Introduction to XHTML

- **Origins and Evolution of HTML and XHTML**
- **Basic Syntax**
- **Standard XHTML Document Structure**
- **Basic Text Markup**
- **Images**
- **Hypertext Links**
- **Lists**
- **Tables**
- **Forms**
- **Syntactic Differences between HTML and XHTML**

This chapter introduces the most commonly used subset of the eXtensible Hypertext Markup Language (XHTML). Because of the simplicity of XHTML, the discussion moves quickly. The chapter begins with a brief history of the evolution of HTML and XHTML, followed by a description of the form of tags and the structure of an XHTML document. Then, tags used to specify the presentation of text are discussed, including those for line breaks, paragraph breaks, headings, and block quotations, as well as tags for specifying the style and relative size of fonts. This discussion is followed by a description of the formats and uses of images in Web documents. Next, hypertext links are introduced. Three kinds of lists—ordered, unordered, and definition—are then covered. After that, the XHTML tags and attributes used to specify tables are discussed. The next section of the chapter introduces forms, which provide the means to collect information from Web clients. Finally, the last section describes the syntactic differences between HTML and XHTML.

➢ Origins and Evolution of HTML and XHTML

HTML is defined with the use of the Standard Generalized Markup Language (SGML), which is an International Standards Organization (ISO) standard notation for describing text-formatting languages. The original intent of HTML was different from those of other text-formatting languages, which dictate all of the presentation details of text, such as font style, size, and color. Rather, HTML was designed to specify document structure at a higher and more abstract level, necessary because HTML specified documents had to be displayable on a variety of computer systems using different browsers.

The addition of style sheets to HTML in the late 1990s advanced its capabilities closer to those of other text-formatting languages by providing ways to include the specification of presentation details. These specifications are introduced in Chapter, “Cascading Style Sheets.”

➢ Versions of HTML and XHTML

The original version of HTML was designed in conjunction with the structure of the Web and the first browser, at Conseil Européen pour la Recherche Nucléaire (CERN), or European Laboratory for Particle Physics. Use of the Web began its meteoric rise in 1993 with the release of MOSAIC, the first graphical Web browser. Not long after MOSAIC was commercialized and marketed by Netscape, Microsoft began developing its browser, Internet Explorer (IE). The release of IE marked the beginning of a four-year marketing competition between Netscape and Microsoft. During this time, both companies worked feverously to develop their own extensions to HTML in an attempt to gain market advantage. Naturally, this competition led to incompatible versions of HTML, both between the two developers and also between older and newer releases within the same company. All of these differences made it a serious challenge to Web content providers to design HTML documents that could be viewed by the different browsers.

In late 1994, Tim Berners-Lee, who developed the initial version of HTML, started the World Wide Web Consortium (W3C), which had as one of its primary purposes to develop and distribute standards for Web technologies, starting with HTML. The first HTML standard, HTML 2.0, was released in 1995. It was followed by HTML 3.2 in early 1997. Up to this point, W3C was trying to catch up with the browser makers, and HTML 3.2 was really just a reflection of the then-current features that had been developed by Netscape and Microsoft. Fortunately, after 1997 the evolution of HTML was dominated by W3C, in part because Netscape had abandoned its browser competition with Microsoft. The browsers produced by the two companies have since drifted ever closer to W3C standards.

The latest version of HTML, 4.01, was approved by W3C in late 1999. The XHTML 1.0 standard was approved in early 2000. XHTML 1.0 is a redefinition of HTML 4.01 using XML. XHTML 1.0 is actually three standards: Strict, Transitional, and Frameset. The Strict standard requires all of XHTML 1.0 be followed. The Transitional standard allows deprecated features (see last paragraph of section) of XHTML 1.0 to be included. The Frameset standard allows the collection of frame elements and attributes to be included, although they have been deprecated. The XHTML 1.1 standard was recommended by W3C in May 2001. This standard, primarily a modularization of XHTML 1.0, drops some of the features of its predecessor—most notably, frames. Work on XHTML 2.0 is underway.

There is a problem with the MIME types used to serve XHTML documents. Because most XHTML documents are currently served with the html/text MIME type, which is incorrect, problems are created with the validation of XHTML 1.1 documents, but not XHTML 1.0 Strict documents. To avoid the issue, all documents in this book are written against the XHTML 1.0 Strict standard.

The latest versions of the most popular browsers—Microsoft Internet Explorer 8 (IE8) and Firefox 3 (FX3)—come close to supporting all of XHTML 1.1.

The addition of presentation details through style sheets in HTML 4.0 made some features of earlier versions obsolete. These features, as well as some others, have been *deprecated*, meaning that they will be dropped from HTML at some time in the future. Deprecating a feature is a warning to users to stop using it because it will not be supported forever. Although even the latest releases of browsers still support the deprecated parts of HTML, we do not include descriptions of them in this book.

➢ HTML versus XHTML

On the one hand, there are some commonly heard arguments for using HTML rather than XHTML, especially XHTML 1.0 Strict. First, because of its lax syntax rules, HTML is much easier to write, whereas XHTML requires a level of discipline many of us naturally resist. Second, because of the huge number of HTML documents available on the Web, browsers will continue to support HTML as far as one can see into the future. Indeed, some older browsers have problems with some parts of XHTML.

On the other hand, there are strong reasons that one should use XHTML. One of the most compelling is that quality and consistency in any endeavor, be it electrical wiring, software development, or Web document development, rely on standards. HTML has few syntactic rules, and HTML processors (e.g., browsers) do not enforce the rules it does have. Therefore, HTML authors have a high degree of freedom to use their own syntactic preferences to create documents. Because of this freedom, HTML documents lack consistency, both in low-level syntax and in overall structure. By contrast, XHTML has strict syntactic rules that impose a consistent structure on all XHTML documents. Furthermore, the fact that there are a large number of poorly structured HTML documents on the Web is a poor excuse for generating more.

Another significant reason for using XHTML is that when you create an XHTML document, its syntactic correctness can be checked, either by an XML browser or by a validation tool (see Section 2.4). This checking process may find errors that could otherwise go undetected until after the document is posted on a site and requested by a client, possibly then only by a specific browser.

The argument that XHTML is difficult to write correctly is obviated by the availability of XHTML editors, which provide a simple and effective approach to creating syntactically correct XHTML documents.[4]

It is also possible to convert legacy HTML documents to XHTML documents by using software tools. Tidy, which is available at http://tidy.source-forge.net, is one such tool.

The remainder of this chapter provides an introduction to the most commonly used tags and attributes of XHTML 1.0.

➢ Basic Syntax

The fundamental syntactic units of HTML are called *tags*. In general, tags are used to specify categories of content. For each category, a browser has default presentation specifications for the specified content. The syntax of a tag is the tag's name surrounded by angle brackets (<and >). Tag names must be written in all lowercase letters. Most tags appear in pairs: an opening tag and a closing tag. The name of a closing tag is the name of its corresponding opening tag with a slash attached to the beginning. For example, if the tag's name is p, the corresponding closing tag is named /p. Whatever appears between a tag and its closing tag is the *content* of the tag. A browser display of an XHTML document shows the content of all of the document's tags; it is the information the document is meant to portray. Not all tags can have content.

The opening tag and its closing tag together specify a container for the content they enclose. The container and its content together are called an *element*.

Consider the following element:

```
<p> This is simple stuff. </p>
```

The paragraph tag, <p>, marks the beginning of the content; the </p>tag marks the end of the content of the paragraph element.

Attributes, which are used to specify alternative meanings of a tag, can appear between an opening tag's name and its right angle bracket. They are specified in keyword form, which means that the attribute's name is followed by an equals sign and the attribute's value. Attribute names, like tag names, are written in lowercase letters. Attribute values must be delimited by double quotes.

Comments in programs increase the readability of those programs. Comments in XHTML have the same purpose. They can appear in XHTML in the following form:

```
<!-- anything except two adjacent dashes -->
```

Browsers ignore XHTML comments—they are for people only. Comments can be spread over as many lines as are needed. For example, you could have the following comment:

```
<!-- PetesHome.html
     This document describes the home document of
     Pete's Pickles
     -->
```

Besides comments, several other kinds of text that are ignored by browsers may appear in an XHTML document. Browsers ignore all unrecognized tags. They also ignore line breaks. Line breaks that show up in the displayed content can be specified, but only with tags designed for that purpose. The same is true for multiple spaces and tabs.

Programmers find XHTML a bit frustrating. In a program, the statements specify exactly what the computer must do. XHTML tags are treated more like suggestions to the browser. If a reserved word is misspelled in a program, the error is usually detected by the language implementation system and the program is not executed. However, a misspelled tag name usually results in the tag being ignored by the browser, with no indication to the user that anything has been left out. Browsers are even allowed to ignore tags that they recognize. Furthermore, the user can configure his or her browser to react to specific tags in different ways.

➢ Standard XHTML Document Structure

Every XHTML document must begin with an xmldeclaration element that simply identifies the document as being one based on XML. This element includes an attribute that specifies the version number, which is still 1.0. The xmldeclaration usually includes a second attribute, encoding, which specifies the encoding used for the document. In this book, we use the Unicode encoding, utf-8. Following is the xmldeclaration element, which should be the first line of every XHTML document:

```
<?xml version = “1.0” encoding = “utf-8”?>
```

Note that this declaration must begin in the first character position of the document file.

The xml declaration element is folllowed immediately by an SGML DOCTYPE command, which specifies the particular SGML document-type definition (DTD) with which the document complies, among other things.[5] The following command states that the document in which it is included complies with the XHTML 1.0 Strict standard:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml1-strict.dtd">
```

A complete explanation of the DOCTYPEcommand requires more effort, both to write and to read, than is justified at

this stage of our introduction to XHTML. An XHTML document must include the four tags <html>, <head>, <title>, and <body>. The <html>tag identifies the root element of the document.
So, XHTML documents always have an <html>tag immediately following the DOCTYPEcommand, and they always end with the closing htmltag, </html>. The htmlelement includes an attribute, xmlns, that specifies the XHTML namespace, as shown in the following element:

```
<html xmlns = “http://www.w3.org/1999/xhtml”>
```

Although the xmlnsattribute’s value looks like a URL, it does not specify a document. It is just a name that happens to have the form of a URL. Namespaces are discussed in Chapter 7, “Introduction to XML.”

An XHTML document consists of two parts, named the *head* and the *body*. The <head> element contains the head part of the document, which provides information about the document and does not provide the content of the document. The body of a document provides the content of the document.

The content of the title element is displayed by the browser at the top of its display window, usually in the browser window’s title bar.

Basic Text Markup

This section describes how the text content of an XHTML document can be formatted with XHTML tags. By *formatting,* we mean layout and some presentation details. For now, we will ignore the other kinds of content that can appear in an XHTML document.

Paragraphs

Text is normally organized into paragraphs in the body of a document. The XHTML standard does not allow text to be placed directly in a document body. Instead, textual paragraphs appear as the content of a paragraph element, specified with the tag <p>. In displaying the content of a paragraph, the browser puts as many words as will fit on the lines in the browser window. The browser supplies a line break at the end of each line. As stated in Section 2.2, line breaks embedded in text are ignored by the browser. For example, the following paragraph might[6] be displayed by a browser as shown in Figure 2.1:

```
<p>
   Mary had
a
   little lamb, its fleece was white as snow. And
 everywhere that
  Mary went, the lamb
 was sure to go.
</p>
```

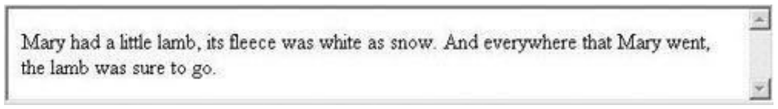

Mary had a little lamb, its fleece was white as snow. And everywhere that Mary went, the lamb was sure to go.

Figure 2.1 Filling lines

Notice that multiple spaces in the source paragraph element are replaced by single spaces in Figure 2.1.
The following is our first example of a complete XHTML document:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml1-strict.dtd">

<!-- greet.html
     A trivial document
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Our first document </title>
  </head>
  <body>
    <p>
      Greetings from your Webmaster!
    </p>
  </body>
</html>
```

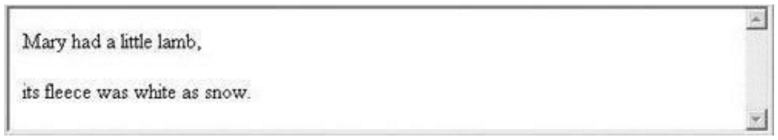
Figure 2.2 shows a browser display of greet.html.

Greetings from your Webmaster!

Figure 2.2 Display of greet.html

If the content of a paragraph tag is displayed at a position other than the beginning of the line, the browser breaks the current line and inserts a blank line. For example, the following line would be displayed as shown in Figure 2.3:

```
<p> Mary had a
little lamb, </p>
<p> its fleece was
white as snow.
</p>
```


Mary had a little lamb,

its fleece was white as snow.

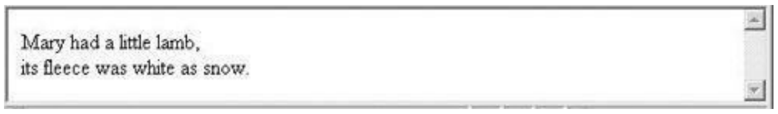
Figure 2.3 The paragraph element

Sometimes text requires a line break without the preceding blank line. This is exactly what the break tag does. The break tag differs syntactically from the paragraph tag in that it can have no content and therefore has no closing tag (because a closing tag would serve no purpose). The break tag is specified as <br />. The slash indicates that the tag is both an opening and closing tag. The space before the slash represents the absent content.[7]

Consider the following markup:

```
<p>
Mary had a little lamb, <br />
  its fleece was white as snow.
</p>
```

This markup would be displayed as shown in Figure 2.4.



Mary had a little lamb,
its fleece was white as snow.

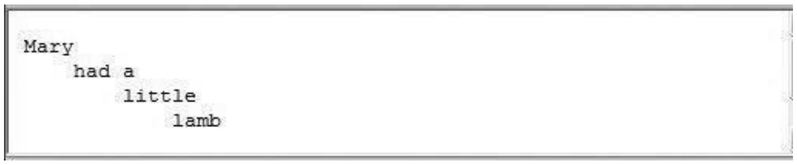
Figure 2.4 Line breaks

Preserving White Space

Sometimes it is desirable to preserve the white space in text—that is, to prevent the browser from eliminating multiple spaces and ignoring embedded line breaks. This can be specified with the pretag—for example,

```
<p><pre>
Mary
    had a
        little
            lamb
</pre>
```

This markup would be displayed as shown in Figure 2.5. Notice that the content of the preelement is shown in monospace, rather than in the default font.



```
Mary
    had a
        little
            lamb
```

Figure 2.5 The pre element

A preelement can contain virtually any other tags, except those that cause a paragraph break, such as paragraph elements.

Headings

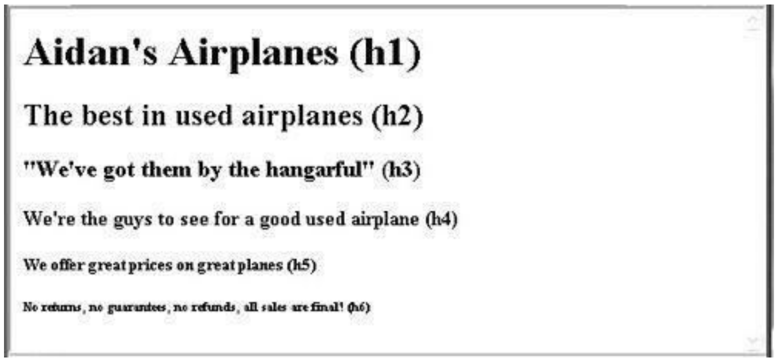
Text is often separated into sections in documents by beginning each section with a heading. Larger sections sometimes have headings that appear more prominent than headings for sections nested inside them. In XHTML, there are six levels of headings, specified by the tags <h1>, <h2>, <h3>, <h4>, <h5>, and <h6>, where <h1>specifies the highest-level heading. Headings are usually displayed in a boldface font whose default size depends on the number in the heading tag. On most browsers, <h1>, <h2>, and <h3>use font sizes that are larger than that of the default size of text, <h4> uses the default size, and <h5>and <h6>use smaller sizes. The heading tags always break the current line, so their content always appears on a new line. Browsers usually insert some vertical space before and after all headings.

The following example illustrates the use of headings:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- headings.html
     An example to illustrate headings
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
```

```
  <head> <title> Headings </title>
  </head>
  <body>
    <h1> Aidan's Airplanes (h1) </h1>
    <h2> The best in used airplanes (h2) </h2>
    <h3> "We've got them by the hangarful" (h3) </h3>
    <h4> We're the guys to see for a good used airplane (h4) </h4>
    <h5> We offer great prices on great planes (h5) </h5>
    <h6> No returns, no guarantees, no refunds,
         all sales are final! (h6) </h6>
  </body>
</html>
```

Figure 2.6 shows a browser display of headings.html.



Aidan's Airplanes (h1)
The best in used airplanes (h2)
"We've got them by the hangarful" (h3)
We're the guys to see for a good used airplane (h4)
We offer great prices on great planes (h5)
No returns, no guarantees, no refunds, all sales are final! (h6)

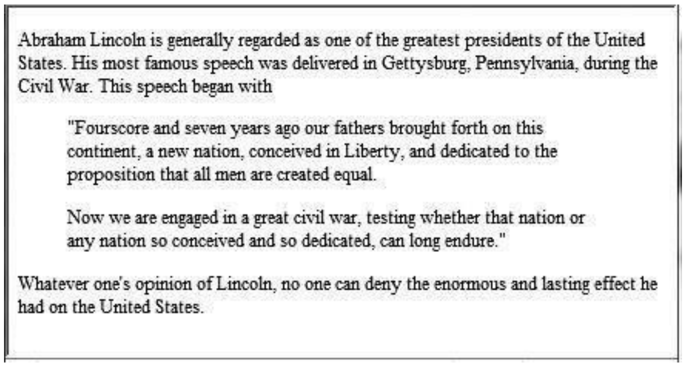
Figure 2.6 Display of headings.html

Block Quotations

Sometimes we want a block of text to be set off from the normal flow of text in a document. In many cases, such a block is a long quotation. The <blockquote> tag is designed for this situation. Browser designers determine how the content of <blockquote>can be made to look different from the surrounding text. In many cases, the block of text is indented, either on the left or right side or both. Another possibility is that the block is set in italics. Consider the following example document:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- blockquote.html
     An example to illustrate a blockquote
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Blockquotes </title>
  </head>
  <body>
    <p>
      Abraham Lincoln is generally regarded as one of the greatest
      presidents of the United States. His most famous speech was
      delivered in Gettysburg, Pennsylvania, during the Civil War.
      This speech began with
    </p>
    <blockquote>
      <p>
        "Fourscore and seven years ago our fathers brought forth on
        this continent, a new nation, conceived in Liberty, and
       dedicated to the proposition that all men are created equal.
      </p>
      <p>
        Now we are engaged in a great civil war, testing whether
        that nation or any nation so conceived and so dedicated,
        can long endure."
      </p>
    </blockquote>
    <p>
      Whatever one's opinion of Lincoln, no one can deny the
      enormous and lasting effect he had on the United States.
    </p>
  </body>
</html>
```

Figure 2.7 shows a browser display of blockquote.html.


Abraham Lincoln is generally regarded as one of the greatest presidents of the United States. His most famous speech was delivered in Gettysburg, Pennsylvania, during the Civil War. This speech began with

"Fourscore and seven years ago our fathers brought forth on this continent, a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal.

Now we are engaged in a great civil war, testing whether that nation or any nation so conceived and so dedicated, can long endure."

Whatever one's opinion of Lincoln, no one can deny the enormous and lasting effect he had on the United States.

Figure 2.7 Display of blockquote.html

Font Styles and Sizes

Early Web designers used a collection of tags to set font styles and sizes. For example, <i>specified italics and <b> specified bold. Since the advent of cascading style sheets (see Chapter 3, "Cascading Style Sheets"), the use of these tags has become passé. There are a few tags for fonts that are still in widespread use, called *content-based style tags*. These tags are called content based because they indicates the particular kind of text that appears in their content. Three of the most commonly used content-based tags are the emphasis tag, the strong tag, and the code tag.

The emphasis tag, <em>, specifies that its textual content is special and should be displayed in some way that indicates this distinctiveness. Most browsers use italics for such content.

The strong tag, <strong>is like the emphasis tag, but more so. Browsers often set the content of strong elements in bold.

The code tag, <code>, is used to specify a monospace font, usually for program code. For example, consider the following element:

```
<code> cost = quantity * price </code>
```

This markup would be displayed as shown in Figure 2.8.

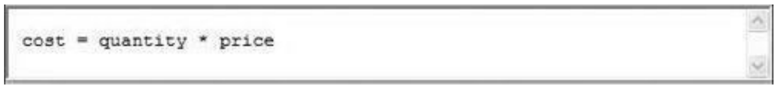

```
cost = quantity * price
```

Figure 2.8 The <code> element

Subscript and superscript characters can be specified by the <sub>and <sup>tags, respectively. These are not content-based tags. For example,

```
X<sub>2</sub><sup>3</sup> + y<sub>1</sub><sup>2</sup>
```

would be displayed as shown in Figure 2.9.

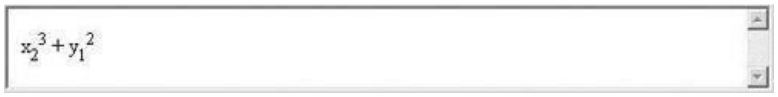

$x_2^3 + y_1^2$

Figure 2.9 The <sub> and <sup>elements

Character-modifying tags are not affected by <blockquote>, except when there is a conflict. For example, if the text content of <blockquote>is set in italics and a part of that text is made the content of an <em>tag, the <em>tag would have no effect.

XHTML tags are categorized as being either block or inline. The content of an *inline* tag appears on the current line. So, an inline tag does not implicitly include a line break. One exception is br, which is an inline tag, but its entire purpose is to insert a line break in the content. A *block* tag breaks the current line so that its content appears on a new line. The heading and block quote tags are block tags, whereas <em> and <strong>are inline tags. In XHTML, block tags cannot appear in the content of inline tags. Therefore, a block tag can never be nested directly in an inline tag. Also, inline tags and text cannot be directly nested in body or form elements. Only block tags can be so nested. That is why the example greet.htmlhas the text content of its body nested in a paragraph element.

Character Entities

XHTML provides a collection of special characters that are sometimes needed in a document but cannot be typed as themselves. In some cases, these characters are used in XHTML in some special way—for example, >, <, and &. In other cases, the characters do not appear on keyboards, such as the small raised circle that represents "degrees" in a reference to temperature. Finally, there is the nonbreaking space, which browsers regard as a hard space—they do not squeeze them out, as they do other multiple spaces. These special characters are defined as *entities,* which are codes for the characters. An entity in a document is replaced by its associated character by the browser. Table 2.1 lists some of the most commonly used entities.

Table 2.1 Some commonly used entities

Character	Entity	Meaning
&	`&amp;`	Ampersand
<	`&lt;`	Is less than
>	`&gt;`	Is greater than
"	`&quot;`	Double quote
'	`&apos;`	Single quote (apostrophe)
¼	`&frac14;`	One-quarter
½	`&frac12;`	One-half
¾	`&frac34;`	Three-quarters
°	`&deg;`	Degree
(space)	`&nbsp;`	Nonbreaking space

The parts of a document can be separated from each other, making the document easier to read, by placing horizontal lines between them. Such lines are called *horizontal rules*, and the block tag that creates them is <hr/>. The <hr/>tag causes a line break (ending the current line) and places a line across the screen. The browser chooses the thickness, length, and horizontal placement of the line. Typically, browsers display lines that are three pixels thick.

Note again the slash in the <hr />tag, indicating that this tag has no content and no closing tag.

- **The metaElement**

The metaelement is used to provide additional information about a document. The meta tag has no content; rather, all of the information provided is specified with attributes. The two attributes that are used to provide information are name and content. The user makes up a name as the value of the name attribute and specifies information through the content attribute. One commonly chosen name is keywords; the value of the contentattribute associated with the keywords are those which the author of a document believes characterizes his or her document. An example is

```
<meta name =
“keywords”
content =
“binary trees,
linked lists,
stacks” />
```

Web search engines use the information provided with the metaelement to categorize Web documents in their indices. So, if the author of a document

seeks widespread exposure for the document, one or more metaelements are included to ensure that it will be found by Web search engines. For example, if an entire book were published as a Web document, it might have the following meta elements:

```
<meta name = "Title" content = "Don Quixote" />
<meta name = "Author"  content = "Miguel Cervantes" />
<meta  name = "keywords"  content = "novel,
 Spanish literature, groundbreaking work" />
```

Images

The inclusion of images in a document can dramatically enhance its appearance (although images slow the document-download process considerably for clients who do not have high-speed Internet access). The image is stored in a file, which is specified by an XHTML request. The image is inserted into the display of the document by the browser.

Image Formats

The two most common methods of representing images are the Graphic Interchange Format (GIF, pronounced like the first syllable of *jiffy*) and the Joint Photographic Experts Group (JPEG, pronounced *jay-peg*) format. Most contemporary browsers can render images in either of these two formats. Files in both formats are compressed to reduce storage needs and provide faster transfer over the Internet.

The GIF format was developed by the CompuServe network service provider for the specific purpose of transmitting images. It uses 8-bit color representations

for pixels, allowing a pixel to have 256 different colors. If you are not familiar with color representations, this format may seem to be entirely adequate. However, with
the color displays on most contemporary computers, a huge number of colors can be displayed but cannot be represented in a GIF image. Files containing GIF images use the .gif(or .GIF) extension on their names. GIF images can be made to appear transparent.

The JPEG format uses 24-bit color representations for pixels, which allows JPEG images to include more than 16 million different colors. Files that store JPEG
images use the .jpg(or .JPGor .jpeg) extension on their names. The compression algorithm used by JPEG is better at shrinking an image than the one used by GIF. This compression process actually loses some of the color accuracy of the image, but because there is so much to begin with, the loss is rarely discernible by the user. Because of this powerful compression process, even though a JPEG image has much more color information than a GIF image of the same subject, the JPEG image can be smaller than the GIF image. Hence, JPEG images are often preferred to GIF images. The disadvantage of JPEG is that it does not support transparency.

A third image format is now gaining popularity: Portable Network Graphics (PNG, pronounced *ping*). PNG was designed in 1996 as a free replacement for GIF after the patent owner for GIF, Unisys, suggested that the company might begin charging royalties for documents that included GIF images.[8] Actually, PNG is a good replacement for both GIF and JPEG because it has the best characteristics of each (the possibility of transparency, as provided by GIF, and the same large number of colors as JPEG). One drawback of PNG is that, because its compression algorithm does not sacrifice picture clarity, its images require more space than comparable JPEG images.[9] Support for PNG in the earlier IE browsers was unacceptably poor, which kept many developers from using PNG. However, IE8 has adequate support for it. Information on PNG can be found at www.w3.org/Graphics/PNG.

The <img />Tag

The image tag, <img />, which is an inline tag, specifies an image that is to appear in a document. In its simplest form, the image tag includes two attributes: src, which specifies the file containing the image; and alt, which specifies text to be displayed when it is not possible to display the image. If the file is in the same directory as the XHTML file of the document, the value of srcis just the image's file name. In many cases, image files are stored in a subdirectory of the directory where the XHTML files are stored. For example, the image files might be stored in a subdirectory named images. If the image file's name is stars.jpgand the image file is stored in the imagessubdirectory, the value of srcwould be as follows:

```
"images/stars.jpg"
```

Some seriously aged browsers are not capable of displaying images. When such a browser finds an <img />tag, it simply ignores its content, possibly leaving the user confused by the text in the neighborhood of where the image was supposed to be. Also, graphical browsers, which *are* capable of displaying images, may have image downloading disabled by the browser user. This is done when the Internet connection is slow and the user chooses not to wait for images to download. It is also done by visually impaired users. In any case, it is helpful to have some text displayed in place of the ignored image. For these reasons, the alt attribute is required by XHTML. The value of altis displayed whenever the browser cannot or has been instructed not to display the image.

Two optional attributes of img—width and height—can be included to specify (in pixels) the size of the rectangle for the image. These attributes can be used to scale the size of the image (i.e., to make it larger or smaller). Care must be taken to ensure that the image is not distorted in the resizing. For example, if the image is square, the widthand height attribute values must be kept equal when they are changed.

The following is an example of an image element:

```
<img src = "c210.jpg" alt = "Picture of a Cessna 210"  />
```

The following example extends the airplane ad document to include information about a specific airplane and its image:

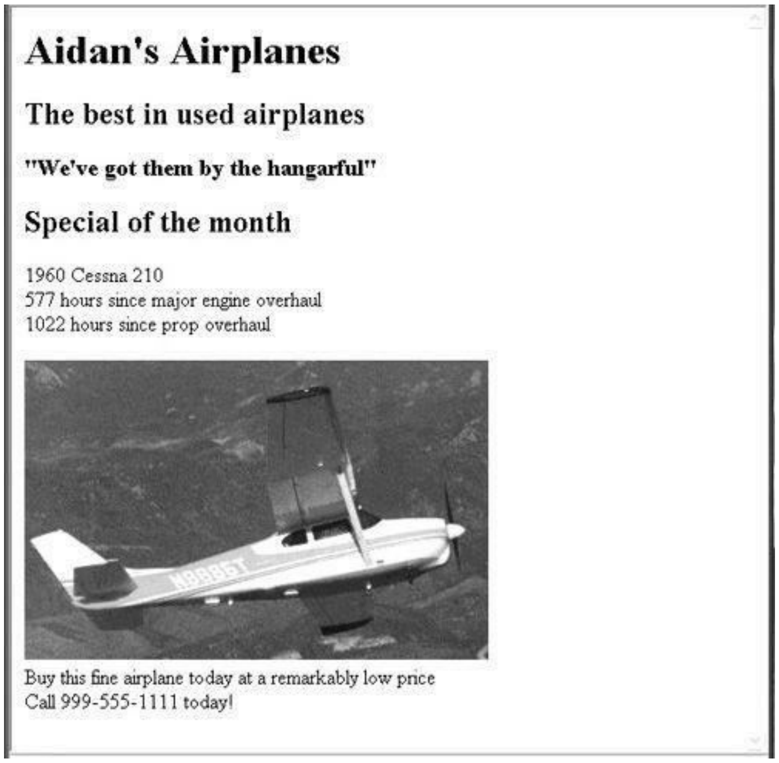
```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- image.html
     An example to illustrate an image
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Images </title>
  </head>
  <body>
    <h1> Aidan's Airplanes </h1>
    <h2> The best in used airplanes </h2>
    <h3> "We've got them by the hangarful" </h3>
    <h2> Special of the month </h2>
    <p>
      1960 Cessna 210 <br />
      577 hours since major engine overhaul<br />
      1022 hours since prop overhaul <br /><br />
      <img src = "c210new.jpg"  alt = "Picture of a Cessna 210" />
      <br />
      Buy this fine airplane today at a remarkably low price
      <br />
      Call 999-555-1111 today!
    </p>
  </body>
</html>
```

Figure 2.10 shows a browser display of image.html.

There is much more to the <img />tag than we have led you to believe. In fact, the <img />tag can include up to 30 different attributes. For descriptions of the rest, visit http://www.w3.org/TR/html401/index/attributes.html.

XHTML Document Validation

The W3C provides a convenient Web-based way to validate XHTML documents against its standards. The URL of the service is http://validator.w3.org/file- upload.html. Figure 2.11 shows a browser display of file-upload.html.



Aidan's Airplanes

The best in used airplanes

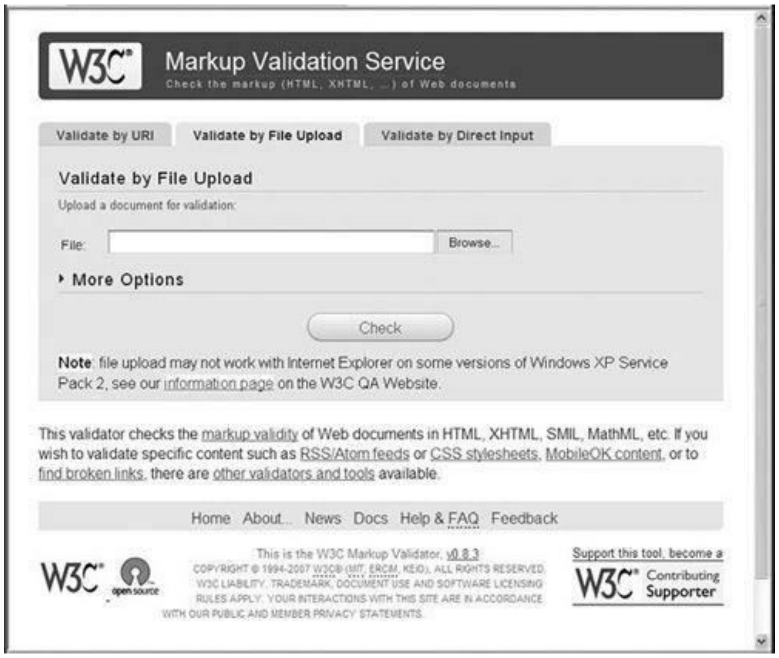
"We've got them by the hangarful"

Special of the month

1960 Cessna 210
577 hours since major engine overhaul
1022 hours since prop overhaul

Buy this fine airplane today at a remarkably low price
Call 999-555-1111 today!

Figure 2.10 Display of image.html



W3C
Markup Validation Service
Check the markup (HTML, XHTML, ...) of Web documents
Validate by URI
Validate by File Upload
Validate by Direct Input

Validate by File Upload

Upload a document for validation:

File:

Note: file upload may not work with Internet Explorer on some versions of Windows XP Service Pack 2, see our information page on the W3C QA Website.

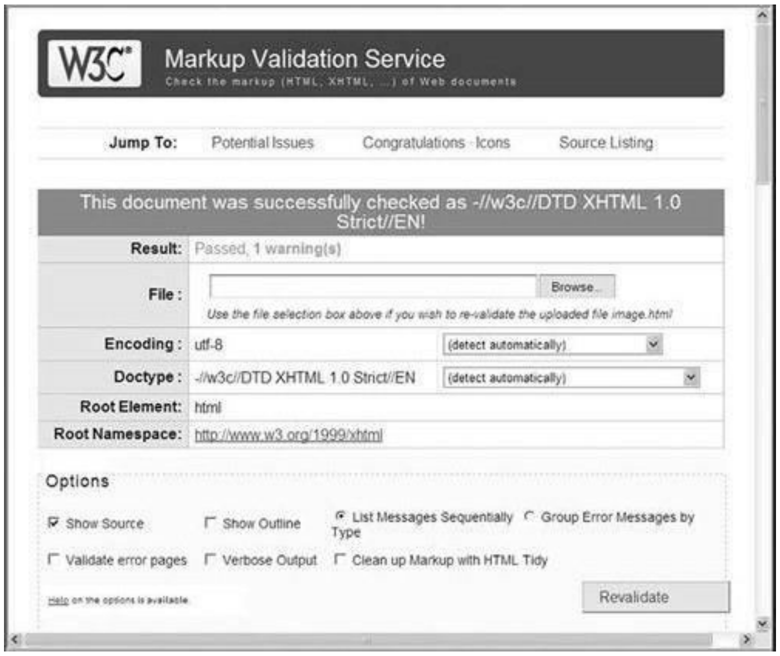
This validator checks the markup validity of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as RSS/Atom feeds or CSS stylesheets, MobileOK content, or to find broken links, there are other validators and tools available.

This is the W3C Markup Validator, v0.8.3
COPYRIGHT © 1994-2007 W3C® (MIT, ERCIM, KEIO), ALL RIGHTS RESERVED. W3C LIABILITY, TRADEMARK, DOCUMENT USE AND SOFTWARE LICENSING RULES APPLY. YOUR INTERACTIONS WITH THIS SITE ARE IN ACCORDANCE WITH OUR PUBLIC AND MEMBER PRIVACY STATEMENTS.

Figure 2.11 Display of file-upload.html, the W3C HTML validation document

The file name of the document to be validated is entered (including the path name) or found by browsing. We recommend that the *More Options* button be clicked and the *Show Source* checkbox be checked, because those actions cause the validation system to furnish a listing of the document in which the lines are numbered. These numbers are referenced in the report provided by the validation system. When the *Check* button is clicked, the specified file is uploaded to the validatorserver, where the validation system is run on it.

Figure 2.12 shows a browser display of the document returned by the validation system for our sample document image.html. Notice that we cut the source listing off in the figure, simply to prevent the figure from spanning more than one page.


Markup Validation Service
Check the markup (HTML, XHTML, ...) of Web documents

Jump To: Potential Issues Congratulations · Icons Source Listing

This document was successfully checked as -//w3c//DTD XHTML 1.0 Strict//EN!

Result:	Passed, 1 warning(s)
File :	Use the file selection box above if you wish to re-validate the uploaded file image.html
Encoding :	utf-8 (detect automatically)
Doctype :	-//w3c//DTD XHTML 1.0 Strict//EN (detect automatically)
Root Element:	html
Root Namespace:	http://www.w3.org/1999/xhtml

Options

[✓ Show Source] Show Outline [✓ List Messages Sequentially] Group Error Messages by Type
Validate error pages Verbose Output Clean up Markup with HTML Tidy

Help on the options is available.

Figure 2.12 HTML validation output for image.html (partial)

One of the most common errors made in crafting XHTML documents is putting text or elements where they do not belong. For example, putting text directly into a body element is invalid.

The XHTML validation system is a valuable tool for producing documents that adhere to W3C standards. The specific standard against which the document is checked is given in the DOCTYPEcommand. Because the DOCTYPE command in image.htmlspecifies the xhtml1-strict.dtdDTD, that document is checked against the XHTML 1.0 Strict standard.

Hypertext Links

A hypertext link in an XHTML document, which we simply call a *link* here, acts as a pointer to some particular place in some Web resource. That resource can be an XHTML document anywhere on the Web, or it may be the document currently being displayed. Without links, Web documents would be boring and tedious to read. There would be no convenient way for the browser user to get from one document to a logically related document. Most Web sites consist of many different documents, all logically linked. Therefore, links are essential to building an interesting Web site.

Links

A link that points to a different resource specifies the address of that resource. Such an address might be a file name, a directory path and a file name, or a complete URL. If a link points to a specific place in any document other than the beginning, that place somehow must be marked. Specifying such places is discussed in Section 2.6.2.

Links are specified in an attribute of an anchor tag (<a>), which is an inline tag. The anchor tag that specifies a link is called the *source* of that link. The document whose address is specified in a link is called the *target* of that link.

As is the case with many tags, the anchor tag can include many different attributes. However, for creating links, only one attribute is required: href(an acronym for *h*ypertext *ref*erence). The value assigned to hrefspecifies the target of the link. If the target is in another document in the same directory, the target is just the document's file name. If the target document is in some other directory, the UNIX pathname conventions are used. So, an XHTML file named c210data.html in a subdirectory of the directory in which the source XHTML file—named, say, airplanes—is specified in the href attribute value as "airplanes/c210data.html". This is the relative method of document addressing. Absolute file addresses could be used in which the entire path name for the file is given. However, relative links are easier to maintain, especially if a hierarchy of XHTML files must be moved. If the document is on some other machine (not the server providing the document that includes the link), obviously the complete URL must be used.

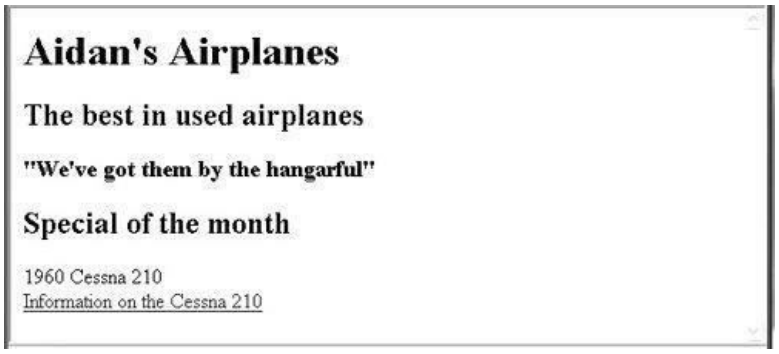
The content of an anchor tag, which becomes the clickable link the user sees, is restricted to text, line breaks, images, and headings. Although some browsers allow other nested tags, that is not standard XHTML and should not be used if you want your documents to be correctly displayed by all browsers. Links are usually implicitly rendered in a different color than that of the surrounding text. Sometimes they are also underlined. When the mouse cursor is placed over the anchor-tag content and the left mouse button is pressed, the link is taken by the browser. If the target is in a different document, that document is loaded and displayed, replacing the currently displayed document. If the target is in the current document, the document is scrolled by the browser to display the part of the document in which the target of the link is defined.

As an example of a link to the top of a different document, consider the following document, which adds a link to the document displayed in Figure 2.10:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- link.html
     An example to illustrate a link
     -->
```

```
      -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> A link </title>
  </head>
  <body>
    <h1> Aidan's Airplanes </h1>
    <h2> The best in used airplanes </h2>
    <h3> "We've got them by the hangarful" </h3>
    <h2> Special of the month </h2>
    <p>
      1960 Cessna 210 <br />
      <a href = "C210data.html"> Information on the Cessna 210 </a>
    </p>
  </body>
</html>
```

In this case, the target is a complete document that is stored in the same directory as the XHTML document. Figure 2.13 shows a browser display of
link.html. When the link shown in Figure 2.13 is clicked, the browser displays the screen shown in Figure 2.14.



Aidan's Airplanes

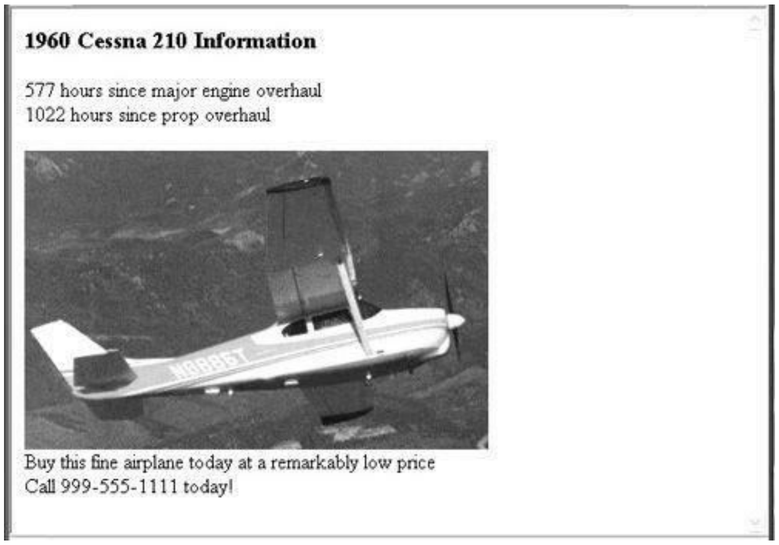
The best in used airplanes

"We've got them by the hangarful"

Special of the month

1960 Cessna 210
Information on the Cessna 210

Figure 2.13 Display of link.html



1960 Cessna 210 Information

577 hours since major engine overhaul
1022 hours since prop overhaul

Buy this fine airplane today at a remarkably low price
Call 999-555-1111 today!

Figure 2.14 Following the link from link.html

Links can include images in their content, in which case the browser displays the image together with the link:

```
<a href = "c210data.html" >
  <img src = "small-airplane.jpg"
       alt = "An image of a small airplane" />
    Information on the Cessna 210
</a>
```

An image itself can be an effective link (the content of the anchor element). For example, an image of a small house can be used for the link to the home page of a site. The content of an anchor element for such a link is just the imageelement.

Targets within Documents

If the target of a link is not at the beginning of a document, it must be some element within the document, in which case there must be some means of specifying it. The target element can include an idattribute, which can then be used to identify it in an hrefattribute. Consider the following example:

```
<h2 id = “avionics”> Avionics </h2>
```

Nearly all elements can include an idattribute. The value of an idattribute must be unique within the document.
If the target is in the same document as the link, the target is specified in the hrefattribute value by preceding the id value with a pound sign (#), as in the following example:

```
<a href = “#avionics”> What about avionics? </a>
```

When the What about avionics?link is taken, the browser moves the display so that the h2element whose idis avionics is at the top.
When the target is a part or fragment of another document, the name of the part is specified at the end of the URL, separated by a pound sign (#), as in this example:

```
<a href = “AIDAN1.html#avionics”> Avionics </a>
```

Using Links

One common use of links to parts of the same document is to provide a table of contents in which each entry has a link. This technique provides a convenient way for the user to get to the various parts of the document simply and quickly. Such a table of contents is implemented as a stylized list of links by using the list specification capabilities of XHTML, which are discussed in Section 2.7.
Links exemplify the true spirit of hypertext. The reader can click on links to learn more about a particular subtopic of interest and then return to the location of the link. Designing links requires some care because they can be annoying if the designer tries too hard to convince the user to take them. For example, making them stand out too much from the surrounding text can be distracting. A link should blend into the surrounding text as much as possible so that reading the document without clicking any of the links is easy and natural.

Lists

We frequently make and use lists in daily life—for example, to-do lists and grocery lists. Likewise, both printed and displayed information is littered with lists. XHTML provides simple and effective ways to specify lists in documents. The primary list types supported are those with which most people are already familiar: unordered lists such as grocery lists and ordered lists such as the assembly instructions for a new bookshelf. Definition lists can also be defined. The tags used to specify unordered, ordered, and definition lists are described in this section.

Unordered Lists

The <ul>tag, which is a block tag, creates an unordered list. Each item in a list is specified with an <li>tag (li is an acronym for *l*ist *i*tem). Any tags can appear in a list item, including nested lists. When displayed, each list item is implicitly preceded by a bullet.

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- unordered.html
     An example to illustrate an unordered list
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Unordered list </title>
  </head>
  <body>
    <h3> Some Common Single-Engine Aircraft </h3>
    <ul>
      <li> Cessna Skyhawk </li>
      <li> Beechcraft Bonanza </li>
      <li> Piper Cherokee </li>
    </ul>
  </body>
</html>
```

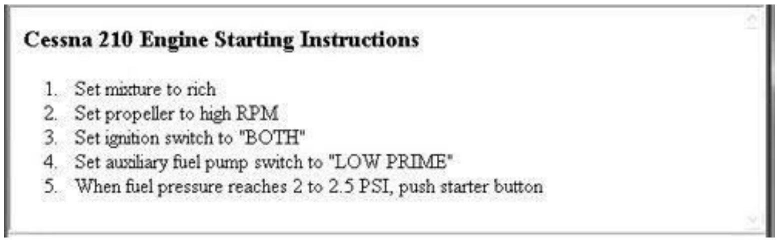
Ordered Lists

Ordered lists are lists in which the order of items is important. This orderedness of a list is shown in the display of the list by the implicit attachment of a sequential value to the beginning of each item. The default sequential values are Arabic numerals, beginning with 1.

An ordered list is created within the block tag <ol>. The items are specified and displayed just as are those in unordered lists, except that the items in an ordered list are preceded by sequential values instead of bullets. Consider the following example of an ordered list:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- ordered.html
     An example to illustrate an ordered list
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Ordered list </title>
  </head>
  <body>
    <h3> Cessna 210 Engine Starting Instructions </h3>
    <ol>
      <li> Set mixture to rich </li>
      <li> Set propeller to high RPM </li>
      <li> Set ignition switch to "BOTH" </li>
      <li> Set auxiliary fuel pump switch to "LOW PRIME" </li>
      <li> When fuel pressure reaches 2 to 2.5 PSI, push
           starter button
      </li>
    </ol>
  </body>
</html>
```

Figure 2.16 shows a browser display of ordered.html.


Cessna 210 Engine Starting Instructions

1. Set mixture to rich
2. Set propeller to high RPM
3. Set ignition switch to "BOTH"
4. Set auxiliary fuel pump switch to "LOW PRIME"
5. When fuel pressure reaches 2 to 2.5 PSI, push starter button

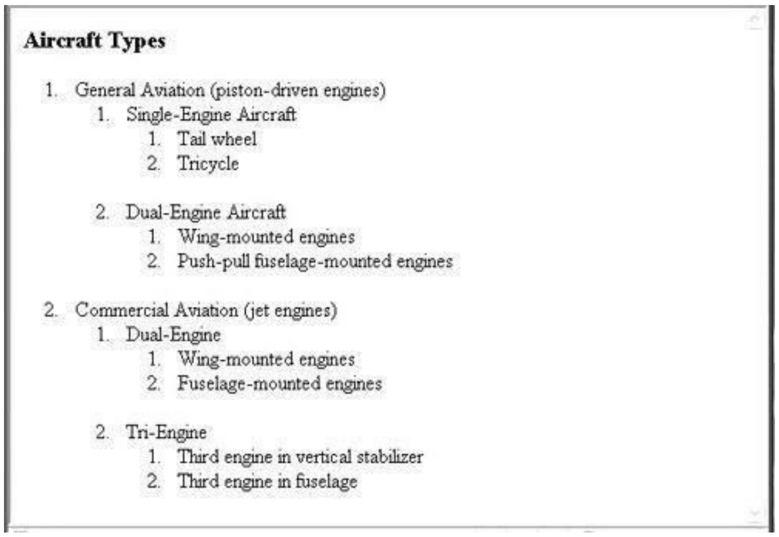
Figure 2.16 Display of ordered.html

As noted earlier, lists can be nested. However, a list cannot be directly nested; that is, an <ol>tag cannot immediately follow an <ol>tag. Rather, the nested list must be the content of an <li>element. The following example illustrates nested ordered lists:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- nested_lists.html
     An example to illustrate nested lists
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Nested lists </title>
  </head>
  <body>
```

```
                         <ol>
                           <li> Tail wheel </li>
                           <li> Tricycle </li>
                         </ol> <br />
                       </li>
                       <li> Dual-Engine Aircraft
                         <ol>
                           <li> Wing-mounted engines </li>
                           <li> Push-pull fuselage-mounted engines </li>
                         </ol>
                       </li>
                     </ol> <br />


  </li>
  <li> Commercial Aviation (jet engines)
    <ol>
      <li> Dual-Engine
        <ol>
          <li> Wing-mounted engines </li>
          <li> Fuselage-mounted engines </li>
        </ol> <br />
      </li>
      <li> Tri-Engine
        <ol>
          <li> Third engine in vertical stabilizer </li>
          <li> Third engine in fuselage </li>
        </ol>
          </li>
        </ol>
      </li>
    </ol>
  </body>
</html>
```

Figure 2.17 shows a browser display of nested_lists.html.



Aircraft Types

1. General Aviation (piston-driven engines)
   1. Single-Engine Aircraft
      1. Tail wheel
      2. Tricycle

   2. Dual-Engine Aircraft
      1. Wing-mounted engines
      2. Push-pull fuselage-mounted engines

2. Commercial Aviation (jet engines)
   1. Dual-Engine
      1. Wing-mounted engines
      2. Fuselage-mounted engines

   2. Tri-Engine
      1. Third engine in vertical stabilizer
      2. Third engine in fuselage

Figure 2.17 Display of nested_lists.html

One problem with the nested lists shown in Figure 2.17 is that all three levels use the same sequence values. Chapter 3 describes how style sheets can be used to specify different kinds of sequence values for different lists.

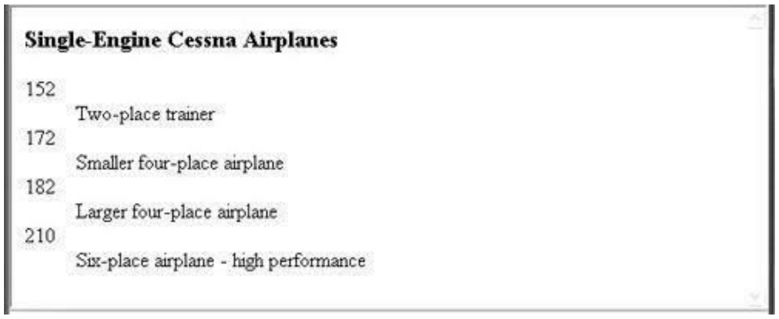
The nested_lists.htmlexample uses nested ordered lists. There are no restrictions on list nesting, provided that the nesting is not direct. For example, ordered lists can be nested in unordered lists and vice versa.

Definition Lists

As the name implies, definition lists are used to specify lists of terms and their definitions, as in glossaries. A definition list is given as the content of a <dl>tag, which is a block tag. Each term to be defined in the definition list is given as the content of a <dt>tag. The definitions themselves are specified as the content of <dd>tags. The defined terms of a definition list are usually displayed in the left margin; the definitions are usually shown indented on the line or lines following the term, as in the following example:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- definition.html
     An example to illustrate definition lists
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Definition lists </title>
  </head>
  <body>
    <h3> Single-Engine Cessna Airplanes </h3>
    <dl>
      <dt> 152 </dt>
      <dd> Two-place trainer </dd>
      <dt> 172 </dt>
      <dd> Smaller four-place airplane </dd>
      <dt> 182 </dt>
      <dd> Larger four-place airplane </dd>
      <dt> 210 </dt>
      <dd> Six-place airplane - high performance </dd>
    </dl>
  </body>
</html>
```

Figure 2.18 shows a browser display of definition.html.



Single-Engine Cessna Airplanes

152
    Two-place trainer
172
    Smaller four-place airplane
182
    Larger four-place airplane
210
    Six-place airplane - high performance

Figure 2.18 Display of definition.html

Tables

Tables are common fixtures in printed documents, books, and, of course, Web documents. Tables provide a highly effective way of presenting many kinds of information.

A table is a matrix of cells. The cells in the top row often contain column labels, those in the leftmost column often contain row labels, and most of the rest of the cells contain the data of the table. The content of a cell can be almost any document element, including text, a heading, a horizontal rule, an image, and a nested table.

Basic Table Tags

A table is specified as the content of the block tag <table>. There are two kinds of lines in tables: the line around the outside of the whole table is called the *border*; the lines that separate the cells from each other are called *rules*. A table that does not include the borderattribute will be a matrix of cells with neither a border nor rules. The browser has default widths for table borders and rules, which are used if the borderattribute is assigned the value "border."Otherwise, a number can be given as border's value, which specifies the border width in pixels. For example, border = "3"specifies a border 3 pixels wide. A bordervalue of "0" specifies no border and no rules. The rules are set at 1 pixel when any nonzero border value is specified. All table borders are beveled to give a three- dimensional appearance, although this is ineffective when narrow border widths are used. The border attribute is the most common attribute for the <table>tag.

In most cases, a displayed table is preceded by a title, given as the content of a <caption>tag, which can immediately follow the opening <table>tag. The cells of a table are specified one row at a time. Each row of a table is specified with a row tag, <tr>. Within each row, the row label is specified by the table heading tag, <th>. Although the <th>tag has *heading* in its name, we call these tags *labels* to avoid confusion with headings created with the <h*x*>tags. Each data cell of a row is specified with a table data tag, <td>. The first row of a table usually has the table's column labels. For example, if a table has three data columns and their column labels are, respectively, Apple, Orange, and Screwdriver, the first row can be specified by the following:

```
<tr>
   <th> Apple </th>
```

```
    <th> Orange </th>
    <th> Screwdriver </th>
</tr>
```

Each data row of a table is specified with a heading tag and one data tag for each data column. For example, the first data row for our work-in-progress table might be as follows:

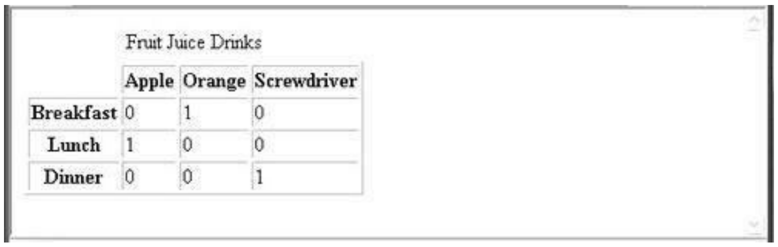
```
<tr>
    <th> Breakfast </th>
    <td> 0  </td>
    <td> 1  </td>
    <td> 0  </td>
</tr>
```

In tables that have both row and column labels, the upper-left corner cell is often empty. This empty cell is specified with a table header tag that includes no content (either <th></th> or just <th/>).

The following document describes the whole table:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- table.html
     An example of a simple table
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> A simple table </title>
  </head>
  <body>
    <table border = "border">
      <caption> Fruit Juice Drinks </caption>
      <tr>
        <th> </th>
        <th> Apple </th>
        <th> Orange </th>
        <th> Screwdriver </th>
      </tr>
      <tr>
        <th> Breakfast </th>
        <td> 0 </td>
        <td> 1 </td>
        <td> 0 </td>
      </tr>
      <tr>
        <th> Lunch </th>
        <td> 1 </td>
        <td> 0 </td>
        <td> 0 </td>
      </tr>
      <tr>
        <th> Dinner </th>
        <td> 0 </td>
        <td> 0 </td>
        <td> 1 </td>
      </tr>
    </table>
  </body>
</html>
```

Figure 2.19 shows a browser display of this table.



Fruit Juice Drinks

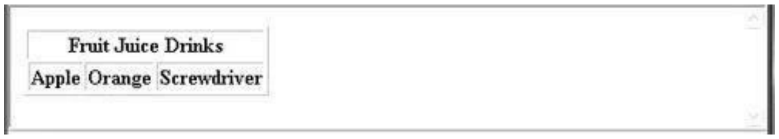
	Apple	Orange	Screwdriver
Breakfast	0	1	0
Lunch	1	0	0
Dinner	0	0	1

Figure 2.19 Display of table.html

The rowspanand colspanAttributes

In many cases, tables have multiple levels of row or column labels in which one label covers two or more secondary labels. For example, consider the display of a partial table shown in Figure 2.20. In this table, the upper-level label Fruit Juice Drinks spans the three lower-level label cells. Multiple-level labels can be specified with the rowspanand colspan attributes.

Figure 2.20 Two levels of column labels



Fruit Juice Drinks		
Apple	Orange	Screwdriver

The colspanattribute specification in a table header or table data tag tells the browser to make the cell as wide as the specified number of rows below it in the table. For the previous example, the following markup could be used:

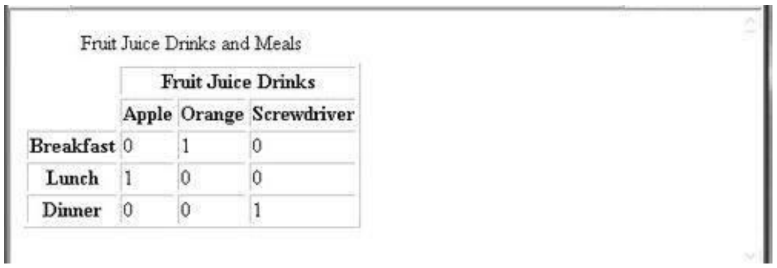
```
<tr>
  <th colspan = "3"> Fruit Juice Drinks </th>
</tr>
<tr>
  <th> Apple </th>
  <th> Orange </th>
  <th> Screwdriver </th>
</tr>
```

If there are fewer cells in the rows above or below the spanning cell than the colspanattribute specifies, the browser stretches the spanning cell over the number of cells that populate the column in the table.[10] The rowspanattribute of the table heading and table data tags does for rows what colspandoes for columns.

A table that has two levels of column labels and also has row labels must have an empty upper-left corner cell that spans both the multiple rows of column labels and the multiple columns. Such a cell is specified by including both rowspanand colspanattributes. Consider the following table specification, which is a minor modification of the previous table:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- cell_span.html
     An example to illustrate rowspan and colspan
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Rowspan and colspan </title>
  </head>
  <body>
    <table border = "border">
      <caption> Fruit Juice Drinks and Meals </caption>
      <tr>
        <td rowspan = "2"> </td>
        <th colspan = "3"> Fruit Juice Drinks </th>
      </tr>
      <tr>
        <th> Apple </th>
        <th> Orange </th>
        <th> Screwdriver </th>
      </tr>
      <tr>
        <th> Breakfast </th>
        <td> 0 </td>
        <td> 1 </td>
        <td> 0 </td>
      </tr>
      <tr>
        <th> Lunch </th>
        <td> 1 </td>
        <td> 0 </td>
        <td> 0 </td>
      </tr>
      <tr>
        <th> Dinner </th>
        <td> 0 </td>
        <td> 0 </td>
        <td> 1 </td>
      </tr>
    </table>
  </body>
</html>
```

Figure 2.21 shows a browser display of cell_span.html.


Fruit Juice Drinks and Meals

	Fruit Juice Drinks		
	Apple	**Orange**	**Screwdriver**
Breakfast	0	1	0
Lunch	1	0	0
Dinner	0	0	1

Figure 2.21 Display of cell_span.html: multiple-labeled columns and labeled rows

The alignand valignAttributes

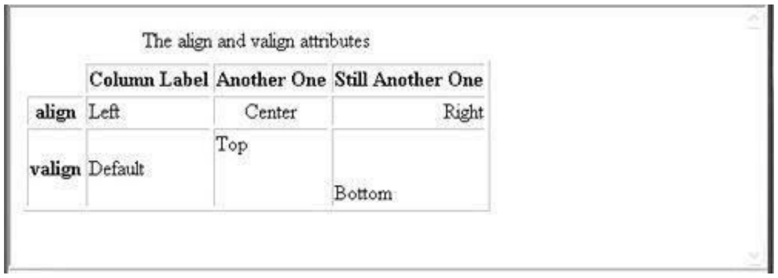
The placement of the content within a table cell can be specified with the alignand valignattributes in the <tr>, <th>, and <td>tags. The alignattribute has the possible values left, right, and center, with the obvious meanings for horizontal placement of the content within a cell. The default alignment for th cells is center; for td cells, it is left. If align is specified in a <tr> tag, it applies to all of the cells in the row. If it is included in a <th> or <td> tag, it applies only to that cell.

The valignattribute of the <th>and <td>tags has the possible values topand bottom. The default vertical alignment for both headings and data is
center. Because valignapplies only to a single cell, there is never any point in specifying center.

The following example illustrates the alignand valignattributes:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- cell_align.html
     An example to illustrate align and valign
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Alignment in cells </title>
  </head>
  <body>
    <table border = "border">
      <caption> The align and valign attributes </caption>
      <tr align = "center">
        <th> </th>
        <th> Column Label </th>
        <th> Another One </th>
        <th> Still Another One </th>
      </tr>
      <tr>
        <th> align </th>
        <td align = "left"> Left </td>
        <td align = "center"> Center </td>
        <td align = "right"> Right </td>
      </tr>
      <tr>
        <th> <br /> valign <br /> <br /> </th>
        <td> Default </td>
        <td valign = "top"> Top </td>
        <td valign = "bottom"> Bottom </td>
      </tr>
    </table>
  </body>
</html>
```

Figure 2.22 shows a browser display of cell_align.html.



The align and valign attributes

	Column Label	Another One	Still Another One
align	Left	Center	Right
valign	Default	Top	Bottom

Figure 2.22 Display of cell_align.html: the align and valign attributes

The cellpaddingand cellspacingAttributes

The table tag has two attributes that can respectively be used to specify the spacing between the content of a table cell and the cell's edge and the spacing between adjacent cells. The cellpaddingattribute is used to specify the spacing between the content of a cell and the inner walls of the cell—often, to prevent text in a cell from being too close to the edge of the cell. The cellspacingattribute is used to specify the distance between cells in a table.

The following document, space_pad.html, illustrates the cellpaddingand cellspacingattributes:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- space_pad.html
     An example that illustrates the cellspacing and
     cellpadding table attributes
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Cell spacing and cell padding </title>
  </head>
  <body>
    <b>Table 1 (space = 10, pad = 30) </b><br /><br />
    <table border = "5"  cellspacing = "10"  cellpadding = "30">
      <tr>
        <td> Small spacing, </td>
        <td> large padding </td>
      </tr>
    </table>
    <br /><br /><br /><br />
    <b>Table 2 (space = 30, pad = 10) </b><br /><br />
    <table border = "5"  cellspacing = "30"  cellpadding = "10">
      <tr>
        <td> Large spacing, </td>
        <td> small padding </td>
      </tr>
    </table>
  </body>
</html>
```

Figure 2.23 shows a browser display of space_pad.html.

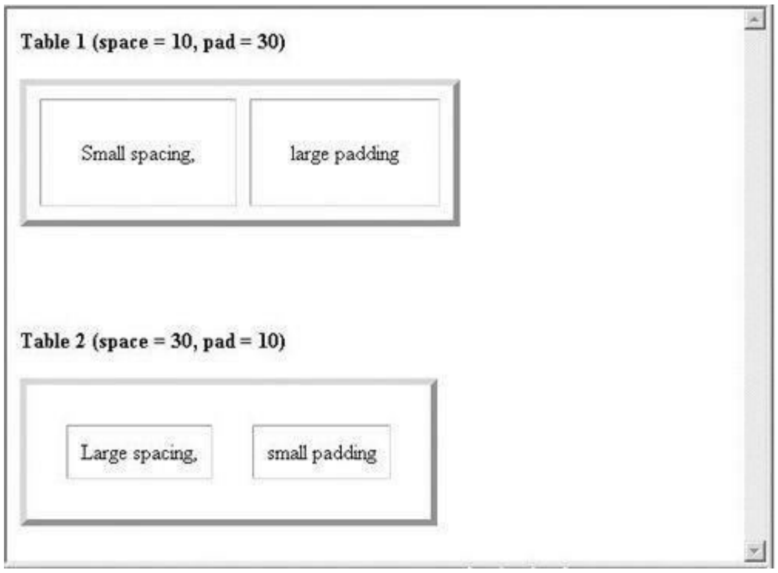


Table 1 (space = 10, pad = 30)

Small spacing,	large padding

Table 2 (space = 30, pad = 10)

Large spacing,	small padding

Figure 2.23 Display of space_pad.html

Table Sections

Tables naturally occur in two and sometimes three parts: header, body, and footer. (Not all tables have a natural footer.) These three parts can be respectively denoted in XHTML with the thead, tbody, and tfoot elements. The header includes the column labels, regardless of the number of levels in those labels. The body includes the data of the table, including the row labels. The footer, when it appears, sometimes has the column labels repeated after the body. In some tables, the footer contains totals for the columns of data above. A table can have multiple body sections, in which case the browser may delimit them with horizontal lines that are thicker than the rule lines within a body section.

Forms

The most common way for a user to communicate information from a Web browser to the server is through a form. Modeled on the paper forms that people frequently are required to fill out, forms can be described in XHTML and displayed by the browser. XHTML provides tags to generate the commonly used objects on a screen form. These objects are called *controls* or *widgets*. There are controls for single-line and multiple-line text collection, checkboxes, radio buttons, and menus, among others. All control tags are inline tags. Most controls are used to gather information from the user in the form of either text or button selections. Each control can have a value, usually given through user input. Together, the values of all of the controls (that have values) in a form are called the *form data*. Every form requires a *Submit* button (see Section 2.9.5). When the user clicks the *Submit* button, the form data is encoded and sent to the Web server for processing. Form processing is discussed in several subsequent chapters (Chapters 9, 11, and 12).

The <form>Tag

All of the controls of a form appear in the content of a <form> tag. A block tag, <form>, can have several different attributes, only one of which, action, is required. The actionattribute specifies the URL of the application on the Web server that is to be called when the user clicks the *Submit* button. In this chapter, our examples of form elements will not have corresponding application programs, so the value of their actionattributes will be the empty string ("").

The methodattribute of <form>specifies one of the two techniques, getor post, used to pass the form data to the server. The default is get, so if no methodattribute is given in the <form>tag, getwill be used. The alternative technique is post. In both techniques, the form data is coded into a text string when the user clicks the *Submit* button.

When the getmethod is used, the browser attaches the query string to the URL of the HTTP request, so the form data is transmitted to the server together with the URL. The browser inserts a question mark at the end of the actual URL just before the first character of the query string so that the server can easily find the beginning of the query string. The get method can also be used to pass parameters to the server when forms are not involved. (This cannot be done with post.) One disadvantage of the getmethod is that some servers place a limit on the length of the URL string and truncate any characters past the limit. So, if the form has more than a few controls, getis not a good choice.

When the postmethod is used, the query string is passed by some other method to the form-processing program. There is no length limitation for the query string with the postmethod, so, obviously, it is the better choice when there are more than a few controls in the form. There are also some security concerns with getthat are not a problem with post.

The <input>Tag

Many of the commonly used controls are specified with the inline tag <input>, including those for text, passwords, checkboxes, radio buttons, and the action buttons *Reset*, *Submit*, and *plain*. The text, password, checkboxes, and radio controls are discussed in this section. The action buttons are discussed in Section 2.9.5.

The one attribute of <input> that is required for all of the controls discussed in this section is type, which specifies the particular kind of control. The control's kind is its type name, such as checkbox. All of the previously listed controls except *Reset* and *Submit* also require a nameattribute, which becomes the name of the value of the control within the form data. The controls for checkboxes and radio buttons require a valueattribute, which initializes the value of the control. The values of these controls are placed in the form data that is sent to the server when the *Submit* button is clicked.

A text control, which we usually refer to as a text box, creates a horizontal box into which the user can type text. Text boxes are often used to gather information from the user, such as the user's name and address. The default size of a text box is often 20 characters. Because the default size can vary among browsers, it is a good idea to include a size on each text box. This is done with the sizeattribute of <input>. If the user types more characters than will fit in the box, the box is scrolled. If you do not want the box to be scrolled, you can include the maxlengthattribute to specify the maximum number of characters that the browser will accept in the box. Any additional characters are ignored. As an example of a text box, consider the following:

```
<form action = "">
  <p>
    <input type = "text"  name = "Name"  size = "25" />
  </p>
</form>
```

Suppose the user typed the following line:

Alfred Paul von Frickenburger

The text box would collect the whole string, but the string would be scrolled to the right, leaving the following shown in the box:

ed Paul von Frickenburger

The left end of the line would be part of the value of Name, even though it does not appear in the box. The ends of the line can be viewed in the box by moving the cursor off the ends of the box.

Notice that controls cannot appear directly in the form content—they must be placed in some block container, such as a paragraph. This is because <input>is an inline tag.

Now consider a similar text box that includes a maxlengthattribute:

```
<form action = "">
  <p>
    <input type = "text"  name = "Name"  size = "25"
           maxlength = "25" />
  </p>
</form>
```

If the user typed the same name as in the previous example, the resulting value of the Nametext box would be as follows:

Alfred Paul von Frickenbu

No matter what was typed after the uin that person's last name, the value of Namewould be as shown.

If the contents of a text box should not be displayed when they are entered by the user, a password control can be used as follows:

```
<input type = "password"  name = "myPassword"
       size = "10" maxlength = "10" />
```

In this case, regardless of what characters are typed into the password control, only bullets or asterisks are displayed by the browser.

There are no restrictions on the characters that can be typed into a text box. So, the string "?!34,:" could be entered into a text box meant for names. Therefore, the entered contents of text boxes nearly always must be validated, either on the browser or on the server to which the form data is passed for processing, or on both.

Text boxes, as well as most other control elements, should be labeled. Labeling could be done simply by inserting text into the appropriate places in the form:

```
Phone: <input type = "text" name = "phone" />
```

This markup effectively labels the text box, but there are several ways the labeling could be better. For one thing, there is no connection between the label and the control. Therefore, they could become separated in maintenance changes to the document. A control and its label can be connected by putting the control and its label in the content of a label element, as in the following element:

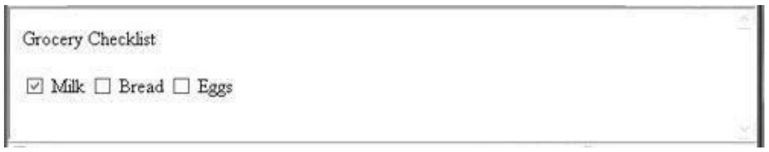
```
<label> Phone: <input type = "text" name = "phone" />
</label>
```

Now the text box and its label are encapsulated together. There are several other benefits of this approach to labeling controls. First, browsers often render the text content of a label element differently to make it stand out. Second, if the text content of a label element is selected, the cursor is implicitly moved to the control in the content of the label. This feature is an aid to new Web users. Third, the text content of a label element can be rendered by a speech synthesizer on the client machine when the content of the label element is selected. This feature can be a great aid to a user with a visual disability.

Checkbox and radio controls are used to collect multiple-choice input from the user. A checkbox control is a single button that is either on or off (checked or not). If a checkbox button is on, the value associated with the name of the button is the string assigned to its valueattribute. A checkbox button does not contribute to the form data if it is off. Every checkbox button requires a nameattribute and a valueattribute in its <input>tag. For form processing on the server, the name identifies the button and the value is its value (if the button is checked). The attribute checked, which is assigned the value checked, specifies that the checkbox button is initially on. In many cases, checkboxes appear in lists, with every one in the list having the same name. Checkbox elements should appear in label elements, for the same reasons that text boxes should. The following example illustrates a checkbox:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- checkbox.html
     An example to illustrate a checkbox
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Checkboxes </title>
  </head>
  <body>
    <p>
      Grocery Checklist
    </p>
    <form action = "">
      <p>
        <label> <input type = "checkbox"  name = "groceries"
               value = "milk"  checked = "checked" /> Milk </label>
        <label> <input type = "checkbox"  name = "groceries"
               value = "bread" /> Bread </label>
        <label> <input type = "checkbox"  name = "groceries"
               value = "eggs" /> Eggs </label>
      </p>
    </form>
  </body>
</html>
```

Figure 2.24 shows a browser display of checkbox.html.


Grocery Checklist

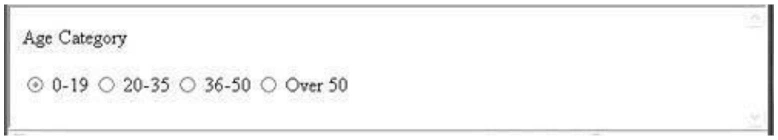
☑ Milk ☐ Bread ☐ Eggs

If the user does not turn on any of the checkbox buttons in our example, milkwill be the value for groceriesin the form data. If the milkcheckbox is left on and the eggscheckbox is also turned on by the user, the values of groceriesin the form data would be milkand eggs.

Radio buttons are closely related to checkbox buttons. The difference between a group of radio buttons and a group of checkboxes is that only one radio button can be on or pressed at any time. Every time a radio button is pressed, the button in the group that was previously on is turned off. Radio buttons are named after the mechanical push buttons on the radios of cars of the 1950s—when you pushed one button on such a radio, the previously pushed button was mechanically forced out. The typevalue for radio buttons is radio. All radio buttons in a group must have the nameattribute set in the <input> tag, and all radio buttons in a group must have the same name value. A radio button definition may specify which button is to be initially in the pressed, or on, state. This specification is indicated by including the checkedattribute, set to the value checked, in the <input>tag of the button's definition. If no radio button in a group is specified as being checked, the browser usually checksthe first button in the group. The following example illustrates radio buttons:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- radio.html
     An example to illustrate radio buttons
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Radio </title>
  </head>
  <body>
    <p>
      Age Category
    </p>
    <form action = "">
      <p>
        <label><input type = "radio"  name = "age"
               value = "under20" checked = "checked" />
               0-19 </label>
        <label><input type = "radio"  name = "age"
               value = "20-35" /> 20-35 </label>
        <label><input type = "radio"  name = "age"
               value = "36-50" /> 36-50 </label>
        <label><input type = "radio"  name = "age"
               value = "over50" /> Over 50 </label>
      </p>
    </form>
  </body>
</html>
```

Figure 2.25 shows a browser display of radio.html.


Age Category

◉ 0-19 ○ 20-35 ○ 36-50 ○ Over 50

Figure 2.25 Display of radio.html

The <select>Tag

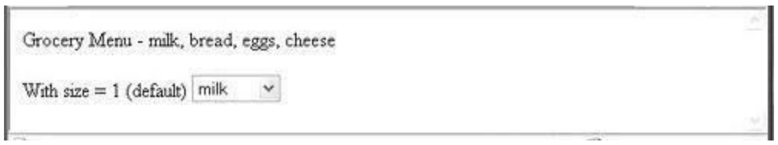
Checkboxes and radio buttons are effective methods for collecting multiple-choice data from a user. However, if the number of choices is large, the form becomes too long to display. In these cases, a menu should be used. A menu is specified with a <select> tag (rather than with the <input> tag). There are two kinds of menus: those in which only one menu item can be selected at a time (which are related to radio buttons) and those in which multiple menu items can be selected at a time (which are related to checkboxes). The default option is the one related to radio buttons. The other option can be specified by adding the multipleattribute, set to the value “multiple”. When only one menu item is selected, the value sent in the form data is the value of the nameattribute of the <select>tag and the chosen menu item. When multiple menu items are selected, the value for the menu in the form data includes all selected menu items. If no menu item is selected, no value for the menu is included in the form data. The nameattribute, of course, is required in the <select>tag.

The sizeattribute, specifying the number of menu items that are to be displayed for the user, can be included in the <select>tag. If no sizeattribute is
specified, the value 1is used. If the value for the sizeattribute is 1and multipleis not specified, just one menu item is displayed, with a downward scroll arrow. If the scroll arrow is clicked, the menu is displayed as a pop-up menu. If either multipleis specified or the sizeattribute is set to a number larger than 1, the menu is usually displayed as a scrolled list.

Each of the items in a menu is specified with an <option>tag, nested in the select element. The content of an <option>tag is the value of the menu item, which is just text. (No tags may be included.) The <option> tag can include the selected attribute, which specifies that the item is preselected. The value assigned to selectedis “selected”, which can be overridden by the user. The following document describes a menu with the default value (1) for size:

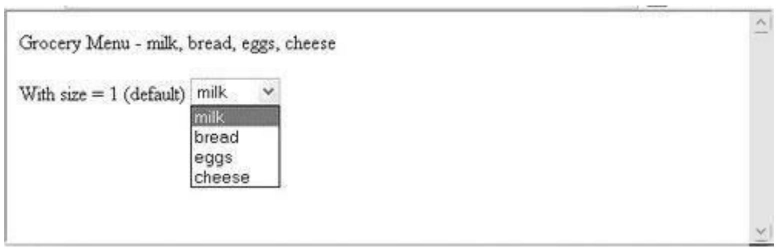
```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- menu.html
     An example to illustrate menus
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Menu </title>
  </head>
  <body>
    <p>
      Grocery Menu - milk, bread, eggs, cheese
    </p>
    <form action = "">
      <p>
        With size = 1 (the default)
        <select name = "groceries">
          <option> milk </option>
          <option> bread </option>
          <option> eggs </option>
          <option> cheese </option>
        </select>
      </p>
    </form>
  </body>
</html>
```

Figure 2.26 shows a browser display of menu.html. Figure 2.27 shows a browser display of menu.htmlafter clicking the scroll arrow. Figure 2.28 shows a browser display of menu.htmlafter modification to set sizeto “2.”


Grocery Menu - milk, bread, eggs, cheese

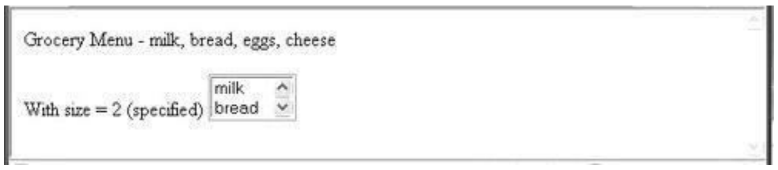
With size = 1 (default) milk

Figure 2.26 Display of menu.html (default size of 1)


Grocery Menu - milk, bread, eggs, cheese

With size = 1 (default) milk

Figure 2.27 Display of menu.html after the scroll arrow is clicked


Grocery Menu - milk, bread, eggs, cheese

With size = 2 (specified)

Figure 2.28 Display of menu.html with size set to 2

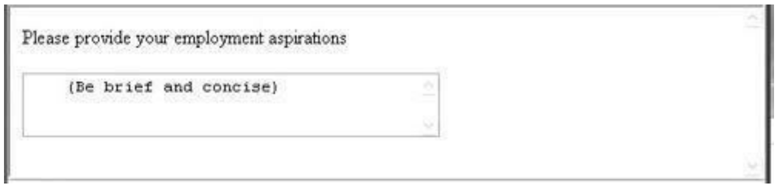
When the multipleattribute of the <select>tag is set, adjacent options can be chosen by dragging the mouse cursor over them while the left mouse button is held down. Nonadjacent options can be selected by clicking them while holding down the keyboard *Control* key.

The <textarea>Tag

In some situations, a multiline text area is needed. The <textarea>tag is used to create such a control. The text typed into the area created by <textarea>is
not limited in length, and there is implicit scrolling when needed, both vertically and horizontally. The default size of the visible part of the text in a text area is often quite small, so the rowsand colsattributes should usually be included and set to reasonable sizes. If some default text is to be included in the text area, it can be included as the content of the text area element. The following document describes a text area whose window is 40 columns wide and three lines tall:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- textarea.html
     An example to illustrate a textarea
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Textarea </title>
  </head>
  <body>
    <p>
      Please provide your employment aspirations
    </p>
    <form action = "handler">
      <p>
        <textarea name = "aspirations"  rows = "3"  cols = "40">
          (Be brief and concise)
        </textarea>
      </p>
    </form>
  </body>
</html>
```

Figure 2.29 shows a browser display of textarea.htmlafter some text has been typed into the area.



Please provide your employment aspirations

(Be brief and concise)

Figure 2.29 Display of textarea.html after some text entry

The Action Buttons

The *Reset* button clears all of the controls in the form to their initial states. The *Submit* button has two actions: First, the form data is encoded and sent to the server; second, the server is requested to execute the server-resident program specified in the actionattribute of the <form>tag. The purpose of such a server-resident program is to process the form data and return some response to the user. Every form requires a *Submit* button. The *Submit* and *Reset* buttons are created with the
<input>tag, as shown in the following example:

```
<form action = ““‘>
    <p>
        <input type = “submit” value = “Submit Form” />
        <input type = “reset” value = “Reset Form” />
<
/
p
```

>
<
/
f
o
r
m
>

Figure 2.30 shows a browser display of *Submit* and *Reset* buttons.

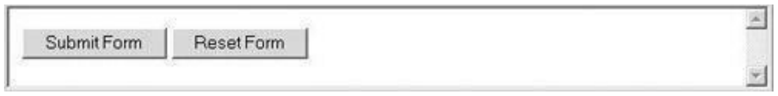


Figure 2.30 Submit and *Reset* buttons

A *plain* button has the type button. *Plain* buttons are used to choose an action.

Example of a Complete Form

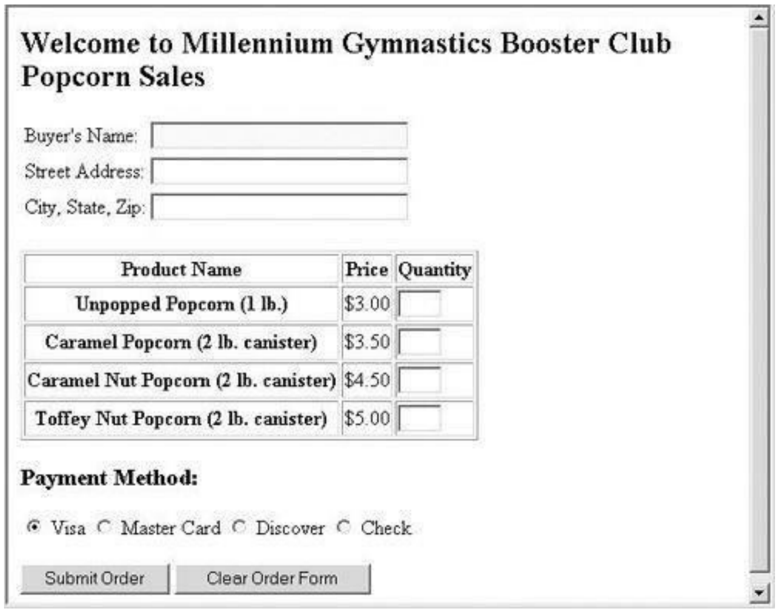
The document that follows describes a form for taking sales orders for popcorn. Three text boxes are used at the top of the form to collect the buyer's name and address. These boxes are placed in a borderless table to force them to align vertically. A second table is used to collect the actual order. Each row of this table names a product with the content of a <td> tag, displays the price with another <td>tag, and uses a text box with size set to 2 to collect the quantity ordered. The payment method is input by the user through one of four radio buttons.

Notice that none of the input controls in this document are embedded in label elements. This is because table elements cannot be labeled, except by using the row and column labels.

Tables present special problems for the visually impaired. The best solution is to use style sheets (see Chapter 3) instead of tables to lay out tabular information.

```
<?xml version = "1.0"  encoding = "utf-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- popcorn.html

     This describes a popcorn sales form document>
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Popcorn Sales Form </title>
  </head>
  <body>
    <h2> Welcome to Millennium Gymnastics Booster Club Popcorn
         Sales
    </h2>

<!-- The next line gives the address of the CGI program -->
    <form action = "">
<!-- A borderless table of text boxes for name and address -->
      <table>
        <tr>
          <td> Buyer's Name: </td>
          <td> <input type = "text"  name = "name"
                      size = "30" />
          </td>
        </tr>
        <tr>
          <td> Street Address: </td>
          <td> <input type = "text"  name = "street"
                      size = "30" />
          </td>
        </tr>
        <tr>
          <td> City, State, Zip: </td>
          <td> <input type = "text"  name = "city"
                      size = "30" />
          </td>
        </tr>
      </table>
    <p />

<!-- A bordered table for item orders -->
      <table border = "border">

<!-- First, the column headings -->
        <tr>
          <th> Product Name </th>
          <th> Price </th>
          <th> Quantity </th>
        </tr>

<!-- Now, the table data entries -->

        <tr>
          <td> Unpopped Popcorn (1 lb.) </td>
          <td> $3.00 </td>
          <td> <input type = "text"  name = "unpop"
                      size = "2" />
          </td>
        </tr>
        <tr>
          <td> Caramel Popcorn (2 lb. canister) </td>
          <td> $3.50 </td>
          <td> <input type = "text"  name = "caramel"
                      size = "2" />
          </td>
        </tr>
        <tr>
          <td> Caramel Nut Popcorn (2 lb. canister) </td>
          <td> $4.50 </td>
          <td> <input type = "text"  name = "caramelnut"
                      size = "2" />
          </td>
        </tr>
        <tr>
          <td> Toffey Nut Popcorn (2 lb. canister) </td>
          <td> $5.00 </td>
          <td> <input type = "text"  name = "toffeynut"
                      size = "2" />
          </td>
        </tr>

      </table>
      <p />
<!-- The radio buttons for the payment method -->
      <h3> Payment Method: </h3>
      <p>
```

```
        <label> <input type = "radio"  name = "payment"
                       value = "visa"  checked = "checked" />
                       Visa
        </label>
        <br />
        <label> <input type = "radio"  name = "payment"
                       value = "mc" /> Master Card
        </label>
        <br />
        <label> <input type = "radio"  name = "payment"
                       value = "discover" /> Discover
        </label>
        <br />
        <label> <input type = "radio"  name = "payment"
                       value = "check" /> Check
        </label>
        <br />
      </p>

<!-- The submit and reset buttons -->
      <p>
        <input type = "submit"  value = "Submit Order" />
        <input type = "reset"  value = "Clear Order Form" />
      </p>
    </form>
  </body>
</html>
```
Figure 2.31 shows a browser display of popcorn.html.



Welcome to Millennium Gymnastics Booster Club Popcorn Sales

Buyer's Name:

Street Address:

City, State, Zip:

Product Name	Price	Quantity
Unpopped Popcorn (1 lb.)	$3.00	
Caramel Popcorn (2 lb. canister)	$3.50	
Caramel Nut Popcorn (2 lb. canister)	$4.50	
Toffey Nut Popcorn (2 lb. canister)	$5.00	

Payment Method:

[✓ Visa] Master Card Discover Check

Figure 2.31 Display of popcorn.html

Chapter 9, "Introduction to PHP," has a PHP script for processing the data from the same form.

Syntactic Differences between HTML and XHTML

The discussion that follows points up some significant differences between the syntactic rules of HTML (or lack thereof) and those of XHTML.

Case sensitivity. In HTML, tag and attribute names are case insensitive, meaning that <FORM>, <form>, and <Form>are equivalent. In XHTML, all tag and attribute names must be all lowercase.

Closing tags. In HTML, closing tags may be omitted if the processing agent (usually a browser) can infer their presence. For example, in HTML, paragraph elements often do not have closing tags. The appearance of another opening paragraph tag is used to infer the closing tag on the previous paragraph. Thus, we have

```
<p>
During Spring, flowers are born. ...
<p>
During Fall, flowers die. ...
```

In XHTML, all elements must have closing tags. For elements that do not include content, in which the closing tag appears to serve no purpose, a slash can be included at the end of the opening tag as an abbreviation for the closing tag. For example, the following two lines are equivalent:

```
<input type = "text" name = "address" > </input>
```

and

```
<input type = "text" name = "address" />
```

Recall that some browsers can become confused if the slash at the end is not preceded by a space.

Quoted attribute values. In HTML, attribute values must be quoted only if there are embedded special characters or white-space characters. Numeric attribute values are rarely quoted in HTML. In XHTML, all attribute values must be double quoted, regardless of what characters are included in the value.

Explicit attribute values. In HTML, some attribute values are implicit; that is, they need not be explicitly stated. For example, if the borderattribute appears in a
<table>tag without a value, it specifies a default-width border on the table. Thus,

```
<table border>
```

specifies a border with the default width. This table tag is invalid in XHTML, in which such an attribute is assigned a string of the name of the attribute:

```
<table border = "border">
```

Other such attributes are checked, multiple, and selected.

id *and* name *attributes*. HTML markup often uses the nameattribute for elements. This attribute was deprecated for some elements in HTML 4.0, which added the idattribute to nearly all elements. In XHTML, the use of idis encouraged and the use of nameis discouraged. In fact, the nameattribute was removed for the anchor and map elements in XHTML 1.1. However, form elements must still use the nameattribute because it is employed in processing form data.

Element nesting. Although HTML has rules against improper nesting of elements, they are not enforced. Examples of nesting rules are (1) an anchor element cannot contain another anchor element, and a form element cannot contain another form element; (2) if an element appears inside another element, the closing tag of the inner element must appear before the closing tag of the outer element; (3) block elements cannot be nested in inline elements; (4) text cannot be directly nested in body or form elements; and (5) list elements cannot be directly nested in list elements. In XHTML, these nesting rules are strictly enforced.

All of the XHTML syntactic rules are checked by the W3C validation software.

Summary

XHTML was derived from SGML. Without the style sheets to be described in Chapter 3, XHTML is capable of specifying only the general layout of documents, with few presentation details. The current version of XHTML is XHTML 1.1, released in 2001.

The tags of XHTML specify how content is to be arranged in a display by a browser (or some other XHTML processor). Most tags consist of opening and closing tags to encapsulate the content that is to be affected by the tag. XHTML documents have two parts: the head and the body. The head describes some things about the document, but does not include any content. The body includes the content, and the tags and attributes which describe the layout of that content.

Line breaks in text are ignored by browsers. The browser fills lines in its display window and provides line breaks when needed. Line breaks can be specified with the <br />tag. Paragraph breaks can be specified with <p>. Headings can be created with the <h*x*>tags, where *x*can be any number from 1 to 6. The
<blockquote>tag is used to set off a section of text. The <sub>and <sup>tags are used to create subscripts and superscripts, respectively. Horizontal lines can
be specified with the <hr /> tag.

Images in GIF, JPEG, or PNG format can be inserted into documents from files where they are stored with the <img />tag. The altattribute of <img />
is used to present a message to the user when his or her browser is unable (or unwilling) to present the associated

image.

Links support hypertext by allowing a document to "point to" other documents, enabling the user to move easily from one document to another. The target of a link can be a different part of the current document or the top or some other part of a different document.

XHTML supports both unordered lists, using the <ul>tag, and ordered lists, using the <ol>tag. Both of these kinds of lists use the <li>tag to define list elements. The <dl>tag is used to describe definition lists. The <dt>and <dd>tags are used to specify the terms and their definitions, respectively.

Tables are easy to create with XHTML, through a collection of tags designed for that purpose. The <table>tag is used to create a table, <tr>to create table rows, <th>to create label cells, and <td>to create data cells in the table. The colspanand rowspanattributes, which can appear in both <th>and <td> tags, provide the means of creating multiple levels of column and row labels, respectively. The alignand valignattributes of the <tr>, <th>, and <td>tags are used to tell the browser exactly where to put data or label values within their respective table cells. The cellpadding and cellspacing attributes are respectively used to specify the distance between the content of a cell and its boundary and the distance between cells in a table.

XHTML forms are sections of documents that contain controls used to collect input from the user. The data specified in a form can be sent to a server-resident program in either of two methods: get or post. The most commonly used controls (text boxes, checkboxes, passwords, radio buttons, and the action buttons *Submit*, *Reset*, and *plain*) are specified with the <input>tag. The *Submit* button is used to indicate that the form data is to be sent to the server for processing. The *Reset* button is used to clear all of the controls in a form. The text box control is used to collect one line of input from the user. Checkboxes are one or more buttons used to select one or more elements of a list. Radio buttons are like checkboxes, except that, within a collection, only one button can be on at a time. A password is a text box whose content is never displayed by the browser.

Menus allow the user to select items from a list when the list is too long to use checkboxes or radio buttons. Menu controls are created with the <select>tag.

A text area control, which is created with the <textarea>tag, creates a multiple-line text-gathering box with implicit scrolling in both directions.

Cascading Style Sheets

- **Introduction**
- **Levels of Style Sheets**
- **Style Specification Formats**
- **Selector Forms**
- **Property Value Forms**
- **Font Properties**
- **List Properties**
- **Color**
- **Alignment of Text**
- **The Box Model**
- **Background Image**

This chapter introduces the concept of a style sheet and explains how style sheets fit into both the philosophy of XHTML and the structure of XHTML documents. An introduction to the three levels of style sheets and the format of style specifications follows. Then, the many varieties of property value forms are described. Next, specific properties for fonts and lists are introduced and illustrated. A discussion of the properties for specifying colors, background images, and text alignment follows. The box model of document elements is then discussed, along with borders and the associated padding and margin properties.

Introduction

We have said that XHTML is concerned primarily with content rather than the details of how that content is presented by browsers. That is not entirely true, however, even with the tags discussed in Chapter, "Introduction to XHTML." Some of those tags—for example, <code>—specify presentation details, or style. However, these presentation specifications can be more precisely and more consistently described with style sheets. Furthermore, many of the tags and attributes that can be used for describing presentation details have been deprecated in favor of style sheets.

Most XHTML tags have associated properties that store presentation information for browsers. Browsers use default values for these properties if the document does not specify values. For example, the <h2>tag has the font-sizeproperty, for which a browser could have the default value of 18 points. A document could specify that the font-sizeproperty for <h2>be set to 20 points, which would override the default value. The new value could apply to one occurrence of an <h2>element or all such occurrences in the document, depending on how the property value is set.

A style sheet is a syntactic mechanism for specifying style information. The idea of a style sheet is not new: Word processors and desktop publishing systems have long used style sheets to impose a particular style on documents. The first style-sheet specification for use in XHTML documents, dubbed Cascading Style Sheets (CSS1), was developed in 1996 by the W3C. In mid-1998, the second standard, CSS2, was released. CSS2 added many properties and property values to CSS1. It also extended presentation control to media other than Web browsers, such as printers. As a result of the incomplete implementation of (and perhaps a lack of interest in) parts of CSS2 by browser implementors, W3C decided to develop a new standard, CSS2.1, which would reflect the level of acceptance of CSS2. Both IE8 and FX3 fully support CSS2.1, which was at the "candidate recommendation" stage as of fall 2009. CSS3 has been in development since the late 1990s.

This chapter covers most of CSS2.1 All of the examples in the chapter work correctly for both IE8 and FX3 browsers.

Perhaps the most important benefit of style sheets is their capability of imposing consistency on the style of Web documents. For example, they allow the author to specify that all occurrences of a particular tag use the same presentation style.

XHTML style sheets are called *cascading* style sheets because they can be defined at three different levels to specify the style of a document. Lower level style sheets can override higher level style sheets, so the style of the content of a tag is determined, in effect, through a cascade of style-sheet applications.

Levels of Style Sheets

The three levels of style sheets, in order from lowest level to highest level, are *inline, document level*, and *external*. Inline style sheets apply to the content of a single XHTML element, document-level style sheets apply to the whole body of a document, and external style sheets can apply to the bodies of any number of documents. Inline style sheets have precedence over document style sheets, which have precedence over external style sheets. For example, if an external style sheet specifies a value for a particular property of a particular tag, that value is used until a different value is specified in either a document style sheet or an inline style sheet. Likewise, document style sheet property values can be overridden by different property values in an inline style sheet. In effect, the properties of a specific tag are those which result from a merge of all applicable style sheets, with lower-level style sheets having precedence in cases of conflicting specifications. There are other ways style specification conflicts can occur. These ways and their resolution are discussed in Section 3.13.

If no style sheet information is specified, the browser default property values are used.

As is the case with tags and tag attributes, a particular browser may not be capable of using the property values specified in a style sheet. For example, if the value of the font-sizeproperty of a paragraph is set to 18 points, but the browser can display the particular font being used only in sizes up to 16 points, the browser obviously cannot fulfill the property specification. In this case, the browser either would substitute an alternative value or would simply ignore the given font-size value and use its default font size.

Inline style specifications appear within the opening tag and apply only to the content of that tag. This fine-grain application of style defeats one of the primary advantages of style sheets—that of imposing a uniform style on the tags of at least one whole document. Another disadvantage of inline style sheets is that they result in style information, which is expressed in a language distinct from XHTML markup, being embedded in various places in documents. It is better to keep style specifications separate from XHTML markup. For this reason, among others, W3C deprecated inline style sheets in XHTML 1.1.[1] Therefore, inline style specifications should be used sparingly. This chapter discusses inline style sheets, but we follow our own advice and make little use of them in our examples.

Document-level style specifications appear in the document head section and apply to the entire body of the document. This is obviously an effective way to impose a uniform style on the presentation of all of the content of a document.

In many cases, it is desirable to have a style sheet apply to more than one document. That is the purpose of external style sheets, which are not part of any of the documents to which they apply. They are stored separately and are referenced in all documents that use them. External style sheets are written as text files with the MIME type text/css. They can be stored on any computer on the Web. The browser fetches external style sheets just as it fetches documents. The <link>tag is used to specify external style sheets. Within <link>, the relattribute is used to specify the relationship of the linked-to document to the document in which the link appears. The hrefattribute of <link>is used to specify the URL of the style sheet document, as in the following example:

```
<link rel = "stylesheet"  type = "text/css"
      href = "http://www.cs.usc.edu/styles/wbook.css" />
```

The link to an external style sheet must appear in the head of the document. If the external style sheet resides on the Web server computer, only its path address must be given as the value of href. An example of an external style sheet appears in Section 3.6.

The @importdirective is an alternative way to use style specifications from other files. The form of this directive is

```
@import url(file name);
```

Notice that the file name is not quoted. There are two differences between linkand @import: (1) @importcan appear only at the beginning of the content of a styleelement,[2] and (2) the imported file can contain markup, as well as style, rules. In fact, sometimes the imported file contains other @importdirectives, along with some style rules.

External style sheets can be validated with the service provided at http://jigsaw.w3.org/css-validator/.

Style Specification Formats

The format of a style specification depends on the level of style sheet. Inline style specifications appear as values of the styleattribute of a tag,[3] the general form of which is as follows:

```
style = "property_1:value_1; property_2:value_2; ...;
         property_n:value_n;"
```

Although it is not required, it is recommended that the last property–value pair be followed by a semicolon.

Document style specifications appear as the content of a style element within the header of a document, although the format of the specification is quite different from that of inline style sheets. The general form of the content of a style element is as follows:[4]

```
<style type = "text/css">
  rule_list
</style>
```

The type attribute of the <style>tag tells the browser the type of style specification, which is always text/css. The type of style specification is necessary because there are other kinds of style sheets. For example, JavaScript, which can be embedded in an XHTML document, also provides style sheets that can appear in style elements.

Each style rule in a rule list has two parts: a selector, which indicates the tag or tags affected by the rule, and a list of property–value pairs. The list has the same form as the quoted list for inline style sheets, except that it is delimited by braces rather than double quotes. So, the form of a style rule is as follows:

```
selector {property_1:value_1; property_2:value_2; ...;
          property_n:value_n;}
```

If a property is given more than one value, those values usually are separated with spaces. For some properties, however, multiple values are separated with commas.

Like all other kinds of coding, complicated CSS rule lists should be documented with comments. Of course, XHTML comments cannot be used here, because CSS is not XHTML. Therefore, a different form of comment is needed. CSS comments are introduced with /*and terminated with */,[5] as in the following element:

```
<style type = "text/css">
  /* Styles for the initial paragraph */
  ...
  /* Styles for other paragraphs */
  ...
</style>
```

External style sheets have a form similar to that of document style sheets. The external file consists of a list of style rules. An example of an external style sheet appears in Section 3.6.

Selector Forms

The selector can have a variety of forms.

Simple Selector Forms

The simplest selector form is a single element name, such as h1. In this case, the property values in the rule apply to all occurrences of the named element. The selector could be a list of element names separated by commas, in which case the property values apply to all occurrences of all of the named elements. Consider the following examples, in which the property is font-sizeand the property value is a number of points:

```
h1 {font-size: 24pt;}
```

h2, h3 {font-size: 20pt;}

The first of these selector forms specifies that the text content of all h1elements must be set in a 24-point font size. The second specifies that the text content of all h2 and h3elements must be set in a 20-point font size.

Selectors can also specify that the style should apply only to elements in certain positions in the document. This is done by listing the element hierarchy in the selector, with only white space separating the element names. For example, the rule

form em {font-size: 14pt;}

applies its style only to the content of emphasis elements that are nested in a form element in the document. This is a *contextual* selector (sometimes called a *descendant* selector).

Class Selectors

Class selectors are used to allow different occurrences of the same tag to use different style specifications. A style class is defined in a style element by giving the style class a name, which is attached to the tag's name with a period. For example, if you want two paragraph styles in a document—say, normaland warning—you could define these two classes in the content of a <style>tag as follows:

p.normal {*property-value list*} p.warning
{*property-value list*}

Within the document body, the particular style class that you want is specified with the class attribute of the affected tag—in the preceding example, the paragraph tag. For example, you might have the following markup:

```
<p class = "normal">
A paragraph of text that we want to be presented in
'normal' presentation style
</p>
<p class = "warning">
A paragraph of text that is a warning to the reader, which
should be presented in an especially noticeable style
</p>
```

Generic Selectors

Sometimes it is convenient to have a class of style specifications that applies to the content of more than one kind of tag. This is done by using a generic class, which is defined without a tag name in its name. In place of the tag name, you use the name of the generic class, which must begin with a period, as in the following generic style class:

.sale {*property-value list*}

Now, in the body of a document, you could have the following markup:

```
<h3 class = "sale"> Weekend Sale </h3>
...
<p class = "sale">
...
</p>
```

idSelectors

An idselector allows the application of a style to one specific element. The general form of an idselector is as follows:[6]

#*specific-id* {*property-value list*}

As you would probably guess, the style specified in the idselector applies to the element with the specific id:

#section14 {font-size: 20}

specifies a font size of 20 points to the element

<h2 id = “section14”>1.4 Calico Cats </h2>

CSS includes still more selector forms. However, because of the lack of browser support for them, they are not discussed here.

Universal Selectors

The universal selector, denoted by an asterisk (*), applies its style to all elements in a document. For example,

```
* {color: red;}
```

makes all elements in the document red.
The universal selector is not often useful.

Pseudo Classes

Pseudo classes are styles that apply when something happens, rather than because the target element simply exists. In this section, we describe and illustrate two pseudo classes: hoverand focus.
Whereas the names of style classes and generic classes begin with a period, the names of pseudo classes begin with a colon. The style of the hoverpseudo class applies when its associated element has the mouse cursor over it. The style of the focuspseudo class applies when its associated element has focus.[7] The following document is illustrative:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- pseudo.html
     Illustrates the :hover and :focus pseudo classes.
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Pseudo Classes </title>
    <style type = "text/css">
      input:hover {background: pink; color: red;}
      input:focus {background: lightblue; color: blue;}
    </style>
  </head>
  <body>
    <form action = "">
      <p>
        <label>
          Your name:
          <input type = "text"  />
        </label>
      </p>
    </form>
  </body>
</html>
```

In pseudo.html, the content of an input element (a text box) turns red with a pink background when the mouse cursor is placed over its content. This happens, however, only when the text box does not have focus. If no text has been typed into the text box, the hover pseudo class has no effect. When the text box acquires focus, the text turns blue with a light blue background and stays like that until the left mouse button is clicked outside the box.

Property Value Forms

CSS includes 60 different properties in seven categories: fonts, lists, alignment of text, margins, colors, backgrounds, and borders. As you probably would guess, not all of these properties are discussed here. The complete details of all properties and property values can be found at the W3C Web site.
Property values can appear in a variety of forms. Keyword property values are used when there are only a few possible values and they are predefined—for example, large, medium, and small. Keyword values are not case sensitive, so Small, SmAlL, and SMALLare all the same as small.
Number values are used when no meaningful units can be attached to a numeric property value. A number value can be either an integer or a sequence of digits with a decimal point and can be preceded by a sign (+or -).
Length values are specified as number values that are followed immediately by a two-character abbreviation of a unit name. There can be no space between the number and the unit name. The possible unit names are px, for pixels; in, for inches; cm, for centimeters; mm, for millimeters; pt, for points (a point is 1/72 inch);

and pc, for picas, which are 12 points. Note that on a display, in, cm, mm, pt, and pc are approximate measures. Their actual values depend on the screen resolution. There are also two relative length values: em, which is the value of the current font size in pixels, and ex, which is the height of the letter *x*.

Percentage values are used to provide a measure that is relative to the previously used measure for a property value. Percentage values are numbers that are followed immediately by a percent sign (%). For example, if the font size were set to 75%, it would make the new current size for the font 75 percent of its previous value. Font size would stay at the new value until changed again. Percentage values can be signed. If preceded by a plus sign, the percentage is added to the previous value; if negative, the percentage is subtracted.

URL property values use a form that is slightly different from references to URLs in links. The actual URL, which can be either absolute or relative, is placed in parentheses and preceded by url, as in the following property:

```
url(tetons.jpg)
```

There can be no space between urland the left parenthesis.

Color property values can be specified as color names, as six-digit hexadecimal numbers, or in RGB form. RGB form is just the word rgb followed by a parenthesized list of three numbers that specify the levels of red, green, and blue, respectively. The RGB values can be given either as decimal numbers between 0 and 255 or as percentages. Hexadecimal numbers must be preceded with pound signs (#), as in #43AF00. For example, powder blue could be specified with

```
fuchsia
```

or

```
rgb(255, 0, 255)
```

or

```
#FF00FF
```

CSS also includes properties for counters and strings, but they are not covered here.

Some property values are inherited by elements nested in the element for which the values are specified. For example, the property background-coloris not inherited, but font-sizeis. Using a style sheet to set a value for an inheritable property for the <body>tag effectively sets it as a default property value for the whole document, as in

```
body {font-size: 16pt}
```

Unless overridden by a style sheet that applies to paragraph elements, every paragraph element in the body of this document would inherit the font size of 16 points.

Font Properties

The font properties are among the most commonly used of the style-sheet properties. Virtually all XHTML documents include text, which is often used in a variety of different situations. This creates a need for text in many different fonts, font styles, and sizes. The font properties allow us to specify these different forms.

Font Families

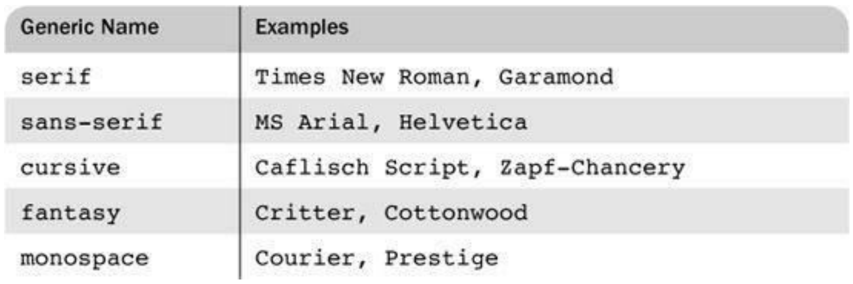
The font-familyproperty is used to specify a list of font names. The browser uses the first font in the list that it supports. For example, the property:

```
font-family: Arial, Helvetica, Futura
```

tells the browser to use Arial if it supports that font. If not, it will use Helvetica if it supports it. If the browser supports neither Arial nor Helvetica, it will use Futura if it can. If the browser does not support any of the specified fonts, it will use an alternative of its choosing.

A generic font can be specified as a font-familyvalue. The possible generic fonts and examples of each are shown in Table 3.1. Each browser has a font defined for each of these generic names. A good approach to specifying fonts is to use a generic font as the last font in the value of a font-familyproperty. For example, because Arial, Helvetica, and Futura are sans-serif fonts,[8] the previous example would be better as follows:

Table 3.1 Generic fonts



Generic Name	Examples
serif	Times New Roman, Garamond
sans-serif	MS Arial, Helvetica
cursive	Caflisch Script, Zapf-Chancery
fantasy	Critter, Cottonwood
monospace	Courier, Prestige

```
font-family: Arial, Helvetica, Futura, sans-serif
```

If a font name has more than one word, the whole name should be delimited by single quotes,[9] as in the following example:

```
font-family: 'Times New Roman'
```

In practice, the quotes may not be mandatory, but their use is recommended because they may be required in the future.

Font Sizes

The font-sizeproperty does what its name implies. For example, the following property specification sets the font size for text to 10 points:

```
font-size: 10pt
```

Many relative font-size values are defined, including xx-small, x-small, small, medium, large, x-large, and xx-large. In addition, smalleror largercan be specified. Furthermore, the value can be a percentage relative to the current font size.

On the one hand, using the relative font sizes has the disadvantage of failing to provide strict font size control. Different browsers can use different values for them. For example, small might mean 10 points on one browser and 8 points on another. On the other hand, using a specific font size has the risk that some browsers may not support that particular size, causing the document to appear different on different browsers.

Font Variants

The default value of the font-variantproperty is normal, which specifies the usual character font. This property can be set to small-capsto specify small capital characters. These characters are all uppercase, but the letters that are normally uppercase are somewhat larger than those that are normally lowercase.

Font Styles

The font-styleproperty is most commonly used to specify italic, as in

```
font-style: italic
```

An alternative to italicis oblique, but when displayed, the two are nearly identical, so obliqueis not a terribly useful font style. In fact, some browsers do not support it, so they display all oblique fonts in italic.

Font Weights

The font-weightproperty is used to specify the degree of boldness, as in

```
font-weight: bold
```

Besides bold, the values normal(the default), bolder, and lightercan be specified. The bolderand lightervalues are taken as relative to the current level of boldness. Specific numbers also can be given in multiples of 100from 100to 900, where 400is the same as normaland 700is the same as bold.

Font Shorthands

If more than one font property must be specified, the values can be stated in a list as the value of the fontproperty. The browser then has the responsibility for determining which properties to assign from the forms of the values. For example, theproperty

```
font: bold 14pt 'Times New Roman' Palatino
```

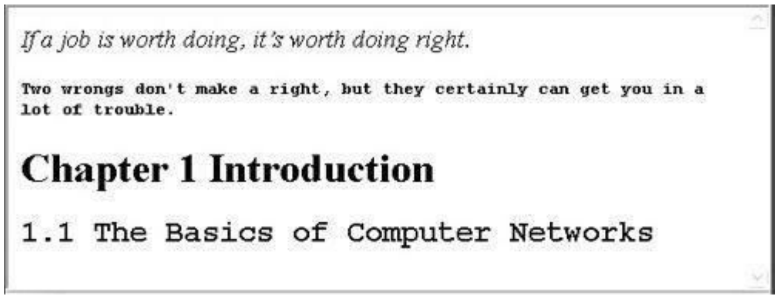
specifies that the font weight should be bold, the font size should be 14 points, and either Times New Roman or Palatino font should be used, with precedence given to Times New Roman.

The order in which the property values are given in a fontvalue list is important. The order must be as follows: The font names must be last, the font size must be second to last, and the font style, font variant, and font weight, when they are included, can be in any order but must precede the font size and font names. Only the font size and the font family are required in the fontvalue list.

The document fonts.htmlillustrates some aspects of style-sheet specification of the font properties in headings and paragraphs:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- fonts.html
     An example to illustrate font properties
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Font properties </title>
    <style type = "text/css">
      p.major {font-size: 14pt;
               font-style: italic;
               font-family: 'Times New Roman';
              }
      p.minor {font: 10pt bold 'Courier New';}
      h2 {font-family: 'Times New Roman';
          font-size: 24pt; font-weight: bold}
      h3 {font-family: 'Courier New'; font-size: 18pt}
    </style>
  </head>
  <body>
    <p class = "major">
      If a job is worth doing, it's worth doing right.
    </p>
    <p class = "minor">
      Two wrongs don't make a right, but they certainly
      can get you in a lot of trouble.
    </p>
    <h2> Chapter 1 Introduction </h2>
    <h3> 1.1 The Basics of Computer Networks </h3>
  </body>
</html>
```

Figure 3.1 shows a browser display of fonts.html.



If a job is worth doing, it's worth doing right.

Two wrongs don't make a right, but they certainly can get you in a lot of trouble.

Chapter 1 Introduction

1.1 The Basics of Computer Networks

Figure 3.1 Display of fonts.html

The following document, called fonts2.html, is a revision of fonts.htmlthat uses an external style sheet in place of the document style sheet used in fonts.html(the external style sheet, styles.css, follows the revised document):

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- fonts2.html
     An example to test external style sheets
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> External style sheets </title>
    <link rel = "stylesheet"  type = "text/css"
          href = "styles.css" />
  </head>
  <body>
    <p class = "major">
      If a job is worth doing, it's worth doing right.
    </p>
    <p class = "minor">
      Two wrongs don't make a right, but they certainly
      can get you in a lot of trouble.
    </p>
    <h2> Chapter 1 Introduction </h2>
    <h3> 1.1 The Basics of Computer Networks </h3>
  </body>
</html>
```

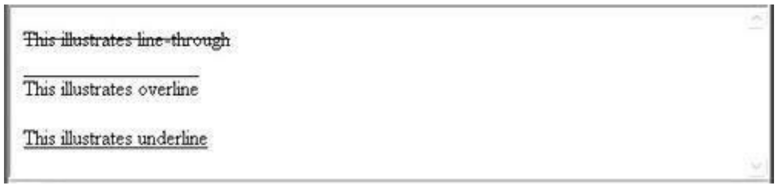
```
/* styles.css - an external style sheet
     for use with fonts2.html
   */
  p.major {font-size: 14pt;
           font-style: italic;
           font-family: 'Times New Roman';
        }
  p.minor {font: bold 10pt 'Courier New';}
  h2 {font-family: 'Times New Roman';
     font-size: 24pt; font-weight: bold}
  h3 {font-family: 'Courier New';
     font-size: 18pt}
```

Text Decoration

The text-decoration property is used to specify some special features of text. The available values are line-through, overline, underline, and none, which is the default. Many browsers implicitly underline links. The nonevalue can be used to avoid this underlining. Note that text-decorationis not inherited. The following document, decoration.html, illustrates the line-through, overline, and underlinevalues:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- decoration.html
     An example that illustrates several of the
     possible text decoration values
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Text decoration </title>
    <style type = "text/css">
      p.delete {text-decoration: line-through}
      p.cap {text-decoration: overline}
      p.attention {text-decoration: underline}
    </style>
  </head>
  <body>
    <p class = "delete">
      This illustrates line-through
    </p>
    <p class= "cap">
      This illustrates overline
    </p>
    <p class = "attention">
      This illustrates underline
    </p>
  </body>
</html>
```

Figure 3.2 shows a browser display of decoration.html.


~~This illustrates line-through~~

This illustrates overline

This illustrates underline

Figure 3.2 Display of decoration.html

The letter-spacing property controls the amount of space between characters in text. The possible values of letter-spacing are any length property values—for example, 3px.

List Properties

Two presentation details of lists can be specified in XHTML documents: the shape of the bullets that precede the items in an unordered list and the sequencing values that precede the items in an ordered list. The list-style-typeproperty is used to specify both of these.

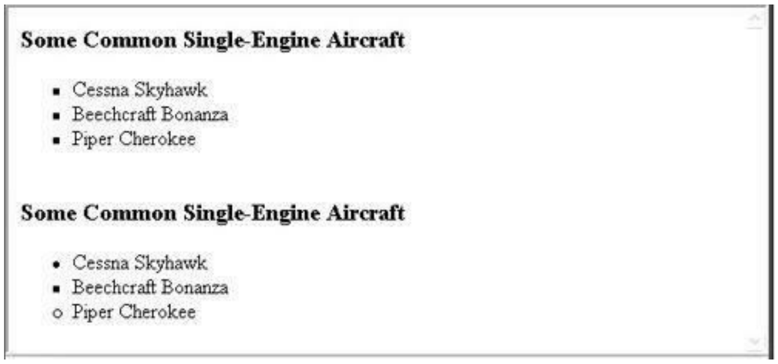
The list-style-typeproperty of an unordered list can be set to disc, circle, square, or none. A discis a small filled circle, a circleis an unfilled circle, and a squareis a filled square. The default property value for bullets is disc. For example, the following markup illustrates a document style sheet that sets the bullet type in all items in unordered lists to square:

```
<!-- bullets1 -->
<style type = "text/css">
  ul {list-style-type: square}
</style>
...
<h3> Some Common Single-Engine Aircraft </h3>
  <ul>
    <li> Cessna Skyhawk </li>
    <li> Beechcraft Bonanza </li>
    <li> Piper Cherokee </li>
  </ul>
```

Style classes can be defined to allow different list items to have different bullet types:

```
<!-- bullets2 -->
<style type = "text/css">
  li.disc {list-style-type: disc}
  li.square {list-style-type: square}
  li.circle {list-style-type: circle}
</style>
...
<h3> Some Common Single-Engine Aircraft </h3>
  <ul>
    <li class = "disc"> Cessna Skyhawk </li>
    <li class = "square"> Beechcraft Bonanza </li>
    <li class = "circle"> Piper Cherokee </li>
  </ul>
```

Figure 3.3 shows a browser display of these two lists.



Some Common Single-Engine Aircraft

▪ Cessna Skyhawk
▪ Beechcraft Bonanza
▪ Piper Cherokee

Some Common Single-Engine Aircraft

• Cessna Skyhawk
▪ Beechcraft Bonanza
○ Piper Cherokee

Figure 3.3 Examples of unordered lists

Bullets in unordered lists are not limited to discs, squares, and circles. Any image can be used in a list item bullet. Such a bullet is specified with the list- style-image property, whose value is specified with the urlform. For example, if small_plane.gifis a small image of an airplane that is stored in the same directory as the XHTML document, it could be used as follows:

```
<style type = "text/css">
  li.image {list-style-image: url(small_airplane.gif)}
</style>
  ...
  <li class = "image"> Beechcraft Bonanza </li>
```

When ordered lists are nested, it is best to use different kinds of sequence values for the different levels of nesting. The list-style-typeproperty can be used to specify the types of sequence values. Table 3.2 lists the different possibilities defined by CSS2.1.

Table 3.2 Possible sequencing value types for ordered lists in CSS2.1

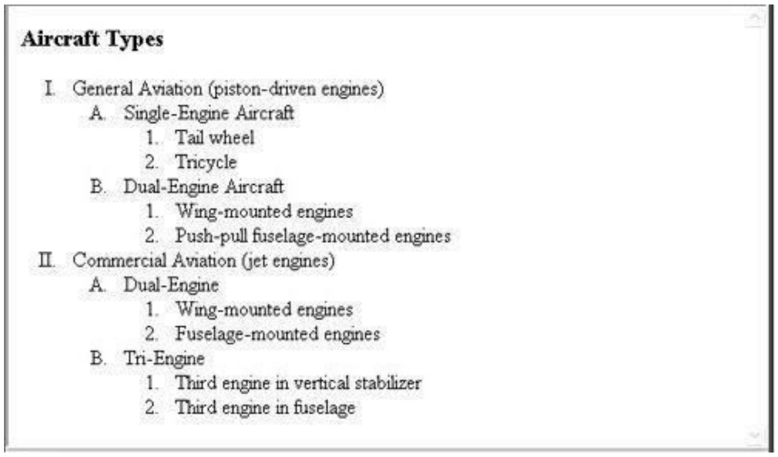
Property Values	Sequence Type

`decimal`	Arabic numerals starting with 1
`decimal-leading-zero`	Arabic numerals starting with 0
`lower-alpha`	Lowercase letters
`upper-alpha`	Uppercase letters
`lower-roman`	Lowercase Roman numerals
`upper-roman`	Uppercase Roman numerals
`lower-greek`	Lowercase Greek letters
`lower-latin`	Same as `lower-alpha`
`upper-latin`	Same as `upper-alpha`
`armenian`	Traditional Armenian numbering
`georgian`	Traditional Georgian numbering

The following example illustrates the use of different sequence value types in nested lists:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- sequence_types.html
     An example to illustrate sequence type styles
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Sequence types </title>
    <style type = "text/css">
      ol {list-style-type: upper-roman;}
      ol ol {list-style-type: upper-alpha;}
      ol ol ol {list-style-type: decimal;}
    </style>
  </head>
  <body>
    <h3> Aircraft Types </h3>
    <ol>
      <li> General Aviation (piston-driven engines)
        <ol>
          <li> Single-Engine Aircraft
            <ol>
              <li> Tail wheel </li>
              <li> Tricycle </li>

            </ol>
          </li>
          <li> Dual-Engine Aircraft
            <ol>


              <li> Wing-mounted engines </li>
              <li> Push-pull fuselage-mounted engines </li>
            </ol>
          </li>
        </ol>
      </li>
      <li> Commercial Aviation (jet engines)
        <ol>
          <li> Dual-Engine
            <ol>
              <li> Wing-mounted engines </li>
              <li> Fuselage-mounted engines </li>
            </ol>
          </li>
          <li> Tri-Engine
            <ol>
              <li> Third engine in vertical stabilizer </li>
              <li> Third engine in fuselage </li>
            </ol>
          </li>
      </ol>
    </li>
  </ol>
 </body>
</html>
```

Figure 3.4 shows a browser display of sequence_types.html.



Aircraft Types

I. General Aviation (piston-driven engines)
   A. Single-Engine Aircraft
      1. Tail wheel
      2. Tricycle
   B. Dual-Engine Aircraft
      1. Wing-mounted engines
      2. Push-pull fuselage-mounted engines
II. Commercial Aviation (jet engines)
   A. Dual-Engine
      1. Wing-mounted engines
      2. Fuselage-mounted engines
   B. Tri-Engine
      1. Third engine in vertical stabilizer
      2. Third engine in fuselage

Figure 3.4 Display of sequence_types.html

Color

If older browsers, older client machines, and CSS validation are taken into account, color is not a simple issue. For one thing, the document may be displayed on monitors of widely varying capabilities. Also, the document may be rendered by browsers that have different abilities to deal with colors. Finally, most color names recognized by browsers prevent CSS validation. This section provides an introduction to how Web sites can deal with these difficulties.

Color Groups

Three levels of collections of colors might be used by an XHTML document. The smallest useful set of colors includes only those that have standard names and are guaranteed to be correctly displayable by all browsers on all color monitors. This collection of 17 colors is called the *named colors*. These are the only color names that allow CSS validation. The names and hexadecimal codes for these named colors are shown in Table 3.3.

Table 3.3 Named colors

Name	Hexadecimal Code	Name	Hexadecimal Code
aqua	00FFFF	olive	808000
black	000000	orange	FFA500
blue	0000FF	purple	800080
fuchsia	FF00FF	red	FF0000
gray	808080	silver	C0C0C0
green	008000	teal	008080
lime	00FF00	white	FFFFFF
maroon	800000	yellow	FFFF00
navy	000080		

Most Web browsers now recognize 140 named colors, although these names are not part of a W3C standard and prevent CSS validation. This collection of colors is given in Appendix B.

A larger set of colors, called the *Web palette,* consists of 216 colors. These colors, which are often called Web-safe colors, are displayable by Windows- and Macintosh-based browsers, but may not be correctly displayed with some old terminals used on UNIX systems. Elements of this set of colors have hexadecimal values for red, green, and blue that are restricted to 00, 33, 66, 99, CC, and FF. These numbers allow all combinations of all increments of 20 percent of each of the three basic colors: red, green, and blue. The colors of the Web palette can be viewed at http://www.web-source.net/216_color_chart.htm. Note that the use of these names prevents CSS validation.

When the limitations of older browsers and monitors are not a consideration, 24-bit (or six-hexadecimal-digit) numbers can be used to specify any one of 16 million colors. Most of the time, when a color is specified that the browser or monitor cannot display, a similar color will be used.

Color Properties

The colorproperty is used to specify the foreground color of XHTML elements. For example, consider the following description of a small table:

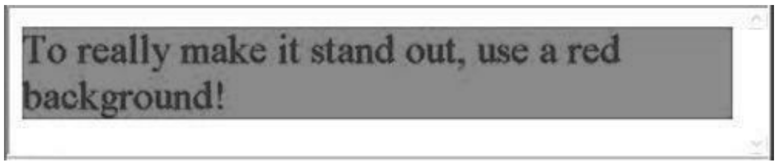
```
<style type = "text/css">
  th.red {color: red}
  th.orange {color: orange}
</style>
  ...
<table border = "5px">
  <tr>
    <th class = "red"> Apple </th>
    <th class = "orange"> Orange </th>
    <th class = "orange"> Screwdriver </th>
  </tr>
</table>
```

The background-color property is used to set the background color of an element, where the element could be the whole body of the document. For example, consider the following paragraph element:

```
<style type = "text/css">
  p.standout {font-size: 24pt; color: blue;
              background-color: red"}
</style>
...
<p class = "standout">
  To really make it stand out, use a red background!
</p>
```

When displayed by a browser, this might appear as shown in Figure 3.5.

Figure 3.5 The background-color property


To really make it stand out, use a red background!

Alignment of Text

The text-indentproperty can be used to indent the first line of a paragraph. This property takes either a length or a percentage value, as in the following markup:

```
<style type = "text/css"> p.indent {text-indent:
    0.5in}
</style>
...
<p class = "indent">
    Now is the time for all good Web programmers to begin using cascading style
    sheets for all presentation details in their documents. No more deprecated tags
    and attributes, just nice, precise style   sheets.
</p>
```

This paragraph would be displayed as follows:

```
          Now is the time for all good Web programmers to begin using cascading style
sheets for all presentation details  in their documents. No more deprecated tags and
attributes, just nice, precise style  sheets.
```

The text-align property, for which the possible keyword values are left, center, right, and justify, is used to arrange text horizontally. For example, the following document-level style sheet entry causes the content of paragraphs to be aligned on the right margin:

p {text-align: right}

The default value for text-alignis left.

The float property is used to specify that text should flow around some element, often an image or a table. The possible values for float are left, right, and none, which is the default. For example, suppose we want an image to be on the right side of the display and have text flow around the left side of the image. To specify this condition, the floatproperty of the image is set to right. Because the default value for text-alignis left, text-alignneed not be set for the text. In the following example, the text of a paragraph is specified to flow to the left of an image until the bottom of the image is reached, at which point the paragraph text flows across the whole window:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- float.html
     An example to illustrate the float property
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> The float property </title>
    <style type = "text/css">
      img {float: right}
    </style>
  </head>
  <body>
    <p>
      <img src = "c210new.jpg"  alt = "Picture of a Cessna 210" />
    </p>
    <p>
      This is a picture of a Cessna 210. The 210 is the flagship
      single-engine Cessna aircraft. Although the 210 began as a
      four-place aircraft, it soon acquired a third row of seats,
      stretching it to a six-place plane. The 210 is classified
      as a high-performance airplane, which means its landing
      gear is retractable and its engine has more than 200
      horsepower. In its first model year, which was 1960,
      the 210 was powered by a 260-horsepower fuel-injected
      six-cylinder engine that displaced 471 cubic inches.
      The 210 is the fastest single-engine airplane ever
      built by Cessna.
    </p>
  </body>
</html>
```

When rendered by a browser, float.htmlmight appear as shown in Figure 3.6, depending on the width of the browser display window.

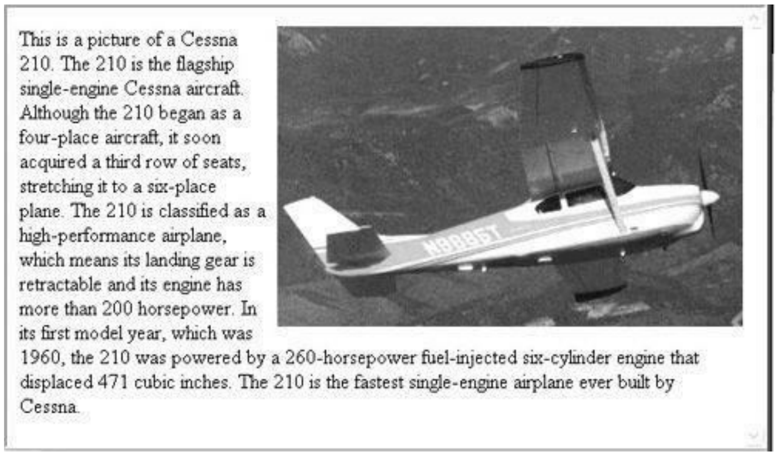
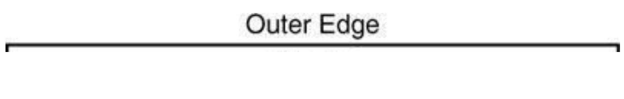

This is a picture of a Cessna 210. The 210 is the flagship single-engine Cessna aircraft. Although the 210 began as a four-place aircraft, it soon acquired a third row of seats, stretching it to a six-place plane. The 210 is classified as a high-performance airplane, which means its landing gear is retractable and its engine has more than 200 horsepower. In its first model year, which was 1960, the 210 was powered by a 260-horsepower fuel-injected six-cylinder engine that displaced 471 cubic inches. The 210 is the fastest single-engine airplane ever built by Cessna.

Figure 3.6 Display of float.html

The Box Model

Virtually all document elements can have borders with various styles, such as color and width. Furthermore, the amount of space between the content of an element and its border, known as *padding*, can be specified, as well as the space between the border and an adjacent element, known as the *margin*. This model is shown in Figure 3.7.



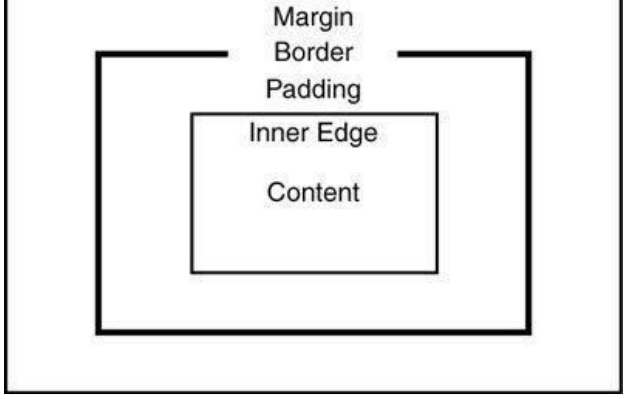


Figure 3.7 The box model

Borders

Every element has a property, border-style, that controls whether the element's content has a border, as well as specifying the style of the border.[10] CSS provides several different border styles, among them dotted, dashed, and double. The default value for border-styleis none, which is why the contents of elements do not normally have borders. The styles of one particular side of an element can be set with border-top-style, border-bottom-style, border-left-style, and border-right-style.

The border-widthproperty is used to specify the thickness of a border. Its possible values are thin, medium(the default), thick, or a length value in pixels. Setting border-widthsets the thickness of all four sides of an element. The widths of the four borders of an element can be different and are specified with border-top-width, border-bottom-width, border-left-width, and border-right-width.

The color of a border is controlled by the border-colorproperty. Once again, the individual borders of an element can be colored differently through the properties border-top-color, border-bottom-color, border-left-color, and border-right-color.

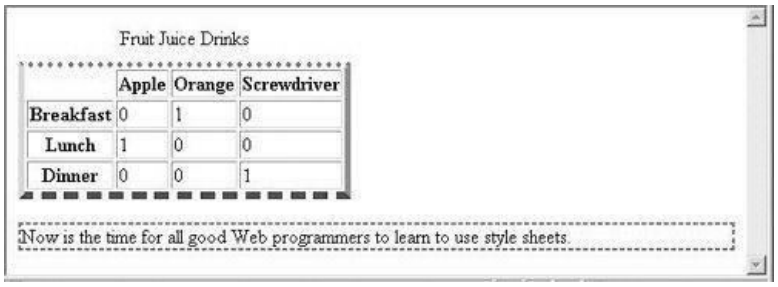
The following document, borders.html, illustrates borders, using a table and a short paragraph as examples:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- borders.html
     An example of a simple table with various borders
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Table borders </title>
    <style type = "text/css">
      table {border-top-width: medium;
             border-bottom-width: thick;
             border-top-color: red;
             border-bottom-color: blue;
             border-top-style: dotted;
             border-bottom-style: dashed;
            }
      p {border-style: dashed; border-width: thin;
         border-color: green;
        }
    </style>
  </head>
  <body>
    <table border = "5">
      <caption> Fruit Juice Drinks </caption>
      <tr>
        <th> </th>
        <th> Apple </th>
        <th> Orange </th>
        <th> Screwdriver </th>




      </tr>
      <tr>
        <th> Breakfast </th>
        <td> 0 </td>
```

```
          <td> 1 </td>
          <td> 0 </td>
        </tr>
        <tr>
          <th> Lunch </th>
          <td> 1 </td>
          <td> 0 </td>
          <td> 0 </td>
        </tr>
        <tr>
          <th> Dinner </th>
          <td> 0 </td>
          <td> 0 </td>
          <td> 1 </td>
        </tr>
      </table>
      <p>
        Now is the time for all good Web programmers to
        learn to use style sheets.
      </p>
    </body>
</html>
```

Notice that if a table has a border that was specified with its borderattribute, the border properties override the original border. In this example, the table has a regular 5-pixel border, but the top and bottom borders are replaced by those specified with the border properties.

The display of borders.htmlis shown in Figure 3.8.



Fruit Juice Drinks

	Apple	Orange	Screwdriver
Breakfast	0	1	0
Lunch	1	0	0
Dinner	0	0	1

Now is the time for all good Web programmers to learn to use style sheets.

Figure 3.8 Borders

If the borderattribute had been left out of the tableelement in borders.html, the table would have a top border and a bottom border only. It would not have left and right borders, nor would it have borders around the cells.

Margins and Padding

Recall from the box model that *padding* is the space between the content of an element and its border. The *margin* is the space between the border of an element and the element's neighbor. When there is no border, the margin plus the padding is the space between the content of an element and its neighbors. In this scenario, it may appear that there is no difference between padding and margins. However, there is a difference when the element has a background: The background extends into the padding, but not into the margin.

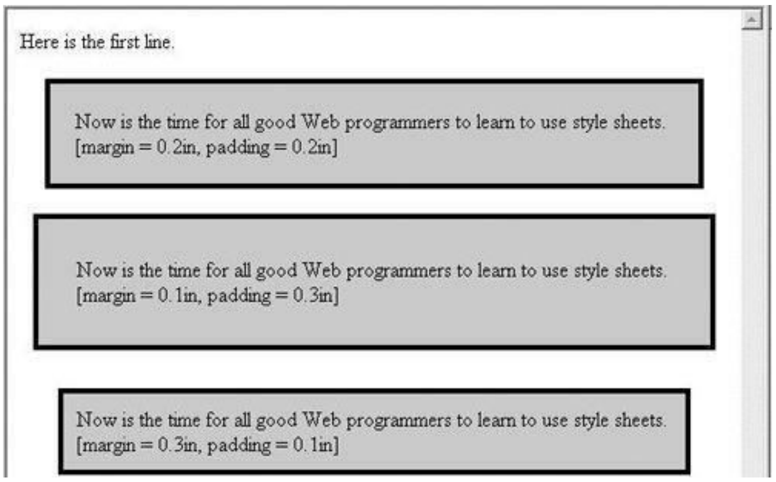
The margin properties are named margin, which applies to all four sides of an element: margin-left, margin-right, margin-top, and margin- bottom. The padding properties are named padding, which applies to all four sides: padding-left, padding-right, padding-top, and padding-bottom.

The following example, marpads.html, illustrates several combinations of margins and padding, both with and without borders:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- marpads.html
     An example to illustrate margins and padding
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Margins and Padding </title>
    <style type = "text/css">
      p.one   {margin: 0.2in;
               padding: 0.2in;
               background-color: #C0C0C0;
               border-style: solid;
```

```
            }
      p.two   {margin: 0.1in;
               padding: 0.3in;
               background-color: #C0C0C0;
               border-style: solid;
              }
      p.three {margin: 0.3in;
               padding: 0.1in;
               background-color: #C0C0C0;
               border-style: solid;
              }
      p.four  {margin:0.4in;
               background-color: #C0C0C0;}
      p.five  {padding: 0.4in;
               background-color: #C0C0C0;
              }
    </style>
  </head>
  <body>


    <p>
      Here is the first line.
    </p>
    <p class = "one">
      Now is the time for all good Web programmers to
      learn to use style sheets. <br /> [margin = 0.2in,
      padding = 0.2in]
    </p>
    <p class = "two">
      Now is the time for all good Web programmers to
      learn to use style sheets. <br /> [margin = 0.1in,
      padding = 0.3in]
    </p>
    <p class = "three">
      Now is the time for all good Web programmers to
      learn to use style sheets. <br /> [margin = 0.3in,
      padding = 0.1in]
    </p>
    <p class = "four">
      Now is the time for all good Web programmers to
      learn to use style sheets. <br /> [margin = 0.4in,
      no padding, no border]
    </p>
    <p class = "five">
      Now is the time for all good Web programmers to
      learn to use style sheets. <br /> [padding = 0.4in,
      no margin, no border]
    </p>
    <p>
      Here is the last line.
    </p>
  </body>
</html>
```

Figure 3.9 shows a browser display of marpads.html.

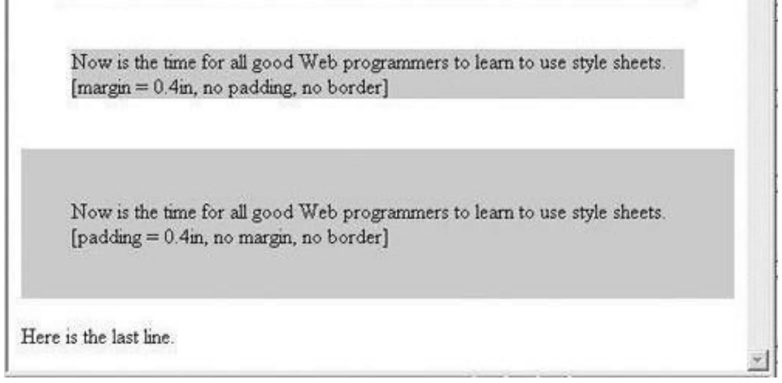


Here is the first line.

Now is the time for all good Web programmers to learn to use style sheets.
[margin = 0.2in, padding = 0.2in]

Now is the time for all good Web programmers to learn to use style sheets.
[margin = 0.1in, padding = 0.3in]

Now is the time for all good Web programmers to learn to use style sheets.
[margin = 0.3in, padding = 0.1in]


Now is the time for all good Web programmers to learn to use style sheets.
[margin = 0.4in, no padding, no border]

Now is the time for all good Web programmers to learn to use style sheets.
[padding = 0.4in, no margin, no border]

Here is the last line.

Figure 3.9 Display of marpads.html

Background Images

The background-image property is used to place an image in the background of an element. For example, an image of an airplane might be an effective background for text about the airplane. The following example illustrates background images:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!-- back_image.html
     An example to illustrate background images
      -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Background images </title>
    <style type = "text/css">
      body {background-image: url(c172.gif);}
      p {margin-left: 30px; margin-right: 30px;
         margin-top: 50px; font-size: 14pt;}
    </style>
  </head>
  <body>
    <p>
      The Cessna 172 is the most common general aviation airplane
      in the world. It is an all-metal, single-engine piston,
      high-wing, four-place monoplane. It has fixed gear and is
      categorized as a non-high-performance aircraft. The current
      model is the 172R.
      The wingspan of the 172R is 36'1". Its fuel capacity is 56
      gallons in two tanks, one in each wing. The takeoff weight
      is 2,450 pounds. Its maximum useful load is 837 pounds.
      The maximum speed of the 172R at sea level is 142 mph.
      The plane is powered by a 360-cubic-inch gasoline engine
      that develops 160 horsepower. The climb rate of the 172R
      at sea level is 720 feet per minute.
    </p>
  </body>
</html>
```

Figure 3.10 shows a browser display of back_image.html.

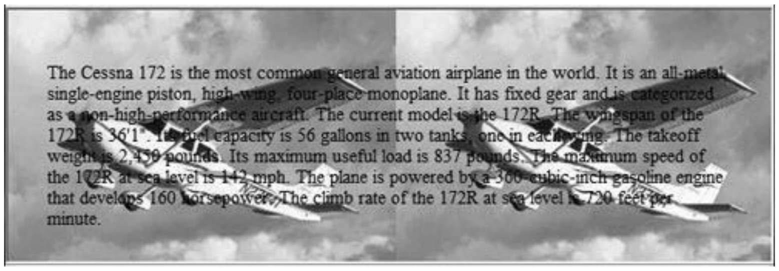

The Cessna 172 is the most common general aviation airplane in the world. It is an all-metal single-engine piston, high-wing, four-place monoplane. It has fixed gear and is categorized as a non-high-performance aircraft. The current model is the 172R. The wingspan of the 172R is 36'1". Its fuel capacity is 56 gallons in two tanks, one in each wing. The takeoff weight is 2,450 pounds. Its maximum useful load is 837 pounds. The maximum speed of the 172R at sea level is 142 mph. The plane is powered by a 360-cubic-inch gasoline engine that develops 160 horsepower. The climb rate of the 172R at sea level is 720 feet per minute.

Figure 3.10 Display of back_image.html

Text over a background image can be difficult to read if the image has areas that are nearly the same color as the text. Therefore, care must be taken in selecting the color of background images. In many cases, images with various kinds of textures in light-gray colors are best.

In the example, notice that the background image is replicated as necessary to fill the area of the element. This replication is called *tiling*. Tiling can be controlled with the background-repeatproperty, which can take the value repeat(the default), no-repeat, repeat-x, or repeat-y. The no-repeatvalue specifies that just one copy of the image is to be displayed. The repeat-xvalue means that the image is to be repeated horizontally; repeat-ymeans that the image is to be repeated vertically. In addition, the position of a nonrepeated background image can be specified with the background-position property, which can take a large number of different values. The keyword values are top, center, bottom, left, and right, all of which can be used in many different combinations. The simplest use is to use one keyword to specify the horizontal placement and one to specify the vertical placement, such as top left, bottom right, and top center. If only one keyword is given, the other is assumed to be center. So, top is equivalent to top center (or centertop), and leftis the same as center left(or left center).

The <span>and <div>Tags

In many situations, we want to apply special font properties to less than a whole paragraph of text. For example, it is often useful to have a word or phrase in a line appear in a different font size or color. The <span> tag is designed for just this purpose. Unlike most other tags, there is no default layout for the content of <span>. So, in the following example, the word totalis not displayed differently from the rest of the paragraph:

```
<p>
  It sure is fun to be in <span> total </span>
  control of text
</p>
```

The purpose of <span>is to change property values of part of a line of content, as in the following example:

```
<style type = "text/css" >
  .spanred {font-size: 24pt;
            font-family: Ariel; color: red}
</style>
...
<p>
  It sure is fun to be in
  <span class = "spanred"> total </span>
  control of text
</p>
```

The display of this paragraph is shown in Figure 3.11.

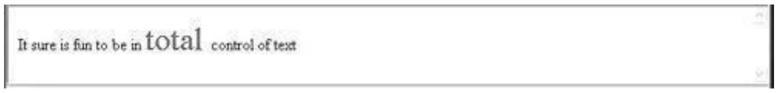

It sure is fun to be in total control of text

Figure 3.11 The <span> tag

It is common for documents to have sections, each consisting of some number of paragraphs, that have their own presentation styles. Using style classes on paragraphs, you can do this with what has already been discussed. It is more convenient, however, to be able to apply a style to a section of a document rather than to each paragraph. This can be done with the <div>tag. As with <span>, there is no implied layout for the content of the <div>tag, so its primary use is to specify presentation details for a section or division of a document.

Consider the following example, in which a division of a document is to use a specific paragraph style:

```
<div class = "primary">
  <p>
  ...
  </p>
  <p>
  ...
  </p>
  <p>
  ...
  </p>
</div>
```

The spanand divelements are used in examples in Chapter 6, “Dynamic Documents with JavaScript.”

Conflict Resolution

When there are two different values for the same property on the same element in a document, there is an obvious conflict that the browser (or other XHTML processor) must resolve. So far, we have considered only one way in which this conflict can occur: when style sheets at two or more levels specify different values for the same property on the same element. This particular kind of conflict is resolved by the precedence of the three different levels of style sheets. Inline style sheets have precedence over document and external style sheets, and document style sheets have precedence over external style sheets. However, property value conflicts can occur in several other ways. For example, a conflict may occur within a single style sheet. Consider the following style specifications, which are next to each other in the same document-level style sheet:

```
h3 {color: blue;} body h3
{color: red;}
```

Both of these specifications apply to all h3elements in the body of the document.

Inheritance is another source of property value conflicts. Such conflicts can occur if a property on a particular element has a value assigned by some style sheet and also inherits a value for that same property. Therefore, some method of resolving conflicts caused by inheritance must be available.

There can be several different origins of the specification of property values. For example, they may come from a style sheet written by the author of the document itself, but they may also come from the browser user and from the browser. For example, an FX3 user can set a minimum font size in the *Tools-Options- Advanced* window. Furthermore, browsers allow their users to write and use their own style sheets. Property values with different origins can have different precedences.

In addition, every property value specification has a particular *specificity*, depending on the particular kind of selector that is used to set it, and those specificities have different levels of precedence. These different levels are used to resolve conflicts among different specifications.

Finally, property value specifications can be marked as being important by including !importantin the specification. For example, in the specification

```
        pecial {font-style: italic !important; font-size: 14}
```

font-style: italicis important, but font-size: 14is normal. Whether a specification has been marked as being important is called the *weight* of the specification. The weight can be either normal or important. Obviously, this is another way to specify the relative precedence that a specification should have in resolving conflicts.

The details of property value conflict resolution, which are complex, will not be discussed here. Rather, what follows is a relatively brief overview of the process of property value conflict resolution.

Conflict resolution is a multistage sorting process. The first step in the process is to gather the syle specifications from the three possible levels of style sheets. These specifications are sorted into order by the relative precedence of the style sheet levels. Next, all of the available specifications (those from style sheets, those from the user, and those from the browser) are sorted by origin and weight in accordance with the following rules, in which the first has the highest precedence:

1. Important declarations with user origin

2. Important declarations with author origin

3. Normal declarations with author origin

4. Normal declarations with user origin

5. Any declarations with browser (or other user agent) origin

Note that user-origin specifications are considered to have the highest precedence. The rationale for this approach is that such specifications often are declared because of some diminished capability of the user, most often a visual impairment.

If there are conflicts after the sorting just described takes place, the next step in their resolution is a sort by specificity. This sort is based on the following rules, in which the first has the highest precedence:

1. id selectors

2. Class and pseudo class selectors

3. Contextual selectors (more element type names means that they are more specific)

4. Universal selectors

If there are still conflicts, they are resolved by giving precedence to the most recently seen specification. For this process, the specifications in an external style sheet are considered to occur at the point in the document where the link element or @importrule that references the external style sheet appears. For example, if a style sheet specifies the following, and there are no further conflicting specifications before the element is displayed, the value used will be the last (in this case, 10pt):

```
p {font-size: 12pt} p {font-
size: 10pt}
```

The whole sorting process that is used to resolve style specification conflicts is called *the cascade.*

Summary

Cascading style sheets were introduced to provide a consistent way to specify presentation details in XHTML documents. Many of the style tags and attributes designed for specifying styles that had crept into HTML were deprecated in HTML 4.0 in favor of style sheets, which can appear at three levels: inline, which apply only to the content of one specific tag; document, which apply to all appearances of specific tags in the body of a document; and external, which are stored in files by themselves and can apply to any number of documents. The property values in inline style sheets are specified in the string value of the styleattribute. Document style sheets are specified in a comment that is the content of a <style> element in the head of the document. External style sheets appear in separate files. Both document-level and external style specifications have the form of a list of style rules. Each style rule has a list of the names of tags and a list of property–value pairs. The latter apply to all occurrences of the named tags.

A style class, which is defined in the content of a <style>element, allows different occurrences of the same tag to have different property values. A generic style-class specification allows tags with different names to use the same presentation style. A pseudo class takes effect when a particular event occurs. There are many different property value forms, including lengths, percentage values, URLs, and colors. Several different properties are related to fonts. The font-familyproperty specifies one or more font names. Because different browsers support different sets of fonts, there are five generic font names. Each browser supports at least one font in each generic category. The font-sizeproperty can specify a length value or one of a number of different named size categories. The font-styleproperty can be set to italicor normal. The font-weight property is used to specify the degree of boldness of text. The fontproperty provides an abbreviated form for font-related properties. The text-decorationproperty is used to specify underlining, overlining, and line-through text.

The list-style-typeproperty is used to specify the bullet form for items in unordered lists. It is also used to specify the sequence type for the items in ordered lists.

A Web content designer must be concerned with the color capabilities of clients' browsers and monitors. The safest set of colors includes just 16 basic colors, all of which have standard names. A much larger set of relatively safe Web colors is the Web palette, which includes 216 colors. The foreground and background colors of the content of a document are specified by the colorand background-colorproperties, respectively.

The first line of a paragraph can be indented with text-indent. Text can be aligned with the text-alignproperty, whose values are left, right, and justify, which means both left and right alignment. When the floatproperty is set to leftor righton an element, text can be made to flow around that element on the right or left, respectively, in the display window.

Borders can be specified to appear around any element, in any color and any of the forms—dotted, solid, dashed, or double. The margin, which is the space between the border (or the content of the element if it has no border) and the element's neighbor, can be set with the margin properties. The padding, which is the space between the content of an element and its border (or neighbor if it has no border) can be set with the padding properties.

The background-imageproperty is used to place an image in the background of an element.

The <span>tag provides a way to include an inline style sheet that applies to a range of text that is smaller than a line or a paragraph. The <div>tag provides a way to define a section of a document that has its own style properties.

Conflict resolution for property values is a complicated process, using the origin of specifications, their specificity, inheritance, and the relative position of specifications.

Review Questions

What is the advantage of document-level style sheets over inline style sheets?

What is the purpose of external style sheets?

What attributes are required in a link to an external style sheet?

What is the format of an inline style sheet?

What is the format of a document-level style sheet, and where does the sheet appear?

What is the format of an external style sheet?

What is the form of comments within the rule list of a document-level style sheet?

What is the purpose of a style class selector?

What is the purpose of a generic class?

Are keyword property values case sensitive or case insensitive?

Why is a list of font names given as the value of a font-familyproperty?

What are the five generic fonts?

In what order must property values appear in the list of a fontproperty?

In what ways can text be modified with text-decoration?

How is the list-style-typeproperty used with unordered lists?

What are the possible values of the list-style-typeproperty when it is used with ordered lists?

If you want text to flow around the right side of an image, which value, rightor left, must be assigned to the floatproperty of the image?

Why must background images be chosen with care?

What are the possible values for the text-alignproperty?

What purpose does the text-indentproperty serve?

What properties are used to set margins around elements?

What are the three ways color property values can be specified?

If you want a background image to be repeated vertically but not horizontally, what value must be set to what property?

What properties and what values must be used to put a dotted border around a text box if the border is red and thin on the left and blue and thick on the right?

What layout information does a <span>tag by itself indicate to the browser?

What is the purpose of the <div>tag?

Which has higher precedence, an id selector or a universal selector?

Which has higher precedence, a user-origin specification or a browser specification?

If there are two conflicting specifications in a document-level style sheet, which of the two has precedence?

Exercises

Create an external style sheet for the chapters of this book.

Create and test an XHTML document that displays a table of football scores from a collegiate football conference in which the team names have one of the primary colors of their respective schools. The winning scores must appear larger and in a different font than the losing scores. The team names must be in a script font.

Create and test an XHTML document that includes at least two images and enough text to precede the images, flow around them (one on the left and one on the right), and continue after the last image.

Create and test an XHTML document that has at least a half page of text and that has a small box of text embedded on the left margin, with the main text flowing around the small box. The embedded text must appear in a smaller font and also must be set in italic.

Create and test an XHTML document that has six short paragraphs of text that describe various aspects of the state in which you live. You must define three different paragraph styles, p1, p2, and p3. The p1style must use left and right margins of 20 pixels, a background color of pink, and a foreground color of blue. The p2style must use left and right margins of 30 pixels, a background color of black, and a foreground color of yellow. The p3style must use a text indent of 1 centimeter, a background color of green, and a foreground color of white. The first and fourth paragraph must use p1, the second and fifth must use p2, and the third and sixth must use p3.

Create and test an XHTML document that describes nested ordered lists of cars. The outer list must have three entries: compact, midsize, and sports. Inside each of these three lists there must be two sublists of body styles. The compact- and midsize-car sublists are two door and four door; the sports-car sublists are coupe and convertible. Each body-style sublist must have at least three entries, each of which is the make and model of a particular car that fits the category. The outer list must use uppercase Roman numerals, the middle lists must use uppercase letters, and the inner lists must use Arabic numerals. The background color for the compact-car list must be pink; for the midsize-car list, it must be blue; for the sports-car list, it must be red. All of the styles must be in a document style sheet.

Rewrite the document of Exercise 3.6 to put all style-sheet information in an external style sheet. Validate your external style sheet with the W3C CSS validation service.

Rewrite the document of Exercise 3.6 to use inline style sheets only.

Create and test an XHTML document that contains at least five lines of text from a newspaper story. Every verb in the text must be green, every noun must be blue, and every preposition must be yellow.

Create and test an XHTML document that describes an unordered list of at least five popular books. The bullet for each book must be a small image of the book's cover. Find the images on the Web.

Use a document style sheet to modify the XHTML document, nested_lists.html in Section 2.7.2 to make the different levels of lists different colors.

Using a document style sheet, modify the XHTML document definition.htmlin Section 2.7.3 to set the font in the dtelements to Courier 12-point font and the dd elements to Times Roman 14-point italic font.

Cascading Style Sheets

- **Introduction**
- **Levels of Style Sheets**
- **Style Specification Formats**
- **Selector Forms**
- **Property Value Forms**
- **Font Properties**
- **List Properties**
- **Color**
- **Alignment of Text**
- **The Box Model**
- **Background Images**

This chapter introduces the concept of a style sheet and explains how style sheets fit into both the philosophy of XHTML and the structure of XHTML documents. An introduction to the three levels of style sheets and the format of style specifications follows. Then, the many varieties of property value forms are described. Next, specific properties for fonts and lists are introduced and illustrated. A discussion of the properties for specifying colors, background images, and text alignment follows. The box model of document elements is then discussed, along with borders and the associated padding and margin properties.

➢ Introduction

We have said that XHTML is concerned primarily with content rather than the details of how that content is presented by browsers. That is not entirely true, however, even with the tags discussed in Chapter, "Introduction to XHTML." Some of those tags—for example, <code>—specify presentation details, or style. However, these presentation specifications can be more precisely and more consistently described with style sheets. Furthermore, many of the tags and attributes that can be used for describing presentation details have been deprecated in favor of style sheets.

Most XHTML tags have associated properties that store presentation information for browsers. Browsers use default values for these properties if the document does not specify values. For example, the <h2>tag has the font-sizeproperty, for which a browser could have the default value of 18 points. A document could specify that the font-sizeproperty for <h2>be set to 20 points, which would override the default value. The new value could apply to one occurrence of an <h2>element or all such occurrences in the document, depending on how the property value is set.

A style sheet is a syntactic mechanism for specifying style information. The idea of a style sheet is not new: Word processors and desktop publishing systems have long used style sheets to impose a particular style on documents. The first style-sheet specification for use in XHTML documents, dubbed Cascading Style Sheets (CSS1), was developed in 1996 by the W3C. In mid-1998, the second standard, CSS2, was released. CSS2 added many properties and property values to CSS1. It also extended presentation control to media other than Web browsers, such as printers. As a result of the incomplete implementation of (and perhaps a lack of interest in) parts of CSS2 by browser implementors, W3C decided to develop a new standard, CSS2.1, which would reflect the level of acceptance of CSS2. Both IE8 and FX3 fully support CSS2.1, which was at the "candidate recommendation" stage as of fall 2009. CSS3 has been in development since the late 1990s.

This chapter covers most of CSS2.1 All of the examples in the chapter work correctly for both IE8 and FX3 browsers.

Perhaps the most important benefit of style sheets is their capability of imposing consistency on the style of Web documents. For example, they allow the author to specify that all occurrences of a particular tag use the same presentation style.

XHTML style sheets are called *cascading* style sheets because they can be defined at three different levels to specify the style of a document. Lower level style sheets can override higher level style sheets, so the style of the content of a tag is determined, in effect, through a cascade of style-sheet applications.

Levels of Style Sheets

The three levels of style sheets, in order from lowest level to highest level, are *inline, document level*, and *external*. Inline style sheets apply to the content of a single XHTML element, document-level style sheets apply to the whole body of a document, and external style sheets can apply to the bodies of any number of documents. Inline style sheets have precedence over document style sheets, which have precedence over external style sheets. For example, if an external style sheet specifies a value for a particular property of a particular tag, that value is used until a different value is specified in either a document style sheet or an inline style sheet. Likewise, document style sheet property values can be overridden by different property values in an inline style sheet. In effect, the properties of a specific tag are those which result from a merge of all applicable style sheets, with lower-level style sheets having precedence in cases of conflicting specifications. There are other ways style specification conflicts can occur. These ways and their resolution are discussed in Section 3.13.

If no style sheet information is specified, the browser default property values are used.

As is the case with tags and tag attributes, a particular browser may not be capable of using the property values specified in a style sheet. For example, if the value of the font-sizeproperty of a paragraph is set to 18 points, but the browser can display the particular font being used only in sizes up to 16 points, the browser obviously cannot fulfill the property specification. In this case, the browser either would substitute an alternative value or would simply ignore the given font-size value and use its default font size.

Inline style specifications appear within the opening tag and apply only to the content of that tag. This fine-grain application of style defeats one of the primary advantages of style sheets—that of imposing a uniform style on the tags of at least one whole document. Another disadvantage of inline style sheets is that they result in style information, which is expressed in a language distinct from XHTML markup, being embedded in various places in documents. It is better to keep style specifications separate from XHTML markup. For this reason, among others, W3C deprecated inline style sheets in XHTML 1.1.[1] Therefore, inline style specifications should be used sparingly. This chapter discusses inline style sheets, but we follow our own advice and make little use of them in our examples.

Document-level style specifications appear in the document head section and apply to the entire body of the document. This is obviously an effective way to impose a uniform style on the presentation of all of the content of a document.

In many cases, it is desirable to have a style sheet apply to more than one document. That is the purpose of external style sheets, which are not part of any of the documents to which they apply. They are stored separately and are referenced in all documents that use them. External style sheets are written as text files with the MIME type text/css. They can be stored on any computer on the Web. The browser fetches external style sheets just as it fetches documents. The <link>tag is used to specify external style sheets. Within <link>, the relattribute is used to specify the relationship of the linked-to document to the document in which the link appears. The hrefattribute of <link> is used to specify the URL of the style sheet document, as in the following example:

```
<link rel = "stylesheet"  type = "text/css"
      href = "http://www.cs.usc.edu/styles/wbook.css" />
```

The link to an external style sheet must appear in the head of the document. If the external style sheet resides on the Web server computer, only its path address must be given as the value of href. An example of an external style sheet appears in Section 3.6.

The @importdirective is an alternative way to use style specifications from other files. The form of this directive is

```
@import url(file name);
```

Notice that the file name is not quoted. There are two differences between linkand @import: (1) @importcan appear only at the beginning of the content of a styleelement,[2] and (2) the imported file can contain markup, as well as style, rules. In fact, sometimes the imported file contains other @importdirectives, along with some style rules.

External style sheets can be validated with the service provided at http://jigsaw.w3.org/css-validator/.

Style Specification Formats

The format of a style specification depends on the level of style sheet. Inline style specifications appear as values of the styleattribute of a tag,[3] the general form of which is as follows:

```
style = "property_1:value_1; property_2:value_2; ...;
         property_n:value_n;"
```

Although it is not required, it is recommended that the last property–value pair be followed by a semicolon.

Document style specifications appear as the content of a style element within the header of a document, although the format of the specification is quite different from that of inline style sheets. The general form of the content of a style element is as follows:[4]

```
<style type = "text/css">
  rule_list
</style>
```

The type attribute of the <style>tag tells the browser the type of style specification, which is always text/css. The type of style specification is necessary because there are other kinds of style sheets. For example, JavaScript, which can be embedded in an XHTML document, also provides style sheets that can appear in style elements.

Each style rule in a rule list has two parts: a selector, which indicates the tag or tags affected by the rule, and a list of property–value pairs. The list has the same form as the quoted list for inline style sheets, except that it is delimited by braces rather than double quotes. So, the form of a style rule is as follows:

```
selector {property_1:value_1; property_2:value_2; ...;
          property_n:value_n;}
```

If a property is given more than one value, those values usually are separated with spaces. For some properties, however, multiple values are separated with commas.

Like all other kinds of coding, complicated CSS rule lists should be documented with comments. Of course, XHTML comments cannot be used here, because CSS is not XHTML. Therefore, a different form of comment is needed. CSS comments are introduced with /*and terminated with */,[5] as in the following element:

```
<style type = "text/css">
  /* Styles for the initial paragraph */
  ...
  /* Styles for other paragraphs */
  ...
</style>
```

External style sheets have a form similar to that of document style sheets. The external file consists of a list of style rules. An example of an external style sheet appears in Section 3.6.

Selector Forms

The selector can have a variety of forms.

Simple Selector Forms

The simplest selector form is a single element name, such as h1. In this case, the property values in the rule apply to all occurrences of the named element. The selector could be a list of element names separated by commas, in which case the property values apply to all occurrences of all of the named elements. Consider the following examples, in which the property is font-sizeand the property value is a number of points:

```
h1 {font-size: 24pt;}
```

```
h2, h3 {font-size: 20pt;}
```

The first of these selector forms specifies that the text content of all h1elements must be set in a 24-point font size. The second specifies that the text content of all h2 and h3elements must be set in a 20-point font size.

Selectors can also specify that the style should apply only to elements in certain positions in the document. This is done by listing the element hierarchy in the selector, with only white space separating the element names. For example, the rule

```
form em {font-size: 14pt;}
```

applies its style only to the content of emphasis elements that are nested in a form element in the document. This is a *contextual* selector (sometimes called a *descendant* selector).

Class Selectors

Class selectors are used to allow different occurrences of the same tag to use different style specifications. A style class is defined in a style element by giving the style class a name, which is attached to the tag's name with a period. For example, if you want two paragraph styles in a document—say, normaland warning—you could define these two classes in the content of a <style>tag as follows:

p.normal {*property-value list*} p.warning
{*property-value list*}

Within the document body, the particular style class that you want is specified with the class attribute of the affected tag—in the preceding example, the paragraph tag. For example, you might have the following markup:

```
<p class = "normal">
A paragraph of text that we want to be presented in
'normal' presentation style
</p>
<p class = "warning">
A paragraph of text that is a warning to the reader, which
should be presented in an especially noticeable style
</p>
```

Generic Selectors

Sometimes it is convenient to have a class of style specifications that applies to the content of more than one kind of tag. This is done by using a generic class, which is defined without a tag name in its name. In place of the tag name, you use the name of the generic class, which must begin with a period, as in the following generic style class:

.sale {*property-value list*}

Now, in the body of a document, you could have the following markup:

```
<h3 class = "sale"> Weekend Sale </h3>
...
<p class = "sale">
...
</p>
```

idSelectors

An idselector allows the application of a style to one specific element. The general form of an idselector is as follows:[6]

#*specific-id* {*property-value list*}

As you would probably guess, the style specified in the idselector applies to the element with the specific id:

```
#section14 {font-size: 20}
```

specifies a font size of 20 points to the element

<h2 id = “section14”>1.4 Calico Cats </h2>

CSS includes still more selector forms. However, because of the lack of browser support for them, they are not discussed here.

Universal Selectors

The universal selector, denoted by an asterisk (*), applies its style to all elements in a document. For example,

```
* {color: red;}
```

makes all elements in the document red.

The universal selector is not often useful.

Pseudo Classes

Pseudo classes are styles that apply when something happens, rather than because the target element simply exists. In this section, we describe and illustrate two pseudo classes: hoverand focus.

Whereas the names of style classes and generic classes begin with a period, the names of pseudo classes begin with a colon. The style of the hoverpseudo class applies when its associated element has the mouse cursor over it. The style of the focuspseudo class applies when its associated element has focus.[7] The following document is illustrative:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- pseudo.html
     Illustrates the :hover and :focus pseudo classes.
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Pseudo Classes </title>
    <style type = "text/css">
      input:hover {background: pink; color: red;}
      input:focus {background: lightblue; color: blue;}
    </style>
  </head>
  <body>
    <form action = "">
      <p>
        <label>
          Your name:
          <input type = "text"  />
        </label>
      </p>
    </form>
  </body>
</html>
```

In pseudo.html, the content of an input element (a text box) turns red with a pink background when the mouse cursor is placed over its content. This happens, however, only when the text box does not have focus. If no text has been typed into the text box, the hover pseudo class has no effect. When the text box acquires focus, the text turns blue with a light blue background and stays like that until the left mouse button is clicked outside the box.

Property Value Forms

CSS includes 60 different properties in seven categories: fonts, lists, alignment of text, margins, colors, backgrounds, and borders. As you probably would guess, not all of these properties are discussed here. The complete details of all properties and property values can be found at the W3C Web site.

Property values can appear in a variety of forms. Keyword property values are used when there are only a few possible values and they are predefined—for example, large, medium, and small. Keyword values are not case sensitive, so Small, SmAlL, and SMALLare all the same as small.

Number values are used when no meaningful units can be attached to a numeric property value. A number value can be either an integer or a sequence of digits with a decimal point and can be preceded by a sign (+or -).

Length values are specified as number values that are followed immediately by a two-character abbreviation of a unit name. There can be no space between the number and the unit name. The possible unit names are px, for pixels; in, for inches; cm, for centimeters; mm, for millimeters; pt, for points (a point is 1/72 inch);

and pc, for picas, which are 12 points. Note that on a display, in, cm, mm, pt, and pc are approximate measures. Their actual values depend on the screen resolution. There are also two relative length values: em, which is the value of the current font size in pixels, and ex, which is the height of the letter *x*.

Percentage values are used to provide a measure that is relative to the previously used measure for a property value. Percentage values are numbers that are followed immediately by a percent sign (%). For example, if the font size were set to 75%, it would make the new current size for the font 75 percent of its previous value. Font size would stay at the new value until changed again. Percentage values can be signed. If preceded by a plus sign, the percentage is added to the previous value; if negative, the percentage is subtracted.

URL property values use a form that is slightly different from references to URLs in links. The actual URL, which can be either absolute or relative, is placed in parentheses and preceded by url, as in the following property:

```
url(tetons.jpg)
```

There can be no space between urland the left parenthesis.

Color property values can be specified as color names, as six-digit hexadecimal numbers, or in RGB form. RGB form is just the word rgb followed by a parenthesized list of three numbers that specify the levels of red, green, and blue, respectively. The RGB values can be given either as decimal numbers between 0 and 255 or as percentages. Hexadecimal numbers must be preceded with pound signs (#), as in #43AF00. For example, powder blue could be specified with

```
fuchsia
```

or

```
rgb(255, 0, 255)
```

or

```
#FF00FF
```

CSS also includes properties for counters and strings, but they are not covered here.

Some property values are inherited by elements nested in the element for which the values are specified. For example, the property background-coloris not inherited, but font-sizeis. Using a style sheet to set a value for an inheritable property for the <body>tag effectively sets it as a default property value for the whole document, as in

```
body {font-size: 16pt}
```

Unless overridden by a style sheet that applies to paragraph elements, every paragraph element in the body of this document would inherit the font size of 16 points.

Font Properties

The font properties are among the most commonly used of the style-sheet properties. Virtually all XHTML documents include text, which is often used in a variety of different situations. This creates a need for text in many different fonts, font styles, and sizes. The font properties allow us to specify these different forms.

Font Families

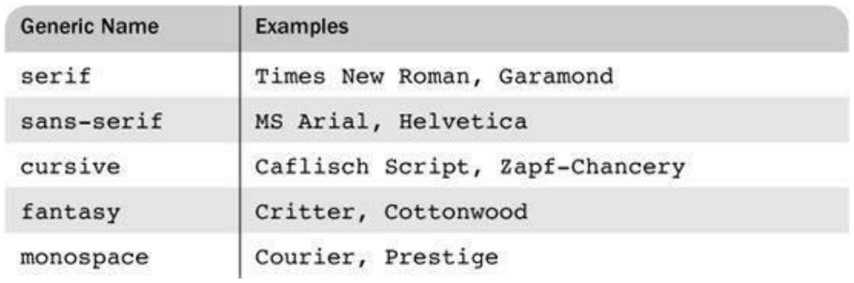
The font-familyproperty is used to specify a list of font names. The browser uses the first font in the list that it supports. For example, the property:

```
font-family: Arial, Helvetica, Futura
```

tells the browser to use Arial if it supports that font. If not, it will use Helvetica if it supports it. If the browser supports neither Arial nor Helvetica, it will use Futura if it can. If the browser does not support any of the specified fonts, it will use an alternative of its choosing.

A generic font can be specified as a font-familyvalue. The possible generic fonts and examples of each are shown in Table 3.1. Each browser has a font defined for each of these generic names. A good approach to specifying fonts is to use a generic font as the last font in the value of a font-familyproperty. For example, because Arial, Helvetica, and Futura are sans-serif fonts,[8] the previous example would be better as follows:

Table 3.1 Generic fonts



Generic Name	Examples
serif	Times New Roman, Garamond
sans-serif	MS Arial, Helvetica
cursive	Caflisch Script, Zapf-Chancery
fantasy	Critter, Cottonwood
monospace	Courier, Prestige

```
font-family: Arial, Helvetica, Futura, sans-serif
```

If a font name has more than one word, the whole name should be delimited by single quotes,[9] as in the following example:

```
font-family: 'Times New Roman'
```

In practice, the quotes may not be mandatory, but their use is recommended because they may be required in the future.

Font Sizes

The font-sizeproperty does what its name implies. For example, the following property specification sets the font size for text to 10 points:

```
font-size: 10pt
```

Many relative font-size values are defined, including xx-small, x-small, small, medium, large, x-large, and xx-large. In addition, smalleror largercan be specified. Furthermore, the value can be a percentage relative to the current font size.

On the one hand, using the relative font sizes has the disadvantage of failing to provide strict font size control. Different browsers can use different values for them. For example, small might mean 10 points on one browser and 8 points on another. On the other hand, using a specific font size has the risk that some browsers may not support that particular size, causing the document to appear different on different browsers.

Font Variants

The default value of the font-variantproperty is normal, which specifies the usual character font. This property can be set to small-capsto specify small capital characters. These characters are all uppercase, but the letters that are normally uppercase are somewhat larger than those that are normally lowercase.

Font Styles

The font-styleproperty is most commonly used to specify italic, as in

```
font-style: italic
```

An alternative to italicis oblique, but when displayed, the two are nearly identical, so obliqueis not a terribly useful font style. In fact, some browsers do not support it, so they display all oblique fonts in italic.

Font Weights

The font-weightproperty is used to specify the degree of boldness, as in

```
font-weight: bold
```

Besides bold, the values normal(the default), bolder, and lightercan be specified. The bolderand lightervalues are taken as relative to the current level of boldness. Specific numbers also can be given in multiples of 100from 100to 900, where 400is the same as normaland 700is the same as bold.

Font Shorthands

If more than one font property must be specified, the values can be stated in a list as the value of the fontproperty. The browser then has the responsibility for determining which properties to assign from the forms of the values. For example, theproperty

```
font: bold 14pt 'Times New Roman' Palatino
```

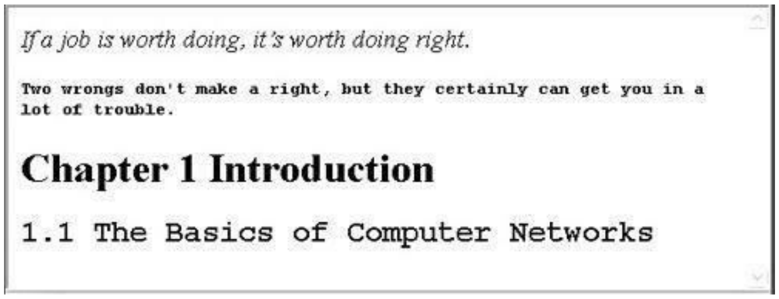
specifies that the font weight should be bold, the font size should be 14 points, and either Times New Roman or Palatino font should be used, with precedence given to Times New Roman.

The order in which the property values are given in a fontvalue list is important. The order must be as follows: The font names must be last, the font size must be second to last, and the font style, font variant, and font weight, when they are included, can be in any order but must precede the font size and font names. Only the font size and the font family are required in the fontvalue list.

The document fonts.htmlillustrates some aspects of style-sheet specification of the font properties in headings and paragraphs:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- fonts.html
     An example to illustrate font properties
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Font properties </title>
    <style type = "text/css">
      p.major {font-size: 14pt;
               font-style: italic;
               font-family: 'Times New Roman';
              }
      p.minor {font: 10pt bold 'Courier New';}
      h2 {font-family: 'Times New Roman';
          font-size: 24pt; font-weight: bold}
      h3 {font-family: 'Courier New'; font-size: 18pt}
    </style>
  </head>
  <body>
    <p class = "major">
      If a job is worth doing, it's worth doing right.
    </p>
    <p class = "minor">
      Two wrongs don't make a right, but they certainly
      can get you in a lot of trouble.
    </p>
    <h2> Chapter 1 Introduction </h2>
    <h3> 1.1 The Basics of Computer Networks </h3>
  </body>
</html>
```

Figure 3.1 shows a browser display of fonts.html.



If a job is worth doing, it's worth doing right.

Two wrongs don't make a right, but they certainly can get you in a lot of trouble.

Chapter 1 Introduction

1.1 The Basics of Computer Networks

Figure 3.1 Display of fonts.html

The following document, called fonts2.html, is a revision of fonts.htmlthat uses an external style sheet in place of the document style sheet used in fonts.html(the external style sheet, styles.css, follows the revised document):

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- fonts2.html
     An example to test external style sheets
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> External style sheets </title>
    <link rel = "stylesheet"  type = "text/css"
          href = "styles.css" />
  </head>
  <body>
    <p class = "major">
      If a job is worth doing, it's worth doing right.
    </p>
    <p class = "minor">
      Two wrongs don't make a right, but they certainly
      can get you in a lot of trouble.
    </p>
    <h2> Chapter 1 Introduction </h2>
    <h3> 1.1 The Basics of Computer Networks </h3>
  </body>
</html>
```

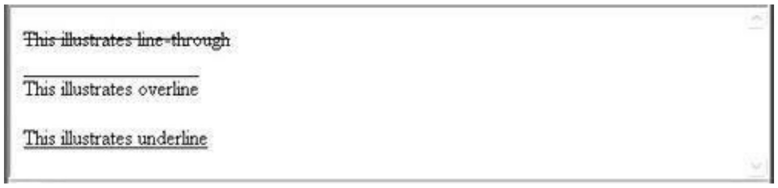
```
/* styles.css - an external style sheet
     for use with fonts2.html
   */
  p.major {font-size: 14pt;
           font-style: italic;
           font-family: 'Times New Roman';
        }
  p.minor {font: bold 10pt 'Courier New';}
  h2 {font-family: 'Times New Roman';
      font-size: 24pt; font-weight: bold}
  h3 {font-family: 'Courier New';
      font-size: 18pt}
```

Text Decoration

The text-decoration property is used to specify some special features of text. The available values are line-through, overline, underline, and none, which is the default. Many browsers implicitly underline links. The nonevalue can be used to avoid this underlining. Note that text-decorationis not inherited. The following document, decoration.html, illustrates the line-through, overline, and underlinevalues:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- decoration.html
     An example that illustrates several of the
     possible text decoration values
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Text decoration </title>
    <style type = "text/css">
      p.delete {text-decoration: line-through}
      p.cap {text-decoration: overline}
      p.attention {text-decoration: underline}
    </style>
  </head>
  <body>
    <p class = "delete">
      This illustrates line-through
    </p>
    <p class= "cap">
      This illustrates overline
    </p>
    <p class = "attention">
      This illustrates underline
    </p>
  </body>
</html>
```

Figure 3.2 shows a browser display of decoration.html.



~~This illustrates line-through~~

This illustrates overline

This illustrates underline

Figure 3.2 Display of decoration.html

The letter-spacing property controls the amount of space between characters in text. The possible values of letter-spacing are any length property values—for example, 3px.

List Properties

Two presentation details of lists can be specified in XHTML documents: the shape of the bullets that precede the items in an unordered list and the sequencing values that precede the items in an ordered list. The list-style-typeproperty is used to specify both of these.

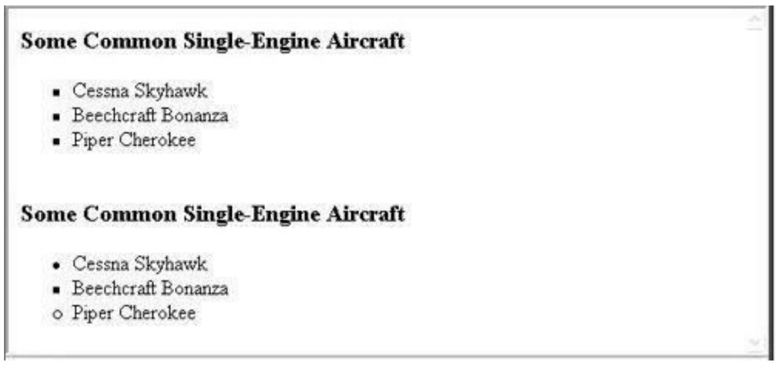
The list-style-typeproperty of an unordered list can be set to disc, circle, square, or none. A discis a small filled circle, a circleis an unfilled circle, and a squareis a filled square. The default property value for bullets is disc. For example, the following markup illustrates a document style sheet that sets the bullet type in all items in unordered lists to square:

```
<!-- bullets1 -->
<style type = "text/css">
  ul {list-style-type: square}
</style>
...
<h3> Some Common Single-Engine Aircraft </h3>
  <ul>
    <li> Cessna Skyhawk </li>
    <li> Beechcraft Bonanza </li>
    <li> Piper Cherokee </li>
  </ul>
```

Style classes can be defined to allow different list items to have different bullet types:

```
<!-- bullets2 -->
<style type = "text/css">
  li.disc {list-style-type: disc}
  li.square {list-style-type: square}
  li.circle {list-style-type: circle}
</style>
...
<h3> Some Common Single-Engine Aircraft </h3>
  <ul>
    <li class = "disc"> Cessna Skyhawk </li>
    <li class = "square"> Beechcraft Bonanza </li>
    <li class = "circle"> Piper Cherokee </li>
  </ul>
```

Figure 3.3 shows a browser display of these two lists.


Some Common Single-Engine Aircraft

- Cessna Skyhawk
- Beechcraft Bonanza
- Piper Cherokee

Some Common Single-Engine Aircraft

- Cessna Skyhawk
- Beechcraft Bonanza
- Piper Cherokee

Figure 3.3 Examples of unordered lists

Bullets in unordered lists are not limited to discs, squares, and circles. Any image can be used in a list item bullet. Such a bullet is specified with the list- style-image property, whose value is specified with the urlform. For example, if small_plane.gifis a small image of an airplane that is stored in the same directory as the XHTML document, it could be used as follows:

```
<style type = "text/css">
  li.image {list-style-image: url(small_airplane.gif)}
</style>
  ...
  <li class = "image"> Beechcraft Bonanza </li>
```

When ordered lists are nested, it is best to use different kinds of sequence values for the different levels of nesting. The list-style-typeproperty can be used to specify the types of sequence values. Table 3.2 lists the different possibilities defined by CSS2.1.

Table 3.2 Possible sequencing value types for ordered lists in CSS2.1

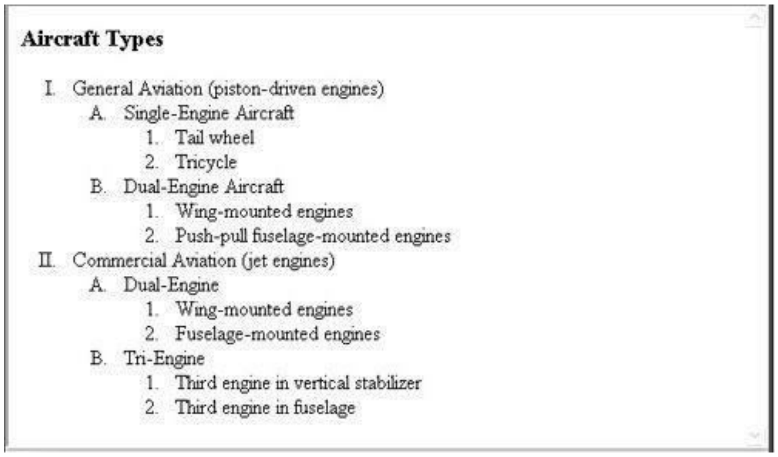
Property Values	Sequence Type

`decimal`	Arabic numerals starting with 1
`decimal-leading-zero`	Arabic numerals starting with 0
`lower-alpha`	Lowercase letters
`upper-alpha`	Uppercase letters
`lower-roman`	Lowercase Roman numerals
`upper-roman`	Uppercase Roman numerals
`lower-greek`	Lowercase Greek letters
`lower-latin`	Same as `lower-alpha`
`upper-latin`	Same as `upper-alpha`
`armenian`	Traditional Armenian numbering
`georgian`	Traditional Georgian numbering

The following example illustrates the use of different sequence value types in nested lists:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- sequence_types.html
     An example to illustrate sequence type styles
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Sequence types </title>
    <style type = "text/css">
      ol {list-style-type: upper-roman;}
      ol ol {list-style-type: upper-alpha;}
      ol ol ol {list-style-type: decimal;}
    </style>
  </head>
  <body>
    <h3> Aircraft Types </h3>
    <ol>
      <li> General Aviation (piston-driven engines)
        <ol>
          <li> Single-Engine Aircraft
            <ol>
              <li> Tail wheel </li>
              <li> Tricycle </li>
            </ol>
          </li>
          <li> Dual-Engine Aircraft
            <ol>
              <li> Wing-mounted engines </li>
              <li> Push-pull fuselage-mounted engines </li>
            </ol>
          </li>
        </ol>
      </li>
      <li> Commercial Aviation (jet engines)
        <ol>
          <li> Dual-Engine
            <ol>
              <li> Wing-mounted engines </li>
              <li> Fuselage-mounted engines </li>
            </ol>
          </li>
          <li> Tri-Engine
            <ol>
              <li> Third engine in vertical stabilizer </li>
              <li> Third engine in fuselage </li>
            </ol>
          </li>
        </ol>
      </li>
    </ol>
  </body>
</html>
```

Figure 3.4 shows a browser display of sequence_types.html.



Aircraft Types

```
I.  General Aviation (piston-driven engines)
    A.  Single-Engine Aircraft
        1.  Tail wheel
        2.  Tricycle
    B.  Dual-Engine Aircraft
        1.  Wing-mounted engines
        2.  Push-pull fuselage-mounted engines
II. Commercial Aviation (jet engines)
    A.  Dual-Engine
        1.  Wing-mounted engines
        2.  Fuselage-mounted engines
    B.  Tri-Engine
        1.  Third engine in vertical stabilizer
        2.  Third engine in fuselage
```

Figure 3.4 Display of sequence_types.html

Color

If older browsers, older client machines, and CSS validation are taken into account, color is not a simple issue. For one thing, the document may be displayed on monitors of widely varying capabilities. Also, the document may be rendered by browsers that have different abilities to deal with colors. Finally, most color names recognized by browsers prevent CSS validation. This section provides an introduction to how Web sites can deal with these difficulties.

Color Groups

Three levels of collections of colors might be used by an XHTML document. The smallest useful set of colors includes only those that have standard names and are guaranteed to be correctly displayable by all browsers on all color monitors. This collection of 17 colors is called the *named colors*. These are the only color names that allow CSS validation. The names and hexadecimal codes for these named colors are shown in Table 3.3.

Table 3.3 Named colors

Name	Hexadecimal Code	Name	Hexadecimal Code
aqua	00FFFF	olive	808000
black	000000	orange	FFA500
blue	0000FF	purple	800080
fuchsia	FF00FF	red	FF0000
gray	808080	silver	C0C0C0
green	008000	teal	008080
lime	00FF00	white	FFFFFF
maroon	800000	yellow	FFFF00
navy	000080		

Most Web browsers now recognize 140 named colors, although these names are not part of a W3C standard and prevent CSS validation. This collection of colors is given in Appendix B.

A larger set of colors, called the *Web palette,* consists of 216 colors. These colors, which are often called Web-safe colors, are displayable by Windows- and Macintosh-based browsers, but may not be correctly displayed with some old terminals used on UNIX systems. Elements of this set of colors have hexadecimal values for red, green, and blue that are restricted to 00, 33, 66, 99, CC, and FF. These numbers allow all combinations of all increments of 20 percent of each of the three basic colors: red, green, and blue. The colors of the Web palette can be viewed at http://www.web-source.net/216_color_chart.htm. Note that the use of these names prevents CSS validation.

When the limitations of older browsers and monitors are not a consideration, 24-bit (or six-hexadecimal-digit) numbers can be used to specify any one of 16 million colors. Most of the time, when a color is specified that the browser or monitor cannot display, a similar color will be used.

Color Properties

The colorproperty is used to specify the foreground color of XHTML elements. For example, consider the following description of a small table:

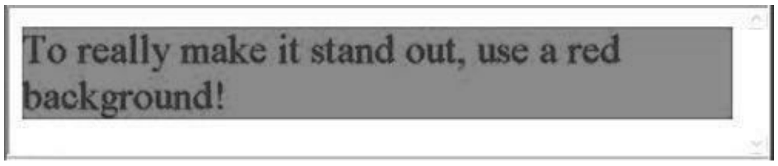
```
<style type = "text/css">
  th.red {color: red}
  th.orange {color: orange}
</style>
  ...
<table border = "5px">
  <tr>
    <th class = "red"> Apple </th>
    <th class = "orange"> Orange </th>
    <th class = "orange"> Screwdriver </th>
  </tr>
</table>
```

The background-color property is used to set the background color of an element, where the element could be the whole body of the document. For example, consider the following paragraph element:

```
<style type = "text/css">
  p.standout {font-size: 24pt; color: blue;
              background-color: red"}
</style>
...
<p class = "standout">
  To really make it stand out, use a red background!
</p>
```

When displayed by a browser, this might appear as shown in Figure 3.5.

Figure 3.5 The background-color property


To really make it stand out, use a red background!

Alignment of Text

The text-indentproperty can be used to indent the first line of a paragraph. This property takes either a length or a percentage value, as in the following markup:

```
<style type = "text/css"> p.indent {text-indent:
    0.5in}
</style>
...
<p class = "indent">
    Now is the time for all good Web programmers to begin using cascading style
    sheets for all presentation details in their documents. No more deprecated tags
    and attributes, just nice, precise style   sheets.
</p>
```

This paragraph would be displayed as follows:

```
          Now is the time for all good Web programmers to begin using cascading style
sheets for all presentation details  in their documents. No more deprecated tags and
attributes, just nice, precise style  sheets.
```

The text-align property, for which the possible keyword values are left, center, right, and justify, is used to arrange text horizontally. For example, the following document-level style sheet entry causes the content of paragraphs to be aligned on the right margin:

p {text-align: right}

The default value for text-alignis left.

The float property is used to specify that text should flow around some element, often an image or a table. The possible values for float are left, right, and none, which is the default. For example, suppose we want an image to be on the right side of the display and have text flow around the left side of the image. To specify this condition, the floatproperty of the image is set to right. Because the default value for text-alignis left, text-alignneed not be set for the text. In the following example, the text of a paragraph is specified to flow to the left of an image until the bottom of the image is reached, at which point the paragraph text flows across the whole window:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- float.html
     An example to illustrate the float property
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> The float property </title>
    <style type = "text/css">
      img {float: right}
    </style>
  </head>
  <body>
    <p>
      <img src = "c210new.jpg"  alt = "Picture of a Cessna 210" />
    </p>
    <p>
      This is a picture of a Cessna 210. The 210 is the flagship
      single-engine Cessna aircraft. Although the 210 began as a
      four-place aircraft, it soon acquired a third row of seats,
      stretching it to a six-place plane. The 210 is classified
      as a high-performance airplane, which means its landing
      gear is retractable and its engine has more than 200
      horsepower. In its first model year, which was 1960,
      the 210 was powered by a 260-horsepower fuel-injected
      six-cylinder engine that displaced 471 cubic inches.
      The 210 is the fastest single-engine airplane ever
      built by Cessna.
    </p>
  </body>
</html>
```

When rendered by a browser, float.htmlmight appear as shown in Figure 3.6, depending on the width of the browser display window.

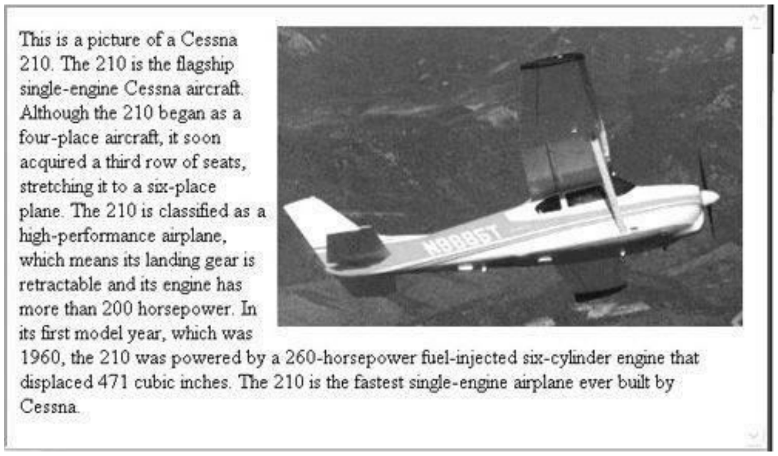
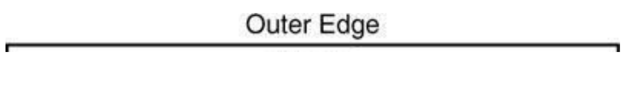

This is a picture of a Cessna 210. The 210 is the flagship single-engine Cessna aircraft. Although the 210 began as a four-place aircraft, it soon acquired a third row of seats, stretching it to a six-place plane. The 210 is classified as a high-performance airplane, which means its landing gear is retractable and its engine has more than 200 horsepower. In its first model year, which was 1960, the 210 was powered by a 260-horsepower fuel-injected six-cylinder engine that displaced 471 cubic inches. The 210 is the fastest single-engine airplane ever built by Cessna.

Figure 3.6 Display of float.html

The Box Model

Virtually all document elements can have borders with various styles, such as color and width. Furthermore, the amount of space between the content of an element and its border, known as *padding*, can be specified, as well as the space between the border and an adjacent element, known as the *margin*. This model is shown in Figure 3.7.



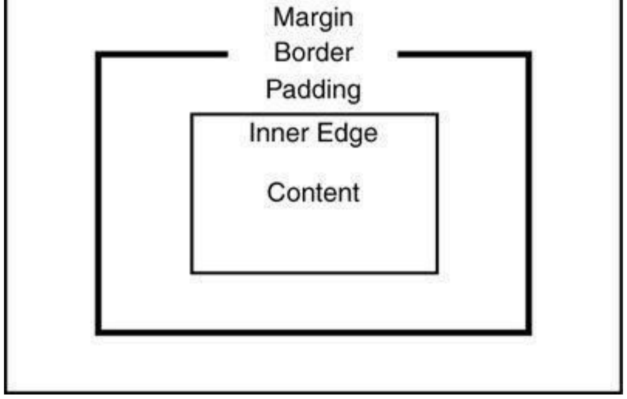


Figure 3.7 The box model

Borders

Every element has a property, border-style, that controls whether the element’s content has a border, as well as specifying the style of the border.[10] CSS provides several different border styles, among them dotted, dashed, and double. The default value for border-styleis none, which is why the contents of elements do not normally have borders. The styles of one particular side of an element can be set with border-top-style, border-bottom-style, border-left-style, and border-right-style.

The border-widthproperty is used to specify the thickness of a border. Its possible values are thin, medium(the default), thick, or a length value in pixels. Setting border-widthsets the thickness of all four sides of an element. The widths of the four borders of an element can be different and are specified with border-top-width, border-bottom-width, border-left-width, and border-right-width.

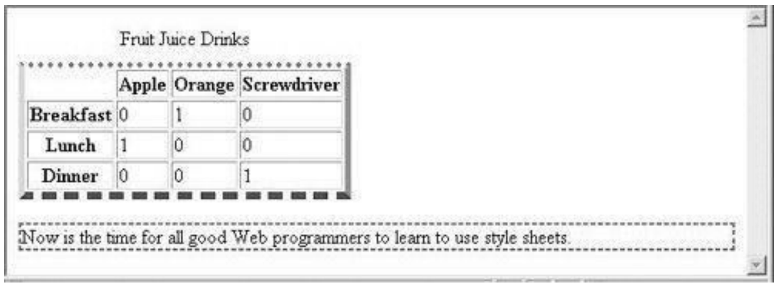
The color of a border is controlled by the border-colorproperty. Once again, the individual borders of an element can be colored differently through the properties border-top-color, border-bottom-color, border-left-color, and border-right-color.

The following document, borders.html, illustrates borders, using a table and a short paragraph as examples:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- borders.html
     An example of a simple table with various borders
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Table borders </title>
    <style type = "text/css">
      table {border-top-width: medium;
             border-bottom-width: thick;
             border-top-color: red;
             border-bottom-color: blue;
             border-top-style: dotted;
             border-bottom-style: dashed;
            }
      p {border-style: dashed; border-width: thin;
         border-color: green;
        }
    </style>
  </head>
  <body>
    <table border = "5">
      <caption> Fruit Juice Drinks </caption>
      <tr>
        <th> </th>
        <th> Apple </th>
        <th> Orange </th>
        <th> Screwdriver </th>




      </tr>
      <tr>
        <th> Breakfast </th>
        <td> 0 </td>
        <td> 1 </td>
        <td> 0 </td>
      </tr>
      <tr>
        <th> Lunch </th>
        <td> 1 </td>
        <td> 0 </td>
        <td> 0 </td>
      </tr>
      <tr>
        <th> Dinner </th>
        <td> 0 </td>
        <td> 0 </td>
        <td> 1 </td>
      </tr>
    </table>
    <p>
      Now is the time for all good Web programmers to
      learn to use style sheets.
    </p>
  </body>
</html>
```

Notice that if a table has a border that was specified with its borderattribute, the border properties override the original border. In this example, the table has a regular 5-pixel border, but the top and bottom borders are replaced by those specified with the border properties.

The display of borders.htmlis shown in Figure 3.8.



Fruit Juice Drinks

	Apple	Orange	Screwdriver
Breakfast	0	1	0
Lunch	1	0	0
Dinner	0	0	1

Now is the time for all good Web programmers to learn to use style sheets.

Figure 3.8 Borders

If the borderattribute had been left out of the tableelement in borders.html, the table would have a top border and a bottom border only. It would not have left and right borders, nor would it have borders around the cells.

Margins and Padding

Recall from the box model that *padding* is the space between the content of an element and its border. The *margin* is the space between the border of an element and the element's neighbor. When there is no border, the margin plus the padding is the space between the content of an element and its neighbors. In this scenario, it may appear that there is no difference between padding and margins. However, there is a difference when the element has a background: The background extends into the padding, but not into the margin.

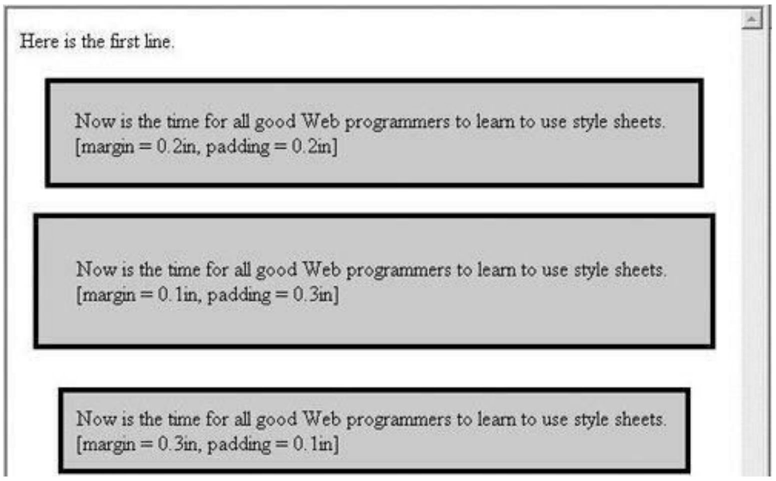
The margin properties are named margin, which applies to all four sides of an element: margin-left, margin-right, margin-top, and margin- bottom. The padding properties are named padding, which applies to all four sides: padding-left, padding-right, padding-top, and padding-bottom.

The following example, marpads.html, illustrates several combinations of margins and padding, both with and without borders:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- marpads.html
     An example to illustrate margins and padding
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Margins and Padding </title>
    <style type = "text/css">
      p.one   {margin: 0.2in;
               padding: 0.2in;
               background-color: #C0C0C0;
               border-style: solid;
```

```
             }
    p.two    {margin: 0.1in;
              padding: 0.3in;
              background-color: #C0C0C0;
              border-style: solid;
             }
    p.three {margin: 0.3in;
              padding: 0.1in;
              background-color: #C0C0C0;
              border-style: solid;
             }
    p.four   {margin:0.4in;
              background-color: #C0C0C0;}
    p.five   {padding: 0.4in;
              background-color: #C0C0C0;
             }
  </style>
 </head>
 <body>


  <p>
    Here is the first line.
  </p>
  <p class = "one">
    Now is the time for all good Web programmers to
    learn to use style sheets. <br /> [margin = 0.2in,
    padding = 0.2in]
  </p>
  <p class = "two">
    Now is the time for all good Web programmers to
    learn to use style sheets. <br /> [margin = 0.1in,
    padding = 0.3in]
  </p>
  <p class = "three">
    Now is the time for all good Web programmers to
    learn to use style sheets. <br /> [margin = 0.3in,
    padding = 0.1in]
  </p>
  <p class = "four">
    Now is the time for all good Web programmers to
    learn to use style sheets. <br /> [margin = 0.4in,
    no padding, no border]
  </p>
  <p class = "five">
    Now is the time for all good Web programmers to
    learn to use style sheets. <br /> [padding = 0.4in,
    no margin, no border]
  </p>
  <p>
    Here is the last line.
  </p>
 </body>
</html>
```

Figure 3.9 shows a browser display of marpads.html.

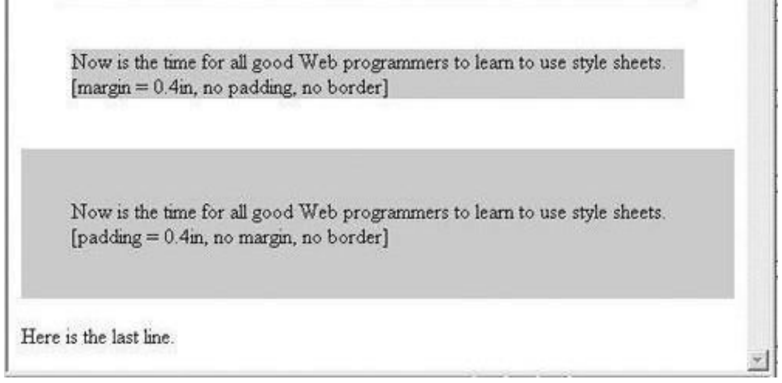


Here is the first line.

Now is the time for all good Web programmers to learn to use style sheets.
[margin = 0.2in, padding = 0.2in]

Now is the time for all good Web programmers to learn to use style sheets.
[margin = 0.1in, padding = 0.3in]

Now is the time for all good Web programmers to learn to use style sheets.
[margin = 0.3in, padding = 0.1in]


Now is the time for all good Web programmers to learn to use style sheets.
[margin = 0.4in, no padding, no border]

Now is the time for all good Web programmers to learn to use style sheets.
[padding = 0.4in, no margin, no border]

Here is the last line.

Figure 3.9 Display of marpads.html

Background Images

The background-image property is used to place an image in the background of an element. For example, an image of an airplane might be an effective background for text about the airplane. The following example illustrates background images:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!-- back_image.html
     An example to illustrate background images
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Background images </title>
    <style type = "text/css">
      body {background-image: url(c172.gif);}
      p {margin-left: 30px; margin-right: 30px;
         margin-top: 50px; font-size: 14pt;}
    </style>
  </head>
  <body>
    <p>
      The Cessna 172 is the most common general aviation airplane
      in the world. It is an all-metal, single-engine piston,
      high-wing, four-place monoplane. It has fixed gear and is
      categorized as a non-high-performance aircraft. The current
      model is the 172R.
      The wingspan of the 172R is 36'1". Its fuel capacity is 56
      gallons in two tanks, one in each wing. The takeoff weight
      is 2,450 pounds. Its maximum useful load is 837 pounds.
      The maximum speed of the 172R at sea level is 142 mph.
      The plane is powered by a 360-cubic-inch gasoline engine
      that develops 160 horsepower. The climb rate of the 172R
      at sea level is 720 feet per minute.
    </p>
  </body>
</html>
```

Figure 3.10 shows a browser display of back_image.html.

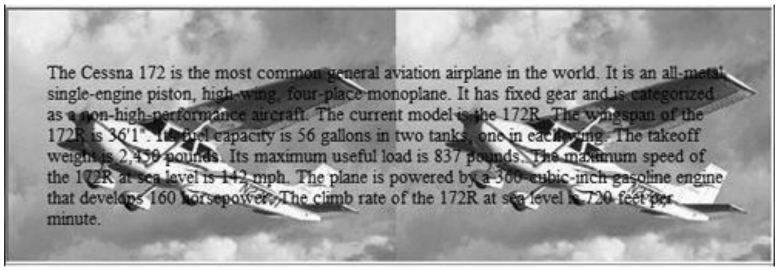

The Cessna 172 is the most common general aviation airplane in the world. It is an all-metal single-engine piston, high-wing, four-place monoplane. It has fixed gear and is categorized as a non-high-performance aircraft. The current model is the 172R. The wingspan of the 172R is 36'1". Its fuel capacity is 56 gallons in two tanks, one in each wing. The takeoff weight is 2,450 pounds. Its maximum useful load is 837 pounds. The maximum speed of the 172R at sea level is 142 mph. The plane is powered by a 360-cubic-inch gasoline engine that develops 160 horsepower. The climb rate of the 172R at sea level is 720 feet per minute.

Figure 3.10 Display of back_image.html

Text over a background image can be difficult to read if the image has areas that are nearly the same color as the text. Therefore, care must be taken in selecting the color of background images. In many cases, images with various kinds of textures in light-gray colors are best.

In the example, notice that the background image is replicated as necessary to fill the area of the element. This replication is called *tiling*. Tiling can be controlled with the background-repeatproperty, which can take the value repeat(the default), no-repeat, repeat-x, or repeat-y. The no-repeatvalue specifies that just one copy of the image is to be displayed. The repeat-xvalue means that the image is to be repeated horizontally; repeat-ymeans that the image is to be repeated vertically. In addition, the position of a nonrepeated background image can be specified with the background-position property, which can take a large number of different values. The keyword values are top, center, bottom, left, and right, all of which can be used in many different combinations. The simplest use is to use one keyword to specify the horizontal placement and one to specify the vertical placement, such as top left, bottom right, and top center. If only one keyword is given, the other is assumed to be center. So, top is equivalent to top center (or centertop), and leftis the same as center left(or left center).

The <span>and <div>Tags

In many situations, we want to apply special font properties to less than a whole paragraph of text. For example, it is often useful to have a word or phrase in a line appear in a different font size or color. The <span> tag is designed for just this purpose. Unlike most other tags, there is no default layout for the content of <span>. So, in the following example, the word totalis not displayed differently from the rest of the paragraph:

```
<p>
  It sure is fun to be in <span> total </span>
  control of text
</p>
```

The purpose of <span>is to change property values of part of a line of content, as in the following example:

```
<style type = "text/css" >
  .spanred {font-size: 24pt;
            font-family: Ariel; color: red}
</style>
...
<p>
  It sure is fun to be in
  <span class = "spanred"> total </span>
  control of text
</p>
```

The display of this paragraph is shown in Figure 3.11.

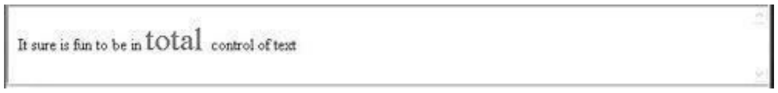

It sure is fun to be in total control of text

Figure 3.11 The <span> tag

It is common for documents to have sections, each consisting of some number of paragraphs, that have their own presentation styles. Using style classes on paragraphs, you can do this with what has already been discussed. It is more convenient, however, to be able to apply a style to a section of a document rather than to each paragraph. This can be done with the <div>tag. As with <span>, there is no implied layout for the content of the <div>tag, so its primary use is to specify presentation details for a section or division of a document.

Consider the following example, in which a division of a document is to use a specific paragraph style:

```
<div class = "primary">
  <p>
  ...
  </p>
  <p>
  ...
  </p>
  <p>
  ...
  </p>
</div>
```

The spanand divelements are used in examples in Chapter 6, “Dynamic Documents with JavaScript.”

Conflict Resolution

When there are two different values for the same property on the same element in a document, there is an obvious conflict that the browser (or other XHTML processor) must resolve. So far, we have considered only one way in which this conflict can occur: when style sheets at two or more levels specify different values for the same property on the same element. This particular kind of conflict is resolved by the precedence of the three different levels of style sheets. Inline style sheets have precedence over document and external style sheets, and document style sheets have precedence over external style sheets. However, property value conflicts can occur in several other ways. For example, a conflict may occur within a single style sheet. Consider the following style specifications, which are next to each other in the same document-level style sheet:

```
h3 {color: blue;} body h3
{color: red;}
```

Both of these specifications apply to all h3elements in the body of the document.

Inheritance is another source of property value conflicts. Such conflicts can occur if a property on a particular element has a value assigned by some style sheet and also inherits a value for that same property. Therefore, some method of resolving conflicts caused by inheritance must be available.

There can be several different origins of the specification of property values. For example, they may come from a style sheet written by the author of the document itself, but they may also come from the browser user and from the browser. For example, an FX3 user can set a minimum font size in the *Tools-Options- Advanced* window. Furthermore, browsers allow their users to write and use their own style sheets. Property values with different origins can have different precedences.

In addition, every property value specification has a particular *specificity*, depending on the particular kind of selector that is used to set it, and those specificities have different levels of precedence. These different levels are used to resolve conflicts among different specifications.

Finally, property value specifications can be marked as being important by including !importantin the specification. For example, in the specification

```
        pecial {font-style: italic !important; font-size: 14}
```

font-style: italicis important, but font-size: 14is normal. Whether a specification has been marked as being important is called the *weight* of the specification. The weight can be either normal or important. Obviously, this is another way to specify the relative precedence that a specification should have in resolving conflicts.

The details of property value conflict resolution, which are complex, will not be discussed here. Rather, what follows is a relatively brief overview of the process of property value conflict resolution.

Conflict resolution is a multistage sorting process. The first step in the process is to gather the syle specifications from the three possible levels of style sheets. These specifications are sorted into order by the relative precedence of the style sheet levels. Next, all of the available specifications (those from style sheets, those from the user, and those from the browser) are sorted by origin and weight in accordance with the following rules, in which the first has the highest precedence:

1. Important declarations with user origin

2. Important declarations with author origin

3. Normal declarations with author origin

4. Normal declarations with user origin

5. Any declarations with browser (or other user agent) origin

Note that user-origin specifications are considered to have the highest precedence. The rationale for this approach is that such specifications often are declared because of some diminished capability of the user, most often a visual impairment.

If there are conflicts after the sorting just described takes place, the next step in their resolution is a sort by specificity. This sort is based on the following rules, in which the first has the highest precedence:

5. id selectors

6. Class and pseudo class selectors

7. Contextual selectors (more element type names means that they are more specific)

8. Universal selectors

If there are still conflicts, they are resolved by giving precedence to the most recently seen specification. For this process, the specifications in an external style sheet are considered to occur at the point in the document where the link element or @importrule that references the external style sheet appears. For example, if a style sheet specifies the following, and there are no further conflicting specifications before the element is displayed, the value used will be the last (in this case, 10pt):

```
p {font-size: 12pt} p {font-
size: 10pt}
```

The whole sorting process that is used to resolve style specification conflicts is called *the cascade.*