1. What is Angular? Why was it introduced?

[Angular was introduced](https://www.simplilearn.com/tutorials/angular-tutorial/what-is-angular) 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.](https://www.simplilearn.com/tutorials/programming-tutorial/advanced-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](https://www.simplilearn.com/tutorials/javascript-tutorial/introduction-to-javascript) that can run efficiently in any environment.

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

[Data binding](https://www.simplilearn.com/tutorials/angular-tutorial/angular-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 elements on the existing page itself. A SPA approach is faster, thus providing a seamless user experience.

5. Differentiate between Angular and AngularJS

The following table depicts the aspects of Angular vs AngularJS in detail:

|  |  |  |
| --- | --- | --- |
| Feature | AngularJS | Angular |
| Language | JavaScript | TypeScript |
| Architecture | Supports Model-View-Controller design | Uses components and directives |
| Mobile support | Not supported by mobile browsers | Supports all popular mobile browsers |
| Dependency Injection | Doesn’t support | Supports |
| Routing | @routeProvider is used to provide routing information | @Route configuration is used to define routing information |
| Management | Difficult to manage with an increase in source code size | Better structured, easy to create and manage bigger applications |

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 -

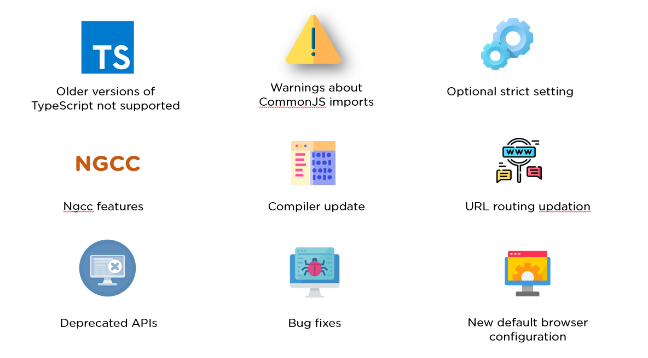
1. [MVC architecture](https://www.simplilearn.com/tutorials/dot-net-tutorial/mvc-architecture) - Angular is a full-fledged MVC framework. It provides a firm opinion on how the application should be structured. It also offers bi-directional data flow and updates the real DOM.
2. Modules: Angular consists of different design patterns like components, directives, pipes, and services, which help in the smooth creation of applications.
3. [Dependency injection](https://www.simplilearn.com/tutorials/angular-tutorial/angular-dependency-injection): Components dependent on other components can be easily worked around using this feature.
4. Other generic advantages include clean and maintainable code, unit testing, reusable components, data binding, and excellent responsive experience.

Angular is a robust front-end JavaScript framework that is widely used for app development. With the increased popularity, there is a high demand for Angular developers. This article on Angular Interview Questions will present some commonly asked questions and how to answer them. The questions are bifurcated into two levels, beginner and advanced.

To brush up your basics, check out our [videos on YouTube](https://www.youtube.com/watch?v=TC1oVXlVE3M&list=PLEiEAq2VkUUIzcMkvJKwXLf13SqV8lPes). You could also refer to our articles for better clarity. Now, without further ado, let's begin with the top Angular interview questions and answers you must be prepared for.

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 elements on the existing page itself. A SPA approach is faster, thus providing a seamless user experience.

### 8. What are the new updates with Angular10?

* Older versions of TypeScript not supported - Previous versions of Angular supported typescript 3.6, 3.7, and even 3.8. But with Angular 10, TypeScript bumped to TypeScript 3.9.
* Warnings about CommonJS imports - Logging of unknown property bindings or element names in templates is increased to the "error" level, which was previously a "warning" before.
* Optional strict setting - Version 10 offers a stricter project setup when you create a new workspace with ng new command.

ng new --strict

NGCC Feature - Addition of NGCC features with a program based entry point finder.

* Updated URL routing
* Deprecated APIs - Angular 10 has several deprecated APIs.
* Bug fixes - With this Angular 10 version, there have been a number of bug fixes, important ones being the compiler avoiding undefined expressions and the core avoiding a migration error when a nonexistent symbol is imported.
* New Default Browser Configuration - Browser configuration for new projects has been upgraded to outdo older and less used browsers.

### 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.

## Angular Interview Questions For Beginners

### 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

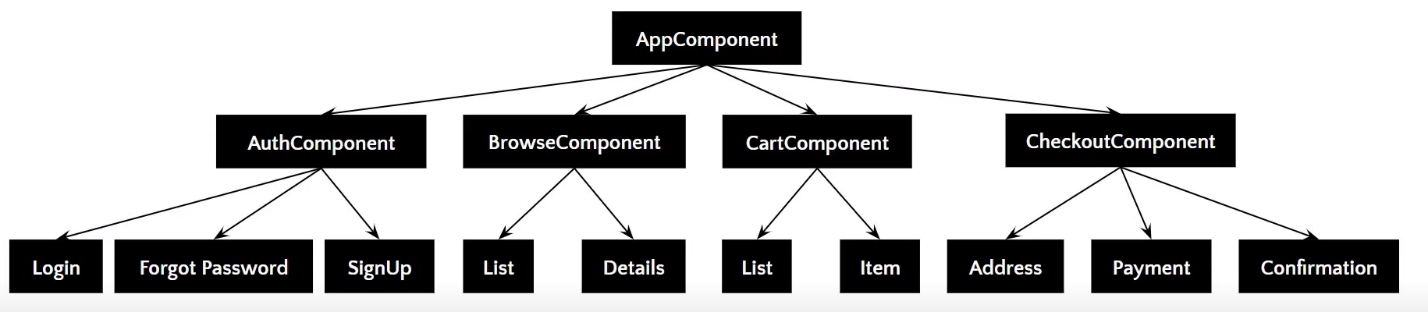
### 12. What is an AOT compilation? What are its advantages?

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.

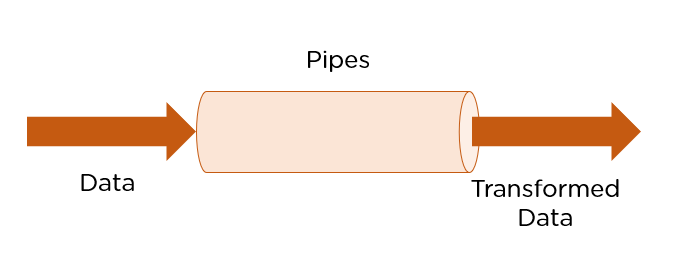
1. Faster rendering
2. Fewer asynchronous requests
3. Smaller Angular framework download size
4. Quick detection of template errors
5. Better security

### 13. What are Components in Angular?



[Components](https://www.simplilearn.com/tutorials/angular-tutorial/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](https://www.simplilearn.com/tutorials/angular-tutorial/angular-pipes) 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:

1. Emulated - styles from the main HTML propagate to the component.
2. Native - styles from the main HTML do not propagate to the component.
3. 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 -

1. ngOnChanges() - Responds when Angular sets/resets data-bound input properties.
2. ngOnInit() - Initialize the directive/component after Angular first displays the data-bound properties and sets the directive/component's input properties/
3. ngDoCheck() - Detect and act upon changes that Angular can't or won't detect on its own.
4. ngAfterContentInit() - Responds after Angular projects external content into the component's view.
5. ngAfterContentChecked() - Respond after Angular checks the content projected into the component.
6. ngAfterViewInit() - Respond after Angular initializes the component's views and child views.
7. ngAfterViewChecked() - Respond after Angular checks the component's views and child views.
8. 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

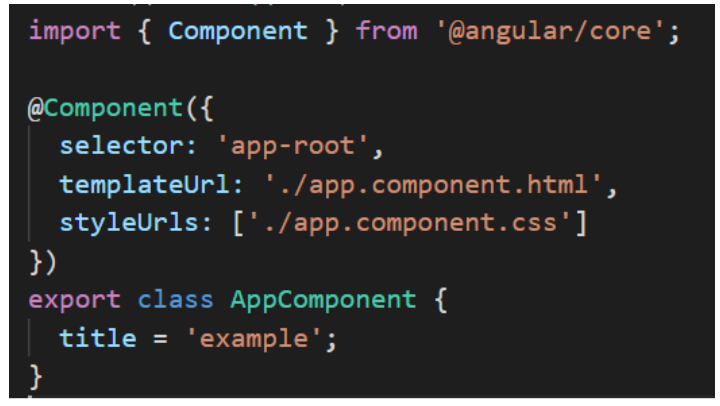
### 26. What is the difference between AOT and JIT?

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.

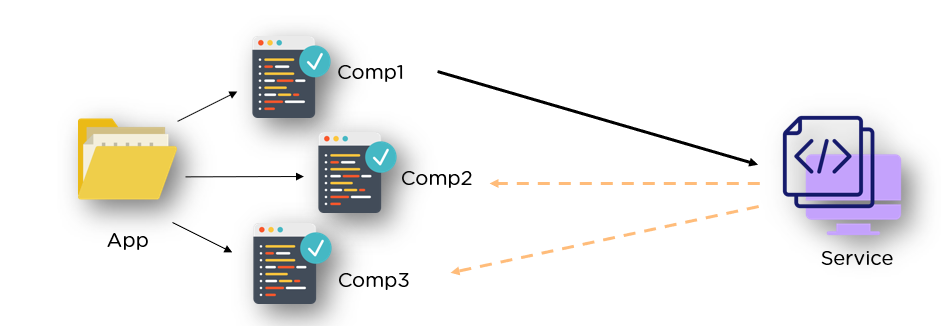
TypeScript class is one that is used to create components. This genre of class is then decorated with the "@Component" decorator. The decorato’s purpose is to accept a metadata object that provides relevant information about the component.



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](https://www.simplilearn.com/tutorials/angular-tutorial/angular-service) 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](https://www.simplilearn.com/tutorials/angular-tutorial/angular-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](https://www.simplilearn.com/tutorials/angular-tutorial/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](https://www.simplilearn.com/tutorials/angular-tutorial/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.

## Angular Interview Questions for Experienced

### 44. What are RxJs in Angular?

RxJs is a library that provides reactive programming support for Angular applications. It allows you to work with asynchronous data streams and handle events over time. RxJs is based on Observables, which are data streams that can be subscribed to and processed using operators. It provides a powerful and flexible way to handle complex asynchronous operations in Angular.

### 45. What exactly is a parameterized pipe?

A parameterized pipe in Angular is a pipe that accepts one or more arguments, also known as parameters. Pipes transform data in Angular templates, and parameterized pipes allow you to customize the transformation based on specific requirements. By passing parameters to a pipe, you can modify its behavior and apply different transformations to the data.

### 46. What are class decorators?

Class decorators in Angular are a type of decorator that can be applied to a class declaration. They are used to modify the behavior of the class or add additional functionality. Class decorators are defined using the @ symbol followed by the decorator name and are placed immediately before the class declaration. They can be used for various purposes, such as adding metadata, applying mixins, or extending the functionality of a class.

### 47. What are Method decorators?

Method decorators in Angular are decorators that can be applied to methods within a class. They are used to modify the behavior of the method or add additional functionality. Method decorators are defined using the @ symbol followed by the decorator name and are placed immediately before the method declaration. They can be used for tasks like logging, caching, or modifying the method's behavior based on specific conditions.

### 48. What are property decorators?

Property decorators in Angular are decorators that can be applied to class properties. They are used to modify the behavior of the property or add additional functionality. Property decorators are defined using the @ symbol followed by the decorator name and are placed immediately before the property declaration. They can be used for tasks like validation, memoization, or accessing metadata associated with the property.

### 49. What are router links?

Router links in Angular are used for navigation within an application. They are defined using the routerLink directive and provide a way to navigate to different routes or components. Router links can be used in HTML templates and are typically placed on anchor (<a>) tags or other clickable elements. By specifying the destination route or component, router links allow users to navigate between different parts of an Angular application.

### 50. What exactly is the router state?

The router state in Angular represents the current state of the Angular router. It contains information about the current route, including the URL, route parameters, query parameters, and other related data. The router state can be accessed and manipulated using the Angular Router service. It provides a way to programmatically navigate, retrieve information about the current route, and handle route changes in Angular applications.

### 51. What does Angular Material mean?

Angular Material is a UI component library for Angular applications. It provides a set of pre-built and customizable UI components, such as buttons, forms, navigation menus, and dialog boxes, that follow the Material Design guidelines. Angular Material simplifies the process of building consistent and visually appealing user interfaces in Angular. It offers a range of features and styles that can be easily integrated into Angular projects.

### 52. What is transpiling in Angular?

Transpiling in Angular refers to the process of converting TypeScript code into JavaScript code that web browsers can execute. Angular applications are built using TypeScript, a superset of JavaScript that adds static typing and additional features to the language. Since browsers can only run JavaScript, the TypeScript code needs to be transpiled into JavaScript before it can be executed. This is typically done using the TypeScript compiler (tsc) or build tools like Angular CLI.

### 53. What are HTTP interceptors?

HTTP interceptors in Angular are a feature that allows you to intercept HTTP requests and responses globally. Interceptors provide a way to modify or handle HTTP requests and responses at a centralized location before they reach the server or client. This can be useful for logging requests, adding authentication headers, handling errors, or modifying request/response data. Interceptors can be registered in the Angular module and are executed in a specific order based on the request/response pipeline.

### 54. What is Change Detection, and how does the Change Detection Mechanism work?

Change Detection in Angular is a mechanism that determines when and how to update the user interface based on changes in the application's data model. Angular uses a tree of change detectors to track changes in component properties and update the DOM accordingly. When a change occurs, Angular performs a process called change detection, which involves checking each component's properties for changes and updating the DOM if necessary. The change detection mechanism is responsible for keeping the UI in sync with the application's data.

55. What is a bootstrapping module?

A bootstrapping module in Angular is the main entry point of an Angular application. It is responsible for starting the application and initializing the root component. The bootstrapping module is typically defined in the main.ts file and is configured in the Angular AppModule. It sets up the necessary environment, loads required modules, and invokes the Angular platform to start the application. The bootstrapping module plays a crucial role in the Angular application's lifecycle.

Angular Scenario Based Interview Questions

56. How do you choose anc from a component template?

You can use various techniques to choose an element from a component template in Angular. One common approach is to use template reference variables. The template defines these variables using the # symbol followed by a name. You can then reference the element using the variable name in your component code. Another approach is to use Angular directives like ViewChild or ViewChildren to query for elements based on CSS selectors or component types. These directives provide more flexibility and control when selecting elements from the component template.

57. How do you deal with errors in observables?

When dealing with errors in observables in Angular, catchError operator can be used to handle and recover from errors. This operator allows you to provide a fallback value or execute alternative logic when an error occurs. You can chain the catchError operator after the observable that might produce an error and define a callback function to handle the error. Within the callback function, you can perform error handling tasks such as logging the error, displaying an error message to the user, or initiating a retry mechanism.

58. How to include SASS into an Angular project?

To include SASS (Syntactically Awesome Style Sheets) into an Angular project, you need to install the required dependencies and configure the project accordingly. Follow these steps:

* Install the node-sass package by running the command npm install node-sass --save-dev.
* Update the angular.json file in your project's root directory.
* Locate the styles property under architect > build > options.
* Change the file extension from .css to .scss to indicate that you are using SASS.
* Rename your existing CSS files to SCSS files (e.g., styles.css to styles.scss).
* Restart the Angular development server for the changes to take effect.

Once SASS is included in your Angular project, you can write your styles using SASS syntax, which provides additional features like variables, mixins, and nested selectors.

59. What happens when we use the script tag within a template?

Using the script tag within an Angular template is not a recommended practice. Angular templates are intended for defining the structure and layout of the user interface, and including scripts directly within the template goes against the separation of concerns principle. When a script tag is used within a template, the browser treats it as part of the HTML content and attempts to execute it. However, Angular's template compiler does not process or execute scripts within templates. Instead, scripts should be placed in separate JavaScript files and included using the appropriate Angular mechanisms, such as component logic or Angular modules.

60. Write a code to share data from the Parent to Child Component?

To share data from a parent component to a child component in Angular, you can make use of input properties. Input properties allow you to pass data from a parent component to a child component. Here's an example:

Parent Component:

typescript

Copy code

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

@Component({

  selector: 'app-parent',

  template: `

    <app-child [message]="parentMessage"></app-child>

  `,

})

export class ParentComponent {

  parentMessage = 'Hello from parent';

}

Child Component:

typescript

Copy code

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

@Component({

  selector: 'app-child',

  template: `

    <p>{{ message }}</p>

  `,

})

export class ChildComponent {

  @Input() message: string;

}

In this example, the parent component (ParentComponent) defines a property parentMessage that holds the data to be shared with the child component (ChildComponent). The parent component then passes this data to the child component using the input property [message]. The child component receives the data through the @Input() decorator and can use it within its template or logic.

61. Create a TypeScript class with a constructor and a function.

Here's an example of a TypeScript class with a constructor and a function:

typescript

Copy code

class Person {

  name: string;

  age: number;

  constructor(name: string, age: number) {

    this.name = name;

    this.age = age;

  }

  sayHello() {

    console.log(`Hello, my name is ${this.name} and I'm ${this.age} years old.`);

  }

}

// Usage

const person = new Person('John', 30);

person.sayHello();

In this example, the Person class represents a person with a name and an age property. The constructor is used to initialize these properties when creating an instance of the class. The sayHello function is a method of the class that logs a greeting message using the person's name and age. Finally, an instance of the Person class is created, and the sayHello function is called to output the greeting message.

Next Steps

We hope that this article on Angular Interview Questions helped you understand the type of questions asked and how to answer them. But, if you wish to learn Angular and perhaps make a career out of it, certification will come in handy.

You can also explore and get familiar with interview questions related to other frontend languages like Angular, CSS or [ReactJS interview questions](https://www.simplilearn.com/tutorials/reactjs-tutorial/reactjs-interview-questions).

Simplilearn's Angular Certification Training Course helps you master front-end web [development with Angular](https://www.simplilearn.com/how-to-become-an-angular-developer-article). You will learn the knack of creating applications with the help of concepts like facilitating the development of single-page web applications, components, typescript, dependency injection, and directives with this course. The course also includes a real-time project to test your skills.

If you are looking to enhance your software development skills, we recommend you check Simplilearn's [Post Graduate Program in Full Stack Web Development](https://www.simplilearn.com/pgp-full-stack-web-development-certification-training-course?source=GhPreviewCoursepages). This course, in collaboration with Caltech CTME, can help you hone the right skills and make you job-ready.

Do you have any questions for us? If you have feedback or questions, drop us a comment in the comments section of this Angular interview questions tutorial. Our experts will get back to you as soon as possible!

FAQs

1. What are the key concepts of Angular?

Some key concepts of Angular include components, modules, templates, data binding, services, dependency injection, and routing. These concepts form the foundation of Angular development and help in building dynamic and scalable web applications.

2. What is Angular coding language?

Angular itself is not a coding language. It is a framework for building web applications using TypeScript. TypeScript is a superset of JavaScript that adds static typing and additional features to JavaScript. Angular leverages TypeScript to provide a more structured and scalable approach to web development.

3. What is a service in Angular interview questions?

In Angular, a service is a class that is used to share data or functionality across different components. Services are responsible for providing specific functionality that can be used by multiple components in an application. They promote reusability, modularity, and maintainability in Angular projects.

4. What is the full form of ng in Angular?

In Angular, "ng" stands for "Angular". It is a convention used in Angular to prefix directives, components, and other Angular-specific entities. For example, ngFor is a built-in Angular directive used for rendering lists based on an array, and ngModel is a directive used for two-way data binding between an input element and a component property. The "ng" prefix helps to distinguish Angular-specific entities from regular HTML elements and attributes.

# **Angular Interview Questions and Answers**

### 1) What is Angular? / What do you know about Angular?

Angular is one of the most popular JavaScript frameworks developed and maintained by Google. It is an open-source front-end web framework based on TypeScript. It is most suited for developing enterprise web applications because its code is reusable and maintainable.

### 2) What are some powerful features integrated into Angular?

Angular integrates some powerful features like declarative templates, end to end tooling, dependency injection and various other best practices that smoothens the development path.

### 3) What is the main purpose of Angular?

The main purpose of using Angular is to create fast, dynamic and scalable web applications. We can create these applications very easily with Angular using components and directives.

Angular was started as a SPA (Single-Page-Application) framework, and now it supports dynamic content based on different users through dependency injection. It provides a platform for easy development of web-based applications and empowers the front end developers in curating cross-platform applications. YouTubeTV is the most popular example that uses Angular.

### 4) What is the difference between AngularJS and Angular?

Let's compare the features of AngularJS and Angular in a tabular form:

**A list of differences between AngularJS and Angular-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** |  | **AngularJS** | **Angular** |
| **Version** |  | AngularJS was the very first version initially released in 2010. It was a browser-side JavaScript used within HTML code and created a revolution in web application development. It is popularly known as AngularJS. | The later Angular versions were a complete rewrite of AngularJS. For example, Angular 2 was initially released in 2016. There is nothing common between Angular2 and AngularJS except the core developer's team. After that, Angular 6, Angular 7, Angular 8, Angular 9, and Angular 10 were released that are very similar to each other. These later versions are known as Angular. |
| **Architecture** |  | AngularJS supports the MVC design model. | Angular uses components and directives. |
| **Supported Language** |  | The recommended and supported language of AngularJS is JavaScript. | The recommended and supported language of Angular is TypeScript. |
| **Expression Syntax** |  | In AngularJS, a specific ng directive is required for the image/property and an event. | Angular uses () for event binding and [] for property binding. |
| **Mobile Support** |  | AngularJS doesn't provide any mobile support. | Angular provides mobile support. |
| **Dependency Injection** |  | There is no concept of Dependency Injection in AngularJS. | Angular supports hierarchical Dependency Injection with uni-directional tree-based change detection. |
| **Routing** |  | In AngularJS, $routeprovider.when() is used for routing configs. | In Angular, @RouteConfig{(?)} is used for the routing config. |
| **Structure** |  | It is the first and basic version, so it is very easy to manage. | It has a very simplified structure that makes the development and maintenance of large applications very easy. |
| **Speed** |  | It is slower because of its limited features. | It is faster than AngularJS because of its upgraded features. |
| **Support** |  | It doesn't provide support or new updates anymore. | It provides active support, and frequent new updates are made. |

### 5) What are the biggest advantages of using Angular?

Following is the list of the biggest advantages of using the Angular framework:

* Angular supports two-way data-binding.
* It follows MVC pattern architecture.
* It supports static templates and Angular template.
* It facilitates you to add a custom directive.
* It also supports RESTfull services.
* Validations are supported in Angular.
* Angular provides client and server communication.
* It provides support for dependency injection.
* It provides powerful features like Event Handlers, Animation, etc.

### 6) What do you understand by Angular expressions? How are Angular expressions different from JavaScript expressions?

Angular expressions are code snippets that are used to bind application data to HTML. Angular resolves the expressions, and the result is returned to where the expression is written. Angular expressions are usually written in double braces: {{ expression }} similar to JavaScript.

**Syntax:**

{{ expression }}

Following is a list of some differences between Angular expressions and JavaScript expressions:

1. The most crucial difference between Angular expressions and JavaScript expressions is that the Angular expressions allow us to write JavaScript in HTML. On the other hand, the JavaScript expressions don't allow.

2. The Angular expressions are evaluated against a local scope object. On the other hand, the JavaScript expressions are evaluated against the global window object. We can understand it better with an example. Suppose we have a component named test:

1. **import** { Component, OnInit } from '@angular/core';
2. @Component({
3. selector: 'app-test',
4. template: `
5. <h4>{{message}}</h4>,
6. styleUrls: ['./test.component.css']
7. })
8. export **class** TestComponent **implements** OnInit {
9. message:string = ?Hello world?;
10. constructor() { }
11. ngOnInit() {
12. }
13. }

In the above example, we can see that the Angular expression is used to display the message property. In the present template, we are using Angular expressions, so we cannot access a property outside its local scope (in this case, TestComponent). This proves that Angular expressions are always evaluated based on the scope object rather than the global object.

3. The Angular expressions can handle null and undefined, whereas JavaScript expressions cannot.

See the following JavaScript example:

1. <!DOCTYPE html**>**
2. **<html** lang="en"**>**
3. **<head>**
4. **<meta** charset="UTF-8"**>**
5. **<meta** name="viewport" content="width=device-width, initial-scale=1.0"**>**
6. **<title>**JavaScript Test**</title>**
7. **</head>**
8. **<body>**
9. **<div** id="foo"**><div>**
10. **</body>**
11. **<script>**
12. 'use strict';
13. let bar = {};
14. document.getElementById('foo').innerHTML = bar.x;
15. **</script>**
16. **</html>**

After running the above code, you see undefined displayed on the screen. Although it's not ideal to leave any property undefined, the user does not need to see this.

Now see the following Angular example:

1. **import** { Component, OnInit } from '@angular/core';
2. @Component({
3. selector: 'app-new',
4. template: `
5. <h4>{{message}}</h4>       `,
6. styleUrls: ['./new.component.css']
7. })
8. export **class** NewComponent **implements** OnInit {
9. message:object = {};
10. constructor() { }
11. ngOnInit() {
12. }
13. }

In the above example, you will not see undefined being displayed on the screen.

4. In Angular expressions, we cannot use loops, conditionals, and exceptions. The difference which makes Angular expressions quite beneficial is the use of pipes. Angular uses pipes (known as filters in AngularJS) to format data before displaying it.

See this example:

1. **import** { Component, OnInit } from '@angular/core';
2. @Component({
3. selector: 'app-new',
4. template: `
5. <h4>{{message | lowercase}}</h4>,
6. styleUrls: ['./new.component.css']
7. })
8. export **class** NewComponent **implements** OnInit {
9. message:string = "HELLO JAVATPOINT";
10. constructor() { }
11. ngOnInit() {
12. }
13. }

In the above example, we have used a predefined pipe called lowercase, which transforms all the letters in lowercase. If you run the above example, you will see the output displayed as "hello javatpoint".

On the other hand, JavaScript does not have the concept of pipes.

### 7) What are templates in Angular?

In Angular, templates contain Angular-specific elements and attributes. These are written with HTML and combined with information coming from the model and controller, which are further rendered to provide the user's dynamic view.

### 8) What is the difference between an Annotation and a Decorator in Angular?

In Angular, annotations are the "only" metadata set of the class using the Reflect Metadata library. They are used to create an "annotation" array. On the other hand, decorators are the design patterns used for separating decoration or modification of a class without actually altering the original source code.

### 9) Why was Angular introduced as a client-side framework?

Before the introduction of Angular, web developers used VanillaJS and jQuery to develop dynamic websites. Later, when the websites became more complex with added features and functionality, it was hard for them to maintain the code. Along with this, there were no provisions of data handling facilities across the views by jQuery. The need for a client-side framework like Angular was obvious that can make life easier for the developers by handling separation of concerns and dividing code into smaller bits of information (components).

Client-side frameworks like Angular facilitate developers to develop advanced web applications like Single-Page-Application. These applications can also be developed using VanillaJS, but the development process becomes slower by doing so.

### 10) How does an Angular application work?

Every Angular app contains a file named angular.json. This file contains all the configurations of the app. While building the app, the builder looks at this file to find the application's entry point. See the structure of the angular.json file:

1. "build": {
2. "builder": "@angular-devkit/build-angular:browser",
3. "options": {
4. "outputPath": "dist/angular-starter",
5. "index": "src/index.html",
6. "main": "src/main.ts",
7. "polyfills": "src/polyfills.ts",
8. "tsConfig": "tsconfig.app.json",
9. "aot": false,
10. "assets": [
11. "src/favicon.ico",
12. "src/assets"
13. ],
14. "styles": [
15. "./node\_modules/@angular/material/prebuilt-themes/deeppurple-amber.css",
16. "src/style.css"
17. ]
18. }
19. }

When the application enters the build section, the options object's main property defines the entry point of the application. The application's entry point is main.ts, which creates a browser environment for the application to run and calls a function called bootstrapModule, which bootstraps the application.

These two steps are performed in the following order inside the main.ts file:

1. **import** { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
2. 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:**

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

In the above file, you can see that AppComponent is getting bootstrapped. It 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:**

1. **import** { Component } from '@angular/core';
2. @Component({
3. selector: 'app-root',
4. templateUrl: './app.component.html',
5. styleUrls: ['./app.component.css']
6. })
7. export **class** AppComponent {
8. title = 'angular';
9. }

Each component is declared with three properties:

1. **Selector -** It is used to access the component.
2. **Template/TemplateURL -** It contains HTML of the component.
3. **StylesURL -** It contains component-specific stylesheets.

Now, Angular calls the index.html file. This file consequently calls the root component that is app-root. The root component is defined in app.component.ts.

**See how the index.html file looks like:**

1. <!doctype html**>**
2. **<html** lang="en"**>**
3. **<head>**
4. **<meta** charset="utf-8"**>**
5. **<title>**Angular**</title>**
6. **<base** href="/"**>**
7. **<meta** name="viewport" content="width=device-width, initial-scale=1"**>**
8. **</head>**
9. **<body>**
10. **<app-root></app-root>**
11. **</body>**
12. **</html>**

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

### 11) Why is Angular preferred over other frameworks? / What are some advantages of Angular over other frameworks?

Due to the following features, Angular is preferred over other frameworks:

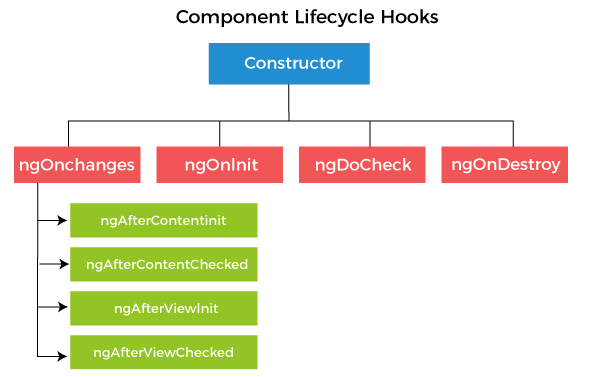
**Extraordinary Built-in Features:** Angular provides several out of the box built-in features like routing, state management, RxJS library, Dependency Injection, HTTP services, etc. That's why the developers do not need to look for the above-stated features separately.

**Declarative UI:** Angular has declarative UI. It uses HTML to render the UI of an application as it is a declarative language. It is much easier to use than JavaScript.

**Long-term Google Support:** Angular is developed and maintained by Google. Google has a long term plan to stick with Angular and provide support.

### 12) What are the different Lifecycle hooks of Angular? Explain them in short.

When the Angular components are created, they enter their lifecycle and remain when they are destroyed. Angular Lifecycle hooks are used to check the phases and trigger changes at specific phases during the entire duration.



**ngOnChanges( ):** This method is called when one or more input properties of the component are changed. The hook receives a SimpleChanges object containing the previous and current values of the property.

**ngOnInit( ):** This is the second lifecycle hook. It is called once, after the ngOnChanges hook. It is used to initialize the component and sets the input properties of the component.

**ngDoCheck( ):** This hook is called after ngOnChanges and ngOnInit and is used to detect and act on changes that Angular cannot detect. In this hook, we can implement our change detection algorithm.

**ngAfterContentInit( ):** This hook is called after the first ngDoCheck hook. This hook responds after the content gets projected inside the component.

**ngAfterContentChecked( ):** This hook is called after ngAfterContentInit and every subsequent ngDoCheck. It responds after the projected content is checked.

**ngAfterViewInit( ):** This hook is called after a component's view or initializing a child component's view.

**ngAfterViewChecked( ):** This hook is called after ngAfterViewInit. It responds after the component's view or when the child component's view is checked.

**ngOnDestroy( ):** This hook is called just before Angular destroys the component. This is used to clean up the code and detach event handlers.

In the above hooks we have described, the ngOnInit hook is the most often used hook. Let's see how to use the ngOnInit hook. If you have to process a lot of data during component creation, it's better to do it inside the ngOnInit hook rather than the constructor:

**See the example:**

1. **import** { Component, OnInit } from '@angular/core';
2. @Component({
3. selector: 'app-test',
4. templateUrl: './test.component.html',
5. styleUrls: ['./test.component.css']
6. })
7. export **class** TestComponent **implements** OnInit {
8. constructor() { }
9. ngOnInit() {
10. **this**.processData();
11. }
12. processData(){
13. // Do something..
14. }
15. }

In the above code, you can see that we have imported OnInit, but we have used the ngOnInit function. This is how we can use the rest of the hooks as well.

### 13) What is AOT in Angular?

In Angular, AOT stands for Ahead-Of-Time compiler. It is used to convert your Angular HTML and TypeScript code into efficient JavaScript code during the build phase before the browser downloads and runs that code. By compiling the application during the build process provides a faster rendering in the browser.

### 14) What is the reason for using the AOT compiler in Angular?

An Angular application is made of several components and their HTML templates. Because of these Angular components and templates, the browsers are not able to understand them directly. So, Angular applications require a compilation process before they run in a browser. That's why AOT compilers are required.

### 15) What are the biggest advantages of AOT in Angular?

Following are the advantages of using the AOT compiler in Angular:

**The rendering is faster:** When we use the AOT compiler, the browser gets a pre-compiled version of the application to download. Here, the browser loads executable code to render the application immediately, without waiting to compile the app first.

**The Angular framework's download size is smaller:** AOT facilitates you not to download the Angular compiler if the app is already compiled. The compiler is roughly half of Angular itself, so omitting it dramatically reduces the application payload.

**Fewer asynchronous requests:** The compiler is used to inline external HTML templates and CSS style sheets within the application JavaScript so, it eliminates separate AJAX requests for those source files.

**Detect template errors earlier:** While using the AOT compiler, developers can easily detect and report template binding errors during the build step before users can see them.

**Better security:** AOT provides better security because it compiles HTML templates and components into JavaScript files before they are served to the client. Because there are no templates to read and no risky client-side HTML or JavaScript evaluation, so the chances for injection attacks are very rare.

### 16) What is JIT in Angular?

In Angular, JIT stands for Just-in-Time compiler. The JIT compiler provides a dynamic translation or run-time compilation, which provides a way of executing computer code that involves compilation during the execution of a program at run time rather than before execution.

### 17) What is the main difference between JIT and AOT in Angular?

Following are the main differences between JIT and AOT compiler in Angular:

* Just-in-Time (JIT) compiler compiles our app in the browser at run-time while Ahead-of-Time (AOT) compiler is used to compile your app at build time on the server.
* The JIT compilation runs by default when you run the ng build (build only), or ng serve (build and serve locally) CLI commands. This is used for development. On the other hand, we have to include the --aot option with the ng build or ng serve command for AOT compilation.
* JIT and AOT are both two ways used to compile code in an Angular project. JIT compiler is used in development mode while AOT is used for production mode.
* JIT is easy to use. We can easily implement features and debug in JIT mode because here we have a map file while AOT does not. On the other hand, the biggest advantage of using AOT for production is that it reduces the bundle size for faster rendering.

### 18) What is the concept of scope hierarchy in Angular?

Angular provides the $scope objects into a hierarchy that is typically used by views. This is called the scope hierarchy in Angular. It has a root scope that can further contain one or several scopes called child scopes.

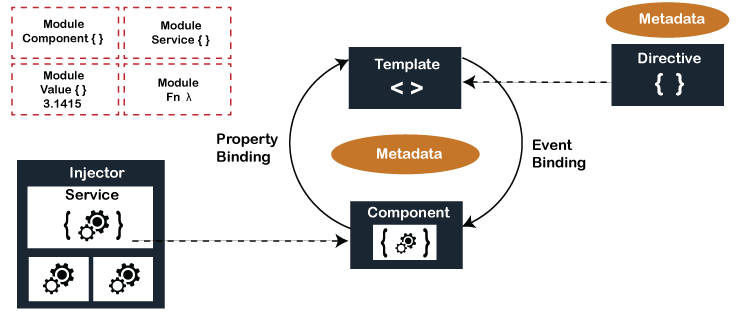
In a scope hierarchy, each view has its own $scope. Hence, the variables set by a view's view controller will remain hidden to other view controllers.

**Following is the typical representation of a Scope Hierarchy:**

1. Root $scope
2. $scope for Controller 1
3. $scope for Controller 2
4. ...
5. ..
6. .
7. $scope for Controller n

### 19) What are the main building blocks of an Angular application? Explain with the pictorial diagram of Angular architecture.

Following are the main building blocks of an Angular application. You can see them in the following picture:



### 20) What is the difference between Observables and Promises in Angular?

In Angular, as soon as we make a promise, the execution takes place, but this is not the case with observables because they are lazy. It means nothing happens until a subscription is made.

|  |  |
| --- | --- |
| **Promise** | **Observable** |
| It emits a single value. | It emits multiple values over a period of time. |
| Not Lazy | Lazy. An observable is not called until we subscribe to the observable. |
| We can not cancel it. | We can cancel it by using the unsubscribe() method. |
|  | Observable provides operators like map, forEach, filter, reduce, retry, retryWhen etc. |

**Let's understand it by an example:**

1. const observable = rxjs.Observable.create(observer =**>** {
2. console.log('This is what inside an observable');
3. observer.next('Hello JavaTpoint');
4. observer.complete();
5. });
6. console.log('Before subscribing an Observable');
7. observable.subscribe((message)=**>** console.log(message));

When you run the above Observable, you can see the following messages displayed in the following order:

1. Before subscribing an Observable
2. This is what inside an observable
3. Hello JavaTpoint

Here, you can see that observables are lazy. Observable runs only when someone subscribes to them. That's why the message "Before subscribing an Observable" is displayed ahead of the message inside the observable.

**Now see the example of a Promise:**

1. const promise = new Promise((resolve, reject) =**>** {
2. console.log('This is what written inside promise');
3. resolve('Hello JavaTpoint');
4. });
5. console.log('Before calling then method on Promise');
6. greetingPoster.then(message =**>** console.log(message));

When you run the above Promise, you will see the messages displayed in the following order:

1. This is what written inside Promise
2. Before calling then method on Promise
3. Hello JavaTpoint

Here, you can see that the message inside Promise is displayed first. This means that the Promise runs first, and then the method is called.

The next difference between them is that Promises are always asynchronous; even when the Promise is immediately resolved. On the other hand, an Observable can be both synchronous and asynchronous.

In the case of the above example, observable is synchronous. Let's see the case where an observable can be asynchronous:

1. const observable = rxjs.Observable.create(observer =**>** {
2. setTimeout(()=**>**{
3. observer.next('Hello JavaTpoint');
4. observer.complete();
5. },3000)
6. });
7. console.log('Before calling subscribe on an Observable');
8. observable.subscribe((data)=**>** console.log(data));
9. console.log('After calling subscribe on an Observable');

When you run the above observable, you will see the messages in the following order:

1. Before calling subscribe on an Observable
2. After calling subscribe on an Observable
3. Hello JavaTpoint

### 22) What are directives in Angular?

A directive is a class in Angular that is declared with a @Directive decorator. Every directive has its own behavior, and you can import them into various components of an application.

### 23) What were the main reasons behind introducing client-side frameworks like Angular?

Before Angular was introduced, the web developers used VanillaJS and jQuery to develop dynamic websites, but the biggest drawback of these technologies is that as the logic of the website grew, the code became more and more complex to maintain. For websites and applications that use complex logic, developers had to put in extra effort to maintain the separation of concerns for the app. Also, jQuery did not provide facilities for data handling across views.

The client-side frameworks like Angular were introduced to overcome the above problems. It provides developers many benefits over VanilaJS and jQuery by providing a new feature called components for handling separation of concerns and dividing code into smaller bits of information.

Client-side frameworks such as Angular facilitate developers to develop advanced web applications like Single-Page-Applications. So, the main reasons behind introducing Angular were to create fast, dynamic, and scalable web applications easily.

#### **Note: We can also develop dynamic websites and SPAs (Single Page Applications) using VanillaJS, and jQuery but by doing so, the development process becomes slower.**

### 24) What is Angular CLI?

Angular CLI is a short form for Angular Command Line Interface. It is a command-line interface to scaffold and build angular apps using node.js style modules.

To use Angular CLI, we have to install it by using the following **npm** command:

1. npm install @angular/cli@latest

**Following is a list of some useful commands which would be very helpful while creating angular projects:**

* **Creating New Project:** ng new
* **Generating Components, Directives & Services:** ng generate/g
* **Running the Project:** ng serve

### 25) What is lazy loading in Angular?

Lazy loading is one of the most powerful and useful concepts of Angular Routing. It makes the web pages easy to download by downloading them in chunks instead of downloading everything in a big bundle. Lazy loading facilitates asynchronously loading the feature module for routing whenever required using the property loadChildren.

See the following example where we are going to load both Employee and Order feature modules lazily.

**See the example:**

1. **const** routes: Routes = [
2. {
3. path: 'employees',
4. loadChildren: () => **import**('./employees/employees.module').then(module => module.EmployeesModule)
5. },
6. {
7. path: 'orders',
8. loadChildren: () => **import**('./orders/orders.module').then(module => module.OrdersModule)
9. },
10. {
11. path: '',
12. redirectTo: '',
13. pathMatch: 'full'
14. }
15. ];

### 26) What is Angular Router?

Angular Router is a mechanism that facilitates users to navigate from one view to the next as users perform application tasks. It follows the concept model of browser's application navigation.

### 27) What do you understand by the router imports?

The Angular Router, representing a particular component view for a given URL, is not part of Angular Core. It is available in a library named @angular/router, and we have to import the required router components. This process is called router imports.

See the following example of how we can import them in the app module:

1. **import** { RouterModule, Routes } from '@angular/router';

### 28) What do you understand by RouterOutlet and RouterLink?

A RouterOutlet is a directive from the router library that acts as a placeholder. It marks the spot in the template where the Router should display the components for that outlet. Router outlet is used as a component.

**Syntax:**

1. <router-outlet></router-outlet>

On the other hand, a RouterLink is a directive on the anchor tags that gives the router control over those elements. Since the navigation paths are fixed, you can assign string values to router-link directive as below,

**Syntax:**

1. **<h1>**Angular Router**</h1>**
2. **<nav>**
3. **<a** routerLink="/todosList" **>**List of todos**</a>**
4. **<a** routerLink="/completed" **>**Completed todos**</a>**
5. **</nav>**
6. **<router-outlet></router-outlet>**

### 29) What are the different router events used in Angular Router?

During each navigation, the Router emits navigation events through the Router.events property. It allows us to track the lifecycle of the route.

**Following is the list of different router events in sequence:**

* NavigationStart
* RouteConfigLoadStart
* RouteConfigLoadEnd
* RoutesRecognized
* GuardsCheckStart
* ChildActivationStart
* ActivationStart
* GuardsCheckEnd
* ResolveStart
* ResolveEnd
* ActivationEnd
* ChildActivationEnd
* NavigationEnd
* NavigationCancel
* NavigationError

### 30) What do you understand by the RouterLinkActive?

The RouterLinkActive is a directive used to toggle CSS classes for active RouterLink bindings based on the current RouterState. i.e., the Router will add CSS classes when this link is active and remove them when the link is inactive.

For example, you can add them to RouterLinks as follows:

1. <h1>Angular Router</h1>
2. <nav>
3. <a routerLink="/todosList" routerLinkActive="active">List of todos</a>
4. <a routerLink="/completed" routerLinkActive="active">Completed todos</a>
5. </nav>
6. <router-outlet></router-outlet>

### 31) What do you understand by the RouterState?

The RouterState is a tree of activated routes. Every node in this tree knows about the "consumed" URL segments, the extracted parameters, and the resolved data. We can access the current RouterState from anywhere in the application by using the Router service and the routerState property.

1. @Component({templateUrl:'template.html'})
2. **class** MyComponent {
3. constructor(router: Router) {
4. **const** state: RouterState = router.routerState;
5. **const** root: ActivatedRoute = state.root;
6. **const** child = root.firstChild;
7. **const** id: Observable<string> = child.params.map(p => p.id);
8. //...
9. }
10. }

### 32) What is HttpClient, and what are the advantages of it?

Most front-end applications use either XMLHttpRequest interface or the fetch() API to communicate with backend services over HTTP protocol. For the same purpose, Angular provides a simplified client HTTP API known as HttpClient. This is based on top of XMLHttpRequest interface. This HttpClient is available in the @angular/common/http package, which you can import in your root module as follows:

1. **import** { HttpClientModule } from '@angular/common/http';

**Following are some of the crucial advantages of HttpClient:**

* HttpClient contains testability features.
* It provides typed request and response objects.
* It can intercept requests and responses.
* It supports Observalbe APIs.
* HttpClient also supports streamlined error handling.

### 33) By default, Angular uses client-side rendering for its applications. Is it possible to make an Angular application to render on the server-side?

Yes, it is possible to make an Angular application to render on the server-side. Angular provides a technology called Angular Universal that can be used to render applications on the server-side.

The crucial advantages of using Angular Universal are as follows:

* Making an Angular application render on the server-side can provide a better user experience. By using this, first-time users can instantly see a view of the application. So, it can be used to provide better UI.
* It can lead to a better SEO for your application. The reason is that many search engines expect pages in plain HTML. So, Angular Universal can ensure that your content is available on every search engine, and it is good for better SEO.
* The server-side rendered applications load faster than normal pages. It is because the rendered pages are available to the browser sooner.

### 34) What is the best way to perform Error handling in Angular?

Error is when the request fails on the server or fails to reach the server due to network issues. In this condition, HttpClient returns an error object instead of a successful response. To resolve this issue, we must handle the component by passing the error object as a second callback to the subscribe() method.

**See the following example to understand how we handle in the component:**

1. fetchUser() {
2. **this**.userService.getProfile()
3. .subscribe(
4. (data: User) => **this**.userProfile = { ...data }, // success path
5. error => **this**.error = error // error path
6. );
7. }

You can write an error message to give the user some meaningful feedback instead of displaying the raw error object returned from HttpClient.

### 35) What do you understand by Angular bootstrapping?

Angular bootstrapping is nothing but to allow developers to initialize or start the Angular application. Angular supports two types of bootstrapping:

* Manual bootstrapping
* Automatic bootstrapping

**Manual bootstrapping:** Manual bootstrapping provides more control to developers and facilitates them regarding how and when they need to initialize the Angular app. It is useful when professionals wish to perform other tasks and operations before Angular compiles the page.

**Automatic bootstrapping:** As the name specifies, automatic bootstrapping is started automatically to start the Angular app. The developers need to add the ng-app directive to the application's root if they want Angular to bootstrap the application automatically. Angular loads the associated module once it finds the ng-app directive and, further, compiles the DOM.

### 36) What is the digest cycle process in Angular?

The digest cycle process in Angular is the process that is used to monitor the watchlist to track changes in the watch variable value. There is a comparison between the present and the previous versions of the scope model values in each digest cycle.

### 37) What are the key differences between a Component and a Directive in Angular?

A Component is a directive that uses shadow DOM to create encapsulated visual behavior. Usually, components are used to create UI widgets by breaking up the application into smaller parts. In short, we can say that a component (@component) is a directive-with-a-template.

**A list of the major differences between a Component and a Directive in Angular:**

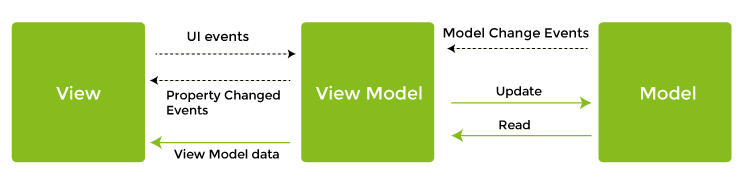
|  |  |
| --- | --- |
| **Component** | **Directive** |
| Components are generally used for creating UI widgets. | Directives are generally used for adding behavior to an existing DOM element. |
| We use @Component meta-data annotation attributes to register a component. | We use @Directive meta-data annotation attributes to register directives. |
| It is used to break up the application into smaller parts called components. | It is used to design re-usable components. |
| Only one component is allowed to be used per DOM element. | Multiple directives are allowed to be used per DOM element. |
| @View decorator or templateurl/template is mandatory in a component. | A Directive doesn't use View. |
| A component is used to define pipes. | In a directive, it is not possible to define Pipes. |

### 38) What do you understand by Angular MVVM architecture?

The MVVM architecture or **Model-View-ViewModel** architecture is a software architectural pattern that provides a facility to developers to separate the development of the graphical user interface (the View) from the development of the business logic or back-end logic (the Model). By using this architecture, the view is not dependent on any specific model platform.

The Angular MVVM architecture consists of the following three parts:

* Model
* View
* ViewModel



**Model:** The Model consists of the structure of an entity and specifies the approach. In simple words, we can say that the model contains data of an object.

**View:** The View is the visual layer of the application. It specifies the structure, layout, and appearance of what a user sees on the screen. It displays the data inside the Model, represents the model, and receives the user's interaction with the view in the form of mouse clicks, keyboard input, screen tap gestures, etc., and forwards these to the ViewModel via the data binding properties. In Angular terms, the View contains the HTML template of a component.

**ViewModel:** The ViewModel is an abstract layer of the application. It is used to handle the logic of the application. It also manages the data of a model and displays it in the view. View and ViewModel are connected with two-way data-binding. If you make any changes in the view, the ViewModel takes a note and changes the appropriate data inside the model.

### 39) What is the purpose of AsyncPipe in Angular?

The AsyncPipe is used to subscribe to an observable or promise and return the latest value it has emitted. When a new value is emitted, the pipe marks the component that has been checked for changes.

See the following example where a time observable continuously updates the view for every 2 seconds with the current time.

**Example:**

1. @Component({
2. selector: 'async-observable-pipe',
3. template: `<div><code>observable|async</code>:
4. Time: {{ time | async }}</div>`
5. })
6. export **class** AsyncObservablePipeComponent {
7. time = **new** Observable(observer =>
8. setInterval(() => observer.next(**new** Date().toString()), 2000)
9. );
10. }

### 40) What do you understand by services in Angular?

In Angular, services are singleton objects that get instantiated only once during the lifetime of an application. An Angular service contains methods that are used to maintain the data throughout the life of an application. Angular services are used to organize as well as share business logic, models, or data and functions with various components of an Angular application.

Angular services offer some functions that can be invoked from an Angular component, such as a controller or directive.

### 41) What is the key difference between a constructor and ngOnInit?

Constructor is a default method in TypeScript classes that are normally used for the initialization purpose. On the other hand, the ngOnInit is specifically an Angular method and is used to define Angular bindings. Even though constructors are getting called first, it is always preferred to move all of your Angular bindings to the ngOnInit method.

See the following example how we can use ngOnInit by implementing OnInit interface as follows:

1. export **class** App **implements** OnInit{
2. constructor(){
3. //called first time before the ngOnInit()
4. }
5. ngOnInit(){
6. //called after the constructor and called  after the first ngOnChanges()
7. }
8. }

### 42) What do you understand by observable and observer in Angular?

**Observable:** An observable is a unique object just like a promise that that is used to manage async code. Observables are not part of the JavaScript language so the developers have to rely on a popular Observable library called RxJS. The observables are created using the new keyword.

See a simple example of observable to understand it better:

1. **import** { Observable } from 'rxjs';
2. **const** observable = **new** Observable(observer => {
3. setTimeout(() => {
4. observer.next('This is a message from Observable!');
5. }, 1000);
6. });

**Observer:** Any object that has to be notified when the state of another object changes is called an observer. An observer is an interface for push-based notifications delivered by an Observable.

**See the structure of an observer:**

1. **interface** Observer<T> {
2. closed?: **boolean**;
3. next: (value: T) => **void**;
4. error: (err: any) => **void**;
5. complete: () => **void**;
6. }

The handler that implements the observer interface for receiving observable notifications is passed as a parameter for observable as follows:

1. myObservable.subscribe(myObserver);

#### **Note: If you don't use a handler for a notification type, the observer ignores notifications of that type.**

### 43) How do you categorize data binding types in Angular?

In Angular, we can categorize data binding types in three categories distinguished by the direction of data flow. These data binding categories are:

* From the source-to-view
* From view-to-source
* View-to-source-to-view

Let's see their possible binding syntax:

|  |  |  |
| --- | --- | --- |
| **Data direction** | **Syntax** | **Type** |
| From the source-to-view(One-way data binding) | 1. {{expression}} 2. [target]="expression" 3. bind-target="expression" | Interpolation, Property, Attribute, Class, Style |
| From view-to-source(One-way data binding) | 1. (target)="statement" 2. on-target="statement" | Event |
| View-to-source-to-view(Two-way data binding) | 1. [(target)]="expression" 2. bindon-target="expression" | Two-way data binding |

### 44) What is multicasting in Angular?

Multicasting or Multi-casting is the practice of broadcasting to a list of multiple subscribers in a single execution.

Let's take a simple example to demonstrate the multi-casting feature:

1. var source = Rx.Observable.from([1, 2, 3]);
2. var subject = **new** Rx.Subject();
3. var multicasted = source.multicast(subject);
4. // These are, under the hood, `subject.subscribe({...})`:
5. multicasted.subscribe({
6. next: (v) => console.log('observerA: ' + v)
7. });
8. multicasted.subscribe({
9. next: (v) => console.log('observerB: ' + v)
10. });

### 45) What do you understand by Angular Material?

Angular Material is a UI component library that is used by professionals to develop consistent, attractive, and completely functional websites, web pages, and web applications. It follows the modern principles of web designing, such as graceful degradation and browser probability, and is capable of doing a lot of fascinating things in website and application development.

### 46) What is lazy loading in Angular? Why is it used?

In Angular, the by default tendency of NgModules is eagerly loaded. It means that as soon as the app loads, all the NgModules are loaded, whether or not they are immediately necessary. That's why lazy loading is required. Lazy loading is mandatory for large apps with lots of routes. This design pattern makes the app load NgModules when they are only required. Lazy loading helps keep initial bundle sizes smaller, which in turn helps decrease load times.

### 47) What is the use of Angular filters? What are its distinct types?

Filters are an essential part of Angular that helps in formatting the expression value to show it to the users. We can easily add filters to services, directives, templates, or controllers. We can also create personalized filters as per requirements. These filters allow us to organize the data in such a way that only the data that meets the respective criteria are displayed. Filters are placed after the pipe symbol ( | ) while used in expressions.

**A list of various types of filters used in Angular:**

* **currency:** It is used to convert numbers to the currency format.
* **filter:** It is used to select a subset containing items from the given array.
* **date:** It is used to convert a date into a necessary format.
* **lowercase:** It is used to convert the given string into lowercase.
* **uppercase:** It is used to convert the given string into uppercase.
* **orderBy: It** is used to arrange an array by the given expression.
* **json:** It is used to format any object into a JSON string.
* **number:** It is used to convert a numeric value into a string.
* **limitTo:** It is used to restrict the limit of a given string or array to a particular number of elements or strings.

### 48) When do we use a directive in Angular?

If you create an Angular application where multiple components need to have similar functionalities, you have to do it by adding this functionality individually to every component. This is not a very easy task. Directives are used to cope up with this situation. Here, we can create a directive with the required functionality and then import the directive to components that require this functionality.

### 49) What are the different types of directives in Angular?

There are mainly three types of directives in Angular:

**Component Directives:** The component directives are used to form the main class in directives. To declare these directives, we have to use the @Component decorator instead of @Directive decorator. These directives have a view, a stylesheet and a selector property.

**Structural directives:** These directives are generally used to manipulate DOM elements. The structural directive has a ' \* ' sign before them. We can apply these directives to any DOM element.

**Following are some example of built-in structural directives:**

**\*ngIf Structural Directive:** \*ngIf is used to check a Boolean value and if it's truthy, the div element will be displayed.

1. **<div** \*ngIf="isReady" class="display\_name"**>**
2. {{name}}
3. **</div>**

**\*ngFor Structural Directive:** \*ngFor is used to iterate over a list and display each item of the list.

1. **<div** class="details" \*ngFor="let x of details" **>**
2. **<p>**{{x.name}}**</p>**
3. **<p>** {{x.address}}**</p>**
4. **<p>**{{x.age}}**</p>**
5. **</div>**

**Attribute Directives:** The attribute directives are used to change the look and behavior of a DOM element. Let's create an attribute directive to understand it well:

**This is how we can create a custom directive:**

Go to the command terminal, navigate to the directory of the angular app and type the following command to generate a directive:

1. ng g directive yellowBackground

This will generate the following directive. Manipulate the directive to look like this:

1. **import** { Directive, ElementRef } from '@angular/core';
2. @Directive({
3. selector: '[appYellowBackground]'
4. })
5. export **class** YellowBackgroundDirective {
6. constructor(el:ElementRef) {
7. el.nativeElement.style.backgroundColor = "yellow";
8. }
9. }

Now, you can easily apply the above directive to any DOM element:

1. **<p** appYellowBackground**>**Hello JavaTpoint**</p>**

### 50) What are string interpolation and property binding in Angular?

String interpolation and property binding are parts of data-binding in Angular. Data-binding is a feature of Angular, which is used to provide a way to communicate between the component (Model) and its view (HTML template). There are two ways of data-binding, one-way data binding and two-way data binding. In Angular, data from the component can be inserted inside the HTML template. Any changes in the component will directly reflect inside the HTML template in one-way binding, but vice-versa is not possible. On the other hand, it is possible in two-way binding.

String interpolation and property binding both are examples of one-way data binding. They allow only one-way data binding.

**String Interpolation:** String interpolation uses the double curly braces {{ }} to display data from the component. Angular automatically runs the expression written inside the curly braces. For example, {{ 5+5 }} will be evaluated by Angular, and the output will be 10. This output will be displayed inside the HTML template.

**Property Binding:** Property binding is used to bind the DOM properties of an HTML element to a component's property. In property binding, we use the square brackets [ ] syntax.

### 51) Is it possible to make an angular application to render on the server-side?

Yes, we can make an angular application to render on the server-side. Angular provides a technology Angular Universal that makes you able to render applications on the server-side.

**Following are the benefits of using Angular Universal:**

**Better User Experience:** It enables users to see the view of the application instantly.

**Better SEO:** Angular Universal ensures that the content is available on every search engine leading to better SEO.

**Load Faster:** Angular Universal ensures that the render pages available to the browsers sooner to make the loading faster server-side application loads faster.

### 52) What is Dependency Injection in Angular?

Dependency injection is an application design pattern that is implemented by Angular. It is used to form the core concepts of Angular. Dependencies are services in Angular which have some specific functionality. Various components and directives in an application can need these functionalities of the service. Angular provides a smooth mechanism by which these dependencies are injected into components and directives.

### 53) Can you demonstrate navigation between different routes in an Angular application?

You can demonstrate the navigation between different routes in an Angular app in the following way. See the following code to demonstrate navigation in an Angular app named "My First App."

1. **import** from "@angular/router";
2. .
3. .
4. .
5. @Component({
6. selector: 'app-header',
7. template: `
8. <nav **class**="navbar navbar-light bg-faded">
9. <a **class**="navbar-brand" (click)="goHome()">My First App</a>
10. <ul **class**="nav navbar-nav">
11. <li **class**="nav-item">
12. <a **class**="nav-link" (click)="goHome()">Home</a>
13. </li>
14. <li **class**="nav-item">
15. <a **class**="nav-link" (click)="goSearch()">Search</a>
16. </li>
17. </ul>
18. </nav>
19. })
20. **class** HeaderComponent {
21. constructor(**private** router: Router) {}
22. goHome() {
23. **this**.router.navigate(['']);
24. }
25. goSearch() {
26. **this**.router.navigate(['search']);
27. }
28. }

### 54) What is the difference between Angular and Backbone.js?

Following are the various notable differences between Angular and Backbone.js:

|  |  |  |
| --- | --- | --- |
| **Comparison Parameter** | **Angular** | **Backbone.js** |
| **Architecture** | Angular works on the MVC architecture and makes use of two-way data binding for driving application activity. | Backbone.js makes use of the MVP architecture and doesn't provide any data binding process. |
| **Type** | Angular is an open-source JavaScript-based front-end web application framework that extends HTML with new attributes. | Backbone.js is a lightweight JavaScript library that uses a RESTful JSON interface and MVP framework. |
| **Data Binding** | Angular is a little bit complex because it uses a two-way data binding process. | On the other hand, Backbone.js has a simple API because it doesn't have any data binding process. |
| **DOM** | Angular's main focus is on valid HTML and dynamic elements that imitate the underlying data for rebuilding the DOM as per the specified rules and then work on the updated data records. | Backbone.js follows the direct DOM manipulation approach for representing data and application architecture changes. |
| **Performance** | Because of its two-way data binding functionality, Angular provides powerful performance for both small and large projects. | Backbone.js is quite a significant upper hand in performance over Angular in small data sets or small web pages. It is not recommended for larger web pages or large data sets due to the absence of any data binding process. |
| **Templating** | Angular supports templating via dynamic HTML attributes. You can add them to the document to develop an easy to understand application at a functional level. | Backbone.js uses Underscore.js templates that aren't fully-featured as Angular templates. |
| **Testing Approach** | The testing approach is lengthy for Angular because it is preferred for building large applications. It uses unit testing. | The testing approach is completely different for Backbone.js because it is ideal for developing smaller webpages or applications. |
| **Community Support** | The angular framework is developed and maintained by Google, so it receives great community support. Here, extensive documentation is available. | Backbone.js also receives a good level of community support, but it only documents on Underscore.js templates, not much else. |