

# Developing Web Applications Using Angular

**Duration:** 5 Days

**US Price:** \$2795

**UK Price:** £1,995 *\*Prices are subject to VAT*

**CA Price:** CDN\$3,275 *\*Prices are subject to GST/HST*

**Delivery Options:** Attend face-to-face in the classroom, [remote-live](#) or via [on-demand streaming](#).

## Description

This course introduces Angular versions 2 through 6. Students interested in Angular 1.x should take the [Developing Web Applications Using AngularJS](#) course instead.

This hands on programming course provides a thorough introduction to the Angular JavaScript Framework including coverage of versions 2 through 6. Attendees will learn the fundamental skills necessary to build Web Applications using Angular and the MVVM (Model-View-ViewModel) design pattern. Topics include using TypeScript and ECMAScript 6 to create object-oriented Angular applications, extending HTML by creating reusable UI components, implementing data-binding, designing and using custom structural and attribute directives, as well as creating and using Angular pipes for formatting and transforming data in the UI. Students will explore creating UX's (User Experiences) by designing Web animations and implementing both template-driven and reactive style forms. Students will learn to use Angular routing to create SPA's (Single Page Applications). The course includes coverage of using DI (Dependency Injection) and Angular services to provide business and data-access logic to the application, both locally as well as communicating with RESTful web services to provide CRUD database operations.

Comprehensive hands on exercises are integrated throughout to reinforce learning and develop real competency.

## Prerequisites

Knowledge of HTML, CSS and JavaScript equivalent to attending the [Website Development with HTML5, CSS and Bootstrap](#) and [JavaScript Programming](#) courses.

Knowledge of jQuery is helpful, but not required.

## Course Overview

### Overview of Angular and the MVVM Design Pattern

- Features and Benefits of Angular
- Angular Architecture
- MVVM Design Pattern Overview
- Downloading Angular
- Choosing an IDE
- Creating a Simple Application with Angular

### Working with ES6 (ECMAScript)

- Enhancements to Legacy JavaScript
- Changes to `var`
- New `let` to Declare Variables
- Block Scoping
- Enhanced `for` Loops
  - `for-of`
  - `for-in`
- Literals and Strings
  - Extended Literal Support
  - Template Literals
  - Tag Functions
  - Object Literal Changes
- Function Enhancements
  - Default Parameters
  - Rest and Spread Operators
  - Function Overloading
  - Arrow Functions/Lambdas
- Changes to Arrays
- Array and Object Destructuring
- Advanced ES6 Features
  - Sets and Maps
  - WeakSets and WeakMaps
- Symbols
- Generators
- Iterators

### Angular Components

- Component LifeCycle

### Using Visual Studio Code

- Downloading and Installing Visual Studio Code
- Generating Angular Projects with the CLI (Angular Command Line Interface)
- Angular Project Structure, Files and Configuration
- Debugging Applications in VS Code and the Browser
- Using the Terminal Window

### Working with TypeScript

- Types
  - Working with Built-In Types
  - Custom Types
- Utilities
  - Using Fat Arrow Syntax
  - Template Strings
- Setting Up and Using Node.js
- Transpiling TypeScript into JavaScript
- TypeScript Compiler Configuration
- TypeScript Declaration Files
- Installing Typings Files

### Angular Modules

- Using Modules to Create an Application
- Default Modules
- Bootstrapping an Application
- Exporting Classes, Functions and Values
- Limiting Scope
- Grouping Modules
- Specifying Module Dependencies
- Organizing Code Files
- Module Testing
- Best Practices

### Component Templates and Data Binding

- Basic Data Binding Concepts

- Component Templates to Define Views
- Using Decorators to Define MetaData
- Styling
  - Per-Component Styling
  - Defining Global Styles in `angular.json`
  - Adding Bootstrap Framework to an Angular App
- Encapsulation
  - ShadowDOM Style Encapsulation
  - View Style Encapsulation
  - No Encapsulation
- ElementRef and Popups
- ExportAs
- Lifecycle Hooks
  - OnInit, OnDestroy, OnChanges, DoCheck
  - AfterContentInit, AfterViewInit
  - AfterContentChecked, AfterViewChecked
- Change Detection
- Passing Data to Components

## Angular Directives

- Built-In Directives
  - NgIf, NgFor, NgClass, NgStyle, NgSwitch, etc.
- Building Custom Directives
  - Using the Renderer2 Service
  - ElementRef
  - Attribute Directives
  - Structural Directives

## Defining and Consuming Services

- Dependency Injection
  - Registering Providers with the Injector
  - Changes to Injection in Angular 6
- Creating a Service
- Consuming a Service

- Interpolation
- One-Way Property Binding
- Two-Way Property Binding
- Event Binding
- Custom Binding
  - Exposing Properties and Events to Parent Controls
  - Custom Property Binding
  - Custom Event Binding

## Working with Pipes

- Text Casing Pipes
- Formatting Numbers and Dates
- Internationalization and Cultures
- Restricting Data Collections with `slice`
- Custom Pipes
  - Implementing the PipeTransform Interface
  - Pure vs Impure Pipes
  - Passing Parameters to Pipes

## Working with Web Services

- Using `HttpClient`
- Importing the HTTP Module
- Creating Requests
- Processing Responses
- Web API
- Using PostMan to Test the Server-Side Service
- Interacting with a RESTful Service
  - POST Requests
  - DELETE Requests

- PUT Requests
    - HEAD Requests
- Dealing with CORS (Cross Origin Resource Sharing)

## Asynchronous Programming in Angular    Angular Forms

- Reactive Programming Model
  - The RxJs Library
    - Observables
    - Observers
    - Subjects
    - Subscriptions
    - Operators
  - Using Operators
    - Creation Operators
    - Filtering of Data Operators
    - Conversion of Data Operators
    - Math and Aggregate Operators
    - Utility Operators
    - Pipeable Operators
  - Using the `async` Pipe
  - Changes to RxJs in Version 6
- The `ngNativeValidate` Directive
  - HTML `novalidate` Attribute
  - Template Driven Forms
    - `ngForm` and `ngModel` in Forms
    - Input and Output Properties
  - Reactive-Driven/Model-Driven Forms
    - `FormGroup`
    - `FormControl`
    - `FormArray`
    - `Validators` Class
  - Using the `FormBuilder` Factory
  - Working with Form State
  - Client-Side Forms Validation

## Angular Routing and Navigation

- Overview of Routing
- Implementing Single Page Applications (SPAs)
- Location Strategies
- Client-Side vs Server-Side Routing
- Working with the Component Router
- Adding Router Imports
- Performing Router Configuration
- Using Router State
- Redirects
- Routing Parameters
- Router Lifecycle Events
- Nesting Routes
- Routing Guards
  - Defining a Guard
  - Registering Guards
  - Securing Routes
  - `CanActivate` Guard
  - `CanActivateChild` Guard
  - `CanDeactivate` Guard
  - `CanLoad` Guard

## Unit Testing and TDD with Angular

- TDD vs End-to-End Testing
- Jasmine Testing Framework
  - Defining Expectations
- Running Tests in Karma
- Using the Angular Unit Test Framework
  - Fulfilling Dependencies
  - Mocking Out Data
  - Creating Testing Fixtures
- Testing Services and HTTP
- Using Test-Doubles (Mocks, Stubs and Spies)
- Testing Components
- Testing Forms

## The Angular Animation System

## Building and Deploying Applications

## The Web Animations API

- States and Transitions
- Entering and Leaving
- Animating Properties
- KeyFrames
- Parallel Animation Groups

## Developing a Deployment Strategy

- Managing Dependencies
- Tree Shaking
- Transpiling
- Linting

Software Skills Training, Inc.

6 Hemlock Drive

Chelmsford, MA 01824

978.250.4983

[www.software-skills-training.com](http://www.software-skills-training.com)

Copyright© 2018 Software Skills Training, Inc.