Mastering Angular

Duration

5 days

Description

Mastering Angular is a comprehensive course designed for developers who want to dive deep into the world of Angular and create efficient, dynamic single-page applications. This course begins by introducing Angular's core constructs, teaching students to use the Angular CLI for development, testing, and deployment, and explaining the differences between various Angular versions. Participants will learn to manage dependencies with NPM and leverage TypeScript for static and strong typing, enhancing application robustness. Advanced topics include building applications with the Standalone API, mastering Angular components, and managing application state with Signals. The course also covers building typed reactive forms, utilizing pipes for data formatting, setting up routing, and capturing URL data. Furthermore, students will learn to consume REST services using HTTP, perform CRUD operations, and conduct both unit and end-to-end testing, ensuring a solid understanding of building and maintaining high-quality Angular applications.

Objectives

- Utilize the Angular core constructs to build Single Page Applications
- Employ the Angular CLI to create, test, and deploy Angular Applications
- Differentiate between major Angular versions and different functionality
- Install necessary modules using NPM
- Embrace static and strong typing with TypeScript
- Build Angular Apps with the Standalone API
- Learn the best principles and practices of Angular Components
- Manage Application State with Signals
- Build Typed Reactive Forms
- Utilize Pipes and built-in data formatting tools
- Setup routing, capture data from URL paths and query strings
- Consume REST services by perfomring CRUD commands Http
- Unit test and E2E parts of the new application

Prerequisites

All attendees must have experience with modern JavaScript or TypeScript, including the new language features like classes, modules, arrow functions, and destructuring.

Training Materials

All students receive comprehensive courseware covering all topics in the course. Courseware is distributed via GitHub in the form of documentation and extensive code samples. Students practice the topics covered through challenging hands-on lab exercises.

Software Requirements

Students will need a free, personal GitHub account to access the courseware. Students will need permission to install Node.js and Visual Studio Code on their computers. Also, students will need permission to install NPM Packages and Visual Studio Extensions. If students are unable to configure a local environment, a cloud-based environment can be provided.

Outline

- Introduction
  - Why Angular?
  - Angular versions
  - Development Setup
  - Testing tools overview
  - Installing/updating packages
- Angular CLI
  - Installation Commands
  - Creating apps
  - Generating components, services, pipes
  - Modules vs Standalone APIs
  - Running Tests
- Bootstap Application
  - Bootstrap Function
  - Bootstrapping a Component
  - Application Configuration
  - App Component Selector and Index Html
- Component Templates
  - Class Data Properties
  - Double Curly Braces
  - Import Structural Directives

- ngIf, ngFor, and ngSwitch
- Extended ngIf, then, else, and ng-template
- ngFor and trackBy
- ngSwitch, ngSwitchCase, and ngSwitchDefault
- ngClass and ngStyle
- CSS Class.x and style.x bindings
- New Template Syntax: @for, @if, @switch

- Components
  - Principles of Components
  - Patterns of Components
  - Building Reusable Components
  - Decorators
  - Static Input
  - Input Expressions
  - Outputs and Events
  - Parent/Child Communication
  - Component Lifecycle
  - Component Composition
  - Calling Web Components from an Angular Component

- Forms
  - Reactive vs. Template forms
  - Typed Reactive Forms
  - FormGroup
  - FormControl
  - Validation
  - Custom Validators
  - Custom Async Validators

- Signals
  - What is Signal?
  - Reactive Programming
  - Using Signals to Manage State
  - Comparison to RxJS
  - Create a Signal
  - Signals API
  - Using a Signal in a Component
  - Using a Signal in a Service

- HTTP Data
  - Mocking a REST API with JSON Server
  - Http Client Module and Standalone Components
  - Async REST API Requests
  - Using Http to Update or Delete data on the server
  - Using RxJS Operators (map, filter, etc.)
  - Http Interceptors
- Pipes
  - What are pipes?
  - Intro to pipes
  - Built-in pipes
  - Piping examples
- Routing
  - What is routing?
  - Routing and Standalone Components
  - URL Structure
  - Path Parameters
  - Query String Parameters
  - Path Matching
  - Wildcard Routes
  - Child Routes
  - Lazy Loading
  - Route Guards
- Testing and Deploying to Production
  - Unit Testing with Jest
  - Configure an Angular Project for Jest
  - Testing with dependencies
  - Testing templates
  - Testing routed components
  - Http Testing Controller
  - Mocking HTTP Responses
  - E2E Testing
- Deploying to Production
  - Deployment Requirements
  - Using the CLI