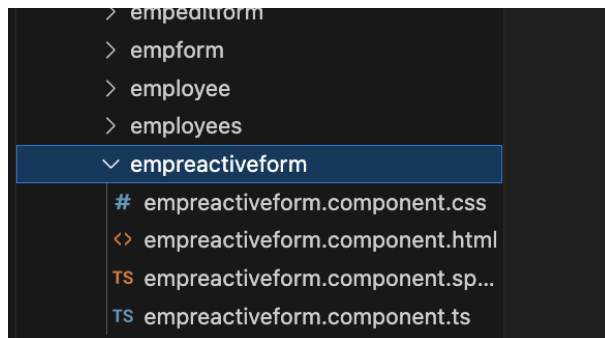


## STEP 13: Reactive Driven Forms

1. Make sure to add **ReactiveFormsModule** in app.module.ts file
2. Add empreactiveform component from within the angulardemo folder as follows:  
ng g c empreactiveform



3. Update the ts component as follows :  
import { [FormControl](#) } from '@angular/forms';

FormControl is a directive that allows you to create and manage a FormControl instance directly.

4. Add below property in the ts component  
name = new [FormControl](#)();

Here you are creating a FormControl called name. It will be bound in the template to an HTML input box for the emp name.

A FormControl constructor accepts three, optional arguments:

- a. the initial data value,
- b. an array of validators,
- c. and an array of async validators.

5. Add below in html component  
<h2>Emp Detail</h2>  
<h3><i>Just a [FormControl](#)</i></h3>  
<label class="center-block">  
Name: <input class="form-control" [formControl]="name">  
</label>

To let Angular know that this is the input that you want to associate to the name FormControl in the class, you need [formControl]="name" in the template on the <input>.

6. Add <app-empreactiveform></app-empreactiveform> in app component html
7. The above is useful when there is onle 1 or 2 form controls. To. Manage a group of form elements we need FormGroup

8. Import following in the component :  
 import { [FormControl](#), [FormGroup](#) } from '@angular/forms';
9. In the class, wrap the FormControl in a FormGroup  
 empForm = new [FormGroup](#) ( { name: new [FormControl](#)() } );
10. Update the template to reflect FormGroup.

```
<form [formGroup]="empForm" novalidate>
  <div class="form-group">
    <label class="center-block">Name: <input class="form-control"
      formControlName="name"> </label>
    </div>
  </form>
```

11. To see the form model, add the following line after the closing form tag.

```
<p>Form value: {{ empForm.value | json }}</p>
<p>Form status: {{ empForm.status | json }}</p>
```

12. For using Validators:  
 For using Validators.required in reactive forms import the Validators symbol.  
 import { [FormBuilder](#), [FormGroup](#), [Validators](#) } from '@angular/forms';
13. To make the name FormControl required, replace the name property in the FormGroup with an array.  
 The first item is the initial value for name; the second is the required validator, Validators.required.  
 name: [", [Validators](#).required ],

14. Update the empreactive form and html as follows for custom validation messages

.ts file :

```
name:FormControl = new FormControl('Shalini', Validators.required);
form:FormGroup ;
constructor(){
  console.log(this.name)
  this.form = new FormGroup({
    name:this.name,
    phone:new FormControl(", [Validators.required, Validators.minLength(10)])
  })
}
```

.html file:

```
<div class="mb-4">
  <h3>Reactive Driven Form Demo</h3>
  <p>{{ form.value | json }}</p>
```

```

<p>{{ form.status | json }}</p>
<form [formGroup]="form">
  <!-- <input type="text" placeholder="Enter"
  name="name" class="form-control" [formControl]="name" /> -->
  <input type="text" placeholder="Enter"
  name="name" class="form-control" formControlName="name" />

  <div *ngIf="name.invalid && name.errors?.['required']">
    <span class="text-danger">Name is required</span>
  </div>

  <input type="text" placeholder="Enter"
  name="phone" class="form-control" formControlName="phone" />

  <div *ngIf="form.controls['phone'].invalid">
    <div *ngIf="form.controls['phone'].errors?.['required']">
      <span class="text-danger">Phone is required</span>
    </div>
    <div *ngIf="form.controls['phone'].errors?.['minlength']">
      <span class="text-danger">Phone should be 10 digits</span>
    </div>
  </div>

</form>
</div>

```

15. Can update all the above code to create form with in-built and custom validations

16. Create a folder validators and a file password.ts file within it. Add below code to create custom validation for password

```

import { AbstractControl, FormControl } from "@angular/forms";

export function hasExclamationMark(input: AbstractControl) {
  const hasExclamation = input.value.indexOf('!') >= 0;
  return hasExclamation ? null : { needsExclamation: true };
}

```

17. Update the ts component as follows:

```

subsemail:FormControl;
empform:FormGroup;
email:FormControl;
address:FormGroup
city:FormControl
password:FormControl;
constructor() {
  this.subsemail = new FormControl("",Validators.required);
  this.email=new FormControl("",Validators.required);
}

```

```

    this.city = new FormControl("", Validators.required);
    this.address = new FormGroup({ city: this.city })
    this.password = new FormControl("", Validators.required);
    console.log(this.subemail);
    this.empform = new FormGroup({
      ename: new FormControl('Sample name', [Validators.required,
Validators.minLength(5)]),
      email: this.email,
      address: this.address,
      password: this.password
    })
  }
  subscribe()
  {
    console.log(this.subemail.value)
  }
}

```

18. Update the HTML as follows:

```

<h1>Reactive Forms</h1>
<h3>{{ empform.value | json }}</h3>
<h3>{{ empform.status }}</h3>
<form [formGroup]="empform">
  <div class="mb-3">
    <input type="text" class="form-control" [ngClass]="{'is-
invalid': empform.controls.ename.invalid}"
    formControlName="ename" placeholder="Enter name"/>
    <div *ngIf="empform.controls.ename.invalid &&
empform.controls.ename.errors?.required">
      <span class="text-danger">It is required</span>
    </div>
    <div *ngIf="empform.controls.ename.errors?.minlength">
      <span class="text-danger">length 5</span>
    </div>
  </div>
  <div class="mb-3">
    <input type="text" class="form-control"
    formControlName="email" placeholder="Enter email"/>
    <div *ngIf="email.invalid && email.errors?.required">
      <span class="text-danger">It is required</span>
    </div>
  </div>
  <div class="mb-3">
    <input type="text" class="form-control"
    formControlName="password" placeholder="Enter password"/>
    <div *ngIf="password.invalid && password.errors?.needsExclamation">
      <span class="text-danger">! is required</span>
    </div>
  </div>
  <div [formGroup]="address">

```

```
<div class="mb-3">
  <input type="text" class="form-control"
    formControlName="city" placeholder="Enter city"/>
  <div *ngIf="city.invalid && city.errors?.required">
    <span class="text-danger">It is required</span>
  </div>
</div>
</div>
```

```
</form>
<input type="text" class="form-control"
  [formControl]="subsemail"
  placeholder="enter email to subscribe">
<button (click)="subscribe()">Subscribe</button>
```