**TYPESCRIPT:**

Typescript is used for type checking. It is used to check all errors related to typescript before runtime.

It is a transpiler or transcompiler based language. It is based on javascript. When we write something in typescript. Typescript compiler turns it into js which then runs into the browser. Js runs in browser, not typescript. In typescript, we write js syntax in the form typed language. By this, all typescript identify error in compiling time. Typescript is the super set of js. It decreases development time by 1/2.

File extension -> filename.ts

**Installation:**

The below will install typescript globally in whole computer.

npm install -g typescript

**Starting:**

Create index.ts file

Then write tsc .\above file name

**Example:**

Tsc .\index.ts

This will create js file

**Create complete project:**

Npm init

Then name of the project

We can change the description entry point etc later

After that, it will create package.json

**For tsconfig.json:**

tsc –init

Like in react, we create components folder and place all of our files in that, in typescript, the best practice is to create src folder and put all src files their. So compiler compiles all the ts files in it.

**Some changes in the ts.config:**

     "outDir": "./dist",                        /\* Redirect output structure to the directory. \*/

     "rootDir": "./src",

After that create src folder. Move index.ts in it. Now we just need to write

tsc

In the vs terminal, that will compile all file in the src folder and generate dist folder containing .js files like index.js

If we want to check output through node, then

Node .\dist\index.js

We can simply create index.html file and use index.js or any .js file that was created from .ts file

**Sumarizing the starting steps:**

1. Npm init
2. tsc –init
3. Some changes in the ts.config:

     "outDir": "./dist",                        /\* Redirect output structure to the directory. \*/

     "rootDir": "./src",

1. create src folder.
2. Index.ts in it.

**To check errors:**

Write tsc in vs terminal

**To check output of .js:**

node .\dist\index.js or file.js

**any Type:**

If we want a type in typescript that can take any value, we use type “any”. Type any is used when want dynamic variable.

let b : number = 10;

// we cannot do this b = "string"

let c : any = 100;

c = "type change due to any";

//as due to any type, we can assign any type to variable c

console.log(c);

Now we can also assign constant type like this. This is like something when we want some serious strict action

let d : true = true;

// d = false; -> gives an error as we assign true type, so it will only accept true

let e : 20 = 20;

// e = 30 or any number; -> gives an error as we assign 20 type, so it will only accept 20

Symbol data type can be used to create symbols

We can also specify certain properties or key value in object data type

let user : {firstName: string, lastName: string} = {

    firstName: "Muhammad Muneeb Waseem",

    lastName: "Waseem"

}

// we can not add more key value pair or more properties in object user as we specify properties initially

console.log(user);

Way of making react app with having typescript:

npx create-react-app quiz-app --template typescript

Continuation of class 16 after completing project quiz app:  
we can define out own type in typescript, which we can later use

type age = number;

let personAge : age = 55;

similarly

ype Person = {

    name : string,

    age : number,

    email? : string //optional property, as declare with ?

};

Email is a option property, as used ? with it

This person type is used in human object

let human : Person = {

    name:"muneeb",

    age: 20,

    email:"@google.com"

}

So its like custom type as per our need

Array in typescript:

With specifying type with array name, we can only pass elements of that types

let arry: number[] = [45, 100, 20];//we cannot pass boolean or string in it

console.log(arry)

//or

/\*let arr: number[]

  arr[0] = 32

  arr[1] = 100

  \*/

//tupple in typescript

Tupple in Typescript:

With tupple, we can define type for each index of an array. With this, we can use elements of more than one tup;e in an array

//in tupple, we can define type for each index

let tup: [string, number, string] = ["karachi", 100, "Pakistan"];

console.log(tup);

readonly type insures that no changes can be made in typescript or react script

Enum in typescript:

If we don’t pass any value keys, then first key or property will have 0 value, then second will have 1 and so on

enum language {

    English,

    Urdu,

    Maths

}

console.log(language.Urdu)

now passing values to them

enum language {

English=100,

Urdu=250,

Maths=300

}

console.log(language.Urdu)

enum language {

    English=100,

    Urdu=250,

    Maths=300

}

console.log(language.Urdu)

Now this time, we get 250. Enum is not directly available in js.

Functions in typescript:

//We only provide number type parameter

function addition (a: number, b:number) {

    return a + b;

}

console.log(addition(2, 5);)

//here we also specify the return type, which is number

function addition (a: number, b:number): number{

    return a + b;

}

console.log(addition

//creating function type, that can be used in any function

type Greet = (num1: number, num2:number) => number

//lets use this type in some function

let add2 : Greet = (num1 : number , num2 : number): number => {

    return num1 + num2

}

console.log(add2(10, 2));

Generic Data type:

//generic type

//we can use it with any type, can be number, string etc.

function personal<T> (a : T): T{

    return a;

}

console.log(personal<number>(10));

console.log(personal<String>("muneeb"));