JavaScript Array Methods

This section provides you with the JavaScript Array methods that allow you to manipulate arrays effectively.

Section 1. Array properties

* [length property](https://www.javascripttutorial.net/javascript-array-length/) – show you how to use the length property of an array effectively.

# JavaScript Array Length

**Summary**: in this tutorial, you’ll learn about the JavaScript Array length property and how to handle it correctly.

## What exactly is the JavaScript Array length property

By definition, the length property of an [array](https://www.javascripttutorial.net/javascript-array/) is an **unsigned, 32-bit integer** that is always **numerically greater than the highest index**in the array.

The value of the length is 232. It means that an array can hold up to 4294967296 (232) elements.

The length property behaves differently depending on the array types including dense and sparse.

### 1) Dense arrays

A dense array is an array where its elements have contiguous indexes starting at zero.

For dense arrays, you can use the length property to get the number of elements in the array. For example:

let colors = ['red', 'green', 'blue'];

console.log(colors.length); *// 3*Code language: JavaScript (javascript)

In this example, the length property returns three, which is the same as the number of elements in the colors array.

The following adds one more element to the colors array:

colors.push('yellow');

console.log(colors.length); *// 4*Code language: JavaScript (javascript)

Now, the length property of the colors array is four.

When you empty the colors array, its length is zero:

colors = [];

console.log(colors.length); *// 0*Code language: JavaScript (javascript)

### 2) Sparse arrays

A sparse array is an array whose elements don’t have contiguous indexes starting at zero.

For example, the [10,, 20, 30] is a sparse array because the indexes of its elements are 0, 2, and 3.

In a sparse array, the length property doesn’t indicate the actual number of elements. It’s a number that is greater than the highest index. For example:

let numbers = [10, , 20, 30];

console.log(numbers.length); *// 4*Code language: JavaScript (javascript)

In this example, the number of elements in the numbers array is three: 10, 20, and 30. The highest index is three. Therefore, the length property returns four.

The following adds an element to the numbers array at the index 10:

numbers[10] = 100;

console.log(numbers.length); *// 11*Code language: JavaScript (javascript)

In this example, the length property returns 11.

## Modifying JavaScript Array length property

JavaScript allows you to change the value of the array length property. By changing the value of the length, you can remove elements from the array or make the array sparse.

### 1) Empty an array

If you set length to zero, the [array will be empty](https://www.javascripttutorial.net/array/4-ways-empty-javascript-array/):

const fruits = ['Apple', 'Orange', 'Strawberry'];

fruits.length = 0;

console.log(fruits); *// []*Code language: JavaScript (javascript)

### 2) Remove elements

If you set the length property of an array to a value that is lower than the highest index, all the elements whose index is greater than or equal to the new length are removed.

The following example changes the length property of the fruits array to two, which removes the third element from the array:

const fruits = ['Apple', 'Orange', 'Strawberry'];

fruits.length = 2;

console.log(fruits); *// [ 'Apple', 'Orange' ]*Code language: JavaScript (javascript)

### 3) Make array sparse

If you set the length property of an array to a value that is higher than the highest index, the array will be spare. For example:

const fruits = ['Apple', 'Orange', 'Strawberry'];

fruits.length = 5;

console.log(fruits); *// [ 'Apple', 'Orange', 'Strawberry', <2 empty items> ]*Code language: JavaScript (javascript)  
Section 2. Adding/removing elements

* [push()](https://www.javascripttutorial.net/javascript-array-push/) – add one or more elements to the end of an array.
* push(newElement);
* push(newElement1,newElement2);
* push(newElement1,newElement2,...,newElementN);

let colors = ['red', 'green', 'blue'];

let cmyk = ['cyan', 'magenta', 'yellow', 'back'];

for (const color of cmyk) {

colors.push(color);

}

Output:

console.log(colors);

['red', 'green', 'blue', 'cyan', 'magenta', 'yellow', 'back']

let numbers = [10, 20, 30];

const length = numbers.push(40, 50);

console.log(length);

console.log(numbers);

let colors = ['red', 'green', 'blue'];

let cmyk = ['cyan', 'magenta', 'yellow', 'back'];

for (const color of cmyk) {

colors.push(color);

}

console.log(colors);

let colors = ['red', 'green', 'blue'];

let cmyk = ['cyan', 'magenta', 'yellow', 'back'];

colors.push(...cmyk);

console.log(colors);

* [unshift()](https://www.javascripttutorial.net/javascript-array-unshift/) – add one or more elements to the beginning of an array.
* let numbers = [30, 40];
* const length = numbers.unshift(20);
* console.log({ length });
* console.log({ numbers });
* Code language: JavaScript (javascript)
* Output:
* { length: 3 }
* { numbers: [ 20, 30, 40 ] }Code language: CSS (css)
* let days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri'];
* let weekends = ['Sat', 'Sun'];
* for (const weekend of weekends) {
* days.unshift(weekend);
* }
* console.log(days);Code language: JavaScript (javascript)
* Output:
* ['Sun', 'Sat', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri']
* [pop()](https://www.javascripttutorial.net/javascript-array-pop/) – remove an element from the end of an array.
* [shift()](https://www.javascripttutorial.net/javascript-array-shift/) – remove the first element from an array.
* [splice()](https://www.javascripttutorial.net/javascript-array-splice/) – manipulate elements in an array such as deleting, inserting, and replacing elements.

JavaScript Array splice: Delete, Insert, and Replace

**Summary**: This tutorial shows you how to use the JavaScript Array’s splice() method to delete existing elements, insert new elements, and replace elements in an array.

JavaScript [Array](https://www.javascripttutorial.net/javascript-array/) type provides a very powerful splice() method that allows you to insert new elements into the middle of an array. Also, you can use this method to delete and replace existing elements as well.

Deleting elements using JavaScript Array’s splice() method

To  delete elements in an array, you pass two arguments into the splice() method as follows:

Array.splice(position,num);Code language: JavaScript (javascript)

The position specifies the position of the first item to delete and the num argument determines the number of elements to delete.

The splice() method changes the original array and returns an array that contains the deleted elements.

Let’s take a look at the following example.

Suppose, you have an array scores that contains five numbers from 1 to 5.

let scores = [1, 2, 3, 4, 5];Code language: JavaScript (javascript)

The following statement deletes three elements of the scores array starting from the first element.

let deletedScores = scores.splice(0, 3);Code language: JavaScript (javascript)

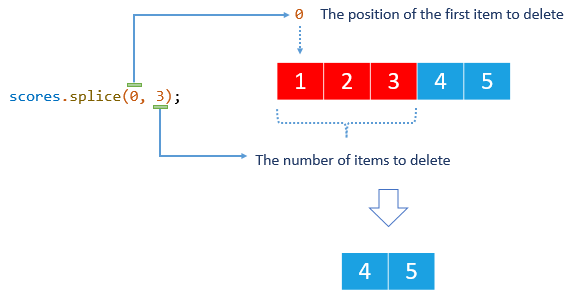
The scores array now contains two elements.

console.log(scores); *// [4, 5]*Code language: JavaScript (javascript)

And the deletedScores array contains three elements.

console.log(deletedScores); *// [1, 2, 3]*Code language: JavaScript (javascript)

The following figure illustrates the scores.splice(0,3) method call above.



Inserting elements using the JavaScript Array splice() method

You can insert one or more elements into an array by passing three or more arguments to the splice() method with the second argument is zero.

Consider the following syntax.

Array.splice(position,0,new\_element\_1,new\_element\_2,...);Code language: JavaScript (javascript)

In this syntax:

* The position specifies the starting position in the array in which the new elements will be inserted.
* The second argument is zero (0) which instructs the splice() method to not delete any array elements.
* The third argument, fourth argument, and so on are the new elements that are inserted into the array.

Note that the splice() method changes the original array. Also, the splice() method does not remove any elements, therefore, it returns an empty array. For example:

Assuming that you have an array named colors with three strings:

let colors = ['red', 'green', 'blue'];Code language: JavaScript (javascript)

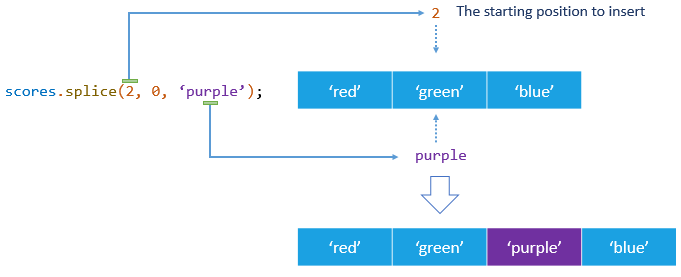
The following statement inserts one element after the second element.

colors.splice(2, 0, 'purple');Code language: JavaScript (javascript)

The colors array now has four elements with the new element inserted in the second position.

console.log(colors); *// ["red", "green", "purple", "blue"]*Code language: JavaScript (javascript)

The following figure demonstrates the method call above.



You can insert more than one element by passing the fourth argument, the fifth argument, and so on to the splice() method as in the following example.

colors.splice(1, 0, 'yellow', 'pink');

console.log(colors);

*// ["red", "yellow", "pink", "green", "purple", "blue"]*Code language: JavaScript (javascript)

Replacing elements using the JavaScript Array splice() method

The splice() method allows you to insert new elements into an array while deleting existing elements simultaneously.

To do this, you pass at least three arguments with the second one that specifies the number of items to delete and the third one that indicates the elements to insert.

Note that the number of elements to delete needs not be the same as the number of elements to insert.

Suppose you have an array of programming languages with four elements as follows:

let languages = ['C', 'C++', 'Java', 'JavaScript'];Code language: JavaScript (javascript)

The following statement replaces the second element with a new one.

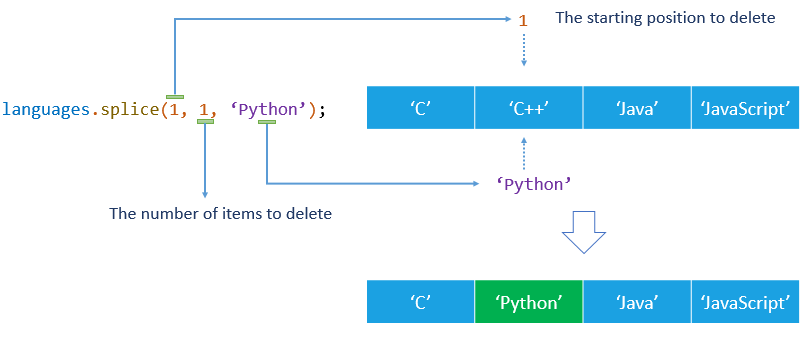
languages.splice(1, 1, 'Python');Code language: JavaScript (javascript)

The languages array now still has four elements with the new second argument is 'Python' instead of 'C++'.

console.log(languages);

*// ["C", "Python", "Java", "JavaScript"]*Code language: JavaScript (javascript)

The following figure illustrates the method call above.



You can replace one element with multiple elements by passing more arguments into the splice() method as follows:

languages.splice(2,1,'C#','Swift','Go');Code language: JavaScript (javascript)

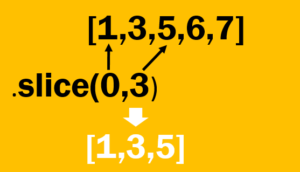
The statement deletes one element from the second element i.e., Java and inserts three new elements into the languages array. The result is as follows.

console.log(languages); *// ["C", "Python", "C#", "Swift", "Go", "JavaScript"]*Code language: JavaScript (javascript)

In this tutorial, you have learned how to use the JavaScript Array splice() method to delete existing elements, insert new elements, and replace elements in an array.

* [slice()](https://www.javascripttutorial.net/javascript-array-slice/) – copy elements of an array.

# 3 Pragmatic Uses of JavaScript Array slice() Method

The Array.prototype object provides the slice() method that allows you to extract subset elements of an [array](https://www.javascripttutorial.net/javascript-array/)and add them to the new array. In this tutorial, we will show you the practical uses of the JavaScript array slice() method.

## Introduction to JavaScript Array slice() method

The slice() method accepts two optional parameters as follows:

slice(start, stop);

Both start and stop parameters are optional.

The start parameter determines the zero-based index at which to start extraction. If the start is undefined, slice() begins at 0.

The stop parameter, as its name implies, is a zero-based index at which to end extraction. The slice() method extracts up to stop-1. It means that the slice() method doesn’t include the element at the stop position in the new array. If you omit the stop parameter, the slice() method will use the length of the array for the stop parameter.

The slice() returns a new array that contains the elements of the original array. It’s important to keep in mind that the slice() method performs the shallow copy of elements to the new array only. In addition, it doesn’t change the source array.

## Clone an array

The slice() is used to clone an array as shown in the following example:

var numbers = [1,2,3,4,5];

var newNumbers = numbers.slice();Code language: JavaScript (javascript)

In this example, the newNumbers array contains all the elements of the numbers array.

## Copy a portion of an array

The typical use of the slice() method is to copy a portion of an array without modifying the source array. Here is an example:

var colors = ['red','green','blue','yellow','purple'];

var rgb = colors.slice(0,3);

console.log(rgb); *// ["red", "green", "blue"]*

Code language: JavaScript (javascript)

The rgb array contains the first three elements of the colors array. The source array colors remains intact.

## Convert array-like objects into arrays

The slice() method is used to convert an array-like object into an array. For example:

function toArray() {

return Array.prototype.slice.call(arguments);

}

var classification = toArray('A','B','C');

console.log(classification); *// ["A", "B", "C"]*

Code language: JavaScript (javascript)

In this example, the arguments of the toArray() function is an array-like object. Inside the toArray() function, we called the slice() method to convert the arguments object into an array.

Section 3. Finding elements

* [indexOf()](https://www.javascripttutorial.net/javascript-array-indexof/) – locate an element in an array.
* [includes()](https://www.javascripttutorial.net/es-next/javascript-array-includes/) – check if an element is in an array.
* [find()](https://www.javascripttutorial.net/es6/javascript-array-find/) – find an element in an array
* [findIndex()](https://www.javascripttutorial.net/es6/javascript-array-findindex/) – find the index of an element in an array.

Section 4. High-order methods

* [map()](https://www.javascripttutorial.net/javascript-array-map/) – transform array elements.
* [filter()](https://www.javascripttutorial.net/javascript-array-filter/) – filter elements in an array.
* [reduce()](https://www.javascripttutorial.net/javascript-array-reduce/) – reduce elements of an array to a value.
* [every()](https://www.javascripttutorial.net/javascript-every/) – check if every element in an array passes a test.
* [some()](https://www.javascripttutorial.net/javascript-array-some/) – check if at least one element in an array passed a test.
* [sort()](https://www.javascripttutorial.net/javascript-array-sort/) – sort elements in an array.
* [forEach()](https://www.javascripttutorial.net/javascript-array-foreach/) – loop through array elements.

Section 5. Manipulating Arrays

* [concat()](https://www.javascripttutorial.net/javascript-array-concat/) – merge two arrays into an array.

Section 6. Creating Arrays

* [of()](https://www.javascripttutorial.net/es6/array-of/) – improve array creation.
* [from()](https://www.javascripttutorial.net/es6/array-from/) – create arrays from array-like or iterable objects.

Section 7. Flattening arrays

* [flat()](https://www.javascripttutorial.net/es-next/javascript-array-flat/) – flatten an array recursively up to a specified depth.
* [flatMap()](https://www.javascripttutorial.net/es-next/javascript-array-flatmap/) – execute a mapping function on every element and flatten the result.

Section 8. Arrays to Strings

* [join()](https://www.javascripttutorial.net/javascript-array-join/) – concatenate all elements of an array into a string separated by a separator.

Section 9. Advanced Operations

* [Destructuring](https://www.javascripttutorial.net/es6/destructuring/) – show you how to assign the elements of an array to variables.
* [Spread operator](https://www.javascripttutorial.net/es6/javascript-spread/) – learn how to use the spread operator effectively.

Section 10. Accessing elements

* [at()](https://www.javascripttutorial.net/javascript-array-at/)– access array elements using both positive and negative indexes.

Section 11. Reversing elements

* [reverse()](https://www.javascripttutorial.net/javascript-array-at/) – reverse the order of elements in place and return the same array with the elements in the reversed order.
* [toReversed()](https://www.javascripttutorial.net/javascript-array-methods/javascript-array-toreversed/) – reverse the order of elements of an array and return the new array with the elements in the reversed order.

Section 11. Multidimensional Array

* [Multidimensional Array](https://www.javascripttutorial.net/javascript-multidimensional-array/) – learn how to work with multidimensional arrays in JavaScript.