CHAPTER 1
Understanding How the Web Works

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

Before you learn the intricacies of HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), and JavaScript—not to mention the back-end programming language PHP—it is important to gain a solid understanding of the technologies that help transform these plain-text files to the rich multimedia displays you see on your computer, tablet, or smartphone when browsing the World Wide Web.

For example, a file containing markup and client-side code (HTML, CSS, and JavaScript) is useless without a web browser to view it, and no one besides yourself will see your content unless a web server is involved—this is especially true when server-side technologies such as PHP are put into the mix. 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 web server to make the content available in the first place, and you must ensure that your content will appear to the end user as you intended.

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. 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 exchange data and documents. 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.

This wasn't a new idea, but his simple Hypertext Markup Language (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.

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 enable people to view web content. 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 web technologies including HTML, CSS, and JavaScript have led us to the world we live in today, one in which more than a billion websites serve trillions (or more) of text and multimedia files.

NOTE

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

These few paragraphs really are a brief history of what has been a remarkable period. Today's college students have never known a time in which the World Wide 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 dynamic web content creation and management as a set of skills possessed by only 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* with, or surrounded by, HTML codes that tell the browser how to display the text—as a heading, as a paragraph, in a bulleted list, 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 this 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'll learn the basics of linking to or creating the various types of multimedia web content found in websites, and for creating dynamic content from server-side scripts using PHP. 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 code that tells 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 markup. In this book, you'll learn about and compose standards-compliant HTML5 markup. One of the many benefits of writing standards-compliant code is that, in the future, you will not have to worry about having to go back to your code to fundamentally alter it so that it works on multiple types of browsers and devices. Instead, your code will (likely) always work as intended for as long as web browsers adhere to standards and the backwards compatibility to previous standards (which is 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 happen 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.

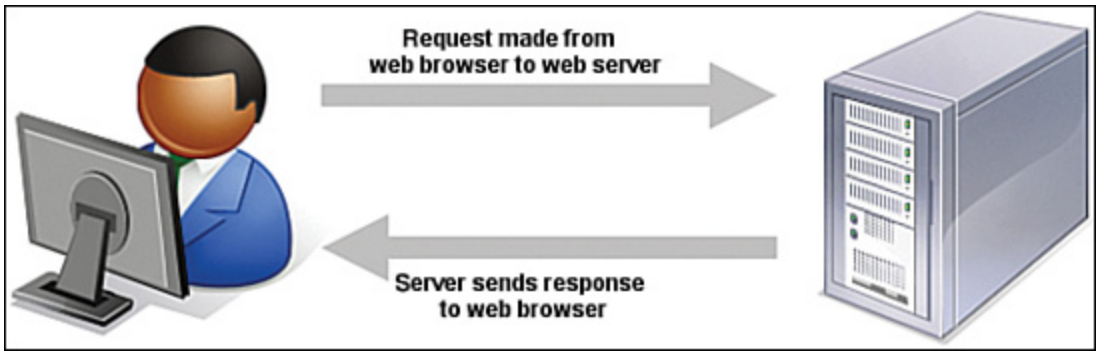

FIGURE 1.1
A browser request and a server response.

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

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

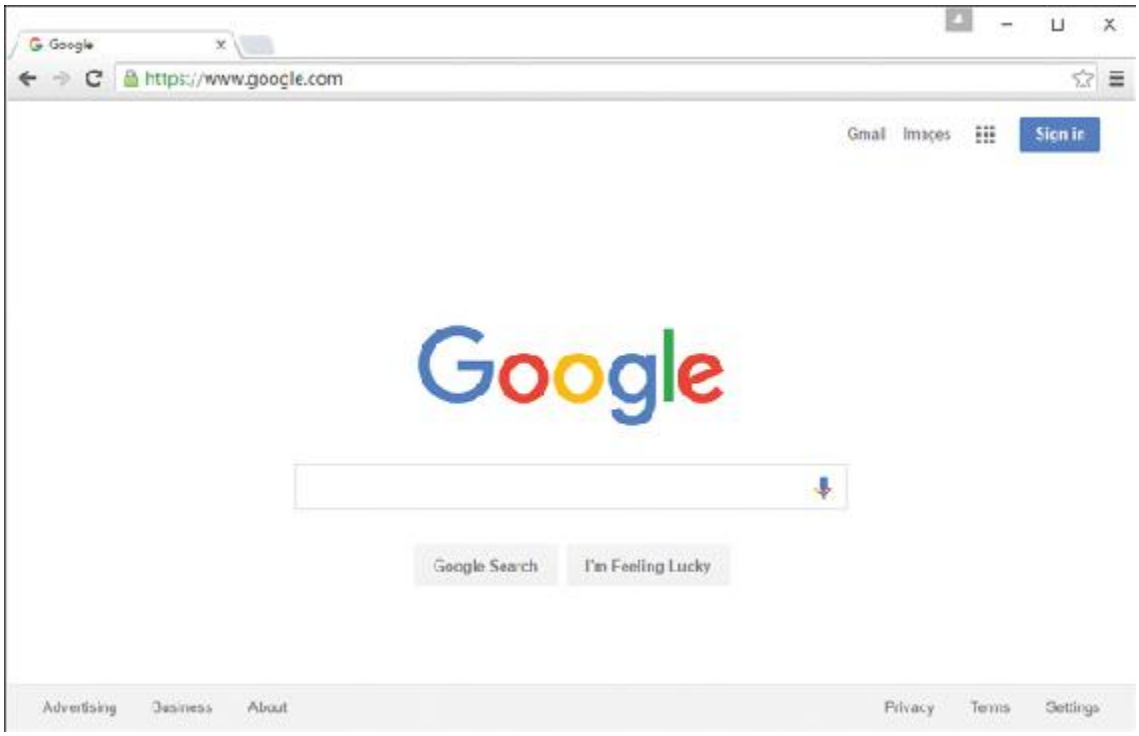

FIGURE 1.2
Visiting www.google.com.

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 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 farther 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 something like this:

Click here to view code image

```
<img alt="Google" height="92" width="272" id="hplogo"
src="/images/branding/
googlelogo/2x/googlelogo_color_272x92dp.png">
```

The HTML code for the image is an `<img>` tag, and it also provides attributes that tell the browser the file source location (`src`), width (`width`), and height (`height`) necessary to display the logo. You'll learn more about attributes throughout later lessons.

4. The browser looks at the `src` attribute in the `<img>` tag to find the source location. In this case, the image `googlelogo_color_272x92dp.png` can be found in a subdirectory of the `images` 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 web address http://www.google.com/images/branding/googlelogo/2x/googlelogo_color_272x92dp.png.
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 would be 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 HTML, CSS, and JavaScript lessons 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 my just telling you that you can work through all the HTML, CSS, and JavaScript lessons in this book without having a web server, having a web server is the recommended method for continuing. Although the appendixes describe how to install a full-blown web server and database on your local machine for personal development, invariably you will want your static or dynamic websites to be visible to the public. 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*). Not this many web hosting providers exist in the world, although it might seem otherwise. 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 follow:

- **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, 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, video) you plan to include in your website (if any)?
- **Bandwidth**—Does the hosting package include enough bandwidth that all the people visiting your site and downloading files can do so without your 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. If you see a wide range of prices offered, you 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 it 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.

NOTE

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

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

One feature of a good hosting provider is that it offers a "control panel" for you to manage aspects of your account. Figure 1.3 shows the control panel for my own hosting account at DailyRazor. 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.

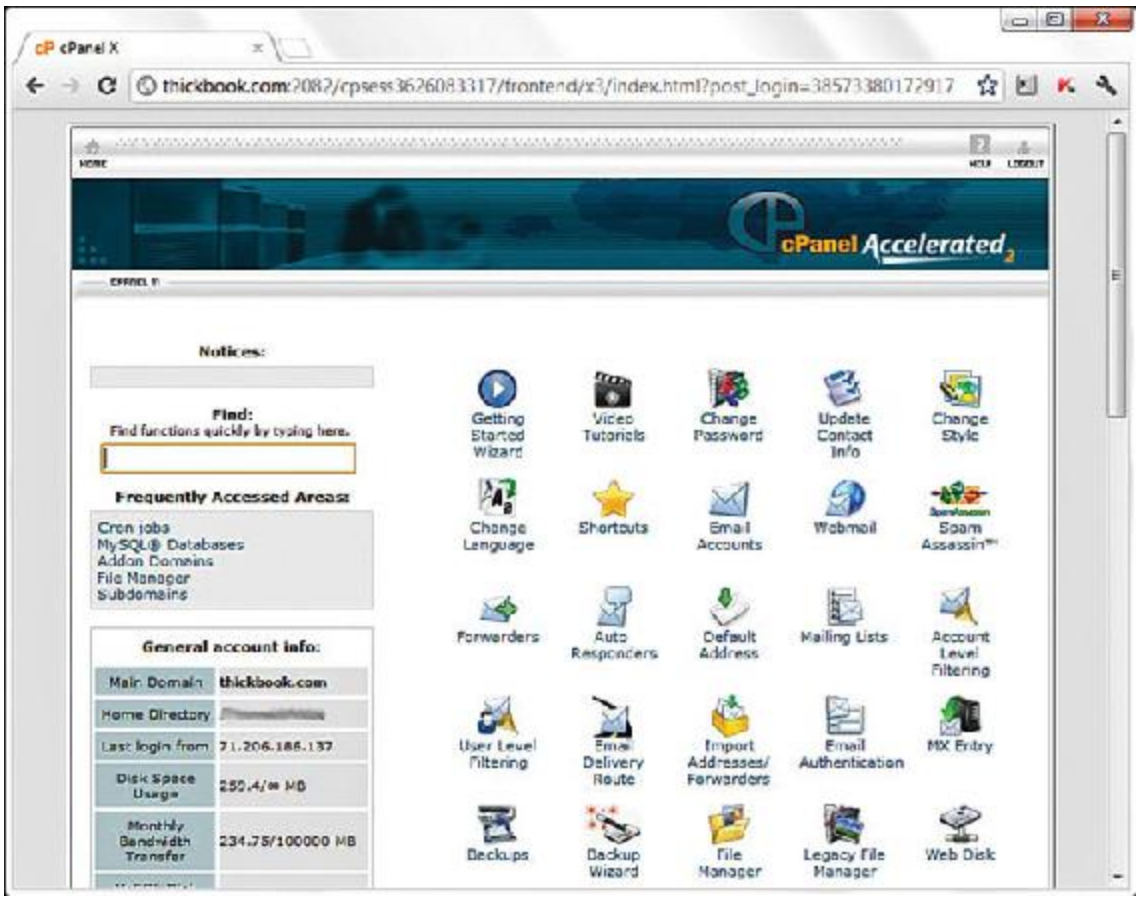

FIGURE 1.3
A sample control panel.

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

Now that we've 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 from your own. From their device types (desktop, laptop, tablet, smartphone) to their screen resolutions, browser types, browser window sizes, and speed of connections—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 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, some specific differences among them 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 and/or changing the size of their viewing windows. All the major web browsers allow users to override the background and fonts the web page author specifies 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 the screen. You can ensure only that you write standards-compliant HTML and CSS.

NOTE

In Chapter 3, "Understanding the CSS Box Model and Positioning," you'll learn a little bit about the concept of responsive web design, in which the design of a site shifts and changes automatically depending on the user's behavior and viewing environment (screen size, device, and so on).

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

You should always test your websites with as many of these web browsers as possible, on standard, portable, and mobile devices:

- Apple Safari (http://www.apple.com/safari/) for Mac
- Google Chrome (http://www.google.com/chrome) for Mac, Windows, and Linux/UNIX
- Microsoft Internet Explorer (http://www.microsoft.com/ie) and Microsoft Edge (https://www.microsoft.com/microsoft-edge) for Windows
- Mozilla Firefox (http://www.mozilla.com/firefox/) 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. You'll learn more about the HTML and CSS used within this file as you move forward in this book.

LISTING 1.1 Our Sample HTML File

Click here to view code image

```
<!DOCTYPE html>
<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 from 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. Be sure your editor does not change the extension you give it; 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 from it is HTML and will render it appropriately.

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 the *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

When 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.

Many FTP clients are freely available to you, but you can also transfer files via the web-based file management 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 the process of installing an FTP client on your own machine.

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

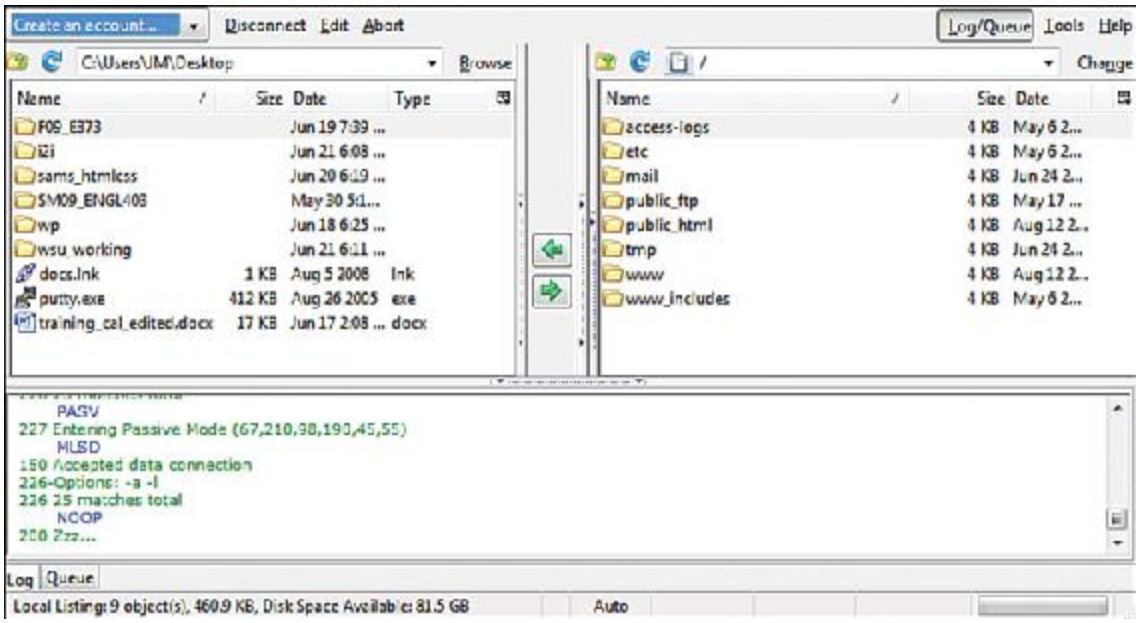

FIGURE 1.4
The FireFTP interface.

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 identical, 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 are prompted to fill out information for the site to which you want to connect, as shown in Figure 1.5.
2. Fill in each of the items shown in Figure 1.5 as described here:
    - The FTP Server is the FTP address of the web server to which you need to send your content. Your hosting provider will have given

you this address. It probably is *yourdomain*.com, but check the information you received when you signed up for service.

- Complete the User Name field and the Password field using the information your hosting provider gave you.

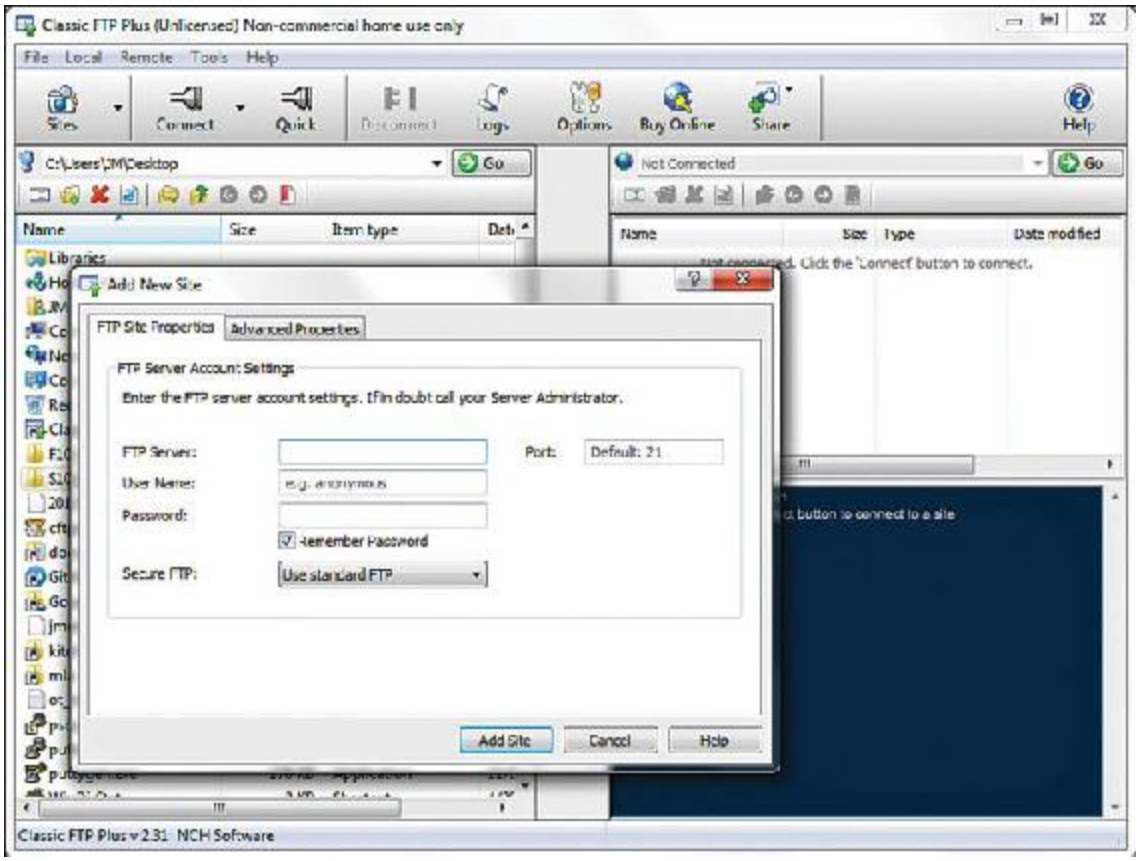

FIGURE 1.5
Connecting to a new site in Classic FTP.

3. You can switch to the Advanced tab and modify the following optional items, shown in Figure 1.6:

   - 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.
   - You can change the values for Initial Remote Directory on First Connection and Initial Local Directory on First Connection, but you might want to wait until you have become accustomed to using the client and have established a workflow.

4. When you're finished with the settings, click Add Site to save them. You can then click Connect to 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 a successful connection, you will see an interface like the one in Figure 1.7, showing the contents of the local directory on the left and the contents of your web server on the right.

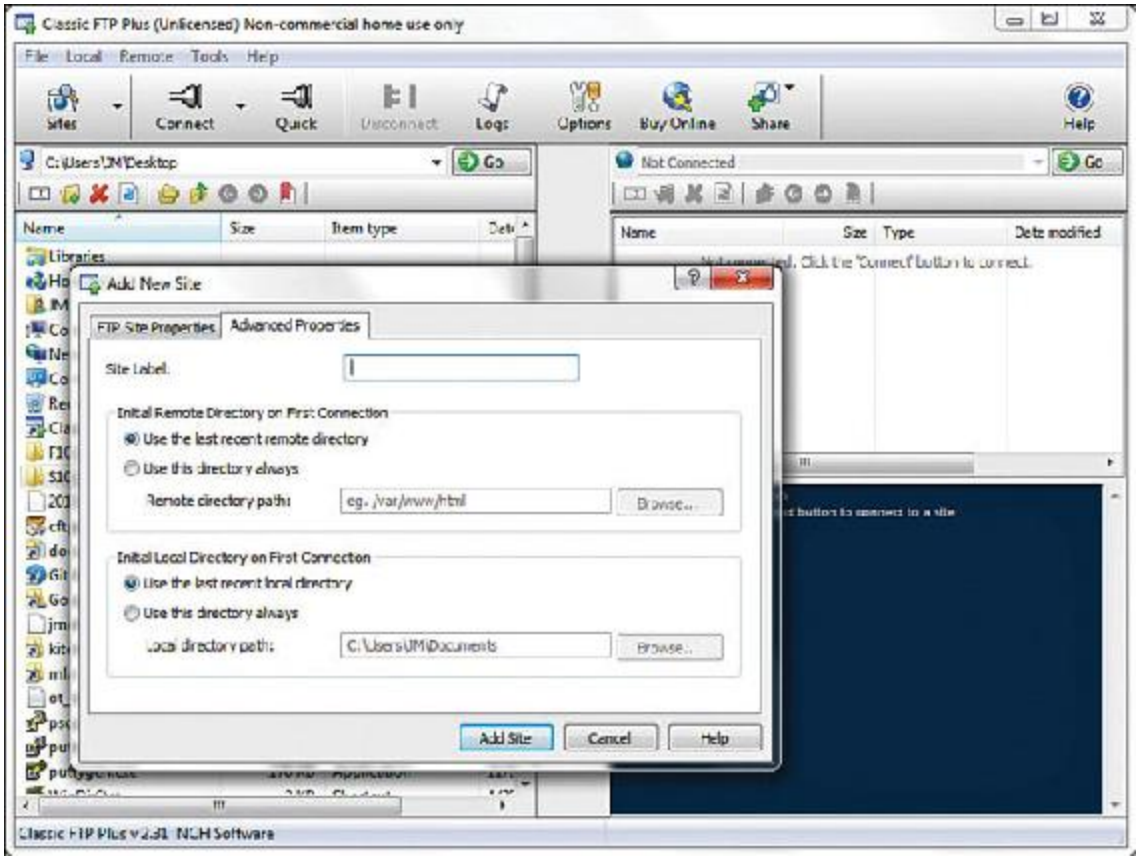

FIGURE 1.6
The advanced connection options in Classic FTP.

5. 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'll learn more about later in this chapter. Often, this directory is named `public_html`, `www` (because `www` has been created as an alias for `public_html`), or `htdocs`. You do not have to create this directory; 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 changes 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).

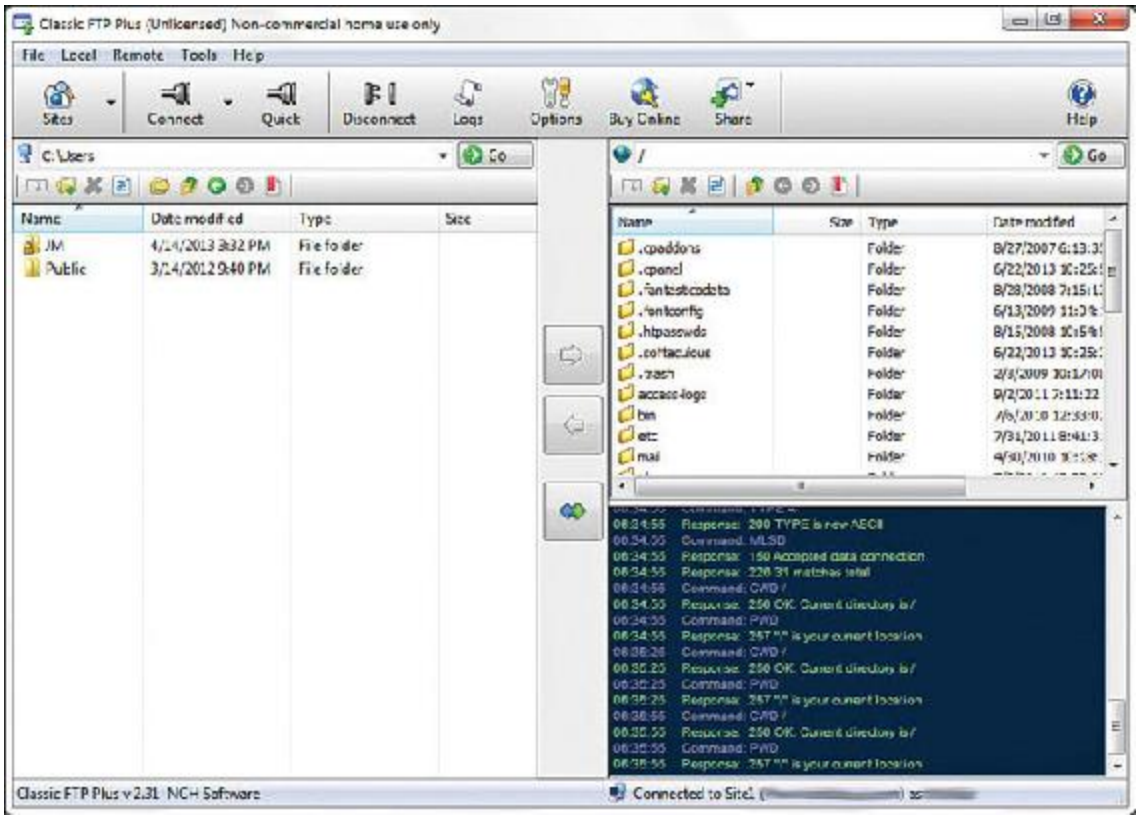

FIGURE 1.7
A successful connection to a remote web server via Classic FTP.

6. 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 if you have to) and click it once to highlight the filename.
7. Click the right-arrow button in the middle of the client interface to send the file to the web server. When the file transfer completes, the right side of the client interface refreshes to show you that the file has made it to its destination.
8. Click the Disconnect button to close the connection, and then exit the Classic FTP program.

These steps are conceptually similar to the steps you 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 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 helps you 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 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 probably changes from

http://www.*companyname*.com/

to

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

or

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

In the preceding 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 entered this URL into your web browser, you would see the rendered `sample.html` file, as shown in Figure 1.8.

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

FIGURE 1.8
The `sample.html` file accessed via a web browser.

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 style sheet files (if you are using more than one) and `js` for JavaScript files. Alternatively, if you know that you will have an area on your website where visitors can download many 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 a file on the Internet that isn't simply a web page. To make available on the Web a file that isn't an HTML file, just upload the file to your website as if it *were* an HTML file, following the instructions 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 Chapter 2, "Structuring HTML and Using Cascading Style Sheets"). In other words, your web server can serve much more than HTML.

Here's a sample of the HTML code 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 with link text that reads `Download my art portfolio!`:

Click here to view code image

```
<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.

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

For instance, you can type either of the following URLs and land on Apple's iPhone informational page:

http://www.apple.com/iphone/

http://www.apple.com/iphone/index.html

Had there been no `index.html` page in the `iphone` 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.

Your hosting provider will already have determined these server configuration options. 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, but 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 ensures that when users type **http://www.*yourdomain*.com/** into their web browsers, the server responds with the content you intended them to see (instead of "Directory Listing Denied" or some other unintended consequence).

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 this content lives on a web server—a remote machine often far 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 to consider when determining whether a web hosting provider fits your needs. After you have selected a web hosting provider, you can begin to transfer files to your web server, which you also learned how to do, using an FTP client. You also learned a 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. In addition, you learned that you can distribute web content on removable media, and you learned 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 helps ensure that 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 sites 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 programming languages—we're saving that for later on in this book, and you'll do just fine with it as well. HTML is a markup language rather than a programming language; you mark up text so that the browser can render the text a certain way. 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 that it was likely created by a visual web design tool—a "what you see is what you get" (WYSIWYG) editor that uses whatever markup its software developer told it to use in certain circumstances—as opposed to being hand-coded (where 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 not standards compliant.

Q. Running 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 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. But overall, having an organized web server directory structure helps 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 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 in order 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. You need 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 enables users to access http://www.*yourdomain*.com/*somedirectory*/ without using a trailing filename and still end up in the appropriate place.
5. No. Using a directory structure for file organization is completely up to you, although using one is highly recommended because it simplifies content maintenance.

Exercises

- Get your web hosting in order—are you going to move through the lessons 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, 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 preceding exercise, place these files on a removable media device such as a USB drive. 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.