Array methods

1) forEach();

forEach() method is used to iterate every element in the array and perform some operations.

This method accepts three parameters.

First parameter prints the **elements** in the array.

Second parameter prints the **index values** of the elements in the array.

Third parameter prints the **array** .

Syntax:

```
let arr=[10,20,30,40,50];

arr.forEach(function(a,b,c){

    return a*2;

})
```

It only modifies the existing array.

For each method will not return anything by default.

If we assign forEach method to any variable it will print **undefined.**

2) map()

Map method also is used to iterate every element in the array and perform some operations.

This method also contains three parameters.

First parameter prints the **elements** in the array.

Second parameter prints the **index values** of the elements in the array.

Third parameter prints the **array** .

The main difference between foreach and map is when we assign a map method to a variable it returns a new array whereas foreach method prints undefined.

Syntax:

```
let arr=[1,2,3,4,5];

let res1=arr.map(function(a,b){

   console.log(a+2);

})
```

3) Filter()

Filter method is used to filter the existing array as per some conditions.

Filter method also contains three parameters.

First parameter prints the **elements** in the array.

Second parameter prints the **index values** of the elements in the array.

Third parameter prints the **array** .

When we compare the value inside the log it returns the true, false, values.

```
let array=[8,5,6,3,4,6,2,1,0,6];

let result=array.filter(function(value,ind,arr){

   return value<2;

})

 console.log(result);
```

reduce() Method :-

The **reduce()** method is a powerful higher-order function in JavaScript used for reducing the elements of an array into a single value. It executes a provided callback function once for each element in the array, resulting in a single output value. Syntax :- const result = array.reduce(callback(accumulator, currentValue [, index [, array]])[, initialValue])

Example :-

const numbers = [1, 2, 3, 4, 5]; const sum = numbers.reduce((acc, num) => acc + num, 0);

console.log(sum); // Output: 15

Aggregating values in arrays.

Example:

let arr=[1,2,3,4,5] let

res=arr.reduce((acc,value) =>{

console.log(acc,value); return

acc*value

} ,5//accumulator//

)

console.log(res) ;

Output:

// 5 1

// 5 2

// 10 3

// 30 4

// 120 5

// 600

reduceRight() Method :-

The **reduceRight()** method is similar to **reduce()**, but it processes the elements of an array from right to left, rather than from left to right. It executes a provided callback function once for each element in the array, resulting in a single output value.

Syntax :-

const result = array.reduceRight(callback(accumulator, currentValue [, index [,

array]])[, initialValue])

Example :-

const numbers = [1, 2, 3, 4, 5];

const reversedSum =

numbers.reduceRight((acc, num) => acc + num, 0); console.log(reversedSum);

Output: 15 (sum from right to left)

Reducing arrays from right to left.

Reduce right ();

let arr1=[1,2,3,4,5] let

res1=arr1.reduceRight((acc,value) =>{

console.log(acc,value); return acc*value

} ,5//accumulator//

)

console.log(res1)

output: It started from right side

// 5 5

// 25 4

// 100 3

// 300 2

// 600 1

// 600

find() Method :-

The **find()** method is used in JavaScript to retrieve the value of the first element in an array that satisfies a provided condition. It returns **undefined** if no such element is found.

Syntax :-

```
const result = array.find(callback(element [, index [, array]])[, thisArg])
```

Example :-

```
const numbers = [1, 2, 3, 4, 5]; const evenNumber =
numbers.find((num) => num % 2 === 0); console.log(evenNumber);
// Output: 2 (first even number found)  Finding
elements in arrays.
```

Find() method:

(it consider the condition satisfied 1st elemnt in array prints) let arr10=[10,13,15,17,21]; let res11=arr10.find((x)=>x>10) console.log(res11)

Output:13

// reverse method(just reverse the array) let arr12=[10,13,15,17,21]; let res13=arr12.reverse() console.log(res13)

//[21, 17, 15, 13, 10]

some() Method :-

some() is a method commonly used in programming languages like JavaScript. It is typically associated with arrays and is used to test whether at least one element in the array passes a test implemented by the provided function. It returns **true** if at least one element in the array passes the test, otherwise it returns **false**

Syntax :-

const result = array.some(callback(element [, index [, array]])[, thisArg])

Example :-

const numbers = [1, 2, 3, 4, 5]; const hasEven = numbers.some((num) =>

num % 2 === 0); console.log(hasEven);

// Output: true (at least one even number exists)

Checking if any element satisfies a condition.

Some()method:

// (if atleast one of element in array sastisfy the condition output will be true otherwise false)

let arr4=[10,13,15,17,21]; let

res5=arr4.some((x)=>x==24) let

res6=arr4.some((x)=>x<=24)

console.log(res5) console.log(res6)

Output:false;

Output: true;

every() Method :-

every() is another method commonly used in programming languages like JavaScript, particularly with arrays. It checks whether all elements in an array pass a certain test implemented by a provided function. It returns **true** if all elements in the array pass the test, otherwise it returns **false**. Syntax :-

const result = array.every(callback(element [, index [, array]])[, thisArg])

Example :-

const numbers = [2, 4, 6, 8, 10]; const allEven =

numbers.every((num) => num % 2 === 0); console.log(allEven);

// Output: true (all numbers are even)

Checking if all elements satisfy a condition.

Every()method:

// (must every elemnt satisfy the condition)

let arr7=[10,13,15,17,21]; let

res8=arr7.some((x)=>x==21) let

res9=arr7.some((x)=>x<=24)

console.log(res8) console.log(res9)

Output:

// true

// true

reverse() Method :-

The **reverse()** method in JavaScript is used to reverse the order of elements in an array. It mutates the original array and returns a reference to the reversed array.

Syntax :-

array.reverse()

Example :-

reverse method(just reverse the array) let

arr12=[10,13,15,17,21]; let

res13=arr12.reverse() console.log(res13) [21,

17, 15, 13, 10]

sort() Method :-

The **sort()** method in JavaScript is used to sort the elements of an array in place and returns the sorted array. By default, the elements are sorted in ascending order based on their string representations. The sorting algorithm used by **sort()** is implementation-dependent, and its behavior may differ between browsers and JavaScript engines.

Syntax :-

```
array.sort([compareFunction])
```

Example :-

```
const fruits = ['banana', 'apple', 'orange', 'grape'];

fruits.sort(); console.log(fruits);

// Output: ['apple', 'banana', 'grape', 'orange'] (sorted alphabetically)
```

Sorting arrays alphabetically or numerically.

Sort() method

```
let arr3=[7,3,2,5,1]; let
res4=arr3.sort()
console.log(res4)
```

output: [1, 2, 3, 5, 7]

```
// it not work for multiple digit number // so we used the below process let
arr32=[105,53,2,75,1]; let res33=arr32.sort((a,b)=>a-b);//(b-a for
descending ordder) console.log(res33)



// output [ 1, 2, 53, 75, 105 ] descending order ascending order

// output [ 105, 75, 53, 2, 1 ] descending order.
```