6 Months Training TR-104 Full Stack Development

January, 2026

Day 1 Report

The first day of the training focused on setting up the development environment and introducing the fundamentals of Angular, a modern front-end framework used for building dynamic, single-page web applications. The session emphasized understanding Angular from scratch using the latest version, following official documentation and best development practices. Practical hands-on exercises were carried out to ensure conceptual clarity and real-world applicability.

Introduction to Angular

Angular is a TypeScript-based front-end framework developed by Google for building scalable and maintainable web applications. It follows a component-based architecture, allowing developers to structure applications into reusable and independent units.

Unlike traditional web development where DOM manipulation is handled manually, Angular manages the UI dynamically by binding data and logic through a structured framework.

Environment Setup

The session began with configuring the system to support Angular development:

- Installation and verification of **Node.js** and **npm**
- Installation of **Angular CLI** using npm
- Creation of an Angular project using **official Angular documentation**
- Understanding the role of the development server (ng serve)
- Use of **Visual Studio Code** and its integrated terminal for development

This ensured a stable and professional development environment aligned with industry standards.

Modern Angular Project Structure (Angular 17+)

Participants explored the **latest Angular project structure**, which differs from older versions by using **standalone components** instead of traditional NgModules.

Key files studied included:

- app.ts – component logic
- app.html – template (UI)
- app.css – component styling
- app.routes.ts – routing configuration

This approach reflects modern Angular best practices and simplifies application architecture.

Core Angular Concepts Covered

1. Components

A component is the fundamental building block of an Angular application. Each component consists of:

- A **selector** (custom HTML tag)
- A **template** (HTML)
- **Styles** (CSS)
- **Logic** (TypeScript class)

The role of the '@Component' decorator in connecting these elements was explained in detail.

2. Selector

The selector defines how and where a component appears in HTML. It acts as a custom HTML element that Angular replaces with the component’s rendered content.

3. Interpolation

Interpolation is used to display component data inside the template using double curly braces {{ }}. It enables one-way data binding from the TypeScript logic to the HTML view.

Introduction to Signals (Modern Angular State Management)

Angular’s modern signal-based reactivity system was introduced. Signals are reactive data containers that Angular monitors for changes.

Key concepts:

- Signals store application state
- Data is read using signal()
- Data is updated using .set()
- Direct assignment to signals is not allowed

This mechanism ensures automatic UI updates without manual DOM manipulation.

Event Binding and User Interaction

The session introduced Angular’s event-binding mechanism to handle user actions:

- (click) event for button interactions
- (input) event for capturing user input
- Use of $event to access browser event objects
- Type casting using HTMLInputElement for safe data access

Participants learned how user actions trigger methods in the component class, which then update signals and automatically refresh the UI.

During the session, multiple practical tasks were performed to gain hands-on experience with modern Angular development concepts and workflows. These tasks focused on understanding component behavior, data binding, state management using signals, and handling user interactions.

Hands-On Tasks Implemented

- Created and successfully ran an Angular project using the Angular CLI, following official documentation guidelines.
- Explored the modern Angular standalone component structure, including app.ts, app.html, and app.css.
- Simplified the default Angular template to understand the core rendering mechanism.
- Implemented interpolation to display dynamic data from the component logic into the template.
- Introduced signals to manage reactive application state.
- Demonstrated correct signal updates using the .set() method instead of direct variable assignment.
- Implemented a counter example to understand signal-based state updates triggered by user actions.
- Created an input field to capture real-time user input using the (input) event.
- Utilized the $event object and type casting (HTMLInputElement) to safely extract user-entered values.
- Updated displayed content dynamically as the user typed, reinforcing one-way data flow from state to UI.

- Implemented a reset button using the (click) event to modify signal values through application logic.
- Demonstrated that a single signal can be updated from multiple sources, such as user input and button clicks.
- Observed Angular's automatic UI updates without manual DOM manipulation.

These tasks reinforced the core Angular principle that the user interface is a reflection of application state, and that all UI changes should be driven through structured data updates rather than direct manipulation of HTML elements.

Key Learnings

- Angular follows a structured, component-based architecture
- Modern Angular uses standalone components and signals
- Interpolation is used only for displaying data
- Signals provide reactive and predictable state management
- .set() is mandatory for updating signal values
- Event binding connects user interaction with application logic
- Angular automatically updates the UI when data changes

Conclusion

The first day of training successfully established a strong foundation in Angular development. Participants gained hands-on experience with setting up the development environment, understanding modern Angular architecture, working with components, selectors, interpolation, and signals, and handling user interactions through event binding. By following official documentation and industry best practices, the session prepared participants to build dynamic, scalable, and maintainable front-end applications. This foundation sets the stage for advanced topics such as routing, services, forms, and full-stack integration in upcoming sessions.