

After reading these chapter, you'll be able to create your own Web pages by hand-coding individual HTML tags. This chapter gives you an overview of many important HTML features.

Web pages are stored in plain text files with the HTM or HTML file extension. You can easily create or read an HTML file with any text editor -- Notepad, for example. An HTML file is a combination of the content you're publishing and the specialized formatting you apply to the content. Everything you see on a Web page, including the special formatting, is represented as text within the HTML file.

Understanding HTML Markup

HTML is a language that specifies how a document looks. Call it a tag or markup language. You change the format of a document by adding a tag to it that specifies exactly how you want the text to look. HTML tags are phrases that begin with an opening bracket (<) and end with a closing bracket (>). For example, the tag that makes text italic is the <I> tag. The tag that makes text bold is the tag. Think of a tag this way: A tag is nothing more than a command you give the Web browser that it will perform before it continues looking at the rest of the HTML file.

Container Tags

Both the <I> and tags are containers because they enclose their contents; you mark the beginning of the content with a tag like and the ending of the content with the corresponding closing tag . Note

that the closing tag begins with the forward slash (/); otherwise, it looks the same as the opening tag. You can insert text within a container, as well as additional HTML tags. To make this container concept clearer, take a look at the following example, which shows you a sentence that contains a bold phrase. The `` tag turns on bold characters and the closing `` turns off bold characters. Within the bold phrase, the `<I>` tag turns on italic and the closing `</I>` tag turns italic off.

```
This <B>sentence <I>contains</I> a bolded</B> phrase and an italicized word.
```

Other HTML tags are not containers. They're tags that just insert a special element into the Web page or perform a specialized function within the Web page. For example, you use the `` tag to add an image to a Web page or the `
` to add a line break. In either case, you don't use a closing tag to mark the end of the block.

HTML Attributes

Many HTML tags also accept parameters called attributes. You use an attribute to provide additional information to the Web browser about how you want to apply a tag. If you add a tag that creates a link, for example, you also use attributes to specify the URL of the Web page to which you're linking. Attributes are names to which you assign a value by using the equal sign (=). For example, to create a link to an HTML file, you assign the URL of the HTML file to the `HREF` attribute: `HREF=example.htm`. You put this assignment within the opening and closing brackets of the `<A>` tag:

```
<A HREF=example.htm>
```

TIP: Anytime that a value contains spaces, you must enclose the value within quotation marks (single or double). For example, ``.

Organizing an HTML Document

Every HTML file follows the same basic structure. Each file begins with the `<HTML>` tag and ends with the closing `</HTML>` tag. Within the `<HTML>` container, you have two additional containers: `<HEAD>` and `<BODY>`. The `<HEAD>` container specifies title information for the Web

page, and <BODY> contains the actual contents of the Web page. Within the <HEAD> container, you find one more container that specifies the title of the Web page as the user sees it in the Web browser's title bar: <TITLE>.

Below you have an example of a complete HTML file. In fact, you can save this listing to an HTML file on your computer and then use it as a template. Note that the contents of each container are indented a few spaces so that you can easily see the structure of the document. Usually, you put an entire container on a single line when it's a simple container like <TITLE> and when it's brief.

```
<HTML>
<HEAD>
  <TITLE>An Example of a Complete HTML File</TITLE>
</HEAD>
<BODY>
  The actual content of the Web page goes in this container.
</BODY>
</HTML>
```

Applying Character Formats Using HTML

Character formatting includes tags like , which bolds text, and <I>, which italicizes text. The table shows you some of the character-formatting tags supported by HTML. Every tag in this table is a container, which means that you must use the corresponding closing tag to turn off that bit of formatting. The following sections show you more specific examples of how to use these tags.

Some of the tags shown are *physical tags*. These are tags that state exactly how to format a bit of text. formats text as bold, for example. , <I>, <S>, <SMALL>, <STRIKE>, <SUB>, <SUP>, and <U> are also physical formatting tags. *Logical tags* don't tell the browser how to draw the text; instead, they tell the browser what the text represents and leaves the actual formatting decisions up to the Web browser. The remaining tags, such as and , are logical formatting tags. You should stick with the logical tags whenever possible to make sure that every Web browser displays your text as well as possible.

This tag	Makes the text...
	Bold

	Emphasized (usually italic)
	Use a specific style, size, color
<I>	Italic
<PRE>	Fixed width
	Emphasized (usually bold)
<SUB>	Subscript
<SUP>	Superscript
<U>	Underlined

Changing the Style of Text

When using a word processor to format text, you highlight the text and click the Bold button to make it bold or click the Italic button to make it italic. Unless you're using an HTML editor, you don't have any such buttons; instead, you use the character formatting tags described in the preceding section. You put the beginning and ending formatting tags around the text that you would have highlighted in your word processor:

This sentence contains a `bolded` word.

The listing below shows you a complete HTML file that uses a handful of formatting. Note that this example uses logical tags so that the browser can make the decisions regarding how to display the text. `` usually corresponds to `<I>` and `` usually corresponds to ``. `<U>` doesn't have a corresponding logical tag.

```
<html>
  <head>
    <title>Font Style Examples</title>
  </head>
  <body>
```

```
    This example shows you how to change the font style of block
    of text.
```

```
    <strong>This text is bold</strong>
```

```
    <em>This text is italicized</em>
```

```
    <u>This text is underlined</u>
```

```
    You can also nest different styles. For example, <i>you can
    bold text within a block of <b>italicized</b> text.</i>
  </body>
</html>
```

Using a Specific Font Face

You don't know for sure what font a Web browser is going to use when displaying text. You specify an exact font, though. You can also specify the size and color of the font that the Web browser uses.

You use the tag in conjunction with the COLOR, FACE, and SIZE attributes. You can specify any combination of those attributes that you like. If you just specify the COLOR attribute, for example, the Web browser just changes the color of the text without changing the font face or size. COLOR can be a color name or value (see sidebar). FACE can be a list of one or more font names, each separated by commas. If the browser doesn't find the first font, it tries the second, and so on. SIZE is the relative size of the text. +1 would indicate one size larger, +2 would indicate two sizes larger, and -1 would indicate one size smaller.

NOTE: If you specify a particular font by name when using the tag, the font must exist on the user's computer and have the same name. Otherwise, the browser tries alternative fonts in the FACE attribute or uses a default font.

Below you have an example of a Web page that uses the tag to make a few different changes to the text. The first tag changes the face to Arial. The second tag makes the enclosed text four sizes bigger, and the last tag changes the color, size, and face of the text. Note the last example where the FACE attribute specifies two different font faces separated by a comma.

```
<html>
  <head>
    <title>Font Style Examples</title>
  </head>
  <body>

    <font face="Arial">
      This example shows you how to change the font of block
      of text.
    </font>

    <font size="+4">
      This example shows you how to change the size of block
      of text.
    </font>

    <font color=RED size="2" face="Arial,Courier">
      This example shows you how to change the font and size of
      block
```

```
    of text.  
  </font>  
  
</body>  
</html>
```

Colors in HTML

Many HTML tags enable you to set a color attribute. The easiest way to assign a color is by using a color name. There are sixteen predefined color names:

AQUA BLACK BLUE FUCHSIA

GRAY GREEN LIME MAROON

NAVY OLIVE PURPLE RED

SILVER TEAL WHITE YELLOW

Applying Paragraph Formats Using HTML

Paragraph formatting includes tags like <P>, which marks a normal paragraph, <H1>, which marks a level one heading, and , which marks a list item. The table lists most of the paragraph formatting tags that HTML supports. All the tags in this table are containers, except for , so you must end the paragraph with the corresponding closing tag.

This tag	Formats the paragraph as...
<BLOCKQUOTE>	A block quote
<CITE>	A citation
<H1>, <H2>... <H6>	A heading (levels 1 to 6)
	An entry in a list
	An ordered list (use with)
<P>	A normal paragraph
	An unordered list (use with)

Creating normal paragraphs

In a word processor, you press Enter to create a new paragraph. The paragraph is a complete unit to which you can apply a variety of formats. In a Web browser, a paragraph has a blank line between it and the paragraphs before and after it. You mark a paragraph in HTML using the <P> tag. The <P> marks the beginning of the paragraph and the </P> marks the end of the paragraph.

```
<html>
  <head>
    <title>Creating Normal Paragraphs</title>
  </head>

  <body>

    <p>This is the first paragraph in the Web page.</p>
    <p>This is the second paragraph in the Web page.</p>
  </body>
</html>
```

CAUTION: The <P> tag doesn't work like the Enter key in your word processor. Don't try to add extra lines between paragraphs using it. Add line breaks using the
 tag, instead.

Organizing Your Web Page with Headings

HTML provides six levels of headings. You use the <H1> through <H6> tags to mark the beginning and end of a heading. The first heading, Hobbies, uses the <H1> tag. The next three headings use the <H2> tag to create a second-level heading. The last four headings use the <H3> tag to create third-level headings. Each heading level displays using a different font size.

```
<html>
  <head>
    <title>Organizing Your Web Page with Headings</title>
  </head>
  <body>

    <h1>Hobbies</h1>
    <p>The above is a first level heading</p>

    <h2>Photography</h2>
    <p>The above is a second level heading</p>
```

```
<h2>Golf</h2>
<p>The above is a second level heading</p>

<h2>Travel</h2>
<p>The above is a second level heading</p>

<h3>Brazil</h3>
<p>The above is a third level heading</p>

<h3>England</h3>
<p>The above is a third level heading</p>

<h3>Scotland</h3>
<p>The above is a third level heading</p>

</body>
</html>
```

Adding Bulleted and Numbered Lists

HTML supports two different kinds of lists: bulleted and numbered. Bulleted lists are unordered because there is no apparent order to the information they contain. It's just a random collection of thoughts. To create a bulleted list, you use the tag to mark the beginning and end of the list. Then, you add an tag for each item you want to put in the list.

```
<html>
  <head>
    <title>Adding Bulleted Lists to Your Web Page</title>
  </head>
  <body>

    My favorite travel destinations include:

    <ul>
      <li>Brazil</li>
      <li>England</li>
      <li>Scotland</li>
      <li>Singapore</li>
    </ul>

  </body>
</html>
```

TIP: To change the bullet that the browser uses, you can use the tag's TYPE attribute. Assign CIRCLE, DISC, or SQUARE to it. This attribute does not behave the same across all Web browsers, though.

Numbered lists are ordered because they show a numbered sequence. You might use a numbered list to present instructions or to present a list of ideas in order of priority. You create a numbered list using the tag. You then add items to the list using the tag, as you did in the preceding example.

```
<html>
  <head>
    <title>Adding Numbered Lists to Your Web Page</title>
  </head>
  <body>

    My favorite travel destinations include:

    <ol>
      <li>Brazil</li>
      <li>England</li>
      <li>Scotland</li>
      <li>Singapore</li>
    </ol>

  </body>
</html>
```

You can change how the Web browser numbers an ordered list using the START and TYPE attributes. Assign the beginning number to START. For example, to begin numbering the list at 5, assign 5 to START: START=5. You can also change the type of number that the Web browser uses for each list item by assigning one of the codes shown in the table to TYPE.

Code	Description
A	Uppercase letters (A, B, C,...)
a	Lowercase letters (a, b, c,...)
I	Uppercase Roman numerals (I, II, III, ...)
i	Lowercase Roman numerals (i, ii, iii,...)
1	Numbers (1, 2, 3,...)

Preformatting Text on the Web Page

Web browsers automatically condense all spaces and linefeeds. For example, if you insert three spaces between two words, the browser reduces that down to one space. If you insert three blank lines between two words, the browser still reduces that to a single space. The reason should be obvious enough: The Web browser formats the text to fit in its window. It breaks lines of text at the margins, for example. That way, you can slap the text into the HTML file without worrying about how it will be formatted in the browser window.

There are times when you want complete control over how the text appears on the page, however. In particular, you might want to lay out a form in a particular way. You might be quoting a poem and want to lay it out to better reflect each verse. Anything you put in a <PRE> container is safe from the Web browser's tinkering. It'll display that text just as it finds it in the HTML file, including spaces, line feeds, tabs.

```
<html>
  <head>
    <title>Preformatting Text on the Web Page</title>
  </head>
  <body>
```

```
    The following text is preformatted so that you control its
    exact placement on the Web page:
```

```
    <pre>Beginning of the line                browser
keeps tabs
```

```
    The browser also will keep spaces and line breaks
    so that you can line up text evenly on the Web page.
```

```
    Great for quotes,
forms, etc.
    </pre>
  </body>
</html>
```

Inserting Images into a Web Page

Web browsers display inline images side by side with the surrounding text. The image enhances the text and vice-versa. You use the tag to insert an image in a particular location of your Web page. The only required attribute is the SRC attribute to which you assign the URL of the image file. You should also use the WIDTH and HEIGHT attributes,

so that the browser knows how much space to set aside for the image. Doing so allows the reader to see the Web page much faster.

```
<html>
  <head>
    <title>Inserting Images into a Web Page</title>
  </head>

  <body>

    

  </body>
</html>
```

CAUTION: Specifying a height and width that is smaller than the size of the image doesn't save download time. The browser still downloads the full image, and then scales it to the height and width.

HTML gives you a bit of control over how the image is positioned on the Web page. First, you can control how the browser aligns the image with the surrounding text by assigning TOP, MIDDLE, or BOTTOM to the tag's ALIGN attribute. You can also cause the image to float to the left or right side of the Web page, which causes the text to wrap around the image, by assigning LEFT or RIGHT to the ALIGN attribute.

Linking Text and Images to Other Web Pages

You use the <A> tag to link words or images to other documents on the Internet. The <A> tag is a container that marks the beginning and end of the anchor. When the user clicks the anchor, the Web browser opens the Web page referred to by the <A> tag's HREF attribute.

The listing shows you an example of a Web page that contains text and graphical links. In the first case, the <A> tag surrounds a bit of text and its HREF attribute points to my Web page. When the user clicks "Wilfrid Laurier University homepage," the Web browser opens the URL it finds in HREF. In the second case, the <A> tag surrounds an image that's

inserted using the tag. When the user clicks the image, the browser opens the URL it finds in the <A> tag.

```
<html>
  <head>
    <title>Linking Text and Images to Other Web Pages</title>
  </head>

  <body>

    <A HREF=http://www.wlu.ca>Wilfrid Laurier University's
Homepage</A>

    <A HREF= http://www.wlu.ca/~wwwua/>
      </p>
    </A>

  </body>
</html>
```

TIP: If you're linking to Web pages on your own site, use relative URLs; don't specify the hostname or even the full path. Only give enough of the path so that the Web browser knows how to find the file relative to the current path. If you create a link to `images/picture.gif` in a Web page found at `http://www.host.com/myweb`, the browser knows to open the file from `http://www.host.com/myweb/images/picture.gif`.

Adding Forms to Your Web Page

You use the <FORM> container to add a form to your Web page. <FORM> marks the beginning of the form and </FORM> marks the end of the form. The Web browser displays any text within this container as it normally does--a fact you can use to create prompts for each field in the form. The <PRE> and </PRE> tags tell the browser that you've preformatted the form, and you don't want the browser messing with it by collapsing all the white space.

```
<html>
  <head>
    <title>A Simple Form with a Text Field</title>
  </head>
  <body>
```

```
<pre>
<form>
  Type your name:
  <input NAME=Name TYPE=TEXT SIZE=30>
</form>
</pre>
</body>
</html>
```

You add fields to the form by using a variety of tags, the most useful of which is the `<INPUT>` tag. In this example, the Web browser displays the prompt Type your name: and then creates a text box next to it due to the `<INPUT>` tag. The `<INPUT>` tag's `NAME` attribute gives a name to the field so that you can identify it when the browser submits the form. Its `TYPE` attribute tells the browser that you want to create a text field, and its `SIZE` attribute specifies the number of characters wide you want the field to be when displayed on the Web page.

Specifying How to Submit a Form

In order to collect the data the user enters into the form, you have to submit it. You can submit a form to a script on the Web server, for which you'll have to get the help of your service provider, or you can submit the form to your Internet mail address.

The `<FORM>` tag provides two attributes you use to specify how you want to submit a form. First, the `ACTION` attribute is the URL to which you want the form submitted. If you are submitting the form to a script on the Web server, you assign the URL of the script to this attribute. If you want to submit the form to your Internet mail address, you assign the URL of your mail address to this attribute (for example, `mailto:gmorbey@wlu.ca`).

Second, the `METHOD` attribute tells the browser how to submit the data. If you assign `POST` to this attribute, the browser sends the data separately from the URL. If you assign `GET` to this attribute, the browser attaches the data to the end of the URL. If you are submitting a form to your Internet mail address, you assign `GET` to `METHOD`.

Adding a Submit Button

After you've told the browser how you want a form submitted, you have to provide a submit button that actually does the work. Take a look at the listing below which is an extension of the previous listing. Notice the METHOD and ACTION attributes in the <FORM> tag. These attributes specify that you want to post the form to the mail address shown (don't forget to include the mailto: portion of the URL). The second <INPUT> tag is the actual Submit button.

```
<html>
  <head>
    <title>A Simple Form with a Text Field</title>
  </head>
  <body>

    <pre>
    <form METHOD=GET ACTION=mailto:gorbey@wlu.ca>
      Type your name:
      <input NAME=Name TYPE=TEXT SIZE=30>

      <input TYPE=SUBMIT VALUE=Submit>
    </form>
    </pre>
  </body>
</html>
```

CAUTION: Internet Explorer does not support posting a form to a mailto: URL. Navigator does, however.

TIP: When the user clicks the Submit button on a form like this, Netscape Navigator warns him that he is submitting a form via Internet mail, which causes his Internet mail address to be known to the recipient.

Adding Input Boxes with the <INPUT> Tag

You create input boxes using the <INPUT> tag. This tag does not require an ending tag like this: </INPUT>. The first <INPUT> tag on the form puts a field called Name on the Web page that is of TEXT type with a length of 30 characters. It looks like this:

```
<INPUT NAME=Name TYPE=TEXT SIZE=30>
```

The NAME attribute gives the control a name so that you can identify the data when it's submitted to a script on the Web server or to your Internet mail address. The TYPE attribute tells the browser that you're creating

an input box. The last attribute, SIZE, tells the browser how many characters wide you want the input box to be.

TIP: You can use the <INPUT> tag's VALUE attribute to prefill the control with data, like this:

```
<INPUT NAME=Name TYPE=TEXT SIZE=30 VALUE="Type your name here">
```

Adding Memo Fields with the <TEXTAREA> Tag

The <TEXTAREA> tag defines what you might call a memo field (a multilined text box). This defines any number of rows and columns in which the user can type text. The following <TEXTAREA> tag shows an example of a memo field that has five rows and 50 columns. <TEXTAREA> is a container, and anything you place inside the container is prefilled in the memo field.

```
<TEXTAREA NAME="Address" ROWS=5 COLS=50>Type your address here</TEXTAREA>
```

Again, you give it a name by using the NAME attribute. You also specify the number of rows and columns by using the ROWS and COLS attributes. The ending tag </TEXTAREA> ends the text area control. The browser prefills the text area with anything you put between the <TEXTAREA> and </TEXTAREA>.

Adding Radio Buttons with the <INPUT> Tag

The following example shows you how to add a group of radio buttons by using the <INPUT> tag. The user can choose one or the other, but not both. You create a radio button by setting the TYPE attribute to RADIO and by giving each button in the group the exact same name. Because both of the following <INPUT> tags have the name Skill and both set the TYPE attribute to RADIO, they both are considered a part of the same group.

```
<INPUT NAME=Skill TYPE=RADIO VALUE="HTML Wizard" CHECKED>HTML Wizard  
<INPUT NAME=Skill TYPE=RADIO VALUE="HTML Wannabe">HTML Wannabe
```

TIP: To specify that you want one of the radio buttons to be checked when the browser first loads the Web page, add the CHECKED attribute to it.

Adding List Boxes with the <SELECT> Tag

The <SELECT> and </SELECT> tags define a list box. This tag is a container in which you add <OPTION> tags to specify each item you want to appear in the list. The user can select one or more items from the list. Here's what the <SELECT> tag looks like:

```
<SELECT NAME=Options MULTIPLE SIZE=4>
  <OPTION>Learn more about basic HTML programming
  <OPTION>Learn more about HTML programming with forms
  <OPTION>Learn more about HTML programming with frames
  <OPTION>Spend the day at the beach
</SELECT>
```

You give a name to the list by using the NAME attribute. You specify the number of elements to display at one time by using the SIZE attribute. You add elements to an HTML list by using the <OPTION> tag. Each <OPTION> tag you put between the <SELECT> and </SELECT> tags appears in the list. The text to the right of the <OPTION> tag is the text the user actually sees in the list. Note that the <OPTION> tag is not a container, so you don't have to close it.

Adding Buttons with the <INPUT> Tag

You create buttons with the <INPUT> tag. Instead of assigning TEXT to TYPE, you assign BUTTON. The VALUE attribute tells the browser what text to display inside the button when the browser draws it:

```
<INPUT NAME=MyButton TYPE=BUTTON VALUE="Click Me">
```

Framing Your Web Site

Frames divide the browser window into sub-windows (frames). You might know them as window panes. You put a different Web page in each frame.

You either divide the browser window into rows first, and then columns, or you divide it into columns first, and then rows. In the first case, you

divide the browser window horizontally. You then can divide each slice into columns. In the second case, you divide the browser window vertically. You then divide each vertical slice into rows.

```
<HTML>
<HEAD>
  <TITLE>A Web Page with Frames</TITLE>
</HEAD>

<BODY>

<FRAMESET ROWS="25%,75%">
  <FRAME SRC="example1.html">
  <FRAMESET COLS="25%,75%">
    <FRAME SRC="example2.html">
    <FRAME SRC="example3.html">
  </FRAMESET>
</FRAMESET>
</BODY>
</HTML>
```

TIP: You can create a site with borderless frames: Set `FRAMEBORDER="NO"` and `BORDER="0"` in the `<FRAMESET>` tag.

Frame-type work starts with the `<FRAMESET>` tag and ends with the `</FRAMESET>` tag. You specify each frame and its associated HTML file within these two tags. You use the `ROWS` attribute to split the frame set up into rows first, or you use the `COLS` attribute to split it up into columns first. You set the value of the `ROWS` and `COLS` attributes to a comma-delimited list of values that indicate how much larger that row or column is. You can use a combination of percentages or pixels.

If you've created three rows or columns by using the `<FRAMESET>` tag, each subsequent frame you define fills these rows in the order the browser encounters them. Each `<FRAME>` tag specifies the HTML file to use in that particular frame. You use the `SRC` attribute to specify the file name--absolute or relative.

You can also further divide a frame using another `<FRAMESET>` tag.. Because the browser already divided into rows, we set the enclosed `<FRAMESET>` tag to divide the second row into columns. The first column is 25 percent of the current row, and the second column is 75 percent of the current row. The browser loads the HTML file specified by

the first <FRAME> tag into the first column and the HTML file specified by the second <FRAME> tag into the second column.

The HTML files that you put in each frame should not have the <HEAD> and <BODY> tags. They should be simple HTML files that begin with the <HTML> tag and end with the </HTML> tag. The browser displays everything between these two tags in the frame.

Scripting Your Web Site

You use scripts to make your Web page dynamic, interactive, and, on the whole, a lot more exciting than an ordinary Web page. You can validate a form before submitting it to the server, for example. Better still, you can control the data that the user inputs into a form as the user enters it. Forms aren't the only HTML object you can control with a script, either. If you can put it on a Web page, you can attach a script to it or control it with a script.

The most popular scripting language is JavaScript, for two reasons. First, both Internet Explorer and Communicator support it. Second, JavaScript is an extremely powerful scripting language that satisfies the most demanding requirements. Note that the only other scripting language available is Microsoft's VBScript, which is loosely based on Visual Basic.

Adding Scripts to Your Web Page

Now that you've got the basics under your belt, let's add some JavaScript to your Web page. You use the <SCRIPT> tag to do just that. This tag has a single attribute called LANGUAGE. You set the value of this attribute to JAVASCRIPT to denote that the script uses the JavaScript language, as opposed to another language such as Microsoft's VBScript. Because <SCRIPT> is a container, you end a script with the closing </SCRIPT> tag. Here's what this tag looks like:

```
<SCRIPT LANGUAGE=JAVASCRIPT>  
    JavaScript Statements  
</SCRIPT>
```

A lot of users use a browser that can't interpret JavaScript. When one of these browsers encounters a script, it ignores the <SCRIPT> and </SCRIPT> tags because it doesn't know how to interpret them. The

browser displays everything in between those two tags as HTML content, however, which is not exactly what you had in mind--is it?

You use an HTML comment to hide scripts from those “scriptless” browsers. Make the opening HTML comment the first line within the script block. The closing HTML comment should be the last line in the script block. Note that you begin this line with a JavaScript comment (//), because after JavaScript starts interpreting code, it thinks that everything else in the script is actual JavaScript code. Script-enabled browsers ignore the comments, while scriptless browsers ignore all of the content between the comments.

```
<SCRIPT LANGUAGE=JAVASCRIPT>
<!--
  function AddSeries( Last )
  {
    intTotal = 0;
    for( j = 1; j <= Last; j++ )
    {
      intTotal += j;
      alert( intTotal );
    }
    return intTotal;
  }

  document.write( "Hello world!" );
//-->
</SCRIPT>
```

Connecting Scripts to Events

You must learn how to connect scripts to the objects on your Web page. Before you're finished reading this chapter, you're going to learn how to download a script from the Internet and embed it into your Web page. Many times, though, you're going to actually have to hook up that script to the objects in your Web page. You don't just drop a new CD player in your stereo cabinet without plugging it into your receiver, do you? The same is true for JavaScript--you have to plug it in.

Most of the objects on a Web page have events.

Name	Event is raised when
onBlur	A form's field loses focus
onChange	The contents of a field or list changes

onClick	The user clicks an object
onFocus	A form's field gains focus
onMouseOut	The mouse moves out of an object
onMouseOver	The mouse moves over an object
onSelect	The user selects a form's field
onSubmit	The user clicks a form's submit button

Recall that objects, events, and event-handlers all work together. You have to associate an object with an event-handler via that object's event. You do that by adding the event's attribute to the object's HTML tag and setting its value to a single JavaScript statement that executes the function. For example, to associate a JavaScript function called `MyFunction()` with the `onClick` event of a button, you write the button's `<INPUT>` tag like this:

```
<INPUT TYPE=BUTTON NAME=BUTTON onClick="MyFunction()">
```

Add another event attribute from the first column of the table to the tag in the previous example to associate an event-handler with that event, like this:

```
<INPUT TYPE=BUTTON NAME=BUTTON onClick="MyFunction()"
MouseOut="ByeMouse()">
```

Suppose, for example, you downloaded a really cool script that displays a message in the status line when the user moves the mouse over a link. You drop the script shown below into your Web page, but nothing happens.

```
<SCRIPT LANGUAGE=JAVASCRIPT>
<!--
  function Message()
  {
    window.status="You just passed over a link; wow!";
  }
//-->
</SCRIPT>
```

The problem is that you haven't plugged it into your Web page. You do that by associating the function called `Message()` with the `onMouseOver` event of each anchor in your HTML file, as shown in the following script.

```
<HTML>
<HEAD>
  <TITLE>Plugging in the New Script</TITLE>
```

```
</HEAD>
<BODY>
  <A HREF="more.htm" onMouseOver="Message()">See more stuff</A>
  <A HREF="less.htm" onMouseOver="Message()">See less stuff</A>
</BODY>
</HTML>
```

Finding Scripts to Put on the Page

The quickest way to add JavaScript flair to your Web page is by using a script that you download from a JavaScript gallery or library. For example, the **Template Studio** (<http://tstudio.usa.net>) is a comprehensive gallery that contains numerous scripts you can use in your Web page. The scripts you find on this site include scripts for detecting the user's browser, dynamically replacing an image, controlling a window, displaying messages, and setting cookies.

Gamelan (<http://javascript.developer.com>) also has a comprehensive site that you can use to find good JavaScript for your Web page. Tutorial examples, games, and other miscellaneous scripts are featured on this Web site. You'll also find documentation that helps a new JavaScript developer get up-to-speed in a heartbeat.

Publishing Your Web Page

If you don't currently have an Internet account, you can't publish your Web. In most cases, you'll find several ISPs from which to choose. When shopping for an ISP, make sure that a certain amount of Web space comes with your account. If not, or if they try to charge you extra for Web space, move on to another ISP.

Also there are numerous providers of free web-space on the Internet such as: Geocities (<http://www.geocities.com>), Tripod (<http://www.tripod.com>), Xoom (<http://www.xoom.com>).

If you do have an Internet account, double-check to make sure that you have Web space available. You can check your ISP's home page or you can ring the support line. Your ISP is probably using UNIX. This is important for five reasons:

- You'll probably use **FTP** (File Transfer Protocol) **to upload** your Web to the Web server.
- You'll probably use your **PPP user name** and **password** to access the FTP server. Your PPP user name is the name you type when you first log onto the ISP.
- You'll probably upload your Web to the path **\public_html**.
- You'll probably give your homepage the file name **index.html**.
- You'll probably browse your Web at **http://www.isp.com/~name**, where *www.isp.com* is the host name of your ISP's Web server and *name* is the user name you use to log onto the ISP (your PPP user name).

Just to make sure, though, you should verify each of these items with your ISP. You need to make sure you know the FTP address to which you'll upload files, the user name and password you'll use to access the FTP server, and the URL you'll use to browse your Web.

Uploading Your Web Pages

The only tool you need in order to post your Web pages to the Internet is an FTP client. Armed with the files that make up your Web site, the host name of your Web server, the path to which you upload your files, and your logon information, use these steps to post your Web to the Internet:

1. Start your FTP client, and **log onto your Web site's FTP server** using your **logon name** and **password**. The host name is probably the same as the host name used when accessing your Web. The logon name and password are probably the same as your PPP name and password.
2. Within the FTP client, change to the folder on the server that will contain your HTML files. If in doubt, switch to a folder called **public_html**; it's the default on most UNIX systems.
3. Make sure the first page in your Web site is called **index.html**. Do this so that when a user types the URL of your Web site without specifying a file name, the server will return your homepage by default.

4. On the Web server, **duplicate the directory structure you used on your computer** so that the relative URLs will match.
5. **Upload** all the files **from the root folder of your Web site**, on your computer, **to public_html** on the server. Then, for each subfolder in your Web site, upload those files to the appropriate folder on the server.
6. Close your FTP client and test your site in your favorite browser.

Checking Your Web Page's Popularity

Most ISPs keep track of usage information for you. This information gives you a good idea of how often someone visits your site, as well as what pages they look at.

Log onto your Web server by using an FTP client and look for a folder called **httpd_logs** (or just **logs**). If you don't see this folder, ask your ISP where they store usage information for your Web site. Within this folder, you find a number of files. Each file is a log and is given a file name that reflects the period of time that spans. For example, a file by the name of 970914.log spans the week of 9/14/97. Your provider will determine how many log files it allows you to keep at one time.

Within the log file, you see a line for each access to your Web site. It shows the source of the hit, which in most cases is the host name of the person's provider, the date and time of the hit, and the HTTP command issued by the user's browser. GET means that the user's browser requested a particular file whose name follows the word GET.

```
s1-tc-ppp20.monmouth.com - - [16/Sep/1997:04:37:12 -0500] "HEAD
/~jerry/books.htm HTTP/1.0" 200 -
p01-48.octacon.co.uk - - [16/Sep/1997:04:37:15 -0500] "GET
/~jerry/clip2.mid HTTP/1.0" 200 5051
```

Keep in mind that each line in the log file doesn't represent a new visitor. Each line represents a request by a user's browser; thus, a person who visits several pages on your site might generate hundreds of lines in the log file.

TIP: Programs such as Web Trends (<http://www.webtrends.com>) can run queries against your log that give you an idea of which pages are most popular, what times of the day do you get the most visits, and so on.

Learning More About HTML on the Web

The Web contains thousands of tutorials and reference works about HTML. The sites described in this section are the best, however. These sites are only a start, because each Web site contains links to other related sites.

TIP: Open

http://www.yahoo.com/Computers_and_Internet/Information_and_Documentation/Data_Formats/HTML in your browser, and choose from hundreds of Web sites dedicated to HTML.

The Web Developer's Virtual Library

URL: <http://www.stars.com>

If you can visit only one site related to HTML, this is it. You won't be disappointed. Not only is this site comprehensive, it's up-to-date with the latest standards. You'll find information about authoring, promotion, and more at this site.

Learning HTML 3.2 by Examples

URL: <http://www.hut.fi/~jkorpela/HTML3.2>

This is an excellent tutorial about HTML 3.2 that rivals many books on the subject. Jukka Korpela, the author of this site, walks you through HTML concept-by-concept and tag-by-tag, providing examples along the way.

HTML Help by the Web Design Group

URL: <http://www.htmlhelp.com>

HTML Help isn't a tag-by-tag reference. Instead, it provides information about style and writing HTML that's cross-browser-compatible. You'll find an overview of what's new in HTML 3.2, information about cascading

style sheets, tips for creating great-looking tables, a tutorial on using frames, and much more.

NCSA--A Beginner's Guide to HTML

URL:

<http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html>

NCSA (The National Center for Supercomputing Applications) is the birthplace of one of the first Web browsers: Mosaic. As such, they're in a unique position to provide a tutorial on HTML. The tutorial you'll find at this URL is a good place to start when learning how to write HTML for the first time.

W3C (World Wide Web Consortium)

URL: <http://www.w3.org>

The W3C is the driving force behind the latest and greatest HTML specifications, including HTML 3.2. The information on this site can be a little heavy. It's not meant for the beginner because it's primarily specifications and research results. Regardless, if you want to find the definitive source for HTML 3.2, this is it.