Learning Web Design

Fourth Edition

A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics

Jennifer Niederst Robbins

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

Learning Web Design, Fourth Edition

by Jennifer Niederst Robbins

Printed in Canada.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly Media books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*safari.oreilly.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editor: Simon St. Laurent

Production Editor: Melanie Yarbrough

Copy Editor: Genevieve d'Entremont

Technical Reviewer: Aaron Gustafson, Matt Menzer, Joel Marsh

Interior Designer: Ron Bilodeau

Cover Designer: Mark Paglietti

Indexer: Ellen Troutman Zaig

Print History:

February 2001:	First edition.
March 2004:	Second edition.
June 2007:	Third edition.
August 2012:	Fourth edition.

ISBN: 978-1-449-31927-4
[TI]

CONTENTS

Preface xi

Part I Getting Started

Chapter 1
Where Do I Start? 3
- Where Do I Start? 4
- What Does a Web Designer Do? 4
- What Languages Do I Need to Learn? 11
- What Do I Need to Buy? 14
- What You've Learned 19
- Test Yourself 20

Chapter 2
How the Web Works 21
- The Internet Versus the Web 21
- Serving Up Your Information 21
- A Word About Browsers 23
- Web Page Addresses (URLs) 24
- The Anatomy of a Web Page 26
- Putting It All Together 30
- Test Yourself 32

Chapter 3
Some Big Concepts You Need to Know 33
- A Dizzying Multitude of Devices 34
- Sticking with the Standards 36
- Progressive Enhancement 36
- Responsive Web Design 38

One Web for All (Accessibility) 41
The Need for Speed (Site Performance) 43
Test Yourself 45

Part II HTML Markup for Structure

Chapter 4
Creating a Simple Page 49
A Web Page, Step by Step 49
Before We Begin, Launch a Text Editor 50
Step 1: Start with Content 53
Step 2: Give the Document Structure 55
Step 3: Identify Text Elements 58
Step 4: Add an Image 61
Step 5: Change the Look with a Style Sheet 64
When Good Pages Go Bad 65
Validating Your Documents 66
Test Yourself 67
Element Review: Document Structure 68

Chapter 5
Marking Up Text 69
Paragraphs 70
Headings 70
Lists 73
More Content Elements 76
Organizing Page Content 79
The Inline Element Roundup 84
Generic Elements (div and span) 95
Some Special Characters 99
Putting It All Together 100
Test Yourself 102
Element Review: Text 104

Chapter 6
Adding Links 105
The href Attribute 106
Linking to Pages on the Web 107
Linking Within Your Own Site 108
Targeting a New Browser Window 118

Mail Links ... 119
Telephone Links ... 120
Test Yourself ... 121
Element Review: Links ... 122

Chapter 7
Adding Images ... 123
First, a Word on Image Formats ... 123
The img Element ... 124
A Window in a Window ... 130
Test Yourself ... 131
Element Review: Images ... 132

Chapter 8
Table Markup ... 133
How Tables Are Used ... 133
Minimal Table Structure ... 135
Spanning Cells ... 139
Table Accessibility ... 142
Wrapping Up Tables ... 144
Test Yourself ... 146
Element Review: Tables ... 146

Chapter 9
Forms ... 147
How Forms Work ... 147
The form Element ... 149
Variables and Content ... 151
The Great Form Control Roundup ... 152
Form Accessibility Features ... 171
Form Layout and Design ... 173
Test Yourself ... 175
Element Review: Forms ... 176

Chapter 10
What's Up, HTML5? ... 181
A Funny Thing Happened on the Way to XHTML 2 ... 182
In the Markup Department ... 185
Meet the APIs ... 189
Video and Audio ... 192

Canvas 198
Final Word 202
Test Yourself 203

Part III CSS for Presentation

Chapter 11
Cascading Style Sheets Orientation 207
The Benefits of CSS 207
How Style Sheets Work 209
The Big Concepts 214
Moving Forward with CSS 221
Test Yourself 223

Chapter 12
Formatting Text 225
The Font Properties 225
Changing Text Color 243
A Few More Selector Types 244
Text Line Adjustments 249
Underlines and Other "Decorations" 252
Changing Capitalization 252
Spaced Out 253
Text Shadow 254
Changing List Bullets and Numbers 259
Test Yourself 261
CSS Review: Font and Text Properties 263

Chapter 13
Colors and Backgrounds 265
Specifying Color Values 265
Foreground Color 272
Background Color 273
Playing with Opacity 275
Introducing…Pseudo-class Selectors 276
Pseudo-element Selectors 279
Attribute Selectors 281
Background Images 284
The Shorthand background Property 293
Like a Rainbow (Gradients) 296
Finally, External Style Sheets 300

Test Yourself ... 303
CSS Review: Color and Background Properties ... 304

Chapter 14
Thinking Inside the Box ... 305
The Element Box ... 305
Specifying Box Dimensions ... 306
Padding ... 312
Borders ... 316
Margins ... 328
Assigning Display Roles ... 333
Adding Drop Shadows to Boxes ... 335
Test Yourself ... 336
CSS Review: Basic Box Properties ... 338

Chapter 15
Floating and Positioning ... 341
Normal Flow ... 341
Floating ... 342
Positioning Basics ... 356
Relative Positioning ... 358
Absolute Positioning ... 359
Fixed Positioning ... 368
Test Yourself ... 370
CSS Review: Floating and Positioning Properties ... 371

Chapter 16
Page Layout with CSS ... 373
Page Layout Strategies ... 373
Page Layout Techniques ... 380
Multicolumn Layouts Using Floats ... 381
Positioned Layout ... 392
Top-to-Bottom Column Backgrounds ... 395
Test Yourself ... 398

Chapter 17
Transitions, Transforms, and Animation ... 399
Ease-y Does It (CSS Transitions) ... 399
CSS Transforms ... 410
Keyframe Animation ... 420

Test Yourself 423
CSS Review: Transitions, Transforms, and Animation 426

Chapter 18
CSS Techniques 427
A Clean Slate (CSS Reset) 427
Image Replacement Techniques 429
CSS Sprites 430
Styling Forms 434
Styling Tables 441
Basic Responsive Web Design 444
Wrapping Up Style Sheets 454
Test Yourself 454
CSS Review: Table Properties 456

Part IV JavaScript for Behaviors

Chapter 19
Introduction to JavaScript 459
What Is JavaScript? 459
Adding JavaScript to a Page 463
The Anatomy of a Script 463
The Browser Object 478
Events 478
Putting It All Together 481
Test Yourself 483

Chapter 20
Using JavaScript 485
Meet the DOM 485
Polyfills 493
JavaScript Libraries 497
Big Finish 501
Test Yourself 502

Part V Creating Web Graphics

Chapter 21
Web Graphics Basics 507
Image Sources 507
Meet the Formats 510
Image Size and Resolution 522
Working with Transparency 526
Introduction to SVG 533
Summing Up Images 538
Test Yourself 539

Chapter 22
Lean and Mean Web Graphics 541
General Image Optimization Strategies 542
Optimizing GIFs 543
Optimizing JPEGs 547
Optimizing PNGs 552
Optimize to File Size 553
Optimization in Review 554
Test Yourself 555

Appendix A
Answers 557

Appendix B
CSS3 Selectors 583

Index 587

PREFACE

Hello and welcome to the fourth edition of *Learning Web Design*.

So much has happened since the previous edition! Just when it looked like things were beginning to settle down with the adoption of web standards by the browser creators and the development community, along comes the "Mobile Web" to shake things up again. With the introduction of smartphones and tablets, the Web is finding its way onto small screens and on-the-go contexts where it never appeared before. This has introduced some rigorous challenges for web designers and programmers as we scramble to find ways to make the experience of using our sites pleasing, regardless of how they might be accessed.

As I write, many of these challenges, such as how to deliver the right image to the right device, are still being debated. It's an incredibly lively time for web design, full of experimentation and collaboration. In ways, it reminds me of the Wild West days of the Web back in 1993 when I started my web design career. So much to figure out! So many possibilities! And to be honest, it's also a tricky time to nail these moving-target technologies and techniques down in a book. To that end, I've done my best to point out the topics that are in flux and provide pointers to online resources to bring you up to date.

There are also two new standards—HTML5 (the fifth major revision of Hypertext Markup Language) and CSS3 (Cascading Style Sheets, Level 3)—available to us now that were only rumors the last time I wrote this book. The HTML section of the book now reflects the current HTML5 standard. I cover the parts of the developing CSS3 standard that are ready for prime time, including a new chapter on adding motion and interactivity with Transitions and Transforms. Our tools allow us to do so much more and in a more efficient way than even a few years ago.

Finally, because JavaScript has become such a significant part of web development, this new edition includes two chapters introducing JavaScript syntax and a few of its uses. I'm no JavaScript expert, but I was very lucky to find someone who is. The JavaScript chapters were written by Mat "Wilto"

THE COMPANION WEBSITE

Be sure to visit the companion website for this book at learningwebdesign.com. It features materials for the exercises, downloadable articles, lists of links from the book, book references, and other good stuff.

Marquis, who is a designer and developer at Filament Group, a member of the jQuery Mobile team, and the Technical Editor at *A List Apart*.

As in the first three editions, this book addresses the specific needs and concerns of beginners of all backgrounds, including seasoned graphic designers, programmers looking for a more creative outlet, office assistants, recent college graduates, work-at-home moms, and anyone else wanting to learn how to design websites. I've done my best to put the experience of sitting in my beginner web design class into a book, with exercises and tests along the way, so you get hands-on experience and can check your progress.

Whether you are reading this book on your own or using it as a companion to a web design course, I hope it gives you a good head start and that you have fun in the process.

How This Book Is Organized

Learning Web Design, Fourth Edition is divided into five parts, each dealing with an important aspect of web development.

Part I: Getting Started

Part I lays a foundation for everything that follows in the book. I start off with some important general information about the web design environment, including the various roles you might play, the technologies you might learn, and tools that are available to you. You'll get your feet wet right away with HTML and CSS and learn how the Web and web pages generally work. I'll also introduce you to some Big Concepts that get you thinking the way modern web designers think about their craft.

Part II: HTML for Structure

The chapters in Part II cover the nitty-gritty of every element and attribute available to give content semantic structure, including the new elements introduced in HTML5. We'll cover the markup for text, links, images, tables, and forms. Part II closes out with an in-depth discussion of HTML5 and how it differs from previous standards.

Part III: CSS for Presentation

In the course of Part III, you'll go from learning the basics of using Cascading Style Sheets for changing the presentation of text to creating multicolumn layouts and even adding time-based animation and interactivity to the page. It also addresses common CSS techniques, including how to create a page using Responsive Web Design.

Part IV: JavaScript for Behaviors

Mat Marquis starts Part IV out with a rundown of JavaScript syntax so you can tell a variable from a function. You'll also get to know some ways that JavaScript is used, including DOM Scripting, and existing

Typographical Conventions Used In This Book

The following typographical conventions are used in this book:

Italic

Used to indicate URLs, email addresses, filenames, and directory names, as well as for emphasis.

Colored roman text

Used for special terms that are being defined and for cross-references.

`Constant width`

Used to indicate code examples and keyboard commands.

`Colored constant width`

Used for emphasis in code examples.

`Constant width italic`

Used to indicate placeholders for attribute and style sheet property values.

JavaScript tools such as polyfills and libraries that let you put JavaScript to use quickly, even if you aren't quite ready to write your own code from scratch.

Part V: Creating Web Graphics

Part V introduces the various file formats that are appropriate for the Web and describes how to optimize them to make their file size as small as possible.

Acknowledgments

I want to thank my editor, Simon St. Laurent, with whom I've had a good run of collaborative projects and I look forward to more. Thanks also go to my contributor, Mat Marquis (*matmarquis.com*), for making JavaScript entertaining and for maintaining good spirits while collaborating with a control freak.

Many smart and lovely people had my back on this edition. I want to thank my primary technical reviewers, Aaron Gustafson (easy-designs.net), Joel Marsh (*thehipperelement.com*), and Matt Menzer, for taking so much time out of their schedules to make sure the details in the chapters were spot on. Thanks also go to the following folks for their "surgical strike" reviews: Anthony Calzadilla, Danny Chapman, Matt Haughey, Gerald Lewis, Jason Pamental, and Stephanie Rieger.

I feel fortunate to know so many of the leaders in this field whose books, articles, presentations, slide decks, and personal contact were the fuel that kept me going. I couldn't have done it without the help of these geniuses (in alphabetical order): Dan Cederholm, Josh Clark, Andy Clarke, Chris Coyier, Brad Frost, Lyza Gardner, Jason Grigsby, Stephen Hay, Scott Jehl, Scott Jenson, Tim Kadlec, Jeremy Keith, Sanders Kleinfeld, Peter-Paul Koch, Bruce Lawson, Ethan Marcotte, Eric Meyer, Karen McGrane, Shelley Powers, Bryan Rieger, Stephanie Rieger, Remy Sharp, Luke Wroblewski, and Jeffrey Zeldman.

It takes a village to make a book, and I'd like to extend my appreciation to the contributions of Melanie Yarbrough (production editor and proofreader), Genevieve d'Entremont (copy editor), Rebecca Demarest (figure production), Newgen (page layout), Ellen Troutmen Zeig (index), Randy Comer (book cover design), and Ron Bilodeau (book interior design).

Finally, I'd like to thank Edie Freedman (best boss ever) for her patience while this book sucked me into a vortex. And to my dearest darlings, Jeff and Arlo, I'm happy to finally say, "I'm back."

About the Author

Jennifer Robbins began designing for the Web in 1993 as the graphic designer for Global Network Navigator, the first commercial website. In addition to this book, she is the author of *Web Design in a Nutshell* and *HTML5 Pocket Reference* (which is also available as an iOS app), both published by O'Reilly. In the past, Jennifer has spoken at many conferences, including Seybold and South By Southwest, and has taught beginning web design at Johnson and Wales University in Providence, RI. She is currently a digital product designer for O'Reilly Media, where she is interested in information architecture, interaction design, and making websites, apps, and ebooks pleasant to use. When not on the clock, Jennifer enjoys making things, indie rock, cooking, and being a Mom.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: *Learning Web Design, Fourth Edition* by Jennifer Robbins. Copyright 2012 Littlechair , Inc., 978-1-449-31927-4.

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

We'd Like to Hear from You

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

(800) 998-9938 (in the United States or Canada)

(707) 829-0515 (international or local)

(707) 829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://oreil.ly/learn_web_design_4e

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our web site at:

http://www.oreilly.com

Colophon

Our look is the result of reader comments, our own experimentation, and feedback from distribution channels. Distinctive covers complement our distinctive approach to technical topics, breathing personality and life into potentially dry subjects. The text font is Linotype Birka; the heading font is Adobe Myriad Pro.

GETTING STARTED

PART I

IN THIS PART

Chapter 1
Where Do I Start?

Chapter 2
How the Web Works

Chapter 3
Some Big Concepts You Need to Know

CHAPTER 1

WHERE DO I START?

The Web has been around for more than 20 years now, experiencing euphoric early expansion, an economic-driven bust, an innovation-driven rebirth, and constant evolution along the way. One thing is certain: the Web as a communication and commercial medium is here to stay. Not only that, it has found its way onto devices such as smartphones, tablets, TVs, and more. There have never been more opportunities to put web design know-how to use.

Through my experience teaching web design courses and workshops, I've had the opportunity to meet people of all backgrounds who are interested in learning how to build web pages. Allow me to introduce you to just a few:

> *"I've been a print designer for 17 years, and now I am feeling pressure to provide web design services."*
>
> *"I work as a secretary in a small office. My boss has asked me to put together a small internal website to share company information among employees."*
>
> *"I've been a programmer for years, but I want to try my hand at design. I feel like the Web is a good opportunity to explore new skills."*
>
> *"I am an artist and I want to know how to get samples of my paintings and sculpture online."*
>
> *"I tinkered with web pages in high school and I think it might be something I'd like to do for a living."*

Whatever the motivation, the first question is always the same: "Where do I start?" It may seem like there is a mountain of stuff to learn, and it's not easy to know where to jump in. But you have to start somewhere.

This chapter attempts to put the learning curve in perspective by answering the most common questions I get asked by people ready to make the leap. It provides an introduction to the disciplines, technologies, and tools associated with web design.

IN THIS CHAPTER

Where do I start?

What does a web designer do?

What languages do I need to learn?

What software and equipment do I need to buy?

Where Do I Start?

Your particular starting point will no doubt depend on your background and goals. However, a good first step for everyone is to get a basic understanding of how the Web and web pages work. This book will give you that foundation. Once you learn the fundamentals, there are plenty of resources on the Web and in bookstores for you to further your learning in specific areas.

There are many levels of involvement in web design, from building a small site for yourself to making it a full-blown career. You may enjoy being a full-service website developer or just specializing in one skill. There are a lot of ways you can go.

If your involvement in web design is purely at the hobbyist level, or if you have just one or two web projects you'd like to publish, you may find that a combination of personal research (like reading this book), taking advantage of available templates, and perhaps even investing in a visual web design tool such as Adobe Dreamweaver may be all you need to accomplish the task at hand. Many Continuing Education programs offer introductory courses to web design and production.

If you are interested in pursuing web design or production as a career, you'll need to bring your skills up to a professional level. Employers may not require a web design degree, but they will expect to see working sample sites that demonstrate your skills and experience. These sites can be the result of class assignments, personal projects, or a simple site for a small business or organization. What's important is that they look professional and have well-written, clean HTML, style sheets, and possibly scripts behind the scenes. Getting an entry-level job and working as part of a team is a great way to learn how larger sites are constructed and can help you decide which aspects of web design you would like to pursue.

I Just Want a Blog!

You don't necessarily need to become a web designer to start publishing your words and pictures on the Web. You can start your own "blog" or personal journal site using one of the free or inexpensive blog hosting services. These services provide templates that generally spare you the need to learn HTML (although it still doesn't hurt). These are some of the most popular as of this writing:

- WordPress (*www.wordpress.com*)
- Blogger (*www.blogger.com*)
- Tumblr (*www.tumblr.com*)

Another drag-n-drop site design and hosting service that goes beyond the blog is Squarespace (*www.squarespace.com*).

What Does a Web Designer Do?

Over the years, the term "web design" has become a catchall for a process that encompasses a number of different disciplines, from user experience design, to document markup, to serious programming. This section describes some of the most common roles.

If you are designing a small website on your own, you will need to wear many hats. The good news is that you probably won't notice. Consider that the day-to-day upkeep of your household requires you to be part-time chef, housecleaner, accountant, diplomat, gardener, and construction worker—but to you it's just the stuff you do around the house. In the same way, as a solo web designer, you may be a part-time graphic designer, writer, HTML author, and information architect, but to you, it'll just feel like "making web pages." Nothing to worry about.

AT A GLANCE

The term "web design" has come to encompass a number of disciplines, including:

- Visual (graphic) design
- User interface and experience design
- Web document and style sheet production
- Scripting and programming
- Content strategy
- Multimedia

There are also specialists out there whom you can hire to fill in the skills you don't have. For example, I have been creating websites since 1993 and I still hire programmers and multimedia developers when my clients require interactive features. That allows me to focus on the parts I do well (in my case, it's the content organization, interface, and visual design).

If you are not interested in becoming a jack-of-all-trades solo web designer, you may choose to specialize and work as part of a team or as a freelance contractor.

Large-scale websites are almost always created by a team of people, numbering from a handful to hundreds. In this scenario, each member of the team focuses on one facet of the site-building process. If that is the case, you may be able to simply adapt your current set of skills (writing, Photoshop, programming, etc.) and interests to the new medium.

I've divided the myriad roles and responsibilities typically covered under the umbrella term "web design" into four very broad categories: design, development, content strategy, and multimedia.

Design

Ah, design! It sounds fairly straightforward, but even this simple requirement has been divided into a number of specializations when it comes to creating sites. Here are a few of the new job descriptions related to designing a site, but bear in mind that the disciplines often overlap and that the person calling herself the "Designer" often is responsible for more than one (if not all) of these responsibilities.

User Experience, Interaction, and User Interface design

Often, when we think of design, we think about how something looks. On the Web, the first matter of business is designing how the site *works*. Before picking colors and fonts, it is important to identify the site's goals, how it will be used, and how visitors move through it. These tasks fall under the disciplines of Interaction Design (IxD), User Interface (UI) design, and User Experience (UX) design. There is a lot of overlap between these responsibilities, and it is not uncommon for one person or team to handle all three.

The goal of the Interaction Designer is to make the site as easy, efficient, and delightful to use as possible. Closely related to interaction design is User Interface design, which tends to be more narrowly focused on the functional organization of the page as well as the specific tools (buttons, links, menus, and so on) that users use to navigate content or accomplish tasks.

A more recent job title in the web design realm is the User Experience Designer. The UX designer takes a more holistic view—ensuring the entire experience with the site is favorable. UX design is based on a solid understanding of users and their needs based on observations and interviews. According to Donald Norman (who coined the term), user experience design includes "all aspects of the user's interaction with the product: how it is perceived, learned, and used." For a website or application, that includes

the visual design, the user interface, the quality and message of the content, and even overall site performance. The experience must be in line with the organization's brand and business goals in order to be successful.

Some of the documents an IxD, UI, or UX designer might produce include:

User research and testing reports

Understanding the needs, desires, and limitations of users is central to the success of the design of the site or web application. This approach of designing around the user's needs is referred to as User Centered Design (UCD), and it is central to contemporary design. Site designs often start with user research, including interviews and observations, in order to gain a better understanding of how the site can solve problems or how it will be used. It is typical for designers to do a round of user testing at each phase of the design process to ensure the usability of their designs. If users are having a hard time figuring out where to find content or how to move to the next step in a process, then it's back to the drawing board.

Wireframe diagrams

A wireframe diagram shows the structure of a web page using only outlines for each content type and widget (Figure 1-1). The purpose of a wireframe diagram is to indicate how the screen real estate is divided and indicate where functionality and content such as navigation, search boxes, form elements, and so on, are placed, without any decoration or graphic design. They are usually annotated with instructions for how things should work so the development team knows what to build.

Site diagram

A site diagram indicates the structure of the site as a whole and how individual pages relate to one another. Figure 1-2 shows a very simple site diagram. Some site diagrams fill entire walls!

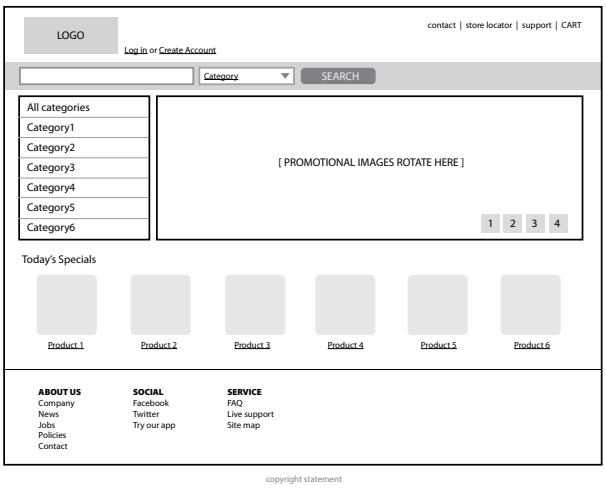

Figure 1-1. Wireframe diagram.

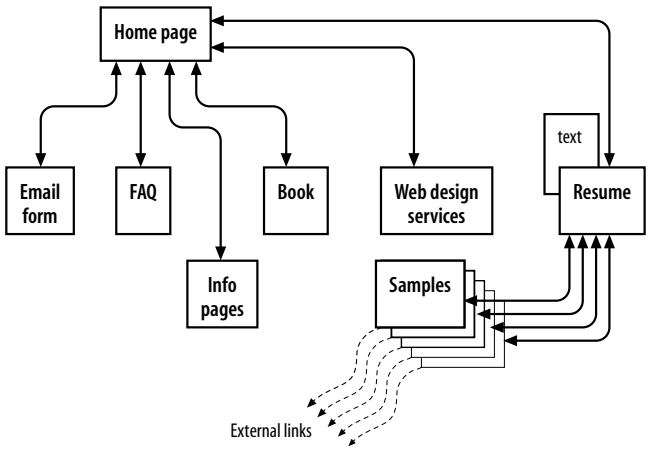

Figure 1-2. A simple site diagram.

Storyboards and user flow charts

A storyboard traces the path through a site or application from the point of view of a typical user (a persona in UX lingo). It usually includes a script and "scenes" consisting of screen views or the user interacting with the screen. The storyboard aims to demonstrate the steps it takes to accomplish tasks, possible options, and also introduces some standard page types. Figure 1-3 shows a simple storyboard. A user flow chart is another method for showing how the parts of a site or application are connected that tends to focus on technical details rather than telling a story. For example, when the user does this, it triggers that function on the server. It is common for designers to create a user flow chart for the steps in a process such as member registration or online payments.

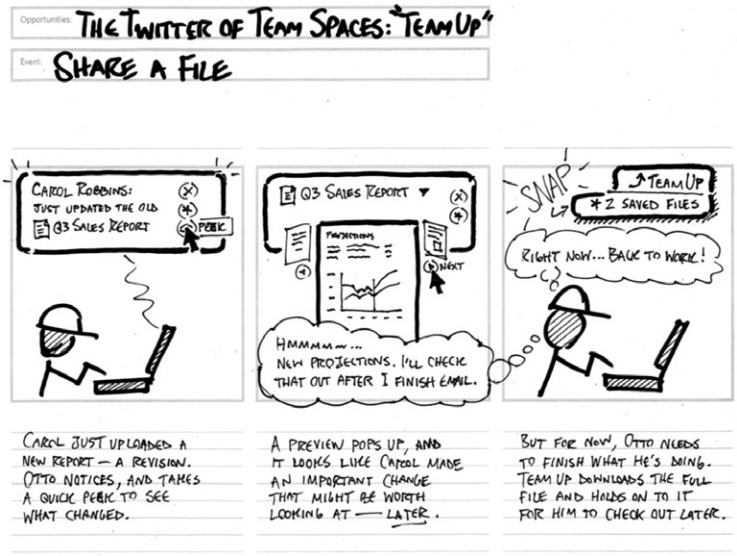

Figure 1-3. A typical storyboard [courtesy of Adaptive Path; drawn by Brandon Schauer].

Visual (graphic) design

Because the Web is a visual medium, web pages require attention to presentation and design. A graphic designer creates the "look and feel" of the site—logos, graphics, type, colors, layout, etc.—to ensure that the site makes a good first impression and is consistent with the brand and message of the organization it represents. Visual designers typically generate sketches of the way the site might look, as shown in Figure 1-4. They may also be responsible for producing the graphic files in a way that is optimized for delivery over the Web (see Chapter 21, Lean and Mean Web Graphics for image optimization techniques).

If you are interested in doing the visual design of commercial sites professionally, I strongly recommend graphic design training as well as a strong proficiency in Adobe Photoshop (the industry standard) or Adobe Fireworks.

Figure 1-4. Look and feel sketches for a simple site.

If you are already a graphic designer, you will be able to adapt your skills to the Web easily, although this will not excuse you from acquiring a solid understanding of HTML, CSS, and other web technologies. Because most sites have at least a few images, even hobbyist web designers will need to know how to create and edit images, at minimum.

Again, I want to note that all of these responsibilities may fall into the hands of one designer who creates both the look and the functionality of a site. But for larger sites with bigger budgets, there is an opportunity to find your own special niche in the design process.

Development

A fair amount of the web design process involves the creation and troubleshooting of the documents, style sheets, scripts, and images that make up a site. At web design firms, the team that handles the creation of the files that make up the website (or templates for pages that get assembled dynamically) is usually called the development or production department.

Web developers may not design the look or structure of the site themselves, but they do need to communicate well with designers and understand the intended site goals so they may suggest solutions that meet those goals.

The broad disciplines that fall under development are authoring, styling, and scripting/programming.

Authoring/markup

Authoring is the term used for the process of preparing content for delivery on the Web, or more specifically, marking up the content with HTML tags that describe its content and function. If you want a job as a web developer, you need to have an *intricate* knowledge of HTML and how it functions on various browsers and devices. The HTML specification is constantly evolving, which means you'll need to keep up with the latest best practices and opportunities as well as bugs and limitations. The good news is, it's not difficult to get started, and from there, you can gradually increase your skills. We'll be dabbling with HTML in Chapter 2, How the Web Works and then discussing it in great detail in Part II of this book.

Styling

In web design, the appearance of the page in the browser is controlled by style rules written in CSS (Cascading Style Sheets). We'll get deep into CSS in Part III of this book (including what "cascading" means!), but for now just know that in contemporary web design, the appearance of the page is handled separately from the HTML markup of the page. Again, if you are interested in working in web development, knowing your way around CSS and how it is supported (or not supported) by browsers is guaranteed to be part of your job description.

Style Tiles

Another approach to capturing the look and feel of a site is to create style tiles, which give examples of color schemes, branding elements, content and UI treatments, and mood boards without applying them to a specific page layout. The idea is to agree upon a consistent visual language for the site. For more on this technique, read the article "Style Tiles and How They Work," by Samantha Warren (*www.alistapart.com/articles/style-tiles-and-how-they-work/*), and visit her excellent site where you can download a template at *styletil.es*.

NOTE

Many visual designers translate their designs into HTML and CSS documents themselves. In fact, there is a popular argument that in order to call yourself a "web designer," you must be able to build your designs yourself, and nearly everyone agrees that your job prospects will be better if you are able to code as well as design.

Scripting and programming

As the Web has evolved into a platform of applications for getting stuff done, programming has never been more important. JavaScript is the language that makes elements on web pages do things. It adds behaviors and functionality to elements in the page and even to the browser window itself.

There are other web-related programming languages as well, including PHP, Ruby, Python, and ASP.NET, that run on the server and process data and information before it is sent to the user's browser. See the sidebar "Frontend Versus Backend" for more information on what happens where.

Web scripting and programming definitely requires some traditional computer programming prowess. While many web programmers have degrees in computer science, it is also common for developers to be self-taught. A few developers I know started by copying and adapting existing scripts, then gradually added to their programming skills with each new project. Still, if you have no experience with programming languages, the initial learning curve may be a bit steep.

Teaching web programming is beyond the scope of this book. JavaScript is introduced in Chapter 19, Introduction to JavaScript (teaching JavaScript could fill a whole book itself). It is possible to turn out content-rich, well-designed sites without the need for programming, so hobbyist web designers should not be discouraged. However, once you get into collecting information via forms or serving information on demand, it is usually necessary to have a programmer on the team. You may also ask your hosting company if they offer the functionality you are looking for in an easy-to-use, canned service.

Frontend Versus Backend

You may hear web designers and developers say that they specialize in either the frontend or backend of website creation.

Frontend design

"Frontend" refers to any aspect of the design process that appears in or relates directly to the browser. This book focuses primarily on frontend web design.

The following tasks are commonly considered to be frontend tasks:

- Graphic design and image production
- Interface design
- Information design as it pertains to the user's experience of the site
- HTML document and style sheet development
- JavaScript

Backend development

"Backend" refers to the programs and scripts that work on the server behind the scenes to make web pages dynamic and interactive. In general, backend web development falls in the hands of experienced programmers, but it is good for all web designers to be familiar with backend functionality.

The following tasks take place on the backend:

- Information design as it pertains to how the information is organized on the server
- Forms processing
- Database programming
- Content management systems
- Other server-side web applications using PHP, JSP, Ruby, ASP.NET, Java, and other programming languages

Content strategy and creation

Third on our list, though ideally first in the actual website creation process, is the critical matter of the site's content itself. Anyone who uses the title "web designer" needs to be aware that everything we do supports the process of getting the content, message, or functionality to our users. Furthermore, good writing can help the user interfaces we create be more effective.

Of course, someone needs to create the content and maintain it—don't underestimate the resources required to do this successfully. In addition, I want to call your attention to two content-related specialists on the modern web development team: the Content Strategist and Information Architect (IA).

When the content isn't written right, the site can't be fully effective. A Content Strategist makes sure that every bit of text on a site, from long explanatory text down to the labels on buttons, supports the brand identity and marketing goals of the company. Content strategy may also extend to data modeling and content management on a large and ongoing scale, such as planning for content reuse and update schedules.

An Information Architect (also called an Information Designer) organizes the content logically and for ease of findability. She may be responsible for search functionality, site diagrams, and how the content and data is organized on the server. Information architecture is inevitably entwined with UX and UI design, and it is not uncommon for a single person or team to perform all roles.

Multimedia

One of the cool things about the Web is that you can add multimedia elements to a site, including sound, video, animation, and even interactive games. You may decide to add multimedia skills, such as audio and video editing or Flash development (see the "A Little More About Flash" sidebar), to your web design tool belt, or you may decide to go all in and become a multimedia specialist. If you are not interested in becoming a multimedia developer, you can always hire one. Web development companies usually look for people who have mastered the standard multimedia tools, and have a good visual sensibility and an instinct for intuitive and creative multimedia design.

A Little More About Flash

Adobe Flash (previously Macromedia Flash, previously FutureSplash) is a multimedia format created especially for the Web. Flash is used for create full-screen animation, interactive graphics, integrated audio and video clips, and even scriptable games and applications, all at remarkably small file sizes. However, recently Flash use has been on the decline due to a number of developments, including:

- Apple's decision not to support Flash on its iPhones and iPads in favor of non-proprietary HTML5 methods.
- Adobe's decision to stop supporting Flash (its own product) for mobile browsers.
- The new programmable **canvas** element in HTML5 that offers some of the same functionality as Flash.
- Criticism that Flash sometimes gets in the way of user goals. For example, who wants to sit through a movie and soundtrack on a restaurant site when all you really want to know is whether they are open on Sunday?
- The fact that a plug-in is required to play Flash media makes some developers squeamish.

In fact, it is not uncommon to hear web professionals cite that "Flash is dead," but despite suddenly becoming the underdog, Flash still has some advantages if used the right way:

- Because it uses vector graphics, Flash files are small and can be resized without loss of detail.
- It is a streaming format, so movies start playing quickly and continue to play as they download.
- You can use ActionScript to add behaviors and advanced interactivity, allowing Flash to be used as the frontend for dynamically generated content or ecommerce functions.
- The Flash plug-in is well-distributed on PCs, so support on desktop browsers is reliable.
- Although HTML5 is promising and rapidly evolving, as of this writing, it cannot match the features and performance of Flash.

Flash is not likely to disappear overnight, but even Adobe is putting its muscle behind HTML5 alternatives.

What Languages Do I Need to Learn?

If you are a visual designer who spends time in Photoshop and Illustrator, you may be put off by needing to learn how to create your designs with text, but I assure you, it's pretty simple to get started. There are also authoring tools that speed up the production process, as we'll discuss later in this chapter.

The following is a list of technologies associated with web development. Which languages and technologies you learn will depend on the role you see yourself in within the web design process. However, I advise *everyone* involved in building websites to know their way around HTML and Cascading Style Sheets, and if you want to do frontend web development for a living, JavaScript know-how is pretty much a job requirement. More technically inclined web professionals may take on server configurations, databases, and site performance, but these are generally not frontend developer tasks (although a basic familiarity with the backend issues never hurts).

AT A GLANCE

Web-related technologies:

- Hypertext Markup Language (HTML)
- Cascading Style Sheets (CSS)
- JavaScript and DOM scripting
- Server-side programming and database management

The World Wide Web Consortium

The World Wide Web Consortium (called the W3C for short) is the organization that oversees the development of web technologies. The group was founded in 1994 by Tim Berners-Lee, the inventor of the Web, at the Massachusetts Institute of Technology (MIT).

In the beginning, the W3C concerned itself mainly with the HTTP protocol and the development of the HTML. Now, the W3C is laying a foundation for the future of the Web by developing dozens of technologies and protocols that must work together in a solid infrastructure.

For the definitive answer on any web technology question, the W3C site is the place to go:

www.w3.org

For more information on the W3C and what they do, see this useful page:

www.w3.org/Consortium/

Hypertext Markup Language (HTML)

You may see HTML and XHTML referred to collectively as (X)HTML.

HTML (HyperText Markup Language) is the language used to create web page documents. There are a few versions of HTML in use today: HTML 4.01 is the most firmly established and the newer, more robust HTML5 is gaining steam and browser support. Both versions have a stricter implementation called XHTML (eXtensible HTML), which is essentially the same language with much stricter syntax rules. We'll get to the particulars of what makes the various versions different in Chapter 10, What's Up, HTML5?.

HTML is not a programming language; it is a markup language, which means it is a system for identifying and describing the various components of a document such as headings, paragraphs, and lists. The markup indicates the document's underlying structure (you can think of it as a detailed, machine-readable outline). You don't need programming skills—only patience and common sense—to write HTML.

The best way to learn HTML is to write out some pages by hand, as we will be doing in the exercises in this book. If you end up working in web production, you'll live and breathe HTML. But even hobbyists will benefit from knowing what is going on under the hood. The good news is that it's simple to learn the basics.

Cascading Style Sheets (CSS)

While HTML is used to describe the content in a web page, it is Cascading Style Sheets (CSS) that describe how that content should *look*. In the web design biz, the way the page looks is known as its presentation. That means fonts, colors, background images, line spacing, page layout, and so on... all controlled with CSS. With the newest version (CSS3), you can even add special effects and basic animation to your page.

CSS also provides methods for controlling how documents will be presented in contexts other than the traditional desktop browser, such as in print and or on devices with small screen widths. It also has rules for specifying the non-visual presentation of documents, such as how they will sound when read by a screen reader (although those are not well supported).

Style sheets are also a great tool for automating production because you can change the way an element looks across all the pages in your site by editing a single style sheet document. Style sheets are supported to some degree by all modern browsers.

NOTE

When this book says "style sheets" it is always referring to Cascading Style Sheets, the standard style sheet language for the World Wide Web.

Although it is possible to publish web pages using HTML alone, you'll probably want to take on style sheets so you're not stuck with the browser's default styles. If you're looking into designing websites professionally, proficiency at style sheets is mandatory.

Style sheets are discussed further in Part III.

JavaScript/DOM scripting

JavaScript is a scripting language that is used to add interactivity and behaviors to web pages, including these (just to name a few):

- Checking form entries for valid entries
- Swapping out styles for an element or an entire site
- Making the browser remember information about the user for the next time she visits
- Building interface widgets, such as expanding menus

JavaScript is used to manipulate the elements on the web page, the styles applied to them, or even the browser itself. There are other web scripting languages, but JavaScript (also called ECMAScript) is the standard and most ubiquitous.

You may also hear the term DOM scripting used in relation to JavaScript. DOM stands for Document Object Model, and it refers to the standardized list of web page elements that can be accessed and manipulated using JavaScript (or another scripting language). DOM scripting is an updated term for what used to be referred to as DHTML (Dynamic HTML), now considered an obsolete approach.

Writing JavaScript is a type of programming, so it may be time-consuming to learn if you have no prior programming experience. Many people teach themselves JavaScript by reading books and following and modifying existing examples. Most web-authoring tools come with standard scripts that you can use right out of the box for common functions.

Professional web developers are required to know JavaScript, however, plenty of visual designers rely on developers to add behaviors to their designs. So while JavaScript is useful, learning to write it may not be mandatory for *all* web designers. Teaching JavaScript is outside the scope of this book; I recommend *Learning JavaScript* by Shelley Powers (O'Reilly, 2006) as a good starting place if you want to learn more.

The Web Design Layer Cake

Contemporary web design is commonly visualized as being made up of three separate "layers."

The content of the document with its (X)HTML markup makes up the Structure Layer. It forms the foundation upon which the other layers may be applied.

Once the structure of the document is in place, you can add style sheets to control how the content should appear. This is called the Presentation Layer.

Finally, the Behavior Layer includes the scripts that make the page an interactive experience.

Server-side programming

Some simple websites are collections of static HTML documents and image files, but most commercial sites have more advanced functionality such as forms handling, dynamically generated pages, shopping carts, content management systems, databases, and so on. These functions are handled by web applications running on the server. There are a number of programming languages and frameworks (listed in parentheses) that are used to create web applications, including:

- PHP (CakePHP, CodeIngniter, Drupal)
- Python (Django, TurboGears)

- Ruby (Ruby on Rails, Sinatra)
- JavaScript (Node.js, Rhino, SpiderMonkey)
- Java (Grails, Google Web Toolkit, JavaServer Faces)
- ASP.Net (DotNetNuke, ASP.Net MVC)

Developing web applications is programmer territory and is not expected of all web designers. However, that doesn't mean you can't offer such functionality to your clients. It is possible to get shopping carts, content management systems, mailing lists, and blogs as prepackaged solutions, without the need to program them from scratch.

What Do I Need to Buy?

It should come as no surprise that professional web designers require a fair amount of gear, both hardware and software. One of the most common questions I'm asked by my students is, "What should I get?" I can't tell you specifically what to buy, but I will provide an overview of the typical tools of the trade.

Bear in mind that while I've listed the most popular commercial software tools available, many of them have freeware or shareware equivalents that you can download if you're on a budget (try CNET's *Download.com*). With a little extra effort, you can get a full website up and running without big cash.

A Quick Introduction to XML

If you hang around the web design world at all, you're sure to hear the acronym XML (which stands for eXtensible Markup Language). XML is not a specific language in itself, but rather a robust set of rules for creating other markup languages.

To use a simplified example, if you were publishing recipes, you might use XML to create a custom markup language that includes the elements `<ingredient>`, `<instructions>`, and `<servings>` that accurately describe the types of information in your recipe documents. Once labeled correctly, that information can be treated as data. In fact, XML has proven to be a powerful tool for sharing data between applications. Despite the fact that XML was developed with the Web in mind, it has actually had a larger impact outside the web environment because of its data-handling capabilities. There are XML files working behind the scenes in an increasing number of software applications, such as Microsoft Office, Adobe Flash, and Apple iTunes.

Still, there are a number of XML languages that are used on the Web. The most prevalent is XHTML, which is HTML rewritten according to the stricter rules of XML (we'll talk more about XHTML in Chapter 10, What's Up, HTML5?). There is also RSS (Really Simple Syndication or RDF Site Summary), which allows your content to be shared as data and read with RSS feed readers; SVG (Scalable Vector Graphics), which uses tags to describe geometric shapes; and MathML, which is used to describe mathematical notation.

As a web designer, your direct experience with XML is likely to be limited to authoring documents in XHTML or perhaps adding an RSS feed or SVG images to a website. Developing new XML languages would be the responsibility of programmers or XML specialists.

Equipment

For a comfortable web development environment, I recommend the following equipment:

A solid, up-to-date computer. Macintosh, Windows, or Linux, is fine. Creative departments in professional web development companies tend to be Mac-based. Although it is nice to have a super-fast machine, the files that make up web pages are very small and tend not to be too taxing on computers. Unless you're getting into sound and video editing, don't worry if your current setup is not the very latest and greatest.

Extra memory. Because you'll tend to bounce between a number of applications, it's a good idea to have enough RAM installed on your computer that allows you to leave several memory-intensive programs running at the same time.

A large monitor. Although not a requirement, a large monitor makes life easier, particularly for a visual designer. (I've seen code-based developers get by just fine on an 11" MacBook Air.) The more monitor real estate you have, the more windows and control panels you can have open at the same time. You can also see more of your page to make design decisions.

If you're using large monitor, just make sure you design for users with smaller monitors and devices in mind.

A scanner and/or digital camera. If you anticipate making your own images and textures, you'll need some tools for creating them. I know a designer who has two scanners: one is the "good" scanner, and the other he uses to scan things like dead fish and rusty pans.

A second computer. Many web designers find it useful to have a test computer running a different platform than the computer they use for development (i.e., if you design on a Mac, test on a PC). Because browsers work differently on Macs than on Windows machines, it's critical to test your pages in as many environments as possible, and particularly on the current Windows operating system. If you are a hobbyist web designer working at home, check your pages on a friend's machine. Mac users should check out the "Run Windows on Your Mac" sidebar.

Mobile devices. The Web has gone mobile! That means it is absolutely critical that you test the appearance and performance of your site on a mobile browser on a smartphone or tablet device. You may already have a smartphone yourself. If you don't have a budget for devices with multiple platforms, ask your friends if you can spend a few minutes looking at your site on theirs. I have one web developer friend who checks out his designs on the phones at his local mobile carrier store (although you might quickly wear out your welcome).

Run Windows on Your Mac

If you have a Macintosh computer with an Intel chip running OS X (Leopard or later), you don't need a separate computer to test in a Windows environment. It is now possible to run Windows right on your Mac using the free Boot Camp application, which allows you to switch to Windows on reboot.

There are several other VM (Virtual Machine) products for Mac OS that allow you to toggle between Mac and Windows, including:

- VMFusion (*www.vmware.com/fusion*) is a commercial product with a free trial you can download.
- Parallels Desktop for Mac (*www.parallels.com*) is also a commercial product with a free trial.
- Oracle VirtualBox (*virtualbox.org*) is a free program that allows you to run a number of guest operating systems, including Windows and several flavors of Unix.

All VM products require that you purchase a copy of Microsoft Windows, but it sure beats buying a whole machine.

Software

There's no shortage of software available for creating web pages. In the early days, we just made do with tools originally designed for print. Today, there are wonderful tools created specifically with web design in mind that make the process more efficient. Although I can't list every available software release, I'd like to introduce you to the most common and proven web design tools. Note that you can download trial versions of many of these programs from the company websites, as listed in the "Popular Web Design Software Links" sidebar later in this chapter.

Web page authoring

Web-authoring tools are similar to desktop publishing tools, but the end product is a web page (an HTML file and its supporting files). These tools provide a visual "WYSIWYG" (What You See Is What You Get, pronounced "whizzy-wig") interface and shortcuts that save you from typing repetitive HTML and CSS. These tools won't excuse you from learning HTML. Even the most sophisticated tools won't generate HTML as clean or well-considered as a professional writing by hand, but they can speed up the process once you know what you're doing.

NOTE

To do the exercises in this book, all you'll need is the text editor that came with your operating system. No special programs are required.

The following are some popular web-authoring programs:

Adobe Dreamweaver. This is the hands-down industry standard due to its relatively clean code and advanced features.

Microsoft Expression Web (Windows only). Part of Microsoft's suite of professional design tools, MS Expression Web boasts standards-compliant code and CSS-based layouts.

Nvu (Linux, Windows, and Mac OS X). Don't want to pay for a WYSIWYG editor? Nvu (pronounced N-view, for "new view") is an open source tool that matches many of the features in Dreamweaver, and you can download it for free at *nvu.com*.

HTML editors

HTML editors (as opposed to WYSIWYG authoring tools) are designed to speed up the process of writing HTML by hand. They do not allow you edit the page visually, so you need to check your work in a browser. Many professional web designers actually prefer to author HTML documents by hand, and they tend to recommend the following:

TextPad (Windows only). TextPad is a simple and inexpensive plain-text code editor for Windows.

Sublime Text (Window, Mac, Linux). This inexpensive and up-and-coming text editor looks stripped down but has a lot of functionality (like color coding and full code overviews) that developers love.

Coda by Panic (Macintosh only). Coda users like its visual workflow, file management tools, and built-in terminal access.

TextMate by MacroMates (Macintosh only). This advanced text editor features project management tools and an interface that is integrated with the Mac operating system. It is growing in popularity because it is customizable, feature-rich, and inexpensive.

BBEdit by Bare Bones Software (Macintosh only). Lots of great shortcut features have made this the leading editor for Mac-based web developers.

Image editing and drawing software

You'll probably want to add images to your pages, so you will need an image-editing program. We'll look at some of the more popular programs in greater detail in Part IV. In the meantime, you may want to look into the following popular web-graphics-creation tools:

Adobe Photoshop. Photoshop is undeniably the industry standard for image creation in both the print and web worlds.

Adobe Photoshop Elements. This lighter version of Photoshop is designed for photo editing and management, but some hobbyists may find that it has all the tools necessary for putting images on web pages.

Adobe Illustrator. Because designers need to create logos, icons, and illustrations at a variety of sizes and resolutions, many start with a vector image in Illustrator for maximum flexibility. You can output web graphics directly from Illustrator, or bring them into Photoshop for additional fine-tuning.

Adobe Fireworks. This web graphics program combines an image editor with tools for creating vector-based illustrations. It also features advanced tools for outputting web graphics.

Corel Paint Shop Pro Photo (Windows only). This full-featured image editor is popular with the Windows crowd, primarily due to its low price.

GIMP, "GNU Image Manipulation Program" (Unix, Windows, Mac). This free image-editing program is similar to Photoshop.

Internet tools

Because you will be dealing with the Internet, you need to have some tools specifically for viewing and moving files over the network:

A variety of browsers. Because browsers render pages differently, you'll want to test your pages on as many browsers as possible, both on the desktop and on mobile devices. The following lists the desktop browsers most commonly used on Windows and Macintosh operating systems:

Windows:	Macintosh OS X:
Internet Explorer (the current version and at least two prior versions)	Safari
Chrome	Chrome
Firefox	Firefox
Safari	Opera
Opera	

And don't ignore the mobile browsers! The following list is an overview of the most commonly used mobile web browsers as of this writing (although who knows what mobile browsers will be important by the time you read this):

- Mobile Safari (iOS)
- Android Browser (Android)
- BlackBerry Browser (RIM)
- Nokia Series 40 and Nokia Browser for Symbian
- Opera Mobile and Mini (installed on any device)
- Internet Explorer Mobile (Windows Phone)
- Silk (Kindle Fire)

A file-transfer program (FTP). An FTP program enables you to upload and download files between your computer and the computer that will serve your pages to the web. The web authoring tools listed earlier all have FTP programs built right in. There are also dedicated FTP programs, as listed here:

Windows	Macintosh OS X:
WS_FTP	Transmit
CuteFTP	Cyberduck
AceFTP	Fetch
Filezilla	

Terminal application. If you know your way around the Unix operating system, you may find it useful to have a terminal (command-line) application that allows you to type Unix commands on the server. This may be useful for setting file permissions, moving or copying files and directories, or managing the server software.

Windows users can install a Linux emulator called Cygwin for command-line access. There is also PuTTY, a free Telnet/SSH client. Mac OS X includes an application called Terminal that is a full-fledged terminal application, giving you access to the underlying Unix system and the ability to use SSH to access other command-line systems over the Internet.

AT A GLANCE

Popular Web Design Software Links

Web page authoring

Adobe Dreamweaver *www.adobe.com*

Microsoft Expression Web *www.microsoft.com/products/expression*

Nvu (open source web page editor) *www.nvu.com*

HTML editing

TextMate by MacroMates for Mac OS *www.macromates.com*

Sublime Text *www.sublimetext.com*

TextPad for Windows *www.textpad.com*

Coda by Panic Software *www.panic.com/coda/*

BBEdit by Bare Bones Software *www.barebones.com*

Image editing and drawing

Adobe Photoshop *www.adobe.com*

Adobe Photoshop Elements *www.adobe.com*

Adobe Illustrator *www.adobe.com*

Adobe Fireworks *www.adobe.com*

Corel Paint Shop Pro Photo *www.corel.com/paintshoppro*

GIMP *gimp.org*

Browsers

Microsoft Internet Explorer (Windows only) *www.microsoft.com/windows/internet-explorer/*

Firefox *www.firefox.com*

Google Chrome *www.google.com/chrome*

Opera *www.opera.com*

Safari *www.apple.com/safari*

Networking

WS_FTP, CuteFTP, AceFTP, and others for Windows available at: *www.download.com*

Transmit (for Macintosh OSX) *www.panic.com/transmit*

Cyberduck (for Macintosh OSX) *cyberduck.ch*

Fetch (for Macintosh OSX) *fetchsoftworks.com*

Cygwin (Linux emulator for Windows) *www.cygwin.com*

PuTTY (telnet/SSH terminal emulator) *www.chiark.greenend.org.uk/~sgtatham/putty/*

What You've Learned

The lesson to take away from this chapter is: "You don't have to learn everything." And even if you want to learn everything eventually, you don't need to learn it all at once. So relax, and don't worry. The other good news is that, while many professional tools exist, it is possible to create a basic website and get it up and running without spending much money by using freely available or inexpensive tools and your existing computer setup.

As you'll soon see, it's easy to get started making web pages—you will be able to create simple pages by the time you're done reading this book. From there, you can continue adding to your bag of tricks and find your particular niche in web design.

exercise 1-1 | Taking stock

Now that you're taking that first step in learning web design, it might be a good time to take stock of your assets and goals. Using the lists in this chapter as a general guide, try jotting down answers to the following questions:

- What are your web design goals? To become a professional web designer? To make personal websites only?
- Which aspects of web design interest you the most?
- What current skills do you have that will be useful in creating web pages?
- Which skills will you need to brush up on?
- Which hardware and software tools do you already have for web design?
- Which tools do you need to buy? Which tools would you like to buy eventually?

Test Yourself

Each chapter in this book ends with a few questions that you can answer to see if you picked up the important bits of information. Answers appear in Appendix A.

1. Match these web professionals with the final product they might be responsible for producing.

A. Graphic designer	_____ HTML and CSS documents
B. Production department	_____ PHP scripts
C. User experience designer	_____ Photoshop page sketch
D. Web programmer	_____ Storyboards

2. What does the W3C do?

3. Match the web technology with its appropriate task:

A. HTML	_____ Checks a form field for a valid entry
B. CSS	_____ Creates a custom server-side web application
C. JavaScript	_____ Identifies text as a second-level heading
D. PHP	_____ Defines a new markup language for sharing financial information
E. XML	_____ Makes all second-level headings blue

4. What is the difference between frontend and backend web development?

5. What is the difference between a web-authoring program and an HTML-editing tool?

HOW THE WEB WORKS

I got started in web design in early 1993—pretty close to the start of the Web itself. In web time, that makes me an old-timer, but it's not so long ago that I can't remember the first time I looked at a web page. It was difficult to tell where the information was coming from and how it all worked.

This chapter sorts out the pieces and introduces some basic terminology. We'll start with the big picture and work down to specifics.

IN THIS CHAPTER

An explanation of the Web, as it relates to the Internet

The role of the server

The role of the browser

Introduction to URLs and their components

The anatomy of a web page

The Internet Versus the Web

No, it's not a battle to the death, just an opportunity to point out the distinction between these two words that are increasingly being used interchangeably.

The Internet is a network of connected computers. No company owns the Internet; it is a cooperative effort governed by a system of standards and rules. The purpose of connecting computers together, of course, is to share information. There are many ways information can be passed between computers, including email, file transfer (FTP), and many more specialized modes upon which the Internet is built. These standardized methods for transferring data or documents over a network are known as protocols.

The Web (originally called the World Wide Web, thus the "www" in site addresses) is just one of the ways information can be shared over the Internet. It is unique in that it allows documents to be linked to one another using hypertext links—thus forming a huge "web" of connected information. The Web uses a protocol called HTTP (HyperText Transfer Protocol). That acronym should look familiar because it is the first four letters of nearly all website addresses, as we'll discuss in an upcoming section.

The Web is a subset of the Internet. It is just one of many ways information can be transferred over networked computers.

Serving Up Your Information

Let's talk more about the computers that make up the Internet. Because they "serve up" documents upon request, these computers are known as servers. More accurately, the server is the software (not the computer itself) that

A Brief History of the Web

The Web was born in a particle physics laboratory (CERN) in Geneva, Switzerland in 1989. There a computer specialist named Tim Berners-Lee first proposed a system of information management that used a "hypertext" process to link related documents over a network. He and his partner, Robert Cailliau, created a prototype and released it for review. For the first several years, web pages were text-only. It's difficult to believe that in 1992, the world had only about 50 web servers, total.

The real boost to the Web's popularity came in 1992 when the first graphical browser (NCSA Mosaic) was introduced, and the Web broke out of the realm of scientific research into mass media. The ongoing development of web technologies is overseen by the World Wide Web Consortium (W3C).

If you want to dig deeper into the Web's history, check out this site:

W3C's History Archives

www.w3.org/History.html

allows the computer to communicate with other computers; however, it is common to use the word "server" to refer to the computer as well. The role of server software is to wait for a request for information, then retrieve and send that information back as quickly as possible.

There's nothing special about the computers themselves...picture anything from a high-powered Unix machine to a humble personal computer. It's the server software that makes it all happen. In order for a computer to be part of the Web, it must be running special web server software that allows it to handle Hypertext Transfer Protocol transactions. Web servers are also called "HTTP servers."

There are many server software options out there, but the two most popular are Apache (open source software) and Microsoft Internet Information Services (IIS). Apache is freely available for Unix-based computers and comes installed on Macs running Mac OS X. There is a Windows version as well. Microsoft IIS is part of Microsoft's family of server solutions.

Every computer and device (modem, router, smartphone, cars, etc.) connected to the Internet is assigned a unique numeric IP address (IP stands for Internet Protocol). For example, the computer that hosts *oreilly.com* has the IP address 208.201.239.100. All those numbers can be dizzying, so fortunately, the Domain Name System (DNS) was developed to allow us to refer to that server by its domain name, "oreilly.com", as well. The numeric IP address is useful for computer software, while the domain name is more accessible to humans. Matching the text domain names to their respective numeric IP addresses is the job of a separate DNS server.

It is possible to configure your web server so that more than one domain name is mapped to a single IP address, allowing several sites to share a single server.

TERMINOLOGY

Open Source

Open source software is developed as a collaborative effort with the intent to make its source code available to other programmers for use and modification. Open source programs are usually available for free.

No More IP Addresses

The IANA, the organization that assigns IP numbers, handed out its last bundle of IP addresses on February 3, 2011. That's right, no more ###.###.###.###-style IPs. That format of IP address (called IPv4) is able to produce 4.3 billion unique addresses, which seemed like plenty when the Internet "experiment" was first conceived in 1977. There was no way the creators could anticipate that one day every phone, television, and object on store shelves would be clamoring for one.

The solution is a new IP format (IPv6, already in the works) that allows for trillions and trillions of unique IP numbers, with the slight snag that it is incompatible with our current IPv4-based network, so IPv6 will operate as a sort of parallel Internet to the one we have today. Eventually, IPv4 will be phased out, but some say it will take decades.

A Word About Browsers

We now know that the server does the servin', but what about the other half of the equation? The software that does the requesting is called the client. People use desktop browsers, mobile browsers, and other assistive technologies (such as screen readers) as clients to access documents on the Web. The server returns the documents for the browser (also referred to as the user agent in technical circles) to display.

The requests and responses are handled via the HTTP protocol, mentioned earlier. Although we've been talking about "documents," HTTP can be used to transfer images, movies, audio files, data, scripts, and all the other web resources that commonly make up web sites and applications.

It is common to think of a browser as a window on a computer monitor with a web page displayed in it. These are known as graphical browsers or desktop browsers and for a long time, they were the only web-viewing game in town. The most popular desktop browsers as of this writing include Internet Explorer for Windows, Chrome, Firefox, and Safari, with Opera bringing up the rear. These days, however, more and more people are accessing the Web on the go using browsing clients built into mobile phones or tablets.

It is also important to keep alternative web experiences in mind. Users with sight disabilities may be listening to a web page read by a screen reader (or simply make their text extremely large). Users with limited mobility may use assistive devices to access links and to type. The sites we build must be accessible and usable for all users, regardless of their browsing experiences.

Even on the desktop browsers that first introduced us to the wide world of the Web, pages may look and perform differently from browser to browser. This is due to varying support for web technologies and the users' ability to set their own browsing preferences.

TERMINOLOGY

Server-side and Client-side

Often in web design, you'll hear reference to "client-side" or "server-side" applications. These terms are used to indicate which machine is doing the processing. Client-side applications run on the user's machine, while server-side applications and functions use the processing power of the server computer.

Intranets and Extranets

When you think of a website, you generally assume that it is accessible to anyone surfing the Web. However, many companies take advantage of the awesome information sharing and gathering power of websites to exchange information just within their own business. These special web-based networks are called intranets. They are created and function like ordinary websites, but they use special security devices (called firewalls) that prevent the outside world from seeing them. Intranets have lots of uses, such as sharing human resource information or providing access to inventory databases.

An extranet is like an intranet, only it allows access to select users outside of the company. For instance, a manufacturing company may provide its customers with passwords that allow them to check the status of their orders in the company's orders database. Of course, the passwords determine which slice of the company's information is accessible.

Web Page Addresses (URLs)

Every page and resource on the Web has its own special address called a URL, which stands for Uniform Resource Locator. It's nearly impossible to get through a day without seeing a URL (pronounced "U-R-L," not "erl") plastered on the side of a bus, printed on a business card, or broadcast on a television commercial. Web addresses are fully integrated into modern vernacular.

Some URLs are short and sweet. Others may look like crazy strings of characters separated by dots (periods) and slashes, but each part has a specific purpose. Let's pick one apart.

The parts of a URL

A complete URL is generally made up of three components: the protocol, the site name, and the absolute path to the document or resource, as shown in Figure 2-1.

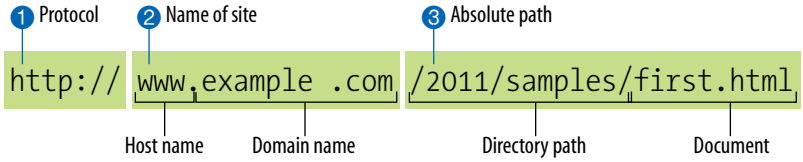

Figure 2-1. The parts of a URL.

❶ `http://`

The first thing the URL does is define the protocol that will be used for that particular transaction. The letters HTTP let the server know to use Hypertext Transfer Protocol, or get into "web mode."

❷ `www.example.com`

The next portion of the URL identifies the website by its domain name. In this example, the domain name is example.com. The "www." part at the beginning is the particular host name at that domain. The host name "www" has become a convention, but is not a rule. In fact, sometimes the host name may be omitted. There can be more than one website at a domain (sometimes called subdomains). For example, there might also be *development.example.com*, *clients.example.com*, and so on.

❸ `/2012/samples/first.html`

This is the absolute path through directories on the server to the requested HTML document, *first.html*. The words separated by slashes are the directory names, starting with the root directory of the host (as indicated by the initial `/`). Because the Internet originally comprised computers running the Unix operating system, our current way of doing things still

Hey, There's No http:// on That URL!

Because nearly all web pages use the Hypertext Transfer Protocol, the **`http://`** part is often just implied. This is the case when site names are advertised in print or on TV, as a way to keep the URL easy to remember.

Additionally, browsers are programmed to add **`http://`** automatically as a convenience to save you some keystrokes. It may seem like you're leaving it out, but it is being sent to the server behind the scenes.

When we begin using URLs to create hyperlinks in HTML documents in Chapter 6, Adding Links, you'll learn that it is necessary to include the protocol when making a link to a web page on another server.

NOTE

Sometimes you'll see a URL that begins with `https://`*. This is an indication that it is a secure server transaction. Secure servers have special encryption devices that hide delicate content, such as credit card numbers, while they are transferred to and from the browser. Look for it the next time you're shopping online.*

follows many Unix rules and conventions, hence the / separating directory names.

To sum it up, the URL in Figure 2-1 says it would like to use the HTTP protocol to connect to a web server on the Internet called *www.example.com* and request the document *first.html* (located in the *samples* directory, which is in the *2012* directory).

Default files

Obviously, not every URL you see is so lengthy. Many addresses do not include a filename, but simply point to a directory, like these:

```
http://www.oreilly.com
http://www.jendesign.com/resume/
```

When a server receives a request for a directory name rather than a specific file, it looks in that directory for a default document, typically named *index.html*. So when someone types the above URLs into their browser, what they'll actually see is this:

```
http://www.oreilly.com/index.html
http://www.jendesign.com/resume/index.html
```

The name of the default file (also referred to as the index file) may vary, and depends on how the server is configured. In these examples, it is named *index.html*, but some servers use the filename *default.htm*. If your site uses server-side programming to generate pages, the index file might be named *index.php* or *index.asp*. Just check with your server administrator or the tech support department at your hosting service to make sure you give your default file the proper name.

Another thing to notice is that in the first example, the original URL did not have a trailing slash to indicate it was a directory. When the slash is omitted, the server simply adds one if it finds a directory with that name.

The index file is also useful for security. Some servers (depending on their configuration) display the contents of the directory if the default file is not found. Figure 2-2 shows how the documents in the *housepics* directory are exposed as the result of a missing default file. One way to prevent people from snooping around in your files is to be sure there is an index file in every directory. Your server administrator may also add other protections to prevent your directories from displaying in the browser.

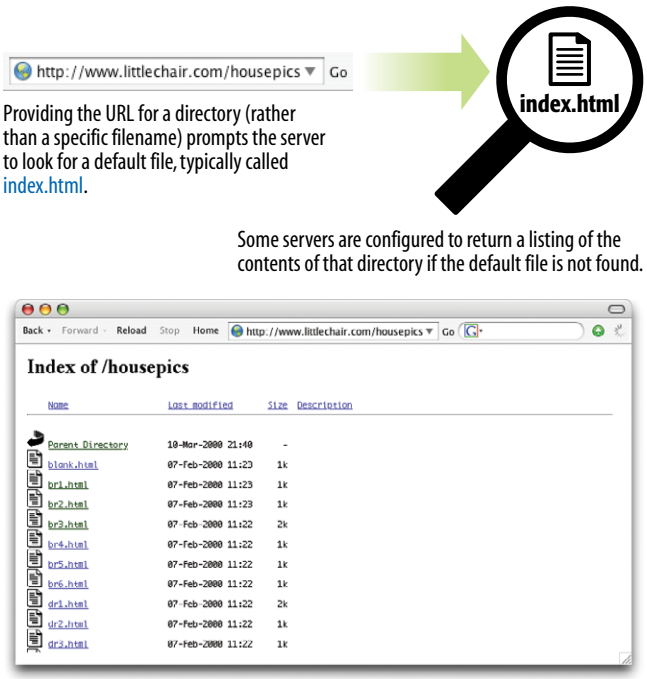

Figure 2-2. *Some servers display the contents of the directory if an index file is not found.*

The Anatomy of a Web Page

We're all familiar with what web pages look like in the browser window, but what's happening "under the hood?"

At the top of Figure 2-3, you see a minimal web page as it appears in a graphical browser. Although you see it as one coherent page, it is actually assembled from four separate files: an HTML document (*index.html*), a style sheet (*kitchen.css*), and two graphics (*foods.gif* and *spoon.gif*). The HTML document is running the show.

HTML documents

You may be as surprised as I was to learn that the graphically rich and interactive pages we see on the Web are generated by simple, text-only documents. This text file is referred to as the source document.

Take a look at *index.html*, the source document for the Jen's Kitchen web page. You can see it contains the text content of the page plus special tags (indicated with angle brackets, < and >) that describe each element on the page.

Adding descriptive tags to a text document is known as "marking up" the document. Web pages use a markup language called HyperText Markup Language, or HTML for short, which was created especially for documents with hypertext links. HTML defines dozens of text elements that make up documents such as headings, paragraphs, emphasized text, and of course, links. There are also elements that add information about the document (such as its title), media such as images and videos, and widgets for form inputs, just to name a few.

It is worth noting briefly that there are actually several versions of HTML in use today. The most firmly established are HTML version 4.01 and its stricter cousin, XHTML 1.0. And you may have heard how all the Web is a-buzz with the emerging HTML5 specification that is designed to better handle web applications and is gradually gaining browser support. I will give you the lowdown on all the various versions and what makes them unique in Chapter 10, What's Up, HTML5?. In the meantime, we have to cover some basics that apply regardless of the HTML flavor you choose.

A quick introduction to HTML markup

You'll be learning the nitty-gritty of markup in Part II, so I don't want to bog you down with too much detail right now, but there are a few things I'd like to point out about how HTML works and how browsers interpret it.

Read through the HTML document in Figure 2-3 and compare it to the browser results. It's easy to see how the elements marked up with HTML tags in the source document correspond to what displays in the browser window.

exercise 2-1 | View source

You can see the HTML file for any web page by choosing View → Page Source or (View → Source) in your browser's menu. Your browser typically opens the source document in a separate window. Let's take a look under the hood of a web page.

1. Enter this URL into your browser:

   www.learningwebdesign.com/4e/materials/chapter02/kitchen.html

   You should see the Jen's Kitchen web page from Figure 2-3.
2. Select View → Page Source (or View → Source) from the browser menu. On Chrome and Opera, View Source is located in the Developer menu. A window opens showing the source document shown in the figure.
3. The source for most sites is considerably more complicated. View the source of *oreilly.com* or the site of your choice. Don't worry if you don't understand what's going on. Much of it will look more familiar by the time you are done with this book.

WARNING

Keep in mind that while learning from others' work is fine, the all-out stealing of other people's code is poor form (or even illegal). If you want to use code as you see it, ask for permission and always give credit to those who did the work.

The web page shown in this browser window consists of four separate files: an HTML text document, a style sheet and two images. Tags in the HTML source document give the browser instructions for how the text is structured and where the images should be placed.

index.html

```
<!DOCTYPE html>
<html>
<head>
<title>Jen's Kitchen</title>
<link rel="stylesheet" href="kitchen.css" type="text/css" >
</head>

<body>
<h1><img src="foods.gif" alt="food illustration"> Jen&rsquo;s Kitchen</h1>

<p>If you love to read about <strong>cooking and eating</strong>, would like to find out about
of some of the best restaurants in the world, or just want a few choice recipes to add to your
collection, <em>this is the site for you!</em></p>

<p><img src="spoon.gif" alt="spoon illustration"> Your pal, Jen at Jen's Kitchen</p>
<hr>
<p><small>Copyright 2011, Jennifer Robbins</small></p>
</body>
</html>
```

kitchen.css

```
body { font: normal 1em Verdana; margin: 1em 10%;}
h1 { font: italic 3em Georgia; color: rgb(23, 109, 109); margin: 1em 0 1em;}
img { margin: 0 20px 0 0; }
h1 img { margin-bottom: -20px; }
small { color: #666666; }
```

foods.gif

spoon.gif

Figure 2-3. The source file and images that make up a simple web page.

First, you'll notice that the text within brackets (for example, **<body>**) does not display in the final page. The browser displays only what's between the tags—the content of the element. The markup is hidden. The tag provides the name of the HTML element—usually an abbreviation such as "h1" for "heading level 1," or "em" for "emphasized text."

Second, you'll see that most of the HTML tags appear in pairs surrounding the content of the element. In our HTML document, **<h1>** indicates that the following text should be a level-1 heading; **</h1>** indicates the end of the heading. Some elements, called empty elements, do not have content. In our sample, the **<hr>** tag indicates an empty element that tells the browser to "insert a thematic divider here" (most browsers indicate the thematic divider with a horizontal rule [line], which is how the **hr** element got its initials).

Because I was unfamiliar with computer programming when I first began writing HTML, it helped me to think of the tags and text as "beads on a string" that the browser interprets one by one, in sequence. For example, when the browser encounters an open bracket (**<**), it assumes all of the following characters are part of the markup until it finds the closing bracket (**>**). Similarly, it assumes all of the content following an opening **<h1>** tag is a heading until it encounters the closing **</h1>** tag. This is the manner in which the browser parses the HTML document. Understanding the browser's method can be helpful when troubleshooting a misbehaving HTML document.

But where are the pictures?

Obviously, there are no pictures in the HTML file itself, so how do they get there when you view the final page?

You can see in Figure 2-3 that each image is a separate file. The images are placed in the flow of the text with the HTML image element (**img**) that tells the browser where to find the graphic (its URL). When the browser sees the **img** element, it makes another request to the server for the image file, and then places it in the content flow. The browser software brings the separate pieces together into the final page. Videos and other embedded media files are added in much the same way.

The assembly of the page generally happens in an instant, so it appears as though the whole page loads all at once. Over slow connections or if the page includes huge graphics or media files, the assembly process may be more apparent as images lag behind the text. The page may even need to be redrawn as new images arrive (although you can construct your pages in a way to prevent that from happening).

Adding a little style

I want to direct your attention to one last key ingredient of our minimal page. Near the top of the HTML document there is a `link` element that points to the style sheet document *kitchen.css*. That style sheet includes a few lines of instructions for how the page should look in the browser. These are style instructions written according to the rules of Cascading Style Sheets (CSS). CSS allows designers to add visual style instructions (known as the document's presentation) to the marked-up text (the document's structure, in web design terminology). In Part III, you'll really get to know the power of Cascading Style Sheets.

Figure 2-4 shows the Jen's Kitchen page with and without the style instructions. Browsers come equipped with default styles for every HTML element they support, so if an HTML document lacks its own custom style instructions, the browser will use its own (that's what you see in the screen shot on the right). Even just a few style rules can make big improvements to the appearance of a page.

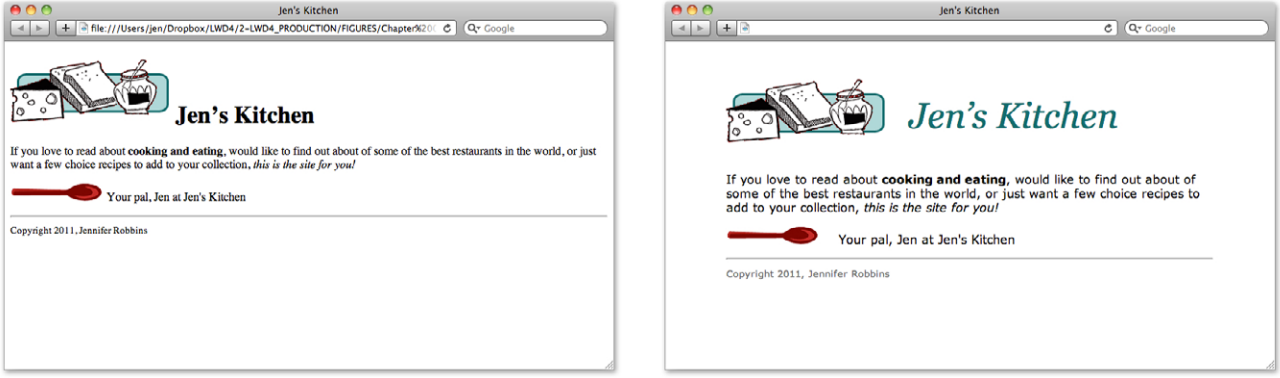

Figure 2-4. The Jen's Kitchen page before (left) and after (right) style rules.

Adding Behaviors with JavaScript

In addition to a document's structure and presentation, there is also a behavior component that defines how things *work*. On the Web, behaviors are defined by a scripting language called JavaScript. We'll touch on it lightly in this book in Part IV, but learning JavaScript from scratch is more than we can take on here. Many designers (myself included) rely on people with scripting experience to add functionality to sites. However, knowing how to write JavaScript is becoming more essential to the "web designer" job description.

Putting It All Together

To wrap up our introduction to how the web works, let's trace a typical stream of events that occurs with every web page that appears on your screen (Figure 2-5).

1. You request a web page by either typing its URL (for example, *http://jenskitchensite.com*) directly in the browser or by clicking on a link on a page. The URL contains all the information needed to target a specific document on a specific web server on the Internet.
2. Your browser sends an HTTP Request to the server named in the URL and asks for the specific file. If the URL specifies a directory (not a file), it is the same as requesting the default file in that directory.
3. The server looks for the requested file and issues an HTTP response.
   a. If the page cannot be found, the server returns an error message. The message typically says "404 Not Found," although more hospitable error messages may be provided.
   b. If the document *is* found, the server retrieves the requested file and returns it to the browser.
4. The browser parses the HTML document. If the page contains images (indicated by the HTML `img` element) or other external resources like scripts, the browser contacts the server again to request each resource specified in the markup.
5. The browser inserts each image in the document flow where indicated by the `img` element. And *voila*! The assembled web page is displayed for your viewing pleasure.

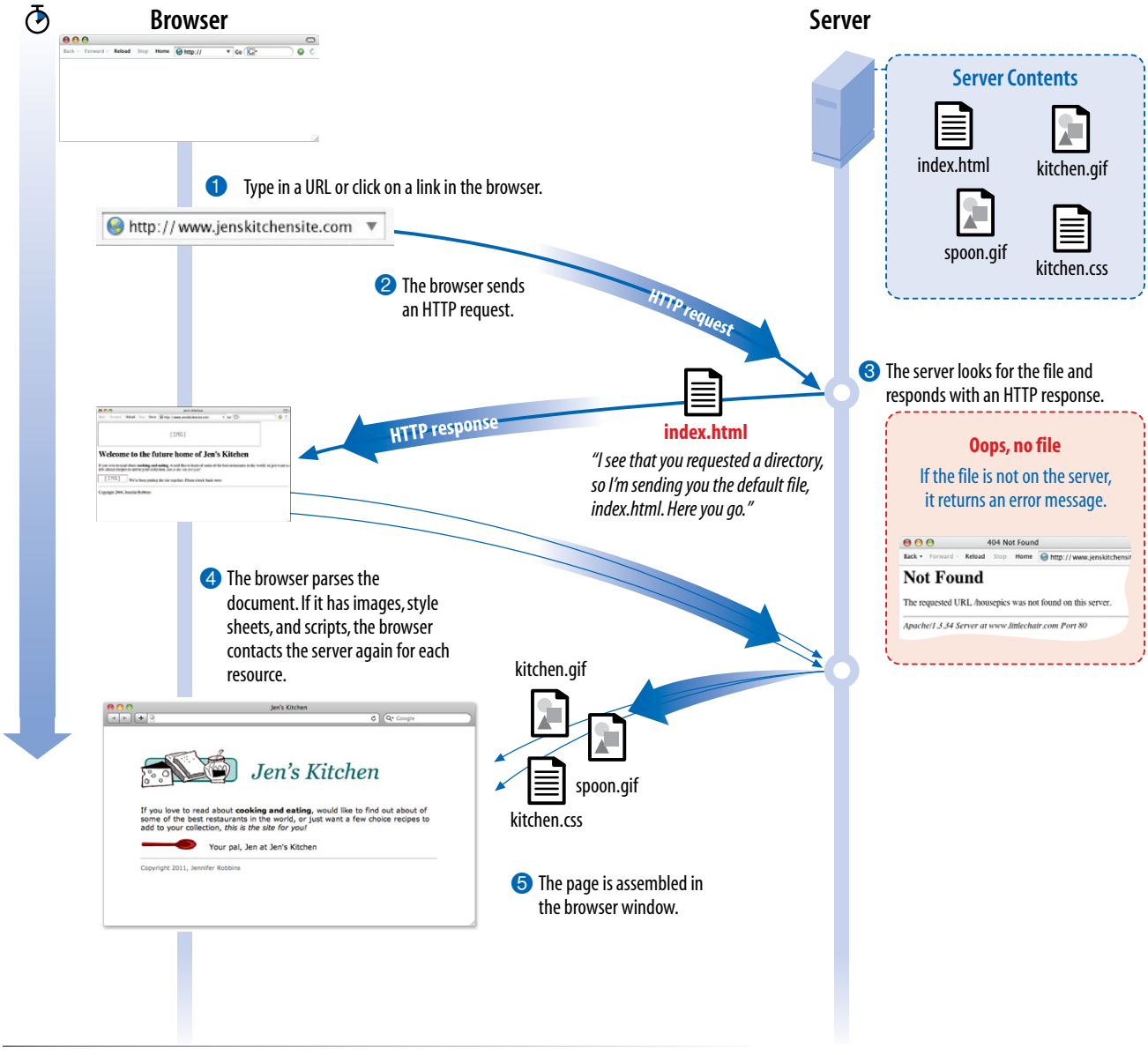

Figure 2-5. How browsers display web pages.

Test Yourself

Let's play a round of "Identify that Acronym!" The following are a few basic web terms mentioned in this chapter. Answers are in Appendix A.

1) HTML	_____	a) Home of Mosaic, the first graphical browser
2) W3C	_____	b) The location of a web document or resource
3) CERN	_____	c) The markup language used to describe web content
4) CSS	_____	d) Matches domain names with numeric IP addresses
5) HTTP	_____	e) A protocol for file transfer
6) IP	_____	f) Protocol for transferring web documents on the Internet
7) URL	_____	g) The language used to instruct how web content looks
8) NCSA	_____	h) Particle physics lab where the Web was born
9) DNS	_____	i) Internet Protocol
10) FTP	_____	j) The organization that monitors web technologies

SOME BIG CONCEPTS YOU NEED TO KNOW

As the Web matures and the number of devices we access it from increases exponentially, our jobs as web designers and developers get significantly more complicated. Frankly, there's a lot more going on out there than I can fit in this book. In the chapters that follow, I will focus on the basic building blocks of web design—HTML elements, CSS styles, a taste of JavaScript, and web graphics production—that will give you a solid foundation for the further development of your skills.

But before we get to the nuts and bolts, I want introduce some Big Concepts that I think every web designer needs to know. We'll look at ideas and concerns that inform our decisions and contribute to the contemporary web design environment. I'll be referring back to the terminology introduced here frequently.

The heart of the matter is that as web designers, we never know exactly how the pages we create will be viewed. We don't know which of the hundreds of browsers might be used, whether it is on a desktop computer or something more portable, how large the browser window will be, what fonts are installed, whether functionality such as JavaScript is enabled, the speed of the Internet connection, whether they are being read by a screen reader, and so on. I think you get the picture. The Big Concepts in this chapter are primarily reactions to and methods for coping with the inescapable element of the Unknown in our medium. They include:

- The multitude of devices
- Web standards
- Progressive enhancement
- Responsive web design
- Accessibility
- Site performance

Because we're just getting started, I will keep the descriptions brief and fairly non-technical. My goal is that you have a basic understanding of what I mean by terms like "progressive enhancement" when you encounter them

IN THIS CHAPTER

The Web on mobile devices
The benefits of web standards
Progressive enhancement
Responsive web design
Accessibility
Site performance

in a later exercise. Many excellent articles and books have been written on each of these topics and their related production techniques, and I'll provide pointers to resources for further reading.

A Dizzying Multitude of Devices

Until 2007, we could be relatively certain that our users were visiting our sites while sitting at their desk, looking at a large monitor, using a speedy Internet connection. We had all more or less settled on 960 pixels as a good width for a web page. Back then, our biggest concern was dealing with the dozen or so desktop browsers and jumping through a few extra hoops to support quirky old versions of Internet Explorer. And we thought we had it rough!

Although you could access web pages and web content on mobile phones prior to 2007, the introduction of the iPhone and Android smartphones as well as a more widespread 3G network heralded a huge shift in how, when, and where we do our web surfing (particularly in the United States, which lagged behind Asia and the EU in mobile technology). Since then, we've seen the introduction of tablets of all different dimensions, as well as web browsers on TVs and other devices. And the diversity is only going to increase. I think mobile guru Brad Frost sums it up nicely in his illustrations in Figure 3-1.

Figure 3-1. Brad Frost sums up the reality of device diversity nicely (bradfrostweb.com).

The challenges of designing for all of these devices goes beyond addressing differing screen sizes. There is a world of difference between using a site over a broadband connection and over a 3G or EDGE network. There are also varying contexts to consider. Users may be sitting at a desk, enjoying some recreational browsing at home, or getting information quickly on the go. Designers need to resist making assumptions about network speed and context based on the screen size. It's not uncommon to leisurely browse the Web on a smartphone while sitting on the couch at home with a solid WiFi connection. And new iPads with high-resolution displays may be accessing the Internet on a pokey 3G connection. In other words, it's complicated!

Soon, more people will be accessing the Web on their mobile and alternative devices than on a desktop computer. Already a significant portion of Americans use their mobile phones as their *only* access to the Internet. That means it is critical to get it right. But to be honest, as of this writing, we haven't entirely figured out how to make all the content we are accustomed to seeing at our desk fit on our handheld devices with an equally pleasing experience. Great strides are being made, and there is a wonderful spirit of collaboration while we figure it out, but the fact is that our tools and technologies are not quite suited for the task and will take some time to catch up.

What I want you to learn here is that the way you see your design as you're working on it on your nice desktop machine is not how it will be experienced by everyone. This fact should be on the mind of all web design professionals.

Mobile Web?

You may hear designers use the term Mobile Web, but the truth is (as Stephen Hay put it in a tweet in 2011; see Figure 3-2) there is no Mobile Web any more than there is a Desktop Web, or a Tablet Web, or so on. There is just the Web, and it can be accessed from all manner of devices. As of this writing, the term "mobile web" is used as sort of a catchall for describing our efforts to adapt our desktop design skills to accommodate a much wider variety of use cases. And as we are finding out, there is more than one way to crack that nut.

Figure 3-2. Stephen Hay's tweet from January 2011. Read his follow-up article at www.the-haystack.com/2011/01/07/there-is-no-mobile-web/.

For further reading

- In his article "*The Coming Zombie Apocalypse*," Scott Jensen takes a thoughtful look at the onslaught of inexpensive networked devices (*designmind.frogdesign.com/blog/the-coming-zombie-apocalypse-small-cheap-devices-will-disrupt-our-old-school-ux-assumptions.htm*). It is definitely worth a read.
- *Mobile First*, by Luke Wroblewski (A Book Apart). Luke was way ahead of the curve on insisting sites work well on mobile devices, and he shares his perspective in this little book that is jam-packed with ideas.

- The Future Friendly site (*futurefriend.ly*) includes a call to arms composed by many of the brightest mobile designers of the day. They concluded that with the landscape changing so rapidly, we can't make our designs future-proof, but we can make them "future friendly." They assemble a number of tips and resources for doing so.

Sticking with the Standards

Sticking with web standards is your primary tool for ensuring your site is as consistent as possible.

So how do we deal with this diversity? One good start is to follow the HTML, CSS, and JavaScript standards as documented by the World Wide Web Consortium (W3C). Sticking with web standards is your primary tool for ensuring your site is as consistent as possible on all standards-compliant browsers (that's approximately 99% of browsers in current use). It also helps make your content forward-compatible as web technologies and browser capabilities evolve. Another benefit is you can tell your clients that you create "standards-compliant" sites, and they will like you more.

The notion of standards compliance may seem like a no-brainer, but it used to be that everyone, including the browser makers, played fast and loose with HTML and scripting. The price we paid was incompatible browser implementations and the need to create sites twice to make them work for everyone. I talk more about web standards throughout this book, so I won't go into too much detail here. Suffice it to say that the web standards are your friends. Everything you learn in this book will get you headed in the right direction.

For further reading

The bible for standards compliance and how it makes good business sense is *Designing with Web Standards* by Jeffrey Zeldman (New Riders). Go read it (when you're done with this book, of course).

Progressive Enhancement

Progressive enhancement is a strategy for coping with unknown browser capabilities.

With a multitude of browsers comes a multitude of levels of support for the web standards. In fact, no browser has implemented all the standards 100%, and there are always new technologies that are slowly gaining steam. Furthermore, users can set their own browser preferences, so they may have a browser that supports JavaScript but have chosen to turn it off. The point here is that we are faced with a wide range of browser capabilities—from basic HTML support only to all the bells and whistles.

Progressive enhancement is one strategy for dealing with unknown browser capabilities. When designing with progressive enhancement, you start with a baseline experience that makes the content or functionality available to even the most rudimentary browsers or assistive devices. From there, you layer on more advanced features for the browsers that can handle them. You might finish with some "nice to have" effects like animation or rounded corners on boxes that enhance the experience for users with the most advanced browsers, but that aren't really critical to the brand or message.

Progressive enhancement is an approach that informs all aspects of page design and production, including HTML, CSS, and JavaScript.

Authoring strategy

When an HTML document is written in logical order and its elements are marked up in a meaningful way, it will be usable on the widest range of browsing environments, including the oldest browsers, future browsers, and mobile and assistive devices. It may not look exactly the same, but the important thing is that your content is available. It also ensures that search engines like Google will catalog the content correctly. A clean HTML document with its elements accurately and thoroughly described are the foundation for accessibility.

Styling strategy

You can create layers of experiences simply by taking advantage of the way browsers parse style sheet rules. Without going into too much technical detail, you can write a style rule that makes an element background red, but also include a style that gives it a cool gradient (a blend from one color to another) for browsers that know how to render gradients. Or you can use a cutting-edge CSS selector to deliver certain styles only to cutting-edge browsers. The knowledge that browsers simply ignore properties and rules they don't understand gives you license to innovate without bringing older browsers to their knees. You just have to be mindful to take care of styling the baseline experience first, then add improvements once the minimum requirements are met.

Scripting strategy

JavaScript is the scripting language that makes web pages interactive and dynamic (updating content on the fly or in response to user input). The Web would be a lot of static brochureware without it. Like other web technologies, there are discrepancies in how browsers handle JavaScript (particularly on non-desktop devices), and some users opt to turn it off entirely. The first rule in progressive enhancement is to make sure basic functionality—such as linking from page to page or accomplishing essential tasks like data submission via forms—is intact even when JavaScript is off. In this way, you ensure the baseline experience, and enhance it when JavaScript is available.

NOTE

Progressive enhancement is the flip side of an older approach to browser diversity called graceful degradation, in which you design the fully enhanced experience first, then create a series of fallbacks for non-supporting browsers.

For further reading

There is no better introduction to the progressive enhancement approach than the book *Adaptive Web Design: Crafting Rich Experiences with Progressive Enhancement*, by Aaron Gustafson (Easy Readers). Aaron is a technical reviewer for this book, but I'd be recommending his excellent primer even if he weren't. See *easy-readers.net/books/adaptive-web-design/* for more information.

Once you have more web development chops, the book *Designing with Progressive Enhancement*, by Todd Parker, Patty Toland, Scott Jehl, and Maggie Costello Wachs (New Riders), is an excellent deep-dive into techniques and best practices. Read more about it at *filamentgroup.com/dwpe/*.

Responsive Web Design

By default, most browsers on small devices such as smartphones and tablets shrink a web page down to fit the screen and provide mechanisms for zooming and moving around the page. Although it technically works, it is not a great experience. The text is too small to read, the links too small to tap, and all that zooming and panning around is distracting.

Responsive web design is a strategy for dealing with unknown screen size.

Responsive web design is a strategy for providing custom layouts to devices based on the size of the viewport (browser window). The trick to responsive web design is serving a single HTML document to all devices, but applying different style sheets based on the screen size in order to provide the most optimized layout for that device. For example, when the page is viewed on a smartphone, it appears in one column with large links for easy tapping. But when that same page is viewed on a large desktop browser, the content rearranges into multiple columns with traditional navigation elements. It's like *magic*! (Except that it's actually just CSS.)

The web design community has been a-buzz about responsive design since Ethan Marcotte first wrote about it and coined the phrase in his article "Responsive Web Design" on A List Apart in 2010 (*www.alistapart.com/articles/responsive-web-design/*). It's become one of the primary tools we use to cope with unknown viewport size.

Figure 3-3 shows some examples of responsive sites at the typical dimensions for a desktop monitor, tablet, and smartphone. You can see many more inspirational examples at the Media Queries gallery site (*mediaqueri.es*) Try opening a design in your browser and then resizing the window very narrow and very wide, and watch as the layout changes based on the window size. *Très* cool.

Open Medical Device Research Library
www.omdrl.org

Smashing Magazine
smashingmagazine.org

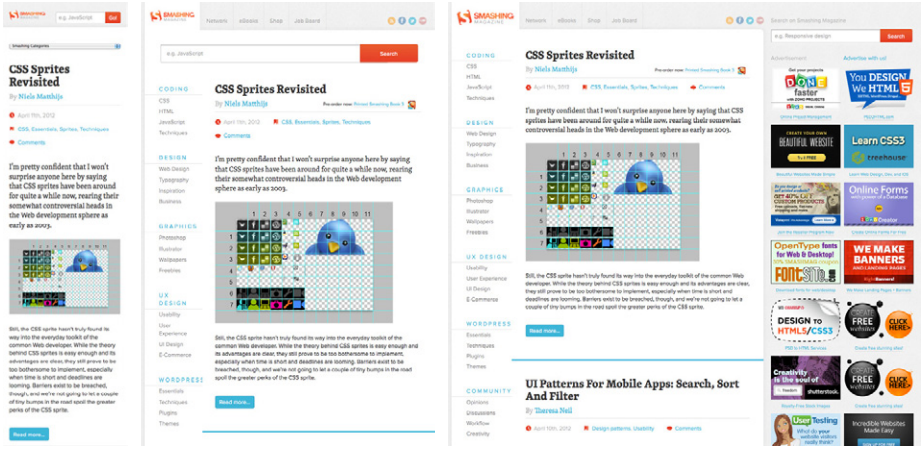

Figure 3-3. Responsive sites' layout changes based on the size of the browser window.

Responsive web design helps with matters of layout, but it is not a solution to all mobile web design challenges. The fact is that providing the best experiences for your users and their chosen device may require optimizations that go beyond adjusting the look and feel. Some problems are better addressed by using the server to detect the device and its capabilities and then make decisions on what to send back. Using progressive enhancement, you can deliver a baseline experience for the most basic browsers and devices, but send enhanced options for devices that can use them.

For some sites and services, it may be preferable to build a separate mobile site (see the "Dedicated Mobile Sites" sidebar) with a customized interface and feature set that takes advantage of phone capabilities like geolocation. That said, although responsive design won't fix everything, it is an important part of the solution for delivering satisfactory experiences on a wide variety of browsers.

Dedicated Mobile Sites

The alternative to a single responsive site is to build an entirely separate site, with a unique URL, that gets served up when requested by a mobile device. Mobile site URLs are commonly prefixed with *m.* or *mobile.* For some types of sites, a dedicated mobile site is the best solution if you know that your mobile users have very different usage patterns than folks seated at a desk. On dedicated mobile sites, the most frequently requested features are highlighted on the first screen, and a lot of the "extra" stuff (like promotions) from the desktop site is simply stripped away. (It makes you wonder what value it adds to the desktop site after all.)

Figure 3-4 compares Walgreens' primary and mobile sites as they appeared mid-2012. You can see that phone users are offered a much more streamlined set of options.

A dedicated mobile site may be the best way to make complex tasks easier for users on smartphones. Luke Wroblewski provides many thoughtful reasons why his service Bagcheck chose a separate site in his article "Why Separate Mobile and Desktop Pages?" (*www.lukew.com/ff/entry.asp?1390*). I recommend you give it a read.

The point here is that responsive web design is not a universal solution. For sites that feature mainly text content, a little layout adjustment may be all that is needed to bring a good reading experience on all devices. For other sites and web applications, a very different experience may be preferred.

The downside of a dedicated mobile site is that it is more than twice the work. It requires additional content planning, design templates, production time, and ongoing maintenance. But if it means giving your visitors the functionality they really need, it is well worth the investment.

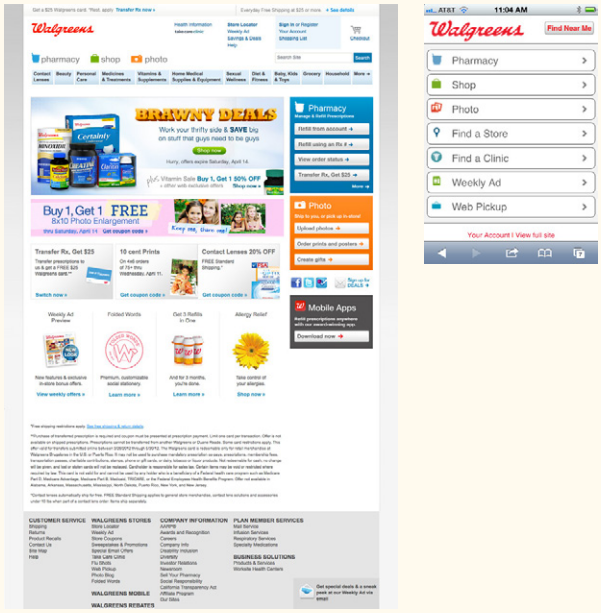

Figure 3-4. *A comparison of primary and dedicated mobile sites.*

NOTE

Even dedicated mobile sites can and should take advantage of responsive techniques to customize their experience from device to device. It isn't necessarily an only one or the other decision. Stephanie Rieger summarizes this point well in her article "Responsiveness is a characteristic," which you can read at stephanierieger.com/responsiveness-is-a-characteristic/.

For further reading

I'll cover responsive web design in more detail in Chapter 18, CSS Techniques, once you have more code experience under your belt. To continue your responsive design education, I recommend the following books.

- Ethan Marcott's book *Responsive Web Design* (A Book Apart) is required reading for budding web designers. It's a short book that is the perfect starting point for learning how responsive web design works and how to try it yourself.
- *Head First Mobile Web*, by Lyza Danger Gardner and Jason Grigsby (O'Reilly Media). This book includes responsive web design, but expands on it, including techniques that take advantage of scripting and server-side detection. It's also extremely entertaining to read, although you'll need some familiarity with CSS and JavaScript to get the most out of it.

One Web for All (Accessibility)

We've been talking about the daunting number of browsers in use today, but so far, we've only addressed visual browsers controlled with mouse pointers or fingertips. It is critical, however, to keep in mind that people access the Web in many different ways—with screen readers, braille output, magnifiers, joysticks, foot pedals, and so on. Web designers must build pages in a manner that creates as few barriers as possible to getting to information, regardless of the user's ability and the device used to access the Web. In other words, you must design for accessibility.

Although intended for users with disabilities such as poor vision or limited mobility, the techniques and strategies developed for accessibility also benefit other users with less-than-optimum browsing experiences, such as handheld devices, or traditional browsers over slow modem connections or with the images and JavaScript turned off. Accessible sites are also more effectively indexed by search engines such as Google. The extra effort in making your site accessible is well worth the effort.

There are four broad categories of disabilities that affect how people interact with their computers and the information on them:

Vision impairment. People with low or no vision may use an assistive device such as a screen reader, braille display, or a screen magnifier to get content from the screen. They may also simply use the browser's text zoom function to make the text large enough to read.

Mobility impairment. Users with limited or no use of their hands may use special devices such as modified mice and keyboards, foot pedals, or joysticks to navigate the Web and enter information.

Auditory impairment. Users with limited or no hearing will miss out on audio aspects of multimedia, so it is necessary to provide alternatives, such as transcripts for audio tracks or captions for video.

Cognitive impairment. Users with memory, reading comprehension, problem solving, and attention limitations benefit when sites are designed simply and clearly. These qualities are helpful to anyone using your site.

The W3C started the Web Accessibility Initiative (WAI) to address the need to make the Web usable for everyone. The WAI site (*www.w3.org/WAI*) is an excellent starting point for learning more about web accessibility. One of the documents produced by the WAI to help developers create accessible sites is the Web Content Accessibility Guidelines (WCAG and WCAG 2.0). You can read them all at *www.w3.org/WAI/intro/wcag.php*. The United States government used the Priority 1 points of the WCAG as the basis for its Section 508 accessibility guidelines (see the sidebar "Government Accessibility Requirements: Section 508"). All sites benefit from these guidelines, but if you are designing a government site, adherence is a requirement.

Government Accessibility Requirements: Section 508

If you create a site receiving federal funding, you are required by law to comply with the Section 508 Guidelines that ensure that electronic information and technology is available to people with disabilities. State and other publicly funded sites may also be required to comply.

The following guidelines, excerpted from the Section 508 Standards at *www.section508.gov*, provide a good checklist for basic accessibility for all websites.

1. A text equivalent for non-text elements shall be provided (e.g., via the "alt" attribute or in element content).
2. Equivalent alternatives for any multimedia presentation shall be synchronized with the presentation.
3. Web pages shall be designed so that all information conveyed with color is also available without color, for example from context or markup.
4. Documents shall be organized so they are readable without requiring an associated style sheet.
5. Row and column headers shall be identified for data tables.
6. Markup shall be used to associate data cells and header cells for data tables that have two or more logical levels of row or column headers.
7. Pages shall be designed to avoid causing the screen to flicker with a frequency greater than 2 Hz and lower than 55 Hz.
8. When pages utilize scripting languages to display content, or to create interface elements, the information provided by the script shall be identified with functional text that can be read by assistive technology.
9. When a web page requires that an applet, plug-in, or other application be present on the client system to interpret page content, the page must provide a link to a plug-in or applet that complies with §1194.21(a) through (l).
10. When electronic forms are designed to be completed online, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.
11. A method shall be provided that permits users to skip repetitive navigation links.
12. When a timed response is required, the user shall be alerted and given sufficient time to indicate more time is required.

Another W3C effort is the WAI-ARIA (Accessible Rich Internet Applications) spec, which addresses the accessibility of web applications that include dynamically generated content, scripting, and advanced interface elements that are particularly confounding to assistive devices. The ARIA Recommendation defines a number of roles for content and widgets that authors can explicitly apply using the `role` attribute. Roles include things like menubar, progressbar, slider, timer, tooltip, and so on, and add an enhanced layer of semantics for those who need it. For the complete list of roles, go to *www.w3.org/TR/wai-aria/roles#role_definitions*.

For further reading

The following resources are good starting points for further exploration on web accessibility:

- The Web Accessibility Initiative (WAI), *www.w3.org/WAI*
- WebAIM: Web Accessibility in Mind, *www.webaim.org*
- *Pro HTML5 Accessibility*, by Joshue O Connor (Professional Apress, 2012)
- *Universal Design for Web Applications: Web Applications that Reach Everyone*, by Wendy Chisholm and Matt May (O'Reilly, 2008)

The Need for Speed (Site Performance)

Although the number of users accessing the Internet on slow dial-up connections is shrinking (5–10% in the US as of this writing), the percentage of folks using mobile phones to access the Web is increasing dramatically and is eventually slated to exceed desktop usage. If you have a smartphone, then you know how frustrating it is to wait for a web page to fully display over a cellular data connection.

But site performance is critical regardless of how your users are accessing your site. A study by Google in 2009* showed that the addition of just 100 to 400 milliseconds to their search results page resulted in reduced searches (–0.2 to –0.6%). Amazon.com showed that reducing page load times by just 100ms resulted in a 1% increase in revenue.† Other studies show that users expect a site to load in under two seconds, and nearly a third of your audience will leave your site for another if it doesn't. Furthermore, those people aren't likely to come back. Google has added site speed to its search algorithm, so if your site is a slow poke, it's not likely to show up in that coveted first screen of results. The takeaway here is site performance (down to the millisecond!) matters a lot.

NOTE

See the article "Effect of Website Speed on Users, Statistics Reveal Slow Loading Times Cost Sites Serious Money" (munchweb.com/effect-of-website-speed) for more fascinating site performance studies.

There are many things you can do to improve the performance of your site, and they fall under the two broad categories of limiting file sizes and reducing the number of requests to the server. The following list only scratches the surface for site optimization, but it gives you a general idea of what can be done.

- Optimizing images so they are the smallest file size possible without sacrificing quality. You'll learn image optimization techniques in Chapter 22, Lean and Mean Graphics.
- Minimize HTML and CSS documents by removing extra character spaces and line returns.
- Keep JavaScript to a minimum.
- Add scripts in a way that they load in parallel with other page assets and don't block rendering.
- Don't load unnecessary assets (such as images, scripts, or JavaScript libraries).
- Reduce the number of times the browser makes requests of the server (known as HTTP requests).

Every trip to the server in the form of an HTTP request takes a few milliseconds, and those milliseconds can really add up. All those little Twitter

* "Speed Matters," *googleresearch.blogspot.com/2009/06/speed-matters.html*

† Statistic from "Make Data Matter," PowerPoint presentation by Greg Linden of Stanford University (2006)

widgets, Facebook Like buttons, and advertisements can make dozens of server requests each. You may be surprised to see how many server requests even a simple site makes.

If you'd like to see for yourself, you can use the developer tool in the Chrome browser to see each request to the server and how many milliseconds it takes. Here's how you do it:

1. Launch the Chrome browser and go to any web page.
2. Go to the View menu and select Developer → Developer Tools. A panel will open at the bottom of the browser.
3. Select the Network tab in the tools view and reload the page. The chart (commonly referred to as a waterfall chart) shows you all the requests made and assets downloaded. The columns on the right show the amount of time each request took in milliseconds. At the bottom of the chart, you can see a summary of the number of requests made and the total amount of data transferred.

Figure 3-5 shows a portion of the performance waterfall chart for my site, *Jenville.com*, which is a simple site (but not as simple as I thought!). You can poke around any site on the Web this way. It can be very educational.

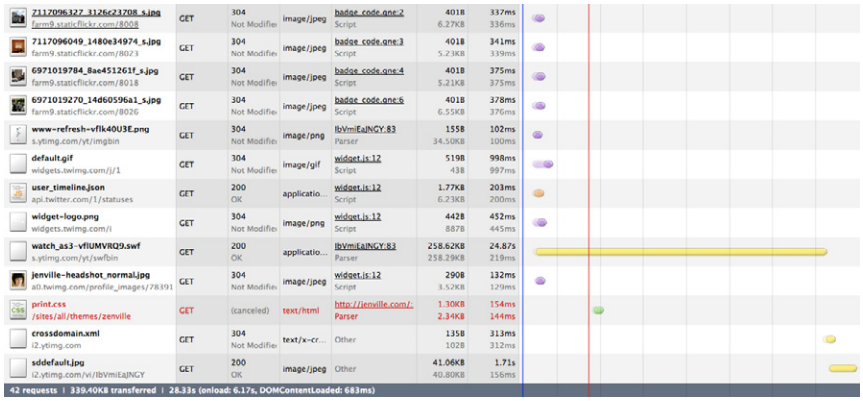

Figure 3-5. *Waterfall charts such as this one created by the Chrome Network developer tool show you the individual server requests made by a web page and the amount of time each request takes.*

I won't address site performance much in this book, but I do want you to keep the importance of keeping file sizes as small as possible and eliminating unnecessary server requests in the back of your mind in your web design work.

For further reading

There are other techniques that are too technical for this book (and frankly, for me), and I figure if you are reading this book, you are probably not quite ready to become a site performance wizard. But when you are ready to take it on, here are some resources that should help:

- Google's site *Make the Web Faster* (*code.google.com/speed/*) is an excellent first stop for learning about site optimization. It compiles a number of excellent tutorials and articles as well as tools for measuring site speed.
- The books *High Performance Web Sites* and *Even Faster Web Sites* (both by Steve Souders and published by O'Reilly Media) provide many best practices for speeding up sites. A good understanding of JavaScript and server functionality is required.

Test Yourself

1. The "mobile web" complicates our jobs as web designers. List at least three unknown factors you need to consider when designing and developing a site.

2. Match the technology or practice on the left with the problem it best addresses.

1. _____ Progressive enhancement	a. Assistive reading and input devices
2. _____ Server-side detection	b. Slow connection speeds
3. _____ Responsive design	c. All levels of browser capabilities
4. _____ WAI-ARIA	d. Determining which device is being used
5. _____ Site performance optimization	e. A variety of screen sizes

3. Web accessibility strategies take into account four broad categories of disabilities. Name at least three, and provide a measure you might take to ensure content is accessible for each.

More Site Performance Tools

Try some of these tools for testing site performance:

- WebPagetest (*webpagetest.org*) is a tool that was originally developed for AOL, but is now available for all to use for free under an open source license. Just type in a URL and WebPagetest returns a waterfall diagram, screenshot, and other statistics.
- Yahoo!'s freely available YSlow tool (*yslow.org*) analyzes a site according to 23 rules of web performance, then gives the site a grade and suggestions for improvement.
- For mobile sites, try Mobitest by Blaze (*www.blaze.io/mobile/*), a free tool for testing website performance on various mobile devices.
- There are also a number of slow connection speed simulators so you can get a feel for your users' experiences over less than ideal network speeds. Sloppy (*www.dallaway.com/sloppy*) is a web tool where you enter a web address and select a modem speed (and wait and wait). Mac OS users can try Slowy (*slowyapp.com*).

4. When would you use a waterfall chart?

5. Responsive web design doesn't solve everything. Describe what it is good for and where it falls short.