**Section 1: Introduction**

Section 2 : Variables and data types

Section 3: Operators

Section 4: Flow control statements

Section 5: Object and arrays

**Section 6 : Function**

**Section 7: Arrow function**

**Section 8: variable prefixes**

**Section 9: Interfaces**

**Section 10: classes**

**Section 11: Inheritance**

**Section 12: Access modifier , encapsulation and static**

**Section 13: polymorphism**

**Section 14: type casting**

**Section 15: Modules**

**Section 16: more types**

**Inheritance :- given example of parent class and child class**

**Encapsulation : private member variable and method but we can access member variable using method**

**Polymorhism:- parent class and child class and we can access child class method using parent class reference**

**Section 1: Introduction**

**Typescript :** is superset of java script or JS++

**TypeScript** is a super set of JavaScript. **TypeScript** builds on top of JavaScript

**TypeScript** is a strongly typed, object oriented, compiled language.

++: Type safe and easy

All the typescript code is trans compiled into java script code by using type script compiler

Compiled

TS----------------------------------JS

TSC

**Note**

A the end browser only understand javascript

During the compilation time we are ensuring type safety

**Advantage**

Type safety

Typescript bring oops in jS

**Typescript playground**

Typescript is strictly typed

Graphical user interface, application

Description automatically generated

**Note :**

We always specify the type

<https://www.typescriptlang.org/play>

Graphical user interface, text, application, email

Description automatically generated

Typescript is strictly typed and it is superset of javascript

Whatever we write in typescript that

**Install typescript**

Step 1: download and install it node js

<https://nodejs.org/en/download/>

**NPM :-**

Node package manager come with node js

Search npm install

**Npm :-**

is use to install various module in javascript

Text

Description automatically generated

**Step 2:**

**Npm install -g typescript :**

that will install latest version of typescript. Tsc(typescript compiler) we use for compile.

Graphical user interface, text

Description automatically generated

Section 2 : Variables and data types

**Introduction**

**Variables :--**

are used to store retrieve data in our application

Instead of remembering memory location. It will act as container for data.

To define a variable of certain type we use **var**

**Data types**

Number var x:number =123 ( it can carry both integer and decimal value)

String var s:string="saurabh"

Boolean var b:boolean=true

Any var a:any="saurabh" ( any is used to carry any type of data)

Enum var e={male,female} ( enum is sequence of constant value)

**Note :**

We should not use any keywords as variable names

**First program using VS code**

The extension of typescript file is **ts**

A screenshot of a computer

Description automatically generated with medium confidence

When we compiled typescript file it will transpiled the file create javascript file and put it in same folder

What is a transpiled language?

Transpiling is a specific term for **taking source code written in one language and transforming into another language that has a similar level of abstraction**

Imp:

Rn this command : **set-executionpolicy remotesigned**

<https://stackoverflow.com/questions/58796490/tsc-ps1-cannot-be-loaded-because-running-scripts-is-disabled-on-this-system>

Text

Description automatically generated

Will use this JS file inside the html and see the output

**Use JS in a html**

A screenshot of a computer

Description automatically generated with medium confidence

**Output :**

open html and see output

Graphical user interface, text, application

Description automatically generated

We successfully use transpiled javascript in html

**Strings**

A screenshot of a computer

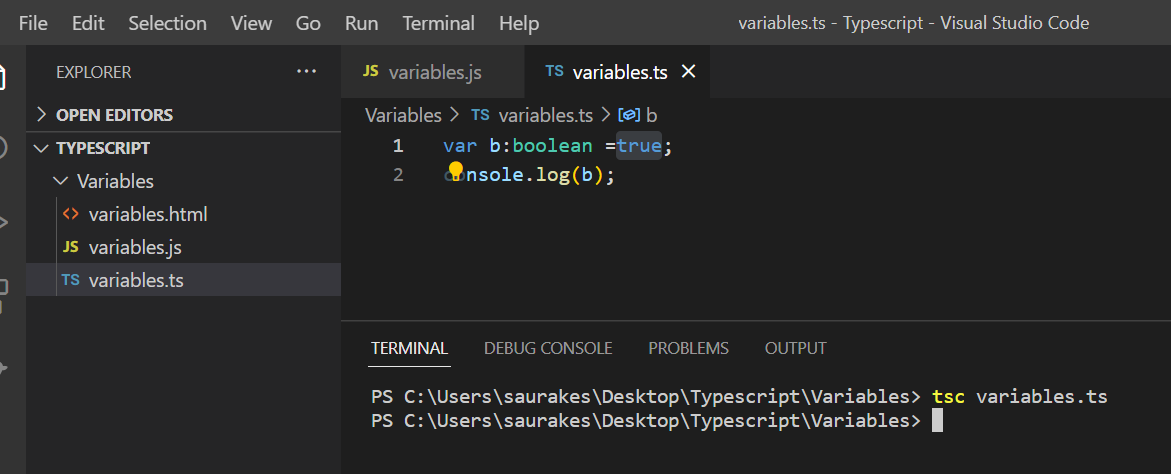
Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

**Boolean**

In boolean we can assign true or false only



//output

Graphical user interface, text, application, email

Description automatically generated

**Any type**

Using any data type we can assign any type of data to a variable.

Even json also we can assign

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

**Homogeneous Array**

**Array is a property**

A screenshot of a computer

Description automatically generated with medium confidence

**Output**

Graphical user interface, text, application, email

Description automatically generated

**Heterogenous array**

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

**Using alert, confirm and prompt**

**Console.log :--** console is inbuilt object and log is function on inbuilt object

**Alert** : display a pop up message to end user.

**Confirm** : display a message and user to confirm particular action

**Prompt** : display a pop up box and prompt user to enter some info

Note :

We can assign prompt to a variable.

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

**Note :**

Prompt allow string data type , we can convert it to other data type also as well.

**Comments**

Single line comment **:- //**

Multi line comment : /\* \*/

Comment in typescript are similar to comment in javascript

**Enum type**

Collection of constant values

We generally use as capital letter in enum, it is just a convention

**Enum cannot be changed but they can access as read only.**

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

These are index value which automatically assigned to enum

**To access value**

Text, chat or text message

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**We want to change index:**

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

**More about Strings**

**\n \t \v :** that will add new line , tab space, vertical space

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

**String Function**

Typescript support all the string properties that javascript has

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

**Union Type**

We can combine different data type using pipe symbol

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

**Section 3: Operators**

**Arithmetic**

**+ - \* / %**

Graphical user interface, text, application

Description automatically generated

Tsc arithmetic.ts

Graphical user interface, text, application, email

Description automatically generated

**Assignment**

A screenshot of a computer

Description automatically generated with medium confidence

**//output**

Graphical user interface, text, application, email

Description automatically generated

**Comparison**

**===** to check two values are same

!==

<

>

<=

>=

Comparison operator will evaluate into boolean **true or false**

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

**Logical**

**&& :** evaluate true if both expression are true

**||** : true if one of the side is true

**!** : reverse logical expression

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

**Ternary**

**testExpression ? Value1: value2**

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

**Section 4: flow control statements**

**If else**

If(condition){

}

Else{

}

A screenshot of a computer

Description automatically generated with medium confidence

// output:

Graphical user interface, text, application, email

Description automatically generated

**Switch**

Switch statement is much more readable then if else ladder

Text

Description automatically generated

// output

Graphical user interface, text, application, chat or text message

Description automatically generated

**Break and case flow**

It gd to use break after each case

If we have common logic for multiple case then use like this

Text

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

**While loop**

**While(condition)**

**{**

**Body;**

**}**

The code will get executed as long as this condition is true

A screenshot of a computer

Description automatically generated with medium confidence

Text, table

Description automatically generated with medium confidence

Program 1:

Validate email id:

**logic is** : to check it contain @ and .

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Program 2**

**Validate a given password**

**Logic :** check password start with capital letter

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Section 5: Object and Arrays

**Introduction:**

Typescript is object oriented programming language

**Object representation help us to map real world problem to a s/w solution**

**Example**

**Amazon**

Product

Shopping card

Order

**Object literal**

In typescript everything internally is an object

The literal syntax of creating object is very similar to java script.

**These values** here are **property of object** and **these values** can be even **function**

**Syntax to declare an object** and access it properties

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, chat or text message, email

Description automatically generated

**For in loop**

It works great against object and it loop through various property in object.

**To get property and value in object, we can use for in loop**

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

**Arrays**

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

**De- structuring Arrays**

It always start from left element .

Helm us to put array into individual variable

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

Here in destructuring array we can use any property

**De- structuring objects**

A screenshot of a computer

Description automatically generated with medium confidence

**In case of objec**t we should use the exact property name

Graphical user interface, text, application, email

Description automatically generated

**Array Function**

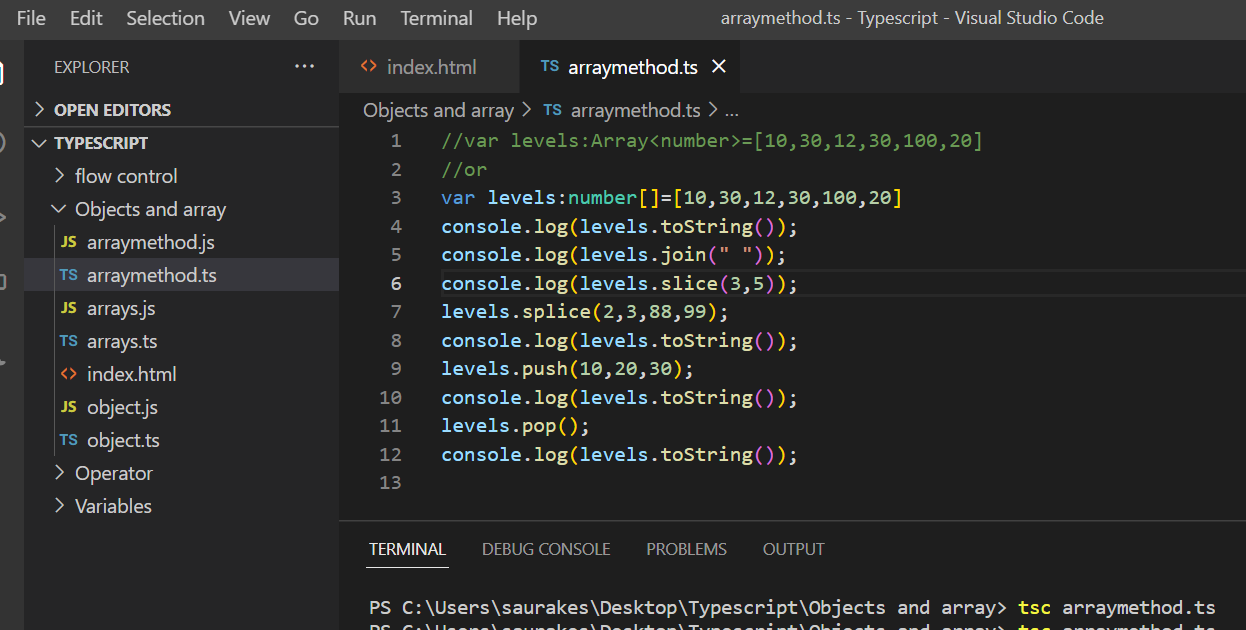
**toString() :** will return **string representation of given array**

Join() :-- it will return a string but it will use separator whatever separator we will pass it

Slice() : it take the start and end and it will cut the array

Splice() : start element, delete element and insert element

Example (2,3,88,99) start from index 2 and delete index 2,3,4 and insert 88 and 99



Graphical user interface, text, application

Description automatically generated

**Section 6 : Function**

**Reuse**

Function functionName(){

Body

}

**First function**

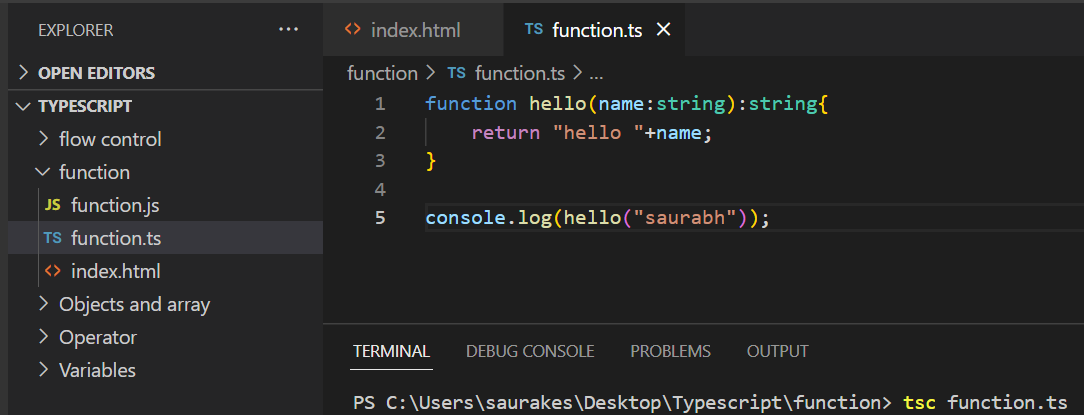
A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

**Passing a parameter**



Graphical user interface, text, application, chat or text message

Description automatically generated

**Passing multiple parameter**

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, chat or text message

Description automatically generated

**Optional parameters**

**Requirement :**

function don’t need all the parameter to work(that is not required to pass all parameter)

that is where optional parameter come in.

A screenshot of a computer

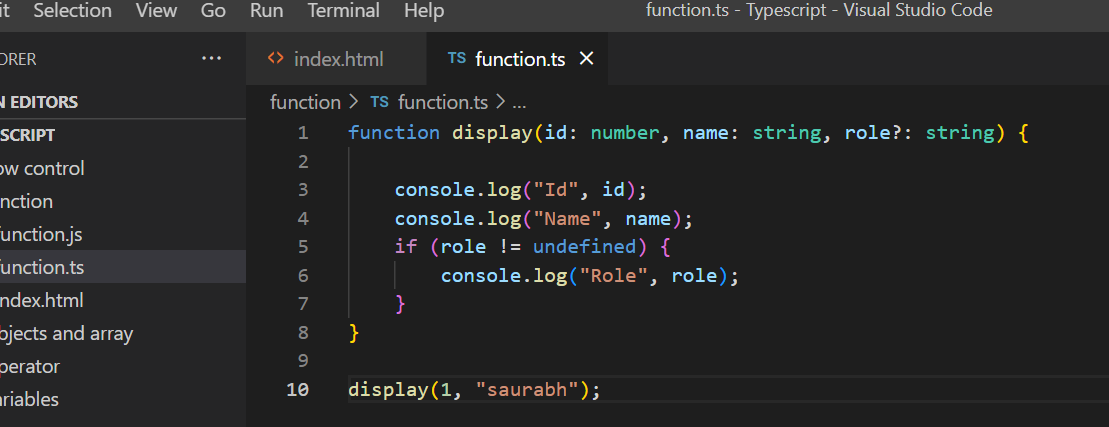
Description automatically generated with medium confidence

We need to use question mark and here role will come as **undefined**.

Therefore we need to put a check to overcome this

Graphical user interface, text, application, chat or text message

Description automatically generated



Graphical user interface, text, application

Description automatically generated

**Default values**

How to defined default values to these parameter.

By default role value will be normal, but if we pass role value it will override default value

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

**Function as parameter**

**We can pass add function as a parameter to other function** and that function will invoke other function and make use of it

Example :

We are passing add function as a parameter to calculator function

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

**Returning a function**

A function will return another function which can be invoked or executed.

Return a function from within another function

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Anonymous functions**

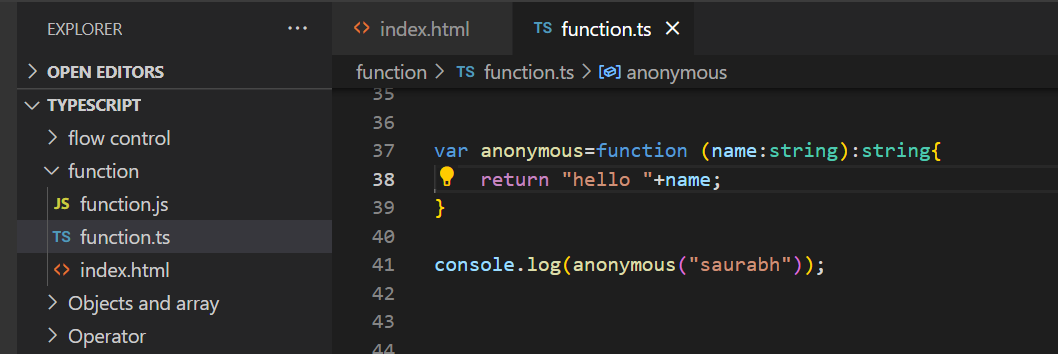
Will convert hello function into anonymous function

Text

Description automatically generated

We need to invoke this function otherwise it is useless.

We need to assign it to variable and invoked that function



Graphical user interface, text, application

Description automatically generated

**Overloading**

**Objective :**

Implementing doubleMe :

Number :---multiply by 2

String :- append same string

**Note :**

We need to restrict doubleMe function to number and string

We can restrict overloaded function to a particular type

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Rest Params**

How to pass any number of arg : **variadic functions (…)** array data structure internally

It will take any number of parameter

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

**Using a type on rest param**

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

**Section 7 Arrow function (like lambda function in java)**

**Arrow function are short cut to use anonymous function without function keyword any braces**

Return is implicit in arrow function but if we want to return it then we need to use curly braces

**Arrow function**

**Var doubleMe =**(num:number) => num\*2;

Var doubleMe = (num:number) => { return num\*2;}

**first arrow function**

A screenshot of a computer

Description automatically generated

Graphical user interface, application

Description automatically generated

**Passing parameter in arrow function :**

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

**Array of arrows function**

**Objective :**

Push 10 arrow function on array by using a loop and then will invoke all arrow function using loop again

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

Section 8 Variable Prefixes

**let**

Let, const, declare :--- that will change scope of variable

Once we use a variable with let it will be available only in that block not outside the block

Let is blocked scope and not present outside block

Text

Description automatically generated

**Const (like final keyword in java)**

Want certain value not to be change in application then use const

It is not supported by all the browser in older version es6

Graphical user interface, text, application

Description automatically generated

**It is just a warning not a compilation issue**

**Const function**

Const is useful in function , we cant define again , it will show warning

Application

Description automatically generated with medium confidence

**Declare;**

Variable that come from external lib , are not present within typescript file are called ambient variable

Section 9 Interfaces

**Introduction**

Interface student{ rules

firstName:string;

lastName:string;

Score:number;

Display():void;

}

Any object that subscribe to this interface or complies with this interface will have all those properties.

It is compile time once js will be created it will be erased.

**Note :**

Above one is object type interface but we can also define interface for arrays, function and classes as well

**Define a object interface**

**Requirement :**

Will create a interface for object called **Product** which is having four field



Using this interface will create object

**Create and object**

A screenshot of a computer

Description automatically generated with medium confidence

**Defining optional paramater**

We can define optional parameter with question mark

Text

Description automatically generated

**Interface are only compile time**

It is mainly for compile time , it doesn't exist at runtime. When js is generated there is no interface

**Function Interface**

**Objective :**

Will learn subscriber function that follow the rules of interface

Define interface for a function

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

**Return type in functional interface**

**Note :**

* Return type is optional when we defined function in interface.
* If we don’t define return type in interface we can return any return type
* Even if we use void as return type in interface we can overwrite it.
* But if we specify a particular return type we cant overwrite it

**Overwrite return type**

Text

Description automatically generated

Return type not defined

Text

Description automatically generated

**Adding method to object interface**

We can define methos also to object interface

Text

Description automatically generated

**Array Interfaces**

We can also create interface for array types

There are two types of interfaces

1:- numeric index

2:- string based index

**Want index to be number and value to be string**

A screenshot of a computer

Description automatically generated with medium confidence

**String indexed Array Interfaces**

Internally it will be represented by object

A screenshot of a computer

Description automatically generated with medium confidence

Item will give key

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated with medium confidence

**Extending interfaces**

Graphical user interface, diagram

Description automatically generated

Instead of defining all the properties in one interface we can define across interface and extend it

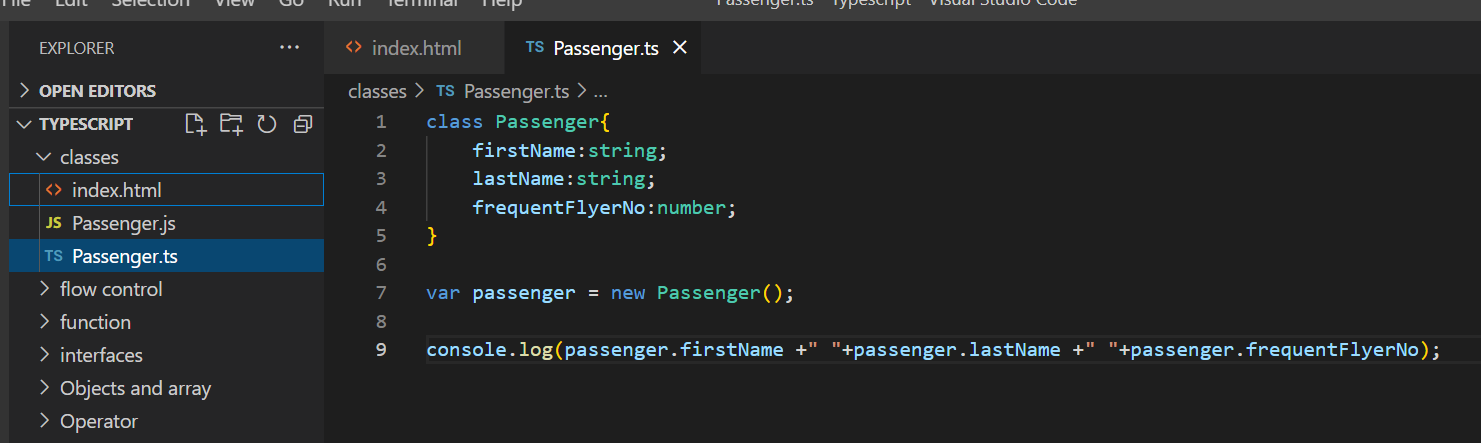
Text

Description automatically generated

Section 10 classes

* Classes are blue print or template to create object.
* It provide rule.
* Classes exists at runtime
* We can create any number of object of this class. We can create constructor, access modifier and all object will get constructor.

**Create a class**



Graphical user interface, text, application

Description automatically generated

**Add a constructor**

Constructor is a function using which we can initialize object property when object is created

It will start with constructor keyword.

Once we create class and constructor we can create any number of object

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

**Add function properties**

**We can also define function inside class**

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

**Using for-in and instance of**

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Requirement :**

We don’t want to display the function , it can be done in multiple ways

Note :

All the function are instance of inbuilt type called **Function**

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Implementing a interface**

**Create an interface and implement using a class**

A screenshot of a computer

Description automatically generated with medium confidence

A picture containing graphical user interface

Description automatically generated

**Section 11 :- Inheritance**

**It is the process of defining a new object with the help of existing object**

Inheritance in Java is **a mechanism in which one object acquires all the properties and behaviors of a parent object**. It is an important part of OOPs (Object Oriented programming system). The idea behind inheritance in Java is that you can create new classes that are built upon existing classes

*Example :*

Iphone x and iphone next

Timeline

Description automatically generated

Child class can access the existing functionality from parent class and if want it can update the functionality as well.

Two imp terms:

Reusability and is-A

All the common code we put in parent class that’s why the term reusable is use

Is-A is a terms , just a r/s

Diagram

Description automatically generated

**Extending a class**

Diagram

Description automatically generated

Text

Description automatically generated

**Create child object**

A screenshot of a computer

Description automatically generated with medium confidence

**Inheriting functionality**

Child class can have both parent and its own functionality

A screenshot of a computer

Description automatically generated with medium confidence

Table

Description automatically generated with low confidence

**Overriding**

**In some cases we don’t want to use parent method in child class**

**Overriding**

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Section 12 : access modifiers, encapsulation and static**

**Access specifier :--Public and readonly**

By default all variables in typescript is public

Don’t want to change value anywhere use readonly with variable

A screenshot of a computer

Description automatically generated with medium confidence

**Encapsulation**

Is a process binding the data and code together

Binding data properties and method properties on object together

Binding data and method together and data can be access only through method

**We cannot access properties directly , we can access it using getter or setter and member variable is private**

Class student{

Private \_name:string;

Get and set

}

Within the method we can do validation of data before assigning to variable(name) to protect our data and making assure

To get correct data through get method.

**Private properties**

**Private field should be access only through accessor method.**

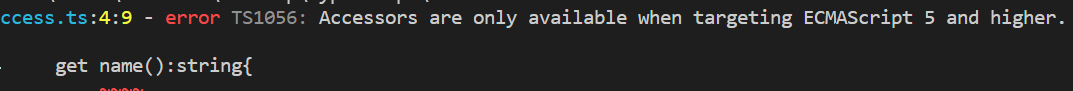
Text

Description automatically generated

Graphical user interface, application

Description automatically generated

**Accessor methods**

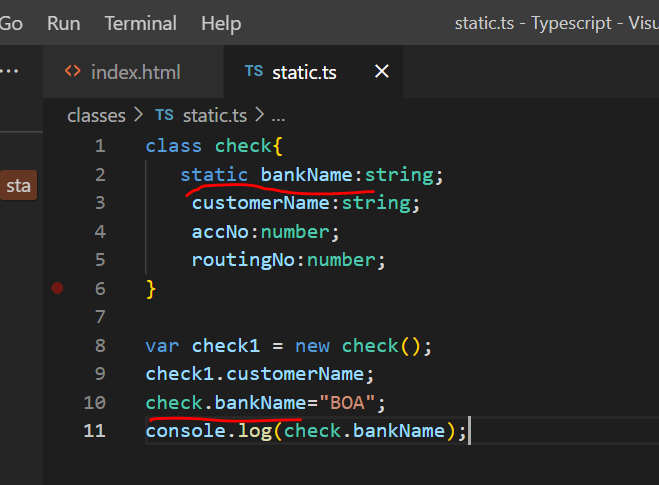




**Using static properties**

Note:

We can access static field only with class name



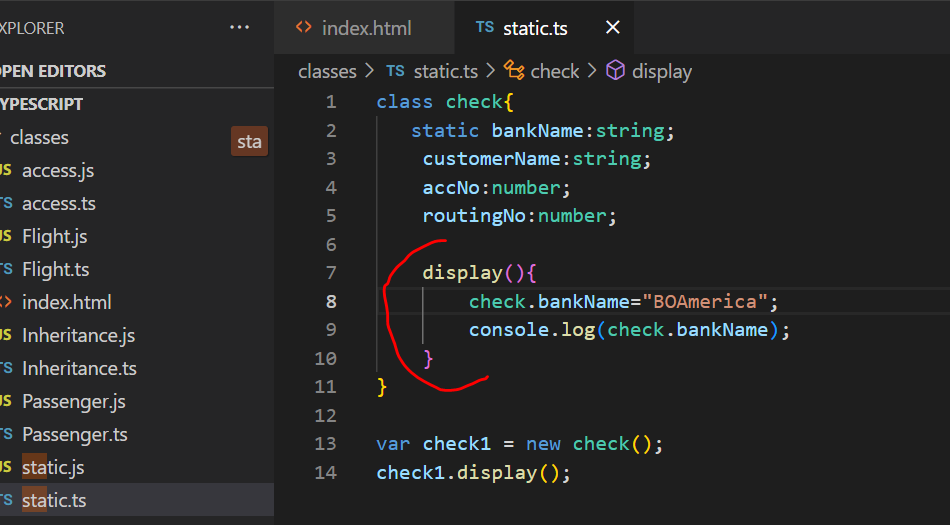
Graphical user interface, text, application

Description automatically generated

**More about static**

**Requirement :-** want to access static field using object

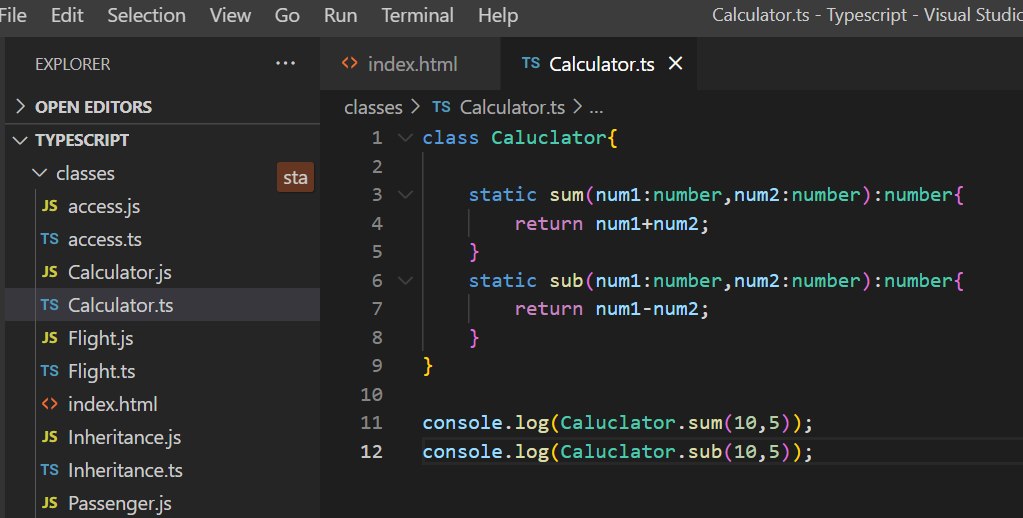
We can create method and there we can do inside class wtever logic we want to write with static field



Graphical user interface, application

Description automatically generated

**Static method**



**Section 13: Polymorphism**

Poly means multi

Morphic means shape

**We can hold child class object by using parent class variable**

**Polymorphism in action**

Parent class holding the reference but depending on child it will invoke that method;

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

Dynamically at run time parent class can point to child class

Section 14 Type casting

**Note :**

Whatever user will enter in prompt will get back as string

parseInt :---cast string into int

**Using the toString method**

**toString :** will represent in string format

Text

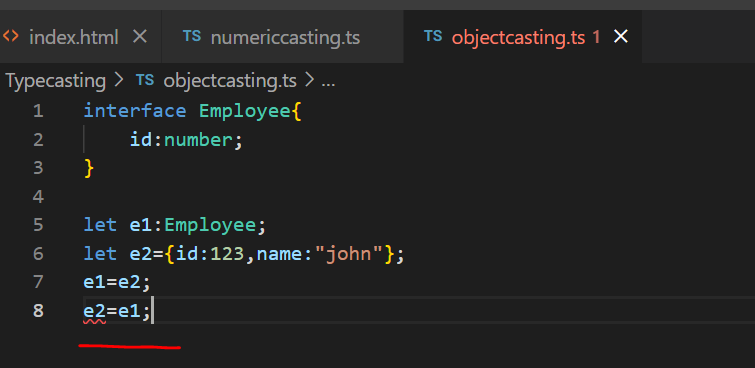
Description automatically generated

Graphical user interface, application, Word

Description automatically generated

**Object casting**

We can only assign data of second variable to the first variable if first variable data or type members contained within the type of second variable



**Section 15:- Modules**

Upto es5 there is no concept of modules

TS uses commonJS lib by default . Starting ES6 it has support of modules.

**Using Function Modules**

**Calc**

**Export** add sub

**Calcuser**

**Import**  add sub

Note :-

Export function can be use in other modules using import

Calcuser :

Now we can use these two function as if they exist in calcuser.module

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

**Import aliasing and alternate export syntax**

How to use alias name with function while importing

1. **Function name as alias in import**

A screenshot of a computer

Description automatically generated with medium confidence

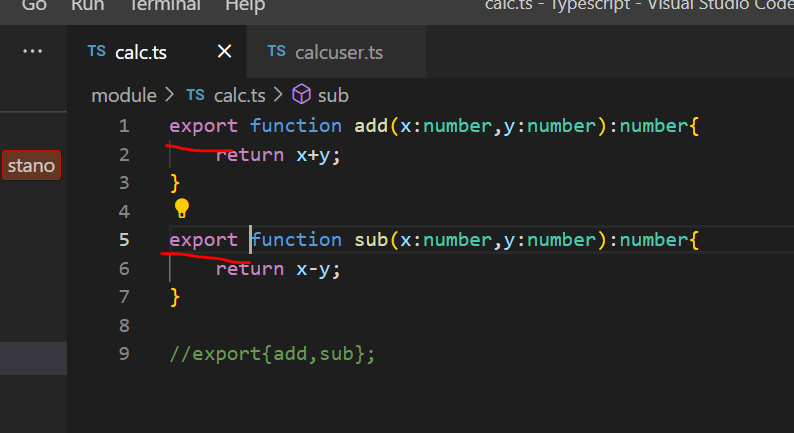
1. **Alias name for entire module we are importing**

Text

Description automatically generated

1. **Alternate syntax of export**

We can also export individual function by using keyword export



**Default exports**

**Default export means without using curly bracket we need to write in import section**

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

**Note**

Only one function can be exported as default, we cannot use default on multiple function

**Class Modules**

We can also use export and import on classes and interface as well

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

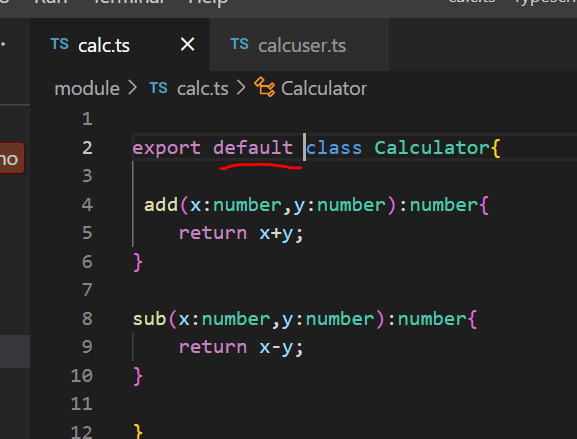
**Aliasing class modules**

**Aliasing class in import**

A screenshot of a computer

Description automatically generated with medium confidence

We can also use default class just like function , we don’t need to write class now in curly bracket in import section



A screenshot of a computer

Description automatically generated with medium confidence

Section 16 - More types

ES6 support map and set

**Map in action**

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**More method on map**

Typescript cant directly deal with iterator

**Array.from** :- will allow to create an array from iterator

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Set**

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Regular expression**

**Allow to validate data in application :** email ,pw ,phone number

It is combination of **metacharacter** and quantifier

**?** Zero or one

**+** one or more

**\***Zero or more

\w include alphanumeric value and also underscore

\d for all the digit or numbers

\s for spaces

\^ a particular string start with

$ a particular string ends with

(?=.\*[A-Z]) atleast once capital symbol from A to Z (.\* means)

Quantifiers

{n} exactly n number of char occur

{m,n} minimum of m char and maximum of m char will occur.

{m,} minimum of m char and maximum of any char can occur.

**Reg ex prog 1**

A screenshot of a computer

Description automatically generated with medium confidence

A picture containing graphical user interface

Description automatically generated

**Reg ex program 2**

**Pw validation :** p/w is having 4 to 10 char

/(?=.\*[A-Z])/ :- any number of capital letter but there should be atleast one

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, application, Word

Description automatically generated

**Date**

A screenshot of a computer screen

Description automatically generated with medium confidence

A picture containing graphical user interface

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application, timeline

Description automatically generated with medium confidence