Chapter 1. Designing Web Pages with HTML 4.0

- 1.1 The HyperText Markup Language
- 1.2 HTML 4.0 and Other HTML Standards
- 1.3 Steps to Publish a Document on the Web
- 1.4 The Basic Structure of HTML Documents
- 1.5 HEAD-High-Level Information About the Page
- 1.6 BODY-Creating the Main Document
- 1.7 Summary

Topics in This Chapter

- An overview of the HyperText Markup Language
- Comparison of HTML 4.0 with other HTML specifications
- Validation of HTML documents
- The process of creating and publishing a Web page
- The fundamental structure of HTML documents
- Common elements in the header of an HTML document
- Inclusion of keyword information for search engines
- Use of the BODY tag to set up the basic look of the page

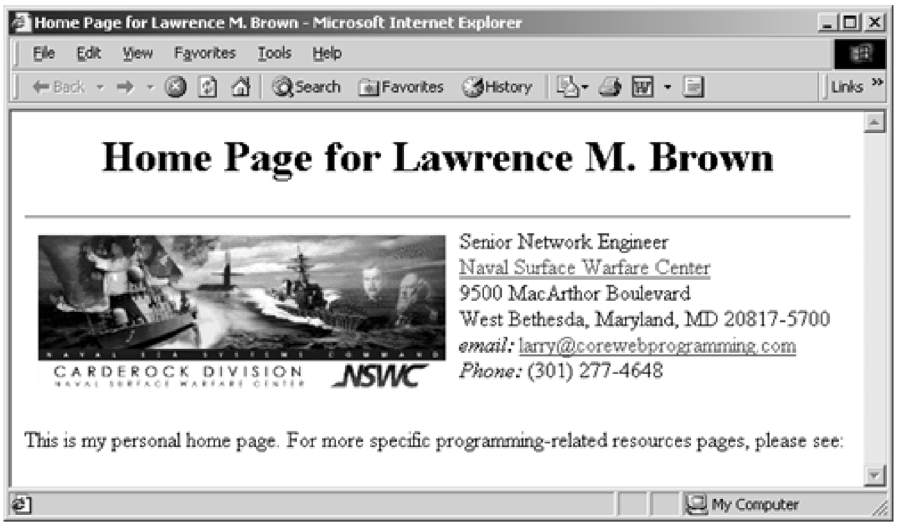
This is the first of five chapters that cover the HyperText Markup Language (HTML). Together, they teach you the techniques for creating professional Web pages. This first chapter examines the underlying structure of an HTML document, and the remaining four chapters cover additional topics for building quality Web pages, including block-level elements, text-level elements, frames, and cascading style sheets. Once you've learned these basics, the remainder of *Core Web Programming* covers advanced topics like Java applets, sockets, database connectivity, Java servlets, JavaServer Pages (JSP), and processing XML. All of these technologies are critical in developing quality Web sites.

The history of the HyperText Markup Language is briefly covered in this chapter, followed by the steps necessary to create and publish a WWW page. We then focus on the general structure of Web pages, describe which HTML elements are required in all documents, and explain how to specify settings that affect the document as a whole. In addition, we show you how to validate your Web page as legal HTML and how to record information in your Web page for use by search engines.

1.1 The HyperText Markup Language

Web pages are created with the HyperText Markup Language, which lets you mix regular text with "markup" tags describing the text. These tags can describe the *appearance* (display this in red) or *layout* (arrange the following in a 3-row, 4-column table) of the text, but the majority simply describe the *content* (this is a main heading) and leave many of the appearance and layout decisions to the browser. For example, Listing 1.1 shows the HTML document used to create the Web page shown in Figure 1-1. For now, don't worry about the details of each of the HTML elements; they are discussed in detail in the rest of Part 1 of this book. However, even at first glance you can pick out some basic features, such as the mix of regular text and elements enclosed in angle brackets and that some but not all the elements come in pairs of the form `<NAME>` and `</NAME>`.

Figure 1-1. This page, rendered in Internet Explorer 5.0 on Windows 2000 Professional, is the result of the document in Listing 1.1.


Home Page for Lawrence M. Brown - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History Links
My Computer

Home Page for Lawrence M. Brown

Senior Network Engineer
Naval Surface Warfare Center
9500 MacArthor Boulevard
West Bethesda, Maryland, MD 20817-5700
email: larry@corewebprogramming.com
Phone: (301) 277-4648

This is my personal home page. For more specific programming-related resources pages, please see:

The Web page shown in Figure 1-1 is the result for a particular browser (Internet Explorer 5.0) on a particular operating system (Windows 2000 Professional) with the browser and desktop preferences (font face, size, and color) set by the user. In addition to honoring user customizations, browsers usually have wide latitude in how they implement the various types of elements, and authors who try to enforce an exact appearance for pages that will be viewed by multiple browsers often end up frustrated. In Chapter 5 (Cascading Style Sheets), you will see a new standard that gives authors a high degree of control over the final look of their pages. But even with style sheets, authors of general Web pages should realize that they cannot control all aspects of the final appearance of their page.

Listing 1.1 An HTML document for a simple home page

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <Title>Home Page for Lawrence M. Brown</Title>
</HEAD>
<BODY BGCOLOR="WHITE">
<H1 ALIGN="CENTER">Home Page for Lawrence M. Brown</H1>
<HR>
<IMG SRC="images/navsea-nswc.gif" WIDTH=300 HEIGHT=117
     HSPACE=10 VSPACE=5 ALIGN="LEFT" ALT="NSWC Logo">
Senior Network Engineer<BR>
<A HREF="http://www.dt.navy.mil/">
Naval Surface Warfare Center</A><BR>
9500 MacArthor Boulevard<BR>
West Bethesda, Maryland, MD 20817-5700<BR>
<I>email:</I> <A HREF="mailto:larry@corewebprogramming.com">
larry@corewebprogramming.com</A><BR>
<I>Phone:</I> (301) 277-4648<BR CLEAR="ALL">
<P>
This is my personal home page. For more specific
programming-related resources pages, please see:
<!-- Rest of Sample Page Deleted -->
</BODY>
</HTML>
```

Core Note

Trying to enforce an exact look for pages that will be viewed by people using a *variety of browsers will only lead to frustration.*

The page design issue is further complicated because Web browsers are not the only type of programs that use HTML documents. A variety of applications can display, print, index, or even synthesize speech based on an HTML document. In this book, however, we concentrate on how HTML documents are used by WWW browsers.

1.2 HTML 4.0 and Other HTML Standards

Along with browser issues, to a lesser degree, authors must also contend with changes in the HTML specification. Until January 1997, HTML 2.0, introduced by the World Wide Web Consortium (W3C), was the most up-to-date standard available. The HTML 2.0 specification describes the capabilities supported by most browsers as of mid-1994. Even before the HTML 2.0 was published, work was under way to define the next-generation specification, known first as HTML+ and later as HTML 3.0. Unfortunately, the vendors that emerged as the dominant players in the Web browser world could not agree on supporting HTML 3.0, and the effort was dropped. Instead, HTML 3.2 (originally code-named "Wilbur"), an intermediate specification, was drafted. Perhaps the specification should have been called "HTML 2.3," because it actually supports fewer features than HTML 3.0. Soon afterwards, in December of 1997, HTML 4.0 became the standard of choice.

Technically, no current browsers completely support the HTML 4.0 specification. However, Netscape 4.0 and later, as well as Internet Explorer 4.0 and later, are mostly compliant with HTML 4.0 specification (release 3.x of those browsers supports only HTML 3.2). Compared to HTML 3.2, the major changes in HTML 4.0 include:

- Addition of document frames
- A preference for style sheets instead of formatting elements and attributes
- Improved cell alignment and row/column grouping in tables
- Mouse and keyboard events for nearly *all* elements
- Internationalization features
- Improved support for nonvisual browsers

Since the HTML 4.0 standard was accepted in 1997, the W3C introduced HTML 4.01, which fixes a few typographical errors and adds a couple of omitted attributes. The HTML 4.01 specification was accepted by the W3C on December 24, 1999. More importantly, the new XHTML 1.0 specification, designed to support XML in Web pages, heavily relies on the HTML 4.01 specification for the meaning of the HTML tags. XML allows you to identify unique structures in your document and to define your own markup formats for the identified structures. XML is beneficial to rich data formats common to e-commerce transactions, vector graphics, and application servers. XML is discussed in Chapter 23 (XML Processing with Java). The XHTML 1.0 specification became official on January 26, 2000.

The official HTML on-line specifications are available from the following locations:

HTML 4.01

http://www.w3.org/TR/html401/

HTML 4.0

http://www.w3.org/TR/1998/REC-html40-19980424/

HTML 3.2

http://www.w3.org/TR/REC-html32.html

XHTML 1.0

http://www.w3.org/TR/xhtml1/

HTML supported by Netscape

http://developer.netscape.com/library/documentation/htmlguid/index.htm

HTML supported by Internet Explorer

http://msdn.microsoft.com/workshop/author/html/reference/elements.asp

1.3 Steps to Publish a Document on the Web

Three main steps are involved in posting a document to the WWW:

1. Create an HTML document.
2. Place the document in a Web-accessible location.
3. Verify that the HTML is legal and results in the presentation you had in mind.

These steps are explained more thoroughly in the following sections.

Create the Document

Since HTML is formed with normal ASCII text, you can create an HTML document with a text editor, such as Notepad, UltraEdit, emacs, or BBEdit. You can also create documents with an HTML editor, such as HomeSite, FrontPage, or Dreamweaver, or by using a utility that converts an existing word processor document to HTML, such as Microsoft Word or WordPerfect. Since HTML is primarily a logical markup language, not a page layout language, automatically converting a word processor document to HTML often results in a Web page that differs somewhat from the original and requires some hand tuning.

Similarly, because browsers are allowed significant flexibility in how they display many HTML elements and because users can select their preferred fonts and colors, a "WYSIWYG" (What You See Is What You Get) HTML editor is not strictly possible, at least in the absence of cascading style sheets (Chapter 5). Of course, an author could always tailor the display to the default look of a particular browser. Nevertheless, HTML editors are a big timesaver: many let you visually lay out tables and frames; most even support style sheets. If you do use an HTML editor, you will want one that allows you to directly edit the HTML if desired, because you will inevitably want to use some capability not supported by a purely graphical HTML editor. In all cases, once the Web page is created, you will want to check the syntax and appearance before posting on a Web server.

Put the Document on the Web

To be accessible to other Internet users, your document needs to be on a computer that is connected to the Internet and running an HTTP server. If you don't already have Internet access through a work or school system or through a commercial Internet Service Provider (ISP), "The List" (http://thelist.internet.com/) gives information on thousands of ISPs throughout the world. Some authors create their Web pages on the same machine from which the pages will be accessed. This approach is common for work and university computers. Frequently, however, authors create the pages on a home or office PC and then upload the page to the server machine. In this latter case, authors need to be sure that they upload to an accessible location. This process is discussed in the following subsections.

Create a Directory for the File

The computer hosting your Web page will be running an HTTP server. HTTP, the HyperText Transfer

Protocol, is discussed in more detail in Chapter 19 (Server-Side Java: Servlets). For now, you only need to know that HTTP is the protocol by which WWW *clients* (browsers) talk to systems that are hosting Web pages. The program that answers the client's request for files is the HTTP *server.* This server takes the URL (Uniform Resource Locator, which is the Web "address") specified by the client and translates the URL into a specific filename on the server's system. Typically, when a client requests a file from a user's directory, the server looks in a subdirectory such as `public_html` or `www` that is not listed in the URL but is part of the real location of the file. The webmaster or system administrator on the machine or ISP you are using can give the definitive name.

Often, Web sites are leased through an ISP and authors use FTP to access the ISP's server to post their Web pages. Typically, the ISP administrator sets up the initial directory structure for your Web site before giving you an account and password. Other users, especially in a corporate or academic setting, may already have an account on a Web server. For a Unix-based Web server, the user must first create a Web-accessible directory (often `public_html` or `www`) with the `mkdir` command and then post the Web pages to the directory created.

For instance, suppose that user `janedoe` wants to publish a page named `test.html` on the system www.some-isp.com. In such a case, she would put the document in `/home/janedoe/public_html/test.html`, assuming that her home directory is `/home/janedoe` and that `public_html` is the "hidden" directory name used by her Web server. Outsiders could access the page by designating a URL of http://www.some-isp.com/~janedoe/test.html.

The symbol "~" is generally interpreted as "the home directory of." Some programs encode characters by their ASCII or ISO Latin-1 numbers, so you sometimes see "%`7E`" instead of "~". So, don't be confused if you see http://some.host/%7Euser/path/.

Put the File in That Directory

If you are working on the same system that is running the HTTP server, you will probably create your HTML documents directly in the target directory (e.g., `public_html`). If you create the document on a remote system that is on the Internet (e.g., via PPP or a direct connection), you can upload the file by using an FTP client such as `Fetch` on a Mac, `ftp.exe` from the main `Windows` directory on a Windows 95/98/2000/NT system, or `/usr/bin/ftp` on a Unix system. Remember that filenames are case sensitive on many operating systems.

In addition to a default hidden directory name, most HTTP servers also have a default filename that will be used if the URL specifies a directory but no filename. Some common defaults are `index.html`, `Welcome.html`, `default.html`, and variations ending in `.htm` instead of `.html`. For instance, Jane's home page might be `/home/janedoe/public_html/index.html` but could be accessed remotely simply by http://www.some-isp.com/~janedoe/. The URL listing the filename explicitly is also legal, for example, http://www.some-isp.com/~janedoe/index.html.

Set File and Directory Permissions

For the file to be accessible on the Web, the file and directory must be readable by the process running the HTTP server. This often means that they must be world readable because the HTTP server is typically run in an unprivileged mode. Many ISPs that use Unix set the proper permissions by default, so this step might not be necessary if you use a commercial service provider. Unix users can look at their `umask` to determine this setting. If it *is* necessary to set the permissions, the Unix incantation would be:

```
Unix> cd
Unix> chmod a+x .
Unix> cd public_html
Unix> chmod a+x .
Unix> chmod a+r file
```

Because Web browsers can display plain-text documents as well as HTML documents, a good first step is to create a simple file called `test.txt` that contains a single line like "`Hello`," put the file in the Web subdirectory of your account, and try to access the file with http://your.isp.com/~your-account/test.txt. If that works, you don't have to mess around with the cryptic file permission settings.

Validate the Document

Once you place a document on the Web, you want to check that the document has no syntax errors. Checking how the page looks in your Web browser is a good first step but is not sufficient. Browsers try to "guess" what to do when they see incorrect HTML, but different browsers may guess differently. So, if your HTML document contains errors, your browser might guess well and the page might still look fine. But the page might look completely different or have portions not display *at all* on other browsers. Because HTML is defined by the Standard Generalized Markup Language (SGML), a program can use the official SGML specification for HTML to verify that your document is conforming. Some HTML editors do this automatically, or you can submit the URL to a free on-line validator. The most popular validator is the W3C HTML Validation Service. In addition, several free or commercial validators can be installed on your local machine. See the following URLs for details:

The W3C HTML Validation Service

http://validator.w3.org/

The Web Design Group Validation Service

http://www.htmlhelp.com/tools/validator/

The W3C CSS Validation Service

http://jigsaw.w3.org/css-validator/

Core Approach

Trust, but verify: check the syntax of your Web pages with a formal HTML validator.

1.4 The Basic Structure of HTML Documents

HTML *elements* are indicated by markup *tags,* which are enclosed by angle brackets. For instance, `<TITLE>` is the starting tag for the `TITLE` element. The tags are not displayed in the resultant Web page but rather provide descriptive information to the browser. HTML elements can also have *attributes,* which supply additional information in the form *attribute*=*value*. For instance, in the tag `<IMG SRC="images/sample.gif">`, `images/sample.gif` is the *value* of the `SRC` *attribute* of the `IMG` *element.* Most nonalphanumeric characters in attribute values require you to have double quotes around the value, so a wise convention is to always enclose values in double quotes unless the value is a simple integer. All HTML element and attribute names are case insensitive, unlike the attribute values. Extra white space is ignored inside most parts of the document and around element names and attributes but may be significant inside attribute values. Interestingly, in the XHTML 1.0 specification, lowercase is required for elements, and all attributes, including numbers, must be enclosed in quotes. This follows the XML rules; see Chapter 23 (XML Processing with Java).

Some elements are *containers,* which have a start tag (e.g., `<BODY>`) and a corresponding end tag beginning with "`</`" (e.g., `</BODY>`). Other elements are stand-alone and only have a single tag (e.g., `<HR>`). If a browser does not recognize a given HTML element or attribute, then the browser simply ignores it. The browser does not ignore the regular text between unrecognized start and end tags; the browser ignores only the tags themselves. HTML comments are enclosed between `<!--` and `-->`, can span multiple lines, and should not contain a double hyphen (i.e., `--`) in the body of the comment.

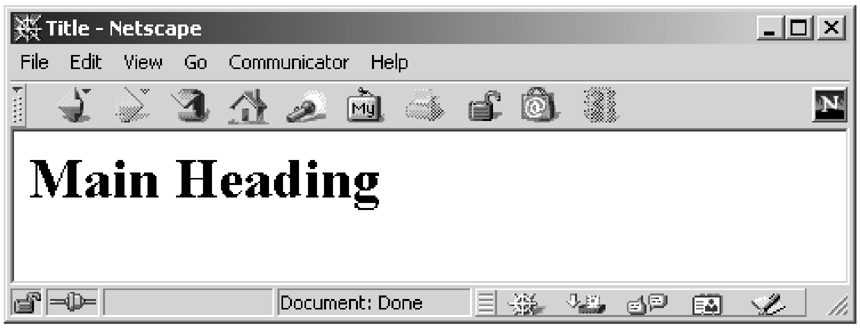
HTML Document Template

HTML documents start with a DOCTYPE declaration that identifies the version of the HTML page to the browser or validation service. After the DOCTYPE statement is an HTML element that contains a HEAD element and a BODY element. The HEAD element must contain a TITLE element. The BODY element, immediately following the HEAD element, often starts with the largest-sized heading (H1), followed by the body of the Web page.

Strictly speaking, only the DOCTYPE and TITLE are required because HTML, HEAD, and BODY can be inferred by the parser. But in practice, the template shown is a recommended starting point for HTML documents, and a good idea is to keep a copy around to insert into new HTML documents if your text editor or HTML editor cannot be configured to insert the tags automatically.

Figure 1-2 shows a standard browser display of the HTML document template in Listing 1.2. The template is a good starting point for all HTML 4.0 documents (except those that use frames; see Chapter 4).

Figure 1-2. A standard HTML 4.0 template, as rendered in Netscape 4.7 on Windows 2000 Professional.



Listing 1.2 `HTML4.0-Template.html`

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Title</TITLE>
</HEAD>

<BODY>
<H1>Main Heading</H1>

<!-- Rest of page goes here -->

</BODY>
</HTML>
```

DOCTYPE Declarations

Depending on the content of the HTML document, one of three DOCTYPE declarations is possible:

Strict HTML 4.01 DOCTYPE

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
```

Transitional HTML 4.01 DOCTYPE

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
          Transitional//EN">
```

Frameset HTML 4.01 DOCTYPE

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">
```

If the document strictly conforms to the HTML 4.01 specification without the use of any deprecated or loosely defined elements and attributes, then add the *strict* `DOCTYPE` statement to the beginning of the HTML document. However, if the document uses deprecated elements (such as fonts and alignments defined outside of a style sheet) or links that target a `FRAME`, then the *transitional* `DOCTYPE` is appropriate. Elements that are deprecated are ones for which a better alternative now exists but are supported for backward compatibility with previous HTML versions. To determine which elements or attributes are deprecated in the HTML 4.01 standard, see the following two URLs:

Index of HTML 4.01 Elements

http://www.w3.org/TR/html4/index/elements.html

Index of HTML 4.01 Attributes

http://www.w3.org/TR/html4/index/attributes.html

If the HTML document defines a set of `FRAME`s, then use the *Frameset* `DOCTYPE`.

Following is a detailed description of the `HTML` element and associated attributes. Throughout this book each HTML element is presented in a reference format clearly stating the syntax for the element, followed by a listing of the legal attributes. The start tag is shown first with all *required* attributes explicitly listed. Ellipses (…) expressed in the start tag indicate that additional material or optional attributes can also be present. The start tag is followed either by the end tag or a statement that no end tag is required for the element. The attributes listed after the element are presented in the order of their usage frequency in documents. Attributes not part of the HTML 4.0 specification but supported by Netscape or Internet Explorer are indicated as "nonstandard." Unless otherwise stated, each HTML element and attribute was tested in English versions of Netscape versions 4.08–4.7 and Internet Explorer versions 4.01–5.5.

HTML Element:	**`<HTML ...> ... </HTML`**
Attributes:	`LANG`, `DIR`, `VERSION`

The `HTML` element immediately follows the `DOCTYPE` and represents a container holding the `HEAD` and `BODY` of the document. The `HTML` tag can optionally contain attributes stating the language of the document, the direction of the text, and the HTML version for the source. Technically, the `HTML` tags are optional because browsers can determine from the source text that the document is HTML. However, including the `HTML` tags is a good, standard practice.

LANG `LANG` defines a language abbreviation to specify the base language of the text. Currently, browsers simply treat the `LANG` attributes as informational and do not change how the text is rendered. However, search engines and spell checkers can take advantage of the language code when examining the document. See RFC 1766 for a listing of language codes. You can access the official RFCs from one of the archieve sites listed at http://www.rfc-editor.org/.

DIR `DIR` specifies the base direction of the text: `LTR` (left-to-right) or `RTL` (right-to-left). Internet Explorer supports the `DIR` attribute; however, depending on the underlying charset encoding, the displayed direction of ASCII characters will vary. Since this new attribute is not supported in Netscape, at least as of Netscape version 4.7, you should reserve `DIR` for intranets that use Internet Explorer exclusively.

Core Warning

Netscape does not support the `DIR` *attribute of the HTML element.*

VERSION The `VERSION` attribute simply states the HTML version of the document. This information is redundant and rarely provided because the browser can determine the HTML version from the `DOCTYPE` statement. The `VERSION` attribute is technically deprecated in the HTML 4.0 standard.

The HTML specification adds `LANG` and `DIR` attributes to *all* HTML elements except `APPLET`, `BASE`, `BASEFONT`, `BR`, `FRAME`, `FRAMESET`, `IFRAME`, `NOFRAMES`, `PARAM`, and `SCRIPT`. The `LANG` and `DIR` attributes are not discussed separately for each HTML element.

1.5 HEAD—High-Level Information About the Page

HTML Element:	**`<HEAD> ... </HEAD>`**
Attributes:	`PROFILE`

The `HEAD` section is the first main section of an HTML document, coming immediately after the `DOCTYPE` declaration and the `<HTML>` start tag and just before the `BODY` section. The `TITLE` element is required and is often the only element inside the `HEAD`. The `PROFILE` attribute does not impact the browser presentation but does provide information for search engines and other user-agents (nonvisual browsers, index servers, development software). The `PROFILE` attribute specifies the URL of a file that provides additional information to search engines for parsing optional `META` data in the `HEAD` section (the `META` element is discussed later in this section). The HTML specification also defines `LANG` and `DIR` attributes for the `HEAD` element. These two attributes are discussed earlier in Section 1.4 on the `HTML` element.

Required HEAD Element

HTML Element:	**`<TITLE> ... </TITLE>`**
Attributes:	None

The `TITLE` element is required in all HTML 4.0 documents and consists of plain text with no other HTML tags. The title can, however, contain character or entity references such as `&copy;` to display © (see Table 2.1). This title normally appears on the bar on the top of the browser window and is typically used in the bookmarks (favorites) and browsing history lists. In addition, Netscape and Internet Explorer add the title of the document in the top banner of printed pages. However, the title does not appear in the main document itself, so should not be confused with the first main heading of the document, often informally viewed by readers as the "title." The HTML 4.0 specification adds `LANG` and `DIR` attributes to the `TITLE` element, but these attributes are not currently supported by Netscape or Internet Explorer. Any use of non-ASCII characters in the title simply results in the document filename being displayed as the title.

Optional HEAD Elements

In addition to the required `TITLE` element, a number of optional elements—`BASE`, `META`, `BGSOUND`, and `LINK`—are occasionally employed by advanced HTML authors. If you're just starting HTML, you're probably better off skipping this part until you have a handle on the more important and more frequently used elements.

HTML Element:	**`<BASE HREF="..." ...>` (No End Tag)**

Attributes:	HREF (required), TARGET

The BASE element specifies the starting location for *relative URLs,* that is, incompletely specified URLs that give the filename without hostname or protocol. The default location is the directory from which the current document was loaded. If a document is copied to a different site but its supporting documents are not, the BASE entry can be used to make sure that relative URLs will still refer to correct locations. The HREF attribute explicitly states the location for all relative URLs. For example, suppose that http://www.microsoft.com/windows2000/Buy-Win2000.html is a document that the authors know will be mirrored at www.apple.com and www.sun.com. In such a case, authors could use a HEAD such as the following:

```
<HEAD>
  <TITLE>Why You Should Buy Windows 2000</TITLE>
  <BASE HREF="http://www.microsoft.com/windows2000/">
</HEAD>
```

You can also supply a TARGET attribute to display the selected link in a new browser window or, when you are using frames, the TARGET attribute can specify the default frame cell in which to display selected links. Authors commonly use `<BASE TARGET="...">` (without the required HREF attribute) to direct all links on a page to the targeted frame cell. See the discussion of frames in Chapter 4 for more details.

HTML Element:	**<META ...> (No End Tag)**
Attributes:	NAME, CONTENT (required), HTTP-EQUIV, SCHEME

META elements can record document information, forward and refresh pages, and include sound files. The exact way in which META tags are used to record document information varies from system to system. However, most information in a META tag is defined by a NAME-CONTENT property pair, where the NAME identifies the property name and the CONTENT identifies the property value. Common entries used with NAME are `author` (person who wrote the document), `description` (brief summary), `keywords` (comma-separated descriptive words to be used by search engines), and `generator` (program that generated the document). Search engines such as Google, Infoseek, and Lycos use the `keywords` entry for their internal indexing (but ignore the entire entry if a word appears more than seven times) and use the `description` entry in lieu of the first part of the document itself when describing the document to the search engine user. The META tag is illustrated in Listing 1.3.

Listing 1.3 A sample HEAD section using META

```
<HEAD>
  <TITLE>Why You Should Buy Windows 2000</TITLE>
  <BASE HREF="http://www.microsoft.com/windows2000/">
  <META id="author" CONTENT="Bill Gates">
  <META id="keywords"
        CONTENT="Windows,Advocacy,OS,Operating Systems">
  <META id="description"
        CONTENT="A summary of the advantages of Windows 2000.">
</HEAD>
```

One use of HTTP-EQUIV is to set automatic refreshing or forwarding of pages by supplying the `Refresh` header. For instance, an on-line newspaper or magazine might have headlines that change periodically and should automatically update every 30 minutes (1800 seconds). Consider a fictional page located at http://www.microsoft.com/windows2000/Buy-Win2000.html. The third line of Listing 1.5 shows how META could automatically reload the same document by specifying `HTTP-EQUIV="Refresh"` and a CONTENT attribute that gives a time of 1800 seconds to reload the page (assuming that the reader remains on the same page for 30 minutes).

Listing 1.4 A Web page with automatic reloading every 30 minutes

```
<HEAD>
  <TITLE>Why You Should Buy Windows 2000</TITLE>
  <META HTTP-EQUIV="Refresh" CONTENT="1800">
</HEAD>
...
```

Instead of reloading the same page, an author might want a document to automatically send readers to a new location (see Listing 1.5). This could happen when the URL of a page has changed and the author wants to leave a forwarding address so that people connecting to the old URL get the new page after five seconds. To do this, use the `Refresh` value but specify a different URL than that of the current page. Be aware however, that on some browsers this breaks the Back button; clicking Back returns to the original page containing the `META` tag, which reforwards to where the users were before they clicked Back.

Listing 1.5 Forwarding to a new destination

```
<HEAD>
  <TITLE>Why You Should Buy Windows 2000 (New Address)</TITLE>
  <META HTTP-EQUIV="Refresh"
        CONTENT="5;
                 URL=http://www.apple.com/Buy-Win2000.html">
</HEAD>
...
```

Note: if the specified URL is a sound file in a format supported by the browser, this method will play the sound file on some browsers. Several browser-specific methods are also available for playing sound files. For instance, Internet Explorer supports the `BGSOUND` element (discussed next) and both Netscape and Internet Explorer support `EMBED` (Section 3.6, "Embedding Other Objects in Documents") for some types of sound files. But be cautious; many users consider it to be a bug, not a feature, when sound files begin playing automatically when users visit a site.

Core Approach

Rather than playing sound files automatically, inform users that a given link will play a sound file or give them a choice of using or not using sound.

`HTTP-EQUIV` can also define the encoding `CHARSET` to use when presenting the text in the browser or user-agent. For example, the following `META` tag indicates that all text should be resolved by the `GB2312` Chinese charset (byte-sequence to character mapping):

```
<META HTTP-EQUIV="Content-type"
      CONTENT="text/html; CHARSET=GB2312">
```

ISO standards define character mappings for many languages. Interestingly, nearly all available charsets include the standard ASCII characters; thus, English text can be added to any document written in another language. However, if the document contains *multiple* languages, then the text should be expressed as 16-bit Unicode characters. A popular encoding of Unicode is UTF-8, where the encoding scheme transforms each 16-bit Unicode character into a variable byte sequence, with the ASCII characters requiring a single 8-bit byte. For additional information on Unicode for expressing multiple languages, see http://www.unicode.org/.

Note that you can assign any HTTP response header to `HTTP-EQUIV` where the `CONTENT` is the corresponding value for the HTTP header. HTTP-EQUIV is simply an HTML extension (that Netscape and Internet Explorer support) that effectively "sets" the response header without formal generation of the particular header as with servlets or CGI. See Chapter 19 (Server-Side Java: Servlets) for further details on HTTP headers.

The SCHEME attribute defines the format of a property value associated with a NAME-CONTENT pair in the META element. Again, the NAME identifies the property name, and the CONTENT identifies the property value. For example in

```
<META SCHEME="Month-Day-Year"
      id="Date"
      VALUE="05-01-2000">
```

the SCHEME attribute can clarify the format of a date property as SCHEME="Month-Day-Year" to distinguish the date from a European format of "Day-Month-Year".

Finally, aside from the use by search engines or other user-agents, the LANG and DIR attributes in a META element are ignored.

HTML Element:	**<BGSOUND SRC="..." ...> (No End Tag)**
Attributes:	SRC (required), LOOP

This is a nonstandard element supported by Internet Explorer for playing sound files. The BGSOUND element can appear in the BODY as well as the HEAD.

SRC SRC specifies the URL of the sound file, which should be in .wav, .au, or MIDI format.

LOOP LOOP specifies how many times the sound file will be repeated. The default is 1. Specifying a value of –1 or INFINITE will result in the sound file playing continuously while the page is open.

HTML Element:	**<SCRIPT TYPE="..." ...> ... </SCRIPT>**
Attributes:	LANGUAGE, SRC, TYPE (required), CHARSET, DEFER

SCRIPT is used for embedded programs, usually in JavaScript. See Chapter 24 (JavaScript: Adding Dynamic Content to Web Pages) for details.

HTML Element:	**<STYLE TYPE="..." ...> ... </STYLE>**
Attributes:	TYPE (required), TITLE, MEDIA

STYLE specifies cascading style sheets, an extremely useful and flexible capability that allows you to specify details about the fonts, colors, backgrounds, margins, and other features used for various elements in the document. See Chapter 5 for a discussion of style sheets.

HTML Element:	**<LINK ...> (No End Tag)**
Attributes:	HREF, REL, REV, TYPE, CHARSET, HREFLANG, MEDIA, ONCLICK, ONDBLCLICK, ONMOUSEDOWN, ONMOUSEUP, ONMOUSEOVER, ONMOUSEMOVE, ONMOUSEOUT, ONKEYPRESS, ONKEYDOWN, ONKEYUP, TARGET, TITLE, ID, CLASS, STYLE

The LINK element provides information on how the current document fits into a larger set of documents by specifying a table of contents location, the previous and next document in the series, an advisory title, and so forth. A HEAD section can contain more than one LINK element.

HREF, REL, REV HREF defines the URL to the linked document. REL gives the relationship of the linked document to the current one; REV gives the reverse relationship, that of the current document to the specified one. The most common types of relationships are CONTENTS, INDEX, GLOSSARY, HELP, NEXT, and PREVIOUS for navigation, MADE for the document author, and STYLESHEET for links to cascading style sheets. For example:

```
<LINK REL=STYLESHEET
      HREF="My-Styles.css"
      TYPE="text/css">
```

TYPE, CHARSET, HREFLANG, MEDIA TYPE defines the MIME type associated with the resource referenced by the link, as in TYPE="text/html". MIME types are described in RFC 1521 (see http://www.rfc-editor.org/ for an up-to-date list of RFC archive sites) and define the format for specific types of files, for example, text, pdf, and gif. The CHARSET attribute indicates the character encoding of the referenced link, for example, CHARSET="ISO-8859-6". The HREFLANG attribute indicates the base language of the link; for example, the syntax HREFLANG="pt" would indicate that the referenced link is written in Portuguese. MEDIA specifies the medium type of the referenced link. Legal values include ALL, AURAL, BRAILLE, HANDHELD, PRINT, PROJECTION, SCREEN (default), SPEECH, TTY, and TV.

The remaining LINK attributes are not supported by current browsers and are not further discussed.

1.6 BODY-Creating the Main Document

HTML documents should have exactly one BODY section defining the main contents of the page. The only exception is a document that uses frames. In such a case, the top-level document simply defines the general layout and specifies which documents go in which frames, omitting the BODY altogether. See Chapter 4 (Frames) for more detail. In nonframes documents, the BODY element contains the text and HTML markup constituting the main document. Because the TITLE portion of the HEAD element appears on the title bar of the window, not in the page itself, and because that title does not always appear on printouts of the document, the BODY normally starts with a "title," often using the largest heading size (H1). The BODY "title" is frequently the exact same text as in the TITLE element. Thus, many HTML documents look like Listing 1.6.

The BODY section uses two major classes of HTML elements, block-level and text-level. The block-level elements normally break the flow of the document and start at the beginning of a new line; block-level elements cause paragraph breaks. Examples of block-level elements include headings, basic text sections, lists, tables, horizontal lines used as dividers, and input forms. Block-level elements can contain text-level elements almost anywhere and nested block-level elements in many places (e.g., list elements and table cells): they are discussed in Chapter 2 (Block-Level Elements in HTML 4.0).

Listing 1.6 HTML 4.0 template with first heading matching title

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>My First Web Page</TITLE>
</HEAD>
<BODY>
<H1>My First Web Page</H1>
<!-- Remainder of HTML Document Here -->
</BODY>
</HTML>
```

The text-level elements don't cause paragraph breaks. Examples of text-level elements include tags for fonts, hypertext links, embedded images, applets, plug-ins, ActiveX components, and image maps. Text-level elements can be embedded inside block-level elements and often nested in other text-level elements but cannot contain block-level elements. They are discussed in Chapter 3 (Text-Level Elements in HTML 4.0).

HTML Element:	**`<BODY ...> ... </BODY>`**
Attributes:	`BACKGROUND`, `BGCOLOR`, `TEXT`, `LINK`, `VLINK`, `ALINK`, `TITLE`, `ONCLICK`, `ONDBLCLICK`, `ONMOUSEDOWN`, `ONMOUSEUP`, `ONMOUSEMOVE`, `ONMOUSEOUT`, `ONKEYPRESS`, `ONKEYDOWN`, `ONLOAD`, `ONUNLOAD`, `ONFOCUS` (nonstandard), `ONBLUR` (nonstandard), `ONERROR` (nonstandard), `ONMOVE` (nonstandard), `ONRESIZE` (nonstandard), `ONDRAGDROP` (nonstandard), `BGPROPERTIES` (nonstandard), `CLASS`, `ID`, `STYLE`

The `BODY` tag is often used without attributes, (e.g., simply `<BODY>` as the start tag), but can optionally contain attributes designating the background image, the background color, and the foreground colors of normal text, unvisited hypertext links, visited hypertext links, and links that are being selected. JavaScript-enabled browsers also support attributes that specify code to be executed under various conditions.

BACKGROUND and BGCOLOR The `BACKGROUND` attribute specifies the URL of an image file that is to be tiled across the background of the page. Image repetition (tiling) minimizes download time of a repetitive background pattern. For instance, a page with colors that fade left-to-right across the page would use a `BACKGROUND` image with fading colors that is very wide but only one pixel high. A good idea is to supply a background color (`BGCOLOR`) as insurance when specifying a background image (`BACKGROUND`) in case the reader has image loading disabled. Most browsers use white (`WHITE` or `#FFFFFF`) or light gray (`SILVER` or `#C0C0C0`) as the default `BGCOLOR`. Although widely used, both the `BACKGROUND` and `BGCOLOR` attributes are technically deprecated, with the recommendation that style sheets be used instead.

TEXT, LINK, VLINK, and ALINK These attributes specify the foreground color of normal text, of unvisited hypertext links (based on the browser's current history), of visited links, and of links currently being selected. This last category is a temporary color to show when the user has pressed the mouse down over a link but not yet released the mouse. Colors can be specified either by a symbolic name taken from the original Windows VGA palette or by the hexidecimal equivalent, that is, a `#` sign followed by six hexadecimal digits, two each for red, green, and blue. Table 1.1 lists color names and their hex equivalents. For systems that can only support 256 colors, the Color-Safe palette defines the subset of 216 colors that will not dither on the monitor. An example of the Color-Safe palette is located at http://www.docnprof.com/safecolorA.html. Both Netscape and Internet Explorer also allow the use of the X11 window system color names. An example of each of the X11 colors is provided at http://www.zdnet.com/devhead/resources/tag_library/misc/x11names.html. Again, these attributes are deprecated in favor of style sheets.

Table 1.1. Predefined color names in HTML 4.0

Color Name	*Hex Equivalent*	*Color Name*	*Hex Equivalent*
AQUA	#00FFFF	NAVY	#000080
BLACK	#000000	OLIVE	#808000
BLUE	#0000FF	PURPLE	#800080
FUCHSIA	#FF00FF	RED	#FF0000
GRAY	#808080	SILVER	#C0C0C0
GREEN	#008000	TEAL	#008080
LIME	#00FF00	WHITE	#FFFFFF

MAROON	#800000	YELLOW	#FFFF00

Most browsers use black (BLACK or #000000) as the default TEXT color, blue (BLUE or, in the case of Netscape, a slightly darker #0000EE) as the default LINK color, dark purple/blue (#551A8B in Netscape) as the default VLINK color, and red (RED or #FF0000) as the default color for ALINK.

TITLE TITLE displays a tool-tip when the mouse is paused over the document. TITLE is supported by Internet Explorer 4.0 and above. However, this attribute is not supported by Netscape, at least as of Netscape version 4.7.

Core Warning

Netscape does not support the TITLE attribute.

ONCLICK, ONDBLCLICK, ONMOUSEDOWN, ONMOUSEUP, ONMOUSEMOVE, ONMOUSEOUT, ONKEYPRESS, ONKEYDOWN, ONLOAD, ONUNLOAD, ONFOCUS, ONBLUR, ONERROR, ONMOVE, ONRESIZE, and ONDRAGDROP These attributes specify JavaScript code that should be executed when the page receives a mouse event or keyboard event or when the focus of the page changes. For further details on JavaScript, see Chapter 24 (JavaScript: Adding Dynamic Content to Web Pages).

BGPROPERTIES In Internet Explorer, supplying BGPROPERTIES="FIXED" signifies that the background image specified by the BACKGROUND attribute should not scroll with the rest of the page. This property is sometimes described as a watermark.

Finally, the cascading style sheet specification also adds CLASS, ID, and STYLE attributes to *all* HTML elements except for BASE, BASEFONT (ID allowed), HEAD, HTML, META, PARAM (ID allowed), SCRIPT, STYLE, and TITLE. These style sheet attributes are not discussed separately for each HTML element. See Chapter 5 (Cascading Style Sheets) for details.

1.7 Summary

HTML is the foundation for creating a professional Web site. Whether providing on-line technical support, stock market quotes, international news, or e-commerce shopping, the information viewed by the user is contained in an HTML document.

Today, most browsers support the HTML 4.0 specification. This chapter covered the basic structure of an HTML document and showed you how to create, validate, and post your first Web page on the Internet.

Every HTML 4.0 document should contain a DOCTYPE declaration followed by an element containing a HEAD and a BODY. The HEAD should always contain a TITLE and can also contain STYLE, META, or other high-level information that specifies the author, gives a forwarding location, or otherwise describes the overall document. The META data is often used by search engines to populate databases with information about the document. The BODY tag itself is often empty, but attributes can be used to specify page colors or images or to designate JavaScript actions that take place in various circumstances.

Once you have the outline of the HTML document in place, you will want to actually put something into the document. The following chapters describe how to add more content to your document. Chapter 2 (Block-Level Elements in HTML 4.0) describes the major text blocks that make up your documents, and Chapter 3 (Text-Level Elements in HTML 4.0) covers the smaller elements that can go inside the various text sections. Then Chapter 4 discusses frames, which allow you to divide your Web page into cells and place a different HTML document in each cell. Finally, users who want to customize the way certain browsers display various HTML elements can use cascading style sheets, as described in Chapter 5.

CONTENTS