Angular

Notes and References :-

Test Settings

In VSCode Goto Testing click on ... views -> select second Test View

1 - Angular Framework

What is Angular?

Angular is a component based framework for building structured, scalable and single page web applications for client side.

Why Angular?

Simple to build SPAs	Flexible and Structured applications	(OOPs friendly)
Cross platform and Open Source	Reuseable Code (Services)	Testability (spec.ts)

What is the difference between JIT and AOT in Angular?

Both JIT and AOT are used to compile Angular Typescript components to JavaScript, because browser understands JavaScript not Typescript.

ANGULAR COMPILE	DETAILS
Just in Time (JIT)	Compiles your application in the browser at **runtime**. This was **default** until Angular 8
Ahead of Time (AOT)	Compiles your application at **build runtime**. This was **default** until Angular 9

How an Angular App gets Loaded and started? What are index.html, app-root, selector and main.ts?

Angular is used to create single page applications SPAs:-

- index.html file is that single page, it loads main.ts compiled as main.js
- main.ts is like the entry point, it compiles the web-app and bootstraps the AppModule to run in the browser.
- AppModule file will then bootstrap the AppComponent
- AppComponent or app-root component is the html which you will see finally.

What is a bootstrapped module and bootstrapped component?

When the Angular application starts:-

- the `first module launched` is the `bootstrapped module`.
- same way the `first component launched` is the `bootstrapped component`.
- We can change that using `main.ts` and `root module`

3 - Components and Modules

What is a component?

A `component is a basic building block` in Angular application. we use `@Component decorator with the metadata` to define a component.

```
@Component({
  selector: 'app-root', //identify each component uniquely
  templateUrl: './app.component.html', //Template is a HTML view
  styleUrl: './app.component.css',
})
```

It may consist of :-

app.component.ts app.component.html app.component.css app.component.spec.ts app.module.ts

What is a selector and template?

A `Selector` is used to `identify each component uniquely` in the component tree.

A `Template` is a `HTML view` of an Angular component.

What is a module? What is app.module.ts file?

`Module` is a place where you can group the `components, directives, pipes and services`, which are related to the application. We can define a module by @NgModule decorator with the metadata (e.g `declarations`, `imports`, `providers` and `bootstrap`).

```
// app.module.ts (we call it root module)
@NgModule({
  declarations: [AppComponent], //multiple components can be part of this
module
  imports: [BrowserModule, AppRoutingModule, FormsModule], // We can use the
services of other modules inside this module
  // BrowserModule must be present in the root module
  providers: [], // we place Injectable here for Dependency injection
(mostly services)
  bootstrap: [AppComponent], // Define from which component to bootstrap the
application
})
export class AppModule {}
```

4 - Data Binding

What is Data Binding?

Data binding is a way to communicate between your component and your view.

Types of Data Binding:

There are 3 types of data binding :-

- `Output` String interpolation and Property Binding (e.g. `{{ data }}, [property] = "data"`)
    - Data flow from component to view.
    - String `interpolation deals with strings only` where as `Property binding also deals with strings and non-string` like boolean values as well.
- `Input` Event Binding (e.g `(event) = "expression"`)
    - Data flow from view to component.
- `Two-Way` Two wat data binding (e.g `[("ngModel")] = "data"`)

Notes : -

5 - Directives

What is Directive?

Directives are classes that add additional behavior to elements.

Types of Directive?

`Types of Directives:` There are 3 types of directives :-

- `Structural` - use to make changes in DOM (e.g `*ngIf`, `*ngFor`, `*ngSwitch`), (starts with * or @)
- `Attribute` - use to change appearance or behavior (e.g `[ngStyle]`, `[ngClass]`)(starts with [])
- `Component` - A component is itself is a Directive using its on template. (starts with @)

5 - Decorators and Pipes

What is Decorator?

- Angular decorators stores metadata about a class, method or property
- Metadata is data that provides information about other data (e.g Book have metadata like title, auther and price etc).

What are the types of Decorators?

We have 4 types of decorators

Class Decorators	Property Decorators	Method Decorator	Parameter Decorators
@NgModule	@Input	@HostListener	@Inject
@Component	@Output		@Self

@Injectable	@ContentChild	@Host
@Directive	@ContentChildren	@SkipSelf
@Pipe		@Optional
	@ViewChild	
	@ViewChildren	
	@HostBinding	

What are Pipes? what are the types of Pipes and Parameterized Pipes

- `Pipes are simple functions`, which accept an `input value` and return a `transformed value`
- Angular provide 2 types of pipes (`Built-in Pipes` and `Custom Pipes`)
- Built-in pipes (`lowercase, uppercase, date, percentage, currency, Decimal, slice, json`)
- Example use of pipe `<span>{{ title | lowercase }}</span>`
- When we pass any parameters to the pipe, it is called `parameterized pipe`
- Example use of parameterized pipe `<span> {{ 1234.56 | currency: 'INR' }}</span>`

What is Chaining Pipes?

- Chaining pipes means using multiple pipes on a data input.
- Example use `{{ dob | data | uppercase }}

6 - Services and Dependency Injection

Explain Services with example?

- A Service a `reusable code` which can be used in multiple components.
- A Service can be injected other components using dependency injection.

How to create a Service in Angular?

```
ng g service logging
```

```
import { Injectable } from "@angular/core";

@Injectable({
  providedIn: "root",
})
export class LoggingService {
  constructor() {}
  log() {
    console.log("here it is");
```

```
  }
}
```

How to use Dependency Injector with Services in Angular?

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrl: './app.component.css',
  providers:[LoggingService]
})
export class AppComponent {
  constructor(private loggingService: LoggingService ) {
    this.loggingService.log();
  }
  clickMe(a: number) {
    alert('I am clicked : ' + a);
  }
}
```

What is Hierarchical Dependency Injection?

We can set the provider of a service at three places (e.g. `Component, Module and Injectable`)

```
@Injectable({
  providedIn: 'root'
})
```

- Component provided service will be available to Components and its children.
- Module provided service will be available to all the components of that module plus their children.
- `@Injectable({ providedIn: 'root'})` is available globally.

What is provider?

- Provider is an object declared inside Decorators
- It informs Angular that a particular Service is available for injecting inside the component.

Role of @Injectable Decorator in a Service?

- It declare the class as Service.
- Also we can use `providedIn: 'root'` of @Injectable to declare the service as globally accessible.

7 - Lifecycle Hooks and Decorators

What are Parent-Child components?

A component have multiple child components.

What is @Input Decorator? How to transfer data from Parent to Child component?

- `@Input` decorator is used to pass data from parent component to child component.
- We need to bind a property in child components like `<app-child [someParam]="txtInput"></app-child>`
- Now use it in child component with `@Input() someParam: string | undefined`

What is @Output Decorator and Event Emitter?

`@Output decorator and event emitter together` are used to pass the data from child component to parent component.

What are Lifecycle hooks in Angular?

A component from its creation to destruction goes through several stages and these stages are called the lifecycle hooks.
Component instantiating, Rendering the component view and destroying the component.

What is a Constructor?

- A Constructor is a method in a class that automatically gets called when the class is instantiated.
- Constructor always run before any Angular hook and it is not a part of lifecycle hooks.
- Constructor is widely used to inject the **dependencies (Services)** into the component.

What is ngOnChanges lifecycle hook in Angular?

`ngOnChanges` interface detects the changes made to only `@Input` property, we can track the changes by `SimpleChanges`

What is ngOnInit lifecycle hook in Angular?

- `ngOnInit` hook is called when the component is created.
- `ngOnInit` called only once during component's lifecycle.
- It is a the most used hook
- We use it to load data, setting some configurations etc. on initializing the component.

What is the difference between Constructor and ngOnInit hook?

ngOnInit	Constructor
ngOnInit is Angular lifecycle hook which is called when the component is created	The Constructor is a method in Typescript class that automatically gets called when the class is being instantiated
ngOnInit is called after ngOnChanges hook	Constructor is called before any lifecycle hook
When ngOnInit called, everything about the component is ready, so it is used to perform	When constructor is called, everything in component is not ready, so its mostly used for **injecting dependencies** only

most
of the **business logic** on component

---

9 - Routing

What is Routing? How to Setup Routing?

- Routing helps in navigating from one view to another view with the help of of URL
- To Setup Routing
  - Create Angular project with routing option
  - Create components for routing
  - Configure Routes.
  - Configure Links.

What is router-outlet?

`router-outlet` is place where your components will be displayed based on the routing URL.

- Example `<router-outlet></router-outlet>`

What are routerLink?

- `routerLink` is used for navigating to a different route.
- Example `<a routerLink="/component1">Component 1</a>`

What are lazy loading?

- To optimized the app load speed we use a technique called lazy loading.
- Lazy loading provide us means of loading less used component on demand.

10 - Observable / HttpClient / RxJS

What is Asynchronous operations?

- JavaScript is a single threaded runtime, your tasks run in parallel.
- Running tasks in parallel are called Asynchronous operations.
- We use Promises and Observables to achieve this.

What is difference between Promise and Observable?

Promises and Observables both are used to transfer the data asynchronously.

Observable	Promises
Emit multiple values over time, also called streaming data	Emit a single value at a time
Lazy - not executed until we subscribe to them using subscribe() method	Eager - execute immediately after creation
Subscriptions are cancellable using unsubscribe() method	Are not cancellable

What is RxJS?

RxJS is JavaScript library that allow us to work with asynchronous data stream with the help of observables.

- Observables are introduced by RxJS library.
- RxJS stands for Reactive Extensions for JavaScript.

Observable	Observer
Stream of Data	Subscriber

What is Observable? How to implement it

Observable are use to stream data to multiple components.

- Import Observable from RxJS Library.
- Create Observable and Emit Data.
- Finally subscribe the Data.
- Example code

```
import { Observable } from "rxjs";
// Create Observable
myObservable = new Observable((items) => {
  console.log("Observable start");
  items.next("a");
  items.next("b");
  items.next("c");
  items.next("d");
});
// Subscribe to Observable
this.myObservable.subscribe((val) => {
  console.log(val);
});
```

What is the role of HttpClient in Angular?

What are the steps for fetching data with HttpClient and Observable?

- HttpClient is built-in service class available in Angular
- Import `HttpClientModule` from `@angular/common/http` to use it.
- Used to performs HTTP requests (e.g GET, POST, PUT, DELETE)
- Receives the response in JSON format mostly.

```
// Inject HttpClient
constructor(private http: HttpClient) {}
// Use HttpClient with Observable
getStudents(): Observable<IStudent[]>{
return this.http.get<IStudent[]>(this.url)
}
```

```
// Receive the response
getStudents().subscribe( data => this.students = data)
```

How to do HTTP Error handling in Angular?

- By using catchError and throwError from RxJS.
- Example

```
// Inject HttpClient
constructor(private http: HttpClient) {}
// Use HttpClient with Observable
getStudents(): Observable<IStudent[]>{
return this.http.get<IStudent[]>(this.url).pipe(catchError((error) => {
  return throwError(() => error)
}))
}
// Receive the response
// getStudents().subscribe( data => this.students = data)
getStudents().subscribe({
    next: data => this.students = data,
    error: errorReceived => this.errorMsg = errorReceived.message
})
// To display error in view
{{ errorMSg}}
```

12 - Angular Forms

What are Angular Forms?

Angular forms are used the handle user's input (e.g Login Form, Registration Form etc)

What are the types Angular Forms?

There are 2 types of Angular Forms

1- Template Driven Forms	2 -Reactive Forms
Most of the code and validation is written and handled in HTML Templates only	Most of the code and validation will be written in component typescript class file

What is the difference between Template Driven Forms and Reactive Forms?

1- Template Driven Forms	2- Reactive Forms
Most of the code and validation is written and handled in HTML Templates only	Most of the code and validation will be written in component typescript class file

Have to add FormsModule in Root module to active it	Have to add ReactiveFormsModule in Root module to activate it
Used when application is simple and have less controls	Used when application is complex and have more controls

How to setup Template Driven Forms?

How to apply Required Field validation in template driven forms?

What is Form Group and Form Control in Angular?

- Form controls are the individual form control.
- Form groups is collection of form controls
- Both are used to track the values and states of the controls.

How to setup Reactive Forms?

How to do validations in reactive forms?

- `Validator` class is used to implement validation in Reactive Forms.