

JavaScript & JQuery Made Easy

by Dr.Liew

Year Published: 2020

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.

Copyright ® 2020 Liew Voon Kiong.

All rights reserved. No part of this e-book may be reproduced or distributed, in any form or by any means, without permission in writing from the author.

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 that, he is the author of other programming books, among them are **Visual Basic 2019 Made Easy**, **Visual Basic 2017 Made Easy**, **Visual Basic 2015 Made Easy**, **Visual Basic 2013 Made Easy**, **Visual Basic 2012, Made Easy**, **Visual Basic 2010 Made Easy**, **Visual Basic 2008 Made Easy**, **Visual Basic 6 Made Easy**, **HTML & CSS Made Easy**, **Blockchain Made Easy** and **Excel VBA Made Easy**.

Table of Contents

Section I	11
JavaScript	11
Chapter 1 Introduction to JavaScript	12
1.1 What is JavaScript?	12
1.2 Why use JavaScript?	12
1.3 The Building Block of JavaScript	13
1.3.1 Objects	13
1.3.2 Properties	14
1.3.3 Methods	15
1.3.4 Events	16
Chapter 2 Writing the Code	17
2.1 The Structure of JavaScript	17
Example 2.1	17
2.2 Variables	18
2.2.1 Declaring Variables	18
2.2.2 Assigning Values to Variables	19
Example 2.2	19
Example 2.3	20
2.3 Operators	20
2.3.1 Arithmetic Operators	20
Example 2.4	20
Example 2.5	21
2.3.2 Assignment Operators	22
2.3.3 Comparison Operators	24
2.3.4 Logical Operators	25
Example 2.6	26
2.3.5 Conditional Operators	27
Example 2.7	27
Example 2.8	28
Chapter 3 Decision Making	29
3.1 The Structure of Decision-Making Procedure	29
Example 3.1	29
Example 3.2 Using if statements together with comparison operators	30
3.2 Using if and else keywords	30
Example 3.3	31
3.3 Using if, else and else if keywords	31
Example 3.4 Conversion of mark to grade	32
3.4 Looping	32
3.4.1 The for Loop	33
Example 3.5	33

3.4.2 The for in Loop	34
Example 3.6: Retrieving the properties of a document	34
Example 3.7: Retrieving the properties of an object	35
Example 3.8: Calculate the sum of numbers in an array	36
3.4.3 The While Loop	37
Example 3.9	37
Chapter 4 Functions	38
4.1 Definition of Function	38
4.2 Creating User-Defined Functions	38
Example 4.1	39
Example 4.2 Future Value Calculator	41
Example 4.3 Enhanced Future Value Calculator	42
Example 4.4: Maximum Number function	43
Example 4.5: A Dice Function	44
Example 4.6 A Graphical Dice	46
Chapter 5 Working with the Document Object	49
5.1 Document Properties	49
5.2 Document Methods	50
Example 5.1: Combining document properties and methods	53
Chapter 6 Working with the Math Object	54
6.1 Math Methods	54
Example 6.1	56
6.2 Math Properties	57
Example 6.2	58
Chapter 7 Working with the String Object	59
7.1 The length Property	59
7.2 The String Methods	59
Example 7.1	62
7.3 Converting Strings to Numbers	63
Example 7.2	63
Chapter 8 Working with the Date Object	65
8.1. The Date Object	65
Example 8.1 Displaying Date and Time	66
8.2 Setting Date and Time	67
Example 8.2	68
Chapter 9 Handling Events	69
9.1 Event Handlers for Links	70
Example 9.1	70
Example 9.2	71
Example 9.3: Future Value Function	72
9.2 Event Handlers for text and textarea	73
Example 9.4: Event Handlers for text Object	74
Example 9.5: Event Handlers for textarea Object	75

9.3 Event Handlers for Buttons	77
Example 9.6: Calculation of Future Value	78
Chapter 10 Working with Form and its Elements	79
10.1 The Structure of the Form	79
Example 10.1 Using the onsubmit event handler to send email	81
Example 10.2	81
Example 10.3: Email Validation	82
10.2 Creating the Selection List	84
Example 10.4: To Check Index of a Selected Item	85
Example 10.5: Selection List of Web Links	86
10.3 Creating Radio Buttons	88
Example 10.6	88
Example 10.7: Item Selection Using Radio Buttons	89
Example 10.8: Image Selection	90
10.4 Creating Checkboxes	92
Example 10.9: Code for Item Selection	93
Example 10.10: To Check Status of Multiple Checkboxes	94
Chapter 11 Creating Arrays	95
11.1 Introduction	95
11.2 Declaring and Allocating Arrays	95
Example 11.1	96
11.3: Declaring Arrays with Known Initial Values	97
11.4 Performing Arithmetic Operations on Values of an Array	98
11.5 Tabulating the Values of an Array	99
11.6 Sorting an Array	100
Chapter 12 Working with Frames	103
12.1 Creating Frames	104
12.2 The Hierarchy of Frames	107
12.3 Frames Manipulation Using JavaScript	108
Example 12.1	108
Example 12.2: A Graphical Dice	109
Chapter 13 Creating Cookies	112
13.1 Creating Cookies	112
13.1.1 Creating Temporary Cookies	112
Example 13.1: Cookie Creation	113
13.1.2 Creating Dated Cookies	114
Example 13.2: Creating Cookie with Expiration Date	114
13.2 Reading Cookies	115
Example 13.3: Reading Cookie	116
Example 13.4: Creating and Reading Cookie	117
Chapter 14 Creating Graphics	118
14.1 Drawing a Straight Line	118
Example 14.1	119
14.2 Drawing a Triangle	120

14.3 Drawing a Rectangle with a Solid Color	121
14.4 Drawing a Rectangle with Color in Gradient	124
14.4.1 Linear Gradient	124
14.4.1 Radial Gradient	125
14.5 Drawing an Arc/ Circle	127
Chapter 15 Creating Multimedia Content	132
15.1 Creating Banner Ads	132
15.1.1 Creating Rotating Banner Ads	132
15.1.2 Creating Rotating Banner Ads with URL Links	134
15.2 Creating a Slideshow	136
15.3 Creating Rollover Effects	139
15.3.1 Creating Rollovers using HTML	139
Example 15.1	139
15.3.2 Creating Rollovers Involving Text and Images	140
15.3.3 Creating Rollovers using JavaScript	142
Chapter 16 Creating Objects	144
16.1 Using Literal Notation	144
Example 16.1	144
Example 16.2	146
16.2 Creating an Object using Object Constructor Notation	147
Chapter 17 Creating Animation	149
17.1 The setInterval() Method	149
Example 17.1	149
Example 17.2 Displaying Current Time	150
17.2 The clearInterval() Method	151
Example 17.3	151
Example 17.4	152
Example 17.5	152
Chapter 18 Interesting Examples	153
18.1 Animated Digital Dice	153
18.2 Animated Graphical Dice	154
18.3 Animated Butterfly	156
18.4 The Countdown Timer	158
18.5 Basic Calculator	159
18.6 Math Drill	161
18.7 Pythagoras Theorem	163
18.8 Quadratic Graph	165
18.9 Music Player	169
18.10 Video Player	171
Section II	173
JQuery	173
Chapter 19 Introduction to jQuery	174

19.1 Introduction	174
19.2 jQuery Basics	175
Example 19.1	176
Example 19.2	178
Chapter 20 jQuery Selectors	179
20.1 The Element Selector	179
Example 20.1	179
20.2 The Class Selector	181
Example 20.2	181
20.3 The id Selector	182
Example 20.3	182
20.4 The * Selector	183
Example 20.4	183
20.5 The Current Element Selector	184
Example 20.5	184
Example 20.6	185
This is my heading	186
20.6 Select the first Element	186
Example 20.7	186
20.7 Selects the first element of the first 	187
Example 20.8	187
Chapter 21 jQuery Events	189
Table 21.1 List of Events	189
21.1 JQuery Event Handler	190
21.2 The Click() event	190
Example 21.1	190
JQuery Click Event	191
21.3 The dblclick() event	191
Example 21.2	191
21.4 The mouseenter() Event	193
Example 21.3	193
21.5 The mouseleave() Event	194
Example 21.4	194
21.6 The mousedown() Event	195
Example 21.5	195
21.7 The mouseup() Event	196
Example 21.6	196
21.8 The hover() Event	197
Example 21.7	197
21.9 The focus() Event	198
Example 21.8	198
21.10 The blur() Event	199
Example 21.9	199
Chapter 22 jQuery Methods	201

22.1 The hide() Method	201
Example 22.1	201
Hide Me	202
22.2 The show() Method	202
Example 22.2	202
22.3 The Speed Parameters	203
Example 22.3	203
22.4 The toggle() Method	205
Example 22.4	205
22.5 The on() Method	206
Example 22.5	206
Example 22.6 Changing Color	207
Example 22.7	207
Example 22.8	209
22.6 The fadeIn() Method	210
Example 22.9	210
22.7 The fadeOut() Method	212
Example 22.10	212
22.8 The fadeToggle() Method	213
Example 22.11	213
22.9 The fadeTo() Method	215
Example 22.12	215
22.10 The slideDown() Method	216
Example 22.13	216
22.11 The slideUp() Method	218
Example 22.14	218
22.12 The slideToggle() Method	219
Example 22.15	219
22.13 The stop() Method	220
Example 22.16	220
22.13 The animate() Method	221
Example 22.17	221
Example 22.18	222
Example 22.19	223
Example 22.20	224
Chapter 23 Chaining	226
Example 23.1	226
Example 23.2	227
Example 23.3	228
Example 23.4	229
Chapter 24 Content Manipulation	231
24.1 Retrieving Content	231
24.1.1 The text() Method	231
Example 24.1	231
24.1.2 The html() Method	232

Example 24.2	232
24.1.3 The val() Method	233
Example 24.3	233
24.1.4 The attr() Method	234
Example 24.4	234
Example 24.5	234
24.2 Modifying Content	236
24.2.1 The text() Method	236
Example 24.6	236
24.2.2 The html() Method	238
Example 24.7	238
24.2.3 The val() Method	239
Example 24.8	239
24.2.4 The attr() Method	240
Example 24.9	240
24.3 Adding New Content	242
24.3.1 The append() Method	242
Example 24.10	242
24.3.2 The prepend() Method	244
Example 24.11	244
24.3.3 The after() Method	245
Example 24.12	245
24.3.4 The before() Method	246
Example 24.13	246
24.4 Removing New Content	247
24.4.1 The remove() Method	247
Example 24.14	247
24.4.2 Removing elements with a Filtering Parameter	249
Example 24.15	249
Example 24.16	250
24.4.3 The empty() Method	252
Example 24.17	252
Chapter 25 Manipulating CSS	254
25.1 The addclass() Method	254
Example 25.1	254
25.2 The removeClass() Method	257
Example 25.2	257
25.3 The toggleClass() Method	260
Example 25.3	260
25.4 The css() Method	262
Example 25.4	262
Example 25.5	264
Chapter 26 Looping	266
26.1 The each() Method	267
Example 26.1	267

Example 26.2	269
26.2 The <code>\$.each()</code> function	271
Example 26.3	271
Example 26.4	272
Chapter 27 Mathematical Operations	275
27.1 Using Arithmetic Operators	275
Example 27.1	275
Example 27.2	277
27.2 Using the <code>Math()</code> Method	279
Table 27.1 <code>Math()</code> Methods	279
Example 27.3	280
Example 27.4	281
Example 27.5	283
Chapter 28 Creating Animation	285
Example 28.1 Digital Dice	285
Example 28.2 Graphical Dice	286

Section I

JavaScript

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 the web. 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 slideshow, 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 JAVA, C++, C# and more. Despite its slightly 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 has been the core tool in building and designing web pages. HTML or HyperText 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, checkboxes, buttons, windows and more. All these elements are objects.

We identify each object in a web page by its name. For example, the default name of a textbox is `Textbox1`; if we insert another textbox, 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 textbox are name, height, width, background color and foreground color, with border or no border, font type ,font 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 a red background.

You can try out the following code using different colors.

```
<Script>  
document.bgcolor="Red"  
</Script>
```

Other properties of the document object include `fgColor`, `linkColor`, `vlinkColor` and more. By the way, `fgColor` means foreground color, `linkColor` means hyperlinks color and `vlinkColor` means visited links color on a web page. We will introduce more properties in later chapters. Please note that some properties might work in some browsers but not others. 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>  
</html>
```

1.3.3 Methods

A method is a task or command that an object can use to execute an action. 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 sometimes 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 the checkbox is a child object of the form. The following code reflects their hierarchical relationship:

```
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, checkbox 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 respond 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

JavaScript does not require any special program to create it; you can simply use any standard text editors such as the Notepad, WordPad , Notepad++ and more to write its code.

2.1 The Structure of JavaScript

The basic structure of the JavaScript that we embed into a HTML document, is

```
<Script>
JavaScript Statements
</Script>
```

This `<Script></Script>` tags tell the web browser that the content in between is part of a JavaScript. Let 's examine the example below:

Example 2.1

```
<html>

<head>

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

<script>

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

</script>

</head>

</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 manipulate data, JavaScript uses variables. A variable is something like a mailbox with a label where its content always changes. 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

String- Store only text

Boolean- display True or False

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`¹. The statement to declare a variable is

```
Var VariableName
```

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 example, `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.

¹ Actually, it is not necessary to use the Var keyword but it is a good practice to include it.

JavaScript reserves 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, it will make debugging your program easier. For example, you can use `Tel_No` for telephone numbers, 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
```

Example 2.2

```
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.

Example 2.3

```
<Script>
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 not necessary to 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

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 2.4

This example demonstrates how mathematical operations are carried out:

```
<Script>

Var number1,number2,sum, diff, product, quotient;
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+" $\times$ " +number2+"=" $+$ product+<br>);
document.write(number1+" $\div$ " +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 2.5

```
<script >  
var a=10;  
var b=-10;  
var 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 detail here.

The simplest operator to assign value to a variable is to use the equal sign (=).

Let's examine some examples below:

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

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

```
var Price= 100;  
var Cost=80;
```

```
var Profit=Price-Cost;
```

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

```
var total+=commission;
```

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

Let us put the above assignment in a JavaScript, as follows:

```
<script>  
var total=100  
var 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
<code>-=</code>	$X-=Y$ is equivalent to $X=X-Y$
<code>*=</code>	$X*=Y$ is equivalent to $X=X*Y$
<code>/=</code>	$X/=$ is equivalent to $X=X/Y$
<code>%=</code>	$X\%=Y$ equivalent to $X=X\%Y$ (returns the remainder of X/Y)

Table 2.2: Assignment Operators

2.3.3 Comparison Operators

In the 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
<code>x==y</code>	<code>x</code> is equal to <code>y</code>
<code>x!=y</code>	<code>x</code> is not equal to <code>y</code>
<code>x>y</code>	<code>x</code> is more than <code>y</code>
<code>x<y</code>	<code>x</code> is less than <code>y</code>
<code>x>=y</code>	<code>x</code> is more than or equal to <code>y</code>
<code>x<=y</code>	<code>x</code> is less than or equal to <code>y</code>

Table 2.3: Comparison Operators

Let us examine the following example:

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

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

The following JavaScript statements produce true or false results.

Statement	True/False
<code>FirstName== 'Ryan'</code>	True
<code>LastName=="George"</code>	False
<code>StudentID=="RJ2012Spring"</code>	False

Attendance>20	True
FirstName!="John"	True
Age<20	False
Year>=3	True
Age<=20	False

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 || 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. 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, if the person's 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 2.5 :

Example 2.6

```
<html>
<head>

<title>Logical Operators</title>
```

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

2.3.5 Conditional Operators

Conditional operators let a JavaScript program execute certain jobs according to certain conditions. The syntax 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 ignore 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 2.7

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>
```

```
<title>Conditional Operators</title> </head>

<body>

<script>
var x=10;
var y=20;
x>y?document.write(x):document.write(y);

</script>

</body>
</html>
```

Example 2.8

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

```
<html>
<body>

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

</script>
</body>
</html>
```