Angular Interview Packet (For 2+ Years Experience)

Overview

- **Candidate Level:** 2+ Years
- **Duration:** 60 Minutes
- **Format:** Theory + Coding Task + Evaluation
- **Tools:** StackBlitz / VS Code / Zoom Screen Share

Interview Flow

1. **Introduction (5 min)** – Greeting, overview of role.
2. **Core Questions (20–25 min)** – Angular, TypeScript, RxJS, Routing.
3. **Practical Task (20–25 min)** – Live coding or pair debugging.
4. **Architecture & Wrap-up (10 min)** – Discussion + feedback.

Section 1: TypeScript & JavaScript Fundamentals

1. Difference between `var`, `let`, and `const`.
2. What are arrow functions and their benefits?
3. Explain hoisting.
4. What is a closure?
5. What are generics in TypeScript?
6. Difference between interface and type.
7. What are access modifiers (`public`, `private`, `protected`)?
8. Why is TypeScript used in Angular?
9. How to handle null and undefined safely?
10. What are decorators in Angular?

Section 2: Core Angular Concepts

1. What is Angular? How is it different from AngularJS?
2. Explain Angular architecture.
3. Define Components, Modules, and Directives.
4. Describe lifecycle hooks (`ngOnInit`, `ngOnDestroy`).
5. Explain Dependency Injection.
6. Describe data binding and its types.
7. Explain Change Detection and strategies.

8. What are pipes? Create a custom pipe.
9. What is lazy loading?
10. Template-driven vs Reactive forms.

Section 3: RxJS and Observables

1. What is an Observable?
2. Observable vs Promise.
3. Explain Subject, BehaviorSubject, ReplaySubject.
4. `switchMap` vs `mergeMap` vs `concatMap`.
5. How to unsubscribe from Observables?
6. How to handle HTTP error responses?
7. Purpose of `takeUntil`.
8. Example use of `debounceTime`.

Section 4: Routing, Services, State

1. How does routing work in Angular?
2. What are Route Guards? (`CanActivate`, `CanDeactivate`)
3. How to share data between unrelated components?
4. How to call APIs using HttpClient?
5. What are Interceptors?
6. Explain state management methods (Service, BehaviorSubject, NgRx).
7. How to implement global error handling?

Section 5: Performance Optimization

1. What is ChangeDetectionStrategy.OnPush?
2. What is trackBy and why is it used?
3. How to improve Angular performance?
4. What is AOT compilation?
5. Lazy loading and preloading strategies.
6. How to reduce bundle size?
7. Tools for performance analysis.

Section 6: Testing & Best Practices

1. How to test a component or service?
2. Difference between TestBed and ComponentFixture.
3. How to mock HTTP requests?
4. Karma vs Jasmine vs Jest.

5. Best practices for Angular code.

Section 7: Behavioral

1. Tell me about a complex bug you fixed.
2. How do you manage deadlines and code quality?
3. How do you stay updated with Angular changes?
4. How do you handle code reviews?

Practical Tasks

Task 1: User Search with Debounce + API

Objective: Test RxJS, HttpClient, and component logic.

Requirements

- Input box for search
- API call to `https://jsonplaceholder.typicode.com/users?name_like=<search>`
- Use `debounceTime(300)` and `switchMap`
- Show loading and error states
- Display results list

Evaluation Points: | Area | Key Check | |------|------------| | RxJS | Uses debounce, switchMap correctly | | Async Handling | Proper loading/error management | | Structure | Clean code separation | | UX | Works smoothly, no reload spam |

Task 2: Reactive Form with Conditional Validation

Objective: Test forms, validation, and dynamic logic.

Requirements

- Fields: name, email, password, role (Admin/User)
- If Admin selected → show Admin Code (required)
- Reactive Form, proper validation, submit logs data

Evaluation Points: | Area | Key Check | |------|------------| | FormGroup | Correct FormBuilder use | | Validation | Dynamic validator setup | | UX | Error messages, input handling | | Code | Clean, modular, readable |

Evaluation Scorecard

Category	Max	Candidate	Notes
TypeScript & JS	10		
Angular Core	20		
RxJS	15		
Forms & Routing	15		
Coding Task	25		
Communication	5		
Total	**90–100**		

Interviewer Checklist

- [ ] Prepare StackBlitz/VSCode setup
- [ ] Review candidate resume
- [ ] Conduct intro
- [ ] Ask from each section (time-boxed)
- [ ] Conduct one coding task
- [ ] Score objectively & write notes immediately

Tip: Always ask follow-up questions like "*Why did you choose that operator?*" to gauge understanding beyond memorization.