JAVASCRIPT

- used to create client-side dynamic pages.

- *object-based scripting language*
- JavaScript Translator (embedded in the browser) is responsible for translating the JavaScript code for the web browser

Application of JavaScript

- used to create interactive websites
    - Client-side validation,
    - Dynamic drop-down menus,
    - Displaying date and time,
    - Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
    - Displaying clocks etc.
- JavaScript provides 3 places to put the JavaScript code: within body tag, within head tag and external JavaScript file.

External JavaScript file

- embed it in many html page
- provides **code re usability**
- external JavaScript file must be saved by .js extension

JavaScript Comment

- meaningful way to deliver message
- advantages of JavaScript comments.
    - **make code easy to understand**
    - **avoid the unnecessary code**

Types of JavaScript Comments

- two types of comments in JavaScript.
    - Single line and multiline comments
    - Single line comments are denoted by double forward slashes (//)
    - Multi line comments start with forward slash astrik and ending with astick forward slash

JavaScript Variable

- name of storage location
- local variable and global variable
- rules while declaring a JavaScript variable
  - 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.
- To declare JavaScript global variables inside function, you need to use **window object(**window.value=90;)

Javascript Data Types

- provides different **data types** to hold different types of values
- two types of data types in JavaScript.
  - Primitive data type
  - Non-primitive (reference) data type
- don't need to specify type of the variable because it is dynamically used by JavaScript engine.
- **five types of primitive data types in JavaScript**
  - **string,number,Boolean,undefined,null**
- **there are three types of non primitive datatypes they are**
  - **object,array,regexp**

JavaScript Operators

- used to perform operations on operands
- types of operators in JavaScript
  - Arithmetic Operators (+,-,*,/,%,++,--)
  - Comparison (Relational) Operators(==,===,!=,!==,>,<,>=,<=,)
  - Bitwise Operators(&,|,^,~,<<,>>,>>>)
  - Logical Operators(&&,||,!)
  - Assignment Operators(=,+=,-=,*=,/=,%=)
  - Special Operators(?:,delete,in,instanceof,new,typeof,void,yield)

JavaScript If-else

- *execute the code whether condition is true or false*
- If Statement
- If else statement
- if else if statement

JavaScript Switch

- convenient than *if..else..if* because it can be used with numbers, characters etc
- don't forget to use break statement in cases because if you not use all the cases after one case will be excecuted

JavaScript Loops

- used *to iterate the piece of code* using for, while, do while or for-in loops.mostly used in array
- there are four types of loops for, while , do-while , for-in loop

JavaScript Functions

- used to perform operations
- can call JavaScript function many times to reuse the code
- JavaScript Functions can have 0 or more arguments.

JavaScript Objects

- entity having state and behavior (properties and method), object-based language
- we don't create class to get the object. But, we direct create objects

There are 3 ways to create objects.

- By object literal
  - object={property1:value1,property2:value2.....propertyN:valueN}
- By creating instance of Object directly (using new keyword)
  - var objectname=new Object();
- By using an object constructor (using new keyword)

JavaScript Array

- object that represents a collection of similar type of elements

3 ways to construct array in JavaScript

1. By array literal
    1. var arrayname=[value1,value2.....valueN];
2. By creating instance of Array directly (using new keyword)
    1. var arrayname=new Array();
3. By using an Array constructor (using new keyword)
    1. var emp=new Array("Jai","Vijay","Smith");

JavaScript String

- object that represents a sequence of characters

2 ways to create string in JavaScript

- By string literal
    - var stringname="string value";
- By string object (using new keyword)
    - var stringname=new String("string literal");

JavaScript Date Object

- 4 ways to create a Date object
    - Date()
    - Date(milliseconds)
    - Date(dateString)
    - Date(year, month, day, hours, minutes, seconds, milliseconds)

getDate() method

- returns the day for the specified date on the basis of local time
- dateObj.getDate()

getDay() method

- value of day of the week
- value of the day starts with 0 that represents Sunday.
- dateObj.getDay()

getFullYear() method

- dateObj.getFullYear()

getHours() getMilliseconds() getMinutes() getSeconds() method

- dateObj.getHours()
- dateObj.getMilliseconds()
- dateObj.getMinutes()
- dateObj.getSeconds()

getUTCDate() getUTCDay() getUTCHours() getUTCMinutes() getUTCMonth() getUTCSeconds() setHours() setMilliseconds() setMinutes() setSeconds()

- dateObj.getUTCDate()
- dateObj.getUTCDay()
- dateObj.getUTCHours()
- dateObj.getUTCMinutes()
- dateObj.getUTCMonth()
- dateObj.getUTCSeconds()
- dateObj.setHours(hoursValue, minValue, secValue, msValue)
- dateObj.setMilliseconds(msValue)
- dateObj.setMinutes(minValue[, secValue[, msValue]]])
- dateObj.setSeconds(secValue[, msValue]]])

Date toDateString() method

- dateObj.toDateString()
- string representing the date portion of a Date object.

JAVASCRIPT

JavaScript Examples

- 1)To write on a web page using javascript
  - <html>
  - <body>
  - <h2>Heading</h2>
  - <script>
  - document.write("Hello Javascript");
  - </script>
  - </body>
  - </html>
- 2)This example shows a alert pop up box when we open the page
  - <html>
  - <body>
  - <script type="text/javascript">
  - alert("Hello Javatpoint");
  - </script>
  - </body>
  - </html>
- 3) Giving alert action to the botton

- **<html>**
- **<head>**
- **<script** type="text/javascript"**>**
- function msg(){
- alert("Hello Javatpoint");
- }
- **</script>**
- **</head>**
- **<body>**
- ** **Welcome to JavaScript** **
- **<form>**
- **<input** type="button" value="click" onclick="msg()"**/>**
- **</form>**
- **</body>**
- **</html>**
- 4) Javascript variables

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

- **</script>**

➢ 5)Javascript local and global variables

- **<script>**
  - var value=50;//global variable
  - function a(){
    - var value=10;//local variable
    - window.a=20
    - alert(value);
    - }
    - Document.write(a);
- **</script>**

➢ 6)global and local variables in one program

- <script>
  - var value=50;//global variable
  - function a(){
    - var value=10;//local variable
- window.b=20;
- alert(value); //accessing with local variable
- alert(window.value); //accessing global variables
- }
  - document.write(b);
- </script>

➢ 7) javascript if-else statement

- **<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>**

➢ 8) Javascript Switch statement

- **<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>**

➢ 9) Javascript for loop (used when number of iterations performed known)

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

➢ 10) Javascript while loop (used when number of iterations performed not known)

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

➢ 11) Javascript do while loop (code excecuted at least once)

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

➢ 12)Javascript function with parameter

- **<script>**

- function getcube(number){
  - alert(number*number*number);
- }
- **</script>**
- **<form>**
- **<input** type="button" value="click" onclick="getcube(4)"**/>**
- **</form>**
- can call function that returns a value and use it in our program

➢ 13)To create a function and use that in the program

- **<script>**
- var add=new Function("num1","num2","return num1+num2");
- document.writeln(add(2,5));
- var pow=new Function("num1","num2","return Math.pow(num1,num2)");
- document.writeln(pow(2,3));
- **</script>**

➢ 14) Javascript object by objext literal

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

➢ 15) creating object using instance of object

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

➢ 16) creating object using object constructer

- **<script>**
- function emp(id,name,salary){
  - this.id=id;
  - this.name=name;
  - this.salary=sa lary;
- }
- e=new emp(103,"Vimal Jaiswal",30000);
- document.write(e.id+" "+e.name+" "+e.salary);
- **</script>**
- Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.

➢ 17)Defining a method in Javascript object

- We can define method in JavaScript object. But before defining method, we need to add property in the function with same name as method
- **<script>**
- function emp(id,name,salary){
  - this.id=id;
  - this.name=name;
  - this.salary=salary;
  - this.changeSalary=changeSalary;
  - function changeSalary(otherSalary){
    - this.salary=otherSalary;
  - }
- }
- e=new emp(103,"Sonoo Jaiswal",30000);
- document.write(e.id+" "+e.name+" "+e.salary);
- e.changeSalary(45000);
- document.write("**
**"+e.id+" "+e.name+" "+e.salary);
- **</script>**

➢ 18) Iterating over an array

- **<script>**
- var emp=["Sonoo","Vimal","Ratan"];
- for (i=0;i**<emp.length**;i++){
  - document.write(emp[i] + "**
**");
- }
- **</script>**

➢ 19)Creating an array using a keyword

- **<script>**
  - var i;
  - var emp = new Array();
  - emp[0] = "Arun";
  - emp[1] = "Varun";
  - emp[2] = "John";
  - for (i=0;i**<emp.length**;i++){
    - document.write(emp[i] + "**
**");
  - }
- **</script>**

➢ 20) JavaScript using array constructor

- **<script>**
- var emp=new Array("Jai","Vijay","Smith");
- for (i=0;i**<emp.length**;i++){
  - document.write(emp[i] + "**
**");
- }
- **</script>**

- 21) Creating a String literal
  - **<script>**
    - var str="This is string literal";
    - document.write(str);
  - **</script>**
- 22) Creating a string object using keyword
  - **<script>**
    - var stringname=new String("hello javascript string");
    - document.write(stringname);
  - **</script>**
- 23) To get current date and display in html page using javascript
  - Current Date and Time: **<span** id="txt"**>**
  - **<script>**
    - var today=new Date();
    - document.getElementById('txt').innerHTML=today;
  - **</script>**
- 24) Program to print only date month and year
  - **<script>**
    - var date=new Date();
    - var day=date.getDate();
    - var month=date.getMonth()+1;
    - var year=date.getFullYear();
    - document.write("**
**Date is: "+day+"/"+month+"/"+year);
  - **</script>**
- 25) Program to print current time of system
  - Current Time: **<span** id="txt"**>**
  - **<script>**
    - var today=new Date();
    - var h=today.getHours();
    - var m=today.getMinutes();
    - var s=today.getSeconds();
    - document.getElementById('txt').innerHTML=h+":"+m+":"+s;
  - **</script>**
- 26) JavaScript Digital clock example

```
<html>
<body>
    Current Time:<span id="time"></span>
    <script>
        window.onload=function(){getTime();}
        function getTime()
        {
            var today=new Date();
```

```
                var date=today.toDateString();
                var h=today.getHours();
                var m=today.getMinutes();
                var s=today.getSeconds();
                function checktime(i)
                {
                        if(i<10)
                        {
                                i="0"+i;
                        }
                        return i;
                }
                m=checktime(m);
                s=checktime(s);
                document.getElementById("time").innerHTML=date+"
"+h+":"+m+":"+s;
                setTimeout(function(){getTime()},1000);
        }
        </script>
</body>
</html>
```

27) Can give Date using a string also

- <script>
  - var date=**new** Date("August 15, 1947 20:22:10");
  - document.writeln(date.getDate())
- </script>

➢ 28) Changing the date portion in the form of string

- **<script>**
- var date=new Date();
- document.writeln(date.toDateString());
- **</script>**

➢ 29) Converting date object to ISO String

- **<script>**
  - var date=new Date(1947, 7, 15, 20, 22, 10);
  - document.writeln(date.toISOString());
- **</script>**

➢ 30)Converting date object into Json String

- **<script>**
  - var date=new Date();
  - var jsonDate=date.toJSON();
  - document.writeln(new Date(jsonDate).toUTCString());
- **</script>**

- 31) Converting date object into string
  - dateObj.toString()
  - **<script>**
    - var date=new Date();
    - document.writeln(date.toString());
  - **</script>**
- 32)Converting date object into time string
  - **<script>**
    - var time=new Date();
    - var time=new Date("August 15, 1947 20:22:10 GMT+0530 (India St andard Time) ");
    - document. writeln(time.toTimeString());
  - **</script>**
- **33) absolute value program**
  - <script>
  - function display()
  - {
    - var x=document.getElementById("num").value;
    - document.getElementById("result").innerHTML=Math.abs(x);
  - }
  - </script>
  - <form>
  - Enter a number: <input type="text" id="num">
  - <input type="button" onclick="display()" value="submit">
  - </form>
  -
- 34) check wether the value is finite,integer or not
  - <script>
  - function check(x,y)
  - {
    - **return** x/y;
  - }
  - document.writeln(Number.isFinite(check(0,10)));
  - document.writeln(Number.isFinite(check(10,0)));
  - document.writeln(Number.isFinite(Infinity));
  - document.writeln(Number.isFinite(-Infinity));
  - document.writeln(Number.isFinite(NaN));
  - document.writeln(Number.isInteger(check(12,2)));
  - document.writeln(Number.isInteger(check(12,5)));
  -
  - </script>

JAVASCRIPT

Example of alert() in javascript

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

Example of confirm() in javascript

```
<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

```
<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

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

Example of setTimeout() in javascript

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

}
</script>

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

Example of JavaScript Screen Object

```
<script>
document.writeln("<br/>screen.width: "+screen.width);
document.writeln("<br/>screen.height: "+screen.height);
document.writeln("<br/>screen.availWidth: "+screen.availWidth);
document.writeln("<br/>screen.availHeight: "+screen.availHeight);
document.writeln("<br/>screen.colorDepth: "+screen.colorDepth);
document.writeln("<br/>screen.pixelDepth: "+screen.pixelDepth);
</script>
```

Accessing field value by document object

```
<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>
```

document.getElementById() method

```
<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>
```

document.getElementsByName() method

```
<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>
```

document.getElementsByTagName() method

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

Javascript - innerHTML

- used mostly in the web pages to generate the dynamic html such as registration form, comment form, links etc.

```
<script type="text/javascript" >
function showcommentform() {
var data="Name:<input type='text' name='name'><br>Comment:<br><textarea rows='5' cols='80'></textarea>
<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

- can be used to write the dynamic text on the html document

```
<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 Math

- Math.abs(num)
- Math.acos(num)
- Math.asin(num)
- Math.atan(num)
- Math.cbrt(num)
- Math.ceil(num)
- Math.cos(num)
- Math.cosh(num)
- Math.exp(num)
- Math.floor(num)
- Math.hypot([num1[, num2[, ...]]])
- Math.log()
- Math.max(num1,num2,....,numN)
- Math.min(num1,num2,...,numN)
- Math.pow(base,exponent)
- Math.random()
- Math.round(num)
- Math.sign(num)
- Math.sin(num)
- Math.sinh(num)
- Math.sqrt(num)
- Math.tan(num)
- Math.tanh(num)
- Math.trunc(num)

JavaScript Number

- var n=new Number(value);
- can direct assign a number to a variable also

JavaScript Number Methods

- Number.isFinite(num)
- Number.isInteger(num)
- Number.parseFloat(a) //converts the string value to the float value
- Number.parseInt(string, radix) //if the string consists of both digits and string then only numbers will be parsed

JavaScript Number toExponential method

- <script>
- var a=989721;
- document.writeln(a.toExponential(2)); //here 2 means we are mentioning that we need //two digits after the decimal point and change it to the exponential format
- </script>

JavaScript Boolean

- **<script>**
- document.write(10**<20**);//true
- document.write(10**<5**);//false
- **</script>**

Browser Object Model

- used to interact with the browser
- default object of browser is window
- window.alert("hello javascript"); or alert("hello javascript");
- can use a lot of properties (other objects) defined underneath the window object like document, history, screen, navigator, location, innerHeight, innerWidth,

Window Object

- **window object** represents a window in browser

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.

- These are the methods of window objects

JavaScript History Object

- window.history
- it has only one property named length
- there are three methods of history object
    - history.forward()
    - history.back()
    - history.go()

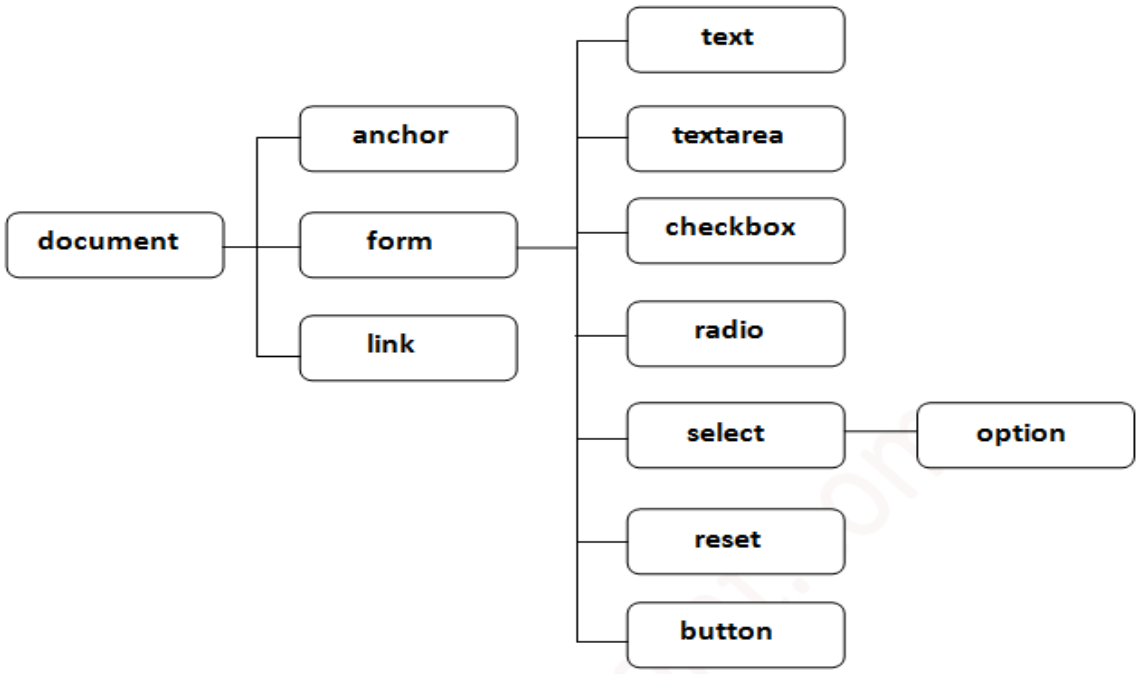
JavaScript Screen Object

- window.screen.propertyname

No.	Property	Description
1	width	returns the width of the screen
2	height	returns the height of the screen
3	availWidth	returns the available width
4	availHeight	returns the available height
5	colorDepth	returns the color depth
6	pixelDepth	returns the pixel depth.

Document Object Model

- **document object** represents the whole html document
- *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*
    - **Properties of Document object**



Methods of document object

- write("string")
- writeln("string")
- getElementById()
- getElementsByName()
- getElementsByTagName()
- getElementsByClassName()

JavaScript Form Validation

- It is important to validate the form submitted by the user because it can have inappropriate values. So, validation is must to authenticate user
- **<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 validatef
orm()" >
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
</form>
```

Password Validation

```
<html>
<body>
<script type="text/javascript">
    function match()
    {
            var pass1=document.form.pass1.value;
            var pass2=document.form.pass2.value;
            if(pass1==pass2)
            {
                    alert("Congrats Your password matched");
            }
            else
            {
                    alert("Please Try again your password not matched");
            }

    }
</script>
<form name="form" onsubmit="match()">
    Password:<input type="password" name="pass1"><br>
    Retype Password:<input type="password" name="pass2"><br>
    <input type="submit" name="">
</form>
</body>
</html>
```

email validation

- **Rules for email validation**
- email id must contain the @ and . character
- There must be at least one character before and after the @.
- There must be at least two characters after . (dot).

```
<script>
function validateemail()
{
var x=document.myform.email.value;
var atposition=x.indexOf("@");
var dotposition=x.lastIndexOf(".");
if (atposition<1 || dotposition<atposition+2 || dotposition+2>=x.length){
  alert("Please enter a valid e-
mail address \n atpostion:"+atposition+"\n dotposition:"+dotposition);
  return false;
  }
}
</script>
<body>
<form name="myform"  method="post" action="#" onsubmit="return validateemail
();">
Email: <input type="text" name="email"><br/>

<input type="submit" value="register">
</form>
```

JavaScript Classes

- JavaScript class contains various class members within a body including methods or constructor

class syntax contains two components:

- Class declarations
- Class expressions

Class Declarations

- name of the class always starts with an uppercase letter.
- <script>
- //Here, we are invoking the class before declaring it.
- var e1=new Employee(101,"Martin Roy");
- var e2=new Employee(102,"Duke William");
- e1.detail(); //calling method
- e2.detail();
-
- //Declaring class
- class Employee
- {
- //Initializing an object
- constructor(id,name)
- {
- this.id=id;
- this.name=name;
- }
- detail()
- {
- document.writeln(this.id+" "+this.name+"
")
- }
- }
- </script>
- Classes can be declared either first or last of script tag
- If it is declared twice then it will either overwrite or throws an error
- class expression can be named or unnamed.

Unnamed Class Expression

- class can be expressed without assigning any name
- <script>
- var emp = class {
- constructor(id, name) {
- this.id = id;
- this.name = name;
- }
- };
- document.writeln(emp.name);
- </script>
- For this type reclaration of classes are allowed

Named Class Expression Example

```
<script>
var emp = class Employee {
  constructor(id, name) {
    this.id = id;
    this.name = name;
  }
};
document.writeln(emp.name);
/*document.writeln(Employee.name);
Error occurs on console:
"ReferenceError: Employee is not defined
*/
</script>
```

HTML Events for the javascript:

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.

return the character from a specific index

```
var str="Javatpoint";
document.writeln(str.charAt(4));
```

JavaScript Classes

- The JavaScript class contains various class members within a body including methods or constructor

The class syntax contains two components:

- Class declarations
    1. name of the class always starts with an uppercase letter

Class Declarations Example

```
<script>
//Declaring class
class Employee
  {
//Initializing an object
    constructor(id,name)
    {
      this.id=id;
      this.name=name;
    }
//Declaring method
    detail()
    {
  document.writeln(this.id+" "+this.name+"<br>")
    }
  }
//passing object to a variable
var e1=new Employee(101,"Martin Roy");
var e2=new Employee(102,"Duke William");
e1.detail(); //calling method
e2.detail();
</script>
```

1. Unlike function declaration, the class declaration is not a part of JavaScript hoisting. So, it is required to declare the class before invoking it.
2. A class can be declared once only. If we try to declare class more than one time, it throws an error

Class expressions

- class can be expressed without assigning any name to it.

```
<script>
//Declaring class
var emp=class
  {
```

```
//Initializing an object
    constructor(id,name)
    {
      this.id=id;
      this.name=name;
    }
//Declaring method
detail()
    {
  document.writeln(this.id+" "+this.name+"<br>")
    }
  }
//passing object to a variable
var e1=new emp(101,"Martin Roy");
var e2=new emp(102,"Duke William");
e1.detail(); //calling method
e2.detail();

//Re-declaring class
var emp=class
  {
//Initializing an object
    constructor(id,name)
    {
      this.id=id;
      this.name=name;
    }
    detail()
    {
  document.writeln(this.id+" "+this.name+"<br>")
    }
  }
//passing object to a variable
var e1=new emp(103,"James Bella");
var e2=new emp(104,"Nick Johnson");
e1.detail(); //calling method
e2.detail();
</script>
```

Here the output will be the variable name

Named Class Expression Example

```
<script>
```

```
var emp = class Employee {
  constructor(id, name) {
    this.id = id;
    this.name = name;
  }
};
document.writeln(emp.name);
/*document.writeln(Employee.name);
Error occurs on console:
"ReferenceError: Employee is not defined
*/
</script>
```

JavaScript Prototype Object

- In a prototype-based approach, all the objects share the same function. This ignores the requirement of creating a new copy of function for each object. Thus, the functions are loaded once into the memory.
- Prototypes allow you to easily define methods to all instances of a particular object. The beauty is that the method is applied to the prototype, so it is only stored in the memory once, but every instance of the object has access to it

```
<script>
function Employee(firstName,lastName)
{
  this.firstName=firstName;
  this.lastName=lastName;
}

Employee.prototype.fullName=function()
  {
    return this.firstName+" "+this.lastName;
  }

var employee1=new Employee("Martin","Roy");
var employee2=new Employee("Duke", "William");
document.writeln(employee1.fullName()+"<br>");
document.writeln(employee2.fullName());
</script>
```

JavaScript Constructor Method

- A JavaScript constructor method is a special type of method which is used to initialize and create an object. It is called when memory is allocated for an object.
- class can contain one constructor method only.
- JavaScript allows us to use parent class constructor through super keyword.

```
<script>
class CompanyName
{
  constructor()
  {
    this.company="Javatpoint";
  }
}
class Employee extends CompanyName {
  constructor(id,name) {
   super();
    this.id=id;
    this.name=name;
  }
}
var emp = new Employee(1,"John");
document.writeln(emp.id+" "+emp.name+" "+emp.company);
</script>
```

JavaScript static Method

- **The static keyword is used to declare a static method.**
- **JavaScript provides static methods that belong to the class instead of an instance of that class**

```
<script>
class Test
{
  static display()
  {
    return "static method is invoked"
  }
  static display()
  {
    return "static method is invoked again"
  }
```

```
}
document.writeln(Test.display());
</script>
invoke a static method within the constructor.
<script>
class Test {
  constructor() {
  document.writeln(Test.display()+"<br>");
  document.writeln(this.constructor.display());
  }

  static display() {
      return "static method is invoked"
  }
}
var t=new Test();
</script>
```

JavaScript Encapsulation

```
<script>
class Student
 {
   constructor()
   {
     var name;
     var marks;
   }
      getName()
      {
        return this.name;
      }
     setName(name)
     {
      this.name=name;
     }

     getMarks()
     {
      return this.marks;
     }
   setMarks(marks)
```

```
    {
       if(marks<0||marks>100)
       {
         alert("Invalid Marks");
       }
      else
       {
         this.marks=marks;
       }
    }
    }
    var stud=new Student();
     stud.setName("John");
     stud.setMarks(110);//alert() invokes
     document.writeln(stud.getName()+" "+stud.getMarks());
</script>
```

JavaScript Inheritance

- The JavaScript inheritance is a mechanism that allows us to create new classes on the basis of already existing classes. It provides flexibility to the child class to reuse the methods and variables of a parent class.
- The extends keyword is used in class expressions or class declarations.
- Using extends keyword, we can acquire all the properties and behavior of the inbuilt object as well as custom classes.
- We can also use a prototype-based approach to achieve inheritance.

```
<script>
class Moment extends Date {
  constructor() {
    super();
  }}
var m=new Moment();
document.writeln("Current date:")
document.writeln(m.getDate()+"-"+(m.getMonth()+1)+"-"+m.getFullYear());
</script>
```

Or

```
<script>
class Bike
{
  constructor()
  {
    this.company="Honda";
```

```
  }
}
class Vehicle extends Bike {
  constructor(name,price) {
   super();
    this.name=name;
    this.price=price;
  }
}
var v = new Vehicle("Shine","70000");
document.writeln(v.company+" "+v.name+" "+v.price);
</script>
```

JavaScript Polymorphism

- core concept of an object-oriented paradigm that provides a way to perform a single action in different forms.
- provides an ability to call the same method on different JavaScript objects.

```
<script>
class A
  {
    display()
    {
      document.writeln("A is invoked<br>");
    }
  }
class B extends A
  {
    display()
    {
      document.writeln("B is invoked");
    }
  }

var a=[new A(), new B()]
a.forEach(function(msg)
{
msg.display();
});
</script>
```

JavaScript WeakMap Object

- type of collection which is almost similar to Map, stores each element as a key-value pair , keys are objects and the values are arbitrary values.
- **Weakmap delete()**
- **Ex: determine whether the WeakMap object contains the specified element.**
- <!DOCTYPE html>
- <html>
- <body>
- <script>
- var wm = new WeakMap();
- var obj1 = {};
- var obj2 = {};
- var obj3= {};
- wm.set(obj1, 'jQuery');
- wm.set(obj2, 'AngularJS');
- wm.set(obj3,'Bootstrap');
- document.writeln("Element present before invoking delete() method: "+ wm.has(obj1)+"
");
- wm.delete(obj1);
- document.writeln("Element present after invoking delete() method: "+wm.has(obj1)+"
");
- wm.set(obj1, "jQuery");
- document.writeln("Element present before invoking delete() method: "+ wm.has(obj1)+"
");
- document.writeln("Element is "+wm.get(obj1));
- </script>
- </body>
- </html>

JavaScript Cookies

- A cookie is an amount of information that persists between a server-side and a client-side. A web browser stores this information at the time of browsing.
- It maintains the state of a user and remembers the user's information among all the web pages.
- We cannot create an instance of Abstract Class.

How Cookies Works?

- When a user sends a request to the server, then each of that request is treated as a new request sent by the different user.

- So, to recognize the old user, we need to add the cookie with the response from the server.
- browser at the client-side.
- Now, whenever a user sends a request to the server, the cookie is added with that request automatically. Due to the cookie, the server recognizes the users.
- In JavaScript, we can create, read, update and delete a cookie by using **document.cookie** property.

- we display the cookie's name-value pair separately.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
    <select id="color" onchange="display()">
        <option value="Select Color">Select Color</option>
        <option value="yellow">Yellow</option>
        <option value="green">Green</option>
        <option value="red">Red</option>
    </select>
    <script type="text/javascript">
        function display()
        {
            var value = document.getElementById("color").value;
            if (value != "Select Color")
            {
                document.bgColor = value;
                document.cookie = "color=" + value;
            }
        }
        window.onload = function ()
        {
            if (document.cookie.length != 0)
            {
                var array = document.cookie.split("=");
                document.getElementById("color").value = array[1];
                document.bgColor = array[1];
            }
        }


    </script>
```

- </body>
- </html>

Cookie expires attribute

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4. </head>
5. <body>
6. <input type="button" value="setCookie" onclick="setCookie()">
7. <input type="button" value="getCookie" onclick="getCookie()">
8.    <script>
9.    function setCookie()
10.   {
11.      document.cookie="username=Duke Martin;expires=Sun, 20 Aug 2030 12:00:00 U
   TC";
12.   }
13.   function getCookie()
14.   {
15.      if(document.cookie.length!=0)
16.      {
17.         var array=document.cookie.split("=");
18.      alert("Name="+array[0]+" "+"Value="+array[1]);
19.      }
20.      else
21.      {
22.      alert("Cookie not available");
23.      }
24.   }
25.   </script>
26. </body>
27. </html>
```

Cookie max-age attribute

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4. </head>
5. <body>
6. <input type="button" value="setCookie" onclick="setCookie()">
7. <input type="button" value="getCookie" onclick="getCookie()">
8.    <script>
```

```
9.    function setCookie()
10.   {
11.      document.cookie="username=Duke Martin;max-
   age=" + (60 * 60 * 24 * 365) + ";"
12.   }
13.   function getCookie()
14.   {
15.      if(document.cookie.length!=0)
16.      {
17.         var array=document.cookie.split("=");
18.      alert("Name="+array[0]+" "+"Value="+array[1]);
19.      }
20.      else
21.      {
22.      alert("Cookie not available");
23.      }
24.   }
25.   </script>
26. </body>
27. </html>
```

Cookie path attribute

If a cookie is created for a webpage, by default, it is valid only for the current directory and sub-directory. JavaScript provides a path attribute to expand the scope of cookie up to all the pages of a website.

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.  </head>
5.  <body>
6.  <input type="button" value="setCookie" onclick="setCookie()">
7.  <input type="button" value="getCookie" onclick="getCookie()">
8.     <script>
9.     function setCookie()
10.    {
11.       document.cookie="username=Duke Martin;max-
   age=" + (60 * 60 * 24 * 365) + ";path=/;"
12.    }
13.    function getCookie()
14.    {
15.       if(document.cookie.length!=0)
```

```
16.       {
17.          var array=document.cookie.split("=");
18.       alert("Name="+array[0]+" "+"Value="+array[1]);
19.       }
20.       else
21.       {
22.       alert("Cookie not available");
23.       }
24.    }
25.    </script>
26. </body>
27. </html>
```

Cookie domain attribute

- **domain=javatpoint.com**

Cookie with multiple Name-Value pairs

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.  </head>
5.  <body>
6.     Name: <input type="text" id="name"><br>
7.     Email: <input type="email" id="email"><br>
8.     Course: <input type="text" id="course"><br>
9.  <input type="button" value="Set Cookie" onclick="setCookie()">
10. <input type="button" value="Get Cookie" onclick="getCookie()">
11. <script>
12.    function setCookie()
13.    {
14. //Declaring 3 key-value pairs
15.       var info="Name="+ document.getElementById("name").value+";Email="+docume
    nt.getElementById("email").value+";Course="+document.getElementById("course").val
    ue;
16. //Providing all 3 key-value pairs to a single cookie
17.       document.cookie=info;
18.    }
19.
20.    function getCookie()
21.    {
```

```
22.       if(document.cookie.length!=0)
23.       {
24.      //Invoking key-value pair stored in a cookie
25.       alert(document.cookie);
26.       }
27.       else
28.       {
29.       alert("Cookie not available")
30.       }
31.    }
32. </script>
33. </body>
34. </html>
```

Cookie with multiple Name-Value pairs using JSON

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
    Name: <input type="text" id="name"><br>
    Email: <input type="email" id="email"><br>
    Course: <input type="text" id="course"><br>
<input type="button" value="Set Cookie" onclick="setCookie()">
<input type="button" value="Get Cookie" onclick="getCookie()">

<script>
 function setCookie()
{
    document.cookie = "name=" + document.getElementById("name").value;
    document.cookie = "email=" + document.getElementById("email").value;
    document.cookie = "course=" + document.getElementById("course").value;
}
function getCookie()
{
    if (document.cookie.length != 0)
    {
        alert("Name="+document.getElementById("name").value+" Email="+documen
t.getElementById("email").value+" Course="+document.getElementById("course").v
alue);
    }
```

```
    else
    {
        alert("Cookie not available");
    }
}
 </script>
</body>
</html>
```

Deleting a Cookie in JavaScript

Different ways to delete a Cookie

These are the following ways to delete a cookie:

- A cookie can be deleted by using expire attribute.
  - document.cookie="name=Martin Roy; expires=Sun, 20 Aug 2000 12:00:00 UTC";
- A cookie can also be deleted by using max-age attribute.
  - document.cookie="name=Martin Roy;max-age=0"; document.cookie="name=Martin Roy;max-age=0";
- We can delete a cookie explicitly, by using a web browser.

```
Deleting using code
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<input type="button" value="Set Cookie1" onclick="setCookie1()">
<input type="button" value="Get Cookie1" onclick="getCookie1()">
<input type="button" value="Delete Cookie1" onclick="deleteCookie1()">
<br>
<input type="button" value="Set Cookie2" onclick="setCookie2()">
<input type="button" value="Get Cookie2" onclick="getCookie2()">
<input type="button" value="Delete Cookie2" onclick="deleteCookie2()">
<br>
<input type="button" value="Display all cookies" onclick="displayCookie()">

<script>
function setCookie1()
```

```
{
    document.cookie="name=Martin Roy";
     cookie1=  document.cookie;
}
function setCookie2()
{
    document.cookie="name=Duke William";
     cookie2=  document.cookie;
}

function getCookie1()
{
    if(cookie1.length!=0)
    {
    alert(cookie1);
    }
    else
    {
       alert("Cookie not available");
    }
}

function getCookie2()
{
    if(cookie2.length!=0)
    {
    alert(cookie2);
    }
    else
    {
       alert("Cookie not available");
    }
}

function deleteCookie1()
{
    document.cookie=cookie1+";max-age=0";
    cookie1=document.cookie;
    alert("Cookie1 is deleted");
}

function deleteCookie2()
```

```
{
    document.cookie=cookie2+";max-age=0";
    cookie2=document.cookie;
   alert("Cookie2 is deleted");
}

function displayCookie()
{
if(cookie1!=0&&cookie2!=0)
{
    alert(cookie1+" "+cookie2);
}
else if(cookie1!=0)
{
    alert(cookie1);
}
else if(cookie2!=0)
{
    alert(cookie2);
}
else{
    alert("Cookie not available");
}

}

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

JavaScript this keyword

- this keyword is a reference variable that refers to the current object
- Ex1:

```
<script>
var address=
{
company:"Javatpoint",
city:"Noida",
state:"UP",
fullAddress:function()
{
```

```
return this.company+" "+this.city+" "+this.state;
}
};


var fetch=address.fullAddress();
document.writeln(fetch);

</script>
Ex 2
<script>
var website="Javatpoint";
function web()
{
document.write(this.website);
}
web();
</script>
Ex3:
<script>
var emp_address = {
    fullAddress: function() {
        return this.company + " " + this.city+" "+this.state;
    }
}
var address = {
    company:"Javatpoint",
    city:"Noida",
    state:"UP",

}

document.writeln(emp_address.fullAddress.call(address));
document.writeln(emp_address.fullAddress.apply(address));</script>
```

JavaScript Debugging

we will find out errors using built-in web browser debugger. To perform debugging, we can use any of the following approaches:

- Using console.log() method
- Using debugger keyword
- <script>

- x = 10;
- y = 15;
- z = x + y;
- debugger;
- document.write(z);
- document.write(a);
- </script>
- Java provides manual debugger so we no need to verify step by step.

JavaScript Hoisting

- JavaScript we can use variables and functions before declaring them.
- we will use the variable and function before declaring them.

JavaScript Variable Hoisting

- <script>
- x=10;
- document.write(x);
- var x;
- </script>

JavaScript Function Hoisting

- <script>
- document.write(sum(10,20));
- function sum(a,b)
- {
- **return** a+b;
- }
- </script>

JavaScript Strict Mode

- sometimes the JavaScript code displays the correct result even it has some errors. To overcome this problem we can use the JavaScript strict mode.
- "use strict"; expression to enable the strict mode
- <script>
- "use strict";
- x=10;
- console.log(x);
- </script>
- we didn't provide the type of variable. Still we are getting an output.
- After using the keyword we will get the correct output

JavaScript Set Object

- used to store the elements with unique values
- **new** Set([iterable])
- A set object iterates its elements in insertion order.

add() method

- setObj.add(value)
- <script>
- var set = **new** Set();
- set.add("jQuery");
- set.add("AngularJS");
- set.add("Bootstrap");
- **for** (let elements of set) {
- document.writeln(elements+"
");
- }
- </script>

clear() method

- **clear()** method is used to remove all the elements from Set object.
- setObj.clear()
- <script>
- var set = **new** Set();
- set.add("jQuery");
- set.add("AngularJS");
- set.add("Bootstrap");
- document.writeln("Size before invoking clear() method: "+ set.size+"
");
- set.clear();
- document.writeln("Size after invoking clear() method: "+set.size);
- </script>

delete() method

- used to remove all the elements from Set object
- setObj.delete()
- <script>
- var set = **new** Set();
- set.add("jQuery");
- set.add("AngularJS");
- set.add("Bootstrap");

- document.writeln("Size before invoking delete() method: "+ set.size+"
");
- set.delete("Bootstrap");
- document.writeln("Size after invoking delete() method: "+set.size);
- </script>

entries() method

- maintains an insertion order.
- setObj.entries()
- <script>
- var set = **new** Set();
- set.add("jQuery");
- set.add("AngularJS");
- set.add("Bootstrap");
- var itr=set.entries();
- **for**(i=0;i<set.size;i++)
- {
- document.writeln(itr.next().value+"
");
- }
- </script>

forEach() method

- executes the specified function once for each value in the Set object. It maintains an insertion order.
- Set considers the value of element as a key.
- <script>
- var set = **new** Set(["jQuery","AngularJS","Bootstrap"]);
- document.writeln("Fetching values :"+"
");
- set.forEach(function display(value1,value2,set)
- {
- document.writeln('key[' + value1 + '] = ' + value2+"
");
- })
- </script>

has() method

- **has()** method indicates whether the Set object contains the specified value
- setObj.has(value)
- <script>
- var set = **new** Set();
- set.add("jQuery");
- set.add("AngularJS");

- set.add("Bootstrap");
- document.writeln(set.has("Bootstrap"));
- </script>

values() method

- **values()** method returns an object of new Set iterator
- maintains an insertion order.
- setObj.values()
- <script>
- var set = **new** Set();
- set.add("jQuery");
- set.add("AngularJS");
- set.add("Bootstrap");
- var itr=set.values();
- document.writeln(itr.next().value+"
");
- document.writeln(itr.next().value+"
");
- document.writeln(itr.next().value);
- </script>

Map Object

clear() method

- used to remove all the elements from Map object.
- mapObj.clear()
- <script>
- var map=**new** Map();
- map.set(1,"jQuery");
- map.set(2,"AngularJS");
- map.set(3,"Bootstrap");
- document.writeln("Size before invoking clear(): "+ map.size+"
");
- map.clear();
- document.writeln("Size after invoking clear(): "+map.size);
- </script>

delete() method

- used to remove all the elements from **Map** object.
- mapObj.delete()
- **Use the above mentioned same code just by replacing with a small line**
- map.delete(2);

entries() method

- **entries()** method returns an object of new map iterator
- mapObj.entries()
- <script>
- var map=**new** Map();
- map.set(1,"jQuery");
- map.set(2,"AngularJS");
- map.set(3,"Bootstrap");
- var itr=map.entries();
- **for**(i=0;i<map.size;i++)
- {
- document.writeln(itr.next().value+"
");
- }
- </script>

forEach() method

- execute the specified function once for each key/value pair in the **Map** object.
- mapObj.forEach(callback[, thisArg])
- <script>
- var map = **new** Map();
- map.set(1,"jQuery");
- map.set(2,"AngularJS");
- map.set(3,"Bootstrap");
- document.writeln("Fetching values and keys :"+"
");
- function display(values,key)
- {
- document.writeln(values+" "+key+"
");
- }
- map.forEach(display);
- </script>

get() method

- document.writeln(map.get(5));//This can be written in the above program

has() method

- mapObj.has(key)
- whether the **Map** object contains the specified key.
- <script>
- var map=**new** Map();

```
map.set(1,"jQuery");
map.set(2,"AngularJS");
map.set(3,"Bootstrap");
document.writeln(map.has(2));
document.writeln(map.has(5));  //this key is not present
document.writeln(map.has("jQuery"));//this is not the key
</script>
```

keys() method

```
mapObj.keys()
<script>
var map=new Map();
map.set(1,"jQuery");
map.set(2,"AngularJS");
map.set(3,"Bootstrap");
var itr=map.keys();
for(i=0;i<map.size;i++)
  {
document.writeln(itr.next().value+"<br>");
  }
  </script>
```

values() method

```
mapObj.values()
<script>
var map=new Map();
map.set(1,"jQuery");
map.set(2,"AngularJS");
map.set(3,"Bootstrap");
var itr=map.values();
document.writeln(itr.next().value+"<br>");
document.writeln(itr.next().value+"<br>");
document.writeln(itr.next().value);
 </script>
```

WeakSet Object

- WeakSet object contains unique objects only.

add() method and delete() method and has() method

- **add()** method is used to add a new object to the end of a **WeakSet** object.
- example to add object to WeakSet object.
- **delete()** method is used to remove the specified object from **WeakSet** object.
- **has()** method indicates whether the **WeakSet** object contains the specified object
- <script>
- var ws = **new** WeakSet();
- var obj={};
- ws.add(obj);
- document.writeln("Element present before invoking delete() method: "+ws.has(obj)+"
");
- ws.delete(obj);
- document.writeln("Element present before invoking delete() method: "+ws.has(obj));
- </script>