UNIT-V
JAVASCRIPT

JavaScript is *an object-based scripting language* that is lightweight and cross-platform. JavaScript is not compiled but translated. The JavaScript Translator (embedded in browser) is responsible to translate the JavaScript code.

JavaScript is used to create interactive websites. It is mainly used for:

- Client-side validation
- Dynamic drop-down menus
- Displaying data and time
- Displaying popup windows and dialog boxes (like alert dialog box, confirm dialog box and prompt dialog box)
- Displaying clocks etc.

Simple example of javascript:

```
<html>
 <body>
    <script language="javascript" type="text/javascript">
      document.write("Hello World!")
   </script>
 </body>
</html>
```

The **script** tag specifies that we are using JavaScript.
The **text/javascript** is the content type that provides information to the browser about the data.
The **document.write()** function is used to display dynamic content through JavaScript.

JavaScript can be written in three places

- within body tag
- within head tag and
- external JavaScript file.

Within body Tag:

```
<html>
<body>
<script type="text/javascript">
alert("Hello Javatpoint");
</script>
</body>
</html>
```

Within head tag:

```
<html>
<head>
<script type="text/javascript">
function msg(){
alert("Hello Javatpoint");
}
</script>
</head>
<body>
<p>Welcome to JavaScript</p>
```

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

External JavaScript file"

We can create external JavaScript file and embed it in many html page. It provides **code re usability** because single JavaScript file can be used in several html pages.An external JavaScript file must be saved by .js extension.

Example:

message.js

```
function msg(){
 alert("Hello Javatpoint");
}
```

index.html

```
<html>
<head>
<script type="text/javascript" src="message.js"></script>
</head>
<body>
<p>Welcome to JavaScript</p>
<form>
<input type="button" value="click" onclick="msg()"/>
</form>
</body>
</html>
```

JavaScript Comment

The **JavaScript comments** are meaningful way to deliver message. It is used to add information about the code, warnings or suggestions so that end user can easily interpret the code. The JavaScript comment is ignored by the JavaScript engine i.e. embedded in the browser.

Ex:

```
<script>
// It is single line comment
document.write("hello javascript");
/* It is multi line comment.
It will not be displayed */
document.write("example of javascript multiline comment");
</script>
```

JavaScript Variable

A **JavaScript variable** is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable.

There are some rules while declaring a JavaScript variable (also known as identifiers).

- Name must start with a letter (a to z or A to Z), underscore(_), or dollar($) sign.
- After first letter we can use digits (0 to 9), for example value1.
- JavaScript variables are case sensitive, for example x and X are different variables.

Example of JavaScript variable

```
<script>
var x = 10;
var y = 20;
var z=x+y;
document.write(z);
</script>
```

JavaScript local variable

A JavaScript local variable is declared inside block or function. It is accessible within the function or block only. For example:

```
<script>
function abc(){
var x=10;//local variable
}
</script>
```

JavaScript global variable

A **JavaScript global variable** is accessible from any function. A variable i.e. declared outside the function or declared with window object is known as global variable. For example:

```
1.  <script>
2.  var value=50;//global variable
3.  function a(){
4.  alert(value);
5.  }
6.  function b(){
7.  alert(value);
8.  }
9.  </script>
```

Javascript Data Types:

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

JavaScript is a **dynamic type language**, means you don't need to specify type of the variable because it is dynamically used by JavaScript engine. You need to use **var** here to specify the data type. It can hold any type of values such as numbers, strings etc. For example:

```
1.  var a=40;//holding number
2.  var b="Rahul";//holding string
```

JavaScript primitive data types

There are five types of primitive data types in JavaScript. They are as follows:

Data Type	Description
String	represents sequence of characters e.g. "hello"
Number	represents numeric values e.g. 100
Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

JavaScript non-primitive data types

The non-primitive data types are as follows:

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression

JavaScript Conditional Statements:

1. If Statement
2. If else statement
3. if else if statement

JavaScript If statement

It evaluates the content only if expression is true.

Syntax:

```
if(expression){
//content to be evaluated
}
```

Ex:

```
<script>
var a=20;
if(a>10){
document.write("value of a is greater than 10");
}
</script>
```

JavaScript If...else Statement

It evaluates the content whether condition is true of false.

Syntax :

```
if(expression){
//content to be evaluated if condition is true
}
else{
//content to be evaluated if condition is false
}
```

Ex:

```
<script>
var a=20;
if(a%2==0){
document.write("a is even number");
}
else{
document.write("a is odd number");
}
</script>
```

JavaScript If...else if statement

It evaluates the content only if expression is true from several expressions.

Syntax:

```
if(expression1){
//content to be evaluated if expression1 is true
}
else if(expression2){
```

```
//content to be evaluated if expression2 is true
}
else if(expression3){
//content to be evaluated if expression3 is true
}
else{
//content to be evaluated if no expression is true
}
```

Ex:

```
<script>
var a=20;
if(a==10){
document.write("a is equal to 10");
}
else if(a==15){
document.write("a is equal to 15");
}
else if(a==20){
document.write("a is equal to 20");
}
else{
document.write("a is not equal to 10, 15 or 20");
}
</script>
```

JavaScript Switch

The **JavaScript switch statement** is used *to execute one code from multiple expressions*.

Synatax:

```
switch(expression){
case value1:
 code to be executed;
 break;
case value2:
 code to be executed;
 break;
.....

default:
 code to be executed if above values are not matched;
}
```

Example of switch statement in javascript:

```
<script>
        var grade='B';
        var result;
        switch(grade){
                case 'A':
                        result="A Grade";
```

```
                break;
            case 'B':
                result="B Grade";
                break;
            case 'C':
                result="C Grade";
                break;
                default:
                result="No Grade";
        }
        document.write(result);
    </script>
```

JavaScript Loops

The **JavaScript loops** are used *to iterate the piece of code* using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array.

There are four types of loops in JavaScript.

1. for loop
2. while loop
3. do-while loop
4. for-in loop

---

1) JavaScript For loop

The **JavaScript for loop** *iterates the elements for the fixed number of times*. It should be used if number of iteration is known. The syntax of for loop is given below.

```
for (initialization; condition; increment)
{
  code to be executed
}
```

Ex:

```
<script>
for (i=1; i<=5; i++)
{
document.write(i + "<br/>")
}
</script>
```

2) JavaScript while loop

The **JavaScript while loop** *iterates the elements for the infinite number of times*. It should be used if number of iteration is not known. The syntax of while loop is given below.

```
while (condition)
{
  code to be executed
}
```

Ex:

```
<script>
var i=11;
while (i<=15)
{
document.write(i + "<br/>");
i++;
}
```

```
</script>
```

3) JavaScript do while loop

The **JavaScript do while loop** *iterates the elements for the infinite number of times* like while loop. But, code is *executed at least* once whether condition is true or false. The syntax of do while loop is given below.

```
do{
   code to be executed
}while (condition);
```

Let's see the simple example of do while loop in javascript.

```
<script>
var i=21;
do{
document.write(i + "<br/>");
i++;
}while (i<=25);
</script>
```

4) JavaScript for in loop

The **JavaScript for in loop** is used *to iterate the properties of an object.*

JavaScript Functions

JavaScript functions are used to perform operations. We can call JavaScript function many times to reuse the code.

Advantage of JavaScript function

There are mainly two advantages of JavaScript functions.

1. **Code reusability**: We can call a function several times so it save coding.
2. **Less coding**: It makes our program compact. We don't need to write many lines of code each time to perform a common task.

JavaScript Function Syntax

The syntax of declaring function is given below.

```
function functionName([arg1, arg2, ...argN]){
//code to be executed
}
```

JavaScript Functions can have 0 or more arguments.

JavaScript Function Example

```
<script>
function msg(){
alert("hello! this is message");
}
</script>
<input type="button" onclick="msg()" value="call function"/>
```

JavaScript Function Arguments

We can call function by passing arguments.

```
<script>
function getcube(number){
alert(number*number*number);
}
</script>
<form>
```

```
<input type="button" value="click" onclick="getcube(4)"/>
</form>
```

Function with Return Value

We can call function that returns a value and use it in our program.

```
<script>
function getInfo(){
return "hello javatpoint! How r u?";
}
</script>
<script>
document.write(getInfo());
</script>
```

JavaScript Objects

A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc. JavaScript is an object-based language. Everything is an object in JavaScript. JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

Creating Objects in JavaScript

There are 3 ways to create objects.

1. By object literal
2. By creating instance of Object directly (using new keyword)
3. By using an object constructor (using new keyword)

1) JavaScript Object by object literal

The syntax of creating object using object literal is given below:

```
object={property1:value1,property2:value2.....propertyN:valueN}
```

As you can see, property and value is separated by : (colon).

Ex:

```
<script>
emp={id:102,name:"Shyam Kumar",salary:40000}
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

2) By creating instance of Object

The syntax of creating object directly is given below:

```
var objectname=new Object();
```

Here, new keyword is used to create object.

Ex:

```
<script>
var emp=new Object();
emp.id=101;
emp.name="Ravi Malik";
emp.salary=50000;
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

3) By using an Object constructor

Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.The this keyword refers to the current object.

Ex:

```
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;
}
e=new emp(103,"Vimal Jaiswal",30000);

document.write(e.id+" "+e.name+" "+e.salary);
</script>
```

Different Objects of JavaScript are:

1. Date
2. Number
3. Boolean
4. Math
5. Window

JavaScript Date Object

The **JavaScript date** object can be used to get year, month and day. You can display a timer on the webpage by the help of JavaScript date object. Date constructors are used to create date object. It provides methods to get and set day, month, year, hour, minute and seconds.

Constructor

You can use 4 variant of Date constructor to create date object.

1. Date()
2. Date(milliseconds)
3. Date(dateString)
4. Date(year, month, day, hours, minutes, seconds, milliseconds)

JavaScript Date Methods

The important methods of date object are as follows:

Method	Description
getFullYear()	returns the year in 4 digit e.g. 2015. It is a new method and suggested than getYear() which is now deprecated.
getMonth()	returns the month in 2 digit from 0 to 11. So it is better to use getMonth()+1 in your code.
getDate()	returns the date in 1 or 2 digit from 1 to 31.
getDay()	returns the day of week in 1 digit from 0 to 6.
getHours()	returns all the elements having the given name value.
getMinutes()	returns all the elements having the given class name.
getSeconds()	returns all the elements having the given class name.
getMilliseconds()	returns all the elements having the given tag name.

Ex:

```
Current Date and Time: <span id="txt"></span>
<script>
var today=new Date();
document.getElementById('txt').innerHTML=today;
</script>
```

JavaScript Math Object:

The **JavaScript math** object provides several constants and methods to perform mathematical operation. Unlike date object, it doesn't have constructors.

Math.sqrt(n): The JavaScript math.sqrt(n) method returns the square root of the given number.

Ex:

```
Square Root of 17 is: <span id="p1"></span>
<script>
document.getElementById('p1').innerHTML=Math.sqrt(17);
</script>
```

Math.random() : The JavaScript math.random() method returns the random number between 0 to 1.

```
Random Number is: <span id="p2"></span>
<script>
document.getElementById('p2').innerHTML=Math.random();
</script>
```

Math.pow(m,n) : The JavaScript math.pow(m,n) method returns the m to the power of n that is m^n.

```
3 to the power of 4 is: <span id="p3"></span>
<script>
document.getElementById('p3').innerHTML=Math.pow(3,4);
</script>
```

JavaScript Number Object:

The **JavaScript number** object *enables you to represent a numeric value*. It may be integer or floating-point. By the help of Number() constructor, you can create number object in JavaScript.

For example:

```
var n=new Number(value);
```

Methods	Description
toExponential(x)	displays exponential value.
toFixed(x)	limits the number of digits after decimal value.
toPrecision(x)	formats the number with given number of digits.
toString()	converts number into string.
valueOf()	coverts other type of value into number.

JavaScript Boolean :

JavaScript Boolean is an object that represents value in two states: *true* or *false*. You can create the JavaScript Boolean object by Boolean() constructor as given below.

```
Boolean b=new Boolean(value);
```

JavaScript Boolean Example

```
1. <script>
2. document.write(10<20);//true
3. document.write(10<5);//false
4. </script>
```

JavaScript Boolean Methods

Method	Description
toSource()	returns the source of Boolean object as a string.
toString()	converts Boolean into String.
valueOf()	converts other type into Boolean.

Window Object:

The **window object** represents a window in browser. An object of window is created automatically by the browser. Window is the object of browser, **it is not the object of javascript**. The javascript objects are string, array, date etc.

Methods of window object

Method	Description
alert()	displays the alert box containing message with ok button.
confirm()	displays the confirm dialog box containing message with ok and cancel button.
prompt()	displays a dialog box to get input from the user.
open()	opens the new window.
close()	closes the current window.
setTimeout()	performs action after specified time like calling function, evaluating expressions etc.

Example of alert() in javascript

It displays alert dialog box. It has message and ok button.

```
<script type="text/javascript">
function msg(){
 alert("Hello Alert Box");
}
</script>
<input type="button" value="click" onclick="msg()"/>
```

Example of confirm() in javascript

It displays the confirm dialog box. It has message with ok and cancel buttons.

```
<script type="text/javascript">
function msg(){
var v= confirm("Are u sure?");
if(v==true){
alert("ok");
}
else{
alert("cancel");
}

}
```

```
</script>

<input type="button" value="delete record" onclick="msg()"/>
```

Example of prompt() in javascript

It displays prompt dialog box for input. It has message and textfield.

```
<script type="text/javascript">
function msg(){
var v= prompt("Who are you?");
alert("I am "+v);

}
</script>

<input type="button" value="click" onclick="msg()"/>
```

Example of open() in javascript

It displays the content in a new window.

```
<script type="text/javascript">
function msg(){
open("http://www.javatpoint.com");
}
</script>
<input type="button" value="javatpoint" onclick="msg()"/>
```

Example of setTimeout() in javascript

It performs its task after the given milliseconds.

```
<script type="text/javascript">
function msg(){
setTimeout(
function(){
alert("Welcome to Javatpoint after 2 seconds")
},2000);

}
</script>

<input type="button" value="click" onclick="msg()"/>
```

FUNCTIONS:

A function is a reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions.

Like any other advanced programming language, JavaScript also supports all the features necessary to write modular code using functions. JavaScript allows us to write our own functions which are user defined functions.

Function Definition

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

```
<script type="text/javascript">
  <!--
    function functionname(parameter-list)
    {
      statements
    }
  //-->
</script>
```

Ex:

```
<script type="text/javascript">
    function sayHello()
    {
      alert("Hello there");
    }
</script>
```

Document Object Model (DOM)

The **document object** represents the whole html document.
When html document is loaded in the browser, it becomes a document object. It is the **root element** that represents the html document. It has properties and methods. By the help of document object, we can add dynamic content to our web page.
As mentioned earlier, it is the object of window. So

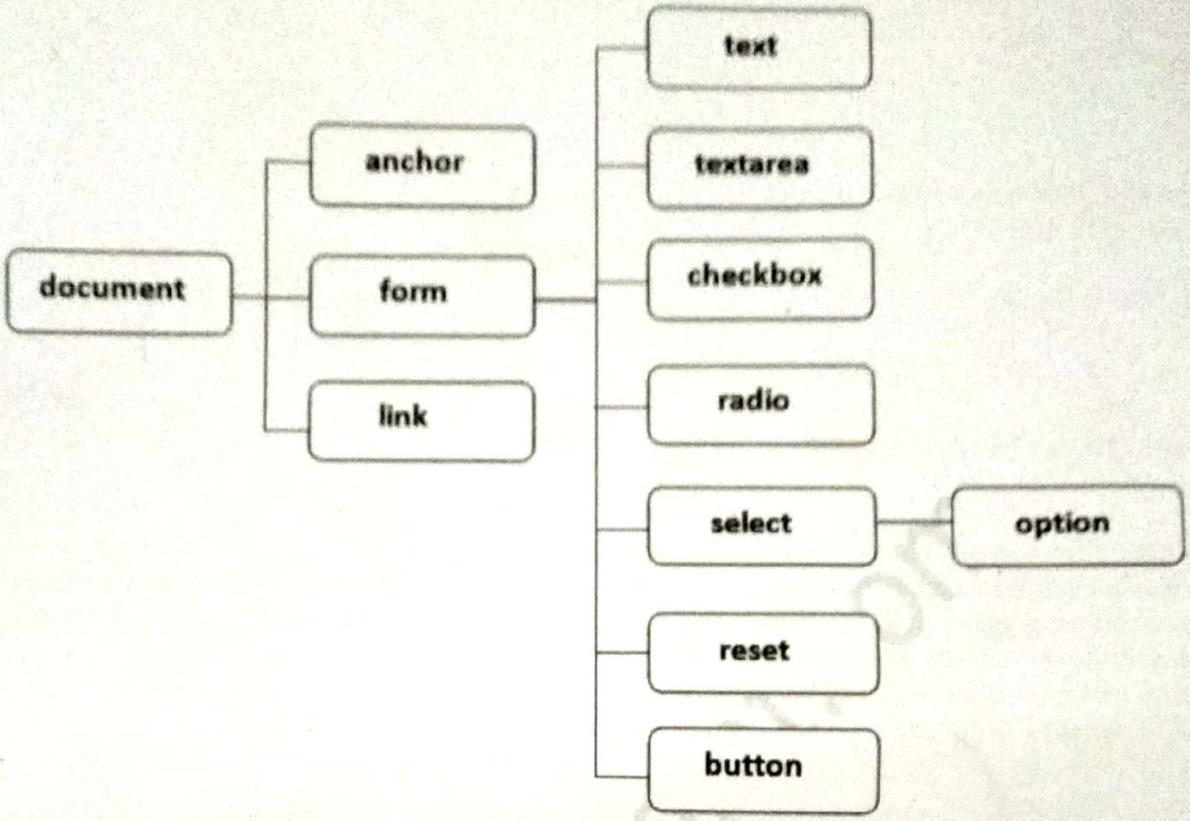
1. window.document

Is same as

1. document

Properties of document object

Let's see the properties of document object that can be accessed and modified by the document object.



Methods of document object

We can access and change the contents of document by its methods.
The important methods of document object are as follows:

Method	Description
write("string")	writes the given string on the doucment.
writeln("string")	writes the given string on the doucment with newline character at the end.
getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.
getElementsByClassName()	returns all the elements having the given class name.

Accessing field value by document object

In this example, we are going to get the value of input text by user. Here, we are using **document.form1.name.value** to get the value of name field.
Here, **document** is the root element that represents the html document.
form1 is the name of the form.
name is the attribute name of the input text.
value is the property, that returns the value of the input text.

Let's see the simple example of document object that prints name with welcome message.

```
<script type="text/javascript">
```

```
function printvalue(){
var name=document.form1.name.value;
alert("Welcome: "+name);
}
</script>

<form name="form1">
Enter Name:<input type="text" name="name"/>
<input type="button" onclick="printvalue()" value="print name"/>
</form>
```

Javascript - document.getElementById() method :

The **document.getElementById()** method returns the element of specified id. In the previous page, we have used **document.form1.name.value** to get the value of the input value. Instead of this, we can use document.getElementById() method to get value of the input text. But we need to define id for the input field.

Let's see the simple example of document.getElementById() method that prints cube of the given number.

```
<script type="text/javascript">
function getcube(){
var number=document.getElementById("number").value;
alert(number*number*number);
}
</script>
<form>
Enter No:<input type="text" id="number" name="number"/><br/>
<input type="button" value="cube" onclick="getcube()"/>
</form>
```

Javascript - document.getElementsByName() method:

The **document.getElementsByName()** method returns all the element of specified name.

The syntax of the getElementsByName() method is given below:

```
document.getElementsByName("name")
```

Here, name is required.

Ex:

In this example, we going to count total number of genders. Here, we are using getElementsByName() method to get all the genders.

```
<script type="text/javascript">
function totalelements()
{
var allgenders=document.getElementsByName("gender");
alert("Total Genders:"+allgenders.length);
}
</script>
<form>
Male:<input type="radio" name="gender" value="male">
Female:<input type="radio" name="gender" value="female">

<input type="button" onclick="totalelements()" value="Total Genders">
```

```
</form>
```

Javascript - document.getElementsByTagName() method:

The **document.getElementsByTagName()** method returns all the element of specified tag name.

The syntax of the getElementsByTagName() method is given below:

```
document.getElementsByTagName("name")
```

Here, name is required.

Ex:

In this example, we going to count total number of paragraphs used in the document. To do this, we have called the document.getElementsByTagName("p") method that returns the total paragraphs.

```
<script type="text/javascript">
function countpara(){
var totalpara=document.getElementsByTagName("p");
alert("total p tags are: "+totalpara.length);

}
</script>
<p>This is a pragraph</p>
<p>Here we are going to count total number of paragraphs by getElementByTagName() method.</p>

<p>Let's see the simple example</p>
<button onclick="countpara()">count paragraph</button>
```

Javascript - innerHTML

The **innerHTML** property can be used to write the dynamic html on the html document. It is used mostly in the web pages to generate the dynamic html such as registration form, comment form, links etc.

Example of innerHTML property

In this example, we are going to create the html form when user clicks on the button.

In this example, we are dynamically writing the html form inside the div name having the id mylocation. We are identifing this position by calling the document.getElementById() method.

```
<script type="text/javascript" >
function showcommentform() {
var data="Name:<input type='text' name='name'><br>Comment:<br><textarea rows='5' cols='80'></t
extarea>
<br><input type='submit' value='Post Comment'>";
document.getElementById('mylocation').innerHTML=data;
}
</script>
<form name="myForm">
<input type="button" value="comment" onclick="showcommentform()">
<div id="mylocation"></div>
</form>
```

Javascript - innerText

The **innerText** property can be used to write the dynamic text on the html document. Here, text will not be interpreted as html text but a normal text. It is used mostly in the we pages to generate the dynamic content such as writing the validation message, password strength etc.

Javascript innerText Example

In this example, we are going to display the password strength when releases the key after press.

```
<script type="text/javascript" >
function validate() {
var msg;
if(document.myForm.userPass.value.length>5){
msg="good";
}
else{
msg="poor";
}
document.getElementById('mylocation').innerText=msg;
}

</script>
<form name="myForm">
<input type="password" value="" name="userPass" onkeyup="validate()">
Strength:<span id="mylocation">no strength</span>
</form>
```

JavaScript Form Validation

It is important to validate the form submitted by the user because it can have inappropriate values. So validation is must. The JavaScript provides you the facility the validate the form on the client side so processing will be fast than server-side validation. So, most of the web developers prefer JavaScript form validation. Through JavaScript, we can validate name, password, email, date, mobile number etc fields.

JavaScript form validation example

In this example, we are going to validate the name and password. The name can't be empty and password can't be less than 6 characters long.

Here, we are validating the form on form submit. The user will not be forwarded to the next page until given values are correct.

```
<script>
function validateform(){
var name=document.myform.name.value;
var password=document.myform.password.value;

if (name==null || name==""){
 alert("Name can't be blank");
 return false;
}else if(password.length<6){
 alert("Password must be at least 6 characters long.");
 return false;
 }
}
</script>
<body>
<form name="myform" method="post" action="abc.jsp" onsubmit="return validateform()" >
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
```

```
</form>
```

JavaScript Retype Password Validation

```
<script type="text/javascript">
function matchpass(){
var firstpassword=document.f1.password.value;
var secondpassword=document.f1.password2.value;

if(firstpassword==secondpassword){
return true;
}
else{
alert("password must be same!");
return false;
}
}
</script>

<form name="f1" action="register.jsp" onsubmit="return matchpass()">
Password:<input type="password" name="password" /><br/>
Re-enter Password:<input type="password" name="password2"/><br/>
<input type="submit">
</form>
```

JavaScript Number Validation

Let's validate the textfield for numeric value only. Here, we are using isNaN() function.

```
<script>
function validate(){
var num=document.myform.num.value;
if (isNaN(num)){
  document.getElementById("numloc").innerHTML="Enter Numeric value only";
  return false;
 }else{
  return true;
  }
 }
 </script>
 <form name="myform" onsubmit="return validate()" >
 Number: <input type="text" name="num"><span id="numloc"></span><br/>
 <input type="submit" value="submit">
 </form>
```

HTML/DOM events for JavaScript

HTML or DOM events are widely used in JavaScript code. JavaScript code is executed with HTML/DOM events. So before learning JavaScript, let's have some idea about events.

Events	Description
onclick	occurs when element is clicked.
ondblclick	occurs when element is double-clicked.
onfocus	occurs when an element gets focus such as button, input, textarea etc.
onblur	occurs when form looses the focus from an element.
onsubmit	occurs when form is submitted.
onmouseover	occurs when mouse is moved over an element.
onmouseout	occurs when mouse is moved out from an element (after moved over).
onmousedown	occurs when mouse button is pressed over an element.
onmouseup	occurs when mouse is released from an element (after mouse is pressed).
onload	occurs when document, object or frameset is loaded.
onunload	occurs when body or frameset is unloaded.
onscroll	occurs when document is scrolled.
onresized	occurs when document is resized.
onreset	occurs when form is reset.
onkeydown	occurs when key is being pressed.
onkeypress	occurs when user presses the key.
onkeyup	occurs when key is released.

What is an Event ?

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

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.

Please go through this small tutorial for a better understanding HTML Event Reference. Here we will see a few examples to understand a relation between Event and JavaScript –

onclick Event Type

This is the most frequently used event type which occurs when a user clicks the left button of his mouse. You can put your validation, warning etc., against this event type.

Example

Try the following example.

Live Demo

```
<html>
   <head>
      <script type = "text/javascript">
         <!--
            function sayHello() {
               alert("Hello World")
            }
         //-->
      </script>
   </head>

   <body>
      <p>Click the following button and see result</p>
      <form>
         <input type = "button" onclick = "sayHello()" value = "Say Hello" />
      </form>
   </body>
</html>
```

Output

onsubmit Event Type

onsubmit is an event that occurs when you try to submit a form. You can put your form validation against this event type.

Example

The following example shows how to use onsubmit. Here we are calling a **validate()** function before submitting a form data to the webserver. If **validate()** function returns true, the form will be submitted, otherwise it will not submit the data.

Try the following example.

```
<html>
   <head>
      <script type = "text/javascript">
         <!--
            function validation() {
               all validation goes here
               .........
               return either true or false
            }
         //-->
      </script>
   </head>

   <body>
      <form method = "POST" action = "t.cgi" onsubmit = "return validate()">
         .......
         <input type = "submit" value = "Submit" />
      </form>
   </body>
</html>
```

onmouseover and onmouseout

These two event types will help you create nice effects with images or even with text as well. The **onmouseover** event triggers when you bring your mouse over any element and the **onmouseout** triggers when you move your mouse out from that element. Try the following example.

Live Demo

```
<html>
   <head>
      <script type = "text/javascript">
         <!--
            function over() {
               document.write ("Mouse Over");
            }
            function out() {
               document.write ("Mouse Out");
            }
         //-->
      </script>
   </head>

   <body>
      <p>Bring your mouse inside the division to see the result:</p>
      <div onmouseover = "over()" onmouseout = "out()">
         <h2> This is inside the division </h2>
      </div>
   </body>
</html>
```

Output