Angular basics

# Flow: -

1. Angular starts **MAIN.ts** file

platformBrowserDynamic().bootstrapModule(AppModule)

1. Here, we bootstrap our angular application and pass **AppModule** as an argument

bootstrap: [AppComponent]

1. This AppModule now bootstraps the root component e.g **AppComponent**

# Components

Components are normal typescript class created by decorating it with **@Component** decorator

@Component() consist of JavaScript object

Modules:-

Angular uses components to build we pages and uses modules to basically bundle different pieces like components of your app into packages

in module, we can see which features does your app have and use

**@NgModule** decorator

it has 4 properties:-

1. **declarations : register every component here**
2. **imports : import every module here**
3. **providers: provide services used in app**
4. **bootstrap: root component of app**

you need to register every component in module

DATABINDING

Data binding is basically related to communication. **Databinding = Communication**

Communication between your typescript code of component(business logic) and the template

**Different ways of communication:-**

* **OUTPUT** data from typescript code to html code in template
* **EVENT** data from template (HTML) to typescript
* **TWO-WAY** : user events and output data oth

|  |
| --- |
| output data  Typescript code Template(HTML)  (Business logic)   1. **String interpolation** :- {{ data }} 2. **Property binding** :- [ property ] = "data"   react user (Events) like click button     1. **Event binding** :- (event) = "expression"   **Two - Way - Binding (combination of both) : react user events and output something at same time**   1. **[( ngModel )] = "data"** |

**Event binding :- Passing data as parameter from template to typescript code**

**(click) = onMethodName($event)**

**($event :- give access to event data**

**//typescript file**

**onMethodName(event : Event)**

**{**

**var result = event.target.value;**

Type cast it to html element type :-

**var result = (<HTMLInputElement>event.target).value;**

**}**

**Two-way Binding**

* **FormsModule** is Required for Two-Way-Binding. Add FormsModule  to the imports[]  array in the AppModule.

import { FormsModule } from '@angular/forms';

* Need **ngModel**  directive

Directives

Directives are instructions in the DOM

Custom Directives having templates (in HTML)

Structural directives:- change structure of DOM i.e. it adds element or it does'nt

\***ngIf** : condition statement

**<p \*ngIf="varName; else ngTemplateName">message</p>**

<ngTemplate # ngTemplateName >

<p>message</p>

< /ngTemplate >

**ngTemplate**: - placeholder

**\*ngFor** :- loop through list of elements

# @input() - Pass data from Parent component to child component

Assign alias name to property using @input() :-

**@input('alias name')**

# @output() - Pass data from child component to parent component

To emit event data from child to parent, we use EventEmitter

**EventEmitter** :- Emit your own events

e.g:-

**@output()** userCreated = new EventEmitter<{userName: string, userData: string}>();

Local reference in template

* use #variableName in any html element, It will hold reference to whole html element and all its properties
* you can use local reference anywhere inside your current template only and not within its corresponding typescript file

View Encapsulation

* Styles applied to particular component are not overridden to child components
* Add **Encapsulation** property in @Component directive to override styles to child components

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css'],

**encapsulation: ViewEncapsulation.None**

})

**ViewEncapsulation.None** :- it will apply all parent component styles to child

**ViewEncapsulation.Emulated** : default, parent styles not applied to child

@ViewChild \* - Give access to local reference in typescript file

@ContentChild :- Get access to local reference variable inside ng-content