REVATURE

Angular Client Interview Questions and Answers

1. Demonstrate a basic understanding of Angular or What is Angular

- Angular is a typescript-based web application framework used to create & build web apps
- It allows us to create Single Page Application (SPA)
- **Gmail, Youtube, PayPal** apps are developed using Angular

2. What is meant by SPA?

- It is a single web page, website, or web application that works within a web browser and loads just a single document.

- It does not need page reloading during its usage, and most of its content remains the same while only some of it needs updating.
- **Gmail, Facebook, Trello, Google Maps**, etc., all are Single Page Applications that offer an outstanding user experience in the browser with no page reloading.

3. Angular workflow or How does Angular work or bootstrapping your angular app?

- Flow: angular.json -> main.ts -> AppModule -> AppComponent -> index.html
- Every Angular app consists of a file named angular.json. This file will contain all the configurations of the app. While building the app, the builder looks at this file to find the entry point of the application.

```
"build": {
  "builder": "@angular-devkit/build-angular:browser",
  "options": {
    "outputPath": "dist/angular-starter",
    "index": "src/index.html",
    "main": "src/main.ts",
    "polyfills": "src/polyfills.ts",
    "tsConfig": "tsconfig.app.json",
    "aot": false,
    "assets": [
      "src/favicon.ico",
      "src/assets"
    ],
    "styles": [
      "./node_modules/@angular/material/prebuilt-themes/deeppurple-amber.css",
      "src/style.css"
    ]
  }
}
```

- Inside the build section, the main property of the options object defines the entry point of the application which in this case is main.ts.
- **main.ts is the entry point of the angular application.**
- The main.ts file creates a browser environment for the application to run, and, along with this, it also calls a function called bootstrapModule, which bootstraps the application. These two steps are performed in the following order inside the main.ts file:

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';


platformBrowserDynamic().bootstrapModule(AppModule)
```

  - In the above line of code, AppModule is getting bootstrapped.
  - The AppModule is declared in the app.module.ts file. This module contains declarations of all the components.
  - Below is an example of app.module.ts file:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';

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

- As one can see in the above file, **AppComponent** is getting bootstrapped.
- This component is defined in **app.component.ts** file. This file interacts with the webpage and serves data to it.
- Below is an example of **app.component.ts** file:

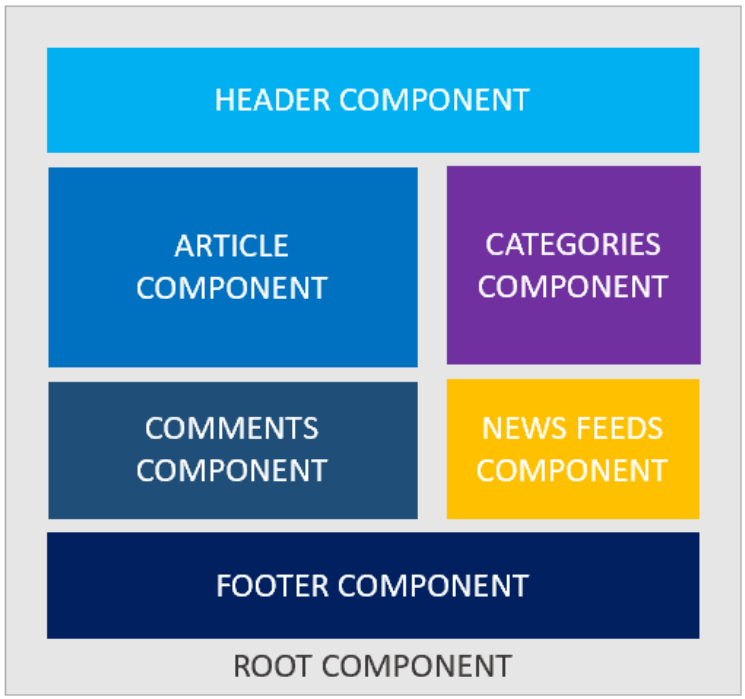
```
import { Component } from '@angular/core';

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

- After this, Angular calls the **index.html** file. This file consequently calls the root component that is **app-root**.
- This is how the **index.html** file looks:

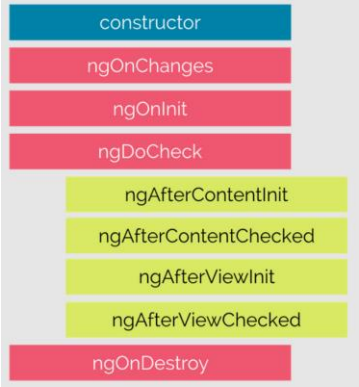
```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Angular</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

- The HTML template of the root component is displayed inside the **<app-root>** tags.
- This is how every angular application works. Or This is how angular application get bootstrapped

4. **What is a component in angular?**
   - Components are the basic building blocks in the Angular application. Components contain the data & UI logic that defines the view and behavior of the web application.

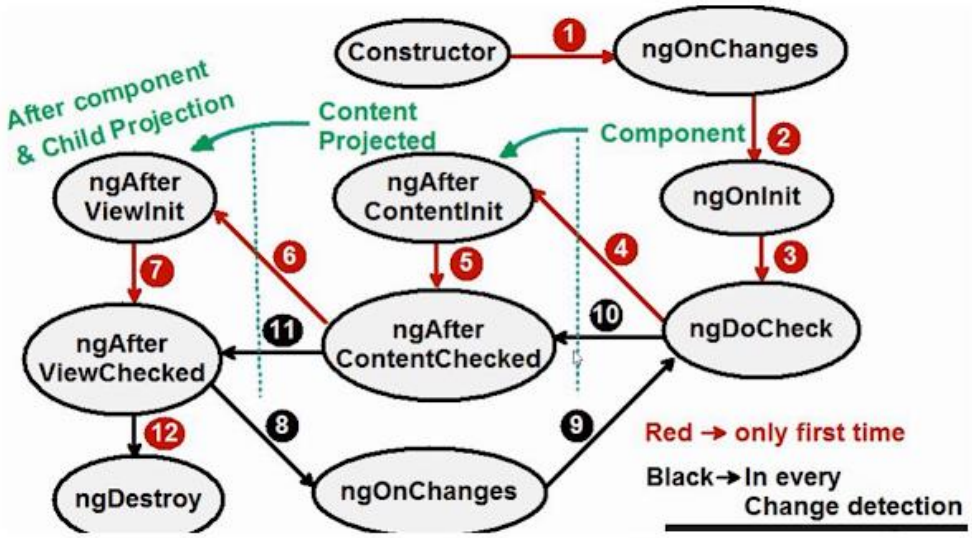


5. **Angular Components Lifecycle or Lifecycle Hooks or LifeCycle Methods**
   - Angular creates a component; renders it; creates and renders its children; checks it when it's data-bound properties change; and destroys it before removing it from the DOM. These events are called **"Lifecycle Hooks"**.
   - Lifecycle Hooks:



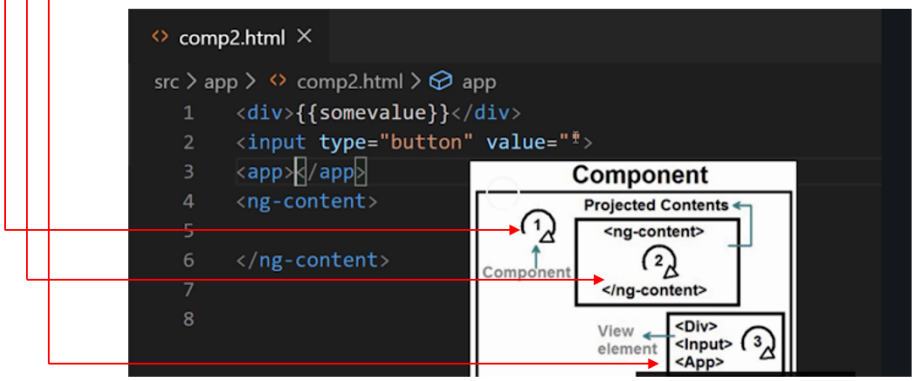
   - `constructor()` - The constructor of the component class gets executed first, before the execution of any other lifecycle hook events. If we need to inject any dependencies into the component, then the constructor is the best place to do so.

- `ngOnChanges()` - Called whenever the input properties of the component change. It returns a *SimpleChanges* object which holds any current and previous property values.
- `ngOnInit()` - Called once to initialize the component and set the input properties. It initializes the component after Angular first displays the data-bound properties.
- `ngDoCheck()` - Called during all change-detection runs that Angular can't detect on its own. Also called immediately after the `ngOnChanges()` method.
- `ngAfterContentInit()` - Invoked once after Angular performs any content projection into the component's view.
- `ngAfterContentChecked()` - Invoked after each time Angular checks for content projected into the component. It's called after `ngAfterContentInit()` and every subsequent `ngDoCheck()`
- `ngAfterViewInit()` - Invoked after Angular initializes the component's views and its child views.
- `ngAfterViewChecked()` - Invoked after each time Angular checks for the content projected into the component. It called after `ngAfterViewInit()` and every subsequent `ngAfterContentChecked()`
- `ngOnDestroy()` - Invoked before Angular destroys the directive or component.

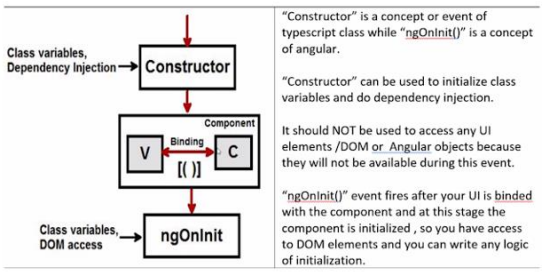


Angular Lifecycle Hooks

- *These are the event which runs when for the first time angular component loaded. Now let's look into the events which fire every time on every change detection.*
- *For example someone goes and type something in the text box or someone goes and click on the button then below are the sequence of events which gets fired, **ngOnChanges() -> ngDoCheck() -> ngAfterContentChecked() -> ngAfterViewChecked()***
- *For understanding there are 3 event sequence,*
  1. *First one is kick start event **ngOnchanges(), ngOnit(), ngDoCheck()***
  2. *Second one is events for projected contents **ngAfterContentInit(), ngAfterContentChecked()***
  3. *Finally third one is View component events **ngAfterViewInit(). ngAfterViewChecked()***



Constructor Vs ngOnInit



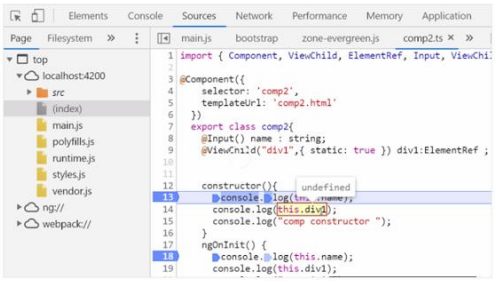
"Constructor" is a concept or event of typescript class while "ngOnInit()" is a concept of angular.

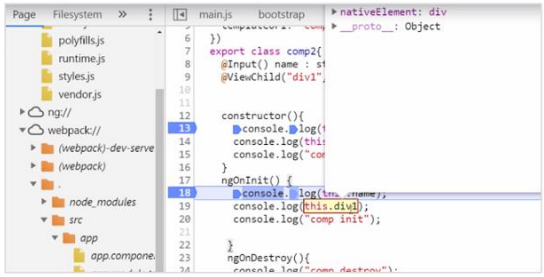
"Constructor" can be used to initialize class variables and do dependency injection.

It should NOT be used to access any UI elements /DOM or Angular objects because they will not be available during this event.

"ngOnInit()" event fires after your UI is binded with the component and at this stage the component is initialized , so you have access to DOM elements and you can write any logic of initialization.

	Constructor	ngOnInit
Concept of	It's a concept of Typescript. It's called by JavaScript framework.	It's a concept of Angular. It's an angular event. It's called by Angular framework.
Binding state	Binding has not happened till this moment so only class variables are accessible.	Binding with the UI has been done so the class variables have values set from the UI.
DOM	Component is not initialized. So, DOM is not accessible.	Component is initialized, DOM is ready and accessible.
Initialization logic	Initialization of class Variables and dependency injection logic can be put here.	All type of Initialization logics can be written here as right from class to component is initialized.





6. What are some advantages of Angular?

- Effective cross platform development
- Two-way data binding in Angular will help users to exchange data from the component to view and from view to the component. It will help users to establish communication bi-directionally.

- The Angular command-line interface (CLI) makes the developer's job easier because it offers a set of helpful tools for coding.
- Angular offers powerful DI (dependency injection) instrument and services to resolve various productivity issues and speed up the development process.
- Modularity of angular application makes our code readable and testable

7. What are the different types of directives in Angular?

- **Component Directives** - Component directives alter the details of how the component should be processed, instantiated, and used at runtime.
- **Structural Directives** Structural directives are used for adding, removing, or manipulating DOM elements.
- **Attribute Directives** - Attribute directives are used to change the look and behavior of the DOM elements.

Custom Directive: Custom directive can also be created if any of the above directives does not solve our purpose for the requirement

8. Structural Directives in Angular?

- Structural directives are used for adding, removing, or manipulating DOM elements
- Structural directives start with an asterisk (*) followed by a directive name.
- There are three built-in structural directives - `ngIf`, `ngFor` and `ngSwitch`.
- The `ngFor` **directive** is used to repeat a part of the HTML template once per each item from an iterable list.
- `ngIf` **directive** allows us to add or remove DOM Elements based upon the Boolean expression. We can also have an else block associated with an ngIf directive.

```
<div *ngIf="age > 55; else elseBlock1">
    {{name}} is a senior citizen
</div>
<ng-template #elseBlock1>
    {{name}} is not a senior citizen
</ng-template>
```

- `ngSwitch` **directive** lets you hide/show HTML elements depending on an expression. `NgSwitchCase` displays its element when its value matches the switch value. `NgSwitchDefault` displays its element when no sibling `NgSwitchCase` matches the switch value.

```
<!-- user to enter any vowels(a, e, i o, u), print any word starting with vowels -->
<input type="text" [(ngModel)]="str" />

```

```
<div [ngSwitch]="str">
    <div *ngSwitchCase="'a'">Entered a!! Word: Apple</div>
    <div *ngSwitchCase="'e'"> Entered e!! Word: Egg</div>
    <div *ngSwitchCase="'i'"> Entered i!! Word: Ice cream</div>
    <div *ngSwitchCase="'o'"> Entered o!! Word: Orange</div>
    <div *ngSwitchCase="'u'"> Entered u!! Word: Umberalla</div>
    <div *ngSwitchDefault> You Entered Constant </div>
</div>
```

9. Attribute Directives in Angular?

- Attribute directives are used to change the look and behavior of the DOM elements.
- Attribute directives are enclosed with the [] square brackets
- There are two built-in attribute directives - **ngClass and ngStyle**
- The **ngClass directive** is used for adding or removing the CSS classes on an HTML element. It allows us to apply CSS classes dynamically based on expression evaluation.

```
<h3 [ngClass]="'red'"> Need your attention</h3>
<div [ngClass]="['red','size20']"> Red Background, Text with Size 20px </div>
<div [ngClass]="{'red':false,'size20':true}">Text with Size 20px</div>
```

- The **ngStyle directive** allows us to dynamically change the style of HTML element based on the expression.

```
Enter the username: <input type='text' [(ngModel)]='name'>
<div [ngStyle]="{'background-color':username === 'Admin' ? 'green' : 'red' }"></div>
```

10. Decorators in Angular?

- Decorators are design patterns or functions that define how Angular features work.
- Angular supports four types of decorators:
  - Class decorators
  - Property decorators
  - Method decorators
  - Parameter decorators

11. Class Decorators in Angular?

- A class decorator tells Angular if a particular class is a component or a module.
- There are various class decorators in Angular, and among them, **@Component** and **@NgModule** are widely used.

12. @Component Decorator

- In app.component.ts file, we export the AppComponent class, and we decorate it with the @Component decorator, imported from the @angular/core package, which takes a few metadata, such as: selector, templateUrl **and** styleUrls**.**

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

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

- selector – just name given for the component. In the index.html file, <app-root> tag corresponds to component's selector. By doing so, Angular will inject the corresponding template of the component.

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

- templateUrl - points to an HTML file that defines what you see on your application.
- styleUrls - points to set of CSS file that defines styles or design for application

13. @NgModule **Decorator**

- @NgModule takes the below metadata to launch the application:
  - declarations — contains a list of components, directives, and pipes, which belong to this module.
  - imports — contains a list of modules, which are used by the component templates in this module reference. For example, we import BrowserModule to have browser-specific services such as DOM rendering, sanitization, and location.
  - providers — the list of service providers that the application needs.
  - bootstrap — contains the root component of the application

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

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

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

14. How to consume API using Angular?

- We are required to import and setup `HttpClient` service in Angular project to consume REST APIs.
- To work with *HttpClient* service in Angular, you need to import the `HttpClientModule` in `app.module.ts` file.
- Then inject `HttpClient` service in constructor method after that you can hit the remote server via HTTP's POST, GET, PUT and DELETE methods.

15. How do you do routing?

- First, we need to run `ng new routing-app –routing` command to create an angular application with routing module
- Make sure `AppRoutingModule` is in the `imports` of `@NgModule` in the `app.module.ts` file
- Add routes in the `routing.module.ts` file

```
1.  import { NgModule } from '@angular/core';
2.  import { RouterModule, Routes } from '@angular/router';
3.  import { LoginComponent } from './Components/UserComponents/login/login.component';
4.  import { RegisterComponent } from
    './Components/UserComponents/register/register.component';
5.  const routes: Routes = [
6.    {path : 'login', component: LoginComponent },
7.    {path : 'register', component: RegisterComponent}
8.  ];
9.
10. @NgModule({
```

```
11.   imports: [RouterModule.forRoot(routes)],
12.   exports: [RouterModule]
13. })
14. export class AppRoutingModule { }
```

- In your `app.component.html` file, we add our routes to the application

```
1.  <h1> Routing Demo </h1>
2.  <nav>
3.      <li><a routerLink="/login">Login</a></li>
4.      <li><a routerLink="/register">register</a></li>
5.  </nav>
6.  <router-outlet></router-outlet>
```

- Here,
  - `routerLink` - is an attribute to an anchor tag which sets the route for the component.
  - `<router-outlet>` - works as a placeholder to load the different components dynamically based on the activated component.

16. How do you handle dependency injection in angular?

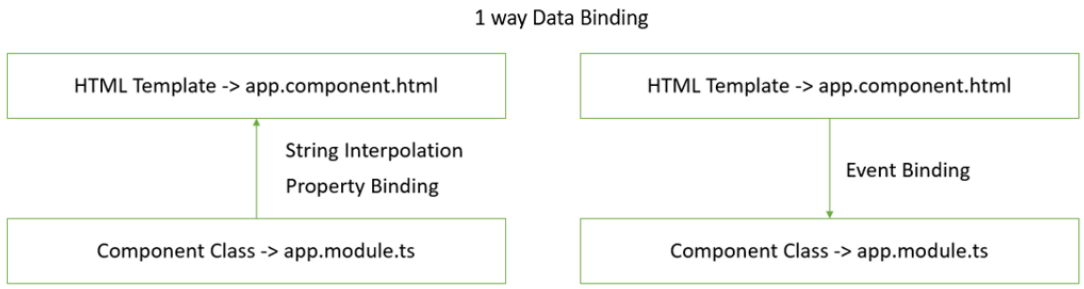
- In Angular, dependencies are typically services.
- The `@Injectable()` decorator marks a class as a service class that can be injected.
- The `@Injectable()` decorator has a `providedIn` property where we specify the provider of the decorated service class.
- By default, providedIn property has values 'root', that means services is injected to the AppModule.

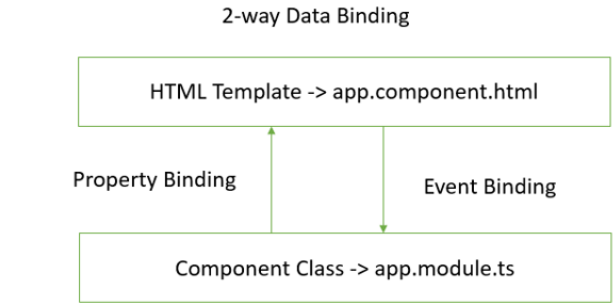
```
1   import { Injectable } from '@angular/core';
2
3   @Injectable({
4     providedIn: 'root'
5   })
6   export class UserService {
7
8     constructor() { }
9   }
```

- Here we are injecting to UserService to the `AppModule`, so all the components able to use this service.

17. What are the ways of databinding in angular?

- Databinding is a technique used to bind the data from an HTML template to a component class (.ts file) or from a component class (.ts file) to an HTML template.
- They are 1 way databinding and 2-way databinding





One-way and Two-way Data Binding in Angular

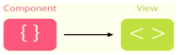
- *One-way and two-way data binding are two of the important ways by which we can exchange data from component to DOM and vice-versa. Data exchange between the component and the view will help us to build dynamic and interactive web applications.*
- *One-way data binding will bind the data from the component to the view (DOM) or from view to the component. One-way data binding is unidirectional. You can only bind the data from component to the view or from view to the component.*

One-way Data Binding

❑ ***1. From Component to View***

There are different techniques of data binding which use one-way data binding to bind data from component to view. Below are some of the techniques, which uses one-way data binding.

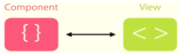
1. ***Interpolation Binding**: Interpolation refers to binding expressions into marked up language.*
2. ***Property Binding**: Property binding is used to set a property of a view element. The binding sets the property to the value of a template expression.*
3. ***Attribute Binding**: Attribute binding is used to set a attribute property of a view element.*
4. ***Class Binding**: Class binding is used to set a class property of a view element.*
5. ***Style Binding**: Style binding is used to set a style of a view element.*



❑ ***2. From View to Component***

1. ***Event Binding**: One-way data binding from view to the component can be achieved by using the event binding technique.*

Two-way Data Binding



❑ ***Both From Component to View & From View to Component***

1. ***[(ngModel)]="value":** Two-way data binding allows you to exchange the data from DOM to component and vice-versa. For two-way data binding, Angular uses the banana in a box syntax(). Banana box syntax is basically the combination of property binding and event binding.*

18. Property Binding

- From Component Class to the HTML Template
- Bind values to the attributes of HTML elements.
- Uses [], square brackets in the html file
- Create a variable in the class, and the bind that value to an attribute for HTML tag

In app.component.html file,

```
<img [src]=imgsrc />
```

Property Binding

In app.component.ts file, inside AppComponent class

```
imgsrc = "https://www.sattvagroup.in/wp-content/uploads/2018/05/Image-Tower.jpg"
```

19. Event Binding

- From HTML template to the component class
- Bind DOM events such as keystrokes, button clicks, mouse overs, touches, etc. to a function in the component.
- Uses (), parentheses in the html file
- Here, we were calling the **OnClick()** function, when the 'Click Here' button is clicked.

In app.component.html file,

```
<button (click)="OnClick()"> Click Here</button>
```

Event Binding

In app.component.ts file, inside AppComponent class

```
OnClick(): void{
  alert("button clicked!!!")
  console.log(this.msg)
}
```

20. String Interpolation

- From the component class to the HTML template
- Uses {{}}, double curly braces in the html

In app.component.html file,

```
<h3>{{title}} is running!!!</h3>
```

String Interpolation

In app.component.ts file, inside AppComponent class

```
title = 'FIRSTANGULARPROJECT';
```

21. What is two-way databinding in angular

- Two-way data binding is achieved by combining **property binding** and **event binding** together.
- Mostly used in forms.
- The Angular uses the **ngModel** directive to achieve two-way binding on HTML **<form>** elements.
- To use the **ngModel** directive, we need to import the **FormsModule** package into our Angular module.
- Here, we enclose **ngModel** directive within [()]

In app.component.html file,

```
<input type="text" placeholder="Enter any message" [(ngModel)]="msg" />
```

Property Binding

Event Binding

In app.component.ts file, inside AppComponent class

```
msg ='';
```

22. Difference between Angular JS and Angular 4 +

Angular JS	Angular 4
Uses MVC architecture to build the applications.	Uses component-based UI to build the applications.

AngularJS is written in JavaScript.	Angular is compatible with the most recent versions of TypeScript that have powerful type checking and object-oriented features.
To bind an image/property or an event with AngularJS, you have to remember the right ng directive.	Angular focuses on "()" for event binding and "[ ]" for property binding.
AngularJS doesn't support mobiles.	Angular support mobiles.

23. Difference between Angular 2 and Angular 4

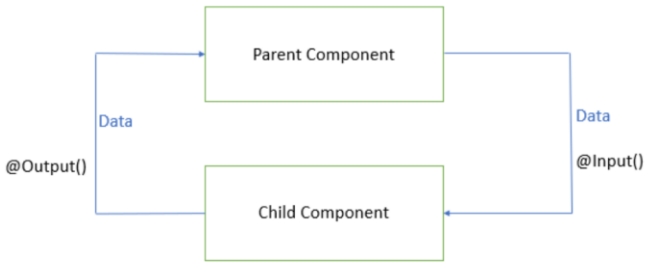
Angular 2	Angular 4
Angular v2.0 uses Typescript, superset of JavaScript, for writing the application.	Angular v4.0 serves to be compatible with the new version of TypeScript 2.1 as well TypeScript 2.2.
Code is not Reduced much	Reduce the size of the generated bundled code up to 60%

24. How would you protect routes?

- Routing guards used to protect the routes.
- Routing guards used to check whether the user should grant or remove access to certain parts of the navigation.
- There are 4 different interfaces act as routing guards:
  - **CanActivate** - decides if the route can be activated.
  - **CanActivateChild** - decides if children of a route can be activated.
  - **CanLoad** - decides if a route can be loaded.
  - **CanDeactivate** - decides if the user can leave a route.

25. How would you pass data from a parent to a child component or a child to a parent component?

- **@Input** decorator used to pass the data from a parent to a child component
- **@Output** decorator used to pass the data from a child to a parent component



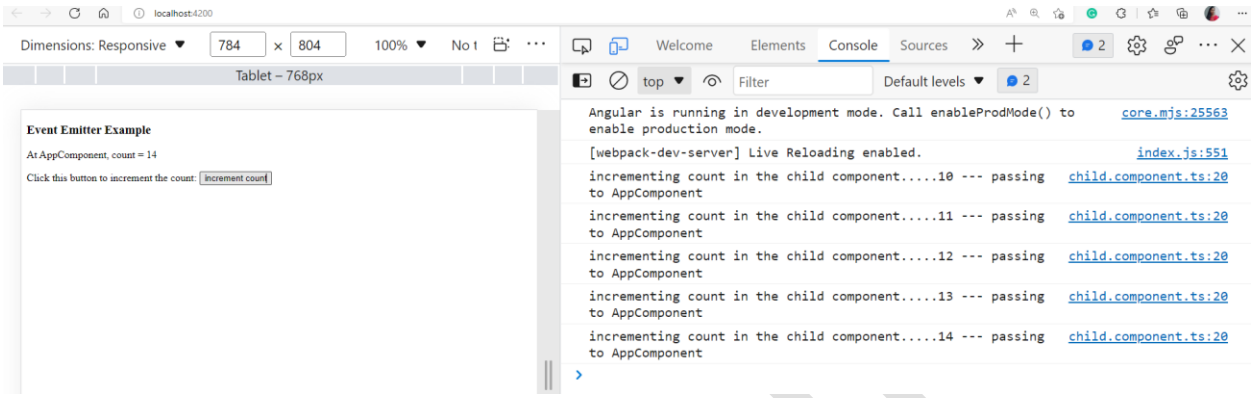
- Consider we have AppComponent as Parent. Let's create a child component using ng g c child command. We'll pass the data from AppComponent to ChildComponent and vice versa.
- In child. component.ts, we create a *change* property and decorate with the @Output() and bound a new instance of EventEmitter to it.
- Also, we have a method - increment() which updates the value of the count property based on the event (clicking on the increment count button) and emits the event changes to its parent component (AppComponent).
- Here, the change property calls the emit() method that emits the count value which can be received by event object $event.

```
1.  import { Component, EventEmitter, Input, Output } from '@angular/core';
2.  @Component({
3.    selector: 'app-child',
4.    template: `
5.      <p> Click this button to increment the count:
6.      <button (click)='increment()'>increment count</button> </p>
7.  `
8.  })
9.  export class ChildComponent  {
10.
11.   @Input()
12.   count: number = 0;
13.
14.   @Output()
15.   change: EventEmitter<number> = new EventEmitter<number>();
16.
17.   increment() {
18.     this.count++;
19.     this.change.emit(this.count);
20.     console.log("incrementing count in the child component....." + this.count + " ---
    passing to AppComponent");
21.   }
22. }
```

- In app.component.ts, we use event binding to get the ***count*** property value from the ChildComponent to the AppComponent

```
1.  import { Component } from '@angular/core';
2.
3.  @Component({
4.    selector: 'app-root',
5.    template: `
6.   <h3> Event Emitter Example </h3>
7.   <p> At AppComponent, count = {{ count }} </p>
8.      <app-child [count]='count' (change)= 'countChange($event)'></app-child>
9.  `
10.
11. })
12. export class AppComponent {
```

```
  count = 9;
  countChange(event: number) {
    this.count = event;
  }
}
```



26. What are some common Angular CLI commands?

- `ng new MyApp` – used to create an angular application named 'MyApp'
- `ng new MyApp --routing` - used to create an angular application named 'MyApp' with the routing module
- `ng g c first` – used to create component named 'first'
- `ng g p changePipe` – used to create pipe named `changePipe'
- `ng g s user` - used to create service named 'user'
- `ng serve` – used to build, run and launch application on HTTP port 4200
- `ng serve -o` - used to build, run and launch application on HTTP port 4200, -o option automatically opens the browser to http://localhost:4200

27. How components communicate with each other in Angular?

- By passing data between from a child to a parent or a parent to a child component, we can use `@Input` and `@Output`.
- By passing data through a service using observables

28. What are Services in angular?

- Services are used to organize and share business logic, models, data, or functions with different components of an Angular application.

```
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { user } from './user';

```

```
@Injectable({
  providedIn: 'root'
})
export class UserserviceService {
        baseurl = 'http://localhost:3000/users';
        constructor(private http: HttpClient) { }

        GetAllUsers() :Observable<user[]>{
          return this.http.get<user[]>(this.baseurl);
        }}
```

29. What are Pipes in angular?

- A pipe takes in data as input and transforms it to the desired output.
- In *app.component.html*, we have built in pipes and custom pipes.
- **Some of the built-in pipes are**:
    - **Date pipe** - Used for formatting dates.
    - **Decimal pipe** - Used for formatting numbers
    - **Currency pipe** - Used for formatting currencies
    - **Lowercase pipe** - Used for converting strings into lowercase.
    - **Uppercase pipe** - Used for converting strings into uppercase.

```
<h2>Built-in Pipes</h2>
<li>{{"Pipes"}} </li>
<li>{{"Pipes" | uppercase}}</li>
<li>{{"Pipes" | lowercase}} </li>
<li>{{dob}}</li>
<li>{{dob | date}}</li>
<li>{{dob | date |uppercase }}</li>
<li>{{17.81922 | number }}</li>
<li>{{17.819227546354 | number: '3.4-6' }}</li>
<li>{{17.81922 | number : '2.0-0'}}</li>
<li>{{365778 | currency}}</li>
<li>{{365778 | currency: 'INR'}}</li>
<h2>Custom Pipes</h2>
<li>{{"Pipes" |firstChar}}</li>
<li>{{"Angular" |firstChar}}</li>
<li>{{"great" |firstChar}}</li>
```

- We can create custom pipes using the `ng g pipe <pipe-name>` command in the terminal with the Angular CLI.
- **For example**, we create a custom pipe to count words by running *the* `ng g pipe firstChar` command in the terminal. The CLI creates 2 files - `firstChar.pipe.spec.ts` and `firstChar.pipe.ts` under `src/app` folder and updates the `app.module.ts` file.
- In `firstChar.pipe.ts`,

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

@Pipe({
  name: 'firstChar'
})
export class FirstCharPipe implements PipeTransform {
  transform(value: string): string {
    return value[0];
  }}
```

- Output:

Built-in Pipes

- Pipes
- PIPES
- pipes
- Fri Jan 15 1999 00:00:00 GMT+0530 (India Standard Time)
- Jan 15, 1999
- JAN 15, 1999
- 17.819
- 017.819228
- 18
- $365,778.00
- ₹365,778.00

Custom Pipes

- P
- A
- g

30. What is the difference between a promise and an observable?

- A Promise emits a single value while Observable can emit multiple values.
- So, while handling a HTTP request, a Promise can manage a single response for the same request, but if there are multiple responses to the same request, then we have to use an Observable.

```
const promise = new Promise((data) =>{
    data(1);
    data(2);
    data(3);    }).then(element => console.log('Promise '+ element));
// Logs:
// Promise 1

const observable = new Observable((data) => {
    data.next(1);
    data.next(2);
    data.next(3);   }).subscribe(element => console.log('Observable ' + element));

// Logs:
//Observable 1
//Observable 2
//Observable 3
```