Comprehensive Angular Programming

Course ISI-1506F 5 Days Instructor-led, Hands on

Course Description

This intensive 5-day, instructor-led Angular training course is a combination of theoretical learning and hands-on labs that includes an introduction to Angular, followed by TypeScript, components, directives, services, HTTPClient, testing, and debugging.

This Angular Training course is packed with useful and actionable information you can apply to your work right away. Learn the fundamentals of basic Angular development such as single-page browser applications, responsive websites, and hybrid mobile applications. Get started today with Angular - the most popular platform for building front-end web applications!

Course Objectives

In this course, you will learn how to:

- Explain Core Angular Concepts and Utilize the CLI
- Set up a complete Angular development environment
- Program using TypeScript
- Work with Angular CLI
- Create Standalone Components, Directives, Services, Pipes, Forms and Custom Validators
- Implement Data Binding and Template Control Flow
- Leverage Angular Signals for Reactivity
- Facilitate Inter-Component Communication
- Build Forms with the Template-Driven and Reactive Approaches
- Utilize Services and Dependency Injection (DI)
- Integrate with REST APIs using HttpClient
- Understand and work with Observables
- Work with Angular Pipes to format data
- Implement Basic Client-Side Routing
- Develop single page Angular applications
- Implement Advanced Routing Scenarios
- Manage Observable Subscriptions and State
- Master RxJS for Reactive Programming
- Write Comprehensive Unit Tests for Angular Applications
- Mock Dependencies and Test Angular Specifics
- Effectively Debug Angular Applications
- Analyze and Resolve Common Angular Errors
- Apply Best Practices for Advanced Angular Development
- Strategically Utilize NgModules

Contact ISInc for more information at 916.920.1700 or by visiting our website at http://www.isinc.com

- Integrate Standalone Components with NgModules

Prerequisites

Students should understand HTML structure, CSS styling, and fundamental JavaScript concepts. General knowledge regarding browser-based application and back-end data servers is useful.

Course Outline

Module 1: Introducing Angular

- What is Angular?
- Central Features of the Angular Framework (Highlighting Signals and Standalone Architecture)
- Appropriate Use Cases
- Angular Building Blocks
- Standalone Component Architecture
- Components, Directives, Pipes and Services
- NgModule Based Architecture
- Installing and Using Angular (CLI)
- Angular Example - with Standalone Component
- Running the Application
- Building and Deploying the Application

Module 2: Introduction to TypeScript

- Angular and TypeScript
- TypeScript Syntax
- The Type System - Defining Variables, Arrays, Primitives
- Type in Functions, Type Inference
- Defining Classes (Constructors, Methods, Visibility)
- Interfaces
- Working with ES6+ Modules
- let vs const (replacing var)
- Arrow Functions
- Template Strings
- Generics

Module 3: Standalone Components

- What is a Component?
- Creating Standalone Components

- The Component Class
- The @Component Decorator
- Component Templates (Inline vs External)
- Using a Component within another Standalone Component
- Component Hierarchy
- The Application Root Component (Standalone Bootstrap)
- Component Lifecycle Hooks
- CSS Styles and Encapsulation

Module 4: Component Templates & Control Flow

- Templates and Data Binding
- Interpolation {{ }}
- Property Binding [ ]
- Attribute Binding [attr.]
- Class Binding [class.] & Style Binding [style.]
- Event Binding ( )
- Two-Way Binding [(ngModel)] (Requires FormsModule or ReactiveFormsModule)
- Control Flow Syntax
- Conditional Rendering: @if, @else if, @else
- List Rendering: @for (with track requirement)
- Switching Logic: @switch, @case, @default
- Template Reference Variables (#var)
- The ng-template, ng-container elements (Use cases beyond basic control flow)
- Attribute Directives (ngClass, ngStyle - imported standalone or via CommonModule)
- Structural Directives

Module 5: Angular Signals & Reactivity

- Introduction to Angular Signals
- What are Signals? Why use them?
- Creating Writable Signals (signal())
- Reading Signal Values
- Updating Signals (set(), update())
- Computed Signals (computed())
- Creating Derived State
- Lazy Evaluation and Memoization
- Effects (effect())
- Running Side Effects in Response to Signal Changes
- Cleanup Logic (manualCleanup, DestroyRef)
- Signals in Components (OnPush change detection interaction)
- Using Signals for Inter-Component Communication (Alternative/Supplement to Input/Output)
- RxJS Interoperability (if needed)

Module 6: Inter-Component Communication

- Data Flow Architecture (Parent-to-Child, Child-to-Parent)
- Parent-to-Child: @Input() Decorator
- Child-to-Parent: @Output() Decorator with EventEmitter
- Using Template Reference Variables for Interaction
- Communication via Services (Shared State)
- Communication via Signals

Module 7: Template-Driven Forms

- Introduction to Template-Driven Forms
- Importing FormsModule (into standalone component or module)
- Setting Up a Form (ngForm)
- Getting User Input (ngModel)
- Two-Way Data Binding ([(ngModel)])
- Form Validation (Built-in validators: required, minlength, etc.)
- Displaying Validation State and Errors (ngControl status classes)
- Handling Different Input Types (Checkboxes, Selects, Radio Buttons)
- Form Submission (ngSubmit)

Module 8: Reactive Forms

- Introduction to Reactive Forms (Programmatic approach)
- Importing ReactiveFormsModule
- The Building Blocks: FormControl, FormGroup, FormArray
- Constructing Forms in the Component Class
- Connecting Template to Form (formGroup, formControlName, formArrayName)
- Getting/Setting Form Values (valueChanges, statusChanges, setValue, patchValue)
- Forms and type safety
- Built-in and Custom Validators
- Displaying Validation Errors
- Dynamic Forms with FormArray

Module 9: Services and Dependency Injection (DI)

- What is a Service?
- Creating a Service (@Injectable)
- Dependency Injection Overview
- Providing Services:
- Root Injector (providedIn: 'root') - Preferred method
- Component/Directive Providers (providers array in decorator)
- Module Providers (providers array in @NgModule - less common now)
- Injecting Services (Constructor Injection)
- Injector Hierarchy Basics

- Using Injection Tokens (InjectionToken)
- Optional Dependencies (@Optional)
- Controlling Visibility (@Host, @SkipSelf)

Module 10: HttpClient

- Using the Angular HttpClient for API Communication
- Importing provideHttpClient()
- Importing HttpClientModule
- Making Requests: GET, POST, PUT, DELETE
- Working with RxJS Observables for HTTP Responses
- Handling Responses (Typed Responses)
- Error Handling (catchError operator)
- Sending Data (Request Body)
- Setting Headers and Request Options
- Working with HttpResponse object (Accessing headers, status)
- Interceptors (Modifying Requests/Responses)

Module 11: Pipes and Data Formatting

- What are Pipes? Transforming Data in Templates
- Using Built-In Pipes (DatePipe, UpperCasePipe, LowerCasePipe, CurrencyPipe, DecimalPipe, PercentPipe, JsonPipe, AsyncPipe)
- Using Pipes in Standalone Components
- Chaining Pipes
- Passing Parameters to Pipes
- Creating Custom Pipes (@Pipe, PipeTransform)
- Standalone Pipes (standalone: true)
- Pure vs. Impure Pipes
- Using the AsyncPipe

Module 12: Routing Basics

- The Angular Component Router
- Setting up Routing (provideRouter - standalone approach)
- Defining Routes (Routes array)
- Displaying Routed Components
- Navigation:
- Declarative (routerLink, routerLinkActive)
- Programmatic (Router.navigate(), Router.navigateByUrl())
- Passing Route Parameters (/:id)
- Retrieving Route Parameters
- Query Parameters

Module 13: Advanced Routing

- Lazy Loading Standalone Components
- Route Guards
- CanActivate, CanActivateChild, CanDeactivate, Resolve
- Functional Guards
- Child Routes (Nested Routes)
- Multiple/Auxiliary Router Outlets
- Router Events

Module 14: Using NgModules

- When to use NgModules
- Creating a project that uses NgModules
- NgModule Basics (@NgModule decorator)
- NgModule Metadata Properties (declarations, imports, exports, providers, bootstrap)
- Setting up Routing for Modules
- Providing Services in NgModules
- Module level HttpClient Configuration
- Lazy Loading Modules
- Feature Modules vs. Root Module
- Shared Modules / Core Module patterns
- Using Standalone Components within NgModules
- Importing NgModules into Standalone components
- When to still use NgModules

Module 15: Observables & RxJS

- Introduction to Observables
- What are Observables? (vs. Promises)
- Advanced RxJS Operators for HTTP (map, filter, tap, switchMap, mergeMap, forkJoin, catchError, retry)
- Managing Subscriptions (AsyncPipe, takeUntil, take(1))
- Creating Observables
- Error Handling Strategies
- Caching HTTP Responses

Module 16: Unit Testing Angular Applications

- Introduction to Unit Testing (Jasmine, Karma/Jest)
- Setting up the Testing Environment (Angular CLI generates boilerplate)
- Writing Test Suites (describe) and Specs (it)
- Using Matchers (expect)
- Setup and Teardown (beforeEach, afterEach, etc.)
- Testing Standalone Components

Contact ISInc for more information at 916.920.1700 or by visiting our website at http://www.isinc.com

- Configuring TestBed for Standalone Components (imports)
- ComponentFixture, DebugElement
- Testing Services (with and without dependencies)
- Mocking Dependencies
- Testing Asynchronous Code (async, fakeAsync, tick)
- Testing Components with Inputs/Outputs
- Testing Routed Components (RouterTestingModule)
- Testing with HttpClientTestingModule
- Code Coverage

Module 17: Debugging Angular Applications

- Overview of Debugging Techniques
- Using Browser Developer Tools (Breakpoints, Console)
- Viewing TypeScript Code in Debugger
- Using debugger Keyword
- Debug Logging (console.log)
- Angular DevTools Browser Extension
- Inspecting Component Structure
- Profiling Change Detection
- Understanding and Debugging Common Errors

Module 18: Lab Exercises

- Getting Started with Angular
- Introduction to TypeScript
- Standalone Components
- Component Templates
- Angular Signals
- Inter-Component Communication
- Template-Driven Forms
- Reactive Forms
- Creating Services
- Using HttpClient
- Pipes and Data Formatting
- Routing Basics
- Advanced Routing
- NgModules Lab
- Observables & RxJS
- Unit Testing for Angular
- Debugging Angular Applications