# Map, Reduce and Filter

There are many useful built-in methods available in JavaScript to work with arrays.

## The map() Method

- The array.map() method allows you to iterate over an array and modify its elements using a specified function.

- array.map () method does not modify the original array. It returns a new array value.

For example,
let arr = [2,3, 4, 5, 6];

Now imagine you have to multiply each array's elements by 4.

You might solve using a for loop as follows:

```
let arr = [2,3, 4, 5, 6];
for (let i = 0; i < arr.length; i++){
  arr[i] = arr[i] * 4;
}
console.log(arr); // [8, 12,16, 20, 24]
```

But you can use the array.map() method to achieve the same result.
Here's an example:

```
let arr = [2, 3, 4, 5, 6];
let modifiedArr = arr.map(function(element){
    return element * 4;
});
console.log(modifiedArr); // [ 8, 12, 16, 20, 24 ]
```

# Reduce Method

- The reduce() method aims to combine the elements in a sequence and give us a reduced single value output.

Syntax:

array.reduce(function, initialVal)

The first argument of the reduce function is called a reducer function, and the second argument is an optional initial value.

**E.g. Using reduce() without initialValue**

```
let arr = [200, 450, 280, 670, 770, 435];
let result = arr.reduce((acc, cur) => acc + cur);
console.log(result);
```

The output will be:

```
2805
```

In the parameter, the first one is the accumulator, which accumulates the results of the execution from the reducer function and the second is the current element in the array.

We have to calculate the sum of all elements in the array. The reducer function executes every element of the array. Each time the reducer function executes, the result is stored in the accumulator. At last the final single value of the accumulator is returned once all the elements in the array are traversed.

Suppose the initial value is not supplied as an argument to the reduce() method. In that case, the accumulator takes the first element in the array, which becomes the accumulator's initial value, and the second element of the array will be the current value.

We called the reduce() method and only passed the first argument, the reducer function. Since there is no initial value, the accumulator grabs the first element in the array to start with, 200. The second element in the array is 450, which is the current value. The reducer function starts executing from the array's second element. It returns the result of 650 after execution, which is the sum of the accumulator and the

current value. The accumulator will start accumulating the result. When all the elements in the array are finished, a single value output is returned, totalling 2,805.

**Using reduce() with initialValue 0**

```
let arr = [200, 450, 280, 670, 770, 435];
let result = arr.reduce((acc, cur) => {
    return acc + cur
}, 0);
console.log(result);
```

Now we will solve the same problem, but this time with an initial value, we call the reduce() method on the array. The accumulator starts with the initial value, and the current value is the first value in the array, which is 200. We also added the second argument, the initial value, which is zero. This time execution of the reducer function starts from the first element in the array. It calculates and returns the sum of the accumulator and current value, which is zero plus 200, and from here onwards, the accumulator starts accumulating the return values for further executions.
Finally, after traversing all the elements inside the salaries array, an output of 28,05 is returned.

## Filter Method

- The filter function is used to filter the value inside an array.
- The array.filter() method is used to create a new array from a given array consisting of only those elements from the given array which satisfy a condition set by the argument method.

E.g. Suppose you have to filter odd values in the array.

```
let arr = [1,3,2,5,7,6];
let ans = arr.filter(isOdd);
console.log(ans);
function isOdd(x){
    return x % 2;
}
```

The output will be:

```
[ 1, 3, 5, 7 ]
```