ANGULAR 17 TRAINING

ANGULAR TRAINING OBJECTIVES

All students will:

- Understand how single-page web application architectures are different than traditional web application architectures
- Use new JavaScript (ES6) language features including Classes, Modules, and Arrow Functions
- Use new TypeScript language features including Types, Decorators, Interfaces, and Generics
- Learn Angular coding and architecture best practices including project layout and using container and presentation components
- Understand and use Angular model-driven forms, observables, dependency injection, and routing
- Communicate with a backend server using Angular's HttpClient to load and save data
- Configure the router and navigate between components
- Unit test all parts of an application including Components, Services, and Pipes
- Understand RxJS and Observables and where they can be used
- Implement Authentication and Authorization in an Angular Application
- Optimize Angular Performance by changing Change Detection Strategies
- Setup new projects from scratch using the Angular CLI
- Scaffold modules, components, services, models, routes, and unit tests in accordance with best practices using the Angular CLI
- Build and deploy an application to production using the Angular CLI
- Write End-to-End Tests (optional; taught only if this applies to your group)

ANGULAR TRAINING OUTLINE

DAY-1

- Components
  - Understanding Components
  - Component Properties & Methods
  - Templates: Inline, Multi-line, and External with Component-relative Paths
- Angular Modules (NgModule)
  - Angular Modules vs. ES Modules
  - Organizing your code into Feature Modules
- CSS Flexbox
- Grid and Responsive Design
- ngx-bootstrap

- Dependency Injection
  - Understanding Dependency Injection
  - Angular's Dependency Injection System
  - Registering
  - Injecting
  - Change detection strategy
- Model-driven Forms (Reactive Forms)
  - Importing the ReactiveFormsModule
  - FormControl, FormGroup, and AbstractControl
  - Binding DOM Elements to FormGroups and FormControls
  - Validation Rules, Messages, and Styles
  - Refactoring Reactive Forms for Reuse
  - Custom Validators
- Communicating with the Server using the HttpClient Service
  - Deciding between Promises or Observables (RxJS)
  - Making an HTTP GET Request
  - Sending data to the server using Http POST and PUT Requests
  - Issuing an Http DELETE Request
  - Intercepting Requests and Responses
- Router
  - Importing the RouterModule
  - Configuring Routes
  - Displaying Components using a RouterOutlet
  - Navigating declaratively with RouterLink

- Navigating with code using the Router
- Accessing parameters using ActivatedRoute

Day-2

- Deploying an Angular Application to Production
  - Building the application using the Angular CLI
  - Deploying to a web server
  - Angular Material UI
- Angular Roadmap for the Future
  - Ivy Renderer
  - Angular Elements
- Unit Testing
  - Tools: Jest
  - Syntax: describe, it, beforeEach, afterEach, matchers
  - Setup and your First Test
  - Testing Terminology: Mock, Stub, Spy, Fakes
  - Angular Testing Terminology: TestBed, ComponentFixture, debugElement, async, fakeAsync, tick, inject
  - Simple Component Test
  - Testing a Pipe
- RxJS and Observables
  - What is an Observable?
  - Creating Observables
  - What is an Observer?
  - Observer Example
  - Operators: map, switchMap, debounceTime, distinctUntilChanged
  - Practical Application of using RxJS
  - Subject
  - Subject Example
  - EventEmitter or Observable
- Working with NGRX

- Security
  - Best Practices
  - Preventing Cross-site Scripting (XSS)
  - Trusting values with the DOMSanitizer
  - HTTP Attacks (CSRF and CSSI)
  - Authentication using JSON Web Tokens (JWT)
  - Authorization: Router Guards
- Change Detection
  - Understanding Zone.js and Change Detection
  - Change Detection Strategies Default and OnPush

- Advanced Routing
    - Lazy-loading Angular Modules
    - Nested or Child Routes
- Advanced Dependency Injection
    - Providers
    - Hierarchical Injection
- Pipes
    - Creating a custom Pipe using PipeTransform
    - Understanding Pure and Impure Pipes

Day-3

Plan micro frontends with strategic design

- Strategic Design Overview
- Sub-domains, bounded context and ubiquitous language in the context of Angular
- Communication between domains
- Monorepos vs. Multi Repos
- Macro vs. micro architecture
- Micro frontends as migration strategy
- Case Studies

Micro Frontends with Module Federation

- Federated Angular: Using Module Federation with Angular
- Dynamic Federation
- Micro frontends and plug-in systems
- Sharing libraries
- Dealing with version conflicts
- Communication between micro frontends

Web Components with Angular Elements

- Micro Frontends Angular Elements
- Widgets with Angular Elements
- Load Web Components dynamically
- Multi-framework/multi-version solutions with Web Components
- Integrate legacy code (e.g. AngularJS code)
- Combining Web Components and Module Federation
- Challenges and workarounds

- Conclusion