# JavaScript Tutorial

**JavaScript Tutorial** for beginners and professionals is a solution of client side dynamic pages.

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

JavaScript is not compiled but translated. The JavaScript Translator (embedded in browser) is responsible to translate the JavaScript code.

#### Where JavaScript is used

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

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

## JavaScript Example

<html>

<body>

<h2>Welcome to JavaScript</h2>

<script>

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

</script>

</body>

</html>

Why Study JavaScript?

JavaScript is one of the **3 languages** all web developers **must** learn:

   1. **HTML** to define the content of web pages

   2. **CSS** to specify the layout of web pages

   3. **JavaScript** to program the behavior of web pages

# JavaScript Example

Javascript example is easy to code. JavaScript provides 3 places to put the JavaScript code: within body tag, within head tag and external JavaScript file.

Let’s create the first JavaScript example.

<html>

<body>

<script type="text/javascript">

document.write("JavaScript is a simple language.");

</script>

</body>

</html>

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. We will learn about document object in detail later.

## 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 JavaScript");

</script>

</body>

</html>

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

Let’s see the same example of displaying alert dialog box of JavaScript that is contained inside the head tag.

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.

<html>

<head>

<script type="text/javascript">

function msg()

{

alert("Hello JavaScript");

}

</script>

</head>

<body>

<p>Welcome to Javascript</p>

<form>

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

</form>

</body>

</html>

# External JavaScript file

We can create external JavaScript file and embed it in many html page.

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.

Let’s create an external JavaScript file that prints Hello Zerozilla in a alert dialog box.

**message.js**

function msg(){

 alert("Hello Zerozilla");

}

Let’s include the JavaScript file into html page. It calls the JavaScript function on button click.

**index.html**

<html>

<head>

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

</head>

<body>

<p>Welcome to JavaScript</p>

<form>

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

</form>

</body>

</html>

# JavaScript Comment

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.

The JavaScript comment is ignored by the JavaScript engine i.e. embedded in the browser.

#### Advantages of JavaScript comments

There are mainly two advantages of JavaScript comments.

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
2. Multi-line Comment

## JavaScript Single line Comment

It is represented by double forward slashes (//). It can be used before and after the statement.

Let’s see the example of single-line comment i.e. added before the statement.

<html>

<body>

<script>

// It is single line comment

document.write("hello javascript");

</script>

</body>

</html>

Let’s see the example of single-line comment i.e. added after the statement.

<html>

<body>

<script>

var a=10;

var b=20;

var c=a+b;//It adds values of a and b variable

document.write(c);//prints sum of 10 and 20

</script>

</body>

</html>

## JavaScript Multi line Comment

It can be used to add single as well as multi line comments. So, it is more convenient.

It is represented by forward slash with asterisk then asterisk with forward slash. For example:

/\* your code here  \*/

It can be used before, after and middle of the statement.

<html>

<body>

<script>

/\* It is multi line comment.

It will not be displayed \*/

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

</script>

</body>

</html>

# JavaScript Variable

A **JavaScript variable** is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable.

There are some rules while declaring a JavaScript variable (also known as identifiers).

1. Name must start with a letter (a to z or A to Z), underscore( \_ ), or dollar( $ ) sign.
2. After first letter we can use digits (0 to 9), for example value1.
3. JavaScript variables are case sensitive, for example x and X are different variables.

## Correct JavaScript variables

var x = 10;

var \_value="java";

## Incorrect JavaScript variables

var  123=30;

var \*aa=320;

## Example of JavaScript variable

Let’s see a simple example of JavaScript variable.

**<script>**

var x = 10;

var y = 20;

var z=x+y;

document.write(z);

**</script>**

#### Output of the above example

30

## JavaScript local variable

A JavaScript local variable is declared inside block or function. It is accessible within the function or block only. For example:

**<script>**

function abc(){

var x=10;//local variable

}

**</script>**

Or,

**<script>**

If(10**<13**){

var y=20;//JavaScript local variable

}

**</script>**

## JavaScript global variable

A **JavaScript global variable** is accessible from any function. A variable i.e. declared outside the function or declared with window object is known as global variable. For example:

**<html>**

**<body>**

**<script>**

**var data=200;//gloabal variable**

**function a(){**

**document.writeln(data);**

**}**

**function b(){**

**document.writeln(data);**

**}**

**a();//calling JavaScript function**

**b();**

**</script>**

**</body>**

**</html>**

#### To know more about global variable (e.g. how global variable is executed by JavaScript execution engine), visit next page.

# JavaScript Global Variable

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

Let’s see the simple example of global variable in JavaScript.

#### <html>

#### <body>

#### <script>

#### var value=50;//global variable

#### function a(){

#### alert(value);

#### }

#### function b(){

#### alert(value);

#### }

#### a();

#### </script>

#### </body>

#### </html>

#### Declaring JavaScript global variable within function

To declare JavaScript global variables inside function, you need to use **window object**. For example:

**Example : window.value**=90;

Now it can be declared inside any function and can be accessed from any function. For example:

<html>

<body>

<script>

function m(){

window.value=100;//declaring global variable by window object

}

function n(){

alert(window.value);//accessing global variable from other function

}

m();

n();

</script>

</body>

</html>

## Internals of global variable in JavaScript

When you declare a variable outside the function, it is added in the window object internally. You can access it through window object also. For example:

var value=50;

function a(){

alert(window.value);//accessing global variable

}

# 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 primitive data types

There are five types of primitive data types in JavaScript. They are as follows:

|  |  |
| --- | --- |
| **Data Type** | **Description** |
| String | represents sequence of characters e.g. "hello" |
| Number | represents numeric values e.g. 100 |
| Boolean | represents boolean value either false or true |
| Undefined | represents undefined value |
| Null | represents null i.e. no value at all |

## JavaScript non-primitive data types

The non-primitive data types are as follows:

|  |  |
| --- | --- |
| **Data Type** | **Description** |
| Object | represents instance through which we can access members |
| Array | represents group of similar values |
| RegExp | represents regular expression |

# JavaScript Operators

JavaScript operators are symbols that are used to perform operations on operands. For example:

var sum=10+20;

Here, + is the arithmetic operator and = is the assignment operator.

There are following types of operators in JavaScript.

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Bitwise Operators
4. Logical Operators
5. Assignment Operators
6. Special Operators

## JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on the operands. The following operators are known as JavaScript arithmetic operators.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Addition | 10+20 = 30 |
| - | Subtraction | 20-10 = 10 |
| \* | Multiplication | 10\*20 = 200 |
| / | Division | 20/10 = 2 |
| % | Modulus (Remainder) | 20%10 = 0 |
| ++ | Increment | var a=10; a++; Now a = 11 |
| -- | Decrement | var a=10; a--; Now a = 9 |

## JavaScript Comparison Operators

The JavaScript comparison operator compares the two operands. The comparison operators are as follows:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| == | Is equal to | 10==20 = false |
| === | Identical (equal and of same type) | 10==20 = false |
| != | Not equal to | 10!=20 = true |
| !== | Not Identical | 20!==20 = false |
| > | Greater than | 20>10 = true |
| >= | Greater than or equal to | 20>=10 = true |
| < | Less than | 20<10 = false |
| <= | Less than or equal to | 20<=10 = false |

## JavaScript Bitwise Operators

The bitwise operators perform bitwise operations on operands. The bitwise operators are as follows:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| & | Bitwise AND | (10==20 & 20==33) = false |
| | | Bitwise OR | (10==20 | 20==33) = false |
| ^ | Bitwise XOR | (10==20 ^ 20==33) = false |
| ~ | Bitwise NOT | (~10) = -10 |
| << | Bitwise Left Shift | (10<<2) = 40 |
| >> | Bitwise Right Shift | (10>>2) = 2 |
| >>> | Bitwise Right Shift with Zero | (10>>>2) = 2 |

## JavaScript Logical Operators

The following operators are known as JavaScript logical operators.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| && | Logical AND | (10==20 && 20==33) = false |
| || | Logical OR | (10==20 || 20==33) = false |
| ! | Logical Not | !(10==20) = true |

## JavaScript Assignment Operators

The following operators are known as JavaScript assignment operators.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Assign | 10+10 = 20 |
| += | Add and assign | var a=10; a+=20; Now a = 30 |
| -= | Subtract and assign | var a=20; a-=10; Now a = 10 |
| \*= | Multiply and assign | var a=10; a\*=20; Now a = 200 |
| /= | Divide and assign | var a=10; a/=2; Now a = 5 |
| %= | Modulus and assign | var a=10; a%=2; Now a = 0 |

## JavaScript Special Operators

The following operators are known as JavaScript special operators.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| (?:) | Conditional Operator returns value based on the condition. It is like if-else. |
| , | Comma Operator allows multiple expressions to be evaluated as single statement. |
| delete | Delete Operator deletes a property from the object. |
| in | In Operator checks if object has the given property |
| instanceof | checks if the object is an instance of given type |
| new | creates an instance (object) |
| typeof | checks the type of object. |
| void | it discards the expression's return value. |
| yield | checks what is returned in a generator by the generator's iterator. |

# 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
2. If else statement
3. if else if statement

### JavaScript If statement

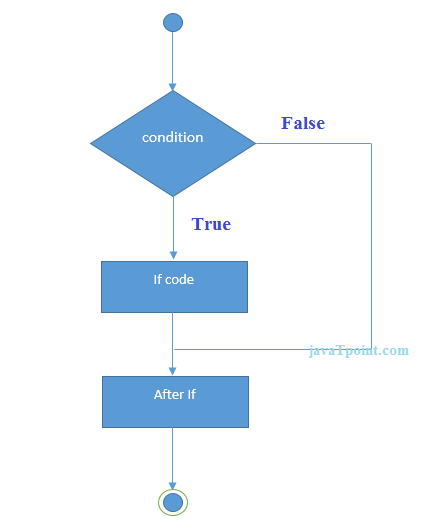
It evaluates the content only if expression is true. The signature of JavaScript if statement is given below.

if(expression){

//content to be evaluated

}

### Flowchart of JavaScript If statement



Let’s see the simple example of if statement in javascript.

#### <html>

#### <body>

#### <script>

#### var a=20;

#### if(a>10){

#### document.write("value of a is greater than 10");

#### }

#### </script>

#### </body>

#### </html>

#### Output of the above example

value of a is greater than 10

### JavaScript If...else Statement

It evaluates the content whether condition is true of false. The syntax of JavaScript if-else statement is given below.

if(expression){

//content to be evaluated if condition is true

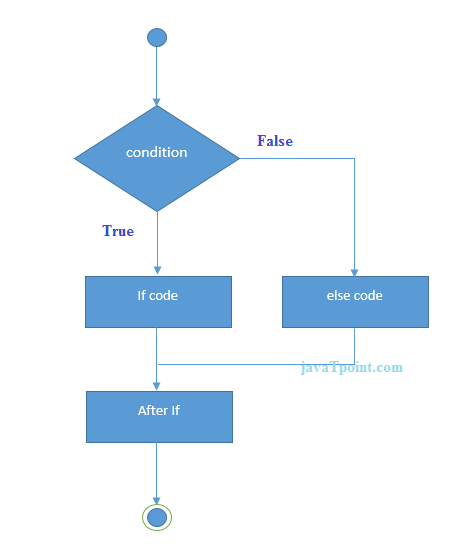
}

else{

//content to be evaluated if condition is false

}

### Flowchart of JavaScript If...else statement



Let’s see the example of if-else statement in JavaScript to find out the even or odd number.

#### <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>

#### Output of the above example

a is even number

### JavaScript If...else if statement

It evaluates the content only if expression is true from several expressions. The signature of JavaScript if else if statement is given below.

if(expression1){

//content to be evaluated if expression1 is true

}

else if(expression2){

//content to be evaluated if expression2 is true

}

else if(expression3){

//content to be evaluated if expression3 is true

}

else{

//content to be evaluated if no expression is true

}

Let’s see the simple example of if else if statement in javascript.

**<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>**

# JavaScript Switch

The **JavaScript switch statement** is used to execute one code from multiple expressions. It is just like else if statement that we have learned in previous page. But it is convenient than if..else..if because it can be used with numbers, characters etc.

The signature of JavaScript switch statement is given below.

switch(expression){

case value1:

 code to be executed;

 break;

case value2:

 code to be executed;

 break;

......

default:

 code to be executed if above values are not matched;

}

Let’s see the simple example of switch statement in javascript.

*<!DOCTYPE html>*

*<html>*

*<body>*

*<script>*

*var grade='B';*

*var result;*

*switch(grade){*

*case 'A':*

*result="A Grade";*

*break;*

*case 'B':*

*result="B Grade";*

*break;*

*case 'C':*

*result="C Grade";*

*break;*

*default:*

*result="No Grade";*

*}*

#### document.write(result);

#### </script>

#### </body>

#### </html>

#### Output of the above example

B Grade

#### The switch statement is fall-through i.e. all the cases will be evaluated if you don't use break statement.

Let’s understand the behavior of switch statement in JavaScript.

#### <!DOCTYPE html>

#### <html>

#### <body>

#### <script>

#### var grade='B';

#### var result;

#### switch(grade){

#### case 'A':

#### result+=" A Grade";

#### case 'B':

#### result+=" B Grade";

#### case 'C':

#### result+=" C Grade";

#### default:

#### result+=" No Grade";

#### }

#### document.write(result);

#### </script>

#### </body>

#### </html>

#### Output of the above example

undefined B Grade C Grade No Grade

# 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

## 1) 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.

for (initialization; condition; increment)

{

    code to be executed

}

Let’s see the simple example of for loop in javascript.

**<!DOCTYPE html>**

**<html>**

**<body>**

**<script>**

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

**{**

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

**}**

**</script>**

**</body>**

**</html>**

Output:

1  
2  
3  
4  
5

## 2) 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.

while (condition)

{

    code to be executed

}

Let’s see the simple example of while loop in javascript.

**<!DOCTYPE html>**

**<html>**

**<body>**

**<script>**

**var i=11;**

**while (i<=15)**

**{**

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

**i++;**

**}**

**</script>**

**</body>**

**</html>**

Output:

11  
12  
13  
14  
15

## 3) 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. The syntax of do while loop is given below.

do{

    code to be executed

}while (condition);

Let’s see the simple example of do while loop in javascript.

**<!DOCTYPE html>**

**<html>**

**<body>**

**<script>**

**var i=21;**

**do**

**{**

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

**i++;**

**}**

**while (i<=25);**

**</script>**

**</body>**

**</html>**

Output:

21  
22  
23  
24  
25

## 4) JavaScript for in loop

The **JavaScript for in loop** is used to iterate the properties of an object. We will discuss about it later.

# 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.

## JavaScript Function Syntax

The syntax of declaring function is given below.

function functionName([arg1, arg2, ...argN]){

 //code to be executed

}

JavaScript Functions can have 0 or more arguments.

## JavaScript Function Example

Let’s see the simple example of function in JavaScript that does not has arguments.

**<script>**

function msg(){

alert("hello! this is message");

}

**</script>**

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

#### Output of the above example

## JavaScript Function Arguments

We can call function by passing arguments. Let’s see the example of function that has one argument.

#### <html>

#### <body>

#### <script>

#### function getcube(number)

#### {

#### alert(number\*number\*number);

#### }

#### </script>

#### <form>

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

#### </form>

#### </body>

#### </html>

#### Output of the above example

**Top of Form**

**Bottom of Form**

## Function with Return Value

We can call function that returns a value and use it in our program. Let’s see the example of function that returns value.

#### <html>

#### <body>

#### <script>

#### function getInfo()

#### {

#### return "hello zerozilla! How r u?";

#### }

#### </script>

#### <script>

#### document.write(getInfo());

#### </script>

#### </body>

#### </html>

#### Output of the above example

hello zerozilla! How r u?

# 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)

## 1) JavaScript Object by object literal

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

**object={property1:value1,property2:value2.....propertyN:valueN}**

As you can see, property and value is separated by : (colon).

Let’s see the simple example of creating object in JavaScript.

#### <html>

#### <body>

#### <script>

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

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

#### </script>

#### </body>

#### </html>

#### Output of the above example

102 Shyam Kumar 40000

## 2) By creating instance of Object

The syntax of creating object directly is given below:

var objectname=new Object();

Here, **new keyword** is used to create object.

Let’s see the example of creating object directly.

**<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>**

#### Output of the above example

101 Ravi 50000

## 3) By using an Object constructor

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

The **this keyword** refers to the current object.

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

**<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>**

#### Output of the above example

103 Vimal Jaiswal 30000

## Defining method in JavaScript Object

We can define method in JavaScript object. But before defining method, we need to add property in the function with same name as method.

The example of defining method in object is given below.

<html>

<body>

<script>

function emp(id,name,salary)

{

var id,name,salary,changeSalary;

this.id=id;

this.name=name;

this.salary=salary;

this.changeSalary=changeSalary;

function changeSalary(otherSalary)

{

this.salary=otherSalary;

}

}

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

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

e.changeSalary(45000);

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

</script>

</body>

</html>

#### Output of the above example

103 Sonoo Jaiswal 30000

103 Sonoo Jaiswal 45000

# 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:

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

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

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

**<html>**

**<body>**

**<script>**

**var emp = ["AK", "PK", "KHAKE"];**

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

**{**

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

**}**

**</script>**

**</body>**

**</html>**

The .length property returns the length of an array.

#### Output of the above example

AK  
PK  
KHAKE

## 2) JavaScript Array directly (new keyword)

The syntax of creating array directly is given below:

var arrayname=new Array();

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

Let’s see the example of creating array directly.

#### <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>

#### 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.

#### <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>

#### 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:

var stringname="string value";

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

**<!DOCTYPE html>**

**<html>**

**<body>**

**<script>**

**var str = "This is string...!";**

**document.write(str);**

**</script>**

**</body>**

**</html>**

Output:

**This is string...!**

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

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

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.

**<!DOCTYPE html>**

**<html>**

**<body>**

**<script>**

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

**document.write(stringname);**

**</script>**

**</body>**

**</html>**

Output:

hello javascript; this is string object

## 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()

## 1) JavaScript String charAt(index) Method

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

**<!DOCTYPE html>**

**<html>**

**<body>**

**<script>**

**var str="javascript";**

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

**</script>**

**</body>**

**</html>**

Output:

v

## 2) JavaScript String concat(str) Method

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

**<!DOCTYPE html>**

**<html>**

**<body>**

**<script>**

**var s1="javascript ";**

**var s2="concat example";**

**var s3=s1.concat(s2);**

**document.write(s3);**

**</script>**

**</body>**

**</html>**

Output:

javascript concat example

## 3) JavaScript String indexOf(str) Method

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

**<!DOCTYPE html>**

**<html>**

**<body>**

**<script>**

**var s1="javascript Indexof";**

**var n=s1.indexOf("a");**

**document.write(n);**

**</script>**

**</body>**

**</html>**

Output:

1

## 4) JavaScript String lastIndexOf(str) Method

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

**<!DOCTYPE html>**

**<html>**

**<body>**

**<script>**

**var s1="javascript last indexof";**

**var n=s1.lastIndexOf("st");**

**document.write(n);**

**</script>**

**</body>**

**</html>**

Output:

13

## 5) JavaScript String toLowerCase() Method

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

**<!DOCTYPE html>**

**<html>**

**<body>**

**<script>**

**var s1="JAVASCRIPT TOLOWERCASE EXAMPLE";**

**var s2=s1.toLowerCase();**

**document.write(s2);**

**</script>**

**</body>**

**</html>**

Output:

javascript tolowercase example

## 6) JavaScript String toUpperCase() Method

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

**<!DOCTYPE html>**

**<html>**

**<body>**

**<script>**

**var s1="JavaScript toUpperCase Example";**

**var s2=s1.toUpperCase();**

**document.write(s2);**

**</script>**

**</body>**

**</html>**

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.

**<!DOCTYPE html>**

**<html>**

**<body>**

**<script>**

**var s1="abcdefgh";**

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

**document.write(s2);**

**</script>**

**</body>**

**</html>**

Output:

cde

## 8) JavaScript String trim() Method

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

**<!DOCTYPE html>**

**<html>**

**<body>**

**<script>**

**var s1=" javascript trim ";**

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

**document.write(s2);**

**</script>**

**</body>**

**</html>**

Output:

javascript trim

# 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

The important methods of date object are as follows:

|  |  |
| --- | --- |
| **Method** | **Description** |
| getFullYear() | returns the year in 4 digit e.g. 2015. It is a new method and suggested than getYear() which is now deprecated. |
| getMonth() | returns the month in 2 digit from 0 to 11. So it is better to use getMonth()+1 in your code. |
| getDate() | returns the date in 1 or 2 digit from 1 to 31. |
| getDay() | returns the day of week in 1 digit from 0 to 6. |
| getHours() | returns all the elements having the given name value. |
| getMinutes() | returns all the elements having the given class name. |
| getSeconds() | returns all the elements having the given class name. |
| getMilliseconds() | returns all the elements having the given tag name. |

### JavaScript Date Example

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

<html>

<body>

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

<script>

var today=new Date();

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

</script>

</body>

</html>

**Output:**

Current Date and Time: Wed Apr 05 2017 12:17:32 GMT+0530 (India Standard Time)

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

**<html>**

**<body>**

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

**<script>**

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

**var day=date.getDate();**

**var month=date.getMonth()+1;**

**var year=date.getFullYear();**

**document.getElementById('txt').innerHTML=day+":"+month+":"+year;**

**</script>**

**</body>**

**</html>**

Output:

Date is: 5/4/2017

### JavaScript Current Time Example

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

<html>

<body>

Current Time: <span id="txt"></span>

<script>

var today=new Date();

var h=today.getHours();

var m=today.getMinutes();

var s=today.getSeconds();

document.getElementById('txt').innerHTML=h+":"+m+":"+s;

</script>

</body>

</html>

**Output:**

Current Time: 12:31:51

### 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.

<html>

<body>

Current Time: <span id="txt"></span>

<script>

window.onload = function()

{ getTime(); }

function getTime()

{

var today = new Date();

var h = today.getHours();

var m = today.getMinutes();

var s = today.getSeconds();

// add a zero in front of numbers<10

m = checkTime(m);

s = checkTime(s);

document.getElementById('txt').innerHTML = h + ":" + m + ":" + s;

setTimeout

(

function()

{

getTime()

},

1000

);

}

//setInterval("getTime()",1000);//another way

function checkTime(i)

{

if (i < 10)

{

i = "0" + i;

}

return i;

}

</script>

</body>

</html>

**Output:**

Current Time: 12:09:45

# JavaScript Math Object

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

## Math.sqrt(n)

The JavaScript math.sqrt(n) method returns the square root of the given number.

<!DOCTYPE html>

<html>

<body>

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

<script>

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

</script>

</body>

</html>

**Output:**

Square Root of 17 is: 4.123105625617661

## Math.random()

The JavaScript math.random() method returns the random number between 0 to 1.

<!DOCTYPE html>

<html>

<body>

Random Number is: <span id="p2"></span>

<script>

// By default it display random number in between 0 and 1

document.getElementById('p2').innerHTML=Math.random();

</script>

</body>

</html>

**Output:**

Random Number is: 0.18144388237419196

## Math.pow(m,n)

The JavaScript math.pow(m,n) method returns the m to the power of n that is mn.

<!DOCTYPE html>

<html>

<body>

3 to the power of 4 is: <span id="p3"></span>

<script>

document.getElementById('p3').innerHTML=Math.pow(3,4);

</script>

</body>

</html>

**Output:**

3 to the power of 4 is: 81

## Math.floor(n)

The JavaScript math.floor(n) method returns the lowest integer for the given number. For example 3 for 3.7, 5 for 5.9 etc.

<!DOCTYPE html>

<html>

<body>

Floor of 4.6 is: <span id="p4"></span>

<script>

document.getElementById('p4').innerHTML=Math.floor(4.6);

</script>

</body>

</html>

**Output:**

Floor of 4.6 is: 4

## Math.ceil(n)

The JavaScript math.ceil(n) method returns the largest integer for the given number. For example 4 for 3.7, 6 for 5.9 etc.

<!DOCTYPE html>

<html>

<body>

Ceil of 4.6 is: <span id="p5"></span>

<script>

document.getElementById('p5').innerHTML=Math.ceil(4.6);

</script>

</body>

</html>

**Output:**

Ceil of 4.6 is: 5

## Math.round(n)

The JavaScript math.round(n) method returns the rounded integer nearest for the given number. If fractional part is equal or greater than 0.5, it goes to upper value 1 otherwise lower value 0. For example 4 for 3.7, 3 for 3.3, 6 for 5.9 etc.

<!DOCTYPE html>

<html>

<body>

Round of 4.3 is: <span id="p6"></span><br>

Round of 4.7 is: <span id="p7"></span>

<script>

document.getElementById('p6').innerHTML=Math.round(4.3);

document.getElementById('p7').innerHTML=Math.round(4.7);

</script>

</body>

</html>

**Output:**

Round of 4.3 is: 4

Round of 4.7 is: 5

## Math.abs(n)

The JavaScript math.abs(n) method returns the absolute value for the given number. For example 4 for -4, 6.6 for -6.6 etc.

<!DOCTYPE html>

<html>

<body>

Absolute value of -4 is: <span id="p8"></span>

<script>

document.getElementById('p8').innerHTML=Math.abs(-4);

</script>

</body>

</html>

**Output:**

Absolute value of -4 is: 4

# 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:

**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:

<!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>

**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 description.

|  |  |
| --- | --- |
| **Methods** | **Description** |
| toExponential(x) | displays exponential value. |
| toFixed(x) | limits the number of digits after decimal value. |
| toPrecision(x) | formats the number with given number of digits. |
| toString() | converts number into string. |
| valueOf() | coverts other type of value into number. |

# 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.

Boolean b=new Boolean(value);

The default value of JavaScript Boolean object is false.

## JavaScript Boolean Example

**<script>**

document.write(10**<20**);//true

document.write(10**<5**);//false

**</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. |

# BOM- 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:

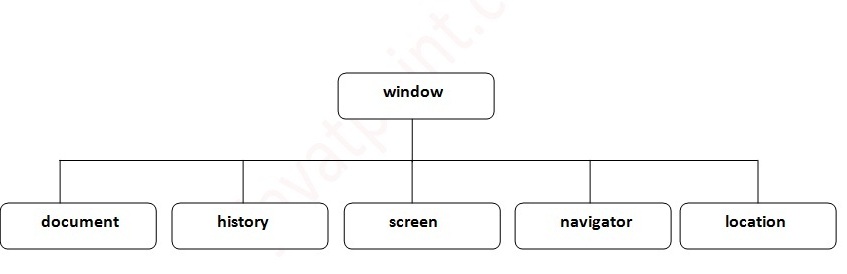
window.alert("hello zerozilla");

is same as:

alert("hello zerozilla");

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.

# 1.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.

#### Note: if html document contains frame or iframe, browser creates additional window objects for each frame.

## 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. |

#### Example of alert() in javascript

It displays alert dialog box. It has message and ok button.

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

function msg()

{

alert("Hello JS");

}

**</script>**

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

#### Example of confirm() in javascript

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

**<html>**

**<body>**

**<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()"/>**

**</body>**

**</html>**

#### Example of prompt() in javascript

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

**<html>**

**<body>**

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

**function msg()**

**{**

**var v= prompt("Who are you?");**

**alert("I am "+v);**

**}**

**</script>**

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

**</body>**

**</html>**

#### Example of open() in javascript

It displays the content in a new window.

**<html>**

**<body>**

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

**function msg()**

**{**

**open("http://www.google.com");**

**}**

**</script>**

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

**</body>**

**</html>**

#### Example of setTimeout() in javascript

It performs its task after the given milliseconds.

**<html>**

**<body>**

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

**function msg()**

**{**

**setTimeout**

**(**

**function()**

**{**

**alert("Well come to Your world..!")**

**},**

**3000**

**);**

**}**

**</script>**

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

**</body>**

**</html>**

# 2. 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:

**window.history  Or, history**

## Property of JavaScript history object

There are only 1 property of history object.

|  |  |  |
| --- | --- | --- |
| **No.** | **Property** | **Description** |
| 1 | length | returns the length of the history URLs. |

## 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

# 3. 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:

**window.navigator Or, navigator**

## Property of JavaScript navigator object

There are many properties of navigator object that returns information of the browser.

|  |  |  |
| --- | --- | --- |
| **No.** | **Property** | **Description** |
| 1 | appName | returns the name |
| 2 | appVersion | returns the version |
| 3 | appCodeName | returns the code name |
| 4 | cookieEnabled | returns true if cookie is enabled otherwise false |
| 5 | userAgent | returns the user agent |
| 6 | language | returns the language. It is supported in Netscape and Firefox only. |
| 7 | userLanguage | returns the user language. It is supported in IE only. |
| 8 | plugins | returns the plugins. It is supported in Netscape and Firefox only. |
| 9 | systemLanguage | returns the system language. It is supported in IE only. |
| 10 | mimeTypes[] | returns the array of mime type. It is supported in Netscape and Firefox only. |
| 11 | platform | returns the platform e.g. Win32. |
| 12 | online | returns true if browser is online otherwise false. |

## 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.

<html>

<body>

<h2>JavaScript Navigator Object</h2>

<script>

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

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

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

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

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

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

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

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

</script>

</body>

</html>

**Output :**

**JavaScript Navigator Object**  
navigator.appCodeName: Mozilla   
navigator.appName: Netscape   
navigator.appVersion: 5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36   
navigator.cookieEnabled: true   
navigator.language: en-GB   
navigator.userAgent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36   
navigator.platform: Win32   
navigator.onLine: true

# 4. JavaScript Screen Object

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

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

**window.screen  Or, screen**

## Property of JavaScript Screen Object

There are many properties of screen object that returns information of the browser.

|  |  |  |
| --- | --- | --- |
| **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.

<html>

<body>

<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>

</body>

</html>

**Output:**

screen.width: 1366   
screen.height: 768   
screen.availWidth: 1366   
screen.availHeight: 738   
screen.colorDepth: 24   
screen.pixelDepth: 24

# Document Object Model

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

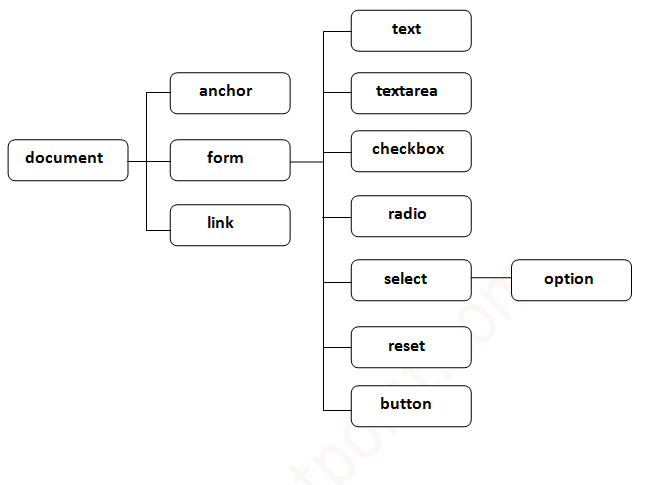
When html document is loaded in the browser, it becomes a document object. It is the **root element** that represents the html document. It has properties and methods. By the help of document object, we can add dynamic content to our web page.

As mentioned earlier, it is the object of window. So

**window.document  Is same as document**

According to W3C - *"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."*

## Properties of document object

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

### Accessing field value by document object

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.

**value** is the property, that returns the value of the input text.

Let's see the simple example of document object that prints name with welcome message.

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

function printvalue(){

var name=document.form1.name.value;

alert("Welcome: "+name);

}

**</script>**

**<form** name="form1"**>**

Enter Name:**<input** type="text" name="name"**/>**

**<input** type="button" onclick="printvalue()" value="print name"**/>**

**</form>**

# 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.

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

**<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

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

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

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.

**<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() method**

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

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

document.getElementsByTagName("name")

Here, name is required.

### 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.

**<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>**

#### Output of the above example

This is a pragraph

Here we are going to count total number of paragraphs by getElementByTagName() method.

Let's see the simple example

count paragraph

### Another example of document.getElementsByTagName() method

In this example, we going to count total number of h2 and h3 tags used in the document.

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

<script>

function counth2(){

var totalh2=document.getElementsByTagName("h2");

alert("total h2 tags are: "+totalh2.length);

}

function counth3(){

var totalh3=document.getElementsByTagName("h3");

alert("total h3 tags are: "+totalh3.length);

}

**</script>**

**<h2>**This is h2 tag**</h2>**

**<h2>**This is h2 tag**</h2>**

**<h3>**This is h3 tag**</h3>**

**<h3>**This is h3 tag**</h3>**

**<h3>**This is h3 tag**</h3>**

**<button** onclick="counth2()"**>**count h2**</button>**

**<button** onclick="counth3()"**>**count h3**</button>**

#### Output of the above example

## This is h2 tag

## This is h2 tag

### This is h3 tag

### This is h3 tag

### This is h3 tag

count h2 count h3

#### Note: Output of the given examples may differ on this page because it will count the total number of para , total number of h2 and total number of h3 tags used in this document.

Top of Form

Bottom of Form

# Javascript - 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.

In this example, we are dynamically writing the html form inside the div name having the id mylocation. We are identifing this position by calling the document.getElementById() method.

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

function showcommentform() {

var data="Name:**<input** type='text' name='name'**><br>**Comment:**<br><textarea** rows='5' cos='80'**></textarea>**

**<br><input** type='submit' value='Post Comment'**>**";

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

}

**</script>**

**<form** name="myForm"**>**

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

**<div** id="mylocation"**></div>**

**</form>**

Top of Form

Bottom of Form

### Show/Hide Comment Form Example using innerHTML

<!DOCTYPE html**>**

**<html>**

**<head>**

**<title>**First JS**</title>**

**<script>**

var flag=true;

function commentform(){

var cform="**<form** action='Comment'**>**Enter Name:**<br><input** type='text' name='name'**/><br/>**

Enter Email:**<br><input** type='email' name='email'**/><br>**Enter Comment:**<br/>**

**<textarea** rows='5' cols='70'**></textarea><br><input** type='submit' value='Post Comment'**/></form>**";

if(flag){

document.getElementById("mylocation").innerHTML=cform;

flag=false;

}else{

document.getElementById("mylocation").innerHTML="";

flag=true;

}

}

**</script>**

**</head>**

**<body>**

**<button** onclick="commentform()"**>**Comment**</button>**

**<div** id="mylocation"**></div>**

**</body>**

**</html>**

# Javascript - innerText

The **innerText** property can be used to write the dynamic text on the html document. Here, text will not be interpreted as html text but a normal text.

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

## Javascript innerText Example

In this example, we are going to display the password strength when releases the key after press.

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

function validate() {

var msg;

if(document.myForm.userPass.value.length**>**5){

msg="good";

}

else{

msg="poor";

}

document.getElementById('mylocation').innerText=msg;

 }

**</script>**

**<form** name="myForm"**>**

**<input** type="password" value="" name="userPass" onkeyup="validate()"**>**

Strength:**<span** id="mylocation"**>**no strength**</span>**

**</form>**

# JavaScript Form Validation

It is important to validate the form submitted by the user because it can have inappropriate values. So validation is must.

The JavaScript provides you the facility the 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.

## JavaScript form validation example

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.

**<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.jsp" 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>**

## JavaScript Number Validation

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

**<script>**

function validate(){

var num=document.myform.num.value;

if (isNaN(num)){

  document.getElementById("numloc").innerHTML="Enter Numeric value only";

  return false;

}else{

  return true;

  }

}

**</script>**

**<form** name="myform" onsubmit="return validate()" **>**

Number: **<input** type="text" name="num"**><span** id="numloc"**></span><br/>**

**<input** type="submit" value="submit"**>**

**</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.

**<script>**

function validate(){

var name=document.f1.name.value;

var password=document.f1.password.value;

var status=false;

if(name.length**<1**){

document.getElementById("nameloc").innerHTML=

" <img src='unchecked.gif'/> Please enter your name";

status=false;

}else{

document.getElementById("nameloc").innerHTML=" <img src='checked.gif'/>";

status=true;

}

if(password.length**<6**){

document.getElementById("passwordloc").innerHTML=

" <img src='unchecked.gif'/> Password must be at least 6 char long";

status=false;

}else{

document.getElementById("passwordloc").innerHTML=" <img src='checked.gif'/>";

}

return status;

}

**</script>**

**<form** name="f1" action="#" onsubmit="return validate()"**>**

**<table>**

**<tr><td>**Enter Name:**</td><td><input** type="text" name="name"**/>**

**<span** id="nameloc"**></span></td></tr>**

**<tr><td>**Enter Password:**</td><td><input** type="password" name="password"**/>**

**<span** id="passwordloc"**></span></td></tr>**

**<tr><td** colspan="2"**><input** type="submit" value="register"**/></td></tr>**

**</table>**

**</form>**

Output:

Top of Form

|  |  |
| --- | --- |
| Enter Name: |  |
| Enter Password: |  |
|  | |

Bottom of 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.

**<script>**

function validateemail()

{

var x=document.myform.email.value;

var atposition=x.indexOf("@");

var dotposition=x.lastIndexOf(".");

if (atposition**<1** || dotposition**<atposition**+2 || dotposition+2**>**=x.length){

  alert("Please enter a valid e-mail address \n atpostion:"+atposition+"\n dotposition:"+dotposition);

  return false;

  }

}

**</script>**

**<body>**

**<form** name="myform"  method="post" action="#" onsubmit="return validateemail();"**>**

Email: **<input** type="text" name="email"**><br/>**

**<input** type="submit" value="register"**>**

**</form>**

# HTML/DOM events for JavaScript

HTML or DOM events are widely used in JavaScript code. JavaScript code is executed with HTML/DOM events. So before learning JavaScript, let’s have some idea about events.

|  |  |
| --- | --- |
| **Events** | **Description** |
| onclick | occurs when element is clicked. |
| ondblclick | occurs when element is double-clicked. |
| onfocus | occurs when an element gets focus such as button, input, textarea etc. |
| onblur | occurs when form looses the focus from an element. |
| onsubmit | occurs when form is submitted. |
| onmouseover | occurs when mouse is moved over an element. |
| onmouseout | occurs when mouse is moved out from an element (after moved over). |
| onmousedown | occurs when mouse button is pressed over an element. |
| onmouseup | occurs when mouse is released from an element (after mouse is pressed). |
| onload | occurs when document, object or frameset is loaded. |
| onunload | occurs when body or frameset is unloaded. |
| onscroll | occurs when document is scrolled. |
| onresized | occurs when document is resized. |
| onreset | occurs when form is reset. |
| onkeydown | occurs when key is being pressed. |
| onkeypress | occurs when user presses the key. |
| onkeyup | occurs when key is released. |

# JavaScript Interview Questions

JavaScript interview questions and answers for provides a list of top 20 interview questions. The frequently asked JavaScript interview questions with answers for beginners and professionals are given below.

### 1) What is JavaScript?

**JavaScript** is a scripting language. It is different from Java language. It is object-based, lightweight and cross platform. It is widely used for client side validation.

### 2) What is the difference between JavaScript and jscript?

Netscape provided the JavaScript language. Microsoft changed the name and called it JScript to avoid the trademark issue.In other words, you can say JScript is same as JavaScript, but it is provided by Microsoft.

### 3) How to write a hello world example of JavaScript?

A simple example of JavaScript hello world is given below. You need to place it inside the body tag of html.

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

document.write("JavaScript Hello World!");

**</script>**

### 4) How to use external JavaScript file?

I am assuming that js file name is message.js, place the following script tag inside the head tag.

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

### 5) Is JavaScript case sensitive language?

Yes.

### 6) What is BOM?

**BOM** stands for Browser Object Model. It provides interaction with the browser. The default object of browser is window.

### 7) What is DOM? What is the use of document object?

**DOM** stands for Document Object Model. A document object represent the html document. It can be used to access and change the content of html.

### 8) What is the use of window object?

The window object is automatically created by the browser that represents a window of a browser.

It is used to display the popup dialog box such as alert dialog box, confirm dialog box, input dialog box etc.

### 9) What is the use of history object?

The history object of browser can be used to switch to history pages such as back and forward from current page or another page. There are three methods of history object.

history.back()

history.forward()

history.go(number): number may be positive for forward, negative for backward.

### 10) How to write comment in JavaScript?

There are two types of comments in JavaScript.

Single Line Comment: It is represented by // (double forward slash)

Multi Line Comment: It is represented by slash with asterisk symbol as /\* write comment here \*/

### 11) How to create function in JavaScript?

To create function in JavaScript, follow the following syntax.

function function\_name(){

//function body

}

### 12) What are the JavaScript data types?

There are two types of data types in JavaScript:

1. Primitive Data Types
2. Non-primitive Data Types

# 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 primitive data types

There are five types of primitive data types in JavaScript. They are as follows:

|  |  |
| --- | --- |
| **Data Type** | **Description** |
| String | represents sequence of characters e.g. "hello" |
| Number | represents numeric values e.g. 100 |
| Boolean | represents boolean value either false or true |
| Undefined | represents undefined value |
| Null | represents null i.e. no value at all |

## JavaScript non-primitive data types

The non-primitive data types are as follows:

|  |  |
| --- | --- |
| **Data Type** | **Description** |
| Object | represents instance through which we can access members |
| Array | represents group of similar values |
| RegExp | represents regular expression |

We will have great discussion on each data type later.

### 13) What is the difference between == and ===?

The == operator checks equality only whereas === checks equality and data type i.e. value must be of same type.

### 14) How to write html code dynamically using JavaScript?

The innerHTML property is used to write the HTML code using JavaScript dynamically. Let's see a simple example:

document.getElementById('mylocation').innerHTML="<h2>This is heading using JavaScript</h2>";

### 15) How to write normal text code using JavaScript dynamically?

The innerText property is used to write the simple text using JavaScript dynamically. Let's see a simple example:

document.getElementById('mylocation').innerText="This is text using JavaScript";

### 16) How to create objects in JavaScript?

There are 3 ways to create object in JavaScript.

1. By object literal
2. By creating instance of Object
3. By Object Constructor

Let's see a simple code to create object using object literal.

emp={id:102,name:"Rahul Kumar",salary:50000}

### 17) How to create array in JavaScript?

There are 3 ways to create array in JavaScript.

1. By array literal
2. By creating instance of Array
3. By using an Array constructor

Let's see a simple code to create array using object literal.

var emp=["Shyam","Vimal","Ratan"];

### 18) What does the isNaN() function?

The isNan() function returns true if the variable value is not a number.

### 19) What is the output of 10+20+"30" in JavaScript?

3030 because 10+20 will be 30. If there is numeric value before and after +, it is treated is binary + (arithmetic operator).

### 20) What is the output of "10"+20+30 in JavaScript?

102030 because after a string all the + will be treated as string concatenation operator (not binary +).

### 21) Difference between Client side JavaScript and Server side JavaScript?

**Client side JavaScript** comprises the basic language and predefined objects which are relevant to running java script in a browser. The client side JavaScript is embedded directly by in the HTML pages. This script is interpreted by the browser at run time.

**Server side JavaScript** also resembles like client side java script. It has relevant java script which is to run in a server. The server side JavaScript are deployed only after compilation.

### 22) In which location cookies are stored on the hard disk?

The storage of cookies on the hard disk depends on OS and the browser.

The Netscape navigator on Windows uses cookies.txt file that contains all the cookies. The path is : c:\Program Files\Netscape\Users\username\cookies.txt

The Internet Explorer stores the cookies on a file username@website.txt. The path is: c:\Windows\Cookies\username@Website.txt.

### 23) What is the real name of JavaScript?

The original name was **Mocha**, a name chosen by Marc Andreessen, founder of Netscape. In September of 1995, the name was changed to LiveScript. In December 1995, after receiving a trademark license from Sun, the name JavaScript was adopted.

### 24) What is the difference between undefined value and null value?

**Undefined value:** A value that is not defined and has no keyword is known as undefined value. For example:

int number;//Here, number has undefined value.

**Null value:** A value that is explicitly specified by the keyword "null" is known as null value. For example:

String str=null;//Here, str has a null value.

### 24) How to set the cursor to wait in JavaScript?

The cursor can be set to wait in JavaScript by using the property "cursor". The following example illustrates the usage:

**<script>**

window.document.body.style.cursor = "wait";

**</script>**

### 25) What is this?

var myArray = [[[]]];

Ans: Three dimensional array.

### 26) Are Java and JavaScript same?

No, Java and JavaScript are the two different languages. Java is a robust, secured and object-oriented programming language whereas JavaScript is a client side scripting language with some limitations.

### 27) What is negative infinity?

Negative Infinity is a number in JavaScript which can be derived by dividing negative number by zero.

### 28) What is difference between View state and Session state?

"View state" is specific to a page in a session whereas "Session state" is specific to a user or browser that can be accessed across all pages in the web application.

### 29) What are the pop up boxes available in JavaScript?

* Alert Box
* Confirm Box
* Prompt Box

### 30) How can we detect OS of the client machine using JavaScript?

The **navigator.appVersion** string can be used to detect the operating system on the client machine.

### 31) How to submit a form using JavaScript by clicking a link?

Let's see the JavaScript code to submit form on clicking the link.

**<form** name="myform" action="index.php"**>**

Search: **<input** type='text' name='query' **/>**

**<a** href="javascript: submitform()"**>**Search**</a>**

**</form>**

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

function submitform()

{

  document.myform.submit();

}

**</script>**

### 32) Is JavaScript faster than ASP script?

Yes, because it doesn't require web server's support for execution.

### 33) How to change the background color of HTML document using JavaScript?

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

document.body.bgColor="pink";

**</script>**

### 34) How to handle exceptions in JavaScript?

By the help of try/catch block, we can handle exceptions in JavaScript. JavaScript supports try, catch, finally and throw keywords for exception handling.

### 35) How to validate a form in JavaScript?

### 36) How to validate email in JavaScript?