## 3. More Array Methods

The

`map()` , `forEach()` , `filter()` and `reverse()` are some of the most commonly used array methods in JavaScript.

### 3.1 Map()

1. The `map()` method creates a new array with the results of calling a function for every array element.
2. The `map()` method calls the provided function once for each element in an array, in order.

**Syntax :**

`array.map(callback(currentValue, index, arr))`

1. Here the callback is a function that is called for every element of array.
2. **currentValue** is the value of the current element and **index** is the array index of the current element. Here index and arr are optional arguments.

JAVASCRIPT

```
const numbers = [1, 2, 3, 4];
const result = numbers.map((number) => number * number);

console.log(result); // [1, 4, 9, 16]
```

### 3.2 forEach()

The

`forEach()` method executes a provided function once for each array element. It always returns undefined.

**Syntax :**

`array.forEach(callback(currentValue, index, arr))`

Here index and arr are optional arguments.

JAVASCRIPT

```
let fruits = ["apple", "orange", "cherry"];

fruits.forEach((fruit) => console.log(fruit));
```

### 3.3 filter()

1. The

   `filter()` method creates a new array filled with all elements that pass the test (provided as a function).

2. A new array with the elements that pass the test will be returned. If no elements pass the test, an empty array will be returned.

**Syntax :**

`array.filter(function(currentValue, index, arr))`

Here index and arr are optional arguments.

JAVASCRIPT

```
const numbers = [1, -2, 3, -4];
const positiveNumbers = numbers.filter((number) => number > 0);

console.log(positiveNumbers); // [1, 3]
```

3.4 reduce()

The

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

**Syntax :**

`array.reduce(function(accumulator, currentValue, index, arr), initialValue)`

Here

`accumulator` is the initialValue or the previously returned value of the function and `currentValue` is the value of the current element, index and arr are optional arguments.

JAVASCRIPT

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

console.log(array1.reduce(reducer)); // 10
```

3.5 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(function(currentValue, index, arr))`

Here index and arr are optional arguments.

JAVASCRIPT

```
let array1 = [32, 33, 16, 20];
const result = array1.every((array1) => array1 < 40);

console.log(result); // true
```

3.6 some()

The

`some()` method tests whether at least one element in the array passes the test implemented by the provided function. It returns a Boolean value.

**Syntax :**

```
array.some(function(currentValue, index, arr))
```

Here index and arr are optional arguments.

JAVASCRIPT

```
const myAwesomeArray = ["a", "b", "c", "d", "e"];
const result = myAwesomeArray.some((alphabet) => alphabet === "d");

console.log(result); // true
```

3.7 reverse()

The

`reverse()` method reverses the order of the elements in an array.The first array element becomes the last, and the last array element becomes the first.

**Syntax :**

```
array.reverse()
```

JAVASCRIPT

```
const myArray = ["iBHubs", "CyberEye", "ProYuga"];
const reversedArray = myArray.reverse();

console.log(reversedArray); // ["ProYuga", "CyberEye", "iBHubs"]
```

3.8 flat()

The

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

**Syntax :**

```
let newArray = arr.flat([depth]);
```

```javascript
const arr1 = [0, 1, 2, [3, 4]];
const arr2 = [0, 1, 2, [[[3, 4]]]];

console.log(arr1.flat()); // [ 0,1,2,3,4 ]
console.log(arr2.flat(2)); // [0, 1, 2, [3, 4]]
```

4. Mutable & Immutable methods

Mutable methods will change the original array and Immutable methods won't change the original array.

| Mutable methods | Immutable methods |
| --- | --- |
| shift() | map() |
| unshift() | filter() |
| push() | reduce() |
| pop() | forEach() |
| sort() | slice() |
| reverse() | join() |
| splice(), etc. | some(), etc. |

Try out Mutable and Immutable methods in the JavaScript Code Playground.