Angular TS – Forms Validations

[**✅ Task 7: Reactive Forms – Validation & Handling** 1](#_Toc204792734)

[**1️⃣ Purpose** 1](#_Toc204792735)

[**2️⃣ Theory** 1](#_Toc204792736)

[**3️⃣ Prerequisites** 1](#_Toc204792737)

[**4️⃣ Code Example (Step-by-Step)** 1](#_Toc204792738)

[**5️⃣ Project Structure Snapshot** 3](#_Toc204792739)

[**6️⃣ Summary** 3](#_Toc204792740)

**✅ Task 7: Reactive Forms – Validation & Handling**

**1️⃣ Purpose**

To build and validate forms using Angular's **Reactive Forms** approach:

* Dynamically handle form state.
* Validate user input with custom error messages.
* Capture and process form data using FormGroup, FormControl, and Validators.

**2️⃣ Theory**

| **Concept** | **Description** |
| --- | --- |
| FormGroup | A collection of FormControls with their state and values. |
| FormControl | Represents a single form input field. |
| Validators | Built-in class to apply rules like required, minLength, etc. |
| ReactiveFormsModule | Required to enable reactive forms functionality. |

**3️⃣ Prerequisites**

✅ Install ReactiveFormsModule in your standalone component.  
✅ Import it manually since we’re not using NgModule.

**4️⃣ Code Example (Step-by-Step)**

**✅ a) Generate Student Form Component**

ng generate component components/student-form --standalone

**✅ b) student-form.component.ts**

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

import { ReactiveFormsModule, FormGroup, FormControl, Validators } from '@angular/forms';

import { CommonModule } from '@angular/common';

@Component({

selector: 'app-student-form',

standalone: true,

imports: [CommonModule, ReactiveFormsModule],

template: `

<h2>🎓 Student Registration Form</h2>

<form [formGroup]="studentForm" (ngSubmit)="onSubmit()">

<label>

Name:

<input type="text" formControlName="name" />

<span \*ngIf="studentForm.get('name')?.invalid && studentForm.get('name')?.touched" style="color:red">

Name is required

</span>

</label>

<br/><br/>

<label>

Email:

<input type="email" formControlName="email" />

<span \*ngIf="studentForm.get('email')?.invalid && studentForm.get('email')?.touched" style="color:red">

Valid Email is required

</span>

</label>

<br/><br/>

<label>

Age:

<input type="number" formControlName="age" />

<span \*ngIf="studentForm.get('age')?.invalid && studentForm.get('age')?.touched" style="color:red">

Age must be between 18 and 60

</span>

</label>

<br/><br/>

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

</form>

<pre>{{ studentForm.value | json }}</pre>

`

})

export class StudentFormComponent {

studentForm = new FormGroup({

name: new FormControl('', Validators.required),

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

age: new FormControl('', [Validators.required, Validators.min(18), Validators.max(60)])

});

onSubmit() {

if (this.studentForm.valid) {

alert('✅ Form Submitted:\n' + JSON.stringify(this.studentForm.value, null, 2));

} else {

alert('❌ Form is invalid!');

}

}

}

**✅ c) Add Route in app.routes.ts**

import { StudentFormComponent } from './components/student-form/student-form.component';

export const appRoutes: Routes = [

{ path: 'student-form', component: StudentFormComponent },

// other routes...

];

**✅ d) Update app.component.ts Navigation**

<nav>

<a routerLink="/">Home</a> |

<a routerLink="/about">About</a> |

<a routerLink="/students">Students</a> |

<a routerLink="/student-form">Student Form</a>

</nav>

**5️⃣ Project Structure Snapshot**

src/

├── app/

│ ├── components/

│ │ └── student-form/ ✅

│ ├── app.routes.ts ✅

│ └── app.config.ts

**6️⃣ Summary**

✅ You now know how to:

* Create a reactive form.
* Use FormGroup, FormControl, and Validators.
* Show validation errors.
* Handle form submissions.