ARRAY ITERATION METHODS

FOR EACH :

The `forEach()` method calls a function for each element in an array.

The `forEach()` method is not executed for empty elements.

Syntax

array.forEach(*function(currentValue, index, arr), thisValue*)

EX:-

```
let sum = 0;
const numbers = [65, 44, 12, 4];
numbers.forEach(myFunction);

function myFunction(item) {
  sum += item;
}
```

MAP:

`map()` creates a new array from calling a function for every array element.

`map()` does not execute the function for empty elements.

`map()` does not change the original array.

Syntax

array.map(*function(currentValue, index, arr), thisValue*)

Examples

```
const numbers = [4, 9, 16, 25];
const newArr = numbers.map(Math.sqrt)
```

MAP( ):

The `filter()` method creates a new array filled with elements that pass a test provided by a function.

The `filter()` method does not execute the function for empty elements.

The `filter()` method does not change the original array.

Syntax

array.filter(*function(currentValue, index, arr), thisValue*)

EX:

```
const ages = [32, 33, 16, 40];
const result = ages.filter(checkAdult);
function checkAdult(age) {
 return age >= 18;
}
```

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 & does not change the original array.

Syntax

array.reduce(*function(total, currentValue, currentIndex, arr), initialValue*)

EX:-

```
const numbers = [175, 50, 25];

document.getElementById("demo").innerHTML = numbers.reduce(myFunc);

function myFunc(total, num) {
  return total - num;
}
```