

6 Months Training TR-104 Full Stack Development

January, 2026

Day 3 Report

The third day of the training focused on completing the remaining core Angular fundamentals required to confidently start working on mini tasks and small projects. The session emphasized mastering modern Angular template features, understanding the difference between UI-based rendering and application logic, and learning best practices for writing clean, maintainable Angular code using the latest framework syntax.

Modern Angular Template Control Flow

The session began with an in-depth exploration of Angular's modern template control flow syntax, introduced in recent Angular versions.

1. Conditional Rendering (@if, @else)

Participants learned how to conditionally display UI elements using @if and @else. The distinction between expressions and statements was clarified to avoid common mistakes.

Key points:

- @if expects a boolean expression, not an if statement
- Conditions control UI rendering, not program logic
- Angular automatically re-renders the UI when the underlying state changes

This reinforced the understanding that template control flow is declarative and reactive in nature.

2. Looping and Dynamic UI Rendering (@for)

The concept of looping over data to generate repeated UI elements was covered using the modern @for syntax.

Key learnings:

- @for is used for UI repetition, not logical iteration
- Lists are rendered based on array data stored in signals
- The track keyword is required for performance optimization
- UI updates automatically when the underlying array changes

Participants implemented dynamic lists and practiced adding and removing items by updating signal-based arrays immutably.

UI-Based vs Logic-Based Constructs

A clear distinction was established between:

- Logic-based constructs (TypeScript if, for)
- UI-based constructs (@if, @for)

This helped solidify the concept that Angular templates describe what should be shown, while TypeScript controls how the application behaves.

Conditional Styling

Participants learned how to dynamically apply CSS classes based on application state using property binding. This demonstrated how visual changes in the UI can react directly to state changes without manual DOM manipulation.

Key takeaway:

- CSS should react to state, not user actions directly

Computed Signals

The session introduced **computed signals**, a modern Angular feature used to derive values from existing signals.

Key points:

- Computed signals do not store data
- They automatically update when dependent signals change
- They help keep templates clean by moving logic into TypeScript
- Computed signals are read-only and cannot be set directly

This encouraged cleaner separation between logic and presentation.

Working with Objects in Templates

Participants practiced handling object-based data using signals. This included accessing nested properties and using object values within template expressions and conditional rendering.

Key concepts:

- Signals holding objects must be accessed using function calls
- Object properties can be safely used in templates
- Objects must be updated immutably to preserve reactivity

Routing (Conceptual Overview)

A conceptual overview of Angular routing was provided to explain how applications support multiple views without reloading the page.

Topics covered:

- Purpose of routing in single-page applications
- Role of router-outlet as a dynamic content placeholder
- How Angular swaps components based on the URL
- Separation of layout components (header/footer) from routed content

This laid the groundwork for future routing implementation.

Hands-On Tasks Implemented

- Implemented conditional UI rendering using @if and @else
- Rendered dynamic lists using @for with proper tracking
- Added and removed list items by updating signal-based arrays
- Applied conditional CSS classes based on application state
- Created computed signals to derive UI-ready values
- Accessed and displayed object-based data in templates
- Followed immutability principles for state updates
- Understood the architectural role of routing and router-outlet

These tasks completed the core Angular foundation required for independent practice.

Key Learnings

- Angular templates use expressions, not statements

- UI logic and application logic are strictly separated
- Signals drive UI reactivity in modern Angular
- Computed signals simplify templates and improve readability
- Objects and arrays must be updated immutably
- Angular's rendering system reacts automatically to state changes
- Routing controls which component is displayed, not page reloads

Conclusion

The third day of training successfully completed the foundational Angular concepts necessary to begin hands-on development with confidence. Participants gained a clear understanding of modern Angular template syntax, reactive state management using signals, computed values, and the architectural role of routing. With these fundamentals in place, participants are now well-prepared to start implementing mini tasks and small projects independently, forming a strong base for advanced Angular topics and real-world application development.