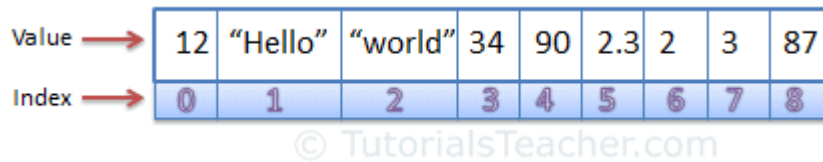


# Array in JavaScript

We have learned that a variable can hold only one value, for example `var i = 1`, we can assign only one literal value to `i`. We cannot assign multiple literal values to a variable `i`. To overcome this problem, JavaScript provides an array.

An array is a special type of variable, which can store multiple values using special syntax. Every value is associated with numeric index starting with 0. The following figure illustrates how an array stores values.

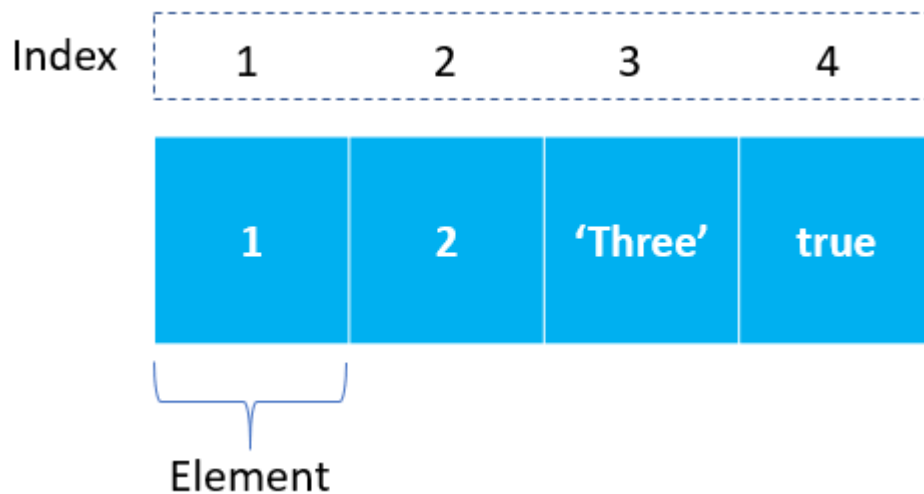


The diagram shows a table representing an array. The top row is labeled 'Value' with a red arrow pointing to the first column. The bottom row is labeled 'Index' with a red arrow pointing to the first column. The table contains 9 columns. The values are 12, "Hello", "world", 34, 90, 2.3, 2, 3, and 87. The indices are 0, 1, 2, 3, 4, 5, 6, 7, and 8. A watermark '© TutorialsTeacher.com' is visible below the table.

Value →	12	"Hello"	"world"	34	90	2.3	2	3	87
Index →	0	1	2	3	4	5	6	7	8

An array is a special variable, which can hold more than one value at a time.

In JavaScript, an array is an ordered list of values. Each value is called an element specified by an index.



JavaScript array has the following characteristics:

1. First, an array can hold values of different types. For example, you can have an array that stores the number and string, and boolean values.
2. Second, the length of an array is dynamically sized and auto-growing. In other words, you don't need to specify the array size upfront.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
var car1 = "Saab";  
var car2 = "Volvo";  
var car3 = "BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

An array can hold many values under a single name, and you can access the values by referring to an index number.

## **Array Initialization**

**There are 2 ways to construct array in JavaScript**

1. By array literal
2. By using an Array constructor (using new keyword)

### **1) JavaScript Array Literal**

Array literal syntax is simple. It takes a list of values separated by a comma and enclosed in square brackets. Using an array literal is the easiest way to create a JavaScript Array.

**Syntax :**

```
var <array-name> = [element0, element1, element2,... elementN];  
var arrayname=[value1,value2.....valueN];
```

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

**Example: Declare and Initialize JS Array**

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

```
var numericArray = [1, 2, 3, 4];
```

```
var decimalArray = [1.1, 1.2, 1.3];
```

```
var booleanArray = [true, false, false, true];
```

```
var mixedArray = [1, "two", "three", 4];
```

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

```
<html>  
<body>  
<script>  
var emp=["Sonoo","Vimal","Ratan"];  
for (i=0;i<emp.length;i++)  
{  
document.write(emp[i] + "<br/>");  
}  
</script>  
</body>  
</html>
```

## 2) JavaScript Array Constructor (new keyword)

You can initialize an array with Array constructor syntax using **new** keyword.

The Array constructor has following three forms.

### Syntax:

```
var arrayName = new Array();           // Javascript Array Directly
```

```
var arrayName = new Array(Number length); // If you know the number of elements that the array will hold
```

```
var arrayName = new Array(element1, element2, element3,... elementN); // Create instance of array by passing arguments in constructor
```

### 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>
```

**If you know the number of elements that the array will hold, you can create an array with an initial size as shown in the following example:**

```
Var scores = new Array(10);
```

**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>
```

## **Access the Elements of an Array**

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

This statement accesses the value of the first element in cars:

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

**Example:**

```
<html>
<body>

<h2>JavaScript Arrays</h2>

<p>JavaScript array elements are accessed using numeric indexes (starting from 0).</p>

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

```
<script>
var cars = ["Saab", "Volvo", "BMW"];
document.getElementById("demo").innerHTML = cars[0];
</script>
```

```
</body>
```

```
</html>
```

## Accessing the First Array Element

```
<html>
```

```
<body>
```

```
<h2>JavaScript Arrays</h2>
```

```
<p>JavaScript array elements are accessed using numeric indexes (starting from 0).</p>
```

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

```
<script>
```

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

```
var first = fruits[0];
```

```
document.getElementById("demo").innerHTML = first;
```

```
</script>
```

```
</body>
```

```
</html>
```

## Accessing the Last Array Element

```
<html>
```

```
<body>
```

```
<h2>JavaScript Arrays</h2>
```

<p>JavaScript array elements are accessed using numeric indexes (starting from 0).</p>

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

<script>

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
var last = fruits[fruits.length-1];  
document.getElementById("demo").innerHTML = last;  
</script>
```

</body>

</html>

## **Array Properties and Methods**

The real strength of JavaScript arrays are the built-in array properties and methods:

### Examples

```
var x = cars.length; // The length property returns the number of elements  
var y = cars.sort(); // The sort() method sorts arrays
```

### **The length Property**

The length property of an array returns the length of an array (the number of array elements).

<body>

<h2>JavaScript Arrays</h2>

<p>The length property returns the length of an array.</p>

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

<script>

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
document.getElementById("demo").innerHTML = fruits.length;  
</script>
```

```
</body>
```

```
</html>
```

## Sort() Method

Sort an array

```
<html>
```

```
<body>
```

```
<p>Click the button to sort the array.</p>
```

```
<button onclick="myFunction()">Try it</button>
```

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

```
<script>
```

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

```
document.getElementById("demo").innerHTML = fruits;
```

```
function myFunction()
```

```
{
```

```
  fruits.sort();
```

```
  document.getElementById("demo").innerHTML = fruits;
```

```
}
```

```
</script>
```

```
</body>
```

```
</html>
```

## Popping Method

The pop() method removes the last element from an array:

```
<html>
```

```
<body>
```

## <h2>JavaScript Array Methods</h2>

## <h2>pop()</h2>

<p>The pop() method removes the last element from an array.</p>

<p id="demo1"></p>

<p id="demo2"></p>

<script>

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

document.getElementById("demo1").innerHTML = fruits;

fruits.pop();

document.getElementById("demo2").innerHTML = fruits;

</script>

</body>

</html>

## Pushing Method

The push() method adds a new element to an array (at the end):

<html>

<body>

## <h2>JavaScript Array Methods</h2>

## <h2>push()</h2>

<p>The push() method appends a new element to an array.</p>

<button onclick="myFunction()">Try it</button>



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

```
<script>
```

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

```
document.getElementById("demo").innerHTML = fruits;
```

```
function myFunction()
```

```
{
```

```
  fruits.push("Kiwi");
```

```
  document.getElementById("demo").innerHTML = fruits;
```

```
}
```

```
</script>
```

```
</body>
```

```
</html>
```

## Shifting Elements Method

Shifting is equivalent to popping, working on the first element instead of the last.

The shift() method removes the first array element and "shifts" all other elements to a lower index.

```
<html>
```

```
<body>
```

```
<h2>JavaScript Array Methods</h2>
```

```
<h2>shift()</h2>
```

<p>The shift() method removes the first element of an array (and "shifts" all other elements to the left):</p>

```
<p id="demo1"></p>
```

```
<p id="demo2"></p>
```

```
<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = fruits;
fruits.shift();
document.getElementById("demo2").innerHTML = fruits;
</script>

</body>
</html>
```

## Using splice() to Remove Elements

With clever parameter setting, you can use splice() to remove elements without leaving "holes" in the array:

```
<html>
<body>

<h2>JavaScript Array Methods</h2>

<h2>splice()</h2>

<p>The splice() methods can be used to remove array elements.</p>

<button onclick="myFunction()">Try it</button>

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

<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits;
function myFunction() {
  fruits.splice(0, 1);
  document.getElementById("demo").innerHTML = fruits;
```

```
}  
</script>
```

```
</body>
```

```
</html>
```

```
<html>
```

```
<body>
```

## Merging (Concatenating) Arrays

The `concat()` method creates a new array by merging (concatenating) existing arrays:

```
<h2>concat()</h2>
```

```
<p>The concat() method is used to merge (concatenate) arrays:</p>
```

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

```
<script>
```

```
var myGirls = ["Cecilie", "Lone"];
```

```
var myBoys = ["Emil", "Tobias", "Linus"];
```

```
var myChildren = myGirls.concat(myBoys);
```

```
document.getElementById("demo").innerHTML = myChildren;
```

```
</script>
```

```
</body>
```

```
</html>
```

## Reverse() Method

Reverse the order of the elements in an array:

```
<html>
```

```
<body>
```

<p>Click the button to reverse the order of the elements in the array.</p>

<button onclick="myFunction()">Try it</button>

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

<script>

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

document.getElementById("demo").innerHTML = fruits;

function myFunction()

{

fruits.reverse();

document.getElementById("demo").innerHTML = fruits;

}

</script>

</body>

</html>

## **Examples of Array:**

<html>

<body>

<script type="text/javascript">

var marks= [30,20,50]

for(i=0;i<marks.length;i++)

{

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

}

</script>

</body>

</html>

## **Example 2 of Array:**

```

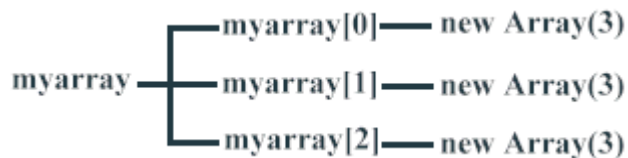
<html>
<body>
<script type="text/javascript">
var marks= new Array(3)
for(i=0;i<marks.length;i++)
    {
        marks[i]=parseInt(prompt("enter marks",0))
    }
for(i=0;i<marks.length;i++)
    {
        document.write(marks[i]+"<br>")
    }
</script>
</body>
</html>

```

## Two dimensional JavaScript Array

We can create two dimensional arrays in JavaScript.

To start things off, lets first illustrate graphically how a two dimensional array is represented:



As you can see, to manually add a dimension to an array, what we need to do is declare a new array on top of each individual array "stem". The translation of this idea to actual codes, as you will see, is actually very simple:

```

var myarray=new Array(3)
for (i=0; i <3; i++)
    myarray[i]=new Array(3)

```

### **EXAMPLE:**

```

<html>
<body>
<script type="text/javascript">
var marks= new Array()

```

```
for(i=0;i<3;i++)
{
    for(j=0;j<3;j++)
    {
        marks[i]=new Array(3)
        marks[i][j]=parseInt(prompt("enter marks"))

    }
}

var start=1
for(i=0;i<3;i++)
{
    for(j=0;j<3;j++)
    {
        marks[i][j]=start
        start=start+1
    }
}

for(i=0;i<3;i++)
{
    for(j=0;j<3;j++)
    {
        document.write(marks[i][j]+"&nbsp;")
    }

    document.write("<br>")
}

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