Angular Interview Questions by JavaScaler

Easy Level Questions and Answers

1. **What is Angular?**
    - Angular is a platform and framework for building single-page client applications using HTML and TypeScript. It is developed and maintained by Google and provides a robust structure for building scalable web applications.
2. **What are the key features of Angular?**
    - Some key features include:
        - Two-way data binding
        - Modular architecture
        - Dependency injection
        - Directives
        - Services
        - Component-based architecture
        - RxJS for reactive programming
        - CLI for easy project management
3. **Explain the architecture of an Angular application.**
    - Angular applications are built using components, each with an HTML template for the view and a TypeScript class for the logic. The architecture includes:
        - Modules: Group related components, services, and other code.
        - Components: Define views and behaviors.
        - Templates: HTML with Angular binding syntax.
        - Services: Provide business logic and data access.
        - Dependency Injection: Manage dependencies.

4. **What are components in Angular?**
    - Components are the fundamental building blocks of Angular applications. They encapsulate the view (HTML template), logic (TypeScript class), and styles (CSS). Components interact with the DOM and other components.
5. **How do you create a new Angular project?**
    - You can create a new Angular project using Angular CLI with the following command:

```
ng new project-name
```

6. **What is the purpose of a module in Angular?**
    - Modules help organize an application into cohesive blocks of functionality. Each Angular application has at least one module, the root module (`AppModule`). Modules can include components, directives, pipes, and services.
7. **What is data binding in Angular? Explain its types.**
    - Data binding is the synchronization between the model and the view. Types of data binding include:
        - Interpolation: `{{ expression }}`
        - Property binding: `[property]="expression"`
        - Event binding: `(event)="handler"`
        - Two-way binding: `[(ngModel)]="property"`
8. **What are directives in Angular?**
    - Directives are classes that add behavior to elements in your Angular applications. Types of directives include:
        - Component directives: Create custom components.
        - Structural directives: Change the DOM layout (e.g., `ngIf`, `ngFor`).
        - Attribute directives: Change the appearance or behavior of an element (e.g., `ngClass`, `ngStyle`).
9. **What is the purpose of the `ngIf` directive?**

- The `ngIf` directive conditionally includes or excludes an element in the DOM based on the value of an expression. If the expression evaluates to true, the element is included; otherwise, it is removed.

10. **How does Angular handle forms?**
    - Angular handles forms in two ways: template-driven forms and reactive forms. Template-driven forms rely on Angular directives in the template, while reactive forms use a more programmatic approach with `FormGroup` and `FormControl` classes.
11. **What is a service in Angular and how is it used?**
    - A service is a class that provides reusable functionality across components. Services are typically used for data access, business logic, or other reusable tasks. They are provided via dependency injection.
12. **What is dependency injection in Angular?**
    - Dependency injection is a design pattern in which a class receives its dependencies from an external source rather than creating them itself. In Angular, the injector provides dependencies to components, services, and other Angular constructs.
13. **Explain the concept of a template in Angular.**
    - A template is an HTML view with Angular-specific syntax for data binding, directives, and other features. Templates define how the component's view should be rendered.
14. **How do you create and use a pipe in Angular?**
    - You create a pipe by decorating a class with the `@Pipe` decorator and implementing the `PipeTransform` interface. To use a pipe, you include it in the component's template using the pipe syntax `{{ value | pipeName }}`.
15. **What is Angular CLI and how is it useful?**
    - Angular CLI (Command Line Interface) is a powerful tool that simplifies the development process. It provides commands for creating, building, testing, and deploying Angular applications.
16. **What is the purpose of `@Input` and `@Output` decorators?**
    - `@Input` decorator is used to pass data from a parent component to a child component. `@Output` decorator is used to emit events from a child

component to a parent component.

17. **Explain routing in Angular.**
    - Routing in Angular allows navigation between different views or components within an application. It is configured using the `RouterModule` and defines routes with path and component associations.
18. **What is the purpose of the `ngFor` directive?**
    - The `ngFor` directive is used to iterate over a collection and render a template for each item in the collection.
19. **How do you handle events in Angular?**
    - Events in Angular are handled using event binding syntax: `(eventName)="handlerMethod($event)"`. The event handler method can then process the event object.
20. **What are lifecycle hooks in Angular? Name a few.**
    - Lifecycle hooks are methods that allow you to tap into key events in a component's lifecycle. Examples include:
        - `ngOnInit()`
        - `ngOnChanges()`
        - `ngDoCheck()`
        - `ngAfterViewInit()`
        - `ngOnDestroy()`
21. **How do you apply CSS styles to Angular components?**
    - CSS styles can be applied to Angular components in three ways:
        - Inline styles: Defined directly in the component's decorator.
        - External stylesheets: Linked in the component's decorator.
        - Global styles: Defined in the application's global styles file.
22. **What is the purpose of `ngModel`?**
    - The `ngModel` directive binds form input elements to a component property, enabling two-way data binding for form controls.
23. **How do you set up two-way data binding in Angular?**

- Two-way data binding is set up using the `ngModel` directive along with the `[()]` syntax, known as the banana-in-a-box syntax: `[(ngModel)]="property"`.

24. **What is the `HttpClient` module in Angular?**
    - The `HttpClient` module is an Angular module that provides a simplified API for HTTP operations, such as GET, POST, PUT, DELETE, etc. It handles HTTP requests and responses, including handling errors and setting headers.
25. **How do you handle errors in Angular applications?**
    - Errors in Angular applications can be handled using several techniques:
        - Using `try-catch` blocks.
        - Error handling in services with the `catchError` operator from RxJS.
        - Implementing a global error handler using `ErrorHandler`.

Medium to Hard Level Questions and Answers

1. **What is the Angular renderer and why is it used?**
    - The Angular renderer is an abstraction that allows Angular to interact with the DOM. It is used to manipulate the DOM elements, attributes, and CSS styles without direct access, enabling support for different rendering environments like the browser, server-side, or Web Workers.
2. **Explain the purpose of Angular zones.**
    - Angular zones, managed by the Zone.js library, track asynchronous operations and help Angular know when to trigger change detection, ensuring the UI updates automatically when the model changes.
3. **How do you optimize an Angular application for better performance?**
    - Optimization techniques include:
        - Enabling Ahead-of-Time (AOT) compilation.
        - Using lazy loading for feature modules.
        - OnPush change detection strategy.
        - Minimizing the use of `ngOnInit` and heavy computations in lifecycle hooks.

    - Reducing the number of watchers.
    - Caching data and optimizing network calls.

4. **What are Angular services and how do you create them?**
    - Angular services are classes that encapsulate business logic, data access, or other reusable functionality. You create a service using Angular CLI with the command:

```
ng generate service service-name
```

5. **Explain lazy loading in Angular.**
    - Lazy loading is a technique to load feature modules only when they are needed, reducing the initial load time of the application. It is configured in the routing module using the `loadChildren` property.
6. **How do you implement reactive forms in Angular?**
    - Reactive forms are implemented using `FormGroup`, `FormControl`, and `FormBuilder`. This approach provides a more robust way to manage form inputs, validation, and reactive state changes.
7. **What is Angular Universal?**
    - Angular Universal is a technology for server-side rendering (SSR) of Angular applications. It helps improve SEO and reduces the initial load time by rendering the application on the server and sending the HTML to the client.
8. **How does Angular handle change detection?**
    - Angular's change detection mechanism checks the state of the application and updates the view when the model changes. It can be triggered manually or automatically through Angular zones.
9. **Explain the difference between `ngOnInit` and `constructor`.**
    - The constructor is used for class instantiation and dependency injection. `ngOnInit` is a lifecycle hook called after Angular initializes the component, suitable for initialization that requires Angular bindings.
10. **What are Angular modules and how do you use them?**
    - Angular modules are containers for components, directives

, pipes, and services. They are defined using the `@NgModule` decorator and can be imported/exported to share functionality across the application.

1. **How do you perform state management in Angular?**
    - State management can be performed using services or more advanced libraries like NgRx, which implements the Redux pattern for managing application state in a predictable and scalable manner.
2. **What is the purpose of the `RouterModule` in Angular?**
    - `RouterModule` provides the necessary services and directives for routing in Angular applications. It allows the configuration of application routes and navigation between different views.
3. **How do you use interceptors in Angular?**
    - Interceptors are used to intercept HTTP requests and responses. They can modify or log requests and responses. Interceptors are implemented by creating a class that implements `HttpInterceptor` and registering it in the providers array.
4. **What is the purpose of the `APP_INITIALIZER` token in Angular?**
    - `APP_INITIALIZER` is used to execute a function or a set of functions during the application initialization phase. It can be used to load configuration settings or perform any asynchronous task before the application starts.
5. **How do you create a custom directive in Angular?**
    - You create a custom directive by decorating a class with the `@Directive` decorator and implementing the desired functionality. You then apply the directive to DOM elements using its selector.
6. **Explain the use of `ng-content` and content projection.**
    - `ng-content` is used for content projection, allowing you to project content from a parent component into a child component's template. It enables the creation of reusable components with customizable content.
7. **How do you implement internationalization (i18n) in Angular?**
    - Internationalization in Angular is implemented using the Angular i18n package. It involves marking text for translation, extracting the translation messages, and providing translations in different languages.
8. **What are resolvers in Angular routing?**

- Resolvers are used to pre-fetch data before a route is activated. They can be implemented by creating a class that implements `Resolve` and configuring it in the route definition.

9. **How do you test Angular components?**
    - Angular components can be tested using Jasmine and Karma. You can write unit tests to test the component logic and DOM interactions, and use Angular testing utilities like `TestBed` for setting up the test environment.
10. **What are Angular animations and how do you implement them?**
    - Angular animations are built using the Angular animations module. You define animations using the `trigger`, `state`, `style`, `animate`, and `transition` functions, and apply them to components using the `[@triggerName]` binding.
11. **Explain the difference between `ViewChild` and `ContentChild`.**
    - `ViewChild` is used to query and access a child component, directive, or element within the view. `ContentChild` is used to query and access a projected content within a component.
12. **How do you manage dependencies in Angular?**
    - Dependencies in Angular are managed using dependency injection. Providers can be registered at different levels (module, component) and injected into classes where needed.
13. **What is Ahead-Of-Time (AOT) compilation in Angular?**
    - AOT compilation is a process that compiles Angular templates and components into optimized JavaScript code during the build time, improving the application's performance and reducing load time.
14. **How do you handle authentication and authorization in Angular applications?**
    - Authentication and authorization can be handled using services that interact with an authentication server. Guards like `AuthGuard` can be used to protect routes, and JWT tokens can be used to manage user sessions.
15. **What is the difference between `@ViewChild` and `@ContentChild`?**

- This question is similar to question 21, and the answer is the same: `ViewChild` is for accessing elements in the component's view, and `ContentChild` is for accessing projected content.

Hard to Very Hard Level Questions and Answers

1. **Explain the Angular compiler and its phases.**
   - The Angular compiler, or ngc, converts Angular templates into JavaScript code. The main phases include parsing, abstract syntax tree (AST) creation, optimization, and code generation.
2. **What is the Ivy renderer in Angular?**
   - Ivy is the latest Angular renderer that improves compilation speed, reduces bundle size, and enhances debugging. It uses incremental DOM and provides more granular control over rendering.
3. **How do you implement dynamic component loading?**
   - Dynamic component loading is implemented using `ComponentFactoryResolver` and `ViewContainerRef`. You can create and insert a component dynamically at runtime.
4. **What are Angular decorators and how do they work?**
   - Angular decorators are functions that modify the behavior of classes, properties, methods, or parameters. Examples include `@Component`, `@Directive`, `@Injectable`, and `@NgModule`.
5. **Explain the concept of dependency injection tokens.**
   - Dependency injection tokens are unique identifiers used for injecting dependencies. They can be strings, class types, or instances of `InjectionToken` for non-class dependencies.
6. **How do you create and use custom pipes in Angular?**
   - Custom pipes are created using the `@Pipe` decorator and implementing the `PipeTransform` interface. They are used in templates with the pipe operator `{{ value | customPipe }}`.
7. **What is the purpose of the `ChangeDetectorRef` class?**
   - `ChangeDetectorRef` is used to control the change detection process manually. It provides methods to mark components for check, detach, and reattach change detectors.

8. **How do you implement server-side rendering (SSR) in Angular?**
    - SSR is implemented using Angular Universal. You set up an Express server to handle requests and render the Angular application on the server, sending the HTML to the client.
9. **What are the best practices for Angular application architecture?**
    - Best practices include:
        - Using feature modules and lazy loading.
        - Keeping components small and focused.
        - Using services for business logic.
        - Implementing state management with NgRx.
        - Writing unit and integration tests.
        - Optimizing performance with AOT and OnPush strategy.
10. **How do you create a custom form control?**
    - Custom form controls are created by implementing `ControlValueAccessor` interface and providing the necessary methods (`writeValue`, `registerOnChange`, `registerOnTouched`, `setDisabledState`).
11. **What is the purpose of the `ngZone.run()` method?**
    - The `ngZone.run()` method is used to execute a function within Angular's zone, triggering change detection after the function executes. This is useful for running code outside Angular's zone and re-entering it when necessary.
12. **Explain how Angular handles security and what measures you should take.**
    - Angular handles security by:
        - Sanitizing inputs to prevent XSS attacks.
        - Using `HttpClient` to handle CSRF attacks.
        - Providing built-in mechanisms for handling authentication and authorization.
        - Using Content Security Policy (CSP).
        - Avoiding the use of `eval()` or dynamically created functions.

13. **How do you create a library in Angular?**
    - You can create a library using Angular CLI with the command:
      This sets up the necessary structure and configuration for developing and packaging the library.

      ```
      ng generate library library-name
      ```

14. **What are the differences between template-driven and reactive forms?**
    - Template-driven forms rely on directives in the template for form control and validation, suitable for simple forms. Reactive forms use a model-driven approach, providing more control and testability, suitable for complex forms.
15. **How do you implement a feature module in Angular?**
    - A feature module is implemented by creating a new module using Angular CLI and declaring components, directives, and pipes related to that feature. It can be lazy-loaded and configured in the routing module.
16. **Explain the use of `async` and `fakeAsync` in Angular testing.**
    - `async` is used to wrap asynchronous test code, allowing the test to wait for asynchronous operations to complete. `fakeAsync` simulates asynchronous passage of time, allowing you to control time-dependent code in tests.
17. **What is the role of the `Renderer2` class in Angular?**
    - `Renderer2` is an abstraction for DOM manipulation, allowing you to perform operations on elements, attributes, and styles in a platform-independent way.
18. **How do you optimize Angular applications for production?**
    - Optimization techniques include:
      - Enabling AOT compilation.
      - Using lazy loading.
      - Minifying and uglifying code.
      - Removing unnecessary polyfills.
      - Using the `OnPush` change detection strategy.

    - Compressing assets.

19. **What are Angular structural directives?**
    - Structural directives alter the DOM layout by adding or removing elements. Examples include `ngIf`, `ngFor`, and `ngSwitch`.
20. **Explain the concept of `trackBy` in `ngFor`.**
    - `trackBy` is a function used with `ngFor` to improve performance by tracking items in a collection by a unique identifier, reducing unnecessary re-renders when items change.
21. **How do you manage application state in Angular using NgRx?**
    - NgRx is a state management library based on Redux. It uses actions, reducers, and selectors to manage and access state in a predictable and scalable way.
22. **How do you handle large data sets in Angular applications?**
    - Techniques for handling large data sets include:
        - Implementing pagination.
        - Using virtual scrolling

.
- Optimizing data fetching with caching.
- Debouncing user inputs.

1. **What are Angular guards and how do you use them?**
    - Guards are services that control access to routes. They implement interfaces like `CanActivate`, `CanDeactivate`, `CanLoad`, and `Resolve` to conditionally allow navigation.
2. **How do you configure and use Angular CLI schematics?**
    - Angular CLI schematics are templates for generating code. You can create custom schematics to automate repetitive tasks by defining rules and using the Angular DevKit.
3. **Explain the concept of `ElementRef` in Angular.**
    - `ElementRef` is a wrapper around a native DOM element. It provides direct access to the element in the component's view. It is used for low-level DOM manipulations but should be used cautiously due to potential security risks.

Questions for 3-5 Years of Experience and Answers

1. **How have you implemented lazy loading in your Angular applications?**
    - Lazy loading is implemented by defining routes with the `loadChildren` property in the routing module. Feature modules are loaded only when the user navigates to the corresponding route, reducing the initial load time.
2. **Can you explain a challenging problem you solved using Angular?**
    - (Answer will vary based on personal experience. Provide a specific example of a complex feature or bug you resolved, detailing the problem, your approach, and the outcome.)
3. **How do you ensure high performance in your Angular applications?**
    - Ensuring high performance involves using techniques such as:
        - Enabling AOT compilation.
        - Implementing lazy loading.
        - Using `OnPush` change detection.
        - Minimizing DOM manipulations.
        - Using trackBy in `ngFor`.
4. **Describe your experience with state management in Angular.**
    - I have used NgRx for managing application state in Angular. This involves defining actions, reducers, and selectors to handle state changes predictably and immutably. It provides a single source of truth and makes state management more maintainable.
5. **How do you handle form validation in Angular?**
    - Form validation in Angular can be handled using:
        - Template-driven forms with built-in validators and custom validators.
        - Reactive forms with `FormBuilder`, `FormGroup`, `FormControl`, and custom validators. Reactive forms provide more control over validation logic and error handling.
6. **Can you provide an example of how you've used Angular services?**

- Angular services are used for encapsulating business logic and data access. For example, I created a `UserService` to handle user authentication, fetch user data, and manage user sessions across the application.

7. **How have you implemented authentication in Angular applications?**
    - Authentication is implemented using JWT tokens. The login component sends user credentials to the backend, receives a token, and stores it in localStorage. HTTP interceptors add the token to requests, and guards protect routes based on authentication status.
8. **Describe a scenario where you used Angular routing.**
    - I implemented Angular routing in a multi-page application where different routes were configured for the home, about, contact, and profile pages. Lazy loading was used for feature modules to improve performance.
9. **What strategies have you used for error handling in Angular applications?**
    - Error handling strategies include:
        - Using `catchError` with HTTP requests.
        - Implementing a global error handler.
        - Displaying user-friendly error messages.
        - Logging errors to an external service for monitoring.
10. **How do you implement responsive design in Angular?**
    - Responsive design is implemented using CSS media queries, Flexbox, and Grid layouts. Angular Material components and BreakpointObserver from Angular CDK can also be used to create responsive UIs.
11. **What is your experience with testing Angular applications?**
    - I have experience writing unit tests for components, services, and pipes using Jasmine and Karma. I also write integration tests using Protractor or Cypress to ensure end-to-end functionality.
12. **How have you managed dependencies in your Angular projects?**
    - Dependencies are managed using Angular's dependency injection system. Services are provided in modules or components, and Angular

CLI helps manage third-party libraries and dependencies in `package.json`.

13. **Can you describe a situation where you optimized an Angular application?**
    - (Answer will vary based on personal experience. Describe a specific instance where you improved performance, such as reducing load time, implementing lazy loading, or optimizing change detection.)
14. **How do you handle versioning and updates in Angular projects?**
    - Versioning is managed using Git for source control. Updates are handled by following Angular's update guide, using `ng update` to manage dependencies, and testing thoroughly to ensure compatibility.
15. **What is your approach to component communication in Angular?**
    - Component communication is handled using `@Input` and `@Output` for parent-child communication, services with shared state for sibling components, and NgRx for managing global state.
16. **How do you manage complex forms in Angular?**
    - Complex forms are managed using reactive forms. I use `FormBuilder` to create form groups and controls, implement custom validators, and handle dynamic form fields.
17. **Describe your experience with Angular Universal.**
    - I have implemented server-side rendering with Angular Universal to improve SEO and reduce initial load time. This involved setting up an Express server, configuring Angular Universal, and ensuring the application is pre-rendered on the server.
18. **How have you used Angular animations in your projects?**
    - Angular animations are used to enhance user experience by adding transitions and animations to elements. I have used the Angular animations module to create slide, fade, and expand/collapse animations.
19. **What is your approach to debugging Angular applications?**
    - Debugging is done using browser developer tools, Angular's error messages, and logging. I also use Augury, an Angular debugging extension, to inspect component trees and states.

20. **How do you implement internationalization (i18n) in Angular?**
    - Internationalization is implemented using Angular's i18n package. This involves marking text for translation, extracting messages, providing translations, and configuring the application to use different locales.
21. **Describe your experience with custom directives and pipes.**
    - I have created custom directives to add reusable behaviors to elements, such as highlighting text or handling click events. Custom pipes are used for formatting data, such as date formatting or currency conversion.
22. **How do you use RxJS in your Angular applications?**
    - RxJS is used for handling asynchronous operations and streams of data. I use Observables to manage HTTP requests, implement reactive forms, and handle real-time updates with WebSockets.
23. **What are some best practices you follow in Angular development?**
    - Best practices include:
        - Using feature modules and lazy loading.
        - Writing unit and integration tests.
        - Keeping components small and focused.
        - Using services for business logic.
        - Following Angular style guide.
        - Regularly updating dependencies.
24. **How do you integrate third-party libraries with Angular?**
    - Third-party libraries are integrated using Angular CLI to install and configure them. I follow the library documentation for setup and usage, ensuring compatibility with Angular.
25. **Can you describe a project where you played a significant role in using Angular?**
    - (Answer will vary based on personal experience. Describe a specific project where you contributed significantly, detailing your role, the challenges faced, and the outcomes achieved.)