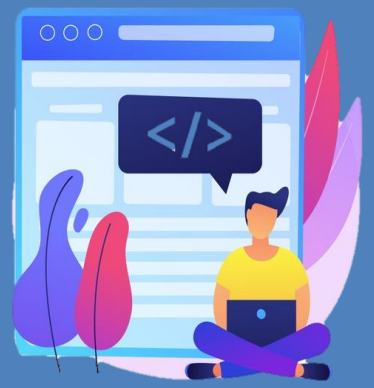


ANGULAR 7

Agenda

1. Angular JS to Angular 7
2. Typescript
3. Angular7 Components
4. Angular7 Modules
5. Angular7 Decorators

greatlearning
Power Ahead

AngularJS to Angular 7

AngularJS to Angular 7

- Angular JS – Launched in 2009, - not a perfect framework back then because of its large size, performance issues and other technical problems.
- It became a motivation behind the entire new versions of Angular - Google developers decided to use really powerful libraries and new features in the next versions.
- Angular 2, 4, 4.3, 5, 6 - was written in Typescript, which is a popular, typed superset of JavaScript introduced by Microsoft.
- Google did finally release the Angular 7 on 18th October 2018 - New update and upgrades were improvements to Angular Material and the core framework, CLI with synchronized major versions, and upgrades to the toolchain.
- The Angular JS 7 version is primarily focused on the Ivy project, which has been going on since past release.
- The Ivy project is basically rewriting the Angular compiler and runtime code to make it better, faster, and smaller.

New Updates - Angular 7

- CLI Prompts
- Angular material & component dev kit (CDK)
- Drag and Drop
- Virtual Scrolling
- Application performance improvements
- Ivy progress
- Documentation updates
- Dependency updates

greatlearning
Power Ahead

Typescript

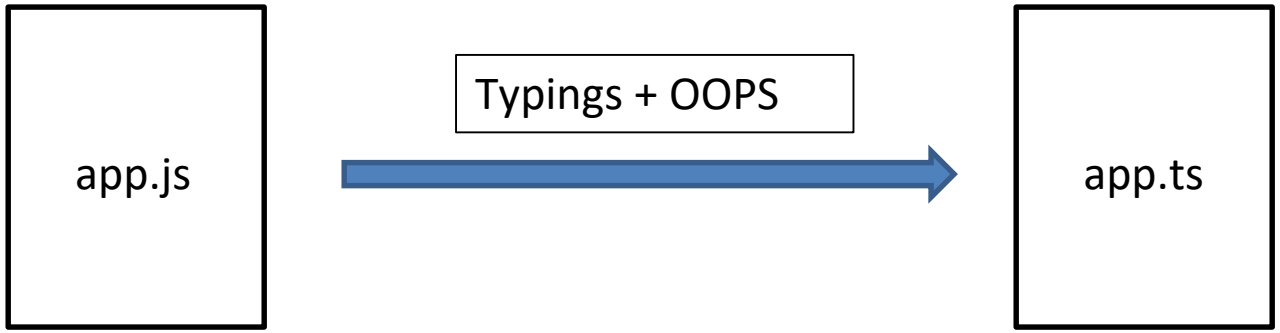
What is Typescript?

1) TypeScript is a free and open source programming language.

2) It is syntactical superset of JavaScript and adds optional static typings and class based object oriented programming to the language.

3) It is developed and maintained by Microsoft.

4) JavaScript file has **.js** extension and TypeScript file has **.ts** extension



Why Typescript?

1. Typing makes code less prone to error

```
let num = 10;

num = "hello"  //Allowed
```

```
let num: number = 10;

num = "hello"  // Not Allowed
```

2. OOP concepts make programming easier

- Classes & Objects
- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

Datatypes

TS demo.ts

```
1   let text: string = "abc";
2
3   let id: number = 2;
4
5   let flag: boolean = true;
6
7   let cars: Array<string> = ["bmw","audi","jaguar"];
8
9   let data: any;
10
11
```

Classes and Objects

TS car.ts

```
1   class Car{
2
3       private speed: number;
4       private color: string;
5
6       constructor(speed: number, color: string){
7
8           this.color = color;
9           this.speed = speed;
10
11      }
12
13  }
14
```

```
let car1 = new Car(200,"red");

let car2 = new Car(400,"blue");

let car3 = new Car(300,"white");
```

Interface

1) When we have abstract information about an object then we can use Interface to represent it.

2) An Interface cannot have a method definition.

TS flyable.ts

```
1   interface Flyable{
2
3       speed: number;
4       acceleration?: number;  //optional property
5
6       fly(speed: number): void;
7
8   }
9
```

Interface

Interfaces are created so that they can be implemented by other classes, promoting code reusability.

TS aeroplane.ts

```
1   class Aeroplane implements Flyable{
2
3       speed: number;
4
5       fly(speed: number){
6           console.log("Flying");
7       }
8
9   }
10
```

Generics

1) Generics allows you to have only one type of object in an array.

2) Generics helps in less error prone code.

demo.ts

```
1  let array : Array<any> = [1,"abc",true];
2
3  let numbersArray : Array<number> = [2,4,4];
4
5
6
```

greatlearning
Power Ahead

Angular7 Components

Angular7 Modules

Angular7 Decorators

What are Decorators?

- The name of the decorator starts with @ symbol followed by brackets and arguments, since decorators are just functions in Typescript.
- Decorators are simple functions that supply metadata to Angular about a particular class, property, value or method.
- Decorators are invoked at runtime.
- It allows you to execute functions. For Ex: @Component executes the component function imported from Angular7.
- When you configure a component for example, you’re providing metadata for that class that tells Angular that we have a component, and that component has a specific configuration.

Common Decorators

- **@NgModule** - Defines a module that contains components, directives, pipes, and providers.
- **@Component** - Declares that a class is a component and provides metadata about the component.
- **@Injectable** - Declares that a class has dependencies that should be injected into the constructor when the dependency injector is creating an instance of this class (mainly for services).
- **@Directive** - Declares that a class is a directive and provides metadata about the directive.
- **@Input & @Output** - Declares an input and output property that you can update via property binding and event binding.

Types of Decorators

- **Class decorators**, e.g. @Component and @NgModule
- **Property decorators** for properties inside classes, e.g. @Input and @Output
- **Method decorators** for methods inside classes, e.g. @HostListener
- **Parameter decorators** for parameters inside class constructors, e.g. @Inject

@Component Decorator

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'app';
}
```

import the "Component" decorator from the angular/core module.

Decorator (meta data)

Defines the name of the HTML tag.

It holds our HTML template.

It holds css files

Defining a class called AppComponent.

The export keyword is used so that the component can be used in other modules in the application.

Template and Style

Template and styles can be added to a component in two ways.

- **TemplateUrl and StylesUrl** - template and styles are separate files, and their respective paths are mentioned in the component file.

TS app.component.ts ✖

```
1   import { Component } from '@angular/core';
2
3   @Component({
4     selector: 'app-root',
5     templateUrl: './app.component.html',
6     styleUrls: ['./app.component.css']
7   })
8   export class AppComponent {
9     title = 'my-app';
10  }
11
12
```

Template and Style

- **Template and Styles** - template and styles are mentioned in the same file.

TS app.component.ts ×

```
1   import { Component } from '@angular/core';
2
3   @Component({
4     selector: 'app-root',
5     template: `
6
7         <h1> Hello World </h1>
8         <p> Adding template in the same page </p>
9
10    `,
11    styles: [`
12
13        h1{
14          color: blue;
15        }
16
17        p{
18          color: red;
19        }
20    `]
21  })
22  export class AppComponent {
23    title = 'my-app';
24  }
25
```

greatlearning
Power Ahead

Thank You