

How HTML Works

The code within an HTML file is enclosed in *tags*. These tags indicate where the formatting should be applied, how the layout should appear, what pictures should be placed in certain locations, and more.

For example, suppose you wanted a certain word to be italicized, like this:

Everything is on sale.

In HTML, there's no Italics button to click, like there is in a word-processing program.

Therefore, you have to "tag" the word that you want to be italicized. The code to turn on italics is `<i>`, and the code to turn italics off is `</i>`. Your HTML code would look something like this:

```
<i>Everything</i> is on sale.
```

That's an example of a *two-sided tag*, which encloses text between opening and closing tags, in this case `<i>` and `</i>`. With a two-sided tag, there is always a corresponding closing tag for every opening tag.

To understand how this system of tagging came about, you need to know that back in the olden days of the Internet, nearly everyone connected to it by using a dial-up modem, at speeds ranging from 2400 bps to 28.8 Kbps. That's *really slow*. Text files transfer much faster than binary files, so for any type of information-sharing system to be popular, it had to be text-based. Otherwise, people would doze off while waiting for a page to load.

People designing Web pages also wanted their pages to be attractive. They couldn't just format pages in a word processor, though, because every word processor handled formatting differently, and it was impossible to know which one a visitor to a site might be using. Word processing files are also much larger than plain text files.

The Web's founding fathers developed an elegant solution. Instead of sending the formatted pages over the Internet, they created an application—a Web browser—that could interpret plain-text code (HTML tags) as formatting instructions. The text could be sent quickly and efficiently in plain-text format, and then be processed and displayed attractively and graphically on the local PC.

HTML worked great all by itself for all kinds of text formatting, but some Web designers wanted to include graphics on their pages. To accommodate this, the `` tag was created, which designers use to refer to a graphic stored on a server. When the Web browser gets to that tag, it requests that the image file be downloaded from the server and displayed on the page. (You'll learn how to insert images in Chapter 9, "Inserting Graphics.")

The `` tag is different in several ways from the `<i>` tag. It is *one-sided*, meaning it does not have a closing tag, and it takes arguments. An *argument* is text within the tag that contains information about how the tag should behave. For example, for an `` tag, you have to specify a source, abbreviated *src*. Here's an example:

```

```

This `` tag uses the `src=` argument, and specifies that the file *tree.gif* be displayed. Many tags accept arguments, either optional or required. You'll see many examples throughout the exercises in this book.

With HTML, you can also create *hyperlinks* from one page to another. When a visitor to a Web site clicks a hyperlink, the Web browser loads the referenced page or jumps to a marked section (a "bookmark") within the same page. You will learn to create hyperlinks in Chapter 5, "Creating Hyperlinks and Anchors."

The tag for a hyperlink is `<a>`, a two-sided tag, but most people wouldn't recognize it without the argument that specifies the file or location to which to jump. For example, to create a hyperlink with the words *Click Here* that jumps to the file *index.htm* when clicked, the coding would look like this:

```
<a href="index.htm">Click Here</a>
```

There's a lot more to HTML, of course, but that's basically how it works. Plain text is marked up with tags that indicate where elements such as formatting, hyperlinks, and graphics should be placed, and a Web browser interprets those tags and displays the page in its formatted state. The trick, of course, is to know which tags to use, and where, and what arguments they need.