Angular Interview Guide – Complete Explanation

1. Core Angular Concepts

What is Angular?

Angular is a TypeScript-based open-source front-end web application platform developed by Google. It allows developers to build single-page applications (SPAs) using HTML, CSS, and TypeScript.

Angular Architecture

Angular follows a modular architecture made of components, services, and modules. - **Modules**: Containers for components, services, directives, pipes. - **Components**: The basic UI building block. - **Templates**: Define the HTML layout. - **Services**: Used for business logic and data access. - **Dependency Injection**: Angular provides built-in DI for services.

Angular CLI

Command Line Interface tool for initializing, developing, scaffolding, and maintaining Angular applications.

Component Lifecycle Hooks

Lifecycle methods Angular calls during a component's life: - `ngOnInit()`: Called after component initialization. - `ngOnChanges()`: Called when input properties change. - `ngDoCheck()`, `ngAfterViewInit()`, `ngOnDestroy()`, etc.

Data Binding

- **Interpolation**: `{{ value }}`
- **Property Binding**: `[src]="imgUrl"`
- **Event Binding**: `(click)="doSomething()"`
- **Two-way Binding**: `[(ngModel)]="value"`

Directives

- **Structural**: `*ngIf`, `*ngFor`
- **Attribute**: `ngClass`, `ngStyle`
- **Custom Directives**: Create using `@Directive()`

Pipes

- Transform data in templates: `{{ dateVal | date }}`
- Custom pipes: `@Pipe({ name: 'myPipe' })`

---

2. Component Communication

@Input and @Output

- `@Input()`: Pass data from parent to child.
- `@Output()`: Send data from child to parent using `EventEmitter`.

ViewChild and ContentChild

- `@ViewChild()`: Access child components or DOM elements.
- `@ContentChild()`: Access projected content inside `<ng-content>`.

Service-based Communication

- Share data using RxJS `Subject` / `BehaviorSubject` in services.

---

3. Routing and Navigation

RouterModule

Defines routes using `Routes[]` and `RouterModule.forRoot()`.

Lazy Loading

Load modules only when routes are accessed to improve performance.

Route Guards

- `CanActivate`: Block route access.
- `CanDeactivate`: Prevent navigation away.
- `Resolve`: Pre-fetch data before loading a route.

Route Parameters

- `/product/:id` for params
- `/product?id=123` for queryParams

---

4. Forms in Angular

Template-driven Forms

Use `ngModel`, declared in HTML. Suitable for simple forms.

Reactive Forms

Use `FormGroup`, `FormControl`, and `FormBuilder` in TS file. Suitable for complex validations.

Validators

- Sync: `Validators.required`, `Validators.minLength`
- Async: Custom validators using observables

FormArray

Dynamic list of controls.

---

5. HTTP Client and Services

HttpClient

Use `HttpClientModule` to perform HTTP requests (GET, POST, etc.)

Interceptors

Intercept requests/responses for headers, auth tokens, etc.

Error Handling

Use `catchError` and centralized error service.

Observables vs Promises

Observables support multiple values over time; Promises resolve once.

---

6. RxJS in Angular

Key Concepts

- `Observable`, `Subject`, `BehaviorSubject`
- Operators: `map`, `switchMap`, `mergeMap`, `debounceTime`, `takeUntil`
- Use `async` pipe to auto-unsubscribe

---

7. Dependency Injection (DI)

Angular's mechanism to provide and inject services: - `@Injectable({ providedIn: 'root' })` - Provided at component/module level - Use `constructor(private myService: Service)` to inject

---

8. Advanced Angular Concepts

Change Detection

- Default: Checks entire component tree
- OnPush: Only when input reference changes

TrackBy in *ngFor

Improves performance by reducing re-rendering

Renderer2

Abstracts DOM manipulation for platform independence

Custom Structural Directives

Manipulate DOM using `TemplateRef`, `ViewContainerRef`

---

9. Performance Optimization

- Lazy Load Modules
- Use `trackBy` in `*ngFor`
- Detach change detectors for static views
- Use `OnPush` strategy
- Avoid memory leaks by unsubscribing

---

10. Testing in Angular

Unit Testing

Use Jasmine + Karma - `TestBed.configureTestingModule()` - Use `spyOn`, `fixture.detectChanges()`

Service Testing

Mock dependencies and HTTP calls

E2E Testing

Use Cypress or Playwright (Protractor is deprecated)

---

11. Project Structure & Best Practices

- Organize by feature modules
- CoreModule: singleton services
- SharedModule: shared pipes/directives/components
- Use `environment.ts` for configurations
- Strict typing, async pipe, avoid logic in templates

---

12. Security

- Use Angular's sanitizer for DOM content
- JWT Authentication with interceptors
- Avoid exposing tokens in frontend
- Enable CORS in backend

---

13. Angular Universal (SSR)

- Server-side rendering for SEO and faster load
- Uses Node.js to render app on server
- Use `@nguniversal/express-engine`

---

14. Miscellaneous

Animations

Use `@angular/animations` with `trigger`, `state`, `transition`

i18n

Support multiple languages with Angular's internationalization API

PWA Support

Enable service workers with `ng add @angular/pwa`

Firebase Integration

Use AngularFire to integrate Firebase services

Let me know if you want interview Q&A per section.