Full Stack Developer

Basics of JavaScript

By,

Kalilur Rahman A R

SNOW Developer

```
/*
-------- CODEKATA PROBLEM SOLVING DAY 1 t0 4 ----------
10 - dec- 2022
-----------------
Basics of Zen portal in JS : JS codekata Day 1
variable declaration :
Types  : var, let, const

var fruit = 'Apple'
let guviMentor = 'Kalil'
const studentName = 'Khalidh'

//reassiging of variable but dont use var keyword :
fruit = 'orange'
guviMentor = 'kasheef'
studentName = 'Mohammed Mafaaz' // You cannot change the variable
constant it is immutable

console.log('Fruit : ', fruit)
console.log('Guvi Mentor : ',guviMentor)
console.log('Student name : ',studentName)

o/p :
Fruit :  orange
Guvi Mentor :  kasheef
Student name :  Khalidh
-----------------------------------
Data types :
1. String --> character " "
2. number --> 546541.64564
3. Boolean --> true or false
4. array --> ['string', number, boolen] - It contains all kind of data
types in single container
```

```
var fruit = 'Apple';
var bool = true;
var number = 4565465;

console.log(fruit, bool, number)
o/p: Apple true 4565465
-----------------------------------
Conditions : (If, else, If else)
Simple If condition :
Eg..,
var money = 5
if (money == 5){
    console.log('Chocii is affordable');
} else {
    console.log('No chocolates');
}
o/p : Chocii is affordable

Else if and Nested if condition :

Eg..,
var money = 10
if (money >= 5){ // greater than and greater than or equal to
    console.log('Big Chocii is affordable');
}if (money >= 2){
    console.log('Small chocolates');
} else {
    console.log('No chocaltes');
}
o/p :
Big Chocii is affordable
Small chocolates
```

```
--------------------------------

Strict mode :

console.log('Strict mode, Will check with data type :',5 === '5')

o/p :

Strict mode, Will check with data type : false

--------------------------------

Loop : For loop - as of now

Eg..,


// single opertions to run multiple times - you can use var and let but
not const

for(let i = 0; i <= 5; i++){

    console.log('Iam called : ', i)

}

o/p:

Iam called :  0

Iam called :  1

Iam called :  2

Iam called :  3

Iam called :  4

Iam called :  5

-----------------------------

Array : It is set of many datatypes which is stored in single container


let array = ['kalil', 'kasheef', 'kareem', 'khalidh'];

console.log('Array is :', array)

o/p: Array is : (4) ['kalil', 'kasheef', 'kareem', 'khalidh']


array.push('Mafaaz'); // will add value at the last

console.log('Array is :', array)

o/p : Array is : (5) ['kalil', 'kasheef', 'kareem', 'khalidh',
'Mafaaz']
```

```
array.pop('Mafaaz'); // will remove only the last value in array

console.log('Array is :', array)

o/p : Array is : (4) ['kalil', 'kasheef', 'kareem', 'khalidh']


array.unshift('Zaynab'); // adding in first index

console.log('Array is :', array)

o/p : Array is : (5) ['Zaynab', 'kalil', 'kasheef', 'kareem',
'khalidh']


array.shift('Zaynab'); // removing from first index

console.log('Array is :', array)

o/p : Array is : (4) ['kalil', 'kasheef', 'kareem', 'khalidh']


// In Array index will starts with zero - 0


let array = ['kalil', 'kasheef', 'kareem', 'khalidh'];

console.log('Length of Array :', array.length); o/p : Length of Array :
4

console.log(array[0]); o/p : kalil


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

    console.log( i +' '+array[i]);

    // Here i is the count of number and array[i] print array

}

o/p :

0 kalil

1 kasheef

2 kareem

3 khalidh


Eg.., --> 2 checking value present in the array if not needs to add

let array = ['kalil', 'kasheef', 'kareem', 'khalidh'];

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

```
    if (array[i] == 'Zaynab'){
        console.log(array[i]);
        break;
    } else {
        array.push('Zaynab');
        break;
    }

}
console.log('The New array is :', array);
```

o/p :

```
The New array is : (5) ['kalil', 'kasheef', 'kareem', 'khalidh',
'Zaynab']
```

---

11 - dec - 2022 :

****

Functions : Its a very important thing in Javascript

Eg.., calling a function need to use function() keyword

```
function kalil(){
    console.log('This is kalil');
    console.log('This is B43 batch');
    console.log('This is FSD');
}
kalil();
```

o/p :

```
This is kalil
This is B43 batch
This is FSD
```

Eg..,2

```
function addNumbers(){
    // set of operations
```

```
    var num1 = 9;
    var num2 = 98;
    var total = num1 + num2;
    console.log('The total is', total)
}
addNumbers();
o/p :
The total is 107
```

Eg.3.., --> Functions with arguements:

```
function addNumbers(num1, num2){
    // set of operations
    var sum = num1 + num2;
    console.log('The sum is', sum)
}
addNumbers(45,58);// calling a function then only it will execute the
set of operations
o/p:
The sum is 103

function subNumbers(num1, num2){
    // set of operations
    var sum = num1 - num2;
    console.log('The sub is', sum)
}
subNumbers(45,58);// calling a function then only it will execute the
set of operations
o/p:
The sub is -13
```

Problem Approach : How to do it?

Eg.,

A car has milege per hour plus speed per hour and fuel capacity and a car how long it can travel

Step 1 : Layman Approach

we know milege is 30 km/ltr, speed is 75 km/hr and fuel capacity is 5 ltr

total km is m * f = 30 * 5 = 150

fuel emptied is 150/75 = 2 hrs this is called Lyaman Approach

step 2 : Think how to do with the code

Eg..,

```
function remainingHoursOfTravel(milege, speed, fuel){
    var totalKilometers = milege * fuel;
    var totalHours = totalKilometers / speed;
    console.log('The total hours is', totalHours, 'Hrs')
}
remainingHoursOfTravel(30, 87, 6)
```

o/p :

The total hours is 2.0689655172413794 Hrs

// Split and join() - keywords

split - means it will split based on the value inside () and put it on array []

example : if kalil is there, if split('l') => ['ka', 'i', '']

Eg..,

```
const userInput = "k a l i l u r";
console.log('The user input : ',userInput);
```

o/p:

The user input : k a l i l u r

```
const output = userInput.split(' ')
```

```
console.log('The output is :', output);
o/p:
The output is : (7) ['k', 'a', 'l', 'i', 'l', 'u', 'r']

const joinOutput = output.join(',');
console.log('The join output is :', joinOutput)
o/p:
The join output is : k,a,l,i,l,u,r
-------------------------------------------------
// strings keyword method
const userInput2 = 'Rahman'
let concatValue = userInput.concat(' '+userInput2); //concatinations
console.log('The concatenate is :', concatValue)
o/p :
The concatenate is : k a l i l u r Rahman

//replace : will replace the word from which to what
let para = 'Hey there you failed the test'
let replaceValue = para.replace('failed', 'passed');
console.log('The replaced word :', replaceValue)
o/p:
The replaced word : Hey there you passed the test

//parseInt : will convert the string into number note: only number
//Method 1
let num = '5';
let num2 = 5;
let total = parseInt(num) + num2;
console.log('The total value is :', total)
o/p:
The total value is : 10
//Method 2:
```

```
console.log(+'6'+ 8); - o/p : 14

console.log(8 + (+ '9')); - o/p : 17

-------------------------------

sort() - Asc to Desc or Desc to Asc :

let samArr = [8, 4, 55 ,62 ,4 ,2 ,4]

let sortArr = samArr.sort();

console.log('The sorted array - default (Ascending order):', sortArr)

o/p :

The sorted array - default (Ascending order): (7) [2, 4, 4, 4, 55, 62,
8]

let reversed = sortArr.reverse()

console.log('The sorted array - (Descending order):', reversed)

o/p:

The sorted array - (Descending order): (7) [8, 62, 55, 4, 4, 4, 2]


//sorting strings and alphabet

let samAlphabet = ['kalil', 'zaynab', 'hubby', 'wife']

let sortAplphabet = samAlphabet.sort()

console.log('The string sorting - (Ascending) :', sortAplphabet)

o/p:

The string sorting - (Ascending) : (4) ['hubby', 'kalil', 'wife',
'zaynab']

let revAlpha = sortAplphabet.reverse()

console.log('The string sorting - (Descending) :', revAlpha)

o/p:

The string sorting - (Descending) : (4) ['zaynab', 'wife', 'kalil',
'hubby']


//chaining or Dot walking

let chsamArr = [8, 4, 55 ,62 ,4 ,2 ,4]

let chsortArr = samArr.sort().reverse();

console.log('The sorted array - (Chaining : Descending order):',
chsortArr)

o/p:
```

```
The sorted array - (Chaining : Descending order): (7) [8, 62, 55, 4, 4,
4, 2]
---------------------------------
// Sample code :


var nameInput = 'kal ilur';
var splitValue = nameInput.split(' ').join('');
for( let i = 0; i < splitValue.length; i++ ){
    console.log(i, splitValue[i])
}
o/p:
0 'k'
1 'a'
2 'l'
3 'i'
4 'l'
5 'u'
6 'r'
-----------------------
// Sample code :
const input = "30 5 75";
const myArr = input.split(' ')
const output = parseInt(myArr[0] * myArr[1] / myArr[2])
console.log('The output is :',output)
o/p:
The output is : 2


------------ or ------------
const input = "30 5 42";
const myArr = input.split(' ')
let milage = myArr[0]
let fuel = myArr[1]
let speed = myArr[2]
```

```
let km = milage * fuel;
let totalHours = km / speed;
console.log('The hours travel :', totalHours)
console.log('The hours travel :', totalHours.toFixed(3)) ***
// toFixed : will reduce the decimal points based on the value
o/p :
The hours travel : 3.5714285714285716
The hours travel : 3.571
-----------------------------------------
****** 17-Dec-2022 ********

-----------------------------------
Note : parseInt() and unary operator are same
console.log('Without decimal :',parseInt(52.53))
o/p : Without decimal : 52
console.log('With decimal :',parseFloat(52.53))
o/p : With decimal : 52.53

// Instead of above code can use unary operator
console.log('With decimal :', +(52.53))
o/p : With decimal : 52.53

trim() : --> It will remove the whitspaces in the front and back side
of the string
-----------------------------------
// Map function()
let inp = '1 23 3 4 5'

let a = inp.split(' ').map(Number);
console.log(a)
o/p : [1, 23, 3, 4, 5]
```

```
// The Best method to sort() or to use sort()
//This case will work for more than 10
const arr = [1,30,4,21,10000];

let b = arr.sort();
console.log('This wont work some cases :',b)
o/p : This wont work some cases : (5) [1, 10000, 21, 30, 4]
//If it is more then 10 means
arr.sort((a,b) =>{
    if(a > b){
        return 1;
    } else if (a < b) {
        return -1;
    } else {
        return 0;
    }
})
console.log('The corrected sorting :', arr);
o/p : The corrected sorting : (5) [1, 4, 21, 30, 10000]
--------------------
Codekata : Array 14:
Ramesh is a student and wants to find out if there is any other student
in his class who has got the same marks as his, in maths.
Help him to find out.

Input Description:
First line contains the number of students in the class
followed by Ramesh’s mark. Second line contains
the marks of all students in the class.

Output Description:
```

Index of student who got mark same as Ramesh’s mark. If no such mark exists, return -1.

Sample Input :

2 10

1 2

Sample Output :

-1

code:

```
let arr = userInput[0].split(' ');
let n = +arr[0];
let mark = +arr[1];

//map(Number) - will convert as a number from array
let res = userInput[1].split(' ').map(Number);
let flag = false;
let answer = 0;

for (let i = 0; i < n; i++){
    if (res[i] === mark){
        answer = i;
        flag = true;
        break;
    }
}

if (flag){
    console.log(answer);
} else {
    console.log('-1');
}
```

---

codekata : array 4

You are given with an array. For each element present in the array

your task is to print the next smallest than that number.

If it is not smallest print -1

Input Description:

You are given a number ‘n’ representing size of array.

And n space separated numbers.

Output Description:

Print the next smallest number present in array and -1 if no smallest is present

Sample Input :

7

10 7 9 3 2 1 15

Sample Output :

7 3 3 2 1 -1 -1

code :

```
let size = 7
let arr = [10, 7, 9, 3, 2, 1, 15];
let res = [];

for(let i = 0; i < size; i++){
  let flag = false;
    for (let j = i + 1; j < size; j++){
        if(arr[j] < arr[i]){
            res.push(arr[j]);
            flag = true;
            break;
        }
```

```
    }
    if(flag === false){
        res.push(-1);
    }
}

console.log(res.join(' '));
```

---

codekata : companies(2)

Given a string and a number K, change every kth character to uppercase from beginning in string.

Sample Testcase :

INPUT

string 2

OUTPUT

sTrInG

```
code :
let str = 'string';
let number = 2;
let newStr = str.split('');
console.log(newStr);
o/p: ['s', 't', 'r', 'i', 'n', 'g']

if(number === 0){
    console.log(str);
    o/p : ----------
} else {
    for(let i = number - 1; i<newStr.length; i+=number){

        // i+=number : i = i + number, if number is 2 then 2 - 1 is 1
        // so it will take 1 as 1 + 2(number) = 3, see the below pic
```

```
        console.log(newStr[i]) o/p : t, i, g
        newStr[i] = newStr[i].toUpperCase();
        console.log(newStr[i]) o/p : T, I, G
    }
}

console.log(newStr.join(""))
o/p :
sTrInG
.........................
Code optimization for above code:
let str = 'string';
let number = 2;
let newStr = str.split('');
console.log(newStr);

if (number != 0){
    for(let i = number - 1; i<newStr.length; i+=number){
        console.log(newStr[i])
        newStr[i] = newStr[i].toUpperCase();
        console.log(newStr[i])
        }
}
console.log(newStr.join(""))
-------------------------------------------------------------
Checking for palindrome :

let str = 'Oyo';
let checkValue = str.toLowerCase();
console.log(checkValue)
o/p : oyo
```

```
let finalStr = checkValue.split("");
let reverseValue = finalStr.reverse();
console.log(reverseValue)
o/p : ['o', 'y', 'o']
let joinValue = reverseValue.join("");
console.log(joinValue)
o/p : oyo

if (checkValue === joinValue){
    console.log("Yes");
} else {
    console.log("No")
}
```

Exponential or finding the power of

Math.power is professional way in Js script

```
let a = 2
let b = 12
let powerOfBase = Math.pow(a, b);
console.log(powerOfBase)

o/p : 4096
```

---

word file is created

Need to create document :

****

Company level question : codekata problem 11 in companies

Given a number N print a right angled traingle structure with the starting

level as single 1 and every immediate proceeding level with 2 more

additional ones than the previous level .Repeat the pattern for N levels.

Input Size : N <= 1000

Sample Testcase :

INPUT

3

OUTPUT

1

1 1 1

1 1 1 1 1

code :

```
let inputs = 3;
let output = "";
for(let i = 0; i<inputs; i++){
    for(let j = 1; j <= i; j++){
        output += "1" + " " +  "1" + " "
    }
    output += "1"
    output += "\n"
}
console.log(output)
```

Note :

In the above program "for" inside the "for" loop it is called Nested for

loops.

steps :

1. if i = 0 then checks with j if it is not same then break the loop

2. if i = 1 checks with j in this case it is same then will come inside the loop and print the o/p : 1 1 and print next o/p and concatenate it then o/p : 1 1 1 and it goes towards new line

Note : In Nested For loop first for loop iteration is 2nd and the second for loop will run twice

---

basics (4) :

Given 3 numbers N , L and R. Print 'yes' if N is between L and R else print 'no'.

Sample Testcase :

INPUT

3

2 6

OUTPUT

yes

code :

```
let N = userInput[0];
let diff = userInput[1].split(" ");

let L = diff[0];
let R = diff[1];
//by function:
function checkDiff(N, L, R){
    if(N > L && N < R){
    console.log("yes");
    } else {
    console.log("no")
    }
}

checkDiff(N, L, R); --> function need to be called then only it will work
```

---

basics(19) :

Given 2 numbers N and K followed by elements of N .Print 'yes' if K exists

else print 'no'.

Sample Testcase :

INPUT

4 2

1 2 3 3

OUTPUT

yes

code :

```
let N = userInput[0].split(" ");
let K = userInput[1].split(" ");
let check = N[1];

function isThere(num, arr){
"why this variable (result)?
 In For loop, while looping all the arrar element, but the result it should be one value in that case inside the function or before the loop starts decalre the variable as "let result = 0"/"let result = true/false" and after the loop check with IF statement and console it"

    var result = 0;
    for (let i = 0; i < arr.length; i++){
        if(arr[i] == num){
            result = 1;
            break;
        }
    }
    if(result == 1){
        console.log("yes");
    } else {
        console.log("no")
    }
}
isThere(check, K);
```

Note : In this above case look at function parameters and calling function

parameters both the name is different so while declaring the function

parameters which is used the same keyword or var name must be used inside

function, but while calling function the variable which is declared

GLOBALLY must used in parameters

Basics(1) :

Given 2 numbers N,M. Print 'yes' if their product is a perfect square else

print 'no'.

Sample Testcase :

INPUT

5 5

OUTPUT

yes

code:

```
let input = userInput[0].split(" ");
let a = parseInt(input[0]);
let b = parseInt(input[1]);

function isSqrt(num1, num2){
    let result = a * b;
    let check = Math.sqrt(result);
    if(check % 1 === 0){
        console.log("yes");
    }else{
        console.log("no");
    }
}
```

```
isSqrt(a, b);
```

---

basics(5) :

checking for composite number:

Given a number N, print 'yes' if it is composite else print 'no'.

Sample Testcase :

INPUT

123

OUTPUT

yes

code :

```
let input = userInput[0];
let arr = [];
for (let i = 0; i < input; i++){
    if(input % i === 0){
        arr.push(i);
    }
}

let c = arr.length;
if (c > 1){
    console.log("yes");
} else {
    console.log("no")
}
```

Hint : Let, const is block scope but Var is Global scope

Eg.., function(){

    let only called here

    const only call here

    It cannot called by out of function

}

.....

But Var keyword even it is declared inside the function it could be called

outside function too

....

Task :

Validate E-mail:

Given a string S check if is a valid email id based on the following condition:

1. @ should be present;

2. @ & . should not be repeated;

3. there should be atleast 4 characters between @ and .;

4. there should be atleast 3 characters before @;

5. the end of mail id should be .com; // slice() method

Input size: |s| < 100000

sample testcase"

INPUT

test@gmail.com

OUTPUT

Yes

code :

Hint : If a particular word needs to present at once or twice then use

"Counter"

```
let id = "kalil@gmail.com"
function isEmail(str){

    let countOfAt = 0;
    let countOfDot = 0;
    let indexOfAt;
    let indexOfDot;
```

```
    for(let i = 0; i < id.length; i++){
        if(str[i] === "@"){
            countOfAt ++;
            indexOfAt = i;
        } else if (str[i] === "."){
            countOfDot++;
            indexOfDot = i;
        }
    }
    //console.log(countOfAt)
    //console.log(countOfDot)
    // printing variables outside the loop it will give you one answer
it
    //wont print it again and again
    if(
        countOfAt === 1 &&
        countOfDot === 1 &&
        (indexOfDot - indexOfAt) >= 5 &&
        indexOfAt >= 3 &&
        str.slice(-4) === ".com"
        ){
            console.log("yes");
    } else {
        console.log("no");
    }
}
isEmail(id);
```

Note : In 3rd condition

k a l @ g m a i l . c o m

0 1 2 3 4 5 6 7 8 9 10 11 12

indexOfDot - indexOfAt = 9 - 3 => 6

but between @ and . it has only 5 indexes that's why its increased to 5

Mathematics(2):

A person saves his monthly saving according to given schema. He saves same

amount of money which is equal to the money saved in immediate previous two

months. Assume, initially he saved 1000 rupees and in first month he saved

another 1000. Your task is to tell how much he had totally saved at the end

of ‘n’ months

Input Description:

You will be given a number ‘n’->No. of months

Output Description:

Print the total savings at the end of ‘n’ months

Sample Input :

1

Sample Output :

2000

code:

```
let month = userInput[0];
let initialSavings = 1000;
let latestSavings = 0;
let overAllSavings = 0;
for(let i = 1; i <= month; i++){
    latestSavings = 1000 * i;
    overAllSavings = latestSavings + initialSavings;
    initialSavings = overAllSavings
```

```
}
console.log(overAllSavings)
```

Note :

"i" is :

1) initialSaving = 2000

2) Already initialSaving = 2000, + (2000) => 4000

3) initialSaving = 4000 so latestSaving = 1000 * 3 => 3000 so total 7000

Maths(3):

Simi is learning about palindromic numbers. Her teacher gave him the task to

count all palindromic numbers present in that range.Simi has told you about

this and want your help. You design an algorithm in order to help simi.

Input Description:

You will be given a number ‘n’

Output Description:

Print the count of all palindromic numbers till ‘n’(inclusive)

Sample Input :

5

Sample Output :

5

code :

```
function noOfPalindrome(num){
        let counter = 0;
        for(let i = 1; i <= num; i++){
        let convertToStr = i.toString();
```

```
        let reverseNumber = convertToStr.split("").reverse().join("");
        if(convertToStr === reverseNumber){
            counter++
        }
    }
        console.log(counter);
}
noOfPalindrome(userInput[0])
```

Main boot class Day 1 : (21 - 01 - 2023)

.....

Hoisting :

You can access your webpage or web based app you will be provided with link

if it is accessed in different region that is hoisting

Note :

Need to create a account on Heroku/Netlify*** its important to hoist the application

First web browser :

NCSA Mosaic was developed at the National center for supercomputing application

at the university of illinois in -> Urbana - champaign.

Next level web browser :

Netscape Navigator version 3.0 by Anderson

Java script is created by Brandon Eich with in 10 days(Microsoft - Internet explorer)

Third person :

Google Chrome(They showed to create multiple browser)

From client side :

3 types

HTML -> HTMP parser -> DOM(Document object model) Tree

JS - Js engine

CSS -> CSS Parser -> CSSOM Tree

Design of JS :

2 simple ideas :

1. key : value - everything

2. Asynchronous - It wont block your work

Browser JS Sandboxed :

Client side like UI, Inspect, console, etc..,

Eg. <script> filname.js </script>

Node JS Stripped :

server side like API, Storing data in server

Eg. Node filename.js, node.exe

Common Internet Protocols :

TCP/IP - used in Email, Web browsing

HTTP/HTTPS

FTP

All protocols has 4 layers - Application layer, Transport layer, Internet layer,

Network Access Layer

HTTP Methods :

SAFE Methods has 2 ways

1. GET - HTTP/1.1 must implement this method

2. HEAD - inspect resources header

MESSAGE with BODY(send data to server)

1. PUT - Deposit data on server - inverse of GET
2. POST - send input data for processing
3. PATCH - partially modifies a resources
4. TRACE - ECHO back recieved message
5. OPTIONS - server capabilities
6. DELETE - Delete a resource - Not Guaranteed

HTTP Return codes : inspect -> network
100 - Hold on
200 - Here you go
300 - Go away
400 - you screwed up
500 - I screwed up(server problem)

----- JS Day 2 - Request and Response cycle : 22/01/2023 -----

---

Primitive and Non primitive datatypes :
Primitive - String("kalil"), Boolean(true/false),
Number(1,2,3,4), float(12.343) --> gives as value
Non Primitive - array, object --> gives as address

Array :
Array Methods
Push --> will add value in end of the array
pop --> will remove the last value from array
unshift --> will add the value in index[0], meaning first value in array
shift --> will remove the value in index[0], meaning first value in array

Note : for push and pop will do multiple times if it is call more times

```
const arr = ['kalil', 'kasheef', 'kareem'];
console.log(arr)
o/p : ['kalil', 'kasheef', 'kareem']

//Accessing an array
console.log(arr[1]);
o/p : kasheef

arr.push("khalidh")
console.log(arr)
o/p: ['kalil', 'kasheef', 'kareem', 'khalidh']

arr.pop();
console.log(arr)
o/p : ['kalil', 'kasheef', 'kareem']

arr.unshift("khalidh");
console.log(arr);
o/p:['khalidh', 'kalil', 'kasheef', 'kareem']

arr.shift("khalidh");
console.log(arr);
o/p:['kalil', 'kasheef', 'kareem']
-------------------------
Objects : (key, value)

const studentObj = {
    name : "kalil",
    age : 24,
    isStudent : false
```

```
}

console.log("student object", studentObj);
o/p:
student object : {name: 'kalil', age: 24, isStudent: false}

Accessing or iterating object :
Note : With the help of Dot walking, can access the object

console.log("student object - Name:", studentObj.name);
o/p: student object - Name: kalil

console.log("student object - Age :", studentObj.age);
o/p: student object - Age : 24

console.log("student object - isStudent:", studentObj.isStudent);
o/p: student object - isStudent: false

changing the value in object :

studentObj.name = "Kasheef Basha";
console.log("student object :", studentObj);
o/p: student object : {name: 'Kasheef Basha', age: 24, isStudent:
false}

studentObj.age = 21;
console.log("student object :", studentObj);
o/p: student object : {name: 'Kasheef Basha', age: 21, isStudent:
false}

studentObj.isStudent = "True";
console.log("student object :", studentObj);
o/p: student object : {name: 'Kasheef Basha', age: 21, isStudent:
'True'}
```

```
-------------------------------------
Nested Array and Objects :

const arrOfObj = [
    {
        name : "kalil",
        age : 24,
        isStudent : false,
        stack : ["html", "css", "react", "php", "Javascript"]
    },
    {
        name : "kasheef",
        age : 21,
        isStudent : true
    }

];
console.log("Array of Objects :",arrOfObj)
o/p :
o/p:
Array of Objects : (2) [{name: 'kalil', age: 24, isStudent: false,
stack: Array(5)},
                        {name: 'kasheef', age: 21, isStudent: true}]

Accessing Array of Objects :
console.log(arrOfObj[0]);
o/p : {name: 'kalil', age: 24, isStudent: false, stack: Array(5)}

console.log(arrOfObj[1]);
o/p : {name: 'kasheef', age: 21, isStudent: true}

console.log(arrOfObj[0].name);
o/p : kalil
```

```
console.log(arrOfObj[0].stack);
o/p : (5) ['html', 'css', 'react', 'php', 'Javascript']

console.log(arrOfObj[0].stack[4]);
o/p : Javascript
```

Copy by value and copy by reference :

copy by value.....for primitive types :

```
let a = 10;
let b = a;
console.log("b :",b); o/p : b : 10
console.log("a :",a); o/p : a : 10
b = 20;
console.log("b :",b); o/p : b : 20
console.log("a :",a); o/p : a : 10
```

Note : For "a", it wont take it as "20" because it has seperate memory allocation

For primitive data types each has seperate memory allocation

copy by referrence......for non primitive types:

```
let obja = {value : 15};
let objb = obja;
console.log("objb :",objb); o/p : objb : {value: 15}
console.log("obja :",obja); o/p : obja : {value: 15}
objb.value = 100;
console.log("objb :",objb); o/p : objb : {value: 100}
console.log("obja :",obja); o/p : obja : {value: 100}
```

Note :

For Non primitive types, it wont create new memory allocation, besides it will create

as memory address Eg.,"1111" it will update ony in that address it wont create a new

address when declaring as b = a in object and arrays.

Window :

console.group(window)

Note : It includes all functionality of browser window

Document :

console.log(document)

It will gives the html content of the window

syntax for getting url of the current window :

```
let url = window.document.URL;
console.log(url);
//o/p: file:///D:/FSD_Guvi/FSD/fsd.html
```

XMLHttp request :

Calling data's from the third party application to the main application

syntax : This is old method

```
let xhr = new XMLHttpRequest();
console.log(xhr);

xhr.open("GET", "https://restcountries.com/v3.1/all");
xhr.send();
xhr.onload = function(){
    const data = JSON.parse(xhr.response)
```

```
    console.log(data);
    console.log(data[0]);
    console.log(data[0].continents);
    console.log(xhr.status);
}
-------------------------------------
***** Javascript Day 3 : JS Array and objects *****
Topics :
Hoisting & scope
function & return keyword
types of function
.....................................

Global Execution Time :
For eg., var a = 10;
It has 2 allocation

Memory           Executuion
---------------------------
 a         .     a = 10
 b         .     b = 10
           .


First it will alocate all the variables, then it will start executuing
the variables


Call stack :


LIFO - Last in first out
Once all the execution is done with GEC/GET then it will come inside
the stack and it will
run the javascript
```

Note : This is only for variable declaration

For function declaring :

```
Memory           Executuion
--------------------------
 a          .     a = 10
 b          .     b = 10
            .
            .
function    . It creates its own GET
            .
            .
            .
```

Example :

```
c();
var a = 10;

function c(){
    var b = 56;
    console.log(b)
}

console.log(a);
```

o/p :

56

10

Note : After declaring the variable, then you can only console it this is only

for variable, but incase of function you can call it before and after too

it will execute it because function will always create its own GEC

Hoisting :

Example :

console.log(a) --> undefined, bcoz it is not yest declared, this is called Hoisting

var a = 10;

console.log(a)

You can show the call stack in browser inspect --> sources --> breakpoint -->

reload window.

Functions in Javascript :

function statement or declaration :

```
name1();
function name1(){
    console.log("Name is")
}
```

o/p : Name is

Function expression :

Declaring a function in the variable is called function expression

```
var test = function(){
    console.log("Hi Kalil")
}
```

test(); --> this variable is holding all the function characterisitics

o/p : Hi Kalil

.....

Parameters and Arguments in functions:

Example:

```
var test = function(a){  --> parameters, it only call inside the local scope/inside(declare)
    console.log("Hi Kalil");
    console.log(a);
}

test();
```

test(a); --> Arguments, can calls outside of the function, while invoking the function

.....

forsdt class function:

Here displayFunction is a forsdt class function because it takes function as a parameters

Example :

```
function addition (num){
    return num + num;
}

function displayFunction(fn){ --> forsdt class function
    return fn(10)
}
console.log(displayFunction(addition)) --> function as a arguements
```

o/p : 20

.....

Anonomous function :

Function declared without name

.....

Immediate invoking function expression:

All the funcion should be with in () see the below example:

```
(function multiply (num){
    console.log(num * num);
})(5)
```

o/p : 25

.....

Arrow function : It introduced in ES6 (EcmaScript)

syntax :

```
var mulByTwoArrow = (num) => {
    return num * 6;
}
console.log("The arrow function :",mulByTwoArrow(45));
```

--------or----------

```
var arrow = (num) => num * 2;
console.log("The multiplication is :",arrow(45))
```

o/p : The arrow function : 270

For loop iteratives :

Instead of writing again and again console.log see the below function

const log = (param) => console.log(param); --> Good example

```
const arr = ["kalil", "kasheef", "kareem", "zaynab", "zehra"];

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

```
    log(arr[i] + " ---> " +i);
}
o/p :
kalil ---> 0
kasheef ---> 1
kareem ---> 2
zaynab ---> 3
zehra ---> 4
.............................

const studentObj = {
    name : "kalilur rahman",
    age : 25,
    married : "yes",
    isStudent : false
}

Mapping methodology:
log(studentObj["name"]);
log(studentObj["age"]);

o/p :
kalilur rahman
25
.............................
getting studentObj keys into array structure:
var studentKeys = Object.keys(studentObj)
log(studentKeys);
o/p :
['name', 'age', 'married', 'isStudent']
.................................
getting studentObj values into array structure :
```

```
var studentValues = Object.values(studentObj)
log(studentValues);
o/p :
['kalilur rahman', 25, 'yes', false]
.........................................
Note :
Only in Array you can use for loop so we converting the keys and values
as array


log("-------------- For each Loop --------------")
For Each Loop : (Array can iterate)
Note :  It has 3 parameters : value, index, array
By Arrow function :

arr.forEach((value, index, actArr)=>{
    console.log(value, index, actArr)
})
o/p :
kalil 0 (5) ['kalil', 'kasheef', 'kareem', 'zaynab', 'zehra']
kasheef 1 (5) ['kalil', 'kasheef', 'kareem', 'zaynab', 'zehra']
kareem 2 (5) ['kalil', 'kasheef', 'kareem', 'zaynab', 'zehra']
zaynab 3 (5) ['kalil', 'kasheef', 'kareem', 'zaynab', 'zehra']
zehra 4 (5) ['kalil', 'kasheef', 'kareem', 'zaynab', 'zehra']
.....................................
By Normal function [For object first you need to change into as Array]
studentKeys.forEach(function (values, index){
    console.log(values,"-->",index)
})
o/p :
name --> 0
age --> 1
married --> 2
isStudent --> 3
```

```
.....................................
log("-------------- For in Loop --------------")
Accessible for objects :
In Array : In For In Loop, Just only can get the index of the array

for(s in arr){
    console.log(s)
}
o/p :
0
1
2
3
4
.........................................
In Object iterations you can get the keys as well as values too :

for(obj in studentObj){

    if (studentObj[obj] == "kalilur rahman"){
        studentObj[obj] = "Zaynab Kalil";
    }

    console.log("keys :", obj,", Values :",studentObj[obj] )
}
console.log(studentObj)
o/p :
keys : name , Values : Zaynab Kalil
keys : age , Values : 25
keys : married , Values : yes
keys : isStudent , Values : false
```

```
{name: 'Zaynab Kalil', age: 25, married: 'yes', isStudent: false}
.........................................
log("-------------- For of Loop --------------")
For of -> can use array and string :

var bingo = "Lays";

for (studentArr in arr){
    console.log(studentArr, arr[studentArr])
}
o/p :
0 kalil
1 kasheef
2 kareem
3 zaynab
4 zehra
.....................................

for (str of bingo){
    console.log(str)
}
o/p :
L
a
y
s
........................

for (k of studentKeys){
    console.log(k,":",studentObj[k]) // Not a best way for object
}
o/p :
```

name : Zaynab Kalil

age : 25

married : yes

isStudent : false

.....

//JSON - Javascript Object Notation - Itis as same as javascript object

.....

*** Java script Day-5 ***

.....

var vs let vs const:

var : it will be hoisted and present in the global scope

let and const : It will be also hoisted but will present in lexical scope(script) not in global.

once it is enter to global it is called temporal deadzone

var:

```
console.log("local :", local)
var local = 15;
console.log("local :", local)
```

o/p :

local : undefined

local : 15

let :

```
console.log("local :", local)
let local = 15;
console.log("local :", local)
```

o/p :

Uncaught ReferenceError: Cannot access 'local' before initialization

const:

```
console.log("local :", local)
```

```
const local = 15;
console.log("local :", local)
o/p :
Uncaught ReferenceError: Cannot access 'local' before initialization

Example :
let local = 45;
local = 34; --> can do that
const value = 23;
value = 45; --> cannot change/reassign value for const

.....................................
Block scoping vs Global scoping :

Global scoping :
var should declare outside the function and calling function to be
outside

let naming = "kalil";
function consoelName(){
    console.log(naming);
}
consoelName()
o/p : kalil

Block scoping :
Declaring variable inside the function is called block scoping
wrong code:
function consoelName(){
    let naming = "kalil";
    //console.log(naming);
}
consoelName(naming)
```

o/p : Uncaught ReferenceError: naming is not defined

Right code :

```
function consoelName(){
    let naming = "kalil";
    console.log(naming);
}
consoelName()
```

o/p : kalil

.....

spread & rest operators :

spread : seperating values in array is called spread

```
const listArr = ["fruits", "veges", "drinks", "cosmetics"];
console.log(...listArr)
```

o/p :

fruits veges drinks cosmetics

Rest : values which are seperated it will put together into box as []

if you are getting values and push in [] then use rest operators.

```
function listOne(arg1, ...arg2){
    console.log(arg1, arg2)
}
listOne("fruit", "veg1", "veg2", "veg3", "veg4", "veg5", "veg6");
```

o/p :

fruit (6) ['veg1', 'veg2', 'veg3', 'veg4', 'veg5', 'veg6']

.....

array & object destructure :

Object destructing :

```
const obj = {
```

```
    name : "kalil",
    age : 25,
    position : "servicenow developer",
    experience : 4

}
Normal way of accessing:
console.log("name :", obj.name);
o/p : name : kalil

destructing way : reversing the object variable declaration
const {name, age, position, experience} = obj;

if (name == "kalil"){
    name = "kalilur Rahman";
}

console.log("name :", name);
console.log("age :", age);
console.log("poistion :", position);
console.log("experience :", experience);

o/p :
name : kalilur Rahman
age : 25
poistion : servicenow developer
experience : 4
.........................................
Array destructing :
const arr1 = ["kalil", "kasheef", "kareem", "khalidh"];
const[name1, name2, ...names] = arr1
console.log(name1);
```

```
console.log(name2);
console.log(names);
console.log(...names);

o/p :
kalil
kasheef
(2) ['kareem', 'khalidh']
kareem khalidh
.....................................
Object property shorthand :

const objArr = [
    {
        name : "kalil",
        pos : "Developer"
    }
]

const name = "kasheef";
const pos = "student"

const newObj = {
    name : name,
    pos : pos
}

//--------------- or -------------------------
//If the key and values are same instead of above code you can use as:
const newObj1 = {
    name,
    pos
```

```
}

objArr.push(newObj);
console.log(objArr);
o/p :
2) [{…}, {…}]
0 : {name: 'kalil', pos: 'Developer'}
1 : {name: 'kasheef', pos: 'student'}
length : 2
[[Prototype]] : Array(0)

console.log(newObj1)
o/p :
{name: 'kasheef', pos: 'student'}
.................................................
Template literals : will be defined as "` `"
console.log(`Tne Name of the Employee is "${objArr[0].name}" and
the position is : ${objArr[0].pos}`);
o/p :
The Name of the Employee is "kalil" and
the position is : Developer.
.........................................
****** JS Day-6 ***********
Classes in Java script :

Factory methods or method : creating single function and getting many
users details
Creating a multiple user at a time :

function createUser(name, age){

    return {
        name,
```

```
        age,
        getDetails : function(){
            console.log(`Hey my name is ${name} and age is ${age}`);
        }
    }
}

const user1 = createUser("kalil", 24);
user1.getDetails();
o/p : Hey my name is kalil and age is 24
const user2 = createUser("kasheef", 21);
user2.getDetails();
o/p : Hey my name is kasheef and age is 21

second example :
function createCar(color, model, brand){

    return {
        color,
        model,
        brand,
        getColor : function(){
            console.log(`The data you asked is ${color}`);
        },
        getModel : function(){
            console.log(`The data you asked is ${model}`);
        },
        getBrand : function(){
            console.log(`The data you asked is ${brand}`);
        },

    }
```

```
}

const vehicle_1 = createCar("red", "i20", "Hyundai");
const vehicle_2 = createCar("Dual tone red", "Asta", "Honda");

vehicle_1.getBrand();
vehicle_2.getBrand();
o/p :
The data you asked is Hyundai
The data you asked is Honda
-----------------------------------
Note : window is also called as object or Global window object
This keyword : it is related to its parent object, it targets only the parent object

Here this refers to global or window object so it will print as kasheef
var name = "kasheef"
console.log(this.name);
o/p : kasheef

let users = {
    name : "kalil",
    age : 24,
    getDetails(){
        console.log(this.name); // it prints only the parent object key
    }
}
users.getDetails();
o/p : kalil
.................................

let details = {
    name : "kalil",
```

```
    age : 34,
    nestedObj : {
        name : "khalidh",
        age : 01,
        getUser(){
            console.log(this.name);
        }
    }
}
```

It will print "khalidh" bcoz, parent obj is nestedObj :

```
details.nestedObj.getUser();
```

o/p :

khalidh

.....

This keyword behavior :

If its used inside normal function it will points out its parent obj.

If its used inside the Arrow function, it will points out Globally declared obj.

```
var name = "This is global window object"

let details1 = {
    name : "kalil",
    age : 34,
    nestedObj : {
        name : "khalidh",
        age : 01,
        normalFunction(){
            console.log("I'm, normal function :", this.name);
        },
```

```
        arrowFunction : () => {
            console.log("I'am, Arrow function :",this.name);
        }
    }
}

details1.nestedObj.normalFunction();
details1.nestedObj.arrowFunction();
```

o/p :

I'm, normal function : khalidh

I'am, Arrow function : This is global window object

.....

How to call this keyword inside the object by using arrow function :

If I call the arrow function inside the object, it will target only inside the object but if we call same arrow function outside the object it will print the global object in window.

Best example :

```
var name = "This is global window object"

let details1 = {
    name : "kalil",
    age : 34,
    nestedObj : {
        name : "khalidh",
        age : 01,
        normalFunction(){
            console.log("I'm, normal function :", this.name);
            var callArrowObj = () => {
                console.log("Iam calling arrow inside the obj :",
this.name)
```

```
            }
            callArrowObj();
        },
        arrowFunction : () => {
            console.log("I'am, Arrow function :",this.name);
        }
    }
}

details1.nestedObj.normalFunction();
details1.nestedObj.arrowFunction();
```

o/p :

```
I'm, normal function : khalidh
Iam calling arrow inside the obj : khalidh
I'am, Arrow function : This is global window object
```

COnstructor method :

Constructor method should be declared as Caps letter name "Car" with parameters

and that parameters must declare with this keyword and can declare with function

too. please read the below example

It also called as encapsulation, it will only call within constructor it won't

go or calls the global / window scope.

Example :

```
function Car(color, model, brand){
    this.color = color;
    this.model = model;
    this.brand = brand;
```

```
    this.getColor = function(){
        console.log(`The data you asked is ${this.color}`);
    };
    this.getModel = function(){
        console.log(`The data you asked is ${this.model}`);
    };
    this.getBrand = function(){
        console.log(`The data you asked is ${this.brand}`);
    }
}

const conVeh_1 = new Car("Red", "A500", "Benz");
console.log(conVeh_1);
conVeh_1.getBrand();
```

o/p :

```
Car {color: 'Red', model: 'A500', brand: 'Benz', getColor: ƒ, getModel: ƒ, …}
The data you asked is Benz
```

```
const conVeh_2 = new Car("Black", "9000", "BMW");
console.log(conVeh_2);
conVeh_2.getColor();
conVeh_2.getBrand();
conVeh_2.getModel();
```

o/p :

```
Car {color: 'Black', model: '9000', brand: 'BMW', getColor: ƒ, getModel: ƒ, …}
The data you asked is Black
The data you asked is BMW
The data you asked is 9000
```

Prototype :

Prototype is used for better scalablity in code it will be declared variables and

function seperately, in future if we need to declare the combined output of variables

directly we can call prototype of "Car". See the below function.

```
function Car(color, model, brand){
    this.color = color;
    this.model = model;
    this.brand = brand;
}

//prototypes :

Car.prototype.getColor = function(){
    console.log(`The data you asked is ${this.color}`);
}
Car.prototype.getModel = function(){
    console.log(`The data you asked is ${this.model}`);
}
Car.prototype.getBrand = function(){
    console.log(`The data you asked is ${this.brand}`);
}

Car.prototype.carDetails = function() {
    console.log(`The brand is ${this.brand} and
The color is ${this.color} and
The color is ${this.model}`);
}
```

```
const conVeh_1 = new Car("Red", "A500", "Benz");
console.log(conVeh_1);
o/p :
Car {color: 'Red', model: 'A500', brand: 'Benz'}

const conVeh_2 = new Car("Black", "9000", "BMW");
console.log(conVeh_2);
o/p :
Car {color: 'Black', model: '9000', brand: 'BMW'}

conVeh_2.getColor();
conVeh_2.getBrand();
conVeh_2.getModel();
conVeh_2.carDetails();

o/p :
The data you asked is Black
The data you asked is BMW
The data you asked is 9000
The brand is BMW and The color is Black and The color is 9000
-----------------------------------------------------------------
Classes : putting together var and function in single block

class Student {
    constructor(name, batch, week){
        this.name = name;
        this.batch = batch;
        this.week = week;
    }

    getDetails(){
        console.log(`Name of the student : ${this.name}.\nBatch is :
${this.batch}.\nClass Type is : ${this.week}.`);
```

```
    }
}

const student1 = new Student("Kalilur Rahman", "B43", "Weekend");
console.log(student1);
student1.getDetails();
o/p :

Student {name: 'Kalilur Rahman', batch: 'B43', week: 'Weekend'}
Name of the student : Kalilur Rahman.
Batch is : B43.
Class Type is : Weekend.
-----------------------------------------------
Inheritance :
Getting an attributes of the parent class, if from child you try to
call the value
to parent then it is undefined.

class Student { --> parent class
    constructor(name, batch, week){
        this.name = name;
        this.batch = batch;
        this.week = week;
    }
    onlyDetails(){
        console.log(`Name of the student : ${this.name}.
        Batch is : ${this.batch}.
        Class Type is : ${this.week}.
        Placed Comapny : ${this.company}`);
    }
}

//placed company : child class
```

```
class Placement extends Student{
    constructor(name, batch, week, company){
        super(name, batch,week); --> this param is from parent class with the help of Super keyword
        this.company = company;
    }
    getDetails(){
        console.log(`Name of the student : ${this.name}.
        Batch is : ${this.batch}.
        Class Type is : ${this.week}.
        Placed Comapny : ${this.company}`);
    }
}

const student1 = new Student("Kalilur Rahman", "B43", "Weekend");
student1.comapny = "Amazon"; --> can call from child by this method
console.log(student1);
student1.onlyDetails();
o/p :
Student {name: 'Kalilur Rahman', batch: 'B43', week: 'Weekend', company: 'Amazon'}
Name of the student : Kalilur Rahman.
Batch is : B43.
Class Type is : Weekend.
Placed Comapny : Amazon

const placedStudent1 = new Placement("kasheef Basha", "B43", "Week days", "Google");
console.log(placedStudent1);
placedStudent1.getDetails();
o/p :
Placement {name: 'kasheef Basha', batch: 'B43', week: 'Week days', company: 'Google'}
```

Name of the student : kasheef Basha.

Batch is : B43.

Class Type is : Week days.

Placed Comapny : Google

*** js DAY & ******

Map in Js :

Map is array method which returns a new array

map(() => {}) -> map inside the arrow function, it is same as forEach loop

Interview question :

map returns values. can assign it to other variable

forEach doesnt return anything, if you try it is undefined

Example :

```
const arr = [1,2,3,4,5,6,7,8];

const arrMap = arr.map((val, idx, accArr) => {
    console.log(`
    val : ${val}
    index : ${idx}
    ActualArr : ${accArr}`)
})
```

o/p :

```
    val : 1
    index : 0
    ActualArr : 1,2,3,4,5,6,7,8

    val : 2
    index : 1
```

```
    ActualArr : 1,2,3,4,5,6,7,8

    val : 3
    index : 2
    ActualArr : 1,2,3,4,5,6,7,8

    val : 4
    index : 3
    ActualArr : 1,2,3,4,5,6,7,8

    val : 5
    index : 4
    ActualArr : 1,2,3,4,5,6,7,8

    val : 6
    index : 5
    ActualArr : 1,2,3,4,5,6,7,8

    val : 7
    index : 6
    ActualArr : 1,2,3,4,5,6,7,8

    val : 8
    index : 7
    ActualArr : 1,2,3,4,5,6,7,8
.....................................................
const arrMap_1 = arr.map((val, idx, accArrr) => {
    return val * 5;
})
console.log(arrMap_1)
o/p :
(8) [5, 10, 15, 20, 25, 30, 35, 40]
```

```
.............................................

const arrOfObj = [
    {
        name : "kalil"
    },
    {
        name : "kasheef"
    },
    {
        name : "kareem"
    }
]

//use case of map : can write in a single line

arrOfObj.map((obj, idx, accArr) => console.log(accArr))
o/p :
(3) [{…}, {…}, {…}]
(3) [{…}, {…}, {…}]
(3) [{…}, {…}, {…}]
.........................................
Filter : it will returns an array
but we have lot of methods in filter try to use w3schools

Example :

const arr = [1,2,3,4,5,6,7,8];

const filterArr = arr.filter((val, idx, accArr) => val === 5);
console.log(filterArr);
o/p : [5] --> it will give only 5, if its present
```

```
const filterArr1 = arr.filter((val, idx, accArr) => val !== 4);
console.log(filterArr1);
o/p : --> it will remove the value if its present, then only print all
array
[1, 2, 3, 5, 6, 7, 8]

// instead of this :
const newArr = [];
for(let i = 0; i < arr.length; i++){
    if(arr[i] !== 5){
        newArr.push(arr[i])
    }
}
console.log(newArr)
o/p : [1, 2, 3, 4, 5, 6, 7, 8]


const filterArr2 = arr.filter((val, idx, accArr) => accArr[2] === 3);
console.log(filterArr2);
o/p : --> checks the arr, if its true it will return all the array
[1, 2, 3, 4, 5, 6, 7, 8]

//Contains in Filter
const check = arr.includes(8);
console.log(check);
o/p : true --> return as boolean value
.............................................
Note : remove duplicate values by using Set constructor.
.............................................

Reduce in JS :
returns a value, it will be used in
```

```
syntax : reduce(()=>{return value})

const arr = [1,2,3,4,5,6,7,8];

const add = arr.reduce((acc, val, index, accArr) => {
    return acc + val;
}, 1);

console.log("reduce returns :", add)
o/p : reduce returns : 37
1+2+3+4+5+6+7+8 = 36 + 1 = 37
.........................................

Task : Map and filter
const arr1 = ["sanjay", "arun", "nagaraj"];

//using map
const result = arr1.map((val, idx, accArr) => {
    console.log(val, idx, accArr)
})

o.p :
sanjay 0 (3) ['sanjay', 'arun', 'nagaraj']
arun 1 (3) ['sanjay', 'arun', 'nagaraj']
nagaraj 2 (3) ['sanjay', 'arun', 'nagaraj']

detele operation :

const removeSanjay = arr1.filter((val) => val !== "sanjay");
console.log(removeSanjay);
o/p :
(2) ['arun', 'nagaraj']
```

```
-----------------------------------------------
prototype of map :
Developing our own prototype or map functtion

Array.prototype.zenkMap = function(fn){

    let getArr = [];
    for(let i = 0; i < this.length; i++){
        getArr.push(fn(this[i], i, this))
    }
    return getArr;
}

const arr = [1,2,3,4,5,6,7,8];

const myPro = arr.zenkMap((val, idx, accArr) => {
    console.log(`
    Values : ${val}.
    Id : ${idx}.
    Actual Array : ${accArr}`)
})
o/p :

    Values : 1.
    Id : 0.
    Actual Array : 1,2,3,4,5,6,7,8

    Values : 2.
    Id : 1.
    Actual Array : 1,2,3,4,5,6,7,8

    Values : 3.
```

```
    Id : 2.
    Actual Array : 1,2,3,4,5,6,7,8

    Values : 4.
    Id : 3.
    Actual Array : 1,2,3,4,5,6,7,8

    Values : 5.
    Id : 4.
    Actual Array : 1,2,3,4,5,6,7,8

    Values : 6.
    Id : 5.
    Actual Array : 1,2,3,4,5,6,7,8

    Values : 7.
    Id : 6.
    Actual Array : 1,2,3,4,5,6,7,8

    Values : 8.
    Id : 7.
    Actual Array : 1,2,3,4,5,6,7,8
.....................................................
Filter own prototype :
Array.prototype.kalilFilter = function(fn){
    let arr = [];
    for(let i = 0; i < this.length; i++){
        if(fn(this[i], i, this)){
            arr.push(this[i]);
        }
    }
```

```
    return arr;
}

const newFiltrer = arr.kalilFilter((val, idx, accArr) => val > 5);
console.log(newFiltrer)
o/p :
[6, 7, 8]
.....................................
Reduce : own prototype

const arr = [1,2,3,4,5,6,7,8];

Array.prototype.zenReduce = function(fn, initValue){
    let acc = initValue;
    for(let i = 0; i < this.length; i++){
        if(acc){
            acc = fn(acc, this[i], i, this);
        } else {
            acc = this[i]
        }
    }
    return acc;
}

const add = arr.zenReduce((acc, val, index, accArr) => {
    return acc + val;
}, 10);

console.log("reduce returns :", add)
o/p :
reduce returns : 46
.,.....................................,
```

```
****** Interview questions example and output************
//1.a Print odd numbers in an array
const log = (param) => console.log(param);
const oddArr = [];
(function oddNum(num){
    for(let i = 1; i < num; i++){
        if(i % 2 != 0){
            oddArr.push(i);
        }
    }
})(45)
console.log(oddArr);
o/p :
[1,3,5,7,……]

log("--------------------------------------------------");

//1.b Convert all the strings to title caps in a string array
const capResult = [];
(function capsFirWord(sentence){
    const splitString = sentence.split(" ");
    for(arr of splitString){
        capResult.push(arr[0].toUpperCase() + arr.substr(1));
        // substr means it will delete the element from a word
    }
})("when your heart breaks your mind starts working")

log(capResult);
o/p :
```

1. **0**: "When"
2. **1**: "Your"
3. **2**: "Heart"
4. **3**: "Breaks"
5. **4**: "Your"
6. **5**: "Mind"

7. **6**: "Starts"
8. **7**: "Working"

```
log("-----------------------------------------------------");

//1.c sum of all numbers in array
const numbers = [];
var sum = 0;
(function (num){
    for(let i = 1; i <= num; i++){
        //numbers.push(i);
        sum = sum + i;
    }
})(4)
log(sum);
o/p : 10
log("-----------------------------------------------------");

//1.d return all the prime numbers in an array.
var primes = [];
(function(start, end){
    for(let i = start; i <= end; i++){
        var isPrime = true;
        for(let j =2; j < i; j++){
            if( i % j === 0){
                isPrime = false;
                break;
            }
        }
        if(isPrime){
            primes.push(i);
        }
    }
```

```
}(1, 5))
log(primes);
```

o/p :

1. Array(4)
    1. **0**: 1
    2. **1**: 2
    3. **2**: 3
    4. **3**: 5

```
log("-------------------------------------------------");

//1.e Return all the palindromes in array
let palindrome = [];
(function(words){
    for(word of words){
        let lowerCase = word.toLowerCase();
        let splitWord = lowerCase.split("");
        let reverseWord = splitWord.reverse();
        let result = reverseWord.join("");
        if(lowerCase === result){
            palindrome.push(lowerCase);
        }
    }
}(["kalil","khalidh","Madam","Dad","Mom"]))
console.log(palindrome)
```

o/p :

1. Array(3)
    1. **0**: "madam"
    2. **1**: "dad"
    3. **2**: "mom"
    4. **length**: 3

```
log("-------------------------------------------------");
/*
function getPalindromes(words) {
    let palindromes = [];
    for (let word of words) {
      let isPalindrome = true;
```

```
      for (let i = 0; i < word.length / 2; i++) {
        //console.log(word[word.length - i - 1].toUpperCase())
        //console.log(word)
        if (word[i] !== word[word.length - i - 1]) {
          isPalindrome = false;
          break;
        }
      }
      if (isPalindrome) {
        palindromes.push(word);
      }
    }
    return palindromes;
  }

  console.log(getPalindromes(["level", "hello", "world", "madam",
"mom"]));

  log("--------------------------------------------------");
*/

//1.f Return median of two sorted arrays of the same size
//median means : The median is the value in the middle of a data set,
//meaning that 50% of data points have a value smaller or equal to the
median
//and 50% of data points have a value higher or equal to the median.
//For a small data set, you first count the number of data points (n)
//and arrange the data points in increasing order.

// 1,2,3,4,5,6,7 = median is 4 , 1,2,3,4,5,6,7,8 = median is 4+5 = 9 %
2 = 4.5

function findMedianSortedArrays(arr1, arr2) {
    let mergedArray = [];
```

```
  let i = 0, j = 0;
  //let arr1Len = arr1.length;
  //let arr2Len = arr2.length;
  while (i < arr1.length && j < arr2.length) {
    if (arr1[i] < arr2[j]) {
      //console.log(arr1[i]);
      //console.log(arr2[j]);

      mergedArray.push(arr1[i]);
      i++;
    } else {
      mergedArray.push(arr2[j]);
      j++;
    }
  }
  while (i < arr1.length) {
    mergedArray.push(arr1[i]);
    i++;
  }
  while (j < arr2.length) {
    mergedArray.push(arr2[j]);
    j++;
  }

  let mid = Math.floor(mergedArray.length / 2);
  if (mergedArray.length % 2 === 0) {
    return (mergedArray[mid - 1] + mergedArray[mid]) / 2;
  } else {
    return mergedArray[mid];
  }
}
```

```
  console.log(findMedianSortedArrays([1, 3, 5], [2, 8, 6]));
o/p : 4
  log("-------------------------------------------------");

  //1.f Remove dulicates from an array
 /*
 The below code is for finding the duplicates with the help of object,
 if its same key then it will be pushed in duplicates...
  function findDuplicates(arr) {
    let duplicates = [];
    let frequency = {};
    for (let i = 0; i < arr.length; i++) {
        //console.log(frequency[arr[i]]);
      if (frequency[arr[i]]) {
        duplicates.push(arr[i]);
      } else {
        frequency[arr[i]] = true;
      }
    }
    return duplicates;
  }

  console.log(findDuplicates([1, 2, 3, 1, 2, 3, 4, 5, 6, 5]));
  */

let uniqueElements = [];
let frequency = {};
(function(arr){
    for(let i = 0; i < arr.length; i++){
        if(!frequency[arr[i]]){
            uniqueElements.push(arr[i]);
            frequency[arr[i]] = true;
```

```
        }
    }
}([1, 2, 3, 1, 2, 3, 4, 5, 6, 5]))


console.log(uniqueElements)
```

o/p :

1. Array(6)
    1. **0**: 1
    2. **1**: 2
    3. **2**: 3
    4. **3**: 4
    5. **4**: 5
    6. **5**: 6

```
console.log(frequency)
```

o/p :

1. Object
    1. **1**: true
    2. **2**: true
    3. **3**: true
    4. **4**: true
    5. **5**: true
    6. **6**: true

```
log("-------------------------------------------------");


// 1.f Rotate an array by k times


//Testing for rotating the array
// let testArr = [1,2,3,4,5,6]
// console.log(testArr.unshift(testArr.pop()))
// console.log(testArr.unshift(testArr.pop()))



(function(arr, k){
    console.log("Before Rotating Array :" + arr);
    for(let i = 1; i <= k; i++){
        arr.unshift(arr.pop());
        console.log(`After rotating ${i} times, New Array will be :
${arr}`);
```

```
    }
    return arr
}([1,2,3,4,5,6], 3))
o/p :
Before Rotating Array :1,2,3,4,5,6
After rotating 1 times, New Array will be : 6,1,2,3,4,5
After rotating 2 times, New Array will be : 5,6,1,2,3,4
After rotating 3 times, New Array will be : 4,5,6,1,2,3

log("--------------------------------------------------");
log("----------- By using Arrow function -------------");

//2.a Print odd numbers in an array
//let resultArray = [];
const primeNum = (num) => {
    let resultArray = [];
    for(let i = 0; i < num; i++){
        if(i % 2 != 0){
            resultArray.push(i)
            // console.log(resultArray)
        }
    }
    console.log(resultArray)
    return resultArray;
}
primeNum(45)
log("--------------------------------------------------");
//2. b Convert all the strings to title caps in a string array
const makeCaps = (sentence) => {
    let result = [];
    let sepSentences = sentence.split(" ");
    for(word of sepSentences){
```

```
        //console.log(word)
        let capsOfFirstWord = word[0].toUpperCase() + word.substr(1);
        // substr means it will delete the word based on the placement
of word
        result.push(capsOfFirstWord);
    }

    console.log(result);
    return result;
}
makeCaps("kalil is a good boy");
log("-------------------------------------------------");
//2.c sum of all numbers in array
let total = 0;
const sumOfArr = (arr) => {
    for(let i = 0; i < arr.length; i++){
        total = total + arr[i];
    }
}
sumOfArr([1,2,3,45,87]);
console.log(total);
log("-------------------------------------------------");

//2.d return all the prime numbers in an array
// prime number : it is only divisible by itself
const findPrime = (start, end) => {
    let result = [];
    for(let i = start; i <= end; i++){
        let isPrime = true;
        for(let j = 2; j < i; j++){
            if(i % j === 0){
                isPrime = false;
                break;
```

```
            }
        }
        if(isPrime){
            result.push(i)
        }
    }
    console.log(result);
    return result;
}

findPrime(1, 10);
log("--------------------------------------------------");

//2.e return all the palindrome in an array
const findPalindrome = (arr) => {
    let result = [];
    for (word of arr){
        let makeLowercase = word.toLowerCase();
        let splitWord = makeLowercase.split("");
        let reverseWord = splitWord.reverse();
        let joinReversedWord = reverseWord.join("");
        if (makeLowercase === joinReversedWord){
            result.push(makeLowercase);
        }
    }
    console.log(result);
    return result;
}
findPalindrome(["madam", "dad", "kalil"])
log("--------------------------------------------------");

2. Use the rest countries API URL -> https://restcountries.com/v3.1/all
and
```

display all the country flags in the console.

3. Use the same rest countries and print all countries names, region, subregion and

populations

Answers for Both 2 and 3rd question

*/

```
let xhrData = new XMLHttpRequest();

//console.log(xhrData);

xhrData.open("GET", "https://restcountries.com/v3.1/all");// Opening this link and getting the data

xhrData.send(); // After getting the data, sends to client

xhrData.onload = () => {

    const data = JSON.parse(xhrData.response);

    //console.log(data);

    for (getArr of data){

        var countryName = getArr["name"].common;

        var regionName = getArr["region"];

        var subRegName = getArr["subregion"];

        var popCount = getArr["population"];

        var flagId = getArr["flag"]


       console.log("Flag : " +flagId);

       console.log("Country : " +countryName+ " | Region : " +regionName+

       " | Sub Region : " +subRegName+ " | Population : " +popCount);

console.log("------------------------------------------------------------");

    }

}
```

o/p : A lot of countries is available just an sample example

Flag : MG

Country : Madagascar | Region : Africa | Sub Region : Eastern Africa | Population : 27691019

Flag : KR

Country : South Korea | Region : Asia | Sub Region : Eastern Asia | Population : 51780579

……

```
// 1. For the given JSON iterate over all for loops(for, for in, for of, for each)


var log = (param) => console.log(param);

log("This is 3rd Task");

log(".....................................");

const studentDetails = {

    name : "Kalilur Rahman",

    place : "Tiruvallur",

    age : 25,

    isProfessional : "ServiceNow Developer",

    degree : "B.Tech.CSE"

}

//General for loop

//Getting keys as arrays


var studentKeys = Object.keys(studentDetails);

console.log("The keys are :",studentKeys);
```

o/p :

1. The keys are : Array(5)
    1. **0**: "name"
    2. **1**: "place"
    3. **2**: "age"
    4. **3**: "isProfessional"
    5. **4**: "degree"
    6. **length**: 5

```
log(".....................For Loop.........................");

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

```
    console.log(i, "-",studentKeys[i]);
}
o/p:
0 '-' 'name'
1 '-' 'place'
2 '-' 'age'
3 '-' 'isProfessional'
4 '-' 'degree'
-----------------------------------------------------
var studentValues = Object.values(studentDetails);
console.log("The values are :", studentValues);
o/p :

for (let i = 0; i < studentValues.length; i++){
    console.log(i, "-",studentValues[i]);
}
o/p :
The values are : Array(5)
0 '-' 'Kalilur Rahman'
1 '-' 'Tiruvallur'
2 '-' 25
3 '-' 'ServiceNow Developer'
4 '-' 'B.Tech.CSE'

// For In Loop :
log("..................For In Loop.............................");
for (obj in studentDetails){

    if (obj == "name"){
        studentDetails[obj] = "Kalilur Rahman A R"; // --> way for
accessing values
    }
    console.log(obj + " : " +studentDetails[obj]);
```

```
}
log(studentDetails);
o/p :
name : Kalilur Rahman A R
place : Tiruvallur
age : 25
isProfessional : ServiceNow Developer
degree : B.Tech.CSE
-----------------------------------------------------
//For of Loop
log("...................For of Loop.............................");

for (arr of studentKeys){
        console.log(arr+ " : " +studentDetails[arr]); // --> way for
accessing values
}
o/p :
name : Kalilur Rahman A R
place : Tiruvallur
age : 25
isProfessional : ServiceNow Developer
degree : B.Tech.CSE
-----------------------------------------------------------
log("...................For each Loop.............................");
log(".........Getting keys..........");
studentKeys.forEach((values, index, actArray) => console.log(values,
index, actArray));
log(".........Getting Values..........");
studentValues.forEach((values, index, actArray) => console.log(values,
index, actArray));

// Create your own resume data in JSON format?

const myResume = {
```

```
    name : "Kalilur Rahman A R",
    Degree : "B.Tect.Computer Science and Engineering",
    experience : "2 years of ServiceNow Developer",
    skills : "MS Office, VS Code, Pycharm",
    languages : "Javascript, Python, HTML, Css",
    address : "No:18, Kumaran street, LK Nagar, Periyakuppam,
Tiruvallur",
    pincode : 602001,
    isMarried : false,
    cgpa : 83.96
}

log("--- Getting keys from myResume ---");
myKeys = Object.keys(myResume);
for (getKey in myKeys){
    console.log(getKey + ". " + myKeys[getKey]);
}

log("--- Getting Values from myResume ---");
myValue = Object.values(myResume);
for (getValue in myValue){
    console.log(getValue + ". " + myValue[getValue]);
}

log("--- Getting keys and Values from myResume ---");
for (myDetails in myResume){
    console.log(myDetails + " : " + myResume[myDetails] + ".");
}
/*
Read about the difference between window, screen and document in java
script?

window : is the execution context and global object for that context's
JavaScript
```

```
document : contains the DOM, initialized by parsing HTML
screen : describes the physical display's full screen

*/
```