When you complete this chapter, you will be able to:

- Describe the current state of HTML
- Move from HTML to XHTML
- Understand variables in the Web design environment
- Describe browser compatibility issues
- Consider connection speed differences
- Code for multiple screen resolutions
- Address operating system issues

In this chapter, you explore the variable factors that affect Web design. You learn how Hypertext Markup Language (HTML), the language used to create documents on the World Wide Web, is constantly evolving, and preview the new markup languages that are changing how you design for the Web. You’ll see how Web browsers affect the way users view your content, and how variations in the user’s browser choice, screen resolution, and connection speed pose specific challenges to creating Web pages that are displayed properly in different computing platforms. Finally, you consider what type of software tool you should use to create your HTML code.
The Current State of HTML

In this section, you explore the evolution of HTML and its future as a markup language for creating Web documents. You analyze current design limitations of HTML, the need for style sheets that allow separation of style from structure, and the usage of hypertext as a means for organizing information.

HTML: Then and Now

When Tim Berners-Lee first proposed HTML at the European Laboratory for Particle Physics (CERN) in 1989, he was looking for a way to manage and share large amounts of information among colleagues. He proposed a web of documents (at first, he called it a mesh) connected by hypertext links and hosted by computers called hypertext servers. As the idea developed, Berners-Lee named the mesh the World Wide Web. He created an application of the Standard Generalized Markup Language (SGML), a standard system for specifying document structure, and called it the Hypertext Markup Language. HTML greatly reduces the complexity of using SGML to facilitate transmission of documents over the Internet.

When Berners-Lee created HTML, he adopted only the elements of SGML necessary for representing basic office documents such as memos and reports. The first working draft of HTML included elements such as titles, headings, paragraphs, and lists. HTML was intended for simple document structure, not for handling today’s variety of information needs. As the Web evolved and expanded, the demands to transport data for transactions such as shopping and banking online has far outgrown the capabilities of HTML. The need for new markup languages and standards to address these demands is handled by the World Wide Web Consortium (W3C).

HTML and the World Wide Web Consortium

HTML has progressed significantly since it was first formalized in 1992. After the initial surge of interest in HTML and the Web, a need arose for a standards organization to set recommended practices that would guarantee the open nature of the Web. The W3C was founded in 1994 at the Massachusetts Institute of Technology to meet this need. The W3C, led by Tim Berners-Lee, sets standards for HTML and provides an open, nonproprietary forum for industry and academic representatives to add to the evolution of this new medium. The unenviable goal of the W3C is to stay ahead of the development curve in a fast-moving industry. The various committees that make up the W3C look to expand and set standards for the many new Web technologies that have emerged. These include Extensible Hypertext Markup Language (XHTML), Extensible Markup Language (XML), Cascading Style Sheets (CSS), and other markup and style languages. You will learn more about these new companion technologies to HTML later in this chapter.
The Current State of HTML

Visit the W3C site at www.w3.org to find out more about HTML, XML, CSS, and the history and future of the Web. You can look up individual element definitions, test your code for validity, or keep up to date on the latest Web developments.

The Limitations of HTML

HTML is a markup language, a structured language that lets you identify common sections of a document such as headings, paragraphs, and lists. An HTML file includes text and HTML markup (or element) tags that identify these sections. The HTML markup tags indicate how the document sections appear in a browser. For example, the <h1> element tags in the following code indicate that the text is a first-level heading:

```
<h1>Welcome to My Web Page</h1>
```

The browser interprets the HTML markup elements and displays the results, hiding the actual markup tags from the user. In the previous code, the user sees only the text “Welcome to My Web Page” formatted as a level-one heading.

HTML adopts many features of SGML, including the cross-platform compatibility that allows different computers to download and read the same file from the Web. Because HTML is cross-platform compatible, it does not matter whether you are working on a Windows PC, Macintosh, or UNIX computer. You can create HTML files and view them on any computer platform.

HTML is not a What You See Is What You Get (WYSIWYG) layout tool. It was intended only to express logical document structure, not formatting characteristics. Although many current HTML editors let you work with a graphical interface, the underlying code they create is basic HTML. However, because HTML was not designed as a layout language, many editing programs create substandard code to accomplish a certain effect. You cannot rely on the HTML editor’s WYSIWYG view to test your Web pages. Because users can view the same HTML file with different browsers and on different machines, the only way to be sure of what your audience sees is to preview your HTML files in the browsers you anticipate your audience will use.

Despite its limitations, HTML is ideal for the Web because it is an open, nonproprietary language that is cross-platform compatible. All of the markup tags are included with every document and usually can be viewed through your browser. Once you are familiar with the HTML syntax, you will find that one of the best ways to learn new coding techniques is to find a Web page you like and view the source code. (You have a chance to view the source code of a Web page in the Hands-on Projects at the end of this chapter.)

The Need for Style Sheets

Style elements such as <font> were introduced by browser developers to help HTML authors bypass the design limitations of HTML. Designers and writers who are accustomed to working with today’s full-featured word processing programs want the same
ability to manipulate and position objects precisely on a Web page just as they can on a printed page. Again, this is not what HTML was designed to do; as with SGML, HTML was intended to represent document structure, not style.

Mixing style information within the structure, as is the case in most of the Web today, limits the cross-platform compatibility of the content. The display information that is embedded in Web pages is tailored toward one type of display medium, the computer screen. With style sheets, the display properties are separate from the content. This accommodates the diverse variety of devices and users that browse the Web. The Web server can determine the type of requesting device and supply a style sheet that matches the device. Figure 1-1 illustrates this concept.

This separation of style and structure was accomplished in 1996 by the W3C’s specification for a Web style language. The style language, named Cascading Style Sheets (CSS), allows authors to create style rules for elements and express them externally in a document known as a style sheet. CSS rules are easy to create and very powerful. For example, assume that you want all of your <h1> headings to appear green and centered
Moving from HTML to XHTML everywhere on your Web site. For every instance of an <h1> element, you would need to include the following code in a standard HTML document:

```html
<font color="green"><h1 align="center">Some Heading Text</h1></font>
```

Using a CSS rule, you can express the same style as follows:

```css
h1 {color: green; text-align: center;}
```

You can place this rule in an external style sheet, and then link every page on your site to that style sheet; with much less code you can achieve the same result. Later, if you want to change the <h1> color to red, you simply revise the style sheet rule to change every page on your site.

Until recently, the adoption of CSS as a standard for style has been limited because of poor and uneven support by the major browsers. The newer browsers, such as Internet Explorer 6.0, Netscape Navigator 7.0, and Opera 7.0, offer more complete and consistent support for CSS. The current trend is to rely more heavily on style sheets to control the visual display of your content. You will learn more about CSS in later chapters of this book.

Organizing Information with Hypertext

The most engaging aspect of browsing the World Wide Web is the linking of information on related topics using hypertext, a nonlinear way of organizing information. When using a hypertext system, you can jump from one related topic to another, quickly find the information that interests you, and return to your starting point or move onto another related topic of interest. As a hypertext author, you determine which terms to create as hypertext links and where users end up when they click a link.

On the Web, clickable hyperlinks, which can be text or images, can connect you to another Web page, for example, or allow you to open or download a file, such as a music, image, movie, or executable file. Although the basic one-way nature of a hypertext link has not changed since the Web was developed, the nature of the destination content has changed greatly. The different types of linked content and media have continually evolved as the Web has grown into a richer, more interactive environment. Taking advantage of these new technologies in any Web site often requires users to have better connection speeds than they normally have with a modem. You will read more about connection speed as a design variable later in this chapter.

Moving from HTML to XHTML

HTML has progressed through a number of versions since its inception. The latest standard is version 4.01, which was released by the W3C in late 1999. This is the last release of HTML in its current state. The next generation of HTML is called the Extensible Hypertext Markup Language (XHTML). The W3C released version 1.0
of XHTML in January 2000; a revised version was released in August 2002. As defined in the W3C XHTML recommendation (www.w3.org/TR/xhtml1/), there are three “flavors” of XHTML:

- **XHTML Strict**—Use this when you want clean structural markup code, free of any markup tags associated with layout. Use XHTML Strict with Cascading Style Sheets to get the font, color, and layout effects you want. If you are beginning a new Web site, you should code to this recommendation.

- **XHTML Transitional**—This type of XHTML is designed for people writing Web pages for the general public. The idea is to take advantage of XHTML features, including style sheets, but make small adjustments to your markup code for those viewing your pages with older browsers, which can’t understand style sheets.

- **XHTML Frameset**—Use this when you want to use frames to partition the browser window into two or more sections. You can learn more about frames by reading the “Working with Frames” chapter on the Online Companion Web site for this book.

How do these three types of XHTML affect you as a Web developer? Your goal should be to create code that matches the strict recommendation, using Cascading Style Sheets for all of your display information. The benefit of the transitional type is that it allows you to gradually migrate from existing HTML code that may still contain font and display information to the more syntactically correct, cleaner markup code necessary to match the strict type. The frameset specification is important only if you plan to use frames to partition the browser window, as described in the “Working with Frames” chapter posted on the Online Companion Web site for this book.

**A Brief Introduction to XML**

In order to understand XHTML, a brief introduction to XML is necessary. Like HTML, XML is also a subset of SGML, but has no predefined elements such as `<h1>` or `<p>`. The major difference between XML and HTML is that HTML is a predefined set of elements that the browser understands, while XML is a metalanguage. The PC Webopedia (www.pcwebopedia.com) defines the “meta” prefix as meaning “about,” so a metalanguage is a language about a language. XML is thus a language that lets you describe a markup language, allowing you to create your own elements to meet your information needs. This flexibility provides:

- The ability to add new elements or attributes to extend the capabilities of HTML
- The ability to design new browsers or applications for different methods of accessing the Internet

XML code looks very similar to HTML code, with some syntactical differences that you will read about in the next section. The major difference between the two languages is
that XML allows you to create elements that describe any type of information you desire. For example, consider that poets might want to create a markup language that expresses the different parts of a poem, as shown in the following code sample:

```xml
<poem>
  <title>An Ode to the Web</title>
  <stanza>
    <line>So many Web sites</line>
    <line>So little time</line>
    <line>And all I want to do</line>
    <line>Is critique their design!</line>
  </stanza>
</poem>
```

Notice that this code looks very much like regular HTML code, except that the tags are not standard, but specific to the type of content they contain. Unlike standard HTML, the browser does not know how to display this information unless a style sheet is supplied that specifies, for example, that the contents of the `<line>` elements should be displayed in the browser as 12-point Helvetica text.

**Benefits of Moving to XHTML**

One of the significant advantages of making HTML part of XML is that XML has stricter code syntax. As more sites adopt XHTML, they will have to clean up code that does not match the standard. With more Web sites using cleaner code, browsers have to do less work judging what is correct code and what is not. Additionally, XHTML is designed to appear properly in browsers that support HTML 4.0.

HTML was originally designed for limited document expression and has not adapted well to the exploding interest in the Web. Because XHTML is based on XML, it is extensible, which means that designers can extend its capabilities, allowing developers to address future markup needs easily. XHTML is also designed to support the variety of new devices that will access the Internet as new technologies emerge. Any XHTML-compliant software must access and display XHTML regardless of the computer or display type.

Because XML allows better data handling, the new version of XHTML works smoothly with database and workflow applications. The next generation of HTML will include advanced support for form elements, defining them more for data handling than presentation, and allowing data to pass among applications and devices with greater ease. Tables will emphasize a data model that can render their content based on the presentation device. For example, tabular data for stock pricing information could be sent to multiple destinations and displayed to best fit the user’s individual display type, such as a personal digital assistant (PDA) or cell phone. This arrangement gives the same data greater value; though it needs only to be generated once, it can be displayed in many ways.
XHTML Syntax Rules

XML, and therefore XHTML, contain a number of syntax rules that are different from those in HTML. XHTML conforms to the following XML syntax rules:

- Documents must be well-formed.
- Elements must nest symmetrically.
- Element names are case sensitive.
- End tags are required.
- Empty elements are signified by a closing slash.
- Attribute values must be contained in quotes.

Documents Must Be Well-Formed

Well-formed means that a document adheres to the syntax rules described in this section. Any document that does not meet the syntax rules will not be accepted as an XHTML document.

Documents Must Contain an XML Declaration

All XML documents must begin with a document declaration. Because an XHTML document is XML, the recommendation states that documents start with the following XML declaration as the first line of code:

```xml
<?xml version="1.0"?>
```

The opening and closing question marks make this a processing instruction, a special type of XML element.

You currently should not include this declaration in your Web page code. Older browsers do not understand the declaration and may display it as text on the Web page.

Elements Must Nest Correctly

You can nest XML tags, but they must not overlap. Each set of opening and closing tags must completely contain any elements that are nested within. For example, the following is incorrect XML syntax:

```xml
<paragraph><bold>some text… </paragraph></bold>
```

The closing tag for the bold attribute must come before the closing tag for the paragraph element. The correct nesting syntax follows:

```xml
<paragraph><bold> some text… </bold></paragraph>
```
XML Is Case Sensitive

XML, unlike HTML, is case sensitive. An XML browser interprets `<PARAGRAPH>` and `<paragraph>` as two different elements. Although not required, the accepted convention in XML is to use all lowercase characters for element and attribute names.

End Tags Are Required

In HTML, certain elements such as the `<p>` element had optional closing tags. This is not allowed in XML, where nonempty elements need a closing tag. For example, the following two `<p>` elements do not have closing tags:

```html
<p>This is the first paragraph.
<p>This is the second paragraph.
```

In XHTML the following example is correct:

```html
<p>This is the first paragraph.</p>
<p>This is the second paragraph.</p>
```

Empty Elements Are Signified by a Closing Slash

Empty elements must either have a closing tag or be marked empty by a slash (/) in the single tag. For example, the `<br>` element, when rendered in XHTML, becomes `<br/>`. The `<img>` element looks like the following:

```html
<img src="photo.jpg"/>
```

Notice the closing slash. Older browsers ignore this, so you can convert empty elements to be XHTML compliant without worrying about whether your pages are displayed properly.

Attribute Values Must Be Contained in Quotes

All attribute values in XML must be contained within quotes, unlike those in HTML. The following is incorrect XML syntax:

```html
<h1 align=center>Heading</h1>
```

The correct syntax follows:

```html
<h1 align="center">Heading</h1>
```

Adopting XHTML Syntax Rules

If you anticipate working with XHTML in the future, you should consider following these syntax rules in your HTML code now. This ensures that the HTML you are creating today will work with XHTML in the future. If you have legacy HTML code, consider revising
it to meet XHTML syntax standards. The following code shows an example of HTML code that is common on the Web today. Although not syntactically correct, this code is displayed properly in the browser:

```html
<H1>Some plain HTML code</h1>
<P ALIGN=CENTER>This is a paragraph of text.
<IMG SRC="xml.gif">
<H3>A bulleted list</h3>
<UL>
  <LI>Item one
  <LI>Item two
  <LI>Item three
</UL>
```

Converting this code to syntactically correct XHTML means applying the stricter syntax rules listed earlier, resulting in the following code:

```html
<h1>Some plain HTML code</h1>
<p align="center">This is a paragraph of text.</p>
<img src="xml.gif"/>
<h3>A bulleted list</h3>
<ul>
  <li>Item one</li>
  <li>Item two</li>
  <li>Item three</li>
</ul>
```

Many shareware and commercial software programs can assist you in bringing your code up to XHTML standards. These include HTML Tidy at http://tidy.sourceforge.net/ and Tidy GUI at http://perso.wanadoo.fr/ablavier/TidyGUI/.

### Style Sheets Are Required

Because XHTML is an application of XML, you must use style sheets to render style in XHTML. Separating data from style means that the same information can be directed to various display devices simply by changing the style sheet. When different style sheets are used, the contents of the same Web page can be displayed on a computer monitor, TV screen, handheld device, or cellular phone screen. This data–once, destination–many format liberates the data and structure of XHTML documents to be used in a variety of applications. A script or applet redesigns the data presentation as it is requested from the server and applies the proper style sheet based on the user’s choice of device.
The following two style sheet languages are currently available for use with XML or XHTML:

- **Cascading Style Sheets**—CSS has recently gained a lot of popularity based on increasing browser support. CSS is an easy-to-use style language that controls only how documents are displayed. The W3C released the second edition of CSS, called CSS2, in 1998.

- **Extensible Style Language (XSL)**—As an application of XML, XSL both describes page formatting and allows XML documents to be transformed from one type to another. XSL supports the use of CSS style rules within an XSL style sheet, so the two style languages complement each other.

### Migrating from HTML to XHTML

When should you choose to adopt XHTML for your Web site rather than HTML? If you are building a new site from scratch, you can start by coding well-formed XHTML, using CSS for display information. If you have an existing site, you have a larger job ahead of you. Adopting XHTML for existing sites should be a gradual process rather than an abrupt shift to the new language. The transitional flavor of XHTML lets you start to adopt the newer syntax while keeping legacy HTML code such as attributes that control page and link colors. As you migrate closer to strict XHTML, you will be cleaning up code on existing pages, planning coding conventions for new pages, and moving display information to CSS. Eventually you will be creating Web pages that contain only structural information, with all display information kept separately in CSS files. Chapters 6 through 9 discuss CSS in detail.

The following list contains steps you need to take to migrate from HTML to XHTML:

1. **Evaluate existing code**—Check for basic compliance with XHTML syntax rules. Are closing tags included? Are all tags lowercase? Are attributes quoted? How much cleanup work is necessary to make the code well formed? Most of this work can be automated in the various HTML editing programs.

2. **Evaluate existing display information**—How much of your code includes deprecated elements such as `<font>` and deprecated attributes such as “bgcolor,” “face,” and “size”? On many sites this information can make up as much as 50% of the existing code. Start thinking about how you can express these characteristics in CSS.

3. **Create coding conventions**—Create coding conventions and follow them throughout the site. Make sure that new content added to the site follows the new coding and CSS standards. The more you standardize, the easier your maintenance chores become.

4. **Start using CSS**—Start by building simple style sheets that express basic characteristics such as page colors, font family, and font size. Consider using more advanced CSS options such as classes that allow you to name and standardize
the various styles for your site. As you build style rules, start to remove the existing display information in the site.

5. Test for backward compatibility—Older browsers ignore new XHTML syntax rules, but you still need to test in older browsers to make sure that your content is legible and readable. Test carefully with your CSS style rules to make sure that they are supported in older browsers; if they are not, either adjust with the older browsers in mind or consider serving a different set of style sheets for older browsers, which some Web servers are capable of doing.

Choosing an HTML Editor

You can create or generate HTML code to build Web pages in many ways. Until recently, the most widely accepted tool was Notepad, the simple text editor that comes with Windows versions from 95 to XP. On the Macintosh, the equivalent tool is TeachText or SimpleText. Many sites on the Web are coded using these text-editing tools, which are easy to use and still relied upon by top-notch HTML authors. They also are the best way to learn HTML because you have to enter every tag by hand. However, fewer designers use simple text editors now that increasingly robust HTML-authoring packages have appeared.

There are a variety of HTML editing programs, such as Adobe GoLive, Microsoft FrontPage, and Macromedia Dreamweaver, to name a few. Some code-based HTML editors, such as Macromedia HomeSite, forgo a WYSIWYG approach. They have become popular because they include many powerful enhancements that Notepad lacks, such as multiple search-and-replace features and syntax checking, while still allowing you to manipulate code at the tag level. Most recent authoring tools offer syntax validation and code conversion as well, which can greatly lessen the tasks of cleaning up older code to match the newer XHTML syntax rules.

Many of the latest office applications now convert documents to HTML. For example, you can create a flyer in your word processor and export it to create an HTML page. You can even create slides in Microsoft PowerPoint or Lotus Freelance Graphics and export them to HTML. This hands-off approach leaves much to be desired for an HTML author because you give up control over the finished product. Additionally, the practice of converting content from a program such as Microsoft Word to HTML is notorious for creating substandard HTML code. You are better off moving away from one of the office applications to a dedicated HTML authoring package if you are serious about creating attractive, portable Web sites.

As with browsers, authoring packages interpret tags based on their own built-in logic. Therefore, a page that you create in an editing interface may look quite different in a browser. Furthermore, many editing packages create complex, substandard code to achieve an effect specified by the user. The more complex code can cause compatibility
problems across different browsers. Remember that HTML is a relatively simple language that is not meant to express complex layouts. Many Web page designers, spoiled by the ease of use of today’s powerful word processors, build complex pages with complicated text effects and spacing. When the editing program translates this for display with simple HTML, it resorts to methods that may result in code that is difficult to update or debug. HTML authors who are accustomed to coding by hand (in Notepad or another text editor) often are surprised to see the code an HTML editing package has generated. To code effectively with HTML, you must be comfortable working directly at the code level. Though you may choose to use one of the many editing packages to generate the basic layout or structure for your page or to build a complex table, be prepared to edit the code at the tag level to fix any discrepancies. You probably will end up working with a combination of tools to create your finished pages.

VARIABLES IN THE WEB DESIGN ENVIRONMENT

Always consider four universal variables when you are designing for the Web. These variables are very important because they affect the way your users view and interact with your Web content. This section describes these variables, their effect on your Web pages, and the steps you can take to code and design effectively with them in mind.

- Browser compatibility
- Connection speed
- Screen resolution
- Operating system

Browser Compatibility Issues

One of the greatest challenges facing HTML authors is designing pages that multiple browsers display properly. Every browser contains a program called a parser that interprets the markup tags in an HTML file and displays the results in the canvas area of the browser interface, as illustrated in Figure 1-2. The logic for interpreting the HTML tags varies from browser to browser, resulting in many possibly conflicting interpretations of the way the HTML file is displayed. As a Web page designer, you must test your work in as many different browsers as possible to ensure that the work you create appears as you designed it. Although your work might seem cross-browser compatible, you may be surprised to see that the results of your HTML code look very different when viewed with different browsers.

As different browsers competed for market share, a set of proprietary HTML elements evolved for the use of each particular browser. Some examples of these elements are <font> and <center>, which were developed specifically for the Netscape browser. <font> eventually became part of the HTML 3.2 specification, but it has been designated
a deprecated element in HTML 4.0. **Deprecated elements** are those that the W3C has identified as obsolete and consequently will not be included in future releases of HTML. However, it is likely that such elements will be supported by browsers for some time. The browser developers would be doing users a disservice (and possibly losing market share) if they removed support for these elements.

Confusing the compatibility issue further are the elements that are strictly proprietary, such as `<marquee>` (Internet Explorer only), which creates scrolling text, and `<blink>` (Netscape Navigator only), which makes text blink on and off. These elements work only within the browser for which they were designed and are ignored by other browsers. Because proprietary elements such as these go against the open, portable nature of the Web, they are not included in the standard maintained by the W3C. Avoid using proprietary elements unless you are sure that your audience is using only the browser for which the elements were designed.

The newer browsers such as Internet Explorer 6.0, Netscape 7.0, and Opera 7.0 offer much better support for the standards released by the W3C. The browser software companies have found that the Web development community benefits from the increased support of the standards. More consistent browsers allow better visual design and increased interactivity for all users.
Most HTML authors do not have the luxury of knowing the age, type, or operating system of the browser that will be used to view their Web pages. Browser and version choices can vary widely based on a number of variables. Many individuals and organizations are reluctant to upgrade software simply because a new version has been released. Other users may have older computers that do not have the processing speed or disk space to handle a newer browser. Although it is a good idea to test with the latest browsers, it also is prudent to test your work in older browsers to maximize the number of people who see your Web pages as you intend.

As discussed earlier, not only are new browsers released frequently, but older browsers still are used by many Web users. The newer browsers support desirable features, such as Cascading Style Sheets, that are not supported by older browsers. Including newly supported features in your page design may significantly affect the way your page is viewed if the browser cannot interpret the latest enhancements. Browsers exhibit subtle differences across computing platforms as well.

Creating Cross-browser Compatible Pages

How can you handle the demands of different browsers while designing attractive Web pages? Some HTML authors suggest that you stick strictly to the W3C standards to ensure portability. Others say that you should push the medium forward by coding to the latest standard and using the most recent enhancements. Some Web sites recommend that you use a particular brand and version of browser to access the site. Let’s examine each of these methods to determine the best way to design your site.

If you would like to download a particular browser, or find out which browser is currently the most popular, visit one of these Web sites:
BrowserNews at www.upsdell.com/BrowserNews/
CNET Browser Info at www.browsers.com.

Standardized Coding

Although it can be difficult to create pages that are always displayed properly, it is not impossible. The best way to create portable Web sites is to strictly follow the standards set by the World Wide Web Consortium. This approach provides the greatest acceptance and uniform display of your content across multiple browsers and operating systems. This “best practices” method of coding is widely supported among sites that are interested in the greatest accessibility. Following the W3C standards does not mean that your site has to be visually uninteresting, although you may have to sacrifice the latest multimedia enhancements. Reliable visual and information design techniques, along with the use of Cascading Style Sheets, can let you overcome many functional limitations.
Cutting-edge Coding

Another strategy to adopt when designing your Web site is to stay at the cutting edge. By requiring the latest software, some designers insist that their users keep up with them. This design strategy can result in visually exciting and interactive sites that keep pace with the latest technology. Often the user must have not only the latest browser version but also plug-in enhancements that render certain media types such as Macromedia Flash animations. **Plug-ins** are helper applications that assist a browser in rendering a special effect. Without the plug-in, your user cannot see the results of your work. Often when a new browser is released, these plug-ins are included for the most widely adopted enhancements. The risk of the cutting-edge approach is that many users may not be able to see the content as it was designed. Sites that use the latest enhancements also may require significant download times for the special effects to load on the user's computer. If sites that adopt the latest technologies do not make sure that their users keep up with the latest connection technologies, browser versions, and plug-ins, their information may go unread.

Browser-specific Coding

Some Web sites are coded for one particular browser or brand of browsers only. The author may have wanted to use a unique enhancement for the site or may have found that the site did not render properly in other browsers. Although this may seem the most expedient coding method, consider the consequences. A site coded for only one browser may alienate a significant number of readers who immediately leave because they do not have the correct browser. On the Web, you never can be sure of the type of browser your user has. However, this method of browser-specific coding may be viable on a company **intranet**, where you know or you can specify that all users have the same brand and version of browser. For the general Web, it is the least desirable choice, because you are limiting the availability of your site.

Solving the Browser Dilemma

You must test your work in as many browsers as possible during the entire development process to make sure that your pages are displayed properly. Knowing your audience is a major step toward correctly implementing your site. For example, you may be building a site that discusses the latest in technology trends. This site will attract computer-savvy users, so you can code for the latest browsers. On the other hand, if you are creating a site that will attract the general public, you should code for the lowest common denominator and make sure your pages appear as designed in every browser. Many general Web users access the Web via America Online, Inc. (AOL), so test your work using their browser as well. If you want to include animations or effects that require a plug-in, use a development tool that already is supported by the major browsers. Make sure that the most important content on your site is rendered in a way that does not rely on the new technology so that users with older browsers still get your message. Finally, if you are designing for an intranet and can mandate the type of software your viewers use, you can work with only one browser in mind.
Considering Connection Speed Differences

Connection speed is another variable that should influence your Web page design. Most users simply will not wait longer than 10–20 seconds for a page to load. If your pages download slowly, your users probably will click to go to another site before they see even a portion of your content. Many designers make the mistake of not testing their pages at different connection speeds. If you do not test, you cannot appreciate what it is like for users to connect at different speeds to your site, and you may lose valuable visitors.

It will still be awhile before the majority of computer users gain high-speed access to the Web. According to an article in the *Boston Globe* (April 19, 2004), “Two in five Net users now have home broadband access. The number of broadband subscribers has increased as telephone companies slash prices to better compete with broadband services. DSL now makes up 42% of the home broadband market, up from 28% in March 2003.” Access via cable modem had been the most reliable high-speed connection to the Web for home users, but Digital Subscriber Line (DSL), the service offered by telephone companies, has gained in popularity. Corporations still rely primarily on T1 or Integrated Services Digital Network (ISDN) connections. Table 1–1 describes the more common types of connection technologies.

**Table 1–1** Common types of connection technologies

<table>
<thead>
<tr>
<th>Technology</th>
<th>Speed</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular telephone line</td>
<td>Up to 56 Kbps</td>
<td>This is still the most common method of connecting to the Internet; however, you’re lucky if you can consistently maintain a connection speed over 44 Kbps.</td>
</tr>
<tr>
<td>ISDN basic</td>
<td>64 Kbps to 128 Kbps</td>
<td>ISDN offers good speed and allows a constant connection to the Internet, but is fairly expensive; ISDN is more common in urban areas and is primarily used by businesses.</td>
</tr>
<tr>
<td>Digital Subscriber Line</td>
<td>512 Kbps to 8 Mbps</td>
<td>DSL uses a single existing phone line to carry both voice and data, and allows a constant connection to the Internet.</td>
</tr>
<tr>
<td>Cable modem</td>
<td>512 Kbps to 52 Mbps</td>
<td>Cable modems are fast, allow a constant connection to the Internet, and you don’t have to dial up to connect, but not all cable systems offer Internet capabilities.</td>
</tr>
</tbody>
</table>

Because the single biggest factor influencing the speed at which your pages are displayed is the size and number of graphics on your Web pages, you should keep your page designs simple with few graphics. As a rule of thumb, no single image on your Web site should exceed 10 to 15 KB. If you know all your users have faster access, you can design your pages to match. For the general public you can consider 56 Kbps as a base connection speed because many users still use modems. You will learn more about how to prepare your images to download quickly in Chapter 8.
Working with the Cache to Improve Download Time

All Web pages are stored on computers called Web servers. When you type a Uniform Resource Locator (URL) address in your browser, it connects to the appropriate Web server and requests the file you specified. The server serves up the file so your browser can download it. The first time you visit a site, the entire contents of the HTML file (which is plain text) and every image referenced in the HTML code is downloaded to your hard drive. The next time you visit this site, your browser downloads and parses the HTML file. The browser checks to see if it has any of the specified images stored locally on the computer’s hard drive in the cache. The cache is the browser’s temporary storage area for Web pages and images. The browser always tries to load images from the cache rather than downloading them again from the Web.

You can make use of the browser’s caching capabilities by reusing graphics as much as possible throughout your site. Once an image is downloaded, it remains in your user’s cache for the number of days specified in the user’s preference settings. Most users do not change the settings, so there is a good chance your graphics will remain on the user’s hard drive a while. Every time the user revisits your site, the cached graphics load locally rather than from the Web server. The browser’s caching capability is a great argument for standardizing the look of your site by using the same navigation, branding, and background graphics throughout. Not only does the consistency increase the usability of your site, but also your pages load faster.

Coding for Multiple Screen Resolutions

No matter how carefully you design pages, you can never know how users view your work because you do not know their monitors’ screen resolution. A computer monitor’s screen resolution is the width and height of the computer screen in pixels. Most monitors can be set to at least two resolutions, whereas larger monitors have a broader range from which to choose. User screen resolution is a factor over which you have no control.

Screen resolution is a function of the monitor’s capabilities and the computer’s video card. The two most common screen resolutions (traditionally expressed as width × height in pixels) are 800 × 600 and 1024 × 768. Some users choose the highest resolution of 1024 × 768, allowing them to display more on the screen. They may have multiple application windows open at the same time. Users at 800 × 600 usually maximize their browser to full screen. As larger monitors become less expensive, even higher screen resolutions are now available, but these two sizes are still the most commonly used. Additionally, users with much older computers may have their resolution set to 640 × 480, but this resolution has become so obsolete that it is no longer a viable consideration for Web developers.

Fixed Design

Figures 1-3 and 1-4 show the same L.L. Bean Web site viewed at different screen resolutions.
Notice in Figure 1-3 that the page is designed to display its content within an 800 × 600 screen resolution. The content entirely fills the browser window, indicating that 800 × 600 is the base screen resolution of this Web site. Users viewing the page at 1024 × 768, as shown in Figure 1-4, see the content aligned to the left side of the page, and the passive white space on the right side of the page fills in the remainder of the screen. You will read more about the use of active and passive white space in Chapter 2.
Flexible Design

In contrast, Figures 1-5 and 1-6 show a Web page that has been designed to adapt to different screen resolutions.

The Web page shown in Figures 1-5 and 1-6 was designed for an 800 × 600 resolution but is adaptable to other resolutions as well. As the screen resolution changes, the middle column expands or contracts to accommodate the varying screen width, while the outside columns...
remain fixed. The designers accomplished this adaptability through variable rather than absolute table widths. You will learn about this technique in Chapter 5. The challenge in using this type of design is that the content must remain organized and legible at all screen resolutions, requiring more testing by the Web developer to ensure success.

Figure 1-5  Flexible design at 800 x 600 resolution
Centered Design

A third choice for handling different screen resolutions is shown in Figures 1-7 and 1-8. These figures show a Web page that has been designed to adapt to different screen resolutions by remaining centered in the user’s browser.

The Web page shown in Figure 1-7 fills the browser window at 800 × 600 resolution. As the screen resolution changes, the Web page stays centered in the browser window, splitting the remaining space into equal amounts on the left and right side of the browser window. Figure 1-8 shows the same page at 1024 × 768. This technique “frames” the page with the leftover space, resulting in more active white space than the left-justified technique shown in Figure 1-4. The benefit of centering a page is that the layout of the content remains unchanged no matter what the user’s screen resolution.

As a Web designer, you decide how to code your Web site to handle different screen resolutions. Most Web sites were once coded to the lowest possible screen resolution, which is 640 × 480. Now that monitors are getting bigger and less expensive, the majority of Web users probably have their screen resolution set to 800 × 600, which is the current standard resolution for most Web sites. If you know your audience is consistently using a higher resolution, you can code to it. Otherwise, code to 800 × 600 resolution to make sure that your content fits most screens. Remember to test at different resolutions to ensure that your user can view your pages properly.
Figure 1-7  Centered design at 800 x 600 resolution

Figure 1-8  Centered design at 1024 x 768 resolution
OPERATING SYSTEM ISSUES

The user’s operating system is the variable over which you have the least control. People use endless combinations of monitors, computers, and operating systems on desktops around the world. In today’s computing environment, three operating systems dominate: the Windows PC platform, the Apple Macintosh platform, and various flavors of UNIX. The best method for dealing with this variety is to test your content on as many operating systems as possible, although this is not realistic for the student or beginning Web designer. Remember the following points about different operating systems:

- **Monitors and display software**—For many technical and physical reasons, the colors you choose and images you prepare for your site can look vastly different on different machines. Screen resolutions and sizes, color depth, and video hardware and software all affect the look of your Web pages. Follow the guidelines on browser-safe colors in Chapter 8 to make your colors as cross-platform compatible as possible.

- **Browser versions**—Not all browsers are the same on all operating systems. Often software companies release different versions of their browsers based on the popularity of the operating system. For this reason, Internet Explorer is often a release behind on the Macintosh platform. Microsoft does not even make a UNIX version of their browser, although Netscape does. The only solution to this problem is to test your work in as many browsers as possible.

- **Font choices**—Installed fonts vary widely from one computer to another. Choose fonts that are commonly used; otherwise, the font you choose, if not installed on the user’s machine, will appear in a default typeface. Read Chapter 7 for more information on this subject.

CHAPTER SUMMARY

Many variables affect the way users view your Web pages. As an HTML author, your goal should be to code pages that are accessible to the largest audience possible. As you plan your Web site, make the following decisions before implementing your site.

- Use Cascading Style Sheets. The style enhancements and control offered by this style language are formidable but are not evenly supported by older browsers. Implement CSS gradually, testing for browser compatibility as you go.

- Decide whether to code to the XHTML standard. If you are starting a new Web site, your best choice is to code to this new standard. If you are working with an existing Web site, decide on the most expedient method for upgrading your existing code to XHTML standards to ensure future compatibility with new tools and browsers.

- Choose the type of editing tool you will use to create your HTML code. You may want to use a WYSIWYG editor to create the general page layout and then rely on Notepad to make corrections to your code. Alternately, a code-based editor such as Macromedia HomeSite lets you work directly with code while enjoying enhancements that Notepad doesn’t support.
Choose the suite of browsers you will use to test your site. Although you will include the latest versions of Netscape and Internet Explorer, consider testing in older versions of each browser as well.

Decide how browser specific your site will be. Your goal is to create a site that is widely accessible to multiple browsers. If you have a narrow audience or specific requirements, you may want to specify one browser as the primary method for viewing your site.

Resolve to test your work continually as you build your site. Test with multiple browsers at different screen resolutions and at different connection speeds. If you can, view your site on multiple platforms such as Windows, Macintosh, and UNIX as well.

**Review Questions**

1. HTML is a subset of which markup language?
2. List three characteristics of HTML that make it ideal for the World Wide Web.
3. What are the benefits of viewing source code on the Web?
5. What is a deprecated element?
6. What is a proprietary element?
7. What style language allows the separation of style from structure in HTML?
8. What are the advantages of using an external style sheet?
9. What feature distinguishes XML from HTML?
10. What are the two types of style languages designed for use with XML?
11. Explain how XML lends itself to customized data applications.
12. What improvements does XHTML promise over existing HTML?
13. Explain how different browsers affect the display of a Web page.
14. Describe the characteristics of coding for the lowest common denominator.
15. Describe how coding using the latest technology can prevent users from accessing your site.
16. List the two most common screen resolutions.
17. Explain how screen resolution affects the display of a Web page.
18. List four common types of Internet connection technologies.
19. Explain how the browser’s caching capability improves download time.
20. Explain the three issues you should consider when designing for multiple operating systems.
HANDS-ON PROJECTS


2. In this project, you edit an existing HTML file to conform to XHTML coding standards.
   a. Copy the project1-2.htm file from the Chapter01 folder provided with your Data Files to the Chapter01 folder in your work folder. (Create a Chapter01 folder, if necessary.)
   b. Using a text editor such as Notepad, open project1-2.htm. The code looks like the following. (The location of your line breaks might differ.)
   
   ```html
   <HTML>
   <HEAD>
   <TITLE>Chapter 1 - Getting Started with Wonder Software</TITLE>
   </HEAD>
   <BODY>
   <img src="logo.jpg" width=100 height=100 alt="logo image">
   <H2>Getting Started with Wonder Software</H2>
   <P>In this section you’ll learn how to add a new user to the Wonder database.
   <h3>To add a new user:</h3>
   <OL>
   <LI>Open the <B>Admin menu</B> and select <B>Users</B>.
   <LI>Choose the <B>Add User...</B> button.
   <LI>Enter the necessary user information.
   </OL>
   <P>
   Note: Make sure to enter a value in all of the user information fields. If you skip a field, the Wonder database will reject the record.
   <TABLE WIDTH=100%>
   <TR >
   <TD WIDTH=50%><A HREF="toc.htm">Table Of Contents</A></TD><TD ALIGN=RIGHT><A HREF="index.htm">Index</A></TD></TR>
   </TABLE>
   ```
   c. The file contains a number of coding errors that make it noncompliant with XHTML standards. Edit the file to correct the coding errors. Add comments to the file that explain each change you make. Refer to the “XHTML Syntax Rules” section of this chapter for help.
   d. Save the file with the same name.

4. Download and install the latest versions of the following browsers onto your computer as necessary:
   - Internet Explorer: www.microsoft.com/windows/ie/
   - Netscape: channels.netscape.com/ns/browsers/
   - Opera: www.opera.com/download/

Test the browsers. Write a short paper explaining your likes and dislikes about each browser. Use examples whenever possible to support your opinion. Consider the following:
   a. Give each browser a test drive by using it for a few hours of Web browsing.
   b. Choose a mainstream Web site and test it with all three browsers. What differences can you find in the way the page is displayed?
   c. Which browser offers the best user experience? Why?
   d. Is one browser faster than the others?

5. Describe three common mistakes that Web designers make when building a Web site.

6. Test the HTML conversion capabilities of a standard office application.
   a. Use your favorite word processing, spreadsheet, or presentation graphics program that supports conversion to HTML.
   b. Create a document and export it to HTML.
   c. Examine and evaluate the HTML code. Look for nonstandard coding techniques or tricks that the program uses to render content into HTML. Write a detailed description of your findings.

7. Test cross-browser compatibility.
   a. Make sure you have recent versions of both Netscape Navigator and Internet Explorer installed on your computer. (See Hands-on Project 1-4 if you need to download and install a browser.)
   b. Browse a variety of Web sites. Make sure to view various pages of the sites in both browsers.
   c. Write a detailed description of how successfully the various sites appear in both browsers. Look for text, layout, and graphic inconsistencies.

8. View source code in a browser.

In the following steps, you view the source code from a live Web page in your browser. Choose the instructions for Internet Explorer 6.0, Netscape 7.0, or Opera 7.0.

To view source code in Internet Explorer 6.0 or Opera 7.0:
   □ Click View on the menu bar, and then click Source. Your system’s text editor, such as Notepad, SimpleText, or WordPad, opens and displays the page’s source code. You then can save the file to your own hard drive and manipulate the code.
To view source code in Netscape Navigator 7.0:

- Click View on the menu bar, then click Page Source. The page’s source code opens in another window of Netscape. You can copy and paste text from this window into your own text or HTML editor, then manipulate and test the code. You cannot edit directly in the Page Source window.

When you copy code from a Web site, remember to respect the author’s copyrights on any original material. Although page layouts cannot be copyrighted, any original text or graphics are the property of the author and should be properly cited.

**Case Project**

To complete the ongoing Case Project for this book, you must create a complete stand-alone Web site. The site must contain between six and ten pages, displaying at least three levels of information. You can choose your own content. For example, you can do a work-related topic, a personal interest site, or a site for your favorite nonprofit organization. The site will be evaluated for cohesiveness, accessibility, and design. At the end of each chapter, you will complete a different section of the project. For Chapter 1, get started by creating a project proposal, as in the following outline. As you progress through the chapters of the book, you will complete different facets of the Web site construction, resulting in a complete Web site.

**Project Proposal**

Create a one- or two-page HTML document stating the basic elements you will include in your Web site. Create this document using your favorite HTML editor or Notepad. At this stage your proposal is primarily a draft. At the end of the next chapter you will have a chance to modify the proposal and supplement the design details.

Include the following items, if applicable:

- **Site title**—Specify the working title for the site.
- **Developer**—Identify yourself and anyone else who will work on the site.
- **Rationale or focus**—Explain the content and goals of the site, such as billboard, customer support, catalog/e-commerce, informational, or resource. Refer to Chapter 3 for help on content types.
- **Main elements outline**—Describe the main features of the site.
- **Content**—Estimate the number of individual Web pages.
- **Target audience**—Describe the typical audience for the site.
- **Design considerations**—List the design goals for the site.
- **Limiting factors**—Identify the technical or audience factors that could limit the design goals of the site.