Angular is a highly popular JavaScript framework utilized for building web applications. This open-source development platform provides an array of comprehensive features and tools that facilitate the creation of interactive and dynamic web applications. Therefore, Angular is considered to be an indispensable tool for any contemporary web development project.

If you're on the lookout for a job in Angular development, it's essential to brush up on your knowledge of the framework before the interview. To assist you in preparing effectively, we've compiled a list of frequently asked Angular interview questions and answers in this article. By reviewing these questions and answers, you'll be better equipped to ace the interview and land the job. So, take your time and study them thoroughly.

1. What is Angular?

Angular is a web application framework that uses TypeScript and is open-source. It is designed for building web applications and is managed by Google. The framework follows the Model-View-Controller (MVC) architecture, which helps developers create modern, interactive, and dynamic web applications. With Angular, developers have access to a range of tools and resources that are essential for building successful web applications.

2. What are some of the important characteristics of Angular?

Angular sets itself apart from other JavaScript frameworks with its impressive array of features. Some notable examples include:

- **Two-way data binding:** Angular offers developers a convenient feature called two-way data binding. With this functionality, developers can effortlessly connect the model and view, ensuring that the view is always in sync with the model. As a result, this simplifies the coding process for web applications, making it easier for developers to create interactive and dynamic web applications.
- **Templates:** Angular helps expedite the development process through its usage of HTML-based templates. This simplifies the procedure and enhances speed.
- **Components:** Angular components serve as the foundation of an application and can be used to construct intricate user interfaces.
- **MVC Architecture:** Angular utilises the MVC (Model-View-Controller) framework, making it simpler to organise the code and keep the application up to date.
- **Routing:** Angular gives developers the opportunity to integrate multiple views into an app and switch between them easily. This provides a great deal of flexibility for creating powerful user experiences.

3. What are the advantages of using Angular?

Angular is a versatile platform that is well-suited for developers at all skill levels, from beginners to experienced professionals. Its user-friendly design and ease of use make it an excellent choice for anyone looking to learn a new programming language.

The Model-View-Controller (MVC) architecture employed by Angular allows developers to organize their code effectively, simplifying the maintenance of applications. This approach is widely used by software engineers due to its convenience, making it an excellent option for creating and maintaining applications.

The framework is comprehensive and robust, giving developers the tools they need to build dynamic and interactive web applications. Its flexibility and power enable it to be used for a wide range of applications.

Angular is scalable and can handle large and complex applications with ease, making it a top choice for these types of projects. Its ability to manage complex projects efficiently sets it apart from other platforms.

Additionally, Angular supports code reusability, which can significantly reduce the time and cost associated with software development.

4. What is a Directive in Angular?

Angular Directives are classes that have been decorated with the @Directive annotation. They are utilized to add customized behavior to an element in the DOM (Document Object Model). This lets developers broaden the HTML language and create custom, reusable components. Directives can be used to change the DOM, attach event handlers, and edit styles.

5. What kinds of Directives are available in Angular?

Angular comes with three different types of directives:

1. **Component Directives:** These are directives with a template. They are used to create reusable components. Examples include @Component, @Input, and @output,
2. **Structural Directives:** These are directives that change the DOM layout by adding and removing DOM elements. Examples include *ngIf, *ngFor, and *ngSwitch.

3. **Attribute Directives:** These are directives that change the appearance or behavior of an existing element. Examples include ngStyle, ngClass, and ngModel.

6. What is the difference between one-way data binding and two-way data binding?

One-way data binding is a process that enables the view to be updated when the model is altered, however the opposite is not true. This mechanism is useful in keeping the view in sync with the model.

Two-way data binding is a process that keeps both the view and the model in sync. This means that when either one of them is altered, the other is updated accordingly. This type of data binding ensures that the changes made to the view are reflected in the model and vice-versa.

7. What is the role of a Component in Angular?

Angular applications are built upon components — the fundamental building blocks of the application. Each component is responsible for a particular view layer, and is composed of HTML, CSS and JavaScript. Components are essential for creating complex user interfaces, and can be reused throughout the application.

8. What does a Service in Angular do??

Angular Service provides an efficient way to reuse code for tasks that need to be done regularly, such as sending HTTP requests, retrieving data from a database, and more. By injecting services into components, developers can easily access and use them while ensuring they do not need to be implemented again.

9. What does Dependency Injection do in Angular?

Angular utilizes dependency injection to give components the services they require. This programming technique involves a class obtaining the instances of objects it needs (known as dependencies) from an outside source instead of generating them itself. The use of dependency injection provides components with the flexibility to be employed in various scenarios and with different sets of dependencies. This not only helps to reduce redundancy in the code (DRY — Don't Repeat Yourself) but also makes it easier to test and maintain.

10. What is the difference between Observables and Promises in Angular?

Observables in Angular are a powerful way to manage asynchronous data, allowing to create, transform and combine streams of data. Observables enable components to reactively manage data and are widely used in modern web frameworks such as Angular.

Promises in Angular is an interface for managing asynchronous code. It allows developers to write asynchronous code in a synchronous-like style, simplifying complex tasks such as making multiple HTTP requests or waiting for data to be returned from an API. Promises can be used to make code easier to read, debug, and test.

Difference between Observables and Promises: An Observable can be canceled, while a Promise cannot. Observables enable multiple values to be emitted over time and enable subscriptions, while a Promise is a single-value object that is either fulfilled or rejected and is not cancellable. Furthermore, Observables provide developers with operators such as map, forEach, filter, reduce, retry, and others, which enable developers to control and manipulate data, whereas Promises do not.

11. What is the purpose of a Router in Angular?

Routers are used to send traffic between different views. Also, they are valuable for adding multiple views to an application and seamlessly transitioning between them.

12. What is the difference between a Component and a Module in Angular?

An Angular application is made up of components — these are the building blocks that are used to create complex user interfaces and can be used repeatedly across the application.

A module is a collection of components, directives, pipes, and services. It is used to group related components and services together and make them easier to maintain.

13. What is the Reactive Form vs Template Driven Forms??

Angular reactive forms are used for complex forms due to their scalability. They allow developers to create dynamic forms while providing an option to track the form's value and validity. They allow users to monitor the current state of the form.

Template-driven forms are used for simple forms due to their simplicity and reduced amount of code needed for implementation. They are best suited for basic forms. Template-driven forms are built using HTML and data binding and can be set up for two-way data binding.

14. What is the purpose of a Pipe in Angular?

A pipe in angular is used to transform data. It is used to format data such as dates, currency, and numbers before displaying it in the view.

15. How many ways do angular components communicate with each other?

Angular components can communicate with each other in the following ways:

1. Input and Output Properties: Components can share information with each other by utilizing @Input and @Output properties. The parent component can provide data to the child component with an input property, while the child component can send data back to the parent component with an output property.

2. Services: A shared service is an ideal way to share data between components. The service can be injected into any component that needs the data and any component can update the data in the service.

3. ViewChild and ContentChild: ViewChild and ContentChild are two decorators that allow a component to access another component and its properties.

4. EventEmitter: EventEmitter allows a component to emit events, which can be listened to by other components.

5. Template Variables: Template variables can be used to refer to components from within the template. This allows components to access one another and their properties.

6. Local References: Local references can be used to refer to components from within the template. This allows components to access one another and their properties.

7. Router: The router can be used to pass data between components when navigating between routes.

16. What is the purpose of Interceptors in Angular?

In Angular, interceptors play an important role in modifying or altering requests and responses before they are handled. This feature is useful for various purposes like authentication, logging, or modifying requests before sending them to the server. Additionally, interceptors can also intercept errors and handle them more effectively, making them a powerful tool for developers.

17. What is the lazy loading architecture in Angular?

Lazy loading is a design pattern employed by Angular, which defers the loading of components until the point that they are actually needed. This approach aids in reducing the application’s initial loading time and optimizes its performance.

The primary benefit of lazy loading is that it enables the application to obtain only the code it requires, at the time it needs it. This can significantly reduce the application’s overall size, making it quicker and more efficient for users to download and use.

18. What are the lifecycle hooks in Angular?

Angular lifecycle hooks are a powerful feature that allow developers to track and perform actions at different stages of a component’s life cycle. These hooks can be used to gain insight into the creation, rendering, updating, and destruction of a component. By leveraging these functions, developers can enhance the user experience and take advantage of their unique capabilities.

19. What are the different kinds of Angular lifecycle hooks?

Angular Lifecycle Hooks:

- **ngOnChanges:** Called when an input/output binding value changes
- **ngOnInit:** Initialize the directive/component after Angular first displays the data-bound properties and sets the directive/component’s input properties
- **ngDoCheck:** Detect and act upon changes that Angular can’t or won’t detect on its own
- **ngAfterContentInit:** Respond after Angular projects external content into the component’s view
- **ngAfterContentChecked:** Respond after Angular checks the content projected into the component
- **ngAfterViewInit:** Respond after Angular initializes the component’s views and child views
- **ngAfterViewChecked:** Respond after Angular checks the component’s views and child views
- **ngOnDestroy:** Cleanup just before Angular destroys the directive/component

20. 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.