GEMRAIN CONSULTING

ANGULAR 10

Course Duration: 3 days; Instructor-led

WHAT YOU WILL LEARN

The Angular 10 training course covers all major components, syntax, and tooling of the Angular web application framework. Angular 10 fulfil the expectations of modern developers who demand fast performance and responsiveness from their web applications. Participants will set up their working environment to have all the tools needed to start building Angular 10 components with minimum effort. The program starts with an introduction of JavaScript, NPM, and TypeScript. Participant will understand TypeScript, a powerful typed superset of JavaScript that compiles to JavaScript. The course dives into component-driven development with Angular components. The participants will also learn to create and use Angular 10 Directives and Pipes. Data binding (both 1 and 2 way) is discussed in the context of the new directive API. Finally, the core parts of Angular such as services, HTTP, routing, form validation are covered to develop applications based on the Model-View-Controller (MVC) architecture.

PREREQUISITES

Before attending this course, students should have general programming experience and knowledge of Angular is compulsory.

METHODOLOGY

This program will be conducted with interactive lectures, PowerPoint presentation, discussion, and practical exercise.

COURSE OBJECTIVES

- Utilize the Angular core constructs to build Single Page Applications.
- In depth knowledge of Rxjs library.
- Angular modules – Feature module, Lazy loading module.
- Build Reactive forms.
- Utilize Material Design.
- Setup routing and URLs.
- Execute CRUD commands on REST APIs over Http.
- Security focus using JWT Authentication.

COURSE OUTLINES

Module 1: RxJS and Observable

- What is an Observable?
- Creating Observables
- What is an Observer?
- Observer Example
- Operators: map, switchMap, debounceTime, distinctUntilChanged
- Practical Application of using RxJS
- Subject
- Event Emitter or Observable

Module 2: Advanced HTTP Client

- Request Options
- Returning an Http Response Object
- Setting Request Headers
- Creating a Simple Observable
- The Observable.create() Method
- Observable Operators
- Piping Operators
- The flatMap() Operator
- The tap () Operator
- The zip () Operator
- Caching HTTP Response
- Making Sequential HTTP Calls
- Making Parallel Calls
- Customizing Error Object with catchError ()
- Error in Pipeline and Recovery

Module 3: Consuming REST Web Services

- REST Web Services and Angular
- Understanding REST
- REST Example – Create / Retrieve / Update / Delete / Client Generated ID / JSON
- Common Angular Tasks for REST Communication
- RESTFul Methods: DELETE / GET / PUT / POST
- Security of REST APIs

Module 4: Advanced Routing

- Routing Overvie
- Routing Enabled Project
- Routing Enabled Feature Module
- Using the Feature Module
- Lazy Loading the Feature Module
- Creating Links for the Feature Module Components

- More About Lazy Loading
- routerLinkActive binding
- Default Route
- Wildcard Route Path
- redirectTo
- Child Routes
- Defining Child Routes
- <router-outlet> for Child Routes
- Links for Child Routes
- Navigation Guards
- Creating Guard Implementations
- Using Guards in a Route
- Route Animations

Module 5: Angular Animations

- What is Animation?
- Animation Configuration
- Animation Techniques
- Animation Concepts
- CSS Property Animation
- Animation Property Settings
- CSS Transforms
- Starting and Stopping Animation
- Animation Events
- Browser Support
- Angular Animations
- Animation Imports
- Named Animation States
- Transitions
- The animate () function
- Triggers
- Assigning Animations to Elements using Trigger
- Invoking Transitions
- Assigning Animation to Routes
- External Animation Definitions

Module 6: Securing the Application using Token Based Auth

- Authentication / Authorization Basics in SPAs
- Token based Auth: Understanding JWT, OAuth 2.0 & OpenID Connect
- Using Social / Cloud based Logins
- Securing Angular Routes
- Securing the Web API

Module 7: Dependency Injection

- Injecting providers
- A new dependency injection system
- @Inject ()
- @Injectable ()
- Tokens
- Registering a provider
- Overriding providers
- Understanding Injectors
- Module imports
- Dependency Injection

Module 8: Angular Store / State management

- Overview of NgRx
- Creating an NgRx Application
- Overview of NgRx/Store
- Model, Action, Reducer, and Application State
- Redux pattern
- Reading, Writing and Removing data in NgRx Store
- Overview of NgRx / Effects
- Generating an Effect file
- Creating an Effect
- Actions Observable
- Dispatching Router-store Actions
- Overview of NgRx / Entity
- Entity State definition
- Entity Adapter
- Entity Selectors
- Custom IDs and State properties
- Overview of NgRx / Schematics
- Scaffolding NgRxapplication with Schematics

Module 9: WebSockets Data in Angular

- Overview
- Web Sockets Use Cases
- Web Socket URLs
- Web Sockets Servers
- Web Socket Client
- The socket.io-client library
- Using socket.io-client in JavaScript
- Setting up socket.io-client in Angular Projects
- Using socket.io-client in an Angular service
- Angular websocket.service Notes:
- The Angular Web Socket Client Sample App
- Angular websocket.component.ts
- The Full websocket.component.ts code
- Implementation Modifications

Module 10: Introduction to Testing Angular Applications

- Unit Testing
- Testing Tools
- Testing Setup
- Typical Testing Steps
- Test Results
- Jasmine Test Suites
- Jasmine Specs (Unit Tests)
- Expectations (Assertions)
- Matchers
- Examples of Using Matchers
- Using the not Property
- Setup and Teardown in Unit Test Suites
- Example of beforeEach and afterEach Functions
- Angular Test Module
- Example Angular Test Module
- Testing a Service
- Injecting a Service Instance
- Test a Synchronous Method
- Test an Asynchronous Method
- Using Mock HTTP Client
- Supplying Canned Response
- Testing a Component
- Component Test Module
- Creating a Component Instance
- The ComponentFixture Class
- Basic Component Tests
- The DebugElement Class
- Simulating User Interaction

Module 11: Debugging

- Debugging Overview
- Basic Debugging Practices
- Development (Debug) Mode
- Selecting Elements to Inspect
- Inspecting Angular Components with ng.probe
- Saving ng.probe Component References
- Modifying Values using Component References
- Using Breakpoints in Angular Code
- Breakpoint in TypeScript Code
- What is Augury?
- Installing Augury
- Opening Augury
- Augury - Component Tree
- Augury - Router Tree.
- Augury - NgModules Tab
- Common Exceptions
- Summary