WHAT IS TYPESCRIPT?

-is not new language, it’s a superset of js. Typescript has additional feature not supported by js

-typscript as –strong typing

-quite a object oriented featured

-catch errors compile time

-great tools,

-Typescript compiles to js for browsers can understand

-first install typescript by “npm install –g typescript” using cmd

-tsc –version to check the version

-create a new folder mkdir ts-hello

-cd into the folder > cd ts-hello

-create a new file by “$code.ts in cmd” or go to VSC and create a file manually

-define a function log that takes a message and log that message

-then declare a global variable declare “hello world”

-then call the log message

-tsc main.ts

-you run the cmd node main.js, you should be able to see “Hello World”

-variable declaration : there are two ways to declare, var or let

-recommend using let

-declare a variable “let count = 5” and when you hover over “count” variable, you will see “let count: number”

-declare a variable “let a;” and when you hover over “a” variable shows “let a: any”

-when we don’t know what the variable a holds we can leave it as “let a;”

-but if you do “let a: number;” and a = 1, a = true, a = ‘a’. We will get compilation errors under “a”

-gives you the error “ type “a” is not assignable to type “number”

-types of declarations

- let a:number;

let b: boolean;

let c: string;

let d: any;

//declare an array of numbers

let e: number[]

-optionally you can initialize this to an array like this let e : number[ ] = [ 1 , 2 , 3 ];

-additionally you can declare any array let f: any[] = [1, true, 'a', false]

-try to avoid this its not good practice.

-another type is enum

-declare a variable and set to a string let message = 'abc';

-when you type “message.” Intellisence gives you lot of options as “anchor,big,bol,charAT,length……”

-its more like a tool tip

-“type assertion”

-lets declare a variable let message = 'abc';

-declare another variable let endsWithC = message.endsWith('c');

-which returns true or false.

-for example let message; is any variable

-declare message and assign it a string,

-declare another variable let endsWithC = message.endsWith('c');

-the above code will not recognize “endsWith” since it was original declared to a string.

-so we need to let typescript know that this a “string” by type assertion

-we need to tell the typescript compiler that this a string, so we use type assertion

-there are two ways to do type assertion

-prefix the variable at the beginning with “< >” and enclose the parent in parethesis let endsWithC = (<string>message)endsWith('c');

-the alternative way for the above is let alternativeWay = (message as string).endsWith('c

-by the way short cut key to copy entire line of code **“Alt + Shift Up/Down”**

-there will a lot of usage of ES6 arrow functions example

- regular JS function let log = function ( message ){

    console.log(message);

}

-ES6 function let doLog = (message) => {

    console.log(message);

}

-if the code is 1 line

-let doLog = (message) =>  console.log(message);

-if there are no parameters

-let doLog = () =>  console.log();

-custom types in typescript

-this is a really bad representation

let drawPoint = (x,y,z,a,b,c,d) => {

    //...

}

Cleaner syntax

let drawPoint = (point) => {

    //...

}

drawPoint({

    x:1,

    y:2

})

This will break if the code gets more example, better way you can use “interface”

interface Point{

    x: number,

    y: number

}

let drawPoint = (point: Point) => {

    //...

}

drawPoint({

    x:1,

    y:2

})

-A Class: Groups variables ( properties ) and functions ( methods ) that are highly related

-When a function is part of a class we call it as method

-class Point{

    x: number;

    y: number;

    draw(){

        //...

    }

    getDistance(another: Point){

        //...

    }

}

class Point{

    x: number;

    y: number;

-the above Point is “class” think of it as “human”

-let point = new Point();

-let point is “object”think of it as “Tom” “Dick” or “Harry”

-Every class can have a constructor a method that is called when we create an instance of this class

-so in the class add method name is “constructor” reserved key word, which can have parameters

-constructor(x: number, y: number){}, x is a number and y is a number

-in this method we can initialize these fields, so this.x = x;

-similarly set this.y = y;

-when you create an object you need to supply these values which is : let point = new Point(1, 2);

-with this you can simply the above code from:

let point = new Point(1, 2);

    point.x = 1;

    point.y = 2;

-to

-let point = new Point(1, 2);

-In Typescript there are 3 “**Access Modifiers**”

-“**public**” , “**private**” and “**protected**”. The most common are public and private

-An “**Access Modifier**” is a keyword that we can apply to a member of a “class” to control its “access” from the outside. By default all members are public

-for example in our “Point class” we have three members: class Point has two fields “**x: number**; and **y: number**”

-and a method “**draw( ){…………………………..}”**

-when you create a point object as in “point and dot” it gives you three options “draw, x and y”

-let point = new Point(1, 2);

    point.

-you will see “**3 members of the class point**” since they are all public

-however if you go back up top and change x:number to private, private x: number;

-in the object “po**int and add dot**” the intelligence will only provide you with “y and draw( )” members

-this is the reason why we use “Access Modifiers” so that we cannot access these members and change them from outside the class.

-**By default** if you **don’t assign an modifier** it will be “**public**”

-typescript has better features for example

- constructor(x: number, y: number){

        this.x = x;

        this.y = y;

    }

-the above code is redundant, this.x is equal to x and this.y equal to y and defined x: number, and y:number

-the above can also be written by prefixing access modifier to constructor.

-constructor(private x?: number, private y?: number){}

-we removed the this.x and this.y and used “?” for any

-these are private so we used “private modifier” if they were public we would use “public modifier”

-a property looks like a field from outside and internally it’s a method in the class. One method it’s a getter or a setter or combination of both get ( ) or set ( )

-**What is Module**?: is a mechanism to a group components, directives pipes and services that are related in such a way that can be combined with other “**modules**” to create and application. In Typescript think of each file as a “**module**”

-we want to move this definition of “point class” to somewhere else in a file called “point.ts”

-in the project create a new file called “point.ts”, back in main.ts select all the “point class” code and paste it in “point.ts”

class Point{

      constructor(private x?: number, private y?: number){}

    draw(){

        console.log('X: ' + this.x + ', Y: ' + this.y)

    }

}

-we created this class point.ts and which is not accessible anywhere, so this file defines its own scope.

-in order for us to use this file from outside we need to export this class to outside. But how?

-add “**export**” keyword to the “**point class**”. From typescript point of view this is a module.

-now we need to go back to “**main.ts**” to use this class “i**mport**” it. But how?

-back in “main.ts” add “import” in curly braces add the name of the type we want to import. Which is point

-if you are importing multiple types you separate them by a “comma” ex: { Point, Output, InIt,……. }

-but we need to put the n**ame of the module** we are importing in quotes ‘ ‘

-name of the module is the rel**ative path of the module**. ‘./point’; automatically typescript will give you location or relative path for you. Now you can create an instance of this “point class” and use it.

-In typescript we divide our program into multiple files, in each file we export one of more files. These types can be classes, functions, simple variables and objects. Wherever you want to use these files you need to import them first. When have an import, export statement on top of the file that file is a module from typescripts point of view. In Angular we also have concepts of modules, but Angular modules are a bit different.

-Building Blocks of Angular Apps

-At the heart of Angular apps there are “Components”. But what is a component?

-Component are logical piece of code for Angular js. It encapsulates the “Data” the “HTML template or mark up’ and the logic for the view. Which is the area of the screen user sees. Example website called “Udemy” you have a navigation bar, sidebar and courses. Now you can think of this as “ one appComponent” or we can split that into “navbar component”, “sidebar component” and “courses component”. Now courses can have multiple courses, then it will component inside components. Then you have ratings inside courses. You can have “rating component” and so on.

-Every application has app component or called “root component”. In real world Angular app will have “root component” and a tree of components.

-Creating Components:

-First you need to “Cr**eate a component**”, second you need to “**Register it in a module**” and third “**Add in an element in an HTML markup**”

-create a new app, called “hello world” by ng new Hello-world, and cd into app, and run ng serve

-**First Step:** create a component by right click on the app folder, select new file and type “courses.component.ts”

-it will be a blank file, we gonna create typescript class, by class CoursesComponent{…………}

-so far to see this class we need to export this, so add export class CoursesComponent{}

-in order change this class to component we need to add some meta data to it. We need to add a decorator. First we need to import the decorator on the top and add the name of the library you are importing from, which is @angular/core

-Then apply to typescript class. @Component( ) which takes a selector,

import { Component } from '@angular/core';

@Component({

    selector: 'courses',

    template: '<h2>Courses</h2>'

})

export class CoursesComponent{}

-this is your basic component, your first step.

-**Second step** is to register this component in the module.

-Go to “app.module.ts” and your “CoursesComponent” under the decorate @NgModule

@NgModule({

  declarations: [

    AppComponent,

    CoursesComponent

  ],

-Whenever you add or register a module under the decorator automatically that module gets imported from the file folder.

-Third Step: the selector of this component selector: 'courses',

-we need to add this element into the html file. Render this by add <courses></courses>