Naming variables

When you create a variable, give the variable a name that is descriptive enough that you don't need a comment to describe what it is. If you need to add a comment to describe the variable usage, the comment will be at the declaration only. If you name the variable in a way that does not require a comment, the meaningful name will be readable throughout your code. Here are some good and bad examples of variable naming:

```
//bad examples
var last;      //last accessed date
var current;   //current vehicle
var changed;   //the vehicle make was changed

//good examples
var lastAccessedDate;
var currentVehicle;
var vehicleMakeWasChanged;
```

Notice the casing that is used in the good examples. The recommended naming convention for JavaScript variables is to use camel casing, which means you start a variable name in lowercase and then capitalize the first letter of each subsequent word that makes up the variable name.

Although a variable name can contain the dollar sign and the underscore, it's usually preferable not to use them. The exception is when assigning jQuery objects (discussed in Chapter 6, "Essential JavaScript and jQuery") to variables, when you might want to begin the variable name with the dollar sign.

Creating the environment

The collection of all variables and their values is commonly referred to as the *environment*. When the environment is created, it contains many standard variables plus the variables you create.

In a web application, each time a webpage is loaded into the browser, a new environment is created, and the old environment is destroyed. Any variables that you create are accessible until a new webpage is loaded.

In a Windows 8 program, an environment is created when the application starts, and the environment is destroyed when the application ends. A variable is accessible as long as your program is running.

Working with functions

A *function* is a grouping of statements that are executed when you call the function. Functions promote code reuse because you can call the function many times from within your code. Functions can have parameters, which enable you to pass data into the function. Functions can also have a return value, so you can return the results of the function to the caller.

In this example, another statement was added to the statement from the previous example. This statement contains an expression that uses the *totalCost* variable to calculate the tax and store it in another variable called *tax*.

Note that you can declare the variable in one statement and initialize it in a different statement, as follows:

```
var totalCost;
var tax;
totalCost = 3 * 21.15;
tax = totalCost * .05;
```

This example shows you how you could declare all your variables first and then initialize the variables later in the program.

The value you assign to a variable is not permanent; it is called a variable because you can change it. The following examples modify the *totalCost* variable:

```
var totalCost = 3 * 21.15;
totalCost = totalCost * .1;
totalCost *= .1;
```

The first example initializes the *totalCost* variable. The second example reads the value of *totalCost*, multiplies the value by .1, and stores the result back into *totalCost*. This overwrites the old value with the new value. The third example is a shortcut for the action in the second example. It uses the *= syntax to indicate that you want to multiply the existing value by .1 and store the result in the same variable:

Rules for naming variables

Every programming language has rules for naming variables, and JavaScript is no exception. You must adhere to the following rules when naming JavaScript variables.

- A variable name can contain numbers, but they cannot begin with a number. Legal examples are *x1*, *y2*, *gift4you*. Illegal examples are *4YourEyes*, *2give*, *1ForAll*.
- Variable names must not contain mathematical or logical operators. Illegal examples are *monday-friday*, *boxes+bags*, *cost*5*.
- Variable names must not contain any punctuation marks of any kind other than the underscore (_) and dollar sign ($). Legal examples are *vehicle_identification*, *first_name*, *last_name*, *$cost*, *total$*. Illegal examples are *thisDoesn'tWork*, *begin;end*, *Many#s*.
- Variable names must not contain any spaces.
- Variable names must not be JavaScript keywords, but they can contain keywords. Illegal examples are function, char, class, for, var. Legal examples are *theFunction*, *forLoop*, *myVar*.
- Variable names are case-sensitive. Examples of different-case variables are *MyData*, *myData*, *mydata*, *MYDATA*.