

Module 8: Handling Forms In Angular

Demo Document 2: Create reactive user registration form

edureka!

edureka!

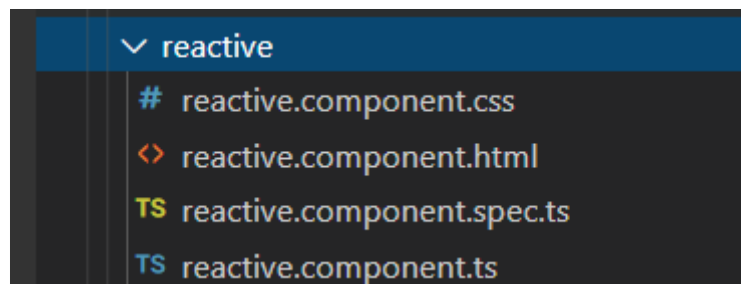
© Brain4ce Education Solutions Pvt. Ltd.

Create reactive user registration form

In this demo, we will see how to create reactive user registration form. In previous demo we have seen how to create template driven user registration form, we will use same form and see how it can be designed using reactive form

Step 1: Create new project using command 'ng new Module8-Demo2'.

Step 2: Create new component under app folder: 'ng g c reactive'



Step 3- Open reactive.component.ts and create form group/form control

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

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

@Component({
  selector: 'app-reactive',
  templateUrl: './reactive.component.html',
  styleUrls: ['./reactive.component.css']
})
export class ReactiveComponent implements OnInit {

  registerForm: FormGroup;
  submitted = false;

  constructor(private formBuilder: FormBuilder) { }

  ngOnInit() {
    this.registerForm = this.formBuilder.group({
      title: ['', Validators.required],
      firstName: ['', Validators.required],
      lastName: ['', Validators.required],
      email: ['', [Validators.required, Validators.email]],
      password: ['', [Validators.required, Validators.minLength(6)]],
      confirmPassword: ['', Validators.required],
      acceptTerms: [false, Validators.requiredTrue]
    });
  }
}
```

Step 4 – Open reactive.component.html and type the below code

```

<div class="d-flex justify-content-center">
  <div class="card m-3">
    <h5 class="card-header">Reactive Form Validation</h5>
    <div class="card-body">
      <form [formGroup]="registerForm" (ngSubmit)="onSubmit()">
        <div class="form-row">
          <div class="form-group col">
            <label>Title</label>
            <select formControlName="title" class="form-control" [ngClass]="{ 'is-invalid': submitted && f.title.errors }" />
              <option value=""></option>
              <option value="Mr">Mr</option>
              <option value="Mrs">Mrs</option>
              <option value="Miss">Miss</option>
              <option value="Ms">Ms</option>
            </select>
          </div>
          <div class="form-group col-5">
            <label>First Name</label>
            <input type="text" formControlName="firstName" class="form-control" [ngClass]="{ 'is-invalid': submitted && f.firstName.errors }" />
          </div>
          <div class="form-group col-5">
            <label>Last Name</label>
            <input type="text" formControlName="lastName" class="form-control" [ngClass]="{ 'is-invalid': submitted && f.lastName.errors }" />
          </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>
        <div class="form-group">
          <label>Email</label>
          <input type="text" formControlName="email" class="form-control" [ngClass]="{ 'is-invalid': submitted && f.email.errors }" />
        </div>
        <div class="form-row">
          <div class="form-group col">
            <label>Password</label>
            <input type="password" formControlName="password" class="form-control" [ngClass]="{ 'is-invalid': submitted && f.password.errors }" />
          </div>
          <div class="form-group col">
            <label>Confirm Password</label>
            <input type="password" formControlName="confirmPassword" class="form-control" [ngClass]="{ 'is-invalid': submitted && f.confirmPassword.errors }" />
          </div>
        </div>
        <div class="form-group form-check">

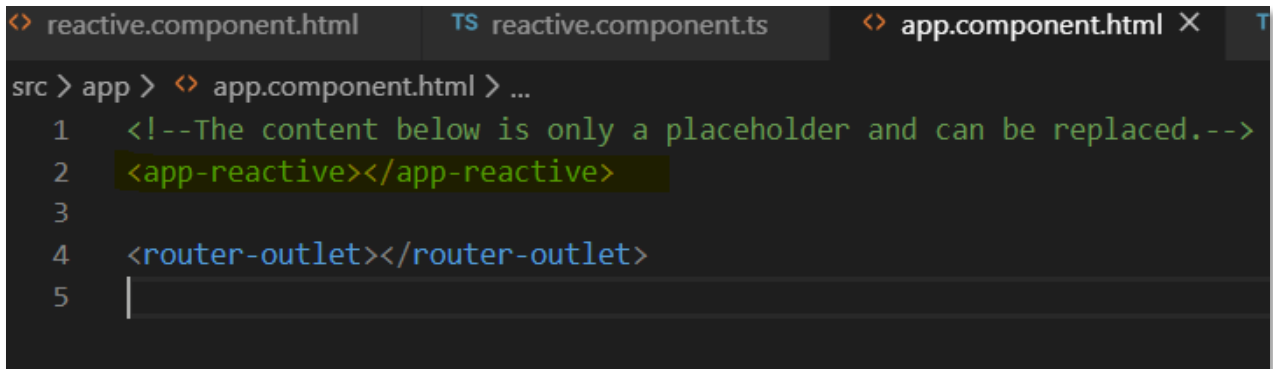
```

```

        <input type="checkbox" formControlName="acceptTerms" id="a
acceptTerms" class="form-check-input" [ngClass]="{ 'is-
invalid': submitted && f.acceptTerms.errors }" />
        <label for="acceptTerms" class="form-check-
label">Accept Terms & Conditions</label>
        <div *ngIf="submitted && f.acceptTerms.errors" class="inva
lid-feedback">Accept Ts & Cs is required</div>
    </div>
    <div class="text-center">
        <button class="btn btn-primary mr-1">Register</button>
        <button class="btn btn-
secondary" type="reset" (click)="onReset()">Clear</button>
    </div>
</form>
</div>
</div>
</div>

```

Step 5 – Open app.component.html and copy <app-reactive> tag as below.



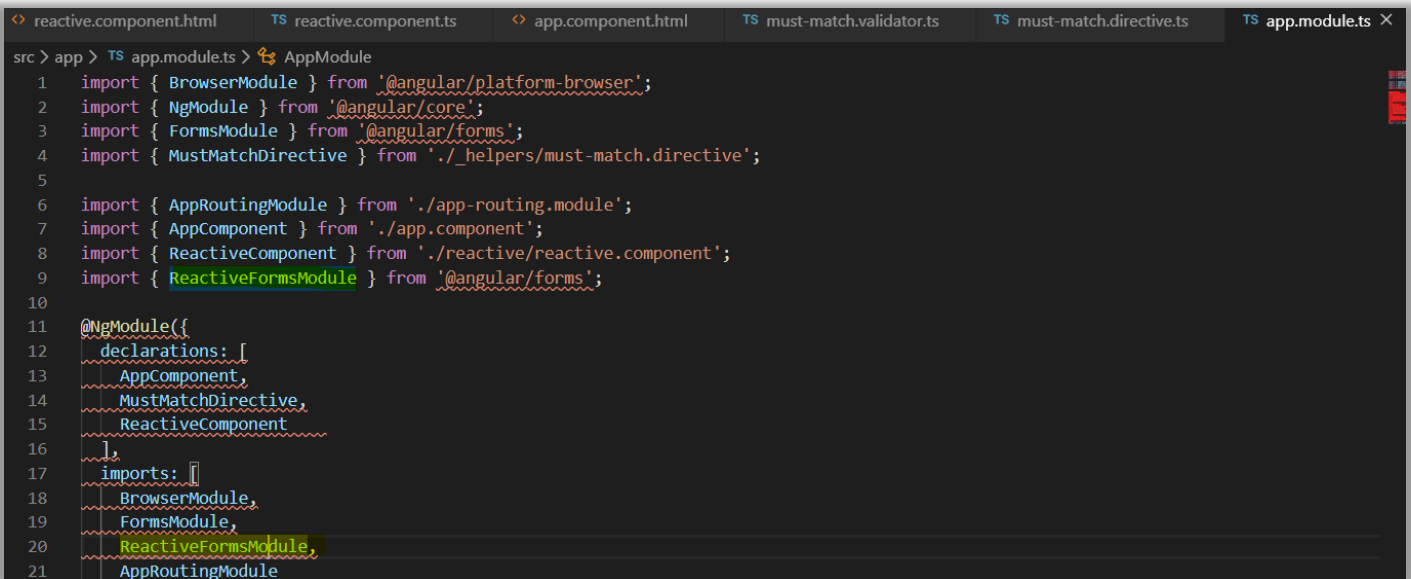
```

src > app > < app.component.html > ...
1  <!--The content below is only a placeholder and can be replaced.-->
2  <app-reactive></app-reactive>
3
4  <router-outlet></router-outlet>
5  |

```

Step 6 - Open app.module.ts and import ReactiveFormsModule as below.

Here is difference between template and reactive form, we are importing ReactiveFormsModule for reactive forms



```
src > app > TS app.module.ts > AppModule
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3  import { FormsModule } from '@angular/forms';
4  import { MustMatchDirective } from '../helpers/must-match.directive';
5
6  import { AppRoutingModule } from './app-routing.module';
7  import { AppComponent } from './app.component';
8  import { ReactiveComponent } from './reactive/reactive.component';
9  import { ReactiveFormsModule } from '@angular/forms';
10
11 @NgModule({
12   declarations: [
13     AppComponent,
14     MustMatchDirective,
15     ReactiveComponent
16   ],
17   imports: [
18     BrowserModule,
19     FormsModule,
20     ReactiveFormsModule,
21     AppRoutingModule
```

Step 6- Open Index.html and add bootstrap and other references.

Step 7- Run application using command 'ng serve --open'. This will open simple reactive form without any validation.



Reactive Form Validation

Title	First Name	Last Name
<input type="text" value="v"/>	<input type="text"/>	<input type="text"/>

Email

Password	Confirm Password
<input type="text"/>	<input type="text"/>

☐ Accept Terms & Conditions

Though both reactive and template driven forms looks the same, but code written for both is different. Reactive form validations are done inside .ts file. We will check that in next module