[Figures are not included in this sample chapter]

Platinum Edition Using HTML 4, XML, and Java 1.2
- 3 -
HTML 4.0 Tag Reference

by Eric Ladd

In this chapter

Reference Scope

This chapter is unique in the book because it is written to serve as a reference for all the tags included in the HTML 4.0 recommendation, as published by the World Wide Web Consortium (W3C). It is a one-stop catalog of each tag, including the tag's attributes, syntax, and examples of uses. By necessity, the chapter covers a large amount of information, but you'll soon come to appreciate the value of having all the relevant facts about all HTML tags--together with tips on how to use them--right at your fingertips.

NOTE: This chapter covers only the tags included in the recommended HTML 4.0 Document Type Definition (DTD), as published by the World Wide Web Consortium in April 1998. Browser-specific extensions to HTML 4.0 are beyond the scope of this chapter, but may be covered elsewhere in the book. The <LAYER> tag introduced by Netscape Communications Corporation, for example, is discussed in detail in Chapter 24, "Introduction to Dynamic HTML," and Chapter 25, "Advanced Netscape Dynamic HTML."
For the most up-to-date status of HTML 4.0, consult http://www.w3.org/TR/PR-html40/, where you will find links to the most current version of the standard and the version just prior to that.

How This Chapter Is Organized

Because of the vast coverage of the chapter, the information presented has been carefully structured to make it as easy as possible for you to look up the tags you need. At the highest level, the chapter is organized into major sections that cover a group of related tags. The major sections and the tags they cover include

- Document structure tags (see p. **62**): <HTML>, <HEAD>, <BASE>, <ISINDEX>, <META>, <LINK>, <SCRIPT>, <STYLE>, <TITLE>, <BDO>, and <BODY>.
- Formatting tags (see p. **72**): <B>, <BASEFONT>, <BIG>, <FONT>, <I>, <S>, <STRIKE>, <SMALL>, <SUB>, <SUP>, <TT>, <U>, <ABBR>, <ACRONYM>, <ADDRESS>, <CITE>, <CODE>, <DEL>, <DFN>, <EM>, <INS>, <KBD>, <Q>, <SAMP>, <STRONG>, <VAR>, <BLOCKQUOTE>,
, <CENTER>, <DIV>, <HR>, <H1>-<H6>, <P>, <PRE>, and <SPAN>.
- List tags (see p. **95**): <LI>, <DIR>, <DL>, <DT>, <DD>, <MENU>, <OL>, and </UL>.
- Hyperlink tags: <A>.
- Image and imagemap tags (see p. **104**): <IMG>, <MAP>, and <AREA>.
- Table tags (see p. **109**): <TABLE>, <CAPTION>, <THEAD>, <TFOOT>, <TBODY>, <COLGROUP>, <COL>, <TR>, <TD>, and <TH>.
- Form tags (see p. **119**): <FORM>, <INPUT>, <SELECT>, <OPTION>, <OPTGROUP>, <TEXTAREA>, <BUTTON>, <LABEL>, <FIELDSET>, and <LEGEND>.
- Frame tags (see p. **129**): <FRAMESET>, <FRAME>, <NOFRAMES>, and <IFRAME>.
- Executable content tags (see p. **133**): <APPLET>, <PARAM>, and <OBJECT>.

> **TIP:** In some cases, tags covered in this chapter get a more thorough treatment in a later chapter of the book. Look for cross-references to point you to this expanded coverage.

Within a given section, several tags are discussed in detail. Specifically, you'll find the following information about each tag:

- The tag's keyword--For example, the <INPUT> tag's keyword is INPUT.
- What kind of tag it is--Every HTML tag is either a *container tag* or a *standalone tag*. A container tag is one that activates an effect and that has a companion tag that discontinues the effect. For example, <I> is a container tag that, together with its companion closing tag </I>, causes all text found between them to be rendered in italic. The <I> tag turns on the italic effect and the </I> tag turns it off.

- A standalone tag is one that does not have a companion tag. For example, the <IMG> tag simply places an image on a page. <IMG> has no effect that was turned on and needs to be turned off, so no closing tag is needed.

  NOTE: Standalone tags are sometimes called empty tags.

- The tag's function--A description of the effect or page element that the tag controls.
- The tag's syntax--HTML is derived from the Standard Generalized Markup Language (SGML) by applying SGML constructs according to a certain set of rules. These rules define a tag's syntax.
- The tag's attributes--An attribute modifies how a tag's effect is applied. Some tags take no attributes, and others may be able to take several. Additionally, attributes can sometimes take on only one of a set number of values. In these cases, the possible values of the attribute are listed along with the attribute. Use of some attributes may be required (such as the SRC attribute for the <IMG> tag), and use of others may be optional. A tag's required attributes, if any, are noted in each attribute discussion.
- Example usage--You can learn more about how a tag is used by looking over the sample code given in the tag description.
- Related tags--Some tags work in conjunction with others to produce an effect. In these cases, you'll find a listing of the other HTML tags related to the one being described. Often, you'll find that the related tags are discussed in the same section.

Within a section, tags are listed alphabetically by keyword, unless they need to be used in a certain order, in which case, they are presented in the order that they are typically used.

Global Attributes

Although most tag attributes tend to be unique to the tag, some are almost universal and usable with any tag. Table 3.1 summarizes these attributes, showing which tags do take the attributes and how each attribute is used.

TABLE 3.1 Global HTML Attributes

Attribute	Purpose	Used With
CLASS	Space-separated list of classes of the tag	All tags but <BASE>, <BASEFONT>, <HEAD>, <HTML>, <META>, <PARAM>, <SCRIPT>, <STYLE>, and <TITLE>.
DIR	Direction for weak or neutral text	All tags but <APPLET>, <BASE>, <BASEFONT>, <BDO>, , <FRAME>, <FRAMESET>, <HR>, <IFRAME>, <PARAM>, and <SCRIPT>.
ID	Unique, document-wide identifier	All tags but <BASE>, <HEAD>, <HTML>, <META>, <SCRIPT>, <STYLE>, and <TITLE>.
LANG	Specifies document	All tags but <APPLET>, <BASE>, <BASEFONT>, ,

	language context	<FRAME>, <FRAMESET>, <HR>, <IFRAME>, <PARAM>, and <SCRIPT>.
STYLE	Binds style information to the tag	All tags but <BASE>, <BASEFONT>, <HEAD>, <HTML>, <META>, <PARAM>, <SCRIPT>, <STYLE>, and <TITLE>.
TITLE	Advisory title	All tags but <BASE>, <BASEFONT>, <HEAD>, <HTML>, <META>, <PARAM>, <SCRIPT>, <STYLE>, and <TITLE>.

The global attribute you'll probably use most often is the STYLE attribute, which is used to assign style information to a tag. To color a level 2 heading red, for example, you could use the HTML:

```
<H2 STYLE="color: red">Red Heading</H2>
```

The ID attribute is also useful when you need to have a unique identifier for a tag. This situation comes into play when you write scripts to support dynamic HTML documents because you frequently want to change the properties of some marked-up text. To do this, you need to be able to address the tag that marks up the text via JavaScript, JScript, or VBScript, and the best way to do that is to give the tag a unique name. Then it becomes fairly simple to address the tag via the browser's object model.

SEE "Introduction to Dynamic HTML," **p. 581**.

LANG can be helpful in situations where you are marking up content in multiple languages. The value of LANG gives browsers a "heads-up" as to what language is being used. LANG is usually set equal to a two-character language code that denotes the language being used. For example, "fr" denotes French; "de" denotes German, and so on. In cases where variants on a language exist, you'll see expanded language codes, such as "en-US" for English spoken in the United States or "en-Br" for English spoken in Britain.

DIR refers to the directionality--left-to-right or right-to-left--of text when it cannot otherwise be deduced from the context of the document. DIR can take on values of LTR (left-to-right) or RTL (right-to-left).

The TITLE attribute enables you to specify descriptive text to associate with the tag. This information may be helpful to nonvisual browsers, such as those that generate speech or Braille output.

Finally, the CLASS attribute enables you to create different classes of the same tag. For example, you might have:

```
<A HREF="xrefs.html" CLASS="cross-reference"> ... </A>
<A HREF="defns.html" CLASS="definition"> ... </A>

<A HREF="biblio.html" CLASS="bibliography"> ... </A>
```

This creates three classes of the <A> tag. After these classes are established, you can reference them elsewhere in your document. One popular application of this is in a style sheet:

```
A.cross-reference {color: navy}
A.definition {color: yellow}
```

```
A.bibliography {color: fuschia}
```

The style information above would color cross-reference links navy blue, definition links yellow, and bibliography links fuschia.

Event Handlers

The HTML 4.0 recommendation also allows for several event handlers that can be used to trigger the execution of script code embedded in an HTML document. Each event handler is tied to a specific event that can occur during a person's use of a browser. When a user submits a form, for example, you can capture that event and launch a field validation script using the onsubmit event handler:

```
<FORM ACTION="register.cgi" METHOD="POST" onsubmit="validate();">
```

> For the specifics on writing scripts for your HTML documents, **see** "Introduction to JavaScripting," **p. 439** or "Active Server Pages and VBScript," **p. 835**.

Thus, when a user clicks the Submit button, the scripted function named "validate" fires and checks the data the user is submitting for appropriate formatting, content, or other validation checks.

Table 3.2 details the event handlers available under HTML 4.0. Most can be used within any HTML element, but a few are limited to specific elements. These special cases are noted in the table.

TABLE 3.2 HTML 4.0 Event Handlers

Event Handler	Triggered when...
`onload`	A document or frameset is loaded; only allowed in the <BODY> and <FRAMESET> elements
`onunload`	A document or frameset is unloaded; only allowed in the <BODY> and <FRAMESET> elements
`onclick`	The mouse button is clicked once
`ondblclick`	The mouse button is clicked twice
`onmousedown`	The mouse button is depressed
`onmouseup`	The mouse button is released
`onmouseover`	The mouse pointer is over a page element
`onmousemove`	The mouse pointer is moved while over a page element
`onmouseout`	The mouse pointer is moved off a page element
`onfocus`	A form field receives focus by tabbing to it or by clicking it with the mouse pointer;

	only allowed in the <INPUT>, <SELECT>, <TEXTAREA>, <LABEL>, and <BUTTON> elements
onblur	A form field loses focus by tabbing out of it or by clicking a different field with the mouse pointer; only allowed in the <INPUT>, <SELECT>, <TEXTAREA>, <LABEL>, and <BUTTON> elements
onkeypress	A key is pressed and released over a page element
onkeydown	A key is pressed over a page element
onkeyup	A key is released over a page element
onsubmit	A form is submitted; only allowed in the <FORM> tag
onreset	A form is reset; only allowed in the <FORM> tag
onselect	A user selects some text in a text field; only allowed in the <INPUT> and <TEXTAREA> elements
onchange	A form field loses focus and its value has changed since gaining focus; only allowed in the <INPUT>, <TEXTAREA>, and <SELECT> elements

Document Structure Tags

Every HTML document has three major components: the HTML declaration, the head, and the body. The document structure tags are those that define each component.

<HTML>

Type:

Container

Function:

Declares the document to be an HTML document. All document content and supporting HTML code goes between the <HTML> and </HTML> tags.

Syntax:

```
<HTML> ... </HTML>
```

Attributes:

Technically speaking, the <HTML> tag can take the VERSION attribute, but this has been deprecated in favor of version information being specified in the <!DOCTYPE> tag (see "Related Tags" below).

Example:

```
<HTML>
```

```
... all content and HTML code goes here ...

</HTML>
```

Related Tags:

Although the <HTML> tag is typically the first tag in a document, it is sometimes preceded by a <!DOCTYPE> tag that specifies what level of HTML conformance the document displays. A document conforming to the HTML 4.0 standard might have a <!DOCTYPE> tag that reads:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
```

Technically, <!DOCTYPE> is an SGML tag, not an HTML tag, so it is acceptable for it to be outside the <HTML> and </HTML> tags.

<HEAD>

Type:

Container

Function:

Contains the tags that compose the document head.

Syntax:

```
<HEAD> ... </HEAD>
```

Attributes:

<HEAD> can take the PROFILE attribute, which gets set equal to a space-separated list of URLs that point to meta data profiles for the document.

Example:

```
<HTML>
<HEAD PROFILE="http://www.server.com/profiles/">
... tags making up the document head go here ...
</HEAD>
... all other content and HTML code goes here ...

</HTML>
```

Related Tags:

A number of tags can be placed between the <HEAD> and </HEAD> tags, including <BASE>, <ISINDEX>, <LINK>, <META>, <SCRIPT>, <STYLE>, and <TITLE>. Each of these is described next.

<BASE>

Type:

Standalone

Function:

Declares global reference values for the HREF and TARGET attributes. The reference or base HREF value is used as a basis for computing all relative URL references. The base TARGET name is used to identify the frame into which all linked documents should be loaded.

Syntax:

```
<BASE HREF="base_url">
```

or

```
<BASE TARGET="frame_name">
```

Attributes:

The <BASE> tag takes either the HREF or the TARGET attribute. A given <BASE> tag can contain only one of these, so if you need to specify a base URL and a base target frame, you need to have two <BASE> tags in the head of your document. These two attributes work as follows:

- HREF--Specifies the reference URL that is used to help compute relative URLs. If the BASE HREF URL is http://www.myserver.com/sports/hockey/skates.html and you use the relative URL pucks.html elsewhere in the document, for example, the relative URL will really point to http://www.myserver.com/sports/hockey/pucks.html.
- TARGET--Specifies the default frame name to which all links are targeted.

NOTE: When used in a <BASE> tag, HREF is typically set to the URL of the document.

Example:

```
<HEAD>
<BASE HREF="http://www.myserver.com/index.html">
<BASE TARGET="bigframe">
...

</HEAD>
```

This code sets the document's base URL to http://www.myserver.com/index.html and the base frame for targeting hyperlinks to the frame named "bigframe".

<ISINDEX>

Type:

Standalone

Function:

Produces a single-line input field used to collect query information.

Syntax:

```
<ISINDEX PROMPT="Please enter the value to search for.">
```

Attributes:

The PROMPT attribute specifies what text should appear before the input field. In the absence of a PROMPT attribute, the text will read "This is a searchable index. Enter search criteria:"

Example:

```
<HEAD>
<ISINDEX PROMPT="Enter the last name of the employee you want to search for:">
...

</HEAD>
```

> **NOTE:** <ISINDEX> was used in the early days when the <FORM> tags had yet to come onto the scene. The W3C has deprecated the <ISINDEX> tag, meaning that it discourages its use in favor of using the <FORM> tags and it expects to drop the tag from the standard in the future.

<META>

Type:

Standalone

Function:

Defines document meta-information, such as keywords, expiration date, author, page generation software used, and many other document-specific items. It also supports the notion of *client pull*--a dynamic document technique in which the browser loads a new document after a specified delay.

Syntax:

```
<META HTTP-EQUIV="header" CONTENT="value">
or
<META NAME="name" CONTENT="value">
```

Attributes:

The <META> tag takes the following attributes:

- HTTP-EQUIV--Specifies a type of HTTP header to be sent with the document. The value of the header is given by the CONTENT attribute. The two most commonly used values of HTTP-EQUIV are REFRESH, which refreshes the page after a specified delay, and EXPIRES, which gives the date after which content in the document is not considered to be reliable.

- NAME--Set equal to the name of the document meta-variable you want to specify. The value of the variable is given in the CONTENT attribute. Typical values for NAME include AUTHOR, KEYWORDS, GENERATOR, and DESCRIPTION. The KEYWORDS value is particularly useful for specifying words you would like a search engine's indexing program to associate with the page.

- SCHEME--Provides information on how to interpret the meta-variable. For example, with the following <META> tag:

  ```
  <META SCHEME="9-digit-ZipCode" NAME="zip" CONTENT="02134-1078">
  ```

- a browser may not know how to interpret "02134-1078" without information from the SCHEME attribute.

- CONTENT--Specifies either the HTTP header or the value of the meta-variable.

Example:

```
<HEAD>
<!-- The first <META> tag instructs the browser to load a new page after 5 seconds.
<!-- This is useful for creating a splash screen effect. -->
<META HTTP-EQUIV="Refresh" CONTENT="5; URL=http://www.myserver.com/index2.html">
<!-- The remaining <META> tags specify author and keyword information. -->
<META NAME="AUTHOR" CONTENT="Eric Ladd">
<META NAME="KEYWORDS" CONTENT="Main page, welcome, neat stuff">
...

</HEAD>
```

<LINK>

Type:

Standalone

Function:

Denotes the linking relationship between two files.

Syntax:

```
<LINK HREF="url_of_linked_file" TITLE="title" REL="forward_relationship" REV="rever
```

Attributes:

The <LINK> tag takes the following attributes:

- CHARSET--Denotes which character encoding scheme to use.
- HREF--Set equal to the URL of the file to which you're making the linking reference.
- HREFLANG--Specifies the language code for the linked file.
- MEDIA--Provides the intended display destination for the linked document. The default value of MEDIA is "screen."
- TARGET--Specifies which frame to target.
- TITLE--Gives the link a descriptive title.
- REL--Specifies the relationship of the linked file to the current file.
- REV--Specifies how the current file relates to the linked file.

Table 3.3 shows some possible values for REL and REV and what these values mean.

TABLE 3.3 Possible Values for the REL and REV Attributes

Value	Meaning
Copyright	Web site's copyright page
Glossary	Glossary of terms for a site
Help	Site help page
Home	Site home page
Index	Site index page
Made	Mail to URL pointing to the email address of the page author
Next	Page that logically follows the current page
Previous	Page that precedes the current page
Stylesheet	File containing style information for the page
TOC	Site table of contents
Up	Page that is above the current page in a site's hierarchy

NOTE: Because so many types of linked files exist, it is permissible to have more than one <LINK> tag in a document.

Example:

```
<HEAD>
<LINK HREF="/style/styles.css" REL="Stylesheet">
<LINK HREF="/index.html" REL="Home">
<LINK HREF="/help.html" REL="Help">
```

```
<LINK HREF="back_one.html" REV="Previous">
...

</HEAD>
```

<SCRIPT>

Type:

Container

Function:

Contains script code referenced in the body of the document.

Syntax:

```
<SCRIPT LANGUAGE="scripting_language">
... script code goes here ...

</SCRIPT>
```

Attributes:

The <SCRIPT> tag can take the following attributes:

- CHARSET--Denotes which character encoding scheme to use.
- DEFER--Specifying the DEFER attribute tells the browser that the script does not generate any document content. This enables the browser to continue parsing and rendering the document without having to execute the script.
- LANGUAGE--Set equal to the scripting language used to write the script. LANGUAGE is being deprecated in favor of using the TYPE attribute.
- SRC--Specifies the URL of a file containing the script code, if not contained between the <SCRIPT> and </SCRIPT> tags.
- TYPE--Set equal to the MIME type of the script code, usually text/javascript or text/vbscript. When specifying a specific version of a scripting language, you can set TYPE equal to a value that includes version information as well (for example, TYPE="text/javascript1.1"). TYPE is a required attribute under HTML 4.0.

```
<SCRIPT LANGUAGE="VBScript">
<!--
Sub ScriptEx
document.write("<HR>")
document.write("<H1 ALIGN=CENTER>Thank you for your submission!</H1>")
document.write("<HR>")
-->

</SCRIPT>
```

NOTE: Script code is often placed between <!-- and --> tags so that browsers that can't process scripts will treat the code as a comment.

Related Tags:

You can use the <NOSCRIPT> tag to specify what a browser should do if it is unable to execute a script contained in the <SCRIPT> and </SCRIPT> tags.

<NOSCRIPT>

Type:

Container

Function:

Provides alternate content to use if a script cannot be executed. A browser might not be able to execute a script because the user has turned scripting off or because it does not know the scripting language used to write the script.

Syntax:

```
<NOSCRIPT>
... alternative to script code goes here ...

</NOSCRIPT>
```

Attributes:

None.

Example:

```
<SCRIPT LANGUAGE="VBScript">
   document.write("Hello, World!");
</SCRIPT>
<NOSCRIPT>
   You either have scripting turned off or your browser does not
   understand VBScript.

</NOSCRIPT>
```

<STYLE>

Type:

Container

Function:

Specifies style information for the document.

Syntax:

```
<STYLE TYPE="mime_type" MEDIA="media_type" TITLE="title">
... style information goes here ...

</HTML>
```

Attributes:

The <STYLE> tag takes the following three attributes:

- MEDIA--Specifies what media types the styles are to be used for (visual browser, speech-based browser, Braille browser, and so on).
- TITLE--Gives the style information a descriptive title.
- TYPE--Set equal to the Internet content type for the style language. You will most likely say TYPE="text/css1" to denote the use of the style language put forward in the Cascading Style Sheets, Level 1 specification. TYPE is a required attribute of the <STYLE> tag.

Example:

```
<STYLE TYPE="text/css1">
<!--
   BODY {font: 10 pt Palatino; color: silver margin-left: 0.25 in}
   H1 {font: 18 pt Palatino; font-weight: bold}
   H2 {font: 16 pt Palatino; font-weight: bold}
   P {font: 12 pt Arial; line-height: 14 pt; text-indent: 0.25 in}
-->

</STYLE>
```

> **NOTE:** Style information is usually contained between <!-- and --> tags so that browsers that cannot process it will treat the style information as a comment.

<TITLE>

Type:

Container

Function:

Gives a descriptive title to a document. Use of the <TITLE> tag is required by the HTML 4.0 DTD for many good reasons. Titles show up in browser window title bars and in bookmark and history listings. In each of these cases, you provide an important reader service when you specify a title because otherwise the browser will display just the document's URL. Additionally, Web search engines, such as Yahoo! and AltaVista, frequently look for title information when they index a

document.

Syntax:

```
<TITLE> ... document title goes here ... </TITLE>
```

Attributes:

None.

Example:

```
<TITLE>
The Advantages of a Corporate Web Site

</TITLE>
```

> **TIP:** Try to keep titles to 40 characters or fewer so that browsers can display them completely.

<BDO>

Type:

Container

Function:

When mixing languages in an HTML document, it sometimes becomes necessary to be sensitive to the direction in which the language is read (left-to-right versus right-to-left). When languages that have mixed directions are used in a document, an approach called the *bidirectional algorithm* is used to ensure proper presentation of the content. In cases where you want to override the bidirectional algorithm for a block of text, you can enclose that text in the <BDO> and </BDO> tags.

Syntax:

```
<BDO DIR="LTR|RTL"> ... directional text goes here ... </BDO>
```

Attributes:

The <BDO> tag takes the DIR attribute, which can be set to LTR to specify left-to-right directionality or to RTL to specify right-to-left directionality.

Example:

```
<BODY LANG="he" ...> <!-- Hebrew language context – RTL directionality>
... <BDO DIR="LTR">Here's some English text.</BDO> ...
...

</BODY>
```

<BODY>

Type:

Container

Function:

Contains all content and tags that compose the document body.

Syntax:

```
<BODY BGCOLOR="background_color" BACKGROUND="background_image"
   LINK="unvisited_link_color" ALINK="active_link_color"
   VLINK="visited_link_color" TEXT="text_color">
 ... document body goes here ...

</BODY>
```

Attributes:

The <BODY> tag takes the following attributes, which focus on global background and coloring properties. Each color-related attribute can be set equal to one of the 16 reserved color names (BLACK, WHITE, AQUA, SILVER, GRAY, MAROON, RED, PURPLE, FUSCHIA, GREEN, LIME, OLIVE, YELLOW, NAVY, BLUE, and TEAL) or to an RGB hexadecimal triplet.

- ALINK--Set equal to the color you want to paint active links (a link is active in the instant that the user clicks it).
- BACKGROUND--Set equal to the URL of an image to use in the document background. The image will be horizontally and vertically tiled if it is not large enough to fill the entire browser screen.
- BGCOLOR--Set equal to the color you want to paint the document's background.
- LINK--Set equal to the color you want to paint unvisited links. (A link is unvisited if a user has yet to click it.)
- TEXT--Set equal to the color you want to paint the body text of the document.
- VLINK--Set equal to the color you want to paint visited links. (A link is visited if a user has already clicked it.)

NOTE: All the attributes listed have been deprecated in favor of using style sheet characteristics to specify the same information.

Example:

```
<BODY BGCOLOR="white" TEXT="#FF0088" LINK="#DD0F00" VLINK="#00FF9A">
```

```
... all document body content and HTML code goes here ...

</BODY>
```

Related Tags:

Dozens of tags are allowed between the <BODY> and </BODY> tags. In fact, with the exception of some of the frame-related tags, any tag in the rest of the chapter can be placed between <BODY> and </BODY>.

By putting together what you've learned in this section, you can come up with a generic HTML document template such as the following:

```
<HTML>
<HEAD>
<TITLE>Document Template</TITLE>
... <META>, <BASE>, <LINK>, <SCRIPT>, <STYLE>, <ISINDEX> tags ...
</HEAD>
<BODY>
... document body content and tags ...
</BODY>

</HTML>
```

When creating a new document, you can use this code to get started, and then fill in tags and other information according to your needs.

Formatting Tags

HTML provides a host of tags that you can use to change how text is displayed on a browser screen. After all, 12-point Times Roman gets a little tiring after a while, and it's nice to give a reader an occasional break from a sea of ordinary text.

You can apply formatting instructions at two levels within a document. The first is at the text level, which means you are marking up at least a single character, but often much more than that. The second is at the paragraph or block level, which means you are formatting a specific logical chunk of the document. This section looks at both types of markup, starting with text-level formatting.

Text-Level Formatting

Text-level formatting can occur in one of two ways. An HTML tag that formats text can make changes to the font properties of the text (*font formatting* or *physical styles*), or it can describe how the text is being used in the context of the document (*phrase formatting* or *logical styles*). The next two sections introduce you to the tags used for each type of formatting.

Font Formatting

<B>

Type:

Container

Function:

Contains text to be rendered in boldface (see Figure 3.1).

FIGURE 3.1 *Boldface text stands out from the plain text around it, drawing the reader's attention to it.*

Syntax:

```
<B> ... bold text goes here ... </B>
```

Attributes:

None.

Example:

```
<B>First Name:</B> <INPUT TYPE="TEXT" NAME="fname">
```

<BASEFONT>

Type:

Standalone

Function:

Sets base size, color, and typeface properties for the body text font. The <BASEFONT> tag and all its attributes have been deprecated in favor of using style sheets.

Syntax:

```
<BASEFONT SIZE="size" COLOR="color" FACE="list_of_typefaces">
```

Attributes:

<FONT> can take any combination of the following attributes:

- COLOR--Set to any of the 16 reserved, English-language color names or an RGB hexadecimal triplet. The default font color is black.
- FACE--Set to a list of typefaces that the browser should use to render the text. The browser will use the first face in the list if that face is available. If not, it will work through the rest of the list and use the first face it finds available.
- SIZE--Set equal to an integer value between 1 and 7. This number is mapped to a font size in

points by the browser, according to the user's preferences. The default SIZE value is 3.

Example:

```
<BASEFONT SIZE=5 COLOR="navy" FACE="Arial,Helvetica,Times">
```

Related Tags:

The <FONT> tag is typically used if you need to modify any of the base font properties specified in the <BASEFONT> tag. The <FONT> tag has been deprecated as well.

<BIG>

Type:

Container

Function:

Contains text to be rendered in a font size bigger than the default font size (see Figure 3.2).

FIGURE 3.2 *Using the <BIG> tag increases the point size that text is rendered in.*

Syntax:

```
<BIG> ... big text goes here ... </BIG>
```

Attributes:

None.

Example:

```
<BIG>D</BIG>rop <BIG>C</BIG>aps are a nice onscreen effect.
```

Related Tags:

The <SMALL> tag has the opposite effect (see later in this chapter).

<FONT>

Type:

Container

Function:

Contains text whose font properties are to be modified. Like most tags that specify presentation information, <FONT> has been deprecated by the W3C.

Syntax:

```
<FONT SIZE="size" COLOR="color" FACE="list of typefaces">
... text with modified font properties ...

</FONT>
```

Attributes:

Note that the <FONT> tag has the same attributes as the <BASEFONT> tag. <FONT> is used to change font properties from the base values provided in the <BASEFONT> tag or from their default values. SIZE can be set to a value between 1 and 7, or it can be set equal to how much larger or smaller you want the font size to go (-1 for one size smaller, +3 for three sizes larger, and so forth). COLOR and FACE work exactly as they did for the <BASEFONT> tag.

Example:

```
<FONT SIZE=+1 COLOR="red">Warning! Warning!</FONT> Danger, Will Robinson!
```

Related Tags:

<FONT> changes properties specified in the <BASEFONT> tag.

<I>

Type:

Container

Function:

Contains text to be rendered in italic (see Figure 3.3).

FIGURE 3.3 *Italicized text can be used to denote emphasis or the title of something.*

Syntax:

```
<I> ... italicized text goes here ... </I>
```

Attributes:

None.

Example:

```
I just bought the Beatles' <I>Abbey Road</I> on CD.
```

<S>, <STRIKE>

Type:

Container

Function:

Contains text to be marked with a strikethrough character. Both the <S> tags have been deprecated by the W3C. You should use style sheets to render strikethrough text instead.

Syntax:

```
<S> ... strikethrough text goes here ... </S>
or
<STRIKE> ... strikethrough text goes here ... </STRIKE>
```

Attributes:

None.

Example:

```
Content that has been struck from the record will be denoted as

follows: <S>removed content</S>.
```

<SMALL>

Type:

Container

Function:

Contains text to be rendered in a font size smaller than the default font size.

Syntax:

```
<SMALL> ... smaller text goes here ... </SMALL>
```

Attributes:

None.

Example:

```
<SMALL>"Sssssssshhh!"</SMALL>, he whispered in a tiny voice.
```

Related Tags:

The <BIG> tag has the opposite effect (see the <BIG> tag section earlier in the chapter).

<SUB>

Type:

Container

Function:

Contains text to be a subscript to the text that precedes it.

Syntax:

```
<SUB> ... subscript text goes here ... </SUB>
```

Attributes:

None.

Example:

```
a<SUB>1</SUB>, a<SUB>2</SUB>, and a<SUB>3</SUB> are the coefficients of

 the variables x, y, and z.
```

<SUP>

Type:

Container

Function:

Contains text to be rendered as a superscript to the text that precedes it (see Figure 3.4).

FIGURE 3.4 *Superscripts are useful for indicating trademark or copyright information.*

Syntax:

```
<SUP> ... superscript text goes here ... </SUP>
```

Attributes:

None.

Example:

```
x<SUP>2</SUP> + y<SUP>2</SUP> = 1 defines the unit circle.
```

<TT>

Type:

Container

Function:

Contains text to be rendered in a fixed-width font. Typically, this font is Courier or some kind of typewriter font (see Figure 3.5).

FIGURE 3.5 *Typewriter text is good for displaying computer-related content or for varying the fonts used in the document.*

Syntax:

```
<TT> ... text to be in fixed-width font goes here ... <TT>
```

Attributes:

None.

Example:

```
The computer will then display the <TT>Login:</TT> prompt.
```

<U>

Type:

Container

Function:

Contains text to be rendered with an underline. The <U> tag has been deprecated by the W3C. If you need to underline text, you can do so using style sheets. However, keep in mind that a user might confuse your underlined text with hypertext and try to click it. Also, in keeping with general typesetting rules, if you italicize for form or style, you should not underline.

Syntax:

```
<U> ... text to be underlined ... </U>
```

Attributes:

None.

Example:

```
All first-year medical students purchase <U>Gray's Anatomy</U>.
```

Phrase Formatting Recall that phrase formatting indicates the meaning of the text it marks up and not necessarily how the text will be rendered on the browser screen. Nevertheless, text marked with a phrase formatting tag will typically have some kind of special rendering to set it apart from unmarked text.

<ABBR>

Type:

Container

Function:

Contains text that is an abbreviation of something. This is useful information for browsers that are not vision-based because it enables them to treat the abbreviation differently. A speech-based browser, for example, could know to look in an abbreviation table for pronunciation if you marked up "Dr." with the <ABBR> tag. That way, it could say the word "doctor" rather than making the "dr" sound you would get by pronouncing the "d" and the "r" together.

Syntax:

```
<ABBR> ... acronym goes here ... </ABBR>
```

Attributes:

None.

Example:

```
She got her doctorate (<ABBR>PhD</ABBR>) from the University of Virginia.
```

<ACRONYM>

Type:

Container

Function:

Contains text that specifies an acronym. This tag is also useful for nonvisual browsers. The tag might tell a speech-based browser to pronounce the letters in the acronym one at a time, for example, rather than trying to pronounce the acronym as a word.

Syntax:

```
<ACRONYM> ... acronym goes here ... </ACRONYM>
```

Attributes:

None.

Example:

```
Practical Extraction and Reporting Language <ACRONYM>(PERL)</ACRONYM> is

a popular CGI scripting language.
```

<ADDRESS>

Type:

Container

Function:

Contains either a postal or an electronic mail address. Text marked with this tag is typically rendered in italic (see Figure 3.6).

FIGURE 3.6 *Marking up address information provides a logical marker for programs processing the document and a visual marker for those reading the document.*

Syntax:

```
<ADDRESS> ... address goes here ... </ADDRESS>
```

Attributes:

None.

Example:

```
If you have any comments, please send them to

<ADDRESS>webmaster@your-isp.com</ADDRESS>.
```

<CITE>

Type:

Container

Function:

Contains the name of a source from which a passage is cited. The source's name is typically rendered in italic.

Syntax:

```
<CITE> ... citation source goes here ... </CITE>
```

Attributes:

None.

Example:

```
According to the <CITE>HTML 4.0 Recommendation</CITE>, the <FONT> tag

has been deprecated.
```

<CODE>

Type:

Container

Function:

Contains chunks of computer language code. Browsers commonly display text marked with the <CODE> tag in a fixed-width font such as Courier.

Syntax:

```
<CODE> ... code fragment goes here ... </CODE>
```

Attributes:

None.

Example:

```
<CODE>
document.location.href = 'index.html';
return TRUE;

</CODE>
```

<DEL>

Type:

Container

Function:

Contains text that has been deleted from the document. The tag is intended mainly for documents with multiple authors and/or editors who would want to see all the content in an original draft, even though it may have been deleted by a reviewer.

> **NOTE:** The idea of logically marking up deleted text is similar to the idea of using revision marks in Microsoft Word. When revision marks are turned on, you can see the deleted text even though it is technically no longer part of the document.

Syntax:

```
<DEL CITE="url" DATETIME="YYYYMMDDThh:mm:ss"> ... deleted text goes here

... </DEL>
```

Attributes:

<DEL> can take two attributes:

- CITE--Provides the URL of a document that explains why the deletion was necessary.
- DATETIME--Puts a "timestamp" on the deletion.

Example:

```
She just got a big<DEL>, huge</DEL> raise.
```

In this example, the use of the word "huge" is redundant, so an astute copy editor would delete it.

Related Tags:

The <INS> tag has a similar function for inserted text.

<DFN>

Type:

Container

Function:

Denotes the defining instance of a term. Internet Explorer will display text tagged with <DFN> in italic, whereas Netscape Navigator will not use any special formatting.

Syntax:

```
<DFN> ... term being introduced goes here ... </DFN>
```

Attributes:

None.

Example:

```
Freud proposed the idea of a <DFN>catharsis</DFN> - a release

of psychic tension.
```

<EM>

Type:

Container

Function:

Contains text to be emphasized. Most browsers render emphasized text in italic.

Syntax:

```
<EM> ... emphasized text goes here ... </EM>
```

Attributes:

None.

Example:

```
Please do <EM>not</EM> disturb the dog.
```

<INS>

Type:

Container

Function:

Contains text that has been inserted into the document after its original draft.

Syntax:

```
<INS> ... inserted text goes here ... </INS>
```

Attributes:

Like <DEL>, <INS> can take two attributes:

- CITE--Provides the URL of a document that explains why the insertion was necessary.

- DATETIME--Puts a "timestamp" on the insertion.

Example:

```
The New World was discovered by <DEL>Magellan</DEL>

<INS>Columbus</INS> in 1492.
```

> **NOTE:** Note how <DEL> and <INS> are used together to strike some text and then to insert a correction in its place.

Related Tags:

The <DEL> tag logically represents deleted text.

<KBD>

Type:

Container

Function:

Contains text that represents keyboard input. Browsers typically render such text in a fixed-width font.

Syntax:

```
<KBD> ... keyboard input goes here ... </KBD>
```

Attributes:

None.

Example:

```
To begin, type <KBD>go</KBD> and press Enter.
```

<Q>

Type:

Container

Function:

Contains a direct quotation to be displayed inline.

Syntax:

```
<Q CITE="URL_of_cited_document"> ... quotation goes here ... </Q>
```

Attributes:

If you're quoting from an online source, you can set the CITE attribute equal to the source's URL. Also, you may wish to consider using the LANG attribute because quotes are denoted with different characters in many languages.

Related Tags:

The <BLOCKQUOTE> tag can also be used to denote quoted text, but block quotes are displayed with increased right and left indents and are not in line with the rest of the body text.

<SAMP>

Type:

Container

Function:

Contains text that represents the literal output from a program. Such output is sometimes referred to as *sample text*. Most browsers will render sample text in a fixed-width font.

Syntax:

```
<SAMP> ... program output goes here ... </SAMP>
```

Attributes:

None.

Example:

```
A common first exercise in a programming course is to write a program

to produce the message <SAMP>Hello World</SAMP>.
```

<STRONG>

Type:

Container

Function:

Contains text to be strongly emphasized. Browsers typically render strongly emphasized text in boldface (see Figure 3.7).

FIGURE 3.7 *The <STRONG> tag is useful for marking up recommendations with extra emphasis.*

Syntax:

```
<STRONG> ... strongly emphasized text goes here ... </STRONG>
```

Attributes:

None.

Example:

```
<STRONG>STOP!</STRONG> Do not proceed any further.  Contact your system

administrator.
```

<VAR>

Type:

Container

Function:

Denotes a variable from a computer program. Variables are typically rendered in a fixed-width font.

Syntax:

```
<VAR> ... program variable goes here ... </VAR>
```

Attributes:

None.

Example:

```
The <VAR>RecordCount</VAR> variable is set to the number of records

that the query retrieved.
```

Block-Level Formatting Tags

Block-level formatting tags are usually applied to larger content than the text-level formatting tags. As such, the block-level tags define major sections of a document, such as paragraphs, headings, abstracts, chapters, and so on. The tags profiled in this section are the ones to turn to when you want to define the block-level elements in a document you're authoring.

<BLOCKQUOTE>

Type:

Container

Function:

Contains quoted text that is to be displayed indented from regular body text (see Figure 3.8).

FIGURE 3.8 *Blockquotes are used to offset longer quoted passages.*

Syntax:

```
<BLOCKQUOTE CITE="URL_of_cited_document"> ... quoted text goes here ...  </BLOCKQUO
```

Attributes:

If you're quoting from an online source, you can set the CITE attribute equal to the source's URL.

Example:

```
Fans of Schoolhouse Rock will always be able to recite the preamble
of the United States Constitution:
<BLOCKQUOTE>

We, the people, in order to form a more perfect Union ...

</BLOCKQUOTE>
```

Related Tags:

The <Q> tag is used to denote quoted text that is to be displayed in line with the body text.

**
**

Type:

Standalone

Function:

Inserts a line break in the document. Carriage returns in the HTML code do not translate to line breaks on the browser screen, so authors often need to insert the breaks themselves. The
 tag is indispensable when rendering text with frequent line breaks, such as addresses or poetry. Unlike the <P> tag or the heading tags,
 adds no additional vertical space after the break.

Syntax:

```
<BR CLEAR="LEFT|RIGHT|ALL">
```

Attributes:

The CLEAR attribute tells which margin to break to when breaking beyond a floating page element, such as an image (see Figure 3.9). Setting CLEAR="LEFT" breaks to the first line in the left margin free of the floating object. CLEAR="RIGHT" breaks to the first clear right margin, and CLEAR="ALL" breaks to the first line in which both the left and right margins are clear.

FIGURE 3.9 *The
 tag can break to the next line or to the next line that is free of floating objects such as images or tables.*

Example:

```
First Name: <INPUT TYPE="TEXT" NAME="fname"><BR>
Last Name: <INPUT TYPE="TEXT" NAME="lname"><BR>
Telephone: <INPUT TYPE="TEXT" NAME="phone"><BR>

Email: <INPUT TYPE="TEXT" NAME="email">
```

<CENTER>

Type:

Container

Function:

Centers all text and other page components it contains.

Syntax:

```
<CENTER> ... centered page components go here ... </CENTER>
```

Attributes:

None.

Example:

```
<CENTER>
<I>A Midsummer Night's Dream</I><BR>
by William Shakespeare

</CENTER>
```

> **NOTE:** The W3C has deprecated the <CENTER> tag in favor of using the <DIV ALIGN="CENTER"> tag (see the following tag) or style sheets for centering.

<DIV>

Type:

Container

Function:

Defines a section or division of a document that requires a special alignment.

Syntax:

```
<DIV ALIGN="LEFT|RIGHT|CENTER|JUSTIFY">
...

</DIV>
```

Attributes:

The ALIGN attribute controls how text contained between the <DIV> and </DIV> tags is aligned. You can set ALIGN equal to LEFT, RIGHT, CENTER, or JUSTIFY, depending on the kind of alignment you need.

Example:

```
<DIV ALIGN="RIGHT">
Everything in this section is right-justified.  Hard to read, isn't it?
...

</DIV>
```

<HR>

Type:

Standalone

Function:

Places a horizontal line on the page (see Figure 3.10).

FIGURE 3.10 *Horizontal rules are a great way to break up a page and give the readers' eyes a rest.*

Syntax:

```
<HR ALIGN="alignment" NOSHADE SIZE="thickness"

WIDTH="pixels_or_percentage_of_screen">
```

Attributes:

The unmodified <HR> tag places a line, 1 pixel thick, across the page. The line will have a shading effect to give the illusion of being three-dimensional. You can change how the default line is displayed by using combinations of the following attributes:

- ALIGN--You can set ALIGN equal to LEFT, RIGHT, or CENTER to change how the horizontal line is aligned on the page. Note that this matters only when you've changed the width of the line to be something less than the browser screen width. The default value of ALIGN is CENTER.
- NOSHADE--Placing the NOSHADE attribute in an <HR> tag suppresses the shading effect and yields a solid line.
- SIZE--SIZE controls the thickness of the line. You set SIZE equal to the number of pixels in thickness you'd like the line to be.
- WIDTH--A line's WIDTH can be specified in one of two ways. You can set it equal to a number of pixels, or you can set it equal to a percentage of the user's browser screen width (or table cell, if you're placing a line inside a cell). Because you can't know the screen resolution settings of every user, you should use the percentage approach whenever possible.

All the attributes of the <HR> tag have been deprecated in favor of using style sheets to control horizontal rule properties.

Example:

```
<HR NOSHADE WIDTH=80% SIZE=4>
<DIV ALIGN="CENTER">Return to the Home Page</DIV>

<HR NOSHADE WIDTH=80% SIZE=4>
```

<H1>-<H6>

Type:

Container

Function:

Establishes a hierarchy of document heading levels. Level 1 has the largest font size. Increasing through the levels causes the font size to decrease. All headings are rendered in boldface and have a little extra line spacing built in above and below them (see Figure 3.11).

FIGURE 3.11 *Headings are rendered in boldface and are usually in a type size different from the body text.*

NOTE: Although the headings' tags are meant to be used in a strictly hierarchical fashion, many authors use them out of sequence to achieve the formatting effects they want.

Syntax:

```
<Hn ALIGN="LEFT|RIGHT|CENTER|JUSTIFY"> ... Level n heading ... </Hn>
```

where n = 1, 2, 3, 4, 5, or 6.

Attributes:

The ALIGN attribute controls how the heading is aligned on the page. You can set a heading's alignment to values of LEFT, RIGHT, CENTER, or JUSTIFY. The default alignment is LEFT.

Example:

```
<H1 ALIGN="CENTER">Table of Contents</H1>
<H2>Chapter 1 – Introduction</H2>
...
<H2>Chapter 2– Prior Research</H2>

...
```

<P>

Type:

Container

Function:

Denotes a paragraph. Most browsers ignore the use of multiple <P> tags to increase the amount of vertical space in a document.

Syntax:

```
<P ALIGN="LEFT|RIGHT|CENTER|JUSTIFY">

paragraph text

</P>
```

Attributes:

The ALIGN attribute controls how text in the paragraph is aligned. You can set ALIGN to LEFT (the default), RIGHT, CENTER, or JUSTIFY.

Example:

```
<P ALIGN="CENTER"><H1>Welcome!</H1></P>
```

NOTE: Although most browsers can parse standalone <P> tags without errors, the HTML 4.0 recommendation discourages the use of empty <P> tags.

<PRE>

Type:

Container

Function:

Denotes text to be treated as preformatted. Browsers render preformatted text in a fixed-width font. Whitespace characters, such as spaces, tabs, and carriage returns, found between the <PRE> and </PRE> tags are not ignored. This makes preformatted text a viable option for presenting tables of information.

Syntax:

```
<PRE WIDTH="width_of_widest_line">
... preformatted text goes here ...

</PRE>
```

Attributes:

The <PRE> tag's WIDTH attribute is set to the number of characters in the widest line of the preformatted text block. This information helps some browsers choose the font size for displaying the text. Use of the WIDTH attribute has been deprecated in HTML 4.0.

Example:

```
<PRE WIDTH=34>
Catalog No.  Item           Price
AZ-1390      Polo Shirt     $29.99
FT-0081      Sweater        $52.99
CL-9334      Belt           $16.99

</PRE>
```

<SPAN>

Type:

Container

Function:

Generic container tag for defining a document block. One popular use is for applying style information.

Syntax:

```
<SPAN STYLE="style information" ALIGN="LEFT|RIGHT|CENTER|JUSTIFY">
```

```
range of text over which style is to be applied

</SPAN>
```

Attributes:

If you're assigning style information, you can set the STYLE attribute to a sequence of as many *characteristic*: *value* pairs as you need to specify the style information you're applying. Valid style characteristics are those put forward in the Cascading Style Sheets Level 2 specification.

The ALIGN attribute can take on the customary values of LEFT, RIGHT, CENTER, and JUSTIFY.

Example:

```
<SPAN STYLE="font-weight: bold; color: red; text-indent: 0.25 in">
Here is some bold, red, text that's indented by one quarter of an inch.

</SPAN>
```

List Tags

Technically, HTML lists are a form of block-level formatting, but because lists are such a useful way of presenting content, the list tags merit their own section in the chapter.

HTML 4.0 continues to support five types of lists, although tags for two of the five have been deprecated. Using the tags in this section, you can create the following types of lists:

- Definition lists
- Directory lists (deprecated)
- Menu lists (deprecated)
- Ordered (numbered) lists
- Unordered (bulleted) lists

Most HTML lists make use of the list item tag, <LI>, so this tag is covered first, followed by the tags you use to create each type of list.

<LI>

Type:

Container

Function:

Denotes an item in a list.

Syntax:

```
<LI TYPE="list_type" START="start_value"> ... list item goes here ... </LI>
```

Attributes:

The <LI> tag can take four attributes:

- COMPACT--Instructs the browser to render the list item in as small a space as possible.
- START--(Ordered lists only) You can change the starting value of the numbering sequence from the default of 1 to any other value you choose.
- TYPE--(Ordered and unordered lists) You can modify the numbering scheme in an ordered list or the bullet character in an unordered list by setting TYPE to one of the list types available. Ordered list types include 1 (Arabic numerals), A (uppercase alphabet), a (lowercase alphabet), I (uppercase Roman numerals), and i (lowercase Roman numerals). The unordered list types include DISC (solid circular bullet), SQUARE (solid square bullet), and CIRCLE (open circular bullet).
- VALUE--Sets the numbering value of the list item.

All the attributes listed above have been deprecated.

> **NOTE:** Even if you are not using an Arabic numeral numbering scheme, you should still set START equal to a numeric value. Browsers know to map the START value to any numbering scheme you've specified in a TYPE attribute. For example, the code:
> <LI TYPE="a" START="4">
> will produce an ordered list beginning with the lowercase letter d.

Example:

```
<LI>Cookie Dough</LI>
<LI>Rocky Road</LI>

<LI>Mint Chocolate Chip</LI>
```

Related Tags:

The <LI> tag is always used in conjunction with one of the other HTML list tags: <DIR>, <MENU>, <OL>, and <UL>.

> **NOTE:** Even though the <LI> tag is technically a container tag, most people use it as a standalone tag, and most browsers are able to interpret standalone <LI> tags correctly.

<DIR>

Type:

Container

Function:

Creates a directory listing. Items in a directory list are bulleted and generally short--usually not more than 20 characters in length. Originally, directory lists were intended for rendering narrow columns of information, such as indexes or telephone directory listings.

The <DIR> tag has been deprecated by the W3C. You should use an unordered list (<UL>) instead.

Syntax:

```
<DIR COMPACT>
<LI>List item 1</LI>
<LI>List item 2</LI>
...

</DIR>
```

Attributes:

The optional COMPACT attribute instructs a browser to reduce the spacing between list items so that the list is rendered in the smallest amount of vertical space possible.

Example:

```
<DIR>
<LI>Mary Garrison, x521</LI>
<LI>Tom Hinkle, x629</LI>
<LI>Pat Joseph, x772</LI>

</DIR>
```

Related Tags:

List items in a directory list are specified with the <LI> tag.

<DL>

Type:

Container

Function:

Denotes a definition list (see Figure 3.12).

FIGURE 3.12 *Definition lists have a term/definition structure similar to a glossary at the back of a book.*

Syntax:

```
<DL COMPACT>
 ... terms and definitions go here ...

</DL>
```

Attributes:

The COMPACT attribute is optional and enables you to compress the list into the smallest vertical space possible on the browser screen.

Example:

```
<DL>
<DT>Browser</DT>
<DD>A program that allow a user to view World Wide Web pages</DD>
<DT>Server</DT>
<DD>A program that fields requests for web pages</DD>

</DL>
```

Related Tags:

Terms in a definition list are specified with the <DT> tag, and their definitions are specified with the <DD> tag.

<DT>

Type:

Container

Function:

Contains a term to be defined in a definition list.

> **NOTE:** Some browsers will automatically render a definition list term in boldface.

Syntax:

```
<DT> ... term being defined goes here ... </DT>
```

Attributes:

None.

Example:

```
<DL>
```

```
<DT>Creatine</DT>
<DD>A nutritional supplement that promotes muscle development</DD>
...

</DL>
```

Related Tags:

Use of the <DT> tag makes sense only in the context of a definition list (between the <DL> and </DL> tags). The <DD> tag is used to give the term's definition.

<DD>

Type:

Container

Function:

Contains a term's definition. The definition is typically indented from the term, making it easier for the reader to see the term-definition structure of the list.

Syntax:

```
<DD> ... term definition goes here ... </DD>
```

Attributes:

None.

Example:

```
<DL>
<DT>HTML</DT>
<DD>A document description language used to author Web pages</DD>
...

</DL>
```

Related Tags:

The <DD> tag should be used only when contained by <DL> and </DL> tags. A term, specified by a <DT> tag, should precede each definition.

<MENU>

Type:

Container

Function:

Creates a menu listing. Menu list items are typically short--usually not more than 20 characters in length-- and are arranged in a single column. Most browsers render a menu list in the same way they render a bulleted list.

The use of menu lists has been deprecated by the W3C. You should use the unordered list tag (<UL>) instead.

Syntax:

```
<MENU COMPACT>
<LI>Menu list item 1</LI>
<LI>Menu list item 2</LI>
...

</MENU>
```

Attributes:

The optional COMPACT attribute is used to reduce vertical spacing between list items.

Example:

```
<MENU COMPACT>
<LI>Enter a Purchase Order</LI>
<LI>Payroll Functions</LI>
<LI>Generate invoices</LI>

</MENU>
```

Related Tags:

List items in a menu listing are specified with the <LI> tag.

<OL>

Type:

Container

Function:

Creates an ordered or numbered list (see Figure 3.13).

Syntax:

```
<OL TYPE="1|A|a|I|i" START="start_value" COMPACT>
<LI>List item 1</LI>
<LI>List item 2</LI>
...

</OL>
```

Attributes:

The <OL> tag can take the following attributes:

- COMPACT--Instructs the browser to reduce vertical spacing between list items.
- START--You can change to a position other than the first position in the ordering scheme by using the START attribute. For example, setting START to 3 with TYPE set equal to I produces a list that begins numbering with III (3 in uppercase Roman numerals).
- TYPE--Controls the numbering scheme used when rendering the list. The default value of 1 indicates the use of Arabic numerals, but you can also choose from uppercase letters (A), lowercase letters (a), uppercase Roman numerals (I), or lowercase Roman numerals (i).

FIGURE 3.13 *Ordered and unordered lists are commonly used on Web pages today.*

All the attributes of the <OL> tag have been deprecated.

Example:

```
Book Outline
<OL TYPE="A">
<LI>HTML</LI>
<LI>XML</LI>
<LI>Dynamic HTML</LI>
<LI>Java</LI>
<LI>JavaScript</LI>

</OL>
```

Related Tags:

List items in an ordered list are specified with the <LI> tag.

<UL>

Type:

Container

Function:

Creates an unordered or bulleted list.

Syntax:

```
<UL TYPE="DISC|SQUARE|CIRCLE" COMPACT>
<LI>List item 1</LI>
<LI>List item 2</LI>
...
```

```
</UL>
```

Attributes:

The <UL> tag can take the following attributes:

- COMPACT--Reduces the vertical spacing between list items.
- TYPE--Enables you to specify which bullet character to use when rendering the list. This can be helpful when nesting bulleted lists because browsers have a default progression of bulleted characters that they use. You can override the browser's choice of bulleted characters in the nested lists by using TYPE.

The attributes listed above have been deprecated in HTML 4.0.

Example:

```
Web Browsers
<UL TYPE="SQUARE">
<LI>Netscape Navigator</LI>
<LI>Microsoft Internet Explorer</LI>
<LI>NCSA Mosaic</LI>

</UL>
```

Related Tags:

List items in an unordered list are specified with the <LI> tag.

Hyperlink Tags

The capability of linking Web resources is what makes the Web so fascinating. By following links, you can be looking up job opportunities one moment and then be reading up on the latest mixed drink recipes the next! Linking between documents is accomplished with the one simple tag described in this section.

<A>

Type:

Container

Function:

The <A> tag can do one of two things, depending on which attributes you use. Used with the HREF attribute, the <A> tag sets up a hyperlink from whatever content is found between the <A> and </A> tags and the document at the URL specified by HREF (see Figure 3.14). When you use the <A> tag with the NAME attribute, you set up a named anchor within a document that can be targeted by other hyperlinks. This helps make navigating a large document easier because you can set up anchors at the

start of major sections and then place a set of links at the top of the document that points to the anchors at the beginning of each section.

FIGURE 3.14 *Hypertext linking between documents puts the "Web" in World Wide Web.*

> **NOTE:** Hypertext links are typically colored and underlined. A linked graphic will be rendered with a colored border. If you don't want a border around your linked image, be sure to specify BORDER=0 in the <IMG> tag you use to place the image.

Syntax:

```
<!-- Setting up a hyperlink -->
<A HREF="URL_of_linked_document" TARGET="frame_name"
   REL="forward_link_type" REV="reverse_link_type"
   ACCESSKEY="key_letter" TABINDEX="tab_order_position">
... hyperlinked element goes here ...
</A>
```

or

```
<!-- Setting up a named anchor -->
<A NAME="anchor_name">
... text to act as named anchor ...

</A>
```

Attributes:

The <A> tag can take a host of attributes, including

- ACCESSKEY--An access key is a shortcut key a reader can use to activate the hyperlink. If you set the access key to the letter "C", for example, Windows users can press Alt+C on their keyboards to activate the link.
- CHARSET--Denotes what character encoding to use for the linked document.
- HREF--Gives the URL of the Web resource to which the hyperlink should point.
- HREFLANG--Denotes the language context of the linked resource.
- NAME--Specifies the name of the anchor being set up.
- REL--Describes the nature of the forward link (see Table 3.3 for possible values).
- REV--Describes the nature of the reverse link (see Table 3.3 for possible values).
- TABINDEX--Specifies the link's position in the document's tabbing order.
- TARGET--Tells the browser into which frame the linked document should be loaded.

- TYPE--Specifies the MIME type of the linked resource.

Examples:

The following code sets up a simple hyperlink:

```
You can learn more about our
<A HREF="prodserv.html TARGET="main" ACCESSKEY="P">

products and services</A> as well.
```

To follow the link, a user can click the hypertext products and services or press Alt+P (on a Windows machine) or Cmd+P (on a Macintosh).

This code establishes a named anchor within a document:

```
...
<A NAME="toc">
<H1>Table of Contents</H1>
</A>

...
```

With the anchor set up, you can point a hyperlink to it by using code such as this:

```
<A HREF="index.html#toc">Back to the Table of Contents</A>
```

Image and Imagemap Tags

Without images, the Web would just be another version of Gopher. Web graphics give pages powerful visual appeal and often add significantly to the messages that authors are trying to convey.

Placing an image on a page is as simple as using the HTML <IMG> tag. In its most basic form, the <IMG> tag needs only one attribute to do its job. However, <IMG> supports as many as 10 attributes that you can use to modify how the image is presented.

SEE "Advanced Graphics," **p. 155**.

<IMG>

Type:

Standalone

Function:

Places an inline image into a document (see Figure 3.15).

FIGURE 3.15 *Pictures, logos, and other graphical effects are placed into a document using the*

<IMG> tag.

Syntax:

```
<IMG SRC="URL_of_image_file"
  WIDTH="width_in_pixels" HEIGHT="height_in_pixels"
  ALT="text_description" BORDER="thickness_in_pixels"
  ALIGN="TOP|MIDDLE|BOTTOM|LEFT|RIGHT"
  HSPACE="horizontal_spacing_in_pixels"
  VSAPCE="vertical_spacing_in_pixels"
  LONGDESC="URL_of_long_description"

  ISMAP USEMAP="map_name">
```

Attributes:

As you can see from the tag's syntax, <IMG> can take several attributes (each attribute is described in detail in this section):

- SRC--Specifies the URL of the file containing the image.
- WIDTH and HEIGHT--Gives the width and height of the image in pixels. Specifying this information in the tag means that the browser can allot space for the image and then continue laying out the page while the image file loads.
- ALT--A text-based description of the image content. Using ALT is an important courtesy to users with nonvisual browsers or with image loading turned off.
- BORDER--Controls the thickness of the border around an image. An image has no border by default. However, a hyperlinked image will be rendered with a colored border. If you don't want the border to appear around your image, you need to set BORDER=0.
- ALIGN--Controls how text flows around the image. TOP, MIDDLE, and BOTTOM alignment aligns text following the image with the top, middle, or bottom of the image, respectively. However, after the text reaches the end of the current line, it will break to a position *below* the image. If you want text to wrap around the entire image, you need to use the LEFT or RIGHT values of the ALIGN attribute to float the image in the left or right margin. Text will wrap smoothly around a floated image.
- HSPACE and VSPACE--Controls the amount of whitespace left around the image. HSPACE is set to the number of whitespace pixels to use on the left and right sides of the image. VSPACE controls the number of whitespace pixels remaining above and below the image.
- LONGDESC--Points to a resource that contains a longer description of the image's content.
- ISMAP--Identifies the image as being used as part of a server-side imagemap.
- USEMAP--Set equal to the name of the client-side imagemap to be used with the image.

Example:

```
<IMG SRC="/images/logo.gif" WIDTH=600 HEIGHT=120
  ALT="Welcome to XYZ Corporation" USEMAP="#main"

  VSPACE=10>
```

One popular use of images is to set up *imagemaps*--clickable images that take users to different URLs, depending on where they click. Imagemaps are popular page elements on many sites because they provide users with an easy-to-use graphical interface for navigating the site (see Figure 3.16).

FIGURE 3.16 *Imagemaps are commonly used as navigation interfaces and are usually accompanied by an equivalent set of hypertext links.*

Imagemaps come in two flavors: server-side and client-side. When a user clicks a server-side imagemap, the coordinates of the click are sent to the server, where a program processes them to determine which URL the browser should load. To accomplish this, the server needs to have access to a file containing information about which regions on the image are clickable and with which URLs those regions should be paired.

With client-side imagemaps, the client (browser) processes the coordinates of the user's click, rather than passing them to the server for processing. This is a more efficient approach because it reduces the computational load on the server and eliminates the opening and closing of additional HTTP connections. For the browser to be able to process a user's click, it has to have access to the same information about the clickable regions and their associated URLs as the server does when processing a server-side imagemap. The method of choice for getting this information to the client is to pass it in an HTML file--usually the file that contains the document with the imagemap, although it does not necessarily have to be this way. HTML 4.0 supports two tags that enable you to store imagemap data in your HTML files: <MAP> and <AREA>. A discussion of these tags rounds out the coverage in this section.

SEE "Imagemaps," **p. 137**.

<MAP>

Type:

Container

Function:

Contains HTML tags that define the clickable regions (hot regions) of an imagemap.

Syntax:

```
<MAP NAME="map_name">
... hot region definitions go here ...

</MAP>
```

Attributes:

The NAME attribute gives the map information a unique name so it can be referenced by the USEMAP attribute in the <IMG> tag that places the imagemap graphic.

Example:

```
<MAP NAME="navigation">
<AREA SHAPE="RECT" COORDS="23,47,58,68" HREF="search.html">
<AREA SHAPE="CIRCLE" COORDS="120,246,150,246" HREF="about.html">
...

</MAP>
```

With the imagemap data defined by the map named navigation, you would reference the map in an <IMG> tag as follows:

```
<IMG SRC="navigation.gif" USEMAP="#navigation">
```

If the map were stored in a file different from the document's HTML file, you would reference it this way:

```
<IMG SRC="navigation.gif" USEMAP="maps.html#navigation">
```

Related Tags:

The <AREA> tag is used to define the individual hot regions in the imagemap. The named map is referenced by the USEMAP attribute of the <IMG> tag.

<AREA>

Type:

Standalone

Function:

Defines a hot region in a client-side imagemap.

Syntax:

```
<AREA SHAPE="RECT|CIRCLE|POLY|DEFAULT" COORDS="coordinate_list"
  HREF="URL_of_linked_document" TARGET="frame_name"
  ALT="text_alternative" TABINDEX="tab_order_position" NOHREF

  ACCESSKEY="key_letter">
```

Attributes:

The <AREA> tag takes a number of attributes, including

- ACCESSKEY--Defines a shortcut key combination that the user can press to activate the hot

region (see the attribute listing for the <A> tag for more details).

- ALT--Provides a text alternative for the hot region in the event that the image does not load or the user has image loading turned off. ALT text is also used by spoken-word browsers for the visually impaired.
- COORDS--Specifies the coordinates that define the hot region. Coordinates are given as a list of numbers, separated by commas. No coordinates are needed when specifying a DEFAULT region.
- HREF--Set equal to the URL of the document to associate with the hot region.
- NOHREF--Using the NOHREF attribute in an <AREA> tag essentially deactivates the hot region by having it point to nothing.
- SHAPE--Specifies the shape of the hot region being defined. Possible values of SHAPE include RECT for rectangles, CIRCLE for circles, POLY for polygons, and DEFAULT for any point on the image not part of another hot region.
- TABINDEX--Defines the hot region's position in the tabbing order of the page.
- TARGET--Specifies into which frame to load the linked document.

NOTE: Each type of hot region has a specific number of coordinate points that you need to specify to completely define the hot region. A rectangular region is defined by the coordinates of the upper-left and lower-right corners, a circular region by the coordinates of the center point and a point along the edge of the region, and a polygonal region by the coordinates of the polygon's vertices.

Example:

```
<MAP NAME="main">
<AREA SHAPE="POLY" COORDS="35,80,168,99,92,145" HREF="profile.html">
<AREA SHAPE="CIRCLE" COORDS="288,306,288,334" HREF="feedback.html">
<AREA SHAPE="DEFAULT" HREF="index.html">

</MAP>
```

Related Tags:

<AREA> tags are allowable only between <MAP> and </MAP> tags.

Table Tags

HTML table tags are not only a great way to present information, but a useful layout tool as well (see Figure 3.17). HTML 4.0 expands the table tags in several important ways:

- Support for rendering parts of the frame around a table, rather than "all or nothing."

- Control over which boundaries to draw between cells.
- Table header, body, and footer sections can be defined as separate entities.

FIGURE 3.17 *Tables make complex page layouts possible because of the very fine alignment control you have within the table.*

This section looks at all the table-related tags and their many attributes.

SEE "Tables," **p. 183**.

<TABLE>

Type:

Container

Function:

Contains all HTML tags that compose a table.

Syntax:

```
<TABLE ALIGN="LEFT|CENTER|RIGHT" BORDER="thickness_in_pixels"
  BGCOLOR="color" WIDTH="pixels_or_percentage_of_browser_width"
  COLS="number_of_columns" CELLPADDING="pixels" CELLSPACING="pixels"
  FRAME="outer_border_rendering" RULES="inner_border_rendering"
  SUMMARY="description_of_table_contents_and_structure">
...

</TABLE>
```

Attributes:

The <TABLE> tag can take the following attributes to modify how the table is presented:

- ALIGN--Controls how the table is aligned on the page. Possible values are LEFT, CENTER, and RIGHT. Tables that are left- or right-aligned will float in the margin, and text can wrap around them.The ALIGN attribute of the <TABLE> tag has been deprecated.
- BORDER--Specifies the thickness of the table border in pixels.
- BGCOLOR--Set equal to the background color to use in the cells of the table.
- CELLPADDING--Controls the amount of whitespace between the contents of a cell and the edge of the cell.
- CELLSPACING--Specifies how many pixels of space to leave between individual cells.

- COLS--Set equal to the number of columns in the table. Knowing this value enables the browser to compose the table faster.
- FRAME--Controls which parts of the table's outer border are rendered. FRAME can take on the values shown in Table 3.4.

TABLE 3.4 Values of the FRAME Attribute of the <TABLE> Tag

Value	Purpose
ABOVE	Displays a border on the top of a table frame
BELOW	Displays a border at the bottom of a table frame
BORDER	Displays a border on all four sides of a table frame
BOX	Same as BORDER
HSIDES	Displays a border on the left and right sides of a table frame
LHS	Displays a border on the left side of a table frame
RHS	Displays a border on the right side of a table frame
VSIDES	Displays a border at the top and bottom of a table frame
VOID	Suppresses the display of all table frame borders

- RULES--Controls which parts of the table's inner borders are displayed. RULES can be set equal to one of the values shown in Table 3.5.

TABLE 3.5 Values of the RULES Attribute of the <TABLE> Tag

Value	Purpose
ALL	Displays a border between all rows and columns
COLS	Displays a border between all columns
GROUPS	Displays a border between all logical groups (as defined by the <THEAD>, <TBODY>, <TFOOT>, and <COLGROUP> tags)
NONE	Suppresses all inner borders
ROWS	Displays a border between all table rows

- SUMMARY--Provides a synopsis of what's in the table and how the table is structured.

- WIDTH--Specifies the width of the table in pixels or as a percentage of the browser screen width.

Example:

```
<TABLE BORDER=2 CELLPADDING=4 FRAME=BORDER RULES=ALL ALIGN=CENTER>
...

</TABLE>
```

Related Tags:

The <TABLE> and </TABLE> tags form the container for all the other table-related tags. The many tags you can use between <TABLE> and </TABLE> include <CAPTION>, <THEAD>, <TFOOT>, <TBODY>, <COLGROUP>, <COL>, <TR>, <TH>, and <TD>.

<CAPTION>

Type:

Container

Function:

Specifies a caption for a table.

Syntax:

```
<CAPTION ALIGN="TOP|BOTTOM|LEFT|RIGHT">
... caption text goes here ...

</CAPTION>
```

Attributes:

The ALIGN attribute gives you fine control over how the caption is placed. Setting ALIGN to TOP or BOTTOM places the caption above or below the table, respectively. Using LEFT or RIGHT floats the caption in the left or right margin. The ALIGN attribute has been deprecated in HTML 4.0.

Example:

```
<CAPTION ALIGN="BOTTOM">
Table 1 - Return on Investment

</CAPTION>
```

<THEAD>

Type:

Container

Function:

Defines the header section of a table. Being able to define the header separately enables the browser to duplicate the header when breaking the table across multiple pages.

Syntax:

```
<THEAD ALIGN="LEFT|CENTER|RIGHT|JUSTIFY|CHAR"
 VALIGN="TOP|MIDDLE|BOTTOM|BASELINE" CHAR="alignment_character"
 CHAROFF="alignment_character_offset">
... rows that comprise the header ...

</THEAD>
```

Attributes:

The <THEAD> tag can take the following four attributes:

- ALIGN--Controls the horizontal alignment within the cells of the table header. ALIGN can take on values of LEFT, RIGHT, CENTER, JUSTIFY, or CHAR. CHAR is used to align cells by a common character.
- CHAR--Specifies the alignment character for when ALIGN="CHAR" is used.
- CHAROFF--Prescribes the offset distance from the alignment character.
- VALIGN--Controls the vertical alignment in the header cells. VALIGN can take on values of TOP, MIDDLE, BOTTOM, or BASELINE.

Example:

```
<THEAD ALIGN="CENTER" VALIGN="BASELINE">
<TR>
<TH>ID #</TH>
<TH>Property</TH>
<TH>Tax Assessment</TH>
...
</TR>

</THEAD>
```

Related Tags:

The rows of the table header are built with <TR>, <TH>, and <TD> tags. Each table header must comprise at least one row.

<TFOOT>

Type:

Container

Function:

Defines the footer section of the table.

Syntax:

```
<TFOOT ALIGN="LEFT|CENTER|RIGHT|JUSTIFY|CHAR"
 VALIGN="TOP|MIDDLE|BOTTOM|BASELINE" CHAR="alignment_character"
 CHAROFF="alignment_character_offset">
...

</TFOOT>
```

Attributes:

<TFOOT> can take the same ALIGN and VALIGN attributes as the <THEAD> tag:

- ALIGN--Controls the horizontal alignment within the cells of the table footer. ALIGN can take on values of LEFT, RIGHT, CENTER, JUSTIFY, or CHAR.
- CHAR--Specifies the alignment character for when ALIGN="CHAR" is used.
- CHAROFF--Prescribes the offset distance from the alignment character.
- VALIGN--Controls the vertical alignment in the footer cells. VALIGN can take on values of TOP, MIDDLE, BOTTOM, or BASELINE.

Example:

```
<TFOOT ALIGN="JUSTIFY" VALIGN="TOP">
<TR>
<TD>&copy; 1998 - Macmillan Computer Publishing USA</TD>
...
</TR>

</TFOOT>
```

Related Tags:

You specify the rows and cells in the table footer by using the <TR>, <TH>, and <TD> tags. A table footer must be made up of at least one row.

<TBODY>

Type:

Container

Function:

Defines the body section of the table.

Syntax:

```
<TBODY ALIGN="LEFT|CENTER|RIGHT|JUSTIFY|CHAR"
 VALIGN="TOP|MIDDLE|BOTTOM|BASELINE" CHAR="alignment_character"
 CHAROFF="alignment_character_offset">
...

</TBODY>
```

Attributes:

<TBODY> can take the following attributes:

- ALIGN--Controls the horizontal alignment within the cells of the table body. ALIGN can take on values of LEFT, RIGHT, CENTER, JUSTIFY, and CHAR.
- CHAR--Specifies the alignment character for when ALIGN="CHAR" is used.
- CHAROFF--Prescribes the offset distance from the alignment character.
- VALIGN--Controls the vertical alignment in the body cells. VALIGN can take on values of TOP, MIDDLE, BOTTOM, or BASELINE.

Example:

```
<TBODY ALIGN="LEFT" VALIGN="BASELINE">
<TR>
<TD>Red Storm Rising</TD>
<TD>1500 pages (paperback)</TD>
<TD>$9.95</TD>
...
</TR>

</TBODY>
```

Related Tags:

You specify the rows and cells in the table body by using the <TR>, <TH>, and <TD> tags. A table body section must contain at least one row.

<COLGROUP>

Type:

Container

Function:

Groups a set of columns so that properties may be assigned to all columns in the group rather than to each one individually.

Syntax:

```
<COLGROUP SPAN="number_of_columns" WIDTH="width_of_column_group"
  ALIGN="LEFT|RIGHT|CENTER|JUSTIFY|CHAR"
 VALIGN="TOP|MIDDLE|BOTTOM|BASELINE"
 CHAR="alignment_character" CHAROFF="alignment character_offset">
...

</COLGROUP>
```

The <COLGROUP> and </COLGROUP> tags have no content or code between them if the properties put forward in the <COLGROUP> tag are to apply to each column in the group. You can also use the <COL> tag between <COLGROUP> and </COLGROUP> to specify column properties for a subgroup of the larger group.

Attributes:

<COLGROUP> can take the following attributes:

- ALIGN--Controls the horizontal alignment within the column group. ALIGN can take on values of LEFT, RIGHT, CENTER, JUSTIFY, or CHAR.
- CHAR--Specifies the alignment character for when ALIGN="CHAR" is used.
- CHAROFF--Prescribes the offset distance from the alignment character.
- SPAN--Tells the browser how many columns are in the group.
- VALIGN--Controls the vertical alignment in the column group. VALIGN can take on values of TOP, MIDDLE, BOTTOM, or BASELINE.
- WIDTH--Specifies how wide (in pixels or in terms of relative width) the enclosed columns should be.

Example:

```
<COLGROUP SPAN=3 ALIGN="CENTER" VALIGN="TOP">
</COLGROUP>
<TR>
<TD>Column 1 - center/top alignment</TD>
<TD>Column 2 - center/top alignment</TD>
<TD>Column 3 - center/top alignment</TD>
<TD>Column 4 - default alignment</TD>
</TR>

</TFOOT>
```

Related Tags:

The <COL> tag can be used between the <COLGROUP> and </COLGROUP> tags to refine column properties for a subset of the column group.

<COL>

Type:

Standalone

Function:

Specifies properties for a column or columns within a group.

Syntax:

```
<COL SPAN="number_of_columns" WIDTH="width_of_column_subgroup"
 ALIGN="LEFT|RIGHT|CENTER|JUSTIFY"
 VALIGN="TOP|MIDDLE|BOTTOM|BASELINE"

 CHAR="alignment_character" CHAROFF="alignment_character_offset">
```

Attributes:

<COL> can take the following attributes:

- ALIGN--Controls the horizontal alignment within the column cells. ALIGN can take on values of LEFT, RIGHT, CENTER, JUSTIFY, or CHAR.
- CHAR--Specifies the alignment character for when ALIGN="CHAR" is used.
- CHAROFF--Prescribes the offset distance from the alignment character.
- SPAN--Tells the browser how many columns to which to apply the property.
- VALIGN--Controls the vertical alignment in the column cells. VALIGN can take on values of TOP, MIDDLE, BOTTOM, or BASELINE.
- WIDTH--Specifies the width (in pixels or in terms of relative width) of the column or column group.

Example:

```
<TABLE BORDER=1>
<COLGROUP>
   <COL ALIGN=CENTER>
   <COL ALIGN=RIGHT>
</COLGROUP>
<COLGROUP>
   <COL ALIGN=CENTER SPAN=2>
</COLGROUP>
<TBODY>
```

```
   <TR>
      <TD>First column in first group, center aligned</TD>
      <TD>Second column in first group, right aligned</TD>
      <TD>First column in second group, center aligned</TD>
      <TD>Second column in second group, center aligned</TD>
   </TR>
</TBODY>

</TABLE>
```

<TR>

Type:

Container

Function:

Defines a row of a table, table header, table footer, or table body.

Syntax:

```
<TR ALIGN="LEFT|RIGHT|CENTER|JUSTIFY|CHAR"
  VALIGN="TOP|MIDDLE|BOTTOM|BASELINE">
...
</TR>
```

Attributes specified in a <TR> tag apply only to the row that the tag is defining and will override any default values.

Attributes:

The <TR> tag can take the following attributes:

- ALIGN--Controls the horizontal alignment within the cells in the row. ALIGN can take on values of LEFT, RIGHT, CENTER, JUSTIFY, or CHAR.
- CHAR--Specifies the alignment character for when ALIGN="CHAR" is used.
- CHAROFF--Prescribes the offset distance from the alignment character.
- VALIGN--Controls the vertical alignment of the cells in the row. VALIGN can take on values of TOP, MIDDLE, BOTTOM, or BASELINE.

Example:

```
<TR BGCOLOR="white" VALIGN="TOP">
<TD>Phone</TD>
<TD>Extension</TD>
<TD>Fax</TD>
...

</TR>
```

Related Tags:

Cells in a row are defined using the <TD> or <TH> tags.

<TD>, <TH>

Type:

Container

Function:

Defines a cell in a table. <TH> creates a header cell whose contents will be rendered in boldface and with a centered horizontal alignment. <TD> creates a regular data cell whose contents are aligned flush left and in a normal font weight. Vertical alignment for both types of cells is MIDDLE by default.

Syntax:

```
<TD ALIGN="LEFT|RIGHT|CENTER|JUSTIFY|CHAR"
 VALIGN="TOP|MIDDLE|BOTTOM|BASELINE"
 CHAR="alignment_character" CHAROFF="alignment_character_offset"
 NOWRAP ROWSPAN="number_of_rows"
 COLSPAN="number_of_columns"
 ABBR="header_cell_abbreviation" AXIS="list_of_category_names"

 HEADERS="list_of_ID headers" SCOPE="ROW|COL|ROWGROUP|COLGROUP">
```

or

```
<TH ALIGN="LEFT|RIGHT|CENTER|JUSTIFY|CHAR"
 VALIGN="TOP|MIDDLE|BOTTOM|BASELINE"
 CHAR="alignment_character" CHAROFF="alignment_character_offset"
 NOWRAP ROWSPAN="number_of_rows"
 COLSPAN="number_of_columns"
 ABBR="cell_abbreviation" AXIS="list_of_category_names"

 HEADERS="list_of_ID headers" SCOPE="ROW|COL|ROWGROUP|COLGROUP">
```

Attributes:

Both the <TH> and <TD> tags can take the following attributes:

- ABBR--Specifies an abbreviation form of a cell's contents.
- ALIGN--Controls the horizontal alignment within the cell. ALIGN can take on values of LEFT, RIGHT, CENTER, JUSTIFY, or CHAR.
- AXIS--Used to group cells into logical categories.
- CHAR--Specifies the alignment character for when ALIGN="CHAR" is used.

- CHAROFF--Prescribes the offset distance from the alignment character.
- COLSPAN--Specifies the number of columns the cell should occupy.
- HEADERS--Provides a list of IDs of cells that provide header information for the current cell.
- NOWRAP--Suppresses text wrapping within the cell.
- ROWSPAN--Specifies the number of rows the cell should occupy.
- SCOPE--A simpler form of the AXIS attribute, SCOPE lets you group cells into rows, columns, row groups, or column groups, instead of arbitrarily named logical groups.
- VALIGN--Controls the vertical alignment of the cell. VALIGN can take on values of TOP, MIDDLE, BOTTOM, or BASELINE.

FIGURE 3.18 *HTML forms gather user input and send that information to a server for processing.*

Example:

```
<TR VALIGN="BOTTOM">
<TH>Column 1 - center/bottom alignment</TH>
<TD VALIGN="MIDDLE">Column 2 - left/middle alignment</TD>
<TD ALIGN="JUSTIFY">Column 3 - justify/bottom alignment</TD>
<TD COLSPAN=2>Columns 4 and 5 - left/bottom alignment</TD>

</TR>
```

Form Tags

HTML forms are a Web surfer's gateway to interactive content. Forms collect information from a user, and then a script or program on a Web server uses the information to compose a custom response to the form submission.

For all the form controls that are available to you as a document author, you need to know surprisingly few tags to produce them. These tags, together with some new tags introduced in the HTML 4.0 recommendation that improve form accessibility for the disabled, are covered in this section.

SEE "Forms," **p. 233**.

<FORM>

Type:

Container

Function:

Contains the text and tags that compose an HTML form (see Figure 3.18).

Syntax:

```
<FORM ACTION="URL_of_processing_script" METHOD="GET|POST"
  TARGET="frame_name" ENCTYPE="MIME_type_of_file_to_upload"
  ACCEPT-CHARSET="acceptable_character_sets"
  ACCEPT="acceptable_MIME_types">
...

</FORM>
```

The <FORM> tag and its attributes are sometimes referred to as the *form header*.

Attributes:

The <FORM> tag takes the following attributes:

- ACCEPT--Specifies a list of acceptable content types (MIME types) that a server processing the form can handle correctly.
- ACCEPT-CHARSET--Set equal to a list of character sets that the form's processing script can handle.
- ACTION--Set equal to the URL of the script or program that will process the form data. ACTION is a required attribute of the <FORM> tag.
- ENCTYPE--Used when you're expecting a file upload as part of the form data submission and is set equal to the expected MIME type of the file.
- METHOD--Refers to the HTTP method used to send the form data to the server. The default METHOD is GET, which appends the data to the end of the processing script URL. If you set METHOD="POST", the form data will be sent to the server in a separate HTTP transaction.
- TARGET--Enables you to target the response from the processing script or program to a specific frame.

Example:

```
<FORM ACTION="/shopping_cart.cfm" METHOD="POST" TARGET="response">
...

</FORM>
```

Related Tags:

The following tags are valid only when used between the <FORM> and </FORM> tags: <INPUT>, <SELECT>, <OPTION>, <OPTGROUP>, <TEXTAREA>, <BUTTON>, <LABEL>, <FIELDSET>, and <LEGEND>. Each of these tags is described in this section.

<INPUT>

Type:

Standalone

Function:

Places one of the following form controls:

- Text, password, or hidden fields
- Check boxes
- Radio buttons
- File upload fields
- Image-based buttons
- Scripted buttons
- Submit and reset buttons

Syntax:

```
<!-- Text and password fields -->
<INPUT TYPE="TEXT|PASSWORD" NAME="field_name" VALUE="default_value"
  SIZE="field_size" MAXLENGTH="maximum_input_length"

  DISABLED READONLY>
```

or

```
<!-- Hidden field -->

<INPUT TYPE="HIDDEN" NAME="field_name" VALUE="field_value">
```

or

```
<!-- Checkbox -->
<INPUT TYPE="CHECKBOX" NAME="field_name" VALUE="field_value"

  CHECKED DISABLED>
```

or

```
<!-- Radio button -->
<INPUT TYPE="RADIO" NAME="field_name" VALUE="field_value"
```

```
  CHECKED DISABLED>
```

or

```
<!-- File upload -->
<INPUT TYPE="FILE" NAME="field_name" VALUE="default_value"

  ACCEPT="acceptable_MIME_types" DISABLED>
```

or

```
<!-- Image-based button -->
<INPUT TYPE="IMAGE" SRC="URL_of_image_file" ALT="text_description"

  ALIGN="TOP|MIDDLE|BOTTOM|LEFT|RIGHT" USEMAP="map_name" DISABLED>
```

or

```
<!-- Scripted button -->
<INPUT TYPE="BUTTON" VALUE="button_label" onclick="script_name"

  DISABLED>
```

or

```
<!-- Submit/reset button -->

<INPUT TYPE="SUBMIT|RESET" VALUE="button_label" DISABLED>
```

Attributes:

The <INPUT> tag is easily the most versatile of all the HTML tags. It has a large number of attributes, although not all are applicable in every situation. The following list examines each variant of the <INPUT> tag (which corresponds to changing values of the TYPE attribute) and notes what each applicable attribute does in that situation.

- Text and password fields (TYPE="TEXT|PASSWORD")--The NAME attribute gives the input field a unique name so it can be identified by the processing script. The VALUE attribute is appropriate for a text field when you want to prepopulate the field with a default value. LENGTH is set equal to the number of characters wide the input field should be onscreen. MAXLENGTH sets an upper limit on how many characters long the input from the field can be. The DISABLED attribute deactivates the field, and READONLY leaves the field active while disallowing the user from typing any new input into it.

- Hidden fields (TYPE="HIDDEN")--NAME and VALUE specify the name of the field and the value to pass to the server.

- Check box (TYPE="CHECKBOX")--NAME gives the check box field a unique name, and VALUE is set equal to the value you want passed to the server if the box is checked. Including CHECKED makes the box preselected, and DISABLED disables the check box altogether.

- Radio buttons (TYPE="RADIO")--NAME gives a name to the entire set of radio buttons. All buttons can have the same NAME because their corresponding VALUEs have to be mutually exclusive options. The CHECKED attribute preselects a radio button and DISABLED shuts down the radio button.

- File upload (TYPE="FILE")--NAME gives the field a unique name, and VALUE is set to the default value of the field (presumably a filename). The ACCEPT attribute provides a set of acceptable MIME types for upload. Specifying the DISABLED attribute deactivates the field.

- Image-based button (TYPE="IMAGE")--The SRC attribute tells the browser where it can find the image file for the button. ALT provides a text-based alternative to the image should the image file not be available. You can use ALIGN to control how the image is aligned on the page. USEMAP is set equal to a client-side imagemap name, enabling you to take different actions depending on where the user clicks. Using the DISABLED attribute shuts off the button.

- Scripted button (TYPE="BUTTON")--Whatever you specify for the VALUE attribute will be the text that appears on the face of the button. The onclick attribute is set equal to the name of the script that is to execute when the button is clicked. If you specify the DISABLED attribute, the scripted button will be deactivated.

- Submit and reset buttons (TYPE="SUBMIT|RESET")--The VALUE attribute specifies what text to place on the button. If DISABLED, the submit or reset button will be turned off.

Additionally, you can use the following attributes with the <INPUT> tag:

- ACCESSKEY--Defines a shortcut key combination that the user can press to give focus to the input field (see the attribute listing for the <A> tag for more details).

- TABINDEX--Defines the input field's position in the tabbing order of the page.

Example:

```
<FORM ACTION="/cgi-bin/submit_it.pl">
Login Name: <INPUT TYPE="TEXT" NAME="login" SIZE=12>
Password: <INPUT TYPE="PASSWORD" NAME="passwd" SIZE=12>
<INPUT TYPE="HIDDEN" NAME="browser" VALUE="IE4">
Sex: <INPUT TYPE="RADIO" NAME="sex" VALUE="F">Female
     <INPUT TYPE="RADIO" NAME="sex" VALUE="M">Male
<INPUT TYPE="BUTTON" VALUE="Check data" onclick="validate()">
<INPUT TYPE="SUBMIT" VALUE="Login">
<INPUT TYPE="RESET" VALUE="Clear">

</FORM>
```

<SELECT>

Type:

Container

Function:

Sets up a list of choices from which a user can select one or many.

Syntax:

```
<SELECT NAME="field_name" SIZE="visible_rows" MULTIPLE DISABLED
 ACCESSKEY="shortcut_key_letter" TABINDEX="tab_position">
...

</SELECT>
```

Attributes:

You can use the following attributes with the <SELECT> tag:

- ACCESSKEY--Defines a shortcut key combination that the user can press to give focus to the select field (see the attribute listing for the <A> tag for more details).
- DISABLED--Deactivates the field.
- MULTIPLE--Enables the user to choose more than one of the options by holding down the Ctrl key and clicking.
- NAME--Gives the field a unique name so it can be identified by the processing script.
- SIZE--Set equal to the number of options that should be visible on the screen.
- TABINDEX--Defines the select field's position in the tabbing order of the page.

NOTE: If you set SIZE=1 and don't specify MULTIPLE, the field will be displayed as a drop-down list. Otherwise, the field appears as a scrollable list of options.

Example:

```
<SELECT NAME="size" SIZE=4>
<OPTION>Small</OPTION>
<OPTION>Medium</OPTION>
<OPTION>Large</OPTION>
<OPTION>X-Large</OPTION>
...

</SELECT>
```

Related Tags:

Individual options in the list are specified using the <OPTION> tag. You can also use the <OPTGROUP> tag to place options into logical groups.

<OPTION>

Type:

Container

Function:

Defines an option in a <SELECT> field listing.

Syntax:

```
<OPTION VALUE="option_value" SELECTED DISABLED LABEL="label_text">
... option text ...

</OPTION>
```

Attributes:

The <OPTION> tag takes the following attributes:

- DISABLED--Makes the option unavailable.
- LABEL--Provides a short label for the menu option. If specified, this label is used in place of the option text itself.
- SELECTED--Preselects an option.
- VALUE--Specifies a value to pass to the browser if the option is selected. If no VALUE is given, the browser will pass the option text to the server for processing.

Example:

```
<SELECT NAME="state" SIZE=5>
<OPTION VALUE="AL">Alabama</OPTION>
<OPTION VALUE="NM" SELECTED>New Mexico</OPTION>
<OPTION VALUE="OK">Oklahoma</OPTION>
...

</SELECT>
```

Related Tags:

The <OPTION> tag is valid only between the <SELECT> and </SELECT> tags. You can place options into logical groups by using the <OPTGROUP> tag.

<OPTGROUP>

Type:

Container

Function:

Defines a logical group of select list options.

Syntax:

```
<OPTGROUP LABEL="label_text" DISABLED>
<OPTION> ... </OPTION>
<OPTION> ... </OPTION>
<OPTION> ... </OPTION>
...

</OPTGROUP>
```

Attributes:

<OPTGROUP> can take two attributes:

- DISABLED--Disables the options in the group.
- LABEL--Specifies a label for the option group.

Example:

```
<OPTGROUP LABEL="months">
<OPTION VALUE="Jan">January</OPTION>
<OPTION VALUE="Feb">February</OPTION>
<OPTION VALUE="Mar">March</OPTION>
...

</OPTION>
```

Related Tags:

The <OPTGROUP> tag should be used only inside the <SELECT> and </SELECT> tags. The only tag allowable inside the <OPTGROUP> and </OPTGROUP> tags is the <OPTION> tag.

<TEXTAREA>

Type:

Container

Function:

Sets up a multiple-line text input window.

Syntax:

```
<TEXTAREA NAME="field_name" ROWS="number_of_rows"
  COLS="number_of_columns" DISABLED READONLY
```

```
  ACCESSKEY="shortcut_key_letter" TABINDEX="tab_position">
... default text to appear in window ...

</TR>
```

Attributes:

The <TEXTAREA> tag can take the following attributes:

- ACCESSKEY--Defines a shortcut key combination that the user can press to give focus to the text input window (see the attribute listing for the <A> tag for more details).
- COLS--Set equal to the number of columns wide the text window should be.
- DISABLED--Deactivates the text window.
- NAME--Assigns a unique name to the input window so that the processing program can identify it.
- READONLY--Leaves the window active, but the user will not be able to change the default text that is displayed.
- ROWS--Set equal to the number of rows high the text window should be.
- TABINDEX--Defines the text window's position in the tabbing order of the page.

Example:

```
<TEXTAREA NAME="feedback" ROWS=10 COLS=40>
We appreciate your comments!  Please delete this
text and type in your feedback.

</TEXTAREA>
```

<BUTTON>

Type:

Container

Function:

Places a button on the form. This type of button is different from the one rendered by <INPUT> because it has improved presentation features, such as three-dimensional rendering and up/down movement when clicked.

Syntax:

```
<BUTTON TYPE="SUBMIT|RESET|BUTTON" NAME="button_name" VALUE="button_value"
  DISABLED ACCESSKEY="shortcut_key_letter" TABINDEX="tab_position">
... text for button face or <IMG> tag ...
```

```
</BUTTON>
```

If text is placed between the <BUTTON> and </BUTTON> tags, that text will appear on the face of the button. If an <IMG> tag is placed between <BUTTON> and </BUTTON>, the image will be used as the button.

Attributes:

You can use the following attributes with the <BUTTON> tag:

- ACCESSKEY--Defines a shortcut key combination that the user can press to click the button (see the attribute listing for the <A> tag for more details).
- DISABLED--Disables the button.
- NAME--Gives the button a unique name.
- TABINDEX--Defines the button's position in the tabbing order of the page.
- TYPE--Set to SUBMIT, RESET, or BUTTON, depending on the type of button you're defining. TYPE="BUTTON" is typically used for defining a scripted button.
- VALUE--Specifies what is passed to the server when the button is clicked.

Example:

```
<BUTTON NAME="validate" VALUE="form_validation" onClick="validate();">
Click here to validate your input.

</BUTTON>
```

<LABEL>

Type:

Container

Function:

Denotes a form field label. Labels are typically text next to the field that prompts the user for the type of input expected. This works fine for text-based browsers, but it makes forms inaccessible for users who are visually impaired and who use speech-based or Braille browsers. Marking field labels with the <LABEL> tag makes it possible to prompt these users for the necessary input.

Syntax:

```
<LABEL FOR="field_ID" ACCESSKEY="shortcut_key_letter">
... label text goes here ...
```

```
</LABEL>
```

Attributes:

The <LABEL> tag takes the following attributes:

- ACCESSKEY--Defines a shortcut key combination that the user can press to give focus to the label (see the attribute listing for the <A> tag for more details).
- FOR--Set equal to the value of the ID attribute for the field that goes with the label.

Example:

```
<LABEL FOR="PW" ACCESSKEY="P">Enter your password:</LABEL>

<INPUT TYPE="PASSWORD" ID="PW" NAME="passwd">
```

Related Tags:

<LABEL> is typically used with the <INPUT>, <SELECT>, or <TEXTAREA> tags.

<FIELDSET>

Type:

Container

Function:

Groups related form input fields.

Syntax:

```
<FIELDSET>
... related input fields ...

</FIELDSET>
```

Attributes:

None.

Example:

```
<FIELDSET>
Login: <INPUT TYPE="TEXT" NAME="login">
Password: <INPUT TYPE="PASSWORD" NAME="passwd">

</FIELDSET>
```

Related Tags:

The <LEGEND> tag can be used to give a field grouping a specific name.

<LEGEND>

Type:

Container

Function:

Names a group of related form fields.

Syntax:

```
<LEGEND ALIGN="LEFT|RIGHT|TOP|BOTTOM" ACCESSKEY="shortcut_key_letter">
... legend text goes here ...

</LEGEND>
```

Attributes:

The <LEGEND> tag has two attributes:

- ACCESSKEY--Defines a shortcut key combination that the user can press to give focus to the legend (see the attribute listing for the <A> tag for more details).
- ALIGN--Controls how the legend text is horizontally aligned with respect to the group of fields and can be set equal to LEFT, RIGHT, TOP, or BOTTOM. ALIGN has been deprecated in favor of using style sheet information to align a legend.

Example:

```
<FIELDSET>
<LEGEND ALIGN="TOP">User Login Information</LEGEND>
Login: <INPUT TYPE="TEXT" NAME="login">
Password: <INPUT TYPE="PASSWORD" NAME="passwd">

</FIELDSET>
```

Related Tags:

<LEGEND> gives a name to a set of fields grouped together by the <FIELDSET> tag.

Frame Tags

Framed layouts are ones in which the browser window is broken into multiple regions called *frames*. Each frame can contain a distinct HTML document, enabling you to display several documents at

once, rather than just one (see Figure 3.19).

FIGURE 3.19 *Frames enable you to keep key page elements (such as navigation) on the screen all the time, while other parts of the page change.*

You need to know only a few tags to set up a framed page. These tags are covered in this section.

SEE "Frames," **p. 211**.

<FRAMESET>

Type:

Container

Function:

Divides the browser window into frames.

Syntax:

```
<FRAMESET ROWS="list_of_row_sizes" COLS="list_of_column_sizes">
...

</FRAMESET>
```

Attributes:

<FRAMESET> can take the ROWS or COLS attribute, but not both at the same time. ROWS specifies how the browser screen should be broken up into multiple rows. ROWS is set equal to a list of values that describe the size of each row. The number of items in the list determines how many rows there will be. The values in the list determine the size of each row. Sizes can be in pixels, percentages of screen depth, or relative to the amount of available space. COLS works the same way, except it will divide the screen into columns.

NOTE: If you try to use ROWS and COLS in the same <FRAMESET> tag, a browser will typically use the first attribute it finds and ignore the second.

Example:

```
<!-- Divide the screen into four rows: 125 pixels, 30% of screen,
     88 pixels, and whatever is left over. -->
<FRAMESET ROWS="125,30%,88,*">
...

</FRAMESET>
```

Related Tags:

<FRAMESET> only breaks up the screen into multiple regions. You need to use the <FRAME> tag to populate each frame with content. Also, you can use the <NOFRAMES> tag to specify alternative content for browsers that cannot process frames.

> **NOTE:** <FRAMESET> tags may be nested to create even more complex layouts.

<FRAME>

Type:

Standalone

Function:

Places content into a frame.

Syntax:

```
<FRAME SRC="URL_of_document" NAME="frame_name" FRAMEBORDER="0|1"
  MARGINWIDTH="width_in_pixels" MARGINHEIGHT="height_in_pixels"

  NORESIZE SCROLLING="YES|NO|AUTO" LONGDESC="URL_of_description">
```

Attributes:

The <FRAME> tag can take several attributes:

- FRAMEBORDER--Setting FRAMEBORDER to 1 turns on the frame's borders; setting it to 0 turns them off.
- LONGDESC--Set equal to the URL of a resource that contains a more detailed description of the frame's content.
- MARGINHEIGHT--Specifies the size (in pixels) of the top margin of the frame.
- MARGINWIDTH--Specifies the size (in pixels) of the left margin of the frame.
- NAME--Gives the frame a unique name so it can be targeted by other tags (such as <A>, <FORM>, and <AREA>).
- NORESIZE--Suppresses the user's ability to drag and drop a frame border in a new location.
- SCROLLING--Controls the presence of scrollbars on the frame. Setting SCROLLING to YES makes the browser always put scrollbars on the frame, setting it to NO suppresses the scrollbars, and setting it to the default of AUTO enables the browser to decide whether the scrollbars are needed.
- SRC--Tells the browser the URL of the HTML file to load into the frame. SRC is a required

attribute of the <FRAME> tag.

Example:

```
<FRAMESET COLS="25%,75%"> <!-- Make 2 columnar frames -->
   <!-- Populate frame #1 -->
   <FRAME SRC="leftframe.html" NORESIZE NAME="left" FRAMEBORDER=0>
   <!-- Populate frame #2 -->
   <FRAME SRC="rightframe.html" NORESIZE NAME="right" FRAMEBORDER=0>
...

</FRAMESET>
```

Related Tags:

The <FRAME> tag is valid only between the <FRAMESET> and </FRAMESET> tags.

<NOFRAMES>

Type:

Container

Function:

Provides an alternative layout for browsers that cannot process frames.

Syntax:

```
<NOFRAMES>
... non-frames content goes here ...

</NOFRAMES>
```

Attributes:

None.

Example:

```
<FRAMESET COLS="25%,75%"> <!-- Make 2 columnar frames -->
   <!-- Populate frame #1 -->
   <FRAME SRC="leftframe.html" NORESIZE NAME="left" FRAMEBORDER=0>
   <!-- Populate frame #2 -->
   <FRAME SRC="rightframe.html" NORESIZE NAME="right" FRAMEBORDER=0>
<NOFRAMES>
Your browser cannot process frames.  Please visit the
<A HREF="/noframes/index.html">non-frames version</A>
of our site.
</NOFRAMES>

</FRAMESET>
```

Related Tags:

<NOFRAMES> is valid only between the <FRAMESET> and </FRAMESET> tags. Your <NOFRAMES> content should be specified before any nested <FRAMESET> tags.

<IFRAME>

Type:

Container

Function:

Places a floating frame on a page. *Floating frames* are best described as "frames that you can place like images."

Syntax:

```
<IFRAME SRC="URL_of_document" NAME="frame_name" FRAMEBORDER="0|1"
  WIDTH="frame_width_in_pixels_or_percentage"
  HEIGHT="frame_height_in_pixels_or_percentage"
  MARGINWIDTH="margin_width_in_pixels"
  MARGINHEIGHT="margin_height_in_pixels"
  SCROLLING="YES|NO|AUTO" ALIGN="TOP|MIDDLE|BOTTOM|LEFT|RIGHT"
  LONGDESC="URL_of_description">
... text or image alternative to the floating frame ...

</IFRAME>
```

Attributes:

The <IFRAME> tag can take the following attributes:

- ALIGN--Controls how the floating frame is aligned, and can be set to TOP, MIDDLE, BOTTOM, LEFT, or RIGHT. TOP, MIDDLE, and BOTTOM alignments make text appear next to the frame, starting at the top, middle, or bottom of the frame. Setting ALIGN to LEFT or RIGHT floats the frame in the left or right margin and enables text to wrap around it.
- FRAMEBORDER--Setting FRAMEBORDER to 1 turns on the floating frame's borders; setting it to 0 turns them off.
- HEIGHT--Specifies the height of the floating frame in pixels.
- LONGDESC--Set equal to the URL of a resource that contains more detail about the contents of the floating frame.
- MARGINHEIGHT--Specifies the size (in pixels) of the top margin of the floating frame.
- MARGINWIDTH--Specifies the size (in pixels) of the left margin of the floating frame.

- NAME--Gives the floating frame a unique name so it can be targeted by other tags (such as <A>, <FORM>, and <AREA>).
- SCROLLING--Controls the presence of scrollbars on the floating frame. Setting SCROLLING to YES makes the browser always put scrollbars on the floating frame, setting it to NO suppresses the scrollbars, and setting it to the default of AUTO enables the browser to decide whether the scrollbars are needed.
- SRC--Tells the browser the URL of the HTML file to load into the floating frame. SRC is a required attribute of the <IFRAME> tag.
- WIDTH--Specifies the width of the floating frame in pixels.

Example:

```
<IFRAME SRC= "float_content.html " WIDTH= "50% " HEIGHT= "50% " ALIGN="RIGHT"
  SCROLLING= "NO " NAME= "floater " FRAMEBORDER=1>
Your browser does not support floating frames. :(

</IFRAME>
```

Executable Content Tags

One of the ways in which Web pages have become more dynamic is through their support of executable content, such as Java applets and ActiveX controls. These page elements are downloaded to the browser and run in its memory space to produce dynamic content on the browser screen.

HTML 4.0 supports two ways for placing executable content: the <APPLET> tag for Java applets and the <OBJECT> tag for other executable objects. These tags, along with the supporting <PARAM> tag, are profiled in this section.

<APPLET>

Type:

Container

Function:

Places a Java applet on a page. The <APPLET> tag has been deprecated in favor of using the more generic <OBJECT> tag to place applets.

Syntax:

```
<APPLET WIDTH= "width_in_pixels " HEIGHT= "height_in_pixels "
  CODEBASE= "base_URL_for_applet " CODE= "applet_class file "
  OBJECT= "serialized_applet_file " NAME= "applet_name "
  ARCHIVE= "archive_list " ALT= "text_alternative "
  ALIGN= "TOP|MIDDLE|BOTTOM|LEFT|RIGHT "
  HSPACE= "pixels " VSPACE= "pixels ">
```

```
...

</APPLET>
```

Attributes:

As the following list demonstrates, many of the <APPLET> tag's attributes are the same as those for the <IMG> tag:

- ALIGN--Positions adjacent text at the TOP, MIDDLE, or BOTTOM of the applet window, or you can float the window in the LEFT or RIGHT margin.
- ALT--Provides a text-based alternative to the applet.
- ARCHIVE--Set equal to a comma-delimited list of archive locations.
- CODE--Specifies the class file.
- CODEBASE--Set equal to the URL of the code.
- HEIGHT--Specifies the height of the applet window in pixels. HEIGHT is a required attribute of the <APPLET> tag.
- HSPACE--Controls the amount of whitespace (in pixels) to the left and right of the applet window.
- NAME--Gives the applet a unique name so that it can be referenced by other Java applets.
- OBJECT--Provides the name of a serialized applet file.
- VSPACE--Controls the amount of whitespace (in pixels) above and below the applet window.
- WIDTH--Specifies the width of the applet window in pixels. WIDTH is a required attribute of the <APPLET> tag.

NOTE: Either the CODE or the OBJECT attribute must be used in an <APPLET> tag. If they are both used and they each specify a different class name, the browser should return an error message.

Example:

```
<APPLET WIDTH=250 HEIGHT=200 CODE= "marquee.class " NAME= "marquee "
  ALT= "Scrolling text marquee applet " ALIGN= "RIGHT "
  HSPACE=5 VSPACE=12>
  <PARAM NAME= "message " VALUE= "Hello World! ">
...

</APPLET>
```

Related Tags:

Parameters are passed to a Java applet using the <PARAM> tag.

<PARAM>

Type:

Standalone

Function:

Passes a parameter to a Java applet (<APPLET>) or other executable object (<OBJECT>).

Syntax:

```
<PARAM ID= "unique_identifier " NAME= "parameter_name "
 VALUE= "parameter_value " VALUETYPE= "DATA|REF|OBJECT "

 TYPE= "expected_content_type ">
```

Attributes:

The <PARAM> tag can take the following attributes:

- ID--Assigns a unique identifying name to the parameter.
- NAME--Provides the name of the parameter.
- TYPE--Tells the browser what the parameter's Internet media (MIME) type is.
- VALUE--Specifies the value of the parameter.
- VALUETYPE--Provides more detail about the nature of the VALUE being passed and can be set to DATA, REF, or OBJECT.

Example:

```
<APPLET WIDTH=300 HEIGHT=224 CODE= "test.class ALT= "Test applet "
   ALIGN= "TOP " NAME= "test " >
   <PARAM ID= "P1 " NAME= "tolerance " VALUE= "0.001 " VALUETYPE= "DATA ">
   <PARAM ID= "P2" NAME= "pi " VALUE= "3.14159 " VALUETYPE= "DATA ">
...

</APPLET>
```

Related Tags:

<PARAM> tags can be used only between the <APPLET> and </APPLET> tags or between the <OBJECT> and </OBJECT> tags.

<OBJECT>

Type:

Container

Function:

Places an executable object on a page.

Syntax:

```
<OBJECT CLASSID= "implementation_info " CODEBASE= "URL_of_object "
  CODETYPE= "MIME_type " DATA= "URL_to_data " TYPE= "data_MIME_type "
  ARCHIVE= "list_of_archives " USEMAP= "map_name " TABINDEX= "tab_position "
  STANDBY= "message_while_loading " DECLARE
  ALIGN= "TEXTTOP|MIDDLE|TEXTMIDDLE|BASELINE|TEXTBOTTOM|LEFT|CENTER|RIGHT "
  WIDTH= "width_in_pixels_or_percentage " NAME= "object_name "
  HEIGHT= "height_in_pixels_or_percentage "
  HSPACE= "pixels " VSPACE= "pixels " BORDER= "pixels ">
...

</OBJECT>
```

Attributes:

The <OBJECT> tag has an exhausting list of attributes, but many of them are the same as those for the <IMG> tag, so they are fairly easy to understand:

- ALIGN--Controls how content adjacent to the object area is aligned. Note that this ALIGN attribute has many more possible values than ALIGN attributes for other tags.
- WIDTH and HEIGHT--Specifies the dimensions of the object area as a number of pixels or as a percentage of available space.
- BORDER--Set equal to the number of pixels that the border thickness should be.
- HSPACE and VSPACE--Controls the amount of whitespace around the object area.

Additionally, <OBJECT> can take these attributes:

- ARCHIVE--Set equal to a comma-delimited list of archive locations.
- CLASSID--Identifies which implementation or release of the object you're using.
- CODEBASE--Set equal to the URL of the object.
- CODETYPE--Describes the code's MIME type.
- DATA--Set equal to list of URLs where data for the object can be found.
- DECLARE--Instructs the browser to declare, but not instantiate, a flag for the object.

- STANDBY--Enables you to display a message to the user while the object is loading.
- TYPE--Specifies the MIME type of the data passed to the object.
- USEMAP--Points to client-side map data, if imagemaps are used.

Example:

```
<OBJECT WIDTH=100% HEIGHT=100 CODETYPE= "application/x-oleobject "
  CLASSID= "CLSID: 1A4DA620-6217-11CF-BE62-0080C72EDD2D "
  CODEBASE= "http://activex.microsoft.com/controls/iexplorer/marquee.ocx "
  HSPACE=5 VSPACE=10 ALIGN=MIDDLE BORDER=0>
  <PARAM NAME= "image " VALUE= "greeting.gif ">
  <PARAM NAME= "speed " VALUE= "7 ">
  <PARAM NAME= "repeat " VALUE= "1 ">
...

</OBJECT>
```

Related Tags:

Parameters passed to the object are given by the <PARAM> tag.