Module 9: Validating Angular Forms

Demo Document 2 – Reactive forms with validation

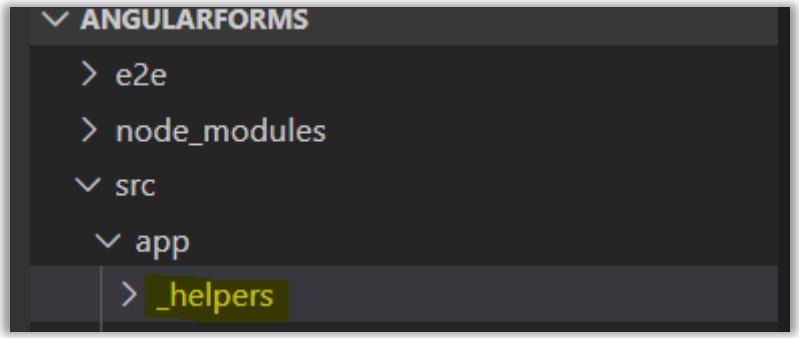
edureka!

Add validation to Reactive form

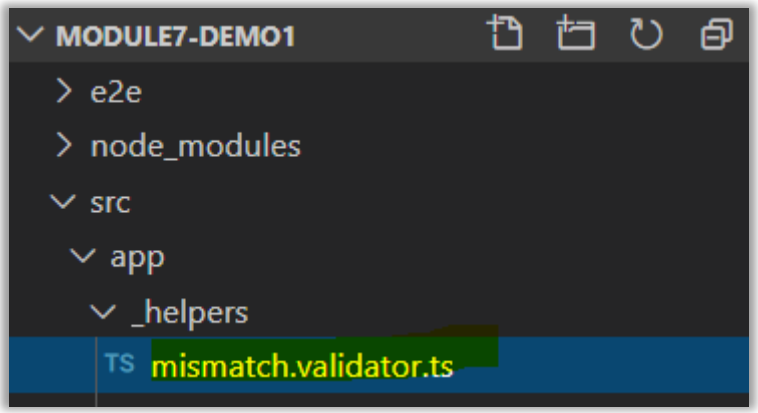
In this demo, we will see how to add validation for different fields for reactive form created in module 8

Step 1: Open project Module8-Demo2, created in module 8 demo2

Step 2: Create _helpers folder inside app folder as shown below



Step 3- Create custom validator to match password and confirm password, right click on _helpers and create file by name mismatch.validator.ts.



Note that, here we are using custom validator not custom directive which is used in template driven form

Step 4 – The mismatch.validator.ts file contains the below code

Purpose of this is to validate password and confirm password fields in the form

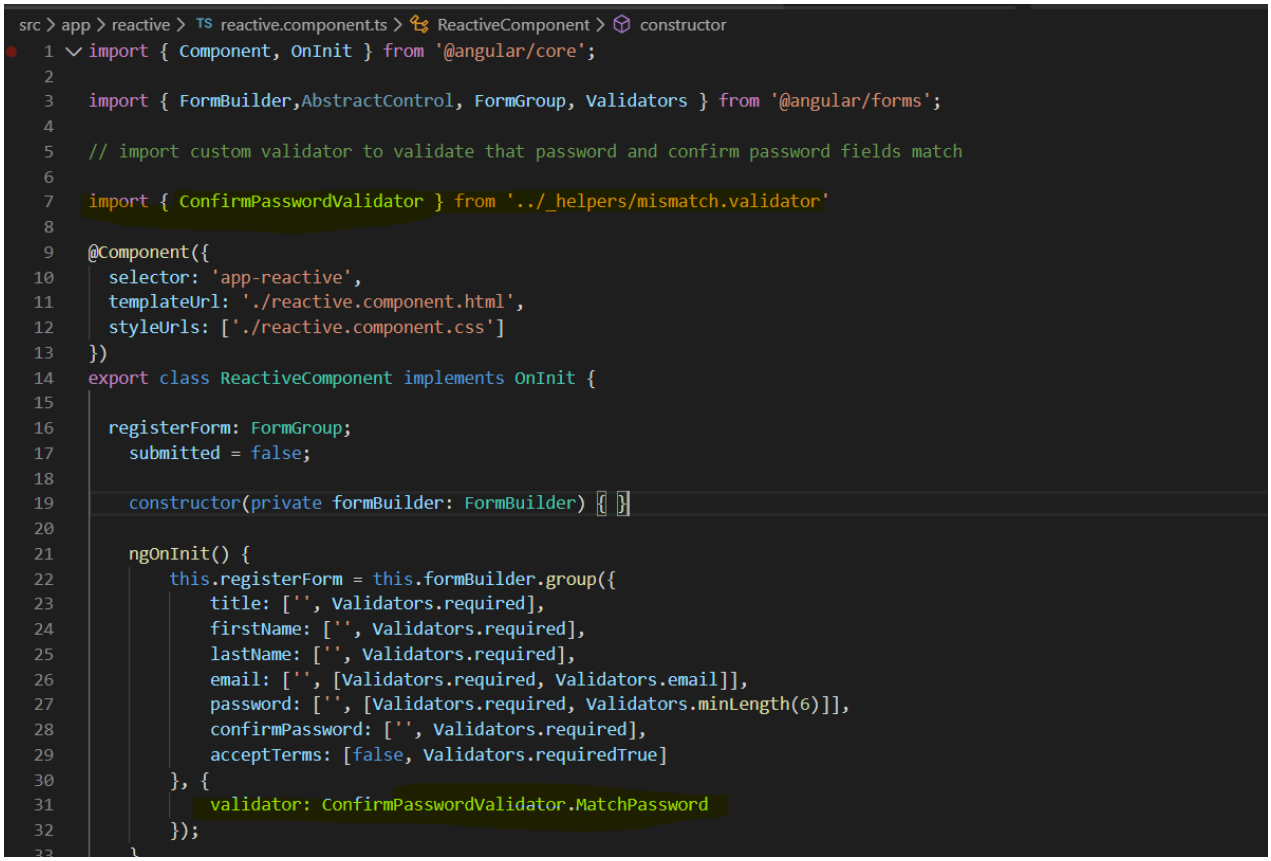
```
import { AbstractControl } from '@angular/forms';
export class ConfirmPasswordValidator {
  static MatchPassword(control: AbstractControl) {
    let password = control.get('password').value;
    let confirmPassword = control.get('confirmPassword').value;
    if (password != confirmPassword) {
      control.get('confirmPassword').setErrors({ ConfirmPassword: true });
```

```
    }
    else {
      return null;
    }
  }
}
```

Step 5- Open reactive.component.ts files and add import 'ConfirmPasswordValidator' as below

```
src > app > reactive > TS reactive.component.ts > ReactiveComponent > constructor
1   import { Component, OnInit } from '@angular/core';
2
3   import { FormBuilder,AbstractControl, FormGroup, Validators } from '@angular/forms';
4
5   // import custom validator to validate that password and confirm password fields match
6
7   import { ConfirmPasswordValidator } from '../_helpers/mismatch.validator'
8
```

Step 6- Add the below line of code to reactive.component.ts file



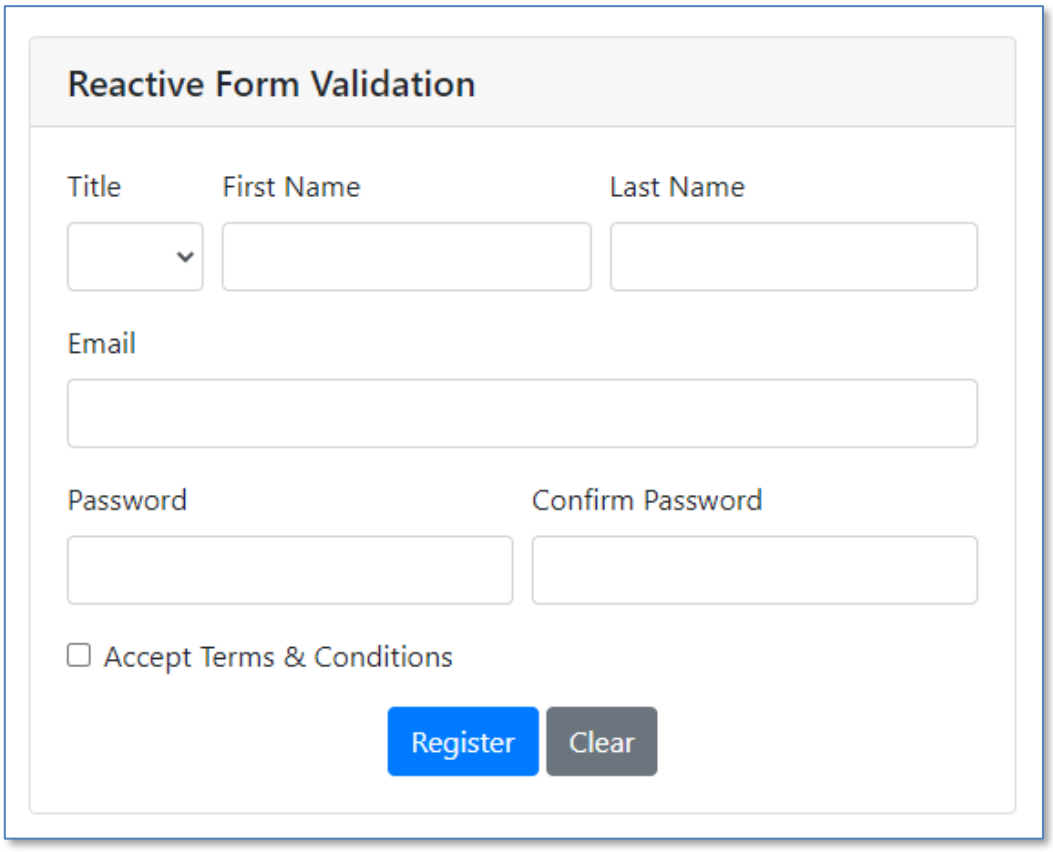
```
src > app > reactive > TS reactive.component.ts > ReactiveComponent > constructor
1   import { Component, OnInit } from '@angular/core';
2
3   import { FormBuilder,AbstractControl, FormGroup, Validators } from '@angular/forms';
4
5   // import custom validator to validate that password and confirm password fields match
6
7   import { ConfirmPasswordValidator } from '../_helpers/mismatch.validator'
8
9   @Component({
10    selector: 'app-reactive',
11    templateUrl: './reactive.component.html',
12    styleUrls: ['./reactive.component.css']
13  })
14  export class ReactiveComponent implements OnInit {
15
16    registerForm: FormGroup;
17      submitted = false;
18
19      constructor(private formBuilder: FormBuilder) { }
20
21      ngOnInit() {
22          this.registerForm = this.formBuilder.group({
23              title: ['', Validators.required],
24              firstName: ['', Validators.required],
25              lastName: ['', Validators.required],
26              email: ['', [Validators.required, Validators.email]],
27              password: ['', [Validators.required, Validators.minLength(6)]],
28              confirmPassword: ['', Validators.required],
29              acceptTerms: [false, Validators.requiredTrue]
30          }, {
31              validator: ConfirmPasswordValidator.MatchPassword
32          });
```

Step 6- Open reactive.component.html file and add the below line of code to confirm password field

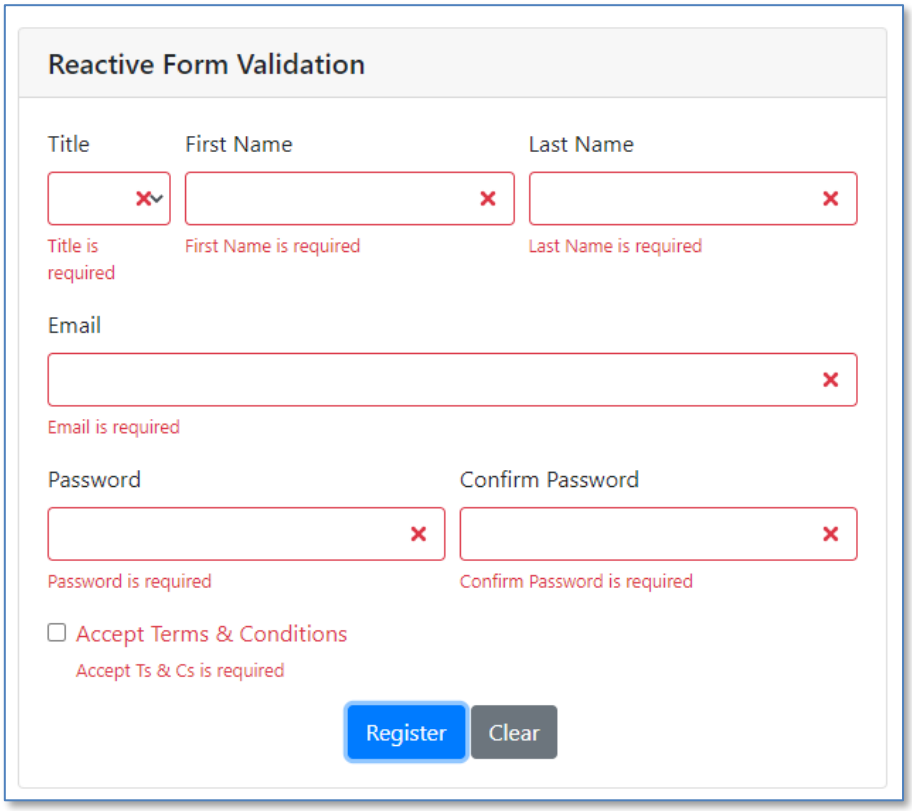
src > app > reactive > reactive.component.html > div.d-flex.justify-content-center > div.card.m-3 > div.card-body > form

```
                    <div *ngIf="f.email.errors.email">Email must be a valid email address</div>
                </div>
            </div>
            <div class="form-row">
                <div class="form-group col">
                    <label>Password</label>
                    <input type="password" formControlName="password" class="form-control" [ngClass]="{ 'is-invalid': submit
                    <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 col">
                    <label>Confirm Password</label>
                    <input type="password" formControlName="confirmPassword" class="form-control" [ngClass]="{ 'is-invalid'
                    <div *ngIf="submitted && f.confirmPassword.errors" class="invalid-feedback">
                        <div *ngIf="f.confirmPassword.errors.required">Confirm Password is required</div>
                        <div *ngIf="registerForm.get('confirmPassword').errors &&
                        registerForm.get('confirmPassword').errors.ConfirmPassword">Passwords must match</div>
                    </div>
                </div>
            </div>
            <div class="form-group form-check">
                <input type="checkbox" formControlName="acceptTerms" id="acceptTerms" class="form-check-input" [ngClass]="{
                <label for="acceptTerms" class="form-check-label">Accept Terms & Conditions</label>
                <div *ngIf="submitted && f.acceptTerms.errors" class="invalid-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>
```

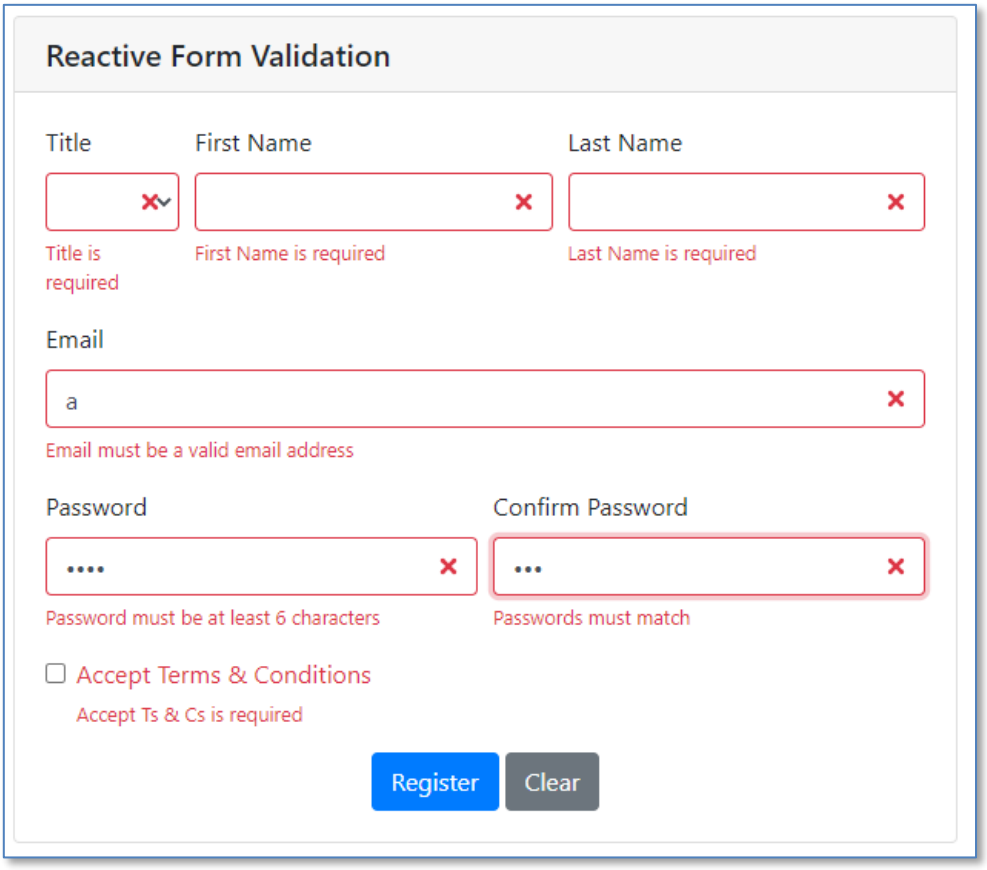
Step 7 – Run app using ng serve command. This will open below window



Now click on register button, you will see all validation messages will be triggered

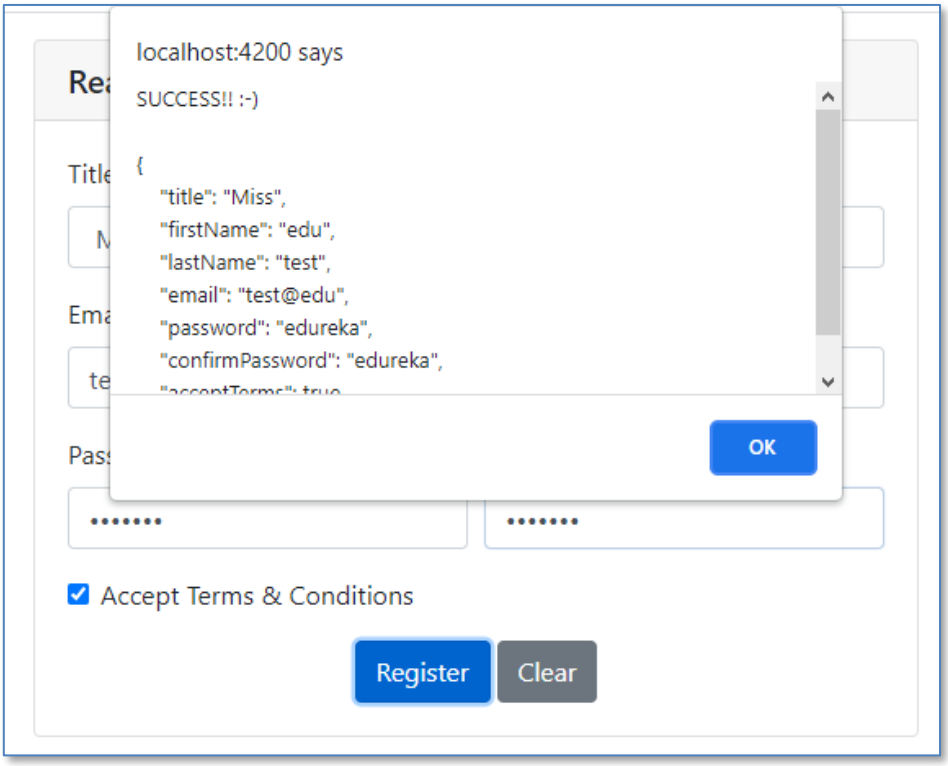


Once you start adding values to fields the validation errors will be removed



Correct all the errors and submit the form

Step 11 – Once all errors are corrected and submitted the form will display a success popup



Major difference between template driven and reactive form is that validation is done in .html file for template driven and in .ts file for reactive form