Java script :-

Javascript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

Advantages of Java Script :-

- **Form validation:-** Checking the form errors Is called form validation
- **Popup ads (small window):-** the ads that will open automatically depending upon the user action
- **Dynamic pages:-** user can modify the content of the webpage at runtime.
    - (or)
    - A webpage which consist updated information

- **Less server interaction** – You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
- **Immediate feedback to the visitors** – They don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased interactivity** – You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **Richer interfaces** – You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

Control Structures

The control structures within Javascript allow the program flow to change within a unit of code or function. These statements can determine whether or not given statements are executed, as well as repeated execution of a block of code.

Contents

1. If...Else Statements

Conditional statements are used to perform different actions based on different conditions.

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

(i) If Statement

Use the if statement to execute some code only if a specified condition is true.

Syntax

```
if (condition)
{
        code to be executed if condition is true
}
```

Note that if is written in lowercase letters. Using uppercase letters (**IF**) will generate a JavaScript error!

(ii) If...else Statement

Use the if....else statement to execute some code if a condition is true and another code if the condition is not true.

Syntax

```
if (condition)
  {
        code to be executed if condition is true
  }
else
  {
        code to be executed if condition is not true
  }
```

(iii) If...else if...else Statement

Use the if....else if...else statement to select one of several blocks of code to be executed.

Syntax

```
if (condition1)
  {
        code to be executed if condition1 is true
  }
else if (condition2)
  {
        code to be executed if condition2 is true
  }
else
  {
        code to be executed if  the condition1 and condition2 is FALSE
  }
```

2. Loops

- Loops execute a block of code a specified number of times, or while a specified condition is true.
- Often when you write code, you want the same block of code to run over and over again in a row.

  Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

  In JavaScript, there are three different kind of loops:

(i) While Loop

Loops execute a block of code a specified number of times, or while a specified condition is true.

The while Loop

The while loop loops through a block of code while a specified condition is true.

Syntax

```
intialization
while (condition)
{
code to be executed
increment or decrement
}
```

Example

The example below defines a loop that starts with i=0. The loop will continue to run as long as i is less than, or equal to 5. **i** will increase by 1 each time the loop runs:

```
<script type="text/javascript">
        var i=0;
        while (i<=5)
        {
                document.write("The number is " + i);
                document.write("<br />");
                i++;
        }
</script>
```

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

(ii). do...while Loop

The do...while loop is a variant of the while loop. This loop will execute the block of code ONCE, and then it will repeat the loop as long as the specified condition is true.

Syntax

```
intialization
do
{
        code to be executed
        increment or decrement
}
while (condition);
```

Example

The example below uses a do...while loop. The do...while loop will always be executed at least once, even if the condition is false, because the statements are executed before the condition is tested:

```
<script type="text/javascript">
        var i=0;
        do
        {
        document.write("The number is " + i);
        document.write("<br />");
        i++;
```

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

```
          }
while (i<=5);
</script>
```

(ii) The for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (intialization;condition;increment or decrement)
{
        code to be executed
}
```

Example

The example below defines a loop that starts with i=0. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs.

Note: The increment parameter could also be negative, and the <= could be any comparing statement.

Example

```
<script type="text/javascript">                              The number is 0
        var i=0;                                             The number is 1
        for (i=0;i<=5;i++)                                   The number is 2
        {                                                    The number is 3
                document.write("The number is " + i);        The number is 4
                document.write("<br />");                    The number is 5
        }
</script>
```

1. **Switch Statement**

Use the switch statement to select one of many blocks of code to be executed.

Syntax

```
switch(variablename)
{
        case 1:
                execute code block 1
                break;
        case 2:
                execute code block 2
                break;
        default:
                code to be executed if variable value  is different

                from case 1 and 2
}
```

<u>**Break and Continue Statements**</u>

The break Statement

The break statement will break the loop and continue executing the code that follows after the loop (if any).

Example

```
 <script type="text/javascript">
var i=0;
for (i=0;i<=10;i++)
{
 if (i==5)
 break;
 document.write("The number is " + i);
 document.write("<br />");

}
 document.write("Hello");
</script>
```

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
Hello
```

The continue Statement

The continue statement will break the current condition and continue with the next value.

Example

```
<script type="text/javascript">
        var i=0
        for (i=0;i<=10;i++)
        {
                if (i==3)
                {
                        continue;
                }
                document.write("The number is " + i);
                document.write("<br />");
         }
</script>
```

```
The number is 0
The number is 1
The number is 2
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
The number is 10
```

JavaScript Functions

- To keep the browser from executing a script when the page loads, you can put your script into a function.
- A function contains code that will be executed by an event or by a call to the function.
- You may call a function from anywhere within a page (or even from other pages if the function is embedded in an external .js file).
- Whenever we call a function immediately it jumps to implementation part of the specifed function.
- After execution of the specified function code the next statement of the javascript code can be executed.

1. Functions (Without parameters)

- In this situation we call a method without passing any parameters.
- **Rules:** Call a function
    - Write implementation part for the function
    - When we call a function immediately it jumps to the implementation part of the function
    - After execution of the function. The next statement of the called function will be executed.

Syntax:

```
functionname();  //Function Calling

function functionname()
{
        some code;  // Implementation part
}
```

When we call function immediately it jumps to implementation part of the function.

```
<script type="text/javascript">
muni();
document.write("hello");
function muni()
{
  document.write("Function without parameter");
}
</script>
```

Output:

```
Function without parameters
hello
```

2. Functions (With parameters)

- In this situation we call a method with parameters.
- **Rules:** Call a function
    - Write implementation part for the function
    - Declare variables to hold the values of called function.
    - When we call a function immediately it jumps to the implementation part of the function
    - After execution of the function. The next statement of the called function will be executed.

```
functionname(parameter1,parameter2,.....); //Function Calling

function functionname(variablename1,variablename2,...)  // Implementation part
{
        some code;
}
```

```
<script type="text/javascript">
muni(4);
```

```
document.write("<br/>"+"hello");
function muni(a)
{
 document.write("result is"+a*a);
}
</script>
```

output

```
result is16
hello
```

3. The return Statement

- The return statement is used to specify the value that is returned from the function.
- So, functions that are going to return a value must use the return statement.
- The example below returns the product of two numbers (a and b):

Example

```
 <script type="text/javascript">
var x=muni(4,3);
document.write("Result is"+x);
function muni(a,b)
{
        return a*b;
}
</script>
```

output

```
result is 12
```

<u>Global Functions</u>

- JavaScript provides seven functions that are available globally in a JavaScript.
- The JavaScript global functions can be used with all the built-in JavaScript objects.

<u>1. **eval()** Function</u>

The **eval()** function evaluates or executes an argument.

If the argument is an expression, eval() evaluates the expression. If the argument is one or more JavaScript statements, eval() executes the statements.

```
var x = 10;
var y = 20;
var a = eval("x * y") + "<br>";
var b = eval("2 + 2") + "<br>";
var c = eval("x + 17") + "<br>";

var res = a + b + c;
```

The result of *res* will be:

```
200
4
27
```

2. **isFinite()** Function

The **isFinite()** function determines whether a number is a finite, legal number.

This function returns false if the value is +infinity, -infinity, or NaN (Not-a-Number), otherwise it returns true.

```
var a = isFinite(123) + "<br>";
var b = isFinite(-1.23) + "<br>";
var c = isFinite(5-2) + "<br>";
var d = isFinite(0) + "<br>";
var e = isFinite("Hello") + "<br>";
var f = isFinite("2005/12/12") + "<br>";

var res = a + b + c + d + e + f;
```

The result of *res* will be:

```
true
true
true
true
false
false
```

3. isNaN() Function

The isNaN() function determines whether a value is an illegal number (Not-a-Number).

This function returns true if the value is NaN, and false if not.

```
var a = isNaN(123) + "<br>";
var b = isNaN(-1.23) + "<br>";
var c = isNaN(5-2) + "<br>";
var d = isNaN(0) + "<br>";
var e = isNaN("Hello") + "<br>";
var f = isNaN("2005/12/12") + "<br>";

var res = a + b + c + d + e + f;
```

The result of *res* will be:

```
false
false
false
false
true
true
```

4.**unescape()** Function

The unescape() function decodes an encoded string.

Encode and decode a string:

```
var str = “m? Visit W3Schools!";
var str_esc = escape(str);
document.write(str_esc + "<br>")
document.write(unescape(str_esc))
```

The output of the code above will be:

```
Need%20tips%3F%20Visit%20W3Schools%21
Need tips? Visit W3Schools!
```

Math Object

- The Math object allows you to perform mathematical tasks.
- The Math object includes several mathematical properties and methods.

  Syntax for using properties/methods of Math:

  Math.property;

  Math.method();

Math object properties.

JavaScript provides eight mathematical constants that can be accessed from the Math object. [Note: By convention, the names of these constants are written in all uppercase letters.]

Constant	Description	Value
`Math.E`	Euler's constant.	Approximately 2.718.
`Math.LN2`	Natural logarithm of 2.	Approximately 0.693.
`Math.LN10`	Natural logarithm of 10.	Approximately 2.302.
`Math.LOG2E`	Base 2 logarithm of Euler's constant.	Approximately 1.442.
`Math.LOG10E`	Base 10 logarithm of Euler's constant.	Approximately 0.434.
`Math.PI`	π—the ratio of a circle's circumference to its diameter.	Approximately 3.141592653589793.
`Math.SQRT1_2`	Square root of 0.5.	Approximately 0.707.
`Math.SQRT2`	Square root of 2.0.	Approximately 1.414.

Math object Methods.

JavaScript provides mathematical methods that can be accessed from the Math object.

Method	Description	Example

`abs( x )`	absolute value of **x**	`abs( 7.2 )` is `7.2`
		`abs( 0.0 )` is `0.0`
		`abs( -5.6 )` is `5.6`

`ceil( x )`	rounds **x** to the smallest integer not less than **x**	`ceil( 9.2 )` is `10.0`
		`ceil( -9.8 )` is `-9.0`

`cos( x )`	trigonometric cosine of **x** (**x** in radians)	`cos( 0.0 )` is `1.0`

`exp( x )`	exponential method e^x	`exp( 1.0 )` is `2.71828`
		`exp( 2.0 )` is `7.38906`

`floor( x )`	rounds **x** to the largest integer not greater than **x**	`floor( 9.2 )` is `9.0`
		`floor( -9.8 )` is `-10.0`

`log( x )`	natural logarithm of **x** (base *e*)	`log( 2.718282 )` is `1.0`
		`log( 7.389056 )` is `2.0`

`max( x, y )`	larger value of **x** and **y**	`max( 2.3, 12.7 )` is `12.7`
		`max( -2.3, -12.7 )` is `-2.3`

`min( x, y )`	smaller value of **x** and **y**	`min( 2.3, 12.7 )` is `2.3`
		`min( -2.3, -12.7 )` is `-12.7`

`pow( x, y )`	**x** raised to power **y** (x^y)	`pow( 2.0, 7.0 )` is `128.0`
		`pow( 9.0, .5 )` is `3.0`

`round( x )`	rounds **x** to the closest integer	`round( 9.75 )` is `10`
		`round( 9.25 )` is `9`

`sin( x )`	trigonometric sine of **x** (**x** in radians)	`sin( 0.0 )` is `0.0`

`sqrt( x )`	square root of **x**	`sqrt( 900.0 )` is `30.0`
		`sqrt( 9.0 )` is `3.0`

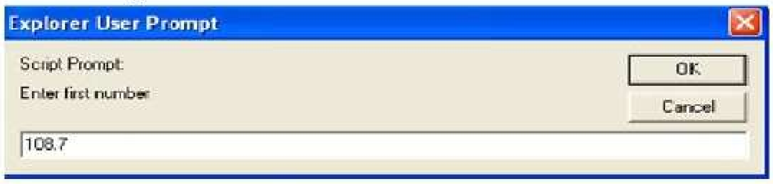
`tan( x )`	trigonometric tangent of **x** (**x** in radians)	`tan( 0.0 )` is `0.0`

Math object Example

The script in our next example (Fig. 10.3) uses a programmer-defined function called **maximum to determine and return the largest of three floating-point values.**

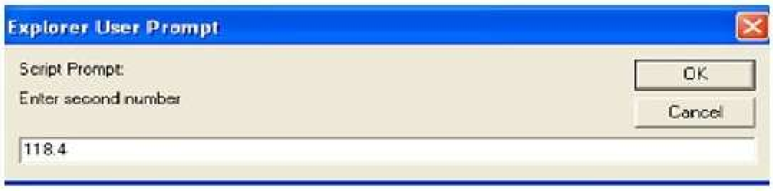
```
<script type="text/javascript">
        var x,y,z,r;
        x=parseFloat(prompt("Enter first number","0"));
        y=parseFloat(prompt("Enter second number","0"));
        z=parseFloat(prompt("Enter third number","0"));
        r=maximum(x,y,z);
        document.write( "First number: " + x +"<br />Second number: " + y +
                        "<br />Third number: " + z +"<br />Maximum is: " + r );

        function maximum(x,y,z)
        {
          return Math.max(x, Math.max(y, z));
        }
 </script>
```



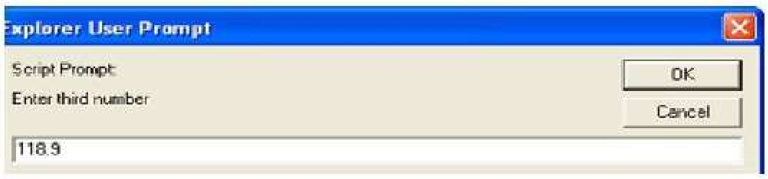
Script Prompt:
Enter first number

108.7



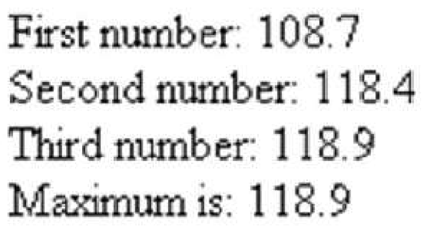
Script Prompt:
Enter second number

118.4



Script Prompt:
Enter third number

118.9



```
First number: 108.7
Second number: 118.4
Third number: 118.9
Maximum is: 118.9
```

String Object

- A string is a series of characters treated as a single unit. A string may include letters, digits and various special characters, such as **+, -, *, /, $ and others.**

Methods of the String Object

- The **String object provides many methods (behaviors) for selecting characters from a string,**
- combining strings (called concatenation), obtaining substrings of a string, searching for substrings within a string, tokenizing a string and converting strings to all uppercase or lowercase letters.

1. charAt() Method Returns the character at the specified index (position)

Syntax: string.charAt(index)

Example: Return the first character of a string:

```
var str = "HELLO WORLD";
var res = str.charAt(0)
```

The result of res will be:

H

2. charCodeAt() Method Returns the Unicode of the character at the specified index

Syntax: string.charCodeAt(index)

Example: Return the Unicode of the first character in a string (the Unicode value for "H"):

```
var str = "HELLO WORLD";
var n = str.charCodeAt(0);
```

The result of n will be:

72

3. concat() Method Joins two or more strings, and returns a new joined strings

Syntax: string.concat(string1, string2, ..., stringX)

Example: Join two strings:

```
var str1 = "Hello ";
var str2 = "world!";
var res = str1.concat(str2);
```

The result of res will be:

Hello world!

4. indexOf() Method The indexOf() method returns the position of the first occurrence of a specified value in a string.

Syntax: string.indexOf(searchvalue, start)

Parameter	Description
Searchvalue	Required. The string to search for
Start	Optional. Default 0. At which position to start the search

Example1.: Find the first occurrence of the letter "e" in a string:

```
var str = "Hello world, welcome to the universe.";
var n = str.indexOf("e");
```

The result of n will be:

1

5. lastIndexOf() Method The lastIndexOf() method returns the position of the last occurrence of a specified value in a string.

Syntax:string.lastIndexOf(searchvalue,start)

Parameter	Description
searchvalue	Required. The string to search for
start	Optional. The position where to start the search (searching backwards).

Example1: Search a string for the last occurrence of "planet":

```
var str = "Hello planet earth, you are a great p  lanet.";
var n = str.lastIndexOf("planet");
```

The result of n will be:

36

6. match() Method Searches a string for a match against a regular expression, and returns the matches

Syntax: string.match(regexp)

Example1:: Search a string for "ain":

var str = "The rain in SPAIN stays mainly in the plain";
var res = str.match(/ain/g);   where g means (**global search**)

The result of res will be an array with the values:

ain,ain,ain

- In the above example we have uppercase letters AIN that is not displayed on the screen.

Note: To search one (or) more uppercase letters use /searchtext/gi

7. replace() Method The replace() method searches a string for a specified value, or a regular expressio and returns a new string where the specified values are replaced.

Syntax: string.replace(searchvalue,newvalue)

Parameter	Description
searchvalue	Required. The value, or regular expression, that will be replaced by the new value
newvalue	Required. The value to replace the searchvalue with

Example1: Return a string where "Microsoft" is replaced with "SKIIMS":

var str = "Visit Microsoft!";
var res = str.replace("Microsoft", "SKIIMS");

The result of res will be:

Visit SKIIMS!

8. search() Method Searches a string for a specified value, or regular expression, and returns the **positio** of the match

Syntax: string.search(searchvalue)

Parameter	Description
searchvalue	Required. A regular expression. A string will automatically be converted to a regular expression.

Example: Search for "W3Schools":

var str = "Visit W3Schools!";
var n = str.search("W3Schools");

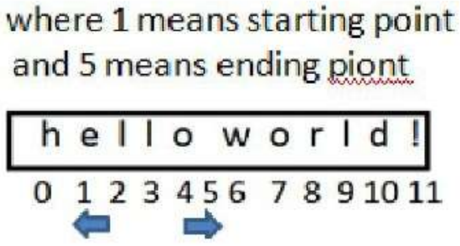
The result of n will be:

6

9. slice() method The slice() method extracts parts of a string and returns the extracted parts in a new string.

Syntax: string.slice(start,end)

Example: Extract parts of a string:



```
var str = "Hello world!";

var res = str.slice(1,5);
```

The result of res will be:

ello

10. split() Method The split() method is used to split a string into an array of substrings, and returns the new array.

Syntax: string.split(separator,limit)

Parameter	Description
separator	Optional. Specifies the character, or the regular expression, to use for splitting the string. If omitted, the entire string will be returned (an array with only one item)
limit	Optional. An integer that specifies the number of splits, items after the split limit will not be included in the array

Example1: Separate each charater by **comma** **,** including white-space:

```
var str = "How are you doing today?";
var res = str.split("");
```
{to get comma for each letter mention “” }

The result of res will be an array with the values:

H,o,w, ,a,r,e, ,y,o,u, ,d,o,i,n,g, ,t,o,d,a,y,?

11. substr() Method This method extracts parts of a string, beginning at the character at the specified position, and returns the specified number of characters.

Syntax: string.substr(start,length)

(i) start Required. The position where to start the extraction. First character is at index 0

(ii) length Optional. The number of characters to extract. If omitted, it extracts the rest of the string

Example1: Extract parts of a string:

```
var str = "Hello world!";
var res = str.substr(1, 4)
```

The result of res will be:

ello

12. toLowerCase() Method

Example: Convert the string to lowercase letters:

```
var str = "Hello World!";
var res = str.toLowerCase();
```

The result of res will be:

```
hello world!
```

13. toUpperCase() Method

Example: Convert the string to uppercase letters:

```
var str = "Hello World!";
var res = str.toUpperCase();
```

The result of res will be:

```
HELLO WORLD!
```

14. trim() Method : The trim() method removes whitespace from both sides of a string.

Note: Some browsers does not support trim() method.

Syntax: string.trim()

Example: Remove whitespace from both sides of a string:

```
var str = "       Hello World!        ";
alert(str.trim());
```

The alert box will display:

```
Hello World!
```

Date Object

The Date object is used to work with dates and times.

(i) Create a Date Object

The Date object is used to work with dates and times.

Date objects are created with the Date() constructor.

Some examples of initiating a date:

```
var today = new Date()
var d1 = new Date("October 13, 1975 11:13:00")
var d2 = new Date(79,5,24)
var d3 = new Date(79,5,24,11,33,0)
```

Date getMethods (How to display date and time)

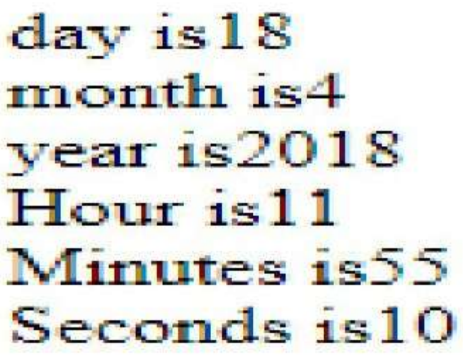
Method	Description
getDate()	Get the day as a number (1-31)
getDay()	Get the weekday as a number (0-6)
getFullYear()	Get the four digit year (yyyy)
getHours()	Get the hour (0-23)
getMilliseconds()	Get the milliseconds (0-999)
getMinutes()	Get the minutes (0-59)
getMonth()	Get the month (0-11)
getSeconds()	Get the seconds (0-59)
getTime()	Get the time (milliseconds since January 1, 1970)

Date Set Methods

Set methods are used for setting a part of a date.

Method	Description
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)

```
<script type="text/javascript">
var d=new Date();
d.setDate(18);
document.write("day is"+d.getDate()+"<br/>");
d.setMonth(4);
document.write("month is"+d.getMonth()+"<br/>");
d.setFullYear(2018);
document.write("year is"+d.getFullYear()+"<br/>");
d.setHours(11);
document.write("Hour is"+d.getHours()+"<br/>");
d.setMinutes(55);
document.write("Minutes is"+d.getMinutes()+"<br/>");
d.setSeconds(10);
document.write("Seconds is"+d.getSeconds()+"<br/>");
</script>
```



```
day is18
month is4
year is2018
Hour is11
Minutes is55
Seconds is10
```

Array Object

The Array object is used to store multiple values in a single variable.

What is an Array?

- An array is a special variable, which can hold more than one value, at a time.
- If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
var car1="Saab";
var car2="Volvo";
var car3="BMW";
```

Create an Array

An array can be defined in three ways.

The following code creates an Array object called myCars:

Syntax 1:

```
var arrayname=new Array();
```

Store data into Array

We can refer to a particular element in an array by referring to the name of the array and the index number. The index number starts at 0.

```
arrayname[0]=value or string;
arrayname [1]= value or string;
arrayname [2]= value or string;
```

Display data from Array

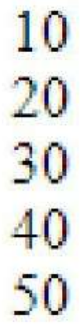
We can display the array elements by using arrayname with index position.

Syntax: document.write(arrayname[indexposition]);

Examples:

Array Creation, Storing and Displaying Example

```
<script type="text/javascript">
var a=new Array();
a[0]=10;
a[1]=20;
a[2]=30;
a[3]=40;
a[4]=50;
document.write(a[0] + "<br/>");
document.write(a[1] + "<br/>");
document.write(a[2] + "<br/>");
document.write(a[3] + "<br/>");
document.write(a[4] + "<br/>");
</script>
```



```
10
20
30
40
50
```

document object

The **document object** represents the whole html document.

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

window.document
or
document

Methods of document object

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

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

document.getElementById():

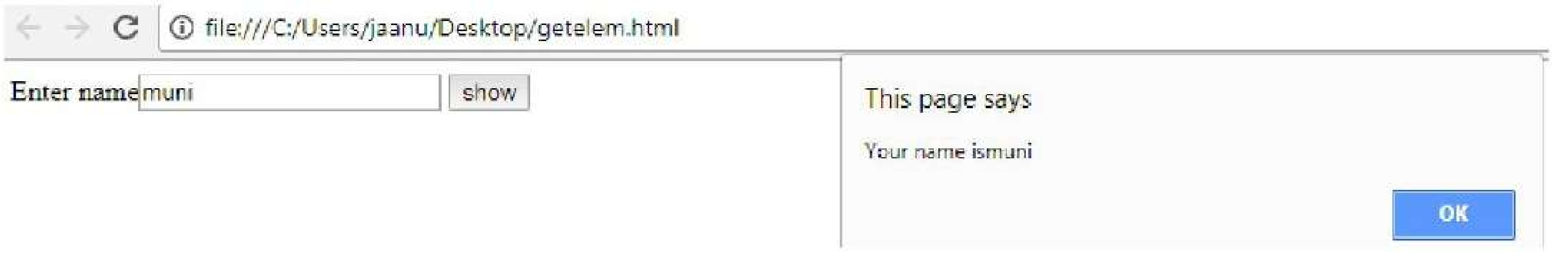
The **document.getElementById()** method returns the element of specified id.

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

Syntax: document.getElementBy("id name").value

```
Enter name<input type="text" id="i" />
<input type="button" value="show" onclick="muni();" />

<script type="text/javascript">
 function muni()
 {
  var r=document.getElementById("i").value;
   alert("Your name is"+r);
  }
</script>
```



file:///C:/Users/jaanu/Desktop/getelem.html

Enter name muni show

This page says

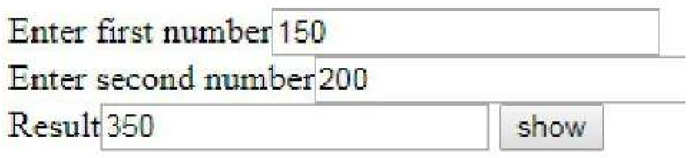
Your name ismuni

OK

Addition of two numbers using getElementById() method

```
Enter first number<input type="text" id="f" /> <br/>
Enter second number<input type="text" id="s" /> <br/>
Result<input type="text" id="r" />
<input type="button" value="show" onclick="muni();" />

<script type="text/javascript">
 function muni()
 {
  var num1=parseInt(document.getElementById("f").value);
  var num2=parseInt(document.getElementById("s").value);
  var res=num1+num2;
  document.getElementById("r").value=res;
  }
</script>
```



Enter first number 150
Enter second number 200
Result 350 show

document.getElementsByName()

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

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

Syntax: document.getElementsByName("name")

```
<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()

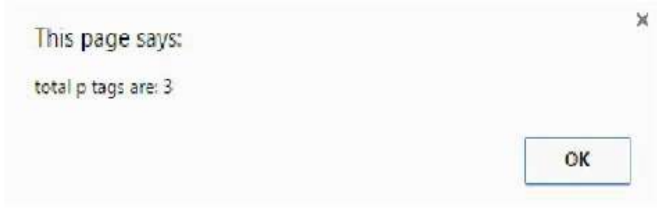
The **document.getElementsByTagName()** method returns all the element of specified tag name.

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

document.getElementsByTagName("name")

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

```
<script type="text/javascript">
function countpara()
{
var totalpara=document.getElementsByTagName("p");
alert("total p tags are: "+totalpara.length);
}
</script>
<p>This is a pragraph</p>
<p>Here we are going to count total number of paragraphs by getElementByTagName() method.</
p>
<p>Let's see the simple example</p>
<button onclick="countpara()">count paragraph</button>
```


This page says:
total p tags are: 3

document object properties

(i) innerText

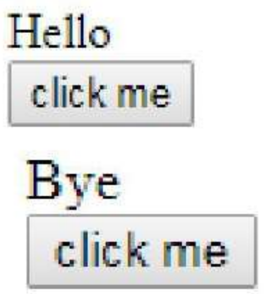
The **innerText** property can be used to write the dynamic text on the html document.

This property does not support the html tag elements by assigning the values.

It is used mostly in the web pages to generate the dynamic content such as writing the validation message, password strength etc.

Example

```
<div id="chan">Hello </div>
<input type="button" value="click me" onclick="change();"/>
<script type="text/javascript">
function change()
{
```


Hello
click me
Bye
click me