**Lesson20 Interface in Angular**

**Notes:-**

**1-If a class that implements the interface fails to provide implementation for all the interface members, the language compiler raises an error alerting the developer that something has been missed.**

**2-TypeScript is strongly type language, this means that the variable we define must assign the type to it.**

**(Has several built-in pre-defined types like string, number, Boolean etc.)**

**3-Use interface keyword to create an interface**

**4-It is common to prefix the interface name with capital letter "I". However, some interfaces in Angular does not have the prefix "I". For example, OnInit interface**

**5-Interface members are public by default and does not require explicit access modifiers. It is a compile time error to include an explicit access modifier. You will see an error message like - public modifier cannot appear on a type member.**

**6-A class that implements an interface must provide implementation for all the interface members unless the members are marked as optional using the? Operator**

**7-Use the implements keyword to make a class implement an interface**

**8-TypeScript interfaces exist for developer convenience and are not used by Angular at runtime. During transpilation, no JavaScript code is generated for an interface. It is only used by Typescript for type checking during development.**

**9-To reduce the amount of code you have to write, consider using short-hand syntax to initialize class properties with constructor parameters**

**Implement of the Interface and the class**

export interface IEmployee {

code: string;

name: string;

gender: string;

annualSalary: number;

dateOfBirth: string;

// To make a property optional use a ?

// A class that implements this interface need

// not provide implementation for this property

department?: string;

computeMonthlySalary(annualSalary: number): number;

}

export class Employee implements IEmployee {

// All the interface mandatory properties are defined

public code: string;

public name: string;

public gender: string;

public annualSalary: number;

public dateOfBirth: string;

// The above class properties are then initialized

// using the constructor parameters. To do something

// like this, TypeScript has a shorthand syntax which

// reduces the amount of code we have to write

constructor(code: string, name: string, gender: string,

annualSalary: number, dateOfBirth: string) {

this.code = code;

this.name = name;

this.gender = gender;

this.annualSalary = annualSalary;

this.dateOfBirth = dateOfBirth;

}

// Implementation of the interface method

computeMonthlySalary(annualSalary: number): number {

return annualSalary / 12;

}

}