Julie C. Meloni

HTML5 Coverage

Sams Teach Yourself

HTML, CSS and JavaScript

All in One

SAMS

Julie C. Meloni

Sams **Teach Yourself**

HTML, CSS and JavaScript

All in One

SAMS 800 East 96th Street, Indianapolis, Indiana, 46240 USA

Sams Teach Yourself HTML, CSS, and JavaScript All in One

ISBN-13: 978-0-672-33332-3
ISBN-10: 0-672-33332-5

Library of Congress Cataloging-in-Publication data is on file.

First Printing November 2011

Acquisitions Editor
Mark Taber

Development Editor
Songlin Qiu

Managing Editor
Sandra Schroeder

Project Editor
Seth Kerney

Copy Editor
Mike Henry

Indexer
Ken Johnson

Proofreader
Jovana San Nicolas-Shirley

Technical Editor
Phil Ballard

Publishing Coordinator
Cindy Teeters

Book Designer
Gary Adair

Compositor
Trina Wurst

Contents at a Glance

Table of Contents

About the Author

Julie C. Meloni is the Lead Technologist and Architect in the Online Library Environment at the University of Virginia. Before coming to the library, she worked for more than 15 years in web application development for various corporations large and small in Silicon Valley. She has written several books and articles on Web-based programming languages and database topics, including the bestselling *Sams Teach Yourself PHP, MySQL, and Apache All in One*.

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

You can email or write directly to let us know what you did or didn't like about this book—as well as what we can do to make our books stronger.

Please note that we cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail we receive, we might not be able to reply to every message.

When you write, please be sure to include this book's title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

Email: feedback@samspublishing

Mail: Sams Publishing
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.

CHAPTER 1
Publishing Web Content

Before learning the intricacies of HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), and JavaScript, it is important that you gain a solid understanding of the technologies that help transform these plain-text files to the rich multimedia displays you see on your computer or handheld device when browsing the World Wide Web. For example, a file containing markup and client-side code HTML and CSS is useless without a web browser to view it, and no one besides yourself will see your content unless a web server is involved. Web servers make your content available to others who, in turn, use their web browsers to navigate to an address and wait for the server to send information to them. You will be intimately involved in this publishing process because you must create files and then put them on a server to make them available in the first place, and you must ensure that your content will appear to the end user as you intended.

WHAT YOU'LL LEARN IN THIS CHAPTER:

- A very brief history of the World Wide Web
- What is meant by the term *web page*, and why that term doesn't always reflect all the content involved
- How content gets from your personal computer to someone else's web browser
- How to select a web hosting provider
- How different web browsers and device types can affect your content
- How to transfer files to your web server using FTP
- Where files should be placed on a web server
- How to distribute web content without a web server
- How to use other publishing methods such as blogs
- Tips for testing the appearance and functionality of web content.

A Brief History of HTML and the World Wide Web

Once upon a time, back when there weren't any footprints on the moon, some farsighted folks decided to see whether they could connect several major computer networks together. I'll spare you the names and stories (there are plenty of both), but the eventual result was the "mother of all networks," which we call the Internet.

Until 1990, accessing information through the Internet was a rather technical affair. It was so hard, in fact, that even Ph.D.-holding physicists were often frustrated when trying to swap data. One such physicist, the now-famous (and knighted) Sir Tim Berners-Lee, cooked up a way to easily cross-reference text on the Internet through hypertext links.

NOTE

For more information about the history of the World Wide Web, see the Wikipedia article on this topic: http://en.wikipedia.org/wiki/History_of_the_Web.

This wasn't a new idea, but his simple HTML managed to thrive while more ambitious hypertext projects floundered. *Hypertext* originally meant text stored in electronic form with cross-reference links between pages. It is now a broader term that refers to just about any object (text, images, files, and so on) that can be linked to other objects. *Hypertext Markup Language* is a language for describing how text, graphics, and files containing other information are organized and linked together.

By 1993, only 100 or so computers throughout the world were equipped to serve up HTML pages. Those interlinked pages were dubbed the *World Wide Web (WWW)*, and several web browser programs had been written to allow people to view web pages. Because of the growing popularity of the Web, a few programmers soon wrote web browsers that could view graphical images along with text. From that point forward, the continued development of web browser software and the standardization of the HTML—and XHTML—languages has lead us to the world we live in today, one in which more than 110 million web servers answer requests for more than 25 billion text and multimedia files.

These few paragraphs really are a brief history of what has been a remarkable period. Today's college freshmen have never known a time in which the Web didn't exist, and the idea of always-on information and ubiquitous computing will shape all aspects of our lives moving forward. Instead of seeing web content creation and management as a set of skills possessed only by a few technically oriented folks (okay, call them geeks if you will), by the end of this book, you will see that these are skills that anyone can master, regardless of inherent geekiness.

Creating Web Content

You might have noticed the use of the term *web content* rather than *web pages*—that was intentional. Although we talk of "visiting a web page," what we really mean is something like "looking at all the text and the images at one address on our computer." The text that we read, and the images that we see, are rendered by our web browsers, which are given certain instructions found in individual files.

Those files contain text that is *marked up*, or surrounded by, HTML codes that tell the browser how to display the text—as a heading, as a paragraph, in a red font, and so on. Some HTML markup tells the browser to display

an image or video file rather than plain text, which brings me back to the point: Different types of content are sent to your web browser, so simply saying *web page* doesn't begin to cover it. Here we use the term *web content* instead, to cover the full range of text, image, audio, video, and other media found online.

In later chapters, you will learn the basics of linking to or creating the various types of multimedia web content found in websites. All you need to remember at this point is that *you* are in control of the content a user sees when visiting your website. Beginning with the file that contains text to display or codes that tell the server to send a graphic along to the user's web browser, you have to plan, design, and implement all the pieces that will eventually make up your web presence. As you will learn throughout this book, it is not a difficult process as long as you understand all the little steps along the way.

In its most fundamental form, web content begins with a simple text file containing HTML or XHTML markup. XHTML is another flavor of HTML; the "X" stands for eXtensible, and you will learn more about it as you continue through the chapters. The most important thing to know from the outset is that all the examples in this book are HTML 4 and XHTML compatible, meaning that they will be rendered similarly both now and in the future by any newer generations of web browsers. That is one of the benefits of writing standards-compliant code: You do not have to worry about going back to your code sometime in the future and changing it because it doesn't work. Your code will likely always work for as long as web browsers adhere to standards (hopefully a long time).

Understanding Web Content Delivery

Several processes occur, in many different locations, to eventually produce web content that you can see. These processes occur very quickly—on the order of milliseconds—and occur behind the scenes. In other words, although we might think all we are doing is opening a web browser, typing in a web address, and instantaneously seeing the content we requested, technology in the background is working hard on our behalf. Figure 1.1 shows the basic interaction between a browser and a server.

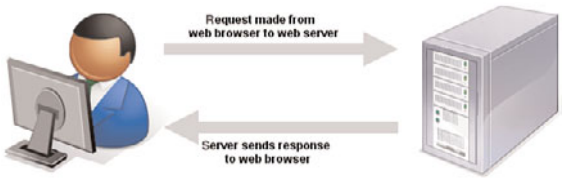


FIGURE 1.1
A browser request and a server response.

However, there are several steps in the process—and potentially several trips between the browser and server—before you see the entire content of the site you requested.

Suppose you want to do a Google search, so you dutifully type **http://www.google.com** in the address bar or select the Google bookmark from your bookmarks list. Almost immediately, your browser will show you something like what's shown in Figure 1.2.

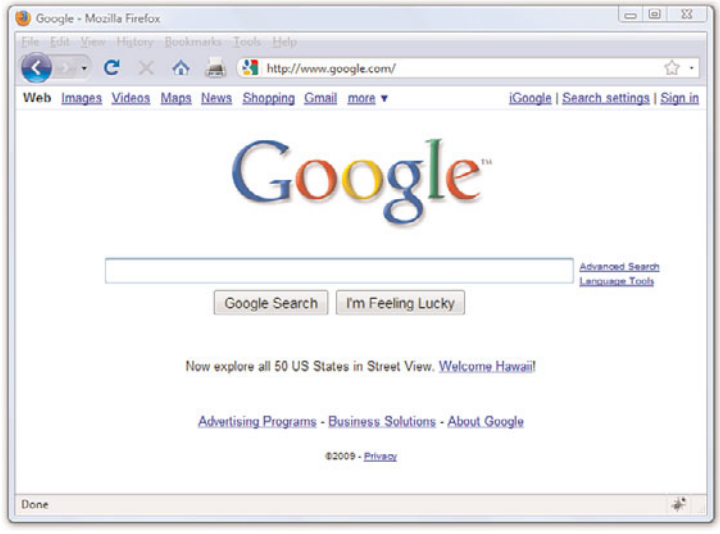


FIGURE 1.2
Visiting www.google.com.

Figure 1.2 shows a website that contains text plus one image (the Google logo). A simple version of the processes that occurred to retrieve that text and image from a web server and display it on your screen is as follows:

1. Your web browser sends a request for the index.html file located at the http://www.google.com/ address. The index.html file does not have to be part of the address that you type in the address bar; you'll learn more about the index.html file further along in this chapter.

2. After receiving the request for a specific file, the web server process looks in its directory contents for the specific file, opens it, and sends the content of that file back to your web browser.
3. The web browser receives the content of the index.html file, which is text marked up with HTML codes, and renders the content based on these HTML codes. While rendering the content, the browser happens upon the HTML code for the Google logo, which you can see in Figure 1.2. The HTML code looks like this:

   ```
   <img src="/logos/logo.gif" width="384" height="121" border="0"
   alt="Google"/>
   ```

   The tag provides attributes that tell the browser the file source location (`src`), width (`width`), height (`height`), border type (`border`), and alternative text (`alt`) necessary to display the logo. You will learn more about attributes throughout later chapters.
4. The browser looks at the `src` attribute in the `<img/>` tag to find the source location. In this case, the image logo.gif can be found in the logos directory at the same web address (www.google.com) from which the browser retrieved the HTML file.
5. The browser requests the file at the http://www.google.com/logos/logo.gif web address.
6. The web server interprets that request, finds the file, and sends the contents of that file to the web browser that requested it.
7. The web browser displays the image on your monitor.

As you can see in the description of the web content delivery process, web browsers do more than simply act as picture frames through which you can view content. Browsers assemble the web content components and arrange those parts according to the HTML commands in the file.

You can also view web content locally, or on your own hard drive, without the need for a web server. The process of content retrieval and display is the same as the process listed in the previous steps in that a browser looks for and interprets the codes and content of an HTML file, but the trip is shorter; the browser looks for files on your own computer's hard drive rather than on a remote machine. A web server is needed to interpret any server-based programming language embedded in the files, but that is outside the scope of this book. In fact, you could work through all the chapters in this book without having a web server to call your own, but then nobody but you could view your masterpieces.

Selecting a Web Hosting Provider

Despite just telling you that you can work through all the chapters in this book without having a web server, having a web server is the recommended method for continuing on. Don't worry—obtaining a hosting provider is usually a quick, painless, and relatively inexpensive process. In fact, you can get your own domain name and a year of web hosting for just slightly more than the cost of the book you are reading now.

If you type **web hosting provider** in your search engine of choice, you will get millions of hits and an endless list of sponsored search results (also known as *ads*). There are not this many web hosting providers in the world, although it might seem like there are. Even if you are looking at a managed list of hosting providers, it can be overwhelming—especially if all you are looking for is a place to host a simple website for yourself or your company or organization.

You'll want to narrow your search when looking for a provider and choose one that best meets your needs. Some selection criteria for a web hosting provider include the following"

- **Reliability/server "uptime"**—If you have an online presence, you want to make sure people can actually get there consistently.
- **Customer service**—Look for multiple methods for contacting customer service (phone, email, and chat) as well as online documentation for common issues.
- **Server space**—Does the hosting package include enough server space to hold all the multimedia files (images, audio, and video) you plan to include in your website (if any)?
- **Bandwidth**—Does the hosting package include enough bandwidth so that all the people visiting your site and downloading files can do so without you having to pay extra?
- **Domain name purchase and management**—Does the package include a custom domain name, or must you purchase and maintain your domain name separately from your hosting account?
- **Price**—Do not overpay for hosting. You will see a wide range of prices offered and should immediately wonder "what's the difference?" Often the difference has little to do with the quality of the service and everything to do with company overhead and what the company thinks they can get away with charging people. A good rule of thumb is that if you are paying more than $75 per year for a basic hosting package and domain name, you are probably paying too much.

Here are three reliable web hosting providers whose basic packages contain plenty of server space and bandwidth (as well as domain names and extra benefits) at a relatively low cost. If you don't go with any of these web hosting providers, you can at least use their basic package descriptions as a guideline as you shop around.

- **A Small Orange (http://www.asmallorange.com)**—The "Tiny" and "Small" hosting packages are perfect starting places for the new web content publisher.
- **DailyRazor (http://www.dailyrazor.com)**—Even its Rookie hosting package is full featured and reliable.
- **LunarPages (http://www.lunarpages.com)**—The Basic hosting package is suitable for many personal and small business websites.

NOTE

I have used all these providers (and then some) over the years and have no problem recommending any of them; predominantly, I use DailyRazor as a web hosting provider, especially for advanced development environments.

One feature of a good hosting provider is that it provides a "control panel" for you to manage aspects of your account. Figure 1.3 shows the control panel for my own hosting account at Daily Razor. Many web hosting providers offer this particular control panel software, or some control panel that is similar in design—clearly labeled icons leading to tasks you can perform to configure and manage your account.

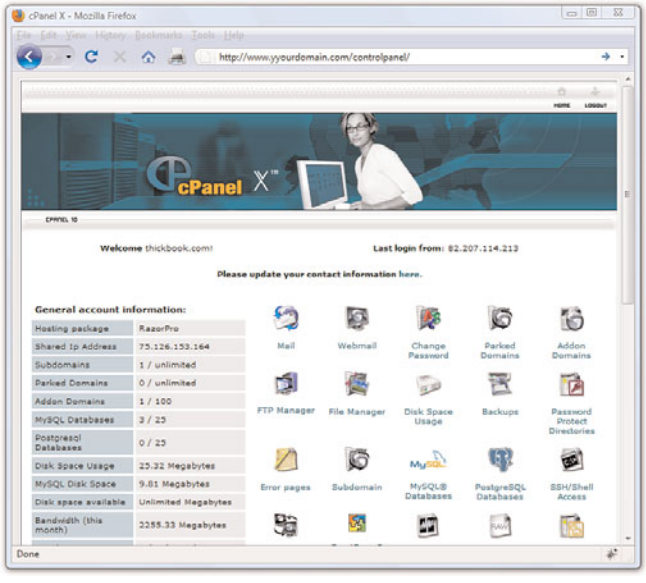


FIGURE 1.3
A sample control panel.

You might never need to use your control panel, but having it available to you simplifies the installation of databases and other software, the viewing of web statistics, and the addition of email addresses (among many other features). If you can follow instructions, you can manage your own web server—no special training required.

Testing with Multiple Web Browsers

Having just discussed the process of web content delivery and the acquisition of a web server, it might seem a little strange to step back and talk about testing your websites with multiple web browsers. However, before you go off and learn all about creating websites with HTML and CSS, do so with this very important statement in mind: Every visitor to your website will potentially use hardware and software configurations that are different than your own. Their device types (desktop, laptop, netbook, smartphone, or iPhone), their screen resolutions, their browser types, their browser window sizes, and their speed of connections will be different—remember that you cannot control any aspect of what your visitors use when they view your site. So, just as you're setting up your web hosting environment and getting ready to work, think about downloading several different web browsers so that you have a local test suite of tools available to you. Let me explain why this is important.

Although all web browsers process and handle information in the same general way, there are some specific differences among them that result in things not always looking the same in different browsers. Even users of the same version of the same web browser can alter how a page appears by choosing different display options or changing the size of their viewing windows. All the major web browsers allow users to override the background and fonts specified by the web page author with those of their own choosing. Screen resolution, window size, and optional toolbars can also change how much of a page someone sees when it first appears on their screens. You can ensure only that you write standards-compliant HTML and CSS.

Do not, under any circumstances, spend hours on end designing something that looks perfect on your own computer—unless you are willing to be disappointed when you look at it on your friend's computer, on your tablet, or on your iPhone.

You should always test your websites with as many of these web browsers as possible:

- Apple Safari (http://www.apple.com/safari/) for Mac and Windows
- Google Chrome (http://www.google.com/chrome) for Windows
- Mozilla Firefox (http://www.mozilla.com/firefox/) for Mac, Windows, and Linux
- Microsoft Internet Explorer (http://www.microsoft.com/ie) for Windows
- Opera (http://www.opera.com/) for Mac, Windows, and Linux/UNIX

Now that you have a development environment set up, or at least some idea of the type you'd like to set up in the future, let's move on to creating a test file.

Creating a Sample File

Before we begin, take a look at Listing 1.1. This listing represents a simple piece of web content—a few lines of HTML that print "Hello World! Welcome to My Web Server." in large, bold letters on two lines centered within the browser window.

LISTING 1.1 Our Sample HTML File

```
<html>
<head>
<title>Hello World!</title>
</head>
<body>
<h1 style="text-align: center">Hello World!<br/>Welcome to My Web
➥Server.</h1>
</body>
</html>
```

To make use of this content, open a text editor of your choice, such as Notepad (on Windows) or TextEdit (on a Mac). Do not use WordPad, Microsoft Word, or other full-featured word-processing software because those programs create different sorts of files than the plain-text files we use for web content.

Type the content that you see in Listing 1.1, and then save the file using **sample.html** as the filename. The .html extension tells the web server that your file is, indeed, full of HTML. When the file contents are sent to the web browser that requests it, the browser will also know that it is HTML and will render it appropriately.

NOTE

You will learn a bit about text editors in Chapter 2, "Understanding HTML and XHTML Connections." Right now, I just want you to have a sample file that you can put on a web server!

Now that you have a sample HTML file to use—and hopefully somewhere to put it, such as a web hosting account—let's get to publishing your web content.

Using FTP to Transfer Files

As you've learned so far, you have to put your web content on a web server to make it accessible to others. This process typically occurs by using *File Transfer Protocol (FTP)*. To use FTP, you need an FTP client—a program used to transfer files from your computer to a web server.

FTP clients require three pieces of information to connect to your web server; this information will have been sent to you by your hosting provider after you set up your account:

- The hostname, or address, to which you will connect
- Your account username
- Your account password

After you have this information, you are ready to use an FTP client to transfer content to your web server.

Selecting an FTP Client

Regardless of the FTP client you use, FTP clients generally use the same type of interface. Figure 1.4 shows an example of FireFTP, which is an FTP client used with the Firefox web browser. The directory listing of the local machine (your computer) appears on the left of your screen and the directory listing of the remote machine (the web server) appears on the right. Typically, you will see right-arrow and left-arrow buttons—as shown in Figure 1.4. The right arrow sends selected files from your computer to your web server; the left arrow sends files from the web server to your computer. Many FTP clients also enable you to simply select files, and then drag and drop those files to the target machines.

There are many FTP clients freely available to you, but you can also transfer files via the web-based File Manager tool that is likely part of your web server's control panel. However, that method of file transfer typically introduces more steps into the process and isn't nearly as streamlined (or simple) as installing an FTP client on your own machine.

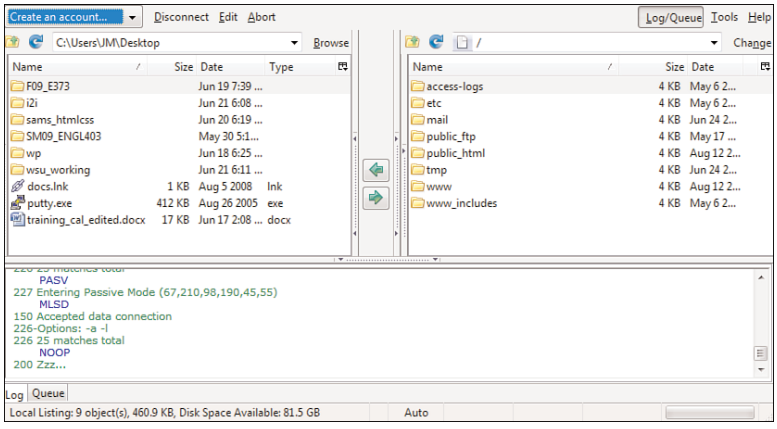


FIGURE 1.4
The FireFTP interface.

Here are some popular free FTP clients:

- Classic FTP (http://www.nchsoftware.com/classic/) for Mac and Windows
- Cyberduck (http://cyberduck.ch/) for Mac
- Fetch (http://fetchsoftworks.com/) for Mac
- FileZilla (http://filezilla-project.org/) for all platforms
- FireFTP (http://fireftp.mozdev.org/) Firefox extension for all platforms

When you have selected an FTP client and installed it on your computer, you are ready to upload and download files from your web server. In the next section, you'll see how this process works using the sample file in Listing 1.1.

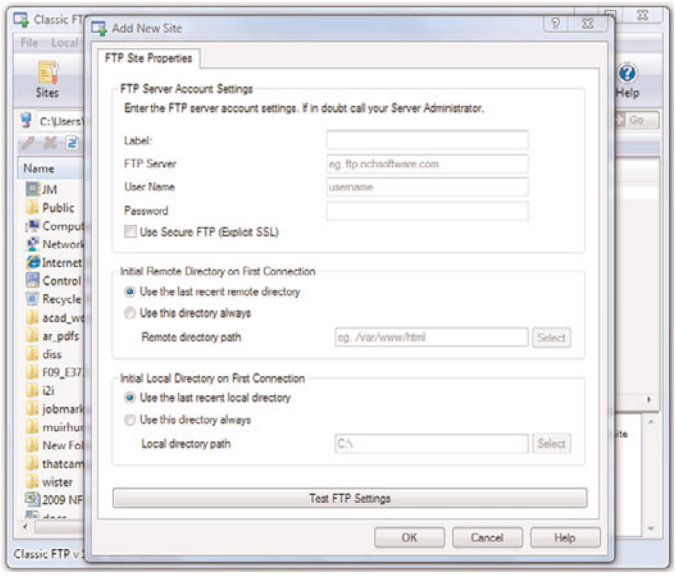
Using an FTP Client

The following steps show how to use Classic FTP to connect to your web server and transfer a file. However, all FTP clients use similar, if not exact, interfaces. If you understand the following steps, you should be able to use any FTP client.

Remember, you first need the hostname, the account username, and the account password.

1. Start the Classic FTP program and click the Connect button. You will be prompted to fill out information for the site to which you want to connect, as shown in Figure 1.5.

FIGURE 1.5
Connecting to a new site in Classic FTP.



2. Fill in each of the items shown in Figure 1.5 as follows:

   - The site Label is the name you'll use to refer to your own site. Nobody else will see this name, so enter whatever you want.
   - The FTP Server is the FTP address of the web server to which you need to send your web pages. This address will have been given to you by your hosting provider. It will probably be *yourdomain*.com, but check the information you received when you signed up for service.
   - The User Name field and the Password field should also be completed using information given to you by your hosting provider.
   - Don't change the values for Initial Remote Directory on First Connection and Initial Local Directory on First Connection until you are used to using the client and have established a workflow.

3. When you're finished with the settings, click OK to save the settings and establish a connection with the web server.

   You will see a dialog box indicating that Classic FTP is attempting to connect to the web server. Upon successful connection, you will see an interface similar to Figure 1.6, showing the contents of the local directory on the left and the contents of your web server on the right.

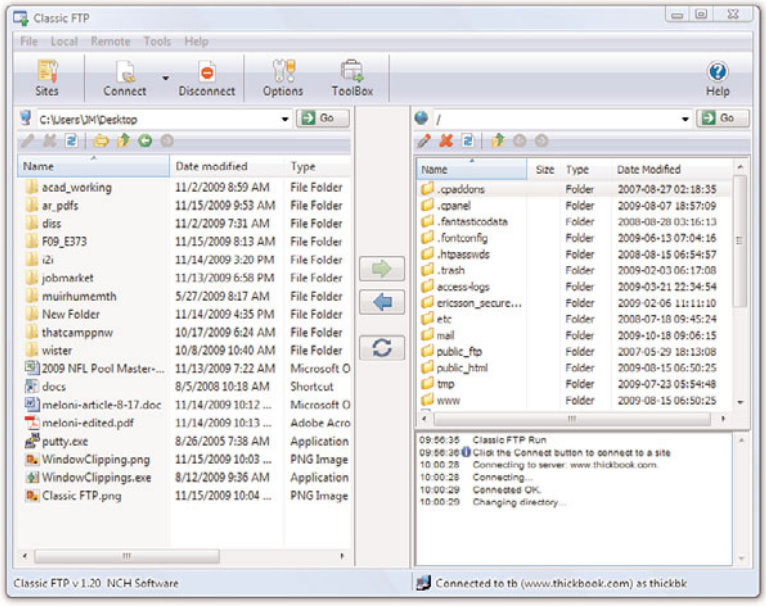


FIGURE 1.6
A successful connection to a remote web server via Classic FTP.

4. You are now *almost* ready to transfer files to your web server. All that remains is to change directories to what is called the *document root* of your web server. The document root of your web server is the directory that is designated as the top-level directory for your web content—the starting point of the directory structure, which you will learn more about later in this chapter. Often, this directory will be named public_html (as shown in Figure 1.6), www (also shown in Figure 1.6, as www has been created as an alias for public_html), or htdocs. This is not a directory that you will have to create because your hosting provider will have created it for you.

   Double-click the document root directory name to open it. The display shown on the right of the FTP client interface should change to show the contents of this directory. (It will probably be empty at this point, unless your web hosting provider has put placeholder files in that directory on your behalf.)

5. The goal is to transfer the sample.html file you created earlier from your computer to the web server. Find the file in the directory listing on the left of the FTP client interface (navigate around if you have to) and click it once to highlight the filename.

6. Click the right-arrow button in the middle of the client interface to send the file to the web server. After the file transfer is completed, the right side of the client interface should refresh to show you that the file has made it to its destination.
7. Click the Disconnect button to close the connection, and then exit out of the Classic FTP program.

These steps are conceptually similar to the steps you will take anytime you want to send files to your web server via FTP. You can also use your FTP client to create subdirectories on the remote web server. To create a subdirectory using Classic FTP, click the Remote menu, and then click New Folder. Different FTP clients will have different interface options to achieve the same goal.

Understanding Where to Place Files on the Web Server

An important aspect of maintaining web content is determining how you will organize that content—not only for the user to find, but also for you to maintain on your server. Putting files in directories will help you to manage those files.

Naming and organizing directories on your web server, and developing rules for file maintenance, is completely up to you. However, maintaining a well-organized server simply makes your management of its content more efficient in the long run.

Basic File Management

As you browse the Web, you might have noticed that URLs change as you navigate through websites. For instance, if you're looking at a company's website and you click on graphical navigation leading to the company's products or services, the URL will probably change from

http://www.companyname.com/

to

http://www.companyname.com/products/

or

http://www.companyname.com/services/

In the previous section, I used the term *document root* without really explaining what that is all about. The document root of a web server is essentially the trailing slash in the full URL. For instance, if your domain is *yourdomain*.com and your URL is http://www.*yourdomain*.com/, the document root is the directory represented by the trailing slash (/). The document root is the starting point of the directory structure you create on your web server; it is the place where the web server begins looking for files requested by the web browser.

If you put the sample.html file in your document root as previously directed, you will be able to access it via a web browser at the following URL:

http://www.*yourdomain*.com/sample.html

If you were to enter this URL into your web browser, you would see the rendered sample.html file, as shown in Figure 1.7.

FIGURE 1.7
The sample.html file accessed via a web browser.

However, if you created a new directory within the document root and put the sample.html file in that directory, the file would be accessed at this URL:

http://www.*yourdomain*.com/newdirectory/sample.html

If you put the sample.html file in the directory you originally saw upon connecting to your server—that is, you did *not* change directories and place the file in the document root—the sample.html file would not be accessible from your web server at any URL. The file will still be on the machine that you know as your web server, but because the file is not in the document root—where the server software knows to start looking for files—it will never be accessible to anyone via a web browser.

The bottom line? Always navigate to the document root of your web server before you start transferring files.

This is especially true with graphics and other multimedia files. A common directory on web servers is called *images*, where, as you can imagine, all the image assets are placed for retrieval. Other popular directories include css for stylesheet files (if you are using more than one) and js for external JavaScript files. Or, if you know you will have an area on your website where visitors can download many different types of files, you might simply call that directory downloads.

Whether it's a ZIP file containing your art portfolio or an Excel spreadsheet with sales numbers, it's often useful to publish files on the Internet that aren't simply web pages. To make a file available on the Web that isn't an HTML file, just upload the file to your website as if it *were* an HTML file, following the instructions provided earlier in this chapter for uploading. After the file is uploaded to the web server, you can create a link to it (as you'll learn in later chapters). In other words, your web server can serve much more than HTML.

Here's a sample of the HTML code that you will learn more about later in this book. The following code would be used for a file named artfolio.zip, located in the downloads directory of your website, and link text that reads "Download my art portfolio!":

```
<a href="/downloads/artfolio.zip">Download my art portfolio!</a>
```

Using an Index Page

When you think of an index, you probably think of the section in the back of a book that tells you where to look for various keywords and topics. The index file in a web server directory can serve that purpose—if you design it that way. In fact, that's where the name originates.

The index.html file (or just *index file*, as it's usually referred to) is the name you give to the page you want people to see as the default file when they navigate to a specific directory in your website. If you've created that page with usability in mind, your users will be able to get to all content in that section from the index page.

For example, Figure 1.8 shows the drop-down navigation and left-side navigation both contain links to three pages: Solutions Overview (the section index page itself), Connection Management, and Cost Management.

The content of the page itself—called index.html and located within the solutions directory—also has links to those two additional pages in the solutions section. When users arrive at the index page of the "solutions" section in this particular website (at least at the time of the snapshot), they can reach any other page in that section (and in three different ways!).

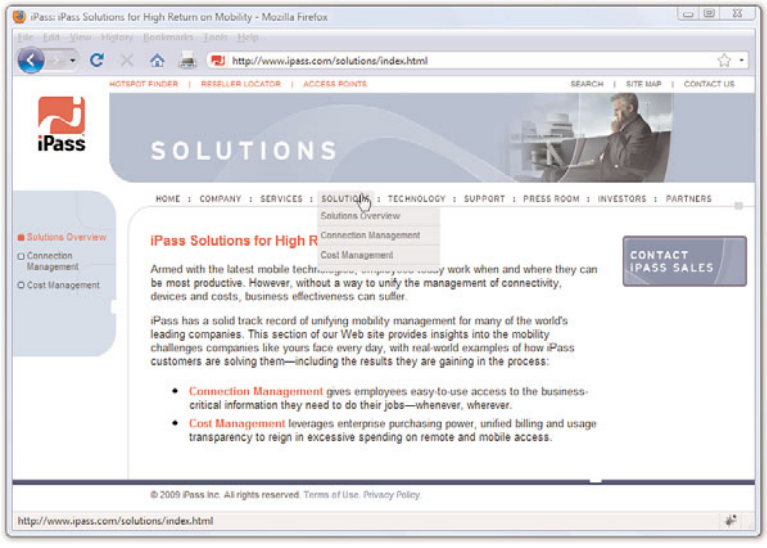


FIGURE 1.8
Showing a good section index page.

Another function of the index page is that when users visit a directory on your site that has an index page, but they do not specify that page, they will still land on the main page for that section of your site—or for the site itself.

For instance, in the previous example, a user could have typed either of the following URLs and landed on the main page of the solutions section of that website:

http://www.ipass.com/solutions/

http://www.ipass.com/solutions/index.html

Had there been no index.html page in the solutions directory, the results would depend on the configuration of the web server. If the server is configured to disallow directory browsing, the user would have seen a "Directory Listing Denied" message when attempting to access the URL without a specified page name. However, if the server is configured to allow directory browsing, the user would have seen a list of the files in that directory.

These server configuration options will have already been determined for you by your hosting provider. If your hosting provider enables you to modify server settings via a control panel, you can change these settings so that your server responds to requests based on your own requirements.

Not only is the index file used in subdirectories, it's used in the top-level directory (or document root) of your website as well. The first page of your website—or *home page* or *main page*, or however you like to refer to the web content you want users to see when they first visit your domain—should be named index.html and placed in the document root of your web server. This will ensure that when users type http://www.*yourdomain*.com/ into their web browsers, the server will respond with content you intended them to see (rather than "Directory Listing Denied" or some other unintended consequence).

Distributing Content Without a Web Server

Publishing HTML and multimedia files online is obviously the primary reason to learn HTML and create web content. However, there are also situations in which other forms of publishing simply aren't viable. For example, you might want to distribute CD-ROMs, DVD-ROMs, or USB drives at a trade show with marketing materials designed as web content—that is, hyperlinked text viewable through a web browser, but without a web server involved. You might also want to include HTML-based instructional manuals on removable media for students at a training seminar. These are just two examples of how HTML pages can be used in publishing scenarios that don't involve the Internet.

This process is also called creating *local* sites; even though there's no web server involved, these bundles of hypertext content are still called *sites*. The local term comes into play because your files are accessed locally and not remotely (via a web server).

Publishing Content Locally

Let's assume you need to create a local site that you want to distribute on a USB drive. Even the cheapest USB drives hold so much data these days—and basic hypertext files are quite small—that you can distribute an entire site *and a fully functioning web browser* all on one little drive.

NOTE

Distributing a web browser isn't required when creating and distributing a local site, although it's a nice touch. You can reasonably assume that users have their own web browsers and will open the index.html file in a directory to start browsing the hyperlinked content. However, if you would like to distribute a web browser on the USB drive, go to http://www.portableapps.com/ and look for Portable Firefox.

Simply think of the directory structure of your USB drive just as you would the directory structure of your web server. The top-level of the USB drive directory structure can be your document root. Or if you are distributing a web browser along with the content, you might have two directories—for example, one named `browser` and one named `content`. In that case, the `content` directory would be your document root. Within the document root, you could have additional subfolders in which you place content and other multimedia assets.

It's as important to maintain good organization with a local site as it is with a remote website so that you avoid broken links in your HTML files. You will learn more about the specifics of linking together files in a later chapter.

Publishing Content on a Blog

You might have a blog hosted by a third-party, such as Blogger or WordPress (among others), and thus have already published content without having a dedicated web server or even knowing any HTML. These services offer *visual editors* in addition to *source editors*, meaning that you can type your words and add visual formatting such as bold, italics, or font colors without knowing the HTML for these actions. But still, the content becomes actual HTML when you click the Publish button in these editors.

However, with the knowledge you will acquire throughout this book, your blogging will be enhanced because you will able to use the source editor for your blog post content and blog templates, thus affording you more control over the look and feel of that content. These actions occur differently from the process you learned for creating an HTML file and uploading it via FTP to your own dedicated web server, but I would be remiss if I did not note that blogging is, in fact, a form of web publishing.

Tips for Testing Web Content

Whenever you transfer files to your web server or place them on removable media for local browsing, you should immediately test every page thoroughly. The following checklist will help ensure that your web content behaves the way you expected. Note that some of the terms might be unfamiliar to you at this point, but come back to this checklist as you progress through this book and create larger projects:

- Before you transfer your files, test them locally on your machine to ensure that the links work and the content reflects the visual design you intended. After you transfer the pages to a web server or removable device, test them all again.
- Perform these tests with as many browsers that you can—Chrome, Firefox, Internet Explorer, Opera, and Safari is a good list—and on both Mac and Windows platforms. If possible, check at low resolution (800×600) and high resolution (1600×1200).
- Turn off auto image loading in your web browser before you start testing so that you can see what each page looks like without the graphics. Check your alt tag messages, and then turn image loading back on to load the graphics and review the page carefully again.
- Use your browser's font size settings to look at each page in various font sizes to ensure that your layout doesn't fall to pieces if users override your font specifications with their own.
- Wait for each page to completely finish loading, and then scroll all the way down to make sure that all images appear where they should.
- Time how long it takes each page to load. Does it take more than a few seconds to load? If so, is the information on that page valuable enough to keep users from going elsewhere before the page finishes loading? Granted, broadband connections are common, but that doesn't mean you should load up your pages with 1MB images.

If your pages pass all those tests, you can rest easy; your site is ready for public viewing.

Summary

This chapter introduced you to the concept of using HTML to mark-up text files to produce web content. You also learned that there is more to web content than just the "page"—web content also includes image, audio, and video files. All of this content lives on a web server—a remote machine often far away from your own computer. On your computer or other device, you use a web browser to request, retrieve, and eventually display web content on your screen.

You learned the criteria you should consider when determining if a web hosting provider fits your needs. After you have a web hosting provider,

you can begin to transfer files to your web server using an FTP client. You also learned a little bit about web server directory structures and file management, as well as the very important purpose of the index.html file in a given web server directory. You discovered that you can distribute web content on removable media, and how to go about structuring the files and directories to achieve the goal of viewing content without using a remote web server. Finally, you learned the importance of testing your work in multiple browsers after you've placed it on a web server. Writing valid, standards-compliant HTML and CSS will help ensure your site looks reasonably similar for all visitors, but you still shouldn't design without receiving input from potential users outside your development team—it is even more important to get input from others when you are a design team of one!

Q&A

Q. I've looked at the HTML source of some web pages on the Internet and it looks frighteningly difficult to learn. Do I have to think like a computer programmer to learn this stuff?

A. Although complex HTML pages can indeed look daunting, learning HTML is much easier than learning actual software programming languages (such as C++ or Java). HTML is a markup language rather than a programming language; you mark-up text so that the text can be rendered a certain way by the browser. That's a completely different set of thought processes than developing a computer program. You really don't need any experience or skill as a computer programmer to be a successful web content author.

One of the reasons the HTML behind many commercial websites looks complicated is because it was likely created by a visual web design tool—a "what you see is what you get" or "WYSIWYG" editor that will use whatever markup its software developer told it to use in certain circumstances—as opposed to being hand-coded, in which you are completely in control of the resulting markup. In this book, you are taught fundamental coding from the ground up, which typically results in clean, easy-to-read source code. Visual web design tools have a knack for making code difficult to read and for producing code that is convoluted and non-standards compliant.

Q. All the tests you recommend would take longer than creating my pages! Can't I get away with less testing?

A. If your pages aren't intended to make money or provide an important service, it's probably not a big deal if they look funny to some users or produce errors once in a while. In that case, just test each page with a couple of different browsers and call it a day. However, if you need to project a professional image, there is no substitute for rigorous testing.

Q. Seriously, who cares how I organize my web content?

A. Believe it or not, the organization of your web content does matter to search engines and potential visitors to your site—you'll learn more about this in Chapter 28, "Helping People Find Your Web Pages." But overall, having an organized web server directory structure will help you keep track of content that you are likely to update frequently. For instance, if you have a dedicated directory for images or multimedia, you will know exactly where to look for a file you want to update—no need to hunt through directories containing other content.

Workshop

The workshop contains quiz questions and exercises to help you solidify your understanding of the material covered. Try to answer all questions before looking at the "Answers" section that follows.

Quiz

1. How many files would you need to store on a web server to produce a single web page with some text and two images on it?
2. What are some of the features to look for in a web hosting provider?
3. What three pieces of information do you need to connect to your web server via FTP?
4. What is the purpose of the index.html file?
5. Does your website have to include a directory structure?

Answers

1. You would need three: one for the web page itself, which includes the text and the HTML markup, and one for each of the two images.
2. Look for reliability, customer service, web space and bandwidth, domain name service, site management extras, and price.
3. The hostname, your account username, and your account password.
4. The index.html file is typically the default file for a directory within a web server. It allows users to access http://www.yourdomain.com/somedirectory/ without using a trailing file name and still end up in the appropriate place.
5. No. Using a directory structure for file organization is completely up to you, although it is highly recommended to use one because it simplifies content maintenance.

Exercises

- Get your web hosting in order—are you going to go through the chapters in this book by viewing files locally on your own computer, or are you going to use a web hosting provider? Note that most web hosting providers will have you up and running the same day you purchase your hosting plan.
- If you are using an external hosting provider, and then using your FTP client, create a subdirectory within the document root of your website. Paste the contents of the sample.html file into another file named index.html, change the text between the `<title>` and `</title>` tags to something new, and change the text between the `<h1>` and `</h1>` tags to something new. Save the file and upload it to the new subdirectory. Use your web browser to navigate to the new directory on your web server and see that the content in the index.html file appears. Then, using your FTP client, delete the index.html file from the remote subdirectory. Return to that URL with your web browser, reload the page, and see how the server responds without the index.html file in place.
- Using the same set of files created in the previous exercise, place these files on a removable media device—a CD-ROM or a USB drive, for example. Use your browser to navigate this local version of your sample website, and think about the instructions you would have to distribute with this removable media so that others could use it.

CHAPTER 2
Understanding HTML and XHTML Connections

The first chapter gave you a basic idea of the process behind creating web content and viewing it online (or locally, if you do not yet have a web hosting provider). In this chapter, we'll get down to the business of explaining the various elements that must appear in an HTML file so that it is displayed appropriately in your web browser.

By the end of the chapter, you'll learn how HTML differs from XHTML and why there are two different languages designed to do the same thing—create web content. In general, this chapter provides a quick summary of HTML and XHTML basics and gives some practical tips to make the most of your time as a web page author and publisher. It's not all theory, however; you do get to see a real web page and the HTML code behind it.

WHAT YOU'LL LEARN IN THIS CHAPTER:

- How to create a simple web page in HTML
- How to include all the HTML Tags that every web page must have
- How to organize a page with paragraphs and line breaks
- How to organize your content with headings
- How to validate your web content
- How to differentiate between HTML, XML, XHTML, and HTML5

Getting Prepared

Here's a review of what you need to do before you're ready to use the rest of this book:

1. Get a computer. I used a computer running Ubuntu (Linux) to test the sample web content and capture the figures in this book, but you can use any Windows, Macintosh, or Linux/UNIX machine to create and view your web content.
2. Get a connection to the Internet. Whether you have a dial-up, wireless, or broadband connection doesn't matter for the creation and viewing of your web content, but the faster the connection, the better for the overall experience. The Internet service provider (ISP), school, or business that provides your Internet connection can help you with the details of setting it up properly. Additionally, many public spaces such as coffee shops, bookstores, and libraries offer free wireless Internet service that you can use if you have a laptop computer with Wi-Fi network support.

NOTE

Not sure how to find an ISP? The best way is to comparison-shop online (using a friend's computer or a public computer that's already connected to the Internet). You'll find a comprehensive list of national and regional ISPs at http://www.thelist.com/.

CAUTION

Although all web browsers process and handle information in the same general way, there are some specific differences among them that result in things not always looking the same in different browsers. Be sure to check your web pages in multiple browsers to make sure that they look reasonably consistent.

NOTE

As discussed in the first chapter, if you plan to put your web content on the Internet (as opposed to publishing it on CD-ROM or a local intranet), you'll need to transfer it to a computer that is connected to the Internet 24 hours a day. The same company or school that provides you with Internet access might also provide web space; if not, you might need to pay a hosting provider for the service.

3. Get web browser software. This is the software your computer needs to retrieve and display web content. As you learned in the first chapter, the most popular browser software (in alphabetical order) is Apple Safari, Google Chrome, Microsoft Internet Explorer, Mozilla Firefox, and Opera. It's a good idea to install several of these browsers so that you can experiment and make sure that your content looks consistent across them all; you can't make assumptions about the browsers other people are using.
4. Explore! Use a web browser to look around the Internet for websites that are similar in content or appearance to those you'd like to create. Note what frustrates you about some pages, what attracts you and keeps you reading others, and what makes you come back to some pages over and over again. If there is a particular topic that interests you, consider searching for it using a popular search engine such as Google (http://www.google.com/) or Bing (http://www.bing.com/).

Getting Started with a Simple Web Page

In the first chapter, you learned that a web page is just a text file that is marked up by (or surrounded by) HTML codes that tell the browser how to display the text. To create these text files, use a *text editor* such as Notepad (on Windows) or TextEdit (on a Mac)—do not use WordPad, Microsoft Word, or other full-featured word-processing software because those create different sorts of files than the plain-text files we use for web content.

Before you begin working, you should start with some text that you want to put on a web page:

1. Find (or write) a few paragraphs of text about yourself, your family, your company, your softball team, or some other subject in which you're interested.
2. Save this text as plain, standard ASCII text. Notepad and most simple text editors always save files as plain text, but if you're using another program, you might need to choose this file type as an option (after selecting File, Save As).

As you go through this chapter, you will add HTML markup (called *tags*) to the text file, thus making it into web content.

When you save files containing HTML tags, always give them a name ending in .html. This is important: If you forget to type the .html at the end of the filename when you save the file, most text editors will give it some other extension (such as .txt). If that happens, you might not be able to find the file when you try to look at it with a web browser; if you find it, it certainly won't display properly. In other words, web browsers expect a web page file to have a file extension of .html.

When visiting websites, you might also encounter pages with a file extension of .htm, which is also an acceptable file extension to use. You might find other file extensions used on the Web, such as .jsp (Java Server Pages), .asp (Microsoft Active Server Pages), or .php (PHP: Hypertext Preprocessor), but these file types use server-side technologies that are beyond the scope of HTML and the chapters throughout this book. However, these files also contain HTML in addition to the programming language; although the programming code in those files is compiled on the server side and all you would see on the client side is the HTML output, if you were to look at the source files, you would likely see some intricate weaving of programming and markup codes.

Listing 2.1 shows an example of text you can type and save to create a simple HTML page. If you opened this file with Firefox, you would see the page shown in Figure 2.1. Every web page you create must include the `<html></html>`, `<head></head>`, `<title></title>`, and `<body></body>` tag pairs.

LISTING 2.1 The `<html>`, `<head>`, `<title>`, and `<body>` Tags

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>The First Web Page</title>
  </head>

  <body>
    <p>
      In the beginning, Tim created the HyperText Markup Language. The Internet
      was without form and void, and text was upon the face of the monitor and
      the Hands of Tim were moving over the face of the keyboard. And Tim said,
      Let there be links; and there were links. And Tim saw that the links were
      good; and Tim separated the links from the text. Tim called the links
```

CAUTION

To reiterate, because it is very important both to the outcome and the learning process itself: Do not create your first HTML file with Microsoft Word or any other HTML-compatible word processor; most of these programs attempt to rewrite your HTML for you in strange ways, potentially leaving you totally confused. Additionally, I recommend that you do *not* use a graphical, what-you-see-is-what-you-get (WYSIWYG) editor, such as Microsoft FrontPage or Adobe Dreamweaver. You'll likely find it easier and more educational to start out with a simple text editor while you're just learning HTML. You can move to visual tools (such as FrontPage and Dreamweaver) after you have a better understanding of what's going on under the hood.

NOTE

If you're using TextEdit on a Macintosh computer, the steps for creating an HTML file are a little different than for using Notepad on a Windows computer. Both are popular text editors, but with the latter, you must first click on the Format menu, select Make Plain Text, and then change the preferences under the Saving header by unchecking the box for Append '.txt' Extension to Plain Text Files. Also, the default preferences are set to show .html documents as they would appear in a browser, which won't allow you to edit them. To fix this, check Ignore Rich Text Commands in HTML Files under the Rich Text Processing header.

LISTING 2.1 Continued

```
      Anchors, and the text He called Other Stuff. And the whole thing together
      was the first Web Page.
    </p>
  </body>
</html>
```

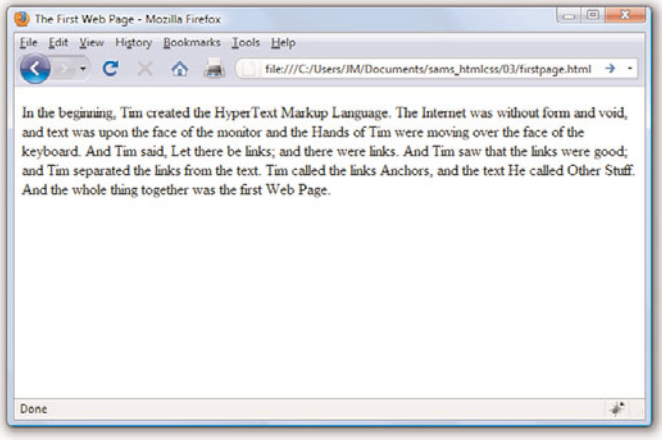


FIGURE 2.1
When you save the text in Listing 2.1 as an HTML file and view it with a web browser, only the actual title and body text are displayed.

NOTE

Technically speaking, HTML5 will be the next web standard but it's not quite at the point of full adoption. Current estimates put the full adoption of HTML sometime in 2011. However, as you learn about important features of HTML and XHTML in this book, I will include notes about how HTML5 features might differ.

In Listing 2.1, as in every HTML page, the words starting with < and ending with > are actually coded commands. These coded commands are called *HTML tags* because they "tag" pieces of text and tell the web browser what kind of text it is. This allows the web browser to display the text appropriately.

The first few lines of code in the web page serve as standard boilerplate code that you will include in all of your pages. This code actually identifies the page as a valid XHTML 1.1 document, which means that, technically, the web page is an XHTML page. All the pages developed throughout the book are XHTML 1.1 pages. Because XHTML is a more structured version of HTML, it's still okay to generally refer to all the pages in the book as HTML pages. By targeting XHTML 1.1 with your code, you are developing web pages that adhere to the very latest web standards. This is a good thing!

TRY IT YOURSELF ▼

Creating and Viewing a Basic Web Page

Before you learn the meaning of the HTML tags used in Listing 2.1, you might want to see exactly how I went about creating and viewing the document itself. Follow these steps:

1. Type all the text in Listing 2.1, including the HTML tags, in Windows Notepad (or use Macintosh TextEdit or another text editor of your choice).
2. Select File, Save As. Be sure to select plain text (or ASCII text) as the file type.
3. Name the file **firstpage.html**.
4. Choose the folder on your hard drive where you would like to keep your web pages—and remember which folder you choose! Click the Save or OK button to save the file.
5. Now start your favorite web browser. (Leave Notepad running, too, so you can easily switch between viewing and editing your page.)

In Internet Explorer, select File, Open and click Browse. If you're using Firefox, select File, Open File. Navigate to the appropriate folder and select the firstpage.html file. Some browsers and operating systems will also enable you to drag and drop the firstpage.html file onto the browser window to view it.

Voilà! You should see the page shown in Figure 2.1.

If you have obtained a web hosting account, you could use FTP at this point to transfer the firstpage.html file to the web server. In fact, from this chapter forward, the instructions will assume you have a hosting provider and are comfortable sending files back and forth via FTP; if that is not the case, please review the first chapter before moving on. Or, if you are consciously choosing to work with files locally (without a web host), be prepared to adjust the instructions to suit your particular needs (such as ignoring the commands "transfer the files" and "type in the URL").

HTML Tags Every XHTML Web Page Must Have

The time has come for the secret language of HTML tags to be revealed to you. When you understand this language, you will have creative powers far beyond those of other humans. Don't tell the other humans, but it's really pretty easy.

NOTE

You don't need to be connected to the Internet to view a web page stored on your own computer. By default, your web browser tries to connect to the Internet every time you start it, which makes sense most of the time. However, this can be a hassle if you're developing pages locally on your hard drive (offline) and you keep getting errors about a page not being found. If you have a full-time web connection via a LAN, cable modem, or DSL, this is a moot point because the browser will never complain about being offline. Otherwise, the appropriate disciplinary action will depend on your breed of browser; check the options under your browser's Tools menu.

NOTE

It isn't terribly important that you understand concepts such as character encoding at this point. What is important is that you include the appropriate boilerplate code in your pages so that they adhere to the latest web standards. As of this writing, XHTML 1.1 is a web standard. HTML5 is not yet a web standard, but if you were creating an HTML5 document, these lines at the beginning of your HTML file would not be necessary.

NOTE

The XML/XHTML boilerplate code isn't strictly required for you to create web pages. You can delete the opening lines of code in the example so that the page starts with the `<html>` tag and it will still open fine in a web browser. The extra code is included to ensure your pages are up to date with the current web standards. Additionally, the extra code enables you to validate your web pages for accuracy, which you'll learn how to do a bit later in this chapter.

Before you get into the HTML tags, let's first address the messy-looking code at the top of Listing 2.1. The first line indicates that the HTML document is, in fact, an XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
```

The version of XML is set to 1.0, which is fairly standard, as is the type of character encoding (UTF-8).

The second and third lines of code in Listing 2.1 are even more complicated looking:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

Again, the specifics of this code aren't terribly important as long as you remember to include the code at the start of your pages. This code identifies the document as being XHTML 1.1, which then allows web browsers to make sure the code meets all the requirements of XHTML 1.1.

Most HTML tags have two parts: an *opening tag*, which indicates where a piece of text begins, and a *closing tag*, which indicates where the piece of text ends. Closing tags start with a `/` (forward slash) just after the `<` symbol.

Another type of tag is the *empty tag*, which is unique in that it doesn't include a pair of matching opening and closing tags. Instead, an empty tag consists of a single tag that starts with a `<` and ends with a `/` just before the `>` symbol.

Following is a quick summary of these three tags just to make sure you understand the role each plays:

- An *opening tag* is an HTML tag that indicates the start of an HTML command; the text affected by the command appears after the opening tag. Opening tags always begin with `<` and end with `>`, as in `<html>`.
- A *closing tag* is an HTML tag that indicates the end of an HTML command; the text affected by the command appears before the closing tag. Closing tags always begin with `</` and end with `>`, as in `</html>`.
- An *empty tag* is an HTML tag that issues an HTML command without enclosing any text in the page. Empty tags always begin with `<` and end with `/>`, as in `<br />` and `<img />`.

For example, the <body> tag in Listing 2.1 tells the web browser where the actual body text of the page begins, and </body> indicates where it ends. Everything between the <body> and </body> tags will appear in the main display area of the web browser window, as shown in Figure 2.1.

The very top of the browser window (refer to Figure 2.1) shows title text, which is any text that is located between <title> and </title>. The title text is also used to identify the page on the browser's Bookmarks or Favorites menu, depending on which browser you use. It's important to provide titles for your pages so that visitors to the page can properly bookmark them for future reference.

You will use the <body> and <title> tag pairs in every HTML page you create because every web page needs a title and body text. You will also use the <html> and <head> tag pairs, which are the other two tags shown in Listing 2.1. Putting <html> at the very beginning of a document simply indicates that the document is a web page. The </html> at the end indicates that the web page is over.

Within a page, there is a head section and a body section. Each section is identified by <head> and <body> tags. The idea is that information in the head of the page somehow describes the page but isn't actually displayed by a web browser. Information placed in the body, however, is displayed by a web browser. The <head> tag always appears near the beginning of the HTML code for a page, just after the opening <html> tag.

The <title> tag pair used to identify the title of a page appears within the head of the page, which means it is placed after the opening <head> tag and before the closing </head> tag. In upcoming chapters, you'll learn about some other advanced header information that can go between <head> and </head>, such as style sheet rules that are used to format the page, as well as the JavaScript you'll learn to write and embed.

The <p> tag used in Listing 2.1 encloses a paragraph of text. You should enclose your chunks of text in the appropriate container tags whenever possible.

NOTE

You no doubt noticed in Listing 2.1 that there is some extra code associated with the <html> tag. This code consists of two attributes (xmlns and xml:lang), which are used to specify additional information related to the tag. These two attributes are standard requirements of all XHTML web pages; the former defines the XML namespace, whereas the latter defines the language of the content. Throughout this book, a standard namespace is defined, and the English language is used. If you are writing in a different language, replace the "en" (for English) with the language identifier relevant to you.

TIP

You might find it convenient to create and save a bare-bones page (also known as a *skeleton* page, or *template*) with just the opening and closing <html>, <head>, <title>, and <body> tags, similar to the document used in Listing 2.1. You can then open that document as a starting point whenever you want to make a new web page and save yourself the trouble of typing all those obligatory tags every time.

Organizing a Page with Paragraphs and Line Breaks

When a web browser displays HTML pages, it pays no attention to line endings or the number of spaces between words. For example, the top version of the poem shown in Figure 2.2 appears with a single space between

all words, even though that's not how it's entered in Listing 2.2. This is because extra whitespace in HTML code is automatically reduced to a single space. Additionally, when the text reaches the edge of the browser window, it automatically wraps to the next line, no matter where the line breaks were in the original HTML file.

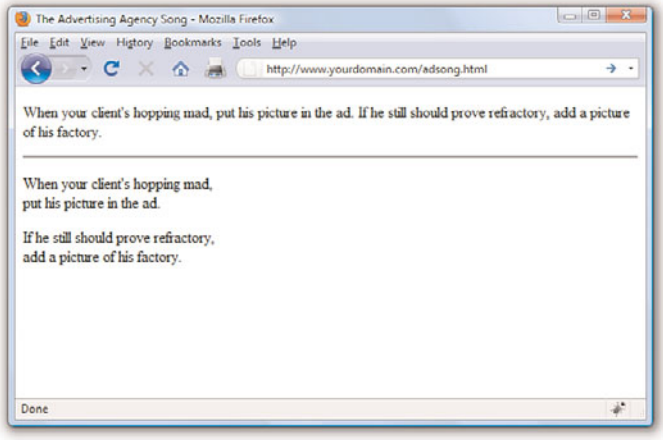


FIGURE 2.2
When the HTML in Listing 2.2 is viewed as a web page, line and paragraph breaks only appear where there are
 and <p> tags.

LISTING 2.2 HTML Containing Paragraph and Line Breaks

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>The Advertising Agency Song</title>
  </head>

  <body>
    <p>
      When your client's    hopping mad,
      put his picture in the ad.

      If he still should    prove refractory,
      add a picture of his factory.

    </p>

    <hr />

    <p>
      When your client's hopping mad,<br />
      put his picture in the ad.
```

LISTING 2.2 Continued

```
    </p>
    <p>
      If he still should prove refractory,<br />
      add a picture of his factory.
    </p>
  </body>
</html>
```

You must use HTML tags if you want to control where line and paragraph breaks actually appear. When text is enclosed within the container tags, a line break will be assumed after the closing tag. In later chapters, you will learn to control the height of the line break using CSS. The
 tag forces a line break within a paragraph. Unlike the other tags you've seen so far,
 doesn't require a closing </br> tag—this is one of those empty tags discussed earlier. Although HTML 4 does not require the / in empty tags, XHTML does and future standards will, so it's important for you to stick to the latest standards and create web pages that are coded properly. Always code empty tags so that they end with />.

The poem in Listing 2.2 and Figure 2.2 shows the
 and tags being used to separate the lines and verses of an advertising agency song. You might have also noticed the <hr /> tag in the listing, which causes a horizontal rule line to appear on the page (see Figure 2.2). Inserting a horizontal rule with the <hr /> tag also causes a line break, even if you don't include a
 tag along with it. Like
, the <hr /> horizontal rule tag is an empty tag and therefore never gets a closing </hr> tag.

CAUTION

You might come across a lot of web content that includes
 instead of
. Or you might see other content that does not include the closing tag. Just remember there is a lot of antiquated web content floating around the Internet, and just because you see it in use doesn't mean it's correct. Save yourself a lot of future work and frustration by adhering to the standards you learn in this book. Developing clean HTML coding habits is a very important part of becoming a successful web designer.

TRY IT YOURSELF ▼

Formatting Text in HTML

Take a passage of text and try your hand at formatting it as proper HTML.

1. Add <html><head><title>My Title</title></head><body> to the beginning of the text (using your own title for your page instead of My Title). Also include the boilerplate code at the top of the page that takes care of meeting the requirements of XHTML.
2. Add </body></html> to the very end of the text.
3. Add a tag at the beginning of each paragraph and a tag at the end of each paragraph.
4. Use
 tags anywhere you want single-spaced line breaks.
5. Use <hr /> to draw horizontal rules separating major sections of text, or wherever you'd like to see a line across the page.

▼ TRY IT YOURSELF

Formatting Text in HTML

continued

CAUTION

If you are using a word processor to create the web page, be sure to save the HTML file in plain-text or ASCII format.

6. Save the file as mypage.html (using your own filename instead of mypage).
7. Open the file in a web browser to see your web content. (Send the file via FTP to your web hosting account, if you have one.)
8. If something doesn't look right, go back to the text editor to make corrections and save the file again (and send it to your web hosting account, if applicable). You then need to click Reload/Refresh in the browser to see the changes you made.

Organizing Your Content with Headings

When you browse through web pages on the Internet, you'll notice that many of them have a heading at the top that appears larger and bolder than the rest of the text. Listing 2.3 is sample code and text for a simple web page containing an example of a heading as compared to normal paragraph text. Any text between `<h1>` and `</h1>` tags will appear as a large heading. Additionally, `<h2>` and `<h3>` make progressively smaller headings, and so on as far down as `<h6>`.

LISTING 2.3 Heading Tags

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>My Widgets</title>
  </head>

  <body>
    <h1>My Widgets</h1>
    <p>My widgets are the best in the land. Continue reading to
    learn more about my widgets.</p>

    <h2>Widget Features</h2>
    <p>If I had any features to discuss,  you can bet I'd do
    it here.</p>

    <h3>Pricing</h3>
    <p>Here, I would talk about my widget pricing.</p>
```

LISTING 2.3 Continued

```
    <h3>Comparisons</h3>
    <p>Here, I would talk about how my widgets compare to my
    competitor's widgets.</p>

  </body>
</html>
```

As you can see in Figure 2.3, the HTML that creates headings couldn't be simpler. In this example, the phrase "My Widgets" is prominently displayed using the `<h1>` tag. To create the biggest (level 1) heading, just put an `<h1>` tag at the beginning and a `</h1>` tag at the end of the text you want to use as a heading. For a slightly smaller (level 2) heading—for information that is of lesser importance than the title— use the `<h2>` and `</h2>` tags around your text. For content that should appear even less prominently than a level 2 heading, use the `<h3>` and `</h3>` tags around your text.

However, bear in mind that your headings should follow a content hierarchy; use only one level 1 heading, have one (or more) level 2 headings after the level 1 heading, use level 3 headings directly after level 2 headings, and so on. Do not fall into the trap of assigning headings to content just to make that content display a certain way. Instead, ensure that you are categorizing your content appropriately (as a main heading, a secondary heading, and so on), while using display styles to make that text render a particular way in a web browser.

NOTE

By now you've probably caught on to the fact that HTML code is often indented by its author to reveal the relationship between different parts of the HTML document. This indentation is entirely voluntary—you could just as easily run all the tags together with no spaces or line breaks and they would still look fine when viewed in a browser. The indentations are for you so that you can quickly look at a page full of code and understand how it fits together. Indenting your code is a very good web design habit and ultimately makes your pages easier to maintain.

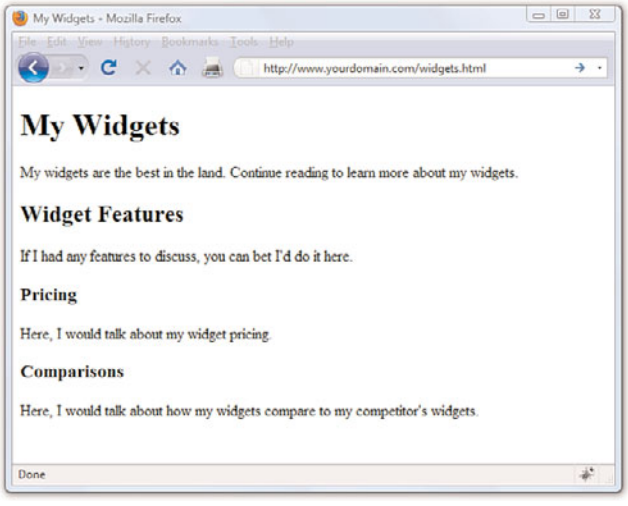


FIGURE 2.3
The use of three levels of headings shows the hierarchy of content on this sample product page.

NOTE

On many web pages nowadays, graphical images of ornately rendered letters and logos are often used in place of the ordinary text headings discussed in this chapter. However, using text headings is one of many search engine optimization (SEO) tips that you will learn about in Chapter 28, "Helping People Find Your Web Pages." Search engines look at heading tags to see how you organize your content; they give higher preference to content that you have indicated is more important (for example, a level 1 heading) versus content that you indicate is of lesser importance (lower-level headings).

CAUTION

Don't forget that anything placed in the head of a web page is not intended to be viewed on the page, whereas everything in the body of the page is intended for viewing.

Theoretically, you can also use `<h4>`, `<h5>`, and `<h6>` tags to make progressively less important headings, but these aren't used very often. Web browsers seldom show a noticeable difference between these headings and the `<h3>` headings anyway, and content usually isn't displayed in such a manner as to need six levels of headings to show the content hierarchy.

It's important to remember the difference between a *title* and a *heading*. These two words are often interchangeable in day-to-day English, but when you're talking HTML, `<title>` gives the entire page an identifying name that isn't displayed on the page itself; it's displayed only on the browser window's title bar. The heading tags, on the other hand, cause some text on the page to be displayed with visual emphasis. There can be only one `<title>` per page and it must appear within the `<head>` and `</head>` tags, whereas you can have as many `<h1>`, `<h2>`, and `<h3>` headings as you want, in any order that suits your fancy. However, as I mentioned before, you should use the heading tags to keep tight control over content hierarchy; do not use headings as a way to achieve a particular look because that's what CSS is for.

You'll learn to take complete control over the appearance of text on your web pages in Parts II and III of this book. Short of taking exacting control of the size, family, and color of fonts, headings provide the easiest and most popular way to draw extra attention to important text.

Validating Your Web Content

In the first chapter, I discussed ways to test your pages; one very important way to test your pages is to *validate* them. Think of it this way: It's one thing to design and draw a beautiful set of house plans, but it's quite another for an architect to stamp it as a safe structure suitable for construction. Validating your web pages is a similar process; in this case, however, the architect is an application—not a person.

In brief, validation is the process of testing your pages with a special application that searches for errors and makes sure your pages follow the strict XHTML standard. Validation is simple. In fact, the standards body responsible for developing web standards—the World Wide Web Consortium (W3C)—offers an online validation tool you can use. To validate a page, follow this URL: http://validator.w3.org/. The W3C Markup Validation Service is shown in Figure 2.4.

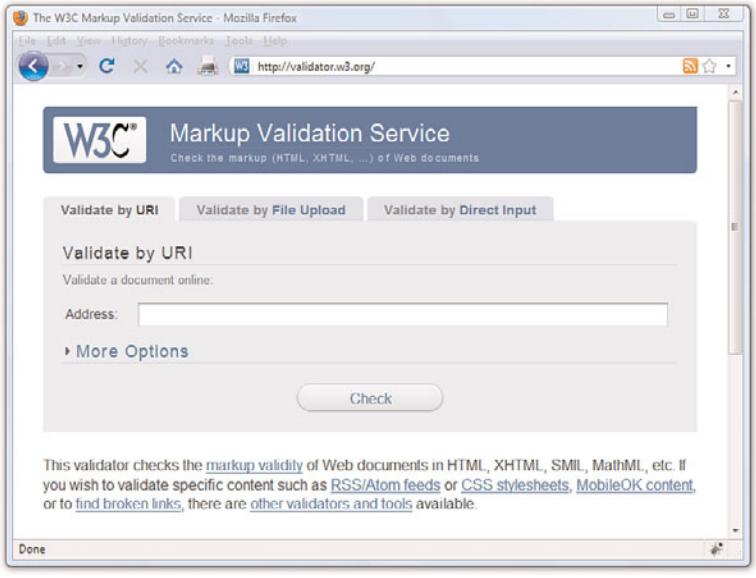


FIGURE 2.4
The W3C Markup Validation Service enables you to validate an HTML (XHTML) document to ensure it has been coded accurately.

If you've already published a page online, you can use the Validate by URI tab. Use the Validate by File Upload tab to validate files stored on your local computer file system. The Validate by Direct Input tab enables you to paste the contents of a file from your text editor. If all goes well, your page will get a passing report (see Figure 2.5).

If the W3C Markup Validation Service encounters an error in your web page, it will provide specific details (including the line numbers of the offending code). This is a great way to hunt down problems and rid your pages of buggy code. Validation not only informs you whether your pages are constructed properly, it also assists you in finding and fixing problems before you post pages for the world to see.

Peeking at Other Designers' Pages

Given the visual and sometimes audio pizzazz present in many popular web pages, you probably realize that the simple pages described in this chapter are only the tip of the HTML iceberg. Now that you know the basics, you might surprise yourself with how much of the rest you can pick up just by looking at other people's pages on the Internet. You can see the HTML for any page by right-clicking and selecting View Source in any web browser.

Don't worry if you aren't yet able to decipher what some HTML tags do or exactly how to use them yourself. You'll find out about all those things in the next few chapters. However, sneaking a preview now will show you the tags that you do know in action and give you a taste of what you'll soon be able to do with your web pages.

TIP

Some web development tools include built-in validation features you can use in lieu of the W3C Markup Validation Service. Some examples include browser extensions such as Firebug (http://getfirebug.com/) and HTML Validator (http://users.skynet.be/mgueury/mozilla/), but many other programs offer similar functionality; check your user documentation.

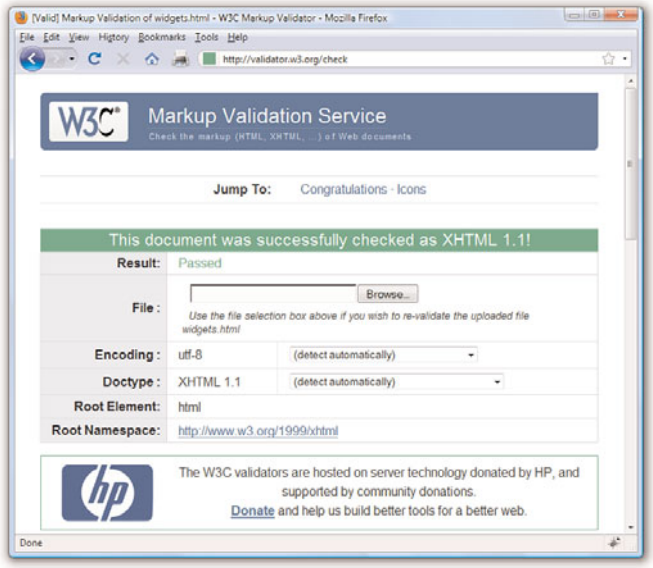


FIGURE 2.5
If a page passes the W3C Markup Validation Service, you know it is ready for prime time.

The Scoop on HTML, XML, XHTML, and HTML5

In its early days, HTML was great because it allowed scientists to share information over the Internet in an efficient and relatively structured manner. It wasn't until later that graphical web browsers were created and HTML started being used to code more than scientific papers. HTML quickly went from a tidy little markup language for researchers to an online publishing language. After it was established that HTML could be jazzed up for graphical browsing, the creators of web browsers went crazy by adding lots of nifty features to the language. Although these new features were neat at first, they compromised the original design of HTML and introduced inconsistencies when it came to how browsers displayed web pages; new features worked on only one browser or another, and you were out of luck if you happened to be running the wrong browser. HTML started to resemble a bad remodeling job of a house—a job done by too many contractors and without proper planning. As it turns out, some of the browser-specific features created during this time have now been adopted as standards whereas others have been dropped completely.

As with most revolutions, the birth of the Web was very chaotic, and the modifications to HTML reflected that chaos. Over the years, a significant

effort has been made to reel in the inconsistencies of HTML and restore some order to the language. The problem with disorder in HTML is that it results in web browsers having to guess at how a page is to be displayed, which is not a good thing. Ideally, a web page designer should be able to define exactly how a page is to look and have it look the same regardless of what kind of browser or operating system someone is using. Better still, a designer should be able to define exactly what a page *means* and have that page look consistent across different browsers and platforms. This utopia is still off in the future somewhere, but a markup language called XML (Extensible Markup Language) began to play a significant role in leading us toward it.

XML is a general language used to create specific languages, such as HTML. It might sound a little strange, but it really just means that XML provides a basic structure and set of rules to which any markup language must adhere. Using XML, you can create a unique markup language to describe just about any kind of information, including web pages. Knowing that XML is a language for creating other markup languages, you could create your own version of HTML using XML. You could even create a markup language called BCCML (Bottle Cap Collection Markup Language), for example, which you could use to create and manage your extensive collection of rare bottle caps. The point is that XML lays the ground rules for organizing information in a consistent manner, and that information can be anything from web pages to bottle caps.

You might be thinking that bottle caps don't have anything to do with the Web, so why mention them? The reason is that XML is not entirely about web pages. XML is actually broader than the Web in that it can be used to represent any kind of information on any kind of computer. If you can visualize all the information whizzing around the globe among computers, mobile phones, handheld computers, televisions, and radios, you can start to understand why XML has much broader applications than just cleaning up web pages. However, one of the first applications of XML is to restore some order to the Web, which is why XML is relevant to learning HTML.

If XML describes data better than HTML, does it mean that XML is set to upstage HTML as the markup language of choice for the Web? No. XML is not a replacement for HTML; it's not even a competitor of HTML. XML's impact on HTML has to do with cleaning up HTML. HTML is a relatively unstructured language that benefits from the rules of XML. The natural merger of the two technologies resulted in HTML's adherence to the rules and structure of XML. To accomplish this merger, a new version of HTML was

formulated that follows the stricter rules of XML. The new XML-compliant version of HTML is known as XHTML. Fortunately for you, you'll actually be learning XHTML throughout this book because it is really just a cleaner version of HTML.

You might have heard about HTML5, which is touted as the next web standard. It will be, but not quite yet. When it does become a web standard, it will not render XHTML useless—HTML5 is not a replacement for XHTML, but instead is a major revision of HTML 4. In other words, XHTML and HTML5 can coexist on the Web, and web browsers that currently support XHTML will also (one day) support HTML5 as well.

The goal of this book is to guide you through the basics of web publishing, using XHTML and CSS as the core languages of those pages. However, whenever possible, I will note elements of the languages that are not present in HTML5, should you want to design your content for even further sustainability. If you gain a solid understanding of web publishing and the ways in which CSS works with the overall markup language of the page (be it XHTML or HTML5), you will be in a good position if you decide you want to move from XHTML to HTML5.

Summary

This chapter introduced the basics of what web pages are and how they work, including the history and differences between HTML and XHTML. You learned that coded HTML commands are included in a text file, and that typing HTML text yourself is better than using a graphical editor to create HTML commands for you—especially when you're learning HTML.

You were introduced to the most basic and important HTML tags. By adding these coded commands to any plain-text document, you can quickly transform it into a bona fide web page. You learned that the first step in creating a web page is to put a few obligatory HTML tags at the beginning and end, including a title for the page. You then mark where paragraphs and lines end and add horizontal rules and headings if you want them. Table 2.1 summarizes all the tags introduced in this chapter.

TABLE 2.1 HTML Tags Covered in Chapter 2

Tag	Function
`<html>...</html>`	Encloses the entire HTML document.
`<head>...</head>`	Encloses the head of the HTML document. Used within the `<html>` tag pair.
`<title>...</title>`	Indicates the title of the document. Used within the `<head>` tag pair.
`<body>...</body>`	Encloses the body of the HTML document. Used within the `<html>` tag pair.
`<p>...</p>`	A paragraph; skips a line between paragraphs.
` `	A line break.
`<hr />`	A horizontal rule line.
`<h1>...</h1>`	A first-level heading.
`<h2>...</h2>`	A second-level heading.
`<h3>...</h3>`	A third-level heading.
`<h4>...</h4>`	A fourth-level heading (seldom used).
`<h5>...</h5>`	A fifth-level heading (seldom used).
`<h6>...</h6>`	A sixth-level heading (seldom used).

Finally, you learned about XML and XHTML, how they relate to HTML, and what HTML5 means in relation to what it is you're learning here.

Q&A

Q. **I've created a web page, but when I open the file in my web browser, I see all the text including the HTML tags. Sometimes I even see weird gobbledygook characters at the top of the page! What did I do wrong?**

A. You didn't save the file as plain text. Try saving the file again, being careful to save it as Text Only or ASCII Text. If you can't quite figure out how to get your word processor to do that, don't stress. Just type your HTML files in Notepad or TextEdit instead and everything should work just fine. (Also, always make sure that the filename of your web page ends in .html or .htm.)

Q. **I've seen web pages on the Internet that don't have `<html>` tags at the beginning. You said pages always have to start with `<html>`. What's the deal?**

A. Many web browsers will forgive you if you forget to include the `<html>` tag and will display the page correctly anyway. However, it's a very good idea to include it because some software does need it to identify the page as valid HTML. Besides, you want your pages to be bona fide XHTML pages so that they conform to the latest web standards.

Workshop

The workshop contains quiz questions and exercises to help you solidify your understanding of the material covered. Try to answer all questions before looking at the "Answers" section that follows.

Quiz

1. What four tags are required in every HTML page?
2. What HTML tags and text would you use to produce the following web content:
   - A small heading with the words `We are Proud to Present`
   - A horizontal rule across the page
   - A large heading with the one word `Orbit`
   - A medium-sized heading with the words `The Geometric Juggler`
   - Another horizontal rule

3. What code would you use to create a complete HTML web page with the title `Foo Bar`, a heading at the top that reads `Happy Hour at the Foo Bar`, followed by the words `Come on down!` in regular type?

Answers

1. `<html>`, `<head>`, `<title>`, and `<body>` (along with their closing tags, `</html>`, `</head>`, `</title>`, and `</body>`).

2. Your code would look like this:

```
<h3>We are Proud to Present</h3>
<hr />
<h1>Orbit</h1>
<h2>The Geometric Juggler</h2>
<hr />
```

3. Your code would look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Foo Bar</title>
  </head>

  <body>
    <h1>Happy Hour at the Foo Bar</h1>
    <p>Come on Down!</p>
  </body>
</html>
```

Exercises

- Even if your main goal in reading this book is to create web content for your business, you might want to make a personal web page just for practice. Type a few paragraphs to introduce yourself to the world and use the HTML tags you've learned in this chapter to make them into a web page.
- Throughout the book, you'll be following along with the code examples and making pages of your own. Take a moment now to set up a basic document template containing the XML declaration, doctype declaration, and tags for the core HTML document structure. That way, you can be ready to copy and paste that information whenever you need it.

This page intentionally left blank

CHAPTER 3
Understanding Cascading Style Sheets

In the previous chapter, you learned the basics of HTML and XHTML, including how to set up a skeletal HTML template for all your web content. In this chapter, you will learn how to fine-tune the display of your web content using *Cascading Style Sheets (CSS).*

The concept behind style sheets is simple: You create a style sheet document that specifies the fonts, colors, spacing, and other characteristics that establish a unique look for a website. You then link every page that should have that look to the style sheet, instead of specifying all those styles repeatedly in each separate document. Therefore, when you decide to change your official corporate typeface or color scheme, you can modify all your web pages at once just by changing one or two entries in your style sheet rather than changing them in all of your static web files. So, a *style sheet* is a grouping of formatting instructions that controls the appearance of several HTML pages at once.

Style sheets enable you to set a great number of formatting characteristics, including exacting typeface controls, letter and line spacing, and margins and page borders, just to name a few. Style sheets also enable sizes and other measurements to be specified in familiar units, such as inches, millimeters, points, and picas. You can also use style sheets to precisely position graphics and text anywhere on a web page, either at specific coordinates or relative to other items on the page.

In short, style sheets bring a sophisticated level of display to the Web. And they do so—you'll pardon the expression—with style.

WHAT YOU'LL LEARN IN THIS CHAPTER:

- How to create a basic style sheet
- How to use style classes
- How to use style IDs
- How to construct internal style sheets and inline styles

NOTE

If you have three or more web pages that share (or should share) similar formatting and fonts, you might want to create a style sheet for them as you read this chapter. Even if you choose not to create a complete style sheet, you'll find it helpful to apply styles to individual HTML elements directly within a web page.

How CSS Works

The technology behind style sheets is called CSS, which stands for Cascading Style Sheets. CSS is a language that defines style constructs such as fonts, colors, and positioning, which are used to describe how information on a web page is formatted and displayed. CSS styles can be stored directly in an HTML web page or in a separate style sheet file. Either way, style sheets contain style rules that apply styles to elements of a given type. When used externally, style sheet rules are placed in an external style sheet document with the file extension .css.

A *style rule* is a formatting instruction that can be applied to an element on a web page, such as a paragraph of text or a link. Style rules consist of one or more *style properties* and their associated values. An *internal style sheet* is placed directly within a web page, whereas an *external style sheet* exists in a separate document and is simply linked to a web page via a special tag—more on this tag in a moment.

NOTE

You might notice that I use the term *element* a fair amount in this chapter (and I will for the rest of the book, for that matter). An *element* is simply a piece of information (content) in a web page, such as an image, a paragraph, or a link. Tags are used to code elements, and you can think of an element as a tag complete with descriptive information (attributes, text, images, and so on) within the tag.

The *cascading* part of the name CSS refers to the manner in which style sheet rules are applied to elements in an HTML document. More specifically, styles in a CSS style sheet form a hierarchy in which more specific styles override more general styles. It is the responsibility of CSS to determine the precedence of style rules according to this hierarchy, which establishes a cascading effect. If that sounds a bit confusing, just think of the cascading mechanism in CSS as being similar to genetic inheritance, in which general traits are passed from parents to a child, but more specific traits are entirely unique to the child. Base style rules are applied throughout a style sheet but can be overridden by more specific style rules.

A quick example should clear things up. Take a look at the following code to see whether you can tell what's going on with the color of the text:

```
<div style="color:green">
  This text is green.
  <p style="color:blue">This text is blue.</p>
  <p>This text is still green.</p>
</div>
```

In the previous example, the color green is applied to the `<div>` tag via the `color` style property. Therefore, the text in the `<div>` tag is colored green. Because both `<p>` tags are children of the `<div>` tag, the green text style

cascades down to them. However, the first <p> tag overrides the color style and changes it to blue. The end result is that the first line (not surrounded by a paragraph tag) is green, the first official paragraph is blue, and the second official paragraph retains the cascaded green color.

If you made it through that description on your own, congratulations. If you understood it after I explained it in the text, congratulations to you as well. Understanding CSS isn't like understanding rocket science, although many people will try to convince you that it is (so that they can charge high consultation fees, most likely!).

Like many web technologies, CSS has evolved over the years. The original version of CSS, known as *Cascading Style Sheets Level 1* (*CSS1)* was created in 1996. The later CSS 2 standard was created in 1998, and CSS 2 is still in use today. All modern web browsers support CSS 2, and you can safely use CSS 2 style sheets without too much concern. So when I talk about CSS throughout the book, I'm referring to CSS 2.

You'll find a complete reference guide to CSS at http://www.w3.org/Style/CSS/. The rest of this chapter explains how to put CSS to good use.

A Basic Style Sheet

Despite their intimidating power, style sheets can be simple to create. Consider the web pages shown in Figure 3.1 and Figure 3.2. These pages share several visual properties that could be put into a common style sheet:

- They use a large, bold Verdana font for the headings and a normal size and weight Verdana font for the body text.
- They use an image named logo.gif floating within the content and on the right side of the page.
- All text is black except for subheadings, which are purple.
- They have margins on the left side and at the top.
- There is vertical space between lines of text.
- The footnotes are centered and in small print.

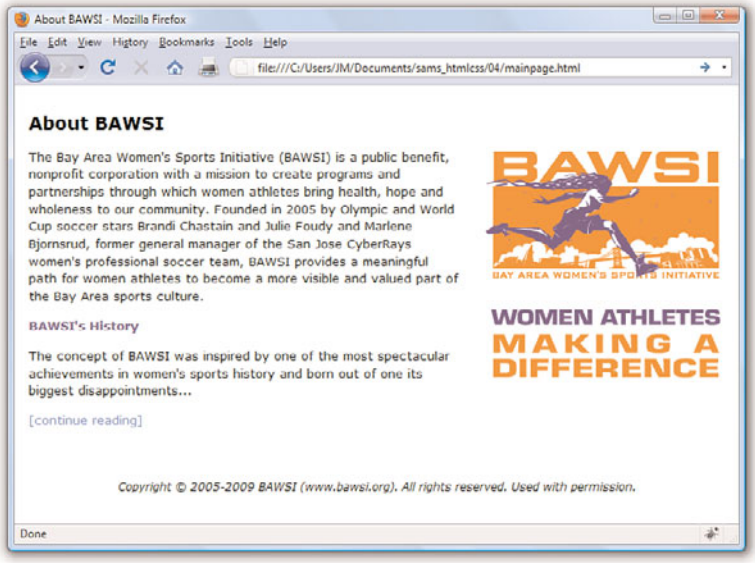


FIGURE 3.1
This page uses a style sheet to fine-tune the appearance and spacing of the text and images.

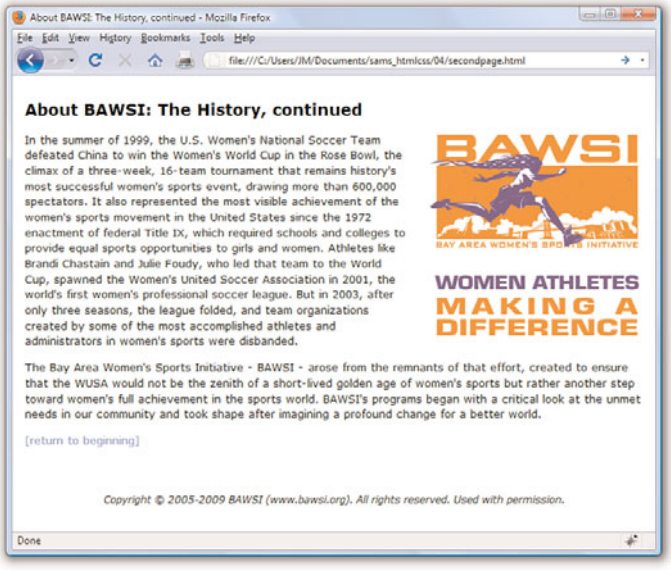


FIGURE 3.2
This page uses the same style sheet as the one shown in Figure 3.1, thus maintaining a consistent look and feel.

Listing 3.1 shows the code for the style sheet specifying these properties.

LISTING 3.1 A Single External Style Sheet

```
body {
  font-size: 10pt;
  font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
  color: black;
  line-height: 14pt;
  padding-left: 5pt;
  padding-right: 5pt;
  padding-top: 5pt;
}

h1 {
  font: 14pt Verdana, Geneva, Arial, Helvetica, sans-serif;
  font-weight: bold;
  line-height: 20pt;
}

p.subheader {
  font-weight: bold;
  color: #593d87;
}

img {
  padding: 3pt;
  float: right;
}

a {
  text-decoration: none;
}

a:link, a:visited {
  color: #8094d6;
}

a:hover, a:active {
  color: #FF9933;
}

div.footer {
  font-size: 9pt;
  font-style: italic;
  line-height: 12pt;
  text-align: center;
  padding-top: 30pt;
}
```

This might initially appear to be a lot of code, but if you look closely, you'll see that there isn't a lot of information on each line of code. It's fairly standard to place individual style rules on their own line to help make style

sheets more readable, but that is a personal preference; you could put all the rules on one line as long as you kept using the semicolon to separate each rule (more on that in a bit). Speaking of code readability, perhaps the first thing you noticed about this style sheet code is that it doesn't look anything like normal HTML code. CSS uses a language all its own to specify style sheets.

Of course, the listing includes some familiar HTML tags. As you might guess, `body`, `h1`, `p`, `img`, `a`, and `div` in the style sheet refer to the corresponding tags in the HTML documents to which the style sheet will be applied. The curly braces after each tag name contain the specifications for how all content within that tag should appear.

In this case, the style sheet says that all `body` text should be rendered at a size of 10 points, in the Verdana font (if possible), with the color black, and 14 points between lines. If the user does not have the Verdana font installed, the list of fonts in the style sheet represents the order in which the browser should search for fonts to use: Geneva, then Arial, and then Helvetica. If the user has none of those fonts, the browser will use whatever default sans serif font is available. Additionally, the page should have left, right, and top margins of 5 points each.

Any text within an `<h1>` tag should be rendered in boldface Verdana at a size of 14 points. Moving on, any paragraph that uses only the `<p>` tag will inherit all the styles indicated by the body element. However, if the `<p>` tag uses a special class named `subheader`, the text will appear bold and in the color #593d87 (a purple color).

NOTE

You can specify font sizes as large as you like with style sheets, although some display devices and printers will not correctly handle fonts larger than 200 points.

The `pt` after each measurement in Listing 3.1 means *points* (there are 72 points in an inch). If you prefer, you can specify any style sheet measurement in inches (`in`), centimeters (`cm`), pixels (`px`), or widths-of-a-letter-m, which are called ems (`em`).

You might have noticed that each style rule in the listing ends with a semicolon (`;`). Semicolons are used to separate style rules from each other. It is therefore customary to end each style rule with a semicolon, so you can easily add another style rule after it.

To link this style sheet to HTML documents, include a `<link />` tag in the `<head>` section of each document. Listing 3.2 shows the HTML code for the page shown in Figure 3.1. It contains the following `<link />` tag:

```
<link rel="stylesheet" type="text/css" href="styles.css" />
```

This assumes that the style sheet is stored under the name `styles.css` in the same folder as the HTML document. As long as the web browser supports style sheets—and all modern browsers do support style sheets—the properties specified in the style sheet will apply to the content in the page without the need for any special HTML formatting code. This confirms the ultimate goal of XHTML, which is to provide a separation between the content in a web page and the specific formatting required to display that content.

LISTING 3.2 HTML Code for the Page Shown in Figure 3.1

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>About BAWSI</title>
    <link rel="stylesheet" type="text/css" href="styles.css" />
  </head>
  <body>
     <h1>About BAWSI</h1>
     <p><img src="logo.gif" alt="BAWSI logo"/>The Bay Area Women's
     Sports Initiative (BAWSI) is a public benefit, nonprofit
     corporation with a mission to create programs and partnerships
     through which women athletes bring health, hope and wholeness to
     our community. Founded in 2005 by Olympic and World Cup soccer
     stars Brandi Chastain and Julie Foudy and Marlene Bjornsrud,
     former general manager of the San Jose CyberRays women's
     professional soccer team, BAWSI provides a meaningful path for
     women athletes to become a more visible and valued part of the
     Bay Area sports culture.</p>
     <p class="subheader">BAWSI's History</p>
     <p>The concept of BAWSI was inspired by one of the most
     spectacular achievements in women's sports history and born out
     of one its biggest disappointments... </p>
     <p><a href="secondpage.html">[continue reading]</a></p>
     <div class="footer">Copyright &copy; 2005-2009 BAWSI
    (www.bawsi.org). All rights reserved.  Used with permission.</div>
  </body>
</html>
```

The code in Listing 3.2 is interesting because it contains no formatting of any kind. In other words, there is nothing in the HTML code that dictates how the text and images are to be displayed—no colors, no fonts, nothing. Yet the page is carefully formatted and rendered to the screen, thanks to the link to the external style sheet, styles.css. The real benefit to this

TIP

In most web browsers, you can view the style rules in a style sheet by opening the .css file and choosing Notepad or another text editor as the helper application to view the file. (To determine the name of the .css file, look at the HTML source of any web page that links to it.) To edit your own style sheets, just use a text editor.

NOTE

Although CSS is widely supported in all modern web browsers, it hasn't always enjoyed such wide support. Additionally, not every browser's support of CSS is flawless. To find out about how major browsers compare to each other in terms of CSS support, take a look at this website: http://www.quirksmode.org/css/contents.html.

approach is that you can easily create a site with multiple pages that maintains a consistent look and feel. And you have the benefit of isolating the visual style of the page to a single document (the style sheet) so that one change impacts all pages.

▼ TRY IT YOURSELF

Create a Style Sheet of Your Own

Starting from scratch, create a new text document called mystyles.css and add some style rules for the following basic HTML tags: `<body>`, `<p>`, `<h1>`, and `<h2>`. After your style sheet has been created, make a new HTML file that contains these basic tags. Play around with different style rules and see for yourself how simple it is to change entire blocks of text in paragraphs with one simple change in a style sheet file.

A CSS Style Primer

You now have a basic knowledge of CSS style sheets and how they are based on style rules that describe the appearance of information in web pages. The next few sections of this chapter provide a quick overview of some of the most important style properties and allow you to get started using CSS in your own style sheets.

CSS includes various style properties that are used to control fonts, colors, alignment, and margins, to name just a few. The style properties in CSS can be generally grouped into two major categories:

- **Layout properties**—Consist of properties that affect the positioning of elements on a web page, such as margins, padding, alignment, and so on
- **Formatting properties**—Consist of properties that affect the visual display of elements within a website, such as the font type, size, color, and so on

Layout Properties

CSS layout properties are used to determine how content is placed on a web page. One of the most important layout properties is the `display` property, which describes how an element is displayed with respect to other elements. There are four possible values for the `display` property: