### ****39. What is multicasting in Angular?****

In Angular, when we are using the HttpClient module to communicate with a backend service and fetch some data, after fetching the data, we can broadcast it to multiple subscribers, all in one execution. This task of responding with data to multiple subscribers is called multicasting. It is specifically useful when we have multiple parts of our applications waiting for some data. To use multicasting, we need to use an RxJS subject. As observables are unicast, they do not allow multiple subscribers. However, subjects do allow multiple subscribers and are multicast.

### ****46. What are the differences between Angular expressions and JavaScript expressions?****

Angular expressions and JavaScript expressions are quite different from each other as, in Angular, we are allowed to write JavaScript in HTML, which we cannot do in plain JavaScript. Also, all expressions in Angular are scoped locally. But, in JavaScript, these expressions are scoped against the global window object. These differences, however, are reconciled when the Angular compiler takes the Angular code we have written and converts it into plain JavaScript, which can then be understood and used by a web browser.

### ****47. What is server-side rendering in Angular?****

In a normal Angular application, the browser executes our application, and JavaScript handles all the user interactions. However, because of this, sometimes, if we have a large application with a big bundle size, our page’s load speed is slowed down quite a bit as it needs to download all the files, parse JavaScript, and then execute it. To overcome this slowness, we can use server-side rendering, which allows us to send a fully rendered page from the server that the browser can display and then let the JavaScript code take over any subsequent interactions from the user.

### ****48. What is Angular Universal?****

Angular Universal is a package for enabling server-side rendering in Angular applications. We can easily make our application ready for server-side rendering using the Angular CLI. To do this, we need to type the following command:

ng add @nguniversal/express-engine

This allows our Angular application to work well with an ExpressJS web server that compiles HTML pages with Angular Universal based on client requests. This also creates the server-side app module, app.server.module.ts, in our application directory.

### ****50. What are HttpInterceptors in Angular?****

HttpInterceptors are part of the @angular/common/http module and are used to inspect and transform HTTP requests and HTTP responses as well. These interceptors are created to perform checks on a request, manipulate the response, and perform cross-cutting concerns, such as logging requests, authenticating a user using a request, using gzip to compress the response, etc.

### ****29. Describe Angular authentication and authorization.****

The login details of a user are given to an authenticate API available on the server. Once the credentials are validated by the server, it returns a JSON web token (JWT), which includes attributes and the data of the current user. Further, the user is easily identified using JWT, and this process is known as authentication.

After logging in, users have various types and levels of access—some can access everything, while others may have restrictions from some resources. Authorization determines the access level of these users.

**32. How can one create a service in Angular?**

Service in Angular is an object that can be substituted. It is wired and combined with the help of dependency injection. Services are developed by getting registered in a module that they need to be executed in. The three methods of creating a service in Angular are as follows:

* Service
* Factory
* Provider

### ****35. What is REST?****

REST in Angular stands for Representational State Transfer. It is an API that works on the request of HTTP. Here, the requested URL points to the data that has to be processed, after which an HTTP function is used to identify the respective operation that has to be performed on the data given. The APIs that follow this method are referred to as RESTful APIs.

### ****38. What is HttpClient, and what are its benefits?****

HttpClient is an Angular module used for communicating with a backend service via the HTTP protocol. Usually, in frontend applications, for sending requests, we use the fetch API. However, the fetch API uses promises. Promises are useful, but they do not offer the rich functionalities that observables offer. This is why we use HttpClient in Angular as it returns the data as an observable, which we can subscribe to, unsubscribe to, and perform several operations on using operators. Observables can be converted to promises, and an observable can be created from a promise as well.

**28. Explain the MVVM architecture.**

The MVVM architecture plays a significant role in eliminating tight coupling between the components. This architecture includes the following three parts:

* **Model:** The model represents the business logic and data of a particular application. In other words, it consists of an entity structure. The model has the business logic, including model classes, remote and local data sources, and the repository.
* **View:** View is the application’s visual layer that comprises the UI code. The view sends the action of the user to the ViewModel. However, it does not receive the response directly. The view must subscribe to the observables that are exposed to it by the ViewModel to receive a response.
* **ViewModel:** ViewModel is the application’s abstract layer that connects the View and the Model and acts as a bridge between the two. It does not know which View needs to be made use of since it does not have any direct access to the View. The two are connected using data binding, and the ViewModel records all the changes that are made to the View and makes the necessary changes to the Model.

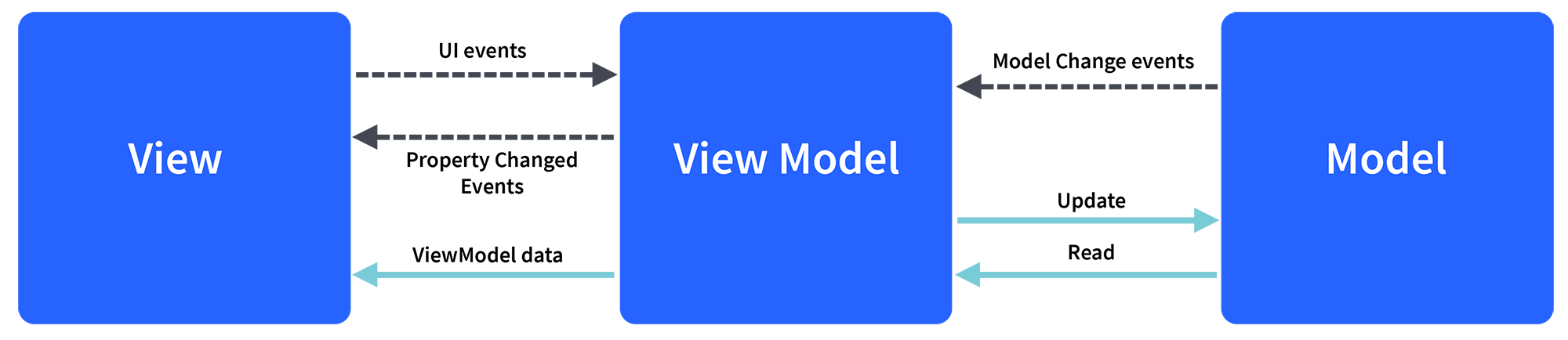
### ****9. What are the differences between Angular decorator and annotation?****

In Angular, decorators are design patterns that help in the modification or decoration of the respective classes without making changes in the actual source code.

Annotations, on the other hand, are used in Angular to build an annotation array. They use the Reflective Metadata library and are a metadata set of the given class

### 15. Explain MVVM architecture

MVVM architecture consists of three parts:  
  
1.Model  
2.View  
3.ViewModel



Model contains the structure of an entity. In simple terms it contains data of an object.  
View is the visual layer of the application. It displays the data contained inside the Model. In angular terms, this will be the HTML template of a component.  
  
ViewModel is an abstract layer of the application. A viewmodel handles the logic of the application. It manages the data of a model and displays it in the view.  
View and ViewModel are connected with data-binding (two-way data-binding in this case). Any change in the view, the viewmodel takes a note and changes the appropriate data inside the model.

**Question 24: How many Change Detectors can there be in the whole application?**  
*Requirement: Expert Angular knowledge*

**Answer:** Each component has its own ChangeDetector. All Change Detectors are inherited from AbstractChangeDetector.

**Question 25: What change detection strategies do you know?**  
*Requirement: Expert Angular knowledge*

**Answer:** There are two strategies – Default and OnPush. If all components use the default strategy, Zone checks the entire tree regardless of where the change occurred. To inform Angular that we will comply with the performance improvement conditions, we need to use the onpush change detection strategy. This will tell Angular that our component depends only on the input and any object that is passed to it should be considered immutable. This is all built on the Principle of the mile automaton, where the current state depends only on the input values.

**Question 26: What is Change Detection, how does Change Detection Mechanism work?**  
*Requirement: Expert Angular knowledge*

**Answer:**Change Detection is the process of synchronizing a model with a view. In Angular, the flow of information is unidirectional, even when using the ng Model to implement two-way binding, which is syntactic sugar on top of a unidirectional flow.

Change Detection Mechanism-moves only forward and never looks back, starting from the root (root) component to the last. This is the meaning of one-way data flow. The architecture of an Angular application is very simple — the tree of components. Each component points to a child, but the child does not point to a parent. One-way flow eliminates the need for a $digest loop.

**Note:**Expect in-depth knowledge of the best practices involved in implementing a design pattern with Angular.

**Question 27: How do you update the view if your data model is updated outside the ‘Zone’?**  
*Requirement: Expert Angular knowledge*

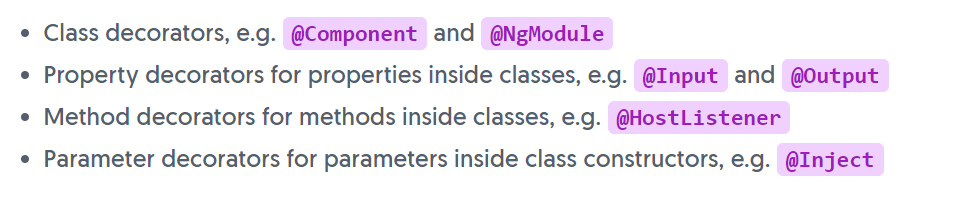
**Answer:**

1. Using the ApplicationRef.prototype.tick method, which will run change detection on the entire component tree.
2. Using NgZone.prototype.run method, which will also run change detection on the entire tree. The run method under the hood itself calls tick, and the parameter takes the function you want to perform before tick.
3. Using the ChangeDetectorRef.prototype.detectChanges method, which will launch change detection on the current component and its children.

**Note:**Expect an expert Angular developer to know about other popular methods like ngOnChanges, ngAfterViewChecked, ngDoCheck, and ngOnDestroy since they are important.

**1.Decorator?**

**Decorators** are a design pattern that is used to separate modification or decoration of a class without modifying the original source code



**2.Types of Dirctives in Angular?**

Directives: A directive is a class in Angular that is declared with a **@Directive** decorator.  
Every directive has its own behaviour and can be imported into various components of an application

Directives can change the appearance or behavior of DOM elements and Angular components.

**There are four types of directives in Angular,**

* Components **directives**.
* Structural **directives**.
* Attribute **directives**.
* Custom **Directive**.

**3.Data Binding?**

It allows us to define communication between the component and view.

This post is primarily focused on what data binding is and the types of data binding available.

* String Interpolation.
* Property Binding.
* Event Binding.
* Two-Way Data Binding.

**4.Filters in Angular?**

Filters are used in Angular JS for data Transformation

Pipes are ysed in Angular for Data transformation

Filters in Angular are used for formatting the value of an expression in order to display it to the user.

These filters can be added to the templates, directives, controllers or services.

Not just this, you can create your own custom filters.Using them, you can easily organize data in such a way

that the data is displayed only if it fulfills certain criteria. Filters are added to the expressions by using the pipe

character |, followed by a filter.

**5. What is a provider in Angular?**

A provider is a configurable service in Angular. It is an instruction to the Dependency Injection system that provides

information about the way to obtain a value for a dependency. It is an object that has a $get() method which is

called to create a new instance of a service. A Provider can also contain additional methods and uses $provide in o

rder to register new providers.

6. **What is Dependency Injection in Angular?**

Dependency Injection (DI) is a software design pattern where the objects are passed as dependencies

rather than hard-coding them within the component. The concept of Dependency Injection comes in handy

when you are trying to separate the logic of object creation to that of its consumption.

The ‘config’ operation makes use of DI that must be configured beforehand while the module gets loaded to retrieve the elements of the application.

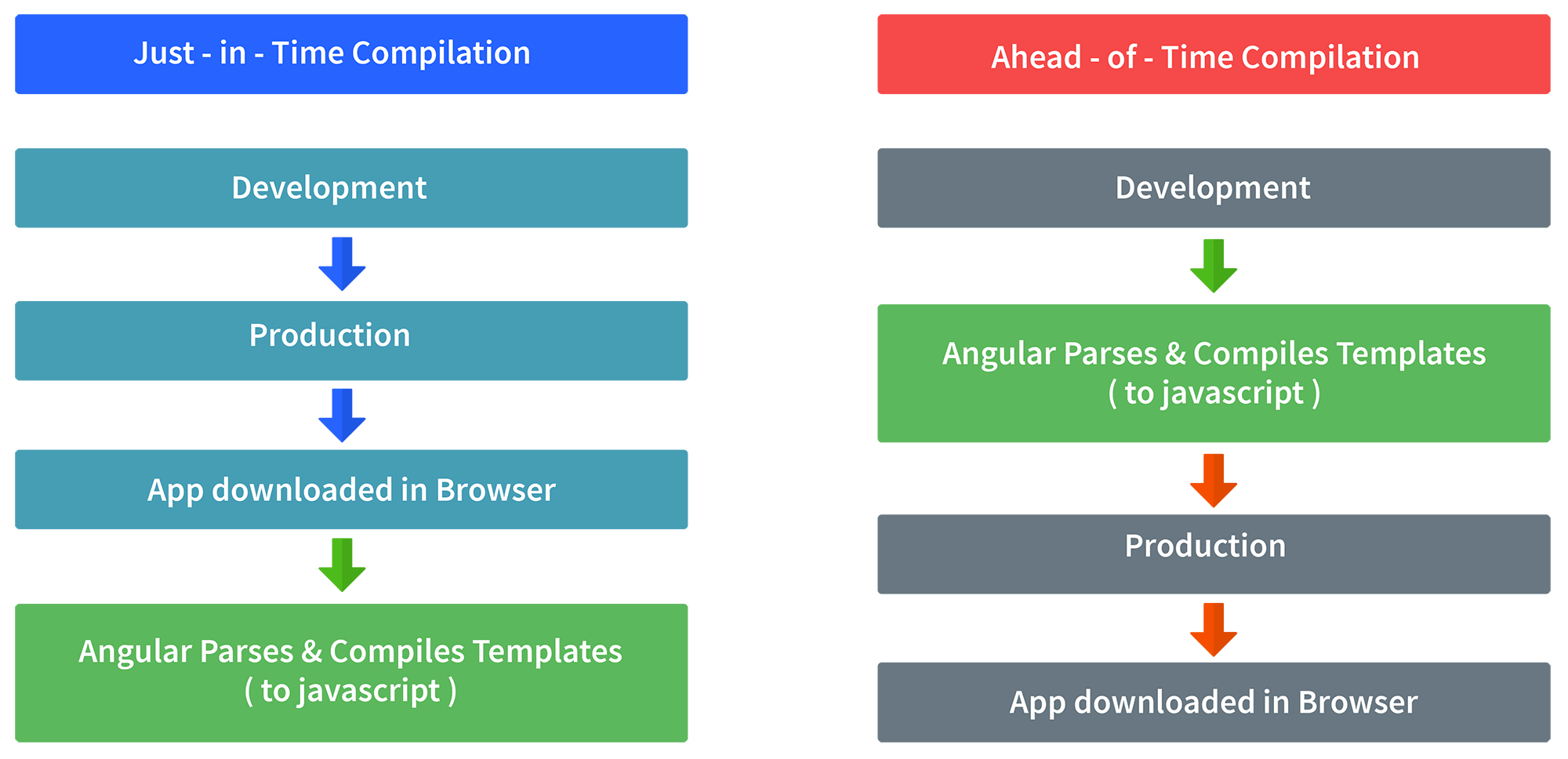
With this feature, a user can change dependencies as per his requirements.

### 7. What is AOT compilation? What are the advantages of AOT?

Every Angular application consists of components and templates which the browser cannot understand. Therefore, all the Angular applications need to be compiled first before running inside the browser.  
  
Angular provides two types of compilation:

 JIT(Just-in-Time) compilation

 AOT(Ahead-of-Time) compilation



In JIT compilation, the application compiles inside the browser during runtime.  
Whereas in the AOT compilation, the application compiles during the build time.  
  
The advantages of using AOT compilation are:

 Since the application compiles before running inside the browser, the browser loads the executable code and renders the application immediately, which leads to **faster rendering**.

 In AOT compilation, the compiler sends the external HTML and CSS files along with the application, eliminating separate AJAX requests for those source files, which leads to **fewer ajax requests**.

 Developers can detect and handle errors during the building phase, which helps in **minimizing errors**.

 The AOT compiler adds HTML and templates into the JS files before they run inside the browser. Due to this, there are no extra HTML files to be read, which provide **better security** to the application.

By default, angular builds and serves the application using JIT compiler:

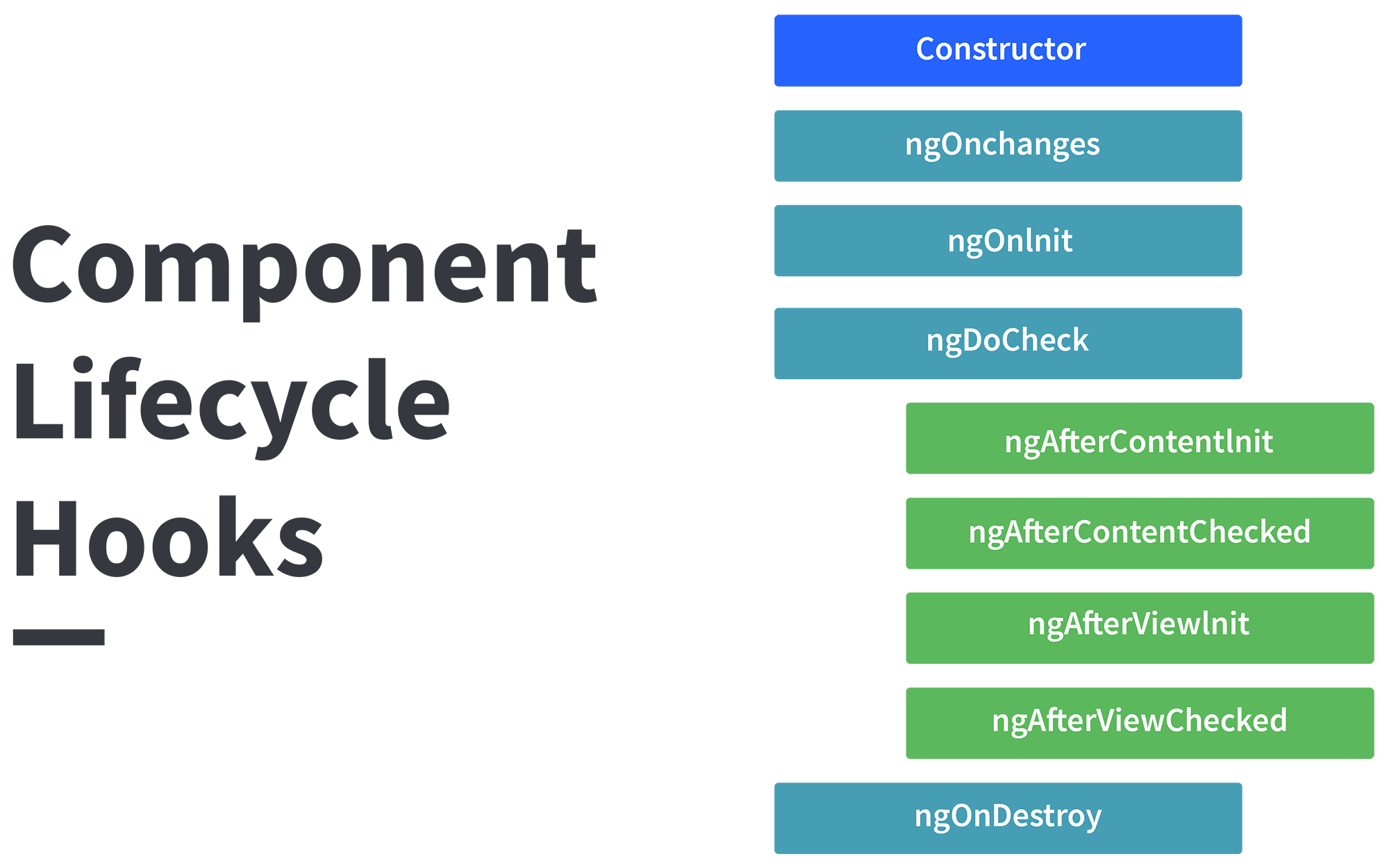
ng build  
ng serve

For using AOT compiler following changes should be made:

ng build --aot  
ng serve --aot

**7. What are lifecycle hooks in Angular? Explain a few lifecycle hooks.**

Every component in Angular has a lifecycle, different phases it goes through from the time of creation to the time it's destroyed. Angular provides **hooks** to tap into these phases and trigger changes at specific phases in a lifecycle.



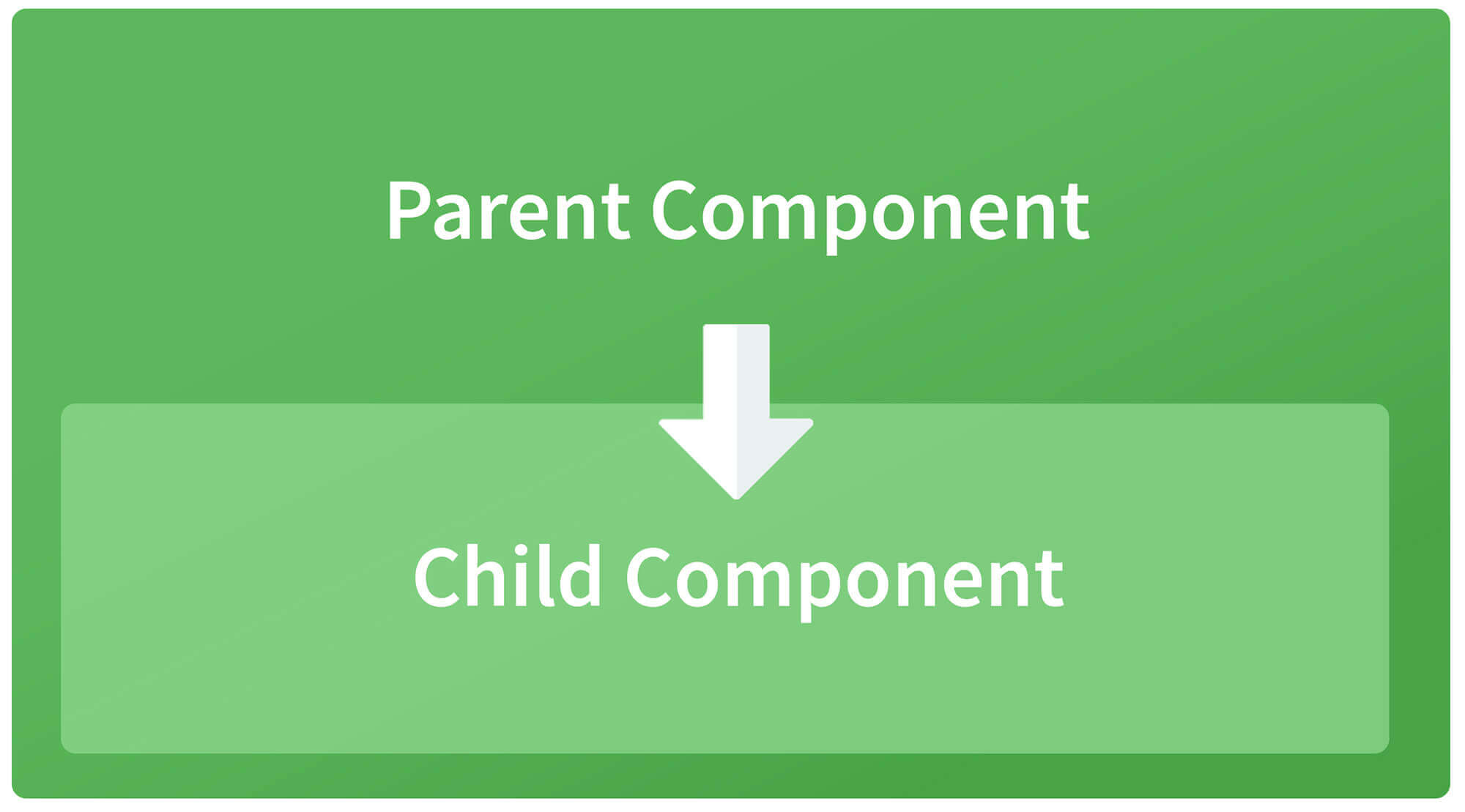
**10. How are observables different from promises?**

The first difference is that an Observable is **lazy** whereas a Promise is **eager**.

|  |  |
| --- | --- |
| Promise | Observable |
| Emits a single value | Emits multiple values over a period of time |
| Not Lazy | Lazy. An observable is not called until we subscribe to the observable |
| Cannot be cancelled | Can be cancelled by using the unsubscribe() method |
|  | Observable provides operators like map, forEach, filter, reduce, retry, retryWhen etc. |

### 13. How does one share data between components in Angular?

Following are the commonly used methods by which one can pass data between components in angular:



**Parent to child using @Input decorator**  
  
Consider the following parent component:

@Component({

selector: 'app-parent',

template: `

<app-child [data]=data></app-child>

` ,

styleUrls: ['./parent.component.css']

})

export class ParentComponent{

data:string = "Message from parent";

constructor() { }

}

In the above parent component, we are passing “data” property to the following child component:

import { Component, Input} from '@angular/core';

@Component({

selector: 'app-child',

template:`

<p>{{data}}</p>

`,

styleUrls: ['./child.component.css']

})

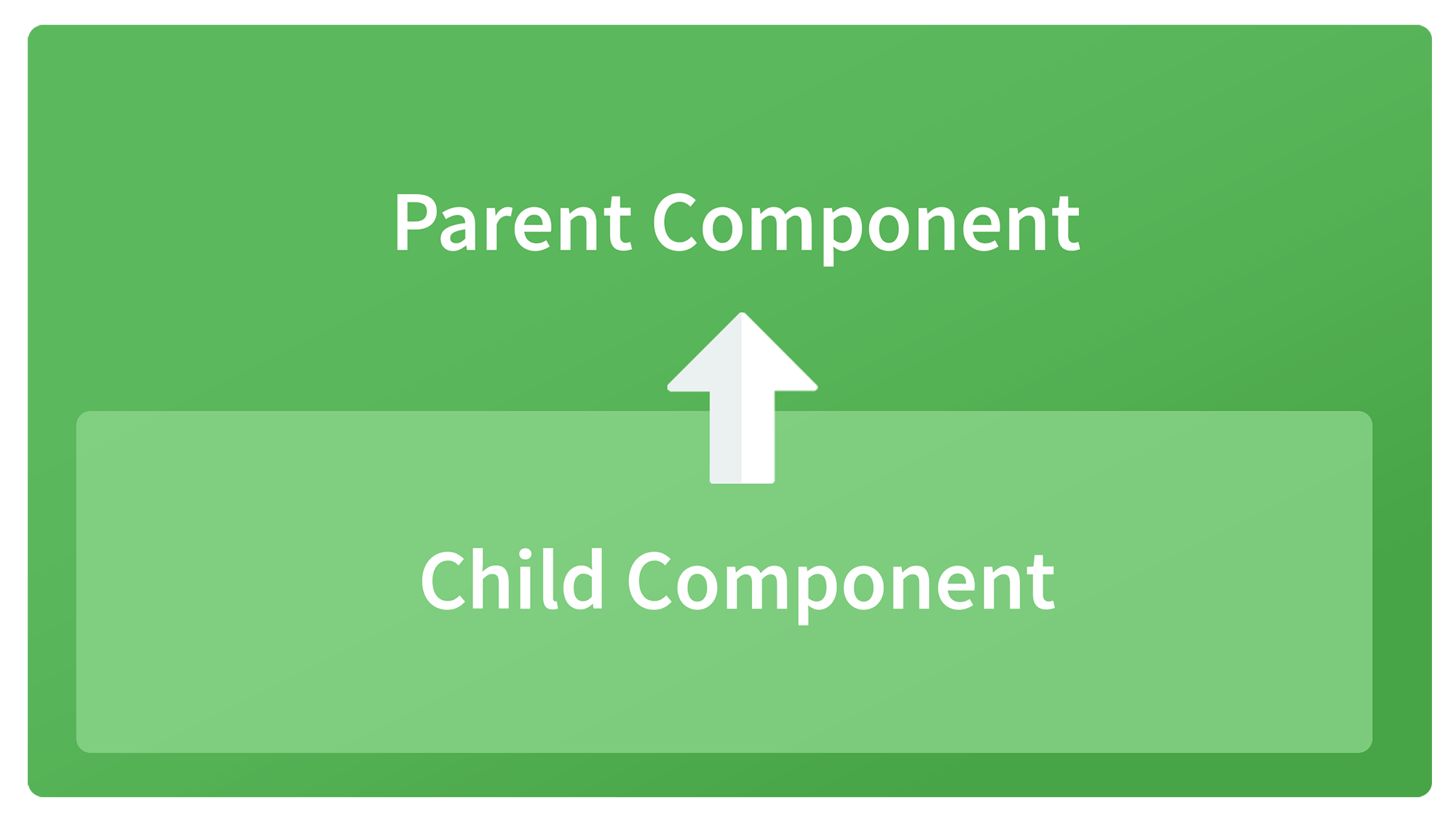
export class ChildComponent {

@Input() data:string

constructor() { }

}

In the child component, we are using @Input decorator to capture data coming from a parent component and using it inside the child component’s template.



**Child to parent using @ViewChild decorator**  
  
Child component:

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

@Component({

selector: 'app-child',

template:`

<p>{{data}}</p>

`,

styleUrls: ['./child.component.css']

})

export class ChildComponent {

data:string = "Message from child to parent";

constructor() { }

}

Parent Component

import { Component,ViewChild, AfterViewInit} from '@angular/core';

import { ChildComponent } from './../child/child.component';

@Component({

selector: 'app-parent',

template: `

<p>{{dataFromChild}}</p>

` ,

styleUrls: ['./parent.component.css']

})

export class ParentComponent implements AfterViewInit {

dataFromChild: string;

@ViewChild(ChildComponent,{static:false}) child;

ngAfterViewInit(){

this.dataFromChild = this.child.data;

}

constructor() { }

}

In the above example, a property named “data” is passed from the child component to the parent component.  
**@ViewChild** decorator is used to reference the child component as “child” property.  
Using the **ngAfterViewInit** hook, we assign the child’s data property to the messageFromChild property and use it in the parent component’s template.  
  
**Child to parent using @Output and EventEmitter**  
  
In this method, we bind a DOM element inside the child component, to an event ( **click** event for example ) and using this event we emit data that will captured by the parent component:  
  
Child Component:

import {Component, Output, EventEmitter} from '@angular/core';

@Component({

selector: 'app-child',

template:`

<button (click)="emitData()">Click to emit data</button>

`,

styleUrls: ['./child.component.css']

})

export class ChildComponent {

data:string = "Message from child to parent";

@Output() dataEvent = new EventEmitter<string>();

constructor() { }

emitData(){

this.dataEvent.emit(this.data);

}

}

As you can see in the child component, we have used **@Output** property to bind an **EventEmitter**. This event emitter emits data when the button in the template is clicked.  
  
In the parent component’s template we can capture the emitted data like this:

<app-child (dataEvent)="receiveData($event)"></app-child>

Then inside the receiveData function we can handle the emitted data:

receiveData($event){

this.dataFromChild = $event;

}

### 14. Explain the concept of Dependency Injection?

Dependency injection is an application design pattern which is implemented by Angular.  
It also forms one of the core concepts of Angular.  
  
**So what is dependency injection in simple terms?**  
Let’s break it down, dependencies in angular are nothing but services which have a functionality. Functionality of a service, can be needed by various components and directives in an application. Angular provides a smooth mechanism by which we can inject these dependencies in our components and directives.  
So basically, we are just making dependencies which are injectable across all components of an application.

#### 15.**Question: Define the ng-content Directive?**

**Answer**: Conventional HTML elements have some content between the tags. For instance:

<**p**>Put your paragraph here</**p**>

Now consider the following example of having custom text between angular tags:

<**app-work**>This won’t work like HTML until you use ng-content Directive</**app-work**>

However, doing so won’t work the way it worked for HTML elements. In order to make it work just like the HTML example mentioned above, we need to use the ng-content Directive. Moreover, it is helpful in building reusable components.

**16.Question: Explain Dependency Injection?**

**Answer:**Dependency injection is an application design pattern that is implemented by Angular and forms the core concepts of Angular.

Let us understand in a detailed manner. Dependencies in Angular are services which have a functionality. Various components and directives in an application can need these functionalities of the service. Angular provides a smooth mechanism by which these dependencies are injected into components and directives.

#### **17.Question: Demonstrate navigating between different routes in an Angular application.**

**Answer**: Following code demonstrates how to navigate between different routes in an Angular app dubbed “Some Search App”:

import **from** "@angular/router";  
.  
.  
.  
@Component({  
  selector: 'app-header',  
  template: `  
<nav **class**="navbar navbar-light bg-faded">  
  <a **class**="navbar-brand" (click)="goHome()">Some Search App</a>   
  <ul **class**="nav navbar-nav">  
    <li **class**="nav-item">  
      <a **class**="nav-link" (click)="goHome()">Home</a>   
    </li>  
    <li **class**="nav-item">  
      <a **class**="nav-link" (click)="goSearch()">Search</a>   
    </li>  
  </ul>  
</nav>  
 `  
})  
**class** **HeaderComponent** {  
  constructor(**private** router: Router) {}   
  goHome() {  
    **this**.router.navigate(['']);   
  }  
  goSearch() {  
    **this**.router.navigate(['search']);   
  }  
}

#### **17.Question: Could you explain services in Angular?**

**Answer:** Singleton objects in Angular that get instantiated only once during the lifetime of an application are called services. An Angular service contains methods that maintain the data throughout the life of an application.

The primary intent of an Angular service is to organize as well as share business logic, models, or data and functions with various components of an Angular application.

The functions offered by an Angular service can be invoked from any Angular component, such as a controller or directive.

#### **18.Question: Explain Angular Authentication and Authorization.**

**Answer**: The user login credentials are passed to an authenticate API, which is present on the server. Post server-side validation of the credentials, a JWT (JSON Web Token) is returned. The JWT has information or attributes regarding the current user. The user is then identified with the given JWT. This is called authentication.

Post logging-in successfully, different users have a different level of access. While some may access everything, access for others might be restricted to only some resources. The level of access is authorization.

#### 19. **Question: Explain the difference between an Annotation and a Decorator in Angular?**

**Answer:** In Angular, annotations are used for creating an annotation array. They are only metadata set of the class using the Reflect Metadata library.

Decorators in Angular are design patterns used for separating decoration or modification of some class without changing the original source code.

#### **20.Question: What is Angular Material?**

**Answer**: It is a UI component library. [Angular Material](https://material.angular.io/) helps in creating attractive, consistent, and fully functional web pages as well as web applications. It does so while following modern web design principles, including browser portability and graceful degradation.

#### **21.Question: What is AOT (Ahead-Of-Time) Compilation?**

**Answer**: Each Angular app gets compiled internally. The Angular compiler takes in the JS code, compiles it and then produces some JS code. This happens only once per occasion per user. It is known as AOT (Ahead-Of-Time) compilation