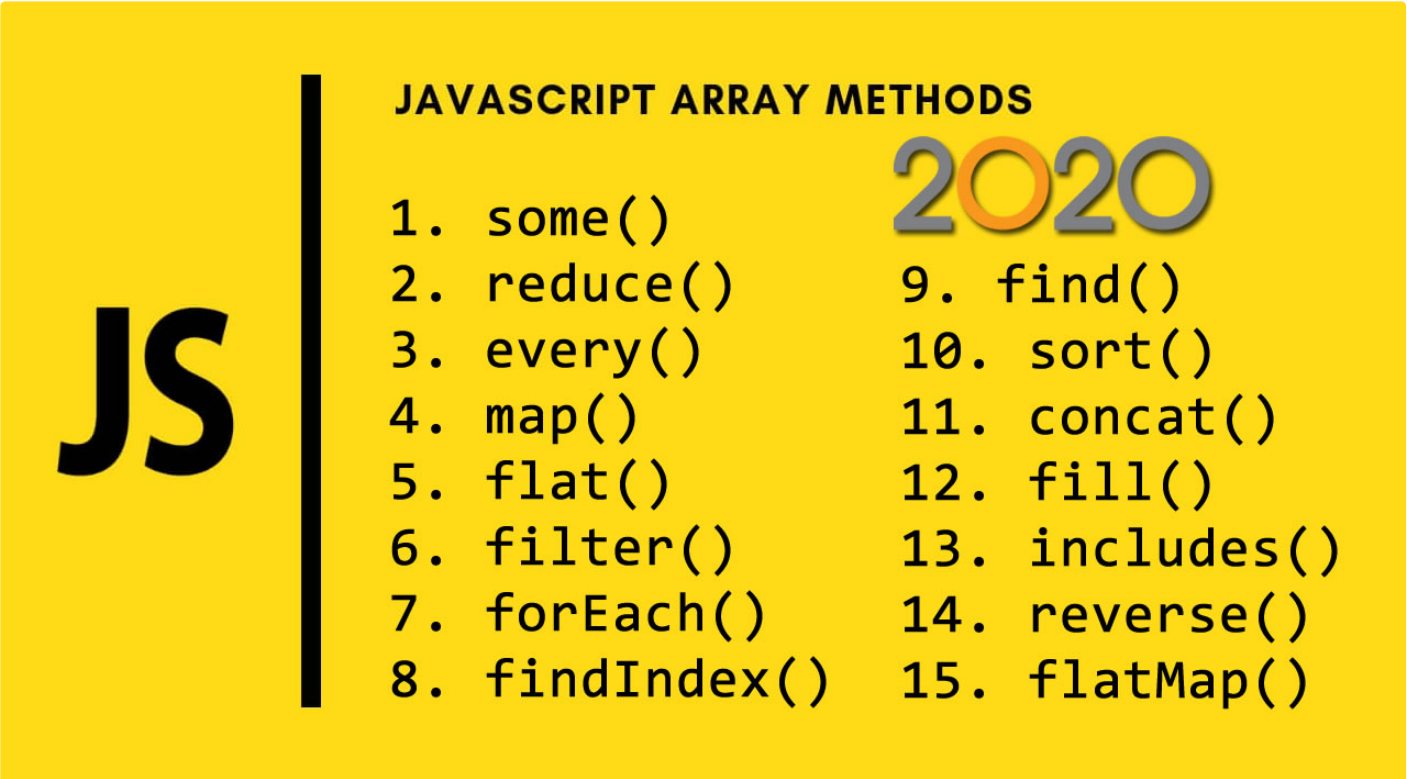

15 must-know JavaScript Array methods in 2020

2020-03-02 04:55 PM 1289

In this JavaScript Array tutorial, we're going to see 15 array methods that can help you manipulate your data properly: some; reduce; every; map; flat; filter; forEach; findIndex; find; sort; concat; fill; includes; reverse; flatMap

In JavaScript, an array is a special variable that is used to store different elements. It has some built-in properties and methods we can use to add, remove, iterate, or manipulate data following our needs. And knowing JavaScript array methods can lift your skills as a developer.

Notice that mostly in this post, we'll simplify the function passed as parameter.

```
// Instead of using this way
myAwesomeArray.some(test => {
  if (test === "d") {
    return test
  }
})
// We'll use the shorter one
myAwesomeArray.some(test => test === "d")
```

Read Also: Top 10 JavaScript array methods you should know

1. some()

This method tests the array with a function passed as a parameter. It will return `true` if at least one element matches the test and `false` for the opposite.

```
const myAwesomeArray = ["a", "b", "c", "d", "e"]

myAwesomeArray.some(test => test === "d")
//-------> Output : true
```

2. reduce()

This method receives a function which has an accumulator and a value as an argument. It applies the function to the accumulator and each value in the array to return at the end just a single value.

```
const myAwesomeArray = [1, 2, 3, 4, 5]

myAwesomeArray.reduce((total, value) => total * value)
// 1 * 2 * 3 * 4 * 5
//-------> Output = 120
```

3. every()

This method tests the array with a function passed as a parameter. It will return `true` if each element of the array match the test and `false` for the opposite.

```
const myAwesomeArray = ["a", "b", "c", "d", "e"]

myAwesomeArray.every(test => test === "d")
//-------> Output : false

const myAwesomeArray2 = ["a", "a", "a", "a", "a"]

myAwesomeArray2.every(test => test === "a")
//-------> Output : true
```

4. map()

This method receives a function as a parameter. And return a new array that contains an image of each element of the array. It will always return the same amount of items.

```
const myAwesomeArray = [5, 4, 3, 2, 1]
myAwesomeArray.map(x => x * x)

//-------> Output : 25
//                  16
//                  9
//                  4
//                  1
```

5. flat()

This method creates a new array that contains the elements holden on the sub-array and flat it into the new array. Notice that, this method will go only one level depth.

```
const myAwesomeArray = [[1, 2], [3, 4], 5]

myAwesomeArray.flat()
//-------> Output : [1, 2, 3, 4, 5]
```

6. filter()

This method receives a function as a parameter. And return a new array that contains all the elements of the array for which the filtering function passed as argument returns `true`.

```
const myAwesomeArray = [
  { id: 1, name: "john" },
  { id: 2, name: "Ali" },
  { id: 3, name: "Mass" },
  { id: 4, name: "Mass" },
]

myAwesomeArray.filter(element => element.name === "Mass")
//-------> Output : 0:{id: 3, name: "Mass"},
//                  1:{id: 4, name: "Mass"}
```

7. forEach()

This method applies a function to each element of the array.

```
const myAwesomeArray = [
  { id: 1, name: "john" },
  { id: 2, name: "Ali" },
  { id: 3, name: "Mass" },
]

myAwesomeArray.forEach(element => console.log(element.name))
//-------> Output : john
//                  Ali
//                  Mass
```

8. findIndex()

This method receives a function as a parameter and will apply it to the array. It returns the index of an element found and which satisfies the test function passed as an argument or `-1` if none satisfies it.

```
const myAwesomeArray = [
  { id: 1, name: "john" },
  { id: 2, name: "Ali" },
  { id: 3, name: "Mass" },
]

myAwesomeArray.findIndex(element => element.id === 3)
//-------> Output : 2

myAwesomeArray.findIndex(element => element.id === 7)
//-------> Output : -1
```

9. find()

This method receives a function as an argument and will apply it to the array. It returns the value of an element found in the array and which satisfies the test function. Otherwise, it returns `undefined`.

```
const myAwesomeArray = [
  { id: 1, name: "john" },
  { id: 2, name: "Ali" },
  { id: 3, name: "Mass" },
]

myAwesomeArray.find(element => element.id === 3)
//-------> Output : {id: 3, name: "Mass"}

myAwesomeArray.find(element => element.id === 7)
//-------> Output : undefined
```

10. sort()

This method receives a function as a parameter. It sorts the elements of an array and returns it.

```
const myAwesomeArray = [5, 4, 3, 2, 1]

// Sort from smallest to largest
myAwesomeArray.sort((a, b) => a - b)
//-------> Output : [1, 2, 3, 4, 5]

// Sort from largest to smallest
myAwesomeArray.sort((a, b) => b - a)
//-------> Output : [5, 4, 3, 2, 1]
```

11. concat()

This method will merge two or more arrays/values by concatenating it. It returns a new array with the elements.

```
const myAwesomeArray = [1, 2, 3, 4, 5]
const myAwesomeArray2 = [10, 20, 30, 40, 50]
myAwesomeArray.concat(myAwesomeArray2)
//-------> Output : [1, 2, 3, 4, 5, 10, 20, 30, 40, 50]
```

12. fill()

This method fills all the elements of a given array with the same value, from a start index (default 0) to an end index (default array.length).

```
const myAwesomeArray = [1, 2, 3, 4, 5]

// The first argument (0) is the value
// The second argument (1) is the starting index
// The third argument (3) is the ending index
myAwesomeArray.fill(0, 1, 3)
//-------> Output : [1, 0, 0, 4, 5]
```

13. includes()

This method will return `true` if the array contains a certain element, and `false` if not.

```
const myAwesomeArray = [1, 2, 3, 4, 5]

myAwesomeArray.includes(3)
//-------> Output : true

myAwesomeArray.includes(8)
//-------> Output : false
```

14. reverse()

This method reverses an array. The first element becomes the last, and the last element will be the first.

```
const myAwesomeArray = ["e", "d", "c", "b", "a"]

myAwesomeArray.reverse()
//-------> Output : ['a', 'b', 'c', 'd', 'e']
```

15. flatMap()

The method applies a function to each element of the array and then flatten the result into an array. It combines `flat()` and `map()` in one function.

```
const myAwesomeArray = [[1], [2], [3], [4], [5]]

myAwesomeArray.flatMap(arr => arr * 10)
//-------> Output : [10, 20, 30, 40, 50]

// With .flat() and .map()
myAwesomeArray.flat().map(arr => arr * 10)
//-------> Output : [10, 20, 30, 40, 50]
```