

Arrays

An array is literally an array where you can store as few or as many items as you like and retrieve one of, some of, or all of those items.

There are different ways to create an array. Manly, to create an array using an array initializer, first create an undefined variable, then add the different values to the array using a comma separated list wrapped in square brackets.

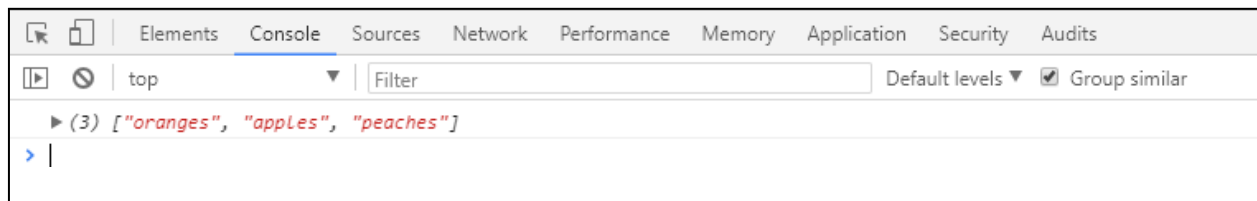
```
var fruits;  
fruits = ['oranges', 'apples', 'peaches']; //Create an array using square brackets  
console.log(fruits);
```

Also, instead of a two lines code, you can write the code in one line of code

```
var fruits=['oranges','apples','peaches']; //Create an array using square brackets  
console.log(fruits);
```

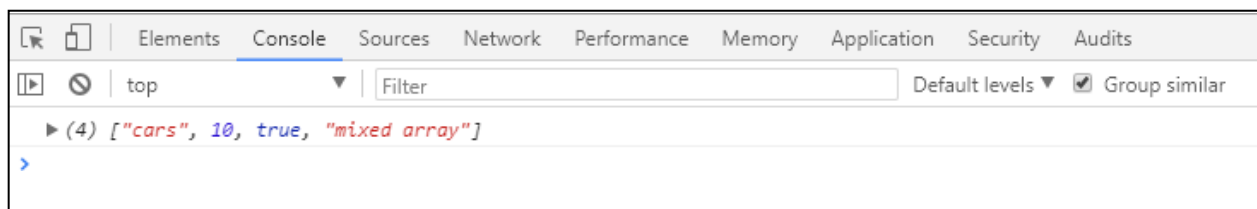
Arrays can also be created by using a constructor **new Array** with an single number parameter

```
fruits = new Array ('oranges', 'apples', 'peaches');  
console.log(fruits);
```



One interesting note about arrays each value in the array can be a separate data type so you can mix strings, Booleans, and numbers all you like.

```
var mixedArray =['cars', 10, true, 'mixed array'];  
console.log(mixedArray);
```



Accessing array elements

Each of the values added to an array is placed in a separate slot and given an index number. **Computer indexes start with zero** so the first item has index number zero, the second number is indexed one, and so on. Once the array is created, you can retrieve a specific value from the array based on its index.

```
fruits = new Array ('oranges', 'apples', 'peaches');  
var x = fruits[2]; // Store string "peaches" in variable x
```

If you need to change a value in the array you simply refer to its index position and assign a new value.

```
fruits = new Array ('oranges', 'apples', 'peaches');  
fruits[0] = "cherries"; // replace the string "oranges" with "cherries"
```

Subarrays

You can also create arrays inside of an array. To do so, we just need to open a square brackets inside of a square brackets:

Syntax

```
var meals= ['breakfast',['egg','bread','juice'], 'lunch',['fish','salad']];
```

```
meals  
▶ (4) ['breakfast', Array(3), 'lunch', Array(2)]
```

```
var countries = ['America',['Canada', 'USA', 'Mexico', 'Brasil', 'Argentina'],  
'Europe',['France', 'England', 'Italy', 'Spain', 'Germany','Switzerland'],  
'Asia',['China', 'India', 'Japan', 'Korean', 'Singapore'] ];
```

```
countries  
▶ (6) ['America', Array(5), 'Europe', Array(6), 'Asia', Array(5)]
```

To retrieve information from a subarray, we write square brackets of the array position follow by the square bracket of the subarray index.

```
countries[1][3];
```

```
> countries[1][3];  
← 'Brasil'
```

String as array

A string can also be treated as an array. We create an string and retrieve the character by using square brackets and the index position of the character.

```
var country = "United States of America "  
country[7] → 'S'
```

```
> var country = "United States of America "  
↳ undefined  
> country[7]  
↳ 'S'
```

Also, we can use string properties such as **length** to retrieve elements out of an array

```
> var country = "United States of America"  
↳ undefined  
> country.length  
↳ 24  
> country[country.length - 1];  
↳ 'a'
```

Properties and methods in arrays

In JavaScript, objects have properties and methods. Properties are pieces of meta information about the object we can retrieve and use. Methods are functions that belong to that object. For more information about the different arrays and how to use them, go to [w3schools.com](https://www.w3schools.com) or developer.mozilla.org

Direct methods

These are the methods that update the array information directly within the array

```
var mixedArray = ['cars', 10, true, 'apples'];  
console.log("Before :", mixedArray);  
//  
// Reverse method  
mixedArray.reverse();  
console.log("After reverse :", mixedArray);  
//  
// Remove the first value of the mixedArray using method "shift"  
mixedArray.shift();  
console.log("After shift method :", mixedArray);  
//  
// Add comma-separated list of values to the front of the array using method  
"unshift"  
mixedArray.unshift("NY", 280);  
console.log("After unshift method :", mixedArray);  
//  
//add comma-separated list values at the end of the array using method "push"  
mixedArray.push(350, "QCC");  
console.log("After push method :", mixedArray);
```



Methods that return values

slice creates a copy of the array and then returns it to us and typically we would place it inside a variable, or use it directly in some sort of function.

```
var fruits = ["apples", "cherries", "beaches", ];  
// create a copy of array fruits  
var fruits1 = fruits.slice();  
console.log("Array fruits: ", fruits);  
console.log("New fruits array using method slide : ", fruits1);
```

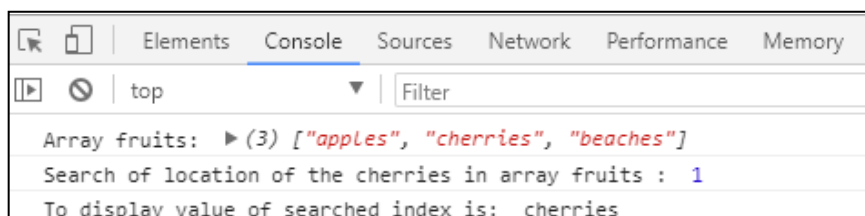


The next method, **indexOf**, gives us the index number for a specific search.

Syntax:

indexOf (value we are searching, from which index position);

```
var fruits = ["apples", "cherries", "beaches", ];  
// create a copy of array fruits  
var fruits1 = fruits.indexOf("cherries", 0); // the syntax of indexOf (value we are  
searching, from which index position)  
console.log("Array fruits: ", fruits);  
console.log("Search of location of the cherries in array fruits : ", fruits1);  
console.log("To display value of searched index is: ", fruits[fruits1]);
```



join method takes all the elements in an array, and join them together in a single string and separate them with a comma.

```
var fruits = ["apples", "cherries", "beaches", ];  
// create a copy of array fruits  
var fruits1 = fruits.join();  
console.log("Array fruits: ", fruits);  
console.log("Display elements in an array separated with a comma : ", fruits1);
```



```
var fruits1 = fruits.join(" % ");
```

```
'apples % cherries % beaches'
```

Now, notice that the commas have no space after them, so if you want to use these as actual HTML, you probably want to change that separator. You can do that by changing it in the arguments between the parentheses.

REFERENCE

Array Method	Description
concat()	Joins two or more arrays, and returns a copy of the joined arrays
copyWithin()	Copies array elements within the array, to and from specified positions
entries()	Returns a key/value pair Array Iteration Object
every()	Checks if every element in an array pass a test
fill()	Fill the elements in an array with a static value
filter()	Creates a new array with every element in an array that pass a test
find()	Returns the value of the first element in an array that pass a test
findIndex()	Returns the index of the first element in an array that pass a test
forEach()	Calls a function for each array element
from()	Creates an array from an object
includes()	Check if an array contains the specified element
indexOf()	Search the array for an element and returns its position
isArray()	Checks whether an object is an array
join()	Joins all elements of an array into a string
keys()	Returns a Array Iteration Object, containing the keys of the original array
lastIndexOf()	Search the array for an element, starting at the end, and returns its position
map()	Creates a new array with the result of calling a function for each array element
pop()	Removes the last element of an array, and returns that element
push()	Adds new elements to the end of an array, and returns the new length
reduce()	Reduce the values of an array to a single value (going left-to-right)
reduceRight()	Reduce the values of an array to a single value (going right-to-left)
reverse()	Reverses the order of the elements in an array
shift()	Removes the first element of an array, and returns that element
slice()	Selects a part of an array, and returns the new array
some()	Checks if any of the elements in an array pass a test
sort()	Sorts the elements of an array
splice()	Adds/Removes elements from an array
toString()	Converts an array to a string, and returns the result
unshift()	Adds new elements to the beginning of an array, and returns the new length
valueOf()	Returns the primitive value of an array