Shailesh Diwanji

30 Angular Interview Questions

For Experienced Candidates

Complete Theory Questions with Detailed Answers

1. What is the difference between AOT and JIT compilation?

JIT (Just-In-Time) compilation compiles the application in the browser at runtime, which increases initial load time but is useful for development with faster build times.

AOT (Ahead-of-Time) compiles the application during the build process, resulting in:

- Faster rendering (no compilation in browser)
- Smaller bundle sizes (compiler not shipped)
- Early error detection during build
- Better security (no eval or new Function)

AOT is preferred for production deployments.

2. Explain the Angular change detection mechanism and its strategies.

Angular's change detection checks component trees for data changes and updates the DOM accordingly. It runs after asynchronous events like HTTP requests, timers, or user interactions.

Two strategies:

- **Default:** Checks the entire component tree from root to leaves whenever an event occurs. Safe but can be inefficient for large applications.
- **OnPush:** Only checks when:
  - Input properties change (by reference)
  - Events are emitted from the component
  - Manual detection is triggered (markForCheck/detectChanges)
  - Observables emit with async pipe

OnPush improves performance by reducing unnecessary checks.

3. What are the different types of data binding in Angular?

Angular supports four types of data binding:

- **Interpolation:** {{value}} - One-way from component to view
- **Property Binding:** [property]="value" - One-way from component to view
- **Event Binding:** (event)="handler()" - One-way from view to component
- **Two-way Binding:** [(ngModel)]="value" - Bidirectional sync between component and view

4. Explain ViewChild, ViewChildren, ContentChild, and ContentChildren.

- **ViewChild/ViewChildren:** Query elements from the component's own template (view). Used to access child components, directives, or DOM elements defined in the template.
- **ContentChild/ContentChildren:** Query elements projected into the component via <ng-content>. Used to access content passed from parent.

ViewChild and ContentChild return a single element, while ViewChildren and ContentChildren return a QueryList of elements.

5. What is the difference between Subject, BehaviorSubject, ReplaySubject, and AsyncSubject?

- **Subject:** Basic observable that multicasts to observers, no initial value, new subscribers don't receive previous values
- **BehaviorSubject:** Requires initial value, emits current value to new subscribers immediately
- **ReplaySubject:** Buffers specified number of values and replays them to new subscribers
- **AsyncSubject:** Only emits the last value when the sequence completes

6. Explain Angular's Dependency Injection hierarchy.

Angular has a hierarchical DI system with multiple injector levels:

- **NullInjector:** Top of the tree, throws error if dependency not found
- **PlatformInjector:** Platform-level services shared across apps
- **RootInjector:** Application-level (providedIn: 'root')
- **ModuleInjector:** Module-level providers
- **ElementInjector:** Component/directive level providers

Angular searches for dependencies from the current injector up the tree until found or NullInjector is reached.

7. What are Angular Guards and their types?

Guards control navigation and access to routes. Types include:

- **CanActivate:** Determines if a route can be activated
- **CanActivateChild:** Determines if child routes can be activated
- **CanDeactivate:** Determines if user can leave a route (e.g., unsaved changes)
- **Resolve:** Pre-fetches data before route activation
- **CanLoad:** Determines if a lazy-loaded module can be loaded
- **CanMatch:** (Angular 14+) Determines if a route can be matched

8. What is the difference between ngOnInit and constructor?

- **Constructor:** TypeScript class constructor, called first when the class is instantiated. Used for dependency injection. Input properties are not yet available.
- **ngOnInit:** Angular lifecycle hook called after the first ngOnChanges. Input properties are initialized. Best place for initialization logic that depends on inputs or needs to call services.

Best practice: Use constructor only for DI, use ngOnInit for component initialization.

9. Explain the purpose of NgZone and when to use it.

NgZone is a wrapper around Zone.js that helps Angular know when to trigger change detection. It tracks asynchronous operations.

Use cases:

- **runOutsideAngular():** Run code outside Angular's zone to prevent triggering change detection (performance optimization for frequent events)
- **run():** Bring code back into Angular's zone to trigger change detection

Useful for third-party libraries or performance-critical operations that don't need to update the UI.

10. What is the difference between Renderer2 and ElementRef?

- **ElementRef:** Provides direct access to DOM elements. Security risk and breaks server-side rendering. Should be avoided when possible.
- **Renderer2:** Angular's abstraction layer for DOM manipulation. Platform-agnostic, works with server-side rendering and Web Workers. Provides methods like createElement, setStyle, setAttribute, etc.

Always prefer Renderer2 over direct DOM manipulation for better security and cross-platform compatibility.

11. Explain lazy loading and its benefits.

Lazy loading loads feature modules on-demand rather than at application startup.
Implemented using loadChildren in routing configuration.

Benefits:

- Reduced initial bundle size
- Faster initial load time
- Better resource utilization
- Improved performance for large applications

Example: `{ path: 'admin', loadChildren: () => import('./admin/admin.module').then(m => m.AdminModule) }`

12. What is the difference between template-driven and reactive forms?

Template-driven forms:

- Logic in template using directives (ngModel)
- Two-way data binding
- Asynchronous, less testable
- Good for simple forms

Reactive forms:

- Logic in component using FormControl, FormGroup
- Explicit, immutable data model
- Synchronous, more testable
- Better for complex forms with dynamic validations
- More scalable and maintainable

13. Explain Angular's trackBy function in ngFor.

trackBy is a function used with ngFor to improve performance by helping Angular identify which items have changed, been added, or removed.

Without trackBy, Angular re-renders all items when the array reference changes. With trackBy, Angular only re-renders items that actually changed.

Example:

```
trackByFn(index, item) { return item.id; }
```

```
<div *ngFor="let item of items; trackBy: trackByFn">
```

14. What are Angular Pipes and can you create custom pipes?

Pipes transform data in templates. Angular provides built-in pipes like date, uppercase, currency, async, etc.

Custom pipes: Created using @Pipe decorator and implementing PipeTransform interface.

Pure vs Impure pipes:

- **Pure (default):** Only executes when input value or reference changes. More performant.
- **Impure:** Executes on every change detection cycle. Use sparingly as it impacts performance.

15. Explain the difference between @Input and @Output decorators.

- **@Input:** Allows parent component to pass data to child component. Enables property binding from parent to child.
- **@Output:** Allows child component to emit events to parent component. Uses EventEmitter to send data from child to parent.

Together they enable component communication: @Input for downward data flow, @Output for upward event flow.

16. What is Angular Universal and its benefits?

Angular Universal enables Server-Side Rendering (SSR) for Angular applications.

Benefits:

- Improved SEO (search engines can crawl content)
- Faster initial page load (pre-rendered HTML)
- Better social media preview (Open Graph tags visible)
- Improved performance on low-powered devices
- Better perceived performance

17. Explain HttpInterceptor and its use cases.

HttpInterceptor intercepts HTTP requests and responses, allowing you to modify them before they're sent or after they're received.

Common use cases:

- Adding authentication tokens to requests
- Logging requests and responses
- Error handling centrally
- Request/response transformation
- Caching
- Loading indicators

Implements HttpInterceptor interface with intercept() method.

18. What is the difference between NgModule and standalone components?

NgModule (traditional):

- Components declared in module's declarations array
- Dependencies imported in module's imports array
- More boilerplate code

Standalone components (Angular 14+):

- standalone: true flag in component decorator
- Direct imports in component's imports array
- Less boilerplate, more modular
- Can be used without NgModule
- Recommended approach for new applications

19. Explain RxJS operators: map, switchMap, mergeMap, and concatMap.

- **map:** Transforms emitted values. Doesn't handle observables.
- **switchMap:** Cancels previous inner observable when new value arrives. Best for search/typeahead.
- **mergeMap:** Runs inner observables concurrently, doesn't cancel. Good for parallel operations.
- **concatMap:** Queues inner observables, processes sequentially. Maintains order.

Choose based on whether you need cancellation, concurrency, or sequential execution.

20. What is Zone.js and its role in Angular?

Zone.js is a library that creates execution contexts (zones) to track asynchronous operations. Angular uses it to know when to trigger change detection.

Zone.js patches asynchronous APIs (setTimeout, Promise, XHR, etc.) to notify Angular when async operations complete, triggering change detection automatically.

Angular 18+ introduced experimental zoneless mode for better performance.

21. Explain Angular's content projection with ng-content.

Content projection allows you to insert content from a parent component into a child component's template using <ng-content>.

Types:

- **Single-slot:** <ng-content></ng-content>
- **Multi-slot:** <ng-content select=".class"></ng-content>
- **Conditional:** <ng-content select="[condition]"></ng-content>

Useful for creating reusable wrapper components like cards, modals, tabs.

22. What are Angular Signals and their advantages?

Signals (Angular 16+) are a reactive primitive for managing state changes.

Advantages:

- Fine-grained reactivity without Zone.js
- Better performance (targeted updates)
- Simpler mental model than RxJS for simple cases
- Computed values automatically update
- Effects run when dependencies change

Types: signal() for writable, computed() for derived, effect() for side effects.

23. Explain the difference between providedIn: 'root' and providers array.

providedIn: 'root':

- Service registered at application root level
- Singleton across the app
- Tree-shakable (removed if not used)
- Recommended approach

providers array:

- Component-level: New instance per component
- Module-level: Shared within module
- Not tree-shakable
- Use when you need multiple instances or limited scope

24. What is ControlValueAccessor and when do you use it?

ControlValueAccessor is an interface that bridges custom form controls with Angular's form APIs (both template-driven and reactive forms).

Implements four methods:

- writeValue(): Updates control value from model
- registerOnChange(): Registers callback for value changes
- registerOnTouched(): Registers callback for touch events
- setDisabledState(): Handles disabled state

Used when creating custom input components that need to work with Angular forms.

25. Explain Angular's animation system.

Angular animations are built on Web Animations API, defined using @angular/animations.

Key concepts:

- **trigger():** Defines animation name
- **state():** Defines end state styles
- **transition():** Defines animation between states
- **style():** Defines CSS styles
- **animate():** Defines timing and easing

Supports enter/leave animations, route transitions, and complex sequences.

26. What is the difference between ng-template, ng-container, and ng-content?

- **ng-template:** Defines a template that isn't rendered by default. Used with structural directives or TemplateRef. Doesn't add to DOM.
- **ng-container:** Logical grouping element that doesn't create DOM element. Useful for applying directives without extra markup.
- **ng-content:** Used for content projection, defines insertion point for transcluded content.

27. Explain Angular's route resolvers and their benefits.

Resolvers pre-fetch data before a route is activated, ensuring data is available when the component initializes.

Benefits:

- Component loads with data already available
- Prevents empty/loading states in components
- Centralizes data fetching logic
- Can handle errors before navigation
- Better user experience

Implement Resolve interface and return Observable, Promise, or value.

28. What are dynamic components and how do you create them?

Dynamic components are created at runtime rather than being declared in templates.

Creation methods:

- **ViewContainerRef:** createComponent() to instantiate components dynamically
- **ComponentFactoryResolver:** (deprecated) Old approach before Ivy
- **NgComponentOutlet:** Directive for simpler cases

Use cases: modals, tooltips, dynamic forms, plugin systems.

29. Explain memory leak scenarios in Angular and how to prevent them.

Common causes:

- Unsubscribed Observables
- Event listeners not removed
- Timers/intervals not cleared
- DOM references held in component

Prevention strategies:

- Use async pipe (auto-unsubscribes)
- Unsubscribe in ngOnDestroy
- Use takeUntil/take operators
- Clear intervals/timeouts in ngOnDestroy
- Remove event listeners
- Use DestroyRef (Angular 16+)

30. What are the best practices for optimizing Angular application performance?

- Use OnPush change detection strategy
- Implement lazy loading for feature modules
- Use trackBy with ngFor
- Enable AOT compilation
- Use pure pipes instead of functions in templates
- Implement virtual scrolling for large lists
- Optimize bundle size (tree-shaking, code splitting)
- Use async pipe to auto-unsubscribe
- Detach change detector when not needed
- Preload lazy-loaded modules strategically
- Use Web Workers for heavy computations
- Implement service workers for caching
- Optimize images and assets
- Use Angular DevTools for profiling