Scenario angular Developer Interview Questions

1. Architecture & Application Design

Q: How would you structure a large-scale Angular application to maintain modularity and scalability?

Scenario: You are tasked with building an enterprise-level application that needs to scale as more features are added. How would you ensure that the app remains maintainable and scalable over time?

Q: Explain the importance of Angular's dependency injection system. How would you use it in a complex project?

Scenario: In a large project, you need to manage services that require multiple dependencies. How do you handle hierarchical injectors and ensure the right instance of a service is injected?

2. Performance Optimization

Q: What techniques would you use to optimize the performance of an Angular application?

Scenario: Your Angular app has performance issues, particularly in rendering large lists of items and handling heavy API requests. What strategies do you employ to improve it?

Q: How would you handle memory leaks in an Angular application?

Scenario: After long use, the application becomes slow due to increasing memory consumption. How would you identify and fix memory leaks in components or services?

3. Angular Forms

Q: How would you approach managing complex forms with nested and dynamic form controls in Angular?

Scenario: You're building a form-heavy application where the form fields depend on previous selections and need to be generated dynamically. How would you handle validation, control generation, and submission?

Q: When should you use reactive forms vs template-driven forms in Angular?

Scenario: Given a project where you need both high flexibility and maintainability, which approach would you recommend for managing forms and why?

4. State Management

Q: What state management solutions would you recommend for an Angular application, and why?

Scenario: You are building a feature-rich Angular application with complex user interactions. How would you manage the application's global state, and what would your approach be in using services or libraries like NgRx?

Q: How do you handle the side effects in an NgRx application?

Scenario: You have a requirement to handle complex async actions like API calls in an NgRx-based Angular application. How would you use effects to manage side effects?

5. Routing and Navigation

Q: How would you implement lazy loading in an Angular application and what benefits does it provide?

Scenario: The application’s initial load time is too long, and only certain sections of the app are needed on demand. How would you optimize the initial load time?

Q: How would you handle authentication and route guarding in Angular?

Scenario: You need to ensure that only authenticated users can access certain routes while managing roles and permissions. How would you implement route guards and manage unauthorized access?

6. Testing & Debugging

Q: How do you test Angular components, services, and directives?

Scenario: You are leading a project with a strict requirement for high test coverage. How would you ensure that all components, services, and directives are properly tested using Jasmine and Karma?

Q: How do you debug Angular applications efficiently?

Scenario: There's an issue in a production environment where a component is not updating as expected. How would you debug and identify the root cause?

7. Version Control and Deployment

Q: How would you manage Angular version upgrades in a large codebase?

Scenario: You are responsible for upgrading an Angular 10 project to Angular 15. How would you plan, execute, and test this migration?

Q: What is the purpose of Ahead-of-Time (AOT) compilation, and when should you use it?

Scenario: Your application is experiencing performance degradation in production, specifically in rendering and execution speed. Would you enable AOT? Why or why not?

8. Advanced Topics

Q: Explain the concept of Angular zones and how Zone.js impacts Angular's change detection.

Scenario: Your app is performing unnecessary change detection checks in zones, affecting performance. How would you reduce the impact and optimize change detection?

Q: How would you implement server-side rendering (SSR) with Angular Universal?

Scenario: SEO is critical for the project you are leading, and you need to improve the initial load time for users on slow networks. How would you implement Angular Universal to achieve this?

1. Architecture & Application Design

Q: How would you structure a large-scale Angular application to maintain modularity and scalability?

Scenario: You are tasked with building an enterprise-level application that needs to scale as more features are added. How would you ensure that the app remains maintainable and scalable over time?

Q: Explain the importance of Angular's dependency injection system. How would you use it in a complex project?

Scenario: In a large project, you need to manage services that require multiple dependencies. How do you handle hierarchical injectors and ensure the right instance of a service is injected?

2. Performance Optimization

Q: What techniques would you use to optimize the performance of an Angular application?

Scenario: Your Angular app has performance issues, particularly in rendering large lists of items and handling heavy API requests. What strategies do you employ to improve it?

Q: How would you handle memory leaks in an Angular application?

Scenario: After long use, the application becomes slow due to increasing memory consumption. How would you identify and fix memory leaks in components or services?

3. Angular Forms

Q: How would you approach managing complex forms with nested and dynamic form controls in Angular?

Scenario: You're building a form-heavy application where the form fields depend on previous selections and need to be generated dynamically. How would you handle validation, control generation, and submission?

Q: When should you use reactive forms vs template-driven forms in Angular?

Scenario: Given a project where you need both high flexibility and maintainability, which approach would you recommend for managing forms and why?

4. State Management

Q: What state management solutions would you recommend for an Angular application, and why?

Scenario: You are building a feature-rich Angular application with complex user interactions. How would you manage the application's global state, and what would your approach be in using services or libraries like NgRx?

Q: How do you handle the side effects in an NgRx application?

Scenario: You have a requirement to handle complex async actions like API calls in an NgRx-based Angular application. How would you use effects to manage side effects?

5. Routing and Navigation

Q: How would you implement lazy loading in an Angular application and what benefits does it provide?

Scenario: The application’s initial load time is too long, and only certain sections of the app are needed on demand. How would you optimize the initial load time?

Q: How would you handle authentication and route guarding in Angular?

Scenario: You need to ensure that only authenticated users can access certain routes while managing roles and permissions. How would you implement route guards and manage unauthorized access?

6. Testing & Debugging

Q: How do you test Angular components, services, and directives?

Scenario: You are leading a project with a strict requirement for high test coverage. How would you ensure that all components, services, and directives are properly tested using Jasmine and Karma?

Q: How do you debug Angular applications efficiently?

Scenario: There's an issue in a production environment where a component is not updating as expected. How would you debug and identify the root cause?

7. Version Control and Deployment

Q: How would you manage Angular version upgrades in a large codebase?

Scenario: You are responsible for upgrading an Angular 10 project to Angular 15. How would you plan, execute, and test this migration?

: What is the purpose of Ahead-of-Time (AOT) compilation, and when should you use it?

Scenario: Your application is experiencing performance degradation in production, specifically in rendering and execution speed. Would you enable AOT? Why or why not?

8. Advanced Topics

Q: Explain the concept of Angular zones and how Zone.js impacts Angular's change detection.

Scenario: Your app is performing unnecessary change detection checks in zones, affecting performance. How would you reduce the impact and optimize change detection?

Q: How would you implement server-side rendering (SSR) with Angular Universal?

Scenario: SEO is critical for the project you are leading, and you need to improve the initial load time for users on slow networks. How would you implement Angular Universal to achieve this?

9. Component Communication

Q: How would you handle parent-child communication in Angular components?

Scenario: You have two components, one parent and one child. The parent needs to pass data to the child, and the child needs to send events to the parent. How do you achieve this?

Q: How do you manage communication between sibling components in Angular?

Scenario: Two sibling components need to share state or data. How would you facilitate this without tightly coupling the components?

10. Change Detection

Q: What are the different change detection strategies in Angular, and when should you use each?

Scenario: Your app is facing performance issues because Angular is running unnecessary change detection cycles. How would you use OnPush strategy to optimize performance?

Q: Explain the concept of zones in Angular and how they relate to change detection.

Scenario: Change detection is triggered more frequently than necessary in your application. How would you control or limit the use of zones to improve performance?

11. Pipes

Q: What is the difference between pure and impure pipes in Angular?

Scenario: You need a pipe that reacts to changes in arrays or objects. How would you decide whether to use a pure or impure pipe?

Q: How would you create a custom pipe in Angular?

Scenario: You need a custom transformation for formatting data in your application. Walk through the process of creating and applying a custom pipe.

12. Directives

Q: What is the difference between structural and attribute directives in Angular?

Scenario: You need to apply dynamic styling and hide/show elements based on conditions. How do you choose between structural and attribute directives?

Q: How would you create a custom directive in Angular?

Scenario: You need to create a directive that dynamically changes the behavior of an element. How would you design and implement it?

13. HTTP and Observables

Q: How do you handle HTTP requests in Angular using HttpClient?

Scenario: You need to interact with a RESTful API to fetch and submit data. Walk through how you would use the HttpClient service in Angular to manage requests.

Q: How do you handle error responses from HTTP requests?

Scenario: Your app needs to display proper messages when HTTP requests fail. How do you handle errors in the HttpClient service?

14. Observables and RxJS

Q: How would you implement debouncing or throttling using RxJS in Angular?

Scenario: Your search input triggers HTTP requests on every keypress. How would you reduce the number of API calls made by implementing debouncing?

Q: What are subjects in RxJS, and how do they differ from regular observables?

Scenario: You need a shared stream of data between components and services. How would you utilize subjects to achieve this?

15. Angular CLI & Build Tools

Q: How would you use Angular CLI to create a new Angular project with specific configurations?

Scenario: You are tasked with creating a new Angular project with routing and SCSS as the stylesheet format. How would you configure this using the CLI?

Q: How do you configure environment-specific settings in an Angular application?

Scenario: You need to handle different configurations for development, staging, and production environments. How would you set this up in Angular?

16. Security

Q: How do you secure Angular applications against common vulnerabilities like XSS or CSRF?

Scenario: You need to protect user data in an Angular application. What are some best practices for preventing vulnerabilities such as cross-site scripting (XSS) or cross-site request forgery (CSRF)?

Q: How would you implement role-based access control in Angular?

Scenario: Your app has different user roles, each with varying permissions to access features and routes. How would you enforce role-based access in Angular?

17. Progressive Web Applications (PWA)

Q: How would you convert an Angular app into a Progressive Web App (PWA)?

Scenario: You need to make your Angular application available offline and installable on user devices. Walk through the process of converting it into a PWA.

Q: How do you handle service workers in an Angular PWA?

Scenario: You need to ensure your Angular PWA caches assets and handles network requests efficiently. How would you configure and manage service workers?

18. Animations

Q: How would you implement animations in an Angular application?

Scenario: You need to add smooth animations when elements appear, disappear, or change state in the UI. How would you implement animations using Angular's built-in animation framework?

Q: How do you optimize Angular animations for performance?

Scenario: Your animations are causing UI lag. How would you ensure they are smooth and performant?

19. Module Federation

Q: What is Module Federation in Angular, and how would you implement it?

Scenario: You are tasked with creating a microfrontend architecture where different teams build independent parts of the application. How would you use Module Federation in Angular for this?

Q: How do you handle shared dependencies in a Module Federation setup?

Scenario: Multiple Angular applications share common libraries like Angular Material or RxJS. How would you manage shared dependencies across these apps?

20. Testing Advanced

Q: How would you test Angular services that have dependencies?

Scenario: You have a service that relies on several other services. How would you use dependency injection and mocks to test this service effectively?

Q: How do you ensure test coverage for asynchronous code in Angular (e.g., HTTP calls)?

Scenario: You have services that make HTTP calls. How would you write tests that properly handle and verify asynchronous responses?

21. Dependency Injection

Q: How do you manage multi-provider injection in Angular?

Scenario: You need to inject multiple instances of the same service with different configurations. How would you achieve this using Angular’s dependency injection system?

Q: How does Angular's dependency injection differ between component-level and root-level providers?

Scenario: You want to create a service that has different instances in different parts of the application. How do you control the scope of a service using providers?

22. Versioning and Migration

Q: How do you handle breaking changes when upgrading Angular versions?

Scenario: You are migrating a project from Angular 12 to 15, but there are breaking changes. How do you ensure the upgrade goes smoothly and manage backward compatibility?

Q: How would you manage dependency updates for a large Angular project?

Scenario: The project uses several third-party libraries that need updating. How do you manage the update process without introducing bugs or conflicts?

1. Architecture & Application Design

Q: How would you structure a large-scale Angular application to maintain modularity and scalability?

Scenario: You are tasked with building an enterprise-level application that needs to scale as more features are added. How would you ensure that the app remains maintainable and scalable over time?

Q: Explain the importance of Angular's dependency injection system. How would you use it in a complex project?

Scenario: In a large project, you need to manage services that require multiple dependencies. How do you handle hierarchical injectors and ensure the right instance of a service is injected?

2. Performance Optimization

Q: What techniques would you use to optimize the performance of an Angular application?

Scenario: Your Angular app has performance issues, particularly in rendering large lists of items and handling heavy API requests. What strategies do you employ to improve it?

Q: How would you handle memory leaks in an Angular application?

Scenario: After long use, the application becomes slow due to increasing memory consumption. How would you identify and fix memory leaks in components or services?

3. Angular Forms

Q: How would you approach managing complex forms with nested and dynamic form controls in Angular?

Scenario: You're building a form-heavy application where the form fields depend on previous selections and need to be generated dynamically. How would you handle validation, control generation, and submission?

Q: When should you use reactive forms vs template-driven forms in Angular?

Scenario: Given a project where you need both high flexibility and maintainability, which approach would you recommend for managing forms and why?

4. State Management

Q: What state management solutions would you recommend for an Angular application, and why?

Scenario: You are building a feature-rich Angular application with complex user interactions. How would you manage the application's global state, and what would your approach be in using services or libraries like NgRx?

Q: How do you handle the side effects in an NgRx application?

Scenario: You have a requirement to handle complex async actions like API calls in an NgRx-based Angular application. How would you use effects to manage side effects?

5. Routing and Navigation

Q: How would you implement lazy loading in an Angular application and what benefits does it provide?

Scenario: The application’s initial load time is too long, and only certain sections of the app are needed on demand. How would you optimize the initial load time?

Q: How would you handle authentication and route guarding in Angular?

Scenario: You need to ensure that only authenticated users can access certain routes while managing roles and permissions. How would you implement route guards and manage unauthorized access?

6. Testing & Debugging

Q: How do you test Angular components, services, and directives?

Scenario: You are leading a project with a strict requirement for high test coverage. How would you ensure that all components, services, and directives are properly tested using Jasmine and Karma?

Q: How do you debug Angular applications efficiently?

Scenario: There's an issue in a production environment where a component is not updating as expected. How would you debug and identify the root cause?

7. Version Control and Deployment

Q: How would you manage Angular version upgrades in a large codebase?

Scenario: You are responsible for upgrading an Angular 10 project to Angular 15. How would you plan, execute, and test this migration?

Q: What is the purpose of Ahead-of-Time (AOT) compilation, and when should you use it?

Scenario: Your application is experiencing performance degradation in production, specifically in rendering and execution speed. Would you enable AOT? Why or why not?

8. Advanced Topics

Q: Explain the concept of Angular zones and how Zone.js impacts Angular's change detection.

Scenario: Your app is performing unnecessary change detection checks in zones, affecting performance. How would you reduce the impact and optimize change detection?

Q: How would you implement server-side rendering (SSR) with Angular Universal?

Scenario: SEO is critical for the project you are leading, and you need to improve the initial load time for users on slow networks. How would you implement Angular Universal to achieve this?

9. Component Communication

Q: How would you handle parent-child communication in Angular components?

Scenario: You have two components, one parent and one child. The parent needs to pass data to the child, and the child needs to send events to the parent. How do you achieve this?

Q: How do you manage communication between sibling components in Angular?

Scenario: Two sibling components need to share state or data. How would you facilitate this without tightly coupling the components?

10. Change Detection

Q: What are the different change detection strategies in Angular, and when should you use each?

Scenario: Your app is facing performance issues because Angular is running unnecessary change detection cycles. How would you use OnPush strategy to optimize performance?

Q: Explain the concept of zones in Angular and how they relate to change detection.

Scenario: Change detection is triggered more frequently than necessary in your application. How would you control or limit the use of zones to improve performance?

11. Pipes

Q: What is the difference between pure and impure pipes in Angular?

Scenario: You need a pipe that reacts to changes in arrays or objects. How would you decide whether to use a pure or impure pipe?

Q: How would you create a custom pipe in Angular?

Scenario: You need a custom transformation for formatting data in your application. Walk through the process of creating and applying a custom pipe.

12. Directives

Q: What is the difference between structural and attribute directives in Angular?

Scenario: You need to apply dynamic styling and hide/show elements based on conditions. How do you choose between structural and attribute directives?

Q: How would you create a custom directive in Angular?

Scenario: You need to create a directive that dynamically changes the behavior of an element. How would you design and implement it?

13. HTTP and Observables

Q: How do you handle HTTP requests in Angular using HttpClient?

Scenario: You need to interact with a RESTful API to fetch and submit data. Walk through how you would use the HttpClient service in Angular to manage requests.

Q: How do you handle error responses from HTTP requests?

Scenario: Your app needs to display proper messages when HTTP requests fail. How do you handle errors in the HttpClient service?

14. Observables and RxJS

Q: How would you implement debouncing or throttling using RxJS in Angular?

Scenario: Your search input triggers HTTP requests on every keypress. How would you reduce the number of API calls made by implementing debouncing?

Q: What are subjects in RxJS, and how do they differ from regular observables?

Scenario: You need a shared stream of data between components and services. How would you utilize subjects to achieve this?

15. Angular CLI & Build Tools

Q: How would you use Angular CLI to create a new Angular project with specific configurations?

Scenario: You are tasked with creating a new Angular project with routing and SCSS as the stylesheet format. How would you configure this using the CLI?

Q: How do you configure environment-specific settings in an Angular application?

Scenario: You need to handle different configurations for development, staging, and production environments. How would you set this up in Angular?

16. Security

Q: How do you secure Angular applications against common vulnerabilities like XSS or CSRF?

Scenario: You need to protect user data in an Angular application. What are some best practices for preventing vulnerabilities such as cross-site scripting (XSS) or cross-site request forgery (CSRF)?

Q: How would you implement role-based access control in Angular?

Scenario: Your app has different user roles, each with varying permissions to access features and routes. How would you enforce role-based access in Angular?

17. Progressive Web Applications (PWA)

Q: How would you convert an Angular app into a Progressive Web App (PWA)?

Scenario: You need to make your Angular application available offline and installable on user devices. Walk through the process of converting it into a PWA.

Q: How do you handle service workers in an Angular PWA?

Scenario: You need to ensure your Angular PWA caches assets and handles network requests efficiently. How would you configure and manage service workers?

18. Animations

Q: How would you implement animations in an Angular application?

Scenario: You need to add smooth animations when elements appear, disappear, or change state in the UI. How would you implement animations using Angular's built-in animation framework?

Q: How do you optimize Angular animations for performance?

Scenario: Your animations are causing UI lag. How would you ensure they are smooth and performant?

19. Module Federation

Q: What is Module Federation in Angular, and how would you implement it?

Scenario: You are tasked with creating a microfrontend architecture where different teams build independent parts of the application. How would you use Module Federation in Angular for this?

Q: How do you handle shared dependencies in a Module Federation setup?

Scenario: Multiple Angular applications share common libraries like Angular Material or RxJS. How would you manage shared dependencies across these apps?

20. Testing Advanced

Q: How would you test Angular services that have dependencies?

Scenario: You have a service that relies on several other services. How would you use dependency injection and mocks to test this service effectively?

Q: How do you ensure test coverage for asynchronous code in Angular (e.g., HTTP calls)?

Scenario: You have services that make HTTP calls. How would you write tests that properly handle and verify asynchronous responses?

21. Dependency Injection

Q: How do you manage multi-provider injection in Angular?

Scenario: You need to inject multiple instances of the same service with different configurations. How would you achieve this using Angular’s dependency injection system?

Q: How does Angular's dependency injection differ between component-level and root-level providers?

Scenario: You want to create a service that has different instances in different parts of the application. How do you control the scope of a service using providers?

22. Versioning and Migration

Q: How do you handle breaking changes when upgrading Angular versions?

Scenario: You are migrating a project from Angular 12 to 15, but there are breaking changes.

How do you ensure the upgrade goes smoothly and manage backward compatibility?

Q: How would you manage dependency updates for a large Angular project?

Scenario: The project uses several third-party libraries that need updating. How do you manage the update process without introducing bugs or conflicts?

23. Architectural Decisions

Q: How would you structure a mono repo for managing multiple Angular applications and shared libraries?

Scenario: You are tasked with managing several Angular applications and a set of shared libraries in a single monorepo. How would you structure it to ensure maintainability and scalability?

Q: What are the key considerations when designing a modular Angular application?

Scenario: You are architecting an Angular app that needs to be divided into multiple feature modules. How do you ensure each module is isolated, reusable, and scalable?

24. Angular Compiler & Ivy

Q: Explain Angular's Ivy engine and its impact on application performance and bundle size.

Scenario: Your team is upgrading from Angular View Engine to Ivy. What are the key benefits of Ivy in terms of application size, performance, and backward compatibility?

Q: How would you optimize build performance in a large Angular application using Ivy?

Scenario: Your build times are becoming excessively long in a large Angular project. How would you leverage Ivy to improve build performance?

25. Microfrontends in Angular

Q: How do you architect microfrontends in an Angular ecosystem?

Scenario: You need to break down a large Angular app into microfrontends, each owned by a different team. How would you handle shared dependencies, routing, and inter-microfrontend communication?

Q: What are the pros and cons of using microfrontends in Angular applications?

Scenario: Management wants to adopt microfrontends for better team isolation. What challenges do you foresee with this approach, and how would you mitigate them?

26. API Integration & GraphQL

Q: How do you integrate GraphQL with Angular for complex data queries?

Scenario: Your application requires efficient fetching of nested data from the backend. How would you integrate GraphQL into your Angular app, and what considerations would you make for performance?

Q: Compare the use of REST and GraphQL in Angular applications.

Scenario: Your team is debating whether to use REST or GraphQL for a new Angular project. What are the pros and cons of each approach in the context of Angular?

27. Advanced State Management

Q: How do you manage complex side effects in NgRx applications?

Scenario: You have a complex NgRx-based application that needs to handle several asynchronous actions, like API requests and caching. How would you architect the effects to handle these side effects efficiently?

Q: What are the common pitfalls when using NgRx, and how can they be avoided?

Scenario: You're introducing NgRx in a large, feature-rich application. What challenges have you experienced with NgRx, and how do you mitigate them?

28. Lazy Loading & Optimization

Q: How would you optimize the loading speed of an Angular application using lazy loading?

Scenario: Your application has slow initial load times due to many feature modules. How would you use lazy loading to improve this without impacting user experience?

Q: How would you handle third-party libraries in an Angular app to ensure they don't bloat the bundle size?

Scenario: You need to use large third-party libraries in your Angular project. How do you ensure they don't negatively impact the bundle size or load times?

29. Server-Side Rendering (SSR)

Q: How would you improve SEO for an Angular app using Angular Universal?

Scenario: Your client's Angular app requires better search engine indexing. How would you implement server-side rendering to improve SEO and optimize the initial page load?

Q: What are the challenges associated with Angular Universal, and how do you overcome them?

Scenario: You are implementing Angular Universal for SSR, but facing issues like asynchronous content not rendering properly. How would you debug and resolve these problems?

30. Internationalization (i18n)

Q: How do you implement internationalization (i18n) in an Angular application?

Scenario: You are developing a multilingual application that needs to support multiple languages, including handling dynamic language changes. How would you implement i18n in Angular?

Q: How would you manage large translation files and ensure the application's performance doesn't degrade?

Scenario: Your app requires extensive translation files that are growing in size. How would you efficiently manage these files and ensure the app's performance remains unaffected?

31. Security Best Practices

Q: How would you secure an Angular application's data flow to prevent man-in-the-middle attacks?

Scenario: Your Angular app handles sensitive financial information. How would you secure data flow, both in transit and at rest, to ensure protection against attacks?

Q: How do you prevent and mitigate common security vulnerabilities in Angular, such as XSS and CSRF?

Scenario: You've identified a potential XSS vulnerability in your Angular app. How would you fix it and ensure similar issues don't arise again?

32. Advanced Routing Techniques

Q: How would you implement route reuse in Angular?

Scenario: Your application contains routes that need to persist their state even when navigating away. How would you implement route reuse strategies to preserve data and performance?

Q: How do you implement custom preloading strategies in Angular routing?

Scenario: Certain routes in your application should be preloaded based on user behavior. How would you implement a custom preloading strategy to optimize performance?

33. Component Libraries and Customization

Q: How do you create a reusable component library in Angular?

Scenario: You are tasked with developing a shared component library that will be used across multiple Angular applications. How would you design, build, and distribute it?

Q: How would you customize a third-party Angular component library to suit project-specific requirements?

Scenario: Your project relies on a third-party Angular UI library, but certain components don't meet the design guidelines. How would you modify or extend these components?

34. Web Workers in Angular

Q: How do you implement web workers in Angular to handle intensive background tasks?

Scenario: Your Angular app needs to perform computationally expensive operations without blocking the UI. How would you implement web workers to handle these tasks?

Q: How would you manage communication between the main thread and web workers in Angular?

Scenario: Your app requires efficient communication between the main thread and a web worker performing a background task. How would you design this communication?

35. Legacy Code Migration

Q: How would you migrate an AngularJS project to Angular 15+?

Scenario: Your company has a large legacy AngularJS application that needs to be upgraded to Angular 15. How would you approach this migration, and what tools would you use?

Q: What are the challenges of migrating a large AngularJS codebase to the latest Angular version, and how do you overcome them?

Scenario: During the migration process, certain features from AngularJS are not compatible with the latest Angular. How would you handle these incompatibilities?

36. Behavioral & Team Leadership

Q: How do you ensure code quality in a large Angular project with multiple teams?

Scenario: You're leading a large Angular project with distributed teams. How do you enforce code quality, code reviews, and consistent standards across teams?

Q: Describe a time when you had to lead an Angular project with tight deadlines. How did you manage both the technical and human aspects of the project?

Scenario: Your team is under pressure to deliver a complex Angular app on a tight deadline. How do you manage team workload, expectations, and technical debt?