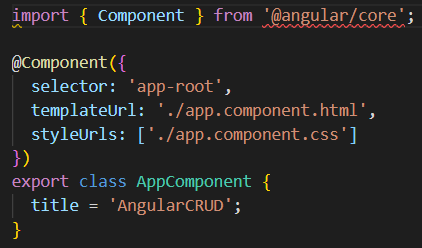
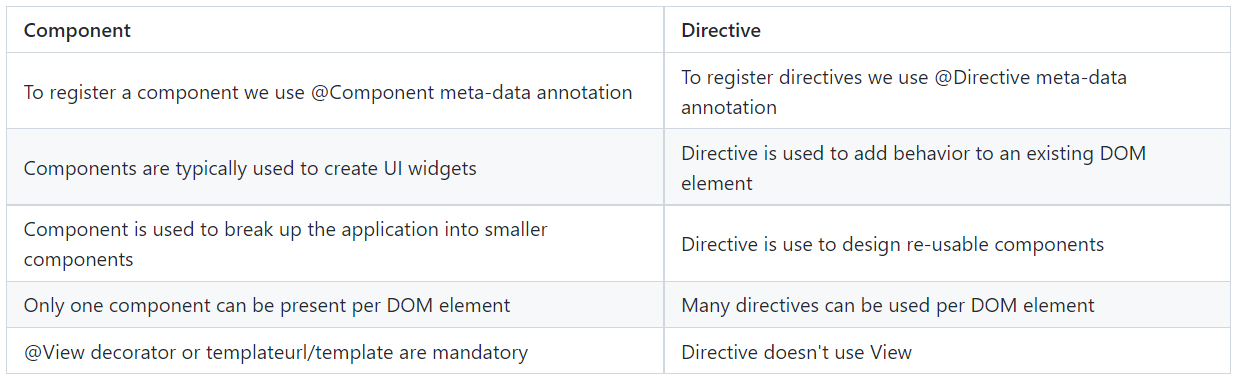
# Basic

## What is Angular

## What is component

* components are the basic building blocks, which control a part of the UI for any application
* ng g c test
* @Component decorator
* component consists of three parts

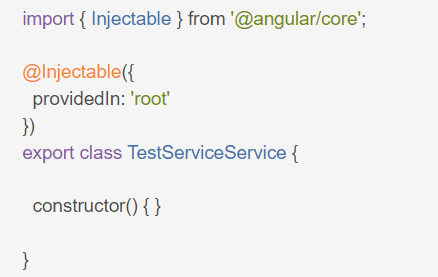


* An Angular application is a tree of Angular components
* Angular components are a subset of directives
  + In a short note, A component(@component) is a directive-with-a-template.
* A component must belong to an NgModule in order for it to be usable by another component or application.
* 

## What are directives?

* Directives add behaviour to an existing DOM element or an existing component instance.
  + 

## What is Service

* Services are objects which get instantiated only once during the lifetime of an application
* The main objective of a service is to share data, functions with different components of an Angular application
* A service is defined using a @Injectable decorator
* A function defined inside a service can be invoked from any component or directive.
* 

## What is Routing

## What is module

* Modules are **logical boundaries** in your application and the application is divided into separate modules to separate the functionality of your application
* A module helps to group components, directives, services, and pipes
* Module decides whether the components, directives, etc can be used by other modules, by exporting or hiding these elements
* Every module is defined with a **@NgModule** decorator
* By default, modules are of **two types**
  + Root Module
    - Every application can have **only one root module**
    - root module is defined inside **app.module.ts**
    - A root module imports BrowserModule
    - 
  + Feature Module
    - To create a feature module- **ng g m test-module**
    - Ng app can have **one or more feature modules**
    - feature module **imports CommonModule**
    - 

## What is App Module file?

* App Module is where root module is defined

## What is Main.ts file

## What is Bootstrapping

## What are Http Interceptors?

## Project Files

### **angular.json**

* + Configuration file for angular
  + Allow you to configure behavior of Angular CLI

### **package.json**

* + Dependency needed for application

### **package.locked.json**

### **tsconfig.json**

* + contain Compiler Option for typescript

### **tslint.json**

## Explain string interpolation and property binding in Angular.

* String interpolation and property binding are parts of data-binding in Angular.
* Data-binding help to communicate between the component (Model) and its view (HTML template).
* Data-binding can be done in 2 ways
  + **one-way binding** 
    - In one-way binding, any changes in the component will directly reflect inside the HTML template but, vice-versa is not possible. Whereas, it is possible in two-way binding
    - **String interpolation and property binding** allow only one-way data binding
    - String interpolation uses the double curly braces {{ }} to display data from the component
    - Using property binding, we can bind the DOM properties of an HTML element to a component's property
    - Property binding uses the square brackets [ ] syntax
  + **two-way binding.**

## How you connect angular to API service

import { HttpClient } from '@angular/common/http';

import { HttpHeaders } from '@angular/common/http';

import { Observable } from 'rxjs';

## What are directives in Angular?

* **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.

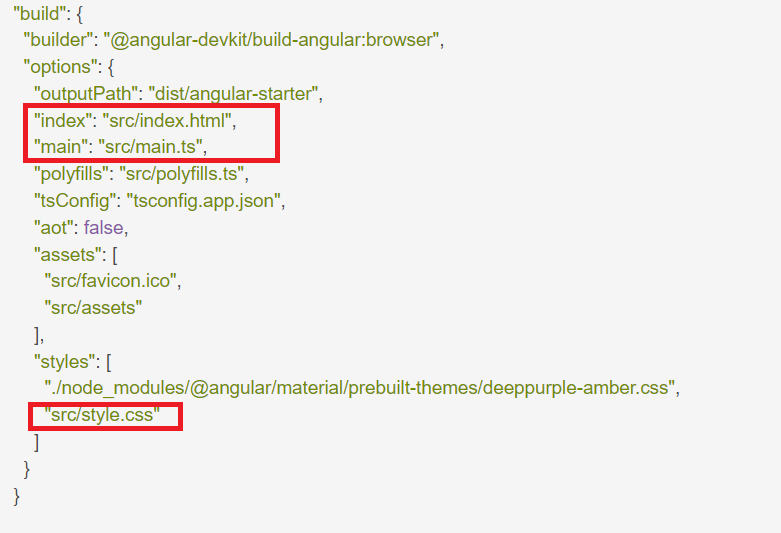
### Types of directives

* **Component directives**
  + These form the main class in directives.
  + we use @Component decorator to declare these directives.
  + directives have a view, a stylesheet and a selector property.
* **Structural directives**
  + used to manipulate DOM elements
  + Every structural directive has a ‘ \* ’ sign before them
  + We can apply these directives to any DOM element
  + Example - \*ngIf and \*ngFor directives
* **Attribute Directives**
  + used to change the **look and behaviour** of a DOM element**.**
  + **ng g directive blueBackground**

## How you connect angular to API service

## Angular structure and files details - **angular.json**

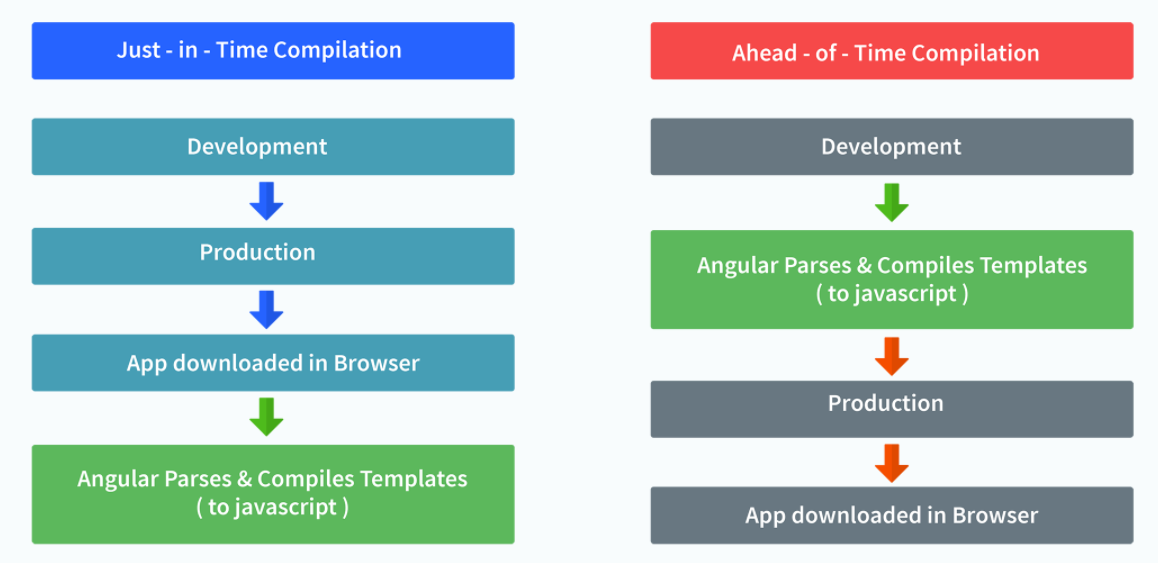
* angular. json will contain all the configurations of the app
* While building the app, the **builder looks at this file** to find the **entry point** of the application
* entry point of the application is main.ts.



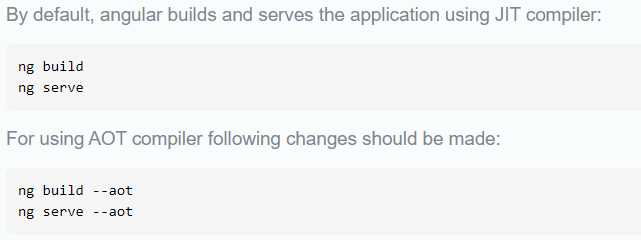
# Intermediate

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

* Angular application consists of components and templates that browser cannot understand directly so angular application must be compiled before rendering into browser
* **Angular support two types of compilation**
  + JIT(Just-in-Time) compilation (**Run time compilation**)
    - In JIT, the app compiles inside the browser **during runtime**
  + AOT(Ahead-of-Time) compilation -
    - AOT compilation, the application compiles during the build time



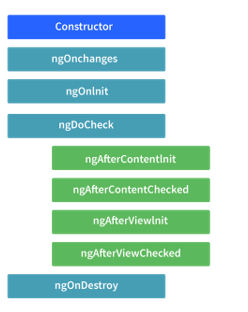
* **The advantages of using AOT compilation are**
  + Since the app compiles before running inside the browser, the browser loads the executable code and renders the application immediately, which leads to **faster rendering**.
  + In AOT compilation, the compiler sends the external HTML and CSS files 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.



## providedIn: 'root' in Angular Service

## Angular Life Cycle Hook

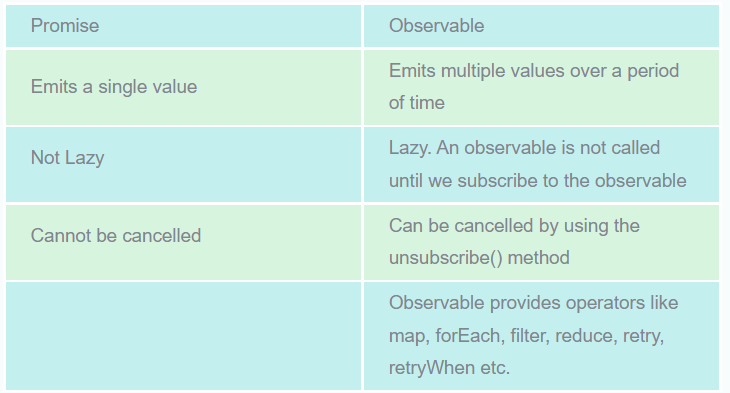
* Every component in Angular has a lifecycle, **different phases it goes through** from the time of creation to the time it's destroyed.
* Angular provides hooks to **tap into these phases and trigger changes at specific phases** in a lifecycle



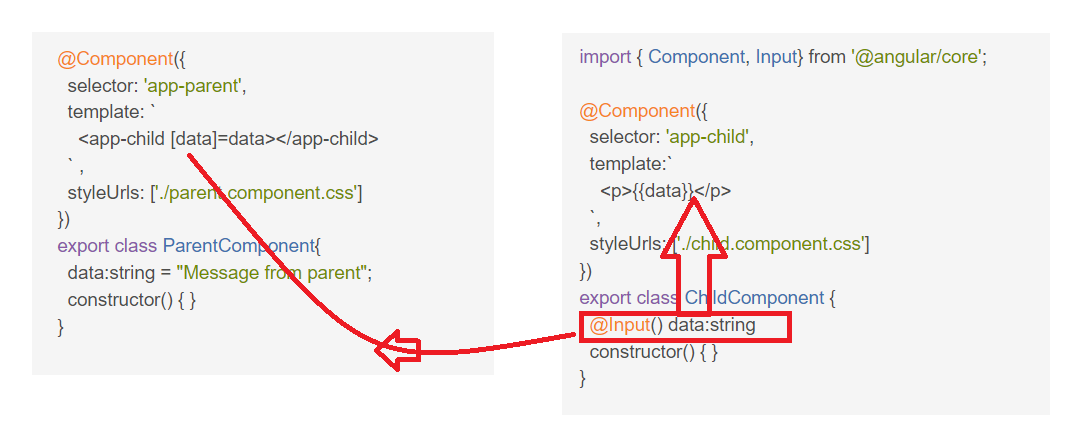
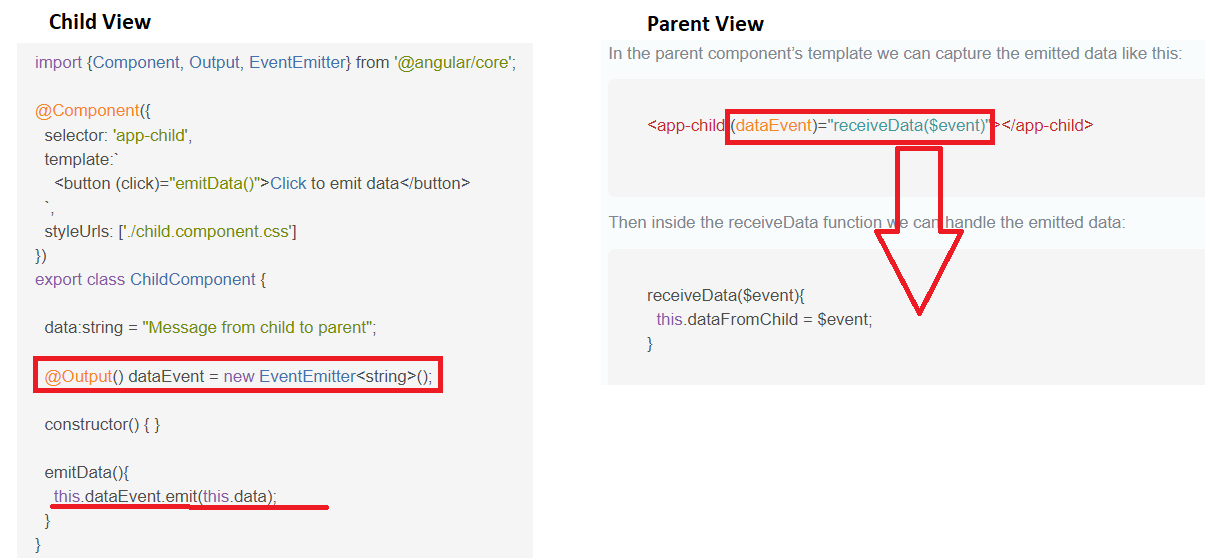
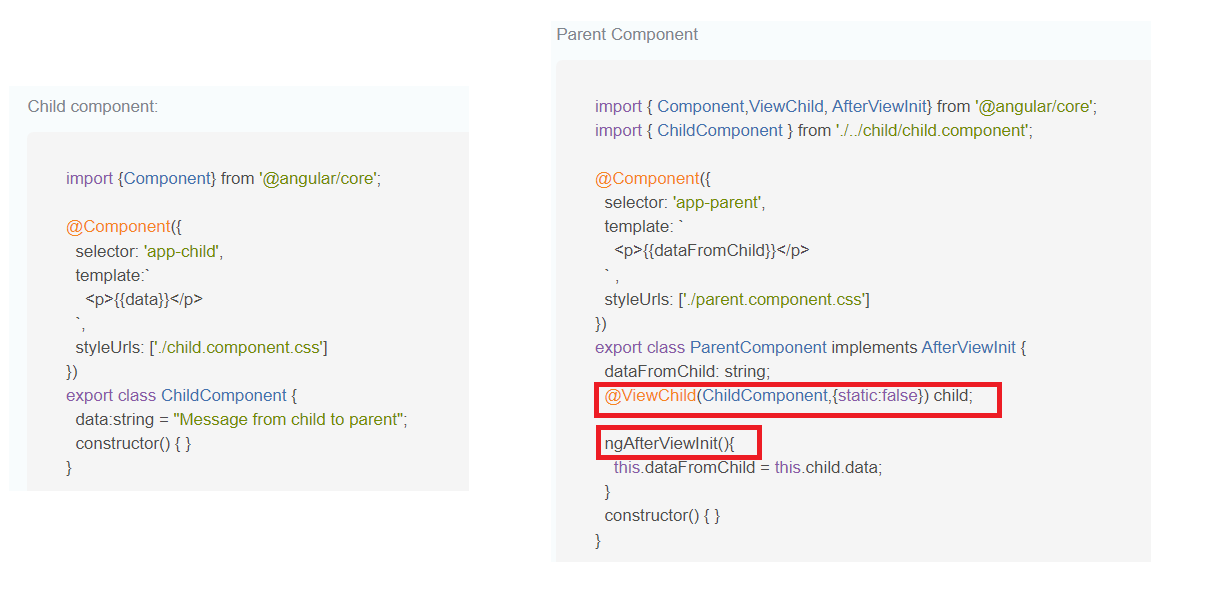
* ngOnChanges( )-
  + This method/hook receives a **SimpleChanges** object which contains the **previous and current values** of the property.
  + Responds when Angular sets/resets data-bound input properties.
* ngOnInit( )
  + This hook gets **called once**, after the ngOnChanges hook.
  + It **initializes the component and sets the input properties** of the component.
* ngDoCheck( )
  + It gets called after ngOnChanges and ngOnInit
  + is used to detect and act on changes that cannot be detected by Angular.
  + We can implement our change detection algorithm in this hook
  + Detect and act upon changes that Angular can't or won't detect on its own.
* ngOnDestroy( )
  + It gets called just before Angular destroys the component.
  + This hook can be used to clean up the code and detach event handlers.

## How are observables different from promises?

* The first difference is that an **Observable is lazy**
* observables are lazy. Observable runs only when someone subscribes to them
* whereas a **Promise is eager**.
* <https://www.interviewbit.com/angular-interview-questions/>



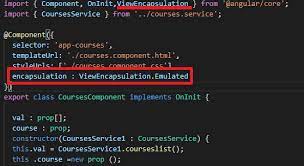
## **How does one share data between components in Angular?**

* **Parent to child**
  + Parent to child using **@Input decorator**
  + In the child component, we are using **@Input decorator** to capture data coming from a parent component
  + 
* **Child to parent – way1** 
  + Child to parent using **@Output** and **EventEmitter**
  + 
* **Child to parent – way2**
  + Child to parent using **@ViewChild decorator**
  + 

## Explain the concept of Dependency Injection?

## What is view encapsulation in Angular?

* View encapsulation defines whether the template and styles defined within the component can affect the whole application or vice versa
* three encapsulation strategies
  + Emulated - styles from the main HTML propagate to the component.
  + Native - styles from the main HTML do not propagate to the component
  + None - styles from the component propagate back to the main HTML and therefore are visible to all components on the page



## What are Template and Reactive forms?

## What is observables

# Advanced

## What are Observables?

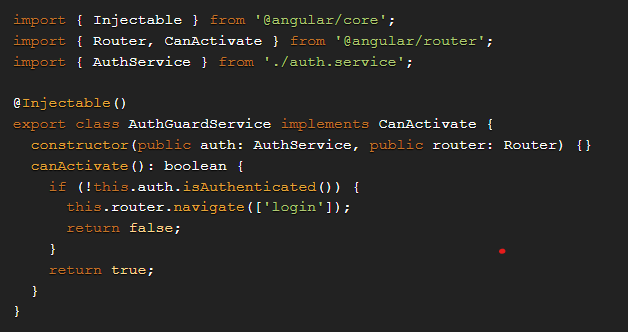
* **Observables** are declarative which provide support **for passing messages between publishers and subscribers** in your application
* used for event handling, asynchronous programming, and handling multiple values
* you define a function for publishing values, but it is not executed until a consumer subscribes to it .The subscribed consumer then receives notifications until the function completes, or until they unsubscribe.

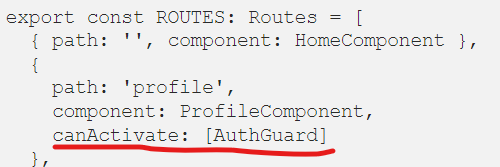
## What is Eager and Lazy loading?

## What is Angular Router?

## What is Routing Guard in Angular?

* Angular’s **route guards** are interfaces which can tell the router **whether or not it should allow navigation**to a requested route.
* They make this decision by looking for a true or false return value **from a class which implements the given guard interface**.
* **five** different types of guards and each of them is called in a particular **sequence**
  + CanActivate
  + CanActivateChild
  + CanDeactivate
  + CanLoad
  + Resolve





* The JwtHelperService class from angular2-jwt can be used for this.
* npm install --save @auth0/angular-jwt
* import { JwtHelperService } from '@auth0/angular-jwt';
* **Note**: The canActivate guard still allows the component for a given route to be activated (but not navigated to). If we wanted to prevent activation altogether, we could use the canLoad guard.

## What is multicasting in Angular?

# Practical

## Implement module?