CHAPTER

2

Introduction to 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.

2.1 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.[1] 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 3, "Cascading Style Sheets."

2.1.1 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

1. Not all text-formatting languages are based on SGML; for example, PostScript and LaTeX are not.

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.[2] 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.[3] 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.

2.1.2 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.

2. XML (eXtensible Markup Language) is the topic of Chapter 7, "Introduction to XML."

3. Validation is discussed in Section 2.5.3.

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.sourceforge.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.

2.2 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*. For example, consider the following element:

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

4. One such editor system is available at `http://www.xstandard.com`.

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.

2.3 Standard XHTML Document Structure

Every XHTML document must begin with an `xml` declaration 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 `xml` declaration 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 `xml` declaration 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 `DOCTYPE` command 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 `DOCTYPE` command, and they always end with the closing `html` tag, `</html>`. The `html` element 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 `xmlns` attribute'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.

2.4 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.

2.4.1 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

5. A document-type definition specifies the syntax rules for a particular category of XHTML documents.

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>
```

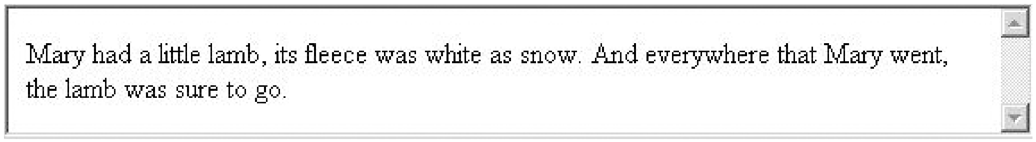

Mary had a little lamb, its fleece was white as snow. And everywhere that Mary went, the lamb was sure to go.

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>
```

6. We say "might" because the width of the display that the browser uses determines how many words will fit on a line.

Figure 2.2 shows a browser display of `greet.html`.

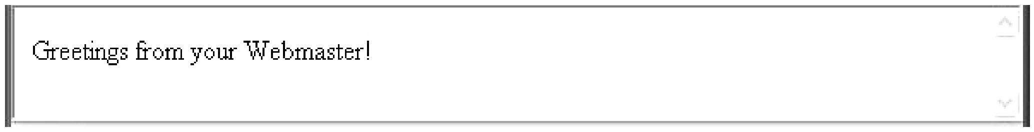
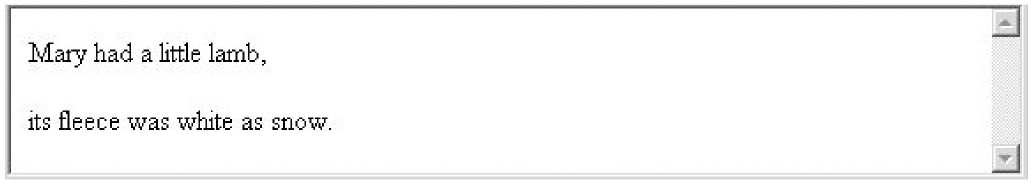

Greetings from your Webmaster!

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

2.4.2 Line Breaks

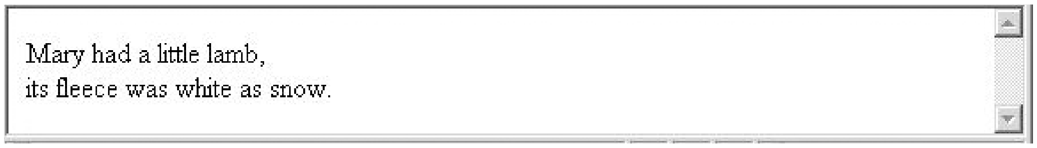
Sometimes text requires a line break without the preceding blank line. This is exactly what the break tag does. The break tag differs syntactically from the paragraph tag in that it can have no content and therefore has no closing tag (because a closing tag would serve no purpose). The break tag is specified 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.

7. Some older browsers have trouble with the tag `<br/>` but not with `<br />`.



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

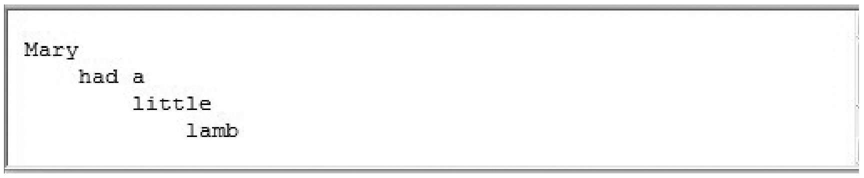
Figure 2.4 Line breaks

2.4.3 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 `pre` tag—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 `pre` element is shown in monospace, rather than in the default font.



```
Mary
    had a
        little
            lamb
```

Figure 2.5 The `pre` element

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

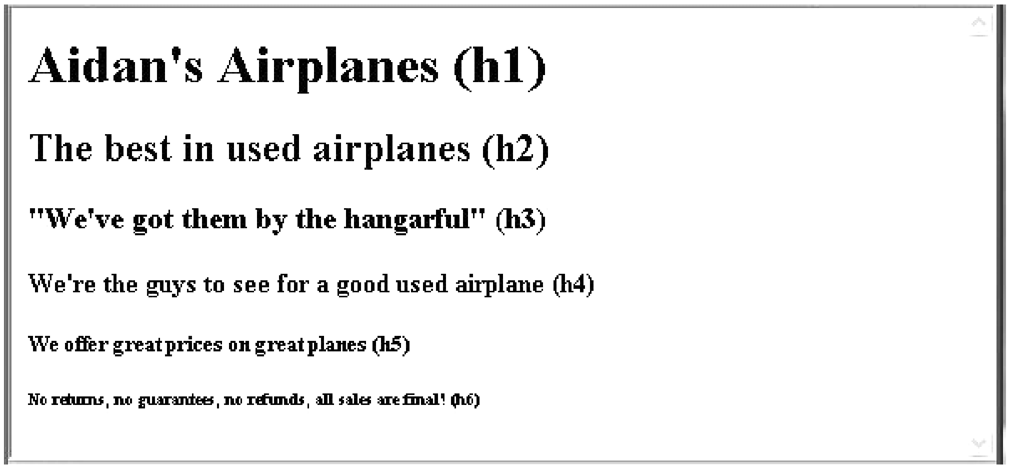
2.4.4 Headings

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)

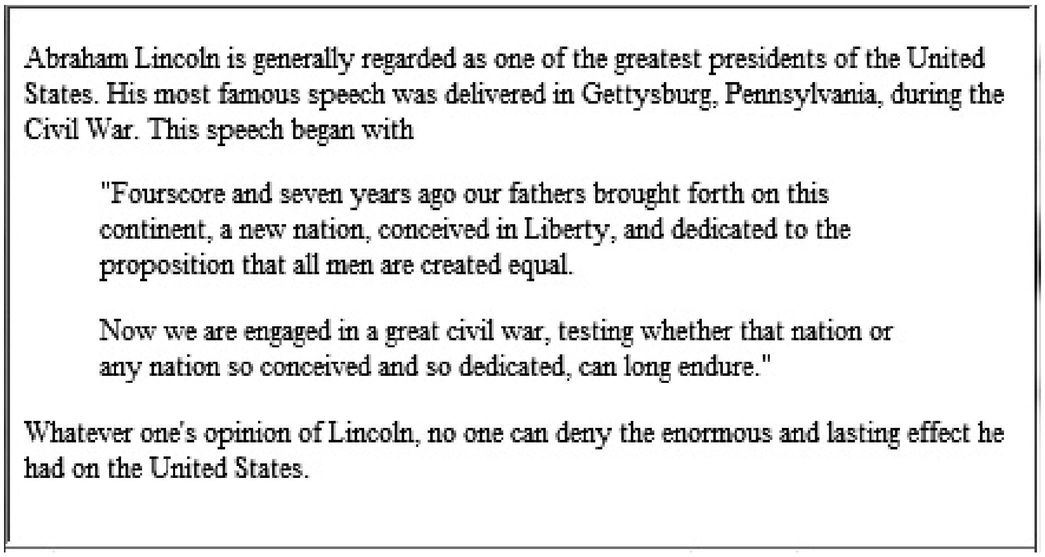
Figure 2.6 Display of `headings.html`

2.4.5 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`

2.4.6 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.

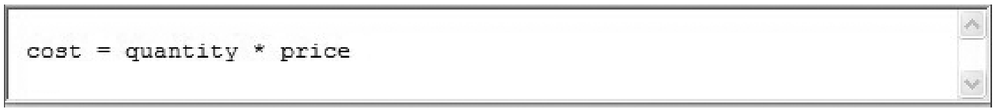

```
cost = quantity * price
```

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.

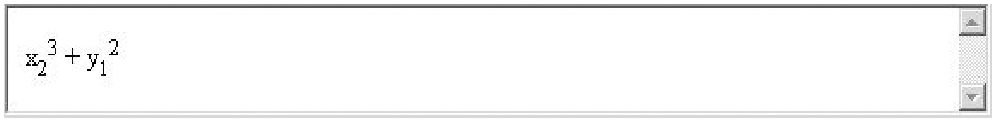

$x_2^3 + y_1^2$

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.html has the text content of its body nested in a paragraph element.

2.4.7 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)
$\frac{1}{4}$	`&frac14;`	One-quarter
$\frac{1}{2}$	`&frac12;`	One-half
$\frac{3}{4}$	`&frac34;`	Three-quarters
°	`&deg;`	Degree
(space)	`&nbsp;`	Nonbreaking space

2.4.8 Horizontal Rules

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.

2.4.9 The `meta` Element

The `meta` element 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 `content` attribute 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 `meta` element to categorize Web documents in their indices. So, if the author of a document seeks widespread exposure for the document, one or more `meta` elements 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" />
```

2.5 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.

2.5.1 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 `.JPG` or `.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`.

2.5.2 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 `src` is 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.jpg` and the image file is stored in the `images` subdirectory, the value of `src` would 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 `alt` is 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 `width` and `height` attribute values must be kept equal when they are changed.

8. The patent expired in the United States in 2003.
9. Space is not the direct issue; download time, which depends on file size, is the real issue.

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:

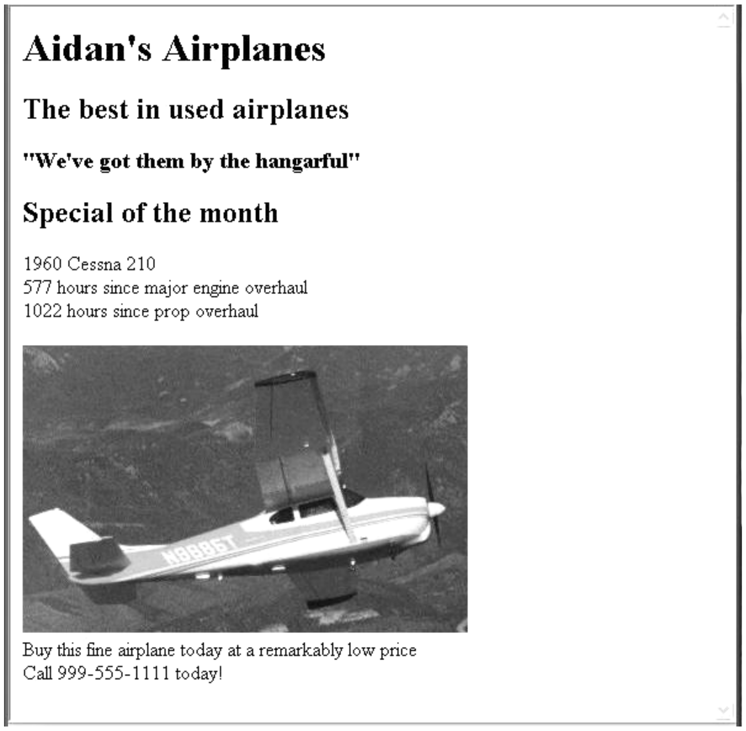
```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- 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`.

2.5.3 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

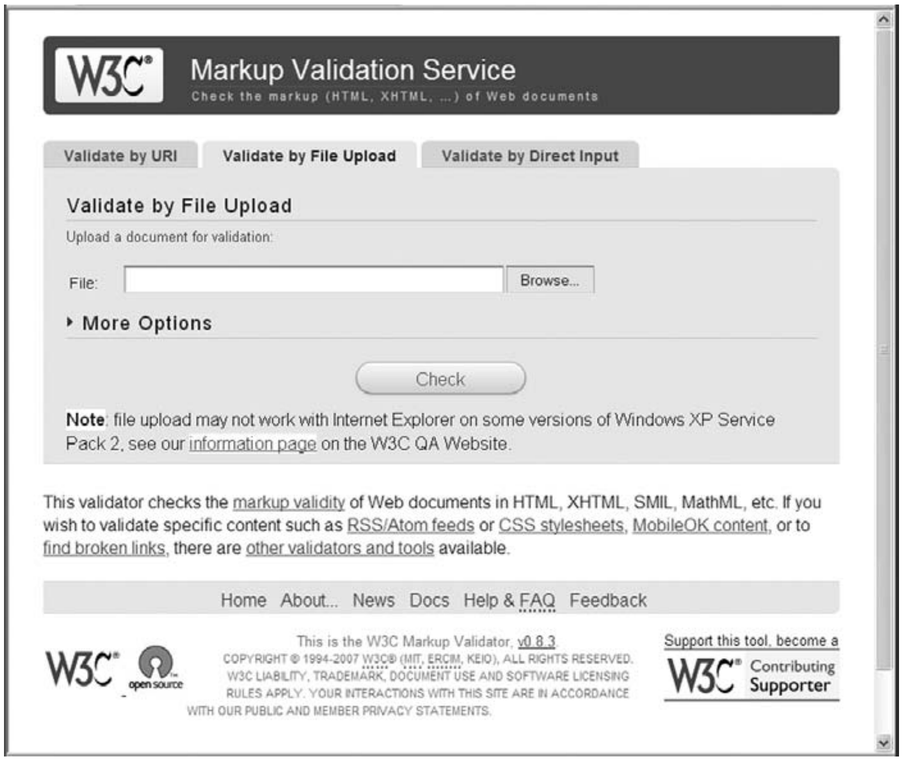
"We've got them by the hangarful"

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`



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

Validate by File Upload

Upload a document for validation:

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.

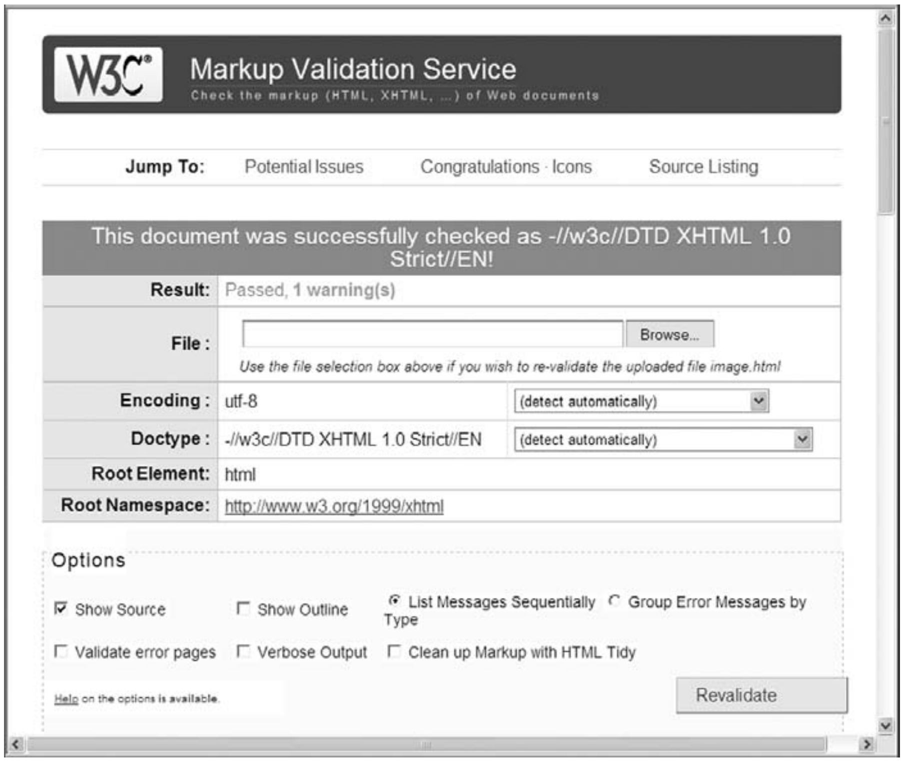
This is the W3C Markup Validator, v0.8.3
COPYRIGHT © 1994-2007 W3C® (MIT, ERCIM, KEIO), ALL RIGHTS RESERVED. W3C LIABILITY, TRADEMARK, DOCUMENT USE AND SOFTWARE LICENSING RULES APPLY. YOUR INTERACTIONS WITH THIS SITE ARE IN ACCORDANCE WITH OUR PUBLIC AND MEMBER PRIVACY STATEMENTS.

Support this tool, become a W3C® Contributing Supporter

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 `validator` server, 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.


W3C®
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 :	Browse... 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 `DOCTYPE` command. Because the `DOCTYPE` command in `image.html` specifies the `xhtml1-strict.dtd` DTD, that document is checked against the XHTML 1.0 Strict standard.

2.6 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.

2.6.1 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 `href` specifies 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.

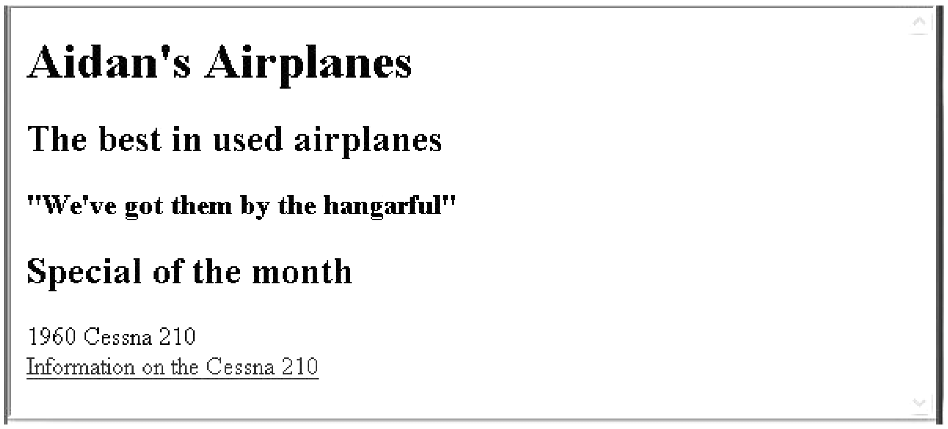
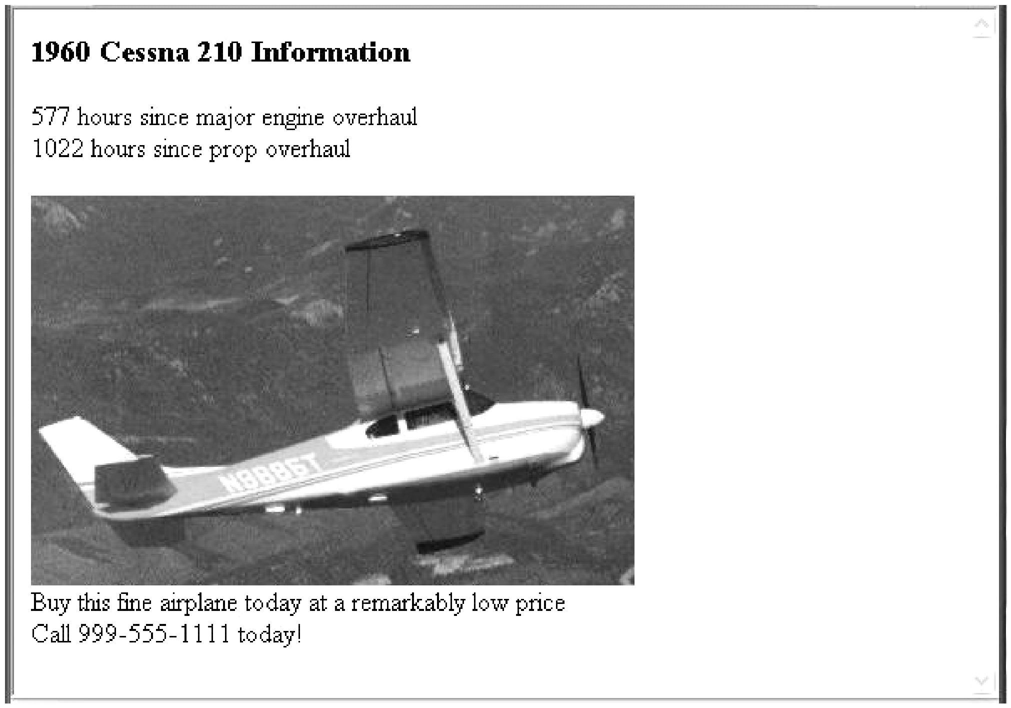

Aidan's Airplanes
The best in used airplanes
"We've got them by the hangarful"
Special of the month
1960 Cessna 210
Information on the Cessna 210

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 image element.

2.6.2 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 `id` attribute, which can then be used to identify it in an `href` attribute. Consider the following example:

```
<h2 id = "avionics"> Avionics </h2>
```

Nearly all elements can include an `id` attribute. The value of an `id` attribute must be unique within the document.

If the target is in the same document as the link, the target is specified in the `href` attribute 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 `h2` element whose `id` is `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>
```

2.6.3 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.

2.7 Lists

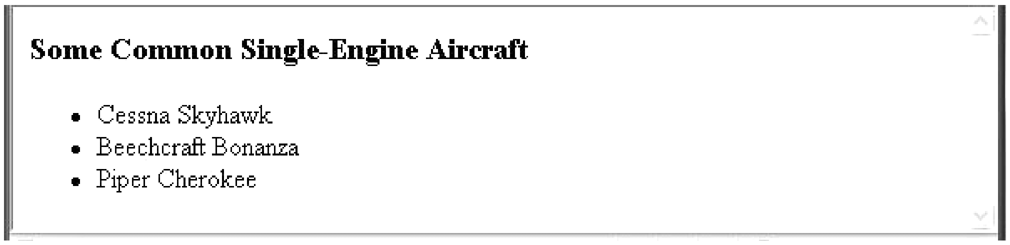
We frequently make and use lists in daily life—for example, to-do lists and grocery lists. Likewise, both printed and displayed information is littered with lists. XHTML provides simple and effective ways to specify lists in documents. The primary list types supported are those with which most people 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.

2.7.1 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. The 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">

<!-- 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>
```

produces a browser display of `unordered.html`, shown in Figure 2.15.



Some Common Single-Engine Aircraft

- Cessna Skyhawk
- Beechcraft Bonanza
- Piper Cherokee

Figure 2.15 Display of `unordered.html`

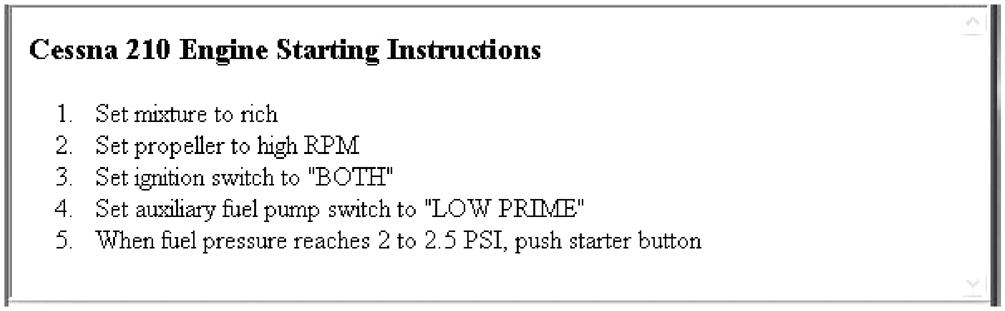
2.7.2 Ordered Lists

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

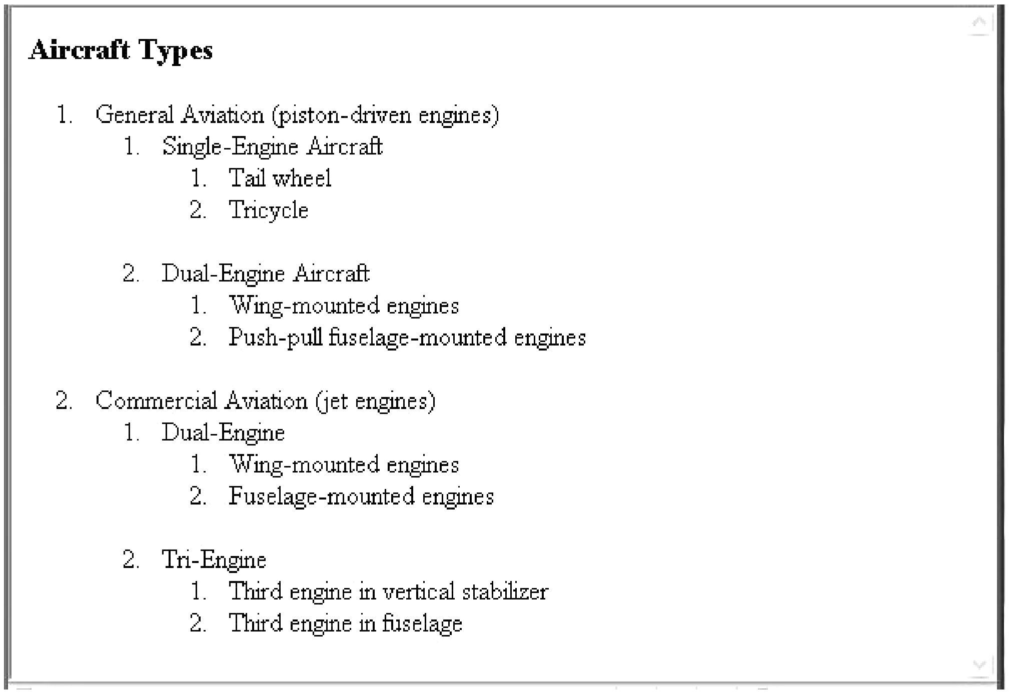
Figure 2.16 Display of `ordered.html`

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>
    <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> <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.

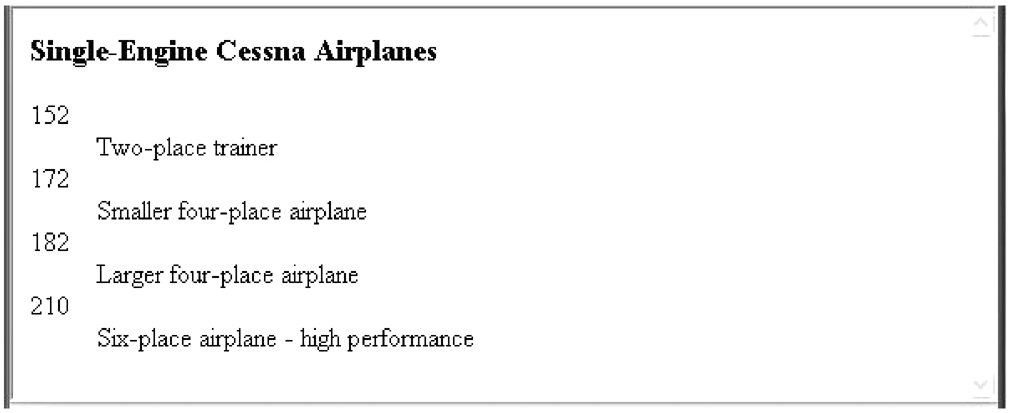
The `nested_lists.html` example uses nested ordered lists. There are no restrictions on list nesting, provided that the nesting is not direct. For example, ordered lists can be nested in unordered lists and vice versa.

2.7.3 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`

2.8 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.

2.8.1 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 `border` attribute 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 `border` attribute 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 `border` value 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 `<hx>` 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 `Screw-driver`, 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>
```

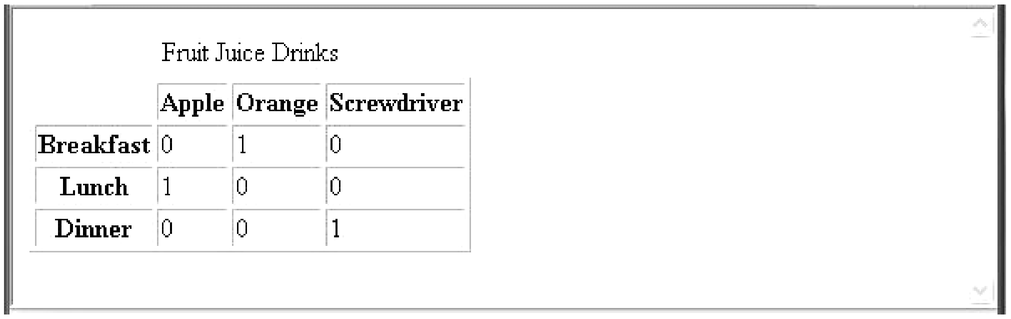
In tables that have both row and column labels, the upper-left corner cell is often empty. This empty cell is specified with a table header tag that includes no content (either `<th></th>` or just `<th />`).

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.

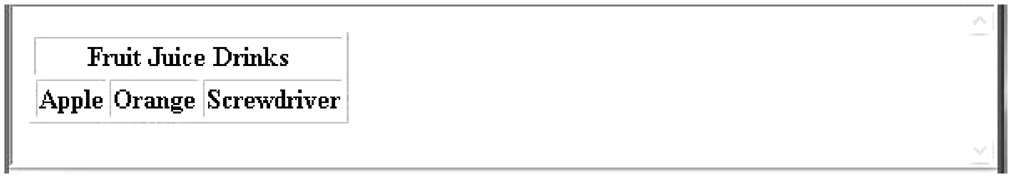


	Fruit Juice Drinks		
	Apple	**Orange**	**Screwdriver**
Breakfast	0	1	0
Lunch	1	0	0
Dinner	0	0	1

Figure 2.19 Display of `table.html`

2.8.2 The `rowspan` and `colspan` Attributes

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 `rowspan` and `colspan` attributes.



Fruit Juice Drinks		
Apple	**Orange**	**Screwdriver**

Figure 2.20 Two levels of column labels

The `colspan` attribute 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 `colspan` attribute specifies, the browser stretches the spanning cell over the number of cells that populate the column in the table.[10] The `rowspan` attribute of the table heading and table data tags does for rows what `colspan` does for columns.

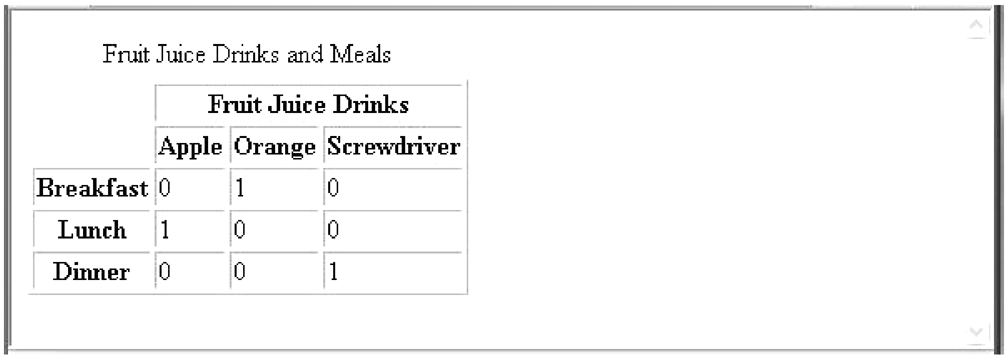
A table that has two levels of column labels and also has row labels must have an empty upper-left corner cell that spans both the multiple rows of column labels and the multiple columns. Such a cell is specified by including both `rowspan` and `colspan` attributes. Consider the following table specification, 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>
```

10. Some browsers add empty row cells to allow the specified span to occur.

```
          <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

2.8.3 The `align` and `valign` Attributes

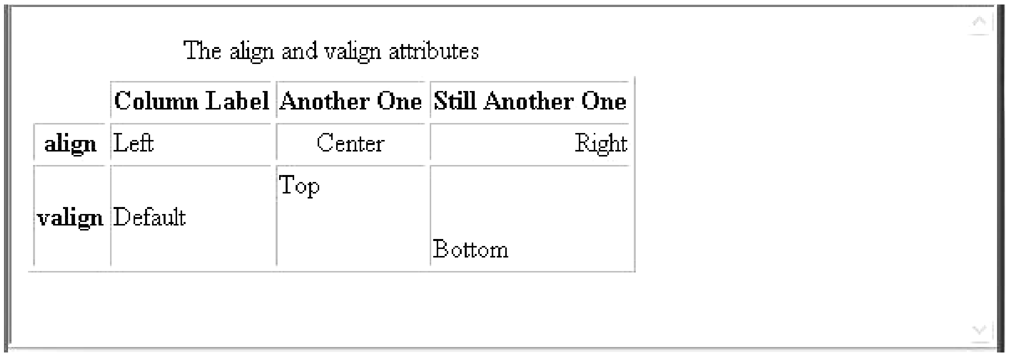
The placement of the content within a table cell can be specified with the `align` and `valign` attributes in the `<tr>`, `<th>`, and `<td>` tags. The `align` attribute has the possible values `left`, `right`, and `center`, with the obvious meanings for horizontal placement of the content within a cell. 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 `valign` attribute of the `<th>` and `<td>` tags has the possible values `top` and `bottom`. The default vertical alignment for both headings and data is `center`. Because `valign` applies only to a single cell, there is never any point in specifying `center`.

The following example illustrates the `align` and `valign` attributes:

```
<?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

2.8.4 The `cellpadding` and `cellspacing` Attributes

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 `cellpadding` attribute 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 `cellspacing` attribute is used to specify the distance between cells in a table.

The following document, `space_pad.html`, illustrates the `cellpadding` and `cellspacing` attributes:

```
<?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`.

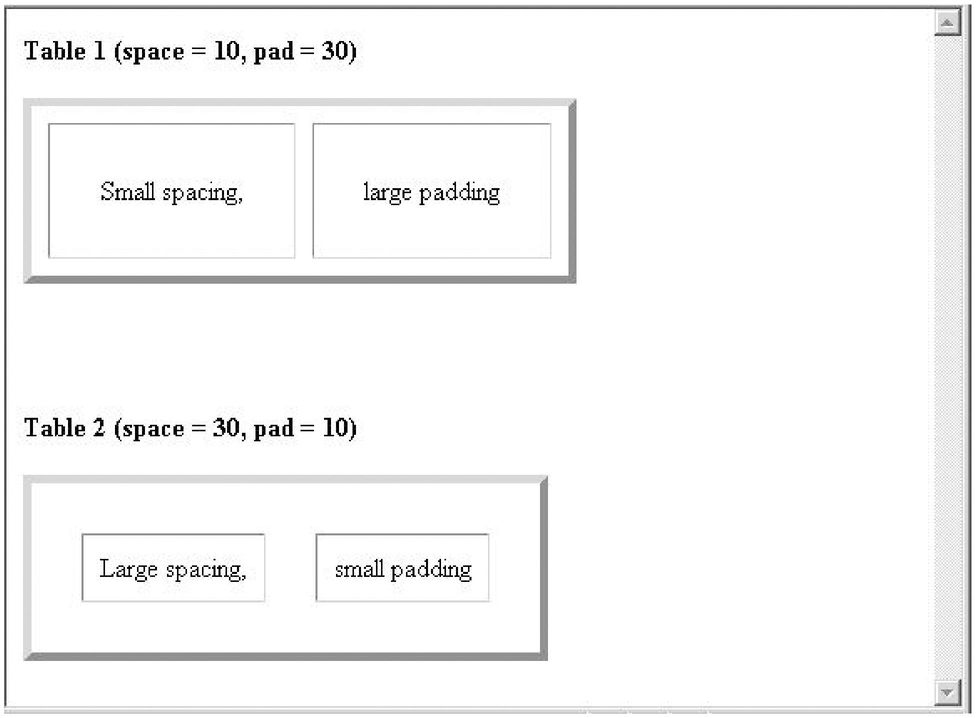


Table 1 (space = 10, pad = 30)

Small spacing,	large padding

Table 2 (space = 30, pad = 10)

Large spacing,	small padding

Figure 2.23 Display of `space_pad.html`

2.8.5 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.

2.9 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).

2.9.1 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 `action` attribute 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 `action` attributes will be the empty string (`""`).

The `method` attribute of `<form>` specifies one of the two techniques, `get` or `post`, used to pass the form data to the server. The default is `get`, so if no `method` attribute is given in the `<form>` tag, `get` will 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 `get` method 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 `get` method 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, `get` is not a good choice.

When the `post` method 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 `post` method, 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 `get` that are not a problem with `post`.

2.9.2 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 `name` attribute, which becomes the name of the value of the control within the form data. The controls for checkboxes and radio buttons require a `value` attribute, 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 `size` attribute 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 `maxlength` attribute 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 `maxlength` attribute:

```
<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 `Name` text box would be as follows:

```
Alfred Paul von Frickenbu
```

No matter what was typed after the `u` in that person's last name, the value of `Name` would 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:

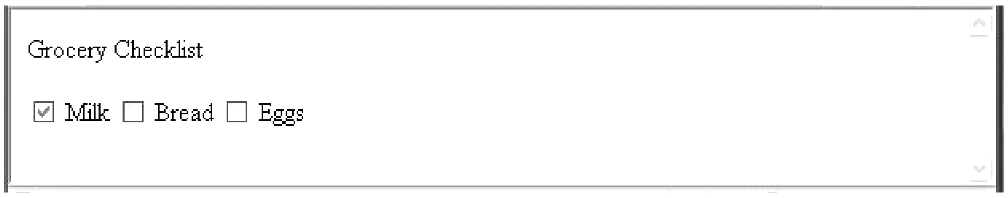
```
<label> Phone: <input type = "text"  name = "phone" />
</label>
```

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 `value` attribute. A checkbox button does not contribute to the form data if it is off. Every checkbox button requires a `name` attribute and a `value` attribute 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

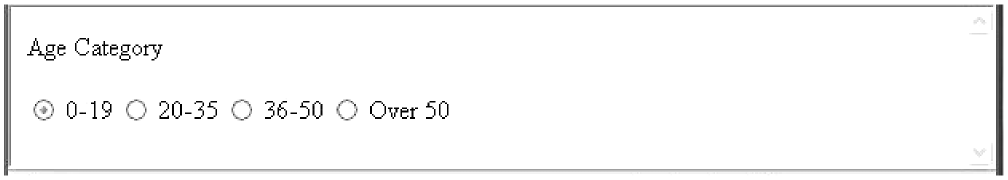
Figure 2.24 Display of `checkbox.html`

If the user does not turn on any of the checkbox buttons in our example, `milk` will be the value for `groceries` in the form data. If the `milk` checkbox is left on and the `eggs` checkbox is also turned on by the user, the values of `groceries` in the form data would be `milk` and `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 `type` value for radio buttons is `radio`. All radio buttons in a group must have the `name` attribute 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 `checked` attribute, 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 checks the 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`

2.9.3 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 `multiple` attribute, set to the value `"multiple"`. When only one menu item is selected, the value sent in the form data is the value of the `name` attribute 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 `name` attribute, of course, is required in the `<select>` tag.

The `size` attribute, specifying the number of menu items that are to be displayed for the user, can be included in the `<select>` tag. If no `size` attribute is specified, the value `1` is used. If the value for the `size` attribute is `1` and `multiple` is 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 `multiple` is specified or the `size` attribute 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 `selected` is `"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.html` after clicking the scroll arrow. Figure 2.28 shows a browser display of `menu.html` after modification to set `size` to `"2."`

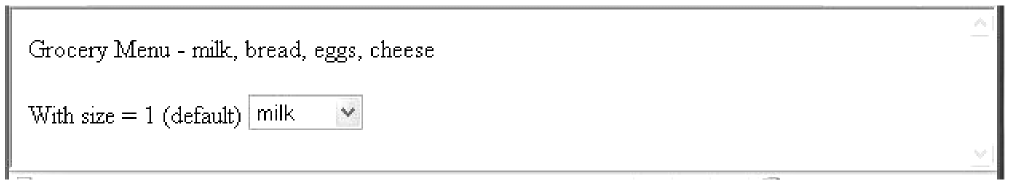


Figure 2.26 Display of `menu.html` (default `size` of `1`)

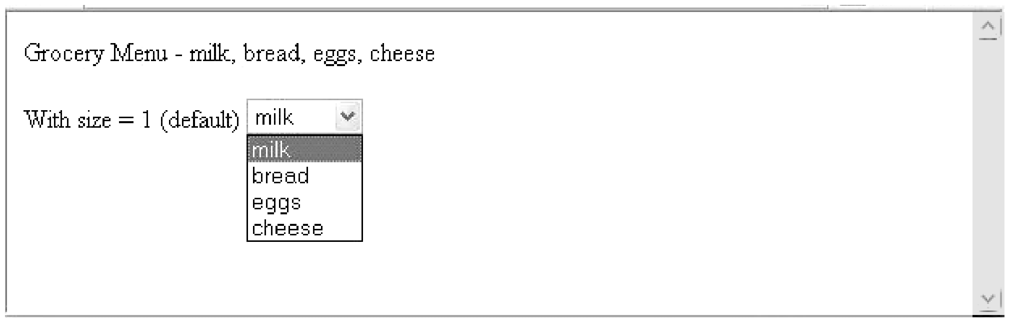


Figure 2.27 Display of `menu.html` after the scroll arrow is clicked

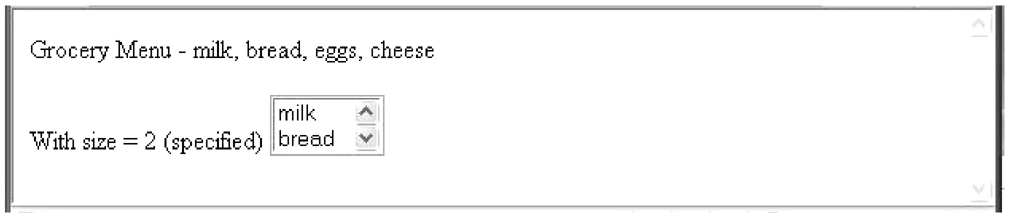

Grocery Menu - milk, bread, eggs, cheese
With size = 2 (specified)
milk
bread

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

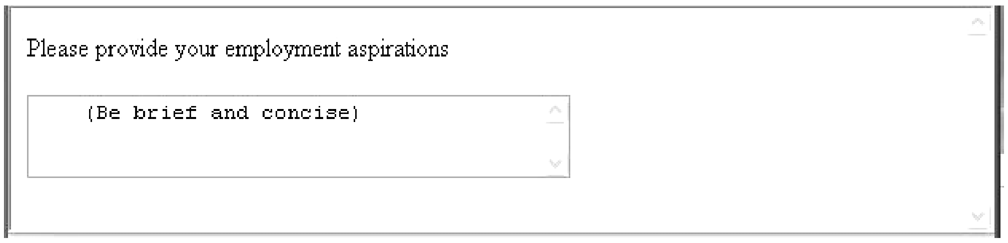
When the `multiple` attribute of the `<select>` tag is set, adjacent options can be chosen by dragging the mouse cursor over them while the left mouse button is held down. Nonadjacent options can be selected by clicking them while holding down the keyboard *Control* key.

2.9.4 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 `rows` and `cols` attributes 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.html` after 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

2.9.5 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 `action` attribute 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>
</form>
```

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

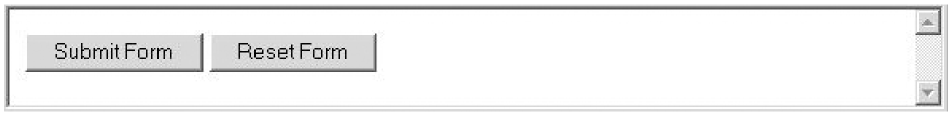


Figure 2.30 *Submit* and *Reset* buttons

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

2.9.6 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.

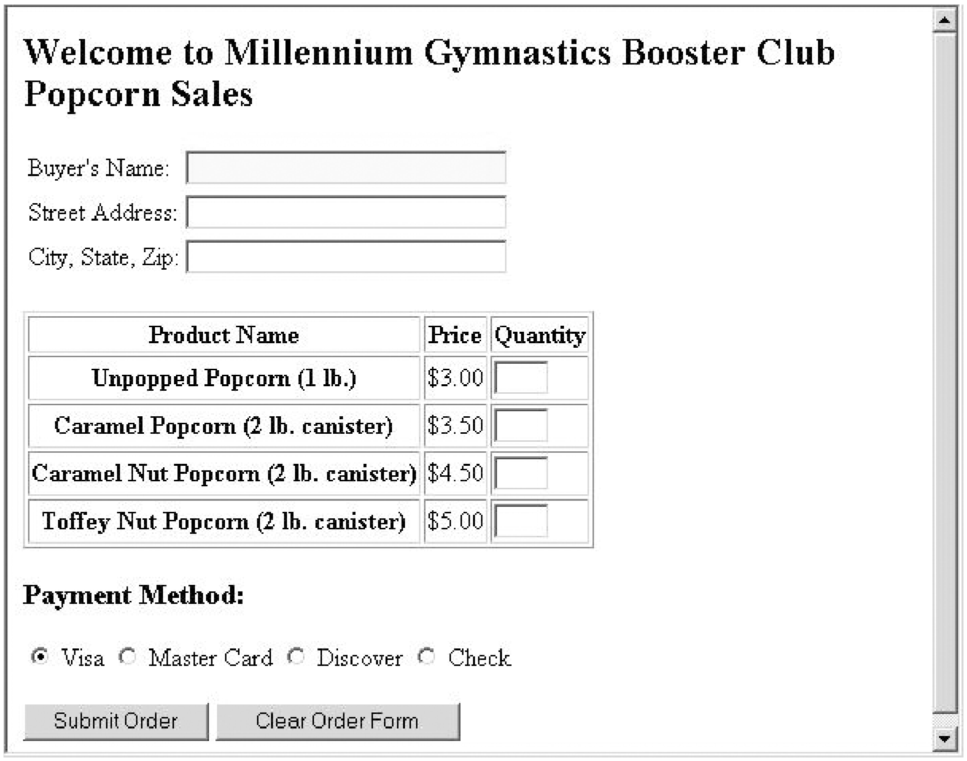
Tables present special problems for the visually impaired. The best solution is to use style sheets (see Chapter 3) instead of tables to lay out tabular information.

```
<?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.

2.10 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 `border` attribute 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 `name` attribute for elements. This attribute was deprecated for some elements in HTML 4.0, which added the `id` attribute to nearly all elements. In XHTML, the use of `id` is encouraged and the use of `name` is discouraged. In fact, the `name` attribute was removed for the anchor and map elements in XHTML 1.1. However, form elements must still use the `name` attribute 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 `<hx>` 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 `alt` attribute 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 `colspan` and `rowspan` attributes, which can appear in both `<th>` and `<td>` tags, provide the means of creating multiple levels of column and row labels, respectively. The `align` and `valign` attributes 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.

Review Questions

2.1 What does it mean for a tag or attribute of XHTML to be deprecated?

2.2 What is the form of an XHTML comment?

2.3 How does a browser treat line breaks in text that is to be displayed?

2.4 What is the difference in the effect of a paragraph tag and a break tag?

2.5 Which heading tags use fonts that are smaller than the normal text font size?

2.6 How do browsers usually set block quotations differently from normal text?

2.7 What does the `<code>` tag specify for its content?

2.8 What are the differences between the JPEG and GIF image formats?

2.9 What are the two required attributes of an `<img />` tag?

2.10 What is the purpose of the `alt` attribute of `<img />`?

2.11 What tag is used to define a link?

2.12 What attribute is required in all anchor tags?

2.13 Does XHTML allow nested links?

2.14 How is the target of a link usually identified in a case where the target is in the currently displayed document but not at its beginning?

2.15 What is the form of the value of the `href` attribute in an anchor tag when the target is a fragment of a document other than the one in which the link appears?

2.16 What is the default bullet form for the items in an unordered list?

2.17 What are the default sequence values for the items in an ordered list?

2.18 What tags are used to define the terms and their definitions in a definition list?

2.19 What is specified when the `border` attribute of a `<table>` tag is set to `"border"`?

2.20 What is the purpose of the `colspan` attribute of the `<th>` tag?

2.21 What is the purpose of the `rowspan` attribute of the `<td>` tag?

2.22 What are the `align` and `valign` attributes of the `<tr>`, `<th>`, and `<td>` tags used for?

2.23 What is the difference between the `cellspacing` and `cellpadding` attributes?

2.24 What are controls?

2.25 Which controls discussed in this chapter are created with the `<input>` tag?

2.26 What is the default size of a text control's text box?

2.27 What is the difference between the `size` and `maxlength` attributes of `<input>` for text controls?

2.28 What is the difference in behavior between a group of checkbox buttons and a group of radio buttons?

2.29 Under what circumstances is a menu used instead of a radio button group?

2.30 What is the drawback of specifying the `multiple` attribute with a menu?

2.31 How are scroll bars specified for text-area controls?

Exercises

2.1 Create, test, and validate an XHTML document for yourself, including your name, address, and e-mail address. If you are a student, you must include your major and your grade level. If you work, you must include your employer, your employer's address, and your job title. This document must use several headings and `<em>`, `<strong>`, `<hr />`, `<p>`, and `<br />` tags.

2.2 Add pictures of yourself and at least one other image (of your friend, spouse, or pet) to the document created for Exercise 2.1.

2.3 Add a second document to the document created for Exercise 2.1 that describes part of your background, using `background` as the link content. This document should have a few paragraphs of your personal or professional history.

2.4 Create, test, and validate an XHTML document that describes an unordered list equivalent to your typical grocery shopping list. (If you've never written a grocery list, use your imagination.)

2.5 Create, test, and validate an XHTML document that describes an unordered list of at least four states. Each element of the list must have a nested list of at least three cities in the state.

2.6 Create, test, and validate an XHTML document that describes an ordered list of your five favorite movies.

2.7 Modify the list of Exercise 2.6 to add nested, unordered lists of at least two actors and/or actresses in your favorite movies.

2.8 Create, test, and validate an XHTML document that describes an ordered list with the following contents: The highest level should be the names of your parents, with your mother first. Under each parent, you must have a nested, ordered list of the brothers and sisters of your parents, in order by age, eldest first. Each of the nested lists in turn must have nested lists that list the children of your uncles and aunts (your cousins)—under the proper parents, of course. Regardless of how many aunts, uncles, and cousins you actually have, there must be at least three list items in each sublist below each of your parents and below each of your aunts and uncles.

2.9 Create, test, and validate an XHTML document that describes a table with the following contents: The columns of the table must have the headings "Pine", "Maple", "Oak", and "Fir". The rows must have the labels "Average Height", "Average Width", "Typical Life Span", and "Leaf Type". You can make up the data cell values.

2.10 Modify, test, and validate an XHTML document from Exercise 2.9 that adds a second-level column label, "Tree", and a second-level row label, "Characteristics".

2.11 Create, test, and validate an XHTML document that defines a table with columns for state, state bird, state flower, and state tree. There must be at least five rows for states in the table. You must include attribute specifications for `cellpadding` and `cellspacing`.

2.12 Create, test, and validate an XHTML document that defines a table with two levels of column labels: an overall label, "Meals", and three secondary labels, "Breakfast", "Lunch", and "Dinner". There must be two levels of row labels: an overall label, "Foods", and four secondary labels, "Bread", "Main Course", "Vegetable", and "Dessert". The cells of the table must contain a number of grams for each of the food categories.

2.13 Create, test, and validate an XHTML document that is the home page of a business, Tree Branches, Unlimited, that sells tree branches. This document must include images and descriptions of at least three different kinds of tree branches. There must be at least one unordered list, one ordered list, and one table. Detailed descriptions of the different branches must be stored in separate documents that are accessible through links from the home document. You must invent several practical uses for tree branches and include sales pitches for them.

2.14 Create, test, and validate an XHTML document that has a form with the following controls:

a. A text box to collect the user's name

b. Four checkboxes, one each for the following items:

   i. Four 100-watt light bulbs for $2.39

   ii. Eight 100-watt light bulbs for $4.29

   iii. Four 100-watt, long-life light bulbs for $3.95

   iv. Eight 100-watt, long-life light bulbs for $7.49

c. A collection of three radio buttons that are labeled as follows:

   i. Visa

   ii. Master Card

   iii. Discover