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.

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.

## 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.

JavaScript Example

1. **<script>**
2. document.write("Hello JavaScript by JavaScript");
3. **</script>**

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.

1. **<html>**
2. **<head>**
3. **<script** type="text/javascript"**>**
4. function msg(){
5. alert("Hello Javatpoint");
6. }
7. **</script>**
8. **</head>**
9. **<body>**
10. **<p>**Welcome to JavaScript**</p>**
11. **<form>**
12. **<input** type="button" value="click" onclick="msg()"**/>**
13. **</form>**
14. **</body>**
15. **</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](https://www.javatpoint.com/javascript-tutorial) file that prints Hello Javatpoint in a alert dialog box.

**message.js**

1. function msg(){
2. alert("Hello Javatpoint");
3. }

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**

1. **<html>**
2. **<head>**
3. **<script** type="text/javascript" src="message.js"**></script>**
4. **</head>**
5. **<body>**
6. **<p>**Welcome to JavaScript**</p>**
7. **<form>**
8. **<input** type="button" value="click" onclick="msg()"**/>**
9. **</form>**
10. **</body>**
11. **</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.
4. It enables both web designers and coders to work with html and js files parallelly and separately, i.e., without facing any code conflictions.
5. The length of the code reduces as only we need to specify the location of the js file.

## Disadvantages of External JavaScript

There are the following disadvantages of external files:

1. The stealer may download the coder's code using the url of the js file.
2. If two js files are dependent on one another, then a failure in one file may affect the execution of the other dependent file.
3. The web browser needs to make an additional http request to get the js code.
4. A tiny to a large change in the js code may cause unexpected results in all its dependent files.
5. We need to check each file that depends on the commonly created external javascript file.
6. If it is a few lines of code, then better to implement the internal javascript code.

JavaScript local variable

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

1. **<script>**
2. function abc(){
3. var x=10;//local variable
4. }
5. **</script>**

Or,

1. **<script>**
2. If(10**<13**){
3. var y=20;//JavaScript local variable
4. }
5. **</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:

1. **<script>**
2. var data=200;//gloabal variable
3. function a(){
4. document.writeln(data);
5. }
6. function b(){
7. document.writeln(data);
8. }
9. a();//calling JavaScript function
10. b();
11. **</script>**

# 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:

1. var a=40;//holding number
2. 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:

1. 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 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.

1. switch(expression){
2. case value1:
3. code to be executed;
4. break;
5. case value2:
6. code to be executed;
7. break;
8. ......
10. default:
11. code to be executed if above values are not matched;
12. }

# 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 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.

1. function functionName([arg1, arg2, ...argN]){
2. //code to be executed
3. }

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.

1. **<script>**
2. function msg(){
3. alert("hello! this is message");
4. }
5. **</script>**
6. **<input** type="button" onclick="msg()" value="call function"**/>**

[**Test it Now**](http://www.javatpoint.com/oprweb/test.jsp?filename=function1js)

#### 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.

1. **<script>**
2. function getcube(number){
3. alert(number\*number\*number);
4. }
5. **</script>**
6. **<form>**
7. **<input** type="button" value="click" onclick="getcube(4)"**/>**
8. **</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.

1. **<script>**
2. function getInfo(){
3. return "hello javatpoint! How r u?";
4. }
5. **</script>**
6. **<script>**
7. document.write(getInfo());
8. **</script>**

[**Test it Now**](http://www.javatpoint.com/oprweb/test.jsp?filename=function3js)

#### Output of the above example

hello javatpoint! How r u?

## JavaScript Function Object

In JavaScript, the purpose of **Function constructor** is to create a new Function object. It executes the code globally. However, if we call the constructor directly, a function is created dynamically but in an unsecured way.

## Syntax

1. new Function ([arg1[, arg2[, ....argn]],] functionBody)

## Parameter

**arg1, arg2, .... , argn** - It represents the argument used by function.

**functionBody** - It represents the function definition.

## JavaScript Function Methods

Let's see function methods with description.

|  |  |
| --- | --- |
| **Method** | **Description** |
| [apply()](https://www.javatpoint.com/javascript-function-apply-method) | It is used to call a function contains this value and a single array of arguments. |
| [bind()](https://www.javatpoint.com/javascript-function-bind-method) | It is used to create a new function. |
| [call()](https://www.javatpoint.com/javascript-function-call-method) | It is used to call a function contains this value and an argument list. |
| [toString()](https://www.javatpoint.com/javascript-function-tostring-method) | It returns the result in a form of a string. |

## JavaScript Function Object Examples

### Example 1

Let's see an example to display the sum of given numbers.

1. **<script>**
2. var add=new Function("num1","num2","return num1+num2");
3. document.writeln(add(2,5));
4. **</script>**

[**Test it Now**](https://www.javatpoint.com/oprweb/test.jsp?filename=JavaScriptFunctionObjectExample1)

**Output:**

7

### Example 2

Let's see an example to display the power of provided value.

1. **<script>**
2. var pow=new Function("num1","num2","return Math.pow(num1,num2)");
3. document.writeln(pow(2,3));
4. **</script>**

# 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:

1. 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.

1. **<script>**
2. emp={id:102,name:"Shyam Kumar",salary:40000}
3. document.write(emp.id+" "+emp.name+" "+emp.salary);
4. **</script>**

[**Test it Now**](http://www.javatpoint.com/oprweb/test.jsp?filename=object1js)

#### Output of the above example

102 Shyam Kumar 40000

## 2) By creating instance of Object

The syntax of creating object directly is given below:

1. var objectname=new Object();

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

Let’s see the example of creating object directly.

1. **<script>**
2. var emp=new Object();
3. emp.id=101;
4. emp.name="Ravi Malik";
5. emp.salary=50000;
6. document.write(emp.id+" "+emp.name+" "+emp.salary);
7. **</script>**

**[Test it Now](http://www.javatpoint.com/oprweb/test.jsp?filename=object2js" \t "_blank)**

#### 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.

1. **<script>**
2. function emp(id,name,salary){
3. this.id=id;
4. this.name=name;
5. this.salary=salary;
6. }
7. e=new emp(103,"Vimal Jaiswal",30000);
9. document.write(e.id+" "+e.name+" "+e.salary);
10. **</script>**

**[Test it Now](http://www.javatpoint.com/oprweb/test.jsp?filename=object3js" \t "_blank)**

#### 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.

1. **<script>**
2. function emp(id,name,salary){
3. this.id=id;
4. this.name=name;
5. this.salary=salary;
7. this.changeSalary=changeSalary;
8. function changeSalary(otherSalary){
9. this.salary=otherSalary;
10. }
11. }
12. e=new emp(103,"Sonoo Jaiswal",30000);
13. document.write(e.id+" "+e.name+" "+e.salary);
14. e.changeSalary(45000);
15. document.write("**<br>**"+e.id+" "+e.name+" "+e.salary);
16. **</script>**

**[Test it Now](http://www.javatpoint.com/oprweb/test.jsp?filename=object4js" \t "_blank)**

#### Output of the above example

103 Sonoo Jaiswal 30000  
103 Sonoo Jaiswal 45000

## JavaScript Object Methods

The various methods of Object are as follows:

|  |  |  |
| --- | --- | --- |
| **S.No** | **Methods** | **Description** |
| 1 | [Object.assign()](https://www.javatpoint.com/javascript-object-assign-method) | This method is used to copy enumerable and own properties from a source object to a target object |
| 2 | [Object.create()](https://www.javatpoint.com/javascript-object-create-method) | This method is used to create a new object with the specified prototype object and properties. |
| 3 | [Object.defineProperty()](https://www.javatpoint.com/javascript-object-defineproperty-method) | This method is used to describe some behavioral attributes of the property. |
| 4 | [Object.defineProperties()](https://www.javatpoint.com/javascript-object-defineproperties-method) | This method is used to create or configure multiple object properties. |
| 5 | [Object.entries()](https://www.javatpoint.com/javascript-object-entries-method) | This method returns an array with arrays of the key, value pairs. |
| 6 | [Object.freeze()](https://www.javatpoint.com/javascript-object-freeze-method) | This method prevents existing properties from being removed. |
| 7 | [Object.getOwnPropertyDescriptor()](https://www.javatpoint.com/javascript-object-getownpropertydescriptor-method) | This method returns a property descriptor for the specified property of the specified object. |
| 8 | [Object.getOwnPropertyDescriptors()](https://www.javatpoint.com/javascript-object-getownpropertydescriptors-method) | This method returns all own property descriptors of a given object. |
| 9 | [Object.getOwnPropertyNames()](https://www.javatpoint.com/javascript-object-getownpropertynames-method) | This method returns an array of all properties (enumerable or not) found. |
| 10 | [Object.getOwnPropertySymbols()](https://www.javatpoint.com/javascript-object-getownpropertysymbols-method) | This method returns an array of all own symbol key properties. |
| 11 | [Object.getPrototypeOf()](https://www.javatpoint.com/javascript-object-getprototypeof-method) | This method returns the prototype of the specified object. |
| 12 | [Object.is()](https://www.javatpoint.com/javascript-object-is-method) | This method determines whether two values are the same value. |
| 13 | [Object.isExtensible()](https://www.javatpoint.com/javascript-objects) | This method determines if an object is extensible |
| 14 | [Object.isFrozen()](https://www.javatpoint.com/javascript-objects) | This method determines if an object was frozen. |
| 15 | [Object.isSealed()](https://www.javatpoint.com/javascript-objects) | This method determines if an object is sealed. |
| 16 | [Object.keys()](https://www.javatpoint.com/javascript-objects) | This method returns an array of a given object's own property names. |
| 17 | [Object.preventExtensions()](https://www.javatpoint.com/javascript-object-preventextensions-method) | This method is used to prevent any extensions of an object. |
| 18 | [Object.seal()](https://www.javatpoint.com/javascript-object-seal-method) | This method prevents new properties from being added and marks all existing properties as non-configurable. |
| 19 | [Object.setPrototypeOf()](https://www.javatpoint.com/javascript-object-setprototypeof-method) | This method sets the prototype of a specified object to another object. |
| 20 | [Object.values()](https://www.javatpoint.com/javascript-object-values-method) | This method returns an array of values. |

# JavaScript Array

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

There are 3 ways to construct array in JavaScript

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

## 1) JavaScript array literal

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

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

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

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

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

[**Test it Now**](http://www.javatpoint.com/oprweb/test.jsp?filename=array1js)

The .length property returns the length of an array.

**Output of the above example**

Sonoo  
Vimal  
Ratan

## 2) JavaScript Array directly (new keyword)

The syntax of creating array directly is given below:

1. var arrayname=new Array();

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

Let's see the example of creating array directly.

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

[**Test it Now**](http://www.javatpoint.com/oprweb/test.jsp?filename=array2js)

**Output of the above example**

Arun  
Varun  
John

## 3) JavaScript array constructor (new keyword)

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

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

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

[**Test it Now**](http://www.javatpoint.com/oprweb/test.jsp?filename=array3js)

**Output of the above example**

Jai  
Vijay  
Smith

## JavaScript Array Methods

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

|  |  |
| --- | --- |
| **Methods** | **Description** |
| [concat()](https://www.javatpoint.com/javascript-array-concat-method) | It returns a new array object that contains two or more merged arrays. |
| [copywithin()](https://www.javatpoint.com/javascript-array-copywithin-method) | It copies the part of the given array with its own elements and returns the modified array. |
| [entries()](https://www.javatpoint.com/javascript-array-entries-method) | It creates an iterator object and a loop that iterates over each key/value pair. |
| [every()](https://www.javatpoint.com/javascript-array-every-method) | It determines whether all the elements of an array are satisfying the provided function conditions. |
| [flat()](https://www.javatpoint.com/javascript-array-flat-method) | It creates a new array carrying sub-array elements concatenated recursively till the specified depth. |
| [flatMap()](https://www.javatpoint.com/javascript-array-flatmap-method) | It maps all array elements via mapping function, then flattens the result into a new array. |
| [fill()](https://www.javatpoint.com/javascript-array-fill-method) | It fills elements into an array with static values. |
| [from()](https://www.javatpoint.com/javascript-array-from-method) | It creates a new array carrying the exact copy of another array element. |
| [filter()](https://www.javatpoint.com/javascript-array-filter-method) | It returns the new array containing the elements that pass the provided function conditions. |
| [find()](https://www.javatpoint.com/javascript-array-find-method) | It returns the value of the first element in the given array that satisfies the specified condition. |
| [findIndex()](https://www.javatpoint.com/javascript-array-findindex-method) | It returns the index value of the first element in the given array that satisfies the specified condition. |
| [forEach()](https://www.javatpoint.com/javascript-array-foreach-method) | It invokes the provided function once for each element of an array. |
| [includes()](https://www.javatpoint.com/javascript-array-includes-method) | It checks whether the given array contains the specified element. |
| [indexOf()](https://www.javatpoint.com/javascript-array-indexof-method) | It searches the specified element in the given array and returns the index of the first match. |
| [isArray()](https://www.javatpoint.com/javascript-array-isarray-method) | It tests if the passed value ia an array. |
| [join()](https://www.javatpoint.com/javascript-array-join-method) | It joins the elements of an array as a string. |
| [keys()](https://www.javatpoint.com/javascript-array-keys-method) | It creates an iterator object that contains only the keys of the array, then loops through these keys. |
| [lastIndexOf()](https://www.javatpoint.com/javascript-array-lastindexof-method) | It searches the specified element in the given array and returns the index of the last match. |
| [map()](https://www.javatpoint.com/javascript-array-map-method) | It calls the specified function for every array element and returns the new array |
| [of()](https://www.javatpoint.com/javascript-array-of-method) | It creates a new array from a variable number of arguments, holding any type of argument. |
| [pop()](https://www.javatpoint.com/javascript-array-pop-method) | It removes and returns the last element of an array. |
| [push()](https://www.javatpoint.com/javascript-array-push-method) | It adds one or more elements to the end of an array. |
| [reverse()](https://www.javatpoint.com/javascript-array-reverse-method) | It reverses the elements of given array. |
| [reduce(function, initial)](https://www.javatpoint.com/javascript-array-reduce-method) | It executes a provided function for each value from left to right and reduces the array to a single value. |
| [reduceRight()](https://www.javatpoint.com/javascript-array-reduceright-method) | It executes a provided function for each value from right to left and reduces the array to a single value. |
| [some()](https://www.javatpoint.com/javascript-array-some-method) | It determines if any element of the array passes the test of the implemented function. |
| [shift()](https://www.javatpoint.com/javascript-array-shift-method) | It removes and returns the first element of an array. |
| [slice()](https://www.javatpoint.com/javascript-array-slice-method) | It returns a new array containing the copy of the part of the given array. |
| [sort()](https://www.javatpoint.com/javascript-array-sort-method) | It returns the element of the given array in a sorted order. |
| [splice()](https://www.javatpoint.com/javascript-array-splice-method) | It add/remove elements to/from the given array. |
| [toLocaleString()](https://www.javatpoint.com/javascript-array-tolocalestring-method) | It returns a string containing all the elements of a specified array. |
| [toString()](https://www.javatpoint.com/javascript-array-tostring-method) | It converts the elements of a specified array into string form, without affecting the original array. |
| [unshift()](https://www.javatpoint.com/javascript-array-unshift-method) | It adds one or more elements in the beginning of the given array. |
| [values()](https://www.javatpoint.com/javascript-array-values-method) | It creates a new iterator object carrying values for each index in the array. |

# JavaScript String

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

There are 2 ways to create string in JavaScript

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

## 1) By string literal

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

1. var stringname="string value";

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

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

[**Test it Now**](http://www.javatpoint.com/oprweb/test.jsp?filename=jsstring1)

**Output:**

This is string literal

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

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

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

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

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

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

[**Test it Now**](http://www.javatpoint.com/oprweb/test.jsp?filename=jsstring2)

**Output:**

hello javascript string

## JavaScript String Methods

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

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

## 1) JavaScript String charAt(index) Method

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

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

# JavaScript Date Object

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

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

## Constructor

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

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

## JavaScript Date Methods

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

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

### JavaScript Date Example

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

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

# JavaScript Math

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

## JavaScript Math Methods

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

|  |  |
| --- | --- |
| **Methods** | **Description** |
| [abs()](https://www.javatpoint.com/javascript-math-abs-method) | It returns the absolute value of the given number. |
| [acos()](https://www.javatpoint.com/javascript-math-acos-method) | It returns the arccosine of the given number in radians. |
| [asin()](https://www.javatpoint.com/javascript-math-asin-method) | It returns the arcsine of the given number in radians. |
| [atan()](https://www.javatpoint.com/javascript-math-atan-method) | It returns the arc-tangent of the given number in radians. |
| [cbrt()](https://www.javatpoint.com/javascript-math-cbrt-method) | It returns the cube root of the given number. |
| [ceil()](https://www.javatpoint.com/javascript-math-ceil-method) | It returns a smallest integer value, greater than or equal to the given number. |
| [cos()](https://www.javatpoint.com/javascript-math-cos-method) | It returns the cosine of the given number. |
| [cosh()](https://www.javatpoint.com/javascript-math-cosh-method) | It returns the hyperbolic cosine of the given number. |
| [exp()](https://www.javatpoint.com/javascript-math-exp-method) | It returns the exponential form of the given number. |
| [floor()](https://www.javatpoint.com/javascript-math-floor-method) | It returns largest integer value, lower than or equal to the given number. |
| [hypot()](https://www.javatpoint.com/javascript-math-hypot-method) | It returns square root of sum of the squares of given numbers. |
| [log()](https://www.javatpoint.com/javascript-math-log-method) | It returns natural logarithm of a number. |
| [max()](https://www.javatpoint.com/javascript-math-max-method) | It returns maximum value of the given numbers. |
| [min()](https://www.javatpoint.com/javascript-math-min-method) | It returns minimum value of the given numbers. |
| [pow()](https://www.javatpoint.com/javascript-math-pow-method) | It returns value of base to the power of exponent. |
| [random()](https://www.javatpoint.com/javascript-math-random-method) | It returns random number between 0 (inclusive) and 1 (exclusive). |
| [round()](https://www.javatpoint.com/javascript-math-round-method) | It returns closest integer value of the given number. |
| [sign()](https://www.javatpoint.com/javascript-math-sign-method) | It returns the sign of the given number |
| [sin()](https://www.javatpoint.com/javascript-math-sin-method) | It returns the sine of the given number. |
| [sinh()](https://www.javatpoint.com/javascript-math-sinh-method) | It returns the hyperbolic sine of the given number. |
| [sqrt()](https://www.javatpoint.com/javascript-math-sqrt-method) | It returns the square root of the given number |
| [tan()](https://www.javatpoint.com/javascript-math-tan-method) | It returns the tangent of the given number. |
| [tanh()](https://www.javatpoint.com/javascript-math-tanh-method) | It returns the hyperbolic tangent of the given number. |
| [trunc()](https://www.javatpoint.com/javascript-math-trunc-method) | It returns an integer part of the given number. |

## Math.sqrt(n)

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

1. Square Root of 17 is: **<span** id="p1"**></span>**
2. **<script>**
3. document.getElementById('p1').innerHTML=Math.sqrt(17);
4. **</script>**

# JavaScript Number Object

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

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

1. var n=new Number(value);

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

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

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

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

**Output:**

102 102.7 130000 16

## JavaScript Number Constants

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

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

## JavaScript Number Methods

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

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

# JavaScript Boolean

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

1. Boolean b=new Boolean(value);

The default value of JavaScript Boolean object is false.

## JavaScript Boolean Example

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

## JavaScript Boolean Properties

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

## JavaScript Boolean Methods

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