Angular provides a powerful and flexible Forms module that allows you to build both template-driven forms and reactive forms. Forms in Angular are used to capture user input and perform actions based on that input. Validations play a crucial role in ensuring that the data entered by users is accurate and meets the required criteria.

Let's explore both template-driven forms and reactive forms, along with validations.

Template-Driven Forms:

Template-driven forms are simpler and are suitable for simple scenarios where the logic is primarily in the template.

1. Import the FormsModule:

```
// app.module.ts
import { FormsModule } from '@angular/forms';

@NgModule({
  imports: [FormsModule],
  // ...
})
export class AppModule { }
```

2. Creating a Template-Driven Form:

```
<!-- app.component.html -->
<form #myForm="ngForm" (ngSubmit)="onSubmit()">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name" ngModel required>

  <label for="email">Email:</label>
  <input type="email" id="email" name="email" ngModel required>

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

3. Handling Form Submission:

```
// app.component.ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
})
```

```
export class AppComponent {
  onSubmit() {
    // Form submission logic
    console.log('Form submitted!');
  }
}
```

4. Form Validations in Template-Driven Forms:

```
<!-- app.component.html -->
<form #myForm="ngForm" (ngSubmit)="onSubmit()">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name" ngModel required minlength="3"
maxlength="30">

  <div *ngIf="myForm.controls.name.invalid && myForm.controls.name.touched">
    <p *ngIf="myForm.controls.name.errors.required">Name is required.</p>
    <p *ngIf="myForm.controls.name.errors.minlength">Name must be at least 3
characters.</p>
    <p *ngIf="myForm.controls.name.errors.maxlength">Name cannot exceed 30
characters.</p>
  </div>

  <label for="email">Email:</label>
  <input type="email" id="email" name="email" ngModel required email>

  <div *ngIf="myForm.controls.email.invalid && myForm.controls.email.touched">
    <p *ngIf="myForm.controls.email.errors.required">Email is required.</p>
    <p *ngIf="myForm.controls.email.errors.email">Invalid email format.</p>
  </div>

  <button type="submit" [disabled]="myForm.invalid">Submit</button>
</form>
```

Reactive Forms:

Reactive forms provide more control and flexibility, especially for complex scenarios and dynamic forms.

1. Import the ReactiveFormsModule:

```
// app.module.ts
import { ReactiveFormsModule } from '@angular/forms';

@NgModule({
  imports: [ReactiveFormsModule],
  // ...
```

```
})
export class AppModule { }
```

2. Creating a Reactive Form:

```
// app.component.ts
import { Component, FormBuilder, FormGroup, Validators } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
})
export class AppComponent {
  myForm: FormGroup;

  constructor(private fb: FormBuilder) {
    this.myForm = this.fb.group({
      name: ['', [Validators.required, Validators.minLength(3),
Validators.maxLength(30)]],
      email: ['', [Validators.required, Validators.email]],
    });
  }

  onSubmit() {
    // Form submission logic
    console.log('Form submitted!');
  }
}
```

3. Form Validations in Reactive Forms:

```
<!-- app.component.html -->
<form [formGroup]="myForm" (ngSubmit)="onSubmit()">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name" formControlName="name">

  <div *ngIf="myForm.get('name').invalid && myForm.get('name').touched">
    <p *ngIf="myForm.get('name').hasError('required')">Name is required.</p>
    <p *ngIf="myForm.get('name').hasError('minlength')">Name must be at least 3
characters.</p>
    <p *ngIf="myForm.get('name').hasError('maxlength')">Name cannot exceed 30
characters.</p>
  </div>

  <label for="email">Email:</label>
  <input type="email" id="email" name="email" formControlName="email">
```

```
    <div *ngIf="myForm.get('email').invalid && myForm.get('email').touched">
      <p *ngIf="myForm.get('email').hasError('required')">Email is required.</p>
      <p *ngIf="myForm.get('email').hasError('email')">Invalid email format.</p>
    </div>

    <button type="submit" [disabled]="myForm.invalid">Submit</button>
  </form>
```

Summary:

- Template-driven forms are simpler and often sufficient for basic use cases.
- Reactive forms provide more control, especially for dynamic forms and complex scenarios.
- Validations can be applied using built-in validators or custom validators.
- Error messages can be displayed conditionally based on the form control's validity and user interaction.

Choose the form approach based on your application's requirements and complexity. Both template-driven and reactive forms are powerful tools in Angular for capturing and validating user input.