

# 7-Day Angular Training Program

**Goal:** By the end, trainees can build, test, optimize, and deploy a production-ready Angular application with real-world features.

---

## Day 1 — Angular Basics & Setup

What to Learn (Why it Matters)

- Install Angular CLI (`ng new`, `ng serve`, project structure).
- Components → `@Component` decorator, templates, styles.
- Modules → `AppModule`, why Angular apps are modular.
- Data Binding → Interpolation `{{ }}`, Property binding `[ ]`, Event binding `( )`.
- Directives → `*ngIf`, `*ngFor`.

Task:

- Create a new Angular project `ng-shop`.
- Build a **Product Card** component with `@Input()` props for product details (name, price, image).

Deliverable: Running Angular app showing a grid of product cards.

---

## Day 2 — Components & Forms

What to Learn

- Component communication → `@Input()`, `@Output()` + `EventEmitter`.
- Template-driven forms (`ngModel`).
- Validation basics (`required`, `minlength`).
- Pipes (built-in: `date`, `currency`, custom pipes).

Task:

- Add a **Product List** page with parent-child communication.
- Create an **Add Product Form** (template-driven) with validation.

- Display product prices using Angular's `currency` pipe.

Deliverable: Products list + add new product with validation.

---

## Day 3 — Services, Dependency Injection & Routing

What to Learn

- Services & DI (`@Injectable`, singleton services).
- Mock data service for products.
- Routing basics (`RouterModule`, `routerLink`).
- Nested routes, route parameters.

Task:

- Create a **Product Service** to manage product data.
- Add **Routing**: Home (`/`), Product List (`/products`), Product Details (`/products/:id`).
- Use a service to fetch product data dynamically.

Deliverable: Navigable app with product detail pages.

---

## Day 4 — HTTP, Observables & Reactive Forms

What to Learn

- `HttpClientModule` → `get`, `post`, `put`, `delete`.
- Observables (`subscribe`, `async` pipe).
- Reactive Forms (`FormGroup`, `FormControl`, `FormBuilder`).
- Error handling (`catchError`, global error handling).

Task:

- Connect the app to a **mock REST API** (json-server).
- Convert Add Product Form → Reactive Form.
- Add a **Signup Form** with reactive validation.

Deliverable: Products fetched from API + working signup form.

---

## Day 5 — Authentication, Guards & State

What to Learn

- JWT-based Auth basics (login/logout).
- Route Guards → `CanActivate`.
- State management → Service-based state, intro to NgRx/Zustand alternative.
- Local Storage for persisting tokens.

Task:

- Add **Login/Signup** with JWT (mock backend).
- Protect **Admin Routes** using `CanActivate`.
- Maintain user session with local storage.

Deliverable: Authenticated user flow with protected product management page.

---

## Day 6 — Advanced Angular (Performance, Interceptors, i18n)

What to Learn

- Interceptors for auth headers & error handling.
- RxJS operators (`map`, `filter`, `switchMap`, `retry`).
- Performance: Change Detection (`OnPush`), `trackBy`, lazy loading modules.
- Internationalization (`ngx-translate` or Angular i18n).
- Accessibility (a11y best practices, ARIA roles).

Task:

- Add **HTTP Interceptor** for attaching tokens + handling errors.
- Lazy load **Admin Module**.
- Add **Language Switcher** (English/Spanish).

Deliverable: Optimized, multilingual app with secure API calls.

---

## Day 7 — Testing, Deployment & Capstone 🚀

What to Learn

- Unit Testing with Jasmine + Karma (`ng test`).
- End-to-End (E2E) testing (Cypress/Playwright or Protractor).
- Build & Deployment → `ng build --prod`, deploy to Firebase Hosting / Vercel / Netlify.
- Angular Universal (SSR) for SEO basics.
- Progressive Web Apps (PWA) with Service Workers.

Final Capstone Project: **Mini E-Commerce Store**

Features:

- Product catalog with filtering & pipes.
- Login/Signup with JWT.
- Authenticated Admin Dashboard (add/edit/delete products).
- Cart + Checkout flow (local storage).
- Multi-language support.
- Responsive & accessible design.

Deliverable: Deployed Angular app (link on GitHub + live demo).

---

## Essential NPM Packages for Angular (Cheat Sheet)

Note: `npm i <pkg>` → runtime dependency | `npm i -D <pkg>` → dev-only

---

### Core / Setup

Angular CLI — project scaffolding, build, test.

```
npm install -g @angular/cli
```

---

### Data Fetching / State

- rxjs — reactive programming (observables/operators).  
(ships with Angular)

**axios** — Promise-based HTTP client (alt to HttpClient).

```
npm i axios
```

**@ngrx/store** — Redux-style state management.

```
npm i @ngrx/store @ngrx/effects @ngrx/entity  
@ngrx/store-devtools
```

---

## Forms & Validation

- **@angular/forms** — built-in (template + reactive forms).  
*(already included in Angular)*

**yup** — schema-based form validation.

```
npm i yup
```

**ngx-validators** — extra validators.

```
npm i ngx-validators
```

---

## Styling & UI

**bootstrap** — CSS framework.

```
npm i bootstrap
```

**tailwindcss** — utility-first CSS framework.

```
npm i -D tailwindcss postcss autoprefixer
```

```
npx tailwindcss init
```

**@angular/material** — official Material UI library.

```
npm i @angular/material @angular/cdk @angular/animations
```

**ngx-toastr** — toast notifications.

**npm i ngx-toastr**

---

## Utilities

**lodash** — utility functions.

**npm i lodash**

**date-fns** — modern date utilities.

**npm i date-fns**

**ngx-spinner** — loading spinner component.

**npm i ngx-spinner**

---

## Internationalization & Accessibility

**@ngx-translate/core @ngx-translate/http-loader** — i18n.

**npm i @ngx-translate/core @ngx-translate/http-loader**

---

## Auth / Security

**jwt-decode** — decode JWT tokens.

**npm i jwt-decode**

**js-cookie** — cookie management.

**npm i js-cookie**

---

## Testing

**cypress** — modern E2E testing.

```
npm i -D cypress
```

**msw (Mock Service Worker)** — mock API for tests/dev.

```
npm i msw
```

---

## Backend / Mock APIs

**json-server** — quick REST API for dev.

```
npm i -D json-server
```

---

## Dev Tools / Code Quality

**eslint** — linting.

```
npm i -D eslint eslint-config-prettier
```

**prettier** — formatting.

```
npm i -D prettier
```

**husky + lint-staged** — pre-commit hooks.

```
npm i -D husky lint-staged
```