JavaScript is used to create client-side dynamic pages.

JavaScript is *an object-based scripting language* which is lightweight and cross-platform.

JavaScript is not a compiled language, but it is a translated language. The JavaScript Translator (embedded in the browser) is responsible for translating the JavaScript code for the web browser.

The programs in this language are called scripts. They can be written right in a web page’s HTML and run automatically as the page loads.

Scripts are provided and executed as plain text. They don’t need special preparation or compilation to run.

When JavaScript was created, it initially had another name: “LiveScript”

Today, JavaScript can execute not only in the browser, but also on the server, or actually on any device that has a special program called [the JavaScript engine](https://en.wikipedia.org/wiki/JavaScript_engine).

The browser has an embedded engine sometimes called a “JavaScript virtual machine”.

Different engines have different “codenames”. For example:

* [V8](https://en.wikipedia.org/wiki/V8_(JavaScript_engine)) – in Chrome, Opera and Edge.
* [SpiderMonkey](https://en.wikipedia.org/wiki/SpiderMonkey) – in Firefox.

**[What can in-browser JavaScript do?](https://javascript.info/intro" \l "what-can-in-browser-javascript-do)**

Modern JavaScript is a “safe” programming language. It does not provide low-level access to memory or CPU, because it was initially created for browsers which do not require it.

JavaScript’s capabilities greatly depend on the environment it’s running in. For instance, [Node.js](https://wikipedia.org/wiki/Node.js) supports functions that allow JavaScript to read/write arbitrary files, perform network requests, etc.

In-browser JavaScript can do everything related to webpage manipulation, interaction with the user, and the webserver.

For instance, in-browser JavaScript is able to:

* Add new HTML to the page, change the existing content, modify styles.
* React to user actions, run on mouse clicks, pointer movements, key presses.
* Send requests over the network to remote servers, download and upload files (so-called [AJAX](https://en.wikipedia.org/wiki/Ajax_(programming)) and [COMET](https://en.wikipedia.org/wiki/Comet_(programming)) technologies).
* Get and set cookies, ask questions to the visitor, show messages.
* Remember the data on the client-side (“local storage”).

## [What makes JavaScript unique?](https://javascript.info/intro" \l "what-makes-javascript-unique)

There are at least *three* great things about JavaScript:

* Full integration with HTML/CSS.
* Simple things are done simply.
* Supported by all major browsers and enabled by default.

JavaScript is the only browser technology that combines these three things.

That’s what makes JavaScript unique. That’s why it’s the most widespread tool for creating browser interfaces.

That said, JavaScript also allows to create servers, mobile applications, etc

Features of JavaScript

There are following features of JavaScript:

1. All popular web browsers support JavaScript as they provide built-in execution environments.
2. JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
3. JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
4. JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
5. It is a light-weighted and interpreted language.
6. It is a case-sensitive language.
7. JavaScript is supportable in several operating systems including, Windows, macOS, etc.
8. It provides good control to the users over the web browsers.

# Code editors

* [Visual Studio Code](https://code.visualstudio.com/) (cross-platform, free).
* [WebStorm](http://www.jetbrains.com/webstorm/) (cross-platform, paid).

Application of JavaScript

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

* Client-side validation,
* Dynamic drop-down menus,
* Displaying date and time,
* Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
* Displaying clocks etc.

**JAVA SCRIPT EXAMPLE**

<html>

<body>

<h2>Welcome to JavaScript</h2>

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

document.write("Hello JavaScript by JavaScript");

</script>

</body>

</html>

## [The “script” tag](https://javascript.info/hello-world" \l "the-script-tag)

JavaScript programs can be inserted almost anywhere into an HTML document using the <script> tag.

The **script** tag specifies that we are using JavaScript.

The **text/javascript** is the content type that provides information to the browser about the data.

The **document.write()** function is used to display dynamic content through JavaScript

**JavaScript provides 3 places to put the JavaScript code: within body tag, within head tag and external JavaScript file.**

3 Places to put JavaScript code

1. Between the body tag of html
2. Between the head tag of html
3. In .js file (external javaScript)

## 1) JavaScript Example : code between the body tag

In the above example, we have displayed the dynamic content using JavaScript. Let’s see the simple example of JavaScript that displays alert dialog box.

<html>

<body>

<script type="text/javascript">

alert("Hello Javatpoint");

</script>

</body>

</html>

## 2) JavaScript Example : code between the head tag

<html>

<head>

<script type="text/javascript">

function msg(){

alert("Hello Javatpoint");

}

</script>

</head>

<body>

<p>Welcome to Javascript</p>

<form>

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

</form>

</body>

</html>

In this example, we are creating a function msg(). To create function in JavaScript, you need to write function with function\_name as given below.

To call function, you need to work on event. Here we are using onclick event to call msg() function.

# **3 .External JavaScript file**

It provides **code re usability** because single JavaScript file can be used in several html pages.

An external JavaScript file must be saved by .js extension. It is recommended to embed all JavaScript files into a single file. It increases the speed of the webpage.

**message.js**

function msg(){

 alert("Hello Javatpoint");

}

Let's include the JavaScript file into [html](https://www.javatpoint.com/html-tutorial)

page. It calls the [JavaScript function](https://www.javatpoint.com/javascript-function)

on button click.

**index.html**

**<html>**

**<head>**

**<script** type="text/javascript" src="message.js"**></script>**

**</head>**

**<body>**

**<p>**Welcome to JavaScript**</p>**

**<form>**

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

**</form>**

**</body>**

**</html>**

Advantages of External JavaScript

There will be following benefits if a user creates an external javascript:

1. It helps in the reusability of code in more than one HTML file.
2. It allows easy code readability.
3. It is time-efficient as web browsers cache the external js files, which further reduces the page loading time.

The **JavaScript comments** are meaningful way to deliver message. It is used to add information about the code, warnings or suggestions so that end user can easily interpret the code.

1. **To make code easy to understand** It can be used to elaborate the code so that end user can easily understand the code.
2. **To avoid the unnecessary code** It can also be used to avoid the code being executed. Sometimes, we add the code to perform some action. But after sometime, there may be need to disable the code. In such case, it is better to use comments.

Types of JavaScript Comments

There are two types of comments in JavaScript.

1. Single-line Comment

**<script>**

// It is single line comment

document.write("hello javascript");

**</script>**

1. Multi-line Comment

**<script>**

/\* It is multi line comment.

It will not be displayed \*/

document.write("example of javascript multiline comment");

**</script>**

**variable**

A variable is just a holder of a value, like a box holding an item.

In JavaScript, you can create 3 types of variables: using const, let, and var statements. Each variable type has different behavior regarding the declaration, initialization, value access, and assignment steps.

**Var**

if (true) {

// Code block scope

var city = 'Gotham';

console.log(city); // logs 'Gotham'

}

console.log(city); // logs 'Gotham'

A **JavaScript global variable** is declared outside the function or declared with window object. It can be accessed from any function.

<html>

<body>

<script>

var value=50;//global variable

function a(){

alert(value);

}

function b(){

alert(value);

}

a();

</script>

</body>

</html>

const

const hero = 'Batman';

hero = 'Joker'; // TypeError: Assignment to constant variable

What distinguishes const variables from let and var is that you cannot assign a new value to a const variable

In the example below hero variable is declared within the scope of the if conditional block. Thus, you can access hero only within that block, but not outside:

if (true) {

// Code block scope

const hero = 'Batman';

console.log(hero); // logs 'Batman'

}

console.log(hero); // throws ReferenceError

## let

let villain = 'Joker';

villain = 'Bane';

console.log(villain); // logs 'Bane'

## You can easily update the value of a let variable, a thing you cannot do with const variables - and that's the main difference between them.

## 

## The scope of the let variables is defined the same way as for const: by a code block or function body.

function greetJoker() {

// Function scope

let villain = 'Joker';

console.log(`Hello, ${villain}!`); // logs 'Hello, Joker!'

}

console.log(`Hello, ${villain}!`); // throws ReferenceError

greetJoker();

# **Javascript Data Types**

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

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

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

var a=40;//holding number

var b="Rahul";//holding string

# **JavaScript Operators**

We know many operators from school. They are things like addition +, multiplication \*, subtraction -, and so on.

In this chapter, we’ll start with simple operators, then concentrate on JavaScript-specific aspects, not covered by school arithmetic.

## [Terms: “unary”, “binary”, “operand”](https://javascript.info/operators" \l "terms-unary-binary-operand)

Before we move on, let’s grasp some common terminology.

* An operand – is what operators are applied to. For instance, in the multiplication of 5 \* 2 there are two operands: the left operand is 5 and the right operand is 2. Sometimes, people call these “arguments” instead of “operands”.
* An operator is unary if it has a single operand. For example, the unary negation - reverses the sign of a number:

<!DOCTYPE html>

<script>

Let x= 1,y=3;

Alert(y-x);

</script>

The following math operations are supported:

* Addition +,
* Subtraction -,
* Multiplication \*,
* Division /,
* Remainder %,
* Exponentiation \*\*

<!DOCTYPE html>

<script>

alert( 5 % 2 ); // 1, a remainder of 5 divided by 2

alert( 8 % 3 ); // 2, a remainder of 8 divided by

</script>

### [Exponentiation \*\*](https://javascript.info/operators" \l "exponentiation)

alert( 2 \*\* 2 ); // 2² = 4

alert( 2 \*\* 3 ); // 2³ = 8

alert( 2 \*\* 4 ); // 2⁴ = 16

# **JavaScript If-else**

The **JavaScript if-else statement** is used *to execute the code whether condition is true or false*. There are three forms of if statement in JavaScript.

1. If Statement-It evaluates the content only if expression is true
2. If else statement-It evaluates the content whether condition is true of false.
3. if else if statement-It evaluates the content only if expression is true from several expressions.

**If anf if else Example**

<html>

<body>

<script>

var a=20;

if(a%2==0){

document.write("a is even number");

}

else{

document.write("a is odd number");

}

</script>

</body>

</html>

**If else if example**

<html>

<body>

<script>

var a=20;

if(a==10){

document.write("a is equal to 10");

}

else if(a==15){

document.write("a is equal to 15");

}

else if(a==20){

document.write("a is equal to 20");

}

else{

document.write("a is not equal to 10, 15 or 20");

}

</script>

</body>

</html>

# The "switch" statement

A switch statement can replace multiple if checks.

It gives a more descriptive way to compare a value with multiple variants.

let a = 2 + 2;

switch (a) {

case 3:

alert( 'Too small' );

break;

case 4:

alert( 'Exactly!' );

break;

case 5:

alert( 'Too big' );

break;

default:

alert( "I don't know such values" );

}

# JavaScript Loops

# **JavaScript Loops**

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

There are four types of loops in JavaScript.

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

## JavaScript For loop

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

<!DOCTYPE html>

<html>

<body>

<script>

for (i=1; i<=5; i++)

{

document.write(i + "<br/>")

}

</script>

</body>

</html>

## JavaScript while loop

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

<!DOCTYPE html>

<html>

<body>

<script>

var i=11;

while (i<=15)

{

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

i++;

}

</script>

</body>

</html>

## JavaScript do while loop

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

<!DOCTYPE html>

<html>

<body>

<script>

var i=21;

do{

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

i++;

}while (i<=25);

</script>

</body>

</html>

# **JavaScript Functions**

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

#### **Advantage of JavaScript function**

There are mainly two advantages of JavaScript functions.

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

<html>

<body>

<script>

function msg(){

alert("hello! this is message");

}

</script>

<input type="button" onclick="msg()" value="call function"/>

</body>

</html>

## JavaScript Function Arguments

We can call function by passing arguments

<html>

<body>

<script>

function msg(){

alert("hello! this is message");

}

</script>

<input type="button" onclick="msg()" value="call function"/>

</body>

</html>

# **JavaScript Objects**

A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

JavaScript is an object-based language. Everything is an object in JavaScript.

JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

## Creating Objects in JavaScript

There are 3 ways to create objects.

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

## JavaScript Object by object literal

property and value is separated by : (colon).

<html>

<body>

<script>

emp={id:102,name:"Shyam Kumar",salary:40000}

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

</script>

</body>

</html>

## By creating instance of Object

<html>

<body>

<script>

var emp=new Object();

emp.id=101;

emp.name="Ravi Malik";

emp.salary=50000;

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

</script>

</body>

</html>

## By using an Object constructor

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

<html>

<body>

<script>

function emp(id,name,salary){

this.id=id;

this.name=name;

this.salary=salary;

}

e=new emp(103,"Vimal Jaiswal",30000);

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

</script>

</body>

</html>

# **JavaScript Array**

**JavaScript array** is an object that represents a collection of similar type of elements.

There are 3 ways to construct array in JavaScript

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

## 1) JavaScript array literal

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

1. var arrayname=[value1,value2.....valueN];

As you can see, values are contained inside [ ] and separated by , (comma).

15.9M

319

History of Java

Let's see the simple example of creating and using array in JavaScript.

1. **<script>**
2. var emp=["Sonoo","Vimal","Ratan"];
3. for (i=0;i**<emp.length**;i++){
4. document.write(emp[i] + "**<br/>**");
5. }
6. **</script>**

The .length property returns the length of an array.

**Output of the above example**

Sonoo  
Vimal  
Ratan

## 2) JavaScript Array directly (new keyword)

The syntax of creating array directly is given below:

1. var arrayname=new Array();

Here, **new keyword** is used to create instance of array.

Let's see the example of creating array directly.

1. **<script>**
2. var i;
3. var emp = new Array();
4. emp[0] = "Arun";
5. emp[1] = "Varun";
6. emp[2] = "John";
8. for (i=0;i**<emp.length**;i++){
9. document.write(emp[i] + "**<br>**");
10. }
11. **</script>**

**Output of the above example**

Arun  
Varun  
John

## 3) JavaScript array constructor (new keyword)

Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

The example of creating object by array constructor is given below.

1. **<script>**
2. var emp=new Array("Jai","Vijay","Smith");
3. for (i=0;i**<emp.length**;i++){
4. document.write(emp[i] + "**<br>**");
5. }
6. **</script>**

**Output of the above example**

Jai  
Vijay  
Smith

# **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)

## 1) By string literal

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

1. var stringname="string value";

Let's see the simple example of creating string literal.

24.4M

552

Exception Handling in Java - Javatpoint

1. **<script>**
2. var str="This is string literal";
3. document.write(str);
4. **</script>**

**Output:**

This is string literal

## 2) By string object (using new keyword)

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

1. var stringname=new String("string literal");

Here, **new keyword** is used to create instance of string.

Let's see the example of creating string in JavaScript by new keyword.

1. **<script>**
2. var stringname=new String("hello javascript string");
3. document.write(stringname);
4. **</script>**

**Output:**

hello javascript string

## JavaScript String Methods

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

|  |  |
| --- | --- |
| **Methods** | **Description** |
| [charAt()](https://www.javatpoint.com/javascript-string-charat-method) | It provides the char value present at the specified index. |
| [charCodeAt()](https://www.javatpoint.com/javascript-string-charcodeat-method) | It provides the Unicode value of a character present at the specified index. |
| [concat()](https://www.javatpoint.com/javascript-string-concat-method) | It provides a combination of two or more strings. |
| [indexOf()](https://www.javatpoint.com/javascript-string-indexof-method) | It provides the position of a char value present in the given string. |
| [lastIndexOf()](https://www.javatpoint.com/javascript-string-lastindexof-method) | It provides the position of a char value present in the given string by searching a character from the last position. |
| [search()](https://www.javatpoint.com/javascript-string-search-method) | It searches a specified regular expression in a given string and returns its position if a match occurs. |
| [match()](https://www.javatpoint.com/javascript-string-match-method) | It searches a specified regular expression in a given string and returns that regular expression if a match occurs. |
| [replace()](https://www.javatpoint.com/javascript-string-replace-method) | It replaces a given string with the specified replacement. |
| [substr()](https://www.javatpoint.com/javascript-string-substr-method) | It is used to fetch the part of the given string on the basis of the specified starting position and length. |
| [substring()](https://www.javatpoint.com/javascript-string-substring-method) | It is used to fetch the part of the given string on the basis of the specified index. |
| [slice()](https://www.javatpoint.com/javascript-string-slice-method) | It is used to fetch the part of the given string. It allows us to assign positive as well negative index. |
| [toLowerCase()](https://www.javatpoint.com/javascript-string-tolowercase-method) | It converts the given string into lowercase letter. |
| [toLocaleLowerCase()](https://www.javatpoint.com/javascript-string-tolocalelowercase-method) | It converts the given string into lowercase letter on the basis of host?s current locale. |
| [toUpperCase()](https://www.javatpoint.com/javascript-string-touppercase-method) | It converts the given string into uppercase letter. |
| [toLocaleUpperCase()](https://www.javatpoint.com/javascript-string-tolocaleuppercase-method) | It converts the given string into uppercase letter on the basis of host?s current locale. |
| [toString()](https://www.javatpoint.com/javascript-string-tostring-method) | It provides a string representing the particular object. |
| [valueOf()](https://www.javatpoint.com/javascript-string-valueof-method) | It provides the primitive value of string object. |
| split() | It splits a string into substring array, then returns that newly created array. |
| trim() | It trims the white space from the left and right side of the string. |

## 1) JavaScript String charAt(index) Method

The JavaScript String charAt() method returns the character at the given index.

1. **<script>**
2. var str="javascript";
3. document.write(str.charAt(2));
4. **</script>**

**Output:**

v

## 2) JavaScript String concat(str) Method

The JavaScript String concat(str) method concatenates or joins two strings.

1. **<script>**
2. var s1="javascript ";
3. var s2="concat example";
4. var s3=s1.concat(s2);
5. document.write(s3);
6. **</script>**

**Output:**

javascript concat example

## 3) JavaScript String indexOf(str) Method

The JavaScript String indexOf(str) method returns the index position of the given string.

1. **<script>**
2. var s1="javascript from javatpoint indexof";
3. var n=s1.indexOf("from");
4. document.write(n);
5. **</script>**

**Output:**

11

## 4) JavaScript String lastIndexOf(str) Method

The JavaScript String lastIndexOf(str) method returns the last index position of the given string.

1. **<script>**
2. var s1="javascript from javatpoint indexof";
3. var n=s1.lastIndexOf("java");
4. document.write(n);
5. **</script>**

**Output:**

16

## 5) JavaScript String toLowerCase() Method

The JavaScript String toLowerCase() method returns the given string in lowercase letters.

1. **<script>**
2. var s1="JavaScript toLowerCase Example";
3. var s2=s1.toLowerCase();
4. document.write(s2);
5. **</script>**

**Output:**

javascript tolowercase example

## 6) JavaScript String toUpperCase() Method

The JavaScript String toUpperCase() method returns the given string in uppercase letters.

1. **<script>**
2. var s1="JavaScript toUpperCase Example";
3. var s2=s1.toUpperCase();
4. document.write(s2);
5. **</script>**

**Output:**

JAVASCRIPT TOUPPERCASE EXAMPLE

## 7) JavaScript String slice(beginIndex, endIndex) Method

The JavaScript String slice(beginIndex, endIndex) method returns the parts of string from given beginIndex to endIndex. In slice() method, beginIndex is inclusive and endIndex is exclusive.

1. **<script>**
2. var s1="abcdefgh";
3. var s2=s1.slice(2,5);
4. document.write(s2);
5. **</script>**

**Output:**

cde

## 8) JavaScript String trim() Method

The JavaScript String trim() method removes leading and trailing whitespaces from the string.

1. **<script>**
2. var s1="     javascript trim    ";
3. var s2=s1.trim();
4. document.write(s2);
5. **</script>**

**Output:**

javascript trim

### **9) JavaScript String split() Method**

1. **<script>**
2. var str="This is JavaTpoint website";
3. document.write(str.split(" ")); //splits the given string.
4. **</script>**

# **JavaScript Date Object**

The **JavaScript date** object can be used to get year, month and day. You can display a timer on the webpage by the help of JavaScript date object.

You can use different Date constructors to create date object. It provides methods to get and set day, month, year, hour, minute and seconds.

## Constructor

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

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

## JavaScript Date Methods

Let's see the list of JavaScript date methods with their description.

10 Sec

History of Java

|  |  |
| --- | --- |
| **Methods** | **Description** |
| [getDate()](https://www.javatpoint.com/javascript-date-getdate-method) | It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of local time. |
| [getDay()](https://www.javatpoint.com/javascript-date-getday-method) | It returns the integer value between 0 and 6 that represents the day of the week on the basis of local time. |
| [getFullYears()](https://www.javatpoint.com/javascript-date-getutcfullyear-method) | It returns the integer value that represents the year on the basis of local time. |
| [getHours()](https://www.javatpoint.com/javascript-date-gethours-method) | It returns the integer value between 0 and 23 that represents the hours on the basis of local time. |
| [getMilliseconds()](https://www.javatpoint.com/javascript-date-getmilliseconds-method) | It returns the integer value between 0 and 999 that represents the milliseconds on the basis of local time. |
| [getMinutes()](https://www.javatpoint.com/javascript-date-getminutes-method) | It returns the integer value between 0 and 59 that represents the minutes on the basis of local time. |
| [getMonth()](https://www.javatpoint.com/javascript-date-getmonth-method) | It returns the integer value between 0 and 11 that represents the month on the basis of local time. |
| [getSeconds()](https://www.javatpoint.com/javascript-date-getseconds-method) | It returns the integer value between 0 and 60 that represents the seconds on the basis of local time. |
| [getUTCDate()](https://www.javatpoint.com/javascript-date-getutcdate-method) | It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of universal time. |
| [getUTCDay()](https://www.javatpoint.com/javascript-date-getutcday-method) | It returns the integer value between 0 and 6 that represents the day of the week on the basis of universal time. |
| [getUTCFullYears()](https://www.javatpoint.com/javascript-date-getutcfullyears-method) | It returns the integer value that represents the year on the basis of universal time. |
| [getUTCHours()](https://www.javatpoint.com/javascript-date-getutchours-method) | It returns the integer value between 0 and 23 that represents the hours on the basis of universal time. |
| [getUTCMinutes()](https://www.javatpoint.com/javascript-date-getutcminutes-method) | It returns the integer value between 0 and 59 that represents the minutes on the basis of universal time. |
| [getUTCMonth()](https://www.javatpoint.com/javascript-date-getutcmonth-method) | It returns the integer value between 0 and 11 that represents the month on the basis of universal time. |
| [getUTCSeconds()](https://www.javatpoint.com/javascript-date-getutcseconds-method) | It returns the integer value between 0 and 60 that represents the seconds on the basis of universal time. |
| setDate() | It sets the day value for the specified date on the basis of local time. |
| setDay() | It sets the particular day of the week on the basis of local time. |
| setFullYears() | It sets the year value for the specified date on the basis of local time. |
| [setHours()](https://www.javatpoint.com/javascript-date-sethours-method) | It sets the hour value for the specified date on the basis of local time. |
| [setMilliseconds()](https://www.javatpoint.com/javascript-date-setmilliseconds-method) | It sets the millisecond value for the specified date on the basis of local time. |
| [setMinutes()](https://www.javatpoint.com/javascript-date-setminutes-method) | It sets the minute value for the specified date on the basis of local time. |
| setMonth() | It sets the month value for the specified date on the basis of local time. |
| [setSeconds()](https://www.javatpoint.com/javascript-date-setseconds-method) | It sets the second value for the specified date on the basis of local time. |
| [setUTCDate()](https://www.javatpoint.com/javascript-date-setutcdate-method) | It sets the day value for the specified date on the basis of universal time. |
| setUTCDay() | It sets the particular day of the week on the basis of universal time. |
| [setUTCFullYears()](https://www.javatpoint.com/javascript-date-setutcfullyear-method) | It sets the year value for the specified date on the basis of universal time. |
| [setUTCHours()](https://www.javatpoint.com/javascript-date-setutchours-method) | It sets the hour value for the specified date on the basis of universal time. |
| setUTCMilliseconds() | It sets the millisecond value for the specified date on the basis of universal time. |
| [setUTCMinutes()](https://www.javatpoint.com/javascript-date-setutcminutes-method) | It sets the minute value for the specified date on the basis of universal time. |
| [setUTCMonth()](https://www.javatpoint.com/javascript-date-setutcmonth-method) | It sets the month value for the specified date on the basis of universal time. |
| [setUTCSeconds()](https://www.javatpoint.com/javascript-date-setutcseconds-method) | It sets the second value for the specified date on the basis of universal time. |
| [toDateString()](https://www.javatpoint.com/javascript-date-todatestring-method) | It returns the date portion of a Date object. |
| [toISOString()](https://www.javatpoint.com/javascript-date-toisostring-method) | It returns the date in the form ISO format string. |
| [toJSON()](https://www.javatpoint.com/javascript-date-tojson-method) | It returns a string representing the Date object. It also serializes the Date object during JSON serialization. |
| [toString()](https://www.javatpoint.com/javascript-date-tostring-method) | It returns the date in the form of string. |
| [toTimeString()](https://www.javatpoint.com/javascript-date-totimestring-method) | It returns the time portion of a Date object. |
| [toUTCString()](https://www.javatpoint.com/javascript-date-toutcstring-method) | It converts the specified date in the form of string using UTC time zone. |
| [valueOf()](https://www.javatpoint.com/javascript-date-valueof-method) | It returns the primitive value of a Date object. |

### **JavaScript Date Example**

Let's see the simple example to print date object. It prints date and time both.

1. Current Date and Time: **<span** id="txt"**></span>**
2. **<script>**
3. var today=new Date();
4. document.getElementById('txt').innerHTML=today;
5. **</script>**

**Output:**

Current Date and Time: Wed Dec 08 2021 13:44:52 GMT+0530 (India Standard Time)

Let's see another code to print date/month/year.

1. **<script>**
2. var date=new Date();
3. var day=date.getDate();
4. var month=date.getMonth()+1;
5. var year=date.getFullYear();
6. document.write("**<br>**Date is: "+day+"/"+month+"/"+year);
7. **</script>**

**Output:**

Date is: 8/12/2021

### **JavaScript Current Time Example**

Let's see the simple example to print current time of system.

1. Current Time: **<span** id="txt"**></span>**
2. **<script>**
3. var today=new Date();
4. var h=today.getHours();
5. var m=today.getMinutes();
6. var s=today.getSeconds();
7. document.getElementById('txt').innerHTML=h+":"+m+":"+s;
8. **</script>**

**Output:**

Current Time: 13:44:52

### **JavaScript Digital Clock Example**

Let's see the simple example to display digital clock using JavaScript date object.

There are two ways to set interval in JavaScript: by setTimeout() or setInterval() method.

1. Current Time: **<span** id="txt"**></span>**
2. **<script>**
3. window.onload=function(){getTime();}
4. function getTime(){
5. var today=new Date();
6. var h=today.getHours();
7. var m=today.getMinutes();
8. var s=today.getSeconds();
9. // add a zero in front of numbers**<10**
10. m=checkTime(m);
11. s=checkTime(s);
12. document.getElementById('txt').innerHTML=h+":"+m+":"+s;
13. setTimeout(function(){getTime()},1000);
14. }
15. //setInterval("getTime()",1000);//another way
16. function checkTime(i){
17. if (i**<10**){
18. i="0" + i;
19. }
20. return i;
21. }
22. **</script>**

# **JavaScript Number Object**

The **JavaScript number** object enables you to represent a numeric value. It may be integer or floating-point. JavaScript number object follows IEEE standard to represent the floating-point numbers.

By the help of Number() constructor, you can create number object in JavaScript. For example:

1. var n=new Number(value);

If value can't be converted to number, it returns NaN(Not a Number) that can be checked by isNaN() method.

You can direct assign a number to a variable also. For example:

Exception Handling in Java - Javatpoint

1. var x=102;//integer value
2. var y=102.7;//floating point value
3. var z=13e4;//exponent value, output: 130000
4. var n=new Number(16);//integer value by number object

[**Test it Now**](https://www.javatpoint.com/oprweb/test.jsp?filename=jsnumber1)

**Output:**

102 102.7 130000 16

## JavaScript Number Constants

Let's see the list of JavaScript number constants with description.

|  |  |
| --- | --- |
| **Constant** | **Description** |
| MIN\_VALUE | returns the largest minimum value. |
| MAX\_VALUE | returns the largest maximum value. |
| POSITIVE\_INFINITY | returns positive infinity, overflow value. |
| NEGATIVE\_INFINITY | returns negative infinity, overflow value. |
| NaN | represents "Not a Number" value. |

## JavaScript Number Methods

Let's see the list of JavaScript number methods with their description.

|  |  |
| --- | --- |
| **Methods** | **Description** |
| [isFinite()](https://www.javatpoint.com/javascript-number-isfinite-method) | It determines whether the given value is a finite number. |
| [isInteger()](https://www.javatpoint.com/javascript-number-isinteger-method) | It determines whether the given value is an integer. |
| [parseFloat()](https://www.javatpoint.com/javascript-number-parsefloat-method) | It converts the given string into a floating point number. |
| [parseInt()](https://www.javatpoint.com/javascript-number-parseint-method) | It converts the given string into an integer number. |
| [toExponential()](https://www.javatpoint.com/javascript-number-toexponential-method) | It returns the string that represents exponential notation of the given number. |
| [toFixed()](https://www.javatpoint.com/javascript-number-tofixed-method) | It returns the string that represents a number with exact digits after a decimal point. |
| [toPrecision()](https://www.javatpoint.com/javascript-number-toprecision-method) | It returns the string representing a number of specified precision. |
| [toString()](https://www.javatpoint.com/javascript-number-tostring-method) | It returns the given number in the form of string. |

# **JavaScript Boolean**

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

1. Boolean b=new Boolean(value);

The default value of JavaScript Boolean object is false.

## JavaScript Boolean Example

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

## JavaScript Boolean Properties

|  |  |
| --- | --- |
| **Property** | **Description** |
| constructor | returns the reference of Boolean function that created Boolean object. |
| prototype | enables you to add properties and methods in Boolean prototype. |

## JavaScript Boolean Methods

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

# **Browser Object Model**

1. [Browser Object Model (BOM)](https://www.javatpoint.com/browser-object-model)

The **Browser Object Model** (BOM) is used to interact with the browser.

The default object of browser is window means you can call all the functions of window by specifying window or directly. For example:

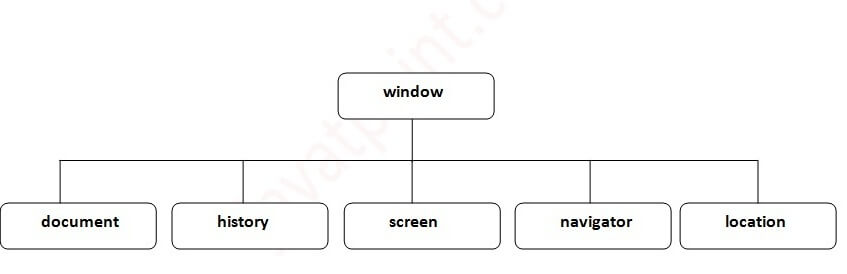
1. window.alert("hello javatpoint");

is same as:

1. alert("hello javatpoint");

You can use a lot of properties (other objects) defined underneath the window object like document, history, screen, navigator, location, innerHeight, innerWidth,

#### **Note: The document object represents an html document. It forms DOM (Document Object Model).**



Visit the next page to learn about window object fully with example.

Window Object:

The **window object** represents a window in browser. An object of window is created automatically by the browser.

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

## Methods of window object

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. |

1. **<script** type="text/javascript"**>**
2. function msg(){
3. alert("Hello Alert Box");
4. }
5. **</script>**
6. **<input** type="button" value="click" onclick="msg()"**/>**

#### **Output of the above example**

#### **Example of confirm() in javascript**

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

1. **<script** type="text/javascript"**>**
2. function msg(){
3. var v= confirm("Are u sure?");
4. if(v==true){
5. alert("ok");
6. }
7. else{
8. alert("cancel");
9. }
11. }
12. **</script>**
14. **<input** type="button" value="delete record" onclick="msg()"**/>**

#### **Output of the above example**

#### **Example of prompt() in javascript**

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

1. **<script** type="text/javascript"**>**
2. function msg(){
3. var v= prompt("Who are you?");
4. alert("I am "+v);
6. }
7. **</script>**
9. **<input** type="button" value="click" onclick="msg()"**/>**

#### **Output of the above example**

#### **Example of open() in javascript**

It displays the content in a new window.

1. **<script** type="text/javascript"**>**
2. function msg(){
3. open("http://www.javatpoint.com");
4. }
5. **</script>**
6. **<input** type="button" value="javatpoint" onclick="msg()"**/>**

#### **Output of the above example**

#### **Example of setTimeout() in javascript**

It performs its task after the given milliseconds.

1. **<script** type="text/javascript"**>**
2. function msg(){
3. setTimeout(
4. function(){
5. alert("Welcome to Javatpoint after 2 seconds")
6. },2000);
8. }
9. **</script>**
11. **<input** type="button" value="click" onclick="msg()"**/>**

#### **Output of the above example**

# **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.

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

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

1. window.history

Or,

1. history

## Methods of JavaScript history object

There are only 3 methods of history object.

|  |  |  |
| --- | --- | --- |
| **No.** | **Method** | **Description** |
| 1 | forward() | loads the next page. |
| 2 | back() | loads the previous page. |
| 3 | go() | loads the given page number. |

Example of history object

Let’s see the different usage of history object.

1. history.back();//for previous page
2. history.forward();//for next page
3. history.go(2);//for next 2nd page
4. history.go(-2);//for previous 2nd page

# **JavaScript Navigator Object:**

The **JavaScript navigator object** is used for browser detection. It can be used to get browser information such as appName, appCodeName, userAgent etc.

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

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

1. window.navigator

Or,

1. navigator

## Methods of JavaScript navigator object

The methods of navigator object are given below.

|  |  |  |
| --- | --- | --- |
| **No.** | **Method** | **Description** |
| 1 | javaEnabled() | checks if java is enabled. |
| 2 | taintEnabled() | checks if taint is enabled. It is deprecated since JavaScript 1.2. |

#### **Example of navigator object**

Let’s see the different usage of history object.

1. **<script>**
2. document.writeln("**<br/>**navigator.appCodeName: "+navigator.appCodeName);
3. document.writeln("**<br/>**navigator.appName: "+navigator.appName);
4. document.writeln("**<br/>**navigator.appVersion: "+navigator.appVersion);
5. document.writeln("**<br/>**navigator.cookieEnabled: "+navigator.cookieEnabled);
6. document.writeln("**<br/>**navigator.language: "+navigator.language);
7. document.writeln("**<br/>**navigator.userAgent: "+navigator.userAgent);
8. document.writeln("**<br/>**navigator.platform: "+navigator.platform);
9. document.writeln("**<br/>**navigator.onLine: "+navigator.onLine);
10. **</script>**

# **JavaScript Screen Object:**

1. The **JavaScript screen object** holds information of browser screen. It can be used to display screen width, height, colorDepth, pixelDepth etc.
2. The navigator object is the window property, so it can be accessed by:

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

1. window.screen

Or,

1. screen

|  |  |  |
| --- | --- | --- |
| **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 JavaScript Screen Object**

Let’s see the different usage of screen object.

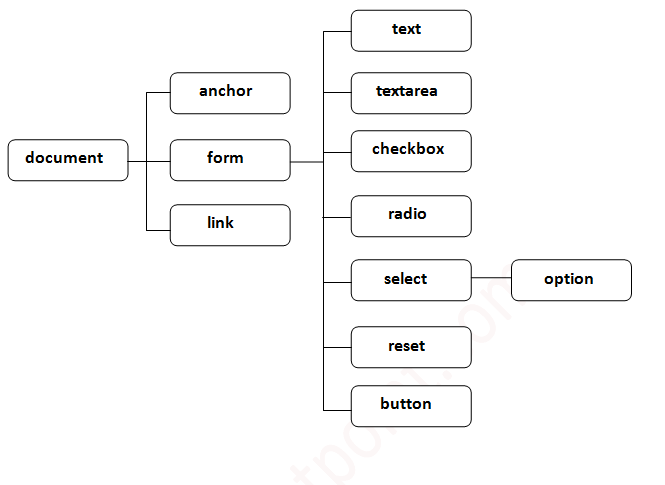
1. **<script>**
2. document.writeln("**<br/>**screen.width: "+screen.width);
3. document.writeln("**<br/>**screen.height: "+screen.height);
4. document.writeln("**<br/>**screen.availWidth: "+screen.availWidth);
5. document.writeln("**<br/>**screen.availHeight: "+screen.availHeight);
6. document.writeln("**<br/>**screen.colorDepth: "+screen.colorDepth);
7. document.writeln("**<br/>**screen.pixelDepth: "+screen.pixelDepth);
8. **</script>**

# **Document Object Model:**

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.

## Properties of document object

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

## Methods of document object

We can access and change the contents of document by its methods.

The important methods of document object are as follows:

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

1. **<script** type="text/javascript"**>**
2. function printvalue(){
3. var name=document.form1.name.value;
4. alert("Welcome: "+name);
5. }
6. **</script>**
8. **<form** name="form1"**>**
9. Enter Name:**<input** type="text" name="name"**/>**
10. **<input** type="button" onclick="printvalue()" value="print name"**/>**
11. **</form>**

In this example, we are going to get the value of input text by user. Here, we are using **document.form1.name.value** to get the value of name field.

Here, **document** is the root element that represents the html document.

**form1** is the name of the form.

**name** is the attribute name of the input text.

# **document.getElementById()**

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

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

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

1. **<script** type="text/javascript"**>**
2. function getcube(){
3. var number=document.getElementById("number").value;
4. alert(number\*number\*number);
5. }
6. **</script>**
7. **<form>**
8. Enter No:**<input** type="text" id="number" name="number"**/><br/>**
9. **<input** type="button" value="cube" onclick="getcube()"**/>**
10. **</form>**

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

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

1. document.getElementsByName("name")

Here, name is required.

### **Example of document.getElementsByName() method**

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

1. **<script** type="text/javascript"**>**
2. function totalelements()
3. {
4. var allgenders=document.getElementsByName("gender");
5. alert("Total Genders:"+allgenders.length);
6. }
7. **</script>**
8. **<form>**
9. Male:**<input** type="radio" name="gender" value="male"**>**
10. Female:**<input** type="radio" name="gender" value="female"**>**
12. **<input** type="button" onclick="totalelements()" value="Total Genders"**>**
13. **</form>**

# **GetElementsByClassName()**

The getElementsByClassName() method is used for selecting or getting the elements through their class name value

1. var ele=document.getELementsByClassName('name');
2. **<html>**
3. **<head>** **<h5>**DOM Methods **</h5>** **</head>**
4. **<body>**
5. **<div** class="Class"**>**
6. This is a simple class implementation
7. **</div>**
8. **<script** type="text/javascript"**>**
9. var x=document.getElementsByClassName('Class');
10. document.write("On calling x, it will return an arrsy-like object: **<br>**"+x);
11. **</script>**
12. **</body>**
13. **</html>**

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

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

1. document.getElementsByTagName("name")

Here, name is required.

How to find Nth Highest Salary in SQL

### **Example of document.getElementsByTagName() method**

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.

1. **<script** type="text/javascript"**>**
2. function countpara(){
3. var totalpara=document.getElementsByTagName("p");
4. alert("total p tags are: "+totalpara.length);
6. }
7. **</script>**
8. **<p>**This is a pragraph**</p>**
9. **<p>**Here we are going to count total number of paragraphs by getElementByTagName() method.**</p>**
10. **<p>**Let's see the simple example**</p>**
11. **<button** onclick="countpara()"**>**count paragraph**</button>**

innerHtml:

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

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

### **Example of innerHTML property**

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

1. **<script** type="text/javascript" **>**
2. function showcommentform() {
3. var data="Name:**<input** type='text' name='name'**><br>**Comment:**<br><textarea** rows='5' cols='80'**></textarea>**
4. **<br><input** type='submit' value='Post Comment'**>**";
5. document.getElementById('mylocation').innerHTML=data;
6. }
7. **</script>**
8. **<form** name="myForm"**>**
9. **<input** type="button" value="comment" onclick="showcommentform()"**>**
10. **<div** id="mylocation"**></div>**
11. **</form>**

Examples:

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

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

1. **<script>**
2. function validateform(){
3. var name=document.myform.name.value;
4. var password=document.myform.password.value;
6. if (name==null || name==""){
7. alert("Name can't be blank");
8. return false;
9. }else if(password.length**<6**){
10. alert("Password must be at least 6 characters long.");
11. return false;
12. }
13. }
14. **</script>**
15. **<body>**
16. **<form** name="myform" method="post" action="abc.jsp" onsubmit="return validateform()" **>**
17. Name: **<input** type="text" name="name"**><br/>**
18. Password: **<input** type="password" name="password"**><br/>**
19. **<input** type="submit" value="register"**>**
20. **</form>**

JavaScript Retype Password Validation

1. **<script** type="text/javascript"**>**
2. function matchpass(){
3. var firstpassword=document.f1.password.value;
4. var secondpassword=document.f1.password2.value;
6. if(firstpassword==secondpassword){
7. return true;
8. }
9. else{
10. alert("password must be same!");
11. return false;
12. }
13. }
14. **</script>**
16. **<form** name="f1" action="register.jsp" onsubmit="return matchpass()"**>**
17. Password:**<input** type="password" name="password" **/><br/>**
18. Re-enter Password:**<input** type="password" name="password2"**/><br/>**
19. **<input** type="submit"**>**
20. **</form>**

JavaScript Number Validation

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

1. **<script>**
2. function validate(){
3. var num=document.myform.num.value;
4. if (isNaN(num)){
5. document.getElementById("numloc").innerHTML="Enter Numeric value only";
6. return false;
7. }else{
8. return true;
9. }
10. }
11. **</script>**
12. **<form** name="myform" onsubmit="return validate()" **>**
13. Number: **<input** type="text" name="num"**><span** id="numloc"**></span><br/>**
14. **<input** type="submit" value="submit"**>**
15. **</form>**

JavaScript validation with image

Let’s see an interactive JavaScript form validation example that displays correct and incorrect image if input is correct or incorrect.

1. **<script>**
2. function validate(){
3. var name=document.f1.name.value;
4. var password=document.f1.password.value;
5. var status=false;
7. if(name.length**<1**){
8. document.getElementById("nameloc").innerHTML=
9. " <img src='unchecked.gif'/> Please enter your name";
10. status=false;
11. }else{
12. document.getElementById("nameloc").innerHTML=" <img src='checked.gif'/>";
13. status=true;
14. }
15. if(password.length**<6**){
16. document.getElementById("passwordloc").innerHTML=
17. " <img src='unchecked.gif'/> Password must be at least 6 char long";
18. status=false;
19. }else{
20. document.getElementById("passwordloc").innerHTML=" <img src='checked.gif'/>";
21. }
22. return status;
23. }
24. **</script>**
26. **<form** name="f1" action="#" onsubmit="return validate()"**>**
27. **<table>**
28. **<tr><td>**Enter Name:**</td><td><input** type="text" name="name"**/>**
29. **<span** id="nameloc"**></span></td></tr>**
30. **<tr><td>**Enter Password:**</td><td><input** type="password" name="password"**/>**
31. **<span** id="passwordloc"**></span></td></tr>**
32. **<tr><td** colspan="2"**><input** type="submit" value="register"**/></td></tr>**
33. **</table>**
34. **</form>**

JavaScript email validation

We can validate the email by the help of JavaScript.

There are many criteria that need to be follow to validate the email id such as:

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

Let's see the simple example to validate the email field.

1. **<script>**
2. function validateemail()
3. {
4. var x=document.myform.email.value;
5. var atposition=x.indexOf("@");
6. var dotposition=x.lastIndexOf(".");
7. if (atposition**<1** || dotposition**<atposition**+2 || dotposition+2**>**=x.length){
8. alert("Please enter a valid e-mail address \n atpostion:"+atposition+"\n dotposition:"+dotposition);
9. return false;
10. }
11. }
12. **</script>**
13. **<body>**
14. **<form** name="myform"  method="post" action="#" onsubmit="return validateemail();"**>**
15. Email: **<input** type="text" name="email"**><br/>**
17. **<input** type="submit" value="register"**>**
18. **</form>**