# **28 Javascript Array Methods: A Cheat Sheet for Developer**

## [**Javascript Cheat Sheet (4 Part Series)**](https://dev.to/devsmitra/series/18393)

**Reference:** [28 Javascript Array Methods: A Cheat Sheet for Developer](https://dev.to/devsmitra/28-javascript-array-hacks-a-cheat-sheet-for-developer-5769)

Let's understand javascript array functions and how to use them.

#### **Array.map()**

###### **Returns a new array with the results of calling a provided function on every element in this array.**

// Code

const list = [1, 2, 3, 4];

list.map((el) => el \* 2); // [2, 4, 6, 8]

#### **Array.filter()**

###### **Returns a new array with all elements that pass the test implemented by the provided function.**

// Code

const list = [1, 2, 3, 4];

list.filter((el) => el % 2 === 0); // [2, 4]

#### **Array.reduce()**

###### **Reduce the array to a single value. The value returned by the function is stored in an accumulator (result/total).**

const list = [1, 2, 3, 4, 5];

list.reduce((total, item) => total + item, 0); // 15

#### **Array.reduceRight()**

###### **Executes a reducer function (that you provide) on each element of the array resulting in a single output value(from right to left).**

// Code

const list = [1, 2, 3, 4, 5];

list.reduceRight((total, item) => total + item, 0); // 15

#### **Array.fill()**

###### **Fill the elements in an array with a static value.**

// Code

const list = [1, 2, 3, 4, 5];

list.fill(0); // [0, 0, 0, 0, 0]

#### **Array.find()**

###### **Returns the value of the first element in the array that satisfies the provided testing function. Otherwise undefined is returned.**

// Code

const list = [1, 2, 3, 4, 5];

list.find((el) => el === 3); // 3

list.find((el) => el === 6); // undefined

#### **Array.indexOf()**

###### **Returns the first index at which a given element can be found in the array, or -1 if it is not present.**

// Code

const list = [1, 2, 3, 4, 5];

list.indexOf(3); // 2

list.indexOf(6); // -1

#### **Array.lastIndexOf()**

###### **Returns the last index at which a given element can be found in the array, or -1 if it is not present. The array is searched backwards, starting at fromIndex.**

// Code

const list = [1, 2, 3, 4, 5];

list.lastIndexOf(3); // 2

list.lastIndexOf(3, 1); // -1

#### **Array.findIndex()**

###### **Returns the index of the first element in the array that satisfies the provided testing function. Otherwise -1 is returned.**

const array = [5, 12, 8, 130, 44];

array.findIndex((element) => element > 13); // 3

#### **Array.includes()**

###### **Returns true if the given element is present in the array.**

// Code

const list = [1, 2, 3, 4, 5];

list.includes(3); // true

list.includes(6); // false

#### **Array.pop()**

###### **Removes the last element from an array and returns that element.**

// Code

const list = [1, 2, 3, 4, 5];

list.pop(); // 5

list; // [1, 2, 3, 4]

#### **Array.push()**

###### **Appends new elements to the end of an array, and returns the new length.**

// Code

const list = [1, 2, 3, 4, 5];

list.push(6); // 6

list; // [1, 2, 3, 4, 5, 6]

#### **Array.shift()**

###### **Removes the first element from an array and returns that element.**

// Code

const list = [1, 2, 3, 4, 5];

list.shift(); // returns the element.

list; // [2, 3, 4, 5]

#### **Array.unshift()**

###### **Adds new elements to the beginning of an array, and returns the new length.**

// Code

const list = [1, 2, 3, 4, 5];

list.unshift(0); // returns the length as 6.

list; // [0, 1, 2, 3, 4, 5]

#### **Array.splice()**

###### **Changes the contents of an array by removing or replacing existing elements and/or adding new elements in place.**

// Code

const list = [1, 2, 3, 4, 5];

list.splice(1, 2); // [2, 3]

list; // [1, 4, 5]

#### **Array.slice()**

###### **Returns a shallow copy of a portion of an array into a new array object selected from begin to end (end not included). The original array will not be modified.**

// Code

const list = [1, 2, 3, 4, 5];

list.slice(1, 3); // [2, 3]

list; // [1, 2, 3, 4, 5]

#### **Array.join()**

###### **Joins all elements of an array into a string.**

// Code

const list = [1, 2, 3, 4, 5];

list.join(', '); // "1, 2, 3, 4, 5"

#### **Array.reverse()**

###### **Reverses the order of the elements in an array.**

// Code

const list = [1, 2, 3, 4, 5];

list.reverse(); // [5, 4, 3, 2, 1]

list; // [5, 4, 3, 2, 1]

#### **Array.sort()**

###### **Sorts the elements of an array in place and returns the array. The default sort order is according to string Unicode code points.**

//code

const array = ['D', 'B', 'A', 'C'];

array.sort(); // ['A', 'B', 'C', 'D']

// OR

const array = [4, 1, 3, 2, 10];

array.sort(); // [1, 10, 2, 3, 4]

array.sort((a, b) => a - b); // [1, 2, 3, 4, 10]

#### **Array.some()**

###### **Returns true if at least one element in the array passes the test implemented by the provided function.**

// Code

const list = [1, 2, 3, 4, 5];

list.some((el) => el === 3); // true

list.some((el) => el === 6); // false

#### **Array.every()**

###### **Returns true if all elements in the array pass the test implemented by the provided function.**

// Code

const list = [1, 2, 3, 4, 5];

list.every((el) => el === 3); // false

const list = [2, 4, 6, 8, 10];

list.every((el) => el%2 === 0); // true

#### **Array.from()**

###### **Creates a new array from an array-like or iterable object.**

//code

const range = (n) => Array.from({ length: n }, (\_, i) => i + 1);

console.log(range(10)); // [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

#### **Array.of()**

###### **Creates a new array with a variable number of arguments, regardless of number or type of the arguments.**

// Code

const list = Array.of(1, 2, 3, 4, 5);

list; // [1, 2, 3, 4, 5]

#### **Array.isArray()**

###### **Returns true if the given value is an array.**

// Code

Array.isArray([1, 2, 3, 4, 5]); // true

Array.isArray(5); // false

#### **Array.at()**

###### **Returns a value at the specified index.**

// Code

const list = [1, 2, 3, 4, 5];

list.at(1); // 2

list.at(-1); // 5

list.at(-2); // 4

#### **Array.copyWithin()**

###### **Copies array elements within the array. Returns the modified array.**

// Code

const list = [1, 2, 3, 4, 5];

list.copyWithin(0, 3, 4); // [4, 2, 3, 4, 5]

NOTE:

* first argument is the target at which to start copying elements from.
* second argument is the index at which to start copying elements from.
* third argument is the index at which to stop copying elements from.

#### **Array.flat()**

###### **Returns a new array with all sub-array elements concatenated into it recursively up to the specified depth.**

// Code

const list = [1, 2, [3, 4, [5, 6]]];

list.flat(Infinity); // [1, 2, 3, 4, 5, 6]

#### **Array.flatMap()**

###### **Returns a new array formed by applying a given callback function to each element of the array,**

// Code

const list = [1, 2, 3];

list.flatMap((el) => [el, el \* el]); // [1, 1, 2, 4, 3, 9]