Angular Interview Questions and Answers

1 The Fundamentals & Core Concepts (The "What is...")

Q: What is Angular? A: Angular is a platform and framework for building client-side single-page applications (SPAs) using HTML, CSS, and TypeScript/JavaScript. It implements core and optional functionality as a set of TypeScript libraries.

Q: What are the key features of Angular? A: Components, Templates, Directives, Dependency Injection, Services, Routing, Forms (Reactive & Template-driven), HTTP Client, Pipes, and ahead-of-time (AOT) compilation.

Q: What is TypeScript and why does Angular use it? A: TypeScript is a statically typed superset of JavaScript that compiles to plain JavaScript. Angular uses it for better tooling, early error detection, superior autocompletion, and improved code maintainability and scalability.

Q: What is a Component? A: A component is a building block of an Angular application, controlling a patch of screen called a view. It consists of a TypeScript class (@Component), an HTML template, and CSS styles.

Q: What is a Module (NgModule)? A: A module is a mechanism to group related components, directives, pipes, and services. It helps in organizing the application into cohesive blocks of functionality. The root module is AppModule.

Q: What is a Template? A: A template is a form of HTML that tells Angular how to render the component. It uses special syntax like interpolation { { }}, property binding [ ], event binding ( ), and directives.

Q: What is Data Binding? Explain the types. A: Data binding is the synchronization between the component and the view.
- Interpolation: {{value}} (Component to DOM).
- Property Binding: [property]="value" (Component to DOM).
- Event Binding: (event)="handler()" (DOM to Component).
- Two-Way Binding: [(ngModel)]="property" (Combination of property and event binding).

Q: What are Directives? A: Directives are classes that add additional behavior to elements in your Angular applications. The three kinds are Components, Structural, and Attribute directives.

Q: Difference between AngularJS (Angular 1.x) and Angular (2+)? A: They are completely different frameworks. Angular uses a component-based architecture, TypeScript, a different change detection mechanism, mobile support, and doesn't use the concept of $scope.

2 Components & Templates (The "How does it work...")

Q: What is the Component Lifecycle? Name the hooks. A: A component instance has a lifecycle that starts when Angular instantiates it and ends when it destroys it. Hooks (like ngOnInit, ngOnChanges) allow you to tap into key moments.
- ngOnChanges: Called when input properties change.
- ngOnInit: Called once after the first ngOnChanges.
- ngDoCheck: Custom change detection.

- ngAfterContentInit: Called after content is projected.
- ngAfterContentChecked: Called after projected content is checked.
- ngAfterViewInit: Called after the component's views are initialized.
- ngAfterViewChecked: Called after the views are checked.
- ngOnDestroy: Called just before the component is destroyed.

Q: What is the difference between ngOnInit and the constructor? A: The constructor is a default method of the TypeScript class, used for dependency injection and initializing class members. ngOnInit is an Angular lifecycle hook where you should place initialization logic that requires the component's input properties to be set.

Q: What are @Input and @Output properties? A: @Input() allows a parent component to pass data to a child component. @Output() allows a child component to emit events to a parent component, typically using EventEmitter.

Q: What is a Template Reference Variable? A: A variable that references a DOM element or directive within a template. It's defined using # or ref-, e.g., <input #phoneInput>. You can then use phoneInput in the template.

Q: What are Structural Directives? A: They shape or reshape the DOM's structure by adding, removing, or manipulating elements. Examples: *ngIf, *ngFor, *ngSwitch.

Q: What are Attribute Directives? A: They change the appearance or behavior of an element, component, or another directive. Examples: NgClass, NgStyle.

Q: What is the difference between *ngIf and [hidden]? A: *ngIf adds or removes the element from the DOM. [hidden] uses CSS (display: none) to hide/show the element. Use *ngIf for expensive components that don't need to be initialized when hidden. Use [hidden] for simple toggling where the element's state should be preserved.

Q: How does *ngFor work? What is the trackBy function? A: *ngFor is a structural directive for rendering a list. The trackBy function improves performance by telling Angular how to track items in an iterable. Without it, Angular re-renders the entire list on any change. With trackBy, it only re-renders changed items.

Q: What are Pipes? Name some built-in ones. A: Pipes are simple functions you can use in template expressions to accept an input value and return a transformed value. Built-in: DatePipe, UpperCasePipe, LowerCasePipe, CurrencyPipe, DecimalPipe, PercentPipe.

Q: How do you create a custom pipe? A: Create a class decorated with @Pipe, implement the PipeTransform interface's transform method, and declare it in an NgModule.

Q: What is pure vs. impure pipe? A: A pure pipe is only executed when Angular detects a pure change to the input value (primitive change or object reference change). An impure pipe is executed during every component change detection cycle, which can be expensive. You specify purity with the pure property in the @Pipe decorator.

3 Services & Dependency Injection (The "Sharing Logic")

Q: What is a Service? A: A service is a broad category encompassing any value, function, or feature that an application needs. It's typically a class with a specific, well-defined purpose (e.g., logging, data fetching).

Q: What is Dependency Injection (DI)? **A:** DI is a design pattern in which a class requests dependencies from external sources rather than creating them itself. Angular's DI framework provides dependencies to a class upon instantiation.

Q: What is an Injector? **A:** The Injector is the main DI container that maintains a list of providers and is responsible for resolving dependencies and creating instances.

Q: What is a Provider? **A:** A provider tells the injector how to create or obtain a dependency. It's typically a mapping between a token (the key) and a recipe (the value, like a class, value, or factory function) for creating the object.

Q: What is the @Injectable decorator? **A:** It marks a class as available to be provided and injected as a dependency. The providedIn metadata allows you to specify where the service should be provided (e.g., 'root').

Q: What are the different ways to provide a service? **A:** - providedIn: 'root' (Singleton, tree-shakable).
- In a specific NgModule's providers array (available to all components in that module).
- In a specific component's providers array (a new instance for each instance of that component).

Q: What is a Singleton Service and how do you create one? **A:** A service for which only one instance exists in the entire application. The preferred way is to use @Injectable({ providedIn: 'root' }).

Q: What is the Hierarchical Dependency Injection system? **A:** Angular has a hierarchical injector system. Injectors are nested, and when a component requests a dependency, Angular starts at the component's injector and goes up the tree until it finds the provider. This allows for service scoping.

Q: What is the @Inject decorator used for? **A:** It's used to manually specify the token for a dependency, which is MOJO is a JavaScript library for reactive programming using Observables, making it easier to compose asynchronous or callback-based code.

Q: What is an Observable? **A:** An Observable represents a stream of data or events that can arrive asynchronously over time. You can subscribe to it to receive notifications.

Q: What is an Observer? **A:** An Observer is a collection of callbacks that knows how to listen to values delivered by an Observable (next, error, complete).

Q: What is a Subscription? **A:** A Subscription represents the execution of an Observable. It has an unsubscribe() method to cancel the execution and free resources.

Q: Why is it important to unsubscribe from Observables? **A:** To prevent memory leaks. If you don't unsubscribe, the Observable's execution context remains in memory even after the component is destroyed.

Q: What are the different ways to unsubscribe? **A:** - Manually calling subscription.unsubscribe() in ngOnDestroy.
- Using the async pipe in the template (highly recommended).
- Using the takeUntil operator with a subject in ngOnDestroy.
- Using the take(1) operator for single-emission observables.

Q: What is the async pipe? **A:** The async pipe subscribes to an Observable or Promise and returns the latest value it has emitted. It automatically handles subscription and unsubscription.

Q: What is the difference between a Promise and an Observable? **A:** - Observable: Lazy (executed only on subscription), cancellable, can emit multiple values, supports operators (map, filter, etc.).
- Promise: Eager (executed immediately), not cancellable, emits a single value, no operators.

Q: What are Subjects? **A:** A Subject is a special type of Observable that allows values to be multicasted to many Observers. It is both an Observable and an Observer.

Q: Difference between Subject, BehaviorSubject, and ReplaySubject? **A:** - Subject: No initial value, only emits values after subscription.
- BehaviorSubject: Requires an initial value and emits the current value to new subscribers.
- ReplaySubject: Emits a specified number of previous values to new subscribers.

Q: Name some common RxJS operators. **A:** map, filter, tap, switchMap, mergeMap, concatMap, catchError, debounceTime, distinctUntilChanged, take, takeUntil.

Q: When would you use switchMap vs mergeMap? **A:** Use switchMap when you want to cancel previous inner observables (e.g., HTTP requests on new user input, like search). Use mergeMap when you want to concurrently handle all inner observables.

Q: How do you handle errors in Observables? **A:** Using the catchError operator inside a pipe. You can return a new observable or throw the error.

Q: What is the difference between of and from? **A:** of converts arguments to an observable sequence. from converts an array, promise, or iterable into an observable.

4 Change Detection & Performance (The "Under the Hood")

Q: What is Change Detection? **A:** Change Detection is the process through which Angular checks if the state of your application has changed and updates the DOM accordingly.

Q: How does Angular's Change Detection work? **A:** By default, Angular uses a mechanism where every component is checked from top to bottom when any asynchronous event occurs (e.g., click, timer, HTTP request) to see if any bindings have changed.

Q: What is Zone.js and what is its role? **A:** Zone.js is a library that monkey-patches asynchronous browser APIs (setTimeout, addEventListener, etc.). It notifies Angular when an asynchronous operation is completed, so Angular knows when to run change detection.

Q: What is OnPush Change Detection strategy? **A:** A component can be configured with changeDetection: ChangeDetectionStrategy.OnPush. This tells Angular to only run change detection for this component when:
- Its @Input references change.
- An event originates from the component or its children.
- An observable linked to the template via the async pipe emits a new value.

Q: When and why would you use the OnPush strategy? **A:** To improve performance, especially in large applications. It prevents Angular from performing unnecessary checks on a component subtree that hasn't changed.

Q: How do you make an OnPush component update when data changes internally? **A:** - Use immutable data patterns for @Inputs (create new references).

- Use observables with the async pipe in the template.
- Manually trigger change detection using ChangeDetectorRef.markForCheck().
- Manually detach/reattach the change detector using ChangeDetectorRef.

Q: What is ChangeDetectorRef? **A:** A base class that provides change detection functionality. Key methods: markForCheck(), detach(), reattach(), detectChanges().

Q: What is the difference between markForCheck() and detectChanges()? **A:** markForCheck() marks the component and its ancestors to be checked in the next change detection cycle. detectChanges() immediately runs change detection on this component and its children.

Q: What are Pure Pipes in the context of performance? **A:** Pure pipes are a performance feature. Because they are only re-executed on pure changes, they prevent unnecessary calculations during change detection cycles.

5 Routing & Navigation (The "Single Page App")

Q: How do you set up routing in Angular? **A:** Define routes in a Routes array, import RouterModule.forRoot(routes) in the root module, and use <router-outlet> as a placeholder and routerLink for navigation.

Q: What is RouterOutlet? **A:** A directive from the router library that acts as a placeholder where the router should display the components for that route.

Q: What is routerLink? **A:** A directive for binding a clickable HTML element to a route. It's the preferred way to navigate in templates.

Q: How do you navigate programmatically? **A:** By injecting the Router service and calling its navigate() or navigateByUrl() method.

Q: How do you pass parameters in a route? **A:** - Route Parameters: /route/:id -> this.route.snapshot.paramM
- Query Parameters: /route?name=value -> this.route.snapshot.queryParamMap.get('name')

Q: What is the difference between paramMap and queryParamMap? **A:** paramMap is for mandatory values part of the route path (/user/:id). queryParamMap is for optional parameters after the ? (/user?id=123).

Q: What is a Route Guard? **A:** An interface that allows you to control navigation. It can allow/prevent navigation, pre-fetch data, etc.

Q: Name the different types of Route Guards. **A:** CanActivate, CanActivateChild, CanDeactivate, Resolve, CanLoad.

Q: What is the purpose of CanActivate and CanLoad? **A:** CanActivate checks if a user can access a route. CanLoad checks if a user can load a lazy-loaded feature module (better for performance).

Q: What is Lazy Loading? **A:** A technique where feature modules are loaded on demand, not at the initial application load. This significantly improves startup time.

Q: How do you implement Lazy Loading? **A:** In the route configuration, use the loadChildren property pointing to the module's file path: loadChildren: () => import('./admin/admin.module').then(m => m.AdminModule)

Q: What are Child Routes (Nested Routes)? **A:** Routes defined within a component that has its own <router-outlet>. They allow for a hierarchical navigation structure.

Q: How do you handle "Page Not Found" (404) scenarios? **A:** Define a wildcard route { path: '**', component: PageNotFoundComponent } as the last route in your configuration.

Q: What is the difference between Router and ActivatedRoute? **A:** Router is for navigating and configuring the router. ActivatedRoute contains information about a route associated with a component currently loaded in an outlet.

6 Forms (The "User Input")

Q: What are the two main approaches to forms? **A:** Template-driven and Reactive (model-driven) forms.

Q: What is the key difference between Template-driven and Reactive forms? **A:** - Template-driven: The form model is implicit, created by directives in the template. Logic resides mostly in the template. Asynchronous.
- Reactive: The form model is explicit, created programmatically in the component class. Logic resides in the component. Synchronous and more testable.

Q: When would you choose one over the other? **A:** Use Template-driven for simple forms with basic validation. Use Reactive for complex forms, dynamic form controls, custom validations, and when you need more control and testability.

Q: What are the key building blocks of Reactive Forms? **A:** FormControl, FormGroup, FormArray, FormBuilder (a service for creating them).

Q: What is FormBuilder? **A:** A helper service that provides syntactic sugar for creating instances of FormControl, FormGroup, and FormArray. It reduces boilerplate.

Q: What are the common Validators? **A:** Validators.required, Validators.minLength, Validators.maxLength, Validators.pattern, Validators.email.

Q: How do you create a Custom Validator? **A:** A function that takes an AbstractControl and returns a validation error object { [key: string]: any } if invalid, or null if valid.

Q: What is the difference between dirty, touched, and pristine? **A:** - dirty: Control's value has changed.
- pristine: Control's value has not changed.
- touched: Control has been blurred (lost focus).
- untouched: Control has not been blurred.

Q: How do you set a value programmatically in Reactive Forms? **A:** Using setValue() (sets value for the entire form group) or patchValue() (sets value for a subset of the form group).

Q: What is the purpose of FormArray? **A:** FormArray manages a collection of form controls dynamically. It's useful for when you have a variable number of controls (e.g., a list of phone numbers).

7 HTTP Client & Communication (The "Talking to Servers")

Q: What is the HttpClient? A: A built-in service for making HTTP requests. It provides a simplified API and built-in support for features like request/response interception, typed responses, and error handling.

Q: How do you make a GET request? A: this.http.get(url). It returns an Observable of the response.

Q: How do you handle errors in HTTP requests? A: Use the catchError operator from RxJS inside the pipe().

Q: What are HTTP Interceptors? A: Interceptors are middleware that can transform outgoing requests or incoming responses. They are useful for adding auth headers, logging, error handling, etc.

Q: How do you create an HTTP Interceptor? A: Create a class that implements HttpInterceptor and defines the intercept method. Then, provide it in the root module using the HTTP$_{I}NTERCEPTORStoken.$

Q: Can you have multiple interceptors? A: Yes, they form a chain. The order of provision defines the order of execution for requests and the reverse order for responses.

Q: What is the purpose of the async pipe with HTTP requests? A: It automatically subscribes to the Observable returned by HttpClient, handles the response, and unsubscribes when the component is destroyed, preventing memory leaks.

Q: How do you pass headers and parameters in an HTTP request? A: Use the HttpParams and HttpHeaders classes with the options object: this.http.get(url, { headers: myHeaders, params: myParams }).

8 Advanced Concepts (The "3-Years-Experience" Stuff)

Q: What is AOT (Ahead-of-Time) Compilation? A: The Angular compiler runs at build time, converting your Angular HTML and TypeScript code into efficient JavaScript code. This is the default in production.

Q: Benefits of AOT over JIT (Just-in-Time)? A: Faster rendering, smaller application size, fewer asynchronous requests, earlier template error detection, better security.

Q: What is Ivy? A: Ivy is Angular's next-generation compilation and rendering pipeline. It's the default engine since Angular 9.

Q: Benefits of Ivy? A: Smaller bundle sizes, faster compilation, better debugging (e.g., ng.probe), improved build performance, and enabling new features like incremental compilation.

Q: What are ng-container and ng-template? A: - ng-template: Defines a template that is not rendered by default. It's used with structural directives.
- ng-container: A logical container that doesn't add an element to the DOM. It's used to group elements without a wrapper.

Q: What is Content Projection (ng-content)? A: A pattern where you insert (project) content from a parent component into a designated area in the child component's template using the

<ng-content> tag.

Q: What is View Encapsulation? **A:** It defines how the component's styles are scoped. Modes: Emulated (default, scoped via attributes), None (global styles), ShadowDom (uses native Shadow DOM).

Q: What is the difference between ng-content and ng-template? **A:** ng-content is for static content projection. ng-template with ngTemplateOutlet is for dynamic content projection, where the child component has more control over the content.

Q: What is an ElementRef? **A:** A wrapper around a native DOM element. It gives you direct access to the DOM element, but using it is discouraged due to security (XSS) and platform separation concerns.

Q: What is Renderer2? **A:** A service that provides an API for manipulating the DOM safely. It's the preferred way over direct ElementRef manipulation because it abstracts the underlying platform.

Q: What is the APP$_{I}NITIALIZERtoken?$**A:**$Itallowsyoutoprovideafunctionthatreturnsapromise.An$

Q: What are Dynamic Components? A: Components that are loaded at runtime, not defined in a template. This requires a ViewContainerRef and a ComponentFactoryResolver (or ViewContainerRef.createComponent in Ivy).

Q: How do you communicate between two unrelated components? A: - Using a Service with a Subject: A shared service with a BehaviorSubject or Subject can act as a message bus.
- Using State Management (NgRx, Akita): For complex application-wide state.
- Using @Output and events bubbling up and then back down (not recommended for distant components).

Q: What is the purpose of the exportAs property in a directive? A: It allows you to expose the directive instance to the template via a template reference variable. E.g., #myDir="ngModel" gives you access to the NgModel directive.

Q: What is Tree-shakable? A: A feature that allows unused code (services, etc.) to be "shaken" out of the final production bundle, reducing its size. Services with providedIn: 'root' are tree-shakable.

9 State Management, Testing & Architecture

Q: What is State Management? When do you need it? A: A pattern to manage the state of your application in a predictable, centralized way. You need it when your application grows and component communication via @Input/@Output becomes complex and hard to debug.

Q: What is NgRx? A: A reactive state management library for Angular inspired by Redux, using RxJS. Core concepts: Actions, Reducers, Selectors, Store, Effects.

Q: Explain the NgRx Data Flow. A: Component dispatches an Action -> Effect (optional side effect) -> Reducer (pure function updates state) -> Store (new state) -> Selector (query state) -> Component updates via async pipe.

Q: What are Unit Tests? A: Tests that isolate a single part of your code (a class, function, or component) and verify its correctness.

Q: What testing frameworks are used in Angular? A: Jasmine (behavior-driven test framework) and Karma (test runner). For E2E, Protractor (currently being deprecated) or Cypress/Playwright.

Q: What is TestBed? A: The primary API for writing unit tests in Angular. It creates a dynamically-constructed Angular testing module where you can configure providers, imports, and declarations for your test.

Q: What is a ComponentFixture? A: A wrapper around a component instance and its corresponding element, providing access to the component, the DOM element, and debugging utilities.

Q: What is fakeAsync and tick? A: Utilities for testing asynchronous code synchronously. fakeAsync wraps the test, and tick() simulates the passage of time, flushing pending asynchronous tasks.

Q: What is debugElement? A: A wrapper around the native element that provides a safe way to query the DOM and trigger event handlers in tests.

Q: What is the difference between compileComponents and configureTestingModule? A: configureTestingModule sets up the testing module. compileComponents is used to asynchronously compile components with external templates (if you are not using the Angular CLI, which handles this automatically).

10 Practical Scenarios & Problem-Solving (The "How would you...")

Q: How would you optimize a slow Angular application? A: Answer not provided in the input.

Q: How do you implement a debounce search functionality? A: Answer not provided in the input.

Q: How do you handle authentication and authorization in Angular? A: Answer not provided in the input.

Q: How do you implement internationalization (i18n)? A: Answer not provided in the input.

Q: How do you make an Angular application SEO-friendly? (SSR) A: Answer not provided in the input.

Q: What is Server-Side Rendering (SSR) and Angular Universal? A: Answer not provided in the input.

Q: How do you update your application to a new Angular version? A: Answer not provided in the input.

Q: How do you handle memory leaks in a large application? A: Answer not provided in the input.

Q: Describe your process for debugging a performance issue. A: Answer not provided in the input.

Q: How do you structure a large enterprise Angular application? A: Answer not provided in the input.

Q: How do you ensure code quality and consistency in a team? A: Answer not provided in the input.

Q: How do you implement a feature that requires real-time updates? A: Answer not provided in the input.

Q: How would you share data between two sibling components? A: Answer not provided in the input.

Q: How do you handle errors globally in an Angular app? A: Answer not provided in the input.

Q: How do you implement a "select all" checkbox in a list? A: Answer not provided in the input.

Q: How do you create a multi-step wizard form? A: Answer not provided in the input.

Q: How do you cache HTTP responses? A: Answer not provided in the input.

Q: How do you implement a custom structural directive? A: Answer not provided in the input.

Q: How do you use Web Workers with Angular? A: Answer not provided in the input.

Q: How do you integrate a third-party library (e.g., D3.js) into Angular? A: Answer not provided in the input.

11 Behavioral & Experience-Based

Q: What has been your most challenging project with Angular? A: Answer not provided in the input.

Q: Describe a time you had to fix a critical bug. What was your process? A: Answer not provided in the input.

Q: How do you stay updated with the latest Angular developments? A: Answer not provided in the input.

Q: What do you like most and least about Angular? A: Answer not provided in the input.

Q: Describe your experience with state management libraries (NgRx, Akita, etc.). A: Answer not provided in the input.

Q: Have you contributed to any open-source projects or created your own libraries? A: Answer not provided in the input.

Q: How do you approach code reviews in an Angular project? A: Answer not provided in the input.

Q: Describe your experience with CI/CD for Angular apps. A: Answer not provided in the input.

Q: How do you mentor junior developers on your team regarding Angular? A: Answer not provided in the input.

Q: What is your experience with mobile development (Ionic, NativeScript)? A: Answer not provided in the input.

12 Tricky & Niche Questions

Q: What is the difference between router.navigate() and router.navigateByUrl()? A: Answer not provided in the input.

Q: What is a no-errors-schema in testing? A: Answer not provided in the input.

Q: What is the defineComponent function? (Ivy-specific) A: Answer not provided in the input.

Q: How does Angular handle two-way binding without ngModel? (Custom @Input & @Output with same name) A: Answer not provided in the input.

Q: What is the keyvalue pipe and when would you use it? A: Answer not provided in the input.

Q: What is the json pipe used for? A: Answer not provided in the input.

Q: How can you force a component to re-render? A: Answer not provided in the input.

Q: What is the purpose of the @Attribute decorator? A: Answer not provided in the input.

Q: What is NgProbeToken? A: Answer not provided in the input.

Q: What is the difference between platform-browser and platform-browser-dynamic? A: Answer not provided in the input.