Infosys Angular Developer Interview Preparation Guide (4 Years Experience)

**Q: What are the key building blocks of Angular?**

A: Angular applications are built using components, templates, modules, services, and directives. Components control views, and services handle business logic.

**Q: Difference between Reactive Forms and Template Driven Forms?**

A: Reactive Forms are more scalable and suitable for complex forms; they use FormBuilder and FormGroup. Template Driven Forms are simpler and use directives like ngModel.

**Q: What is the difference between Subject and BehaviorSubject in RxJS?**

A: Subject does not hold a current value, while BehaviorSubject stores the last emitted value and emits it immediately to new subscribers.

**Q: How does Angular implement lazy loading?**

A: Lazy loading is implemented using Angular’s router by loading feature modules asynchronously using the `loadChildren` property.

**Q: What are lifecycle hooks in Angular?**

A: `ngOnInit`, `ngOnDestroy`, `ngOnChanges`, etc. are lifecycle hooks that allow developers to act during specific phases of component lifecycle.

**Q: What is the difference between interface and type in TypeScript?**

A: Interfaces are mainly used for object shape declaration and support declaration merging. Types can represent unions, intersections, and primitives.

**Q: Explain generics in TypeScript with an example.**

A: Generics allow creating reusable components. Example: `function identity<T>(arg: T): T { return arg; }` can work with any type.

**Q: What is a closure in JavaScript?**

A: A closure is a function that has access to its outer function’s scope even after the outer function has returned.

**Q: Difference between == and === in JavaScript?**

A: `==` checks for value equality with type coercion, while `===` checks for both value and type equality.

**Q: What is event delegation?**

A: Event delegation is a technique of attaching a single event listener to a parent element to handle events from its child elements.

**Q: How does Flexbox work in CSS?**

A: Flexbox is a layout model that allows elements to align and distribute space within a container. Use `display: flex`, and then control layout with properties like `justify-content` and `align-items`.

**Q: Difference between relative, absolute, fixed, and sticky positioning in CSS?**

A: Relative is based on its normal position, absolute is based on the nearest positioned ancestor, fixed is relative to the viewport, and sticky toggles between relative and fixed based on scroll.

**Q: How do SCSS variables and mixins help in styling?**

A: Variables store values like colors or fonts for reuse. Mixins are reusable pieces of styles that can accept arguments to generate dynamic CSS.

# Detailed Infosys Angular Developer Interview Questions & Answers

## 1. Angular (14+ or current version)

**Q: What is the difference between ngIf and \*ngIf?**

A: `ngIf` is a directive used to conditionally include a template in the DOM. `\*ngIf` is shorthand for structural directive syntax which creates or removes a portion of the DOM tree.

**Q: What is the role of ng-template?**

A: `ng-template` is used to define template fragments that are not rendered by default. It works with structural directives like \*ngIf to control view rendering.

**Q: Explain Subject, BehaviorSubject, ReplaySubject with use cases.**

A: `Subject` emits to current subscribers only. `BehaviorSubject` stores and emits the last value to new subscribers. `ReplaySubject` replays a specified number of emitted values to new subscribers.

**Q: How does Angular handle two-way binding?**

A: Angular uses the `[(ngModel)]` syntax to implement two-way data binding between the DOM and component class properties.

**Q: How do you share data between unrelated components?**

A: Use a shared service with `Subject` or `BehaviorSubject`, or a state management library like NgRx to share data between unrelated components.

**Q: How do you implement lazy loading?**

A: Lazy loading is done using Angular Router’s `loadChildren` property in route configuration to load modules asynchronously.

**Q: What are observables and how are they different from promises?**

A: Observables can emit multiple values over time and support operators like `map`, `filter`. Promises emit only one value and cannot be cancelled.

**Q: How do you handle form validation in reactive forms?**

A: Use `FormControl`, `FormGroup`, and built-in validators like `Validators.required`. Custom validators can be added for complex logic.

## 2. TypeScript

**Q: What’s the difference between interface and type?**

A: `interface` is used to define object shapes and supports declaration merging. `type` can represent unions, intersections, and primitive types.

**Q: How does type inference work?**

A: TypeScript automatically infers types when explicit type annotations are not provided, based on the value assigned.

**Q: What are generics in TypeScript? Give a practical use case.**

A: Generics allow defining reusable functions or classes with variable types. Example: `function identity<T>(arg: T): T { return arg; }`

**Q: What is the difference between any, unknown, and never?**

A: `any` disables type checking. `unknown` is type-safe but requires checking before use. `never` represents unreachable code or functions that throw.

**Q: How would you create a strongly-typed function that accepts multiple types?**

A: Use generics or union types like: `function combine(a: number | string, b: number | string): string | number { ... }`

## 3. JavaScript (ES6+)

**Q: What is the output of this in arrow vs regular function?**

A: Arrow functions don't have their own `this` and inherit it from the lexical scope. Regular functions have their own `this` depending on how they're called.

**Q: Explain event delegation with a practical example.**

A: Attach a single event listener to a parent element and use `event.target` to identify the clicked child. Improves performance for dynamic elements.

**Q: Difference between == and ===**

A: `==` allows type coercion. `===` checks both value and type strictly.

**Q: What is a closure and how does it work?**

A: A closure is a function that retains access to its lexical scope, even when executed outside that scope.

**Q: How does the event loop work in JavaScript?**

A: It manages the execution of tasks and asynchronous callbacks. The call stack processes tasks, and async callbacks are handled in the event queue.

## 4. HTML/CSS/SCSS/Bootstrap

**Q: How does CSS specificity work?**

A: Specificity is calculated based on the number of ID, class, and element selectors. Higher specificity rules override lower ones.

**Q: What is the difference between em, rem, %, px?**

A: `em` is relative to the parent font-size, `rem` is relative to root, `%` is relative to parent dimension, and `px` is absolute.

**Q: How do you center a div horizontally and vertically?**

A: Use Flexbox: `display: flex; justify-content: center; align-items: center; height: 100vh;`

**Q: What’s the difference between relative, absolute, fixed, sticky?**

A: `relative`: relative to itself; `absolute`: to nearest positioned ancestor; `fixed`: to viewport; `sticky`: toggles between relative and fixed on scroll.

**Q: How do you write responsive layouts using SCSS or Bootstrap?**

A: Use Bootstrap grid system or SCSS media queries. SCSS: `@media (max-width: 768px) { ... }`; Bootstrap: `col-md-6`, `d-flex` utilities.

## 5. Managerial Round Preparation

**Q: Can you walk me through a challenging bug you solved?**

A: Yes, in a recent Angular app, we faced a performance issue with heavy DOM updates. I optimized it using OnPush change detection and reduced unnecessary subscriptions.

**Q: How do you manage performance optimization in a large Angular app?**

A: Use lazy loading, OnPush strategy, detach change detection, memoization, and RxJS operators to control emissions.

**Q: How do you ensure quality in deliverables?**

A: I follow code reviews, unit tests, linting, use pre-commit hooks, and maintain strong documentation and testing strategies.

**Q: How do you keep up with Angular updates?**

A: I follow Angular Blog, GitHub releases, participate in community forums, and test new features in sandbox apps.

# Advanced Angular Interview Topics with Questions & Answers

**Q: Explain Angular Architecture.**

A: Angular applications are built using modules, components, services, and directives. Modules group related features. Components control views via templates and logic. Services contain business logic and data interaction, injected via dependency injection. Routing handles navigation.

**Q: What is the difference between Reactive and Template-Driven Forms?**

A: Reactive Forms are model-driven, offer better scalability, and use `FormGroup`, `FormControl`. Template-driven forms use directives like `ngModel`, suitable for simpler forms. Reactive Forms provide more control and validation flexibility.

**Q: What are switchMap, mergeMap, concatMap, forkJoin, and combineLatest in RxJS?**

A: `switchMap` cancels previous observables. `mergeMap` runs all inner observables concurrently. `concatMap` queues them. `forkJoin` waits for all to complete and emits final values. `combineLatest` emits the latest value from each observable whenever one emits.

**Q: What is NgRx and how does it help in Angular?**

A: NgRx is a reactive state management library based on Redux. It uses a unidirectional data flow, actions, reducers, selectors, and effects to manage and interact with global application state.

**Q: Explain Dependency Injection in Angular.**

A: Dependency Injection (DI) is a design pattern where a class receives its dependencies from external sources rather than creating them. Angular uses DI to provide services and other dependencies to components and services.

**Q: How is lazy loading implemented in Angular routing?**

A: Lazy loading is configured in Angular routing by using `loadChildren` with dynamic imports. This defers module loading until the route is activated, improving performance.

**Q: What are the main lifecycle hooks in Angular?**

A: `ngOnInit` (initialization), `ngOnChanges` (input changes), `ngOnDestroy` (cleanup), `ngAfterViewInit`, etc. help manage component behavior during its lifecycle.

**Q: What are Angular Signals (Angular 16+)?**

A: Signals are reactive primitives introduced in Angular 16 that notify the framework of state changes automatically, enabling fine-grained reactivity and optimized change detection.

**Q: How do you communicate between components in Angular?**

A: Use @Input/@Output for parent-child. Services with Subject or BehaviorSubject for unrelated components. EventEmitters can also be used.

**Q: What is ViewEncapsulation in Angular?**

A: ViewEncapsulation determines how styles are scoped. `Emulated` (default) scopes styles to the component, `None` applies globally, `ShadowDom` uses browser’s shadow DOM.

**Q: How do you optimize Angular apps for performance?**

A: Use `OnPush` change detection, lazy loading, trackBy in ngFor, avoiding memory leaks, preloading strategies, and efficient RxJS operator usage.

**Q: How do you use Angular Material, PrimeNG, or DevExtreme?**

A: These are UI component libraries. Import the needed modules and use components like MatTable, PrimeNG DataTable, or DevExtreme Grids. Customize via properties and events.

**Q: How do you create custom pipes and directives?**

A: Custom pipes implement `PipeTransform` and are decorated with `@Pipe`. Directives extend DOM behavior and are decorated with `@Directive`. Both are reusable and declarative tools.