Type Script Fundamentals :-

→ It is superset of javascript with new features.
→ strong typing
→ OOP's concept
→ Compile type errors are there

Typescript —Transpile→ Javascript {for browsers}

* Installation of TS

```
npm install -g typescript
tsc --version
```

to run `tsc main.ts` {this will transpile .ts code}
`node main.js`

→ Variables :-

two ways to define variable

```
var <name of variable> = value;
let <  "  "  > = " ;
```

* When "var" is used to declare, its scope is given to nearest function
* when "let" is used scope is minimized

→ Types :-

```
let a: number → numbers
let a: boolean → T/F
let a: string → " "
```

let d: any
let e: number[ ] → [1, 2, 3]
let f: any [ ] → array of any type

→ enum <var name> [data = value, ---- }
key.

→ let <var> = < >. key to fetch the value.

* Type Assertions :-

To define data type to a variable to use it as preffered.

(< type> variable)
Eg (< string> name) → will be as a string.

* Arrow functions :-

let < function name > = (Args) =>[Logic }

* Interfaces :-

To use custom types in typescript while accepting data.

→ Example :-

```
Interface Point {
        x: number,
        Y: number
        }

let draw = (point: Point) => {
                //...
        }
```

- now this function will only expect type as a Point which is defined as an interface.

* Class in TS :-

To follow the property of cohesion, same type of fuctions & data must be part of same unit. This unit is known as "class"

Example

```
class Point {
        X: number
        Y: number
        draw (): {
        }
        get destance (another: Point)
        {
        }
}
```

this function will use class vars

this fuction takes point as class input.

__/__/____

→ Objects :- (instance)

It forms an image of a class with customized data.

```
let point = new Point()
```

point → object variable

Point() → class

* Now we can use this "object" variable for using classes functions

* to use variable of a class inside "this.<variable>" is used to refer to the variable

→ Constructors in a class :-

When we want to assign some arguments to a class object when its being initialized, constructors are used.

Example

```
class Num {
    x : number.
    constructor (num : number)
    {
        this.x = num
    }
}
```

assigning the value to x ←

let x = new Num(3) { this will set x = 3 }

→ & to define that wether constructor data is optional, we can add question mark after variable

example :-

constructor (num ? : number)

↑ this question mark

* Access Specifies :-

Private :- to define that variable or fuction can be used only within the class

Public :- to define that variables or fuction can be accessed anywhere.

Example Private x : numble

Public draw () {

}

↳ this is a public fuction

* Access Modifiers in Constructor :-

```
class Point {
    constructor ( private x? number, privat Y: numb)
    {
    }
}
```

→ This will automattical assign passed values to the variable which are private in constructor and can be used within class

→ No need to re-assigning the values

* Properties of Class :-

1.) Getter :-

for getting a value with scope till class only, outside the class we use "iget" function for the value

```
private _x = 10;

get x() {
    return this._x
}
```

2.) Setter :-

In the same manner to set value of a variable

```
set x (value)
{
    this._x = value
}
```

→ Modules :-

As we cannot define all the classes & code in single file, we can export these classes & can be imported from file.

```
export class Point {
    //..
}
```

& to import

```
import { Point } from "Location of .ts"
```