**Reactive Forms Validation App Component**

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!! :-)')

    }

}

## Reactive Forms Validation App Template

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>

## Reactive Forms Validation App Module

There isn't much going on in the app module other than the standard stuff, the main thing you need to remember for using reactive forms in Angular is to import the ReactiveFormsModule from '@angular/forms' and include it in the imports array of the @NgModule decorator.

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

import { BrowserModule } from '@angular/platform-browser';

import { ReactiveFormsModule } from '@angular/forms';

import { AppComponent } from './app.component';

@NgModule({

  imports: [

    BrowserModule,

    ReactiveFormsModule

  ],

  declarations: [

    AppComponent

  ],

  bootstrap: [AppComponent]

})

export class AppModule { }