Reactive Forms Module

Create application

>ng new formdemo

>cd formdemo

>ng serve

Open it in VS Code

Create new component

>ng g c register

Add this component in app.component.html file

<app-register></app-register>

We are working with Reactive Forms so need to import reactive for in app.module.ts

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

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

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

import { AppRoutingModule } from './app-routing.module';

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

import { RegisterComponent } from './register/register.component';

@NgModule({

  declarations: [

    AppComponent,

    RegisterComponent

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    ReactiveFormsModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

Before working with register component let’s understand by default ts file is asking for all variable value (initialization required)

If you want to remove that rule then open tsconfig.json file and add one line in compiler options

{

  "compileOnSave": false,

  "compilerOptions": {

    "strictPropertyInitialization": false,

    "baseUrl": "./",

    "outDir": "./dist/out-tsc",

    "forceConsistentCasingInFileNames": true,

    "strict": true,

Save the file.

Include bootstrap >npm install bootstrap and edit styles.css file

/\* You can add global styles to this file, and also import other style files \*/

@import '../node\_modules/bootstrap/dist/css/bootstrap.min.css';

Let’s start creating Reactive form.

Open register.component.ts.

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

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

@Component({

  selector: 'app-register',

  templateUrl: './register.component.html',

  styleUrls: ['./register.component.css']

})

export class RegisterComponent implements OnInit {

  //form declared

  registerForm: FormGroup;

  submitted: boolean = false;

  //builder dependency injected through constructor

  constructor(private builder: FormBuilder) { }

  ngOnInit(): void {

    this.registerForm = this.builder.group({

      firstName: ['', Validators.required],

      lastName: ['', Validators.required],

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

      password:['',[Validators.required,Validators.minLength(8)]]

    })

  }

  get form() {

    return this.registerForm.controls;

  }

  onSubmit() {

    this.submitted = true;

    if (this.registerForm.invalid)

      return;

    else

      alert("Form submitted for the Approval")

  }

}

Register.component.html

<div class="container">

    <div class="row">

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

            <h2 class="text-danger">Registration Form</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 && form['firstName'].errors}">

                    <div \*ngIf="submitted && form['firstName'].errors" class="invalid-feedback">

                        <div \*ngIf="form['firstName'].errors['required']">

                            First Name cannot be Empty

                        </div>

                    </div>

                </div>

                <div class="form-group">

                    <label>Last Name</label>

                    <input type="text" formControlName="lastName" class="form-control"

                    [ngClass]="{'is-invalid':submitted && form['lastName'].errors}">

                    <div \*ngIf="submitted && form['lastName'].errors" class="invalid-feedback">

                        <div \*ngIf="form['lastName'].errors['required']">

                            Last Name cannot be Empty

                        </div>

                    </div>

                </div>

                <div class="form-group">

                    <label>Email</label>

                    <input type="text" formControlName="email" class="form-control"

                    [ngClass]="{'is-invalid':submitted && form['email'].errors}">

                    <div \*ngIf="submitted && form['email'].errors" class="invalid-feedback">

                        <div \*ngIf="form['email'].errors['required']">

                            Email cannot be Empty

                        </div>

                        <div \*ngIf="form['email'].errors['email']">

                            Enter Valid EmailId

                        </div>

                    </div>

                </div>

                <div class="form-group">

                    <label>Password</label>

                    <input type="password" formControlName="password" class="form-control"

                    [ngClass]="{'is-invalid':submitted && form['password'].errors}">

                    <div \*ngIf="submitted && form['password'].errors" class="invalid-feedback">

                        <div \*ngIf="form['password'].errors['required']">

                            Password cannot be Empty

                        </div>

                        <div \*ngIf="form['password'].errors['minlength']">

                            Password must have atleast 8 character

                        </div>

                    </div>

                </div>

                <br>

                <div class="form-group">

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

                </div>

            </form>

        </div>

    </div>

</div>