JavaScript

Dr. Liew Voon Kiong

Made Easy

Liability

The purpose of this book is to provide basic guides for people interested in JavaScript. Although every effort and care has been taken to make the information as accurate as possible, the author shall not be liable for any error, harm or damage arising from using the instructions given in this book.

About the Author

Dr. Liew Voon Kiong holds a bachelor degree in Mathematics, a master degree in Management and a doctoral degree in Business Administration. He has been involved in JavaScript programming for more than 10 years. He has also created the popular online JavaScript Tutorial at javascript-tutor.net . Besides, he is the author of other programming books, among them are **Visual Basic 2010 Made Easy , Visual Basic 6 Made Easy** and **Excel VBA Made Easy.**

Table of Contents

Chapter 1
Introduction to JavaScript

1.1 What is JavaScript?

JavaScript is a scripting language that works with HTML to enhance web pages and make them more interactive. Simply say, JavaScript is a scripting language for HTML. It makes up of a sequence of statements that give instructions for the computer to perform certain tasks, for examples, like providing a response to the user, plays a song, starts a slide show, and displays advertisements and more.

JavaScript can turn a web page into a lively interactive platform for the World Wide Web users! By using JavaScript , you can add sound, date, time, change the color of the web page according to certain day, pre-validate data entered into a form by the users before it is sent to the server, search through a database, set options based on users preferences and much more

JavaScript is a simple programming language with lower learning barrier than the full-featured programming languages like Visual Basic, JAVA, C++, Turbo Pascal and more. Despite its limited capabilities, it is an object oriented programming language as it deals with objects, methods, properties and data.

1.2 Why use JavaScript?

Since the invention of the World Wide Web and the creation of various web browsers, HTML was the core tool in building and designing web pages. HTML or Hyper Text Markup Language was the only language that was used to present text and graphics as well as links to the World Wide Web users in the 90's of the last century. Although it was a vast improvement from the earlier text-only browsers like Gopher, it was relatively passive and static; it cannot interact much with the user. That is why we need JavaScript to make browsing the web a more interesting and useful experience for the Internet users.

With the invention of JavaScript and other web development tools, present day web pages are smartly designed and packed with fancy features like floating menus, interactive advertisements, animation and more. Web developers simply cannot create these features using plain HTML. Among the new tools that help web developers to create those fancy features, JavaScript is one of the most prominent ones.

1.3 The Building Block of JavaScript

As JavaScript is an object oriented programming language; therefore the building block of the JavaScript program code is made up of objects as well as methods, properties and events associated with the objects.

1.3.1 Objects

The purpose of a JavaScript program is to manipulate the elements of a web page, such as documents, forms, radio buttons, check boxes, buttons, windows and more. All the aforementioned elements are objects.

We identify each object in a web page by its name. For example, the default name of a text box is `Textbox1`; if we insert another text box, the default name will be `TextBox2`. We can change the name of an object so that it is meaningful and easier for us to identify it, like `Txt_StudentName` instead of just `TextBox1`.

1.3.2 Properties

Every object of a web page has a number of properties. The properties of an object reflect its characteristics or its behaviors. For example, the properties of a text box are name, height, width, background color and foreground color, with border or no border, font type and size and more. We can specify the property of an object using the following syntax:

```
Object.property
```

For example, `document.bgcolor=”red”` creates a web page with red background.

You can try out the following code using different colors.

```
<Script language=”JavaScript”>
     document.bgcolor=”Red”
</Script>
```

Other properties of the document object include `fgColor`[1]`, linkColor,` `vlinkColor` and more. By the way, `fgColor` means foreground color, `linkColor` means hyperlinks color and `vlinkColor` means visited links color in a web page. We will introduce more properties in later chapters. Please note that some properties might work in some browsers but not others. Most work fine in IE but not Chrome. You can try out the following code:

```
<html>
<head>
      <script language="javascript">
      document.bgColor="black"
      document.linkColor="cyan"
      document.vlinkColor="yellow"
      document.fgColor="white"
      </Script>
</head>
<body>
</body>
```

1.3.3 Methods

[1] Note that JavaScript is case sensitive, so you must type fgColor instead of fgcolor. Fontpage automatically shows you the properties once you type a period at the end of the object name.

A method is a task or command performed by an object. For example, to display a phrase on a web page, we can use the write method; the syntax is `document.write (‘Phrase”)`. For example, `document.write(“Welcome to JavaScript”)` will display the “Welcome to JavaScript “ message. The general syntax to associate an object with a method is:

```
Object.Method
```

Since JavaScript is an OOP language, objects are arranged in a hierarchical manner, therefore sometime we need to define the parent objects too. Like this

```
ParentObject.ObjectName.ChildObjectName.Method
```

For example, a document is the parent object of a form and checkbox is a child object of the form, we can write the following code:

```
Document.Form1.Checbox1.Click
```

Beside the write method, methods associated with the form object and the controls that belong to the form object are frequently used. The common controls of a form are text box, check box and radio button. Another example is the submit method associated with the form, the code is,

```
Document.Form1.Submit
```

We will learn more about methods in later chapters.

1.3.4 Events

An event is an execution of the JavaScript code triggered by an action from the user via the keyboard or mouse. For example, clicking the submit button on a web page will trigger an event that validates the input data and transmits the data to the server. Examples of events are `Click, Load, KeyDown, KeyPress, DblClick , Select` and more.

To response to the aforementioned events, JavaScript uses event handlers. Some of the common event handlers are:

- `OnBlur`
- `OnClick`
- `OnChange`
- `OnFocus`
- `OnLoad`
- `onSubmit`
- `OnSelect`
- `OnUnload`
- `OnMouseOver`
- `OnKeyPress`
- `OnKeyDown`

We will learn how to write codes for the event handlers in later chapters.

Chapter 2
Writing the Code

2.1 The Structure of JavaScript

JavaScript does not require any special program to run it; you can use any standard text editors such as the Notepad, WordPad or FrontPage to type it, as it is embedded within a HTML document. The basic structure of the JavaScript that we embed into a HTML document, is

```
<Script language ="JavaScript”>
JavaScript Statements
</Script>
```

This `<Script></Script>` tags tell the web browser that the content in between is part of a script. Now in order to inform the browser which type of script is used by the page, you have to specify the `"type"` attribute of the script tag. The value in our case would be `"text/javascript"` since we want to use the JavaScript. Let me illustrate how do write the code using the example below:

Example 2.1

```
      <html>
         <head>

            <title>A First Program in JavaScript</title>

            <script language=”javascript” type = "text/javascript">

                document.write( "<b> Welcome to Our first program
</b>" )
```

```
          </script>

     </head>

  <body>

  </body>

  </html>
```

2.2 Variables

As we have discussed in chapter 1, JavaScript can accept data from the user, validates data and sends data to the server. To process and manipulates data, JavaScript uses variables. A variable is something like a mailbox with a label where its contents always change. When JavaScript creates a variable, it can store data or values input by the user or from some other sources. These values can vary as the user can input new data or values. To identify a particular variable, we always give it a name. For example, if we want to create a variable to store students' names, we can name it `StudentName`. We divide variables into a few types, as follows:

- Numeric – store numerical values where JavaScript can perform mathematical operations on them using standard operators
- String- Store only text
- Boolean- display True or False values

2.2.1 Declaring Variables

In JavaScript, we need to declare a variable before we can use it. The keyword we use to declare a variable in JavaScript is `Var`[2]. The statement to declare a variable is

```
Var VariableName
```

[2] Actually, it is not necessary to use the Var keyword but it is a good practice to include it.

There are a few rules regarding variable names in JavaScript, these rules are:

- ➢ The variable name must begin with a letter or an underscore (_). It cannot begin with a number or non-alphabetical characters. For examples, `name$ and _name` are valid variable names but `$name`, `2score` are invalid variable names.
- ➢ Space is not allowed within a variable name. For example, `my name` is not a valid variable name but `my_name` is a valid variable name.
- ➢ Variable names are case sensitive. For example, `Garden` and `garden` are different variables in JavaScript.
- ➢ JavaScript reserve certain words as statements or commands, these words cannot be used to assign variable names. Examples of such words are `if, for, else, goto, true, while` etc.

There are no hard and fast rules that you must obey in choosing a variable name, but it is best that you choose a name that is meaningful as it makes debugging your program easier. For example, you can use `Tel_No` for telephone number, which is better than just using a variable x.

2.2.2 Assigning Values to Variables

After you have created a variable, you can assign a value to it. The value can be in the form of number (numeric value), text (String) or Boolean (true or false).The syntax to assign a value in JavaScript is

```
var variable_name= value
```

Examples

```
var mark = 20
var sell_price=100
var age = 40
var height = 180
var temp = 30
var First_Name="George"
var car_model = " Laser 100"
var Company_Name = ' Ultra Mobile LLC '
var paid = true
var alive = false
var positive = true
```

We can also initialize the variable with mathematical operators like this,

```
Var CostPrice = 50
Var SellPrice =80
Var Profit=SellPrice-CostPrice
```

The above statements initialize the value of profit using the initial value of `Sellprice` minus the initial value of `CostPrice.`

Let us look at one working example:

Example 2.2

```
<Script language="JavaScript">

var             Author_Name=" John Brandon",
                 Author_ID      = "A1234",
                 ISBN             ="0-7789-0234-6",
                 Royalty        = 100000;
                 document.writeln("Author
Name:"+Author_Name+"<BR>");
               document.writeln(" Author ID:"+Author_ID+"<BR>");
               document.writeln("ISBN:"+ISBN+"<BR>");
               document.writeln("Royalty Payment: $"+Royalty)
</Script>
```

*It is neither necessary to use the keyword `var` to declare the variables nor put semicolons at the end of each statement in JavaScript, but it is good practice to include them.

2.3 Operators

2.3.1 Arithmetic Operators

JavaScript is not just a scripting language; it can also perform arithmetic calculations. It can interact with online users by receiving data from them, process the data and then output the results to them. The arithmetic operators in JavaScript are shown in Table 2.1

Table2.1: Arithmetic Operators

Operators	Arithmetic Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (Returns the remainder) Example: 10%3=1
++	Increment by 1
- -	Decrement by 1

Table2.1: Arithmetic Operators

Example 2.3

This example demonstrates how mathematical operations are carried out:

```
<Script  language= “JavaScript/Text”>
      number1=10
      number2=8
      sum=number1+number2
      diff= number1-number2
      product=number1*number2
      quotient=number1/number2
      document.write(number1+”+”+number2+”=”+sum+<br>)
      document.write(number1+”-”+number2+”=”+diff+<br>)
      document.write(number1+”x”+number2+”=”+product+<br>)
      document.write(number1+”÷”+number2+”=”+quotient+<br>)
</Script>
```

The results are displayed below:

```
10+8=18
10-8=2
10x8=80
10÷8=1.25
```

The following example shows the usage of more arithmetic operators:

Example 2.4

```
<script language="javascript">
      a=10
      b=-10
      c=b%3
      a++
      b--
      document.write(a+"<br>")
      document.write(b+"<br>")
      document.write(c)
</script>
```

The output results are shown here

```
 11
-11
-1
```

2.3.2 Assignment Operators

JavaScript uses assignment operators to assign value to a variable. We have seen how it is done in the earlier section, but we shall examine them in details here.

The simplest operator to assign value to a variable is to use the equal sign (=). Let us examine some examples below:

```
Name="John", Price=50, Age=21, Password="qwerty1987"
```

We can also assign values using arithmetic operators. For example,

```
Price= 100
Cost=80
Profit=Price-Cost
```

In addition, we can combine two arithmetic operators to form an assignment operator. For example,

```
total+=commission
```

The above expression in long form is `total=total +commission`.

Let us put the above assignment in actual JavaScipt Code, as follows:

```
<script language="javascript">
      total=100
      commission=15
      total+=commission
      alert("Total="+total)
</script>
```

The value of total is 100+15=115

Table 2.2 shows other assignment operators

Operator	Example
-=	X-=Y is equivalent to X=X-Y
*=	X*=Y is equivalent to X=X*Y
/=	X/= is equivalent to X=X/Y
%=	X%=Y equivalent to X=X%Y (returns the remainder of X/Y)

Table 2.2: Assignment Operators

2.3.3 Comparison Operators

In previous section, we have learned the usage of arithmetic operators in JavaScript, in this section; we shall introduce the comparison operators. Using comparison operators, we can compare two values in an expression and evaluate whether the comparison produces a true or false result. Table 2.3 lists the comparison operators.

Comparison Operator	Description
x==y	x is equal to y
x!=y	x is not equal to y
x>y	x is more than y
x<y	x is less than y
x>=y	x is more than or equal to y
x<=y	x is less than or equal to y

Table 2.3: Comparison Operators

Let us examine the following example:

A college student has the following information in the college database.

```
FirstNname: Ryan   , LastName: John, StudentID: RJ2011Fall,
Age:  21, Attendance: 30, Year: 3
```

The following JavaScript statements produce true or false results.

Statement	**True/False**
FirstName== ‘Ryan’	True
LastName==”George”	False
StudentID==”RJ2012Spring”	False
Attendance>20	True
FirstName!= “John”	True
Age<20	False
Year>=3	True
Age<=20	True

Table 2.4

2.3.4 Logical Operators

Besides the comparison operators, JavaScript uses logical operators to make decisions. They are often used to verify information, particularly the userID and password. There are three logical operators here, as shown below:

```
&&  represents AND

||  represents OR
```

```
    ! represents NOT
```

The logical operators && and II are used together with the comparison operators while the logical operator ! operates by itself. The logical expressions are as follows:

```
Value1 && Value 2

Value1 || Value 2

! Value
```

For the `AND (&&)` operator, the logical expression is true if and only if the values on both sides are true, it is false if one of the values is false. The concept is illustrated in the following example:

```
if (UserName=="John201" && Password=="qwert19" ){ message="Login
Successful"}
```

If both the UserName and the Password are correct, then login is successful.

Here is another example:

```
if ( mark>=60 && mark<80){grade="B"}
```

if the mark is more than and equal to 60 and the mark is less than 80, the grade is B.

For the `OR (||)` operator, the logical expression is true if one of the values is true, it is false if only the two values are false. The concept is illustrated in the following example:

```
if (age>=12 ||  height>140){Entrance Fee="$10"}
```

If the age of the person is more than 12, even though his or her height is less than 140, the entrance fee is $10. On the other hand, it the person age is less than 12 but his or her height is greater than 140; the entrance fee is also $10.

For the `Not(!)` operator, it reverses the logical statement. The concept is illustrated in the following example:

`!(Status= ‘pass’)` produces a false result, that is fail.

Here is another example:

```
   If (! (stockIndex<1000)) document.write ("The market is
good")
```

The application of the logical operators is shown in Example 2.7 :

Example 2.7

```
<html>
<head>
<title>Logical Operators</title>
</head>
<body>
<script language="javascript">
      var age, height, entrancefee
      age=window.prompt("Enter your age:","0")
      height=window.prompt("Enter your height:","0")
      if (age>=12 || height>=140){entrancefee="$10"}
      else {entrancefee="$8"};
      document.write("The entrance fee is: "+entrancefee);
</script>
</body>
```

2.3.5 Conditional Operators

Conditional operators let JavaScript program executes certain jobs according to certain conditions. The format of the conditional operator is

```
Conditional Expression ? Execution 1: Execution 2
```

The conditional operator comprises three parts, the first part is the conditional expression, the second part is execution 1 and the last part is execution 2. The question mark **?** separates the conditional expression and execution 1 whilst the colon **:** separates execution 1 and execution 2. If the conditional expression is true, then JavaScript will run execution 1 and ignores execution2. However, if the conditional expression is false, then JavaScript will ignore execution 1 and run execution 2. Therefore, the symbol **?** and **:** completely replace If and Else.

Example 2.8

In this example, since the expression x>y is false; JavaScript executes the second action, i.e. to display 20 in the browser.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1252">
<title>Conditional Operators</title>
</head>
<body>
<script language="javascript">
x=10
y=20
x>y?document.write(x):document.write(y)

</script>
</body>
```

Example 2.9

We can modify the above example to make it more interactive by prompting the user to enter the values.

```
<html>
<head>
</head>
<body>

      <script language="javascript">
      x=window.prompt("Enter first value",0)
      y=window.prompt("Enter second value",0)
      x>y?document.write(x):document.write(y)

</script>
</body>
```