Create new app named as formdemo

>ng new formdemo

Change the directory

>cd formdemo

Start your app

>ng serve

**Go to app.component.html**

Delete everything and write below code

<h1>Form Demo</h1>

Note: to prepare reactive form we need to import ReactiveFormModule and FormMudule

App.module.ts

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

import { FormsModule, 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,

    FormsModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

create new component register

>ng g c register

Edit 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 {

  //declare formgroup variable

  registerForm:FormGroup;

  //declare boolean variable to get form submitted or not

  submitted:boolean=false;

  //inject formBuilder dependency to create form with validators

  constructor(private builder: FormBuilder) { }

  //when component initiate we will write code here to bbuild form with validators

  ngOnInit(): void {

    this.registerForm= this.builder.group({

      firstName:["",Validators.required],

      lastName:["",Validators.required],

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

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

    });

  }

  //when user click on submit button

  OnSubmit(){

    this.submitted=true;

    if(this.registerForm.invalid)

      return;

    else

      alert("Form Submitted and Send For Approval")

  }

  //to access form control in view to display the errors

  get f(){

    return this.registerForm.controls;

  }

}

Edit register.component.html

<div class="jumbotron">

    <div class="container">

        <div class="row">

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

                <h2>Reactive Form Demo</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}"/>

                        <!--custom validation-->

                        <div \*ngIf="submitted && f['firstName'].errors">

                            <div \*ngIf="submitted && f['firstName'].errors['required']">

                                First Name is Required and Can't 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 && f['lastName'].errors}"/>

                        <!--custom validation-->

                        <div \*ngIf="submitted && f['lastName'].errors">

                            <div \*ngIf="submitted && f['lastName'].errors['required']">

                                Last Name is Required and Can't be Empty

                            </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}"/>

                        <!--custom validation-->

                        <div \*ngIf="submitted && f['email'].errors">

                            <div \*ngIf="submitted && f['email'].errors['required']">

                                Email is Required and Can't be Empty

                            </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}"/>

                        <!--custom validation-->

                        <div \*ngIf="submitted && f['password'].errors">

                            <div \*ngIf="submitted && f['password'].errors['required']">

                                First Name is Required and Can't be Empty

                            </div>

                        </div>

                    </div>

                    <div class="form-group">

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

                    </div>

                </form>

            </div>

        </div>

    </div>

</div>

To remove compile time error in latest angular version

Edit tsconfig.json file

/\* To learn more about this file see: https://angular.io/config/tsconfig. \*/

{

  "compileOnSave": false,

  "compilerOptions": {

    "strictPropertyInitialization": false,

    "baseUrl": "./",

    "outDir": "./dist/out-tsc",

    "forceConsistentCasingInFileNames": true,

    "strict": true,

    "noImplicitOverride": true,

    "noPropertyAccessFromIndexSignature": true,

    "noImplicitReturns": true,

    "noFallthroughCasesInSwitch": true,

    "sourceMap": true,

    "declaration": false,

    "downlevelIteration": true,

    "experimentalDecorators": true,

    "moduleResolution": "node",

    "importHelpers": true,

    "target": "es2017",

    "module": "es2020",

    "lib": [

      "es2020",

      "dom"

    ]

  },

  "angularCompilerOptions": {

    "enableI18nLegacyMessageIdFormat": false,

    "strictInjectionParameters": true,

    "strictInputAccessModifiers": true,

    "strictTemplates": true

  }

}

Now add bootstrap to your app

Edit indix.html file

<!doctype html>

<html lang="en">

<head>

  <meta charset="utf-8">

  <title>Formdemo</title>

  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="icon" type="image/x-icon" href="favicon.ico">

  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css" integrity="sha384-zCbKRCUGaJDkqS1kPbPd7TveP5iyJE0EjAuZQTgFLD2ylzuqKfdKlfG/eSrtxUkn" crossorigin="anonymous">

</head>

<body>

  <app-root></app-root>

  <script src="https://cdn.jsdelivr.net/npm/jquery@3.5.1/dist/jquery.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-fQybjgWLrvvRgtW6bFlB7jaZrFsaBXjsOMm/tB9LTS58ONXgqbR9W8oWht/amnpF" crossorigin="anonymous"></script>

</body>

</html>

Also bind register component to your app

<h1>Form Demo</h1>

<app-register></app-register>