Lesson 5: Introduction to JavaScript

Objectives

By the end of this lesson, you will be able to:

● 5.1: Define scripting languages and explain their functionality in Web development.
● 5.2: Define JavaScript, describe its main features and uses, and list characteristics of JavaScript that affect Web developers and design.
● 5.3: Compare JavaScript to other languages used in Web development.
● 5.4: Embed JavaScript code into HTML5 documents.
● 5.5: Define the roles of JavaScript objects, properties and methods in scripting and programming.
● 5.6: Define the role of JavaScript variables in scripting and programming.
● 5.7: Define the role of JavaScript expressions and operators in scripting and programming.
Pre-Assessment Questions

1. Which of the following statements about JavaScript is true?
   a. Mobile browsers cannot use JavaScript.
   b. JavaScript runs the same on all browsers.
   c. JavaScript is a compiled language.
   d. JavaScript is platform-independent.

2. In programming, developers use what two other terms to refer to the attributes and behaviors assigned to objects?
   a. Characteristics and actions
   b. Properties and methods
   c. Objects and variables
   d. Events and actions

3. For what purpose do developers use scripting languages such as JavaScript in their HTML pages?
HTML5 and JavaScript

You may recall the earlier discussion about the "Web development trifecta," which consists of HTML5, Cascading Style Sheets (CSS) and JavaScript. By now you understand that HTML5 provides instructions that render Web page elements, and CSS provides instructions for positioning and formatting those elements to create a page's look and feel. So what role does JavaScript play? See Figure 5-1.

Web development of the future is expected to depend heavily on HTML5 application program interfaces (APIs). As mentioned earlier in the course, an API is a source code specification that enables components of an application or program to work together to produce the desired functionality. APIs provide an open environment for Web application development that does not rely on proprietary browser plug-ins.

HTML5 APIs are written using HTML5, CSS and JavaScript. Together, these technologies create applications for Web pages that can easily adapt to various desktop computer types, mobile devices, gaming consoles, smart TVs and so forth. Many expect to see mass adoption of HTML5 APIs in the near future as mobile devices continue to multiply.

Some functions of the HTML5 APIs include:

- Media (audio and video).
- Document editing.
- Cross-document messaging.
- MIME type and protocol handler registration.
- Web storage.

For a complete list of HTML5 APIs from one of the W3C members, visit [http://platform.html5.org/](http://platform.html5.org/). You should bookmark this Web page.
Before you can create HTML5 APIs, however, you must understand the basics of JavaScript. JavaScript is a scripting language that provides the interactivity required for APIs. The more you understand about JavaScript, the more control you will have over HTML APIs. You will study APIs in detail in the next lesson. This lesson will teach you the basics of JavaScript.

### Introduction to Scripting

When the World Wide Web first became popular, Hypertext Markup Language (HTML) was the only language an author could use to create Web pages. HTML is not a programming language but a markup language used to format Web pages, and it has many limitations. HTML displays text and graphics on a Web page, but offers limited interactivity within the Web page.

Most computer users, whether they use Windows, Apple, Linux or some combination, are now accustomed to graphical user interfaces. They click buttons to execute command sequences, enter values into text boxes, and choose from menu lists. This increase in user abilities and expectations has resulted in a continual improvement of HTML, as well as the advent of powerful scripting languages such as JavaScript.

A **scripting language** is a programming language designed to enable computer users to write useful programs easily. Scripting languages include JavaScript, Perl, PHP (PHP Hypertext Preprocessor), Tcl (Tool Command Language), ColdFusion, VBScript and REXX.

In today’s computer-driven world, there is less distinction between programming and scripting languages than there used to be. In general, full programming languages such as C, C++ and Java traditionally must be compiled before they can be run, whereas scripting languages are traditionally interpreted without an explicit compilation step. It is this difference in the execution environment that distinguishes a “scripting language” as a programming language run in an interpreted environment.

Of the many scripting languages available for various purposes, perhaps the most commonly used is JavaScript. It is widely used for client-side scripts in Web pages, to add interactivity to pages without demanding resources from a server. It is also commonly used in server-side programming, as well as game development. And as you are learning, it is an important component in developing mobile applications.

### JavaScript Characteristics

**JavaScript** can make a Web page rich and dynamic. For most programming tasks, you need the server to interpret a script and to return the appropriate responses. JavaScript performs its functionality on the client side, which reduces processing time dramatically.

Before you start writing JavaScript code, you should take a close look at the characteristics and features of this language.
JavaScript is a scripting language

JavaScript is a client-side and server-side scripting language that adds dynamic interactivity to Web pages. It is considered a scripting language because it supports scripts, which are programs that interpret and automate task execution. Scripting languages are not compiled; instead they can perform a list of tasks that would otherwise be executed one-at-a-time by a person.

JavaScript is object-based and object-oriented

JavaScript is an **object-based** language because it derives functionality from a collection of built-in objects. With JavaScript, you can also create your own objects.

JavaScript also has object-oriented programming capabilities. **Object-oriented** is a common term in programming languages. An object-oriented program is a collection of individual objects that perform different functions. These objects are usually related in a hierarchical manner, in which new objects and subclasses of objects inherit the properties and methods of the objects above them in the hierarchy. The concept of objects will be discussed further later in this lesson.

![TechNote](image.png)

*JavaScript has long been considered object-based but not object-oriented. Its object-oriented programming applications are somewhat recent developments.*

JavaScript is event-driven

The Web is based upon an event-driven model. For example, whenever you click an item on a Web page, an event occurs. You are in control: You can choose to click or not click, move the mouse or not move the mouse, or change the URL at will. Because of the unpredictability of a user’s actions, programming modules (called methods or functions) can be created that are independent of each other.

Events can trigger functions. Event triggers can be as simple as the user clicking a button, clicking or moving the mouse over a hyperlink, or entering text into a text field. Scripting can be tied to any of these events. You will learn to use JavaScript to instruct the browser what to do or display using the event that you designate as the trigger for the script.

Client-side JavaScript is part of the text within your HTML document. When your browser retrieves a scripted page, it executes the JavaScript programs and performs the appropriate operations in response to user events.

JavaScript is platform-independent

Because JavaScript programs are typically designed to run within HTML documents, they are not tied to any specific hardware platform or operating system. However, JavaScript programs are tied to a specific **user agent**. Generally, these user agents are browsers. Theoretically, you can implement the same JavaScript program on any current user agent, such as Internet Explorer, Firefox, Safari or Chrome (see Figure 5-2).
Keep in mind that each user agent tends to implement JavaScript differently. Different vendors and user agent versions can complicate your JavaScript implementation. Because you usually cannot guarantee that a user will access your JavaScript code using a specific user agent, take care to create code that will run on as many platforms as possible. At a minimum, you should test your scripts in the latest versions of Internet Explorer, Firefox, Chrome and Safari. One of the most common mistakes in JavaScript is that developers test their JavaScript code insufficiently.

JavaScript also provides a way for you to determine the user agent used to access your programs. You can use this function to your advantage by informing your users who may experience your script differently than intended.

**JavaScript enables quick development**

Because JavaScript does not require time-consuming compilation, scripts can be developed quickly. This advantage is enhanced by the fact that most of the interface features, such as forms and other graphical user interface (GUI) elements, are handled by the browser and HTML code. JavaScript developers need not worry about creating or handling these elements of their applications.

**JavaScript is relatively easy to learn**

JavaScript does not have all of the complex rules associated with Java or other full object-oriented languages. But as with any language, it takes some time and effort to learn it effectively. Depending on your previous experience, JavaScript can be relatively easy to learn. JavaScript makes an excellent first programming language to learn because the basic syntax is simple, you can debug it quickly, and literally thousands of examples with comments are available for you to look up and study.

**JavaScript vs. Other Languages**

JavaScript has similarities to some other scripting languages. There are various implementations, or “flavors,” of JavaScript available. It is useful to know about the past and current versions of the JavaScript language, and how each affects browser support.

By contrast, a common misconception about JavaScript is that it is a derivative of Java. It is not. This section clarifies the relationships between JavaScript and some other languages to which it is often compared.
**JavaScript, JScript and ECMA**

The Microsoft implementation of JavaScript is called JScript. Several differences exist between these two flavors. Although some of the differences are minor, others cause compatibility problems.

Netscape, the inventor of JavaScript, announced in 1997 that the European Computer Manufacturer’s Association (ECMA) had approved JavaScript as an international standard.

The ECMA version intends to diminish the differences between the JavaScript and JScript flavors.

ECMA attempted to promote this standard as “ECMAScript,” but the name did not gain widespread popularity. The following description is quoted from the ECMA Web site:

“This ECMA Standard is based on several originating technologies, the most well-known being JavaScript (Netscape) and JScript (Microsoft).”

For more information about ECMA, consult its home page at www.ecma-international.org.

In summary, JavaScript is the implementation of the ECMA standard. JScript is Microsoft’s version of ECMA, designed to run specifically on Windows.

**VBScript**

VBScript is another scripting language that has similarities to JavaScript. Both extend the capabilities of static Web pages.

VBScript was the original Microsoft Web-scripting language, based on the powerful and popular Visual Basic language. However, VBScript is rarely found within Web pages anymore; it is used some on the server side and is most commonly used within Microsoft’s proprietary programs, such as Word and Excel.

**Visual Basic and VBScript**

VBScript is based on the Microsoft Visual Basic language. VBScript has similarities to the full programming language, but is designed as a lightweight language for Microsoft environments.

**JavaScript vs. Java**

Although the names are similar, JavaScript and Java are very different languages. Although both languages use some syntax elements based on the C programming language, there is no relationship between JavaScript and Java.

Java is a full programming language that was developed by Sun Microsystems. It is generally used to create stand-alone applications and a special type of mini-application called a Java applet. Applets are written in Java, compiled, and then referenced via the HTML `<object>` element in a Web page (or the `<applet>` element in versions prior to HTML 4.01). However, Java is rarely used for Web site enhancement because much better options exist, including JavaScript (Figure 5-3).
Embedding JavaScript into HTML5 Documents

JavaScript code can reside:

- Within an HTML document, as a code block or as inline commands in HTML elements.
- In an external file that is attached to an HTML document.

In this course, you will learn all these techniques.

OBJECTIVE
5.4: Embedding JavaScript into HTML5

NOTE:
If you are interested in learning more JavaScript code and techniques such as inline scripting, consider taking the CIW JavaScript Specialist course.

NOTE:
Consider the differences between JavaScript and Java. You may have been exposed to many of the misconceptions caused by the similar names of these two languages.

Figure 5-3: JavaScript is not related to Java

CIW Online Resources – Online Exercise

Visit CIW Online at http://education.Certification-Partners.com/CIW to complete an interactive exercise that will reinforce what you have learned about this topic.

Exercise 5-1: Introduction to JavaScript
Exercise 5-2: Scripting languages

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>
<title>Page Title</title>
<script>
JavaScript code goes here
</script>
</head>
<body>
HTML page text
<script>
JavaScript code goes here too
</script>
HTML page text
Before HTML5, a type and language attribute were required to define JavaScript (
<script type="text/javascript" language="JavaScript">). Because JavaScript is now
the default scripting language in HTML5, these attributes are no longer required.
The type attribute is optional in HTML5 and the language attribute is being
deprecated.

**Inline scripting**
An additional technique for placing script within an HTML document is called inline
scripting. Inline scripting uses event attributes scripted directly into HTML element tags
to call functions that are specified elsewhere in <script> tags. For example:

```html
<a href="#" onMouseOver="helloFunction();">Click Here</a>
```

Sometimes the JavaScript function is coded directly into the HTML tag, although this is
not considered the best design practice.

For simplified teaching purposes, the main techniques we will use in this course are
inline and embedded JavaScript in the HTML files. However, it is important for you to
know how to attach external JavaScript files as well.

**Linking external JavaScript to an HTML file**
You can add JavaScript to an HTML5 document by creating an external text file with the
.js file name extension, placing the JavaScript code in the .js file, then referencing the
external .js file in your HTML document with a <script> tag.

Similar to attaching a style sheet, this technique allows you to add the same JavaScript
functionality to multiple pages in your site (or multiple sites) without having to add the
script to each HTML page. This approach is particularly useful for complex scripts.

For example, to attach a JavaScript file titled `newScript.js` to your HTML5 page, use the
following <script> tag in the <head> section of your HTML:

```html
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>
<title>Page Title</title>

<script src="newScript.js"></script>

</head>
<body>

HTML page text
</body>
</html>
```

Many developers favor the technique of linking script files to HTML in order to completely
separate script from page content. Similarly, linked CSS is sometimes favored over
embedded CSS because it separates formatting from page content. Ideally, you would
keep content, script and formatting all separate from each other. However, depending on
the scope of your site, the purpose of your scripts and your needs for formatting, separate
files are not always the most efficient approach.
Linking and embedding JavaScript in the same HTML document

Note that if you include the `src` attribute in a `<script>` element, then any text you place between `<script src="script.js">` and `</script>` will be ignored.

Therefore, if you want to embed JavaScript code in an HTML5 document to which you are also linking an external JavaScript file, then you must use separate `<script>` statements to call the external file and embed a separate script. For example:

```html
<head>
<meta charset="utf-8"/>
<title>Page Title</title>
<script src="newScript.js"></script>
<script> JavaScript code goes here </script>
</head>
<body>
<script> JavaScript code goes here </script>
</body>
```

This technique can be useful if you have a complex script that you want to attach to multiple HTML files, as well as a simple script you want to add only to one page, or a simple script you want to run in a different part of the page.

JavaScript and Common Programming Concepts

Some key JavaScript concepts may not make sense right away. However, as this course examines various examples and labs, you will acquire a better understanding of these concepts.

Scripting languages sometimes provide less functionality than full programming languages, but are usually easier to learn. Your investment in learning a scripting language is valuable if you later decide to learn a full programming language because you will have gained basic conceptual awareness of common programming practices.

The following sections introduce some common programming concepts in the context of JavaScript. Keep in mind as you study them that these concepts apply to other more complex programming languages as well. However, the specific applications you are learning in this course belong to JavaScript.

Objects, Properties and Methods

Traditional software programmers tend to write applications using C++, Visual Basic or Java code. These three high-level languages provide rich functionality to the program developer using objects. In programming, objects encapsulate pre-designated attributes and behavior. They are often grouped with similar objects into classes.

In programming, a class is a template used to create objects. Any object created from the class is called an instance of that class. The class provides each instance with default values and behaviors that can be changed.

Like real-life objects, JavaScript objects have certain attributes and behaviors (see Figure 5-4). Developers refer to attributes and behaviors with three other terms:
Lesson 5: Introduction to JavaScript

- **Properties** — various attributes of an object, such as height, color, font size, sentence length and so forth.

- **Values** — the specific qualities of properties. For instance, the statement `color="red"` assigns a value to a property.

- **Methods** — the actions that an object can be made to perform, such as a calculation, an onscreen move or the writing of text into a window. Methods often describe the actions that an object performs with its properties.

To further explain objects, properties, values and methods, consider a pen in terms of object-based programming. It is clearly an object. A pen has definite, discernible properties, such as a length, ink color, point style and so forth. Two pens may have similar properties, yet they may have different values for those properties: All pens will have a color (i.e., property), but all pens will not have the same color (i.e., value).

A pen has methods as well. It can write, spin, leak, and have its cap removed or replaced. A developer creating a virtual pen object would simulate natural attributes and behavior so that, for example, if users try to write with pens, they might receive errors directing them to first remove the caps.

A sentence on a page is another example of an object. You can think of a sentence as a string of words. But to a programmer, that sentence might be an object that has minimum properties of characters, color and font. The values would include number of characters (or sentence length), color type and font style. In addition, you may want to invoke a method in which you can convert that sentence to all uppercase or all lowercase letters.

Think of it this way: Objects are reusable units of code that have behaviors and attributes. An object’s behaviors are generally referred to as methods, and its attributes are referred to as properties. An object’s methods determine the actions the object can perform. Its properties can be used to set or determine the object’s state. An object can also contain collections of properties, generally in key/value pairs.

Ultimately, objects can be defined as collections of properties, with collections of methods that operate on those properties. As you work and develop an understanding of JavaScript, you will see many examples of objects along with their properties and methods.
An example of object properties and methods

Think of a car as an object. A car object might have the following properties, or characteristics:

ObjectCar.Make
ObjectCar.Model
ObjectCar.Year
ObjectCar.Color
ObjectCar.Price

These properties can set or check the car object’s state. Some properties might describe the object’s current state: ObjectCar.Location is an example of a car object’s property that could refer to its current state.

The car object’s speedometer, oil pressure gauge and temperature gauge can represent a collection of properties. Remember that key/value pairs are used to define collections. With the car object, the following key/value pairs could be used:

<table>
<thead>
<tr>
<th>Key</th>
<th>Possible Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speedometer</td>
<td>65 mph</td>
</tr>
<tr>
<td>Oil pressure</td>
<td>160 lbs</td>
</tr>
<tr>
<td>Temperature</td>
<td>190 degrees</td>
</tr>
</tbody>
</table>

The car object might also contain the following methods, or actions it can perform:

ObjectCar.Drive()
ObjectCar.Accelerate()
ObjectCar.TurnLeft()
ObjectCar.TurnRight()
ObjectCar.Brake()

Public vs. private methods and properties

These methods indicate the actions that the car object is made to perform. An object reveals itself through its public methods. These public methods form the object’s interface.

Using the car object, two public methods could be drive and accelerate, which are considered public methods because they can be accessed without going into the car object’s inner workings.

An object’s inner workings represent its private methods. Using the car object, examples of private methods might be the interoperation of the pistons, rotor and camshaft. The car’s driver need not be concerned with how these private methods operate, as long as the car drives and accelerates.

Objects used in programming operate similarly. The object’s properties and methods only need to be accessed, used and reused when necessary, without concern for the object’s inner workings.

Think of objects as small software packages that can be used in various situations and in different places in an application. Reusing objects is one of the primary advantages of object-oriented and object-based programming.
Commonly used methods in JavaScript

Thus far you have seen examples of objects, properties and methods that a developer could create to serve his needs. However, JavaScript also provides some pre-defined objects and methods to perform basic tasks.

Some commonly used basic JavaScript methods include:

- `alert()` — creates a pop-up box with the specified message string, which the user can dismiss by clicking a button in the box.

- `prompt()` — creates a pop-up box with the specified message string and requests user input into a text field in the box.

- `confirm()` — creates a pop-up box with the specified message string and requests user confirmation (yes or no) by clicking the OK or Cancel button in the box.

- `document.write()` — writes the specified message string in the page.

Some methods return values and some do not. For example, the `alert()` method and the `document.write()` method do not return values to any other variable or function. However, the `prompt()` method returns a value in the form of a text string, which you can then pass to a variable or to another method, such as `alert()`.

In the following lab, you will create a script using the `alert()` method. This lab demonstrates a script with a single statement. Suppose you work for the Web team of a midsize company. Your supervisor says that she wants the company Web site to provide users with a more interactive experience when they visit the site. You point out that you can use a simple JavaScript `alert()` method to communicate with users by displaying a pop-up message to greet the user when he or she arrives at the site.

### Lab 5-1: Using the JavaScript `alert()` method

In this lab, you will use the JavaScript `alert()` method to display a message to the user.

1. **Windows Explorer:** Copy the `Lesson05` folder from your student lab files to your **Desktop**.

2. **Editor:** Open `Lab_5-1.htm` in your student lab files.

3. **Editor:** Locate the `<script>` block in the `<head>` section of the document. Within the block, add an `alert()` method with the message "Good Morning!" as the text within the alert box, as shown below in bold:

   ```html
   <script>
   alert("Good Morning!");
   </script>
   ```

4. **Editor:** Save `Lab_5-1.htm`.

5. **Browser:** Open `Lab_5-1.htm`. You should see a dialog box that resembles Figure 5-5. If you do not, verify that the source code you entered is correct.
6. **Browser:** After you click **OK**, your screen should resemble Figure 5-6.

![Figure 5-5: Alert message](image1)

![Figure 5-6: File displayed following JavaScript statement](image2)

7. Close your browser and text editor.

---

**Variables**

A **variable** is a named space of memory. It is a container that allows you to store a value, which you can then access repeatedly and use as needed in your script. Like many other programming languages, JavaScript allows you to declare variables and use their stored values.

Once you obtain a value (such as a user name) and store it for reuse, you can use the data in any way that you see fit. To store data for reuse, you need to assign it as the value for a variable.

**Naming variables**

Variables should have meaningful and descriptive names. Consider what each variable represents within the context of your script, then choose an appropriate name. Even though JavaScript can be lenient when it comes to naming variables, you still must follow certain rules. Following are basic rules for naming variables in JavaScript:
• The first character of the variable must be a letter, the underscore (_ ) character or the dollar sign ($ ). No other initial characters are allowed.
• Subsequent characters can be letters, numbers and/or underscore characters.

The following are examples of valid variable names:
• address1
• _contact
• $allowance
• lastName

The following examples would not be allowed as variable names because they use forbidden opening characters:
• 1stName
• ?subname
• @location

Be aware that the dollar sign ($) can also be used in JavaScript as a single-letter name for a function or as shorthand to refer to the jQuery namespace. These uses are beyond the scope of this course, but you should be aware that the $ sign has some specific uses.

A common naming convention in JavaScript is to use two words with no space between them, and capitalize the second word but not the first. For example, to store first names, you might use a variable called firstName. This practice is called ”camel case” (e.g., camelCase) because the capital letter looks like a hump in the word.

Case-sensitivity in JavaScript
It is important to note that JavaScript is case-sensitive, so the variable name ”Result” is not the same as the variable names ”result” or ”RESULT.” It is good coding practice to ensure that your variables are uniform and also make sense not only to the developer, but also to whomever else might follow to update the script. Remember to use consistent capitalization.

CIW Online Resources – Online Exercise
Visit CIW Online at http://education.Certification-Partners.com/CIW to complete an interactive exercise that will reinforce what you have learned about this topic.

Exercise 5-4: JavaScript variable names

Assigning variables
Variables can store data such as descriptions (a sum, name or location), values (x=24) or expressions (x+y=z). Consider the following code:

    var userName = "Cheshire Cat";

In this example, the variable is declared using the var keyword. The variable is a user’s name, and the value here is Cheshire Cat. This variable is stored and can be called upon whenever necessary. For example, you can collect and store a user’s name, then use this value (the name) to personalize further interactions in your page:
var userName = "Cheshire Cat";
alert("Hello, " + userName);

Notice that the value is assigned to the variable using the equal symbol (=), which is called the assignment operator in JavaScript. The plus sign (+) is also an operator. The line of code in which you assign a value to a variable is called an assignment expression. You will learn more about operators and expressions in the next sections.

The name "variable" is appropriate because the values you store will vary from one variable to the next. You should also note that a variable’s value may change over the course of a script sequence.

Variables are one of the most important and most used features in any programming language. It is considered good practice to standardize the names and naming conventions you use for your variables (such as firstName, lastName), then use that standard format consistently. It will save you a lot of time and effort. Remember case-sensitivity!

**Variable data types**

A data type is a definition of the type of information that a variable holds. Variables can store five data types in JavaScript: number, string, Boolean, Object and null. An additional data type — undefined — occurs when a variable has no value assigned. Other languages use many more data types. Table 5-1 lists examples of the values that these variable data types represent.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Any numeric value (e.g., 3, 5.3, or 45e8) Used for mathematical operations</td>
</tr>
<tr>
<td>string</td>
<td>Any string of alphanumeric characters (e.g., &quot;Hello, World!&quot;, &quot;555-1212&quot; or &quot;KA12V2B334&quot;) Used for words or for numbered phrases that are not mathematically manipulated</td>
</tr>
<tr>
<td>Boolean</td>
<td>True or false values only</td>
</tr>
<tr>
<td>Object</td>
<td>A reference to a JavaScript object</td>
</tr>
<tr>
<td>null</td>
<td>A special keyword for the null value (e.g., If a user enters nothing in a text box then submits the form, the text box has null value) <em>Note: Compare null value to empty value, in which a user types and enters only a space in the text box; the value of the text box is empty, but can be counted as a character.</em></td>
</tr>
<tr>
<td>undefined</td>
<td>Occurs when a variable has no value assigned yet <em>Note: The undefined value is different from null, which is actually a value of nothing.</em></td>
</tr>
</tbody>
</table>

**Literals**

Literals are the actual data values you provide in JavaScript. These are fixed values that you assign to variables in your script. Table 5-2 lists three examples of literals used in JavaScript.
When assigning literal values to variable names, you place string values inside quotation marks. Numeric values that you want to manipulate mathematically should not be placed inside quotation marks.

### Table 5-2: JavaScript literal types

<table>
<thead>
<tr>
<th>Literal Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer literal</td>
<td>56</td>
</tr>
<tr>
<td>Floating-point literal</td>
<td>23.45</td>
</tr>
<tr>
<td>String literal</td>
<td>&quot;Howdy!&quot;</td>
</tr>
</tbody>
</table>

### Expressions and Operators

An **expression** is a part of a statement that is evaluated as a value. The exception to this is an assignment expression, which assigns a value to a variable (which you saw in the preceding section). An expression can use any combination of variables, literals, operators and other expressions.

In JavaScript, **operators** are symbols or characters used in expressions to store or evaluate a value, generally manipulating **operands** in the process. An operand represents a piece of data (i.e., a value) that is manipulated in some manner by an operator. Operators come in many varieties based on their purpose (e.g., assignment, arithmetic, logical, etc.).

Expressions rely on various operators. Sometimes using certain operators can produce subtle differences in the results, and some operators perform different operations depending on the values used with them.

Table 5-3 describes the basic types of expressions used in JavaScript and shows examples of the operators used to implement them.

### Table 5-3: JavaScript expression types and operators

<table>
<thead>
<tr>
<th>Expression Type</th>
<th>Purpose</th>
<th>Example</th>
<th>Explanation</th>
<th>Example Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment</td>
<td>Assigns a value to a variable</td>
<td>myNumber = 25;</td>
<td>The value 25 has been assigned to the variable myNumber.</td>
<td>=</td>
</tr>
<tr>
<td>Arithmetic</td>
<td>Evaluates to a number</td>
<td>25 + 75;</td>
<td>This expression evaluates to the sum of 25 and 75.</td>
<td>+</td>
</tr>
<tr>
<td>String</td>
<td>Evaluates to a string</td>
<td>&quot;Hello, &quot; + &quot;Newman&quot;;</td>
<td>This evaluates to a new string that says 'Hello, Newman'.</td>
<td>+</td>
</tr>
</tbody>
</table>
### Table 5-3: JavaScript expression types and operators (cont’d)

<table>
<thead>
<tr>
<th>Expression Type</th>
<th>Purpose</th>
<th>Example</th>
<th>Explanation</th>
<th>Example Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical</td>
<td>Evaluates to true or false</td>
<td>25 &lt; 75;</td>
<td>Because 25 is less than 75, this expression evaluates to the Boolean value of true.</td>
<td>&lt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25 &gt;= 75;</td>
<td>Because 25 is not greater than or equal to 75, this expression evaluates to the Boolean value of false.</td>
<td>&gt;=</td>
</tr>
<tr>
<td>Comparison</td>
<td>Compares two values and returns a true or false value</td>
<td>z == 10</td>
<td>Returns true if z is 10</td>
<td>==</td>
</tr>
<tr>
<td>Conditional</td>
<td>Makes decisions in a script; the expression (condition) will return either true or false, then evaluate the corresponding expression</td>
<td>(a &gt; b) ? a++ : a--;</td>
<td>If a is greater than b, then the value of a will be increased by 1; if a is less than b, then the value of a will be decreased by 1</td>
<td>?</td>
</tr>
</tbody>
</table>

### CIW Online Resources – Online Exercise

Visit CIW Online at [http://education.Certification-Partners.com/CIW](http://education.Certification-Partners.com/CIW) to complete an interactive exercise that will reinforce what you have learned about this topic.

**Exercise 5-6: JavaScript expressions**

**Concatenation**

Concatenation is used frequently in JavaScript to combine text strings, especially in conjunction with the `prompt()` and `alert()` methods. Because the `prompt()` method always returns a string, concatenation allows you to combine strings for further manipulation. For example, you may want to personalize an alert message. To do this, you can concatenate a message such as "Good Morning!" with the result of a prompt screen in which the user enters his or her name. Following is an example of this type of syntax:

```javascript
alert("Begin message " + prompt("Message to user", "default text") + " End message.");
```

In this example, the processing of the `prompt()` method will take priority over the processing of the `alert()` method. A method that is defined inside another method will always execute before the outer method.

In the following lab, you will use the JavaScript `prompt()` method to capture user input. You will prompt the user for his or her name. You will then tie that name, using the `+` operator, to the phrase "Good morning, ". When the user enters a name, a message will be displayed saying "Good morning, name." including the name entered by the user.
Suppose your Web team supervisor liked your idea for using JavaScript to add pop-ups to the site, but says that she wants the site experience to be personalized to each visitor. You point out that you can use a simple JavaScript `prompt()` method to collect information from the user, such as his or her own name, and then display that information in a pop-up message as well. Now each site visitor can have a personalized interaction when they visit your site.

### Lab 5-2: Using the JavaScript `prompt()` method

In this lab, you will use the JavaScript `prompt()` method with concatenation to request and capture user input.

1. **Editor:** Open `Lab_5-2.htm` in your student lab files.

2. **Editor:** Locate the `<script>` block with the `alert()` method that has been defined for you. Modify the source code by adding a `prompt()` method that asks for the user’s name. Concatenate the user’s input with the existing text and add a closing period after the user input. Do this by adding the following code as shown in bold:

   ```html
   <script>
   alert("Good morning, " + prompt("What is your name?", ") + ".");
   </script>
   
3. **Editor:** Save `Lab_5-2.htm`.

4. **Browser:** Open `Lab_5-2.htm`. When the page loads, you should see a prompt dialog box that resembles Figure 5-7. If not, verify that the source code you entered is correct.

   ![Figure 5-7: User prompt dialog box](image)

5. **Browser:** Enter your name in the text field, and then click **OK**. Your screen should display a message that resembles the one shown in Figure 5-8.

   ![Figure 5-8: Alert message box](image)

6. **Browser:** When you click **OK**, your screen should resemble Figure 5-6 (from the previous lab).

7. **Browser:** Reload the page to run the script again. This time, do not enter any text in the prompt, then click **OK**. The alert will display the message "Good morning, ." This is evidence of an empty string, which is a string that contains no characters.
8. **Browser:** Reload the page again. This time, click **Cancel** (with or without first entering any text). The alert will display the message "Good morning, null." When no data is entered by the user, the `prompt()` method returns a null value, which is converted to the string "null" in this return display. Be sure to consider how the user’s actions might affect any JavaScript methods you use that incorporate user input.

9. Close your browser and text editor.

In this lab, the `prompt()` method is processed first and the user’s input is then concatenated into the expression. In other words, the `prompt()` method will take precedence over the `alert()` method in a JavaScript statement. In fact, JavaScript statements always execute from the inside out.

### Writing HTML dynamically: The `document.write()` method

Thus far in this lesson, you have created simple pop-up windows to alert and prompt your users. Another JavaScript mechanism to communicate with the user is to write text into the document as the page loads in the browser. To perform this task, you use the `document.write()` method, which belongs to the `document` object. The `document.write()` method allows you to create text that is dynamically written to the window as the script is executed. You can write out plain text or you can mix HTML tags with the text being written.

The basic syntax of the `document.write()` method is as follows:

```html
<body>
<script>

    document.write("How wonderful to meet you, "+ 

    prompt("Hello, my name is Patrick. What's yours?", 

    "+ 

    ");

</script>
</body>
```

This syntax is different from `alert()` and `prompt()` in that it includes the specification of a new object, called `document`. The `document` object represents the HTML document rendered in the browser window. The first two methods you used — `alert()` and `prompt()` — are methods of the `window` object in JavaScript, which represents a browser window. Those methods could also have been written as follows:

```javascript
    window.alert();
    window.prompt();
    window.confirm();
```

Because the `window` object is the default object, specifying `window` before each method is not necessary. But now you are using the `write()` method of the `document` object, which is not the default object and therefore needs to be referred to explicitly. You make this reference using **dot notation**. Typical syntax in dot notation is as follows:

```javascript
    objectName.methodName();
    objectName.propertyName;
```
You will see dot notation used in the JavaScript throughout this course. For now, you need to know that by scripting `document.write("Hello")` you are writing the text "Hello" directly to the page as it loads.

The following lab demonstrates the use of `document.write()`. It is similar to Lab 5-2, but uses `document.write()` instead of `alert()` to return text to the user. A portion of the text will be collected from the user with the `prompt()` method. Suppose your Web team supervisor liked your idea for using JavaScript to add alerts and prompts to the site. Now she asks if there is a way to incorporate the info entered by users into a Web page instead of a pop-up. You tell her yes, you can use a simple JavaScript `document.write()` method to do that.

### Lab 5-3: Using the JavaScript `document.write()` method

In this lab, you will use the `document.write()` method to customize a page for the user.

1. **Editor:** Open `Lab_5-3.htm` in your student lab files.

2. **Editor:** Locate the `<script>` block with the `prompt()` method that has been defined for you. Notice that this time the `<script>` block is in the `<body>` section of the HTML file, not the `<head>` section. This instructs the browser to execute the script in the content area of the HTML document, associating it with the page rather than the browser window.

3. **Editor:** Modify the source code to use a `document.write()` statement. Concatenate the results of the `prompt()` method into the `document.write()` expression. Designate the output of the `document.write()` as an `<h4>` level greeting that displays the text "Welcome, user's name." The text "user's name" will be the return value from the `prompt()` method. Be sure to end with a period and close the `<h4>` tag after inserting the user's name. Do this by adding the following code as shown in bold:

   ```javascript
   <script>
   document.write("<h4>Welcome, " + prompt("What is your name?", ") + "</h4>";)
   </script>
   ```

4. **Editor:** Save `Lab_5-3.htm`.

5. **Browser:** Open `Lab_5-3.htm`. You should see a dialog box that resembles Figure 5-9.

   ![Figure 5-9: User prompt](image)

6. **Browser:** Type your name in the dialog box, and then click **OK**. Your screen should resemble Figure 5-10.
7. Edit the initial prompt to contain a message within the text entry field. Place the text string you want inside of the now-empty quotation marks. For example, you could insert the following text shown in bold:

```
prompt("What is your name?","Thank you for entering your name here");
```

This addition alters the initial alert box as shown in Figure 5-11.

```
Figure 5-11: Customizing initial prompt
```

8. Close your browser and text editor.

Note that in this lab, you were able to include an HTML heading tag as part of the text that was written to the screen. HTML can be freely interspersed with text when using the `document.write()` method. Note also that the `prompt()` method takes processing precedence over the `document.write()` method when both are used in the same expression.

**NOTE:**
You can study the terms you have learned in this lesson in Activity 5-1: Reviewing basic scripting terms.

Visit CIW Online at [http://education.Certification-Partners.com/CIW](http://education.Certification-Partners.com/CIW) to take the Course Mastery review of this lesson or lesson segment.

**Course Mastery Lesson 5**
Case Study

Client-Side Stride

TimeWarp E-Calendar Company’s Web site had a major problem: January was coming, and its site had barely survived the New Year’s rush the previous year.

The TimeWarp Web site allowed users to design a custom calendar, download it, then print it out. The calendar program was created using the Java programming language. The Java program was a huge drain on the TimeWarp server resources.

However, TimeWarp found a solution that used JavaScript to modify the calendar. The user could pay for the calendar and download the pages to his or her system, and run the JavaScript program locally. When the calendar was finished, the user could print it or save it for further editing later. But after the initial download, TimeWarp E-Calendar’s server would not need to be involved at all.

* * *

Consider this scenario and answer the following questions:

- What are some advantages of shifting some or all of a process load from the server to the client using JavaScript? What are some disadvantages?

- Even if you cannot rewrite an entire program in JavaScript, it is almost always possible to shift at least some code from the server to the client. What are some examples of common Web site functionality that can be performed faster or better on a user’s computer (the client) rather than on the server?
Lesson Summary

Application project

The World Wide Web provides many valuable JavaScript resources. Use any good search engine to locate several JavaScript resources. Also, you can find some resources listed in an appendix to this course.

Begin investigating sites that provide language documentation, tutorials and code examples. Become familiar with the various resources on the World Wide Web. Determine and document which sites:

• Provide the most information
• Are the easiest to use
• Will benefit you the most as you learn the JavaScript language.

As you begin writing code, you will find that these types of resources are essential in several ways:

• They help you learn the language.
• They save you development time when you implement the language.
• They help you stay up-to-date over time as the technologies and languages you learn are evolved and improved.

Skills review

In this lesson, you were introduced to scripting and learned the basics of JavaScript. Like other scripting languages, JavaScript is interpreted and not compiled to any particular machine or operating system. It derives its functionality from a collection of built-in objects. Although the names are similar, Java and JavaScript are different languages. Java is a full object-oriented programming language developed by Sun Microsystems, whereas JavaScript is an object-based scripting language that normally runs in a user agent such as a browser. You learned that you can embed JavaScript within an HTML document or link to an external JavaScript file by using the <script> element. You also learned that programmers refer to attributes and behaviors in JavaScript (and in other programming languages) by three specific terms: properties, values and methods.

Now that you have completed this lesson, you should be able to:

✓ 5.1: Define scripting languages and explain their functionality in Web development.
✓ 5.2: Define JavaScript, describe its main features and uses, and list characteristics of JavaScript that affect Web developers and design.
✓ 5.3: Compare JavaScript to other languages used in Web development.
✓ 5.4: Embed JavaScript code into HTML5 documents.
✓ 5.5: Define the roles of JavaScript objects, properties and methods in scripting and programming.
✓ 5.6: Define the role of JavaScript variables in scripting and programming.
✓ 5.7: Define the role of JavaScript expressions and operators in scripting and programming.
Lesson 5 Review

1. In programming, a characteristic of an object (such as width or color) is called what?

2. What advantage do scripting languages such as JavaScript offer over programming languages such as Java?

3. Name at least three key characteristics of JavaScript.

4. Where does client-side JavaScript code reside, and how is it activated?

5. In programming, what is an object?

6. In programming, developers refer to an object's attributes and behaviors by what other terms?