Chapter 1

The Least You Need to Know about HTML, CSS, and the Web

In This Chapter

- Creating HTML in text files
- Serving and browsing Web pages
- Understanding links and URLs
- Understanding basic HTML syntax
- Understanding basic CSS

Welcome to the wonderful world of the Web, (X)HTML, and CSS. With just a little knowledge, some practice, and something to say, you can build your own little piece of cyberspace or improve on existing work.

You'll notice we use (X)HTML throughout this book. This is an acronym we made up to stand for "either HTML or XHTML," where HTML is Hypertext Markup Language, and XHTML is Extensible Hypertext Markup Language. Although HTML and XHTML aren't exactly identical, they're enough like each other for this reference to make sense.

This book is your down-and-dirty guide to understanding Web documents, sprucing up an existing page, or creating complex and exciting pages that integrate intricate designs, multimedia, and scripting.

The best way to start working with HTML is to jump right in, so that's what this chapter does: It brings you up to speed on the basics of how (X)HTML and CSS work behind the scenes inside Web pages, introducing you to their underlying building blocks. When you're done with this chapter, you'll know how (X)HTML and CSS work so you can start creating or editing Web pages right away.

Web Pages in Their Natural Habitat

Web pages can accommodate many kinds of content, such as *text, graphics, forms, audio and video files,* and even *interactive games.*

Browse the Web for only a moment, and you see a buffet of information and content displayed in many ways. Every Web site is different, but most have one thing in common: the Hypertext Markup Language (also known as HTML). You'll also run into Extensible Hypertext Markup Language (XHTML) and Cascading Style Sheets (CSS) pretty regularly, too.

Whatever information a Web page contains, every Web page is created using HTML (or some reasonable facsimile). HTML is the mortar that holds Web pages together; graphics, content, and other information are the bricks; CSS tells Web pages how they should look when on display.

HTML files that produce Web pages are just text documents, as are XHTML and CSS files. This use of text documents is why the Web works as well as it does. Text is a universal language for computers. Any text file you create on a Windows computer — including any HTML, XHTML, or CSS file — works equally well on a Mac or any other operating system.

But Web pages aren't *merely* text documents. Web pages are made with special, attention-deprived, sugar-loaded text called HTML, XHTML, or CSS. Each uses its own specific set of instructions that you include (along with your content) inside text files to specify how a page should look and behave.

Stick with us to discover everything you need to know about (X)HTML and CSS!

Hypertext

Special instructions in HTML permit lines of text to point (that is, *link*) to something else in cyberspace. Such pointers are called *hyperlinks.* Hyperlinks are the glue that holds the World Wide Web together. In your Web browser, hyperlinks usually appear in blue and are underlined. When you click a hyperlink, it takes you somewhere else.

Hypertext or not, a Web page is a text file, which means you can create and edit a Web page in any application that creates plain text (such as Notepad or TextEdit). Some software tools offer fancy options and applications (covered in Chapter 23) to help you create Web pages, but they generate the same text files that you create with plain-text editors. We're of the opinion, though, that those just getting started with HTML are best served by a simple text editor. Just break out Notepad on the PC (or TextEdit on the Mac), and you're ready to go.

Steer clear of word processors like WordPad or Microsoft Word when creating HTML. They introduce all kinds of extra code to Web pages that you may neither want nor need.

The World Wide Web comes by its name honestly. It's quite literally a web of online pages hosted on Web servers around the world, connected in trillions of ways by hyperlinks that tie pages together. Without such links, the Web would be just a bunch of standalone pages.

Much of the Web's value comes from its ability to link to pages and other resources (such as images, downloadable files, and media presentations) on either the same Web site or at another site. For example, USA.gov (`www.usa.gov`) is a *gateway* Web site — its sole function is to provide access to other Web sites. If you aren't sure which government agency handles first-time loans for homebuyers, or you want to arrange a tour of the Capitol, visit the site shown in Figure 1-1 to find out.

Figure 1-1: USA.gov uses hyperlinks to help visitors find government information.

Web browsers were created specifically for the purpose of reading HTML instructions (known as *markup*) and displaying the resulting Web page.

Markup lives in a text file (with your content) to give orders to a browser. For example, look at the page shown in Figure 1-2. You can see how the page is made up and how it is formatted by examining its underlying HTML.

Figure 1-2: To achieve its present good looks, this Web page incorporates multiple parts and numerous bits of HTML and CSS markup.

This page includes an image, a heading that describes the page, several paragraphs of text about one of your authors, and an address block with links to a résumé and a list of publications.

However, different components of the page use different formatting:

- The heading at the top of the page is larger than text in the paragraphs.
- Blocks of text are separated by more blank space than between contiguous lines of text within blocks.
- Some text is in white, some orange, and some light blue.

The browser knows to display these components of the page in specific ways thanks to the *HTML markup,* shown in Listing 1-1. (You'll see Listing 1-1 in all its glory at the end of the chapter.)

Any text enclosed between angle brackets (less-than and greater-than signs: < >) is an HTML *tag* (often called the *markup*). For example, a `p` within brackets (`<p>…</p>` tags) identifies text inside paragraphs. The markup between `<style>` and `</style>` tags at the head of the file uses CSS to define the look and feel for various HTML elements used on this page. That's really all there is to it. You embed the markup in a text file, along with text for readers to view, to tell the browser how to display your Web page.

Tags and the content between (and within) the tags are collectively called *elements.* Angle brackets < > enclose HTML and XHTML markup, curly braces `{ }` enclose CSS markup.

Browsers

The user's piece in the Web puzzle is a Web browser. Web browsers read instructions written in HTML, XHTML, and CSS, and use those instructions to display Web page content on your screen.

You should always write your HTML with the idea that people will view the content using a Web browser. Just remember that there's more than one kind of browser out there, and each one comes in several versions.

Usually, Web browsers request and display Web pages available via the Internet from a Web server. You can also display HTML pages you've saved on your own computer before making them available on a Web server on the Internet. When you're developing your own HTML pages, you view these pages (called *local* pages) in your browser. You can use local pages to get a good idea of what people see after the page goes live on the Internet.

Each Web browser interprets HTML in its own way. The same HTML may not look exactly alike from one browser to the next. When you work with basic HTML, variations will be minor, but as you add other elements (such as scripting and multimedia), rendering markup gets hairy.

Chapter 2 shows how to use a Web browser to view a local copy of your first Web page.

Some people use text-only Web browsers, such as Lynx, because either

- They're visually impaired and can't use a graphical display.
- They like a lean, fast Web browser that displays only text.

A bevy of browsers

The Web world is full of browsers of many shapes and sizes — or rather versions and feature sets. Some popular browsers are Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, and Google Chrome. Other browsers, such as Opera and Lynx, are also widely used. As an HTML developer, you must think beyond your own browser experience and preferences. Every user has his or her personal browser preferences and settings.

Each browser renders HTML a bit differently. Every browser handles JavaScript, multimedia, style sheets, and other HTML add-ins differently too. Throw different operating systems into the mix, and things get really fun.

Usually differences between browsers are minor. But sometimes a combination of HTML, text, and media brings a specific browser to its knees. When you work with HTML, test your pages on as many different browsers as you can. Install at least three different browsers on your own system for testing. We recommend the latest versions of Internet Explorer, Firefox, and Chrome.

Yahoo! has a fairly complete list of browsers at

```
http://dir.yahoo.com/Computers_and_Internet/Software/Internet/World_Wide_Web/
        Browsers
```

Web servers

Your HTML pages aren't much good if you can't share them with the world. Web servers make that possible. A *Web server* is a computer that

- Connects to the Internet
- Runs Web-server software
- Responds to requests from Web browsers for Web pages

Almost any computer can be a Web server, including your home computer. But Web servers generally are computers dedicated to the task. You don't need to be an Internet or computer guru to publish your Web pages, but you must find a Web server to serve your pages:

- If you're building pages for a company Web site, your IT department may have a Web server. (Ask your IT guru for the information.)
- If you're starting a new site, you need a host for your Web pages.

  Finding an inexpensive host is easy — all it takes is a simple Google search. One inexpensive host is GoDaddy (`www.godaddy.com`), with current monthly fees as low as $1.99 a month. You can even find free

hosts for your Web site with a little effort. Free Web Hosts maintains a list of free host providers. Check them out at `www.free-webhosts.com`. Chapter 3 shows how to determine your hosting needs and find the perfect provider.

Anatomy of a URL

The Web is made up of billions of resources, each of them linkable. A resource's exact location is the key to linking to it. Without an exact address (a *Uniform Resource Locator,* or *URL*), you can't use the Address bar in a Web browser to visit a Web page directly.

URLs are the standard addressing system for Web resources. Each resource (Web page, site, or individual file) has a unique URL. URLs work a lot like your postal address. Figure 1-3 identifies the components of a URL.

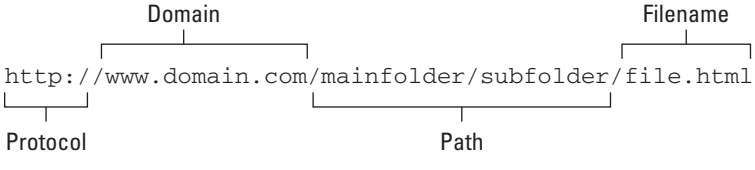

Figure 1-3: The components of a URL help it define an exact location for a file on the Web.

Introducing Internet protocols

Interactions between browsers and servers are made possible by a set of computer-communication instructions: Hypertext Transfer Protocol (HTTP). This protocol defines how browsers should request Web pages and how Web servers should respond to those requests.

HTTP isn't the only protocol at work on the Internet. The Simple Mail Transfer Protocol (SMTP) and Post Office Protocol (POP) make e-mail exchange possible, and the File Transfer Protocol (FTP) allows you to upload, download, move, copy, and delete files and folders across the Internet. The good news is that Web browsers and servers do all the HTTP work for you, so you only have to put your pages on a server or type a Web address into a browser.

To see how HTTP works, check out David Gourley and Brian Totty's chapter on HTTP Messages, available through Google book search with "understanding http transactions" as the search string. Start your search at `http://books.google.com`, then scroll down until you see the link to "HTTP: the definitive guide" and check out Page 80.

Each URL component helps define the location of a Web page or resource:

- **Protocol:** Specifies the protocol the browser follows to request the file.

  The Web page protocol is `http://` (the usual start to most URLs).
- **Domain:** Points to the general Web site (such as `www.sun.com`) where the file resides. A domain may host a few files (like a personal Web site) or millions of files (like a large corporate site, such as `www.sun.com`).
- **Path:** Names the sequence of folders through which you must navigate to get to a specific file.

  For example, to get to a file in the `evangcentral` folder that resides in the `developers` folder, you use the `/developers/evangcentral/` path.
- **Filename:** Specifies which file in a directory path the browser accesses.

Although the URL shown in Figure 1-3 is not publicly accessible, it points to the domain and offers a path that leads to a specific file named `file.html`:

```
http://www.domain.com/mainfolder/subfolder/file.html
```

Chapter 6 provides the complete details on how you use HTML and URLs to add hyperlinks to your Web pages, and Chapter 3 shows how to obtain a URL for your own Web site after you're ready to move it to a Web server.

(X)HTML's Component Parts

The following section removes the mystery from the *X*. This section shows

- The differences between HTML and XHTML
- How HTML is written (its *syntax*)
- Rules that govern use of HTML (and XHTML)
- Names for important pieces and parts of HTML (and XHTML) markup
- How to make the best, most correct use of (X)HTML capabilities

HTML and XHTML: What's the difference?

HTML is *Hypertext Markup Language,* markup developed in the late 1980s and early 1990s to describe Web pages. HTML is now enshrined in numerous standard descriptions *(specifications)* from the World Wide Web Consortium (W3C). The last HTML specification was done in 1999.

When you put an *X* in front of HTML to get XHTML, you get a new, improved version of HTML based on the *eXtensible Markup Language (XML).* XML is designed to work and behave well with computers, software, and the Internet.

The original formulation of HTML has some irregularities that can cause heartburn for software that reads HTML documents. XHTML, on the other hand, uses an extremely regular and predictable syntax that's easier for software to handle. XHTML will replace HTML someday, but HTML keeps on ticking. This book covers both varieties and shows you the steps to put the X in front of your own HTML documents and turn them into XHTML.

- Most HTML and XHTML markup is identical.
- In a few cases, HTML and XHTML markup looks a little different.
- In a few cases, HTML and XHTML markup must be used differently.

This book shows how to create code that works in both HTML and XHTML.

Syntax and rules

HTML is a straightforward language for describing Web page contents. XHTML is even less demanding. Their components are easy to use — when you know how to use a little bit of (X)HTML. Both HTML and XHTML markup have three types of components:

- **Elements:** Identify different parts of an HTML page by using tags.
- **Attributes:** Information about an instance of an element.
- **Entities:** Non-ASCII text characters, such as copyright symbols (©) and accented letters (É). Entities originate from the Standard Generic Markup Language, or SGML.

Every bit of HTML and/or XHTML markup that describes a Web page's content includes some combination of elements, attributes, and entities.

This chapter covers basic form and syntax for elements, attributes, and entities. Parts II and III of the book show how elements and attributes:

- Describe kinds of text (such as paragraphs or tables)
- Create an effect on the page (such as changing a font style)
- Add images and links to a page

Markup color-coding

As we present HTML, XHTML, and CSS information in our code samples, we use color-coding to help you distinguish what's what by way of markup. Here is a color key that you should keep in mind as you read all of our code listings:

- **Purple** Indicates the `DOCTYPE` declaration used in (X)HTML documents. This is actually a totally different markup language known as the Standard Generalized Markup Language, or SGML. SGML is used to identify what specific set of rules that (X)HTML documents follow in their construction and content. It also applies to codes for character entities, such as the following:

```
&pos;
&123;
```

- **Light green** Indicates ordinary garden variety XHTML and HTML markup
- **Dark green** Indicates XML markup
- **Orange** Indicates Cascading Style Sheet, or CSS, markup
- **Blue** Indicates JavaScript

We only colorize markup in code listings and code blocks because it affects readability too much when code appears in body copy. In that case, we simply use a different, monospaced font — as you'll see in the discussions of the `<html>`, `<head>`, and `<title>` elements in our first paragraph that discusses HTML markup here.

One more thing: If you use an HTML editor, such as HTML Kit, Dreamweaver, Kompozer, or whatever, you find these tools also use text color to help you identify different kinds of markup. The thing is that none of these tools do this the same way, and none of them match the way we do it here — we picked out colors that would be easy to see (and distinguish) when viewed on a four-color printed page; whereas others picked their colors to look good on LCD displays.

Elements

Elements are the building blocks of (X)HTML. You use them to describe every piece of text on your page. Elements are made up of tags and the content within those tags. There are two main types of elements:

- Elements with content made up of a tag pair and whatever content sits between the opening and closing tags in the pair
- Elements that insert something into the page, using a single tag

Tag pairs

Elements that describe content use a *tag pair* to mark the beginning and the end of the element. Start and end tag pairs look like this:

```
<tag>...</tag>
```

Content — such as *paragraphs, headings, tables,* and *lists* — always uses a tag pair:

- The start tag (`<tag>`) tells the browser, "The element begins here."
- The end tag (`</tag>`) tells the browser, "The element ends here."

Actual content is what occurs between a start tag and an end tag. For example, the Ed Tittel page in Listing 1-1 uses a paragraph element (`<p>`) to surround text for a paragraph (we omit CSS inline markup for clarity):

```
<p>Ed started writing about computing subjects in 1986 for a
Macintosh oriented monthly magazine. By 1989 he had contributed to such
publications as LAN Times, Network World, Mac World, and LAN Magazine. He worked
on his first book in 1991, and by 1994 had contributed to over a dozen different
titles.</p>
```

Single tags

Elements that insert something into the page are called *empty elements* (because they enclose no content) and use just a single tag, like this:

```
<tag />
```

Images and line breaks insert something into the HTML file, so they use one tag.

One key difference between XHTML and HTML is that, in XHTML, all empty elements must end with a slash before the closing greater-than symbol. This is because XHTML is based on XML, and the XML rule is that you close empty elements with a slash, like this:

```
<tag/>
```

However, to make this kind of markup readable inside older browsers, you should insert a space before the closing slash, like this:

```
<tag />
```

This space allows older browsers to ignore the closing slash (because they don't know about XHTML). Newer browsers that understand XHTML ignore the space and interpret the tag exactly, which is `<tag/>` (as per the XML rules).

HTML doesn't require a slash with empty elements, but this markup is *deprecated* (that is, identified as obsolete even though it still occurs in some markup). An HTML empty element looks like this:

```
<tag />
```

Listing 1-1 uses the image element (`<img />`) to include an image on the page:

```
<img src="images/header.gif" alt="header graphic" width="794" height="160" />
```

The `<img />` element references an image. When the browser displays the page, it replaces the `<img />` element with the file that it points to (an attribute does the pointing, as shown in the next section). Following the XHTML rule introduced earlier, what appears in HTML as `<img>` appears in XHTML as `<img />` (and this applies to all single tag elements).

You can't make up HTML or XHTML elements. Legal elements for (X)HTML belong to a very specific set — if you use elements that aren't part of that set, every browser ignores them. The elements you can use are defined in the HTML 4.01 or XHTML 1.0 specifications. (The specs for HTML 4.01 can be found at `www.w3.org/TR/html4`, while the specs for XHTML 1.0 can be found at `www.w3.org/TR/xhtml1`.)

Nesting

Many page structures combine nested elements. Think of your nested elements as *suitcases* that fit neatly inside one another.

For example, a bulleted list uses two kinds of elements:

- The `<ul>` element specifies that the list is unordered (bulleted).
- The `<li>` elements mark each item in the list.

When you combine elements by using this method, be sure you close the inside element completely before you close the outside element:

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
</ul>
```

Attributes

Attributes allow variety in how an element describes content or works. Attributes let you use elements differently depending on circumstances. For example, the `<img />` element uses the `src` attribute to specify the location of the image you want to include on your page:

```
<img src="images/header.gif" alt="header graphic" width="794" height="160" />
```

In this bit of HTML, the `<img />` element itself is a general flag to the browser that you want to include an image; the `src` attribute provides the specifics on the image you want to include — `header.gif` in this instance. Other attributes (such as `width` and `height`) provide information about how to display that image, while the `alt` attribute provides a text alternative to the image that a text-only browser can display (or a text-to-speech reader can read aloud, for the visually impaired).

Chapter 7 describes the `<img />` element and its attributes in detail.

You include attributes within the start tag of the element you want them with — after the element name but before the ending sign, like this:

```
<tag attribute="value" attribute="value">
```

XML syntax rules decree that attribute values must always appear in quotation marks, but you can include the attributes and their values in any order within the start tag or within a single tag.

Every (X)HTML element has a collection of attributes that can be used with it, but you can't mix and match attributes and elements however you please. Some attributes can take any text as a value because the value could be anything, like the location of an image or a page you want to link to. Others have a specific list of values the attribute can take, such as your options for aligning text in a table cell.

The HTML 4.01 and XHTML 1.0 specifications define exactly which attributes you can use with any given element and which values (if explicitly defined) each attribute can take.

Each chapter in Parts II and III covers which attributes you can use with each (X)HTML element. Also, see our online content for complete lists of deprecated (X)HTML tags and attributes.

Entities

Text makes the Web possible, but it has limitations. *Entities* are special characters that you can display on your Web page.

Non-ASCII characters

Basic American Standard Code for Information Interchange (ASCII) text defines a fairly small number of characters. It doesn't include some special characters, such as *trademark symbols, fractions,* and *accented characters.*

For example, if we translate a paragraph of text from the page in Figure 1-2 into German, the result includes three *u* characters with umlauts *(ü),* as shown in Figure 1-4.

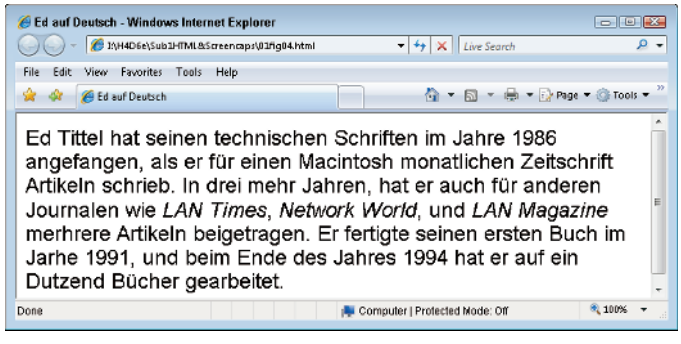

Figure 1-4: ASCII text can't represent all text characters, so HTML entities do the job instead.

ASCII text doesn't include an umlauted *u,* so HTML uses *entities* to represent such characters. The browser replaces the entity with the character it references. Each entity begins with an ampersand (&) and ends with a semicolon (;); entities come originally from SGML, so we color-code them in purple to reflect their origins. The following markup shows entities in bold:

```
<html>
<head>
<style type="text/css">
  body {
    font-family: sans-serif;
    font-size: large;
    }
  cite {
    font-family: serif;
    font-style: italic;
    }
```

```
</style>
<title>Ed auf Deutsch</title>
</head>
<body>
<p>Ed Tittel hat seinen technischen Schriften im Jahre 1986 angefangen, als er
f&uuml;r einen Macintosh monatlichen Zeitschrift Artikeln schrieb. In drei mehr
Jahren, hat er auch f&uuml;r anderen Journalen wie <cite>LAN Times</cite>,
<cite>Network World</cite>, und <cite>LAN Magazine</cite> merhrere Artikeln
beigetragen. Er fertigte seinen ersten Buch im Jarhe 1991, und beim Ende des
Jahres 1994 hat er auf ein Dutzend B&uuml;cher gearbeitet.</p>
</body>
</html>
```

The entity that represents the umlauted *u* is `&uuml;`.

(X)HTML character codes

The encodings for the ISO-Latin-1 character set are supplied by default, and related entities (a pointer to a complete table appears in Chapter 24) can be invoked and used without special contortions. But using other encodings mentioned earlier requires inclusion of special markup to tell the browser it must interpret Unicode character codes. (Unicode is an international standard — ISO standard 10646, in fact — that embraces enough character codes to handle most unique alphabets, plus plenty of other symbols and nonalphabetic characters as well.) This special markup takes the form `<meta http-equiv="Content-Type" content="text/html; charset=UTF 8">`; because the value for `charset` reads `UTF-8`, you can reference common Unicode values that appear in Chapter 24.

Although today's browsers support UTF-8 across the board, you can expect to see support for UTF-16 character codes showing up in the next year or two. This will let browsers deal more effectively with non-Roman alphabets like Arabic, kata kana (Japanese), or Hangul (Korean), which some browsers struggle to render correctly today.

Tag characters

HTML-savvy software assumes that some HTML characters, such as the greater-than and less-than signs, are meant to be hidden and not displayed on your finished Web page. If you actually want to show a greater-than or less-than sign on your page, you're going to have to make your wishes clear to the browser. The following entities let you display characters that normally are part of the hidden HTML markup:

- **less-than sign (<):** `<`
- **greater-than sign (>):** `>`
- **ampersand (&):** `&`

The < and > signs are used in markup, but these symbols are *instructions to the browser* and won't show up on the page. If you need these symbols on the Web page, include the entities for them in your markup, like this:

```
<p>The paragraph element identifies some text as a paragraph:</p>
<p>&lt;p&gt;This is a paragraph.&lt;/p&gt;</p>
```

In the preceding markup, the first line uses *tags* to describe a paragraph, and the second line shows how *entities* describe the < and > symbols.

Figure 1-5 shows these entities as characters in a browser window.

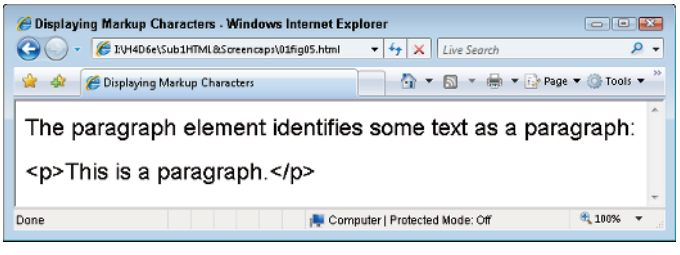

Figure 1-5: Entities let <, >, and & symbols appear in a browser window.

Parts Is Parts: What Web Pages Are Made Of

Comments include text in (X)HTML files that isn't displayed in the final page. Each comment is identified with two special sequences of markup characters:

- Begin each comment with the string <!--
- End each comment with the string -->

In the following code, comments explain how each markup element functions and where it fits into the HTML markup hierarchy.

Elements are organized into a structure:

- Some elements can occur only inside other elements.
- Some elements are required for a well-structured (X)HTML document.

```
<html>  <!-- This tag should always occur at or near the beginning of any
            well-formed HTML document -->
<head>  <!-- The head element supplies information to label the whole HTML
            document -->
```

```
<title>Welcome to Ed Tittel.com</title> <!-- The text in the title element
            appears in the title bar of the browser window when the page
            is viewed -->
</head> <!-- closes the head element -->

<body>  <!-- The content that appears on any Web page appears or is
            invoked from inside the body element -->
        <!-- Skip a bunch of copy here . . . -->
<!-- Subtitle text -->
  <h1>Contact:</h1>
  <!-- List -->
    <ul>
      <li><b>Email:</b> etittel at yahoo dot com</li>
      <li><b>Address:</b> 2443 Arbor Drive, Round Rock, TX 78681-2160</li>
      <li><b>Phone:</b> 512-252-7497 (No solicitors, please)</li>
      <li>List of publications available in: <a href="docs/v_et.doc"
          target="_blank">MS Word</a></li>
      <li>Resume available in: <a href="docs/Resu-et13.doc" target="_
           blank">
          MS Word</a></li>
    </ul></body>  <!-- End of the body section -->
</html>  <!-- End of the HTML document -->
```

The preceding document is broken into a `head` and a `body`. Within each section, certain kinds of elements appear. Many combinations are possible — and that's what you see throughout this book!

To see complete, valid HTML files for any and all screen captures of pages we build in this book, visit the Web site at `www.dummieshtml.com` and check the area for each chapter. The preceding markup appears therein as `01Listing01.html`, for example.

Organizing HTML text

Beyond the division into head and body sections, text can be organized in plenty of ways in HTML documents.

Document heads

Inside the `head` section, you can define all kinds of labels and information besides a title, primarily to describe the document that follows, such as the character sets used, meta data about the current document, scripts to be invoked, and style information. The `body` section is where real content lives and most (X)HTML elements appear.

Document headings

Headings (denoted using elements `h1` through `h6`) are different from the HTML document head. Individual headings structure the text that follows them, whereas the `head` identifies or describes the whole document.

In the Ed Tittel page example, the `h1` element sets off the `Contact` block at the bottom of the page.

Paragraphs and more

When you want running text on a Web page, the paragraph element, `p` (which includes the `<p>` and `</p>` tags), breaks text into paragraphs. You can also create horizontal rules (lines) by using the `<hr />` element.

HTML also includes all kinds of ways to emphasize or identify text inside paragraphs; Parts II and III of this book show a few of them.

Lists

HTML permits easy definition of unordered or bulleted lists. Various mechanisms to create other kinds of lists, including numbered lists, are also available. Lists can be nested within lists to create as many levels of hierarchy as your list might need (perhaps when outlining a complex subject or modeling a table of contents with several heading levels you want to represent). Chapter 5 covers creating lists in more detail.

Tables

In addition to providing a variety of listing mechanisms, HTML also includes markup for defining tables. (Tables were really popular at one time in HTML design, and they were used for all kinds of page layouts; today, they're used for tables, as they should be.) Structure is part of how markup works, so within the definition of a table, you can

- Distinguish between column heads and table data
- Manage how rows and columns are laid out

Cascading Style Sheet markup

CSS markup can occur in separate style-sheet documents, in a block of text in the head of an HTML document, or appended in the style attribute within individual HTML elements — and even in some combination of all three such forms! What CSS does is provide much more detailed control over

font selection, use of color for text and backgrounds, positioning of text and other elements on the page, and (as the old Ronco ad intones) "much, much more."

You delve into CSS in detail in Part III of this book, but we cover bits and pieces of CSS throughout the book as appropriate for the subject matter at hand. You can build a Web site without using CSS (using CSS makes more work), but it's the right tool for precise control over look and layout!

Images in HTML Documents

Adding an image to any HTML document is easy. Careful and well-planned use of images adds greatly to Web pages. Chapter 7 shows how to grab images from files. Chapter 9 shows how to use complex markup to position and flow text around graphics. Along the way, you also discover how to select and use interesting and compelling images to add both allure and information to your Web pages.

Links and navigation tools

Web page structure should help visitors find their way around collections of pages, look for (and hopefully, find) items of interest, and get where they most want to go quickly and easily. Links provide the mechanism to bring people into your Web pages, so Chapter 6 shows how to

- Reference external items or resources
- Jump from one page to the next
- Jump around inside a page
- Add structure and organization to your pages

  The importance of structure and organization increases in relation to the amount of information that you want to present to your visitors.

Navigation tools (which establish standard mechanisms and tools for moving around inside a Web site) provide ways to create and present your Web page (and site) structure to visitors as well as mechanisms for users to grab and use organized menus of choices

When you add everything up, your result should be a well-organized set of information and images that are easy to understand, use, and navigate.

Listing 1-1: Meet an Author!

Listing 1-1 is reproduced in its entirety here, color-coded to distinguish the various types of markup it uses. Lest you think this is mere vanity on Ed's part, we also hasten to point out that this is the basis for the "About me" page described in Chapter 16 of this book, which we hope only makes it more interesting, rather than the reverse!

Listing 1-1: Ed Tittel's "About Me" Web Page

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.
              w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Ed Tittel - Edtittel.com</title>
<style type="text/css">

body {
  background-image: url(images/background_page.gif);
  font-family: Verdana, Arial, Helvetica, sans-serif;
  font-size: .9em;
  line-height: 1.3;
  color: #FFF;
  margin: 0px;
  padding: 0px; }
#container{
  width: 794px;
  margin: 0px auto; }
#headerGraphic{
  background-image: url(images/header.gif);
;
  width: 794px;
  height: 160px; }
b {
  font-weight: bold;
  }
h2 {
  font-weight: bold;
  font-size: 1.5em;
  color:#96CDFF;
  border-bottom: 1px solid white; }

h1 {
  font-weight: bold;
  font-size: 1.2em;
  color:#96CDFF; }
ul{
  list-style-type: none;
  margin: 0px;
  padding: 0px; }
```

```
a:link {
  font-weight : bold;
  text-decoration : none;
  color: #FF7A00;
  background: transparent; }
a:visited {
  font-weight : bold;
  text-decoration : none;
  color: #91a3b4;
  background: transparent; }
a:hover {
  color: #FA0000;
  background: transparent;
  text-decoration : underline; }
a:active {
  color: #494949;
  background: transparent;
  font-weight : bold;
  text-decoration : underline; }
</style>
</head>

<body>
<div id="container">
<!-- Top graphic of Ed and title -->
<div id="headerGraphic"></div>
<!-- Header text -->
  <h2>About me</h2>
  <!-- Paragraphs -->
    <p>Ed Tittel has been working in and around the computer industry since the
    early 1980s, at which point he left academia to work as a programmer. After
    seven years of writing code and managing development projects, he switched
    to the softer side of the industry in pre-sales technical and marketing
    roles. In the period from 1981 to 1994 he worked for 6 companies that
    included Information Research Associates, Burroughs, Schlumberger, and
    Novell.</p>
    <p>Ed started writing about computing subjects in 1986 for a Macintosh
    oriented monthly magazine. By 1989 he had contributed to such publications
    as LAN Times, Network World, Mac World, and LAN Magazine. He worked on his
    first book in 1991, and by 1994 had contributed to over a dozen different
    titles.</p>
    <p>Ed has been freelancing full-time since 1994, with two
    brief stints of other employment interspersed therein (1987-8 at Tivoli,
    and 2006 at NetQoS, Inc.). He has contributed to over 140 computer
    books, including numerous ...For Dummies titles, college textbooks,
    certification preparation materials, and more. These days, Ed revises an
    occasional book, writes for Tom's Hardware, TechTarget, and ITExpertVoice,
    and teaches online courses for large corporations such as HP.</p>
    <p>To learn more about Ed's professional history, please
    read his <a href="bio.htm">professional bio</a>.</p>
```

(continued)

Listing 1-1 *(continued)*

```
    <!-- Subtitle text -->
    <h1>Contact:</h1>
    <!-- List -->
    <ul>
      <li><b>Email:</b> etittel at yahoo dot com</li>
      <li><b>Address:</b> 2443 Arbor Drive, Round Rock, TX 78681-2160</li>
      <li><b>Phone:</b> 512-252-7497 (No solicitors, please)</li>
      <li>List of publications available in: <a href="docs/v_et.doc"
          target="_blank">MS Word</a></li>
      <li>Resume available in: <a href="docs/Resu-et13.doc" target="_blank">
          MS Word</a></li>
    </ul>

</div>
</body>
</html>
```

That's a huge amount of HTML to pore over at the very beginning of this book. Please take our word for it, though: If you read enough of this book's contents, all of it makes perfect sense!

If you check out our Web site for this book (`www.dummieshtml.com`), you find it's broken down chapter by chapter. If you grab the downloads for Chapter 1, you find the source code for the page shown in Listing 1-1, named `aboutme.html`. You also want to grab two image files — `background_page.gif` and `header.gif`. The HTML files for various other screen shots in this chapter depicting Web pages we've built are also part of the Chapter 1 downloads (there's no file named `01fig01.html` in this collection — that's Uncle Sam's page! — but you will find pages named `01fig04.html` and `01fig05.html`).