Angular Services - Detailed Concepts

1. Minimal Angular Dev Environment Setup

- Node.js v18 managed via NVM
- Angular CLI installed globally: npm install -g @angular/cli
- Configured PATH in ~/.zshrc for npm global bin
- Project scaffolding: ng new angular-service-demo
- TypeScript configured via tsconfig.json (strict mode recommended)
- Use: ng serve to run dev server

2. Default Angular Services

Categories:

- Platform Browser: Title, Meta
- HTTP: HttpClient, HttpHeaders
- Routing: Router, ActivatedRoute
- Forms: FormBuilder, Validators
- Observables: Subject, BehaviorSubject
- Others: Renderer2, DOCUMENT, ViewContainerRef

3. Angular AsyncPipe

- Used to auto-subscribe/unsubscribe from Observables or Promises
- Syntax: *ngIf="user$ | async as user"
- Cleaner than manual subscription
- Common in *ngFor and *ngIf use cases
- Avoid multiple async pipes in template -> assign once using 'as'

4. Custom Services and @Injectable()

- Created using Angular CLI: ng generate service services/logger
- Marked with @Injectable({ providedIn: 'root' }) for global singleton
- Use in component: constructor(private logger: LoggerService) {}
- Service scope options: root, any, platform, component-level
- Required for injecting other services inside this service

Angular Services - Detailed Concepts

5. Fetching Data via HttpClient in Custom Service

- HttpClientModule must be imported in AppModule

- Service method: getPosts(): Observable<Post[]>

- Component uses: posts$ = this.dataService.getPosts();

- Template uses: *ngIf="posts$ | async as posts

6. Modifying Data - CRUD Operations with HttpClient

Service methods:

- Create: post() - this.http.post<Post>(this.API_URL, post)

- Read: get(), getById()

- Update: put() - replaces entire resource

- Patch: patch() - partial update

- Delete: delete()

Best Practices:

- Always return Observable<T>

- Handle errors using catchError

- Abstract business logic in services, not components

- Use interfaces for data types

- Stateless services promote testability