|  |
| --- |
|  |
| Angular 8 |
|  |
| Angular – Reactive Form |

**TechBrain Express**

December 12, 2019

Authored by: Ashu

Template Forms, Event Handling, Two Way Data Binding, Validation

* **Demonstrate implementation of forms using Reactive Forms with validations**

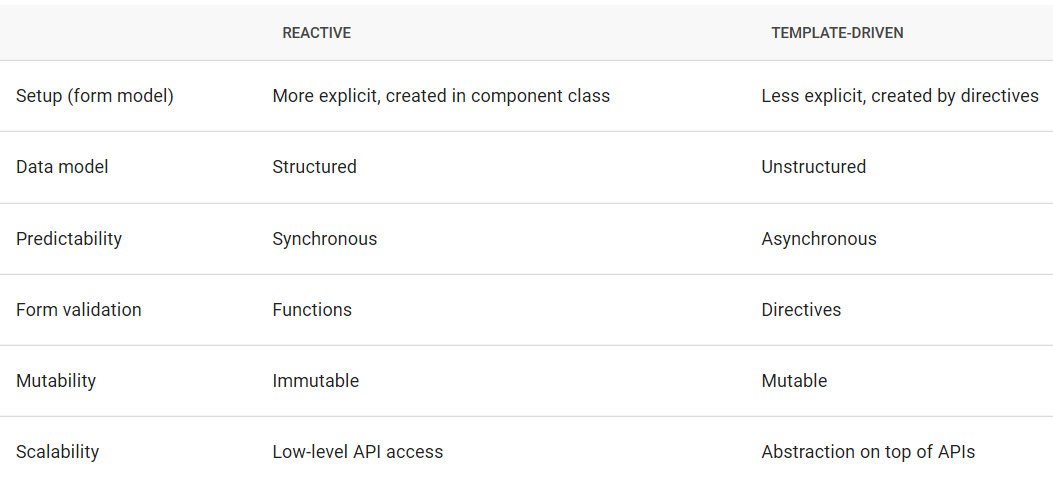
**FOLLOW:**

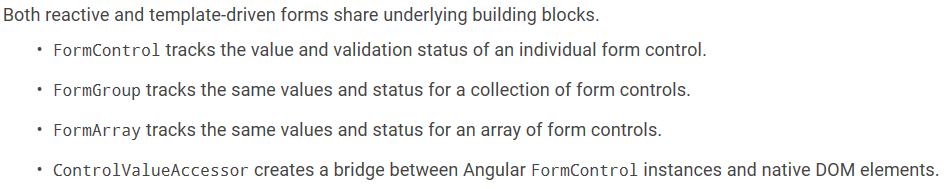
<https://blog.angular-university.io/introduction-to-angular-2-forms-template-driven-vs-model-driven> /

# What is Reactive Form? (Model Driven Form)

*Reactive forms* provide a model-driven approach to handling form inputs whose values change over time. Reactive forms use an explicit and immutable approach to managing the state of a form at a given point in time. Each change to the form state returns a new state, which maintains the integrity of the model between changes.

* Reactive forms are built around observable streams, where form inputs and values are provided as streams of input values, which can be accessed synchronously.
* Reactive forms also provide a straightforward path to testing because you are assured that your data is consistent and predictable when requested. Any consumers of the streams have access to manipulate that data safely.





Reactive form is build completely based on code and validation at runtime based on decision making in code.We need to create component tree.

Downside: More code.

## Advantages and disadvantages of Reactive Forms

You are probably wondering what we gained here. On the surface there is already a big gain: We can now unit test the form validation logic.

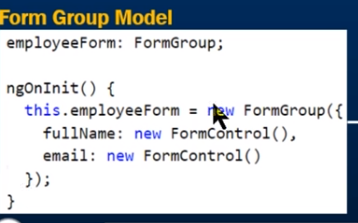
We can do that just by instantiating the class, setting some values in the form controls and perform assertions against the form global valid state and the validity state of each control.

But this is just one possibility. The FormGroup and   
FormControl classes provide an API that allows us to build UIs using a completely different programming style known as Functional Reactive Programming.

## Classes for creation Form Tree

* FormGroup – Will be linked to form creation. One form group can have another form group
* FormControl - Linked to controls within form

Model we create using these two classes is Form Group Model



## Lab: Create skeleton of Book Form to accept book data in HTML.

1. Create a new component Reactbookform.

Ng g c react-book-form.

1. Add this module in **angular.module.ts**

import { ReactBookFormComponent } from './reactbookform/react-book-form.component';

@NgModule({

declarations: [

AppComponent,

HomeComponent,

BookComponent,

BookListComponent,

LoginComponent,

AdminComponent,

ReactBookFormComponent

],

1. Update react-Book-form.ts file to dynamicaaly generate the form controls

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

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

@Component({

selector: 'app-react-book-form',

templateUrl: './react-book-form.component.html',

styleUrls: ['./react-book-form.component.css']

})

export class ReactBookFormComponent implements OnInit {

bookForm: FormGroup;

constructor() { }

ngOnInit() {

this.bookForm=new FormGroup({

id: new FormControl(),

title: new FormControl(),

author: new FormControl(),

price: new FormControl(),

});

}

}

1. Update react-book-form.html

<form class="form-horizontal">

<h3 class="panel-title"> Create a Book </h3>

</div>

<div class="form-group">

<label class="col-sm-2 control-label" for="Id">Id</label>

<div class="col-sm-8">

<input id="Id" type="text" class="form-control">

</div>

</div>

<div class="form-group">

<label class="col-sm-2 control-label" for="Title">Title</label>

<div class="col-sm-8">

<input id="Title" type="text" class="form-control">

</div>

</div>

<div class="form-group">

<label class="col-sm-2 control-label" for="Author">Author</label>

<div class="col-sm-8">

<input id="Author" type="text" class="form-control">

</div>

</div>

<div class="form-group">

<label class="col-sm-2 control-label" for="Price">Price</label>

<div class="col-sm-8">

<input id="Price" type="text" class="form-control">

</div>

</div>

</form>

1. Verify output



## Lab: Add a button to Book Form and Submit display submitted values on console.

Step 1: Add button to react-book-form.html

<div class="form-group">

<button type="submit" class="btn btn-primary ml-4"> Save </button>

</div>

Step 2: Call onSubmit() method to display form data.

<form [formGroup]="bookForm" (ngSubmit)="onSubmit()" class="form-horizontal" >

Step3: Define on Submit()

onSubmit(): void{

console.log(this.bookForm.value);

}

# Reactive Form Validation

Refer” <https://atom-morgan.github.io/reactive-forms-in-angular/>

Refer: <https://blog.mailtrap.io/angular-email-validation/>

1. Validators class to your imports and declare your form controls with arrays instead of simple string values.

**Sync validators:** functions that take a control instance and immediately return either a set of validation errors or null. You can pass these in as the second argument when you instantiate a [FormControl](https://angular.io/api/forms/FormControl).

**Async validators:** functions that take a control instance and return a Promise or Observable that later emits a set of validation errors or null. You can pass these in as the third argument when you instantiate a [FormControl](https://angular.io/api/forms/FormControl).

1. Adding validation to the form is very simple too. Just add the Validators class to your imports and declare your form controls with arrays instead of simple string values.

The first value in the array is the initial form value and the second value is for the validator(s) to use. Notice how multiple validator can be used on the same form control by wrapping them into an array:

The first value in the array is the initial form value and the second value is for the validator(s) to use.

1. Validators function Refer: <https://angular.io/api/forms/Validators>
2. Required
3. RequiredTrue ( To make sure user have check the I agree checkbox before enabling submit button)
4. Email
5. Pattern
6. Min
7. Max
8. minLength
9. maxLength

## Lab: Convert the BookForm into Reactive Form Module to accept book data.

Step 1: Bind the Form group, Form Control defined in ts file with Book form. Make the name must be same

export class ReactBookFormComponent implements OnInit {

bookForm: FormGroup;

//We have bind this bookFrom in html and same with belw controls

constructor() { }

ngOnInit() {

this.bookForm=new FormGroup({

id: new FormControl(),

title: new FormControl(),

author: new FormControl(),

price: new FormControl(),

});

}

<form [formGroup]="bookForm" class="form-horizontal">

<h3 class="panel-title"> Create Book </h3>

<div class="form-group">

<label class="col-sm-2 control-label" for="Id">Id</label>

<div class="col-sm-8">

<input id="Id" formControlName="id" type="text" class="form-control">

</div>

</div>

<div class="form-group">

<label class="col-sm-2 control-label" for="Title">Title</label>

<div class="col-sm-8">

<input id="Title" formControlName="title" type="text" class="form-control">

</div>

</div>

<div class="form-group">

<label class="col-sm-2 control-label" for="Author">Author</label>

<div class="col-sm-8">

<input id="Author" formControlName="author" type="text" class="form-control">

</div>

</div>

<div class="form-group">

<label class="col-sm-2 control-label" for="Price">Price</label>

<div class="col-sm-8">

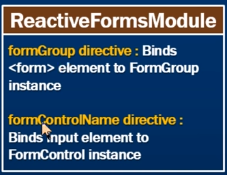
<input id="Price" formControlName="price" type="text" class="form-control">

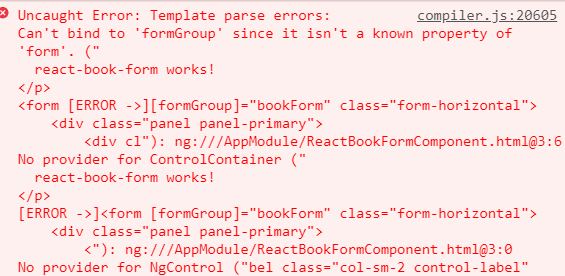
</div>

</div>

</form>

Step 2: Run and analyse. It will raise following error because we have not added Reactive module





Step 3: Add ReactModuleForm

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

@NgModule({

declarations: [

AppComponent,

HomeComponent,

BookComponent,

BookListComponent,

LoginComponent,

AdminComponent,

ReactBookFormComponent

],

imports: [

BrowserModule,

AppRoutingModule,

//HttpModule,

HttpClientModule,

FormsModule,

ReactiveFormsModule

],

Step 2: Run Now

## Lab:Create a Author component to accept author details.

Step 1: Create reactvalidateAuthor > ng g c reactvalidateAuthor

Step 2: Create author.ts file

export class Author{

aid: number;

aFullName: string;

aEmail: string;

aConfirmEmail:string;

constructor(){

}

}

Step 3: Update reactvalidateauthor.ts file

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

export class ReactValidateAuthorComponent implements OnInit {

authForm: FormGroup;

submitted : boolean =false;

constructor( private fb: FormBuilder ){}

ngOnInit() {

this.authForm = this.fb.group({

aid: ['', Validators.required], // This adds a read vertical line in front of field

aFullName: ['', Validators.required],

aEmail: [''],

aConfirmEmail: ['']

});

} //nginit

get f() { return this.authForm.controls; }

onSubmit(): void{

this.submitted=true;

console.log(this.authForm.value);

console.log('Valid?', this. authForm.valid); // true or false

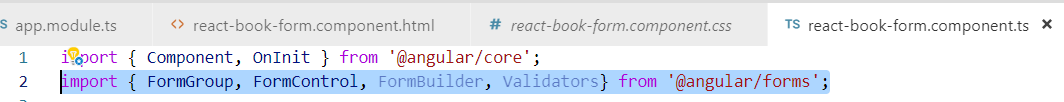
console.log('FullName', this.authForm.value.aFullName);

console.log('Email', this.authForm.value.aEmail);

alert('SUCCESS!! :-)\n\n' + JSON.stringify(this.authForm.value))

}

}

Step4: Include Validators and FormBuilder

Step 5:

<form [formGroup]="authForm" (ngSubmit)="onSubmit()" class="form-horizontal" >

<h3> Create Author </h3>

<div class="form-group">

<label class="col-sm-2 control-label" for="Id">Id</label>

<div class="col-sm-8">

<input id="Id" formControlName="aid" type="text" class="form-control">

</div>

</div>

<div class="form-group" [ngClass]="{ 'has-errors': authForm.get('aFullName').errors }">

<label class="col-sm-2 control-label" for="FullName">Author</label>

<div class="col-sm-8">

<input id="FullName"

name="FullName"

formControlName="aFullName"

type="text"

class="form-control" >

<div \*ngIf="submitted && f.aFullName.errors" class="invalid-feedback"></div>

<div \*ngIf="authForm.get('aFullName').errors?.required ">Full Name is required </div>

</div>

</div>

<div class="form-group">

<label class="col-sm-2 control-label" for="Email">Email</label>

<div class="col-sm-8">

<input id="Email" formControlName="aEmail" type="text" class="form-control">

</div>

</div>

<div class="form-group">

<label class="col-sm-2 control-label" for="cEmail">Confirm Email</label>

<div class="col-sm-8">

<input id="cEmail" formControlName="aConfirmEmail" type="text" class="form-control">

</div>

</div>

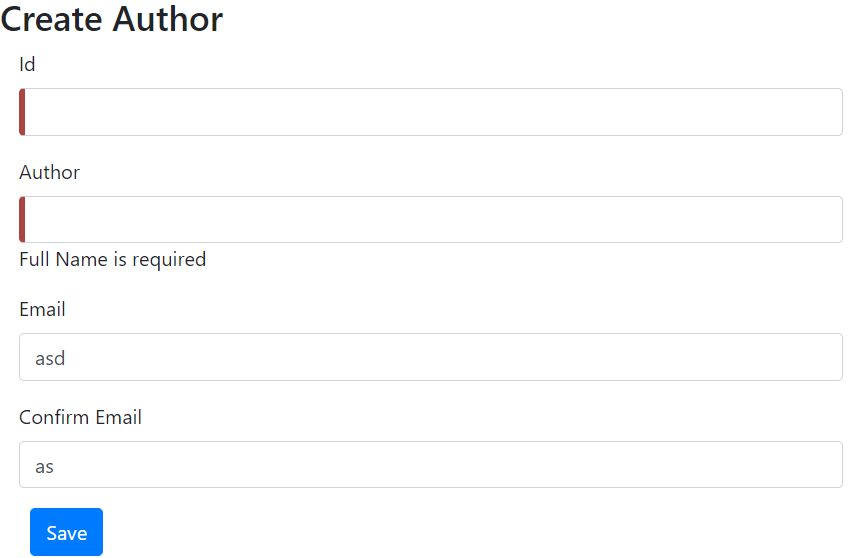
<div class="form-group">

<button type="submit" class="btn btn-primary ml-4"> Save </button>

</div>

</form>

**Note:** 1. Added a getter 'f' as a convenience property to make it easier to access form controls from the template. So for example you can access the confirmPassword field in the template using f.aFullName instead of authForm.controls.aFullName.

? is Safe Operator . we should tell Angular render this div only when we have errors. If we do not use the safe operator, Angular tries to render something that does not exist. This results in crash.  


## Lab: Enhance author page to validate email and email must match confirmEmail

Note: Bootstrap’s class.is-invalid class to the input elements and tie it to the condition that the email fields should be invalid and touched.

<form [formGroup]="authForm" (ngSubmit)="onSubmit()" class="form-horizontal" >

<h3> Create Author </h3>

<div class="form-group">

<label class="col-sm-2 control-label" for="Id">Id</label>

<div class="col-sm-8">

<input id="Id" formControlName="aid" type="text" class="form-control">

</div>

</div>

<div class="form-group" [ngClass]="{ 'has-errors': authForm.get('aFullName').errors }">

<label class="col-sm-2 control-label" for="FullName">Author</label>

<div class="col-sm-8">

<input id="FullName" name="FullName"

formControlName="aFullName"

type="text"

class="form-control" >

<div \*ngIf="submitted && f.aFullName.errors" class="is-invalid"></div>{{submitted}}

<div \*ngIf="authForm.get('aFullName').errors?.required ">Full Name is required </div>

</div>

</div>

<div class="form-group">

<label class="col-sm-2 control-label" for="Email">Email</label>

<div class="col-sm-8">

<input id="Email" formControlName="aEmail" type="text" class="form-control">

<div \*ngIf="submitted && f.aEmail.errors" class="is-invalid"> </div>

<div \*ngIf="f.aEmail.errors?.required">Email is required</div>

<div \*ngIf="f.aEmail.errors.email">Email must be a valid email address</div>

</div>

</div>

</div> <div class="form-group">

<label class="col-sm-2 control-label" for="cEmail">Confirm Email</label>

<div class="col-sm-8">

<input id="cEmail" formControlName="aConfirmEmail" type="text" class="form-control">

<div \*ngIf="submitted && f.aConfirmEmail.errors" class="is-invalid"> </div>

<div \*ngIf="f.aConfirmEmail.errors?.required">Email is required</div>

<div \*ngIf="f.aConfirmEmail.errors?.email">Email must be a valid email address</div>

</div>

</div>

<div class="form-group">

<button type="submit" class="btn btn-primary ml-4"> Save </button>

</div>

</form>

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

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

import { Author } from '../author';

@Component({

selector: 'app-react-validate-author',

templateUrl: './react-validate-author.component.html',

styleUrls: ['./react-validate-author.component.css']

})

export class ReactValidateAuthorComponent implements OnInit {

authForm: FormGroup;

submitted : boolean =false;

constructor( private fb: FormBuilder ){}

ngOnInit() {

this.authForm = this.fb.group({

aid: ['', Validators.required], // This adds a read vertical line in front of field

aFullName: ['', Validators.required],

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

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

});

} //nginit

get f() { return this.authForm.controls; }

onSubmit(): void{

this.submitted=true;

console.log(this.authForm.value);

console.log('Valid?', this. authForm.valid); // true or false

console.log('FullName', this.authForm.value.aFullName);

console.log('Email', this.authForm.value.aEmail);

// stop here if form is invalid

if (this.authForm.invalid) {

return;

}

alert('SUCCESS!! :-)\n\n' + JSON.stringify(this.authForm.value))

}

}

Reference: <https://jasonwatmore.com/post/2019/06/14/angular-8-reactive-forms-validation-example>

Functional Reactive Programming