# **10 JavaScript Array Methods Every Developer Should Know**

Introduction

JavaScript Arrays are list-like objects that provide a way to store multiple values together inside a single variable. The JavaScript Array object comes bundled with a large number of utility methods that let you perform some frequently needed operations on the arrays with minimal effort. In this article, we'll go over 10 array methods that every JavaScript developer should know:

forEach method

This method allows you to iterate over the elements of an array.

1. **const** countries = ['India', 'Russia', 'Canada', 'Australia'];
2. countries.forEach((country, i) => {
3. console.log(`${i} : ${country}`);
4. });
6. /\* -- Output
7. 0 : India
8. 1 : Russia
9. 2 : Canada
10. 3 : Australia
11. \*/

The forEach method accepts a callback function that accepts the current element that is being processed in the array. It can also accept one more additional parameter *index*that contains the index of the current element in the array.

filter method

As the name suggests filter method is used to filter the elements of a JavaScript array based on the condition provided in the callback back method. It returns a new array which contains the elements that pass the condition provided inside the callback function.

1. **const** countries = ['India', 'Russia', 'Canada', 'Australia'];
2. **const** filtered = countries.filter((country) => {
3. **return** country.charAt(0) === 'I'
4. });
6. console.log(filtered);
8. // Output:
9. //  ["India"]

In the above example, we're filtering the list of countries which starts with the letter *I*at the beginning of their names.

map method

The map method accepts a callback function that accepts the current array element and creates a new array based on the output of the callback function provided.

1. **const** countries = ['India', 'Russia', 'Canada', 'Australia'];
2. **const** updatedCountries = countries.map((country) => {
3. **return** country.toUpperCase()
4. });
6. console.log(updatedCountries);
7. // Output - ["INDIA", "RUSSIA", "CANADA", "AUSTRALIA"]

In the above example, the callback function to the map method capitalizes each element of the array and creates a new array consisting of country names in capitalized format.

findIndex method

The findIndex method accepts a callback function that contains the criteria to search the array element and returns the index of the first element that matches the criteria.

1. **const** countries = ['India', 'Russia', 'Canada', 'Australia'];
2. **const** index = countries.findIndex((country) => {
3. **return** country.length > 5;
4. });
6. console.log(index); // Output : 1

In the above code, the findIndex will return 1 as the first name that matches on criteria is on the 1st index.

some method

The some method checks if at least one of the array elements meet the criteria provided in the callback function. If the criteria is met it'll return true otherwise it'll return false.

1. **const** countries = ['India', 'Russia', 'Canada', 'Australia'];
2. **const** flag = countries.some((country) => {
3. **return** country.length > 8;
4. });
6. console.log(flag);  // Output : true

In the above code, Australia meets the requirements; i.e., its length is greater than 8. Hence the some method will return true, even if the rest of the elements fails the criteria.

every method

The every method is similar to some method. It also checks if the elements of the array meet the criteria provided by the callback function. But, unlike the some method it returns true only when all the elements of the array meet the criteria.

1. **const** countries = ['India', 'Russia', 'Canada', 'Australia'];
2. **const** flag = countries.every((country) => {
3. **return** country.length > 8;
4. });
6. console.log(flag); // Output : false

In the above code, the flag variable will contain false as the only element that passes the test is *Australia*.

includes method

The includes method checks if a specific value exists in the array. If the value exists it returns true otherwise it'll return false.

1. **const** countries = ['India', 'Russia', 'Canada', 'Australia'];
2. console.log(countries.includes('India')); // Output : true
3. console.log(countries.includes('Japan')); // Output : false

In the above code, the first console.log will output true while the second will produce false. As the value *India*exists in the array while *Japan*does not.

reverse method

The reverse method simply reverses the position of elements in the array. One thing to keep it in mind is it mutates the existing array and returns the reference to that array.

1. **const** countries = ['India', 'Russia', 'Canada', 'Australia'];
2. console.log(countries.reverse());
3. // Output: ["Australia", "Canada", "Russia", "India"]

concat method

The concat method could be used to merge two arrays.

1. **const** countries = ['India', 'Russia', 'Canada', 'Australia'];
2. **const** moreCountries = ['Japan', 'Nepal'];
3. **const** allCountries = countries.concat(moreCountries);
4. console.log(allCountries);
6. // Output: ["India", "Russia", "Canada", "Australia", "Japan", "Nepal"]

In the above code snippet, the concat method is simply adding Japan and Nepal at the end of the first array i.e. countries, and returning the new array which is being stored inside the allCountries variable.

splice method

The splice method is used to add, remove or update the elements of a JavaScript array. Let's understand this in detail with some examples:

1. **const** countries = ['Japan', 'Russia', 'Canada', 'Australia'];
2. countries.splice(0, 1);
3. console.log(countries);
5. // Output: ["Russia", "Canada", "Australia"]

In the above example, we're creating a countries array with 4 values. Then we're using the splice method to delete the first element of the array. The first parameter 0 defines the index from where the splice method will start deleting elements and the second parameter one indicates the number of elements that need to be deleted. Since we've specified 1 it'll delete only the first element. If we would have specified 2 as the second parameter, it would have deleted the 2 elements from the array starting from index 0 i.e. Japan and Russia.

You could also add new elements to the array using the splice method.

1. **const** countries = ['Japan', 'Russia', 'Canada', 'Australia'];
2. countries.splice(0, 1, 'India')
3. console.log(countries);
5. // Output: ["India", "Russia", "Canada", "Australia"]

The above code will first remove the first element from the array and then will return a new element mentioned as the third parameter in that place.

And that's it!

I hope you enjoyed this article. In case you've any queries or feedback regarding this article, please do let me know in the comment section.