Introduction to JavaScript

It is a client side scripting language.

It is a light weight programming language.

It is case sensitive.

It is an interpreted language. (Does not require compiler)

It is executed by client browser.

Its extension is .js [external JavaScript]

It provides programming tool for Html designers.

It can read/write form fields.

It can detect client browsers.

It can trigger events based on user action. (Mousemove, click, onblur)

It supports cookies for storing information for state management.

We can use it for performing form validation.

It is embedded in html itself.

Html provides <script>...</script> tag for embedding JavaScript.

Syntax: -

```
<script type=”text/JavaScript”>
        //logic.
</script>
```

Syntax:

```
<Html>

        <Head>

                <Title>JavaScript</title>

                <script type="text/JavaScript">

                        <!--

                Define the JavaScript here

                        //-->

                </script>

        </head>

        <Body>

        </body>

</html>
```

```
<!doctype html>
<html>
	<head>
		<Title>JavaScript demo</title>
		<script type="text/JavaScript">
			function displayMessage()
			{
				alert("Welcome guest! ");
			}
		</script>
	</head>
	<body>
		<h1>JavaScript demo</h1>
		<form name="form1">
			<input type="button" value="clickMe"
onclick="displayMessage()"/>
		</form>
	</body>
</html>
```

Note:

To work with JavaScript, we must enable JavaScript in your browser. (Enable debugging of JavaScript)

Note: -

JavaScript can be embedded into html doc in two ways

1) Internal JavaScript

In case of Internal JavaScript, we can embed JavaScript functions directly into the Html document using <script>... </script> section.

```
<script type="text/JavaScript">
        //logic
</script>
```

2) External JavaScript

In this case, create a JavaScript file and save with .js

And embed .js into the .html

In .js file, <script> tag is not required.

It includes only functions.

It is useful for embedding same script for multiple html documents.

I.e. we can create an external JavaScript file (.js) and embed into the required pages(.html)

Once, .js file is created, we can embed into any .html page using <script> tag.

Syntax: - [in html embedding .js file]

```
 <script src="filename.js" type="text/JavaScript">
</script>
```

Example:

j1.js

```
        function add(a,b)
        {
                return a+b;
        }
        function multifly(a,b)
        {
                return  a*b;
        }
```

HTML Page:

```
<!doctype html>
<html>
        <head>
                <title>External JavaScript</title>
                <script type="text/JavaScript" src="j1.js">
                </script>
```

```
<script type="text/JavaScript">
                function foo()
                {
                        var a=document.getElementById("t1");
                        var b=document.getElementById("t2");
                        var c=add(parseInt(a.value),parseInt(b.value));
                        document.getElementById("t3").value=c;
                }
            </script>
        </head>
<body>
        <form name="form1">
            value 1: <input type="text" id="t1" /> <br/>
            value 2: <input type="text" id="t2" /> <br/>
            Result: <input type="text" id="t3"/> <br/>
            <input type="button" value="clickMe" onclick="foo()"/>
        </form>
        </body>
</html>
```

Example

myscript.js

```
	function foo(a ,b)

	{

		var c=parseInt(a)+parseInt(b);

		return c;

	}

	function greet()

	{

		alert("welcome guest!");

	}
```

Html page

```
<!doctype html>

<html>

	<head>

			<title>External JavaScript</title>

			<script type="text/JavaScript" src="myscript.js">

			</script>

			<script type="text/JavaScript">

				function foo1()

				{

				var a=document.form1.t1.value;

				var b=document.form1.t2.value;
```

```
                    var result=foo(a,b);
            document.getElementById("s1").innerHTML="<b> The result is:"+result
+"</b>";
                    }
                </script>
    </head>
    <body onload="greet()">
        <form name="form1">
            value1 : <input type="text" name="t1" /> <br/>
            value2 : <input type="text" name="t2" /> <br/>
            <input type="button" value="clickMe" onclick="foo1()"/> <br/>
            <span id="s1"></span>
        </form>
    </body>
</html>
```

JavaScript basics

In JavaScript, we find the following:

object

method (function)

property (variable)

window is the base object in JavaScript

window has 3 derived types : document, location, history

document has many elemements

document : a,table,form, div, span and etc...

To access any of these elements we use (.) dot operator:

Example:

document.formname.fieldname.value

document.form1.t1.value

Methods

var str=new String("Cecity Software");

str.length --- it is property

str.toUpperCase()--- it is method

Example:

```
<!doctype html>

<html>

        <head>

                <title>dynamic content</title>

                <script type="text/JavaScript">

                        function foo()

                        {

                                var today=new Date();

                                document.getElementById("span1").innerHTML=today;

                        }

                </script>

        </head>

        <body>

                <form name="form1">

                        <h1> The current Date & Time : <span id="span1"></span></h1>

                <input type="button" value="Show Date" onclick="foo()"/>

                </form>

        </body>

</html>
```

Note:

innerHTML is used to add the html content on the web page dynamically into to any html tag.

Syntax: -

```
document.getElementById("someId").innerHTML="some content";
```

Variable declaration

Syntax: -

```
var varname;
```

Example:

```
var a=100;

var x='a';

var f=34.43;

var name="Cecity";

var today=new Date();

var users=new Array();
```

Note: -

We use **var** to declare any type of variables (strings, integers, decimals and etc...)

Scope of variables

- Global variables
- Local variables.

Global Variables

The variables declared outside the functions are global.

These scope & lifetime is till page is exited.

Example:

```
<script type="text/JavaScript">
        var x; //global
        function fun()
        {
                var a;
                a=10;  //local
                x=100; //global
                alert(x);
        }
        function foo()
        {
                alert(x);
                alert(a) ; //error
        }
</script>
```

Local variables

These are local to the specified function and destroyed once the function is exited.

Example:

```
<script type="text/JavaScript">
        var a; //global variable
        var b; //global variables
        function calculate()
        {
                a=10;
                b=20;
                var c=eval(a+b);   //local variable
                alert(c);
        }
        function foo()
        {
                a=100;
                b=200;
                c=a* b; // invalid : error
        }
</script>
```

Accessing form fields:

Accessing input from the user form:

Note:

In HTML form for every field, we can include **name** attribute. (name is applicable for only input fields)

So that, when we submit the form to the server and the content of the form is sent to the server in the form of name/value format. (Query string)

You can also include **ID** attribute to any html form field. (Id is for uniquely identifying each element)

This **ID** attribute is used in JavaScript DOM (document object model).

You can also use both (id as well as name).

Note:

Name attribute is applicable only for <form> and elements under <form>

Id attribute is applicable for any html tag.

Note:

For every element we must have unique id.

But there can be the same name for multiple form fields.

Name attribute is only for form fields.

We can't use name attribute for normal html tags.

Name attribute is for sending the form content to the server (name/value pair).

ID attribute is for manipulating html tags using JavaScript.

Syntax: -

document.formname.fieldname.value   [by using name]

OR

document.getElementById("fieldid").value [by using id]

document.getElementsByName("name");

document.getElementsByTagName("tagname");

Example:

```
<!doctype html>

<html>

        <head>

                <title>login form</title>

                <script type="text/JavaScript">

                        function validate()

                        {

                                var uid=document.form1.txtuser.value;

                                var psw=document.form1.txtpsw.value;

                        if(uid=="satya" && psw=="satya")

                                {

                                        alert("valid credentials");

                                        return true;

                                }

                                else

                                {

                                        alert("invalid credentials");

                                document.form1.txtuser.focus();

                                        return false;

                                }

                        }
```

```
		</script>
	</head>
	<body bgcolor="#b5b5b5">
		<form name="form1" id="form1">
			<h3>Please Login Here</h3>
			<p>user name : <input type="text" name="txtuser" /> </p>
			<p>Password : <input type="password" name="txtpsw"/></p>
			<input type="button" value="Login" onclick="validate()"/>
		</form>
	</body>
</html>
```

Comments in JavaScript

```
//singleline
/*multiline */
```

Note: -

Before browsing your html page with JavaScript, please enable scripting in your browser.

If the scripting is not enabled, the JavaScript will not run.

If browser does not support JavaScript, the script is displayed on the webpage as a plain text.

To avoid this problem, we comment the script using html special comment.

Syntax: -

```
        <!--
                script
        //-->
```

In this case, if browser supports the script, the script will be executed.

If not, the browser ignores the script.

Example:

```
<script type="text/JavaScript">
        <!--
        document.write("<h1>welcome JS</h>");
        document.write(Date());
        //-->
</script>
```

Operators in JavaScript

Arithmetic

+ , - ,*,/,%

Assignment

= ,+=,-=,*=,/=,%=

Logical Operators

&&,|| ,!

Conditional operator

?:

Increment & Decrement

++ ,--

Relational Operators OR Comparison operators

==,!=,< ,> ,<= ,>=

Control statements

if

if..else

if... else if ...else

Syntax:

```
if(condition)

{

        //execute the script, if the condition is true, otherwise ignores.

}


if(condition)

{

        //execute if the condition is true, otherwise skip it and execute else part

}

else

{

        //execute when the condition is false.

}
```

Similarly: if.. else if... else

```
if(condition1)

        {

                //block1

        }

        else if(condition2)

        {

                //block2

        }

        else if(....)

        {

        }

        else if(conditionn)

        {

                //block n

        }

        else

        {

                //default block

        }
```

Syntax: while

```
while(condition)

{

}
```

Syntax: do-while

```
do

{

        //logic

}while(test);
```

Syntax: for

```
for(;;)

{

        //logic

}
```

Syntax: for Switch

```
switch(expression)
{
        case choice1:
                block1
                break;
        case choice2:
                block2
                break;
        ...
        ...
        case choicen:
                blockn
                break;
        default:
                default block
}
```

Note:

The expression and choice should be of same type (either numeric or character type)

break:

It breaks the current block and comes out of that block.

continue:

It breaks the current iteration and continues the next iteration.

return

It exits the function

for...in

It iterates over each item of a collection without any test condition. (e.g. Array)

Syntax:-

```
for(index in arrayName)
{
    //script
}
```

Example:

```
var arr=new Array();
    arr[0]=100;
    arr[1]=200;
    arr[2]=300;
```

To get all these items we use for...in

```
for(index in arr)
{
        document.write(arr[index]);
}
```

Note: -

Index represents the index of each item.

It is a local variable.

Example:

```
var persons={id:"101",name:"satya",job:"manager"};
document.write(persons.id);
document.write(persons.name);
document.write(persons.job);
```

Example:

```
<!doctype html>
<html>
        <head>
                <title>for in demo</title>
        </head>
        <body>
                <script type="text/JavaScript">
                        var persons={id:"101",name:"satya",job:"manager"};
                        document.write(persons.id +"<br/>");
                        document.write(persons.name  +"<br/>");
                        document.write(persons.job  +"<br/>");
                </script>

        </body>

</html>
```

Syntax: for forEach()

```
        listName.forEach(function(entry){

                //logic

        });
```

Example:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script>

        function Init() {
            var employees = [{ empno: 101, ename: 'satya', salary: 1000 }, {
empno: 102, ename: 'sat', salary: 3000 }];
            var empList = "<table
border='1'><tr><th>EmpNo</th><th>Ename</th><th>Salary</th></tr>";

            employees.forEach(function (entry) {
                empList += "<tr><td>" + entry.empno + "</td><td>" + entry.ename +
"</td><td>" + entry.salary + "</td></tr>";
            });
            empList += "</table>";
            document.getElementById("listData").innerHTML = empList;
        }
    </script>
</head>
<body>
    <form name="form1">
        <input type="button" value="Click Me" onclick="Init()" />
        <span id="listData">

        </span>

    </form>

</body>
</html>
```

JavaScript popup functions (in-built window functions)

- alert()
- prompt()
- confirm()

alert()

It shows the given content in a popup dialog box.

Syntax:-

```
alert("message");
```

prompt()

It enables us to accept input through a dialogbox from user.

Syntax: -

```
var x=prompt("message","defautlvalue");

var name=prompt("enter name","wipro");
```

confirm()

It returns true or false based on the selected option on the confirm dialog box.

[Accept OR Reject]

Example:

```
var flag=confirm('Message');

if(flag)

        alert('true');

else

        alert('false');
```

Example:

```
<!doctype html>

<html>

        <head>

                <title>popup demo</title>

        </head>

        <body>

                <script type="text/JavaScript">

                        var name=prompt("enter name","satya");

                        var flag=confirm("Are you "+name +"?");

                        if(flag)

                                alert("welecome "+name);

                        else

                                alert("you are not authorized");


                </script>

        </body>

</html>
```

Working with user defined functions

Syntax: -

```
function functionName(arg1,arg2,...)
{
    //logic
    return value;
}
```

Note:

return is optional .

Example:

```
<script type="text/JavaScript">
    function sum(a,b)
    {
        var s;
        s=a+b;
        return s;
    }
        var s=sum(2,3);
        var ss=sum("Cecity","Inc");
</script>
```

Note:

We use + for addition of numeric values.

We use + for concatenation of two strings.

Note: -

We can invoke function directly in <script> section.

We can also invoke a function using certain events of the form elements.

e.g.

button click

mousemove

keypress

etc...

HTML DOM Events

HTML DOM events allow JavaScript to register different event handlers on elements in an HTML document.

Events are normally used in combination with functions, and the function will not be executed before the event occurs (such as when a user clicks a button).

onclick, onblur, onfocus, onmouseover, onmouseout, ondblclick

Note: -

We use, + for concatenation of two strings as well as addition of two numbers.

It is performed based on the type of data.

If at all, you want to convert string to integer or string to float, we use the following functions.

parseInt(); : string to integer

parseFloat(); : string to float

Example:

```
var x="100";

var y="200";

var z=x+y ;  // output : 100200  , not 300

var result=parseInt(x)+parseInt(y); // output : 300
```

Creating Custom Objects in JavaScript:

Methods and properties of string object

charAt() : returns the character at given index.

indexOf() : returns the index value of given character. (first occurence)

length : returns the length of given string. (no of characters including white space)

split() : splits the given string based on given deliminator and returns array of strings

substring() : returns a string from another string fron start index to last index.

substr() : returns a string from another string from startindex and specified number of charactrs.

Example:.

```
var str="Cecity";
        str.charAt(0);
var str="helpdesk@Cecity.com";
        str.indexOf("@");
var str="HTML programing";
        str.length;
var emps="111:satya:1000";
                var emp=emps.split(":");
                emp[0] : 111
                emp[1] : satya
                emp[2] : 1000
var str="Cecity software";
var s=str.substring(8,15);  //returns  :software
```

var s1=str.substr(0,7) ; //returns : Cecity

substring(firstindex,lastindex);

substr(firstindex,offset);

offset : it specifies the number of characters to return

var arr=str.split(" ") ; // it splits the given string based on deliminator and returns an array

arr[0] : wipro

arr[1] : tech

e..g.

var emp="111:satya:1000000";

var emdetails=emp.split(":");

empdetails[0] : 111

empdetails[1] : satya

empdetails[2] : 1000000

var ss=str.subString(0,2); // it returns the sub string from start index and specifed number of characters.

isNaN(str) : it checks ,whether given data is numeric or not .

It returns true, if not a number.

It returns false,if it is number

Example:

```
 var mob="12345";
if(mob.length==10)
{
        if(isNaN(mob)==false)
        {
                alert("it's number");
        }
}
```

charAt(position)

concat(v1, v2,..)

indexOf(char/substring)

slice(start, end)   : for substring

split(delimiter) : returns array of sub strings .

subString(from, to) : sub string from from to end

HTML Form Validation:

onsubmit : it is an event handler for <form> element.

It allows us to perform the form validation.

If the form is valid, then it allows us to submit the page.

It the form not valid, It stops submitting the page and give's error message.

Steps to perform form validation:

Step 1: define all the required validation functions

Step 2: define a common function (form validation)and call all the validations , if all returns true ,then return true in the common function. otherwise return false.

Step 3: bind the common validation function to the onsubmit event handler of the <form> element.

Syntax: -

```
<form name="formname" onsubmit="return commonfunction()">
</form>
```

Example:

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

    <title></title>

    <script>

        function fnRequired(arg) {

            var val = arg.value;

            if (val == "" || val == null) {

                alert("field is missing");

                arg.focus();

                return false;

            }

            else {

                return true;

            }

        }


        function fnValidateForm() {

            var user = document.getElementById("txtusername");

            var password = document.getElementById("txtpassword");
            if (fnRequired(user) && fnRequired(password)) {
                return true;
            }
            else {
                return false;
            }
        }
    </script>
</head>
<body>
        <form name="loginform" onsubmit="return fnValidateForm()">
            <h3>Please Login here...</h3>
            <p>
                Username : <input type="text" id="txtusername"  />
            </p>
            <p>
               Password: <input type="password" id="txtpassword"  />
            </p>
            <p>
                <input type="submit" value="submit" />
            </p>
        </form>
```

```
</body>

</html>
```

Assignment:

1) Validate the user form

   User name must require (text field must not empty)

   Password must require

   Confirm password must match with password

   Email should be in valid format (dotnetsatya@gmail.com)

   Mobile should have only 10 digits (only numeric)

   Age should be between 20 and 25

   Zip code should be 6 digits

Event Handlers

- onclick
  - button
- onchange,onfocus,onblur
  - textbox
- onload,onunload
  - body
- onsubmit ,onreset
  - form

onmousemove

onmousedown

onmouseover

onmouseout

onmouseup

onkeyup

onkeydown

onkeypress

onfocus [textbox] : while entering into textbox

onblur   [textbox] : while leaving the textbox

onchange [textbox] : while chaning the textbox content.

Syntax: for onsubmit

```
<form name="form1" onSubmit="return functionName()">
    <input type="submit" value="register" />
</form>
```

Note: -

If return value is true, the page is posted to server.

If return value is false, the page will not posted to server. Means there are some validations in page.

Example: for onload

```
<script type="text/JavaScript">
    function greet()
    {
        alert("tea break");
    }
</script>
<body onload="greet()">
    <form name="form">
</form>
</body>
```

```
document.bgColor="";  // change background color dynamically
document.fgColor="";  //change foreground color dynamically .
```

Example:

```
        function foo()
                {
                        document.bgColor="#b4b4b4";
                        document.fgColor="red";
                }
<a href="" onmouseover="foo()" onmouseout="foo1()">change color</a>
```

Example:

This example for checking, whether there exist a digit in the given string.

```
fuction fun() {
        var str = new String();
        str = "wiprog1mail.com";
        var i;
        var flag = 0;
        for (i = 0; i < str.length; i++) {
            if (str.charAt(i)>= '0' && str.charAt(i) <= '9') {
                flag = 1;
                break;
            }
            else{
                flag = 0;
            }
        }
```

```
            if (flag == 1) {
               alert("digit found");
            }
            else {
               alert("not found");
            }
      }
```

Dynamically loading image into the page

Syntax:

```
            document.images["imageId"].src="url";
                        or
            document.getElementById("imageId").src="url";


document.images["img1"].src="logo.gif";
            Or
document.getElementById("img1").src="logo.gif";
<img id="img1" src=" " />
```

Example:

```
<!doctype html>
<html>
        <head>
                        <title>Image rollover</title>
                <script type="text/JavaScript">
                        function foo1()
                        {
                                document.getElementById("image1").src="cam1.png";
                        }
                        function foo2()
                        {
document.getElementById("image1").src="plain.jpg";
                        }
                </script>
        </head>
        <body>
                <form name="form1">
                        <img src="plain.jpg" id="image1" alt="plain"
onmouseover="foo1()" onmouseout="foo2()"/>
                </form>         </body></html>
```

Assignments:

Create a User registration form and perform the validation

user id

password

confirm password

email

mobile

Submit (perform validations on submit button)

All fields are required

User id should be more than 8 char

Password should contain at least one digit

Confirm password should be same as password

Email should be correct format

Mobile should have Only 10 digits

Help:

```
function fun() {
        var x = document.f1.t1.value;
        if (isNaN(x)==false) {
           alert("number");
        }
        else {
           alert("not number");
        }
      }
var uid="satya";
        if(uid.length<8)
                alert("invalid");
var mobile="324543";
        if(mobile.length==10)
        {
                if(isNaN(mobile)==true)
                        alert("not valid");
                else
                alert("valid");
        }
        else
        alert("invalid");
```

Dynamically Applying Styles using JavaScript

We use CSS to apply styles to any element in HTML.

CSS is a static.

CSS is applied during page load.

e.g.

```
<head>
        <title>css </title>
        <style>
                p
                {
                        background-color:silver;
                        color:olive;
                        font-family:verdana;


                }
        </style>
</head>
<body>
        <p>Welcome Cecity</p>
</body>
```

Instead, we need to apply styles to an element at runtime.

By using event handlers, we need to apply the styles to the page.

e.g.

Whenever i do mouseover on image, the image should grove.

When i do mouse out, it should reset the image.

When i do mouseover on paragraph, font should increase and color should change.

For this purpose, we JavaScript DOM with style object.

Syntax: -

```
document.getElementById("someId").style.property=value;
```

Example:

Take one Image; change the Position of image from Left to right by using JavaScript.

To change the position of an element, we use top and left properties.

To make the element as movable, you must apply position of that element as absolute.

In CSS, we apply position of an element in two ways

1) Relative position

In case of relative position, we can't move the element at runtime

It is for inline flow of an element.

2) Absolute position.

In this case, we can give whatever position we want.

Here, we can also move the element dynamically.

Syntax: -

```
<element style="position:absolute">
</element>
```

e.g.

```
<p style="position:absolute;top:50px;left:100px">
        paragraph
</p>
```

Note:

The default position is relative. (Inline flow)

javascript:void(0)

Tell me, when i click anchor (link), what will happen?

It will navigate to the specified URL.

OR

It will refresh the same page.

Instead of Refreshing the page or navigating other URL, i need to call some JavaScript function.

For this purpose, we use void (0) function on the link.

Syntax: -

```
<a href="javascript:void(0)">click me</a>
```

Example:

Take a paragraph or any element

Take two anchors (links)

One for Show element

Other for Hide element.

```
<!doctype html>

<html>

        <head>

                <title>style</title>

                <style>

                        #p1

                        {

                                color:olive;

                                background-color:silver;

                        }

                </style>


                <script>

                        function show()

                        {

document.getElementById("p1").style.display="block";

                        }


                        function hide()

                        {

document.getElementById("p1").style.display="none";

                        }
                </script>
        </head>
        <body>
        <a id="a1" href="javascript:void(0)" onclick="show()">Show</a>


&nbsp;&nbsp;&nbsp;
        <a href="javascript:void(0)" id="a2" onclick="hide()">Hide</a>


<br/>
                <p id="p1">Welcome Cecity</p>
        </body>
</html>
```

Working with Date () in JavaScript

Syntax: -

```
      var d=new Date();   // Date() object

            d.getDay();

            d.getMonth();

            d.getYear();

            d.getMinutes();

            d.getHours();

            d.getSeconds()

            d.setDate(year,month,day);
```

Example:

```
      var d=new Date();

            d.setDate(2012,06,25);


 function displayDate() {

      var today = new Date();

      alert(today.toString());

      alert(today.toDateString());

      alert(today.toGMTString());

      alert(today.toLocaleDateString());

      alert(today.toLocaleTimeString());

   }
```

Delay functions

- setTimeout()
- clearTimeout()

Note: -

In general, the functions are invoked immediately after the function call Or Once event fires.

Instead, we need to call after some time delay.

For this purpose, we use setTimeout() function.

setTimeout()

It invokes a function after certain time delay.

It is used to set the time delay.

clearTimeout()

It is used to stop the delay applied by setTimeout() function.

It is used to clear the time delay.

Syntax: -

Variable=setTimeout('functioname()',timeinmillisec);

clearTimeout(variable);

Example:

```
<html>

        <head>

                <title>mouse events</title>

        <script type="text/JavaScript">

                function fun1()

                {

                setTimeout("foo()",3000);

                }

                function foo()

                {

                document.bgColor="cyan";

                alert("page loaded after 3 secs");

                }

        </script>

        </head>

        <body onload="fun1()">

                <h1> setTimeout </h1>

        </body>


</html>
```

Example for Slide show of images

Example on Image Show

Assume that there are some images with names: l1.jpg, l2.jpg,…..

Now, we need to show images one by one with some time delay on button click

For this, we use setTimeout() and clearTimeout()

```
<!doctype html>

<html>

        <head>

                <title>image slide show</title>

                <script type="text/JavaScript">

                        var t;

                        var i=1;

                                function startShow()

                                {

                                document.getElementById("image1").src="l"+i+".jpg";

                                i++;

                                t=setTimeout("startShow()",2000);

                                }
```

```
            function stopShow()
                        {
                                    clearTimeout(t);
                                    i=1;
                        }
            </script>
        </head>
        <body>
            <form name="form1">
                <img id="image1" src="l1.jpg" height="300px" width="400px"/>
<br/>
        <input type="button" value="Start" onclick="startShow()"/>
<br/>
        <input type="button" value="Stop" onclick="stopShow()"/>
            </form>
        </body></html>
```

Note:

In the above examle, we are using setTimeout() to call the same function again and again with some delay. (Recursive)

Location object

It is used to dynamically load a new page into the browser.

It used to reload the current page with other page.

It is also used to get URL information of the current page.

Syntax: -

window.location="url":

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

enable/disable an element

Here, if the element is enabled, then we can do some action on that element.

If the element is disabled, the element is presented on UI, but no action is performed on that.

Syntax:

document.formname.fieldname.disabled=true /false

document.getElementById("id").disabled=true/false;

Example:

Take a form, with textbox for username and submit button.

Here, when text in textbox is empty, disable the submit button. Otherwise enable it.

```
<!doctype html>

<html>

        <head>

                <title>enabled demo</title>

                <script type="text/JavaScript">

                        function foo()

                        {

                        if(document.form1.txtuser.value=="" ||

document.form1.txtuser.value==null)

                        {


                        document.form1.btnsubmit.disabled=true;

                        }

                        else

                        {

                        document.form1.btnsubmit.disabled=false;

                        }

                        }

                </script>

        </head>
```

```
<body>

                <form name="form1">

                        User name : <input type="text" name="txtuser" onblur="foo()" />

<br/>

                        <input type="button" name="btnsubmit" value="submit" />

                </form>

        </body>

</html>
```

Working with select element (dropdownlist)

Syntax: for dropdown list

For single selection

```
<select name="ddlgender" id="list">

        <option value="" selected="selected">text</option>

        <option value="">text</option>

</select>
```

For multiple selection (listbox)

```
<select name="list" id="list" multiple>
        <option value="">text</option>
        ..........
</select>
```

document.formname.listname.options[index].text OR value

Or use DOM

```
var list=documement.getElementById("list");
```

In case of Single Selection:

```
                list.options[list.selectedIndex].text/value;
```

In case of multiple Selections:

```
        var list=document.getElementById("list");
        for(i=0;i<list.length;i++)
        {
                if(list[i].selected)
                {       list[i].text/value;
                }
        }
```

Make list empty

```
document.formName.listName.options.length=1 [ except the very first one ]
```

Remove the selected item

```
document.formname.listname.options[index]=null

document.formname.listname.options[2]=null
```

To specify the size of list:

```
<select name="list"  size="4" multiple="multiple">

        <option value="value" selected>text</option>

</select>


var as = document.form1.list.value;   or text

var e = document.getElementById("list");

  var strUser = e.options[e.selectedIndex].value;

 var strUser=e.options[e.selectedIndex].text
```

Example:

```
<!doctype html>

<html>

        <head>

                <title>dropddown list</title>

                <script type="text/JavaScript">

                        function foo()

                        {

        document.getElementById("s1").innerHTML="Your Gender
is:"+document.form1.gender.options[document.form1.gender.options.selectedIndex].text;

                        }

                </script>

        </head>

        <body>

                <form name="form1">

                        Select Gender : <select name="gender">

                        <option value="0" selected>Choose Gender</option>

                                        <option value="1">Male</option>

                                        <option value="2">Female</option>

                        </select>

                        <br/>

                <span id="s1"></span>

                        <input type="button" value="clickMe" onclick="foo()"/>
                </form>
        </body>
</html>
```

Example:

```
<!doctype html>
<html>
        <head>
                <title>dropddown list</title>
                <script type="text/JavaScript">
                        function foo()
                        {
                        var l=document.getElementById("list");
                        var str="";
                        for(var i=0;i<l.length;i++)
                        {
                                if(l[i].selected)
                                {
                                        str+="<li> "+l[i].text+"</li> ";
                                }
                        }
                document.getElementById("s1").innerHTML=str;
                        }
```

```
		</script>
	</head>
	<body>
		<form name="form1">
		Select  your Interests:
		<select id="list" multiple="multiple" size="5" onchange="foo()">
			<option value="1" selected>Movies</option>
			<option value="2">Sports</option>
			<option value="3">Drawing</option>
			<option value="4">Browsing</option>
			<option value="5">Reading</option>
			<option value="6">fighting</option>
		</select>
	<span id="s1"></span>
		</form>
	</body>
</html>
```

Assignments:

- Validate the list (<select> tag), whether option is selected or not
- Apply the background color of the page, based on the selected color in the list.

```
document.bgColor="colorname";

document.getElementById("div1").style.backgroundColor="colorname";
```

- Display the date of birth based on selection in <select> tag.

  [Hint: for day: <select>, for month :< select>, for year: <select>]

- Show the list of Cities based on the state value. (two list, one of state, other for city)

Working with checkbox

It enables us to check multiple items.

It provides check/uncheck.

It provides checked property to check the status of checkbox.

Syntax: -

```
<input type="checkbox" id="check1" name="check1" value="xxx"/>
```

```
document.getElementById("check1").checked=true

document.getElementById("check1").value
```

Note:

To group multiple checkboxes, give the same to all the check boxes.

For multiple: give the same name

```
var list=document.getElementsByName("checkboxname");

for(i=0;i<list.length;i++)

{

        if(list[i].checked)

                {

                        alert(list[i].value);

                }

}
```

Assignment:

Create a web page to select multiple Hobbies OR interests, Display the selected Options down in the page.

Working with radio button

It enables us to select one of the multiple options.

Syntax:

```
<input type="radio" id="r1" name="r1"/>

document.getElementByID("radio1").checked=true
```

Example:

```
<!doctype html>

<html>

        <head>

                <title>dropddown list</title>

                <script type="text/JavaScript">

                        function foo()

                        {

                        if(document.getElementById("others").checked==true)

                        {

                        document.getElementById("div1").style.display="block";

                        }
                        else
                        {
                        document.getElementById("div1").style.display="none";
                        }
                        }

function foo1()
        {
                var l;
                if(document.getElementById("others").checked==true)
                {
                        l=document.form1.txtlangauge.value;
                }
                else
                {
        l=document.form1.language.options[document.form1.language.options.selectedIn
dex].text;
                }
                document.getElementById("s1").innerHTML="<b> Your Selected
language:"+l+"</b>";
        }
                </script>
        </head>
        <body>

                <form name="form1">

                Choose Your Spoken Language : <select name="language">

                                        <option value="0" selected>Choose
language</option>

                                        <option value="1">English</option>

                                        <option value="2">Hindi</option>

                                        </select>

                        <br/>

                        <input type="checkbox" id="others" value="Others"
onclick="foo()"/> Others

        <br/>


<div id="div1" style="display:none">

                Enter Your Language: <input type="text" name="txtlangauge"/>

                        </div>

                <input type="button" value="clickMe" onclick="foo1()"/>

                <span id="s1"></span>

                </form>

        </body>

</html>
```

Assignment:

- Select your gender by using radio Buttons

Window object

It is a base object in JavaScript DOM.

It refers to the Current window.

It provides various derived objects

window

- location
- document
- history

It also provides various methods to open() new window and close that window.

window.open()

It enables us to create new window dynamically and load some page into that.

Syntax: -

window.open("URL", "name", "attributes")

URL: it represents, which page should be displayed in the new window.

The "name" parameter lets you assign a target name to the newly opened window for links on your page to target.

The 3rd parameter, "attributes", is where you control the look of the opened window

Attributes	Description
channelmode	Specifies if window should be opened in channel mode. IE only.
fullscreen	Specifies if window should be opened in full screen mode. IE only.
height	Specifies the height of the window.
left	Specifies the x coordinates of the window in pixels. IE only.See "screenX" as well.
location	Specifies if the location bar of the window should be included.
menubar	Specifies if the menu bar of the window should be included.
resizable	Specifies if window should be resizable.
screenX	Specifies the x coordinates of the window in pixels. NS only. See "left" as well.

screenY Specifies the x coordinates of the window in pixels.NS only. See "top" as well.

scrollbars Specifies if window should contain scrollbars.

status Specifies if the status bar of the window should be included.

toolbar Specifies if the toolbar of the window (ie: reload button) should be included.

top Specifies the y coordinates of the window in pixels. IE only. See "screenY" as well.

width Specifies the width of the window.

Example:

window.open("page2.htm", "win1","width=300,height=200,menubar")

window.open('page2.htm', "win1",'width=600,height=500,status')

window.open('page2.htm', "win1",'width=600,height=500,status,resizable')

Example:

```
<!doctype html>

<html>

        <head>

                <title>popups demo</title>

        <script type="text/JavaScript">

                function foo()

                {

                window.open("http://www.google.com",

"win1","width=300,height=200,menubar");

                }

        </script>

        </head>

        <body>

                <form name="form1">

                        <input type="button" value="open popup" onclick="foo()"/>

                        <br/>

                        <input type="button" value="close"

onclick="JavaScript:window.close()"/>

                </form>

        </body>

</html>
```

Note: -

If property is specified, that is applied to that window. if not ,it does not apply .

window.close();

It enables us to close the current window.

Note:

We can also close newly opened window with the reference variable.

Syntax: -

```
var newwindow=window.open("url","name","attributes");

newwindow.close();
```

Example:

```
<!doctype html>

<html>

        <head>

                <title>popups demo</title>

        <script type="text/JavaScript">

                var w;

                function foo()

                {

                w=window.open("http://www.google.com",

"win1","width=300,height=200,menubar");

                }
                function foo1()
                {
                w.focus();
                w.close();
                }
        </script>
        </head>
        <body>
                <form name="form1">
                        <input type="button" value="open popup" onclick="foo()"/>
                        <br/>
                        <input type="button" value="close" onclick="foo1()"/>
                </form>
        </body>
</html>
```

Adding dynamic content to the newly opened window

Generally, we use window.open() to create a new window and open some page into that window.

Instead, we can also use window.open() for creating an empty page and we can add some content dynamically to that page.

Syntax: -

```
var mywindow=window.open("","name","attributes");

mywindow.document.write("some text");

mywindow.document.bgColor="cyan";

mywindow.document.fgColor="red";
```

Example:

```
<!doctype html>

<html>

        <head>

                <title>popups demo</title>

        <script type="text/JavaScript">

                function foo()

                {

                        var

mywin=window.open("","win1","height=300px,width=300px");
```

```
                              mywin.document.write("<html><head><title>new
win</title></head><body bgcolor='cyan'><h1>Newly Opened window</h1>");
                              mywin.document.write("<form name='form1'>");
                              mywin.document.write("user name:<input type='text'/><br/> ");
                              mywin.document.write("<input type='button' value='clickme'/>");
                              mywin.document.write("</form>");
                              mywin.document.write("</body></html>");
                              mywin.focus();
                    }
          </script>
          </head>
          <body>
                    <form name="form1">
                              <input type="button" value="open popup" onclick="foo()"/> <br/>
                    </form>
          </body>
</html>
```

Note:

We can also add dynamic content to the opener window.

mywindow.opener.document.write("<p>This is the source window!</p>");

It enables us to add some content to the Source window (parent)

```
          mywindow.moveBy(250,250);
```

```
mywindow.moveTo(0,0);

window.print();

window.resizeTo(500,500);

window.scrollBy(100,100);

window.scrollTo(100,500);
```

Screen object

```
screen.height

screen.width

screen.availHeight

screen.availWidth

screen.colorDepth

screen.pixelDepth
```

history object

```
history.length

window.history.back()

window.history.forward()

window.history.go(-2)  : two pages back

window.history.go(2)  : two pages forward
```

location object

Gets/sets the URL of the current page

Location contains information about the current URL of the browser.

The most common usage of Location is simply to use it to automatically navigate the user to another page:

```
<script type="text/JavaScript">
window.location="http://www.google.com"
</script>
```

window.location.host : server name

window.location.href : complete url

window.location.pathname : directory name

window.location.protocol : channel (http)

window.location.assign("http://www.w3schools.com") : loads new document.

window.location.reload()

window.location.replace("http://www.w3schools.com")

window.top.location="somepage.htm";

navigator object

Use the navigator object to detect client's browser

navigator.appCodeName

navigator.appName

navigator.appVersion

navigator.cookieEnabled

navigator.language : only NS and firefox only

mimeTypes[] : only NS and firefox only

Timing Interval functions

Note:

setTimeout()

clearTimeout()

These are used to apply some time delay. So that the function will be invoked after the given delay time.

But the function is invoked only once.

To invoke the same function again and again, we use recursive method. Instead of this, we use setInterval() to cal a function again and again with some time delay (no recursive method required).

Timing Functions

- setInterval()
- clearInterval()

These functions are used to invoke a function again and again, with time delay.

```
var v=setInterval("clock()",1000);
```

Here, clock () is called again and again for every 1000 milliseconds

clearInterval()

It is used to stop the function call of the setInterval() function.

Example:

```
function clock()

{

var d=new Date();

var t=d.toLocaleTimeString();

document.getElementById("clock").value=t;

}

clearInterval(v)
```

Example:

```
<!doctype html>
<html>
	<head>
		<title>timer demo</title>
		<script type="text/JavaScript">
			var v;
			function clock()
			{
				var d=new Date();
				var t=d.toLocaleTimeString();
				document.getElementById("p1").innerHTML=t;
			}
			function start()
			{
				v=setInterval("clock()",1000);
			}
			function stop()
			{
				clearInterval(v);
			}
		</script>
```

```
</head>

<body>

        <form name="form1">

                <p id="p1"></p>

        <input type="button" value="start Timer" onclick="start()" /> <br/>

        <input type="button" value="stop timer" onclick="stop()"/>

        </form>

</body>         </html>
```

JavaScript Cookies (client side)

A cookie is a small text file that is stored in your browser (client machine). It contains some data.

It allows only small amount of data (in kb).

It is used to store frequently used data on the page, so that we can access it in the subsequent requests.

We use cookies for state management technique. It is a client side state management.

Once a cookie is created, we can access it any where in the web site (in the same page or another page)

It is an application level.

We can also use cookies for security in session management.

As soon as you login, the user info is stored in the cookie and that cookie is sent to the server.

The server can read the user info from the cookie and server will indentify the user.

The cookies are used implicitly in ASP.NET Authentication mechanism.

It consists of the following parts (for each cookie)

- A name-value pair containing the actual data
- An expiry date after which it is no longer valid
- The domain and path of the server it should be sent to

Note: -

You can add multiple cookies to the browser.

Each cookie has unique name/value pair.

To work with cookie, you need to enable the cookies in browser.

As soon as you request a page from a server to which a cookie should be sent, the cookie is added to the HTTP header. Server side programs can then read out the information and decide that you have the right to view the page you requested or that you want your links to be yellow on a green background.

We use cookies to determine, whether user is authenticated or anonymous.
Based on the cookie value, we indentify the user and grant the permission to access the resources.

name-value

Each cookie has a name-value pair that contains the actual information. The name of the cookie is for your benefit, you will search for this name when reading out the cookie information

Expiry date

Each cookie has an expiry date after which it is trashed. If you don't specify the expiry date the cookie is trashed when you close the browser. This expiry date should be in UTC (Greenwich) time.

Domain and path

Each cookie also has a domain and a path. The domain tells the browser to which domain the cookie should be sent. If you don't specify it, it becomes the domain of the page that sets the cookie

The path gives you the chance to specify a directory where the cookie is active

You can specify (/) to use default domain and path

Creating cookie and reading cookie

document.cookie

Cookies can be created, read and erased by JavaScript. They are accessible through the property document.cookie

Example:

document.cookie =

  'key1=Cecity; expires=sat, 12 Jan 2013 20:47:11 UTC; path=/'

Description:

First the name-value pair ('key1=Cecity')

Then a semicolon and a space

Then the expiry date in the correct format ('expires=Thu, 27 June 2012 20:47:11 UTC')

Again a semicolon and a space

Then the path (path=/)

Example for cookie

```
<!doctype html>

<html>

        <head>

                <title>cookie demo</title>

                <script type="text/JavaScript">

                        function setCookie()

                        {

                                var user=document.form1.txtname.value;

                                var name="c1="+user;

                                var edate="sat, 4 Aug 2012 20:47:11 UTC"

                        document.cookie=name+"; expires="+edate+"; path=/";

                        }

                        function getcookie()

                        {

                                var str=document.cookie;

                                var name=str.split("=");

                                if(name[1]!=null && name[1]!="")

                                {

                        document.getElementById("s1").innerHTML=name[1];

                                }

                                else

                                {
```

```
document.getElementById("s1").innerHTML="Guest!";
                                        }
                              }
                    </script>
          </head>
          <body onload="getcookie()">
                    <form name="form1">
                              Welcome :<span id="s1"></span> <br/>
          enter name: <input type="text" name="txtname" /> <br/>
                    <input type="button" value="set cookie" onclick="setCookie()"/> <br/>
                    <input type="button" value="get cookie" onclick="getcookie()"/>
                    </form>
          </body>
</html>
```

HTML5 Web Storage

With HTML5, web pages can store data locally within the user's browser.

Earlier, this was done with cookies. However, Web Storage is more secure and faster. The data is not included with every server request, but used ONLY when asked for. It is also possible to store large amounts of data, without affecting the website's performance.

Overcome following drawbacks:

- Cookies are included with every HTTP request, thereby slowing down your web application by transmitting the same data.
- Cookies are included with every HTTP request, thereby sending data unencrypted over the internet.
- Cookies are limited to about 4 KB of data . Not enough to store required data.

The data is stored in name/value pairs, and a web page can only access data stored by itself.

Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.

Note: Web storage is supported in Internet Explorer 8+, Firefox, Opera, Chrome, and Safari.

HTML5 Web Storage Objects

HTML5 Web Storage provides two new objects for storing data on the client:

- localStorage - stores data with no expiration date
- sessionStorage - stores data for one session (data is lost when the tab is closed)

The localStorage Object

The localStorage object stores the data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

```
//Store
localStorage.lastname = "Smith";
// Retrieve
document.getElementById("result").innerHTML=localStorage.lastname
;
```

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script>

        function Init() {
            var name = document.getElementById("txtName").value;
            this.localStorage.FirstName = name;
            if (this.localStorage.FirstName) {
                document.getElementById("p1").innerHTML = "Hello.." +
this.localStorage.FirstName;
            }
        }
        function foo() {
            if (this.localStorage.FirstName) {
                document.getElementById("p1").innerHTML = "Hello.." +
this.localStorage.FirstName;
            }
            else {
                document.getElementById("p1").innerHTML = "Hello..Guest!";
            }
        }
    </script>
</head>
<body onload="foo()">
    <form name="form1"  >
        <p id="p1"></p>
       Name : <input type="text" id="txtName" /><br />
        <input type="submit" value="submit" onclick="Init()" />        <br />

    </form>

</body>
</html>
```

The syntax for removing the "lastname" localStorage item is as follows:

```
localStorage.removeItem("lastname");
```

The sessionStorage Object

The sessionStorage object is equal to the localStorage object, **except** that it stores the data for only one session. The data is deleted when the user closes the browser window.

Exception Handling in JavaScript

It is used to handle the runtime errors.

If at all, we get any runtime error, the browser will show debug error. it attempts to stop the execution of the page.
To avoid this problem, we use exception handling technique. (try..catch())

In this case, if there exists any run time error, we can handle it and show the error message.

We can continue the execution, instead of aborting.

Using try. Catch block

try. Catch block in JavaScript is very much similar to the regular C# or Java try..catch block. The suspected code will be kept in try block and all exceptions which will occur in the try block will be caught in catch block.

```
try

{

        //code expected to be thrown exceptions

}

catch(err)

{

        alert(err.name+" "+ err.message);

}
```