

# Common Array Method Cheat Sheet

🕒 Created	@Sep 17, 2020 9:50 AM
▼ Lambda Unit 1	
🔗 Materials	
☰ Presenter	
☑ Reviewed	<input type="checkbox"/>
▼ Type	
🔗 URL	

## Introduction

The point of this is to better understand the most common JavaScript Array Methods. Below you will find an MDN Code example or an Example that I have used from toy projects, or with a code challenge.

I created this for myself but felt like sharing it would allow for others to profit off of the time that I spent on putting this together. I knew that one of my weaknesses was dealing with array methods, and I know that going into JS, React, Redux, and Node.js that I would need to become an expert.

I researched over the course of a few days what the most common array methods were (shout out to Chris Atoki for some of them as well) and then I went one by one and grabbed a code example, and the sample code. This is meant to be used in combination with other resources (thus the MDN links) to get a rapid refresh on the usage, and then more in-depth if needed.

Additional Resources:

Most Important Array Methods in JavaScript

Arrays in JavaScript are list-like objects whose prototypes have methods to perform some operations on them. In this article, we will see the

<https://www.codespot.org/javascript-most-important-array-methods/>



Common Array Methods in ES6

reduce() method executes a reducer function (that we provide) on each element of the array, resulting in a single value.


<https://medium.com/@ishan02016/most-common-array-methods-in-es6-616f41d91ea8>



<https://programmingwithmosh.com/javascript/most-common-array-operations-that-you-should-learn-in-javascript/>

### 15 Must-Know JavaScript Array Methods

Arrays are wonderful and a very particular type in JavaScript. There are many useful built-in properties and methods that will help you

 <https://livecodestream.dev/post/2020-06-05-15-must-know-javascript-array-methods/>

Scrip  
/  
ods

<https://www.youtube.com/watch?v=MsQ2zqpHxkU>

Covers: Map, forEach, Filter, Reduce

## .reverse


Example

```
(arrow) reverse = arr => arr.reverse()
```

MDN Link:

### Array.prototype.reverse()

The reverse() method reverses an array. The first array element becomes the last, and the last array element becomes the first. The reversed array. The reverse method transposes the elements of the calling array object in place,

 [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/reverse](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/reverse)



Usage:

## .length

Example

```
let myAlphabet = ['A', 'B', 'C', 'D', 'E', 'F', 'G'];  
myAlphabet.length  
//Expected return: 7
```

MDN Link:

### Array.prototype.length

The length property of an object which is an instance of type Array sets or returns the number of elements in that array. The value is an unsigned, 32-bit integer that is always numerically greater than the highest index in the array.

 [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/length](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/length)



## .forEach

Example:


```
let myAlphabet = ['A', 'B', 'C', 'D', 'E', 'F', 'G'];
myAlphabet.forEach(e => console.log(e));
```

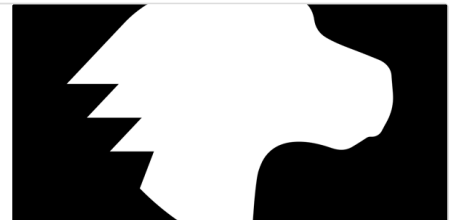
MDN Link:

### Array.prototype.forEach()

The forEach() method executes a provided function once for each array element. arr.forEach(callback(currentValue [, index [, array]]), thisArg)

forEach() calls a provided function once for each element in an array in

 [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/forEach](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach)



## .filter

Example:


```
const words = ['spray', 'limit', 'elite', 'exuberant', 'destruction', 'present'];

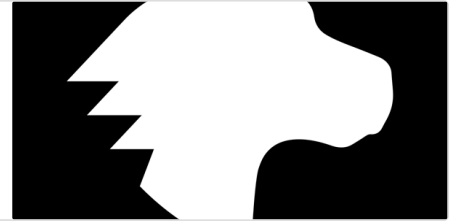
const result = words.filter(word => word.length > 6);
```

MDN Link:

### Array.prototype.filter()

The method creates a new array with all elements that pass the test implemented by the provided function. Function is a predicate, to test each element of the array. Return true to keep the element, false otherwise. It

 [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/filter](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter)



## .includes

Example:

```
const array1 = [1, 2, 3];

console.log(array1.includes(2));
// expected output: true

const pets = ['cat', 'dog', 'bat'];

console.log(pets.includes('cat'));
// expected output: true

console.log(pets.includes('at'));
// expected output: false
```

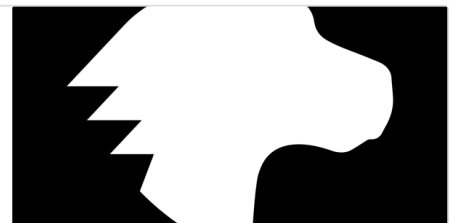
MDN Link:

### Array.prototype.includes()

The includes() method determines whether an array includes a certain value among its entries, returning true or false as appropriate.

arr.includes(valueToFind[, fromIndex]) The value to search for. Note: When

 [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/includes](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/includes)



## .join

Example:

```
const elements = ['Fire', 'Air', 'Water'];

console.log(elements.join());
// expected output: "Fire,Air,Water"

console.log(elements.join(''));
// expected output: "FireAirWater"
```

```
console.log(elements.join('-'));  
// expected output: "Fire-Air-Water"
```

MDN Link:

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/join](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/join)

## .map


Example:

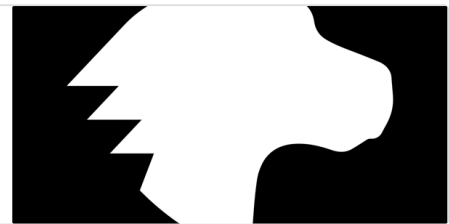
```
const array1 = [1, 4, 9, 16];  
  
// pass a function to map  
const map1 = array1.map(x => x * 2);  
  
console.log(map1);  
// expected output: Array [2, 8, 18, 32]
```

MDN Link:

### 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. `let new_array = arr.map(function callback( currentValue[, index[, array]]) { // return element for`

 [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/map](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map)



## .pop

Example:

```
const plants = ['broccoli', 'cauliflower', 'cabbage', 'kale', 'tomato'];  
  
console.log(plants.pop());  
// expected output: "tomato"  
  
console.log(plants);  
// expected output: Array ["broccoli", "cauliflower", "cabbage", "kale"]
```

```
plants.pop();

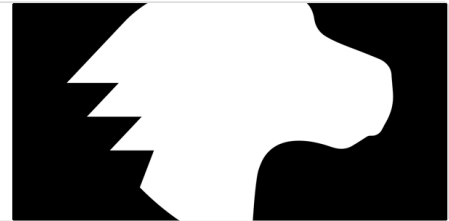
console.log(plants);
// expected output: Array ["broccoli", "cauliflower", "cabbage"]
```

MDN Link:

### Array.prototype.pop()

The pop() method removes the last element from an array and returns that element. This method changes the length of the array. arrName.pop() The removed element from the array; if the array is empty. The pop method

 [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/pop](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/pop)



## .push

Example:

```
const animals = ['pigs', 'goats', 'sheep'];


const count = animals.push('cows');
console.log(count);
// expected output: 4
console.log(animals);
// expected output: Array ["pigs", "goats", "sheep", "cows"]

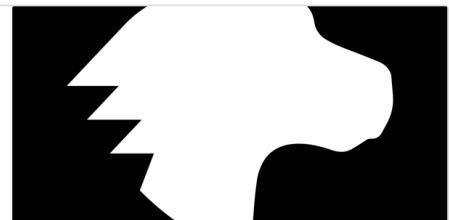
animals.push('chickens', 'cats', 'dogs');
console.log(animals);
// expected output: Array ["pigs", "goats", "sheep", "cows", "chickens", "cats", "dogs"]
```

MDN Link:

### Array.prototype.push()

The push() method adds one or more elements to the end of an array and returns the new length of the array. The element(s) to add to the end of the array. The new property of the object upon which the method was called. The

 [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/push](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/push)



## .shift

Example:

```
const array1 = [1, 2, 3];

const firstElement = array1.shift();

console.log(array1);
// expected output: Array [2, 3]

console.log(firstElement);
// expected output: 1
```

## .unshift

Example:

```
const array1 = [1, 2, 3];


console.log(array1.unshift(4, 5));
// expected output: 5

console.log(array1);
// expected output: Array [4, 5, 1, 2, 3]
```

MDN Link:

### Array.prototype.unshift()

The `unshift()` method adds one or more elements to the beginning of an array and returns the new length of the array. The new property of the object upon which the method was called. The `unshift` method inserts the given values to

 [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/unshift](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/unshift)



## .reduce

Example:

```
const array1 = [1, 2, 3, 4];
const reducer = (accumulator, currentValue) => accumulator + currentValue;

// 1 + 2 + 3 + 4
console.log(array1.reduce(reducer));
// expected output: 10
```

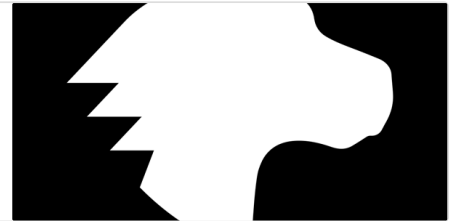
```
// 5 + 1 + 2 + 3 + 4
console.log(array1.reduce(reducer, 5));
// expected output: 15
```

MDN Link:

### Array.prototype.reduce()

The `reduce()` method executes a reducer function (that you provide) on each element of the array, resulting in single output value. The reducer function takes four arguments: Your reducer function's returned value is assigned to the

 [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/Reduce](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce)



## .slice

Example:

```
const animals = ['ant', 'bison', 'camel', 'duck', 'elephant'];

console.log(animals.slice(2));
// expected output: Array ["camel", "duck", "elephant"]

console.log(animals.slice(2, 4));
// expected output: Array ["camel", "duck"]

console.log(animals.slice(1, 5));
// expected output: Array ["bison", "camel", "duck", "elephant"]
```