

Introduction to JavaScript

Many people *still* thinks that JavaScript is part of Java platform, which is not true

JavaScript has nothing to do with Java, only common thing between them is word "Java", much like in Car and Carpet,

The JavaScript programming language, developed by the person called Brendan Eich of Netscape. and was originally known as Live Script.

Nowadays java script not only used on client side, it is also used on server side using node.js and people are doing object oriented development. *Node.js* is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications

JavaScript is primarily used to bring interactivity into web pages, though there are other alternatives like Flash, JavaScript is the most popular one easy to use libraries like [jQuery](#) and [jQuery UI](#).

JavaScript to validate user input, create animation and cool effects in HTML page and can do lot of interactive stuff e.g. reacting on button click, mouse movement, image click etc.

How is JavaScript different from Java?

JavaScript does not create applets or stand-alone applications. In its most common form, JavaScript resides inside HTML documents, and can provide levels of interactivity to web pages that are not achievable with simple HTML. (Change the appearance of the webpage)

Key differences between Java and JavaScript:

- Java is an OOP programming language while Java Script is an OOP scripting language.
- Java creates applications that run in a virtual machine or browser while JavaScript code is run on a browser only.
- Java code needs to be compiled while JavaScript code are all in text.
- They require different plug-ins.

Variables:

- Variables are used to store data.
- A variable is a "container" for information you want to store. A variable's value can change during the script. You can refer to a variable by name to see its value or to change its value.

Simple Programs:

```
<html>

<head>

<script>

document.write("hello" + "<br/>" );

document.write("World"+ "<br/>");

var a=10;

var name="John";

document.writeln("the var value is " +a+"<br/>");

document.writeln("name is " + name);

</script>

</head>

<body>

</body>

</html>
```

Output is

```
hello
BSAu
the var value is 10
name is lisa
```

```
<SCRIPT>

alert("Hello World")

</SCRIPT>
```

```
Var no = 34.567890565
```

```
Var no = - 34444444.2323
```

```
Var no=null;
```

```
var pi = 3.14;
```

```
var person = "John Doe";
```

```
var answer = 'Yes I am!'
```

Write a program to display the output

My name is Lissy and age is 39

JavaScript Popup Boxes

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

Program:1

```
<html>
```

```
<head><title>My Page</title></head>
```

```
<body>
```

```
<script>
```

```
window.alert("Welcome to my site!")
```

```
window.confirm("Are you sure you want to quit?")
```

```
window.prompt("please enter user name")
```

```
</script>
```

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

Program:2

```
<html>
```

```
<head><title>My Page</title></head>
```

```
<body>
```

```
<script>
```

```
var x=window.confirm("Are you sure you want to quit")
```

```
if (x)
```

```
  window.alert("accepted.")
```

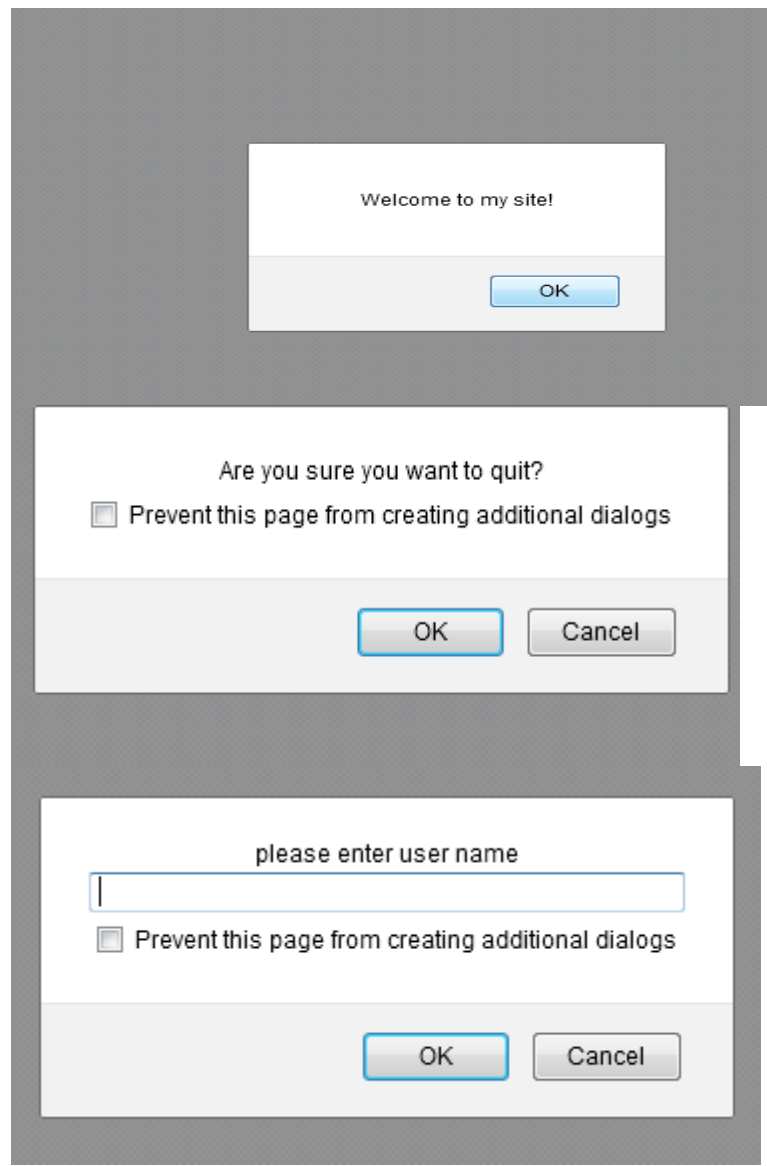
```
else
```

```
  window.alert("rejected")
```

```
</script>
```

```
</body>
```

```
</html>
```



Mathematical Operators

Arithmetic Operators:

You can use the mathematical operators with variables. The mathematical operators are these:

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic on numbers (literals or variables).

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

Operator	Example	Same As
=	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

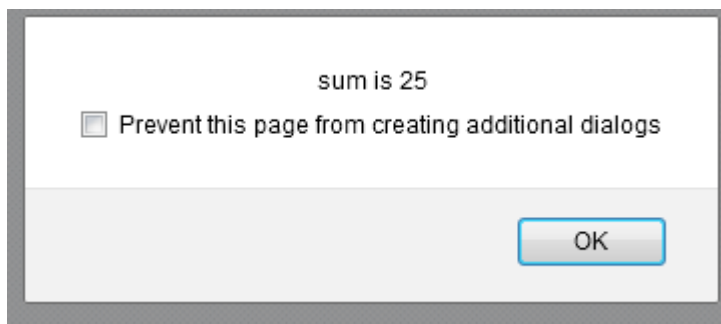
1. Write a javascript program to add two numbers:

```
<html>
```

```
<head>
```

```
<script>
```

```
var a=parseInt(prompt("enter the value for a:"));  
  
var b= parseInt(prompt("enter the value for b:"));  
  
tot =a+b;  
  
alert("sum is " + tot);  
  
</script>  
  
<head>  
  
<body bgcolor="cyan">  
  
</body>  
  
</html>
```



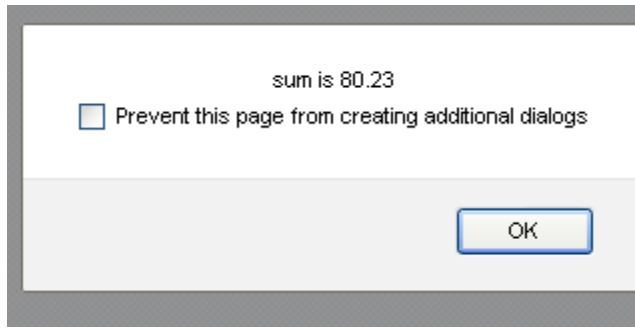
To add two float numbers:

```
<html>  
  
<head>  
  
<script>  
  
var a=parseFloat(prompt("enter the value for a:"));  
  
var b= parseFloat(prompt("enter the value for b:"));  
  
tot =a+b;  
  
alert("sum is " + tot);  
  
</script>
```

```
<head>

<body bgcolor="cyan">

</body></html>
```



Decision Making Statement

Like other languages, JavaScript supports conditional statements which are used to perform different actions based on different conditions.

JavaScript supports the following forms of if..else statement

- if statement
- if...else statement
- if...else if... statement.

Simple If Statement

if (*condition*)

```
{
    block of code to be executed if the condition is true
}
```

The block of JavaScript code will be executed if a condition is true.

Example:

```
if (age>=16)
```

```
{
```

```
alert("eligible to enter UG level");  
  
}
```

if...else statement:

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

Here JavaScript expression is evaluated. If the resulting value is true, the given statement(s) in the 'if' block, are executed. If the expression is false, then the given statement(s) in the else block are executed.

Example:

```
if (age>=16)  
  
{  
  
    alert("eligible to enter UG level");  
  
}  
  
else  
  
{  
  
    alert("eligible for school level:");  
  
}
```

Example:2

Snippet:

```
varno = 31 % 2;
```



```
if(no == 1)

{
document.write("Odd number");
}
else

{
document.write("Even number");
}
```

if...else if... statement

The **if...else if...** statement is an advanced form of **if...else** that allows JavaScript to make a correct decision out of several conditions.

```
if ( first_condition)

{

}

else if ( second_condition)

{

}
```

Example:

```
if (time <10)

{
    greeting = "Good morning";
}

elseif (time <15) {
    greeting = "Good aft";
}

else {
    greeting = "Good evening";
}
```

```
<html>
<head>
<script>
varname,mark;
name=prompt("\n enter the studentname:");
mark=parseInt(prompt("enter the mark of the student"));
if (mark>=90)
document.write("<h1> GRADE : A</h1>");
else if(mark>=80)
document.write("<h1> GRADE: B</h1>");
else if(mark>=70)
document.write("<h1> GRADE: C</h1>");
else
document.write("<h1> GRADE: D</h1>");
</script>
</head>
</html>
```

Write a javascript program to find the biggest among three numbers

```
<html>
<head>
<script>
var a=parseInt(prompt("enter the value for a:"));
var b= parseInt(prompt("enter the value for b:"));
var c=parseInt(prompt("enter the value for b:"));
if (a>b && a>c)
document.write("a is >");
else if (b>a && b>c)
document.write("b is >");
else
document.write("<h1>c is ></h1>");
</script>
</head>
<body bgcolor="cyan">
</body>
</html>
```



c is >

Switch statement - Multibranching decision making statement

The switch statement is used to perform different actions based on different conditions. The expression is evaluated, based on the result of the expression, one of many blocks of code will be executed.

How it works:

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.
- If there is no match, the default block will be executed.

Syntax:

```
switch(variable/expression)
{
case 1:
    //execute your code
    break;

case n:
    //execute your code
    break;

default:
    //execute your code
    break;
}
```

Snippet:

```
var age = 17;

switch (age)

{

case 17:

document.write("under 18");

break;

case 24:

document.write("over 18");

break;

default:

document.write("can't tell");

break;

}
```

Write a program to display the day for the value entered by the user using switch statement:

```
<html>

<head>

<script>

var val=parseInt(prompt("enter the value between [0-6]"));

switch(val)

{

case 0: alert("sunday");
```

```
break;

case 1: alert("Monday");

break;

case 2: alert("Tuesday");

break;

case 3: alert("wednesday:");

        break;


case 4:alert("Thursday:");

        break;

case 5: alert("Friday");

break;

case 6: alert("saturday:");

break;

default: document.write("enter the value bet 0 to 6");

break;

}

</script>

</head>

<body bgcolor="cyan">

</body>

</html>
```

Looping Statement

if you want to run the same code over and over again any one of the looping statement can be used. For loop, while loop , do loop and for in loop.

The While Loop

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

Syntax

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

Javascript while loops

Program to find the sum of first n numbers.

```
<html>

<head><title>My Page</title></head>

<body>

<script>

var counter = 1;

var sum=0;

while (counter < 11)

{

sum = sum + counter;

counter++;

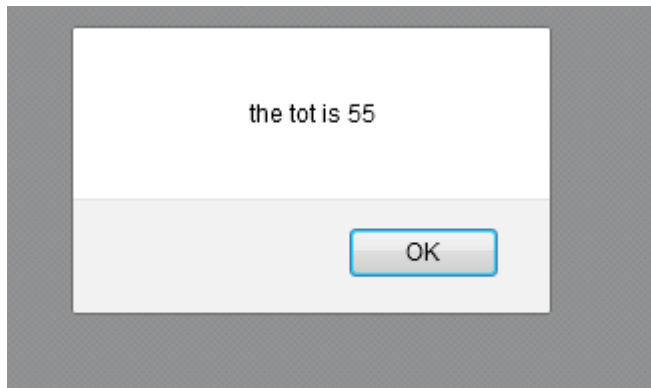
}

window.alert("the tot is "+sum)
```

</script>

</body>

</html>



For Loops

Loops can execute a block of code a number of times.

The '**for**' loop is the most compact form of looping. It includes the following three important parts

—

- The **loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
- The **test statement** which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
- The **iteration statement** where you can increase or decrease your counter.

Syntax

The syntax of for loop in JavaScript is as follows —

```
for(initialization; test condition; iteration statement)
{
Statement(s) to be executed if test condition is true
}
```


}

JavaScript Objects

Real Life Objects, Properties, and Methods

In real life, a car is an **object**.

A car has **properties** like weight and color, and **methods** like start and stop:

Object	Properties	Methods
	<code>car.name = Fiat</code> <code>car.model = 500</code> <code>car.weight = 850kg</code> <code>car.color = white</code>	<code>car.start()</code> <code>car.drive()</code> <code>car.brake()</code> <code>car.stop()</code>

Object Properties

The name:values pairs (in JavaScript objects) are called **properties**.

```
var person = { firstName:"John", lastName:"Doe", age:50, eyeColor:"blue" };
```

Accessing Object Properties

You can access object properties in two ways:

objectName.propertyName

or

objectName["propertyName"]

In JavaScript, almost "everything" is an object.

- Booleans can be objects (or primitive data treated as objects)
- Numbers can be objects (or primitive data treated as objects)
- Strings can be objects (or primitive data treated as objects)
- Dates are always objects
- Maths are always objects
- Regular expressions are always objects
- Arrays are always objects
- Functions are always objects
- Objects are objects

creating instance of Object

The syntax of creating object directly is given below:

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

```
<script>
```

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

```
</script>
```

The JavaScript for/in statement loops through the properties of an object:

Syntax:

```
for(variableinobject)
{
Statementor block to execute
}
```

In each iteration, one property from **object** is assigned to **variableinobject** and this loop continues till all the properties of the object are exhausted.

Example

Declaration syntax:

```
var person = { firstName:"Nisa", lastName:" Lee", age:40, eyeColor:"blue" };
```

The values are written as **name : value** pairs.

Program:

```
<html>

<head>

<script>

var person={ name:"pooja",age:23,addr:"sdfsdf" };

for(var i in person )

document.write("  " + person[i]);

</script>

</head>

<body>

</body>

</html>
```

Output is

pooja 23 sdfsdf

Example:2

```
<script>

var emp=new Object();
```

```
emp.id=101;

emp.name="Ravi Malik";

emp.salary=50000;

document.write(emp.id+" "+emp.name+" "+emp.salary);

</script>
```

Array

Array is a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Syntax:

Creating an Array

Using an array literal is the easiest way to create a JavaScript Array.

Syntax:

```
var array-name = [item1, item2, ...];
```

Sample Program:

```
<html>

<head>

<script>

var cars = ["Verito", "Volvo", "BMW", "Zen"];

for(var i=0; i<cars.length; i++)

alert(cars[i]);

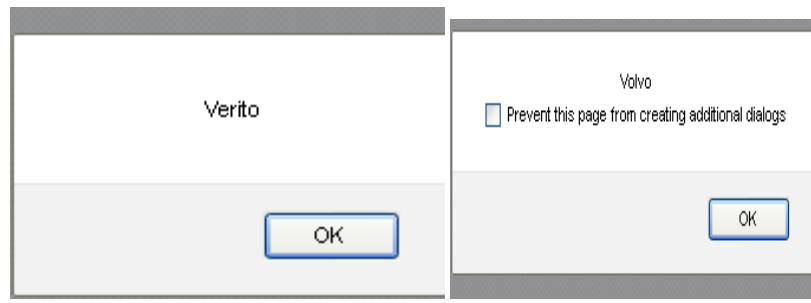
</script>
```

</head>

<body>

</body>

</html>



Access the Elements of an Array

You refer to an array element by referring to the **index number**.

```
var name = cars[0];
```

```
document.write(name[0]);
```

Adding Array Elements

The easiest way to add a new element to an array is using the push method:

```
<head>
```

```
<script>
```

```
var cars = ["Verito", "Volvo", "BMW","Zen"];
```

```
for(var i=0;i<cars.length;i++)
```

```
alert("item" +( i+1)+cars[i]);
```

```
alert("adding element to an array");
```

```
cars.push("audi");
```

```
cars.push("xylo");  
alert(cars[4]);  
alert(cars[5]);  
alert("removing elt");  
cars.pop("xylo");  
for(var i=0;i<cars.length;i++)  
alert("item" +( i+1)+cars[i]);  
</script>  
</head>  
<body>  
</body>  
</html>
```

Create an Array:

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

Sample Program: (List of numbers or names can be given as input)

```
<html>  
<head>  
<script>  
Var arr = new Array();  
for(i=0; i<5; i++)
```

```

arr.push(prompt('Enter content for item ' + i + ':', ''));

for(i=0; i<5; i++)

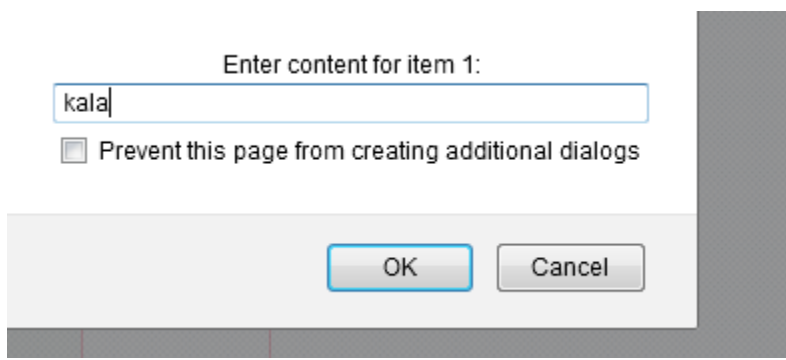
alert("item " + i+ ": " + arr[i]);

</script>

</head>

</html>

```



```

<html>
<head>
<script>
var name=["lisa","tisa","nisa","kala","pooja","laila"];
for(i=0;i<6;i++)
{
document.write(name[i]+"<br/>");
}
document.writeln(name.sort() +"<br/>");
</script>
</head>
</html>

```

```

lisa
tisa
nisa
kala
pooja

```

laila

kala,laila,lisa,nisa,pooja,tisa

Array Methods:

The `Array` object has the following methods:

1. `push()`
2. `pop()`
3. `concat()`
4. `join()`
5. `shift()`
6. `unshift()`
7. `slice()`
8. `splice()`
9. `reverse()`
10. `sort()`

`concat()` joins two arrays and returns a new array.

Snippet

```
Var myArray=newArray( "1", "2", "3");  
myArray=myArray.concat( "a", "b", "c");
```

now your array becomes

```
["1", "2", "3", "a", "b", "c"]
```

Snippet:

```
var first=["Mon","tue","wed"];
```

```
var second=["one","two","three"];
```

```
var third=["A","B","C"];
```

```
var result=first.concat(second,third);
```

Join Method -joins all elements of an array into a string.It behaves just like toString()

```
<html>

<body>

<script>

var fruits = ["Banana", "Orange", "Apple", "Mango"];

var list =fruits.join(" - ");

document.writeln(list);

</script>

</body>
```

The output is

Banana - Orange - Apple - Mango

`</html>``shift()` removes the first element from an array and returns that element.

Shift Method:removes the first element from an array and returns that element.

```
var myArray=new Array ("1","2","3");
var first =myArray.shift();
```

now array becomes ["2", "3"]

`unshift()` adds one or more elements to the front of an array and returns the new length of the array.

```
Var myArray=new Array ("1","2","3");
myArray.unshift("4","5");
```

-->now array becomes ["4", "5", "1", "2", "3"]

`slice(start_index, upto_index)` -extracts a section of an array and returns a new array. The *first element* you want to *remove is specified and the second parameter specified is the element you want.*

```
var myArray = new Array ("a", "b", "c", "d", "e");  
myArray = myArray.slice(1, 4);  
returning [ "b", "c", "d"]
```

`splice(index, count_to_remove, addElement1, addElement2, ...)`

removes elements from an array and (optionally) replaces them. The first parameter indicates the position in the array at which new elements starts and the second parameter indicates how many elements to be deleted from the original array. If you don't want to delete the elements then set this to 0.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
```

```
fruits.splice(2, 0, "Lemon", "Kiwi");
```

The output is

Banana,Orange,Lemon,Kiwi,Apple,Mango

`reverse()` transposes the elements of an array: the first array element becomes the last and the last becomes the first.

```
var myArray = new Array ("1", "2", "3");  
myArray.reverse();
```

The output is

```
[ "3", "2", "1" ]
```

sort() sorts the elements of an array.

```
var myArray = new Array("Wind", "Rain", "Fire");  
myArray.sort();  
// sorts the array so that myArray = [ "Fire", "Rain", "Wind" ]
```

Function in javascript:

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

Advantage of JavaScript function

There are mainly two advantages of JavaScript functions.

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

Syntax

The basic syntax is shown here.

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

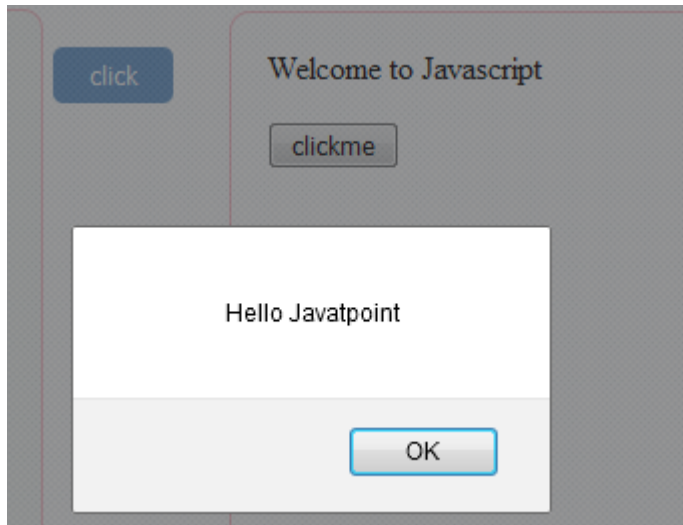
Calling a function:

```
<html>  
<head>  
<script type="text/javascript">  
function msg()
```

```

{
alert("Hello Javatpoint");
}
</script>
</head>
<body>
<p>Welcome to Javascript</p>
<form>
<input type="button" value="clickme" onclick="msg()"/>
</form>
</body>
</html>

```



Functionwith Arguments

```

<script>

function getcube(number){

alert(number*number*number);

}

</script>

<form>

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

</form>

```

Function with Return Value

```
<script>

function getInfo()

{

return "hello javatpoint! How r u?";

}
```

```
</script>
```

```
<script>

document.write(getInfo());
```

```
</script>
```

```
<html>
```

```
<head>
```

```
<script>
```

```
function test()
```

```
{
```

```
var name;
```

```
name=prompt("\n enter your name");
```

```
alert("your name is " +name);
```

```
document.write("<h1> your name is </h1>"+ "<h2>"+name+"</h2>");
```

```
}
```

```
</script>
```

```
</head>
```

```
<body onLoad="test()">
```

</body>

</html>

<html>

<body>

<p id="demo"></p>

<script>

function myFunction(a, b)

{

return a * b;

}

document.getElementById("demo").innerHTML =

myFunction(4, 3);

</script>

</body>

</html>

```
<html>
```

```
<head>
```

```
<script>
```

```
Functiongetdata()
```

```
{
```

```
var name=prompt("\n enter then name:");
```

```
varbp=parseInt(prompt("\n enter the salary:"));
```

```
var da=parseInt(prompt("\n enter the allowance:"));
```

```
var ded=parseInt(prompt("\n entre the ded:"));
```

```
cal(name,bp,da,ded);
```

```
}
```

```
Functioncal(name,bp,da,ded)
```

```
{
```

```
GP=bp+da;
```

```
NP=GP-ded;
```

```
document.write("gross pay is :"+GP+"<br/>");

document.write("net pay is :"+NP+"<br/>");

}

</script>

<body>

<p> GROSS PAY and NET PAY</p>

<form>

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

</form>

</body>
```

JavaScript String

The **JavaScript string** is an object that represents a sequence of characters.

There are 2 ways to create string in JavaScript

1. By string literal
2. By string object (using new keyword)

By string literal

The string literal is created using double quotes. The syntax of creating string using string literal is given below:

Syntax:1

```
var stringname="string value";
```

```
<script>
```

```
var str="Welcome";
```

```
document.write(str);
```

```
</script>
```

Syntax:2

The syntax of creating string object using new keyword is given below:

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

```
document.write(stringname);
```

JavaScript String Methods

Let's see the list of JavaScript string methods with examples.

- charAt(index)
- concat(str)
- indexOf(str)
- lastIndexOf(str)
- toLowerCase()
- toUpperCase()
- slice(beginIndex, endIndex)
- trim()

charAt(index) Method – Returns the character at the given index

```
<script>

var str="javascript";

document.write(str.charAt(2));

</script>
```

concat(str) Method- concatenates or joins two strings.

```
<script>

var s1="javascript ";

var s2="concat example";

var s3=s1.concat(s2);

document.write(s3);

</script>
```

Output is

javascriptconcat example

indexOf(str) Method- returns the index position of the given string.

```
<script>

var s1="javascript from javatpoint indexof";

var n=s1.indexOf("from");

document.write(n);
```

</script>

lastIndexOf(str) Method-returns the last index position of the given string.

```
<script>
var s1="javascript from javatpoint indexof";
var n=s1.lastIndexOf("java");
document.write(n);
</script>
```

Output is :16

toLowerCase() Method- returns the given string in lowercase letters.

```
<script>

var s1="JavaScript toLowerCase Example";

var s2=s1.toLowerCase();

document.write(s2);

</script>
```

Output is

javascripttolowercase example

toUpperCase() Method - returns the given string in uppercase letters.

```
<script>

var s1="JavaScript toUpperCase Example";

var s2=s1.toUpperCase();

document.write(s2);
```

</script>

slice(beginIndex, endIndex) Method-returns the parts of string from the given string. In slice() method, begin **Index is inclusive** and end **Index is exclusive**.

<script>

```
var s1="abcdefgh";
```

```
var s2=s1.slice(2,5);
```

```
document.write(s2);
```

</script>

Output is →cde

String trim() Method - removes leading and trailing whitespaces from the string.

<script>

```
var s1="  javascript trim  ";
```

```
var s2=s1.trim();
```

```
document.write(s2);
```

</script>

JavaScript Math Object

The **JavaScript Math Object** might not be used quite as often as the string object, but it is really useful when dealing with numbers. Sometimes we need to find the nearest integer to a number with a decimal (float). Other times, we need to do some sophisticated mathematics on them, such as powers or square roots. The math object is also where random number generation resides. Number in JavaScript are really simple and so is the Math Object.

The Math object

The built-in Math object includes mathematical constants and functions. You do not need to create the Math object before using it.

abs(x) - gets absolute value of x

Example

```
document.write(Math.abs(-1));
```

Output is : 1

ceil(x) - gets x rounded up to the nearest integer

```
document.write(Math.ceil(2.6));
```

ans is 3

floor(x) - gets x rounded up to the nearest integer

```
document.write(Math.floor(2.6));
```

ans is 2

[Round](#)

How to round a specified number to the nearest whole number

```
<html>
<body>
<script type="text/javascript">
document.write(Math.round(7.25))
</script>
</body>
</html>
```

Output is : 7

[Random number](#)

The random method returns a random number between 0 and 1

```
<html>
<body>
<script type="text/javascript">
document.write(Math.random())
</script>
</body>
</html>
```

Output is : 0.9572631977203739

[Random number from 0-10](#)

How to write a random number from 0 to 10, using the round and the random method.

```
<html>
<body>
<script type="text/javascript">
no=Math.random()*10
document.write(Math.floor(no))
</script>
</body>
</html>
```

Output is : 8 or 1 or 2....

max(x,y,z,...) - gets the highest number of the arguments

```
document.write(Math.max(2,3,4,5));
```

Program:

```
<html>
<body>
<script type="text/javascript">
document.write(Math.max(2,4))
</script>
</body>
</html>
```

min(x,y,z,...) - gets the lowest number of the arguments

```
document.write(Math.min(2,3,4,5));
```

pow(x,y) - gets x to the power of y

```
Example document.write(Math.pow(2,3));
```

sqrt(x) - gets the square root of x

```
document.write(Math.sqrt(16));
```

DATE OBJECTS

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

You create an instance of the Date object with the "new" keyword.

JavaScript date objects are obviously used to track time

To store the current date in a variable called "my_date":

```
Var my_date=new Date()  
my_date.getDate()
```

You can also write a date inside the parentheses of the Date() object, like this:

```
new Date("Month dd, yyyyhh:mm:ss")  
new Date("Month dd, yyyy")  
new Date(yy,mm,dd,hh,mm,ss)  
new Date(yy,mm,dd)  
new Date(milliseconds)
```

Example:

```
Var my_date=new Date("October 12, 1988 13:14:00")  
Var my_date=new Date("October 12, 1988")  
Var my_date=new Date(88,09,12,13,14,00)  
Var my_date=new Date(88,09,12)
```

```
Var my_date=new Date(500)
```

Common Date Methods

The date object can either get or set times and can compare times.

- `getFullYear()` - gets the year (Ex. 2012)
- `getMonth()` - gets the month, where January is 0 and December is 11
- `getDate()` - gets the day of the month (1 to 31)
- `getDay()` - gets the day of the week, where Sunday is 0 and Saturday is 6
- `getHours()` - gets the hour (0 to 23)
- `getMinutes()` - gets the minutes (0 to 59)
- `getSeconds()` - gets the seconds (0 to 59)
- `getMilliseconds()` - gets the milliseconds, remember that there are 1000 milliseconds in a second

Example:1

```
<script type="text/javascript">  
var d = new Date();  
document.write(d.getDay());  
</script>
```

Output is : 1

Example:2

```
<script type="text/javascript">  
var d = new Date(2012,3,1,20,28,56,445); //year, month, day, hours, minutes,  
seconds, milliseconds  
document.write(d);  
</script>
```

Comparing Times

```

<script type="text/javascript">
var oldTime = new Date(2012,3,1,20,28,56,445);
var nowTime = new Date();
if(oldTime.getTime() <= nowTime.getTime())
{
document.write("nowTime is greater than oldTime");
}
else
{
document.write("oldTime is greater than nowTime");
}
</script>

```

Output is nowTime is greater than oldTime
Display weekday

A simple script that allows you to write the name of the current day instead of the number. Note that the array object is used to store the names, and that Sunday=0, Monday=1 etc.

```

<html>
<body>
<script type="text/javascript">
var d = new Date();
var
weekday=newArray("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
document.write("Today is " + weekday[d.getDay()])
</script>
</body>
</html>

```

[Display full date](#)

How to write a complete date with the name of the day and the name of the month.

Output is Today is Monday

```

<html>
<body>
<script type="text/javascript">

```



```

var d = new Date();
var weekday=new
Array("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturda
y")
var monthname=new
Array("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec
")
document.write(weekday[d.getDay()] + " ")
document.write(d.getDate() + ". ")
document.write(monthname[d.getMonth()] + " ")
document.write(d.getFullYear())
</script>
</body>
</html>

```

Output is: Monday 14. Mar 2016

document.getElementById

There are many ways of accessing form elements, one such method is

document.getElementById

The getElementById() method returns the element that has the ID attribute with the specified value. This method is one of the most common methods in the HTML DOM, and is used almost every time you want to manipulate, or get info from, an element on your document.

Returns *null* if no elements with the specified ID exists. An ID should be unique within a page. However, if more than one element with the specified ID exists, the getElementById() method returns the first element in the source code.

Syntax:

document.getElementById("some id"). This is defined in the HTML DOM.

```

<script type="text/javascript">
functionnotEmpty(){
    var myTextField = document.getElementById('myText');
    if(myTextField.value != "")
        alert("You entered: " + myTextField.value)
    else
        alert("Would you please enter some text?")
}

```

```
</script>
<input type='text' id='myText' />
<input type='button' onclick='notEmpty()' value='Form Checker'
/>
```

The DOM (**D**ocument **O**bject **M**odel) is the official W3C standard for accessing HTML elements.

```
<html>
<body>
<h1>Converting a date to world standard format.</h1>
<p id="demo"></p>
<script>
var d = new Date();
document.getElementById("demo").innerHTML = d.toUTCString();
</script>
</body>
</html>
```

Output is :

Converting a date to world standard format.

Mon, 14 Mar 2016 16:41:12 GMT

```
<html>
<body>
<h1>My First JavaScript</h1>
<p>Please input a number.</p>
<input id="demo" type="text">
<script>
functionmyFunction()
{
var x=document.getElementById("demo").value;
if(x==""||isNaN(x))
alert("Not Numeric");
else
alert("you are correct");
}
```

```
</script>
<button type="button" onclick="myFunction()">Click Me!</button>
</body></html>
```



```
<html>
<body>
<p id="demo">Click the button to change the color of this paragraph.</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
var x = document.getElementById("demo");
x.style.color = "red";
}
</script>
```

Click the button to change the color of this paragraph.

Try it

we can use `document.getElementById()` method to get value of the input text. But we need to define id for the input field.

```
<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>
```

Program to display digital clock:

```
<html>

<head>

<script>

Function starttime()

{

var t=new Date();

var h=t.getHours();

var m=t.getMinutes();
```

```
var s=t.getSeconds();  
m=checkTime(m);  
s=checkTime(s);  
document.getElementById('txt').innerHTML=h+":"+m+":"+s;  
setTimeout('starttime()',500);  
}
```

```
functioncheckTime(i)  
{  
if(i<10)  
i="0"+i;  
return(i);  
}
```

```
</script>
```

```
<body onload="starttime()" >
```

```
<p id='txt'></p>
```

```
</body>
```

```
</html>
```

Output is

14:38:55

JavaScript Form Validation

Form validation normally used to occur at the server, after the client had entered all the necessary data and then pressed the Submit button. If the data entered by a client was incorrect or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information. This was really a lengthy process which used to put a lot of burden on the server.

JavaScript provides a way to validate form's data on the client's computer before sending it to the web server. Form validation generally performs two functions.

- **Basic Validation** – First of all, the form must be checked to make sure all the mandatory fields are filled in. It would require just a loop through each field in the form and check for data.
- **Data Format Validation** – Secondly, the data that is entered must be checked for correct form and value. Your code must include appropriate logic to test correctness of data.
- The JavaScript provides you the facility to validate the form on the client side so processing will be fast than server-side validation. So, most of the web developers prefer JavaScript form validation.

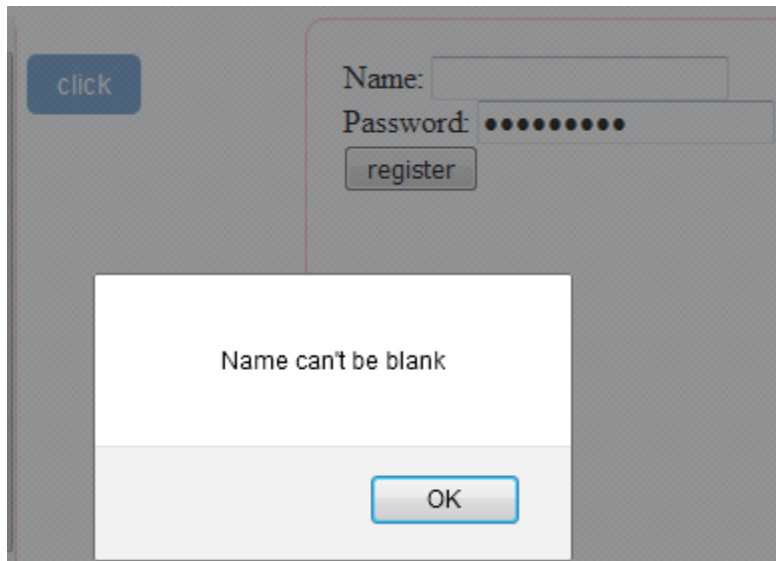
Through JavaScript, we can validate name, password, email, date, mobile number etc fields.

Program to validate the name and password. The name should not be empty and password can't be less than 6 characters long.

```
<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.asp" onsubmit="return validateform()" >
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
</form>
```



JavaScript Retype Password Validation

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

function matchpass()
{
var firstpassword=document.f1.password.value;
var secondpassword=document.f1.password2.value;

if(firstpassword==secondpassword)
{
return true;
}
else{
alert("password must be same!");
```



```
return false;
```

```
}
```

```
}
```

```
</script>
```

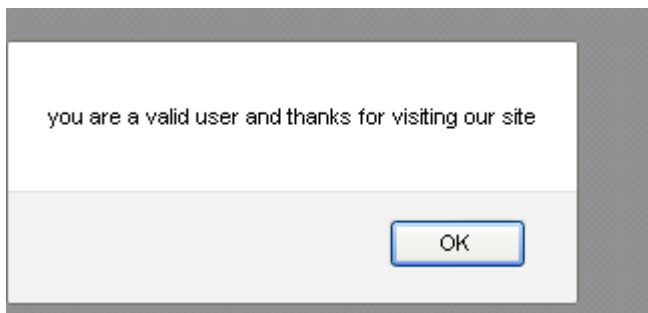
```
<form name="f1" action="register.jsp" onsubmit="return matchpass()">
```

```
Password:<input type="password" name="password" /><br/>
```

```
Re-enter Password:<input type="password" name="password2"/><br/>
```

```
<input type="submit">
```

```
</form>
```



Program for Form Validation

Name	<input type="text"/>
EMail	<input type="text"/>
Zip Code	<input type="text"/>
Country	[choose yours] ▼
	<input type="submit" value="Submit"/>

Name	<input type="text"/>
EMail	<input type="text" value="anny@gmail.com"/>
Zip Code	<input type="text" value="12345"/>
Country	<div> <div>USA ▼</div> <div> [choose yours] <div> USA UK INDIA </div> </div> </div>

```

<body>
<form action="test.asp"  name="myForm"
onsubmit="return(validate());">
<table cellpadding="2" cellspacing="2" border="1">

<tr>
<td align="right">Name</td>
<td><input type="text" name="Name" /></td>
</tr>

```

```
<tr>
<td align="right">EMail</td>
<td><input type="text" name="EMail" /></td>
</tr>
```

```
<tr>
<td align="right">Zip Code</td>
<td><input type="text" name="Zip" /></td>
</tr>
```

```
<tr>
<td align="right">Country</td>
<td>
<select name="Country">
<option value="-1" selected>[choose yours]</option>
<option value="1">USA</option>
<option value="2">UK</option>
<option value="3">INDIA</option>
</select>
</td>
</tr>
```

```
<tr>
<td align="right"></td>
<td><input type="submit" value="Submit" /></td>
</tr>
```

```
</table>
</form>
```

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

Basic Form Validation

<html>

<head>

<script type="text/javascript">

function validate()

{

if(document.myForm.Name.value == "")

{

alert("Please provide your name!");

document.myForm.Name.focus() ;

return false;

}

if(document.myForm.Email.value == "")

{

alert("Please provide your Email!");

document.myForm.Email.focus() ;

return false;

}

if(document.myForm.Zip.value == "" ||

isNaN(document.myForm.Zip.value) ||

document.myForm.Zip.value.length != 5)

{

alert("Please provide a zip in the format #####");

document.myForm.Zip.focus() ;

return false;

}

if(document.myForm.Country.value == "-1")

{

alert("Please provide your country!");

return false;

}

return(true);

}

```
</script>
</head>
</html>
```

JavaScript -Built in Object

Every web page resides inside a browser window which can be considered as an object. The way document content is accessed and modified is called the **Document Object Model**, or **DOM**. The Objects are organized in a hierarchy.

- **Window object** – Top of the hierarchy. It is the outmost element of the object hierarchy. A window object is created automatically with every instance of a <body> or <frameset> tag.
- **Navigator Object**- Contains information about the clients browser.
- **Screen**- Contains information about the clients display screen
- **History**- contains the visited URL in the browser window
- **Location**-Contains information about the current URL

Window Object:

The Window Object is the top object in the JavaScript Object hierarchy. The window object provides methods and properties for dealing with the actual browser window, including objects for each frame. The **window object** represents a window in browser. An object of window is created automatically by the browser.

Window is the object of browser, it is not the object of javascript. The javascript objects are string, array, date etc.

Methods of window object

The important methods of window object are as follows:

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.
createPopup()	Creates a pop-up window
blur()	Removes focus from the current window
focus()	Sets focus to the current window

Window Object Properties

closed	Returns a Boolean value indicating whether a window has been closed or not
defaultStatus	Sets or returns the default text in the statusbar of a window
history	The history object contains the URLs visited by the user
innerHeight	Returns the inner height of a window's content area
innerWidth	Returns the inner width of a window's content area
location	The location object contains information about the current URL.
name	Sets or returns the name of a window
navigator	Contains information about the visitor's browser.
outerHeight	Returns the outer height of a window, including toolbars/scrollbars
outerWidth	Returns the outer width of a window, including toolbars/scrollbars
pageXOffset	Returns the pixels the current document has been scrolled (horizontally) from the upper left corner of the window
pageYOffset	Returns the pixels the current document has been scrolled (vertically) from the upper left corner of the window
parent	Returns the parent window of the current window
screenLeft	Returns the horizontal coordinate of the window relative to the screen
screenTop	Returns the vertical coordinate of the window relative to the screen
screenX	Returns the horizontal coordinate of the window relative to the screen
screenY	Returns the vertical coordinate of the window relative to the screen
self	Returns the current window
status	Sets or returns the text in the statusbar of a window

top	Returns the topmost browser window
---------------------	------------------------------------

Example programs for methods in Window Object

```
<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="click me" onclick="msg()"/>
```

JavaScript History Object

The JavaScript history object represents an array of URLs visited by the user. By using this object, you can load previous, forward or any particular page.

window.history

Example of History object:

```
history.back();

history.forward();

history.go(2);

history.go(-2);
```

Methods

back() - Go to the previous URL entry in the history list. This does the same thing as the browser back button. The following HTML code creates a back button:

```
<FORM>
<INPUT TYPE="button" VALUE="Go Back" onClick="history.back()" ">
</FORM>
```

forward()- Go to the next URL entry in the history list. This does the same thing as the browser forward button.

```
<FORM>
<INPUT TYPE="button" VALUE="Go Forward"
onClick="history.forward()" ">
</FORM>
```

=====

There are several methods for move/resize a window, which is not maximized.

win.moveBy(x,y)

Moves the window relatively x pixels right and y pixels down. Negative values are accepted.

win.moveTo(x,y)

Moves the window to the given coordinates on the screen.

win.resizeBy(width,height)

Resize the window by the given width/height. Negative parameters are accepted.

win.resizeTo(width,height)

Resize the window to the given size.

JavaScript Screen Object

The **JavaScript screen object** holds information of browser screen. It can be used to display screen width, height, colorDepth, pixelDepth etc.

Property of JavaScript screen object

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.

Example of 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>
```

Output is :

```
screen.width: 1366
screen.height: 768
screen.availWidth: 1366
screen.availHeight: 728
screen.colorDepth: 24
screen.pixelDepth: 24
```

The navigator object is the window property, so it can be accessed by:

Navigator Object

The navigator object contains information about the browser.

Property	Description
appName	Returns the code name of the browser
appVersion	Returns the name of the browser
cookieEnabled	Returns the version information of the browser
geolocation	Determines whether cookies are enabled in the browser
language	Returns a Geolocation object that can be used to locate the user's position
onLine	Returns the language of the browser
platform	Determines whether the browser is online
product	Returns for which platform the browser is compiled
userAgent	Returns the engine name of the browser
	Returns the user-agent header sent by the browser to the server

Navigator Object Methods

Method	Description
javaEnabled()	Specifies whether or not the browser has Java enabled

```

<script type="text/javascript">
document.write(navigator.appCodeName + "<br>");
document.write(navigator.appName + "<br>");
document.write(navigator.appVersion + "<br>");
document.write(navigator.cookieEnabled + "<br>");
document.write(navigator.platform + "<br>");
document.write(navigator.userAgent + "<br>");
</script>

```