mdn web docs

Array.prototype.every()

The `every()` method of `Array` instances tests whether all elements in the array pass the test implemented by the provided function. It returns a Boolean value.

Try it

JavaScript Demo: Array.every()

```
const isBelowThreshold = (currentValue) => currentValue < 40;

const array1 = [1, 30, 39, 29, 10, 13];

console.log(array1.every(isBelowThreshold));
// Expected output: true

```

Run ›

Reset

```
> true
```

Syntax

JS

```
every(callbackFn)
every(callbackFn, thisArg)
```

Parameters

`callbackFn`

A function to execute for each element in the array. It should return a truthy value to indicate the element passes the test, and a 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 `every()` was called upon.

`thisArg` Optional

A value to use as `this` when executing `callbackFn`. See iterative methods.

Return value

`true` unless `callbackFn` returns a falsy value for an array element, in which case `false` is immediately returned.

Description

The `every()` method is an iterative method. It calls a provided `callbackFn` function once for each element in an array, until the `callbackFn` returns a falsy value. If such an element is found, `every()` immediately returns `false` and stops iterating through the array. Otherwise, if `callbackFn` returns a truthy value for all elements, `every()` returns `true`. Read the iterative methods section for more information about how these methods work in general.

`every` acts like the "for all" quantifier in mathematics. In particular, for an empty array, it returns `true`. (It is vacuously true that all elements of the empty set satisfy any given condition.)

`callbackFn` is invoked only for array indexes which have assigned values. It is not invoked for empty slots in sparse arrays.

The `every()` method is generic. It only expects the `this` value to have a `length` property and integer-keyed properties.

Examples

Testing size of all array elements

The following example tests whether all elements in the array are 10 or bigger.

```
JS
function isBigEnough(element, index, array) {
  return element >= 10;
}
[12, 5, 8, 130, 44].every(isBigEnough); // false
[12, 54, 18, 130, 44].every(isBigEnough); // true
```

Check if one array is a subset of another array

The following example tests if all the elements of an array are present in another array.

```
JS
const isSubset = (array1, array2) =>
  array2.every((element) => array1.includes(element));

console.log(isSubset([1, 2, 3, 4, 5, 6, 7], [5, 7, 6])); // true
console.log(isSubset([1, 2, 3, 4, 5, 6, 7], [5, 8, 7])); // false
```

Using the third argument of callbackFn

The `array` argument is useful if you want to access another element in the array. The following example first uses `filter()` to extract the positive values and then uses `every()` to check whether the array is strictly increasing.

```
JS
const numbers = [-2, 4, -8, 16, -32];
const isIncreasing = numbers
  .filter((num) => num > 0)
  .every((num, idx, arr) => {
    // Without the arr argument, there's no way to easily access the
    // intermediate array without saving it to a variable.
    if (idx === 0) return true;
    return num > arr[idx - 1];
  });
console.log(isIncreasing); // true
```

Using every() on sparse arrays

`every()` will not run its predicate on empty slots.

```
JS
console.log([1, , 3].every((x) => x !== undefined)); // true
console.log([2, , 2].every((x) => x === 2)); // true
```

Calling every() on non-array objects

The `every()` method reads the `length` property of `this` and then accesses each property with a nonnegative integer key less than `length` until they all have been accessed or `callbackFn` returns `false`.

```
JS
const arrayLike = {
  length: 3,
  0: "a",
  1: "b",
  2: "c",
  3: 345, // ignored by every() since length is 3
};
console.log(
  Array.prototype.every.call(arrayLike, (x) => typeof x === "string"),
); // true
```

Specifications

Specification
ECMAScript Language Specification # sec-array.prototype.every

Browser compatibility

Report problems with this compatibility data on GitHub

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS
`every`	✓ Chrome 1	✓ Edge 12	✓ Firefox 1.5	✓ Opera 9.5	✓ Safari 3	✓ Chrome 18 Android	✓ Firefox 4 for Android	✓ Opera 10.1 Android	✓ Safari on iOS

Tip: you can click/tap on a cell for more information.

✓ Full support

See also

- Polyfill of `Array.prototype.every` in `core-js`
- Indexed collections guide
- `Array`
- `Array.prototype.forEach()`