Javascript - 1

Introduction to Scripting Language

TEKNOTURF

Computer language with a series of commands within a file.

Capable of being executed without being compiled.

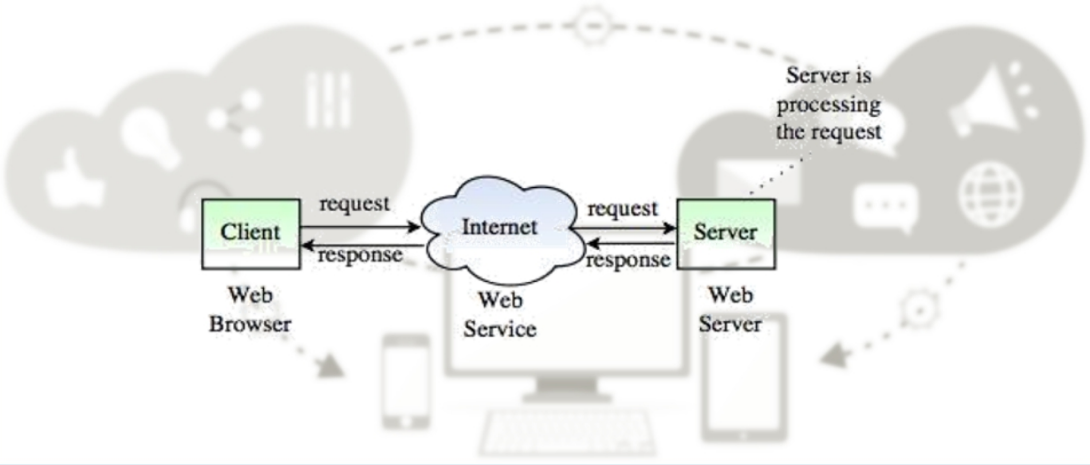
Script provides changes to the webpage.

Two types of Scripting

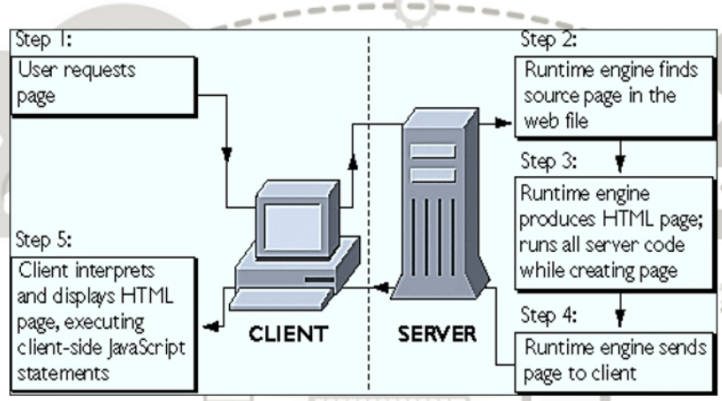
- Server side scripting
- Client side scripting

Server Side Scripting

TEKNOTURF



Client Side Scripting



JavaScript Introduction

- JavaScript can be used in **client and server side**.
- Traditionally on **client side**
  - processing user input
  - validations
- Latest Implementation
  - **Client-side/Single page application**
- On **Server-Side** using **Node JS**
  - It can create standalone application, Network application, Web Application
  - **REST services**, service layers
- From traditional validation module to
  - Javascript libraries like **JQuery, DOJO, React** and so on
  - Javascript frameworks **like Angular, Ember** and so on

Advantages of JavaScript

- Improves transaction response time
- Reduces server load
- Less server interaction – Can validate user input before sending the page off to the server.
- Provides immediate feedback

Embedding JavaScript in HTML

JavaScript is included in an HTML file with the help of <script> tag

There are two methods to embed javascript in to **HTML code**.

Internal Script

External Script

```
<script type="text/javascript">
    document.write("<h1>Hello World!</h1>");
</script>
```

```
<head>
<script type="text/Javascript">
    //  Javascript  Code
</script>
</head>
```

Inline JavaScript

```
<body>
<script type="text/Javascript">
    //  JavaScript  Code
</script>
</body >
```

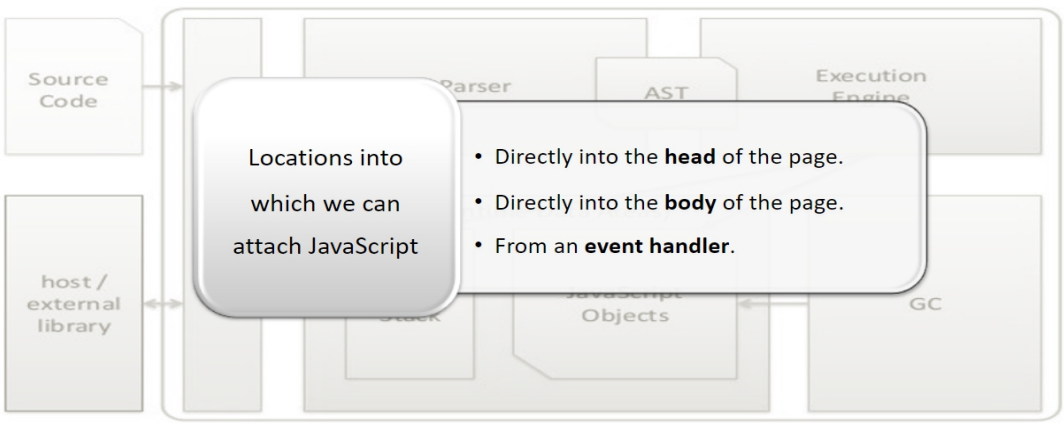
External JavaScript

- **JavaScript can be put in a separate .js file**
  - Includes the external java script file in a HTML file using the code

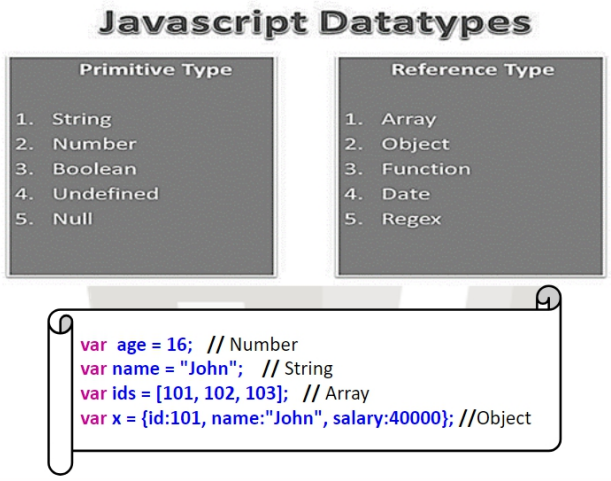
```
<script src="myJavaScriptFile.js">
</script>
```

  - An **external .js** file can be used in multiple HTML pages wherever necessary
  - The **external .js** file **cannot** itself **contain a <script> tag**
- **JavaScript can be put in an HTML form object, such as a button**
  - This JavaScript will be executed when the form object is used

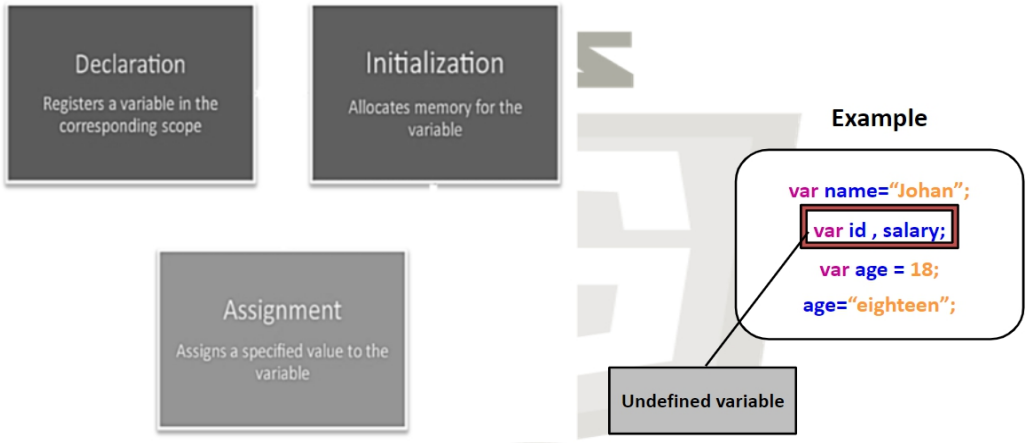
Execution of JavaScript



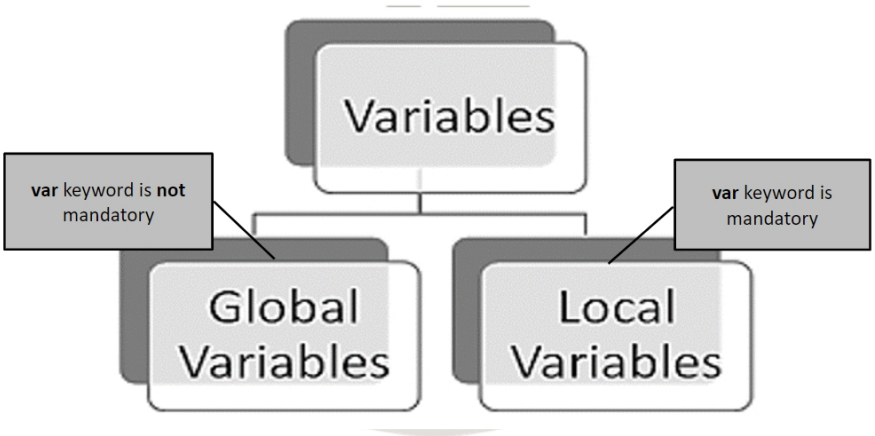
Data Types



JavaScript Variables



Scope of variables



Scope of variables

```
<body>
<p id="demo"></p>
<script>
   myCar();
   document.getElementById("demo").innerHTML = "My Car Name: " + carName;
   function myCar()
 {
       carName = "BMW";
  }
</script>
</body>
```

```
var a = 20;  ──────────→ Global Variable
function checkVariable(){
        var a = 23;  ──────→ Local Variable
  }
```

Operators

Arithmetic Operators

Operator	Description	Examples
+	Addition	var z = 5 + 2;
-	Subtraction	var z = 5 - 2;
*	Multiplication	var z = 5 * 2;
/	Division	var z = 5 / 2;
%	Modulus	var z = 5 % 2;
++	Increment	var x = 5; z = x++;
--	Decrement	var x = 5; z = x--;

Assignment Operators

Operator	Description	Examples
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

Operators contd..

Conditional Operators

Operator	Description	Example
? : (Conditional)	If Condition is true? Then value X : Otherwise value Y	var x=10 , y=20; var z=(x<y)?x:y;

Operators contd..

TEKNOTURF

typeof Operator

- The typeof operator is a unary operator
- It returns String

Example

```
var data=10;
var result = typeof data;
document.write(result);
```

Type	Return data
Number	"number"
String	"string"
Boolean	"boolean"
Object	"object"

Comments in JavaScript

TEKNOTURF

Comments are used to provide additional information about the JavaScript code, and make it more readable.

These statements will not get executed by the browser.

Single line comment

- **// Statements**

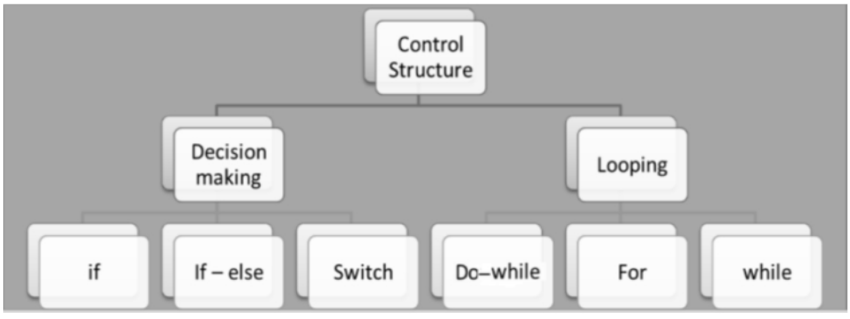
Multi line comment

- **/* Statements */**

Programming Constructs in JavaScript

TEKNOTURF

Decides the flow of execution of the JavaScript program.



if Statement

TEKNOTURF

Syntax

```
if (condition)
{
    //Statement(s)  to  be executed
       if the condition is true
}
```

Example

```
<script type="text/javascript">
     var num=10;
     if (num % 2 ==0)
     {
        document.write("num is Even");
     }
</script>
```

If..else statement

TEKNOTURF

Syntax

```
if (condition)
{
    //Statement(s)  to  be executed if the
       condition is true
}
else
{
    //Statement(s)  to  be executed if the
       condition is false
}
```

Example

```
<script type="text/javascript">
     var num=10;
     if (num % 2 ==0)
     {
        document.write("num is Even
");
     }
     else
     {
        document.write("num is Odd
");
     }
</script>
```

If..else if..else statement

TEKNOTURF

Syntax

```
if (condition1)
{
    //Statement(s)  to be executed if
       the condition1 is true
}
else if(condition2)
{
    //Statement(s)  to be executed if
       the condition2 is false
}
.....
else
{
    //statement to be executed
}
```

Example

```
<script type="text/javascript">
    var num=10;
    if (num  > 0)
    {
     document.write("num is Positive ");
    }
    else if(num < 0)
    {
      document.write("num is Negative
          ");
    }
    else
    {
      document.write("num is equal to
         zero ");
    }
</script>
```

Switch statement

TEKNOTURF

Syntax

```
switch(expression)
{
 case n:
          code block
          break;
 case n:
          code block
          break;
 default:
          code block
}
```

Example

```
<script type="text/javascript">
var exp = 0;
switch(exp)
{
    case 1 :
        document.write("the value is positive");
        break;
    case  -1 :
        document.write("the value is negative");
        break;
    default :
        document.write("the value is zero");
}
</script>
```

for Loop

TEKNOTURF

Syntax

```
<script type="text/javascript">
   var initvalue;
   for(initvalue=startvalue;initvalue condtion;inc/decvalue)
   {
          //Statements to be executed
   }
</script>
```

Example

```
<script type="text / javascript">
   var sum=0;
    for(var i=1 ; i<=5 ; i++)
    {
          sum=sum+i;
    }
    document.write("The sum is : "+sum);
</script>
```

while Loop

TEKNOTURF

Syntax

```
while (expression)
{
          Statement(s) to be executed
          if expression is true
}
```

Example

```
<script type="text/javascript">
    var sum=0;
    var i=1;
    while(i<=5)
    {
       sum=sum+i;
       i++;
    }
    document.write("the sum is :  "+sum);
</script>
```

do – while Loop

TEKNOTURF

Syntax

```
do
{
    Statement(s)
        to be executed;
} while (expression);
```

Example

```
<script type="text/javascript">
  var sum=0;
  var i=1;
  do
  {
        sum=sum+i;
        i++;
   }while(i<=5);
  document.write("the sum is :"+sum);
</script>
```

for .. in loop

TEKNOTURF

Syntax

```
for (variablename in object)
{
    statement or block to execute
}
```

Example

```
<script type="text/javascript">
    var ids = [101,102,103];
    var data;
for(data in ids)
{
   document.write(ids[data]+" ");
}
</script>
```

Built-in Functions

TEKNOTURF

- Array Methods
- String Methods
- Boolean Methods
- Math Methods
- Number Methods
- RegExp Methods
- Date Methods
- Date Static Methods
- String HTML wrappers

Built-in Functions

TEKNOTURF

Function	Description
isNan	Determines whether value is a legal number or not.
isFinite	To find whether a number is a finite legal number.
eval	Executes JavaScript source code.
number	Converts object to the corresponding number value.
string	Converts object to the corresponding string value.

Function	Description
parseInt	Converts string value to integer.
parseFloat	Converts string value to floating point number.
escape	Encodes the string value into world wide acceptable format.
encodeURI	To encode URI.
decodeURI	To decode URI.
encodeURIComponent	To encode URI component.
decodeURIComponent	To decode URI component.

Built-in Functions

TEKNOTURF

```
document.write(isNaN(0));
```

```
var obj2=new Boolean(0);
document.write(String(obj2));
```

```
document.write(escape("this is javascript escape function!!"));
```

```
document.write(encodeURI("http://www.techstrikers.com/test.php?id=23&str=this is test"));
```

```
isFinite("5678");
isFinite("isFinite");
isFinite("5678-34");
```

```
var obj1=new String("7893");
document.write(parseInt(obj1));
```

Example

TEKNOTURF

Syntax

```
function functionname(parameter-list)
{
    statements
}
```

Example

```
<script type="text/javascript">
function display()
{
    document.write("Welcome to
      JavaScript");
}
display();
</script>
```

Function call

JavaScript Functions

TEKNOTURF

Functions with parameters and without return data

```
<script type="text/javascript">
function add(no1 , no2)
{
    var sum = no1 + no2;
    document.write(sum);
}
add(10,20);
</script>
```

Function with parameters and return data

```
<script type="text/javascript">
function add(no1 , no2)
{
    var sum = no1 + no2;
     return sum;
}
var sum =add(10,20);
document.write(sum);
</script>
```

Javascript - 2

Event Handling

- Event – an action that is fired (initiated) within a webpage.
- JavaScript is Single Thread.
- It is so useful in creating interactive web sites.
- JavaScript uses asynchronous callback.
- Simplest way to run .js code in response to an event is to use an event handler (function)

Event Handling

Event	Description
onchange	Script runs when the element changes
onsubmit	Script runs when the form is submitted
onreset	Script runs when the form is reset
onselect	Script runs when the element is selected
onblur	Script runs when the element loses focus
onfocus	Script runs when the element gets focus
onkeydown	Script runs when key is pressed
onkeypress	Script runs when key is pressed and released
onkeyup	Script runs when key is released
onclick	Script runs on a mouse click
ondblclick	Script runs on a mouse double-click
onmousedown	Script runs when mouse button is pressed
onmousemove	Script runs when mouse pointer moves
onmouseout	Script runs when mouse pointer moves out of an element
onmouseover	Script runs when mouse pointer moves over an element
onmouseup	Script runs when mouse button is released

Event Handling

Example :

```
<form>
    <input type="button" name="test" value="Click me"
                    onclick="inform()">
</form>
```

```
<script>
function inform()
{
   alert("You have activated me by clicking the grey
                                                button!")
}
</script>
```

When the user clicks the button, **"inform()"** will be called.

JavaScript Validation

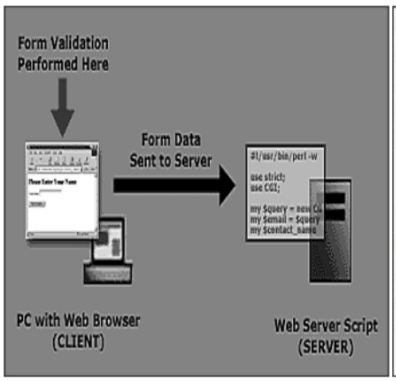
JavaScript data validation happens before form is submitted.

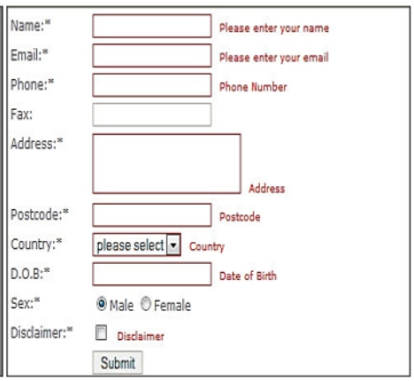
Server-side application validation happens after the form is submitted to the application server.

Form validation performs the following functions :

- Basic Validation
- Data Format Validation

JavaScript Validation





JavaScript Validation

```
<form name="register" action="#" method="post">
First Name <input type="text" name="fname" > <br/>
Last Name <input type="text" name="lname" > <br/>
<input type="button" value="Register" onclick="validateData()">
<br/>
</form>
```

```
var fname=document.register.fname.value;
var lname=document.register.lname.value;

if(fname==null || fname== " "|| lname==null ||
lname.trim()=="")
{
    document.getElementById("msg").innerHTML +="Enter value
for name <br />";
}
```

JavaScript Validation

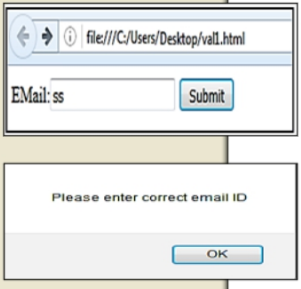
The data entered in a form can be validated for its format.

Our code must include appropriate logic to test the correctness of data.

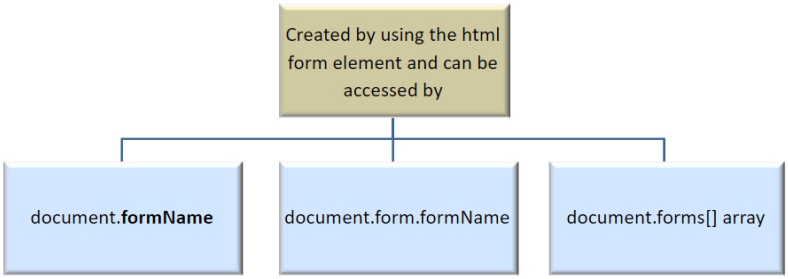
```
<body>
 <form name="myForm" action="welcomePage.html" method="post" >
Email:<input type="text" name="email" id="email" onBlur="return validateEmail()"/>
<input type="submit" value="Submit" onsubmit="return validateEmail()">
</form>
```

JavaScript Validation

```
<script type="text/javascript">
    function validateEmail()
     {
          var emailID = document.myForm.EMail.value;
          atpos = emailID.indexOf("@");
          dotpos = emailID.lastIndexOf(".");
          if(atpos < 1 || ( dotpos - atpos < 2 ))
          {
            alert("Please enter correct email ID");
            document.myForm.EMail.value="" ;
            document.getElementById("email").focus() ;
            return false;
          }
       return true ;
    }
</script>
</body>
```



Form Object



Form Object

Property	Description
action	Presents the action attribute.
autocomplete	Presents the autocomplete attribute (on / off)
encoding	Presents the enctype attribute.
length	Presents the number of elements on a form.
method	Presents the forms method attribute.
name	Presents the name attribute of the form.
noValidate	Presents if form data needs to be validated or not (true / false)
target	Presents the target attribute of the form. It represents the name of the frame or window to which the form submission response is sent by the server.

Form Object

```
<form id="myForm" action="homepage.html" >
<table>
<tr><td>User name </td><td><input type="text" name="uname"></td></tr>
<tr><td>Password</td><td> <input type="password" name="pwd"></td></tr>
<tr><td colspan="2"><input type="button" value="Submit"  onClick = "changeAction()" >
</td> </tr>
</table>  </form>
<div id="msg" ></div>
<script>
      function changeAction()
      {
            document.getElementById("myForm").action = "form_action.asp";
            document.getElementById("myForm").autocomplete = "off;
            document.getElementById("msg").innerHTML = "The value of the action
               attribute was changed";
      }
</script>
```

Form Object - Methods

```
<body>
<form name="register" action="registerUser.html">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="button" onclick="myFunction()" value="Submit form">
</form>
<script>
function myFunction()
{
   document.register.submit();
}
</script>
</body>
```

Form accessed as document.formName and form submission invoked using submit() method

Form Event Handler

```
<body >

<form name="register" onreset="return display()" onsubmit="return
displayValues()">
  First name: <input type="text" name="fname"   onblur = "alert(this.value)" ><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" >
  <input type="reset" >
</form>

<script>

function display()
{
  document.register.fname.value="Pearson";
  document.register.lname.value="David";
  return false;
}
```

Form Events onset, onreset and onblur used

Accessing a textbox value from a form

Form Event Handler

Executed when onsubmit event occurs

```
function displayValues()
{
var fname=document.register.fname.value;
var lname=document.register.lname.value;
alert("First name is "+fname+" Last name is
"+lname);
return false;
}
</script>
</body>
```

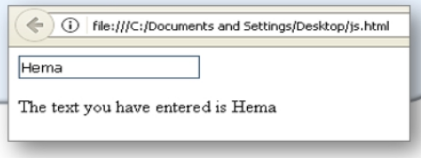
Text Object

<input type=”text” name="firstname">

Properties	Methods	Events
• defaultValue	• blur()	• onBlur
• form	• focus()	• onChange
• name	• select()	• onFocus
• type		• onSelect
• value		

Text Object

```
<html>
<body>
<form>
<input type="text" onblur="display(this)"/>
<p id="demo"></p>
<script type="text/javascript">
function display(str)
{
    document.getElementById("demo").innerHTML = "The text you have entered is “
                                                +str.value;
}
</script>
</body> </html>
```



Button Object

<input type=***"button"*** *name*="myButton" value="Press This“ onClick="clickFunction">

The input type could be "button" or "submit" or "reset".

Properties	Methods	Events
• disabled	• blur() - Takes the focus away from the radio button.	• onBlur
• form	• click() - This function acts as if the user clicked the button.	• onClick
• name	• focus() - Gives the focus to the checkbox.	• onFocus
• type		
• value		

Checkbox Object


```
<INPUT TYPE="checkbox" NAME="Name1" VALUE="1" CHECKED onClick="clickFunction">
```

The option "CHECKED" sets the button so it is selected when it is initially displayed.

Properties

- checked
- defaultChecked
- form
- name
- type
- value

Checkbox Object

Methods

- blur()
- click()
- focus()

Events

- onBlur
- onClick
- onFocus

Radio Object

Represents an HTML <input> element with type="radio"

Properties

- checked
- defaultChecked
- form
- name
- type
- value

Methods

- blur()
- click()
- focus()

Radio Object

Events

- onBlur
- onClick
- onFocus

```
<body>
  <form name="form1">
  <p><input type=radio name="seats" value="sleeper">Sleeper</p>
  <p><input type=radio name="seats" value="semisleeper">Semi Sleeper</p>
  <p><input type=radio name="seats" value="normal">Normal</p>
  <p><input type=button value="Show Selected Seat"
     onClick="getSelectedSeat(this.form.seats)"></p>
  <p><div id="msg"></div></p>
  </form>
```

Select Object

Represents HTML <select> element

Properties

- form
- length
- name
- options
- selectedIndex
- type

Methods

- blur()
- focus()
- add()
- remove()

Select Object

Events

- onBlur
- onChange
- onFocus

```
<body>
<form>
<select id="technology“
onchange="selectMethodsDemo()" >
 <option>HTML</option>
 <option>CSS3</option>
 <option>Javascript</option>
 <option>JQuery</option>
</select>
</form>
```

Javascript - 3

Implicit Objects in JavaScript

- Object-based Language
- Has State and Behaviour
- Template based
- Array, String, Number, Boolean, Date, Math are few implicit objects in JavaScript.

String

Array of characters

```
var name= "Teknoturf";
var name= new
String("Teknoturf");
```

Property

```
name.length ➔ 9
```

Special Characters

```
name="John \"David\"";
```

Code	Outputs
\'	single quote
\"	double quote
\\	backslash

String Methods

Method	Description
charAt()	Returns the character at the specified index (position)
concat()	Joins two or more strings, and returns a copy of the joined strings
indexOf()	Returns the position of the first occurrence of a specified value in a string
lastIndexOf()	Returns the position of the last occurrence of a specified value in a string
localeCompare()	Compares two strings in the current locale
replace()	Searches within a string for a value and returns a new string with the replaced value
search()	Searches a string for a value and returns the position of the match

URL String Encoding and Decoding

The encodeURI() function is used to encode a URI.

This function encodes special characters, except: , / ? : @ & = + $ #

```
<p>Click the button to encode a URI.</p>
<button onclick="testEncode()">Try it</button>
<p id="uridemo"></p>
<script>
function testEncode ()
{
   var uri = "teknoturf infoservices?name=Tinå&location=USA";
   var res = encodeURI(uri);
   document.getElementById("uridemo").innerHTML = res;
}
</script>
```

teknoturf%20infoservices?name=Tin%C3%A5 &location=USA

URL String Encoding and Decoding

```
<p>Click the button to decode a URI after encoding it.</p>
<button onclick="testDecode()">Try it</button>
<p id="uridemo"></p>
<script>
function testDecode() {
   var uri = "teknoturf infoservices?name=Tinå&location=USA";
   var encode = encodeURI(uri);
   var decode = decodeURI(encode);
   var res = "Encoded URI: " + encode + "<br>" + "Decoded URI: " + decode;
   document.getElementById("uridemo").innerHTML = res;
}
</script>
```

Encoded URI: teknoturf%20infoservices?name=Tin%C3%A5&location=USA
Decoded URI: teknoturf infoservices?name=Tinå&location=USA

Math Object

Properties	Description	Example
PI	Returns PI (approx. 3.14)	document.write(Math.PI); //returns 3.141592653589793
SQRT1_2	Returns the square root of 1/2 (approx. 0.707)	document.write(Math.SQRT1_2); //returns 0.7071067811865476
SQRT2	Returns the square root of 2 (approx. 1.414)	document.write(Math.SQRT2); //returns 1.4142135623730951

Math Functions

Function	Description	Example
abs(x)	Returns the absolute value of x	Math.abs(-7.25); // 7.25
ceil(x)	Returns x, rounded upwards to the nearest integer	Math.ceil(1.4) //2
floor(x)	Returns x, rounded downwards to the nearest integer	Math.floor(1.6); //1
max(x,y,z,..., n)	Returns the number with the highest value	Math.max(5, 10); //10
min(x,y,z,...,n)	Returns the number with the lowest value	Math.min(5, 10); //5
pow(x,y)	Returns the value of x to the power of y	Math.pow(4, 3); //64

Date Object

Helps to work with dates in JavaScript

Can be created in many ways

- new Date() Tue Jan 24 2017 11:59:10 GMT+0530 (India Standard Time)
- new Date(milliseconds) Tue Jan 24 2017 11:59:00 GMT+0530 (India Standard Time)
- new Date(dateString) Tue Jan 24 2017 12:02:10 GMT+0530 (India Standard Time)
- new Date(year, month, day, hours, minutes, seconds, milliseconds)

Examples

```
<script type="text/javascript">
      var  date1 = new Date();
      var  date2 = new Date("January 24 , 2017 11:59:00");
      var  date3 = new Date(2017,0,24,12,02,10,15,20);
      document.write(date1+"<br />"+date2+"<br />"+date3);
   </script>
```

Date Object

Date Formats – can be used when constructing date object

ISO Format - (YYYY-MM-DD) -- DD and MM are optional
Long Date - (MMM DD YYYY) -- year, month, and day can be in any order
Short Date - (MM/DD/YYYY) -- Either "/" or "-" can be used as a separator

Method	Description
getDate()	Get the day as a number (1-31)
getDay()	Get the weekday as a number (0-6)
getFullYear()	Get the four digit year (yyyy)
getHours()	Get the hour (0-23)
getMilliseconds()	Get the milliseconds (0-999)
getMinutes()	Get the minutes (0-59)
getMonth()	Get the month (0-11)

Date Function

```
<script type="text/javascript">
 var currDate= new Date()
var year=currDate.getFullYear()
var month=currDate.getMonth()+1
var day=currDate.getDate()

document.write("Today's date is: ")
document.write(year+"/"+month+"/"+day)
</script>
```

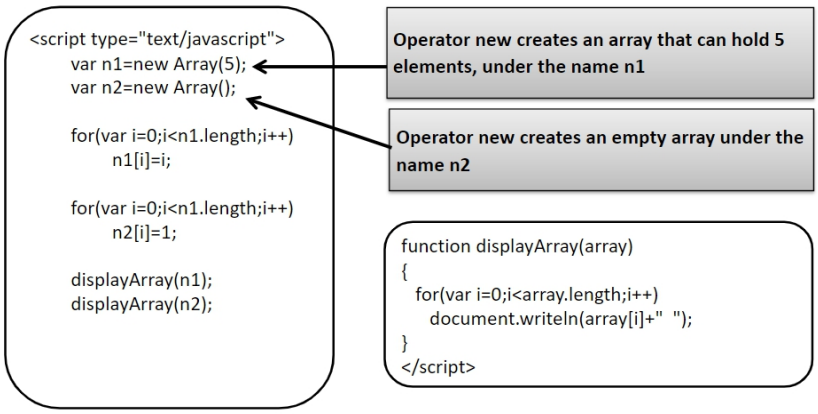
```
birthday = new Date(1998,2,14)
weekDay = birthday.getDay()

alert(weekDay) //alerts 6
```

Arrays

- Array is a data structure consisting of similar data items sharing a common name
- JavaScript arrays are “dynamic” entities where the size can be dynamically changed
- At each individual location is an element, which is accessed by its position or index.
- The first element in every array is at 0th position.
- In general, the nth element of an array c is referred to as c[n-1]

Arrays - Example




```
<script type="text/javascript">
    var n1=new Array(5);
    var n2=new Array();

    for(var i=0;i<n1.length;i++)
        n1[i]=i;

    for(var i=0;i<n1.length;i++)
        n2[i]=1;

    displayArray(n1);
    displayArray(n2);
```

```
function displayArray(array)
{
  for(var i=0;i<array.length;i++)
    document.writeln(array[i]+"  ");
}
</script>
```

Arrays

Method	Description
concat()	Joins two or more arrays, and returns a copy of the joined arrays
indexOf()	Searches the array for an element and returns its position
join()	Joins all elements of an array into a string
lastIndexOf()	Searches the array for an element, starting at the end, and returns its position
pop()	Removes the last element of an array, and returns that element
push()	Adds new elements to the end of an array, and returns the new length
reverse()	Reverses the order of the elements in an array
shift()	Removes the first element of an array, and returns that element

Arrays

```
<script type="text/javascript">
var names=["John","Pinky","George"];
document.write("<b>Concatenated Names : </b>"+names.join(" ")+"<br />");
document.write("<b>Index of \"Pinky\" : </b>"+names.indexOf("Pinky")+"<br />");
document.write("<b>Pop last element : </b>"+names.pop()+"<br />");
document.write("<b>Push \"Tom\" element at last </b>"+names.push("Tom")+"<br />");
document.write("<b>Reverse Elements : </b>"+names.reverse()+"<br />");
document.write("<b>Shift elements from 1st : </b>"+names.shift()+"<br />");
document.write("<b>Unshift elements to 1st : </b>"+names.unshift("Tom")+"<br />");
document.write("<b>Slice element from 0 to 2 : </b>"+names.slice(0,2)+"<br />");
document.write("<b>Sort Array : </b>"+names.sort()+"<br />");
document.write("<b>Splice element at pos 1 :
      </b>"+names.splice(1,0,"Prem","Caesar"));
document.write(names+"<br />");
document.write("<b>Delete an element at position 0 : </b>"+names.splice(0,1)+"<br />");
document.write("<b>Name List : </b>"+names);
</script>
```

Boolean Object

Syntax

```
var x = new Boolean(expression);
```

Function	Description
toSource	Returns a string which represents the source code of a boolean object.
toString	Returns a string representing the specified boolean object.
valueof	Returns the primitive value of a boolean object.

Initial Boolean value : false

```
new Boolean (false);
new Boolean ();
new Boolean ("");
new Boolean (0);
new Boolean (null);
```

Boolean Methods

Boolean Object

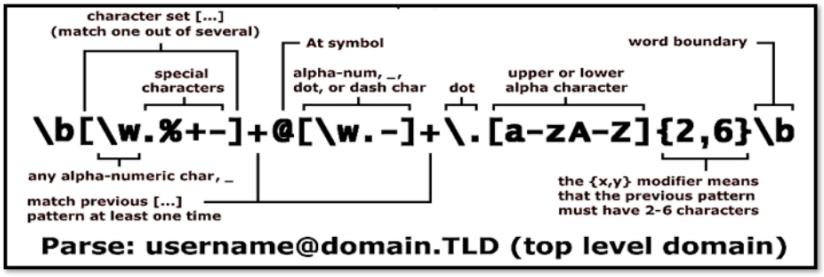
Example

```
<html>
  <head>
    <title>JavaScript toString() Method</title>
  </head>
  <body>
    <script type="text/javascript">
      var flag = new Boolean(false);
      document.write( "flag.toString is : " + flag.toString()+"<br>" );
      document.write( "flag.valueOf is : " + flag.valueOf()+"<br>");
      document.write( "flag source is : " + flag.toSource());
    </script>
  </body>
</html>
```

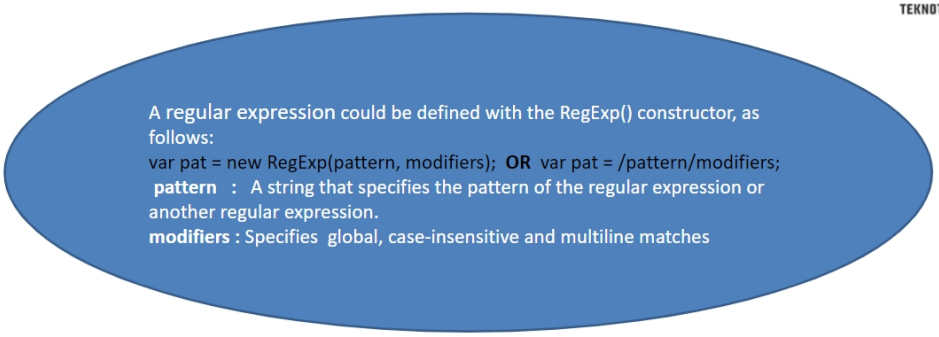
```
flag.toString is : false
flag.valueOf is : false
flag source is : (new Boolean(false))
```

RegExp Object

A regular expression is an object that describes a pattern of characters.



RegExp Object



A modifier can be : "g" – Finds all global matches
"i" – Does case-insensitive matches
"m" – Does multiline matches

RegExp Object

In Regular expressions, brackets have a special meaning.

They are used to find a range of characters.

Expression	Description
[abc]	Finds any one character between the brackets.
[^abc]	Finds any one character NOT between the brackets.
[0-9]	It matches any decimal digit from 0 through 9.
[^0-9]	It matches any decimal digit not from 0 through 9.
[a-z]	It matches any character from lowercase a through lowercase z.
[A-Z]	It matches any character from uppercase A through uppercase Z.
[a-Z]	It matches any character from lowercase a through uppercase Z.
(x/y)	Finds any of the alternatives specified . x or y

RegExp Object Methods

Here is a list of the propertaies associated with RegExp and their description.

Method	Description
exec()	Returns the first match. If no match returns null Syntax: RegExpObject.exec(string);
test()	Returns true or false Syntax: RegExpObject.test(string);
toString()	Syntax: RegExpObject.toString ( );

RegExp Object


```
<body>
<p> To do a global case-insensitive search for the characters "a","i" and
"s" in the string
<h3 style="color:red" >"I saw Susie sitting in a shoeshine shop“ </h3>
</p>
<p id="demo"></p>
<script>
   var str = "I saw Susie sitting in a shoeshine shop";
   var patt1 = /[ais]/gi;    //or var patt1 = new RegExp(/[ais]/,"gi");
   var result = str.match(patt1);
   document.getElementById("demo").innerHTML = result;
</script>
</body>
```

JavaScript Document Object Model -DOM

- Every web page displayed inside a browser window can be considered as an object.
- JavaScript arranges objects in a Document Object Model or DOM.
- The DOM defines the logical structure of objects and the way in which an object is accessed and manipulated.

JAVASCRIPT DOCUMENT OBJECT MODEL -DOM

Object	JavaScript Object Name
A frame within the browser window	frame
The history list combining the Web pages the user has already visited in the current session	history
The Web browser being run by the user	navigator
The URL of the current Web page	location
The Web page currently shown in the browser window	window
A hyperlink on the current Web page	link
A target or anchor on the current Web page	anchor
A form on the current Web page	form

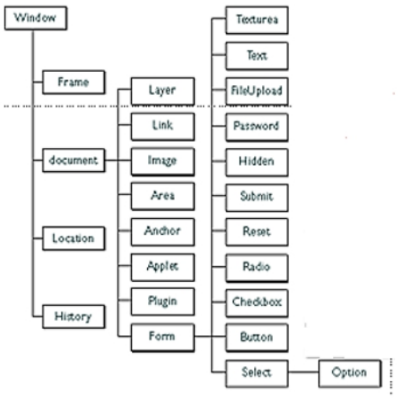
DOM Hierarchy

JavaScript uses Document Object Model(DOM) to navigate the HTML document in a hierarchy

The document object model can be thought of as a hierarchy moving from the most general object to the most specific.

Example:

To access the text field element :
document.form.text



DOM Properties

There are several ways of working with properties.
- the value of a property can be changed
- property's value can be stored in a variable
- you can test whether the property equals a specified value using an If...then expression

Some properties are **read - only**, which means you can read the property value, but cannot modify it.

The syntax for changing the value of a property is : **object.property = expression**

Window Object

Window object is the top level object in DOM

Window Object Properties:

- Window.innerHeight – represents the inner height of the browser window
- Window.innerWidth – represents the inner width of the browser window
- **Note : It works for IE, Chrome, Firefox, Opera and Safari**

Window Object Methods:

- Window.open() – opens a new window
- Window.close() – closes the current window
- Window.moveTo() – moves the current window
- Window.resizeTo() – resizes the current window

Window Object Example

```
<!DOCTYPE html>
<html>
<body>
<button onclick="openWindow()">Open gmail.com in a new  window </button>
<button onclick="closeWindow()">Close the window mail.com)</button>
<script>
var myWindow;
function openWindow()
{
   myWindow = window.open("http://www.gmail.com", "_blank", "width=500,
height=500");
}
function closeWindow() {
   myWindow.close();
}
</script>
</body>
</html>
```

_blank : Loads the URL in a new window

_self : URL replaces the current page

Window Object Example

```
<!DOCTYPE html>
<html><body>
<script>
var newWindow = window.open("","msgWindow", "width=300,height=100,
                            menubar=yes,  status=yes,resizable=yes");

newWindow.document.write("<html><head><title>New Window
</title></head><body><h1>Hello World</h1></body></html>");
</script>
</body>
</html>
```

Frame Object

It is the representation of HTML frame which belongs to a HTML frameset.

Is a property of the window object.

Properties

- frames - An array of frames in a window or frame set.
- name - Name of the frame defined using the name attribute of the <frame> tag.
- Length - Length of the frames array.
- parent - Parent of the current frame.
- self - Current frame.

Frame Object

Methods

- blur()
- focus()
- setInterval()
- clearInterval()
- setTimeout(expression, milliseconds) - Makes a timeout value and returns a timeout ID. After the number of milliseconds expire, the expression is run.
- clearTimeout(timeout)

Events

- onBlur
- onFocus

Navigator Object

It is an object that has information about the browser

Properties

- **appName :** The name of the browser ex: Mozilla Firefox
- **appVersion :** The version of the browser which may include a compatibility value and operating system name.
- **cookieEnabled :** Boolean value depending on whether cookies are enabled in the browser.
- **mimeTypes :** An array of MIME type descriptive strings that are supported by the browser. Internet Explorer supports the mimeTypes collection, but it is always empty
- **userAgent :** Describes the browser associated user agent header

Navigator Object


```
<body>
<button onclick="getBrowserName()"> Click Here to know the Navigator
                    Properties</button>
<p id="navigatorProperties"></p>

<script>
function getBrowserName() {
   var name = "Name of the browser is " + navigator.appName;
   name = name+"<br />Version info " + navigator.appVersion;
   name = name+"<br />Cookies enabled " + navigator.cookieEnabled;
   name = name+"<br />User-agent header sent: " + navigator.userAgent;

   document.getElementById(" navigatorProperties ").innerHTML = name;
}
sss
</script>
```

Navigator Object - Methods

javaEnabled()	Returns a boolean indicating whether javascript is enabled or not in the browser
taintEnabled()	Returns a boolean indicating whether the tainting is enabled or not in the browser. This method is removed in JavaScript version 1.2. Tainting is a security protection mechanism for data.

Document Object

- Becomes a document object
- Each page has only one document object
- In HTML, DOM is a node.
- Document object is the root node
- Provides properties and methods
- Document object refers the html document's <body> tag

Document Object - Properties

Property	Description
cookie	Returns the value of the cookie
domain	Returns domain name of the document server
bgColor	Sets the background color Ex : document.bgColor="#FFFFFF"
fgColor	Sets the text color attribute in the <body> tag
title	Returns the title of the page
forms	Returns an array containing an entry for each form in the document

Note: The document is a part of the Window object and can be accessed as **window.document**

Document Object - Properties

```
<body>
    <h2>Learning Objects in JavaScript</h2>
      <script type="text/javascript">
          document.title="JavaScript Object
Example";
          document.bgColor="grey";
          document.fgColor="white";
 </script>
</body>
```

Document Object - Methods

Method	Description
document.getElementById()	Finding an element by element id
document.getElementsByTagName()	Finding elements by tag name
document.getElementsByClassName()	Finding elements by class name
document.forms[]	Finding the form element with id passed as argument

Inside the document we have a form object

```
<form name="userlogin">
 User name : <input type="text" id="userId" name="userName">
</form>
```

getElementById()

getElementById() is a function that helps to access or set the document elements directly

To set the text between container tags like <p>, ,,<td> etc.

Syntax: document.geteEementById("elementId").innerText="value";

To get the text between container tags like <p>, ,,<td> etc.

Syntax: document.getElementById("elementId").value;

" elementId" represents the value of the **"id"** attribute of the form element like textbox, radio button, check box, text area, etc.

getElementById()

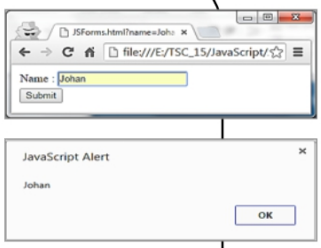
According to the DOM, we can access the value inside the textbox using JavaScript in several ways like:

```
document.getElementById("userId").value;
                    or
document.userlogin.userId.value;
                    or
document.forms["userlogin"]["userid"].value
```

Note:
id should be unique

Access Form Element By Field name

```
<head>
 <script type="text/javascript">
   function getData(){
     var name  =  document.myForm.name.value;
     alert(name);
   }
 </script>
</head>
<body>
 <form name="myForm" onSubmit="getData()">
   <table>
     <tr><td>Name :</td><td><input type="text" name="name"/></td></tr>
     <tr><td colspan="2"><input type="submit" value="Submit"/></td></tr>
   </table>
 </form>
</body>
```



getElementsByClassName()

This method is used to access the element using its class name.

Syntax: var variablename =
document.getElementsByClassName("classname")

Example: var values =
document.*getElementsByClassName("msgCheckbox")*;
//This gets all the elements with the class name "msgChecBox"

Individual elements can be accessed using index.

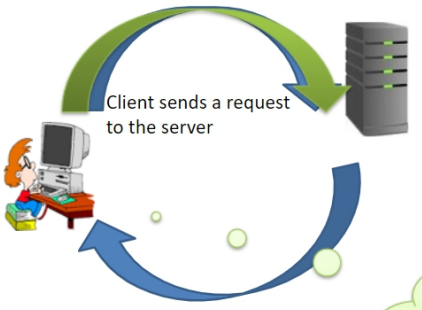
options[0] .value - represents the first element in the collection

getElementsByClassName()


```
<!DOCTYPE html>
<html>
<script>
function check()
{
    var values=document.getElementsByClassName("msgCheckbox");
    document.getElementById("div1").innerHTML="You have selected : ";
    for(var  i=0 ; I < values.length ; i++)
    {
      if(values[i].checked)
          document.getElementById("div1").innerHTML + = values[i].value+",";
    }
}
</script>
```

Cookies

Client sends a second request to the server



Server sends response back to the client

In the client machine, a text file called cookie, will hold the information about the user. Along with the request, the cookie information will be passed to the server. So the server can easily understand it is from the same client.

Now, how does the server know that the request is from the same client?

Read the cookie value

To read the cookie value :	var cookievalue=document.emailid;
To delete the cookie, set the expires parameter to a passed date as :	• document.cookie = “emailid=; expires=Mon, 27 Feb 2017 00:00:00 UTC”;

Working with Regular Expressions

Regular expressions are patterns used to match character combinations in strings.

In JavaScript, regular expressions are also objects. These patterns are used with the exec and test methods of RegExp.

Can be used with match, replace, search and split methods of string.

Quantifiers

Expression	Description
+	Represents "One or more", same as {1,}.
*	Represents "Zero or more", same as {0,}.
?	Represents "Zero or one", same as {0,1}.
$	Represents value at the end
^	Represents value at the beginning. But if it is inside [^] – Not in the given range

Quantifiers

Expression	Description
k+	It matches any string containing at least one occurrence of 'k'.
k*	It matches any string containing zero or more k's.
k?	It matches any string containing zero or one k's.
k{N}	It matches any string containing a sequence of N k's
k{2,3}	It matches any string containing a sequence of two or three k's
k{2,}	It matches any string containing a sequence of at least two k's.
k$	It matches any string with k at the end of it.
^k	It matches any string with k at the beginning of it.
?=k	Matches any string that is followed by a specific string k
?!k	Matches any string that is not followed by a specific string k

Quantifiers

Example

```
<body>
<p> To do a global search for an "e" followed by zero or more "t" in the string
<h3 style="color:red" >"The secret of getting ahead is getting started."</p>
<p id="demo" style="color:green"></p> </h3>
<script>
   var str = "The secret of getting ahead is getting started.";
   var patt1 = /et*/g;
   var result = str.match(patt1);
   document.getElementById("demo").innerHTML = result;
</script>
</body>
```

Meta characters

Expression	Description
.	Finds a single character
\s	Finds a whitespace character (space, tab, newline) .
\S	Finds a non-whitespace character
\d	Finds a digit (0-9)
\D	Finds a non-digit
\w	Finds a word character (a-z, A-Z, 0-9, _)
\W	Finds a non-word character
\b	Finds a match at the beginning/end of a word
\B	Finds a match not at the beginning/end of a word
\0	Finds a NULL character

Meta Characters - Example


```
var str="5678941982384";
   var patt1 = /[1-5]/g;
   var result = str.match(patt1);
   document.writeln("<br /> [1-5]/g in \""+str+"\" is "+result);

//output  :  5,4,1,2,3,4
-------------------------------------------------------------------------------
   var str="Hello7815";
   var patt1 = /\d/g;
   var result = str.match(patt1);
   document.writeln("<br />"+result);

//output  :  7,8,1,5
-------------------------------------------------------------------------------
   var str="People\nwho\nlive\nin\nglass\nhouses\nshouldn't\nthrow\nstones";
   var patt1 = /es$/gm;
   var result = str.match(patt1);
   document.writeln(result);

//output : es,es
```

RegExp Modifiers

Using “i” modifier to do a case–insensitive search for characters in the string

```
<body>
<p>Example to do a case-insensitive search for "Asia" in the string “ Indians are the
Italians of Asia and vice versa." .</p>
<p id="demo" style="color:red"></p>
<script>
 var str = "Indians are the Italians of Asia and vice versa.";
 var patt1  =  /asia/i ;      var result = str.match(patt1);
 if(result!=null)
document.getElementById("demo").innerHTML =  "Asia string found in
\""+str+"\"";
 else
document.getElementById("demo").innerHTML =  "Asia string not found  in
""+str+"\"";
</script></body>
```

We can also create RegExp object as:var patt1=new RegExp(“*/Asia/*\", "i")

RegExp using String methods

In JavaScript, regular expressions are often used with following **string methods**: match() , search(), replace()

The match () This method is used to retrieve the matches when matching a string against a regular expression. **Syntax:** string.match (param/regexp)

The search() method uses an expression to search for a match, and returns the position of the first occurrence of match. **Syntax:** string.search(regexp);

The replace() method returns a modified string where the pattern is replaced.

Sytax: string.replace(regexp/substr, newSubStr/function[, RegExp flags]);