# JavaScript Arrays

## What is an Array

Arrays are complex variables that allow us to store more than one value or a group of values under a single variable name. JavaScript arrays can store any valid value, including strings, numbers, objects, functions, and even other arrays, thus making it possible to create more complex data structures such as an array of objects or an array of arrays.

Let's suppose you want to store the name of colors in your JavaScript code. Storing the color names one by one in a variable could look something like this:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=storing-single-values)

let color1 = "Red";

let color2 = "Green";

let color3 = "Blue";

But what happens if you need to store the state or city names of a country in variables and this time this not just three may be hundred. It is quite hard and boring to store each of them in a separate variable. Also, using so many variables simultaneously and keeping track of them all will be a very difficult task. And here array comes into play. Arrays solve this problem by providing an ordered structure for storing multiple values or a group of values.

## Creating an Array

The simplest way to create an array in JavaScript is enclosing a comma-separated list of values in square brackets ([]), as shown in the following syntax:

var myArray = [*element0*, *element1*, ..., *elementN*];

Array can also be created using the Array() constructor as shown in the following syntax. However, for the sake of simplicity previous syntax is recommended.

var myArray = new Array(*element0*, *element1*, ..., *elementN*);

Here are some examples of arrays created using array literal syntax:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=creating-arrays)

let colors = ["Red", "Green", "Blue"];

let fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

let cities = ["London", "Paris", "New York"];

let person = ["John", "Wick", 32];

**Note:** An array is an ordered collection of values. Each value in an array is called an element, and each element has a numeric position in an array, known as its index.

## Accessing the Elements of an Array

Array elements can be accessed by their index using the square bracket notation. An index is a number that represents an element's position in an array.

Array indexes are zero-based. This means that the first item of an array is stored at index 0, not 1, the second item is stored at index 1, and so on. Array indexes start at 0 and go up to the number of elements minus 1. So, array of five elements would have indexes from 0 to 4.

The following example will show you how to get individual array element by their index.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=access-individual-elements-of-an-array)

let fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

document.write(fruits[0]); // Prints: Apple

document.write(fruits[1]); // Prints: Banana

document.write(fruits[2]); // Prints: Mango

document.write(fruits[fruits.length - 1]); // Prints: Papaya

**Note:** In JavaScript, arrays are really just a special type of objects which has numeric indexes as keys. The [typeof](https://www.tutorialrepublic.com/javascript-tutorial/javascript-data-types.php" \l "typeof) operator will return "object" for arrays.

## Getting the Length of an Array

The length property returns the length of an array, which is the total number of elements contained in the array. Array length is always greater than the index of any of its element.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=get-the-length-of-an-array)

let fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

document.write(fruits.length); // Prints: 5

## Looping Through Array Elements

You can use [for](https://www.tutorialrepublic.com/javascript-tutorial/javascript-loops.php#for) loop to access each element of an array in sequential order, like this:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=loop-through-an-array-using-for-loop)

let fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

// Iterates over array elements

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

document.write(fruits[i] + "<br>"); // Print array element

}

ECMAScript 6 has introduced a simpler way to iterate over array element, which is for-of loop. In this loop you don't have to initialize and keep track of the loop counter variable (i).

Here's the same example rewritten using the for-of loop:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=loop-through-an-array-using-for-of-loop)

let fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

// Iterates over array elements

for(let fruit of fruits) {

document.write(fruit + "<br>"); // Print array element

}

You can also iterate over the array elements using for-in loop, like this:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=loop-through-an-array-using-for-in-loop)

let fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

// Loop through all the elements in the array

for(let i in fruits) {

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

}

**Note:** The for-in loop should not be used to iterate over an array where the index order is important. The for-in loop is optimized for iterating over [object's properties](https://www.tutorialrepublic.com/javascript-tutorial/javascript-objects.php), you should better use a for loop with a numeric index or for-of loop.

## Adding New Elements to an Array

To add a new element at the end of an array, simply use the push() method, like this:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=add-a-new-element-at-the-end-of-an-array)

let colors = ["Red", "Green", "Blue"];

colors.push("Yellow");

document.write(colors); // Prints: Red,Green,Blue,Yellow

document.write(colors.length); // Prints: 4

Similarly, to add a new element at the beginning of an array use the unshift() method, like this:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=add-a-new-element-at-the-beginning-of-an-array)

let colors = ["Red", "Green", "Blue"];

colors.unshift("Yellow");

document.write(colors); // Prints: Yellow,Red,Green,Blue

document.write(colors.length); // Prints: 4

You can also add multiple elements at once using the push() and unshift() methods, like this:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=add-multiple-elements-to-an-array-at-once)

let colors = ["Red", "Green", "Blue"];

colors.push("Pink", "Voilet");

colors.unshift("Yellow", "Grey");

pop and shift = remove the values

push and unshift = add the valus

document.write(colors); // Prints: Yellow,Grey,Red,Green,Blue,Pink,Voilet

document.write(colors.length); // Prints: 7

## Removing Elements from an Array

To remove the last element from an array you can use the pop() method. This method returns the value that was popped out. Here's an example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=remove-the-last-element-from-an-array)

let colors = ["Red", "Green", "Blue"];

let last = colors.pop();

document.write(last); // Prints: Blue

document.write(colors.length); // Prints: 2

Similarly, you can remove the first element from an array using the shift() method. This method also returns the value that was shifted out. Here's an example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=remove-the-first-element-from-an-array)

let colors = ["Red", "Green", "Blue"];

let first = colors.shift();

document.write(first); // Prints: Red

document.write(colors.length); // Prints: 2

**Tip:** The push() and pop() methods runs faster than unshift() and shift(). Because push() and pop() methods simply add and remove elements at the end of an array therefore the elements do not move, whereas unshift() and shift() add and remove elements at the beginning of the array that require re-indexing of whole array.

## Adding or Removing Elements at Any Position

The splice() method is a very versatile array method that allows you to add or remove elements from any index, using the syntax arr.splice(startIndex, deleteCount, elem1, ..., elemN).

This method takes three parameters: the first parameter is the index at which to start splicing the array, it is required; the second parameter is the number of elements to remove (use 0 if you don't want to remove any elements), it is optional; and the third parameter is a set of replacement elements, it is also optional. The following example shows how it works:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=add-or-remove-array-elements-at-any-index)

let colors = ["Red", "Green", "Blue"];

let removed = colors.splice(0,1); // Remove the first element

document.write(colors); // Prints: Green,Blue

document.write(removed); // Prints: Red (one item array)

document.write(removed.length); // Prints: 1

removed = colors.splice(1, 0, "Pink", "Yellow"); // Insert two items at position one

document.write(colors); // Prints: Green,Pink,Yellow,Blue

document.write(removed); // Empty array

document.write(removed.length); // Prints: 0

removed = colors.splice(1, 1, "Purple", "Voilet"); // Insert two values, remove one

document.write(colors); //Prints: Green,Purple,Voilet,Yellow,Blue

document.write(removed); // Prints: Pink (one item array)

document.write(removed.length); // Prints: 1

The splice() method returns an array of the deleted elements, or an empty array if no elements were deleted, as you can see in the above example. If the second argument is omitted, all elements from the start to the end of the array are removed. Unlike [slice()](https://www.tutorialrepublic.com/javascript-tutorial/javascript-arrays.php#slice) and [concat()](https://www.tutorialrepublic.com/javascript-tutorial/javascript-arrays.php" \l "concat) methods, the splice() method modifies the array on which it is called on.

## Creating a String from an Array

There may be situations where you simply want to create a string by joining the elements of an array. To do this you can use the join() method. This method takes an optional parameter which is a separator string that is added in between each element. If you omit the separator, then JavaScript will use comma (,) by default. The following example shows how it works:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=join-all-elements-of-an-array-into-a-string)

let colors = ["Red", "Green", "Blue"];

document.write(colors.join()); // Prints: Red,Green,Blue

document.write(colors.join("")); // Prints: RedGreenBlue

document.write(colors.join("-")); // Prints: Red-Green-Blue

document.write(colors.join(", ")); // Prints: Red, Green, Blue

You can also convert an array to a comma-separated string using the toString(). This method does not accept the separator parameter like join(). Here's an example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=convert-an-array-into-a-comma-separated-string)

let colors = ["Red", "Green", "Blue"];

document.write(colors.toString()); // Prints: Red,Green,Blue

## Extracting a Portion of an Array

If you want to extract out a portion of an array (i.e. subarray) but keep the original array intact you can use the slice() method. This method takes 2 parameters: start index (index at which to begin extraction), and an optional end index (index before which to end extraction), like arr.slice(startIndex, endIndex). Here's an example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=extract-out-a-portion-of-an-array)

let fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

let subarr = fruits.slice(1, 3);

document.write(subarr); // Prints: Banana,Mango

If endIndex parameter is omitted, all elements to the end of the array are extracted. You can also specify negative indexes or offsets —in that case the slice() method extract the elements from the end of an array, rather then the begining. For example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=extract-all-elements-from-an-array-beyond-specific-index)

let fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

document.write(fruits.slice(2)); // Prints: Mango,Orange,Papaya

document.write(fruits.slice(-2)); // Prints: Orange,Papaya

document.write(fruits.slice(2, -1)); // Prints: Mango,Orange

## Merging Two or More Arrays

The concat() method can be used to merge or combine two or more arrays. This method does not change the existing arrays, instead it returns a new array. For example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=merge-two-arrays)

let pets = ["Cat", "Dog", "Parrot"];

let wilds = ["Tiger", "Wolf", "Zebra"];

// Creating new array by combining pets and wilds arrays

let animals = pets.concat(wilds);

document.write(animals); // Prints: Cat,Dog,Parrot,Tiger,Wolf,Zebra

The concat() method can take any number of array arguments, so you can create an array from any number of other arrays, as shown in the following example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=merge-multiple-arrays)

let pets = ["Cat", "Dog", "Parrot"];

let wilds = ["Tiger", "Wolf", "Zebra"];

let bugs = ["Ant", "Bee"];

// Creating new array by combining pets, wilds and bugs arrays

let animals = pets.concat(wilds, bugs);

document.write(animals); // Prints: Cat,Dog,Parrot,Tiger,Wolf,Zebra,Ant,Bee

## Searching Through an Array

If you want to search an array for a specific value, you can simply use the indexOf() and lastIndexOf(). If the value is found, both methods return an index representing the array element. If the value is not found, -1 is returned. The indexOf() method returns the first one found, whereas the lastIndexOf() returns the last one found.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=search-an-array-for-a-specific-value)

let fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

document.write(fruits.indexOf("Apple")); // Prints: 0

document.write(fruits.indexOf("Banana")); // Prints: 1

document.write(fruits.indexOf("Pineapple")); // Prints: -1

Both methods also accept an optional integer parameter from index which specifies the index within the array at which to start the search. Here's an example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=search-an-array-for-a-specific-value-beyond-certain-index)

let arr = [1, 0, 3, 1, false, 5, 1, 4, 7];

// Searching forwards, starting at from- index

document.write(arr.indexOf(1, 2)); // Prints: 3

// Searching backwards, starting at from index

document.write(arr.lastIndexOf(1, 2)); // Prints: 0

You can also use includes() method to find out whether an array includes a certain element or not. This method takes the same parameters as indexOf() and lastIndexOf() methods, but it returns true or false instead of index number. For example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=find-whether-an-array-includes-a-certain-value)

let arr = [1, 0, 3, 1, false, 5, 1, 4, 7];

document.write(arr.includes(1)); // Prints: true

document.write(arr.includes(6)); // Prints: false

document.write(arr.includes(1, 2)); // Prints: true

document.write(arr.includes(3, 4)); // Prints: false

If you want to search an array based on certain condition then you can use the JavaScript find() method which is newly introduced in ES6. This method returns the value of the first element in the array that satisfies the provided testing function. Otherwise it return undefined.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=search-an-array-based-on-certain-condition)

let arr = [1, 0, 3, 1, false, 5, 1, 4, 7];

let result = arr.find(function(element) {

return element > 4;

});

document.write(result); // Prints: 5

There is one more method similar to find() is findIndex() method, which returns the index of a found element in the array instead of its value. For example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=search-an-array-based-on-certain-condition-and-find-index)

let arr = [1, 0, 3, 1, false, 5, 1, 4, 7];

let result = arr.findIndex(function(element) {

return element > 6;

});

document.write(result); // Prints: 8

The find() method only looks for the first element that satisfies the provided testing function. However, if you want to find out all the matched elements you can use the filter() method.

The filter() method creates a new array with all the elements that successfully passes the given test. The following example will show you how this actually works:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=filter-an-array)

let arr = [1, 0, 3, 1, false, 5, 1, 4, 7];

let result = arr.filter(function(element) {

return element > 4;

});

document.write(result); // Prints: 5,7

document.write(result.length); // Prints: 2

# JavaScript Sorting Arrays

## Sorting an Array

Sorting is a common task when working with arrays. It would be used, for instance, if you want to display the city or county names in alphabetical order.

The JavaScript Array object has a built-in method sort() for sorting array elements in alphabetical order. The following example demonstrates how it works:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=sort-an-array-alphabetically)

let fruits = ["Banana", "Orange", "Apple", "Papaya", "Mango"];

let sorted = fruits.sort();

alert(fruits); // Outputs: Apple,Banana,Mango,Orange,Papaya

alert(sorted); // Outputs: Apple,Banana,Mango,Orange,Papaya

## Reversing an Array

You can use the reverse() method to reverse the order of the elements of an array.

This method reverses an array in such a way that the first array element becomes the last, and the last array element becomes the first. Here's an example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=reverse-the-order-of-an-array)

let counts = ["one", "two", "three", "four", "five"];

let reversed = counts.reverse();

alert(counts); // Outputs: five,four,three,two,one

alert(reversed); // Output: five,four,three,two,one

**Note:** The sort() and reverse() method modifies the original array and return a reference to the same array, as you can see in the above examples.

## Sorting Numeric Arrays

The sort() method may produce unexpected result when it is applied on the numeric arrays (i.e. arrays containing numeric values). For instance:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=sort-a-numeric-array)

let numbers = [5, 20, 10, 75, 50, 100];

numbers.sort(); // Sorts numbers array

alert(numbers); // Outputs: 10,100,20,5,50,75

As you can see, the result is different from what we've expected. It happens because, the sort() method sorts the numeric array elements by converting them to strings (i.e. 20 becomes "20", 100 becomes "100", and so on), and since the first character of string "20" (i.e. "2") comes after the first character of string "100" (i.e. "1"), that's why the value 20 is sorted after the 100.

To fix this sorting problem with numeric array, you can pass a compare function, like this:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=sort-a-numeric-array-correctly-using-compare-function)

let numbers = [5, 20, 10, 75, 50, 100];

// Sorting an array using compare function

numbers.sort(function(a, b) {

return a - b;

});

alert(numbers); // Outputs: 5,10,20,50,75,100

As you can see, this time we've got the correct result. Let's see how it works.

When compare function is specified, array elements are sorted according to the return value of the compare function. For example, when comparing a and b:

* If the compare function returns a value less than 0, then a comes first.
* If the compare function returns a value greater than 0, then b comes first.
* If the compare function returns 0, a and b remain unchanged with respect to each other, but sorted with respect to all other elements.

Hence, since 5 - 20 = -15 which is less than 0, therefore 5 comes first, similarly 20 - 10 = 10 which is greater than 0, therefore 10 comes before 20, likewise 20 - 75 = -55 which is less than 0, so 20 comes before 75, similarly 50 comes before 75, and so on.

## Finding the Maximum and Minimum Value in an Array

You can use the apply() method in combination with the Math.max() and Math.min() to find the maximum and minimum value inside an array, like this:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=find-the-maximum-and-minimum-value-in-an-array)

let numbers = [3, -7, 10, 8, 15, 2];

// Defining function to find maximum value

function findMax(array) {

return Math.max.apply(null, array);

}

// Defining function to find minimum value

function findMin(array) {

return Math.min.apply(null, array);

}

alert(findMax(numbers)); // Outputs: 15

alert(findMin(numbers)); // Outputs: -7

The apply() method provides a convenient way to pass array values as arguments to a function that accepts multiple arguments in an array-like manner, but not an array (e.g. Math.max() and Math.min() methods here). So, the resulting statement Math.max.apply(null, numbers) in the example above is equivalent to the Math.max(3, -7, 10, 8, 15, 2).

## Sorting an Array of Objects

The sort() method can also be used for sorting object arrays using the compare function.

The following example will show you how to sort an array of objects by property values:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=sort-an-array-of-objects)

let persons = [

{ name: "Harry", age: 14 },

{ name: "Ethan", age: 30 },

{ name: "Peter", age: 21 },

{ name: "Clark", age: 42 },

{ name: "Alice", age: 16 }

];

// Sort by age

persons.sort(function (a, b) {

return a.age - b.age;

});

console.log(persons);

// Sort by name

persons.sort(function(a, b) {

let x = a.name.toLowerCase(); // ignore upper and lowercase

let y = b.name.toLowerCase(); // ignore upper and lowercase

if(x < y) {

return -1;

}

if(x > y) {

return 1;

}

// names must be equal

return 0;

});

console.log(persons);