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)

## JavaScript array literal

<html>

<body>

<script>

var emp=["Sonoo","Vimal","Ratan"];

for (i=0;i<emp.length;i++){

document.write(emp[i] + "<br/>"); }

</script>

</body>

</html>

## JavaScript Array directly (new keyword)

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

<html>

<body>

<script>

var i;

var emp = new Array();

emp[0] = "Arun";

emp[1] = "Varun";

emp[2] = "John";

for (i=0;i<emp.length;i++){

document.write(emp[i] + "<br>");

}

</script>

</body>

</html>

## 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.

**<html>**

**<body>**

**<script>**

**var emp=new Array("Jai","Vijay","Smith");**

**for (i=0;i<emp.length;i++){**

**document.write(emp[i] + "<br>");**

**}**

**</script>**

**</body>**

**</html>**

# **JavaScript String**

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

## By string literal

The string literal is created using double quotes.

**<!DOCTYPE html>**

**<html>**

**<body>**

**<script>**

**var str="This is string literal";**

**document.write(str);**

**</script>**

**</body>**

**</html>**

## By string object (using new keyword)

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

**<!DOCTYPE html>**

**<html>**

**<body>**

**<script>**

**var stringname=new String("hello javascript string");**

**document.write(stringname);**

**</script>**

**</body>**

**</html>**

# **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.

**<html>**

**<body>**

**Current Date and Time: <span id="txt"></span>**

**<script>**

**var today=new Date();**

**document.getElementById('txt').innerHTML=today;**

**</script>**

**</body>**

**</html>**

# **JavaScript Math**

The **JavaScript math** object provides several constants and methods to perform mathematical operation. Unlike date object, it doesn't have constructors.

**<!DOCTYPE html>**

**<html>**

**<body>**

**Square Root of 17 is: <span id="p1"></span>**

**<script>**

**document.getElementById('p1').innerHTML=Math.sqrt(17);**

**</script>**

**</body>**

**</html>**

# **JavaScript Number Object**

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

**<!DOCTYPE html>**

**<html>**

**<body>**

**<script>**

**var x=102;//integer value**

**var y=102.7;//floating point value**

**var z=13e4;//exponent value, output: 130000**

**var n=new Number(16);//integer value by number object**

**document.write(x+" "+y+" "+z+" "+n);**

**</script>**

**</body>**

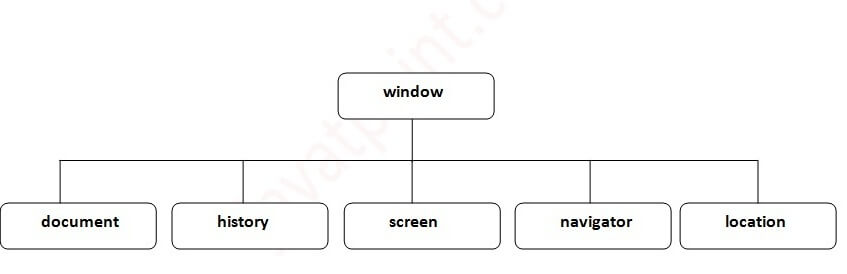
**</html>**

# **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");



# **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.

<!DOCTYPE html>

<script>

function msg(){

alert("Hello Alert Box");

}

</script>

<input type="button" value="click" 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

1. window.history

# **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.

<html>

<body>

<h2>JavaScript Navigator Object</h2>

<script>

document.writeln("<br/>navigator.appCodeName: "+navigator.appCodeName);

</script>

</body>

</html>

# **JavaScript Screen Object**

The **JavaScript screen object** holds information of browser screen

<html>

<body>

<script>

document.writeln("<br/>screen.width: "+screen.width);

document.writeln("<br/>screen.height: "+screen.height)

</script>

</body>

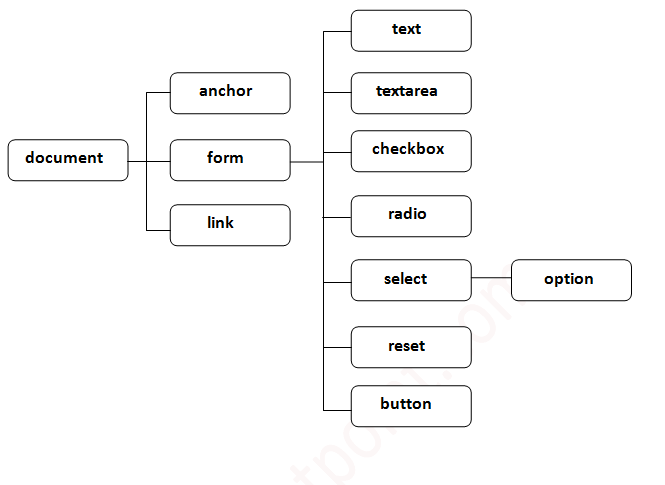
</html>

# **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.

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



# **Javascript-document.getElementById() method**

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

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

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

function getcube(){

var number=document.getElementById("number").value;

alert(number\*number\*number);

}

**</script>**

**<form>**

Enter No:**<input** type="text" id="number" name="number"**/><br/>**

**<input** type="button" value="cube" onclick="getcube()"**/>**

**</form>**

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

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

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

function totalelements()

{

var allgenders=document.getElementsByName("gender");

alert("Total Genders:"+allgenders.length);

}

**</script>**

**<form>**

Male:**<input** type="radio" name="gender" value="male"**>**

Female:**<input** type="radio" name="gender" value="female"**>**

**<input** type="button" onclick="totalelements()" value="Total Genders"**>**

**</form>**

# **document.getElementsByTagName()**

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

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

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

function countpara(){

var totalpara=document.getElementsByTagName("p");

alert("total p tags are: "+totalpara.length);

}

**</script>**

**<p>**This is a pragraph**</p>**

**<p>**Here we are going to count total number of paragraphs by getElementByTagName() method.**</p>**

**<p>**Let's see the simple example**</p>**

**<button** onclick="countpara()"**>**count paragraph**</button>**

# **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.

<html>

<body>

<script type="text/javascript" >

function showcommentform() {

var data="Name:<br><input type='text' name='name'><br>Comment:<br><textarea rows='5' cols='50'></textarea><br><input type='submit' value='comment'>";

document.getElementById('mylocation').innerHTML=data;

}

</script>

<form name="myForm">

<input type="button" value="comment" onclick="showcommentform()">

<div id="mylocation"></div>

</form>

</body>

</html>