**JS | Array Methods You Should Master Today -** [Shrihari Murali](https://medium.com/@shriharim006?source=post_page-----52b4e7f77981--------------------------------) Aug 4, 2022

One of the most powerful building blocks in JavaScript language is arrays. Arrays are commonly found in many programming languages and are useful for storing data.

JavaScript also includes helpful built in features known as array methods. Here are some you can learn and use it in your code.

1. filter()
2. forEach()
3. map()
4. find()
5. findIndex()
6. sort()
7. reduce()
8. some()
9. every()

Let’s pick each with an example.

**1. filter()**

Filter is used in the case where we need to filter out some values from an array.

**Example:**

const myArray = [2,4,5,7,9,12,14];  
const divisibleByTwo = myArray.filter((number) => number % 2 === 0);  
console.log(divisibleByTwo);

**2. forEach()**

The forEach() method calls a function and iterates over the elements of an array. The forEach() method can also be used on Maps and Sets. The forEach method passes a callback function for each element of an array together with the following parameters:

* Current Value (required) — The value of the current array element
* Index (optional) — The current element’s index number
* Array (optional) — The array object to which the current element belongs

**Syntax:**

array.forEach(function(currentValue, index, array))

**Example:**

const numbers = [1, 2, 3, 4, 5];// Try these on your console to see the result.numbers.forEach(function(number) {  
 console.log(number);  
});numbers.forEach((number, index) => {  
 console.log('Index: ' + index + ' Value: ' + number);  
});numbers.forEach((number, index, array) => {  
 console.log(array);  
});

**3. map()**

map() **creates a new array from calling a function for every array element**. map() calls a function once for each array element.   
map() does not execute the function for empty elements. map() does not change the original array.

**Example:**

const arr = [3, 4, 5, 6];  
const modifiedArr = arr.map(function(element){  
 return element \*3;  
});  
console.log(modifiedArr); // [9, 12, 15, 18]

**4. find()**

ES6 introduced a new method called find() .

The find() method returns the first element in an array that satisfies a provided condition.

**Syntax:**

array.find(function(currentValue, index, array),thisValue);

* **function:** Callback function.
* **currentValue:** holds the current element.
* **index:** It is an optional parameter that holds the index of the current element.
* **array:** It is an optional parameter that holds the array object the current element belongs to.
* **thisValue:** This parameter is optional. If a value is to be passed to the function to be used as its “this” value else the value “**undefined**” will be passed as its “**this**” value.

**Example:**

const array = [5, 12, 8, 130, 44];const found = array.find(element => element > 10);console.log(found); // expected output: 12

**5. findIndex()**

The findIndex() method returns the index of the first array element that satisfies the provided test function or else returns -1.

**Syntax:**

array.findIndex(function(currentValue, index, array), thisValue)

**Example:**

const ranks = [1, 5, 7, 8, 10, 7]; const index = ranks.findIndex(rank => rank === 7); console.log(index); // output :2

**6. sort()**

The sort() method sorts the elements of an array *in place* and returns the reference to the same array, now sorted; default sort order ascending.

const city = ["California", "Barcelona", "Paris", "Kathmandu"];// sort the city array in ascending order  
const sortedArray = city.sort();console.log(sortedArray);   
// Output: [ 'Barcelona', 'California', 'Kathmandu', 'Paris' ]

**7. reduce()**

The reduce() method executes a reducer function for array element. The reduce() method returns a single value: the function's accumulated result. The reduce() method does not execute the function for empty array elements. The reduce() method does not change the original array.

**Syntax:**

array.reduce(callback(accumulator, currentValue), initialValue)

**Example**:

const message = ["JavaScript ", "is ", "fun."];  
// function to join each string elements  
function joinStrings(accumulator, currentValue) {  
 return accumulator + currentValue;  
}// reduce join each element of the string  
const joinedString = message.reduce(joinStrings);console.log(joinedString);  
  
// Output: JavaScript is fun.

**8. some()**

The some() method checks if any array elements pass a test (provided as a callback function). The some() method executes the callback function once for each array element. The some() method returns true (and stops) if the function returns true for one of the array elements.

**Syntax:**

*array*.some(*function(value, index, array), this*)

**Example:**

const array = [1, 2, 3, 4, 5];// checks whether an element is even  
const even = (element) => element % 2 === 0;console.log(array.some(even));  
// output: true

**9. every()**

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

**Syntax:**

array.every(callback(currentValue), thisArg)

The every() method takes in:

* **callback** — The function to test for each array element. It takes in:
* **currentValue** — The current element being passed from the array.
* **thisArg** (optional) — Value to use as this when executing callback. By default, it is undefined.

**Example:**

function checkAdult(age) {  
 return age >= 18;  
}  
  
const ageArray = [34, 23, 20, 26, 12];  
const check = ageArray.every(checkAdult); // false  
  
if (!check) {  
 console.log("All members must be at least 18 years of age.")  
}  
  
// using arrow function  
const check1 = ageArray.every(age => age >= 18); // false  
console.log(check1);

hope this helps to solve array problems that you will face every day in your programming.