Arrays in Javascript

In javascript size of array grows or shrinks dynamically.

Array allows to store heterogeneous data

To create a array

var a=new Array(2,3,4,"xxxxx")

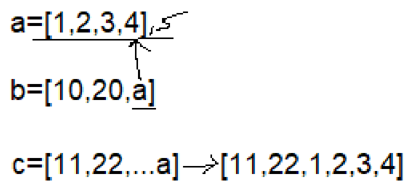
var b=[1,2,3,"xxxxx"]

to create a copy of the array

var c=[1,2,3,...a]   //... is a spread operator of array, length of c is 7

To copy the reference

var d=[1,2,3,a]   //length of d is 4



a=[1,2,3,4],5

b=[10,20,a]

c=[11,22,...a]→[11,22,1,2,3,4]

arr.push(val)	to add the value at the end of the array	arr=[12,20,100,1000,8]
		arr.push(23)
		[12,20,100,1000,8,23]

arr.pop()	to delete the value from the end	arr=[12,20,100,1000,8] var v=arr.pop() [12,20,100,1000]

arr.unshift(val)	to add the value at the beginning	arr=[12,20,100,1000,8]
		arr.unshift(23)
		[23,12,20,100,1000,8]

arr.shift()	to delete the value from the beginning	arr=[12,20,100,1000,8] var v=arr.shift() [20,100,1000,8]
arr.splice(position, number)	delete the number of elements starting from the given position	arr=[1,2,3,4,5,6] arr.splice(3,2) //4 and 5 will be deleted [1,2,3,6]

arr.splice(position, number,list of values)	delete the number of elements starting from the given position, and replace it with the list of values	arr=[1,2,3,4,5,6]
		
		arr.splice(3,2,100,200,300,400,500)
		//4 and 5 will be replaced by the list of values
		[1,2,3, 100,200,300,400,500,6]

arr.splice(position, 0,list of values)	to add the values at the given position	arr=[1,2,3,4,5,6]

		arr.splice(3,0,100,200,300,400,500) //all the values will be added at the position [1,2,3, 100,200,300,400,500,4,5,6]

arr.indexOf(value)	find the position of the first occurrence of the given value
	This function is useful when the value is known
		arr=[12,13,12,15,13]
		arr.indexOf(13)
		1

arr.findIndex(predicate function)	predicate function -→ accepts one parameter and returns true / false
	
	findIndex function will find the position of the value for which predicate function returns true, and returns -1 if none of the number satisfies the condition
		arr=[12,13,12,15,13,20]
		
		//to find index of first value which is divisible by 5
		
		//findindex will give the index of 15,
		
		arr.findIndex((val,index,arr)=>val%5==0)
		//use return keyword inside {}
		arr.findIndex((val,index,arr)=>{return val%5==0})
		
		arr.findIndex(val=>val%5==0)

arr.find (predicate function)	predicate function -→ accepts one parameter and returns true / false
	
	find function will find the value in the array for which predicate function returns true, and returns undefined, if none of the number matches the condition
		arr=[12,13,12,15,13,20]
		
		//to find the first value which is divisible by 5
		
		//find will give the value 15,
		
		arr.find ((val,index,arr)=>val%5==0)
		//use return keyword inside {}
		arr.find ((val,index,arr)=>{return val%5==0})
		
		arr.find (val=>val%5==0)

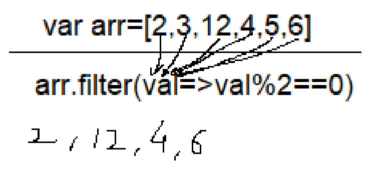
arr.filter(predicate function)	predicate function -→ accepts one parameter and returns true / false
	
	filter function will find the all the values for which predicate function returns true
		arr=[12,13,12,15,13,20]
		
		//to find all values which is divisible by 5
		
		//filter will give the array of all values which are divisible by 5
		
		arr.filter((val,index,arr)=>val%5==0)
		//use return keyword inside {}
		arr.filter((val,index,arr)=>{return val%5==0})

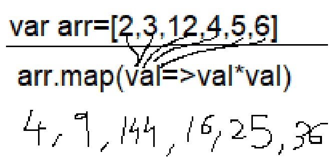
		arr.filter(val=>val%5==0)

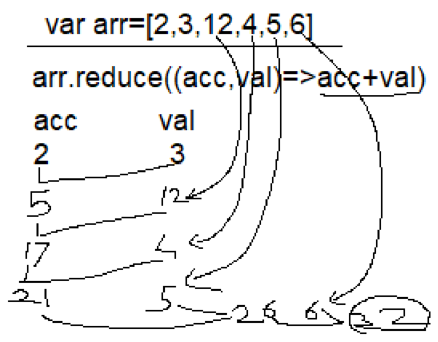
arr.map(coverter function)	map function will apply the given expression on every value in the array and return a new value	arr=[12,13,12,15,13,20]
		
		//to find squares of all the numbers
		
		arr.map((val,index,arr)=>val*val)
		[144,169,144,225,169,400]
		//use return keyword inside {}
		arr.map((val,index,arr)=>{return val*val})
		
		arr.map(val=>val*val)

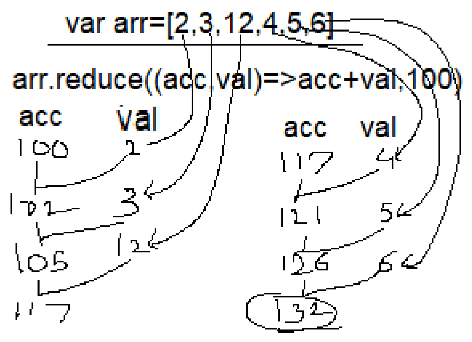
arr.reduce((acc,val)=>acc+val)	The reduce function will reduce multiple values into single value	arr=[12,13,12,15,13,20]
		
		//to find sum of all the numbers
		
		arr.reduce((acc,val)=>acc+num)
		85

arr.sort()	It will sort the data in the array by using ascii value	arr=[12,20,100,1000,8]
		
		//it will perform ascii sorting
		arr.sort()
		100,1000,12,20,8
		
		function compare(a,b){
		 /* if(a<b)
		    return -1;
		  else if (a==b){
		    return 0;
		  else
		    return 1;*/
		  return a-b;
		}
		}
		arr=[12,20,100,1000,8]
		
		//perform numeric sort
		arr.sort(compare)
		arr.sort((a,b)=>a-b)
		8,12,20,100,1000









JSON (Javascript Object notation)

ob={id:12,name:"Rujuta",skills:["Java","c++"]}

JSON is a string which looks like a javascript object

jsonob='{"id":12,"name":"Rujuta","skills" :["Java","c++"]}'

JSON.stringify(javascript object)	It will convert javascript object into JSON object
JSON.parse(json object)	It will convert JSON object into javascript object

If there is any asynchronous function, and to that function, if you pass other function as a parameter, then the function we pass is called as callback function

var id=setTimeout(function,duration)	It will call the function only once after given duration is over
var id=setInterval(function,duration)	It will keep calling the function after given interval
clearInterval(id)	It will stop the effect of setInterval
clearTimeout(id)	It will stop the delay of clearTimeout

Date class has following constructors

new Date() // current date

new Date(2024,11,11) //new Date(year, month, date

new Date(milliseconds) // milliseconds will be converted into date, starting from 1 Jan 1970;

getMonth()	It will return month between 0 to 11 jan -0 feb-1
getFullYear()	returns a year
getDate()	to get only date
getTime()	to convert date into milliseconds