Accenture Angular Interview Questions

Basic

1. What is Angular?

Answer: Angular is a front-end framework developed by Google for building dynamic, single-page web applications. It allows for the creation of rich user interfaces using HTML, CSS, and TypeScript.

2. What is TypeScript in Angular?

Answer: TypeScript is a superset of JavaScript that provides optional static typing. Angular uses TypeScript to improve code quality and enhance error detection during development.

3. What is a Component in Angular?

Answer: A component is the basic building block of an Angular application. It controls a part of the UI and is composed of HTML, CSS, and TypeScript code. Each component is defined by a `@Component` decorator.

4. What is a Module in Angular?

Answer: A module is a container that groups related components, services, directives, and pipes. It is represented by the `NgModule` decorator and helps in organizing the code.

5. Explain the concept of Data Binding in Angular.

Answer: Data binding is a mechanism in Angular that connects the UI (view) and the business logic (model). It enables communication between the component class and the template using one-way or two-way data binding.

6. What are Directives in Angular?

Answer: Directives are instructions in Angular that modify the behavior or appearance of HTML elements. They come in three types: structural (`*ngIf`, `*ngFor`), attribute, and custom directives.

7. What is the purpose of the `ngIf` directive?

Answer: The `ngIf` directive conditionally renders or removes an element from the DOM based on a Boolean expression. If the condition is true, the element is added; otherwise, it is removed.

8. What is a Service in Angular?

Answer: A service is a class that contains reusable logic and is used for sharing data or functionality between different parts of the application. Services are often injected into components using Dependency Injection.

9. What is Dependency Injection in Angular?

Answer: Dependency Injection (DI) is a design pattern used to manage dependencies in Angular. It allows components and services to request dependencies from Angular's injector, improving modularity and testability.

10. What is Routing in Angular?

Answer: Routing is a mechanism in Angular that allows navigation between different views or components. It is managed by the Angular Router, which maps URLs to components.

11. What is the role of the `@NgModule` decorator?

Answer: The `@NgModule` decorator defines an Angular module. It organizes an application into cohesive blocks, such as components, directives, and services, and declares their dependencies.

12. What is a Template in Angular?

Answer: A template in Angular is an HTML view with additional Angular-specific syntax, such as interpolation and directives. It defines how the component's data should be displayed in the browser.

13. What are Pipes in Angular?

Answer: Pipes are used to transform data in Angular templates. For example, the `date` pipe can format a date object, and the `uppercase` pipe can transform text to uppercase.

14. What is the difference between `ngIf` and `ngSwitch`?

Answer: The `ngIf` directive is used for simple conditional rendering, while `ngSwitch` is used to switch between multiple views based on a specific condition, similar to a `switch` statement in programming.

15. What is Angular CLI?

Answer: Angular CLI (Command Line Interface) is a tool for scaffolding, building, and managing Angular projects. It helps in generating components, services, and modules, and automates the build process.

16. What is Lazy Loading in Angular?

Answer: Lazy loading is a technique in Angular that delays the loading of a feature module until it is needed, improving application performance by reducing the initial load time.

17. How do you handle forms in Angular?

Answer: Angular provides two approaches for handling forms: Template-driven forms (simpler, declarative approach using the template) and Reactive forms (more flexible, uses model-driven approach with `FormGroup` and `FormControl`).

18. What is Interpolation in Angular?

Answer: Interpolation is a way to bind data from the component class to the template. It uses the double curly braces syntax (`{{ }}`) to display dynamic content in the HTML.

19. What is Two-way Data Binding in Angular?

Answer: Two-way data binding allows automatic synchronization of data between the component and the view. It is achieved using the `[(ngModel)]` directive, which binds the model and the view together.

20. What is an Angular Lifecycle Hook?

Answer: Lifecycle hooks are methods that allow developers to tap into key events in a component's life cycle, such as creation, updates, and destruction. Examples include `ngOnInit`, `ngOnChanges`, and `ngOnDestroy`.

Intermediate

1. What is Change Detection in Angular?

Answer: Change detection in Angular is the mechanism by which the framework checks the state of the application and updates the DOM when data changes. Angular uses zone.js to detect when asynchronous tasks complete, and it then triggers change detection automatically.

2. What is the difference between `@Input` and `@Output` in Angular?

Answer: `@Input` allows a parent component to pass data to a child component, while `@Output` allows the child component to send data back to the parent component by emitting events through an `EventEmitter`.

3. Explain the purpose of `ngOnChanges` in Angular.

Answer: `ngOnChanges` **is a lifecycle hook that is called whenever an input-bound property of a component changes. It allows you to react to changes to** `@Input()` **data in child components.**

4. What is the difference between Observable and Promise in Angular?

Answer: Observables are lazy and can handle multiple values over time (streams), while Promises are eager and resolve to a single value. Observables offer more flexibility, such as cancellation and retrying operations, and are part of Angular's reactive programming model.

5. How does Angular handle error handling in HTTP requests?

Answer: Angular provides a built-in `HttpClient` **module that supports error handling using the** `catchError` **operator from RxJS. This allows developers to gracefully handle HTTP errors like network failures or incorrect API calls.**

6. What is AOT (Ahead-of-Time) Compilation in Angular?

Answer: AOT compilation pre-compiles the Angular templates and components at build time, rather than at runtime. This reduces the size of the application, increases the loading speed, and catches template errors early.

7. What is Dependency Injection (DI) Hierarchical Injector in Angular?

Answer: Angular uses a hierarchical dependency injection system. The injectors form a tree, with the root injector at the top. Components can have their injectors, and Angular resolves dependencies by looking for the required service in the local injector, moving upward if needed.

8. What are Guards in Angular Routing?

Answer: Guards in Angular are used to control navigation to and from routes. There are four types: `CanActivate`, `CanActivateChild`, `CanDeactivate`, **and** `Resolve`**. They allow or deny access to routes based on logic, such as authentication.**

9. What is a Singleton Service in Angular?

Answer: A singleton service in Angular is a service that is instantiated only once and shared across the entire application. By providing a service in the root module (`@Injectable({ providedIn: 'root' })`), it becomes a singleton.

10. Explain ViewEncapsulation in Angular.

Answer: ViewEncapsulation defines how styles are applied to components. Angular provides three encapsulation modes: `Emulated` (default, styles are scoped), `None` (styles are global), and `ShadowDOM` (uses the shadow DOM to encapsulate styles).

11. What is a Resolver in Angular?

Answer: A resolver is used in Angular routing to pre-fetch data before a route is activated. It allows data to be retrieved asynchronously and provided to the component before rendering, ensuring that the component has the necessary data upon loading.

12. How does Angular handle lazy loading of modules?

Answer: Angular enables lazy loading by defining routes in the `RouterModule` using the `loadChildren` property. This allows feature modules to be loaded only when the associated route is visited, improving the application's initial load performance.

13. What is `ng-content` and how is it used?

Answer: `ng-content` is used for content projection, which allows you to insert dynamic content from a parent component into the template of a child component. It acts as a placeholder in the child component, and the content is projected from the parent component.

14. What are Angular Pipes, and how do you create a custom pipe?

Answer: Angular Pipes transform data in templates. To create a custom pipe, use the `@Pipe` decorator and implement the `PipeTransform` interface. Custom pipes allow for reusable transformation logic for specific data formats.

15. Explain how you would optimize performance in an Angular application.

Answer: Performance optimization techniques in Angular include using AOT compilation, lazy loading modules, employing OnPush change detection strategy, debouncing user input, optimizing bundle sizes using tools like Webpack, and using pure pipes for transformation.

16. What is `@ViewChild` and `@ContentChild` in Angular?

Answer: `@ViewChild` is used to access a template reference or a child component instance in the view, while `@ContentChild` is used to query and get references to projected content passed into the component via `<ng-content>`.

17. What is the role of `FormBuilder` in Angular Reactive Forms?

Answer: `FormBuilder` is a service in Angular used to create and manage forms more efficiently in reactive forms. It simplifies the process of creating `FormGroup` and `FormControl` instances with less boilerplate code.

18. Explain the `async` pipe in Angular.

Answer: The `async` pipe in Angular automatically subscribes to an observable or promise and returns its latest value. It also handles unsubscription when the component is destroyed, avoiding memory leaks.

19. What is the purpose of `RouterModule.forRoot()` and `RouterModule.forChild()`?

Answer: `RouterModule.forRoot()` is used to register the main application routes and initialize the router at the root level, whereas `RouterModule.forChild()` is used to register routes within a feature module in a lazy-loaded module.

20. What is the purpose of `ngZone` in Angular?

Answer: `ngZone` is an Angular service used to control the scope of asynchronous operations that can trigger change detection. By using `ngZone.runOutsideAngular()`, you can perform actions that won't trigger change detection, improving performance for heavy operations.

Advanced

1. What is the Change Detection Strategy in Angular, and how can you optimize it?

Answer: Angular provides two change detection strategies: `Default` and `OnPush`. In the default mode, Angular checks every component's change, but `OnPush` optimizes performance by checking only when the component's inputs change. To optimize, use `OnPush` for components that don't need frequent updates and immutable data structures.

2. Explain how Angular's Dependency Injection works at a multi-level hierarchy.

Answer: Angular's Dependency Injection (DI) system forms a hierarchy, where each component has its injector. Services provided at the module level (root injector) are available throughout the app, while services provided at the component level are specific to that component and its children.

3. What is the difference between `@HostListener` and `@HostBinding`?

Answer: `@HostListener` is used to listen for events on the host element, allowing you to react to events like clicks or key presses. `@HostBinding` is used to bind a property (e.g., class, style) of the host element to a component property dynamically.

4. How would you implement state management in Angular without using NgRx?

Answer: Without NgRx, you can implement state management using Angular services as singletons to store and manage state. Use `BehaviorSubject` from RxJS to store the application state and emit updates. Components can subscribe to this state and react accordingly.

5. What are the key differences between Reactive Forms and Template-Driven Forms in Angular?

Answer: Reactive Forms offer more control and are model-driven, making them suitable for complex forms. They use `FormControl`, `FormGroup`, and `FormBuilder` classes. Template-Driven Forms are easier for simple forms, rely heavily on the template, and use directives like `ngModel`.

6. What is Tree Shaking in Angular?

Answer: Tree shaking is a build optimization technique that removes unused code during the build process. Angular, using tools like Webpack, eliminates dead code to reduce the size of the final bundle, improving performance.

7. How does Angular handle memory leaks, and how can you prevent them?

Answer: Angular's automatic garbage collection helps manage memory, but leaks can still occur, especially with subscriptions. To prevent memory leaks, always unsubscribe from observables using `takeUntil`, `async` pipe, or `ngOnDestroy()` lifecycle hook for manual unsubscription.

8. What is Dynamic Component Loading in Angular, and how can you achieve it?

Answer: Dynamic component loading allows components to be created and inserted into the DOM at runtime. This can be achieved using Angular's `ComponentFactoryResolver` or more recently, `ViewContainerRef.createComponent()`.

9. How does Angular handle animations, and what is the role of the `@angular/animations` module?

Answer: Angular's animation system is built on top of the `@angular/animations` module, which allows the definition of animations using triggers and states. Animations are applied by associating triggers with elements in templates and controlling transitions between states.

10. Explain the role of `ngZone.runOutsideAngular()` and its benefits.

Answer: `ngZone.runOutsideAngular()` allows you to execute code that doesn't trigger change detection. It is beneficial when performing operations that don't impact the UI, such as heavy computations, to prevent unnecessary change detection cycles and improve performance.

11. What are custom structural directives, and how do you create one in Angular?

Answer: Custom structural directives manipulate the DOM structure (e.g., adding or removing elements). To create one, use the `@Directive` decorator and implement `TemplateRef` and `ViewContainerRef`. For example, create a directive that shows/hides content based on a condition.

12. What are Zone.js and its significance in Angular?

Answer: Zone.js is a library that patches asynchronous operations (e.g., `setTimeout`, HTTP requests) and notifies Angular when these operations complete. Angular uses Zone.js to manage change detection by triggering updates when tasks complete.

13. How do you optimize bundle size in an Angular application?

Answer: Bundle size optimization strategies include lazy loading, using AOT compilation, enabling Tree Shaking, minimizing external libraries, compressing assets, and using Webpack to split bundles.

14. What is Ivy Renderer, and how does it differ from the older View Engine in Angular?

Answer: Ivy is Angular's latest rendering engine, designed to improve compilation speed, reduce bundle size, and provide more efficient change detection. It allows better tree-shaking and lazy loading. Ivy also provides new APIs for dynamic component rendering.

15. How does Angular handle i18n (Internationalization) and what are its key features?

Answer: Angular provides built-in support for i18n via the `@angular/localize` package. It allows you to translate content into different languages by extracting messages using `ng extract-i18n`, creating translation files, and then building the app for different locales.

16. What is the difference between `Subject`, `BehaviorSubject`, `ReplaySubject`, and `AsyncSubject` in Angular?

Answer:

- **`Subject`: Emits data to all subscribers from the moment of subscription.**
- **`BehaviorSubject`: Emits the latest value to new subscribers and subsequent values to all.**
- **`ReplaySubject`: Replays a specified number of previous values to new subscribers.**
- **`AsyncSubject`: Emits only the last value when the observable completes.**

17. What is the purpose of the `Renderer2` service in Angular?

Answer: `Renderer2` is a service for safely manipulating DOM elements and attributes. It provides an abstraction layer that ensures compatibility across different rendering environments (e.g., server-side rendering with Angular Universal).

18. How do you test Angular components that depend on services using `TestBed`?

Answer: Angular provides the `TestBed` utility for configuring and initializing the environment for unit testing. Services can be mocked using `TestBed.configureTestingModule()` with `spyOn()` or `MockService` to test components independently of service implementations.

19. What is `ngx-translate` and how do you integrate it into an Angular application?

Answer: `ngx-translate` is a library that provides internationalization for Angular applications. It allows you to load translations asynchronously from JSON files and switch languages dynamically using services like `TranslateService` and `TranslatePipe`.

20. What is Angular Universal, and what benefits does server-side rendering (SSR) provide?

Answer: Angular Universal enables server-side rendering of Angular applications. It improves performance by rendering HTML on the server, reducing the time to first contentful paint (FCP), improving SEO, and enhancing the user experience for slow networks or devices.