Chapter 6

Creating a Web Page and Entering Text

CONTENTS

With the basics behind you, it's time to start creating your first HTML pages. As has already been mentioned, the basic building block of an HTML page is text. To create these pages, all you really need is a text editor and a Web browser for testing your creation (you'll eventually need a graphics program to create and edit your graphics, too). So let's look at the basic tools for Web publishing, and then create your own HTML template.

The Tools for Web Publishing

I've already mentioned it above-all you need is a text editor. In Windows 95, that's Notepad or WordPad. For Mac users, SimpleText is the perfect HTML editor. UNIX users can opt for VI or Emacs. Basically, all you need to remember is that HTML pages, while they include the `.htm` or `.html` file extensions, are simply ASCII text files. Any program that generates ASCII text files will work fine as an HTML editor-even a word processor like WordPerfect or Microsoft Word.

Tip
If you create an HTML page in a word processor, don't forget to use the Save As command to save it as an ASCII text file.

You'll also need a Web browser to check on the appearance of your Web page as you create it. All Web browsers should have the ability to load local pages from your hard drive, just as they can load HTML pages across the Web. Check the menu of your Web browser (if it's a graphical browser) for a command like File, Open (see fig. 6.1).

You may have heard of some dedicated HTML editing programs that are designed to make your work in HTML easier. They do indeed exist, and they can be very useful. Unfortunately, many of them also hide the HTML codes from the designer, so they would be difficult for us to use as you learn how HTML works. Once you understand HTML, though, it can be a great benefit to use one of these browsers. I'll talk about some of them in Chapters 27, 28, and 29.

Document Tags

The first HTML tags you're going to look at are the document tags. These are the tags that are required for every HTML page you create. They define the different parts of the document.

Just like a magazine article, an HTML document has two distinct parts-a head and a body. The head of the HTML document is where you enter the title of the page. It's also used for some more advanced commands that you'll study later in Chapters, 10, 19, 22 and 23.

To create the head portion of your HTML document and to give the document a title, type the following in your text editor:

```
<HEAD>
<TITLE>My First Page</TITLE>
</HEAD>
```

This tells a Web browser what information should be considered to be in the head portion of of the document, and what it should call the document in the title bar of the browser window.

If you've got a head, then you'll need a body, right? The body is where you'll do most of your work-you'll enter text, headlines, graphics, and all your other Web goodies. To add the body section, start after the `</HEAD>` tag, and enter the following:

```
<BODY>

</BODY>
```

Between these two tags, you'll eventually enter the rest of the text and graphics for your Web page.

There's one last thing you need to consider. In order that all Web browsers understand that this is an HTML document (remember that you're saving it as ASCII text, so the browser could be confused), you need to add some tags on either side of the head and body tags you've created. Above the first `<HEAD>` tag, enter the following:

```
<HTML>
```

After the last `</BODY>` tag, type the following:

```
</HTML>
```

Now, at least as far as your Web browser is concerned, you have a complete Web document!

Example: Creating an HTML Template

Let's take what you know and create a template. By saving this template as a generic text file, you'll have a quick way to create new HTML files-simply load the template and use the File, Save As command to save it as your new Web page.

Start by entering the following in a blank text file:

```
<HTML>
<HEAD>
<TITLE>Enter Title Here</TITLE>
</HEAD>
<BODY>

</BODY>
</HTML>
```

And that's it. Now save this as an ASCII text file called `template.html` (or `template.htm` if you're using DOS or Windows 3.1). Now, whenever you're ready to create a new HTML document, simply load `template.html` into your text editor and use the Save As command to rename it.

Note
If you use a word processor to create your HTML files, you may notice that sometimes you get more than one option for saving files as ASCII text. So which one is right? Fortunately, it doesn't really matter. The big problem comes in editing the text on different platforms since DOS-based machines (including Windows) and Macs treat returns and linefeeds differently. If you plan to edit a Mac-created HTML file on a DOS machine, for instance, choose DOS text when you save it. Funny little newline characters will now appear in a Mac text editor, but everything will look good on the DOS side.

Example: Hello World

When learning a new programming language, it's traditional that the first program you create is designed to say "Hello World." Well, HTML isn't a programming language-but I can use the Hello World example to prove that your template is a complete Web document.

Load the `template.html` file into your text editor, and use the Save As command to rename it `hello_world.html` or something similar. Now, edit the document so that it looks like this:

```
<HTML>
<HEAD>
<TITLE>Hello World Page</TITLE>
</HEAD>
<BODY>
Hello World!
</BODY>
</HTML>
```

Select the File, Save command from your text editor. Now load your Web browser and select the Open File (or similar) command from the File menu. In the dialog box, find the document `hello_world.html` and select OK to load it into your Web browser. If everything goes as planned, your browser should display something similar to figure 6.2.

And that's a Web page!

Understanding Tags: Container and Empty Tags

In creating your HTML template, you've already dealt with some of the most basic tags in HTML. The first thing you should notice about these HTML tags is that all tags include < and > on either side of the tag's command. This is how HTML recognizes tags. If you don't use the brackets, then a Web browser will assume your commands are text that you want displayed-even if that text is the same as an HTML command.

While a Web browser would consider the following to be a tag:

```
<HTML>
```

that same Web browser would interpret the following as text to be displayed on-screen:

```
HTML
```

Tip
Tags are not case-sensitive, so they don't have to be all uppercase-even though that's how they appear in this book. I suggest you type them as uppercase, though, since it makes them stand out in your text editor.

Because tags aren't considered text by the document, they also don't show up in the document. If the browser interprets something as a tag, it won't appear in the browser window.

Container Tags

You may have noticed that for every tag, such as the title tag, you actually entered two different HTML commands-an "on" tag and an "off" tag. The off tag is the same as the on tag, except for the `/` after the <.

In HTML, tags that include both an on and an off tag are called *container tags*. These tags wrap around text in your document and perform some sort of formatting on the text. They hold, or contain, the text between the two tags. The title, HTML, head, and body tags are all container tags-the relevant text goes between the on and off tags.

Container tags always have the following form:

```
<TAG>text being formatted or defined</TAG>
```

In fact, you've already been introduced to a fairly common container tag in the first chapter of this book, the `<EM>` (emphasis tag). An example of the emphasis tag would be:

```
Here's some <EM>really important</EM> text.
```

Because `<EM>` is an implicit formatting tag, it's up to the browser to decide what to do to the text between the on and off tags. But only the words `really important` will be affected in this example, since they're the only text that is being "contained" by the tags.

Empty Tags

All other tags in HTML fall into one other category, called *empty tags*. These tags have only an on tag-there are no off tags. The reason for this is that empty tags don't act on blocks of text. Instead, they do something all on their own. An example of this would be the <HR> (horizontal rule) tag. This tag draws a line across the width of your document. For example:

```
The following is a horizontal line:

<HR>
The rest of this is just more text.
```

When viewed in a Web browser, the two sentences will be separated by a horizontal line, as in figure 6.3.

Entering Paragraph Text on Your Web Page

With your template prepared, and with an understanding of the two types of tags in HTML, you're ready to enter text on a Web page. As mentioned earlier, all the text that you enter on a page should come between the <BODY> and </BODY> tags. Like <EM>, the body tags are container tags that tell a Web browser what parts of the HTML document should be displayed in the browser window.

You've seen that you can just type text into an HTML document and it will be displayed in the browser. Technically, though, most of the text you type should be in another container tag: the <P> (paragraph) tag. This tag is used to show a Web browser what text in your document constitutes a paragraph. For the most part, Web browsers ignore more than one space between words and will ignore returns that you add to your HTML file while you're creating it.

In order to give the appearance of paragraphs, then, you have to use the paragraph container tag. The paragraph tag uses the following format:

```
<P>Here is the text for my paragraph. It doesn't matter how long it is, how many
spaces are between the words or when I decide to hit the return key. It will create a
new paragraph only when I end the tag and begin with another one.
</P>

<P> Here's the next paragraph. </P>
```

Note
Although it is technically a container tag, the </P> tag is not required at the ends of paragraphs by HTML 2.0. This tends to cause a little confusion. Many people end up using <P> as an empty tag, assuming that it's designed to insert a line break at the end of paragraphs (or even to create multiple blank lines). That's not its purpose. Using <P> as a container, as I've shown previously, gets the most reliable results in all different types of browsers. In the spirit of good HTML, the container is used to isolate all the text you want to call a "paragraph." Then it lets the browser render that in the way its programmers feel is most appropriate.

Like the emphasis tag, the paragraph container tells the Web browser that all of the text between the on and off tags is in a single paragraph. When you start another paragraph, the Web browser will drop down a line between the two.

Here's that same example, except you'll throw in some spaces. Remember, spaces and returns almost never affect the way the text will be displayed on the screen. In a paragraph container, the browser will ignore more than one space and any returns.

```
<P>Here is the text for my paragraph.
It doesn't matter how long it is, how many spaces are between the words
or when I decide to hit the return key. It will create a new paragraph
only when I end the tag and begin with another one. </P>

<P> Here's the next paragraph. </P>
```

Both this example and the previous example will be displayed in the Web browser in exactly the same way.

The
 Tag for Line Breaks

But what if you want to decide where a line is going to end Consider the example of entering an address in a Web document, as follows:

```
<P>
Richard Smith
14234 Main Street
Anycity, ST 00001
</P>
```

It looks about right when you type it into your text editor. However, when it displays in a Web browser, it looks like figure 6.4.

We already know what the problem is: Web browsers ignore extra spaces and returns! But if you put each of those lines in a paragraph container, you'd end up with a space between each line-and that would look wrong, too.

The answer is the empty tag
, which forces a line return in your Web document. Properly formatted, your address would look like this:

```
<P>
Richard Smith<BR>
14234 Main Street<BR>
Anycity, ST 00001<BR>
</P>
```

And it would look just right in your Web browser, just as in figure 6.5.

The Comment Tag

There's one other tag I'd like to discuss in this chapter, called the comment tag. This tag is fairly unique, in that it's actually used to make the Web browser ignore anything the tag contains. That can be text, hypertext links, image links, even small scripts and programs.

Tip
It's best to delete obsolete links and tags from your documents, rather than just using the comment tag. Some browsers will display certain tags even if they are "commented out."

For now, you'll use the comment tag to hide text. The point in hiding the text is that it allows you to create a private message that is intended to remind you of something or to help those who view the raw HTML document to understand what you're doing. That's why it's called the comment tag. For instance:

```
<!--This is a comment that won't display in a browser-->
```

The comment tag isn't the most elegant in HTML, but it usually works. Anything you type between `<!--` and `-->` should be ignored by the browser. Even multiple lines are ignored-as with most tags, the comment tag ignores returns.

Generally, you'll use the comment tag for your own benefit-perhaps to mark a point in a particular HTML document where you need to remember to update some text, or perhaps to explain a particularly confusing part of your page. Since it's fairly easy for anyone to view your raw HTML document, you might also use the comment tag to create a copyright message or give information about yourself (see the sidebar).

Viewing the Source of Web Pages

Ever been out on the Web looking at a particularly well-designed HTML document-and wondering how they did it?
If you'd like to, most browsers will let you view the document source for any Web page they can load. This allows you to download the raw HTML codes and ASCII text, just as if you'd created the page yourself.
To do this, select the View Document command in the Edit menu of your Web browser (the command may differ slightly, so look for a similar name if you can't find View Document). What results is the plain ASCII text file that was used to create that Web page.
Depending on your browser, this source file will either be displayed in the browser window, or saved to your hard drive and displayed in the default text editor. If the source is displayed in the browser window, then select File, Save As to save the source to your hard drive.
Now you might be able to imagine how comments can come in handy. If you would rather not have people copy and use the source from your Web pages (or if your pages contain otherwise copyrighted material that you want to protect), you can use the comment tag to let others know that you consider the page your property. For instance:

```
<!--Contents of this document Copyright 1996 Todd Stauffer. Please do not copy or otherwise reproduce the source HTML code of this document without permission.-->
```

Of course, that's not to say that you shouldn't also offer a visible copyright notice or other legal disclaimers. But comments within the code tend to talk directly to folks a little more HTML-savvy. Using a comment tag like this is a great way to encourage other Web designers to ask you before using your HTML pages for their own private use. (But if they don't ask, any legal problems are your own I'm afraid.)

Note

Don't let this confuse you, but the comment tag is an unusual one. It's not really a container tag, since it doesn't have two similar tags that are differentiated only by `/` in the second tag. At the same time, it's difficult to describe as an empty tag, since it does do something to text in the document.

Example: Creating a Complete Web Page

Let's take everything you've learned and build a complete Web page. Start by loading the template and using Save As to create a new document for this example. (Call it `test1.html` or something similar.)

Now, create a document that looks something like Listing 6.1. You should have to change only the title text; enter the other text between the body tags.

Listing 6.1 `test1.html` Testing Tags

```
<HTML>
<HEAD>
<TITLE>The Testing Tags Page</TITLE>
<!--This page is Copyright 1996 Todd Stauffer-->
</HEAD>
<BODY>
<P>On this page we're reviewing the different types of tags that we've learned in this
chapter. For instance, this is the first paragraph.</P>
<P>In the second paragraph, I'm going to include the name and address of one of my
favorite people. Hopefully it's formatted correctly.<BR>
Tom Smith<BR>
1010 Lovers Lane<BR>
Anywhere, US 10001<BR>
</P>
<HR>
<P>Now I'll start a <EM>completely new</EM> idea, since it's coming after a horizontal
line.</P>
<!--Don't forget to update this page with the completely new idea here.-->
</BODY>
</HTML>
```

When you've finished entering the text and tags (you can use your own text if you like; just try to use all of the tags we've reviewed in the chapter), use the Save command in your text editor. Now switch to your Web browser, and load your new page using the Open File (or similar) command.

If everything went well, it should look something like figure 6.6.

Review Questions

1. Is it necessary to use a special program to create HTML pages?
2. In what file format are HTML pages saved? What file extension should be used for an HTML document?
3. What are the three basic document tags?
4. What tag have you learned is appropriate for the head area of an HTML document?
5. What's the first thing you should do after loading an HTML template you've created into a text editor program?
6. What is the main difference between container and empty tags?
7. Give one example of an empty tag.
8. Why is the comment tag different from most other container tags?
9. True or false. All text for your Web page should be typed between the body container tags.
10. Aside from line spacing, what is the main difference between the `<BR>` and `<P>` tags?
11. Use your Web browser to view and save the main source code for the following Web document: **http://www.ibm.com/index.html**. (You may also need to use a text editor, depending on your Web browser's capabilities.)

Review Exercises

1. Create a document that uses nothing but <P> container tags to break up text. Then, create a document that uses nothing but
 tags. What's the difference in your browser?
2. Try adding additional <P> or
 tags to your documents between lines or text or paragraphs. Do they add extra lines in your browser? View them from more than one browser. (Hint: adding lines between paragraphs for multiple
 or <P> tags is not supported by the HTML standard, although some popular browsers recognize them.)
3. Add a standard "header comment" to your template using the comment tag. This is a great idea, especially if you develop HTML pages for your company-after all, documenting your efforts is what the comment tag is all about. Here's an example for a template, which can be altered every time you create a new document:

```
<!--
Page Designer: Todd A. Stauffer
Creation Date: 00 Month 9?
Revision Date: 00 Month 9?
File type: HTML 2.0     -->
```

Chapter 7

Changing and Customizing HTML Text

CONTENTS

HTML 2.0 is a standard created after the fact. What I mean is that HTML was already in wide use when the standard was finally written. As a result, there tend to be a few different ways to do the same things. You'll take a look at most of them, and I'll try to explain the theory behind each. I'll also recommend one or two options that best do what you're interested in accomplishing-and just leave the rest of the options for you to consult if the occasion ever demands.

Creating Headers and Headlines

One of the first things you might have wondered when you were entering text in Chapter 6 is, "How can I change the size of the text?" HTML 2.0 doesn't have any explicit tags or commands for changing the font size within a document (although Netscape HTML does). Instead, it relies on the implicit header tags to do this.

Header tags are containers, and unlike many other HTML tags, they double as paragraph tags. Ranging from level 1 to level 6, headers allow you to create different levels of emphasized headlines to help you organize your documents. The following is an example; see figure 7.1 for the results:

```
<H1>Header Level One is the largest for headlines or page titles</H1>
<H2>Level Two is a little smaller for major subheads</H2>
<H3>Level Three is again smaller, for minor subheads</H3>
<P>This is regular text.</P>
```

```
<H4>Level Four is about the same size as regular text, but emphasized</H4>
<H5>Level Five: again emphasized, but smaller than regular text</H5>
<H6>Level Six is generally the smallest header</H6>
```

You cannot include a header tag on the same line as regular text, even if you close the header tag and continue with unaltered text. A header tag has the same effect as a <P>, in that it creates a new line after its "off" tag. The following:

```
<H1>This is a header</H1> And this is plain text.
```

offers the same results as:

```
<H2>This is also a header</H2>
<P>And this is also plain text</P>
```

In both cases, the Web browser will place the header text and plain text on different lines, with the header text appearing larger and the plain text appearing "normal" in size.

Note
The HTML standard technically requires that using a particular header level requires that the larger header tags be used previously. So, for instance, if you use an <H2> tag, you should have an <H1> tag somewhere before it. Very few browsers (if any) actually require this and, for the most part, HTML designers use header tags as simply a way to change the size of text for emphasis. That's how I use them, even going so far as to use <H5> or <H6> for "fine print" on my pages. If you're an absolute stickler for standards, though, realize that it's more correct to only use header tags for true headers in your documents, and then only in order (i.e., <H1>, <H2>, <H3>, and so on).

Example: A Topical Discussion

Now, with the addition of the header tags, you're suddenly able to add a level of organization to your pages that was lacking previously. Using the horizontal line and emphasis tags you saw in Chapter 6, it's possible to create a very useful text-oriented HTML document with what you now know.

Let's start just with headers and regular text. Load your HTML template into a text editor and save it as a new HTML document (`headers.html` or something similar). Then fill in the template's body section using both header containers and paragraph containers (see Listing 7.1).

Listing 7.1 `headers.html` The Template's HTML Body Section

```
<BODY>
<H1>Welcome to my home on the Web</H1>
<P>Hi there! My name is Mark Williamson, and I'm an active participant
in the Web. Aside from my Internet journeys I'm also a big fan of the
science-fiction writer Wilhelm Norris, and I love collecting models of
television spacecraft. As far as the boring stuff goes, I work as a Macintosh
programmer in Carmel, California.</P>
<H2>My Work</H2>
<P>I've recently moved from programming in a Microsoft Windows
environment to a Macintosh environment, and I must admit that I've been more than a
little overwhelmed. Fortunately I've had good help from local user groups and my co-
workers...plus, they've introduced me to some exceptional tools for Mac
```

```
programming.</P>
<H3>ProGraph</H3>
<P>If you've never worked in a visual programming environment, you're
in for a treat. With my background in Windows and UNIX C programming, I was surprised
how quickly I picked up this object-oriented concept. I definitely recommend it!</P>
<H3>MetroWerks</H3>
<P>I can't imagine I even need to say anything about this. It's hands-down the best C
and C++ development environment ever created for Macintosh. In my opinion, it's the
best created for any platform!</P>
<H5>This document contains opinions that are my own and do not
necessarily reflect those of my employer.</H5>
</BODY>
```

Entering text and using header tags in this way allows us to create a document that has more of the feel of a well-outlined magazine article, or even a chapter in a book. You may have noticed that this book uses different-sized headlines to suggest that you're digging deeper into a subject (smaller headlines) or beginning a new subject (bigger headlines). HTML allows you to do the same thing with the header tag (see fig. 7.2).

Implicit and Explicit Text Emphasis

Implicit tags are those that allow the browser to choose, within limitations, how the marked-up text will be displayed. Header tags are actually an example of an implicit tag, since the HTML designer has no control over how much bigger or smaller a header tag will be. Although most browsers will render header tags in somewhat similar ways, others (for instance, nongraphical browsers) have to come up with another system for emphasis, such as underlining or highlighting the text.

Because HTML was originally created with the overriding mission of being displayed on nearly any computer system, implicit tags for emphasis were a necessity. HTML allows the designer to decide what text will be emphasized. But only explicit tags tell the Web browser how to render that text.

Explicit Styles

Explicit tags are also often called *physical tags*, since they very specifically tell the Web browser how you want the text to physically appear. The browser is given no choice in the matter.

The basic explicit tags are containers that let the user mark text as bold, italic, or underlined (see Table 7.1).

Table 7.1 HTML Physical Container Tags

Tags	*Meaning*
<B>, </B>	Bold text
<I>, </I>	Italic text
<U>, </U>	Underlined text

Note:
Not all browsers will render underlined text (notable among them is Netscape Navigator), because hypertext links are also often displayed as underlined, which could potentially be confusing.

With these tags, the browser really has no choice-it must either display the text as defined or, if it can't do that, then it must add no emphasis to the text. This is both good and bad for you as the designer. If you prefer that text not be emphasized at all if it can't be italic, for example, then you should use the <I> tag.

Another feature of explicit (physical) tags is that they can generally be used in combination with other tags. As you'll see in the next section, this isn't always a good idea with implicit tags. For instance, most graphic browsers will render the following example by applying both tags to the text (see fig. 7.3).

```
<H1><I>Welcome Home!</I></H1>
<B><I>This is bold and italic</I></B>
```

Implicit HTML Tags

Implicit styles are often called *logical styles*, since they allow the browser some freedom in how it will display the text. These tags, like the header tags, are generally relative to one another, depending on the browser being used to view them. See Table 7.2 for some of the common implicit (logical) tags

Table 7.2 Some Basic Logical HTML Tags

Tags	*Meaning*	*Generally Rendered as…*
<EM>, </EM>	Emphasis	Italic text
<STRONG>, </STRONG>	Strong emphasis	Bold text
<TT>, </TT>	Teletype	Monospaced text

Table 7.2 includes a section that tells you how these tags are often rendered in graphical Web browsers. There's no rule for this, though, and the tags don't necessarily have to be rendered in that way.

There are two other distinctions between these tags and the physical tags (such as bold and italic) that you've already discussed. First, these logical tags will always be rendered by any Web browser that views them. Even text browsers (which are unable to show italic text) will display the <EM> or <STRONG> tags by underlining, boldfacing, or highlighting the text.

Second, these tags are generally not effective when used together. Where <B><I>*text*</I></B> will sometimes offer useful results, <EM><STRONG>*text*</STRONG></EM> rarely will. Combining these tags with other tags (such as header tags or physical tags) is often either ineffective or redundant.

Note
My warning about combining logical tags isn't always applicable, even though it's a good rule to follow. Netscape Navigator, for instance, will render both <EM> and <STRONG> tags simultaneously with others. (Used together, the tags would result in bold, italicized text in Navigator.)

Example: Physical versus Logical

Here's a great way to kill two birds with one stone. With this example you can get a feel for using both the physical and the logical tags discussed above. At the same time, you can also test these tags in your browser to see how they're displayed. (If you have more than one browser, test this example in all of them. That way you can see how different browsers interpret logical tags.)

To begin, load your template file in a text editor, and rename it something intuitive, like `tagtest1.html`. Then, enter the text between the body tags as it appears in Listing 7.2.

Listing 7.2 `tagtest1.html` HTML Body Tags Text

```
<BODY>
<P>
This is a test of the <B>bold tag</B><BR>
This is a test of the <STRONG>strong emphasis tag</STRONG><BR>
</P>
<P>
This is a test of the <I>italics tag</I><BR>
This is a test of the <EM>emphasis tag</EM><BR>
</P>
<P>
This is a test of the <B><I>bold and italics tags together</I></B><BR>
This is a test of the <STRONG><EM>strong and emphasis tags together</EM>
</STRONG><BR>
</P>
<P>
While we're at it, does <U>underlined text</U> appear in this browser?<BR>
And what does <TT>teletype text</TT> look like?<BR>
</P>
</BODY>
```

Note
Remember that using `<STRONG>` and `<EM>` together is not recommended in the HTML 2.0 standard. We did it just as an example to see how it renders in your browser.

When you've finished entering this text, save the file again in your text editor, then choose the Load File command in your Web browser to display the HTML document. If you have other Web browsers, see how those respond to the tags, too (see fig. 7.4).

Other Implicits: Programming, Quoting, and Citing

At the beginning of this chapter, I mentioned that the proliferation of HTML tags took place before the standard was ever conceived of-which might explain some of the tags that we discuss in this section. For the most part, these tags are implicit (logical) and aimed directly at certain areas of expertise. At the same time, however, the bulk of these tags will look exactly the same in a Web browser.

Programmer's HTML Tags

One of the early, more common uses for HTML was for documenting computer programs and offering tips or advice to computer programmers. Part of the HTML 2.0 standard, then, offers some implicit (logical) HTML tags that allow HTML designers to mark text in a way that makes it easier to present computer programming codes. Those tags are in Table 7.3.

Table 7.3 HTML Tags for Computer Programming

Tags	*Meaning*	*Generally Rendered as…*
`<CODE>, </CODE>`	Programming lines	Monospaced (like `<TT>`)
`<KBD>, </KBD>`	Keyboard text	Monospaced

<SAMP>, </SAMP>	Sample output	Monospaced
<VAR>, </VAR>	Variable	Italic

Notice that the majority of these tags are often displayed in exactly the same way-in the default monospaced font for the browser. Then why use them?

First, not all browsers will necessarily follow the "general" way. Some browsers will actually render these tags in slightly different ways from one another, so that <SAMP>, for instance, might appear in a slightly larger font than <CODE>.

Note
These tags had more meaning with earlier browsers like Mosaic, which used to allow users to define their own fonts and sizes for specific tags. In an era where browsers give the designer control over actual font families and sizes (see Chapters 19 and 21), these tags are used less and less.

Second, using these tags is a great way to internally document your HTML pages, so that you can tell at a glance what certain text is supposed to be. This will help you later when you return to the document to update it or fix errors-especially as the document becomes more complex.

Quoting, Citing, Definitions, and Addresses

Along the same lines as the HTML "programmer's" tags, you have available certain implicit tags that work as typographer's or publisher's codes. As shown in Table 7.4, these codes often work in ways similar to others you've already seen-with a few twists.

Table 7.4 HTML Publisher-Style Tags

Tags	*Meaning*	*Generally Rendered as…*
<CITE>, </CITE>	Bibliographical citation	Italic text
<BLOCKQUOTE>, </BLOCKQUOTE>	Block of quoted text	Indented text
<DFN>, </DFN>	Term definition	Regular text
<ADDRESS>, </ADDRESS>	Street or e-mail address	Italic text

Again, notice that the <CITE> tag isn't going to be rendered any differently from the italics, emphasis, or variable tags you've seen previously. The <DFN> tag is often not rendered as any special sort of text at all, whereas the <ADDRESS> tag is identical in function to the italics tag.

So the best use for these tags (with the exception of the <BLOCKQUOTE> tag) is as internal documentation of your HTML documents. Remember, of course, that some browsers may render them slightly differently from what is suggested in Table 7.4.

Example: Using the <BLOCKQUOTE> and <ADDRESS> Tags

The only really new tag in the Table 7.4 is the <BLOCKQUOTE> tag. This tag usually indents the left margin of regular text in the browser window, just as you might find a blocked quotation formatted in a printed document.

Also as part of the tag, <BLOCKQUOTE> generally adds a return or one extra line on either side of the tag, so no paragraph tags are needed. Paragraph tags should, however, be used to contain text on either side of the blockquote.

Although the <ADDRESS> tag is similar to italics or emphasis, I've thrown in an example of using it correctly. Remember to include a line break after each line of the address.

To begin this example, create and save a new HTML document from the template you created in Chapter 6. Enter Listing 7.3 between the body tags.

Listing 7.3 `emphasis.html` The <BLOCKQUOTE> and <ADDRESS> Tags

```
<BODY>
<P>I believe it was Abraham Lincoln who once said (emphasis is mine):
<BLOCKQUOTE>Four score and seven years ago our <B>forefathers</B> brought
forth on this continent a new nation, conceived in <I>liberty</I> and
dedicated to the proposition that all men are created <EM>equal</EM>.
</BLOCKQUOTE>
It was something like that, wasn't it?
</P>
<P>If you liked this quote, feel free to write me at:<BR>
<ADDRESS>
Rich Memory<BR>
4242 Sumtin Street<BR>
Big City, ST 12435<BR>
</ADDRESS>
</P>
</BODY>
```

Notice that an off paragraph tag isn't required before you get into the address tag-remember, <ADDRESS> works very much as italics does, and the
 tag is designed to work as well inside a paragraph container as it does outside one. So you can put the paragraph tag after the address, to contain both address listing and the text in the same paragraph.

What does all of this look like? Take a look at figure 7.5. <BLOCKQUOTE>, unlike some of the tags you've looked at, really does offer unique abilities that make it worth using in your documents.

Preformatted Text

Are you ready to break some of the rules of HTML that I've been harping on over the last two chapters? That's what you're about to do-in fact, you're going to break two. I've said over and over that the HTML 2.0 standard is not designed for layout. In fact, you haven't even learned how to put two blank lines between paragraphs.

I've also said that spaces and returns in between tags (like the paragraph tag) don't matter. Well, there is at least one exception to this rule: the <PRE> tag.

The <PRE> (preformatted text) tag is designed to allow you to keep the exact spacing and returns that you've put between the on and off tags. The basic reasoning behind this tag is the notion that every once in a while you'd like your text to stay exactly as you put it-for instance, in a mathematical formula, or if you create a table. While there are other ways to do both tables and math, they don't fall under the HTML 2.0 standard. On top of that, you can use <PRE> for a number of other reasons: lists, lining up decimals for dollar figures, and even poetry.

Consider the following example:

```
<P>Oh beautiful, for spacious skies,
For amber waves of grain.
For purple mountains' majesty,
Above the fruited plains.</P>
```

Sure it's a familiar refrain, but it won't look so familiar in a browser if you leave it between paragraph tags. Instead, you can use the <PRE> tag to keep things exactly the way you want them:

```
<PRE>Oh beautiful, for spacious skies,
     For amber waves of grain.
For purple mountains' majesty,
     Above the fruited plains.</PRE>
```

In a browser, it'll look exactly the way you want it to (see fig. 7.6).

You may have noticed that the preformatted text is in a monospaced font-it will always be that way. Otherwise, the <PRE> tag works pretty much like the paragraph font, except that it lets you decide where the line breaks and spaces will appear. Look at the following example:

```
<PRE>I    simply want to make this <B>really</B> clear to you.

</PRE>
```

With the above code, the browser will display this line in nearly exactly the same way as it would using the <P> tag, except that it will be in a monospaced font, and the extra spaces and extra return will appear as well. In fact, there will be two blank lines below the line of text-one for the return, and one for the </PRE> tag itself.

You can even use the <PRE> tags to create extra lines in a document without typing any text between them. This example adds two blank lines to a document:

```
<PRE>
</PRE>
```

For each additional blank line you want to add, just press Enter after the first tag one time.

Note
There is one potential drawback to the <PRE> tag. It doesn't allow the browser screen to wrap text automatically-instead, users need to expand their browser window if you use particular long lines within a <PRE> container. Just keep this in mind, and make sure your lines of text are reasonably short so that all browsers can view them without scrolling.

Example: Creating Your Own Layout with the <PRE> Tag

Let's take a look at a couple of different reasons why you might want to use the <PRE> tag in your HTML documents. Start by loading your template and choosing the Save As command in your text editor to save the file as pre_test.html, or something similar.

Now between the body tags, let's create an example that uses some of the benefits of preformatting-the ability to center text and choose your own margins, for example. How? Let's format some screenplay dialogue (see Listing 7.4).

Tip
Text between <PRE> tags is easier to align if you hit Enter after the on tag, then start typing. Doing so will add an extra line, though.

Listing 7.4 pre_test.html Create Your Own Layout

```
<BODY>
<P>
<TT>
<B>(Int) Rick's Apartment, Late Afternoon</B><BR>
Rick is busying himself with his personal computer when Linda walks through the door
from the kitchen. Startled, Rick bolts upright from his chair and swats frantically at
the keyboard trying to make something disappear. Linda moves closer to the
computer.</TT></P>
<PRE>

                    Linda
                     (confused)
               What were you doing?

                    Rick
               Just the finances.

                    Linda
               But you already printed checks
               last Sunday.

                    Rick
               I know. But Tuesday is when I, uh,
               enter my gambling debts. (Sighs deeply.)
               Honey, I'm in big trouble.
</PRE>
</BODY>
```

It takes a little tapping on the space bar, but with the <PRE> tag you can create some fairly elaborate layouts for getting your point across-especially when layout is just as important as the text itself. In a browser, it comes out looking like a big-budget picture script (see fig. 7.7).

Example: Using <PRE> for Spaces and Tables

In the same way that you created the film script using the <PRE> tag, you can also format a primitive table using the <PRE> tag along with some others. The key to making this work correctly is alignment. Realize

that each space taken up by a character of an invisible tag (like <B>) will not appear in the browser's display, so you'll need to compensate.

Tip
One way to keep the columns in a table straight is to type your table first, and then add emphasis tags afterward.

Load your template and save it as `pre_tbl.html`. Now enter Listing 7.5 between the body tags.

Listing 7.5 `pre_tbl.html` Creating Spaces and Tables

```
<BODY>
<PRE>

</PRE>
<HR>
<H2>Price Per Item in Bulk Orders</H2>
<PRE>


Quantity        XJS100      RJS200       YJS50     MST3000

1-50             $40         $50         $75        $100
50-99            $35         $45         $70        $95
100-200          $30         $40         $65        $90
200+             $25         $35         $55        $75

</PRE>
<H5>Prices do not include applicable sales taxes.</H5>
</BODY>
```

You may need to play with the spacing a bit to line everything up. Save the HTML document, then choose the Open File command in your browser to proof it. Keep playing with it until it looks right.

Tip
If you use a more advanced text editor or word processor, fight your urge to use the Tab key to align <PRE> elements. Use the spacebar instead.

Once you have everything aligned correctly, it's actually a fairly attractive and orderly little table (see fig. 7.8).

Note
You may be tempted to use <B> or another emphasis tag for the column heads in your table. Realize, however, that it is nearly impossible to align columns so that they will appear correctly in every browser when one row is bold and other rows are plain text. Different browsers make bold text a fraction wider than regular text, making the row increasingly misaligned. Even if it looks good in your browser, chances are it won't work in all of them.

Review Questions

1. What are the other names for explicit and implicit tags?
2. What is the difference between an explicit and an implicit tag?
3. Why is the <B> (bold) tag considered explicit?
4. Will the <I> tag work in a text-based browser like Lynx? How about the <EM> tag?
5. What programmer's HTML tag is usually displayed differently from the others?
6. Why would you use a programmer's HTML tag?
7. Is it possible to have more than one paragraph of text in a single <BLOCKQUOTE> container?
8. What other common HTML tag is similar to the <PRE> tag?
9. Can you use other tags, such as <B> or <I>, within <PRE> containers?

Review Exercises

1. Create a document that uses all of the different implicit and explicit layout tags discussed, and note how your browser(s) render them. Also note what happens when you combine tags and view them in your browser(s).
2. What creates spaces in your browser? Create a document that uses multiple
 and <P> tags, and returns between <PRE> tags to add blank lines to your document. Then test the page in your browser to see which are most reliable. (In most cases, it should be <PRE>, but it's interesting to note the differences from browser to browser.)
3. Create a document using the <PRE> tag to work as an invoice or bill of sale, complete with aligned dollar values and a total. Remember not to use the Tab key and avoid using emphasis tags like <B> or <EM> within your list.

Chapter 8

Displaying Text in Lists

CONTENTS

You've probably all heard that one of the best ways to communicate a great deal of information in a short amount of time is by using bulleted lists to convey the message. That philosophy was not lost on the early creators and designers of Web pages, and various tags allow for easy formatting of a number of styles of lists, including both bulleted and nonbulleted incarnations.

Using Lists in HTML

List tags, like paragraphs and preformatted text, are generally HTML containers that are capable of accepting other container and empty tags within their boundaries. These list tags are responsible for affecting the spacing and layout of text, not the emphasis, so they are applied to groups of text, and allow individual formatting tags within them.

Most HTML lists are created following the form:

```
<LIST TYPE>
<ITEM> First item in list
<ITEM> Second item in list
<ITEM> Third item
</LIST TYPE>
```

Each of the items appears on its own line, and the `<ITEM>` tag itself is generally responsible for inserting either a bullet point or the appropriate number, depending on the type of list that's been defined. It's also

possible that the <ITEM> tag could insert no special characters (bullets or otherwise), as is the case with definition listings.

You'll look at each type in the following sections. The basics to remember are to use the main container tags for list type and the individual empty tags to announce each new list item. The type of list you choose is basically a question of aesthetics.

Ordered and Unordered Lists

It might be better to think of these as *numbered* (ordered) and *bulleted* (unordered) lists, especially when we're talking about their use in HTML. The only drawback to that is the fact that the HTML codes for each suggest the ordered/unordered names. For numbered/ordered lists, the tag is <OL>, and for bulleted/unordered lists, the tag is <UL>. Confused yet? That's my job.

For either of these lists, a line item is designated with the empty tag <LI>. In the case of ordered lists, the <LI> tag inserts a number; for unordered lists, it inserts a bullet point. Examples of both follow. The following is an ordered list:

```
<OL>
<LI> Item number one.
<LI> Item number two.
<LI> Item number three.
</OL>
```

And here's an unordered list:

```
<UL>
<LI> First item.
<LI> Second item.
<LI> Third Item.
</UL>
```

Once you've got one of these under your belt, the other looks pretty familiar, doesn't it? To see how these look in a browser, check figure 8.1. (Note that I've added a line of text before each to make each list easier to identify.)

As I've already mentioned, both ordered and unordered lists can take different types of internal HTML tags. It's even possible to include paragraph, line break, and header tags in lists.

Note
In the HTML 2.0 standard, it's considered bad form to use the header tags in bulleted lists, since your goal is probably only to change the size of the text for emphasis. Header tags are designed for page organization, not emphasis. Most browsers will interpret them correctly, but you should also stop to consider that they usually look pretty ugly in lists.

While you may see the potential in creating ordered lists that conform to standard outlining conventions (for instance, Roman numerals and letters), HTML 2.0 doesn't really help much. There is no way to change the <LI> number from Arabic numbers, and there's no way in HTML 2.0 to create a list that starts with something other than 1.

Netscape, however, has added both of these abilities, and you can be much freer in your outline, as long as you warn your users ahead of time to view your page with Netscape Navigator (or a Netscape-compatible browser). Refer to Chapter 19 for more on this.

Example: Formatting Within Lists

Different formatting within lists can offer some dramatically different results, and you should take some time to experiment. Load and save your template as a new HTML document, and enter Listing 8.1 (or similar experiments) within the body tags.

Listing 8.1 `lists.html` Formatting Example

```
<BODY>
<P>The following are some of the things that little boys are made of:</P>
<UL>
<LI> Dirt
<LI> Snails
<LI> Puppy-dog <B>tails</B>
<LI> Worms
<LI> Various ramblings from <I>Boy Scout Magazine</I>
<LI> An affinity for volume controls
</UL>
<P> And, in order of importance, here are the things that little girls are made
of:</P>
<OL>
<LI><P>An instinctive ability to listen and reason. Although relational in their
logic, and often not as <I>spatial</I> and detached in their thinking, a superior
empathetic capability general makes little girls better at conflict resolution.<P>
<LI> Outstanding memories. Little girls can remember things like
addresses with little or no difficulty. Consider this long lost professor of my aging
mother whose address she can still recall:<BR>
<ADDRESS>
1472 Wuthering Heights Circle<BR>
Poetsville, CT 31001<BR>
</ADDRESS>
She visited once, and his dogs were mean to her.</P>
<LI> The gift of <STRONG>Absolute control</STRONG> over all things sentient.
</OL>
</BODY>
```

Notice that, in every instance, only a new `<LI>` tag is capable of creating a new line in the list. Nearly any other type of HTML markup is possible within a given line item. Once you've saved this document, call it up in a browser and notice how it's formatted (see fig. 8.2).

Directories, Definitions, and Menus

Your other lists have something in common with one another that they don't share with ordered and unordered lists: all of them use some permutation of the previous line-item system, but none of them consistently use numbers or bullets. Directories and menus are basically just plain lists. Definitions are unique among all lists because they offer two levels of line items within the list structure-one for the definition item and one for the definition itself.

Directory and Menu Lists

To create a directory or menu list, you start with its respective container tag: <DIR> or <MENU>. Of these two, the directory list is probably more useful. Most browsers don't currently render the <MENU> command consistently-some use a bulleted list, others use no bullets. The following is an example of <MENU>:

```
<MENU>
<LI>House Salad
<LI>Fresh <B>Soup of the Week</B>
<LI>Buffalo Wings
<LI>Escargot
<LI>Liver and Onions
<LI>Turkey Sandwich, <EM>open faced</EM>
<LI>Turkey Sandwich, <EM>pre-fab</EM>
</MENU>
```

Note
You might use the <MENU> tag when creating a list of hypertext links. It's thought that future interpretations of the menu list may be built into future browsers, and that designers will eventually see more benefit in using the <MENU> tag.

In theory, the <DIR> tag is a little more limiting. It's designed as a mechanism for listing computer file directories in HTML pages. Technically, it doesn't support interior HTML tags, although most browsers will display them. The <DIR> tag is also supposed to be limited to 24 characters (for some unknown reason) and show the filenames in rows and columns, like a DIR/W command in MS-DOS, but the bulk of browsers seems to ignore both of these constraints as well, as in the following example:

```
<DIR>
<LI> autoexec.bat
<LI> config.sys
<LI> .signature
<LI> .password
<LI> System Folder
<LI> commaand.com
<LI> .kernel
</DIR>
```

Most browsers (including Netscape) will use the same font and layout for menus and directories as they will for unordered lists. In some cases, browsers will display one or the other (more often directory lists) without a bullet point, which can make them mildly useful. Some browsers can be set to a different font for directories and menus (versus ordered lists). So you may want to use these types, if only because some Web-savvy users' browsers will make an effort to display them differently (see fig. 8.3).

Definition Lists

The final list tag is the definition list, which is designed to allow for two levels of list items, originally conceived to be the defined term and its definition. This is useful in many different ways, though, and is also nice for its consistent lack of bullet points or numbering items (as opposed to the menu and directory listings, which are often rendered haphazardly by browsers).

The tags for this list are the container tag <DL> (definition list) and two empty tags, <DT> (definition term) and <DD> (definition). The <DT> tag is designed (ideally) to fit on a single line of your Web page, although it will wrap to the beginning of the next line if necessary. The <DD> tag will accept a full paragraph of text, continuously indented beneath the <DT> term. The following is an example of all three tags:

```
<DL>
<DT><B>hero</B> <I>(n.)</I>
<DD>A person admired for his or her brave or noble deeds.
<DT><B>hertz</B> <I>(n.)</I>
<DD>A unit used in the measurement of the frequency of electromagnetic waves
<DT><B>hex</B> <I>(n.)</I>
<DD>An evil spell or magical curse, generally cast by a witch.
</DL>
```

Notice that standard HTML mark-up is permissible within the boundaries of a definition list, and that using bold and italics for the defined terms adds a certain dictionary-like quality (see fig. 8.4).

Tip
Not all browsers will display definition lists in the same way, so adding spaces to <DT> items (to get them to line up with the <DD> text) is often a waste of time.

It should also be pointed out that just because definition lists allow for two different types of list items, you needn't necessarily use both. Using just the <DT> tag in your list, for instance, will result in a list not unlike an unordered list-except that nearly all browsers will display it without bullets:

```
<DL>
<DT>Milk
<DT>Honey
<DT>Eggs
<DT>Cereal
</DL>
```

And, although more difficult to find a use for, the <DD> item could be used on its own to indent paragraphs repeatedly. This book occasionally uses a similar device.

```
<P>I must say that I was shocked at his behavior. He was:
<DL>
<DD><I>Rude.</I> Not rude in your standard sort of affable way, or even in a way that
would be justifiable were he immensely weathly or critically wounded. It was just a
rudeness spilling over with contempt.
<DD><I>Unjust.</I> If there was something he could accuse you of falsely,
he would do it. I could almost see him skulking around his apartment after a
particularly unsucessful party, doing his best to find things stolen, which he could
blame on people who hadn't actually bothered to show up.
</DL>
</P>
```

The definition list offers some additional flexibility over the standard lists, giving you more choices in the way you layout the list items (see fig. 8.5).

Example: HTML Within Lists

With the definition list, there are many things you can accomplish with formatting. You can experiment with different HTML tags to see how they react within the list. Remember that, within the <DL> and </DL> tags, the two data item tags, <DT> and <DD>, reign supreme. For instance, even a new paragraph within a <DD> tag will stay indented in most browsers.

Load your template and choose the Save As command to give it a new name. Then type Listing 8.2 between the body tags (see fig. 8.6).

Listing 8.2 `lists2.html` HTML Within Lists

```
<BODY>
<H1>Computer Terms</H1>
<DL>
<DT><B>CPU</B>
<DD>Central Processing Unit. This is the "brain" of a computer, where
instructions created by the computers system software and application
software are carried out.
<DT><B>Hard Drive</B>
<DD>Sometimes called a <I>fixed drive</I>, this is a device (generally
mounted inside a computer's case) with spinning magnetic plates that is
designed to store computer data. When a file is "saved" to the hard drive, it is
available for accessing at a later time.<BR>
Most system software and application programs are also stored on the
computer's internal hard drive. When an applications name is typed or icon is accessed
with a mouse, the application is loaded from the hard drive in RAM and run by the
system software.
<DT><B>Application Software</B>
<DD>Computers programs used to create or accomplish something on a computer, as
distinct from system software. Examples of computer application software might
include:<BR>

WordPerfect (a word processing application)<BR>
Microsoft Excel (a spreadsheet application)<BR>
QuarkXPress (a desktop publshing application)<BR>
Corel Draw (a computer graphics application)<BR>
</DL>
<BODY>
```

Using the `<BR>` tag allows you to create an impromptu list within the list, although everything remains indented because it's ultimately under the influence of the `<DD>` tag. The definition item tags (`<DT>` and `<DD>`) stay in effect until another instance of a definition item tag is encountered or until the `</DL>` tag ends the definition list.

Nesting Tags and Combining List Types

Since most of your HTML lists can accept HTML tags within their list items, it stands to reason that you could potentially create lists within lists. In fact, creating a list, then creating another list as part of an item in that first list is how you can create an outline in HTML.

Nesting Tags

The idea of nesting comes to us from computer programming. Nesting is essentially completing an entire task within the confines of another task. For HTML, that means completing an HTML tag within the confines of another container tag. This could be something like the following:

```
<P>She was easily the most <EM>beautiful</EM> girl in the room.</P>
```

This is an example of correctly nesting the <EM> tag within a paragraph container. On the other hand, many browsers would still manage to display this next code:

```
<P>She was easily the most <EM>beautiful</P> girl in the room.</EM>
```

But this second example is really poorly constructed HTML. It often works, but the <EM> tag isn't properly nested inside the <P>. In this example, that doesn't matter too much, since you can still reason out what this statement is trying to do.

With lists, however, things can get complicated. So it's best to remember the "nesting" concept when you begin to add lists within lists. As far as HTML is concerned, a nested list works as marked-up text within the previous list item. When the next list item is called for, HTML moves on.

Lists Within Lists

Let's look at an example of a simple nested list:

```
<OL>
<LI>Introduction
<LI>Chapter One
     <OL>
     <LI> Section 1.1
     <LI> Section 1.2
     <LI> Section 1.3
     </OL>
<LI>Chapter Two
</OL>
```

Tip
It's a good idea to indent nested lists as shown in the example. The browser doesn't care-it's just easier for you (or other designers) to read in a text editor. (Regardless of your spacing, most browsers will indent the nested lists-after all, that's the point.)

Notice that the nested list acts as a sublevel of the `Chapter One` list item. In this way, you can simulate an outline in HTML. Actually, the nested list is just HTML code that is part of the <LI>`Chapter One` list item. As you saw in Listing 8.2, you can use the
 tag to create a line break in a list element without moving on to the next list item. Following the same theory, an entire nested list works as if it's a single list item in the original list.

The following:

```
<OL>
<LI>Section Five<BR>
   This section discusses ducks, geese, finches and swans.
<LI>Section Six
</OL>
```

is essentially the same as the list that follows:

```
<OL>
<LI>Section Five
     <OL>
     <LI> Ducks
     <LI> Geese
```

```
     <LI> Finches
     <LI> Swans
     </OL>
<LI> Section Six
</OL>
```

In both cases, the nest HTML container is simply a continuation of the first list item. Both the text after the `<BR>` in the first example and the ordered list in the second example are part of the list item labeled `Section Five`. That list item is over when the next list item (`Section Six`) is put into effect (see fig. 8.7).

Combining List Types

When nesting lists, it's also possible to nest different types of lists within one another. This is useful when you'd like to vary the types of bullets or numbers used in an outline form. For instance:

```
<OL>
<LI>Introdution
<LI>Company Financial Update
    <UL>
    <LI>First Quarter
    <LI>Second Quarter
    <LI>Third Quarter
    <LI>Fourth Quarter
    </UL>
<LI>Advertising Update
    <UL>
    <LI>Results of Newspaper Campaign
    <LI>Additions to Staff
    <LI>New Thoughts on Television
    </UL>
<LI>Human Resources Update
</OL>
```

There's nothing terribly difficult to learn here-just the added benefit of being able to nest different types of lists within others. You're still simply adding HTML markup code to items in the original list. This time, however, you have more choice over how your outline looks (see fig. 8.8).

Example: Nesting Definition Lists

Although creating outlines is nice, more often you're interested in presenting actual information on your Web pages. Doing that in an outline form can often be helpful to your Web users. You have a number of different ways you can do that, including nesting paragraphs within ordered and unordered lists. Or you can just use definitions lists.

Load your template and choose the Save As command to rename it. Then enter the following text between the body tags:

```
<BODY>
<H2>About Our Company</H2>
<OL>
<LI>Our Leaders
     <DL>
     <DT><B>Richard B. McCoy, CEO</B>
     <DD> Raised on small farm in Indiana, Dr. McCoy dreamed of something
bigger. By the time he'd graduated from Harvard Business School with an MBA, he'd
```

```
already realized part of his dream. He'd married the most beautiful woman he'd ever
met and was the proud father of a baby girl. From there, his life took control of his
career, and his new found interest in parenting launched his idea of building the
better baby bed. His invention, the SleepMaker 3000, was an instant success. Twenty
years later, he finds his family room couch is enough incentive for him to take a long
nap on Saturdays after a good morning round of golf.
     <DT><B>Leslie R. Gerald, CFO</B>
     <DD> Denying the fact that she's an accountant is nearly a full-time
pursuit for Ms. Gerald. Having graduated at the top of her class at Northwestern
University, her life has been about 1/3 accounting, 1/3 daredevil athleticism and 1/3
sleep. In the meantime she's found time for a steady beau, decorating her mountain
retreat and writing a book called <I>It's More Exciting Than You Think </I> about,
believe it or not, flying ultra-light aircraft.
     <DT><B>David W. Deacon, VP of Marketing</B>
     <DD> Known as "Dave" to anyone he's ever spoken to for more than five minutes,
Mr. Deacon displays the calm friendliness of the consummate salesman, with a twist. He
actually is a nice guy. When he's not doing his best to promote our products, Dave is
well known in the community as a service volunteer. Last year he was awarded Seattle's
prestigous Man of the Year award in recognition of over 500 volunteer hours and over
$50,000 in personal contribution to various area charities.
     </DL>
      <BR>
<LI>Employees of the Month
     <UL>
     <LI> January: Bill Cable, IS
     <LI> February: Janet Smiles, Marketing
     <LI> March: Rich Lewis, Finance
     <LI> April: Wendy Right, Vendor Relations
     <LI> May: Alice Cutless, Area Sales
     <LI> June: Dean Wesley, Training
     </UL>
</OL>
</BODY>
```

Combining different types of lists, then, is a great way to organize your Web site in such a way that it's easy to get at interesting information. At the same time, it's still possible to present that information in many different ways using various list tags (see fig. 8.9).

Review Questions

1. What are the two basic tags in an HTML list?
2. What does a `<LI>` create when used in an unordered list?
3. Can you change the style of numbers in an ordered list (using HTML 2.0 standards)?
4. Which is less likely to display with bullet points-a directory list or a menu list?
5. Can you use other HTML tags (such as `<B>` or `<EM>`) within HTML list containers?
6. What is unique about the definition list style?
7. Do definition lists have to be used for words and their definitions?
8. Does HTML force you to include both a `<DT>` and a `<DD>` tag in your definition lists?
9. Is nesting something that happens only in HTML lists?
10. Which of these is an example of a nested list?
    (A)

```
<OL>
<LI>Groceries<BR>
    Milk<BR>
```

```
    Soup<BR>
    Ice Cream<BR>
<LI>Other groceries
</OL>
```

(B)

```
<OL>
<LI>Groceries
   <UL>
   <LI> Milk
   <LI> Soup
   <LI> Ice Cream
   </UL>
<LI>Other Groceries
</OL>
```

11. What type of HTML lists would you use to create an outline, the major points of which were numbered and the minor points used bullets?

Review Exercises

1. Create a list using the <DIR> and <MENU> tags. View each in your different browser and note how some browsers render these differently from one another.
2. Create a <DL> definition list with nothing but <DD> elements, and one with nothing but <DT> elements. Notice how they're rendered in your browser. Definition lists used in this way are often very useful.
3. Use nested definition lists to create your own HTML "outline." You can use the <DL> elements to number your own outline elements, like the following:

```
<DL>
<DT> I. Introduction
   <DL>
   <DT> A. Welcome!
   <DT> B. Description of Mission Statement
   <DT> C. Conventions in this Report
   </DL>
<DT> II. Chapter One
</DL>
```

Chapter 9

Adding Graphics to Your Web Pages

CONTENTS

Now that you've seen the many ways you can add some character to your text-and use different tags to better communicate your ideas-it's time to jazz up your pages a little bit. Let's add some graphics!

First, though, you should know a couple of important things about placing graphics. Some of these considerations may seem a bit foreign to you, especially if you're a graphic designer or commercial artist. You have to think in a slightly different way about graphics for your HTML pages.

Creating and Manipulating Graphics

It's no secret that a lot of Web design has transitioned from manipulating text-based HTML documents to designing and integrating compelling graphics into Web pages. As the Web has become more commercial, its graphical content has become more professional. If you're not up to the task of creating professional graphics, don't worry too much; programs are available that will help you. Also, it's more important that graphics further the usefulness of the text. The graphics in and of themselves are not the point. The point is to make your Web pages more exciting and informative.

It is a fact, however, that Web sites are leaping forward daily into a more professional, more graphical presentation of Web-based information. Commercial artists and designers are continuing to find new niches on the Web. If you're a skilled computer artist, congratulations; this is where you'll put your skills to use. If you're not, that's OK, too. Any Web designer needs to be able to manipulate and edit graphics in a program such as Adobe Photoshop or CorelDRAW!, but you don't necessarily have to *create* those graphics, if that's not your forte.

Creating Graphics for the Web

As you get started with a program such as Photoshop or CorelDRAW!, keep in mind that the most important consideration in creating Web graphics is the file size. File size isn't generally the first consideration for creating print graphics; almost any print shop or prepress house will accept large storage cartridges or tapes that provide access to your huge full-color graphics. Not so on the Web. Your target is as small as possible-between 15K and 35K for larger (bigger on the screen) files.

You can come up with graphics to use on your Web pages in many ways. Eventually, any graphic that you use needs to be in a standard file format (for example, GIF or JPEG) and relatively small. But how you come up with the final graphic has a lot to do with the information that you're trying to communicate and with your skills as an artist. The following are some of the different ways you might come up with Web graphics:

- **Create graphics in a graphics application**. Many programs for both professional and amateur artists can output GIF- or JPEG-format files for use on the Web. Among these programs are Adobe Photoshop, CorelDRAW!, Fractal Painter, and Fractal Dabbler.

Tip
Any graphics program, even Microsoft Paint, can create Web graphics, although you may need to use another program to change the graphic to an acceptable file format.

- **Download public-domain graphics**. Tons of sites on the Internet allow you to download icons, interface elements, and other graphics for your Web site. At the same time, public-domain clipart collections (such as those available on CD-ROM) can be useful for Web pages.
- **Use scanned photographs**. Using scanned photographs (especially those that you've taken yourself) is a great way to come up with graphics for your Web pages. Unless you have access to scanning hardware, though, you may need to pay someone to scan the photos.
- **Digital cameras**. Cameras are available that allow you to take photos that can be downloaded directly from the camera to your computer. While some of this equipment can be very expensive, cameras under $500 do exist, and those photos can easily be converted for use on the Web.
- **Use PhotoCDs**. Many photo development shops can create digital files of your photographs (from standard 35mm film or negatives) and save those files in PhotoCD format. Most CD-ROM drives allow you to access these photos, which you can then change to GIF or JPEG format and display on your Web pages.

Example: Creating Graphics in Paint Shop Pro

A popular program for creating Web graphics in Windows and Windows 95 is Paint Shop Pro, which has the added advantage of being try-before-you-buy shareware. To download Paint Shop Pro, access the URL **http://www.jasc.com/pspdl.html** with your Web browser, and find the hypermedia link for downloading the program for your particular version of Windows.

Note
As with any shareware program, you should register Paint Shop Pro (by sending in the requested fee) if you find it useful.

Paint Shop Pro arrives as a PKZip-compressed file archive, so you also need a program on your hard drive to unzip it when the download is complete. (WinZip is available from **http://www.winzip.com/**.) Then install the program in Windows and start it. You should see a window like the one shown in figure 9.2.

You can use Paint Shop Pro to create a simple graphic, such as a logo or title, for your Web pages. Using the flood-fill tool, for example, allows you to select a color and "pour" it into the window, creating a background color for the rest of your graphic.

Click the fill-tool icon and then choose a color from the color palette. To apply that color to your graphic, click in the graphic window.

Now select the text tool, choose another color from the palette, and click the graphic window. Type your text (your company name, for example) in the dialog box; then click OK. Now you should be able to drag the text around the window. When you have the text arranged correctly, click anywhere in the window to place the text permanently (see fig. 9.3).

Before you save this graphic, you should make it as physically small as possible so that it works well on your Web page. To cut the image down a bit, select Paint Shop Pro's rectangular selector tool. Click somewhere near the top left corner of the graphic (at the point you want to make the new top left corner of your cropped image), and drag the mouse pointer to the other side (bottom right corner) of the image. When you release the mouse pointer, a thin box should appear around this slightly smaller portion of your graphic. From the menu, choose Image, Crop, and the graphic is cropped to that size. If everything went well, you have a smaller graphic that is just as useful for your Web site.

Our last step is to save the graphic in a file format that's useful for the Web. Choose File, Save As. In the Save As dialog box that appears, you can select the file type from a drop-down list (see fig. 9.4). Select either GIF or JPEG, type a filename, and click OK.

Now you've created a graphic for use on your Web page. Use the Windows Explorer or File Manager to check the file size. You want the file to be somewhere around 20K-an ideal size for a Web page graphic.

Manipulating Web Graphics

After you decide what graphics to use, the next step is to manipulate and edit those graphics for best use on the Web. The preceding section discussed some of this manipulation (cropping and saving a graphic to make it as small as possible). Following are some other ways to use graphics applications to make your images lean, attractive, and useful:

- **Keep graphics small**. Creating smaller graphics in the first place, and using the cropping tool to take out backgrounds and extra space, are great ways to keep graphics to a manageable size.
- **Use fewer colors**. Many graphics applications allow you to decide how much color information should be included in the file. Do you want to use a possible 256 colors or millions of colors? The fewer colors you choose, the smaller your image file will be (see fig. 9.5).

Note
Programs will often describe the number of colors in a graphic using either a number or something called bit-depth. An 8-bit graphic, for instance, offers 256 colors. How do you calculate these numbers? Two to the power of the bit-depth is the number of possible colors (2_8 = 256 colors; 2_{16} = 65536 colors).

- **Create thumbnail graphics**. At times, displaying a large graphic may be necessary, especially if your user chooses to view it. You can give users this option by creating thumbnail graphics in your graphics programs and then using the thumbnails as links to identical (but much larger) graphics files. This method allows you to create pages that contain many images, all of which are scaled down considerably (and, therefore, download more quickly). If a user wants to view one of the graphics at full size, he or she can simply click the thumbnail graphic.

Note
Some browsers (notably, Netscape) can be used to resize the graphics on-the-fly. Although this is convenient for the designer, the entire file still must be transferred across the Internet, thereby negating the benefits that smaller thumbnail graphics offer in terms of downloading speed.

Example: Creating Thumbnails with LView Pro

Another must-have program for most Windows-based Web designers is LView Pro, a shareware graphics-manipulation program. Although the program has some of the same features as Paint Shop Pro, LView is designed less for creating images and more for changing them from one size to another or from one file format to another.

You can download LView by accessing the Web URL **http://world.std.com/~mmedia/lviewp.html**. Choose the version for your flavor of Windows, down-load it to your computer, extract it from its Zip archive, install it in Windows, and start it.

To resize an image to create a thumbnail, follow these steps:

1. Choose File, Open. The Open dialog box appears.
2. In the Open dialog box, find the image that you want to resize.
3. With the image in a window on the desktop, choose Edit, Resize. The Resize Image window appears (see fig. 9.6).
4. Now you can use the slider controls or enter a new size for your thumbnails. A good rule is somewhere around 75 pixels wide (width is the first field after New Size in the dialog box). Changing the width also changes the height in order to preserve the aspect ratio of your images.
5. When you have finished resizing, click OK.

Tip
If you plan to offer many thumbnails on one page, it's a good idea to make them a uniform width (or height) to keep the page orderly.

When you create thumbnails, you'll probably want to maintain the aspect ratio of the current graphic in resizing, so that LView keeps the height and width of the new graphic at the same ratio as the original

graphic, making the thumbnail smaller but similarly proportioned. Don't forget to save the new file with a slightly different name, using the appropriate file extension (GIF or JPG).

Tip
Whenever an application gives you the choice, you should save GIF files as interlaced GIFs and JPEGs as progressive JPEGs. This lets the graphics display faster in many browsers.

Creating Transparent GIFs

One very popular way to edit Web graphics is to create transparent GIFs. This process allows you to make one of the colors of your graphic (generally the background color) transparent, so that the Web page's color scheme or background graphics shows through (see fig. 9.7). Most often, it's used to give the illusion that the graphic is part of your Web page. You can use this method to add impact to your pages and to limit the size of your graphics by doing away with elaborate backgrounds.

To be rendered with a transparent background, a GIF file must be saved in the GIF89a file format. This can be done with Paint Shop Pro, LView Pro, Transparency for the Mac, and many other programs. Saving a file in this format is simply a matter of deciding what color is going to be the transparent color when the GIF is displayed.

Tip
Giving the image in your transparent GIF a shadow (in a graphics application) enhances the appearance of a graphic floating directly over the page.

Creating Transparent GIFs in Transparency for the Mac

One of the easiest ways to create a transparent GIF on the Mac is to use a simple application called Transparency. You can download the program from the Web page **http://www.med.cornell.edu/~giles/projects.html** or **http://www.med. cornell.edu/~giles/projects.html**.

After you download and install Transparency, double-click the program icon to start it. Pull down the File menu and choose Open. In the Open dialog box that appears, open the GIF file that you want to change to a transparent GIF. Your image is then presented in its own window (see fig. 9.8).

Point to the color in the GIF that you want to turn transparent. As you hold down the mouse button, a color palette appears, with the current color selected. If you want that color to turn transparent, release the mouse button. If you want some other color to be transparent (or if you prefer to use no transparency), point to the color that you want to make transparent and release the mouse button. To turn off transparency, simply select the box marked None at the top of the palette.

Now pull down the File menu and choose the Save As GIF89a command. Rename the file (or use the same name, if you want), and save it. The file now should appear in a Web browser as a transparent GIF.

Example: Creating Transparent GIFs in LView Pro

Windows users can create transparent GIFs in LView Pro. To do so, follow these steps:

1. Load the program, and choose File, Open to open a graphics file. The Open dialog box appears.
2. If the file isn't already a GIF image, choose Retouch, Color Depth, and convert the file to a Palette Image.
3. Select 256 colors in the palette creation and quantizing options, and uncheck the Enable Floyd-Steinberg Dithering checkbox.
4. Click OK.
5. Now you can decide which color will appear transparent. Choose Retouch, Background Color and then click the color that should be transparent. You can also use the dropper (click the Dropper button) to select the color that should be transparent (see fig. 9.9).
6. With the correct color selected, choose File, Save As, and save the graphic as a GIF89a. The background color will appear transparent in a Web browser's window.

Embedding Graphics in Web Pages

When your graphics are created, cropped, resized, and saved in the appropriate formats, you're ready to add them to your Web pages. To add graphics, you use an empty tag called the `<IMG>` (image) tag, which you insert into the body section of your HTML document as follows:

```
<IMG SRC="image URL">
```

or

```
<IMG SRC="path/filename">
```

`SRC` accepts the name of the file that you want to display, and *`image URL`* (or *`path/filename`*) *is the absolute (full URL) or relative path (for a local file or a file in the current directory) to the image. As the first example shows, you can display on your page any graphic file that is generally available on the Internet, even if the file resides on a remote server. For graphics files, however, it is much more likely that the file is located on the local server, so a path and filename are sufficient.*

You could enter the following text in a browser:

```
<HR>
<P>This is a test of the Image tag. Here is the image I want to
display:</P>
<IMG SRC="image1.gif">
<HR>
```

In this case, `<IMG SRC="image1.gif">` is a relative path URL, suggesting that the file `image1.gif` is located in the same directory as the HTML document. The result would be displayed by a browser as shown in figure 9.10.

Tip
You'll learn more about absolute and relative URLs in Chapter 10, "Hypertext and Creating Links."

An absolute URL is essential, however, if you were accessing an image on a remote site, as in the following example:

```
<IMG SRC="http://www.graphcom.com/pub/graphics/image1.gif">
```

(This example is fictitious.) Please realize that using a URL to a distant site on the Internet causes that site to be accessed every time this <IMG> tag is encountered on your page, so you should probably have some sort of arrangement with that Web site's system administrator before you link to a graphic on their server.

Adding Graphics to Other HTML Tags

You can add graphics links to HTML tags to do various things, including placing graphics next to text (within paragraphs) and even including graphics in lists. The following example displays the graphic flush with the left margin, with the bottom of the text that follows the image aligned with its bottom edge:

```
<P><IMG SRC="start.gif"> It's time to start our adventure in the world of
the Web. As you'll see below, there is much to learn. </P>
```

Words at the end of the first line wrap below the image (see fig. 9.11).

Another popular use for graphics is including them in HTML lists. Best suited for this task is the <DL> (definition) list, which allows you to use your own graphics as bullet points. (Ordered and unordered lists display their numbers or bullets in addition to the graphic.)

A <DT> (definition term) tag can accept more than one <DD> (definition) element, so you can create a bulleted list as follows:

```
<DL>
<DT>
<DD><IMG SRC="bullet.gif"> This is the first point
<DD><IMG SRC="bullet.gif"> This is the second point
<DD><IMG SRC="bullet.gif"> Here's the third point
<DD><IMG SRC="bullet.gif"> And so on.
</DL>
```

Tip
If you're not up to creating your own bullet points, many archives of common bullets, graphics, and clipart images exist on the Web. Try CERN's images at **http://www.w3.org/hypertext/WWW/Icons** or a popular site like Randy's Bazaar at **http://www.infi.net/~rdralph/icons/**.

At the same time, you could use a definition list in conjunction with thumbnail graphics in a list that uses both the <DT> and <DD> tags. An example might be the following real estate agent's pages (see fig. 9.12):

```
<DL>
<DT><IMG SRC="Small_House14101.GIF">
<DD><EM>14101 Avondale</EM> This executive 3/2/2 is nestled among the live oak, with a
beautiful view of the foothills. $139,900.
<DT><IMG SRC="Small_House3405.GIF">
<DD><EM>3405 Main</EM> This timeless beauty is a cottage made for a prince (and/or
princess!) Spacious 2/1/1 is cozy and functional at the same time, with all-new
updates to this 1880s masterpiece. $89,995.
</DL>
```

The ALT Attribute

None of the HTML tags that you've encountered so far offer the option of a tag attribute-an option that somehow affects or enhances the way the tag is displayed on-screen.

The `ALT` attribute for the `<IMG>` tag is designed to accept text that describes the graphic, in case a particular browser can't display the graphic. Consider the plight of users who use Lynx or a similar text-based program to surf the Web (or users of graphical browsers that choose not to auto-load graphics). Because those users can't see the graphic, they'll want to know what they're missing.

The `ALT` attribute works this way:

```
<IMG SRC="image URL" ALT="Text description of graphic">
```

The following is an example:

```
<IMG SRC="image1.gif" ALT="Logo graphic">
```

For people whose browsers can't display the graphic, the `ALT` attribute tells them that the graphic exists and explains what the graphic is about.

Tip
Test your site with the Load Images option turned off so that you can see how your `ALT` text displays.

The `ALIGN` Attribute

`<IMG>` can accept another attribute that specifies how graphics appear relative to other elements (like text or other graphics). Using the `ALIGN` attribute, you can align other elements to the top, middle, or bottom of the graphic. It follows this format:

```
<IMG SRC="image URL" ALIGN="direction">
```

Note
The `ALIGN="BOTTOM"` attribute isn't necessary, because it is the default setting for the `<IMG>` tag.

The `ALIGN` attribute is designed to align text that comes after a graphic with a certain part of the graphic itself. An image with the `ALIGN` attribute set to `TOP`, for example, has any subsequent text aligned with the top of the image, like in the following example:

```
<IMG SRC="image1.gif" ALIGN=TOP> Descriptive text aligned to top.
```

Giving the `<IMG>` tag an `ALIGN="MIDDLE"` attribute forces subsequent text to begin in the middle of the graphic (see fig. 9.13):

```
<IMG SRC="image1.gif" ALIGN="MIDDLE"> Descriptive text aligned to middle.
```

Order among the attributes that you assign to an image tag is unimportant. In fact, because `SRC="URL"` is technically an attribute (although a required one), you can place the `ALIGN` or `ALT` attribute before the `SRC` information. Anywhere you put attributes, as long as they appear between the brackets of the `<IMG>` tag, is acceptable.

Example: Adding Graphics to Your Web Site

Now that you've learned how to add images to your Web pages, you have almost doubled the things that you can do on the Web. In this example, you add graphics to a typical corporate Web page, using a couple of methods that you've learned.

To start, you need to create some graphics for your home page. If you have a corporate logo and a scanner handy, go ahead and scan in some graphics. Alternatively, you can use a graphics program to create, crop, and save your graphics as GIF or JPEG files. While you're at it, you may want to create some of your GIFs as transparent GIFs.

Create a logo, a special bullet, and a photo for use on the page. Name your GIFs `LOGO.GIF`, `BULLET.GIF`, *and* `PHOTO.GIF`, *or something similar. (If you have already created a Web site, feel free to name the files according to the organizational system that you're using for the site. You can also use JPEG graphics if you so desire.) Then load your HTML template, and save it as a new HTML document. Between the body tags, type something like Listing 9.1.*

Listing 9.1 `images.html` Using `<IMG>` to Create Images

```
<BODY>
<IMG SRC="logo.gif" ALT="RealCorp Logo">
<H1>Welcome to RealCorp's Web Site</H1>
<H2><IMG SRC="photo.gif" ALT="Photo of CEO Bob MacFay" ALIGN=MIDDLE><EM>I'm Bob
MacFay, CEO of RealCorp...</EM></H2>
<P>We at RealCorp make it our business to be as productive and hard working as you
are. That's why we've set up this Web site...to work a little harder, so you don't
have to. Take a look at the various services our company offers, and maybe you'll see
why we like to say, "We're the hardest working corporation all week, every week."</P>
<DL>
<DT>
<DD><IMG SRC="bullet.gif" ALT="-" ALIGN=MIDDLE> Full service plans for any size of
customers
<DD><IMG SRC="bullet.gif" ALT="-" ALIGN=MIDDLE> On-time service calls, any time, any
day of the week
<DD><IMG SRC="bullet.gif" ALT="-" ALIGN=MIDDLE> Fully-equipped mobile
troublshooting vans
<DD><IMG SRC="bullet.gif" ALT="-" ALIGN=MIDDLE> Time honored appreciate for quality
over expediency
</DL>
</BODY>
```

Although the `ALT` *attribute is optional and the bulleted list may survive without it, the example uses ASCII to substitute hyphens for the bullet graphics if the browser can't display images. In most cases, you'll want to describe an image that a user can't view. For an element such as a bullet, though, you can use the* `ALT` *attribute to substitute an ASCII character for the graphic.*

For the photo of the CEO, the `<IMG>` *tag is called within the* `<H2>` *tag, because the* `<H2>` *container (like a paragraph) otherwise would insert a carriage return and force the words* `I'm Bob MacFay...` *to appear below the photo. Including the* `<IMG>` *tag inside the* `<H2>` *tag allows the text to appear next to the photo (see fig. 9.14).*

Play with this example a little bit to get a feel for when you should place the `<IMG>` *tag within another HTML container and when you can leave the tag out on its own. A page sometimes looks completely different, based only on where you place your image tags.*

Review Questions

1. *What's the single most significant concern in creating graphics for display on the Web?*
2. *True or false. The number of colors used to create a graphic can affect the size of the graphic file.*
3. *What are the two most common graphic formats used on the Web? Can you use other formats?*
4. *What does it mean when a graphic format (such as JPEG) is lossy?*
5. *Name four ways that you can obtain graphics for your Web site.*
6. *What is the ideal size range for Web graphics?*
7. *What are thumbnail graphics?*
8. *What specific file format must a GIF be saved in for it to work as a transparent GIF?*
9. *When used with the `<IMG>` tag, what sort of command or HTML element is `SRC`?*
10. *What is the purpose of the `ALT` attribute?*
11. *True or false. The `<IMG>` tag automatically inserts a carriage return after displaying its graphic.*
12. *Why do you never have to set the `ALIGN` attribute to `BOTTOM`?*

Review Exercises

1. *Use your graphics program to save the same graphic as both a GIF and a JPEG image. Then create a Web page that loads both. Note the differences in size and quality.*
2. *Create a GIF image and turn the background transparent with your graphics program (Paint Shop Pro, LView Pro, or Transparency for the Mac, among others). Load both the original and the transparent GIF into your browser (create a Web page if necessary), and notice the difference that transparency makes. Also note whether or not the file size changes.*
3. Use the *`ALIGN`* attribute to an *`<IMG>`* tag to align another image to the top of the first image. Play with this feature, aligning images to *`TOP`*, *`MIDDLE`*, and *`BOTTOM`*.

Chapter 10

Hypertext and Creating Links

CONTENTS

Now that you've seen in detail the ways you can mark up text for emphasis and add images to your Web pages, it's time to take the leap into making these pages useful on the World Wide Web by adding hypertext links. The anchor tag for hypertext links is simple to add to your already-formatted pages. You'll see how URLs are useful for creating hypermedia links and links to other Internet services.

Using the <A> Tag

The basic link for creating hypertext and hypermedia links is the <A>, or anchor, tag. This tag is a container, which requires an </A> to signal the end of the text, images, and HTML tags that are to be considered to be part of the hypertext link. Here's the basic format for a text link:

```
<A HREF="URL">Text describing link</A>
```

Be aware that HREF, although it's something that you'll use with nearly every anchor tag you create, is simply an attribute for the <A> tag. Displayed in a browser, the words *Text describing link would appear underlined and in another color (on a color monitor) to indicate that clicking that text initiates the hypertext link.*

The following is an example of a relative link:

```
<A HREF="products.html">Our Product Information</A>
```

If the HTML document to which you want a link is located elsewhere on the Internet, you simply need a more complete, absolute URL, such as the following:

```
<A HREF="http://www.bignet.net/realcorp/products.html">Our Product Information</A>
```

In either case, things end up looking the same in a browser (see fig. 10.1).

Section Links

Aside from creating hypertext links to documents on your local computer or elsewhere on the Internet, you can create links to other parts of the same document in which the link appears. These "section" links are useful for moving people to a new section that appears on the same Web page without forcing them to scroll down the entire page.

Doing this, though, requires two instances of the anchor tag-one that serves as the hypertext link and another that acts as a reference point for that link, following this format:

```
<A HREF="#section_name">Link to another section of this document</A>
<A NAME="section_name">Beginning of new section</A>
```

Notice that the anchor tag that creates the hyperlink is similar to the anchor tags that you have used previously. The only difference is the pound sign (#) used at the beginning of the `HREF` *text. This sign tells the anchor that it is looking for a section within the current document, as opposed to within an external HTML document.*

The `NAME` *attribute is used to create the actual section within the current HTML document. The text that the* `NAME` *attribute contains is relatively unimportant, and it won't be highlighted or underlined in any way when displayed by a browser.* `NAME` *is nothing more than an internal reference; without it, though, the link won't work.*

Note
Remember to use the pound sign (#) only for the actual hypertext link, not the `NAME` anchor. Also, realize that the `NAME` text is case-sensitive and that the associated `HREF` text should use the same case for all letters as does the `NAME`. If the `HREF` calls for `Section_ONE`, and the `NAME` is actually `Section_One`, the link will not work.

Example: A More Effective Definition List

In Chapter 8, "Displaying Text in Lists," you worked with the definition list tags available to use in HTML and, in some cases, actually used them for a list of definitions. You do that again in this section, but this time you use section links to move directly to the words that interest you.

Load the HTML template into your text editor, and choose the Save As command in your text editor to create a new file. In the body of your HTML document, type Listing 10.1 or something similar.

Listing 10.1 `listlink.html` Creating a Definition List

```
<BODY>
<H2>The Definition List</H2>
<P>Click one of the following words to move to its definition in the list:
```

```
<BR>
<A HREF="#EPITHET">epithet</A><BR>
<A HREF="#EPITOME">epitome</A><BR>
<A HREF="#EPOCH">epoch</A><BR>
<A HREF="#EPOXY">epoxy</A><BR>
<A HREF="#EQUAL">equal</A><BR>
</P>
<HR>
<DL>
<DT><A NAME="EPITHET"><B>ep i thet</B></A>
<DD><EM>noun.</EM> a descriptive, often contemptuous word or phrase
<DT><A NAME="EPITOME"><B>ep it o me</B></A>
<DD><EM>noun.</EM> someone who embodies a particular quality
<DT><A NAME="EPOCH"><B>ep och</B></A>
<DD><EM>noun.</EM> a division in time; a period in history or geology
<DT><A NAME="EPOXY"><B>ep ox y</B></A>
<DD><EM>noun.</EM> a synthetic, heat-sensitive resin used in adhesives
<DT><A NAME="EQUAL"><B>e qual</B></A>
<DD><EM>adj.</EM> having the same quality or status; having enough strength, courage,
and so on.
<DD><EM>noun.</EM> a person or thing that is equal to another; a person
with similar rights or status
</DL>
</BODY>
```

In the example, clicking one of the words that appears as a hyperlink in the first section of the paragraph moves the browser window down to that link's associated `NAME` anchor, so that the definition becomes the focal point of the user's attention. Obviously, using section links would be of greater use in a larger list. Consider the implications for turning an entire dictionary into HTML documents.

Also notice that anchors can be placed within the confines other HTML tags, as in the first paragraph container and in the definition lists of the example. In general, anchor tags can be acted on by other HTML tags as though they were regular text. In the case of hyperlinked text, the underlining and change in color in graphical browsers take precedence, but the hyperlinked text also has any other qualities of the surrounding text (for example, indenting with the rest of the definition text).

In figure 10.2, notice which anchors cause the text to become a hyperlink and how the anchor tags respond within other container tags.

Using Relative URLs

Go back and look at the hypertext links that we discussed at the beginning of this chapter (as opposed to section links). In most cases, the URL referenced by the `HREF` attribute within the anchor tag needs to be an absolute URL, unless it references a file located in the same directory as the current HTML document.

But consider the case of a well-organized Web site, as set out in Chapter 5, "What You Need for a Web Site." That chapter discussed the fact that it's not always the best idea to drop all your Web site's files into the same directory, especially for large sites that contain many graphics or pages. How do you create links to files that may be on the same server but not in the same directory?

*One obvious way is to use an absolute URL for every link in your Web site. If the current page is **http://www.fakecorp.com/index.html**, and you want to access a specific page that you organized into your `products` directory, you could simply create a link like the following, using an absolute URL:*

```
<A HREF="http://www.fakecorp.com/products/new_prods.html>Our new
products</A>
```

These absolute URLs can get rather tedious, not to mention the fact that if you happen to change the name of your Web server or move your site to another basic URL, you'll probably have to edit every page in your site to reflect the new URLs.

Adding the <BASE> Tag

The <BASE> tag is used to establish the absolute base for relative URLs used in your document's hypertext links. This tag is especially useful when your Web pages may appear in different subdirectories of a single main directory, as in some of the organizational types discussed in Chapter 5. The format of the <BASE> tag is as follows:

```
<BASE HREF="absolute URL">
```

Note that the <BASE> tag is designed to appear only between the <HEAD> tags.

It may be helpful to think of <BASE> as doing something similar in function to a DOS path statement. The <BASE> tag tells the browser that relative URLs within this particular Web document are based on the URL defined in the <BASE> tag. The browser then assumes that relative URLs derive from the URL given in the <BASE> tag and not necessarily from the current directory of the HTML document.

Consider a document named ***http://www.fakecorp.com/products/list.html*** *that looks something like this:*

```
<HEAD>
<TITLE>Page One</TITLE>
</HEAD>
<BODY>
<A HREF="index.html">Back to Index</A>
</BODY>
```

In this example, the browser tries to find a document named `index.html` *in the directory* `products`*, because the browser assumes that all relative addresses are derived from the current directory. Using the <BASE> tag, however, changes this example a bit, as follows:*

```
<HEAD>
<BASE HREF="http://www.fakecorp.com/">
<TITLE>Page One</TITLE>
</HEAD>
<BODY>
<A HREF="index.html">Back to Index</A>
</BODY>
```

Now the browser looks for the file `index.html` *in the main directory of this server, regardless of where the current document is stored (such as in the* `products` *directory). The browser interprets the relative URL in the anchor tag as though the complete URL were* ***http://www.fakecorp.com/index.html****.*

Tip
If you plan to create a large Web site, you may want to add the <BASE> tag (complete with the base URL) to your HTML template file.

Using the <BASE> tag to point to your Web site's main directory allows you to create the different types of organization systems described in Chapter 5 by using relative URL statements to access HTML documents in different subdirectories.

Example: A Hybrid-Style Web Site

Chapter 5 discussed the hybrid style of Web site organization, which allows you to put some common files (such as often-used graphics) in separate directories and to organize unique files with their related HTML pages.

*In this example, you create an HTML document called `products.html`, located at the URL **http://www.fakecorp.com/products/products.html**. Some of your graphics are maintained in a subdirectory of the main directory of this Web site; the subdirectory is called `graphics/`. You also have links to other pages in the main directory and in a subdirectory called `about/`. Figure 10.3 shows this graphically.*

For this example, you create only one Web page. To test the page, however, you want to create a directory structure similar to the previously outlined directory structure and include all the files mentioned.

Begin by saving your template file as `products.html`. Then, in your text editor, enter Listing 10.2.

Listing 10.2 `basetag.html` Creating a Directory Structure

```
<HTML>
<HEAD>
<TITLE>Our Products</TITLE>
<BASE HREF="http://www.fakecorp.com/">
</HEAD>
<BODY>
<IMG SRC="products/prod_ban.gif">
<H2>Our Products</H2>
<P>Here's a listing of the various product types we have available. Click
the name of the product category for more information:</P>
<DL>
<DT>
<DD><IMG SRC="graphics/bullet.gif"> <A HREF="products/pc_soft.html">
PC Software</A>
<DD><IMG SRC="graphics/bullet.gif"> <A HREF="products/mac_soft.html">
Macintosh Software</A>
<DD><IMG SRC="graphics/bullet.gif"> <A HREF="products/pc_hard.html">
PC Hardware</A>
<DD><IMG SRC="graphics/bullet.gif"> <A HREF="products/mac_soft.html">
Macinotsh Hardware</A>
</DL>
<HR>
<A HREF="index.html">Return to Main</A>
</BODY>
</HTML>
```

Notice that all the hypertext link `HREF` commands are pointing to pages that are relative to the <BASE> URL, which is set for the main directory of the Web site. With <BASE> set, it's no longer appropriate simply to enter a filename for your relative URL, even if the file is in the current directory (for example, `products/`). If all goes well and all your references hold up, your page is displayed as shown in figure 10.4.

Note
Notice that the <BASE> HREF also affects graphics placed with the <IMG> tag. Remember to use relative addresses for images that take the <BASE> address into account. Only HTTP documents and images are affected by <BASE>, though, and not other URL types (like **ftp://** and **gopher://**).

Creating Links to Other Internet Services

Here's where the real power of URLs comes into play. Remember that an URL can be used to describe almost any document or function that's available on the Internet? If something can be described in an URL, a hyperlink can be created for it. In the following section, you start with e-mail.

Hyperlinks for E-Mail Messages

Creating a hyperlinked e-mail address is simple. Using the `mailto:` type of URL, you can create the following link:

```
<A HREF="mailto:tstauffer@aol.com">Send me e-mail</A>
```

In many graphical browsers, this URL often loads an e-mail window, which allows you to enter the subject and body of an e-mail message and then send it via your Internet account (see fig. 10.5). Even many of the major online services support this hyperlink with their built-in e-mail systems.

Not all Web browsers accept the `mailto:` style of URL, however, and most of those don't return an error message. If you use this type of link, you may want to warn users. Something like the following text should work well for users of nongraphical browsers:

```
<P>If your browser supports the mailto: command, click <A
HREF="mailto:tstauffer@aol.com">here</A> to send me an e-mail message.
</P>
```

Other Internet Services

Using the various types of URLs discussed in Chapter 3, you can create links to nearly all other types of Internet services as well. For Gopher sites, for example, a hypertext link might look like the following example:

```
<A HREF="gopher://marvel.loc.gov/">the Library of Congress Gopher</A>
```

Most Web browsers can display Gopher menus. In most cases, clicking a gopher link points the browser at the Gopher site, and the Gopher menu appears in the browser window.

You can create links that cause the Web browser to download a file from an FTP server, as follows:

```
<P>You can also <A HREF="ftp://ftp.fakecorp.com/pub/newsoft.zip">download</A>the
latest version of our software.
```

When the connection to the FTP server has been negotiated, the file begins to download to the user's computer (see fig. 10.6). Depending on the Web browser, this file may not be formatted correctly. Each

browser needs to be set up to accept files of a certain type (such as the PKZip format file in the preceding example).

Note
Most browsers can accept hyperlinks only to anonymous FTP servers. You generally should not include in your HTML documents links to FTP servers that require usernames and passwords.

Again, most browsers have some mechanism (sometimes built into the browser window) for reading UseNet newsgroups. Some browsers launch a separate program to read UseNet groups. In either case, you can create a link like the following:

```
<A HREF="news:news.answers">UseNet Help Newsgroup</A>
```

This link loads whatever UseNet reading features the browser employs and displays the specified newsgroup (see fig. 10.7). As discussed in Chapter 3, the `news:` URL type does not require a particular Internet server address to function. Each browser should be set up with its own links to the user's news server.

Other Links for the `<HEAD>` Tag

You can create a couple more tags in the `<HEAD>` section of your HTML documents. These tags are of varying levels of real-world usefulness, so you may want to read this section quickly and refer to it again later if you have a question. The two tags discussed in the following sections are `<LINK>` and `<ISINDEX>`.

The `<LINK>` Tag

The `<LINK>` tag is designed to establish a hypertext relationship between the current document and another URL. Most of the time, the `<LINK>` tag does not create a clickable hypertext link in the user's Web viewer window. It's a little beyond the scope of this book, but programs can be written to take advantage of the `<LINK>` tag, such as a program that creates a toolbar that makes use of the relationship defined.

The `<LINK>` tag generally has either of the following formats:

```
<LINK HREF="URL" REL="relationship">
```

or

```
<LINK HREF="URL" REV="relationship">
```

For the most part, `<LINK>` is used to create an author-defined structure to other HTML documents on a Web site. The attribute `REL`, for example, defines the relationship of the `HREF` URL to the current document. Conversely, `REV` defines the relationship between the current document and the `HREF`'ed URL.

Following are two examples of `<LINK>` statements:

```
<LINK HREF="http://www.fakecorp.com/index.html" REL="PARENT">
<LINK HREF="http://www.fakecorp.com/product2.html" REV="CHILD">
```

In the HTML 2.0 standard, these definitions are relatively irrelevant-at least publicly on the Web. You more commonly find these statements used within certain organizations (perhaps companies employing an intranet), especially for advanced Web-based documentation efforts and for efforts that use HTML and SGML (as discussed in Chapter 1, "What is HTML?") together.

HTML 3.0 more than likely will introduce more widespread use of the `<LINK>` statement and other `<HEAD>` tags for more tangible benefits.

You may want to use one `<LINK>` frequently: the `REV="MADE"` link, which tells users who created the HTML page. Although this use of `<LINK>` doesn't actually call up a `mailto:` link in most browsers, some may recognize it eventually. In the meantime, it gives people who view your source code the e-mail address of the author, as in the following example:

```
<LINK HREF="mailto:tstauffer@aol.com" REV="MADE" REL="AUTHOR">
```

You also should include a `mailto:` anchor tag in the body of your document to allow people to respond to your Web page. Using both is encouraged, but it's ultimately up to you.

Tip
You can find more information about `<LINK>`, and the various values for `REL/REV`, at **`http://www.sq.com/papers/Relationships.html`**.

The `<ISINDEX>` Tag

Adding the `<ISINDEX>` tag to the `<HEAD>` of your document allows some Web-server search engines to search your Web pages for keywords. If your Web server offers such a search engine and the user's browser supports these searches, the user will be presented with a simple search box when this page is loaded. The user can then enter the text he or she wants to search for on your page.

The tag itself is very straightforward and requires no further attributes, as the following example shows:

```
<HEAD>
<ISINDEX>
</HEAD>
```

Note
If someone else runs your Web server, you may want to ask that person whether you should include the `<ISINDEX>` tag. If the administrator offers a server-based search engine, he or she may have you use the `<ISINDEX>` tag, or he or she may insert it into your document himself or herself.

Example: A Graphical, Hyperlinked Listing

Another interesting use of lists and hypertext links features the `<DL>` list, with an interesting twist. This example throws in thumbnail versions of some graphics that suggest what the links access. The user can access a link by clicking the associated graphic.

This example shows a popular HTML menuing format; it offers a low-bandwidth way to offer a visual reference for a database-style Web site. On a page such as this, you could list artwork, movie reviews, other

Web sites, a company's products, a list of people, screen shots of computer programs, or just about anything else graphical.

Create a new HTML document from your template, and then enter text and tags according to the example in Listing 11.3.

Listing 11.3 `linkmenu.html` Creating a Graphics Listing

```
<BODY>
<H2>Suggested Search and Directory Pages</H2>
<P>Ready to Search the Net? Click the associated icon to jump to that
particular Web search page.</P>
<DL>
<DT><A HREF="guide.infoseek.com"><IMG SRC="infoseek.gif"> The Infoseek
Engine</A>
<DD>Infoseek offers a broad range of searching and directory options, and
is a fine place to start your search on the Web. It's also possible to
search other services, like UseNet and Classifieds. <I>Tip:</I> For best
results, put proper names or complete phrases in quotes, like "Microsoft
Windows".
<DT><A HREF="www.yahoo.com"><IMG SRC="yahoo.gif"> The Yahoo Directory</A>
<DD> Widely regarded as the earliest attempt to organize the Web, Yahoo
remains a formidable directory of links to useful sites. Searching isn't as
comprehensive as some others, but the directory is the main reason to use Yahoo,
anyway.
<DT><A HREF="www.lycos.com"><IMG SRC="lycos.gif"> The Lycos Search Engine</A>
<DD> Image isn't everything, and Lycos doesn't give the prettiest search
results. But if you're comfortable with relatively plain listings, Lycos
offers one of the larger databases of Web Sites available.
</DL>
</BODY>
```

Review Questions

1. *Is `HREF` a tag or an attribute?*
2. *Do local links and distance links look any different when they are viewed in a browser?*
3. *What type of link is `<A HREF="#intro">Intro</A>`? Can you tell from the link what document will be accessed?*
4. *Is it possible to include HTML markup tags (such as emphasis tags) inside anchor tags?*
5. *What is the purpose of the `<BASE>` tag, and in what part of the HTML document does it appear?*
6. *True or false. The `<BASE>` tag's `HREF` attribute requires a relative URL.*
7. *Would the following link succeed? (Assume that the e-mail address is correct.)*

   ```
   <A HREF="mailto://buddy@aol.com">Mail me!</A>
   ```
8. *What two attributes for the `<LINK>` tag are discussed in this chapter?*
9. *Does the `<LINK>` tag create a hypertext link in the browser window?*
10. *If your Web server is administered by someone else, what's the best way to find out whether the `<ISINDEX>` tag will do you any good?*

Review Exercises

1. *Create a hypertext link that points to a section of another document. (Hint: use the URL and a section name, like **http://www.fakecorp.com/products.html#clothing**). Don't forget the `NAME` anchor in the second document.*
2. *Using the `<BASE>` tag, change the following so that the URL and image SRC attribute are relative:*

```
<BODY>
<IMG SRC="http://www.fakecorp.com/images/logo.gif">
<P> Welcome to <A HREF="http://www.fakecorp.com/about.html">BigCorp</A> on the
World Wide Web!</P>
</BODY>
```

3. *Create a page about your hobbies and interests. (This might be a great About page for your personal Web site.) On the page, include links to interesting sites that coincide with your description. (For instance, if you like sports, you might create a link to **http://www.cnn.com/SPORTS/** for the benefit of your users.)*

Chapter 12

Clickable Image Maps and Graphical Interfaces

CONTENTS

In Chapter 11, "Using Links with Other HTML Tags," you spent some time creating clickable images, which make Web pages more graphically appealing and (ideally) a little more intuitive. This chapter takes creating a graphical interface to your Web site one step further.

With image maps, you can create an entire interface for your Web pages and sites that rivals the interfaces of popular multimedia games, graphical operating environments, and interactive kiosks. The first 11 chapters of this book have said that the Web is about text, but that fact doesn't mean that you can't use some great graphics to spice up your presentation.

Image Maps Defined

The *map* part of *image map* conjures up two separate images. First, image maps on Web sites often act like road maps for the Web site, adding interface elements that make it easier to get around on the Web site. Second, the word *map* also suggests the way that image maps are created. Image maps begin life as normal graphics (usually in GIF or JPEG format), designed with the Web in mind. Then another program is used to map *hot zones* (clickable areas) on top of the graphics.

When put in place on a Web page, an image map allows users to access different HTML pages by clicking different parts of the graphic. Because each hot zone has an associated URL, and because each hot zone corresponds to part of the graphic, maneuvering about a Web site becomes more interesting, graphical, and interactive.

Example: The Apple Web Site

Apple Computer offers a very interesting example of an image map on the main page of its Web site. To check out the page, load your graphical Web browser, connect to the Internet (if you're not already connected), and enter **http://www.apple.com/**.

When the page loads in your browser, you'll see the interface, which looks a little like a futuristic hand-held computer, on-screen.

Note
Notice how long it can take a graphical interface to load over your connection, especially if you use a modem to access the Internet.

This example isn't terribly structured, but it allows you to play with the image map interface. You may already have a good deal of experience with such interfaces, especially if you've spent a lot of time on the Web.

By simply pointing at part of the graphic, you may be able to bring up a URL in the status bar at the bottom of your browser bar (see fig. 12.1). This bar shows you where the various hot zones for the image map are and at what coordinates your mouse pointer appears.

Check out one more thing. If the image map fills your screen, scroll down in your browser window so that you can see what's below the interface on Apple's Web page. The text directly below the interface almost exactly mirrors the hyperlink options you have with the image map, because image maps, unlike clickable graphics, don't offer an `ALT` statement for the various hot zones. So you have to include additional links to cater to your users of nongraphical browsers.

Understanding How Image Maps Work

Creating an image map involves three steps: creating the graphic, mapping the graphic for hot zones, and placing the correct information (along with the correct programs) on the Web server itself. This section discusses the Web server; the next section talks about defining hot zones.

For more information on creating graphic images for Web pages, see Chapter 9, "Creating and Embedding Graphics."

To offer your users the option of using image maps, you must have a special map server program running on your Web server. For UNIX-based servers, this program will most often be NCSA Imagemap; other platforms have their own map server programs.

The Map Server Program

When a user clicks an image map on a Web page, the browser determines the coordinates of the graphic (in pixels) that describe where the user clicked. The browser then passes these numbers to the map server program, along with the name of the file that contains the URLs that correspond to these coordinates.

NCSA Imagemap, then, simply accepts the coordinates and looks them up in the database file that defines the hot zones for that image map. When NCSA Imagemap finds those coordinates and their associated URL, it sends a "connect to URL" command (just as a hypertext link does) that causes your browser to load the appropriate HTML document.

Note
If you're running your own WebStar or MacHTTP server from a Macintosh, you can use a map server called MapServe, which you can download from `http://www.spub.ksu.edu/other/machttp_tools/mapserve/mapserve.html`. For the most part, other commercial Web servers for UNIX and Windows NT include map server capabilities.

The Map Definition File

To determine which parts of the image map are linked to which URLs, the map server program must have a map definition file at its disposal. This file is generally a text file with the extension MAP, stored somewhere in the `CGI-BIN` directory for your Web site. Exactly where this file is stored depends on the combination of your Web server and map server. Let it suffice to say that you'll need to consult your server's documentation or your ISP.

The map definition file looks something like figure 12.2. You can create this file and save it as a standard ASCII text file with the appropriate extension; fortunately, you probably won't have to.

You can define different shapes in the file; these shapes correspond to the shapes of the hot zones that overlay the graphic that you want to use for your image. Each set of coordinates creates a point on the graphic. The coordinates are expressed in pixels, with each pair of numbers representing the number of pixels to the right and down, respectively, from the top left corner of your graphic.

The shapes require a different number of points to define them. Rectangles require two points, for example, and polygons require as many points as necessary. Luckily, the number of points involved isn't something that you'll have to worry about. Simply by using a map editing program for Windows or Macintosh (discussed later in this chapter in the sections, "MapEdit for Microsoft Windows and XWindows" and "WebMap for Macintosh"), you can automatically create the map definition file required for your map server.

Note
You can create image maps without map servers and map definition files by using a technology called client-side image maps. Currently a Netscape technology, this technology eventually may become an HTML 3.0 standard. For more information, see Chapter 17, "Client-Side Image Maps."

The Various Shapes of Hot Zones

This section briefly defines the shapes of hot zones. Hot zones can be in any of the following shapes:

- **rect (rectangle)**-This shape requires two points: the upper left coordinates and the lower right coordinates.
- **circle**-To create a circular region, you need coordinates for a center point and an edge point. The circle is then computed with that radius.
- **point**-A point requires only one coordinate. The map server software decides which point the mouse pointer was closest to when the shape was clicked (provided that the click didn't occur in another hot zone).
- **poly (polygon)**-You can use up to 100 sets of coordinates to determine all the vertices for the polygon region.
- **default**-Any part of the graphic that is not included in another hot zone is considered to be part of the default region, as long as no point zones are defined. If a point is defined, then default is redundant, since the map server will evaluate any click (outside of a hot zone) and choose the nearest point.

Defining Your Image Map Hot Zones

As a designer, you are responsible for doing two things in the hot zone definition process. First, you need to define the hot zones to create the image map-that is, you need to decide what URL the coordinates will correspond to when the image map is clicked. Second, you need to create the map definition file that makes the hot zone information available to the Web server. For Windows and Macintosh users, luckily, programs that do both are available.

MapEdit for Microsoft Windows and X-Windows

Available for all flavors of Windows (Windows 95, Windows 3.1, and Windows NT) and for most types of UNIX, MapEdit is a powerful program that allows you to graphically define the hot zones for your image maps. You can access and download the latest version of this program via the MapEdit Web site (**http://www.boutell.com/mapedit/**).

When you have the program installed and you double-click its icon to start it, follow these steps to define your map:

1. Choose File, Open/Create from the MapEdit menu. The Open/Create Map dialog box appears.
2. In the Open/Create Map dialog box, enter the name of the map definition file you want to create and the name of the graphic file you want to use for your map. You should also use the radio buttons to determine whether you'll use CERN or NCSA map definitions. (Consult your map server software or ISP if you're not sure whether to use CERN or NCSA.)
3. Click the OK button. The Creating New Map File dialog box appears. In this dialog box, click Yes. After a moment, MapEdit displays your image file.
4. To create a new hot zone, choose the shape from the Tools menu; then click one time for each point required for the shape. For a rectangle, click once to start the rectangle and then click where you'd like the opposite corner of the triangle to appear. For a circle, click for the middle, and then drag out the circle and click when you've got the right radius. The triangle tool is actually a "polygon" tool, so click for each point in the polygon. Then, right-click at the last point (to connect your last point to the first point and complete the shape).
5. When the shape is created, the Object URL dialog box appears (see fig. 12.3). Enter the URL that you want to associate with your new hot zone. (You also can enter comments, if you want.) Then click OK to continue.
6. Add more shapes by following steps 4 and 5 until you finish mapping your graphic.

7. Choose File, Save. Now you have a `.MAP` file for your image map.

Tip
By choosing File, Edit Default URL, you can determine whether your image map includes a default URL for clicks outside your hot zones.

Example: MapEdit and a Simple Button Bar

In this example, you use MapEdit to create a simple button bar-a little like the menu bar that you created with clickable graphics in Chapter 11, except for the fact that this one is an image map. Start by drawing an appropriate graphic in a graphics application and saving it as a GIF file. For this example, name the file `testbar.gif`. Then follow these steps:

1. Open MapEdit, and choose File, Open/Create. The Open/Create Map dialog box appears.
2. In this dialog box, enter **testbar.map** for the map file and **testbar.gif** for the graphics file. (If you saved the GIF file in a different directory, use the Browse button to find and load it.)
3. When the graphic loads, pull down the Tools menu and make sure that Rect is selected.
4. Draw rectangles for the buttons, providing an appropriate URL for each button. For this example (four buttons in all), use the following URLs:
   http://www.fakecorp.com/index.html
   http://www.fakecorp.com/product.html
   http://www.fakecorp.com/service.html
   http://www.fakecorp.com/help.html
5. Choose File, Edit Default URL. The Default URL dialog box appears.
6. Enter the following URL:
   http://www.fakecorp.com/error.html
7. Choose File, Save.
8. Choose File, _Quit.

You've created your map definition file. To look at the file, open Notepad (or a similar text editor), and load the file `testbar.map` into it. The file should look something like figure 12.4 (although the coordinates are bound to be slightly different).

WebMap for Macintosh

If you're a Macintosh user, you can use a program called WebMap, which is similar to MapEdit. You can download WebMap from **http://www.city.net/cnx/software/webmap.html**. Install the program; then double-click its icon to start it.

To create an image map in WebMap, follow these steps:

1. Choose File, Open.
2. In the Open dialog box, select the graphic that you want to use for your map and the name of the map definition file that you want to create.
3. Click the OK button. After a moment, MapEdit displays your image file.
4. To create a new hot zone, choose the shape from the floating tool palette, and drag to create a hot zone. For a rectangle, circle, or oval, click and hold the mouse in the top left corner of your shape, drag the mouse to make the shape the desired size, and then release the mouse button. To create a

polygon, choose the polygon shape from the tool palette and then click once on the graphic for each point in your polygon. To complete the shape, click once on the first point you created.

5. When the shape is created, enter the URL in the space provided above the graphic file (see fig. 12.5). You can use the pointer tool (the one that looks like a mouse pointer) to select different shapes that you've created and then edit their URLs.
6. To create a default URL, use the pointer tool to click the graphic background (not a shape). `Default URL` should appear in the comment window. Then enter the default URL in the URL text box.

To create your map definition file, pull down the File menu and choose Export As Text. In the resulting dialog box, you can name your map file and save it in CERN or NCSA format. Now you're free to save the graphic and quit the program.

Adding Image Maps to Your Web Page

After you create your image map and your map definition file, you're ready to add a link for your image map to your HTML page. You can accomplish this task in a couple of ways, depending on your Web server. In essence, though, the only major difference between an image map and a clickable image (refer to Chapter 11) is a new attribute for the `<IMG>` tag: `ISMAP`.

Image maps follow this format:

```
<IMG SRC="graphic.ext" ISMAP>
```

Note
It's perfectly acceptable to add other `<IMG>` tag attributes (such as `ALT`) to your image map definition.

Using the `ISMAP` attribute doesn't do much for you unless the image map is also a hyperlink, so the following code is everything that you need to add an image map to your Web page:

```
<A HREF="URL"><IMG SRC="graphic.ext" ISMAP></A>
```

Our next step is to figure out what to use as the URL in this hyperlink.

The Image Map URL

The URL that you're interested in accessing isn't a particular Web page, because using an URL to a particular Web page would defeat the image map concept; the link would act like a regular clickable graphic. Instead, you want the URL to access the map definition file. You'll have to ask your ISP (or figure out for yourself) where on the server the map file is stored.

Some Web servers allow you to store the map definition file anywhere on the server; the servers are smart enough to figure out that you're accessing a map definition file and take care of the rest. In that case, you could simply store the map definition file in the current directory and access it as follows:

```
<A HREF="mymap.map"><IMG SRC="mymap.gif" ISMAP></A>
```

If you have an understanding server, this method may work for you.

Other servers may require you to access a particular directory on the server, such as the `/cgi-bin/` or `/bin/` directory, where server scripts (mini computer programs) are stored. In such a case, something like the following examples may be the way to access the image map:

```
<A HREF="http://www.myserver.com/cgi-bin/mymap.map><IMG SRC="mymap.gif" ISMAP></A>
```

or

```
<A HREF="http://www.myserver.com/bin/mymap.map><IMG SRC="mymap.gif" ISMAP></A>
```

If the server requires you to access one of these scripting directories, though, it may not want you to access the map definition file directly. Instead, the server will want you to use an alias.

Some servers store all map information in a single database file (often called `imagemap.conf`) and require you to access information within the database by using an alias. You and your Web server administrator have to determine what this alias is. In that case, your link would look more like the following:

```
<A HREF="http://www.myserver.com/bin/mymap><IMG SRC="mymap.gif" ISMAP></A>
```

Example: Testing Your Link

The best way by far to participate in this example is to confer with your ISP, place your map definition file on the Web server, and test it from a remote location using the correct URL. If that procedure doesn't work, you can manage some testing on your own.

Save your template as a new HTML file, and have an image-mapped graphic handy in the same directory. Then enter Listing 12.1 between the `<BODY>` tags.

Listing 12.1 `img_map.html` Adding Image Maps in HTML

```
<BODY>
<A HREF="http://www.server.com/mymap.map><IMG SRC="mymap.gif" ISMAP ALT=
"My Image Map"></A>
<H2>Welcome to my page!</H2>
</BODY>
```

Note
If you're going to test this example on an actual Web server, you need to replace the URL with the appropriate one for your Web site (and add the type of link to your map info file that's required for your server). Also, use the real name of the mapped GIF file in the `<IMG>` tag.

Save the HTML file and then load it in a graphical browser. If your graphic came up, chances are that you set the `<IMG>` tag correctly. Notice that many browsers do not display a colored link border around the graphic, because the graphic is now considered to be an image map.

Before clicking any of the hot zones, move your mouse pointer around on the image map graphic. If you have a status bar at the bottom of your browser window, you may notice that the link keeps changing (see fig. 12.6). Along with the URL of your map definition file, you should be seeing the current coordinates of your pointer. All this information is sent to the map server to help it figure out what region you clicked. (If

you're testing this image map from your local drive, the status bar test is the only part of the example that will work.)

Now, if you are testing your image map on the Web server, go ahead and click the map to make sure that all the links work. If you're viewing the image map locally, turn off the graphics-loading option in your browser, and reload the page. You should notice that there's now no way to access the hyperlinks in the image map-that's why you also need text links for your image map pages.

Image Map Design Tips

This chapter has covered creating and linking an image map to your Web page fairly thoroughly. Image maps are a bit of a departure from standard text-markup HTML, however, so you should learn a little bit of design theory and Web-related netiquette before you leave this chapter. Please try to keep some of the following suggestions in mind when you add image maps to your Web pages:

- **Use image maps sparingly**. The best way to use an image map is as a clickable menu bar or some other easy-to-recognize interface element. The point isn't really to see how graphical you can make your Web pages-just how intuitive.
- **Remember that image maps are usually little more than big graphics files**. Ultimately, the key to graphics on the Web is keeping them small. Even if your image map is incredibly attractive, users will be annoyed if they have to wait many minutes for their four possible choices to download to their browsers. Use all the tips in Chapter 9 to keep your graphic as small as possible, and use image maps only to enhance usability.
- **Image maps require redundant text links**. Unless you plan to leave out everyone who can't view your graphics, you need to create text links that do everything that your image map does. Remember that with clickable graphics, the `ALT` attribute takes care of the problem. The `ALT` attribute doesn't work for image maps, because a single image map graphic can have many links, so you need to create an identical text link on your page for every hot zone link in your image map.
- **Stick to normal shapes whenever possible**. Rules are made to be broken, but in general, you should try to be conservative with your image maps (see fig. 12.7). A graphic that looks as though it has rectangular buttons should function as though it has rectangular buttons. In other words, make your hot zones correspond logically to the image map graphics. Random hot zones randomly annoy users.

Review Questions

1. Why are the graphics discussed in this chapter called *image maps*?
2. What three steps do you follow to create an image map?
3. What file format is the map definition file saved in?
4. Is it important to know what type of map server program your Web server is using? Why or why not?
5. How do you find out where to store your map definition file?
6. How many points are required for a rectangle in a map definition file? What is the maximum number of points that you can use for a polygon?
7. True or false. You can create an image map without a map editing program.
8. Which files must you create for an image map to work?
9. Do the shapes (rect, poly, point, and so on) that you draw in a map editing program show up in the Web browser window? Why or why not?
10. Why is defining a default map definition redundant if you have already defined a point?

11. Aside from the URL to the map definition file, what information does the Web browser send to the Web server? What does the designer do to make this happen?

Review Exercises

1. Create two different map definition files for the same graphic, one using the CERN method and one using NCSA. Compare the two definition files and notice the differences.
2. Again create two different map definition files for the same graphic, this time using all polygon shapes for one of the definitions and all squares for the other definition. Compare the two definition files. Are polygons considerably more complicated than standard shapes?
3. Create a button bar (or menu bar) using a series of clickable graphics. Then, create a similar button bar using an image map. Which takes more work? Which will take more time to download to a browser (i.e., which method takes up more drive space)?

Chapter 13

HTML Forms

CONTENTS

The next set of HTML tags are designed to allow you to enhance the interactivity of your Web pages by increasing your ability to request information from users. Using the forms tags, you can ask users to enter text information, choose from menus, mark checkboxes, make choices from radio buttons, and then send that information to the Web server for processing.

Using Forms and Form-Capable Browsers

Although the forms tags are a part of the HTML 2.0 standard, it's important to recognize that not all browsers are capable of viewing them-especially older browsers and text-based browsers. Users need to have forms-aware browsers, like the current versions of NCSA Mosaic, Netscape Navigator, and Microsoft Internet Explorer, among others. Generally, other browsers will simply ignore the forms commands if they can't deal with them.

Tip
It's a good idea to let your users know that they're about to jump to a form-based page

whenever possible. Forms pages are a waste of time for users of older browsers that don't support them.

The idea behind a Web form is simple-it allows you to accept information or answers from your users with varying levels of guidance. Users can be asked to type answers, choose their answers from a list of possibilities you create, or even be limited to choosing one answer from a number of options that you specify.

That data is then passed on to the Web server, which hands it to a script, or small program, designed to act on the data and (in most cases) create an HTML page in response. In order to deal with forms data then, you need to understand a little something about scripting, or programming, for a Web server-or know someone who does. While learning to program is beyond the scope of this book, we'll look at how these scripts work in Chapter 14, "Form Design and Data Gathering with CGI Scripts."

Note
Most Web server scripts are written in Perl, C, or UNIX shell scripts. If your Web server is DOS, Windows, or Mac based, however, you may have other options. Some DOS Web servers allow you to script in the DOS batch language, while some Windows servers can accept Visual Basic scripts (not to be confused with Microsoft's new Visual Basic Script language). Mac Web servers generally allow for AppleScript or Frontier scripting.

Creating the Form

In an HTML document, forms are set between the `<FORM>` container tags. The form container works as follows:

```
<FORM METHOD="how_to_send" ACTION="URL of script">
...form data...
</FORM>
```

Notice that the `<FORM>` tag takes two attributes: `METHOD` and `ACTION`. The `METHOD` attribute accepts either `POST` or `GET` as its value. `POST` is by far the more popular, as it allows for a greater amount of data to be sent. `GET` is a little easier for Web programmers to deal with, and is best used with single responses, like a single textbox.

The second attribute is `ACTION`, which simply accepts the URL for the script that will process the data from your form. Most often the script is stored in a directory called `bin/` or `cgi-bin/` located on your Web server.

An example of the `<FORM>` tag then, would be the following:

```
<FORM METHOD="SEND" ACTION="http://www.fakecorp.com/cgi-bin/register_script">
</FORM>
```

As with any HTML container tag, this implementation of the `<FORM>` tag has actually created a complete form (just like `<P>` and `</P>` is a complete paragraph). Unfortunately, our complete form doesn't *do* anything yet, so that's somewhat academic.

Note
You can't nest forms within one another. You need to add the end tag `</FORM>` for the first form before creating another one in the same document. Generally, browsers will ignore any new occurrences of the `<FORM>` tag, since the purpose of the tag is to tell the browser how to submit data to the server, and different parts of the form can't be submitted in different ways.

Example: Someone Else's Form

Let's take a quick look at a form that's been created by someone else-one that most seasoned Web browsers have encountered at one time or another. Load up your Web browser and point it to **http://webcrawler.com/**.

This is the WebCrawler page, a Web search engine offered by America Online. Your next step is to view the source of this document. Select the View Document Source command in your Web browser's Edit menu. What you see will look something like figure 13.1.

Note
Nearly all graphical browsers have a View Source command. Look in the Edit menu for this command or a command with a similar name. The HTML source of the current Web document will then be displayed or saved as a text file.

Notice a couple of things here. The `<FORM>` tag at WebCrawler is using the `ACTION` and `METHOD` attributes that were discussed. `ACTION` is accessing a script called `WebQuery` found in the `cgi-bin/` directory of the Web server. The `METHOD` used is `SEND`.

Tip
Although you shouldn't copy others' work, don't forget that you can always use View Source commands to learn how something was done on the Web.

Text Fields and Attributes

One of the more common uses for forms is to accept multiple lines of text from a user, perhaps for feedback, bug reports, or other uses. To do this, use the `<TEXTAREA>` tag within your form. You can set this tag to control the number of rows and columns it displays, although it will generally accept as many characters as the user desires to enter. It takes the following form:

```
<TEXTAREA NAME="variable_name" ROWS="number" COLS="number">
default text
</TEXTAREA>
```

It may surprise you to find that `<TEXTAREA>` is a container tag, since it just puts a box for typing on the page. What's contained in the tag is the default text-so you can guide your users by letting them know what you'd like entered there. For instance:

```
<FORM>
<TEXTAREA NAME="comments" ROWS="4" COLS="40">
Enter comments about this Web site.
```

```
Good or Bad.
</TEXTAREA>
</FORM>
```

The default text appears in the textbox just as typed. Notice in figure 13.2 that text inside the <TEXTAREA> tag works like <PRE> formatted text. Any returns or spaces you add to the text are displayed in the browser window. In fact, notice that by hitting Return after the opening <TEXTAREA> tag, I'm inserting a blank line at the top of the textarea (in many browsers).

The NAME attribute is a variable name for this string of text. It gets passed on to your processing script on the Web server. ROWS and COLS can accept different numbers to change the size of the textarea box, but you should take care that the majority of browsers can see the entire box on-screen. It's best to limit COLS to 80, and ROWS to something like 24 (the typical size for a text-based computer screen). But it's up to you.

<TEXTAREA> will also accept one other attribute: WRAP. WRAP can be set to OFF (which is the default if WRAP is not included), VIRTUAL, or PHYSICAL. Setting wrap to PHYSICAL forces the browser to include actual line breaks in the text when sending it to the Web server. VIRTUAL makes the textbox seem to offer line wrap, but sends a continuous stream of words to the Web server (unless the user has entered returns on his or her own).

Example: Web-based Feedback Form

I mentioned before that <TEXTAREA> is commonly used to gather feedback from your Web users. To create a small form to do just that, save your default template as a new HTML document and enter the following:

```
<BODY>
<H3>Feedback Form</H3>
<P>Please take a moment to tell us what you thought of the Web site.<BR>
Your Feedback is appreciated!</P>
<FORM METHOD="POST" ACTION="cgi-bin/feedback">
Enter your comments below:<BR>
<TEXTAREA NAME="comments" ROWS="10" COLS="70" WRAP="VIRTUAL">
Dear BigCorp:
</TEXTAREA>
</FORM>
</BODY>
```

You can see how this looks in figure 13.3. Notice in the example that some descriptive text is enclosed inside the <FORM> tag, but outside of the <TEXTAREA> tag. This is completely legal-it just lets you explain what the purpose of the textarea is.

You may have realized that there's something lacking in this sample form. There's no way to submit the user's entry! You'll get to that in the next section, when I discuss this next tag for form entry.

The <INPUT> Tag

Our next tag for HTML forms give you the opportunity to be a bit more picky about the type of input you're going to accept from the user. The <INPUT> tag follows the following format:

```
<INPUT TYPE="type_of_box" NAME="variable" SIZE="number" MAXLENGTH="number">
```

Now, technically, the only required attributes are TYPE and NAME. Some other "types" of the input tag will also accept the attribute VALUE. But first, let's look at the different types of <INPUT>.

Note
By the way, notice that <INPUT> is an empty tag. There's no </INPUT> element.

TEXT

The first possible value for the TYPE attribute is TEXT, which creates a single-line textbox of a length you choose. Notice that the length of the box and the maximum length entered by the user can be set separately. It's possible to have a box longer (or, more often, shorter) than the maximum number of characters you allow to be entered. Here's an example of a textbox:

```
Last name: <INPUT TYPE="TEXT" NAME="last_name" SIZE="40" MAXLENGTH="40">
```

When appropriately entered between <FORM> tags, this <INPUT> yields a box similar to figure 13.4. If desired, the attribute VALUE can be used to give the textbox a default entry, as in the following example:

```
Type of Computer: <INPUT TYPE="TEXT" NAME="computer" SIZE="50" MAXLENGTH="50"
VALUE="Pentium">
```

PASSWORD

The PASSWORD option is nearly identical to the TEXT option except that it responds to typed letters with bullet points or a similar scheme (chosen by the browser) to keep the words from being read. A sample password box could be the following:

```
Enter Password: <INPUT TYPE="PASSWORD" NAME="password" SIZE="25" MAXLENGTH="25">
```

When characters are typed into this textbox, they are shown on the screen as in figure 13.5.

Recognize that the text is still stored as the text typed by the user-not as bullet points or similar characters.

CHECKBOX

This value for TYPE allows you to create a checkbox-style interface for your form. This is best used when there are two possible values for a given choice-and no others. You can also determine whether or not a checkbox will already be checked (so that it must be unchecked by the user, if desired), by using the attribute CHECKED. Here's an example of adding checkboxes to a form:

```
Type of computer(s) you own:<BR>
<INPUT TYPE="CHECKBOX" NAME="Pentium" CHECKED> Pentium
<INPUT TYPE="CHECKBOX" NAME="486"> 486-Series PC
<INPUT TYPE="CHECKBOX" NAME="Macintosh"> Macintosh
```

In this example, it's possible to check as many of the options as are presented. CHECKBOX evaluates each item separately from any others. Figure 13.6 illustrates how CHECKBOX is displayed in a browser.

RADIO

Like CHECKBOX, RADIO is designed to offer your user a choice from pre-determined options. Unlike CHECKBOX, however, RADIO is also designed to accept only one response from among its options. RADIO uses the same attributes and basic format as CHECKBOX.

RADIO requires that you use the VALUE attribute, and that the NAME attribute be the same for all of <INPUT> tags that are intended for the same group. VALUE, on the other hand, should be different for each choice. For instance, look at the following example:

```
Choose the computer type you use most often:<BR>
<INPUT TYPE="RADIO" NAME="Computer" VALUE="P" CHECKED> Pentium
<INPUT TYPE="RADIO" NAME="Computer" VALUE="4"> 486-Series PC
<INPUT TYPE="RADIO" NAME="Computer" VALUE="M"> Macintosh
<INPUT TYPE="RADIO" NAME="Computer" VALUE="O"> Other
```

With RADIO, it's important to assign a default value, since it's possible that the user will simply skip the entry altogether. While the user can't check more than one, he or she can check none. So choose the most common value and set it as CHECKED, just so that the form-processing script doesn't have trouble.

Note
Of course, if you want to give your user the option of choosing none, then you can leave off the CHECKED attribute. It's more complete and obvious for the user, however, to include another radio button with a VALUE of none, and make it the CHECKED choice.

HIDDEN

This <INPUT> type technically isn't "input" at all. Rather, it's designed to pass some sort of value along to the Web server and script. It's generally used to send a keyword, validation number, or some other kind of string to the script so that the script knows it's being accessed by a valid (or just a particular) Web page. The <INPUT TYPE="Hidden"> tag takes the attributes NAME and VALUE.

Note
This isn't really terribly covert, since an intrepid user could simply choose View Source to see the value of the hidden field. It's more useful from a programmer's standpoint. For instance, on a large Web site, the hidden value might tell a multi-purpose script which particular form (among many) is sending the data, so the script knows how to process the data.

RESET

The <INPUT> tag has built into it the ability to clear an HTML form. RESET simply creates a push button (named with the VALUE string) that resets all of the elements in that particular FORM to their default values (erasing anything that the user has entered). An example would be the following:

```
<INPUT TYPE="RESET">
```

With a VALUE statement, you could enter the following:

```
<INPUT TYPE="RESET" VALUE="Reset the Form">
```

The results are shown in figure 13.7.

SUBMIT

The <INPUT> tag also has a type that automatically submits the data that's been entered into the HTML form. The SUBMIT type accepts only the attribute VALUE, which can be used to rename the button. Otherwise, the only purpose of the Submit button is to send off all the other form information that's been entered by your user. See the following two examples (see fig. 13.8):

```
<INPUT TYPE="SUBMIT">
<INPUT TYPE="SUBMIT" VALUE="SEND IT IN!">
```

You can use just about anything you want for the VALUE, although it's best to remember that really small words, like *OK*, don't look great as buttons. To make a button larger, enter the VALUE with spaces on either end, like in the following:

```
<INPUT TYPE="SUBMIT" VALUE="       GO      ">
```

Example: A More Complete Form

Along with all the other <INPUT> types, now you've finally got a way to submit data. So, let's create a more involved form that includes some of these examples-a subscription form.

Save your HTML template to create a new document. Then, enter something similar to Listing 13.1.

Listing 13.1 `scrp_frm.html` Creating a Complete Form

```
<BODY>
<H2>Subscribe to CorpWorld</H2>
<P>Interested in receiving daily email updates of all the latest exploits of BigCorp?
Well, now you can. And, best of all, it's free! Just fill out this form and submit it
by clicking the "Send it In" button. We'll put you on our mailing list, and you'll
receive your first email in 3-5 days.</P>
<FORM METHOD="Send" ACTION="http://www.fakecorp.com/cgi-bin/subscribe">
Please complete all of the following:<BR>
First Name: <INPUT TYPE="Text" Name="first" SIZE="25" MAXLENGTH="24"><BR>
Last Name:  <INPUT TYPE="Text" Name="last" SIZE="35" MAXLENGTH="34"><BR>
Business:   <INPUT TYPE="Text" Name="business" SIZE="50" MAXLENGTH="49"><BR>
We must have a correct email address to send you the newsletter:<BR>
Email:      <INPUT TYPE="Text" Name="email" SIZE="50" MAXLENGTH="49"><BR>
How did you hear about BigCorp's email letter?<BR>
<INPUT TYPE="RADIO" NAME="hear" VALUE="web" CHECKED>Here on the Web
<INPUT TYPE="RADIO" NAME="hear" VALUE="mag">In a magazine
<INPUT TYPE="RADIO" NAME="hear" VALUE="paper">Newspaper story
<INPUT TYPE="RADIO" NAME="hear" VALUE="other">Other
<BR> Would you care to be on our regular mailing list?<BR>
<INPUT TYPE="CHECKBOX" NAME="snailmail" CHECKED> Yes, I love junk mail<BR>
<INPUT TYPE="RESET">
<INPUT TYPE="SUBMIT" VALUE="Send it in!">
</FORM>
</BODY>
```

Notice that, for text type <INPUT> boxes, the MAXLENGTH is one less than the size of the box. This tends to look a little better, but choosing the size is up to you. Figure 13.9 shows how it looks on a Web page. (You'll get to straightening everything out and making it look great in Chapter 14.)

Creating Pop-Up and Scrolling Menus

The last types of input that you can offer to users of your Web page revolve around the <SELECT> tag, which can be used to create different types of pop-up and scrolling menus. This is another element designed specifically for allowing users to make a choice-they can't enter their own text. The <SELECT> tag requires a NAME attribute and allows you to decide how many options to display at once with the SIZE attribute.

Using <SELECT>

Also notice that, like <TEXTAREA>, <SELECT> is a container tag. Options are placeed between the two <SELECT> tags, each with a particular VALUE that gets associated with <SELECT>'s NAME attribute when chosen. The following is the basic format:

```
<SELECT NAME="variable">
<OPTION SELECTED VALUE="value"> Menu text
<OPTION VALUE="value"> Menu text
...
</SELECT>
```

The attribute SELECTED is simply designed to show which value will be the default in the menu listing. *value can be anything you want to pass on to the Web server and associated script for processing. An example might be:*

```
Choose your favorite food:
<SELECT NAME="food">
<OPTION SELECTED VALUE="ital"> Italian
<OPTION VALUE="texm"> TexMex
<OPTION VALUE="stek"> SteakHouse
<OPTION VALUE="chin"> Chinese
</SELECT>
```

You can also use the SIZE attribute to decide to display the menu in its entirety, by simply changing the first line of the example to the following:

```
<SELECT NAME="food" SIZE="4">
```

Both examples are shown in figure 13.10.

In the first example, selecting the menu item with the mouse causes the menu to pop-up on the page. The user can then select from the choices. In the second example, it's necessary to click the desired item.

Allowing More than One Selection

One more attribute for the <SELECT> tag allows the user to select more than one option from the menu. Using the MULTIPLE attribute forces the menu to display in its entirety, regardless of the SIZE attribute. An example might be the following: (the result appears in figure 13.11):

```
What type of cars does your family own (select as many as apply)?
<SELECT NAME="cars" MULTIPLE>
<OPTION VALUE="sedan"> Sedan
<OPTION VALUE="coupe"> Coupe
<OPTION VALUE="mivan"> Minivan
<OPTION VALUE="covan"> Conversion Van
<OPTION VALUE="statn"> Stationwagon
<OPTION VALUE="sport"> SUV (4x4)
<OPTION VALUE="truck"> Other Truck
</SELECT>
```

Example: Order Form

With all of these possibilities for the form, you can manage some fairly complete data entry interfaces for users. Consider this one: an online order form. Used in conjunction with a secure Web site, this form could be used to process purchase orders over the Internet.

Save your template as a new HTML file and enter Listing 13.2's example text between the `<BODY>` *tags.*

Listing 13.2 `ordr_frm.html` Creating an Order Form

```
<BODY>
<H3>Online Order Form</H3>
<P> Please enter your name, billing address and shipping address. Please
don't forget the order number from our online catalog listings. Thanks for shopping
BigCorp!</P>
<FORM METHOD="SEND" ACTION="http://www.fakecorp.com/cgi-bin/order">
<HR>
Please enter a full name and address for BILLING purposes:<BR>
First Name: <INPUT TYPE="TEXT" NAME="first" SIZE="25" MAXLENGTH="24"><BR>
Last Name: <INPUT TYPE="TEXT" NAME="last" SIZE="35" MAXLENGTH="34"><BR>
Address: <INPUT TYPE="TEXT" NAME="address" SIZE="60" MAXLENGTH="59"><BR>
City: <INPUT TYPE="TEXT" NAME="city" SIZE="25" MAXLENGTH="24">
State: <INPUT TYPE="TEXT" NAME="state" SIZE="3" MAXLENGTH="2"> ZIP:
<INPUT TYPE="TEXT" NAME="zip" SIZE="6" MAXLENGTH="5"><BR>
<HR>
<INPUT TYPE="CHECKBOX" NAME="same_add"> Check if Shipping Address is
different from Mailing Address
<HR>
<TEXTAREA NAME="ship_add" ROWS="3" COLS="60" WRAP="PHYSICAL">Enter shipping address
here if different from above.
</TEXTAREA>
<HR>
Please enter the code for the product you wish to purchase: <INPUT TYPE=
"TEXT" NAME="prod_num" SIZE="7" MAXLENGTH="6">
<HR>
How would you like to pay for this?<BR>
<SELECT NAME="credit">
<OPTION SELECTED VALUE="mast"> MasterCard
<OPTION VALUE="visa"> Visa
<OPTION VALUE="amex"> American Express
<OPTION VALUE="disc"> Discover
</SELECT>
Please enter the card number: <INPUT TYPE="TEXT" NAME="cred_num" SIZE="17"
MAXLENGTH="16"><BR>
Expiration date (01/99): <INPUT TYPE="TEXT" NAME="exp_date" SIZE="6"
MAXLENGTH="5"><HR>
<BR>
```

```
Please take care that everything is filled out correctly, then click "Submit Order."
If you'd like, you can select the "Reset" button to start again. Clicking the "Submit
Order" button will send your order to BigCorp and your credit card will be
charged.<BR>
<INPUT TYPE="reset">
<BR>
<INPUT TYPE="submit" VALUE="Submit Order">
</FORM>
</BODY>
```

Here you've taken advantage of most of the options available to you for forms (see fig. 13.12). Notice that, if the checkbox for Check if Shipping Address is different from Mailing Address is left unchecked, you can assume (in the processing script) that the textarea can be ignored. Also notice how using the `MAXLENGTH` attribute for `State:` and `ZIP:` allows you some very basic error checking in these fields. At least, you know that users are entering the correct number of characters.

Once the user clicks the Submit Order button, the script on your Web server takes over. The script should be designed to accept the data, add it to your internal order-processing database (if appropriate), and respond to the submission with an HTML page confirming the order and offering any additional help or instructions. Then, hopefully, the product will ship on time!

Review Questions

1. *What are the two values for the `<FORM>` attribute `METHOD`? Which are you more likely to use?*
2. *What does the `ACTION` attribute accept?*
3. *What is a `<TEXTAREA>` form element used for? How does the user enter data?*
4. *`<TEXTAREA>` is a container tag. What does it contain?*
5. *Why sort of element is `TYPE` as it relates to the `<INPUT>` tag?*
6. *Aside from how they look, what's the major difference between checkboxes and radio buttons?*
7. *How do you define a checkbox or radio button as the default value?*
8. *How do you tell an HTML form to send its data to the Web server?*
9. *What type of interface element does the `<SELECT>` tag display?*
10. *If you use the attribute `MULTIPLE` with the `<SELECT>` tag, what happens to the way the menu displays?*
11. *How do you define the default value in a `<SELECT>` menu?*

Review Exercises

1. *Create a simple form that lets your user send you an e-mail message. (Hint: you can use the `mailto:` type of URL to actually cause the form to mail the form data to your e-mail account.)*
2. *Create a form that offers the following choices in a pop-up menu, a series of radio buttons, and a list of checkboxes. Make a different value the default in each. The choices are: North, South, East, and West.*
3. *Using a Select menu, create two different menus of the following items. Make one a pop-up menu and the other a scrolling menu. The choices are: Life, Liberty, Happiness, Death, and Taxes.*

Chapter 15

Adding Tables to Your Documents

CONTENTS

Many chapters ago you learned to use the <PRE> tag to create preformatted tables that align your data and text for easy reading. The HTML 3.0 table specification, however, takes you far beyond that. Tables are a great addition to any Web site-especially sites that need to offer a lot of information in an easy-to-read way. Unfortunately, tables can't be viewed by all browsers. So, you need to proceed with a little caution and consideration.

Note
The current HTML 3.0 tables standard, in its entirety, isn't viewable by many browsers. In this chapter, everything discussed is part of the HTML 3.0 standard-but something less than the entire thing. Creating tables in this way will make your tables viewable in the widest number of browsers.

Creating a Table

Tables work a lot like HTML list tags, in that you must use the table container tag to hold together a group of tags that define each individual row. The main container is the <TABLE> tag, which uses enclosing tags for table rows (<TR>) and table data (<TD>). Most tables will also use an element for table headers (<TH>) which is generally used as the title text for rows and columns.

Tables take the following format:

```
<TABLE>
<CAPTION>Caption text for table</CAPTION>
<TR><TH>column1</TH><TH>column2</TH><TH>column3</TH>
<TR><TD>row1data1</TD><TD>row1data2</TD><TD>row1data3</TD>
```

```
<TR><TD>row2data1</TD><TD>row2data2</TD><TD>row2data3</TD>
...
</TABLE>
```

An example of a table using this format might be the following:

```
<TABLE>
<CAPTION>Team Members for 3-Person Basketball</CAPTION>
<TR><TH>Blue Team</TH><TH>Red Team</TH><TH>Green Team</TH>
<TR><TD>Mike R.</TD><TD>Leslie M.</TD><TD>Rick G.</TD>
<TR><TD>Julie M.</TD><TD>Walter R.</TD><TD>Dale W.</TD>
<TR><TD>Bill H.</TD><TD>Jenny Q.</TD><TD>Fred B.</TD>
</TABLE>
```

After you work with HTML list containers, it's fairly easy to make the jump to creating tables in HTML. You can see how this table looks in figure 15.1.

The <TABLE> Tag

The <TABLE> tag is actually a rather complex creature, at least insofar as it can accept many different attributes. Some of the attributes are more useful than others, so let's look at the most useful of them as they currently stand:

- **ALIGN**. The ALIGN attribute is used to determine where the chart will appear relative to the browser window. Valid values are ALIGN=LEFT and ALIGN=RIGHT. As an added bonus, text will wrap around the table (if it's narrow enough) when the ALIGN=LEFT or ALIGN=RIGHT attributes are used.
- **WIDTH**. The WIDTH attribute sets the relative or absolute width of your table in the browser window. Values can be either percentages, as in WIDTH="50%", or absolute values. With absolute values, you must also include a suffix that defines the units used, as in px for pixels or in for inches (e.g., WIDTH="3.5in"). Absolute values for table widths are discouraged, though.
- **COLS**. The COLS attribute specifies the number of columns in your table, allowing the browser to draw the table as it downloads.
- **BORDER**. The BORDER attribute defines the width of the border surrounding the table. Default value is 1 (pixel).
- **CELLSPACING**. The CELLSPACING attribute tells the browser how much space to include between the walls of the table and between individual cells. (Value is a number in pixels.)
- **CELLPADDING**. The CELLPADDING attribute tells the browser how much space to give data elements away from the walls of the cell. (Value is a number in pixels.)

It is definitely not necessary to use all of these attributes for your table-in fact, the simple table example earlier didn't use any of them. Often, however, they will come in handy.

Example: Playing with Table Attributes

This is another fairly freeform example. Let's look at the difference between a plain table and a table embellished with a few attributes. Insert Listing 15.1 in a new HTML document.

Listing 15.1 `badtable.html` Creating a Plain Table

```
<BODY>
<H2> BigCorp's Computer Systems </H2>
```

```
<P>We use only the highest quality components and software for all of our
Wintel computer systems. Plus, if you don't see a configuration you like,
call (or email) and let us know. We'll custom build to please!</P>
<TABLE>
<CAPTION>BigCorp's Computer Systems and Specifications</CAPTION>
<TR><TH>System 486</TH><TH>System 586</TH><TH>System 686</TH>
<TR><TD>486DX2-66 CPU</TD><TD>120 MHZ AMD586</TD><TD>200 Mhz Pentium Pro</TD>
<TR><TD>8 MB RAM</TD><TD>16 MB RAM</TD><TD>16 MB RAM</TD>
<TR><TD>500 MB HD</TD><TD>1 GB HD</TD><TD>1.4 GB HD</TD>
<TR><TD>14.4 Modem</TD><TD>28.8 Modem</TD><TD>28.8 Modem</TD>
<TR><TD>desktop case</TD><TD>minitower case</TD><TD>tower case</TD>
<TR><TD>DOS/Win 3.1</TD><TD>Windows 95</TD><TD>Windows NT 4.0</TD>
</TABLE>
</BODY>
```

Now, take a quick glance at how this looks in a browser (see fig. 15.2).

Last time we tried a simple table, it communicated its data well. But this one is fairly ineffective, with everything lined up so poorly. Using just the attributes only mentioned, though, you can change this table so that it looks better to the user and is easier to read.

All that needs to change is the first `<TABLE>` tag:

```
<TABLE BORDER ALIGN="LEFT" CELLSPACING="3" CELLPADDING="3">
```

That makes for a much nicer looking table, complete with borders and lines for cells, and a comfortable amount of spacing to separate cell data elements from one another (see fig. 15.3).

The rest of this example is up to you. Play with `CELLSPACING` and `CELLPADDING` without a border, for instance, or increase all values out of proportion. See the range of what's available, to help you choose how to format your tables in the future.

Captions, Table Headers, and Table Data

To round out your tables, you have the other basic tags to examine. You've already successfully used `<CAPTION>`, `<TH>`, and `<TD>`, but each has its own attributes and abilities that you need to know about.

<CAPTION>

The `<CAPTION>` tag is a container for reasons that may be obvious-it allows you to nest other HTML tags within the description. For instance:

```
<CAPTION><B>Table 3.1 from the book <I>Life in El Salvador</I></B></CAPTION>
```

Just about any sort of markup tags are possible inside the `<CAPTION>` tags, although some-like list tags-wouldn't make much sense.

The `<CAPTION>` tag has one attribute, `ALIGN`. `ALIGN="TOP"` and `ALIGN="BOTTOM"` are encouraged. By default, text is also aligned to center (horizontally). By `TOP` and `BOTTOM`, I'm referring to the entire table; the caption will default to the top of the table if not otherwise specified. To align the caption to `BOTTOM`, for instance, enter the following:

```
<CAPTION ALIGN="BOTTOM">Table of Common Foods</CAPTION>
```

The <CAPTION> tag is commonly the first tag just inside the <TABLE> tag (although this placement is not required). Regardless of where you place the <CAPTION> tag, however, you must use ALIGN to force it to the bottom of the table. Otherwise, it will appear at the top, according to its default.

Let's create an entire table and use the ALIGN attribute to the <CAPTION> tag to force the caption to the bottom, like this:

```
<BODY>
<H3>Favorite Ice Cream Flavors</H2>
<TABLE BORDER>
<CAPTION ALIGN="BOTTOM">Data from the <I>New Jersey Times</I></CAPTION>
<TR><TH>Date</TH><TH>Chocolate</TH><TH>Vanilla</TH>
<TR><TH>1970</TH><TD>50%</TD><TD>50%</TD>
<TR><TH>1980</TH><TD>76%</TD><TD>24%</TD>
<TR><TH>1990</TH><TD>40%</TD><TD>60%</TD>
</TABLE>
</BODY>
```

When the browser interprets this table, it should place the caption at the bottom of the table, centered horizontally (see fig. 15.4).

Table Rows

Table rows (<TR>) can accept one attribute you should concern yourself with-ALIGN. The ALIGN attribute is used to determine how text will appear (horizontally) in each of the rows data cells. For instance:

```
<TR ALIGN="CENTER"><TH>Date</TH><TH>Chocolate</TH><TH>Vanilla</TH>
<TR ALIGN="CENTER"><TH>1970</TH><TD>50%</TD><TD>50%</TD>
<TR ALIGN="CENTER"><TH>1980</TH><TD>76%</TD><TD>24%</TD>
<TR ALIGN="CENTER"><TH>1990</TH><TD>40%</TD><TD>60%</TD>
```

Here, I've added the ALIGN attribute (with a value of CENTER) to the rows in the previous example. Notice now that all cells center data horizontally (see fig. 15.5). This ALIGN attribute can also accept LEFT and RIGHT.

Note
HTML 3.0 also supports another useful attribute, VALIGN, which accepts the values TOP, BOTTOM, and CENTER. Using this attribute, you can choose to align cells vertically as well as horizontally. Until they support VALIGN, non-HTML 3.0 browsers should ignore VALIGN. Unfortunately, those are currently the most popular browsers!

Table Data and Rows

You've already used the <TH> and <TD> tags to include headers and data in your tables. You may have noticed that, essentially, the only difference between the two is that <TH> emphasizes (boldfaces) the text and <TD> does not. Now, technically, the <TH> is a tag that the browser interprets as a header and thus displays text in a way that's distinct from the <TD> tag. In practice, that generally means it's turned bold.

Aside from accepting nearly any type of HTML markup tags within them, both tags can accept four attributes (in most HTML versions). These are ALIGN, VALIGN, COLSPAN, and ROWSPAN. If you were to add all of these attributes, a typical <TH> (or <TD>) tag would be formatted like the following:

```
<TH ALIGN="direction" VALIGN="direction" COLSPAN="number" ROWSPAN="italics">
```

ALIGN is used to align the data within the cell horizontally, accepting values of LEFT, RIGHT, and CENTER. Note that ALIGN is redundant when used with the ALIGN attribute of <TR>, unless it is used to override the <TR ALIGN=> setting.

VALIGN is used to align the data vertically within cells. Possible values are TOP, BOTTOM, and CENTER. COLSPAN and ROWSPAN are used to force a cell to span more than one column or row, respectively. An example of this might be:

```
<TABLE BORDER>
<TR><TH>Student</TH><TH>Test 1</TH><TH>Test 2</TH><TH>Average</TH>
<TR><TH>Mike M.</TH><TD>100</TD><TD>75</TD><TD ROWSPAN="3">N/A</TD>
<TR><TH>Susan T.</TH><TD>80</TD><TD>95</TD>
<TR><TH>Bill Y.</TH><TD COLSPAN="2">Dropped Course</TD>
</TABLE>
```

Viewed in a browser, the table looks like figure 15.6.

Example: An Events Calendar

One interesting way to use a table is to create a calendar, which is possible with what we now know about attributes for tables and table elements. Let's create a calendar for November 1996. We'll also throw in some hypertext links that would (presumably) be used to discuss events planned for those days. Enter Listing 15.2 in a new HTML document.

Listing 15.2 `calendar.html` Using HTML Tables to Create a Calendar

```
<BODY>
<H2>Coming Events</H2>
<P>Click any of the days highlighted in the calendar to read about the event scheduled
for that day.</P>
<TABLE BORDER WIDTH="75%">
<CAPTION>BigCorp's Calendar of Events - November 1996</CAPTION>
<TR ALIGN="CENTER"><TH>Sun</TH><TH>Mon</TH><TH>Tue</TH><TH>Wed</TH><TH>Thu</TH>
<TH>Fri</TH><TH>Sat</TH>
<TR ALIGN="CENTER"><TD COLSPAN="5"> </TD><TD>1</TD><TD>2</TD>
<TR ALIGN="CENTER"><TD>3</TD><TD>4</TD><TD>5</TD><TD>6</TD><TD>7</TD><TD>8</TD>
<TD>9</TD>
<TR ALIGN="CENTER"><TD>10</TD><TD><A
HREF="nov11.html">11</A></TD><TD>12</TD><TD>13</TD><TD><A
HREF="nov14.html">14</A></TD><TD>15</TD><TD>16</TD>
<TR ALIGN="CENTER"><TD><A HREF="nov17.html">17</A></TD><TD>18</TD><TD>19</TD>
<TD><A HREF="nov20.html">20</A></TD><TD>21</TD><TD>22</TD><TD>23</TD>
<TR ALIGN="CENTER"><TD>24</TD><TD>25</TD><TD>26</TD><TD>28</TD><TD>29</TD><TD>
30</TD><TD>31</TD>
</TABLE>
</BODY>
```

Notice the   in the <TD> tag that is defined with COLSPAN? That is an escape sequence for Web browsers that tells it "I want a non line-breaking space here." Without that, the extra-long cell won't be rendered correctly (with a complete border) because there's nothing in that cell. With it, this table looks like a calendar (see fig. 15.7).

Example: Product Specifications

One thing that hasn't really been touched on so far is the possibility of including images in tables. It's definitely possible, and just about as easy as anything else you've done with tables.

In this example, let's create a product specifications table for a couple of our company's computer systems. With liberal use of the ALIGN and VALIGN attributes, this should come out looking rather pretty. Insert Listing 15.3 in a new HTML document.

Listing 15.3 `aligntbl.html` Using `ALIGN` and `VALIGN` with Images in an HTML Table

```
<BODY>
<H2>Product Specifications</H2>
<P>The following table will tell you a little more about our computer
systems. Clicking on the name of each system will tell you even more,
offering a full-size photo of the system and some suggestions on
peripherals.</P>
<HR>
<TABLE BORDER CELLSPACING="2" CELLPADDING="2">
<CAPTION>Our System Configurations</CAPTION>
<TR ALIGN="CENTER"><TH>Photo</TH><TH>Name</TH><TH>RAM</TH><TH>Hard
Drive</TH><TH>Video</TH><TH>Expansion</TH><TH>Case</TH>
<TR ALIGN="CENTER"><TD><IMG SRC="sml_6001.GIF"></TD><TD><A HREF="6001.html">
System 6001-60</A></TD><TD>8 MB</TD><TD>500 MB</TD><TD>1 MB PCI</TD><TD>4 PCI
Slots</TD><TD ROWSPAN="2">Desktop</TD>
<TR ALIGN="CENTER"><TD><IMG SRC="sml_7001.GIF"></TD><TD><A HREF="7001.html">
System 7001-75</A></TD><TD>16 MB</TD><TD>1.0 GB</TD><TD>1 MB PCI
</TD><TD>5 PCI
Slots</TD>
<TR ALIGN="CENTER"><TD><IMG SRC="sml_8001.GIF"></TD><TD><A HREF="8001.html">
System 8001-120</A></TD><TD>20 MB</TD><TD>1.6 GB</TD><TD>2 MB PCI
</TD><TD>5 PCI
Slots</TD><TD>Tower</TD>
</TABLE>
</BODY>
```

Graphics look very nice in tables, and they work well to enliven what would otherwise be drier, text-heavy information (like computer specs). I've offered up some creative uses of attributes in this example, but I think it was worth it (see fig. 15.8).

Review Questions

1. Why doesn't this chapter discuss the entire HTML 3.0 tables standard?
2. Does the ALIGN attribute for <TABLE> allow text to wrap around the table?
3. What does the in stand for in the attribute definition WIDTH="3.5in" for the <TABLE> tag?
4. What's the different between the attributes CELLPADDING and CELLSPACING?
5. True or false. You must always define a value for the BORDER attribute to the <TABLE> tag.

6. If I had tmhe following example:

```
<TABLE>
<TR><TH>Soup</TH><TD>Chicken Noodle</TD>
<TR><TH>Salad</TH><TD>Tossed Green</TD>
<CAPTION>My favorite foods</CAPTION>
</TABLE>
```

   where would the <CAPTION> text appear relative to the table?
7. Is it possible to ALIGN all of the data cells in a particular row with the <TR> tag?
8. What happens in the following example?

```
<TD>Ted David<BR>Mike Rogers<BR>Bill Howell</TD>
```

9. Which is used for horizontal alignment when used as an attribute to the <TD> tag, ALIGN or VALIGN?
10. What possible reason could there be to force a <TD ALIGN> tag definition to override a <TR ALIGN> tag?

Review Exercises

1. Create a caption, aligned to the bottom of the table, that includes an image. Does it work correctly?
2. Create a table that uses images as the column headers.
3. Create a table of "thumbnail" images, with an small image, description of the image, and the image's filename in each row. Make each image clickable, so that a larger image appears (on a new page) when the user clicks the thumbnail.
4. Create a table with no visible border (BORDER="0"). With this table, it's possible to lay out very intricate pages, with text and graphics aligned to the left or right of the page. Use the table to place a paragraph of text on the left side of the page and three clickable graphics on the right side. (Hint: Use ROWSPAN on the paragraph's cell.)