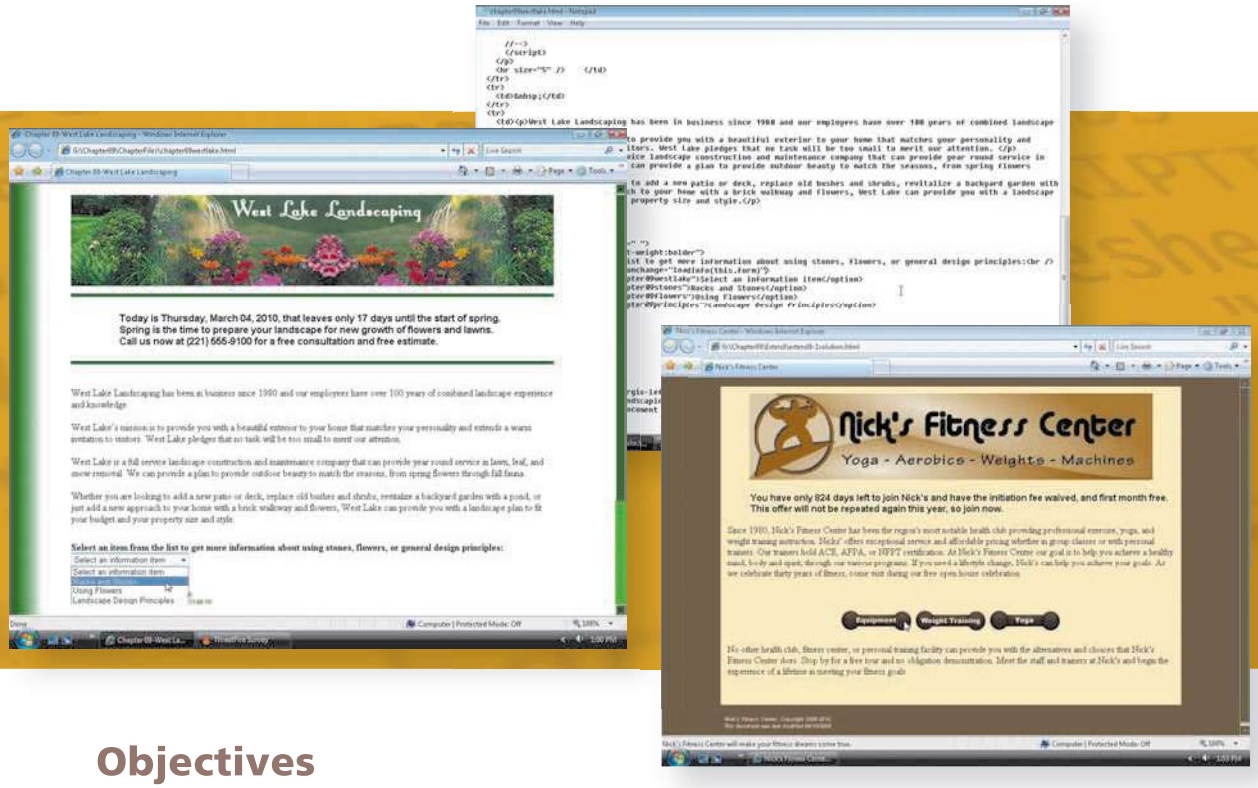
9 Integrating JavaScript and HTML



Objectives

You will have mastered the material in this chapter when you can:

- Discuss how to integrate JavaScript and HTML
- Insert <script> tags on a Web page
- Define and describe JavaScript variables
- Extract the current system date
- Calculate the number of days from the current date to a future date
- Describe the write() method of the document object
- Save the HTML file and test the Web page
- Write a dynamic message to a Web page
- Write a user-defined function that changes the color of the browser's scroll bar
- Construct a URL from a select list option choice
- Use the document's location property to link to a new Web page
- Use the lastModified property to display the last modified document date

9 Integrating JavaScript and HTML

Introduction

Many individuals, companies, and organizations rely on their Web sites as a key vehicle to communicate with friends, members, and current and future clients. Web pages often announce upcoming events, provide updated information, or act as a sales tool or catalog. Regardless of the content, Web page features should spark visitors' interests and entice them to return to the Web site. As with an advertisement, a Web page has six to seven seconds to attract and retain someone's attention.

An effective way to make Web pages interesting and useful is to include dynamic content and to make the Web page interactive. One way to add dynamic content to a Web page is to integrate JavaScript code onto the Web page. This chapter shows you how to add JavaScript code to a Web page to make it more dynamic and attractive.

Project — Creating the West Lake Landscaping Web Page

West Lake Landscaping has asked you to create a Web page that is dynamic and interactive with users, that is colorful and easy to navigate, and which showcases its landscaping skills and offerings. You have suggested a colored scroll bar to help balance the look of the page with the margin and banner colors, and a dynamic message that can be easily modified to address current issues or themes in landscaping or user needs. You also feel that using a drop-down menu will make the Web page more interactive for users, and keep the home page simple in its design. Another suggestion is to include the date the Web page was last modified so that users know the Web page content is current.

Chapter 9 shows how to add JavaScript code to an HTML file using these dynamic and interactive features to create the Web pages shown in Figure 9–1. The West Lake Landscaping home page includes a greeting that indicates the number of days until spring (Figure 9–1a). The home page also includes a select list (drop-down menu) containing items that link to three Web pages: a Web page with information about stones and rocks in landscaping (Figure 9–1b), a Web page with information on using flowers (Figure 9–1c), and a Web page with information about the principles of landscape design (Figure 9–1d).

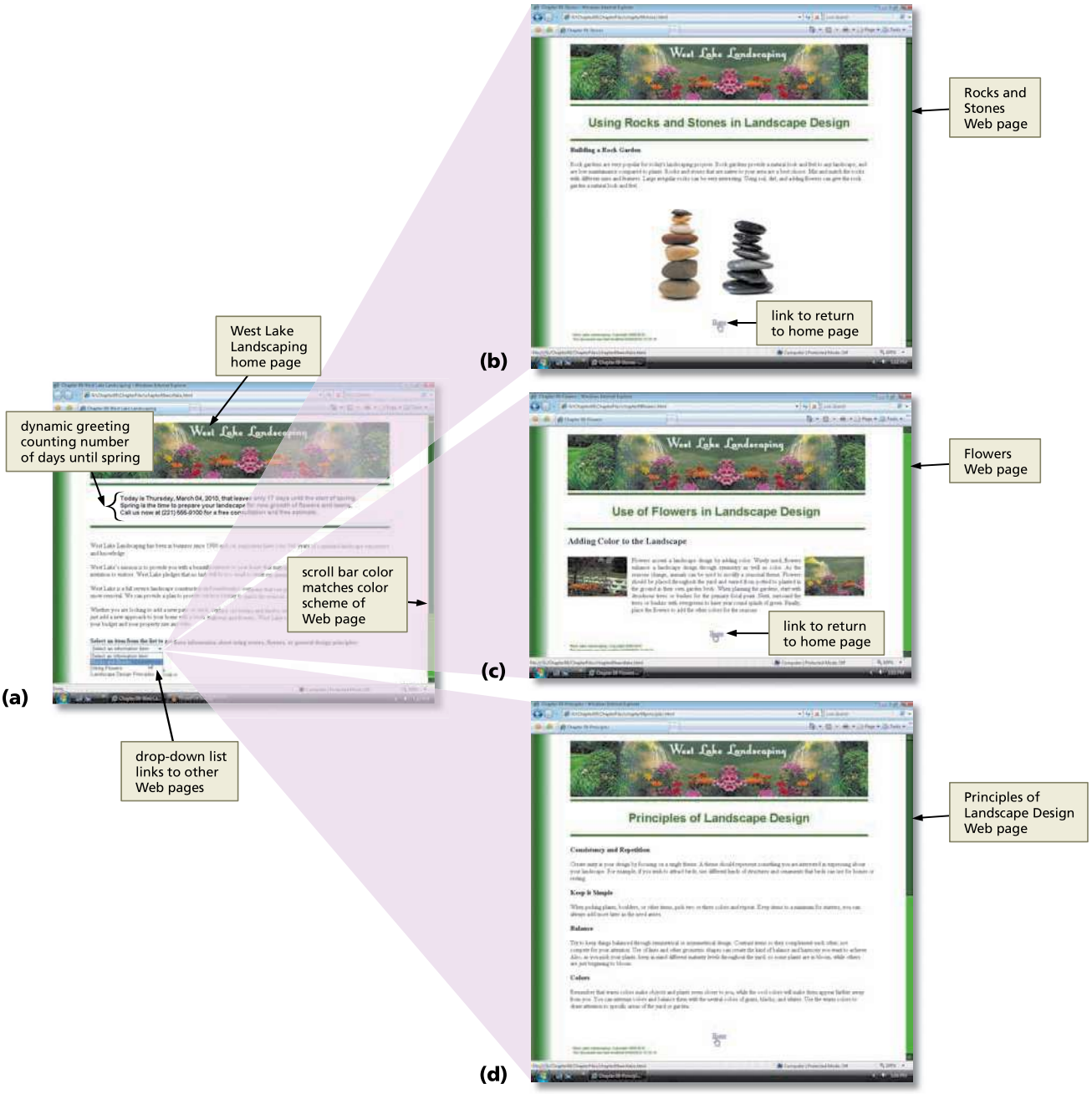


Figure 9–1

Overview

As you read through this chapter, you will learn how to integrate JavaScript onto a Web page as shown in Figure 9–1 by performing these general tasks:

- Write the embedded JavaScript code to display a countdown message
- Write a JavaScript function to change the color of the scroll bar
- Write a JavaScript function that uses a drop-down menu object to link to a new Web page
- Write the embedded JavaScript code to display the date the Web page was last modified and a copyright message

Plan Ahead

General Project Guidelines

When adding JavaScript or any scripting language to a Web page document, the actions you perform and decisions you make will affect the appearance and characteristics of the finished Web page. Before you write the JavaScript code, you should follow these general guidelines:

1. **Determine what you want the code to accomplish.** The JavaScript tasks in this chapter change the color of the scroll bar, use a drop-down menu (HTML <select/options>) to link to other Web pages, calculate a future date and display a dynamic message that includes the current date and number of days until another date, and display a copyright notice and date the Web page was last modified at the bottom of the Web page.
2. **Determine where on the Web page you want the code to appear.** JavaScript code is usually placed in the <head> section of the HTML code. When necessary, JavaScript must be embedded in specific locations of the HTML code so it can display specific text or images at that location on the Web page. The JavaScript user-defined functions are placed in the <head> section (you learn about user-defined functions later in the chapter). In the <body> of the HTML code, separate JavaScript <script> sections will display the dynamic message and the date last modified with the copyright information.

   The dynamic message displays beneath the West Lake Landscaping banner. The date last modified and copyright information display at the bottom of the Web page.
3. **Determine where you want to store the Web page during development**. The storage location of HTML code and associated images is very important. A best practice is to create folders to organize HTML files and graphics in a specific location. This practice makes finding and maintaining links to graphics and other Web site pages easy.

When necessary, more specific details concerning the above guidelines are presented at appropriate points in the chapter. The chapter also will identify the actions performed and decisions made regarding these guidelines during the creation of the Web page shown in Figure 9–1 on the previous page.

BTW

JavaScript Tutorials

Many Web sites provide help for JavaScript developers. For more information about links, search for key words such as "JavaScript Tutorials" or "JavaScript Help" in any good search engine.

JavaScript

Before adding JavaScript code to your Web page, you should understand some basics about the programming language. **JavaScript** is an event driven, object-based, programming language that provides various types of functionality to Web pages, such as the ability to interact with the user. An **event driven** programming language is one that responds to events, such as a user clicking a Submit or Calculate button. JavaScript is **object-based** because it is a scripting language that uses built-in objects that belong to the browser.

Built-in objects are values that are common to a browser (arrays, dates, strings, etc.), and neither depend on nor belong to another object. Table 9–1 contains a general list of the built-in JavaScript objects common to many browsers. JavaScript developers can create new objects based on the built-in objects and the new objects inherit properties from the original objects. For more information about these objects, see the JavaScript Quick Reference in Appendix E.

Table 9–1 Built-in JavaScript Objects

Object	Description
Array	Returns an ordered set of values
Boolean	Converts objects to Boolean values
Date	Accesses the system time and date
Document	Represents the content of a browser's window
Form	Represents forms created with the <form> tag
Function	Accesses information about specific functions
History	Keeps track of Web pages visited
Image	Represents images created with the <img> tag
Location	Switches to a new Web page
Math	Performs calculations
Navigator	Obtains information about the current Web browser
Number	Supports special constants
Screen	Gives platform-specific information about the user's screen
String	Represents a set of characters
Window and Frame	Represents a browser window or every frame within a browser window; every frame is a window and uses the same properties and methods as the window object

JavaScript objects have properties and methods. **Properties** are attributes that describe an object's characteristics. As shown in Table 9–2, an object and its property are written by separating the object from its property with a period. A specific value can be assigned to a property as shown in the following example:

```
dog.breed="terrier"
```

where the dog is the object, breed is the property, and terrier is the value assigned to the property.

BTW

Object Oriented Programming (OOP)
Object-oriented programming is an approach to programming in which the data and the code that operates on the data are packaged into a single unit called an object.

Table 9–2 Object and Property

General form:	object.property
Comment:	where the object is stated first, then a period, then the descriptive property. A value can be assigned to the property, or the property can return a value as shown in the examples below.
Examples:	document.bgColor="lightblue" browser=navigator.appName

Methods are actions that an object can perform. For example, methods associated with the dog object might be eat, fetch, and sit up. An object and one of its methods would be written as:

```
dog.fetch()
```

where dog is the object and fetch is a method of the dog object. Methods are followed by parentheses, which may be empty, or may contain an argument.

BTW

JavaScript Methods and Arguments
Not all JavaScript methods require an argument. With some methods, if an argument is used, it generates an error.

An **argument** is a value given to a method. Some methods require arguments, and others do not. For example, given a dog object and the fetch() method, a statement could be written as:

```
dog.fetch("ball")
```

where the argument "ball" describes what the dog fetches.

As shown in Table 9–3, the general form of writing an object with its method is similar to writing objects and properties.

Table 9–3 Object and Method

General form:	objectname.method(argument value)
Comment:	where objectname is the object, method is the action, and parameters are optional items or instructions the method should use. A period separates the object name from the property or method name.
Examples:	document.write("Some text") window.alert("This is a message") var ToDayDate=Date.toString( )

User-Defined Functions

A **user-defined function** is JavaScript code written by a Web developer to perform a particular task. The function can be used whenever that task is needed, eliminating the need to repeat the code several times throughout an application. JavaScript also contains a number of built-in functions called **global functions**, such as close() to close a window, open() to open a window, and print() to print the contents of a window. Most of these functions actually belong to the window object, but because the window object is assumed, they are called built-in functions. For a complete list of built-in functions, see the JavaScript Quick Reference in Appendix E.

BTW

Other Scripting Languages
JavaScript and VBScript are the two main client-side script languages used today. There are over a dozen other scripting languages that work on specific platforms or operating systems. Perl and PHP are other popular scripting languages.

Most JavaScript user-defined functions are called or activated using event handlers. An **event** is the result of an action, such as a mouse click or a document loading. An **event handler** is JavaScript's way to associate an action with a function. In this project, you first write the functions, and then the event handlers that will associate the functions with specific events, such as loading the Web page.

Adding JavaScript to a Web Page

Now that you have planned what you need the JavaScript to accomplish and where it should appear, you can begin modifying the Web page using a text editor.

To Open an Existing HTML File

The following step shows how to open the chapter09.html file included in the Data Files for Students.

1

- Start Notepad, and, if necessary, maximize the window. Click Format on the menu bar. If the Word Wrap command does not have a check mark next to it, click Word Wrap.
- With a USB flash drive connected to one of the computer's USB ports, click File on the menu bar and then click Open.
- If Computer is not displayed in the Favorite Links section, drag the top or bottom edge of the Open dialog box until Computer is displayed.

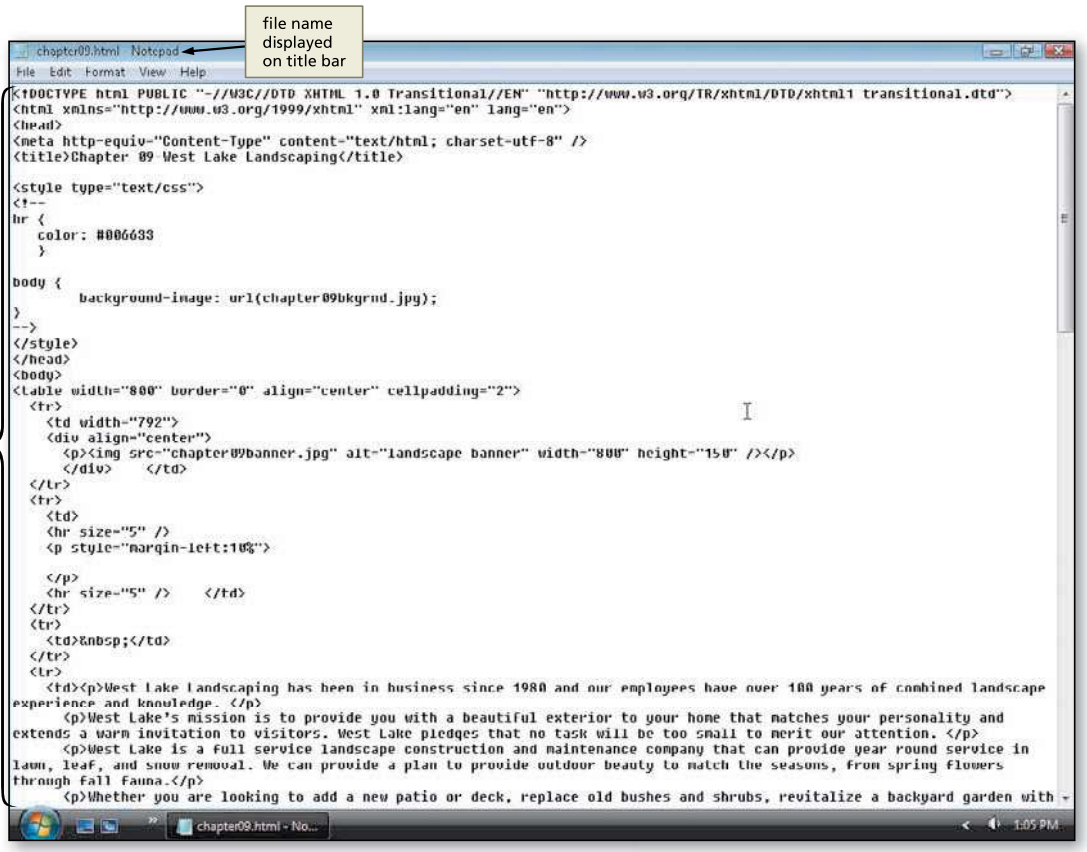


Figure 9–2

- Click Computer in the Favorite Links section to display a list of available drives.
- If necessary, scroll until UDISK 2.0 (G:) appears in the list of available drives.
- Double-click USB drive (G:). Open the Chapter09 folder, and then double-click the ChapterFiles folder in the list of available folders.
- If necessary, click the Files of type box arrow, and then click All Files. Click chapter09.html in the list of files.
- Click the Open button in the Open dialog box to display the HTML code for the chapter09.html Web page as shown in Figure 9–2.

Inserting <script> Tags in HTML Code

JavaScript code can be placed anywhere in the HTML code in a <script> section. A standard JavaScript coding practice is to place variables and user-defined functions in the <head> section and other JavaScript code sections in the <body> section. JavaScript code sections placed in the <body> section are used to embed JavaScript code at specific locations on the Web page, such as to display dynamic messages.

Plan Ahead

Some JavaScript code must be embedded in the HTML code so it will display text at that location. Such code must be placed in a <script> section. To display a dynamic message on a Web page, you must find the location in the HTML code where you want the message to display. To display a message with the current date and the number of days until a specific future date, you must:

- **Create the <script> section in the appropriate HTML location**. In this project, the appropriate location is in a table cell.
- **Define variables**. A set of variables must be defined to work with the system date to calculate the number of days from today to the start of spring.
- **Calculate the number of days until spring**. To calculate the number of days to the start of spring, you write code to subtract the current date from the future date.
- **Display a message string**. Using the document.write() method, write the message to the Web page.
- **Close the <script> section**. All HTML tags must have a closing tag. If you fail to close the <script> section, you will have undesired results.

BTW

Script Tag Attributes
Although the language attribute has been deprecated, many browsers still accept it, and many JavaScript coders still use it. The type attribute is supported by the XHTML and XML standards.

In this chapter, you will use only JavaScript features that work in the latest versions of Microsoft Internet Explorer. JavaScript sections always start with a <script> tag, which indicates the language being used. Similar to other HTML tags, the JavaScript <script> tag has a start <script> tag and an end </script> tag.

The general form of the script tag is shown in Table 9–4. The <script> tag supports several attributes, including src, type, and defer. In the past, the start <script> tag was written as <script language="JavaScript">, but as noted in the Table 9–4 example, the preferred style is to use the type attribute. If the type or language attribute is omitted, most browsers default to JavaScript.

Table 9–4 JavaScript Section

General form:	<script src="url" type="content-type" language="language" defer>
Comments:	where script is the script tag, src specifies the location of an external script URL, type indicates the content-type or specific scripting language, language is a deprecated attribute, and defer is a Boolean attribute that indicates whether the script is going to generate any document content. The src, language, and defer attributes are optional.
Example:	<script type="text/javascript"> <!- -Hide from old browsers miscellaneous JavaScript code //--> </script>

BTW

JavaScript Comments
Comments can be added to JavaScript in two ways. The double slash [//] is used to indicate a comment for a single line. To comment multiple lines, begin a comment with a slash and asterisk [/*] and end the comment with an asterisk and slash [*/].

To Enter the Start <script> and Comment Tags

The following steps illustrate how to enter the <script> and HTML comment tags.

1

- Click the blank line (line 32) press the spacebar to indent as shown below the <p style= "margin-left:10%"> tag, as shown in Figure 9–3.

```
}
-->
</style>
</head>
<body>
<table width="800" border="0" align="center" cellpadding="2">
  <tr>
    <td width="792">
    <div align="center">
      <p><img src="chapter09banner.jpg" alt="landscape banner" width="800" height="150" /></p>
      </div>    </td>
  </tr>
  <tr>
    <td>
    <hr size="5" />
    <p style="margin-left:10%">
    </p>
    <hr size="5" />    </td>
  </tr>
  <tr>
    <td> </td>
  </tr>
  <tr>
```

line 32

press ENTER key at end of each line

Figure 9–3

2

- Type `<script type="text/javascript">` as the beginning of the script and then press the ENTER key.
- Type `<!--Hide from old browsers` and then press the ENTER key (Figure 9–4).

insertion point

```
</style>
</head>
<body>
<table width="800" border="0" align="center" cellpadding="2">
  <tr>
    <td width="792">
    <div align="center">
      <p><img src="chapter09banner.jpg" alt="landscape banner" width="800" height="150" /></p>
      </div>    </td>
  </tr>
  <tr>
    <td>
    <hr size="5" />
    <p style="margin-left:10%">
    <script type="text/javascript">
    <!--Hide from old browsers

    </p>
    <hr size="5" />    </td>
  </tr>
  <tr>
    <td> </td>
  </tr>
  <tr>
    <td><p>West Lake Landscaping has been in business since 1980 and our employees have over 100
experience and knowledge. </p>
      <p>West Lake's mission is to provide you with a beautiful exterior to your home that matche
extends a warm invitation to visitors. West Lake pledges that no task will be too small to merit
      <p>West Lake is a full service landscape construction and maintenance company that can provi
lawn, leaf, and snow removal. We can provide a plan to provide outdoor beauty to match the seasons
through fall fauna.</p>
      <p>Whether you are looking to add a new patio or deck, replace old bushes and shrubs, revita
a pond, or just add a new approach to your home with a brick walkway and flowers, West Lake can pr
plan to
```

code to begin JavaScript section embedded in body of HTML code and end </script> and comment tags will be entered when script section code is completed

Figure 9–4

Q&A

Why is an > end bracket not included on the HTML comment line?

Because the comment does not actually end until after the JavaScript code that you do not want to display in an older browser. The end comment tag (//-->) follows the JavaScript code.

JavaScript Variables

As in other programming languages, JavaScript uses **variables** to store values temporarily in internal memory. A variable's value can change, depending on the results of an expression or data entered by a user from a form. Variables must have a unique name, which must adhere to the naming conventions outlined in Table 9–5. JavaScript variable names are case sensitive, which means the variable name months is different from the variable name Months.

Table 9–5 Naming Conventions for JavaScript Variables

Rule	Valid Name Examples	Invalid Name Examples
Name must begin with a letter or underscore	menu	$menu
Rest of name must be letters, numerals, or underscores	Last_Name	Last-name
Name may not use spaces, hyphens, or punctuation	ZipCode	zip.code or zip code
Name may not contain JavaScript objects, properties, and reserved words	xNow	Date

JavaScript variables are considered global, unless the variable is defined within a user-defined function, in which case it is considered a local variable. **Global** means that the variable value is available for use anywhere inside the HTML file Web page. To define a variable as global, it must be declared in the <script> section before any of the user-defined functions. **Local** means that the variable's value is available only in the function in which it is defined.

A variable's **data type**, the type of data it stores, such as text or numbers, must be known so the computer knows how to store the data. JavaScript has four data types: numeric, string, date, or Boolean. **Numeric data types** hold numbers. **String data types** are variables that hold characters or a combination of letters, numbers, or symbols. **Date data types** contain a date and time. **Boolean data types** contain logical data that can be one of two values, such as True/False or Yes/No.

JavaScript variables are **loosely typed**, which means they do not have to be assigned an initial specific data type as in other programming languages. Instead, JavaScript indicates the data type by declaring the variable with an initial value. This feature allows variables to be flexible and store any data type. Web developers, however, do not recommend changing a variable's data type in the middle of JavaScript code. This action may create an error, which can be very difficult to find. Table 9–6 shows the general form of declaring a variable and assigning a value to it.

BTW

Undefined Variables
If a variable's value, which has not been defined or declared previously, is used or displayed on the Web page, JavaScript assigns the value "undefined" to that variable. An undefined variable can cause errors in mathematical calculations.

Table 9–6 Assigning Values to Variables

General form:	var variableName=value
Comment:	where var is an optional keyword to designate a variable; variableName is a valid variable name; and value is the string, numeric, date, or Boolean value being assigned to the variable.
Examples:	var NickName="Jasper" // This variable is a string data type var lineCnt=1 // This variable is a numeric data type var Continue=false // This variable is a Boolean data type

In the examples in Table 9–6, the keyword var, meaning variable, appears before the variable name. A **keyword**, or **reserved word**, is a word with special meaning in a programming language. The JavaScript var keyword is optional for global variables; however, it is good programming practice to precede the variable name with the var keyword. In addition, the var keyword is required for local variables defined within a function.

Extracting the Current System Date

The built-in Date() object accesses the current system date and time. On the West Lake Landscaping Web page, the Date() object and several of its methods are used to extract the current system date and then display it on the Web page as part of the greeting.

Plan Ahead

For the dynamic greeting on the home page, you need to calculate the number of days between two dates. First, you must extract information about the current date using the following steps:

- Obtain the current system date with the Date() object and create a new object instance
- Use the toLocaleString() method to convert the date to a string to be manipulated
- Use the substring() method to extract the month from the string
- Use the substring() method to extract the day of the week from the string
- Use the indexOf() method to locate the position of the year in the string
- Use the substr() method to extract the year from the string

To manipulate the Date() object, a new object instance must be created. Table 9–7 shows the general form of the JavaScript statement to create a new object instance, which uses the new keyword and assigns the built-in object to a variable. This variable is referred to as an **object instance variable**.

BTW

The Date() Object
The Date() object can use three other methods to build a string for a current date: getDate(), getMonth(), or getFullYear(). The getDate() method returns the date in the month; the getMonth() returns the value of the month as a number from 0 to 11 with 0 representing January; and the getFullYear() method returns the four digit year. Because the getMonth() method returns an integer that represents the month, the developer must add 1 to the result to get the current month.

Table 9–7 Creating a New Object Instance

General form:	var variableName=new Builtin_Object
Comments:	where variableName is the name of the new object instance, new is the required keyword, and Builtin_Object is the name of the object from which the new object instance is to be created.
Examples:	var sysDate=new Date() var sysDate=new Date("March 21, 2010")

The parentheses in the Date() object means a specified date and time other than the current system date and time can be used. The first example shown in Table 9–7 has no specific date value provided, thus the current system date and time from the computer is stored in the variable sysDate. The second example in Table 9–7 has a specific date value enclosed within quotation marks inside the parentheses. That value, March 21, 2010, is assigned to the object instance variable sysDate.

Converting the System Date to a String To use the date and time value stored in the variable sysDate, the variable first must be converted to a string using the toLocaleString() method. Table 9–8 shows the general form of the toLocaleString() method.

Table 9–8 toLocaleString() Method

General form:	var variable=dateString.toLocaleString()
Comment:	where dateString is an object instance and the toLocaleString() method converts an object instance of the Date() to a string using the default display format used on the client computer.
Example:	var curDate=sysDate.toLocaleString()
Result:	curDate contains the date and time stored as: Day of the Week, Month, Date, Year HH:MM:SS

Once the current system date has been converted to a string, the JavaScript indexOf(), substring(), and substr() methods can be used to extract the day, the month, the date, the year, and the hours (HH), minutes (MM), and seconds (SS) to be displayed on the Web page.

BTW

The IndexOf() Method
Items in a select list are indexed in the order in which they appear in the list. The first item is indexed as 0 (zero). Set the selectedIndex property to 0 (zero) to clear any existing selected items.

Using the indexOf() Method Table 9–9 explains how the indexOf() method searches a string for a particular value, which is enclosed within the quotation marks, and then returns the relative location of the parameter value within the string. If the search finds the value in the search string object, the indexOf() method returns the relative position of the value within the string object. If the search value is not found, the indexOf() method returns a negative one (-1).

Table 9–9 indexOf() Method

General form:	var position=stringValue.indexOf("x")
Comment:	where stringValue is a string in which a search is conducted, x is the value to be searched for within the stringValue, and position is the location of x in the string. The value x must be a literal value.
Examples:	curDate="March 21, 2010" dateLocate=curDate.indexOf(",")
Result:	returns the relative position of the comma found in the string value of curDate: 8

Using the substring() Method to Extract the Month from a String The substring() method uses two parameters (x,y), where x is the starting point of the string and y is the location of the last character needed. If only an x parameter is given, the substring() method returns all the characters starting at that position through the end of the string. Table 9–10 describes the general form of the substring() method.

Table 9–10 substring() Method

General form:	var variable=string.substring(x,y)
Comment:	where string is any string object. The substring method extracts a portion of a string, starting at location x and ending at location y. x and y may be constants or variables.
Example:	weekDay=dayofweek.substring(0, dateLocate)
Result:	the variable weekDay contains the substring

Using the substr() Method Table 9–11 describes the substr() method. The substr() method is similar to the substring() method, in that it extracts part of a string. Although the methods perform similar functions in JavaScript, they differ in how they use parameter values. The substring() method uses the exact byte locations in a string to extract part of the string between the x and y locations, whereas the substr() method uses a length value to extract y number of characters starting at x location.

Table 9–11 substr() Method

General form:	var variable=string.substr(x,y)
Comment:	where string is any string object instance. This method extracts a portion of a string, starting at location x for a length of y. x and y may be constants or variables.
Example:	year=dayofweek.substr(yearLocate, 4)
Result:	the variable year contains the four-digit year

Both the substring() and substr() methods use relative addressing as a means of locating characters in a string. A **relative address** is the location of a byte in a string of bytes, as defined by its position relative to the first byte in that string. As an example, assume the data in Table 9–12 is a string value stored in the variable birthDay. The address of the first byte in a string of characters is zero (0). To extract the year 2010 from the string using the substring() method, the JavaScript code would be written as:

```
birthDay.substring(17,20)
```

Table 9–12 Relative Addressing

S	u	n	d	a	y	,		M	a	r	c	h		7	,		2	0	1	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Table 9–13 shows the JavaScript code used to extract the system date for the West Lake Landscaping Web page. The JavaScript code uses the toLocaleString(), substr(), substring(), and indexOf() methods of the Date() object to obtain the current system date and then extract the weekday, the date, the month, and the year. Once these values have been extracted and assigned to variables, they can be displayed on the Web page.

Table 9–13 Code to Extract the System Date

Line	Code
41	`var today = new Date()`
42	`var dayofweek = today.toLocaleString()`
43	`dayLocate = dayofweek.indexOf(" ")`
44	`weekDay = dayofweek.substring(0, dayLocate)`
45	`newDay = dayofweek.substring(dayLocate)`
46	`dateLocate = newDay.indexOf(",")`
47	`monthDate = newDay.substring(0, dateLocate+1)`
48	`yearLocate = dayofweek.indexOf("2010")`
49	`year = dayofweek.substr(yearLocate, 4)`

Line 41 creates the new date object instance variable, today, and assigns the current system date and time to the variable. Line 42 converts the date and time value stored in the today variable to a string, using the toLocaleString() method and assigns it to the dayofweek variable. To find the day of the week in the string, the indexOf() method in line 43 looks for the first blank space in the string. Line 44 uses the substring() method to extract the day of the week, while line 45 extracts the remainder of the string, which includes the month, day, year, and time. Line 46 looks for the comma that separates the date from the year using the indexOf() method. Using the address of the comma, the substring() method in line 47 extracts the date. To find the year, line 48 uses the indexOf() method and the current year to determine the starting address of the year. Line 49 uses the substr() method, using the starting address from the indexOf() method and length of the year, which is four characters long.

To Extract the Current System Date Using the Date() Object

The step on the next page illustrates how to write JavaScript code that uses the Date() object and its methods to extract the current system date.

1

- If necessary, click the chapter 09 - Notepad button on the task-bar to activate the Notepad window.
- Click line 41 below the <!--Hide from old browsers statement.
- Enter the JavaScript code shown in Table 9–13. Press ENTER at the end of each complete line of code. Enter the current year in the indexOf() method on line 48.

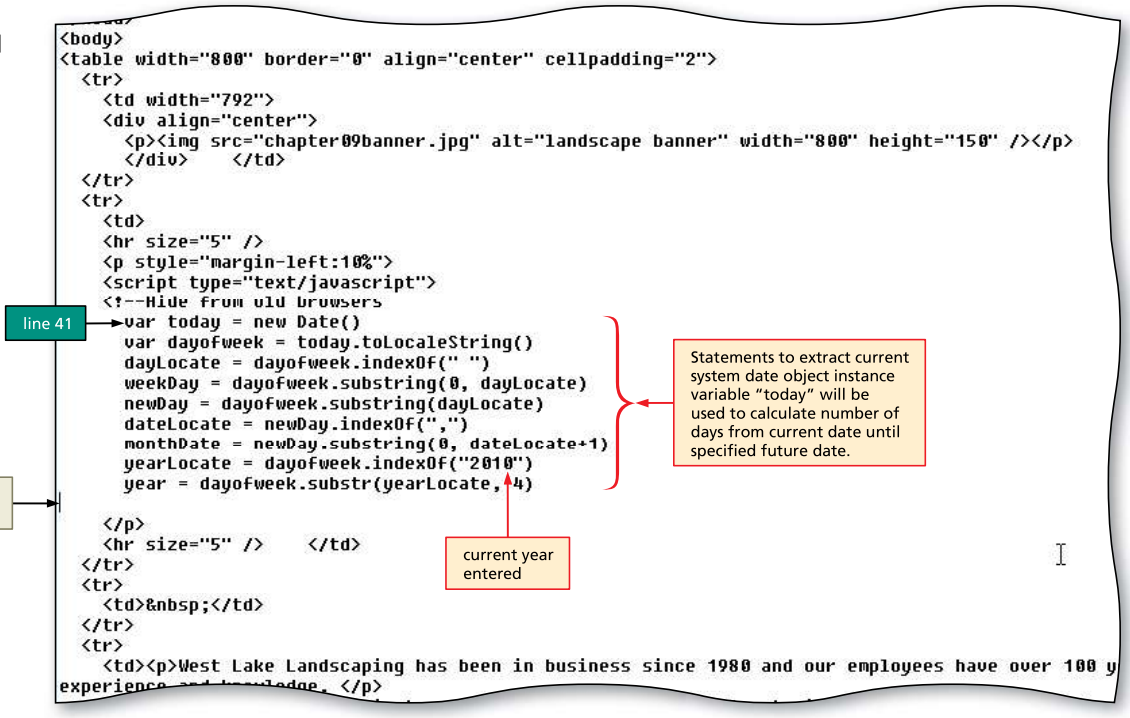


Figure 9–5

- After typing the last line in Table 9–13, press the ENTER key once more to leave space for additional JavaScript code.
- Compare what you typed to Figure 9–5. Correct any errors before proceeding.

Q&A Why is some of the code in Figure 9–5 indented?

The code is indented with the SPACEBAR for ease of reading. It does not affect the execution of the code. You may want to indent sections of code two or three spaces to make it easier to identify.

Q&A Can a date only be entered as, "March 21, 2010" in the Date() object?

The date can be entered as 2010, 2, 21 in the Date object. Because the Date() object starts numbering the months with 0, that means March is 2.

Calculating the Number of Days until a Future Event

Calculating the number of days until a future date can be useful for a dynamic greeting. With the West Lake Landscaping Web page, each time users view the Web page in a browser, the Web page displays a greeting that notifies them of the number of days until the start of spring.

Plan Ahead

The steps required to calculate a future date for a dynamic greeting include:

- Creating a Date() object instance with the future date and the current date
- Calculating the milliseconds between the current date and the future date using the getTime() method
- Subtracting to determine the total number of milliseconds between the current date and the future date
- Converting the number of milliseconds to days
- Eliminating any decimal portions of a day to make the number of days an integer

Creating a Date() Object Instance to Store a Future Date To calculate the number of days until a future date, an object instance of the Date() object must be created using the future date. As previously discussed, the Date() object can have a specific literal date as a value, which is assigned to an object instance variable. For example, the JavaScript code to set the date to the first day of spring, which is March 21, 2010, is written as follows:

```
var springDate = new Date("March 21, 2010")
```

The object instance variable springDate now will contain the future date of March 21, 2010.

To Create a Date() Object Instance to Store a Future Date

The following step illustrates how to enter code to create an object instance of the Date() object that stores a future date for the start of spring.

1. Click line 51.
2. Substituting the current year for the spring date March 21, 2010, type `var springDate = new Date("March 21, 2010)` to set the date to the start of spring and then press the ENTER key (Figure 9–6).

```
  <tr>
    <td>
    <hr size="5" />
    <p style="margin-left:10%">
    <script type="text/javascript">
    <!--Hide from old browsers
      var today = new Date()
      var dayofweek = today.toLocaleString()
      dayLocate = dayofweek.indexOf(" ")
      weekDay = dayofweek.substring(0, dayLocate)
      newDay = dayofweek.substring(dayLocate)
      dateLocate = newDay.indexOf(",")
      monthDate = newDay.substring(0, dateLocate+1)
      yearLocate = dayofweek.indexOf("2010")
      year = dayofweek.substr(yearLocate, 4)
line 51 →var springDate = new Date("March 21, 2010")
    </p>
    <hr size="5" />    </td>
  </tr>
  <tr>
    <td> </td>
  </tr>
  <tr>
    <td><p>West Lake Landscaping has been in business since 1980 and our employees have over 100 y
experience and knowledge. </p>
      <p>West Lake's mission is to provide you with a beautiful exterior to your home that matches
extends a warm invitation to visitors. West Lake pledges that no task will be too small to merit
      <p>West Lake is a full service landscape construction and maintenance company that can provi
lawn, leaf, and snow removal. We can provide a plan to provide outdoor beauty to match the seasons
through fall fauna.</p>
```

code to assign future date springDate variable

insert current year

Figure 9–6

Calculating Milliseconds Between Two Dates Using the getTime() Method The next step is to calculate the milliseconds between the current date and the actual date of the first day of spring using the getTime() method of the Date() object. The getTime() method returns the number of milliseconds that have elapsed since January 1, 1970 at 00:00:00 and another date. Calculating the number of milliseconds between two dates is easier than trying to count actual days, because each month has a different number of days and it may be necessary to take leap years into account. After determining the number of milliseconds, you then can convert that value to days.

To determine the number of milliseconds between a current date and another date, the JavaScript code should be written to subtract the value returned by the getTime() method of the future date and the value returned by the getTime() method of the current system date. For example, in this chapter, the JavaScript code is written as follows:

```
var daysToGo = springDate.getTime()-today.getTime()
```

To Calculate Milliseconds Between Two Dates Using the getTime() Method

The variable daysToGo will contain the number of milliseconds between the future spring date and the current system date. The following step shows how to enter the JavaScript code to calculate the milliseconds between the future spring date and the current system date.

1

- Click line 52, if necessary.
- Type `var daysToGo = springDate.getTime()-today.getTime()` to calculate the number of milliseconds and then press the ENTER key (Figure 9–7).

```
</head>
<body>
<table width="800" border="0" align="center" cellpadding="2">
  <tr>
    <td width="792">
    <div align="center">
      <p><img src="chapter09banner.jpg" alt="landscape banner" width="800" height="150" /></p>
      </div>    </td>
  </tr>
  <tr>
    <td>
    <hr size="5" />
    <p style="margin-left:10%">
    <script type="text/javascript">
    <!--Hide from old browsers
      var today = new Date()
      var dayofweek = today.toLocaleString()
      dayLocate = dayofweek.indexOf(" ")
      weekDay = dayofweek.substring(0, dayLocate)
      newDay = dayofweek.substring(dayLocate)
      dateLocate = newDay.indexOf(",")
      monthDate = newDay.substring(0, dateLocate+1)
      yearLocate = dayofweek.indexOf("2010")
      year = dayofweek.substr(yearLocate, 4)

      var springDate = new Date("March 21, 2010")
      var daysToGo = springDate.getTime()-today.getTime()

    </p>
    <hr size="5" />    </td>
  </tr>
  <tr>
    <td> </td>
  </tr>
  <tr>
    <td><p>West Lake Landscaping has been in business since 1980 and our employees have over 100
experience and knowledge. </p>
      <p>West Lake's mission is to provide you with a beautiful exterior to your home that matches
extends a warm invitation to visitors. West Lake pledges that no task will be too small to merit
      <p>West Lake is a full service landscape construction and maintenance company that can provi
lawn, leaf, and snow removal. We can provide a plan to provide outdoor beauty to match the seasons
```

line 52

code to calculate number of milliseconds between current date and future date

future date object from line 51

current date object from line 41

chapter09.html - No...

Figure 9–7

Converting Milliseconds to Days and Rounding Up Using the ceil() Method After calculating the number of milliseconds between the current date and the first day of spring, the next step is to convert the milliseconds to days. To convert milliseconds to days, the JavaScript code is written to divide the number of milliseconds stored in the daysToGo variable by the product of 1000*60*60*24. This expression represents the 1,000 milliseconds in a second, the 60 seconds in a minute, the 60 minutes in an hour, and the 24 hours in a day.

The value returned from the calculation daysToGo/(1000*60*60*24) probably will contain a decimal value. The West Lake Landscaping Web page, however, should display the number of days to the first day of spring as an integer value. The ceil() method of the Math object is used to round up to the nearest integer (for example, if the result is 12.843 days, 13 will display because of the ceil() method). The general form of the ceil() method is shown in Table 9–14.

Table 9–14 Math ceil() Method

General form:	var variable=ceil(value)
Comment:	where value may be the result of any calculation. The ceil() method returns a value that rounds the value up to the next highest integer.
Examples:	var myResult=ceil(-3.8) var myNumber=ceil(4.179)
Result:	myResult is -3 myNumber is 5

The JavaScript code for the West Lake Landscaping Web page is written as follows:

```
var daysToSpring = Math.ceil(daysToGo/(1000*60*60*24))
```

which first finds the product of 1000*60*60*24, then divides the value stored in the daysToGo variable by this product, and then raises the result (rounds up) to the next highest integer.

To Convert Milliseconds to Days and Round Up Using the ceil() Method

The following step illustrates how to enter the JavaScript code to convert the number of milliseconds to days.

1. Click line 53, if necessary.
2. Type `var daysToSpring = Math.ceil(daysToGo/(1000*60*60*24))` to convert milliseconds to days and then press the ENTER key twice (Figure 9–8).

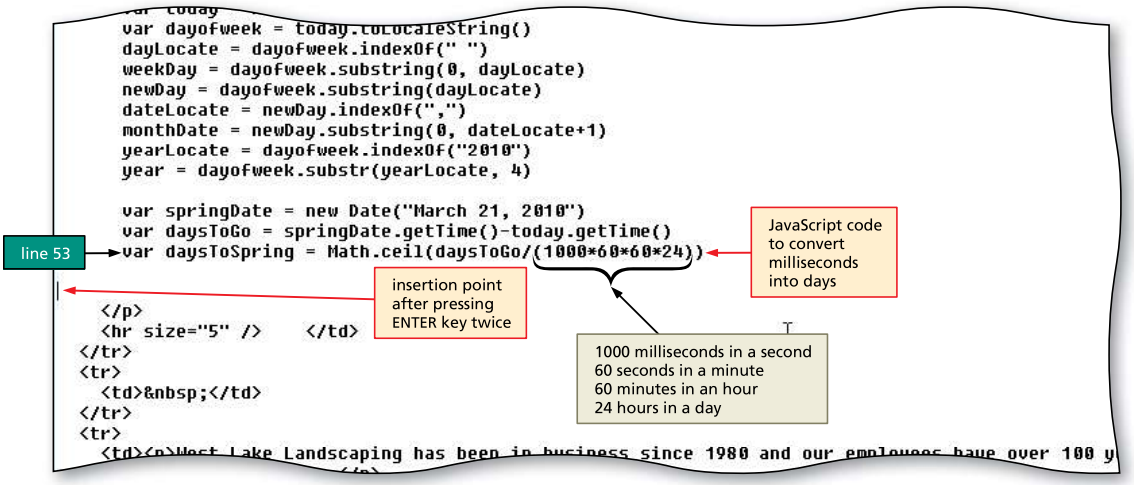


Figure 9–8

Plan Ahead

Using JavaScript to write dynamic text to a Web page.
Before writing dynamic text to a Web page, you need to determine what your message will say and how you will format it for display. Most messages will be a combination of text strings and variables. Text can be formatted using standard HTML tags or using embedded styles. Embedded styles can be placed in or tags to format text.

Writing Text and Variable Values to a Web Page

After the number of days until the first day of spring has been calculated, this data can be written on the Web page using JavaScript. To write data directly to the Web page, JavaScript uses the write() or writeln() methods of the document object. These methods use a string value, with any embedded HTML formatting tags, as a parameter inside the parentheses. Table 9–15 shows the general form of the write() and writeln() methods.

Table 9–15 write() and writeln() Methods

General form:	document.write("text string") document.writeln("text string")
Comment:	where text string is any combination of HTML tags, text, and variables.
Examples:	document.write(" Welcome to my Web page ") document.writeln(" Welcome to my Web page ")

BTW

JavaScript writeln() Method
The writeln() method works only in HTML tags that are sensitive to the new line character ("/n") or the carriage return, line feed, such as the <textarea> tag.

The write() and writeln() methods can embed any HTML tag for formatting the text string. Notice in the first example that the embedded tag has an attribute. The values for attributes must be enclosed in single quotation marks, not double quotation marks, because double quotation marks are used to enclose the entire string content.

To display the contents of a variable as part of a text string, JavaScript code can be written to concatenate the text and the variable. **Concatenate** means to join or link together. The symbol for concatenation is the plus sign (+). Table 9–16 shows the code to write a simple welcome message that concatenates text with the values stored in several variables.

Table 9–16 Code to Write a Message to the Web Page

Line	Code
55	`document.write("<p style='margin-left:10%; font-family:Arial, sans-serif; font-weight:bold; font-size:14'>Today is "+weekDay+" "+monthDate+" "+year+", that leaves only "+ daysToSpring + " days until the start of spring.")`
56	`document.write(" Spring is the time to prepare your landscape for new growth of flowers and lawns. ")`
57	`document.write("  Call us now at (221) 555-9100 for a free consultation and free estimate.</p>")`

In line 55, the write() method is used to write data directly to a Web page. The text message string encloses the HTML tags in quotation marks within the text message string. The plus sign (+) concatenates the three variables weekDay, monthDate, and year together, along with the text message and variable daysToSpring. Placement of quotation marks is important and you must include closing HTML tags. Lines 56 and 57 complete the message.

To Write Text and Variable Values to a Web Page

The following step illustrates how to display text and variables using the write() method.

1

- Click line 55, if necessary.
- Enter the JavaScript code shown in Table 9–16 to concatenate the message text with the stored values. Press the ENTER key only at the end of each complete line of code.
- Press the ENTER key an additional time after line 57 (Figure 9–9).

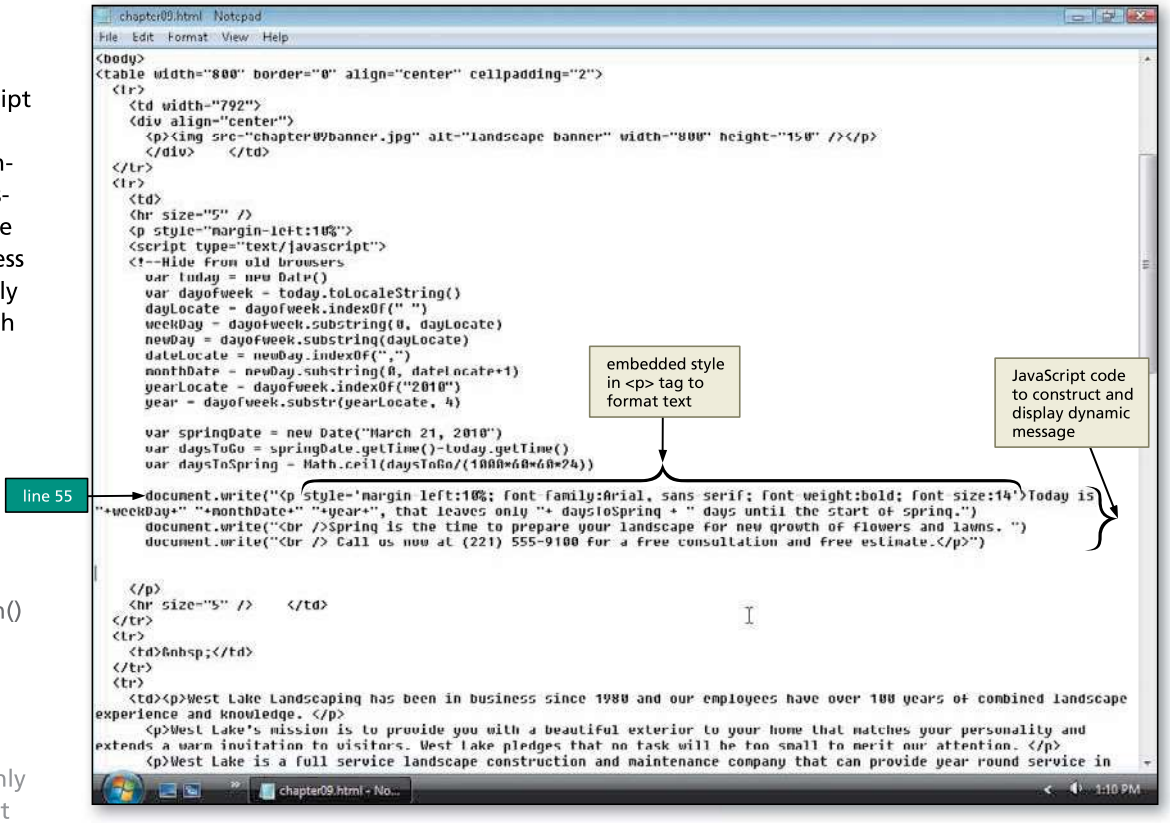


Figure 9–9

Q&A Why does writeln() not place a new line in an HTML document?

The writeln() method works only in HTML tags that either are sensitive to the new line character ("\n") or the carriage return, line feed, such as the <textarea> tag.

Q&A How does JavaScript know to concatenate instead of add when it sees the plus sign (+)?

When the JavaScript interpreter identifies the values surrounding a plus sign (+) and they are not numeric values, it will attempt to join them together.

BTW

HTML Comments within JavaScript

Within a <script> section, an HTML comment often is added to hide the JavaScript from old browsers. This comment is a tag that begins with <!-- and ends with //-->. If you fail to close the HTML comment properly, the remainder of your Web page will not be visible.

Completing the JavaScript Section

As you know, all HTML tags must have start and end tags to separate them from other page elements. To complete this section of JavaScript code, it is necessary to add the end comment tag and the end </script> tag. Table 9–17 shows the code used to close the start <script> tag on line 39 and the comment tag on line 40, as entered in Figure 9–4 on page HTML 393.

Table 9–17 Closing the Script Section

Line	Code
59	//-->
60	</script>

Line 59 ends the comment, <!--Hide from old browsers, that was started on line 39. The end </script> tag on line 60 ends the JavaScript code section and prevents the HTML code that follows from being interpreted as JavaScript code.

To Enter the End Comment and </script> Tags

The following step shows how to enter the end comment tag and the end </script> tag.

1. If necessary, click blank line 59.
2. Enter the JavaScript code shown in Table 9–17 and do not press the ENTER key after the last line (Figure 9–10).

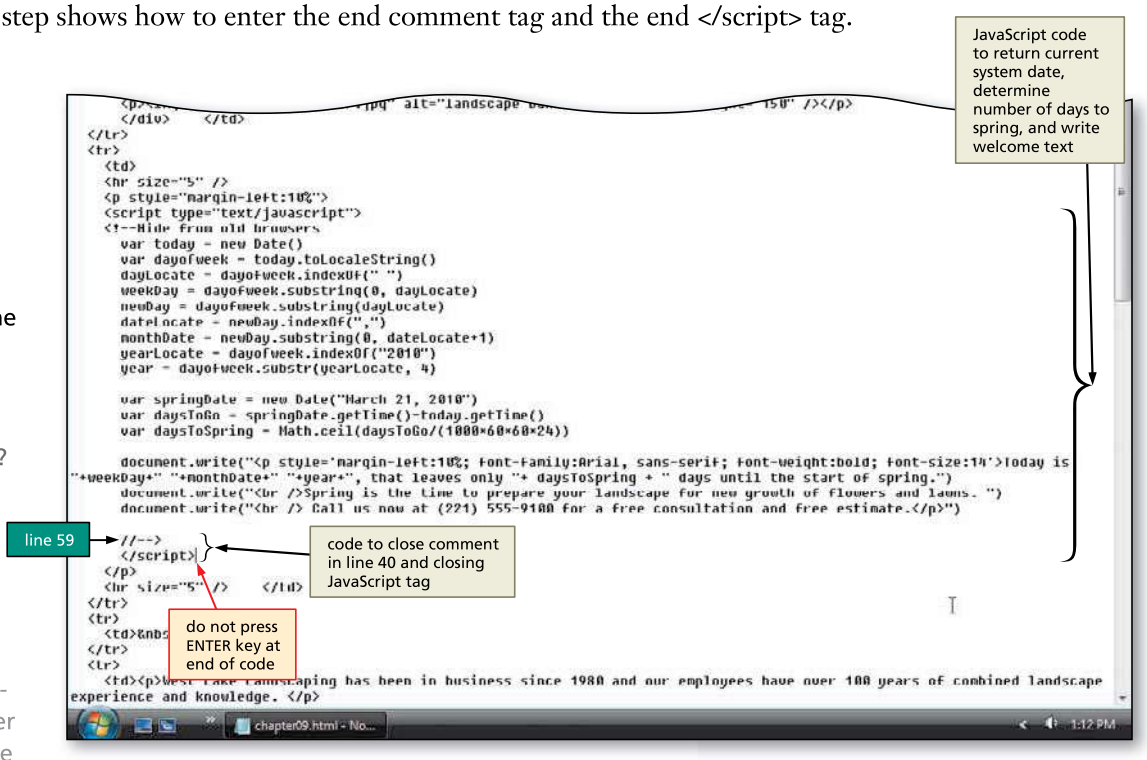


Figure 9–10

Q&A

What happens if a <script> section is not closed properly?

If the HTML comment line is used and it is not closed properly, the rest of the Web page document will be treated as a comment. When the user attempts to view the Web page, nothing will appear after that comment line.

Q&A

Why is the comment needed in <script> sections?

Some browsers or mobile Web devices do not interpret JavaScript code correctly, so the comment hides the JavaScript code. If the comment line is not closed properly, the Web page may not display from the point of the comment forward, thus giving the impression the Web page is blank. If this occurs, always check to ensure the end comment tag was included.

Saving the HTML File and Testing the Web Page

With the first section of JavaScript code for the West Lake Landscaping Web page complete, you should save the file and test the Web page in a browser.

To Save an HTML File

1

- With a USB flash drive connected to one of the computer's USB ports, click File on the Notepad menu bar and then click Save As. Type `chapter09westlake.html` in the File name text box (do not press the ENTER key).
- If Computer is not displayed in the Favorite Links section, drag the top or bottom edge of the Save As dialog box until Computer is displayed.
- Click Computer in the Favorite Links section to display a list of available drives.
- If necessary, scroll until UDISK 2.0 (G:) appears in the list of available drives.

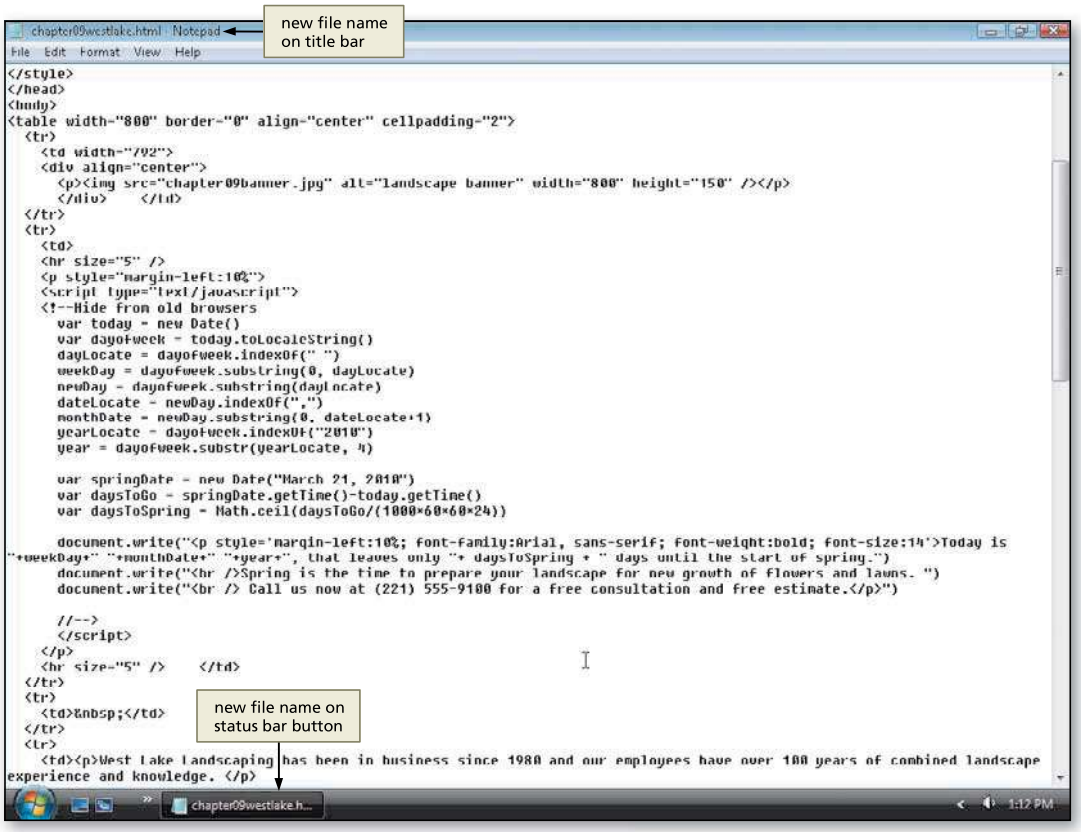


Figure 9–11

- If necessary, open the Chapter09\ChapterFiles folder.
- Click the Save button in the Save As dialog box to save the file on the USB flash drive with the name chapter09westlake.html (Figure 9–11).

To Test the JavaScript on a Web Page

Once you complete your JavaScript code, you should test the code in a browser. The step on the next page illustrates how to open a browser, such as Internet Explorer, and load the chapter09westlake.html Web page to test if the JavaScript works correctly.

1

- Click the Start button on the Windows Vista taskbar to display the Start menu.
- Click Internet Explorer (or another browser icon) on the Start menu. If necessary, click the Maximize button to maximize the browser window.
- Click the Address bar to select the URL on the Address bar.
- Type `g:\Chapter09\ChapterFiles\chapter09westlake.html` in the Address box.
- Press ENTER to display the Web page (Figure 9–12). If a security message appears, read and follow the instructions in the following Q&A.

Q&A When I started my Web page in Internet Explorer, a bar displayed across the top that said, "To help protect your security, Internet Explorer has restricted this webpage from running scripts or ActiveX controls that could access your computer. Click here for options…" What should I do?

For the Web page you just created, you can click that bar, and then click the Allow Blocked Content option that displays and then click Yes in the confirmation dialog box. With other Web pages that you are not familiar with, however, it is not advisable to let them run.

Q&A What should I do if the Web page is not displayed properly?

If your Web page is not displayed correctly, close any error message and then click the Notepad button on the taskbar. Check your JavaScript code according to Figures 9–3 through 9–10 on pages HTML 393 through HTML 404. Correct any errors, save the file, click the Internet Explorer taskbar button to activate the browser, and then click the Refresh button on the Standard Buttons toolbar.

Figure 9–12

Displaying the Last Modified Document Date

Most Web developers agree that a Web page should display the date the Web page was last modified, so visitors are aware of how current the Web page content is. For the West Lake Landscaping Web page, the date last modified should appear at the bottom of the page in a smaller font to keep the message from distracting the user (see Figure 9–1 on page HTML 387).

Plan Ahead

Writing the date last modified on the Web page.
The purpose of displaying the date a Web page was last modified is to make sure the user knows how current the information is contained within the Web page. When writing this content you should follow these guidelines:

- Determine what your message will be. Many Web pages display a copyright notice with the date last modified.
- Create a text message string enclosed in quotation marks and include the document.lastModified property with the text string.
- To write directly to the Web page requires the use of the document write() method written in its own <script> section before the closing </body> tag.

JavaScript provides an easy way to display the date by using the lastModified property of the document object. The lastModified property displays the date in the form of mm/dd/yyyy followed by the time in the form of hh:mm:ss. Table 9–18 shows the general form of the lastModified property of the document.

BTW

Using document.lastModified
The lastModified property might return different values than expected with older browsers and unknown Web servers.

Table 9–18 lastModified Property

General form:	document.lastModified
Comment:	where lastModified is a property of the document object that returns the date the document was last saved.
Example:	document.write("This Web page was last modified " + document.lastModified)

Table 9–19 shows the JavaScript code to create a <script> section to display the date and time the document was last modified and a copyright message.

Table 9–19 Code to Display the lastModified Date

Line	Code
91	`<script type="text/javascript">`
92	`<!--Hide from old browsers`
93	`document.write("<p style='margin-left:10%; font-family:Arial, sans-serif; font-size:xx-small; color:#006600'>")`
94	`document.write("West Lake Landscaping, Copyright 2009-2010. ")`
95	`document.write(" This document was last modified "+document.lastModified+"</p>")`
96	`//-->`
97	`</script>`

Lines 91 and 92 include the required start <script> tag and start comment tag. Lines 93 through 95 are document.write() statements that display the copyright information and the date the document was last modified. Line 94 writes the copyright message and line 95 uses the document.lastModified property with a text string. Lines 96 and 97 close the <script> section.

To Include the Date Last Modified in a Text String

The following step shows how to enter JavaScript code to include the date last modified and a copyright message in a text string.

1

- If necessary, click the Notepad button on the taskbar to activate the Notepad window.
- Click blank line 91 below the second   paragraph tag after the closing </table> tag.
- Enter the JavaScript code shown in Table 9–19 on the previous page. Press the ENTER key after each line but not after the last </script> line (Figure 9–13).

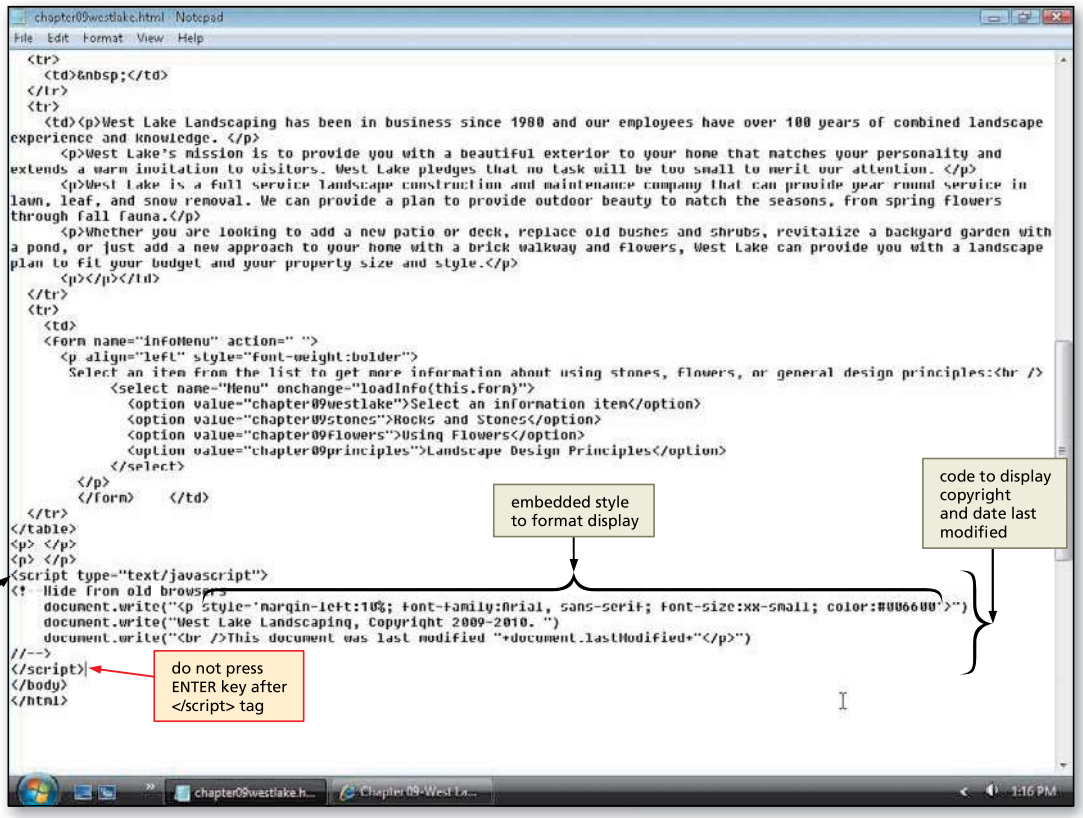


Figure 9–13

Writing a JavaScript User-Defined Function

As previously discussed, a function is JavaScript code that is written to perform certain tasks repeatedly. Web developers use user-defined functions to perform specific tasks. Functions replace large sets of JavaScript codes that are too large to fit within an HTML attribute. Instead, functions are placed anywhere in a JavaScript section in the HTML code.

Plan Ahead

Writing user-defined functions.
User-defined functions are normally written in the <head> section so that this code is loaded before the remainder of the Web page. The user-defined functions in the West Lake Landscaping Web page do the following:

- Change the color of the scroll bar to match the Web page colors.
- Use a <select> list as a drop-down menu to link to other Web pages.

The code in the user-defined function in the <head> section is not executed until a JavaScript statement calls the function. To **call** a function means to have JavaScript execute the function. The general form of a user-defined function is shown in Table 9–20.

Table 9–20 User-Defined Functions

General form:	function functionName(optional parameters) {
	JavaScript Code
	}

Comment:	where functionName is the name of the user-defined function, the optional parameters represent values or properties passed to the function that will be used by the function in the JavaScript code. JavaScript code is the statements that execute.

Examples:	function showBrowserName() {
	alert("You are using " +navigator.appname)
	}
	
	function getSum(myform) {
	var sum= document.Calculator.Amount1.value+document.Calculator.Amount2.value
	}

The naming conventions for user-defined functions are similar to those for variables. A function name must begin with a letter, may contain numerals and an underscore, but may not contain any spaces, punctuation (such as periods or commas), or reserved words. Table 9–21 shows valid and invalid user-defined function names. Values or parameters are passed to the function by placing a variable name between the parentheses.

BTW

Placing JavaScript Functions
Always place your JavaScript function definitions in the <head> section to ensure they are loaded completely before they are called.

Table 9–21 Valid Function Names

Valid Function Names	Invalid Function Names	Reason
verifyForm()	3Ddisplay()	Starts with a number
get_Cookie()	make.cookie()	No periods allowed
calcPayment()	calc payment()	No spaces allowed
popWind()	pop-upWindow()	No hyphens allowed

Changing the Color of the Browser Scroll Bar

Plan Ahead

Changing the color of the browser scroll bar.
Because cascading style sheets (CSS) does not have an official standard style for changing colors of the browser scroll bar, the scroll bar color can be changed with a JavaScript user-defined function. To change the scroll bar color, follow these guidelines:

- JavaScript must have access to the object (the scroll bar). Use the getElementsByTagName() method of the document object to assign the "HTML" object to a variable. The getElementsByTagName() method returns an array of elements belonging to the identified object and all associated properties to that object.
- Using the variable as an object, JavaScript can set values to the various scroll bar properties: FaceColor, ArrowColor, HighlightColor, 3DlightColor, DarkshadowColor, TrackColor, and ShadowColor.
- Assign a color that matches or compliments the colors on the Web page to at least the FaceColor and TrackColor.

Table 9–22 describes the general form of the JavaScript getElementsByTagName() method. For more information about the getElementsBy methods, see the JavaScript Quick Reference in Appendix E.

Table 9–22 getElementsByTagName()

General form:	getElementsByTagName('html')
Comments:	where getElementsByTagName() is a method of the document object and 'html' is the object to be returned. Tag element must be entered as a string in single quotation marks and is case sensitive. The returning values are returned in an array format, so that each element can be referenced individually by an array value. The returned value also can use any properties associated with that value. The example shows the method to create an object of the html tag styles named styleObject.
Example:	styleObject=document.getElementsByTagName('html')[0].style

To modify the colors of the scroll bar, use the styleObject object name with the standard scroll bar properties: FaceColor, ArrowColor, HighlightColor, 3DlightColor, DarkshadowColor, TrackColor, and ShadowColor. For example, to change the FaceColor of the scroll bar write

```
styleObject.scrollbarFaceColor="#006600"
```

where the color must be written as a standard color name, a hexadecimal value, or using the rgb() values method.

On the West Lake Landscaping Web page, the scroll bar colors should be changed so the scroll bar appears in green and the scroll bar track appears in light green. Table 9–23 shows the code to change the scroll bar colors.

BTW

Color Values
For the most flexibility in using colors, Web developers suggest using either the hexadecimal version or the rgb() method version to assign a color. Be careful in using a standard color name. Color names like "lightblueaqua" might not be recognized by the style property.

Table 9–23 Code to Change the Browser Scroll Bar Color

Line	Code
6	`<script type="text/javascript">`
7	`<!--Hide from old browsers`
8	`function scrollColor() {`
9	`  styleObject=document.getElementsByTagName('html')[0].style`
10	`  styleObject.scrollbarFaceColor="#006600"`
11	`  styleObject.scrollbarTrackColor="#00aa00"`
12	`}`

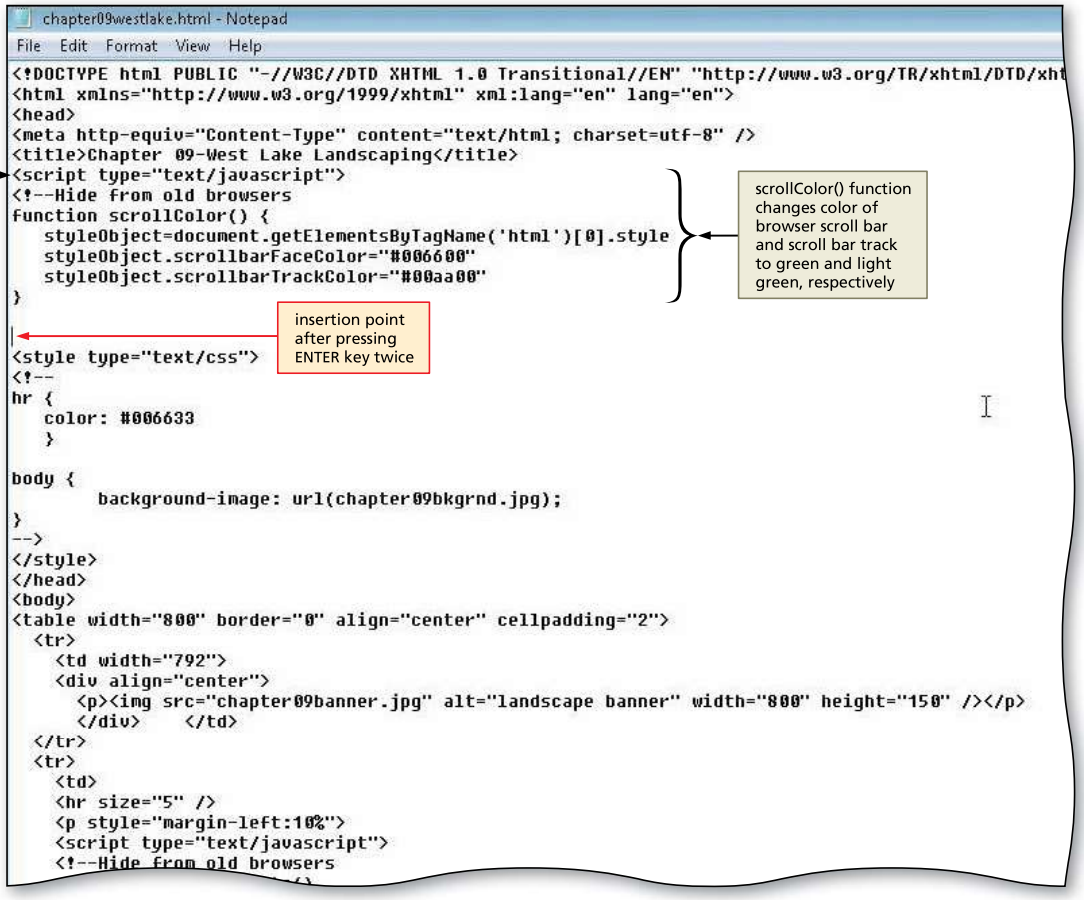
Lines 6 and 7 include the required start <script> tag and start comment tag. Line 8 defines the function name as scrollColor(). Line 9 defines the style property of the <html> tag and assigns it to an object, so face color and track color can be changed in lines 10 and 11. Line 10 changes the color of the scroll bar face to green using a hexadecimal value. Line 11 changes the scroll bar track color to light green using a hexadecimal value. Line 12 closes the scrollColor() function.

To Enter User-defined Functions in the <head> Section

The following step illustrates how to enter the user-defined function to change the browser scroll bar color in the <head> section.

1

- If necessary, click the chapter09westlake.html – Notepad icon on the taskbar.
- Click blank line 6 directly below the <title> tags.
- Enter the JavaScript code shown in Table 9–23.
- Press the ENTER key twice after the last } to leave a blank line between user-defined functions (Figure 9–14).




```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml/DTD/xht
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Chapter 09-West Lake Landscaping</title>
<script type="text/javascript">
<!--Hide from old browsers
function scrollColor() {
   styleObject=document.getElementsByTagName('html')[0].style
   styleObject.scrollbarFaceColor="#006600"
   styleObject.scrollbarTrackColor="#00aa00"
}

<style type="text/css">
<!--
hr {
   color: #006633
   }

body {
        background-image: url(chapter09bkgrnd.jpg);
}
-->
</style>
</head>
<body>
<table width="800" border="0" align="center" cellpadding="2">
  <tr>
    <td width="792">
    <div align="center">
      <p><img src="chapter09banner.jpg" alt="landscape banner" width="800" height="150" /></p>
      </div>    </td>
  </tr>
  <tr>
    <td>
    <hr size="5" />
    <p style="margin-left:10%">
    <script type="text/javascript">
    <!--Hide from old browsers
```

Figure 9–14

Using the Location Object and selectedIndex Property to Link to a New URL

As shown in Figure 9–1 on page HTML 387, the West Lake Landscaping Web page also includes a drop-down list object that allows users to select items from a drop-down menu. Depending on the item selected in the select list, the code will link users to one of three Web pages containing information about using rocks and stones, landscaping with flowers, or landscape design principles.

Plan Ahead

Using the select list as a drop-down menu.
To use a <select> list as a drop-down menu, a user-defined function must make use of the following:

- The window's location property (window.location), to which a URL can be assigned, and which changes the location of the Web page in the browser.
- The selectedIndex property to identify which item was selected in the drop-down menu list.

BTW

The selectedIndex Property
The selectedIndex property cannot be used alone: it must be used in full reference to the particular select list, the form it belongs to, and its options.

When a user selects an item in the select list, the **selectedIndex** property of the select list returns the value of the selected item. The selectedIndex values of the items in a select list are considered to be numbered, starting with zero for the first item. The second item is number one, and so on. The value returned by the selectedIndex property is an integer, starting with zero for the first item in the list. Table 9–24 shows the general form of the selectedIndex property.

Table 9–24 selectedIndex Property

General form:	var varname=formName.SelectListName.selectedIndex
Comments:	where varname is a variable, formName is the identifier of the form that holds the select list, SelectListName is the name of the select list, and selectedIndex is a property that returns an integer corresponding to the position of the item in the list.
Example:	var menuSelect=myForm.Menu.selectedIndex

As you learned in Chapter 7, the text that appears for each item in a select list is enclosed in <option> tags. The option tag also supports a value attribute, as shown in the code in Figure 9–15. The value in the value attribute describes the item and can be assigned to a variable.

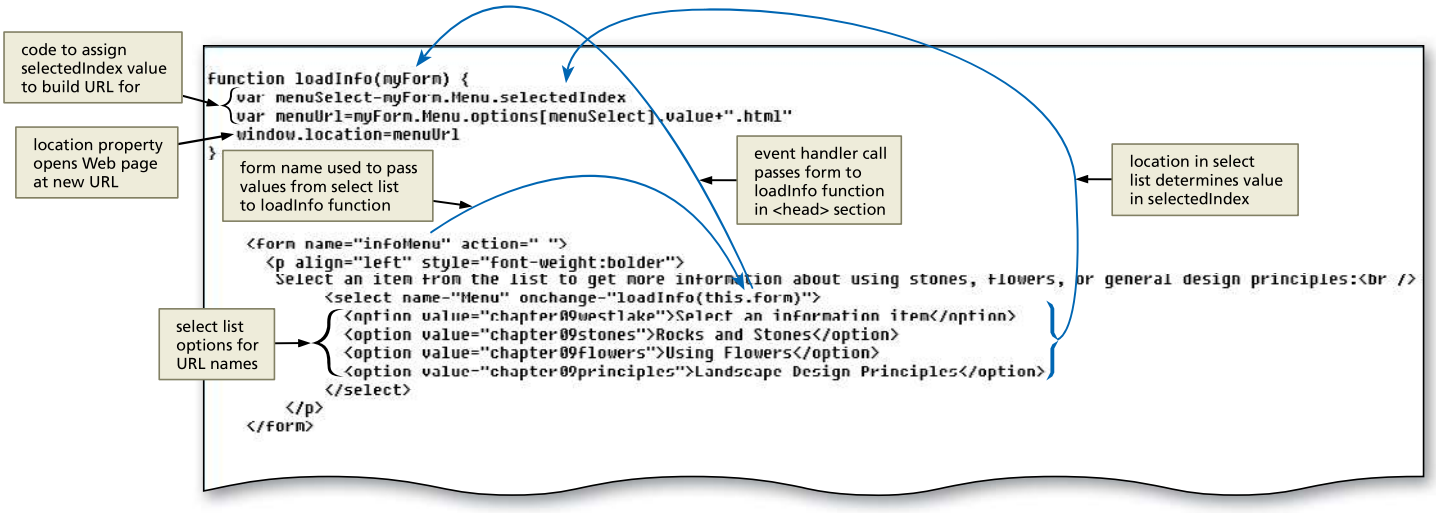


Figure 9–15

This variable can then be used to assign the new Web page location to the window's location property. This statement will load a new URL into the browser. Table 9–25 shows the general form of the location property.

Table 9–25 Location Property

General form:	object=window.location or window.location=URL
Comments:	where object is a variable or some other object that can display the URL of the current window, and URL is the address of the Web page to display. The use of the window object is optional.
Examples:	myform.textbox.value=window.location window.location="http://www.scsite.com"

Table 9–26 shows the JavaScript code for a loadInfo() function that uses the selectedIndex value to determine which item in a list was selected, assigns the value attribute for that item, and then uses that variable to create a URL.

Table 9–26 Code to Change Location

Line	Code
14	`function loadInfo(myForm) {`
15	`   var menuSelect=myForm.Menu.selectedIndex`
16	`   var menuUrl=myForm.Menu.options[menuSelect].value+".html"`
17	`   window.location=menuUrl`
18	`}`
19	
20	`//-->`
21	`</script>`

The selectedIndex property then is used on the object name of the select list and the form. Line 14 defines the function name. The form object, myForm, is passed to the function from the select list name attribute. In line 15, the selectedIndex statement assigns the numerical value of the item selected from the list to the variable menuSelect. In this line, menuSelect is a variable name, myForm is the identifier of the form that holds the select list, and Menu is the name of the select list. The options property of Menu refers to the <option> tag in the select list, while selectedIndex indicates the integer value of the item selected in the select list. Line 16 concatenates the value attribute of the selected item (menuSelect) with the file extension .html to create a URL. The URL name is concatenated to the .html file name extension using the plus sign (+). Line 17 uses that URL to load that Web page into the browser window. Line 18 closes the function. Lines 20 and 21 close the <script> section for these two user-defined functions.

To Enter the User-defined Function to Link to a New URL using the Drop-Down Menu List

The following step illustrates how to enter the user-defined function to link to a new URL using the drop-down menu.

1

- Click line 14 if necessary.
- Enter the JavaScript code shown in Table 9–26 to enter the options and links for the drop-down menu list.
- Do not press the ENTER key after the last line (Figure 9–16).

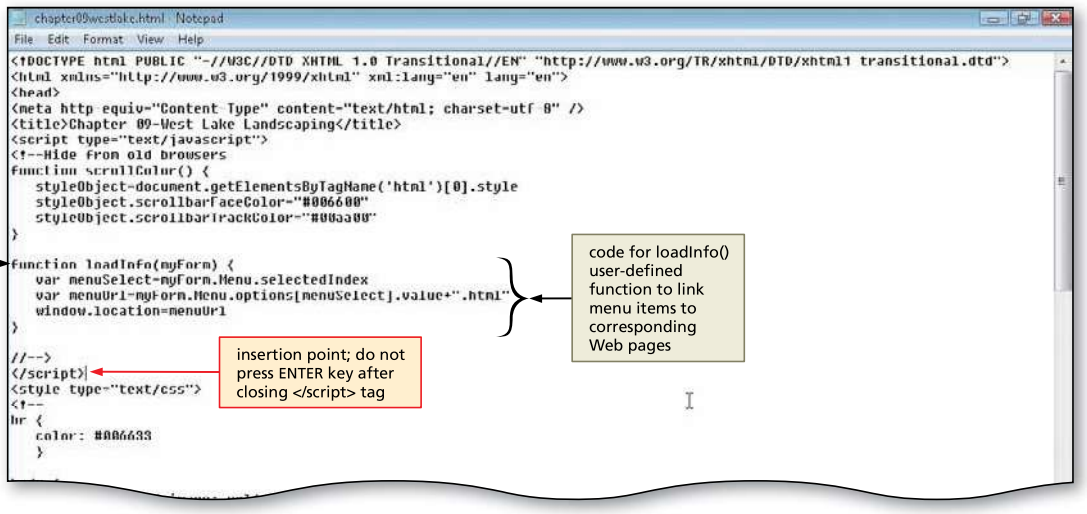


Figure 9–16

Calling JavaScript Functions Using Event Handlers

Now that you have added user-defined functions to change the scroll bar color and create a drop-down menu list, you need to add code that tells when these functions are to be called. JavaScript has two basic methods to call functions. One method to call a function is to use event handlers and object methods. The other method is to code the function name in a JavaScript section at the logical point of execution. The user-defined functions written in this chapter execute using event handlers.

Plan Ahead

Using event handlers to call user-defined functions.
Event handlers must be placed with the object (such as a button, drop-down list, or HTML tag) that controls the event. In this chapter, the events are load and change, so you use the event handlers onload and onchange to call the user-defined functions. In this chapter, you will:

- Place the onload event handler in the <body> tag.
- Place the onchange event handler in the <select> tag that starts the drop-down menu list.

As you have learned, an event is the result of an action, such as a mouse click or a window loading. An event handler is a way to associate that action with a function. For example, when a user clicks a button or a check box, a JavaScript user-defined function may be associated with that event. The associated function will execute if the event is captured and then **triggers**, or calls, the JavaScript user-defined function. The general form of an event handler is shown in Table 9–27.

Table 9–27 Event Handlers

General form:	<tag attribute eventhandler="JavaScript code">
Comment:	where tag is the HTML tag; attribute is a property of the tag that can have a value assigned to it, eventhandler is the name of the JavaScript event handler, and JavaScript code is the instruction to execute, usually in the form of a function name.
Example:	<body onload="scrollColor()">

JavaScript event handlers make Web pages more dynamic and interactive by allowing JavaScript code to execute only in response to a user action, such as a mouse click or selection of an item in a list. For a complete list of event handlers, see the JavaScript Quick Reference in Appendix E.

To Associate a User-defined Function with the onload Event

An event handler not directly associated with a user action is the onload event. The onload event triggers the associated function when the Web page has completed loading into the browser. The following steps illustrate how to enter JavaScript code to associate the scrollColor() user-defined function with the onload event.

1. Click to the right of the y in the <body> tag in line 34, as shown in Figure 9–17.

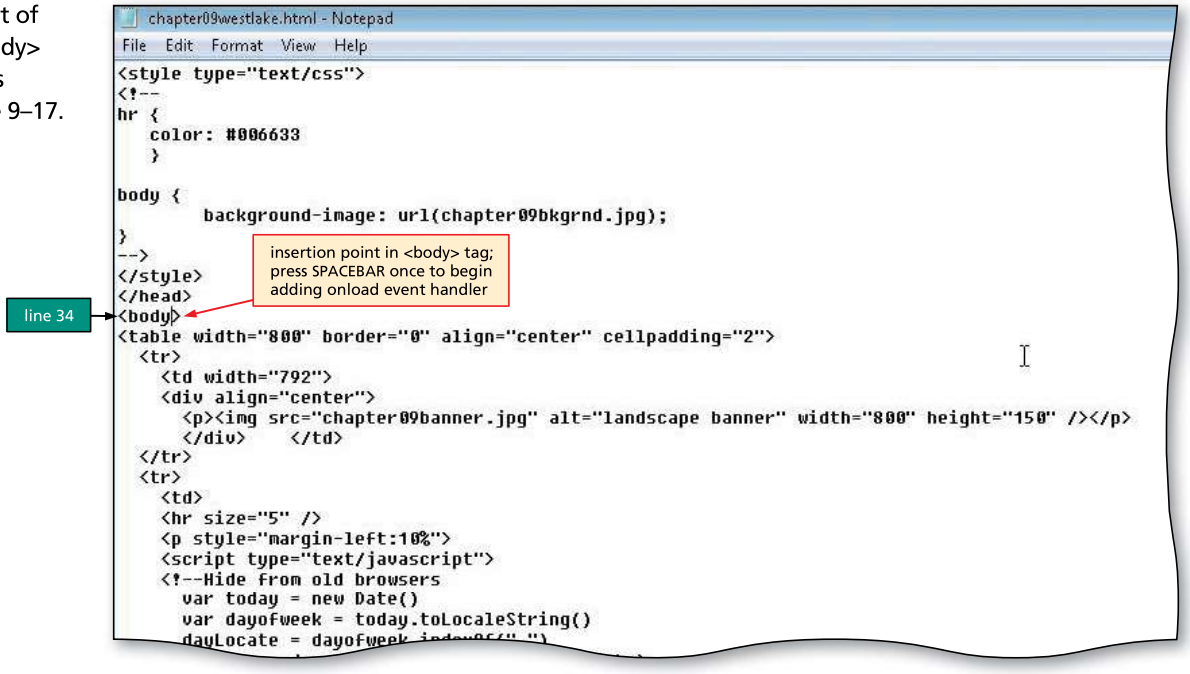


Figure 9–17

2. Press the SPACEBAR once and then type `onload="scrollColor()"` within the <body> tag. Do not press the ENTER key (Figure 9–18).

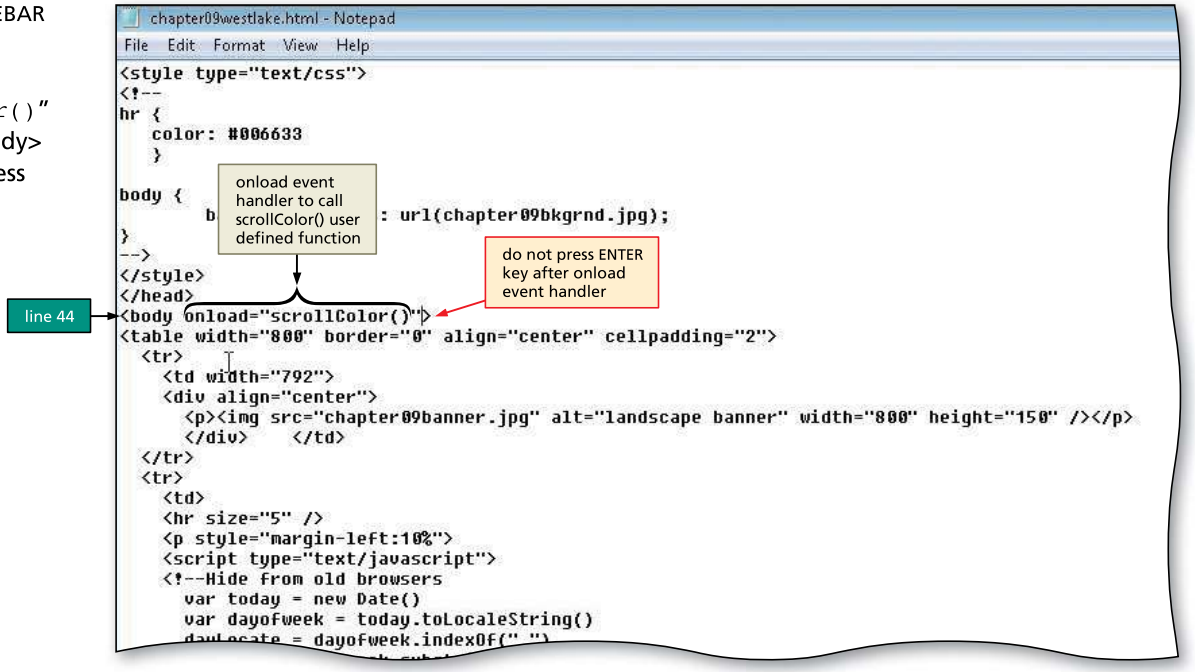


Figure 9–18

To Associate a User-defined Function with the Select List

The West Lake Landscaping Web page also uses the onchange event handler. The onchange event handler is triggered when the value of an object changes. For example, when the user selects a list item in the select list, the value of the select list is changed. This change triggers the associated user-defined function loadInfo(). The following step illustrates how to associate the user-defined function loadInfo() with the select list.

1

- Click to the right of "Menu" in line 85.
- Press the SPACEBAR once and then type `onchange="loadInfo(this.form)"` within the <select> tag. Do not press the ENTER key (Figure 9–19).

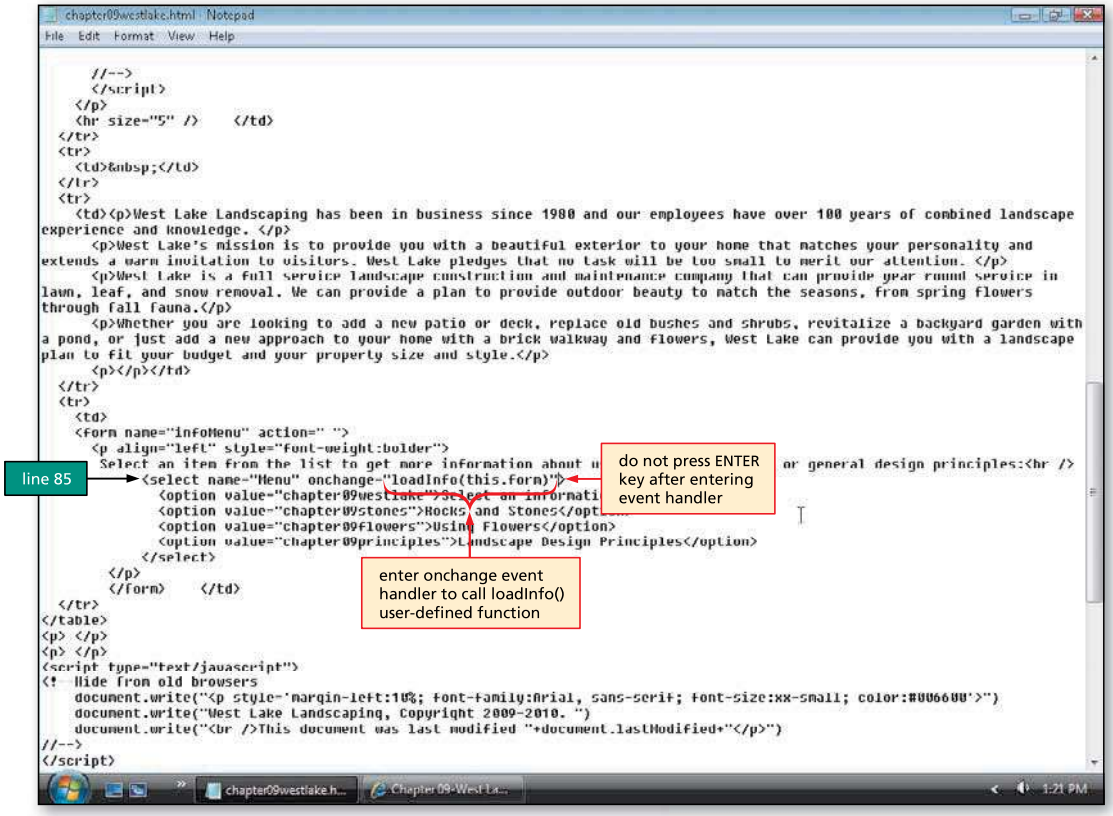


Figure 9–19

To Save an HTML File and View and Test the Completed Web Page

With the code for the West Lake Landscaping Web page complete, you should save the HTML file and view the Web page in a browser to confirm the Web page is displayed and functioning as desired. The following step shows how to save an HTML file and then view and test the Web page in a browser.

1

- With the USB drive plugged into your computer, click File on the menu bar.
- Click Save on the File menu.
- Click the chapter 09-WestLake Landscaping – Windows Internet Explorer button on the taskbar.
- Click the Refresh button on the Standard Buttons toolbar.
- Select Rocks and Stones in the drop-down menu list (Figure 9–20a) to display the Rocks and Stones page.
- Click the Back button on the Standard Buttons toolbar to return to the West Lake Landscaping page.

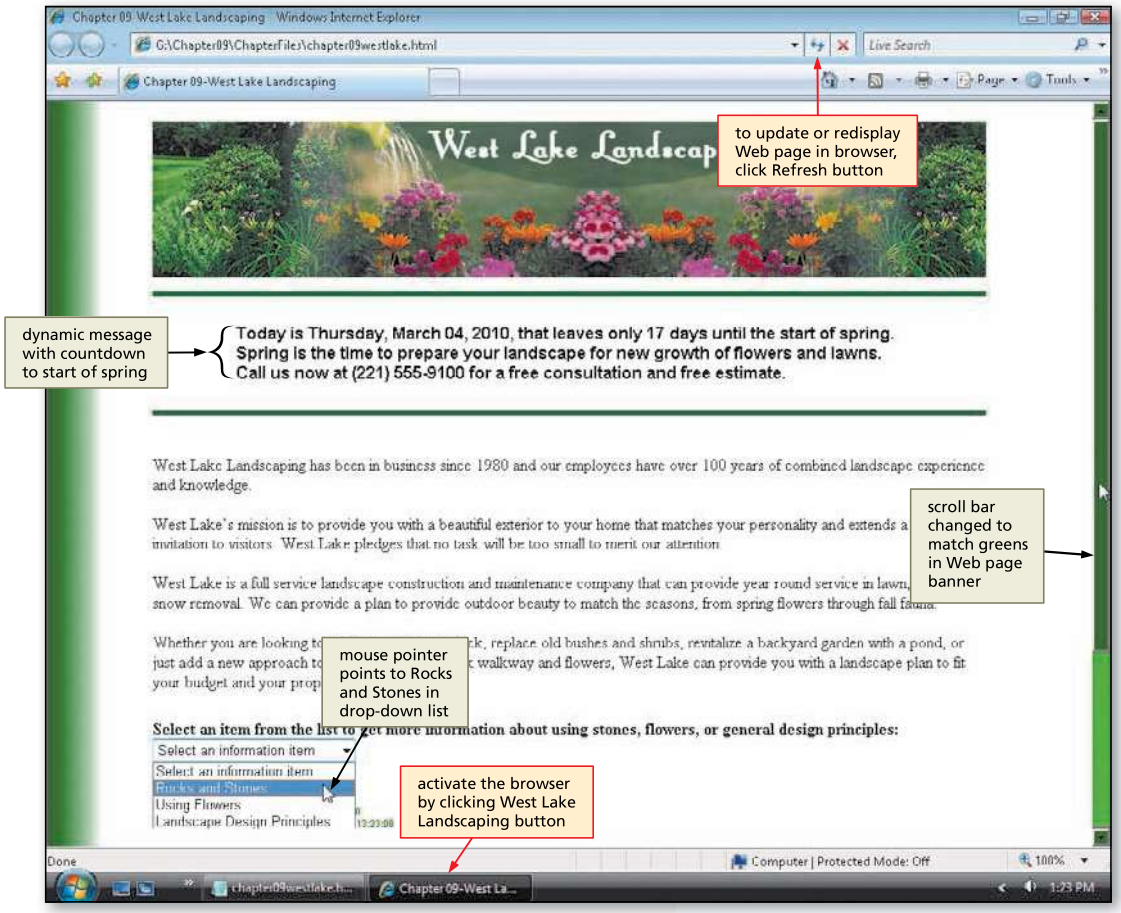


Figure 9–20a

- Select Using Flowers in the drop-down menu list (Figure 9–20b) to display the Use of Flowers in Landscape Design page.
- Click the Home link at the bottom of the page to return to the West Lake Landscaping home page, and select Landscape Design Principles from the drop-down menu list to display the Design Principles page.
- Click the Home link at the bottom of the page to return to the West Lake Landscaping home page.

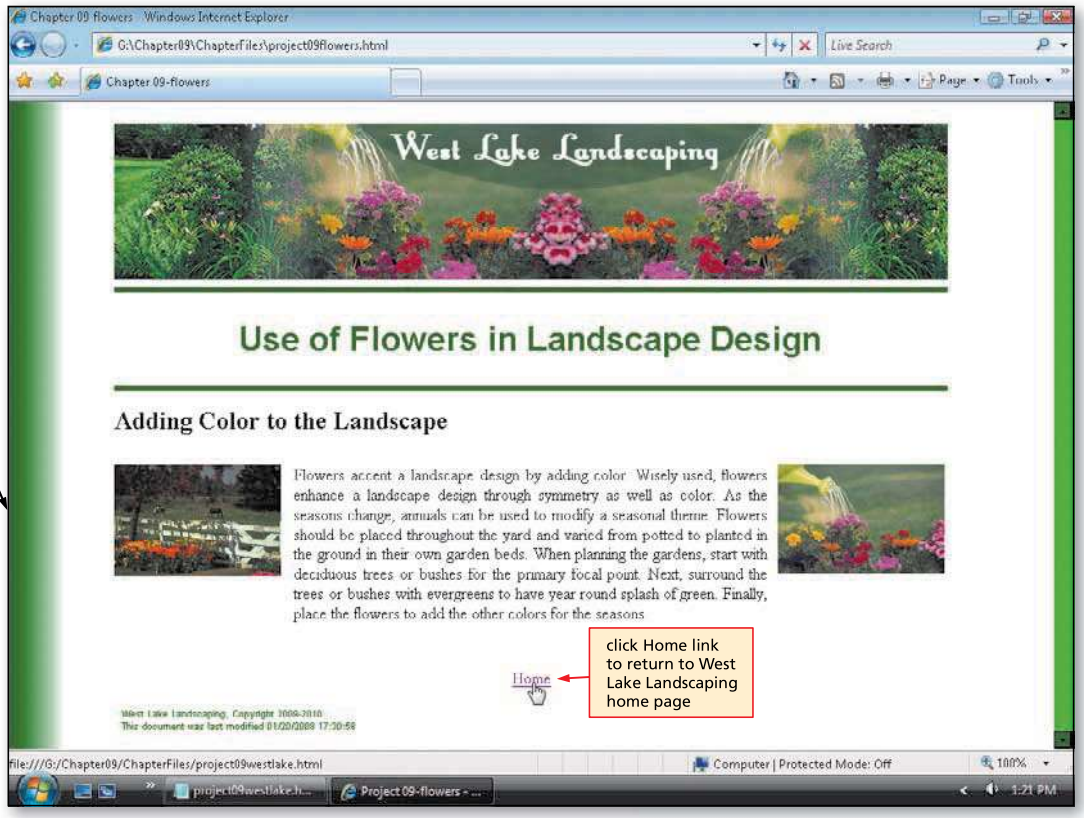


Figure 9–20b

Q&A

Is there any way to display all the errors on a Web page at once?

Internet Explorer does not offer this feature. If the JavaScript code is missing periods, is missing quotation marks, or has misspelled words, the Web page displays with errors. To continue loading the Web page, click the OK button in the dialog box. The browser will cease to process any more JavaScript code, but will load what it can of the Web page. After you fix the errors, refresh the Web page to see if any other errors are found.

Other Ways

1. In Internet Explorer, refresh the page by clicking Refresh on the View menu or pressing F5.
2. In Mozilla Firefox, click Reload current page button on Navigation Toolbar, click Reload on the View menu, or press CTRL+R.

To Validate a Web Page

Now that you have tested the Web page and made sure the JavaScript works as desired, you should validate the page at the w3.org Web site. The following step illustrates how to validate an HTML file.

1

- Open Internet Explorer and navigate to the Web site `validator.w3.org`.
- Click the Validate by File Upload tab.
- Click the Browse button.
- Locate the chapter09westlake.html file on your storage device and click the file name.
- Click the Open button on the Choose file dialog box and the file name will be inserted into the File box.
- Click the Check button.

To Print an HTML File

After completing and testing the Web page, you should print the HTML file using Notepad for future reference. The following step shows how to print the chapter09westlake.html file using Notepad.

1

- If necessary, click the chapter09westlake.html - Notepad button on the taskbar.
- Click File on the menu bar and then click Print. Click the Print button in the Print dialog box.

To Quit Notepad and a Browser

1

- Click the Close button on the browser title bar.
- Click the Close button on the Notepad window title bar.

Chapter Summary

In this chapter, you learned basic JavaScript concepts and how to write and insert JavaScript code to make your Web page more dynamic and interactive. The items listed below include all the new HTML and JavaScript skills you have learned in this chapter.

1. Enter the Start <script> and Comment Tags (HTML 393)
2. Extract the Current System Date Using the Date() Object (HTML 397)
3. Create a Date() Object Instance to Store a Future Date (HTML 399)
4. Calculate Milliseconds Between Two Dates Using the getTime() Method (HTML 400)
5. Convert Milliseconds to Days and Round Up Using the ceil() Method (HTML 401)
6. Write Text and Variable Values to a Web Page (HTML 403)
7. Enter the End Comment and </script> Tags (HTML 404)
8. Test the JavaScript on a Web Page (HTML 405)
9. Include the Date Last Modified in a Text String (HTML 408)
10. Enter User-defined Functions in the <head> Section (HTML 411)
11. Enter the User-defined Function to Link to a New URL using the Drop-Down Menu List (HTML 413)
12. Associate a User-defined Function with the onload Event (HTML 414)
13. Associate a User-defined Function with the Select List (HTML 416)

BTW

Quick Reference
For a list of JavaScript statements and their associated attributes, see the JavaScript Quick Reference (Appendix E) at the back of this book, or visit the HTML Quick Reference Web page (scsite.com/HTML5e/qr).

Learn It Online

Test your knowledge of chapter content and key terms.

Instructions: To complete the Learn It Online exercises, start your browser, click the Address bar, and then enter the Web address `scsite.com/html5e/learn`. When the HTML Learn It Online page is displayed, click the link for the exercise you want to complete and read the instructions.

Chapter Reinforcement TF, MC, and SA

A series of true/false, multiple choice, and short answer questions that test your knowledge of the chapter content.

Flash Cards

An interactive learning environment where you identify chapter key terms associated with displayed definitions.

Practice Test

A series of multiple choice questions that test your knowledge of chapter content and key terms.

Who Wants To Be a Computer Genius?

An interactive game that challenges your knowledge of chapter content in the style of a television quiz show.

Wheel of Terms

An interactive game that challenges your knowledge of chapter key terms in the style of the television show, *Wheel of Fortune*.

Crossword Puzzle Challenge

A crossword puzzle that challenges your knowledge of key terms presented in the chapter.

Apply Your Knowledge

Reinforce the skills and apply the concepts you learned in this chapter.

Adding User-Defined Functions

Instructions: Start Notepad. Open the file apply9-1.html from the Chapter09\Apply folder of the Data Files for Students. See the inside back cover of this book for instructions on downloading the Data Files for Students, or contact your instructor for information about accessing the required files.

The apply9-1.html file is a partially completed HTML file that you will use for this exercise. Figure 9–21 shows the Apply Your Knowledge Web page as it should be displayed in a browser after the JavaScript has been added. This problem requires changing the scroll bar color with a user-defined function, using JavaScript to display a dynamic message, using JavaScript to display a copyright and date last modified at the bottom of the Web page, and adding an event handler in the <body> to activate the user-defined function.

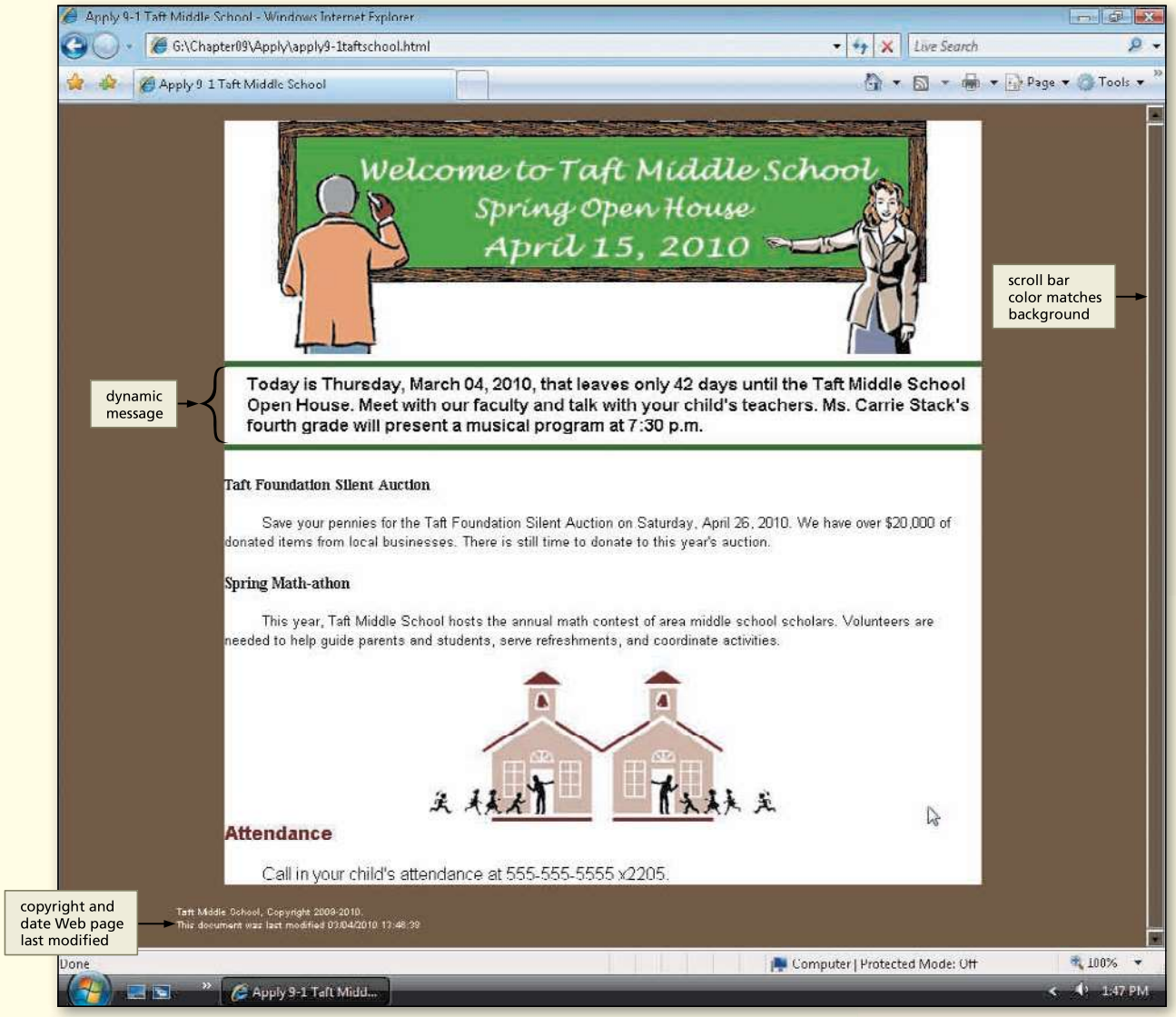


Figure 9–21

Perform the following tasks:

1. Enter the beginning of a JavaScript code section for a user-defined function in the <head> section of the Web page before the <style> tag. Be sure to include a comment line to hide the JavaScript from old browsers.
2. Using the code in Table 9–23 on page HTML 410 as a guide, enter the code for a user-defined function called scrollColor() to change the scroll bar face color and scroll bar track color to the hexadecimal values shown in Table 9–28.

Table 9–28 Scroll Bar Colors

scroll bar face color: #655028
scroll bar track color: #d2af7d

3. Be sure to enter the closing brace for the function, followed by the closing HTML tags to close the <script> section.

Continued >

Apply Your Knowledge *continued*

4. Write code to start a new JavaScript code section at line 50 after the HTML comment line 49: <!--JavaScript code-->.
5. Using the code in Table 9–13, Figures 9–6, 9–7, and 9–8 and Table 9–16 as a guide, enter the JavaScript code to display the countdown message shown in Figure 9–21 on the previous page. Use your own current and future dates for this Web page.
6. Using the code in Table 9–19 as a guide, write a copyright message and the date the document was last modified at the bottom of the Web page, as shown in Figure 9–21.
7. Enter the onload event handler in the <body> tag to call the scrollColor() function.
8. Save the revised file in the chapter09\Apply folder using the file name apply9-1solution.html.
9. Start your browser. Enter the URL g:\Chapter09\Apply\apply9-1solution.html in the address box to view and test the Web page in your browser.
10. If any errors occur, check the code against Steps 1 through 7, make any required changes, save the file using the same file name, and then refresh the Web page in the browser.
11. Submit the revised HTML file and Web page in the format specified by your instructor.

Extend Your Knowledge

Extend the skills you learned in this chapter and experiment with new skills. You will need to search the Internet to complete the assignment.

Learning More about Displaying Messages

Instructions: Start Notepad and your browser. Open the file extend9-1.html from the Chapter09\Extend folder of the Data Files for Students. See the inside back cover of this book for instructions on downloading the Data Files for Students, or contact your instructor for information about accessing the required files.

Perform the following tasks:

1. Search the Internet for the JavaScript instructions on how to display a message on the status bar of your browser. (*Hint*: Look for properties of the Windows object.)
2. Write the code for a user-defined function that assigns the message "Nick's Fitness Center will make your fitness dreams come true." on the status bar.
3. Using the code in Table 9–23 on page HTML 410 as a guide, enter the code for a user-defined function called scrollColor() to change the scroll bar colors to the hexadecimal values shown in Table 9–29.

Table 9–29 Scroll Bar Colors

scroll bar face color: #715a2d	scroll bar track color: #dfcfae
scroll bar arrow color: #d29117	scroll bar highlight color: #000000

4. In the second row of the table in the HTML code, add the code for the dynamic message. Pick a date about 30 days from the current date to use in the calculation. Embed a style in the document write() method that sets the left margin to 5 percent, the font family to Arial, sans-serif, the font-weight to bold, and the font size of 14 point.
5. The message should display similar to the Web page shown in Figure 9–22.
6. Add the onload event handler to the <body> tag to call this function when the Web page loads.

7. Use a JavaScript section to display the copyright and the date last modified for Nick's Fitness Center. Use the substr() or substring() method as discussed in Tables 9–10 and 9–11. Use one of these methods to display only the year portion of the date last modified message.
8. Save the revised file in the Chapter09\Extend folder using the file name extend9-1solution.html.
9. Start your browser. Enter the URL g:\Chapter09\Extend\extend9-1solution.html in the address box to view and test the Web page in your browser.
10. If any errors occur, check the code against Steps 1 through 7, make any required changes, save the file using the same file name, and then refresh the Web page in the browser.
11. Submit the revised HTML file and Web page in the format specified by your instructor.

Figure 9–22

Make It Right

Analyze the JavaScript code on a Web page and correct all errors.

Correcting Syntax Errors and Inserting Missing Code

Instructions: Start your browser. Open the file makeitright9-1.html from the Chapter09\MakeItRight folder of the Data Files for Students. See the inside back cover of this book for instructions for downloading the Data Files for Students, or see your instructor for information on accessing the files required in this book.

Continued >

Make It Right *continued*

The Web page is an announcement for a fall event called Bayfield Days. This Web page has four errors that you are to find and correct.

Perform the following tasks:

1. When you open the makeitright9-1.html file in the browser, you will notice that the scroll bar did not change color and that the dynamic message did not display between the horizontal lines.
2. Save the HTML file in the Chapter09\MakeItRight folder using the file name makeitright9-1solution.html.
3. Compare the code in the user-defined function to the code to change the scroll bar color in Table 9–23 on page HTML 410. Make the changes necessary to change the scroll bar color on the Bayfield Days Web page.
4. Compare the code in Tables 9–13 and 9–16 to the code in the dynamic message in the Bayfield Days Web page. Make corrections as necessary.
5. Make sure the user-defined functions are called properly by the correct event handlers, and that they are in the correct locations.
6. Save the corrected HTML file and test it using your browser. If errors occur, check your code and save again. Your Web page should look similar to Figure 9–23.
7. Submit the revised HTML file and Web page in the format specified by your instructor.

Figure 9–23

In the Lab

Design and/or create a Web page using the guidelines, concepts, and skills presented in this chapter. Labs are listed in order of increasing difficulty.

Lab 1: Creating a Web Page for the College Theater

Problem: You belong to the Huysken College Theater Club and have offered to create a Web site to promote the campus theater. You create the Web page shown in Figure 9–24, which includes changing the color of the scroll bar, using a drop-down menu to link to related Web pages, and add the copyright and date last modified at the bottom of the Web page.

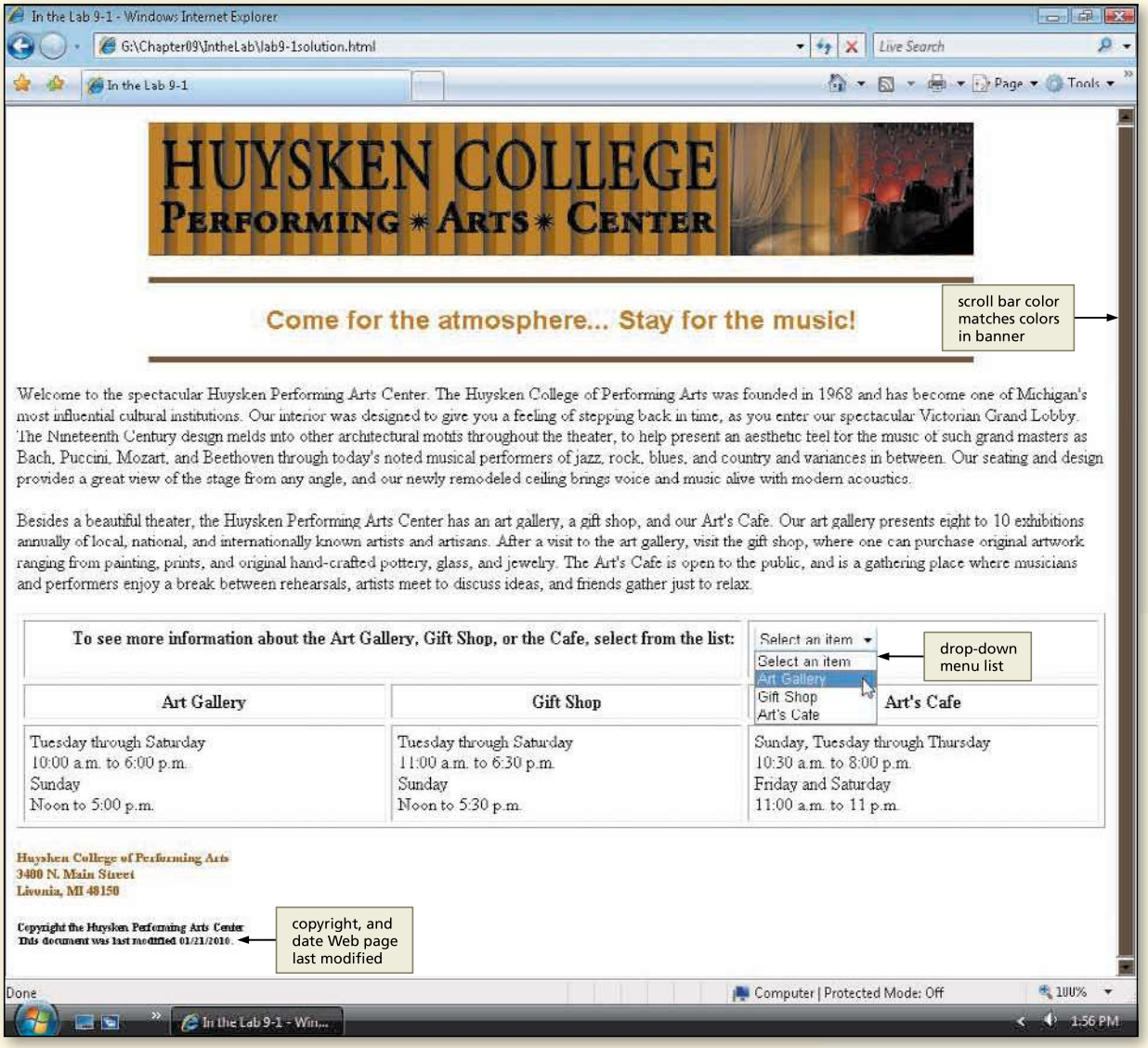


Figure 9–24

Continued >

In the Lab *continued*

Instructions: Perform the following tasks.

1. Start Notepad and open the lab9-1.html file from the Chapter09\IntheLab folder of the Data Files for Students.
2. Save the file as lab9-1solution.html.
3. Start a new <script> section in the <head> section under the <title> tag.
4. Write a JavaScript function called infoMenu() that uses the selectedIndex value of the moreInfo <select> list in the menuInfo <form>. Use the code in Table 9–26 as a guide.
5. Write another JavaScript function to change the scroll bar colors called scrollColor(). Use the code in Table 9–23 as a guide, and use the following color values for the scroll bar face and track colors:

   face color: #604000

   track color: #bf8a20
6. Add the onchange event handler in the <select> tag to call the loadMenu() function. Then, add the onload event handler to the <body> tag to call the scrollColor() user-defined function.
7. At the bottom of the Web page, after the HTML comment <!-- copyright and date last modified --> and before the closing </body> tag, write the JavaScript code to display the copyright and date last modified message as shown in Figure 9–24. Embed a style in the document write() method to set the font family to Arial, Helvetica, sans-serif, and to set the font size to xx-small. On the second line, the date document was last modified should display only the date, not the time, using a substring() method on the lastModified property.
8. Save the completed HTML file and test it using your browser. If an error occurs, check your code and save and test again.
9. Submit the completed HTML file and Web page in the format specified by your instructor.

In the Lab

Lab 2: Mountain Valley Recycling

Problem: You are an intern at Mountain Valley Recycling. Your supervisor knows of your Web page development experience and asks you to modify the company's Web page to include links to other pages of information using a drop-down menu. In addition, your supervisor wants to add a countdown in a dynamic announcement that Mountain Valley Recycling will be at the county fair grounds to collect any household hazardous materials. You suggest adding some color to the scroll bar, and copyright and the date last modified information. You add the JavaScript to make the Web page appear as in Figure 9–25.

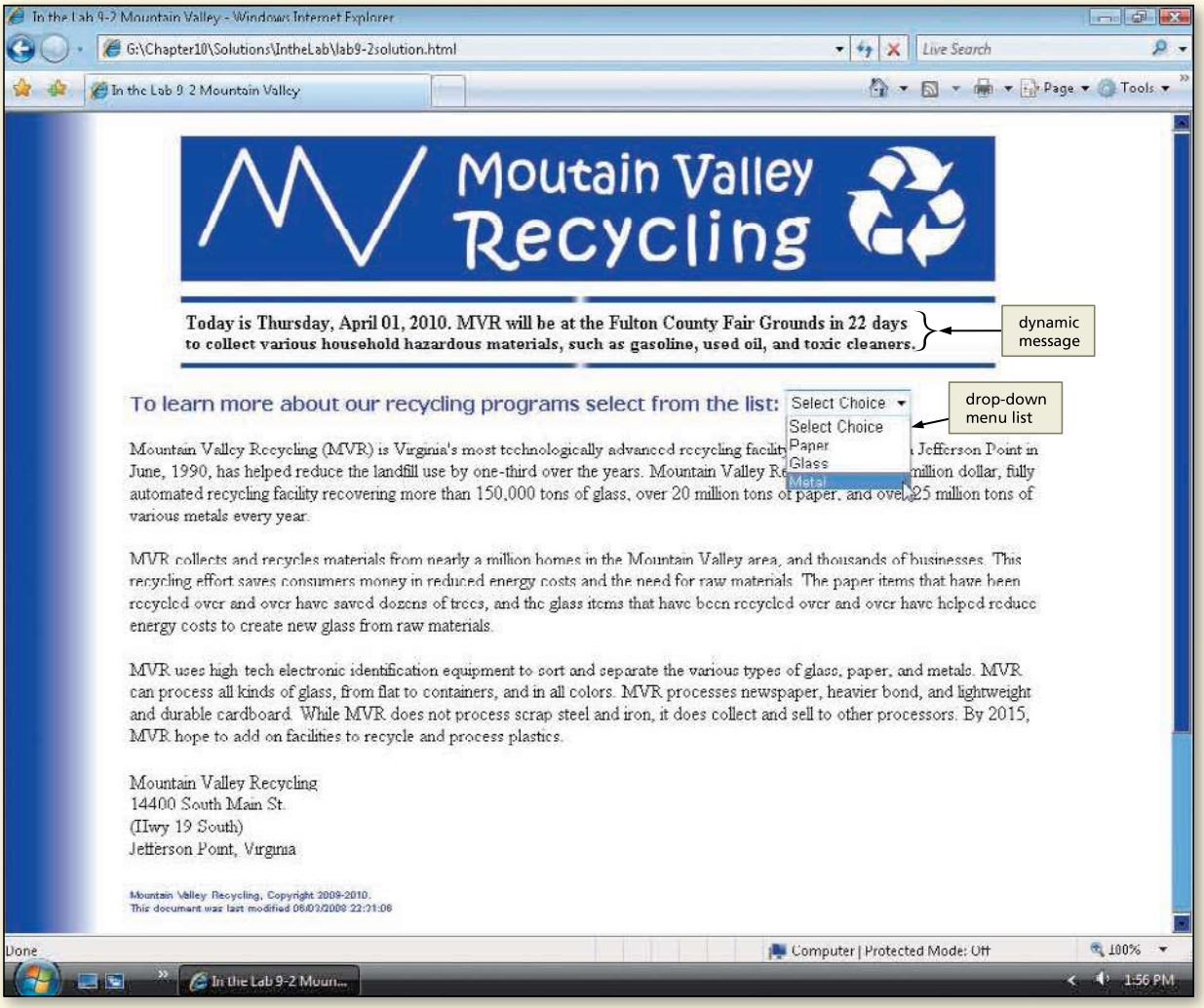


Figure 9–25

Instructions: Perform the following tasks:

1. Start Notepad and open the lab9-2.html file from the Chapter09\IntheLab folder of the Data Files for Students.
2. Save the file as lab9-2solution.html.
3. Start a new <script> section in the <head> section, on the line following the <title> tag, for two user-defined functions.
4. Write a JavaScript function to change the scroll bar colors called valleyScroll(). Use the code in Table 9–23 as a guide, and use the following color values for the scroll bar face and track colors:

   face color: #0000cc

   track color: #6daff0
5. Write a JavaScript function called menuLinks() that uses the selectedIndex value of the menuList <select> list in the recycleMenu <form>. Use the code in Table 9–26 as a guide.
6. In the table cell beneath the blue divider line, write the JavaScript script code to take the current date and calculate the number of days until the fair. Use a future date associated with the current date in your code. The script code should display a dynamic message as shown in Figure 9–25.
7. Before the closing </body> tag, write the JavaScript code to display the copyright and the date the Web page was last modified.

Continued >

In the Lab *continued*

8. Add the event handler to call the valleyScroll() function and the event handler to call the menuLinks() function. Place these event handlers in the appropriate HTML tags.
9. Save the completed HTML file and test it using your browser. If an error occurs, check your code and save and test again.
10. Submit the completed HTML file and Web page in the format specified by your instructor.

STUDENT ASSIGNMENTS

In the Lab

Lab 3: The Rocky Mountain Outdoor Sportsmens Show

Problem: You work as an event planner for the Rocky Mountain Arena. The annual Outdoor Sportsmens show is coming soon. Your assignment is to create a Web page that announces the upcoming event (Figure 9–26).

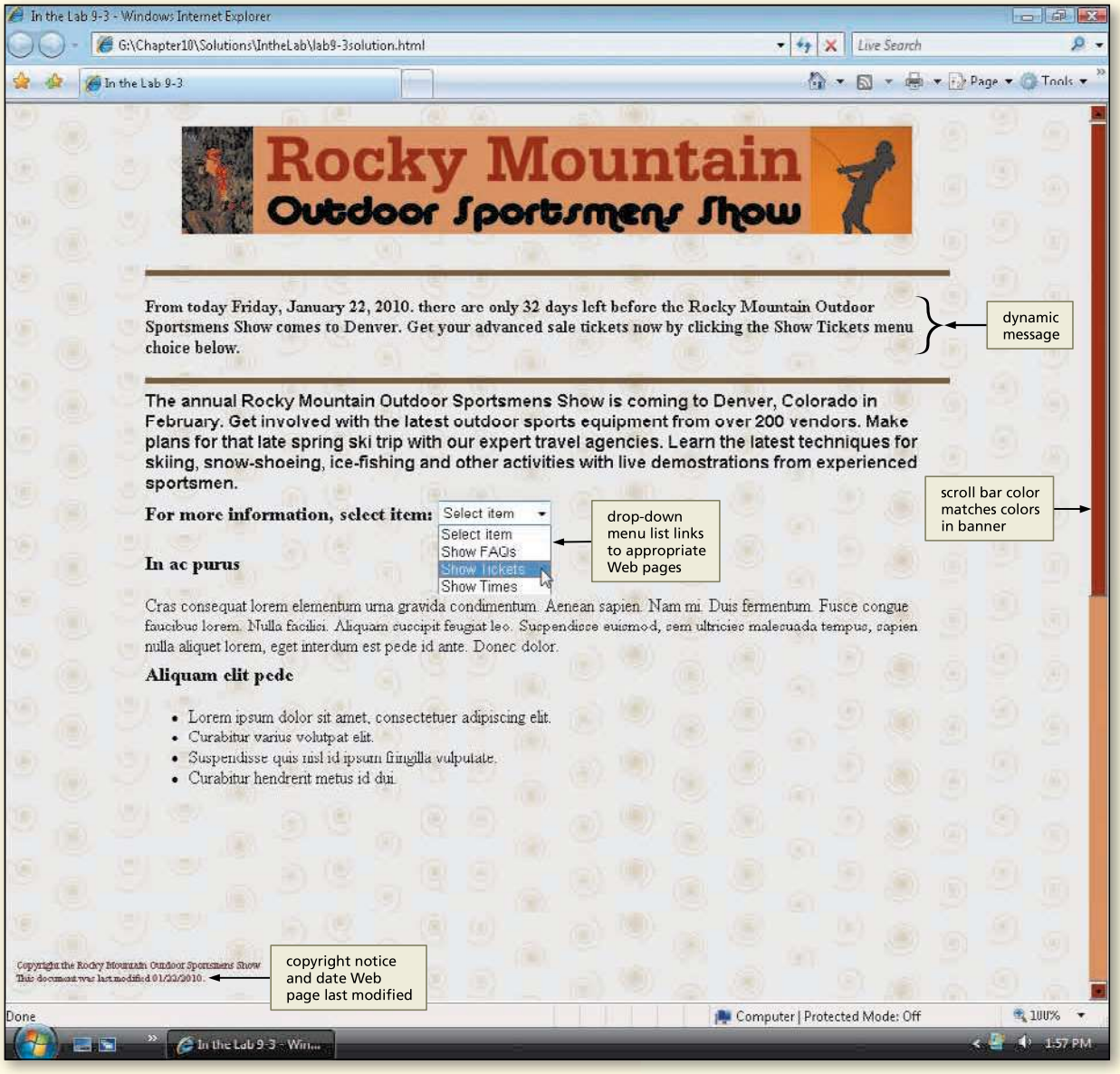


Figure 9–26

Instructions: Perform the following tasks:

1. Start Notepad and open the lab9-3.html file from the Chapter09\IntheLab folder. Save the file as lab9-3solution.html.
2. Using the techniques learned in this chapter, write the JavaScript code to create a dynamic message with a countdown, change the scroll bar color, and use the selectedIndex in the <select> tag to link to new Web pages. The menu links are Show FAQs, Show Tickets, and Show Times. Add a copyright notice and the date the Web page was last modified. Use a future date near the current date for this lab. For the scroll bar face color use #990000 and for scroll bar track color use #dc9b60.
3. Save the completed HTML file and test it using your browser. If an error occurs, check your code, and save and test again.
4. Submit the completed HTML file and Web page in the format specified by your instructor.

Cases and Places

Apply your creative thinking and problem solving skills to design and implement a solution.

• Easier •• More Difficult

• 1: Expanding the Chapter Web Page

West Lake Landscaping has received numerous requests for more information to be added to its Web site. One of the most common requests has been more information about decks. Using the material presented in this chapter, modify your chapter09westlake.html file to add two additional links in the drop-down menu list.

One link should be to a Web page about decks and the other is a link to garden design organizations (http://www.apld.com/). Use the file case9-1deck.html in the CasesPlaces folder of the Data Files for Students for the deck link and modify this page by replacing the simple greeting with a dynamic greeting announcing the deck exposition, change the color of the scroll bar, and add the copyright and date last modified information on the bottom of this Web page. Save as case9-1decksolution.html.

•• 2: Create the Shopper Newspaper Web Site

As a summer intern for the East San Alameda Heights Crosstown Shopper you have been asked to create the online version of the shopper. The page should have a drop-down menu list with links to coupons, real estate ads, and personal classified Web pages. The shopper page should change the scroll bar face color to #999999 and the track color to #cccccc. Make up an event the Crosstown Shopper might advertise. Create a dynamic message indicating the number of days to that event. Be sure to add the copyright, the URL, and the date last modified in a JavaScript section at the end of the shopper page and at the end of the case9-2coupons.html, case9-2personals.html, case9-2realestate.html pages found in the data files folder.

•• 3: Create the Tri City Community College Web Site

As the newly hired webmaster for Tri City Community College, you are going to redesign its Web site. You want to add a dynamic message to announce the number of days until the upcoming Job Fair. You use a drop-down menu to link to financial aid, student housing, and student life Web pages, and write a JavaScript user-defined function to change the URL location.

Because CSS does not have a standard for changing scroll bar colors, you add a JavaScript user-defined function to change the face color to #cb5b30, and the track color to #6b5030. You want a dynamic message that displays the current date and then calculates the number of days to the upcoming job fair. At the bottom of the Web page you display the copyright information and the date the Web page was last modified. Display only the date, not the time, with the lastModified property. Be sure to add the proper event handlers in the correct locations to call the user-defined functions. Use files from the Chapter09\CasesPlaces folder of the Data Files for Students to help build your Web page.

•• 4: Create a Family Web Page

Make It Personal

Many families have begun sharing information via the Internet. Many people use MySpace® and Facebook® to share photos and other information. Carefully consider your own family and then use the concepts and techniques presented in this chapter to create a Web page that announces a birthday, wedding, anniversary, or some other special family event. Use a dynamic message to display the current date and the number of days to the event. Make your page long enough so that the scroll bar is active and write the JavaScript user-defined function to change the scroll bar color. Create a drop-down menu list to link to other Web pages that you have created or that already exist, such as links to family or friends on MySpace or Facebook. Add a copyright notice and add the date the page was last modified at the bottom in small print. Be sure to check spelling and grammar on the Web pages that you create.

•• 5: Critique an Existing Web Site

Working Together

Many organizations in your community have Web pages. Each team member should search for these organization Web pages. Your team should find at least eight to ten Web sites. Print the Web pages and critique the layout and information presented on the page. Try to determine if the Web site has used JavaScript or some other scripting method (you can look at the source code). As a group, list four to five features that you like on each page, and four to five features you think could be improved on each page. If you find that a page used JavaScript, make note of how JavaScript was used for future reference in this text. Write up your evaluation and critique as a team and hand in the printed Web pages with the critique report.

This page intentionally left blank

10 Creating Pop-Up Windows, Adding Scrolling Messages, and Validating Forms

Objectives

You will have mastered the material in this chapter when you can:

- Write a JavaScript user-defined function to display a scrolling message
- Write a JavaScript user-defined function to validate form data
- Write a JavaScript user-defined function to calculate loan payments
- Define if and if...else statements, conditionals, and operands
- Write a JavaScript user-defined function to format output in a text field
- Describe how to display a pop-up window

10 Creating Pop-Up Windows, Adding Scrolling Messages, and Validating Forms

Introduction

In Chapter 9, you learned how to integrate JavaScript into an HTML document using variables and objects, and how to write JavaScript user-defined functions called by event handlers. This chapter reinforces these skills and shows you how to create a scrolling message that displays a text message in a form text field, using JavaScript to validate the data users enter into forms. The validation techniques discussed in this chapter use the if…else statement; parseInt(), parseFloat(), and isNaN() built-in functions; the Math object's pow() method; and the Number object's toFixed() method. Finally, the chapter discusses how to display a pop-up window.

Project — Statewide Realty Mortgage Loan Calculator

Many bank and real estate Web sites include monthly payment calculators that help buyers determine what their monthly payments will be for a car, mortgage, or other type of loan. These calculators generally allow buyers to input basic loan information — total amount, interest rate, and number of years — that is used to determine the monthly payment.

Recently, Statewide Realty has decided to improve their Web site based on a recent customer-satisfaction survey. One of the most-requested items was a simple loan calculator that would allow customers to estimate their monthly mortgage payments. As one of the Web developers, you have been assigned to create this new Web page.

You decide to create an interactive form that allows customers to enter the mortgage amount, interest rate, and number of years for the loan. After entering the information, users click a Calculate button to display the monthly mortgage payment or the Reset button to clear the text boxes. You suggest adding a simple scrolling message box that urges customers to take advantage of current low mortgage rates. You also suggest adding a pop-up window to promote Statewide's online home finder service. Figure 10–1 shows the pop-up window, the scrolling message, and the user input form for the mortgage calculator.

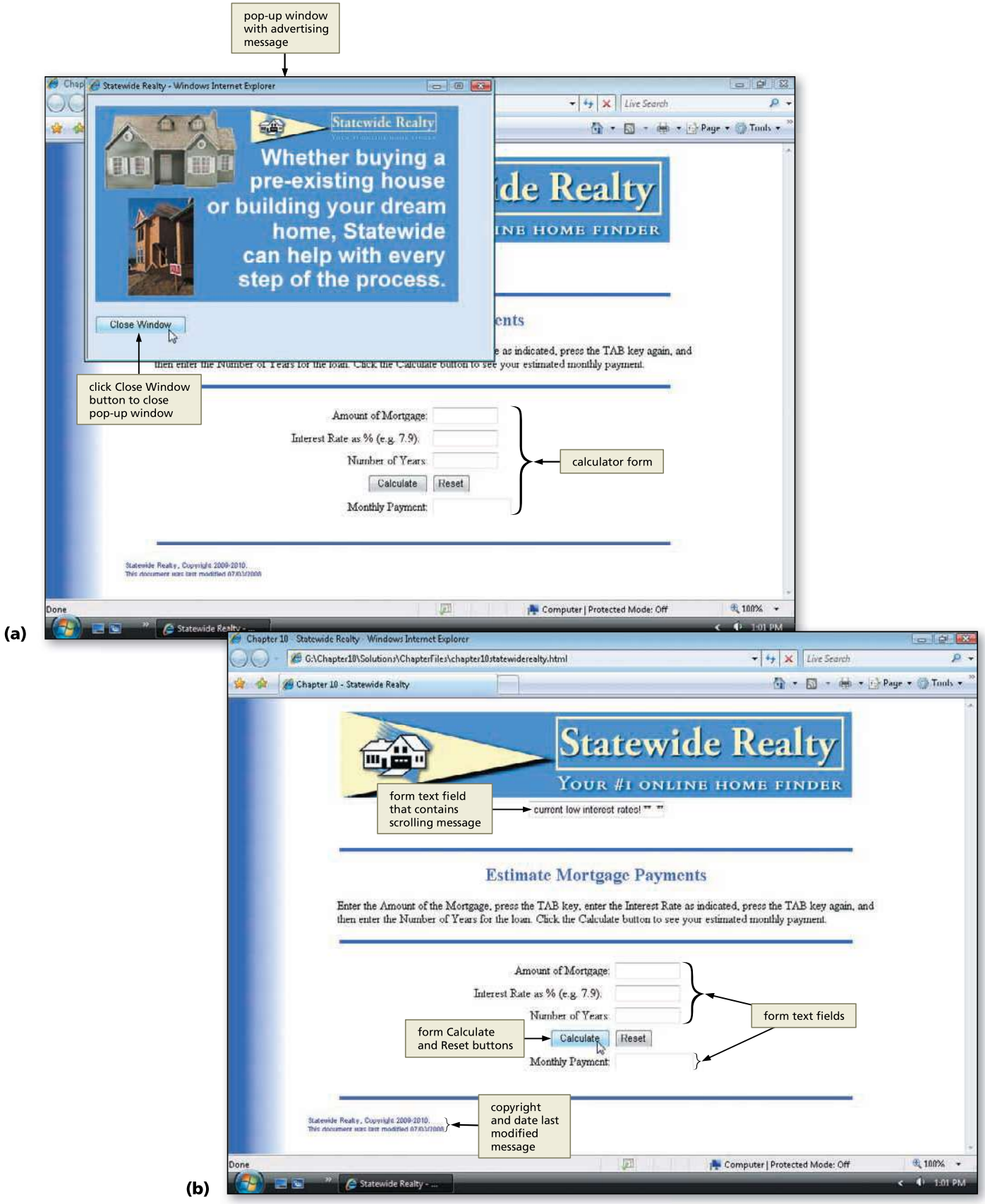

pop-up window with advertising message
Statewide Realty - Windows Internet Explorer
Statewide Realty
Whether buying a pre-existing house or building your dream home, Statewide can help with every step of the process.
Close Window
click Close Window button to close pop-up window
Live Search
Page
Tools
de Realty
INE HOME FINDER
ents
e as indicated, press the TAB key again, and
then enter the Number of Years for the loan. Click the Calculate button to see your estimated monthly payment.
Amount of Mortgage:
Interest Rate as % (e.g. 7.9):
Number of Years:
Calculate
Reset
Monthly Payment:
calculator form
Statewide Realty, Copyright 2009-2010.
This document was last modified 07/03/2008
Done
Computer | Protected Mode: Off
100%
1:01 PM
(a)
Chapter 10 - Statewide Realty - Windows Internet Explorer
G:\Chapter10\Solutions\ChapterFiles\chapter10statewiderealty.html
Chapter 10 - Statewide Realty
Statewide Realty
YOUR #1 ONLINE HOME FINDER
form text field that contains scrolling message
current low interest rates!
Estimate Mortgage Payments
Enter the Amount of the Mortgage, press the TAB key, enter the Interest Rate as indicated, press the TAB key again, and then enter the Number of Years for the loan. Click the Calculate button to see your estimated monthly payment.
Amount of Mortgage:
Interest Rate as % (e.g. 7.9):
Number of Years:
form text fields
form Calculate and Reset buttons
Calculate
Reset
Monthly Payment:
copyright and date last modified message
Statewide Realty, Copyright 2009-2010.
This document was last modified 07/03/2008
Done
Computer | Protected Mode: Off
100%
1:01 PM
(b)

Figure 10–1

Overview

As you read this chapter, you will learn how to write embedded JavaScript code to create the Web pages shown in Figures 10–1a and 10–1b by performing these general tasks:

- Open an existing HTML file and add JavaScript code.
- Create a scrolling message in a text field.
- Calculate a mortgage payment based on loan amount, interest rate, and number of years.
- Validate data entered into a form.
- Format the monthly payment to display as currency.
- Open a pop-up window when the Web page initially loads.
- Convert text to numeric values using built-in functions.
- Display the date the Web page was last modified.
- Save, validate, and test the Web pages.
- Print the HTML code and Web pages.

Plan Ahead

General Project Guidelines

When adding JavaScript or any scripting language to a Web page document, the actions you perform and decisions you make will affect the appearance and characteristics of the finished Web page. Before you write the JavaScript code, you should follow these general guidelines:

- **Determine what you want the code to accomplish.** For this chapter's project, you want to create a scrolling message in a text field, add a pop-up window, create a form for user input, validate the user input, perform a calculation based on the user input, output a result formatted as currency, and display the date the Web page was last modified.
- **Determine where in the Web page you want the code to appear.** All the JavaScript code in this chapter will be placed in the <head> section of the HTML code in user-defined functions. Event handlers will call these functions as needed.
- **Determine the overall Web page appearance.** When the Web page first loads, a pop-up window is displayed. The Web page also includes a text message that scrolls continuously. Data for the mortgage calculation is entered in a form, validated, and the results are displayed in currency format. The date the page was last modified displays at the bottom of the page.
- **Determine the data validation requirements.** Before the monthly payment can be calculated, the data entered in the form must be validated. The loan amount, interest rate, and the number of years for the loan must be numeric, not blank, and greater than zero. If the data does not meet these criteria, an alert message box notifies the user and positions the insertion point in the appropriate text field.
- **Determine the calculations needed.** You will need a formula for calculating the monthly payment. This formula is given later in the chapter.

When necessary, more specific details concerning the above guidelines are presented at appropriate points in the chapter. The chapter also will identify the actions performed and decisions made regarding these guidelines during the creation of the Web page shown in Figure 10–1 on the previous page.

Inserting a Scrolling Message on a Web Page

A simple way to provide a Web site visitor with information is to add a scrolling text message to a Web page. Companies often use scrolling messages on their Web sites to highlight breaking news, key products, or special promotions. A scrolling text message can appear either in a text field within the Web page or on the status bar in the browser window. Because visitors to a Web page often do not look at the status bar, most Web developers agree that a scrolling message in a text field on the Web page is a better location.

A scrolling message has four basic components:

- The display object (a form text field)
- The text message to scroll in the text field
- The position of the next character in the text message
- A time delay

The **display object** identifies where the scrolling message is displayed, which, in this project, is in a form text field. The scrolling **message** is a text string assigned to a variable. The text string is what the user sees when the message is displayed. The **position** is the location of the next character in the text string. The **delay** regulates the speed in which the characters display in the text field.

BTW

Placement of Scrolling Text

Another reason to avoid placing scrolling text on the status bar is that it can be missed easily by the visitor.

The first step in creating the scrolling message for the Statewide Realty Web page is to create the display object (the text field). The text field is part of a simple form containing only the text field to display the scrolling message. The form and text field for the scrolling message are positioned below the title image. You begin by opening an existing HTML document, and adding the code to create a form and text field.

You must name the form and the form text field objects. These names serve as the object and properties used in the JavaScript code to assign the message string to the text field. The size attribute of the text field indicates the display width of the text field. Table 10–1 shows the HTML code to create the form and a text field for the scrolling message.

Table 10–1 Code to Create a Form and a Text Field

Line	Code
32	`<form name="msgForm" action="">`
33	`<input type="text" name="scrollingMsg" size="25" />`
34	`</form>`

Line 32 starts the form and uses the name attribute to give the form the unique name, msgForm. Line 33 indicates the input box is a text type, which means it can receive data. The text field is named scrollingMsg and is set to a size of 25. Line 34 is the closing <form> tag.

To Open an Existing HTML File

As in Chapter 9, you will integrate JavaScript into an existing HTML document. The following step shows how to open the chapter10.html file included in the Data Files for Students.

1

- Start Notepad, and, if necessary, maximize the window. If Word Wrap is not enabled, click Format on the menu bar and then click Word Wrap to enable it.
- With a USB drive plugged in to your computer, click File on the menu bar and then click Open.

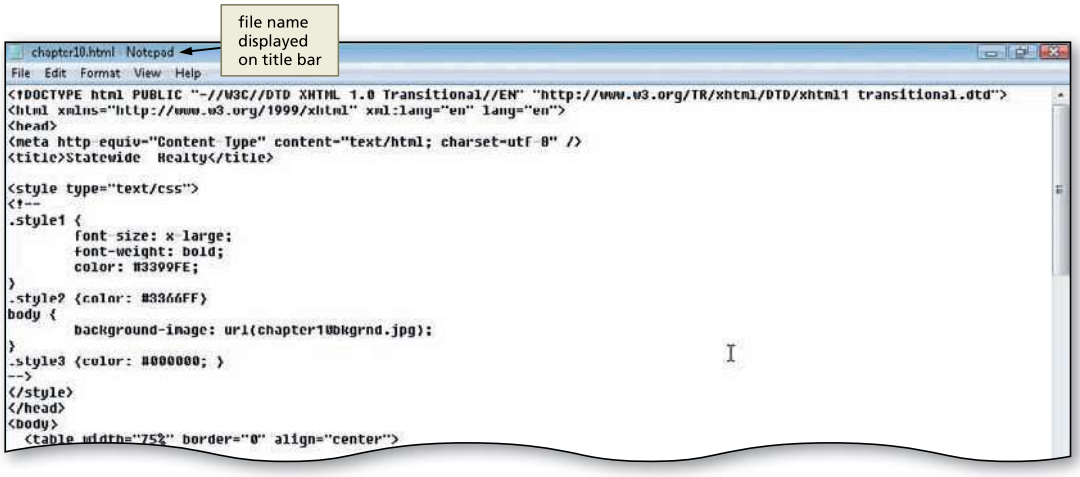


Figure 10–2

- If necessary, navigate to the Chapter10\ChapterFiles folder on the USB drive.
- If necessary, click the Look in box arrow, and then click All Files to display all files in the Chapter10\ChapterFiles folder.
- Click chapter10.html in the list of files.
- Click the Open button to open the chapter10.html file in Notepad (Figure 10–2).

To Create a Form Text Field to Display a Scrolling Message

The following step illustrates how to create a form and a form text field to display a scrolling message.

1

- Click line 32 below the closing <p align="center"> tag.
- Enter the JavaScript code shown in Table 10–1 to enter the HTML code to create the form and text field (Figure 10–3).

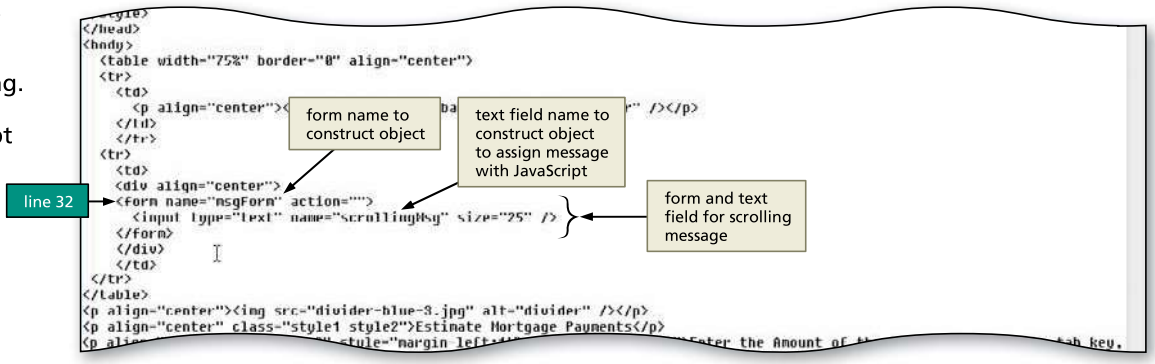


Figure 10–3

Q&A Can more than one scrolling message be placed on a Web page?

Generally not, especially with older browsers. Too many recursive calls can overflow the programming stack and crash the browser.

Creating the scrollingMsg() User-defined Function

The scrollingMsg() function requires two variables and performs five tasks. The two variables are:

- The scrollMsg variable represents the message
- The beginPos variable represents the current character in the text message.

The five tasks the scrollingMsg() function performs are:

- Assigns the string message to the display object (which, in this project, is the text field)
- Increments the position variable by 1 to place the next character in the text message in the display object
- Uses an if statement to test for the end of the message
- If the text has scrolled to the end of the message, starts over with the first character
- Makes the display continuous and regulates the speed of the display using the setTimeout() method set to 200 milliseconds

Table 10–2 shows the code to begin the JavaScript section, declare and initialize the scrollMsg and beginPos variables, declare the scrollingMsg() function, and assign the first characters of the message to the text field. *Note: Because of the limitations of this textbook page, Lines 8 and 11 look like more than one line. However, each will be entered as a single line, pressing Enter only at the end of the entire numbered line.*

Table 10–2 Code to Begin the scrollingMsg() Function

Line	Code
6	`<script type="text/javascript">`
7	`<!--Hide from old browsers`
8	`var scrollMsg = " ** Take advantage of the current low interest rates! ** "`
9	`var beginPos = 0`
10	`function scrollingMsg() {`
11	`document.msgForm.scrollingMsg.value = scrollMsg.substring (beginPos,scrollMsg.length)+scrollMsg.substring(0,beginPos)`

Lines 6 and 7 start the <script> section of the Web page file. Line 8 declares the scrollMsg variable and assigns the message string, ** Take advantage of the current low interest rates! **, to it. The spaces at the beginning and end of the message string ensure that spaces appear at both ends of the message. Line 9 declares the beginPos variable, used to indicate the beginning position of the text string, and initializes it to zero. Line 10 declares the function scrollingMsg(). Line 11 assigns the message string to the text field using the object document.msgForm.scrollingMsg.value, which is derived from the form object and the input object. Figure 10–4 illustrates the relationship between these objects and how the statement is derived.

BTW

The Marquee <marquee> Tag

To make it easier to build scrolling messages, Microsoft developed the <marquee> tag. The direction attribute in the <marquee> tag controls scrolling up, down, left, or right. Internet Explorer recognizes the <marquee> tag, but other browsers do not. To create a scrolling message that works with Internet Explorer and other browsers, use a form text field, as discussed in this chapter.

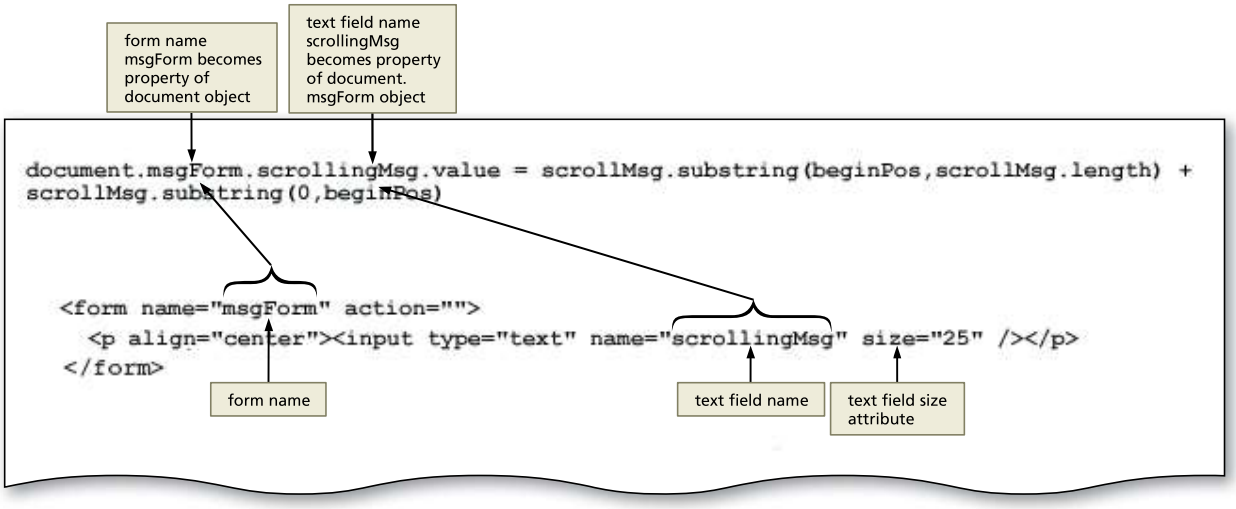


Figure 10–4

Line 11 also assigns the next character to the form text field object. The text field object is constructed using the form (msgForm) as an attribute of the document and the text field object (scrollingMsg) as an attribute of the msgForm object. The JavaScript code then assigns the string message to the input text field object (scrollingMsg) using the value attribute.

The rest of the assignment statement in line 11 uses the substring() method and concatenates the remainder of the scrollMsg variable to the beginning of the scrollMsg variable. As you learned in Chapter 9, the substring() method needs two parameters (x,y), where x is the starting point of the string and y is the location of the last character needed. This statement tells the scrollingMsg() function to assign the next character in the string message to the text field, to make the message appear as if it is scrolling.

To Create the scrollingMsg() User-Defined Function

The following step shows how to create the scrollingMsg() user-defined function and define its variables.

1

- Click line 6, the blank line below the <title> tag.
- Enter the JavaScript code shown in Table 10–2 to enter the beginning script tags and define the variables used in the scrolling message, using the SPACEBAR to indent as shown, and then press the ENTER key (Figure 10–5).

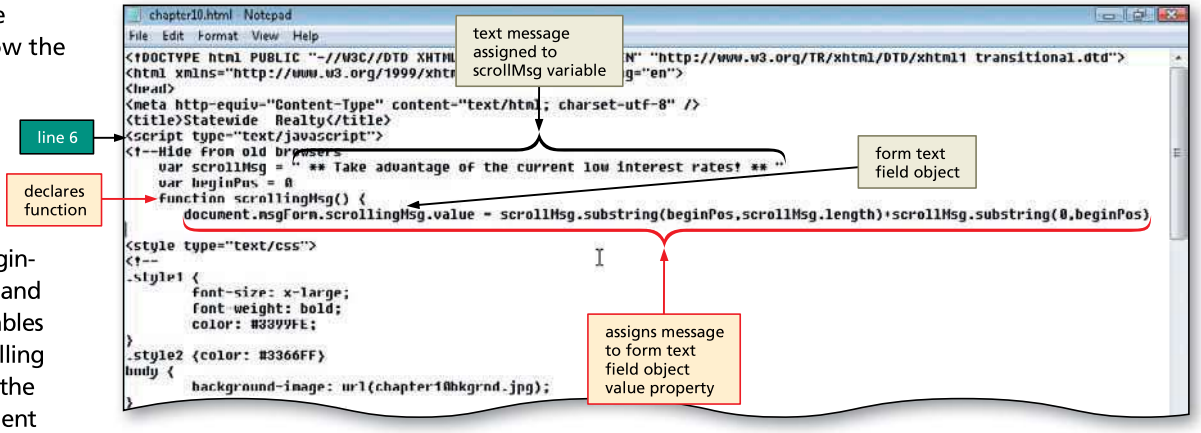


Figure 10–5

Incrementing the Position Locator Variable After declaring the scrollingMsg() function, the next step is to increment the beginPos variable and append the next character from the message string to the text field. To cause the message to scroll in the text field, the position locator variable (beginPos) must be incremented by one. Table 10–3 describes the various ways JavaScript statements can be used to increment variables.

Table 10–3 Incrementing a Variable

Statement	Explanation
`variable=variable+1`	Executes the expression on the right side of the equal sign and assigns the result to a variable on the left side
`variable+=1`	Adds the number after the equal sign to a variable
`variable++`	Adds 1 to a variable, increments after the assignment
`++variable`	Adds 1 to a variable before the assignment

Once incremented, the new value of the position locator variable, beginPos, allows the substring() method in line 11 to extract the next character in the message string and append it to the end of the message in the text field.

To Enter the Code to Increment the Position Locator Variable

The following step illustrates how to enter the code to increment the position counter.

1. Click line 12.
2. Press SPACEBAR to indent under the previous line, then type `beginPos = beginPos + 1` to increment the position locator by one, and then press the ENTER key (Figure 10–6).

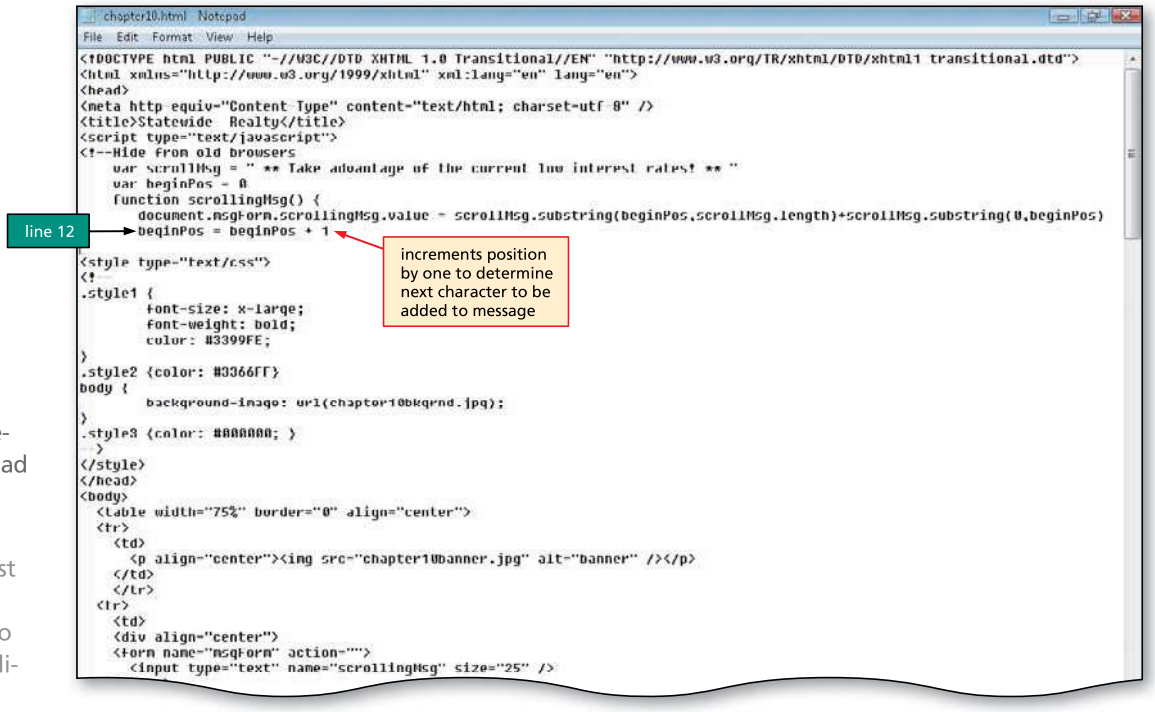


Figure 10–6

Q&A Why did we write the increment statement this way instead of using one of the other methods?

This way is the most common and easiest for beginners to understand. In addition, developers should use a format that can be recognized by anyone who might have to modify their code after the initial implementation.

Entering an if Statement After incrementing the position location variable (beginPos) by one, the JavaScript code must determine if the current value of beginPos exceeds the length of the message string. The loan payment calculator will use an if statement to determine if the current value of the beginPos variable is greater than the length of the message. An **if statement** is used to test a condition and then take one or more actions, based on the results of the test. The general form of the if statement is shown in Table 10–4. The if statement tests a **condition**, which is any comparison of values that evaluates to true or false. If the result of the comparison is true, the JavaScript code within the braces is executed. If the result of the comparison is false, the code after the closing brace is executed. Figure 10–7 shows the flowchart that corresponds to an if statement.

Table 10–4 If Statement

General form:	`if (condition) {`
	`  JavaScript statements if condition true`
	`}`

Comment:	where condition is the comparison of values. All conditions must be placed in parentheses. If the result of the comparison is true, JavaScript executes the statements between the curly braces. If the result of the comparison is false, the JavaScript statements after the closing brace are executed.

Example:	`if (beginPos>scrollMsg.length) {`
	`  beginPos=0`
	`}`

BTW

The if Statement
JavaScript if statements are an integral part of the programming language. They are used to define one or more statements that only should be executed based on the result of a conditional test, which controls the flow of logic.

As shown in the example in Table 10–4, the conditions use symbols called operators to indicate what type of comparisons should be made between the values. Table 10–5 shows the conditional operands used for comparisons. For more information about conditional operands, see the JavaScript Quick Reference in Appendix E.

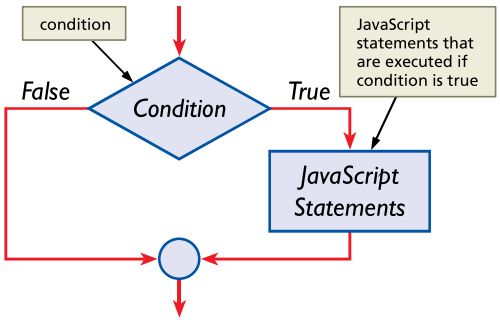


Figure 10–7

Table 10–5 Conditional Operators

Operand	Example	Results
==	(a==b)	True if a equals b
===	(a=== b)	True if a equals b and the data are of the same type
!=	(a!=b)	True if a does not equal b
!==	(a!==b)	True if a does not equal b and/or the data are not of the same type
>	(a>b)	True if a is greater than b
<	(a<b)	True if a is less than b
>=	(a>=b)	True if a is greater than or equal to b
<=	(a<=b)	True if a is less than or equal to b
&&	(a==b) && (x<y)	True if both conditions are true (a equals b and x is less than y)
||	(a!=b) || (x>=a)	True if either condition is true (a does not equal b or x is greater than or equal to a)

BTW

Conditional Operators
Conditional operators are symbols used to compare two values, called operands. The operator compares the two operands and returns a logical value based on whether the comparison is true.

To make the scrolling message work properly, an if statement is used to determine if the current value of beginPos is greater than the number of characters in the message string. The flowchart and sample code shown in Figure 10–8 illustrate how the if statement compares the beginning position variable (beginPos) with the overall length of the message (scrollMsg.length).

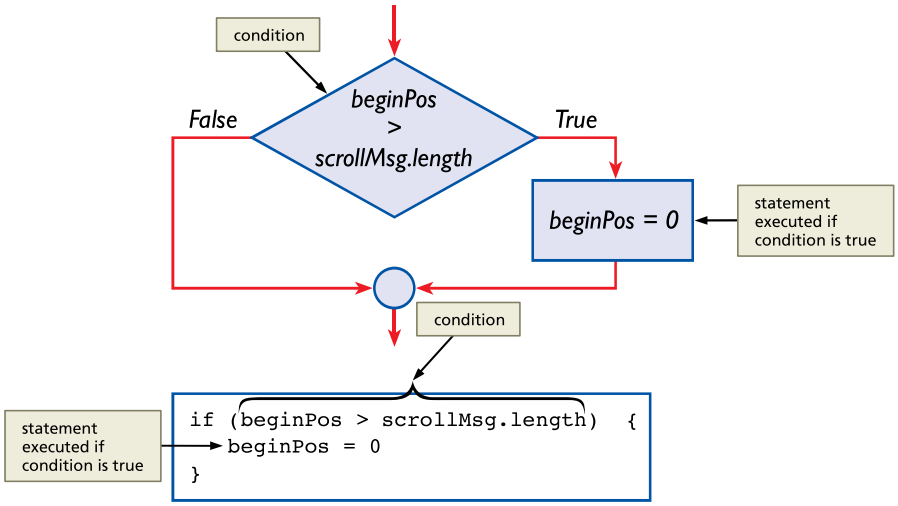


Figure 10–8

BTW

Operands
An operand is a numerical, string, logical, or object data type or value. Operands must be of the same data type, or you will not get a true comparison result.

If the current value of the beginPos variable exceeds the length of the scrollMsg variable, the statement assigns the value zero to the beginPos variable. By setting beginPos to zero, the code sets the string message so the first character of the string appears in the text field.

To Enter an if Statement

The following step illustrates how to enter an if statement.

1

- Click line 13.
- Press the SPACEBAR to indent under the line above.
- Type `if (beginPos > scrollMsg.length) {` to enter the if statement and then press the ENTER key.
- Press the SPACEBAR to indent under the open parenthesis in (beginPos.
- Type `beginPos=0` to reset the variable to zero if the condition is true, and then press the ENTER key.
- Press the SPACEBAR to indent under and line up with the if statement.

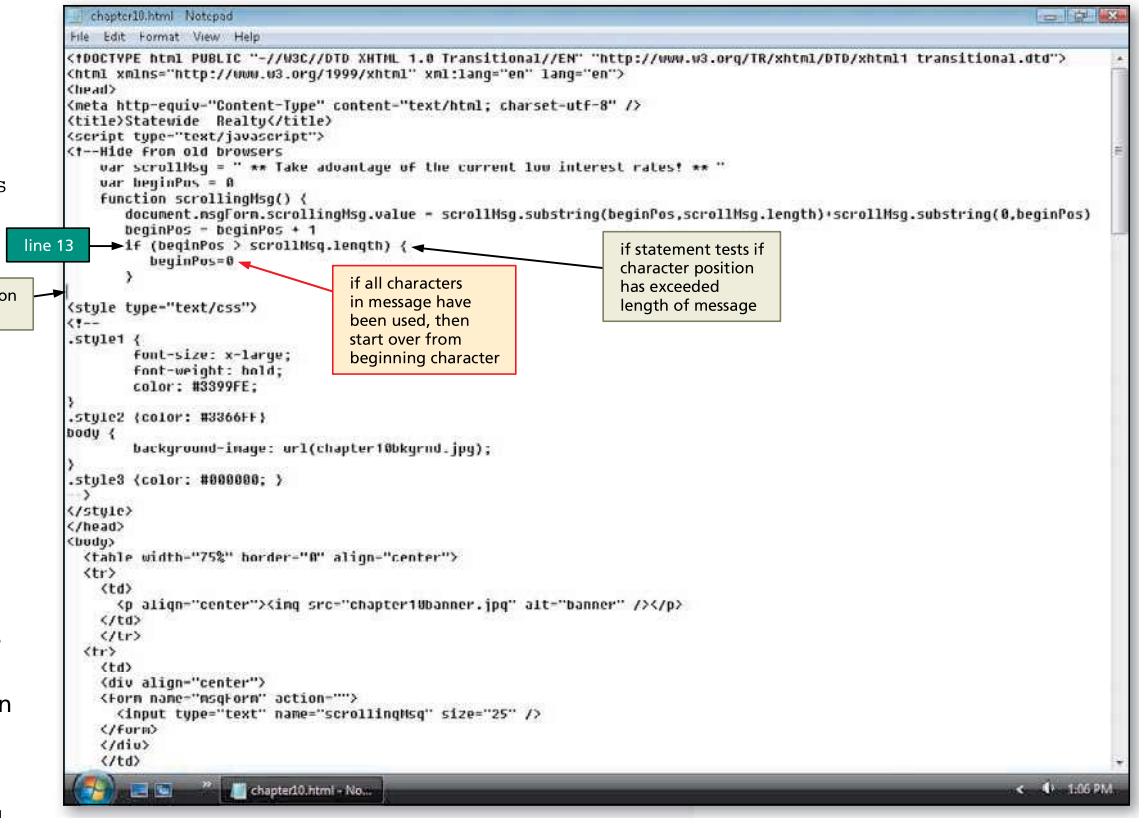


Figure 10–9

- Type `}` to close the if statement and then press the ENTER key (Figure 10–9).

Q&A

Do all JavaScript if statements have to be written in this format?

When only one statement follows the condition, like in this example, the statement could have been written as follows: `if (beginPos > scrollMsg.length) beginPos=0`. Note that the braces have been dropped for just one statement to be executed if the condition is true. If more than one statement needs to be executed, then the braces must be used to create the block of statements.

Using the setTimeout() Method to Create a Recursive Call To have the message text scroll continuously in the text field, you use a programming technique called **recursion**, in which a function is called within itself, creating an "endless loop." The setTimeout() method calls a function or evaluates an expression after a specified amount of time has elapsed, which is measured in milliseconds. The general form of the setTimeout() method is shown in Table 10–6.

BTW

Recursion
In this chapter's project, recursion is used to keep a routine going indefinitely or until some other function is called to stop it. Normally, recursive functions should have a mechanism that terminates the function when it completes its task.

Table 10–6 setTimeout() Method

General form:	`setTimeout("instruction", time delay in milliseconds)`
Comment:	where instruction is any valid JavaScript statement and time delay is expressed in number of milliseconds
Example:	`window.setTimeout("scrollingMsg()",200)`

In the scrollingMsg() user-defined function, the setTimeout() method continuously displays characters and regulates the speed of the characters displaying in the text field. The setTimeout() method calls the scrollingMsg() user-defined function from within the scrollingMsg() user-defined function. This recursive call to the scrollingMsg() function is what makes the message scroll in the text field continually.

To Add the setTimeout() Method to Create a Recursive Call

The following step illustrates how to add the setTimeout() method to create a recursive call to the scrollingMsg() function.

1

- If necessary, click line 16.
- Press the SPACEBAR to indent under the closing brace, then type `window.setTimeout("scrollingMsg()",200)` to call scrollingMsg() from within itself and then press the ENTER key.
- Press the SPACEBAR to indent and then type `}` to close the function and then press the ENTER key two times (Figure 10–10).

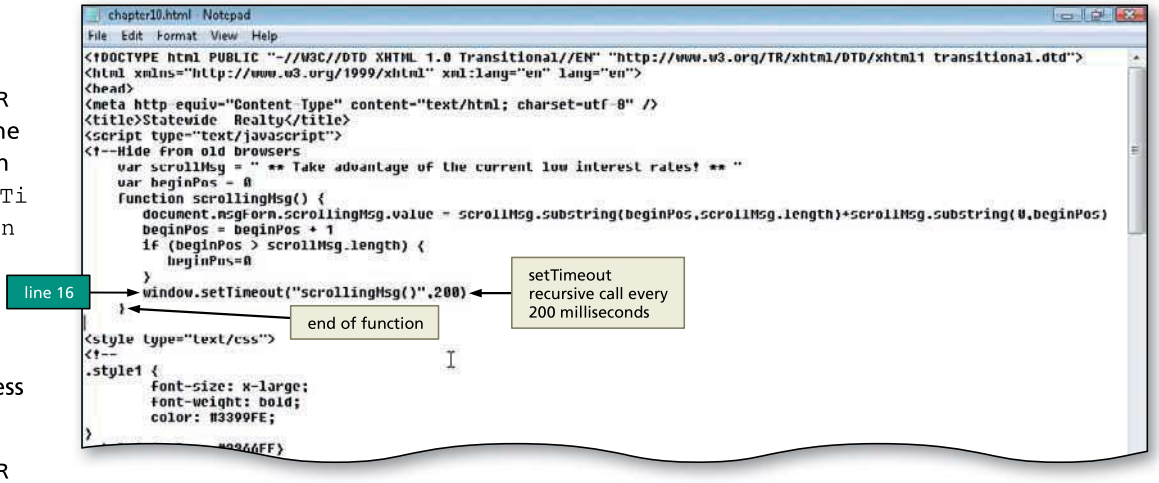


Figure 10–10

Q&A How do we know how fast to make the scrolling?

The best way is to try several different speeds and ask potential users to look at it and indicate their preference.

Q&A What if we changed the number from 200 to 2000?

The text would display one character every two seconds and would be so boring to watch, you would lose the interest of your user.

To Complete a JavaScript Section

The following step shows how to enter the JavaScript code to complete the <script> section.

1

- If necessary, click line 19.
- Type `//-->` to close the comment to hide the JavaScript and then press the ENTER key.
- Type `</script>` to close the <script> section and then do not press the ENTER key (Figure 10–11).

Q&A

Can I use one hyphen and the greater than sign to end the comment?

No, it must be two hyphens and the greater than sign to close the comment started above.

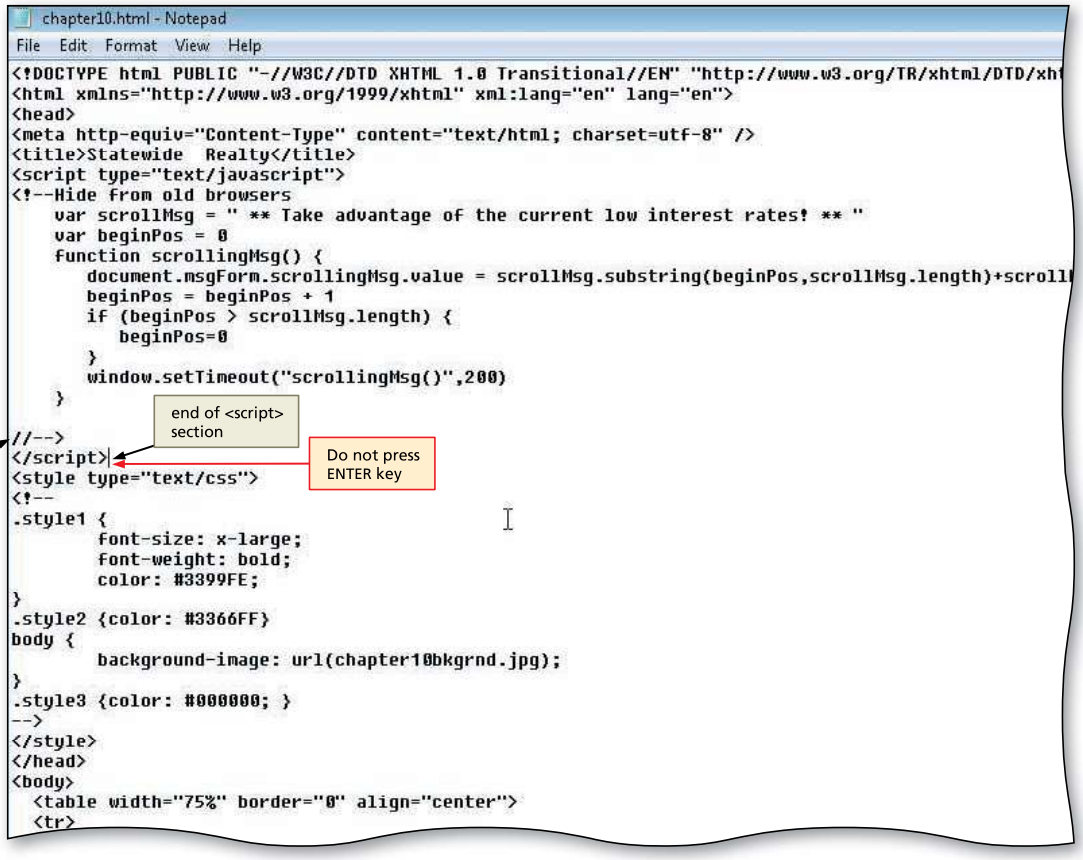


Figure 10–11

Adding an Onload Event Handler

BTW

Event Handlers
Some older browsers have a problem recognizing mixed case event handlers. Although the newer versions of Web browsers have fixed the problem, they still recognize event handlers using all lowercase characters.

The last step in adding a scrolling message to a Web page is to add an event handler to start the scrolling message when the Web page loads. As discussed in Chapter 9, an event is an action, such as a mouse click or a window loading. An event handler is a way to associate that action with a function. The event handler to start the scrolling message is the onload event handler.

The JavaScript standard uses both upper- and lowercase in spelling event handlers, as shown in Table 10–7. In this text, however, to be XHTML-compliant and to pass XML validation, developers spell the event handlers in all lowercase characters because XHTML treats event handlers as tag attributes. The XHTML standard requires all tag attributes to be lowercase.

Table 10–7 shows some of the event handlers and the associated objects. As the table indicates, event handlers can be used only with certain objects. For example, the onclick event handler is used to trigger JavaScript code when a user clicks a button or link, while the onload event handler is used to trigger JavaScript code when a document is loaded into the browser window. For more information about event handlers, see the JavaScript Quick Reference in Appendix E.

Table 10–7 Objects and Associated Event Handlers

Object	Event Handler
button	`onClick, onDblClick`
document	`onLoad, onUnload`
form	`onSubmit, onReset, onBlur, onKeydown, onKeypress, onKeyup`
hyperlink	`onClick, onMouseover, onMouseout, onDblClick, onMousemove, onMousedown`
image	`onLoad, onAbort, onError, onMousemove, onMousedown`
input box	`onBlur, onChange, onFocus, onKeypress, onKeyup, onKeydown`
Submit button	`onClick`
window	onLoad, `onUnload, onBlur, onFocus`

In this chapter, the onload event handler calls the scrollingMsg() function, using the following statement:

```
onload="scrollingMsg()"
```

where onload is the event handler and the scrollingMsg() function is the code that is executed as the result of the event. The statement is entered in the <body> tag to indicate that the onload event handler should call the scrollingMsg() function when the Web page loads.

To Enter the onload Event Handler to Call the scrollingMsg() Function

The following step illustrates how to enter the onload event handler to call the scrollingMsg() function.

1

- Click to the right of the y in the body in line 36.
- Press the SPACEBAR once.
- Type `onload= "scrollingMsg()"` to add the event handler and do not press the ENTER key (Figure 10–12).

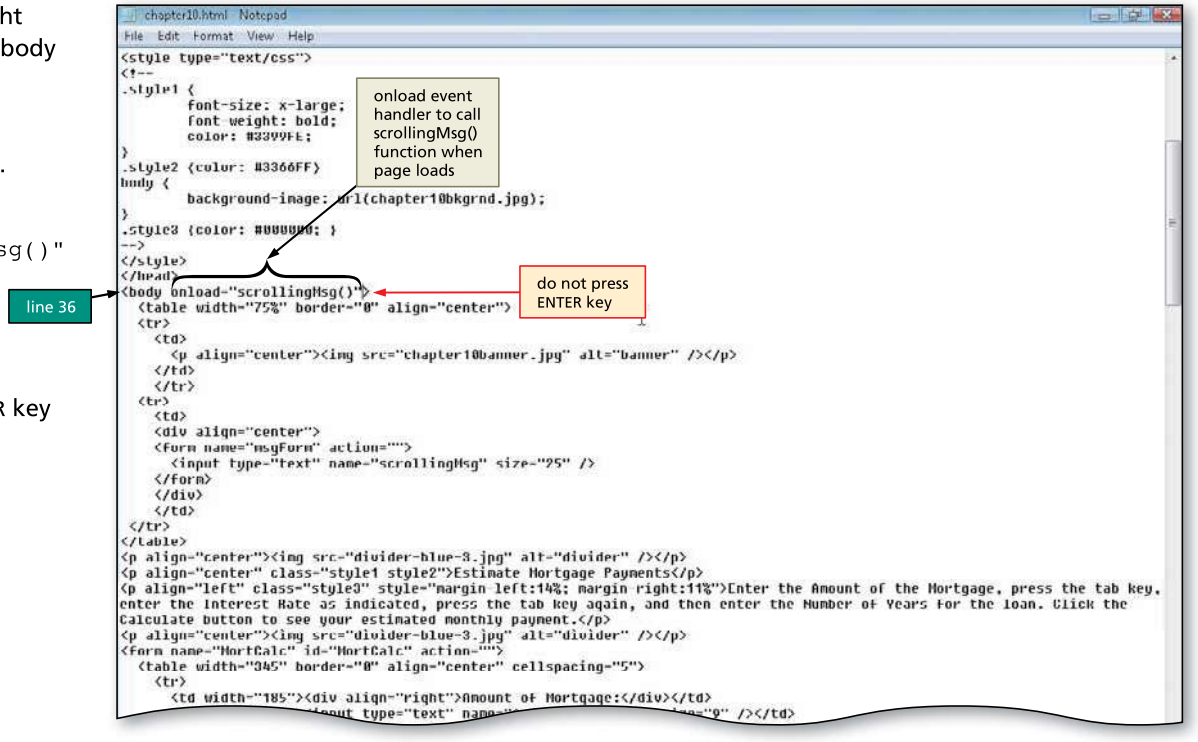


Figure 10–12

To Save an HTML File and Test a Web Page

With the code for the scrollingMsg() function complete and the onload event handler added to call the function when the Web page loads, you should save the HTML file and test the Web page. The following step illustrates how to save the HTML file and then test the Web page.

1

- With a USB drive plugged into your computer, click File on the Notepad menu bar and then click Save As. Type `chapter10 statewiderealty.html` in the File name text box (do not press the ENTER key).
- If necessary, browse to the USB drive and open the Chapter10\ChapterFiles folder.
- Click the Save button in the Save As dialog box to save the file on the USB drive with the name chapter10 statewiderealty.html.
- Start your browser. If necessary, click the Maximize button.
- Type `g:\chapter10\ChapterFiles\chapter10 statewiderealty.html` in the Address box and then press the ENTER key.
- If necessary, click the security bar under the tabs, click Allow Blocked Content... (Figure 10–13a), and if necessary click Yes in the Security Warning dialog box to display the scrolling message (Figure 10–13b).

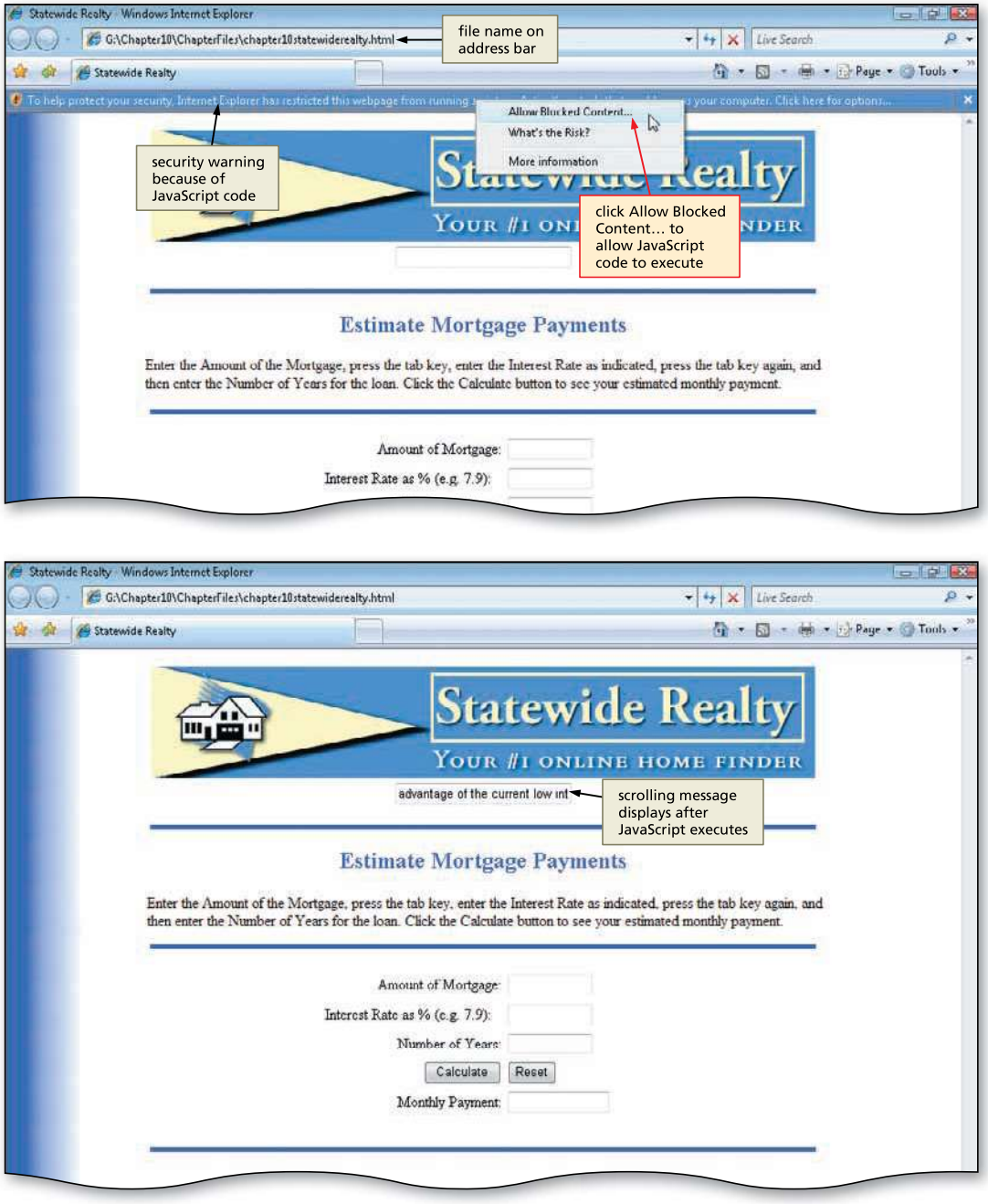


Figure 10–13

Q&A What if I do not see a security bar?

That simply means that tight restrictions and security are not set on your browser.

Validating a form.
In order to calculate the monthly payment, the values entered into the text fields must be valid numbers. A user-defined function called Calc() follows these steps to validate the text field entries:

- Convert the text field value to a numeric value using the parseInt() or parseFloat() function
- Test the value to be numeric using the isNaN (is Not a Number) function and checking that the value is greater than zero
- If the value is not a number or is zero or less, display a message, clear the text field, and position the insertion point in that text field

To call the Calc() validation function, an onClick event handler is added to the form.

Plan Ahead

Adding a Loan Payment Calculator

The mortgage loan payment calculator form shown in Figure 10–14 requests user input. The form, which is named MortCalc, already has been created in the HTML file. JavaScript code must be added to validate the input, calculate the monthly payment, and display the results in the MortCalc form. In order for the calculator to work, each text field must have a valid data entry. You will write a user-defined function called Calc() to perform these three tasks.

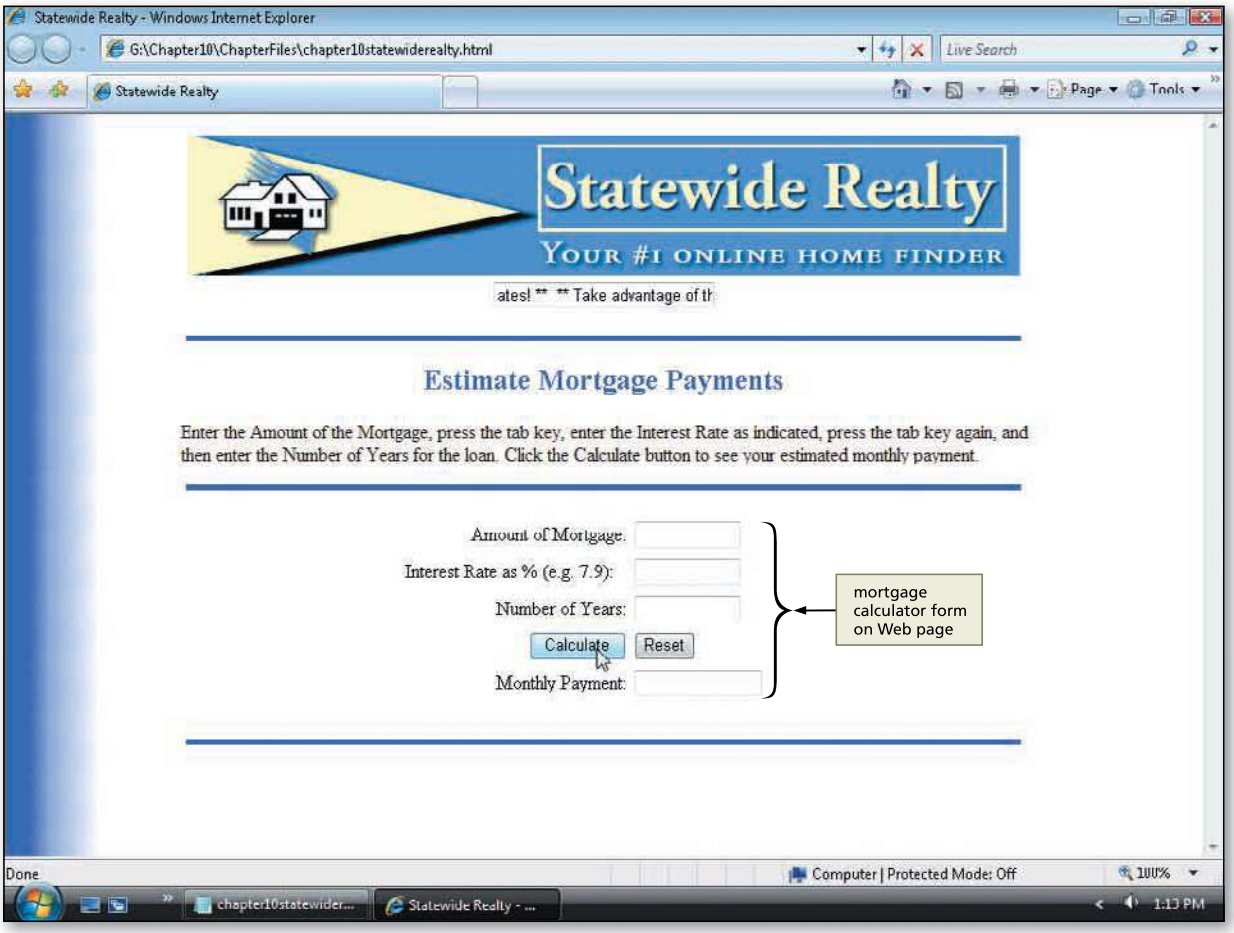


Figure 10–14

BTW

Validating Web Form Data

Web developers use multiple techniques to validate Web forms using JavaScript. Some developers choose to validate each item as it is entered by using a combination of onchange event handlers and user-defined functions. It is important to validate Web forms because some Web databases are sensitive to invalid data and may crash, or mathematical formulas may cease to function when using invalid data.

Validating Forms Using Nested if…else Statements

You can use different techniques to validate forms. This chapter uses a series of nested if…else statements, which is like the if statement except that it specifies statements to execute if the condition is false, as shown in the flowchart in Figure 10–15. Much like the if statement, an if…else statement tests a condition. If the condition is true, the statements between the curly braces after the if statement execute. If the condition is false, the statements between the braces after the else statement execute.

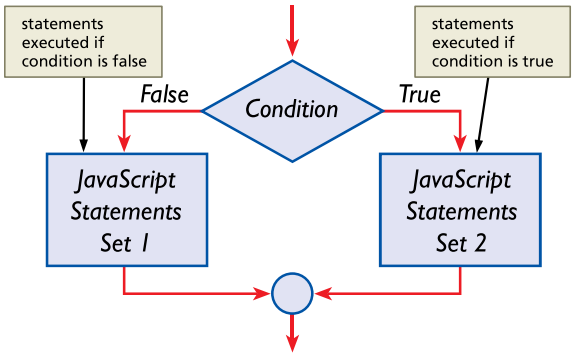


Figure 10–15

The validation algorithm begins by converting the text field value to a number. The if…else statement tests if the value entered in a text field is invalid (a true condition). If true, an error message is displayed, the text field is cleared, and the insertion point is placed back in the text field. This occurs until the user enters valid data in the text field. If the value entered in the text field is valid (a false condition), the next text field is examined until all text fields are validated. The validation process is shown in the flowchart in Figure 10–16.

BTW

Domains

A domain is a range of acceptable values for a field or column in a database. Using an HTML select list can ensure that accurate values are entered.

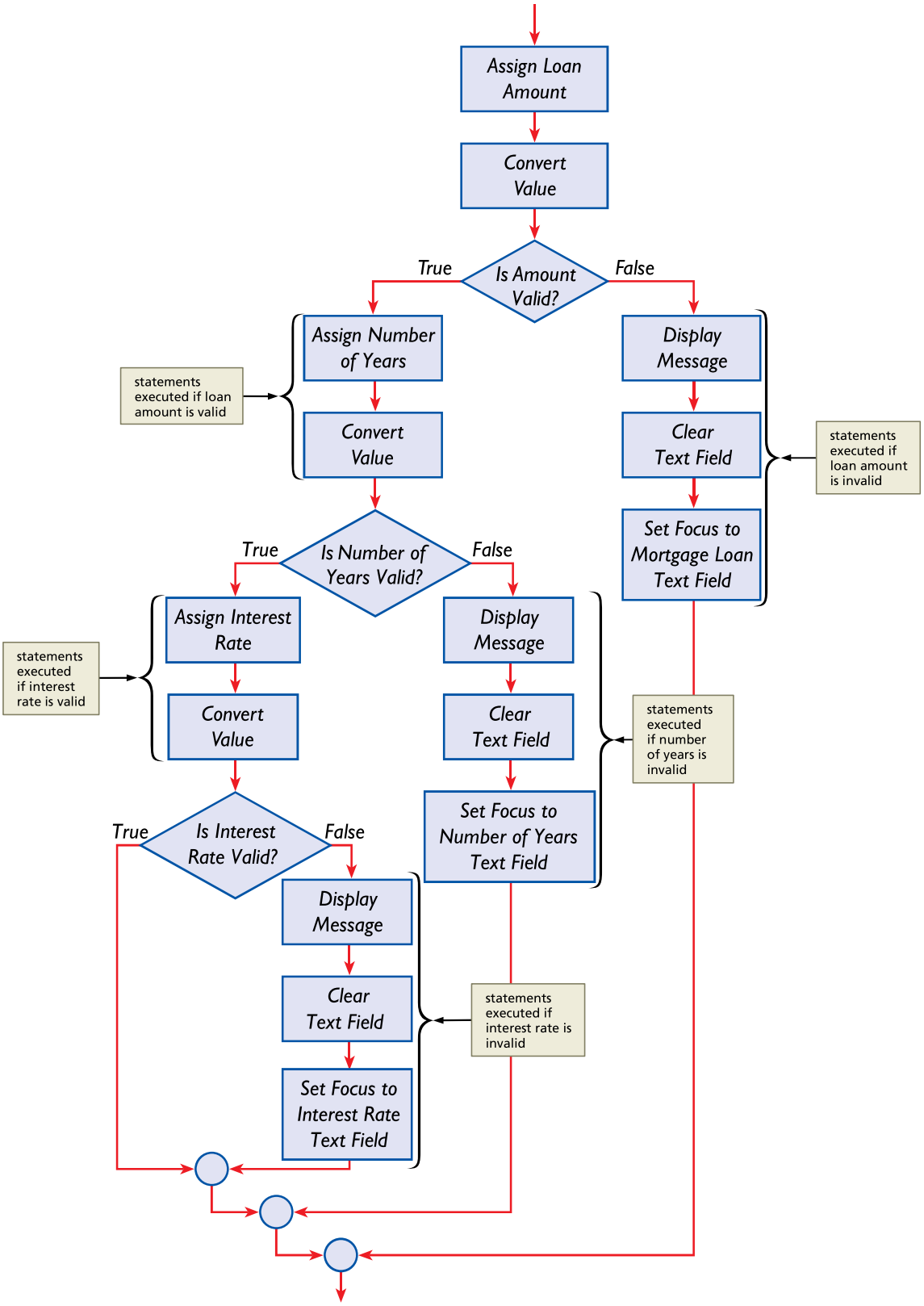

Assign Loan Amount
Convert Value
Is Amount Valid?
True
False
Assign Number of Years
Convert Value
statements executed if loan amount is valid
Display Message
Clear Text Field
Set Focus to Mortgage Loan Text Field
statements executed if loan amount is invalid
Is Number of Years Valid?
True
False
Assign Interest Rate
Convert Value
statements executed if interest rate is valid
Display Message
Clear Text Field
Set Focus to Number of Years Text Field
statements executed if number of years is invalid
Is Interest Rate Valid?
True
False
Display Message
Clear Text Field
Set Focus to Interest Rate Text Field
statements executed if interest rate is invalid

Figure 10–16

This validation design is necessary because of the event-driven nature of JavaScript. When a user triggers an event that calls a function, processing stays within that function until all statements execute. Because all the statements execute in a function, the form validation routine uses nested if…else statements to ensure each text field is validated correctly. By nesting if…else statements, you can place an if…else statement inside another as shown in Figure 10–17.

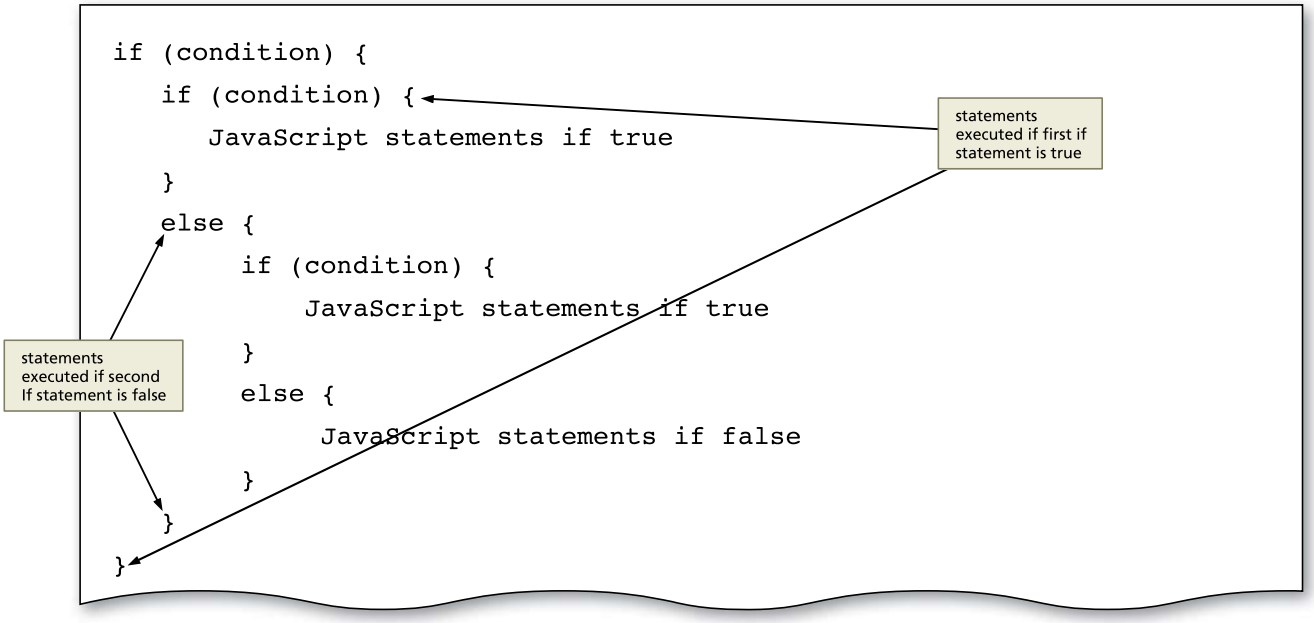


Figure 10–17

BTW

Radix or Number Base
Radix is the number base to which the integer value should be converted. The use of the numeral 2 represents binary, 8 represents octal, and 16 represents hexadecimal.

Using Built-In Functions to Validate Data When validating data, the JavaScript code may have to evaluate several criteria — for example, to ensure that a text field is not blank or that it contains numeric data (not text or characters). JavaScript accepts data entered into a text field as text character data, which means that the values must be converted to a number before they can be tested or validated. Table 10–8 describes the two built-in functions (parseInt() and parseFloat()) used to convert values and one function (isNaN()) used to test if the converted value is a number.

BTW

The parseFloat() Function
The parseFloat() built-in function parses a string argument and converts the value into a decimal floating-point number. If the first character cannot be converted to a number, the result is NaN, which means "not a number."

Table 10–8 Built-In Functions: parseInt(), parseFloat(), isNaN()

General form:	`variable = parseInt(value, base)`
Comment:	converts to an integer. Value is any string, which can be a variable or literal; base is the number base to which you want the string converted. A base of 2 means binary base number, an 8 means octal, and a 10 means decimal. The function returns an integer value, stripping the value after the decimal point.
Example:	`parseInt(loanAmount,10)`
General form:	`variable = parseFloat(value)`
Comment:	converts to a floating point number. Value is any string, which can be a variable or literal, representing a floating-point number. A floating-point number is one with a fractional or decimal value (including percentages). The function returns the value as a floating-point number.
Example:	`parseFloat(loanAmount)`
General form:	`isNaN(value)`
Comment:	isNaN means is Not a Number. Value is any value, which can be a variable or literal. The function returns a Boolean condition of true or false.
Example:	`isNaN(loanAmount)`

Figure 10–18 shows how the values of the form are passed to the Calc() user-defined function. Table 10–9 shows the general form of the JavaScript statement used to assign a null, or other, value to a text field object within a form.

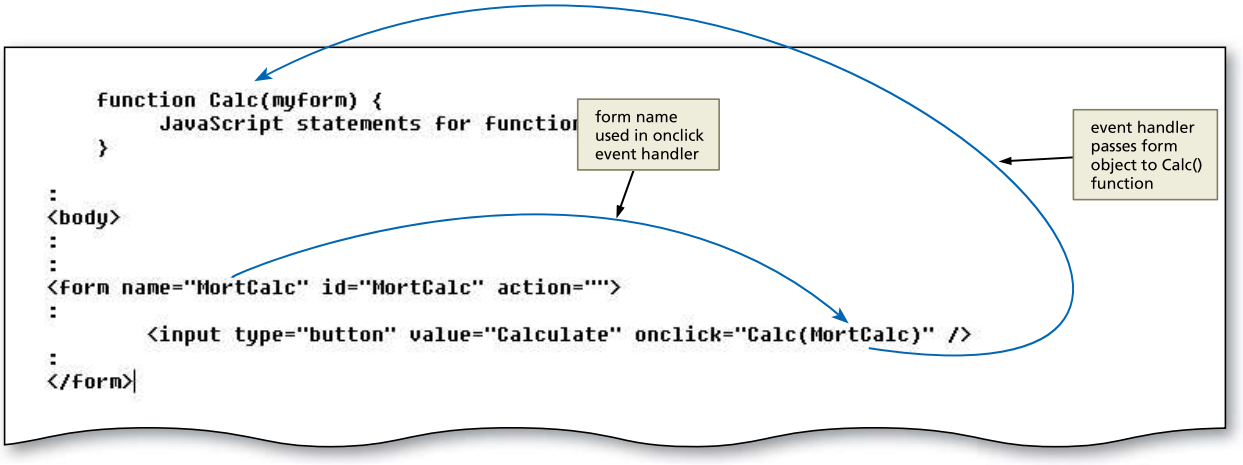


Figure 10–18

Table 10–9 Assignment Statement

General form:	`document.formname.textfieldname.value=variable_or_literal`
Comment:	where formname is the name of the form; textfieldname is the name of a text field in the form; value is the attribute; and the variable_or_literal is the value assigned to the text field.

Examples:	`document.MortCalc.Amount.value=LoanAmt`
	`document.MortCalc.Amount.value="12500"`
	`document.MortCalc.Amount.value=""`

To place the insertion point back in a specific text field in the form, the focus must be set for that text field. Setting the focus means giving attention to an object. JavaScript uses the **focus() method** to give attention to an object. When the focus is set to an object, such as the Amount text field, the JavaScript statement automatically positions the insertion point in the text field. Table 10–10 shows the general form of the focus() method.

Table 10–10 focus() Method

General form:	`document.formname.objectname.focus()`
Comment:	where formname is the name of the form that contains the object; and objectname identifies the object to which focus should be set.
Examples:	`document.MortCalc.Amount.focus()`

Table 10–11 shows the code to enter the Calc() user-defined function and the statements necessary to validate the mortgage loan amount using the parseInt() and isNaN() functions.

BTW

The isNaN() Built-in Function

The isNaN() function to test whether a value is not a number is the only function that tests a numeric value as the argument. The test uses the NOT operator and returns a Boolean value of true or false.

Table 10–11 Code for Calc() Function to Validate the Loan Amount

Line	Code
19	function Calc(myForm) {
20	var mortAmount=document.MortCalc.Amount.value
21	var mortAmount=parseInt(mortAmount,10)
22	if (isNaN(mortAmount) || (mortAmount<=0)) {
23	alert("The loan amount is not a valid number!")
24	document.MortCalc.Amount.value=" "
25	document.MortCalc.Amount.focus()
26	}

Line 19 declares the Calc() function and passes any values entered or selected in the MortCalc form to the function. Line 20 can assign the data entered in the Amount of Mortgage text field to the mortAmount variable. Line 21 converts that value to an integer. The if statement beginning on line 22 checks the condition to see if the value for the mortAmount variable is not a number or if the value entered is less than or equal to zero. If the result of the condition is true (that is, the value entered is not a number or it is a negative number), the function notifies the user with an alert message (line 23) that the loan amount is not valid; clears the data entered in the Amount of Mortgage text field (line 24); and then sets the focus back to the Amount of Mortgage text field (line 25). The brace in line 26 closes the if statement.

To Start the Calc() Function and Nested if…else Statements to Validate Form Data

The following step illustrates how to enter the Calc() user-defined function and the if statement that validates the loan amount value entered in the Amount of Mortgage text field.

1

- If necessary, click the chapter10 statewiderealty.html - Notepad button on the taskbar to display the Notepad window.
- Click line 19.
- Press the ENTER key once to create a blank line, and then position the insertion point on the blank line (line 19).

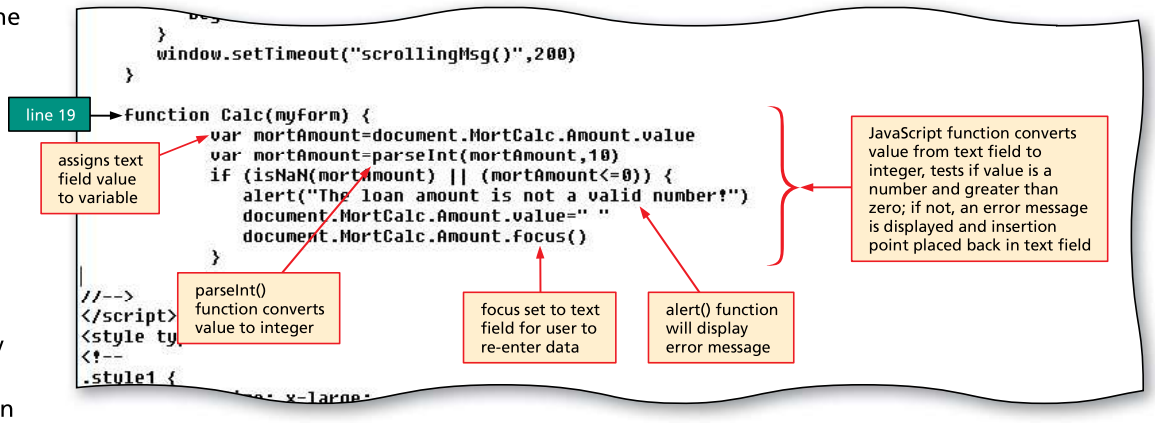


Figure 10–19

- Enter the JavaScript code shown in Table 10–11 using the SPACEBAR to indent as shown to start the Calc() user-defined function, define the variables for the loan amount, convert the loan amount to numeric values, and validate that the values are numeric (Figure 10–19).

Completing the Validation and Adding the Event Handler

Because an event must be executed until completion, the Calc() function validates all the entered values. Table 10–12 shows the code to validate the Interest Rate as % text field using the parseFloat() and isNaN() functions. If the interest rate data is valid, the function proceeds to validate the value in the Number of Years field and convert it to a floating-point number.

Table 10–12 Code to Validate the Interest Rate

Line	Code
27	`else {`
28	`var mortRate=document.MortCalc.Rate.value`
29	`var mortRate=parseFloat(mortRate)`
30	`if (isNaN(mortRate) || (mortRate<=0)) {`
31	`alert("The interest rate is not a valid number!")`
32	`document.MortCalc.Rate.value=" "`
33	`document.MortCalc.Rate.focus()`
34	`}`

Line 27 is an else statement that executes if the mortAmount data is valid and the function should proceed to validate the data entered in the Interest Rate as % text field. Line 29 passes the value in the Interest Rate as % text field to the mortRate variable, and converts it to a floating-point number using the parseFloat() function. Because the interest rate is a floating-point number, you must use the parseFloat() function to keep the interest rate a floating-point number.

The if statement in line 30 tests the mortRate variable to determine if it is a number or if the value is less than or equal to zero. If the result of the condition is true, an alert message (line 31) notifies the user that the interest rate is not valid. Line 32 clears the data entered in the Interest Rate as % text field, and then line 33 sets the focus back to the Interest Rate as % text field. The brace in line 34 closes the if statement. If the mortRate data is valid, the function then proceeds to validate the Number of Years.

Table 10–13 shows the code used to validate the value entered in the Number of Years text field.

Table 10–13 Code to Convert and Validate the Years Entered Value

Line	Code
35	`else {`
36	`var mortYears=document.MortCalc.Years.value`
37	`var mortYears=parseInt(mortYears,10)`
38	`if (isNaN(mortYears) || (mortYears<=0)) {`
39	`alert("The number of years is not a valid number!")`
40	`document.MortCalc.Years.value=" "`
41	`document.MortCalc.Years.focus()`
42	`}`
43	`}`
44	`}`
45	`}`

Line 35 is an else statement that executes the statements if the if condition on line 30 is false. Line 37 converts years to an integer, using the parseInt() function. The if statement beginning in line 38 checks the condition to determine if the mortYears value is greater than zero. If the number of years is not valid, line 39 displays a message and line 40 places the focus back in the Years text field. The braces in lines 42 through 45 close the nested if…else statements and the function.

To End the Nested if…else Statements to Validate Form Data

The following step shows how to enter the else portions of the nested if…else statements in the Calc() function to validate the Interest Rate as % text field and the Number of Years.

1

- If necessary, click line 27.
- Enter the JavaScript code shown in Table 10–12 to validate the interest rate, using the SPACEBAR to indent the code as shown in Figure 10–20.
- Press the ENTER key.
- Continue on line 35.
- Enter the JavaScript code shown in Table 10–13 to validate the number of years for the loan, using the SPACEBAR to align the code as shown.
- Press the ENTER key to finish the else portions of the nested if...else statements (Figure 10–20).

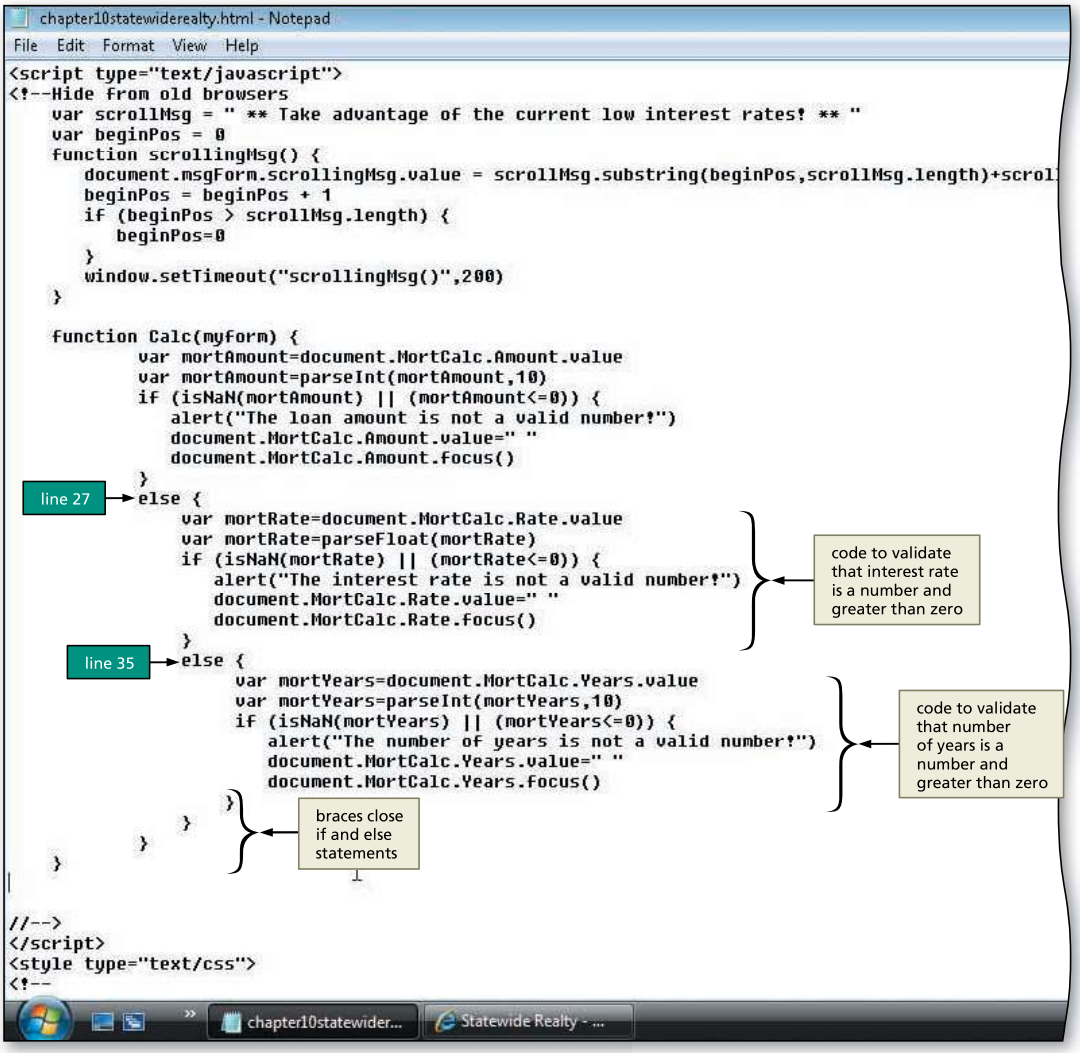


Figure 10–20

Q&A

Why is the year not converted to a floating point number?

Most loans, especially mortgage loans, are NOT made on part of a year, so the number of years should be an integer.

To Enter an onclick Event Handler to Call the Calc() Function

The last step in adding form validation to the mortgage loan payment calculator is to add an event handler to trigger the Calc() function when the user clicks the Calculate button. After entering data in the form, a user clicks the Calculate button, which triggers the Calc() function to validate the data entered in the form, using the if…else statements and built-in functions entered in previous steps. The following step shows how to enter the onclick event handler to call the Calc() function.

1

- Scroll down to the HTML code for the form and then click line 101, right after the closing quote in "Calculate" and before the rightmost /> bracket.
- Press the SPACEBAR once.
- Type `onclick="Calc(MortCalc)"` to add the event handler to the Calculate button, but do not press the ENTER key (Figure 10–21).

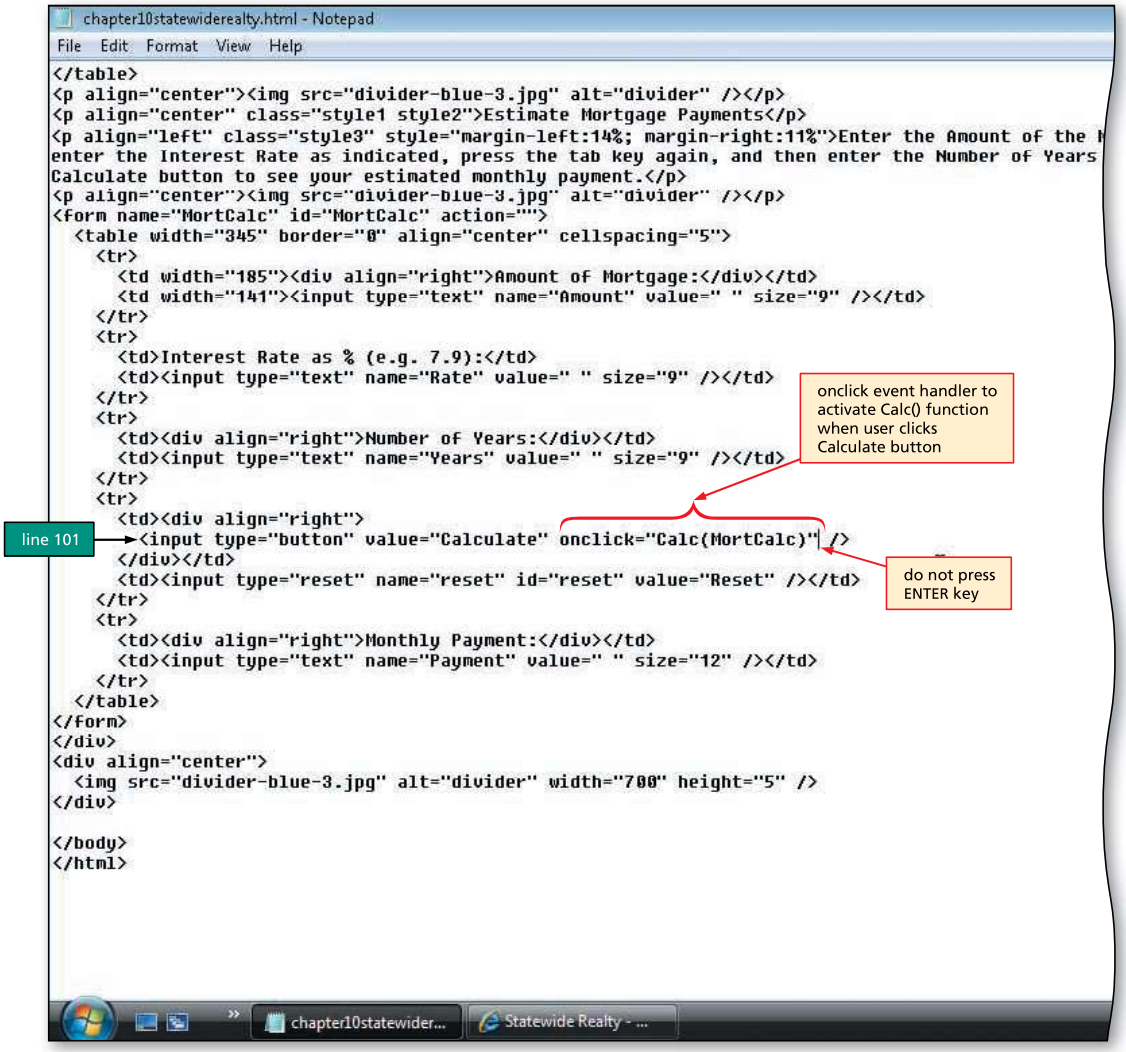


Figure 10–21

To Save an HTML File and Test a Web Page

With the JavaScript code for the form validation entered, the Web page can be saved and tested in a browser. The Calc() function will validate the text field entries, but will not yet calculate the monthly payment. The following step shows how to save the HTML file and test the Web page.

1

- With the USB drive plugged into your computer, click File on the menu bar and then click Save.
- Click the browser button on the taskbar.
- Click the Refresh button on the browser toolbar.
- When the Web page is displayed, click the Amount of Mortgage text field.
- Enter test data set 1, as shown in Table 10–14.
- Press the TAB key to move the insertion point to the next text field.
- When you have entered test data set 1, click the Calculate button at the bottom of the form (Figure 10–22).

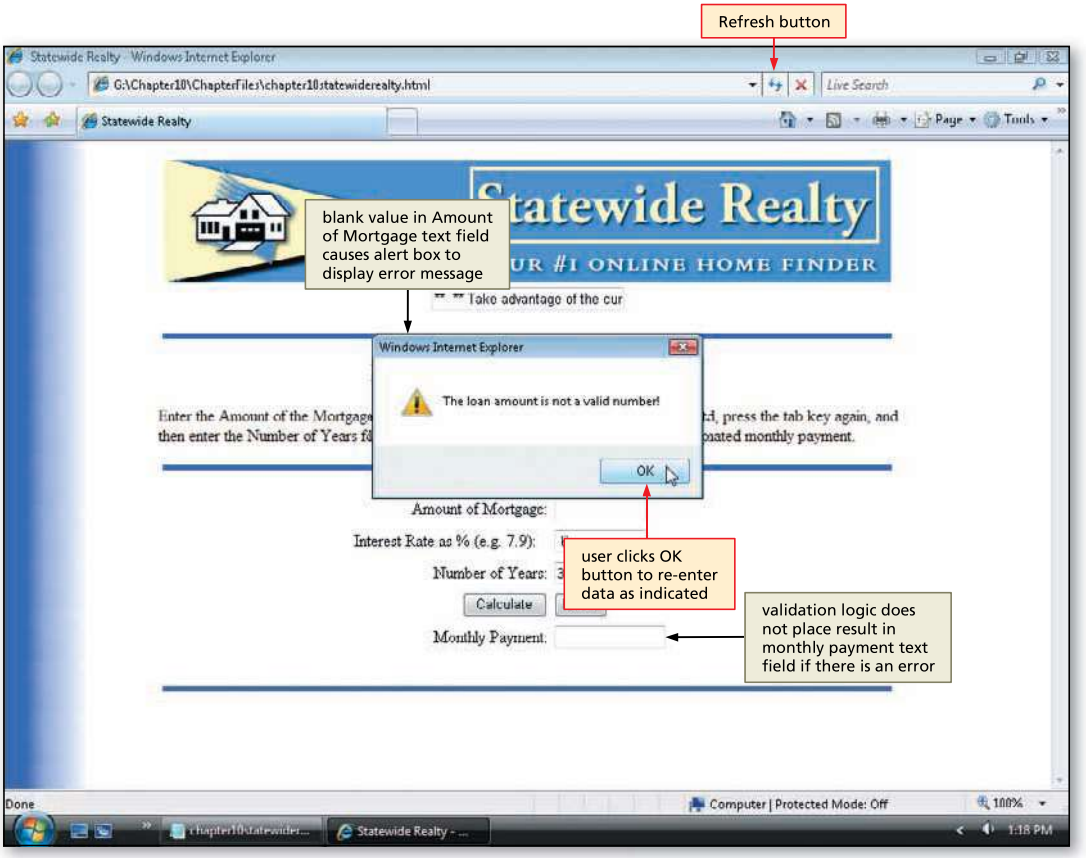


Figure 10–22

- When the message box is displayed, click the OK button.
- Click the Reset button at the bottom of the form.
- Repeat Steps 5 through 9, using test data sets 2, 3, and 4, as shown in Table 10–14.

Table 10–14 Test Data Set

Data Set	Amount of Mortgage	Interest Rate %	Number of Years	Comment
1		6	30	The loan amount is not a valid number!
2	109000	A	4	The interest rate is not a valid number!
3	99000	6	30	No error messages
4	193000	5.9	-30	The number of years is not a valid number!

Plan Ahead

Calculating the monthly payment.
The monthly() function requires three parameters: the mortgage loan amount (mortAmount), the interest rate (mortRate), and the number of years that the payments will be made (mortYears). These values are passed from the Calc() user-defined function to the monthly() function. The steps to calculate the monthly payment are as follows:

- The function call statement passes the three variables — mortAmount, mortRate, and mortYears — to the monthly() function.
- Convert the monthly interest rate to Irate by dividing mortRate by 1200
- Convert the number of years to Pmts by multiplying mortYears by 12
- Calculate the monthly payment with the following formula:

  mortAmount * (Irate / (1 - (1 / Math.pow(1+Irate,Pmts))))
- Return the monthly payment as a fixed decimal value to two decimal places using the toFixed(2) method.

Adding the Monthly Payment Calculation

With the JavaScript code for the form validation complete, the next step is to add code to the Calc() function to calculate the monthly payment. First, a statement must be added to the Calc() function to call a user-defined function, named monthly(), which calculates the monthly payment. The monthly() function uses the valid data in the form and calculates the monthly payment. The result is the monthly payment, which is returned as a floating-point value.

The placement of the monthly() function within the Calc() function is important so that, if a value in a text field is invalid, the function does not attempt to process invalid data and return an undefined result. To place this function properly, one more else statement must be added to the Calc() function, as shown in Table 10–15.

Table 10–15 Code to Call the monthly() Function

Line	Code
43	`else {`
44	`var mortPayment=monthly(mortAmount,mortRate,mortYears)`
45	`document.MortCalc.Payment.value=mortPayment`
46	`}`

Line 43 adds an additional else statement to the nested if…else statements. Line 44 calls the monthly() function and passes the loan amount, interest rate, and number of years for the loan as variables: mortAmount, mortRate, and mortYears. The result is stored in a temporary variable named monthlyPmt. Line 45 assigns the result to the Monthly Payment text field on the form. Line 46 is the closing brace for the additional else statement.

To Enter Code to Call the monthly() Function

The following step illustrates how to enter the final else statement and the function call that passes the required values to the monthly() function.

1

- Click line 43 and then press the ENTER key to insert a blank line.
- Click the blank line just inserted (line 43).
- Press the SPACEBAR to indent under the closing brace in line 42, then enter the JavaScript code shown in Table 10–15 to call the monthly user-defined function and assign the result to the payment text field but do not press the ENTER key (Figure 10–23).

Q&A

If I try to execute this Web page now, will an error occur?

Yes, because the monthly() user-defined function has not been written and entered.

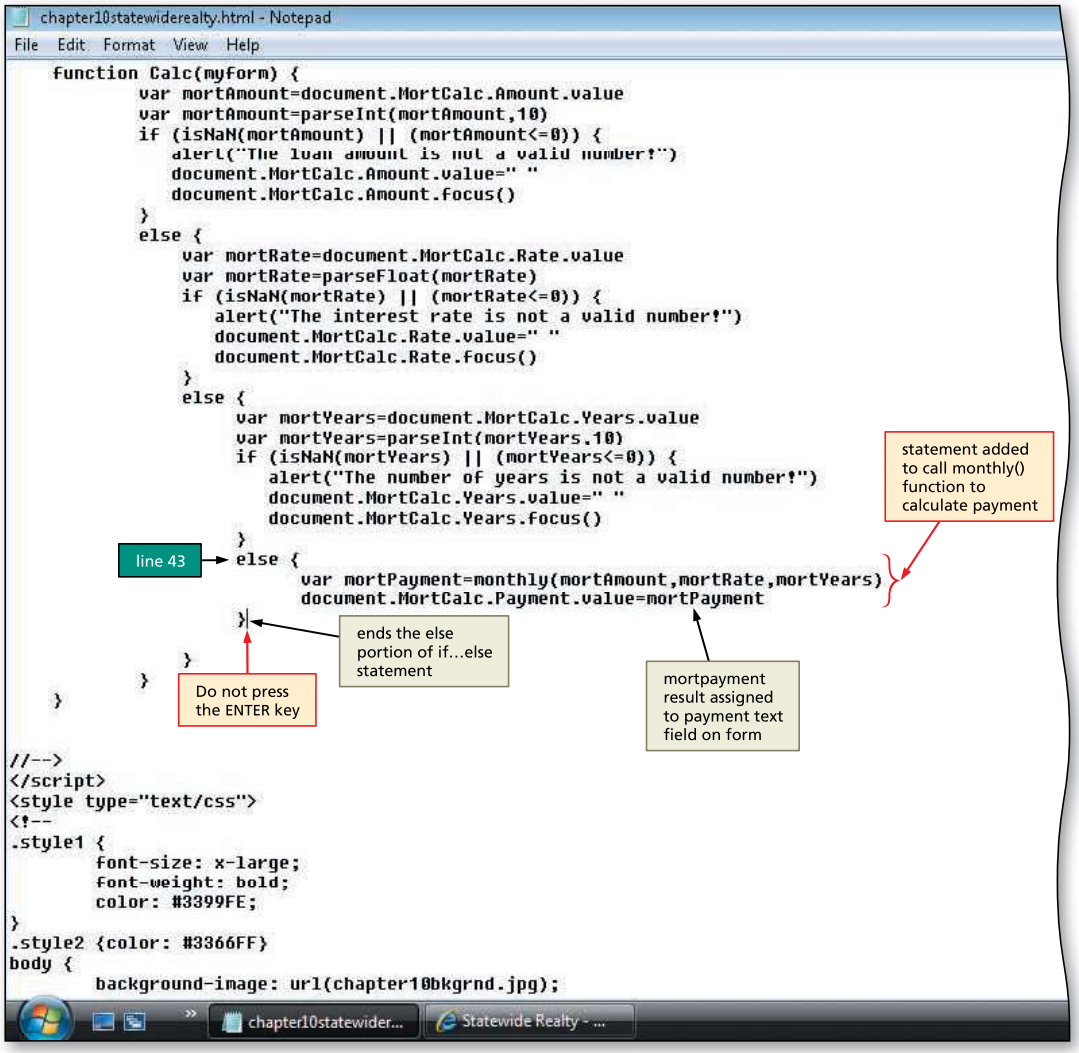


Figure 10–23

Creating the monthly() User-Defined Function The monthly() function is a user-defined function to calculate the monthly payment amount. The JavaScript code for the monthly() function is shown in Table 10–16.

Table 10–16 Code for monthly() User-Defined Function

Line	Code
51	function monthly(mortAmount,mortRate,mortYears) {
52	var Irate=mortRate/1200
53	var Pmts=mortYears*12
54	var Amnt=mortAmount * (Irate / (1 - (1 / Math.pow(1+Irate,Pmts))))
55	return Amnt.toFixed(2)
56	}

Line 51 declares the monthly() function. Line 52 determines the monthly interest rate percentage by dividing the annual rate by 1200. The result is assigned to the Irate (interest rate) variable. Line 53 determines the number of monthly payments on the loan, by multiplying the number of years in the loan by 12. The resulting value is assigned to the Pmts variable.

Line 54 is the formula for calculating a monthly payment based on the amount of the loan, the monthly interest percentage, and the number of monthly payments. The mathematical representation of the formula is:

$$\text{loan amount} * (\text{monthly interest rate} / (1 - (1 / (1 + \text{monthly interest rate})^{\text{number of payments}})))$$

JavaScript, however, does not use typical programming language symbols to represent exponentiation in code. Instead, to calculate the expression $(1 + \text{monthly interest rate})^{\text{number of payments}}$, JavaScript uses the pow() method associated with the Math object. Table 10–17 shows the general form of the pow() method.

Table 10–17 Math.pow() Method

General form:	`Math.pow(number, exponent)`
Comment:	where number is the value raised to the power of the exponent value. The pow() method accepts variables (X,n), constants (2,3), or both (Sidelength,2).

Examples:	`Math.pow(2,3)`
	`Math.pow(X,n)`
	`Math.pow(Sidelength,2)`

BTW

The Math Object
The Math object cannot be used to create other objects. Most of the properties of the Math object return preset values. Other properties really are methods and act as functions.

The return statement in line 55 tells the function to send the results of the expression back as a fixed decimal value with a length of two. The Number object's toFixed() method returns a value set to a specific decimal length, as shown in Table 10–18.

Table 10–18 Number.toFixed() Method

General form:	`Number.toFixed(digits)`		
Comment:	where digits is the exact number of digits after the decimal point. The number is rounded or padded with zeros if necessary.		
Examples:	Pmt = 234.8932 Amt = 843.6778	Pmt.toFixed(3) Amt.toFixed(2)	Result: 234.893 Result: 843.68

To Create the monthly() Function

The following step illustrates how to enter the monthly() user-defined function to calculate the monthly payment on a mortgage loan.

1

- Click line 51.
- Position the insertion point on the blank line directly above the //--> tag. (If necessary, insert a blank line above the tag.)
- Enter the JavaScript code shown in Table 10–16 to write the code to calculate the monthly payment and then press the ENTER key twice (Figure 10–24).

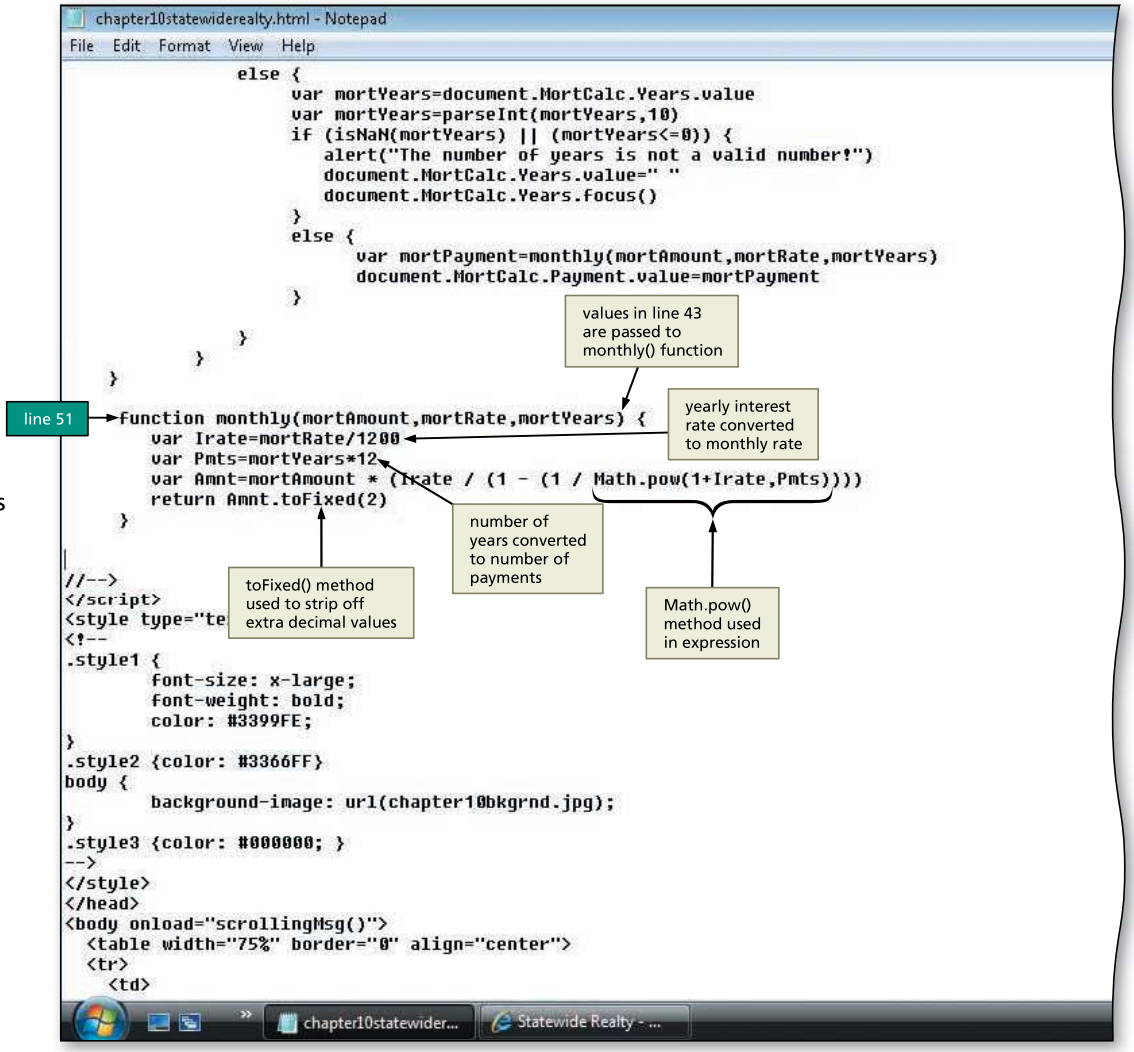


Figure 10–24

BTW

Math.round() Method

The Math.round() method returns the nearest integer value of a floating-point number. Thus, 28.453 will return 28. By multiplying the original number by 10 raised to the power of the number of decimals needed and then dividing by the same number, an original floating-point number also can be changed. For example, Math.round(9.453*100)/100 returns 9.45 and Math.round(9.454*10)/10 returns 9.5.

To Save an HTML File and Test a Web Page

Now that the monthly payment can be calculated, this is a good place to test the Web page. The following step shows how to save the HTML file and test the Web page.

1

- With the USB drive plugged into the computer, click File on the menu bar and then click Save.
- Click the browser button on the taskbar.
- Click the Refresh button on the browser toolbar.
- If necessary, click the Amount of Mortgage text field to place the insertion point in the text field.
- Type `73000` in the Amount of Mortgage text field and then press the TAB key.
- Type `6` in the Interest Rate as % (e.g. 7.9) text field and then press the TAB key.

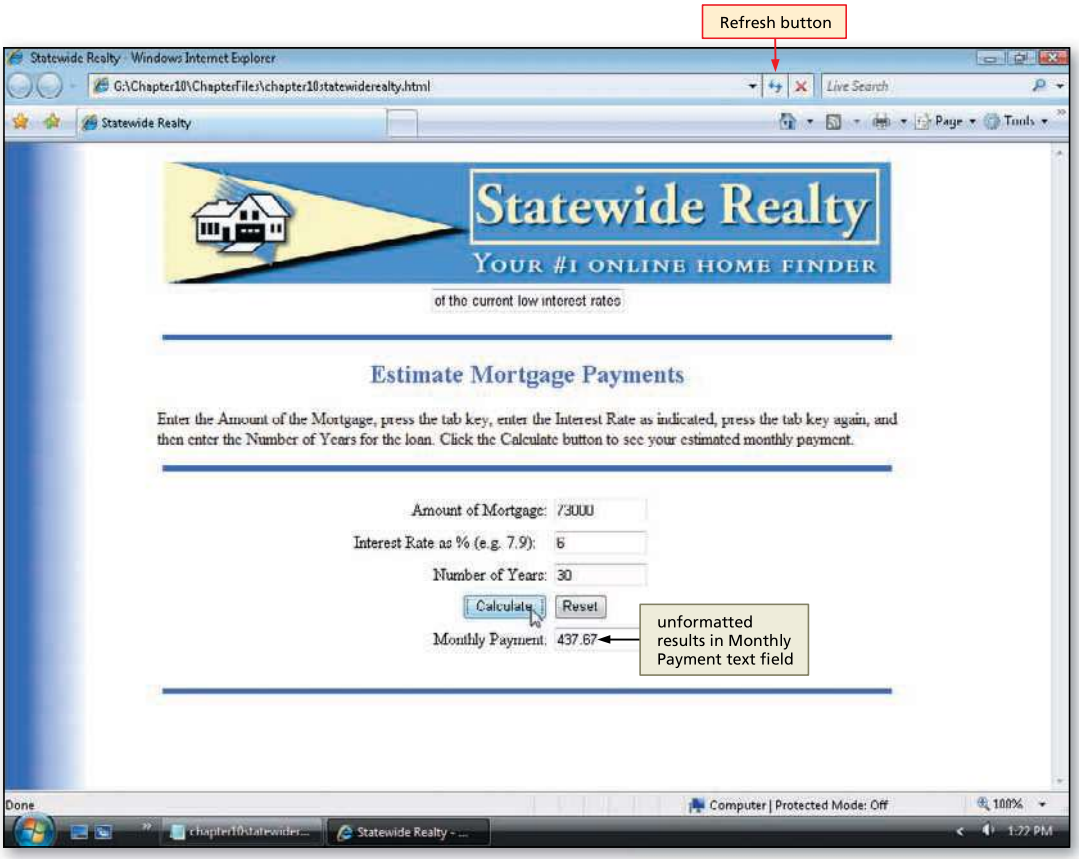


Figure 10–25

- Type `30` in the Number of Years text field and then click the Calculate button (Figure 10–25).

Q&A

What if my result was slightly different? Did I do something wrong?

If your result is only a few pennies different, it is probably just a difference in the math processor on your computer. If the value is hundreds or even thousands of dollars off, check your formulas on lines 52 through 54.

Formatting the Monthly Payment Output as Currency

As shown in Figure 10–25, the mortgage loan payment calculator currently displays the monthly payment amount as a value with two decimal places. To set the form to display the monthly payment amount in a currency format with a dollar sign, the dollarFormat() function is used. First, you must enter a statement that passes the resulting string object of the monthly payment Calc() function to the dollarFormat() function. The dollarFormat() function then analyzes the string, adds commas, and returns the number with a dollar sign and two decimal places.

Plan Ahead

Formatting output results.
To format the result, the dollarFormat() function performs these seven basic steps:

1. Use the string value passed to the function, which separates the dollars from the cents based on the position of the decimal point
2. Determine the location of the decimal point using the indexOf() method
3. Separate the value to the left of the decimal point as the dollar amount and the value to the right of the decimal point as the cents amount
4. Insert commas every three positions in dollar amounts exceeding 999
5. Reconstruct the result string value with two decimal places
6. Insert a dollar sign immediately to the left of the first digit without spaces
7. Return the completed formatted value

The value is assigned to the Monthly Payment text field in the form.

Using the indexOf() Method The **indexOf()** method is used to search a string for a particular value and returns the relative location of that value within the string. The indexOf() method searches the string object for the desired value, which is enclosed within the quotation marks. Table 10–19 shows the general form of the indexOf() method.

Table 10–19 indexOf() Method

General form:	`var position = stringname.indexOf("c")`
Comment:	where position is a variable; stringname is any string object; and "c" is the value for which the function searches.
Example:	`var decipos = valuein.indexOf(".")`

If the search value is found in the string object, the indexOf() method returns the relative position of the value within the string object. If the search value is not found, the indexOf() method returns a negative one (–1). In this chapter, the indexOf() method is used to search for a decimal point in the monthly payment amount. Figure 10–26 provides an example of how the indexOf() method works.

```
<form name="regForm">
    <input type="text" name="emailAddr" value="" />

value of emailAddr: someName@some_emailer.com
var tEmail=document.regForm.emailAddr.value
var atPos=tEmail.indexOf("@")
value of atPos: 8
```

value entered in text field

value passed to variable

indexOf() method searches for position of @ sign

result returned to calling statement

Figure 10–26

Beginning the dollarFormat() Function and Formatting the Dollars Portion The dollarFormat() function initializes the variable that will return the formatted value and the variable used to manipulate the unformatted value. Most programmers agree it is a good programming practice to clear and initialize variables to ensure the data is valid. Table 10–20 shows the JavaScript code used to add the dollarFormat() function.

Table 10–20 Code for the dollarFormat() Function

Line	Code
58	`function dollarFormat(valuein) {`
59	`var formatStr=""`
60	`var Outdollars=""`
61	`var decipos=valuein.indexOf(".")`
62	`if (decipos==-1)`
63	`decipos=valuein.length`

Line 58 declares the dollarFormat() function and the valuein variable. Lines 59 and 60 clear the variables used to assemble the formatted output by assigning null (or empty) values. The indexOf() method in line 61 returns a value indicating the location of the decimal point — a value stored in the decipos variable. This value indicates at what position to concatenate the decimal values. Line 62 tests the condition of the decipos variable. If the value of decipos is –1, then decipos is set equal to the length of the string, as shown in line 63. If the value of decipos is 0, then the input value (valuein) is an integer and no decimal values need to be modified.

To Enter the dollarFormat() Function

The following step shows how to enter the dollarFormat() function and initialize the variables.

1. Click line 58 above the closing comment //-->.
2. Enter the JavaScript code from Table 10–20 to begin the dollarFormat() function and then press the ENTER key (Figure 10–27).

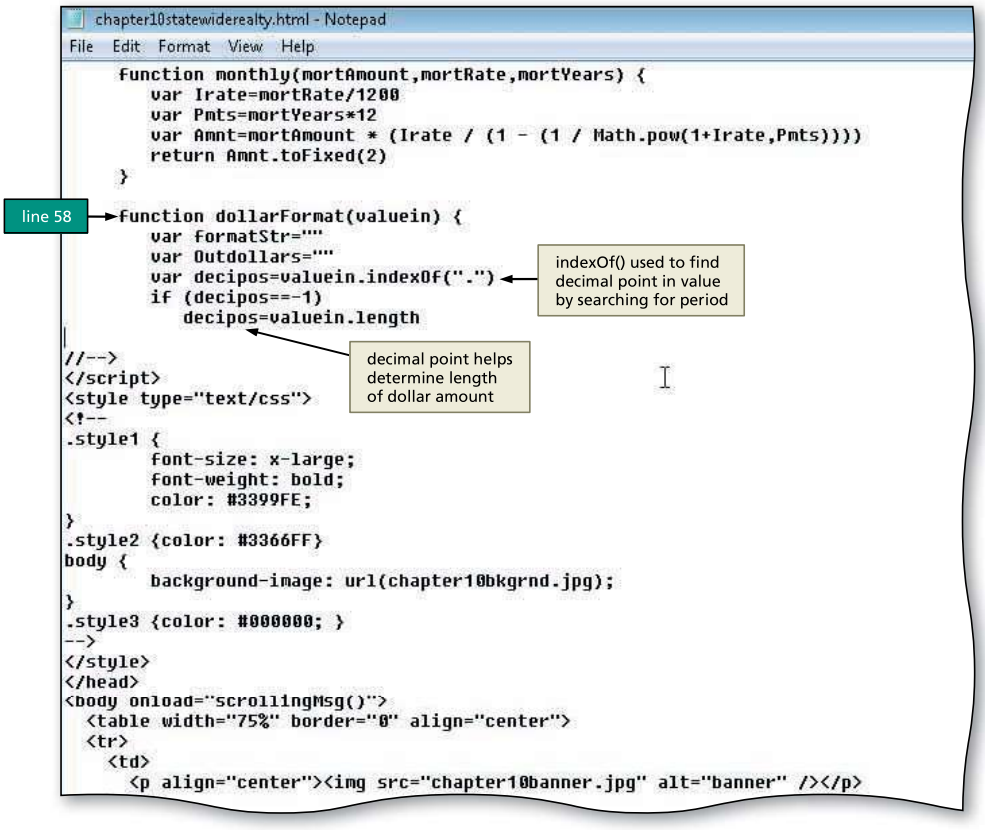


Figure 10–27

Plan Ahead

Using a while loop to insert commas every three digits in a number.
To place a comma every three digits, use a while loop. The following steps describe the logic of the while loop:

1. Extract three digits from the dollar value, starting from the right by subtracting 3 from the length of the dollar value (dollen).
2. Verify three digits have been subtracted and then insert a comma in the output string.
3. Decrement the length of the dollar value to look for the next group of three digits.
4. The process (loop) is complete when no more groups of three digits exist and the length of dollen is zero.

Using an if...else Statement and while Loop to Extract the Dollars Portion and Insert Commas A **loop** is a series of statements that executes repeatedly until it satisfies a condition. JavaScript has two types of loops: for loops and while loops. Both types of loops use the logic illustrated by the flowchart in Figure 10–28. Both loops first test a condition to determine if the instructions in the loop are to be executed.

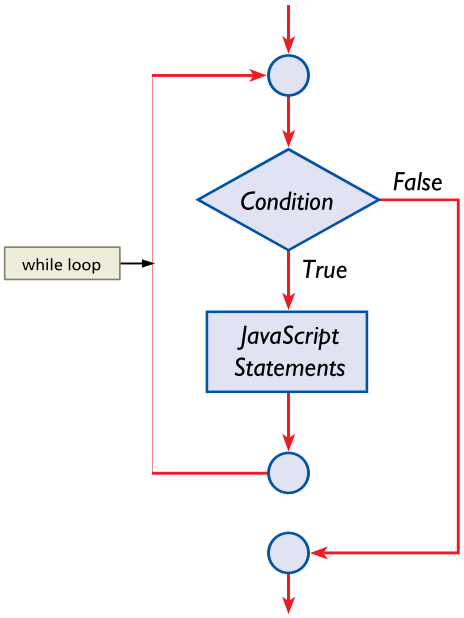


Figure 10–28

The **for loop** relies on a conditional statement using numeric values and thus often is referred to as a counter-controlled loop. Table 10–21 shows the general form of the for loop.

Table 10–21 for Loop

General form:	`for (start; stop; counter-control) {`
	`  JavaScript statements`
	`}`

Comment:	where start is a variable initialized to a beginning value; stop is an expression indicating the condition at which the loop should terminate; and the counter-control is an expression indicating how to increment or decrement the counter. Semicolons separate the three variables.

Examples:	`for (j=1; j<5; j++) {`
	`for (ctr=6; ctr>0; ctr--) {`
	`for (itemx=1; itemx<10; itemx=itemx+2) {`

The while loop relies on a conditional statement that either can use a numeric value or a string. Table 10–22 shows the general form of the while loop.

Table 10–22 while Loop

General form:	`while (condition) {`
	`  JavaScript statements`
	`}`

Comment:	where condition is either a numeric value or a string; and the JavaScript statements execute while the result of the condition is true.

Examples:	`while (ctr < 6) {`
	`while (isNaN(temp)) {`
	`while (Response != "Done") {`

In this chapter, the while loop is used in formatting the dollar value of the Monthly Payment value. The dollars portion is represented by the digits to the left of the decimal point. If the dollars portion of the mortgage loan payment contains more than three digits, commas need to be inserted. Table 10–23 shows the JavaScript statements used to determine the length of the dollar value and placement of the commas.

Table 10–23 Code for Determining the Length of the Dollar Value and Comma Placement

Line	Code
64	`var dollars=valuein.substring(0,decipos)`
65	`var dollen=dollars.length`
66	`if (dollen>3) {`
67	`  while (dollen>0) {`
68	`  tDollars=dollars.substring(dollen-3,dollen)`
69	`  if (tDollars.length==3) {`
70	`  Outdollars=","+tDollars+Outdollars`
71	`  dollen=dollen-3`
72	`  } else {`
73	`  Outdollars=tDollars+Outdollars`
74	`  dollen=0`
75	`  }`
76	`  }`
77	`if (Outdollars.substring(0,1)==",")`
78	`  dollars=Outdollars.substring(1,Outdollars.length)`
79	`else`
80	`  dollars=Outdollars`
81	`}`

The substring() method in line 64 uses the decipos value (the location of the decimal point) to extract the dollar value (the variable dollars). Next, a series of statements determines the length of the dollar value and then assigns the length to the variable dollen (line 65). Line 66 begins the if statement that determines if the dollar portion of the value is longer than three digits. The while loop routine (lines 67 through 76) places a comma every three digits, while the length of the dollar value is greater than three digits. Line 68 extracts three digits starting from the right by subtracting 3 from the length of the dollar value (dollen). Line 69 verifies three digits and line 70 inserts a comma in the output string. Line 71 decrements the length of the dollar value to look for the next group of three digits. When

no more groups of three digits exist, the length of dollen is set to zero (line 74) and the loop terminates at line 76. The statements in lines 77 through 80 prevent the code from inserting a comma if only three digits are to the left of the decimal point.

To Enter an if…else Statement and while Loop to Extract the Dollar Portion of the Output and Insert Commas

The following step illustrates how to enter the if…else statement and while loop to extract the dollar portion of the output and insert commas into the output if needed.

1

- If necessary, click line 64, the line directly below the statement, decipos=valuein.length.
- Enter the JavaScript code as shown in Table 10–23 on the previous page using the SPACEBAR to indent as shown to extract the dollar portion of the output and insert the appropriate commas and then press the ENTER key (Figure 10–29).

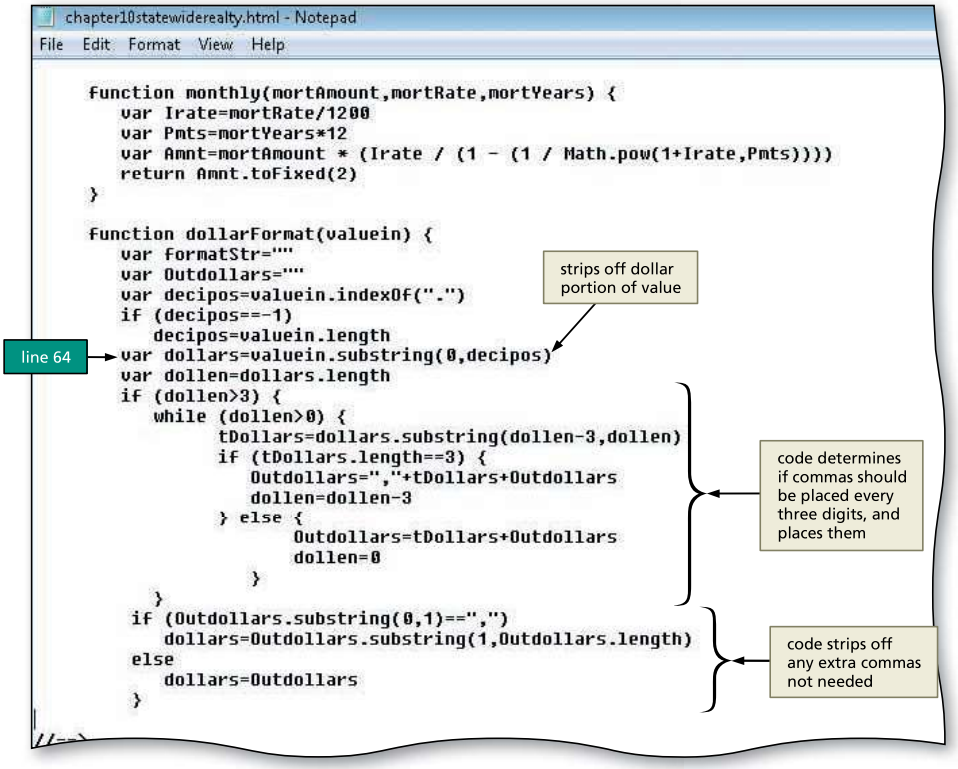


Figure 10–29

Reconstructing the Formatted Output and Returning the Formatted Value Next, the JavaScript statements must be written to reconstruct (concatenate) the formatted dollars and cents output into a formatted payment amount value, store the payment amount value in the formatStr variable, and return the formatStr value. Table 10–24 shows the statements needed to complete this task.

Table 10–24 Code for Reconstructing the Formatted Output and Returning the Formatted Value

Line	Code
82	var cents=valuein.substring(decipos+1,decipos+3)
83	var formatStr="$"+dollars+"."+cents
84	return formatStr
85	}

Line 82 extracts the two decimal places and assigns them to a variable to be used to reconstruct the formatted value. Line 83 reconstructs the values, concatenating a dollar sign ($) and the decimal value, and assigns the value to the formatStr variable. Line 84 returns the formatted value to the dollarFormat() function.

To Reconstruct the Formatted Output and Return the Formatted Value

The following step illustrates how to reconstruct the formatted output and return the formatted value to the calling function.

1

- If necessary, click line 82.
- Enter the JavaScript code from Table 10–24 using the SPACEBAR to indent as shown to reconstruct the formatted output and return the formatted value and then press the ENTER key (Figure 10–30).

chapter10statewiderealty.html - Notepad

File Edit Format View Help

```
    function dollarFormat(valuein) {
       var formatStr=""
       var Outdollars=""
       var decipos=valuein.indexOf(".")
       if (decipos==-1)
          decipos=valuein.length
       var dollars=valuein.substring(0,decipos)
       var dollen=dollars.length
       if (dollen>3) {
          while (dollen>0) {
                tDollars=dollars.substring(dollen-3,dollen)
                if (tDollars.length==3) {
                   Outdollars=","+tDollars+Outdollars
                   dollen=dollen-3
                } else {
                      Outdollars=tDollars+Outdollars
                      dollen=0
                   }
          }
        if (Outdollars.substring(0,1)==",")
           dollars=Outdollars.substring(1,Outdollars.length)
        else
           dollars=Outdollars
        }
        var cents=valuein.substring(decipos+1,decipos+3)
        var formatStr="$"+dollars+"."+cents
        return formatStr
    }
//-->
</script>
<style type="text/css">
<!--
.style1 {
        font-size: x-large;
        font-weight: bold;
        color: #3399FE;
}
.style2 {color: #3366FF}
body {
        background-image: url(chapter10bkgrnd.jpg);
}
.style3 {color: #000000; }
-->
</style>
```

line 82

code entered to reconstruct output with dollar sign and decimal value

closes function

chapter10statewider... Statewide Realty - ...

Figure 10–30

To Pass the Monthly Payment Value to the dollarFormat() Function

To have the monthly payment value appear in the Monthly Payment text field formatted as currency, it must be passed to the dollarFormat() function. Because the dollarFormat() function manipulates a string value, the monthly payment result first must be converted to a string using the toString() method. In Chapter 9, the toString() method was used to convert a date value to a string. In this chapter, the toString() method is used to convert the monthly payment to a string that the dollarFormat() function can manipulate.

The following step shows how to enter the JavaScript statements needed to pass the monthly payment as a string object to the dollarFormat() function.

1

- Scroll up to and click line 45 (the line that starts document. MortCalc).
- Highlight and delete the variable, mortPayment, after the = symbol.
- Type `dollarFormat (mortPayment. toString())` to replace mortPayment with the dollarFormat() function using the mortPayment value. Do not press the ENTER key (Figure 10–31).

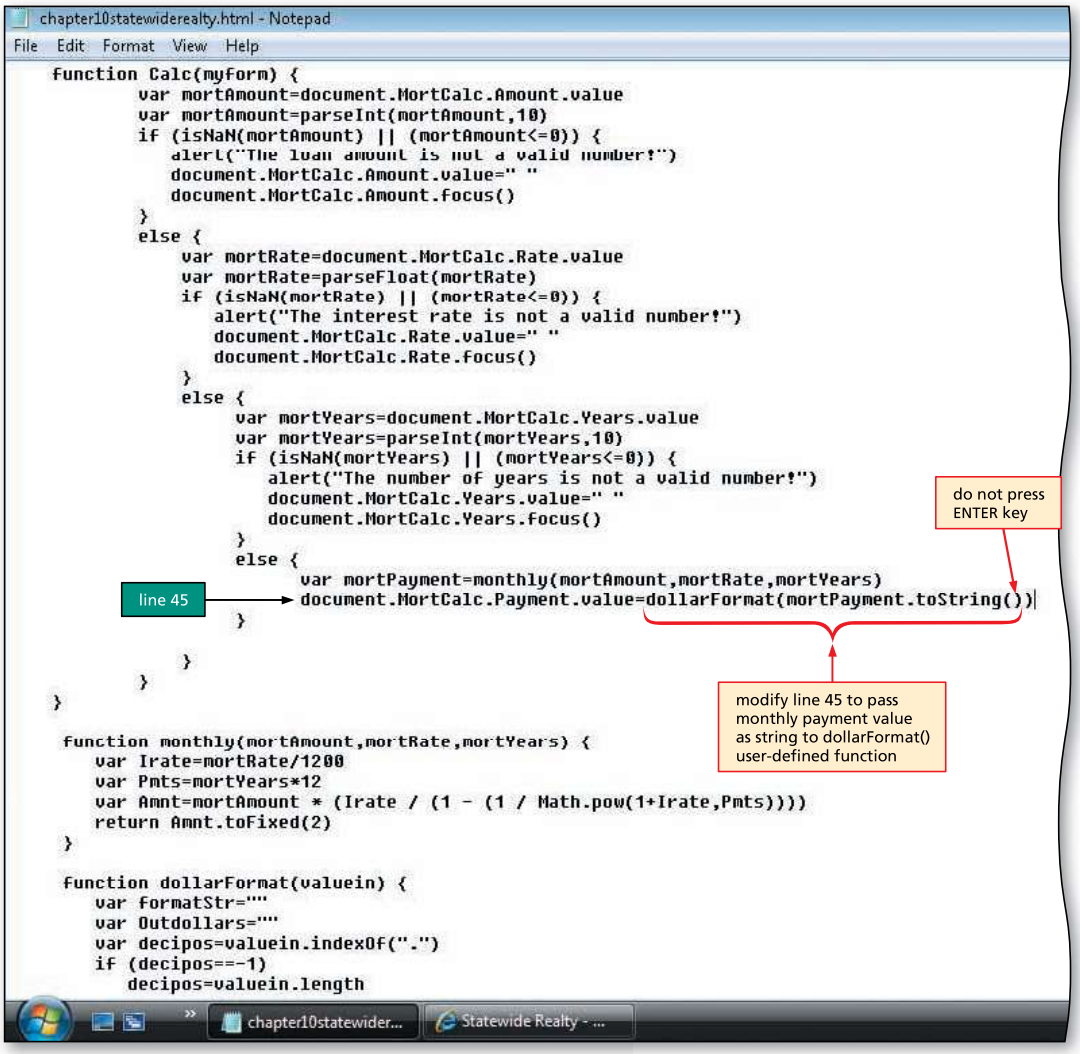


Figure 10–31

To Save an HTML File and Test a Web Page

The following step shows how to save the HTML file and test the Web page.

1

- With the USB drive plugged into your computer, click File on the menu bar and then click Save.
- Click the browser button on the taskbar.
- Click the Refresh button on the browser toolbar.
- Enter the test data as follows: Amount of Mortgage: `173000`; Interest Rate as % (for example, 7.9): `6`; and Number of Years: `30`.
- Click the Calculate button. The result should be formatted as shown in Figure 10–32.

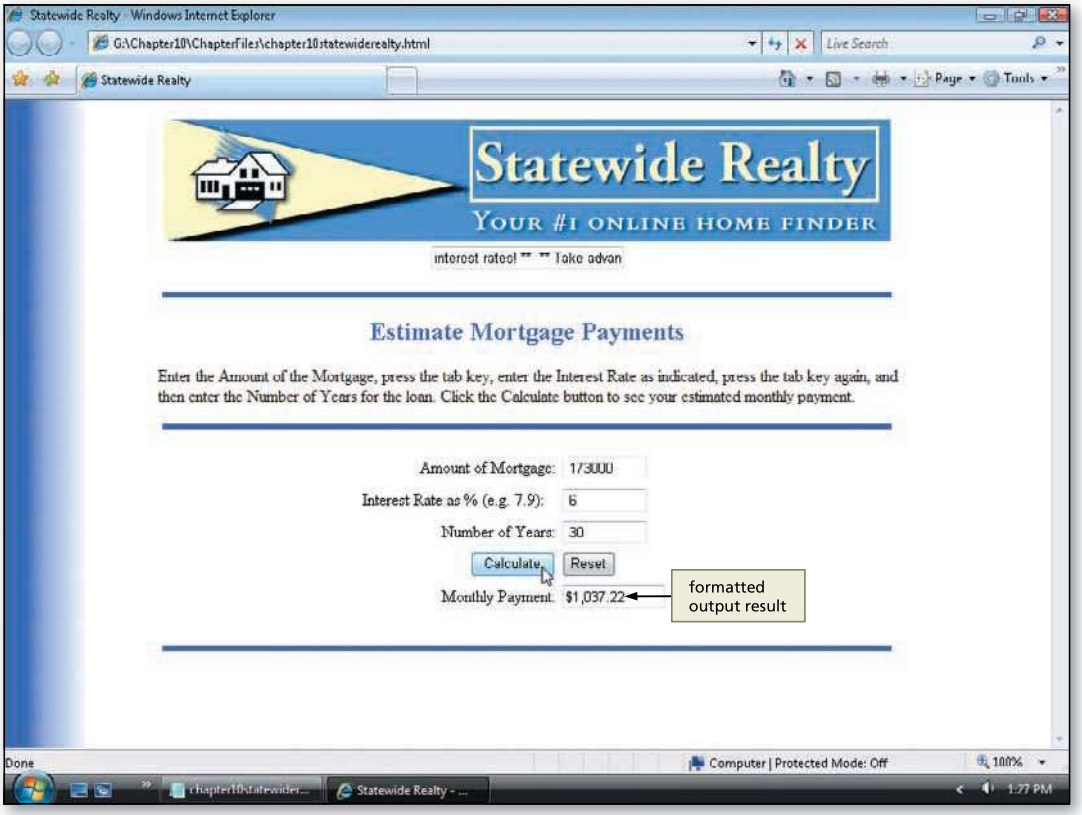


Figure 10–32

Planning pop-up windows.
A pop-up window is used to add additional information to Web page. You must decide what features you want the pop-up window to utilize. Ask yourself these questions:

- Do you want to include the status bar, title bar, address bar, scroll bars, toolbars, and menu bar?
- Do you want the user to be able to resize the window?
- What width and height should the window be?

Each of these properties can be set as properties in the open() method, creating a customized look for that particular pop-up window.

Plan Ahead

Adding a Pop-Up Window

As you have learned in this chapter, the alert() method is one way to display messages to a user. These message boxes, however, only display text on a gray background. To create more visually interesting messages, you can use JavaScript to open and display another HTML file in a separate window that displays colors, graphics, animations, and other media. Such a window is called a **pop-up window**, because it appears over the previously opened browser window.

The **open() method** is used to create a pop-up window. Table 10–25 shows the general form of the open() method.

Table 10–25 open() Method

General form:	`var windowname=open("window file name(URL)", "object name", "window features")`
Comment:	where windowname is an optional name of a window object (required only if you need to refer to the pop-up window in any other Web page); window file name is the name of the HTML file; and window features describe how the window should appear.
Examples:	`open("Adwindow.htm", "AdWin", "resize=off,titlebar=false")`

BTW

Pop-up Windows
In addition to alert() message boxes, Web developers use pop-up windows to display more information to gain attention. Use pop-up windows sparingly, because having many pop-up windows can annoy visitors.

As shown in Table 10–25, when adding the open() method to create a pop-up window, all of the pop-up window features must be enclosed within one set of quotation marks. Table 10–26 describes the more commonly used attributes of the open() method, which are used to define pop-up window features. For more information about the open() method, see the JavaScript Quick Reference in Appendix E.

Table 10–26 open() Method Attributes

Feature	Description	Written As	Comments
location	Includes address bar	"location=yes"	
menubar	Includes menu bar	"menubar=yes"	
resize	Allows user to resize	"resizeable=yes"	
scrollbars	Includes scroll bars	"scrollbars=yes"	
status	Includes status bar	"status=yes"	
titlebar	Removes title bar	"titlebar=false"	Default is true
toolbar	Includes toolbar	"toolbar=yes"	
width	States width in pixels	"width=220"	
height	States height in pixels	"height=450"	

In this chapter, the open() method is used to open a pop-up window that will display information about Statewide Realty services. You insert code for the popupAd function and open() method, and then add an event handler to call the popupAd() function when the page is loaded. Earlier in the chapter, the onload event handler was associated with the scrollingMsg() function. You also will use the onload event handler for the popupAd() function. Multiple functions can be associated with the same event handler.

Table 10–27 shows the code to create the user-defined function popupAd(). The chapter10notice.html file already has been created and is stored in the Chapter10\ChapterFiles folder of the Data Files for Students.

Table 10–27 Code to Open chapter10notice.html Pop-Up Window

Line	Code
85	`function popupAd() {`
86	`open("chapter10notice.html","noticeWin","width=550,height=360")`
87	`}`

Line 85 declares the popupAd() function. The statement in line 86 opens the chapter10notice.html Web page as a pop-up window that is 550 pixels wide and 360 pixels high.

To Enter the popupAd() Function to Open a Pop-Up Window

The following step illustrates how to enter the popupAd() user-defined function that contains the open() method to open the chapter10notice.html file in a pop-up window.

1

- Click line 85, the line directly above the //--> tag.
- Enter the JavaScript code from Table 10–27 to open a pop-up window and press the ENTER key (Figure 10–33).

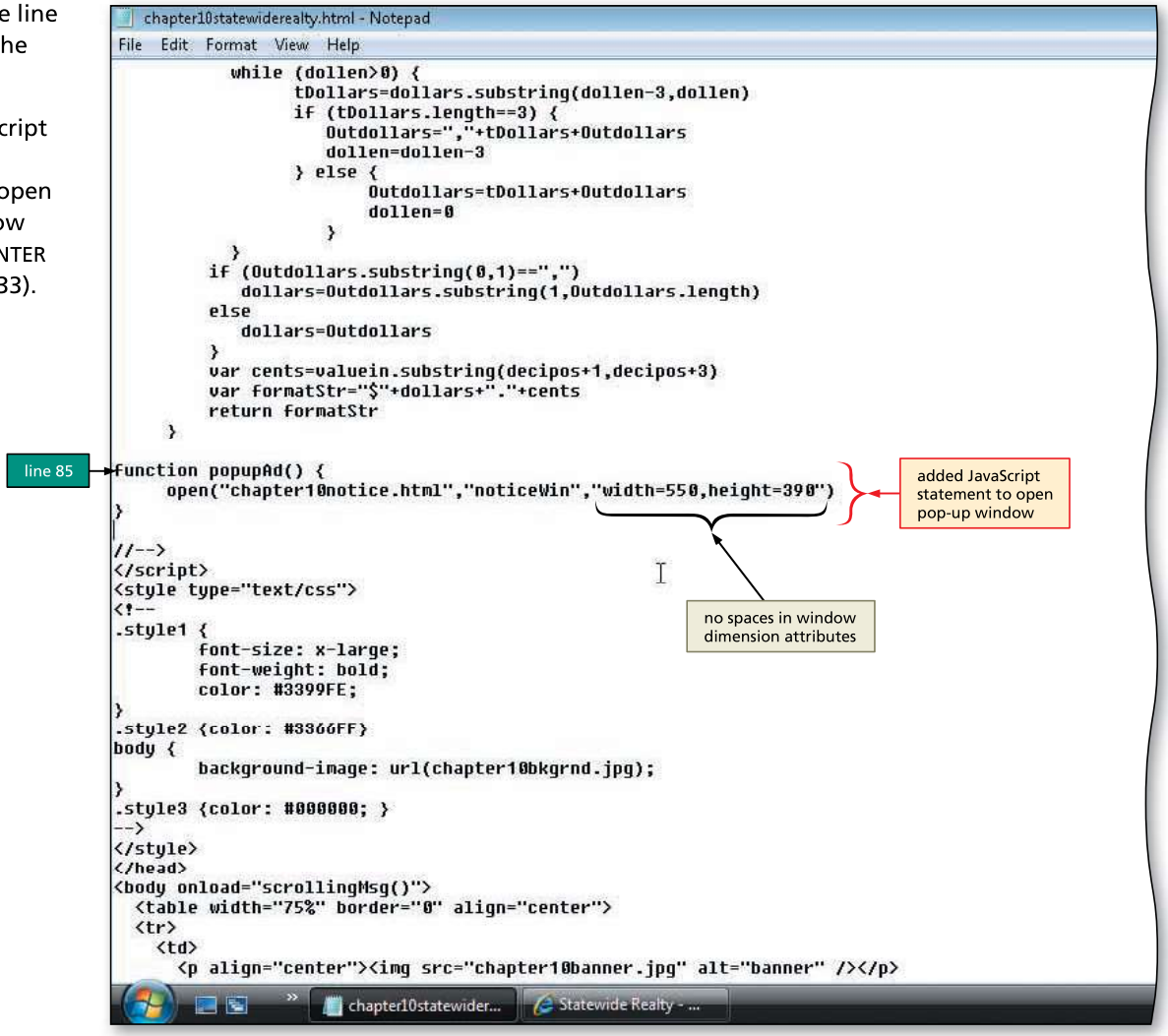


Figure 10–33

To Add the Event Handler to Call the popupAd()Function

Now we need to add the onload event handler to open the pop-up window when the Web page loads. Each function name is enclosed in one set of quotation marks and separated by a semicolon. The following step illustrates how to add the second function call to the onload() event handler.

1

- Position the insertion point in front of the "> at the end of line 106 (the line that begins <body onload=) (Figure 10–34).

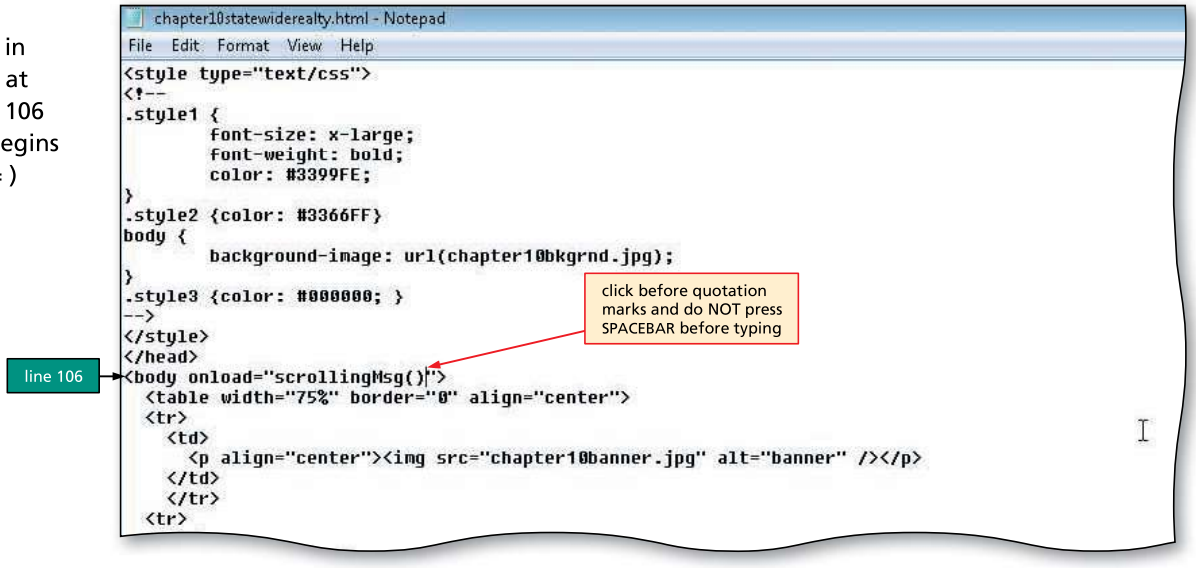


Figure 10–34

- Type ; popupAd() to add the call to the popupAd() user-defined function to the left of the second quotation mark (Figure 10–35).

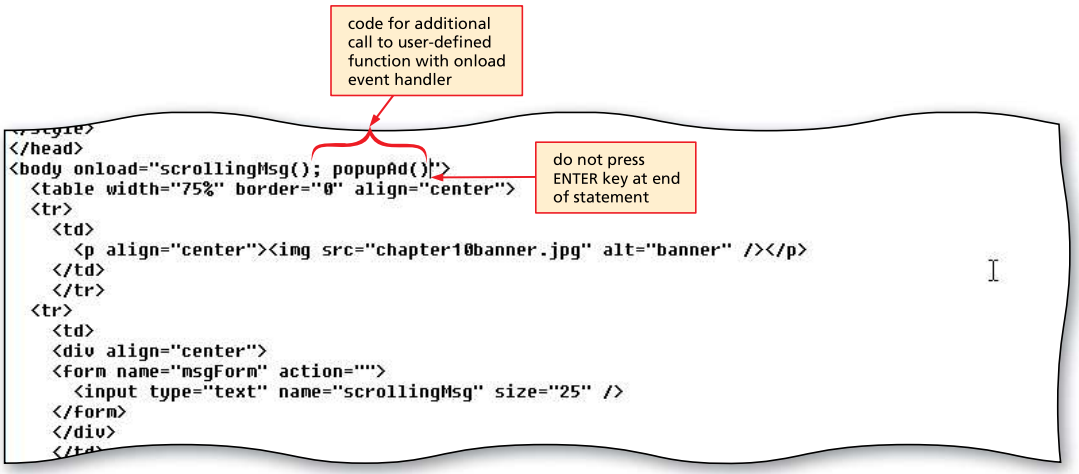


Figure 10–35

Q&A

Is this the only way to open a pop-up window?

An open() method can be placed anywhere in the <script> section in the <head> section. Once the <head> section loads in the browser, it will execute the "stand-alone" open() method, opening a pop-up window. The code must be in the <head> section, however, in order for this to work properly.

Adding the Date Last Modified

As you learned in Chapter 9, the purpose of displaying the date a Web page was last modified is to make sure the user knows how current the information is contained on the Web page. You added JavaScript code to the Westlake Landscaping Web page to display the date and time a file was last modified. The Statewide Realty Web page should include similar JavaScript code to display just the date the Web page was modified, not

the time. To display just the date, the code in Table 10–28 uses the substring() method to grab just the date. The parameters used in the substring() method only return the date for Microsoft Internet Explorer properly.

Table 10–28 Code to Display the Date Last Modified Using the substring() Method

Line	Code
157	`<script type="text/javascript">`
158	`<!--Hide from old browsers`
159	`document.write("<p style='margin-left:10%; font-family:Arial, sans-serif; font-size:xx-small; color:#0000ff'>")`
160	`document.write("Statewide Realty, Copyright 2009-2010. ")`
161	`document.write(" This document was last modified "+document.lastModified.substring(0,10)+"</p>")`
162	`//-->`
163	`</script>`

Lines 157 and 158 are the beginning <script> section tags. Line 159 is the document.write() method with an embedded style in a <p> tag. Line 160 writes the copyright statement and line 161 writes the date last modified. The substring(0,10) method in line 161 extracts the first 10 characters of the date and time — mm/dd/yyyy — so only the date is displayed, not the time. Lines 162 and 163 complete the JavaScript section.

To Display the Date Last Modified Using the substring() Method

The following step illustrates how to enter the JavaScript code to display the date the file was last modified using the substring() method.

1

- Click line 157 (the blank line between the and </body> tags).
- Enter the JavaScript code from Table 10–28 to enter the copyright and date last modified code, pressing the ENTER key only at the end of each complete line. Do not press ENTER after the last </script> line (Figure 10–36).

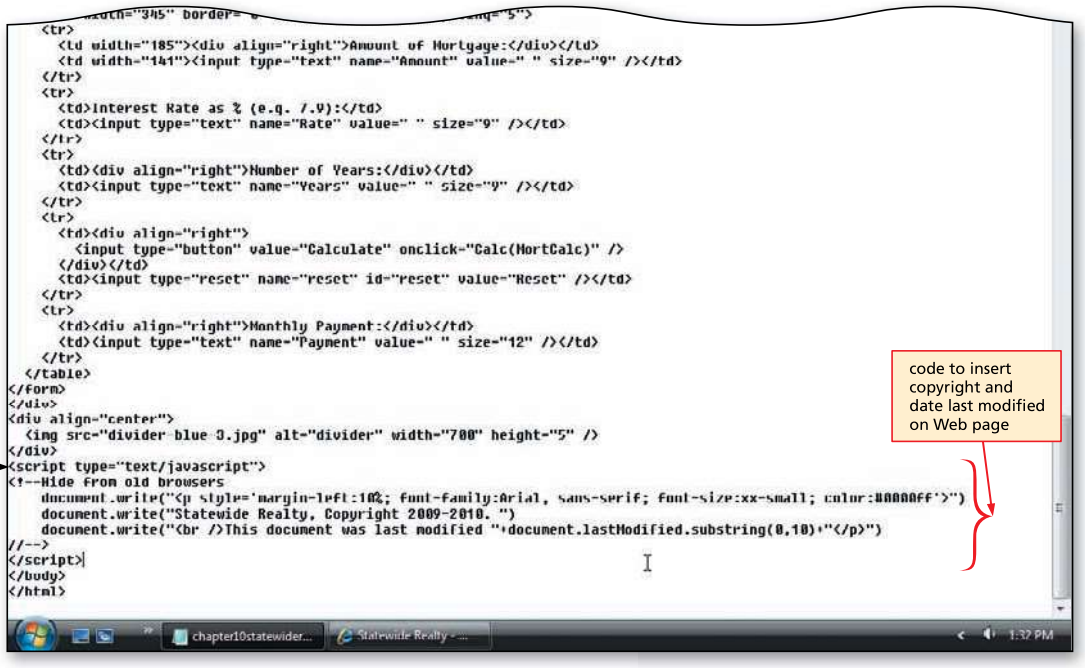


Figure 10–36

To Save and Validate an HTML File, Test a Web Page, and Print the HTML File

The code for the Statewide Realty Web page with a mortgage loan payment calculator and pop-up window is complete. Now you should save the HTML file, test the JavaScript code using a Web browser, and then print the HTML file.

1

- With the USB drive plugged into your computer, click File on the menu bar and then click Save.
- Validate the Web page.
- Click the browser button on the taskbar.
- Click the Refresh button on the browser toolbar.
- Click the Close Window button to close the pop-up window.
- If necessary, scroll down to verify that the bottom of the Web page displays the date the page was last modified (the date the file was saved) as shown in Figure 10–37.

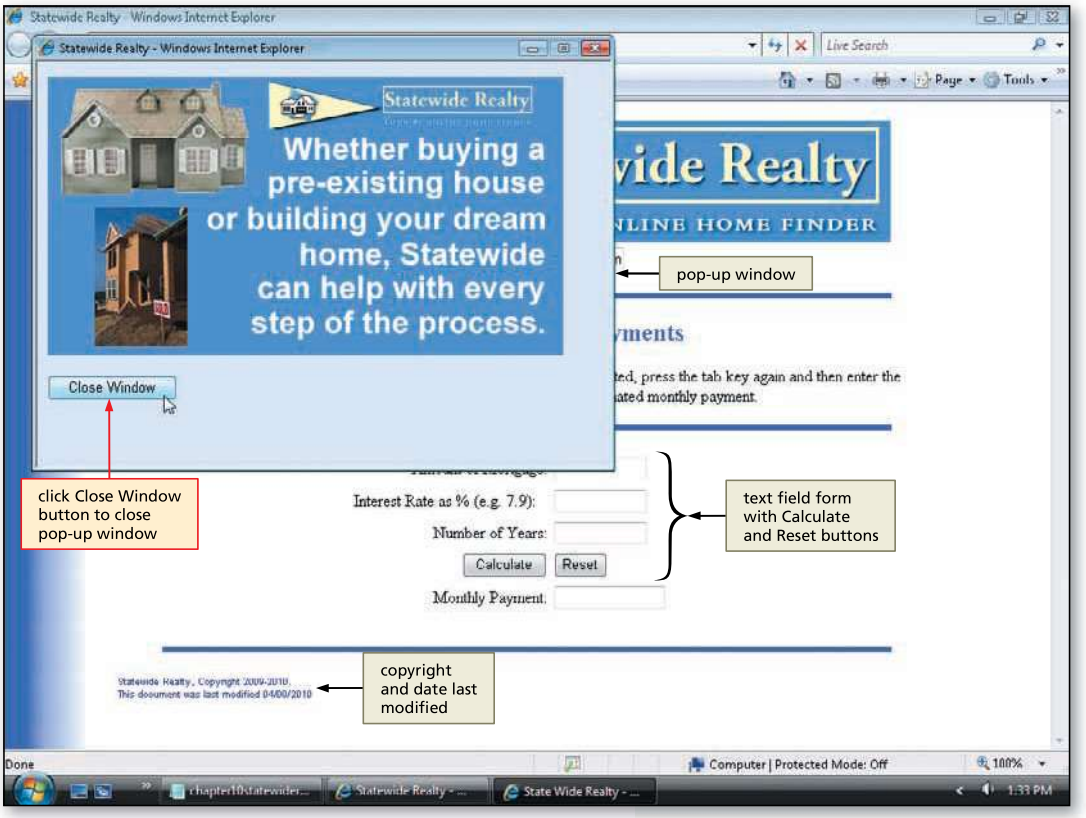


Figure 10–37

- If necessary, click the chapter10statewiderealty - Notepad button on the taskbar to activate the Notepad window.
- Click Print on the File menu to print the HTML file.

To Close Notepad and a Browser

1

- Click the Close button on the browser title bar.
- Click the Close button on the Notepad window title bar.

BTW

Quick Reference
For a list of JavaScript statements and their associated attributes, see the JavaScript Quick Reference (Appendix E) at the back of this book, or visit the HTML Quick Reference Web page (scsite.com/HTML5e/qr).

Chapter Summary

This chapter described how to write JavaScript to create a scrolling message, a pop-up window, if and if…else statements, pass values to a user-defined function, how to validate the data entered into a form and convert text to numeric values using the parseInt(), parseFloat(), and isNaN() built-in functions, and how to format string output results to display as currency. The items listed below include all the new JavaScript skills you have learned in this chapter.

1. Create a Form Text Field to Display a Scrolling Message (HTML 438)
2. Create the scrollingMsg() User-Defined Function (HTML 440)
3. Enter the Code to Increment the Position Locator Variable (HTML 441)
4. Enter an if Statement (HTML 444)
5. Add the setTimeout() Method to Create a Recursive Call (HTML 445)
6. Enter the onload Event Handler to Call the scrollingMsg() Function (HTML 447)
7. Start the Calc() Function and Nested if…else Statements to Validate Form Data (HTML 454)
8. End the Nested if...else Statements to Validate Form Data (HTML 456)
9. Enter an onclick() Event Handler to Call the Calc() Function (HTML 457)
10. Enter Code to Call the monthly() Function (HTML 460)
11. Create the monthly() Function (HTML 462)
12. Enter the dollarFormat() Function (HTML 465)
13. Enter an if…else Statement and while Loop to Extract the Dollar Portion of the Output and Insert Commas (HTML 468)
14. Reconstruct the Formatted Output and Return the Formatted Value (HTML 469)
15. Pass the Monthly Payment Value to the dollarFormat() Function (HTML 470)
16. Enter the popupAd() Function to Open a Pop-Up Window (HTML 473)
17. Add the Event Handler to Call the popupAd() Function (HTML 474)
18. Display the Date Last Modified Using the substring() Method (HTML 475)

Learn It Online

Test your knowledge of chapter content and key terms.

Instructions: To complete the Learn It Online exercises, start your browser, click the address bar, and then enter the Web address `scsite.com/html5e/learn`. When the HTML Learn It Online page is displayed, click the link for the exercise you want to complete and read the instructions.

Chapter Reinforcement TF, MC, and SA

A series of true/false, multiple choice, and short answer questions that test your knowledge of the chapter content.

Flash Cards

An interactive learning environment where you identify chapter key terms associated with displayed definitions.

Practice Test

A series of multiple choice questions that test your knowledge of chapter content and key terms.

Who Wants To Be a Computer Genius?

An interactive game that challenges your knowledge of chapter content in the style of a television quiz show.

Wheel of Terms

An interactive game that challenges your knowledge of chapter key terms in the style of the television show, *Wheel of Fortune*.

Crossword Puzzle Challenge

A crossword puzzle that challenges your knowledge of key terms presented in the chapter.