1

The Basics of HTML

Before you begin to code HTML pages for the Web, it is important to understand some of the technology, standards, and syntax behind the Web. This chapter introduces you to HTML and answers the following questions:

- What is the World Wide Web?
- How does the Web work?
- What is HTML?
- What is the basic syntax of HTML?

Subsequent chapters in this section delve into the specifics of HTML, covering various tags you can use to format your pages.

What Is the World Wide Web?

The Internet is a worldwide network of computers all attached in a global networking scheme. This scheme, known as TCP/IP, assigns and uses unique addresses to communicate between computers on the Internet.

The World Wide Web is a network of computers that, using the Internet, are able to exchange text, graphics, and even multimedia content using standard protocols. *Web servers* — special computers that are set up for the distinct purpose of delivering content — are placed on the Internet with specific content for others to access. *Web clients* — which are generally desktop computers but can also be dedicated terminals, mobile devices, and more — access the servers' content via a browser. The *browser* is a specialized application for displaying Web content.

For example, Google maintains many Web servers that connect to their database of content found on the Web. You use your home or office PC to connect to the servers via a browser such as Microsoft's Internet Explorer or Mozilla's Firefox (shown in Figure 1-1).

Figure 1-1

If you were to make a diagram of the relationships between all the technical components involved in requesting and delivering a document over the Web, it would resemble the diagram shown in Figure 1-2.

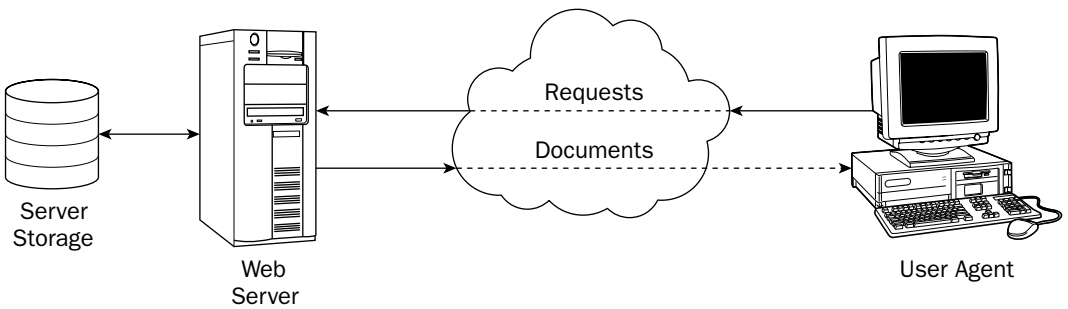

Figure 1-2

Creating a Web

The Web was created as a replacement for the aging Gopher protocol. Gopher allowed documents across the Internet to be linked to each other and searched. The inclusion of *hyperlinks*—embedded links to other documents on the Web—gives the resulting technology its name because it resembles a spider's web.

Figure 1-3 shows a graphic representation of a handful of sites on the Web. When a line is drawn between the sites that link to one another, the *web* becomes more obvious.

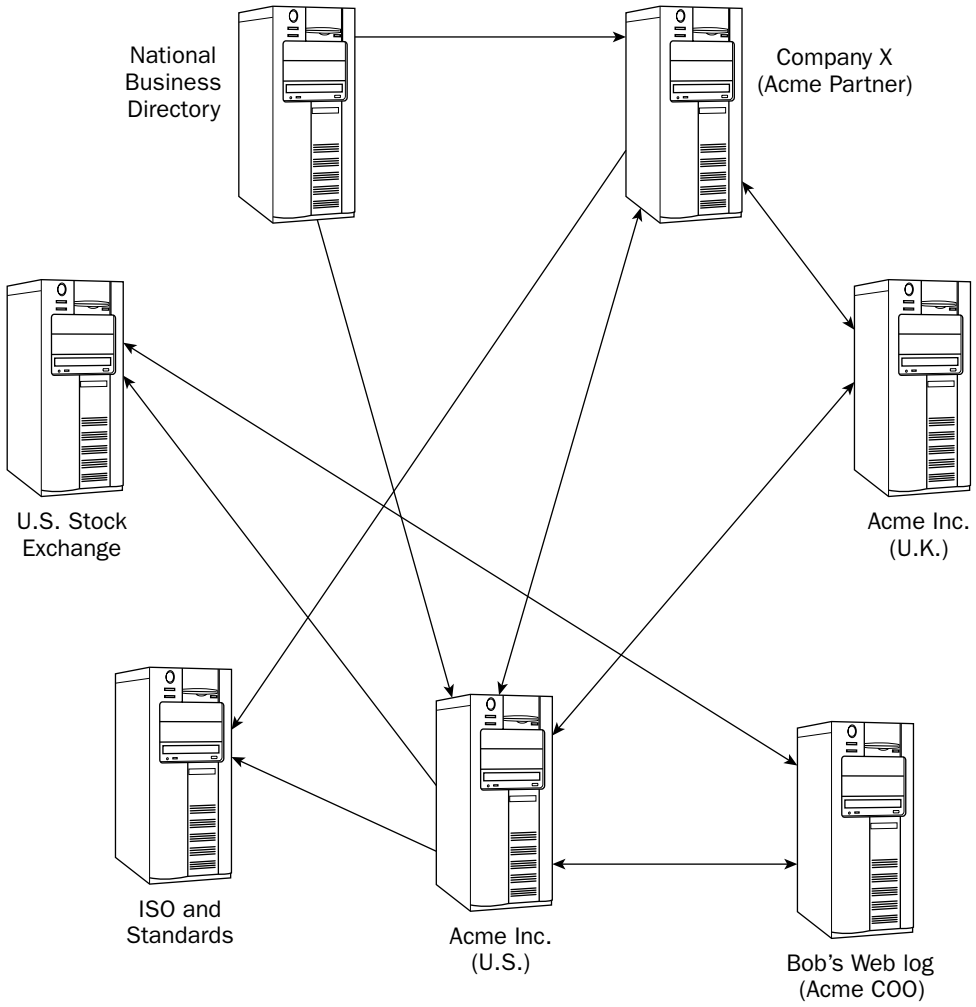

Figure 1-3

However, the Web doesn't operate as the diagram would have you believe. One Web site doesn't go to another for information; your browser requests the information directly from the server where the information can be found. For example, suppose you are on the Acme Inc US site in Figure 1-3 and click the link to Company X. The Acme Inc US server doesn't handle the request for the external page; your browser reads the address of the new page from the hyperlink and requests the information from the server that actually hosts that page (Company X in the example from Figure 1-3).

Hyperlinks contain several pieces of vital information that instruct the Web browser where to go for the content. The following information is provided:

- The protocol to use (generally HTTP)
- The server to request the document from
- The path on the server to the document
- The document's name (optional)

The information is assembled together in a URL. The information is presented in the following form:

- The protocol followed by a colon (for example, `http:`)
- The fully qualified domain name of the server, prefixed by two slashes (for example, `//www.google.com`)
- The path to the file being requested, beginning with a slash, with a slash between each directory in the path and a slash at the end (for example, `/options/`)
- The name of the file being requested (for example, `index.html`)

Most Web servers are configured to deliver specific documents if the browser doesn't explicitly request a document. These specific documents differ between server applications and configurations but are generally documents such as ***`index.html`*** *and* ***`home.html`****. For example, the following two URLs will return the same document (* ***`index.html`****):*

http://www.google.com/options/
http://www.google.com/options/index.html

Taken all together, a URL resembles that shown in Figure 1-4.

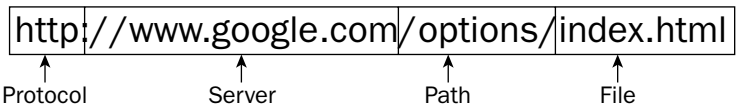

Figure 1-4

HTTP: The Protocol of the Web

As previously mentioned, the Web operates by sending data using specific protocols. The main protocol used for the Web is Hypertext Transfer Protocol (HTTP). HTTP defines how the computers on the Web, specifically the server and client, exchange data.

Although HTTP is the protocol of choice for the Web, most browsers support additional protocols such as the File Transfer Protocol (FTP).

Much like other protocols, an HTTP conversation consists of a handful of commands from the client and a stream of data from the server. Although discussing the whole HTTP protocol is beyond this book's scope, it is important to grasp the basics of how the protocol operates. By using a telnet client, you can "talk" to a Web server and try the protocol manually as shown in the following code (text typed by the user appears with a gray background):

```
telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET /index.html HTTP/1.1
Accept: text/plain,text/html
Host: localhost
User-Agent: Telnet

HTTP/1.1 200 OK
Date: Sun, 17 Oct 2004 23:47:49 GMT
Server: Apache/1.3.26 (Unix) Debian GNU/Linux PHP/4.1.2
Last-Modified: Sat, 26 Oct 2002 09:12:14 GMT
ETag: "19b498-100e-3dba5c6e"
Accept-Ranges: bytes
Content-Length: 4110
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
   <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1">
   <META NAME="GENERATOR" CONTENT="Mozilla/4.05 [en] (X11; I; Linux 2.3.99-pre3
i686) [Netscape]">
   <META NAME="Author" CONTENT="johnie@debian.org (Johnie Ingram)">
   <META NAME="Description" CONTENT="The initial installation of Debian/GNU
Apache.">
   <TITLE>Welcome to Your New Home Page!</TITLE>
</HEAD>
<BODY TEXT="#000000" BGCOLOR="#FFFFFF" LINK="#0000EF" VLINK="#55188A"
ALINK="#FF0000">

<BR>

<H1>Welcome to Your New Home in Cyberspace!</H1>

<HR NOSHADE>
<BR>

<IMG ALIGN="right" ALT="" HEIGHT="247" WIDTH="278" SRC="icons/jhe061.gif">
```

```
<P>This is a placeholder page installed by the <A
HREF="http://www.debian.org/">Debian</A>
release of the <A HREF="http://www.apache.org/">Apache</A> web server package,
because no home page was installed on this host.
...
</BODY>
</HTML>
Connection closed by foreign host.
```

The telnet client is started with the name of the host to connect to and the port number (80):

```
telnet localhost 80
```

Once the client is connected, the server waits for a command. In this case, the client (our telnet session) sends a block of commands, including the following:

- The document to be retrieved and the protocol to return the document (`GET` and `HTTP 1.1`)
- The types of documents the client expects or can support (plain text or HTML text)
- The host the request is destined for (typically the fully qualified domain name of the server)
- The name of the user agent (browser) doing the requesting (`Telnet`)

```
GET /index.html HTTP/1.1
Accept: text/plain,text/html
Host: localhost
User-Agent: Telnet
```

Note that only the first three pieces of data are necessary; the user agent name is provided only as a courtesy to the Webmaster on the server as it gets recorded in the server logs accordingly.

This block of commands is known as the header and is required to be followed by a blank line, which indicates to the server that the client is done with the header. The server then responds with information of its own, including the following:

- A response to the command

```
HTTP/1.1 200 OK
```

- The current date (as known by the server)

```
Date: Sun, 17 Oct 2004 23:47:49 GMT
```

- The server identification string, which usually identifies the type and capabilities of the server but can be configured differently

```
Server: Apache/1.3.26 (Unix) Debian GNU/Linux PHP/4.1.2
```

- Information about the document being delivered (date modified, size, encoding, and so on)

```
Last-Modified: Sat, 26 Oct 2002 09:12:14 GMT
ETag: "19b498-100e-3dba5c6e"
Accept-Ranges: bytes
Content-Length: 4110
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1
```

- The content of the document itself (in this case, the default Debian/GNU Linux Apache welcome page)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
   <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1">
. . .
```

A few seconds after the full document is delivered, the server closes the connection.

```
Connection closed by foreign host.
```

This dialog is HTTP at its simplest, but it does a good job of illustrating how the protocol works.

Hypertext Markup Language

Hypertext Markup Language (HTML) was devised as an easy means to format textual documents. HTTP is the method for delivering HTML documents, which the client browser then renders into an on-screen image. This section covers the development and evolution of HTML.

In the Beginning — HTML

HTML and HTTP were both invented by Tim Berners-Lee, who was then working as a computer and networking specialist at a Swiss research institute. He wanted to give the institute's researchers a simple markup language that would enable them to share their research papers via the Internet. Berners-Lee based HTML on Standard Generalized Markup Language (SGML), an international standard for marking up text for presentation on a variety of physical devices. The basic idea of SGML is that the document's *structure* should be separated from its *presentation.*

To date, HTML has gone through four major standards, including the latest, 4.01. In addition to HTML, Cascading Style Sheets (CSS) and Extensible Markup Language (XML) have also provided valuable contributions to the way of the Web.

> *Most of the standards used on the Web are developed and/or ratified by the World Wide Web Consortium (W3C). The resulting specifications can be found online at the W3C Web site,* **`www.w3c.org`**.

HTML 1.0

HTML 1.0 was never specified by the W3C, as it predated the organization. The standard supported a few basic tags and graphics, although the latter needed to be in GIF format if used in-line or JPEG format if the image was out-of-line. You couldn't specify the font, background images, or colors, and there were no tables or forms. At the time, only one browser, Mosaic 1.0, was available to view Web documents. However, the standard became the stepping-stone to the modern Web.

HTML 2.0

The HTML 2.0 standard provided a wealth of improvement over the 1.0 version. Background colors and images were supported, as were tables and rudimentary forms. Between 1.0 and 2.0, a new browser was launched (Netscape), and several HTML features created to support features in the new browser became part of the 2.0 standard.

HTML 3.2

The HTML 3.2 standard significantly increased the capability of HTML and the Web. Many of the new features enabled Web designers to create feature-rich and elegant designs via new layout tags. Although the 3.2 specification introduced Cascading Style Sheets (CSS level 1), browsers were slow to adopt the new way of formatting. However, the standard did not include frames, but the feature was implemented in the various browsers anyway.

> *There was an HTML 3.0 proposed standard, but it could not be ratified by the W3C in time. Hence, the next ratified standard was HTML 3.2.*

HTML 4.0

HTML 4.0 did not introduce many new features, but it ushered in a new way of implementing Web design. Instead of using explicit formatting parameters in HTML tags, HTML 4.0 encouraged moving the formatting parameters to style sheets instead. HTML 3.2 had become burdensome to support with several dozen tags with several parameters each. The 4.0 standard emphasized the use of CSS, where formatting changes could be made within one document (the style sheet) instead of individually editing every page on a site.

XML 1.0

The Extensible Markup Language (XML) was created as a stepping-stone to bring Standard Generalized Markup Language (SGML) concepts to HTML. Although it was the precursor to HTML, SGML was not widely endorsed. As such, the W3C sought to create a usable subset of SGML targeted specifically toward the Web. The new standard was meant to have enough flexibility and power to provide traditional publishing applications on the Web. XML became part of the new XHTML standard.

CSS 1.0 and 2.0

As mentioned previously, Cascading Style Sheets were devised to move formatting methods used in Web documents into centralized, formal style sheets. A CSS document contains formatting specifics for various tags and is applied to applicable documents. This mechanism provides a formal means to separate the formatting from the content of a page.

When the formatting needs to change, the CSS document alone can be updated, and the changes are then reflected in all documents that use that style sheet. The "cascade" in the name refers to the feature that allows styles to be overridden by subsequent styles. For example, the HR department Web pages for Acme Inc. can use the company style sheet but also use styles specific for the individual department. The result is that all of the Acme Inc. Web pages look similar, but each department has a slightly unique look and feel.

Note: CSS is covered in depth in Part II of this book.

HTML 4.01

Heralded as the last of the HTML standards, 4.01 fixed errors inherent in the 4.0 specification and made the final leap to embracing CSS as the vehicle for document formatting (instead of using parameters in HTML tags).

XHTML 1.0

Extensible Hypertext Markup Language is the latest standard for Web documents. This standard infuses the HTML 4.01 standard with extensible language constructs courtesy of XML. It was designed to be used in XML-compliant environments yet be compatible with standard HTML 4.01 user agents. As of this writing, adoption of the XHTML standard for Web documents has been slow. Although most browsers natively support HTML 4.01, most do not support the extensibility features of XHTML 1.0.

HTML Concept and Syntax

The concept and use of HTML is straightforward. Individual tags — special text strings that are interpreted as formatting commands by the browser — are placed within a document to lend structure and format accordingly. Each tag has a beginning and an ending tag; everything between the tags is formatted according to the tag's parameters or related style sheet.

HTML Tags

Each tag begins with a left-pointing angle bracket (<) and ends with a right-pointing angle bracket (>). Between the brackets are keywords that indicate the type of tag. Beginning tags include any parameters necessary for the tag; ending tags contain only the keyword prefixed by a slash.

For example, if you want a word to be bold in a document, you would surround it with bold tags (`<b>` and `</b>`) similar to the following:

```
If I wanted this word to be <b>bold</b> I would use bold tags.
```

Many tags require children tags to operate. For example, the `<table>` tag itself only marks the position in the document where a table will appear; it does nothing to format the table into rows and columns. Several child tags — `<tr>` for rows, `<td>` for cells/columns, and so on — are used between the beginning and ending `<table>` tags accordingly:

```
<table border="0" >
<tr>
  <td>Cell 1</td>
  <td>Cell 2</td>
</tr>
```

```
<tr>
  <td>Cell 3</td>
  <td>Cell 4</td>
</tr>
</table>
```

Notice how the tags are closed in the opposite order they were opened. Although intuitive for structures like tables, it isn't always as intuitive. For example, consider the following fragment where the phrase "italic and bold" is formatted with italic and bold tags:

This sentence uses *******italic and bold******* *tags for emphasis.*

Although this example would generally not cause a problem when rendered via a user agent, there are many instances where overlapping tags can cause problems. Well-formed HTML always uses nested tags — tags are closed in the exact opposite order that they were opened.

Simple Rules for Formatting HTML Documents

As with most programming languages, formatting plays a big role in writing Web documents that not only display as intended but also are easily understood and maintained. These simple rules should always be followed when creating Web documents:

- **Use liberal white space.** Browsers ignore superfluous white space, so you can make use of it to create documents that are more easily read and maintained. Insert blank lines and follow standard coding rules for indentation whenever possible.
- **Use well-formed code.** This means following the XHTML standard to the letter — not taking shortcuts that some browsers allow. In particular, you should pay attention to the following:
  - Always include a `<doctype>` tag (`<doctype>` and other document-level tags are discussed in Chapter 2.
  - Elements (tags) must be nested, not overlapping.
  - All nonempty elements need to be terminated. Most browsers allow for nonclosed elements, but to meet the XHTML standard you need to supply closing tags for each open one (for example, supply a closing paragraph tag [`</p>`] for every open one [`<p>`]).
  - All tags need to be closed. Although the HTML standard allows tags such as `<hr>` without a closing tag — in fact, the `<hr>` tag has no closing mate — in XML all tags must be closed. In the case of tags like `<hr>`, you close the tag by putting a slash at the end of the tag: `<hr />`.
  - All attribute values must be quoted. Again, most browsers allow nonquoted attributes, but the XHTML standard does not.
  - All attributes must have values. Older HTML standards allowed for tags similar to the following:

```
<input type="checkbox" checked>
```

However, XHTML does not allow attributes without values (for example, `checked`). Instead, you must supply a value, such as the following:

```
<input type="checkbox" checked="checked">
```

- **Comment your code.** Using the comment tag pair (`<!--` and `-->`) should be as natural as commenting code in programming languages. Especially useful are comments at the end of large blocks, such as nested tables. It can help identify which part of the document you are editing:

```
</table>  <!-- End of floating page -->
```

Your First Web Page

As you will see in the other chapters within this section, many elements can make up a Web document, and you can use many HTML entities to format your documents. However, the following simple example uses only the basic, necessary tags to produce a page.

Example: A Simple Web Page

This example produces a simple HTML document with one line of text, using the bare minimum number of HTML tags.

Source

Type the following code into a document and save it, in plain text format, as `sample.html` on your local hard drive.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
<!-- sample.html - A simple, sample web document -->
<head>
  <title>A simple HTML document</title>
</head>
<body>
<p>This is sample text.</p>
</body>
</html>
```

Output

Now open the document in a Web browser. In most graphical operating environments, you can simply use a file manager to find the `sample.html` file and then double-click on it. Your default Web browser should open and load the file. If not, select Open (or Open File) from the File menu and find the `sample.html` file using the browser's interface.

Your screen should resemble that shown in Figure 1-5.

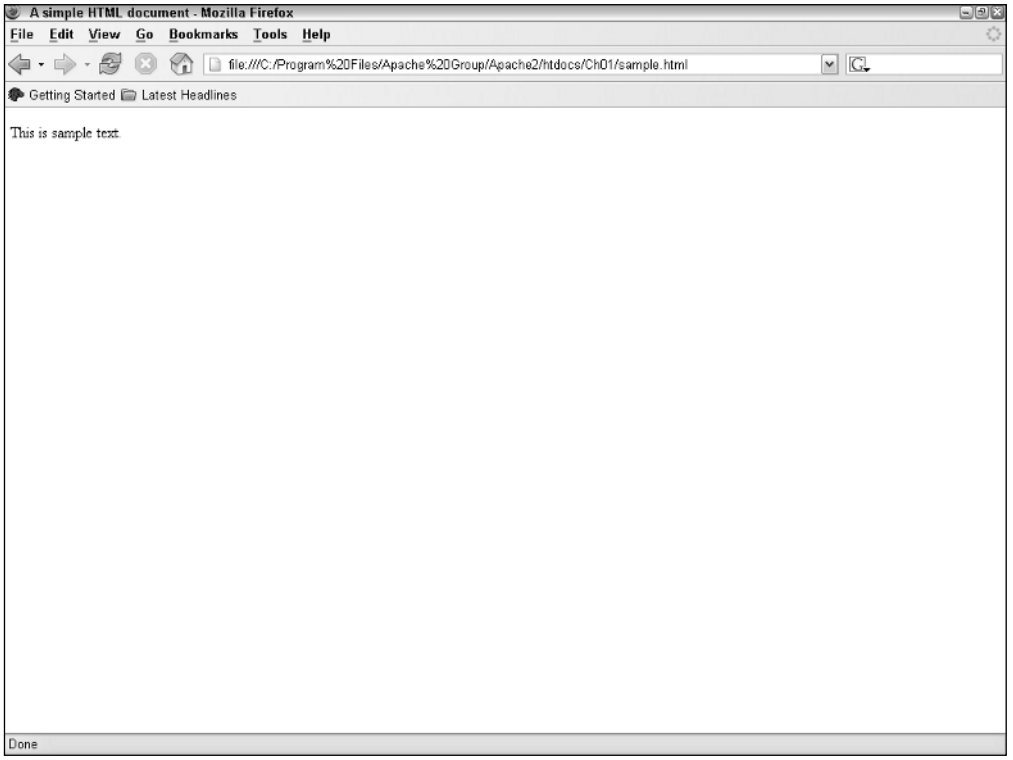

Figure 1-5

At this point, you may be asking yourself, "Why don't I need a Web server?" The reason is simple: The browser loads and interprets the HTML file from the local hard drive; it doesn't have to request the file from a server. However, the file uses only HTML, which is interpreted only by the client side. If you used any server-side technologies (Perl, PHP, and so on), you would have to load the sample file onto a Web server that had the appropriate capabilities to process the file before giving it to the client. More information on server-side technologies can be found in Parts V and VI of this book.

Summary

This chapter introduced you to the World Wide Web and the main technology behind it, HTML. You saw how the Web works, how clients and servers interact, and what makes up a hyperlink. You also learned how HTML evolved and where it is today. This basic background serves as a foundation for the rest of the chapters in this section, where you will learn more about specific HTML coding.