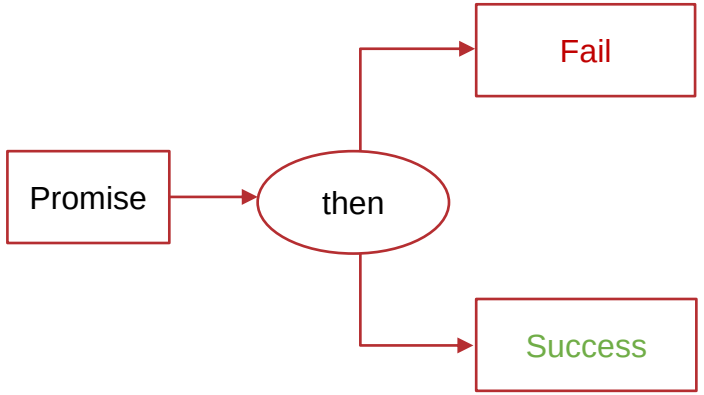
ANGULAR INTERVIEW QUESTIONS

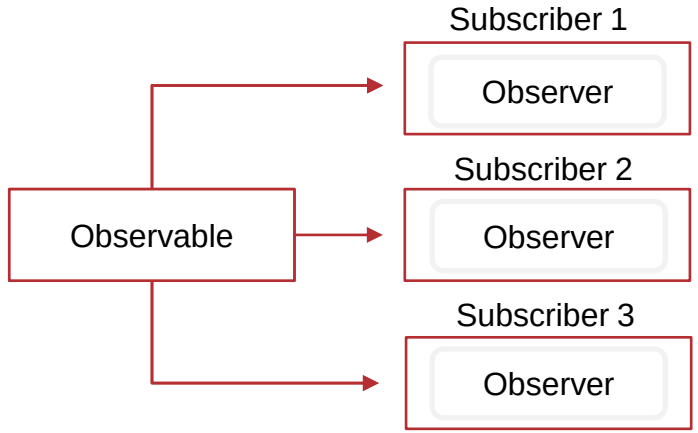
1

Difference between Promise and Observables ?

Promises and Observables both handle the asynchronous call only.
Here are the differences between them:

Promise	Observables
• Emits only a single value at a time • Calls the services without .then and .catch • Cannot be canceled • Does not provide any operators	• Emits multiple values over a period of time • Is not called until we subscribe to the Observable • Can be canceled by using the unsubscribe() method • Provides the map, forEach, filter, reduce, retry, and retryWhen operators





2

Difference between declarations, providers, and import in NgModule ?

Declaration

- Declaration are used to make directives.
- Used to declare components, directives, pipes that belongs to the current module.
- Ex: AppComponent
- Defined in Declarations array in **@NgModule**

```
@NgModule({
    declarations: [ ],
)}
```

Providers

- Provider are used to make services.
- Used to inject the services required by components, directives, pipes in our module.
- Ex: UserService
- Defined in Providers array in **@NgModule**

```
@NgModule({
    providers: [ ],
)}
```

Imports

- Imports makes the exported declarations of other modules available in the current module.
- Used to import supporting modules likes FormsModule, RouterModule, etc.
- Ex: BrowserModule
- Defined in Imports array in **@NgModule**

```
@NgModule({
    imports: [ ],
)}
```

3

Difference between Component and Directive ?

Here are the differences between them:

Component	Directive
• To register a component we use @Component meta-data annotation. • Components are typically used to create UI widgets. • Component is used to break down the application into smaller components. • Only one component can be present per DOM element. • @View decorator or templateurl/template are mandatory.	• To register a directive we use @Directive meta-data annotation. • Directives are used to add behavior to an existing DOM element. • Directive is used to design re-usable components. • Many directives can be used per DOM element. • Directive doesn't use View

4

Difference between Constructor and NgOnInit ?

Here are the differences between them:

Constructor	NgOnInit
• An instance of the component is generated by using the constructor method, which is a typical TypeScript constructor. • Initiating the component's basic properties, injecting dependencies, and setting up class properties are the main uses for it. • It's a standard part of creating JavaScript/TypeScript classes and is not unique to the component lifecycle of Angular. • The dependency injection system of Angular is frequently used in the constructor to inject services, making such services accessible for use within the component. **Ex:** constructor(private someService: SomeService) { // Basic initialization this.property = 'initial value'; }	• The OnInit interface of Angular provides a lifecycle hook called the ngOnInit method. It is triggered just before the first ngOnChanges hook but after Angular has initialized the component's data-bound properties. Directives are used to add behavior to an existing DOM element. • It is often used for more complex initialization operations, like sending HTTP queries, creating subscriptions, and carrying out other operations that call for access to the component's inputs and data bindings. Many directives can be used per DOM element. • ngOnInit is only ever called once throughout a component's lifetime, usually shortly after the component is built and its inputs are bound. **Ex:** ngOnInit() { // Advanced initialization tasks this.loadData(); this.setupSubscriptions(); }

5

Difference between NgModule and JavaScript module ?

Here are the differences between them:

NgModule	JavaScript Module
• NgModule bounds declarable classes only. • List the module's classes in declarations array only. • It only export the declarable classes it owns or imports from other modules. • Only one component can be present per DOM element. • Extend the entire application with services by adding providers to provides array.	• There is no restriction classes. • Can define all member classes in one giant file. • Directive is used to design re-usable components. • It can export any classes. • Can't extend the application with services.

6

Difference between Reactive forms and Template driven forms ?

Here are the differences between them based on feature:

Feature	Reactive	Template-driven
Form Model setup	Created(FormControl instance) in component explicitly.	Created by directives.
Data update	Synchronous	Asynchronous
Form custom validation	Defined as Functions	Need knowledge of the change detection process.
Testing	No interaction with change detection cycle.	Need knowledge of the change detection process.
Mutability	Immutable(by always returning new value for FormControl instance).	Mutable(Property always modified to new value).
Scalability	More scalable using low-level APIs.	Less scalable using due to abstraction on APIs.

7

Difference between ng-template, ng-container, and ng-content ?

Ng-template

As its name suggests, Angular uses the template element ng-template> in conjunction with structural directives (*ngIf, *ngFor, [ngSwitch], and custom directives). These template components only function in the presence of structural directives, which enable us to design a template that only renders elements when necessary and the DOM. It helps in the development of dynamic templates that are flexible and configurable.

```
<div>
Ng-template Content
<div *ngIf=”false else
showNgTemplateContent”>
Shouldn't be displayed
</div>
</div>

<ng-template
#showNgTemplateContent> Should
be displayed
</ng-template>
```

Ng-container

ng-container is an extremely simple directive that allows you to group elements in a template that doesn’t interfere with styles or layout because Angular doesn’t put it in the DOM

This is helpful if you don’t want any extra div on DOM, you can simply use
ng-container. For eg: If there are two structural directives being called on one div as below:

```
<ng-container *ngIf="details">
<div *ngFor="let info of details">
{{ info.content }}
</div>
</ng-container>
```

Ng-content

ng-content is used to project content into Angular components. You use the **<ng-content></ng-content>** tag as a placeholder for that dynamic content, then when the template is parsed Angular will replace that placeholder tag with your content.

Ex: app.component.html

```
<app-detail>

<strong class="app">Senior
Developer</strong>

</ app-detail>
```

Ex: user.component.html

```
<h3><span>Hello, I am </span>
<ng-content select=".app"></ng-
content> </h3>
```

8

Difference between @Inject and @Injectable ?

@Inject and @Injectable both decorators we used for dependency Injection in our angular application.

@Inject	@Injectable
A dependent parameter of a class constructor's "Inject" decorator defines a unique provider of the dependency. The metadata for the elements to be injected must be specified at the constructor parameter level using the @Inject() Decorator. Without it, the type of the parameters is utilized (@Inject(SomeType) obj is identical to obj:SomeType). The class constructor parameter must be injected via the @Inject protocol, which is used by Angular. It is applicable in this way: `@Injectable()` `export class SomeService {` `constructor(@Inject(Http) private http:Http,` `@Inject('sometoken') obj) {` `}` `}`	A class is provided with the @Injectable decorator Injected as a dependency. Assume we have a dependency class called User Details. This class gives us information or some metadata about the dependencies we need to add to the related class constructor. This class decorator doesn't need any parameters. No dependency will be injected without the @Injectable decorator. It may be utilized as: *@Injectable()* *export class SomeService {* *constructor(private http:Http) {* *}* *}*

9

Difference between Pure pipe and Impure Pipe ?

Angular provides us with an organized way to build frontend web apps. One entity that it has is pipes.

Pure	Impure
It works when the component is loaded. We can use like this: @Pipe({ name: 'sort', pure: false }) export class myPipe implements PipeTransform { transform(value: any, args?: any): any { //your logic here and return the result }	It works for every change in the component. We can use like this: @Pipe({ name: 'sort', pure: true }) export class myPipe implements PipeTransform { transform(value: any, args?: any): any { //your logic here and return the result }

10

Difference between ViewChild and ViewChildren ?

In Angular, the decorators ViewChild and ViewChildren are used to access items, components, or directives within a component's view. They are frequently used to communicate with child elements or components in Angular application. Here below are the differences between them:

ViewChild	ViewChildren
• ViewChild is used to query and access a single element, component, or directive that matches the specified selector. It returns the first matching element found within the view of the component. • You can use ViewChild to get a reference to a single instance of a child component or element. • It is declared using the @ViewChild decorator. **Ex:** @ViewChild(ChildComponent) childComponent: ChildComponent;	• ViewChildren is used to query and access multiple elements, components, or directives that match the specified selector. It returns a QueryList containing all matching elements within the view of the component. • You can use ViewChildren to get references to multiple instances of child components or elements that match the selector. • It is declared using the @ViewChildren decorator. **Ex:** @ViewChildren(ChildComponent) children Components: QueryList<ChildComponent>;