**📘 Template-driven Forms in Angular**

**✅ What are Template-driven Forms?**

Template-driven forms in Angular are forms where the logic and validations are primarily written in the **HTML template**. Angular uses **FormsModule** to manage and bind form input values using **two-way data binding** with [(ngModel)].

They are simple, easy to use, and best suited for **basic to moderately complex forms**.

**🧠 Theory**

**🔹 Key Features**

* Uses FormsModule
* Based on **directives** such as ngModel, ngForm, required, minlength, etc.
* Automatic form tracking with NgForm
* Uses **two-way binding** via [(ngModel)]
* Less scalable than Reactive Forms, but easier to set up

**🔸 Important Directives**

| **Directive** | **Purpose** |
| --- | --- |
| ngModel | Binds input field with component property |
| required, minlength, maxlength | Built-in validators |
| #formRef="ngForm" | References the form object |
| formControlName | (Used in Reactive Forms only) |
| [(ngModel)] | Two-way binding |
| ngForm | Automatically added to <form> tag |

**🛠️ Coding Example**

**📂 Step 1: Import FormsModule**

In app.module.ts:

ts

CopyEdit

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';

@NgModule({

declarations: [AppComponent],

imports: [BrowserModule, FormsModule],

bootstrap: [AppComponent]

})

export class AppModule { }

**📜 Step 2: Template (HTML) — app.component.html**

html

CopyEdit

<h2>Template-driven Form Example</h2>

<form #userForm="ngForm" (ngSubmit)="onSubmit(userForm)">

<div>

<label>Name:</label>

<input type="text" name="name" required [(ngModel)]="user.name" #name="ngModel">

<div \*ngIf="name.invalid && name.touched" style="color:red">

Name is required.

</div>

</div>

<div>

<label>Email:</label>

<input type="email" name="email" required [(ngModel)]="user.email" #email="ngModel">

<div \*ngIf="email.invalid && email.touched" style="color:red">

Valid email is required.

</div>

</div>

<div>

<label>Gender:</label>

<select name="gender" [(ngModel)]="user.gender">

<option value="male">Male</option>

<option value="female">Female</option>

</select>

</div>

<button type="submit" [disabled]="userForm.invalid">Submit</button>

</form>

**📘 Step 3: Component Code — app.component.ts**

ts

CopyEdit

import { Component } from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

})

export class AppComponent {

user = {

name: '',

email: '',

gender: 'male'

};

onSubmit(form: any) {

console.log('Form Submitted!', form.value);

}

}

**📌 Notes**

* The form gets automatically tracked by Angular using the ngForm directive.
* You can use form.valid, form.invalid, form.touched, etc., for conditions.
* You can access form controls using template reference variables like #name="ngModel".

**📚 Built-in Validators**

| **Validator** | **Usage** |
| --- | --- |
| required | Makes the field mandatory |
| minlength="3" | Minimum character limit |
| maxlength="10" | Maximum character limit |
| pattern="[a-zA-Z ]\*" | Regex pattern matching |
| email | Checks for valid email format |

Example:

html

CopyEdit

<input type="text" name="username" required minlength="3" [(ngModel)]="user.username">

**🧪 Advantages**

* Simple and declarative syntax
* Ideal for small to medium forms
* Quick setup with minimal code

**🚫 Disadvantages**

* Less scalable
* Less control than Reactive Forms
* Difficult to test

Q