Forms in Angular

February 23, 2025

What We’ll Cover:

1. Template-Driven Forms (Basic Forms with ngModel)
2. Reactive Forms (Using FormBuilder and FormGroup)
3. Form Validation (Required, Minlength, Custom Validators)
4. Handling Form Submission
5. Displaying Validation Errors to Users

Template-Driven Forms

- **Concept**: Simple forms driven by the template
- **Setup**: Requires FormsModule
- **Example**:

```
@Component({

  standalone: true,

  imports: [FormsModule],

  template: `

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

        <input name="name" ngModel>

        <button type="submit">Submit</button>

        </form>

  `

})

export class TemplateFormComponent {

  onSubmit(form: NgForm) {       console.log(form.value); // { name: "input value" } }}
```

Key Point: ngModel binds data, #f="ngForm" tracks form state

Use Case: Quick, small forms

Reactive Forms

- **Concept**: Forms defined programmatically in TypeScript
- **Setup**: Requires ReactiveFormsModule, FormBuilder
- **Example**:

```
@Component({

  standalone: true,

  imports: [ReactiveFormsModule],

  template: `

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

        <input formControlName="name">

        <button type="submit">Submit</button>

        </form>

  `

})

export class ReactiveFormComponent {

  form = this.fb.group({     name: ['']  });

  constructor(private fb: FormBuilder) {}

  onSubmit() {console.log(this.form.value); // { name: "input value" }}} //
```

Key Point: More control, better for complex forms

Form Validation

- **Concept**: Ensure user input meets requirements
- **Built-in Validators**: required, minLength, etc.
- **Reactive Example**:

```
form = this.fb.group({
  name: ['', [Validators.required, Validators.minLength(3)]]
});
```

Custom Validator:

```
noSpaces(control: AbstractControl) {
  return control.value.includes(' ') ? { noSpaces: true } : null;
}
form = this.fb.group({
  name: ['', [Validators.required, this.noSpaces]]
}); ///
```

Use Case: Enforce rules (e.g., no spaces in usernames)

Handling Form Submission

- **Concept**: Process form data on submit
- **Template-Driven**:

```
onSubmit(form: NgForm) {
  if (form.valid) {
        console.log('Submitted:', form.value);
  }
}
```

Reactive:

```
onSubmit() {
  if (this.form.valid) {
        console.log('Submitted:', this.form.value);
  }
}
```

Key Point: Check valid before processing

Demo: Submit → Log data

Displaying Validation Errors

- **Concept**: Show users what’s wrong
- **Template-Driven**:

```
template: `
  <input name="name" ngModel required #name="ngModel">
  @if (name.touched && name.invalid) {
        <p>Name is required!</p>
  }
`
```

Reactive:

```
template: `
  <input formControlName="name">
  @if (form.get('name')?.touched && form.get('name')?.invalid) {
        <p>Name is required!</p>
  }
` ///
```

Key Point: Use touched/dirty to avoid premature errors

Putting It Together

- **Mini-Project**: Task creation form
- **Code**:

```
@Component({
  standalone: true,
  imports: [ReactiveFormsModule],
  template: `
        <form [formGroup]="taskForm" (ngSubmit)="onSubmit()">
        <input formControlName="task" placeholder="New Task">
        @if (taskForm.get('task')?.touched && taskForm.get('task')?.invalid) {
        <p>Task is required and must be 3+ chars!</p>}
        <button type="submit" [disabled]="taskForm.invalid">Add</button>
        </form>
        <ul>@for (task of tasks; track task) { <li>{{ task }}</li> }</ul> `})
```

```
export class TaskFormComponent {

  tasks: string[] = [];

  taskForm = this.fb.group({

        task: ['', [Validators.required, Validators.minLength(3)]]

  });

  constructor(private fb: FormBuilder) {}

  onSubmit() {

        if (this.taskForm.valid) {

        this.tasks.push(this.taskForm.value.task);

        this.taskForm.reset();

        }

  }

} //Demo: Add tasks with validation
```

Key Takeaways

- Template-driven: Simple, template-based forms
- Reactive: Powerful, TypeScript-driven forms
- Validation: Built-in and custom rules
- Submission: Handle data when valid
- Errors: Guide users with clear feedback