ANGULAR INTERVIEW QUESTIONS

1. What is Angular? Why was it introduced?

-> Angular was introduced to create Single Page applications. This framework brings structure and consistency to web applications and provides excellent scalability and maintainability.Angular is an open-source, JavaScript framework wholly written in TypeScript. It uses HTML's syntax to express your application's components clearly.

2. What is TypeScript?

-> TypeScript is a superset of JavaScript that offers excellent consistency. It is highly recommended, as it provides some syntactic sugar and makes the code base more comfortable to understand and maintain. Ultimately, TypeScript code compiles down to JavaScript that can run efficiently in any environment.

3. What is data binding? Which type of data binding does Angular deploy?

-> Data binding is a phenomenon that allows any internet user to manipulate Web page elements using a Web browser.

It uses dynamic HTML and does not require complex scripting or programming.

We use data binding in web pages that contain interactive components such as forms, calculators, tutorials, and games.

Incremental display of a webpage makes data binding convenient when pages have an enormous amount of data.

Angular uses the two-way binding.

Any changes made to the user interface are reflected in the corresponding model state.

Conversely, any changes in the model state are reflected in the UI state.

This allows the framework to connect the DOM to the Model data via the controller.

However, this approach affects performance since every change in the DOM has to be tracked.

4. What are Single Page Applications (SPA)?

-> Single-page applications are web applications that load once with new features just being mere additions to the user interface. It does not load new HTML pages to display the new page's content, instead generated dynamically. This is made possible through JavaScript's ability to manipulate the DOM lements on the existing page itself. A SPA approach is faster, thus providing a seamless user experience.

6. What are decorators in Angular?

-> Decorators are a design pattern or functions that define how Angular features work. They are used to make prior modifications to a class, service, or filter. Angular supports four types of decorators, they are

1.Class Decorators

2.Property Decorators

3.Method Decorators

4.Parameter Decorators

7. Mention some advantages of Angular.

-> Some of the common advantages of Angular are –

- **MVC architecture** - Angular is a full-fledged MVC framework. It provides a firm opinion on how the application should be structured. It also offers bi-directional data flow and updates the real DOM.
- **Modules** - Angular consists of different design patterns like components, directives, pipes, and services, which help in the smooth creation of applications.
- **Dependency injection** - Components dependent on other components can be easily worked around using his feature.
- Other generic advantages include clean and maintainable code, unit testing, reusable components, data binding, and excellent responsive experience.

9. What are Templates in Angular?

-> Angular Templates are written with HTML that contains Angular-specific elements and attributes. In combination with the model and controller's information, these templates are further rendered to provide a dynamic view to the user.

10. What are Annotations in Angular?

-> Annotations in Angular are used for creating an annotation array. They are the metadata set on the class that is used to reflect the Metadata library.

11. What are Directives in Angular?

-> Directives are attributes that allow the user to write new HTML syntax specific to their applications. They execute whenever the Angular compiler finds them in the DOM. Angular supports three types of directives.

1.Component Directives

2.Structural Directives

3.Attribute Directives

12. What is an AOT compilation? What are its advantages?

-> The Ahead-of-time (AOT) compiler converts the Angular HTML and TypeScript code into JavaScript code during the build phase, i.e., before the browser downloads and runs the code.

Some of its advantages are as follows :

1.Faster rendering

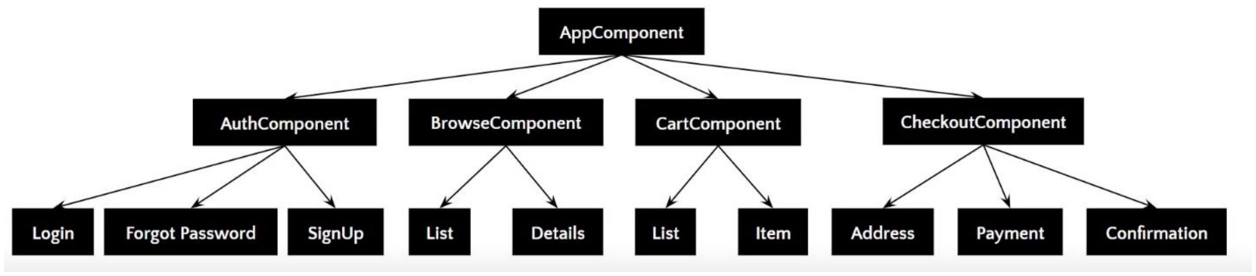
2.Fewer asynchronous requests

3.Smaller Angular framework download size

4.Quick detection of template errors

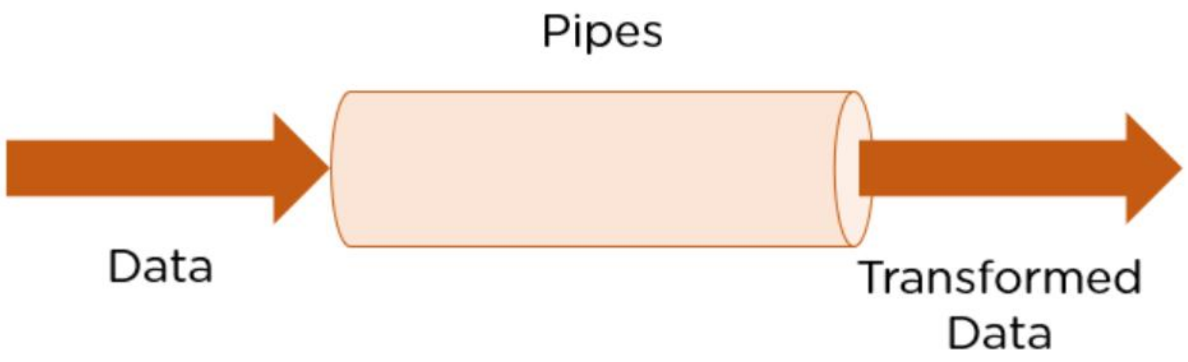
5.Better security

13. What are Components in Angular?



Components are the basic building blocks of the user interface in an Angular application. Every component is associated with a template and is a subset of directives. An Angular application typically consists of a root component, which is the AppComponent, that then branches out into other components creating a hierarchy.

14. What are Pipes in Angular?



->Pipes are simple functions designed to accept an input value, process, and return as an output, a transformed value in a more technical understanding. Angular supports several built-in pipes. However, you can also create custom pipes that cater to your needs.

Some key features include:

1.Pipes are defined using the pipe "|" symbol.

2.Pipes can be chained with other pipes.

3.Pipes can be provided with arguments by using the colon (:) sign.

15. What is the PipeTransform interface?

-> As the name suggests, the interface receives an input value and transforms it into the desired format with a transform() method. It is typically used to implement custom pipes.

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

 @Pipe({

  name: 'demopipe'

})
```

```
export class DemopipePipe implements PipeTransform {

  transform(value: unknown, ...args: unknown[]): unknown {

    return null;

  }

}
```

16. What are Pure Pipes?

-> These pipes are pipes that use pure functions. As a result of this, a pure pipe doesn't use any internal state, and the output remains the same as long as the parameters passed stay the same. Angular calls the pipe only when it detects a change in the parameters being passed. A single instance of the pure pipe is used throughout all components.

17. What are Impure Pipes?

-> For every change detection cycle in Angular, an impure pipe is called regardless of the change in the input fields. Multiple pipe instances are created for these pipes. Inputs passed to these pipes can be mutable. By default, all pipes are pure. However, you can specify impure pipes using the pure property, as shown below.

```
@Pipe({

  name: 'demopipe',

  pure : true/false

})

export class DemopipePipe implements PipeTransform {
```

18. What is an ngModule?

-> NgModules are containers that reserve a block of code to an application domain or a workflow. @NgModule takes a metadata object that generally describes the way to compile the template of a component and to generate an injector at runtime. In addition, it identifies the module's components, directives, and pipes, making some of them public, through the export property so that external components can use them.

19. What are filters in Angular? Name a few of them.

-> Filters are used to format an expression and present it to the user. They can be used in view templates, controllers, or services. Some inbuilt filters are as follows :

- date - Format a date to a specified format.
- filter - Select a subset of items from an array.
- Json - Format an object to a JSON string.
- limitTo - Limits an array/string, into a specified number of elements/characters.
- lowercase - Format a string to lowercase.

20. What is view encapsulation in Angular?

-> View encapsulation defines whether the template and styles defined within the component can affect the whole application or vice versa. Angular provides three encapsulation strategies:

- Emulated - styles from the main HTML propagate to the component.
- Native - styles from the main HTML do not propagate to the component.
- None - styles from the component propagate back to the main HTML and therefore are visible to all components on the page.

21. What are controllers?

-> AngularJS controllers control the data of AngularJS applications. They are regular JavaScript Objects. The ng-controller directive defines the application controller.

22. What do you understand by scope in Angular?

-> The scope in Angular binds the HTML, i.e., the view, and the JavaScript, i.e., the controller. It as expected is an object with the available methods and properties. The scope is available for both the view and the controller. When you make a controller in Angular, you pass the $scope object as an argument.

23. Explain the lifecycle hooks in Angular

-> In Angular, every component has a lifecycle. Angular creates and renders these components and also destroys them before removing them from the DOM. This is achieved with the help of lifecycle hooks. Here's the list of them -

- ngOnChanges() - Responds when Angular sets/resets data-bound input properties.
- ngOnInit() - Initialize the directive/component after Angular first displays the data-bound properties and sets the directive/component's input properties/
- ngDoCheck() - Detect and act upon changes that Angular can't or won't detect on its own.
- ngAfterContentInit() - Responds after Angular projects external content into the component's view.
- ngAfterContentChecked() - Respond after Angular checks the content projected into the component.
- ngAfterViewInit() - Respond after Angular initializes the component's views and child views.
- ngAfterViewChecked() - Respond after Angular checks the component's views and child views.
- ngOnDestroy - Cleanup just before Angular destroys the directive/component.

24. What is String Interpolation in Angular?

-> String Interpolation is a one-way data-binding technique that outputs the data from TypeScript code to HTML view. It is denoted using double curly braces. This template expression helps display the data from the component to the view.

{{ data }}

25. What are Template statements?

-> Template statements are properties or methods used in HTML for responding to user events. With these template statements, the application that you create or are working on, can have the capability to engage users through actions such as submitting forms and displaying dynamic content.

For example,

<button (click)="deleteHero()">Delete hero</button>

The template here is deleteHero. The method is called when the user clicks on the button.

Angular Interview Questions For Advanced Level

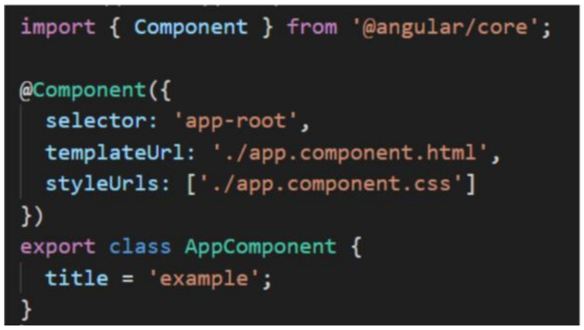
26. What is the difference between AOT and JIT?

-> Ahead of Time (AOT) compilation converts your code during the build time before the browser downloads and runs that code. This ensures faster rendering to the browser. To specify AOT compilation, include the --aot option with the ng build or ng serve command.

The Just-in-Time (JIT) compilation process is a way of compiling computer code to machine code during execution or run time. It is also known as dynamic compilation. JIT compilation is the default when you run the ng build or ng serve CLI commands.

27. Explain the @Component Decorator.

-> TypeScript class is one that is used to create components. This genre of class is then decorated with the "@Component" decorator. The decorato's purpose is to accept a metadata object that provides relevant information about the component.



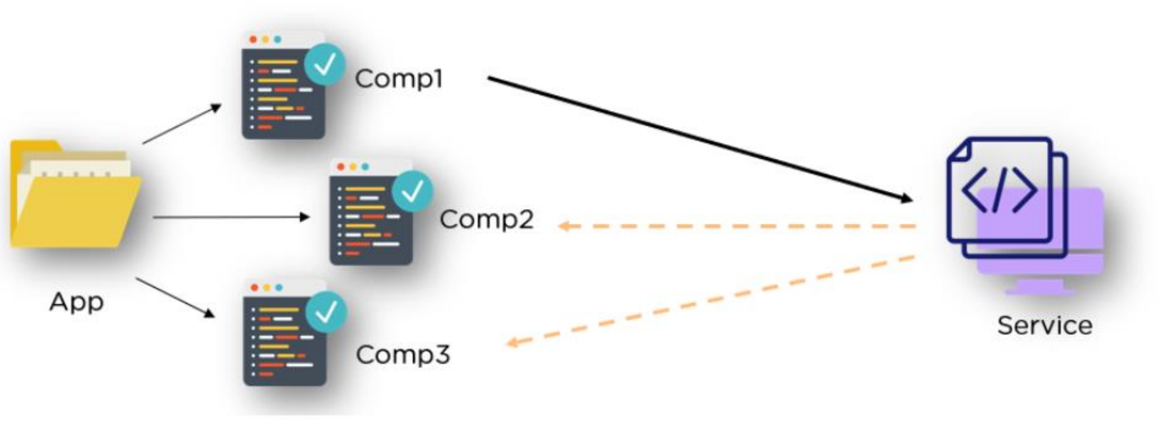
```
import { Component } from '@angular/core';

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

The image above shows an App component - a pure TypeScript class decorated with the “@Component” decorator. The metadata object that gets accepted by the decorator provides properties like templateUrl, selector, and others, where the templateUrL property points to an HTML file defining what you see on the application.

28. What are Services in Angular?

-> Angular Services perform tasks that are used by multiple components. These tasks could be data and image fetching, network connections, and database management among others. They perform all the operational tasks for the components and avoid rewriting of code. A service can be written once and injected into all the components that use that service.



29. What are Promises and Observables in Angular?

-> While both the concepts deal with Asynchronous events in Angular, Promises handle one such event at a time while observables handle a sequence of events over some time.

Promises - They emit a single value at a time. They execute immediately after creation and are not cancellable. They are Push errors to the child promises.

Observables - They are only executed when subscribed to them using the subscribe() method. They emit multiple values over a period of time. They help perform operations like forEach, filter, and retry, among others. They deliver errors to the subscribers. When the unsubscribe() method is called, the listener stops receiving further values.

30. What is ngOnInit? How is it defined?

-> ngOnInit is a lifecycle hook and a callback method that is run by Angular to indicate that a component has been created. It takes no parameters and returns a void type.

```
export class MyComponent implements OnInit {
  constructor() { }
  ngOnInit(): void {
    //....
  }
}
```

31. How to use ngFor in a tag?

-> The ngFor directive is used to build lists and tables in the HTML templates. In simple terms, this directive is used to iterate over an array or an object and create a template for each element.

```
<ul>
   <li *ngFor = "let items in itemlist"> {{ item }} </li>
  </ul>
```

1. "Let item" creates a local variable that will be available in the template
2. "Of items" indicates that we are iterating over the items iterable.
3. The * before ngFor creates a parent template.

32. What are Template and Reactive forms?

-> Template-driven approach

- In this method, the conventional form tag is used to create forms. Angular automatically interprets and creates a form object representation for the tag.
- Controls can be added to the form using the NGModel tag. Multiple controls can be grouped using the NGControlGroup module.
- A form value can be generated using the "form.value" object. Form data is exported as JSON values when the submit method is called.
- Basic HTML validations can be used to validate the form fields. In the case of custom validations, directives can be used.
- Arguably, this method is the simplest way to create an Angular App.

Reactive Form Approach

- This approach is the programming paradigm oriented around data flows and propagation of change.
- With Reactive forms, the component directly manages the data flows between the form controls and the data models.
- Reactive forms are code-driven, unlike the template-driven approach.
- Reactive forms break from the traditional declarative approach.
- Reactive forms eliminate the anti-pattern of updating the data model via two-way data binding.
- Typically, Reactive form control creation is synchronous and can be unit tested with synchronous programming techniques.

33. What is Bootstrap? How is it embedded into Angular?

-> Bootstrap is a powerful toolkit. It is a collection of HTML, CSS, and JavaScript tools for creating and building responsive web pages and web applications.

There are two ways to embed the bootstrap library into your application.

1. Angular Bootstrap via CDN - Bootstrap CDN is a public Content Delivery Network. It enables you to load the CSS and JavaScript files remotely from its servers.
2. Angular Bootstrap via NPM - Another way to add Bootstrap to your Angular project is to install it into your project folder by using NPM (Node Package Manager).

```
npm install bootstrap
```

```
npm install jquery
```

34. What is Eager and Lazy loading?

-> Eager loading is the default module-loading strategy. Feature modules under Eager loading are loaded before the application starts. This is typically used for small size applications.

Lazy loading dynamically loads the feature modules when there's a demand. This makes the application faster. It is used for bigger applications where all the modules are not required at the start of the application.

35. What type of DOM does Angular implement?

-> DOM (Document Object Model) treats an XML or HTML document as a tree structure in which each node is an object representing a part of the document.

Angular uses the regular DOM. This updates the entire tree structure of HTML tags until it reaches the data to be updated. However, to ensure that the speed and performance are not affected, Angular implements Change Detection.

With this, you have reached the end of the article. We highly recommend brushing up on the core concepts for an interview. It’s always an added advantage to write the code in places necessary.

36. Why were client-side frameworks like Angular introduced?

-> Client-side frameworks like Angular were introduced to provide a more responsive user experience. By using a framework, developers can create web applications that are more interactive and therefore provide a better user experience.

Frameworks like Angular also make it easier for developers to create single-page applications (SPAs). SPAs are web applications that only need to load a single HTML page. This makes them much faster and more responsive than traditional web applications.

Overall, client-side frameworks like Angular were introduced in order to improve the user experience of web applications. By making web applications more responsive and easier to develop, they provide a better experience for both developers and users.

37. How does an Angular application work?

-> An Angular application is a Single Page Application, or SPA. This means that the entire application lives within a single page, and all of the resources (HTML, CSS, JavaScript, etc.) are loaded when the page is first loaded. Angular uses the Model-View-Controller, or MVC, architecture pattern to manage its data and views. The Model is the data that the application uses, the View is what the user sees, and the Controller is responsible for managing communication between the Model and the View.

When a user interacts with an Angular application, the Angular framework will automatically update the View to reflect any changes in the data. This means that Angular applications are very responsive and fast, as the user does not need to wait for the page to reload in order to see updated data.

Angular applications are also very scalable, as they can be divided into small modules that can be loaded independently of each other. This means that an Angular application can be easily extended with new functionality without having to rewrite the entire application.

Overall, Angular applications are very fast, responsive, and scalable. They are easy to develop and extend, and provide a great user experience.

The following is is an example of coding from an angular.json file:

```
 "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"
         ]
       }
     }
```

38. Explain components, modules and services in Angular.

-> Components, modules and services are the three fundamental building blocks in Angular. Components are the smallest, self-contained units in an Angular application. They are typically used to represent a view or UI element, such as a button or a form input field.

Code example:

```
import { Component, OnInit } from '@angular/core';
   @Component({
    selector: 'app-test',
    templateUrl: './test.component.html',
    styleUrls: ['./test.component.css']
   })
   export lass TestComponent implements OnInit {
    constructor() {}
    ngOnInit() {
     }
   }
```

Modules are larger units that group together one or more related components. Services are singleton objects that provide specific functionality throughout an Angular application, such as data access or logging.

Code example:

```
import { BrowserModule } from '@angular/platform-browser';
    import { NgModule } from '@angular/core';
    import { AppComponent } from './app.component';
    import { TestComponent } from './test/text.component';
    @NgModule({
      declarations: [
        AppComponent,
        TestComponent
      ],
      imports: [
        BrowserModule
      ],
      providers: [],
      bootstrap: [AppComponent]
    })
    export class AppModule { }
```

Each component in Angular has its own isolated scope. This means that a component's dependencies (services, other components, etc.) are not accessible to any other component outside of its own scope. This isolation is important for ensuring modularity and flexibility in an Angular application.

Services, on the other hand, are not isolated and can be injected into any other unit in an Angular application (component, module, service, etc.). This makes them ideal for sharing data or functionality across the entire app.

Code example:

```
import { Injectable } from '@angular/core';
    @Injectable({
```

```
  providedIn: 'root'
})
export class TestServiceService {
  constructor() { }
}
```

When designing an Angular application, it is important to keep these three building blocks in mind. Components should be small and self-contained, modules should group together related components, and services should provide shared functionality across the entire app. By following this design principle, you can create an Angular application that is modular, flexible, and easy to maintain.

39. How are Angular expressions different from JavaScript expressions?

-> One major difference between Angular expressions and JavaScript expressions is that Angular expressions are compiled while JavaScript expressions are not. This means that Angular expressions are more efficient since they're already pre-processed. Additionally, Angular expressions can access scope properties while JavaScript expressions cannot. Finally, Angular expressions support some additional features such as filters and directives which aren't available in JavaScript expressions.

Javascript expression example:

```
<!DOCTYPE html>
   <html lang="en">
   <head>
     <meta charset="UTF-8">
     <meta name="viewport" content="width=device-width, initial-scale=1.0">
     <title>JavaScript Test</title>
   </head>
   <body>
     <div id="foo"><div>
   </body>
   <script>
     'use strict';
     let bar = {};
     document.getElementById('foo').innerHTML = bar.x;
```

```
    </script>

    </html>
```

Angular expression example:

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

    @Component({

      selector: 'app-new',

      template: `

          <h4>{{message}}</h4>

       `,

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

    })

    export class NewComponent implements OnInit {

      message:object = {};

      constructor() { }

      ngOnInit() {

      }

    }
```

40. Angular by default, uses client-side rendering for its applications.

-> This means that the Angular application is rendered on the client-side — in the user's web browser. Client-side rendering has a number of advantages, including improved performance and better security. However, there are some drawbacks to using client-side rendering, as well. One of the biggest drawbacks is that it can make your application more difficult to debug.

Client-side rendering is typically used for applications that are not heavily data-driven. If your application relies on a lot of data from a server, client-side rendering can be very slow. Additionally, if you're using client-side rendering, it's important to be careful about how you load and cache your data. If you're not careful, you can easily end up with an application that is very slow and difficult to use. When rendered on the server-side, this is called Angular Universal.

41. How do you share data between components in Angular?

-> Sharing data between components in Angular is simple and easy. To share data, all you need to do is use the Angular CLI to generate a new service. This service can be injected into any component and will allow the components to share data.

To generate a new service, use the following Angular CLI command:

```
ng generate service my-data-service
```

This will create a new service file called my-data-service.ts in the src/app folder.

Inject the service into any component that needs to share data:

```
import { MyDataService } from './my-data.service';
```

```
constructor(private myDataService: MyDataService) { }
```

Once injected, the service will be available in the component as this.myDataService.

To share data between components, simply use the setData() and getData() methods:

```
this.myDataService.setData('some data');
```

```
const data = this.myDataService.getData();
```

42. Explain the concept of dependency injection.

-> Dependency injection is a technique used to create loosely coupled code. It allows pieces of code to be reused without the need for hard-coded dependencies. This makes code more maintainable and easier to test. Dependency injection is often used in frameworks like AngularJS, ReactJS, and VueJS. It is also possible to use dependency injection in vanilla JavaScript. To use dependency injection in JavaScript, you need a dependency injection library. There are many different libraries available, but they all work in basically the same way.

The first step is to create a dependency injection container. This is a simple object that will hold all of the dependencies that your code needs. Next, you need to register all of the dependencies that your code will need with the container. The registration process will vary depending on the library you are using, but it is usually just a matter of providing the dependency's name and constructor function.

Once all of the dependencies have been registered, you can then inject them into your code. The injection process will again vary depending on the library you are using, but it is usually just a matter of providing the dependency's name. The library will then take care of instantiating the dependency and passing it to your code.

Dependency injection can be a great way to make your code more modular and easier to maintain. It can also make it easier to unit test your code, as you can inject mock dependencies instead of the real ones. If you are using a framework that supports dependency injection, then it is probably already being used in your code. If you are not using a framework, then you can still use dependency injection by choosing a library and following the steps outlined above.

Code example:

```
import { Injectable } from '@angular/core';
   @Injectable({
    providedIn: 'root'
   })
   export class TestService {
    importantValue:number = 42;
    constructor() { }
    returnImportantValue(){
      return this.importantValue;
    }
   }
```

The injectable dependencies are created after adding the @Injectable decorator to a class. The dependency above is then injected into the following component:

```
import { TestService } from './../test.service';
   import { Component, OnInit } from '@angular/core';
   @Component({
    selector: 'app-test',
    templateUrl: './test.component.html',
    styleUrls: ['./test.component.css']
   })
   export class TestComponent implements OnInit {
    value:number;
    constructor(private testService:TestService) { }
    ngOnInit() {
      this.value = this.testService.returnImportantValue();
    }
   }
```

43. Explain MVVM architecture.

-> MVVM architecture is an architectural pattern used mainly in software engineering. It stands for Model-View-ViewModel. MVVM is a variation of the traditional MVC (Model-View-Controller) software design pattern. The main difference between the two is that MVVM separates the user interface logic from the business logic, while MVC separates the data access logic from the business logic. This separation of concerns makes it easier to develop, test, and maintain software applications.

The Model layer in MVVM architecture is responsible for storing and managing data. It can be a database, a web service, or a local data source. The View layer is responsible for displaying data to the user. It can be a graphical user interface (GUI), a command-line interface (CLI), or a web page. The ViewModel layer is responsible for handling user input and updating the View layer accordingly. It contains the business logic of the application.

MVVM architecture is often used in conjunction with other software design patterns, such as Model-View-Presenter (MVP) and Model-View-Controller (MVC). These patterns can be used together to create a complete software application.

MVVM architecture is a popular choice for modern software applications. It allows developers to create applications that are more responsive and easier to maintain. Additionally, MVVM architecture can be used to create applications that can be easily ported to different platforms.