FULL STACK NOTES

* What is javaScript

Javascript is cross-platform object oriented scripting language used for interactive web pages,manilulation of UI.

JS can be used for both side on server side also client side ,

Server side -> node.js

Client side -> browser objects DOM

It contain standard libraries , have array , dsate , control structure

Advantages

1. **Interactivity and User Experience**:

JavaScript allows developers to create dynamic and interactive web pages. It enables real-time content updates, animations, form validations, and other features that enhance user experience

1. **Versatility**:

JavaScript can be used for both client-side and server-side development. On the client-side, it interacts with the DOM (Document Object Model) to manipulate HTML and CSS. On the server-side, platforms like Node.js allow JavaScript to handle server operations, databases, and APIs.

1. **Cross-Platform**:
2. Asynchronous Programming

* JS vs ECMA script
* Dynamically typed

Datatypes of variable are assigned at runtime according to value.

Eg. Var a= 10;

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

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

a=”Hello”

* Variable Declaration

Var a= 10;

Var-> declaration

10-> initialization

* Naming convention

Nam of variable start with a-z,A-Z,\_,$

* Scope

Global scope –

Function scope -

Module scope-

Block scope-

* Difference

Var

\*) Var is function scope where as let or const is block scope.

\*) Var allow redeclaration but let ,const not allow.

\*)Var supports the hoisting property but let, const is not support.

Hoisting- use a variable before declaring it .

a =10;

var a=20;

console.log(a);

but it read as

var a;

a=10;

a=20;

the output is 20;

* Var or let initialisation is optional but is not possible in const

Let

Const

Direct

* Questions

1)

Const scope =”outer”

Function checkscope()

{

Const scope=”inner”;

Return scope;

}

Console.log(checkscope);

Answer is inner

* DataTypes

8 data type are there

7 is primitive datatype;

These are Number, String , bollean ,null , undefined , symbol , bigInt

1 is Non-Primitive is object.

Const sym=Symbol();

Const sym1=Symbol(“first”)

Note- new keyword is used create new value with construction

But not new key word is not used with symbol

It will generate immutable keys

Eg

Const sym1=Symbol(“first”)

Const Sym2=Symbole(“first”)

If(sym1==sym2)—it is not true

Special values use

1. Possitive infinity
2. Negative infinity
3. NAN

Math.max()---negative infinity

Math.min()--- positive infinity

HOW is it use – NUMBER.NEGATIVE\_INFINITY

NAN—

Eg.

“Hello” \* 5 - NAN

“5” \* 5 – 25

parseInt()—convert into integer

parsefloat() – convert into float

OPRERATORS

1)X&&=1== X&&(X==1)

2)X||=1=X||(X=1)

3)X??=1 operator nulish operator

Const a{duration : 50}

a.speed??=25

a.duration??=50;

Var a=1n+4

.. the >>> which is not use with bigInt.

8n>>>2n

Which is not give any output

Suggest to change the operator .

FUNCTION

1)simple function

2)function expression

3)IIEF – immediately invoke function expression

4)Annonymous function

5)Using construction

6)Recursive function

7)function hoisting

8)function scope,closure property

9)Function argument

Nested function:function inside function

IIFE- Immediatly invoked function expression

Self invoked anonymous function

(function(x)

{

console.log(x\*x);

})(5),

recursion are apply in application in tree traversal

functions expression and arrow function do not support hoisting property

in which use a closure properties; Run =function B(x)

{ return x+y; }

closure property:the function binds with its self by environment in which its declared

Arguments

    function Sum(a,b)

    {

        let c;

        c=a+b;

        return c;

    }

    // console.log(Sum(3,4))

    // console.log(Sum(3))

    // console.log(Sum(3,4,5))

    console.log(Sum(3,'4',5))

function Sum()

{

Let sum=0;

For(let i=0;i<arguments.length;i++)

{

Sum+=arguments[i];

}

Return sum;

}

console.log(Sum(2,4,6,7,0)) Argument is aaray like structure stored multiple .in which also we can use arguments.length

in argument all properties of array is not works.

# JS ARRAY

1. Hetrogenous in nature( different data type)

Eg. Const arr=[“sandeeep”,42,null,[7,8,9],{}]

1. Js Array is resizable (dynamic)
2. Array is 0 indexed
3. If we create the copy an array using inbuilt js operation then shallow copy is created

What is difference between shallow copy and deep copy

Declarations of array

1)Declare array using literals.

Const arr=[6,7,8,9,10,”hello”]

2)Declare array using constructor

Const arr=new Array(1,2,3,4);

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

console.log(arr);

undefined is datatype and value also

the output of the arr is 1 2 empty-item

if we console index 2 then output is undefined

Nested Array , Multidimentional Array

Const arr=[6,7,8,[11,12],[13,14]] // 2d array

Arr[3][0]= 11

Arr[3][1]=12

Const arr=[6,7,8,[11,12],[13,14,[1,2,3,4]]] // 2d array

* Methods that are used in Array

Push,pop, unshift, slice , concat , join , from , isArray , includes , indexOf, split

* Push

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

console.log(arr.push(6))

push return length of new array

const arr=[1,2,3,4,6]

console.log(arr.push([1,2,34,]))

console.log(arr)

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

arr.push(...[4,5,6])

console.log(arr)

… is used to avoid nested array

Or that use for merging to array

* Pop

const arr=[1,2,3,4,6]

console.log(arr.pop())

return the last element from the last

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

const a=arr.pop(6)

console.log(a)

Not give any error

* unshift

const arr=[1,2,3,4,6]

console.log(arr.unshift(8))

insert value from starting from starting

* Shift

const arr=[1,2,3,4,6]

console.log(arr.shift())

delete element from the starting or return element

Important question betwren Slice vs Splice

* slice

const arr=[1,2,3,4,6]

console.log(arr.slice(1,3))

output is 2,3 In array form

\*) slice return the new array . or no change in original change

1. Is starting index
2. Is excluded

const arr=[1,2,3,4,6]

console.log(arr.slice(0))

it return the duplicate array

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

const newarr=arr.slice(-1);

console.log(newarr)

print last element  
exclude the last element

Splice

1)splice operation is performed on original array

2) splice insertion

3) splice deletion

const fruits=["Apple", "Banana", "Orange" , " Grapes"];

console.log(fruits.splice(2,0,"Guava","mango"))

1. Starting index

0-deletion count

,"Guava","mango")- is inserting in array

\*)no delete in element from array. Or guava and mango is added from starting index

console.log(fruits)

* Concat

const arr1=["Apple", "Banana", "Orange" , " Grapes"];

const arr2=[1,2,3,4,5]

const arr3=[1,2,3,4,5]

const newarr1=arr1.concat(arr2)

const newarr=arr1.concat(arr2,arr3)

const newarr2=[...arr1,...arr2,...arr3]

console.log(newarr2)

… = spread operator

* Join

const arr=[1,2,3];

const newarr=arr.join();

console.log(newarr)

output- 1,2,3

Convert into string

Eg. If manipulation of number and after then send into string then we are using Join()

const arr=[1,2,3];

const newarr=arr.join("$");//”” this is work as delimenator

console.log(newarr)

ouput - 1$2$3

const arr=[1,2,3];

const newarr=arr.join("$");

console.log(Array.isArray(newarr))

Array – constructor

> Type of

1)typeOf used to find the index of array

2) if element is not present then return -1.

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

 console.log(arr.indexOf(1))

retrun 0

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

 console.log(arr.indexOf([1,2,3,4]))

return -1

this is not working in object and array

if we want use on object then we can use

* findOf()
* Include

It use for checking the element is present in array or not .

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

console.log(arr.includes(4));

true

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

console.log(arr.includes(9));`

false

* From

It is a static method of array.

It is with constructor.

console.log(Array.from("Ranjeet"))

output --

[

  'R', 'a', 'n',x

  'j', 'e', 'e',

  't'

]

console.log(Array.from([1,2,3,4],(x)=>x+x))

[ 2, 4, 6, 8 ]

How it is work 1+1,2+2,3+3,4+4

It creates the array

* Split

const split="my name is ranjeet"

let answer = split.split(" ")

console.log(answer)

output is [ 'my', 'name', 'is', 'ranjeet' ]

it return array

const wordBank="ranjeet tiwari RanjeetTiwari";

let newa=wordBank.split(" ")[0]

output is ranjeet

const wordBank="ranjeet tiwari RanjeetTiwari";

let newa=wordBank.split(" ")[0][0]

console.log(newa)

output is r

* Flat---

const arr=[1,2,3,1,[1,2,3,4,[1,2,3,4,55]]]

 console.log(arr.flat(Infinity))

output is

[

   1, 2, 3, 1, 1, 2,

   3, 4, 1, 2, 3, 4,

  55

]

const arr=[1,2,3,1,[1,2,3,4,[1,2,3,4,55]]]

 const newar=arr.flat(Infinity)

 console.log(newar)

* isArray

it is contructor .

const arr=[1,2,3,1,[1,2,3,4,[1,2,3,4,55]]]

const result=Array.isArray(arr);

console.log(result)

* ---------------- Iterators---------------
* forEach
* map
* filter

const arr=[1,23,4,5,98,23,65]

const newarr=arr.filter((e)=>(e%2==0))

console.log(newarr)

const arr=[1,23,4,5,98,23,65]

const newarr=arr.filter((e)=>{

  if(e%2==0)

  {

    return true

  }

})

console.log(newarr)

* reduce

it is use to find a one solution like sum,min,max

const arr=[1,23,4,5,98,23,65]

const sum=arr.reduce((prev,curr,currIndex,arr)=>{

 return prev;

})

console.log(sum)

prev and curr in starting is index one

max of array

const arr=[1,23,4,5,98,23,65]

const sum=arr.reduce((prev,curr,currIndex,arr)=>{

     prev+=curr;

     return prev

},0)//0 is used to initialise with 0

console.log(sum)

const arr=[1,23,4,5,98,23,65]

const max=arr.reduce((prev,curr,currIndex,arr)=>{

     if(curr>prev)

      prev=curr

     return prev

},0)

console.log(max)

* Object
* // object //
* const obj=new Object();
* const obj1={
* name:"ranjeet",
* age:22
* }
* const obj2={}
* obj2.name="Ranjeet";
* console.log(obj2.name)

const obj1={

   name:"ranjeet",

   age:22

}

const obj2={}

obj2.name="Ranjeet";

obj2.name="Sagar"

console.log(obj2.name)

if I change the value new with const it can changeable;

const obj1={

   name:"ranjeet",

   age:22

}

const obj2={}

obj2.name="Ranjeet";

obj2.name="Sagar"

delete obj2.name

console.log(obj2.name)

delete operation can be use for deleting the attributes

const obj3={

    "First Name":"Sandeep",

}

console.log(obj3["First Name"])

1)if space in key then access in square baracket

const obj3={

    "First Name":"Sandeep",

    "10th\_Marks":78

}

console.log(obj3["10th\_Marks"])

2)in this case key is invalid because start with number . then we can acces with square bracket

This are two anomalis

const obj3={

    "First Name":"Sandeep",

    "10th\_Marks":78,

    display\_marks:()=>{

        console.log(`${obj3["First Name"]} has ${obj3["10th\_Marks"]}`)

}

}

console.log(obj3.display\_marks())

ouput Sandeep has 78

undefined

// const obj3={

//     "First Name":"Sandeep",

//     "10th\_Marks":78,

//     display\_marks:()=>{

//         console.log(`${obj3["First Name"]} has ${obj3["10th\_Marks"]}`)

// }

// }

// const obj4={

//     "first Name":"Ranjeet",

//     "!0th marks":68,

//     display\_marks:obj3.display\_marks()

// }

Output -- Sandeep has 78

exception

const obj3={

    "First Name":"Sandeep",

    "10th\_Marks":78,

    display\_marks:function(){

        console.log(`${this["First Name"]} has ${this["10th\_Marks"]}`)

}

}

const obj4={

    "first Name":"Ranjeet",

    "!0th marks":68,

    display\_marks:this[obj3.display\_marks()]

}

Ouput: Sandeep has 78

In this case function is call then it console the ouput

const obj3={

    "First Name":"Sandeep",

    "10th\_Marks":78,

    display\_marks:function(){

        console.log(`${this["First Name"]} has ${this["10th\_Marks"]}`)

}

}

const obj4={

    "First Name":"Ranjeet",

    "10th\_Marks":68,

    display\_marks:function(){

        console.log(`${this["First Name"]} has ${this["10th\_Marks"]}`)

}

}

obj3.display\_marks();

obj4.display\_marks();

function show(){

    console.log(`${this["First Name"]} has ${this["10th\_Marks"]}`)

}

const obj3={

    "First Name":"Sandeep",

    "10th\_Marks":78,

    display\_marks:show

}

const obj4={

    "First Name":"Ranjeet",

    "10th\_Marks":68,

    display\_marks:show

}

obj3.display\_marks();

obj4.display\_marks();

a ganeralise function is used in many object

console.log(obj4.hasOwnProperty("fisrt name"))

it is used check have this attribute or not

Q3.

const arr=[

     {

        name:"Sandeep Rana",

        age:42

     },

     {

        name:"Rohit Sharma",

        age:34

     },

     {

        name:"Ranjeet Tiwari",

        age:22

     }

]

const newarr=arr.map((ele)=>{

    return(ele.name.split(" ")[0][0] + ele.name.split(" ")[1][0] )

})

console.log(newarr);

output: [ 'SR', 'RS', 'RT' ]

Calculate bill

const categories = [{

  id: "C1",

  categoryName: "Platters",

  superCategory: {

    superCategoryName: "South Indian",

    id: "SC1"

  }

}];

const menu = [

  {

    id: "item1",

    itemName: "Butter Roti",

    rate: 20,

    taxes:

      [

        {

          name: "Service Charge",

          rate: 10,

          isInPercent: false

        },

        {

          name: "GST",

          rate: 5,

          isInPercent: true

        }

      ],

    category:

    {

      categoryId: "C2"

    }

  },

  {

    id: "item2",

    itemName: "kadhai Panner",

    rate: 20,

    taxes:

      [

        {

          name: "Service Charge",

          rate: 10,

          isInPercent: false

        },

        {

          name: "GST",

          rate: 5,

          isInPercent: true

        }

      ],

    category:

    {

      categoryId: "C2"

    }

  },

  {

    id: "item3",

    itemName: "chur chur naan",

    rate: 20,

    taxes:

      [

        {

          name: "Service Charge",

          rate: 10,

          isInPercent: false

        },

        {

          name: "GST",

          rate: 5,

          isInPercent: true

        }

      ],

    category:

    {

      categoryId: "C2"

    }

  }

];

bill = {

  id: "B1",

  billNumber: 1,

  opentime: "06 Nov 2020 14:19",

  customerName: "CodeQuotient",

  billItems: [

    {

      id: "item2",

      quantity: 3,

      discount: {

        rate: 10,

        isInPercent: true

      }

    },

    {

      id: "item3",

      quantity: 5,

      discount: {

        rate: 10,

        isInPercent: false

      }

    },

  ]

};

calculate(bill);

function calculate(bill) {

  let totalAmount = 0;

  const billdetail = [];

  for (const billItem of bill.billItems) {

   const menuItem = menu.filter(v => v.id === billItem.id)[0];

    if (menuItem) {

      let itemTotal = menuItem.rate \* billItem.quantity;

      let discount = billItem.discount.rate;

      //discount

      if (billItem.discount.isInPercent) {

        itemTotal -= (itemTotal \* discount / 100);

      }

      else {

        itemTotal -= discount;

      }

      //tax

      for(const tax of menuItem.taxes)

      {

        if(tax.isInPercent)

        {

          itemTotal+=(menuItem.rate\*tax.rate/100)

        }

        else

        {

          itemTotal+=tax.rate;

        }

      }

     //detail add

     totalAmount+=itemTotal;

     billdetail.push({

      itemname:menuItem.itemName,

      Quantity:billItem.quantity,

      itemTotal:itemTotal,

      rate:menuItem.rate

     })

    }

  }

  console.log(totalAmount)

    for(let i of billdetail)

    {

      console.log(`${i.itemname} @ ${i.rate} \* ${i.Quantity} = ${i.itemTotal}`)

    }

}

ASYNCHRONOUS PROGRAMING // Non blocking

* Javascript is Single threaded language. Javascript by default is synchronous Top to botton

{

--Statement 1

--Statement 2

--Statement 3

--Statement 4

}

By default behavoir javascript to excuted line by line. At a time one statement is executes

Problem of synchrousely is

Img1Space--

When data is fetch then other statement is excuted . it is waiting . javascript is blocking nature.

Event -loop, micro queue, microStack

setTimeOut- delay introduce using setTImeout

setInterval-

CallBack

Promise

Asynce await

* Concept of Callback (used before ES6)

Mimic the server

Main problem in callback

Nesting – callback hell ( higher order function ) one by one callback executed

To solve this we use promises

Promise.all

Fetch take two then

Fetch().then().then()

FETCH

For data - <https://jsonplaceholder.typicode.com/>

FETCH RETURN S THE PROMISE

How fetch statement works

Promise.all

interview Question

Q.1(one promise give error than then remaining url is working or not )

Ans-In Promise.all even one url is gives error then remaining promises is also not working .

Async Await

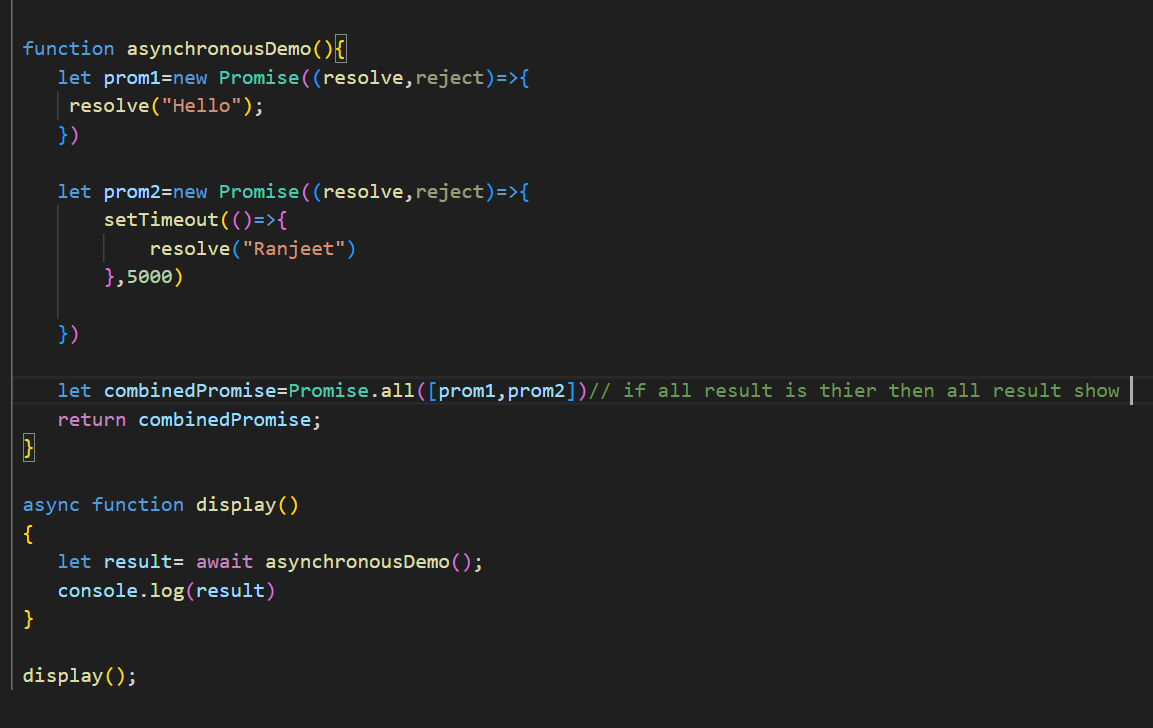
Function f()

{

Return 1;

}





**XMLHTTPS /XHR/AJAX**

***TYPESCRIPT***

***/// typeScript definition //interaction***

***// Feature // Function overloading***

***//Need Interfacees***

***//Transpiler   tsc //class***

***//  Enironment***

***// install i typeScript -g***

***//check    version//***

***//tsconfig    // tsc init***

***// setting the input and output folder / not yet***

***// debugging // not yet***

***///DataTypes***

***//Array***

***//tuple***

***//enum***

***// any***

***// unknown***

***// never***

***//type Annomation***

***// type inference***

***//***

***// function***

***//union***

***//type narrowing***

***// optional parameter  ?***

***// defualt parameter***

***// object:***

***//type alias / custom type***

***// Literals***

***// optional chaining***

***// Nillables***

***Type script statically check the error.***

***For compiling the code we can use typescript compiler.***

***tsc-compiler***

* ***Typescript is superset of javascript.it means all the code written in javascript is valid in typescript.***
* ***Typescript provide us some additional feature.***

***Eg.***

1. ***Strong typing,***
2. ***objected oriented features like classes, interfaces***
3. ***Compile time error handing.***

* ***TypeScriopt come with script compiler (tsc- typescript compiler).***

***Use- it transpiles typescript to java script.***

***Transpile means— convert high level to high level.***

* ***Tsc compiler change into javascript and uses ecmaScript 6 befores version. This is called Downlabeling.***

***Tsc init …***

***For the tsconfig.json file***

* ***Data type in typescript***

***String ,Boolean,number,null,undefined,symol,,bigint,object - java script***

***//Any,array,typle,enum,union,never,unknown***

***let b:bigint=32456655n - it gives error in ecmascript 2016***

***if we wat to use by changing the to ec2020***

1. ***String***
2. let firstname:string='Ranjeet' ;
3. let b:string=`welcome ${firstname}v to chitkara`

***2)Boollean***

***let isEven: boolean=true;***

***3)Array***

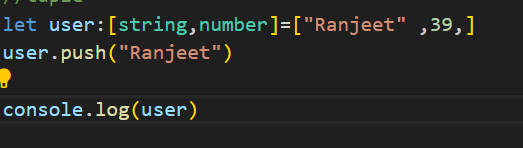
***let arr=[1,2,3,3,'String',true]***

***let arr1:number[]=[1,'Ranjeet',] ///it gives error***

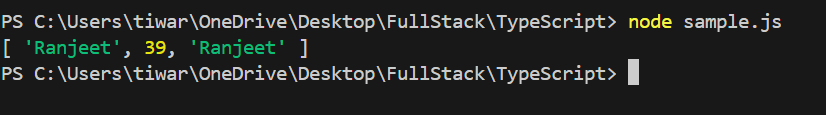
***let arr2:(string | number)[]=[] /// (| this is use for enum)***

1. ***Tuples***

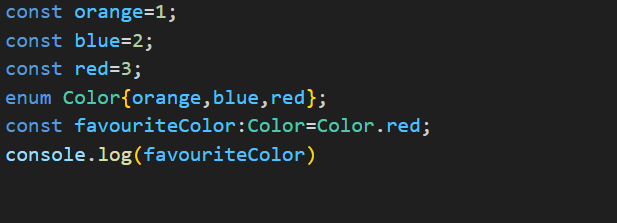
***It is fixed size array. And also defined***

******

***output***

******

1. ***Enum***

******

***It start with zero***

******

***It start with 1.***

***Enum are related values that are***

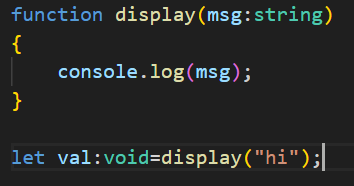
******

***Enum is converted in to function in js file.***

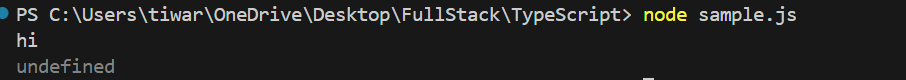
1. ***Difference with never and void ?***

***--Void – null and undefined can value in void but not in never.***

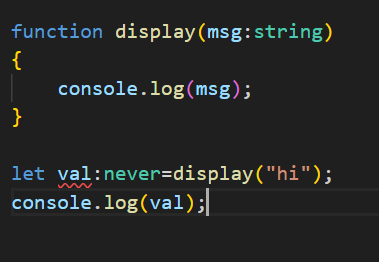
***If a function have void data type then cannot return anything. By default it is void.***

******

***Output***

******

***--Never***

******

***It gives error***

***Type inference – automatically take data type***

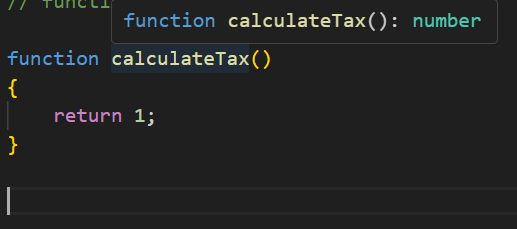
1. ***Type Annotation***

***Ans:***

***Function***

***By default return type void***

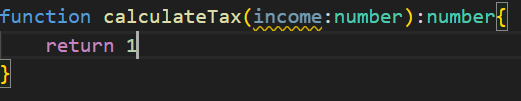
***And return value undefined***

******

***This is type inference to autoselect type***

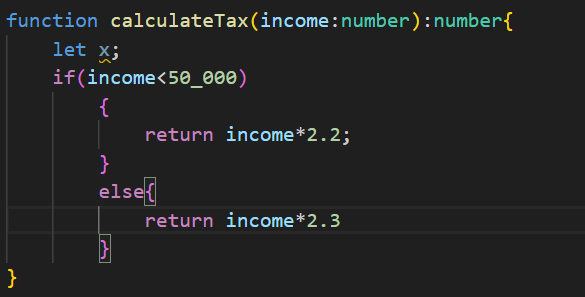
***// in type Checking***

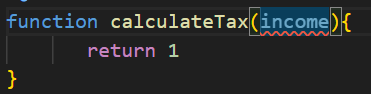
1. ***noUnusedParamter : true;***

******

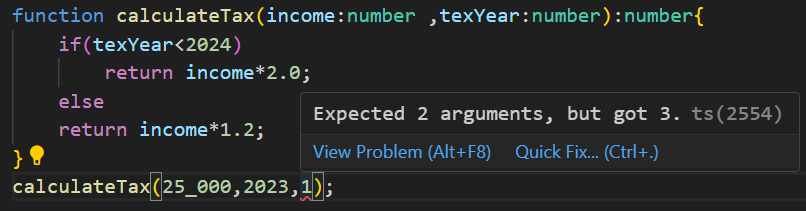
***for unused functions that return something***

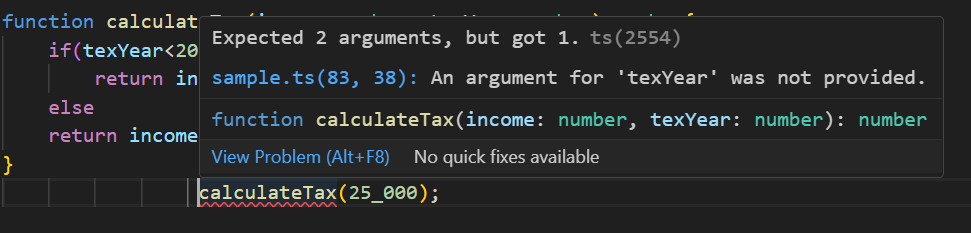
1. ***"noUnusedLocals": true,***

******

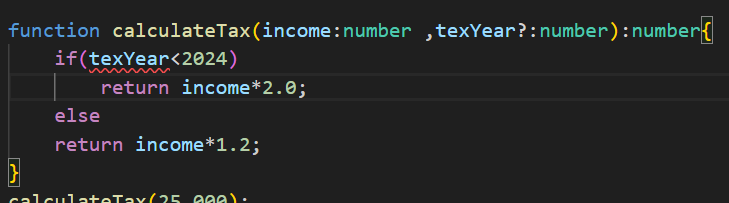
******

1. ***"noImplicitReturns": true,***

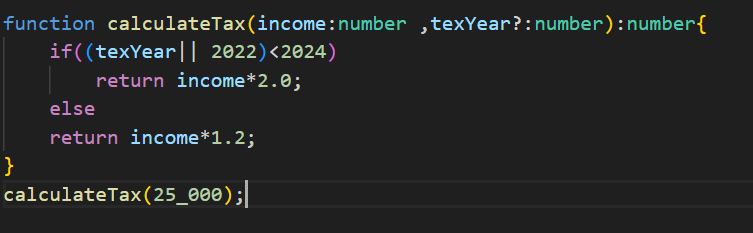
******

******

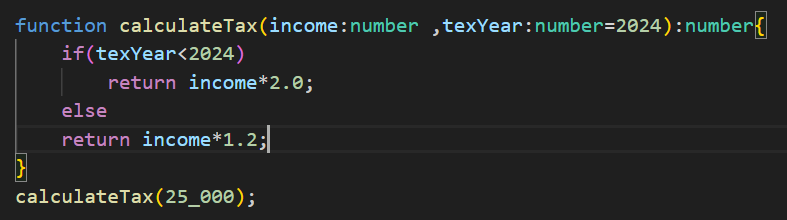
***Error in ts***

******

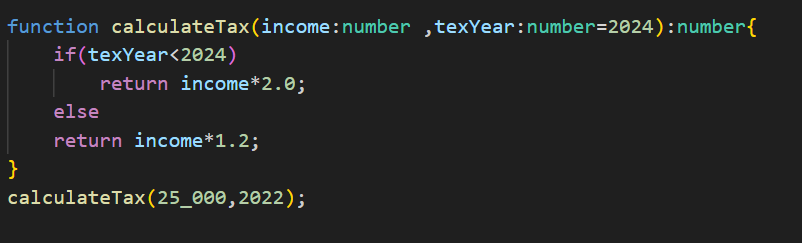
***If we want to do then***

******

***Another method***

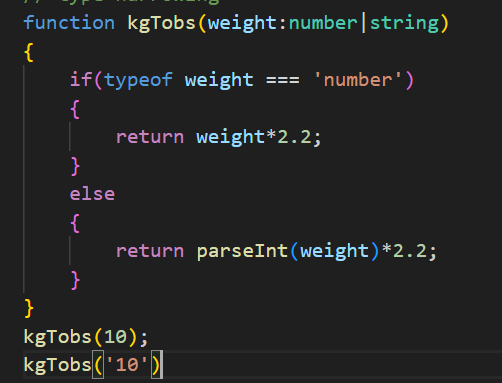
******

***// void id the default returning type***

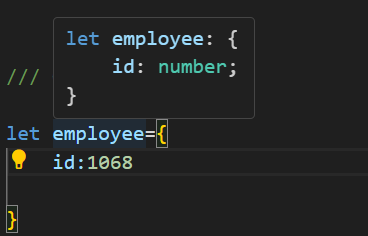
******

***Overright the 2022 on 2024***

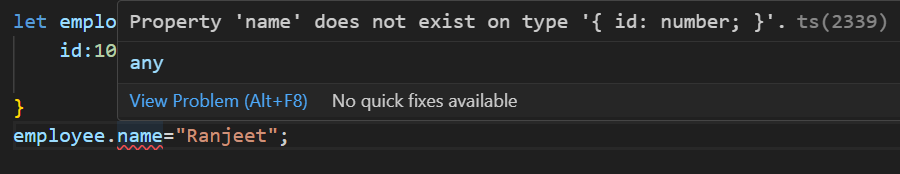
* ***Type Narrowing***

******

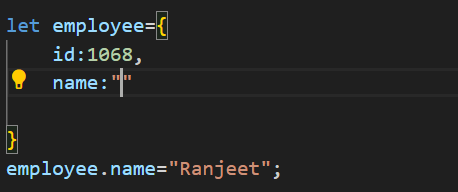
* ***Objects***

******

***Shape of object***

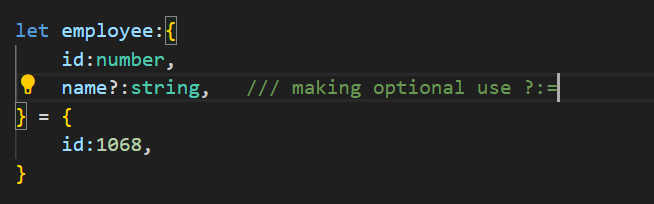
******

***It is not possible in ts . but it is possible in js***

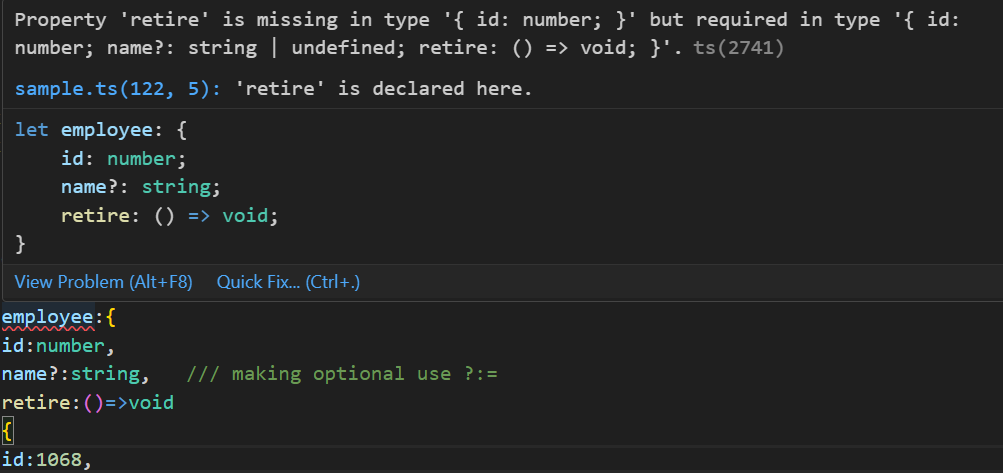
******

***This is modatory to assign it***

1. ***Optinal***

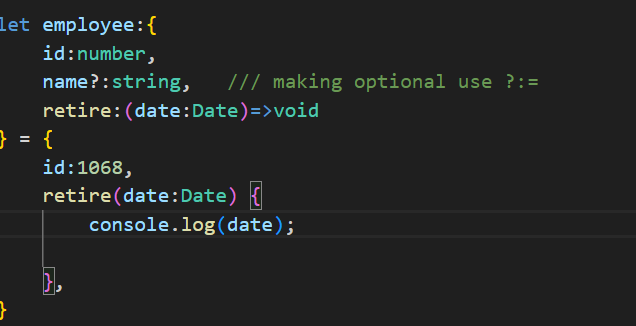
******

1. ***Function in object***

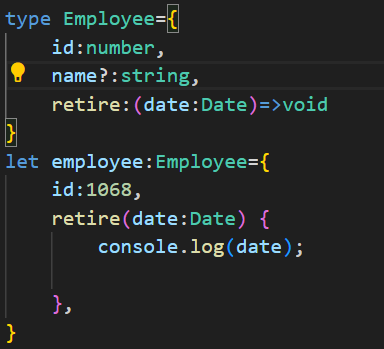
******

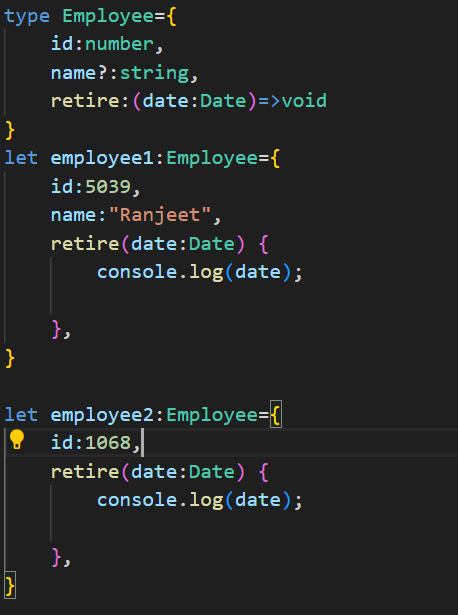
***Error because we are not decalred retire***

***Correct this***

******

1. ***Custum type // type alias***

******

******

***Type alias with***

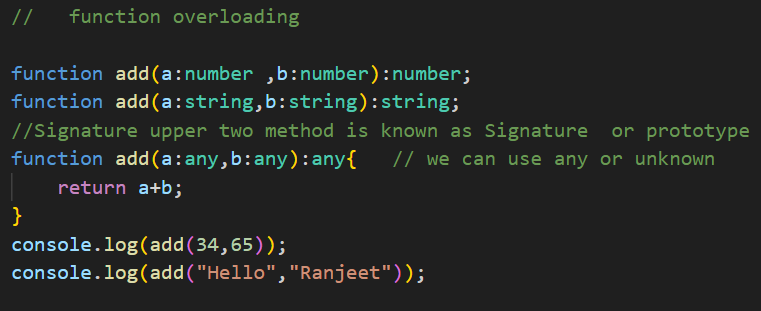
* ***Assignment 1***

***write a function in typescript to calculate the factorial of the number***

* ***Assignment 2***

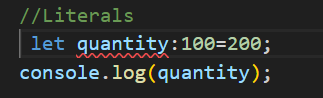
***Write a function to convert a temperature a feh to cel***

* ***function overloading***

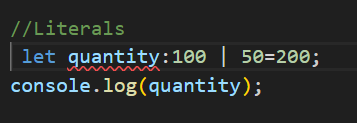
******

1. In which number of parameters are same but data type can be change.
2. while function overloading we declared the function with two different prototype which may contains same number of argument . but with different datatype. We give a common implementation

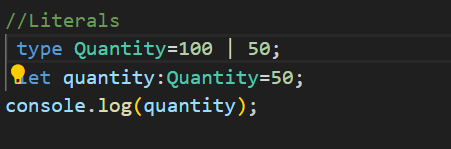
* ***Literals***

******

If I put different with 100 then its give error



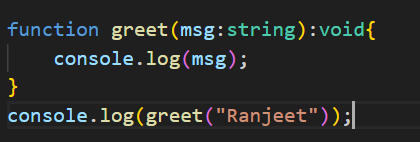
Again its gives error



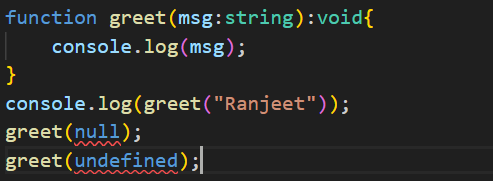
It is the solution

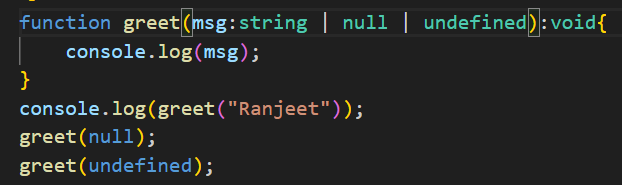


* ***Nullables***

******

******

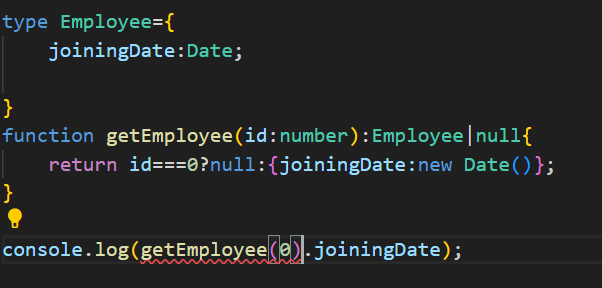
******solved

******

 second method

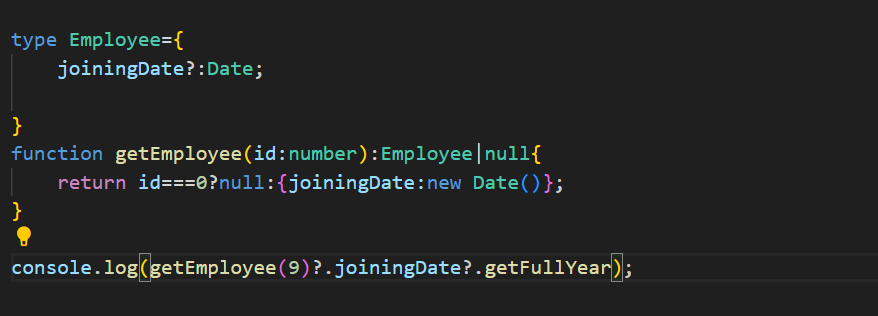
1. It is not recommended to this because we want to check null values.

* ***Optional Chaining***

******

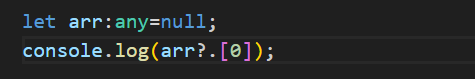
* It canbe corrected with two way

1. If else
2. Put ?
3. Optional operator access property

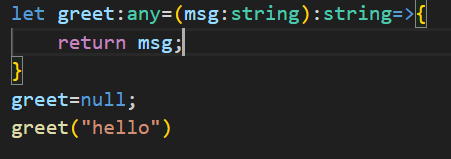


If I make joiningDate optional

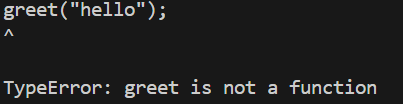
1. optinal operator accessing the element



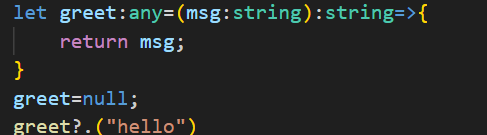
1. optional operator accessing the call



gives error

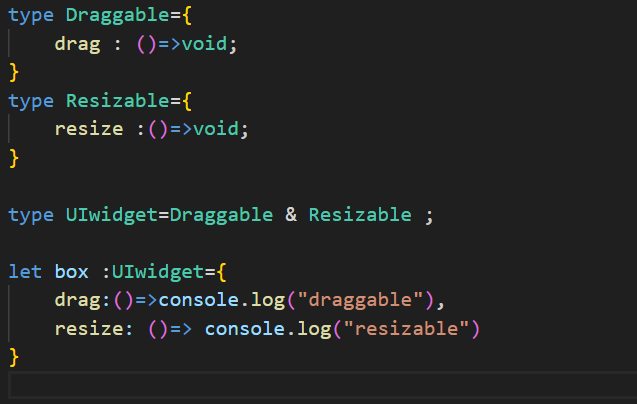


It can be solve with



Add optinal operator

* ***Intersection***



* ***Interfaces***

To provide structure of object

Mongo Db

Collection in no Sql

DBS

1. Collection -- more than one collection can possible
2. Document inside collection – more than one document can be possible . and the size of document can be vary
3. Cannot changes on local , admin
4. Use command is use for create database and for switching to database.
5. To show collection use keyword is (show collections)

* **Node**
* Node.js

1. Modules -> Scope
2. Events
3. In-built
4. Custom module
5. Creating Http

* Commands

1. npm init -y -- // it create package.json