Angular

Standalone components, signals and control flow.

01 Install & CLI

```
npm install -g @angular/cli
ng new my-app
ng serve --open          # localhost:4200
ng build                 # production build
ng g c user-card         # component
ng g s services/auth     # service
```

--dry-run previews what ng g would write. Upgrade with ng update @angular/cli @angular/core.

02 Standalone component

```
@Component({
  selector: 'app-profile',
  imports: [Avatar],
  template: `<app-avatar /> {{ name }}`,
})
export class Profile {
  name = 'Alice';
}
```

Standalone is the default (v19+): no standalone: true, no NgModule. imports lists what the template uses.

03 Template syntax

```
<h1>{{ title }}</h1>
<p>{{ price | currency }}</p>
<img [src]="imageUrl" [alt]="name">
<div [class.active]="isActive"></div>
<button (click)="save($event)">Save</button>
<input #q (keyup.enter)="search(q.value)">
<input [(ngModel)]="name">
```

[(ngModel)] needs FormsModule. Only key events take modifiers; call event.preventDefault() yourself.

04 Control flow

```
@if (user) { <p>Hi, {{ user.name }}</p> }
@else { <p>Sign in</p> }
@for (item of items; track item.id) {
  <li>{{ $index }}: {{ item.name }}</li>
} @empty { <li>No items</li> }
@switch (status) {
  @case ('error') { <p>Error</p> }
  @default { <p>Ready</p> }
}
```

No imports needed. track is required: a unique id, or $index for a static list. @if (user$ | async; as u) names the result.

05 Signals

```
count = signal(0);
double = computed(() => this.count() * 2);
constructor() {
  effect(() => console.log(this.count()));
}
add() { this.count.update((n) => n + 1); }
```

Read by calling: {{ count() }}. Components are OnPush by default (v22), so keep async state in signals.

06 Inputs, outputs & model

```
stars = input.required<number>();
label = input('Rate');       // with default
rated = output<number>();    // an event
open = model(false);         // two-way
```

```
<app-rating [stars]="5" [(open)]="isOpen" />
```

Read this.stars(); emit with this.rated.emit(5). They replace @Input() and @Output().

07 Lifecycle hooks

constructor	inject() here; no inputs yet
ngOnChanges	An input changed
ngOnInit	Once, after the first inputs
ngAfterViewInit	View and children are ready
ngOnDestroy	Clean up subscriptions, timers

Content hooks run before view hooks.

08 Queries & content projection

```
child = viewChild(Avatar);   // or undefined
items = viewChildren<ElementRef>('item');
```

```
<!-- card.html -->
<ng-content select="[card-header]" />
<ng-content />
<!-- usage -->
<app-card><h2 card-header>Hi</h2>
  <p>Body</p></app-card>
```

Queries are signals; contentChild() reads projected content.

09 Pipes

```
@Pipe({ name: 'truncate' })
export class TruncatePipe {
  transform(text: string, limit = 100) {
    return text.length > limit
      ? text.slice(0, limit) + '...' : text;
  }
}
```

{{ bio | truncate: 50 }}. Built in: date, currency, number, uppercase, slice, json, async.

Angular

Services, routing, forms, data and tests.

10 Services & DI

```
@Injectable({ providedIn: 'root' })
export class UserService {
  private http = inject(HttpClient);
  getAll() {
    return this.http.get<User[]>('/users');
  }
}
```

Inject it anywhere with inject(UserService). Plain values use an InjectionToken: { provide: API_URL, useValue: '/api' }.

11 Routes & navigation

```
export const routes: Routes = [
  { path: 'users/:id', component: Profile },
  { path: 'admin', loadChildren: () =>
      import('./admin/routes')
        .then((m) => m.ADMIN_ROUTES) },
  { path: '**', component: NotFound },
];
```

```
<a [routerLink]="['/users', id]">Profile</a>
<router-outlet />
```

Register with provideRouter(routes); ** goes last. Navigate with inject(Router).navigate(['/users', id]); read :id from inject(ActivatedRoute).snapshot.paramMap.

12 Guards

```
const authGuard: CanActivateFn = () =>
  inject(AuthService).isLoggedIn() ||
  inject(Router).createUrlTree(['/login']);
```

Add canActivate: [authGuard] to a route. Return a UrlTree to redirect, not false. Resolvers use ResolveFn<T>.

13 Reactive forms

```
form = inject(FormBuilder).group({
  name: ['', Validators.required],
  email: ['', Validators.email],
});
```

```
<form [formGroup]="form"
      (ngSubmit)="save()">
  <input formControlName="email">
</form>
```

Needs ReactiveFormsModule. Read form.valid, form.value.

14 HttpClient & interceptors

```
this.http.get<User[]>('/users',
  { params: { q: term } });
this.http.post<User>('/users', user);
const auth: HttpInterceptorFn = (r, next) =>
  next(r.clone({ headers: r.headers.set(
    'Authorization', `Bearer ${token}`) }));
```

HttpClient needs no setup since v21. Add interceptors with provideHttpClient(withInterceptors([auth])).

15 RxJS operators

switchMap	Cancel the previous call: search
mergeMap	Run every call in parallel
concatMap	One at a time, in order
exhaustMap	Ignore new events while busy
forkJoin	Wait for all to complete
debounceTime(300)	Wait for typing to pause

Unsubscribe with takeUntil or the async pipe.

16 httpResource & linkedSignal

```
user = httpResource<User>(
  () => `/api/users/${this.userId()}`);
method = linkedSignal(() =>
  this.cart().length > 5 ? 'air' : 'road');
```

```
@if (user.hasValue()) {
  <p>{{ user.value().name }}</p>
} @else if (user.error()) { <p>Failed</p> }
```

value() throws in the error state, so guard it.

17 Deferrable views

```
@defer (on viewport; prefetch on idle) {
  <heavy-chart [data]="data" />
} @placeholder (minimum 500ms) {
  <p>Chart goes here</p>
} @loading (after 100ms; minimum 1s) {
  <app-spinner />
}
```

Triggers: on idle (default), viewport, interaction, hover, immediate, timer(5s), when cond.

18 Testing

```
it('shows the name', async () => {
  const fixture =
    TestBed.createComponent(Profile);
  await fixture.whenStable();
  expect(fixture.nativeElement.textContent)
    .toContain('Alice');
});
```

ng test runs Vitest in new projects. Mock HTTP calls with provideHttpClientTesting() and HttpTestingController.