Forms

Mohamed Mukthar Ahmed

208

Forms

Mohamed Mukthar Ahmed

209

Forms are Crucial

- The most **crucial** aspect of your web application
- Handling user input with forms is the **cornerstone** of many common applications.
- Applications use forms
  - to enable users to log in,
  - to update a profile,
  - to enter sensitive information, and
  - to perform many other data-entry tasks.
- Angular provides two different approaches to handling user input through forms:
  - **Template-driven**
  - **Reactive**

Mohamed Mukthar Ahmed

Forms are Crucial

- Thankfully, **Angular** has tools to build forms
  - **Form** foundation **classes** encapsulate the inputs in our forms and give us objects to work with them
  - **Validators** give us the ability to validate inputs, any way we'd like
  - **Observers** let us watch our form for changes and respond accordingly

Mohamed Mukthar Ahmed

Choosing an approach

- **Template-driven** forms and **Reactive** forms process and manage form data differently.
- Each approach offers different **advantages**.
- **Template-driven** forms
  - Rely on directives in the template to create and manipulate the underlying object model.
  - They are useful for adding a simple form to an app, such as an email list signup form.
  - They're straightforward to add to an app, but they don't scale as well as reactive forms.
  - If you have very basic form requirements and logic that can be managed solely in the template, template-driven forms could be a good fit.

Mohamed Mukthar Ahmed

212

Choosing an approach

- **Template-driven** forms and **Reactive** forms process and manage form data differently.
- Each approach offers different advantages.
- **Reactive** forms
  - Provide direct, explicit access to the underlying form's object model.
  - Compared to template-driven forms, they are more robust: they're more scalable, reusable, and testable.
  - If forms are a key part of your application, or you're already using reactive patterns for building your application, use reactive forms.

Mohamed Mukthar Ahmed

213

Key Differences

- The table summarizes the **key differences** between template-driven and reactive forms.

Key Difference		
Factors	**Template-Driven**	**Reactive**
Setup of form model	Implicit, created by directive	Explicit, created in component class
Data model	Unstructured and mutable	Structured and immutable
Data flow	Asynchronous	Synchronous
Form validation	Directives	Functions

Mohamed Mukthar Ahmed

Common Form foundation classes

- Both template-driven and reactive forms are built on the following **base classes**

Foundation Classes	
Class	**Details**
FormControl	Tracks the value and validation status of an individual form control
FormGroup	Tracks the same values and status for a collection of form controls
FormArray	Tracks the same values and status for an array of form controls
ControlValueAccessor	Creates a bridge between Angular FormControl instances and built-in DOM elements

Mohamed Mukthar Ahmed

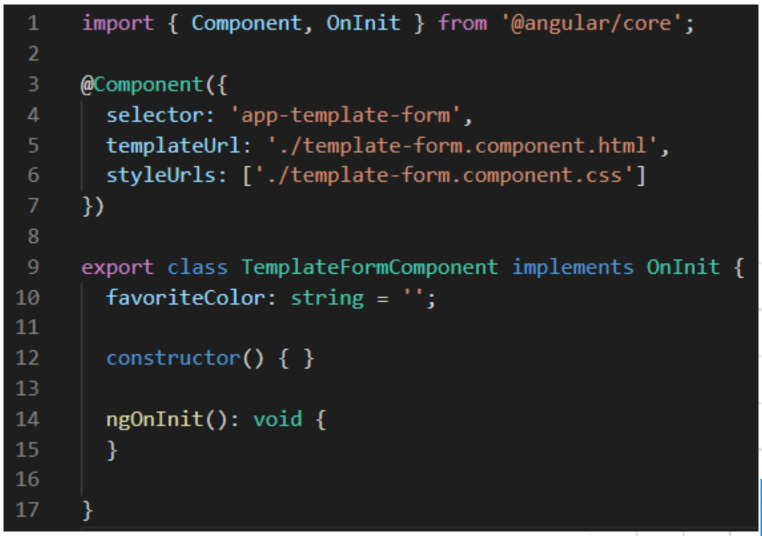
Setup - Template-driven Forms

- In template-driven forms, the form model is **implicit**, rather than explicit.
- The directive **NgModel** creates and manages a **FormControl** instance for a given form element.
- The following component implements the same input field for a single control, using template-driven forms.

Mohamed Mukthar Ahmed

216

Template-driven Form Component



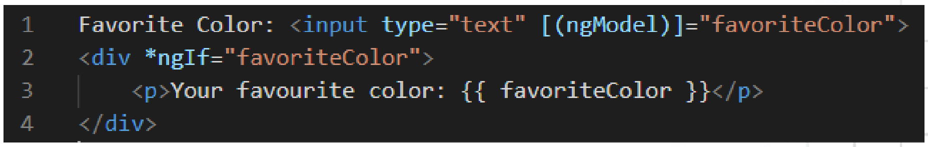
```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-template-form',
  templateUrl: './template-form.component.html',
  styleUrls: ['./template-form.component.css']
})

export class TemplateFormComponent implements OnInit {
  favoriteColor: string = '';

  constructor() { }

  ngOnInit(): void {
  }

}
```

Mohamed Mukthar Ahmed

217

Setup - Template-driven Forms

- In a template-driven form the source of truth is the template.
- You do not have direct programmatic access to the FormControl instance

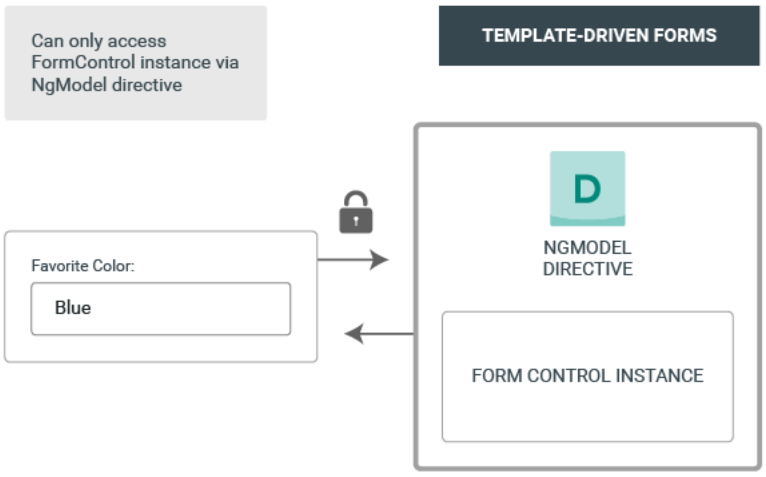


```
Favorite Color: <input type="text" [(ngModel)]="favoriteColor">
<div *ngIf="favoriteColor">
    <p>Your favourite color: {{ favoriteColor }}</p>
</div>
```

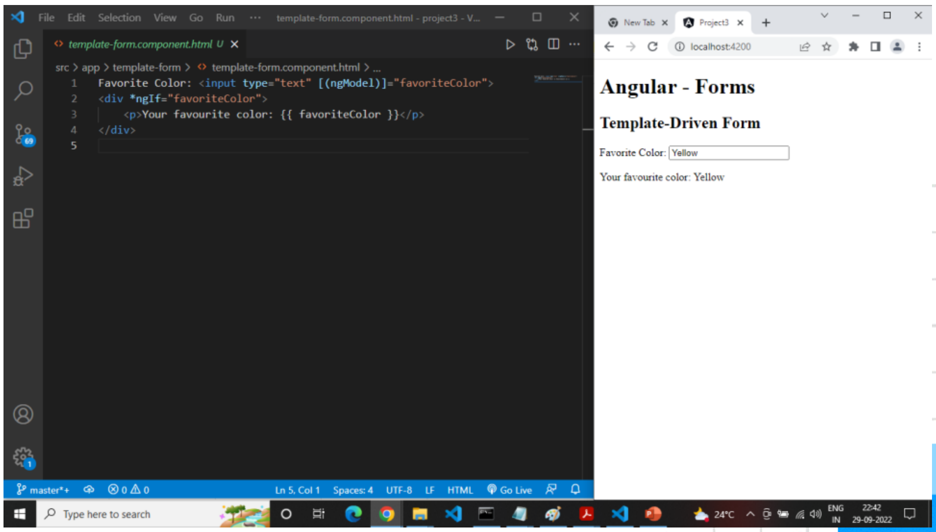
- Observe the use of **NgModel** directive.

Mohamed Mukthar Ahmed

Template-Driven Form



Mohamed Mukthar Ahmed



```
Favorite Color: <input type="text" [(ngModel)]="favoriteColor">
<div *ngIf="favoriteColor">
    <p>Your favourite color: {{ favoriteColor }}</p>
</div>
```

Angular - Forms

Template-Driven Form

Favorite Color: Yellow

Your favourite color: Yellow

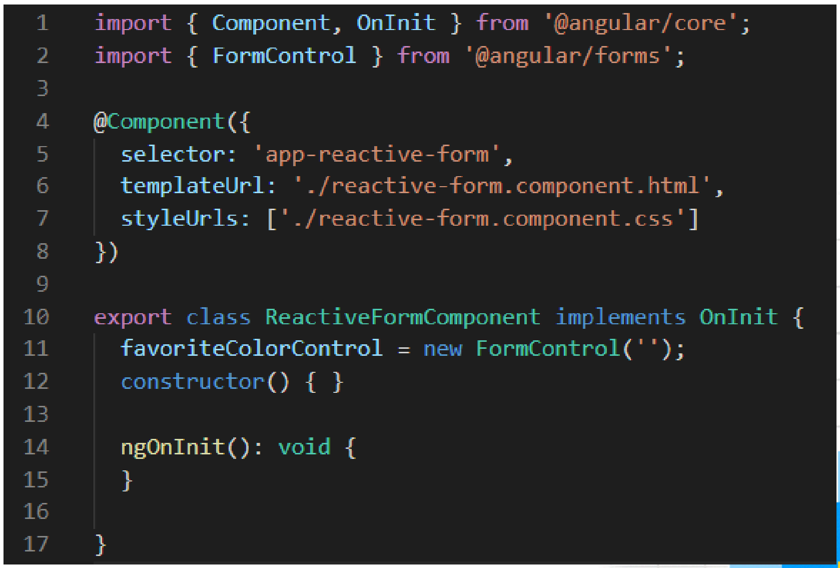
Mohamed Mukthar Ahmed

Setup - Reactive Forms

- With **reactive** forms, you define the form model directly in the component class.
- The **[formControl]** directive links the **explicitly** created **FormControl** instance to a specific form element in the view, using an internal value accessor.
- The following component implements an input field for a single control, using reactive forms.
- In this example, the form model is the **FormControl** instance.

Mohamed Mukthar Ahmed

Reactive Form Component

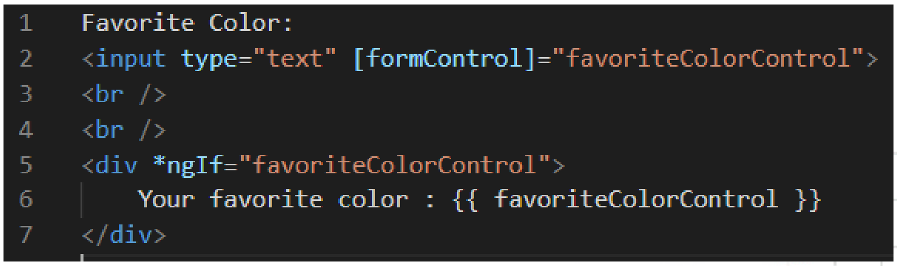


```
import { Component, OnInit } from '@angular/core';
import { FormControl } from '@angular/forms';

@Component({
  selector: 'app-reactive-form',
  templateUrl: './reactive-form.component.html',
  styleUrls: ['./reactive-form.component.css']
})

export class ReactiveFormComponent implements OnInit {
  favoriteColorControl = new FormControl('');
  constructor() { }

  ngOnInit(): void {
  }

}
```

Mohamed Mukthar Ahmed

222

Reactive Form Template



```
Favorite Color:
<input type="text" [formControl]="favoriteColorControl">
<br />
<br />
<div *ngIf="favoriteColorControl">
    Your favorite color : {{ favoriteColorControl }}
</div>
```

Mohamed Mukthar Ahmed

223

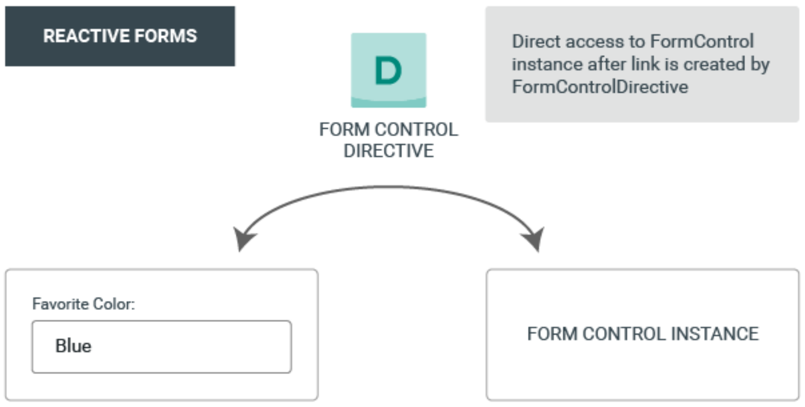
Setup - Reactive Forms

- In reactive forms, the form model is the source of truth.
- It provides the value and status of the form element at any given point in time, through the **[formControl]** directive on the input element.

Mohamed Mukthar Ahmed

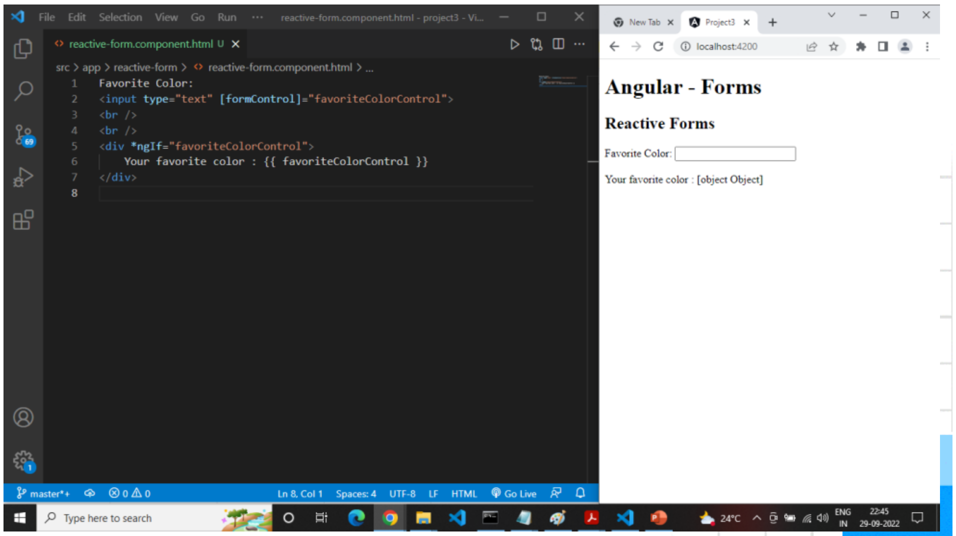
224

Setup - Reactive forms


REACTIVE FORMS
D
FORM CONTROL DIRECTIVE
Direct access to FormControl instance after link is created by FormControlDirective
Favorite Color:
Blue
FORM CONTROL INSTANCE

Mohamed Mukthar Ahmed

225



reactive-form.component.html

```
Favorite Color:
<input type="text" [formControl]="favoriteColorControl">
<br />
<br />
<div *ngIf="favoriteColorControl">
    Your favorite color : {{ favoriteColorControl }}
</div>
```

localhost:4200

Angular - Forms

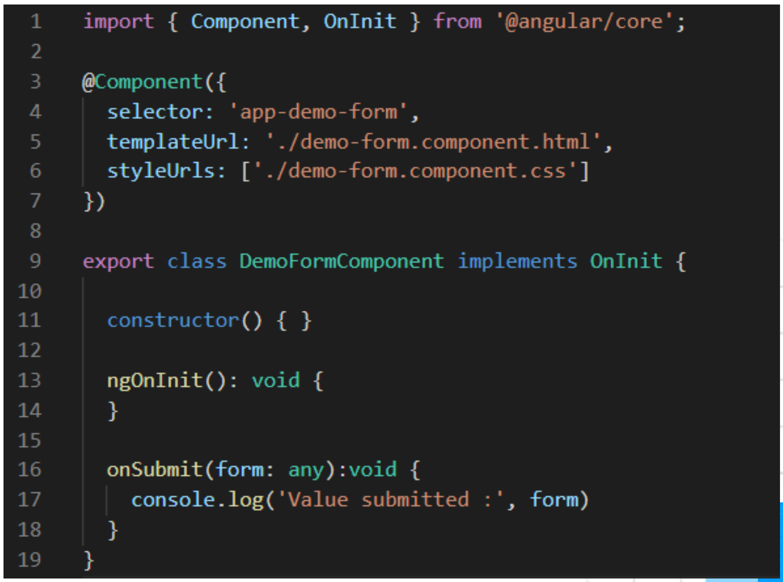
Reactive Forms

Favorite Color:

Your favorite color : [object Object]

226

The onSubmit Event

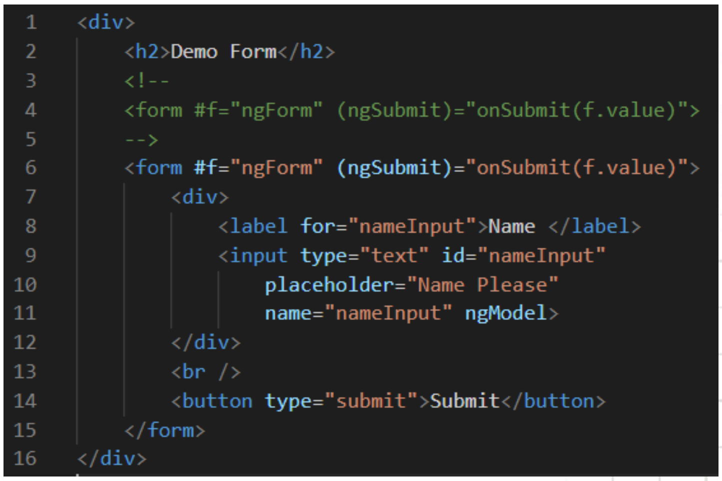


```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-demo-form',
  templateUrl: './demo-form.component.html',
  styleUrls: ['./demo-form.component.css']
})

export class DemoFormComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

  onSubmit(form: any):void {
    console.log('Value submitted :', form)
  }
}
```

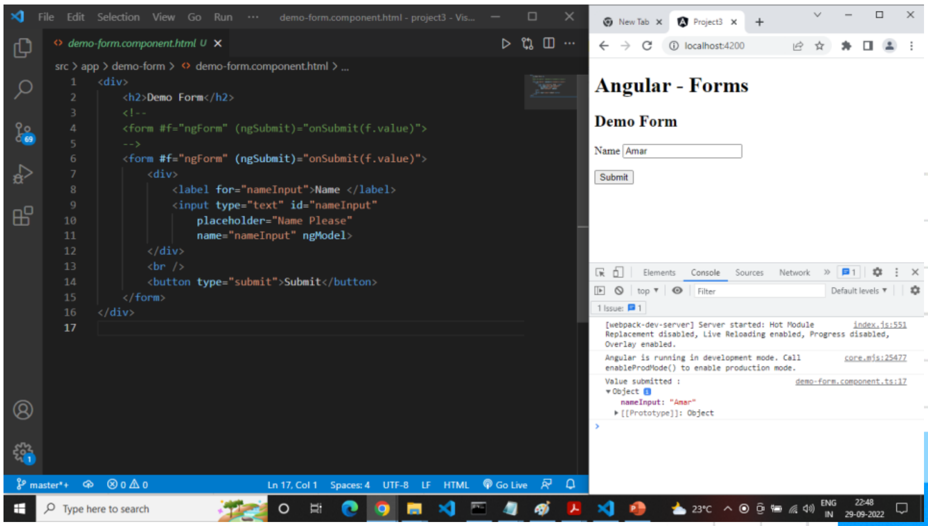
227

The onSubmit Event

- Template uses the **ngSubmit** directive



```html
<div>
    <h2>Demo Form</h2>
    <!--
    <form #f="ngForm" (ngSubmit)="onSubmit(f.value)">
    -->
    <form #f="ngForm" (ngSubmit)="onSubmit(f.value)">
        <div>
            <label for="nameInput">Name </label>
            <input type="text" id="nameInput"
                placeholder="Name Please"
                name="nameInput" ngModel>
        </div>
        <br />
        <button type="submit">Submit</button>
    </form>
</div>
```



```html
<div>
    <h2>Demo Form</h2>
    <!--
    <form #f="ngForm" (ngSubmit)="onSubmit(f.value)">
    -->
    <form #f="ngForm" (ngSubmit)="onSubmit(f.value)">
        <div>
            <label for="nameInput">Name </label>
            <input type="text" id="nameInput"
                placeholder="Name Please"
                name="nameInput" ngModel>
        </div>
        <br />
        <button type="submit">Submit</button>
    </form>
</div>
```

Angular - Forms

Demo Form

Name: Amar

Submit

```
[webpack-dev-server] Server started: Hot Module          index.js:551
Replacement disabled, Live Reloading enabled, Progress disabled,
Overlay enabled.
Angular is running in development mode. Call          core.mjs:25477
enableProdMode() to enable production mode.
Value submitted :                          demo-form.component.ts:17
▼Object
    nameInput: "Amar"
  ▶[[Prototype]]: Object
```

Form Validation

- **Validation** is an integral part of managing any set of forms.
- Whether you're checking for required fields or querying an external API for an existing username etc.
- Angular provides a set of built-in validators as well as the ability to create custom validators.

Mohamed Mukthar Ahmed

230

Form Validation

- Angular provides a set of built-in validators as well as the ability to create custom validators.

Form Validation	
Form	**Details**
Template-driven forms	Tied to template directives, and must provide custom validator directives that wrap validation functions
Reactive forms	Define custom validators as functions that receive a control to validate

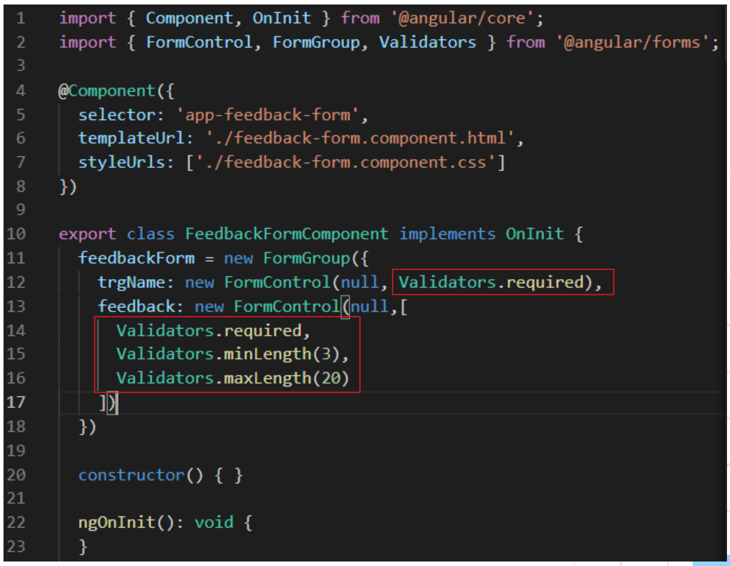
- For more information, see

  https://angular.io/guide/form-validation

Mohamed Mukthar Ahmed

231

Feedback Form - Validators

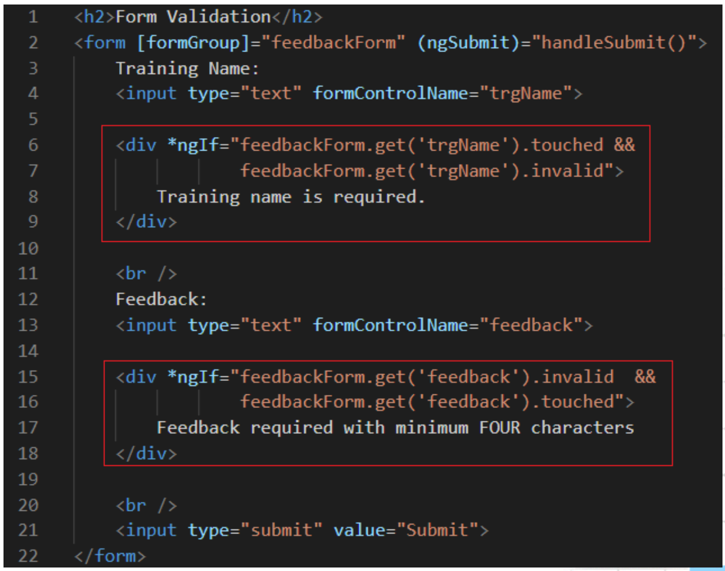


```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';

@Component({
  selector: 'app-feedback-form',
  templateUrl: './feedback-form.component.html',
  styleUrls: ['./feedback-form.component.css']
})

export class FeedbackFormComponent implements OnInit {
  feedbackForm = new FormGroup({
    trgName: new FormControl(null, Validators.required),
    feedback: new FormControl(null,[
      Validators.required,
      Validators.minLength(3),
      Validators.maxLength(20)
    ])
  })

  constructor() { }

  ngOnInit(): void {
  }
```

Mohamed Mukthar Ahmed

232

Feedback Form - Validators

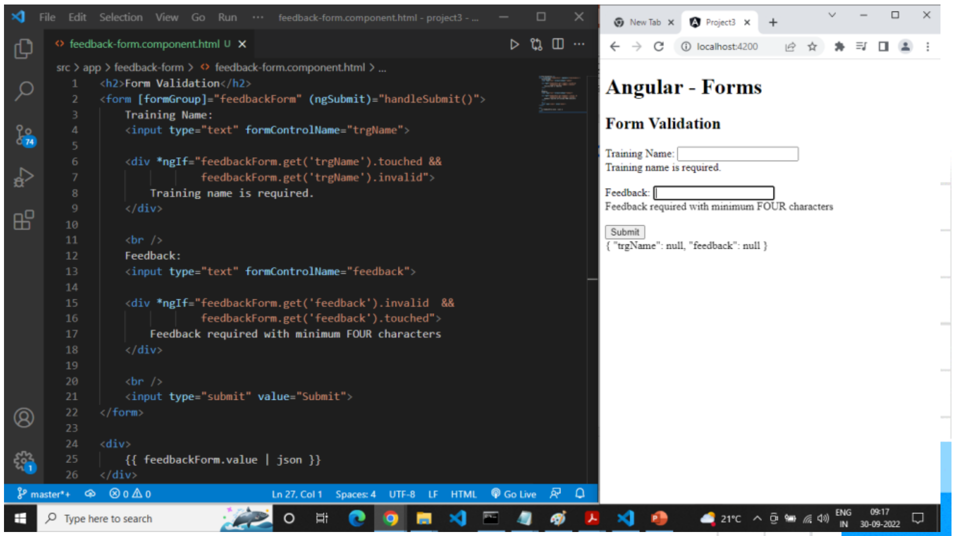


```
<h2>Form Validation</h2>
<form [formGroup]="feedbackForm" (ngSubmit)="handleSubmit()">
    Training Name:
    <input type="text" formControlName="trgName">

    <div *ngIf="feedbackForm.get('trgName').touched &&
            feedbackForm.get('trgName').invalid">
        Training name is required.
    </div>

    <br />
    Feedback:
    <input type="text" formControlName="feedback">

    <div *ngIf="feedbackForm.get('feedback').invalid  &&
            feedbackForm.get('feedback').touched">
        Feedback required with minimum FOUR characters
    </div>

    <br />
    <input type="submit" value="Submit">
</form>
```

Mohamed Mukthar Ahmed

233

Feedback Form - Validators



feedback-form.component.html

src > app > feedback-form > feedback-form.component.html > ...

```html
<h2>Form Validation</h2>
<form [formGroup]="feedbackForm" (ngSubmit)="handleSubmit()">
    Training Name:
    <input type="text" formControlName="trgName">

    <div *ngIf="feedbackForm.get('trgName').touched &&
                feedbackForm.get('trgName').invalid">
        Training name is required.
    </div>

    <br />
    Feedback:
    <input type="text" formControlName="feedback">

    <div *ngIf="feedbackForm.get('feedback').invalid  &&
                feedbackForm.get('feedback').touched">
        Feedback required with minimum FOUR characters
    </div>

    <br />
    <input type="submit" value="Submit">
</form>

<div>
    {{ feedbackForm.value | json }}
</div>
```

localhost:4200

Angular - Forms

Form Validation

Training Name:
Training name is required.

Feedback:
Feedback required with minimum FOUR characters

Submit

```
{ "trgName": null, "feedback": null }
```

Mohamed Mukthar Ahmed

Q&A

Mohamed Mukthar Ahmed