# **Array**

<https://www.w3schools.com/js/js_arrays.asp>

# Array Methods

Note: Slice and concat always return new array

toString() & join() Change Existing Array

* Both uses to convert array to String
* Both convert Array to String by separating them using comma called element separator
* Join() is specially use for customize element separator like \*, ( , &, \*
* Join() specially use for space separator (“ ”)

Const arr = [12, “Adil”, 45];

arr.toString() => 12,Adil,45

arr.join(“-”) => 12-Adil-45

push(), pop(), shift(), unshift() Change Existing Array

* Push() Add element / elements / Array in the end of an Array
* Pop() remove last element from the end
* Unshift() Add element /elements / Array in the [0] index of an Array
* Shift() remove element from [0] index of an Array
* For String use (“12”)
* For Number use (12)
* For Array use (variable)

Splice() Change Existing Array

* Splice(1,1, “Adil”, 45, arr)
* 1 = add after first element
* 1 = after adding delete next one
* “Adil” = Can add string
* 45 = Can add number
* Arr = Can add whole array as well

Concat() Create a New Array

* Concat() is as same as push()
* Can add element / elements / arrays at the end of an array
* The only difference between push() and concat() is push() make changes to existing array and concat() create a new array
* Concat() you have to first create a variable and then store data to it by using concat() if not so it will give you an error
* For String use (“12”)
* For Number use (12)
* For Array use (variable)

Const arr = [12,45,47]

Let arr2 = arr.concat(12) => [12,45,47,12]

Slice() Create a New Array

* Slice(4,7)
* 4 = cut first 4 index elements
* 7 = cut piece after 7 index
* Final result will be [4,6,7]

Const arr = [1,2,3,4,5,6,7,8,9]

Includes() & indexOf()

* First store them in variable before use
* These are also use in Strings so create a new array and change existing Array is not valid for them

Find()

* Find method is an important method of array and find the first match value or element of an array in the form of **number**
* But filter() method find all the elements matches the condition and return an **array**

<script>

let arrray = [25, 30, 45, 66, 88]

let data = arrray.find((val) => {

return val > 25

})

console.log(data, typeof data); //30 'number'

let data2 = arrray.filter((val) => {

return val > 25

})

console.log(data2, typeof data2); //[30, 45, 66, 88] 'object'

</script>

# Sorting of an Array

Sort(), reverse() Change existing Array

* Sort() is use to sort String in array, in **A-Z** order.
* Sort().reverse() is use to sort String in array, in **Z-A** order.
* Always remember function use for array has the following parameters function (index,value,array)
* Index = if you want to show index in print then you can access the index in function by using it as function parameter
* Value = elements of an Array
* Array = Array or variable of array in which array is stored

Formula to sort Numbers ***- infinity to + infinity*** Create a new array

* There are 2 methods to write a function first one is name the function and then describe it

const points = [40, 100, 1, 5, 25, 10];

function myFunction(a,b){

    return a-b

}

points.sort(myFunction);

console.log(points);

* Directly describe a function without name

const points = [40, 100, 1, 5, 25, 10];

points.sort(function(a,b){return a-b});

console.log(points);

Formula to sort Numbers ***+ infinity to – infinity*** Create New array Array

* There are 2 methods to write a function first one is naming the function and then describe it.

const points = [40, 100, 1, 5, 25, 10];

function myFunction(a,b){

    return b-a

}

points.sort(myFunction);

console.log(points);

* **Directly describe a function without name**

const points = [40, 100, 1, 5, 25, 10];

points.sort(function(a,b){return b-a});

console.log(points);

Lowest value and highest value of an array Create New array Array

* We can find min and max value through the above sorting process of numbers by
* As the above algorithm bring 100 into [0] index and 1 [last index] so

const points = [40, 100, 1, 5, 25, 10];

points.sort(function(a,b){return a-b});

console.log(points[0]); //return 1 as [0] index

&

const points = [40, 100, 1, 5, 25, 10];

points.sort(function(a,b){return b-a});

console.log(points[0]); // return 100 as [0] index

If somehow array consist of both Numbers and string values then apply this formula

const arr1 = ["adil","shafqat","younas","aqil","abeeha"]

    arr1.sort()

    const arr2 = [1,2,-3,4,50,6,72,8,9,arr1]

    function myFunction(a,b){

        return a - b

    }

    arr2.sort(myFunction)

    console.log(arr2)

# Array Iteration

Note: for and forEach loop can return each element but you can’t access values from out side for it use map() which is a high order function

* Always store values in variable, variable inside of loops
* In this case const element = points[i] + “<br>” is a variable
* Always remember function use for array has the following parameters function (index,value,array)
* If I use return statement then it will give me an Array but if I store it on variable and then log it, it behave like single element
* Index = if you want to show index in print then you can access the index in function by using it as function parameter
* Value = elements of an Array
* Array = Array or variable of array in which array is stored

Best use of iteration

//in this case you can access variable outside with all change values in it

let myArray = [1,2,3,4,5,6,7,8,9]

let text = ""

for(let x of myArray){

  text += x + "<br>"

}

document.write(text)

//in this case you can access values inside function

let myArray2 = [4,5,1,2,5,4,1]

for(let y of myArray2){

  document.write("<br>" + y + "<br>")

}

const points = [40, 100, 1, 5, 25, 10];

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

    const element = points[i] + "<br>";

    document.write(element)

}

&

const points = [40, 100, 1, 5, 25, 10];

points.forEach(function(value){

    const newVal = value + "<br>"

    document.write(newVal)

})

Map() for Array Create a new Array

* Map() method create a new array
* It reflect the desire changes for each element of an Array
* It mapping the changes of array to new Array
* In console.log() no “<br>” is needed
* In document.write() “<br>” is needed to break elements
* If I use return statement then it will give me an Array but if I store it on variable and then log it, it behaves like single element

const numbers = [45, 4, 9, 16, 250];

function myFunction(value, index, array) {

    return value\*2

    //use return statement gives an array

    //use variable gives single elements

    //if you use variable then you are unable to access values out of scoop

}

const maping = numbers.map(myFunction);

console.log(maping);

Map() for multi-value Create a new Array

let a = [45, 44, 55, 2, 3, 9, 4, 1, 6, 5, 56, "adil", "bqil", "zadil", "adeel", 45, 55, 77];

        function myFunction(value) {

            return value + "<br>"

        }

        let map = a.map(myFunction)

        document.write(map.join(" "));

Map() for Object Create a new Array

* Return all names not specific one like filter

const numbers = [

    {name:"adil",eduction:"Msc"},

    {name:"qll",eduction:"Bsc"},

    {age:45,members:4},

];

function myFunction(value){

    return value.eduction

}

let map = numbers.map(myFunction)

console.log(map);

filter() Create a new Array or array of object

* Create a new array
* Filter() is for category match
* Filter the elements which pass the test
* Muje ye waly chahiyan bus

  const numbers = [45, 4, 9, 16, 25];

    function myFunction(value){

        return value > 9

    }

    let filter = numbers.filter(myFunction)

    console.log(filter);

an other example of filter()

const companies=[

    {name:"Google",category:"Product Based",start:1981,end:2004},

    {name:"Amazon",category:"Product Based",start:1992,end:2008},

    {name:"Paytm",category:"Product Based",start:1999,end:2007},

    {name:"Coforge",category:"Service Based",start:1989,end:2010},

    {name:"Mindtree",category:"Service Based",start:1989,end:2010}

 ];

 const ages=[33,12,20,16,5,54,21,44,61,13,15,45,25,64,32];

 const data = companies.filter((value)=>{

    if (value.end===2010) {

        return true

    }

 })

 console.log(data);

sort() Create a new Array or array of object

* Sort() A-Z cover in string document
* Sort() Z-A cover in string document
* Sort() Numbers cover in string document
* It return 1 if condition true
* It return -1 if condition false
* Sort() Object
* In object sort() can be perform by attributes like we have here name, category, start , end.
* You just here to mention the attribute name and the farther case will deal sort() whether it is number ,string

    const companies=[

    {name:"Google",category:"Product Based",start:1981,end:2004},

    {name:"Amazon",category:"Product Based",start:1992,end:2008},

    {name:"Paytm",category:"Product Based",start:1999,end:2007},

    {name:"Coforge",category:"Service Based",start:1989,end:2010},

    {name:"Mindtree",category:"Service Based",start:1989,end:2010}

 ];

 const ages=[33,12,20,16,5,54,21,44,61,13,15,45,25,64,32];

 const data = companies.sort((c1,c2)=>{

    if (c1.end > c2.end ) { //as it compares obj1 and obj2

        return 1

    } else {

        return -1

    }

 })

 console.log(data);

if you want to sort object by A-Z

    const companies=[

    {name:"Google",category:"Product Based",start:1981,end:2004},

    {name:"Amazon",category:"Product Based",start:1992,end:2008},

    {name:"Paytm",category:"Product Based",start:1999,end:2007},

    {name:"Coforge",category:"Service Based",start:1989,end:2010},

    {name:"Mindtree",category:"Service Based",start:1989,end:2010}

 ];

 const ages=[33,12,20,16,5,54,21,44,61,13,15,45,25,64,32];

 const data = companies.sort((c1,c2)=>{

    if (c1.name > c2.name ) {

        return 1

    } else {

        return -1

    }

 })

 console.log(data);

reduce() Create a new Array

Note: Reduce is use only for numbers

* Reduce() is use to reduce whole array into a single value
* It take two parameters first is function and second is initial value
* Reduce function takes two parameter as well first is accumulator and second is value
* In reduce() you don’t need to write accum +=
* Just simply write return accum + value

Average , sum of array by using reduce is given below

  const numbers = [45, 4, 9, 16, 25];

    function myFunction(accu,value){

        return accu + value / numbers.length

    }

    const average = numbers.reduce(myFunction,0)

    console.log(average);

const numbers2 = [45, 4, 9, 16, 25];

    function myFunction(accu,value){

        return accu + value

    }

    const sum = numbers2.reduce(myFunction,0)

    console.log(sum);