Make Statements

JavaScript code is composed of a series of instructions called "statements", which are generally executed in top-to-bottom order as the browser's JavaScript engine proceeds through the script.

Each statement may contain any of the following components:

- **Keywords** – words that have special significance in the JavaScript language.
- **Operators** – special characters that perform an operation on one or more operands.
- **Values** – text strings, numbers, Boolean **true** or **false**, **undefined**, and **null**.
- **Expressions** – units of code that produce a single value.

An "expression" produces a value, whereas a "statement" performs an action.

In earlier JavaScript code each statement had to be terminated by a **;** semicolon character – just as each sentence must end with a **.** period (full stop) character in the English language. This is now optional so may be omitted unless you wish to write multiple statements on a single line. In that case, the statements do need to be separated by a semicolon, like this:

statement **;** ***statement*** **;** ***statement***

Some JavaScript programmers still prefer to end each statement with a semicolon. The examples in this book choose to omit them for the sake of concision but the choice is yours.

The JavaScript interpreter ignores tabs and spaces ("whitespace") so you should use spacing to make your code more readable. For example, when adding two numbers:

total = 100 + 200 rather than **total=100+200**

Use the space bar to indent statements, as tab spacing may be treated differently when viewing the code in text editors.

JavaScript statements are often grouped together within **{ }** curly brackets ("braces") in function blocks that can be repeatedly called to execute when required. It is good practice to indent those statements by two spaces to improve readability, like this:

```
{
  statement
  statement
  statement
}
```

The rules that govern the JavaScript language is called "syntax", and it recognizes two types of values – fixed and variable. Fixed numeric and text string values are called "literals":

- **Number literals** – whole number integers, such as **100**, or floating-point numbers such as **3.142**.
- **String literals** – text within either double quotes, such as **"JavaScript Fun"**, or single quotes such as **'JavaScript Fun'**.

Decide on one form of quotes to use in your code for string literals and stick with it for consistency. The examples in this book use single quotes.

Variable values are called, quite simply, "variables" and are used to store data within a script. They can be created using the JavaScript **let** keyword – for example, **let total** creates a variable named "total". The variable can be assigned a value to store using the JavaScript **=** assignment operator, like this:

```
let total = 300
```

Other JavaScript operators can be used to form expressions that will evaluate to a single value. Typically, an expression may be enclosed within **( )** parentheses like this expression that comprises numbers and the JavaScript **+** addition operator and evaluates to a single value of 100:

```
( 80 + 20 )
```

Expressions may also contain variable values too, like this expression that comprises the previous variable value, the JavaScript **-** subtraction operator, and a number, to also evaluate to a single value of 100:

```
( total - 200 )
```

JavaScript is a case-sensitive language so variables named **total** and **Total** are regarded as two entirely different variables.

It is good practice to add explanatory comments to your JavaScript code to make it more easily understood by others, and by yourself when revisiting the code later. Anything that appears on a single line following **//** double slashes or between **/*** and ***/** character sequences on one or more lines will be ignored.

```
let total = 100  // This code WILL be executed.

/* let total = 100
  This code will NOT be executed. */
```

It is often useful to "comment-out" lines of code to prevent their execution when debugging code.