CHAPTER 1

HTML and the Modern Web

In this chapter you will learn about:

- Creating Web pages with HTML
- The history of HTML
- Working with HTML5
- Choosing an HTML editor
- Using good coding practices

In this chapter, you explore how HTML is used along with CSS to create Web pages, and learn the history of how HTML has evolved to its current state. As you will see, designing Web pages has changed dramatically in the last few years. You will examine the latest release of HTML, called HTML5, and see how it can adapt to a variety of Web design needs. You will learn about the new elements and capabilities of HTML5, how to choose the best syntax for the Web pages you are going to create, and how to create correct code. Finally, you consider what type of software tool you can use to create your HTML code, and how to use good coding practices to make sure your work is useful now and in the future.

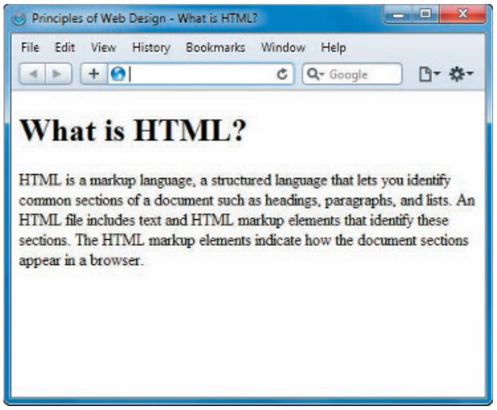
Creating Web Pages with HTML

In today's Web, people shop, trade stocks, watch videos, share photos, play games, communicate via social networks, interact using live chat and video, and much more. They perform all of these activities using **Web pages,** text documents that Web browsers interpret and display. Despite the diversity of content, all of the Web pages on the World Wide Web have one thing in common. They all must be created using some form of the **Hypertext Markup Language (HTML)**. It is astonishing to think that the entire World Wide Web, used every day by millions for shopping, research, banking, and a myriad of other applications is based on a simple text-based markup language that is easy to learn and use.

HTML is a **markup language**, a structured language that lets you identify common sections of a Web page such as headings, paragraphs, and lists with markup tags that define each section. For example, the <h1> element in the following code indicates that the text is a first-level heading:

```
<h1>What is HTML?</h1>
```

Web pages are simply text documents that use HTML to tell the browser how to display each document section. Figure 1-1 shows a basic Web page and the code that is used to create it.


Principles of Web Design - What is HTML?
File Edit View History Bookmarks Window Help
Google

What is HTML?

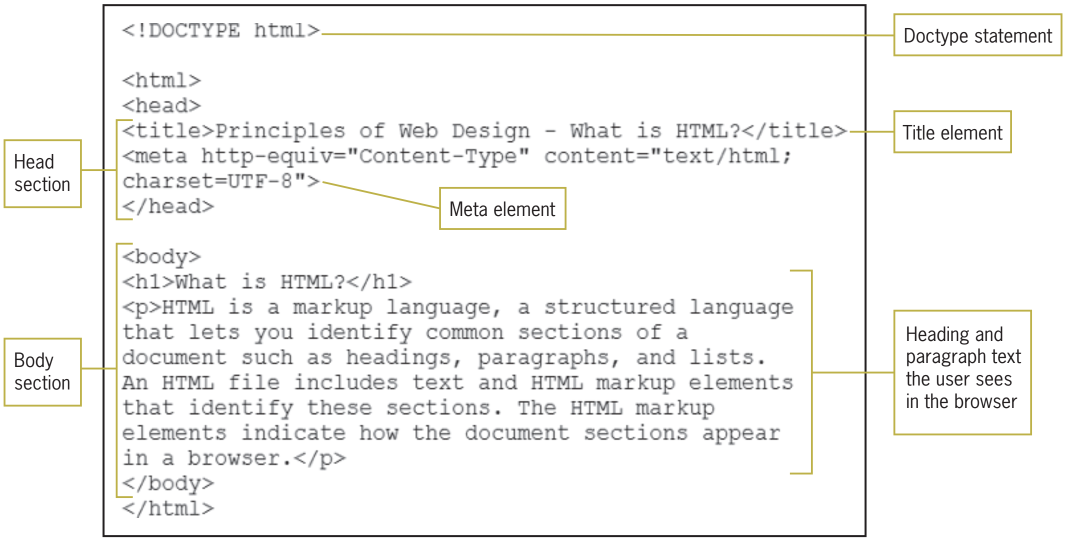
HTML is a markup language, a structured language that lets you identify common sections of a document such as headings, paragraphs, and lists. An HTML file includes text and HTML markup elements that identify these sections. The HTML markup elements indicate how the document sections appear in a browser.

```
<!DOCTYPE html>

<html>
<head>
<title>Principles of Web Design - What is HTML?</title>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
</head>

<body>
<h1>What is HTML?</h1>
<p>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 elements
that identify these sections. The HTML markup
elements indicate how the document sections appear
in a browser.</p>
</body>
</html>
```

Figure 1-1 Basic Web page and its HTML code

Structure of a Basic Web Page

An HTML file includes the text that the user sees in the browser, contained within HTML markup elements the user cannot see that identify document sections and elements as shown in Figure 1-2.



```
<!DOCTYPE html>

<html>
<head>
<title>Principles of Web Design - What is HTML?</title>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
</head>

<body>
<h1>What is HTML?</h1>
<p>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 elements
that identify these sections. The HTML markup
elements indicate how the document sections appear
in a browser.</p>
</body>
</html>
```

Figure 1-2 Structural elements in an HTML file

The HTML code in this example demonstrates the basic structure of an HTML document. First, the **Document Type**, or doctype for short, specifies the rules for the document language so the browser knows how to interpret the HTML code and display it properly.

After the doctype statement, the opening <html> tag and its closing </html> tag at the end of the page are the root element of the document. A **root element** is the container element for all other elements in the document.

After the root element is the <head> tag. The two main sections of an HTML document are the head and body sections, represented by the <head> and <body> elements. The head section is the container for all of the descriptive information about the document, including the document title, coding standards, links to external style sheets, and scripting code for interaction. None of the content in the head section appears in the browser window.

The head section contains the important <title> element. This element contains the title of the document, which shows in the title bar of the browser. Document titles should clearly describe the page, contain key terms, and be understandable out of their Web site context. For example, "Home Page" as a title is meaningless outside of the context of its related content. The contents of <title> is a primary source of information for search engines and is often the first text users see in a list of search results. The head section also contains the <meta> element, which defines the content type as type "text/html" and declares the character set for the document.

The body section includes the content that the user sees in the browser window. The body of the document can contain text, images, or audio content, forms for gathering information, interactive content, and hypertext links to other Web resources. Here is where all of the various structural elements that make up HTML come into play. For example, document headings are marked with one of a variety of heading tags, such as <h1> or <h2>, signifying a top-level or secondary-level heading. Paragraph content is marked with the <p> element. HTML offers many elements to expressly mark each section of a document. Appendix A contains a complete list of the HTML elements and their usage.

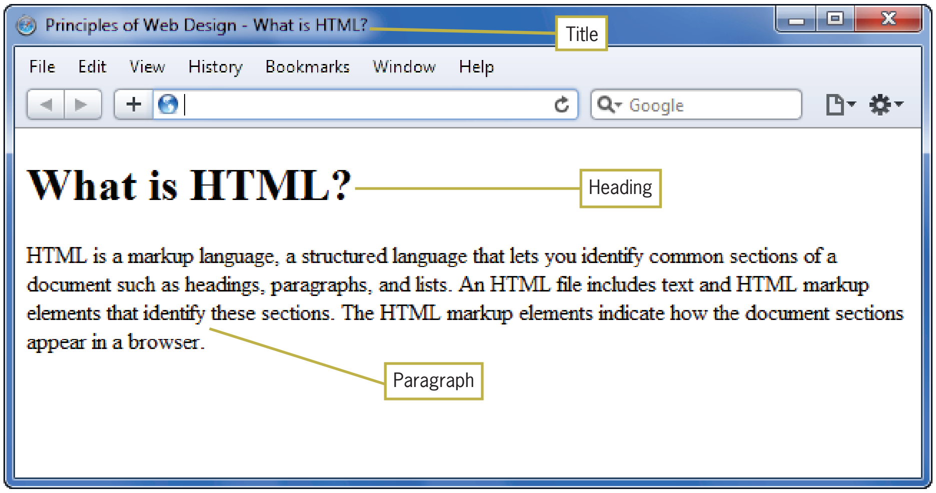
Once you are familiar with the HTML syntax, you will find that one of the best ways to learn new coding techniques is to find a Web page you like and view the source code.

All of the major browsers support a similar action to view the source code. Right-click a blank spot in the browser window and choose a command such as View Page Source or View Source.

You should use the HTML elements properly to describe each section of the document based on the logical structure of the document. For example, mark headings as headings, long quotes as <blockquote>, paragraphs as <p>, and so on. In the earlier days of the Web, HTML elements were often misused depending on how they looked in the browser rather than for their structural meaning. Avoid using this type of markup, and express all display information with Cascading Style Sheets, which you will read more about later in this section.

HTML in the Browser

The browser interprets the HTML markup elements and displays the results, hiding the actual markup from the user. The user sees only the text "What is HTML?" formatted as a level-one heading and the paragraph text formatted as a paragraph. Figure 1-3 shows the browser's rendition of the HTML code shown in Figure 1-2. Each HTML element contains basic display information to organize and present contents in the browser, such as heading elements that are displayed in a bolder and larger font than paragraph elements. Notice also that the title is displayed in the title bar of the browser.


Principles of Web Design - What is HTML?
Title
File Edit View History Bookmarks Window Help
Google
Heading
Paragraph

What is 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 elements that identify these sections. The HTML markup elements indicate how the document sections appear in a browser.

Figure 1-3 Browser interprets the HTML markup and displays the results

As you begin to design and build Web pages, remember that each browser interprets HTML in its own way, based on its **rendering engine**. A browser's rendering engine is software that reads the document's HTML code and associated CSS style information and displays the resulting formatted content in the browser window. Each major browser has its own rendering engine. Although most of your Web pages should look similar in most browsers, it is essential that you test your work in different Web browsers to make sure that your Web pages are rendered as consistently as possible. You will learn more about browser differences in Chapter 2.

Adding Style with CSS

To add presentation information to Web pages, Web designers use a style language called **Cascading Style Sheets (CSS)**. With CSS you can display information for different devices, such as a cell phone or computer screen, lay out page designs, and control typography, color, and many other presentation characteristics.

Style elements such as <font> were introduced by browser developers to help Web designers bypass the design limitations of HTML. Using elements such as <font> to embed style information within the structure, as is the case in most early Web development, limits the cross-platform compatibility of the content. The display information that is embedded in Web pages is tailored towards one type of display medium, the computer screen. A **style sheet** is a set of style rules that describes the display characteristics of a document. With style sheets, the presentation properties are separate from the content. This accommodates the variety of devices and users that browse the Web, as shown in Figure 1-4.

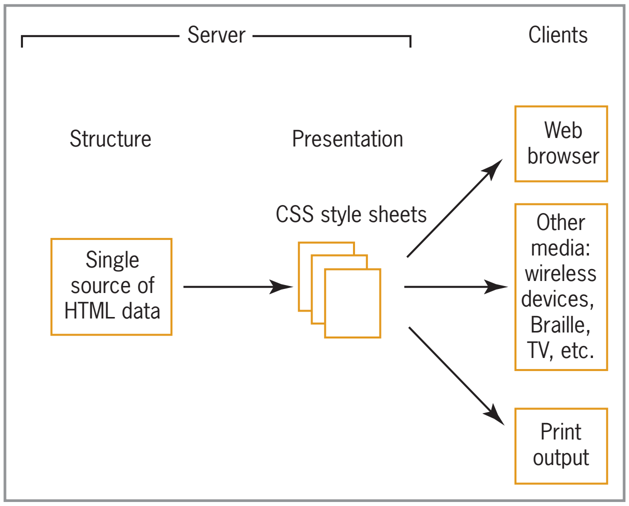

Server
Clients
Structure
Presentation
CSS style sheets
Single source of HTML data
Web browser
Other media: wireless devices, Braille, TV, etc.
Print output

Figure 1-4 Formatting data for multiple destinations

CSS lets you control the presentation characteristics of an entire Web site with a single style sheet document. For example, assume that you want all of your <h1> headings to appear green and centered everywhere on your Web site. Prior to CSS you might have included the following code for every instance of an <h1> element:

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

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

```
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; the single rule controls every <h1> element in your Web site. 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.

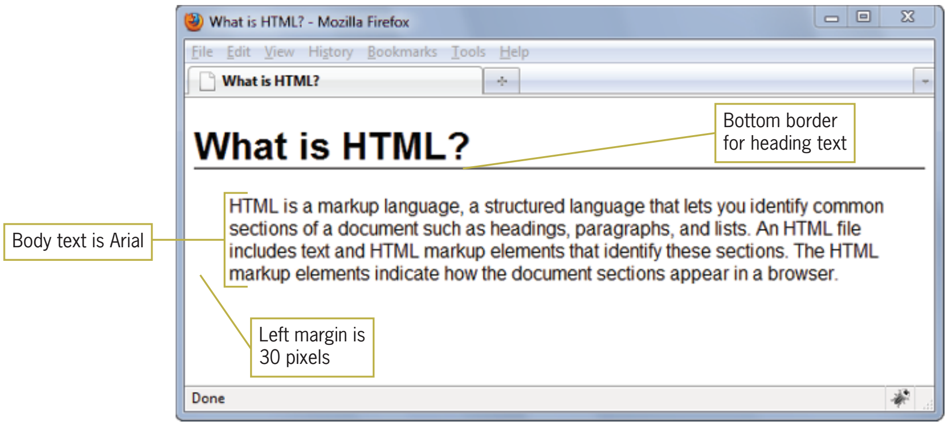
Through much of Web history, the adoption of CSS as a standard for style has been limited because of poor and uneven support by the major browsers. Modern browsers such as Internet Explorer, Firefox, Opera, Safari, and others offer more complete and consistent support for CSS, freeing designers to work with this powerful style language. Modern Web design requires CSS. You will learn about CSS in later chapters of this book.

Let's revisit the Web page shown in Figures 1-2 and 1-3. Adding some simple CSS code adds style to the page. Notice the style section in Figure 1-5. The style rules specify that the body text for the page will be Arial, heading 1 will have a bottom border, and the paragraph will have a 30-pixel left margin. Figure 1-6 shows the results of the addition of style rules.

```
<!DOCTYPE html>

<html>
<head>
<title>Principles of Web Design - What is HTML?</title>

<style type="text/css">
body {font-family: arial;}
h1 {border-bottom: solid 1px;}
p {margin-left: 30px;}
</style>

</head>

<body>
<h1>What is HTML?</h1>
<p>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 elements that
identify these sections. The HTML markup elements
indicate how the document sections appear in a
browser./p>
</body>
</html>
```

CSS style section

Figure 1-5 CSS style section contains presentation information


What is HTML? - Mozilla Firefox
File Edit View History Bookmarks Tools Help
What is HTML?
Bottom border for heading text
Body text is Arial
Left margin is 30 pixels
Done

What is 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 elements that identify these sections. The HTML markup elements indicate how the document sections appear in a browser.

Figure 1-6 Result of adding the CSS style rules

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 topic. As a Web designer, 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 continually evolve as the Web continues to grow into an increasingly richer and more interactive environment.

The History of HTML

Now that you've seen the basics of how HTML works, it is important to know how the Web has evolved to its current state. As a Web designer, you will encounter all types of HTML coding practices in the real world. Understanding the evolution of HTML will help you understand various Web design methods. As a Web designer, you may encounter sites that completely comply with current standards, ones that still use design trends and code from years ago, and others that mix coding styles in improbable ways. Many sites still use table-based designs, which are outmoded and now rendered obsolete by CSS. Web design tools can create all different kinds of code. Proprietary software implementations manipulate code to get the exact result they want. Although it is easy to say that everyone should use Web standards, to be a successful Web designer you need to understand the past, present, and future directions of HTML, coding standards, and common practices, many of which may not match the standards you learn about in this book.

When Tim Berners-Lee first proposed HTML at the European Laboratory for Particle Physics (CERN) in 1989, he was looking for a way to easily manage and share information among scientific colleagues over the Internet. Until this time, the complexity of using the Internet for exchanging messages and sharing files limited its use to groups of specialists in defense, academia, and science.

Berners-Lee joined the ideas of a simple tool for reading the documents (the browser), rules for creating a document markup language (HTML), and a communications protocol that allowed hypertext linking through Uniform Resource Locators (URLs). This accessible, simple interface made using the Internet available to the public. Not only could people read documents, they could easily create them using the easy-to-understand HTML.

As Berners-Lee developed the idea of a Web of documents connected by hypertext links and hosted by computers called hypertext servers, he created a simplified application of the **Standard Generalized Markup Language (SGML)**, a standard system for specifying document structure, which he called the Hypertext Markup Language. HTML significantly 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 version of HTML included roughly 20 elements that represented basic document structure such as titles, headings, paragraphs, and lists. HTML was intended for simple document structure, not for handling today's varied and complex information needs.

A Need for Standards

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. To meet this need, the **World Wide Web Consortium (W3C)** was founded in 1994 at the Massachusetts Institute of Technology. The W3C sets standards for HTML and provides an open, nonproprietary forum for industry and academic representatives. The various committees that make up the W3C look to expand and set standards for the many new Web technologies that have emerged.

After the W3C was founded, the popularity of the Web grew exponentially. By the mid-1990s, companies started to realize that having a Web presence was vital. Publishing companies started reproducing their paper-based content on the Web. Every advertisement contained a Web address. More and more companies hired print designers to develop Web sites. At this time, Web design was a haphazard affair. HTML was a designer's nightmare, primarily because it was intended for basic page structure, not to create multicolumn print-type layouts. Most computer monitors used a 640 x 480 resolution and many only supported 256 colors, which limited design choices. Multiple browsers, each with their own proprietary elements, competed for market share, and many Web sites were coded primarily for one browser or another. Web designers would manipulate the language in any way they saw fit to achieve a desired design result. After HTML tables were introduced, they became the designer's tool of choice, because they allow the creation of multicolumn layouts. Although designed for data, designers manipulated tables as they needed to achieve the design they wanted.

As the Web grew, designers learned they could get away with manipulating HTML because Web browsers are very forgiving of nonstandard coding. Even if you coded your page incorrectly, you had a good chance that your results would look fine in the browser. If you left out closing tags, or used tags in the wrong order, the page would still be displayed. Web browsers did not output error messages if a page contained a coding error. If your Web site worked with coding errors, you would have no reason to fix them, resulting in an Internet full of Web pages with coding errors.

As development on the Web continued to evolve into more data-based applications, such as shopping and banking, interoperability became an issue. If every organization codes and manages their Web site content in their own way, exchanging data between organizations becomes difficult. The more everyone followed the same standards, the easier it would be to write browser, application, and database software that "talked" to each other. Jointly developed standards, rather than ones dictated by one vendor, would benefit everyone.

The 1999 release of HTML 4.01 attempted to address a number of these issues. HTML 4.01 supported CSS. This easy-to-use style language would remove style elements and attributes from the HTML code and replaced them with style rules. This separation of style information from the structure of the HTML document is crucial to the interoperability of HTML, as display information can be customized for the device viewing the Web page, such as a cell phone or computer monitor. HTML 4.01 also deprecated a number of display elements, such as the <font> element. A **deprecated element** means that the element would be removed from future releases of HTML. The W3C was recommending that Web designers stop using these elements in favor of CSS. Many deprecated elements persist to this day, and it is not uncommon to find them in many Web pages and in software programs that create Web pages.

Table 1-1 shows the history of HTML through its releases.

Version	Release Date	Highlights
HTML1.1	1992	First informal draft
HTML 2.0	1995	First release supported by graphical browsers; documents written in HTML 2.0 can still be viewed in all browsers
HTML 3.2	1997	Introduced forms and tables
HTML 4.01	1999	Added support for style sheets, and increased support for scripting and interactivity
HTML5	Future final release, in use today	Latest version adds page layout elements, audio/visual elements, enhanced animation and graphic support

Table 1-1 History of HTML

XML and XHTML: A New Direction

After the release of HTML 4.01, the W3C turned in a different direction for markup languages. In 1997, the W3C released XML, the **Extensible Markup Language**. Although not a direct replacement for HTML, XML has capabilities that are essential to software developers creating applications for the Web. Where HTML is a predefined set of elements that the browser understands, XML lets developers define their own markup language. Software developers could then create elements that matched the data names in their databases. For example, a developer could create a set of elements that described customer information, such as <name> and <city> that would match the names in a database, easing the transition of data to the Web. Additionally, XML is a stricter language than HTML. XML documents must be syntactically correct to be processed by a software application. This means that only one error in an XML document will keep the document from being processed. This is a very different model from HTML, where the syntax is less strict.

XML code looks very similar to HTML code, with some syntactical differences that you will read about in later in this chapter. 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:

```
<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 CSS style rules are added to specify, for example, that the contents of each <line> element should be displayed in the browser on a separate line or in the color blue.

The W3C saw that XML syntax could provide a solution to the problem of widely varying HTML coding standards, and they started to move in this direction with the evolution of XML-based languages, trying to create a unified syntax under which the entire Web could operate. Towards this end, the W3C reformulated HTML in XML, keeping all the same elements and attributes as HTML 4.01, and named it the **Extensible Hypertext Markup Language**, or XHTML 1.0.

XHTML follows the rules of XML, so it follows the markup rules for XML. In short, XHTML requires that documents follow some basic rules, stated briefly:

- Documents must be well formed.
- All tags must nest properly and not overlap.

- Use all lowercase for element names.
- Always use closing tags.
- Empty elements are marked with a closing slash.
- Attribute values must be contained in quotation marks.

Problems with XHTML

Web designers adopted the new language and syntax, hoping to standardize coding conventions. They adopted CSS to gain benefits in Web site maintenance, interoperability, and adapting content to multiple destination media. The beneficial result is that a majority of commercial Web sites have moved to much leaner, standardized code with all presentation and layout information described by CSS. At the same time, many Web sites and software applications still use legacy style coding conventions. In many instances, relaxed rules had to be applied to projects with legacy content. Also, many Web sites are still created by novices who want to put up a site quickly but who don't want to understand the intricacies of XHTML.

As the W3C continued down the XML-based language path, dissatisfaction increased in the Web developer community. When the W3C issued their first drafts of the XHTML 2.0 recommendation, they announced that XHTML 2.0 would not be backwards compatible with XHTML 1.0 or HTML 4.0. This meant that all of the content on the Web, and all of the work that had been done to develop HTML to its current state would no longer be valid. Further, it dropped familiar elements such as <img> for images and the <a> element for hypertext links, and supported the unforgiving error handling of XML.

A Proposal for HTML5

These decisions moved sharply away from the existing direction of Web development and the long-standing ease of use of HTML. In 2004, an independent group of browser vendors and representatives of the Web development community reacted to the strictness of XHTML2.0 and joined to create a proposal for HTML5. This independent group named themselves the Web Hypertext Application Technology Working Group (WHATWG). After a few years of wrangling over the direction of Web languages, the W3C announced in 2007 that they would restart the effort to support HTML, and in 2009 they shut down the XHTML 2.0 working group completely.

The W3C's response to the development community's desires for the next generation of HTML highlights the unusual nature of the Web, where standards are not dictated by one vendor, but rather decided upon and influenced by the people who work with HTML and Web design day to day. As you will see later in this chapter, HTML5 supports standards-based coding, is compatible with both XHTML 1.0 and HTML 4.01, and supports new elements for better structuring of Web pages. HTML5 is adaptable to the modern interactive Web, where Web pages are not just static documents, but applications.

Working with HTML5

In this section, you learn about basic HTML5 markup and the two different syntaxes allowed within HTML5. You will learn how to choose the correct syntax for your Web pages, and review the new elements in HTML5. You will see how the new HTML5 layout elements will work, and how HTML5 supports interaction in the browser. However, keep in mind that if you are coding Web pages, you will have to wait for browser support before you start adding new HTML5 page layout elements to your code.

HTML5 is the fifth major revision of HTML. It comes long after the last major revision, which was in 1999. HTML5 attempts to address the shortcomings of previous versions of HTML, while addressing the needs of modern Web design and the application-based future of the Web. HTML5 is intended to not only describe page structure and layout, but offers a variety of new features:

- Logical layout elements, such as <nav> (for a navigation bar), <section>, <header>, and <footer>
- Elements designed for different types of rich media, including <video> and <audio>
- Animations that play directly in the browser window without plug-ins using the new <canvas> element
- Support for applications in the browser, including drag and drop, local data storage, and background data processing

HTML5 also removes some features from previous versions of HTML, as described in the following list:

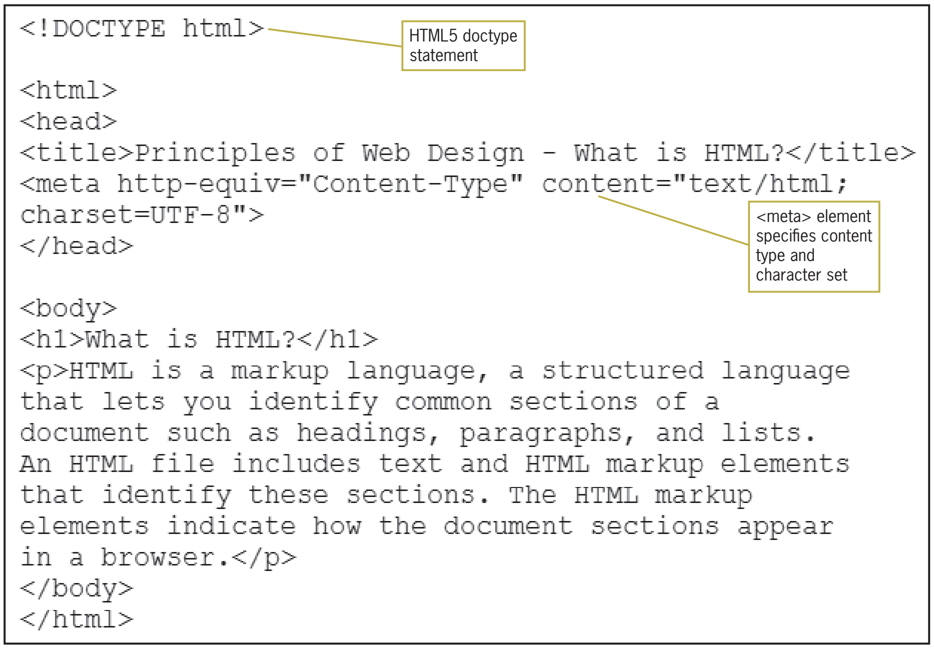
- All display elements have been removed in favor of CSS for presentation.
- Framesets and frames have been removed because of their impact on accessibility.

HTML5 is compatible with both HTML 4.01 and XHTML 1.0 and supports both the "looser" coding style associated with earlier HTML versions and a stricter syntax that is based on XML.

HTML5 looks almost exactly like previous versions of HTML. Figure 1-7 shows a sample of HTML5 code. Note two important conventions in this sample:

The HTML5 <!DOCTYPE> statement is less complicated than in previous versions of HTML.

The <meta> element specifies the document content type and character set. Many pages leave out this critical piece of information that tells the browser how to correctly interpret your HTML code.



```
<!DOCTYPE html>

<html>
<head>
<title>Principles of Web Design - What is HTML?</title>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
</head>

<body>
<h1>What is HTML?</h1>
<p>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 elements
that identify these sections. The HTML markup
elements indicate how the document sections appear
in a browser.</p>
</body>
</html>
```

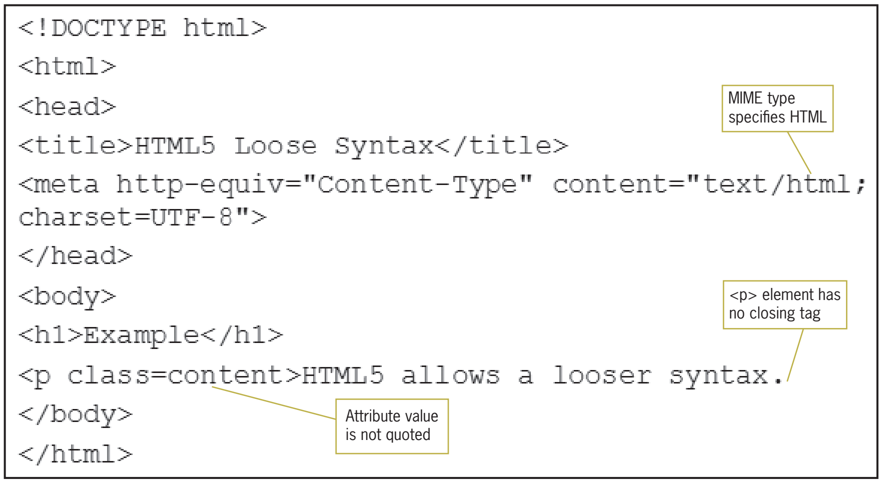
Figure 1-7 Sample HTML5 document

HTML5 Loose and Strict Syntaxes

HTML5 offers two syntaxes. One is based on HTML syntax, and the other on stricter XML syntax rules, which makes it compatible with XHTML.

HTML Version of HTML5

The HTML version of HTML5 is more relaxed and allows authors to use shortcuts in their code. Figure 1-8 shows the looser syntax with two examples of code shortcuts. Notice that the content type specifies HTML.



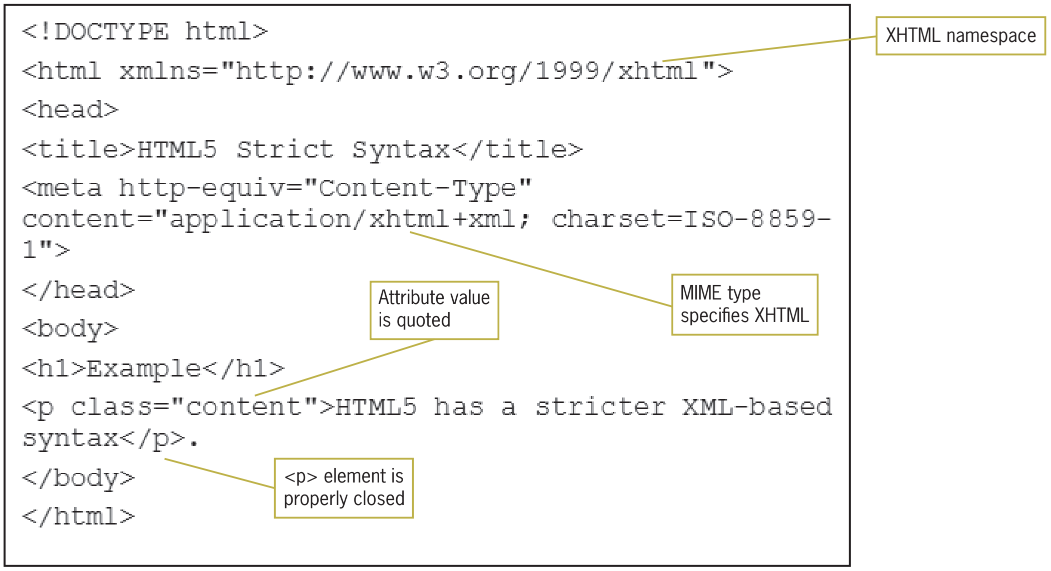
```
<!DOCTYPE html>
<html>
<head>
<title>HTML5 Loose Syntax</title>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
</head>
<body>
<h1>Example</h1>
<p class=content>HTML5 allows a looser syntax.
</body>
</html>
```

Figure 1-8 Looser HTML5 syntax

Notice in this code that the <p> element has no closing tag, and the class attribute has no quotes around the content value. This more relaxed version of HTML5 is backwards compatible with HTML 4.01. The MIME Type, described later in this section, declares the document as an HTML document.

XHTML Version of HTML5

The stricter syntax rules of HTML5 are consistent with XHTML syntax, as shown in Figure 1-9. XML syntax rules are applied to the code and the two shortcuts removed. The XHTML Namespace qualifier and XHTML MIME type declare this as an XHTML document.



```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>HTML5 Strict Syntax</title>
<meta http-equiv="Content-Type"
content="application/xhtml+xml; charset=ISO-8859-
1">
</head>
<body>
<h1>Example</h1>
<p class="content">HTML5 has a stricter XML-based
syntax</p>.
</body>
</html>
```

XHTML namespace
MIME type specifies XHTML
Attribute value is quoted
<p> element is properly closed

Figure 1-9 Stricter HTML5 syntax

Choosing the Correct Syntax

HTML5 allows a mixture of these two types of syntax into one document, called a polyglot document by the W3C. Polyglot means "mixed languages," and the name is appropriate for the type of HTML syntax that is the most usable, that is, declaring the document as HTML, but using the stricter syntax rules of XML. This lets you standardize and consistently code to a stricter, more consistent coding convention for any type of professional Web site.

The looser HTML syntax that is allowed in HTML5 is appropriate for anyone building a noncommercial Web site. However, in a professional Web development environment, it is a best practice to code using syntax that follows the basic XML syntax rules, which you will read more about later in this chapter. With XML syntax, the code you create for Web content can have multiple purposes and potentially be used in a variety of display and application environments. You can choose to code to XML standards but still declare the document MIME type as HTML, so that error handling is not a problem.

For example, it is now commonplace in the publishing industry to repurpose content that was originally destined for Web publication. This content can be organized and used in a content management system that allows output to be printed or displayed in different environments that meets users' individual needs. The multipurposing of content is often called **single-sourcing**, where one source of content is maintained but disseminated to different users or devices. Data in a single location can be maintained and updated more easily. The data also has greater value because it can be displayed and used in different ways.

Another reason for choosing XML syntax in your HTML5 code is to ensure consistent coding practices across an organization. Web site development should always formulate and apply consistent coding conventions across all Web content to ensure uniformity. With XML syntax, the rules are already created and just need to be followed. XML rules can be easily tested for compliance by using a **validator**, software that checks an HTML document for syntax errors. You can use either an online validator or a built-in validator that is common to most HTML development tools. If you create coding conventions and stick to them, your Web content will be more adaptable to different purposes, compatible with more software applications, and better prepared for future use.

Choosing the Correct Document Type and MIME type

The key to displaying your Web pages correctly in the browser is stating the correct <!DOCTYPE> and MIME type.

The DOCTYPE Statement

The <!DOCTYPE> statement originates in SGML, and was used to point to a **Document Type Definition (DTD)**, which contains the elements, attributes, and syntax rules for a language like HTML or XML.

HTML5 is no longer based on SGML, so the <!DOCTYPE> statement primarily exists to make sure that browsers display the page in standards mode. When you include a document type, the browser operates in "standards mode," presenting the document using W3C rules. If there is no document type, the browser operates in "quirks mode," which renders the page like an older browser, allowing the "quirky" code from the earlier days of Web development.

The standard doctype statement for HTML5 looks like this:

```
<!DOCTYPE html>
```

MIME Type

The **Multipurpose Internet Mail Extensions (MIME)** originated as a standard for e-mail, but has grown to defining content types for the Web. It is the MIME type that determines the type of document you are presenting. Even in a document written in strict XHTML, if the MIME type is set to text/html, then the document is served as HTML.

Every document should contain a <meta> element in the <head> section that specifies the content type and character set. The content value is the document's MIME type. The document's character set is specified as charset=utf-8, which is the standard character set for the Web.

Your <meta> element should look like this:

```
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
```

Putting all of this together, the beginning of all HTML5 documents should look like the following code, with the addition of a page name in the <title> element:

```
<!DOCTYPE html>
<html>
<head>
<title></title>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
</head>
```

You may want to create a blank template file that has this code in it, and then copy it and rename it every time you need to create a new HTML file.

Creating Syntactically Correct Code

You should follow six basic rules to create syntactically correct HTML5 code. These are the same rules used for both XML and XHTML, applied in an HTML environment, and were listed earlier in the introduction to XHTML.

- Documents must be well formed.
- All tags must nest properly and not overlap.
- Use all lowercase for element names.
- Always use closing tags.
- Empty elements are marked with a closing slash.
- Attribute values must be contained in quotation marks.

Documents Must Be Well Formed

A **well-formed document** is one that adheres to the syntax rules described in this section.

All Tags Must Nest Properly and Not Overlap

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

```
<p><strong>some text... </p></strong>
```

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

```
<p><strong>some text...</strong></p>
```

Use All Lowercase for Element Names

Even though HTML5 allows uppercase and lowercase element names, it is considered good coding practice to always use lowercase. Use all lowercase characters for element and attribute names when writing your code, this will ensure that your code can be XHTML compatible if it needs to be used in some type of content management or application processing environment.

Always Use Closing Tags

In the looser HTML5 syntax, certain elements such as the <p> element are allowed optional closing tags. For example, the following two <p> elements do not have closing tags:

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

Even though HTML5 allows this, a much better practice is to always close all elements. The following example shows this syntax.

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

Empty Elements Are Marked with a Closing Slash

Empty elements must be marked "empty" by a slash (/) in the single tag. For example, the <br> element, becomes <br />. The <img> element looks like the following:

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

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

In previous versions of HTML and XHTML, authors left a blank space in front of the slash, as in <br />. This is no longer necessary in HTML5.

Attribute Values Must Be Contained in Quotes

In the looser HTML5 syntax, attribute values do not have to be quoted as shown in the following example:

```
<p class=copy>HTML5 is the newest Web language.</p>
```

Even though HTML5 allows this, your code is more compatible when you always contain all attribute values within quotes. The preferred syntax follows:

```
<p class="copy">HTML5 is the newest Web language.</p>
```

Element Categories

In HTML5, elements are divided into categories by use, as shown in the following list. Some elements may fall into more than one category.

- Metadata content
- Flow content
- Sectioning root
- Sectioning content
- Heading content
- Phrasing content
- Embedded content
- Interactive content
- Transparent

Metadata Content

These are the elements that reside in the head section of the document, such as <title>, <script>, and <style>. These elements contain the document **metadata**, which is information about the document itself, such as how to present the document, or what other documents are related to this one, such as style sheets.

Flow Content

Flow content elements are most of the elements that are used within the body section of the document. This category includes all of the standard HTML tags, such as <p>, <div>, and <blockquote>, as well as the newer page layout elements introduced in HTML5 such as <article>, <header>, and <footer>.

Sectioning Root

Content on Web pages is divided into sections. The <body> element is the root or parent element of all content sections, making it a sectioning root. Other elements in this category include <blockquote> and <td>.

Sectioning Content

Sectioning content divides a document into sections, each of which can have its own set of headings. These elements group sections of content on the page. Elements in this category include <section>, <article>, and <nav>. Sectioning elements are an exciting feature of HTML5, and most of these elements are new. Make sure that browsers support these new elements before beginning to use them in your Web pages. In the meantime, continue to use the <div> element with classes to create sections. You will read more about this in Chapter 7, Page Layouts.

Heading Content

This category includes the heading elements, <h1> thru <h6>, plus the new <header> element for creating heading sections.

Phrasing Content

Phrasing content includes elements that are used within lines of text in a document. These include <em>, <strong>, and <span>. These elements were called inline elements in HTML 4.01.

Embedded Content

Embedded content elements load external content into the Web page. This content can be images, videos or audio files, or Adobe Flash files. The <img> element as well as the new <audio> and <video> elements are part of this category.

Interactive Content

Interactive elements let users interact with the Web page content. Interactivity depends on the user's browser and input device, such as a mouse, keyboard, touch screen, or voice input. Elements in this category include the <a> element, which creates clickable hyperlinks, plus the <audio> and <video> elements if the user can control the content, such as by using play or pause buttons. Flash animations and other content types, such as presentations or Web-based learning, also accept user interaction. Form controls are also part of this category.

Transparent

Some elements have transparent content models, which means that their allowed content is inherited from their parent element. They may contain any content that their parent element may contain, in addition to any other allowances or exceptions described for the element. For example, the <a> element often occurs within a <p> element parent, and will usually follow the content rules of its parent element.

New Elements in HTML5

HTML5 has a number of new elements, as listed in Table 1-2. Not all of these elements are supported by current browsers. Make sure to test carefully and check for browser support before using these new elements.

Element	Description
<article>	A section for the main page content, such as a newspaper or magazine article, blog entry, or any other independent item of content
<aside>	A section for side content such as a pull quote or other content related to the main article
<audio>	Contains sound, streaming audio, or other aural content
<canvas>	Lets scripting applications dynamically render graphics, animations, or other visual images
<command>	Defines a command action for interaction with the user
<datalist>	Contains drop-down list content for older browsers; used for legacy compatibility

Table 1-2 New Elements in HTML5 *(continues)*

(continued)

<details>	Contains additional information or controls that the user can obtain on demand
<embed>	Represents an insertion point for an external application or interactive content
<figure>	Contains an image or graphic with an optional caption using the <legend> element
<footer>	Typically contains information such as who wrote the Web page, links to related documents, and copyright notices
<header>	Typically contains headings and subheadings that describe the page content
<mark>	Includes a string of text in a document marked or highlighted for reference purposes
<meter>	Includes a measurement within a known range, or a fractional value
<nav>	Contains primary navigation links to other pages or to content within the page
<output>	Contains the result of a calculation
<progress>	Represents the completion progress of a task
<rp>	Stands for ruby parentheses, which hide ruby text <rt> from browsers that do not support the <ruby> element
<rt>	Contains ruby text, used with the <ruby> element
<ruby>	Allows markup of text content with ruby annotations. Ruby annotations are descriptive strings of text presented next to base text, primarily used in East Asian typography as a guide for pronunciation or to include other annotations.
<section>	Specifies a generic section of a Web page; a section, in this context, is a thematic grouping of content, typically with its own heading
<source>	Specifies multiple media resources for media elements
<time>	Contains a date or time
<video>	Contains video content

Table 1-2 New Elements in HTML5

Attributes in HTML5

Elements can contain attributes that set properties for an element. In the following code sample, the <div> element has a class attribute with a value of *article*.

```
<div class="article">
```

In earlier versions of HTML, a wide variety of attributes contained presentation information, such as the size, color, or alignment of text. In HTML5, all display information is specified with CSS, so far fewer attributes are necessary. Some elements have specific attributes, while other attributes are global and can be applied to any element. Some of the more important global attributes include style, title, and class. You will find a complete list of HTML5 attributes in Appendix A.

Obsolete Elements in HTML5

Many elements have been removed from HTML5, most of which were used for presentation effects that are better handled with a CSS style. Examples of elements that are obsolete in HTML5 include some that are commonly avoided in Web design, such as <font>. Also, the <frameset> and <frames> elements are no longer available, so frame-based sites are obsolete. The <iframe> element is still available to embed a page within a page. You will find a complete list of obsolete elements in Appendix A.

Using HTML5 Elements for Page Structure

Generally, all Web pages share some common characteristics. For example, most Web pages have some type of header, navigation, article and figure content, footers, and possibly sidebars that contain related content. In current Web design, and as you will learn in later chapters, page layout sections are currently marked up using a combination of <div> elements and id or class names, as shown in Figure 1-10.

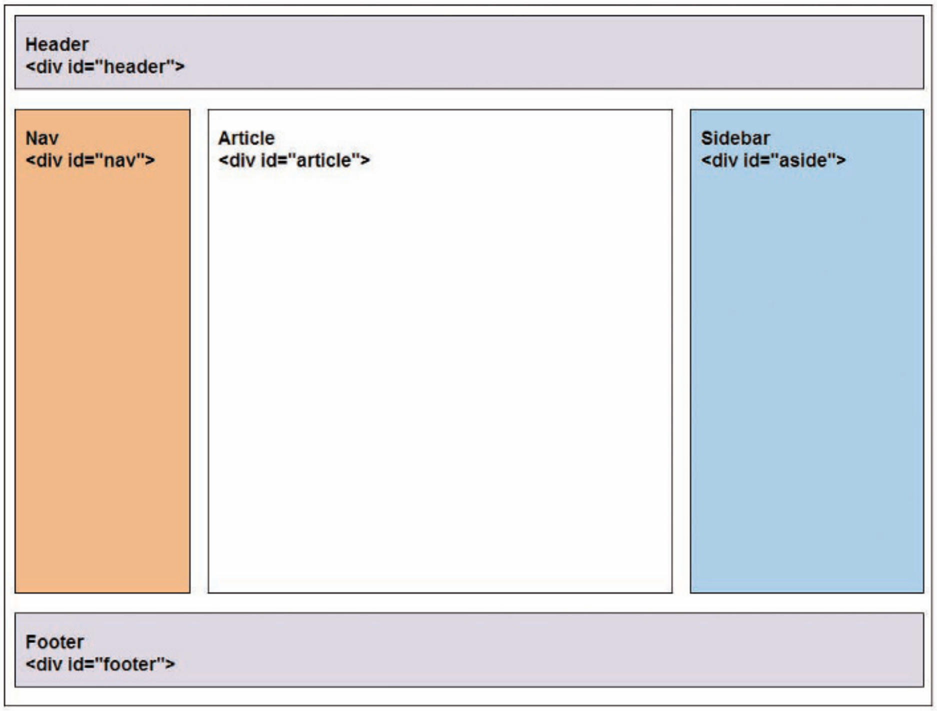


Figure 1-10 Five page sections using <div> elements and id names

The id names shown in Figure 1-10 may vary, but generally all of the page components shown occur in one form or another on every Web site. HTML5 standardizes the naming conventions for each of these sections and provides an element for each and other document components as well.

HTML5 offers a new set of elements for describing document structure. These new elements are based on the methods that have become popular on the Web as browsers offered increasing support for layouts created with CSS. These techniques supplant the use of tables for page layout, instead using the division element, or <div>, to structure the page. The <div> elements are named with an id or class attribute, such as:

```
<div id="article">This is the main content of the Web page</div>
```

HTML5 replaces the use of <div> with named elements to structure the page. The <article> element can be used instead of the <div> element, for example:

```
<article>This is the main content of the Web page</article>
```

Using HTML5, the Web page shown in Figure 1-10 can now be marked up as shown in Figure 1-11.

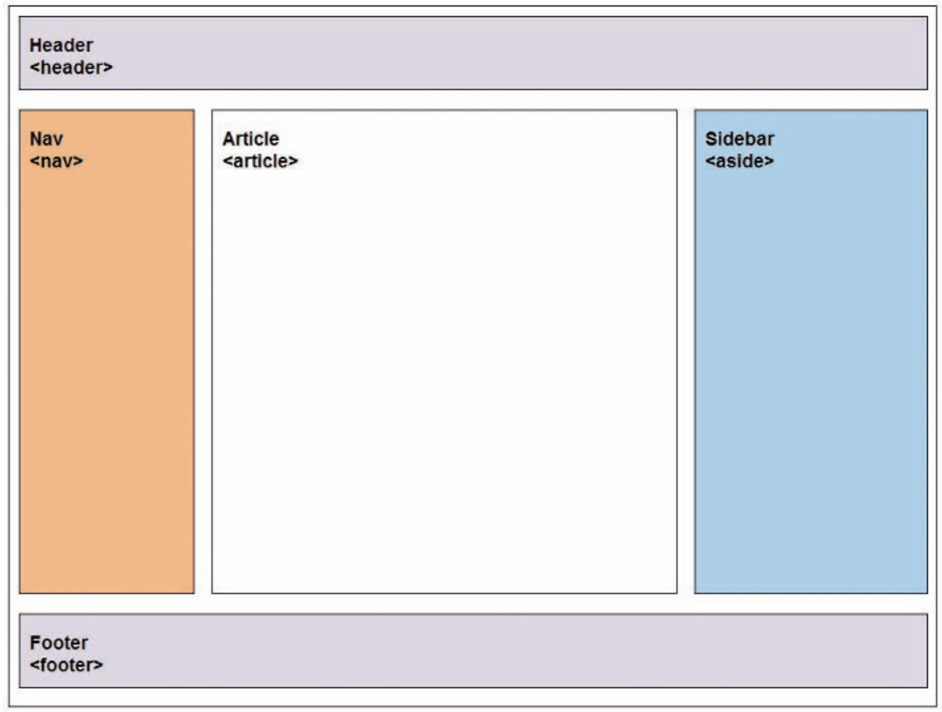


Figure 1-11 Page sections using HTML5 elements

The HTML5 elements that are designed for page layout include the following:

- <header>—Contains the page header content
- <nav>—Contains the navigation elements for the page
- <article>—Contains the primary page content
- <section>—Defines sections or groupings of the page content
- <aside>—Contains additional content such as a quote or sidebar
- <figure>—Contains images for the article content
- <footer>—Contains the page footer content, typically information such as copyrights, policies and legal information, and contact information.

These elements offer benefits for building page layouts. They standardize the naming conventions for parts of a Web page, allowing software tools to apply consistent rules for content management, presentation, and processing. They allow Web pages to share information across multiple delivery platforms and presentation devices.

The <section> element is for thematic groupings of content, each with its own heading structure. This means that each section can have up to six headings (<h1> through <h6>), which can create more complex page structures. The <section> element also allows automatically generated tables of content or outlines of page content.

As with all new advances in HTML, the browsers need to catch up to support the latest enhancements. This means developers have to add new code and prepare new releases of their software, and these new browser releases have to be accepted and downloaded by the general Web-browsing public. Remember that if you are coding Web pages, you will have to wait for browser support before you start adding these new page layout elements to your code. In the meantime, the naming conventions developed for these new elements, such as <nav> or <article>, can be used now as class names until browser support catches up. You will learn more about this in Chapter 7.

Interactive Capabilities in HTML5

HTML5 supports a number of new features for applications and interaction with users. Browser support varies for these new features, so evaluate and test them carefully before implementation.

Audio and Video

The <audio> and <video> media elements let developers embed audio or video streams into a Web page.

Drawing Canvas

The <canvas> element provides a bitmap drawing area for displaying graphs, drawings, games, or other visual elements dynamically in the browser.

Background Application Processing

"Web workers" are background scripting processes that can run in the browser while the user is performing other actions. This means that calculations, data transfers, and other background processing are available to support more complex applications.

Local Data Storage

As browser-based applications become more complex, they need to be able to store data locally on the user's machine, rather than having to interact with the server each time data needs to be read or written. This task is currently handled by **cookies**, small pieces of text-based data that is stored on the user's machine. Cookies transfer only small pieces of information with every request to the Web server. Part of the HTML5 effort is a recommendation called Web Storage that specifies how larger amounts of data can be stored and retrieved from the user's computer by Web applications.

Choosing an HTML Editor

You can create or generate HTML code to build Web pages in many ways. The two ways to create HTML are using a code-based editor or a WYSIWYG (What You See Is What You Get) editor. Some HTML editors combine a WYSIWYG view and a code-based view. Others have built-in validators that check your code for syntax errors. Many have FTP (file transfer protocol) clients that publish your Web pages to your Web host. You will read more about publishing your Web site in Chapter 3.

You do not have to spend a lot of money to acquire an HTML editor. Many are low-priced shareware and others are available free. You can also use simple text editors as HTML-editing programs. Notepad, the basic text editor that comes with Windows versions from 95 to Windows 7, is still available for a quick edit, to view page code, or to create entire Web pages. On the Macintosh, you can use TextEdit to create HTML files. Many sites on the Web are coded using these text-editing tools, which are easy to use and still relied upon by many 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.

Full-featured HTML-editing programs include Adobe Dreamweaver and Microsoft Expression Web (which replaces FrontPage), to name two. Some code-based HTML editors, such as Adobe 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. The 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 HTML syntax rules. See Chapter 3 for a list of HTML editors.

Many of the latest office applications also 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 and export them to HTML. This hands-off approach leaves much to be desired for an HTML author, however, because you give up control over the finished product. In addition, the practice of converting content from a program such as Microsoft Word to HTML is notorious for creating substandard HTML code.

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 when displayed in a browser. Furthermore, many editing packages create complex, substandard code to achieve an effect specified by the user. This complex code can cause compatibility problems across different browsers. HTML authors who are accustomed to coding by hand (in Notepad or another text editor) often are surprised to see the complex code an HTML editing package generates.

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.

Using Good Coding Practices

In the past, Web designers wrote nonstandard code, employing a "whatever works" mentality to trick the browser into presenting the results they wanted. Some examples of this are using the heading tags <h1> through <h6> solely for their font sizes rather than as logical headings, or manipulating the <table> elements into a layout tool instead of a container for data as they were intended.

These are bad habits that are best left behind. This section provides the following guidelines for creating code that ensures the greatest standards-compliance, presentation, and usefulness of your content:

- Stick to the standards.
- Use semantic markup.
- Validate your code.

Stick to the Standards

The best way to create well-coded Web sites is to strictly follow the standards set by the W3C. This approach provides the greatest acceptance and most 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 widest 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 CSS, can let you overcome many functional limitations.

Coding with standards and respecting the W3C mandate to separate content structure from presentation information makes your content more accessible and portable to different devices and destinations. Who would have envisioned that, at the outset of the Web, content would be delivered on cell phones or televisions? Who can imagine what devices will be used in the future? Content that is designed to standards is more meaningful to search engines, has better accessibility, is displayed more consistently in multiple browsers, and has a longer life and greater chance of being accessible in future applications.

The Web Standards project (*www.webstandards.org*) is "a grassroots coalition fighting for standards that ensure simple, affordable access to Web technologies for all." This site is a good place to get started on reading more about the history, trends, and current state of Web standards development.

Use Semantic Markup

Semantic markup is descriptive markup that identifies the intended use of document sections. Semantic markup accurately describes each piece of content. Although this may sound like an obvious use of markup elements, until recently this logical use of the HTML elements was largely ignored by the Web development community. For example, HTML authors know that an <h1> element signifies a block of text as a first-level heading, but many times this element is used simply for its ability to increase the font size of text. The <ul> element, which indicates a bulleted list, was used for its indenting characteristics. When you use semantic markup, the document elements match the meaning and usage of the document sections; a <p> signifies a paragraph, a <blockquote> is for a lengthy quotation, and so on. Semantically correct markup is readable not only by humans but by software as well, lending itself to improved recognition by search engines, news agents, and accessibility devices.

Semantic markup is the basis for the evolution of the Web into a universal medium for data, information, and knowledge exchange as described by the W3C in their Semantic Web Activity group (*www.w3.org/2001/sw*). Although this vision of the Web is still in early development, the emphasis on correct usage of markup elements benefits the accessibility and longevity of your Web content.

Validate Your Code

Another step towards standards compliance is to validate your code. **Valid code** conforms to the usage rules of the W3C. For example, a basic HTML rule states that only certain elements can appear in the head section. The lack of valid code is a major problem for the future of a standards-based Web. A recent survey of 2.4 million Web pages found that 99 percent did not meet W3C standards. Although this number may have decreased since the study, a quick survey of almost any Web site shows that few have valid code.

Valid code enhances browser compatibility, accessibility, and exchange of data. Whether you write HTML or XHTML code, you should validate your code to make sure it conforms to W3C standards. Validation is so easy to perform, it is hard to understand why many Web designers apparently ignore it. To validate your code, simply use a software program called a validator to read your code and compare it to the rules in the DTD. The validator generates a list of validation errors. Many HTML editors contain built-in validators, or you can use a Web-based validator such as the W3C validation service at *http://validator.w3.org*, which can validate HTML5.

Using a validator is an eye-opening experience for any Web designer. The most common mistakes that make your code invalid include:

A new offering from theW3C is Unicorn, an all-in-one validator that checks both HTML and CSS code for errors on any Web page you specify. You can also upload a file if you do not have a live Web site (*http://validator.w3.org/unicorn*).

- No doctype declaration
- Missing closing tags
- Missing alt attributes in <img> elements
- Incorrect tag nesting
- Unquoted attributes

If you compare this list to the XML syntax rules listed earlier in this chapter, you can see how moving to standardized coding practices helps clean up most of these common coding errors.

Migrating from Legacy HTML to HTML5

If you are moving your existing Web site to HTML5, the transition should be a gradual process rather than an abrupt shift to the new language. The looser syntax of HTML 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

HTML5, you will be cleaning up code on existing pages, planning coding conventions for new pages, removing deprecated elements, 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.

The following list outlines steps you need to take to migrate from legacy HTML to HTML5:

1. *Evaluate existing code*—Check for basic compliance with consistent 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 presentation information*—How much of your code includes obsolete elements such as <font> and obsolete attributes such as "bgcolor," "face," and "size"? On many sites, this information can make up as much as 50 percent of the existing code. Start thinking about how you can convert these presentation characteristics to 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*—Remember 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.

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:

- Make sure to check for support of new HTML5 elements, and test carefully before adding them to your Web site.
- Use the HTML5 naming conventions to name the content sections of your site, even if only using them as class or id names. This will help you ease the migration to the new HTML5 page layout elements as they become supported by browsers.
- Use Cascading Style Sheets. The style enhancements and control offered by this style language are impressive, but are not evenly supported by older browsers. Implement CSS gradually, testing for browser compatibility as you go.
- Decide whether to code to the XML 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 XML standards to ensure future compatibility with new tools and browsers.
- Use good coding practices by writing standards-based markup that always includes a document type and content type and is correctly validated.
- Choose the type of editing tool needed to create your HTML code. If you choose a WYSIWYG editor, make sure it creates standards-based code. Avoid using word-processing programs to create Web pages.

Key Terms

Cascading Style Sheets (CSS)—A style language, created by the W3C, that allows complete specifications of style for HTML documents. CSS allows HTML authors to write style rules that affect the display of Web pages. CSS style information is contained either within an HTML document, or in external documents called style sheets.

cookie—A small piece of text-based data that a Web page stores on the user's machine. Cookies transfer small pieces of information with every request to the Web server.

deprecated element—An element that the W3C has identified as obsolete.

Document Type (doctype)—The section of an HTML document that specifies the rules for the document language so the browser knows how to interpret the HTML code and display it properly.

Document Type Definition (DTD)—A set of rules that contains all the elements, attributes, and usage rules for the markup language you are using.

Extensible Hypertext Markup Language (XHTML)—XHTML is HTML 4.01 reformulated as an application of XML.

hypertext—A nonlinear way of organizing information. When you are using a hypertext system, you can skip from one related topic to another, find the information that interests you, and then return to your starting point or move on to another related topic of interest.

Hypertext Markup Language (HTML)—The markup language that defines the structure and display properties of a Web page. The HTML code is interpreted by the browser to create the displayed results. HTML is an application of SGML. *See also* Standard Generalized Markup Language.

markup language—A structured language that lets you identify common elements of a document such as headings, paragraphs, and lists.

metadata—Information about the document itself, such as how to present the document, or what other documents are related to the current one, such as style sheets.

Multipurpose Internet Mail Extensions (MIME)—Originally a standard for e-mail, MIME now defines content types for the Web. It determines the type of document presented in an HTML file.

rendering engine—A program contained in every browser that interprets the markup tags in an HTML file and displays the results in the browser.

root element—The container element for all other elements in the document. In an HTML document, the root element is <html>.

semantic markup—Descriptive markup that identifies the intended use of document sections. Semantic markup accurately describes each piece of content.

single-source—To create content that can serve multiple purposes and be distributed to different users or devices.

Standard Generalized Markup Language (SGML)—A standard system for specifying document structure using markup tags.

style sheet—A set of style rules that describes a document's display characteristics. There are two types of style sheets: internal and external.

valid code—Markup code that conforms to the usage rules of the W3C.

validator—A software program that checks an HTML document for syntactical errors.

Web page—A text document that is interpreted and displayed by Web browser software.

well-formed document—A syntactically correct XML or XHTML file.

World Wide Web Consortium (W3C)—Founded in 1994 at the Massachusetts Institute of Technology to standardize Web markup languages. The W3C, led by Tim Berners-Lee, sets standards for markup languages and provides an open, nonproprietary forum for industry and academic representatives to add to the evolution of HTML.

Review Questions

1. What does the <!DOCTYPE> statement specify?
2. What is the function of the root element?
3. What are the main sections of an HTML document?
4. Why is the content of the <title> element important?
5. How should display information be expressed for a Web page?
6. What function does the browser's rendering engine perform?
7. What are the advantages of using an external style sheet?
8. What feature distinguishes XML from HTML?
9. What are the two HTML5 syntaxes?
10. What is a well-formed document?
11. What is document metadata?

12. Where is metadata stored?

13. List three new HTML5 elements that are designed for page layout.

14. What is the function of the new HTML5 <canvas> element?

15. What does *semantic markup* mean?

Hands-On Projects

1. Download and install the latest versions of the following browsers onto your computer as necessary:
   - Internet Explorer
   - Firefox
   - Opera
   - Safari
   - Google Chrome

2. Build a basic HTML file and test it in the browser.

   a. Copy the **project1.htm** file from the Chapter01 folder provided with your Data Files to the Chapter01 folder in your work folder. (Create the Chapter01 folder, if necessary.)

   b. In your HTML editor, open **project1.htm** and examine the code. Notice that only the basic HTML elements are included to create the head and body sections of the document.

```
<!DOCTYPE html>

<html>
<head>

</head>
<body>



</body>
</html>
```

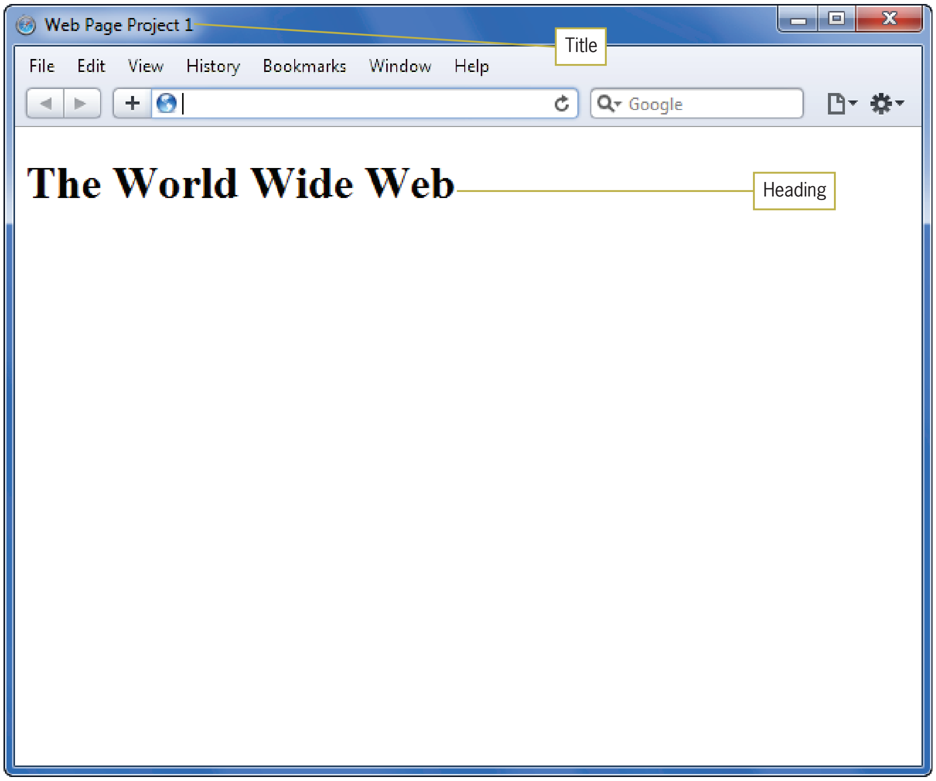
c. Add the <title> element with a title for your document and a <meta> element that defines the document type as shown in color in the following code.

```
<!DOCTYPE html>

<html>
<head>
<title>Web Page Project 1</title>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
</head>
<body>




</body>
</html>
```

d. Add an <h1> element with a heading for the document. The <h1> element must be contained within the <body> element.

```
<!DOCTYPE html>

<html>
<head>
<title>Web Page Project 1</title>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
</head>
<body>
<h1>The World Wide Web</h1>




</body>
</html>
```

e. Save your file and view it in your browser. It should look like Figure 1-12.



The World Wide Web

Figure 1-12 Adding a title and heading to the Web page

f. Add two paragraphs of content immediately following the <h1> element as shown.

```
<!DOCTYPE html>

<html>
<head>
<title>Web Page Project 1</title>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
</head>
<body>
<h1>The World Wide Web</h1>
<p>The World Wide Web, abbreviated as WWW and
commonly known as the Web, is a system of
interlinked hypertext documents accessed via the
Internet.</p>
<p>With a web browser, one can view web pages
that may contain text, images, videos, and other
multimedia and navigate between them by using
```

```
hyperlinks.</p>
</body>
</html>
```

g. Add one more paragraph and a bulleted list as shown.

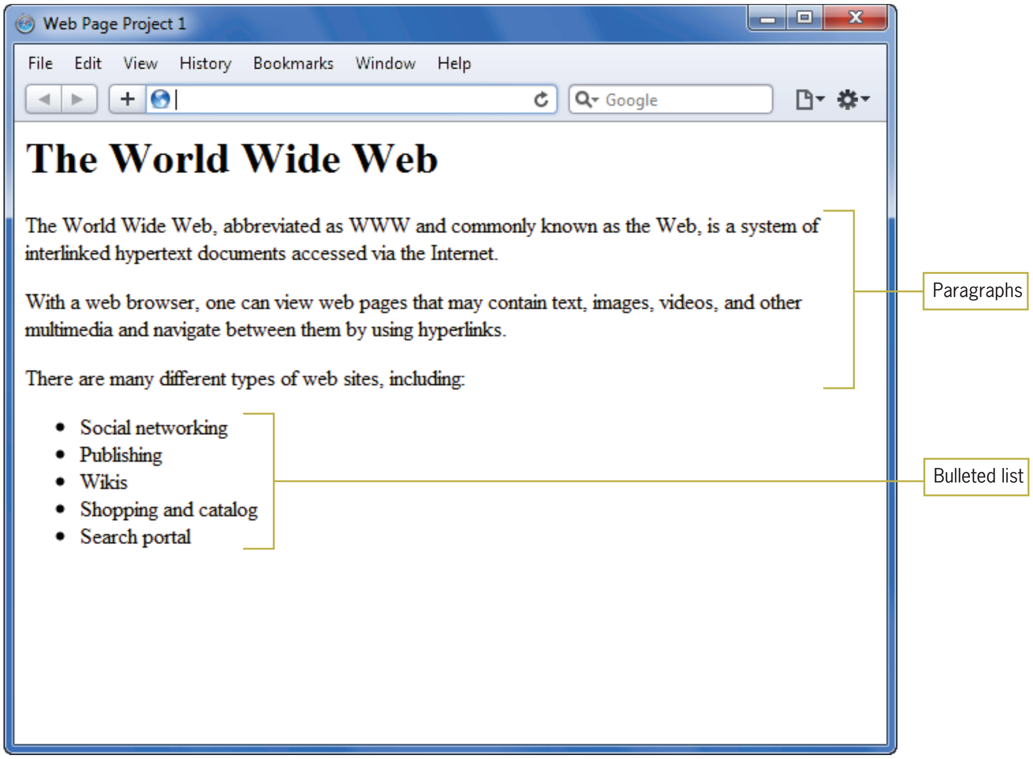
```
<html>
<head>
<title>Web Page Project 1</title>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
</head>
<body>
<h1>The World Wide Web</h1>
<p>The World Wide Web, abbreviated as WWW and
commonly known as the Web, is a system of
interlinked hypertext documents accessed via the
Internet.</p>
<p>With a web browser, one can view web pages
that may contain text, images, videos, and other
multimedia and navigate between them by using
hyperlinks.</p>
<p>There are many different types of web sites,
including:</p>
<ul>
<li>Social networking</li>
<li>Publishing</li>
<li>Wikis</li>
<li>Shopping and catalog</li>
<li>Search portal</li>
</ul>
</body>
</html>
```

h. Add a comment at the bottom of the page that includes your name as shown. Comments do not appear in the browser window.

```
<html>
<head>
<title>Web Page Project 1</title>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
</head>
<body>
<h1>The World Wide Web</h1>
<p>The World Wide Web, abbreviated as WWW and
commonly known as the Web, is a system of
interlinked hypertext documents accessed via the
Internet.</p>
```

```
<p>With a web browser, one can view web pages
that may contain text, images, videos, and other
multimedia and navigate between them by using
hyperlinks.</p>
<p>There are many different types of web sites,
including:</p>
<ul>
<li>Social networking</li>
<li>Publishing</li>
<li>Wikis</li>
<li>Shopping and catalog</li>
<li>Search portal</li>
</ul>
<!-- Web page project #1 by Your Name -->
</body>
</html>
```

i. Save your file and view it in your browser. It should look like Figure 1-13.



The World Wide Web

The World Wide Web, abbreviated as WWW and commonly known as the Web, is a system of interlinked hypertext documents accessed via the Internet.

With a web browser, one can view web pages that may contain text, images, videos, and other multimedia and navigate between them by using hyperlinks.

There are many different types of web sites, including:

- Social networking
- Publishing
- Wikis
- Shopping and catalog
- Search portal

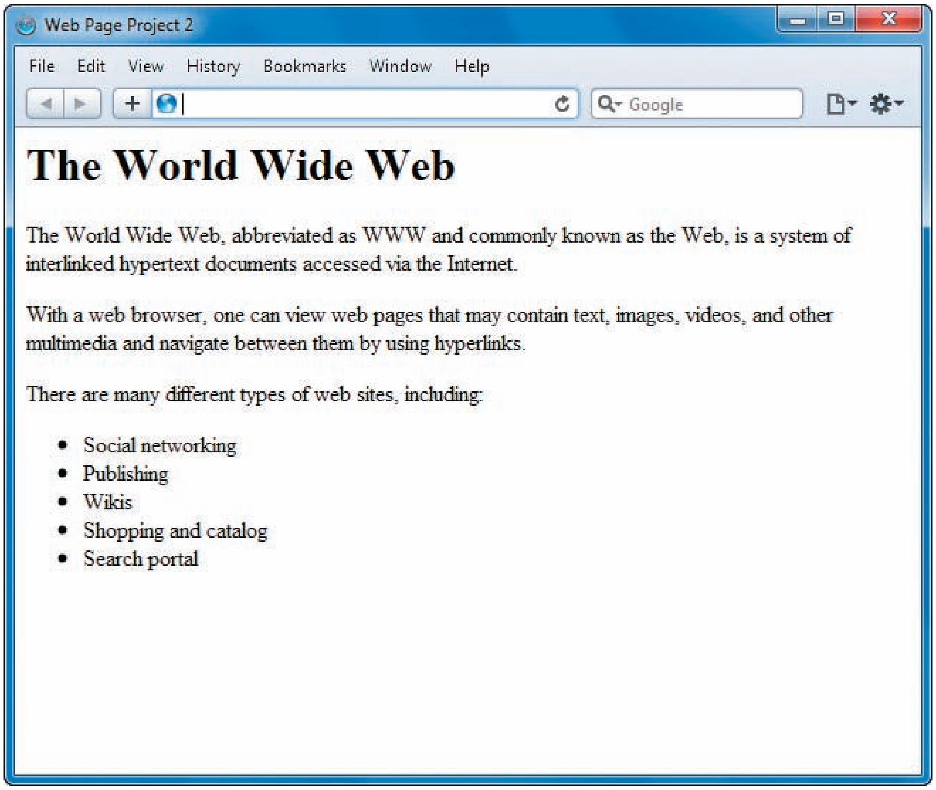
Figure 1-13 Completed project1.htm file

3. Add CSS style rules to a basic HTML file and test it in the browser.

   a. Create a copy of the **project1.htm** file and rename it **project2.htm**.

   b. In an HTML editor, open the **project2.htm** file. Change the title to Web Page Project 2. Change the comment at the end of the document to Web page project #2 by Your Name. Save the file.

   c. Open the **project2.htm** file in your browser. It will look Figure 1-14.



The World Wide Web

The World Wide Web, abbreviated as WWW and commonly known as the Web, is a system of interlinked hypertext documents accessed via the Internet.

With a web browser, one can view web pages that may contain text, images, videos, and other multimedia and navigate between them by using hyperlinks.

There are many different types of web sites, including:

- Social networking
- Publishing
- Wikis
- Shopping and catalog
- Search portal

Figure 1-14 Beginning project2.htm file

d. Return to the HTML editor and add a <style> element in the <head> section to contain your style rules, as shown in color in the following code. Leave a line or two of white space between the <style> tags to contain the style rules.

```
<html>
<head>
<title>Web Page Project 2</title>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
<style type="text/css">

</style>
</head>
<body>
<h1>The World Wide Web</h1>
<p>The World Wide Web, abbreviated as WWW and
commonly known as the Web, is a system of
interlinked hypertext documents accessed via the
Internet.</p>
<p>With a web browser, one can view web pages
that may contain text, images, videos, and other
multimedia and navigate between them by using
hyperlinks.</p>
<p>There are many different types of web sites,
including:</p>
<ul>
<li>Social networking</li>
<li>Publishing</li>
<li>Wikis</li>
<li>Shopping and catalog</li>
<li>Search portal</li>
</ul>
<!-- Web page project #1 by Your Name -->
</body>
</html>
```

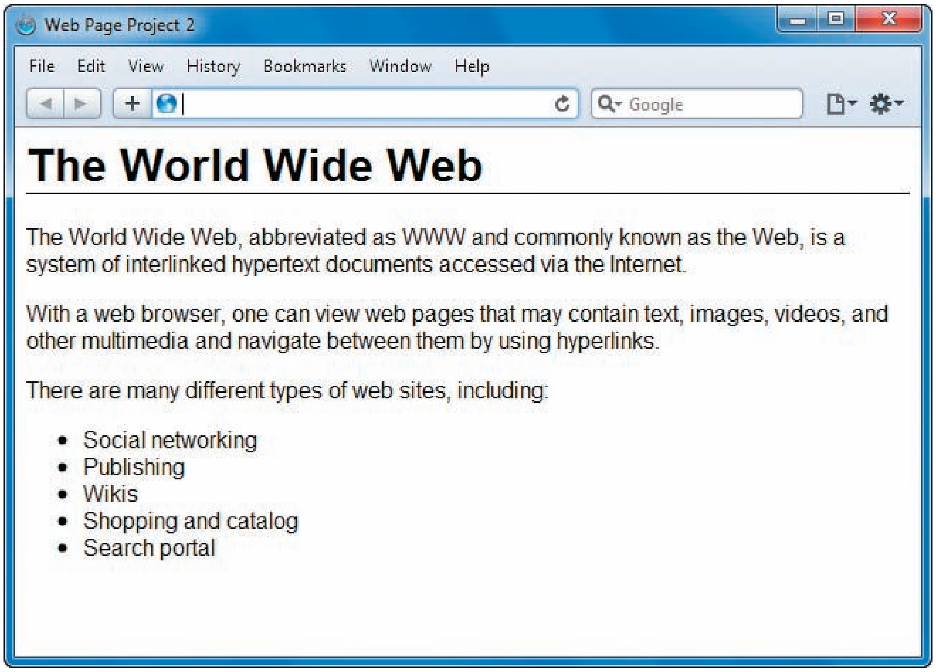
e. Add a style rule that sets the font-family for the page as shown below. Notice the style rule contains both a specific style (Arial) and a fallback style (sans-serif) in case the user does not have Arial as an installed font.

```
<style type="text/css">
body {font-family: arial, sans-serif;}
</style>
```

f. Write another style rule that adds a solid thin bottom border to the <h1> heading element.

```
<style type="text/css">
body {font-family: arial, sans-serif;}
h1 {bottom-border: solid thin;}
</style>
```

g. Save your file and view it in your browser. It should look like Figure 1-15.



The World Wide Web

The World Wide Web, abbreviated as WWW and commonly known as the Web, is a system of interlinked hypertext documents accessed via the Internet.

With a web browser, one can view web pages that may contain text, images, videos, and other multimedia and navigate between them by using hyperlinks.

There are many different types of web sites, including:

- Social networking
- Publishing
- Wikis
- Shopping and catalog
- Search portal

Figure 1-15 Completed project2.htm file

4. Visit the World Wide Web Consortium Web site, and find the Web-based validator at *validator.w3.org/unicorn*. Validate the code from the previous two exercises.

5. Convert a text document to HTML from scratch. Open a text file, save it as an HTML file, and then add all markup elements to create a basic Web page. View the browser result, and then validate the file.

6. Download and install at least two different HTML-editing programs that have code-based editing abilities. Write a short report detailing the capabilities of each. Create an HTML document that contains your findings and is viewable in the browser.

Individual Case Project

To complete the ongoing Individual Case Project for this book, you must create a complete stand-alone Web site. The site must have 6–10 pages that display at least three levels of information. You can choose your own content. For example, you can focus on a work-related topic, create a personal interest site, or design a site for your favorite nonprofit organization. The site will be evaluated for cohesiveness, accessibility, compliance with W3C standards, and visual 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, using 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 Chapter 2, 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, "Planning the Site," 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.

Team Case Project

To complete the ongoing Team Case Project for this book, you and your team must create a complete stand-alone Web site. Your team should ideally consist of 3–4 members assigned by your instructor. The site must contain between 16 and 20 pages that display at least three levels of information. You will choose your own topic. For example, you can focus on a work-related topic, create a personal interest site, or develop a site for a fictional organization. The site will be evaluated for cohesiveness, accessibility, compliance with W3C standards, and visual 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, using the following outline. As you progress through the remaining chapters of the book, you will complete different facets of the Web site construction, resulting in a complete Web site.

Project Proposal

Collaborate to 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 Chapter 2, 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.
- *Development roles*—Identify each team member and individual responsibilities for the project.
- *Need*—Describe the need the Web site will satisfy. What is the purpose of the site? Is there an interest group whose needs are not satisfied? Is there a target niche you are trying to fill?
- *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, "Planning the Site," 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.
- *Development schedule, milestones, and deliverables*—Using the dates in your class syllabus as a basis, build a development schedule that indicates milestones and deliverables for each team member.