HTML

MANUAL OF STYLE

A Clear, Concise Reference for **Hypertext Markup Language** (Including **HTML5**)

Fourth Edition

- Get up to speed fast building dynamic Web pages, Wiki articles, blog posts, and more
- Learn through dozens of HTML5, CSS3, and JavaScript examples

Larry Aronson

HTML
Manual of Style

This page intentionally left blank

HTML Manual of Style

A Clear, Concise Reference for Hypertext Markup Language (Including HTML5)

Fourth Edition

Larry Aronson

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Cape Town • Sydney • Tokyo • Singapore • Mexico City

Library of Congress Cataloging-in-Publication Data:

Aronson, Larry.
HTML manual of style : a clear, concise reference for hypertext markup language (including HTML5) / Larry Aronson.
p. cm.
ISBN 978-0-321-71208-0 (pbk. : alk. paper)
1. Hypertext systems. 2. HTML (Document markup language) I. Title.
QA76.76.H94A7624 2010
006.7'4--dc22

2010031013

ISBN-13: 978-0-321-71208-0
ISBN-10: 0-321-71208-0
Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, IN.

Editor-in-Chief
Mark Taub

Acquisitions Editor
Trina MacDonald

Development Editor
Songlin Qiu

Managing Editor
Kristy Hart

Project Editor
Anne Goebel

Copy Editor
Gayle Johnson

Indexer
Erika Millen

Proofreader
Apostrophe Editing Services

Technical Reviewers
Elliotte Rusty Harold
Benjamin Schupak

Publishing Coordinator
Olivia Basegio

Interior Designer
Bumpy Design

Cover Designer
Anne Jones

This book is dedicated to my wife, Heidi Cohen,
a much better writer than I am, who provided encouragement
and inspiration throughout the project.

It was also written in fond memory of people who were
major forces in my life, especially my parents,
Manny and Dorothy Aronson, and my
friends Andy Cohen, Bobo Lewis, Milan Stitt, and
Lynne Thigpen.

Contents at a Glance

Table of Contents

Preface

What This Book Is About and Why You'll Find It Useful

This book is about using HTML to put your stuff on the Web. HTML (HyperText Markup Language) is the language that tells a web browser what to do with the text, images, and other media—the stuff—you want others to see. There are good ways to use these tools, and there are bad ways. Web browsers are smart application programs. They can take badly written HTML and still present a respectable-looking web page. However, there are still very good reasons for learning how to write good markup. This book is about creating web pages that

- Are pleasing to look at and fun to play with
- Are friendly to search engine robots
- Are easy to update and maintain over time

The Web can be understood through a number of metaphors that allow us to think of a website as a place within a realm we explore. We even socialize within its "spaces." But that is just a useful illusion. Under the hood, the Web is not like that at all. Chapter 1 introduces the client/server technology that web authors and developers use to create the illusion. Even if you consider yourself an experienced web user, Chapter 1 is worth skimming.

Chapter 2 is all about the elements of HTML, including some of the more interesting HTML5 additions. It has many examples illustrating how to mark up documents semantically so that the resulting web page provides all the right information to readers, both human and robot, and that it is easy to update.

Our first obligation in design is to please ourselves. With good document structure, a website can be easily styled in a consistent manner across all pages. Chapter 3 explains, with many examples, how to use Cascading Style Sheet (CSS) statements to apply styling to document elements and create people-pleasing web pages.

Chapter 4 is about using HTML as a contributor to other websites that accept marked-up content. Five examples are given: blogging, Google Docs, eBay selling, Wikipedia editing, and HTML email.

Despite the many options for putting content online, sometimes your organization's objectives or your personal goals dictate building and running your own website. Chapter 5 explores many of the issues involved, including website structure, organization and navigation, and search engine optimization.

At the end of this book you'll find quick-reference guides to HTML elements and CSS properties, including the new elements and properties in the HTML5 and CSS3 draft specifications. There is no list of references to resources. The W3C's website at w3.org and Wikipedia's articles on HTML and CSS should cover anything from a technical perspective. You know how to search the Web for other guides, tutorials, and examples.

Finally, this book is about you, because you are changing from a person who uses the Web for information and services to one who contributes to the Web. People are discovering that the joys of online shopping pale in comparison to the pleasures of creative collaboration. There is a place on this new Web for your stuff, and this book is about how to create content with style. I hope you will find it useful.

What's Not Covered in This Book and Why Not

This book is not intended to be a complete reference guide to HTML5. Such a book would be at least three times larger than this one and would be out of date shortly after publication. Web technologies are changing fast. The information in this book is based on the World Wide Web Consortium's (W3C) draft recommendation for a proposed HTML5 standard. Although that might sound a bit tenuous, much of the draft specification has already been adopted by our favorite browsers (even though certainly much will change by the time the standard is officially approved). That being the case, I don't claim to be an authority on HTML5, only an author of a book on HTML5.

Along with HTML and CSS, JavaScript plays a part in some of this book's examples. Teaching JavaScript is way beyond the scope of this book, but it is included for two reasons. First, the HTML5 specification formalizes the behavior of document elements in response to user actions using JavaScript syntax and methods. Second, JavaScript libraries, such as jQuery, provide rich new vocabularies of element behaviors that previously were unavailable to web authors and developers.

Other technologies play an important part in the operation of some websites, but they are not really discussed in this book. If you want to learn about using Microsoft's Silverlight technology or Adobe's Flash platform to develop web pages, you've reading the wrong book. As a freelance developer, I tend

to favor tools that are free and community-supported. It is not that I think the tools I use are superior to these other technologies; I just have never used them, and I don't have a basis of comparison.

How This Book Happened and What's to Come

The World Wide Web was born more than 20 years ago on the border of Switzerland. When I first became aware of the Web, I was working as a consultant for one of those large Wall Street investment firms that no longer exist. It was the fall of 1993, and I was converting a mainframe-based system for modeling mortgage-backed derivatives to run on a minicomputer. I was in the office of the network administrator, whom I had become friends with, and he was showing off a cool application he had recently downloaded from the University of Illinois' FTP site. It was called Mosaic. My life was about to change, and I was ready for it.

At the time, I was already into the world of hypertext applications as an avid fan and user of Apple Computer's HyperCard application. I had created a number of "stacks" (which is what HyperCard programs were called) for myself and others. I kept up with the field by participating in the Usenet newsgroup, alt.hypertext, and local discussion groups on Panix, an early Internet service provider (ISP) based in New York City.

I immediately saw the potential of Mosaic and the Web in its seamless integration of anonymous File Transfer Protocol (FTP) and hypertext navigation. Prior to the availability of Mosaic, to read a particular document, first you had to know where that document was on the Internet. Then you logged in to that FTP server, downloaded the file, and opened it for reading. Not only did Mosaic automate these intermediate steps, it also helped you find the next document you were interested in.

It was an exciting time. Dozens of new websites were appearing every week. Updated versions of the web browsers available then—Mosaic, MacWeb, WinWeb, and Arena—were released frequently, supporting more HTML markup elements and new authoring abilities, such as centered text and inline images. Every day, new techniques were discovered and shared in newsgroup discussions and at usergroup meetings.

In the early summer of 1994, shortly after the U.S. government allowed the Internet to go commercial, I was contacted by another frequent newsgroup contributor, Clay Shirky. He asked if I would be interested in meeting his publisher, who was looking for a knowledgeable author to write a book about HTML. Clay had other commitments, so I became the author of the first book

on Web publishing. Clay is an excellent writer. His books *Voices from the Net*, *Here Comes Everybody*, and *Cognitive Surplus* are must-reads for anyone looking to explore the cultural impact of new technologies.

The first edition of this book came out at the end of 1994, and a second edition, *HTML3 Manual of Style*, was published a year later.[1] Fast-forward 14 years to the fall of 2009, and I'm attending meetups and blogging about HTML5. Another author of programming books, Elliotte Rusty Harold, emails me, wondering if I would be interested in talking to his publisher about redoing *HTML Manual of Style*. Talks led to a formal book proposal and a contract, and now I'm an author again.

In reviewing the second edition, I came across this paragraph in the Preface:

> This is a book in the middle. The first edition was written just before HTML2 was finalized. Today, HTML is in the middle of the transition to level 3. The Web itself is moving from an academic to a commercial focus, and yours truly is in the middle of a career change from programmer/analyst to author/lecturer. Some of the topics covered herein are illustrated using products that were still in beta testing, which means that my best guess today may not accurately describe where the Web will be tomorrow. This book will get you started in Web publishing; the rest of your education will come online.

The sense of that paragraph is true again today. The Web is undergoing a major technological upgrade as it expands from its commercial focus to encompass and shape our social activities. Support for the emerging HTML5 and CSS3 specifications by the principal browser makers are making it possible for Web authors and developers to create exciting new websites and applications. It is safe to say that the Web will change over the next couple of years more than it has in the last decade. That excitement is also what this book is about.

1. A third edition, *HTML3.2 Manual of Style*, was published in 1997 without my participation after ZD Press was acquired by a larger publishing company. So technically, this book is the fourth edition of *HTML Manual of Style*.

HTML and the Web

Chapter 1

HTML is the framework of the Web. This chapter describes how the Web works and provides a bit of Web history for context. You will learn about the client/server architecture of the Web and how it is hyperlinked. I'll present the Web Bestiary of acronyms and definitions and discuss the philosophy and implications of HTML5.

Although this chapter is about the Web and HTML, it actually contains very little HTML. If you want to get right into learning the HTML language, skip this chapter and go to the next. You can come back here later to help consolidate what you have learned.

HTML: The Language of the Web

HyperText Markup Language (HTML) is the language of the Web. If you could listen to the conversation between your computer and the websites you visit, you would hear HTML spoken. **Web servers** accept requests from your browser as you visit and interact with the sites they host. In reply, the servers return marked-up content that your browser formats into the web page you see. Web servers also send requests to each other, gathering and exchanging data that power search engines and make a rich variety of social and commercial transactions possible.

HTML is not a programming language like C, Perl, or Ruby. HTML is a semantic language for marking up text. The markup provides a description of the content that Web browsers use to construct the corresponding web page.

Links are defined in HTML. This ability to have active references in a document to other documents, no matter where they are physically located, is very powerful. All of the Web's resources are addressable using a Uniform Resource Locator (URL). Any information can be easily located and linked with related content, creating frictionless connectivity.

The Web hosts many protocols and practices, but HTML is the foundation, providing the basic language to mark up text content into a structured document by describing the roles and attributes of its various elements. A companion technology, **Cascading Style Sheets (CSS)**, lets you select document elements and apply styling rules for presentation. CSS rules can be mixed into the HTML code or can reside in external files that can be employed across an entire website. This keeps content creators and site designers from stepping all over each other's work. HTML describes the page's content elements, and CSS tells the browser how they should look (or sound.) The browser can override the CSS instructions or ignore them.

Example 1.1 creates a very simple web page. You can copy this HTML code into a plain text file on your computer and open it in any browser. Give it a filename ending in the extension .html.

Example 1.1: HTML for a very simple web page

```
<!DOCTYPE html>
<html>
<head>
  <title>Example 1.1</title>
  <style type="text/css">
    h1 { text-align: center; }
  </style>
</head>

<body>
  <h1>Hello World Wide Web</h1>
  <p>
    Welcome to the first of many webpages.
    I promise they will get more interesting than this.
  </p>
</body>
</html>
```

The code in Example 1.1 (shown in boldface) consists of two parts: a document *body* containing the page's content, preceded by a head section that contains information about the document. In this example, the head section contains the document's title and a CSS style rule to center the page's heading. The body consists of a level 1 heading followed by a paragraph. The result should look something like Figure 1.1.

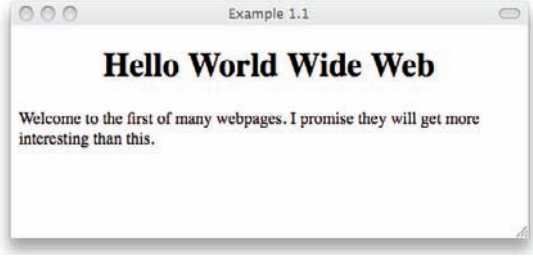

Figure 1.1: A simple web page

This brings up a fundamental principle about how the Web works: Web authors should not make assumptions about their readers, the characteristics of their display devices, or their formatting preferences. This is especially important with mobile Web users and people with visual disabilities. A Web author or developer shouldn't even assume that a site visitor is human! Websites are constantly visited by automated programs that gather and catalog information about the Web. The general term user agent is used to describe any software application or program that can talk to a web server. A modern website regards visits from all user agents with the same importance as human visitors using Web browsers. The best approach is to keep the HTML simple so that it provides a semantic description of the various content elements and leaves the presentation details to the reader.

The other major player on the Web programming team is **JavaScript**, a programming language that runs inside a browser and manipulates HTML page elements in response to user actions and other events. There are other scripting languages besides JavaScript, but it is the most popular. Also, JavaScript syntax and terms are used in the HTML5 specification. Like CSS, JavaScript code can be embedded within the HTML source code of a web page or can be imported from a separate file. User agents other than browsers generally ignore JavaScript and other embedded executable code. It can be dangerous for robots.

Robots?!

Robots are a very important class of Web user. They are automated computer programs that run on Internet servers and visit web pages the same way people do using a browser. But instead of presenting the page, the robot analyzes it, stores information about the page in a database, and decides what page to visit next using that information. This is how Google, Yahoo!, Bing, and other search engines work. Other robots perform similar data collection for marketing and academic purposes. Robots are often called "spiders" because of how they seem to "crawl" over the Web from one link to the next. Also, there are malicious robots. These automatic programs leave spam comments on blogs or look for security loopholes to gain control of resources with which they should not be messing. *Bad robots!*

When creating content for the Web, you generally are not concerned with any of this. Most of the HTML structure that deals with browsers, robots, and widgets is supplied by the Web editing software you use or by server-side scripts and template systems. If you are editing content directly online, all you need to understand is how to mark up the content with simple HTML elements. Web developers—that is, programmers as opposed to authors—need to fully understand how these three principal components—HTML, CSS, and scripting—work together to form the framework of the Web (see Figure 1.2).

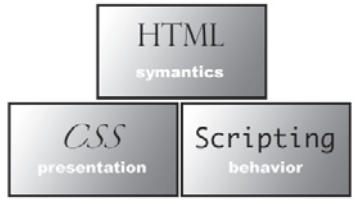

Figure 1.2: The three components of a web page

By the way, did I mention that all of this is essentially *free*? It is free in two senses of the word. It's free because there is no acquisition cost, and free because you can use it for your own purposes. With only minor limitations, all the HTML, CSS, and scripting that go into a Web page are available for you to examine, copy, and reuse. **Tim Berners-Lee**, the inventor of HTML, the URL, and the HTTP protocol that web servers and user agents use to talk to each other, put all these components into the public domain. Working at CERN, the European Center for Nuclear Research, he was trying to find a better way for large teams of researchers, working in different countries with different word

processors, to quickly publish research papers. Patent rights and Nobel Prizes were at stake. In a post to the alt.hypertext newsgroup on August 6, 1991, which was effectively the Web's birth announcement, Berners-Lee wrote:

> The WWW project was started to allow high energy physicists to share data, news, and documentation. We are very interested in spreading the web to other areas, and having gateway servers for other data. Collaborators welcome!

Twenty years later, Berners-Lee is still very much involved in the evolution of the Web as head of the **World Wide Web Consortium (W3C)**. I stress "evolution" here to point out that, while the Web has transformed society, freeing us to work and play in a global sea of information, a lot of that happened by accident. HTML is still a work in progress.

A Bit of Web History

The early Web was text only—without images or colors—and browsers worked in line mode. In other words, you cursor-keyed your way through page links sequentially, like browsing on a low-end cell phone. It was not until 1993 that a graphical browser called **Mosaic** was made available from the University of Illinois National Center for Supercomputing Applications (NCSA) in Champaign-Urbana, Illinois. Mosaic was easy enough to install and use on Windows, Macintosh, and UNIX computers.

Mosaic was written by a group of graduate students—principally, Marc Andreessen and Eric Bina. They built Mosaic because they were excited by the possibilities of hypertext and were dissatisfied by the browsers available at the time. They were supposed to be working on their master's projects.

Mosaic was the progenitor of all modern browsers. It displayed inline images, multiple font families, weights, and styles, and it supported a pointing device (a mouse). Distribution of the technology and Mosaic trademarks was managed for the NCSA by the Spyglass Corporation and was licensed by Microsoft, which rewrote the source code and called it **Internet Explorer**.

After graduating from the University of Illinois, Andreessen teamed up with Dr. Jim Clark to form Netscape Corporation. Dr. Clark was the former CEO of Silicon Graphics, Inc., whose sexy, powerful graphics computers/workstations revolutionized Hollywood moviemaking. The **Netscape Navigator** browser introduced major innovations and became extremely popular because Netscape Corp. did something quite astounding for the software industry at

the time—it gave away Navigator! At its peak, Netscape had captured close to 90% of the browser market.

In 1994, something wonderful happened. Vice President **Al Gore**, as chairman of the Clinton administration's Reinventing Government program, arranged for the National Science Foundation (NSF) to sell the Internet to a consortium of telecommunications companies. This ended the NSF's strict "no commercial use" policy and gave birth to the dotcom era and jokes about Al Gore inventing the Internet. In mid-1994 there were 2,738 websites. By the end of that year there were more than 10,000.[1]

From the beginning, competition to commercialize the Internet was fierce. In the mid-1990s, the tech community was abuzz about the "browser wars" as browser makers threw dozens of extra features into their software, adding many new elements to HTML that appealed to their respective markets. Netscape added features that appealed to graphic designers, including support for jpeg images, page background colors, and a controversial FONT tag that allowed Web designers to specify text sizes and colors. Microsoft bundled Internet Explorer into its Windows operating system and tied Web publishing into its Microsoft Office product line. These moves resulted in considerable legal troubles for Microsoft. These problems lasted until 2001, when the U.S. government suddenly dropped its antimonopoly suit against the corporation in the first days of George W. Bush's presidency.

Other companies introduced browsers with interesting ideas but never captured any significant market share from Netscape and Microsoft. Arena, an HTML3 test bed browser written by Dave Raggett of Hewlett-Packard (HP), introduced support for tables, text flow around images, and inline mathematical expressions. Sun Microsystems came out with a browser named **HotJava** that generated a lot of interest. It was written in **Java**, a programming language that Sun developed originally for the purpose of controlling TV set-top boxes. Sun repurposed the language for the Internet with the dream of turning the browser into a platform for small, interactive applications called applets that would run in a virtual Java machine in your PC. Sun put Java into the public domain to encourage its adoption. This allowed Microsoft to make and market its own version of the language. Microsoft's Java was sufficiently different from Sun's version to make using applets (not to mention writing them) difficult. Although the Java language eventually gained widespread use in building in-house corporate applications, HotJava died along with Sun's Internet dreams.

1. Wikipedia: http://en.wikipedia.org/wiki/List_of_websites_founded_before_1995

On a related note, a company called **WebTV Networks** produced a low-cost Internet appliance and service for consumers to browse the Web and do email on their TV sets using a wireless keyboard and remote control. Despite funding difficulties and an on-again/off-again relationship with Sony Corporation that almost killed the project, WebTV succeeded in bringing the Web and email to nearly a million customers seeking to avoid the cost and complexity of personal computer ownership.

To illustrate how weird Web-related events can get, according to Wikipedia, WebTV was for a brief time classified as a military weapon by the U.S. government and was banned from export because it used strong encryption. In 1997, Microsoft bought WebTV and rebranded it as MSN TV to expand its Web offering. Without marketing the service or servicing its customers, MSN TV died a few years later. But the WebTV technology survived, eventually resurfacing in Microsoft's **Xbox** gaming console.

One of my favorite Web browsers was **Virtual Places**, created by an Israeli company, Ubique. Virtual Places combined Web browsing with Internet chat software and enabled collaborative Web surfing. It turned any web page into a virtual chat room where you and other visitors were represented by avatars—small personal icons that you could move around the page. Whatever you typed in a floating window would appear in a cartoon balloon over your avatar's head. It had a "tour bus" feature that allowed a teacher, for example, to take a group of students to websites around the world and back.

Unfortunately, the server overhead in keeping open connections and tracking avatar positions kept Virtual Places from expanding as the number of websites exploded. At the time, Netscape was updating Navigator every few weeks. Because Ubique couldn't keep up, nobody used Virtual Places as their default Web browser. AOL bought Ubique for no apparent reason and sold it to IBM a few years later. IBM used some of the technology in its software for corporate communications and collaboration. Virtual Places died during the **dotcom crash** at the start of the twenty-first century, but the avatars survived.

While Java was hot, Netscape developed JavaScript, a scripting language that ran in the Netscape Navigator browser and allowed Web developers to add dynamic behaviors to the HTML elements of a web page. Despite having the same first four letters, JavaScript and the Java programming language are quite different. It is suspected that Netscape changed the name from LiveScript just because of the buzz around Java. Superficially, the code looks similar because both are object-oriented programming (OOP) systems and have similar syntax.

America Online (AOL) acquired Netscape in 1998, and the browser's source code was made public. Eventually, this became the foundation on which the Mozilla organization built the Firefox browser. Other companies followed suit, and over the ensuing years, a variety of graphical browsers based on Netscape came to market. Microsoft's **Internet Explorer (IE)** browser improved with each new version and eventually became the most popular browser due to its bundling with the Windows operating system.

The browser wars ended with the dotcom crash, and manufacturers began to bring their browsers into compliance with emerging standards. Under the W3C's guidance, HTML language development slowed and stabilized on an HTML4 specification. The use of CSS was promoted to give Web developers finer control over typography and page layout over a much wider selection of devices. HTML attributes and actions (more about these later) were generalized. The HTML syntax was modified slightly to conform to **XML (eXtensible Markup Language)**, and a transition path was provided to the merging of the two in the **XHTML** specification.

The way HTML source code looks has changed. Currently, most websites are written to the HTML4 and/or XHTML standards, in which valid markup element and attribute names are written using lowercase letters. By contrast, a web page written to the HTML3 standard is filled with names written in all uppercase letters. This convention emerged from early website developers, who had to write HTML without the benefit of text editors that provided color syntax highlighting. Using uppercase names provided contrast that distinguished the markup from the content.

More importantly, the ways in which content creators, software developers, and people in general use the Web has evolved dramatically. This change is encapsulated in the term **Web 2.0**. Although this suggests a new version of the World Wide Web, it does not refer to any new technical specifications. Instead, it refers to the changing nature of web pages. The features and functionality that characterize a Web 2.0 site are a matter of debate. Web 2.0 is better understood as simply a recognition that today's websites do new things with newer technology than yesterday's websites.

Many of these changes have come about due to the embrace of **open source** as a philosophy of design and development by the tech community. Much of the software that powers the Web is nonproprietary. It is freely available for people to use, copy, modify, and redistribute as they please. Open-source development has greatly reduced the cost of software development while increasing its availability, stability, and ease of use. Equally interesting is that

the Web is self-documenting. Information about what is on the Web, how it is organized, and how it can be used is everywhere on the Web.

Hypertext Content and Online Media

Content is everything. Online, it is HTML markup that tells your browser what that content means and how to present it to you. The concept of markup comes from traditional print publishing, in which a writer supplies the content, which an editor then marks up with instructions for the printer, specifying the layout and typography of the work. The printer, following the markup, typesets the pages and reproduces copies for distribution.

With the Web and HTML, the author and the editor are often the same person. The work, or content, lives in a linked set of HTML files on a web server. The content is not distributed in discrete copies, as in the print publication model. Instead, copies of web pages are served in response to user requests. The information returned by the web server is processed by the user's browser to display a web page in a window or tab.

Often the content of a web page does not reside in an HTML file but is generated dynamically by the web server from information stored in a database, using templates to produce **web pages**. It is common for web page to encompass resources from other servers. That is, a request a browser sends to a web server may result in that web server making requests of other servers. These distinctions, however, are immaterial to the user's browser. It just downloads whatever the web server provides without caring how that content was created or who marked it up.

The technological concepts are simple: an open exchange of data and information about that data (metadata), including content and markup. As a connected world of places to visit, the Web is more than a metaphor. The language of the Web, including verbs such as *surf*, *browse*, *visit*, *search*, *explore*, and *navigate*, and nouns such as *site*, *home page*, *destination*, *gateway*, and *forum*, creates a very real experience of being someplace.

Uniform Resource Locators (URLs)

How does a browser know what to request of a web server? How does your browser know which web server, of the millions in the world, to ask? The answer, as you've probably guessed, is links! A **link** is a reference, embedded in the content of a document, to another resource on the Web. This is the essence of hypertext media.

The destination of a link is given by a string of characters called a **Uniform Resource Locator (URL)**. A special bit of HTML markup, called the anchor element, makes this portion of text, or that image or those buttons, "active." When you click one, your browser requests a new document from the web server indentified in the URL.

In addition to links, URLs are used in HTML to load images, video, and other online media into a page; to apply stylesheets and create pop-up windows; and to specify where form input should be sent. In HTML a URL can be in partial form, often called a relative URL. A browser fills in any missing parts of the URL from the corresponding parts of the current page's URL to create a full URL. This neat trick makes it easy to relocate a website. A full URL starts with the protocol to use for the transfer. The URL design is universal and can reference other Internet things besides Web resources. We will go into more detail later. For now, suffice it to say that the Web's protocol is HyperText Transport Protocol, abbreviated as "http" or "https" when used in a URL. The "s" means that a secure (that is, encrypted) connection is made to the web server so that nobody eavesdropping on the conversation between your browser and the web server can steal anything important, such as a credit card number. Otherwise, the https protocol works the same way as http. By having secure transactions at the protocol level, web page authors and developers can write HTML that works in either environment.

The web server address comes after the protocol designation. Following that, the path to the file or resource is given. (There's more, but this will do for now.) Thus, when you click a link whose defining anchor element[2] contains a URL, such as http://www.google.com/about.html, your browser understands this as a request to open a connection to the Internet server, *www.google.com*, using the HTTP protocol and to get the resource, *about.html*.

Of course, you do not always have to click a link or button to get somewhere on the Web. You can just type a portion of a URL into the location window at the top of your browser, and you are taken there. Alternatively, you can open an HTML file from your local computer. (Web developers commonly do this when working on a website.)

Web Browsers and Servers

As intelligent as Web browsers currently are, web servers are smarter still. A single web server can host hundreds of different websites, manage many different types of content, read/write information from/to databases, and speak

2. `<a href="http://www.google.com/about.html">About Google</a>`

multiple languages, both human and artificial. A web server knows who you are (to be precise, it knows the Internet address of your computer and what browser is being used), it keeps track of each request you make, and it logs whether it was able to comply with the request.

The Web has a **client/server** architecture, as illustrated in Figure 1.3. Most Internet protocols are client/server, including File Transfer Protocol (FTP), email, and many online games. A web server is a computer that resides on a rack somewhere, or is tucked into a back closet, patiently waiting for a client program to send it a request it can fulfill. As far as the web server is concerned, anything that sends it a request is considered an important client. In Web-speak, the client programs are called user agents. Web browsers are the most important user agents. Robots, or "bots" as they are sometimes called, are another kind.

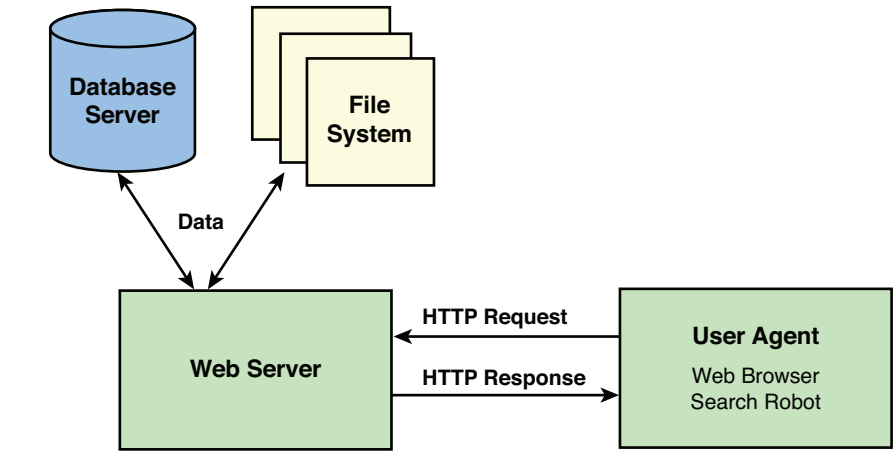

Figure 1.3: The Web's client/server architecture

Widgets can also be user agents. Loosely defined, a widget is a small computer program. It is packaged so that it can be easily installed as an extension of a larger computer program, such as a web browser or mobile device, and it runs in its user interface. A widget can, in response to a mouse click or other user action, send requests to web servers just like browsers and robots do. Unlike robots running on large servers, organizing large masses of information, a widget typically uses the returned information to update the content in a specific page element.

Widgets come in many varieties and are rarely harmful. They run within the browser's security setup and are generally isolated from your computer's file system. However, they can cause trouble if they are not well written. The problems include messing up the display of a web page, using up too much of the browser resources, or even causing a browser to crash.

Any stand-alone computer application or software program that exchanges information over the Web (Twitter clients, for example) is a user agent. So are the automatic software update programs that come with computer operating systems. So is the online Help feature of Microsoft Word or, for that matter, an Xbox, Nintendo, or PlayStation game console. Many of the apps on a modern smartphone are user agents, sending requests to web servers and using the returned information to do something useful or keep you informed.

Every web browser must provide three basic functions: 1) It must provide a control interface for human users; 2) it must exchange information with other computers; and 3) it must interpret HTML and render a web page. We are primarily interested in this last function—how HTML is understood by a browser and how that determines what is seen on the page. Many browser makers use the same open source, HTML rendering engines and differ mostly in their user interfaces. As a result, only four browser types cover most Web surfing: Internet Explorer, Mozilla (Firefox, Flock), Webkit (Safari, Chrome), and everything else (mobile phone browsers, legacy versions of IE, and Internet appliances).

As with browsers, several different web servers are in use today, hosting nearly a quarter billion websites in total. By far the most popular web server, according to a November 2009 survey by Netcraft, is **Apache**, an open-source product from the Apache Foundation. It hosts about half of all sites worldwide. The next most popular web server is the Internet Information Server (IIS) from Microsoft, with about one-third of the market. The remaining web servers are Google Web Server (GWS), which the company uses internally to host its massive search engine and user sites; nginx (pronounced "engine X"), a free, lightweight, high-performance server written by Igor Sysoev; and Qzone, a Chinese web server used by QQ.com to host upward of 20 million blogs under its domain.

When a web server receives a request from a user agent, all it has to do is figure out which file to return. Actually, it is a bit more complicated than that. Apache, for example, has a modular structure with "hooks" that allow a systems administrator to include custom components. Apache analyzes

the incoming request, applying defaults and rewriting rules. It determines whether to satisfy the request by returning the contents of a file or by executing a program and returning the output. If the requested resource requires authentication, Apache returns a status code instructing the browser to resubmit the request after prompting for a username/password combination. The HTTP request contains additional information such as the name of the browser or user agent and the preferred language. This enables Apache to provide a different page for mobile users or to substitute a translation of the requested page if one is available.

Web browsers and servers speak many other Internet protocols. Browsers are, in a sense, the Swiss army knives of Internet clients. Web servers have plug-in interfaces to email, database, FTP, streaming video players, and other services. Web servers can also make requests to each other and serve as mirrors or proxies for each other.

The Web Bestiary

This section contains a lot of acronyms and definitions. Much of the descriptive material is taken from Wikipedia. In a very real sense, Wikipedia represents the current usage and understanding of these terms by the Web community. I've listed them in order of decreasing importance, or their likelihood of ever coming up in casual conversation. This list is by no means complete.

- **HTML (HyperText Markup Language)** The predominant markup language for web pages. It provides a means to create structured documents using semantic tags for such things as headings, paragraphs, lists, links, quotes, and other items. It lets you embed images and other media objects and can be used to create interactive forms.
- **CSS (Cascading Style Sheets)** The language for describing the presentation (that is, the formatting and layout) of an HTML document. CSS is designed to enable the separation of document content from the details of how it should be presented, including the typography, positioning, colors, and margins. This separation improves content accessibility and provides more flexibility in controlling presentation characteristics.
- **JavaScript** An object-oriented scripting language. Although JavaScript has other uses, we are concerned here about client-side JavaScript—the version that runs inside a user's browser and manipulates HTML page

elements. JavaScript code can be embedded within the HTML elements of a web page or imported from a separate file. Not all web pages have JavaScript components, and users can turn off their browsers' JavaScript engine if they want to. Robots generally ignore JavaScript code as they examine web pages.

- **HTTP (HyperText Transport Protocol)** The set of rules governing how user agents, web browsers, and the like send requests to a web server and how the web server responds to the request. The web server returns a status code and data, or sometimes just the status code, when something goes wrong. The familiar 404 error code is returned when the web server cannot find what you are looking for. There are two primary HTTP request methods. A *Get* request is typically sent by your browser when you click a link with the intention of going to another web page. A *Post* request is typically sent when you click a form's submit button, essentially asking that the web server do something with your input.
- **CGI (Common Gateway Interface)** A protocol for dynamically generating web pages in response to a get request or form submission. The term is typically used as an adjective to indicate a server-side process, such as CGI script. CGI programs are typically written using a scripting language such as Perl, Ruby, C, vBasic, or Python. Many websites are entirely driven by CGI processes, although the relative number of such sites has probably been declining as newer technologies, such as AJAX and PHP, have become popular.
- **AJAX (Asynchronous JavaScript and XML)** The most recent versions of JavaScript and other client-side scripting languages contain features that a developer can use to create web pages that can make independent HTTP requests to the server while the page is loading or anytime thereafter. AJAX is the set of techniques used to create web pages with elements that can be independently updated with new content in response to a user's mouse click or some other event without having to reload the entire page. This is how many widgets work.
- **XML (eXtensible Markup Language)** A set of rules for marking up documents that emphasizes generality and global usability. It is widely used to transmit arbitrarily structured data in mixed client/server environments. XML and HTML are compatible members of a family of markup languages called Standard Generalized Markup Language (SGML). HTML is an SGML language with a specific Document Object

Model (DOM) focused on describing hypertext documents. The two technologies are combined in the XHTML specification.

- **JSON (JavaScript Object Notation)** Although based on JavaScript, JSON is a language-independent system for representing data objects. It is simpler than XML and is often used as an alternative to XML in AJAX applications to transfer data objects between a server and a script running in a user's browser.
- **CMS (Content Management System)** An application program or a package of software tools that facilitates the creation of web pages and automates their maintenance using a Web-based interface for authoring, editing, and administration. The term has broader use beyond the Web. For our purposes, it refers to any site or software that generates web pages from content stored in a database and provides a means of creating, editing, and managing that content without requiring knowledge of HTML, CSS, or FTP. A good CMS permits you to directly enter HTML with the content for finer control of web page presentation. **Blogs** are a form of content management system.
- **Flash (Adobe Flash, formerly Macromedia Flash)** A popular multimedia platform for adding animation and interactivity to web pages. Flash is commonly used to create animations, advertisements, and various interactive components, to integrate video into web pages and to develop rich Internet applications. Some websites are done entirely in Flash. However, this is now considered a poor practice, partly because the content of a Flash site is generally inaccessible to robots.
- **PHP (PHP Hypertext Preprocessor)** PHP originally stood for Personal Home Page. The PHP Group, the informal organization that currently oversees the development of the language, decided to expand the meaning of PHP a few years ago and gave us the current recursive acronym. PHP is a server-side technology for dynamically generating websites. It is powerful and easy to write but often difficult to read. A PHP file intermixes program logic—PHP statements enclosed in special tags—with HTML markup. When a request is sent to a web server for a file ending with the .php extension, the web server preprocesses the coded file, executes the PHP instructions, and returns an HTML document to the user's browser. Many modern Web applications, such as the popular blogging software WordPress, are written in PHP.

- **FTP (File Transfer Protocol)** An Internet protocol for transferring files from one computer to another, usually using a stand-alone application. Web browsers and page editors also use FTP to upload and download files. Dozens of FTP clients are available. One of the most popular is FileZilla, a free, open-source program that runs on Windows, Macintosh, and UNIX computers.
- **jQuery (JavaScript Query Language)** A library of JavaScript functions (often called a framework) that simplifies the development of dynamic, interactive web pages. It provides a language for selecting DOM elements and giving them complex behaviors. jQuery takes care of cross-browser differences in the DOM and facilitates the use of AJAX. In much the same way that CSS does with web page presentation, jQuery encourages the separation of semantic HTML markup from the descriptions of how HTML elements should respond to events. jQuery makes Web programming fun.
- **RSS (Real Simple Syndication)** An XML protocol for distributing content. Such distributed content from a website is called a *feed* and provides an alternative means for users to access the content. Users can subscribe to feeds using a number of stand-alone newsreaders or by using the feed-reading facilities incorporated into their browsers and email clients. Feeds from one website can also be embedded into web pages on another site in a syndicated publishing model. RSS is quite popular but evolved in an ad hoc way and is not a recognized standard. A newer feed protocol called Atom is more robust and follows the applicable standards.
- **DNS (Domain Name System)** A system for assigning names to computers connected to the Internet or a private network. It translates domain names meaningful to humans into the numerical addresses associated with networking equipment for the purpose of locating these devices worldwide. The Domain Name System can be thought of as the "phone book" for the Internet.
- **DOM (Document Object Model)** A dictionary and grammar for interpreting HTML. A DOM describes HTML elements and their attributes and properties and how they are used to create web pages. DOMs are published in a form that can be read by both humans and machines. Every web browser has at least one DOM, and most modern browsers conform to DOMs published by the W3C. Yet there are still

some differences in browser behavior arising from coding bugs, DOM misinterpretation, and edge conditions where browser behavior is not fully defined.

In this book, whenever you encounter the term DOM, it means the *W3C's draft specification for HTML5* as interpreted by your favorite browser. Your browser may or may not support this or that new HTML5 element when you experiment with the examples given. The same is true of any particular editing tool or environment you like to use. My aim is to present HTML that works reliably across all modern browsers and is pleasing to all user agents.

HTML5 and Web Standards

Over the past two decades, HTML has evolved through several iterations—HTML, HTML2, HTML3, HTML3.2, HTML4, HTML4.1, XHTML. These changes have been driven by both standards-setting organizations, such as the W3C, and individual software companies, such as Netscape and Microsoft. HTML5 is the next iteration. Technically, it is not yet a standard, and it will not be for several years. It is the W3C's working draft for the standard that it will eventually recommend to official standards organizations around the world. Still, browser manufacturers are already adopting HTML5 features.

For now, HTML5 is best thought of as a directional guide to good standards of practice in Web design. New HTML5 elements and attributes provide a richer description of online documents as interactive multimedia spaces. Prior HTML versions (HTML4 and XHTML) are tied to a print metaphor of a page to which interactive capabilities and media support have been added ad hoc. Many pages on the Web are the online equivalent of printed pages. In contrast, HTML5 encourages a broader conception of the Web as a unified, intelligent, interactive, hyperlinked medium.

For online document authors, HTML5 adds **new elements** to define document sections (the section element) and new section subelements to define page headers (header) and footers (footer). Section headings can be composed of heading groups (hgroup) and can contain the new navigation (nav) element. HTML4 provided only a single division element (div) for these purposes, and coders used id and class attributes to make the distinction in usage. There is a new article element (article) and a means (the aside element) to designate text that's tangential to the main topic. There is even an element for indicating sarcastic remarks (sarcasm) in the W3C draft specification, but I think this is an inside joke.

For Web developers, the HTML5 draft specification for the first time describes how the browser should expose HTML elements to scripts. Using JavaScript syntax, it describes the methods that scripts may call on document objects. In other words, it describes what commands a given HTML element understands and obeys. Previous HTML specification referred generally to ECMAScript, a standardized family of languages that includes JavaScript, JScript (Microsoft's version of JavaScript), and ActionScript (Adobe's scripting language for Flash). The use of JavaScript in this book is not meant to imply the exclusion of other scripting languages.

Equally exciting is the new HTML5 canvas element. It provides a bitmap canvas area that scripts can draw on or load images and video into. A canvas element can be used to render graphs, game graphics, or other visual images on-the-fly. There are also new elements for creating meters (meter) and progress bars (progress). There are also new element attributes that allow parts of a document to be moved around the page or edited in place and saved across sessions.

Even with all these new features, HTML5 emphasizes simplicity. This is achieved by segregating the description of document content from the descriptions of presentation and interactive behavior. Web authors are encouraged to code the minimal HTML necessary to provide a semantic description of a document. This is what **Web Standards** is all about: the standards of practice that create web pages that display well on all devices and that are pleasing to everyone and everything that reads them.

Allow me to expand on this last point. **Search** has changed how we use the Web. Although a work must be read and understood by people, it is just as important that the information to help people find that work be properly constructed. In other words, a web page must be both robot-friendly and people-friendly.

This dictum of being friendly to everything (within reason) goes beyond just being browser- and robot-friendly. The Web embraces all kinds of devices, including phones, tablets, netbooks, computers, game consoles, and large public video displays, as well as devices for the visually handicapped. The Web also embraces all languages and writing systems, including right-to-left languages such as Hebrew and Farsi and ideographic character sets such as Japanese and Chinese.

We are entering the age of the collaborative Web. It is important to think about pleasing the coauthors, contributors, curators, archivists, and translators who will work with your documents long after you write them.

Do We All Have to Learn HTML5 Now?

The short answer is no. First of all, new versions of the HTML specification do not make older versions obsolete. For example, the first home page I ever created looks the same in Firefox and Chrome today as it did in Mosaic and Arena in 1994. What's important is the assurance that the web pages we build today will look and function the same in another 15 years. We may update those pages for marketing and aesthetic reasons, but we will not be forced to edit them for technical reasons. Second, if you already know some HTML, it is not a matter of learning a new language or dialect, but simply incorporating new elements into your HTML vocabulary.

If you are a content creator/editor using Web-based tools to update web pages and post articles, you need to know that any HTML markup you use in a blog post, press release, or email newsletter will be the same in all your readers' browsers. It is best for you to stick with the elements and attributes of HTML4 until HTML5 has been more widely adopted and more guidance is forthcoming on how to use the new features.

If you design websites and keep up with tech trends on a regular basis, you will learn from your online resources about browser support for new HTML5 elements, which you can incorporate into your work with appropriate fall-backs and cross-browser testing. Now is the time to play with HTML5, while you reexamine your Web design and development methods. The HTML5 Web is collaborative.

If you manage a Web design company or development shop, your websites are probably sophisticated enough that you already do browser detection. My suggestion is to let one of your programmers become your HTML5 specialist, creating HTML5-aware versions of some of your in-development and existing websites.

Summary

Here are the important points to remember from this chapter:

- HTML is a semantic markup language for online, hypertext-linked documents.
- The Web has a client/server architecture. Web servers respond to requests from user agents such as web browsers, search robots, and web page editors.

- HTML is supported by many other technologies, the most important of which are Cascading Style Sheets (CSS) for describing the presentation aspects of page elements, and JavaScript for describing element behaviors.
- The Web is global and collaborative. Observing Web standards in creating documents will help others build upon your work.
- HTML5 provides new elements and attributes for Web designers to work with. However, it is still a draft specification and thus should be seen as a guide for future projects, when more support is available.

This page intentionally left blank

Index

Symbols

A

B

C

D

E

F

G

H

I

J-K

L

M

N

O

P

Q

R

S

T

U

V

W-X-Y-Z