**Array.prototype.map()**

The **map()** method **creates a new array** populated with the results of calling a provided function on every element in the calling array.

const array1 = [1, 4, 9, 16];

// Pass a function to map

const map1 = array1.map(x => x \* 2);

console.log(map1);

// Expected output: Array [2, 8, 18, 32]

## [Syntax](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map#syntax)

map(callbackFn)

map(callbackFn, thisArg)

### [Parameters](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map#parameters)

callbackFn

A function to execute for each element in the array. Its return value is added as a single element in the new array. The function is called with the following arguments:

element

The current element being processed in the array.

index

The index of the current element being processed in the **array.**

array

The array map() was called upon.

thisArg Optional

A value to use as this when executing callbackFn.

### [Return value](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map#return_value)

A new array with each element being the result of the callback function.

## [Description](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map#description)

The map() method is an [iterative method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#iterative_methods). It calls a provided callbackFn function once for each element in an array and constructs a new array from the results.

callbackFn is invoked only for array indexes which have assigned values. It is not invoked for empty slots in [sparse arrays](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Indexed_collections#sparse_arrays).

The map() method is a [copying method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#copying_methods_and_mutating_methods). It does not alter this. However, the function provided as callbackFn can mutate the array. Note, however, that the length of the array is saved before the first invocation of callbackFn. Therefore:

* callbackFn will not visit any elements added beyond the array's initial length when the call to map() began.
* Changes to already-visited indexes do not cause callbackFn to be invoked on them again.
* If an existing, yet-unvisited element of the array is changed by callbackFn, its value passed to the callbackFn will be the value at the time that element gets visited. [Deleted](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/delete) elements are not visited.
* The map() method is [generic](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#generic_array_methods). It only expects the this value to have a length property and integer-keyed properties.
* Since map builds a new array, calling it without using the returned array is an anti-pattern; use [forEach](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach) or [for...of](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for...of) instead.

# Array.prototype.forEach()

The **forEach()** method executes a provided function once for each array element.

## [Syntax](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach#syntax)

forEach(callbackFn)

forEach(callbackFn, thisArg)

### [Parameters](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach#parameters)

callbackFn

A function to execute for each element in the array. Its return value is discarded. The function is called with the following arguments:

element

The current element being processed in the array.

index

The index of the current element being processed in the array.

array

The array forEach() was called upon.

thisArg Optional

A value to use as this when executing callbackFn.

### [Return value](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach#return_value)

undefined.

## [Description](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach#description)

The forEach() method is an [iterative method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#iterative_methods). It calls a provided callbackFn function once for each element in an array in ascending-index order. Unlike [map()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map), forEach() always returns [undefined](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/undefined) and is not chainable. The typical use case is to execute side effects at the end of a chain.

callbackFn is invoked only for array indexes which have assigned values. It is not invoked for empty slots in [sparse arrays](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Indexed_collections#sparse_arrays).

forEach() does not mutate the array on which it is called, but the function provided as callbackFn can. Note, however, that the length of the array is saved before the first invocation of callbackFn. Therefore:

* callbackFn will not visit any elements added beyond the array's initial length when the call to forEach() began.
* Changes to already-visited indexes do not cause callbackFn to be invoked on them again.
* If an existing, yet-unvisited element of the array is changed by callbackFn, its value passed to the callbackFn will be the value at the time that element gets visited. [Deleted](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/delete) elements are not visited.

# Array.prototype.filter()

The **filter()** method creates a [shallow copy](https://developer.mozilla.org/en-US/docs/Glossary/Shallow_copy) of a portion of a given array, filtered down to just the elements from the given array that pass the test implemented by the provided function

## [Syntax](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter#syntax)

Copy to Clipboard filter(callbackFn)

filter(callbackFn, thisArg)

### [Parameters](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter#parameters)

callbackFn

A function to execute for each element in the array. It should return a [truthy](https://developer.mozilla.org/en-US/docs/Glossary/Truthy) value to keep the element in the resulting array, and a [falsy](https://developer.mozilla.org/en-US/docs/Glossary/Falsy) value otherwise. The function is called with the following arguments:

element

The current element being processed in the array.

index

The index of the current element being processed in the array.

array

The array filter() was called upon.

thisArg Optional

A value to use as this when executing callbackFn. See [iterative methods](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#iterative_methods).

### [Return value](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter#return_value)

A [shallow copy](https://developer.mozilla.org/en-US/docs/Glossary/Shallow_copy) of a portion of the given array, filtered down to just the elements from the given array that pass the test implemented by the provided function. If no elements pass the test, an empty array will be returned.

## [Description](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter#description)

The filter() method is an [iterative method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#iterative_methods). It calls a provided callbackFn function once for each element in an array, and constructs a new array of all the values for which callbackFn returns a [truthy](https://developer.mozilla.org/en-US/docs/Glossary/Truthy) value. Array elements which do not pass the callbackFn test are not included in the new array.

callbackFn is invoked only for array indexes which have assigned values. It is not invoked for empty slots in [sparse arrays](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Indexed_collections#sparse_arrays).

The filter() method is a [copying method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#copying_methods_and_mutating_methods). It does not alter this but instead returns a [shallow copy](https://developer.mozilla.org/en-US/docs/Glossary/Shallow_copy) that contains the same elements as the ones from the original array (with some filtered out). However, the function provided as callbackFn can mutate the array. Note, however, that the length of the array is saved before the first invocation of callbackFn. Therefore:

* callbackFn will not visit any elements added beyond the array's initial length when the call to filter() began.
* Changes to already-visited indexes do not cause callbackFn to be invoked on them again.
* If an existing, yet-unvisited element of the array is changed by callbackFn, its value passed to the callbackFn will be the value at the time that element gets visited. [Deleted](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/delete) elements are not visited.

# Array.prototype.reduce()

The **reduce()** method executes a user-supplied "reducer" callback function on each element of the array, in order, passing in the return value from the calculation on the preceding element. The final result of running the reducer across all elements of the array is a single value.

The first time that the callback is run there is no "return value of the previous calculation". If supplied, an initial value may be used in its place. Otherwise the array element at index 0 is used as the initial value and iteration starts from the next element (index 1 instead of index 0).

Perhaps the easiest-to-understand case for reduce() is to return the sum of all the elements in an array

The reducer walks through the array element-by-element, at each step adding the current array value to the result from the previous step (this result is the running sum of all the previous steps) — until there are no more elements to add.

## [Syntax](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/reduce#syntax)

reduce(callbackFn)

reduce(callbackFn, initialValue)

### [Parameters](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/reduce#parameters)

callbackFn

A function to execute for each element in the array. Its return value becomes the value of the accumulator parameter on the next invocation of callbackFn. For the last invocation, the return value becomes the return value of reduce(). The function is called with the following arguments:

accumulator

The value resulting from the previous call to callbackFn. On first call, initialValue if specified, otherwise the value of array[0].

currentValue

The value of the current element. On first call, the value of array[0] if an initialValue was specified, otherwise the value of array[1].

currentIndex

The index position of currentValue in the array. On first call, 0 if initialValue was specified, otherwise 1.

array

The array reduce() was called upon.

initialValue Optional

A value to which accumulator is initialized the first time the callback is called. If initialValue is specified, callbackFn starts executing with the first value in the array as currentValue. If initialValue is not specified, accumulator is initialized to the first value in the array, and callbackFn starts executing with the second value in the array as currentValue. In this case, if the array is empty (so that there's no first value to return as accumulator), an error is thrown.

### [Return value](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/reduce#return_value)

The value that results from running the "reducer" callback function to completion over the entire array.

### [Exceptions](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/reduce#exceptions)

[TypeError](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/TypeError)

Thrown if the array contains no elements and initialValue is not provided.

## [Description](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/reduce#description)

The reduce() method is an [iterative method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#iterative_methods). It runs a "reducer" callback function over all elements in the array, in ascending-index order, and accumulates them into a single value. Every time, the return value of callbackFn is passed into callbackFn again on next invocation as accumulator. The final value of accumulator (which is the value returned from callbackFn on the final iteration of the array) becomes the return value of reduce().

callbackFn is invoked only for array indexes which have assigned values. It is not invoked for empty slots in [sparse arrays](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Indexed_collections#sparse_arrays).

Unlike other [iterative methods](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#iterative_methods), reduce() does not accept a thisArg argument. callbackFn is always called with undefined as this, which gets substituted with globalThis if callbackFn is non-strict.

reduce() is a central concept in [functional programming](https://en.wikipedia.org/wiki/Functional_programming), where it's not possible to mutate any value, so in order to accumulate all values in an array, one must return a new accumulator value on every iteration. This convention propagates to JavaScript's reduce(): you should use [spreading](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Spread_syntax) or other copying methods where possible to create new arrays and objects as the accumulator, rather than mutating the existing one. If you decided to mutate the accumulator instead of copying it, remember to still return the modified object in the callback, or the next iteration will receive undefined.

reduce() does not mutate the array on which it is called, but the function provided as callbackFn can. Note, however, that the length of the array is saved before the first invocation of callbackFn. Therefore:

* callbackFn will not visit any elements added beyond the array's initial length when the call to reduce() began.
* Changes to already-visited indexes do not cause callbackFn to be invoked on them again.
* If an existing, yet-unvisited element of the array is changed by callbackFn, its value passed to the callbackFn will be the value at the time that element gets visited. [Deleted](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/delete) elements are not visited.