A Comparative Study of Web Language Support for Mobile Web Browsers

Katie A. Siek, Ashraf Khalil, Yong Liu, Nick Edmonds, Kay H. Connelly
Indiana University
Security for Ubiquitous Resources Group
{*ksiek, akhalil, yonliu, ngedmond, connelly* }@cs.indiana.edu

ABSTRACT

The surg of ubiquitous devices and WiFi in the last few years have given the general public access to information via the world wide web from anywhere at anytime. Mobile web access challenges developers to create web pages that are usable and familiar to the user without unduly burdening the information appliance's limited system resources. The best way to achieve these goals is to follow the readily available standards geared towards effective presentation of information in a wide variety of information appliances. This paper reports on a comprehensive comparison of six popular mobile web browsers and the language functionality they support.

1. INTRODUCTION

Advances in network connectivity have provided us with a wide range of ubiquitous devices designed to allow users to access content anytime from anywhere. These information appliances (PDAs, cellular phones, etc.) utilize the world wide web to provide their users with up-to-date information at their convenience. Because of the limited system resources and minimal interface capabilities of these systems, it is imperative that content be formatted such that it places little burden on the system and retains information and usability.

The best way to achieve these goals is to establish and follow standards geared towards effective presentation of information in a wide variety of information appliances. Fortunately standards already exist which describe how to format the most common types of content. The problem is that standards compliance varies among mobile web browser vendors.

Standards compliance is particularly important in mobile devices because (1) users of these devices want to access information quickly and with a minimal amount of interpretation and (2) users should be presented with a manageable amount of information given the interface constraints of their device.

In this paper, we tested and compared six popular web browsers and the language functionality they support. We evaluated web browsers on the Palm OS and Pocket PC platforms. AvantGo, Eudora-Web (Eudora), and WebToGo are freely available while Web Pro and Pocket Internet Explorer (Explorer) are provided with Palm OS and Pocket PC respectively. ThunderHawk has a free 30 day evaluation period. Specification compliance was tested by created web pages with HTML 4.0 strict as specified in the W3C's mobile access guidelines with emphasis on CSS support, forms, frames, image maps, tables, objects and images, and scripts.

This paper begins by reviewing the browsers and functionality we test. We summarize our results next. We end the paper with a discussion about our results and future research directions.

2. BROWSERS

In this section we discuss how we obtained, installed, and received web page data on the browsers.

2.1 WebPro

Our Tungsten T3 came with WebPro 3.0 preinstalled. WebPro is not typically installed on every kind of Palm PDA. For those who do not have WebPro on their PDA, they can easily download the browser from [4] for $35. Since WebPro came preinstalled on our device, we do not know how easy installation of the software is on other devices. WebPro can view web pages by uploading them from a memory card or the Internet via Bluetooth or WiFi access points depending on the device's capabilities. For our experiments, we viewed webpages online via our Bluetooth access point and offline using a secure memory card.

2.2 AvantGo

AvantGo is a freely available browser available from [15]. We installed the AvantGo 5.5 Build 95 browser on our Palm PDA and AvantGo conduit software on our desktop computer. The AvantGo website provides only Microsoft Window's conduit software, however conduit software is available for other operating system platforms from third parties. Users log on to the AvantGo website and identify channels (websites) they would like to download on their PDA. The website allows users to identify how many links should be downloaded, how often to update the webpage, and if images should be downloaded as well. The next time the PDA is hot synced, the channels are downloaded to the PDA for viewing. We logged into the AvantGo website, created a custom channel (our test suite website [13]), downloaded our test suite during a hot sync operation, and used online and offline testing. AvantGo allows users to view websites offline or online if the PDA is equipped with WiFi capabilities.

2.3 Eudora

The EudoraWeb 2.1 browser can be downloaded for free from [1]. Installing Eudora was an easy process. The browser did not require any special configurations before accessing the Internet. We viewed webpages online via our Bluetooth network. Eudora only offers online browsing.

2.4 Pocket Internet Explorer

Explorer 4.01 was preinstalled on our iPAQ 4155. It is the default web browser of all Pocket PC systems. There are no monthly or annual charges. Since Explorer was preinstalled on iPAQ series devices, we do know how easy installation is on other devices. Explorer can view HTML files by loading them from built-in memory or some external memory cards. Explorer can view online web pages via Bluetooth or WiFi connections. In our experiments, we used Explorer to browse our online test suite web pages on our WiFi network.

2.5 ThunderHawk

ThunderHawk version 1.1 is a third party mobile web browser for Pocket PC systems. It can be easily downloaded from [11]. The trial version of this software is fully functional, but expires in 30 days. The standard version costs $49.95 per year or $5.95 per month. The browser is easy to install - an automatic installation program takes care of all the file copying and registry updating tasks. ThunderHawk can only view web pages online. In our experiments, we used ThunderHawk to browse our online test suite web pages via our WiFi network.

2.6 WebToGo

WebToGo 4.1.1 can be obtained from [5] or many online shareware sites. WebToGo is free to try, while a full version costs $33. Installation was simple and straightforward on both the Palm and Pocket PC platforms. Configuration was likewise trivial once connectivity via WiFi was established. For our experiments we downloaded content to local storage.WebToGo is capable of browsing both local and remote content.

3. FUNCTIONALITY

Functionality was tested by exploring each browser's compliance with HTML 4.0 strict as specified in the W3C's mobile access guidelines with emphasis on CSS support, forms, frames, image maps, tables, objects and images, and scripts. Unless otherwise noted, we used the W3C's HTML 4.01 strict specification as our reference when coding our test suite [7]. Our test suite is available from [13].In this section, we describe how we coded our web pages and explain anything we decided not to test from the specification.

3.1 CSS Support

We used a modified version of the W3C's CSS Mobile Test Suite for our CSS support part of the test suite [9]. The CSS Mobile Test Suite assists web developers decide how much of the W3C's Mobile Profile a web browser supports. The test suite covers twelve areas: CSS2 syntax and basic data types, Selectors, Assigning Property Values, Cascading and Inheritance, Media Types, Box Model, Visual Formatting Model, Visual Formatting Model Details, Visual Effects, Lists, Colors and Backgrounds, Fonts, and Text.

We modified the test suite by combining some of these areas into one area and cutting down on the number of examples they tested. For example, if text appeared on the left side of an image, we assumed text would appear on the right side of an image too if we changed the alignment attribute. In addition to the test suite, we used a page from CSS1's test suite to see how much CSS1 and CSS2 each browser supported [2].

As we stated previously, we modified W3C's CSS Mobile Test suite. Thus, our coding consisted of copying the code base over to our server, taking out redundant examples, and modifying some of the CSS to ensure the examples we grouped together would not interfere with each other. We felt that in the time allotted, we could not create a better test suite than the W3C had provided us.

3.2 Forms

The form part of our test suite focused on various types of controls. The target control types we tested included `label, text field, text area, radio button, checkbox, menu,` submit button, file select, and image. Attributes related to each type of control were also tested. For example we tested attributes such as `border, name, src, width, height, vspace, hspace` for image control. We assign numbers and descriptions to each control on our form test to assist in finding the weaknesses of our browsers. If any malfunctions were detected on a browser, the number and description told us what part of the form failed.

3.3 Frames

We tested four frame elements in our test suite - frameset, frame, target, and iframe. In the frameset element part, we tested two attributes `rows` and `cols`. For the frame element, `src`, `noresize`, `scrolling`, and `frameborder` were tested. We tested `target` and `base` attributes for the target element. Only `width` and `height` attributes were tested for the inline frame element.For the inline frame, the width and height attributes are tested. The elements and attributes we selected assisted us in understanding how well specific mobile browsers supports frames.

3.4 Image Maps

For the image map portion of our test suite, we were only interested in the client-side image map and thus only covered client-side HTML specifications. We were not able to fully test the client-side image map specifications because most of the specification is not applicable to mobile browsers. Access keys, tab order, and tag information are not supported on mobile browsers that use a stylus as opposed to a mouse or traditional keyboard for navigation.

3.5 Tables

Our test suite tested for all elements in HTML table specifications. We did not test inherited attributes and non-visual user agents. Inherited attributes such as document-wide identifiers and inline style were not tested because these elements are tested elsewhere in our test suite.

3.6 Images

When coding the images and objects part of our test suite we used older and newer tags. HTML 4 introduced the `object` tag as a way for web developers to inserts current (images, applets, movies, etc.) and future media types into their web pages. We decided to test the older, non-deprecated element (`img`) as well as the `object` tag.

As discussed before we decided not to use deprecated elements (for instance `applet`) because the elements will not be used in the future and are not beneficial to future web developers if the browsers eventually do not support them. We opted to use the `object` tag for embedded web pages instead of the `iframe` tag because we tested `iframe` in our frames part of the test suite.

```
<OBJECT data="./code/pics/mark.png" type="image/png">
Object tag with png image
</OBJECT>

<OBJECT data="./code/pics/mark.gif" type="image/gif">
Object tag with gif image
</OBJECT>
```

Figure 1: Example of `object` tag with various image types.

The first part of our images test suit tested the use of the `object` tag with various image types (png, gif, jpeg, tiff) as shown in Figure 1. The second part of our image test suite tested the use of the `img` tag with different attribute parameters for the short description (`alt`, long description (`longdesc`), height and width. The third part of our image test suite tested how the `object` tag handled multimedia (video, flash, etc.). Finally, we finished the test suite by embedding a web page using an `object` tag.

3.7 Scripts

Initially our examination of scripting support was aimed at evaluating how completely and correctly each browser implemented JavaScript. Since we were evaluating standards compliance we elected to use the ECMAScript ECMA-262 3rd edition standard [8].

Despite the large amount of Java/ECMAScript in use on the web, very little of the content is strictly standards compliant. Because of the liberties taken by browser developers in implementing ECMAScript, many web developers have developed a library of tips, tricks, and kludges to allow their scripts to work on a variety of browsers. Because we were evaluating standards compliance as opposed to trying to write portable code, many excellent sites were of little use in testing our browsers.

In the absence of a good ECMAScript test suite, we elected to test compliance with the underlying Document Object Model (DOM) in the hopes that compliance at this level would provide a good indication of scripting support. Additionally, the W3C has a DOM test suite available which is accessible via the web and runs in a browser, as opposed to via an external Java framework [6] [3]. The W3C's test suite also conveniently provides a list of each failed test which would make it very easy to compare the browsers. Unfortunately while the DOM test suite ran well on several conventional browsers, none of the mobile browsers were able to successfully load the testing harness, which prevented them from being able to run the test suite. Reasons ranged from difficulties with frames to an inability to load some or all of the test harness code. These difficulties elucidate the need for a lightweight test suite that has fewer built-in assumptions about browser capabilities.

Having been frustrated by the inability of our browsers to run the full featured test suite available, we settled for coding some basic JavaScript tests and borrowing others from sources on the web [12] [16]. These tests include simple tests of core functionality and basic event handling.

4. RESULTS

In this section we will compare and contrast how each functionality was handled by mobile web browsers. For each functionality, we will discuss specifically what was tested and how we created ratings for each browser's performance.

4.1 CSS

Before testing for specific CSS functionality, we used the W3C's *CSS1 versus CSS2* test. We found that most of our browsers fully supported basic CSS1 functionality, however few supported CSS2 functionality as shown in Table 1.Once we had our results,we decided to test more CSS1 specific functionality than CSS2.

For the *syntax and basic type test*, we saw if browsers could interpret comments correctly, follow margin sizes, font sizes, and side bar coloring. Explorer was the only browser that could interpret all of the CSS code correctly. WebToGo rendered the comment, length, and font test correctly. WebPro and ThunderHawk were able to read the comments correctly, but nothing else. Thus, WebPro and ThunderHawk do not fully support syntax and basic types.

The color of words, sentences, and headings were tested in the *selectors and universal selectors tests*. We found that the browsers either supported selectors and universal selectors or did not, with WebToGo being the only exception. WebToGo mistakenly read a selector that started with a digit, thus failing one of our selector tests.

For the *importance test* we checked to see if the CSS rule deemed `important!` could override other conflicting rules. Only Explorer, ThunderHawk, and WebToGo interpreted `important!` rules correctly.

The *inheritance test* works similar to any other programming language. If a CSS rule says the web page has a yellow background and black text and another rule says that a specific selector, `P` has green text, then when `P` is used the green text will appear on a yellow background inherited from the first rule. Inheritance was fully supported by Explorer, ThunderHawk, and WebToGo. WebPro had difficulty using inheritance with `important!` rules and thus only partially supported inheritance.

The *cascading order test* ensured various bullet points could have different colors. Explorer, ThunderHawk and WebPro fully supported bullet points in various colors. WebToGo partially supported the cascading order test because the browser only used the first cascading order rule (`LI`), even when we used two unordered lists and had a rule for the embedded list (`LI LI`).

The *import test* insured the browsers could read an imported CSS file and use the rules. Similar to our other tests, Explorer, ThunderHawk, WebPro, and WebToGo fully supported the import test.

The *media dependent test* ensured the browsers could interpret CSS rules specific to handheld computers. Only WebPro, ThunderHawk, and Explorer successfully interpreted the handheld specific CSS rules.

We created colorful boxes around words and sentences to *test border support*. Explorer was the only browser that rendered the colorful boxes around the words and sentences. WebPro and AvantGo created light grey boxes around the specific words and sentences.

In the *margin test*, we looked for specific colors in the margin and margin sizes. WebPro, ThunderHawk, and Explorer successfully completed the test. AvantGo was only able to show the correct colors, but no the margin sizes.

	WebPro	AvantGo	Eudora	Explorer	ThunderHawk	WebToGo
CSS1 Support	◒	●	○	●	●	○
CSS2 Support	○	○	○	◒	◒	◒
Syntax and Basic Types	○	○	○	●	○	◒
Selectors	●	○	○	●	●	◒
Universal Selectors	●	○	○	●	●	○
Importance	○	○	○	●	●	●
Inheritance	◒	○	○	●	●	●
Cascading Order Test	●	○	○	●	●	◒
Import	●	○	○	●	●	●
Media Dependence	●	○	○	●	●	○
Border Test	○	○	○	●	○	○
Margin Test	●	◒	○	●	●	○
Padding Test	○	○	○	●	●	●
Background Test	◒	○	○	●	◒	○
Display Test	○	○	○	●	◒	◒
List Test	○	○	○	◒	◒	◒
Font Test	○	○	○	◒	◒	○

Table 1: CSS Results

For the *padding test*, we created text blocks with specific sized spacing around the text.Only Explorer, ThunderHawk, and WebToGo supports CSS padding. Even though AvantGo and WebPro did not support padding, the text was easier to read when compared with the hard coded padding supported in the Explorer text blocks.

The *background test* used color backgrounds, image backgrounds, and attached image backgrounds. Attached image backgrounds ensures an image stays where it is independent of scrolling. AvantGo did not support color or image backgrounds. WebPro and ThunderHawk supported color backgrounds, but not image backgrounds. Explorer supported all backgrounds except attached image backgrounds.

We tested *display functionality* by allowing text to overflow out of text boxes, float beside images, float between images, and under images. In addition, we tested if text could be indented properly via rules and made invisible in our display test. Explorer was the only browser to support all of our display tests with the exception of text overflow. ThunderHawk supported text overflow, and text floating, but not indentation. The browser did not let the text overflow from the text box. AvantGo could not handle text overflow or indentation rules, however the browser was able to float text correctly with images - but only images boxes were present - no images. WebToGo could only rendered indentation rules and floating text with images correctly. WebPro did not support any of our display tests.

Lists with various bullet points were tested - letters, Roman numerals, images, dots, squares, etc. Explorer supported dots, circles, Roman numerals, letters, numbers, no bullets - everything except images. ThunderHawk supported Roman numerals, letters, numbers, no bullets, images, and all kinds of bullet shapes except circles and dots. WebToGo supported Roman numerals, letters, and numbers, but only rendered squares when characters were not used as bullets. AvantGo and WebPro only supported dots or counting numbers as lists - no Roman numerals, letters, or images for lists.

Different font sizes, width, style, and families were tested in our *font test*. Explorer could control font size, width, and style, but could only support a small subset of font families (sans-serif and monospace). ThunderHawk supported small cap fonts and sans-serif fonts. WebPro and WebToGo could only support font weight. AvantGo used the same font throughout the test with no variation.

We can see from Table 1, Explorer supports CSS the best. Current web developers should stick with using CSS1 commands and not expect anything but sans-serif fonts and basic listing elements.

4.2 Forms

When we compared the performance of our browsers on our form test, we found AvantGo, Explorer, ThunderHawk, and WebToGo had the same performance as shown in Table 2. All four browsers passed tests on `label, text field, text area, radio button, checkbox, menu,` and `image controls`.. The only element the browsers had difficulty rendering was the file select control. In other words, the browser did not allow users to brow the local files system to upload files. WebPro and Eudora had slightly worse performance than the other browsers because they failed the image control - the browsers could not render the images properly.

4.3 Frames

Explorer and ThunderHawk outperformed the rest of the browsers, shown in Table 3, in the frame test because they supported the `frame` tag and all the associated attributes. None of the browsers were able to render the `frameborder` attribute because they did not support frames. ThunderHawk was the only browser that supported inline frame, but it could not render the `noresize` attribute properly.

Compared to Explorer and ThunderHawk, the other browsers did not do nearly as well. WebPro only partially supported frame attributes because the browser simply concatenated all the pages in different frames into one large page. WebToGo and Eudora didn't support frames either. In our test, the browsers only showed links in the position where the web contents was supposed to be. However, by clicking these links, users could still visit the corresponding

	WebPro	**AvantGo**	**Eudora**	**Explorer**	**ThunderHawk**	**WebToGo**
`label` control	●	●	●	●	●	●
`text field` control	●	●	●	●	●	●
`text area` control	●	●	●	●	●	●
`radio button` control	●	●	●	●	●	●
`checkbox` control	●	●	●	●	●	●
`menu` control	●	●	●	●	●	●
Submit button control	●	●	●	●	●	●
File select control	○	○	○	○	○	○
`Image` control	○	●	○	●	●	●

Table 2: Forms Results

	WebPro	**AvantGo**	**Eudora**	**Explorer**	**ThunderHawk**	**WebToGo**
`rows` attribute	◒	○	○	●	●	◒
`cols` attribute	◒	○	○	●	●	◒
`src` attribute	◒	○	◒	●	●	◒
`noresize` attribute	○	○	○	●	○	○
`scrolling` attribute	◒	○	○	●	●	○
`frameborder`	○	○	○	○	○	○
`target` attribute	◒	○	◒	●	●	◒
`base` attribute	◒	○	◒	●	●	◒
Inline frame	○	○	○	○	●	○

Table 3: Frames Results

frame web pages. AvantGo failed to open any HTML pages with `frame` tags.

4.4 Image Maps

Our image map test determined if the browsers could render the image and if users could navigate with the image map. As Table 4 shows, only Explorer fully supported image maps. ThunderHawk partially supports image maps because the browser allowed the map to be navigated despite not showing an image. All other browsers failed to support the image map.

4.5 Tables

From Table 5, we can see that all of the browsers we tested with the exception of ThunderHawk supported the border attribute which controls the width of the frame around the table. The rules attribute, which specifies which rules will appear between cells within a table, was not supported by any of the browsers. The frame attribute, that specifies which side of the frame surrounding the table will be visible, is supported by all browsers except WebPro. Column groups, which include `colgroup` and `col` elements, were not supported by any of the browsers. Column groups allow structural division within a table and thus columns with the same structure can be combined in one group. Raw groups are only supported by Explorer and ThunderHawk browsers. Raw groups which include `thead`, `tfoot`, and `tbody` elements allow rows to be combined into table head, body or table foot sections. Eudora does not support any of the table attributes and it only supports the basic table elements (`tr, td, and th`). Mobile browsers that only support basic table elements are difficult to read because a browser that supports only the basic table elements shows the text within the table without any formatting. This may lead the text of different cells to merge together and appear in the wrong place and thus lead to poor readability.

It is worth mentioning that, even though some browsers do not support many table elements and attributes, the degree of clarity and readability they offer vary significantly from one to another. We found that Explorer and ThunderHawk were the easiest to read followed by WebPro, AvantGo, WebToGo, and Eudora.

4.6 Images

ThunderHawk and AvantGo were clearly the best browsers for images as shown in Table 6. They were the only two browsers who could handle`object` tags correctly. AvantGo differed from ThunderHawk in that it could not display tif images. ThunderHawk and Explorer were the only browsers that printed the `alt` tag when it could not render the `object` tag. WebToGo and Eudora did not display images or tags.

ThunderHawk and Explorer were the only two browsers who could completely render `img` tags and width or height changes. AvantGo, WebPro, and WebToGo rendered the `img` tags, but did not render the height or width changes correctly. Eudora could not render the images, but did print the alt tags.

None of the mobile browsers tested supported other media such as various video formats, however they did fully support embedded HTML documents.ThunderHawk and Explorer were the only browsers that printed the `alt` tag when it could not render the multimedia video. Currently, if web developers want to create web pages for mobile devices, we recommend they use `img` tags and embedded HTML documents to ensure the widest variety of browsers are supported.

	WebPro	AvantGo	Eudora	Explorer	ThunderHawk	WebToGo
ImageMap support	○	○	○	●	◒	○

Table 4: Image Map Results

	WebPro	AvantGo	Eudora	Explorer	ThunderHawk	WebToGo
Border attribute	●	●	○	●	○	○
Rules attribute	○	○	○	●	○	●
`cellpadding` attribute	●	●	○	●	●	◒
`cellspacing` attribute	●	●	○	●	●	●
Width attribute	●	●	○	●	●	○
`frame` attribute	●	●	○	●	●	○
`valign` attribute	●	●	○	●	●	○
`colspan, rowspan` attribute	●	●	○	●	●	●
Caption element	●	●	●	●	●	●
`col` and `colgroup` elements	○	○	○	○	○	○
`tfoot, thead,` and `tbody` elements	○	○	○	●	●	○
`tr, td,` and `th` element	●	●	●	●	●	●

Table 5: Table Results

4.7 Scripts

Initially, we attempted to test compliance with the W3C's Document Object Model using the supplied test suite [3]. Unfortunately, none of the browsers were able to successfully load the testing harness, which prevented them from being able to run the tests. Thus, as shown in Table 7, all browsers failed this test.

Our next goal was to see if the browsers supported basic core functionality such as alerts and simple methods on basic data types (i.e. type conversion, numerical calculations, logical tests, etc.). Some of these tests utilized simple scripts we wrote, while others were borrowed from [16]. As shown in Table 7, WebPro, AvantGo, Explorer, WebToGo, and ThunderHawk were all able to successfully complete these tests with only minimal errors if any. It should be noted that many desktop web browsers failed some individual elements of these tests. Eudora was unable to complete the tests as it does not support JavaScript.

Finally, some basic event handling was tested using [12]. Several event types were tested: `onClick, onBlur, onChange, onSubmit`, and `onMouseOver`. Once again referencing Table 7, it can be seen that WebPro, AvantGo, Explorer, and ThunderHawk were all able to successfully complete these tests. WebToGo was unable to complete these tests as it only supports server side JavaScript. Once again, Eudora was unable to complete the tests due to a lack of JavaScript support.

From these tests it is evident that JavaScript support varies widely from browser to browser, and that no single browser has a complete implementation. These tests also reveal the need for a DOM test suite for mobile web browsers that makes fewer assumptions about the capabilities available, or at least a lightweight testing harness around the current test suite.

5. DISCUSSION

Some mobile web browsers have guidelines for web developers to follow in order to create optimal web pages for their browsers. In this section we will compare what the browser specific documentation says the browser supports with our results.

5.1 WebPro

According to [17], WebPro supports CSS, Frames, Forms, Tables, JavaScript 1.5, and various image formats for HTML 4.01. Our test results showed that WebPro supports most table and form features, but it poorly supports CSS and Frames. We also found that image maps are not supported.

According to Palm, the newest release of WebPro version 3.0.1 has Frame support. We used WebPro 3.0 for our tests because the browser came with our Tungsten T3.

5.2 AvantGo

The AvantGo channel developer guide claims to support a major portion of HTML 4, CSS 1.0 style attributes, DOM level 1, a small bit of DOM level 2, and some JavaScript [10]. The developer guide has detailed tables of all the elements and attributes it supports.

AvantGo did pass our overview CSS1 test with flying colors, however it did not pass our tests specifically dealing with borders, padding, display, lists, background, or fonts as the guide says it supports. However, the guide says AvantGo can handle table `rules` element, `tfoot`, and `thead` even though we found it cannot. The most surprising difference we found was the guide's claim of DOM level 1 support, even though the browser failed our DOM tests. The guide does not mention image maps. AvantGo performed just as the guide said it would for forms, frames, and JavaScript.

	WebPro	AvantGo	Eudora	Explorer	ThunderHawk	WebToGo
`object` tag support	○	●	○	○	●	○
`img` tag support	◒	◒	◒	●	●	◒
Other media support	○	○	○	○	○	○
Embedded HTML support	●	●	●	●	●	●

Table 6: Images Results

	WebPro	AvantGo	Eudora	Explorer	ThunderHawk	WebToGo
W3C DOM Test Suite	○	○	○	○	○	○
ECMAScript Methods (Static Content)	●	●	○	●	●	●
ECMAScript Events	●	●	○	●	●	○

Table 7: Scripting Results

5.3 Eudora

Eudora browser specifications state that the browser supports standard HTML mark-up and forms, but not frames. CSS, tables, and image map support are not discussed in [19]. We found that Eudora supports most form features and minimally supports tables, but it does not support CSS, frames, and image maps.

5.4 Explorer

We could not find a detailed specification of the version of Pocket Internet Explorer used in our test. According to some technical support articles [14] posted on Microsoft's website, the newest version of Explorer is supposed to support CSS, JavaScript, and basic features of HTML 4.0. Our test result show Explorer had very good support for CSS1 and part of CSS2 specification. The browser did pretty well in the form, table, frame and image map tests, with exception of the `iframe` test. It missed some `object` tags in image support test, but could correctly rendered all `img` tags.

5.5 ThunderHawk

The supported features documented in ThunderHawk's specification include CSS1, most important features of CSS2 (like positioning), various image formats (GIF, JPEG, PNG, PJPEG), JavaScript 1.5, and HTML 4.0 [11]. Our test result showed ThunderHawk performed as the specification promised. The browser passed the CSS1 test, part of the CSS 2 test, image test (even on the image formats which was not mentioned in its specification, such as TIF), and most of the HTML 4.01 specified features. Surprisingly, ThunderHawk even supports `inline` frames. It was the only browser in our test which passed the `inline` frame test.

5.6 WebToGo

The WebToGo 4.1.1 specification claims to support HTML 4.1, CSS, two image formats (GIF and JPEG), and JavaScript [18]. While it does support some portion of all of these features, the support is by no means complete. WebToGo failed the CSS1 test completely and only partially supported CSS2. Like the other browsers, it did have excellent form support, but left much to be desired in the frame tests. Difficulties continued with image maps, which it failed completely, and tables, where those that it did display were unclear and difficult to read. Finally it only supports server-side JavaScript, though in all fairness, the developer states this in the specifications. WebToGo claims to support many important web standards, but fails to note which version (as in the case of CSS), or does not provide complete support (as in the case of CSS and HTML 4.1). WebToGo is inappropriate for browsing interactive content, and is a non-optimal choice for browsing static content.

6. CONCLUSION

The W3C acknowledged mobile devices offer a different user experience than traditional desktop browsers in 1998 and has since developed guidelines and standards for developers. However, the guidelines, standards, and groups creating the guidelines are always changing names and reorganizing forcing developers to look at individual browser specifications or program for the largest common denominator. As our paper has shown, browser specification claims do not always mean the browser necessarily supports the functionality.

Until we have a way to validate mobile browser code from a unified mobile browser standard, we suggest mobile web developers use CSS1 attributes - basic listing elements and sans-serif fonts. Programmers should use <`img`> tags until <`object`> tags become more widely used and supported. Avoid image maps and frames for readability and easy navigation. Forms, tables, and ECMA/JavaScripting are okay to use on most mobile browsers.

7. ACKNOWLEDGMENTS

Katie A. Siek is supported in part by a National Physical Science Consortium Fellowship and by a stipend from Sandia National Laboratories/CA. Kay H. Connelly is partially supported by a grant from the Lilly Endowment.

8. REFERENCES

[1] Eudora. http://www.eudora.com/products/.

[2] W3C's CSS1 Test Suite: CSS1 vs. CSS2. http://www.w3.org/Style/CSS/Test/CSS1/current/sec03.htm.

[3] W3C's DOM test suite. http://www.w3.org/DOM/Test/.

[4] Web browser pro. http://www.palmone.com/us/software/webbrowserpro/.

[5] WebToGo. http://www.webtogo.de/uk/.

[6] World Wide Web Consortium. http://www.w3c.org.

[7] HTML 4.01 Specification. http://www.w3.org/TR/html4/, December 1999. D. Raggett, A. Le Hors, and I. Jacobs, eds.

[8] Standard ECMA-262, ECMAScript Language Specification 3rd. ed. http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf, December 1999.

[9] W3C's CSS Mobile Test suite. http://www.w3.org/Style/CSS/Test/Mobile/1.0/current/, March 2002.

[10] AvantGo. AvantGo Channel Development Guide: Version 2.0. Technical report, iAnywhere, 2003. http://www.ianywhere.com/avantgo/developer/channel_developer.

[11] Bitstream. Thunderhawk. http://www.bitstream.com/wireless/getthnow.html.

[12] CA Explorit Science Center, Davis. Explorit's JavaScript Test Page. http://www.dcn.davis.ca.us/GO/EXPLORIT/java/JavaScript.html.

[13] Security for Ubiquitous Resources Group. SURG Test Suite. http://www.cs.indiana.edu/cgi-pub/surg/www@10/, 2004.

[14] HPC. Overview of Pocket Internet Explorer. http://www.hpcfactor.com/support/cesd/s/0001.asp. Product introduction.

[15] Inc. iAnywhere Solutions. Avantgo. http://www.avantgo.com, November 2003.

[16] Brian J. McCloud. MauveCloud's Core JavaScript Test Suite. http://www.mauvecloud.net/jscore/.

[17] palmOne. *White Paper: The palmOne Web Pro Browser: Unleashing the Potential of Wireless Information Access*, 2003.

[18] palmsource. Webtogo mobile internet, 2004. http://palmsource.palmgear.com.

[19] Qualcomm. *Eudora Internet Suite 2.1: EudoraWeb Browser*, 2004. http://www.eudora.com/products/unsupported/internetsuite/eudoraweb.html.