and then right-click/
  control click "
    + "and select 'Enter current date' from the context menu to replace the "
    + "currently selected text with the current date.";

var my_cm:ContextMenu = new ContextMenu();
my_cm.customItems.push(new ContextMenuItem("Enter current date", enterDate));
function enterDate(obj:Object, menuItem:ContextMenuItem) {
  var today_str:String = new Date().toString();
  var date_str:String = today_str.split(" ", 3).join(" ");
  my_txt.replaceSel(date_str);
}
my_txt.menu = my_cm;
```

この例で使用している `MovieClip.getNextHighestDepth()` メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、`MovieClip.getNextHighestDepth()` メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

`setFocus (Selection.setFocus メソッド )`

replaceText (TextField.replaceText メソッド)

```
public replaceText(beginIndex:Number, endIndex:Number, newText:String) : Void
```

指定されたテキストフィールドの beginIndex パラメータと endIndex パラメータで指定した文字範囲を、newText パラメータの内容に置き換えます。

メモ : このメソッドは、スタイルシートをテキストフィールドに適用する場合には動作しません。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

beginIndex:Number - 置換範囲の開始インデックスの値。

endIndex:Number - 置換範囲の終了インデックスの値。

newText:String - 指定された文字範囲の置き換えに使用されるテキスト。

例

次に例では、my_txt というテキストフィールドを作成し、テキスト dog@house.net をこのフィールドに割り当てます。indexOf() メソッドを使用して、指定されたシンボル (@) が最初に出現する場所を探します。このシンボルが見つかると、指定されたテキスト (インデックス 0 とシンボルの間) がストリング bird に置き換わります。このシンボルが見つからなかった場合は、[出力] パネルにエラーメッセージが表示されます。

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 320, 22);
my_txt.autoSize = true;
my_txt.text = "dog@house.net";

var symbol:String = "@";
var symbolPos:Number = my_txt.text.indexOf(symbol);
if (symbolPos>-1) {
  my_txt.replaceText(0, symbolPos, "bird");
} else {
  trace("symbol '"+symbol+"' not found.");
}
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

restrict (TextField.restrict プロパティ)

```
public restrict : String
```

ユーザーがテキストフィールドに入力できる文字のセットを示します。restrict プロパティの値が null である場合は、任意の文字を入力できます。restrict プロパティの値が空のストリングである場合は、いずれの文字も入力できません。restrict プロパティの値が文字のストリングである場合は、そのストリング内の文字のみをテキストフィールドに入力できます。ストリングは左から右へスキャンされます。範囲はダッシュ (-) を使用して指定できます。これはユーザーの操作のみを制限します。スクリプトは任意のテキストをテキストフィールドに入力できます。このプロパティはプロパティインスペクタの [フォントのアウトラインの埋め込み] チェックボックスと同期しません。

^ でストリングが始まる場合、そのストリングに含まれる文字を除いて、すべての文字を入力できます。ストリングが ^ 以外で始まる場合は、そのストリングに含まれる文字のみを入力できます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、大文字、スペース、および数値のみをテキストフィールドに入力できます。

```
my_txt.restrict = "A-Z 0-9";
```

次の例では、小文字を除くすべての文字を入力できます。

```
my_txt.restrict = "^a-z";
```

円記号を使用して ^ または - のリテラルを入力できます。許容される円記号のシーケンスは \-、\^、または \\ です。円記号をストリングの 1 文字として実際に使用するには、ActionScript で指定するときに円記号を 2 つ続けて指定する必要があります。たとえば、次のコードで表されているのはダッシュ (-) とキャレット (^) のみです。

```
my_txt.restrict = "\\-\\^";
```

ストリング内では任意の場所で ^ を使用し、入力できる文字と除外する文字を切り替えることができます。次のコードでは、大文字のみを入力できます。ただし、大文字の Q を除きます。

```
my_txt.restrict = "A-Z^Q";
```

\u エスケープシーケンスを使用して、restrict ストリングを構築できます。次のコードでは、ASCII 32 (スペース) から ASCII 126 (チルダ) までの文字のみを入力できます。

```
my_txt.restrict = "\u0020-\u007E";
```

_rotation (TextField._rotation プロパティ)

public _rotation : Number

テキストフィールドの元の位置からの回転角を度単位で指定します。時計回りに回転させる場合は 0 ～ 180 の値を指定します。反時計回りに回転させる場合は 0 ～ -180 の値を指定します。この範囲を超える値は、360 に加算または 360 から減算され、範囲内に収まる値になるように調整されます。たとえば、my_txt._rotation = 450 というステートメントは my_txt._rotation = 90 と同義です。

デバイスフォントを使用しているテキストフィールドに対する回転値の使用はサポートされていません。テキストフィールドで _rotation を使用するには、埋め込みフォントを使用する必要があります。

対応バージョン : ActionScript 1.0、Flash Player 6

例

この例では、ダイナミックテキストフィールド my_txt を作成してから、次の ActionScript を使用してフォントの埋め込みとテキストフィールドの回転を行う必要があります。my font への参照は、リンケージを my font に設定して、ライブラリ内のフォントシンボルを参照します。

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.font = "my font";

this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 160, 120);
my_txt.wordWrap = true;
my_txt.embedFonts = true;
my_txt.text = "Hello world";
my_txt.setTextFormat(my_fmt);
my_txt._rotation = 45;
```

テキストフィールドにその他のフォーマットを適用するには、TextFormat class を使用します。

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

_rotation (Button._rotation プロパティ), _rotation (MovieClip._rotation プロパティ), TextFormat

scroll (TextField.scroll プロパティ)

```
public scroll : Number
```

テキストフィールドのテキストの垂直座標です。scroll プロパティは、長い文節内の特定の段落にユーザーを誘導したり、スクロールテキストフィールドを作成したりする場合に便利です。このプロパティは、取得および変更が可能です。

垂直スクロールの単位は行数であり、水平スクロールの単位はピクセル数です。水平スクロールをピクセル単位で指定するのは、一般的に使用されるフォントのほとんどがプロポーショナルフォントであり、文字の幅が一定でないためです。垂直スクロールの場合は、通常、1 行のテキストの一部だけ表示されるよりも行全体が表示されることが好まれるため、行単位でスクロールします。1 行の中に複数のフォントが存在する場合でも、使用されている最大のフォントに合わせて行の高さが調整されます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、スクロールテキストフィールド my_txt の最大値を設定します。テキストフィールドをスクロールする 2 つのボタン、scrollUp_btn および scrollDown_btn を作成します。次の ActionScript コードを FLA ファイルまたは AS ファイルに追加します。

```
this.createTextField("scroll_txt", this.getNextHighestDepth(), 10, 10, 160, 20);
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 30, 320, 240);
my_txt.multiline = true;
my_txt.wordWrap = true;
for (var i = 0; i<10; i++) {
  my_txt.text += "Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed
  diam nonummy "
    + "nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.";
}
scrollUp_btn.onRelease = function() {
  my_txt.scroll--;
  scroll_txt.text = my_txt.scroll+" of "+my_txt.maxscroll;
};
scrollDown_btn.onRelease = function() {
  my_txt.scroll++;
  scroll_txt.text = my_txt.scroll+" of "+my_txt.maxscroll;
};
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

hscroll (TextField.hscroll プロパティ),maxscroll (TextField.maxscroll プロパティ)

selectable (TextField.selectable プロパティ)

public selectable : Boolean

テキストフィールドが選択可能であるかどうかを示すブール値です。値 true は、テキストが選択可能であることを示します。selectable プロパティを使用することで、テキストフィールドが選択可能かどうかを指定できますが、編集可能かどうかは指定できません。ダイナミックテキストフィールドは、編集可能でない場合でも選択可能にすることができます。ダイナミックテキストフィールドが選択可能でないということは、フィールド内のテキストを選択できないことを意味します。

selectable を false に設定すると、テキストフィールド内のテキストはマウスやキーボードからの選択コマンドに応答しなくなり、[コピー] コマンドを使用してテキストをコピーすることができなくなります。selectable を true に設定すると、テキストフィールド内のテキストはマウスやキーボードを使用して選択できるようになります。テキスト入力フィールドではなくダイナミックテキストフィールドの場合でも、この方法でテキストを選択できます。テキストは、[コピー] コマンドを使用してコピーすることができます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、現在の日付と時刻を反映して常に更新される、選択可能なテキストフィールドを作成します。

```
this.createTextField("date_txt", this.getNextHighestDepth(), 10, 10, 100, 22);
date_txt.autoSize = true;
date_txt.selectable = true;

var date_interval:Number = setInterval(updateTime, 500, date_txt);
function updateTime(my_txt:TextField) {
  my_txt.text = new Date().toString();
}
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

setNewTextFormat (TextField.setNewTextFormat メソッド)

```
public setNewTextFormat(tf:TextFormat) : Void
```

テキストフィールドに新しいデフォルトのテキストフォーマットを設定します。新しいデフォルトのテキストフォーマットは、新しく挿入するテキスト (replaceSel() メソッドで挿入したテキストまたはユーザーが入力したテキストなど) に使用される新しいテキストフォーマットです。テキストを挿入すると、新しく挿入されたテキストにその新しいデフォルトのテキストフォーマットが割り当てられます。

新しいデフォルトのテキストフォーマットは、TextFormat オブジェクトである textFormat を使用して指定します。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

tf:TextFormat - TextFormat オブジェクト。

例

次の例では、実行時に新しいテキストフィールド (my_txt) を作成し、いくつかのプロパティを設定します。新しく挿入されたテキストのフォーマットが適用されます。

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.bold = true;
my_fmt.font = "Arial";
my_fmt.color = 0xFF9900;

this.createTextField("my_txt", 999, 0, 0, 400, 300);
my_txt.wordWrap = true;
my_txt.multiline = true;
my_txt.border = true;
my_txt.type = "input";
my_txt.setNewTextFormat(my_fmt);
my_txt.text = "Oranges are a good source of vitamin C";
```

関連項目

getNewTextFormat (TextField.getNewTextFormat メソッド), getTextFormat (TextField.getTextFormat メソッド), setTextFormat (TextField.setTextFormat メソッド)

setTextFormat (TextField.setTextFormat メソッド)

```
public setTextFormat([beginIndex:Number], [endIndex:Number],
  textFormat:TextFormat) : Void
```

textFormat パラメータで指定したテキストフォーマットを、テキストフィールド内のテキストの一部またはすべてに適用します。textFormat は、必要なテキストフォーマットの変更を指定する TextFormat オブジェクトである必要があります。textFormat の null 以外のプロパティのみがテキストフィールドに適用されます。textFormat で null に設定されているプロパティは適用されません。デフォルトで、新しく作成された TextFormat オブジェクトのプロパティはすべて null に設定されます。

メモ : このメソッドは、スタイルシートをテキストフィールドに適用する場合には動作しません。

TextFormat オブジェクトのフォーマット情報には、文字レベルフォーマットと段落レベルフォーマットの 2 種類があります。テキストフィールド内の各文字にも、フォント名やフォントサイズ、ボールド、イタリックなどの文字固有のフォーマット設定があります。

段落の場合は、段落の最初の文字を調べて段落全体のフォーマット設定を決定します。段落のフォーマット設定には、左マージン、右マージン、インデントなどがあります。

setTextFormat() メソッドは、テキストフィールドの各文字、文字の範囲、またはテキスト全体に適用するテキストフォーマットを変更します。これらのシンタックスを次の表に示します。

シンタックス	説明
my_textField.setTextFormat(textFormat: TextFormat)	textFormat のプロパティをテキストフィールドのすべてのテキストに適用します。
my_textField.setTextFormat(beginIndex: Number, textFormat:TextFormat)	textFormat のプロパティを beginIndex 位置の文字に適用します。
my_textField.setTextFormat(beginIndex: Number, endIndex:Number, textFormat:TextFormat)	textFormat パラメータのプロパティを beginIndex 位置から endIndex 位置までのテキスト範囲に適用します。

ユーザーによって手作業で挿入されたテキスト、または TextField.replaceSel() を使用して置き換えられたテキストは、テキストの挿入か所に指定されているフォーマットではなく、新しいテキスト用のテキストフィールドのデフォルトフォーマットが適用されます。新しいテキスト用のテキストフィールドのデフォルトフォーマットを設定するには、TextField.setNewTextFormat() を使用します。

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

beginIndex:Number (オプション) - 該当するテキスト範囲の最初の文字を指定する整数。beginIndex および endIndex を指定しない場合、TextFormat は TextField 全体に適用されます。

endIndex:Number (オプション) - 該当するテキスト範囲の直後の文字を指定する整数。beginIndex を指定して endIndex を指定しない場合、TextFormat は beginIndex で指定された 1 つの文字に適用されます。

textFormat:TextFormat - 文字と段落のフォーマット情報を含む TextFormat オブジェクト。

例

次の例では、テキストの 2 つの異なるストリングにテキストフォーマットを設定します。setTextFormat() メソッドを呼び出し、my_txt テキストフィールドに適用します。

```
var format1_fmt:TextFormat = new TextFormat();
format1_fmt.font = "Arial";
var format2_fmt:TextFormat = new TextFormat();
format2_fmt.font = "Courier";

var string1:String = "Sample string number one."+newline;
var string2:String = "Sample string number two."+newline;

this.createTextField("my_txt", this.getNextHighestDepth(), 0, 0, 300, 200);
my_txt.multiline = true;
my_txt.wordWrap = true;
my_txt.text = string1;
var firstIndex:Number = my_txt.length;
my_txt.text += string2;
var secondIndex:Number = my_txt.length;

my_txt.setTextFormat(0, firstIndex, format1_fmt);
my_txt.setTextFormat(firstIndex, secondIndex, format2_fmt);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

TextFormat, setNewTextFormat (TextField.setNewTextFormat メソッド)

sharpness (TextField.sharpness プロパティ)

```
public sharpness : Number
```

この TextField インスタンス内の文字エッジのシャープネスです。このプロパティは、テキストフィールドの antiAliasType プロパティが "advanced" に設定されている場合にのみ適用されます。sharpness の範囲は -400 ～ 400 の数値です。sharpness をこの範囲外の値に設定しようとすると、値は範囲内の最も近い値 (-400 または 400) に設定されます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

この例では、sharpness がそれぞれ 400、0、および -400 に設定された 3 つのテキストフィールドを作成します。リンケージ識別子が "Times-12" に設定されたライブラリにフォントが埋め込まれていることが前提です。フォントを埋め込むには、次の手順を実行します。

- ライブラリを開きます。
- ライブラリの右上隅にある [ライブラリ] オプションメニューをクリックします。
- ドロップダウンリストから [新しいフォント] を選択します。
- フォントに "Times-12" という名前を付けます。
- ドロップダウンリストから "Times New Roman" を選択します。
- [OK] ボタンをクリックします。
- 新しく作成したフォントを右クリックし、[リンケージ] を選択します。
- [ActionScript に書き出し] チェックボックスをオンにします。
- [OK] ボタンをクリックして、デフォルトの識別子 "Times-12" をそのまま使用します。

```
var my_format:TextFormat = new TextFormat();
my_format.font = "Times-12";

var my_text1:TextField = this.createTextField("my_text1",
  this.getNextHighestDepth(), 10, 10, 400, 100);
my_text1.text = "This text has sharpness set to 400."
my_text1.embedFonts = true;
my_text1.antiAliasType = "advanced";
my_text1.gridFitType = "pixel";
my_text1.sharpness = 400;
my_text1.setTextFormat(my_format);

var my_text2:TextField = this.createTextField("my_text2",
  this.getNextHighestDepth(), 10, 40, 400, 100);
my_text2.text = "This text has sharpness set to 0."
my_text2.embedFonts = true;
my_text2.antiAliasType = "advanced";
my_text2.gridFitType = "pixel";
my_text2.sharpness = 0;
my_text2.setTextFormat(my_format);
```

```
var my_text3:TextField = this.createTextField("my_text3",
  this.getNextHighestDepth(), 10, 70, 400, 100);
my_text3.text = "This text has sharpness set to -400."
my_text3.embedFonts = true;
my_text3.antiAliasType = "advanced";
my_text3.gridFitType = "pixel";
my_text3.sharpness = -400;
my_text3.setTextFormat(my_format);
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

関連項目

gridFitType (TextField.gridFitType プロパティ),antiAliasType (TextField.antiAliasType プロパティ)

_soundbuftime (TextField._soundbuftime プロパティ)

public _soundbuftime : Number

サウンドのストリーミングを開始するまでにサウンドをプリバッファする秒数です。

メモ : このプロパティを TextField オブジェクトに対して指定することもできますが、実際にはロードされたすべてのサウンドに適用されるグローバルプロパティであるので、単に _soundbuftime という形で値を指定することもできます。このプロパティを TextField オブジェクトに対して設定すると、実際にグローバルプロパティに設定されます。

詳細および使用例については、グローバルプロパティ _soundbuftime を参照してください。

対応バージョン : ActionScript 1.0、Flash Player 6

関連項目

_soundbuftime プロパティ

styleSheet (TextField.styleSheet プロパティ)

```
public styleSheet : StyleSheet
```

テキストフィールドにスタイルシートを関連付けます。スタイルシートの作成方法については、TextField.StyleSheet クラスを参照してください。

テキストフィールドに関連付けられているスタイルシートは、いつでも変更することができます。使用されているスタイルシートを変更すると、テキストフィールドは新しいスタイルシートを使って再描画されます。スタイルシートを null または undefined に設定すると、そのスタイルシートは削除されます。使用されているスタイルシートを削除すると、テキストフィールドはスタイルシートを使用せずに再描画されます。

メモ : スタイルシートを削除すると、TextField.text と TextField.htmlText の両方の内容が変更され、以前にスタイルシートによって適用されたフォーマットが組み込まれます。フォーマットを除く TextField.htmlText の元の内容を保存するには、スタイルシートを削除する前に値を変数に格納しておきます。

対応バージョン : ActionScript 1.0、Flash Player 7

例

次の例では、実行時に news_txt という名前で新しいテキストフィールドを作成します。ステージ上の 3 つのボタン、css1_btn、css2_btn、および clearCss_btn を使用して、news_txt に適用されるスタイルシートの変更や、このテキストフィールドのスタイルシートの消去を行うことができます。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
this.createTextField("news_txt", this.getNextHighestDepth(), 0, 0, 300, 200);
news_txt.wordWrap = true;
news_txt.multiline = true;
news_txt.html = true;
var newsText:String = "<p class='headline'>Description</p> Method; "
  + "starts loading the CSS file into styleSheet. The load operation is
  asynchronous; "
  + "use the <span class='bold'>TextField.StyleSheet.onLoad</span> "
  + "callback handler to determine when the file has finished loading. "
  + "<span class='important'>The CSS file must reside in exactly the same "
  + "domain as the SWF file that is loading it.</span> For more information
  about "
  + "restrictions on loading data across domains, see Flash Player security
  features.";

news_txt.htmlText = newsText;
```

```
css1_btn.onRelease = function() {
  var styleObj:TextField.StyleSheet = new TextField.StyleSheet();
  styleObj.onLoad = function(success:Boolean) {
  if (success) {
    news_txt.styleSheet = styleObj;
    news_txt.htmlText = newsText;
  }
  };
  styleObj.load("styles.css");
};

css2_btn.onRelease = function() {
  var styleObj:TextField.StyleSheet = new TextField.StyleSheet();
  styleObj.onLoad = function(success:Boolean) {
  if (success) {
    news_txt.styleSheet = styleObj;
    news_txt.htmlText = newsText;
  }
  };
  styleObj.load("styles2.css");
};

clearCss_btn.onRelease = function() {
  news_txt.styleSheet = undefined;
  news_txt.htmlText = newsText;
};
```

次のスタイルがテキストフィールドに適用されます。次の 2 つの CSS ファイルを、前に作成した FLA ファイルまたは AS ファイルと同じディレクトリに保存します。

この CSS ファイルを "styles.css" というファイル名で保存します。

```
.important {
  color: #FF0000;
}
.bold {
  font-weight: bold;
}
.headline {
  color: #000000;
  font-family: Arial,Helvetica,sans-serif;
  font-size: 18px;
  font-weight: bold;
  display: block;
}
```

この CSS ファイルを "styles2.css" というファイル名で保存します。

```
.important {
  color: #FF00FF;
}
.bold {
  font-weight: bold;
}
.headline {
  color: #00FF00;
  font-family: Arial,Helvetica,sans-serif;
  font-size: 18px;
  font-weight: bold;
  display: block;
}
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

関連項目

StyleSheet (TextField.StyleSheet)

tabEnabled (TextField.tabEnabled プロパティ)

```
public tabEnabled : Boolean
```

テキストフィールドが自動タブ順に含まれるかどうかを指定します。デフォルト値は undefined です。

tabEnabled プロパティが undefined または true である場合、オブジェクトは自動タブ順に含まれます。tabIndex プロパティにも値を設定すると、オブジェクトはカスタムタブ順にも含まれます。tabEnabled が false の場合は、tabIndex プロパティが設定されていても、オブジェクトは自動タブ順またはカスタムタブ順に含まれません。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、複数のテキストフィールド one_txt、two_txt、three_txt、および four_txt を作成します。テキストフィールド three_txt の tabEnabled プロパティは false に設定され、自動タブ順から除外されます。

```
this.createTextField("one_txt", this.getNextHighestDepth(), 10, 10, 100, 22);
one_txt.border = true;
one_txt.type = "input";
this.createTextField("two_txt", this.getNextHighestDepth(), 10, 40, 100, 22);
two_txt.border = true;
two_txt.type = "input";
this.createTextField("three_txt", this.getNextHighestDepth(), 10, 70, 100, 22);
```

```
three_txt.border = true;
three_txt.type = "input";
this.createTextField("four_txt", this.getNextHighestDepth(), 10, 100, 100, 22);
four_txt.border = true;
four_txt.type = "input";

three_txt.tabEnabled = false;
three_txt.text = "tabEnabled = false;";
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

tabEnabled (Button.tabEnabled プロパティ), tabEnabled (MovieClip.tabEnabled プロパティ)

tabIndex (TextField.tabIndex プロパティ)

```
public tabIndex : Number
```

SWF ファイル内のオブジェクトのタブ順をカスタマイズできます。ボタン、ムービークリップ、またはテキストフィールドインスタンスの tabIndex プロパティを設定できます。デフォルトの値は undefined です。

SWF ファイルに現在表示されているオブジェクトに tabIndex プロパティがある場合は、自動タブ順序が無効になり、SWF ファイルのオブジェクトの tabIndex プロパティからタブ順序が計算されます。カスタムタブ順には、tabIndex プロパティを持つオブジェクトのみが含まれます。

tabIndex プロパティは正の整数である必要があります。オブジェクトのタブ順は、その tabIndex プロパティに従って昇順に決定されます。tabIndex の値が 1 であるオブジェクトは、tabIndex の値が 2 であるオブジェクトよりも優先されます。2 つのオブジェクトの tabIndex が同じ値である場合、オブジェクトのタブ順は undefined になります。

tabIndex プロパティによって定義されるカスタムタブ順は *flat* です。つまり、SWF ファイル内のオブジェクトの階層関係は無視されます。SWF ファイルで tabIndex プロパティを持つすべてのオブジェクトは、タブ順序に従って配置されます。タブ順序は tabIndex の値の順番に従います。tabIndex の値が同じである 2 つのオブジェクト間では、優先順が undefined になります。複数のオブジェクトの tabIndex に同じ値を使用しないでください。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の ActionScript は、4 つのテキストフィールドを動的に作成し、作成したテキストフィールドをカスタムタブ順に割り当てます。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
this.createTextField("one_txt", this.getNextHighestDepth(), 10, 10, 100, 22);
one_txt.border = true;
one_txt.type = "input";
this.createTextField("two_txt", this.getNextHighestDepth(), 10, 40, 100, 22);
two_txt.border = true;
two_txt.type = "input";
this.createTextField("three_txt", this.getNextHighestDepth(), 10, 70, 100, 22);
three_txt.border = true;
three_txt.type = "input";
this.createTextField("four_txt", this.getNextHighestDepth(), 10, 100, 100, 22);
four_txt.border = true;
four_txt.type = "input";

one_txt.tabIndex = 3;
two_txt.tabIndex = 1;
three_txt.tabIndex = 2;
four_txt.tabIndex = 4;
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

tabIndex（Button.tabIndex プロパティ）, tabIndex（MovieClip.tabIndex プロパティ）

_target (TextField._target プロパティ)

public _target : String（読み取り専用）

テキストフィールドインスタンスのターゲットパス。_self は現在のウィンドウ内の現在のフレームを指定します。_blank は新しいウィンドウを指定します。_parent は現在のフレームの親を指定します。_top は現在のウィンドウ内のトップレベルのフレームを指定します。

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の ActionScript は、テキストフィールド my_txt を作成し、この新しいフィールドのターゲットパスをスラッシュ表記とドット表記の両方で出力します。

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 100, 22);
trace(my_txt._target); // output: /my_txt
trace(eval(my_txt._target)); // output: _level0.my_txt
```

この例で使用されている MovieClip.getNextHighestDepth() メソッドでは、Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

text (TextField.text プロパティ)

```
public text : String
```

テキストフィールドの現在のテキストを示します。行は復帰文字 ("\r", ASCII 13) で区切られます。このプロパティは、テキストフィールドが HTML である場合でも、HTML タグが付いていない通常のフォーマットなしのテキストを示します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、HTML テキストフィールド my_txt を作成し、HTML フォーマットのテキストストリングをこのフィールドに割り当てます。htmlText プロパティをトレースすると、HTML フォーマットのストリングが [出力] パネルに表示されます。text プロパティの値をトレースすると、HTML タグの付いたフォーマットされていないストリングが [出力] パネルに表示されます。

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 400, 22);
my_txt.html = true;
my_txt.htmlText = "<B>Lorem ipsum dolor sit amet.</B>";

trace("htmlText: "+my_txt.htmlText);
trace("text: "+my_txt.text);
```

これにより次の出力が生成されます。

```
 htmlText: <P ALIGN="LEFT"><FONT FACE="Times New Roman" SIZE="12"
   COLOR="#000000" KERNING="0">
 <B>Lorem ipsum dolor sit amet.</B></FONT></P>
 text: Lorem ipsum dolor sit amet.
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

htmlText (TextField.htmlText プロパティ)

textColor (TextField.textColor プロパティ)

```
public textColor : Number
```

テキストフィールドのテキストの色を示します。16 進数カラーシステムでは、6 桁の数値を使って色の値を示します。1 つの桁で、16 種類の値 (文字) を指定できます。値の範囲は、0 ～ 9、A ～ F です。黒は (#000000) と表され、その反対の白は (#FFFFFF) と表されます。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の ActionScript を実行すると、テキストフィールドが作成され、そのカラープロパティが赤に変更されます。

```
this.createTextField("my_txt", 99, 10, 10, 100, 300);
my_txt.text = "this will be red text";
my_txt.textColor = 0xFF0000;
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

textHeight (TextField.textHeight プロパティ)

```
public textHeight : Number
```

テキストの高さをピクセル単位で示します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、テキストフィールドを作成し、そのフィールドにテキストストリングを割り当てます。trace ステートメントを使用して、テキストの高さと幅を [出力] パネルに表示します。次に、autoSize プロパティを使用してテキストフィールドのサイズを変更し、新しい高さと幅も [出力] パネルに表示します。

```
this.createTextField("my_txt", 99, 10, 10, 100, 300);
my_txt.text = "Sample text";
trace("textHeight: "+my_txt.textHeight+", textWidth: "+my_txt.textWidth);
trace("_height: "+my_txt._height+", _width: "+my_txt._width+"\n");
my_txt.autoSize = true;
trace("after my_txt.autoSize = true;");
```

```
trace("_height: "+my_txt._height+", _width: "+my_txt._width);
```

次の情報が出力されます。

```
textHeight: 15, textWidth: 56
_height: 300, _width: 100

after my_txt.autoSize = true;
_height: 19, _width: 60
```

関連項目

textWidth (TextField.textWidth プロパティ)

textWidth (TextField.textWidth プロパティ)

```
public textWidth : Number
```

テキストの幅をピクセル単位で示します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

TextField.textHeight の例を参照してください。

関連項目

textHeight (TextField.textHeight プロパティ)

thickness (TextField.thickness プロパティ)

```
public thickness : Number
```

この TextField インスタンス内の文字エッジの太さです。このプロパティは、antiAliasType() が "advanced" に設定されている場合にのみ適用されます。

thickness の範囲は -200 ～ 200 の数値です。thickness をこの範囲外の値に設定しようとすると、値は範囲内の最も近い値 (-200 または 200) に設定されます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

この例では、2 つのテキストフィールドを作成し、thickness の値として一方には -200 を、他方には 200 を適用します。リンケージ識別子が "Times-12" に設定されたライブラリにフォントが埋め込まれていることが前提です。フォントを埋め込むには、次の手順を実行します。

- ライブラリを開きます。
- ライブラリの右上隅にある [ライブラリ] オプションメニューをクリックします。
- ドロップダウンリストから [新しいフォント] を選択します。
- フォントに "Times-12" という名前を付けます。
- ドロップダウンリストから "Times New Roman" を選択します。
- [OK] ボタンをクリックします。
- 新しく作成したフォントを右クリックし、[リンケージ] を選択します。
- [ActionScript に書き出し] チェックボックスをオンにします。
- [OK] ボタンをクリックして、デフォルトの識別子 "Times-12" をそのまま使用します。

```
var my_format:TextFormat = new TextFormat();
my_format.font = "Times-12";

var my_text1:TextField = this.createTextField("my_text1",
  this.getNextHighestDepth(), 10, 10, 300, 30);
my_text1.text = "thickness = 200";
my_text1.antiAliasType = "advanced";
my_text1.border = true;
my_text1.thickness = 200;
my_text1.embedFonts = true;
my_text1.setTextFormat(my_format);

var my_text2:TextField = this.createTextField("my_text2",
  this.getNextHighestDepth(), 10, 50, 300, 30);
my_text2.text = "thickness = -200."
my_text2.antiAliasType = "advanced";
my_text2.thickness = -200;
my_text2.border = true;
my_text2.embedFonts = true;
my_text2.setTextFormat(my_format);
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

関連項目

antiAliasType (TextField.antiAliasType プロパティ)

type (TextField.type プロパティ)

```
public type : String
```

テキストフィールドのタイプを指定します。2 つの値があります。"dynamic" はダイナミックテキストフィールドを指定します。このフィールドをユーザーが編集することはできません。"input" はテキスト入力フィールドを指定します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、2 つのテキストフィールド username_txt および password_txt を作成します。両方のテキストフィールドにテキストが入力されますが、password_txt では password プロパティが true に設定されています。このため、password_txt に入力した文字はアスタリスクで表示されます。

```
this.createTextField("username_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
username_txt.border = true;
username_txt.type = "input";
username_txt.maxChars = 16;
username_txt.text = "hello";

this.createTextField("password_txt", this.getNextHighestDepth(), 10, 40, 100,
  22);
password_txt.border = true;
password_txt.type = "input";
password_txt.maxChars = 16;
password_txt.password = true;
password_txt.text = "world";
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

_url (TextField._url プロパティ)

public _url : String (読み取り専用)

テキストフィールドを作成した SWF ファイルの URL を取得します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、テキストフィールドを作成した SWF ファイルの URL と、ロードされた SWF ファイルを取得します。

```
this.createTextField("my_txt", 1, 10, 10, 100, 22);
trace(my_txt._url);

var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  trace(target_mc._url);
};
var holder_mcl:MovieClipLoader = new MovieClipLoader();
holder_mcl.addListener(mclListener);
holder_mcl.loadClip("best_flash_ever.swf",
  this.createEmptyMovieClip("holder_mc", 2));
```

この例をテストすると、テストしている SWF ファイルの URL と、ファイル best_flash_ever.swf が [出力] パネルに表示されます。

この例で使用されている MovieClipLoader クラスでは、Flash Player 7 以降が必要です。

variable (TextField.variable プロパティ)

public variable : String

テキストフィールドに関連付けられた変数の名前です。このプロパティのタイプは String です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、テキストフィールド my_txt を作成し、このフィールドに変数 today_date を関連付けます。変数 today_date を変更すると、my_txt に表示されるテキストが更新されます。

```
this.createTextField("my_txt", 1, 10, 10, 200, 22);
my_txt.variable = "today_date";
var today_date:Date = new Date();

var date_interval:Number = setInterval(updateDate, 500);
function updateDate():Void {
  today_date = new Date();
}
```

_visible (TextField._visible プロパティ)

public _visible : Boolean

テキストフィールド *my_txt* が表示されるかどうかを示すブール値です。_visible プロパティが false に設定されている非表示のテキストフィールドは、使用できません。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、テキストフィールド my_txt を作成します。ボタン visible_btn は、my_txt の表示と非表示を切り替えます。

```
this.createTextField("my_txt", 1, 10, 10, 200, 22);
my_txt.background = true;
my_txt.backgroundColor = 0xDFDFDF;
my_txt.border = true;
my_txt.type = "input";

visible_btn.onRelease = function() {
  my_txt._visible = !my_txt._visible;
};
```

関連項目

_visible (Button._visible プロパティ),_visible (MovieClip._visible プロパティ)

_width (TextField._width プロパティ)

public _width : Number

ピクセル単位で示したテキストフィールドの幅。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ステージ上の 3 番目のテキストフィールドの幅と高さを変更するときに使用できる 2 つのテキストフィールドを作成します。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 40, 160, 120);
my_txt.background = true;
my_txt.backgroundColor = 0xFF0000;
my_txt.border = true;
my_txt.multiline = true;
my_txt.type = "input";
my_txt.wordWrap = true;
```

```
this.createTextField("width_txt", this.getNextHighestDepth(), 10, 10, 30, 20);
width_txt.border = true;
width_txt.maxChars = 3;
width_txt.restrict = "0-9";
width_txt.type = "input";
width_txt.text = my_txt._width;
width_txt.onChanged = function() {
  my_txt._width = this.text;
}

this.createTextField("height_txt", this.getNextHighestDepth(), 70, 10, 30, 20);
height_txt.border = true;
height_txt.maxChars = 3;
height_txt.restrict = "0-9";
height_txt.type = "input";
height_txt.text = my_txt._height;
height_txt.onChanged = function() {
  my_txt._height = this.text;
}
```

この例をテストするときには、width_txt と height_txt に新しい値を入力して、my_txt のサイズを変更してみてください。

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

_height (TextField._height プロパティ)

wordWrap (TextField.wordWrap プロパティ)

```
public wordWrap : Boolean
```

テキストフィールドのテキストを折り返すかどうかを示すブール値。wordWrap の値が true である場合は、テキストフィールドのテキストを折り返し、false である場合は折り返しません。

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の例では、実行時に作成されるテキストフィールド内の長いテキストに、wordWrap の設定がどのように影響するかを示します。

```
this.createTextField("my_txt", 99, 10, 10, 100, 200);
my_txt.text = "This is very long text that will certainly extend beyond the width
  of this text field";
my_txt.border = true;
```

[制御]-[ムービープレビュー] を選択して、Flash Player で SWF ファイルをテストします。次に、ActionScript に戻ってコードに次の行を追加し、再度 SWF ファイルをテストします。

```
my_txt.wordWrap = true;
```

_x (TextField._x プロパティ)

```
public _x : Number
```

親ムービークリップのローカル座標を基準にしてテキストフィールドの x 座標を設定する整数。テキストフィールドがメインのタイムラインにある場合、その座標系はステージの左上隅を (0,0) として参照します。変形されているムービークリップの内部にテキストフィールドがある場合、そのテキストフィールドの座標系はそれを囲むムービークリップのローカル座標系になります。したがって、反時計回りに 90 度回転したムービークリップに囲まれているテキストフィールドは、反時計回りに 90 度回転した座標系を継承します。テキストフィールドの座標は、基準点の位置を参照します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、マウスをクリックした場所にテキストフィールドを作成します。作成したテキストフィールドには、そのテキストフィールドの現在の *x* 座標と *y* 座標が表示されます。

```
this.createTextField("coords_txt", this.getNextHighestDepth(), 0, 0, 60, 22);
coords_txt.autoSize = true;
coords_txt.selectable = false;
coords_txt.border = true;

var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  coords_txt.text = "X:"+Math.round(_xmouse)+", Y:"+Math.round(_ymouse);
  coords_txt._x = _xmouse;
  coords_txt._y = _ymouse;
};
Mouse.addListener(mouseListener);
```

この例で使用している `MovieClip.getNextHighestDepth()` メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、`MovieClip.getNextHighestDepth()` メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

_xscale (TextField._xscale プロパティ), _y (TextField._y プロパティ), _yscale (TextField._yscale プロパティ)

_xmouse (TextField._xmouse プロパティ)

public _xmouse : Number (読み取り専用)

テキストフィールドを基準にしたポインタの x 座標を返します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ステージ上に 3 つのテキストフィールドを作成します。mouse_txt インスタンスは、ステージを基準にしたマウスの現在位置を表示します。textfield_txt インスタンスは、my_txt インスタンスを基準としたマウスポインタの現在位置を表示します。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
this.createTextField("mouse_txt", this.getNextHighestDepth(), 10, 10, 200, 22);
mouse_txt.border = true;
this.createTextField("textfield_txt", this.getNextHighestDepth(), 220, 10, 200,
  22);
textfield_txt.border = true;
this.createTextField("my_txt", this.getNextHighestDepth(), 100, 100, 160, 120);
my_txt.border = true;

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  mouse_txt.text = "MOUSE ... X:" + Math.round(_xmouse) + ",\tY:" +
  Math.round(_ymouse);
  textfield_txt.text = "TEXTFIELD ... X:" + Math.round(my_txt._xmouse) + ",\tY:"
  +
  Math.round(my_txt._ymouse);
}

Mouse.addListener(mouseListener);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

_ymouse (TextField._ymouse プロパティ)

_xscale (TextField._xscale プロパティ)

```
public _xscale : Number
```

テキストフィールドの基準点から適用する、テキストフィールドの水平方向の拡大・縮小率 (パーセンテージ) を決定します。デフォルトの基準点は (0,0) です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、scaleUp_btn インスタンスおよび scaleDown_btn インスタンスをクリックしたときに、my_txt インスタンスを拡大・縮小します。

```
this.createTextField("my_txt", 99, 10, 40, 100, 22);
my_txt.autoSize = true;
my_txt.border = true;
my_txt.selectable = false;
my_txt.text = "Sample text goes here.";

scaleUp_btn.onRelease = function() {
  my_txt._xscale = 2;
  my_txt._yscale = 2;
}
scaleDown_btn.onRelease = function() {
  my_txt._xscale /= 2;
  my_txt._yscale /= 2;
}
```

関連項目

_x (TextField._x プロパティ), _y (TextField._y プロパティ), _yscale (TextField._yscale プロパティ)

_y (TextField._y プロパティ)

```
public _y : Number
```

親ムービークリップのローカル座標を基準にしたテキストフィールドの y 座標。テキストフィールドがメインのタイムラインにある場合、その座標系はステージの左上隅を (0,0) として参照します。変形されているムービークリップの内部にテキストフィールドがある場合、そのテキストフィールドの座標系はそれを囲むムービークリップのローカル座標系になります。したがって、反時計回りに 90 度回転したムービークリップに囲まれているテキストフィールドは、反時計回りに 90 度回転した座標系を継承します。テキストフィールドの座標は、基準点の位置を参照します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

TextField._x の例を参照してください。

関連項目

_x（TextField._x プロパティ）, _xscale（TextField._xscale プロパティ）, _yscale（TextField._yscale プロパティ）

_ymouse (TextField._ymouse プロパティ)

public _ymouse : Number（読み取り専用）

テキストフィールドを基準にしたポインタの y 座標を示します。

対応バージョン： ActionScript 1.0、Flash Player 6

例

TextField._xmouse の例を参照してください。

関連項目

_xmouse（TextField._xmouse プロパティ）

_yscale (TextField._yscale プロパティ)

public _yscale : Number

テキストフィールドの基準点から適用する、テキストフィールドの垂直方向の拡大・縮小率（パーセンテージ）。デフォルトの基準点は (0,0) です。

対応バージョン： ActionScript 1.0、Flash Player 6

例

TextField._xscale の例を参照してください。

関連項目

_x（TextField._x プロパティ）, _xscale（TextField._xscale プロパティ）, _y（TextField._y プロパティ）

TextFormat

```
Object
  |
  +-TextFormat
```

```
public class TextFormat
extends Object
```

TextFormat クラスは、文字フォーマット情報を表します。TextFormat クラスを使用して、テキストフィールドの特定のテキストフォーマットを作成します。静止テキストおよびダイナミックテキストフィールドの両方にテキストフォーマットを適用できます。TextFormat クラスのプロパティは、デバイスフォントおよび埋めこみフォントに適用されます。ただし、埋め込みフォントの場合、ボールドとイタリックのテキストには、実際に特定のフォントが必要です。埋め込みフォントでボールドまたはイタリックのテキストを表示する場合は、そのフォントのボールド体およびイタリック体を埋め込む必要があります。

ContextMenu オブジェクトのメソッドを呼び出す前に、`new TextFormat()` コンストラクタを使用して ContextMenu オブジェクトを作成する必要があります。

TextFormat パラメータを `null` に設定すると、それらのパラメータが未定義であることを示すことができます。次の例のように `TextField.setTextFormat()` を使用してテキストフィールドに TextFormat オブジェクトを適用すると、その定義済みのプロパティだけが適用されます。

```
this.createTextField("my_txt", this.getNextHighestDepth(), 0, 0, 100, 22);
my_txt.autoSize = true;
my_txt.text = "Lorem ipsum dolor sit amet...";

var my_fmt:TextFormat = new TextFormat();
my_fmt.bold = true;
my_txt.setTextFormat(my_fmt);
```

このコードはまず、空の TextFormat オブジェクトを作成します。このオブジェクトのプロパティはすべて `null` となります。次に、`bold` プロパティを定義済みの値に設定します。この例で使用している `MovieClip.getNextHighestDepth()` メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、`MovieClip.getNextHighestDepth()` メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

`my_txt.setTextFormat(my_fmt)` というコードはテキストフィールドのデフォルトのテキストフォーマットのうち `bold` プロパティだけを変更します。これは、`my_fmt` で定義されているのが `bold` プロパティだけであるためです。残りのプロパティはいずれも変更されません。

`TextField.getTextFormat()` を呼び出すと、すべてのプロパティが定義された TextFormat オブジェクトが返されます。`null` のプロパティはありません。

対応バージョン： ActionScript 1.0、Flash Player 6

関連項目

setTextFormat (TextField.setTextFormat メソッド),
getTextFormat (TextField.getTextFormat メソッド)

プロパティ一覧

オプション	プロパティ	説明
	align:String	段落の整列の設定を示すストリングです。
	blockIndent:Number	ブロックのインデントをポイント単位で示す数値です。
	bold:Boolean	テキストフィールドがボールド体であるかどうかを指定するブール値です。
	bullet:Boolean	テキストが箇条書きリストにあるかどうかを示すブール値です。
	color:Number	テキストの色を示します。
	font:String	このテキストフォーマットでのテキストフォント名を示すストリングです。
	indent:Number	左マージンから段落の先頭文字までのインデントを示す整数です。
	italic:Boolean	このテキストフォーマットのテキストをイタリックにするかどうかを示すブール値です。
	kerning:Boolean	カーニングの有効 / 無効を示すブール値です。
	leading:Number	行間の垂直の行送りを示す整数。
	leftMargin:Number	段落の左マージンをポイント単位で示します。
	letterSpacing:Number	文字間に均等に配分されるスペースの量です。
	rightMargin:Number	段落の右マージンをポイント単位で示します。
	size:Number	このテキストフォーマットでのテキストのポイントサイズです。
	tabStops:Array	カスタムタブストップを負以外の整数の配列として指定します。
	target:String	ハイパーリンクを表示するターゲットウィンドウを示します。
	underline:Boolean	このテキストフォーマットを使用するテキストにアンダーラインを表示するか (true)、または表示しないか (false) を示すブール値です。
	url:String	このテキストフォーマットを使用するテキストのハイパーリンク先の URL を示します。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター一覧

署名	説明
TextFormat([font: String], [size:Number], [color:Number], [bold:Boolean], [italic:Boolean], [underline:Boolean], [url:String], [target:String], [align:String], [leftMargin:Number], [rightMargin:Number], [indent:Number], [leading:Number])	指定されたプロパティを使用して TextFormat オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	getTextExtent(text: String, [width: Number]) : Object	my_fmt に指定したフォーマットのテキストストリング text のテキスト寸法情報を返します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

align (TextFormat.align プロパティ)

```
public align : String
```

段落の整列の設定を示すストリングです。静止テキストおよびダイナミックテキストにこのプロパティを適用できます。次の一覧に、このプロパティに指定できる値を示します。

- "left"– は段落を左揃えにします。
- "center"– は段落を中央揃えにします。
- "right"– は段落を右揃えにします。
- "justify"– は段落を両端揃えにします。これは Flash Player 8 で追加された値です。

デフォルト値 null はプロパティを未設定にします。

対応バージョン： ActionScript 1.0、Flash Player 6 - "justify" 値は、Flash Player 8 から使用できます。

例

次の例では、両端揃えに設定された align プロパティを示します。テキストが水平方向に均等な間隔で表示されるよう、各行の文字が横に広がります。

```
var format:TextFormat = new TextFormat();
format.align = "justify";

var txtField:TextField = this.createTextField("txtField",
  this.getNextHighestDepth(), 100, 100, 300, 100);
txtField.multiline = true;
txtField.wordWrap = true;
txtField.border = true;
txtField.text = "When this text is justified, it will be "
  + "spread out to more cleanly fill the horizontal "
  + "space for each line. This can be considered an "
  + "improvement over regular left-aligned text that "
  + "will simply wrap and do no more.";
txtField.setTextFormat(format);
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

blockIndent (TextFormat.blockIndent プロパティ)

```
public blockIndent : Number
```

ブロックのインデントをポイント単位で示す数値。ブロックのインデントは、テキストのブロック全体、つまりテキストのすべての行に適用されます。一方、通常のインデント (TextFormat.indent) は各段落の先頭行にのみ影響します。このプロパティが null である場合、TextFormat オブジェクトはブロックのインデントを指定しません。

対応バージョン : ActionScript 1.0、Flash Player 6

例

この例では、境界線のあるテキストフィールドを作成し、ブロックのインデントを 20 に設定します。

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.blockIndent = 20;

mytext.text = "This is my first test field object text";
mytext.setTextFormat(myformat);
```

bold (TextFormat.bold プロパティ)

```
public bold : Boolean
```

テキストフィールドがボールド体であるかどうかを指定するブール値。デフォルト値 null はプロパティを未設定にします。値が true の場合は、テキストがボールド体になります。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ボールド体の文字を含むテキストフィールドを作成します。

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.bold = true;

this.createTextField("my_txt", 1, 100, 100, 300, 100);
my_txt.multiline = true;
my_txt.wordWrap = true;
my_txt.border = true;
my_txt.text = "This is my test field object text";
my_txt.setTextFormat(my_fmt);
```

bullet (TextFormat.bullet プロパティ)

```
public bullet : Boolean
```

テキストが箇条書きリストにあるかどうかを示すブール値。箇条書きリストでは、テキストの各段落がインデントされます。箇条書きシンボルは、各段落の先頭行の左に表示されます。デフォルト値は null です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、実行時に新しいテキストフィールドを作成し、そのフィールドに改行を含むストリングを入力します。TextFormat クラスを使用して文字にフォーマットを設定し、テキストフィールドの各行に箇条書きシンボルを追加します。次の ActionScript を使用します。

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.bullet = true;

this.createTextField("my_txt", 1, 100, 100, 300, 100);
my_txt.multiline = true;
my_txt.wordWrap = true;
my_txt.border = true;
my_txt.text = "this is my text"+newline;
my_txt.text += "this is more text"+newline;
my_txt.setTextFormat(my_fmt);
```

color (TextFormat.color プロパティ)

```
public color : Number
```

テキストの色を示します。たとえば、0xFF0000 は赤、0x00FF00 は緑など、3 つの 8 ビットの RGB コンポーネントを示す数値です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、テキストフィールドを作成し、テキストの色を赤に設定します。

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.blockIndent = 20;
my_fmt.color = 0xFF0000; // hex value for red

this.createTextField("my_txt", 1, 100, 100, 300, 100);
my_txt.multiline = true;
my_txt.wordWrap = true;
my_txt.border = true;
my_txt.text = "this is my first test field object text";
my_txt.setTextFormat(my_fmt);
```

font (TextFormat.font プロパティ)

```
public font : String
```

このテキストフォーマットでのテキストフォント名を示すストリング。デフォルト値 null はプロパティを未設定にします。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、テキストフィールドを作成し、フォントを Courier に設定します。

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.font = "Courier";

mytext.text = "this is my first test field object text";
mytext.setTextFormat(myformat);
```

getTextExtent (TextFormat.getTextExtent メソッド)

```
public getTextExtent(text:String, [width:Number]) : Object
```

my_fmt に指定したフォーマットのテキストストリング text のテキスト寸法情報を返します。text ストリングは標準テキスト (非 HTML) として扱われます。

このメソッドは、6 つのプロパティを持つオブジェクトを返します。ascent、descent、width、height、textFieldHeight、および textFieldWidth の 6 つです。すべてのプロパティ値はピクセル単位です。

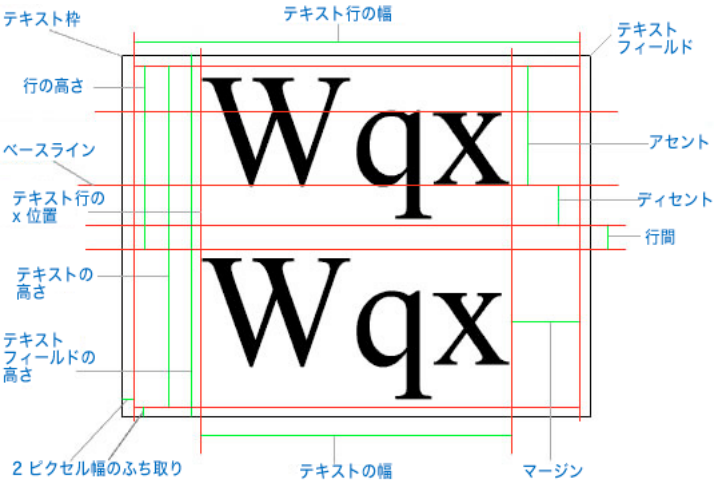
width パラメータを指定した場合は、指定したテキストに折り返しが適用されます。これにより、指定したすべてのテキストを表示するために必要なテキストボックスの高さを判断することができます。

ascent プロパティはテキスト行のベースラインよりも上の長さ、descent プロパティはテキスト行のベースラインよりも下の長さを表します。最初のテキスト行のベースラインは、テキストフィールドの基準点に ascent パラメータ値を加算した位置になります。

width プロパティと height プロパティは、テキストストリングの幅と高さを示します。textFieldHeight プロパティと textFieldWidth プロパティは、テキストストリング全体を表示するのに必要なテキストフィールドオブジェクトの高さと幅を表します。テキストフィールドの周囲には 2 ピクセル幅の " ふち取り " があるので、textFieldHeight の値は height + 4 になります。同様に、textFieldWidth の値は常に width + 4 になります。

テキストメトリクスに基づいてテキストフィールドを作成する場合は、height ではなく textFieldHeight を、width ではなく textFieldWidth を使用してください。

次の図に、この寸法を示します。



TextFormat オブジェクトをセットアップする際には、フォント名、フォントサイズ、行間を含め、テキストフィールドを作成する際とまったく同じようにすべての属性を設定する必要があります。行間のデフォルト値は 2 です。

対応バージョン： ActionScript 1.0、Flash Player 6 - width パラメータは、Flash Player 7 でサポートされています。

パラメータ

`text:String` - ストリング。

`width:Number` (オプション) - 指定したテキストを折り返す幅を示す数値 (ピクセル単位)。

戻り値

`Object` - `width`、`height`、`ascent`、`descent`、`textFieldHeight`、`textFieldWidth` の各プロパティを持つオブジェクト。

例

次の例では、指定したフォーマットを使ったテキストストリングを表示する、ストリングと同じサイズの単一行テキストフィールドを作成します。

```
var my_str:String = "Small string";

// Create a TextFormat object,
// and apply its properties.
var my_fmt:TextFormat = new TextFormat();
```

```
with (my_fmt) {
  font = "Arial";
  bold = true;
}

// Obtain metrics information for the text string
// with the specified formatting.
var metrics:Object = my_fmt.getTextExtent(my_str);

// Create a text field just large enough to display the text.
this.createTextField("my_txt", this.getNextHighestDepth(), 100, 100,
  metrics.textFieldWidth,
metrics.textFieldHeight);
my_txt.border = true;
my_txt.wordWrap = true;
// Assign the same text string and TextFormat object to the my_txt object.
my_txt.text = my_str;
my_txt.setTextFormat(my_fmt);
```

次の例では、指定したフォーマットのテキストストリングを表示できるだけの高さで、複数行の 100 ピクセル幅のテキストフィールドを作成します。

```
// Create a TextFormat object.
var my_fmt:TextFormat = new TextFormat();
// Specify formatting properties for the TextFormat object:
my_fmt.font = "Arial";
my_fmt.bold = true;
my_fmt.leading = 4;

// The string of text to be displayed
var textToDisplay:String = "Adobe Flash Player 9 text metrics demonstration.";

// Obtain text measurement information for the string,
// wrapped at 100 pixels.
var metrics:Object = my_fmt.getTextExtent(textToDisplay, 100);

// Create a new TextField object using the metric
// information just obtained.
this.createTextField("my_txt", this.getNextHighestDepth(), 50, 50-
  metrics.ascent, 100,
metrics.textFieldHeight);
my_txt.wordWrap = true;
my_txt.border = true;
// Assign the text and the TextFormat object to the TextObject:
my_txt.text = textToDisplay;
my_txt.setTextFormat(my_fmt);
```

indent (TextFormat.indent プロパティ)

```
public indent : Number
```

左マージンから段落の先頭文字までのインデントを示す整数。正の値は通常のインデントを示します。負の値も使用できますが、負のインデントが適用されるのは、左マージンが 0 より大きい場合だけです。0 より大きいマージンを設定するには、TextFormat オブジェクトの `indent` プロパティまたは `blockIndent` プロパティを使用します。デフォルト値 `null` はプロパティを未設定にします。

対応バージョン： ActionScript 1.0、Flash Player 6 - 負の値を使用できるようになったのは Flash Player 8 からです。

例

次の例では、テキストフィールドを作成し、インデントを 10 に設定します。

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.indent = 10;

mytext.text = "this is my first test field object text";
mytext.setTextFormat(myformat);
```

関連項目

`blockIndent (TextFormat.blockIndent プロパティ )`

italic (TextFormat.italic プロパティ)

```
public italic : Boolean
```

このテキストフォーマットのテキストをイタリックにするかどうかを示すブール値。デフォルト値 `null` はプロパティを未設定にします。

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の例では、テキストフィールドを作成し、テキストスタイルをイタリックに設定します。

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;
```

```
var myformat:TextFormat = new TextFormat();
myformat.italic = true;

mytext.text = "This is my first text field object text";
mytext.setTextFormat(myformat);
```

kerning (TextFormat.kerning プロパティ)

```
public kerning : Boolean
```

カーニングの有効 / 無効を示すブール値です。カーニングでは、読みやすくするために、特定の文字ペア間にあらかじめ決められたスペースを配置します。デフォルト値 `false` はカーニングを無効にします。

カーニングは、埋め込みフォントに対してのみサポートされています。Courier New など特定のフォントでは、カーニングがサポートされていません。

`kerning` プロパティは、Windows で作成された SWF ファイルでのみサポートされます。Macintosh で作成された SWF ファイルではサポートされません。ただし、Windows SWF ファイルは Windows バージョン以外の Flash Player で再生可能であり、カーニングも適用されます。

大きなフォントの見出しを使用する場合など、必要な場合にのみカーニングを使用してください。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、2 つのテキストフィールドを示します。最初のテキストフィールドのフォーマットでは、`kerning` が `false` に設定された赤いテキストを使用します。2 番目のテキストフィールドのフォーマットでは、`kerning` が `true` に設定された青いテキストを使用します。この例を使用するには、フォントシンボルをライブラリに追加し、フォントとして Arial を選択します。このフォントの [リンケージプロパティ] ダイアログボックスで、識別子名を `"Font 1"` に設定し、[ActionScript に書き出し] を選択した後に [最初のフレームに書き出し] を選択します。

```
var fmt1:TextFormat = new TextFormat();
fmt1.font = "Font 1";
fmt1.size = 50;
fmt1.color = 0xFF0000;
fmt1.kerning = false;

var fmt2:TextFormat = new TextFormat();
fmt2.font = "Font 1";
fmt2.size = 50;
fmt2.color = 0x0000FF;
fmt2.kerning = true;
```

```
this.createTextField("tf1", this.getNextHighestDepth(), 10, 10, 400, 100);
tf1.embedFonts = true;
tf1.text = "Text 7AVA-7AVA";
tf1.setTextFormat(fmt1);

this.createTextField("tf2", this.getNextHighestDepth(), 10, 40, 400, 100);
tf2.embedFonts = true;
tf2.text = tf1.text;
tf2.setTextFormat(fmt2);
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している `MovieClip.getNextHighestDepth()` メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

leading (TextFormat.leading プロパティ)

```
public leading : Number
```

行間の垂直の行送りを示す整数。デフォルト値 `null` はプロパティを未設定にします。

Flash Player 8 では、行間の行送りがテキストの高さより小さい負の行送りがサポートされています。負の行送りは、見出しなどのように、テキスト行をまとめて非常に近くに配置する場合に便利です。テキストの重複を防止するには、すべて大文字のテキストなど、デセンダを含まないテキスト行に対して負の行送りを使用します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、テキストフィールドを作成し、行間を 10 に設定します。

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.leading = 10;

this.createTextField("my_txt", 1, 100, 100, 100, 100);
my_txt.multiline = true;
my_txt.wordWrap = true;
my_txt.border = true;
my_txt.text = "This is my first text field object text";
my_txt.setTextFormat(my_fmt);
```

leftMargin (TextFormat.leftMargin プロパティ)

```
public leftMargin : Number
```

段落の左マージンをポイント単位で示します。デフォルト値 null はプロパティを未設定にします。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、テキストフィールドを作成し、左マージンを 20 ポイントに設定します。

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.leftMargin = 20;

mytext.text = "this is my first test field object text";
mytext.setTextFormat(myformat);
```

letterSpacing (TextFormat.letterSpacing プロパティ)

```
public letterSpacing : Number
```

文字間に均等に配分されるスペースの量です。この数値は、各文字の後の送りに追加されるピクセル数を示します。負の値を指定すると、文字の間隔が狭くなります。

システムフォントでは整数値のみがサポートされます。ただし、埋め込みフォントの場合は、浮動小数点 (非整数) 値 (2.6 など) を指定できます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、2 つの TextFormat オブジェクトを使用し、テキストフィールド内の異なるテキスト範囲に対して、letterSpacing の値をそれぞれ正と負に設定します。

```
this.createTextField("mytext", this.getNextHighestDepth(), 10, 10, 200, 100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var format1:TextFormat = new TextFormat();
format1.letterSpacing = -1;

var format2:TextFormat = new TextFormat();
format2.letterSpacing = 10;

mytext.text = "Eat at \nJOE'S.";
mytext.setTextFormat(0, 7, format1);
mytext.setTextFormat(8, 12, format2);
```

SWF ファイルにバージョン 2 のコンポーネントがある場合は、この例で使用している MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 コンポーネントの DepthManager クラスを使用します。

rightMargin (TextFormat.rightMargin プロパティ)

public rightMargin : Number

段落の右マージンをポイント単位で示します。デフォルト値 null はプロパティを未設定にします。

対応バージョン: ActionScript 1.0、Flash Player 6

例

次の例では、テキストフィールドを作成し、右マージンを 20 ポイントに設定します。

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.rightMargin = 20;

mytext.text = "this is my first test field object text";
mytext.setTextFormat(myformat);
```

size (TextFormat.size プロパティ)

public size : Number

このテキストフォーマットでのテキストのポイントサイズ。デフォルト値 null はプロパティを未設定にします。

対応バージョン: ActionScript 1.0、Flash Player 6

例

次の例では、テキストフィールドを作成し、テキストサイズを 20 ポイントに設定します。

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.size = 20;

mytext.text = "This is my first text field object text";
mytext.setTextFormat(myformat);
```

tabStops (TextFormat.tabStops プロパティ)

```
public tabStops : Array
```

カスタムタブストップを負以外の整数の配列として指定します。各タブストップはピクセル単位で指定します。カスタムタブストップを指定しないと (null)、タブストップはデフォルトの 4 (平均文字幅) になります。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、タブストップが 40 ピクセルごとのテキストフィールドと、タブストップが 75 ピクセルごとのテキストフィールドを作成します。

```
this.createTextField("mytext",1,100,100,400,100);
mytext.border = true;
var myformat:TextFormat = new TextFormat();
myformat.tabStops = [40,80,120,160];
mytext.text = "A\tB\tC\tD"; // \t is the tab stop character
mytext.setTextFormat(myformat);

this.createTextField("mytext2",2,100,220,400,100);
mytext2.border = true;
var myformat2:TextFormat = new TextFormat();
myformat2.tabStops = [75,150,225,300];
mytext2.text ="A\tB\tC\tD";
mytext2.setTextFormat(myformat2);
```

target (TextFormat.target プロパティ)

```
public target : String
```

ハイパーリンクを表示するターゲットウィンドウを示します。ターゲットウィンドウが空のストリングである場合、テキストはデフォルトのターゲットウィンドウ _self に表示されます。カスタム名を選択することも、次の 4 つの名前のいずれかを選択することもできます。_self は現在のウィンドウ内の現在のフレームを指定します。_blank は新しいウィンドウを指定します。_parent は現在のフレームの親を指定します。_top は現在のウィンドウ内のトップレベルのフレームを指定します。TextFormat.url プロパティが空のストリングまたは null の場合は、このプロパティを取得および設定することはできますが、プロパティには何の影響もありません。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、アドビ システムズ社の Web サイトへのハイパーリンクを持つテキストフィールドを作成します。この例では、TextFormat.target を使用して、アドビ システムズ社の Web サイトを新しいブラウザウィンドウに表示します。

```
var myformat:TextFormat = new TextFormat();
myformat.url = "http://www.Adobe.com";
myformat.target = "_blank";

this.createTextField("mytext",1,100,100,200,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;
mytext.html = true;
mytext.text = "Go to adobe.com";
mytext.setTextFormat(myformat);
```

関連項目

url （TextFormat.url プロパティ）

TextFormat コンストラクタ

```
public TextFormat([font:String], [size:Number], [color:Number], [bold:Boolean],
  [italic:Boolean], [underline:Boolean], [url:String], [target:String],
  [align:String], [leftMargin:Number], [rightMargin:Number], [indent:Number],
  [leading:Number])
```

指定されたプロパティを使用して TextFormat オブジェクトを作成します。この TextFormat オブジェクトのプロパティを変更して、テキストフィールドのフォーマットを変更できます。

null 値を設定したパラメータは未設定になります。すべてのパラメータはオプションです。省略したパラメータは null として扱われます。

対応バージョン： ActionScript 1.0、Flash Player 6

パラメータ

font:String（オプション）- テキストのフォント名を示すストリング。

size:Number（オプション）- ポイントサイズを示す整数。

color:Number（オプション）- このテキストフォーマットを使用するテキストの色。たとえば、0xFF0000 は赤、0x00FF00 は緑など、3 つの 8 ビットの RGB コンポーネントを示す数値です。

bold:Boolean（オプション）- テキストがボールド体であるかどうかを示すブール値。

italic:Boolean（オプション）- テキストがイタリック体であるかどうかを示すブール値。

underline:Boolean（オプション）- テキストが下線付きであるかどうかを示すブール値。

url:String (オプション) - このテキストフォーマットのテキストのハイパーリンク先である URL。url が空のストリングである場合、テキストにはハイパーリンクがありません。

target:String (オプション) - ハイパーリンクを表示するターゲットウィンドウ。ターゲットウィンドウが空のストリングである場合、テキストはデフォルトのターゲットウィンドウ _self に表示されます。url パラメータに空のストリングまたは null 値を指定した場合は、このプロパティを取得または設定することはできますが、プロパティには何の影響もありません。

align:String (オプション) - 段落の整列の設定を示すストリング。"left" は段落を左揃えにします。"center" は段落を中央揃えにします。"right" は段落を右揃えにします。

leftMargin:Number (オプション) - 段落の左マージンをポイント単位で示します。

rightMargin:Number (オプション) - 段落の右マージンをポイント単位で示します。

indent:Number (オプション) - 左マージンから段落の先頭文字までのインデントを示す整数。

leading:Number (オプション) - 行間の垂直の行送りを示す数値。

例

次の例では、TextFormat オブジェクトを作成し、stats_txt テキストフィールドをフォーマットしてから、テキストを表示する新しいテキストフィールドを作成します。

```
// Define a TextFormat which is used to format the stats_txt text field.
var my_fmt:TextFormat = new TextFormat();
my_fmt.bold = true;
my_fmt.font = "Arial";
my_fmt.size = 12;
my_fmt.color = 0xFF0000;
// Create a text field to display the player's statistics.
this.createTextField("stats_txt", 5000, 10, 0, 530, 22);
// Apply the TextFormat to the text field.
stats_txt.setNewTextFormat(my_fmt);
stats_txt.selectable = false;
stats_txt.text = "Lorem ipsum dolor sit amet...";
```

別の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Strings" フォルダに移動して Strings.fla ファイルにアクセスします。

underline (TextFormat.underline プロパティ)

```
public underline : Boolean
```

このテキストフォーマットを使用するテキストにアンダーラインを表示するか (true)、または表示しないか (false) を示すブール値です。これは、<U> タグによって設定されるアンダーラインと似ていますが、このタグの場合はデセンダが正しくスキップされないので、本物のアンダーラインではありません。デフォルト値 null はプロパティを未設定にします。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、テキストフィールドを作成し、テキストスタイルを下線に設定します。

```
this.createTextField("mytext",1,100,100,200,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.underline = true;
mytext.text = "This is my first text field object text";
mytext.setTextFormat(myformat);
```

url (TextFormat.url プロパティ)

```
public url : String
```

このテキストフォーマットを使用するテキストのハイパーリンク先の URL を示します。url プロパティが空のストリングである場合、テキストにはハイパーリンクがありません。デフォルト値 null はプロパティを未設定にします。

対応バージョン : ActionScript 1.0、Flash Player 6

例

この例では、Adobe Web サイトへのハイパーリンクが設定されたテキストフィールドを作成します。

```
var myformat:TextFormat = new TextFormat();
myformat.url = "http://www.adobe.com";

this.createTextField("mytext",1,100,100,200,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;
mytext.html = true;
mytext.text = "Go to Adobe.com";
mytext.setTextFormat(myformat);
```

TextRenderer (flash.text.TextRenderer)

```
Object
  |
  +-flash.text.TextRenderer

public class TextRenderer
extends Object
```

TextRenderer クラスには、埋め込みフォントの高度なアンチエイリアス機能が用意されています。高度なアンチエイリアスに設定すると、小さいフォントを非常に高品質でレンダリングできます。高度なアンチエイリアスは、小さいフォントのテキストが多いアプリケーションで使用します。非常に大きいフォント (48 ポイント以上) に対して高度なアンチエイリアスを使用することはお勧めしません。高度なアンチエイリアスは、Flash Player 8 でのみ使用できます。

テキストフィールドで高度なアンチエイリアスを設定するには、TextField インスタンスの antiAliasType プロパティを設定します。次の例では、"CustomFont" というリンケージ識別子を持つ共有フォントがライブラリ内に必要です。

```
var txtFormat:TextFormat = new TextFormat();
txtFormat.font = "CustomFont";

var label:TextField = this.createTextField("label", this.getNextHighestDepth(),
  10, 10, 200, 20);
label.setNewTextFormat(txtFormat);
label.text = "Hello World";
label.embedFonts = true;
label.antiAliasType = "advanced";
```

高度なアンチエイリアスには、線の太さとエッジのシャープネスの両方を連続的に変調する CSM (Continuous Stroke Modulation) 機能が用意されています。高度な機能として、setAdvancedAntialiasingTable() メソッドを使用すると、特定の書体やフォントサイズの設定を定義できます。

対応バージョン : ActionScript 1.0、Flash Player 8

関連項目

antiAliasType (TextField.antiAliasType プロパティ)

プロパティ一覧

オプション	プロパティ	説明
static	displayMode:String	高度なアンチエイリアス処理をしたテキストのレンダリングを制御します。
static	maxLevel:Number	高度なアンチエイリアス用の ADF 品質レベルを設定します。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

メソッド一覧

オプション	署名	説明
static	setAdvancedAntialiasingTable(fontName:String, fontStyle:String, colorType:String, advancedAntialiasingTable:Array) : Void	フォントのカスタム CSM (Continuous Stroke Modulation) ルックアップテーブルを設定します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

displayMode (TextRenderer.displayMode プロパティ)

```
public static displayMode : String
```

高度なアンチエイリアス処理をしたテキストのレンダリングを制御します。テキストの表示品質は主観的です。Flash Player がさまざまな条件に最適な設定を使用しようとしても、デザイナーは別の外観や操作性をテキストに対して選択する場合があります。また、displayMode を使用すると、デザイナーは Flash Player のサブピクセル選択をオーバーライドし、ユーザーのハードウェアに依存しない表示の一貫性を実現できます。

デザイナーは、高度なアンチエイリアスに対応したグローバルサブピクセルレンダリングモードの設定に以下の displayMode の値を使用することができます。

- "default" は、Flash Player で LCD モードまたは CRT モードを選択できるようにします。
- "lcd" は、強制的に LCD サブピクセルアンチエイリアスを使用します。フォントおよびハードウェアによっては、この設定によって非常に高い解像度のテキストまたはテキストの " 色付け " が行われる場合があります。
- "crt" は、強制的にグレースケールアンチエイリアスを表示します。この設定によってテキストの " 色付け " は回避されますが、" ぼやけて " 表示されると感じるユーザーもいます。

メモ： このプロパティは ActionScript 2.0 で使用できますが、使用できるのは、Flash Player 9 以降にパブリッシュする場合のみです。

対応バージョン： ActionScript 2.0、Flash Player 9

maxLevel (TextRenderer.maxLevel プロパティ)

```
public static maxLevel : Number
```

高度なアンチエイリアス用の ADF 品質レベルを設定します。有効な値は 3、4、7 のみです。デフォルトは 4 です。

高度なアンチエイリアスは、ADF を使用して、字形を決定するアウトラインを表します。品質が高くなるほど、ADF 構造用により多くのキャッシュ領域が必要となります。値 3 を指定すると、必要なメモリ量は最小で、品質は最低となります。フォントが大きければ、より多くのキャッシュ領域が必要となります。フォントサイズが 64 ピクセルの場合、品質レベルは 3 から 4、または 4 から 7 (レベルが既に 7 に設定されている場合を除く) に上がります。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、SWF ファイル全体に対して maxLevel 値を指定し、その値が設定されたテキストフィールドを表示します。この例でテキストを正しく表示するには、リンケージ識別子 "CustomFont" のフォントシンボルが使用可能である必要があります。

```
import flash.text.TextRenderer;
TextRenderer.maxLevel = 3;

var txtFormat:TextFormat = new TextFormat();
txtFormat.font = "CustomFont";
txtFormat.size = 64;

var label:TextField = this.createTextField("label", this.getNextHighestDepth(),
  10, 10, 500, 100);
label.setNewTextFormat(txtFormat);
label.text = "Hello World";
label.embedFonts = true;
trace("TextRenderer.maxLevel: " + TextRenderer.maxLevel);
```

setAdvancedAntialiasingTable (TextRenderer.setAdvancedAntialiasingTable メソッド)

```
public static setAdvancedAntialiasingTable(fontName:String, fontStyle:String,
  colorType:String, advancedAntialiasingTable:Array) : Void
```

フォントのカスタム CSM (Continuous Stroke Modulation) ルックアップテーブルを設定します。Flash Player は、フォントに最適な CSM を検出しようとします。Flash Player で提供される CSM が適切ではない場合、setAdvancedAntiAliasingTable() メソッドを使用して独自の CSM をカスタマイズできます。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

fontName:String - 設定を適用するフォントの名前。

fontStyle:String - 使用できるフォントスタイルは "bold"、"bolditalic"、"italic"、"none"。

colorType:String - この値は "dark" または "light" のいずれか。

advancedAntialiasingTable:Array - 指定されたフォントの GSM 設定の配列。各設定は、次のプロパティを持つオブジェクトです。

- fontSize
- insideCutOff
- outsideCutOff

advancedAntialiasingTable 配列には、異なるフォントサイズの CSM を指定する複数の項目を含めることができます。例を参照してください。

fontSize は、設定が適用されるサイズです (ピクセル単位)。

高度なアンチエイリアスは、ADF (Adaptively sampled Distance Fields) を使用して、字形を決定するアウトラインを表します。Adobe Flash Player では、外部カットオフ値 (outsideCutOff : この値未満だと密度が 0 に設定される) と、その内部カットオフ値 (insideCutOff : この値を超過すると密度が最大密度値 (255 など) に設定される) が使用されます。これら 2 つのカットオフ値間でマッピング関数を実行すると、外部カットオフの 0 から内部カットオフの最大密度までの直線になります。

外部カットオフ値と内部カットオフ値を調整すると、線の太さとエッジのシャープネスに影響します。これら 2 つのパラメータ間の間隔は、標準のアンチエイリアスメソッドのフィルタ半径の 2 倍に相当します。間隔が狭ければ、エッジがシャープになり、間隔が広ければ、エッジがぼやけて、フィルタの適用度が高い状態になります。間隔が 0 の場合、結果の密度イメージは 2 層のビットマップとなります。間隔が非常に広ければ、結果の密度イメージのエッジは水彩画のようになります。

通常、ユーザーは、小さなポイントサイズではシャープでコントラストのはっきりしたエッジを、アニメーションテキストや大きなポイントサイズではぼやけたエッジを好みます。

通常、外部カットオフは負の値、内部カットオフは正の値で、中間点は 0 の近辺になります。これらのパラメータを調整して、中間点を負の無限大に向かってシフトすると、線の太さが増します。中間点を正の無限大に向かってシフトすると、線の太さは減ります。外部カットオフ値が常に内部カットオフ値以下になるようにする必要があります。

例

次の例では、2 つのアンチエイリアス項目と 2 つのテキストフィールドを作成して示します。この例が動作するには、リンケージ識別子が "myArial" の共有フォントが SWF ファイルに埋め込まれている必要があります。フォントを埋め込むには、次の手順を実行します。

- ライブラリを開きます。
- ライブラリの右上隅にある [ライブラリ] オプションメニューをクリックします。
- ポップアップメニューから [新しいフォント] を選択します。
- フォントの名前を myArial にします。
- [フォント] ポップアップメニューで [Arial] を選択します。
- [OK] をクリックします。
- 新しく作成したフォントを右クリックし、[リンケージ] を選択します。
- [ActionScript に書き出し] チェックボックスをオンにします。
- [OK] をクリックして、デフォルトの識別子 myArial を受け入れます。

```
import flash.text.TextRenderer;

var antiAliasEntry_1 = {fontSize:24, insideCutoff:1.61, outsideCutoff:-3.43};
var antiAliasEntry_2 = {fontSize:48, insideCutoff:0.8, outsideCutoff:-0.8};
var arialTable:Array = new Array(antiAliasEntry_1, antiAliasEntry_2);

var lbl_1:TextField = createLabel(0, 0, 300, 100, 24);
var lbl_2:TextField = createLabel(0, 100, 300, 100, 48);

TextRenderer.setAdvancedAntialiasingTable("Arial", "none", "dark", arialTable);

function createLabel(x:Number, y:Number, width:Number, height:Number,
  fontSize:Number):TextField {
  var depth:Number = this.getNextHighestDepth();

  var tmpTxt = this.createTextField("txt_" + depth, depth, x, y, width, height);
  tmpTxt.antiAliasType = "advanced";
  tmpTxt.gridFitType = "pixel";
  tmpTxt.border = true;
  tmpTxt.text = "Hello World";
  tmpTxt.embedFonts = true;
  tmpTxt.setTextFormat(getTextFormat(fontSize));
  return tmpTxt;
}
```

```
function getTextFormat(fontSize:Number):TextFormat {
  var tf:TextFormat = new TextFormat();
  tf.align = "center";
  tf.size = fontSize;
  tf.font = "myArial";
  return tf;
}
```

TextSnapshot

```
Object
  |
  +-TextSnapshot

public class TextSnapshot
extends Object
```

TextSnapshot オブジェクトを使用すると、ムービークリップ内の静止テキストを操作できます。たとえば、ダイナミックテキストでは不可能な高い精度でテキストをレイアウトできます。ただし、テキストへのアクセスは読み取り専用になります。

TextSnapshot オブジェクトはコンストラクタを使用して作成するのではなく、`MovieClip.getTextSnapshot()` メソッドで取得します。

対応バージョン： ActionScript 1.0、Flash Player 7 - SWF ファイルは、Flash Player 6 以降用にパブリッシュして、Flash Player 7 以降で再生する必要があります。

関連項目

getTextSnapshot (MovieClip.getTextSnapshot メソッド)

プロパティ一覧

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

メソッド一覧

オプション	署名	説明
	findText(startIndex:Number, textToFind:String, caseSensitive:Boolean) : Number	指定された TextSnapshot オブジェクト内を検索し、最初に (または startIndex の後に) 見つかった textToFind の位置を返します。
	getCount() : Number	TextSnapshot オブジェクト内の文字数を返します。
	getSelected(start:Number, [end:Number]) : Boolean	選択されたテキストが TextSnapshot オブジェクトの指定範囲に存在するかどうかを表すブール値を返します。
	getSelectedText ([includeLineEndings:Boolean]) : String	対応する TextSnapshot.setSelected() メソッドで指定されたすべての文字を含むストリングを返します。
	getText(start:Number, end:Number, [includeLineEndings:Boolean]) : String	start パラメータと end パラメータで指定されたすべての文字を含むストリングを返します。
	getTextRunInfo (beginIndex:Number, endIndex:Number) : Array	連続したテキストに関する情報を含むオブジェクトの配列を返します。
	hitTestTextNearPos (x:Number, y:Number, [closeDist:Number]) : Number	TextSnapshot オブジェクト内のどの文字が、TextSnapshot オブジェクトのテキストを含むムービークリップの指定座標 *x*、*y* 上またはその付近に存在するかを評価します。
	setSelectColor (color:Number) : Void	TextSnapshot.setSelected() メソッドで選択された文字を強調表示するときに使用する色を指定します。
	setSelected(start:Number, end:Number, select:Boolean) : Void	選択または選択解除する TextSnapshot オブジェクトの文字範囲を指定します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

findText (TextSnapshot.findText メソッド)

```
public findText(startIndex:Number, textToFind:String, caseSensitive:Boolean) :
  Number
```

指定された TextSnapshot オブジェクト内を検索し、最初に (または startIndex の後に) 見つかった textToFind の位置を返します。textToFind が見つからなかった場合は、-1 を返します。

対応バージョン : ActionScript 1.0、Flash Player 7 - SWF ファイルは、Flash Player 6 以降用にパブリッシュして、Flash Player 7 以降で再生する必要があります。

パラメータ

startIndex:Number - 指定されたテキストの検索対象となる、TextSnapshot テキスト内のインデックスの開始位置を指定します。

textToFind:String - 検索対象のテキスト。ストリングリテラル (引用符で囲む) または変数を指定します。

caseSensitive:Boolean - textToFind で検索する際に大文字と小文字を区別するかどうかを指定するブール値。区別する場合は true、区別しない場合は false です。

戻り値

Number - 指定されたテキストが最初に見つかった位置を示す、ゼロから始まるインデックス位置。テキストが見つからなかった場合は -1 を返します。

例

次に、このメソッドの使用方法を示します。このコードを使用するには、"TextSnapshot Example" というテキストを含む静止テキストフィールドをステージに配置してください。

```
var my_mc:MovieClip = this;
var my_snap:TextSnapshot = my_mc.getTextSnapshot();
var index1:Number = my_snap.findText(0, "Snap", true);
var index2:Number = my_snap.findText(0, "snap", true);
var index3:Number = my_snap.findText(0, "snap", false);
trace(index1); // 4
trace(index2); // -1
trace(index3); // 4
```

関連項目

getText (TextSnapshot.getText メソッド)

getCount (TextSnapshot.getCount メソッド)

```
public getCount() : Number
```

TextSnapshot オブジェクト内の文字数を返します。

対応バージョン : ActionScript 1.0、Flash Player 7 - SWF ファイルは、Flash Player 6 以降用にパブリッシュして、Flash Player 7 以降で再生する必要があります。

戻り値

`Number` - TextSnapshot オブジェクト内の文字数。

例

次の例は、TextSnapshot オブジェクト内の文字数を返す方法を示しています。このコードを使用するには、"TextSnapshot Example" というテキスト (およびこのテキストのみ) を含む静止テキストフィールドをステージに配置してください。

```
var my_mc:MovieClip = this;
var my_snap:TextSnapshot = my_mc.getTextSnapshot();
var count:Number = my_snap.getCount();
var theText:String = my_snap.getText(0, count, false);
trace(count); // 20
trace(theText); // TextSnapshot Example
```

関連項目

`getText (TextSnapshot.getText メソッド )`

getSelected (TextSnapshot.getSelected メソッド)

```
public getSelected(start:Number, [end:Number]) : Boolean
```

選択されたテキストが TextSnapshot オブジェクトの指定範囲に存在するかどうかを表すブール値を返します。

すべての文字を検索するには、`start` には 0 を指定し、`end` には `TextSnapshot.getCount()` (または十分に大きな数値) を指定します。1 つの文字だけを検索する場合は、`start` パラメータに 1 を加えた値を `end` パラメータとして指定します。

対応バージョン : ActionScript 1.0、Flash Player 7 - SWF ファイルは、Flash Player 6 以降用にパブリッシュして、Flash Player 7 以降で再生する必要があります。

パラメータ

start:Number - 検査する範囲の最初の文字を示すインデックス位置。start に指定できる値は、0 ～ TextSnapshot.getCount() - 1 です。start が負の値の場合、0 が使用されます。

end:Number (オプション) - 検査する最後の文字に 1 を加えたインデックス位置。end に指定できる値は、0 ～ TextSnapshot.getCount() です。end パラメータで指定されたインデックス位置の文字は、取り出されるストリングには含まれません。このパラメータを省略すると、TextSnapshot.getCount() が使用されます。end の値が start に指定された値と同じか小さい場合、start + 1 が使用されます。

戻り値

Boolean - 指定された範囲の少なくとも 1 文字が、対応する TextSnapshot.setSelected() メソッドで選択されているかどうかを示すブール値。選択されている場合は true、選択されていない場合は false です。

例

次に、このメソッドの使用方法を示します。このコードを使用するには、"TextSnapshot Example" というテキストを含む静止テキストフィールドをステージに配置してください。ライブラリで、静止テキストフィールドで使用するフォントを選択して、[ActionScript に書き出し] を選択します。タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var my_snap:TextSnapshot = this.getTextSnapshot();
var count:Number = my_snap.getCount();
my_snap.setSelected(0, 4, true);
my_snap.setSelected(1, 2, false);

var firstCharIsSelected:Boolean = my_snap.getSelected(0, 1);
var secondCharIsSelected:Boolean = my_snap.getSelected(1, 2);
trace(firstCharIsSelected); // true
trace(secondCharIsSelected); // false
```

関連項目

getCount (TextSnapshot.getCount メソッド), getText (TextSnapshot.getText メソッド), getSelectedText (TextSnapshot.getSelectedText メソッド), setSelected (TextSnapshot.setSelected メソッド)

getSelectedText (TextSnapshot.getSelectedText メソッド)

```
public getSelectedText([includeLineEndings:Boolean]) : String
```

対応する TextSnapshot.setSelected( ) メソッドで指定されたすべての文字を含むストリングを返します。TextSnapshot.setSelected( ) メソッドで文字が指定されない場合は、空のストリングが返されます。

includeLineEndings に true を指定した場合は、返されるストリングに改行文字 (newline) が挿入され、返されるストリングは入力範囲よりも長くなる場合があります。includeLineEndings に false を指定した場合、または省略した場合は、何の文字も追加されずに選択されたテキストが返されます。

対応バージョン : ActionScript 1.0、Flash Player 7 - SWF ファイルは、Flash Player 6 以降用にパブリッシュして、Flash Player 7 以降で再生する必要があります。

パラメータ

includeLineEndings:Boolean (オプション) - ブール値。戻り値のストリングに改行文字 (newline) を挿入するか (true)、または挿入しないか (false) を指定します。デフォルト値は false です。

戻り値

String - 対応する TextSnapshot.setSelected( ) メソッドで指定されたすべての文字を含むストリング。

例

次に、このメソッドの使用方法を示します。このコードを使用するには、"TextSnapshot Example" というテキストを含む静止テキストフィールドをステージに配置してください。ライブラリで、静止テキストフィールドで使用するフォントを選択して、[ActionScript に書き出し] を選択します。タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var my_snap:TextSnapshot = this.getTextSnapshot();
var count:Number = my_snap.getCount();
my_snap.setSelected(0, 4, true);
my_snap.setSelected(1, 2, false);

var theText:String = my_snap.getSelectedText(false);
trace(theText); // Text
```

SWF ファイルをテストすると、指定された文字の周りに色つきの矩形が表示されます。

関連項目

getSelected (TextSnapshot.getSelected メソッド),
setSelected (TextSnapshot.setSelected メソッド)

getText (TextSnapshot.getText メソッド)

public getText(start:Number, end:Number, [includeLineEndings:Boolean]) : String

start パラメータと end パラメータで指定されたすべての文字を含むストリングを返します。文字が指定されない場合は、空のストリングが返されます。

すべての文字を取得するには、start には 0 を指定し、end には TextSnapshot.getCount() (または十分に大きな数値) を指定します。1 つの文字だけを返す場合は、endIndex に beginIndex + 1 を指定します。

includeLineEndings に true を指定した場合は、返されたストリングの適当な位置に改行文字 (newline) が挿入され、返されるストリングは入力範囲よりも長くなる場合があります。includeLineEndings に false を指定した場合、または省略した場合は、何の文字も追加されずに選択されたテキストが返されます。

対応バージョン : ActionScript 1.0、Flash Player 7 - SWF ファイルは、Flash Player 6 以降用にパブリッシュして、Flash Player 7 以降で再生する必要があります。

パラメータ

start:Number - 返されるストリングに含まれる先頭文字位置を示す整数。start に指定できる値は、0 ～ TextSnapshot.getCount() - 1 です。start が負の値の場合、0 が使用されます。

end:Number - TextSnapshot オブジェクトの検査を終了する位置のインデックスに 1 を加えた整数。end に指定できる値は、0 ～ TextSnapshot.getCount() です。end パラメータで指定されたインデックス位置の文字は、取り出されるストリングには含まれません。このパラメータを省略すると、TextSnapshot.getCount() が使用されます。end の値が start に指定された値と同じか小さい場合、start + 1 が使用されます。

includeLineEndings:Boolean (オプション) - ブール値。戻り値のストリングに改行文字 (newline) を挿入するか (true)、または挿入しないか (false) を指定します。デフォルト値は false です。

戻り値

String - 指定された範囲の文字を含むストリング。指定された範囲に文字が存在しない場合は空のストリングを返します。

例

次の例は、指定された TextSnapshot オブジェクト内の文字数を返す方法を示しています。このコードを使用するには、"TextSnapshot Example" というテキストを含む静止テキストフィールドをステージに配置してください。

```
var my_mc:MovieClip = this;
var my_snap:TextSnapshot = my_mc.getTextSnapshot();
var count:Number = my_snap.getCount();
var theText:String = my_snap.getText(0, count, false);
trace(count); // 20
trace(theText); // TextSnapshot Example
```

関連項目

getCount (TextSnapshot.getCount メソッド), getSelectedText (TextSnapshot.getSelectedText メソッド)

getTextRunInfo (TextSnapshot.getTextRunInfo メソッド)

```
public getTextRunInfo(beginIndex:Number, endIndex:Number) : Array
```

連続したテキストに関する情報を含むオブジェクトの配列を返します。各オブジェクトは、2 つのメソッドパラメータで指定した文字範囲内の 1 つの文字に対応します。

メモ : 大きな範囲のテキストに対して getTextRunInfo() メソッドを使用すると、大きなオブジェクトを返すことができます。beginIndex パラメータおよび endIndex パラメータで定義するテキスト範囲を制限することをお勧めします。

対応バージョン : ActionScript 1.0、Flash Player 7 - SWF ファイルは、Flash Player 6 以降用にパブリッシュして、Flash Player 7r19 以降で再生する必要があります。

パラメータ

beginIndex:Number - 文字範囲内の最初の文字を示すインデックス値。

endIndex:Number - 文字範囲内の最後の文字を示すインデックス値。

戻り値

Array - 指定された範囲内の特定の文字に関する情報が含まれる個々のオブジェクトで構成されたオブジェクトの配列。各オブジェクトには、次のプロパティがあります。

- indexInRun– 選択した連続したテキストでなく、ストリング全体を基準とした文字のゼロから始まる整数のインデックスです。
- selected– 文字が選択されるかどうかを示すブール値です。選択される場合は true、選択されない場合は false です。

- font– 文字のフォント名です。
- color– 文字のカラーとアルファ (透明度) の値の組み合わせです。16 進数の最初の 2 桁はアルファ値を表し、残りの桁はカラー値を表します。例には、10 進数を 16 進数に変換するメソッドが含まれています。
- height– 文字の高さです (ピクセル単位)。
- matrix_a、matrix_b、matrix_c、matrix_d、matrix_tx、matrix_ty– 文字に関する図形変換を定義するマトリックス値です。通常、垂直テキストのマトリックスは [1 0 0 1 x y] の形式です。ここで、x と y は親ムービークリップ内の文字の位置です。テキストの高さは関係ありません。このマトリックスは親ムービークリップの座標系にあり、そのムービー自身または親ムービーでの変換を含みません。
- corner0x、corner0y、corner1x、corner1y、corner2x、corner2y、corner3x、corner3y– 親ムービークリップの座標系に基づく、文字の境界ボックスの頂点です。これらの値は、文字に使用するフォントが SWF ファイルに埋め込まれている場合にのみ使用できます。

例

次に、このメソッドの使用方法を示します。このコードを使用するには、"AB" というテキストを含む静止テキストフィールドをステージ上に作成してください。次の図のように、テキストフィールドを 45 度回転させ、2 番目の文字にはカラーが 0xFFFFFF、アルファが 50% の上付きのスタイルを設定します。

次のスクリプトは、テキストフィールド内の各文字の getTextRunInfo() プロパティを示しています。

```
var myTS:TextSnapshot = this.getTextSnapshot();
var myArray:Array = myTS["getTextRunInfo"](0, myTS.getCount());
for (var i = 0; i < myTS.getCount(); i++) {
  trace("indexInRun: " + myArray[i].indexInRun);
  trace("selected: " + myArray[i].selected);
  trace("font: " + myArray[i].font);
  trace("color: " + decToHex(myArray[i].color));
  trace("height: " + myArray[i].height);
  trace("matrix_a: " + myArray[i].matrix_a);
  trace("matrix_b: " + myArray[i].matrix_b);
  trace("matrix_c: " + myArray[i].matrix_c);
  trace("matrix_d: " + myArray[i].matrix_d);
  trace("matrix_ty: " + myArray[i].matrix_tx);
  trace("matrix_tx: " + myArray[i].matrix_ty);
  trace(" ");
}
```

```
function decToHex(dec:Number) {
   var hexString:String = "";
   if (dec > 15) {
      hexString = decToHex(Math.floor(dec / 16));
   }
   var hexDigit = dec - 16 * (Math.floor(dec / 16));
      if (hexDigit > 9) {
         hexDigit = String.fromCharCode(hexDigit + 55);
      }
      hexString = hexString + hexDigit;
      return hexString;
}
```

これにより次の出力が生成されます。

```
indexInRun: 0
selected: false
font: Times New Roman
color: FF000000
height: 28.6
matrix_a: 0.0316612236983293
matrix_b: 0.0385940558426864
matrix_c: -0.0385940558426864
matrix_d: 0.0316612236983293
matrix_ty: 22.75
matrix_tx: 40.35

indexInRun: 0
selected: false
font: Times New Roman
color: 80000000
height: 28.6
matrix_a: 0.0316612236983293
matrix_b: 0.0385940558426864
matrix_c: -0.0385940558426864
matrix_d: 0.0316612236983293
matrix_ty: 49
matrix_tx: 45.5
```

この例では、decToHex() メソッドを使用して color プロパティの 10 進値を 16 進値に変換します。

関連項目

Matrix (flash.geom.Matrix)

hitTestTextNearPos (TextSnapshot.hitTestTextNearPos メソッド)

```
public hitTestTextNearPos(x:Number, y:Number, [closeDist:Number]) : Number
```

TextSnapshot オブジェクト内のどの文字が、TextSnapshot オブジェクトのテキストを含むムービークリップの指定座標 *x*、*y* 上またはその付近に存在するかを評価します。

`closeDist` の値を省略した場合または 0 を指定した場合、*x*、*y* 座標で指定される位置は、TextSnapshot オブジェクトの境界ボックス内にある必要があります。

このメソッドは、文字メトリック情報を含むフォントを使用する場合にのみ正常に機能します。ただしデフォルトでは、静止テキストフィールドにはこの情報は含まれません。したがって、インデックス値の代わりに -1 が返されます。インデックス値が返されるように、Flash オーサリングツールで強制的にフォントの文字メトリック情報を含めることができます。これを行うには、そのフォントを使用するダイナミックテキストフィールドを追加して、そのダイナミックテキストフィールドで [文字オプション] を選択し、埋め込むフォントのアウトラインを 1 つ以上の文字に対して指定します。どの文字を指定しても、静止テキストフィールドでその文字が使用されていてもかまいません。

対応バージョン : ActionScript 1.0、Flash Player 7 - SWF ファイルは、Flash Player 6 以降用にパブリッシュして、Flash Player 7 以降で再生する必要があります。

パラメータ

`x:Number` - TextSnapshot オブジェクトのテキストを含むムービークリップの *x* 座標。

`y:Number` - TextSnapshot オブジェクトのテキストを含むムービークリップの *y* 座標。

`closeDist:Number` (オプション) - テキストの検索が可能な、*x*、*y* からの最大距離。この距離は、各文字の中心位置を基準にして評価されます。デフォルト値は 0 です。

戻り値

`Number` - 指定された *x*、*y* 座標に最も近い、TextSnapshot オブジェクトの文字のインデックス値。文字が見つからない場合、またはフォントに文字メトリック情報が含まれていない場合には -1 を返します。

例

次に、このメソッドの使用方法を示します。このコードを使用するには、"TextSnapshot Example" というテキストを含む静止テキストフィールドをステージに配置してください。ライブラリで、静止テキストフィールドで使用するフォントを選択して、[ActionScript に書き出し] を選択します。コードをテストするには、SWF ファイルを実行し、マウスポインタを画面のテキスト上に置いてください。

```
var my_ts:TextSnapshot = getTextSnapshot();
this.onMouseMove = function() {
  var hitIndex:Number = my_ts.hitTestTextNearPos(_xmouse, _ymouse, 0);
  my_ts.setSelected(0, my_ts.getCount(), false);
  if (hitIndex >= 0) {
    my_ts.setSelected(hitIndex, hitIndex + 1, true);
  }
};
```

関連項目

getTextSnapshot (MovieClip.getTextSnapshot メソッド), _x (MovieClip._x プロパティ), _y (MovieClip._y プロパティ)

setSelectColor (TextSnapshot.setSelectColor メソッド)

```
public setSelectColor(color:Number) : Void
```

TextSnapshot.setSelected() メソッドで選択された文字を強調表示するときに使用する色を指定します。常に透明色が使用されます。透明度値を指定することはできません。

このメソッドは、文字メトリック情報を含むフォントを使用する場合にのみ正常に機能します。ただしデフォルトでは、静止テキストフィールドにはこの情報は含まれません。したがって、インデックス値の代わりに -1 が返されます。インデックス値が返されるように、Flash オーサリングツールで強制的にフォントの文字メトリック情報を含めることができます。これを行うには、そのフォントを使用するダイナミックテキストフィールドを追加して、そのダイナミックテキストフィールドで [文字オプション] を選択し、埋め込むフォントのアウトラインを 1 つ以上の文字に対して指定します。どの文字を指定しても、静止テキストフィールドでその文字が使用されていてもかまいません。

対応バージョン : ActionScript 1.0、Flash Player 7 - SWF ファイルは、Flash Player 6 以降用にパブリッシュして、Flash Player 7 以降で再生する必要があります。

パラメータ

color:Number - 対応する TextSnapshot.setSelected() メソッドで選択された文字の境界に使用する色。0x *RRGGBB* 形式で指定します。

例

次に、このメソッドの使用方法を示します。このコードを使用するには、"TextSnapshot Example" というテキストを含む静止テキストフィールドをステージに配置してください。ライブラリで、静止テキストフィールドで使用するフォントを選択して、[ActionScript に書き出し] を選択します。タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var my_snap:TextSnapshot = this.getTextSnapshot();
var count:Number = my_snap.getCount();
my_snap.setSelectColor(0xFF0000);
my_snap.setSelected(0, 4, true);
my_snap.setSelected(1, 2, false);

var theText:String = my_snap.getSelectedText(false); // get the selected text
trace(theText); // Text
```

SWF ファイルをテストすると、指定された文字の周りに色つきの矩形が表示されます。

関連項目

setSelected (TextSnapshot.setSelected メソッド)

setSelected (TextSnapshot.setSelected メソッド)

```
public setSelected(start:Number, end:Number, select:Boolean) : Void
```

選択または選択解除する TextSnapshot オブジェクトの文字範囲を指定します。選択された文字は、文字の境界ボックスに合わせて背景色付きの矩形で描画されます。境界ボックスの色は、`TextSnapshot.setSelectColor()` で定義されます。

すべての文字を選択または選択解除するには、`start` には 0 を指定し、`end` には `TextSnapshot.getCount()` (または十分に大きな数値) を指定します。1 つの文字だけを指定する場合は、`endIndex` に `start + 1` を指定します。

文字の選択状態は個々にマーキングされるため、このメソッドを何回か呼び出すことで、複数の文字を選択できます。つまり、このメソッドを使用しても、既にこのメソッドで選択されている他の文字については選択解除されません。

このメソッドは、文字メトリック情報を含むフォントを使用する場合にのみ正常に機能します。デフォルトでは、静止テキストフィールドにはこの情報は含まれません。したがって、選択されたテキストが画面に選択状態で表示されない場合があります。選択されたテキスト全体が選択状態で表示されるように、Flash オーサリングツールで強制的にフォントの文字メトリック情報を含めることができます。そのためには、ライブラリで、静止テキストフィールドで使用するフォントを選択して、[ActionScript に書き出し] を選択します。

対応バージョン： ActionScript 1.0、Flash Player 7 - SWF ファイルは、Flash Player 6 以降用にパブリッシュし、Flash Player 7 以降で再生する必要があります。

パラメータ

start:Number - 選択する範囲の最初の文字の位置。start に指定できる値は、0 ～ TextSnapshot.getCount() - 1 です。start が負の値の場合、0 が使用されます。

end:Number - 検査する最後の文字のインデックスに 1 を加えた整数です。end に指定できる値は、0 ～ TextSnapshot.getCount() です。end パラメータで指定されたインデックス位置の文字は、取り出されるストリングには含まれません。このパラメータを省略すると、TextSnapshot.getCount() が使用されます。end の値が start に指定された値と同じか小さい場合、start + 1 が使用されます。

select:Boolean - テキストを選択するかどうかを指定するブール値。選択する場合は true、選択しない場合は false です。

例

次に、このメソッドの使用方法を示します。このコードを使用するには、"TextSnapshot Example" というテキストを含む静止テキストフィールドをステージに配置してください。ライブラリで、静止テキストフィールドで使用するフォントを選択して、[ActionScript に書き出し] を選択します。タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var my_snap:TextSnapshot = this.getTextSnapshot();
var count:Number = my_snap.getCount();
my_snap.setSelected(0, 4, true);
my_snap.setSelected(1, 2, false);

var theText:String = my_snap.getSelectedText(false);
trace(theText); // Text
```

関連項目

getCount (TextSnapshot.getCount メソッド)

Transform (flash.geom.Transform)

```
Object
  |
  +-flash.geom.Transform

public class Transform
extends Object
```

Transform クラスは、MovieClip オブジェクトに適用されるカラー変換と座標操作に関するデータを収集します。

Transform オブジェクトの取得は、通常、MovieClip オブジェクトの transform プロパティの値を取得することによって行います。

対応バージョン： ActionScript 1.0、Flash Player 8

関連項目

transform (MovieClip.transform プロパティ), ColorTransform (flash.geom.ColorTransform), Matrix (flash.geom.Matrix)

プロパティ一覧

オプション	プロパティ	説明
	colorTransform:ColorTransform	ムービークリップ内のカラーを全体的に調整する値を格納している ColorTransform オブジェクトです。
	concatenatedColorTransform:ColorTransform (読み取り専用)	このオブジェクトおよびルートレベルまでのすべての親オブジェクトに適用される、結合されたカラー変換を表す ColorTransform オブジェクトです。
	concatenatedMatrix:Matrix (読み取り専用)	このオブジェクトおよびルートレベルまでのそのすべての親オブジェクトに結合された変換マトリックスを表す Matrix オブジェクトです。
	matrix:Matrix	ムービークリップの拡大・縮小、回転、および変換に影響を与える値を格納している変換用 Matrix オブジェクトです。
	pixelBounds:Rectangle	ステージ上の MovieClip オブジェクトの境界を示す矩形を定義する Rectangle オブジェクトです。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター一覧

署名	説明
Transform(mc:MovieClip)	特定の MovieClip オブジェクトに割り当てる新しい Transform オブジェクトを作成します。

メソッド一覧

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

colorTransform (Transform.colorTransform プロパティ)

public colorTransform : ColorTransform

ムービークリップ内のカラーを全体的に調整する値を格納している ColorTransform オブジェクトです。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、ColorTransform オブジェクト blueColorTransform を Transform オブジェクト trans に適用します。この ColorTransform は、MovieClip rect のカラーを赤から青に変換します。

```
import flash.geom.Transform;
import flash.geom.ColorTransform;

var rect:MovieClip = createRectangle(20, 80, 0xFF0000);

var trans:Transform = new Transform(rect);
trace(trans.colorTransform);
// (redMultiplier=1, greenMultiplier=1, blueMultiplier=1, alphaMultiplier=1,
   redOffset=0, greenOffset=0, blueOffset=0, alphaOffset=0)

var blueColorTransform:ColorTransform = new ColorTransform(0, 1, 1, 1, 0, 0, 255,
   0);

rect.onPress = function() {
   trans.colorTransform = blueColorTransform;
   trace(trans.colorTransform);
   // (redMultiplier=0, greenMultiplier=1, blueMultiplier=1, alphaMultiplier=1,
   redOffset=0, greenOffset=0, blueOffset=255, alphaOffset=0)
}

function createRectangle(width:Number, height:Number, color:Number,
   scope:MovieClip):MovieClip {
   scope = (scope == undefined) ? this : scope;
   var depth:Number = scope.getNextHighestDepth();
```

```
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
    mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

関連項目

ColorTransform (flash.geom.ColorTransform)

concatenatedColorTransform (Transform.concatenatedColorTransform プロパティ)

public concatenatedColorTransform : ColorTransform (読み取り専用)

このオブジェクトおよびルートレベルまでのすべての親オブジェクトに適用される、結合されたカラー変換を表す ColorTransform オブジェクトです。異なるレベルで異なるカラー変換を適用した場合、それぞれの変換が連結されて、このプロパティ用の 1 つの ColorTransform オブジェクトになります。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、2 つの Transform オブジェクトを親と子の両方の MovieClip オブジェクトに適用します。blueColorTransform 変数が Transform オブジェクト parentTrans に適用され、親と子の両方の MovieClip オブジェクトのカラーが青に調整されます。parentTrans と childTrans がどのように組み合わされて child.concatenatedColorTransform となるかを確認できます。

```
import flash.geom.Transform;
import flash.geom.ColorTransform;

var parentRect:MovieClip = createRectangle(20, 80, 0xFF0000);
var childRect:MovieClip = createRectangle(10, 40, 0x00FF00, parentRect);

var parentTrans:Transform = new Transform(parentRect);
var childTrans:Transform = new Transform(childRect);

var blueColorTransform:ColorTransform = new ColorTransform(0, 1, 1, 1, 0, 0, 255,
  0);

parentTrans.colorTransform = blueColorTransform;

trace(childTrans.concatenatedColorTransform);
// (redMultiplier=0, greenMultiplier=1, blueMultiplier=1, alphaMultiplier=1,
  redOffset=0, greenOffset=0, blueOffset=255, alphaOffset=0)
trace(childTrans.colorTransform);
```

```
// (redMultiplier=1, greenMultiplier=1, blueMultiplier=1, alphaMultiplier=1,
   redOffset=0, greenOffset=0, blueOffset=0, alphaOffset=0)
trace(parentTrans.concatenatedColorTransform);
// (redMultiplier=0, greenMultiplier=1, blueMultiplier=1, alphaMultiplier=1,
   redOffset=0, greenOffset=0, blueOffset=255, alphaOffset=0)

function createRectangle(width:Number, height:Number, color:Number,
   scope:MovieClip):MovieClip {
   scope = (scope == undefined) ? this : scope;
   var depth:Number = scope.getNextHighestDepth();
   var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
   mc.beginFill(color);
   mc.lineTo(0, height);
   mc.lineTo(width, height);
   mc.lineTo(width, 0);
   mc.lineTo(0, 0);
   return mc;
}
```

関連項目

ColorTransform (flash.geom.ColorTransform)

concatenatedMatrix (Transform.concatenatedMatrix プロパティ)

public concatenatedMatrix : Matrix (読み取り専用)

このオブジェクトおよびルートレベルまでのそのすべての親オブジェクトに結合された変換マトリックスを表す Matrix オブジェクトです。異なるレベルで異なる変換マトリックスを適用した場合、それぞれのマトリックスが連結されて、このプロパティ用の 1 つのマトリックスになります。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、2 つの Transform オブジェクトを子と親の両方の MovieClip オブジェクトに適用します。scaleMatrix 変数が Transform オブジェクト parentTrans に適用され、親と子の両方の MovieClip オブジェクトが拡大 / 縮小されます。parentTrans と childTrans がどのように組み合わされて child.concatenatedMatrix となるかを確認できます。

```
import flash.geom.Transform;
import flash.geom.Matrix;

var parentRect:MovieClip = createRectangle(20, 80, 0xFF0000);
var childRect:MovieClip = createRectangle(10, 40, 0x00FF00, parentRect);

var parentTrans:Transform = new Transform(parentRect);
var childTrans:Transform = new Transform(childRect);
```

```
var scaleMatrix:Matrix = new Matrix();
scaleMatrix.scale(2, 2);

parentTrans.matrix = scaleMatrix;

trace(childTrans.concatenatedMatrix); // (a=2, b=0, c=0, d=2, tx=0, ty=0)
trace(childTrans.matrix); // (a=1, b=0, c=0, d=1, tx=0, ty=0)
trace(parentTrans.concatenatedMatrix); // (a=2, b=0, c=0, d=2, tx=0, ty=0)

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

matrix (Transform.matrix プロパティ)

```
public matrix : Matrix
```

ムービークリップの拡大・縮小、回転、および変換に影響を与える値を格納している変換用 Matrix オブジェクトです。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、Matrix オブジェクト `scaleMatrix` を Transform オブジェクト `trans` に適用します。この Matrix は、MovieClip `rect` を 2 倍に拡大します。

```
import flash.geom.Transform;
import flash.geom.Matrix;

var rect:MovieClip = createRectangle(20, 80, 0xFF0000);

var trans:Transform = new Transform(rect);
trace(trans.matrix); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

var scaleMatrix:Matrix = new Matrix();
scaleMatrix.scale(2, 2);

rect.onPress = function() {
  trans.matrix = scaleMatrix;
  trace(trans.matrix); // (a=2, b=0, c=0, d=2, tx=0, ty=0)
```

```
}

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

関連項目

Matrix (flash.geom.Matrix)

pixelBounds (Transform.pixelBounds プロパティ)

```
public pixelBounds : Rectangle
```

ステージ上の MovieClip オブジェクトの境界を示す矩形を定義する Rectangle オブジェクトです。

対応バージョン： ActionScript 1.0、Flash Player 8

例

次の例では、Transform オブジェクト trans を作成し、その pixelBounds プロパティをトレースします。pixelBounds は MovieClip オブジェクトの getBounds() メソッドおよび getRect() メソッドと同じ値を持つ境界ボックスを返すことに注意してください。

```
import flash.geom.Transform;

var rect:MovieClip = createRectangle(20, 80, 0xFF0000);
var trans:Transform = new Transform(rect);
trace(trans.pixelBounds); // (x=0, y=0, w=20, h=80)

var boundsObj:Object = rect.getBounds();
trace(boundsObj.xMin); // 0
trace(boundsObj.yMin); // 0
trace(boundsObj.xMax); // 20
trace(boundsObj.yMax); // 80

var rectObj:Object = rect.getRect();
trace(rectObj.xMin); // 0
trace(rectObj.yMin); // 0
trace(rectObj.xMax); // 20
trace(rectObj.yMax); // 80
```

```
function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

Transform コンストラクタ

```
public Transform(mc:MovieClip)
```

特定の MovieClip オブジェクトに割り当てる新しい Transform オブジェクトを作成します。

新しい Transform オブジェクトを作成すると、MovieClip オブジェクトの `transform` プロパティを取得して、そのオブジェクトを取得できます。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

`mc:MovieClip` - 新しい Transform オブジェクトを適用する MovieClip オブジェクト。

例

次の例では、Transform オブジェクト `trans` を作成して MovieClip rect に適用します。Transform オブジェクトの `trans` と `rect.transform` が同じ値を含んでいても、等しいと評価されないことを確認できます。

```
import flash.geom.Transform;

var rect:MovieClip = createRectangle(20, 80, 0xFF0000);

var trans:Transform = new Transform(rect);

trace(rect.transform == trans); // false

for(var i in trans) {
  trace(">> " + i + ": " + trans[i]);
  // >> pixelBounds: (x=0, y=0, w=20, h=80)
  // >> concatenatedColorTransform: (redMultiplier=1, greenMultiplier=1,
  blueMultiplier=1, alphaMultiplier=1, redOffset=0, greenOffset=0, blueOffset=0,
  alphaOffset=0)
  // >> colorTransform: (redMultiplier=1, greenMultiplier=1, blueMultiplier=1,
  alphaMultiplier=1, redOffset=0, greenOffset=0, blueOffset=0, alphaOffset=0)
  // >> concatenatedMatrix: (a=1, b=0, c=0, d=1, tx=0, ty=0)
```

```
  // >> matrix: (a=1, b=0, c=0, d=1, tx=0, ty=0)
}

for(var i in rect.transform) {
  trace(">> " + i + ": " + rect.transform[i]);
  // >> pixelBounds: (x=0, y=0, w=20, h=80)
  // >> concatenatedColorTransform: (redMultiplier=1, greenMultiplier=1,
  blueMultiplier=1, alphaMultiplier=1, redOffset=0, greenOffset=0, blueOffset=0,
  alphaOffset=0)
  // >> colorTransform: (redMultiplier=1, greenMultiplier=1, blueMultiplier=1,
  alphaMultiplier=1, redOffset=0, greenOffset=0, blueOffset=0, alphaOffset=0)
  // >> concatenatedMatrix: (a=1, b=0, c=0, d=1, tx=0, ty=0)
  // >> matrix: (a=1, b=0, c=0, d=1, tx=0, ty=0)
}

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

Video

```
Object
  |
  +-Video
```

```
public class Video
extends Object
```

Video クラスを使用すると、SWF ファイルに埋め込むことなく、ライブストリーミングビデオをステージ上に表示することができます。ビデオをキャプチャするには、`Camera.get()` を使用します。Flash Player 7 以降用にパブリッシュされたファイルでは、Video クラスを使用して、HTTP 経由またはローカルファイルシステムから Flash Video (FLV) ファイルを再生することもできます。詳細については、NetConnection クラスと NetStream クラスを参照してください。

Flash Player 7 では、Sorenson Spark ビデオコーデックでエンコーディングされた Flash ビデオ (FLV) をサポートしています。Flash Player 8 では、Sorenson または On2 VP6 コーデックでエンコーディングされた Flash ビデオ (FLV) をサポートし、アルファチャネルもサポートしています。On2 VP6 ビデオコーデックは、古いテクノロジよりも少ない帯域幅を使用し、追加の非ブロックフィルタとリンギング除去フィルタを提供します。

Flash コンテンツで、段階的なダウンロードまたは Flash Media Server を使って Flash ビデオを動的にロードする場合、ユーザーが Flash Player 8 を使ってコンテンツを表示する限り、Flash Player 8 用の SWF を再度パブリッシュせずに、On2 VP6 ビデオを使用できます。On2 VP6 ビデオを Flash SWF 6 または 7 でストリーム再生するかダウンロードすることで、Flash Player 8 用に SWF ファイルを再作成しなくて済みます。

コーデック	コンテンツ (SWF) のバージョン (パブリッシュするバージョン)	Flash Player のバージョン (再生に必要なバージョン)
Sorenson Spark	6	6, 7, 8
	7	7, 8
On2 VP6	6	8*
	7	8
	8	8

* Flash コンテンツで Flash ビデオ (FLV) を動的にロードしている場合、ユーザーが Flash Player 8 を使用してコンテンツを表示している限り、Flash Player 8 用の SWF を再度パブリッシュせずに、On2 VP6 ビデオを使用できます。On2 VP6 ビデオのパブリッシュと再生は、Flash Player 8 でのみサポートされています。

ビデオオブジェクトはムービークリップと同じように使用できます。ステージ上に配置する他のオブジェクトと同様に、ビデオオブジェクトの各種プロパティを制御できます。たとえば、_x および _y プロパティを使用してステージ上でビデオオブジェクトを移動したり、_height および _width プロパティを使用してサイズを変更することもできます。

ビデオストリームを表示するにはまず、ステージ上にビデオオブジェクトを配置します。次に Video.attachVideo() を使用し、ビデオオブジェクトにビデオストリームを割り当てます。

- [ライブラリ] パネルを表示していない場合は、[ウィンドウ]-[ライブラリ] を選択して表示します。
- [ライブラリ] パネルのタイトルバーの右側にあるオプションメニューをクリックして [新規ビデオ] を選択し、埋め込み Video オブジェクトをライブラリに追加します。
- Video オブジェクトをステージにドラッグし、プロパティインスペクタを使用して一意のインスタンス名 (my_video など) を設定します。"Video" という名前にはしないでください。

対応バージョン： ActionScript 1.0、Flash Player 6 - Flash Video (FLV) ファイルの再生は Flash Player 7 で追加された機能です。On2 VP6 コーデックとアルファチャネルが Flash Player 8 で使用できるようになりました。

関連項目

NetConnection, NetStream

プロパティー覧

オプション	プロパティ	説明
	_alpha:Number	指定された Video オブジェクトのアルファ透明度値を示します。
	deblocking:Number	ポストプロセッシング中に、デコードされたビデオに適用される非ブロッキングフィルタのタイプを示します。
	_height:Number	Video オブジェクトの高さをピクセル単位で示します。
	height:Number (読み取り専用)	ビデオストリームの高さをピクセル単位で指定する整数。
	_name:String	指定された Video オブジェクトのインスタンス名を示します。
	_parent:MovieClip	現在の Video オブジェクトを含むムービークリップまたはオブジェクトを示します。
	_rotation:Number	Video オブジェクトの元の位置からの回転角を度単位で示します。
	smoothing:Boolean	ビデオを拡大・縮小する際にスムージング (補間) するかどうかを指定します。
	_visible:Boolean	*my_video* で指定された Video オブジェクトを表示するかどうかを示します。
	_width:Number	Video オブジェクトの幅をピクセル単位で示します。
	width:Number (読み取り専用)	ビデオストリームの幅をピクセル単位で指定する整数。
	_x:Number	親ムービークリップのローカル座標を基準にした Video オブジェクトの x 座標を示します。
	_xmouse:Number (読み取り専用)	マウス位置の *x* 座標を示します。
	_xscale:Number	Video オブジェクトの基準点から適用されるように Video オブジェクトの水平スケール (*percentage*) を示します。
	_y:Number	親ムービークリップのローカル座標を基準にした Video オブジェクトの y 座標を示します。
	_ymouse:Number (読み取り専用)	マウス位置の y 座標を示します。
	_yscale:Number	Video オブジェクトの基準点から適用されるように Video オブジェクトの垂直スケール (*percentage*) を示します。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

メソッド一覧

オプション	署名	説明
	attachVideo(source: Object) : Void	ステージ上の Video オブジェクトの境界内に表示するビデオストリーム (*source*) を指定します。
	clear() : Void	Video オブジェクトに現在表示されているイメージをクリアします。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

_alpha (Video._alpha プロパティ)

public _alpha : Number

指定された Video オブジェクトのアルファ透明度値を示します。有効な値は 0 (完全な透明) ～ 100 (完全な不透明) です。デフォルト値は 100 です。ムービークリップ内で _alpha が 0 に設定されているオブジェクトは、非表示の場合でもアクティブです。

対応バージョン : ActionScript 1.0、Flash Player 8

関連項目

_visible (Video._visible プロパティ)

attachVideo (Video.attachVideo メソッド)

```
public attachVideo(source:Object) : Void
```

ステージ上の Video オブジェクトの境界内に表示するビデオストリーム (*source*) を指定します。ビデオストリームは、NetStream.play() コマンドを使用して表示される FLV ファイル、Camera オブジェクト、または null のいずれかになります。*source* に null を指定すると、Video オブジェクト内でビデオが再生されなくなります。

FLV ファイルにオーディオしか格納されていない場合は、このメソッドを使用する必要はありません。FLV ファイルのオーディオは、NetStream.play() コマンドが発行されると自動的に再生されます。

FLV ファイルに関連付けられたオーディオを制御するには、MovieClip.attachAudio() を使用して、ムービークリップにオーディオをアタッチします。その後、Sound オブジェクトを作成することにより、オーディオのいくつかの特性を制御できます。詳細については、MovieClip.attachAudio() を参照してください。

対応バージョン : ActionScript 1.0、Flash Player 6 - Flash Video (FLV) ファイルが Flash Player 7 で使用できるようになりました。

パラメータ

`source:Object` - ビデオデータまたは NetStream オブジェクトをキャプチャする Camera オブジェクト。Video オブジェクトへの接続を閉じるには、*source* パラメータに null を指定します。

例

次の例では、ライブビデオをローカルに再生します。

```
var my_video:Video; //my_video is a Video object on the Stage
var active_cam:Camera = Camera.get();
my_video.attachVideo(active_cam);
```

次の例は、SWF ファイルと同じディレクトリに格納された、録画済みの "video1.flv" ファイルを再生します。

```
var my_video:Video; // my_video is a Video object on the Stage
var my_nc:NetConnection = new NetConnection();
my_nc.connect(null);
var my_ns:NetStream = new NetStream(my_nc);
my_video.attachVideo(my_ns);
my_ns.play("video1.flv");
```

関連項目

カメラ , NetStream

clear (Video.clear メソッド)

```
public clear() : Void
```

Video オブジェクトに現在表示されているイメージをクリアします。このメソッドは、たとえば Video オブジェクトを非表示にせずにスタンバイ情報を表示したい場合に便利です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ユーザーが pause_btn インスタンスをクリックしたときに、Video オブジェクト (my_video) 内で再生されている video1.flv を一時停止し、クリアします。

```
var pause_btn:Button;
var my_video:Video; // my_video is a Video object on the Stage
var my_nc:NetConnection = new NetConnection();
my_nc.connect(null);
var my_ns:NetStream = new NetStream(my_nc);
my_video.attachVideo(my_ns);
my_ns.play("video1.flv");
pause_btn.onRelease = function() {
  my_ns.pause();
  my_video.clear();
};
```

関連項目

attachVideo (Video.attachVideo メソッド)

deblocking (Video.deblocking プロパティ)

```
public deblocking : Number
```

ポストプロセッシング中に、デコードされたビデオに適用される非ブロッキングフィルタのタイプを示します。次の 2 つの非ブロックフィルタを使用できます。1 つは Sorenson コーデックにあり、もう 1 つは On2 VP6 コーデックにあります。許容値は次のとおりです。

- 0 (デフォルト) - ビデオコンプレッサは必要に応じて非ブロックフィルタを適用します。
- 1 - 非ブロックフィルタを使用しません。
- 2 - Sorenson 非ブロックフィルタを使用します。
- 3 - On2 非ブロックフィルタを使用し、リンギング除去フィルタは使用しません。
- 4 - On2 非ブロックフィルタと高速の On2 リンギング除去フィルタを使用します。
- 5 - On2 非ブロックフィルタとより良い On2 リンギング除去フィルタを使用します。
- 6 - 5 と同様。
- 7 - 5 と同様。

Sorenson コーデックの使用時にビデオに 2 より大きいモードを選択すると、Sorenson デコーダはデフォルトで内部的にモード 2 になります。

非ブロックフィルタの使用は全体的な再生のパフォーマンスに影響します。高帯域幅のビデオには、通常は必要ありません。このフィルタを有効にしたビデオは、低性能のシステムでは再生が困難なことがあります。

対応バージョン： ActionScript 1.0、Flash Player 6

例

次の例では、ビデオオブジェクト my_video 内の video1.flv を再生し、ユーザーによる video1.flv の非ブロックフィルタ動作の変更を許可します。ビデオオブジェクト my_video と ComboBox インスタンス deblocking_cb をファイルに追加してから、次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
var deblocking_cb:mx.controls.ComboBox;
var my_video:Video; // my_video is a Video object on the Stage
var my_nc:NetConnection = new NetConnection();
my_nc.connect(null);
var my_ns:NetStream = new NetStream(my_nc);
my_video.attachVideo(my_ns);
my_ns.play("video1.flv");

deblocking_cb.addItem({data:0, label:'Auto'});
deblocking_cb.addItem({data:1, label:'No'});
deblocking_cb.addItem({data:2, label:'Yes'});

var cbListener:Object = new Object();
cbListener.change = function(evt:Object) {
  my_video.deblocking = evt.target.selectedItem.data;
};
deblocking_cb.addEventListener("change", cbListener);
```

ComboBox インスタンスを使用して、video1.flv の非ブロックフィルタの動作を変更します。

_height (Video._height プロパティ)

```
public _height : Number
```

Video オブジェクトの高さをピクセル単位で示します。

対応バージョン： ActionScript 1.0、Flash Player 8

関連項目

_width (Video._width プロパティ)

height (Video.height プロパティ)

public height : Number [読み取り専用]

ビデオストリームの高さをピクセル単位で指定する整数。ライブストリームの場合、この値はビデオストリームをキャプチャしている Camera オブジェクトの Camera.height プロパティと同じです。FLV ファイルの場合、この値は FLV として書き出されたファイルの高さになります。

このプロパティは、ステージ上の実際の Video オブジェクトのサイズとは関係なく、キャプチャしたのと同じサイズでユーザーに対してビデオを表示する場合などに使用します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、FLV ファイルの height と width の値に合わせて、Video Symbol インスタンスのプロパティ _height と _width を設定します。

この例を使用するには、最初に "myVideo" というインスタンス名で新しい Video シンボルを作成し、このスクリプトと同じコンテキストに置きます。NetStream.Play.Start ステータスコードがトリガされると、height および width プロパティは 0 になります。NetStream.Buffer.Full のステータスの受け取り時にビデオのサイズを変更することで、ビデオのサイズを正しく設定してから、ビデオの先頭フレームが表示されるようにできます。

```
var netConn:NetConnection = new NetConnection();
netConn.connect(null);
var netStrm:NetStream = new NetStream(netConn);

myVideo.attachVideo(netStrm);
netStrm.play("Video.flv");

netStrm.onStatus = function(infoObject:Object) {
  switch (infoObject.code) {
    case 'NetStream.Play.Start' :
    case 'NetStream.Buffer.Full' :
      myVideo._width = myVideo.width;
      myVideo._height = myVideo.height;
      break;
  }
}
```

関連項目

_height (MovieClip._height プロパティ),width (Video.width プロパティ)

_name (Video._name プロパティ)

```
public _name : String
```

指定された Video オブジェクトのインスタンス名を示します。

対応バージョン : ActionScript 1.0、Flash Player 8

_parent (Video._parent プロパティ)

```
public _parent : MovieClip
```

現在の Video オブジェクトを含むムービークリップまたはオブジェクトを示します。現在のオブジェクトとは、_parent を参照する ActionScript コードがあるオブジェクトです。現在のオブジェクトの上位のムービークリップまたはオブジェクトへの相対パスを指定するには、_parent プロパティを使用します。

_parent を使用して表示リストの複数のレベルを上に移動するには、次のようにします。

```
this._parent._parent._alpha = 20;
```

対応バージョン : ActionScript 1.0、Flash Player 8

関連項目

_root プロパティ , _target (MovieClip._target プロパティ)

_rotation (Video._rotation プロパティ)

```
public _rotation : Number
```

Video オブジェクトの元の位置からの回転角を度単位で示します。時計回りに回転させる場合は 0 ～ 180 の値を指定します。反時計回りに回転させる場合は 0 ～ -180 の値を指定します。この範囲を超える値は、360 に加算または 360 から減算され、範囲内に収まる値になるように調整されます。たとえば、my_video._rotation = 450 というステートメントは my_video._rotation = 90 と同義です。

対応バージョン : ActionScript 1.0、Flash Player 8

smoothing (Video.smoothing プロパティ)

```
public smoothing : Boolean
```

ビデオを拡大・縮小する際にスムージング (補間) するかどうかを指定します。スムージングを行うには、Flash Player が高品質モードである必要があります。デフォルト値は false (スムージングなし) です。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、ボタン (smoothing_btn) を使用して、ビデオ my_video が SWF ファイル内で再生するときに、このビデオに適用される smoothing プロパティを切り替えます。ボタン smoothing_btn を作成し、次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
this.createTextField("smoothing_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
smoothing_txt.autoSize = true;

var my_nc:NetConnection = new NetConnection();
my_nc.connect(null);
var my_ns:NetStream = new NetStream(my_nc);
my_video.attachVideo(my_ns);
my_ns.play("video1.flv");
my_ns.onStatus = function(infoObject:Object) {
  updateSmoothing();
};
smoothing_btn.onRelease = function() {
  my_video.smoothing = !my_video.smoothing;
  updateSmoothing();
};
function updateSmoothing():Void {
  smoothing_txt.text = "smoothing = "+my_video.smoothing;
}
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

_visible (Video._visible プロパティ)

```
public _visible : Boolean
```

my_video で指定された Video オブジェクトを表示するかどうかを示します。

対応バージョン : ActionScript 1.0、Flash Player 8

_width (Video._width プロパティ)

```
public _width : Number
```

Video オブジェクトの幅をピクセル単位で示します。

対応バージョン : ActionScript 1.0、Flash Player 8 - 読み取り専用プロパティとして。

関連項目

_height (Video._height プロパティ)

width (Video.width プロパティ)

public width : Number (読み取り専用)

ビデオストリームの幅をピクセル単位で指定する整数。ライブストリームの場合、この値はビデオストリームをキャプチャしている Camera オブジェクトの Camera.width プロパティと同じです。FLV ファイルの場合、この値は FLV ファイルとして書き出されたファイルの幅になります。

このプロパティは、ステージ上の実際の Video オブジェクトのサイズとは関係なく、キャプチャしたのと同じサイズでユーザーに対してビデオを表示する場合などに使用します。

対応バージョン : ActionScript 1.0、Flash Player 6

例

Video.height の例を参照してください。

_x (Video._x プロパティ)

public _x : Number

親ムービークリップのローカル座標を基準にした Video オブジェクトの x 座標を示します。Video オブジェクトがメインのタイムラインにある場合、その座標系はステージの左上隅を (0,0) として参照します。変形されているムービークリップ内にある Video オブジェクトの座標系は、Video オブジェクトを囲むムービークリップのローカル座標系になります。したがって、反時計回りに 90 度回転したムービークリップの場合、そのムービークリップの子は、反時計回りに 90 度回転した座標系を継承します。Video オブジェクトの座標は、基準点の位置を参照します。

対応バージョン : ActionScript 1.0、Flash Player 8

関連項目

_xscale (Video._xscale プロパティ), _y (Video._y プロパティ), _yscale (Video._yscale プロパティ)

_xmouse (Video._xmouse プロパティ)

public _xmouse : Number (読み取り専用)

マウス位置の *x* 座標を示します。

対応バージョン : ActionScript 1.0、Flash Player 8

関連項目

Mouse, _ymouse (Video._ymouse プロパティ)

_xscale (Video._xscale プロパティ)

public _xscale : Number

Video オブジェクトの基準点から適用されるように Video オブジェクトの水平スケール (*percentage*) を示します。デフォルトの基準点は (0,0) です。

ローカル座標系を拡大・縮小すると、_x および _y プロパティの設定に影響します。この設定は、整数のピクセル数で表されます。

対応バージョン : ActionScript 1.0、Flash Player 8

関連項目

_x (Video._x プロパティ), _y (Video._y プロパティ), _yscale (Video._yscale プロパティ), _width (Video._width プロパティ)

_y (Video._y プロパティ)

public _y : Number

親ムービークリップのローカル座標を基準にした Video オブジェクトの y 座標を示します。Video オブジェクトがメインのタイムラインにある場合、その座標系はステージの左上隅を (0,0) として参照します。変形されているムービークリップ内にある Video オブジェクトの座標系は、Video オブジェクトを囲むムービークリップのローカル座標系になります。したがって、反時計回りに 90 度回転したムービークリップの場合、そのムービークリップの子は、反時計回りに 90 度回転した座標系を継承します。Video オブジェクトの座標は、基準点の位置を参照します。

対応バージョン : ActionScript 1.0、Flash Player 8

関連項目

_x (Video._x プロパティ), _xscale (Video._xscale プロパティ), _yscale (Video._yscale プロパティ)

_ymouse (Video._ymouse プロパティ)

public _ymouse : Number (読み取り専用)

マウス位置の y 座標を示します。

対応バージョン : ActionScript 1.0、Flash Player 8

関連項目

Mouse, _xmouse (Video._xmouse プロパティ)

_yscale (Video._yscale プロパティ)

```
public _yscale : Number
```

Video オブジェクトの基準点から適用されるように Video オブジェクトの垂直スケール (*percentage*) を示します。デフォルトの基準点は (0,0) です。

ローカル座標系を拡大・縮小すると、_x および _y プロパティの設定に影響します。この設定は、整数のピクセル数で表されます。

対応バージョン : ActionScript 1.0、Flash Player 8

関連項目

_x (Video._x プロパティ), _xscale (Video._xscale プロパティ),
_y (Video._y プロパティ), _height (Video._height プロパティ)

XML

```
Object
  |
  +-XMLNode
    |
    +-XML
```

```
public class XML
extends XMLNode
```

XML ドキュメントツリーをロード、解析、送信、構築、および操作するには、XML クラスのメソッドとプロパティを使用します。

XML クラスのメソッドを呼び出す前に、コンストラクタ new XML() を使用して XML オブジェクトのインスタンスを作成する必要があります。

XML ドキュメントは、Flash では XML クラスで表現されます。階層ドキュメントの各エレメントが、XMLNode オブジェクトで表されます。

次のメソッドとプロパティについての情報は、XMLNode クラスを参照してください。appendChild()、attributes、childNodes、cloneNode()、firstChild、hasChildNodes()、insertBefore()、lastChild、nextSibling、nodeName、nodeType、nodeValue、parentNode、previousSibling、removeNode()、toString()。

『ActionScript リファレンスガイド』の前のバージョンでは、上記のメソッドとプロパティは、XML クラスで説明されていました。現在は XMLNode クラスで説明されています。

メモ : XML オブジェクトおよび XMLNode オブジェクトは、W3C DOM レベル 1 勧告 (http://www.w3.org/tr/1998/REC-DOM-Level-1-19981001/level-one-core.html) に準拠しています。この勧告では、Node インターフェイスおよび Document インターフェイスが規定されています。Document インターフェイスは Node インターフェイスから継承され、`createElement()` や `createTextNode()` などのメソッドが追加されています。ActionScript では、XML オブジェクトと XMLNode オブジェクトの機能の違いは、同様の方針で設計されています。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 6 ではネイティブオブジェクトになり、パフォーマンスが大幅に向上しました。

関連項目

appendChild (XMLNode.appendChild メソッド), attributes (XMLNode.attributes プロパティ), childNodes (XMLNode.childNodes プロパティ), cloneNode (XMLNode.cloneNode メソッド), firstChild (XMLNode.firstChild プロパティ), hasChildNodes (XMLNode.hasChildNodes メソッド), insertBefore (XMLNode.insertBefore メソッド), lastChild (XMLNode.lastChild プロパティ), nextSibling (XMLNode.nextSibling プロパティ), nodeName (XMLNode.nodeName プロパティ), nodeType (XMLNode.nodeType プロパティ), nodeValue (XMLNode.nodeValue プロパティ), parentNode (XMLNode.parentNode プロパティ), previousSibling (XMLNode.previousSibling プロパティ), removeNode (XMLNode.removeNode メソッド), toString (XMLNode.toString メソッド)

プロパティ一覧

オプション	プロパティ	説明
	contentType:String	XML.send() メソッドまたは XML.sendAndLoad() メソッドを呼び出したときにサーバーに送られる MIME コンテンツタイプ。
	docTypeDecl:String	XML ドキュメントの DOCTYPE 宣言についての情報を指定します。
	idMap:Object	id 属性が割り当てられた XML ファイル内のノードを含むオブジェクト。
	ignoreWhite:Boolean	デフォルト設定は false です。
	loaded:Boolean	XML ドキュメントが正常にロードされたかどうかを示すプロパティ。

オプション	プロパティ	説明
	status:Number	XML ドキュメントが XML オブジェクトに正常に解析されたかどうかを示す数値を自動的に設定し、返します。
	xmlDecl:String	ドキュメントの XML 宣言についての情報を指定するストリング。

XMLNode クラスから継承されるプロパティ

attributes (XMLNode.attributes プロパティ), childNodes (XMLNode.childNodes プロパティ), firstChild (XMLNode.firstChild プロパティ), lastChild (XMLNode.lastChild プロパティ), localName (XMLNode.localName プロパティ), namespaceURI (XMLNode.namespaceURI プロパティ), nextSibling (XMLNode.nextSibling プロパティ), nodeName (XMLNode.nodeName プロパティ), nodeType (XMLNode.nodeType プロパティ), nodeValue (XMLNode.nodeValue プロパティ), parentNode (XMLNode.parentNode プロパティ), prefix (XMLNode.prefix プロパティ), previousSibling (XMLNode.previousSibling プロパティ)

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベントの一覧

イベント	説明
onData = function(src:String) {}	サーバーからの XML テキストのダウンロードが完了するか、サーバーからの XML テキストのダウンロード中にエラーが発生すると呼び出されます。
onHTTPStatus = function(httpStatus:Number) {}	Flash Player がサーバーから HTTP ステータスコードを受け取ると、呼び出されます。
onLoad = function(success:Boolean) {}	XML ドキュメントをサーバーから受信すると、Flash Player によって呼び出されます。

コンストラクター覧

署名	説明
XML(text:String)	新しい XML オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	addRequestHeader (header:Object, headerValue:String) : Void	POST アクションによって送信される HTTP リクエストヘッダ (Content-Type や SOAPAction など) を追加または変更します。
	createElement (name:String) : XMLNode	パラメータで指定された名前を持つ新しい XML エレメントを作成します。
	createTextNode (value:String) : XMLNode	指定されたテキストを持つ新しい XML テキストノードを作成します。
	getBytesLoaded() : Number	XML ドキュメントに対してロード (ストリーミング) されたバイト数を返します。
	getBytesTotal() : Number	XML ドキュメントのサイズをバイト単位で返します。
	load(url:String) : Boolean	指定された URL から XML ドキュメントを読み込み、指定された XML オブジェクトの内容をダウンロードされた XML データで置き換えます。
	parseXML(value:String) : Void	value パラメータで指定された XML テキストを解析し、指定された XML オブジェクトに XML ツリーを設定します。
	send(url:String, [target:String], [method:String]) : Boolean	指定された XML オブジェクトを XML ドキュメントにエンコードし、指定された target URL に送ります。
	sendAndLoad(url:String, resultXML:XML) : Void	指定された XML オブジェクトを XML ドキュメントにエンコードし、POST メソッドを使用して、指定された URL に送ります。さらに、サーバーの応答をダウンロードし、パラメータで指定された resultXMLobject にその応答をロードします。

XMLNode クラスから継承されるメソッド

appendChild (XMLNode.appendChild メソッド), cloneNode (XMLNode.cloneNode メソッド), getNamespaceForPrefix (XMLNode.getNamespaceForPrefix メソッド), getPrefixForNamespace (XMLNode.getPrefixForNamespace メソッド), hasChildNodes (XMLNode.hasChildNodes メソッド), insertBefore (XMLNode.insertBefore メソッド), removeNode (XMLNode.removeNode メソッド), toString (XMLNode.toString メソッド)

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

addRequestHeader (XML.addRequestHeader メソッド)

```
public addRequestHeader(header:Object, headerValue:String) : Void
```

POST アクションによって送信される HTTP リクエストヘッダ (Content-Type や SOAPAction など) を追加または変更します。シンタックス 1 では、2 つのストリング (header および headerValue) をメソッドに渡します。シンタックス 2 では、ヘッダー名とヘッダー値を交互に含むストリングの配列を渡します。

同じヘッダー名に対して複数の呼び出しを実行すると、呼び出しのたびに前の呼び出しで設定された値が上書きされます。

次に示すリクエストヘッダーは使用できません。また、制限されたこれらのストリングは、大文字と小文字が区別されません。したがって、たとえば Get、get、GET はすべて使用できません。

Accept-Charset, Accept-Encoding, Accept-Ranges, Age, Allow, Allowed, Connection, Content-Length, Content-Location, Content-Range, Date, Delete, ETag, Expect, Get, Host, Keep-Alive, Last-Modified, Location, Max-Forwards, Options, Post, Proxy-Authenticate, Proxy-Authorization, Public, Put, Range, Referer, Retry-After, Server, TE, Trace, Trailer, Transfer-Encoding, Upgrade, URI, User-Agent, Vary, Via, Warning, WWW-Authenticate, x-flash-version.

対応バージョン : ActionScript 1.0、Flash Player 6

パラメータ

header:Object - HTTP リクエストヘッダー名を表すストリング。

headerValue:String - header に関連付けられた値を表すストリング。

例

次の例では、SOAPAction というカスタム HTTP ヘッダーを my_xml という XML オブジェクトに追加します。値は Foo です。

```
my_xml.addRequestHeader("SOAPAction", "'Foo'");
```

次の例では、headers という配列を作成します。この配列には、HTTP ヘッダーとその値が交互に格納されます。この配列を addRequestHeader() メソッドにパラメータとして渡します。

```
var headers:Array = new Array("Content-Type", "text/plain", "X-
  ClientAppVersion", "2.0");
my_xml.addRequestHeader(headers);
```

関連項目

addRequestHeader (LoadVars.addRequestHeader メソッド)

contentType (XML.contentType プロパティ)

```
public contentType : String
```

XML.send() メソッドまたは XML.sendAndLoad() メソッドを呼び出したときにサーバーに送られる MIME コンテンツタイプ。デフォルト値は application/x-www-form-urlencoded です。これはほとんどの HTML フォームで使用される標準の MIME コンテンツタイプです。

対応バージョン : ActionScript 1.0、Flash Player 6

例

次の例では、新しい XML ドキュメントを作成し、そのドキュメントのデフォルトのコンテンツタイプを確認します。

```
// create a new XML document
var doc:XML = new XML();

// trace the default content type
trace(doc.contentType); // output: application/x-www-form-urlencoded
```

次の例では、XML パケットを定義し、XML オブジェクトのコンテンツタイプを設定します。その後データをサーバーに送り、ブラウザウィンドウに結果を表示します。

```
var my_xml:XML = new XML("<highscore><name>Ernie</name><score>13045</score></
  highscore>");
my_xml.contentType = "text/xml";
my_xml.send("http://www.flash-mx.com/mm/highscore.cfm", "_blank");
```

F12 キーを押して、ブラウザ内でこの例をテストします。

関連項目

send (XML.send メソッド),sendAndLoad (XML.sendAndLoad メソッド)

createElement (XML.createElement メソッド)

```
public createElement(name:String) : XMLNode
```

パラメータで指定された名前を持つ新しい XML エレメントを作成します。初期状態では、新しいエレメントには、親、子、および兄弟はありません。このメソッドは、エレメントとして新しく作成された XML オブジェクトへの参照を返します。このメソッドと `XML.createTextNode()` メソッドは、XML オブジェクトのノードを作成するためのコンストラクタメソッドです。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`name:String` - 作成する XML エレメントのタグ名。

戻り値

`XMLNode` - XMLNode オブジェクト。XML エレメントです。

例

次の例では、`createElement()` メソッドを使用して 3 つの XML ノードを作成します。

```
// create an XML document
var doc:XML = new XML();

// create three XML nodes using createElement()
var element1:XMLNode = doc.createElement("element1");
var element2:XMLNode = doc.createElement("element2");
var element3:XMLNode = doc.createElement("element3");

// place the new nodes into the XML tree
doc.appendChild(element1);
element1.appendChild(element2);
element1.appendChild(element3);

trace(doc);
// output: <element1><element2 /><element3 /></element1>
```

関連項目

createTextNode (XML.createTextNode メソッド)

createTextNode (XML.createTextNode メソッド)

```
public createTextNode(value:String) : XMLNode
```

指定されたテキストを持つ新しい XML テキストノードを作成します。初めは、新しいノードには親がありません。また、テキストノードは、子または兄弟を持つことができません。このメソッドは、新しいテキストノードを表す XML オブジェクトへの参照を返します。このメソッドと `XML.createElement()` メソッドは、XML オブジェクトのノードを作成するためのコンストラクタメソッドです。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`value:String` - ストリング。新しいテキストノードを作成するためのテキストです。

戻り値

`XMLNode` - XMLNode オブジェクト。

例

次の例では、`createTextNode()` メソッドを使用して XML テキストノードを 2 つ作成し、それらを既存の XML ノードに追加します。

```
// create an XML document
var doc:XML = new XML();

// create three XML nodes using createElement()
var element1:XMLNode = doc.createElement("element1");
var element2:XMLNode = doc.createElement("element2");
var element3:XMLNode = doc.createElement("element3");

// place the new nodes into the XML tree
doc.appendChild(element1);
element1.appendChild(element2);
element1.appendChild(element3);

// create two XML text nodes using createTextNode()
var textNode1:XMLNode = doc.createTextNode("textNode1 String value");
var textNode2:XMLNode = doc.createTextNode("textNode2 String value");

// place the new nodes into the XML tree
element2.appendChild(textNode1);
element3.appendChild(textNode2);
```

```
trace(doc);
// output (with line breaks added between tags):
// <element1>
// <element2>textNode1 String value</element2>
// <element3>textNode2 String value</element3>
// </element1>
```

関連項目

createElement (XML.createElement メソッド)

docTypeDecl (XML.docTypeDecl プロパティ)

```
public docTypeDecl : String
```

XML ドキュメントの DOCTYPE 宣言についての情報を指定します。XML テキストが XML オブジェクトに解析された後、XML オブジェクトの XML.docTypeDecl プロパティは、XML ドキュメントの DOCTYPE 宣言のテキスト (たとえば、<!DOCTYPE greeting SYSTEM "hello.dtd">) に設定されます。このプロパティは、XML ノードオブジェクトでなく DOCTYPE 宣言のストリング表現を使用して設定されます。

ActionScript の XML パーサーは、妥当性検証用パーサーではありません。DOCTYPE 宣言はパーサーにより読み取られ、XML.docTypeDecl プロパティに格納されますが、Dtd の妥当性検査は行われません。

解析中に DOCTYPE 宣言が見つからなかった場合、XML.docTypeDecl プロパティは undefined に設定されます。XML.toString() メソッドは、XML.xmlDecl に保存されている XML 宣言のすぐ後、XML オブジェクト内の他のテキストよりも前に、XML.docTypeDecl の内容を出力します。XML.docTypeDecl が undefined である場合、DOCTYPE 宣言は出力されません。

対応バージョン: ActionScript 1.0、Flash Player 5

例

次の例では、XML.docTypeDecl プロパティを使用して、XML オブジェクトに DOCTYPE 宣言を設定します。

```
my_xml.docTypeDecl = "<!DOCTYPE greeting SYSTEM \"hello.dtd\">";
```

関連項目

xmlDecl (XML.xmlDecl プロパティ)

getBytesLoaded (XML.getBytesLoaded メソッド)

```
public getBytesLoaded() : Number
```

XML ドキュメントに対してロード (ストリーミング) されたバイト数を返します。getBytesLoaded() の値と getBytesTotal() の値を比較すると、XML ドキュメントの何%がロードされたかを判断できます。

対応バージョン : ActionScript 1.0、Flash Player 6

戻り値

Number - ロードされたバイト数を示す整数。

例

次の例では、XML.getBytesLoaded() メソッドと XML.getBytesTotal() メソッドを使用して XML.load() コマンドの進捗状況を出力する方法を示します。XML.load() コマンドの URL パラメータは、HTTP を使用する有効な XML ファイルを指定するように値を置き換える必要があります。この例を使用してハードディスク上にあるローカルファイルをロードしようとしても、ムービープレビューモードの Flash Player ではローカルファイル全体がロードされるため、正常に機能しません。

```
// create a new XML document
var doc:XML = new XML();

var checkProgress = function(xmlObj:XML) {
  var bytesLoaded:Number = xmlObj.getBytesLoaded();
  var bytesTotal:Number = xmlObj.getBytesTotal();
  var percentLoaded:Number = Math.floor((bytesLoaded / bytesTotal ) 100);
  trace ("milliseconds elapsed: " + getTimer());
  trace ("bytesLoaded: " + bytesLoaded);
  trace ("bytesTotal: " + bytesTotal);
  trace ("percent loaded: " + percentLoaded);
  trace ("---------------------------------");
}

doc.onLoad = function(success:Boolean) {
  clearInterval(intervalID);
  trace("intervalID: " + intervalID);
}
doc.load("[place a valid URL pointing to an XML file here]");
var intervalID:Number = setInterval(checkProgress, 100, doc);
```

関連項目

getBytesTotal (XML.getBytesTotal メソッド)

getBytesTotal (XML.getBytesTotal メソッド)

```
public getBytesTotal() : Number
```

XML ドキュメントのサイズをバイト単位で返します。

対応バージョン : ActionScript 1.0、Flash Player 6

戻り値

Number - 整数。

例

XML.getBytesLoaded() の例を参照してください。

関連項目

getBytesLoaded (XML.getBytesLoaded メソッド)

idMap (XML.idMap プロパティ)

```
public idMap : Object
```

id 属性が割り当てられた XML ファイル内のノードを含むオブジェクト。ノードを格納する各オブジェクトのプロパティの名前は id 属性の値に一致します。

次のような XML オブジェクトがあるとします。

```
<employee id='41'>
<name>
John Doe
</name>
<address>
601 Townsend St.
</address>
</employee>

<employee id='42'>
<name>
Jane Q. Public
</name>
</employee>
<department id="IT">
Information Technology
</department>
```

この例では、この XML オブジェクトの idMap プロパティは、41、42、IT という 3 つのプロパティを持つオブジェクトです。これらのプロパティはそれぞれ対応する id 値を持つ XMLNode です。たとえば、idMap オブジェクトの IT プロパティは次のノードになります。

```
<department id="IT">
Information Technology
</department>
```

XML オブジェクトの parseXML() メソッドを使用して、idMap プロパティをインスタンス化する必要があります。

同じ id 値を持つ XMLNode が複数ある場合は、idNode オブジェクトの対応するプロパティは、次のように、解析された最後のノードのプロパティです。

```
var x1:XML = new XML("<a id='1'><b id='2' /><c id='1' /></a>");
x2 = new XML();
x2.parseXML(x1);
trace (x2.idMap['1']);
```

次のコードは <c> ノードを出力します。

```
<c id='1' />
```

対応バージョン： ActionScript 1.0、Flash Player 8

例

idMapTest.xml という名前のテキストファイルを作成し、そのファイルに次のテキストをコピーできます。

```
<?xml version="1.0"?>
<doc xml:base="http://example.org/today/" xmlns:xlink="http://www.w3.org/1999/
  xlink">
<head>
<title>Virtual Library</title>
</head>
<body>
<paragraph id="linkP1">See <link xlink:type="simple"
  xlink:href="new.xml">what's
new</link>!</paragraph>
<paragraph>Check out the hot picks of the day!</paragraph>
<olist xml:base="/hotpicks/">
<item>
<link id="foo" xlink:type="simple" xlink:href="pick1.xml">Hot Pick #1</link>
</item>
```

```
<item>
<link id="bar" xlink:type="simple" xlink:href="pick2.xml">Hot Pick #2</link>
</item>
<item>
<link xlink:type="simple" xlink:href="pick3.xml">Hot Pick #3</link>
</item>
</olist>
</body>
</doc>
```

その後、XML ファイルと同じディレクトリに SWF ファイルを作成できます。その SWF ファイルに次のスクリプトをコピーできます。

```
var readXML = new XML();
readXML.load("idMapTest.xml");
readXML.onLoad = function(success) {
  myXML = new XML();
  myXML.parseXML(readXML);
  for (var x in myXML.idMap){
    trace('idMap.' + x + " = " + newline + myXML.idMap[x]);
    trace('_____________' + newline);
  }
}
```

この SWF ファイルをテストすると、次のような出力が表示されます。

```
idMap.bar =
<link id="bar" xlink:type="simple" xlink:href="pick2.xml">Hot Pick #2</link>
_____________

idMap.foo =
<link id="foo" xlink:type="simple" xlink:href="pick1.xml">Hot Pick #1</link>
_____________

idMap.linkP1 =
<paragraph id="linkP1">See <link xlink:type="simple"
 xlink:href="new.xml">what's

new</link>!</paragraph>
_____________
```

ignoreWhite (XML.ignoreWhite プロパティ)

```
public ignoreWhite : Boolean
```

デフォルト設定は false です。true を設定すると、空白のみを含むテキストノードは解析処理中に破棄されます。先行空白または後続空白があるテキストノードは影響を受けません。

シンタックス 1: 次のコードに示すように、XML オブジェクトごとに ignoreWhite プロパティを設定できます。

```
my_xml.ignoreWhite = true;
```

シンタックス 2: 次のコードに示すように、XML オブジェクトにデフォルトの ignoreWhite プロパティを設定できます。

```
XML.prototype.ignoreWhite = true;
```

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、空白のみのテキストノードが含まれる XML ファイルをロードします。foyer タグは 14 個の空白文字で構成されます。この例を実行するには、"*flooring.xml* " という名前のテキストファイルを作成し、そのファイルに次のタグをコピーします。

```
<house>
<kitchen> ceramic tile </kitchen>
<bathroom>linoleum</bathroom>
<foyer> </foyer>
</house>
```

"*flooring.fla*" という名前で新しい Flash ドキュメントを作成し、それを XML ファイルと同じディレクトリに保存します。次のコードをメインタイムラインに配置します。

```
// Create a new XML object.
var flooring:XML = new XML();

// Set the ignoreWhite property to true (default value is false)
flooring.ignoreWhite = true;

// After loading is complete, trace the XML object.
flooring.onLoad = function(success:Boolean) {
  trace(flooring);
}

// Load the XML into the flooring object.
flooring.load("flooring.xml");
```

```
// Output (line breaks added for clarity):
<house>
  <kitchen> ceramic tile </kitchen>
  <bathroom>linoleum</bathroom>
  <foyer />
</house>
```

flooring.ignoreWhite の設定を false に変更するか、単純に flooring.ignoreWhite が含まれるコード行を削除する場合、foyer タグの 14 個の空白文字は削除されません。

```
...
// Set the ignoreWhite property to false (default value).
flooring.ignoreWhite = false;
...
// Output (line breaks added for clarity):
<house>
  <kitchen> ceramic tile </kitchen>
  <bathroom>linoleum</bathroom>
  <foyer> </foyer>
</house>
```

他の例については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0\XML_BlogTracker" フォルダに移動して XML_blogTracker.fla ファイルにアクセスし、"ActionScript2.0\XML_LanguagePicker" フォルダに移動して XML_languagePicker.fla ファイルにアクセスします。

load (XML.load メソッド)

```
public load(url:String) : Boolean
```

指定された URL から XML ドキュメントを読み込み、指定された XML オブジェクトの内容をダウンロードされた XML データで置き換えます。URL は相対 URL で、HTTP を使用して呼び出されます。ロード処理は非同期です。このため、load() メソッドの実行が完了しても、すぐには処理は終了しません。

load() メソッドを実行すると、XML オブジェクトの loaded プロパティが false に設定されます。XML データのダウンロードが終了すると、loaded プロパティが true に設定され、onLoad イベントハンドラが呼び出されます。XML データは、完全にダウンロードされるまで解析されません。XML オブジェクトに XML ツリーが既に含まれていた場合、その XML ツリーは破棄されます。

カスタム関数を定義して、XML オブジェクトの onLoad イベントハンドラが呼び出されたときに実行することができます。

メモ : ロード中のファイルに ASCII 文字以外の文字 (英語以外の多くの言語に存在する) が含まれている場合は、ASCII のような非 Unicode 形式ではなく UTF-8 または UTF-16 エンコーディング形式でファイルを保存することをお勧めします。

このメソッドを使用するときは、Flash Player セキュリティモデルを考慮してください。

Flash Player 8 :

- 呼び出し元 SWF ファイルが ローカルファイルシステムのサンドボックスにあり、ターゲットリソースがネットワーク上のサンドボックスにある場合、データをロードできません。
- 呼び出し元 SWF ファイルがネットワーク上のサンドボックスにあり、ターゲットリソースがローカルにある場合、データをロードできません。

詳細については、次の参照先を参照してください。

- 『ActionScript 2.0 の学習』の「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

Flash Player 7 以降では、クロスドメインポリシーファイルによって、Web サイトでのリソースへのクロスドメインアクセスを許可することができます。Flash Player 7 以降で SWF ファイルを実行している場合、`url` パラメータはまったく同じドメインに属している必要があります。たとえば、www.someDomain.com に存在する SWF ファイルは、store.someDomain.com に存在するソースからデータをロードできません。これは、両方のファイルが同じドメインに属していないためです。

Flash Player 7 より前のバージョンの Player で SWF ファイルを実行している場合、`url` パラメータは、呼び出し元の SWF ファイルと同じスーパードメインに属している必要があります。スーパードメインは、ファイルの URL の左端の要素を削除することで求められます。たとえば、www.someDomain.com に存在する SWF ファイルは、store.someDomain.com に存在するソースからデータをロードできます。これは、どちらのファイルも同じスーパードメイン someDomain.com に属しているためです。

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 7 ではビヘイビアが変更されました。

パラメータ

`url`:`String` - ロードする XML ドキュメントが置かれている場所の URL を表すストリング。呼び出し元の SWF ファイルが Web ブラウザで実行されている場合、`url` は SWF ファイルと同じドメインに属している必要があります。

戻り値

`Boolean` - ブール値。パラメータが指定されない (null が指定される) 場合は `false`、それ以外の場合は `true` が返されます。`onLoad()` イベントハンドラを使用して、XML ドキュメントが正常にロードされたかどうかをチェックします。

例

次のコードの例では、XML.load() メソッドを使用しています。

```
// Create a new XML object.
var flooring:XML = new XML();

// Set the ignoreWhite property to true (default value is false).
flooring.ignoreWhite = true;

// After loading is complete, trace the XML object.
flooring.onLoad = function(success) {
  trace(flooring);
};

// Load the XML into the flooring object.
flooring.load("flooring.xml");
```

flooring.xml ファイルの内容と、この例で生成される結果については、XML.ignoreWhite プロパティの例を参照してください。

関連項目

ignoreWhite (XML.ignoreWhite プロパティ), loaded (XML.loaded プロパティ), onLoad (XML.onLoad ハンドラ), useCodepage (System.useCodepage プロパティ)

loaded (XML.loaded プロパティ)

```
public loaded : Boolean
```

XML ドキュメントが正常にロードされたかどうかを示すプロパティ。XML オブジェクト用にカスタム onLoad() イベントハンドラが定義されていない場合、XML.load() の呼び出しで開始されたドキュメントのロード処理が正常に完了した場合は true になり、それ以外の場合は false になります。ただし、XML オブジェクトの onLoad() イベントハンドラのカスタムビヘイビアを定義した場合は、必ずその関数で onload を設定してください。

対応バージョン：ActionScript 1.0、Flash Player 5

例

次の例では、簡単なスクリプトで XML.loaded プロパティを使用します。

```
var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean) {
  trace("success: "+success);
  trace("loaded: "+my_xml.loaded);
  trace("status: "+my_xml.status);
};
my_xml.load("http://www.flash-mx.com/mm/problems/products.xml");
```

onLoad() ハンドラが呼び出されると、[出力] パネルに情報が表示されます。呼び出しが正常に終了すると、[出力] パネルに loaded ステータスが true と表示されます。

```
success: true
loaded: true
status: 0
```

関連項目

load (XML.load メソッド), onLoad (XML.onLoad ハンドラ)

onData (XML.onData ハンドラ)

```
onData = function(src:String) {}
```

サーバーからの XML テキストのダウンロードが完了するか、サーバーからの XML テキストのダウンロード中にエラーが発生すると呼び出されます。このハンドラは XML の解析前に呼び出されるため、このハンドラを使用して、Flash XML パーサーを使用する代わりにカスタム解析ルーチンを呼び出すことができます。src パラメータは、ダウンロード中にエラーが発生しなければ、サーバーからダウンロードされた XML テキストを含むストリングになります。エラーが発生した場合、src パラメータは undefined になります。

デフォルトでは、XML.onData イベントハンドラは XML.onLoad を呼び出します。XML.onData イベントハンドラを無効にしてカスタムビヘイビアを使用することができます。ただし、その場合、自分で実装する XML.onData で呼び出さない限り、XML.onLoad は呼び出されません。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

src:String - ストリングまたは undefined。サーバーから送られる生の (通常は XML 形式の) データです。

例

次の例では、XML.onData イベントハンドラのデフォルトの動作を示します。

```
XML.prototype.onData = function (src:String) {
  if (src == undefined) {
    this.onLoad(false);
  } else {
    this.parseXML(src);
    this.loaded = true;
    this.onLoad(true);
  }
}
```

XML.onData イベントハンドラをオーバーライド (上書き定義) して、XML テキストを解析せずに取得することができます。

関連項目

onLoad (XML.onLoad ハンドラ)

onHTTPStatus (XML.onHTTPStatus ハンドラ)

```
onHTTPStatus = function(httpStatus:Number) {}
```

Flash Player がサーバーから HTTP ステータスコードを受け取ると、呼び出されます。このハンドラを使用することにより、HTTP ステータスコードを取得してそれに基づいて処理を行えます。

onHTTPStatus ハンドラは onData ハンドラの前に呼び出され、ロードが失敗した場合には undefined 値を返して onLoad の呼び出しを実行します。onHTTPStatus がトリガされるたびに onData が続いてトリガされます。この動作は、onHTTPStatus をオーバーライドしているかどうかには関係ありません。onHTTPStatus ハンドラを最も効果的に使用するには、onHTTPStatus の呼び出しの結果を取得する適切な関数を記述します。その後、onData または onLoad ハンドラ関数の中で、その結果を使用できます。onHTTPStatus が呼び出されない場合は、URL リクエストが試みられなかったことを示します。これは、その要求が SWF ファイルのセキュリティサンドボックス規則に違反している場合に起こる可能性があります。

Flash Player がサーバーからステータスコードを取得できない場合、または Flash Player がサーバーと通信できない場合、デフォルト値 0 が ActionScript コードに渡されます。値 0 は、(たとえば正しくない形式の URL が要求された場合などに) どのプレーヤーでも生成される可能性があります。次のブラウザはサーバーからの HTTP ステータスコードを Flash Player に渡すことができないため、これらのブラウザで実行される Flash Player プラグインは常に値 0 を生成します。Netscape、Mozilla、Safari、Opera、Internet Explorer for Macintosh。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

`httpStatus:Number` - サーバーから返された HTTP ステータスコード。たとえば、値 404 は、要求された URI と一致するものがサーバーで見つからなかったことを示します。HTTP エラーの値は、ftp://ftp.isi.edu/in-notes/rfc2616.txt にある HTTP 仕様書のセクション 10.4 と 10.5 に記載されています。

例

次の例では、onHTTPStatus を使用してデバッグを効率的に行う方法を示します。まず、HTTP ステータスコードを収集し、それらの値とタイプを XML オブジェクトのインスタンスに渡します (この例では、インスタンスメンバー this.httpStatus と this.httpStatusType を実行時に作成します)。次に、onData メソッドはこれらの値とタイプを使用して、デバッグ時に役立つ可能性のある HTTP 応答に関する情報を出力します。

```
var myXml:XML = new XML();

myXml.onHTTPStatus = function(httpStatus:Number) {
  this.httpStatus = httpStatus;
  if(httpStatus < 100) {
    this.httpStatusType = "flashError";
  }
  else if(httpStatus < 200) {
    this.httpStatusType = "informational";
  }
  else if(httpStatus < 300) {
    this.httpStatusType = "successful";
  }
  else if(httpStatus < 400) {
    this.httpStatusType = "redirection";
  }
  else if(httpStatus < 500) {
    this.httpStatusType = "clientError";
  }
  else if(httpStatus < 600) {
    this.httpStatusType = "serverError";
  }
}

myXml.onData = function(src:String) {
  trace(">> " + this.httpStatusType + ": " + this.httpStatus);
  if(src != undefined) {
    this.parseXML(src);
    this.loaded = true;
    this.onLoad(true);
  }
```

```
  else {
    this.onLoad(false);
  }
}

myXml.onLoad = function(success:Boolean) {
}

myXml.load("http://blogs.adobe.com/mxna/xml/
  rss.cfm?query=byMostRecent&languages=1");
```

関連項目

onHTTPStatus (LoadVars.onHTTPStatus ハンドラ), load (XML.load メソッド), sendAndLoad (XML.sendAndLoad メソッド)

onLoad (XML.onLoad ハンドラ)

```
onLoad = function(success:Boolean) {}
```

XML ドキュメントをサーバーから受信すると、Flash Player によって呼び出されます。XML ドキュメントが正常に受信された場合、success パラメータは true です。ファイルが受信されなかったか、サーバーから応答を受信する際にエラーが発生した場合は、success パラメータには false が渡されます。デフォルトでは、このメソッドの実行形態は非アクティブになります。デフォルトの実行形態を無効にするには、カスタムアクションを含む関数を割り当てます。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

success:Boolean - XML.load() または XML.sendAndLoad() の処理によって XML オブジェクトが正常にロードされた場合は true、それ以外の場合は false になるブール値。

例

次の例には、簡単な電子商取引店頭アプリケーション用の ActionScript が含まれています。sendAndLoad() メソッドは、ユーザーの名前とパスワードを含む XML エレメントを転送し、XML.onLoad ハンドラを使用してサーバーからの応答を処理します。

```
var login_str:String = "<login username=\""+username_txt.text+"\"
  password=\""+password_txt.text+"\" />";
var my_xml:XML = new XML(login_str);
var myLoginReply_xml:XML = new XML();

myLoginReply_xml.ignoreWhite = true;

myLoginReply_xml.onLoad = function(success:Boolean){
```

```
  if (success) {

    if ((myLoginReply_xml.firstChild.nodeName == "packet") &&
      (myLoginReply_xml.firstChild.attributes.success == "true")) {
      gotoAndStop("loggedIn");
    } else {
      gotoAndStop("loginFailed");
    }

  } else {
    gotoAndStop("connectionFailed");
  }

};

my_xml.sendAndLoad("http://www.flash-mx.com/mm/login_xml.cfm",
  myLoginReply_xml);
```

関連項目

load (XML.load メソッド),sendAndLoad (XML.sendAndLoad メソッド),function ステートメント

parseXML (XML.parseXML メソッド)

```
public parseXML(value:String) : Void
```

value パラメータで指定された XML テキストを解析し、指定された XML オブジェクトに XML ツリーを設定します。XML オブジェクト内の既存のツリーはすべて破棄されます。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

value:String - 解析後に指定の XML オブジェクトに渡される XML テキストを表すストリング。

例

次の例では、XML パケットを作成して解析します。

```
var xml_str:String = "<state name=\"California\">
<city>San Francisco</city></state>"

// defining the XML source within the XML constructor:
var my1_xml:XML = new XML(xml_str);
trace(my1_xml.firstChild.attributes.name); // output: California

// defining the XML source using the XML.parseXML method:
var my2_xml:XML = new XML();
my2_xml.parseXML(xml_str);
trace(my2_xml.firstChild.attributes.name); // output: California
```

send (XML.send メソッド)

```
public send(url:String, [target:String], [method:String]) : Boolean
```

指定された XML オブジェクトを XML ドキュメントにエンコードし、指定された target URL に送ります。

このメソッドを使用するときは、Flash Player セキュリティモデルを考慮してください。

- Flash Player 8 では、呼び出し元 SWF ファイルが信頼されないコードとしてローカルのサンドボックスに置かれている場合、このメソッドは使用できません。
- Flash Player 7 以降では、呼び出し元 SWF ファイルがローカルファイルである場合、このメソッドは使用できません。

詳細については、次の参照先を参照してください。

- 『ActionScript 2.0 の学習』の「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`url:String` - 指定された XML オブジェクトの宛先 URL。

`target:String` (オプション) - サーバーが返したデータを表示するブラウザウィンドウ。

- `_self` は、現在のウィンドウ内の現在のフレームを指定します。
- `_blank` は、新規ウィンドウを指定します。
- `_parent` は、現在のフレームの親を指定します。
- `_top` 現在のウィンドウ内の最上位のフレームを指定します。

target パラメータを指定しない場合は、`_self` を指定したのと同じ結果になります。

`method:String` (オプション) - 使用される HTTP プロトコル。`"GET"` または `"POST"` のいずれかです。ブラウザでは、デフォルト値は `"POST"` です。Flash テスト環境では、デフォルト値は `"GET"` です。

戻り値

`Boolean` - パラメータが指定されていない場合は `false`、それ以外の場合は `true`。

例

次の例では、XML パケットを定義し、XML オブジェクトのコンテンツタイプを設定します。その後データをサーバーに送り、ブラウザウィンドウに結果を表示します。

```
var my_xml:XML = new XML("<highscore><name>Ernie</name><score>13045</score>
  </highscore>");
my_xml.contentType = "text/xml";
my_xml.send("http://www.flash-mx.com/mm/highscore.cfm", "_blank");
```

F12 キーを押して、ブラウザ内でこの例をテストします。

関連項目

sendAndLoad (XML.sendAndLoad メソッド)

sendAndLoad (XML.sendAndLoad メソッド)

```
public sendAndLoad(url:String, resultXML:XML) : Void
```

指定された XML オブジェクトを XML ドキュメントにエンコードし、POST メソッドを使用して、指定された URL に送ります。さらに、サーバーの応答をダウンロードし、パラメータで指定された resultXMLobject にその応答をロードします。サーバーの応答は、XML.load() メソッドで使用される方法と同じ方法でロードされます。

sendAndLoad() メソッドを実行すると、XML オブジェクトの loaded プロパティが false に設定されます。XML データのダウンロードが終了すると、データが正常にロードされた場合は loaded プロパティが true に設定され、onLoad イベントハンドラが呼び出されます。XML データは、完全にダウンロードされるまで解析されません。XML オブジェクトに XML ツリーが既に含まれていた場合、それらの XML ツリーは破棄されます。

このメソッドを使用するときは、Flash Player セキュリティモデルを考慮してください。

Flash Player 8 :

- 呼び出し元 SWF ファイルが ローカルファイルシステムのサンドボックスにあり、ターゲットリソースがネットワーク上のサンドボックスにある場合、データをロードできません。
- 呼び出し元 SWF ファイルがネットワーク上のサンドボックスにあり、ターゲットリソースがローカルにある場合、データをロードできません。

詳細については、次の参照先を参照してください。

- 『ActionScript 2.0 の学習』の「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

Flash Player 7 以降では、クロスドメインポリシーファイルによって、Web サイトでのリソースへのクロスドメインアクセスを許可することができます。Flash Player 7 以降で SWF ファイルを実行している場合、url パラメータはまったく同じドメインに属している必要があります。たとえば、www.someDomain.com に存在する SWF ファイルは、store.someDomain.com に存在するソースからデータをロードできません。これは、両方のファイルが完全に同じドメインに属していないためです。

Flash Player 7 より前のバージョンの Player で SWF ファイルを実行している場合、url パラメータは、呼び出し元の SWF ファイルと同じスーパードメインに属している必要があります。スーパードメインは、ファイルの URL の左端の要素を削除することで求められます。たとえば、www.someDomain.com に存在する SWF ファイルは、store.someDomain.com に存在するソースからデータをロードできます。これは、どちらのファイルも同じスーパードメイン someDomain.com に属しているためです。

対応バージョン： ActionScript 1.0、Flash Player 5 - Flash Player 7 ではビヘイビアが変更されました。

パラメータ

url:String - ストリング。指定された XML オブジェクトの宛先 URL です。呼び出し元の SWF ファイルが Web ブラウザで実行されている場合、url は SWF ファイルと同じドメインに属している必要があります。詳細については、「説明」を参照してください。

resultXML:XML - XML コンストラクタメソッドで作成されたターゲット XML オブジェクトです。このオブジェクトはサーバーから返される情報を受信します。

例

次の例には、簡単な電子商取引店頭アプリケーション用の ActionScript が含まれています。XML.sendAndLoad() メソッドは、ユーザーの名前とパスワードを含む XML エレメントを転送し、onLoad ハンドラを使用してサーバーからの応答を処理します。

```
var login_str:String = "<login username=\""+username_txt.text+"\"
  password=\""+password_txt.text+"\" />";
var my_xml:XML = new XML(login_str);
var myLoginReply_xml:XML = new XML();
myLoginReply_xml.ignoreWhite = true;
myLoginReply_xml.onLoad = myOnLoad;
my_xml.sendAndLoad("http://www.flash-mx.com/mm/login_xml.cfm",
  myLoginReply_xml);
function myOnLoad(success:Boolean) {
  if (success) {
    if ((myLoginReply_xml.firstChild.nodeName == "packet") &&
      (myLoginReply_xml.firstChild.attributes.success == "true")) {
      gotoAndStop("loggedIn");
    } else {
      gotoAndStop("loginFailed");
    }
  } else {
    gotoAndStop("connectionFailed");
  }
}
```

関連項目

send（XML.send メソッド）,load（XML.load メソッド）,loaded（XML.loaded プロパティ）,onLoad（XML.onLoad ハンドラ）

status (XML.status プロパティ)

public status : Number

XML ドキュメントが XML オブジェクトに正常に解析されたかどうかを示す数値を自動的に設定し、返します。数値ステータスコードとその説明を次に示します。

- 0 エラーなし。解析が正常に終了しました。
- -2 CDATA セクションが適切に終了されていません。
- -3 XML 宣言が適切に終了されていません。
- -4 DOCTYPE 宣言が適切に終了されていません。
- -5 コメントが適切に終了されていません。
- -6 XML エレメントの形式が正しくありませんでした。
- -7 メモリ不足です。
- -8 属性値が適切に終了されていません。
- -9 開始タグに対応する終了タグがありません。
- -10 対応する開始タグのない終了タグが見つかりました。

対応バージョン： ActionScript 1.0、Flash Player 5

例

次の例では、SWF ファイルに XML パケットをロードします。XML のロードと解析が成功したかどうかに応じて、ステータスメッセージが表示されます。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
var my_xml:XML = new XML();
my_xml.onLoad = function(success:Boolean) {
  if (success) {
  if (my_xml.status == 0) {
    trace("XML was loaded and parsed successfully");
  } else {
    trace("XML was loaded successfully, but was unable to be parsed.");
  }
  var errorMessage:String;
  switch (my_xml.status) {
  case 0 :
    errorMessage = "No error; parse was completed successfully.";
    break;
```

```
  case -2 :
    errorMessage = "A CDATA section was not properly terminated.";
    break;
  case -3 :
    errorMessage = "The XML declaration was not properly terminated.";
    break;
  case -4 :
    errorMessage = "The DOCTYPE declaration was not properly terminated.";
    break;
  case -5 :
    errorMessage = "A comment was not properly terminated.";
    break;
  case -6 :
    errorMessage = "An XML element was malformed.";
    break;
  case -7 :
    errorMessage = "Out of memory.";
    break;
  case -8 :
    errorMessage = "An attribute value was not properly terminated.";
    break;
  case -9 :
    errorMessage = "A start-tag was not matched with an end-tag.";
    break;
  case -10 :
    errorMessage = "An end-tag was encountered without a matching
    start-tag.";
    break;
  default :
    errorMessage = "An unknown error has occurred.";
    break;
  }
  trace("status: "+my_xml.status+" ("+errorMessage+")");
  } else {
  trace("Unable to load/parse XML. (status: "+my_xml.status+")");
  }
};
my_xml.load("http://www.helpexamples.com/flash/badxml.xml");
```

XML コンストラクタ

```
public XML(text:String)
```

新しい XML オブジェクトを作成します。XML クラスのメソッドを呼び出す前に、コンストラクタを使用して XML オブジェクトを作成する必要があります。

メモ : `createElement()` メソッドおよび `createTextNode()` メソッドを使用して、XML ドキュメントツリーにエレメントとテキストノードを追加します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`text:String` - 新しい XML オブジェクトを作成するために解析される XML テキスト。

例

次の例では、新しい空の XML オブジェクトを作成します。

```
var my_xml:XML = new XML();
```

次の例では、`source` パラメータで指定された XML テキストを解析することによって XML オブジェクトを作成し、新しく作成した XML オブジェクトに XML ドキュメントツリーを設定します。

```
var other_xml:XML = new XML("<state name=\"California\"><city>San Francisco</
  city></state>");
```

関連項目

createElement (XML.createElement メソッド), createTextNode (XML.createTextNode メソッド)

xmlDecl (XML.xmlDecl プロパティ)

```
public xmlDecl : String
```

ドキュメントの XML 宣言についての情報を指定するストリング。XML ドキュメントが XML オブジェクトに解析された後、このプロパティはドキュメントの XML 宣言のテキストに設定されます。このプロパティは、XML ノードオブジェクトでなく XML 宣言のストリング表現を使用して設定されます。解析中に XML 宣言が見つからなかった場合、プロパティは `null` に設定されます。`XML.toString()` メソッドは、XML オブジェクト内の他のテキストの前に、`XML.xmlDecl` プロパティの内容を出力します。`XML.xmlDecl` プロパティに `undefined` 型が含まれている場合、XML 宣言は出力されません。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、ステージと同じサイズのテキストフィールド my_txt を作成します。このテキストフィールドには、SWF ファイルにロードされる XML パケットのプロパティが表示されます。ドキュメントタイプ宣言が my_txt に表示されます。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.font = "_typewriter";
my_fmt.size = 12;
my_fmt.leftMargin = 10;

this.createTextField("my_txt", this.getNextHighestDepth(), 0, 0, Stage.width,
  Stage.height);
my_txt.border = true;
my_txt.multiline = true;
my_txt.wordWrap = true;
my_txt.setNewTextFormat(my_fmt);

var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean) {
  var endTime:Number = getTimer();
  var elapsedTime:Number = endTime-startTime;
  if (success) {
    my_txt.text = "xmlDecl:"+newline+my_xml.xmlDecl+newline+newline;
    my_txt.text += "contentType:"+newline+my_xml.contentType+newline+newline;
    my_txt.text += "docTypeDecl:"+newline+my_xml.docTypeDecl+newline+newline;
    my_txt.text += "packet:"+newline+my_xml.toString()+newline+newline;
  } else {
    my_txt.text = "Unable to load remote XML."+newline+newline;
  }
  my_txt.text += "loaded in: "+elapsedTime+" ms.";
};
my_xml.load("http://www.helpexamples.com/crossdomain.xml");
var startTime:Number = getTimer();
```

この例で使用している MovieClip.getNextHighestDepth() メソッドには Flash Player 7 以降が必要です。SWF ファイルにバージョン 2 のコンポーネントがある場合は、MovieClip.getNextHighestDepth() メソッドではなく、バージョン 2 のコンポーネントの DepthManager クラスを使用します。

関連項目

docTypeDecl (XML.docTypeDecl プロパティ)

XMLNode

```
Object
  |
  +-XMLNode
```

```
public class XMLNode
extends Object
```

XML ドキュメントは、Flash では XML クラスで表現されます。階層ドキュメントの各エレメントが、XMLNode オブジェクトで表されます。

対応バージョン： ActionScript 1.0、Flash Player 5

関連項目

XML

プロパティ一覧

オプション	プロパティ	説明
	attributes:Object	指定された XML インスタンスのすべての属性を含むオブジェクト。
	childNodes:Array (読み取り専用)	指定された XML オブジェクトの子の配列。
	firstChild:XMLNode (読み取り専用)	指定された XML オブジェクトを評価し、親ノードの子リスト内の最初の子を参照します。
	lastChild:XMLNode (読み取り専用)	ノードの子リスト内の最後の子を参照する XMLNode 値。
	localName:String (読み取り専用)	XML ノード名のローカル名部分。
	namespaceURI:String (読み取り専用)	XML ノードに接頭辞が付いている場合、namespaceURI はその接頭辞 (URI) の xmlns 宣言の値となります。これは通常、名前空間 URI と呼ばれます。
	nextSibling:XMLNode (読み取り専用)	親ノードの子リスト内の次の子ノードを参照する XMLNode 値。
	nodeName:String	XML オブジェクトのノード名を表すストリング。
	nodeType:Number (読み取り専用)	nodeType の値。XML エレメントの場合は 1、テキストノードの場合は 3 になります。
	nodeValue:String	XML オブジェクトのノード値。

オプション	プロパティ	説明
	parentNode:XMLNode (読み取り専用)	指定された XML オブジェクトの親ノードを参照する XMLNode 値です。ノードに親がない場合は null を返します。
	prefix:String (読み取り専用)	XML ノード名の接頭辞部分。
	previousSibling: XMLNode (読み取り専用)	親ノードの子リスト内の前の子ノードを参照する XMLNode 値。

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

コンストラクター一覧

署名	説明
XMLNode(type:Number, value:String)	XMLNode コンストラクタを使用すると、内容を指定するストリングとノードタイプを表す数値に基づいて、XML ノードをインスタンス化できます。

メソッド一覧

オプション	署名	説明
	appendChild(newChild: XMLNode) : Void	指定されたノードを XML オブジェクトの子リストに追加します。
	cloneNode (deep:Boolean) : XMLNode	指定された XML オブジェクトと同じタイプ、名前、値、および属性を持つ新しい XML ノードを作成し、返します。
	getNamespaceForPrefix (prefix:String) : String	ノードに指定された接頭辞に関連付けられている名前空間 URI を返します。
	getPrefixForNamespace (nsURI:String) : String	ノードに指定された名前空間 URI に関連付けられている接頭辞を返します。
	hasChildNodes() : Boolean	XML オブジェクトに子ノードがあるかどうかを指定します。

オプション	署名	説明
	insertBefore(newChild:XMLNode, insertPoint:XMLNode) : Void	insertPoint ノードの前に newChild ノードを XML オブジェクトの子リストに挿入します。
	removeNode() : Void	指定された XML オブジェクトをその親から削除します。
	toString() : String	指定された XML オブジェクトを評価し、ノード、子、および属性を含む XML 構造体のテキスト表現を作成し、結果をストリングとして返します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

appendChild (XMLNode.appendChild メソッド)

public appendChild(newChild:XMLNode) : Void

指定されたノードを XML オブジェクトの子リストに追加します。このメソッドは、パラメータ childNode で参照されるノードに直接実行されます。ノードのコピーが追加されるわけではありません。追加するノードが別のツリー構造内に既に存在している場合は、ノードを新しい場所に追加すると現在の場所からノードが削除されます。パラメータ childNode が、別の XML ツリー構造内に既に存在しているノードを参照している場合、追加される子ノードは、既存の親ノードから削除された後で新しいツリー構造に置かれます。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

newChild:XMLNode - 現在の場所から *my_xml* オブジェクトの子リストに移動されるノードを表す XMLNode。

例

この例では、次の処理を記載されている順序で実行します。

- 2 つの空の XML ドキュメント、doc1 および doc2 を作成します。
- createElement() メソッドを使用して新しいノードを作成し、appendChild() メソッドを使用してそのノードを XML ドキュメント doc1 に追加します。
- ルートノードを doc1 から doc2 に移動して、appendChild() メソッドを使用してノードを移動する方法を示します。
- doc2 からルートノードを複製し、そのノードを doc1 に追加します。
- 新しいノードを作成し、それを XML ドキュメント doc1 のルートノードに追加します。

```
var doc1:XML = new XML();
var doc2:XML = new XML();

// create a root node and add it to doc1
var rootnode:XMLNode = doc1.createElement("root");
doc1.appendChild(rootnode);
trace ("doc1: " + doc1); // output: doc1: <root />
trace ("doc2: " + doc2); // output: doc2:

// move the root node to doc2
doc2.appendChild(rootnode);
trace ("doc1: " + doc1); // output: doc1:
trace ("doc2: " + doc2); // output: doc2: <root />

// clone the root node and append it to doc1
var clone:XMLNode = doc2.firstChild.cloneNode(true);
doc1.appendChild(clone);
trace ("doc1: " + doc1); // output: doc1: <root />
trace ("doc2: " + doc2); // output: doc2: <root />

// create a new node to append to root node (named clone) of doc1
var newNode:XMLNode = doc1.createElement("newbie");
clone.appendChild(newNode);
trace ("doc1: " + doc1); // output: doc1: <root><newbie /></root>
```

attributes (XMLNode.attributes プロパティ)

```
public attributes : Object
```

指定された XML インスタンスのすべての属性を含むオブジェクト。XML.attributes オブジェクトには、XML インスタンスの各属性に対して 1 つずつ変数があります。この変数はオブジェクトの一部として定義されているので、通常はオブジェクトのプロパティとして参照されます。各属性の値は、対応するプロパティにストリングとして保存されます。たとえば、color という名前の属性がある場合、次のコードで示すように、color をプロパティ名として指定して、その属性の値を取得することができます。

```
var myColor:String = doc.firstChild.attributes.color;
```

対応バージョン： ActionScript 1.0、Flash Player 5

例

次の例では、XML ノードの属性を読み書きする方法を示します。

```
var doc:XML = new XML("<mytag name='Val'> item </mytag>");
trace(doc.firstChild.attributes.name); // Val

doc.firstChild.attributes.order = "first";
trace (doc.firstChild); // <mytag order="first" name="Val"> item </mytag>

for (attr in doc.firstChild.attributes) {
  trace (attr + " = " + doc.firstChild.attributes[attr]);
}

// order = first
// name = Val
```

childNodes (XMLNode.childNodes プロパティ)

```
public childNodes : Array ( 読み取り専用 )
```

指定された XML オブジェクトの子の配列。配列内の各エレメントは、子ノードを表す XML オブジェクトへの参照です。これは読み取り専用プロパティであり、子ノードを操作する場合には使用できません。子ノードを操作するには、appendChild()、insertBefore()、および removeNode() メソッドを使用します。

テキストノード (nodeType == 3) の場合、このプロパティは undefined になります。

対応バージョン： ActionScript 1.0、Flash Player 5

例

次の例では、XML.childNodes プロパティを使用して子ノードの配列を返す方法を示します。

```
// create a new XML document
var doc:XML = new XML();

// create a root node
var rootNode:XMLNode = doc.createElement("rootNode");

// create three child nodes
var oldest:XMLNode = doc.createElement("oldest");
var middle:XMLNode = doc.createElement("middle");
var youngest:XMLNode = doc.createElement("youngest");

// add the rootNode as the root of the XML document tree
doc.appendChild(rootNode);
```

```
// add each of the child nodes as children of rootNode
rootNode.appendChild(oldest);
rootNode.appendChild(middle);
rootNode.appendChild(youngest);

// create an array and use rootNode to populate it
var firstArray:Array = doc.childNodes;
trace (firstArray);
// output: <rootNode><oldest /><middle /><youngest /></rootNode>

// create another array and use the child nodes to populate it
var secondArray:Array = rootNode.childNodes;
trace(secondArray);
// output: <oldest />,<middle />,<youngest />
```

関連項目

nodeType (XMLNode.nodeType プロパティ), appendChild (XMLNode.appendChild メソッド), insertBefore (XMLNode.insertBefore メソッド), removeNode (XMLNode.removeNode メソッド)

cloneNode (XMLNode.cloneNode メソッド)

`public cloneNode(deep:Boolean) : XMLNode`

指定された XML オブジェクトと同じタイプ、名前、値、および属性を持つ新しい XML ノードを作成し、返します。deep に true を設定すると、すべての子ノードのクローンが再帰的に作成されるため、元のオブジェクトのドキュメントツリーが正確に複製されます。

返されたノードのクローンは、クローン作成元のアイテムのツリーとは関連がなくなります。その結果、nextSibling、parentNode、および previousSibling の値はすべて null になります。deep パラメータが false の場合、または *my_xml* ノードに子ノードがない場合は、firstChild と lastChild も null です。

対応バージョン：ActionScript 1.0、Flash Player 5

パラメータ

deep:Boolean - ブール値。true に設定した場合、指定された XML オブジェクトの子のクローンが再帰的に作成されます。

戻り値

XMLNode - XMLNode オブジェクト。

例

次の例では、XML.cloneNode() メソッドを使用してノードのコピーを作成する方法を示します。

```
// create a new XML document
var doc:XML = new XML();

// create a root node
var rootNode:XMLNode = doc.createElement("rootNode");

// create three child nodes
var oldest:XMLNode = doc.createElement("oldest");
var middle:XMLNode = doc.createElement("middle");
var youngest:XMLNode = doc.createElement("youngest");

// add the rootNode as the root of the XML document tree
doc.appendChild(rootNode);

// add each of the child nodes as children of rootNode
rootNode.appendChild(oldest);
rootNode.appendChild(middle);
rootNode.appendChild(youngest);

// create a copy of the middle node using cloneNode()
var middle2:XMLNode = middle.cloneNode(false);

// insert the clone node into rootNode between the middle and youngest nodes
rootNode.insertBefore(middle2, youngest);
trace(rootNode);
// output (with line breaks added):
// <rootNode>
// <oldest />
// <middle />
// <middle />
// <youngest />
// </rootNode>

// create a copy of rootNode using cloneNode() to demonstrate a deep copy
var rootClone:XMLNode = rootNode.cloneNode(true);

// insert the clone, which contains all child nodes, to rootNode
rootNode.appendChild(rootClone);
trace(rootNode);
// output (with line breaks added):
// <rootNode>
// <oldest />
// <middle />
// <middle />
// <youngest />
// <rootNode>
// <oldest />
```

```
// <middle />
// <middle />
// <youngest />
// </rootNode>
// </rootNode>
```

firstChild (XMLNode.firstChild プロパティ)

public firstChild : XMLNode (読み取り専用)

指定された XML オブジェクトを評価し、親ノードの子リスト内の最初の子を参照します。このプロパティは、ノードに子がないときは null です。ノードがテキストノードの場合、このプロパティは undefined です。これは読み取り専用プロパティなので、子ノードの操作には使用できません。子ノードを操作するには、appendChild() メソッド、insertBefore() メソッド、および removeNode() メソッドを使用します。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、XML.firstChild を使用してノードの子ノードをループ処理する方法を示します。

```
// create a new XML document
var doc:XML = new XML();

// create a root node
var rootNode:XMLNode = doc.createElement("rootNode");

// create three child nodes
var oldest:XMLNode = doc.createElement("oldest");
var middle:XMLNode = doc.createElement("middle");
var youngest:XMLNode = doc.createElement("youngest");

// add the rootNode as the root of the XML document tree
doc.appendChild(rootNode);

// add each of the child nodes as children of rootNode
rootNode.appendChild(oldest);
rootNode.appendChild(middle);
rootNode.appendChild(youngest);

// use firstChild to iterate through the child nodes of rootNode
for (var aNode:XMLNode = rootNode.firstChild; aNode != null; aNode =
  aNode.nextSibling) {
  trace(aNode);
}
```

```
// output:
// <oldest />
// <middle />
// <youngest />
```

次の例は、"Examples" ディレクトリの "XML_languagePicker" FLA ファイルから採られています。これは languageXML.onLoad イベントハンドラ関数定義にあります。

```
// loop through the strings in each language node
// adding each string as a new element in the language array
for (var stringNode:XMLNode = childNode.firstChild; stringNode != null;
  stringNode = stringNode.nextSibling, j++) {
  masterArray[i][j] = stringNode.firstChild.nodeValue;
}
```

スクリプト全体を表示するには、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/XML_LanguagePicker" フォルダに移動して XML_LanguagePicker.fla ファイルにアクセスします。

関連項目

appendChild (XMLNode.appendChild メソッド), insertBefore (XMLNode.insertBefore メソッド), removeNode (XMLNode.removeNode メソッド)

getNamespaceForPrefix (XMLNode.getNamespaceForPrefix メソッド)

```
public getNamespaceForPrefix(prefix:String) : String
```

ノードに指定された接頭辞に関連付けられている名前空間 URI を返します。URI を特定するために、getPrefixForNamespace() は XML 階層をノードから上方向に検索し、指定された prefix に関連付けられている、最初の xmlns 宣言の名前空間 URI を必要に応じて返します。

指定された接頭辞に対して定義されている名前空間がない場合は、null 値が返されます。

prefix として空のストリング ("") を指定する場合、ノードに対して定義されているデフォルトの名前空間 (たとえば xmlns="http://www.example.com/") があると、このメソッドはデフォルトの名前空間 URI を返します。

対応バージョン : ActionScript 1.0、Flash Player 8

パラメータ

prefix:String - このメソッドが返す名前空間に関連付けられている接頭辞。

戻り値

String - 指定された接頭辞に関連付けられている名前空間。

例

次の例では、新しい空の XML オブジェクトを作成し、getNamespaceForPrefix() の呼び出しの結果を出力します。

```
function createXML():XMLNode {
  var str:String = "<Outer xmlns:exu=\"http://www.example.com/util\">"
    + "<exu:Child id='1' />"
    + "<exu:Child id='2' />"
    + "<exu:Child id='3' />"
    + "</Outer>";
  return new XML(str).firstChild;
}

var xml:XMLNode = createXML();
trace(xml.getNamespaceForPrefix("exu")); // output: http://www.example.com/util
trace(xml.getNamespaceForPrefix("")); // output: null
```

関連項目

getPrefixForNamespace (XMLNode.getPrefixForNamespace メソッド), namespaceURI (XMLNode.namespaceURI プロパティ)

getPrefixForNamespace (XMLNode.getPrefixForNamespace メソッド)

```
public getPrefixForNamespace(nsURI:String) : String
```

ノードに指定された名前空間 URI に関連付けられている接頭辞を返します。接頭辞を特定するために、getPrefixForNamespace() は XML 階層をノードから上方向に検索し、nsURI に一致する名前空間 URI に関連付けられている、最初の xmlns 宣言の接頭辞を返します。

その URI に対応する xmlns 宣言がない場合、null が返されます。その URI に対応する xmlns 宣言はあるが、その宣言に関連付けられている接頭辞がない場合は、空のストリング ("") が返されます。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

nsURI:String - このメソッドが返す接頭辞に関連付けられている名前空間 URI。

戻り値

String - 指定された名前空間に関連付けられている接頭辞。

例

次の例では、新しい空の XML オブジェクトを作成し、getPrefixForNamespace() の呼び出しの結果を出力します。Outer XML ノードは、xmlDoc 変数で表され、名前空間 URI を定義し、その URI を exu 接頭辞に関連付けます。定義済みの名前空間 URI ("http://www.example.com/util") を指定して getPrefixForNamespace() を呼び出すと、接頭辞 exu が返されますが、未定義の URI ("http://www.example.com/other") を指定してこのメソッドを呼び出すと、null が返されます。最初の exu:Child ノードも、child1 変数によって表され、名前空間 URI("http://www.example.com/child") を定義しますが、その URI を接頭辞に関連付けません。定義されているが接頭辞に関連付けられていない名前空間 URI を指定してこのメソッドを呼び出すと、空のストリングが返されます。

```
function createXML():XMLNode {
  var str:String = "<Outer xmlns:exu=\"http://www.example.com/util\">"
    + "<exu:Child id='1' xmlns=\"http://www.example.com/child\"/>"
    + "<exu:Child id='2' />"
    + "<exu:Child id='3' />"
    + "</Outer>";
  return new XML(str).firstChild;
}

var xmlDoc:XMLNode = createXML();
trace(xmlDoc.getPrefixForNamespace("http://www.example.com/util")); // output:
  exu
trace(xmlDoc.getPrefixForNamespace("http://www.example.com/other")); // output:
  null

var child1:XMLNode = xmlDoc.firstChild;
trace(child1.getPrefixForNamespace("http://www.example.com/child")); // output:
  [empty string]
trace(child1.getPrefixForNamespace("http://www.example.com/other")); // output:
  null
```

関連項目

getNamespaceForPrefix (XMLNode.getNamespaceForPrefix メソッド), namespaceURI (XMLNode.namespaceURI プロパティ)

hasChildNodes (XMLNode.hasChildNodes メソッド)

```
public hasChildNodes() : Boolean
```

XML オブジェクトに子ノードがあるかどうかを指定します。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

Boolean - 指定された XMLNode オブジェクトに子ノードがある場合は true、それ以外の場合は false。

例

次の例では、新しい XML パケットを作成します。このコードは、ルートノードに子ノードがある場合は各子ノードをループして、そのノードの名前と値を表示します。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
var my_xml:XML = new XML("hankrudolph");
if (my_xml.firstChild.hasChildNodes()) {
// use firstChild to iterate through the child nodes of rootNode
  for (var aNode:XMLNode = my_xml.firstChild.firstChild; aNode != null;
  aNode=aNode.nextSibling) {
    if (aNode.nodeType == 1) {
      trace(aNode.nodeName+":\t"+aNode.firstChild.nodeValue);
    }
  }
}
```

次の情報が [出力] パネルに表示されます。

```
output:
username: hank
password: rudolph
```

insertBefore (XMLNode.insertBefore メソッド)

```
public insertBefore(newChild:XMLNode, insertPoint:XMLNode) : Void
```

`insertPoint` ノードの前に `newChild` ノードを XML オブジェクトの子リストに挿入します。`insertPoint` が XMLNode オブジェクトの子でない場合、挿入は失敗します。

対応バージョン： ActionScript 1.0、Flash Player 5

パラメータ

`newChild:XMLNode` - 挿入される XMLNode オブジェクト。

`insertPoint:XMLNode` - メソッドの呼び出し後に `newChild` ノードに従う XMLNode オブジェクト。

例

次の例では、2 つの既存のノード間に新しい XML ノードを挿入します。

```
var my_xml:XML = new XML("<a>1</a>\n<c>3</c>");
var insertPoint:XMLNode = my_xml.lastChild;
var newNode:XML = new XML("<b>2</b>\n");
my_xml.insertBefore(newNode, insertPoint);
trace(my_xml);
```

関連項目

XML, cloneNode (XMLNode.cloneNode メソッド)

lastChild (XMLNode.lastChild プロパティ)

public lastChild : XMLNode (読み取り専用)

ノードの子リスト内の最後の子を参照する XMLNode 値。XML.lastChild プロパティは、ノードに子がないときは null です。このプロパティは子ノードの操作には使用できません。子ノードを操作するには、appendChild() メソッド、insertBefore() メソッド、および removeNode() メソッドを使用します。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、XML.lastChild プロパティを使用して XML ノードの子ノードで繰り返し処理を実行します。この処理は、ノードの子リストの最後の項目から始まり、ノードの子リストの最初の子で終了します。

```
// create a new XML document
var doc:XML = new XML();

// create a root node
var rootNode:XMLNode = doc.createElement("rootNode");

// create three child nodes
var oldest:XMLNode = doc.createElement("oldest");
var middle:XMLNode = doc.createElement("middle");
var youngest:XMLNode = doc.createElement("youngest");

// add the rootNode as the root of the XML document tree
doc.appendChild(rootNode);

// add each of the child nodes as children of rootNode
rootNode.appendChild(oldest);
rootNode.appendChild(middle);
rootNode.appendChild(youngest);

// use lastChild to iterate through the child nodes of rootNode
for (var aNode:XMLNode = rootNode.lastChild; aNode != null; aNode =
  aNode.previousSibling) {
  trace(aNode);
}

// output:
// <youngest />
// <middle />
// <oldest />
```

次の例では、新しい XML パケットを作成し、XML.lastChild プロパティを使用して、ルートノードの子ノードで繰り返し処理を実行します。

```
// create a new XML document
var doc:XML = new XML("");

var rootNode:XMLNode = doc.firstChild;

// use lastChild to iterate through the child nodes of rootNode
for (var aNode:XMLNode = rootNode.lastChild; aNode != null;
  aNode=aNode.previousSibling) {
  trace(aNode);
}

// output:
// <youngest />
// <middle />
// <oldest />
```

関連項目

appendChild (XMLNode.appendChild メソッド), insertBefore (XMLNode.insertBefore メソッド), removeNode (XMLNode.removeNode メソッド), XML

localName (XMLNode.localName プロパティ)

public localName : String (読み取り専用)

XML ノード名のローカル名部分。名前空間の接頭辞を除いたエレメント名です。たとえば、ノード <contact:mailbox/>bob@example.com</contact:mailbox> には、ローカル名 "mailbox" と接頭辞 "contact" があり、この 2 つで完全なエレメント名 "contact.mailbox" が構成されます。

名前空間の接頭辞には、XML ノードオブジェクトの prefix プロパティ経由でアクセスできます。nodeName プロパティは、接頭辞とローカル名を含む完全な名前を返します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

この例では、同じディレクトリに格納されている SWF ファイルと XML ファイルを使用します。"SoapSample.xml" という名前の XML ファイルには、次のコードが含まれています。

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
<soap:Body xmlns:w="http://www.example.com/weather">
<w:GetTemperature>
<w:City>San Francisco</w:City>
</w:GetTemperature>
</soap:Body>
</soap:Envelope>
```

SWF ファイルのソースには、次のスクリプトが含まれています (Output ストリングのコメントに注意)。

```
var xmlDoc:XML = new XML()
xmlDoc.ignoreWhite = true;
xmlDoc.load("SoapSample.xml")
xmlDoc.onLoad = function(success:Boolean)
{
  var tempNode:XMLNode = xmlDoc.childNodes[0].childNodes[0].childNodes[0];
  trace("w:GetTemperature localname: " + tempNode.localName); // Output: ...
  GetTemperature
  var soapEnvNode:XMLNode = xmlDoc.childNodes[0];
  trace("soap:Envelope localname: " + soapEnvNode.localName); // Output: ...
  Envelope
}
```

namespaceURI (XMLNode.namespaceURI プロパティ)

```
public namespaceURI : String ( 読み取り専用 )
```

XML ノードに接頭辞が付いている場合、namespaceURI はその接頭辞 (URI) の xmlns 宣言の値となります。これは通常、名前空間 URI と呼ばれます。xmlns 宣言は、XML 階層で現在のノードか、それより上位のノードにあります。

XML ノードに接頭辞が付いていない場合、namespaceURI の値は、定義済みのデフォルトの名前空間 (たとえば xmlns="http://www.example.com/") があるかどうかによって異なります。デフォルト名前空間がある場合、namespaceURI プロパティの値はデフォルトの名前空間の値です。デフォルトの名前空間がない場合、そのノードの namespaceURI プロパティは空のストリング ("") です。

getNamespaceForPrefix() メソッドを使用すると、特定の接頭辞に関連付けられている名前空間を特定できます。namespaceURI プロパティは、ノード名に関連付けられている接頭辞を返します。

対応バージョン : ActionScript 1.0、Flash Player 8

例

次の例では、接頭辞の使用によって namespaceURI がどのように変わるのかを示します。SWF ファイルと XML ファイルは同じディレクトリに格納されています。XML ファイル SoapSample.xml には、次のタグが含まれています。

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
<soap:Body xmlns:w="http://www.example.com/weather">
<w:GetTemperature>
<w:City>San Francisco</w:City>
</w:GetTemperature>
</soap:Body>
</soap:Envelope>
```

SWF ファイルのソースには、次のスクリプトが含まれています (Output ストリングのコメントに注意)。tempNode の場合、これは w:GetTemperature ノードを表し、namespaceURI の値は soap:Body タグで定義されます。soapBodyNode の場合、これは soap:Body ノードを表し、namespaceURI の値は、1つ上のノードの soap 接頭辞の定義によって決まります。soap:Body ノードの w 接頭辞の定義によっては決まりません。

```
var xmlDoc:XML = new XML();
xmlDoc.load("SoapSample.xml");
xmlDoc.ignoreWhite = true;
xmlDoc.onLoad = function(success:Boolean)
{
  var tempNode:XMLNode = xmlDoc.childNodes[0].childNodes[0].childNodes[0];
  trace("w:GetTemperature namespaceURI: " + tempNode.namespaceURI);
    // Output: ... http://www.example.com/weather

  trace("w:GetTemperature soap namespace: " +
  tempNode.getNamespaceForPrefix("soap"));
    // Output: ... http://www.w3.org/2001/12/soap-envelope

  var soapBodyNode:XMLNode = xmlDoc.childNodes[0].childNodes[0];
  trace("soap:Envelope namespaceURI: " + soapBodyNode.namespaceURI);
    // Output: ... http://www.w3.org/2001/12/soap-envelope
}
```

次の例では、接頭辞の付いていない XML タグを使用します。また、同じディレクトリに格納されている SWF ファイルと XML ファイルを使用します。XML ファイル NoPrefix.xml には、次のタグが含まれています。

```
<?xml version="1.0"?>
<rootnode>
<simplenode xmlns="http://www.w3.org/2001/12/soap-envelope">
<innernode />
</simplenode>
</rootnode>
```

SWF ファイルのソースには、次のスクリプトが含まれています (Output ストリングのコメントに注意)。rootNode はデフォルトの名前空間を定義していないので、その namespaceURI 値は空のストリングです。simpleNode はデフォルトの名前空間を定義しているため、その namespaceURI 値はデフォルトの名前空間です。innerNode はデフォルトの名前空間を定義していませんが、simpleNode によって定義されるデフォルトの名前空間を使用します。そのため、namespaceURI 値は simpleNode 値と同じです。

```
var xmlDoc:XML = new XML()
xmlDoc.load("NoPrefix.xml");
xmlDoc.ignoreWhite = true;
xmlDoc.onLoad = function(success:Boolean)
{
```

```
  var rootNode:XMLNode = xmlDoc.childNodes[0];
  trace("rootNode Node namespaceURI: " + rootNode.namespaceURI);
    // Output: [empty string]

  var simpleNode:XMLNode = xmlDoc.childNodes[0].childNodes[0];
  trace("simpleNode Node namespaceURI: " + simpleNode.namespaceURI);
    // Output: ... http://www.w3.org/2001/12/soap-envelope

  var innerNode:XMLNode = xmlDoc.childNodes[0].childNodes[0].childNodes[0];
  trace("innerNode Node namespaceURI: " + innerNode.namespaceURI);
    // Output: ... http://www.w3.org/2001/12/soap-envelope
}
```

関連項目

getNamespaceForPrefix (XMLNode.getNamespaceForPrefix メソッド), getPrefixForNamespace (XMLNode.getPrefixForNamespace メソッド)

nextSibling (XMLNode.nextSibling プロパティ)

public nextSibling : XMLNode (読み取り専用)

親ノードの子リスト内の次の子ノードを参照する XMLNode 値。子リストに次のノードがない場合、このプロパティは null になります。このプロパティは子ノードの操作には使用できません。子ノードを操作するには、appendChild() メソッド、insertBefore() メソッド、および removeNode() メソッドを使用します。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次に示す例は、XML.firstChild プロパティの例から引用したもので、XML.nextSibling プロパティを使用して XML ノードの子ノードをループ処理する方法を示しています。

```
for (var aNode:XMLNode = rootNode.firstChild; aNode != null; aNode =
  aNode.nextSibling) {
  trace(aNode);
}
```

関連項目

firstChild (XMLNode.firstChild プロパティ), appendChild (XMLNode.appendChild メソッド), insertBefore (XMLNode.insertBefore メソッド), removeNode (XMLNode.removeNode メソッド), XML

nodeName (XMLNode.nodeName プロパティ)

```
public nodeName : String
```

XML オブジェクトのノード名を表すストリング。XML オブジェクトが XML エレメントである場合 (nodeType == 1)、nodeName は XML ファイル内のノードを表すタグの名前です。たとえば、TITLE は HTML TITLE タグの nodeName です。XML オブジェクトがテキストノードである (nodeType == 3) 場合、nodeName は null です。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、エレメントノードとテキストノードを作成し、それぞれのノード名をチェックします。

```
// create an XML document
var doc:XML = new XML();

// create an XML node using createElement()
var myNode:XMLNode = doc.createElement("rootNode");

// place the new node into the XML tree
doc.appendChild(myNode);

// create an XML text node using createTextNode()
var myTextNode:XMLNode = doc.createTextNode("textNode");

// place the new node into the XML tree
myNode.appendChild(myTextNode);

trace(myNode.nodeName);
trace(myTextNode.nodeName);

// output:
// rootNode
// null
```

次の例では、新しい XML パケットを作成します。このコードは、ルートノードに子ノードがある場合は各子ノードをループして、そのノードの名前と値を表示します。次の ActionScript を FLA ファイルまたは AS ファイルに追加します。

```
var my_xml:XML = new XML("hankrudolph");
if (my_xml.firstChild.hasChildNodes()) {
  // use firstChild to iterate through the child nodes of rootNode
  for (var aNode:XMLNode = my_xml.firstChild.firstChild; aNode != null;
  aNode=aNode.nextSibling) {
    if (aNode.nodeType == 1) {
      trace(aNode.nodeName+":\t"+aNode.firstChild.nodeValue);
    }
  }
}
```

次のノード名が [出力] パネルに表示されます。

```
output:
username: hank
password: rudolph
```

関連項目

nodeType（XMLNode.nodeType プロパティ）

nodeType (XMLNode.nodeType プロパティ)

public nodeType : Number（読み取り専用）

nodeType の値。XML エレメントの場合は 1、テキストノードの場合は 3 になります。

nodeType は、W3C DOM レベル 1 勧告 (www.w3.org/tr/1998/REC-DOM-Level-1-19981001/level-one-core.html) の NodeType の一覧に記載されている数値です。次の表は、これらの数値の一覧です。

整数値	定義されている定数
1	ELEMENT_NODE
2	ATTRIBUTE_NODE
3	TEXT_NODE
4	CDATA_SECTION_NODE
5	ENTITY_REFERENCE_NODE
6	ENTITY_NODE
7	PROCESSING_INSTRUCTION_NODE
8	COMMENT_NODE
9	DOCUMENT_NODE
10	DOCUMENT_TYPE_NODE
11	DOCUMENT_FRAGMENT_NODE
12	NOTATION_NODE

Flash Player に組み込まれている XML クラスがサポートするのは 1 (ELEMENT_NODE) と 3 (TEXT_NODE) だけです。

対応バージョン：ActionScript 1.0、Flash Player 5

例

次の例では、エレメントノードとテキストノードを作成し、それぞれのノードタイプをチェックします。

```
// create an XML document
var doc:XML = new XML();

// create an XML node using createElement()
var myNode:XMLNode = doc.createElement("rootNode");

// place the new node into the XML tree
doc.appendChild(myNode);

// create an XML text node using createTextNode()
var myTextNode:XMLNode = doc.createTextNode("textNode");

// place the new node into the XML tree
myNode.appendChild(myTextNode);

trace(myNode.nodeType);
trace(myTextNode.nodeType);

// output:
// 1
// 3
```

関連項目

nodeValue (XMLNode.nodeValue プロパティ)

nodeValue (XMLNode.nodeValue プロパティ)

```
public nodeValue : String
```

XML オブジェクトのノード値。XML オブジェクトがテキストノードである場合、nodeType は 3 であり、nodeValue はノードのテキストです。XML オブジェクトが XML エレメントである (nodeType が 1 の) 場合、nodeValue は null であり、読み取り専用です。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、エレメントノードとテキストノードを作成し、それぞれのノード値をチェックします。

```
// create an XML document
var doc:XML = new XML();

// create an XML node using createElement()
var myNode:XMLNode = doc.createElement("rootNode");
```

```
// place the new node into the XML tree
doc.appendChild(myNode);

// create an XML text node using createTextNode()
var myTextNode:XMLNode = doc.createTextNode("textNode");

// place the new node into the XML tree
myNode.appendChild(myTextNode);

trace(myNode.nodeValue);
trace(myTextNode.nodeValue);

// output:
// null
// myTextNode
```

次の例では、XML パケットを作成して解析します。このコードは各子ノードをループ処理し、firstChild プロパティと firstChild.nodeValue を使用してノード値を表示します。firstChild を使用してノードの内容を表示した場合は、& エンティティが保持されます。しかし、明示的に nodeValue を使用した場合は、アンパサンド文字 (&) に変換されます。

```
var my_xml:XML = new XML("mortongood&evil");
trace("using firstChild:");
for (var i = 0; i<my_xml.firstChild.childNodes.length; i++) {
  trace("\t"+my_xml.firstChild.childNodes[i].firstChild);
}
trace("");
trace("using firstChild.nodeValue:");
for (var i = 0; i<my_xml.firstChild.childNodes.length; i++) {
  trace("\t"+my_xml.firstChild.childNodes[i].firstChild.nodeValue);
}
```

次の情報が [出力] パネルに表示されます。

```
 using firstChild:
 morton
 good&amp;evil

 using firstChild.nodeValue:
 morton
 good&evil
```

関連項目

nodeType (XMLNode.nodeType プロパティ)

parentNode (XMLNode.parentNode プロパティ)

public parentNode : XMLNode (読み取り専用)

指定された XML オブジェクトの親ノードを参照する XMLNode 値です。ノードに親がない場合は null を返します。これは読み取り専用プロパティなので、子ノードの操作には使用できません。子ノードを操作するには、appendChild() メソッド、insertBefore() メソッド、および removeNode() メソッドを使用します。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、XML パケットを作成し、username ノードの親ノードを [出力] パネルに表示します。

```
var my_xml:XML = new XML("<login><username>morton</
  username><password>good&evil</password></login>");

// first child is the <login /> node
var rootNode:XMLNode = my_xml.firstChild;

// first child of the root is the <username /> node
var targetNode:XMLNode = rootNode.firstChild;
trace("the parent node of '"+targetNode.nodeName+"' is:
  "+targetNode.parentNode.nodeName);
trace("contents of the parent node are:\n"+targetNode.parentNode);
```

出力 (見やすくするために改行を追加):

```
the parent node of 'username' is: login
contents of the parent node are:
<login>
<username>morton</username>
<password>good&evil</password>
</login>
```

関連項目

appendChild (XMLNode.appendChild メソッド), insertBefore (XMLNode.insertBefore メソッド), removeNode (XMLNode.removeNode メソッド), XML

prefix (XMLNode.prefix プロパティ)

public prefix : String (読み取り専用)

XML ノード名の接頭辞部分。たとえば、ノード <contact:mailbox/>bob@example.com </contact:mailbox> には、ローカル名 "mailbox" と接頭辞 "contact" があり、この 2 つで完全なエレメント名 "contact.mailbox" が構成されます。

XML ノードオブジェクトの nodeName プロパティは、接頭辞とローカル名を含む完全な名前を返します。エレメント名のローカル名部分には、localName プロパティ経由でアクセスできます。

対応バージョン : ActionScript 1.0、Flash Player 8

例

SWF ファイルと XML ファイルは同じディレクトリに格納されています。"SoapSample.xml" という名前の XML ファイルには、次のコードが含まれています。

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
<soap:Body xmlns:w="http://www.example.com/weather">
<w:GetTemperature>
<w:City>San Francisco</w:City>
</w:GetTemperature>
</soap:Body>
</soap:Envelope>
```

SWF ファイルのソースには、次のスクリプトが含まれています (Output ストリングのコメントに注意)。

```
var xmlDoc:XML = new XML();
xmlDoc.ignoreWhite = true;
xmlDoc.load("SoapSample.xml");
xmlDoc.onLoad = function(success:Boolean)
{
  var tempNode:XMLNode = xmlDoc.childNodes[0].childNodes[0].childNodes[0];
  trace("w:GetTemperature prefix: " + tempNode.prefix); // Output: ... w
  var soapEnvNode:XMLNode = xmlDoc.childNodes[0];
  trace("soap:Envelope prefix: " + soapEnvNode.prefix); // Output: ... soap
}
```

previousSibling (XMLNode.previousSibling プロパティ)

public previousSibling : XMLNode (読み取り専用)

親ノードの子リスト内の前の子ノードを参照する XMLNode 値。ノードに前の兄弟ノードがない場合は、null を返します。このプロパティは子ノードの操作には使用できません。子ノードを操作するには、appendChild() メソッド、insertBefore() メソッド、および removeNode() メソッドを使用します。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次に示す例は、XML.lastChild プロパティの例から引用したもので、XML.previousSibling プロパティを使用して XML ノードの子ノードをループ処理する方法を示しています。

```
for (var aNode:XMLNode = rootNode.lastChild; aNode != null; aNode =
  aNode.previousSibling) {
  trace(aNode);
}
```

関連項目

lastChild (XMLNode.lastChild プロパティ), appendChild (XMLNode.appendChild メソッド), insertBefore (XMLNode.insertBefore メソッド), removeNode (XMLNode.removeNode メソッド), XML

removeNode (XMLNode.removeNode メソッド)

public removeNode() : Void

指定された XML オブジェクトをその親から削除します。また、ノードのすべての子孫も削除します。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、XML パケットを作成し、指定された XML オブジェクトとその子孫ノードを削除します。

```
var xml_str:String = "<state name=\"California\"><city>San Francisco</city></
  state>";

var my_xml:XML = new XML(xml_str);
var cityNode:XMLNode = my_xml.firstChild.firstChild;
trace("before XML.removeNode():\n"+my_xml);
cityNode.removeNode();
trace("");
trace("after XML.removeNode():\n"+my_xml);
```

```
// output (line breaks added for clarity):
//
// before XML.removeNode():
// <state name="California">
// <city>San Francisco</city>
// </state>
//
// after XML.removeNode():
// <state name="California" />
```

toString (XMLNode.toString メソッド)

```
public toString() : String
```

指定された XML オブジェクトを評価し、ノード、子、および属性を含む XML 構造体のテキスト表現を作成し、結果をストリングとして返します。

トップレベルの XML オブジェクト (コンストラクタで作成されたオブジェクト) の場合は、XML.toString() メソッドがドキュメントの XML 宣言 (保存場所は XML.xmlDecl プロパティ) を出力し、その後にドキュメントの DOCTYPE 宣言 (保存場所は XML.docTypeDecl プロパティ) を続け、さらにオブジェクト内のすべての XML ノードのテキスト表現を続けます。XML 宣言は、XML.xmlDecl プロパティが undefined の場合は出力されません。DOCTYPE 宣言は、XML.docTypeDecl プロパティが undefined の場合は出力されません。

対応バージョン : ActionScript 1.0、Flash Player 5

戻り値

String - ストリング。

例

次のコードでは、toString() メソッドを使用して XMLNode オブジェクトをストリングに変換し、次に String クラスの toUpperCase() メソッドを使用します。

```
var xString = "<first>Mary</first>"
     + "<last>Ng</last>"
var my_xml:XML = new XML(xString);
var my_node:XMLNode = my_xml.childNodes[1];
trace(my_node.toString().toUpperCase());
     // output: <LAST>NG</LAST>
```

関連項目

docTypeDecl (XML.docTypeDecl プロパティ), xmlDecl (XML.xmlDecl プロパティ)

XMLNode() コンストラクタ

```
public XMLNode(type:Number, value:String)
```

XMLNode コンストラクタを使用すると、内容を指定するストリングとノードタイプを表す数値に基づいて、XML ノードをインスタンス化できます。

対応バージョン： ActionScript 1.0、Flash Player 8

パラメータ

`type:Number` - ノードタイプを表す整数。

整数値	定義されている定数
1	ELEMENT_NODE
2	ATTRIBUTE_NODE
3	TEXT_NODE
4	CDATA_SECTION_NODE
5	NTITY_REFERENCE_NODE
6	ENTITY_NODE
7	PROCESSING_INSTRUCTION_NODE
8	COMMENT_NODE
9	DOCUMENT_NODE
10	DOCUMENT_TYPE_NODE
11	DOCUMENT_FRAGMENT_NODE
12	NOTATION_NODE

Flash Player の XML クラスがサポートするのはノードタイプ 1 (ELEMENT_NODE) と 3 (TEXT_NODE) だけです。

`value:String` - テキストノードの場合、ノードのテキストです。エレメントノードの場合は、タグの内容となります。

例

```
var ELEMENT_NODE:Number = 1;
var node1:XMLNode = new XMLNode(ELEMENT_NODE, "fullName");

var TEXT_NODE:Number = 3;
var node2:XMLNode = new XMLNode(TEXT_NODE, "Justin Case");
```

```
// Create a new XML document
var doc:XML = new XML();

// Create a root node
var rootNode:XMLNode = doc.createElement("root");

// Add the rootNode as the root of the XML document tree
doc.appendChild(rootNode);

// Build the rest of the document:
rootNode.appendChild(node1);
node1.appendChild(node2);

trace(doc);

// Output: Justin Case
```

XMLSocket

```
Object
  |
  +-XMLSocket

public class XMLSocket
extends Object
```

XMLSocket クラスはクライアントソケットを実装しており、Flash Player を実行するコンピュータはこのソケットを使用して、IP アドレスまたはドメイン名で識別されるサーバーコンピュータと通信することができます。XMLSocket クラスは、リアルタイムのチャットシステムなど待ち時間を短くすることが求められるクライアント / サーバーアプリケーションに適しています。従来の HTTP ベースチャットソリューションは頻繁にサーバーをポーリングし、HTTP 要求を使用して新しいメッセージをダウンロードします。それに対して、XMLSocket チャットソリューションはサーバーに対して開いた接続を維持するため、サーバーはクライアントから要求を受けずにただちに着信メッセージを送ることができます。XMLSocket クラスを使用するには、サーバーコンピュータは XMLSocket クラスで使用されるプロトコルに対応したデーモンを実行する必要があります。プロトコルの説明を次の一覧に示します。

- XML メッセージは、全二重 TCP/IP ストリームソケット接続を介して送られます。
- 各 XML メッセージは、ゼロ (0) バイトで終了する完全な XML ドキュメントです。
- 1 つの XMLSocket 接続を使用して送受信できる XML メッセージの数に制限はありません。

XMLSocket オブジェクトがサーバーに接続する方法と場所については、次の制限があります。

- XMLSocket.connect() メソッドが接続できる TCP ポートの番号は、1024 以上です。この制限により、XMLSocket オブジェクトと通信するサーバーデーモンにも、1024 以上のポート番号を割り当てる必要があります。1024 未満のポート番号は、FTP、Telnet、HTTP などのシステムサービスによって使用されることが多いため、XMLSocket オブジェクトはセキュリティ上の理由からこれらのポートにアクセスできません。ポート番号の制限により、これらのリソースが不適切にアクセスおよび乱用される可能性が少なくなります。
- XMLSocket.connect() メソッドは、SWF ファイルが存在するのと同じドメイン内のコンピュータにしか接続できません。この制限は、ローカルで再生される SWF ファイルには適用されません。この制限は、loadVariables()、XML.sendAndLoad()、および XML.load() のセキュリティ規則と同じです。特定のドメインからのアクセスを許可するセキュリティポリシーファイルを作成すると、SWF ファイルが存在するドメイン以外で実行されるサーバーデーモンに接続できます。

XMLSocket オブジェクトと通信するようにサーバーを設定すると、問題が発生する可能性があります。アプリケーションがリアルタイムのインタラクティブ機能を必要としない場合は、XMLSocket クラスの代わりに、loadVariables() 関数、または Flash の HTTP ベース XML サーバー接続 (XML.load()、XML.sendAndLoad()、XML.send()) を使用します。XMLSocket クラスのメソッドを使用するには、まず new XMLSocket コンストラクタを使用して XMLSocket オブジェクトを作成する必要があります。

対応バージョン： ActionScript 1.0、Flash Player 5

プロパティー覧

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

イベント一覧

イベント	説明
onClose = function() {}	開いた接続がサーバーによって閉じられたときにだけ呼び出されます。
onConnect = function(success: Boolean) {}	XMLSocket.connect() で開始した接続要求が成功したか失敗したときに、Flash Player によって呼び出されます。

イベント	説明
`onData = function(src: String) {}`	ゼロ (0) バイトで終了する XML メッセージがサーバーからダウンロードされると、呼び出されます。
`onXML = function(src: XML) {}`	XML ドキュメントを含む指定の XML オブジェクトが、開いている XMLSocket 接続を通して届いたときに、Flash Player によって呼び出されます。

コンストラクター一覧

署名	説明
`XMLSocket()`	新しい XMLSocket オブジェクトを作成します。

メソッド一覧

オプション	署名	説明
	`close() : Void`	XMLSocket オブジェクトで指定された接続を閉じます。
	`connect(url: String, port:Number) : Boolean`	指定の TCP ポート (1024 以上) を使用して、指定のインターネットホストへの接続を確立し、接続が正常に確立されたかどうかにより `true` または `false`, を返します。
	`send(data: Object) : Void`	`object` パラメータで指定された XML オブジェクトまたはデータをストリングに変換し、その後ろにゼロ (0) バイトを付加してサーバーに転送します。

Object クラスから継承されるメソッド

`addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)`

close (XMLSocket.close メソッド)

```
public close() : Void
```

XMLSocket オブジェクトで指定された接続を閉じます。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の簡単な例では、XMLSocket オブジェクトを作成し、サーバーへの接続を行ってから、接続を閉じます。

```
var socket:XMLSocket = new XMLSocket();
socket.connect(null, 2000);
socket.close();
```

関連項目

connect (XMLSocket.connect メソッド)

connect (XMLSocket.connect メソッド)

```
public connect(url: String, port: Number) : Boolean
```

指定の TCP ポート (1024 以上) を使用して、指定のインターネットホストへの接続を確立し、接続が正常に確立されたかどうかにより true または false, を返します。インターネットホストコンピュータのポート番号がわからない場合は、ネットワーク管理者に問い合わせてください。

host パラメータに対して null を指定した場合は、XMLSocket.connect() を呼び出す SWF ファイルが存在するホストに接続します。たとえば、www.yoursite.com から SWF ファイルをダウンロードした場合、host パラメータに null を指定することは www.yoursite.com の IP アドレスを入力することと同じです。

Flash Player 7 より前のバージョンの Player で SWF ファイルを実行している場合、host は、呼び出し元の SWF ファイルと同じスーパードメインに属している必要があります。たとえば、www.someDomain.com に存在する SWF ファイルは、store.someDomain.com に存在する SWF ファイルから変数をロードできます。どちらのファイルも同じスーパードメイン someDomain.com に属しているからです。

Flash Player 7 以降で SWF ファイルを実行している場合、host はまったく同じドメインに属している必要があります。たとえば、www.someDomain.com に置かれている SWF ファイルが、Flash Player 5 用にパブリッシュされていて、Flash Player 7 以降で実行される場合、www.someDomain.com に存在する SWF ファイルからしか変数をロードできません。異なるドメインから変数をロードする場合は、アクセスされる側の SWF ファイルをホスティングするサーバーにクロスドメインポリシーファイルを置いておく必要があります。

load() メソッドを実行すると、XML オブジェクトの loaded プロパティが false に設定されます。XML データのダウンロードが終了すると、loaded プロパティが true に設定され、onLoad イベントハンドラが呼び出されます。XML データは、完全にダウンロードされるまで解析されません。XML オブジェクトに XML ツリーが既に含まれていた場合、その XML ツリーは破棄されます。

XMLSocket.connect() が返す値が true である場合、接続プロセスの初期段階は成功です。この後で、最終的な接続が成功したか失敗したかを判断するために XMLSocket.onConnect メソッドが呼び出されます。XMLSocket.connect() メソッドが false を返した場合は、接続に失敗しています。

このメソッドを使用するときは、Flash Player セキュリティモデルを考慮してください。

- Flash Player 8 では、呼び出し元 SWF ファイルがローカルファイルシステムのサンドボックスに置かれている場合、このメソッドは使用できません。
- Flash Player 7 以降では、クロスドメインポリシーファイルの展開により、Web サイトでのリクエスタからリソースへのクロスドメインアクセスを許可することができます。

詳細については、次の参照先を参照してください。

- 『ActionScript 2.0 の学習』の「セキュリティについて」
- Flash Player 9 セキュリティに関するホワイトペーパー (http://www.adobe.com/go/fp9_0_security_jp)
- Flash Player 8 セキュリティ関連 API に関するホワイトペーパー (http://www.adobe.com/go/fp8_security_apis_jp)

対応バージョン : ActionScript 1.0、Flash Player 5 - Flash Player 7 ではビヘイビアが変更されました。

パラメータ

url:String - ストリング。FQDN (完全修飾ドメイン名)、つまり *aaa.bbb.ccc.ddd* という形式の IP アドレスです。SWF ファイルが存在するホストサーバーに接続するために、null を指定することもできます。呼び出し元の SWF ファイルが Web ブラウザで実行されている場合、host は SWF ファイルと同じドメインに属している必要があります。ドメインでの SWF に対する制限の詳細については、このメソッドの説明を参照してください。

port:Number - 数値。接続の確立に使用するホスト上の TCP ポート番号です。ポート番号は、1024 以上でなければなりません。

戻り値

Boolean - 接続が成功した場合は true、それ以外の場合は false。

例

次の例では、SWF ファイルが存在するホストに XMLSocket.connect( ) を使用して接続し、接続が成功したか失敗したかを示す戻り値を trace を使用して表示します。

```
var socket:XMLSocket = new XMLSocket()
socket.onConnect = function (success:Boolean) {
  if (success) {
  trace ("Connection succeeded!")
  } else {
  trace ("Connection failed!")
  }
}
if (!socket.connect(null, 2000)) {
  trace ("Connection failed!")
}
```

関連項目

onConnect (XMLSocket.onConnect ハンドラ),function ステートメント

onClose (XMLSocket.onClose ハンドラ)

```
onClose = function() {}
```

開いた接続がサーバーによって閉じられたときにだけ呼び出されます。このメソッドのデフォルトの実行形態では、アクションを実行しません。デフォルトの実行形態を無効にするには、カスタムアクションを含む関数を割り当てます。

対応バージョン : ActionScript 1.0、Flash Player 5

例

次の例では、開いた接続がサーバーによって閉じられた場合に、trace ステートメントを実行します。

```
var socket:XMLSocket = new XMLSocket();
socket.connect(null, 2000);
socket.onClose = function () {
  trace("Connection to server lost.");
}
```

関連項目

onConnect (XMLSocket.onConnect ハンドラ),function ステートメント

onConnect (XMLSocket.onConnect ハンドラ)

```
onConnect = function(success:Boolean) {}
```

XMLSocket.connect() で開始した接続要求が成功したか失敗したときに、Flash Player によって呼び出されます。接続に成功した場合、success パラメータは true です。それ以外の場合、success パラメータは false です。

このメソッドのデフォルトの実行形態では、アクションを実行しません。デフォルトの実行形態を無効にするには、カスタムアクションを含む関数を割り当てます。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

success:Boolean - ソケット接続が正常に確立されているかどうかを示すブール値 (true または false)。

例

次の例では、簡単なチャットアプリケーションで onConnect メソッドに対して置換関数を指定するプロセスを示します。

このスクリプトは、コンストラクタメソッドで XMLSocket オブジェクトを作成した後、onConnect イベントハンドラが呼び出されたときに実行されるカスタム関数を定義します。この関数は、接続が正常に確立されたかどうかに応じて、どの画面を表示するかを制御します。接続が正常に確立されると、startChat というフレーム上のメインのチャット画面が表示されます。接続が失敗すると、connectionFailed というフレーム上のトラブルシューティング情報を示す画面が表示されます。

```
var socket:XMLSocket = new XMLSocket();
socket.onConnect = function (success) {
  if (success) {
    gotoAndPlay("startChat");
  } else {
    gotoAndStop("connectionFailed");
  }
}
```

最後に、接続が開始されます。connect() が false を返すと、SWF ファイルは connectionFailed というフレームに直接送られ、onConnect は呼び出されません。connect() が true を返すと、SWF ファイルは "Please wait" 画面の waitForConnection というフレームにジャンプします。SWF ファイルは、onConnect ハンドラが呼び出されるまで waitForConnection フレーム上にあります。この呼び出しがいつ行われるかは、ネットワークの待ち時間によって決まります。

```
if (!socket.connect(null, 2000)) {
  gotoAndStop("connectionFailed");
} else {
  gotoAndStop("waitForConnection");
}
```

関連項目

connect (XMLSocket.connect メソッド), function ステートメント

onData (XMLSocket.onData ハンドラ)

```
onData = function(src:String) {}
```

ゼロ (0) バイトで終了する XML メッセージがサーバーからダウンロードされると、呼び出されます。XMLSocket.onData を無効にして、サーバーから送られたデータを XML として解析せずに取得できます。これは、転送するデータパケットのフォーマットが一定でない場合に、受け取ったデータを Flash Player で XML として解析するのでなく、直接操作するのに適しています。

デフォルトでは、XMLSocket.onData メソッドは XMLSocket.onXML メソッドを呼び出します。XMLSocket.onData を無効にしてカスタムビヘイビアを使用すると、XMLSocket.onXML は呼び出されなくなります。XMLSocket.onXML は XMLSocket.onData で呼び出す必要があります。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

src:String - サーバーから送られたデータを含むストリング。

例

この例では、src パラメータはサーバーからダウンロードされた XML テキストを含むストリングです。ゼロ (0) バイトのターミネータは、ストリングに含まれません。

```
XMLSocket.prototype.onData = function (src) {
  this.onXML(new XML(src));
}
```

onXML (XMLSocket.onXML ハンドラ)

```
onXML = function(src:XML) {}
```

XML ドキュメントを含む指定の XML オブジェクトが、開いている XMLSocket 接続を通して届いたときに、Flash Player によって呼び出されます。XMLSocket 接続を使用してクライアントとサーバーの間で転送できる XML ドキュメントの数に制限はありません。各ドキュメントは、値ゼロ (0) のバイトで終了します。Flash Player がゼロバイトを受信すると、直前のゼロバイト以降に受信されたか、またはこれが最初に受信されたメッセージである場合は接続が確立された後に受信された、すべての XML を解析します。解析された XML の各集合は 1 つの XML ドキュメントとして処理され、onXML メソッドに渡されます。

このメソッドのデフォルトの実行形態では、アクションを実行しません。デフォルトの実行形態を無効にするには、独自に定義したアクションを含む関数を割り当てます。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`src:XML` - サーバーから受信した解析済み XML ドキュメントを含む XML オブジェクト。

例

次の例では、簡単なチャットアプリケーションで onXML メソッドのデフォルトのインプリメンテーションを無効にします。関数 myOnXML は、次の形式で 1 つの XML エレメント MESSAGE を認識するようにチャットアプリケーションに指示します。

```
<MESSAGE user="John" text="Hello, my name is John!" />.
```

次の displayMessage() 関数は、ユーザーが受信したメッセージを表示するユーザー定義関数であるとします。

```
var socket:XMLSocket = new XMLSocket();
socket.onXML = function (doc) {
  var e = doc.firstChild;
  if (e != null && e.nodeName == "MESSAGE") {
    displayMessage(e.attributes.user, e.attributes.text);
  }
}
```

関連項目

function ステートメント

send (XMLSocket.send メソッド)

```
public send(data:Object) : Void
```

object パラメータで指定された XML オブジェクトまたはデータをストリングに変換し、その後ろにゼロ (0) バイトを付加してサーバーに転送します。object が XML オブジェクトである場合、ストリングは XML オブジェクトの XML テキスト表現です。送信操作は非同期です。つまり、転送処理はただちに終了しますが、データが転送されるのは、その後です。XMLSocket.send() メソッドは、データが正常に転送されたかどうかを示す値を返しません。

myXMLSocket オブジェクトが XMLSocket.connect() を使ってサーバーに接続されていない場合、XMLSocket.send() 操作は失敗します。

対応バージョン : ActionScript 1.0、Flash Player 5

パラメータ

`data:Object` - サーバーに転送する XML オブジェクトまたは他のデータ。

例

次の例では、XML オブジェクト my_xml をサーバーに送るためにユーザー名とパスワードを指定する方法を示します。

```
var myXMLSocket:XMLSocket = new XMLSocket();
var my_xml:XML = new XML();
var myLogin:XMLNode = my_xml.createElement("login");
myLogin.attributes.username = usernameTextField;
myLogin.attributes.password = passwordTextField;
my_xml.appendChild(myLogin);
myXMLSocket.send(my_xml);
```

関連項目

connect (XMLSocket.connect メソッド)

XMLSocket() コンストラクタ

```
public XMLSocket()
```

新しい XMLSocket オブジェクトを作成します。初期状態では、XMLSocket オブジェクトはサーバーに接続されません。オブジェクトをサーバーに接続するには、XMLSocket.connect() を呼び出す必要があります。

対応バージョン： ActionScript 1.0、Flash Player 5

例

次の例では、XMLSocket オブジェクトを作成します。

```
var socket:XMLSocket = new XMLSocket();
```

XMLUI

```
Object
  |
  +-XMLUI
```

```
public class XMLUI
extends Object
```

XMLUI オブジェクトにより、Flash オーサリングツールの拡張機能 (ビヘイビア、コマンド、エフェクト、ツールなど) のカスタムユーザーインターフェイスとして使用される SWF ファイルとやり取りすることができます。

Macromedia Flash MX 2004 以降のバージョンには、ビヘイビア、コマンド (JavaScript API)、エフェクト、ツールなどいくつかの拡張機能が備わっています。上級ユーザーがこうした機能を使用することで、オーサリングツールの機能を拡張または自動化できます。XML to UI エンジンをいずれかの拡張機能と組み合わせて使用することで、必須または任意のパラメータを拡張機能で受け付ける際に表示するダイアログボックスを作成できます。ダイアログボックスは XML タグを使用して定義するか、SWF ファイルを作成して表示します。XMLUI オブジェクトにより、上級ユーザーは、この方法で使用される SWF ファイルとやり取りすることができます。

対応バージョン： ActionScript 1.0、Flash Player 7

プロパティ一覧

Object クラスから継承されるプロパティ

constructor (Object.constructor プロパティ), __proto__ (Object.__proto__ プロパティ), prototype (Object.prototype プロパティ), __resolve (Object.__resolve プロパティ)

メソッド一覧

オプション	署名	説明
static	accept() : Void	現在の XMLUI ダイアログボックスを " 受け入れ " 状態にして終了します。
static	cancel() : Void	現在の XMLUI ダイアログボックスを " キャンセル " 状態にして終了します。
static	get(name:String) : String	現在の XMLUI ダイアログボックスの指定プロパティの値を取得します。
static	set(name:String, value:String) : Void	現在の XMLUI ダイアログボックスの指定プロパティの値を変更します。

Object クラスから継承されるメソッド

addProperty (Object.addProperty メソッド), hasOwnProperty (Object.hasOwnProperty メソッド), isPropertyEnumerable (Object.isPropertyEnumerable メソッド), isPrototypeOf (Object.isPrototypeOf メソッド), registerClass (Object.registerClass メソッド), toString (Object.toString メソッド), unwatch (Object.unwatch メソッド), valueOf (Object.valueOf メソッド), watch (Object.watch メソッド)

accept (XMLUI.accept メソッド)

```
public static accept() : Void
```

現在の XMLUI ダイアログボックスを " 受け入れ " 状態にして終了します。ユーザーが [OK] ボタンをクリックする操作と同じです。

対応バージョン : ActionScript 1.0、Flash Player 7

cancel (XMLUI.cancel メソッド)

```
public static cancel() : Void
```

現在の XMLUI ダイアログボックスを " キャンセル " 状態にして終了します。ユーザーが [キャンセル] ボタンをクリックする操作と同じです。

対応バージョン : ActionScript 1.0、Flash Player 7

get (XMLUI.get メソッド)

```
public static get(name:String) : String
```

現在の XMLUI ダイアログボックスの指定プロパティの値を取得します。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

`name:String` - 取得する XMLUI プロパティの名前。

戻り値

`String` - プロパティ値をストリングで返します。

set (XMLUI.set メソッド)

```
public static set(name:String, value:String) : Void
```

現在の XMLUI ダイアログボックスの指定プロパティの値を変更します。

対応バージョン : ActionScript 1.0、Flash Player 7

パラメータ

`name:String` - 変更する XMLUI プロパティの名前。

`value:String` - 指定されたプロパティの設定値。

第 3 章

使用されなくなった ActionScript 3

ActionScript の進化に伴い、使用されなくなった言語エレメントが数多く存在します。このセクションでは、使用を避ける項目を示し、代わりに使用できる言語エレメントを紹介します。使用されなくなったエレメントは Flash Player 8 でまだ機能しますが、これらのエレメントをコードで使用しないことをお勧めします。使用されなくなったエレメントのサポートは今後、保証されません。

使用されなくなったクラスの一覧

オプション	クラス名	説明
	Color	**非推奨** Flash Player 8 以降では使用しないでください。Color クラスの代わりに flash.geom.ColorTransform クラスを使用します。

使用されなくなった関数の一覧

オプション	関数名	説明
	call (frame: Object)	**非推奨** Flash Player 5 以降では使用しないでください。このアクションの代わりに `function` ステートメントを使用します。
	chr (number: Number) String	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに `String.fromCharCode()` を使用します。
	ifFrameLoaded ([scene: String], frame: Object)	**非推奨** Flash Player 5 以降では使用しないでください。この関数は使用されなくなりました。`MovieClip._framesloaded` プロパティを使用することをお勧めします。
	int (value:Number) Number	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに `Math.floor()`(正の値の場合) および `Math.ceil`(負の値の場合) を使用します。

オプション	関数名	説明
	length (expression:String, variable:Object) Number	**非推奨** Flash Player 5 以降では使用しないでください。この関数およびすべてのストリング関数は使用されなくなりました。String クラスのメソッドと `String.length` プロパティを使用して同じ処理を行うことをお勧めします。
	mbchr (number:Number)	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに `String.fromCharCode()` メソッドを使用します。
	mblength (string:String) Number	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに String クラスのメソッドとプロパティを使用します。
	mbord (character: String) Number	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに `String.charCodeAt()` を使用します。
	mbsubstring (value: String, index:Number, count:Number) String	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに `String.substr()` を使用します。
	ord (character: String) Number	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに String クラスのメソッドとプロパティを使用します。
	random (value:Number) Number	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに `Math.random()` を使用します。
	substring (string: String , index: Number , count: Number) String	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに `String.substr()` を使用します。
	tellTarget (target: String , statement(s))	**非推奨** Flash Player 5 以降では使用しないでください。ドット (.) 表記と `with` ステートメントを使用することをお勧めします。
	toggleHighQuality ()	**非推奨** Flash Player 5 以降では使用しないでください。この関数の代わりに `_quality` を使用します。

使用されなくなったプロパティの一覧

オプション	プロパティ名	説明
	Button. _highquality	**非推奨** Flash Player 7 以降では使用しないでください。このプロパティの代わりに `Button._quality` を使用します。
	MovieClip. _highquality	**非推奨** Flash Player 7 以降では使用しないでください。このプロパティの代わりに `MovieClip._quality` を使用します。
	TextField. _highquality	**非推奨** Flash Player 7 以降では使用しないでください。このプロパティの代わりに `TextField._quality` を使用します。
	_highquality	**非推奨** Flash Player 5 以降では使用しないでください。このプロパティの代わりに `_quality` を使用します。
	maxscroll	**非推奨** Flash Player 5 以降では使用しないでください。このプロパティの代わりに `TextField.maxscroll` を使用します。
	scroll	**非推奨** Flash Player 5 以降では使用しないでください。このプロパティの代わりに `TextField.scroll` を使用します。

使用されなくなった演算子の一覧

演算子	説明
<> (inequality)	**非推奨** Flash Player 5 以降では使用しないでください。この演算子は使用されなくなりました。`!=` (`inequality`) 演算子を使用することをお勧めします。
add (concatenation (strings))	**非推奨** Flash Player 5 以降では使用しないでください。Flash Player 5 以降用のコンテンツを作成する場合は、加算 (+) 演算子を使用することをお勧めします。この演算子は Flash Player 8 以降ではサポートされていません。
and (論理積 AND)	**非推奨** Flash Player 5 以降では使用しないでください。論理積 (AND) (`&&`) 演算子を使用することをお勧めします。
eq (equality (strings))	**非推奨** Flash Player 5 以降では使用しないでください。この演算子の代わりに `==` (`equality`) 演算子を使用します。
ge (大きいか等しい (ストリング))	**非推奨** Flash Player 5 以降では使用しないでください。この演算子の代わりに >= (より大きいか等しい) 演算子を使用します。

演算子	説明
gt (より大きい (ストリング))	**非推奨** Flash Player 5 以降では使用しないでください。この演算子の代わりに > (より大きい) 演算子を使用します。
le (小さいか等しい (ストリング))	**非推奨** Flash Player 5 以降では使用しないでください。この演算子の代わりに <= (より小さいか等しい) 演算子を使用します。
lt (より小さい (ストリング))	**非推奨** Flash Player 5 以降では使用しないでください。この演算子の代わりに < (より小さい) 演算子を使用します。
ne (不等価 (ストリング))	**非推奨** Flash Player 5 以降では使用しないでください。この演算子の代わりに `!=` (inequality) 演算子を使用します。
not (論理積 NOT)	**非推奨** Flash Player 5 以降では使用しないでください。この演算子の代わりに `!` (logical NOT) (等価) 演算子を使用します。
or (論理積 OR)	**非推奨** Flash Player 5 以降では使用しないでください。この演算子の代わりに `||` (logical OR) 演算子を使用します。

索引

記号

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

あ

え

お

か

き

ま