CRACK YOUR ANGULAR INTERVIEW

Bhairab Dutt

Mahesh Verma

ABOUT THE BOOK

This book is based on Angular Interview Questions which are mainly asked in most of the interviews. We are pleased to say that we got the idea of this book while looking for the perfect solutions for our Angular-related queries. Unfortunately, in most cases, we couldn't find enough information or satisfactory solutions. Often, we had to search through multiple websites, tutorials, and links to collect enough information to resolve our queries and after that we used to compile for further convenience. After facing this type of situation, we planned to write a book which covers almost every Angular-based interview question in addition to covering almost all the topics of Angular.

We think this book will help you to prepare in a better way towards cracking your interviews. This is the reason behind the title of book, i.e., **"Crack Your Angular Interview"**.

We believe that you also believe that reading multiple questions related to a subject is a great way to learn and revise the concept. It helps to enhance the knowledge in a great helpful way. In this book, we tried to cover most of the important concepts of each version of Angular in the form of interview questions and answers with examples.

Although we have tried to make this book as accurate as possible but if there is something that is not required or you find an error in the book, please let us know.

ABOUT THE AUTHORS

Bhairab Dutt

Bhairab Dutt has an experience of 8+ years in developing enterprise-level applications using: C#, ASP.NET4.0 with MVC, Angular 2, Angular 4+, LINQ, SQL Server, AJAX, XML, JavaScript, and jQuery; and analyzing the business requirements and translating it to technical specifications.

He holds a vast experience in all the phases of software development life cycle (SDLC) including requirements gathering, analysis, design, reviews, coding, testing, debugging, documenting, unit and integration testing.

Bhairab is a good communicator with a bunch of interpersonal skills that help him in leveraging technical, business acumen to communicate effectively with client executives and their respective teams and recognizing project business needs and presenting solutions.

Mahesh Verma

I have been working for 5 years in software developing field.
I have designed and developed applications using C#, ASP.NET4.0 with MVC, LINQ, SQL Server, AngularJS, Angular (version 2+) etc.
Experience in all the phases of software development life cycle (SDLC): including requirements gathering, analysis, design, reviews, coding, testing, debugging, documenting, unit and integration testing.
I love to do work with client side framework like Angular, Angular (ver. 2+), React, Vue, etc. and learn all the new technologies also with the expertise to grasp new concepts quickly and utilize the same in a productive manner.

Index

Folder Structure

- 31 What is the purpose of maints file?
- 32 What is the role of packagejson file?
- 33 How packagejson file different from package-lockjson file?
- 34 What is the role of tsconfigjson file?
- 35 What is the role of Angular-clijson file?
- 36 What is the use of “assets” folder in Angular?

Data Binding

- 37 What is data binding in Angular?
- 38 How many types of binding supported by Angular?
- 39 What is Interpolation? Give an example
- 40 What is Property binding? Where you can implement property binding?
- 41 How many ways you can achieve property binding? What is the benefit of using property binding?
- 42 When we used the Property binding and when we use Interpolation?
- 43 What is Class binding? Give an example
- 44 What is Style binding? Give an example
- 45 What is Attribute binding? What is the benefit of using it
- 46 What is Event binding? Give an example
- 47 Where you can implement event binding?
- 48 Name some common events that you can use for event binding?
- 49 What is Two way binding? Give an example
- 50 What is template reference variable?
- 51 How many ways we can create template reference variable?
- 52 Can we use template reference variable using select (combo box) If yes, then how?
- 53 How Angular ensure about content security at the time of binding?
- 54 What is ngModel? or What is the role of ngModel?
- 55 Which binding we will use for dynamic css?

Components

- 56 What is component in Angular ?
- 57 How can we create a component?
- 58 Component is directive or not?
- 59 What is component decorator?
- 60 What are the different properties of a component object?
- 61 What is the mandatory property of @Component() decorator function?
- 62 What is ViewEncapsulation?
- 63 What is changeDetection Property?
- 64 What is the difference between templeteUrl and template?
- 65 What is the difference between styleUrl and style?

- 66 What is the difference between providers present in component and present in appmodulets file?
- 67 What is Dynamic Component?
- 68 What is the use of ComponentFactoryResolver Service?
- 69 Give an example of Dynamic Component?
- 70 What is nested component?
- 71 What is the role of selector?
- 72 What is entry Component?
- 73 What is component hook lifecycle?
- 74 If there is nested component (parent and child) then order of lifecycle hook is same or different?
- 75 What is the difference between @ViewChild and @ViewChildren?
- 76 What is the difference between @ContentChild and @ContentChildren?
- 77 What is the difference between ngDoCheck and ngOnChange?
- 78 How can we pass data from component to component?
- 79 How can we pass data from parent component to child component using @Input
- 80 How can we pass data from child component to parent component using @Output
- 81 What is Event Emitter?

Directives

- 82What is directives?
- 83 How many types of directives supported by Angular?
- 84 What is @Directive decorator?
- 85 Can we create constructor in our directive If yes then how
- 86 What is Component directive Give an example
- 87 What is structural directive
- 88 Name some structural directives provided by Angular
- 89 What is the difference between ngIf and hidden?
- 90 How can we use "then" and "else" keywords with *ngIf directive? Explain with an example
- 91 What are the keywords that we used at the time of ngSwitchCase and what is the role of that keywords?
- 92 What is *ngFor ? and what are the exported values of ngFor directive?
- 93 What is the difference between ngFor and ngForOf?
- 94 Why "*" is prefix with structural directive? Can we use structural directive without using *?
- 95 How many structural directives can we implement on a single element?
- 96 How can list items be tracked by default?
- 97 Can we provide our own mechanism for tracking the elements? If yes, then how?
- 98 What are attribute directives? Give an example
- 99 What is the difference between attribute directive and component directive?
- 100 What is ng-template?
- 101 What is Host Listner

- 102 What is ElementRef?
- 103 What is the purpose of @HostBinding?

Pipes

- 104 What is pipes and how we can use in Angular ?
- 105 Name some built-in pipe provided by the Angular
- 106 Can I create custom pipe in Angular?
- 107 What is Chaining pipe?
- 108 What is @Pipe Decorator?
- 109 How many types of Pipes support by Angular?
- 110 How can you define or set your pipe as pure or impure?
- 111 Can I create any pipe in Angular if yes then How?
- 112 What is an AsyncPipe in Angular?

Services

- 113 What is services?
- 114 HTTP Services?
- 115 What are the different HTTP Verbs supported by Angular
- 116 What are the difference between Patch() and Put()?
- 117 Why we are using services for transfer the data while we have @Input and @Output?
- 118 How we can send the value from one component to another component using service and their has not relation of parent and child?
- 119 What is providers?
- 120 What is Dependency Injection (DI) ?
- 121 Explain about @Injectable?
- 122 How many services we can add into Providers ?
- 123 If we have imported the service in the appmodulets and then using the service then it will give error or run successfully IF error then which type of error?
- 124 How many decorator has in Angular ?
- 125 What is backtick and how we are using in Angular?
- 126 What is DOM Shadowing ?
- 127 If we send any number value from child to parent component then the emitter will number or string?

Dependency injection

- 128 What is Dependency Injection?
- 129 What is RXJS?
- 130 What is the difference between promises and observable ?
- 131 What is subscribers ?
- 132 How we can handle the error in Angular?

Routing

- 133 What is Routing in Angular?
- 134 How can you define routing in Angular?
- 135 What is a RouterOutlet?
- 136 What is Router links?
- 137 What is Router State?
- 138 What are Router events?
- 139 What is Wildcard route?
- 140 What is pathMatch property in routing?
- 141 How can we pass parameter in Routing?
- 142 What are the guard interfaces supported by router?
- 143 Difference between [routerLink] and routerLink?

Form

- 144 What is Form and how many strategies we have for develop the form
- 145 What is Template driven and reactive form and why we are using in Angular?
- 146 Which one the steps which we are using to build a form with template driven ?
- 147 When we will use the form using Template driven, Which module we have to add and where ?
- 148 How many ways by which we can add form validation?
- 149 How we will use Template-Driven Form Validation ?
- 150 Why we are checking dirty and touched?
- 151 What is Reactive Forms?
- 152 When we will use the form using Reactive, which module we have to add and where?
- 153 What is Dynamic Form and how we can implement it in Angular application?
- 154 Which classes we are using for show with form according to status of form:

Testing Based

- 155 What is unit testing?
- 156 What is the need of Karma and jasmine in Angular?
- 157 How we can use Karma and Jasmine in Angular in Angular for testing?
- 158 What is the need of ng test ?
- 159 What is the need of testts in Angular?
- 160 What is test bed?
- 161 We have extension spects in Angular application what is use if it?

Fundamental Questions

1. What is AngularJS?

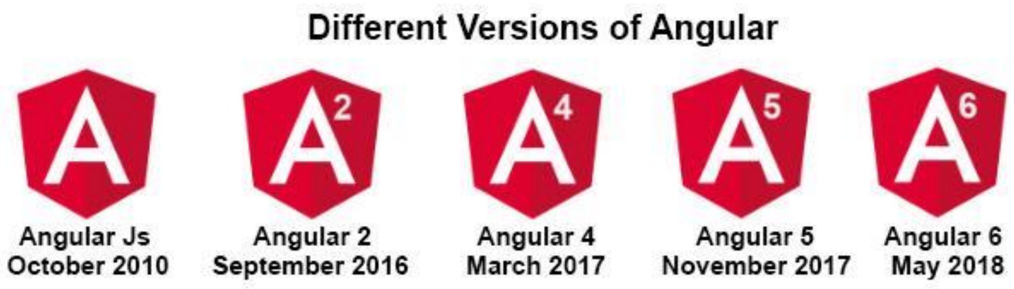
Ans. AngularJS was introduced in 2010, as a JavaScript framework for building client-side applications. It helps you to create single-page applications or one-page web applications that only require HTML, CSS, and JavaScript on the client side.

2. What is Angular and what are the various versions of Angular?

Ans.

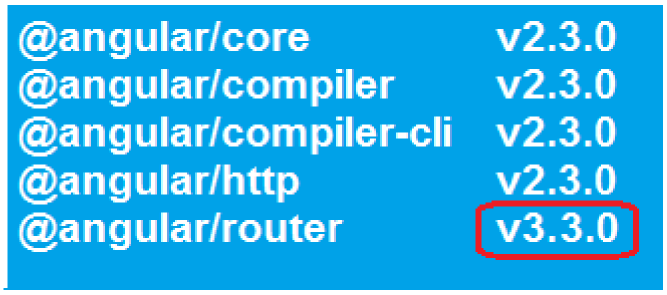
Microsoft's TypeScript team and Google's Angular team together rewrote the AngularJS and announced Angular 2 in 2016. We can say that it is a reborn version of AngularJS which is totally based on TypeScript.

It is also a framework for building client-side application in HTML, CSS and TypeScript.



3. Why was version 3 skipped?

Ans. To avoid the confusion due to the miss alignment of the router's package version which is already distributed as version 3.3.0 with Angular 2, i.e. Angular 2 contains router file version 3.3.0.



@angular/core	v2.3.0
@angular/compiler	v2.3.0
@angular/compiler-cli	v2.3.0
@angular/http	v2.3.0
@angular/router	v3.3.0

4. Why we need Angular?

Ans. As explained above, Angular is completely rewritten, now you need to know why we are using Angular while we have AngularJS. So, the answer is Angular is designed by the team for all the devices. With AngularJS, a developer got problem when using the application on mobile or any other device except web. Thus, the team decided they will develop Angular which can be used for all the devices, as Mobile, iPhones, Tablets etc.
It's the product of Google so every developer wants to learn Angular nowadays. There are a few main things one should know.

- **Easier:** Angular is easier to learn; within a limited time, we can learn it and create the applications with sample CRUD feature.
- **Performance:** Its performance is way better as compared to AngularJS because the execution is very fast when we perform any operation in the application.
- **TypeScript**: Nowadays, most of the developers are using TypeScript because it's easier to learn since it is based on OOPS concepts and is a superset of JavaScript. As you know, most of the developers have knowledge of OOPS, so they can learn it in a minimal time.
- **Angular Testing Easy**: When we work on an Angular project, we can easily test the application by creating the test cases and it's easier than others.
- **Increased Developer Productivity**: Angular increases the developer productivity because a developer can easily learn and complete any task or feature quickly.
- **Work in Coding pattern**: With the help of Angular, you can work on a good consistent coding pattern because it provides codelyzer by which you can write consistent code and discover potential errors.
- **Angular uses full featured Routing**: Routing is yet another great feature of Angular. Angular performs navigation from one view to another and works very fast. It supports lazy loading that allows you to load the pieces of code on demand means entire code will not run one time.
- **Easily build and use only required file**: When we build the Angular v2 - v6 application, the application will build and required file will store in a dist file. In the dist folder, only necessary files will be saved.
- **Change detection:** It's a very important feature of Angular. It refreshes the running application whenever the changes in code are made.

5. What is the problem in Angular JS so we are using Angular?

Ans. The main reason to introduce Angular was that it’s based on OOPs concepts. All the OOPs concepts can be supported by Angular. The second thing is that Angular is designed for web as well as mobile devices.

6. Which language we can use with Angular for Component creation?

Ans. Primarily, we use TypeScript with Angular. However, we can use several other languages too; such as - TypeScript, JavaScript ES5, ES6, and Dart.

7. What is the need of TypeScript in Angular?

Ans. We use TypeScript in Angular for creating the application with components, services etc. and when we create the application, we can use the OOPs concepts.

8. Can I use another language except TypeScript for Angular?

Ans. Yes, JavaScript ES5,ES6 and DART.

9. What is Angular CLI?

Ans. Angular CLI is a command line interface which we use for building the Angular applications using node.js. It provides us the features by which we can easily create all the required files as component, service, pipes etc. with the help of only command line in a good structure. It's based on webpack so can easily bundle the application automatically when that application is built.

10. Why do we use Angular CLI while we have another methods for Angular application creation?

Ans. Yes, we can create the application with another methods also, such as using Visual Studio (any version - 2015, 2017) but with Angular CLI, it's way easier to build a robust application.

11. What is the difference between various versions of Angular?

Ans. **Difference b/w AngularJS and Angular 2**

- **<u>Version</u>** AngularJS is the first version of Angular released in 2010. It is also known as Angular 1. Angular 2 is the second version of Angular released in 2016.
- **<u>JavaScript/TypeScript</u>** JavaScript is used in AngularJS whereas in Angular 2, TypeScript version 1.8 is used.
- **<u>Approach</u>** AngularJS is based on controller and view communication using $scope whereas Angular 2 is a component based approach.
- **<u>Reusability of code</u>** We cannot reuse the code in AngularJS but Angular 2 provides us with the facility to reuse the code.

- ❖ **Communication among components** As compared to AngularJS, it is an easy task to make communication among components in Angular 2.
- ❖ **Unit Testing** The testing of the application requires less efforts in Angular 2 as compared to AngularJS.
- ❖ **Performance** Angular 2 is faster than AngularJS.
- ❖ **Mobile Device support** AngularJS doesn't support mobile devices whereas Angular 2 supports Mobile devices.
- ❖ **Language used** AngularJS uses only JavaScript whereas Angular 2 has more choices in language, like TypeScript, JavaScript, Dart, PureScript etc.

Difference b/w Angular 2 and Angular 4

- Angular 4 is an advanced version of Angular 2 with some modifications in core libraries.
- **Released in** Angular 2 was released in 2016 and Angular 4 was released in 2017.
- **TypeScript** Angular 2 supports TypeScript version 1.8 whereas Angular 4 supports TypeScript version 2.1 & version 2.2.
- **Performance** As compared to Angular 2, Angular 4 is faster. In Angular 4, animation features are separated from the core package, i.e., if don't require animation package, then we don't need to import them. This leads to reduce the bundle size and improve performance.
- In Angular 4, we can use "else" block along with "ngif" which is not possible in Angular 2.

Comparison b/w Angular 4 and Angular 5

- ★ Angular 4 supports TypeScript version 2.1 & 2.2 whereas Angular 5 supports TypeScript version 2.3.
- ★ In Angular 5, we have in-built optimizer that helps to remove unnecessary code from application. But this feature is not present in Angular 4.
- ★ In Angular 4, we can't share the application's state between server and client side .But this is possible in Angular 5 by using "Angular Universal State Transfer API and DOM Support".
- ★ Angular 5 has improved compiler as compared to Angular 4.
- ★ In Angular 4,unnecessary new lines ,white spaces and tabs are created whereas in Angular 5, it is our choice whether we want to insert them or not.
- ★ In Angular 5, we can restrict these unnecessary new lines ,tabs ,white spaces at two levels:
    - at application level
    - at individual component level

★ In Angular 4, we need “i18n” for internationalization in our application. But in Angular 5, we don’t need “i18n” for internationalization because it provides standardization (new date, currency pipe, number etc.) across all browsers.
★ In Angular 5, we can use multiple names for both Directives and Components which is not possible in Angular 4.
★ In Angular 4, we use @angular/HTTP module for all HTTP requests Whereas in Angular 5, we use Httpclient module which comes under @angular/common/http package.
★ In Angular 5, Lambda symbol can be used in place of naming functions.

12. What are the prerequisites for Angular?

Ans. 1. We need to install latest version of "node".

It is basically a runtime environment for executing JavaScript code into the browser. Node provides some tools that we need to build Angular project.

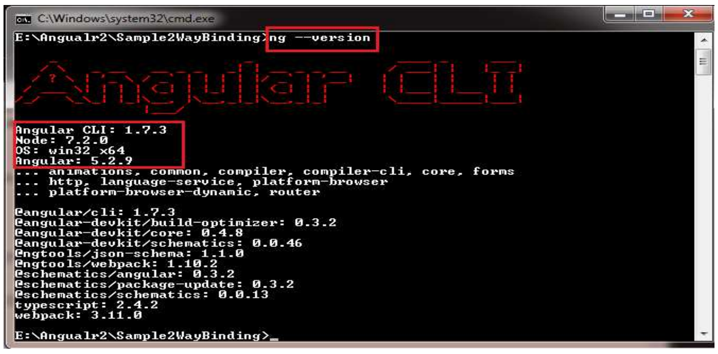
- Install node from nodejs.org (https://nodejs.org/en/)
- minimum version required for building Angular application is 6.9

2. Then, we need a tool called NPM (Node Package Manager) to install any third-party library.

3. Angular CLI (Command Line Interface) is a tool that we use to create a new Angular project.

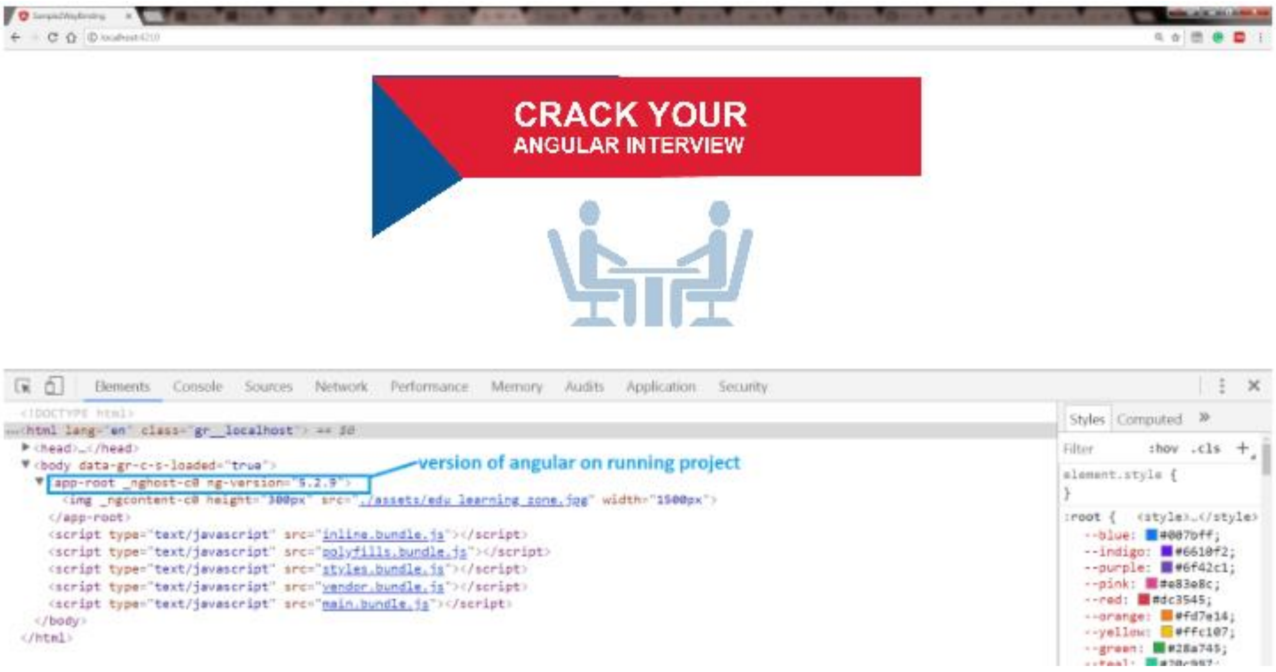
13. How can we check the version of Angular?

Ans. We can check the version of Angular in many ways -
1. By using cmd
ng --v, ng --version, ng –v



```
E:\Angualr2\Sample2WayBinding>ng --version

Angular CLI: 1.7.3
Node: 7.2.0
OS: win32 x64
Angular: 5.2.9
... animations, common, compiler, compiler-cli, core, forms
... http, language-service, platform-browser
... platform-browser-dynamic, router

@angular/cli: 1.7.3
@angular-devkit/build-optimizer: 0.3.2
@angular-devkit/core: 0.4.8
@angular-devkit/schematics: 0.0.46
@ngtools/json-schema: 1.1.0
@ngtools/webpack: 1.10.2
@schematics/angular: 0.3.2
@schematics/package-update: 0.3.2
@schematics/schematics: 0.0.13
typescript: 2.4.2
webpack: 3.11.0

E:\Angualr2\Sample2WayBinding>_
```

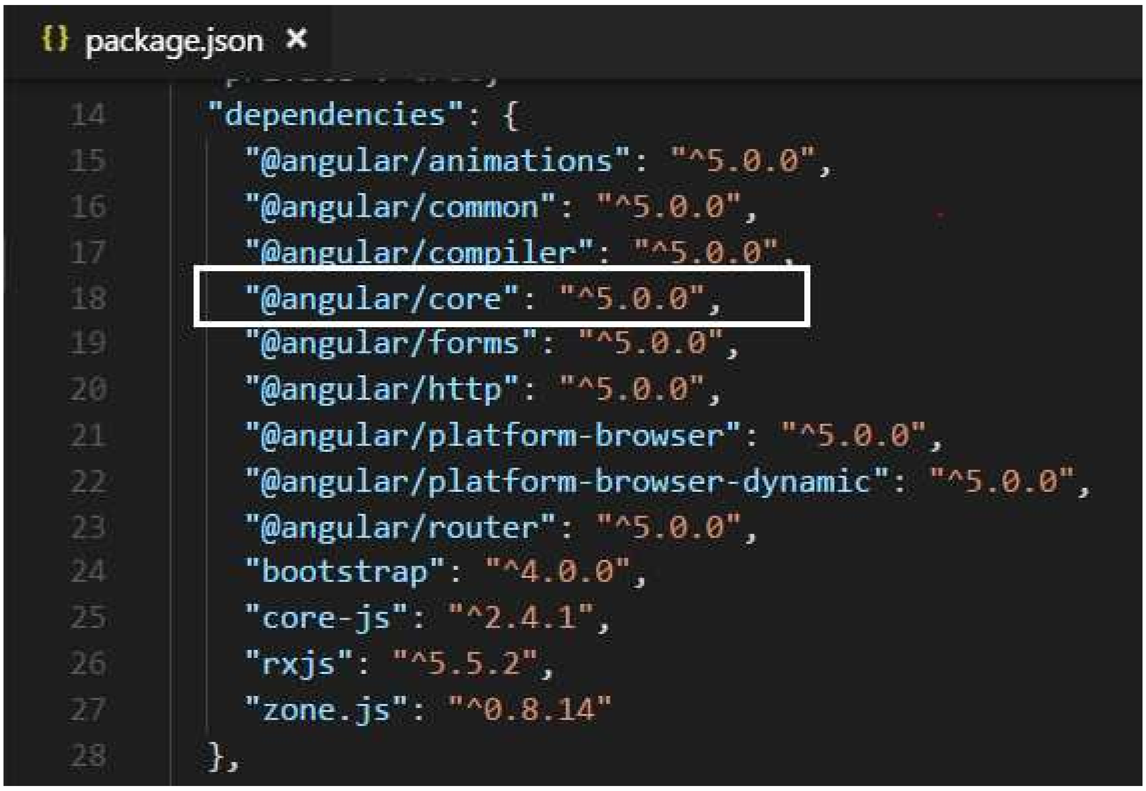
2. By using browser's developer tool (Press F12).



```
<!DOCTYPE html>
…<html lang="en" class="gr__localhost"> == $0
  ▶ <head>…</head>
  ▼ <body data-gr-c-s-loaded="true">
    ▼ <app-root _nghost-c0 ng-version="5.2.9">
        <img _ngcontent-c0 height="300px" src="./assets/edu_learning_zone.jpg" width="1500px">
      </app-root>
      <script type="text/javascript" src="inline.bundle.js"></script>
      <script type="text/javascript" src="polyfills.bundle.js"></script>
      <script type="text/javascript" src="styles.bundle.js"></script>
      <script type="text/javascript" src="vendor.bundle.js"></script>
      <script type="text/javascript" src="main.bundle.js"></script>
    </body>
</html>
```

```
element.style {
}
:root {    <style>…</style>
  --blue: #007bff;
  --indigo: #6610f2;
  --purple: #6f42c1;
  --pink: #e83e8c;
  --red: #dc3545;
  --orange: #fd7e14;
  --yellow: #ffc107;
  --green: #28a745;
```

3. By using the package.json file.



```
"dependencies": {
  "@angular/animations": "^5.0.0",
  "@angular/common": "^5.0.0",
  "@angular/compiler": "^5.0.0",
  "@angular/core": "^5.0.0",
  "@angular/forms": "^5.0.0",
  "@angular/http": "^5.0.0",
  "@angular/platform-browser": "^5.0.0",
  "@angular/platform-browser-dynamic": "^5.0.0",
  "@angular/router": "^5.0.0",
  "bootstrap": "^4.0.0",
  "core-js": "^2.4.1",
  "rxjs": "^5.5.2",
  "zone.js": "^0.8.14"
},
```

14. What are the best IDE for Angular?

Ans. The best IDEs for Angular are the following.

Sublime Text Webstrom Microsoft Visual code ATOM

15. How can we run two Angular project simultaneously?

Ans. For running an Angular project, we use the following command.

ng serve

With the help of this command, the Angular project runs on port number 4200, i.e., localhost:\\4200. Now, if we want to run another project or want to change the port number of the same project, then we can use the following command.

ng serve --port 4210

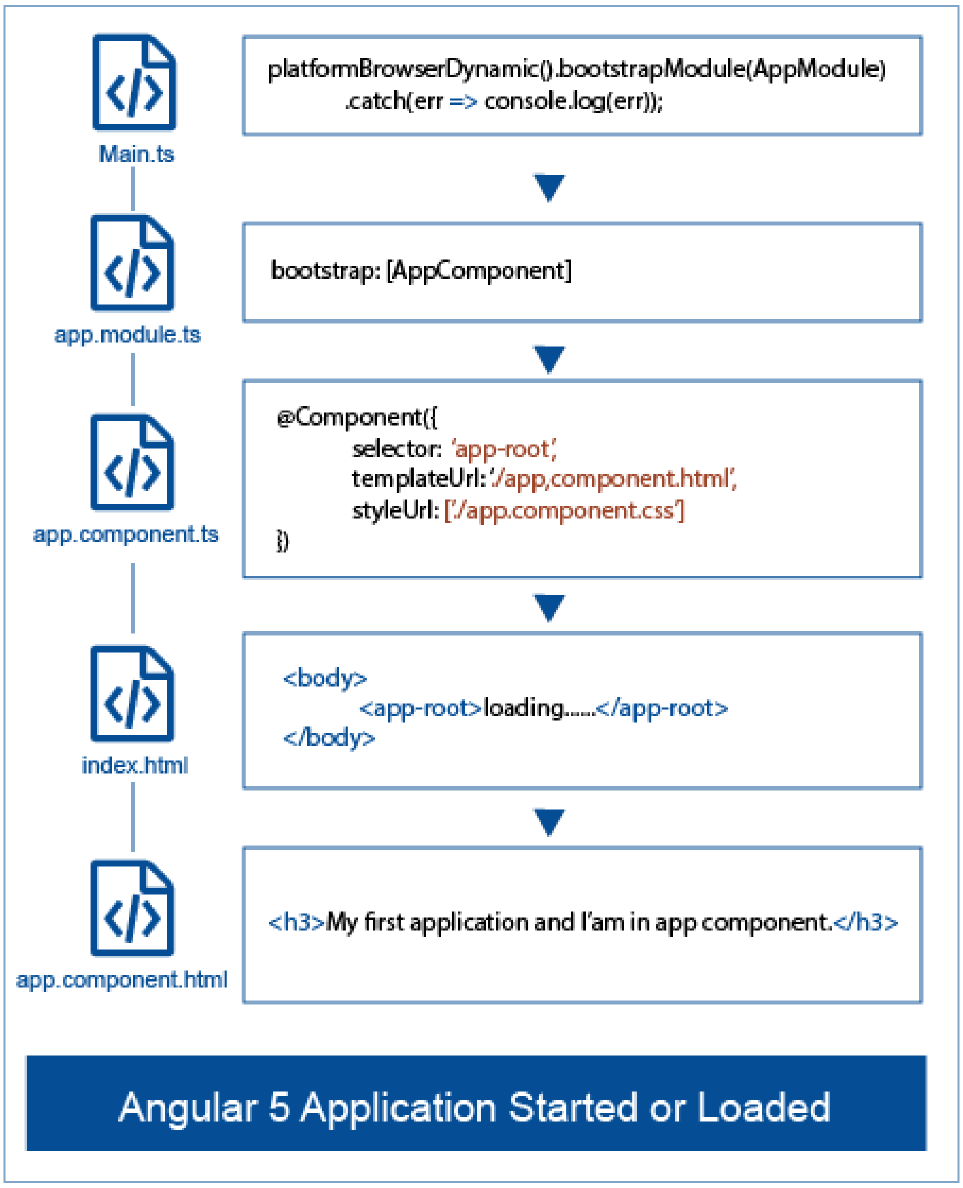
16. What is bootstrapping in Angular? Is it possible to start Angular in any other way rather than app.module? If yes, then how?

Ans. Bootstrapping means starting an Angular application. Yes, it is possible. We can start Angular using our defined component instead of "app.module". For this, we should follow the following steps.

1. Make a module file, say "test.module.ts", in app folder.
2. Make a component using command "*ng g c test*" in app folder.
3. Go to "test.module" file and register your component in this file and use "TestComponent" in declaration and bootstrap.
4. Go to the "main.ts" file, put your module name "TestModule" into bootStrapModule method.
5. Go to "index.html" file and put the test component tag inside body tag.

17. How does an Angular application get start?

Ans. An Angular application gets loaded or started by the following ways.



```
platformBrowserDynamic().bootstrapModule(AppModule)
        .catch(err => console.log(err));
```

```
bootstrap: [AppComponent]
```

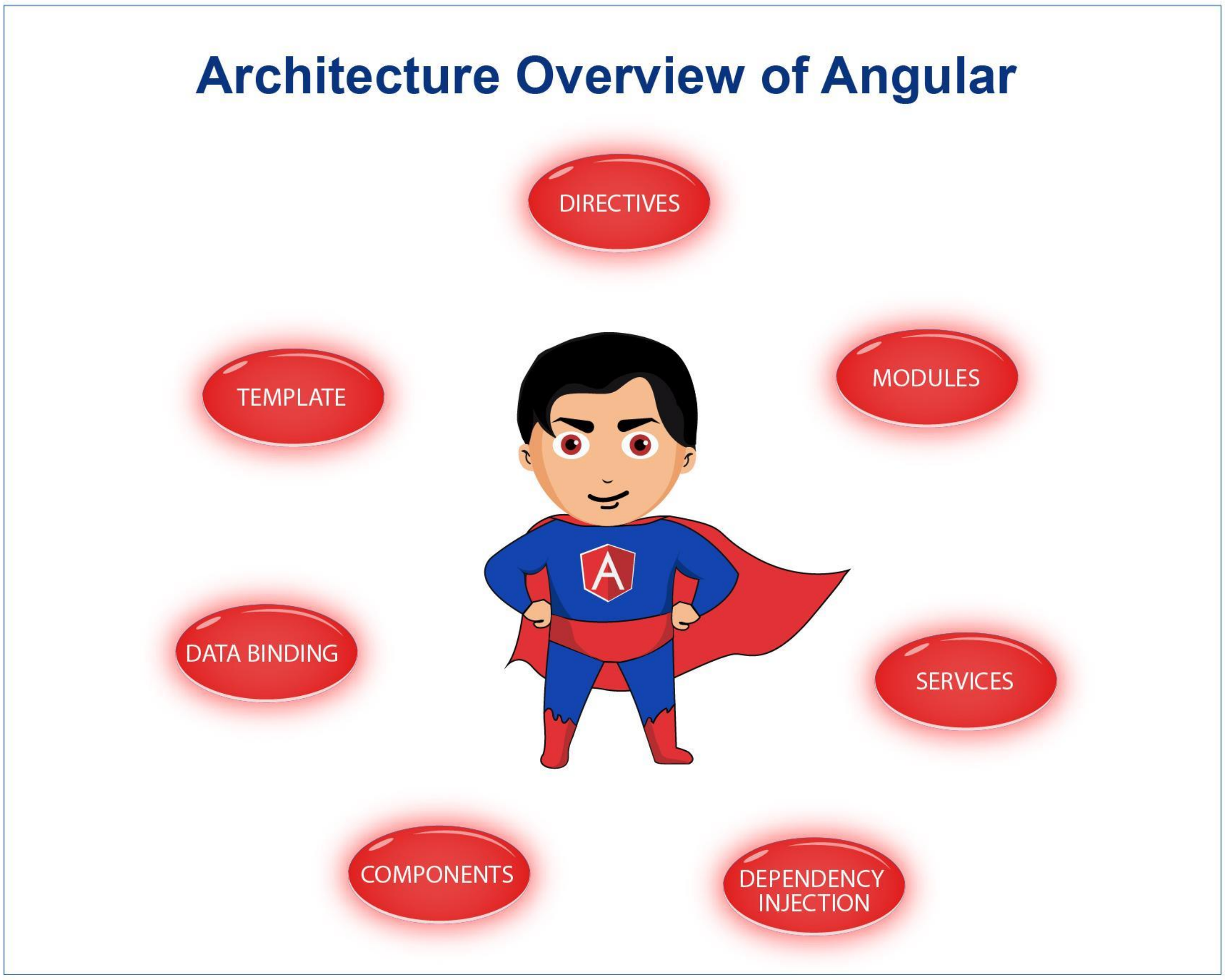
```
@Component({
        selector: 'app-root',
        templateUrl: './app,component.html',
        styleUrl: ['./app.component.css']
})
```

```
<body>
        <app-root>loading......</app-root>
</body>
```

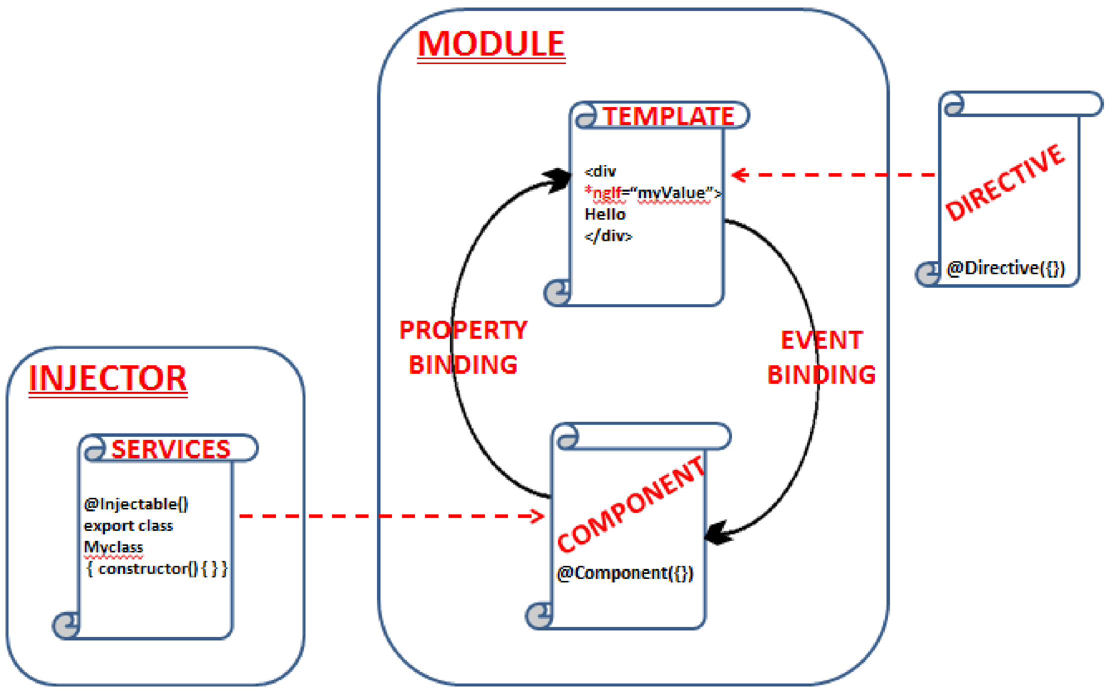
```
<h3>My first application and I'am in app component.</h3>
```

18. What is the Architecture Overview of Angular?

Ans. The main building blocks of an Angular application are defined below.



Flow representation of Angular blocks.



COMPONENT
A component is a TypeScript file which encapsulates the data, the HTML markup, and the logic for a View. You can create as many components as required. Every application has at least one root component which is called app component.
DATA BINDING
Data Binding means to bind the View (HTML content) with Component's field. That is whenever we display dynamic data on a view (HTML) from Component, data binding is used. There are many types of data binding, such as Property Binding, Event Binding, Two-way Binding etc.
TEMPLATE
Basically, a component needs to have a View. To define a View, you can define a "template". User Interface appears due to this template.
MODULES
It is a mechanism to group components, directives, services and other module to perform a particular task.
DEPENDENCY INJECTION
It is a way to create objects that depend upon other objects. Dependency Injection system supplies the dependent objects when it creates an instance of an object.
SERVICES
Services are JavaScript functions that are responsible for performing a particular task. We can use services by the concept of dependency injection.
DIRECTIVES
A directive is used to extend the power of HTML attributes.

19. What is webpack?

Ans. Webpack is a tool used by an Angular CLI to create bundles for JavaScript and stylesheets. Webpack injects these bundles into index.html file at runtime.

20. We cannot reload our Angular application after changing in code. It automatically changes this code. How?

Ans. Just because of **HMR (Hot Module Replacement/Reload)** which is a feature of webpack. Due to HMR, we are able to see our changes without reloading the whole Angular application on a web browser.

21. Which files are bundled and injected in index.html at runtime by webpack?

Ans. The following files are bundled and injected in index.html at runtime.
- inline.bundle.js
- polyfills.bundle.js
- styles.bundle.js
- vendor.bundle.js
- main.bundle.js

22. Can we create the Angular application without Angular CLI?

Ans. Yes, with the help of Visual Studio. Any version of VS can do; such as - Visual studio 2015 or Visual Studio 2017 or later version.

23. What is npm and what is the need for Angular application?

NPM (Node Package manager) is the package manager which we are using for installing the dependencies which is required for the application. In Angular, we have package.json file in which all the dependencies have added which we have installed. When we need any new dependencies, we can simply install using below command.

npm install <package name > --save .

It adds the installed package in node_modules folder and also, adds the dependencies in package.json file.

24. How Webpack is different to SystemJS?

Webpack is a totally different from SystemJS. It does not do same as SystemJS but in case of SystemJS, the systemjs.config.js allows us to configure the way in which module names are matched with their corresponding files. WebPack is a module bundler which bundles a file for application.

25. What is the difference between AOT and JIT?

Ans. In Angular we have AOT (Ahead of Time) and JIT (Just In Time) Compilation for compile the application. There are few differences between JIT and AOT:

1- JIT stands for Just In Time so by the name you can easily understand its compile the code just in time means in the browser while AOT stands for Ahead of Time so according to the name it compiles the code at the time of building.
2- According to compilation, JIT loads the application slowly while AOT loads the application more quickly as compared to that of JIT.
3- In case of JIT, Template Binding errors will show at the time of showing the application while in case of AOT Template binding errors will show at the building time.
4- In case of JIT the bundles size is more than AOT because in case of AOT the bundle size is half of the JIT bundle size.

26. What is transpilation concept of Angular?

Ans. Transpiling is the process to convert the code from one high level language to another high level language.

All modern browsers can only understand JavaScript and in Angular we wrote all code in TypeScript. So, in Angular, transpilation means that what you have written in TypeScript gets converted to another high level language, which is JavaDcript.

27. Can we add more than one component in bootstrap in @NgModule?

Ans. Yes, we can add more than one components within bootstrap in @NgModule.

Commands Based Question

28. How can we run our Angular project? Which is the default port used by Angular?

Ans. We can run our Angular application using "ng serve" command, and by default, Angular uses port no 4200.

29. What is the difference between ng serve and ng serve –open?

Ans. *ng serve* and *ng serve --open* - both the commands are used to run our Angular application. However, if we use "ng serve" to run a project, then we need to open the browser manually and give address "localhost:\\4200" to see the project. In case of "ng serve --open" command, it opens the browser and runs the command automatically.

30. What are the commands for the following?

For creating new component	ng g c MyComponentName
For creating new service	ng g s MyServiceName
For creating new module	ng g m MyModuleName
For creating new directive	ng g d MyDirectiveName
For creating new pipe	ng g p MyPipeName
For creating routing guard	ng g g GuardName
For creating class	ng g cl MyClassName
For creating interface	ng g i MyInterfaceName
For creating enum	ng g e MyEnumName

Folder Structure

31. What is the purpose of main.ts file?

Ans. The main.ts file is the main file which is the start point of our application. As you have read before about the main method the same concepts are here in the Angular application. It's the file for the bootstrapping the application by the main module as .bootstrapModule (AppModule). It means according to main.ts file, Angular loads this module first.

32. What is the role of package.json file?

Ans. It's a most important file for the Angular application. There are many settings in this file including dependencies and devDependencies. When we run npm install, Angular installs all the dependencies as defined in this file.

33. How package.json file is different from package-lock.json file?

Ans. Whenever npm modifies the node_module tree or package.json file, then package-lock.json is automatically generated.

- ✓ This file is intended to be committed into source repositories, and serves various purposes:
- ✓ Describe a single representation of a dependency tree such that teammates, deployments, and continuous integration are guaranteed to install exactly the same dependencies.
- ✓ Provide a facility for users to "time-travel" to previous states of node_modules without having to commit the directory itself.
- ✓ To facilitate greater visibility of tree changes through readable source control diffs.
- ✓ And optimize the installation process by allowing npm to skip repeated metadata resolutions for previously-installed packages.

34. What is the role of tsconfig.json file?

Ans. The tsconfig.json is the configuration file and there are many setting for TypeScript compiler. According to these settings the TypeScript code to compile into Javascript so that browser can understand it .

35. What is the role of Angular-cli.json file?

Ans. The Angular-cli.json is the configuration file of Angular application in which we have many configuration settings which is required for Angular application.

36. What is the use of “assets” folder in Angular?

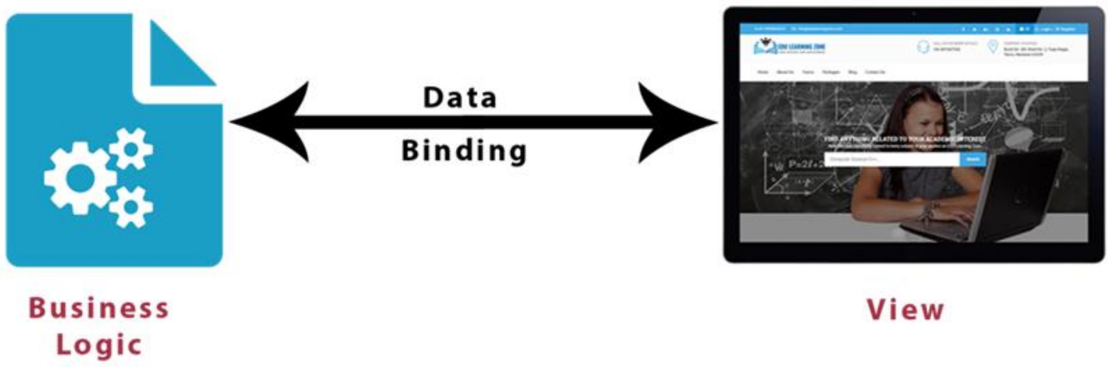
Ans. Whenever we build our Angular application with "npm run build" or "ng build --prod" commands, then Angular CLI moved all of our assets into the dist folder. It will do the same when it sees that there are images inside the assets folder.

So, we can say that assets folder is used in Angular for maintaining the Angular assets as image etc which doesn't have to be modified while compiling.

Data Binding

37. What is data binding in Angular?

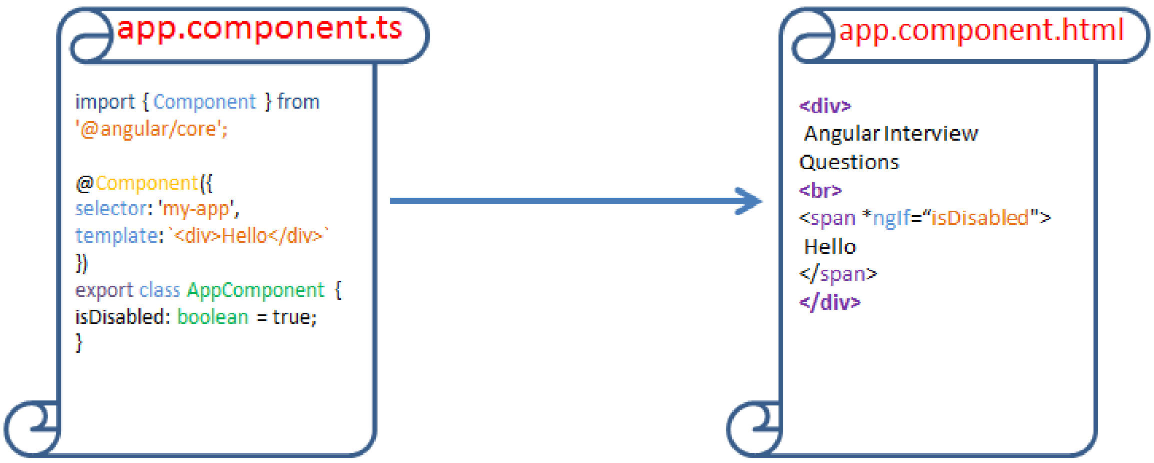
Ans. Data binding is the process that establishes a connection between the application UI and business logic. Basically, it acts as a bridge between UI (View) and the business logic (View Model) for the application.



38. How many types of binding are supported by Angular?

Ans. On the basis of data direction (i.e. movement of data), binding is divided into three parts

1. Source to View



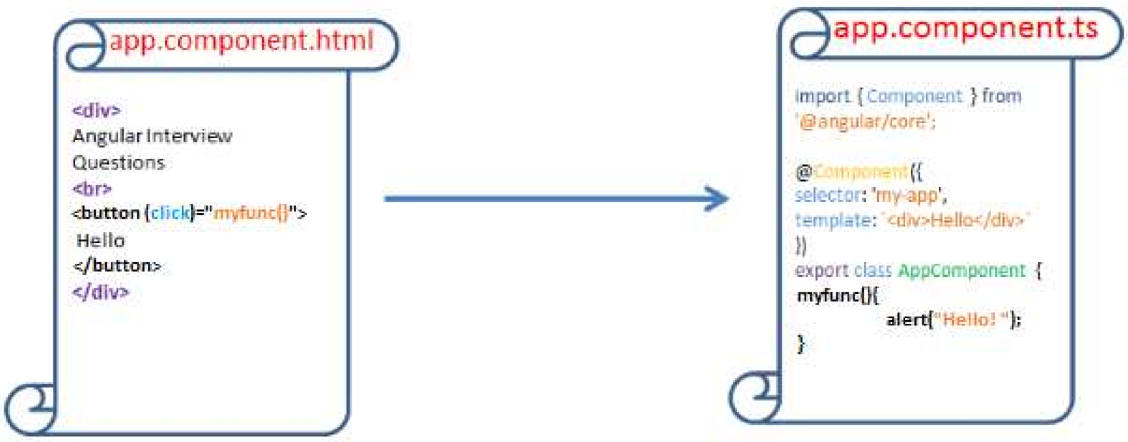
```
import { Component } from
'@angular/core';

@Component({
selector: 'my-app',
template: `<div>Hello</div>`
})
export class AppComponent {
isDisabled: boolean = true;
}
```

```
<div>
 Angular Interview
Questions
<br>
<span *ngIf="isDisabled">
 Hello
</span>
</div>
```

INTERPOLATION **PROPERTY BINDING** **CLASS BINDING**

ATTRIBUTE BINDING **STYLE BINDING**

2. View to Source

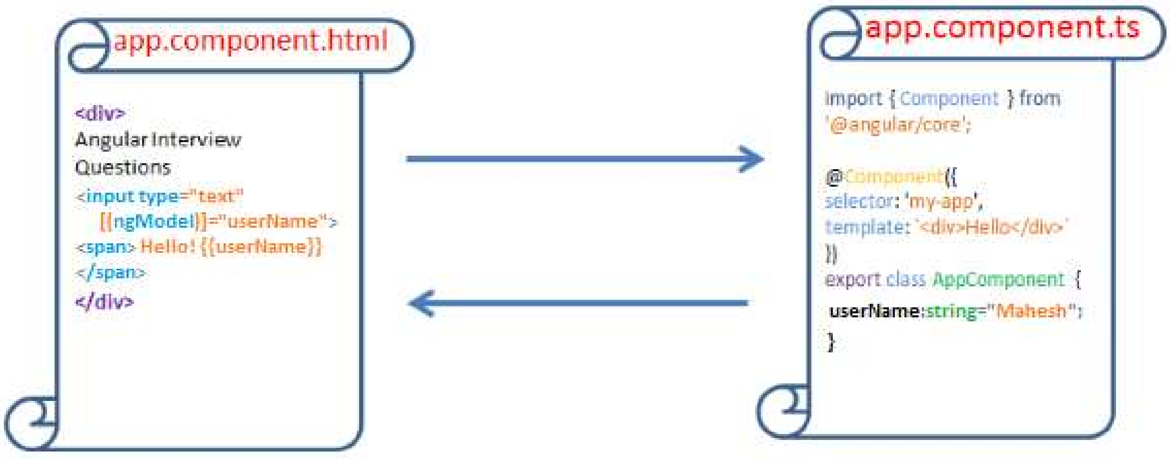


```
<div>
Angular Interview
Questions
<br>
<button (click)="myfunc()">
 Hello
</button>
</div>
```

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

@Component({
selector: 'my-app',
template: `<div>Hello</div>`
})
export class AppComponent {
myfunc(){
          alert("Hello! ");
}
```

EVENT BINDING

3. Two Way Binding



```
<div>
Angular Interview
Questions
<input type="text"
   [(ngModel)]="userName">
<span> Hello! {{userName}}
</span>
</div>
```

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

@Component({
selector: 'my-app',
template: `<div>Hello</div>`
})
export class AppComponent {
 userName:string="Mahesh";
}
```

39. What is Interpolation? Give an example.

Ans. Interpolation allows you to define properties in a component class, and communicate these properties to and from the template. This is the simplest form of data binding. It also helps to solve expression.

For example -

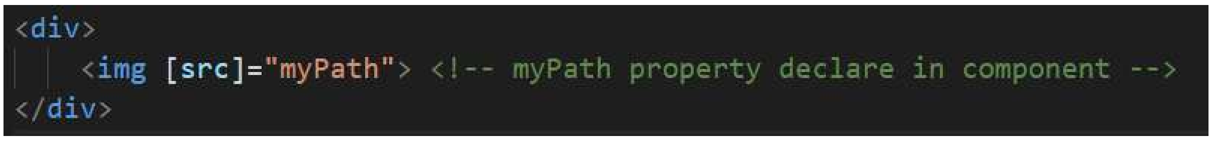
```
<div>
    User Name :- {{userName}} <!-- example of string interpolation> -->
    Sum of 2 and 2 is :- {{2+2}}
</div>
```

40. What is Property binding? Where you can implement property binding?

Ans. In this binding, you pass the data from the Component (.ts file) to View (HTML file) to set the value of a given element.

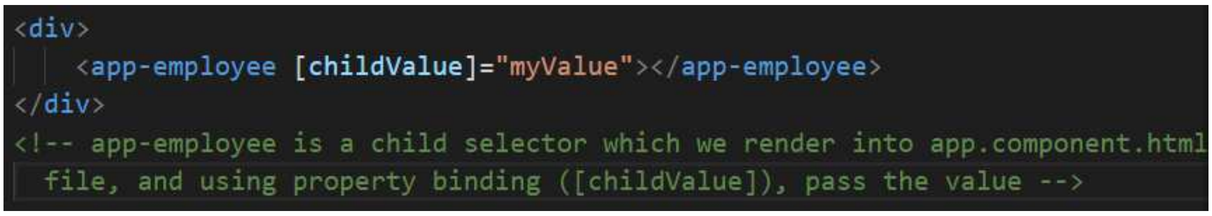
You can implement property binding on the following targets.

1. Element Property - implement property binding on DOM elements like



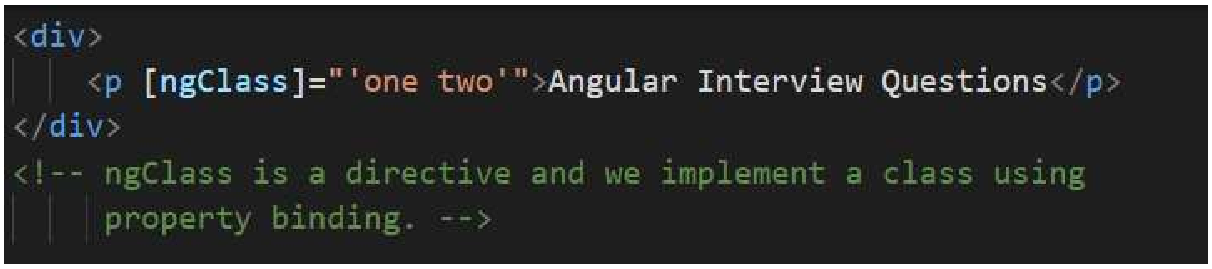
```
<div>
    <img [src]="myPath"> <!-- myPath property declare in component -->
</div>
```

2. Component Property - implement property binding on input elements (want to pass property from Parent's to Child Component) like



```
<div>
    <app-employee [childValue]="myValue"></app-employee>
</div>
<!-- app-employee is a child selector which we render into app.component.html
  file, and using property binding ([childValue]), pass the value -->
```

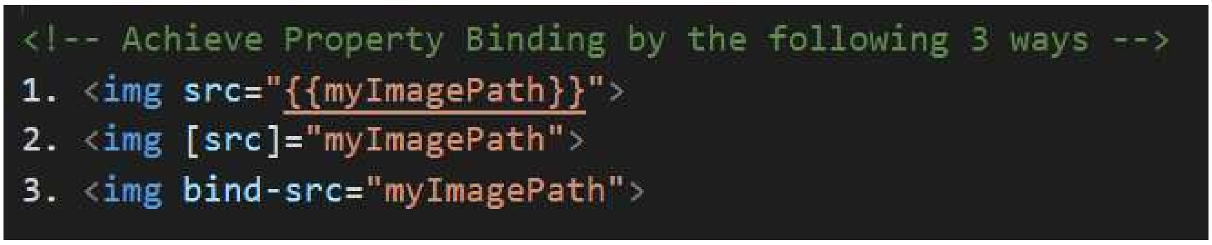
3. Directive Property - implement property binding on directive, like



```
<div>
    <p [ngClass]="'one two'">Angular Interview Questions</p>
</div>
<!-- ngClass is a directive and we implement a class using
      property binding. -->
```

41. How many ways you can achieve property binding? What is the benefit of using property binding?

Ans. You can achieve property binding by three ways.

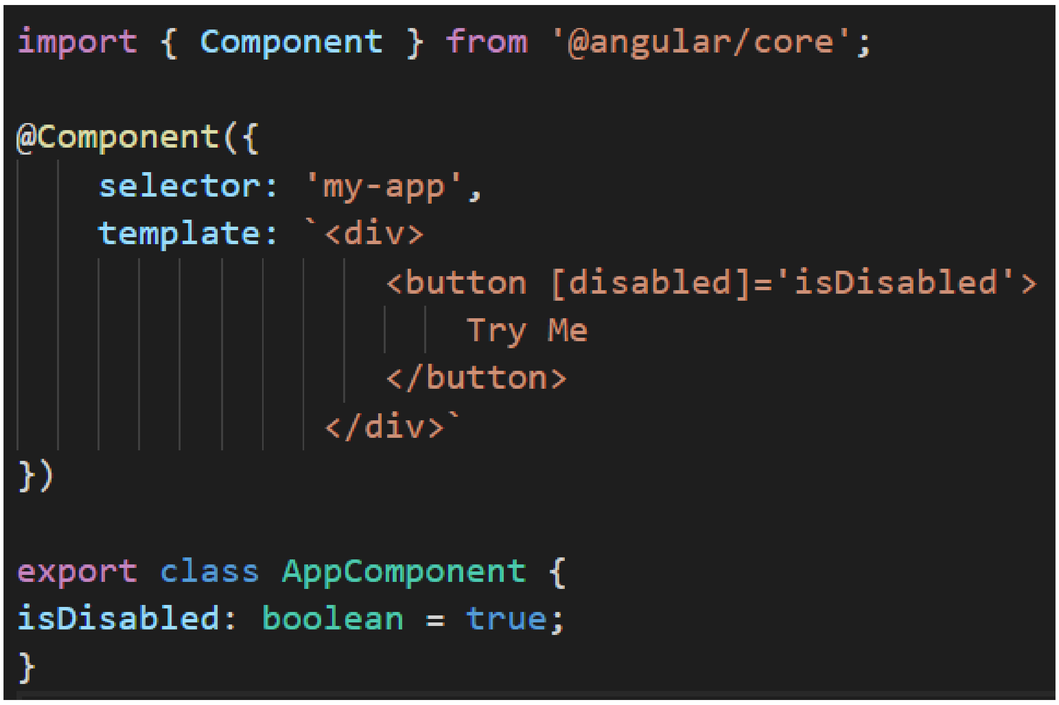


```
<!-- Achieve Property Binding by the following 3 ways -->
1. <img src="{{myImagePath}}">
2. <img [src]="myImagePath">
3. <img bind-src="myImagePath">
```

You can use first method, i.e., property binding using interpolation if and only if the value (myImagePath) you are defining is a string otherwise use other 2 methods.
The benefit of property binding is that you can easily control element property value of a View from the Component (.ts file) and change as per your requirement.

42. When do we use the Property binding and when we use Interpolation?

Ans. When rendering data values as strings, you can choose any one, there is no difference. But, whenever you want to set an element property to a non-string data value, you must use property binding.



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

@Component({
    selector: 'my-app',
    template: `<div>
                <button [disabled]='isDisabled'>
                    Try Me
                </button>
            </div>`
})

export class AppComponent {
isDisabled: boolean = true;
}
```

43. What is Class binding? Give an example.

Ans. Class binding is used when we want to add and remove CSS class names from an element's class attribute based on some condition.

Syntax of class binding is very similar to property binding except it start with the prefix 'class', optionally followed by a dot(.) and the name of a CSS class like

```
<button class="btn" [class.btn-primary]="isApplied" >Hello</button>
```

Here, isApplied is a boolean property which is define inside the Component.

44. What is Style binding? Give an example.

Ans. Whenever we bind a component field to our inline HTML styles; this is called style binding. Suppose, if we want to apply some inline style on the basis of some condition, then we use Style binding.

Syntax of style binding is very similar to class binding except it start with the prefix 'style', optionally followed by a dot(.) and the name of a style. Like

```
<button class="btn" [style.backgroundColor]="isApplied?'#337ab7':'red'" >
  Style
</button>
```

Here, isApplied is a boolean property which is define inside the Component.

45. What is Attribute binding? What is the benefit of using it?

Ans. Attribute binding is used to bind attribute of an element with the field of a component. It is useful when we need to bind attribute,i.e. attribute cannot be bind with property as well as interpolation method. For example –

```
<tr>
  <td colspan="{{1 + 1}}">Employee's Record</td>
</tr>
<!-- Above line generates an error, because "colspan" is an attribute
  not a property, so, we bind it like -->
<td [attr.colspan]="1+1">Employee's Record</td></tr>
```

46. What is Event binding? Give an example.

Ans. Event binding is used to handle events raised from the DOM like keystrokes, mouse movements, clicks and so on. Or we can say that whenever you want to pass data from View (HTML) to Component

Event binding syntax consists of a target event name within parentheses on the left of an equal sign, and a quoted template statement on the right.

Example

The following event binding listens for the button's click events, calling the component's callMyMethod() method whenever a click occurs.

```
<button class="btn btn-primary" (click)="callMyMethod()">
  Click Me
</button>
```

47. Where you can implement event binding?

Ans. You can implement event binding on the following targets.

1. Element Event - implement event binding on DOM elements like,

```
<button (click)="callMyMethod()"> Click Me  </button>
```

2. Component Event - implement event binding on output elements (want to pass property from Child's to Parent Component) like -

```
<app-employee (deleteEmployee)="deleteRecord()"></app-employee>
```

3. Directive Event - implement event binding on directive, like

```
<div (myClick)="clicked=$event" clickable>click me</div>
```

48. Name some common events that you can use for event binding?

Ans. Some common events are:

focus, blur, submit, scroll, cut, copy, paste, keydown, keypress, keyup, mouseenter, mousedown, mouseup, click, dbclick.

49. What is Two-way binding? Give an example.

Ans. Two-way binding means changes in the View (UI) can automatically change in the Model (Component's field) and vice-versa. The syntax of two way binding is different from other binding. Angular use parenthesis inside square brackets to achieve this functionality, like [()].

For achieve two-way binding we need NgModel directive.

Example:

```
<input type="text" [(ngModel)]="userName" placeholder="Enter your name"/>
<p>{{userName}}</p>
```

50. What is template reference variable?

Ans. If you want to access DOM properties of your element then you can use "Template Reference Variable", or we can say, to allow for elements to access other elements from within a template, you can create reference variables on elements, known as template reference variables.

51. How many ways we can create template reference variable?

Ans. You can create template reference variable by two ways i.e. by using "ref" keyword; in the below example, "myVariable" is a template variable.

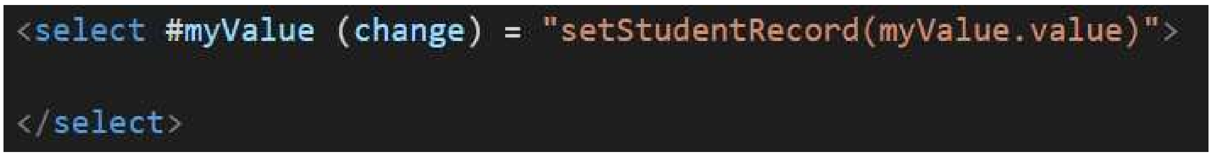
```
<input type="text" placeholder="Click on button" ref-myVariable>
```

By using "#" , in the below example "myVariable" is a template variable.

```
<input type="text" placeholder="Click on button" #myVariable>
```

52. Can we use template reference variable using select (combo box). If yes, then how?

Ans. Yes, we can use template reference variable with select. For example,



```
<select #myValue (change) = "setStudentRecord(myValue.value)">

</select>
```

Look at the above code, #myValue is a template reference variable. The selected value of select box can be accessed by myValue.value.

53. How Angular ensure about content security at the time of binding?

Ans. It's sanitizes the values before displaying them. It will not allow HTML with script tags to leak into the browser, neither with interpolation nor property binding.

For example –

    myTitle= 'Template <script>alert("evil never sleeps")</script>Syntax';

Then, after sanitizes it removes the script tag.

54. What is ngModel? What is the role of ngModel?

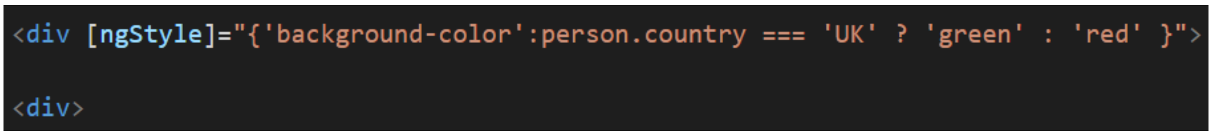
Ans. Angular doesn't come with built-in two-way data binding anymore, but with APIs that allow to implement this type of binding using property and event bindings. ngModel comes as a built-in directive as part of the FormsModule to implement two-way data binding and should be preferred when building components that serve as custom form controls.

55. Which binding we will use for dynamic css?

Ans. NgStyle and NgClass we are using for dynamic css.

For example:

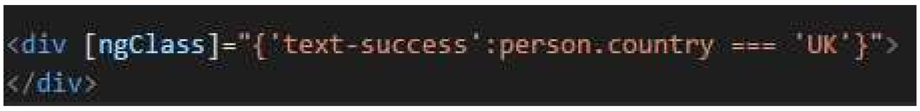
NgStyle:



```
<div [ngStyle]="{'background-color':person.country === 'UK' ? 'green' : 'red' }">

<div>
```

In the above we are checking the country value which is of person object and showing the style according to condition.

NgClass:



```
<div [ngClass]="{'text-success':person.country === 'UK'}">
</div>
```

As you can see in the above, we are checking the country value and according to value applying the class using Ngclass.

Components Based

56. What is a component in Angular?

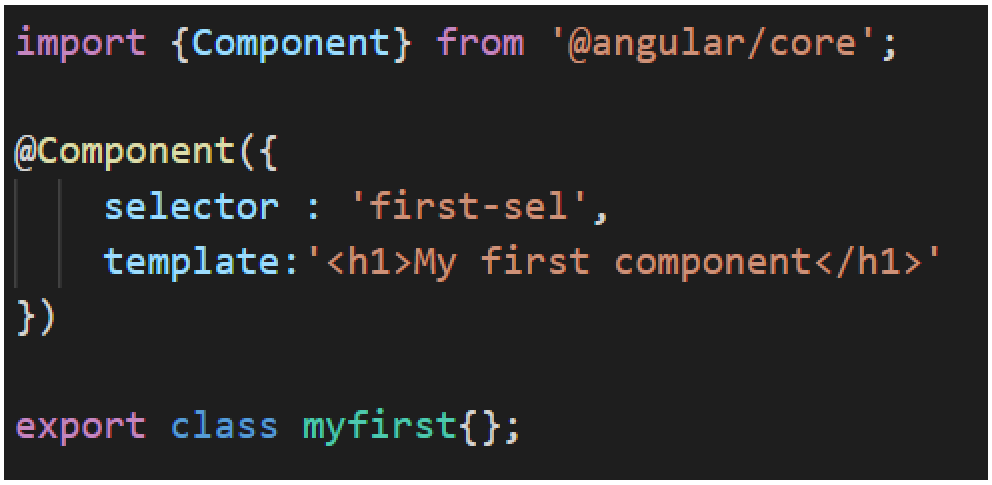
Ans. A component encapsulates the data, the HTML markup, and the logic for a View. You can create as many components as required**.** Angular is a component-based architecture which allows us to work on smaller and more maintainable pieces that can also be used in different places. Every application has at least one root component which is called app component.

57. How can we create a component?

Ans. To create a component, we need to follow these three steps.

1. Create a component.

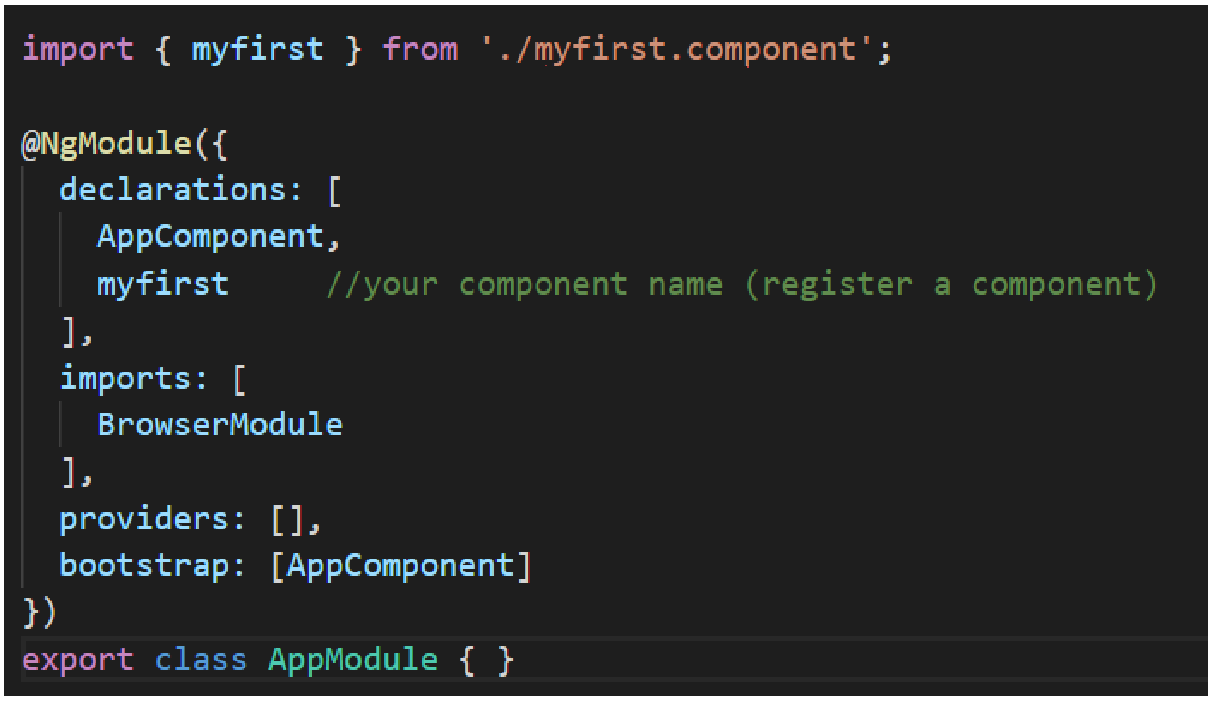
We can create a component with the help of @Component decorator function. For creating a component create any TypeScript file and write the below code.



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

@Component({
    selector : 'first-sel',
    template:'<h1>My first component</h1>'
})

export class myfirst{};
```

2. Register a component in a module.

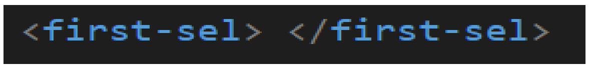
After creating your component as step 1, just register your component into app.module.ts file. Like



```
import { myfirst } from './myfirst.component';

@NgModule({
  declarations: [
    AppComponent,
    myfirst     //your component name (register a component)
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

3. Add an element in an HTML markup.

After create and register your component just call the component selector where you want to render it. Like



```
<first-sel> </first-sel>
```

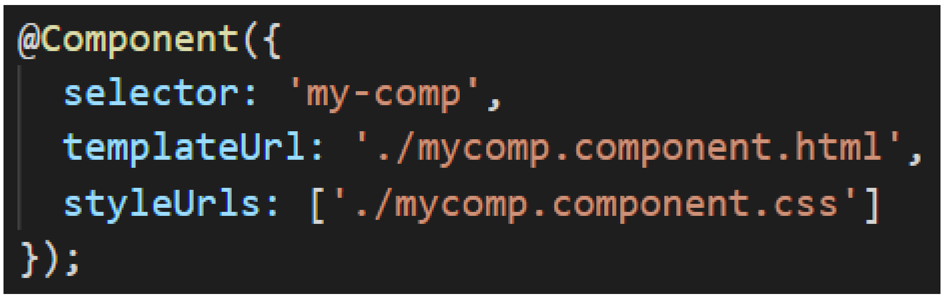
58. Component is directive or not?

Ans. Yes, Component is Directive in which we have template.

59. What is component decorator?

Ans. Component decorator allows you to mark a class as an Angular component and provide additional metadata that determines how the component should be processed, instantiated and used at runtime as you can see below:

Component decorator:



```
@Component({
  selector: 'my-comp',
  templateUrl: './mycomp.component.html',
  styleUrls: ['./mycomp.component.css']
});
```

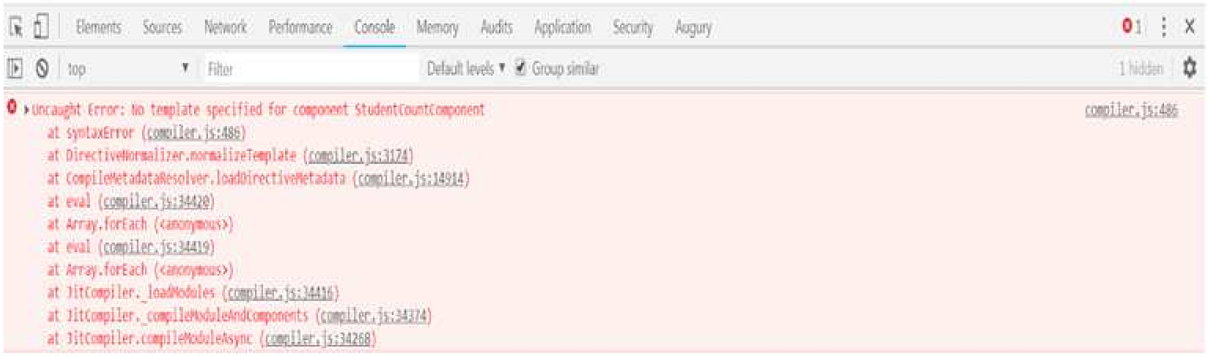
60. What are the different properties of a component object?

Ans. Properties of a component objects are:

1. changeDetection	2. Providers	3. moduleId
4. templateUrl	5. Template	6. styleUrls
7. Styles	8. Animations	9. Encapsulation
10. Interpolation	11. entryComponent	12. preserveWhitespaces

61. What is the mandatory property of @Component() decorator function?

Ans. "selector" and "template or templateUrl" are the mandatory properties for @Component() decorator function. If you miss the selector property then how can you use it, because selector is used to render the template, and, if you miss the template property then it generates compile time error like:



```
Uncaught Error: No template specified for component StudentCountComponent        compiler.js:486
    at syntaxError (compiler.js:486)
    at DirectiveNormalizer.normalizeTemplate (compiler.js:3174)
    at CompileMetadataResolver.loadDirectiveMetadata (compiler.js:14914)
    at eval (compiler.js:34420)
    at Array.forEach (<anonymous>)
    at eval (compiler.js:34419)
    at Array.forEach (<anonymous>)
    at JitCompiler._loadModules (compiler.js:34416)
    at JitCompiler._compileModuleAndComponents (compiler.js:34374)
    at JitCompiler.compileModuleAsync (compiler.js:34268)
```

62. What is ViewEncapsulation?

Ans. ViewEncapsulation decides whether the styles defined in a component can affect the entire application or not. There are three ways to do this in Angular:

Emulated: When we will set the ViewEncapsulation.Emulated with the @Component decorator, Angular will not create a Shadow DOM for the component and style will be scoped to the component. One thing need to know necessary it's the default value for encapsulation.

Native: When we will set the ViewEncapsulation.Native with the @Component decorator, Angular will create Shadow DOM for the component and style will create Shadow DOM from the component so style will be show also for child component.

None: When we will set the ViewEncapsulation.None with the @Component decorator, the style will show to the DOM's head section and is not scoped to the component. There is no Shadow DOM for the component and the component style can affect all nodes in the DOM.

63. What is changeDetection Property?

Ans. When a component is instantiated, Angular creates a change detector, which is responsible for propagating the component's bindings. We can use changeDetection like

changeDetection: ChangeDetectionStrategy

ChangeDetectionStrategy takes two value like

ChangeDetectionStrategy#OnPush sets the strategy to CheckOnce (on demand).
ChangeDetectionStrategy#Default sets the strategy to CheckAlways.

64. What is the difference between templeteUrl and template?

Ans. The templateUrl and template both are the part of @Component decorator. We are using templateUrl when we will add the path of any template file means html file and use template when we need to add only html text of entire html page which required for the component.

65. What is the difference between styleUrl and style?

Ans. The styleUrl and style, both are the part of @Component decorator. We prefer using styleUrl when we add the path of any style file and style when we need to add only required style for the page required for the component.

66. What is the difference between providers present in component and present in app.module.ts file?

Ans. Basically, provider is used to implement the concept of dependency injection (DI). It takes a list of services name. So, if we inject service name into Provider inside the app.module.ts file then it behaves like global service which present all over the application and we can use any component, however, if we inject service name into Provider inside component file then it behave local service which we can only use in current component.

67. What is Dynamic Component?

Ans. Component templates are not always fixed. An application may need to load new components at runtime. So, when we create component in such a way that they can load runtime, these types of component called Dynamic Component.

68. What is the use of ComponentFactoryResolver Service?

Ans. Whenever, we create a Dynamic Component then we need to use this service. This service can be used to render a component instance into another component's template. However, this is similar to the $compile method in AngularJS but this is better than the old $compile method as it encapsulates each template into a child component.

69. Give an example of Dynamic Component?

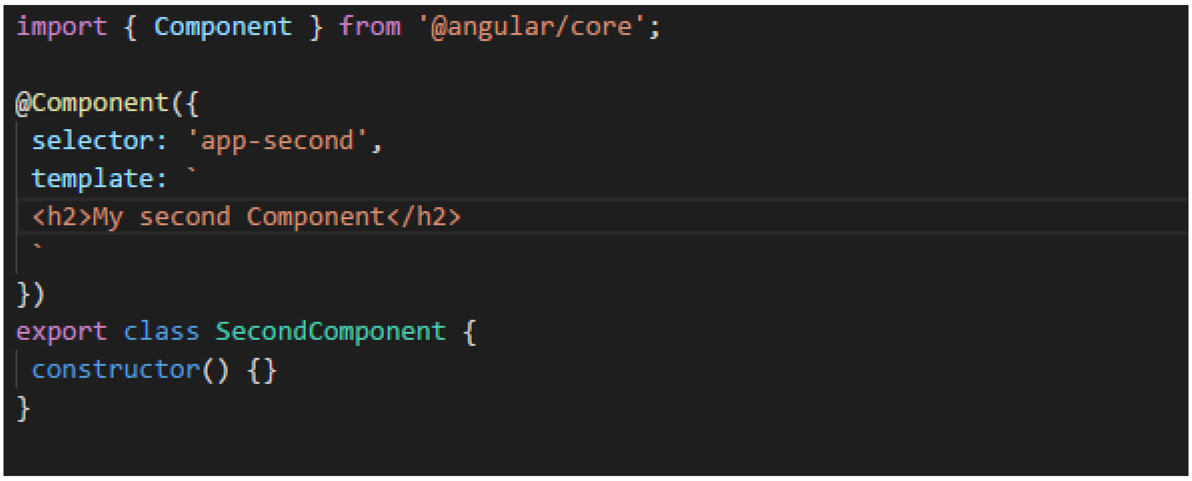
Ans. When we have the requirement, We have one component as app.component.ts with template of html in which we have 2 button as First component and Second component value . Now we have to render the desired component html in the page by click on button means when we click on first button the "First component" html will render and If we will click on "Second component" button, the second component html will render so we will achieve this requirement by Dynamic component using component factory resolver uses .

70. What is nested component?

Ans. When we will create any two component as first component and second component. Now if we will use one component selector within another component so its called nested component:

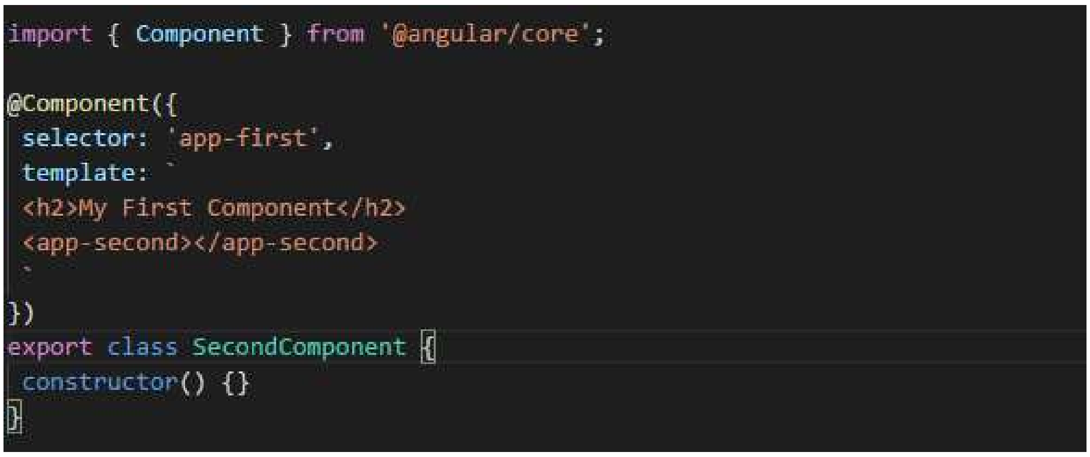
For example:

If we have one component as:



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

@Component({
 selector: 'app-second',
 template: `
 <h2>My second Component</h2>
 `
})
export class SecondComponent {
 constructor() {}
}
```

Another Component as:



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

@Component({
 selector: 'app-first',
 template: `
 <h2>My First Component</h2>
 <app-second></app-second>
 `
})
export class SecondComponent {
 constructor() {}
}
```

You can see in above, I am using the selector of one component with in another component so it’s the example of nested component. We can say one component as Parent as in above First component where we have added the selector of second component and child as Second component.

71. What is the role of selector?

Ans. We are using selector of any component for identifying this component in templates.

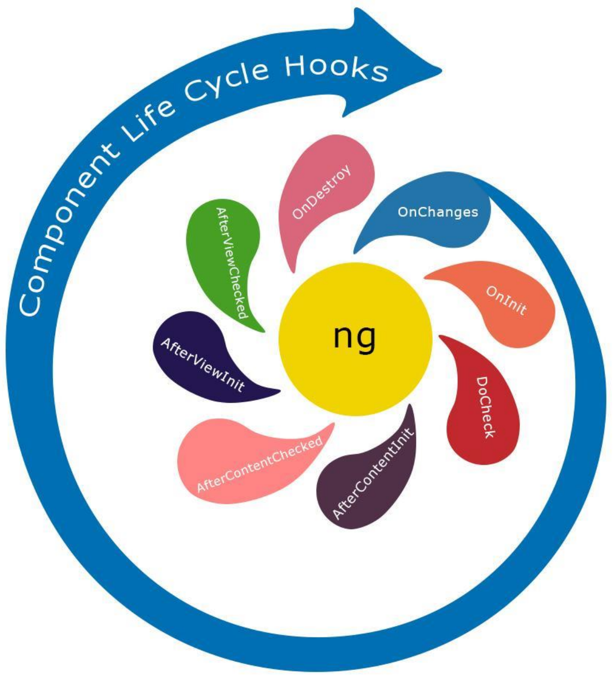
72. What is entry Component?

Ans. The “entry Component” property contains the list of components that will be generated at run time. Adding the component to entryComponents tells the offline template compiler to compile the components so that they can be used at run time.

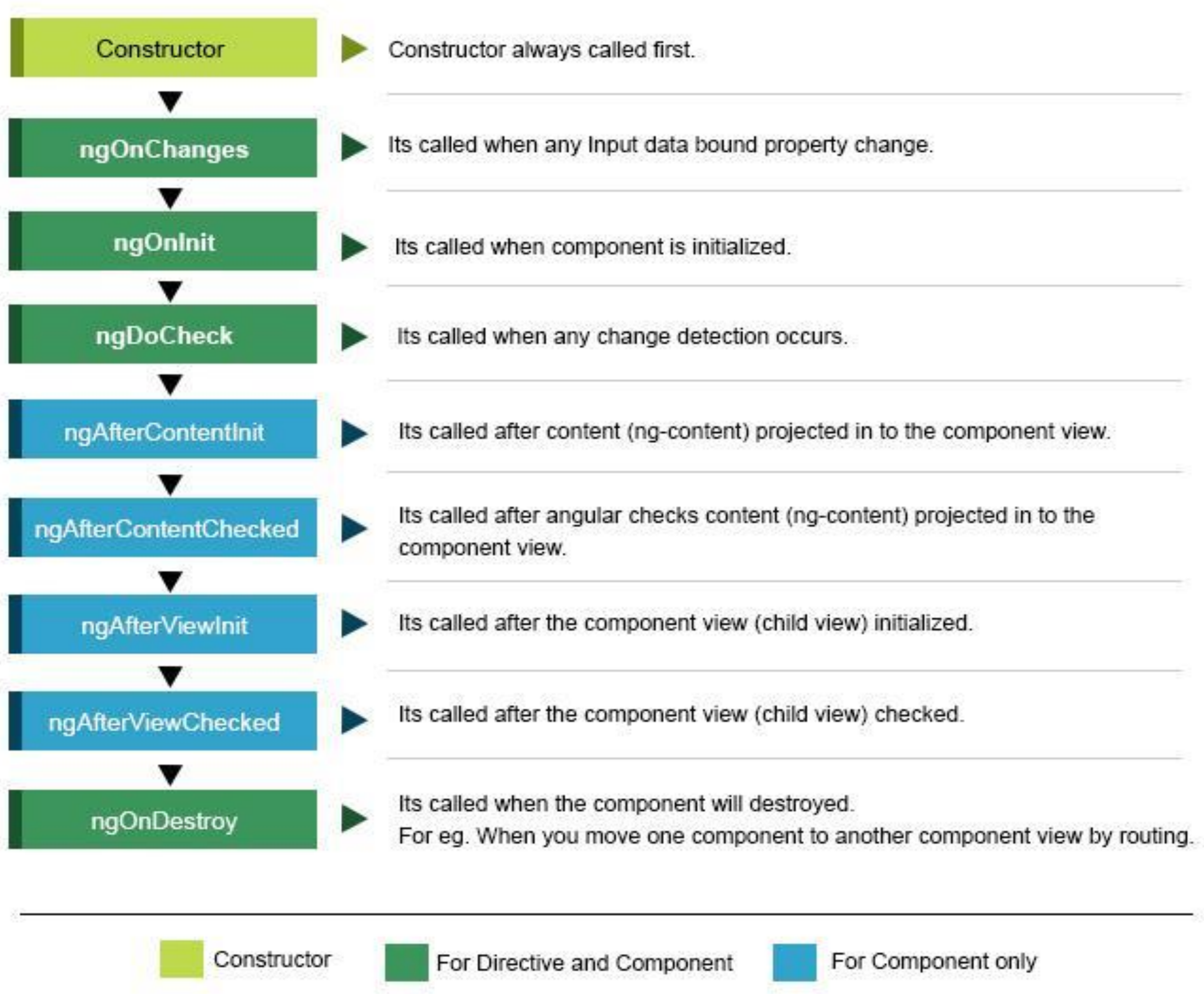
73. What is component hook lifecycle?

Ans. A component also goes through many phases after creation until its end. Angular is responsible for maintaining and taking care of all these phases. In each phase or stage, y*ou can implement your own code or logic*, which helps us a lot.

ANGULAR COMPONENT HOOK LIFECYCLE:



These below are the main details for component lifecycle hook:

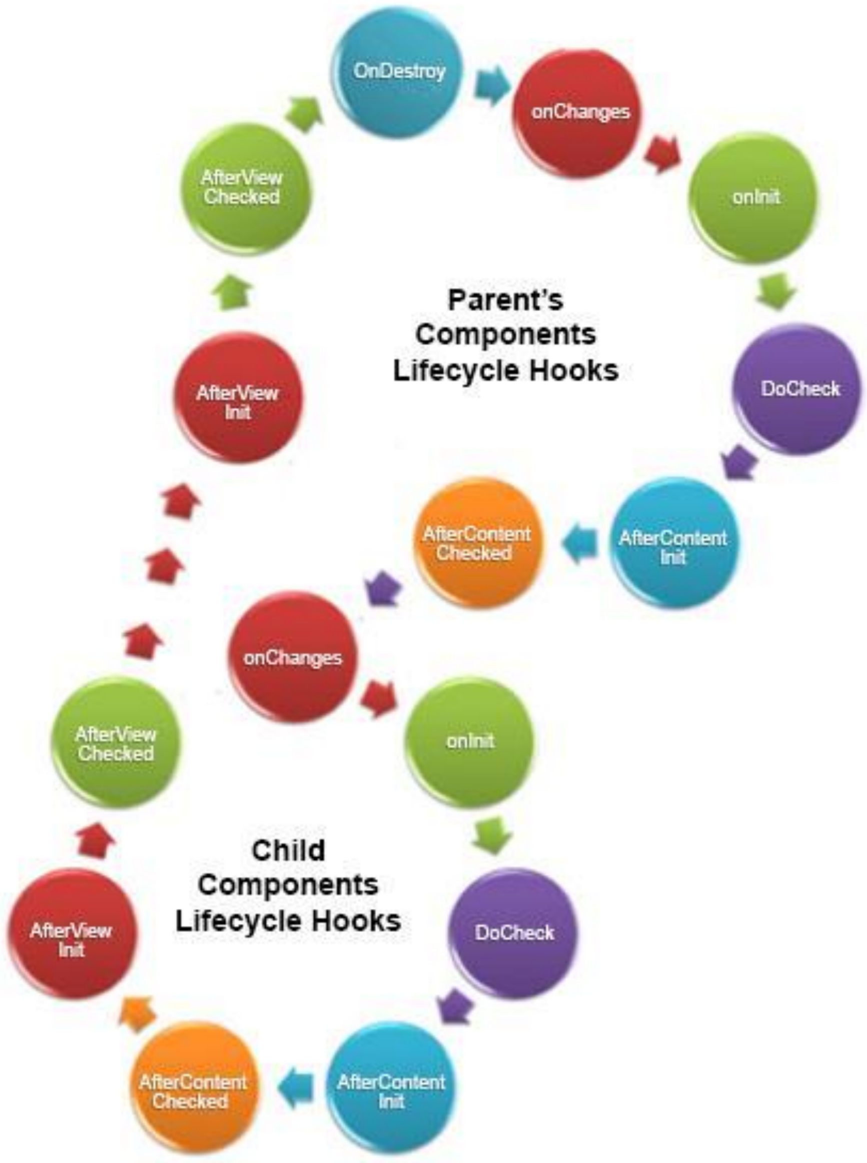


Constructor	Constructor always called first.
ngOnChanges	Its called when any Input data bound property change.
ngOnInit	Its called when component is initialized.
ngDoCheck	Its called when any change detection occurs.
ngAfterContentInit	Its called after content (ng-content) projected in to the component view.
ngAfterContentChecked	Its called after angular checks content (ng-content) projected in to the component view.
ngAfterViewInit	Its called after the component view (child view) initialized.
ngAfterViewChecked	Its called after the component view (child view) checked.
ngOnDestroy	Its called when the component will destroyed. For eg. When you move one component to another component view by routing.

🟨 Constructor 🟩 For Directive and Component 🟦 For Component only

74. If there is nested component (parent and child) then order of lifecycle hook is same or different?

Ans. No the order of lifecycle hook is different as you can show below



75. What is the difference between @ViewChild and @ViewChildren?

Ans. @ViewChild and @ViewChildren both are decorator, The basic difference between both are @ViewChild() provides the instance of another component or directive in a parent component and then parent component can access the methods and properties of that component or directive. In this way, by using @ViewChild() a component can communicate with another component or a directive. But if we want to access multiple child references then we will use @ViewChildren using query list.

76. What is the difference between @ContentChild and @ContentChildren?

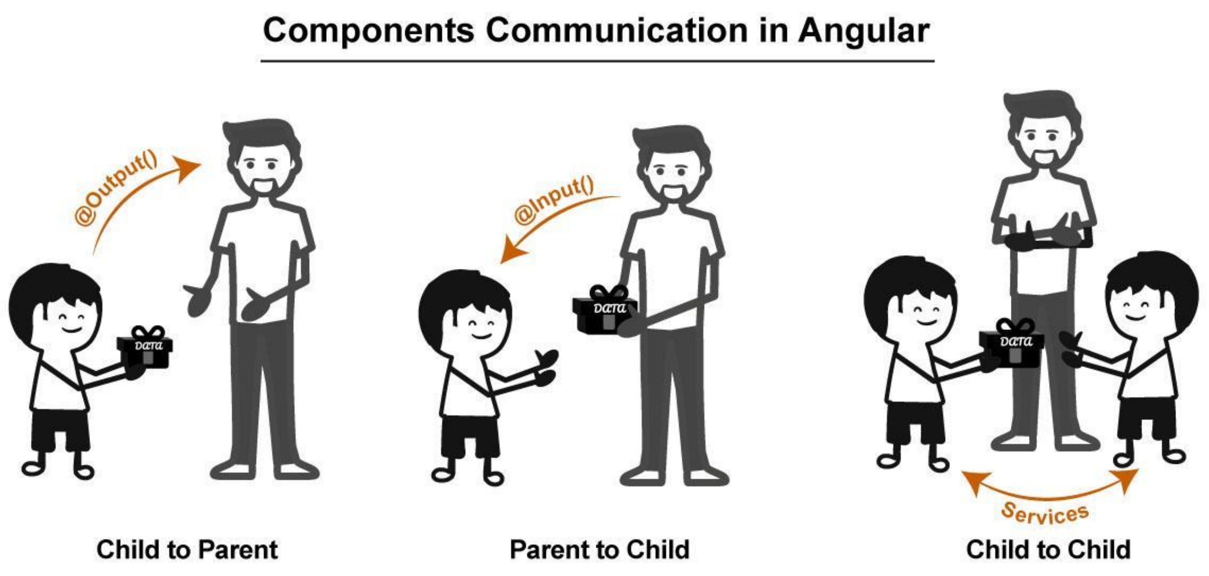
Ans. @ContentChild and @ContentChildren both are decorators and we are using these when we want to fetch single child element or all child elements from content DOM means within <ng-content></ng-content>.

77. What is the difference between ngDoCheck and ngOnChange?

Ans. ngDoCheck is very similar to ngOnChanges hook, the major difference is that ngOnChanges does not detect all the changes made to the input properties. It detects changes for those properties which are passed by value. However, ngDoCheck detects changes for those properties also which are passed by reference such as arrays.

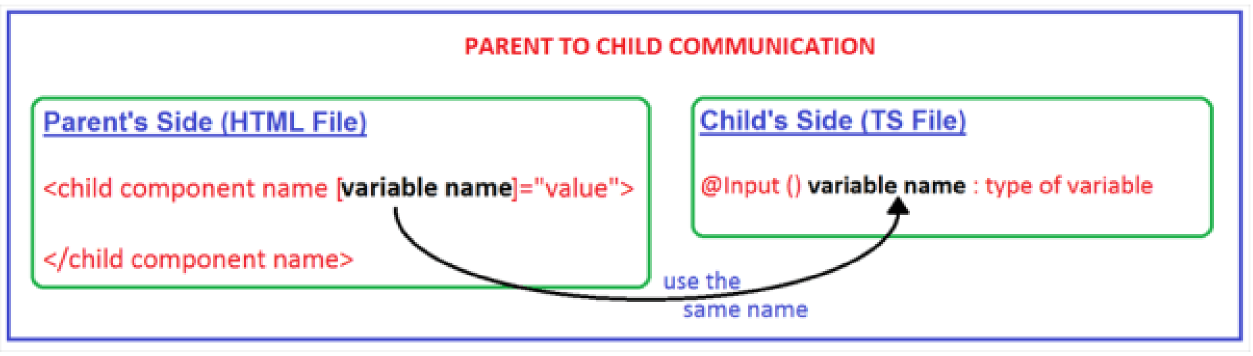
78. How can we pass data from component to component?

Ans.



We can pass the data from component either parent and child relationships or not. If component are separate and we want to pass data between component so we can use services with get and set properties.set for setting the values which you want and get for getting the values which added by one component .

79. How can we pass data from parent component to child component using @Input.
Ans.



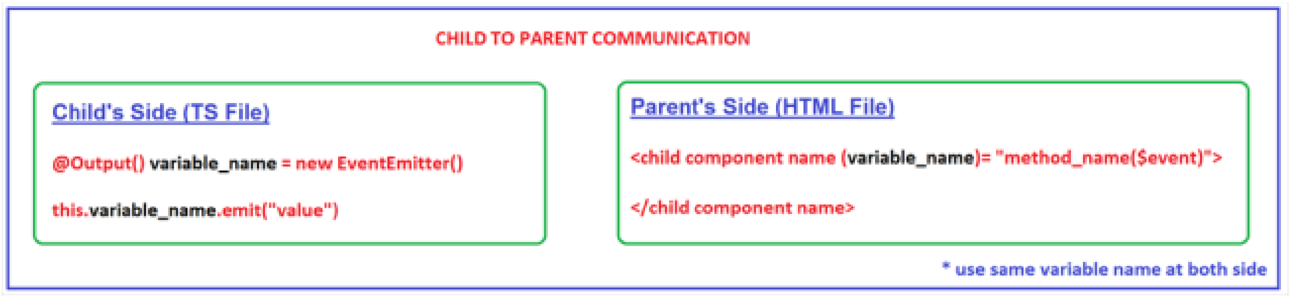
PARENT TO CHILD COMMUNICATION
Parent's Side (HTML File)
<child component name [variable name]="value">
</child component name>
Child's Side (TS File)
@Input () variable name : type of variable
use the same name

As per the above picture, it’s very easy to pass data from Parent to Child, just remember the following steps,

1. At the parent side, Declare a variable & use it as a Property Binding and assign a value, when parent call it, like <child component [variable name]=”value”> </child component>
2. At child side, catch this variable in component (.ts file) using @Input decorator function, like @Input () variable name: type of variable
3. Use it into child’s HTML page.

80. How can we pass data from child component to parent component using @Output.

Ans.



CHILD TO PARENT COMMUNICATION

Child's Side (TS File)

@Output() variable_name = new EventEmitter()

this.variable_name.emit("value")

Parent's Side (HTML File)

<child component name (variable_name)= "method_name($event)">

</child component name>

* use same variable name at both side

As per the above picture, it’s very easy to pass data from Child to Parent, just remember the following steps.

Child’s Side

1. Go to the HTML page, use event binding and call a function to set the value.
2. Go to component page (ts) and declare a property with @Output() like,
@Output () variable name=new EventEmitter();
3. And emit data using this variable like this.variablename.emit(passing-data)

Parent’s Side

1. Go to the HTML page, use the same variable name as it is defined in child component with @Output(), as an event binding <child component selector (variable name)= ”method name ($event)”>
2. Go to the component page and set the value inside the method and use it into parent’s HTML.

81. What is Event Emitter?
Ans. As we are using component in Angular and there are many events which we are using so component emits the event using @output and event emitter. When we pass the value from child to parent, the Parent can emit the value when wanted using event emitter.

For example:

```
@Output() Change: EventEmitter<number> = new EventEmitter<number>();
```

Directives related

82. What is directive?

Ans. Angular directives are used to extend the HTML vocabulary i.e. they decorate html elements with new behaviours and help to manipulate html elements attributes in interesting way.

83.How many types of directives are supported by Angular?

Ans. Angular supports 3 types of directives i.e. Component, Structural, and Attribute.

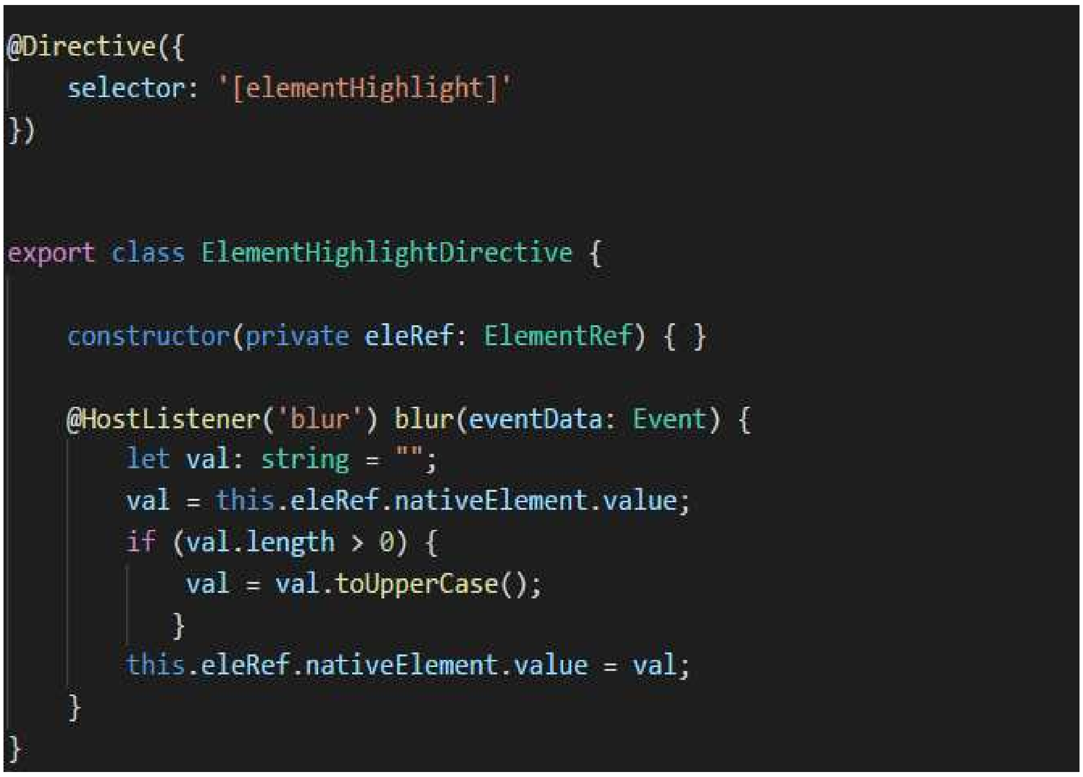
ANGULAR DIRECTIVES
Component
Structural
Attribute

84. What is @Directive decorator?

In Angular we are using @Directive when we will create the custom directive according to requirement of application. In which we will add selector property for directive.

85. Can we create constructor in our directive. If yes then how.

Ans. Yes, we can create constructor in our directive, like this.



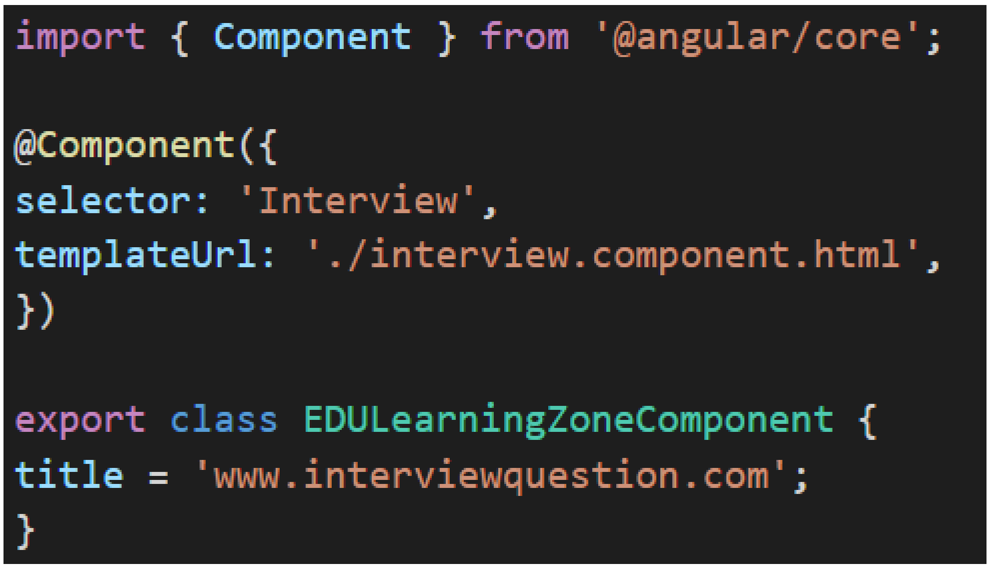
```
@Directive({
    selector: '[elementHighlight]'
})


export class ElementHighlightDirective {

    constructor(private eleRef: ElementRef) { }

    @HostListener('blur') blur(eventData: Event) {
        let val: string = "";
        val = this.eleRef.nativeElement.value;
        if (val.length > 0) {
            val = val.toUpperCase();
        }
        this.eleRef.nativeElement.value = val;
    }
}
```

Here, when the directive gets created Angular can inject an instance of something called ElementRef into this constructor.

86. What is Component directive? Give an example

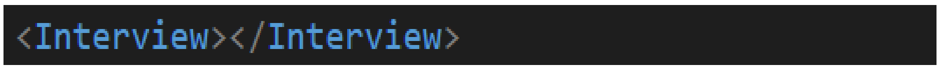
Ans. Component directive is a special kind of directive which is always come up with template (view) and always present in an Angular application. To create a component directive we use @Component() decorator function.

For example -



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

@Component({
selector: 'Interview',
templateUrl: './interview.component.html',
})

export class EDULearningZoneComponent {
title = 'www.interviewquestion.com';
}
```

Now, for using this directive, you should use like this



```
<Interview></Interview>
```

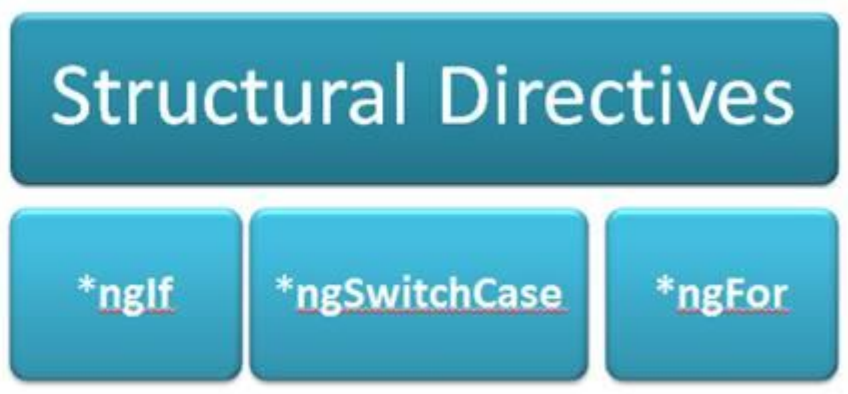
This is what we said, directive with template.

87. What is structural directive?

Ans. Structural directive modifies the structure of DOM by adding or removing DOM elements. Basically, you can say, it works on DOM. These directives always use * as prefix.

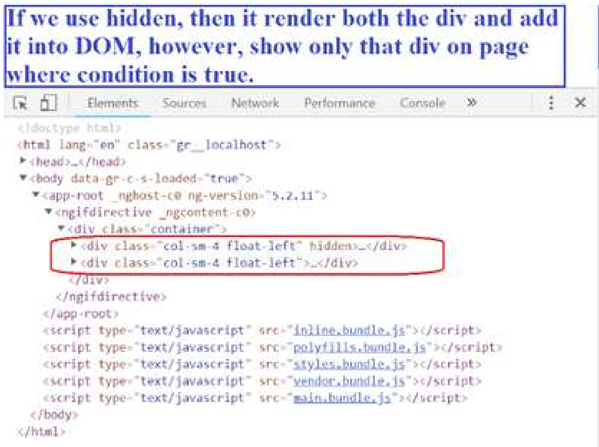
88. Name some structural directives provided by Angular.

Ans.



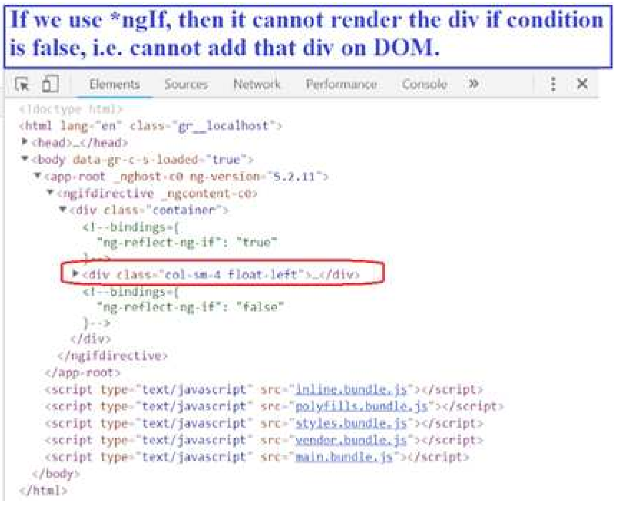
89. What is the difference between ngIf and hidden?

Ans. They both are different due to their rendering methods, i.e. *ngIf can render and add element into DOM if condition is true, whereas hidden property had already rendered and added elements into DOM, it only shows and hides these elements on the basis of condition.



If we use hidden, then it render both the div and add it into DOM, however, show only that div on page where condition is true.

```
<!doctype html>
<html lang="en" class="gr__localhost">
 ▶<head>…</head>
 ▼<body data-gr-c-s-loaded="true">
   ▼<app-root _nghost-c0 ng-version="5.2.11">
     ▼<ngifdirective _ngcontent-c0>
       ▼<div class="container">
         ▶<div class="col-sm-4 float-left" hidden>…</div>
         ▶<div class="col-sm-4 float-left">…</div>
        </div>
      </ngifdirective>
    </app-root>
    <script type="text/javascript" src="inline.bundle.js"></script>
    <script type="text/javascript" src="polyfills.bundle.js"></script>
    <script type="text/javascript" src="styles.bundle.js"></script>
    <script type="text/javascript" src="vendor.bundle.js"></script>
    <script type="text/javascript" src="main.bundle.js"></script>
  </body>
</html>
```

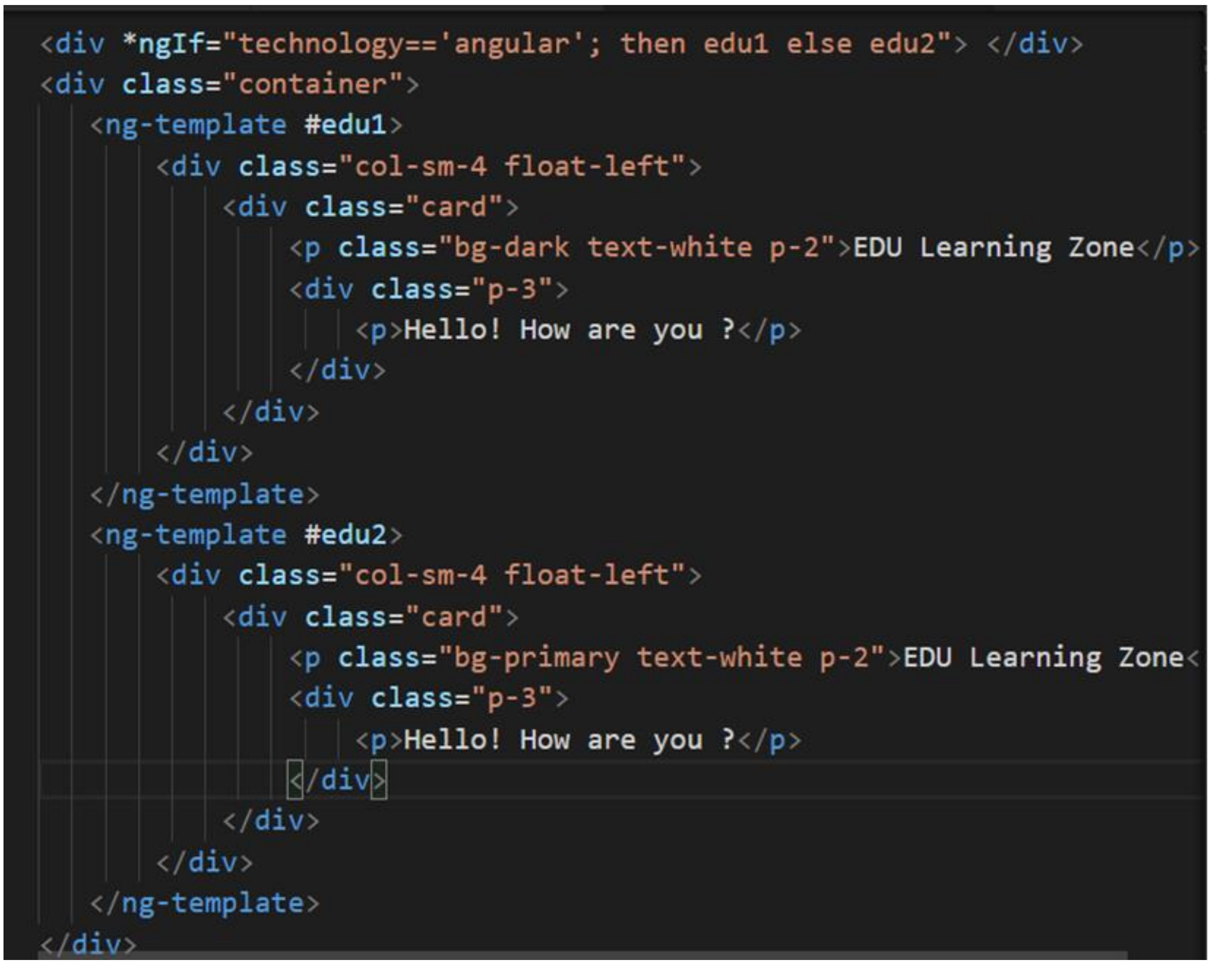


If we use *ngIf, then it cannot render the div if condition is false, i.e. cannot add that div on DOM.

```
<!doctype html>
<html lang="en" class="gr__localhost">
 ▶<head>…</head>
 ▼<body data-gr-c-s-loaded="true">
   ▼<app-root _nghost-c0 ng-version="5.2.11">
     ▼<ngifdirective _ngcontent-c0>
       ▼<div class="container">
          <!--bindings={
            "ng-reflect-ng-if": "true"
          }-->
         ▶<div class="col-sm-4 float-left">…</div>
          <!--bindings={
            "ng-reflect-ng-if": "false"
          }-->
        </div>
      </ngifdirective>
    </app-root>
    <script type="text/javascript" src="inline.bundle.js"></script>
    <script type="text/javascript" src="polyfills.bundle.js"></script>
    <script type="text/javascript" src="styles.bundle.js"></script>
    <script type="text/javascript" src="vendor.bundle.js"></script>
    <script type="text/javascript" src="main.bundle.js"></script>
  </body>
</html>
```

90. How can we use "then" and "else" keywords with *ngIf directive? Explain with an example.

Ans. With the help of "ng-template" and "template reference variables" we can use "else" and "then" keywords with ngIf, because if condition is true then it will execute the "then" part otherwise execute "else" part. For example



```
<div *ngIf="technology=='angular'; then edu1 else edu2"> </div>
<div class="container">
    <ng-template #edu1>
        <div class="col-sm-4 float-left">
            <div class="card">
                <p class="bg-dark text-white p-2">EDU Learning Zone</p>
                <div class="p-3">
                    <p>Hello! How are you ?</p>
                </div>
            </div>
        </div>
    </ng-template>
    <ng-template #edu2>
        <div class="col-sm-4 float-left">
            <div class="card">
                <p class="bg-primary text-white p-2">EDU Learning Zone<
                <div class="p-3">
                    <p>Hello! How are you ?</p>
                </div>
            </div>
        </div>
    </ng-template>
</div>
```

91. What are the keywords that we used at the time of ngSwitchCase and what is the role of that keywords?

Ans. To implement ngSwitch, we basically use 3 keywords.

ngSwitch : It uses binding property and takes an expression that returns the switch value.
ngSwitchCase : It contains the element for matched value.
ngSwitchDefault : If there is no match in ngSwitchCase then it executes.

92. What is *ngFor and what are the exported values of ngFor directive?

Ans. The ngFor directive is used to render a list of objects. For example (create and render a name list). The exported values of ngFor directives are –

index - it gives the index number of the current item.
first - it returns boolean value, i.e. returns true if the item in the list is the first.
last - it returns boolean value, i.e. returns true if the item in the list is the last.
even - it returns boolean value, i.e. returns true if the item has an even index in the list
odd - it returns boolean value, i.e. returns true if the item has an odd index in the list
$implicit :T :- The value of an individual item in the iterable (used with ngForOf)

93. What is the difference between ngFor and ngForOf?

Ans. ngFor and ngForOf are not the different things - they are actually the selectors of the NgForOf directive.

Whenever, we use *ngFor, the Angular compiler converts it into *ngForOf, like below.

We write

```
<div *ngFor="let record of items"> </div>
```

Due to *, first it converts into

```
<template [ngFor]="let record of items">
  <div> </div>
</template>
```

Then, due to "let", which is present in "let record of items", it again compiles and converts into

```
<template ngFor let-item="$implicit" [ngForOf]="items">
  <div>...</div>
</template>
```

94. Why "*" is prefix with structural directive? Can we use structural directive without using *?

Ans. Angular translates the * into a <ng-template> element, wrapped around the host element, for example

```
<div *ngIf="student"class="name">
  {{student.name}}
</div>
```

Angular convert this into

```
<ng-template [ngIf]="student">
    <div class="name">{{student.name}}</div>
</ng-template>
```

Yes, we can use structural directive without using *, but in that case, line of code will increase like

```
<ng-template [ngIf]="student">
    <div class="name">{{student.name}}</div>
</ng-template>
```

95. How many structural directives can we implement on a single element?

Ans. one, if we want to use more than one structural directive on a single element then we need to use <ng-container> like in the below line, there are 2 structural directives on one element i.e. *ngFor and *ngIf, which is not possible

```
<tr *ngFor="let record of studentList" *ngIf="record.gender==‘Male’">
```

So, we should use ng-container like

```
<ng-container *ngFor="let record of studentList">
  <tr *ngIf="record.gender='Male'">
    <td>{{record.name}}</td>
  </tr>
</ng-container>
```

96. How can list items be tracked by default?

Ans. Angular does not has any information about the object so it cannot specify which property it shall use for tracking, so, by default, Angular tracks objects by their identity.

97. Can we provide our own mechanism for tracking the elements? If yes, then how?

Ans. Yes, we can provide our own mechanism for tracking items in a list by using trackBy. We need to pass a function to trackBy, and this function takes two arguments, an “index” and the “current item”. trackBy is used for performance optimization.
For example

```
<ul>
  <li *ngFor="let record of studentList; index as i;  trackBy: studentRecord">
      {{i + 1}}: {{record.studentId}} -
      {{record.name}} - {{record.age}}
  </li>
</ul>
```

In component, define function which is used in track by

```
studentRecord(index: number, student: Student) {
  return student.studentId;
}
```

98. What are attribute directives? Give an example.

Ans. Attribute directives are used as attributes of elements, basically, they change the appearance or behavior of an element, component or another directive. By default, Angular provides two attribute directives. They are:-

NgClass :
By using this we can add or remove CSS class dynamically i.e. on the basis of certain condition.
NgStyle:
By using this we can add or remove styles dynamically i.e. on the basis of certain condition.
Example to implement ngClass and ngStyle

```
<div class="container">
  <div class="col-sm-4 float-left">
    <div class="card" [ngStyle]="{'width':isEven? '400px':'300px'}">
      <p [ngClass]="{'bg-dark text-white p-2':isEven,
      'bg-primary text-white p-2':!isEven}">EDU Learning Zone</p>
      <div class="p-3">
        <p>Hello! How are you ?</p>
      </div>
    </div>
  </div>
</div>
```

Now, if the value of "isEven" property (define in component) is true then it will implement "bg-dark text-white p-2" class, otherwise it will implement "bg-primary text-white p-2".

Similarly, if value of "isEven" property (define in component) is true then it will take width as 400px, otherwise, it will take width as 300px.

99. What is the difference between attribute directive and component directive?

Attribute Directives
Attribute Directives add behavior to an existing DOM element or an existing component instance.

They help us to create reusable component.
@Directive decorator function is used to create an attribute/structural directive.

Component Directives
A component, rather than adding/modifying behavior, actually creates its own view (hierarchy of DOM elements) with attached behavior.

It helps us to break up an Angular application into smaller component.
@Component decorator function is used to create component directives.

100. What is ng-template?

Ans. ng-template is an Angular element for rendering HTML. It is never displayed directly. In fact, before rendering the view, Angular replaces the <ng-template> and its contents with a comment.

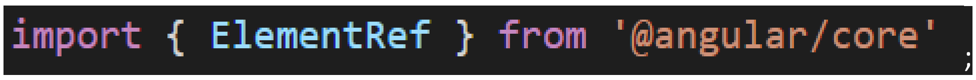
101. What is Host Listner?

Ans. **@HostListener** is the decorator in Angular which we are using during the custom directives creation so you can say, we can use it for events on the host element or component.

102. What is ElementRef?

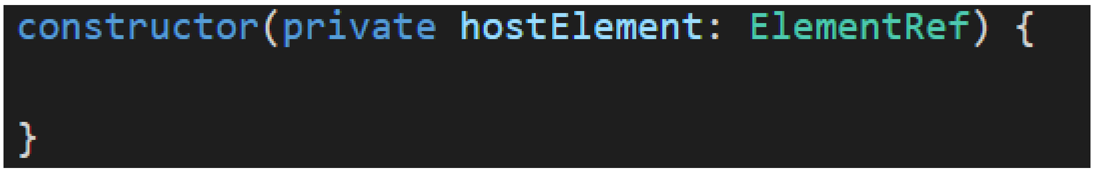
Ans. ElementRef is a class that can hold a reference to a DOM element. In Angular we can use it when we will play with custom directive. We are following below steps when we will use ElementRef:

1- Import the ElementRef from @Angular/core as:



```
import { ElementRef } from '@angular/core';
```

2- Inject it within constructor of component as:



```
constructor(private hostElement: ElementRef) {

}
```

3- Now we can use as DOM manipulation with the help of hostElement which we have added in the constructor.

103. What is the purpose of @HostBinding?

Ans. @HostBinding is also the decorator which we will use at the time of custom directive creation. In Angular, we use it for setting the properties on the element or component that hosts the directive.

Pipes related Questions

104. What are pipes and how we can use pipe in Angular?

Ans. In every application, we fetch data from database and display it on our View, but most of the times, we need to format (transform) the data before displaying it. For this, we use PIPES**.** For example, database gives the date like “Fri Apr 15 1988 00:00:00 GMT-0700” but we want to display on View like April 15, 1988, for this we use pipes.

105. Name some built-in pipe provided by the Angular.

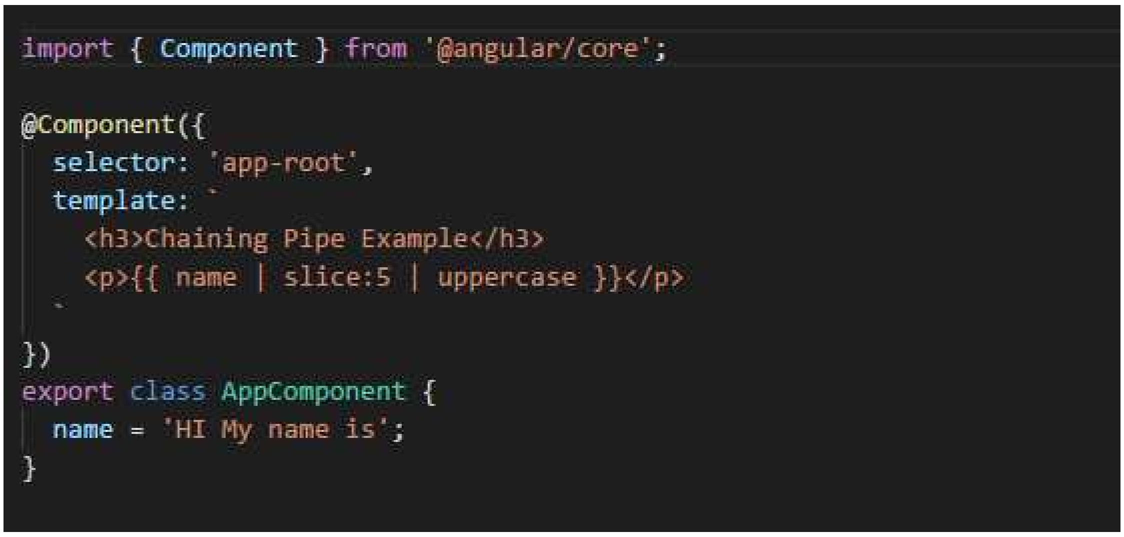
Ans. In Angular we have many built-in pipes in which few are following;
1- Currency
2- Date
3- Decimal
4- LowerCase
6- UpperCase
7- Percent
8-Slice
9- Async

106. Can I create custom pipe in Angular?

Ans. Yes, In Angular we have some built in **pipes**, but you can also build your own pipes. A **pipe** takes in a value or values and then returns a value after transform according to filter criteria which you will add within TypeScript (.ts) file.

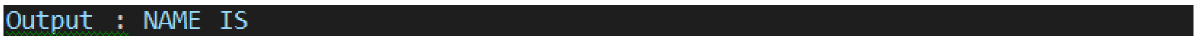
107. What is a Chaining pipe?

Ans. When we use more than one pipe in a single statement then it is called “Chaining pipe”.



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

@Component({
  selector: 'app-root',
  template: `
    <h3>Chaining Pipe Example</h3>
    <p>{{ name | slice:5 | uppercase }}</p>
  `
})
export class AppComponent {
  name = 'HI My name is';
}
```

In the above, we will are using the chaining of pipe and the output will be like below.



```
Output  : NAME  IS
```

108. What is @Pipe Decorator?

Ans. The @Pipe decorator allows you to define the pipe name that you'll use within template expressions. It must be a valid JavaScript identifier.

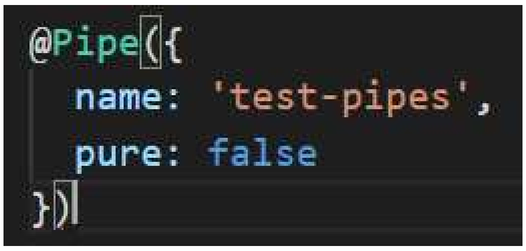
109. How many types of Pipes support by Angular?

Ans. There are two categories of Pipes.

a) Pure Pipes : Angular executes a pure pipe only when it detects a pure change to the input value. A pure change is either a change to a primitive input value (String, Number, Boolean, Symbol) or a changed object reference (Date, Array, Function, Object).
b) Impure Pipes : Angular executes an impure pipe during every component change detection cycle. An impure pipe is called often, as often as every keystroke or mouse-move.

110. How can you define or set your pipe as pure or impure?

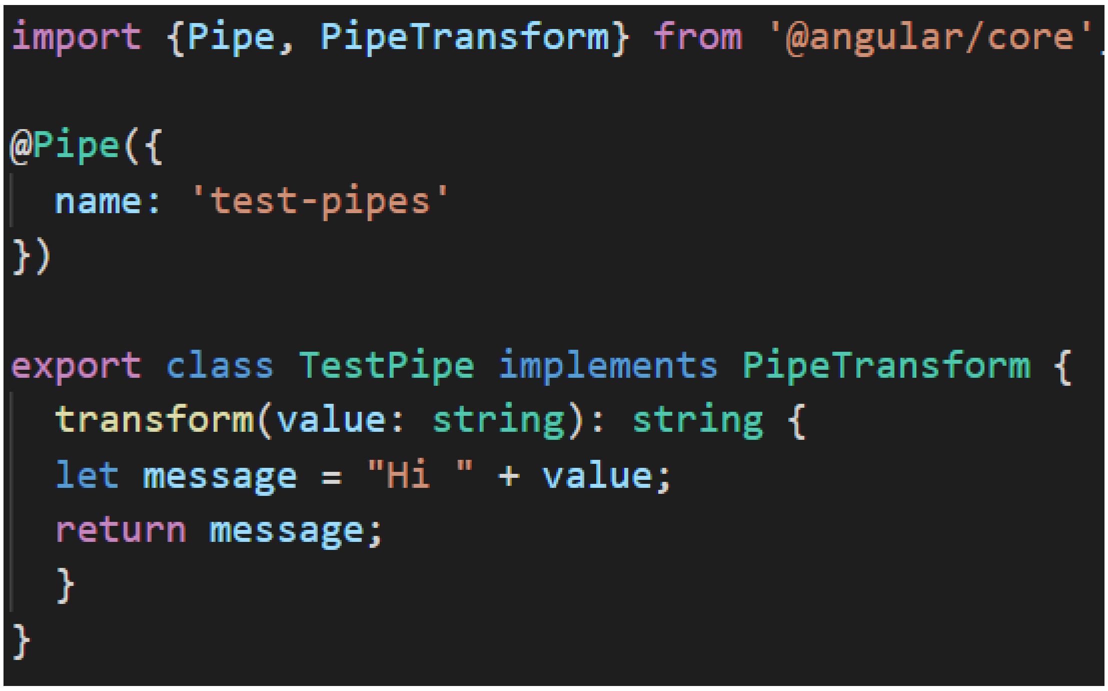
Ans. You make a pipe impure by setting its pure flag to false.



```
@Pipe({
  name: 'test-pipes',
  pure: false
})
```

111. Can I create any pipe in Angular if yes then How?

Ans. Yes, we can create as:



```
import {Pipe, PipeTransform} from '@angular/core'

@Pipe({
  name: 'test-pipes'
})

export class TestPipe implements PipeTransform {
  transform(value: string): string {
  let message = "Hi " + value;
  return message;
  }
}
```

112. What is an AsyncPipe in Angular?

Ans. We can use the temporary property to hold the content return by observable or promise and later them we bind the same content to the template. But if you want to get rid of this temporary property and bind this content directly to the template then you should use AsyncPipe.

Services

113. What is services?

We are using Services in Angular mainly for reusability. If we need same code in many components then we can create the services. There are many uses of services –

1- Communicate between two component when pass the data and both component are separate.
2- Use the HTTP services for get/post/update the required data by an external datasource.

114. HTTP Services?

Ans. HTTP Services is the services for get /post/delete/update the external data using http module .In Angular we need to import the http module for use the http service.

115. What are the different HTTP Verbs supported by Angular.

Ans. GET, POST, PUT, PATCH, and DELETE.

116. What are the difference between Patch() and Put()?

Ans. Patch and Put both will use for update the data, but the major difference is that Put is used for updating all the data (all fields), however, Patch is used for updating partial data (a few fields).

For example-

Suppose, we have a "address form" object in which few fields of address related and if user want to change their address means all the fields of "address from" object so, in this case we will use Put method, but if user want to change only street not locality or area so, in this case we use Patch method.

In short, we can say that

PUT - For update a resource (by replacing it with a new version)
PATCH - For update part of a resource (if available and appropriate)

117. Why we are using services for transfer the data while we have @Input and @Output?

Ans. We are using services for http services and communication between data within component when the component has not the parent and child relationship.

118. How we can send the value from one component to another component using service and there is no relation between a parent and child?

Ans. We can pass the value in this case using services with the help of get and set method.

119. What is providers?

Ans. Providers mainly the instruction for getting the value for dependency injection. In Angular, it's a method by which we can use the singleton pattern. We can add the dependency within providers as services etc. in the providers section of app.module.ts and all the component can use it.

120. What is Dependency Injection (DI)?

Ans. DI is wired into the Angular framework and used everywhere to provide new components with the services or other things they need. Components consume services; that is, you can inject a service into a component, giving the component access to that service class.

121. Explain about @Injectable?

Ans. @Injectable is the decorator which we are using in Angular application so that Angular knows that a class can be used within the dependency injector.

For example when we create the service and we want to use it with application, we will use @Injectable within services when create. However, a component can consume service if service cannot use @Injectable but a service cannot consume other service without this decorator.

122. How many services we can add into Providers?

Ans. It's an array. We can add multiple dependencies with the providers so we can add multiple services with the providers.

123. If we have imported the service in the app.module.ts and then using the service, it will give error or run successfully. If there is an error, then which type of error?

In the above case we will get the error: "**No provider for UserService".**

124. How many decorators are there in Angular?

Ans. We have mainly following decorator as:

1. Class decorators
Class decorator is Top level decorator and they allow us to tell Angular that a particular class is a component, or module.
For example : @Component and @NgModule
2. Property decorator
These are second most common decorators that we are using within Angular application. We can create specific properties with in class with the help of property decorator and use with the class when communication within parent and child or child and parent.
Example : @Input and @Output
3. Parameter decorator
Parameter decorators also important and play a important role within Angular application. We can use when we have to inject the parameter .
Example: @Inject
4. Method decorator.
Method decorator are also a most common decorators that we are using within Angular application. We can create specific methods within our class with required functionality.
Example: @HostListner.

125. What is backtick and how we use it in Angular?

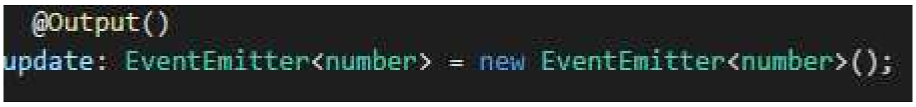
Ans. In Angular 2-4 or later version we are using Backtick for multiple line HTML. For example, if we want the long HTML file data within separate lines, then we can use the Backtick.

126. What is DOM Shadowing?

Dom Showing allows us to hide DOM logic behind other elements. We can say it enables us to apply the style according to scope and requirement of application.

127. If we send any number value from child to parent component then the emitter will number or string?

Ans. In the above case we will use number as below.


```
  @Output()
update: EventEmitter<number> = new EventEmitter<number>();
```

Dependency Injection

128. What is Dependency Injection?

Dependency injection is a design pattern. Angular has its own DI framework, which is typically used in the design of Angular applications to increase their efficiency and modularity.

Dependencies are services or objects that a class needs to perform its function. DI is a coding pattern in which a class asks for dependencies from external sources rather than creating them itself.

129. What is RXJS?

Ans. RxJS (Reactive Extensions for Java Script) is a Library for reactive programming using observables.

130. What is the difference between promises and observable?

Ans. There are following difference between Promises and Observable:

1. Promises handle single event when as async operation completes or fail while Observable is like a stream allows to pass zero or more event.
2. Promises are less preferable while Observable is more preferable than promises because its providers the feature of promises or more.
3. Promises is not cancellable while observables are cancellable when any async operation is not needed more.

131. What is subscribers?

Ans. We are using Subscribers for subscribe the observables.

132. How we can handle the error in Angular?

Ans. We can handle the error using try catch in the Angular application also we can handle the error using error section when subscribe the observables.

ROUTING BASED

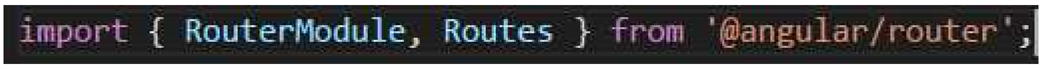
133. What is Routing in Angular?

Ans. Every application consists of many pages, so with the help of Routing we can move from one page to another. Or you can say, Routing helps a user in navigating to different pages using links.

134. How can you define routing in Angular?

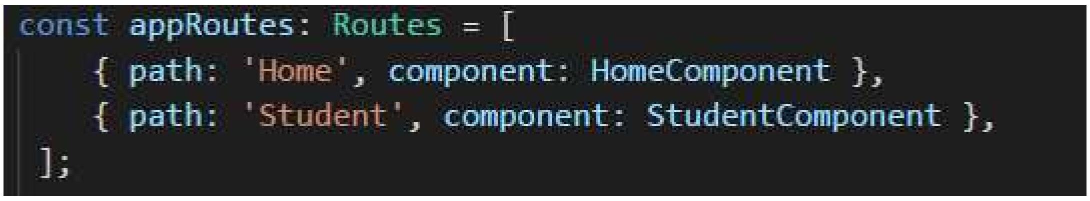
Ans. You can define routes in Angular by the following steps.

1. Import the Routes and RouterModule from @angular/router in app.module.ts, like



```
import { RouterModule, Routes } from '@angular/router';
```

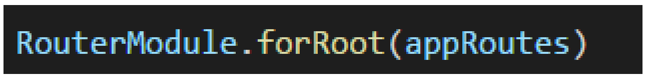
2. Create a const of Routes type and define your routes here, like



```
const appRoutes: Routes = [
    { path: 'Home', component: HomeComponent },
    { path: 'Student', component: StudentComponent },
 ];
```

Import these two components(Home,Student) also.

3. To add the routes in the application used RouterModule.forRoot inside the @NgModule, like



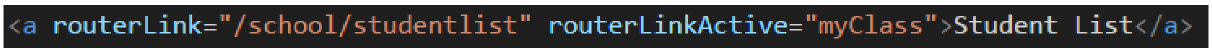
```
RouterModule.forRoot(appRoutes)
```

135. What is a RouterOutlet?

Ans. RouterOutlet is a substitution for templates rendering the components. In other words, it represents or renders the components on a template at a particular location.

136. What is Router links?

Ans. routerLink is a directive which is used in view (html) to tell the Angular that which route need to navigate. For example-



```
<a routerLink="/school/studentlist" routerLinkActive="myClass">Student List</a>
```

137. What is Router State?

Ans. After the end of each successful navigation life cycle, the router builds a tree of ActivatedRoute objects that make up the current state of the router. With the help of 'routerState' property you can access state of routes at anywhere in the application.

138. What are Router events?

Ans. During each navigation, the Router emits navigation events through the Router.events property. These events range from when the navigation starts and ends too many points in between.

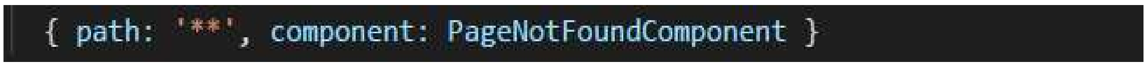
Some of the router events are as follow:

NavigationStart
An event triggered when navigation starts.
RoutesRecognized
An event triggered when the Router parses the URL and the routes are recognized.
RouteConfigLoadStart
An event triggered before the Router lazy loads a route configuration.
RouteConfigLoadEnd
An event triggered after a route has been lazy loaded.
NavigationEnd
An event triggered when navigation ends successfully.
NavigationCancel
An event triggered when navigation is canceled. This is due to a Route Guard returning false during navigation.
NavigationError
An event triggered when navigation fails due to an unexpected error.

139. What is Wildcard route?

Ans. Add a wildcard route to intercept invalid URLs and handle them gracefully. A wildcard route has a path consisting of two asterisks. It matches every URL. The router will select this route if it can't match a route earlier in the configuration. A wildcard route can navigate to a custom "404 Not Found" component or redirect to an existing route.

The router selects the route with a first match wins strategy. Wildcard routes are the least specific routes in the route configuration. Be sure it is the last route in the configuration.



```
{ path: '**', component: PageNotFoundComponent }
```

140. What is pathMatch property in routing?

Ans. pathMatch is a property which tell the router that how to match a URL to the path of a route. You can set pathMatch property value as a "full" or as a "prefix".

The **pathMatch:'full'** results in a route hit when the remaining, unmatched segments of the URL match ''.

The **pathMatch:'prefix'** which tells the router to match the redirect route when the remaining URL begins with the redirect route's prefix path.

```
const appRoutes: Routes = [
  { path: 'home', component: HomeListComponent },
  { path: 'test', component: TestListComponent },
  { path: '',   redirectTo: '/test', pathMatch: 'full' },
  { path: '**', component: PageNotFoundComponent }
];
```

141. How can we pass parameter in Routing?

Ans. 1. On html page, use like

```
<a [routerLink]="['/Student', student.id]">Student Record</a>
```

2. After that, in component (.ts) file

a) Inject ActivatedRoute in constructor like

```
constructor(private route : ActivatedRoute) {}
```

b) After that, use like

```
 this.route.params.subscribe(
   (params : Params) => {
     this.id = params["id"];
   }
);
```

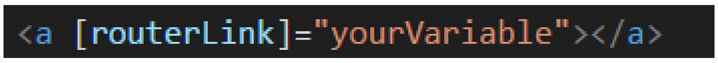
142. What are the guard interfaces supported by router?

Ans. The router supports multiple guard interfaces:

CanActivate to mediate navigation to a route.
CanActivateChild to mediate navigation to a child route.
CanDeactivate to mediate navigation away from the current route.
Resolve to perform route data retrieval before route activation.
CanLoad to mediate navigation to a feature module loaded asynchronously.

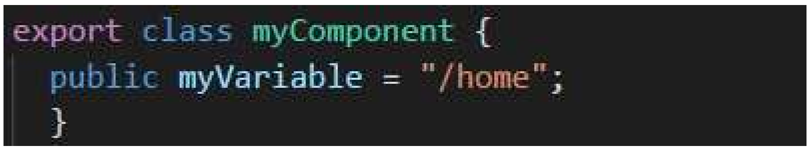
143. Difference between [routerLink] and routerLink?

Ans. Whenever you want to pass url as a dynamic then we use routeLink in square bracket as a property binding, like below.



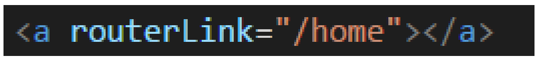
```
<a [routerLink]="yourVariable"></a>
```

So this variable (yourVariable) could be defined inside your class and it should have a value like below:



```
export class myComponent {
  public myVariable = "/home";
  }
```

However, when we use without bracket you are passing string only and you can't change it, it's static (hard coded).



```
<a routerLink="/home"></a>
```

Form Based Questions

144. What is Form and how many strategies we have for developing the forms?

Ans. Forms are required for login to the application, show the request in the page and submit the data and perform the operation within the page which one required in the application.

We have mainly 2 strategies which we are using for develop the form.

1. Template Driven
2. Reactive

145. What is Template driven and reactive form and why we are using in Angular?

In Angular, when we work with form for get, post, and update etc. the data, we have two approaches - either we can use Template driven form or the Reactive form approach.

Template Driven
Template driven forms are forms where we write all the logic, validations, controls etc. within the template part of the code (html code).

Reactive
Reactive form is basically a model driven. In this approach, you can add your logic to the component class and your templates.

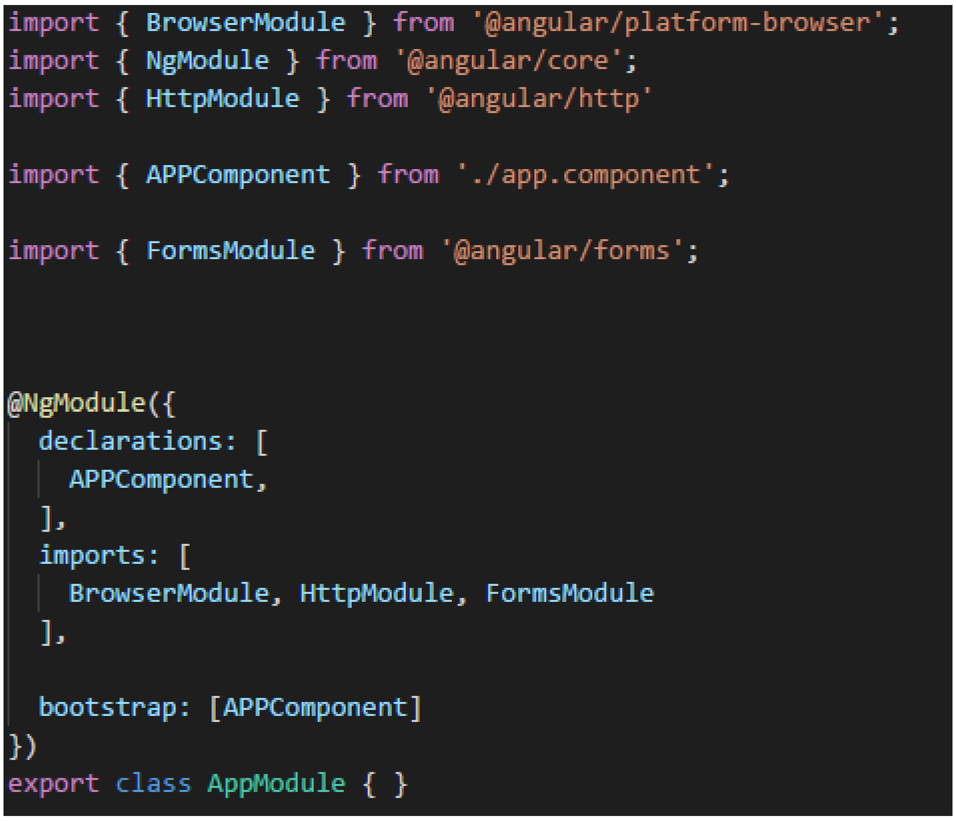
146. Which one the steps which we are using to build a form with template driven?

Ans. We have mainly below steps which we are using to build the form:

1. Create the model class as interface according to form fields.
2. Now create the component and its template according to form which we will use for the form
3. Add the name for control name and ngModel with each form control for two-way data binding.
4. Add CSS which one you want to use for show the UI of form.
5. Add the validation with required fields and according to form
6. Add the ngSubmit with the form for submit the form
7. Add the button which only enabled until the form is valid means all validation is completed.

147. When we use the form using Template-driven approach, which module we have to add and where ?

We will import the FormsModule from Angular/forms and add within imports array in the app.module.ts as:



```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { HttpModule } from '@angular/http'

import { APPComponent } from './app.component';

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



@NgModule({
  declarations: [
    APPComponent,
  ],
  imports: [
    BrowserModule, HttpModule, FormsModule
  ],

  bootstrap: [APPComponent]
})
export class AppModule { }
```

148. How many ways by which we can add form validation?

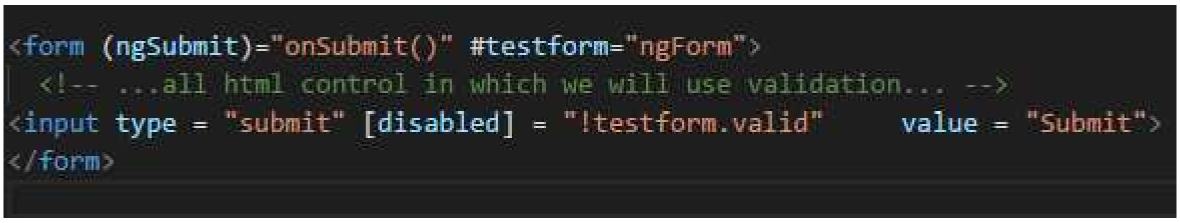
Ans. We can add form validation by two ways as Template driven and Reactive.

149. How we will use Template-Driven Form Validation?

Ans. When we use the template-driven form validation we will add the html validation with each form control as required, max length, etc.

Also we will add the directive as *ngIf, ngModel, etc. which match these attributes with the validator function.

In this validation form is valid only if all validation result true. We will use button which enabled only if form is valid as you can see below.



```
<form (ngSubmit)="onSubmit()" #testform="ngForm">
  <!-- ...all html control in which we will use validation... -->
<input type = "submit" [disabled] = "!testform.valid"     value = "Submit">
</form>
```

150. Why we are checking dirty and touched?

Ans. If you want to use in your application display errors only if user do below 2 things before these we will prevent the errors:

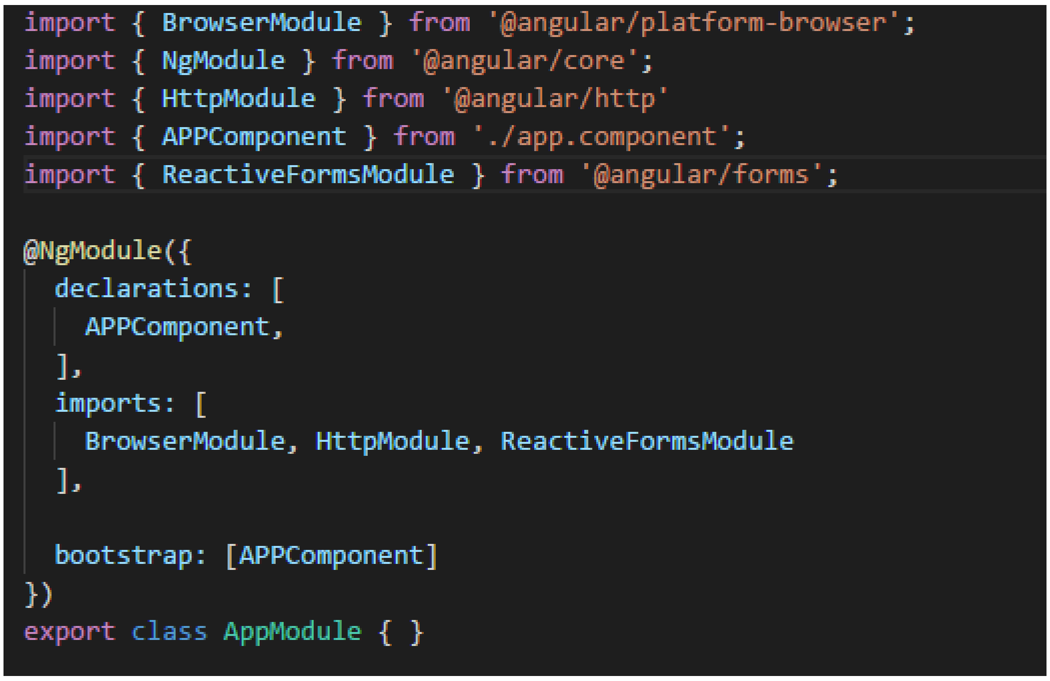
Dirty : If user change the value then its treat as dirty
Touched : If user move to control and then move to another control means on blurs the form control then its treat as Touched .

151. What is Reactive Forms?

Ans. Reactive forms provides us the model-driven approach when we will build the form using form controls. In this approach we can use multiples controls in a group and validation form controls so we can say with this approach, we can easily implement more advance forms.

152. When we will use the form using Reactive approach, which module we have to add and where?

Ans. We will implement as Template driven the difference only within template driven we are using FormsModule and here we are using ReactiveForms Module within app.module.ts as below.



```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { HttpModule } from '@angular/http'
import { APPComponent } from './app.component';
import { ReactiveFormsModule } from '@angular/forms';

@NgModule({
  declarations: [
    APPComponent,
  ],
  imports: [
    BrowserModule, HttpModule, ReactiveFormsModule
  ],

  bootstrap: [APPComponent]
})
export class AppModule { }
```

153. What is Dynamic Form and how we can implement it in Angular application?

In Angular we can build the dynamic form according to our requirement. We can create the form dynamically using metadata and object model if we want.

For more details, we can view.

https://angular.io/guide/dynamic-form

154. Which classes we are using for show with form according to status of form:

We are using following class:

1- ng-valid
2-ng-invalid
3-ng-pending
4-ng-pristine
5-ng-dirty
6-ng-untouched
7-ng-touched

Testing Based Question

155. What is unit testing?

Ans. Unit testing is the great feature by which we can easily test the application feature which we have added within component, services etc. In Angular, we have a spec.ts file for each component where we can add the unit testing.

156. What is the need of Karma and jasmine in Angular?

Ans. Karma is the task runner and Jasmine is the framework which we are using for unit testing in Angular application.

157. How we can use Karma and Jasmine in Angular for testing?

Ans. When we create new application within Angular CLI, the Angular CLI will install the Karma and Jasmine for unit testing within Angular application which we can use using ng test command and execute the test cases.

158. What is the need of ng test?

Ans. ng test is the command which we are using for run the unit test cases with the help of karma and Jasmine. When we executed the ng test command in the Angular cli it will run all the test cases according to application and show the success or fail result.

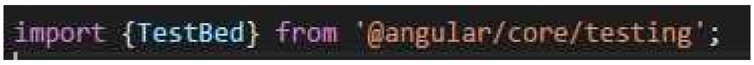
159. What is the need of test.ts in Angular?

Ans. As you know we are using Karma in Angular application for unit testing. In the karma.config file, we have setting in which we add the test.ts file path that is used by Angular application.

It's the entry point of the tests for the application.

160. What is test bed?

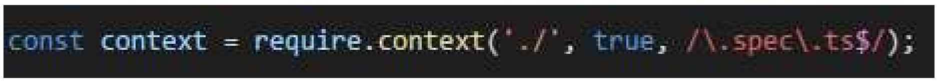
Ans. TestBed is the object and powerful unit testing tool which we can import from angular/core/testing within our Angular application as.



```
import {TestBed} from '@angular/core/testing';
```

161. We have extension .spec.ts in Angular application what is use if it?

Ans. Spec.ts denote it's a test file in which we will add the code for unit testing for component. In Angular, we have test.ts in which we have setting by which Angular application identified the test file is spec.ts as below.



```
const context = require.context('./', true, /\.spec\.ts$/);
```