**Angular Learning - Week 3 (4 AUG - 10 AUG)**

### 📅 Topics Covered in Week 3 (Phase 3)

* Angular Forms Overview
* Template-Driven Forms
* Reactive Forms
* Form Validation (Built-in & Custom)
* Form Events & States

## 1. Angular Forms Overview

Angular offers two powerful approaches:

* **Template-Driven Forms** (easy, less code, form logic in HTML)
* **Reactive Forms** (more control, logic in TypeScript, scalable)

Import required module:

import { FormsModule, ReactiveFormsModule } from '@angular/forms';  
@NgModule({ imports: [FormsModule, ReactiveFormsModule] })

## 2. Template-Driven Forms

Form is defined in the template using [(ngModel)].

### ✏️ Example:

<form #userForm="ngForm" (ngSubmit)="onSubmit(userForm)">  
 <input name="name" [(ngModel)]="user.name" required />  
 <input name="email" [(ngModel)]="user.email" email required />  
 <button type="submit" [disabled]="!userForm.valid">Submit</button>  
</form>

user = { name: '', email: '' };  
onSubmit(form: NgForm) {  
 console.log(form.value);  
}

## 3. Reactive Forms

Form structure is defined in the component class using FormGroup & FormControl.

### ✏️ Example:

import { FormGroup, FormControl, Validators } from '@angular/forms';  
  
userForm = new FormGroup({  
 name: new FormControl('', Validators.required),  
 email: new FormControl('', [Validators.required, Validators.email])  
});  
  
onSubmit() {  
 console.log(this.userForm.value);  
}

<form [formGroup]="userForm" (ngSubmit)="onSubmit()">  
 <input formControlName="name" />  
 <input formControlName="email" />  
 <button type="submit" [disabled]="!userForm.valid">Submit</button>  
</form>

## 4. Validation Techniques

### ✅ Built-in Validators:

* required, minlength, maxlength, email, pattern

### ✅ Custom Validator:

function forbiddenNameValidator(control: AbstractControl): ValidationErrors | null {  
 return control.value === 'admin' ? { forbiddenName: true } : null;  
}

name: new FormControl('', [Validators.required, forbiddenNameValidator])

## 5. Form States & Status

this.userForm.valid; // true/false  
this.userForm.invalid;  
this.userForm.pristine;  
this.userForm.dirty;  
this.userForm.touched;

<p \*ngIf="userForm.controls.email.errors?.required && userForm.controls.email.touched">  
 Email is required.  
</p>

## 💻 Week 3 Project: User Registration Form

### Goal:

* Collect user name, email, and password
* Validate form fields
* Show error messages if invalid

### Reactive Form Code:

registrationForm = new FormGroup({  
 name: new FormControl('', Validators.required),  
 email: new FormControl('', [Validators.required, Validators.email]),  
 password: new FormControl('', [Validators.required, Validators.minLength(6)])  
});

<form [formGroup]="registrationForm" (ngSubmit)="submitForm()">  
 <input formControlName="name" placeholder="Name" />  
 <input formControlName="email" placeholder="Email" />  
 <input formControlName="password" type="password" placeholder="Password" />  
 <button type="submit">Register</button>  
</form>

### Result:

* ✅ Shows error messages
* ✅ Disables button if invalid
* ✅ Logs user data on submit