Forms

Forms are the mainstay of business applications. You use forms to log in, submit a help request, place an order, book a flight, schedule a meeting, and perform countless other data-entry tasks.

In developing a form, it's important to create a data-entry experience that guides the user efficiently and effectively through the workflow.

Developing forms requires design skills (which are out of scope for this page), as well as framework support for two-way data binding, change tracking, validation, and error handling, which you'll learn about on this page.

* Build an Angular form with a component and template.
* Use [ngModel](https://angular.io/api/forms/NgModel) to create two-way data bindings for reading and writing input-control values.
* Track state changes and the validity of form controls.
* Provide visual feedback using special CSS classes that track the state of the controls.
* Display validation errors to users and enable/disable form controls.
* Share information across HTML elements using template reference variables.

# Template-driven forms

You can build forms by writing templates in the Angular [template syntax](https://angular.io/guide/template-syntax) with the form-specific directives and techniques described in this page.

You can build almost any form with an Angular template—login forms, contact forms, and pretty much any business form. You can lay out the controls creatively, bind them to data, specify validation rules and display validation errors, conditionally enable or disable specific controls, trigger built-in visual feedback, and much more.

## Template-Driven Forms

The app component doesn't need to do much since the form fields and validators are defined in the template when using Angular template-driven forms. The component defines a model object which is bound to the form fields in the template in order to give you access to the data entered into the form from the app component.

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

@Component({

  selector: 'app',

  templateUrl: 'app.component.html'

})

export class AppComponent {

  model: any = {};

  onSubmit() {

    alert('SUCCESS!! :-)\n\n' + JSON.stringify(this.model))

  }

}

The app component template contains all the html markup for displaying the example registration form in your browser. The form input fields use the [(ngModel)] directive to bind to properties of the model object in the app component. Validation is implemented using the attributes required, minlength and email, the Angular framework contains directives that match these attributes with built-in validator functions.

The form binds the submit event to the onSubmit() event handler in the app component using the Angular event binding (ngSubmit)="onSubmit()". Validation messages are displayed only after the user attempts to submit the form for the first time, this is controlled with the f.submitted property of the #f="ngForm" Angular template variable.

<!-- main app container -->

<div class="jumbotron">

    <div class="container">

        <div class="row">

            <div class="col-md-6 offset-md-3">

                <h3>Angular 6 Template-Driven Form Validation</h3>

                <form name="form" (ngSubmit)="f.form.valid && onSubmit()" #f="ngForm" novalidate>

                    <div class="form-group">

                        <label for="firstName">First Name</label>

                        <input type="text" class="form-control" name="firstName" [(ngModel)]="model.firstName" #firstName="ngModel" [ngClass]="{ 'is-invalid': f.submitted && firstName.invalid }" required />

                        <div \*ngIf="f.submitted && firstName.invalid" class="invalid-feedback">

                            <div \*ngIf="firstName.errors.required">First Name is required</div>

                        </div>

                    </div>

                    <div class="form-group">

                        <label for="lastName">Last Name</label>

                        <input type="text" class="form-control" name="lastName" [(ngModel)]="model.lastName" #lastName="ngModel" [ngClass]="{ 'is-invalid': f.submitted && lastName.invalid }" required />

                        <div \*ngIf="f.submitted && lastName.invalid" class="invalid-feedback">

                            <div \*ngIf="lastName.errors.required">Last Name is required</div>

                        </div>

                    </div>

                    <div class="form-group">

                        <label for="email">Email</label>

                        <input type="text" class="form-control" name="email" [(ngModel)]="model.email" #email="ngModel" [ngClass]="{ 'is-invalid': f.submitted && email.invalid }" required email />

                        <div \*ngIf="f.submitted && email.invalid" class="invalid-feedback">

                            <div \*ngIf="email.errors.required">Email is required</div>

                            <div \*ngIf="email.errors.email">Email must be a valid email address</div>

                        </div>

                    </div>

                    <div class="form-group">

                        <label for="password">Password</label>

                        <input type="password" class="form-control" name="password" [(ngModel)]="model.password" #password="ngModel" [ngClass]="{ 'is-invalid': f.submitted && password.invalid }" required minlength="6" />

                        <div \*ngIf="f.submitted && password.invalid" class="invalid-feedback">

                            <div \*ngIf="password.errors.required">Password is required</div>

                            <div \*ngIf="password.errors.minlength">Password must be at least 6 characters</div>

                        </div>

                    </div>

                    <div class="form-group">

                        <button class="btn btn-primary">Register</button>

                    </div>

                </form>

            </div>

        </div>

    </div>

</div>

# Reactive Forms

The app component defines the form fields and validators for our registration form using an Angular FormBuilder to create an instance of a FormGroup that is stored in the registerForm property. The registerForm is then bound to the form in the template below using the [formGroup] directive.

I also added a getter 'f' as a convenience property to make it easier to access form controls from the template. So for example you can access the email field in the template using f.email instead of registerForm.controls.email.

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

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

@Component({

    selector: 'app',

    templateUrl: 'app.component.html'

})

export class AppComponent implements OnInit {

    registerForm: FormGroup;

    submitted = false;

    constructor(private formBuilder: FormBuilder) { }

    ngOnInit() {

        this.registerForm = this.formBuilder.group({

            firstName: ['', Validators.required],

            lastName: ['', Validators.required],

            email: ['', [Validators.required, Validators.email]],

            password: ['', [Validators.required, Validators.minLength(6)]]

        });

    }

    // convenience getter for easy access to form fields

    get f() { return this.registerForm.controls; }

    onSubmit() {

        this.submitted = true;

        // stop here if form is invalid

        if (this.registerForm.invalid) {

            return;

        }

        alert('SUCCESS!! :-)')

    }

}

The app component template contains all the html markup for displaying the example registration form in your browser. The form element uses the [formGroup] directive to bind to the registerForm FormGroup in the app component above.

The form binds the form submit event to the onSubmit() handler in the app component using the Angular event binding (ngSubmit)="onSubmit()". Validation messages are displayed only after the user attempts to submit the form for the first time, this is controlled with the submitted property of the app component.

<!-- main app container -->

<div class="jumbotron">

    <div class="container">

        <div class="row">

            <div class="col-md-6 offset-md-3">

                <h2>Angular 6 Reactive Form Validation</h2>

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

                    <div class="form-group">

                        <label>First Name</label>

                        <input type="text" formControlName="firstName" class="form-control" [ngClass]="{ 'is-invalid': submitted && f.firstName.errors }" />

                        <div \*ngIf="submitted && f.firstName.errors" class="invalid-feedback">

                            <div \*ngIf="f.firstName.errors.required">First Name is required</div>

                        </div>

                    </div>

                    <div class="form-group">

                        <label>Last Name</label>

                        <input type="text" formControlName="lastName" class="form-control" [ngClass]="{ 'is-invalid': submitted && f.lastName.errors }" />

                        <div \*ngIf="submitted && f.lastName.errors" class="invalid-feedback">

                            <div \*ngIf="f.lastName.errors.required">Last Name is required</div>

                        </div>

                    </div>

                    <div class="form-group">

                        <label>Email</label>

                        <input type="text" formControlName="email" class="form-control" [ngClass]="{ 'is-invalid': submitted && f.email.errors }" />

                        <div \*ngIf="submitted && f.email.errors" class="invalid-feedback">

                            <div \*ngIf="f.email.errors.required">Email is required</div>

                            <div \*ngIf="f.email.errors.email">Email must be a valid email address</div>

                        </div>

                    </div>

                    <div class="form-group">

                        <label>Password</label>

                        <input type="password" formControlName="password" class="form-control" [ngClass]="{ 'is-invalid': submitted && f.password.errors }" />

                        <div \*ngIf="submitted && f.password.errors" class="invalid-feedback">

                            <div \*ngIf="f.password.errors.required">Password is required</div>

                            <div \*ngIf="f.password.errors.minlength">Password must be at least 6 characters</div>

                        </div>

                    </div>

                    <div class="form-group">

                        <button [disabled]="loading" class="btn btn-primary">Register</button>

                    </div>

                </form>

            </div>

        </div>

    </div>

</div>