Chapter 1

Getting Acquainted with HTML and XHTML

In this chapter, we'll introduce you to HTML and XHTML, which, as you'll see, are markup languages that you can use to develop documents for a variety of online uses. This chapter aims to help you build the vocabulary and basic understanding you'll need to actually *use* HTML and XHTML.

Specifically, in this chapter you'll find the following topics:

- Identifying uses for HTML and XHTML
- Understanding the difference between HTML and XHTML
- Determining whether HTML or XHTML is better for your needs
- Learning the tools you need to use HTML and XHTML
- Recognizing HTML and XHTML code
- Applying HTML and XHTML markup correctly
- Applying rules specific to XHTML markup
- Accessing other HTML and XHTML resources

Why Use HTML or XHTML?

HTML and XHTML are both markup languages. A *markup language* is a system of codes that identify parts and characteristics of documents published online. Before you say, "Yikes, that sounds techie!" consider that you're probably already familiar with the most common application for HTML and XHTML—Web pages. HTML and XHTML are not what you see on screen as a Web page; instead, they're the behind-the-scenes "code" that tells Web browsers what to display.

Let's look at some possible uses for HTML and XHTML:

Developing Web sites Using HTML- or XHTML-developed Web pages and sites, users can jump from topic to topic, view images, fill out forms, submit information, and search databases, among many other possibilities. Basically, these markup languages provide many of the capabilities and functionality that you're accustomed to using when you visit Web sites.

Developing intranet or extranet sites HTML and XHTML are commonly used to develop *intranet sites*—Web sites accessed by people within a company or organization from one or more locations—or *extranet sites*, used by people from a specific group of companies or organizations that routinely share information among themselves.

Creating help files or documentation HTML and XHTML can also be used to develop online help files that are accessible on any platform. Online help files allow developers to produce documentation inexpensively.

Developing network applications HTML and XHTML are even used for developing applications, such as training programs or online classes, or for providing access to databases through Web pages.

Developing kiosk applications You can also use HTML and XHTML to help implement kiosk applications—those stand-alone computers with the neat touch-screen capabilities.

Delivering information via wireless devices HTML and XHTML can be used to display documents on a variety of wireless devices, including Web-enabled phones, personal digital assistants, and handheld computers. It's expected that this will be an area of tremendous growth over the next few years.

What's the Difference Between HTML and XHTML?

In general, you'll find that HTML and XHTML are very similar, and most Web browsers handle both equally well. The main differences include:

XHTML is pickier about how you type the code (the markup) into the documents. As you'll see later in this chapter, XHTML requires your documents and markup to follow very specific rules. For example, whereas HTML lets you use uppercase, lowercase, or a combination of the two when marking up your documents, XHTML requires that you use only lowercase letters. Similarly, although you can get away with using only an opening element in HTML—say, using just an opening paragraph element (`<p>`) to mark a paragraph—XHTML requires that you always use both opening and closing tags, like this: `<p>...</p>`. These XHTML rules aren't hard to learn or follow, but you will need to pay special attention as you develop XHTML documents and make sure all of the details are accurate and complete.

NOTE *If you're familiar with HTML, you're probably used to hearing the word* tag *to describe HTML markup. You might think of tags as being generally equivalent to "elements," with "tag" being the less formal term. Technically, the tag (or tag pair) refers to the elements and the surrounding <> and </> characters. Throughout this book, we'll generally use the term* element *to refer to both the perhaps more familiar "tag" and the more accurate "element," unless we particularly need to emphasize the components of a tag—as we will later in this chapter.*

XHTML can provide more capabilities and flexibility than HTML does. XHTML is based on a larger markup language called *XML* (Extensible Markup Language, which we'll cover in Part VI), which provides developers with powerful capabilities and flexibility far beyond what HTML offers. For example, suppose you want to automatically generate a table of contents for your Web site pages. Because XHTML documents have more structure and are more predictable than HTML documents, you'll find that automatically extracting specific pieces of XHTML documents to generate this content is easier than extracting specific pieces of HTML. In general, XHTML gives you a bit more power and flexibility than you have with straight HTML, yet it doesn't have XML's steep learning curve.

XHTML can provide a good start if you or your company may move to using a more structured markup language in the future. While XHTML doesn't have the depth of structure that XML or SGML (Standard Generalized Markup Language) offers, it does have basic document-level structure and regularity. So, while transferring from using XHTML to one of these markup languages is a complex undertaking, XHTML will give you a far better start than HTML would.

So, are we saying that you should choose XHTML rather than HTML? The short answer is that, really, there's no reason not to use XHTML. In fact, we would encourage you (even if you're new to markup languages or new to developing documents for online uses) to use XHTML. It's not that much harder to learn and use than HTML, following the rules can provide you with some good planning and document-development skills, and it gives you future options and flexibility.

TIP *Web browsers handle HTML and XHTML equally well, and XHTML is not much more difficult to implement than HTML (and it's more flexible). So, we recommend using XHTML in most cases.*

The long answer is, first consider what your needs are. Are you developing a simple personal Web site? Documents for an intranet or extranet site where only HTML is being used? Documents that have to be developed as quickly as possible, then will be discarded shortly thereafter? Or documents that will need to be maintained by very un-technical folks? If so, then HTML might be the better choice for you.

If, however, you are developing more than a minimal Web site, a site that will likely require some amount of maintenance or revision, or a site in which other parts are already using XHTML, then using XHTML is highly recommended. Similarly, if you're developing documents for online help, kiosks, or wireless devices, we'd highly recommend using XHTML simply because it's more flexible, forward-looking, and compatible with XML (on which XHTML is based).

NOTE *Throughout this book, we'll be primarily using XHTML for examples of documents and code. Note that these examples will work equally well whether you're using HTML or XHTML.*

Where Did HTML and XHTML Come From?

The roots of *HTML* (which is the predecessor to XHTML) go back to the late 1980s and early 1990s. That's when Tim Berners-Lee first developed HTML to provide a simple way for scientists at CERN (a particle physics laboratory in Geneva, Switzerland) to exchange reports and research results on the Web. HTML is based on a formal definition created using a powerful *meta-language*—a language used to create other languages—called the *Standard Generalized Markup Language (SGML)*. SGML is an International Organization for Standardization (ISO) standard tool designed to create markup languages of many kinds.

By the early 1990s, the power and reach of the World Wide Web was becoming well known, and CERN released HTML for unrestricted public use. CERN eventually turned HTML over to an industry group called the World Wide Web Consortium (W3C), which continues to govern HTML and related markup-language specifications. Public release of HTML (and its companion protocol, the *Hypertext Transfer Protocol, or HTTP*, which is what browsers use to request Web pages, and what Web servers use to respond to such requests) launched the Web revolution that has changed the face of computing and the Internet forever.

In the years since HTML became a public standard, HTML has been the focus of great interest, attention, and use. The original definition of HTML provided a way for us to identify and mark up *content*—specific information judged to be of sufficient importance to deliver online—without worrying too much about how that information looked, or how it was presented and formatted on the user's computer display. But as commercial interest in the Web exploded, graphic designers and typographers involved in Web design found themselves wishing for the kind of presentation and layout controls that they received from software such as PageMaker and QuarkXPress. HTML was never designed as a full-fledged presentation tool, but it was being pulled strongly in that direction, often by browser vendors, such as Microsoft and Netscape, who sought market share for their software by accommodating the desires of their audience.

Unfortunately, these browser-specific implementations resulted in variations in the HTML language definitions that weren't supported by all browsers and resulted in functionality that wasn't part of any official HTML language definition. Web designers found themselves in a pickle—forced either to build Web pages for the lowest common denominator that all browsers could support, or to build Web pages that targeted specific browsers that not all users could necessarily view or appreciate.

Basically, XHTML was created as a means to address this discrepancy. XHTML provides a way to take advantage of a newer, more compact underlying meta-language called XML (Extensible Markup Language) that is inherently extensible and, therefore, open-ended. More important, XHTML helps rationalize and consolidate a Web markup landscape that had become highly fragmented (a result of different and incompatible implementations of HTML). There are many other good reasons for using standard-compliant HTML or XHTML, and you'll learn more about them later in this chapter and throughout this book.

The HTML and XHTML specifications—all versions, revisions, and updates—are maintained by the W3C and can be found at `www.w3.org`. Taking a tour through the W3C Web pages can be quite useful, in terms of learning what issues are considered most pressing or least important to the people making the specifications. In addition, you can find out if your personal concerns are being addressed.

Continued on next page

Where Did HTML and XHTML Come From? *(continued)*

At the time this book was written, the primary issues being addressed by the HTML Working Group were pushing XHTML in the direction of modularization (Chapter 25) and toward use of Schemas (Chapter 29, on the Web). Both of these issues reflect the importance of moving HTML toward standards that are:

- Compatible with existing browsers and tools
- Flexible enough to accommodate non-traditional browsers, like handheld computers and Web-enabled cell phones
- Consistent and predictable enough to be readily parsed and processed by Web servers and other programs to provide tailored content

That said, you do not need to worry about the future of HTML as you know it. No plans are under way to make any changes or improvements to HTML 4.01, and XHTML 1.0 is explicitly defined as being a "reformulation" of HTML 4.01 in XML; therefore, your investment of time and effort in learning HTML or XHTML won't be wasted. In our opinion, given the number of Web pages out there, there's virtually no possibility that HTML 4.01 or XHTML will change substantially or cease to exist in the next several years—high-end Web sites and information delivery systems will evolve to take advantage of the new features that the W3C defines, but all of the basics (for example, everything in this book) will continue to work for the foreseeable future.

You'll find more history about markup languages in Chapter 21, which discusses XML in the context of how it fits into HTML and SGML (Standard Generalized Markup Language).

What Tools Do You Need?

Whether you're planning to develop HTML or XHTML documents, you'll need three basic tools:

- A plain-vanilla text editor, which you will use to create and save your documents. The "HTML and XHTML Editors" sidebar provides more information about some of the editors that are available; however, keep in mind that we highly recommend using a plain-text editor for developing HTML and XHTML documents—at least when you're learning to use these markup languages.
- A Web browser, which you will use to view and test your documents. In fact, you should have multiple Web browsers available so that you can see how your documents look when viewed in these different browsers.
- A validator, which you will use to verify that your HTML or XHTML documents are developed and coded correctly. We recommend using a validator if you're developing HTML documents; however, because browsers can often display HTML code that's a bit sloppy or not completely "standard," validating your HTML documents isn't an absolute must. If you're developing XHTML documents, however, you *must* use a validator as part of your document-development and publishing process. The XHTML standard has specific rules to follow, and if you want to use XHTML, you must ensure that your XHTML code is applied correctly.

Let's take a more in-depth look at these necessary tools.

Text Editors

Text editors force you to *hand-code,* meaning that you, not the software, enter the code. Hand-coding helps you learn HTML and XHTML elements, attributes, and structures, and it lets you see exactly where you've made mistakes. Also, with hand-coding, you can easily include the newest enhancements in your documents, whereas WYSIWYG editors don't have those enhancements available until updated product versions are released.

Some basic text editors are:

- Notepad for all Windows versions
- vi or pico (command line), or GEdit or Kate (GUI) for Linux/Unix
- TeachText or SimpleText for Macintosh

WARNING *Using a word processing program such as Word, StarOffice, or even WordPad to create HTML documents often inadvertently introduces unwanted formatting and control characters, which can cause problems. HTML and XHTML require plain text, so either make a special effort to save all documents as plain text within such applications, or take our advice and use a text editor instead.*

HTML and XHTML Editors

In general, editors fall into two categories:

Text- or code-based editors, which show you the HTML or XHTML code as you're creating documents A variety of code-based HTML and XHTML editors exist for Windows, Macintosh, and Linux/Unix, and most of them are fairly easy to use (and create HTML code just as you wrote it). That said, as we write this chapter, few editors are available that produce XHTML code (or rigorously standard-compliant HTML code, for that matter).

However, it's possible to use an HTML editor or a simple text editor to create an initial version of your XHTML documents and then to make use of a special-purpose tool, such as HTML Tidy or HTML-Kit, to transform your HTML into equivalent, properly formatted XHTML. Because this requires a bit more savvy than we assume from our general readership, this material is aimed only at those more experienced readers to whom this kind of approach makes sense.

WYSIWYG ("What You See Is What You Get") editors, which show the results of code, similar to the way it will appear in a browser, as you're formatting your document Simple WYSIWYG editors, such as Netscape Composer and Microsoft FrontPage Express, are good for quickly generating HTML documents. These editors give you a close approximation of page layout, design, and colors, and are good for viewing the general arrangement of features. However, they do not give you as much control over the final appearance of your document as code-based editors do. Additionally, although there have been improvements over the last couple of years, WYSIWYG editors notoriously introduce unneeded elements, non-compliant code, and other characteristics that purists object to.

After you've developed a few HTML documents and understand basic HTML principles, you may choose to use both a WYSIWYG editor and a code-based editor. For example, you can get a good start on your document using a WYSIWYG HTML editor and then polish it (or fix it) using a code-based one. Others prefer Web editing/publishing tools like Dreamweaver, which offers a combined approach with both WYSIWYG and code-based modes.

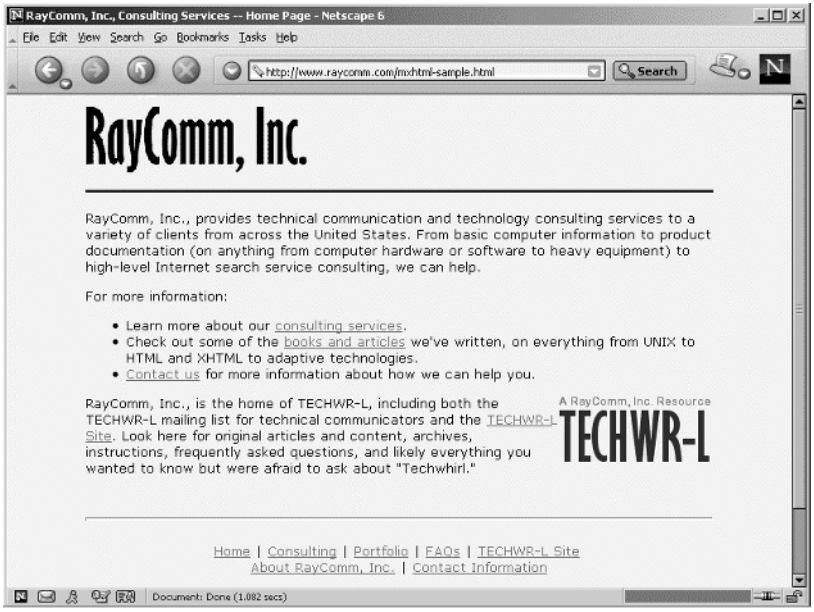
For now, though, we recommend that you hand-code HTML and XHTML using a plain-text editor.

Web Browsers

The most common browsers are Microsoft Internet Explorer (IE) and Netscape Navigator; however, many other browsers are also available for virtually all computer platforms and online services. (AOL's browser is based on IE and functionally nearly the same.) We're especially fond of Opera (available free at `www.opera.com`) and Amaya (available free at `www.w3.org/Amaya/`) because they often support advanced features and functions better and sooner than the more popular IE and Netscape browsers do.

As you're developing HTML or XHTML documents, keep in mind that exactly how your documents appear varies from browser to browser and from computer to computer. For example, most browsers in use today are *graphical browsers*: they can display elements other than text. A *text-only* browser can display—you guessed it—only text. How your documents appear in each of these types of browsers differs significantly, as shown in Figures 1.1 and 1.2.

FIGURE 1.1
An HTML document displayed in Netscape Navigator



Even graphical browsers tend to display things a bit differently. For example, one browser might display a first-level heading as 15-point Times New Roman bold, whereas another might display the same heading as 14-point Arial italic. In both cases, the browser displays the heading bigger and more emphasized than regular text, but the specific text characteristics vary. Figures 1.3 and 1.4 show how two other browsers display the same XHTML document.

FIGURE 1.2
The same HTML document viewed in Lynx, a text-only browser

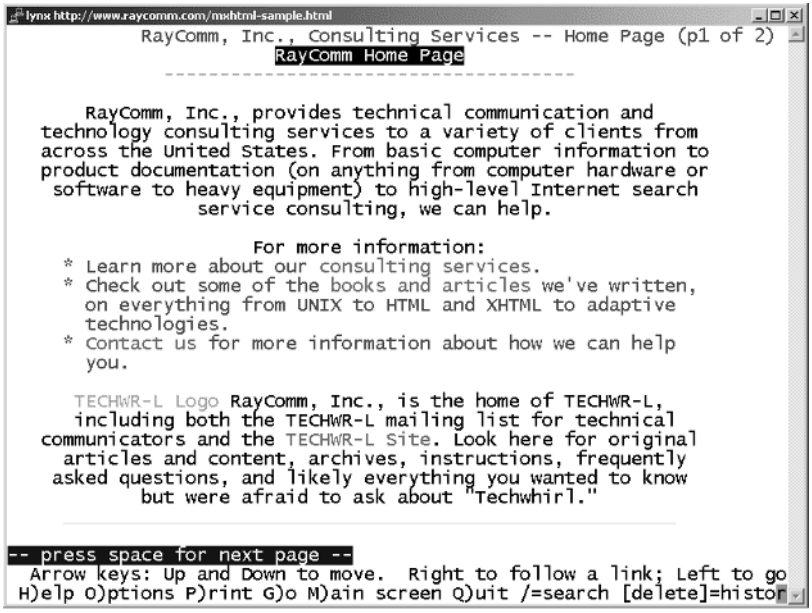


FIGURE 1.3
The W3C Amaya browser has its own unique look and feel

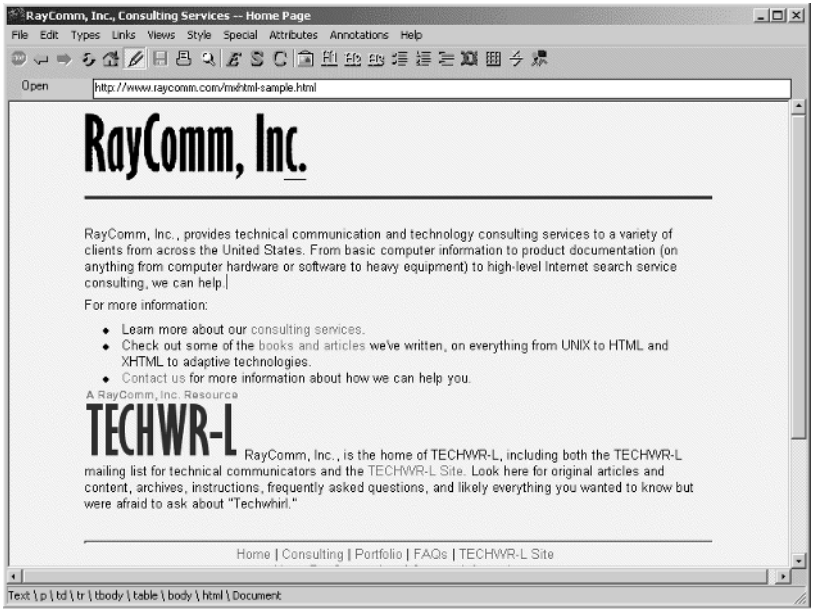
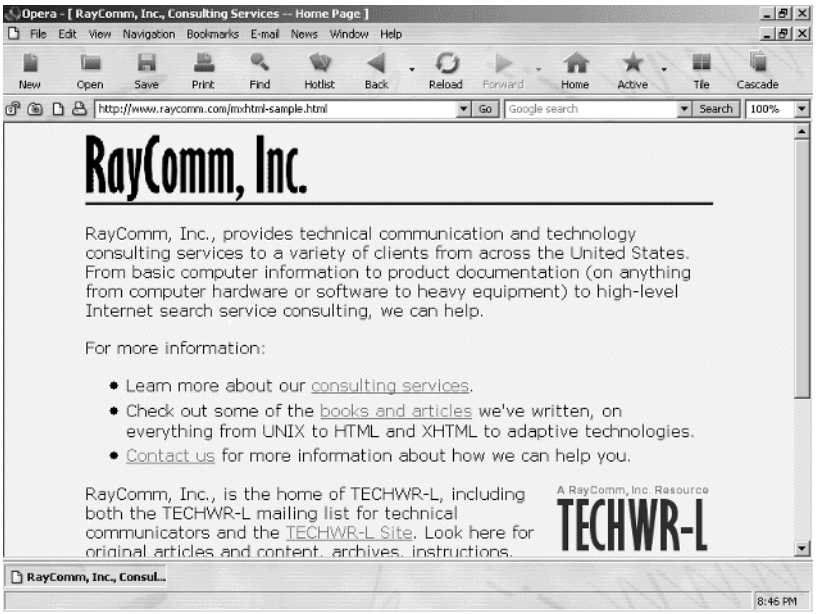


FIGURE 1.4
The Opera browser shows the same document with slightly different formatting



TIP *Finally, your user's computer settings can also make a big difference in how your HTML or XHTML documents appear. For example, the computer's resolution and specific browser settings can alter a document's appearance.*

So, as you're developing and viewing your documents, remember that what you see may look a bit different to your users. Test your documents in as many different browsers as possible, at as many different resolutions and color settings as possible, on as many different computers as possible. You won't be able to test for all variations, but you should be able to get a good idea of what your users might see.

The W3C Validator

You should also use an HTML or XHTML *validator*, which is a tool that examines your documents and verifies that the documents follow the rules for applying code and document structure.

Although a number of validators exist, the W3C Validator is the most definitive, because it's developed by the same folks who developed the HTML and XHTML specification. To use it, first open your browser. Then:

1. Go to `http://validator.w3.org/file-upload.html`.
2. Browse your local hard disk, and upload the file you want using the validator's interface.

If you're in luck, what you get back looks like what's shown in Figure 1.5. If you're not in luck, you will need to find out how to read and interpret the validator's sometimes-cryptic error messages. Because this interpretation is a substantial chore, we've devoted Chapter 20 to this topic; you may want to read it through before your first encounter with the validator.

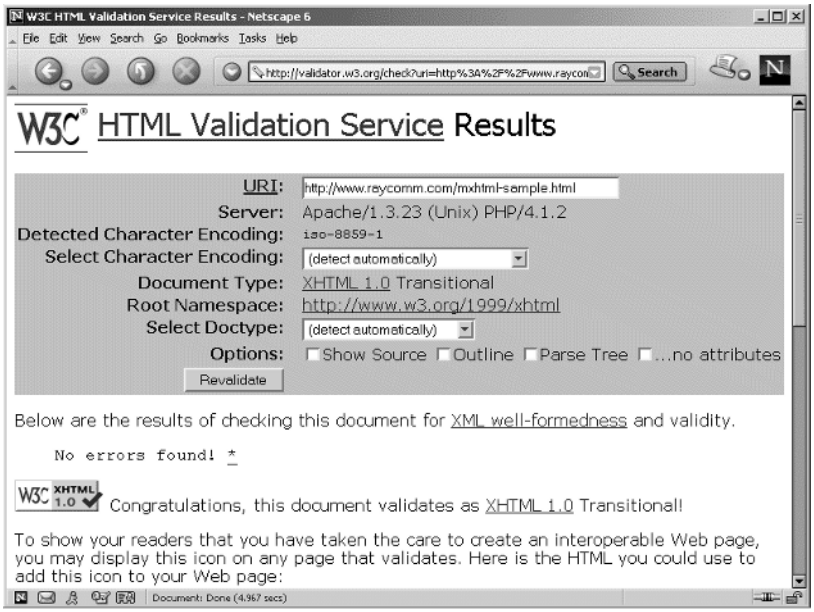


FIGURE 1.5
When the W3C validator finds no errors, its output is both short and very sweet!

You should make validation part of your standard authoring process. That way, you'll get the best possible guarantee that most browsers will be able to view and display the contents of your documents.

NOTE *Remember, validating your XHTML is necessary to ensure that it really is compliant with the XHTML standards.*

What Does HTML and XHTML Code Look Like?

As Figure 1.6 shows, HTML and XHTML documents are plain-text files. They contain no images, no sounds, no videos, and no animations; however, they can include *pointers*, or links, to these file types, which is how Web pages end up looking as if they contain non-text elements.

As you can see, HTML and XHTML documents look nothing like the Web pages you view in your browser. Instead, documents are made up of *elements* and *attributes* that work together to identify document parts and tell browsers how to display them. Listing 1.1 shows the elements and attributes that create the Web page shown in Figures 1.1 through 1.4.

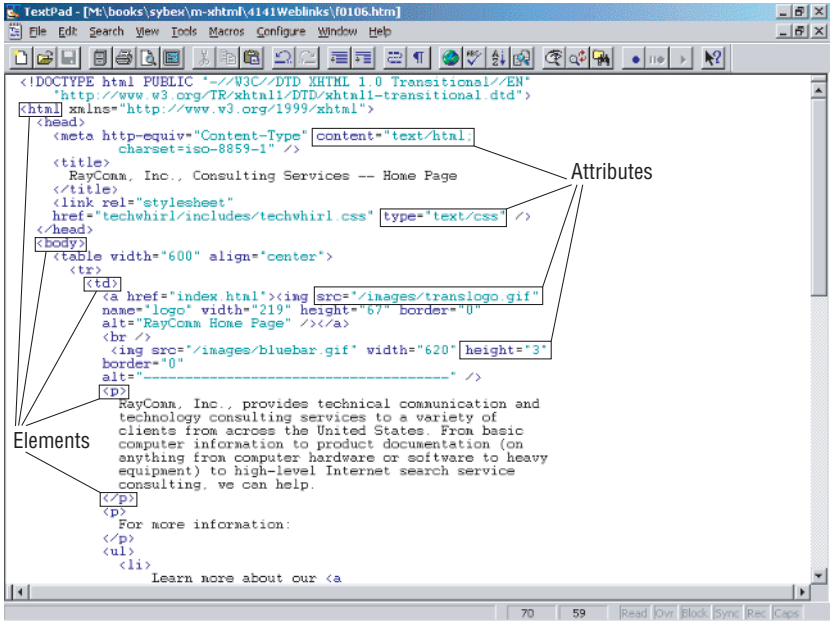


FIGURE 1.6
HTML and XHTML documents are just text files, containing the code and content you provide

LISTING 1.1: HTML AND XHTML CODE INCLUDES CONTEXT, ELEMENTS, ATTRIBUTES, AND LINKS THAT FORM A 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=iso-8859-1" />
    <title>
      RayComm, Inc., Consulting Services -- Home Page
    </title>
    <link rel="stylesheet"
    href="techwhirl/includes/techwhirl.css" type="text/css" />
  </head>
  <body>
    <table width="600" align="center">
      <tr>
        <td>
          <a href="index.html"><img src="/images/translogo.gif"
          name="logo" width="219" height="67" border="0"
          alt="RayComm Home Page" /></a>
          <br />
           <img src="/images/bluebar.gif" width="620"
```

```
 height="3" border="0"
alt="-------------------------------------" />
<p>
 RayComm, Inc., provides technical communication and
 technology consulting services to a variety of
 clients from across the United States. From basic
 computer information to product documentation (on
 anything from computer hardware or software to heavy
 equipment) to high-level Internet search service
 consulting, we can help.
</p>
<p>
  For more information:
</p>
<ul>
  <li>
     Learn more about our <a
     href="consulting.html">consulting services</a>.
  </li>
  <li>
    Check out some of the <a href="books.html">books
    and articles</a> we've written, on everything
    from UNIX to HTML and XHTML to adaptive
    technologies.
  </li>
  <li>
    <a href="contact.html">Contact us</a>
      for more information about how we can help you.
  </li>
</ul>
<p>
 <img src="/techwhirl/images/techwhirllogo.gif"
 alt="TECHWR-L Logo" align="right" /> RayComm, Inc.,
 is the home of TECHWR-L, including both the TECHWR-L
 mailing list for technical communicators and the <a
 href="/techwhirl/">TECHWR-L Site</a>. Look here for
 original articles and content, archives,
 instructions, frequently asked questions, and likely
 everything you wanted to know but were afraid to ask
 about "Techwhirl."
</p>
<br clear="all" />
<hr />
<p class="centered">
  <a href="index.html">Home</a> | <a
  href="consulting.html">Consulting</a> | <a
  href="portfolio.html">Portfolio</a> | <a
  href="faqs.html">FAQs</a> | <a
```

```
            href="/techwhirl/index.html">TECHWR-L Site</a>
            <br />
            <a href="aboutraycomm.html">About RayComm, Inc.</a>
            | <a href="contact.html">Contact Information</a>
          </p>
          <p class="centered">
            Last modified on 1 April, 2002
            <br />
            Site contents Copyright &copy; 1997 - 2002 RayComm,
            Inc.
            <br />
             Send comments to <a
            href="mailto:webmaster@raycomm.com">webmaster@raycomm.com</a>.
            <br />
          </p>
        </td>
      </tr>
    </table>
  </body>
</html>
```

TIP *Throughout this book, we use the term* users *to describe the people who view and use the HTML and XHTML documents you develop.*

Understanding Elements

HTML and XHTML elements serve two primary functions. First, they identify *logical document parts*—that is, major structural components in documents, such as headings (`h1`, a heading level 1, for example), numbered lists (`ol`, also called ordered lists), and paragraphs (`p`). For example, if you want to include a paragraph component in an HTML or XHTML document, you type the text and apply the appropriate elements (`<p>` to the beginning of the paragraph and `</p>` at the end) to that text, as this snippet from Listing 1.1 shows:

```
<p>
  RayComm, Inc., provides technical communication and
  technology consulting services to a variety of
  clients from across the United States. From basic
  computer information to product documentation (on
  anything from computer hardware or software to heavy
  equipment) to high-level Internet search service
  consulting, we can help.
</p>
```

And, voila, the paragraph element (`<p>...</p>`) marks that document part to be a paragraph.

Second, some elements refer to other things that are not included in the HTML or XHTML document itself. Whereas the `<p>` and `<h1>` elements just mentioned refer to paragraph and heading components within the document itself, elements can also mark *pointers*—essentially just links—to other documents, images, sound files, video files, multimedia applications, animations, applets, and so on. For example, if you want to include an image of your company's product in your document, rather than pasting an image directly into the document (as you might in a word processing file), you include an element that points to the file location of that image, as shown here:

```
<img src="logo.gif" alt="logo" />
```

In this example, the `img` (image) element points to a logo file (`logo.gif`) that the browser should display. This illustrates that browsers rely on information within the HTML or XHTML document to tell them what to display, as well as how to display it.

Understanding Attributes

Some HTML and XHTML elements take modifying values called *attributes*, which provide additional information about the elements, such as:

- What other files should be accessed, such as an image file
- What alternative text should be associated with the element
- Which style classes should be used to format the element

Let's assume you want to center a heading 1 in the browser window. You'd start with your heading and elements, like this:

```
<h1>A heading goes here</h1>
```

Next, you'd add the `style` and `type` attributes to the opening element, like this:

```
<h1 style="text-align:center">A centered heading goes here</h1>
```

In this example, the heading level one element includes attributes that specify the style to be aligned in the center. As you can see, attributes normally have two parts: the attribute name (`style=`, in this example) and the value (`"text-align:center"`). The value *should* appear in quotes in HTML and *must* appear in quotes in XHTML.

About Common Attributes

In HTML and XHTML, there are several attributes that can be applied to nearly all elements; these are known as the *common* attributes. They include:

id="name" Assigns a unique name to an element within a document.

style="style" Allows the author of the document to use Cascading Style Sheets (CSS) as attribute values or to define the presentation parameters for that specific element. You can use the `style` attribute with all elements except `html`, `head`, `title`, `meta`, `style`, `script`, `param`, `base`, and `basefont`.

class="name" Assigns a class or a set of classes to an element. This attribute is frequently used with CSS to establish the display properties for a particular subset of elements.

Continued on next page

About Common Attributes *(continued)*

lang="language code" Specifies the language of the content contained by the element. For example, `lang="en"` declares that English is the language used.

dir="ltr|rtl" Specifies the direction text should be displayed. This doesn't seem like an important attribute unless you remember that many of the world's languages are not read from left to right. Yes, `ltr` means "left to right," and `rtl` means "right to left."

title="text" Functions in a manner similar to the `title` element but applies only to a specific element instead of an entire document. Caveat: The attribute's *behavior* is not defined by the HTML or XHTML specification. Instead, the way that behavior is rendered is left up to the browser, and the content is usually presented as a pop-up tooltip when readers hover the mouse pointer over the text. This attribute is currently most useful on sites or documents that the author knows will be viewed by users of Internet Explorer 5, Opera 6, or Netscape 6 (or later) browsers. The `title` attribute cannot be used with the following elements: `html`, `head`, `meta`, `title`, `script`, `param`, `base`, and `basefont`.

Typing Elements and Attributes Correctly

As our examples so far should illustrate, elements and attributes are reasonably intuitive. Although markup can occasionally be cryptic, you can generally get some idea of an element or attribute function from its name.

Before we get started on typing elements and attributes, be aware that entering elements and attributes varies *slightly* depending on whether you're using HTML or XHTML. Remember that we mentioned XHTML is a bit pickier? Well, entering XHTML markup is where that pickiness comes into play. Again, though, XHTML is not harder; you'll just have to pay a bit more attention as you're learning to use it. In the next sections, we'll do the following:

- Describe general guidelines for typing HTML and XHTML elements and attributes
- Show you how to *nest* elements (apply more than one element to a document part)
- Explain specific rules for typing XHTML elements and attributes
- Help you improve readability of your HTML and XHTML documents

Typing Elements and Attributes in Either HTML or XHTML (General Information)

To begin, all elements are composed of *element names* that are contained within *angle brackets* (< >). The angle brackets simply tell browsers that the text between them represents HTML or XHTML markup rather than ordinary text content. Some sample elements look like these:

- `<h2>` (for heading level 2)
- `<p>` (for document paragraph)
- `<strong>` (to emphasize a particular section of content)

Tip *You'll learn more about these elements and their uses in Chapter 2.*

Second, many elements are designed to contain content; they use a pair of tags, where actual content occurs between the *opening tag* (for example, `<h1>`) and the corresponding *closing tag* (`</h1>`). Both tags look alike, except the closing tag includes a forward slash (`/`) to denote the end of the element container. To apply tags to something in your document, place an opening tag before the content that should be associated with the element you want to use, and place the closing tag after it, as these examples show:

```
<h1>Information to which the tags apply</h1>
```

or

```
<title>Correctly Formed Title</title>
```

When typing elements, be particularly careful *not* to include extra spaces within the tag itself, as in this erroneous example:

```
< title >Incorrectly Formed Title< /title >
```

If you include spaces within the elements, browsers may not recognize the element and may not display the content correctly (or at all). Sometimes, a browser might display the markup itself because it's unable to distinguish improperly formed markup from normal element content.

TIP *When creating HTML or XHTML markup by hand, enter both the opening and closing tags at the same time. That way, you won't forget the closing tag.*

Keep in mind that you'll also use the occasional *empty elements*, which do not include a closing tag. Some empty elements include the line break element (`<br />`) and the image element (`<img />`), which do not require the closing element but include a space and a forward slash (`/`)after the element name. This example shows the break element (`<br />`), which would put in a line break after each line:

```
<p>
  RayComm, Inc., provides technical communication and<br />
  technology consulting services to a variety of<br />
  clients from across the United States. From basic<br />
  computer information to product documentation (on<br />
  anything from computer hardware or software to heavy<br />
  equipment) to high-level Internet search service<br />
  consulting, we can help.
</p>
```

We'll point out these empty elements throughout this book and show you how to use them correctly.

NOTE *Empty elements in HTML—as opposed to XHTML—do not require the closing / character. However, the extra / causes no problems in HTML, so we customarily use it in both HTML and XHTML.*

As we discussed, elements don't usually appear by themselves; often you'll also include attributes that provide supporting information about the element. The `style=` attribute in this example indicates that the heading level 1 should be centered:

```
<h1 style="text-align:center">A centered heading goes here</h1>
```

As you enter attributes, remember these guidelines:

- Include the attribute within the element after the element name, as in `<h1 style="text-align:center">`.
- Use spaces to separate attributes from other attributes and the element itself, as in `<h1 id="5325a" style="text-align:center">`.
- Enclose attribute values in quotes, as in `style="text-align:center"`. The quotes are required in XHTML and part of developing "correct" HTML. Learning to include them now will help you in the future, when newer specs are released that insist on this convention.

Finally, be aware that:

- HTML allows you to type in elements (and attributes) using uppercase, lowercase, or a combination of both. It's not picky, and browsers will display HTML code using whichever capitalization choice you make.
- XHTML requires that elements and attributes—with a few exceptions—must be typed using all lowercase. We'll point out exceptions throughout this book; however, we recommend that you use lowercase for both HTML (for good practice, should you ever move to XHTML in the future) and XHTML (because lowercase is required). The examples in this book will all be lowercase.

Applying More than One Element to a Document Component (Nesting Elements)

In the preceding examples, you saw how you apply elements around the text to which they apply. Suppose, though, that you need to apply more than one element to a document component. For example, say you have a paragraph that also includes a few words that you want to emphasize. To apply more than one element to a particular piece of content, you nest the tags. *Nesting* means placing one set of tags inside another set. For example, to apply strong emphasis to a word within a paragraph, you nest the `strong` element within the paragraph `p` element, as follows:

```
<p>The <strong>right</strong> way to use strong emphasis is to
   enclose only those words you want to emphasize inside a
   strong element.</p>
```

When you nest elements, the first opening tag must be matched by a corresponding closing tag at the end of the related block of content, and the second opening tag must be closed with a corresponding closing tag immediately after its related content block. XHTML is quite insistent that you nest tags in the right order. Therefore, a block of text like this:

```
<p>The last word gets strong <strong>emphasis.</p></strong>
```

is invalid because it closes the outside `p` element before closing the nested (or inside) `strong` element. It's also technically incorrect for HTML, but such issues *usually* don't cause problems in HTML.

Typing XHTML Elements and Attributes

With the general information and guidelines for typing elements and attributes established, we'll now take a look at the specific rules you'll need to follow for developing XHTML documents. Although

some of these XHTML rules seem pointless or arbitrary, keep in mind that the XHTML specification is based on the XML specification, and it applies some of the same constraints and conventions of that specification. Although you may not be interested in moving to XML, you still need to apply these rules to your XHTML documents.

Some of the following rules will look familiar; we mentioned a few in the general guidelines for both HTML and XHTML documents.

All XHTML Elements Must Be Closed

All XHTML elements must be balanced with an opening and a closing tag, which is referred to as making the document *well formed*:

```
HTML:  <p>This is a paragraph.
XHTML: <p>This is a paragraph.</p>
```

If you're already familiar with HTML, you're probably wondering what you do with the `img` and `br` elements in XHTML. These (and other) empty elements accept attributes but do not contain character data. Using HTML, these empty elements generally include just the opening tag, like this:

```
<img src="logo.gif" alt="Corporate Logo">
```

XHTML is a tad different in that you have to include markup to terminate the empty element, in one of two ways:

- Terminate the tag with a space and a slash, as follows:
  `<img src="logo.gif" alt="Corporate Logo" />`
- Add a closing tag, as follows:
  `<img src="logo.gif" alt="Corporate Logo"></img>`

The first option saves space and time, and logically makes a little more sense. You can use either approach, but we suggest (and most developers use) the first option, rather than the latter.

WARNING *In XHTML, most white space within a tag is not significant. For example,* `<img src="logo.gif" alt="Logo"/>` *is the same as* `<img src="logo.gif" alt="Logo" />`*. In the second example, you should notice white space between the closing quotation mark (") and the forward slash (/). However, older browsers have problems interpreting the first example because there is no white space separating these items. If you add a space before the /, you can ensure that most older browsers will interpret the empty element without any problems.*

All Attributes Must Have Values

One of the trickier XHTML rules is that all attributes must have values. At first glance, this may seem to be an easy rule to grasp, but there's a trick or two you have to master when applying it to XHTML. For most attributes, it's fairly straightforward. For example:

```
<table align="center">
```

In this case, the `align` attribute has to have a value (`"center"`) to tell the processor just how to align the table.

As you advance your skills, you'll encounter a few situations where including the value isn't as straightforward. For example, in Chapter 6, we discuss forms, which use the `<input>` element. This element accepts what's called a *stand-alone value* (an attribute where basically the name and the value are the same thing) called `disabled`, which simply turns off the function. The element and attribute might look like this:

```
<input disabled>
```

Or, if you added other attributes, the element would look something like the following:

```
<input disabled name="pet" value="cat">
```

Because this element is empty, we need to terminate it, like this:

```
<input disabled name="pet" value="cat" />
```

Although the preceding attribute is fine according to HTML, it's not well formed according to XHTML, because all attributes must have values. The problem is that these stand-alone attributes do not have any predefined values, and therefore, a value is not really needed. However, according to XHTML's rules, it must have a value, so a workaround was created. You set the attribute equal to itself:

```
<input disabled="disabled" name="pet" value="cat" />
```

All Attribute Values Must Be Enclosed in Quotes ("")

Another XHTML rule is that all attribute values be delimited with quotation marks. HTML allows several attribute values to be defined without quotation marks—although the specification recommends that they always be used.

This rule is easy enough:

```
HTML:  <table align=center>
XHTML: <table align="center">
```

The HTML example does not contain quotation marks, and the XHTML example does.

TIP *In many HTML examples, you're likely to see the attribute value in uppercase (*`CENTER`*). Most attribute* values *are not case sensitive. In this book, we generally use lowercase to define attribute values, for readability.*

XHTML Is Case Sensitive

As we mentioned, XHTML is case-sensitive and requires—with few exceptions—that all elements and attributes be typed using lowercase letters. This may be a sticky point if you've been using HTML and are accustomed to using uppercase or a combination of upper- and lowercase. Nonetheless, using all lowercase letters just takes some getting used to:

```
HTML:  <TABLE>…</TABLE>
XHTML: <table>…</table>
```

Elements Must Be Nested Correctly

In the previous section, we described the concept of nesting, where you apply one or more elements within another element. For example, the following markup defines a `title` element that is nested with the `head` element:

```
<head>
   <title>Document title</title>
</head>
```

This may seem straightforward; however, there are cases where people make mistakes. For example, can you spot the mistake in the following markup?

```
<p>You can bold a <b>word</p></b>
```

The problem is that the tags are overlapping; no one element is nested within the other. The golden rule is "what you open first, you close last." To correct this syntax, you would write:

```
<p>You can bold a <b>word</b></p>
```

Notice how the `b` element is nested completely within the `p` element.

TIP *When referring to an element that is nested within another element, we call the nested element a* child *of the container element; the container is the* parent *element. Throughout this book we refer to nested elements as "children of a parent element."*

There are a few other XHTML-specific rules that apply to certain elements and attributes. We'll mention these as we come to them in this book. The XHTML rules described here are the ones you will need to know as you're getting started.

Improving Document Readability

Throughout the book, because of the width limits of the printed page, we wrap and indent code lines that are meant to be written all on one line. This doesn't mean you have to wrap or indent the code you develop, but we do recommend using hard returns in your code to help make the lines a bit shorter and easier to read. Doing so does not affect how browsers display your documents (unless you inadvertently put in a space between an angle bracket and an element name); it just makes that document easier for you and others who may have to maintain the code to read when you're editing its contents.

For example, take a look at this code:

```
<!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><title>
Mastering HTML Document Title</title></head><body>Mastering
HTML Document Body</body></html>
```

Although we've included line breaks so that the code won't run off the book's page, we could improve the readability by separating some of the code a bit, like this:

```
<!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>
      <title>Mastering HTML Document Title</title>
   </head>
   <body>
       Mastering HTML Document Body
   </body>
</html>
```

No question which one's easier to read or follow, right?

What Other Resources Can Help?

In addition to this book, you can find information, resources, and specifications on the Web. In particular, the W3C site, as well as several product-specific Web sites, will help you learn, use, and keep up with changes in HTML (unlikely to change) and XHTML (somewhat more likely to change).

Visit the W3C

The W3C was created in 1994 at the Massachusetts Institute of Technology (MIT) to oversee the development of Web standards, eventually including the XHTML standard. This consortium defines and publishes HTML, XHTML, and numerous other Web-related standards, along with information about the elements and attributes that may legally appear within HTML or XHTML documents. So, an excellent way to monitor changes is to visit the W3C site at `www.w3.org/MarkUp`. There you'll find new releases of XHTML standards and information about HTML standards.

For more information on proposed standards and other developments in Web-related specifications, such as Cascading Style Sheets (CSS) and XML specifications, visit the W3C's home page at `www.w3.org`.

Can you use new elements and attributes as they become available? For the most part, yes. By the time many popular elements and attributes become part of a standard, they already work with many or most browsers. However, some elements and attributes (including some that were introduced with HTML 4) did not have wide or stable browser support when that specification was released and, to this day, do not have nearly the breadth of support that some other elements and attributes enjoy. We'll point these out throughout this book and show you how they differ from previous versions of HTML.

Monitor Netscape and Microsoft Sites

When HTML was the prevailing Web markup standard, each time Netscape or Microsoft released a new browser version, it would also add new markup *extensions*, which are browser-specific, nonstandard elements and attributes. Some of these extensions were useful, some less so. However, as a whole, any nonstandard elements introduced into HTML caused problems both for Web developers and for users. Fortunately, far fewer extensions seem to be introduced now that XHTML has made the scene, but you should still be aware of what's added with each new browser release, if only to know what progress the browsers have made in supporting the elements and styles that are already defined.

If you're considering using extensions in your XHTML documents, keep in mind that they're not standard and that the W3C validator will not recognize or validate nonstandard markup. Also, extensions that are specific to a particular browser (for example, Netscape) will probably not work

in other browsers (such as IE or Opera). For this reason, we strongly recommend that you refrain from using extensions and use only standard HTML or XHTML elements and attributes. This way, you'll not only be able to validate your documents to make sure they're syntactically correct, but you can also be reasonably sure that all your users can access the information you provide therein.

You can find Netscape's elements and attributes at

```
http://developer.netscape.com/docs/manuals/htmlguid/index.htm
```

And you will find Microsoft's elements and attributes at

```
http://msdn.microsoft.com/library/
```

under the Web Development, HTML subsections. (Note that these pages move frequently, so you may need to browse a little to get there.)

Monitor Other Sites

Although definitive information comes from the W3C, you should also check other reliable resources for information about HTML and XHTML. Here's a list of sites to check regularly.

Organization	URL
Web Design Group	`www.htmlhelp.com`
Web Developer's Virtual Library	`www.wdvl.com`
HTML Writer's Guild	`www.hwg.org`
WebMonkey	`www.webmonkey.com`
CNET's Builder.com	`www.builder.com`
Oasis	`www.oasis-open.org`
Zvon	`www.zvon.org`
Google Web Directory	`http://directory.google.com/Top/Computers/Data_Formats/Markup_Languages/HTML/References/`
WebReference.com	`www.webreference.com/`

Where to Go from Here

This chapter gave you a brief overview of HTML and XHTML—what they are, what they're used for, how to enter their tags and attributes correctly, and what supplemental resources might be helpful to you. Although you haven't done any HTML or XHTML coding yet, you should now possess a good foundation of basic concepts and terminology.

From here, we suggest you proceed to Chapter 2, where you'll learn more details about XHTML document syntax and structure and will create your first HTML or XHTML document. You might also browse Part IV, "Developing Web Sites," to learn about the HTML and XHTML document life cycle as well as about developing and publishing Web sites.