Top 100+ Angular Interview Questions and Answers in 2024

Atif Khan | Last updated on April 9, 2024 | 8.9K Views

Previous Next

Angular is one of the most in-demand skills in today's job market, but it is also very difficult to get a good grasp of the entire framework, especially from an interview perspective. Keeping that in mind, we have come up with a list of well-curated Angular interview questions that will help you ace your Angular interviews.

Table of Contents

Show More

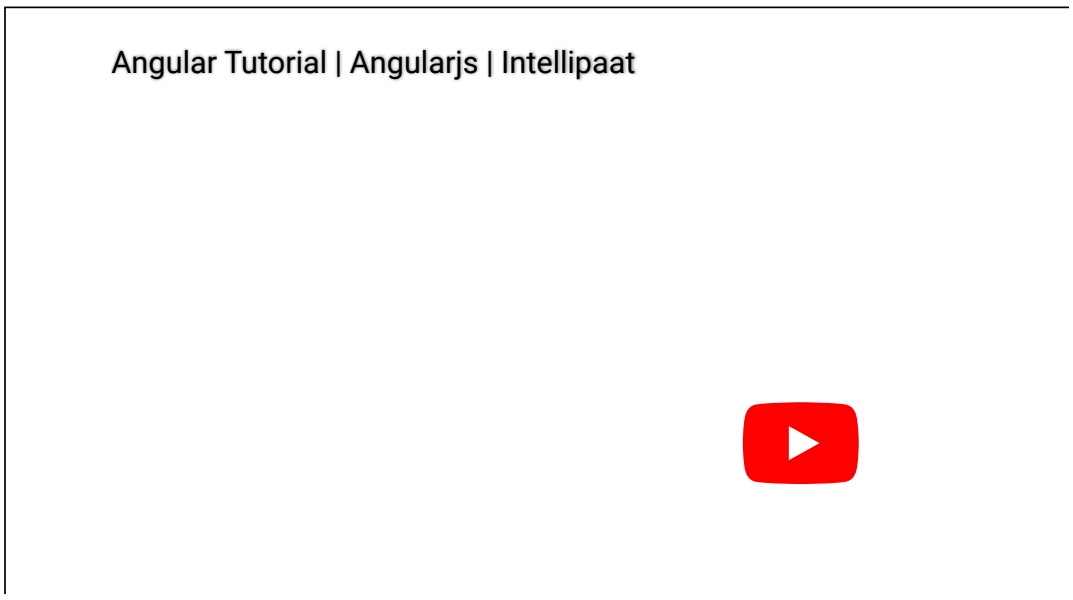
Did You Know?

Brought to You by Google: Angular was created and maintained by Google. They efficiently check and resolve the issues and bugs within the framework
Amazing Angular Router: Angular acts as a server-side router that doesn't check for hashes for location, unlike other frameworks like React.
Zero Tedious Work: To create a new component, you can simply copy-paste the code into a new code without the need to write the code again.

Angular Interview Questions and Answers

Angular is one of the most popular JavaScript single-page application frameworks. It is also quite vast and difficult to navigate. Even more difficult is to get answers to the interview questions on Angular as it is very confusing to know which parts of the Angular framework you should familiarize yourself with. Hence, we have come up with a compilation of the most frequently asked Angular interview questions and answers.

Check out this Angular tutorial video designed to help you better understand the Angular framework:



Angular Basic Interview Questions for Freshers

Let's start with the basic Angular interview questions and answers for freshers.

1. What is Angular?

Angular is an open-source web application development framework created by Google. It is used to build front-end and single-page applications that run on JavaScript. It is a full-fledged framework, i.e, it takes care of many aspects of front–end web applications such as HTTP requests, routing, layout, forms, reactivity, validation, etc.

2. What are the technologies used in Angular?

Angular is a modern front-end JavaScript framework developed by Google. Angular itself makes use of several technologies for several reasons to accomplish certain tasks easily as well as to allow developers to have a better experience while developing applications with it. Angular uses TypeScript, which is a superscript of JavaScript. So, any valid JavaScript is a valid TypeScript. However, TypeScript allows us to write JavaScript as a strongly typed language, and we can define our own types as well, which makes catching bugs much easier. It also uses RxJS, which allows developers to better deal with asynchronous operations.

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

Before JavaScript-based client-side frameworks, the way dynamic websites worked was by taking a template that is nothing but HTML code with spaces left empty for feeding data and content into those templates. This data was usually fetched from a database. After combining the template and data, we would serve the generated HTML content back to the user. As you can see, it was a bit complicated, and in some cases, it took a lot of processing.

To overcome these issues, people came up with another approach in which they send the necessary data to render a page from their **web servers** to the web browsers and let JavaScript combine this data with a predefined template. Even mobile phones are now powerful enough to do this kind of processing, so the servers can now just send the data to a client over the internet in a recognizable format, i.e., JSON, XML, etc. This drastically reduces the processing done on the servers and improves performance.

Want to learn Angular 6? Check out our [Angular 6 Tutorial](#)!

4. How does an Angular application work?

The working of Angular is based on its components. So the working of the Angular application starts with the configuration file **ANGULAR.JSON**. The builder refers to this file to find the paths, configurations and the main file. Now the process finally starts. Next comes the **MAIN.TS** file that acts as the entry point for the configuration file. It basically helps in creating the browser environment that enables it to run the application.

An Angular application operates according to the principles of a client-side web framework. Upon loading the application in a web browser, the following sequence of steps takes place:

- Initialization: The Angular framework initiates the application by bootstrapping the root component, which serves as the entry point.
- Component Rendering: Angular renders components based on their predefined templates. Each component contains a template that describes the structure and layout of the user interface (UI).
- Data Binding: Angular enables data binding, which facilitates the synchronization of data between the component's model and the view. This synchronization allows for dynamic updates in real time, ensuring that the view consistently and accurately represents the current state of the data.
- Component Interaction: Components communicate with one another through input and output properties, as well as services, facilitating the exchange of data and functionality across different parts of the application.
- Event Handling: Angular manages user interactions, such as button clicks or form submissions, by capturing and responding to events. These events trigger specific actions or updates within the application.
- Dependency Injection: Angular utilizes dependency injection to handle the application's dependencies and promote modularity. This simplifies the integration of services and reusable code modules that provide specific functionality to components.
- Routing: Angular incorporates a router module that enables navigation within the application. Routes are defined, determining which components are displayed based on the current URL or user actions.
- Compilation and Bundling: Before deployment, an Angular application is typically compiled and bundled. This process optimizes performance by reducing file sizes and merging multiple files into a single bundle.
- Deployment: Following bundling, the application can be deployed to a web server or hosting service, making it accessible to users.

5. What is TypeScript?

The TypeScript feature offered by Angular is preferred by a majority of Front-End Developers. TypeScript helps in efficiently detecting bugs and helps in easy compilation by its automatic populating functionality. Also, it offers rich interfaces,

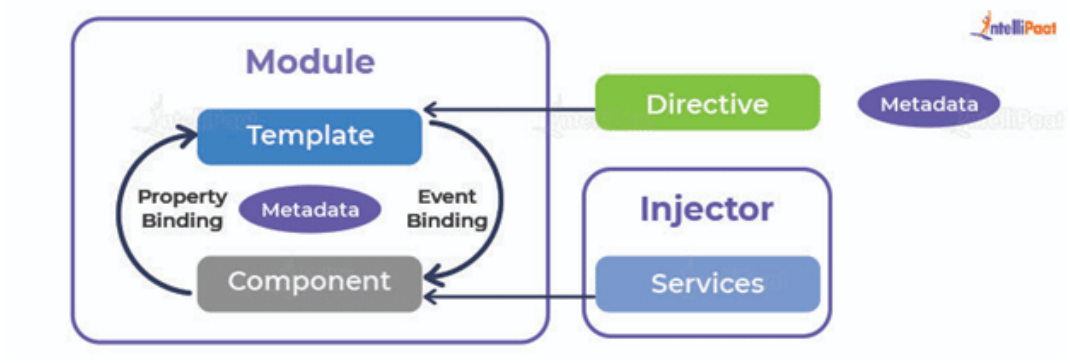
access modifiers, hybrid types etc. All these combined lead to reduction in the developing time.

Check out the [Web Developer Technical Interview Questions](#).

6. Write a pictorial diagram of Angular architecture?

The architecture of Angular comprises the following elements. The pictorial representation of the same is given below:

- Components and Templates
- Ng Modules
- Metadata
- Data Binding
- Directives
- Services
- Dependency Injection



7. What is metadata?

Using metadata is how we tell Angular how to process a class. When we use a component, it acts as a class unless we tell that it's a component, and we do this with the help of metadata. Metadata is attached in TypeScript using a decorator. Decorators are functions that know the configuration of classes and how they are supposed to work.

Check out our [PHP Interview Questions](#) to ace your next interview!

8. What is the difference between constructor and ngOnInit?

The difference between constructor and ngOnInit is given below:

Basis	Constructor	ngOnInit
Objective.	The objective of a Constructor is to start class members	**ngOnInit** is used in case of startup/announcement and avoids things to work in builders.
Usage	A Constructor should be used to set up Dependency Injection, Initialization of class fields, etc.	**ngOnInit** is used to write the work code that executes as soon as the class is instantiated.

9. How is Dependency Hierarchy formed?

In Angular, the formation of a Dependency Hierarchy is established through the Angular Dependency Injection (DI) system. The DI system manages the dependencies between different components within an Angular application.

The process of forming a Dependency Hierarchy in Angular involves the following steps:

- **Dependency Definition:** Components, services, or other Angular constructs declare their dependencies in their respective constructor parameters. These

dependencies are specified using TypeScript types or Angular-specific decorators.

- **Provider Registration:** In Angular, providers assume the responsibility of generating and overseeing instances of dependencies. These providers are enlisted within the dependency injection container of the Angular application, which serves as a repository for available providers.
- **Dependency Resolution:** When a component or service is requested, Angular's DI system resolves its dependencies by examining the constructor parameters. It then looks up the corresponding providers from the dependency injection container.

Hierarchical Dependency Injection: Angular's DI system follows a hierarchical approach for resolving dependencies. Each component or service has its own injector, which contains providers specific to that component or service. If a requested dependency is not found in the current injector, Angular recursively searches the parent injectors until it finds a matching provider or reaches the top-level injector.

10. What is the purpose of the async pipe?

The async pipe's purpose is to mark the components that need to be checked for changes. It subscribes to an Observable or Promise and returns the latest value it has emitted. Once this new value is emitted, the components are marked by the async pipe. When any component is destroyed, the async pipe detaches or unsubscribes automatically. Similarly, if the expression reference of the component changes, the async pipe detaches or unsubscribes from the old Observable or Promise and subscribes to a new one.

11. What is the option to choose between inline and external templates?

Usually, inline templates are used for small codes and external templates are used for comparatively bigger views. However, the choice of inline or external templates is sometimes based on organizational policy, situations etc.

12. What is the purpose of the 'ngFor' directive?

The purpose of the 'ngFor' directive in Angular is to iterate over a collection or an array and generate repetitive HTML elements or components based on each item in the collection. It allows for dynamic rendering of content by repeating a template block for each item in the specified collection. The 'ngFor' directive is commonly used to display lists, tables, or any other structured data where repetitive rendering is required. It provides a convenient way to handle and manipulate collections of data in Angular templates.

13. What is the purpose of the ngIf directive?

The purpose of the ngIf directive is to remove or recreate a part of the DOM tree in alignment with an expression. If the **ngIf** directive finds the expression being evaluated to be false, the element is removed from the tree, else a matching element is inserted into the DOM tree.

14. What happens if you use the script tag inside the template?

If we use script tag inside template, Angular marks the value as unsafe and automatically initiates the process of sanitizing it. This eradicates the script tag, but the content is kept safe, i.e. the text element. This entire process results in eliminating the risk of injection attacks.

15. What are template expressions?

A template expression in Angular is an expression that is represented in double curly braces **'{{ }}'** and produces a value. The template expression is executed by Angular and is assigned to a property of a binding target. Now, the binding target can be any of these- an HTML element, a directive or even a component.

16. What are template statements?

The methods or properties in Angular that are used in HTML in response to user events are called template statements. These template statements allow your application to engage users through actions like dynamic content display or form submissions etc.

17. What is the difference between Angular and AngularJS?

Following are some of the major and significant differences between **Angular and AngularJS**:

Features	Angular	AngularJS
Architecture	It makes use of directives and components	It supports the Model-View-Controller or MVC model
Language	It uses TypeScript language, a superset of JavaScript that is typed statistically	It uses JavaScript, a dynamically typed language
Expression Syntax	Angular uses **()** to bind an event while **[]** to bind a property	It requires professionals to use the correct **ng** directive to bind a property or an event
Mobile Support	Angular offers mobile support	Unlike Angular, AngularJS does not offer mobile support
Routing	It uses **@RouteConfig{(...)}**	It uses **$routeprovider.when()**
Dependency Injection	It supports hierarchical dependency injection, along with a unidirectional tree-based change direction	It does not support dependency injection
Structure	Its simplified structure makes it easy for professionals to develop and maintain large applications easily	It is comparatively less manageable

18. What are some advantages of using Angular?

Using Angular has several advantages, which are listed below:

- Angular is built using TypeScript, which allows developers to write strongly typed code that will get transpiled into JavaScript. The benefits of a strongly typed code are that it is easy to read, maintainable, and less prone to errors. Also, it provides better tooling with type hints and code completion.
- Angular allows us to separate our code into modules, which can be used to wrap functionalities related to a specific task such as HTTP communication, data validation, routing, etc.
- Angular has a large ecosystem of tools, libraries, frameworks, plugins, etc. that make the whole development experience much faster and enjoyable. These tools and libraries include Angular CLI, RxJS, NgRx, etc.

19. How do you categorize data binding types?

To categorize data binding in Angular, it is divided into various types. Data Binding in Angular is categorized into following types:

- One Way Data Binding
- Two-Way Data Binding

In One way data binding, the changes in the state affect the view from component to view template. On the contrary it is also possible that the change in the view affects the state by changing it from view template to component.

Now, coming to Two-way Data Binding, the changes in the view can lead to change in the model. Similarly, any changes in the model can change the view from component to view template.

The various types of Two-Way Data Binding are:

- **Interpolation**
- Property binding
- Class binding
- Style binding
- Attribute binding
- Event binding
- Two-way binding

20. What is a parameterized pipe?

In Angular, pipes are used to transform raw data into the required format before the final display to the end-users. These pipes are broadly categorized into 2 categories:

1- Built-in Pipes,

2- Custom Pipes

Built-in Pipes are further divided into 2 types – Parameterized Pipes and Chaining Pipes.

Parameterized Pipes refer to the pipes that carry parameters. We can use these pipes to pass n number of parameters by giving a colon (:) in the command.

21. What are custom elements?

A custom element is used to extend HTML, wherein you can define a tag whose content is created and controlled by JavaScript code.

22. Do I need to bootstrap custom elements?

No, you do not need to bootstrap custom elements as they are bootstrapped automatically when added to the DOM. Also, the custom elements in Angular get automatically destroyed when removed from the DOM.

23. How do you define typings for custom elements?

Defining typings for custom elements in Angular can be done by using NgElement and WithProperties exported from @angular/elements. The following component is a simple container with input property:

```
@Component(…)
class MyDialog {
@Input() content: string;
}
```

24. Explain how custom elements work internally?

Let us understand, in steps, the internal working of custom elements:

- **Registration of the custom elements:** Angular registers custom elements using the createCustomElement() function. This function converts a component into a class that can be registered with the browser as a custom element.
- **Addition of the custom element to DOM**: The custom element is added to DOM in a similar manner as it is in HTML.
- **Browser instantiates component based class:** Once the custom element is added to the DOM, an instance of the registered class is created by the browser and added to the DOM.
- **Data binding and Change detection:** In the final step, the created instance enables data binding and change detection. The template content is rendered using the component and DOM data.

25. How to transfer components to custom elements?

There are two important steps to be followed in order to transfer components to custom elements:

- **Creating a custom element class:** As a first step, build a custom element class using the createCustomElement() function provided by Angular. The function converts an Angular component (including its dependencies) to a custom element. The NgElementConstructor interface is implemented through this conversion which, in turn, creates a constructor class that is used for producing a self-bootstrapping instance.
- **Registering element class with browser:** The customElements.define() function is used to register the configured constructor, and its associated custom-element tag with the browser's CustomElementRegistry.

26. What are the mapping rules between Angular components and custom elements?

The important mapping rules between Angular components and custom elements are given below:

- The component input properties are parsed with the corresponding attributes for the custom element using the createCustomElement() API.
- TThe Component outputs are dispatched as HTML Custom Events and have the name of the custom event that also matches the output name.

27. How are observables different from promises?

Although both promises and observables are used to handle asynchronous requests in JavaScript, they work in very different ways. Promises can only handle a single event at a time, while observables can handle a sequence of asynchronous events over a period of time. Observables also provide us with a wide variety of operators that allow us to transform data flowing through these observables with ease.

A promise is just a way to wrap asynchronous operations so that they can be easily used, while an observable is a way to turn asynchronous operations into a stream of data that flows from a publisher to a subscriber through a well-defined path with multiple operations transforming the data along the way.

28. What is a custom pipe?

A custom pipe is a feature of angular, that enables you to create and use your own reusable data transformation functions in templates. Pipes are used to transform data before displaying it in the user interface.

A custom pipe is specifically created by the developer to perform a specific data transformation or manipulation task. It encapsulates a function that takes an input value, applies a transformation, and returns the transformed output value. Custom pipes are defined as classes in Angular and decorated with the **@Pipe decorator.**

To use a custom pipe, you can include it in your Angular application's module and then utilize it in templates by piping the data through the pipe in the template expression. The output of the pipe can be directly used for display or further processing in the template.

Custom pipes offer a way to encapsulate and reuse common data transformations across different components and templates. They promote code modularity, reusability, and readability by separating the data transformation logic from the component code.

Commonly, custom pipes are defined as follows:

```
import { Pipe, PipeTransform } from '@angular/core';
@Pipe({name: 'Pipename'})

export class Pipeclass implements PipeTransform {
   transform(parameters): returntype { }
}
```

here,

'Pipename' is the name of the pipe.

'Pipeclass' is the name of the class assigned to the custom pipe.

'Transform' is the function to work with the pipe.

'Parameters' are the parameters which are passed to the pipe.

'Returntype' refers to the type of value that the pipe returns.

29. Give an example of a custom pipe?

Let us now have a look at the example below where we have declared a custom pipe to convert a number to its square in the Angular application. Please note that we are using the **ng generate** pipe command to create this custom pipe.

ng generate pipe square

```
// Output
CREATE src/app/square.pipe.spec.ts (187 bytes)
CREATE src/app/square.pipe.ts (217 bytes)
UPDATE src/app/app.module.ts (2931 bytes)
```

30. What is the difference between pure and impure pipe?

Basis	Pure Pipe	Impure Pipe
Meaning	A pure pipe is called when a change in the value or the parameters passed to a pipe is detected by Angular.	In every change detection cycle, regardless of whether there is any change in the value or parameter passed, an impure pipe is invoked.
Syntax	@Pipe({ name: 'filterPipe', pure: true }) export class FilterPipe {}	@Pipe({ name: 'filterPipe', pure: false }) export class FilterPipe
Shareability	Pure pipes can be shared across various usages and that too without having any effect on the output result	Impure Pipes cannot be shared because they might affect the internal state from outside

Determination of Output change.	Input values or parameters can determine the output value or the change in it.	Input values cannot determine the output value or the change in it.

Angular Interview Questions for Experienced Professionals (2 to 5 Years)

31. What is a bootstrapping module?

In Angular, the root module used for bootstrapping or launching the application is known as the 'Bootstrapping Module'. A Bootstrapping Module is present in every Angular app and it is stored in the AppModule class. Infact, the Bootstrapping module is also called the AppModule.

32. Explain how to use HttpClient with an example?

The following are the generally followed steps to use the **HttpClient :**

- Firstly, start by creating an HttpClient instance.
- Next, create an instance of one of the methods.
- Command HttpClient to execute the method.
- After the execution, read the response.
- Finally, release the connection.
- And deal with the response.

33. What are lifecycle hooks in Angular?

When building an Angular app, there are times when we need to execute some code at some specific event—such as when a component is initialized or displayed on the screen or when the component is being removed from the screen. This is what lifecycle hooks are used for. For example, if we have some event listener attached to an HTML element in a component, such as a button click or form submission, we can remove that event listener before removing the component from the screen, just like we can fetch some data and display it on the screen in a component after the component is loaded on the screen. To use a lifecycle hook, we can override some methods on a component, such as **ngOnInit** or **ngAfterViewInit**. These methods, if available on a component, will be called by Angular automatically. This is why these are called lifecycle hooks.

34. What are templates?

Angular templates are written using HTML that includes attributes and elements that are specific to Angular. The templates are further combined with the data from the controller and the model, which can be rendered to offer the user a dynamic view.

Angular Interview Questions for 2 to 3 Years Experience

35. How can you read the full response?

In order to read the full response in Angular, the following code should be used:

```
getUserResponse(): Observable<HttpResponse> {
  return this.http.get(
    this.userUrl, { observe: 'response' });
}
```

36. How do you perform Error handling?

In Angular, error handling can be done by writing a function using HttpClient along with catchError from RxJS.To handle errors, Angular's HttpClient parses JSON responses and returns a JavaScript object in the **OBSERVABLES**.

37. What is content projection?

In Angular, Content projection refers to a pattern where you can insert, or project, the content you want to use inside a different component. For example, consider a Card component which can accept the content provided by another component.

38. What is the difference between PROMISE & OBSERVABLES?

Basis	OBSERVABLES	PROMISE
Values	Observables have the capacity to emit multiple values over a given period of time.	A promise emits only a single value over a given period of time.
Execution	They can be executed only when subscribed using subscribe() method.	Promises can be executed as soon as they get created.
Cancellation	Observables are cancellable as they consist of subscriptions that can be cancelled using the unsubscribe() method.	Promises once executed are non cancellable.
Operations	Can provide operations like map for forEach, filter, reduce, retry, and retryWhen operators.	Promises do not offer any operations.
Errors	An observable pushes the errors to the subscribers	A promise pushes the errors to the child promises.

39. What do you mean by data binding?

In Angular, data binding refers to the mechanism of establishing a connection between the component's data (model) and the user interface (view) elements. It allows for the automatic synchronization and communication of data between the component and the template.

Data binding in Angular provides a way to keep the data in the component and the UI in sync, ensuring that any changes made to the data are reflected in the view, and vice versa. This two-way communication allows for a seamless interaction between the user and the application.

40. What are some disadvantages of using Angular?

Although Angular provides quite a lot of benefits, there are some disadvantages of using it as well. They are as follows:

- Getting good SEO results on an Angular application can be a bit difficult and may need a bit of configuration.
- Angular has a lot of features packed into it, so getting to know each of them and learning how to use them effectively together can be a little difficult.
- Angular can add quite a lot of weight to your JavaScript bundle, so using it for smaller projects can be very inefficient and may significantly increase the load size.

Interested in learning React JS? Click here to learn more about this React JS Certification!

41. What do you mean by string interpolation?

String interpolation in Angular, also known as the mustache syntax, only allows one-way data binding. It is a special syntax that makes use of double curly braces {{}} so that it can display the component data. Inside the braces are the JavaScript expressions that Angular needs to execute to retrieve the result, which can further be inserted into the HTML code. Moreover, as part of the digest cycle, these expressions are regularly updated and stored.

42. What are the differences between Angular decorator and annotation?

In Angular, decorators are design patterns that help in the modification or decoration of the respective classes without making changes in the actual source code.

Annotations, on the other hand, are used in Angular to build an annotation array. They use the Reflective Metadata library and are a metadata set of the given class.

43. What is an AOT compilation in Angular?

The AOT (ahead-of-time) compiler in Angular converts Angular HTML and TypeScript code into JavaScript code during the build phase. This makes the rendering process much faster. This compilation process is needed since Angular uses TypeScript and HTML code. The compiler converts the code into JavaScript, which can then be effectively used by the browser that runs our application.

44. What are the advantages of AOT?

AOT compilation has several advantages as mentioned below:

Fast rendering: Since, after compilation, the browser would download a pre-compiled version of our application, it can render the application immediately without compiling the app.

Less asynchronous requests: It takes external HTML templates and CSS style sheets and inlines them within the application JavaScript, which reduces the number of separate Ajax requests.

Smaller download size: The compiler will minify the code for us so that the download size is less.

Template error detection: During the compilation phase, any issues in the templates will be detected and reported by the compiler so that they can be corrected before production.

45. What are the three phases of AOT?

The three phases of AOT are:

- code analysis
- code generation
- template type checking

46. What are the components in Angular?

Components are the basic building block of the user interface in Angular. A component consists of HTML, CSS, and JavaScript for a specific portion of a user interface. We can think of these as a custom HTML element that only Angular can understand. These components are isolated, i.e., styles and code from one component do not affect other components as they get namespaced by the compiler. These components are then pieced together by the Angular framework to build the user interface for the browser to render.

47. What are dynamic components?

Dynamic Components in the Angular framework are the components that help in building large-scale applications easily. Dynamic components are usually instantiated and placed in the application at runtime.

48. What is the purpose of base href tag?

The href attribute is used to specify the base URL for all relative URLs on a page.During navigation, the base href tag is used by the Angular router as a base path to the component, template, and module files.

49. What are modules in Angular?

A module is the logical boundary of an application. It is used to encapsulate code dealing with a specific aspect of the application, such as routing, HTTP, validation, etc. The main reason why modules are used is to enhance application composability. For example, if we wish to implement validation logic using different libraries, then for the one we have already implemented, we can create a new validation module and replace the current one with the new one, and our application would work just the same. In Angular, we create a module using the **NgModule** decorator.

50. What is DOM?

The full form of DOM is Document Object Model, and it is responsible for representing the content of a web page and changes in the architecture of an application. Here, all the objects are organized in the form of a tree, and the document can easily be modified, manipulated, and accessed only with the help of APIs.

51. What are services in Angular?

A service in Angular is a term that covers broad categories of functionalities. A service is any value, function, or feature that an app needs. A service is typically used to accomplish a very narrow purpose such as HTTP communication, sending data to a cloud service, decoding some text, validating data, etc. A service does one thing and does it well. It is different from a component as it is not concerned with HTML or any other kind of presentation logic. Normally, a component uses multiple services to accomplish multiple tasks.

52. What is the difference between jQuery and Angular?

The main difference between jQuery and Angular is that jQuery is a JS library, whereas Angular is a JS frontend framework. Some of the other differences between the two are mentioned below:

- Unlike jQuery, Angular offers two-way data binding
- Unlike Angular, jQuery does not offer support for the RESTful API
- Angular supports deep linking routing, while jQuery does not
- Form validation is available in Angular but not in jQuery

Although they have their differences, Angular and jQuery also have their set of similarities, like both jQuery and Angular expressions consisting of variables, operators, and literals.

Angular Interview Questions for 4 to 5 Years Experience

53. What is two-way data binding?

Two-way data binding is done in Angular to ensure that the data model is automatically synchronized in the view. For example, when a user updates some data in a model and that model is being displayed in multiple places in a component, that update should be reflected in all the places.

Two-way data binding has been supported in Angular for a long time. Although, it is something that should be used with careful consideration as it could lead to poor application performance or performance degradation as time goes on. It is called two-way data binding because we can change some data that is in the component model from the view that is HTML, and that change can also propagate to all other places in the view where it is displayed.

Looking to get started with AngularJS? Head on to our blog on AngularJS tutorial!

54. What are pipes in Angular?

When we are trying to output some dynamic data in our templates, we may sometimes need to manipulate or transform the data before it is put into our templates. Though there are several ways of doing that, in Angular, using pipes is the most preferred way. A pipe is just a simple function, which we can use with expressions in our templates.

Pipes are extremely useful as we can use them throughout our application after declaring them just once and registering them with the Angular framework. Some of the most common built-in pipes in Angular are UpperCasePipe, LowerCasePipe, CurrencyPipe, etc.

55. What are observables in Angular?

An observable is a declarative way using which we can perform asynchronous tasks. Observables can be thought of as streams of data flowing from a publisher to a subscriber. They are similar to promises as they both deal with handling asynchronous requests. However, observables are considered to be a better alternative to promises as the former comes with a lot of operators that allow developers to better deal with asynchronous requests, especially if there is more than one request at a time.

Observables are preferred by many developers as they allow them to perform multiple operations such as combining two observables, mapping an observable into another observable, and even piping multiple operations through an observable to manipulate its data.

56. How do you chain pipes?

The syntax used for chaining pipes in Angular is given in the following example:

```
Today is {{ today | date:’fullDate’ | uppercase}}.
```

57. What does Angular Material mean?

Angular Material is a UI component library that allows professionals to develop consistent, attractive, and completely functional websites, web pages, and web applications. It becomes capable of doing so by following modern principles of web designing, such as graceful degradation and browser probability.

58. What is RxJS?

RxJS is a library, and the term stands for Reactive Extensions for JavaScript. It is used so that we can use observables in our JavaScript project, which enables us to perform reactive programming. RxJS is used in many popular frameworks such as Angular because it allows us to compose our asynchronous operations or callback-based code into a series of operations performed on a stream of data that emits values from a publisher to a subscriber. Other languages such as Java, Python, etc. also have libraries that allow them to write reactive code using observables.

59. What is bootstrapping?

Angular bootstrapping, in simple words, allows professionals to initialize or start the Angular application. Angular supports both manual and automatic bootstrapping. Let's briefly understand the two.

- **Manual bootstrapping**: It gives more control to professionals with regards to how and when they need to initialize the Angular app. It is extremely useful in places where professionals wish to perform other tasks and operations before the Angular compiles the page.
- **Automatic bootstrapping**: Automatic bootstrapping can be used to add the ng-app directive to the application's root, often on the tag if professionals need Angular to automatically bootstrap the application. Angular loads the associated module once it finds the ng-app directive and, further, compiles the DOM.

Want to learn more about Angular Bootstrap? Check out our Angular Bootstrap Tutorial!

60. What do you mean by dependency injection?

Dependency injection (DI) in Angular is a software design pattern in which the objects can be passed in the form of dependencies instead of hard-coding them in the respective components. This concept is extremely handy when it comes to separating the object logic creation from its consumption.

The function 'config' uses DI that needs to be configured so that the module can be loaded to retrieve the application elements. Besides, this feature allows professionals to change dependencies based on necessities.

61. What is the digest cycle process in Angular?

Digest cycle in Angular is the process in which the watch list is monitored to track changes in the watch variable value. In each digest cycle, there is a comparison between the present and the previous versions of the scope model values.

62. What are the distinct types of Angular filters?

Filters are a part of Angular that helps in formatting the expression value to show it to the user. They can be added to services, directives, templates, or controllers. You also have the option to create personalized filters as per requirements. These filters allow you to organize the data easily such that only the data that meets the respective criteria are displayed. Filters are placed after the pipe symbol (|) while used in expressions.

Various types of filters in Angular are mentioned below:

currency: It converts numbers to the currency format

filter: It selects a subset containing items from the given array

date: It converts a date into a necessary format

lowercase: It converts the given string into lowercase

uppercase: It converts the given string into uppercase

orderBy: It arranges an array by the given expression

json: It formats any object into a JSON string

number: It converts a number value into a string

limitTo: It restricts the limit of a given string or array to a particular number of elements or strings

63. How can one create a service in Angular?

A service in Angular is an object that can be substituted. It is wired and combined with the help of dependency injection. Services are developed by getting registered in a module that they need to be executed in. The three methods of creating a service in Angular are as follows:

- Service
- Factory
- Provider

Angular Advanced Interview Questions for Senior Developers (6 to 10 Years)

64. What does subscribing mean in RxJS?

In RxJS, when using observables, we need to subscribe to an observable to use the data that flows through that observable. This data is generated from a publisher and is consumed by a subscriber. When we subscribe to an observable, we pass in a function for the data and another function for errors so that, in case there is some error, we can show some message or process the message in some way.

65. What is an Angular Router?

Routing in a single-page front-end application is the task of responding to the changes in the URL made by adding and removing content from the application. This is a complicated task as we first need to intercept a request that changes the browser's URL as we do not wish for the browser to reload. Then, we need to determine what content to remove and what content to add, and finally, we change the browser's URL as well to show the user the current page they are on.

As we can see, this can be very difficult to implement, especially in multiple applications. That is why Angular comes with a full routing solution for a single-page application. In this, we can define routes with matching components and let Angular handle the routing process.

66. What is the purpose of the common module in Angular?

In Angular, the common module that is available in the package **@angualr/common** is a module that encapsulates all the commonly needed features of Angular, such as services, pipes, directives, etc. It contains some sub-modules as well such as the **HttpClientModule**, which is available in the **@angular/common/http package.** Because of the modular nature of Angular, its functionalities are stored in small self-contained modules, which can be imported and included in our projects if we need these functionalities.

67. What are the differences between AngularJS and Angular?

AngularJS is the previous version of Angular, which is a complete rewrite, i.e., there are several differences between the two that we can highlight.

- **Architecture**: AngularJS supports the MVC architecture in which the model contains the business logic; the view shows the information fetched from the models, and the controller manages interactions between the view and the model by fetching data from the model and passing it to the view. On the other hand, Angular architecture is based on components where instead of having separate pieces for logic, presentation, etc., we now have a single self-contained piece of the user interface that can be used in isolation or included in a big project.
- **Language**: In AngularJS, we could only use JavaScript. However, in Angular, we can use both TypeScript and JavaScript.
- **Mobile support**: In AngularJS, we do not get mobile browser support out of the box, but in Angular, we do get mobile support for all popular mobile browsers.

Angular Interview Questions for 6 to 7 Years of Experience

68. What is the scope in Angular?

A scope is an object in Angular referring to the application model. It is a context for executing expressions. These scopes are organized in a hierarchical form that is similar to the application's DOM structure. A scope helps in propagating various events and watching expressions.

69. How do you create directives using CLI?

To create a directive using Angular CLI, the following steps are used:

- Start a new project using Angular CLI through the following command:

```
'ng new [application-name]'
```

- Now change the directory into a new directory through the command:

```
'cd [application-name]'
```

- Once done with changing the directory, use the following command to generate a new directive:

```
'ng generate directive [path-to-directives/my-directive]'
```

70. What is a rule in Schematics?

A rule in Schematic refers to a set function that takes a tree, transforms it and finally returns a new tree.

71. What is Schematics CLI?

Schematics CLI is used to transform web-based applications projects.

72. What is HttpClient, and what are its benefits?

HttpClient is an Angular module used for communicating with a back-end service via the HTTP protocol. Usually, in front-end applications, for sending requests, we use the fetch API. However, the fetch API uses promises. Promises are useful, but they do not offer the rich functionalities that observables offer. This is why we use **HttpClient** in Angular as it returns the data as an observable, which we can subscribe to, unsubscribe to, and perform several operations on using operators. Observables can be converted to promises, and an observable can be created from a promise as well.

73. What is multicasting in Angular?

In Angular, when we use the **HttpClient** module to communicate with a backend service and fetch some data, after fetching the data, we can broadcast it to multiple subscribers in one execution. This task of responding with data to multiple subscribers is called multicasting. It is specifically useful when we have multiple parts of our applications waiting for some data. To use multicasting, we need to use an RxJS subject. As observables are unicast, they do not allow multiple subscribers. However, subjects do allow multiple subscribers and are multicast.

Check out this Angular Certification course to get an in-depth understanding of the Angular framework.

74. What is a directive in Angular?

A directive in Angular is used to extend the syntax and capabilities of a normal HTML view. It have special meaning and are understood by its compiler. When Angular begins compiling the TypeScript, CSS, and HTML files into a single JavaScript file, it scans through the entire code and looks for a directive that has been registered. In case it finds a match, then the compiler changes the HTML view accordingly.

Angular is shipped with many directives. However, we can build our directives and let Angular know what they do so that the compiler knows about them and uses them during the compilation step.

75. What will happen if you do not supply a handler for an observer?

When a handler is not supplied to a notification type, the observer automatically ignores notifications of that type and the observer instance publishes values only when it is subscribed to.

76. What are angular elements?

Angular elements refer to the Angular components that are a web standard for defining new HTML elements and are packed as custom elements. These custom elements are also called Web Components. The Angular custom elements bootstrap automatically when added to the DOM tree.

77. What is the browser support of Angular Elements?

Since Angular is built on the latest web platform standards, Angular elements are supported by Chrome, Edge (Chromium-based), Firefox, Opera, and Safari and other currently existing browsers via polyfills.Polyfills enable the running of full Angular applications.

78. What is the role of SPA in Angular?

SPA stands for Single Page Application. This technology only maintains one page, index.HTML, even when the URL changes constantly. SPA technology is easy to build and extremely fast in comparison to traditional web technology.

79. What are Angular building blocks?

The following building blocks play a crucial role in Angular:

- **Components**: A component can control numerous views wherein each of the views is a particular part on the screen. All Angular applications have a minimum of one component called the root component. This component is bootstrapped in the root module, the main module. All the components include the logic of the application that is defined in a class, while the main role of the class is to interact with the view using an API of functions and properties.
- **Data binding**: Data binding is the process in which the various sections of a template interact with the component. The binding markup needs to be added to the HTML template so that Angular can understand how it can connect with the component and template.
- **Dependency injection**: It uses DI so that it can offer the necessary dependencies: mainly services, to the new components. The constructor parameters of a component inform Angular regarding the numerous services needed by the component, and DI provides a solution that gives the necessary dependencies to the new class instances.
- **Directives**: Angular templates are of a dynamic nature, and directives help Angular understand how it can transform the DOM while manifesting the template.
- **Metadata**: Classes have metadata attached to them with the help of decorators so that Angular will have an idea of processing the class.
- **Modules**: Module or NgModule is a block of code organized using the necessary capabilities set, having one specific workflow. All Angular applications have at

least one module, the root module, and most of the applications have numerous modules.

- **Routing**: The Angular router helps interpret the URL of a browser to get a client-generated experience and view. This router is bound to page links so that Angular can go to the application view as soon as the user clicks on it.
-
- **Services**: Service is a vast category that ranges from functions and values to features that play a significant role in Angular applications.
- **Template**: The view of each component is linked with a template, and an Angular template is a type of HTML tag that allows Angular to get an idea of how it needs to render the component.

80. Explain the MVVM architecture.

The MVVM architecture plays a significant role in eliminating tight coupling between the components. This architecture includes the following three parts:

- **Model:** The model represents the business logic and data of a particular application. In other words, it consists of an entity structure. The model has the business logic, including model classes, remote and local data sources, and the repository.
- **View**: Within Angular, the view embodies the graphical user interface (UI) of an application. Its primary role involves presenting the data sourced from the component and managing user interactions. Constructed through HTML templates, the view dynamically renders and adjusts its content in accordance with the component's data and the application's logic.
- **ViewModel**: ViewModel is the application's abstract layer that connects the View and the Model and acts as a bridge between the two. It does not know which View needs to be made use of since it does not have any direct access to the View. The two are connected using data binding, and the ViewModel records all the changes that are made to the View and makes the necessary changes to the Model.

81. Describe Angular authentication and authorization.

The login details of a user are given to an authentication API available on the server. Once the credentials are validated by the server, it returns a JSON web token (JWT), which includes attributes and the data of the current user. Further, the user is easily identified using JWT, and this process is known as authentication.

After logging in, users have various types and levels of access—some can access everything, while others may have restrictions from some resources. Authorization determines the access level of these users.

82. What is REST?

REST in Angular stands for Representational State Transfer. It is an API that works on the request of HTTP. Here, the requested URL points to the data that has to be processed, after which an HTTP function is used to identify the respective operation that has to be performed on the data given. The APIs that follow this method are referred to as RESTful APIs.

83. Explain Angular CLI.

Angular CLI is otherwise known as Angular command-line interface. Angular supports CLI tools that give professionals the ability to use them to add components, deploy them instantly, and perform testing and many such functions.

84. What is schematic?

Schematics are template-based code generators that enable complex logic. It is usually packaged into collections and installed with npm. The aim of schematics is to transform a software project by generating or modifying codes.

85. Explain the different kinds of Angular directives.

There are three kinds of directives in Angular. Let's discuss them:

- **Components**: A component is simply a directive with a template. It is used to define a single piece of the user interface using TypeScript code, CSS styles, and the HTML template. When we define a component, we use the component decorated with the @ symbol and pass in an object with a selector attribute. The selector attribute gives the Angular compiler the HTML tag that the component is associated with so that now when it encounters this tag in HTML, it knows to replace it with the component template.
- **Structural:** Structural directives are used to change the structure of a view. For example, if we wish to show or hide some data based on some property, we can do so by using the **ngIf** directive, or if we wish to add a list of data in the markup, we can use ***ngFor,** and so on. These directives are called structural directives because they change the structure of the template.
- **Attribute**: Attribute directives change the appearance or behavior of an element, component, or another directive. They are used as the attributes of elements. Directives such as **ngClass** and **ngStyle** are attribute directives.

Want to become an Angular expert? Join this Angular JS Course in Bangalore now!

86. What are the different types of compilers used in Angular?

In Angular, we use two different kinds of compilers:

- Just-in-time (JIT) compiler
- Ahead-of-time (AOT) compiler

Both these compilers are useful but for quite different purposes. The JIT compiler is used to compile TypeScript to JavaScript as our browsers cannot understand TypeScript but only JavaScript. This compilation step is done in a development environment, i.e., when less time is needed to be spent on compilation and more in development to quickly iterate over features. The JIT compiler is used when we use **ng serve** or **ng build** command to serve the app locally or create an uncompressed build of the entire codebase.

On the other hand, the AOT compiler is used to create a minified production build of the entire codebase, which can be used in production. To use the AOT compiler, we have to use the **ng build** command with the **–prod blog: ng build –prod**. This instructs the Angular CLI to create an optimized production build of the codebase. This takes a bit more time because several optimizations, such as minification, can take time but for production builds, this time can be spared.

87. What is server-side rendering in Angular?

In a normal Angular application, the browser executes our application, and JavaScript handles all the user interactions. However, because of this, sometimes, if we have a large application with a big bundle size, our page's load speed is slowed down quite a bit as it needs to download all the files, parse JavaScript, and then execute it. To overcome this slowness, we can use server-side rendering, which allows us to send a fully rendered page from the server that the browser can display and then let the JavaScript code take over any subsequent interactions from the user.

88. What is Angular Universal?

Angular Universal is a package for enabling server-side rendering in Angular applications. We can easily make our application ready for server-side rendering using the Angular CLI. To do this, we need to type the following command:

```
ng add @nguniversal/express-engine
```

This allows our Angular application to work well with an ExpressJS web server that compiles HTML pages with Angular Universal based on client requests. This also creates the server-side app module, app.server.module.ts, in our application

directory.

89. What are HttpInterceptors in Angular?

HttpInterceptors are part of the **@angular/common/HTTP** module and are used to inspect and transform HTTP requests and HTTP responses as well. These interceptors are created to perform checks on a request, manipulate the response, and perform cross-cutting concerns, such as logging requests, authenticating a user using a request, using gzip to compress the response, etc.

90. How does one share data between components in Angular?

There is not one but various methods to share data between components in Angular. They are mentioned as below:

- **Parent to Child**: via Input
- **Child to Parent**: via Output() and EventEmitter
- **Child to Parent**: via ViewChild
- **Unrelated Components**: via a Service

Angular Interview Questions for 8 to 10 Years Experience

91. What are the differences between Angular expressions and JavaScript expressions?

Angular expressions and JavaScript expressions are quite different from each other as, in Angular, we are allowed to write JavaScript in HTML, which we cannot do in plain JavaScript. Also, all expressions in Angular are scoped locally. But, in JavaScript, these expressions are scoped against the global window object. However, these differences are reconciled when the Angular compiler takes the Angular code we have written and converts it into plain JavaScript, which can then be understood and used by a web browser.

Check out this Full Stack Web Developer Course to become a Full Stack Web Developer.

92. What is the difference between interpolated content and the content assigned to the innerHTML property of a DOM element?

The angular interpolation happens when in our template, we type some JavaScript expression inside double curly braces **'{{ someExpression() }}'**. This is used to add dynamic content to a web page. However, we can do the same by assigning some dynamic content to the innerHTML property of a DOM element. The difference between the two is that, in Angular, the compiler always escapes the interpolated content, i.e., HTML is not interpreted, and the browser displays the code as it is with brackets and symbols, rather than displaying the output of the interpreted HTML. However, in innerHTML, if the content is HTML, then it is interpreted as the HTML code.

93. What is ng-content and its purpose?

The usage of **ng-content** in Angular is to insert the content dynamically inside the component. It helps in increasing component reusability and passing content inside the component selector.

94. What is ngcc?

The ngcc(Angular Compatibility Compiler) is a tool used in Angular to upgrade node_module, compiled with non-ivy ngc into ivy compliant format.

Table of Contents

95. What is folding?

In Angular, it might be possible while generating the code that some of the non-exported members are folded. This is called Folding, i.e the process in which the evaluation of an expression is done by the collector and result is recorded in the .metadata.json, is known as Folding.

96. What are macros?

A macro is a portion of a programming code that helps automate routine tasks. It usually runs in an Excel environment. In Angular, Macros is supported in functions, static methods, etc. Consider the below example of a Macros code in Angular :

```
export function wrapInArray<T>(value: T): T[] {
return [value];
}
```

97. What is the state function?

The state function in Angular declares an animation state within a trigger attached to an element. The following is the syntax for the state function:

```
state(name: string, styles: AnimationStyleMetadata, options?: { par
```

98. What is the Style()?

The Style function in Angular is used to declare a key/value object that contains CSS properties/styles and is used for an animation. The syntax of the Style function is given by:

```
style(tokens: "*" | { [key: string]: string | number; } | ("*" | {
```

99. What is NgZone?

NgZone is a service provided by Angular that allows you to execute functions in the Angular zone through a run() method. This execution is carried out when NgZone triggers change detection automatically at the right time.

```
export class AppComponent implements OnInit {
constructor(private ngZone: NgZone) {}
ngOnInit() {
// New async API is not handled by Zone, so you need to use ngZone
// to make the asynchronous operation callback in the Angular zone
// trigger change detection automatically.
someNewAsyncAPI(() => {
this.ngZone.run(() =>
// update the data of the component
});
});
}
}
```

100. What is NoopZone?

NoopZone helps Angular figure out and know when to trigger the change detection. However, in the event when Angular needs to be used without a zone, the NoopZone too needs to be configured.

Hopefully, these Angular interview questions in 2023 have helped you get a better grasp of Angular as a framework, as well as its various features and capabilities. These frequently asked questions have been created to give you a better understanding of the kinds of questions asked in interviews, so they will help you in understanding Angular interviews and Angular as a framework. These Angular interview questions with their answers might have whetted your appetite for learning more about the framework. They will surely help you to ace your next job interview.

Angular Scenario-Based Interview Questions

101. You need to create an UI that will adapt based on the role of the user. What will be your approach.

To achieve the role-based UI, we need to implement Angular Guards and directives for dynamic content display. This ensures a proper UI experience, which also makes sure of security and can be reused.

```
@Injectable({ providedIn: 'root' })
export class AuthService {
  // Mock role check
  hasRole(role: string): boolean {
    return localStorage.getItem('userRole') === role;
  }
}

@Directive({
  selector: '[appRoleBasedAccess]'
})
export class RoleBasedAccessDirective {
  constructor(private el: ElementRef, private authService: AuthSer

  @Input() set appRoleBasedAccess(role: string) {
    if (!this.authService.hasRole(role)) {
      this.el.nativeElement.style.display = 'none';
    }
  }
}
```

102. A form with numerous inputs is experiencing sluggish responses, especially during data entry. How can you enhance the performance of the form so that the functionality or the user experience does not get affected.

To enhance the form and improve its performance we can implement “ChangeDetectionStrategy.OnPush” which is a strategy that the default change detector uses to detect changes. It reduces the load of processing inputs. This improves the form responsiveness by changing the rendering cycle to whenever a change is done. It minimizes the DOM updates

```
@Component({
  selector: 'app-user-form',
  changeDetection: ChangeDetectionStrategy.OnPush
})
export class UserFormComponent {
  userForm = new FormGroup({
    name: new FormControl('')
  });

  constructor() {
    this.userForm.get('name').valueChanges.pipe(
      debounceTime(300)
    ).subscribe(value => {
      // Handle the changes made
    });
  }
}
```

103. The Project involves fetching data from multiple APIs on a single page. Create a strategy to fetch and join this data in an organized manner. Also, ensure that there is minimal impact on user experience and performance of the code.

We can use “forkJoin” or “combineLatest” operators from the RxJS. These operators helps when we have multiple detectables that rely on each other for some calculations or determination.

```
constructor(private http: HttpClient) {}
fetchData() {
  forkJoin({
    data1: this.http.get('/api/data1'),
```

```
    data2: this.http.get('/api/data2')
  }).subscribe(({ data1, data2 }) => console.log(data1, data2));
}
```

Angular Coding Interview Questions

104. Create a custom pipe to convert strings to title case.

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

@Pipe({
  name: 'titleCase'
})
export class TitleCasePipe implements PipeTransform {
  transform(value: string): string {
    if (!value) return value;
    return value.replace(/\w\S*/g, (txt) => {
      return txt.charAt(0).toUpperCase() + txt.substr(1).toLowerCa
    });
  }
}
```

105. Build a reactive form with validation.

```
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl, Validators } from '@angular/forms

@Component({
  selector: 'app-user-form',
  templateUrl: './user-form.component.html',
  styleUrls: ['./user-form.component.css']
})
export class UserFormComponent implements OnInit {
  userForm: FormGroup;

  ngOnInit() {
    this.userForm = new FormGroup({
      'name': new FormControl('', Validators.required),
      'email': new FormControl('', [Validators.required, Validator
      'password': new FormControl('', [Validators.required, Valida
    });
  }

  onSubmit() {
    console.log(this.userForm.value);
  }
}
```

106. Implement a service with HttpClient.

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Injectable({
  providedIn: 'root'
})
export class UserDataService {
  constructor(private http: HttpClient) {}

  fetchUserData() {
    this.http.get('https://jsonplaceholder.typicode.com/users')
      .subscribe(data => console.log(data));
  }
}
```

107. Use ngFor directive to display a list.

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

@Component({
  selector: 'app-item-list',
  template: `
    <ul>
      <li *ngFor="let item of items">{{ item }}</li>
    </ul>
  `
})
export class ItemListComponent {
  items = ['Item 1', 'Item 2', 'Item 3'];
}
```

108 Create a directive to change the background color of an element.

```
import { Directive, ElementRef, Renderer2, OnInit } from '@angular

@Directive({
  selector: '[appHighlight]'
})
export class HighlightDirective implements OnInit {
  constructor(private el: ElementRef, private renderer: Renderer2)

  ngOnInit() {
    this.renderer.setStyle(this.el.nativeElement, 'backgroundColor
  }
}
```

Angular Salary Trends

Job Role	Average Salary in India	Average Salary in the USA
Frontend Developer (Angular) (0-9 years experience)	Minimum – ₹4L /yr	Minimum – $1,72,956 /yr
	Average – ₹18L /yr	Average – $1,72,000 /yr
	Highest – ₹25L /yr	Highest – $2,34,851 /yr

Angular Job Trends

According to the Bureau of Labor Statistics, there will be a growth of 8% in the job requirement for AngularJS, which is higher than the national average.

- **Global Trends:** The introduction of a declarative approach in AngularJS marks a new way to manage templates. With 1,327 new jobs in the United States and a tally of 30,000+ jobs in India, Angular is making a comeback with more changes in Angular v17.
- **Growth Projections:** With 25% growth in software developer jobs, the requirement for Angular Developer will rise to 8%, leading to more jobs.

Angular Roles and Responsibilities

According to the job posted on LinkedIn by NodeFlair, the following are the responsibilities and skills required for jobs in AngularJS:

Role: Angular Developer

1. Responsibilities

- Build and shape the website and its interface
- Ensure features are built to be fast and can grow without issues
- Solve complex needs with well-thought-out software solutions

- Work closely with project leaders, platform heads, product managers, sales teams, and analysts to fully understand what the software needs to do
- Lead and work together with the team that builds the product
- Plan for new updates and organize how work is done

2. Skills Required

- Involved in all stages of software development (planning, building, launching, and updating)
- Skilled in writing clean and effective JavaScript/Typescript, HTML, and CSS code (familiarity with Less or Sass is a plus)
- Capable of creating user interface components that are easy to use again, grow without limits, and easy to maintain
- Exceptional at using Angular 2+ and RxJS. Understanding how to manage data and state in Angular, like with NgRx, is beneficial.
- Understands AngularJS and knows how to switch from it to Angular 2+.
- Experience in building, explaining, and keeping up a library of frontend components using the newest Angular version

Conclusion

We hope this set of Angular interview questions will help you prepare for your interviews. Best of luck in your endeavors!

If you want to start your career or elevate your skills in web development, you can enroll in our Advanced Certification in Full Stack Web Development or enroll in Intellipaat's Executive Post Graduate Certification in Full Stack Web Development and get certified today.

If you want to dive deep into more digital marketing interview questions, join Intellipaat's Web Technology Community and get answers to your queries from like-minded enthusiasts.