6 Months Training – TR-104 Full Stack Development

January 2026
Day 2 – Core Angular Concepts & Data Binding

Objective of the Day

The objective of Day 2 was to understand the core working principles of Angular, focusing on how data flows between the component logic and the user interface. The session aimed to build clarity on templates, data binding, event handling, and control flow, forming the backbone of Angular application development.

Detailed Work Description

Understanding Angular Components in Depth

The session began with a deeper exploration of Angular components, which were introduced on Day 1. A component was explained as a combination of:

- **Template (HTML)** – Defines the UI
- **Component Class (TypeScript)** – Contains data and logic
- **Styles (CSS)** – Defines appearance
- **@Component Decorator** – Connects metadata with logic

The importance of keeping logic separate from presentation was emphasized to maintain clean and maintainable code.

Role of the @Component Decorator

The @Component decorator was studied in detail. Its responsibilities include:

- Declaring the component selector
- Linking the template file
- Associating styles with the component
- Defining whether the component is standalone

This decorator enables Angular to recognize a class as a component and manage its lifecycle.

Selectors and Component Rendering

Selectors were explained as **custom HTML tags** that represent Angular components. When Angular encounters a selector in HTML, it replaces it with the rendered component view.

Example concept:

- <app-root> acts as the entry point of the Angular application.

This demonstrated how Angular dynamically constructs the UI.

Interpolation ({{ }})

Interpolation was introduced as a **one-way data binding technique** used to display component data inside the template.

Key points:

- Interpolation flows from TypeScript → HTML
- It is read-only
- Used only for displaying values

This reinforced the idea that Angular controls how data appears in the UI.

Event Binding and User Interaction

Event binding was a major focus of Day 2. Angular uses parentheses () to bind events.

Examples discussed:

- (click) – triggers logic on button click
- (input) – captures user input in real time

The concept of **$event** was introduced:

- $event represents the browser event object
- It carries information such as user input or click details

Type casting using HTMLInputElement was demonstrated to safely extract values from input fields.

Template Control Flow

Angular's template control flow was discussed to conditionally display or repeat elements.

Key directives explained:

- *ngIf – conditionally renders elements
- *ngFor – loops through lists and displays data

The difference between:

- JavaScript if
- Angular template *ngIf

was clearly explained:

- JavaScript if works in logic
- *ngIf controls DOM rendering

Understanding Data Flow in Angular

A crucial concept explained was that Angular follows unidirectional data flow:

- Data flows from component to template
- UI updates automatically when data changes

This removed the need for manual DOM manipulation and ensured predictable UI behavior.

Introduction to User Input Handling

The session demonstrated how user input is captured and processed in Angular:

- Input fields trigger (input) events
- Component variables store user-entered data
- UI reflects updated values automatically

This helped understand how Angular connects user interaction with application logic.

Hands-On Tasks Implemented

The following practical tasks were performed during the session:

- Created component variables to store data
- Displayed dynamic values using interpolation
- Implemented button click handling using (click)
- Captured input field values using (input)
- Used $event to extract input data

- Controlled UI rendering using *ngIf
- Displayed lists dynamically using *ngFor
- Observed real-time UI updates without page reload

These exercises strengthened the understanding of Angular's reactive behavior.

Key Learnings

- Angular components control both logic and UI
- @Component decorator defines component metadata
- Interpolation enables one-way data binding
- Event binding connects user actions to logic
- $event carries browser event data
- *ngIf and *ngFor control template rendering
- Angular automatically updates the UI on data changes
- Manual DOM manipulation is not required

Common Mistakes Identified and Clarified

- Confusing JavaScript if with Angular *ngIf
- Trying to modify DOM manually instead of updating data
- Misunderstanding $event as a custom variable
- Expecting two-way binding by default

These mistakes were clarified through examples and practice.

Conclusion

Day 2 successfully strengthened the understanding of Angular's core concepts, especially data binding and event handling. By combining theory with hands-on practice, the session ensured clarity on how Angular manages UI updates through data changes. This knowledge laid the groundwork for advanced topics such as signals, services, and reactive forms, which are essential for real-world Angular applications.