Curs 3

Modules, Components, Data Binding, Directives, Pipes

I Angular Project Structure

- app folder: contains "modules" and "components" for our Angular application.
- node_modules folder: contains all packages installed for this project
- assets folder: contains resources such as images, styles, icons, etc.
- environments folder: contains the environment configuration constants that help while building the angular application
- favicon.io: the icon appears in the browser tab of our application
- index.html: Basic HTML file
- main.ts: the starting point of our application. it bootstraps/starts the AppModule from the app.module.ts file
- polyfills.ts: this file is used to compile our TypeScript to specific JavaScript methods
- styles.css: global CSS file.
- tests.ts: main test file
- .browserslistrc file: browser compatibility and versions
- .editorconfig: this file deals with consistency in code editors to organize some basics such as indentation and whitespaces
- angular.json: defines the structure of our application. It includes settings associated with our application (+env).

→ karma.conf.js : configuration file for the Karma Test Runner. It is used in Unit Testing
→ package.json : the npm configuration file. All the dependencies mentioned in this file. We can modify dependency versions as per our need on this file
→ package-lock.json : whenever we change something on the node_modules or package.json, this file will be generated.
→ README.md : created by default. it contains our project description
→ tsconfig.app.json : overrides the tsconfig.json file with relevant app-specific configurations
→ tsconfig.base.json : introduced in Angular 10+. It has the same configuration as compared to tsconfig.json file
→ tsconfig.json : TypeScript compiler configuration file. This is responsible for compiling TypeScript to JavaScript so that the browser will understand
→ tsconfig.spec.json : overrides the tsconfig.json file with app-specific ~~unit~~ unit test configurations while running the "ng test" command
→ tslint.json : static analysis tool. This file keeps track of the TypeScript code for readability, maintainability, and functionality errors.

Curs 3

II Decorators

Definition: in Angular, the decorators are used to store metadata about a class, method, property or parameter.

Each decorator has a list of arguments which can be set.

- Class decorator: { @NgModule, @Component, @Directive }
- Parameter decorator: { @Input, @Output }

III Angular modules

Definitions: Angular modules keep application code organized by blocks of functionality and features. A root module acts as the starting point for an Angular application.

Each Angular app has a module (called root) to which we can link other modules called feature modules.

IV Angular components

The components are the main building blocks for an Angular application. Each component is composed of:

- HTML template: what renders on the page
- typescript class: that defines the behavior
- CSS selector: define the tag of component, this will be used to call the component in a template
- CSS styles

V Angular Standalone Components

Components, directives, and pipes can now be standalone. Angular classes marked as standalone do not need to be declared in a NgModule.

Standalone components specify their dependencies directly instead of getting them through NgModules. Standalone components can also be imported into existing NgModules-based contexts.

Advantages:

- lazy loading
- simplify creation of components
- simplify project structure and module management

VI Data binding

Data binding allows to define communication between a component and the DOM.

Types:

1) Interpolation: {{value}}
2) Property binding: [property]="value"
3) Event binding: (event)="method()"
4) Two way data binding: [(ngModel)]="property"

Interpolation refers to embedding expressions into the text between HTML element tags and within attribute assignments. By default, interpolation uses as its delimiter the double curly braces.

Curs 3

Property binding flows a value in one direction, from a component's property into a target element property.

When an event is triggered, the desired method is called in the component.

Two way data binding allows bidirectional dataflow.

VII @Input() and @Output()

Definition: @Input() and @Output() represents common component communication scenarios in which two or more components share information.

1) @Input()

Send a message from parent component to a child component.

2) @Output()

Send a message from a child component to the parent component.

VIII Pipes

Pipes let you declare display-value transformations in your template HTML.

```
{{ title | uppercase }}
```

You can create custom pipes that can manipulate your displayed data in any desired way.

```
@Pipe ( {name: 'addYears', standalone: true} )
export class AddYearsPipe implements PipeTransform {
    transform(value: number): string {
        return value + " years")
    }
}
```

IX Directives

There are three kinds of directives in Angular:

1. Components: directives with a template
2. Attribute directives: change the appearance or behavior of an element, component, or another directive
3. Structural directives: change the DOM layout by adding and removing DOM elements
   a) ngIf: makes a DOM element to be added or removed based on a boolean expression
   b) ngFor: makes a DOM element to be added for how many times the boolean expression is true

X Commands

→ create a new app: ng new app_name
→ create a new module: ng generate module_name
→ create a new component: ng generate component_name
→ install packages (when needed): npm install
→ run the app (by default runs on http://localhost:4200)

```
ng serve
```