Chai aur code Javascript\_29/10/2024 :

Go to mdn , w3schools for reading the things when needed

// today let us start the javascript

// to get more confidence on the things we need have the projects

// therefore let us make the projects after the basics are learnt

// this confidence is important to crack the interviews

// googling is jot a bad thing

//

**Lecture 2 : Setting up environment in local machine for Javascript**

// let us learn about the system settings

// we can execute it in browser

// we can just make the file with the .js extension

// why not use the .txt extension

// we need to use the .js for the extension

//

Difference between text , js files:

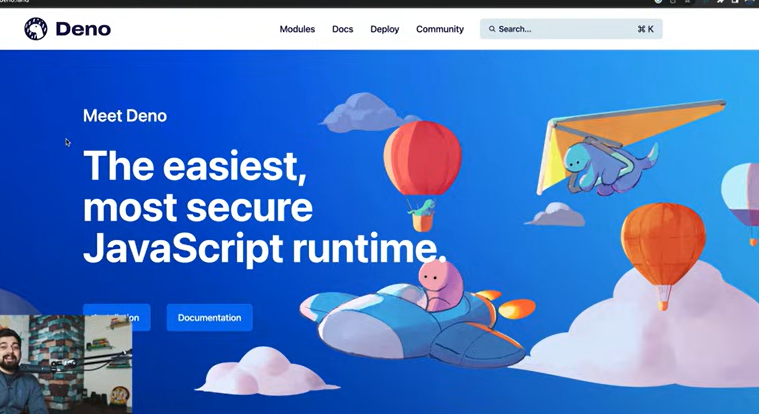
// the softwares are going to have the meaning for all the things , this can be compiler or interpreter

// initially ,we used to execute JS suing a different way

// in recent days the sw from the browser is made separate , we can run this thing differently in a separate file

Similar to executing the c, c++ we have a node.js for executing the JS

There are many other things for the





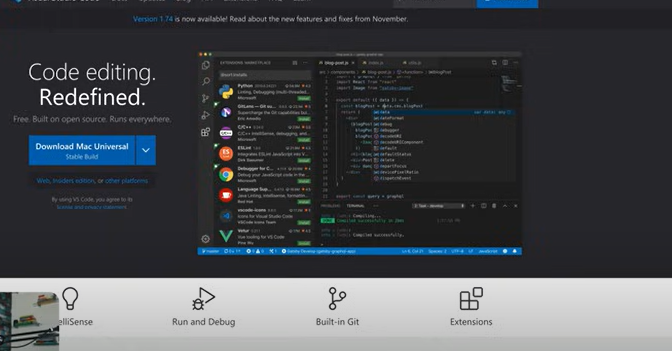
We have these softwares to execute the JS

Just install them and do next next etc

We have already installed the node js

LTS is used for the long term usage

Use vscode for the code writing



—----------------

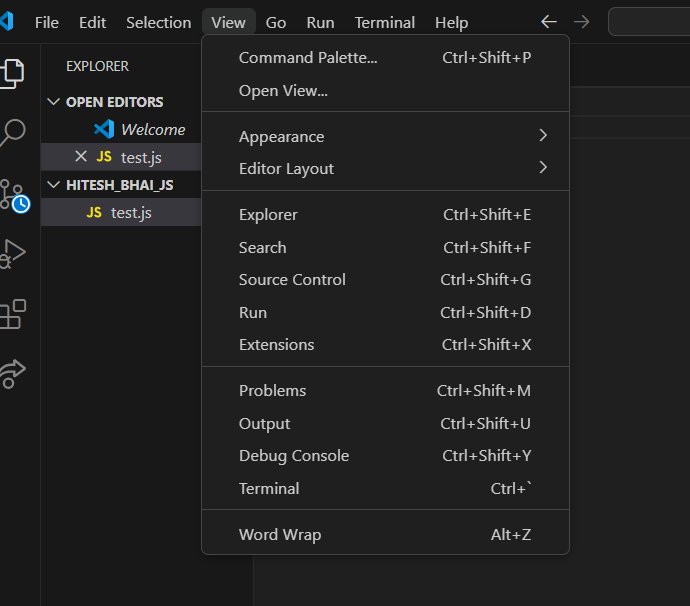
Version checking :

node -v

v20.14.0

—---------------------

Opening terminal :



—-------------------------

# **Lecture 3 : Save and work on Github for Javascript**

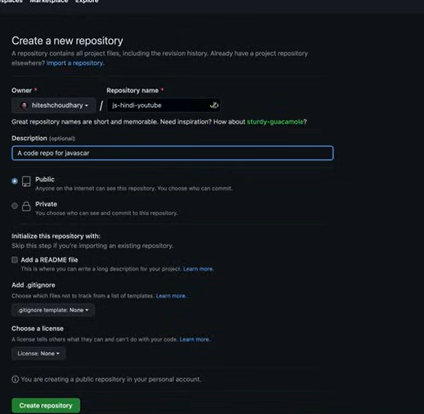
// using git for the command line

// keeping versions

// keeping collaborations

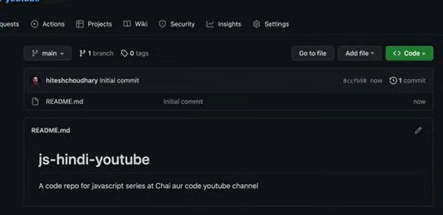
STEPS :

1. Go to github
2. Make new repo
3. Name it
4. Make description

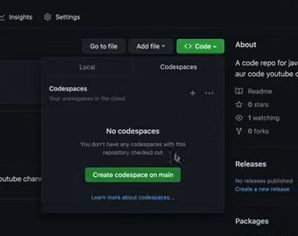


Make a readme file , public

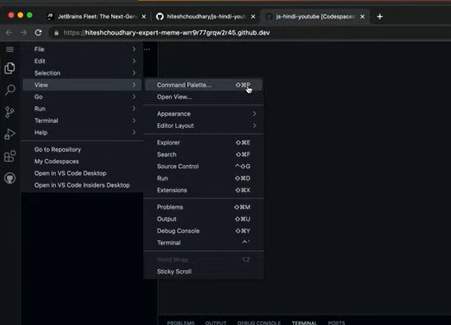
1. Create repository
2. We get this page



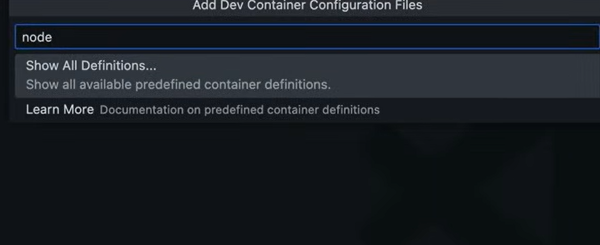
1. We click code



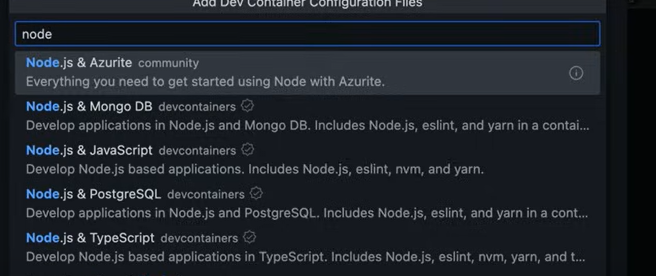
1. Select code space etc
2. Things will be open in vs code
3. In the behind scene we have the vs code , in a virtual machine
4. We need to amke environemnt here
5. Therefore how to do this



1. G to command platte
2. And then click >container



1. Go to show all definitions



Then select node and then select node and js

19. We get a code execution file

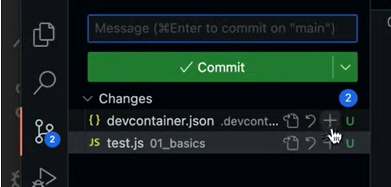
20. Do rebuilt if suggested

21. Click new file ,and folder



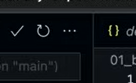
Path for folder and file

**How to save**



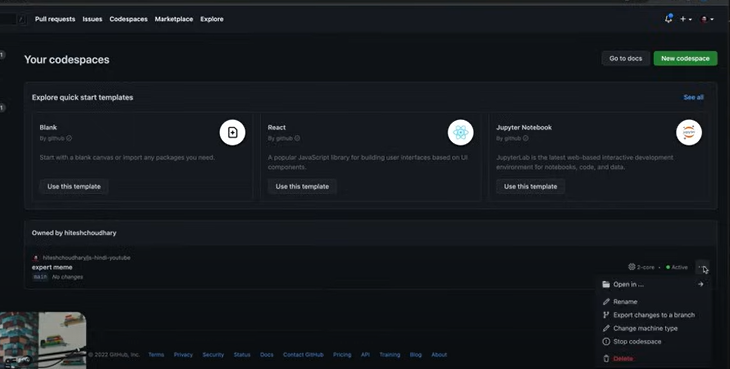
Go to this and it will be tracking things

, then we do commit by doing the description



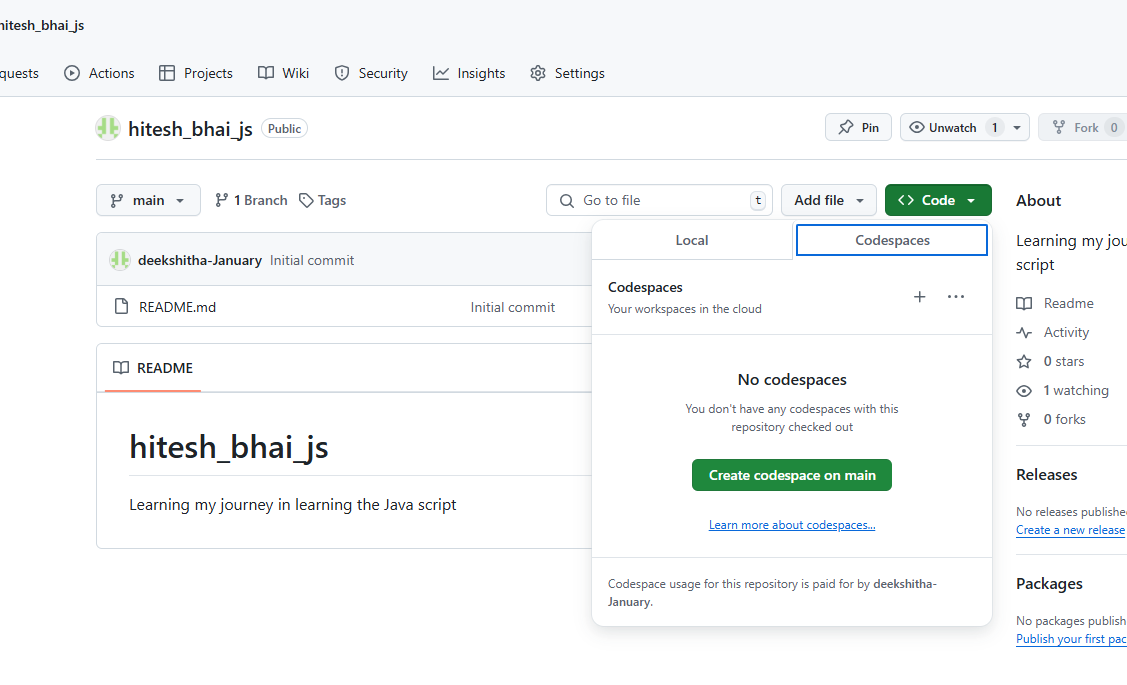
After clicking the … , we get puh options

We need to off the amcching

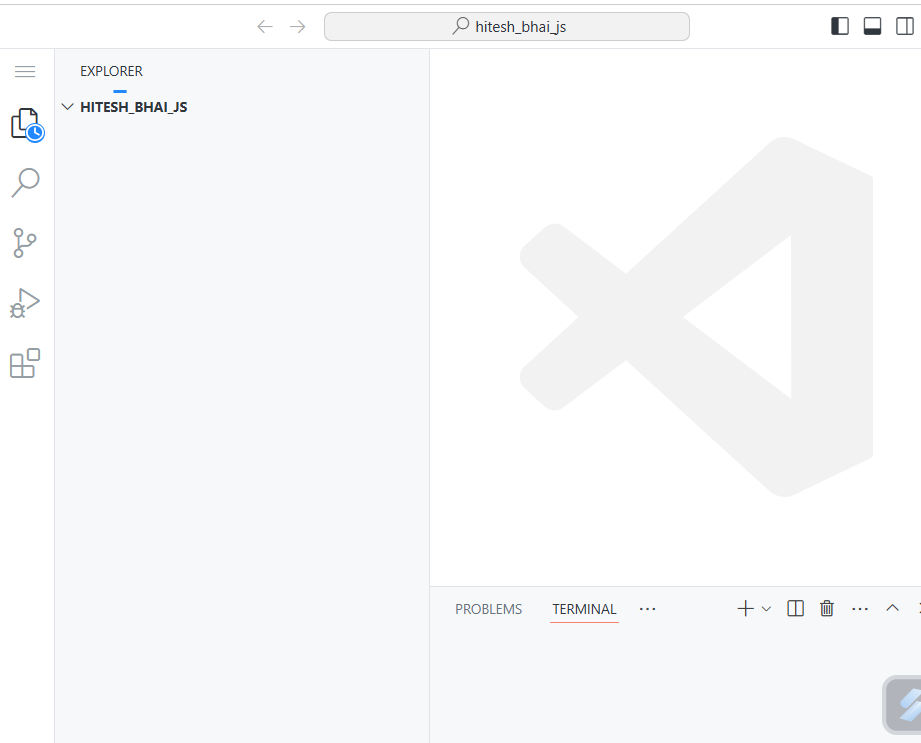


Go to code spaces and then delete the things

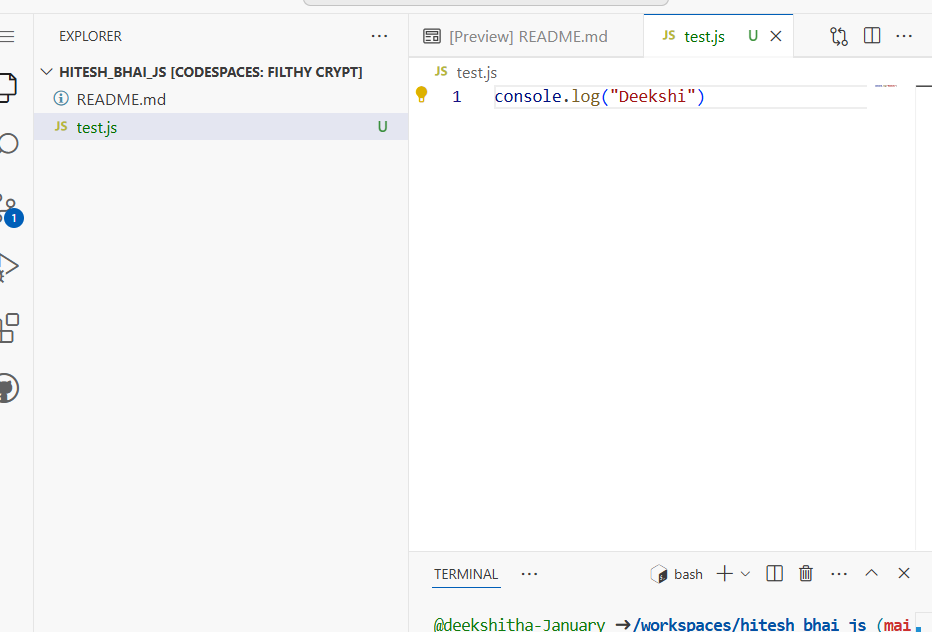
My own



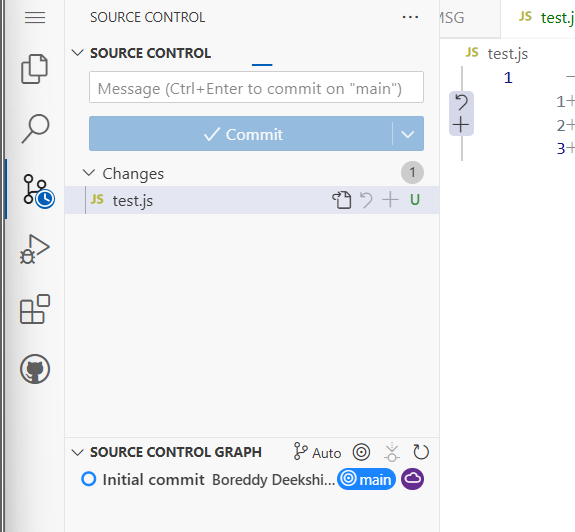
—------------------



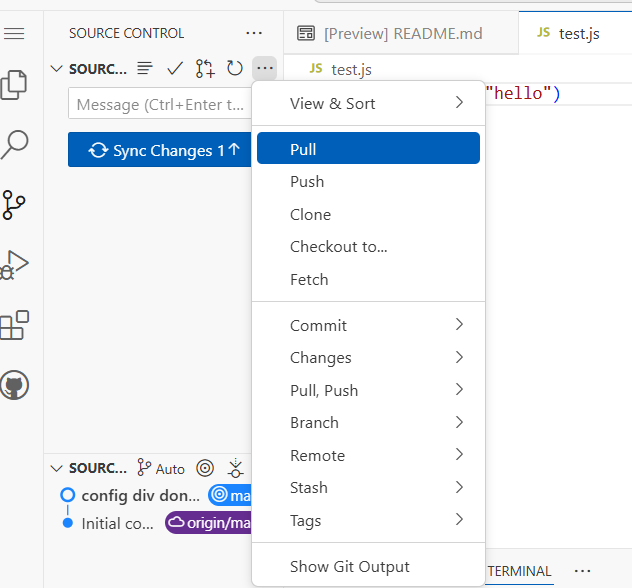
—--------------------



—---------------------

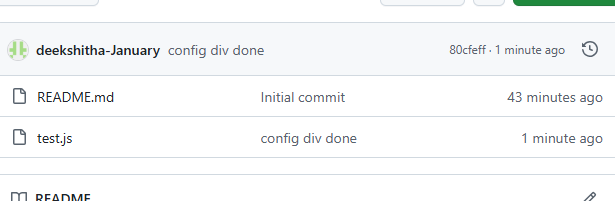


—----------------



—-----------------give message and then do commit and then push on clicking …

—----------------



You can see changes here

—------------------

LECTURE :

# **Let, const and var ki kahani**

// using git for the command line

// keeping versions

// keeping collaborations

// variables are nothing but the memory to store the things

// /if we are making some e-commerce

// then u are going to ask to fill some things

// now they are going to fill those things

// and then now we have all the things entered

// we are going to store the things

// therefore we are going to need memory location

// we have the things like const , var

//

Investigation ki padai :

// let us declare the conet

// there is no need of keeping the naming convention , make them as if they are understandable

//

Declaring the things

const accountId = 12345

let accountEmail = "lahari@gmail.com"

var accountPassword = "12345"

accountCity = "Jaipur"

// const mena the values cant be chnage d

// kuch chhesze change nahi honi chahiye

// we cant change the values

// printing the things

// running is simple that si node filename.js

const accountId = 12345

let accountEmail = "lahari@gmail.com"

var accountPassword = "12345"

accountCity = "Jaipur"

accountId = 2

console.log(accountId)

error

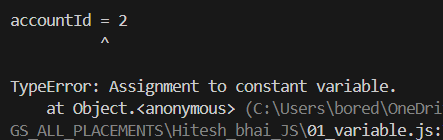
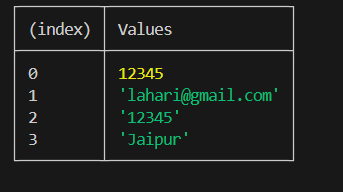


Table method :



const accountId = 12345

let accountEmail = "lahari@gmail.com"

var accountPassword = "12345"

accountCity = "Jaipur"

console.log(accountId)

console.table([accountId, accountEmail, accountPassword, accountCity])

The braces [ ] are important

**CONTS , LET , VAR why all these**

// var can be declared in 2 ways

// using var , let

// in early days the var is going to not have the scope

// it dies not have any control on that

// inside a block the variable is declared and outside something is declared , there is a problem of scope

// they were used interchangeably

// prefer not to use var , because of the block scope and the functional scope

// use const , let not var

// we can write the name without the keyword

//

accountName = "CanaraBank"

//the above is also fine

Null , undefined , define these at home

// if we don't declare a variable , or if we don't initialize a variable then it is classed as that type undefined

// if the variable is wantedly assigned value NULL , then it is NULLa

**LECTURE 5 : Datatypes and ECMA standards**

// in the old java script , the objects , classes, arrow functions were not there

// the foundation want to not kharab old code , by using the new code

// therefore we need to write this strict thing

// and also the standards were set

// if we use this strict once then everything is going to be very strict

// but this is not needed to use , default the standards are default

// but using is a good practice



ALERTS :

// generally alerts are used in the browser , tokae this happen in the node js , we have the different syntax to use this in the node

Semicolon :

// in the react , node , js we are not going to use ;

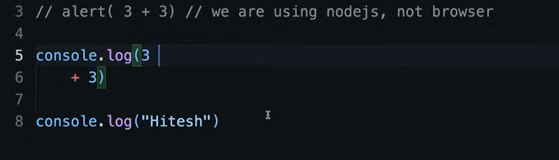
// mainly this is for the readability

//

// in real world coding , here semicolon is not needed

//





This is not readable

ECMA :

This is going to write standards of JS

// it has limited data types

// we have the number , string ,bool , symbol , bigint ,

CODE :

let name = "deekshi"

console.log(name)

let num = 34

console.log(num)

let isLoggedin = false

console.log(isLoggedin)

let state = null

console.log(state)

let message

console.log(message)

console.log("-------------Break--------------")

console.log(typeof name)

console.log(typeof num)

console.log(typeof isLoggedin)

console.log(typeof state)

console.log(typeof message)

console.log(typeof undefined)

console.log(typeof null)

// to find uniqueness we use symbols

// the figma is goingt o sue this a lot

Output :

deekshi

34

false

null

undefined

-------------Break--------------

string

number

boolean

object

undefined

undefined

object

**LECTURE 6 : Data Type conversion confusion**

CODE :

let score = 33

console.log(typeof score)

console.log(typeof(score))

score = "33"

console.log(typeof score)

console.log(typeof(score))

// we aer going to read about the type script

// thsi can be done for somethings only not all things

let num = Number(score)

console.log(num)

console.log(typeof(num))

number

number

string

string

33

number

CODE:

let name = "wer233"

let value = Number(name)

console.log(value)

console.log(typeof(value))

// here the things of conversion are not going to show any error , but they are converted since they cant be converted they are assigned the value NAN

// that means not a number

NaN

number

CODE:

let name = null

let value = Number(name)

console.log(value)

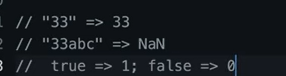
console.log(typeof(value))

// converting null to number is 0

// converting undefined to number is NaN

// converting a alphabetical involved string is going to be NaN

Converting to numbers :



CODE:

// ny=umbers to boolean

let isLoggedin = 1

console.log(typeof(isLoggedin))

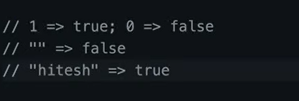
let val = Boolean(isLoggedin)

console.log(val)

number

true

CONVERTING TO BOOL :



Converting to string :

let number = 33

console.log(number)

let val = String(number)

console.log(typeof(val))

console.log(val)

33

string

33

**LECTURE 7 : Why string to number conversion is confusing**

CODE:

// string operations

//this is not a problem , when both the things are strings

let str1 = "Boreddy"

let str2 = "deekshitha"

let str3 = str1 + str2

console.log(str3)

CODE :

console.log(1+ "2")

console.log("1"+2)

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

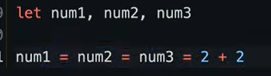
console.log(1+2 +"2")

// these all things are important for exam , not for the real world , we are going to make use of the parenthesis here

// a;ll these outputs depend on the things written in the ECMA script

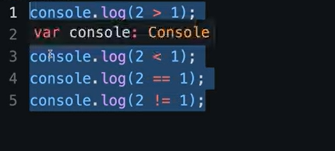
The below is not good code





Prefix , post fix already needed

**Lecture8 : Comparison of data types in javascript**



These are already known



The value cant be predicted

CODE:

console.log("2"> 1)

console.log("01">1)

true

false

CODE :

console.log(null >0)

console.log(null == 0)

console.log(null >=0)

false

false

true

In typescript all these aren't allowed to compare different things

**===, !==**

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

console.log(2!=="2")

// the === is going to check both , if both true then only true

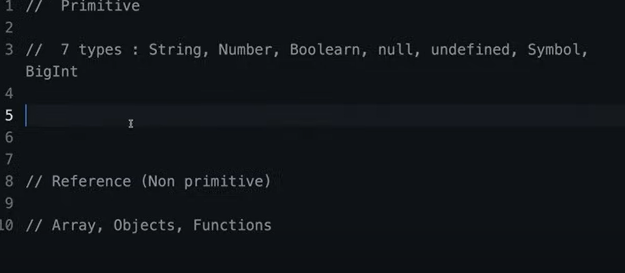
// in !==, one false is enough to make true

**LECTURE 9 : Data types of javascript summary**

// based on accessing data the data is fo 2 types

// like the primitive and non primitive data types

//



CODE: Symbols

const score = 100

const scoreValue = 100.5

const isLogegdin = false

const outsideTemp = null

let userEmail ;

const id = Symbol('123')

const anotherId = Symbol("123")

console.log(id===anotherId);

Console.log bigNumber = 34444444444444

// here we see that the value assigned is same here to id , anotherId , but they are compared and resulted not same

CODE : arrays , objects , functions

// arrays , objects , functions

const heroes = ["shaktiman", "heman", "padman"]

console.log(heroes)

const heroines = {

bahubali : "anushka",

eega: "samatha"

}

console.log(heroines)

// function fun\_name(){

//

// }

// the above is the au from the function

function greetings (){

console.log("konnichiwa");

}

greetings();

console.log(typeof(greetings)) // function , this is the object function

console.log(typeof(heroes)) // array ->object

console.log(typeof(heroines)) // object

CODE :

const bigNum = 23456n

console.log(typeof(bigNum))

**LECTURE 11 : Stack and Heap memory in javascript**

// here the memory is collected al=utomaticcaly

// how memory allocation happens here ein JS

// they are mainly stack , queue

// primitive types in stack

// non primitive types - in hepa

// when we allocate , the things the values are copied , there changing is not reflected in original value

// in heap , the values are referred if they are assigned

// therefore the changes are going to be reflected in the real values also

STACK MEMORY :

let myChannel = "infinity japan"

let anotherChannel = myChannel

console.log(myChannel)

console.log(anotherChannel)

anotherChannel = "hamaraJapan"

console.log(myChannel)

// this is not going to be changed even if we change the anotherChannel

console.log(anotherChannel)

Heap memory :

let user1 = {

name : "deekshi",

password :1234

}

let user2 = user1

console.log(user1)

console.log(user2)

user2.name = "lahari"

console.log(user1)

// user changed by user2

console.log(user2)

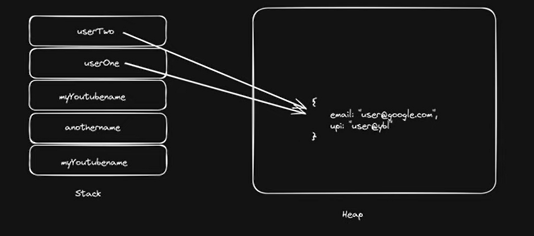
{ name: 'deekshi', password: 1234 }

{ name: 'deekshi', password: 1234 }

{ name: 'lahari', password: 1234 }

{ name: 'lahari', password: 1234 }

DIAGRAM :



**LECTURE 11 : Strings in Javascript**

// strings

const name = "hitesh"

const repoCount = 30

console.log(name + repoCount + "Value")

Way of concatenating strings

Using string interpolation :

// strings

const name = "hitesh"

const repoCount = 30

console.log(`name is ${name} and count is ${repoCount}`)

String as object using String declaration :

const gameName = new String("subway surf")

console.log(gameName)

console.log(typeof(gameName))

[String: 'subway surf']

object

Using normal way :

const gameName = "subway surf"

console.log(gameName)

console.log(typeof(gameName))

subway surf

string

CODE :

const gameName = "subway surf"

console.log(gameName)

console.log(typeof(gameName))

console.log(gameName[3])

console.log(gameName.length)

// this is not a function , above one

// the below 2 are functions

console.log(gameName.toLocaleLowerCase())

console.log(gameName.toLocaleUpperCase())

console.log(gameName.charAt(3))

// to find char at postion 3

console.log(gameName.indexOf('y'))

More methods :

const gameName = "subway surf"

const newString = gameName.substring(1,5)

console.log(newString)

// this is going to assign the new things to the newString , alst index is not included

Trim :

const gameName = "subway surf"

const newStringOne = ' hitesh '

console.log(newStringOne)

console.log(newStringOne.trim())

hitesh

hitesh

Replace :

const gameName = "subway surf"

const newStringOne = ' hitesh '

console.log(newStringOne)

console.log(newStringOne.trim())

const url = "https://hiteshChaudary.com/hitesh"

console.log(url.replace("hitesh","lahari"))

https://lahariChaudary.com/hitesh

Code : include

const gameName = "subway surf"

console.log(gameName.includes("way"))

True

Code : split

const gameName = "subway @ surf"

console.log(gameName.split("@"))

[ 'subway ', ' surf' ]

Returns an array

**Lecture 12 : Number and Maths in Javascript**

const score = 300

console.log(score)

const num = new Number(400)

console.log(num)

300

[Number: 400]

CODE : Number functions

const score = 300

console.log(score)

const num = new Number(400)

console.log(num)

console.log(num.toString())

console.log(num.toString().length)

// this si important in ecommerce

console.log(num.toFixed(2))

const number2 = 123.4543

console.log(number2.toPrecision(3))

const balance = 1000000000

console.log(balance.toLocaleString())

console.log(balance.toLocaleString('en-IN'))

// the above is in Indian standard

300

[Number: 400]

400

3

400.00

123

1,00,00,00,000

1,00,00,00,000

**MATHS :**



—----------------

Functions and properties

// it is a default library in JS

//

console.log(Math)

console.log(Math.abs(-4))

console.log(Math.round(4.5))

console.log(Math.ceil(4.5))

console.log(Math.floor(3.9))

console.log(Math.max(3,2,-1,9))

console.log(Math.random())

console.log(Math.random()\*10)

console.log(Math.random()\*10 +1)

// to avoid 0 in the case of 0.03

Object [Math] {}

4

5

5

3

9

0.6091780610323743

2.667518983765509

1.4999397111309412

Code :

const min = 10

const max = 20

console.log(Math.random() \* (max-min+1)\*+min)

**LECTURE 13 : Date and time in depth in javascript**

// the date is calculated in milli seconds

// it is calculated from jan 1, 1970

//

Code :

let myDate = new Date()

console.log(myDate)

2024-10-29T16:08:59.732Z

let myDate = new Date()

console.log(myDate.toString())

// output

// Tue Oct 29 2024 21:39:29 GMT+0530 (India Standard Time)

console.log(myDate.toLocaleString())

// 29/10/2024, 9:40:27 pm

console.log(myDate.toDateString())

// Tue Oct 29 2024

console.log(typeof(myDate))

// object

let myCreatedDate = new Date(2024, 0, 23)

console.log(myCreatedDate.toDateString())

// Tue Jan 23 2024

// month starts with 0 here in JS, format is yy/mm/dd

let myCreatedDate = new Date(2024, 0, 23, 4,5)

console.log(myCreatedDate.toLocaleString())

// 23/1/2024, 4:05:00 am

CODE:

let myCreatedDate = new Date("2024-02-13")

console.log(myCreatedDate.toDateString())

// Tue Feb 13 2024

// here month is normal only

TIMESTAMPS :

let myTimeStamp = Date.now()

console.log(myTimeStamp)

// 1730218603375

// converting date to time

let myCreatedDate = new Date("2024-02-13")

console.log(myCreatedDate.getTime())

// 1707782400000

// seconds

console.log(Date.now())

// 1730218723116

console.log(Math.floor(Date.now()/1000))

// 1730218752

Finding specific things :

let newDate = new Date()

console.log(newDate)

// 2024-10-29T16:20:57.209Z

console.log(newDate.getMonth())

// 9

console.log(newDate.getDay())

// 2

**LECTURE 14 : Array in Javascript**

// arrays

// () - parenthesis , []- brackets , {} - braces

// elements can be of any data type

// array is an object

// it can store multiple elements of any time

// it is dynamic , can add many values

// array elemnst can be accessed suing indexes

const myArray = [0,2,3,4,5, true, "Hello"]

CODE for creating arrays

const myArray = [0,2,3,4,5, true, "Hello"]

console.log(myArray[2])

// 3

const heroes = ["hanuman", "bahubali", "shakti"]

console.log(heroes)

// [ 'hanuman', 'bahubali', 'shakti' ]

const myNames = new Array(1,23,43,32)

console.log(myNames)

// [ 1, 23, 43, 32 ]

// arrays deep copy , shallow copying

//

Array methods :

// array methods

//

const myArray = [3,5,2,3]

console.log(myArray)

// [ 3, 5, 2, 3 ]

myArray.push("helen")

console.log(myArray)

// [ 3, 5, 2, 3, 'helen' ]

// here array has been changed , we have added a new value to this

myArray.pop()

console.log(myArray)

// [ 3, 5, 2, 3 ]

Shift and the unshift they are going to work at the friend , therefore we need to shift the things

**Unshift - push at the begin**

**Shift - pop at the begin**

const nums = [3,4,5,6]

console.log(nums)

// [ 3, 4, 5, 6 ]

nums.shift()

console.log(nums)

// [ 4, 5, 6 ]

nums.unshift("hello")

console.log(nums)

// [ 'hello', 4, 5, 6 ]

Index, includes :

const nums = [3,4,5,6]

console.log(nums)

// [ 3, 4, 5, 6 ]

console.log(nums.includes(3))

console.log(nums.indexOf(2))

console.log(nums.indexOf5)

[ 3, 4, 5, 6 ]

true

-1

undefined

JOIN :

const myArray = [2,3,4,"hanji"]

const newArray = myArray.join()

console.log(newArray)

// 2,3,4,hanji

Converted to string , [] are removed

Slice , splice

const myName =[1,2,3,"hello",32,45,67,]

const newArr = myName.slice(1,4)

console.log(newArr)

// [ 1, 45, 67 ]

console.log(myName)

// [ 2, 3, 'hello', 32 ]

// in slice no effect on original string , and the part is last ndex excluded

const newArray2 = myName.splice(1,4)

console.log(newArray2)

// [ 1, 2, 3, 'hello', 32, 45, 67 ]

console.log(myName)

// [ 1, 45, 67 ]

// difference is , that , in splice the slice is going to be full ,a dn it is cut from the original string

**LECTURE 15 : Array part 2 in Javascript Hindi**