Estimated Time:
1.5 hours

LESSON 6

Introducing JavaScript

OBJECTIVES

Upon completion of this lesson, you should be able to:

- Display Web page text with JavaScript.
- Use HTML formatting tags both in and around JavaScript code.
- Create conditional statements with JavaScript.
- Use the JavaScript alert() method.
- Access the browser status line with JavaScript.

HTML is undoubtedly the most widely used technology on the World Wide Web. It is the backbone on which all Web pages are based. However, HTML has limitations, so over time, developers have created a wide variety of other technologies to enhance and extend the basic capabilities of standard HTML. One of these widely used companion technologies is JavaScript. JavaScript provides Web developers with tools to add functionality and a little flair to static Web pages.

VOCABULARY

binary code
compiler
condition
conditional statement
interpretation
keywords
methods
objects
operators
parameter list
programming language
properties
<script> </script> tags
scripting language
status line
syntax
token
variable

Hello World Wide Web

VOCABULARY

programming language
scripting language
compiler
interpretation
binary code
syntax
objects
methods

JavaScript is sometimes referred to as a ***programming language***, but it is more accurate to call it a ***scripting language***. The difference between a programming language and a scripting language is subtle but important to understand. Both types of languages must be converted from a human-readable form to a machine-readable form.

For programming languages, the process is performed before the program runs by a specialized piece of software called a ***compiler***. The programmer controls this conversion process.

With a scripting language, however, there is no need for the programmer to explicitly initiate the code-conversion process. It happens automatically when the source code is processed by the target program. To be more specific, an HTML document must be written by a person and then processed by a Web browser. When that document contains embedded JavaScript code, that code is interpreted by the browser and converted into its machine-readable form when the page is loaded. ***Interpretation*** is the term programmers use to describe the line-by-line conversion process that occurs automatically at run time.

In many cases, the output of a JavaScript function will be nothing more than one, or perhaps several, lines of text that are inserted into the host Web page. The resulting HTML page is then processed by the browser just as it would be if it had been keyed into the source document by a person.

The real power of embedding JavaScript code into Web pages comes from the fact that the resulting text can change from one day to the next or even from one minute to the next. It is entirely possible for one person to enter a particular URL into his Web browser and see a Web page that is completely different from the page that is seen by another person who enters the exact same URL. These different-looking Web pages can be the result of differences in time, differences in location, or even differences in Web browsers. JavaScript is capable of detecting various conditions in the current operating environment and reacting accordingly. This concept is explored in greater detail in Step-by-Step 6.3.

It is easy for a Web browser to detect whether a particular Web page contains embedded JavaScript code. All that is required is for the person who creates the document to use the <script> tag to mark the beginning of a JavaScript section and the </script> tag to indicate the end of that section. The Web browser will interpret everything between these two tags as JavaScript source code rather than standard HTML text. The browser will then convert the script (via the interpretation process) into its equivalent machine-readable form called ***binary code***. This binary code will then be executed, and its output (if any) will be inserted into the HTML text stream and displayed as if it had been typed into the original HTML document by a person.

It is important for you to understand that the scripts you embed between the <script> and </script> tags cannot be just any text you care to put in there. On the contrary, the text must conform to certain rules, or the Web browser will display an unpleasant error message on the screen when you try to view your page. This is precisely why JavaScript is called a scripting language—because it must adhere to specific rules of grammar known as program ***syntax***. In this lesson, you learn about JavaScript objects and methods, as well as JavaScript keywords and operators. Once you have mastered these basic language elements, you will be able to start building useful, sophisticated scripts in no time.

The primary purpose of JavaScript is to generate text that will be inserted into the standard HTML text stream. JavaScript is essentially made up of a number of invisible entities called ***objects*** that contain a well-defined set of capabilities. For JavaScript programmers to make use of these capabilities, they must call upon the services of one or more specialized functions known as ***methods*** within those objects. The programmer invokes the services of these methods by entering the name of the object, followed by a period (the . character), followed by the method name.

EXTRA FOR EXPERTS

What is a string? *String* is a term that is well understood by experienced programmers and Web developers but might be new to beginners. In the context of HTML and JavaScript, a string is nothing more than a sequence of one or more characters. A string can consist of a single word, a complete sentence, or an entire chapter of a book. Normally, strings contain meaningful text, but this is not a requirement. A meaningless sequence of gibberish or cryptic symbols can also be a string, and there is virtually no limit to the length of a string.

Method names are always followed by a parameter list, even though the list is sometimes empty. Perhaps the best way to understand method parameters is to visualize a list of ingredients for a recipe. The ***parameter list*** simply provides the method with the information it needs to perform its function correctly. The syntax of the parameter list consists of an opening parenthesis, zero or more parameter items, and a closing parenthesis. For example, if you want to invoke the write method of the JavaScript object called "document," you enter the following line of code:

```
document.write("A string of text.");
```

Now that you have seen a simple example of JavaScript coding, you can incorporate this concept in an actual HTML document with embedded JavaScript.

VOCABULARY

parameter list

WARNING

Your particular Web browser might display various warning messages as you try to work through the following JavaScript activities. If you see such a warning, simply click the option to enable JavaScript. None of the activities in this book are capable of damaging your computer system in any way.

Step-by-Step 6.1

In this step-by-step, you create a very simple HTML file with embedded JavaScript. Pay particular attention to the <script> </script> tags and the JavaScript code that appears between them.

1. Open Notepad, SimpleText, or your favorite text editor.
2. Create a new blank document, if necessary.
3. Enter the HTML and JavaScript text exactly as shown in **Figure 6–1**.

FIGURE 6–1 js-one.html file

```
<html>

<head>
<title>HTML and JavaScript</title>
</head>

<body>
<script>
  document.write("Hello World Wide Web!");
</script>
</body>

</html>
```

4. Save your newly created HTML text file as **js-one.html** or **js-one.htm** in the same folder you saved the files from the previous lessons.
5. Start your favorite Web browser, and then open the **js-one.html** or **js-one.htm** file to view the file as a Web page. The Web page on your screen should look like the one shown in **Figure 6–2**.

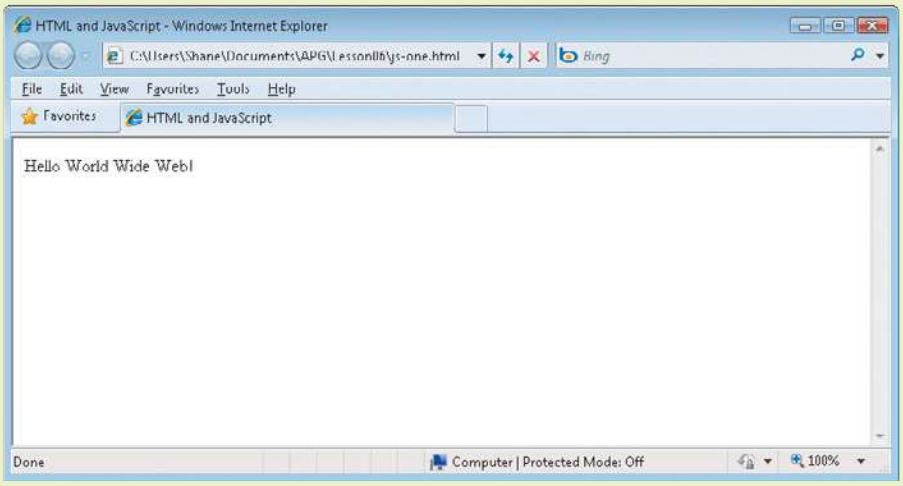

FIGURE 6–2
Your first Web page with embedded JavaScript

WARNING

If your browser displays the message, "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." click the message bar, click Allow Blocked Content, and then click Yes in the Security Warning dialog box.

At this point, you might be thinking that this Web page does not look all that impressive. But don't give up on JavaScript yet because you are just getting started. By the time you have worked through a few more step-by-steps, you will begin to see that JavaScript is capable of much more than this simple Web page demonstrates.

Enhancing Your Web Page

As was mentioned earlier in this lesson, the JavaScript method **document.write()** simply inserts a string of characters into the standard HTML text stream. Another way to think of it is that after the browser has finished processing the HTML document, the end result is that the <script> tag, the </script> tag, and everything in between the two will be stripped out of the page and replaced by the output of the write() method. This is the parameter of the write() method. Any HTML formatting tags you put before or after the <script> </script> tags will be processed just as they would be in a page without any embedded JavaScript code. To illustrate this point, modify the Web page you just created in Step-by-Step 6.1. HTML formatting tags can enhance the "Hello World Wide Web!" message so that it looks a little more appealing on the screen.

Step-by-Step 6.2

In this step-by-step, you add HTML formatting tags and a second embedded script. The formatting tags will make the resulting Web page look a little nicer. A second message on the page helps you to understand the interaction between HTML and JavaScript.

1. Open Notepad, SimpleText, or your favorite text editor if it is not already open.
2. If the **js-one.html** or **js-one.htm** file is not already open in the text editor, click File on the menu bar, and then click **Open** to open the **js-one.html** or **js-one.htm** file you created in the previous step-by-step.
3. Click **File** on the menu bar, click **Save As**, and then save the HTML file as **js-two.html** or **js-two.htm** in the same folder to preserve the js-one.html file.
4. Add the two sections of HTML and JavaScript code after the <body> tag and then after the </script> tag exactly as shown in bold in this step and in bold in **Figure 6–3**.

```
<div align="center">
<h1>


</h1>
<h3>
<script>
  document.write("Welcome to the exciting
  world of JavaScript");
</script>
</h3>
</div>
```

EXTRA FOR EXPERTS

One of the characteristics of the JavaScript language that novice programmers tend to struggle with is the use of "curly" braces. The opening brace ({) indicates the beginning of a statement block, and the closing brace (}) marks the end of that block. Although the rules of JavaScript syntax are somewhat flexible when it comes to the placement of these braces, we strongly recommend that you follow the style shown in this book. If you start getting sloppy or inconsistent with the way you position your braces, it is very easy to cause problems in your scripts. When the number of opening braces does not correspond to the number of closing braces, your Web browser will report a syntax error. However, it is also possible to position braces in such a way that no syntax error will occur, but the script will not execute correctly. This type of problem is frustrating and often very difficult to find and correct. You can save yourself a lot of time and headaches by following a few simple rules:

1. Always place the opening brace directly below the keyword to which it belongs.

2. Always indent the statements contained within the statement block.

3. Always place the closing brace so that it is vertically aligned with its corresponding opening brace.

FIGURE 6–3 js-two.html file

```
<html>

<head>
<title>HTML and JavaScript</title>
</head>

<body>
<div align="center">
<h1>
<script>
  document.write("Hello World Wide Web!");
</script>
</h1>
<h3>
<script>
  document.write("Welcome to the exciting world of JavaScript");
</script>
</h3>
</div>
</body>

</html>
```

5. Save the changes to the file **js-two.html** or **js-two.htm**.
6. Start your favorite Web browser and then open the **js-two.html** or **js-two.htm** file to view the file as a Web page in the browser. The Web page on your screen should look like the one shown in **Figure 6–4**.

FIGURE 6–4
Formatted JavaScript Web page

If you understand the HTML material presented in Unit 1, you should be able to clearly see the purpose of the and header tags that were added to this document. You should also understand that the output string generated by the first occurrence of the document.write() method is inserted into the HTML text stream between the   tags as well as between the <h1> </h1> tags. Likewise, the output string produced by the second occurrence of document.write() appears between the   tags and the <h3> </h3> tags.

In other words, the <h1> </h1> tags only affect the appearance of the first output string, and the <h3> </h3> tags only affect the appearance of the second output string. The   tag pair will affect the display position of both output strings.

Creating Conditional Statements in JavaScript

The astute student might look at the previous two step-by-steps and ask "Why use JavaScript to display text messages on the screen? Wouldn't it be much easier to enter the text into the HTML document and not bother with the script tags and the use of the document.write() method?"

Well, yes, it would be easier to enter the code that can display simple static text messages in a browser using HTML. In fact, the Web browser would display the resulting Web page slightly faster because it would not need to call on the services of the JavaScript interpreter. However, JavaScript is capable of performing a lot more functions than simply writing text to the screen. So, you learn with a simple example, and then expand your knowledge and abilities by creating more exciting Web pages that have to use JavaScript.

In the next step-by-step, you use one of the most powerful features of the JavaScript language, the ***conditional statement***. Every programming language possesses the ability to make decisions. Or, to put it in more technical terms, every language gives programmers the ability to evaluate a specific condition and then perform different actions depending on the results of that evaluation.

The syntax of the conditional statement in JavaScript is very important. The statement begins with the keyword ***if***, and then a condition is specified within a pair of parentheses. A ***keyword*** is recognized as part of the language definition. It is reserved by the language and cannot be used as a ***variable***—a user-defined name for a memory location whose value can change over time. Some examples of keywords are **if**, **else**, and **return**.

The condition is followed by a statement block that consists of an opening brace ({), one or more JavaScript statements, and then a closing brace (}). The shell of a JavaScript conditional statement is as follows:

```
if (<condition>)
{
   statement 1;
   statement 2;
   statement 3;
      .
      .
      .
   statement N;
}
```

WARNING

You might be tempted to place HTML formatting tags (such as , <h1>, <h3>, and so on) inside of the <script> and </script> tags. Don't do it! You need to remember that the Web browser will interpret everything you enter between the script tags as JavaScript source code. HTML tags do not conform to the rules of JavaScript syntax; therefore, the JavaScript interpreter cannot process them. If you make the mistake of entering HTML tags within a JavaScript code block, your browser will respond with an unpleasant error message.

VOCABULARY

conditional statement

keyword

variable

The JavaScript **if** statement also supports an optional **else** clause, which defines the action to take if the specified condition is not true. The **else** keyword appears immediately after the statement block of the if clause and is accompanied by a statement block of its own. An example of a JavaScript conditional statement that includes the optional **else** clause follows:

```
if (<condition>)
{
   statement i1;
   statement i2;
   statement i3;
      .
      .
      .
   statement iN;
}
else
{
   statement e1;
   statement e2;
   statement e3;
      .
      .
      .
   statement eN;
}
```

Now that you know the basic structure of JavaScript **if** and **if-else** statements, you can learn a little about the condition part of the syntax (shown as <condition> in the preceding JavaScript examples). A JavaScript condition will always consist of two ***tokens*** separated by a relational ***operator***. A token can either be a variable name (such as x or count) or a literal constant (such as *10* or "*hello*"). The relational operator may be any one of the symbols shown in **Table 6–1**.

VOCABULARY
tokens
operator

TABLE 6–1 Relational Operators

OPERATOR	MEANING
==	Is equal to
!=	Is not equal to
<	Is less than
>	Is greater than
<=	Is less than or equal to
>=	Is greater than or equal to

Now that you have learned how to create conditional statements in JavaScript, you can put that knowledge to work in an actual program.

Step-by-Step 6.3

In this step-by-step, you include a simple conditional statement in your Web document and learn a useful programming technique in the process. It is fairly common for Web content developers to want the Web page to perform a different task depending on the type of browser a particular user is using to view the Web page. You enter a JavaScript program that determines whether the active browser is Internet Explorer, and based on that determination, the page reacts accordingly.

1. Open Notepad, SimpleText, or your favorite text editor if it is not already open.
2. If it is not already open in the text editor, click **File** on the menu bar, and then click **Open** to open the file **js-two.html** or **js-two.htm** you created in the previous step-by-step.
3. Click **File** on the menu bar, click **Save As**, and then save the file as **js-three.html** or **js-three.htm** in the same folder you saved the files from the previous step-by-steps to preserve the js-two.html file.
4. Enter the JavaScript code exactly as shown here and in bold in **Figure 6–5**.

```
<script>
  if (navigator.appName == "Microsoft
  Internet Explorer")
{
  document.write("You are using Internet
  Explorer.");
}
else
{
  document.write("You are not using Internet
  Explorer.<br>");
  document.write("Perhaps you are using
  Firefox or Safari.");
}
</script>
```

FIGURE 6–5 js-three.html file

```
<html>

<head>
<title>HTML and JavaScript</title>
</head>

<body>
<div align="center">
<h1>
<script>
  document.write("Hello World Wide Web!");
</script>
</h1>
<h3>
<script>
  document.write("Welcome to the exciting world of JavaScript");
</script>
</h3>
<script>
  if (navigator.appName == "Microsoft Internet Explorer")
  {
    document.write("You are using Internet Explorer.");
  }
  else
  {
    document.write("You are not using Internet Explorer.<br>");
    document.write("Perhaps you are using Firefox or Safari.");
  }
</script>
</div>
</body>

</html>
```

5. Save your changes to the **js-three.html** or **js-three.htm** file.
6. Start your favorite Web browser and then open your **js-three.html** or **js-three.htm** file to view the file as a Web page in the browser. The Web page on your screen should look like the one shown in **Figure 6–6** or **Figure 6–7** depending on the browser you are using.

FIGURE 6–6
Web page viewed with Internet Explorer 8

FIGURE 6–7
Web page viewed with Firefox 3.5

Even though you didn't add a lot of code to the JavaScript program, you did apply several important concepts. Take a minute to review those concepts to make sure you have a solid understanding of what is happening.

First of all, the condition being evaluated in this JavaScript code fragment is:

```
(navigator.appName == "Microsoft Internet Explorer")
```

Earlier in this lesson, you learned that JavaScript objects contain special functions called methods that perform various tasks. It is also true that JavaScript objects contain ***properties*** that programmers can access to obtain information about the object. In this case, you are utilizing the **appName** property of the **navigator** object to determine the application name of the current Web browser. In this context, the

VOCABULARY
properties

term *navigator* can be used interchangeably with the term *browser*. If this name is Microsoft Internet Explorer, the program knows that the user is running some version of Internet Explorer. Otherwise, it knows that the user is not running Internet Explorer but is using a different kind of browser.

The second important concept to learn here is that once the condition has been evaluated, either the **if** statement block or the **else** statement block will be executed—never both. If the result of the condition is true, the **if** block will run. If the condition is false, the **else** statement block will run. It's as simple as that.

There is one final point in this example you need to recognize. So far, you learned the concept of embedding JavaScript code into HTML documents. But it turns out that it is also possible, as well as useful, to embed HTML tags in JavaScript text strings. Look carefully at the first call to document.write() inside the **else** statement block. Notice that the **
** tag is embedded within the output text string. The purpose of this tag is to tell the browser to execute a break command so that the second string of text will appear on a separate line, rather than on the same line as the first text string. When your Web page displayed in the browser, the message appeared on two lines.

TECHNOLOGY TIMELINE

What's with Those Semicolons?

If you were to study the source code for many JavaScript-enabled Web pages, you would discover that most of them contain semicolons (;) at the end of each statement, but not all of them. Strictly speaking, the rules of JavaScript syntax do not require a semicolon at the end of each line. However, all of the examples in this book, and in most other JavaScript references, include the terminating semicolons. This is because many other programming languages (such as C, Pascal, Java, and so on) require the presence of semicolons and will report syntax errors if they are not found. Using semicolons in your JavaScript code will help you get used to this coding standard and will ease the transition to other languages should you ever need to do so. After all, it is easier to learn a good habit than to unlearn a bad one, right?

Using the JavaScript alert() Method

In the first three step-by-steps in this lesson, you used the document.write() method, a common way for JavaScript programs to convey a message to a user viewing a Web page. However, scripts can get a Web page visitor's attention in other ways as well. One such way is by means of the JavaScript **alert()** method.

The purpose of the alert() method is to allow a JavaScript program to display a special dialog box that will notify the user that an unexpected event has occurred or that some kind of user input is required. Just as the write() method is part of the document object and the appName property is part of the navigator object, the alert() method is also part of an object called **window**. The window object is just like every other JavaScript object in most respects, but it does have one unique characteristic. Specifically, window is considered to be the JavaScript default object, which means it is not necessary to use its name explicitly. In other words, any method or property that appears in a script without an explicit reference to an object is automatically assumed to be part of the window object. Therefore, JavaScript programmers can alert users by invoking either window.alert(), or, simply, alert(). Most developers use the latter, shorter form.

Step-by-Step 6.4

In this step-by-step, you add code to your JavaScript file, which alerts the user as to the Web browser that he or she is using to view the Web page. This alert dialog box appears on the screen in addition to the Web browser text message you created in Step-by-Step 6.3.

1. Open Notepad, SimpleText, or your favorite text editor if it is not already open.
2. If the **js-three.html** or **js-three.htm** file is not already open in the text editor, click **File** on the menu bar and then click **Open** to open the **js-three.html** or **js-three.htm** file you created in the previous step-by-step.
3. Click **File** on the menu bar, click **Save As**, and then save the file as **js-four.html** or **js-four.htm** to preserve the js-three.html file.
4. Enter the JavaScript code to create the alerts just below each document.write() method, as shown in this step and exactly as shown in bold in **Figure 6–8**.

```
alert("Internet Explorer detected.");

alert("Internet Explorer not found.");
```

FIGURE 6–8 js-four.html file

```
<html>

<head>
<title>HTML and JavaScript</title>
</head>

<body>
<div align="center">
<h1>
<script>
  document.write("Hello World Wide Web!");
</script>
</h1>
<h3>
<script>
  document.write("Welcome to the exciting world of JavaScript");
</script>
</h3>
<script>
  if (navigator.appName == "Microsoft Internet Explorer")
```

Continued on next page ⟩⟩

FIGURE 6–8 js-four.html file

Continued from previous page

```
  {
    document.write("You are using Internet Explorer.");
    alert("Internet Explorer detected.");
  }
  else
  {
    document.write("You are not using Internet Explorer.<br>");
    document.write("Perhaps you are using Firefox or Safari.");
    alert("Internet Explorer not found.");
  }
</script>
</div>
</body>

</html>
```

5. Save your changes to the **js-four.html** or **js-four.htm** file.
6. Start your favorite Web browser and then open the **js-four.html** or **js-four.htm** file to view the file as a Web page in the browser. The Web page on your screen should look like the one shown in **Figure 6–9** or **Figure 6–10** depending on the browser you are using to view the Web page.

FIGURE 6–9
Web page viewed with Internet Explorer 8

FIGURE 6–10
Web page viewed with Firefox 3.5

7. Click **OK** in the warning box to close it.

Hopefully, this step-by-step has helped you see how the alert() method can be useful in various Web page development situations. It is normally used in JavaScript programs when the user needs to be made aware that some unexpected error condition has occurred. It can also be used when the program needs some kind of user acknowledgement before proceeding. For example, you might use an alert box that says "The following process might take several minutes to complete. Click OK to proceed." In either case, the alert() method is an effective way to convey a message to the user.

TECHNOLOGY TIMELINE

Which Came First—Java or JavaScript?

Many people who only have a casual knowledge of Internet technologies tend to think that the Java programming language and the JavaScript scripting language are the same thing. Even those who know they are not the same thing might not know how they relate to each other or which one was developed first. The history of these languages is interesting. Java was created first by Sun Microsystems, Inc. Sun released its cross-platform programming language to the general public in 1995, and it has grown in popularity at an unprecedented rate ever since.

But Sun was not the only company looking for ways to enhance the capabilities of standard HTML. Netscape Communications Corporation was also busy working on technologies to give Web developers a way to embed user-programmable scripts into static HTML documents. They knew that they needed to incorporate a well-defined syntax into their design. Netscape employees observed how popular the Java language was becoming, so they licensed the Java name from Sun and used the Java syntax in their own scripting language. The result of Netscape's efforts became known as JavaScript, and it has also enjoyed a great deal of success in the Internet software development sector.

Accessing the Browser Status Line

You probably have noticed that when your Web browser loads an HTML document that contains many objects such as text and graphics, it displays various messages in a bar at the bottom of the browser window. This area is called the ***status line***, and it can be accessed from within a JavaScript program. In addition to the document.write() method and the alert() method, the status line is another way in which a JavaScript Web page can communicate information to the user.

Accessing the status line of the browser is a very simple task. The message (if any) that the browser displays in the status line is stored as a simple text string in a window object property called **status**. When a JavaScript program assigns a new string value to the status property, the browser responds by displaying this value as a message on the status line.

VOCABULARY

status line

WARNING

Some versions of Firefox have JavaScript access to the status line disabled by default. To allow status line access, open the Options dialog box by clicking **Tools** on the menu bar and then clicking **Options**. In the Options dialog box, click the **Content** tab and make sure the "Enable JavaScript" option is checked. Next, open the Advanced JavaScript Settings dialog by clicking the **Advanced** button. Make sure the "Change status bar text" option is checked and then save your JavaScript settings.

Step-by-Step 6.5

In this step-by-step, you add two new JavaScript statements that assign text strings to the window object's status property. **Window** is the default JavaScript object. These statements simply reinforce the messages that are already being displayed by the alert() method.

1. Open Notepad, SimpleText, or your favorite text editor if it is not already open.
2. If the **js-four.html** or **js-four.htm** file is not already open in the text editor, click **File** on the menu bar, and then click **Open** to open the **js-four.html** or **js-four.htm** file you created in the previous step-by-step.
3. Click **File** on the menu bar, click **Save As**, and then save the file as **js-five.html** or **js-five.htm** to preserve the js-four.html file.
4. Enter the status JavaScript code just below the document.write() method code and above the alert() method code exactly as shown in this step and in bold in **Figure 6–11**.

```
window.status = "Internet Explorer detected.";

window.status = "Internet Explorer not found.";
```

FIGURE 6–11 js-five.html file

```
<html>

<head>
<title>HTML and JavaScript</title>
</head>

<body>
<div align="center">
<h1>
<script>
  document.write("Hello World Wide Web!");
</script>
</h1>
<h3>
<script>
  document.write("Welcome to the exciting world of JavaScript");
</script>
</h3>
<script>
  if (navigator.appName == "Microsoft Internet Explorer")
  {
    document.write("You are using Internet Explorer.");
    window.status = "Internet Explorer detected.";
    alert("Internet Explorer detected.");
  }
  else
  {
    document.write("You are not using Internet Explorer.<br>");
    document.write("Perhaps you are using Firefox or Safari.");
    window.status = "Internet Explorer not found.";
    alert("Internet Explorer not found.");
  }
</script>
</div>
</body>

</html>
```

5. Save your changes to the **js-five.html** or **js-five.htm** file.
6. Start your favorite Web browser and then open the **js-five.html** or **js-five.htm** file to view the file as a Web page in the browser. The Web page on your screen should look like the one shown in **Figure 6–12** or **Figure 6–13** depending on the browser you are using.

FIGURE 6–12
Web page viewed with Internet Explorer 8

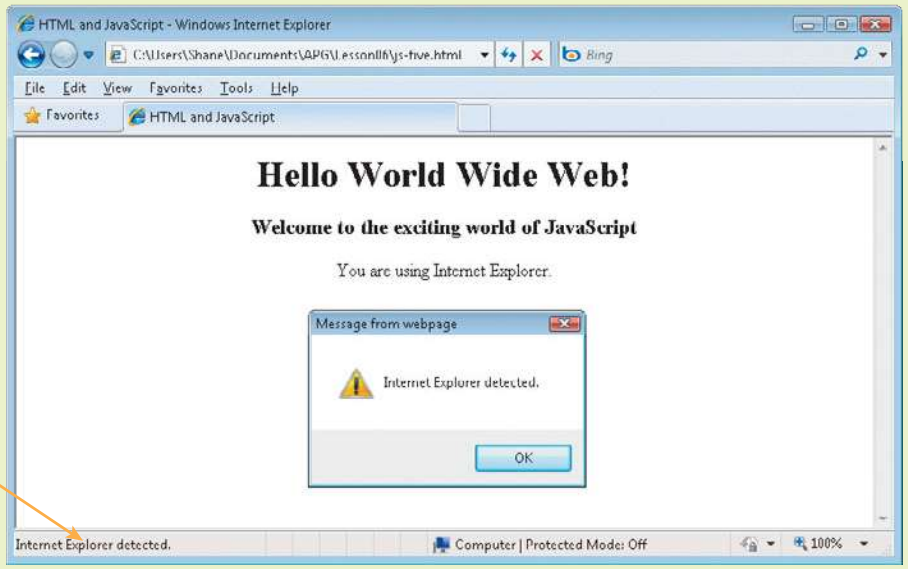

FIGURE 6–13
Web page viewed with Firefox 3.5

7. Click **OK** in the warning box to close it.

In this step-by-step, you used the browser's status line to merely echo the same message that is displayed in an alert dialog box. This was done for the sake of simplicity, but it is not typical for professional Web developers to do this. Normally, the status line is used to let the user know what the browser is doing. This is especially helpful whenever the browser is about to perform an operation that could take an unusually long time to complete. For example, it is common to see a "Loading…" message or something similar whenever the browser initiates the download of a large graphic image, large program, or large data file. Displaying this type of message helps the user know that the browser is really doing something and that the system hasn't crashed.

TECHNOLOGY TIMELINE

Hello World!

The next time you are in a local bookstore, take a few minutes to skim through the first chapter of several different programming books. In many cases, you will see that the first programming example in these books will demonstrate how to display the phrase "Hello, World!" in that particular computer language. Why do you suppose it is so common for authors to begin their books this way? Well, the answer is simple.

In 1978, two employees of Bell Laboratories, Brian Kernighan and Dennis Ritchie, published a book titled *The C Programming Language*. This book has proven itself to be one of the most enduring programming tutorials in the history of computer software. Over the past 30 years, this book has undergone only one revision, and the second edition is still in print today! The very first programming exercise in this book explains how to use the C function **printf()** to display the phrase "Hello, World!" Though it is unlikely that Kernighan and Ritchie intended to start an informal tradition among authors of programming books, that is exactly what they did! We took the liberty of modifying the phrase slightly to "Hello World Wide Web!", but the first JavaScript program in this book is still based on their simple example from over 30 years ago.

SUMMARY

In this lesson, you learned:

- The purpose of JavaScript.
- How to display Web page text with JavaScript.
- How to use JavaScript objects.
- How to use JavaScript methods.
- How to use HTML formatting tags both in and around JavaScript code.
- How to create conditional statements with JavaScript.
- How to use the JavaScript alert() method.
- How to access the browser status line with JavaScript.

VOCABULARY REVIEW

Define the following terms:

binary code
compiler
condition
conditional statement
interpretation
keywords
methods
objects
operators
parameter list
programming language
properties
<script> </script> tags
scripting language
status line
syntax
token
variable

REVIEW QUESTIONS

TRUE / FALSE

Circle T if the statement is true or F if the statement is false.

T F 1. Strictly speaking, JavaScript is a programming language, not a scripting language.

T F 2. Compilers convert human-readable source code into machine-readable binary code.

T F 3. JavaScript requires a compiler.

T F 4. JavaScript code is embedded in HTML documents between <script> and </script> tags.

T F 5. Web browsers can process HTML tags that are placed between the <script> and </script> tags.

T F 6. Embedding HTML tags in JavaScript output strings will cause a syntax error.

T F 7. JavaScript objects include both properties and methods.

T F 8. JavaScript method names are always followed by a list of zero or more parameters.

T F 9. The default JavaScript object is called document.

T F 10. JavaScript requires a semicolon at the end of each statement.

FILL IN THE BLANK

Complete the following sentences by writing the correct word or words in the blanks provided.

1. The symbols { and } are called _____.
2. The JavaScript _____ object contains a method called write().
3. The JavaScript if statement supports an optional _____ clause.
4. JavaScript conditions are composed of two _____ separated by a relational operator.
5. The _____ relational operator means "is not equal to."
6. The JavaScript scripting language incorporates the syntax of the _____ programming language.
7. The appName property belongs to the JavaScript _____ object.
8. The status property and the alert() method belong to the JavaScript _____ object.
9. The JavaScript window object is considered to be the _____ object.
10. appName and status are both examples of JavaScript object _____.

WRITTEN QUESTIONS

Write a short answer to each of the following questions:

1. How does JavaScript enhance the capabilities of HTML?

2. What is a JavaScript object? What are methods and properties?

3. Describe the syntax of a JavaScript conditional statement.

4. What is the unique characteristic of the JavaScript window object?

5. Briefly describe three ways in which a JavaScript program can convey information to the user.

PROJECTS

PROJECT 6–1: BASIC JAVASCRIPT KNOWLEDGE

Review the JavaScript-enabled Web page you created for Step-by-Step 6.1. GreatApplications, Inc., would like you to modify this page so the phrase "Hello World Wide Web!" is replaced with "Welcome to GreatApplications!" Save your new Web page as **Project 6-1.html** or **Project 6-1.htm**, and make sure it functions correctly by viewing it in a Web browser.

PROJECT 6–3: BROWSER TYPES

Review the JavaScript-enabled Web page you created for Step-by-Step 6-5. This page is "Internet Explorer oriented," meaning that it is looking to see whether Internet Explorer is the current browser. Internet Explorer browsers are processed by the **if** statement block, whereas any other type of browser is processed within the **else** statement block. Change this Web page so that it is no longer Internet Explorer oriented, but rather oriented toward a different browser, such as Firefox, Safari, or Chrome. In your new Web page, your browser of choice will be processed within the **if** clause, whereas Internet Explorer and all other browsers are processed by the **else** clause. Make sure that you change the output text appropriately in addition to the **if** statement condition. Save your results as **Project 6-3.html** or **Project 6-3.htm**, and view the file in at least two different browsers to ensure that it functions correctly.

PROJECT 6–2: HTML FORMATTING TAGS

Review the JavaScript-enabled Web page you created for Step-by-Step 6-3. This document contains HTML formatting tags that are located outside the <script> and </script> tags, as well as embedded within the output text strings (the
 tag). Demonstrate your ability to work with HTML formatting tags in both ways by enhancing the appearance of this Web page. Give the page an appropriate background color, and also make the output text appear in a new color (other than black) by means of external HTML tags. Then cause the output text to appear in a different font by embedding <font> tags within the JavaScript code. Save your resulting Web page as **Project 6-2.html** or **Project 6-2.htm**, and view it in your favorite browser.