TOP

100 ANGULAR INTERVIEW QUESTIONS 2023

HAPPY RAWAT

PREFACE

ABOUT THE BOOK

This book contains top 100 Angular Interview Questions. This book is based on the research of Angular interview questions asked in top IT and Tech companies like TCS, Accenture, Infosys, Wipro, HCL, IBM, Tech Mahindra, CTS, HP.

ABOUT THE AUTHOR

Happy Rawat have more than 14 years of experience in Software development. He helps candidates in clearing technical interview in tech companies.

Table of Contents

Chapter 1: Angular Framework

Q1. What is Angular?

Q2. What are Angular advantages?

Q3. What is the difference between AngularJS and Angular?

Q4. What is NPM?

Q5. What is CLI tool?

Q6. What is Typescript? What are the advantages of Typescript over Javascript?

Q7. Where to store static files in Angular project?

Q8. What is the role of Angular.json file in Angular?

Q9. What is the difference between JIT and AOT in Angular?

Chapter 2: Components & Modules

Q10. What are Components in Angular?

Q11. What is a Selector and Template?

Q12. What is Module in Angular? What is app.module.ts file?

Q13. How an Angular App gets Loaded and Started? What are index.html, app-root, selector and main.ts?

Q14. What is a Bootstrapped Module & Bootstrapped Component?

Chapter 3: Data Binding

Q15. What is Data Binding in Angular?

Q16. What is String Interpolation in Angular?

Q17. What is Property Binding in Angular?

Q18. What is Event Binding in Angular?

Q19. What is Two way Binding in Angular?

Chapter 4: Directives

Q20. What are Directives? What are the type of directives?

Q21. What is *ngIf Structural directive?

Q22. What is *ngFor Structural directive?

Q23. What is *ngSwitch Structural directive?

Q24. What is [ngStyle] Attribute directive?

Q25. What is [ngClass] Attribute directive?

Q26. What is the difference between Component, Attribute and Structural Directives?

Chapter 5: Decorator & Pipes

Q27. What is Decorator?

Q28. What are the types of Decorator?

Q29. What are Pipes? What are the types of Pipes & Parameterized Pipes?

Q30. What is Chaining Pipes?

Chapter 6: Services & Dependency Injection

Q31. Explain Services with Example?

Q32. How to create Service in Angular?

Q33. How to use Dependency Injector with Services in Angular?

Q34. What is Hierarchical Dependency Injection?

Q35. What is Provider in Angular?

Q36. What is the role of @Injectable Decorator in a Service?

Chapter 7: Decorators & Lifecycle-Hooks

Q37. What are Parent-Child Components?

Q38. What is @Input Decorator? How to transfer data from Parent component to Child component?

Q39. What is @Output Decorator and Event Emitter?

Q40. What are Lifecycle Hooks in Angular?

Q41. What is a Constructor in Angular?

Q42. What is ngOnChanges life cycle hook in Angular?

Q43. What is ngOnInit life cycle hook in Angular?

Q44. What is the difference between constructor and ngOnInit?

Chapter 8: Routing

Q45. What is Routing? How to setup Routing?

Q46. What is router outlet?

Q47. What are router links?

Chapter 9: Observable\ HttpClient\ RxJS

Q48. What are Asynchronous operations?

Q49. What is the difference between Promise and Observable?

Q50. What is RxJS?

Q51. What is Observable? How to implement Observable ?

Q52. What is the role of HttpClient in Angular?

Q53. What are the steps for fetching the data with HttpClient & Observable?

Q54. How to do HTTP Error Handling in Angular?

Chapter 10: Typescript-Basics

Q55. What is Typescript? Or What is the difference between Typescript and Javascript?

Q56. How to install Typescript and check version?

Q57. What is the difference between let and var keyword?

Q58. What is Type annotation?

Q59. What are Built in/ Primitive and User-Defined/ Non-primitive Types in Typescript?

Q60. What is “any” type in Typescript?

Q61. What is Enum type in Typescript?

Q62. What is the difference between void and never types in Typescript?

Q63. What is Type Assertion in Typescript?

Q64. What are Arrow Functions in Typescript?

Chapter 11: Typescript - OOPS

Q65. What is Object Oriented Programming in Typescript?

Q66. What are Classes and Objects in Typescript?

Q67. What is Constructor?

Q68. What are Access Modifiers in Typescript?

Q69. What is Encapsulation in Typescript?

Q70. What is Inheritance in Typescript?

Q71. What is Polymorphism in Typescript?

Q72. What is Interface in Typescript?

Q73. What's the difference between extends and implements in TypeScript ?

Q74. Is Multiple Inheritance possible in Typescript?

Chapter 12: Angular Forms

Q75. What are Angular Forms? What are the type of Angular Forms?

Q76. What is the difference between Template Driven Forms & Reactive Forms?

Q77. How to setup Template Driven Forms?

Q78. How to apply Required field validation in template driven forms?

Q79. What is Form Group and Form Control in Angular?

Q80. How to setup Reactive Forms?

Q81. How to do validations in reactive forms?

Chapter 13: Authentication/ JWT/ Auth Gurad/ HTTP Interceptor

Q82. What is Authentication & Authorization in Angular?

Q83. What is JWT Token Authentication in Angular?

Q84. How to Mock or Fake an API for JWT Authentication?

Q85. How to implement the Authentication with JWT in Angular?

Q86. What is Auth Guard?

Q87. What is HTTP Interceptor?

Q88. How to Retry automatically if there is an error response from API?

Q89. What are the parts of JWT Token?

Q90. What is Postman?

Q91. Which part of the request has the token stored when sending to API?

Chapter 14: Components Communication

Q92. What are the various ways to communicate between the components?

Q93. What is ContentProjection? What is <ng-content>?

Q94. What is Template Reference Variable in Angular?

Q95. What is the role of ViewChild in Angular?

Q96. How to access the child component from parent component with ViewChild?

Q97. What is the difference between ViewChild and ViewChildren? What is QueryList?

Q98. What is ContentChild?

Q99. What is the difference between ContentChild & ContentChildren?

Q100. Compare ng-Content, ViewChild, ViewChildren, ContentChild & ContentrChildren?

TOP 100 ANGULAR INTERVIEW QUESTIONS

Chapter 1: Angular Framework

Q1. What is Angular?

Q2. What are Angular advantages?

Q3. What is the difference between AngularJS and Angular?

Q4. What is NPM?

Q5. What is CLI tool?

Q6. What is Typescript? What are the advantages of Typescript over Javascript?

Q7. Where to store static files in Angular project?

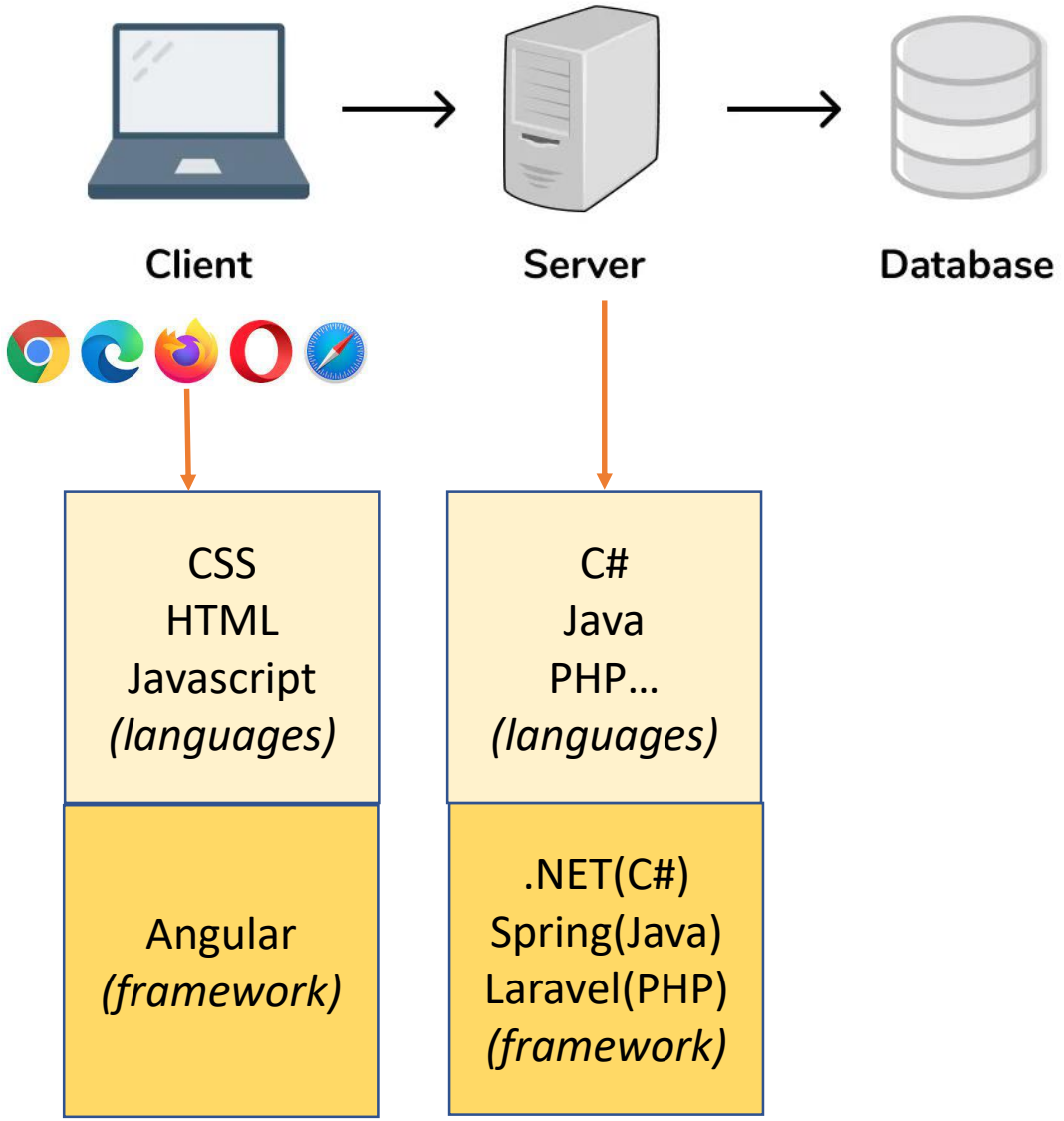
Q8. What is the role of Angular.json file in Angular?

Q9. What is the difference between JIT and AOT in Angular?

Q1 What is Angular?

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

- Like we have .NET, Spring, Laravel framework for Server side programming. Similarly we have Angular framework for Client side programming.



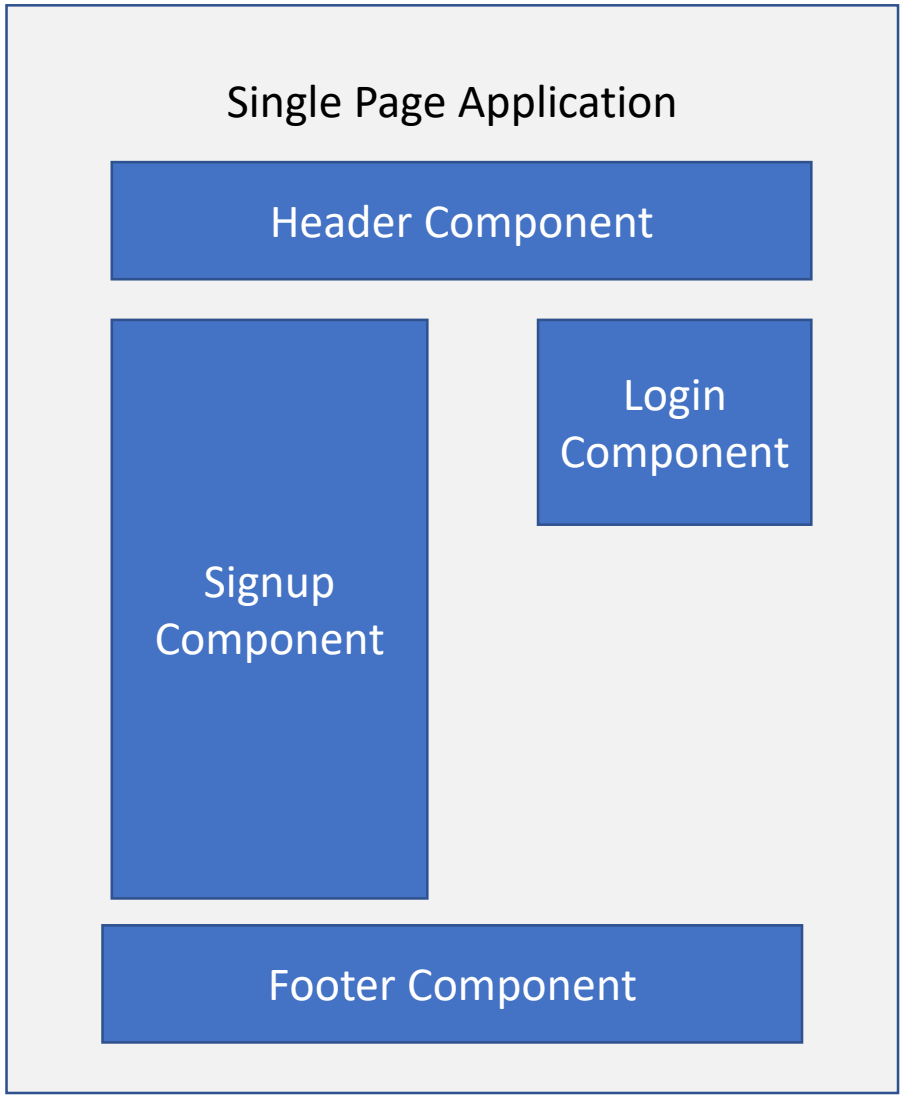
Q1

What is Angular?

- What is single page application?

In Single Page Application, we only have one page and the body content of that page will be updated as per the request.

For example, in the image we have single page with multiple components. These components will be replaced by the new components to display the data but the page will remain same.



Q2

What are Angular advantages?

1. It is Relatively simple to build Single Page Applications (by using Components)

2. Flexible and structured client applications (by using OOPS concepts)

3. Angular is cross platform and open source(Free to use)

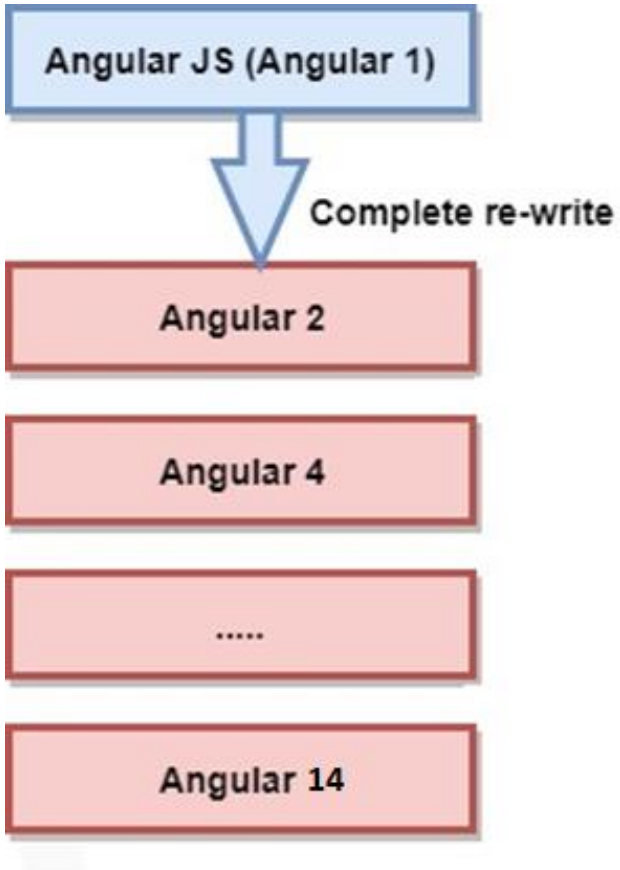
4. Writing Reusable Code is easy (by using Services)

5. Testing is easy (by using spec.ts)

Q3

What is the difference between Angular and Angular JS? OR What are the advantages of Angular over AngularJS?

- In the image, you can see Angular History from Angular JS to Angular 14 and more….



Q3

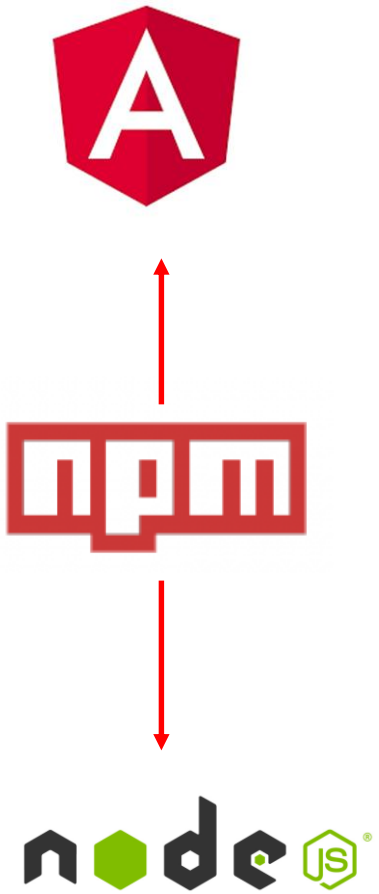
What is the difference between Angular and Angular JS? OR What are the advantages of Angular over AngularJS?

Angular JS	Angular
1. It only supports JavaScript.	It support both JavaScript and TypeScript.
2. This framework has a model-view-controller (MVC) architecture.	This framework has a component based architecture.
3. It does not have CLI tool.	It has CLI tool.
4. It does not use Dependency Injection.	It uses Dependency Injection.
5. It does not support mobile browsers.	It also support mobile browsers.
6. It is not so fast	It is very fast

Q4

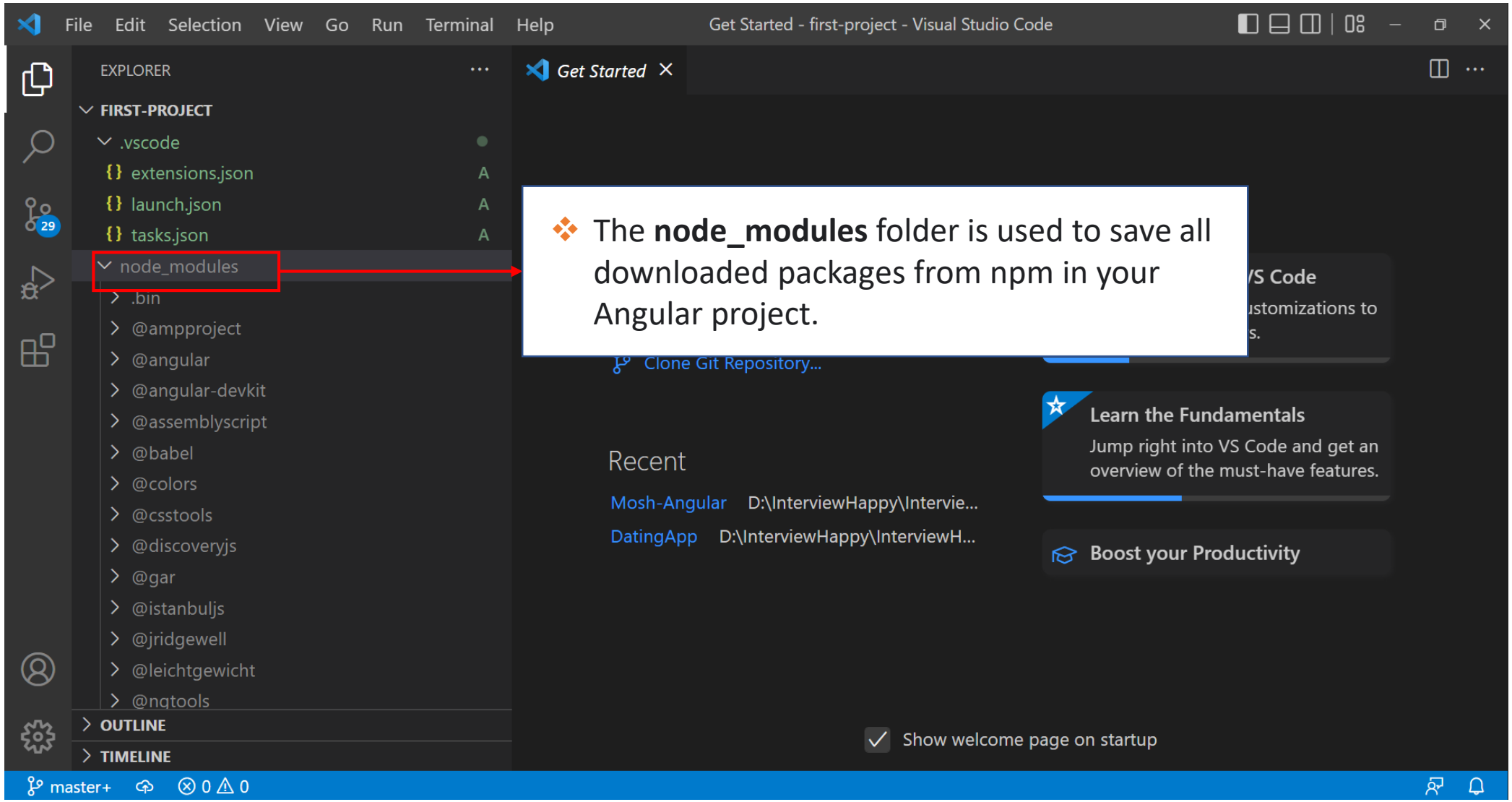
What is NPM?

- NPM(Node package manager) is an **online repository,** from where you can get thousands of **free libraries** which can be used in your **angular** or node js projects.



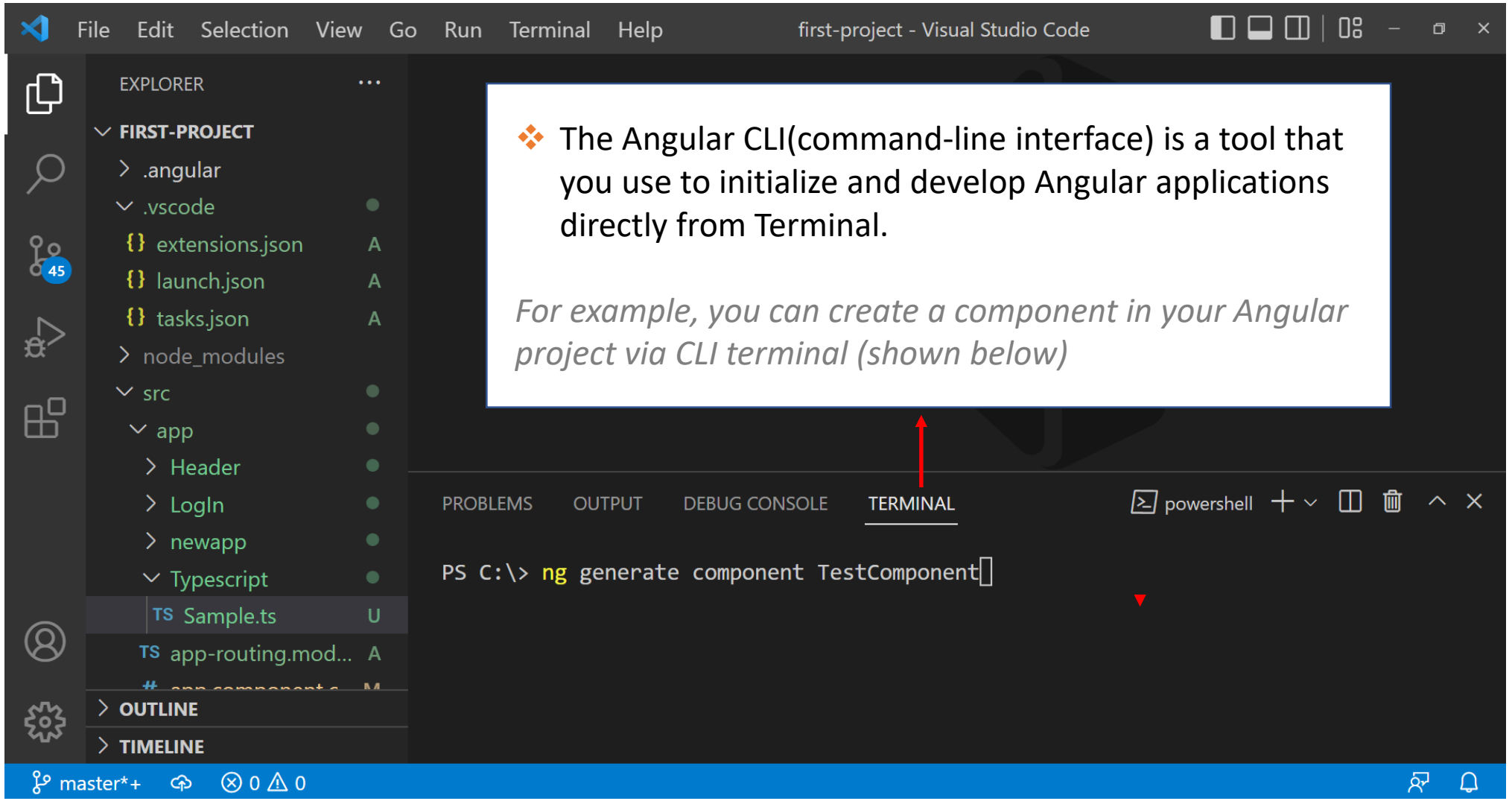
Q4

What is NPM?



Q5

What is CLI tool?



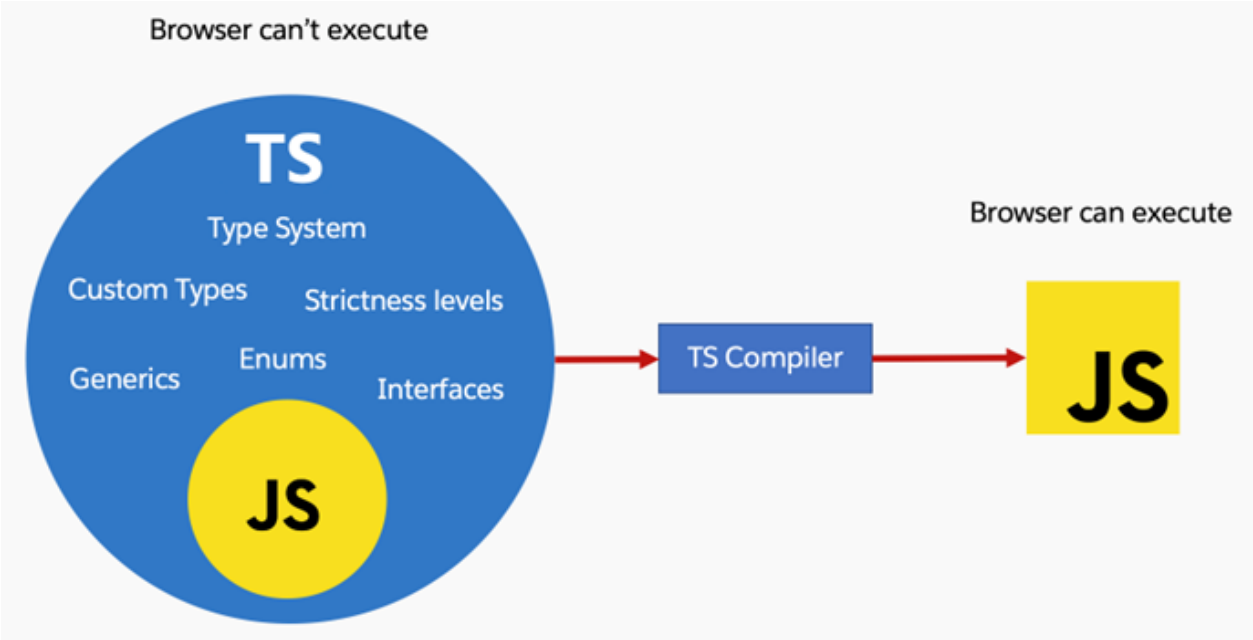
Q6

What is Typescript? What are the advantages of Typescript over Javascript?

- Typescript is an open-source programming language.
  It's advantage are:

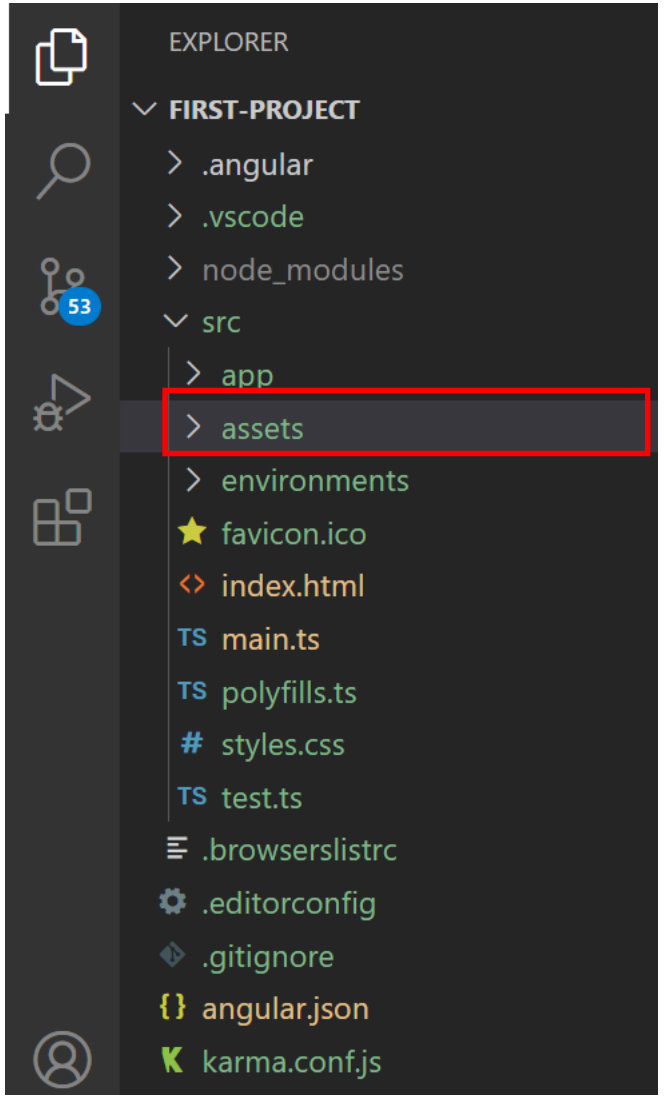
1. Typescript is a strongly typed language.
2. Typescript is a superset of JavaScript.
3. It has Object oriented features.
4. Detect error at compile time.

- In image, you can see TS(Typescript) contains JS(Javascript) and other additional features(type system, custom types, generics, enums, interfaces).
- Browser can't execute typescript, so finally TS Compiler will convert the TS to JS only, which browser can understand.



Q7 Where to store static files in Angular project?

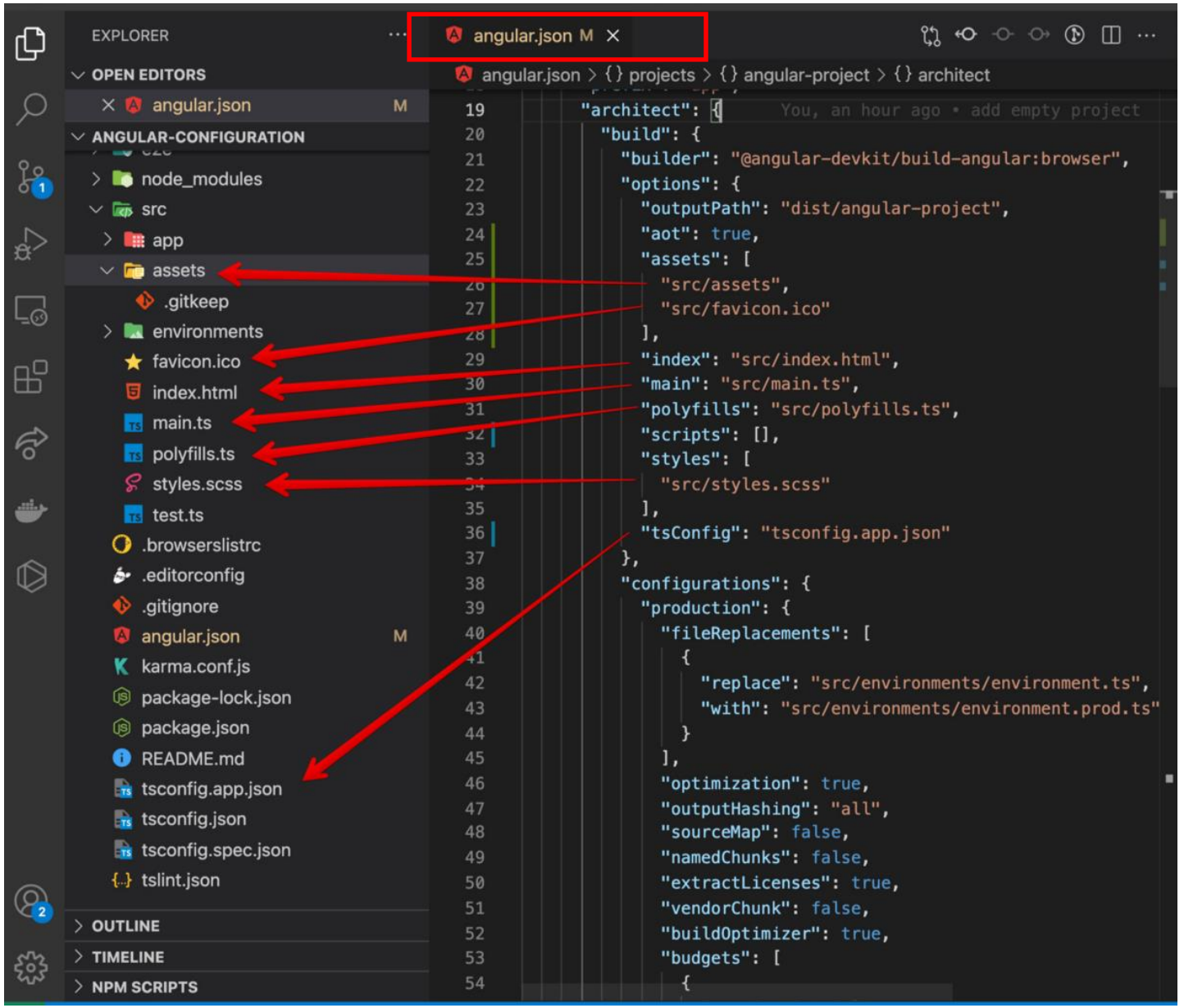
- Store static files in **assets folder** of the Angular project.



Q8 What is the role of Angular.json file in Angular?

- The angular.json file is the **primary configuration** file for an Angular project.

- *Configuration file means, in image you see the angular.json file contains all the links of all the files in Angular project.*



Q9

What is the difference between JIT and AOT in Angular? V. IMP.

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

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

- All the latest Angular versions use AOT to compile Typescript to Javascript.
- **Advantage of AOT** - Loading in AOT is much quicker than the JIT, because it already has compiled your code at build time.

TOP 100 ANGULAR INTERVIEW QUESTIONS

Chapter 2: Components & Modules

Q10. What are Components in Angular?

Q11. What is a Selector and Template?

Q12. What is Module in Angular? What is app.module.ts file?

Q13. How an Angular App gets Loaded and Started? What are index.html, app-root, selector and main.ts?

Q14. What is a Bootstrapped Module & Bootstrapped Component?

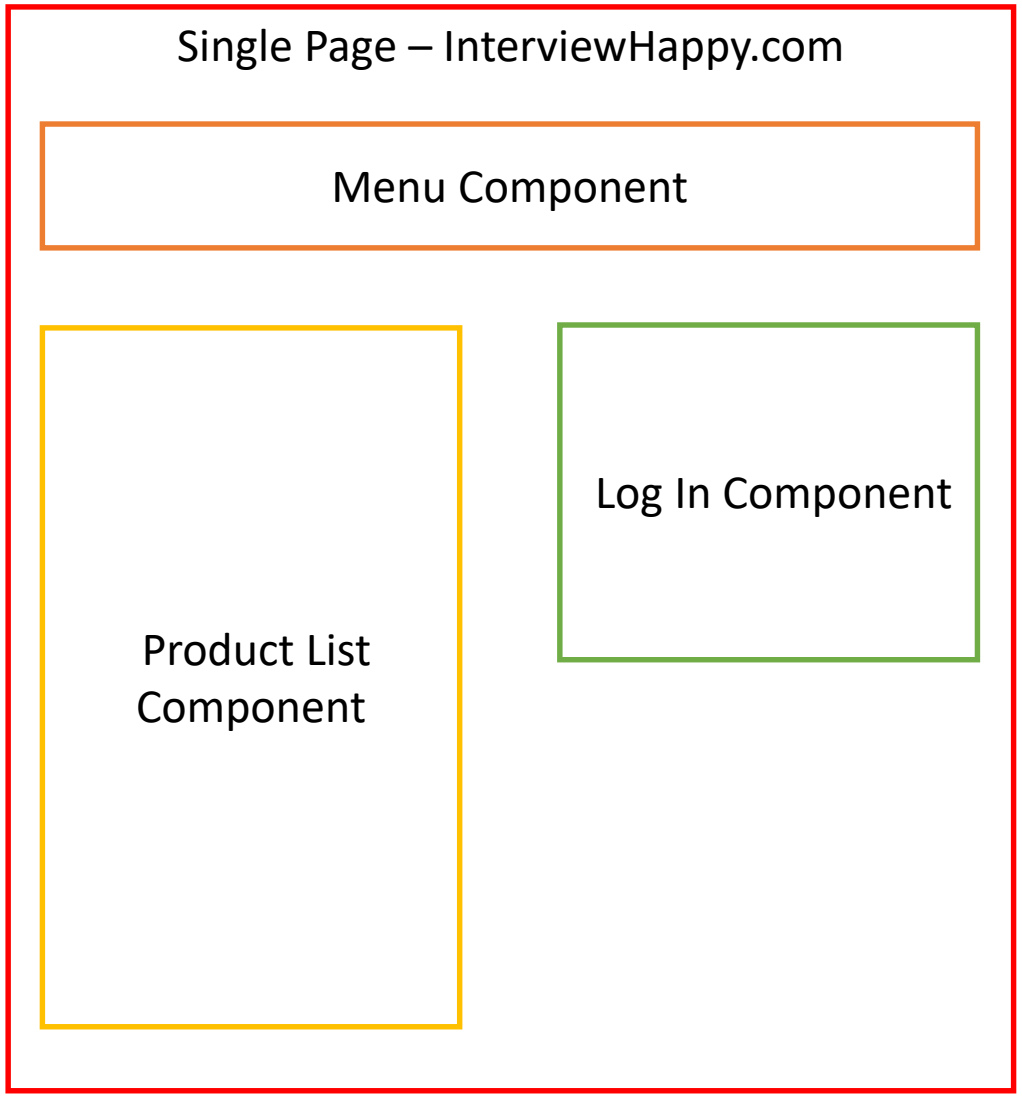
Q10

What are Components in Angular?

- Components are the most basic UI building block of an Angular app.

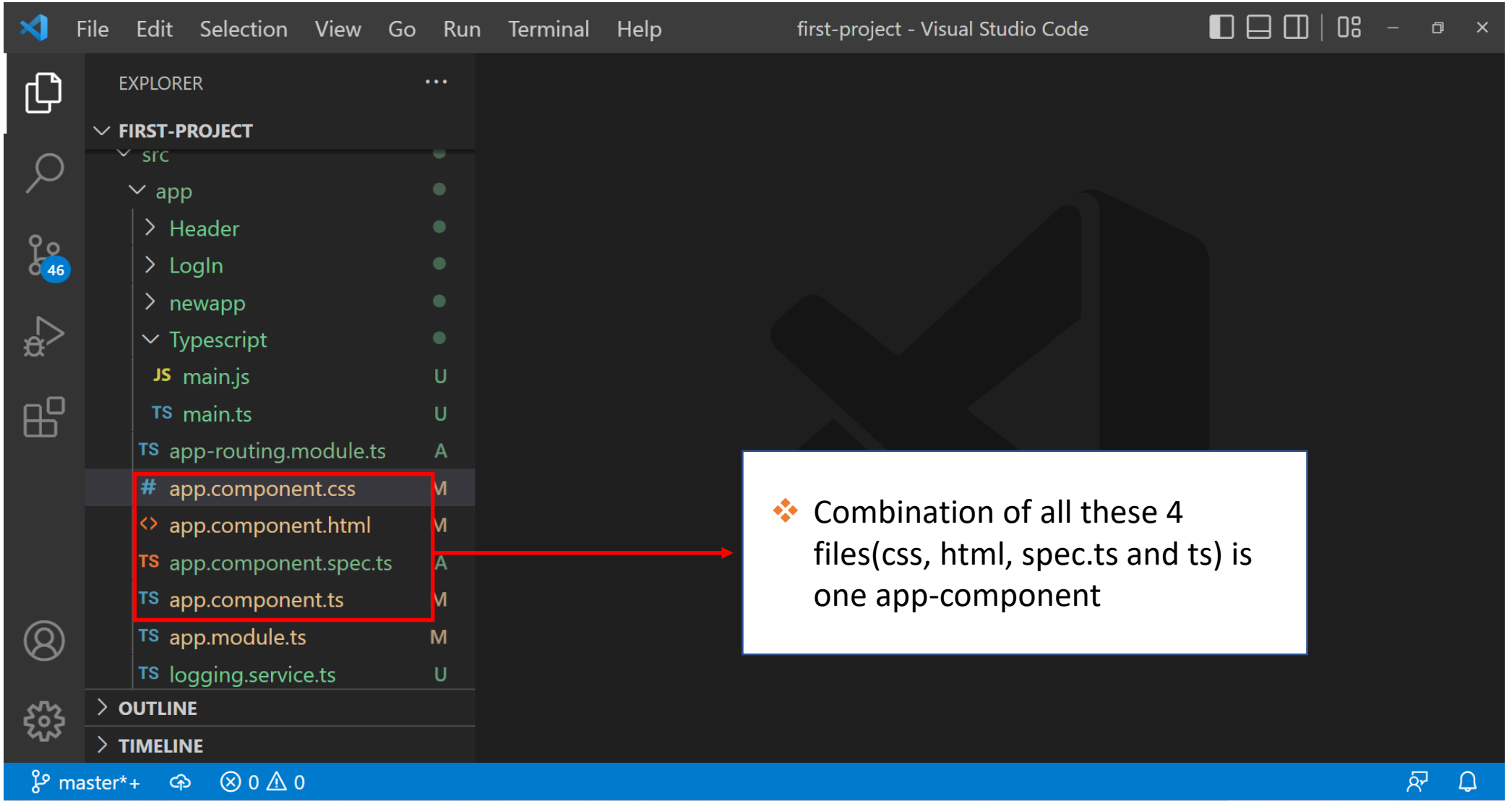
- Remember Angular is a Single Page Application.

For example, in the image we have single page with multiple components. These components will be replaced by the new components to display the data but the page will remain same. So, all these small-small components are like the building block only.



Q10

What are Components in Angular?



Q11 What is a Selector and Template?

- A selector is used to **identify each component uniquely** into the component tree.

Terminal Help app.component.ts - first-project - Visual Stu

TS app.component.ts A

src > app > TS app.component.ts > ...

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'first-project';
}
```

rminal Help index.html - first-project - Visual Studi

index.html M

rc > index.html > ...

```html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>FirstProject</title>
  <base href="/">
  <meta name="viewport" content="width=device
  <link rel="icon" type="image/x-icon" href="
</head>
<body>

  <app-root></app-root>

</body>
</html>
```

- *In first image, the app-component selector name is "app-root". And in second picture we are using this selector to position the component in index.html.*

Q11

What is a Selector and Template ?

- A Template is a **HTML view** of an Angular component.

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

@Component({
  selector: 'app-root',
  template: `<h3>Welcome to Angular Tutorials<h3>`
})

export class AppComponent {
  title = 'MyAngularApp';
}
```

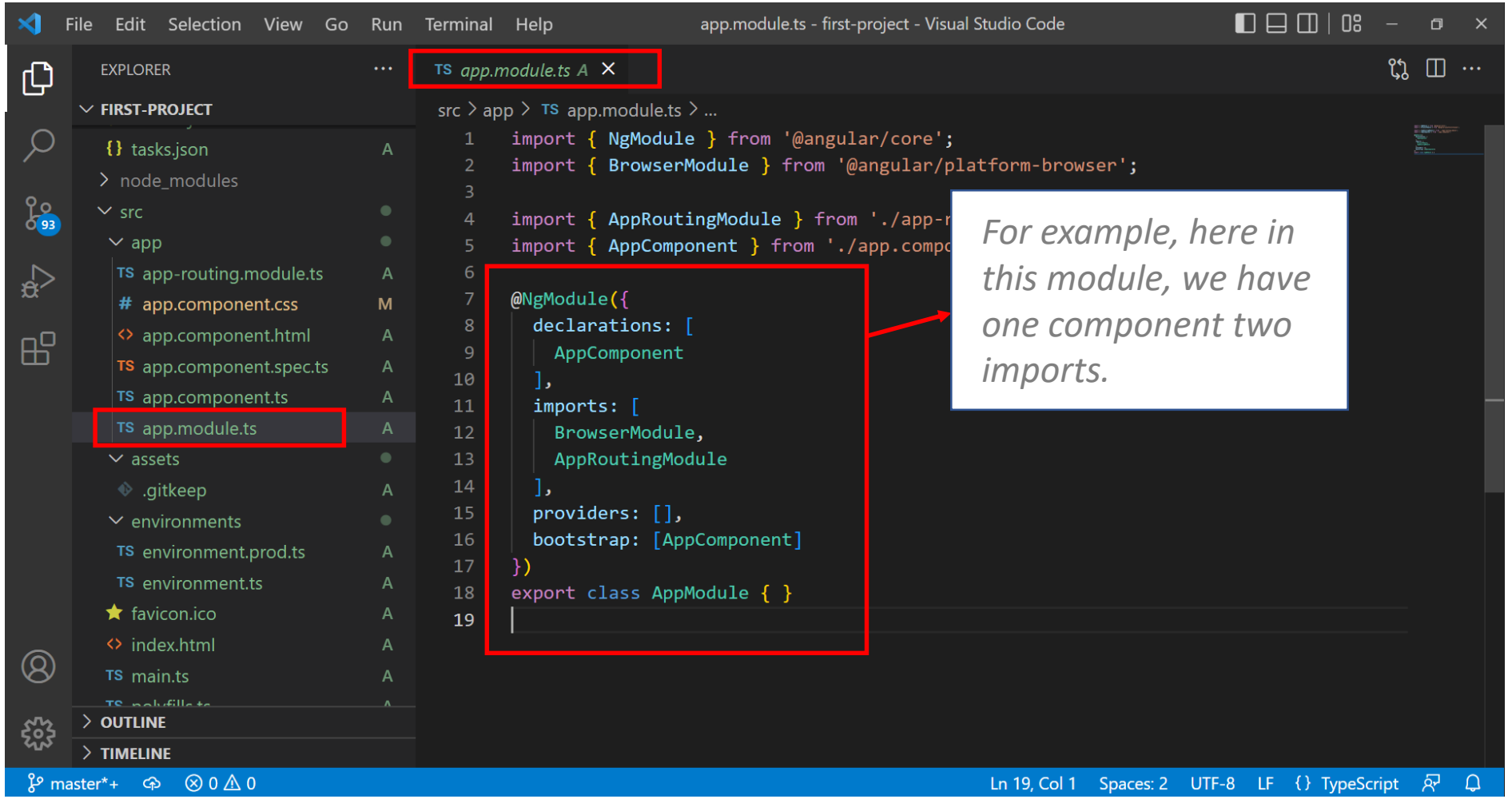
- And templateUrl is the link of the HTML template file.

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'first-project';
}
```

Q12

What is Module in Angular? What is app.module.ts file?

- Module is a place where you can group the components, directives, pipes, and services, which are related to the application.

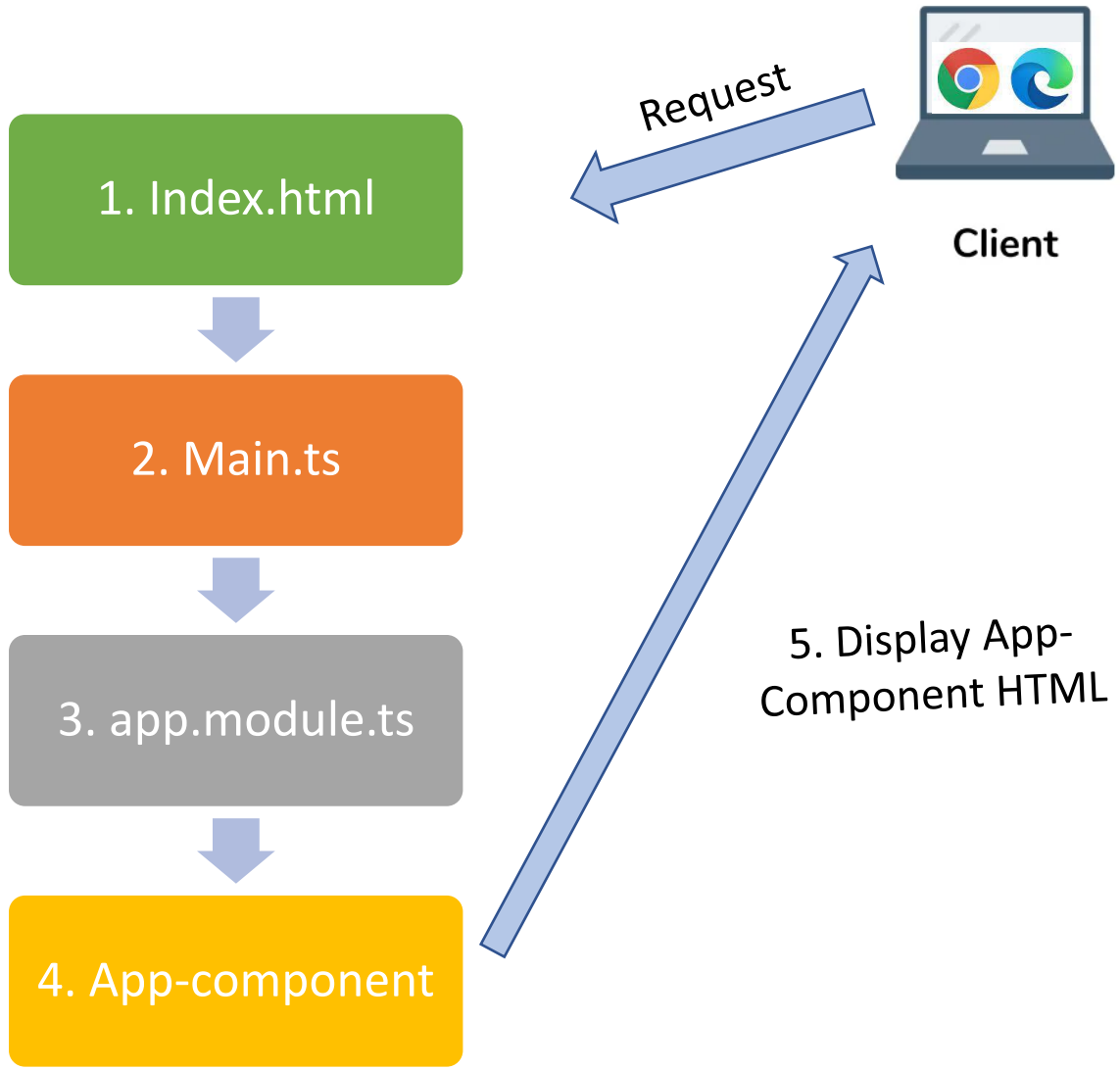


Q13

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

- 5 Steps executed when client send request from browser:

1. First, request hit index.html page.
2. Second, Index.html will invoke main.js file(which is the javascript version of main.ts file.)
3. Third, main.ts compiles the web-app and bootstraps the app.module.ts file.
4. Fourth, App-Module file will then bootstrap the app-component.
5. Finally, the App-component html will be displayed.



Q13

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

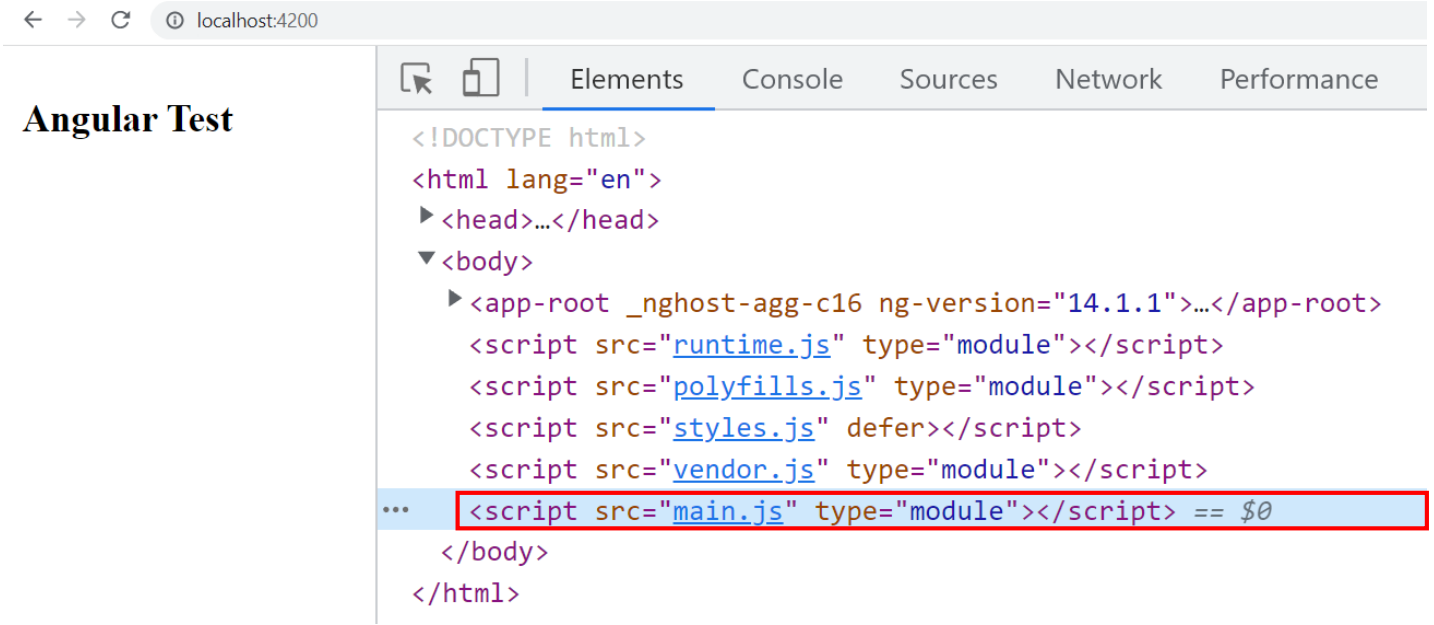
Now let's explore each step-in detail:

1. First step is request will hit index.html.

2. Then second step is, Index.html will invoke main.js file which is the javascript version of main.ts file.

Angular is used to create Single Page Applications. Index.html file is that single page.

See in image, when client hit index.html, then angular cli load these js files, with the html file. One js file among them is main.js, which will be executed then.



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

3. Third step is, main.ts compiles the web-app and bootstraps the app.module.ts file.

In the image, you can see how main.ts has bootstrapped the AppModule file.

TS main.ts A

src > TS main.ts > ...

```
1   import { enableProdMode } from '@angular/core';
2   import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
3
4   import { AppModule } from './app/app.module';
5   import { environment } from './environments/environment';
6
7   if (environment.production) {
8     enableProdMode();
9   }
10
11  platformBrowserDynamic().bootstrapModule(AppModule)
12    .catch(err => console.error(err));
13
```

Q13

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

4. Fourth step is, App-Module file will then bootstrap the app-component.

In the image, you can see how app.module.ts has bootstrapped the AppComponent file.

5. Fifth step is, App-component html will be sent to client and displayed in the browser.

TS app.module.ts A ×

src > app > TS app.module.ts > ...

```
1   import { NgModule } from '@angular/core';
2   import { BrowserModule } from '@angular/platform-browser';
3
4   import { AppRoutingModule } from './app-routing.module';
5   import { AppComponent } from './app.component';
6
7   @NgModule({
8     declarations: [
9       AppComponent
10    ],
11    imports: [
12      BrowserModule,
13      AppRoutingModule
14    ],
15    providers: [],
16    bootstrap: [AppComponent]
17  })
18  export class AppModule { }
19
```

Q14

What is a Bootstrapped Module & Bootstrapped Component?

- The module which loaded first, when the Angular application start is called Bootstrapped module.

- By default, AppModule is the Bootstrapped module in Agular project.

TS main.ts A

src > TS main.ts > ...

```
1   import { enableProdMode } from '@angular/core';
2   import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
3
4   import { AppModule } from './app/app.module';
5   import { environment } from './environments/environment';
6
7   if (environment.production) {
8     enableProdMode();
9   }
10
11  platformBrowserDynamic().bootstrapModule(AppModule)
12    .catch(err => console.error(err));
13
```

Q14

What is a Bootstrapped Module & Bootstrapped Component?

- The component which loaded first, after the module is called Bootstrapped component.
- By default, AppComponent is the Bootstrapped component in Agular project.

TS app.module.ts A

src > app > TS app.module.ts > ...

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

TOP 100 ANGULAR INTERVIEW QUESTIONS

Chapter 3: Data Binding

Q15. What is Data Binding in Angular?

Q16. What is String Interpolation in Angular?

Q17. What is Property Binding in Angular?

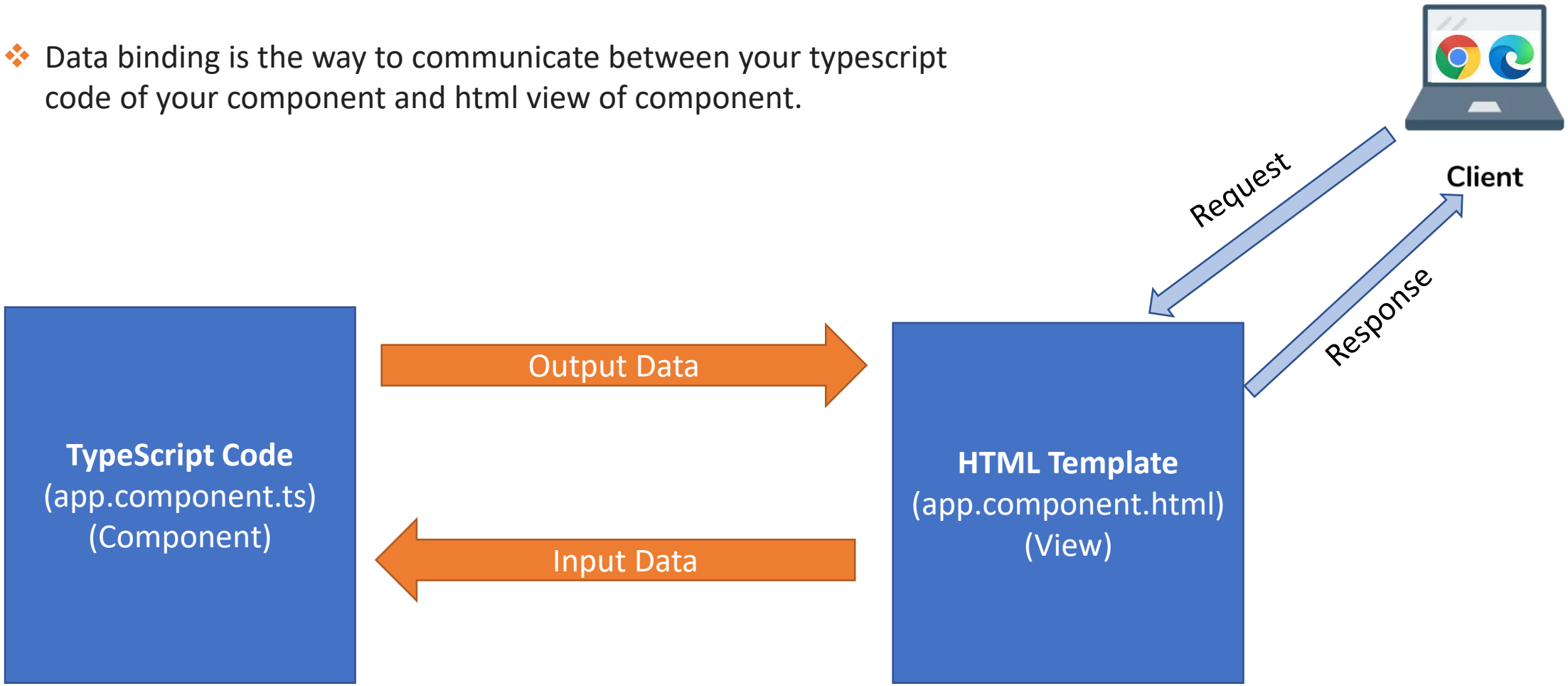
Q18. What is Event Binding in Angular?

Q19. What is Two way Binding in Angular?

Q15

What is Data Binding in Angular? V. IMP.

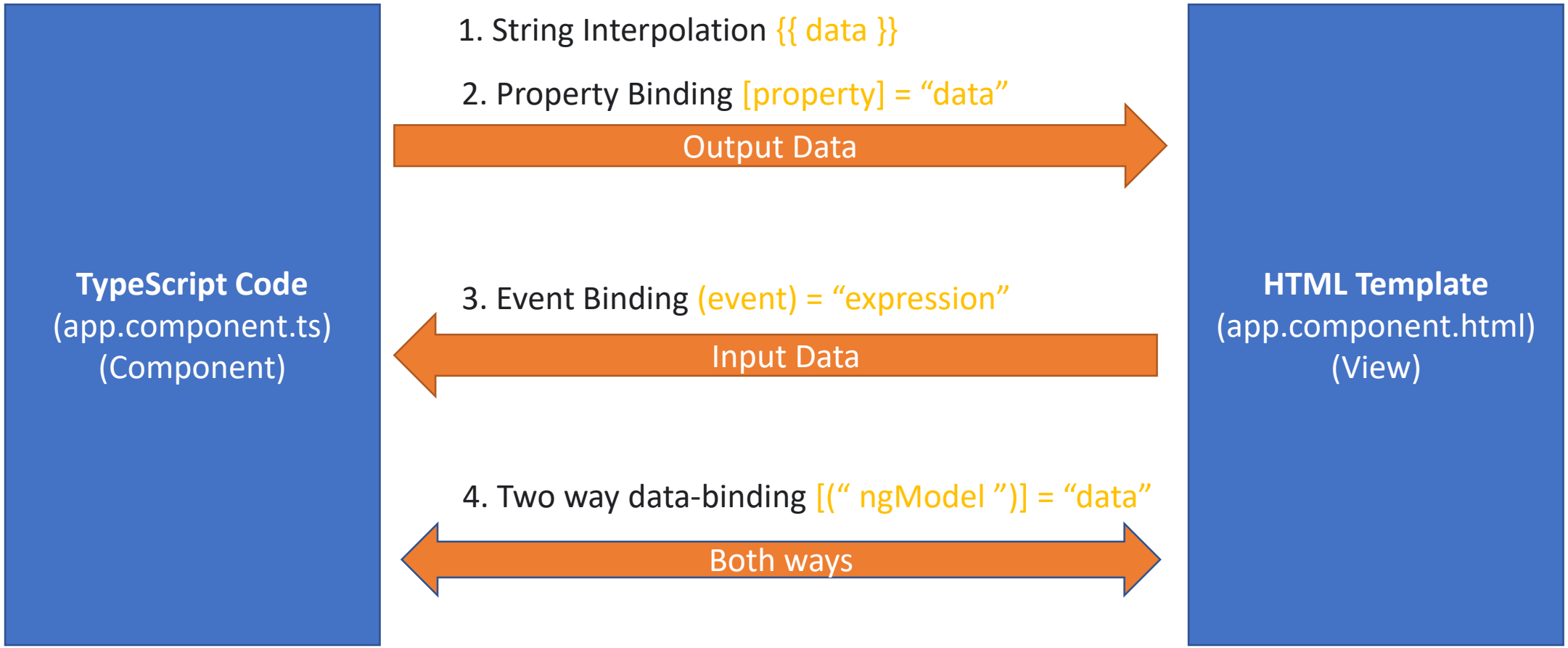
- Data binding is the way to communicate between your typescript code of your component and html view of component.



Q15

What is Data Binding in Angular? V. IMP.

- 4 Types of data binding in Angular:



Q16

What is String Interpolation in Angular?

- String Interpolation is a one-way data-binding technique that is used to transfer the data from a TypeScript code(component) to an HTML template (view).

1. String interpolation can work on string type only not numbers or any other type.

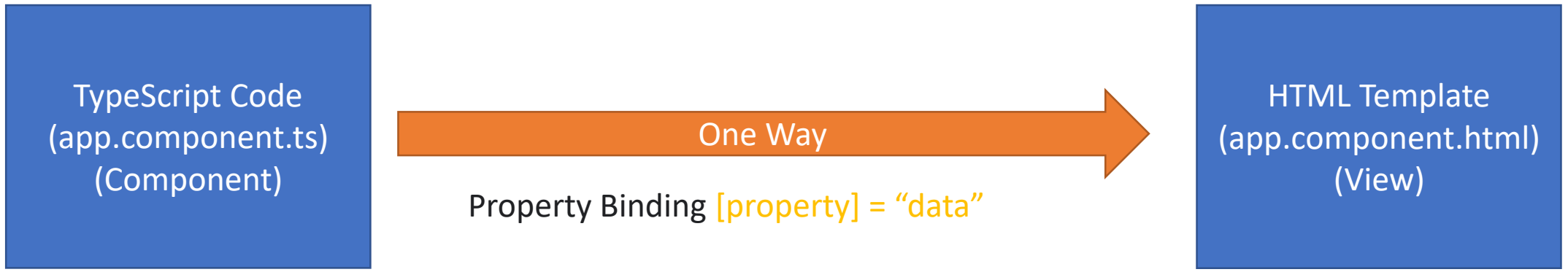
2. It is represented inside {{data}} double curly braces.

Q17

What is Property Binding in Angular?

❖ Property binding is a superset of interpolation.

It can do whatever interpolation can do. In addition, it can set an element property to a non-string data value like Boolean.



❖ When to use property binding and when to use interpolation?

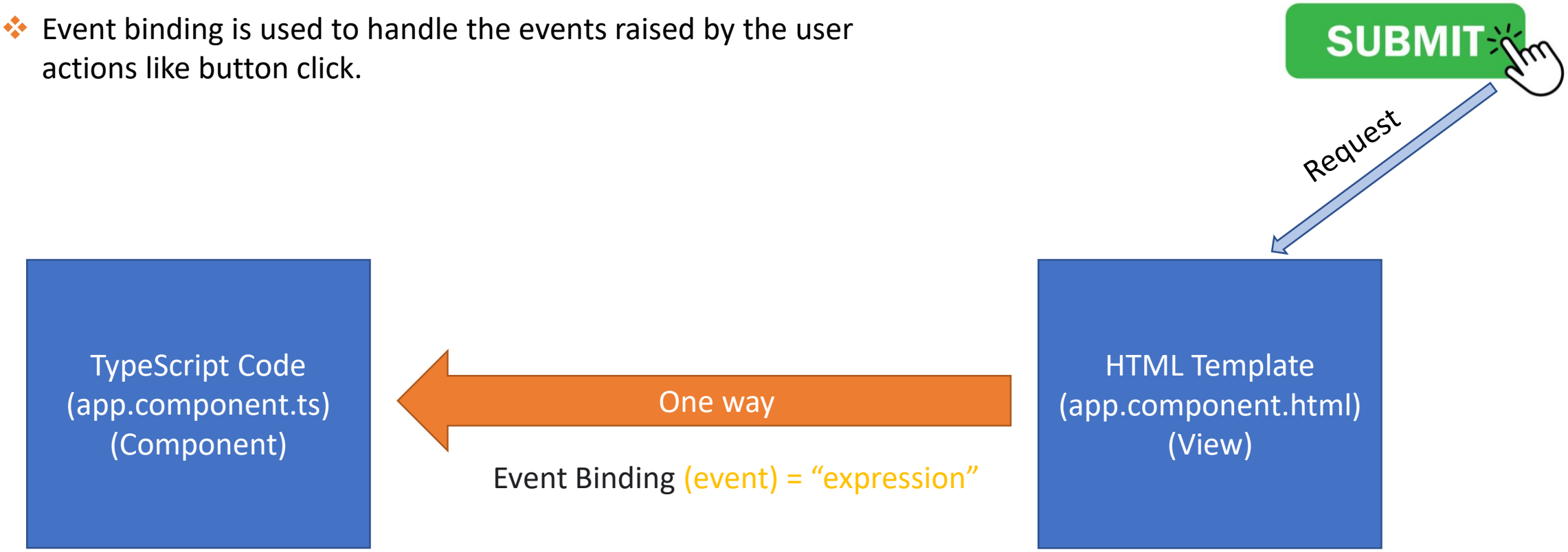
1. When you want to transfer string only, then use interpolation because of its **simplicity**(just {{data}}) to use.

2. When you want to transfer any type other than string, then use property binding.

Q18

What is Event Binding in Angular?

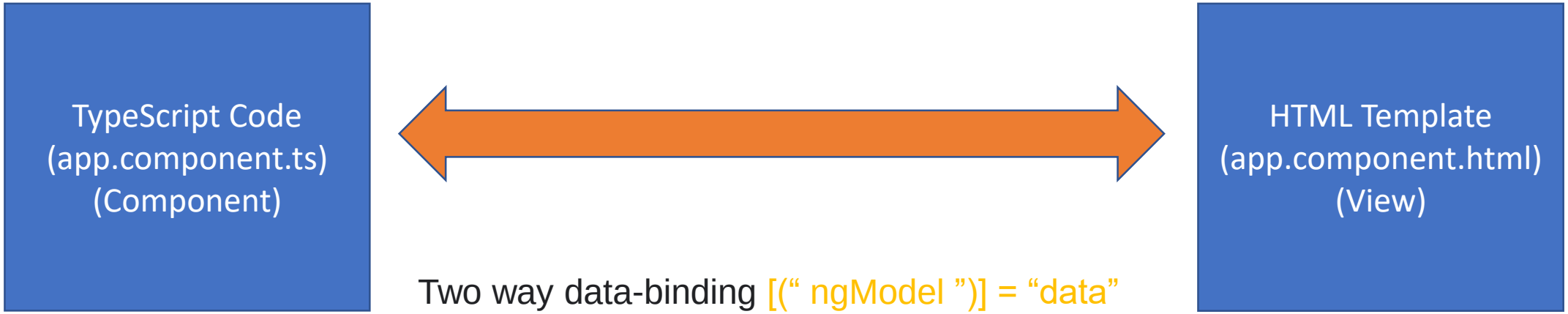
- Event binding is used to handle the events raised by the user actions like button click.



Q19

What is Two way Binding in Angular? V. IMP.

- Two-way data binding in Angular will help users to exchange data from the view to component and then from component to the view at the same time.



TOP 100 ANGULAR INTERVIEW QUESTIONS

Chapter 4: Directives

Q20. What are Directives? What are the type of directives?

Q21. What is *ngIf Structural directive?

Q22. What is *ngFor Structural directive?

Q23. What is *ngSwitch Structural directive?

Q24. What is [ngStyle] Attribute directive?

Q25. What is [ngClass] Attribute directive?

Q26. What is the difference between Component, Attribute and Structural Directives?

Q20

What are Directives? What are the type of directives?

- Directives are classes that add additional behavior to elements in your Angular applications.

- *For example, in below image, we can change the color of button element by using directives.*

Q20

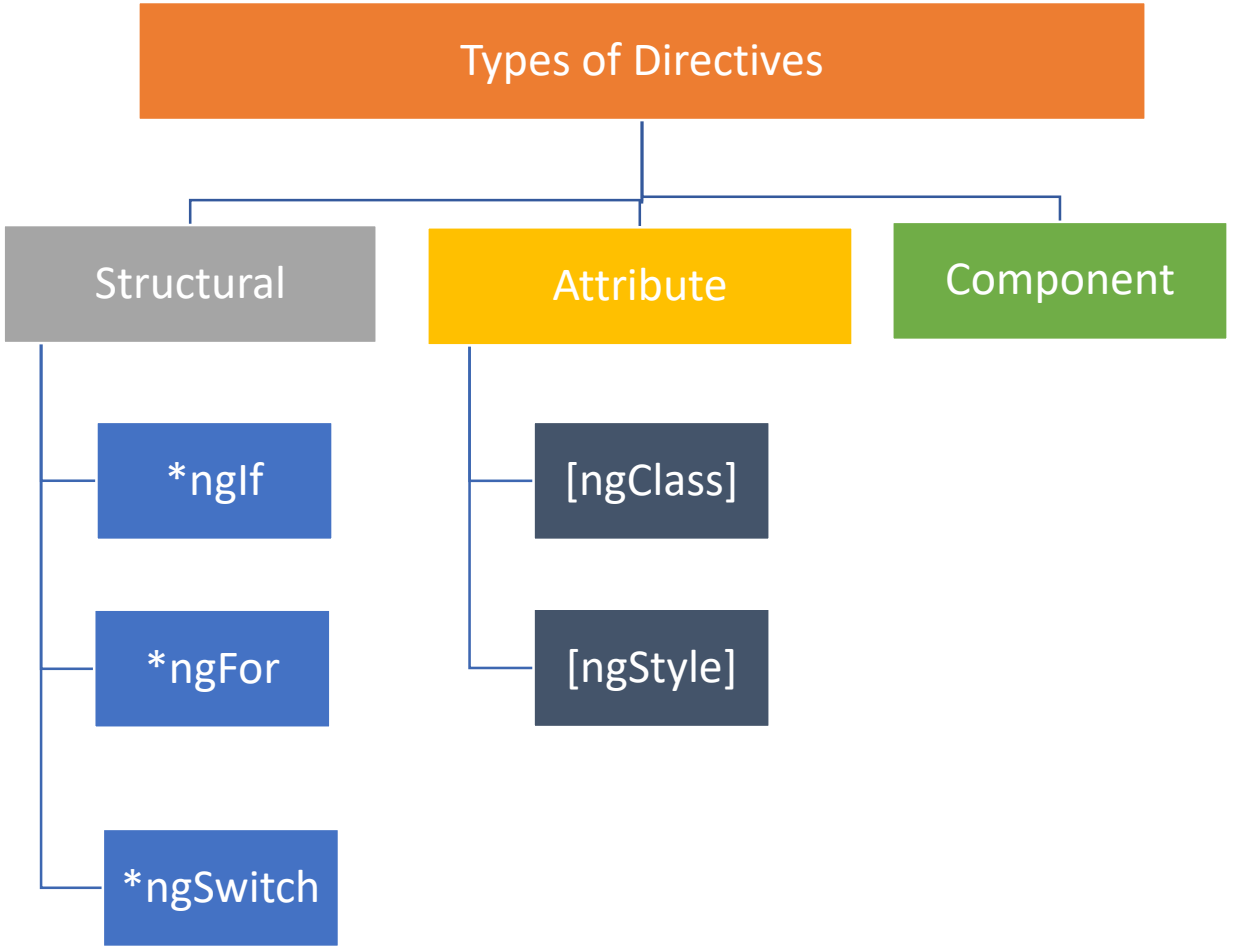
What are Directives? What are the type of directives?

❖ Types of Directives:

1. **Structural Directives** are classes that change appearance of DOM by adding or removing elements.

2. **Attribute directives** change the appearance or behavior of an element.

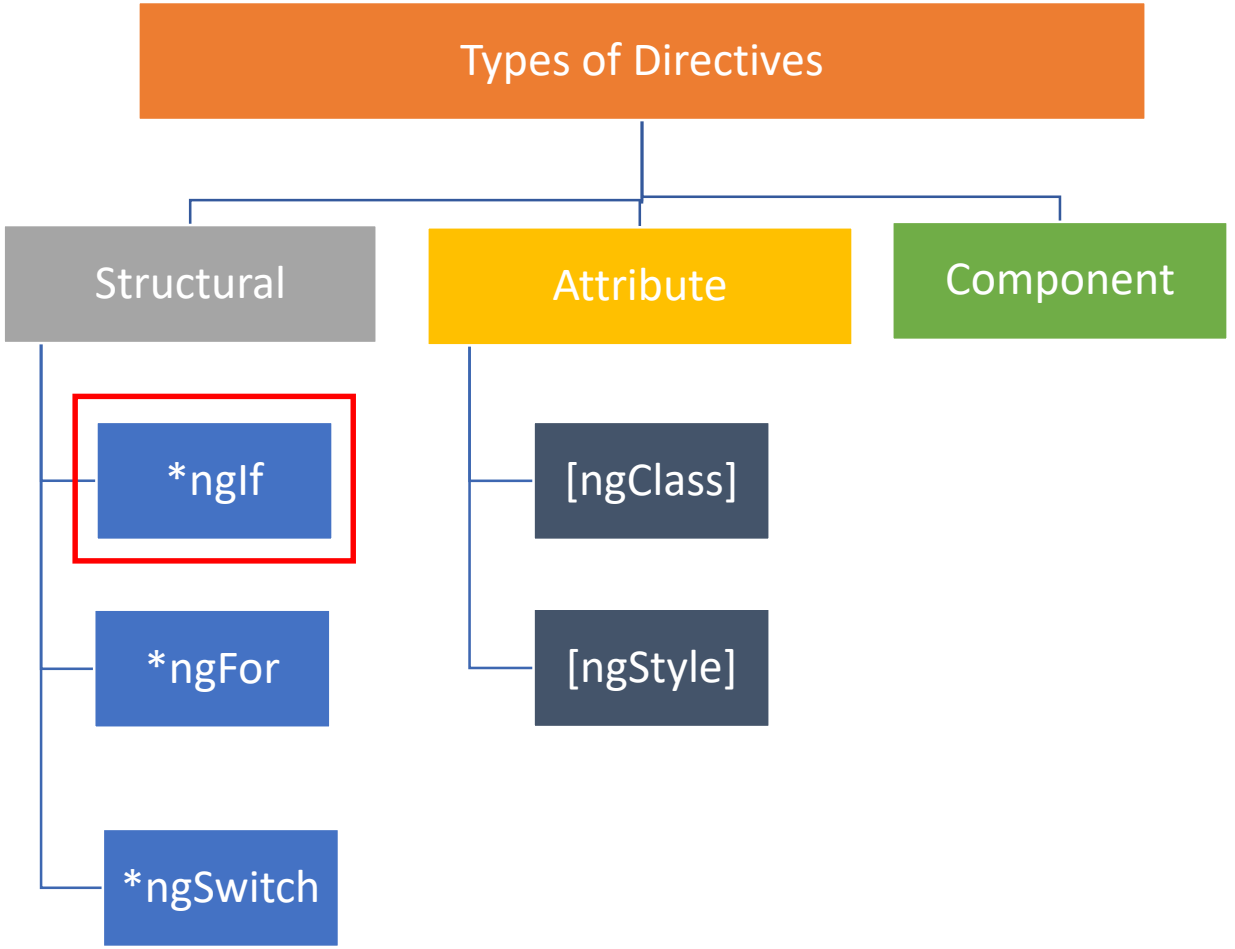
3. **Component directive** are directives with its own template.



Q21

What is *ngIf Structural directive?

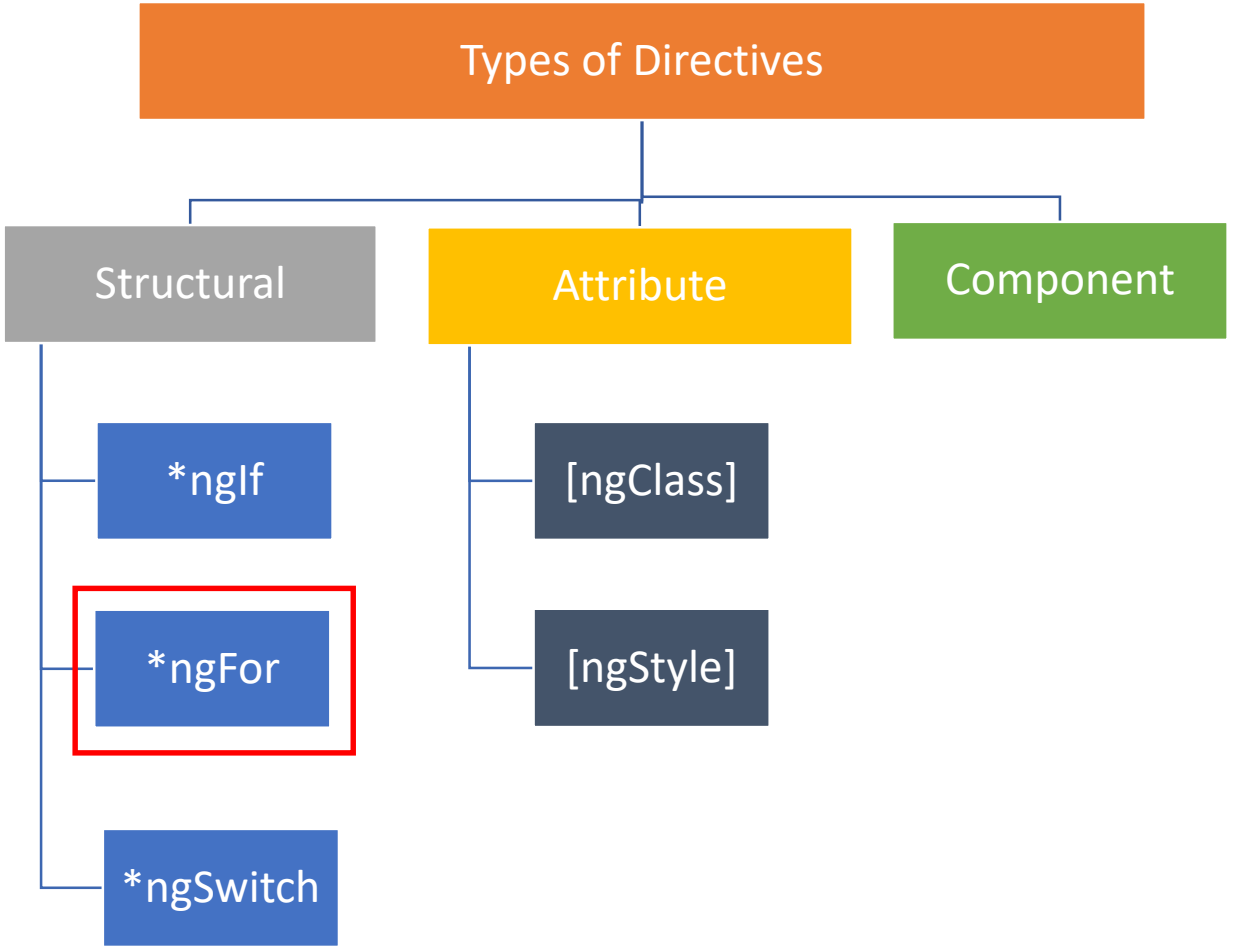
- *ngIf is a structural directive, which is used to add or remove items based on the **if condition**.



Q22

What is *ngFor Structural directive?

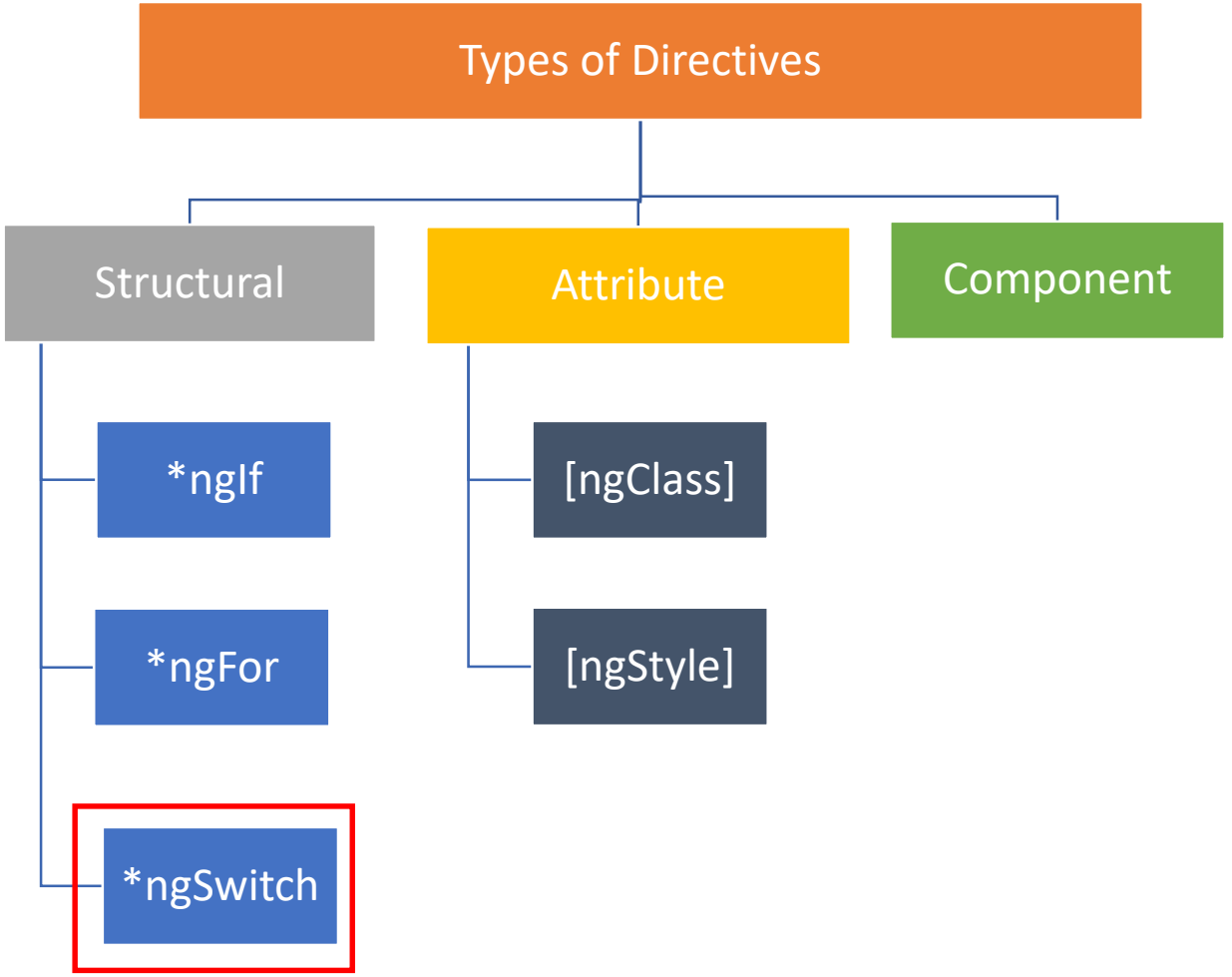
- *ngFor is a structural directive, which is used to **iterate** a list of items and then display them.



Q23

What is *ngSwitch Structural directive?

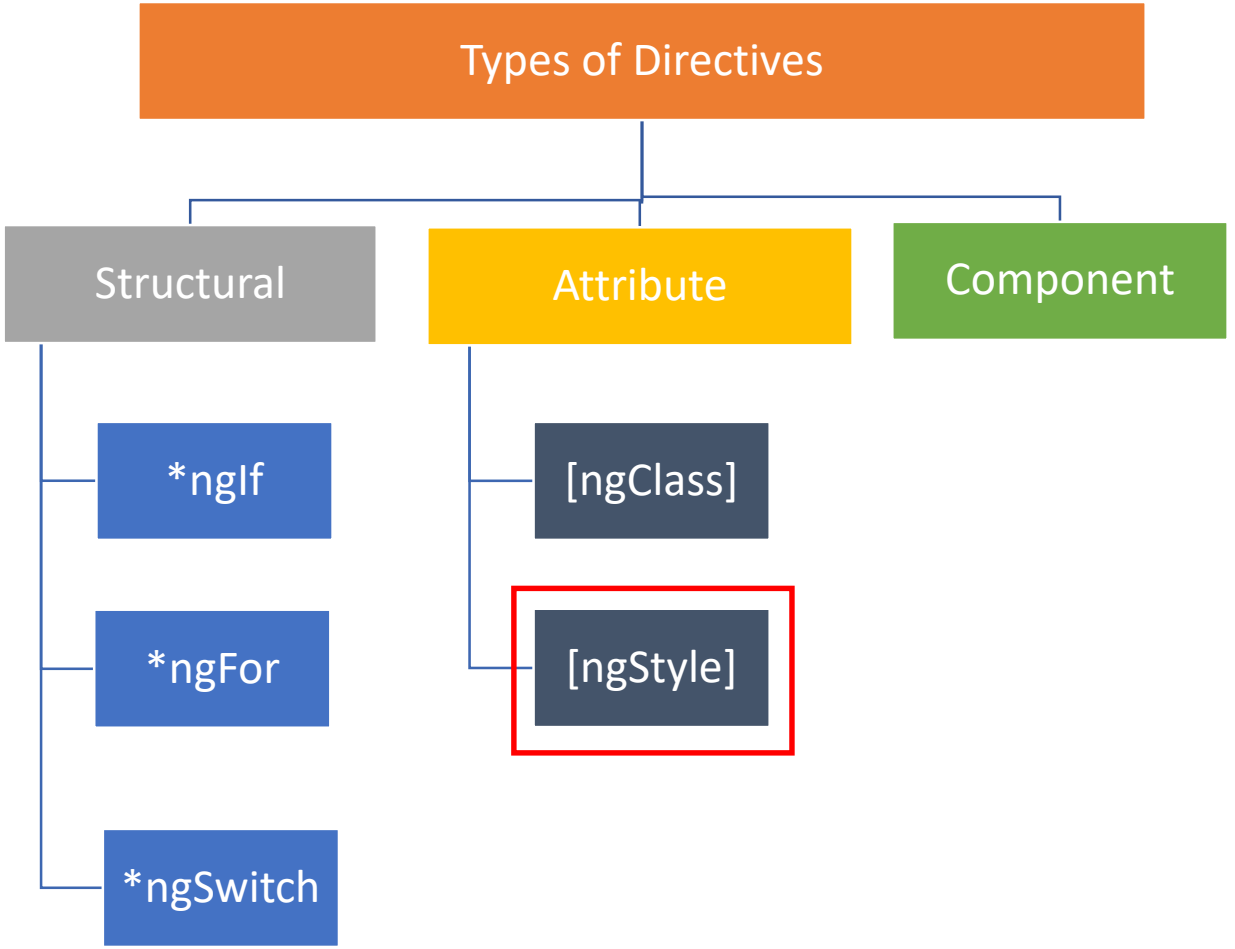
- ngSwitch is a structural directive, which is used in combination with *ngSwitchCase and *ngSwitchDefault that are both structural directives.



Q24

What is [ngStyle] Attribute directive?

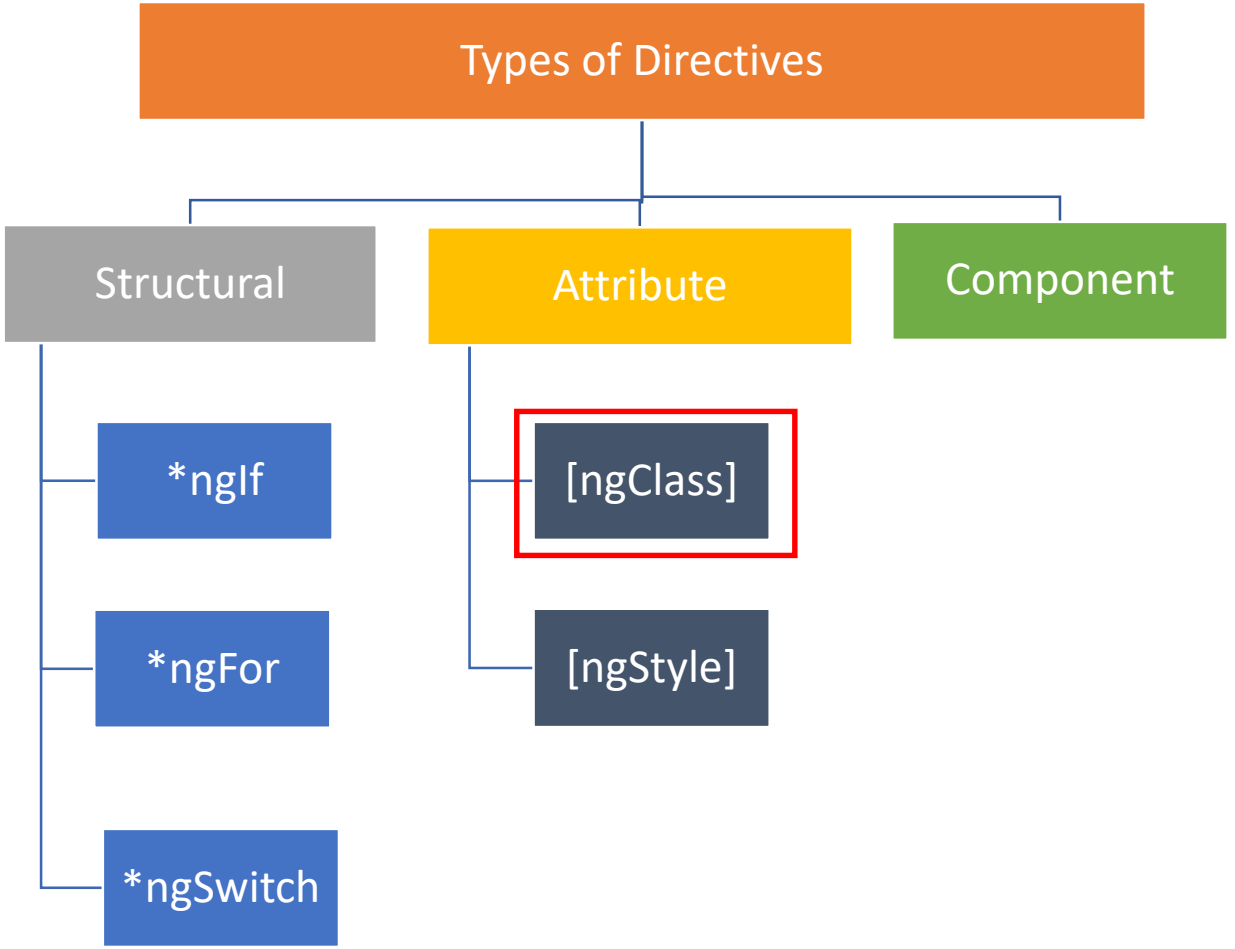
- [ngStyle] is an attribute directive, which is used to updates **styles** of the HTML element.



Q25

What is [ngClass] Attribute directive?

- [ngClass] is an attribute directive, which is used to add and remove **CSS classes** on an HTML element.



Q26

What is the difference between Component, Attribute and Structural Directives?

Component Directive	Structural Directive	Attribute Directive
1. Component directive is responsible for showing the first whole view. It is the most used one.	Structural directive is responsible for adding and deleting html elements in the view.	Attribute directive is responsible for changing the appearance of view by adding or removing styles/cssclasses of the html elements.
2. Starts with @ sign. Like: @Component	Starts with * sign. Like: *ngIf, *ngFor, *ngSwitch	Set inside square brackets. Like: [ngClass], [ngStyle]

TOP 100 ANGULAR INTERVIEW QUESTIONS

Chapter 5: Decorator & Pipes

Q27. What is Decorator?

Q28. What are the types of Decorator?

Q29. What are Pipes? What are the types of Pipes & Parameterized Pipes?

Q30. What is Chaining Pipes?

Q27

What is Decorator? V. IMP.

- Angular decorators store **metadata** about a class, method, or property.

For example, in image @Component decorator store metadata information about the app-component.

- All decorators are represented with @ symbol.

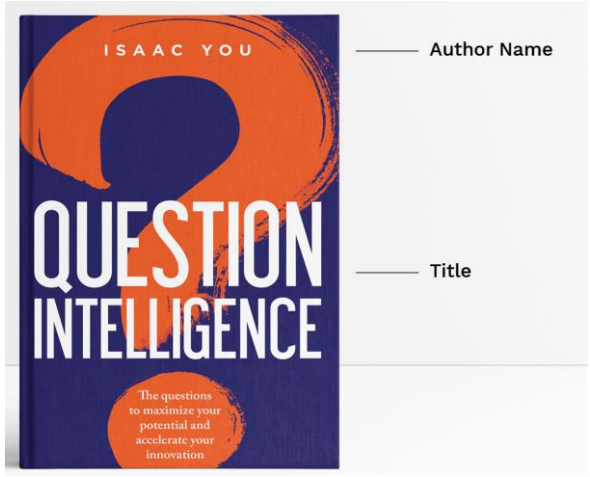
```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {

  title = 'first-interview';

}
```

Metadata

- What is Metadata?

Metadata is "data that provides information about other data“.

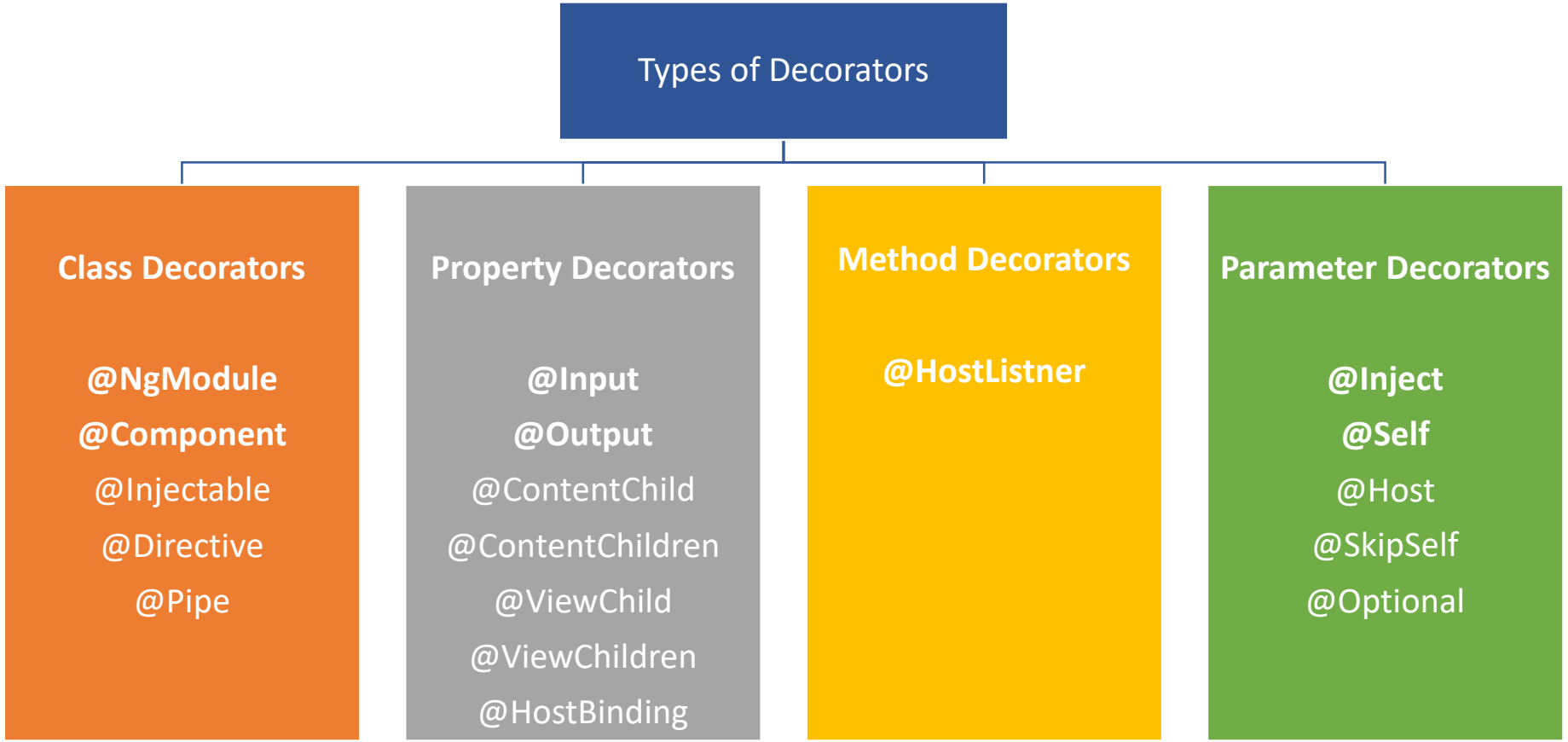
For example, in image we have a book. Now things like author name, title, price, publisher etc etc, all this information about the book is metadata only. Meaning metadata gives you some information about the book, but it is not the main content or data of the book. This is metadata.



Q28

What are the types of Decorator?

- Here are the Types of decorators – Class, Property, Method and Parameter are the categories of decorators. We will use these decorators later in different parts of the Angular applications.



Q29 What are Pipes? What are the types of Pipes & Parameterized Pipes?

- Pipes are simple functions which accept an input value and return a **transformed value**.

For example, in image the input is “Interview Happy” and the transformed value is “interview happy”.

```
<h2>{{"Interview Happy" | lowercase}}</h2>

<!-- Output - interview happy -->

<h2>{{"Interview Happy" | uppercase}}</h2>

<!-- Output - INTERVIEW HAPPY -->
```

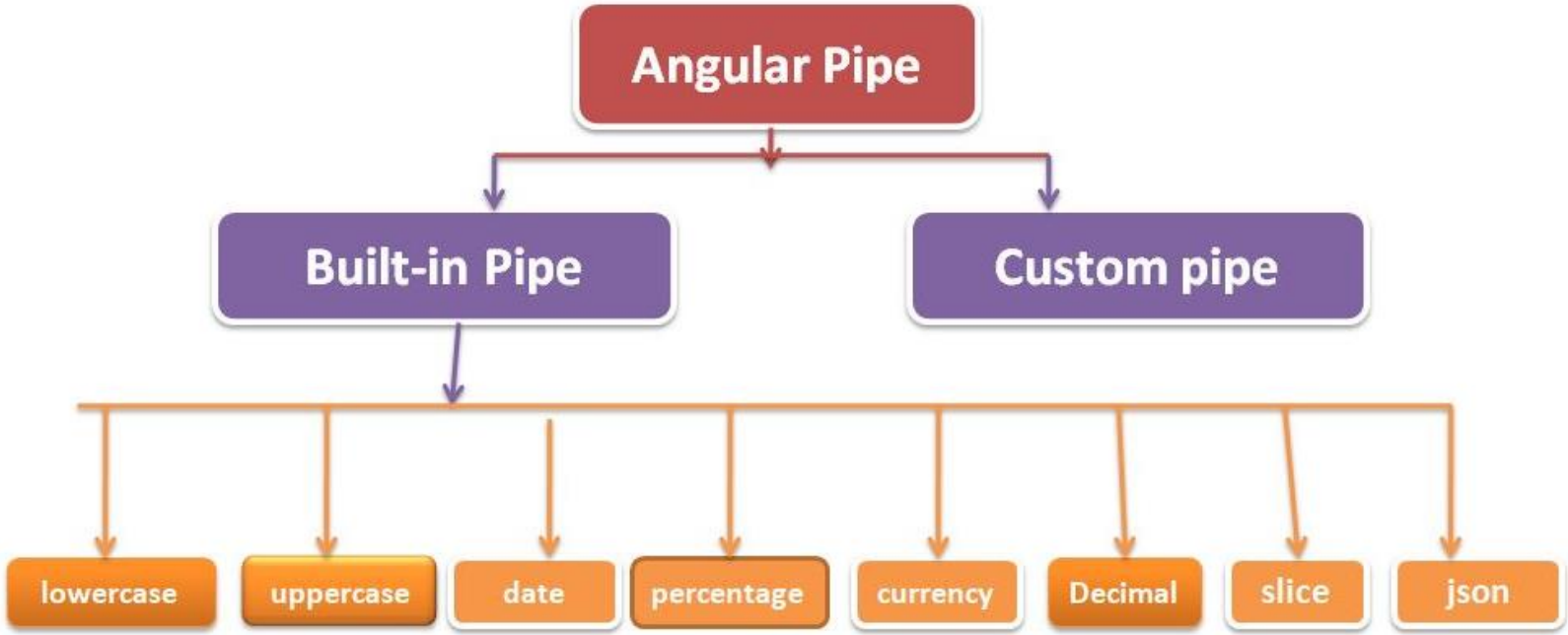
Q29

What are Pipes? What are the types of Pipes & Parameterized Pipes?

❖ Types of Pipes:

1. Built-in Pipes are provided by Angular framework. Like lowercase, uppercase, date etc.

2. Custom pipe, we can create and develop ourselves.



What are Pipes? What are the types of Pipes & Parameterized Pipes?

- When we pass any parameters to the pipes, it is called **parameterized pipes.**

```
<!-- Slice pipe will trim 10 characters from start. -->

<!-- We are passing 10 as the parameter. -->

<h2>{{"Interview Happy" | slice:10}}</h2>

<!-- Output - Happy -->
```

Q30

What is Chaining Pipes?

- The chaining pipes use multiple pipes on a data input.

For example, in below image one pipe is for slice and then other pipe will further transform the result to uppercase

```
<!-- Chaining Pipes -->

<h2>{{"Interview Happy" | slice:10 | uppercase}}</h2>

<!-- Output - HAPPY -->
```

TOP 100 ANGULAR INTERVIEW QUESTIONS

Chapter 6: Services & Dependency Injection

Q31. Explain Services with Example?

Q32. How to create Servicein Angular?

Q33. How to use Dependency Injector with Services in Angular?

Q34. What is Hierarchical Dependency Injection?

Q35. What is Provider in Angular?

Q36. What is the role of @Injectable Decorator in a Service?

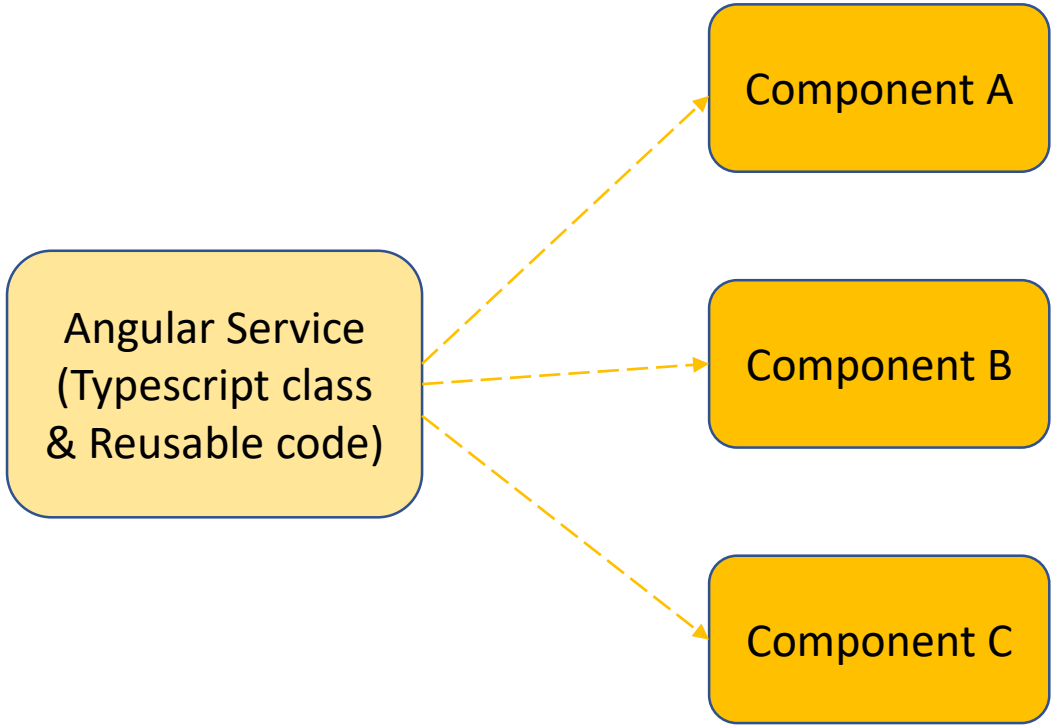
Q31

Explain Services with Example? V. IMP.

- A service is a typescript class and **reusable code** which can be used in multiple components.

- How to use services in components?

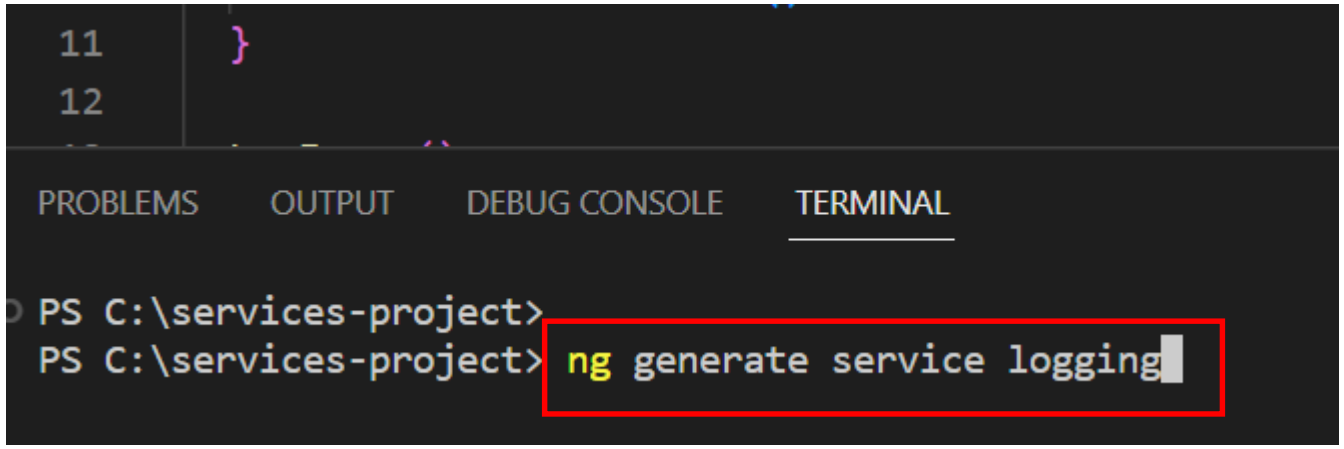
Service can be implemented in components with the help of **dependency injection**.



Q32

How to Create Service in Angular?

- *Below is the command to generate service of name “logging” in Angular*



Q33

How to use Dependency Injector with Services in Angular? V. IMP.

- 4 Steps to inject service dependency in the component:

1. First step is, create the service – (LoggingService).

```
//1. Create the Service
@Injectable()

export class LoggingService{
    logToConsole()
    {
        console.log("clicked logged");
    }
}
```

How to use Dependency Injector with Services in Angular? V. IMP.

2. Second step is, set the providers as the service name(LoggingService), inside the any component decorator as shown below.

```
@Component({
    selector: 'app-login',
    templateUrl: './login.component.html',

    providers: [LoggingService]
})

export class LoginComponent{
```

How to use Dependency Injector with Services in Angular? V. IMP.

3. Third step is, inside the **constructor** parameter create a new property(loggingService) and then assign the Service type(LoggingService) here.

```
@Component({
    selector: 'app-login',
    templateUrl: './login.component.html',

    providers: [LoggingService]
})

export class LoginComponent{

    constructor(private loggingService: LoggingService){

    }
```

How to use Dependency Injector with Services in Angular? V. IMP.

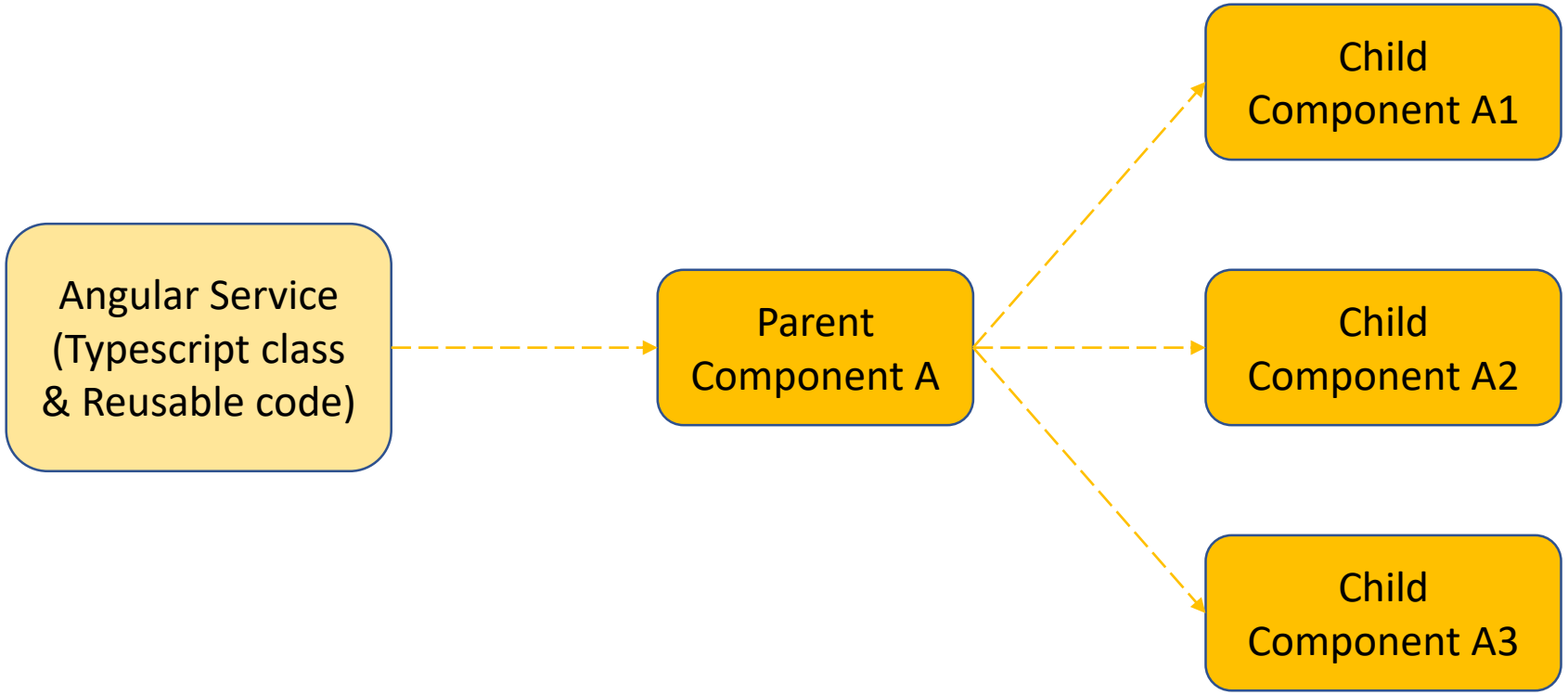
4. Finally use the property(loogingService), to call the method(logToConsole) of the LoggingService.

```
export class LoginComponent{

    constructor(private loggingService: LoggingService){

    }

    onClick(){

        this.loggingService.logToConsole();
    }
}
```

Q34

What is Hierarchical Dependency Injection?

- As per Angular, if we will inject the service in parent component, then service will be available to all it's child components, but if we will inject the service directly in child component then it will not be available for Parent and siblings components.



Q35

What is Provider in Angular?

- A provider is an object declared inside decorators which inform Angular that a particular service is available for injecting inside the components.

Provider can be set in module or in the component as shown below.

```
10  @NgModule({
11    declarations: [
12      AppComponent,
13      LoginComponent,
14      MenuComponent
15    ],
16    imports: [
17      BrowserModule,
18      AppRoutingModule
19    ],
20    providers: [LoggingService],
21    bootstrap: [AppComponent, LoginCo
22  })
23  export class AppModule { }
24
```

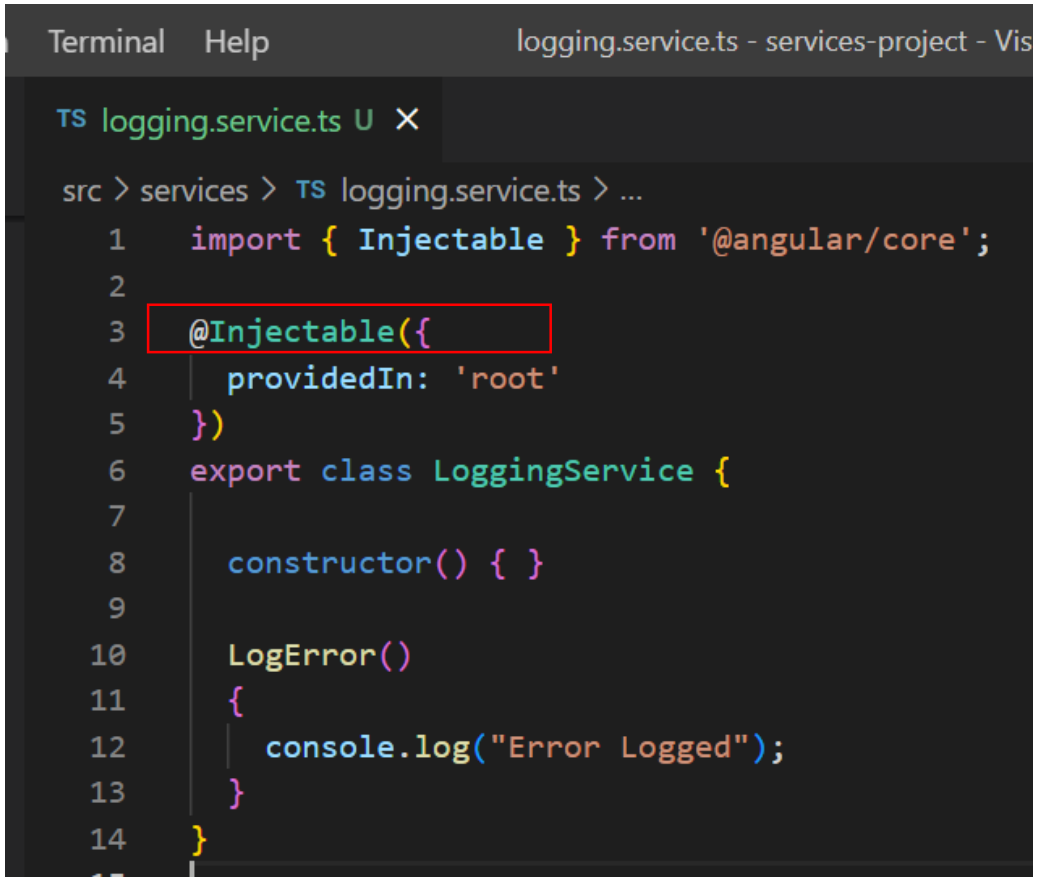
```
@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css'],
  providers: [LoggingService]
})
export class LoginComponent implements OnInit {

  constructor(private loggingService: LoggingServi

    this.loggingService.LogError();
    //const ls = new LoggingService();
    //ls.LogError();
    // console.log("Error Logged1");
    // console.log("Error Logged2");
    // console.log("Error Logged3");
  }
```

Q36

What is the role of @Injectable Decorator? How to use one service inside another service?

- With @Injectable decorator one service can be used by another service.

For example, in image we set the LoggingService as Injectable. That means, now any other service also can use LoggingService methods.



TOP 100 ANGULAR INTERVIEW QUESTIONS

Chapter 7: Decorators & Lifecycle-Hooks

Q37. What are Parent-Child Components?

Q38. What is @Input Decorator? How to transfer data from Parent component to Child component?

Q39. What is @Output Decorator and Event Emitter?

Q40. What are Lifecycle Hooks in Angular?

Q41. What is a Constructor in Angular?

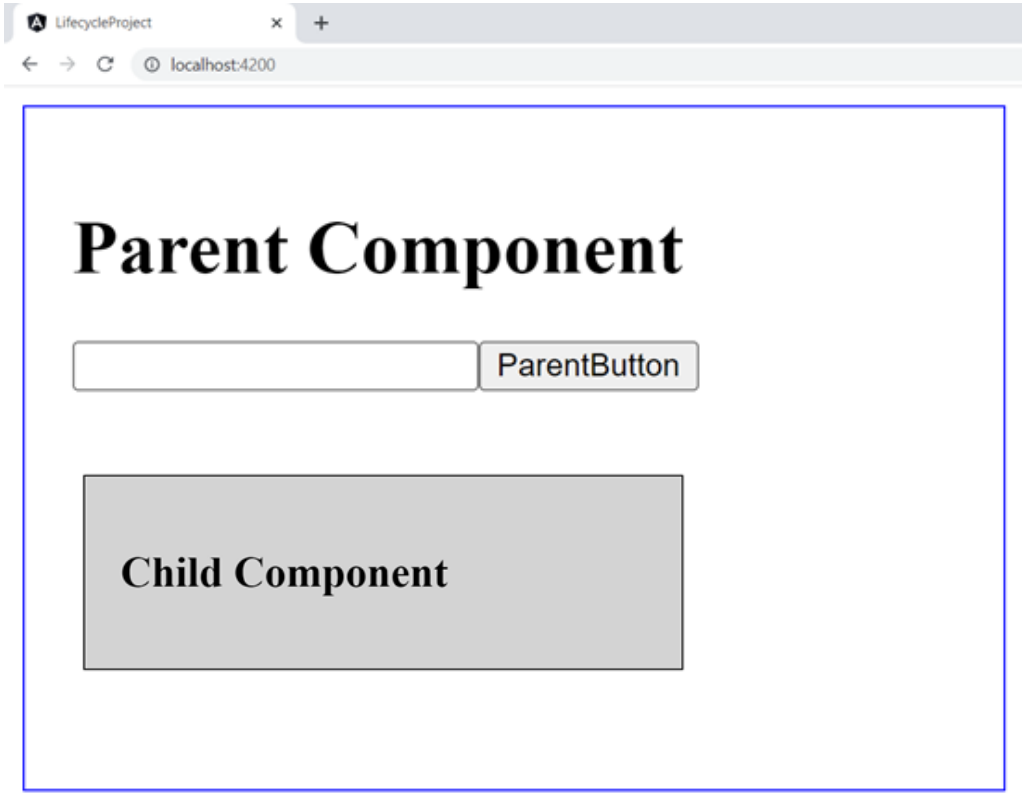
Q42. What is ngOnChanges life cycle hook in Angular?

Q43. What is ngOnInit life cycle hook in Angular?

Q44. What is the difference between constructor and ngOnInit?

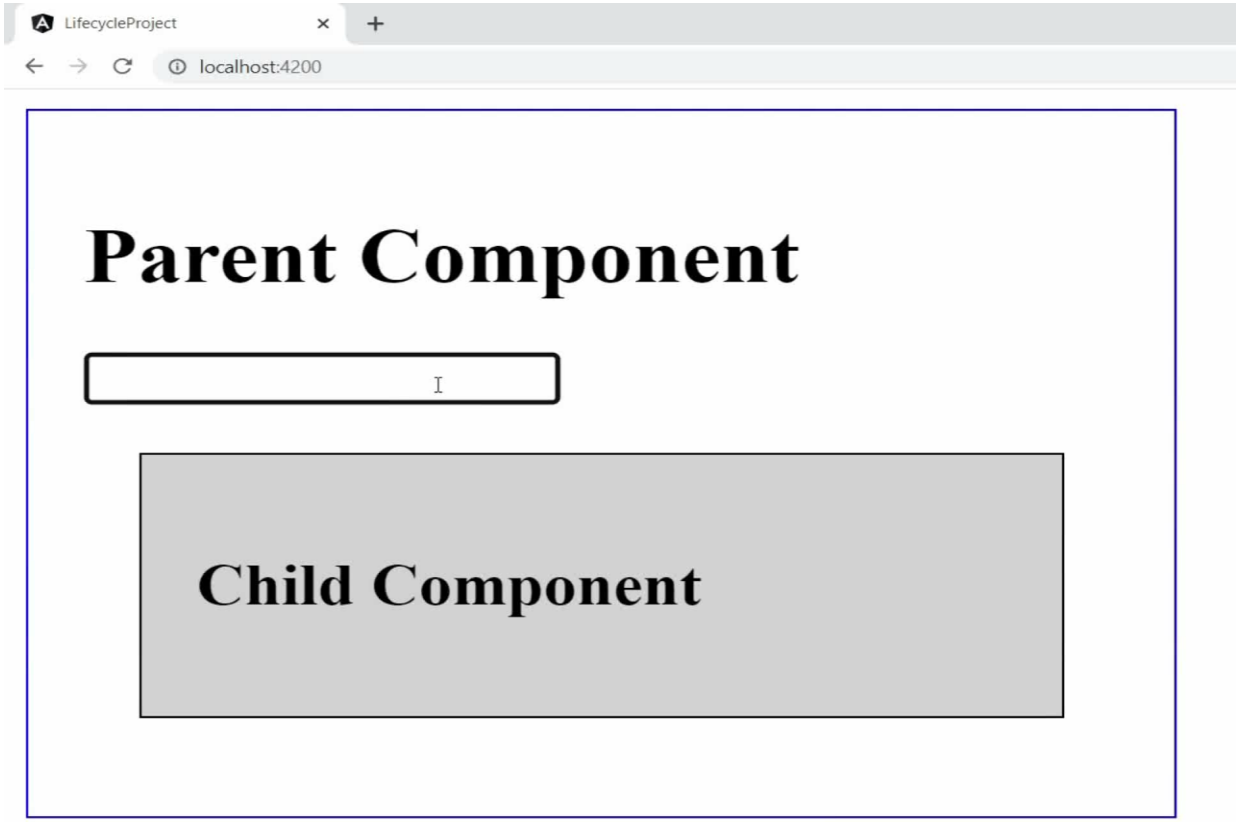
Q37 What are Parent-Child Components?

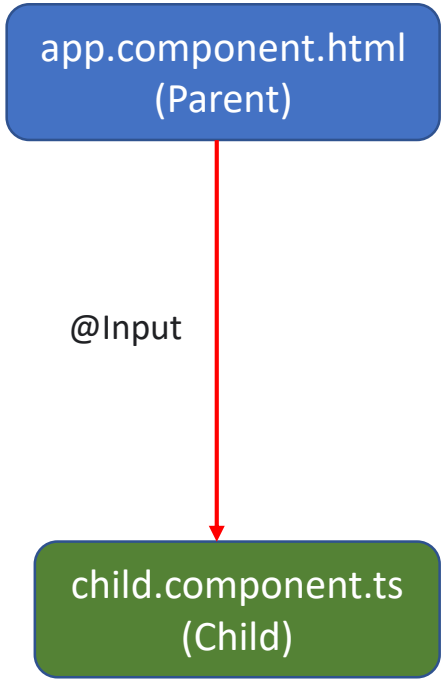
- Parent component is the outer component.
- Child components are the components inside the parent component.
- Most of the time app-component is only the parent component and rest of the components are child components



Q38 What is @Input Decorator? How to transfer data from Parent component to Child component?

- @Input decorator is used to pass data from parent component to child component.

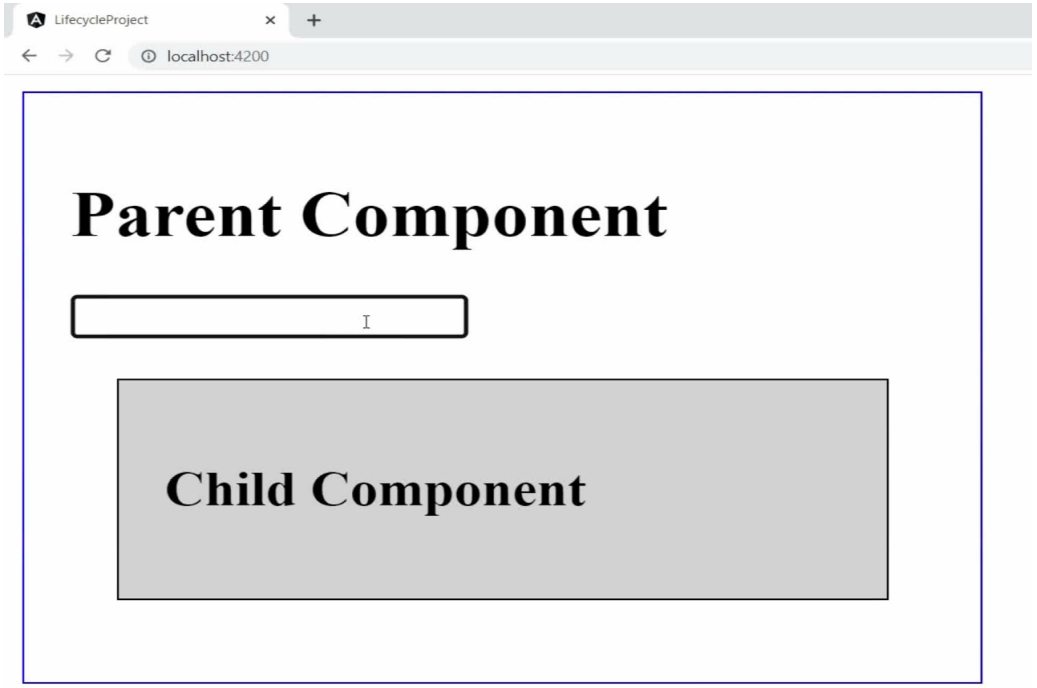


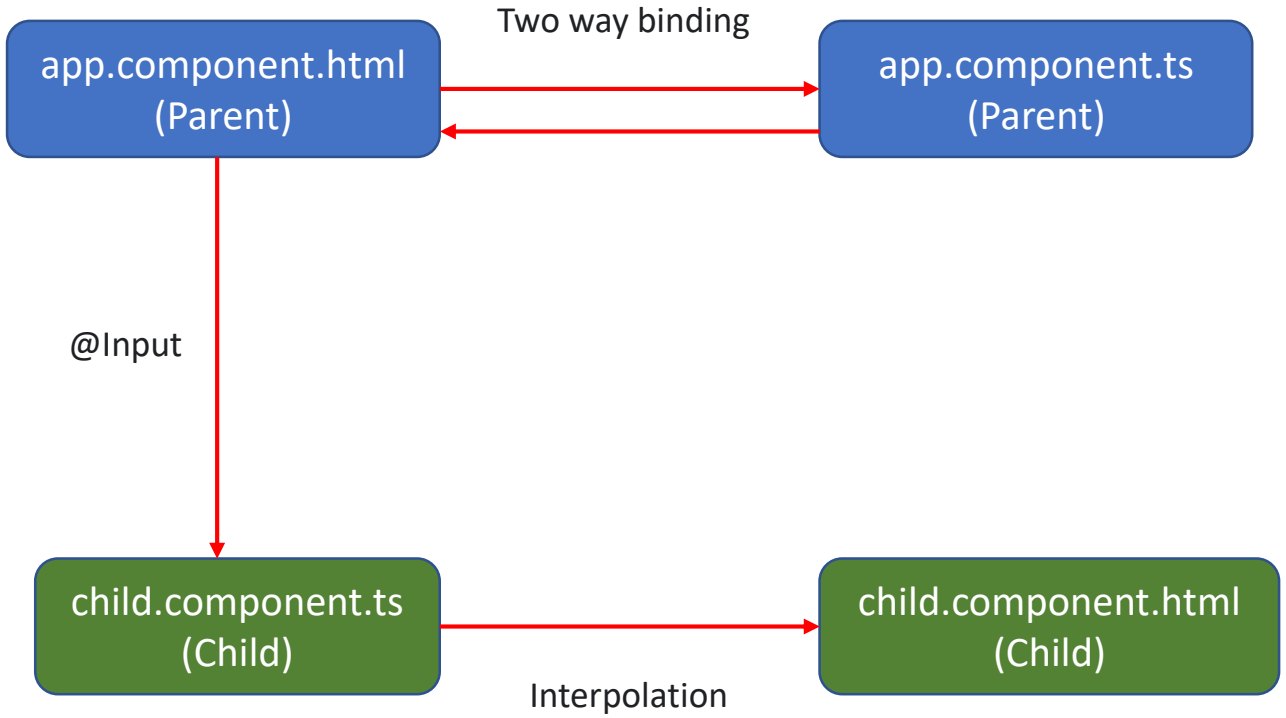


Q38

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

- @Input decorator is used to pass data from parent component to child component.

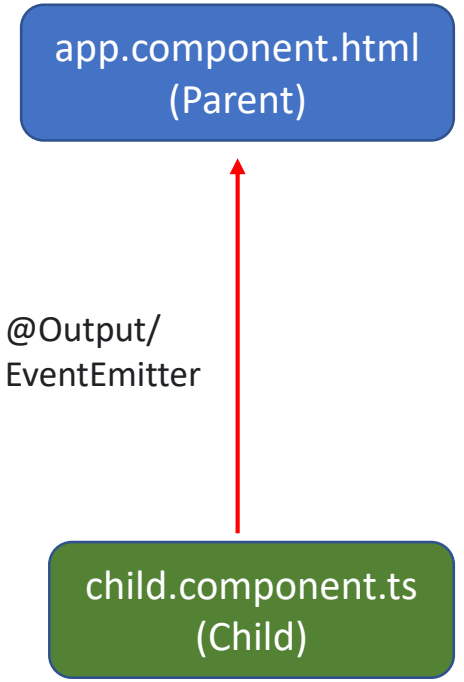




Q39

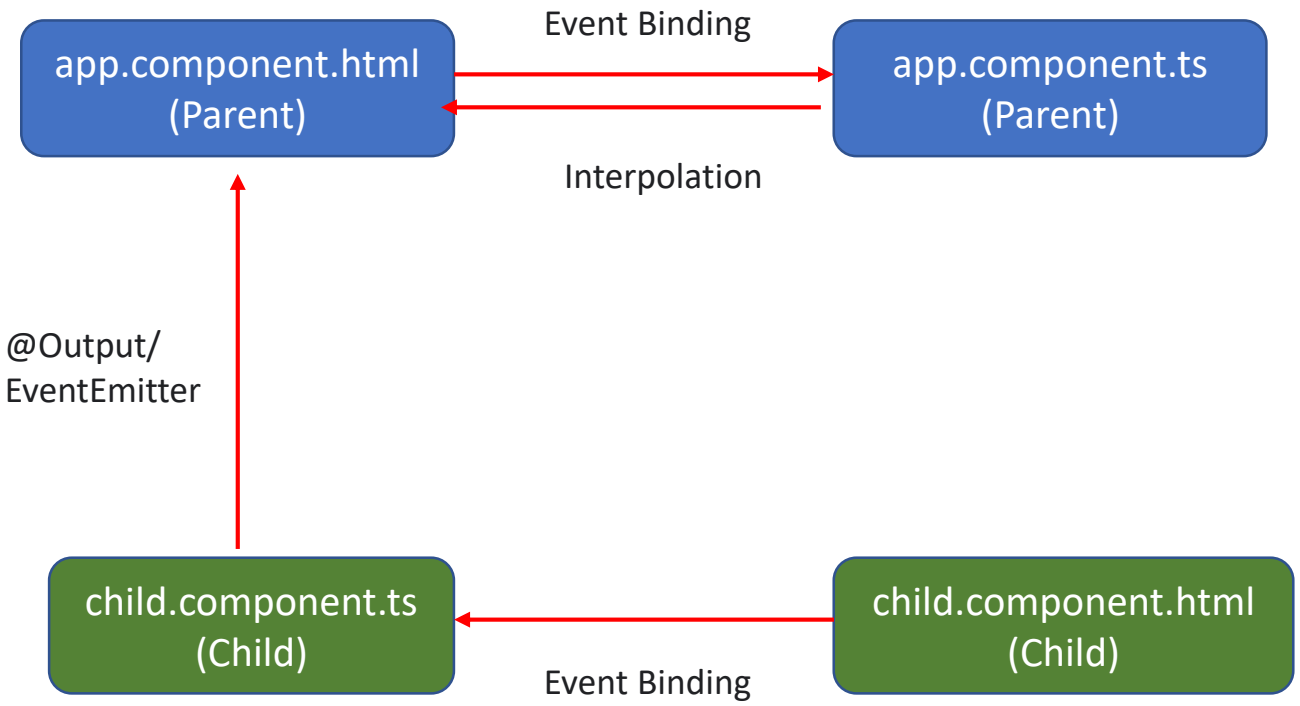
What is @Output Decorator and Event Emitter?

- @Ouptut decorator and event emitter together are used to pass data from child component to parent component.



Q39 What is @Output Decorator and Event Emitter?

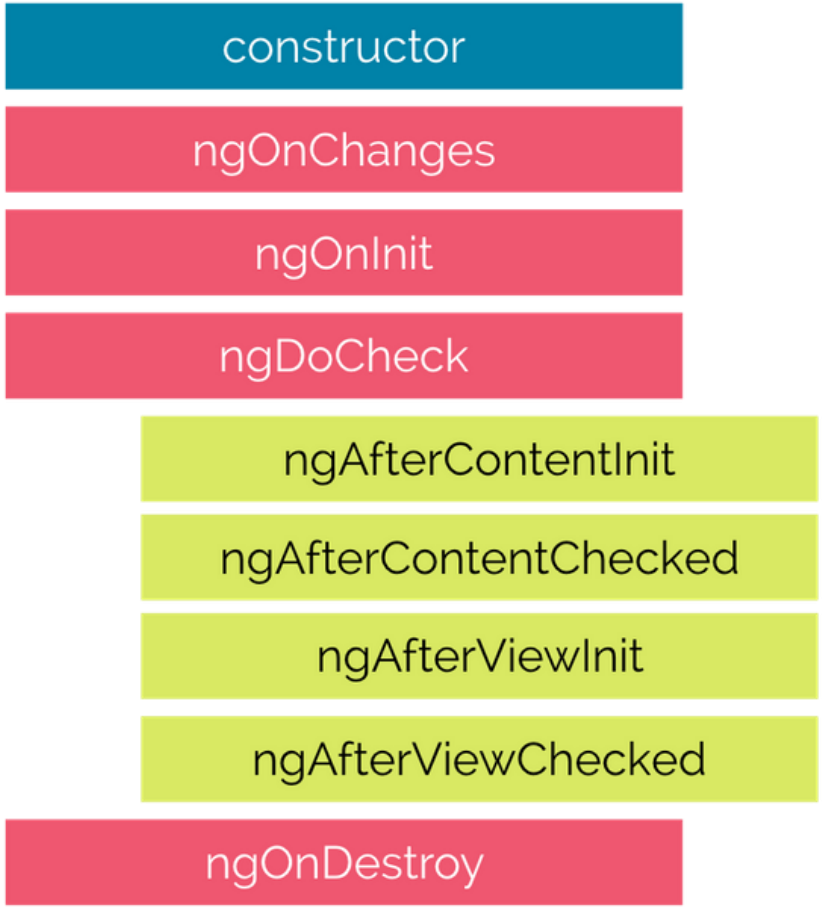
- @Ouptut decorator and event emitter together are used to pass data from child component to parent component.



Q40

What are Lifecycle Hooks in Angular? V. IMP.

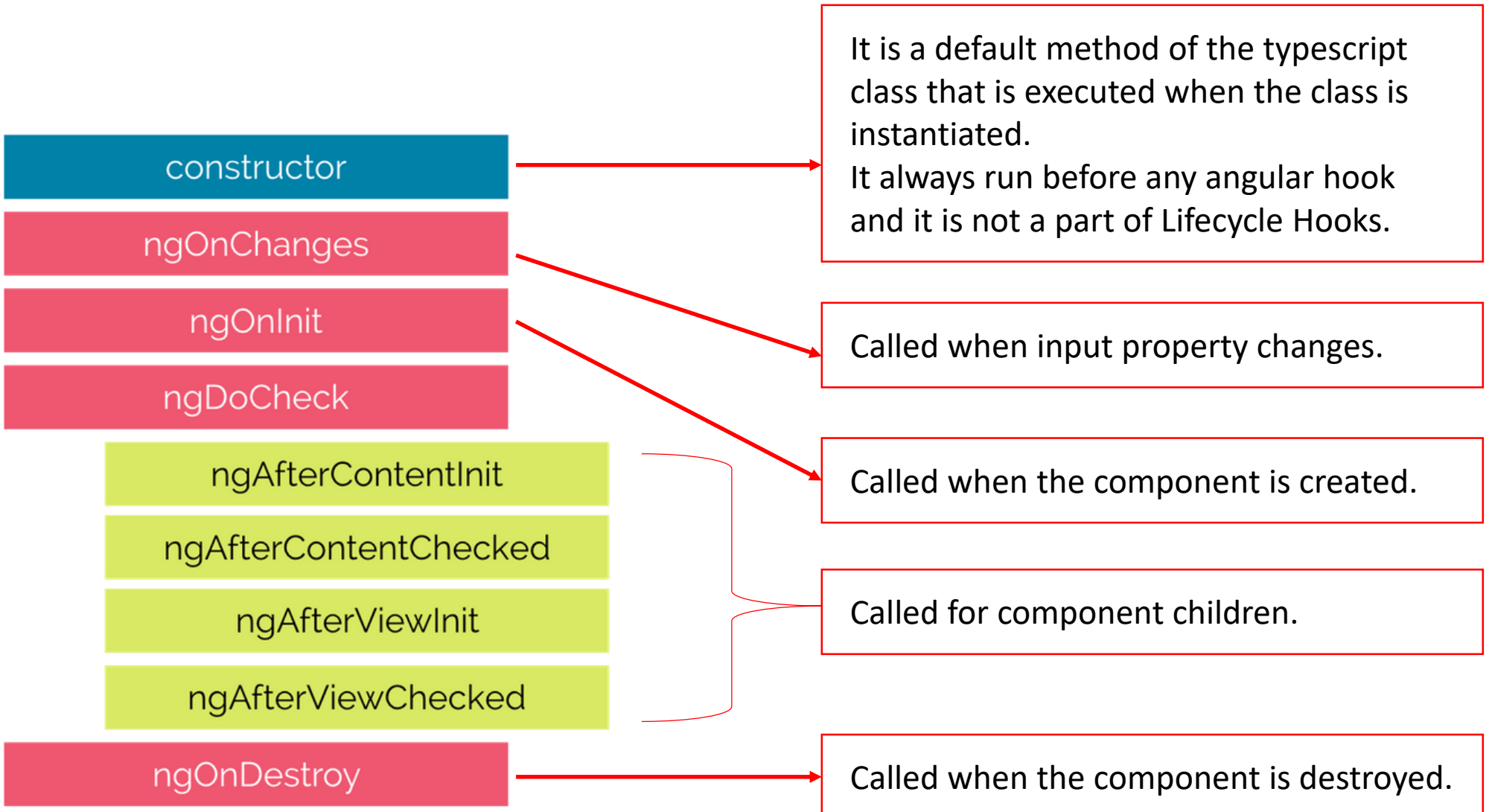
- A component from creation to destruction goes through several stages and these stages are the life cycle hooks.
- The stages will cover activities like:
  - Component instantiating.
  - Rendering the component html view.
  - Creating the child components(optional).
  - Destroying the component.
- Angular provides these hooks, inside them we can code for appropriate functionality.



Q40

What are Lifecycle Hooks in Angular? V. IMP.

- Here are some details about the hooks:



Q41

What is a Constructor in Angular?

- The constructor is a method in a **TypeScript class** that automatically gets called when the class is being **instantiated**.

- Whether constructor is a life cycle hook?

Constructor always run before any angular hook and it is not a part of Lifecycle Hooks.

- One example, when we have to write code inside constructor?

Constructor is widely used to inject **dependencies(services)** into the component class.

sample.component.ts U

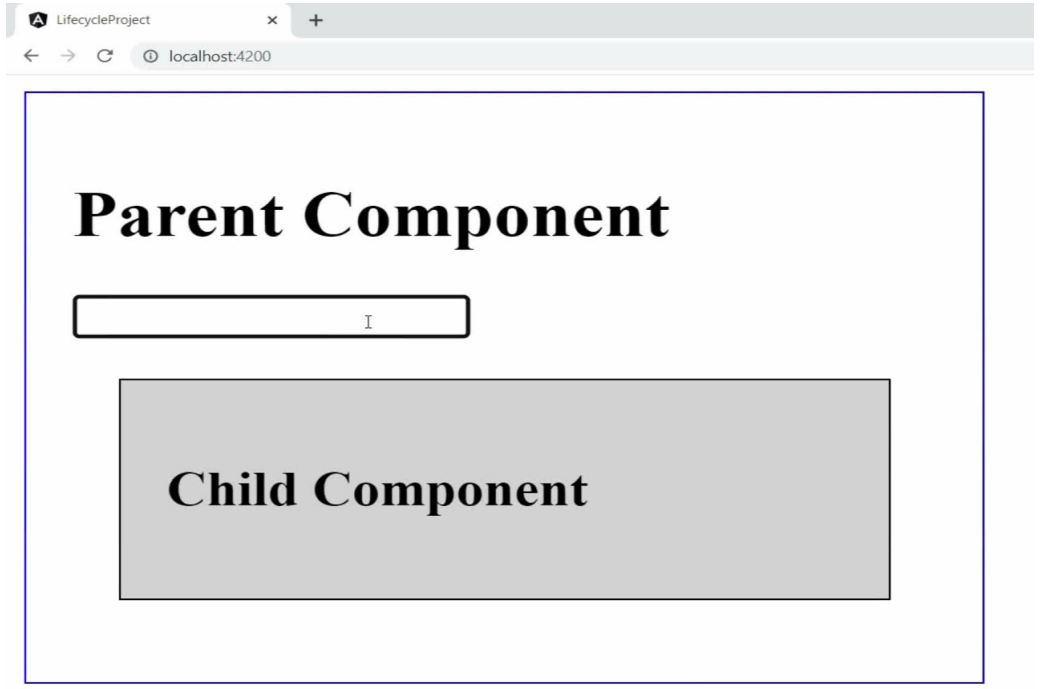
src > app > sample > TS sample.component.ts > ...

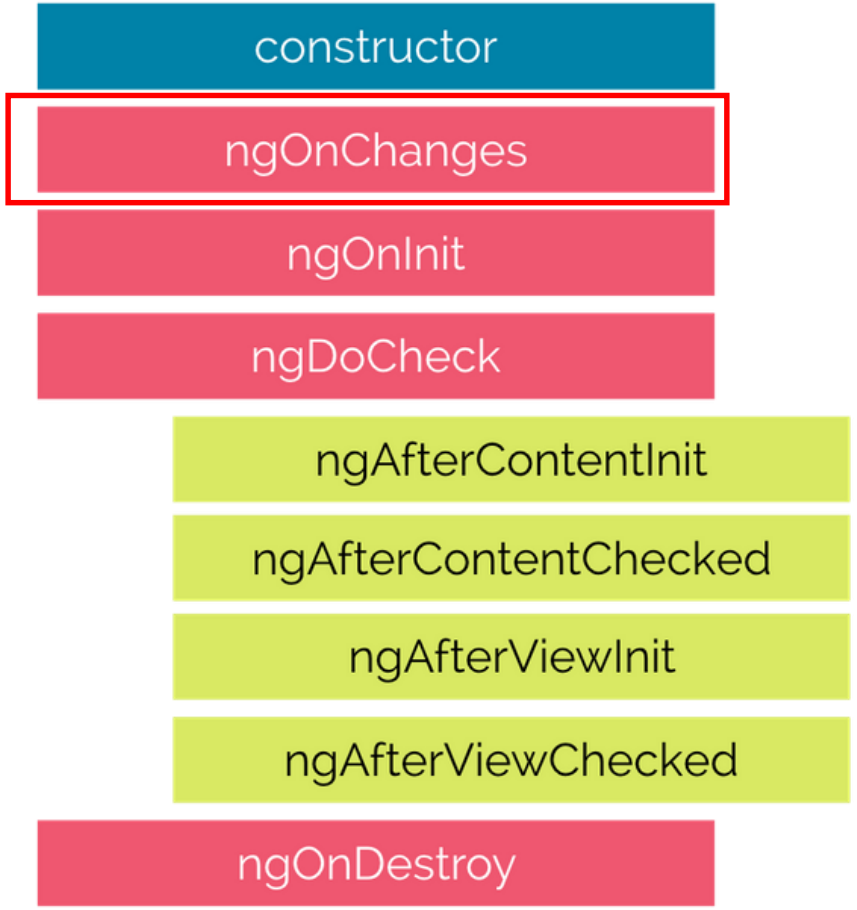
```typescript
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-sample',
  templateUrl: './sample.component.html',
  styleUrls: ['./sample.component.css']
})
export class SampleComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}

```

Q42

What is ngOnChanges life cycle hook in Angular?

- The ngOnChnages is the first life cycle hook, which angular fires when it detects any changes to the **input** property.





Q43

What is ngOnInit life cycle hook in Angular?

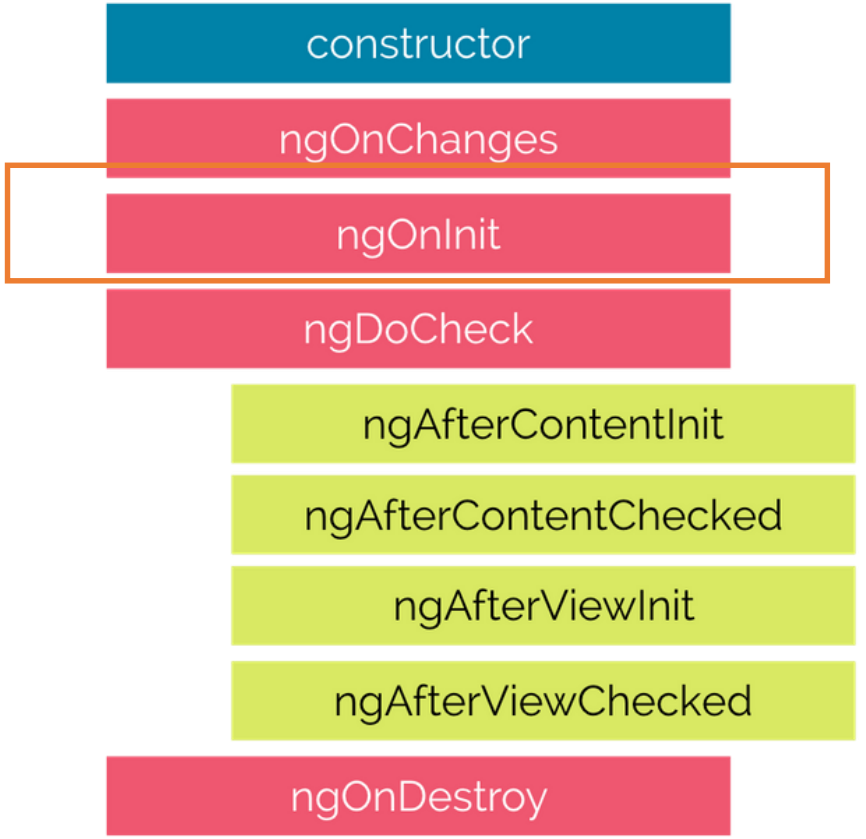
1. NgOnInit signals the activation of the **created** component.

2. This is the second hook and called after ngOnChanges.

3. NgOnInit called only once during lifecycle.

4. By default, present in the component.

```
@Component({
  selector: 'app-sample',
  templateUrl: './sample.component.html',
  styleUrls: ['./sample.component.css']
})
export class SampleComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```



Q44

What is the difference between Constructor and ngOnInit? V. IMP.

ngOnInit	Constructor
1. NgOnInit is an **Angular lifecycle hook**, which signals the activation of the created component.	The constructor is a method in a **TypeScript class,** that automatically gets called when the class is being instantiated.
2. ngOnInit is called **after** ngOnChanges lifecycle-hook.	Constructor is called **before** any lifecycle-hook.
3. When ngOnInit called, everything about component is already ready, so it’s use is to perform most of the **business logic** on component.	When constructor called, everything in component is not ready, so it's mostly used for **injecting dependencies** only**.**

TOP 100 ANGULAR INTERVIEW QUESTIONS

Chapter 8: Routing

Q45. What is Routing? How to setup Routing?

Q46. What is router outlet?

Q47. What are router links?

Q45

What is Routing? How to setup Routing? V. IMP.

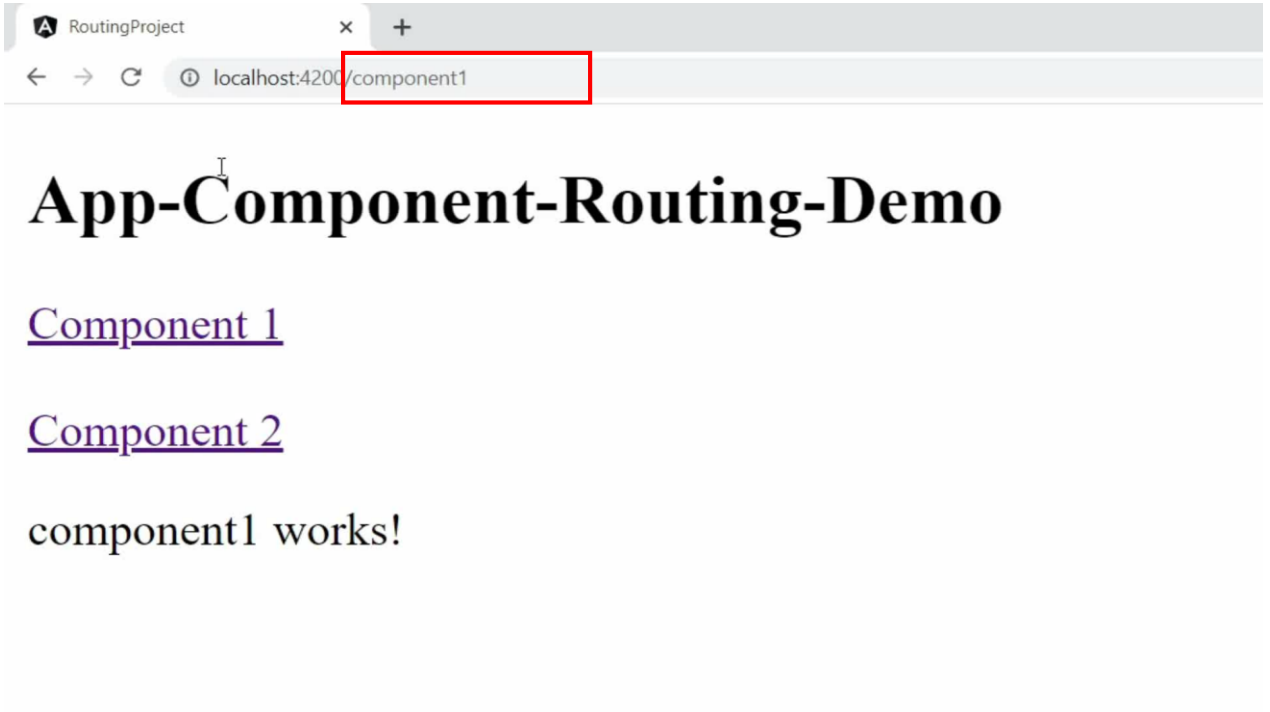
- Routing helps in navigating from one view to another view with the help of URL.

See here we have two Component links, Component 1 and Component 2 in App-Component.

Now when I click Component 1 link, then the url is updated to component1 and the component displayed is component 1.

Then when I click Component 2 link, then the url is updated to component2 and the component displayed is component 2.

So, in a single page we are navigating from one component to another component, that is routing.



Q46

What is router-outlet in Angular?

- Router-outlet in Angular works as a **placeholder,** which is used to load the different components dynamically based on the activated component via Routing.

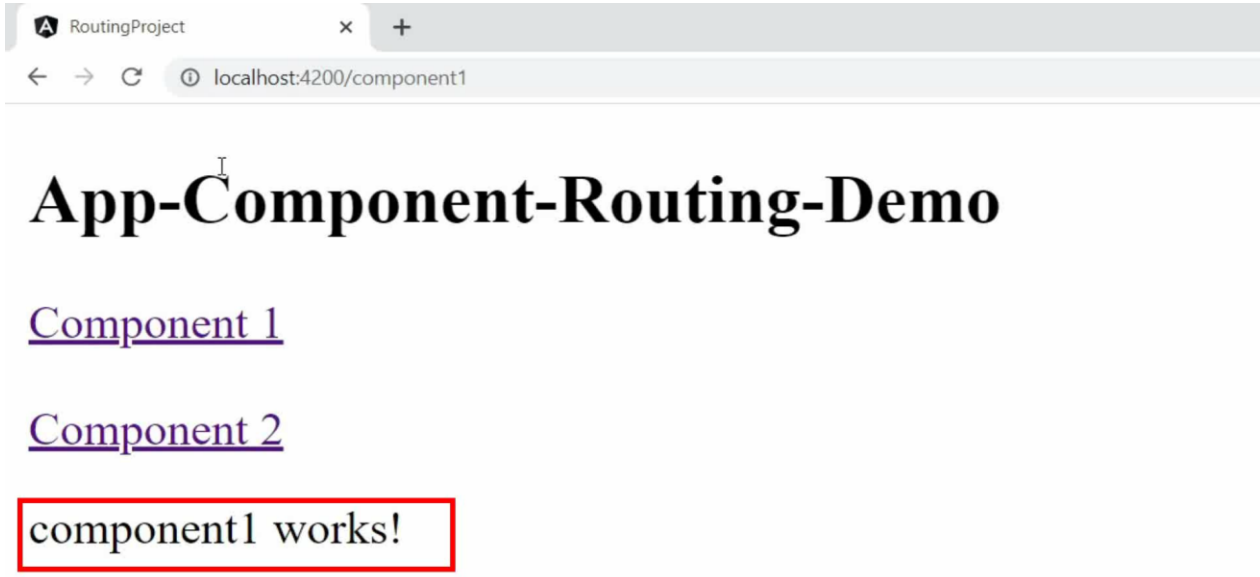
Terminal Help app.component.html - routing-project - Visual Studio

app.component.html M

src > app > app.component.html > ...

Go to component

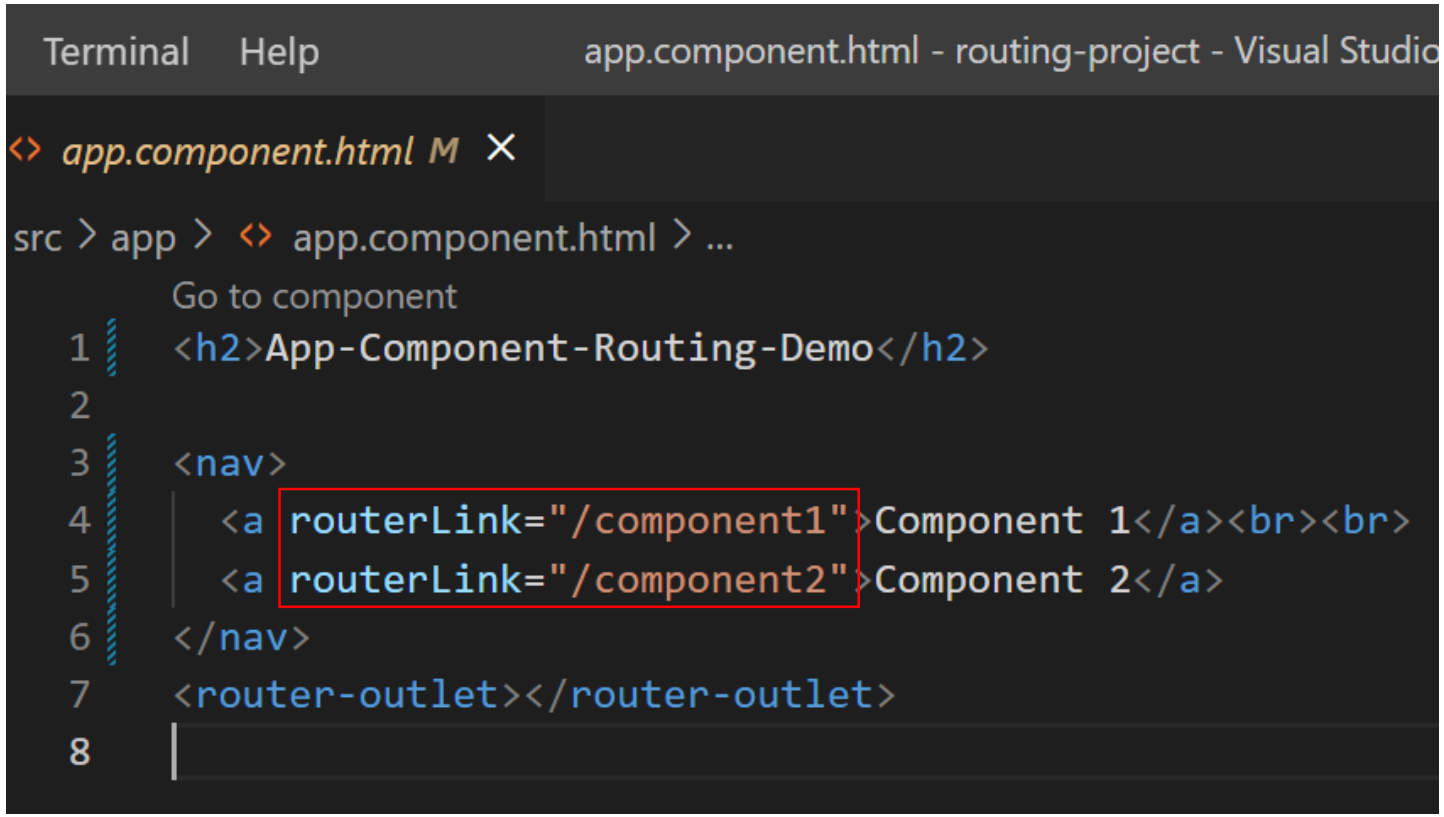
```
1  <h2>App-Component-Routing-Demo</h2>
2
3  <nav>
4    <a routerLink="/component1">Component 1</a><br><br>
5    <a routerLink="/component2">Component 2</a>
6  </nav>
7  <router-outlet></router-outlet>
8
```



Q47

What is router-Link in Angular?

- RouterLink is used for navigating to a different route.



TOP 100 ANGULAR INTERVIEW QUESTIONS

Chapter 9: Observable\ HttpClient\ RxJS

Q48. What are Asynchronous operations?

Q49. What is the difference between Promise and Observable?

Q50. What is RxJS?

Q51. What is Observable? How to implement Observable

Q52. What is the role of HttpClient in Angular?

Q53. What are the steps for fetching the data with HttpClient & Observable?

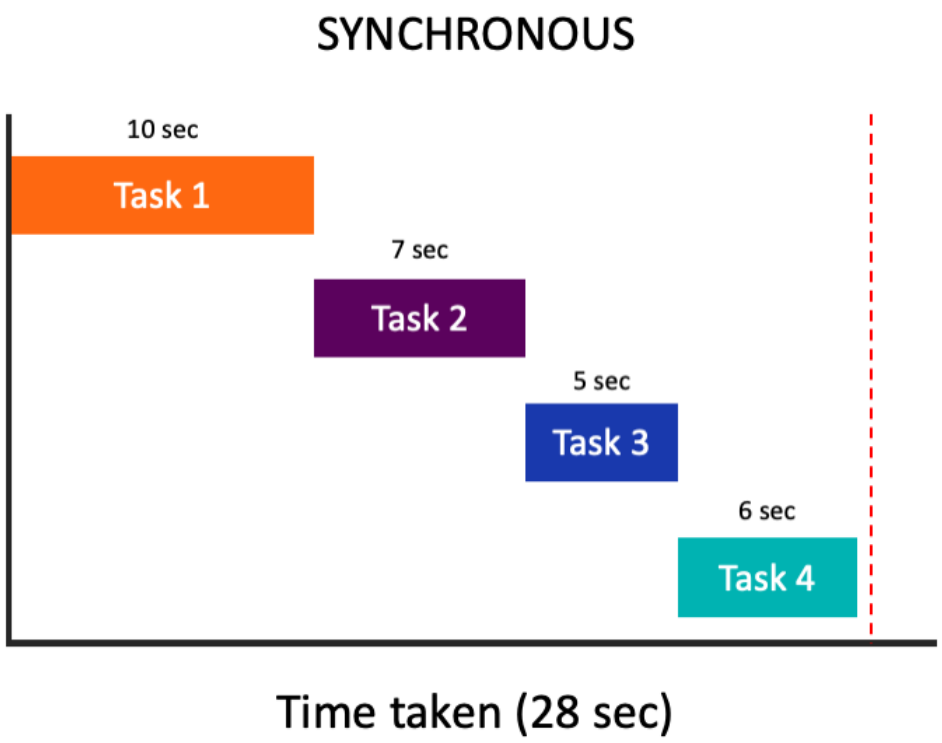
Q54. How to do HTTP Error Handling in Angular?

Q48

What are Asynchronous operations?

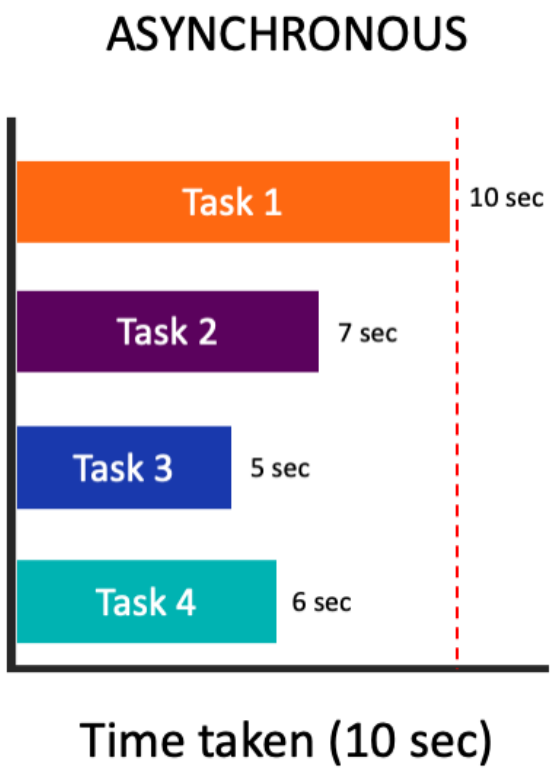
- In synchronous operations, tasks are executed in sequence and one by one.

*In image, you can see Task1, Task2, Task3, Task4 are executing **one by one** and taking in total more time 28 sec*



- In Asynchronous operations, tasks are executed in sequence and one by one.

*In image, you can see Task1, Task2, Task3, Task4 are executing **parallely** and taking in total less time 10 sec*



Q48

What are Asynchronous operations?

- How to do asynchronous operations in Angular?

Observables are used to perform asynchronous operations and handle asynchronous data

Q49

What is the difference between Promise and Observable? V. IMP.

Observables	Promises
1. Emit **multiple** values over a period of time. Also called streaming of data.	Emit a **single** value at a time.
2. Are lazy: they're not executed until we **subscribe** to them using the subscribe() method.	Are not lazy: execute **immediately** after creation.
3. Have subscriptions that are **cancellable** using the unsubscribe() method.	Are **not cancellable**.

Q49 What is the difference between Promise and Observable? V. IMP.

❖ What is the advantage of Observable over Promises?

1. In Observable, streaming of data will display continuous data to end user. So, the user has not to wait.

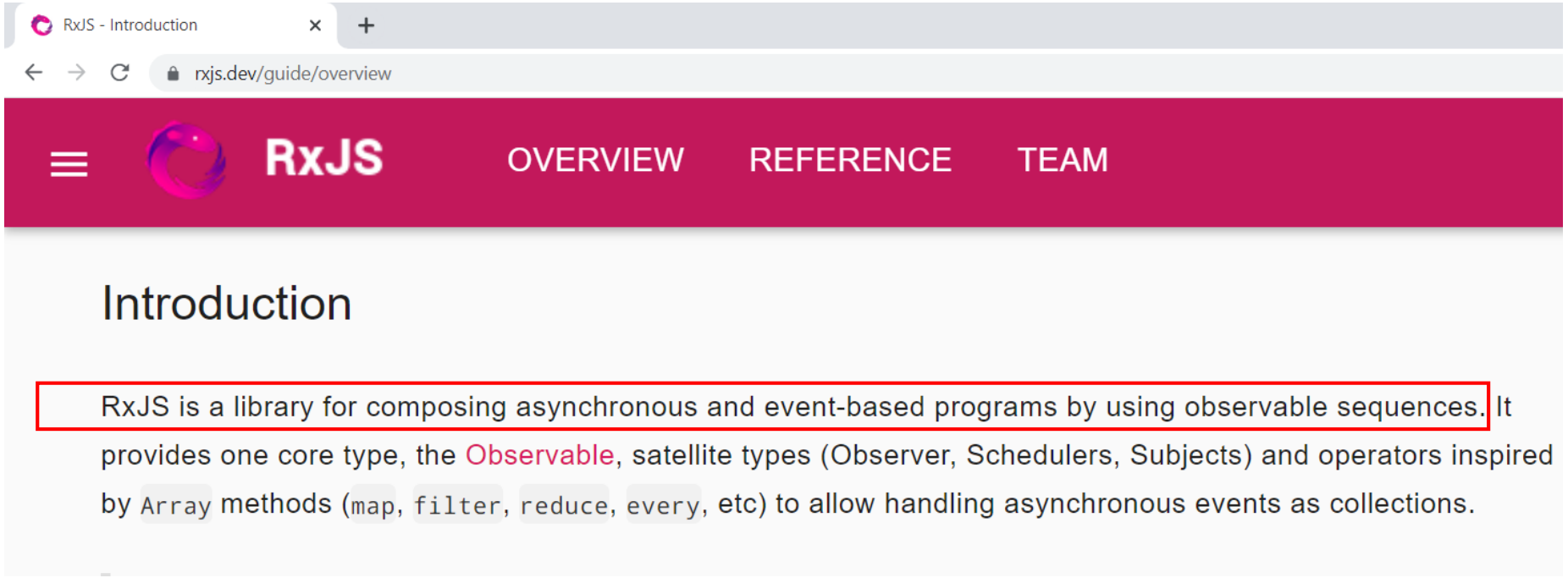
In Promises, end user have to wait until the whole data is not available, which is not good.

2. Observables are lazy, means they will not execute until someone subscribe them, which is a good thing.

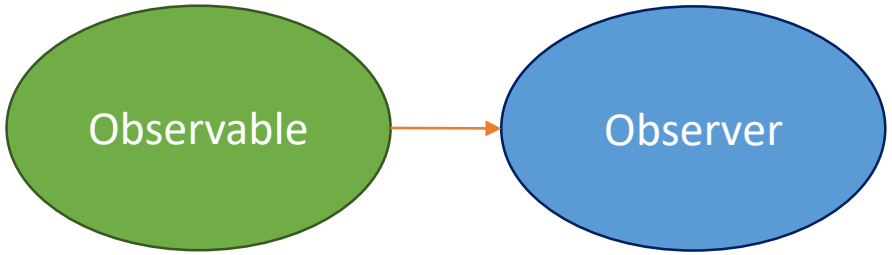
Promises are not lazy, means they will immediately execute even if there is no subscriber, which is not good.

Q50

What is RxJS?



- In simple words - RxJS is a javascript library, that allow us to work with asynchronous data stream with the help of observables.
- Observables are introduced by **RxJS** library.
- RxJS stands for Reactive Extensions for JavaScript.



Q51

What is Observable? How to implement Observable?

V. IMP.

- Observables are used to **stream data** to multiple components.
- Basically, to receive the data from API's and continuously stream it to the multiple components.
- It's a 3 step process:

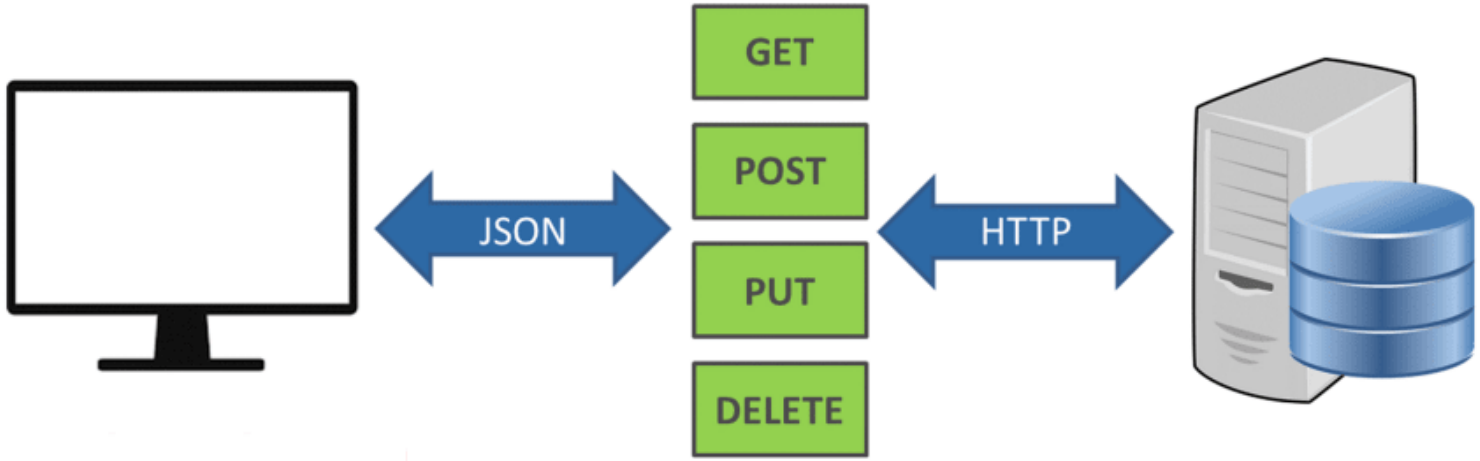
1. Import Observable from RxJS Library.
2. Create Observable & Emit Data.
3. Finally subscribe the Data.

Q52

What is the role of HttpClient in Angular?

- HttpClient is a **built-in service** class provided by Angular, which is used to perform HTTP requests(GET/POST/PUT/DELETE) to send and receive data from API or Servers.

- Package name is - @angular/common/http package



Q53

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

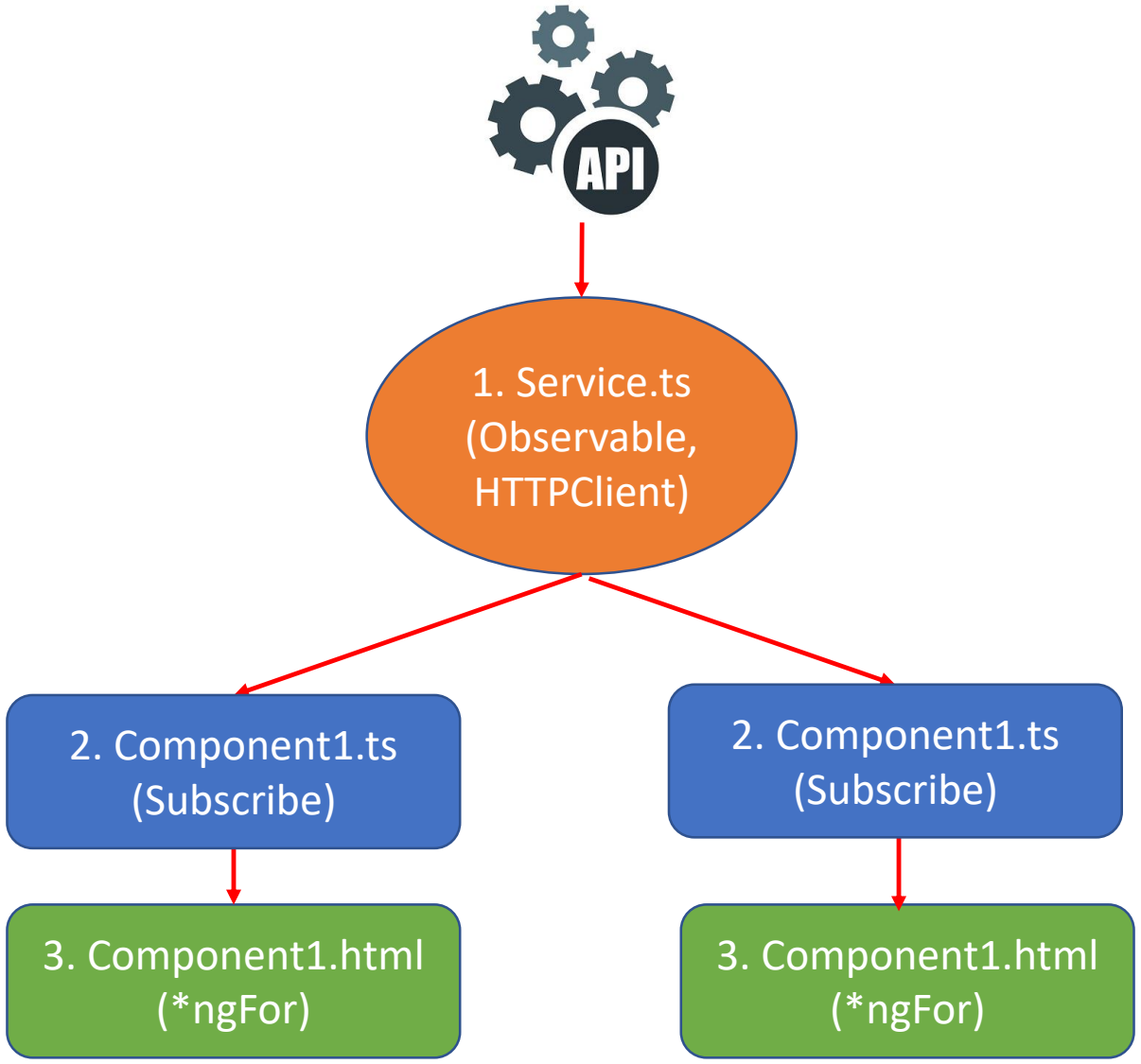
❖ 3 Steps to fetch data with HTTPClient & Observables:

Suppose we want to fetch a list of students from API.

1. Create a service and configure observable and HTTPClient to receive the data from the API.

2. In the component class file, use Subscribe method to receive the data sent by the observable.

3. Finally in component HTML file, use the for loop to iterate and display the data list.



Q53

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

1. Create a service and configure observable and HTTPClient to receive the data from the API.

```
@Injectable({
  providedIn: 'root'
})
export class StudentService {

  private _url: string = "assets/data/students.json";

  constructor(private http: HttpClient) { }

  getStudents(): Observable<IStudent[]>{

      return this.http.get<IStudent[]>(this._url);
  }

}
```

Q53

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

2. In the component class file, use Subscribe method to receive the data sent by the observable.

```
export class AppComponent implements OnInit{

  public students : any[] = [];

  constructor(private _studentService: StudentService) {
  }

  ngOnInit()
  {
      this._studentService.getStudents()
      .subscribe(data => this.students = data);
  }
}
```

Q53

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

2. Finally in component HTML file, use the for loop to iterate and display the data list.

```
<ul *ngFor="let student of students">
  <li>Student Id - {{student.id}}</li>
  <li>Student Name - {{student.name}}</li>
</ul>
```

Q54

How to do HTTP Error Handling in Angular? V. IMP.

- HTTP Error Handling can be done by using **catchError** and **throwError** functions from rxjs.

```
//The pipe is used for chaining the fucntions,
//which will execute in sequence then.
return this.http.get<IStudent[]>(this._url).pipe(
  catchError((error) => {
      return throwError(() => error);
  })
);
```

TOP 100 ANGULAR INTERVIEW QUESTIONS

Chapter 10: Typescript-Basics

Q55. What is Typescript? Or What is the difference between Typescript and Javascript?

Q56. How to install Typescript and check version?

Q57. What is the difference between let and var keyword?

Q58. What is Type annotation?

Q59. What are Built in/ Primitive and User-Defined/ Non-primitive Types in Typescript?

Q60. What is “any” type in Typescript?

Q61. What is Enum type in Typescript?

Q62. What is the difference between void and never types in Typescript?

Q63. What is Type Assertion in Typescript?

Q64. What are Arrow Functions in Typescript?

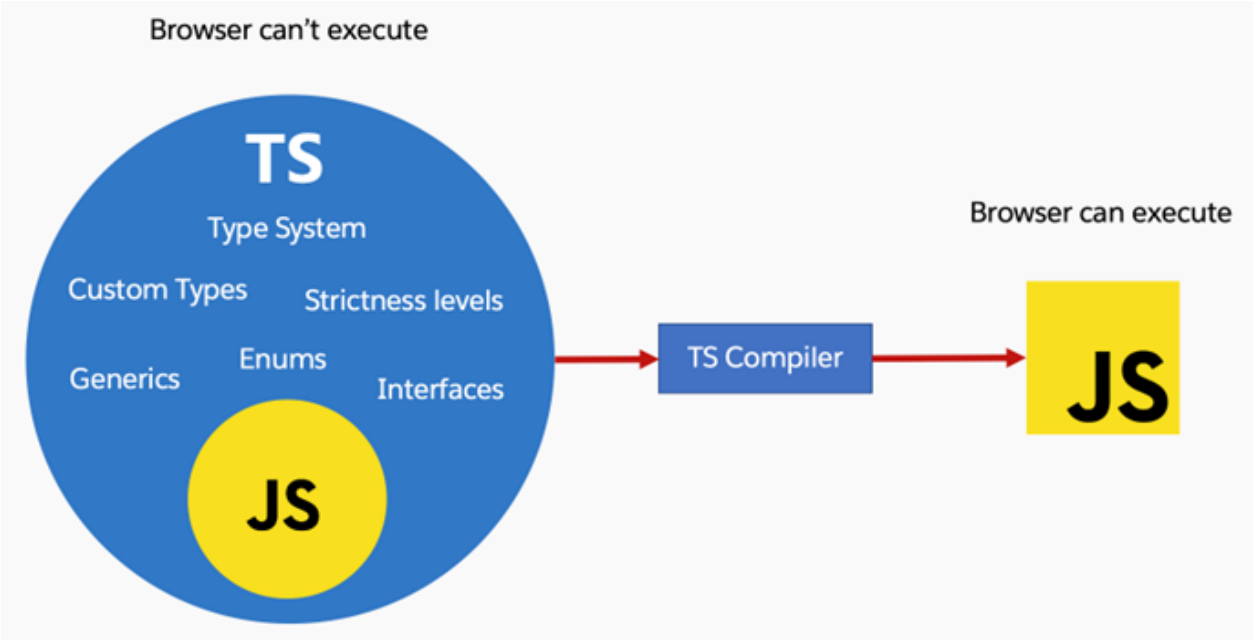
Q55

What is Typescript? What are the advantages of Typescript over Javascript? V. IMP.

- Typescript is an open-source programming language.
  It's advantage are:

1. Typescript is a strongly typed language.
2. Typescript is a superset of JavaScript.
3. It has Object oriented features.
4. Detect error at compile time.

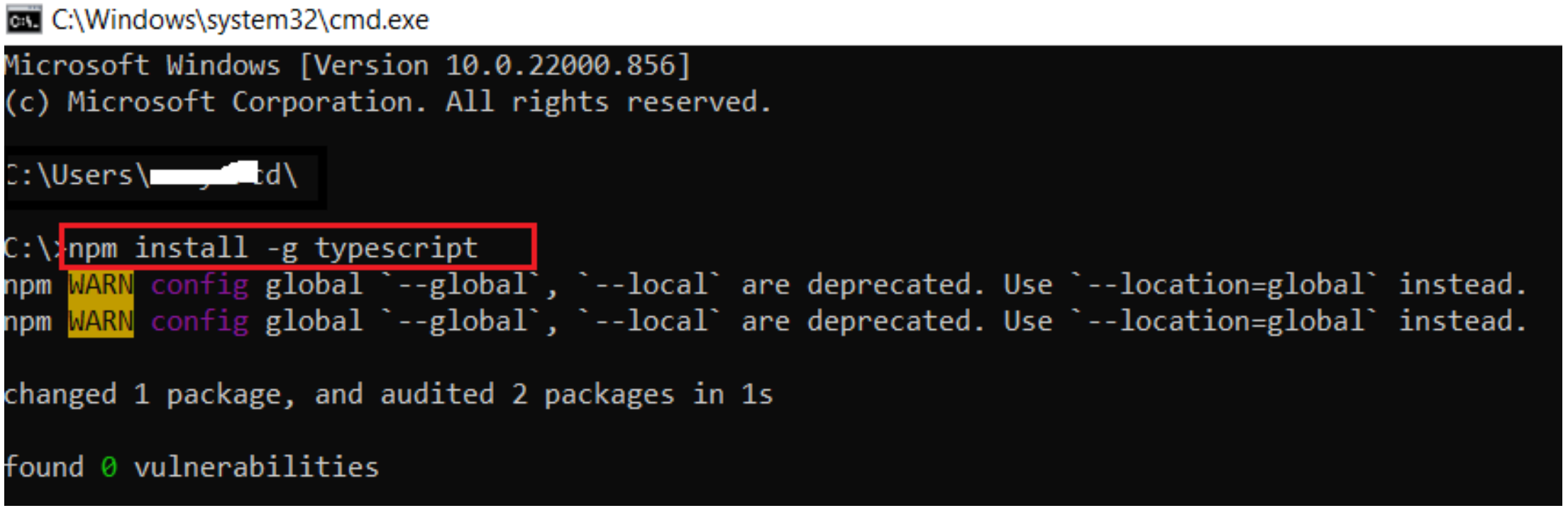
- In image, you can see TS(Typescript) contains JS(Javascript) and other additional features(type system, custom types, generics, enums, interfaces).
- Browser can't execute typescript, so finally TS Compiler will convert the TS to JS only, which browser can understand.



Q56

How to install Typescript and check version?

- Here is the command which can be used to install typescript on your system(without Angular).



Q57

What is the difference between let and var keyword?

- var and let are both used for variable declaration, but the difference between them is that **var is global function scoped and let is block scoped**.

```
function fnLetVar()
{
    //let keyword
    for(let i=0; i<5; i++)
    {
        console.log("Inside " + i);
    }
    console.log("Outside " + i);

    //var keyword
    for(var j=0; j<5; j++)
    {
        console.log("Inside " + j);
    }
    console.log("Outside " + j);
}

fnLetVar();
```

Compile time error, since let variable “i” scope is only limited inside for loop.

No error, since var variable “j” scope is not limited inside for loop, but inside the whole function fnLetVar().

Q58

What is Type annotation in Typescript?

- Type annotations helps the compiler in checking the types of variable and avoid errors.

For example, in below image when we set variable “i” as number. And now if you assign “i” a string or Boolean value, then Typescript will show the compile time error, which is good.

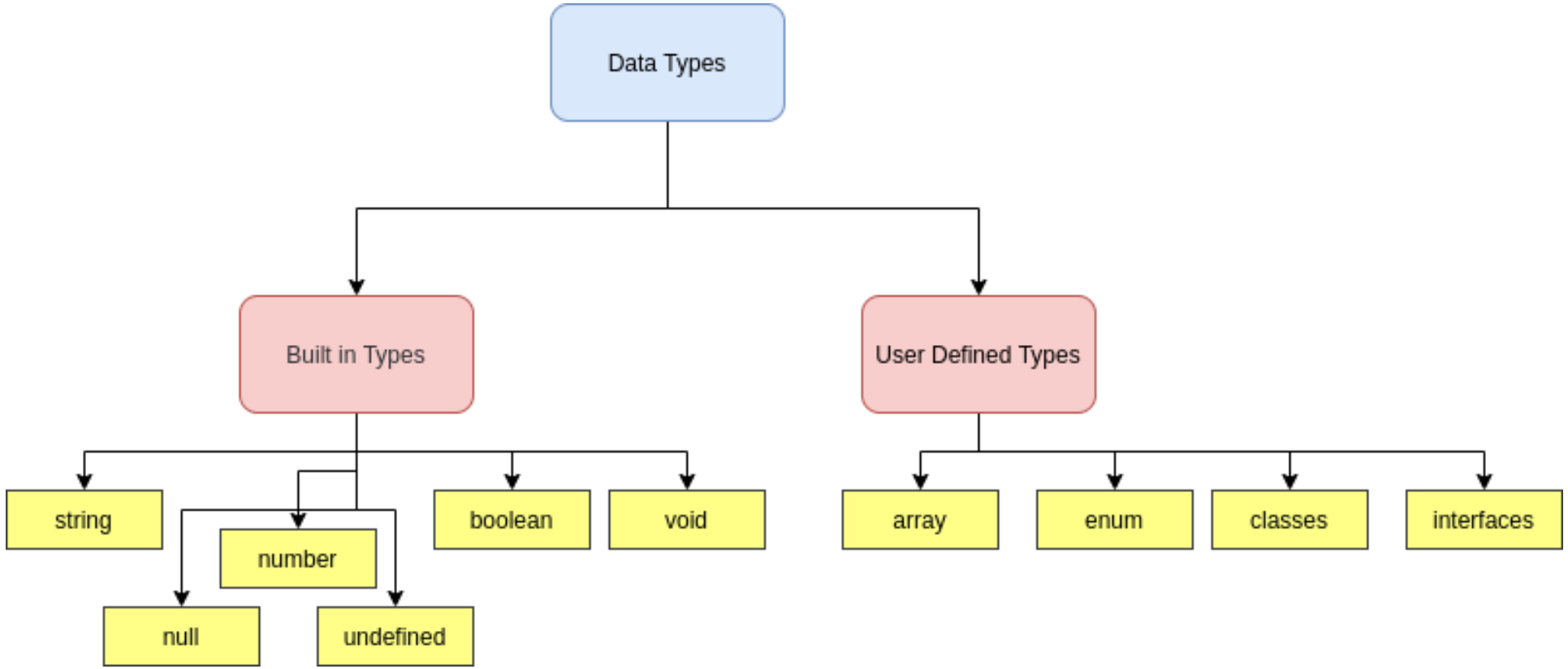
```
let i : number;

i = 1;

i = "Happy";

i = true;
```

Q59

What are Built in/ Primitive and User-Defined/ Non-primitive Types in Typescript?

- Built-in Data types are used to store simple data. Such as string, number, Boolean.

- User-Defined types are used to store complex data. Such as array, enum, classes.



Q60

What is “any” type in Typescript? V. IMP.

- When a value is of type any, then **no typechecking(no compile time errror)** will be done by compiler and the flexibility is there to do anything with this variable.

For example, see in the image there is no compile time error for assigning different types to variable “x” because it is of any type.

```
let x: any;

x = 1;

x = "Happy";

x = true;

x();

x.AnyProperty = 10;
```

Q61

What is Enum type in Typescript?

- Enums allows to define a set of **named constants**.

Suppose you have a family of constants(Directions) like this in image.
Now if you want to insert some direction, for example SouthWest between 1 and 2, then it will be a problem, as you have to change this in all of your application.

```
const DirectionNorth = 0;
const DirectionSouth = 1;
const DirectionWest = 2;
const DirectionEast = 3;

let dir = DirectionEast;
```

So here enum is a better way as you don't have to assign the numbers then.

```
enum Direction {
    North,
    South,
    West,
    East
}

let dirNew = Direction.East;
```

Q62

What is the difference between void and never types in Typescript?

- void means no type. It is used when the function will return empty.

In image, fnVoid() function will return empty.

```
let message = "Happy";

function fnVoid(message: string): void {

    console.log("void " + message);
}
```

- never means it will return never. It is used to throw error only.

For example, in image fnNever function which will not reach to its last line, and it always throws an exception.

```
function fnNever(message: string): never {

    throw new Error(message);

}
```

Q63

What is Type Assertion in Typescript?

- Type assertion is a technique that informs the compiler about the type of a variable.

For example, in below image, if we do not put <String>, then typescript will try to assume the type of "secondname" by itself automatically.
But by putting <String>, we are asserting and informing the compiler about the type of secondName as String and now compiler will not do automatic typechecking.

```
let myname: any = "Happy";

// Conversion from any to string
let secondName = <String> myname;

console.log(secondName);
console.log(typeof secondName);
```

Q64

What are Arrow Functions in Typescript? V. IMP.

- An **arrow function** expression is a compact alternative to a traditional function expression.

In below image, you can see how the same function can be written using arrow function.

```
//Normal function approach
let x = function(a, b)
{
    a * b;
}

//Arrow function approach
let y = (a, b) =>  a * b;
```

TOP 100 ANGULAR INTERVIEW QUESTIONS

Chapter 11: Typescript - OOPS

Q65. What is Object Oriented Programming in Typescript?

Q66. What are Classes and Objects in Typescript?

Q67. What is Constructor?

Q68. What are Access Modifiers in Typescript?

Q69. What is Encapsulation in Typescript?

Q70. What is Inheritance in Typescript?

Q71. What is Polymorphism in Typescript?

Q72. What is Interface in Typescript?

Q73. What's the difference between extends and implements in TypeScript

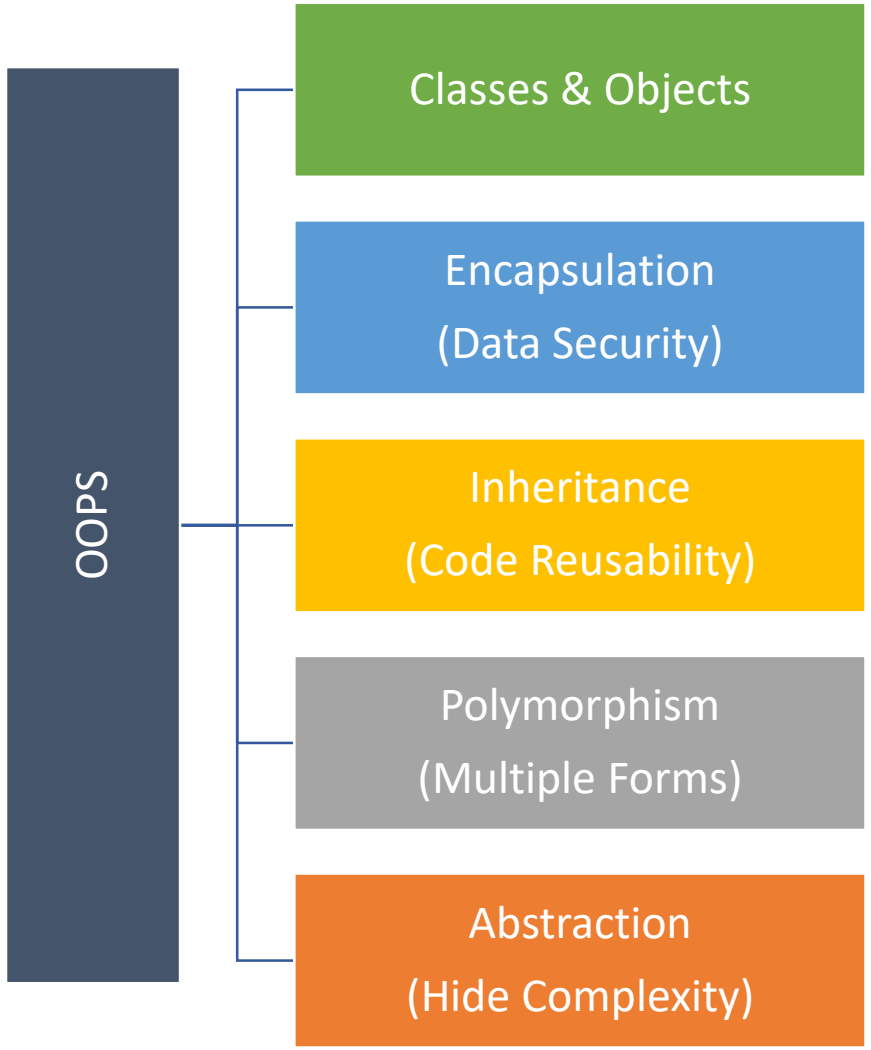
Q74. Is Multiple Inheritance possible in Typescript?

Q65

What is Object Oriented Programming(OOPS) in Typescript? V. IMP.

- Object oriented programming is used to design structured and better applications.

In the image, you can see the concepts used in object oriented programming.



Q66

What are Classes and Objects in Typescript?

- Classes are blueprint for individual objects.
- Objects are instances of a class.

Person is a class with two properties(name, age) and one method(familyCount)

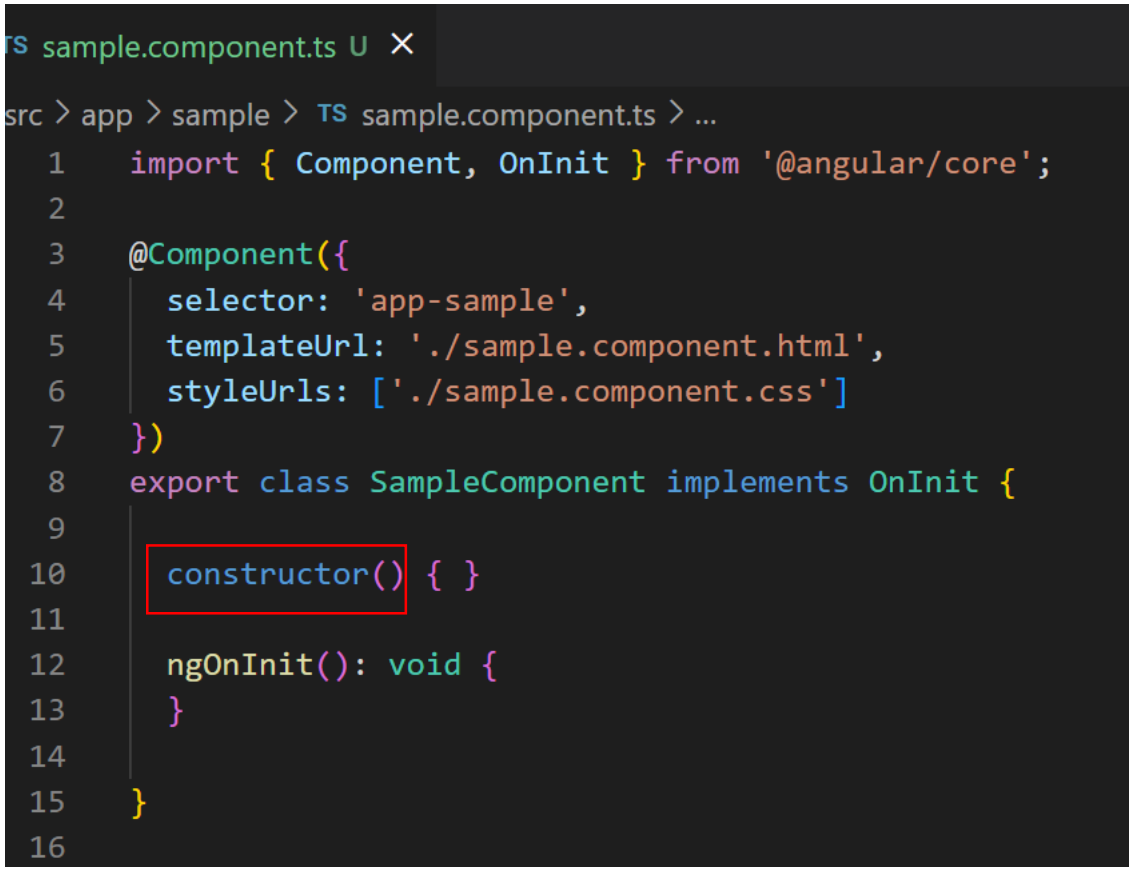
objPerson is the object/instance of Person class. By this object only we can set Person class properties and call the method of the class.

```typescript
class Person {
    name: string;
    age: number;

    familyCount(){
      return 6;
    }
}

var objPerson = new Person();

let myName = objPerson.name = "Happy";
let myAge = objPerson.age = 10;

let countFamily = objPerson.familyCount();
```

Q67

What is a Constructor in Angular?

- The constructor is a method in a **TypeScript class** that automatically gets called when the class is being **instantiated**.
- Constructor always run before any angular hook and it is not a part of Lifecycle Hooks.
- Constructor is widely used to inject **dependencies(services)** into the component class.



Q68

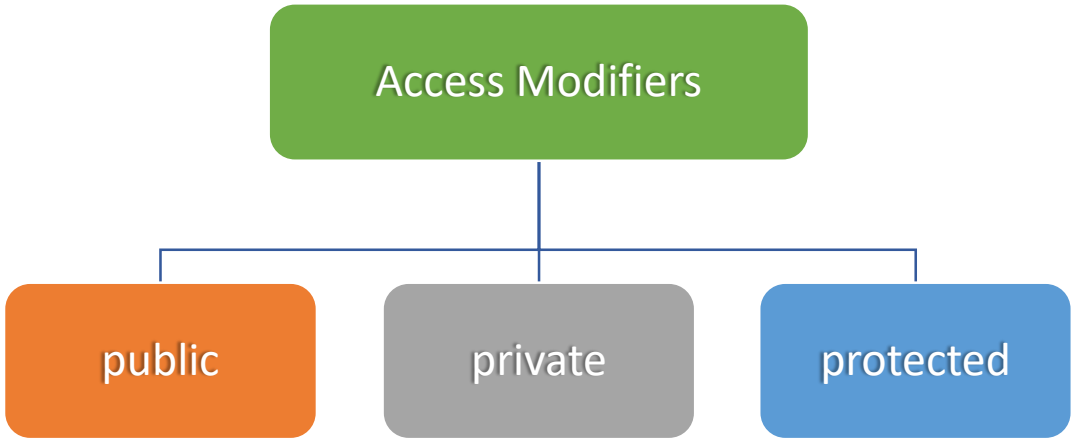
What are Access Modifiers in Typescript?

❖ 3 types of Access modifiers in Typescript:

1. The public modifier allows class varibale, properties and methods to be accessible from all locations.

2. The private modifier limits the visibility to the same-class only.

3. The protected modifier allows properties and methods of a class to be accessible within same class and within subclasses/derived/child.



Q69

What are Access Modifiers in Typescript?

1. empId no error, as it is public.

2. empName error, as it is out of the class and it is private.

3. empAge error, as it is out of the class and it is protected.

3. But in PermanentEmployee empAge no error, because it is inside the derived class(PermanentEmployee) of Employee.

```
class Employee
{
    public empId: number;
    private empName: string;
    protected empAge: number;
}

var objEmployee = new Employee();
objEmployee.empId = 10;
objEmployee.empName = "Amit";
objEmployee.empAge = 100;

class PermanentEmployee extends Employee
{
    empId = 100;
    empAge = 30;
}
```

Q70

What is Encapsulation in Typescript?

- Encapsulation is the wrapping up of data and function together to access the data.

For example, here in code _empId is the data. We can make it public and then it can be accessible outside the Employee class.

But as per OOPS concept encapsulation this is wrong as it can lead to data security issues.

Therefore, we should mark this _empId data as private. And then create a function(getEmpId), which will help in accessing this _empId outside of this class.

Outside the class, in the last line we are calling getEmpId method to access/dispaly the data _empId.

```
class Employee
{
    private _empId: number; //data

    getEmpId(){
        return this._empId;
    }
}

let objEmployee = new Employee();

console.log(objEmployee.getEmpId());
```

- What is the advantage of Encapsulation?

This is good for data security to prevent direct data access.

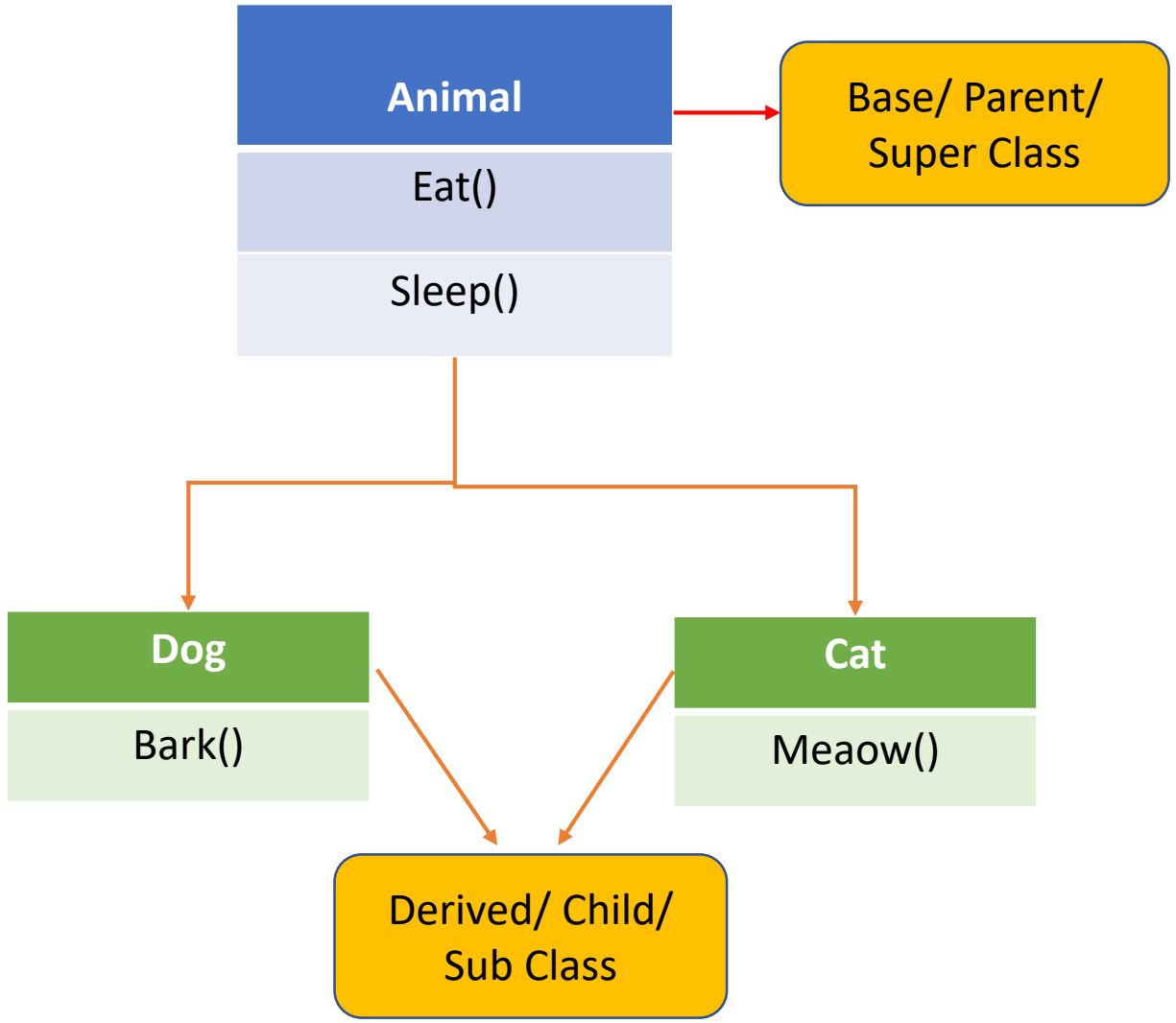
Q71

What is Inheritance in Typescript?

- Inheritance is a feature where derive class will acquire the properties and methods of base class.

For example, in the image Dog and Cat class are derived from Animal base class.

Therefore, Eat() and Sleep() methods of Animal base class will be automatically available inside Dog and Cat derived class without even writing it.



- What is the advantage of Inheritance?

Inheritance is good for reusability of code. As you can write one method in base class, and then use it in multiple derived classes.

Q71

What is Inheritance in Typescript?

```
class Animal{
    Eat(){
        console.log("eat");
    }
}

class Dog extends Animal{
    Bark(){
        console.log("bark");
    }
}

let objectDog = new Dog();

objectDog.Eat();

objectDog.Bark();
```

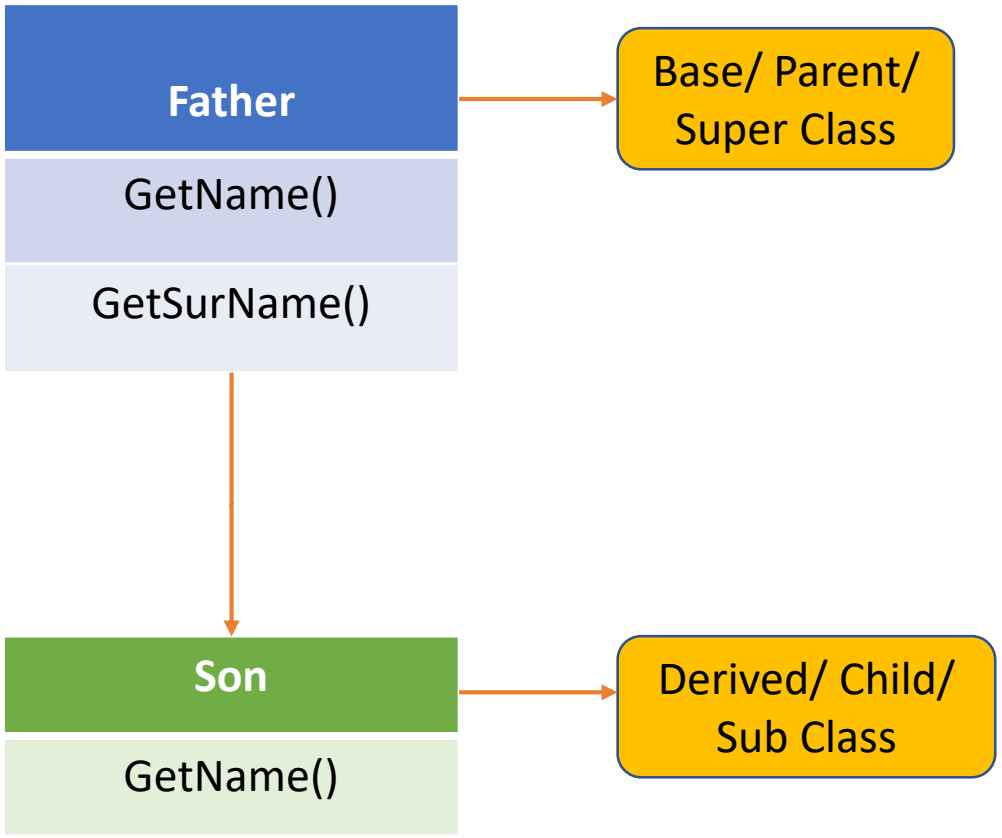
Both Eat() and Bark() method are accessible via Dog class object. Eat is from the base class but.

Q72

What is Polymorphism in Typescript?

- Polymorphism means same name method can take multiple Forms.

For example, in the image Son can inherit the GetSurName() method implementation of father. But the GetName() method implementation should be different for son. So son can have a different implementation but the method name can be same as GetName().



- What is the advantage of Polymorphism?

Like in dictionary, a word "Running" can have multiple meanings: Running a race or Running a computer.

Similarly in programming, Polymorphism can be used to do multiple things with the same name.

Q72

What is Polymorphism in Typescript?

GetName() method of Father class object output is different, and Getname() method of Son class object output is different. Same name method taking multiple forms.

```
class Father {
    GetName() {
        console.log('Amit');
    }
    GetSurName() {
        console.log('Singh');
    }
}

class Son extends Father {
    GetName() {
        console.log('Ravi');
    }
}
let objectFather = new Father();
objectFather.GetName();
//Output: Amit

let objectSon = new Son();
objectSon.GetName();
//Output: Ravi
```

Q72

What is Interface and why to use them in Typescript? V. IMP.

- An interface is a contract where methods are only declared.

- What is the advantage of creating Interface?

For example, we have multiple classes PermanentEmployee, ContractEmployee etc.
They all must have the Role() method, but the body of Role() method can be different.
Now to maintain consistency we will create the Interface IEmployee, with just declaration of Role() method and implement this interface in all the classes.
This will make sure all the Employees classes are in sync with the Role method. It will also help in unit testing.

- One advantage is, multiple inheritance can be achieved via interfaces only.

- Also, if we create interfaces for the classes in application then unit testing is very easy, otherwise it is difficult.

```typescript
interface IEmployee
{
    Role(): void;
}

class PermanentEmployee implements IEmployee{

    Role(){
        console.log("Lead");
    }
}

class ContractEmployee implements IEmployee{

    Role(){
        console.log("Developer");
    }
}
```

Q73

What's the difference between extends and implements? V. IMP.

- **Extends** used for base class inheritance.

```
//Parent Class
class Employee{
    Salary()
    {
        console.log("salary");
    }
}

//Child Class
class PermanentEmployee extends Employee {

}
```

- **Implements** used for interface inheritance.

```
interface IEmployee{

    Salary();

}

class PermanentEmployee implements IEmployee {
    Salary() {
        {
            console.log("50000");
        }
    }
}
```

Q74

Is Multiple Inheritance possible in Typescript?

- Multiple inheritance means when one derived class is inherited from multiple base classes.

- Multiple inheritance in typescript only possible via Interfaces.

See in first image there is the compile time error in Employee2 as multiple base classes are not allowed, but in second image with multiple interfaces iit is working.

```
//Parent Class 1
class Employee1{
    Salary1()
    {
        console.log("salary");
    }
}

//Parent Class 2
class Employee2{
    Salary2()
    {
        console.log("salary");
    }
}

//Child Class
class PEmployee extends Employee1, Employee2 {

}
```

```
interface IEmployee1{

    Salary1();
}
interface IEmployee2{

    Salary2();

}

class TEmployee implements IEmployee1, IEmployee2 {
    Salary1() {
        console.log("300000");
    }
    Salary2() {
            console.log("500000");
    }
}
```

TOP 100 ANGULAR INTERVIEW QUESTIONS

Chapter 12: Angular Forms

Q75. What are Angular Forms? What are the type of Angular Forms?

Q76. What is the difference between Template Driven Forms & Reactive Forms?

Q77. How to setup Template Driven Forms?

Q78. How to apply Required field validation in template driven forms?

Q79. What is Form Group and Form Control in Angular?

Q80. How to setup Reactive Forms?

Q81. How to do validations in reactive forms?

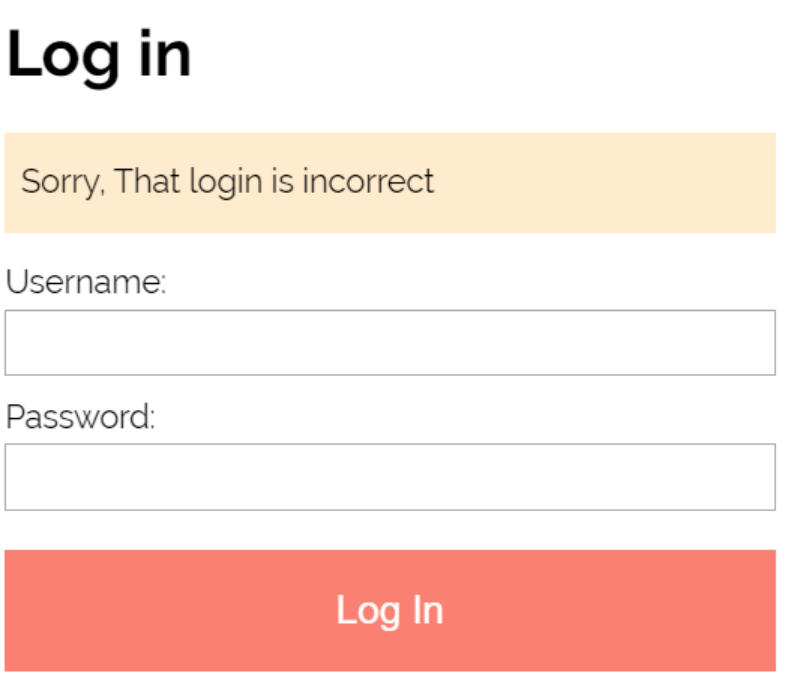
Q75

What are Angular Forms? What are the types of Angular Forms? V. IMP.

- **Angular forms** are used to handle user's input.

For example, login form, registration form, feedback form, update profile form etc etc.

All these are forms, in which user will input and then submit the information.



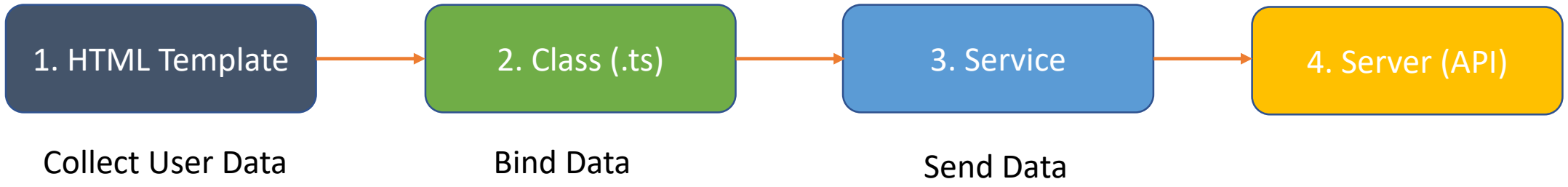
Q75

What are Angular Forms? What are the types of Angular Forms?

V. IMP.

❖ 4 steps data flow process in Angular forms-

1. The user will input the data in the HTML template file. .
2. Then data will be bind in the component typescript class file.
3. Then it will be sent to the Angular service.
4. Angular service will then finally send it to the Server which will receive the data and process the data. (In most cases this server will be an API only.)



Q75

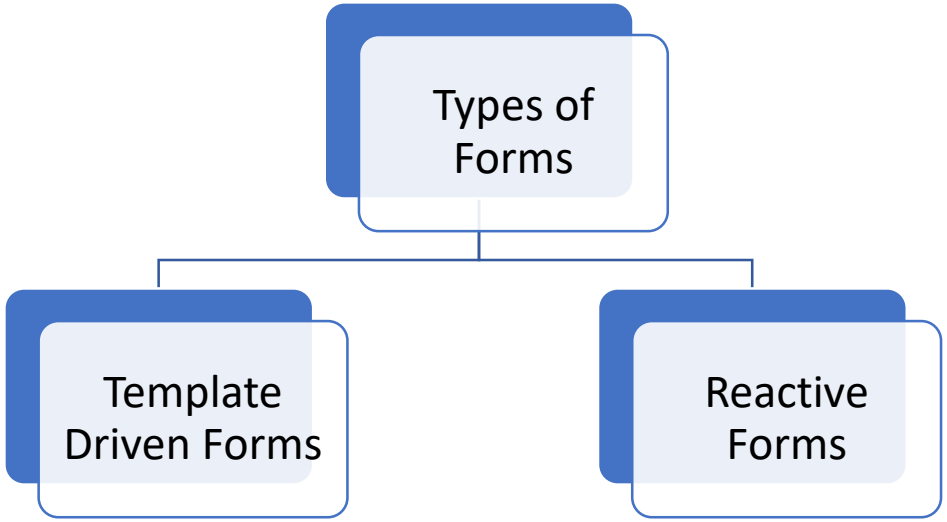
What are Angular Forms? What are the types of Angular Forms?

V. IMP.

- There are 2 types of forms in Angular:

1. Template driven forms - In template driven forms, most of the code will be written and handled in HTML templates only.

2. Reactive forms - in reactive forms most of the code will be written in component typescript class file.



Q76

What is the difference between Template Driven Forms & Reactive Forms? V. IMP.

Template Driven Forms	Reactive Forms
1. Most code and validation logic is written in **HTML template**.	Most code and validation logic is written in **component typescript** class file.
2. Have to add **FormsModule** in AppModule to activate it.	Have to add **ReactiveFormsModule** in AppModule to activate it.
3. Used when application is simple and have **less controls**.	Used when application is complex and have **more controls**.

Q77

How to setup Template Driven Forms?

- *Below is the sample code to setup login form using template driven form approach in html template file. Here we have used template reference variable (#loginForm).*

```
<h3>Template Driven Form</h3>

<form #loginForm="ngForm" (ngSubmit)="login(loginForm.value)">

  <input type="text" required name="email" placeholder="Email" ngModel>
  <br><br>

  <input type="password" name="password" placeholder="Password" ngModel>
  <br><br>

  <button [disabled]="loginForm.invalid">Login</button>

</form>
```

Q78

How to apply Required field validation in template driven forms?

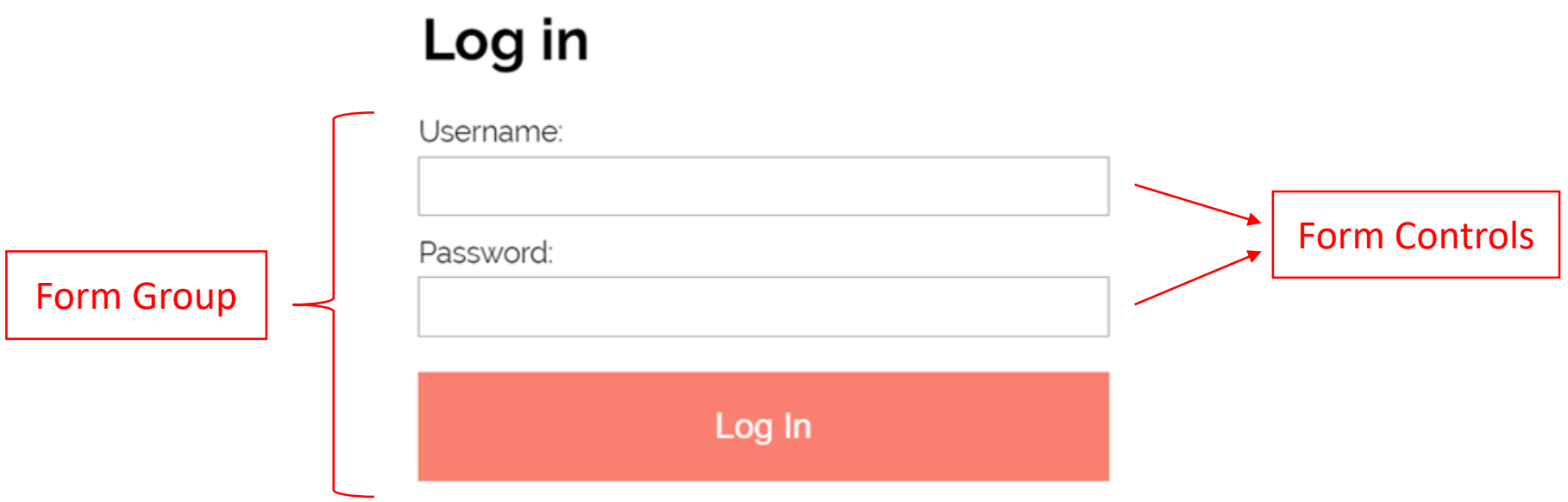
- *For example, if you want to put required field validation in email field, then put the required keyword there, which will make the form invalid until the email is empty. Also, you can disable the button if the form is invalid.*

```
<h3>Template Driven Form</h3>

<form #loginForm="ngForm" (ngSubmit)="login(loginForm.value)">

  <input type="text" required name="email" placeholder="Email" ngModel>
  <br><br>

  <input type="password" name="password" placeholder="Password" ngModel>
  <br><br>

  <button [disabled]="loginForm.invalid">Login</button>

</form>
```

Q79

What is Form Group and Form Control in Angular?

- Form controls are the **individual** elements.
- Form groups is a **collection** of form controls.



Q79

What is Form Group and Form Control in Angular?

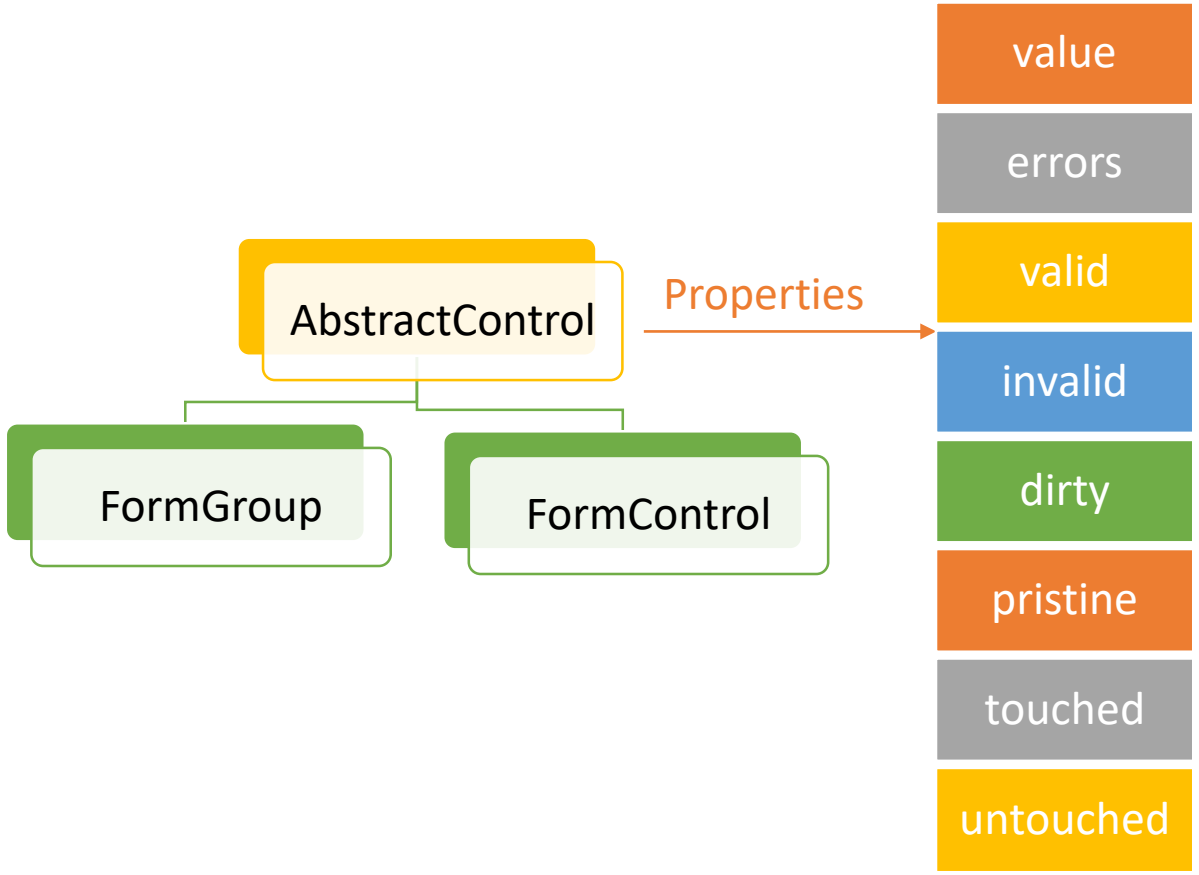
❖ What is the use of FormGroup and FormControl?

Both FormGroup and FormControl are used to **track** the values and states of controls.

In image you can see, both formgroup and formcontrol are inherited from AbstractControl class. AbstractControl class has these properties - Value, errors, valid, invalid, dirty, pristing, touched and touched.

Now these properties will help in **tracking the changes in each and every control**. For example, value property will be the value of any input field.

That's why FormGroup and FormControl are required to keep track of the changes in the controls.



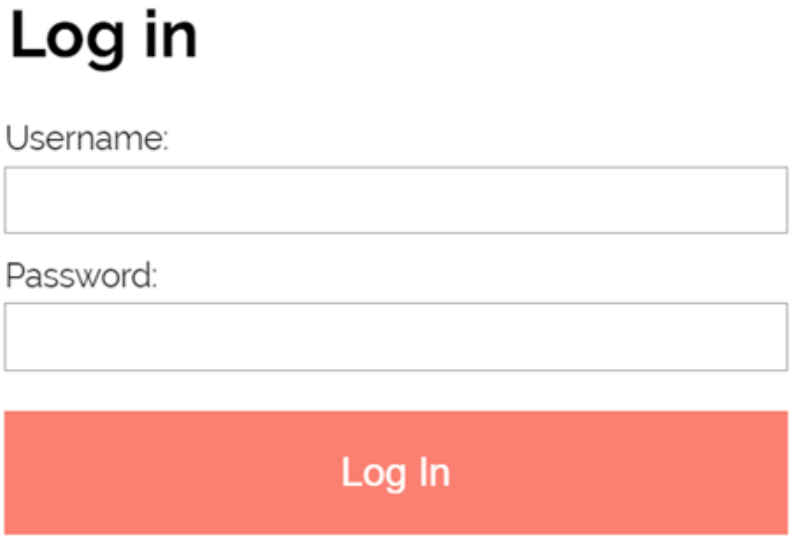
Q80

How to setup Reactive Forms? V. IMP.

- Below are some important points about Reactive Forms:

1. In reactive forms most code and validation logic is written in **component typescript** file.

2. Reactive Forms can be activated by adding **ReactiveFormsModule** in AppModule.



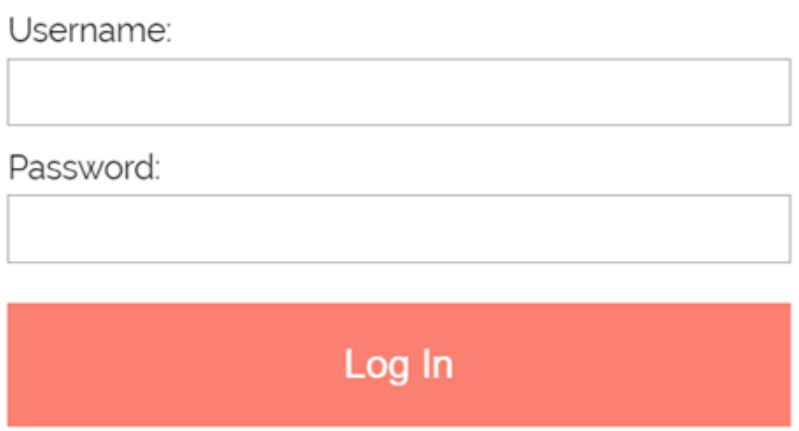
- When to use Reactive forms and when to use template driven forms?

Reactive Forms are used when application is **complex** and have more number of controls.
Template driven forms is used when number of controls are very less in the form.
Normally we prefer reactive forms in general.

Q80

How to setup Reactive Forms? V. IMP.

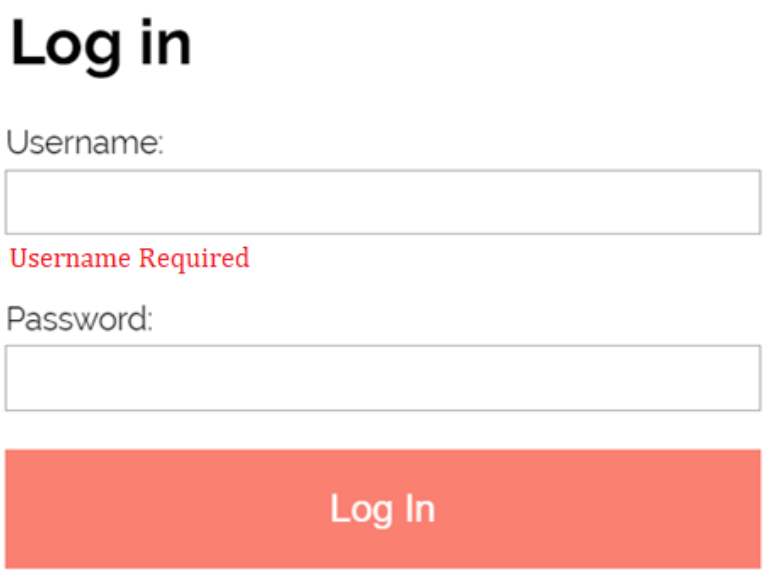
```
export class AppComponent{

  loginForm = new FormGroup({
    username: new FormControl(null, Validators.required),
    password: new FormControl()
  });

  submitLogin(): void{
    console.log(this.loginForm);
  }

  validateUsername()
  {
    return this.loginForm.get('username');
  }
}
```



Q81

How to do validation in reactive forms?

```
export class AppComponent{

  loginForm = new FormGroup({
    username: new FormControl(null, Validators.required),
    password: new FormControl()
  });

  submitLogin(): void{
    console.log(this.loginForm);
  }

  validateUsername()
  {
    return this.loginForm.get('username');
  }
}
```



TOP 100 ANGULAR INTERVIEW QUESTIONS

Chapter 13: Authentication/ JWT/ Auth Gurad/ HTTP Interceptor

Q82. What is Authentication & Authorization in Angular?

Q83. What is JWT Token Authentication in Angular?

Q84. How to Mock or Fake an API for JWT Authentication?

Q85. How to implement the Authentication with JWT in Angular?

Q86. What is Auth Guard?

Q87. What is HTTP Interceptor?

Q88. How to Retry automatically if there is an error response from API?

Q89. What are the parts of JWT Token?

Q90. What is Postman?

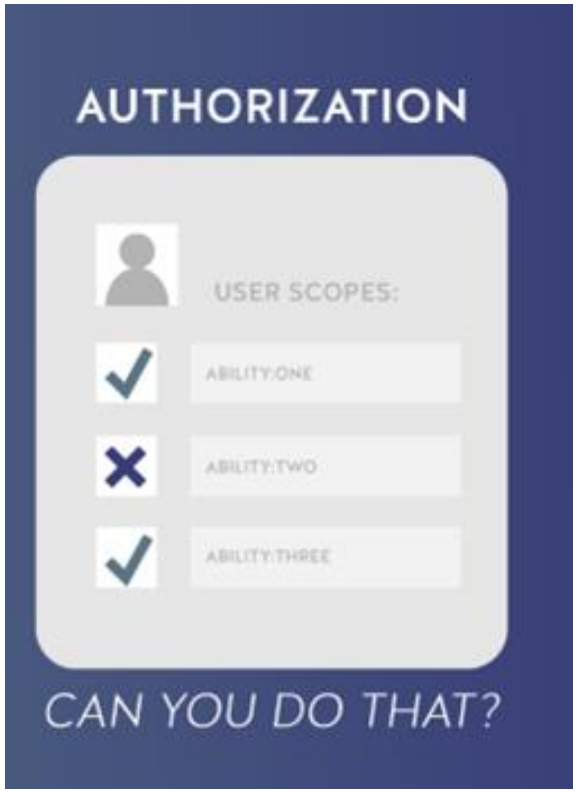
Q91. Which part of the request has the token stored when sending to API?

Q82

What is Authentication & Authorization in Angular?

- Authentication is the process to verify the **user's identity**.

- Authorization is the process of allowing a user to **access** the resources as per the role.



- Authentication is always done before Authorization.

Q83

What is Token(JWT) Authentication Process in Angular? V. IMP.

- JWT authentication is a process to authenticate Angular user credentials to get or post data from API's.

Upcoming is the image, where we will see, how the request will start from Angular and what are the steps to authenticate the request and get the data back from the API.

Q83

What is Token(JWT) Authentication Process in Angular? V. IMP.

Front-end/ Client-side

Middleware/ Server-side/ Backend

1. POST: {username, password}

2. Authenticate & create JWT **Token**

3. Return Response {JWT token}

4. Store JWT token at local storage

5. Request Data {JWT token: Header}

6. Validate token signature

7. Send Data

Q84

How to Mock or Fake an API for JWT Authentication?

Angular developer need not to develop the API. This question is better explained in the video.

Q85

How to implement the Authentication with JWT in Angular? V. IMP.

❖ Steps to implement JWT Authentication in Angular:

1. Create a reactive login form, setup email and password.
2. Create one SubmitLogin method and check whether login form is valid or not.
3. If valid, call service(AuthService) method(proceedLogin) and send the loginForm values and use subscribe method to get the result from the observable.
4. Store the token in local storage of browser. So that for next subsequent request, only token is sent, not the credentials again.

```
export class LoginComponent{

  constructor(private service: AuthService, private route: Router) {
  }

  public loginForm = new FormGroup({
    email: new FormControl(),
    password: new FormControl()
  });

  SubmitLogin(): void{
    if(this.loginForm.valid)
    {
      this.service.proceedLogin(this.loginForm.value).subscribe(result => {
        if(result != null)
        {

          localStorage.setItem('token', result.access_token);

          this.route.navigate(['home']);
        }
      });
    }
  }
}
```

Go to Line/Column

Q85

How to implement the Authentication with JWT in Angular? V. IMP.

5. Create Service(AuthService).
6. Inside service set API URL(apiurl)
7. Setup HTTPClient to send the request using API url.
8. Use Observalbe to receive the data or toke from the request and return it to the component.

- Why we created service and have not directly written this code in component?

So that all the components can use this reusable service method.

```
export class AuthService {

  apirul = 'http://localhost:8000/auth/login';

  constructor(private http: HttpClient) { }

  proceedLogin(user: any): Observable<any>
  {
    return this.http.post(this.apirul, user);
  }

  IsLoggedIn(){
    return localStorage.getItem('token') != null;
  }

  GetToken(){
    return localStorage.getItem('token');
  }

}
```

Q85

How to implement the Authentication with JWT in Angular? V. IMP.

- Here are the Angular features we have used to implement Authentication in Angular:

1. Services
2. Routing
3. Reactive Forms
4. HttpClientModule
5. Dependency Injection
6. Observable & Subscriber

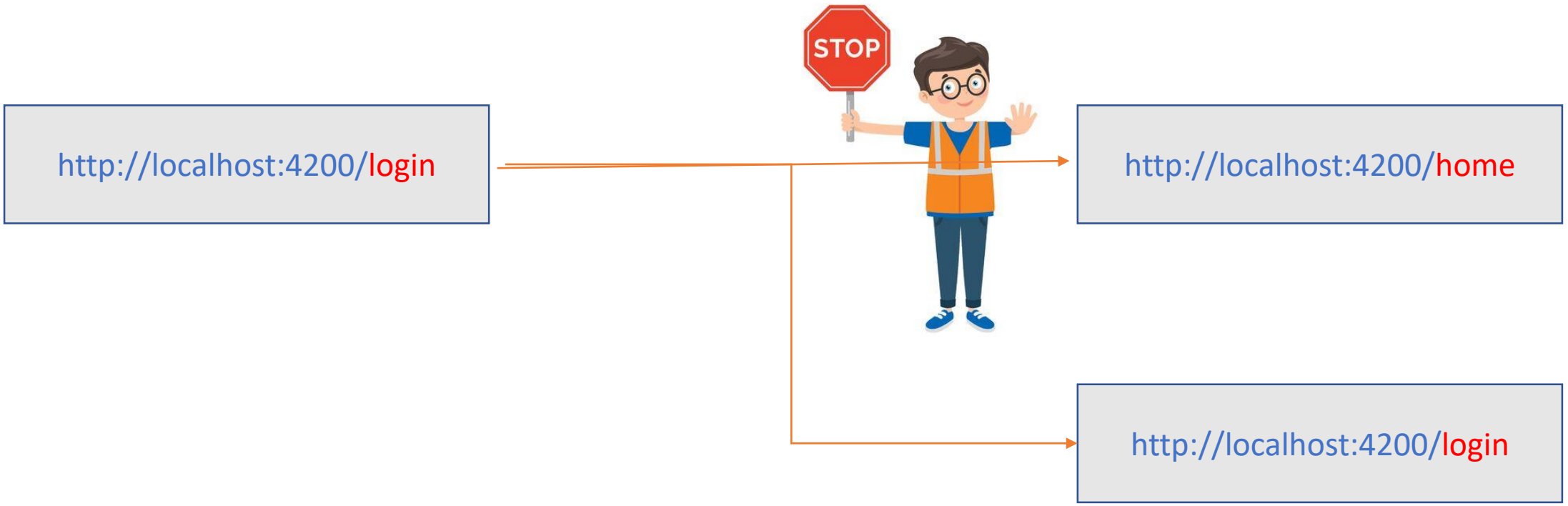
Q86

What is Auth Guard or Route Guard? When to use it? V. IMP.

- Auth guard is used for guarding the routes.

For example, in the image, suppose you are at this login url. Now without entering the username and password, by just changing the url manually, you want to go to the home component. This is not right. So, to prevent this action we use auth guards.

This guard will redirect your request to some other component which you will set in the guard or to the login component only.

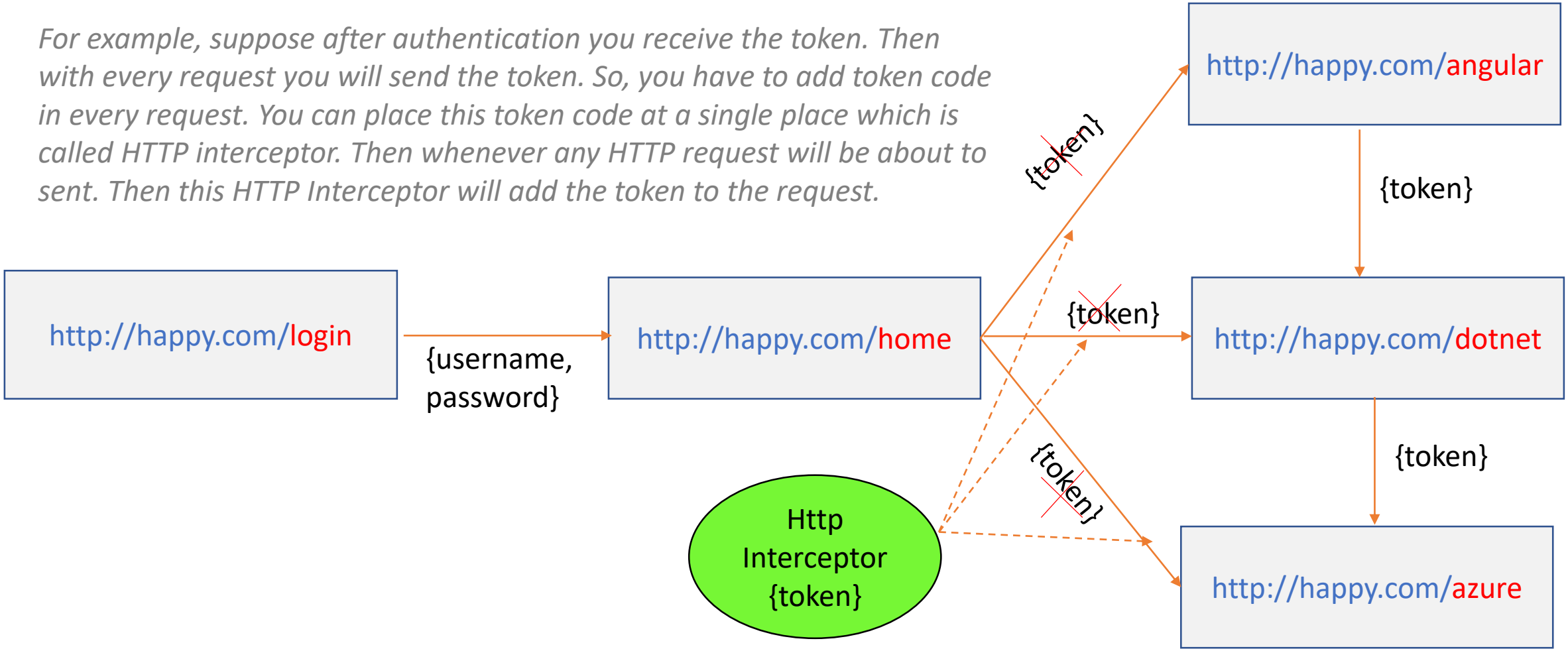


Q87

What is HTTP Interceptor? When & why to use it? V. IMP.

- Interceptor is a special Angular Service, that **can be used to intercept all the request and response calls and modify them to our requirement**.

For example, suppose after authentication you receive the token. Then with every request you will send the token. So, you have to add token code in every request. You can place this token code at a single place which is called HTTP interceptor. Then whenever any HTTP request will be about to sent. Then this HTTP Interceptor will add the token to the request.



Q87

What is HTTP Interceptor? When & why to use it? V. IMP.

```
@Injectable()
export class TokenInterceptor implements HttpInterceptor {

  constructor(private service: AuthService) {}

  intercept(request: HttpRequest<unknown>, next: HttpHandler): Observable<HttpEvent<unknown>> {
    //return next.handle(request);
    let token = this.service.GetToken();

    let requestWithToken = request.clone({
      setHeaders: {
        Authorization: 'Bearer' + token
      }
    });

    return next.handle(requestWithToken);
  }
}
```

Q88

How to Retry automatically if there is an error response from API?

Suppose your angular is requesting something from the API. But the API is down or there is some network issue. Then you want your Angular App to retry again automatically.

- The answer is we will use Retry operator of RxJS library.

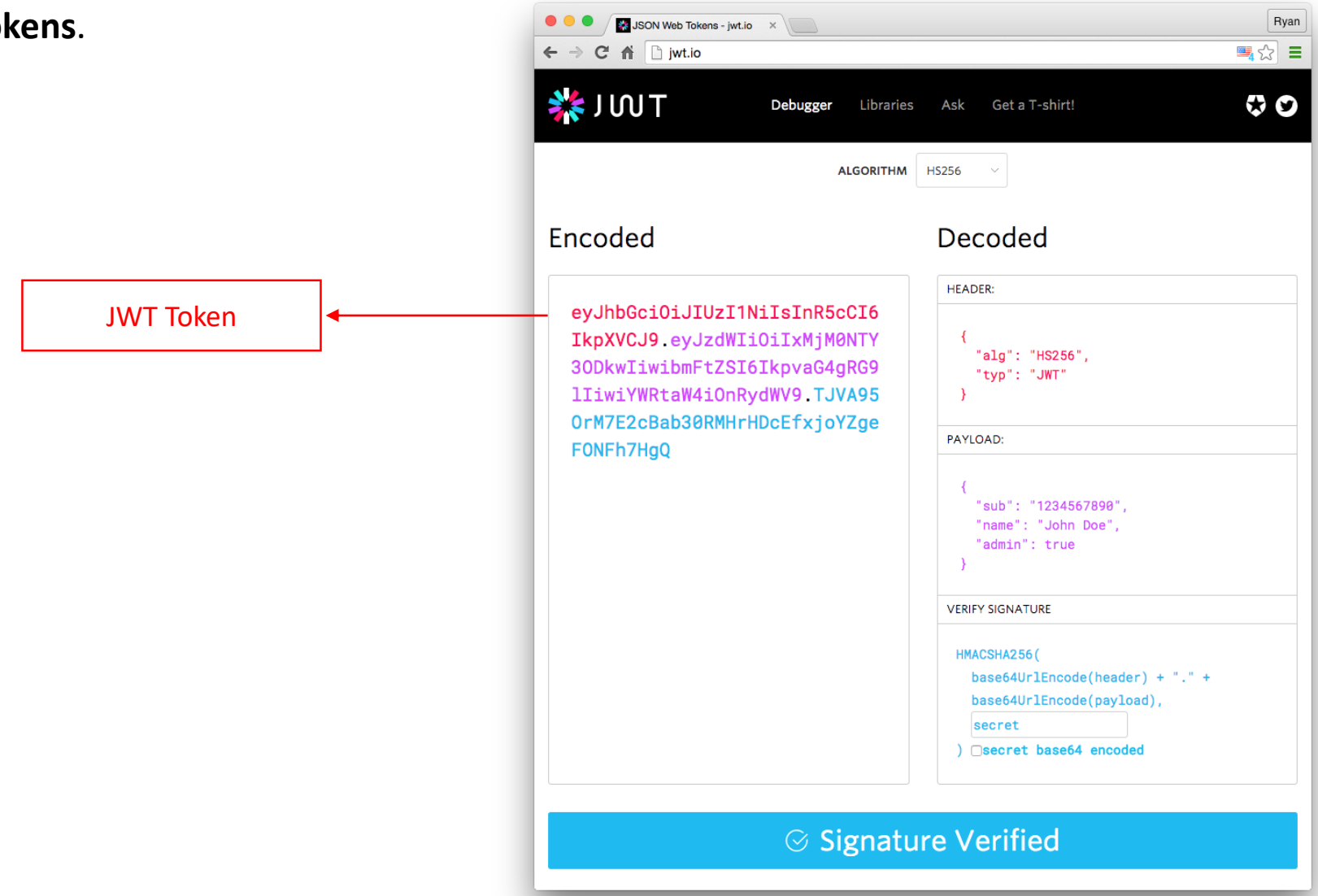
In image, inside the interceptor if connection fail one time, then Angular will automatically retry 3 more times to make the request.

```
return next.handle(requestWithToken).pipe(
  retry(3)
);
```

Q89 What are the parts of JWT Token?

- JWT stands for **JSON Web Tokens**.
  It has 3 parts:

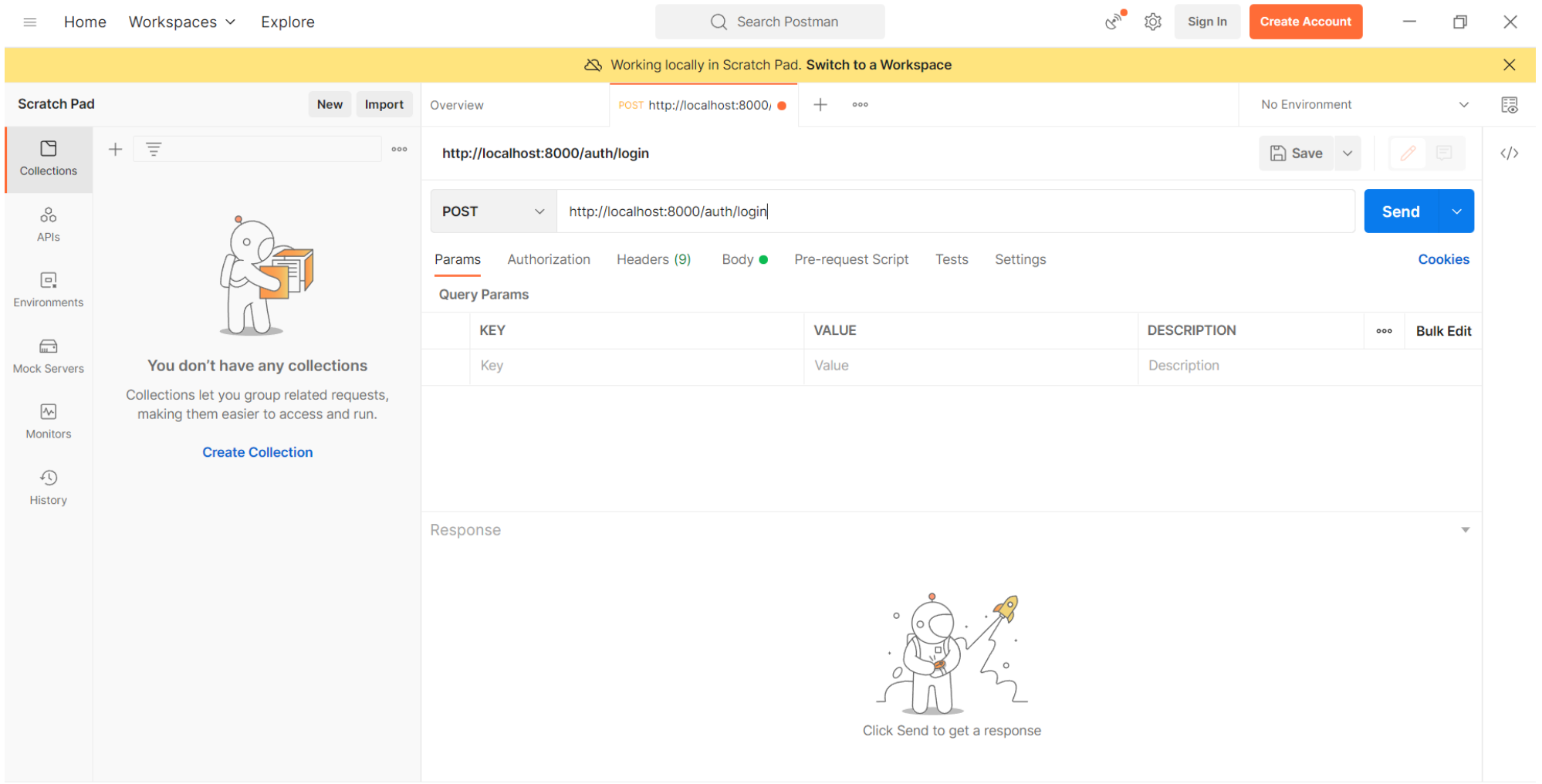
1. Header
2. Payload
3. Signature



Q90

What is Postman?

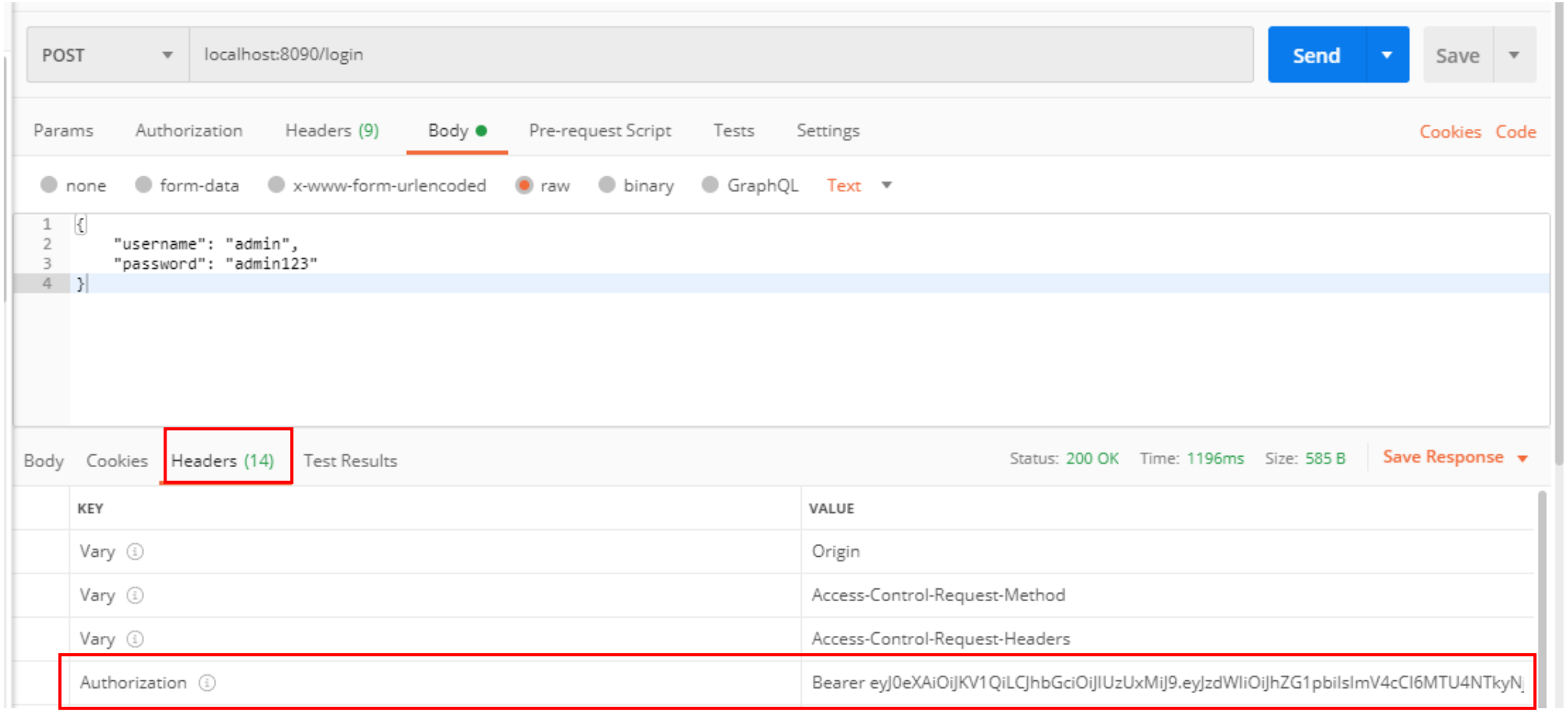
- Postman is a tool for **testing** API's.



Q91

Which part of the request has the token stored when sending to API?

- Token is sent in the HEADER part of the request.



TOP 100 ANGULAR INTERVIEW QUESTIONS

Chapter 14: Components Communication

Q92. What are the various ways to communicate between the components?

Q93. What is ContentProjection? What is <ng-content>?

Q94. What is Template Reference Variable in Angular?

Q95. What is the role of ViewChild in Angular?

Q96. How to access the child component from parent component with ViewChild?

Q97. What is the difference between ViewChild and ViewChildren? What is QueryList?

Q98. What is ContentChild?

Q99. What is the difference between ContentChild & ContentChildren?

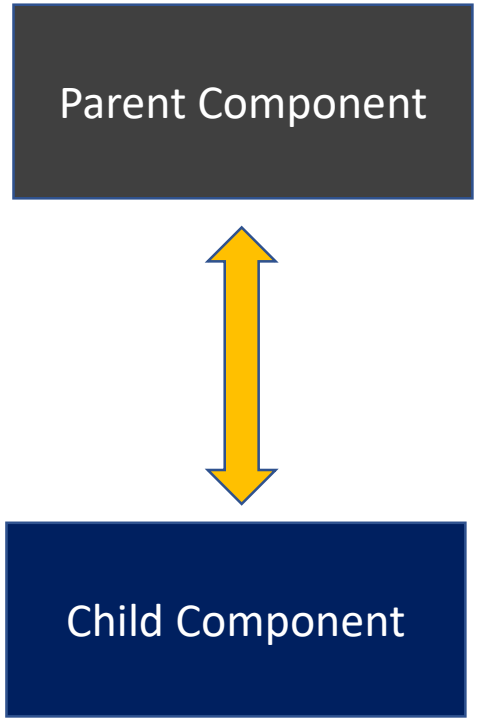
Q100. Compare ng-Content, ViewChild, ViewChildren, ContentChild & ContentrChildren?

Q92

What are the various ways to communicate between the components?

- When we have multiple components in a single page application. Then many a times we have to send and receive data and content(html) from one component to another component. This is called communication between components.

- We also call these communicating components as parent and child components.



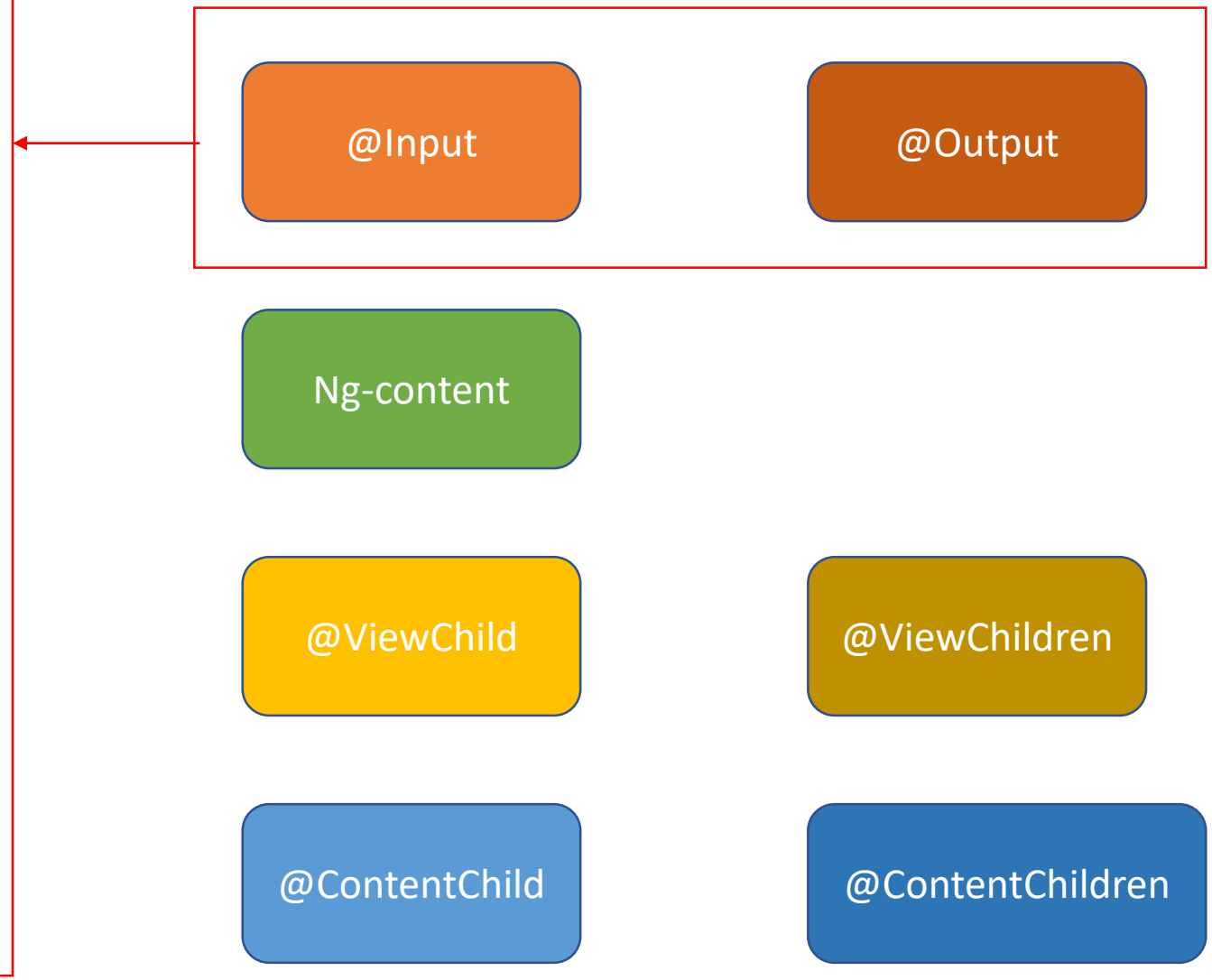
Q92

What are the various ways to communicate between the components?

- We already discussed @Input and @Output decorators in previous questions.
- @Input decorator is used to transfer data from parent to child component.
- @Output decorator is used to transfer data from child to parent component.

In upcoming questions we will discuss rest of the ways to transfer content between components.

*Remember @Input and @Output is used to transfer **data** and rest of decorators are used to transfer content(html).*



Q93 What is Content-Projection? What is <ng-content>?

- ng-content is used to insert the dynamic/variable html content from one component to another component.

For example, in the first image we have the <app-sport> selector of the child component in parent component html. And in the second image we have the html of child component app-sport. When the page will be displayed then <ng-content> of title class in child component will be replaced by Football dynamically.

Football

This is one of the top sports.
It is very healthy to play.
Hurreeey...

Play Football

```
<app-sport>
  <h4 class="title">Football</h4>
  <h4 class="footer">Play Football</h4>
</app-sport>
```

```
<div class="card" style="background-color: lightblue;">

    <ng-content select=".title"></ng-content>

    <p>This is one of the top sports.<br>
        It is very healthy to play.<br>
        Hurreeey...<br>
    </p>

    <ng-content select=".footer"></ng-content>

</div>
```

Q94

What is Template Reference Variable in Angular?

- Template Reference Variable in angular is used to access all the properties of any element inside DOM.

For example, in mage #inputBox is the template reference variable of that input element. Then the getInput(inputBox.value) method is used to transfer the value property of the #inputBox. That's how we can send different properties of template reference variable to component class file.

Then in next image, we are receiving the value property in the parameter and printing it via console.log.

```
<h2>Template Reference Variable</h2>

<input type="text" #inputBox>

<br><br>
<button (click)="getInput(inputBox.value)">Click</button>
```

```
export class AppComponent {

  getInput(value: any)
  {
    console.log(value);
  }

}
```

Q95 What is the role of ViewChild in Angular? V. IMP.

- @ViewChild decorator is used to access elements of html template file from the component typescript file.

For example, in image you can see emailRef is the template reference variable in html file and it is accessed in component class via @ViewChild decorator.

app.component.html M

src > app > app.component.html > ...

Go to component

```
1  <br><br>
2
3  <input #emailRef type="text">
4  <br><br>
5
6  <input type ="password">
7  <br><br>
8
9  <button>Login</button>
10
```

app.component.ts M

rc > app > TS app.component.ts > ...

```
2
3   @Component({
4     selector: 'app-root',
5     templateUrl: './app.component.html',
6     styleUrls: ['./app.component.css']
7   })
8   export class AppComponent {
9
10    title = 'View-Child';
11
12    @ViewChild('emailRef') emailElementRef: ElementRef | undefined;
13
14  }
```

Q95

What is the role of ViewChild in Angular? V. IMP.

- When to use @viewchild?

1. When we want to access the element of html component file in the component class file of same component.

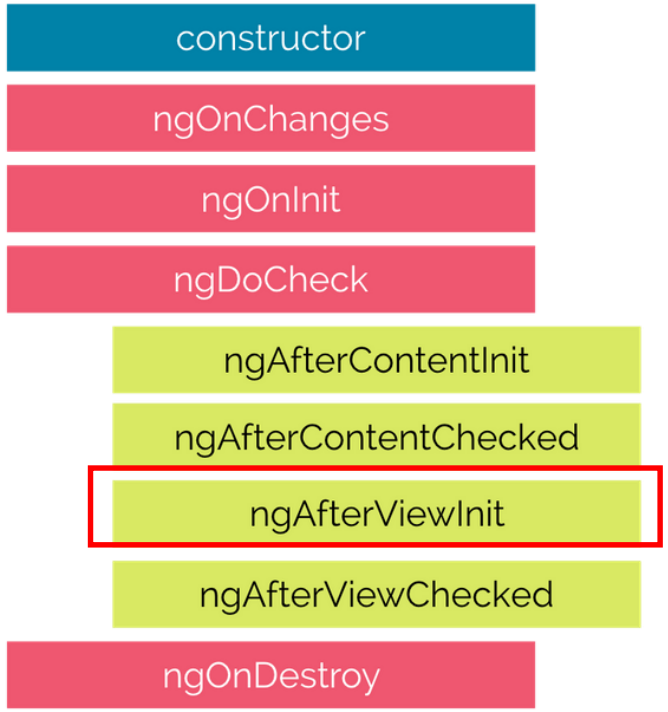
2. When we want to access the element of child html component file in the parent component class file.

Q96

How to access the child component from parent component with ViewChild?

- To access the child component from parent component, we have to write the code inside the **ngAfterViewInit** lifecycle hook of the parent component class file.

Because if we will write the code in ngOnInit or any other lifecycle hook before ngAfterViewInit, then may be the child component is still not fully developed till then and this may lead to error.




```
export class AppComponent  implements AfterViewInit {

  @ViewChild(ChildComponent, {static:false}) childComponentProp: ChildComponent | undefined;

  ngAfterViewInit(): void {

    console.log(this.childComponentProp?.SendToParent());

  }

}
```

Q97

What is the difference between ViewChild and ViewChildren? What is QueryList? V. IMP.

- @ViewChildren does the same thing as @ViewChild decorator, but @ViewChild can access only one element, but @ViewChildren can access a **list of elements**.

```
<input #emailRef type="text">
<input #emailRef type="text">
<input #emailRef type="text">
<br><br>
```

```
export class AppComponent  implements AfterViewInit {

  title = 'View-Child';

  @ViewChildren('emailRef') emailElementRefChildren: QueryList<ElementRef> | undefined;

  ngAfterViewInit(): void {

    this.emailElementRefChildren?.forEach((item)=>item.nativeElement.value = 100);
  }

}
```

Q97

What is the difference between ViewChild and ViewChildren? What is QueryList? V. IMP.

- QueryList used to manage the **list of elements**.
- It is used with @ViewChildren to refer the list of elements of type ElementRef.

```
export class AppComponent  implements AfterViewInit {

  title = 'View-Child';

  @ViewChildren('emailRef') emailElementRefChildren: QueryList<ElementRef> | undefined;

  ngAfterViewInit(): void {

    this.emailElementRefChildren?.forEach((item)=>item.nativeElement.value = 100);
  }

}
```

Q98

What is ContentChild?

- ContentChild is used to send variable or projected content from one component html to another component class file.

- To access the element in child component, we have to write the code inside the **ngAfterContentInit** lifecycle hook of the child component class file.

```
<app-child>

    <h3 #refTitle>Football</h3>

</app-child>
```

```
@ContentChild('refTitle') title: ElementRef | undefined;

ngAfterContentInit(): void {

  console.log(this.title?.nativeElement.innerText);

}
```

Q99

What is the difference between ContentChild & ContentChildren?

- @ContentChildren does the same thing as @ContentChild decorator, but @ContentChild can access only one element, but @ContentChildren can access a **list of elements**.

```
<app-child>
    <h3 #refTitle>Football</h3>

    <h4 #refTitle>Running</h4>

    <h6 #refTitle>Cricket</h6>
</app-child>
```

```
@ContentChildren('refTitle') titleChildren: QueryList<ElementRef> | undefined;

ngAfterContentInit(): void {

  //For each loop to iterate all the elements in list.
  this.titleChildren?.forEach((item)=> console.log(item.nativeElement.innerText));
}
```

Q100

Compare ng-Content, ViewChild, ViewChildren, ContentChild & ContentrChildren? V. IMP.

	<ng-content> (Content Projection)	@ViewChild	@ViewChildren	@ContentChild	@ContentChildren
1. Use Case	Used to display **variable content** from child component html to parent component html.	Used to send content from: 1. Same component html template -> Component class. 2. Child component -> Parent Component class.	Used same as @ViewChild, but in addition it can handle **multiple elements**.	Used to send **projected content** from component HTML template file -> Another component class file.	Used same as @ContentChild, but in addition it can handle **multiple elements**.
2. Example/ Declaration	*<ng-content select=".title"></ng-content>*	*@ViewChild('emailRef') emailElement: ElementRef*	*@ViewChildren('emailRef') emailElement: QueryList<ElementRef>*	*@ContentChild('emailRef') emailElement: ElementRef*	*@ContentChildren('emaiRef') emailElement: QueryList<ElementRef>*
3. Template reference variable	No	Yes	Yes	Yes	Yes
4. Type	No type, since it's a html to html content transfer.	**ElementRef**. (It's a wrapper around an element)	**QueryList<ElementRef>**. (It's a list of ElementRef)	ElementRef.	QueryList<ElementRef>