

# JavaScript Array Methods Cheat Sheet



**PART ONE**

**JS**





## Array.prototype.toString

Converts an array into a string of comma-separated array values:

```
let names = ["Zoe", "Adam", "Dan"]  
  
let strNames = names.toString() // Zoe,Adam,Dan
```





## Array.prototype.join

Similar to toString, but you can specify the separator:

```
let names = ["Zoe", "Adam", "Dan"]  
  
let strNames = names.join(" and ") // Zoe and Adam and Dan
```

JS

JS





## Array.prototype.concat

Create a new array by concatenating existing arrays:

```
let nums = [1, 2, 3]
let nums2 = [4, 5, 6]
let nums3 = [7, 8, 9]

let concatArr = nums.concat(nums2, nums3)
// [1, 2, 3, 4, 5, 6, 7, 8, 9]
```





## Array.prototype.splice

The splice() method can be used to add new items to an array:

```
let arr = ["Danny", "Joe"]  
  
arr.splice(1, 0, "Alice", "Amy")  
  
console.log(arr) // ['Danny', 'Alice', 'Amy', 'Joe']
```

JS

JS





## Array.prototype.slice

slice() slices out a piece of an array, and returns it in a new array:

```
let arr = ["Danny", "Joe", "Amy"]  
  
let slice = arr.slice(1) // ['Joe', 'Amy']
```





## Array.prototype.indexOf

Find the first index that contains a certain value (searches from left to right):

```
let arr = ["Danny", "Joe", "Amy", "Joe"]  
let index = arr.indexOf("Joe") // 1
```





## Array.prototype.lastIndexOf

Find the last index that contains a certain value (searches from right to left):

```
let arr = ["Danny", "Joe", "Amy", "Joe"]  
let index = arr.lastIndexOf("Joe") // 3
```







## Higher-order array methods in JavaScript

Array methods what is a higher-order function?

*A higher-order function is a function that accepts functions as arguments, and/or returns a function. So, higher-order functions are functions that operate on other functions.*

*In JavaScript, these methods are shared between array instances via prototypal inheritance from `Array.prototype`.*





## Array.prototype.forEach

Find the last index that contains a certain value (searches from right to left):

```
let numbers = [1, 2, 3, 4]

numbers.forEach(n => console.log(n))
// 1
// 2
// 3
// 4
```





## Array.prototype.map

Find the last index that contains a certain value (searches from right to left):

```
let numbers = [1, 2, 3, 4]

// Double all numbers
let doubledNumbers = numbers.map(n => n * 2) // [2, 4, 6, 8]

// Only double numbers at odd indexes
let doubledOddIndexNumbers = numbers.map((n, i) => {
  if (i % 2 === 1) return n * 2
  else return n
}) // [1, 4, 3, 8]
```





## Array.prototype.filter

The filter method is used to filter out array elements that fail a boolean test. Only elements that pass the test are allowed through into the new return array.

```
let cities = [
  { name: "Stokington", rivers: 3 },
  { name: "Phillydelfia", rivers: 6 },
  { name: "New Ports", rivers: 2 },
]

let moreThanTwoRivers = cities.filter(c => c.rivers > 2)
// [
//   { name: 'Stokington', rivers: 3 },
//   { name: 'Phillydelfia', rivers: 6 },
// ];
```

JS

JS



## Array.prototype.reduce

The reduce method runs the call-back function on each array element, and reduces the array down into a single value.



```
let numbers = [1, 2, 3, 4]

let total = numbers.reduce((total, currentNum) => total + currentNum) // 10
```





## Array.prototype.some

The some method checks if some array values pass a test. It returns either true or false.

```
let numbers = [4, 6, 14, 16]


let isSomeGreaterThan6 = numbers.some(n => n > 6) // true
let isSomeLessThan4 = numbers.some(n => n < 4) // false
```





## Array.prototype.every

every is similar to the some method, but checks if every value in the array passes a certain test, rather than just some.



```
let numbers = [4, 6, 14, 16]

let isEverythingGreaterThan6 = numbers.every(n => n > 6) // false
let isEverythingLessThan20 = numbers.some(n => n < 20) // true
```





## Array.prototype.flat

The `flat()` method creates a new array with all sub-array elements concatenated into it recursively up to the specified depth.

```
let arr = [1, 2, 3, [1, 2], 4]

let flatArr = arr.flat() // [1, 2, 3, 1, 2, 4]
```



JS







## Array.prototype.find

The find method returns the first element in the array that passes a certain test.

```
let stock = [  
  { item: "ketchup", quantity: 32 },  
  { item: "mayo", quantity: 9 },  
  { item: "hot sauce", quantity: 12 },  
]  
  
let mayo = stock.find(s => s.item === "mayo")  
// { item: 'mayo', quantity: 9 }
```





## Array.prototype.findIndex

Same as find, but returns the index instead of the value:

```
let stock = [  
  { item: "ketchup", quantity: 32 },  
  { item: "mayo", quantity: 9 },  
  { item: "hot sauce", quantity: 12 },  
]  
  
let mayoIndex = stock.findIndex(s => s.item === "mayo")  
// 1
```





## Array.prototype.sort

sort puts an array's elements in ascending order. It is an "in-place" sorting algorithm - meaning that it mutates the original array and returns it.

By default, sort works on strings:

```
let names = ["Zoe", "Adam", "Dan"]  
  
names.sort()  
  
console.log(names) // ['Adam', 'Dan', 'Zoe']
```



JS





## Array.prototype.sort

For numbers, we need to pass a comparison call-back function:

```
let numbers = [3, 1, 7, 2]

numbers.sort((a, b) => a - b)

console.log(numbers) // [1, 2, 3, 7]
```



JS

