Angular

This 5-day accelerated Angular 21 program provides broad coverage of modern Angular development, combining essential fundamentals, current framework features, and selected hands-on exercises to help participants build practical working knowledge in a short time.

Overview

This 5-day accelerated Angular 21 program is designed for developers who need broad and practical exposure to modern Angular development within a short timeframe. The course covers TypeScript foundations, standalone architecture, components, forms, services, signals, RxJS, API integration, routing, testing, performance, security, and deployment, while also introducing current Angular capabilities such as Signal Forms, Resource API, accessibility, and internationalization. With selected hands-on exercises and guided demonstrations throughout the program, participants gain practical working knowledge of Angular 21 and a clear understanding of how its major features fit together in real-world application development.

Introduction

Angular 21 represents a modern evolution of frontend application development with a strong emphasis on performance, fine-grained reactivity, standalone APIs, and an improved developer experience. This course introduces participants to the latest Angular development model, including standalone components, modern control flow, signals-based state handling, and contemporary routing and HTTP integration patterns. The program is designed to help developers move beyond basic UI construction into building responsive, structured, and scalable applications using Angular current best practices. It provides a clear path from framework fundamentals to advanced implementation concepts, making it suitable for teams and professionals who want to work effectively with modern Angular in real projects.

Topics Covered

- Introduction to TypeScript
- Introduction to Angular
- Introduction to Single Page Applications
- Building with Components
- Data Binding and Event Handling
- Attribute Directives

- Modern Control Flow and Structural Rendering
- Template Driven Forms
- Reactive Forms
- Services and Dependency Injection
- Signals and Reactive Programming
- Signal Forms and Modern Form State
- RxJS and Observables
- HTTP Client and API Integration
- Modern Data Fetching with Resource API
- HTTP Resource
- Pipes and Data Formatting
- Accessibility and Internationalization
- Angular Routing and Navigation
- Component Lifecycle and Change Detection
- State Management Patterns
- Content Projection and Dynamic Components
- Testing in Angular
- Performance Optimization
- Security Best Practices
- Environment Configuration and Build

Prerequisites

Before attending this course, students should have general programming experience and knowledge of HTML, CSS, and JavaScript.

Audience:

Frontend developers, web developers, software engineers, and teams modernizing Angular applications.

Duration: 5 Days

Course Outline

Day 1 - TypeScript and Angular Foundation

1. Introduction to TypeScript
    - 1.1. Overview
    - 1.2. Environment Setup
    - 1.3. Basic Syntax
    - 1.4. Types, Variables and Operators
    - 1.5. Decision Making and Loops
    - 1.6. Functions and Arrow Functions
    - 1.7. Generics
    - 1.8. Enums
    - 1.9. Numbers and Strings
    - 1.10. Arrays
    - 1.11. Tuples
    - 1.12. Union and Intersection Types
    - 1.13. Interfaces, Classes and Objects
    - 1.14. Type Aliases
    - 1.15. Modules (ES Modules)
    - 1.16. Decorators
    - 1.17. Utility Types
    - 1.18. Summary
    - 1.19. Lab 1: TypeScript Fundamentals
        - Build a TypeScript-based inventory model
        - Define interfaces and classes
        - Use generics for reusable utilities
        - Implement enum and union types
        - Compile and run using ts-node
2. Introduction to Angular
    - 2.1. Why Angular 21
    - 2.2. Angular 21 Key Features
    - 2.3. Angular CLI Overview
    - 2.4. Installing and Using Angular 21
    - 2.5. Creating the First Angular Project
    - 2.6. Project Structure Overview
    - 2.7. Standalone Components Architecture
    - 2.8. Dependency Injection Overview
    - 2.9. Change Detection Overview
    - 2.10. What Is New in Angular 21
    - 2.11. Summary

2.12. Lab 2: Create Your First Angular App
Install Angular CLI
Create a standalone Angular 21 project
Generate components
Run and explore dev tools

3. Introduction to Single Page Applications
3.1 What Is a Single Page Application (SPA)
3.2 SPA Workflow
3.3 Traditional Web Application Architecture
3.4 SPA Advantages
3.5 SPA Challenges
3.6 Implementing SPAs Using Angular 21
3.7 SPA Using Standalone Components
3.8 Component-Based Navigation
3.9 Displaying Components Dynamically
3.10 Implementing SPA Using Angular Router
3.11 Summary
3.12 Lab 3: Build a Simple SPA Layout
Create layout components
Add navigation
Dynamically load views

Day 2 - Components, Binding and Forms

4. Building with Components
4.1. Introduction to Angular Components
4.2. Standalone Components
4.3. Component Metadata and Decorators
4.4. Templates and Inline Templates
4.5. Interpolation - Text, Objects and Arrays
4.6. Component Styling
4.7. View Encapsulation
4.8. Integrating External CSS Frameworks
4.9. Creating Header Component
4.10. Creating Footer Component
4.11. Creating Product Component
4.12. Summary
4.13. Lab 4: Build a Product Dashboard UI Create:
Header,
Sidebar,
Product List,
Product Card Component

5. Data Binding and Event Handling
    - 5.1. Angular Binding Syntax
    - 5.2. One-Way Data Binding
    - 5.3. Property Binding
    - 5.4. Attribute Binding
    - 5.5. Event Binding
    - 5.6. Event Binding Examples
    - 5.7. Template Reference Variables
    - 5.8. Two-Way Binding Using ngModel
    - 5.9. Input and Output Properties
    - 5.10. Parent-Child Component Communication
    - 5.11. Summary
    - 5.12. Lab 5: Parent-Child Communication
        - Pass product data to child components
        - Emit events from child to parent
        - Implement two-way binding

6. Attribute Directives
    - 6.1. What Are Directives
    - 6.2. Directive Types
    - 6.3. Built-in Attribute Directives
    - 6.4. Dynamic Class Binding
    - 6.5. Dynamic Style Binding
    - 6.6. Conditional Styling
    - 6.7. Dynamic Property Binding
    - 6.8. Controlling Element Visibility
    - 6.9. Dynamic Image and Hyperlink Binding
    - 6.10. Summary

7. Modern Control Flow and Structural Rendering
    - 7.1. Introduction to Angular Control Flow
    - 7.2. Conditional Rendering Using @if
    - 7.3. Else and Nested Conditions
    - 7.4. Looping Using @for
    - 7.5. Tracking Items with track
    - 7.6. Accessing Index and Context Variables
    - 7.7. Conditional Switching Using @switch
    - 7.8. Rendering Templates Dynamically
    - 7.9. Performance Benefits of Modern Control Flow
    - 7.10. Summary

8. Template Driven Forms
    - 8.1. Overview of Angular Forms
    - 8.2. Creating a Basic Angular Form
    - 8.3. Binding Input Fields
    - 8.4. Accessing Form State
    - 8.5. Handling Form Submission
    - 8.6. HTML5 Validation
    - 8.7. Angular Validation Overview
    - 8.8. Built-in Validators
    - 8.9. Validation States
    - 8.10. Displaying Validation Errors
    - 8.11. Disabling Submit for Invalid Forms
    - 8.12. Working with Checkboxes
    - 8.13. Select (Drop-Down) Fields
    - 8.14. Date and Number Inputs
    - 8.15. Radio Buttons
    - 8.16. Summary

9. Reactive Forms
    - 9.1. Introduction to Reactive Forms
    - 9.2. Registering Reactive Forms Module
    - 9.3. Creating FormControl
    - 9.4. Registering Controls
    - 9.5. Managing Control Values
    - 9.6. Creating FormGroup
    - 9.7. Connecting Model and View
    - 9.8. Nested Form Groups
    - 9.9. Updating Form Values
    - 9.10. Using FormBuilder
    - 9.11. Built-in Validators
    - 9.12. Custom Validators
    - 9.13. Dynamic Forms Using FormArray
    - 9.14. Summary
    - 9.15. Lab 6: Advanced Reactive Forms
    - 9.16. Create dynamic product form using FormArray
    - 9.17. Add custom validator
    - 9.18. Show real-time validation errors
    - 9.19. Submit and log structured JSON

Day 3 - Services, Signals and HTTP

10. Services and Dependency Injection
    - 10.1. What Is a Service
    - 10.2. Creating Services
    - 10.3. Dependency Injection Fundamentals
    - 10.4. Injecting Services
    - 10.5. Service Scopes
    - 10.6. Using Services in Components
    - 10.7. Shared Services
    - 10.8. Optional Dependencies
    - 10.9. Host Dependencies
    - 10.10. Summary
    - 10.11. Lab 7: Shared Product Service
      - Create centralized state service
      - Inject into multiple components
      - Refactor local state into service

11. Signals and Reactive Programming
    - 11.1. Introduction to Angular Signals
    - 11.2. Creating Signals
    - 11.3. Reading and Updating Signal Values
    - 11.4. Computed Signals
    - 11.5. Effect Functions
    - 11.6. Signal-Based State Management
    - 11.7. Signals vs Observables
    - 11.8. Using Signals with Components
    - 11.9. Signal-based component inputs
    - 11.10. Signal performance patterns
    - 11.11. Avoiding unnecessary effects
    - 11.12. Summary
    - 11.13. Lab 8: Signal-Based Cart System
      - Create cart signal
      - Compute total price
      - Auto-update UI using computed signals

12. Signal Forms and Modern Form State
    - 12.1. Introduction to Signal Forms
    - 12.2. Why Signal Forms
    - 12.3. Signal-based form state
    - 12.4. Validation patterns
    - 12.5. Comparing Signal Forms with Reactive Forms

- 12.6. When to use Signal Forms
- 12.7. Limitations and adoption considerations
- 12.8. Summary

13. RxJS and Observables
    - 13.1. Introduction to RxJS
    - 13.2. Observables Overview
    - 13.3. Creating Observables
    - 13.4. Subscribing and Unsubscribing
    - 13.5. Common RxJS Operators
    - 13.6. Subject and BehaviorSubject
    - 13.7. Async Pipe
    - 13.8. Interoperability Between Signals and Observables
    - 13.9. switchMap, mergeMap, concatMap
    - 13.10. Error handling with catchError
    - 13.11. takeUntil and memory management
    - 13.12. Combining streams
    - 13.13. Summary

14. HTTP Client and API Integration
    - 14.1. Angular HTTP Client Overview
    - 14.2. Setting Up HTTP Client
    - 14.3. Creating API Services
    - 14.4. Making HTTP GET Requests
    - 14.5. HTTP Headers and Options
    - 14.6. POST, PUT and DELETE Requests
    - 14.7. Handling HTTP Responses
    - 14.8. Error Handling Strategies
    - 14.9. Observables in HTTP
    - 14.10. Consuming APIs in Components
    - 14.11. HTTP Interceptors
    - 14.12. Authentication headers
    - 14.13. Global error handling
    - 14.14. Loading indicators
    - 14.15. Environment configuration
    - 14.16. Lab 9: REST API Integration
    - 14.17. Connect to mock API
    - 14.18. Implement CRUD operations
    - 14.19. Add interceptor for logging
    - 14.20. Handle loading and error states

15. Modern Data Fetching with Resource API and httpResource
    - 15.1. Introduction to Resource API
    - 15.2. Understanding httpResource
    - 15.3. Loading, value and error states
    - 15.4. Signal-friendly data fetching patterns
    - 15.5. Comparing httpResource with HttpClient and RxJS
    - 15.6. When to use each approach
    - 15.7. Summary

Day 4 - Routing, Architecture, and Advanced Angular

16. Pipes and Data Formatting
    - 16.1. What Are Pipes
    - 16.2. Built-in Pipes
    - 16.3. Using Pipes in Templates
    - 16.4. Chaining Pipes
    - 16.5. Using Pipes in TypeScript
    - 16.6. Decimal Pipe
    - 16.7. Currency Pipe
    - 16.8. Date Pipe
    - 16.9. Creating Custom Pipes
    - 16.10. Pure and Impure Pipes
    - 16.11. Summary

17. Accessibility and Internationalization
    - 17.1. Accessibility fundamentals in Angular
    - 17.2. Semantic HTML and accessible component design
    - 17.3. Keyboard navigation and focus management
    - 17.4. ARIA roles and attributes
    - 17.5. Accessible forms and validation feedback
    - 17.6. Introduction to Angular i18n
    - 17.7. Locale-aware formatting
    - 17.8. Dates, numbers, and currency by locale
    - 17.9. Structuring multilingual Angular applications
    - 17.10. Summary

18. Angular Routing and Navigation
    - 18.1. Routing Overview
    - 18.2. Angular Router Architecture
    - 18.3. Route Configuration Using Standalone APIs
    - 18.4. Router Outlet
    - 18.5. Navigation Links
    - 18.6. Programmatic Navigation

- 18.7. Route Parameters
- 18.8. Query Parameters
- 18.9. Retrieving Route Data
- 18.10. Lazy Loading Routes
- 18.11. Handling Invalid Routes
- 18.12. Route Guards (CanActivate, CanDeactivate)
- 18.13. Resolver
- 18.14. Preloading strategies
- 18.15. Standalone route configuration
- 18.16. Nested routes
- 18.17. Summary
- 18.18. Lab 10: Secure Admin Route
  - Implement login mock
  - Protect route using guard
  - Load admin module lazily

19. Component Lifecycle and Change Detection
    - 19.1. Lifecycle Hooks Overview
    - 19.2. ngOnInit, ngOnDestroy
    - 19.3. afterRender hooks (modern APIs)
    - 19.4. OnPush Change Detection
    - 19.5. Manual Change Detection
    - 19.6. Performance implications

20. State Management Patterns
    - 20.1. Local component state
    - 20.2. Service-based state
    - 20.3. Signal-based global state
    - 20.4. When to use RxJS vs Signals
    - 20.5. Introduction to NgRx (Conceptual Overview)

21. Content Projection and Dynamic Components
    - 21.1. ng-content
    - 21.2. Multiple slots
    - 21.3. ViewContainerRef
    - 21.4. Dynamic component loading
    - 21.5. Portals concept

Day 5 - Testing, Performance, Security, and Deployment

22. Testing in Angular
    - 22.1. Testing Strategy Overview
    - 22.2. Unit Testing Components
    - 22.3. Testing Services
    - 22.4. TestBed
    - 22.5. Mocking Dependencies
    - 22.6. Testing Reactive Forms
    - 22.7. Testing HTTP Calls
    - 22.8. Introduction to E2E Testing
    - 22.9. Lab 12: Write Unit Tests
      - Test product service
      - Test form validation
      - Mock HTTP calls

23. Performance Optimization
    - 23.1. Lazy Loading Best Practices
    - 23.2. Code Splitting
    - 23.3. OnPush Strategy
    - 23.4. trackBy Optimization
    - 23.5. Signals Performance Tuning
    - 23.6. Avoiding Memory Leaks
    - 23.7. Bundle Analysis

24. Security Best Practices
    - 24.1. XSS Prevention
    - 24.2. Angular Sanitization
    - 24.3. Route Protection
    - 24.4. Token Storage Strategies
    - 24.5. CORS Overview
    - 24.6. Secure HTTP Communication

25. Environment Configuration and Build
    - 25.1. Angular Environments
    - 25.2. Production Build
    - 25.3. AOT Compilation
    - 25.4. Build Optimization
    - 25.5. Source Maps
    - 25.6. Debugging Production Issues

26. Final Lab (Capstone Project)
    - 26.1. Build a complete Product Management System
    - 26.2. Features:
    - 26.3. Authentication mock
    - 26.4. Protected admin routes
    - 26.5. Product CRUD with API
    - 26.6. Reactive form validation
    - 26.7. Signal-based cart
    - 26.8. Animations
    - 26.9. Route guards
    - 26.10. HTTP interceptor
    - 26.11. Unit tests
    - 26.12. Production build