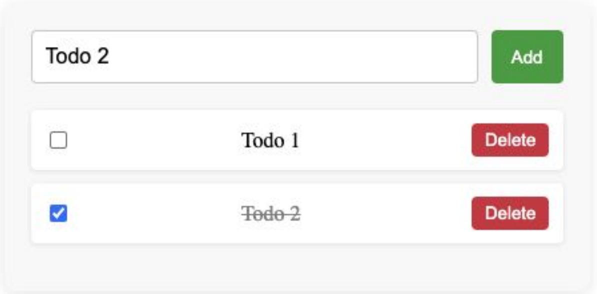
Angular Demo

TA - Weijun Li

Learning Objectives



🎯 Create an Angular project and understand its basic structure.

🎯 Use Data Binding to synchronize the UI with the application state.

🎯 Use Event Binding to respond to user interactions.

🎯 Understand how Angular automatically updates the UI when todoList changes.

https://github.com/mvsovo/INF133Angular

Creating Angular project

- Install Angular globally
  - `npm install -g @angular/cli`

- Create a new Angular project
  - `ng new angular-todo`
  - `cd angular-todo`
  - `ng serve`

Creating a New Component

Angular Project Structure

app.component.ts	Main component (entry point of the app)
app.component.html	HTML template for the main component
app.component.css	Styles for the main component
app.routes.ts	Defines routing (maps URLs to components)
app.config.ts	Application-wide configuration settings
index.html	Main HTML entry point (renders <app-root>)
main.ts	Bootstraps Angular and initializes the app

Three Main Types of Data Binding in Angular

Interpolation	Display variable values in the UI	{{ componentVariable }}
Property Binding	Bind values to HTML properties	<element [attribute]="value">
Event Binding	Respond to user interactions	<element (event)="function()">
Two-Way Binding	Sync data between UI and component	<input [(ngModel)]="value">

To do List Structure

We need a data structure to store the tasks:

- **todoList** → Stores an array of tasks, each task has:
  - id → Unique identifier
  - text → Task description
  - completed → Whether the task is finished
- **nextId** → Keeps track of task IDs to ensure uniqueness.

```
export class AppComponent {
 todoList: { id: number; text: string; completed: boolean }[] = [];
 nextId = 1;
}
```

Interpolation - Displaying Data in the UI

Interpolation allows us to display component data inside the HTML template using {{ }}.

```
       <span>{{ todoItem.text }}</span>
```

- todoItem.text **comes from todoList in app.component.ts**. -> The value inside [ ] **must be a variable from app.component.ts**.
- todoItem.text **is inserted into {{ }} inside the <span> tag**.
- When todoItem.text changes in todoList, **the UI updates automatically**.

```
todoList: { id: number; text: string; completed: boolean }[] = [];
// If todoList = [{ id: 1, text: "Buy groceries", completed: false }]
//The UI displays:
Buy groceries
```

Property Binding - Binding Data to Attributes

Property binding allows Angular to dynamically set HTML element properties based on component data.

```html
        <input type="checkbox"  (change)="toggleCompleted(todoItem.id)"  [checked]="todoItem.completed">
```

[checked]="todoItem.completed" binds the checkbox state to completed.

If completed = true, the checkbox is **checked**.

If completed = false, the checkbox is **unchecked**.

Event Binding - Responding to User Interactions

Event binding allows Angular to listen for user interactions (such as clicks) and call a function in the component.

User clicks the checkbox → (change) event triggers.
Calls toggleCompleted(todoItem.id).
Function finds the task and toggles completed (true ↔ **false)**.
Angular automatically updates the UI

```
  <input type="checkbox" (change)="toggleCompleted(todoItem.id)" >


// .ts
toggleCompleted(taskId: number) {
   const task = this.todoList.find(t => t.id === taskId);
   if (task) {
     task.completed = !task.completed;
   }
```

Activity

Modify the delete button so that its **text dynamically changes** based on whether the task is completed.

1. **Find the delete button in app.component.html.**
2. **Modify the button to show different text for completed vs. incomplete tasks.**
3. **Use @if inside the button to dynamically change its text.**
4. **Ensure Angular automatically updates the text when todoItem.completed changes.**

Activity

Modify the delete button so that its **text dynamically changes** based on whether the task is completed.

```
<button (click)="deleteTask(todoItem.id)">
         @if (todoItem.completed) {
           <span>Remove completed task</span>
         } @else {
           <span>Remove task</span>
         }
   </button>
```

Activity

Display the total number of tasks in the To-Do List using **data binding**