**Array methods**

**forEach() Array method**

* The forEach() method calls a function for each element in an array.
* The forEach() method is not executed for empty elements. And it return nothing
* The forEach() method takes two arguments first arguments takes as a callback function and the second arguments is thisValue (thisValue Optional. Default undefined. A value passed to the function as its this value.)
* First arguments also takes three arguments current value, index, array itself
* In JavaScript, the forEach() method does not change the original array

**Syntex.**

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

let arr=[1,2,3,4,5,6,7]

arr.forEach((value,index,array)=>{

    console.log(`Index of Array is ${index} And value of array is ${value\*2}`)

    if(arr[index]==arr.length){

        console.log(`\n Array ${array}`) // return array itself.

    }

}, this)

console.log(this)// return empty object

console.log(arr) //the forEach() method does not change the original array

let arr0=[10,20,30,40];

arr.forEach(function(val,index){

    console.log(index, "==>", val);

})

// I want to add 100 in each element of array

let arr1=[10,20,30,40];

arr1.forEach((val,index)=>{

    console.log(index, "==>", arr1[index]+100);

    console.log(index, "==>", val+100);

})

// Array contructor has prototype . using this prototype we can create custom forEach, map.. so on

// Implement custom forEach//////////////////////////////////////

let arr2=[1,2,3,4,5];// parent object

Array.prototype.customForEach=function(callback){

    const array = this // this refers to parent object

    for(let i=0; i<array.length; i++){

      callback(array[i],i,array)// call the callback and pass array element, index and array itself.

    }

}

arr2.customForEach((element, index, array)=>{// just i useed custom forEach function

  console.log(element,index)

  if(index==array.length-1){

    console.log(array)

  }

});

///////////////////////////////////////////////////////////

let array=[

    {Name:"Anurag", Age:22, State:"Bihar"},

    {Name:"Raman", Age:24, State:"Bihar"},

    {Name:"Aman", Age:23, State:"Bihar"},

]

Array.prototype.customMap=function(callback){

    const arr=this

    for(let i=0; i<arr.length;i++){

       callback(arr[0], i, arr)

    }

}

array.customMap((element,i, array)=>{

    console.log(element,i,array)

})

**map() Array method**

map method is use to create a new array with the result on some operation.

the value its callback return are used to make a new array.

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.

arr.map(value,index,arr)

 let arr=[20,30,40,50];

    let newArray=arr.map((value) =>{

        return value\*2;// we can perform various operation with return value

    });

    console.log("<br>",newArray,"<br>");

// //Array of object

// // if you want to create a new array with object value you can youse map function

let arr2=[

    {fname:"Anurag",lname:"Kumar" },

    {fname:"ravi",lname:"Kumar" },

    {fname:"Rohan",lname:"Kumar" }

];

let NewArr=arr2.map((value)=>{

    return value.fname + " " + value.lname;

})

console.log(NewArr)

let Square=[2,3,4,5,6,7];

let number=Square.map(( value,index)=>{

    return `square of Index ${index} is ${value\*value}`;

})

console.log(number);

**filter() Array method**

//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

//Example long meethod

let arr=[2,3,4,5,6,7,8,9];

let isEven=(number)=>{

    return  number%2==0;

}

let FilteredVal=arr.filter(isEven);

console.log(FilteredVal);

//Short method

let arr1=[20,30,24,34,35,28,21,22];

let getVal=arr1.filter((arrElement)=>{

     return arrElement>24

})

console.log(getVal);

**reduce() Array method**

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 got its name from the functionality it provides, which is to iterate

//and “reduce” an array's values into one value.

//The reduce() method works in a similar manner to the forEach() method, with the added ability to collect

//the result of each iteration as a single value.

The reduce() method takes two arguments first arguments takes as a callback function and the second arguments as a initial value for accumulator.

First arguments also takes four arguments accumulator,

current value, index, array itself

Accumulator: it store the previous result

//Let’s try to get the total sum of array element using the reduce() method. First, you need to call the reduce()

// method and pass two parameters to the callback function: accumulator and current value.

const arr=[10,20,30,40,50];

let toatalSum=arr.reduce((accumulator,item)=>{

    return accumulator+item;

    // Accumulator     arrayVal     Return

    //     0       +    10        = 10

    //     10      +    20        = 30

    //     30      +    30        = 60

    //     60      +    40        = 100

    //     100     +    500       = 150

}, 0)// zero is the initial value for  accumulator.

console.log(toatalSum);

|  |  |
| --- | --- |
| **Map() method** | **forEach()** |
| 1. **it return a new array** | 1. **It return nothing** |
| 1. **It not modify of old array** | 1. **It modefy the old array** |
| 1. **In map method we do chaining wit filter** | 1. **In forEach we cant chaining because it not return any new array** |
|  |  |
|  |  |