

**Syllabus for JavaScript**

1. Introduction
2. Syntax
3. Statement and Comment
4. Variable
5. Data types
6. Operator
7. String
8. Event
9. Alert
10. Loop statement
11. Switch statement
12. While and do while Statement
13. For Loop statement
14. Function
15. Page Redirect
16. Page Printing

**1. Introduction**

**What JAVA is?**

Java is a programming language developed in 1995 for Sun Microsystems. Its main purpose is to be able to run on all different types of operating systems. In other words, a program written in Java can run on a Microsoft computer and an Apple computer. All you need is to download the Java run-time to your computer and you can run any Java applications.

Java can be used on the Web, but I haven’t seen an example of a site using Java in a long time. If someone is telling you they’ll build your website using Java, they probably mean JavaScript.

**What JAVASCRIPT is?**

JavaScript is built to run on the Web. How it executes and how well it performs is dependent on the type of browser you are using (Check out this website to help you determine: What is a browser?).

JavaScript was decried, denigrated and dismissed by many programmers for years, including myself. The relatively recent rise of “fancy” websites that update quickly without reloading a page (Ajax or asynchronous JavaScript execution) has brought JavaScript into prominence, and has even made an old hater like me at least acknowledge its importance.

If a salesperson is talking to you about using JavaScript to build something on your website, they are probably talking about implementing some sort of “fancy” interface feature to make the site more appealing to your visitors.

**What is JQuery?**

jQuery is what made it palatable and fun to start using JavaScript again. It is the most popular JavaScript library in use today, and if you are building a site you should be using it. If you are having a site built, when the sales person talks about JavaScript, it is almost certain that the developer will be utilizing jQuery to make it happen.

**2. Syntax**

JavaScript can be implemented using JavaScript statements that are placed within the script... script HTML tags in a web page.

You can place the script tags, containing your JavaScript, anywhere within you web page, but it is normally recommended that you should keep it within the head tags.

<script type=”text/javascript”>

Document.write (“Hello World”);

</script>

**3. Statement & Comment**

// Single line comment

/\* Multiple Line Comment \*/

**4. Variable**

One of the most fundamental characteristics of a programming language is the set of data types it supports. These are the type of values that can be represented and manipulated in a programming language.

* **Numbers** e.g. 123, 120.50 etc.
* **Strings** e.g. "This text string" etc.
* **Boolean** e.g. true or false.

Java does not make a distinction between integer values and floating-point values. All numbers in JavaScript are represented as floating-point values. JavaScript represents numbers using the 64-bit floating-point format defined by the IEEE 754 standard.

<script type=”text/javascript”>

var x=50;

var y=25;

var z=30;

document.write(x);

document.write('<br>');

document.write(y);

document.write('<br>');

document.write(z);

</script>

**5. Data Types**

<script type="text/javascript">

//integer

var myvariable = 20;

document.write(myvariable);

//float

var myvariable\_float = 30.95;

document.write(myvariable\_float);

//string

var myvariable\_string = 'what is your name? Answer is “my name is PANKAJ"';

document.write(myvariable\_string);

</script>

**6. Operator**

Let us take a simple expression 4 + 5 is equal to 9. Here 4 and 5 are called operands and ‘+’ is called the operator. JavaScript supports the following types of operators.

* Arithmetic Operators
* Comparison Operators
* Logical (or Relational) Operators
* Assignment Operators
* Conditional (or ternary) Operators

1. **Arithmetic Operators**

JavaScript supports the following arithmetic operators − Assume variable A holds 10 and variable B holds 20, then –

* 1. + Addition = Adds two operands Ex: A + B will give 30
  2. – Subtraction = Subtracts the second operand from the first Ex: A - B will give -10
  3. \* Multiplication = Multiply both operands Ex: A \* B will give 200
  4. / Division = Divide the numerator by the denominator Ex: B / A will give 2
  5. % Modulus = Outputs the remainder of an integer division Ex: B % A will give 0
  6. ++ Increment = Increases an integer value by one Ex: A++ will give 11
  7. -- Decrement = Decreases an integer value by one Ex: A-- will give 9

**Example : -**

<html>

<body>

<script type=”text/Javascript”>

var a = 33, b = 10, c = "Test", linebreak = "<br/>";

document.write("a + b = "); result = a + b; document.write(result); document.write(linebreak);

document.write("a - b = "); result = a - b; document.write(result); document.write(linebreak);

document.write("a / b = "); result = a / b; document.write(result); document.write(linebreak);

document.write("a % b = ");result = a % b; document.write(result); document.write(linebreak);

document.write("a + b + c = "); result = a + b + c; document.write(result); document.write(linebreak);

a = a++;document.write("a++ = "); result = a++; document.write(result); document.write(linebreak);

b = b--; document.write("b-- = "); result = b--; document.write(result); document.write(linebreak);

</script>

</body>

</html>

**OUTPUT**

a + b = 43

a - b = 23

a / b = 3.3

a % b = 3

a + b + c = 43Test

a++ = 33

b-- = 10

1. Comparison Operators

JavaScript supports the following comparison operators − Assume variable A holds 10 and variable B holds 20, then −

1. = = Equal Checks if the value of two operands are equal or not, if yes, then the condition becomes true. Ex: A == B is not true.
2. != Not Equal Checks if the value of two operands are equal or not, if the values are not equal, then the condition becomes true. Ex: A! = B is true.
3. > Greater than Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true. Ex: A > B is not true.
4. < Less than Checks if the value of the left operand is less than the value of the right operand, if yes, then the condition becomes true. Ex: A < B is true.
5. >= Greater than or Equal to Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true. Ex: A >= B is not true.
6. <= Less than or Equal to Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true. Ex: A <= B is true.

**Example : -**

<html>

<body>

<script type=”text/Javascript”>

var a = 10, b = 20, linebreak = "<br />";

document.write("(a == b) => "); result = (a == b); document.write(result); document.write(linebreak);

document.write("(a < b) => "); result = (a < b); document.write(result); document.write(linebreak);

document.write("(a > b) => "); result = (a > b); document.write(result); document.write(linebreak);

document.write("(a != b) => "); result = (a != b); document.write(result); document.write(linebreak);

document.write("(a >= b) => ");result = (a >= b); document.write(result)document.write(linebreak);

document.write("(a <= b) => "); result = (a <= b);document.write(result); document.write(linebreak);

</script>

</body>

</html>

**OUTPUT**

(a == b) => false

(a < b) => true

(a > b) => false

(a != b) => true

(a >= b) => false

a <= b) => true

1. **Logical Operator**
   1. && Logical AND If both the operands are non-zero, then the condition becomes true. Ex: A && B is true.
   2. || Logical OR If any of the two operands are non-zero, then the condition becomes true. Ex: A | | B is true.
   3. ! Logical NOT Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false. Ex: ! A && B is false

<html>

<body>

<script type=”text/Javascript”>

var a = true; var b = false; var linebreak = "<br />";

document.write("(a && b) => "); result = (a && b); document.write(result); document.write(linebreak);

document.write("(a || b) => "); result = (a || b); document.write(result); document.write(linebreak);

document.write("!(a && b) => "); result = (!(a && b)); document.write(result); document.write(linebreak);

</script>

</body>

</html>

**OUTPUT**

(a && b) => false

(a || b) => true

!(a && b) => true

1. **Assignment Operator**
2. = Simple Assignment Assigns values from the right side operand to the left side operand Ex: C = A + B will assign the value of A + B into C
3. += Add and Assignment It adds the right operand to the left operand and assigns the result to the left operand. Ex: C += A is equivalent to C = C + A
4. −= Subtract and Assignment It subtracts the right operand from the left operand and assigns the result to the left operand. Ex: C -= A is equivalent to C = C – A
5. \*= Multiply and Assignment It multiplies the right operand with the left operand and assigns the result to the left operand. Ex: C \*= A is equivalent to C = C \* A
6. /= Divide and Assignment It divides the left operand with the right operand and assigns the result to the left operand. Ex: C /= A is equivalent to C = C / A
7. %= Modules and Assignment It takes modulus using two operands and assigns the result to the left operand. Ex: C %= A is equivalent to C = C % A

<html>

<body>

<script type=”text/Javascript”>

var a = 33; var b = 10; var linebreak = "<br />";

document.write("Value of a => (a = b) => "); result = (a = b); document.write(result); document.write(linebreak);

document.write("Value of a => (a += b) => "); result = (a += b); document.write(result); document.write(linebreak);

document.write("Value of a => (a -= b) => "); result = (a -= b); document.write(result). document.write(linebreak);

document.write("Value of a => (a \*= b) => "); result = (a \*= b); document.write(result); document.write(linebreak);

document.write("Value of a => (a /= b) => "); result = (a /= b); document.write(result); document.write(linebreak);

document.write("Value of a => (a %= b) => "); result = (a %= b); document.write(result); document.write(linebreak);

</script>

</body>

</html>

**OUTPUT**

Value of a => (a = b) => 10

Value of a => (a += b) => 20

Value of a => (a -= b) => 10

Value of a => (a \*= b) => 100

Value of a => (a /= b) => 10

Value of a => (a %= b) => 0

1. **Conditional Operators**

The conditional operator first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation. –

* 1. ? : Conditional If Condition is true? Then value X : Otherwise value Y

**Example:-**

<html>

<body>

<script type=”text/Javascript”>

var a = 10; var b = 20; var linebreak = "<br />";

document.write ("((a > b) ? 100 : 200) => "); result = (a > b) ? 100 : 200; document.write(result); document.write(linebreak);

document.write ("((a < b) ? 100 : 200) => "); result = (a < b) ? 100 : 200; document.write(result); document.write(linebreak);

</script>

</body>

</html>

**OUTPUT**

((a > b) ? 100 : 200) => 200

((a < b) ? 100 : 200) => 100

**6. String**

The String object lets you work with a series of characters; it wraps Javascript's string primitive data type with a number of helper methods. As JavaScript automatically converts between string primitives and String objects, you can call any of the helper methods of the String object on a string primitive.

**Syntax**

Use the following syntax to create a String object –

var val = new String(string);

**Example :-**

<script type="text/javascript">

var name = "PANKAJ";

// escape charecter " \ "

//var massage = "Hello world,\" I am very Happy\"";

//document.write(massage);

// concantation

var sentance1 = " Hello World";

var sentance2 = " I am Very Happy ";

document.write(sentance1 + sentance2 );

// concantation with number

//var x =20;

//document.write('my age is ' + x);

// HTML in sring

document.write("<h1> This is Heading in string</h1> ");

</script>

**8. Alert**

Let us take a simple expression 4 + 5 is equal to 9. Here 4 and 5 are called operands and ‘+’ is called the operator. JavaScript supports the following types of operators.

* Alert On Page Load
* Alert On Button Click
* Alert On Confirmation Box
* Alert On Promot Box

1. **Alert On Load**

<html>

<body>

<script type="text/javascript">

alert("ENTER TEXT")

</script>

</body>

</html>

1. **Alert On Button Click**

<html>

<body>

<!--alert On page load -->

<button onmouseover="myFunction()">Click Me</button>

<script>

function myFunction() {

alert("Alert on function call");

}

</script>

</body>

</html>

1. **Alert On Confirmation Box**

<html>

<body>

<!-- Confirmation Box on click -->

<button onclick="Confirm()">Click for fonfirmation box</button>

<p id="demo"></p>

<p>without demo id</p>

<script>

function Confirm() {

var x;

if (confirm("Press a button!") == true) {

x = "You pressed OK!";

} else {

x = "You pressed Cancel!";

}

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

1. **Alert On Promot Box**

<html>

<body>

<!--Promt Box -->

<button onclick="Prompt()">Prompt</button>

<p id="demo"></p>

<script>

function Prompt() {

var person = prompt("", "Please Enter Your Age");

if (person != null) {

document.getElementById("demo").innerHTML =

"Hello " + person + "! How are you today?";

}

}

</script>

</body>

</html>

**9. Event**

When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.

Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable. Events are a part of the Document Object Model (DOM) Level 3 and every HTML element contains a set of events which can trigger JavaScript Code.

## Input Events

[onblur - When a user leaves an input field](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onblur)  
[onchange - When a user changes the content of an input field](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onchange)  
[onchange - When a user selects a dropdown value](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_dropdown)  
[onfocus - When an input field gets focus](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onfocus)  
[onselect - When input text is selected](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onselect)  
[onsubmit - When a user clicks the submit button](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onsubmit)  
[onreset - When a user clicks the reset button](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onreset)  
[onkeydown - When a user is pressing/holding down a key](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onkeydown)  
[onkeypress - When a user is pressing/holding down a key](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onkeypress)  
[onkeyup - When the user releases a key](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onkeyup)  
[onkeyup - When the user releases a key](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onkeyup2)  
[onkeydown vs onkeyup - Both](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onkeydownup)

**Mouse Events**

[onmouseover/onmouseout - When the mouse passes over an element](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onmouse)  
[onmousedown/onmouseup - When pressing/releasing a mouse button](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onmousedown)  
[onmousedown - When mouse is clicked: Alert which element](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_srcelement)  
[onmousedown - When mouse is clicked: Alert which button](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onmousedown2)  
[onmousemove/onmouseout - When moving the mouse pointer over/out of an image](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onmousemove)  
[onmouseover/onmouseout - When moving the mouse over/out of an image](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onmouseover)  
[onmouseover an image map](http://www.w3schools.com/js/tryit.asp?filename=tryjs_imagemap)

**Click Events**

[Acting to the onclick event](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events)  
[onclick - When button is clicked](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onclick)  
[ondblclick - When a text is double-clicked](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_ondblclick)

**Load Events**

[onload - When the page has been loaded](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_body_onload)  
[onload - When an image has been loaded](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_img_onload)  
[onerror - When an error occurs when loading an image](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onerror)  
[onunload - When the browser closes the document](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onunload)  
[onresize - When the browser window is resized](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onresize)

**10. Loop Statement**

1. If Statement
2. If .. Else Statement
3. If .. Else .. If Statement
4. **If Statement**

The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

Syntax

If (expression) {

Statement(s) to be executed if expression is true

}

Example :-

<html>

<body>

<p id="demo">Age must be above 18</p>

<p>Set the variable to different value and then try...</p>

<script type="text/javascript">

var age = 10;

if( age > 18 ){ document.getElementById("demo").innerHTML = 'Eligible For Licences';

}

</script>

</body>

</html>

1. **If .. Else Statement**

The 'if...else' statement is the next form of control statement that allows JavaScript to execute statements in a more controlled way.

Syntax

If (expression) {

Statement(s) to be executed if expression is true

}

Else{

State ment(s) to be executed if expression is False

}

Example :-

<html>

<body>

<p id="demo">Age must be above 18</p>

<p>Set the variable to different value and then try...</p>

<script type="text/javascript">

var age = 15;

if( age > 18 ){

document.write("<b>Qualifies for driving</b>");

}

else{

document.write("<b>Does not qualify for driving</b>");

}

</script>

</body>

</html>

1. **If .. Else If.. Statement**

The if...else if... statement is an advanced form of if…else that allows JavaScript to make a correct decision out of several conditions.

Syntax

If (expression 1) {

Statement(s) to be executed if expression1 is true

}

Else If (expression 2) {

Statement(s) to be executed if expression2 is true

}

Else if If (expression 3) {

Statement(s) to be executed if expression 3 is true

}

Else{

State ment(s) to be executed if all expression is False

}

Example :-

<html>

<body>

<p id="demo">Age must be above 18</p>

<p>Set the variable to different value and then try...</p>

<script type="text/javascript">

var book = "history";

var librarian = 20 + book;

if( librarian === 20 + book ){

document.write("<b>Librainan Msg Book</b>");

}

else if( book == "maths" ){

document.write("<b>Maths Book</b>");

}

else if( book == "economics" ){

document.write("<b>Economics Book</b>");

}

else{

document.write("<b>Unknown Book</b>");

}

</script>

</body>

</html>

**11. Switch Statement**

You can use multiple if...else…if statements, as in the previous chapter, to perform a multiway branch. However, this is not always the best solution, especially when all of the branches depend on the value of a single variable.

Starting with JavaScript 1.2, you can use a switch statement which handles exactly this situation, and it does so more efficiently than repeated if...else if statements.

Syntax

switch (expression ) {

case condition 1 : statement(s);

break;

case condition 2 : statement(s);

break;

case condition 3 : statement(s);

break;

default : Statement(s);

}

Example :-

<html>

<body>

<script type="text/javascript">

var grade='B';

document.write("Entering switch block<br />");

switch (grade)

{

case 'A': document.write("Good job<br />");

break;

case 'B': document.write("Pretty good<br />");

break;

case 'C': document.write("Passed<br />");

break;

case 'D': document.write("Not so good<br />");

break;

case 'F': document.write("Failed<br />");

break;

default: document.write("Unknown grade<br />")

}

document.write("Exiting switch block");

</script>

</body>

</html>

**12. While and Do while Statement**

While writing a program, you may encounter a situation where you need to perform an action over and over again. In such situations, you would need to write loop statements to reduce the number of lines.

JavaScript supports the following forms of While Loop –

1. While Loop
2. Do..While
3. **While Loop**

The most basic loop in JavaScript is the **while** loop which would be discussed in this chapter. The purpose of a **while** loop is to execute a statement or code block repeatedly as long as an **expression** is true. Once the expression becomes **false,** the loop terminates.

Syntax

While (expression 1) {

Statement(s) to be executed if expression1 is true

}

Example :-

<html>

<body>

<script type="text/javascript">

var count = 0;

document.write("Starting Loop ");

while (count < 9){

document.write("Current Count : " + count + "<br />");

count++;

}

document.write("Loop stopped!");

</script>

</body>

</html>

Output

Starting Loop

Current Count : 0

Current Count : 1

Current Count : 2

Current Count : 3

Current Count : 4

Current Count : 5

Current Count : 6

Current Count : 7

Current Count : 8

Loop stopped!

1. **Do ... While Loop**

The **do...while** loop is similar to the **while** loop except that the condition check happens at the end of the loop. This means that the loop will always be executed at least once, even if the condition is **false**.

Syntax

do{

Statement(s) to be executed;

} while (expression);

Example :-

<html>

<body>

<script type="text/javascript">

var count = 0;

document.write("Starting Loop" + "<br />");

do{

document.write("Current Count : " + count + "<br />");

count++;

}

while (count < 5);

document.write ("Loop stopped!");

</script>

</body>

</html>

**Output**

Starting Loop

Current Count : 0

Current Count : 1

Current Count : 2

Current Count : 3

Current Count : 4

Loop Stopped!

**13. For Loop**

The '**for**' loop is the most compact form of looping. It includes the following three important parts −

* The **loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
* The **test statement** which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
* The **iteration statement** where you can increase or decrease your counter.

You can put all the three parts in a single line separated by semicolons.

### Syntax

for (initialization; test condition; iteration statement){

Statement(s) to be executed if test condition is true

}

### Example

Try the following example to learn how a for loop works in JavaScript.

<html>

<body>

<script type="text/javascript">

var count;

document.write("Starting Loop" + "<br />");

for(count = 0; count < 10; count++){

document.write("Current Count : " + count );

document.write("<br />");

}

document.write("Loop stopped!");

</script>

</body>

</html>

**Output**

Starting Loop

Current Count : 0

Current Count : 1

Current Count : 2

Current Count : 3

Current Count : 4

Current Count : 5

Current Count : 6

Current Count : 7

Current Count : 8

Current Count : 9

Loop stopped!

**14. Function**

Before we use a function, we need to define it. The most common way to define a function in JavaScript is by using the **function** keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

**Syntax**

The basic syntax is shown here.

<script type="text/javascript">

function functionname(parameter-list)

{

statements

}

</script>

**Example**

Try the following example. It defines a function called sayHello that takes no parameters −

<script type="text/javascript">

function sayHello()

{

alert("Hello there");

}

</script>

**Calling a Function**

To invoke a function somewhere later in the script, you would simply need to write the name of that function as shown in the following code.

<html>

<head>

<script type="text/javascript">

function sayHello()

{

document.write ("Hello there!");

}

</script>

</head>

<body>

<p>Click the following button to call the function</p>

<form>

<input type="button" onclick="sayHello()" value="Say Hello">

</form>

<p>Use different text in write method and then try...</p>

</body>

</html>

**Output**

**Function**

**Parameters**

Till now, we have seen functions without parameters. But there is a facility to pass different parameters while calling a function. These passed parameters can be captured inside the function and any manipulation can be done over those parameters. A function can take multiple parameters separated by comma.

**Example**

Try the following example. We have modified our **sayHello** function here. Now it takes two parameters.

<html>

<head>

<script type="text/javascript">

function sayHello(name, age)

{

document.write (name + " is " + age + " years old.");

}

</script>

</head>

<body>

<p>Click the following button to call the function</p>

<form>

<input type="button" onclick="sayHello('Zara', 7)" value="Say Hello">

</form>

<p>Use different parameters inside the function and then try...</p>

</body>

</html>

**The return Statement**

A JavaScript function can have an optional **return** statement. This is required if you want to return a value from a function. This statement should be the last statement in a function.

For example, you can pass two numbers in a function and then you can expect the function to return their multiplication in your calling program.

**Example**

Try the following example. It defines a function that takes two parameters and concatenates them before returning the resultant in the calling program.

<html>

<head>

<script type="text/javascript">

function concatenate(first, last)

{

var full;

full = first + last;

return full;

}

function secondFunction()

{

var result;

result = concatenate('Zara', 'Ali');

document.write (result );

}

</script>

</head>

<body>

<p>Click the following button to call the function</p>

<form>

<input type="button" onclick="secondFunction()" value="Call Function">

</form>

<p>Use different parameters inside the function and then try...</p>

</body>

</html>

## 14. Page Redirection

Before we use a function, we need to define it. The most common way to define a function in JavaScript is by using the **function** keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

Example 1 :-

<html>

<head>

<script type="text/javascript">

function Redirect() {

window.location="http://www. enosislearning.com";

}

</script>

</head>

<body>

<p>Click the following button, you will be redirected to home page.</p>

<form>

<input type="button" value="Redirect Me" onclick="Redirect();" />

</form>

</body>

</html>

Example 2 :-

<html>

<head>

<script type="text/javascript">

function Redirect() {

window.location="http://www.enosislearning.com";

}

document.write("You will be redirected to main page in 10 sec.");

setTimeout('Redirect()', 10000);

</script>

</head>

<body>

</body>

</html>