

Javascript: Array methods cheatsheet



Igor Gonchar · [Follow](#)

4 min read · Jun 8, 2021

Listen

Share

I remember how complicated it was at the beginning of my career to understand how all the Array class methods work, differ from each other and, more importantly — to keep all of them in my mind.

Recently I saw some Infographics regarding JS Array methods in the web. Taking it as a basis, I've fixed some of its errors, added another mostly used functions. And voilà! Here is my evolved Array methods cheatsheet. In such a form its understanding and remembering becomes much an easier task.

Before describing each of the methods in detail, these guidance remarks will help you at the beginning.



- each item in array, one by one



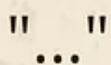
(===) - accepts callback function



() - accepts value



[...] - different arrays



- string

The main focus here, is to differ:

- methods that require a callback function as an argument from those, who expect an actual value;
- does the output creates a new array or modifies the existing one.

So, with all these key-points in mind, let's go!

Methods returning new array:

I've tried to sort the methods in the order of priority usage — the most used ones are on the top.

Creates new Array:

`[▲●■●].map(□ → ■) → [■■ ■■ ■■ ■■]`

`[▲●■●].filter(□ === ●) → [●●]`

`[▲●■●].join("-") → "▲-●-■-●"`

`[▲●].concat([■●]) → [▲●■●]`

`[▲●[■●]].flat() → [▲●■●]`

`[▲●■●●▲].slice(2, 4) → [■●]`

<https://igorgo.nl>

Map — returns a new array. It goes each element one by one and modifies them by calling a provided function.

Filter — returns a new array that will contain only those elements, for which the provided filtering function returns `true`.

Join — creates and returns a new string by concatenating all of the elements in an array, but these elements will be separated by the indicated separator.

Concat — combines two arrays and returns a third one, as a final result.

Flat — if an array contains nested ones, it will move all of them on the higher level, that will result just one new array. It's also possible to specify the needed depth of nesting.

Slice — with the provided indexes, it takes a specified section of an original array and creates a new one with only these elements. Starting index — is included, but end one — is not.

Methods mutating current array:

The key point to remember with these methods — that they are not returning a new array, as previous functions were doing. But they actually mutating the initial array.

Edits current Array:

$[\triangle \bullet \blacksquare].forEach(\text{circle with X}) \longrightarrow [\blacksquare \blacksquare \blacksquare]$

$[\triangle \bullet \blacksquare \bullet].push(\bullet) \longrightarrow [\triangle \bullet \blacksquare \bullet \bullet]$

$[\triangle \bullet \blacksquare \bullet \bullet] \xrightarrow{\text{pop()}} [\triangle \bullet \blacksquare \bullet]$

$[\text{X} \triangle \bullet \blacksquare \bullet \bullet].shift() \longrightarrow [\bullet \blacksquare \bullet \bullet]$

$[\triangle \bullet \blacksquare \bullet].sort() \longrightarrow [\triangle \bullet \bullet \blacksquare]$

$[\triangle \bullet \blacksquare \bullet].fill(\blacksquare, 1) \longrightarrow [\triangle \blacksquare \blacksquare \blacksquare]$

Push — adds an element to the end of the current array. Surprisingly, the method call will return not an updated array, but integer value — the new length of the array.

Pop — removes the last element from an array and returns the removed element itself.




Shift — removes the **first** element from an array and returns that removed element.




Sort — sorts the elements of an array in ascending order and returns this array, but already sorted. It's also possible to pass a callback function, that will describe the sorting logic.



Fill — adds or replaces all the elements of an array from a start index to an end index with a provided value.

Methods performing search:




Searches in current Array:



.find( === ) → 


0 1 2 3 .findIndex( === ) → 2


0 1 2 3 .indexOf() → 1

.some( === ) → true

.every( === ) → false

.includes() → true

Find — returns the **value** of the first element in the provided array that satisfies the provided testing function.

FindIndex — returns the **index** of the first element in the array that satisfies the provided testing function.

IndexOf — returns the first **index** at which a given element can be found in the array. The difference from *findIndex*, is that here the element/value itself is passed as an argument, instead of callback function.

Some — returns boolean value whenever **at least one** element in the array passes the provided testing callback function.

Every — returns `true` if **every** element in this array satisfies the testing function.

Includes — returns boolean if a provided element is present **at least once** in the array. The difference from *some* method —is that here the element/value itself is passed as an argument, instead of callback function.

Conclusion:

I hope this cheatsheet will be useful both: for beginners and for experienced developers. I do believe that understanding of new things and especially refreshing them in memory is much more effective with Infographics approach.

Please find below the combined list of JavaScript Array methods, that can be downloaded and used in your study.

Creates new Array:

`[▲●■●].map(⊠ → ■) → [■■■■]`

`[▲●■●].filter(⊠ === ●) → [●●]`

`[▲●■●].join("-") → "▲-●-■-●"`

`[▲●].concat([■●]) → [▲●■●]`

`[▲●[■●]].flat() → [▲●■●]`

`[0▲1●2■3●4].slice(2, 4) → [■●]`

Edits current Array:

`[▲●■].forEach(⊠ → ■) → [■■■]`

`[▲●■●].push(●) → [▲●■●●]`

`[▲●■●●].pop() → [▲●■●]`

`[▲●■●●].shift() → [●■●●]`

`[▲●■●].sort() → [▲●●■]`

`[▲●■●].fill(■, 1) → [▲■■■]`



Searches in current Array:


`[▲●■●].find(⊠ === ●) → ●`

.find() → 
.findIndex() → 2
.indexOf() → 1
.some() → true
.every() → false
.includes() → true



- each item in array, one by one

( === ) - accepts callback function

() - accepts value

[...] [...] - different arrays

Open in app ↗

Sign up

Sign In



Search Medium



JavaScript

Arrays

Array Methods

Cheatsheet



Follow