Introduction to Angular Programming

Course ID #: 7000-1180-ZZ-Z
Hours: 21
Delivery Method: Group Internet Based

Course Content

Description:

This Angular training course gives learners a solid introduction to Angular and teaches them how to build modern, robust web applications. Learners dive into the core concepts of the Angular framework, with a special emphasis on the latest features like Signals for enhanced reactivity and the streamlined Standalone Component architecture. The course begins with Angular fundamentals and the essentials of TypeScript, laying a solid foundation for subsequent topics. Learners then dive into component creation, data flow, UI development, inter-component communication, forms (template-driven and reactive), services/DI, HttpClient, pipes, and routing. By the end of this Angular training, learners gain practical skills for developing feature-rich single-page applications.

Objectives:

Upon successful completion of this course, students will:

- 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

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.

Target Audience:

Developers and Software Engineers

Introduction to Angular Programming

Course ID #: 7000-1180-ZZ-Z
Hours: 21
Delivery Method: Group Internet Based

Topics:

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

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

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

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

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)

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

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)

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

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)

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)

Introduction to Angular Programming

Course ID #: 7000-1180-ZZ-Z
Hours: 21
Delivery Method: Group Internet Based

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

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

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

Register for this class by visiting us at:
www.tcworkshop.com or by calling us at 800-639-3535

NASBA CPE details are provided on the following pages.

Introduction to Angular Programming

Course ID #: 7000-1180-ZZ-Z
Hours: 21
Delivery Method: Group Internet Based

NASBA Information

Level: Intermediate
Advanced Preparation:
Attendance Requirement: To be awarded the full credit hours, you must sign in and attend the entire course.
Recommended Field(s) of Study: Computer Software & Applications
Recommended CPEs: 23.40

Policies: Course Registration, Cancellation, Refund, and Complaint Resolution
For more information regarding administrative policies such as complaint and program cancellation policies, please contact our offices at 800-639-3535 or visit us at: **www.tcworkshop.com**

Official National Registry Statement:
The Computer Workshop is registered with the National Association of State Boards of Accountancy (NASBA) as a sponsor of continuing professional education on the National Registry of CPE Sponsors. State boards of accountancy have final authority on the acceptance of individual courses for CPE credits. Complaints regarding registered sponsors may be submitted to the National Registry of CPE Sponsors through its website: www.nasbaregistry.org

NOTE: Since our information is in multiple places on our website or in PDF format that is sent to clients, we have provided our normal course content with the NASBA Information added along with links to our policy page on the web. We will add our name to the Official National Registry Statement after we are approved.