Angular Interview Questions & Answers

For 2 Years Experience

Prepared by ChatGPT

July 1, 2025

1. What is Angular?

Angular is a TypeScript-based open-source front-end web application framework developed and maintained by Google. It is designed for building single-page applications (SPA), which load a single HTML page and dynamically update the content as users interact with the app, avoiding full page reloads.

Angular follows a component-based architecture where the UI is broken down into reusable components. It offers powerful features like two-way data binding, dependency injection, routing, and modular development. Angular also supports reactive programming through RxJS, making it ideal for developing scalable, maintainable, and performant web applications.

2. What are the key building blocks of Angular?

The core building blocks of an Angular application include:

- **Components:** Control a part of the screen (UI). Each component consists of a TypeScript class (logic), HTML template (view), and CSS styles.
- **Modules:** Angular apps are modular. The root module (usually AppModule) groups components, services, and other modules together.
- **Templates:** Define the HTML layout and display dynamic data using Angular syntax such as interpolation ({{}}) and directives.
- **Metadata:** Decorators like `@Component` or `@NgModule` provide Angular with information about classes and how to process them.
- **Services:** Classes that provide business logic or data, often injected into components via Dependency Injection.
- **Directives:** Instructions in the DOM that change the structure or appearance of elements (`*ngIf`, `*ngFor`).
- **Routing:** Manages navigation within the app without full page reloads.
- **Pipes:** Transform displayed data, such as formatting dates or currencies.

3. What is data binding? Explain its types.

Data binding connects data between the component class and the HTML template, enabling the view and component to stay in sync.

Types:

- **One-way Binding:**
  - **Interpolation ({{ }}):** Displays component data in the template.
  - **Property Binding (`[property]="expression"`):** Binds an element property to component data.
  - **Event Binding (`(event)="handler()"`):** Listens to user events and calls component methods.
- **Two-way Binding (`[(ngModel)]`):** Combines property and event binding to synchronize data both ways — from component to view and view to component, commonly used with forms.

4. What is the difference between constructor and ngOnInit?

Constructor:

- Runs when the component class is instantiated.
- Used to initialize class members and inject dependencies.
- Should avoid heavy or Angular-dependent logic here.

ngOnInit:

- Angular lifecycle hook called after the constructor and after Angular has set input properties.
- Ideal place for component initialization tasks such as fetching data or setting up logic that depends on inputs.

5. How does Angular Dependency Injection work?

Dependency Injection (DI) is a design pattern where Angular automatically provides required services to a component or other services.

- Services are marked with `@Injectable()` and registered as providers.
- When a component declares a service dependency in its constructor, Angular's injector creates or reuses a singleton instance of that service and injects it.
- This promotes modular, reusable, and testable code since services are centrally managed.

6. What are Angular Directives?

Directives are classes that manipulate the DOM or add behavior to elements.

- **Structural Directives:** Change DOM layout by adding/removing elements. Examples:
  - `*ngIf` — conditionally includes an element
  - `*ngFor` — repeats an element for each item in a list

- **Attribute Directives:** Change the appearance or behavior of an element without changing its structure. Examples:
  - `[ngClass]` — dynamically add/remove CSS classes
  - `[ngStyle]` — dynamically apply styles

7. What is Angular Routing?

Routing allows navigation between different views or components in a SPA without refreshing the entire page.

- Routes are defined with a path and a component.
- Angular Router listens to URL changes and renders the corresponding component.
- Supports features like lazy loading, route guards, and parameterized routes.

8. What are Angular Pipes?

Pipes transform displayed data in templates.

- Built-in pipes include `date`, `uppercase`, `currency`, `json`.
- Example usage:

```
{{ birthday | date:'shortDate' }}
```

- You can also create custom pipes for specific transformations.

9. What is the difference between Template-driven and Reactive Forms?

Template-driven Forms:

- Simpler and mainly defined in the template using directives like `ngModel`.
- Suitable for simple forms.

Reactive Forms:

- Form model and validation logic are defined in the component class using `FormGroup` and `FormControl`.
- Better for complex forms requiring dynamic validation or reactive patterns.

10. What is Angular CLI?

Angular CLI (Command Line Interface) is a tool that helps developers create, build, serve, test, and maintain Angular projects with simple commands like:

- `ng new` to create a new project
- `ng serve` to run a development server
- `ng generate` to create components, services, etc.

11. What is an Observable? How is it used in Angular?

An Observable is a stream of data that can emit multiple values over time, part of the RxJS library Angular uses. Observables support operations like map, filter, and reduce, and they are lazy (don't run until subscribed).

Angular uses Observables for handling asynchronous data, such as HTTP requests, event handling, and reactive forms. They allow better control over async operations and cancellation.

Example:

```
this.http.get('/api/users').subscribe(data => {
  console.log(data);
});
```

12. Explain Angular lifecycle hooks. Name some important ones.

Lifecycle hooks are methods Angular calls at specific moments in a component's life. Important hooks:

- `ngOnChanges()` – called when input properties change
- `ngOnInit()` – called once after first `ngOnChanges`, good for initialization
- `ngDoCheck()` – for custom change detection
- `ngAfterViewInit()` – after component's views are initialized
- `ngOnDestroy()` – cleanup before component destroyed (unsubscribe, etc.)

13. What is lazy loading in Angular?

Lazy loading is a technique to load feature modules only when needed, instead of loading everything at startup. This improves initial load time and performance. It is implemented using the router's `loadChildren` property.

14. What are decorators in Angular?

Decorators add metadata to classes, telling Angular how to process them. Examples include:

- `@Component` – marks a class as a component
- `@NgModule` – marks a class as an Angular module
- `@Injectable` – marks a class as a service that can be injected

15. How do you share data between Angular components?

Common methods:

- Parent to child: use `@Input()` decorator
- Child to parent: use `@Output()` with `EventEmitter`
- Between unrelated components: use a shared service with RxJS Subjects or BehaviorSubjects
- Using state management libraries like NgRx for larger apps

16. What is Angular Universal?

Angular Universal is Angular's server-side rendering (SSR) solution. It renders Angular apps on the server, which improves SEO, performance, and faster initial page loads, especially useful for content-heavy or public-facing sites.

17. What is zone.js in Angular?

`zone.js` is a library Angular uses to detect asynchronous activities like clicks, HTTP calls, or timers. It patches async APIs and triggers Angular's change detection automatically whenever an async operation completes.

18. How do you optimize Angular application performance?

Optimization techniques include:

- Implement lazy loading for feature modules
- Use the `OnPush` change detection strategy where possible
- Use `trackBy` with `*ngFor` to reduce DOM manipulation
- Avoid complex computations in templates
- Minimize watchers and subscriptions
- Use Ahead-of-Time (AOT) compilation
- Bundle and minify code for production

19. What is Change Detection in Angular?

Change Detection is Angular's mechanism to check for changes in component data and update the DOM accordingly. It runs after events, timers, or async operations to keep the UI in sync with the model. Angular provides different strategies like Default and OnPush for optimizing change detection.

20. What is the difference between Observable and Promise?

Observable	Promise
Can emit multiple values over time	Emits a single value or error once
Lazy — does not execute until subscribed	Eager — executes immediately
Can be cancelled (unsubscribe)	Cannot be cancelled
Supports operators like map, filter	Does not support operators
Used widely in Angular for async data	Common in native JS and async calls

Bonus Questions

21. What are Angular Guards and how are they used?

Angular Guards control access to routes and prevent unauthorized navigation. They are implemented as services that implement specific guard interfaces:

- `CanActivate`: Checks if a route can be activated (access allowed).
- `CanDeactivate`: Checks if navigation away from a route is allowed (e.g., warn unsaved changes).
- `CanActivateChild`: Checks if child routes can be activated.
- `CanLoad`: Prevents lazy-loaded modules from loading unless condition is met.

Guards return `true` or `false` (or an Observable/Promise of those) to allow or deny navigation.

22. Explain Ahead-of-Time (AOT) vs Just-in-Time (JIT) compilation.

JIT Compilation:

- Compilation happens in the browser at runtime.
- Slower startup, larger bundle size.
- Useful for development and debugging.

AOT Compilation:

- Compilation happens at build time, before deployment.
- Smaller bundle size and faster startup time.
- Produces highly optimized JavaScript code.

Angular CLI uses AOT by default for production builds.

23. What are the advantages of using Angular?

- Component-based architecture makes development modular and reusable.
- TypeScript support provides type safety and tooling.
- Powerful data binding (one-way and two-way).
- Dependency Injection makes code modular and testable.
- Built-in routing for SPA navigation.
- RxJS integration for reactive programming.
- CLI tool for easy scaffolding and maintenance.
- Cross-platform development for web, mobile, and desktop.
- Strong community support and Google backing.

24. How do you handle errors in Angular HTTP requests?

Use RxJS `catchError` operator inside HTTP request Observables to catch errors gracefully.

Example:

```
this.http.get('/api/data').pipe(
  catchError(error => {
    console.error('Error occurred:', error);
    return throwError(() => new Error('Something bad happened'));
  })
).subscribe(...);
```

Also, Angular's `HttpInterceptor` can be used to globally handle HTTP errors for all requests.

25. What is the difference between ViewChild and ContentChild?

ViewChild	**ContentChild**
Accesses a child component, directive, or DOM element within the component's own template.	Accesses projected content inserted into the component via `<ng-content>`.
Used to interact with elements declared inside the component template.	Used to interact with external content passed to the component.
Runs after `ngAfterViewInit()` lifecycle hook.	Runs after `ngAfterContentInit()` lifecycle hook.