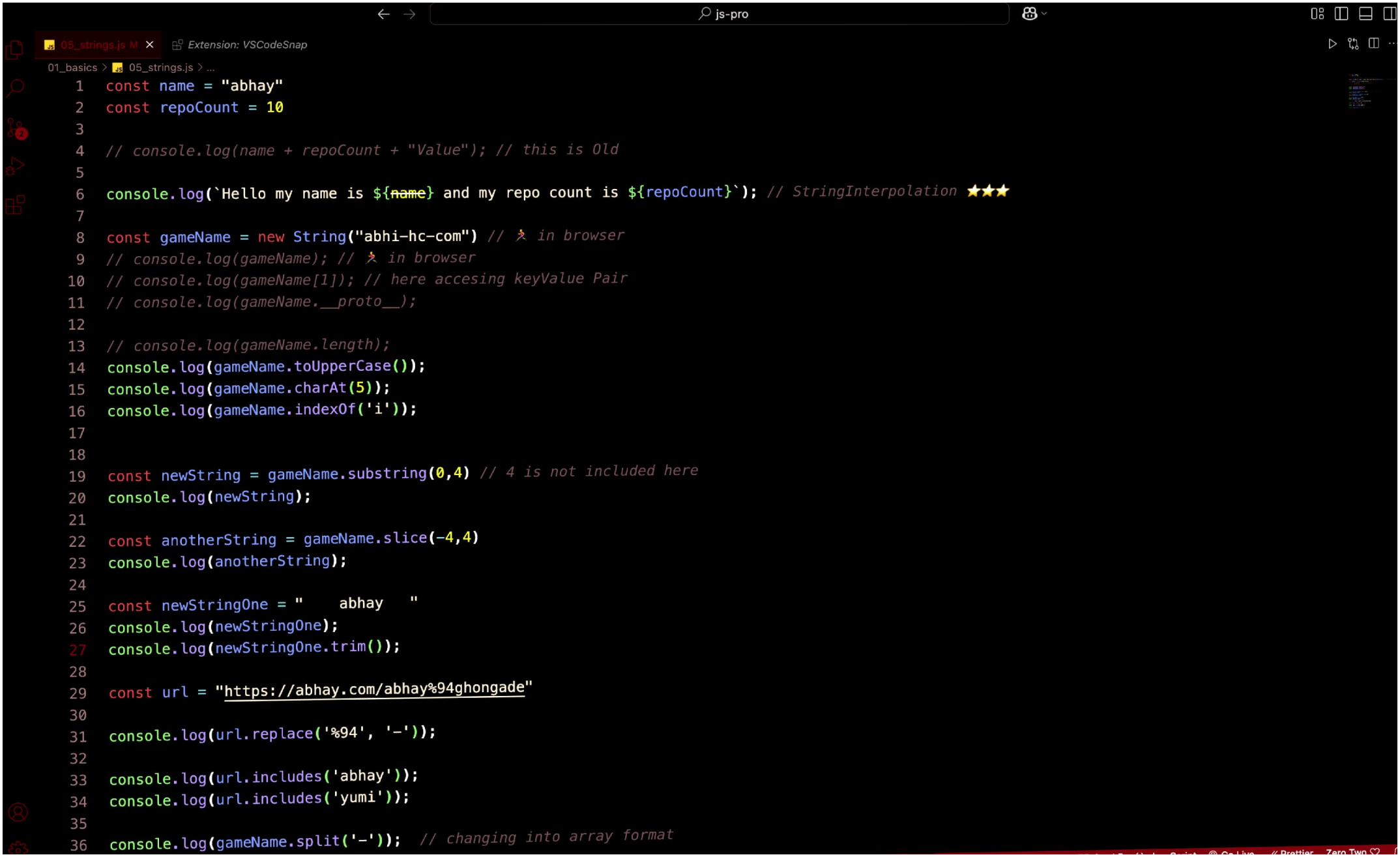
```js
const accountId = 2151711245042
let accountEmail = "ghongadeabhay60@gmail.com"
var accountPassword = "4321"
accountCity = "jalgaon"
let accountState;
// it will make undefined id no value assigned

// accountId = 2 // not allowed coz of const
// (node.js will not exicute)

accountEmail = "abhi.com"
accountPassword = "2121212"
accountCity = "bnglore"

console.log(accountId);

/*
Prefer not to use var
because of issue in block scope and fuctional scope
*/

console.table([accountId, accountEmail, accountCity, accountPassword, accountState])
// node 01_basics/01_variables.js
```

```
"use strict" // treat all JS code as newer version

// alert("3 + 3") // error in node js, not browser

console.log(3 +
    3
) // code readibillity should be high

console.log("abhay")


let name = "abhay"
let age = 20
let isloggedIn = false

// number => 2 to power 53
// bigint
// string => ""
// Boolean => true/false
// null => standalone value
// undefined =>
// Symbol => unique



// object

console.log(typeof age);
console.log(typeof null); // object (in terminal) o/p
console.log(typeof undefined); // undefined in terminal
```

```
let score = "abhay"

// console.log(typeof score);
// console.log(typeof (score));

let valueInNumber = Number(score)
// console.log(typeof valueInNumber);
// console.log(valueInNumber);  // NaN NotaNumber

// "33" => 33
// "33abc" => Nan
// true => 1; false => 0


let isLoggedIn = "abhay"
let booleanIsLoggedIn = Boolean(isLoggedIn)
// console.log(booleanIsLoggedIn);

// 1 => true; 0 => false
// "" => false
// "abhay" => true



let someNumber = 33

let stringNumber = String(someNumber)
// console.log(stringNumber)
// console.log(typeof stringNumber)

// ********************  Operations **********************

let value = 3
let negValue = -value
// console.log(negValue);

// console.log(2-2);
// console.log(2+2);
// console.log(2*2);
// console.log(2**3);
// console.log(2/3);
// console.log(2%3);

let str1 = "hello"
let str2 = " abhay"

let str3 = str1 + str2
console.log(str3);

// console.log(1 + "2");
// console.log("1" + 2);
// console.log("1" + 2 + 2);
// console.log(1 + 2 + "2");

// console.log( (3 + 4) * 5 % 3 ); // do-not write tricky code

// console.log(+true);
// console.log(+"");



let num1, num2, num3

num1 = num2 = num3 = 2 + 2  // no value to it readibility is prior

let gameCounter = 100
gameCounter++ // prefix (valueIncementerFirst) ++gameCounter postfix (valueIncementerAfterUsage)
console.log(gameCounter);
```

```
// console.log(2 > 1);   // ✅
// console.log(2 >= 1);   // ✅
// console.log(2 < 1);   // ✅
// console.log(2 <= 1);   // ✅
// console.log(2 != 1);   // ✅


// console.log("2" > 1);    // ! data type must be same while comparing ↔
// console.log("02" > 1);   // !

console.log(null > 0);   // avoid 🚫
console.log(null == 0);   // avoid 🚫
console.log(null >= 0);   // avoid 🚫


console.log(undefined == 0);   // avoid 🚫
console.log(undefined > 0);   // avoid 🚫
console.log(undefined < 0);   // avoid 🚫

// ===

console.log("2" === 2);
```


js-pro
05_strings.js M
Extension: VSCodeSnap
01_basics > 05_strings.js > ...
1 const name = "abhay"
2 const repoCount = 10
3
4 // console.log(name + repoCount + "Value"); // this is Old
5
6 console.log(`Hello my name is ${name} and my repo count is ${repoCount}`); // StringInterpolation ⭐⭐⭐
7
8 const gameName = new String("abhi-hc-com") // 🏃 in browser
9 // console.log(gameName); // 🏃 in browser
10 // console.log(gameName[1]); // here accesing keyValue Pair
11 // console.log(gameName.__proto__);
12
13 // console.log(gameName.length);
14 console.log(gameName.toUpperCase());
15 console.log(gameName.charAt(5));
16 console.log(gameName.indexOf('i'));
17
18
19 const newString = gameName.substring(0,4) // 4 is not included here
20 console.log(newString);
21
22 const anotherString = gameName.slice(-4,4)
23 console.log(anotherString);
24
25 const newStringOne = "   abhay   "
26 console.log(newStringOne);
27 console.log(newStringOne.trim());
28
29 const url = "https://abhay.com/abhay%94ghongade"
30
31 console.log(url.replace('%94', '-'));
32
33 console.log(url.includes('abhay'));
34 console.log(url.includes('yumi'));
35
36 console.log(gameName.split('-')); // changing into array format

```js
const score = 400
// console.log(score);

const balance = new Number(100)
// console.log(balance);

// console.log(balance.toString().length); // converting type of to string to get more properties
// console.log(balance.toFixed(2)); // =>100.00 while makeing e-commerse application for precision

const otherNumber = 1123.8966 // 23.8966=> 23.8
// 1123.8966=> 1.12e+3
// console.log(otherNumber.toPrecision(3)); // 123.8966=> 124

const hundreds = 1000000 // toLocaleString()=> 1,000,000
// console.log(hundreds.toLocaleString('en-In')); // 10,00,000

// Min_Value Max_Value by [Number.] in browser 🏃

// ++++++++++++++++++++ Maths ++++++++++++++++++++++
// library by-default

/*
console.log(Math); // console.log(Math); in browser 🏃
console.log(Math.abs(-4)); // .abs changes -ve => +ve
console.log(Math.round(5.7)); // => 6 ⭐
console.log(Math.ceil(4.2)); // => 5 [top value]
console.log(Math.floor(4.2)); // => 4 [lowest value]

console.log(Math.min(1, 2, 0.2, 0.314521 , 0.00032));
// finds min value in a array
console.log(Math.max(1, 2, 0.2, 0.314521 , 0.00032));
// finds max value in a array
*/
console.log(Math.random());
console.log((Math.random()*10) + 1);
console.log(Math.floor(Math.random()*10) + 1);

const min = 10
const max = 20
// ⭐⭐⭐
console.log(Math.floor(Math.random() * (max - min + 1))+ min);

```

```javascript
// date javascript mdn 🔎
/*
let myDate = new Date()
console.log(myDate.toString());
console.log(myDate.toDateString());
// console.log(myDate.toISOString());
// console.log(myDate.toJSON());
// console.log(myDate.toLocaleDateString());
console.log(myDate.toLocaleString());
console.log(typeof myDate);   // => object [instance] ⭐
*/

// let myCreatedDate1 = new Date(2024, 06, 07, 21, 50)
let myCreatedDate2 = new Date("2024-06-07")
let myCreatedDate = new Date("06-07-2024")
// console.log(myCreatedDate.toDateString());
// console.log(myCreatedDate.toLocaleString());

let myTimeStamo = Date.now()

// console.log(myTimeStamo); // millisec Values counted frm 1970
// console.log(myCreatedDate.getTime()); //while making clone 🤡
// console.log(Math.floor(Date.now()/1000));


let newDate = new Date()
console.log(newDate.getMonth() + 1 );
console.log(newDate.getDate());
console.log(newDate.getFullYear());

// console.log(`${newDate.getDate()} and the month ${newDate.getFullYear()} ${newDate.getMinutes()} minutes`);

newDate.toLocaleString('default',{
    weekday: "long", // Ctrl + space [short-cut key]
 // timeZone: ''
})   // defines objects and many parameters ⭐⭐
```

```js
// primitive (callByValue)

// 7 types(thala7) : String, Number, Boolean, null, undefined, Symbol, BigInt

const score = 100 // dynamically type lang coz it takes any dataTypes
const scoreValue = 100.3

const isLoggedIn = false
const outsideTemp = null // => Object
let userEmail; // => undefined

const id = Symbol('123')  // => symbol
const anotherId = Symbol('123') // => symbol

// console.log(id === anotherId);

const bigNumber = 6565164448484564884648n


// Reference (Non primitive)

// Array, Objects, Functions

const heros = ["shaktiman", "naagraj", "thanos"] // heros=> Obje
let myObj = { // => Object
    name: "abhay",
    age: 22,
}

const myFunction = function () { // myFunction(returns)=> functionObject
    console.log("Hello world");
}


console.log(typeof myObj);


// ++++++++++++++++++++++++++++++++++++++++++++

// Stack (Primitive[copy]), Heap (Non-Primitive[refrenceOfOriginalValue])

let myYtName = "abhay.com"

let anotherName = myYtName // copy in stack of myYt
anotherName = "chaiaurcode"

console.log(myYtName);
console.log(anotherName);


let userOne = { // OriginalValueRefrence in Heap
    email: "user@google.com",
    upi: "user@ybl"
}

let userTwo = userOne

userTwo.email = "abhay@google.com"

console.log(userOne.email);
console.log(userTwo.email);
```

```
// Array javascript mnd
// const numbers = [1, 2, 3, 4 ] 🏃
// numbers 🏃

const myArr = [0, 1, 2, 3, 4, 5]
const myHerros = ["ironman", "thanos"]

const myArr2 = new Array(1,2,3,4) // automatically ads square bracket
// console.log(myArr[1]);

// Array methods

// myArr.push(6)
// myArr.push(7)
// myArr.pop()

// myArr.unshift(9)
// myArr.shift()

// console.log(myArr.includes(9)); // gives ans 1 or 0 and imp ⭐ boolean datatype
// console.log(myArr.indexOf(9)); // => -1 [not present in array then]

// const newArr = myArr.join()

// console.log(myArr);
// console.log(typeof newArr); // => string




// slice, splice 🔪⭐

console.log("A",myArr);

const myn1 = myArr.slice(1, 3)
console.log(myn1);

console.log("B", myArr);

const myn2 = myArr.splice(1, 3) // array⭐manipulation

console.log(myn2);
console.log("C", myArr); // array⭐manipulation
```

```
// const myArr = [1, 2, 3, 4, 5] 🏃
const marval_heros =["thor", "ironman", "spiderman"]
const dc_heros =["superman", "flash", "batman"]

// marval_heros.push(dc_heros) // pushes same array

// console.log(marval_heros);  // =>[ 'thor', 'ironman', 'spiderman', [ 'superman', 'flash', 'batman' ] ]
// console.log(marval_heros[3][1]); // =>flash {for accesing}

// const allHeros = marval_heros.concat(dc_heros) // concat returns new array
// console.log(allHeros); // => [ 'thor', 'ironman', 'spiderman', 'superman', 'flash', 'batman' ]


// काच का ग्लास था drop करा बिखर जाएगा spread होजाएगा.
const all_new_heroes = [...marval_heros, ...dc_heros, ...dc_heros] // spread out values milengi ek ek element millega


// console.log(all_new_heroes);

const another_arrray =[1, 2, 3, [4, 5, 6], 7,[6, 7, [4, 5]]]

const real_another_array = another_arrray.flat(Infinity)
console.log(real_another_array);



console.log(Array.isArray("Abhay"));
console.log(Array.from("Abhay"));
console.log(Array.from({name: "abhay"})); // intresting 🚂

let scor1 = 100
let scor2 = 200
let scor3 = 300

console.log(Array.of(scor1, scor2,scor3));
```

```
// Object.create
// singelton (object by making a conster)अपने तरेह का एक ही object है

// object literals

const mySym = Symbol("key1") // define symbol


const jsUser = {
    name: "Abhay", // we can define keys as well as values
    "full name": "Abhay Ghongade",
    location: "jalgaon",
    email: "abh@.com",
    [mySym]: "mykey1",   // correct syntax for using symbol act as a key 🚂
    isLoggedIn: false,
    lastLoginDays: ["Monday", "Sunday"]
}

// console.log(jsUser.lastLoginDays); // if key is in string then not works eg="full name":
// console.log(jsUser["name"]);
// console.log(jsUser["full name"]);
// console.log(typeof jsUser[mySym]); // ⭐🚂


// jsUser.email = "abhay@chatgpt.com" // for changing values
// Object.freeze(jsUser) // freezes so no one could change
// jsUser.email = "abhay@madarchood.com" // not propogate(change)
// console.log(jsUser);


jsUser.greeting = function (){ // greeting is common
    console.log("Hello JS user");
}
jsUser.greetingTwo = function (){
    console.log(`Hello JS user, ${this.name}`);// `stringinterpolation` backticks // this(shows properties of oblect)
}

console.log(jsUser.greeting());
console.log(jsUser.greetingTwo());

// umdefined ⁉
```

```js
// const tinderUser = new Object() //singelton Object
 const tinderUser = {} // Non-singelton Object

// left  + up  teminal copy

tinderUser.id = "123abc" // unique id देदेते है
tinderUser.name = "samm"
tinderUser.isLoggedIn = false

//  console.log(tinderUser);
const regularUser = {
    email: "abhay@gmail.com",
    fullname: {
        userfullname : {
            firstname: "abhay",
            lastname: "ghongade" // Nesting object के अंदर object
        }
    }
}

// console.log(regularUser.fullname);
// console.log(regularUser.fullname.userfullname); // fullname?
// console.log(regularUser.fullname.userfullname.lastname);

const obj1 = {1: "a", 2: "b"}
const obj2 = {3: "a", 4: "b"}
const obj4 = {6: "a", 7: "b"}

// const obj3 = {obj1, obj2} // mtd
// const obj3 = Object.assign({}, obj1, obj2, obj4) // object assign

// काच का ग्लास था drop करा बिखर जाएगा spread होजाएगा.
const obj3 = {...obj1, ...obj2} //
// console.log(obj3);



const user = [ // array of objects from database
    {
        id: 1,
        email: "abh@gmail.com"
    },
    {
        id: 1,
        email: "abh@gmail.com"
    },
    {
        id: 1,
        email: "abh@gmail.com"
    },
]

user[1].email
console.log(tinderUser);

console.log(Object.keys(tinderUser)); // => comes in array so we can apply loops
console.log(Object.values(tinderUser));
console.log(Object.entries(tinderUser)); // array mai array

console.log(tinderUser.hasOwnProperty('isLoggedIn')); // to check it has property

// ++++++++++++++++++++++ de_Structuring ++++++++++++++++++++++

const course = {
    coursename: "js in h",
    price: "999",
    courseInstructor: "hitesh"
}

// course.courseInstructor
const {courseInstructor: instructor} = course // destructuring object into small value

// console.log(courseInstructor);
console.log(instructor);


/* // react example for de Structuring using {} {company}
const navbar = ({company}) => {

}

navbar(company = "abhay") */


// +++++++++++++++ JSON {} ++++++++++++++++
// +++++++++++++++ API's call {JSON Object format } https://api.github.com/users/hiteshchoudhary conert into object (fletch)
// {
//     "key in string": "values in string",
//     "name": "abhi",
//     "coursename": "js",
//     "price": "0"
// }

// API in format of Array(applying loops etc)
[
    {},
    {},
    {}
]

// https://randomuser.me/  => https://jsonformatter.org/
```

```js
// code ko package mai band kar diya hai
function sayMyName() { // scope, defination
    console.log("A");
    console.log("B");
    console.log("H");
    console.log("A");
    console.log("Y");
}

// sayMyName'is_refrence' ()'is_execution'
// ⬆️⬆️⬆️useful while react or onclick & dom Manipulation
// sayMyName()



/*

function addTwoNumbers(number1, number2){ //(number1, number2)Parameters
    console.log(number1 + number2); // type 1️⃣
}
addTwoNumbers(3, "a") // (3, 4)Arrgument
addTwoNumbers(3, "4")
addTwoNumbers(3, null)

const result = addTwoNumbers(3,5)
// problem ⬇️
console.log("Result: ",result);

*/

function addTwoNumbers(number1, number2){ //(number1, number2)Parameters

    let result = number1 + number2 // type 2️⃣
    /* console.log("abhay"); */
    return result
    console.log("abhay"); // this is unreachable code coz of return

    /* return number1 + number2 */ // type 3️⃣
    // return सिर्फ variable से store कर सकते है | नाकी console log se ⭐
}

const result = addTwoNumbers(3,5)
// console.log("Result: ",result); // =>Undefined for type1️⃣ & 8 for 2️⃣



function logInUserMessage(username /* = "sam |so no reuqurment of if statement"  */) {
    if (!username/* username=== undefined */) {
        console.log("please enter a username");
        return
    }
    return `${username} just logged in`
}

console.log(logInUserMessage(/* when u not pass nothing */));
```

```
/* ◆shoping cart in e-commerse website
◆problem!! user keeps on adding multiple no. of items
◆so, u dont know how many no. of argument/items are going to come
◆then how to create parameters in function */

function calculateCartPrice(/* val1, val2, */ ...num1 /* ...is a rest/spread operator specified upon useCase */) { // also cant use two operator at time error⚠
    return num1
}

// console.log(calculateCartPrice(200, 500, 400, 2000));


/* ◆object pass Usecase*/
const user = {
    username: "abhay",
    price: 199
}

function handleObject(anyobject){
    console.log(`Usernmae is ${anyobject.username} and price is ${anyobject.price}`);
}

handleObject(user) /* functionCall */
/* handleObject({ // or direct Object pass
    username: "sam",
    price: 99
}) */




/* ◆Array pass Usecase*/
const myNewArray = [200, 400 ,100, 600]

function returnSecondValue(getArray) {
    return getArray[2]
}

// console.log(returnSecondValue(myNewArray));
console.log(returnSecondValue([200, 400, 500, 1200]));
```

```
/*
🔸जब तीनो variable काम कर रहे थे तो जरूरत क्या थी तीनो को लाने की
🔸var block scope के उपर काम नही करता है
*/

/* ⭐ global scope की कहाणी अंदर scope में available होती है */

// var c = 300
let a = 300

if (true) {
    /* ⚠️ block scope के अंदर की कहाणी बाहर leak नही जानी चाहिये */
    let a =10
    const b = 20
    // var c = 30
    console.log("INNER: ", a);
} /* curly brases हर language में scope है
    💥 जितनी बार curly brases आएगा उतनी बार scope आएगा */


// console.log(a);
// console.log(b);
// console.log(c);


/*
🚽
🍕जब आप browser में inspect करके console में scope अलग है
🌠और जब आप code environment node में run करते है तो global scope अलग है
*/



/*
Part 2️⃣
💠Nested scope
💡छोटा बच्चा बडो से ice-cream मांग सकता है लेकीन, बडे छोटे बच्चो से ice-cream नही मांग सकते
💠closure(basic not detailed) child func can access parent variable
similar to this nested scope works
*/

function one() {
    const username = "abhay"

    function two() {
        const website = "youtube"
        console.log(username);
    }
    /* जितनी बार func declare और call करते है उनके लिये अलग से call stack बनता है  */
    // console.log(website /* error❌ */);

    two() // execution

}

// one ()



if (true) {
    const username = "abhay"
    if (username === "abhay") {
        const website = " study"
        // console.log(username + website);
    }
    // console.log(website /* error❌ */);
}

// console.log(username /* error❌ */);



// +++++++++++++++++ intresting +++++++++++++++++

console.log(addOne(5)); // func is only declared so no error

function addOne(num) {
    return num + 1
}

// addTwo(5 /* error❌ */) /* coz of ||hoisting|| func is declared & holded in variable */

const addTwo /* callAs"Expression" like variable very powerfull it can hold JSON values, func almost anyting can be hold inside variable */ = function (num) {
    return num + 2
}

addTwo(5)
```

```js
const user = {
    username: "abhay",
    price: 999,

    welcomeMessage: function () {
        console.log(`${this.username} , welcome to website`);
        // this is used for refering current context here current context is between object for acccesing scope-variables
        console.log(this);
    }

}

// user.welcomeMessage()
// user.username = "sam" //context change kar diya (matlab ye chheej hai kis bare mai)
// user.welcomeMessage()

// console.log(this); // current context => empty Object
/* console.log(this) 🏃🌍 biggest globle Object in 🌍 browser is "Window(capture's event)"*/

/*
function chai(){
    let username = "abhay"
    console.log(this.username); // => undefined
}

chai()
*/

/*
const chai = function () {
    let username = "abhay"
    console.log(this.username); // इसको भी नही पता
}
*/

/*       Now 💠Arrow function         */

const chai = () => {
    let username = "abhay"
    console.log(this);  // => {}
}


// chai()

// const addTwo = (num1, num2) => {
//     return num1 + num2
// }

/*          🔸2️⃣Implicite return     */
// const addTwo = (num1, num2) =>  num1 + num2

/*          🔸3️⃣Paramthesis use    */
// const addTwo = (num1, num2) =>  ( num1 + num2 ) // useful while react

const addTwo = (num1, num2) =>  ( {username: "hitesh"} )

console.log(addTwo(4,9));




const myArray = [2, 5, 4, 7, 8]

// myArray.forEach(function () {}) // eg of correct syntax
// myArray.forEach(() => {})
// myArray.forEach(() => ())
```

```
// Immediately Invoked Function Expression (IIFE)
// 🚬 globle scope के polution से problem होती है कई बार तो globle scope जो भी variable या declaration से polution हटाने के लिये हमने IIFE का इस्तेमाल किया

(function chai (){
    // named IIFE
    console.log(`DB CONNECTED`);
})();

( (name /* parameter pass */) => {
    // un-named IIFE
    console.log(`DB CONNECTED TWO ${name /* parameter pass */}`);
})('abhay')
```

```
// कूच situation हो तो ये control execute हो "or"
// कूच situation हो तो ये control execute हो


// if
const isUserLoggedIn = true
const temprature = 5
/* //1️⃣
if (temprature === 52) {
    console.log("less than 50");
} else {
    console.log(`temprature is greater than 50 ${temprature}`);
}

console.log("execute");
*/

// <, >, <=, >=, ==, !=, ===, !==


/* //2️⃣
const score = 200

if (score > 100) {
    let power = "fly"  // var is a global scope
    console.log(`User power: ${power}`);
}

console.log(`User power: ${power}`);
*/


/* //3️⃣
const balance = 1000

// if (balance > 500) console.log("test"),console.log("test2"); // 🔸2️⃣Implicite scope in 03_basics/03_Arrow.js ❌❌❌but do not write such a code


if (balance < 500) {
    console.log("less than 500");

}   else if (balance < 750) {
    console.log("less than 750");

} else if (balance < 900) {
    console.log("less than 900");

} else {  // their is condition for else
    console.log("less than 1200");

}
*/

  //4️⃣
const isUserLoggedIn2 = true
const debitCard = true
const loggedInFromGoogle = false
const loggedInFromEmail = true

if (isUserLoggedIn2 && debitCard && 2==3 /* 2==2 ⭐all conditon true*/) {
    console.log("allow to buy course");
}

if (loggedInFromGoogle || loggedInFromEmail /* multiple or can be used || ⭐one/atleast one condition true अपना  काम करेगा condition के अंदर जाऊंगा*/) {
    console.log("User logged in");
}
```

```
// बोहत ज्यादा check करणा होता है तो if else लिखते लिखते परेशान हो जाते है


/* switch (key) {
    case value:

        break;

    default:
        break;
} */


const month = "march"

switch (month) {
    case "jan":
        console.log("january");

        break; // control flow ko break kar deta hai
    case "feb":
        console.log("feb");

        break;
    case "march":
        console.log("march");

        break; // jha pe bhi case match ho hogaya sara ka sara code run karta hai except default
    case "april":
        console.log("april");

        break;

    default:
        console.log("default case match");

        break;
}
```

```
// const userEmail = "@bhay.ai"// string के अंदर मान लिया गया है कि truthy value hai
// const userEmail = ""// string के अंदर मान लिया गया है कि falsy value hai

const userEmail = []// Array के अंदर मान लिया गया है कि truthy value hai

if (userEmail) {
    console.log("got user email");

} else {
    console.log("Dont have user email");

}


/* falsy values 🔕
false, 0, -0, BigInt 0n, "", null, undefined, NaN
 */

/* truthy values 🔔
"0", 'false',  " ", [], {}, function(){}
 */


// ++++++++++++ How to detect Array or Object is empty?

// if (userEmail.length === 0) {
//     console.log("Array is empty");

// }


const emptyObj = {}

if (Object.keys(emptyObj).length === 0) {
    console.log("Object is empty");

}

/*
false == 0      => true
false == ''     => true
0 == ''         => true
 */


// Nullish Coalescing Operator (??): null Undefined

let val1;
// val1 = 5 ?? 10 // 5
// val1 = null ?? 10 // 10
var1 = undefined ?? 15 // 15
val1 = null ?? 10 ?? 30 // jo first ata hai vo assign hota jaise 10



console.log(val1);
console.log(var1);


//++++++++++++ Terniary Operator not same to nullish Coalescing Operator

// condition ? true : false
const iceTeaPrice = 100
iceTeaPrice <= 80 ? console.log("less than 80") : console.log("more than 80");

```

```
// for

for (let index = 0/* is statement main loop kabhi aega hi nahi baadmaii */; index < 10/*↙ cond checking hoti hai ↩️↩️*/;/*↩️↩️*/ index++)
{   const element = index;
    if (element == 5) {
        // console.log("5 is best number");

    }
    // console.log(element);//↙database ko call karna hai/ print karana hai ye sab kam hoga

/* just block scope khatam hone se pahele index ki value ko badha deta hai (↗execution control)*/}

// console.log(element);


for (let i = 1; i <= 10; i++) {
/*     console.log(`
        Outer loop value: table no.${i}
        `);

    for (let j = 1; j <= 10; j++) {
    // console.log(`Inner loop value: ${j} and Inner loop value: ${i}`);
        console.log(`${i}  *  ${j}   =   ${i*j}`);


    }
 */}


 let myArray = ["flash", "batman", "superman"]
// console.log(myArray.length);

 for (let index = 0; index < myArray.length; index++) {
    const element = myArray[index];
    // console.log(element);

 }


 for (let index = 0; index <= 10; index++) {
    // console.log(`${index}`); // done
    if (index == 0) {
        // console.log(`Fuck off Bitch ${index}`);

    }
 }




// 🔸break and continue

/* for (let index = 1; index <= 20; index++) {
    if (index == 5) {
        console.log(`Detected 5`);
        break // contol flow ko break karke block scope sidha jump karke sidha line 60 pe bahar
    }
    console.log(`${index}`);

}
 */

for (let index = 1; index <= 20; index++) {
    if (index == 5) {
        console.log(`Detected 5`);
        continue // continue ek bar skip kardo `ignore ` no 5 is printed
    }
    console.log(`${index}`);

}
```

```
1
2  let index = 0
3  while (index <= 10) {
4      // console.log(`value of index is ${index}`);
5      index = index + 2
6  }
7
8
9
10
11 let myArray = ["flash", "superman", "batman"]
12 let arr = 0
13 while (arr < myArray.length) {
14     // console.log(`value is ${myArray[arr]}`);
15     arr = arr + 1
16
17 }
18
19
20 let score = 11
21
22 do { // kam pehele condition badmai || udhari wala system
23     console.log(`score is ${score}`);
24     score++
25 } while (score <= 10);
```

```
// For of

// ["", "" ,""]
// [{}, {}, {}]

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

for (const num of arr/* object  से मतलब ये है कि कीस चीज पे loop lagana hai*/) {
    // console.log(num);

}

const greetings = "Hello world"
for (const greet of greetings) {
    // console.log(`Each char is ${greet}`);

}

// Maps

const map = new Map()
map.set('IN', "India")
map.set('USA', "United States of America")
map.set('Fr', "France")
map.set('IN', "India")

// console.log(map);
for (const [key, value] of map) {
    // console.log(key, ':-', value);

}



const myObject = {
    Game1 : 'NFS',
    Game2 : 'bgmi',
}

// for (const [G, values] of myObject) {
//     console.log([G, ':-', values]);
    /*! Forof loop does not works for object */
// }
```

```javascript
/*----⬇️ forin loop on Object ----*/

const myObject = {
    js : 'javascript',
    cpp: 'C++',
    rb: 'ruby',
    swift: "swift by apple"
}

for (const key in myObject) {
    // console.log(`${key} shortcut is for ${myObject[key]}`);

}


/*----⬇️ forin loop on array ----*/

const programming = ["js", "rb", "py", "java", "cpp"]

for (const key in programming) {
    // console.log(key, programming[key], `${programming[key]}`);

}


/*----⬇️ forin loop on maps ----*/

const map = new Map()
map.set('IN', "India")
map.set('USA', "United States of America")
map.set('Fr', "France")
map.set('IN', "India")

for (const key in map) {
    console.log(`${map[key]}`);
    /*! Forin loop not works on maps because it is not iterable */

}
```

```
1  const coding = ["js", "ruby", "java", "python", "cpp"]
2
3  /*--- variation 1 for printing by function---*/
4  coding.forEach( function (val) {
5      // console.log(val);
6
7  } )
8
9
10 /*--- variation 2 for printing by arrow function---*/
11
12 coding.forEach( (item)  => {
13     // console.log(item);
14
15 } )
16
17
18 /*--- variation 3 for printing by ---*/
19
20 function printeMe(item) {
21     console.log(item);
22
23 }
24
25 coding.forEach(printeMe())
```

```
/* const coding = ["js", "ruby", "java", "python", "cpp"]


const values = coding.forEach( (item) => { // forEach koi bhi value return hi nahi karta hai
    // console.log(item);
    return item
})

console.log(values);
 */


const myNums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

// const newNums = myNums.filter( (num) => /* num > 4 */{
//     return num > 4
// })
// console.log(newNums);


// const newNums = []

// myNums.forEach( (num) => {
//     if (num > 4) {
//         newNums.push(num)
//     }
// })

// console.log(newNums);



const books = [
    { title: 'Book One', genre: 'Fiction', publish: 1981, edition: 2004 },
    { title: 'Book Two', genre: 'Non-Fiction', publish: 1992, edition: 2008 },
    { title: 'Book Three', genre: 'History', publish: 1999, edition: 2007 },
    { title: 'Book Four', genre: 'Non-Fiction', publish: 1989, edition: 2010 },
    { title: 'Book Five', genre: 'Science', publish: 2009, edition: 2014 },
    { title: 'Book Six', genre: 'Fiction', publish: 1987, edition: 2010 },
    { title: 'Book Seven', genre: 'History', publish: 1986, edition: 1996 },
    { title: 'Book Eight', genre: 'Science', publish: 2011, edition: 2016 },
    { title: 'Book Nine', genre: 'Non-Fiction', publish: 1981, edition: 1989 },
  ];

  let userBooks = books.filter( (bk) => bk.genre === 'History' )

  userBooks = books.filter( (bk) =>  {
    return bk.publish >= 1995 && bk.genre === 'History'
} )

  console.log(userBooks);

```

```
1  const myNumbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2
3  // const newNums = myNumbers.map( (num) => { return num + 10})
4  // personaly map acha lagta hai forEach see
5
6
7  /* 🔸chaining by maps */
8  const newNums = myNumbers
9                  .map((num) => num * 10 ) // sabhi return hoga
10                 .map( (num) => num + 1 )
11                 .map( (num) => num - 3 )
12                 .filter((num) => num >= 40) // true or false
13 /* Indendt |Tab space|  */
14 console.log(newNums);
15
```

```
// shopping cart addition of total price

const myNums = [1, 2, 3]

// const myTotal = myNums.reduce(function (acc, currval) {
//     console.log(`acc: ${acc} and currval ${currval}`);

//     return acc + currval
// }, 0)

const myTotal = myNums.reduce( (acc, curr) => acc + curr, 0)

console.log(myTotal);


// example

const shoppingCart = [
    {
        itemname: "lana",
        price: 2999
    },
    {
        itemname: "mia",
        price: 999
    },
    {
        itemname: "lexi",
        price: 5999
    },
    {
        itemname: "alyx",
        price: 12999
    },
]

const total = shoppingCart.reduce((acc, item) => acc + item.price, 0)

console.log(total);

```

```
// console.dir(document🏃)
// console.log(document.baseURL🏃)
// console.log(document.links[]🏃)

// HTML is a collection converted into array
<!-- document.getElementById('') -->
<!-- document.getElementById('firstHeading').innerHTML = "<h1>abhay</h1>" -->

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">

    <title>DOM learnig</title>
</head>
<body>
    <div class="bg-black">
        <h1 class="heading">DOM (Document Object Model)</h1>
        <p>Lorem ipsum dolor sit amet.</p>
    </div>
</body>
</html>
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DOM</title>
</head>
<body style="background-color: black; color: white;">
    <div class="parent">
        <!-- This is a comment -->
        <div class="day">Monday</div>
        <div class="day">Tuesday</div>
        <div class="day">Wednesday</div>
        <div class="day">Thursday</div>
    </div>
</body>
<script>
    const parent = document.querySelector('.parent')
    /*
    console.log(parent);
    console.log(parent.children); // we get HTMLCollection(4) [div.day, div.day, div.day, div.day] array like collection
    console.log(parent.children[1].innerHTML); // for accesing element
 */

/*
    for (let i = 0; i < parent.children.length; i++) {
       console.log(parent.children[i].innerHTML);

    }
 */

    parent.children[1].style.colour = "orange"
    // console.log(parent.firstElementChild/* property */);
    // console.log(parent.lastElementChild/* property */);


    const dayOne = document.querySelector/* dom it is a tree structure and querySelector traveses in tree => it requires time */('.day')
    // console.log(dayOne.parentElement/* trackes parent */);
    // console.log(dayOne.nextElementSibling/* trackes Sibling */);

    console.log("NODES: ", parent.childNodes/* shows nodelist and what does browser does behind the senes complex tree structure bana raha hai */);



</script>
</html>
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DOM</title>
</head>
<body style="background-color: black; color: white;">

</body>
<script>
    const div = document.createElement('div'/* here elemment is created h1, div */)
    console.log(div);
    div.className = "main" // ClassName and id are attributes
    div.id = Math.round(Math.random() * 10 + 1)
    div.setAttribute("title", "generated title")
    div.style.backgroundColor = "green"
    div.style.padding = "12px"
    // div.innerText = "assh karte hai"
    /* .className, .id, .innerText(values ko over-write karata hai jo bhi values hai select kr le lata hai fir se karta hai ) ke uper log jyada .setAttribute preffer karte hai kuu ki (ye direct hi uske andar set karta hai so, ek rounter bachta hai yaha pe) */
    const addText = document.createTextNode("chaloo assh karte hai")


    document.body.appendChild(div)

</script>
</html>
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DOM</title>
</head>
<body style="background-color: #212121; color: #fff;">
    <ul class="language">
        <li>javascript</li>
    </ul>
</body>
<script>
    function addLanguage(langName) {
       const li = document.createElement('li');
       li.innerHTML = `${langName}`
       document.querySelector('.language').appendChild(li)
    }
    addLanguage("python")
    addLanguage("typescript")

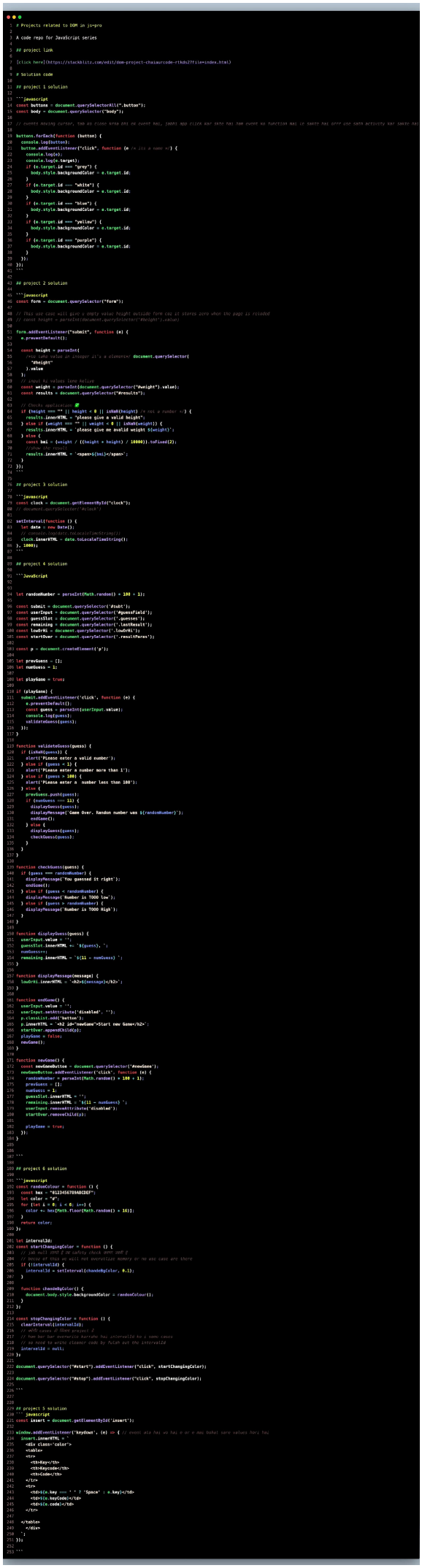
    function addOptiLanguage(langName){
        const li = document.createElement('li'); // concentrated seprately bcoz of scope
        li.appendChild(document.createTextNode(langName)) // always go with append child ⭐ this method
        document.querySelector('.language').appendChild(li)
    }
    addOptiLanguage("go lang")

    //Edit values
    const secondLang = document.querySelector("li:nth-child(2)")
    console.log(secondLang);
    //secondLang.innerHTML = "BKL" 1️⃣
    const newli = document.createElement('li')
    newli.textContent = "Mojo"
    secondLang.replaceWith(newli) //2️⃣

    //edit 3️⃣
    const firstLang = document.querySelector("li:first-child")
    firstLang.outerHTML = '<li>Typo-script</li>'

    //remove
    const lastLang = document.querySelector("li:last-child")
    lastLang.remove()

</script>
</html>
```



```
# Projects related to DOM in js-pro

A code repo for JavaScript series

## project Link

[click here](https://stackblitz.com/edit/dom-project-chaiaurcode-rtkds2?file=index.html)

# Solution code

## project 1 solution

```javascript
const buttons  = document.querySelectorAll(".button");
const body = document.querySelector("body");

// events moving cursor, tab ka close krna dni ne event hai, jabhi app click kar skte hai hme event ka function mai ie sakte hai orr use same activity kar skte hai

buttons.forEach(function (button) {
  console.log(button);
  button.addEventListener("click", function (e /* e its a name */) {
    console.log(e);
    console.log(e.target);
    if (e.target.id === "grey") {
      body.style.backgroundColor = e.target.id;
    }
    if (e.target.id === "white") {
      body.style.backgroundColor = e.target.id;
    }
    if (e.target.id === "blue") {
      body.style.backgroundColor = e.target.id;
    }
    if (e.target.id === "yellow") {
      body.style.backgroundColor = e.target.id;
    }
    if (e.target.id === "purple") {
      body.style.backgroundColor = e.target.id;
    }
  });
});
```

project 2 solution

```javascript
const form = document.querySelector("form");

// This use case will give a empty value height outside form cos it stores zero when the page is reloaded
// const height = parseInt(document.querySelector('#height').value)

form.addEventListener("submit", function (e) {
  e.preventDefault();

  const height = parseInt(
    /* to take value as integer it's a element/ document.querySelector(
      "#height"
    ).value
  );
  // input ki values lene kelliye
  const weight = parseInt(document.querySelector("#weight").value);
  const results = document.querySelector("#results");

  // Check validation
  if (height === "" || height < 0 || isNaN(height) /* not a number */) {
    results.innerHTML = "please give a valid height";
  } else if (weight === "" || weight < 0 || isNaN(weight)) {
    results.innerHTML = `please give me avalid weight ${weight}`;
  } else {
    const bmi = (weight / ((height * height) / 10000)).toFixed(2);
    //show the result
    results.innerHTML = `<span>${bmi}</span>`;
  }
});
```

project 3 solution

```javascript
const clock = document.getElementById("clock");
// document.querySelector('#clock')

setInterval(function () {
  let date = new Date();
  // console.log(date.toLocaleTimeString())
  clock.innerHTML = date.toLocaleTimeString();
}, 1000);
```

project 4 solution

```JavaScript


let randomNumber = parseInt(Math.random() * 100 + 1);

const subMit = document.querySelector('#subt');
const userInput = document.querySelector('#guessField');
const guessSlot = document.querySelector('.guesses');
const remaining = document.querySelector('.lastResult');
const lowOrHi = document.querySelector('.lowOrHi');
const startOver = document.querySelector('.resultParas');

const p = document.createElement('p');

let prevGuess = [];
let numGuess = 1;

let playGame = true;

if (playGame) {
  subMit.addEventListener('click', function (e) {
    e.preventDefault();
    const guess = parseInt(userInput.value);
    console.log(guess);
    validateGuess(guess);
  });
}

function validateGuess(guess) {
  if (isNaN(guess)) {
    alert('Please enter a valid number');
  } else if (guess < 1) {
    alert('Please enter a number more than 1');
  } else if (guess > 100) {
    alert('Please enter a  number less than 100');
  } else {
    prevGuess.push(guess);
    if (numGuess === 11) {
      displayGuess(guess);
      displayMessage(`Game Over. Random number was ${randomNumber}`);
      endGame();
    } else {
      displayGuess(guess);
      checkGuess(guess);
    }
  }
}

function checkGuess(guess) {
  if (guess === randomNumber) {
    displayMessage(`You guessed it right`);
    endGame();
  } else if (guess < randomNumber) {
    displayMessage(`Number is TOOO Low`);
  } else if (guess > randomNumber) {
    displayMessage(`Number is TOOO High`);
  }
}

function displayGuess(guess) {
  userInput.value = '';
  guessSlot.innerHTML += `${guess}, `;
  numGuess++;
  remaining.innerHTML = `${11 - numGuess} `;
}

function displayMessage(message) {
  lowOrHi.innerHTML = `<h2>${message}</h2>`;
}

function endGame() {
  userInput.value = '';
  userInput.setAttribute('disabled', '');
  p.classList.add('button');
  p.innerHTML = `<h2 id="newGame">Start new Game</h2>`;
  startOver.appendChild(p);
  playGame = false;
  newGame();
}

function newGame() {
  const newGameButton = document.querySelector('#newGame');
  newGameButton.addEventListener('click', function (e) {
    randomNumber = parseInt(Math.random() * 100 + 1);
    prevGuess = [];
    numGuess = 1;
    guessSlot.innerHTML = '';
    remaining.innerHTML = `${11 - numGuess} `;
    userInput.removeAttribute('disabled');
    startOver.removeChild(p);

    playGame = true;
  });
}


```

project 6 solution

```javascript
const randomColour = function () {
  const hex = "0123456789ABCDEF";
  let color = "#";
  for (let i = 0; i < 6; i++) {
    color += hex[Math.floor(Math.random() * 16)];
  }
  return color;
};

let intervalId;
const startChangingColor = function () {
  // jab null nhi h to safety check nhi lgti h
  // hence of this we will not overutilize memory or no use case are there
  if (!intervalId) {
    intervalId = setInterval(changeBgColor, 0.1);
  }

  function changeBgColor() {
    document.body.style.backgroundColor = randomColour();
  }
};

const stopChangingColor = function () {
  clearInterval(intervalId);
  // more cases of these project 2
  // hum bar bar overwrite karrahe hai intervalId ko 1 same-saves
  // so need to write cleaner code by falsh out the intervalId
  intervalId = null;
};

document.querySelector("#start").addEventListener("click", startChangingColor);

document.querySelector("#stop").addEventListener("click", stopChangingColor);


```

project 5 solution

```javascript
const insert = document.getElementById('insert');

window.addEventListener('keydown', (e) => { // event ata hai wo hai e or e mai bahut sare values hoti hai
  insert.innerHTML = `
    <div class='color'>
    <table>
    <tr>
      <th>Key</th>
      <th>Keycode</th>
      <th>Code</th>
    </tr>
    <tr>
      <td>${e.key === ' ' ? 'Space' : e.key}</td>
      <td>${e.keyCode}</td>
      <td>${e.code}</td>
    </tr>

  </table>
    </div>
  `;
});


```
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>html Events</title>
</head>
<body style="background-color: #414141; color: aliceblue">
    <h2>Amazing image</h2>
    <div >
        <ul id="images">
            <li><img width="200px" id="photoshop" src="https://images.pexels.com/photos/3561339/pexels-photo-3561339.jpeg?auto=compress&cs=tinysrgb&w=1600&lazy=load" alt="photoshop"></li>
            <li><img width="200px" id="japan" src="https://images.pexels.com/photos/3532553/pexels-photo-3532553.jpeg?auto=compress&cs=tinysrgb&w=1600&lazy=load" alt=""></li>
            <li><img width="200px" id="river" src="https://images.pexels.com/photos/3532551/pexels-photo-3532551.jpeg?auto=compress&cs=tinysrgb&w=1600&lazy=load" alt=""></li>
            <li><img width="200px" id="owl" src="https://images.pexels.com/photos/3532552/pexels-photo-3532552.jpeg?auto=compress&cs=tinysrgb&w=1600&lazy=load" alt=""  ></li>
            <!-- 1️⃣ onclick="alert('owl')"  wrong aproach for alert it gives problem, big scalable code -->
            <li><img width="200px" id="prayer" src="https://images.pexels.com/photos/2522671/pexels-photo-2522671.jpeg?auto=compress&cs=tinysrgb&w=1600&lazy=load" alt=""></li>
            <li><a style="color: aliceblue;" href="https://google.com" id="google">Google</a></li>
        </ul>
   </div>
</body>
<script>// javscript is a sequential language

   // document.getElementById('owl').onclick = function(){
   //      alert("owl clicked") // 2️⃣ their can be problems also here
   //   }


   // attachEvent 3️⃣ // jQuery - on


// to learn this events
    // type, timestamp, defaultPrevented
    // target, toElement, srcElement🔄, currentTarget,
    //💻 clientX, clientY, screenX, screenY
    // altkey, ctrlKey, shiftKey, keyCode



   document.getElementById('owl').addEventListener('click', function(){
//        alert("owl clicked again")
   }) //4️⃣★ addEventListener si very powerful || third parameter is false but is by-default false('click', function(){}, false)

   document.getElementById('owl').addEventListener('click', function(e){
//        console.log(e);
}, false)


   document.getElementById('images').addEventListener('click', function(e){
        console.log("clicked inside ul");
}, false)  // capturing event when true && Bubbleing event when false depens on use case
   document.getElementById('owl').addEventListener('click', function(e){
        console.log("owl clicked");
        e.stopPropagation() // stopPropogation see event bullele hoke upar ke element mai nahi jayega
}, false)  // capturing event when true && Bubbleing event when false depens on use case


   document.getElementById('google').addEventListener('click',function(e){
      e.preventDefault() // google ko track krkre uska behaviour change krte hai
      e.stopPropagation()
      console.log("google clicked");
   }, false)

</script>
</html>
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>html Events</title>
</head>
<body style="background-color: #414141; color: aliceblue">
    <h2>Amazing image</h2>
    <div >
        <ul id="images">
            <li><img width="200px" id="photoshop" src="https://images.pexels.com/photos/3561339/pexels-photo-3561339.jpeg?auto=compress&cs=tinysrgb&w=1600&lazy=load" alt="photoshop"></li>
            <li><img width="200px" id="japan" src="https://images.pexels.com/photos/3532553/pexels-photo-3532553.jpeg?auto=compress&cs=tinysrgb&w=1600&lazy=load" alt=""></li>
            <li><img width="200px" id="river" src="https://images.pexels.com/photos/3532551/pexels-photo-3532551.jpeg?auto=compress&cs=tinysrgb&w=1600&lazy=load" alt=""></li>
            <li><img width="200px" id="owl" src="https://images.pexels.com/photos/3532552/pexels-photo-3532552.jpeg?auto=compress&cs=tinysrgb&w=1600&lazy=load" alt=""  ></li>
            <!-- 1️⃣ onclick="alert('owl')"  wrong aproach for alert it gives problem, big scalable code -->
            <li><img width="200px" id="prayer" src="https://images.pexels.com/photos/2522671/pexels-photo-2522671.jpeg?auto=compress&cs=tinysrgb&w=1600&lazy=load" alt=""></li>
            <li><a style="color: aliceblue;" href="https://google.com" id="google">Google</a></li>
        </ul>
   </div>
</body>
<script>
    // task:- if we clicked a image it must be removed
    // event spillover means when  cliked on list iteams all the reaming items got removed
    document.querySelector('#images').addEventListener('click',function(e){
        console.log(e.target.tagName);
        if (e.target.tagName === 'IMG'){
            console.log(e.target.id);
            let removeIt = e.target.parentNode
            removeIt.remove()
        }



    }, false)



        // removeIt.parentNode.removeChild(removeIt)

</script>
</html>
```

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta
      name="viewport"
      content="width=
    , initial-scale=1.0"
    />
    <title>Document</title>
  </head>
  <body>
    <h1>Chai aur JavaScript</h1>
    <button id="start">start</button>
    <button id="stop">Stop</button>
  </body>
  <script>
    let intervalID;

    const sayDate = function () {
      console.log("hi1", Date.now());
      //    console.log(str, Date.now()); // third parmeter is not working like this in my tests
    };
    document.getElementById("start").addEventListener("click", function (){
      intervalID = setInterval(sayDate, 1000);
    });

    // setInterval(sayDate, 1000, "hi") // third parmeter is not working like this in my tests

    document.getElementById("stop").addEventListener("click", function () {
      clearTimeout(intervalID);
      console.log("STOPPED!");
    });

    // clearInterval(intervalID); // memory garbage collection hogaya
  </script>
</html>

```

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta
      name="viewport"
      content="width=
    , initial-scale=1.0"
    />
    <title>Document</title>
  </head>
  <body>
    <h1>Chai aur code</h1>
    <button id="stop">Stop</button>
  </body>
  <script>
    //   setTimeout(function(){
    //       console.log("Abhay");
    //    }, 2000)
    const sayAbhay = function () {
      console.log("Abhay");
    };
    const changeText = function () {
      document.querySelector("h1").innerHTML = "Love u all";
    };

    const changeMe = setTimeout(changeText, 2000);

    document.querySelector('#stop').addEventListener('click', function() {
      clearTimeout(changeMe);
      console.log("STOPPED!");
  });

  </script>
</html>

```

```
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <style>
        .card {
            background-color: #333;
            color: white;
            padding: 20px;
            margin: 20px;
            border-radius: 10px;
            text-align: center;
          }
          .card img {
            border-radius: 50%;
            width: 100px;
            height: 100px;
          }
    </style>
  </head>
  <body style="background-color: #212121">

    <div id="card-container"></div>

  </body>
  <script>
    const requestUrl = "https://api.github.com/users/hiteshchoudhary";
    const xhr = new XMLHttpRequest();
    xhr.open("GET", requestUrl);
    // next open call
    xhr.onreadystatechange = function () {
      console.log(xhr.readyState);
      if (xhr.readyState === 4) {
        const data = JSON.parse (this.responseText);
        console.log(typeof data);
        console.log(data.followers);
        console.log(data.avatar_url);
        console.log(xhr.responseText);

        const cardContainer = document.getElementById("card-container");
        const card = document.createElement("div");
        card.classList.add("card");
        card.innerHTML = `
          <img src="${data.avatar_url}" alt="avatar" />
          <h1>${data.name}</h1>
          <p>${data.bio}</p>
          <p>${data.location}</p>
          <p>${data.followers} followers</p>
          <p>${data.following} following</p>
        `;
        cardContainer.appendChild(card);
      }
    };
    console.log("HERE");
    xhr.send();
  </script>
</html>
```

```js
// most of the time we consume the promises only

// we need to know how promises are created

// when there were no promises, then async await was used for Database operations
// then promises were introduced it had a good syntax
// libraries:- Q and Bluebird(fav of hitesh) were used to access all functionalities of promises
// such as:- .fetch, .then, catch
// eventually this libraries were appritiated and now integrated in the javascript itself
// but it is not necessary to use

// promises अपने अंदर call-back function लेता है।
// promise को variable में hold kiya hai
const promiseOne = new Promise(function (resolve, reject) {
  // Do an async task
  // Database calls,cryptography, network
  setTimeout(function () {
    console.log("async task completed");
    /**method**/
    resolve();
  }, 1000);
});

promiseOne.then(function () {
  console.log("Promise consumed");
}); // resolve has connection with .then()   is used to consume the promise
// .then() has a callback function which automatically gets arguments for value returned by resolve

/* with-out holding in a variable */
new Promise(function (resolve, reject) {
  setTimeout(function () {
    console.log("async task 2");
    resolve();
  }, 1000);
}).then(function () {
  console.log("Promise 2 consumed");
});

/* how to pass data that came from network */
const promiseThree = new Promise(function (resolve, reject) {
  setTimeout(function () {
    console.log("async task 3");
    resolve({ username: "chai", email: "koi@toodedo.com" }); // most of the time the data passed ther is present in object form
  }, 1000);
});

promiseThree.then(function (user) {
  console.log(user);
});

/*promiseFour*/
const promiseFour = new Promise(function (resolve, reject) {
  setTimeout(function () {
    let error = true; // file_access/web_request/net_request krri to huuii nahi huiiee tabhi kuch tobolna padega
    // acctually mai resolve hoga ya reject hoga
    if (!error /*error_nahi_hai*/) {
      resolve({ username: "hitesh", password: "1234" });
    } else {
      reject("ERROR: something went wrong");
    }
    console.log();
    resolve();
  }, 1000);
});

// thier not only one dot_then() or dot_catch() we can use as many as dot_then()
// for avoiding callback hell
promiseFour
  .then((user) => {
    console.log(user);
    return user.username;
  })
  .then((username) => {
    // this is chaining jo value return huii hai usi value ko next then() me pass krega
    // usefuel at the time of database coonection
    console.log(username);
  })
  .catch((error) => {
    // error will be catched here
    console.log(error);
  })
  .finally(() => console.log("The promise is either resolved or rejected"));

/*Promise 5*/
const promiseFive = new Promise(function (resolve, reject) {
  setTimeout(function () {
    let error = false;
    if (!error) {
      resolve({ username: "javascript", password: " 2154" });
    } else {
      reject("ERROR: JS went wrong");
    }
  }, 1000);
});

// we cam handle promise with async and await!
// want to learn how to debugg to check errors
async function consumepromisefive() {
  try {
    const response = await promiseFive; // promise:- eventual complition object
    // promise is a object we can't consume it like promiseFive()
  } catch (error) {
    console.log(response);
  }
}

consumepromisefive();

/*
async function getAllUsers() {
  try {
    const response = await fetch('https://jsonplaceholder.typicode.com/users')
    // console.log(response); it is printing

    const data = await response.json()
    console.log(data);

  } catch (error) {
    console.log("E:",error);
}}

getAllUsers()
*/

/**now doing this in .then() and .catch() format**/

fetch("https://api.github.com/users/hiteshchoudhary")
  .then((response) => {
    return response.json();
  })
  .then((data) =>{
    console.log(data);
  })
  .catch((error) => console.log(error));

```

```
# notes

## XML Http Request
```javascript
https://api.github.com/users/hiteshchoudhary // https://randomuser.me/api/
https://jsonformatter.org/ helps for reading api // fetch नया आया है 2015


XMLHttpRequest was used before 2015 it is a heavily AJAX programming


https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest
https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/readyState


/* **********Console.log********** */
is console part of javascript
technically yes and technically not
console and many api's such as document is not a part of core javascript
basic arithmetic operations, functions exections and some loops are there in care JS

console.log is a [dev/debugger tool] the run time inject console.log
https://github.com/v8
then go in d8/console.cc and .h is called header

JS runs on C++ language
Python runs on C language
core language is this only but it becomes like a rapper on it

/* **********promise********** */

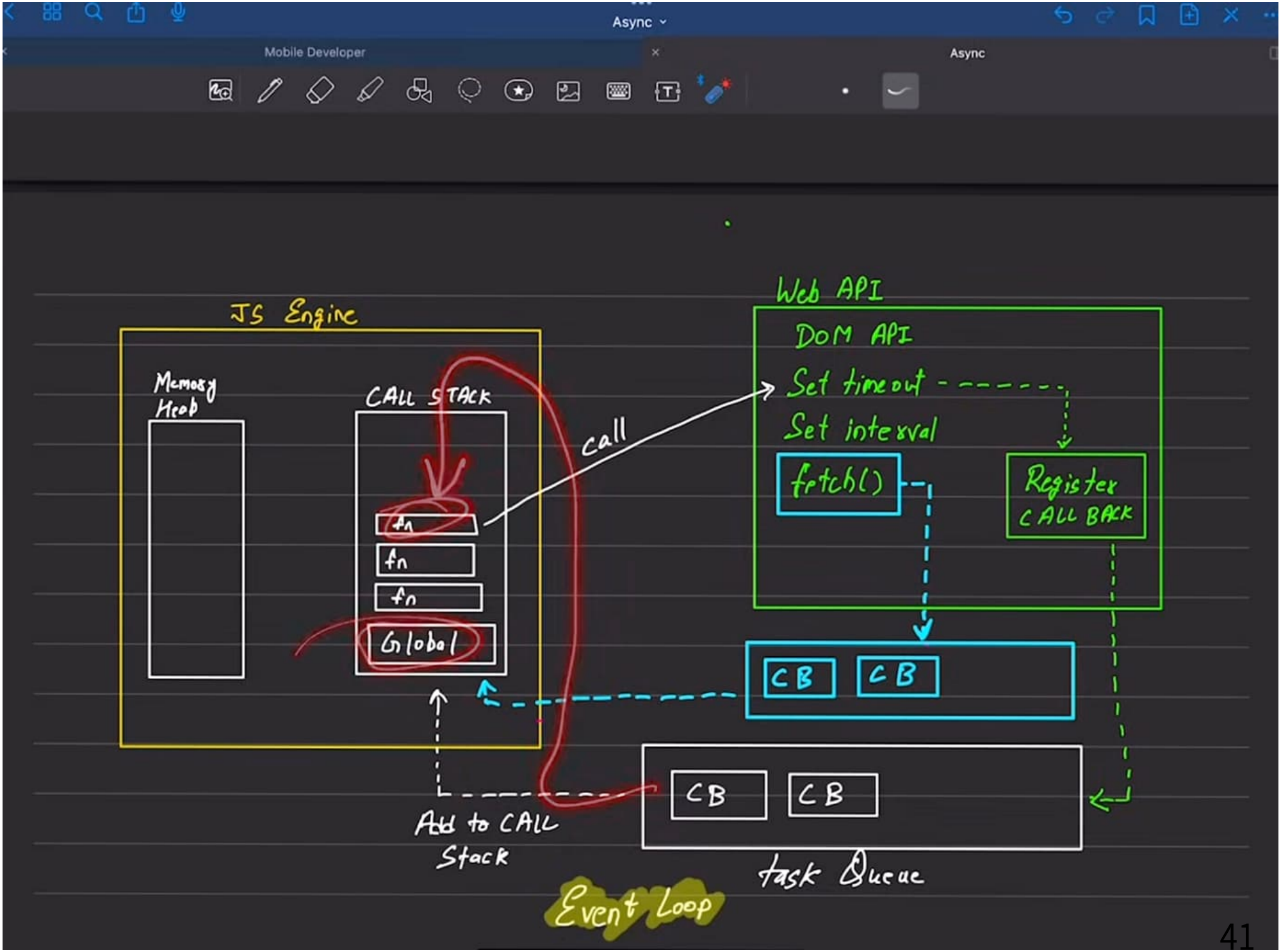
promise completes in future in operations such as
cryptograpy operation,
network,
file system,
in mobile devleopment
to activate sound devices, to activate camera devices

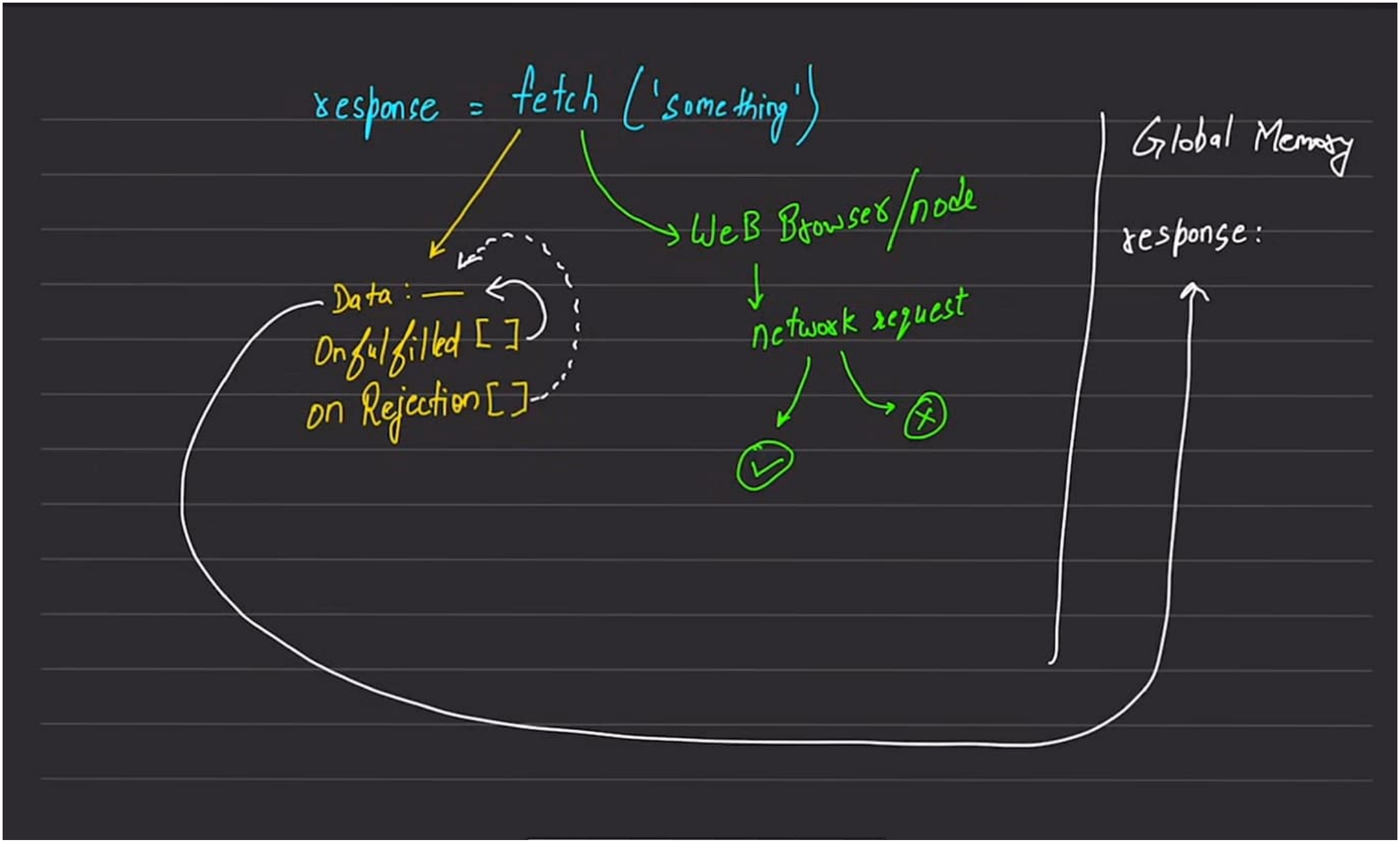
```
[V8](https://github.com/v8)

[reponse](https://jsonplaceholder.typicode.com/users)

[fetch information](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API)
[fetch specification](https://fetch.spec.whatwg.org/#fetch-method)
![there is a special queue(micro task queue/priority queue)](Screenshot_2024-12-26-14-04-26-98_d11ae654701a5fadb313ea6fa9a6836f.jpg)
![alt text](Screenshot_2024-12-26-14-10-27-22_d11ae654701a5fadb313ea6fa9a6836f.jpg)
404 error is a onfulsield[] or resolve
[For re-understanding](https://youtu.be/Rive84an6Lc)

```


Async
Mobile Developer
Async
JS Engine
Memory Heap
CALL STACK
fn
fn
fn
Global
call
Web API
DOM API
Set timeout
Set interval
fetch()
Register CALL BACK
CB
CB
CB
CB
Add to CALL Stack
task Queue
Event Loop


response = fetch ('something')
Global Memory
response:
Data: —
Onfulfilled [ ]
on Rejection [ ]
Web Browser/node
network request

```js
const user = {
  username: "abhay",
  loginCount: "8",
  signedIn: true,

  getUserDetails: function () {
    // console.log("got user details from database");
    // console.log(`username: ${this.username}`);
    console.log(this);
  },
};

console.log(user.username);
// console.log(user.getUserDetails());
// console.log(this); // in global scope

// const promiseOne = new/*constructor function*/ Promise()
// const date = new/*constructor function*/ Date()
//  this constructor function allow,s एक ही object literal से आप  multiple instance बना सको
// new is used for making new context

function User(username, loginCount, isLoggedIn) {
  this.username = username;
    this.loginCount = loginCount;
    this.isLoggedIn = isLoggedIn

    this.greetings = function () {
        console.log(`Welcome ${this.username}`);

    }

    return this // write verbos code (implicitly return define hota hai)
}

const userOne = User("hitesh", 12, true)
const userTwo = User("bumrah", 11, false) // values overright kar di

console.log(userOne.constructor);
// console.log(userTwo);


// instanceof on mdn : don,t depend on spoonfeeding
```

```js
function multiplyBy5(num) {
  return num * 5;
}

multiplyBy5.power = 2;

console.log(multiplyBy5(5)); // function function भी है
console.log(multiplyBy5.power);
console.log(multiplyBy5.prototype); // function object भी है
// prototype कि भी properties होती है

/*****************************************************************************/

function createUser(username, score) {
  this.username = username;
  this.score = score;
}

createUser.prototype.increament = function () {
  this.score++; // (जिस का मतलब This) जिसने ने भी बुलाया है उसके पास जाओ |
};

createUser.prototype.printMe = function () {
  // this is call [method/propeties] that is created
  console.log(`score is ${this.score}`);
};
const chai = new createUser("chai", 25); // जो properties inject कि है ये बताने का काम new keyword karta hai
const tea = createUser("tea", 250);

chai.printMe();

// what is hd/sd 2 and hd 3


/*

Here's what happens behind the scenes when the new keyword is used:

A new object is created: The new keyword initiates the creation of a new JavaScript object.

A prototype is linked: The newly created object gets linked to the prototype property of the constructor function. This means that it has access to properties and methods defined on the constructor's prototype.

The constructor is called: The constructor function is called with the specified arguments and this is bound to the newly created object. If no explicit return value is specified from the constructor, JavaScript assumes this, the newly created object, to be the intended return value.

The new object is returned: After the constructor function has been called, if it doesn't return a non-primitive value (object, array, function, etc.), the newly created object is returned.

*/
```

```
// let my= "abhay     "
// let myluck = "zatu  ";


// console.log(myName.truelength);


let myHeros = ["thor", "spiderman"]

let heroPower = {
    thor : "hammer",
    spiderman : "sling",

    getspiderPower: function () {
        console.log(`Spidy power is ${this.spiderman}`);

    }
}

Object.prototype.abhay = function(){
    console.log(`abhay is present in all oblects`);

}

Array.prototype.heyAbhay = function () {
    console.log(`abhay says helllo`);

}


// heroPower.abhay()
myHeros.abhay()
myHeros.heyAbhay()
// heroPower.heyabhay()

// ++++++++++++++++ INHERITANCE ++++++++++++++++++

const User = {
    name: "chai",
    email: "chai@google.com"
}

const Teacher = {
    makeVideo: true
}

const TeachingSupport = {
    isAvailable : false
}

const TASupport = {
    makeAssingment: 'JS assignment',
    fullTime: true,
    __proto__: TeachingSupport // taking refrence or borowing properties by sing this syntax/keyword
}

/* we can take acess from out side also */
Teacher.__proto__ = User
// teacher bhi user ki sari properties access krr sakta hai
// kisi aur ki properties ko kaise access krr sakte hai yahi prototypal inheritance hai

/* modern syntax */
Object.setPrototypeOf(TeachingSupport, Teacher)

let anotherUsername = "CanISmellUrPantiess      "

String.prototype.trueLength = function(){
    console.log(`${this}`);
    console.log(`True length is: ${this.trim().length}`);
}

anotherUsername.trueLength()
```

```
function SetUsername(username) {
  // complex DB calls
  this.username = username;
  console.log("called"); /* 1️⃣ */

}

function createUser(username, email, password) {
//   SetUsername(username); /* 1️⃣ hame लग्राहा hai ki ye call किया  है  🍑 javascript ने sirf refrence दिया है  */

//   SetUsername.call(username); /* 2️⃣ अब ये technically call हो रहा है bcoz reference hold करके रख रहा है  */

(this.email = email),
SetUsername.call(this, username); /* 3️⃣ मेरा वाला this नहीं आपका वाला this use करूँगा , context pass krni ki kahai this ki*/
  (this.password = password);
}

const chai = new createUser("chai", "chai@fb.com", "123");
console.log(chai);

```

```
// the javascript we are useing is after ES6 और ये Syntactic sugar ही है अभी भी ।

// Syntactic sugar refers to certain language features in programming that make the code easier to read or write


// class User { // class अब एक key-word है JS में
//     constructor(username, email, password){
//         /* for context =>*/ this.username = username ;
//         this.email = email;
//         this.password = password
//     }

//     // password CLEAR_TEXT_FORMAT में तो नहीं रखोगे means unencrypted

//     encryptPassword(){ // है तो ये भी function ही पर class के अंदर है तो method बोलने लग गये
//         return /* variable ले लेते है ${}*/`${this.password}abc`

//     } /* और भी method add केआर देते है इसमें  */
//     changeUsername(){
//         return `${this.username.toUpperCase()}`
//     }
// }

// //  इससे अब एक user बना लेते है chai ↓
// const chai = new User("chai", "chai@gmail.com","123")

// console.log(chai.encryptPassword());
// console.log(chai.changeUsername());

/* do un-commented and commented for understanding */


// BEHIND_THE_SEEN

function User(username, email, password){
    this.username = username ;
    this.email = email;
    this.password = password
}

User.prototype.encryptPassword = function(){
    return `${this.password}abc`
}
User.prototype.changeUsername = function(){
    return `${this.username.toUpperCase()}`
}

const tea = new User("tea", "tea@gmail.com", "123")

console.log(tea.encryptPassword());
console.log(tea.changeUsername());

// this is about class constructor
```

```
class User {
    constructor(username){
        this.username = username
    }
    /* एक method भी है इसमें logMe */
    // logMe क्या करता है username दिया है यूएस username की value set करके देता है
    // variable inject kardiya और class ke pass current context ka access hai kui ki this. puri class mai sabb jagah available hai hi
    logMe(){
        console.log(`USERNAME IS ${this.username}`);

    }
}// ये तो होगया Basic class बनाना


// लेकिन आप इस user को lms/owneer बनाना पड़ेगा कभी teacher कभी admin कभी student

/* वहाँ पे protype hua था अब extends, sugar लगा दिया गया है */
class Teacher extends User{
    constructor(username, email, password){
        /* username के लिए call करना पड़ता था 🍑 अब class का syntax है */
        /* तो फिर ये super keyword reffer करेगा की भाईसाहब कौनसी class extend कर रहे थे => user class के अंदर इस constructer में जाता हू this automatically BTS लेजाऊँगा उस username की value set करदूँगा, उस username की value यहाँ पर होजाएगी, और आप उसका acces directly भी ले पाओगे */
        super(username)
        this.email = email
        this.password = password
    }
    addCourse (){
        console.log(`A new course was added by ${this.username}`);

    }
}

const chai = new Teacher("chai", "chai@gmail.com", "123")
// chai.addCourse()
chai.logMe()

const masalaChai = new User("masalaChai")
masalaChai.logMe()

console.log(chai instanceof Teacher);

```

```
class User{
    constructor(username){
        this.username = username
    }
    logMe(){
        console.log(`Username: ${this.username}`);

    }
    static createId(){ // कई बार ऐस बोहत सारी situtation होगी इस method का acces हर उस object को नहीं देना चाहते जो इस class से instanciate हुआ है । उसके लिए सिर्फ़ static लगदो key-word के सामने
        return '123'
    }
}

const hitesh = new User("hitesh")
// console.log(hitesh.createId());

class Teacher extends User {
    constructor(email, username /* order matter nhi krta*/){
        super(username)
        this.email = email
    }
}

const iphone = new User("iphone", "i@phone.com")
// iphone.logMe()
console.log(iphone.createId())

```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>React</title>
</head>
<body>
    <button>Button Clicked</button>
</body>
<script>
    class React{
        constructor(){
            this.library = React
            this.server = "http://localhost:300"

            // requiremrnt
            // document made by c++
            document
                .querySelector('button')
                .addEventListener('click', this.handleClick.bind(this))
                // bind मतलब जोड़ना
        }
        handleClick (){
            console.log("button clicked");
            console.log(this);


        }
    }

    const app = new React()
</script>
</html>
```

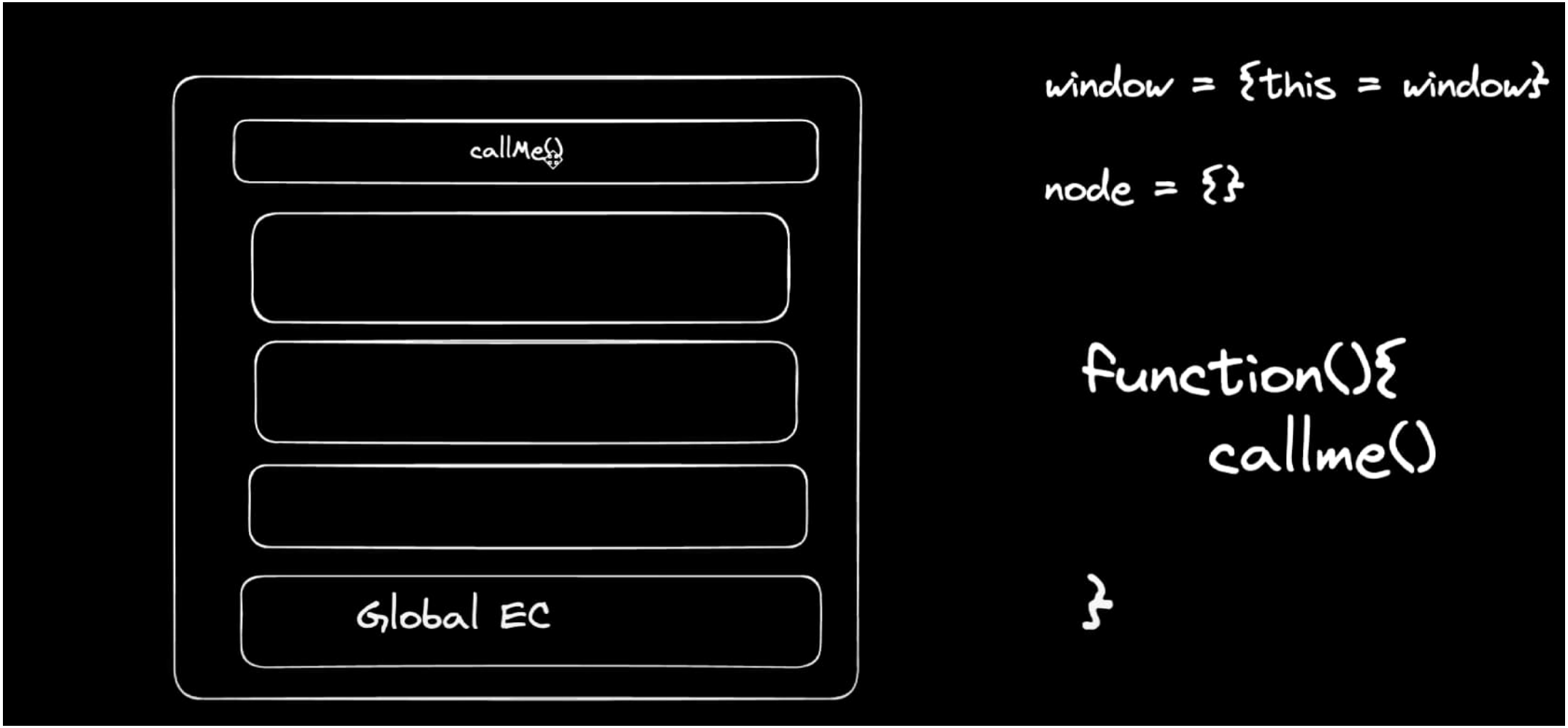
```
/*
 object prototype की properties inherit है और object की खुद की भी properties है
जैसे की constructurs new_keyword class सब यही से चाल रहा है */

const discriptor = Object.getOwnPropertyDescriptor(Math, "PI");
/*
  PI जो है property का name है और 3.14 उसकी value है || PI की value single quote double quote किसी में भी लिख सकते है */

// console.log(discriptor); // check this

/*
console.log(Math.PI);
Math.PI = 5
console.log(Math.PI); // value change नहीं हो रही है 🍑 किऊ नहीं हो रही
 */

const chai = {
  name: "jinger",
  price: 243,
  isAvailable: true,

  orderChai: function () {
    console.log(`chai nhi bani /* code faat gaya */`);
  },
};
console.log(Object.getOwnPropertyDescriptor(chai, "name"));

Object.defineProperty(chai, 'name',{
    // writable: false,
    enumerable: false,
    // configurable: true
})

console.log(Object.getOwnPropertyDescriptor(chai, "name"));

for (let [key, value] of Object.entries(chai)) { // forof loop objects के लिए easy रहता है
  if (typeof value !== 'function') { //for handling code

      console.log(`${key} : ${value}`);
}
}

```

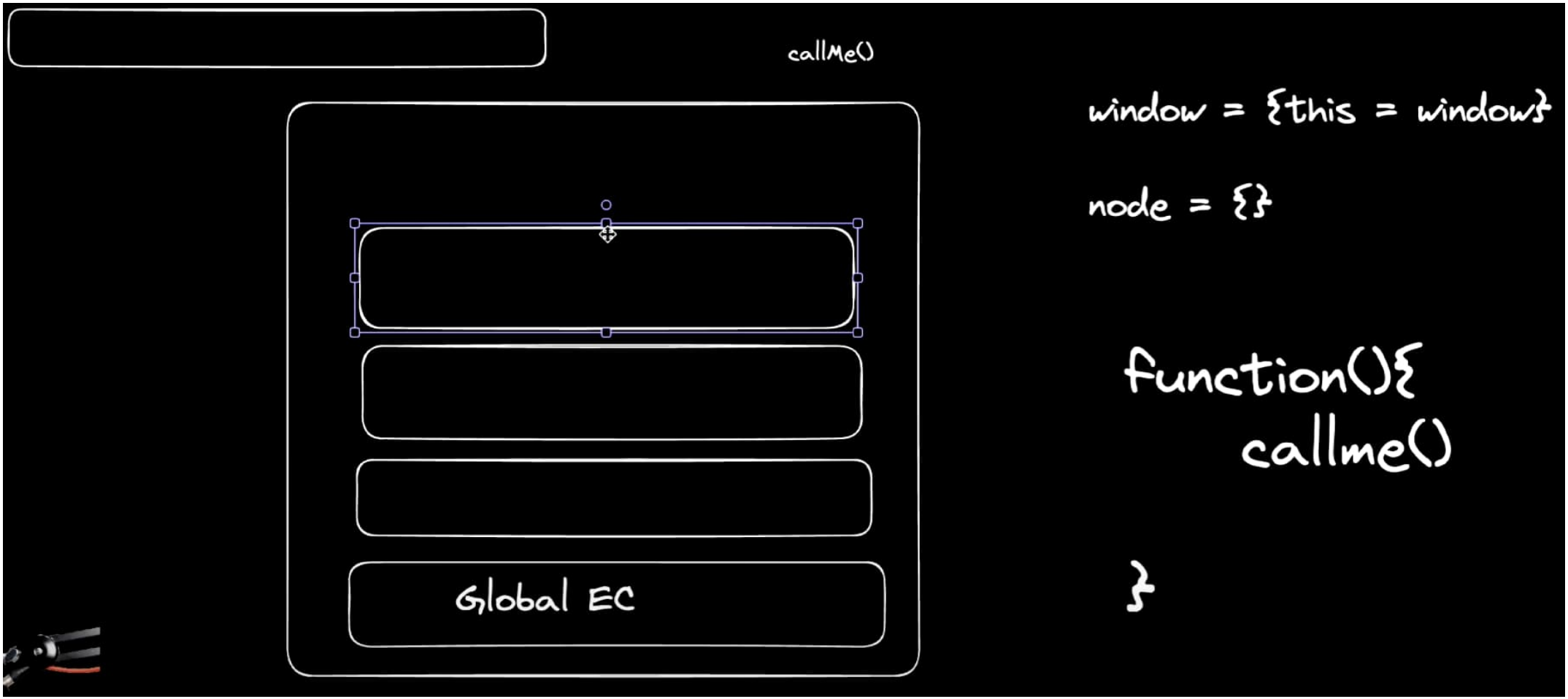
```
// 90 % syntax is class based syntax
class User {
  // under the hood protoype ही चल रहा है.ये सब rapper hai
  constructor(email, password) {
    this.email = email;
    this.password = password
  }
  get email(){
    return this._email.toUpperCase()
  }

  set email(value){
    return this._email = value
    /*  get और set में race लग जाती है
    > उसका soln is this._properties  */
  }

  get password() {
    // जितने भी properties बनाते हो उससे getters और setters method बन्न जाते है
    // return this._password.toUpperCase();
    return `${this._password}proCoder`
  }

  set password(value) {
    /* getter define किया है तो setter भी define करना पड़ेगा
      > getter और setters by defaut हर code में होते है
      > hard coding, fixed values or data are written directly into the source code of a program.
      > whereas in soft coding, data or configuration information is stored in a separate location, such as a configuration file or database, and read into the program at runtime.
        */
    this._password = value.toUpperCase();

  }
}

const hitesh = new User("hi@hitesh.ai", "worldsGreatest");

console.log(hitesh.password);
console.log(hitesh.email);

/* ⭐ कई बार हमे fine grain control चाहिए होता
>   है की password नहीं बताऊंगा
>   की या फिर encrypted password return दु

>   jo properties ka access सबको नहीं देना चाहते
>   या फिर कोई लेना चाहता है acess to fir उसमे customize code kr सकते है .
>   in sab senarios ke liye geeters or setters istamal kiye jata hai
 */

```

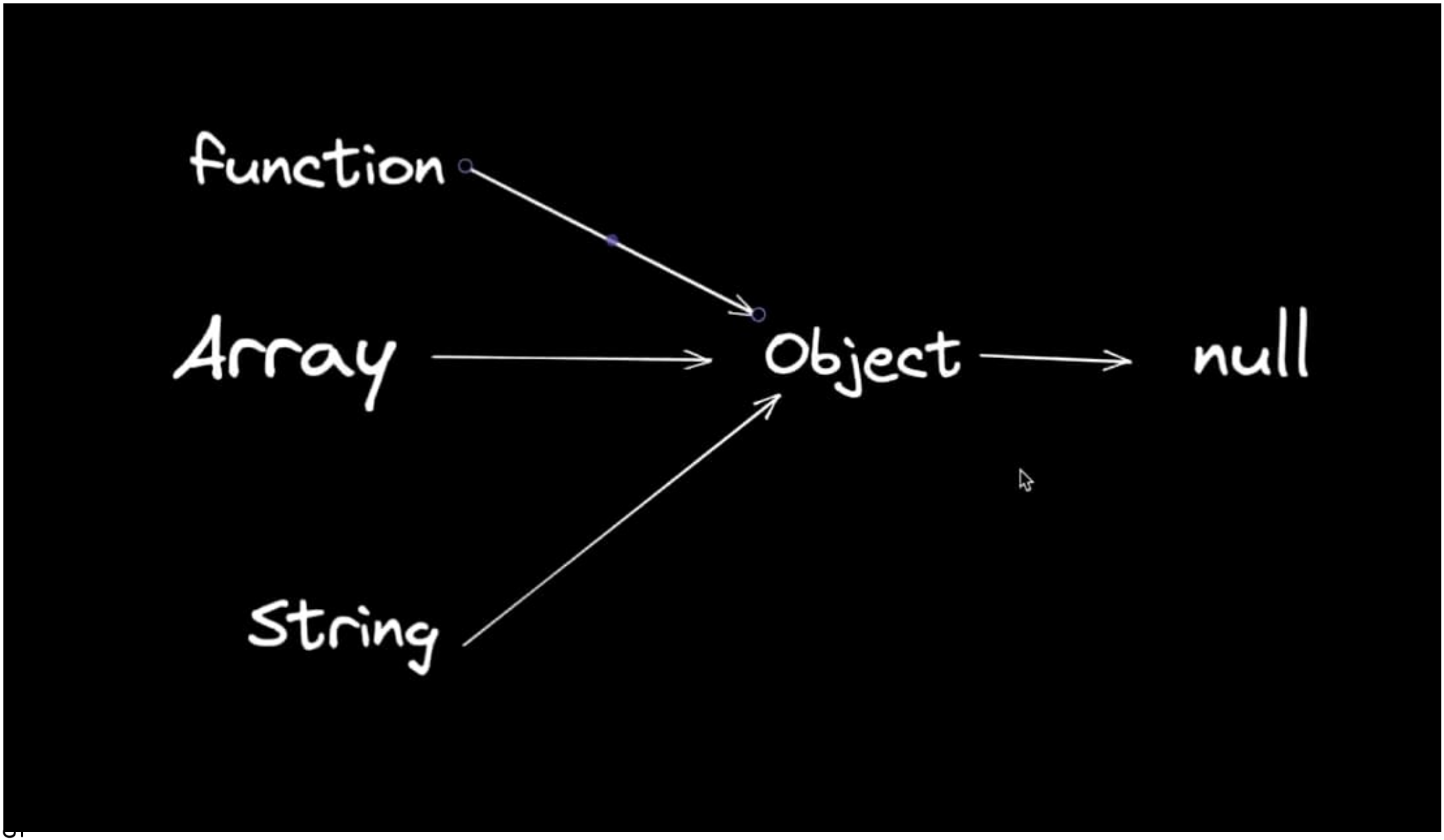
```
// पहले तो classes नहीं होती थी तो ऐसे होता था

function User(email, password){
    this._email = email;
    this._password = password;

    /* function अपने ऐप में object भी है और function भी है
        तो object की properties को भी call कर सकते हो
        parameters (इस method का this नहीं hota है, कौनसी property overwrite करना चाहते हो, {Object इसके अंदर जो भी properties define करना चाहते हो krr सकते ho} )   */
    Object.defineProperty(this, 'email', {
        get: function(){
            return this._email.toUpperCase()
        },
        set: function(value){
            this._email = value
        }
    })
    Object.defineProperty(this, 'password', {
        get: function(){
            return this._password.toUpperCase()
        },
        set: function(value){
            this._password = value
        }
    })
}

const chai = new User("chai@gmail.com", "chai")

console.log(chai.email);

```

```
// ये वाला syntax ज्यादा use नहीं होता है rarely देखोगे इस syntax ko
// underscore kya dikhata hai ki mai almost ek private property define krr raha hu
// ya fir ye normal users ke use mai nahi aa rahi
// get(le ke ane kaa aur ched chaad krna[overwrite kardiya hai]) aur set use krne se itna meaning nahi reh jata underscore ka
// getters aur setters method nhi rehta ye ek special method ban jata hata hai jo mai propertie ke upar rak raha hu

const User = {
  _email: "h@hc.com",
  _password: "abc",

  get email() {
    return this._email.toUpperCase();
  },
  set email(value) {
    this._email = value;
  },
};

/* here we can use factory function > object.create ये array में भी होते है
> new is a constructor function
> User ke base pe ek object create kroo aur usko t ke andar refer krr do */
const tea = Object.create(User);
console.log(tea._email);

```

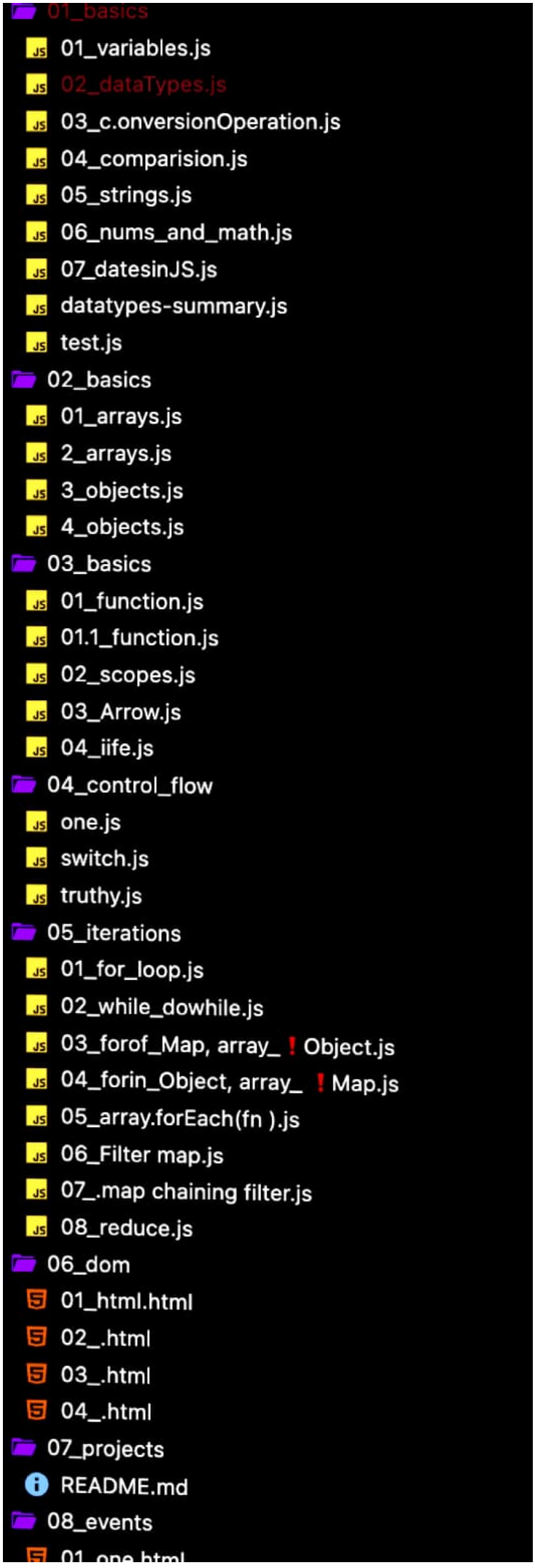
```
# javascript and classes

## OOP (programming pardaime)

## Object

- collection of properties and methods
- toLowerCase

## why use OOP

coz of mess upcode (spegatie)
java good feature such as services and get instances

## parts of OOP

Object literal {}

- Constructor function
- Classes
- Instances (new, this)

## 4 pillars

Abstraction (hide details eg:-javascript fetch)
Encapsulation (rapper on data)
Interitance (  )
polymorphisom ( many variations)

## javascript protypial behaviour( searches parents, grand-parents, goes till root threfore= array(0) )
<span style="color:red;">JS haaar nahi manta, yaha nhi mila aurr upar yaha bhi nah mila to aurr upar</span>

const newHero = ["hulk", "spiderman"] // in console

- gives access of this
- classes
- this keyword working by protypial behaviour
- protypial inheritance comes in javascript is coz of JS protypial behaviour

function technically object ko bhi refrence krta hai

![alt text](<WhatsApp Image 2024-12-30 at 14.54.51.jpeg>)
In the end their are stters and getters also after object

![alt text](<Screenshot 2021-09-29 at 12.00.00 PM.png>)// problem is that it only reads jpg or jpeg not png
```

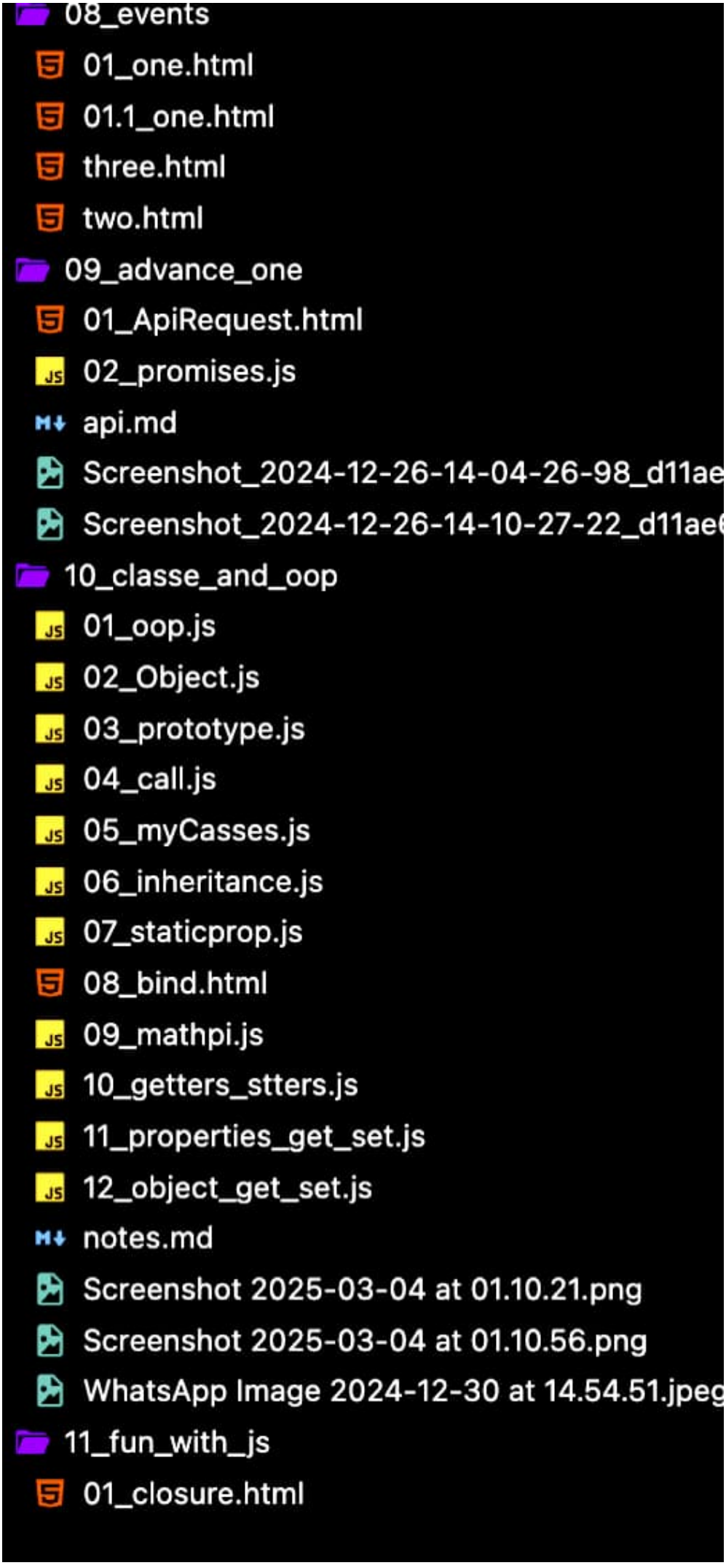

callMe()
Global EC
window = {this = window}
node = {}
function(){
callme()
}


callMe()
Global EC
window = {this = window}
node = {}
function(){
callme()
}


function
Array
Object
null
String

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Closure</title>
</head>
<body style="background-color: #313131;">

</body>

<script>
    function outer (){
        let username = "abhi"
        function inner (){
            console.log("inner ",username);

        }
        inner()
    }
    console.log("TOO OUTTER",username)

</script>
</html>
```


01_basics
01_variables.js
02_dataTypes.js
03_c.onversionOperation.js
04_comparision.js
05_strings.js
06_nums_and_math.js
07_datesinJS.js
datatypes-summary.js
test.js
02_basics
01_arrays.js
2_arrays.js
3_objects.js
4_objects.js
03_basics
01_function.js
01.1_function.js
02_scopes.js
03_Arrow.js
04_iife.js
04_control_flow
one.js
switch.js
truthy.js
05_iterations
01_for_loop.js
02_while_dowhile.js
03_forof_Map, array_ ! Object.js
04_forin_Object, array_ ! Map.js
05_array.forEach(fn).js
06_Filter map.js
07_.map chaining filter.js
08_reduce.js
06_dom
01_html.html
02_.html
03_.html
04_.html
07_projects
README.md
08_events
01_one.html


08_events
01_one.html
01.1_one.html
three.html
two.html
09_advance_one
01_ApiRequest.html
02_promises.js
api.md
Screenshot_2024-12-26-14-04-26-98_d11ae
Screenshot_2024-12-26-14-10-27-22_d11ae
10_classe_and_oop
01_oop.js
02_Object.js
03_prototype.js
04_call.js
05_myCasses.js
06_inheritance.js
07_staticprop.js
08_bind.html
09_mathpi.js
10_getters_stters.js
11_properties_get_set.js
12_object_get_set.js
notes.md
Screenshot 2025-03-04 at 01.10.21.png
Screenshot 2025-03-04 at 01.10.56.png
WhatsApp Image 2024-12-30 at 14.54.51.jpeg
11_fun_with_js
01_closure.html