

Form Module

- An Angular form coordinates a set of data-bound user controls, tracks changes, validates input, and presents errors.
- Angular support two types of forms Template Driven and Reactive
- **Template Driven**
 - These forms are built by creating templates using Angular template syntax with form-specific directives.
 - Create properties that are to be bound to controls in the form.
 - Use two way data binding with ngModel directive.
- **Reactive Forms (Model Driven Forms)**
 - In this we create a tree of Angular form control objects in the component class and bind them to native form control elements in the component template.
 - You create and manipulate form control objects directly in the component class

Steps to create Reactive Forms

- ✓ Import ReactiveFormsModule and include in modules array of Module

- ✓ Create data model
- ✓ Create reactive form component like FormGroup and FormControl
- ✓ Create template -html

➤ Example 1- Template Driven Form

- Step 1: In the application root module import FormsModule from @angular/forms. List FormsModule in imports array of the module

```
TS app.module.ts X
src > app > TS app.module.ts > AppModule
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3
4  import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6  import { FormsModule } from '@angular/forms';
7  @NgModule({
8    declarations: [
9      AppComponent
10   ],
11   imports: [
12     BrowserModule,
13     AppRoutingModule,
14     FormsModule
15   ],
16   providers: [],
17   bootstrap: [AppComponent]
18 })
19 export class AppModule { }
20
```

- Step 2- Generate a component named 'form' using the below command

ng g component form

- Step 3 - Create the form in formcomponent.html as below:-
Specify form name using NgForm directive to the <form> tag.

We have also added function **onSubmit** and assigned **user.value** to it. We need to create the model form controls by adding the **ngModel** directive and the **name** attribute. Thus, wherever we want Angular to access our data from forms, add ngModel to that tag as shown above. Now, if we have to read the email id and password, we need to add the ngModel across it.

```
<form #user="ngForm" (ngSubmit)="onSubmit(user)">
<input type="text" placeholder="emailid" name="emailId" ngModel<br>
del="email"><br>
<input type="text" placeholder="password"><br>
<input type="submit" (click)="onSubmit()" value="Go">
<br>
</form>
```

```
rc > app > form > <> form.component.html > form > input
1  <form #user="ngForm" (ngSubmit)="onSubmit(user.value)">
2    <input type="text" placeholder="emailid" name="emailId" ngModel <br>
3    <input type="text" placeholder="password" name="password" ngModel><br>
4    <input type="submit" value="Go"><br>
5
6    </form>
7
```

- Step 4 - Let us now create the function in the **form.component.ts** and fetch the values entered in the form.

```
onSubmit(user:any)
{
  console.log(user);
  alert('Hello Boss '+user.emailId);
}
```

```

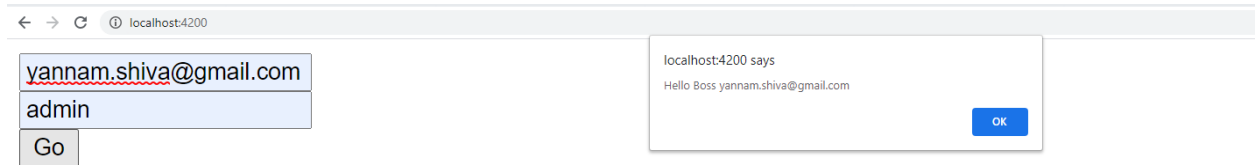
src > app > form > TS form.component.ts > ...
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4      selector: 'app-form',
5      templateUrl: './form.component.html',
6      styleUrls: ['./form.component.css']
7  })
8  export class FormComponent implements OnInit {
9
10     constructor() { }
11
12     ngOnInit(): void {
13     }
14
15     onSubmit(user:any)
16     {
17         console.log(user);
18         alert('Hello Boss '+user.emailId);
19     }
20
21 }
22

```

- Step 5: Next, navigate to your project's folder and run the local development server using the following command:

ng serve

- A local development server will start listening on the `http://localhost:4200/` address. Open your web browser and navigate to the `http://localhost:4200/` address to see your app up and running. The contents from the above html file will be displayed in the browser as shown below –



➤ Example 2- Template Driven Form

- Step 1- Generate a component named 'form1' using the below command

ng g component form1

- Step 2 - Create the form in formcomponent.html as below:-

```
<div class="container">
```

```
<div class="row">
```

```
<div class="form-bg">
```

```
<form #regForm="ngForm" (ngSubmit)="register(regForm)"  
class="text-center">
```

```
<h1>Registration page</h1>
```

```
<br>
```

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

```
<input type="text" class="form-control"  
placeholder="firstname" name="firstname" ngModel>
```

```
</div>
```

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

```
<input type="text" class="form-control"  
placeholder="lastname" name="lastname" ngModel>
```

</div>

<div class="form-group">

**<input type="text" class="form-control"
placeholder="email" name="email" email required ngModel
#email="ngModel">**

</div>

<div class="text-center">

**<button type="submit" class="btn btn-primary" [disabled]
=!regForm.valid> Register</button>**

</div>

</form>

</div>

</div>

</div>

```

src > app > form1 > form1.component.html > ...
1  <div class="container">
2  <div class="row">
3
4      <div class="form-bg">
5      <form #regForm="ngForm" (ngSubmit)="register(regForm)" class="text-center">
6          <h1>Registration page</h1>
7          <br>
8          <div class="form-group">
9              <input type="text" class="form-control" placeholder="firstname" name="firstname" ngModel>
10
11          </div>
12          <div class="form-group">
13              <input type="text" class="form-control" placeholder="lastname" name="lastname" ngModel>
14
15          </div>
16          <div class="form-group">
17              <input type="text" class="form-control" placeholder="email" name="email" email required ngModel #email="ngModel">
18
19          </div>
20          <div class="text-center">
21              <button type="submit" class="btn btn-primary" [disabled] =!regForm.valid> Register</button>
22
23          </div>
24
25
26
27      </form>
28
29
30
31      </div>
32  </div>
33
34
35
36 </div>
37

```

- Step 3 - Let us now create the function in the **form1.component.ts** and fetch the values entered in the form.

```

register(regForm:any){
    var firstname=regForm.controls.firstname.value;
    var lastname=regForm.controls.lastname.value;
    var email=regForm.controls.email.value;
    console.log(regForm);
}

```

```

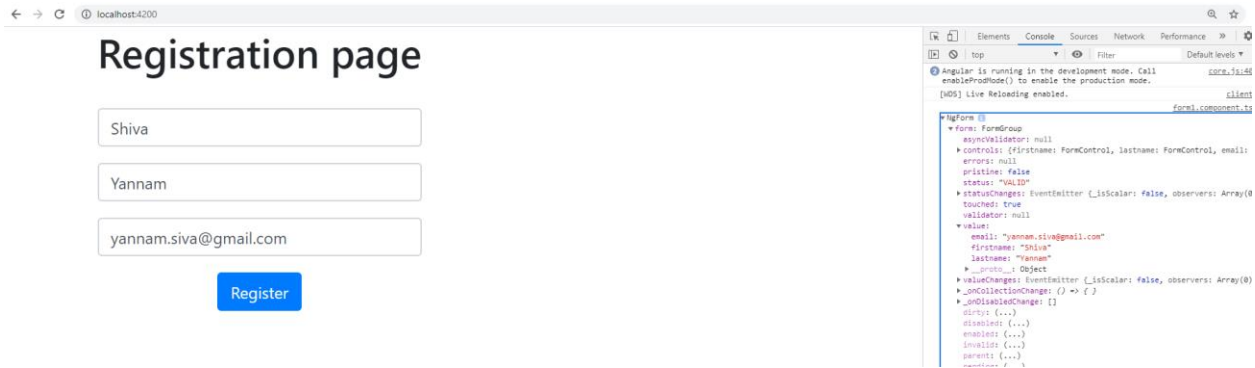
src > app > form1 > TS form1.component.ts > ...
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4      selector: 'app-form1',
5      templateUrl: './form1.component.html',
6      styleUrls: ['./form1.component.css']
7  })
8  export class Form1Component implements OnInit {
9
10     constructor() { }
11
12     ngOnInit(): void {
13     }
14     register(regForm:any){
15         var firstname=regForm.controls.firstname.value;
16         var lastname=regForm.controls.lastname.value;
17         var email=regForm.controls.email.value;
18         console.log(regForm);
19     }
20 }
21
22 }
23

```

- Step 4: Next, navigate to your project's folder and run the local development server using the following command:

ng serve

- A local development server will start listening on the `http://localhost:4200/` address. Open your web browser and navigate to the `http://localhost:4200/` address to see your app up and running. The contents from the above html file will be displayed in the browser as shown below –



- To perform validation on the form level:-

You can use the built-in form validation or also use the custom validation approach. For the firstname, lastname & email id, we have added the following validation parameter – required

For the submit button, we have added disabled in the square bracket, which is given value - !regForm.valid. Thus, if the regForm.valid is not valid, the button will remain disabled and the user will not be able to submit it. Also to provide a custom message printed when we are not entering the email and submitting we can use the code below:-

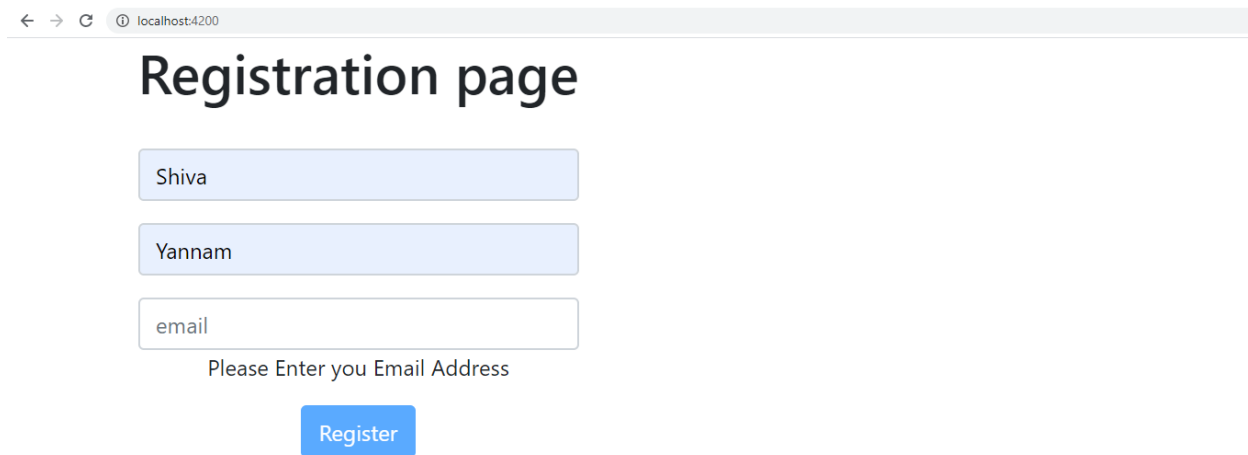
```
<div class="form-group">
```

```
  <input type="text" class="form-control" placeholder="email" name="email" email required ngModel #email="ngModel">
```

```
  <span *ngIf="email.touched && !email.valid">Please Enter you Email Address</span>
```

```
</div>
```

- Browser window:-



A screenshot of a web browser window displaying a registration page. The browser's address bar shows 'localhost:4200'. The page title is 'Registration page'. It features three input fields: the first contains 'Shiva', the second contains 'Yannam', and the third is labeled 'email' with a placeholder text 'Please Enter you Email Address'. Below the email field is a blue 'Register' button.

➤ **Example 1- Reactive Forms**

- Step 1:- Import ReactiveFormsModule and include in modules array of Module

```
TS app.module.ts X  <> app.component.html  TS form1.component.ts  TS app-routing.module.ts
src > app > TS app.module.ts > AppModule
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3
4  import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6  import { FormsModule } from '@angular/forms';
7  import { ReactiveFormsModule } from '@angular/forms';
8  import { FormComponent } from './form/form.component';
9  import { Form1Component } from './form1/form1.component';
10 @NgModule({
11   declarations: [
12     AppComponent,
13     FormComponent,
14     Form1Component
15   ],
16   imports: [
17     BrowserModule,
18     AppRoutingModule,
19     FormsModule,
20     ReactiveFormsModule
21   ],
22   providers: [],
23   bootstrap: [AppComponent, Form1Component]
24 })
25 export class AppModule { }
26
27
```

- Step 2 :- Create a new component named form2

ng g component form2

- Step 3 :- Define the form (formGroup) in a component class:

In our component class we will define the formGroup and also formControls within our formGroup.

formGroup: Entire form will be treated as formGroup and here we can give a name to the formGroup.

formControl: Each 'input' in our form will be mapped with formControl. This formControl is like our input box, select box, radio buttons etc. This form control will contain the value and validations of form field.

We want this form to be there when we load the page so we have to write this formGroup on our TS file's ngOnInit() method and before starting we have to give name or declare the formGroup.

Form2.component.ts

```
export class Form2Component implements OnInit {
```

```
  formdata:FormGroup;
```

```
    emailid: any;
```

```
    password:any;
```

```
    constructor() { }
```

```
  ngOnInit(): void {
```

```
    this.formdata=new FormGroup({
```

```
      emailid:new FormControl("yannam.shiva@gmail.com"),
```

```
      password:new FormControl("rani")
```

```
    })
```

```
  }
```

```
  onClickSubmit(data){
```

```
    this.emailid=data.emailid;
```

```
    this.password=data.password;
```

```
}  
}
```

In above code we have initialized formGroup's object with different form fields(formControl's objects). And these formControls are having one argument that is, values for email as 'yannam.shiva@gmail.com' and password as 'rani' for initializing our form fields. So initially all the form fields would be filled with this values.

```
TS form2.component.ts X  
src > app > form2 > TS form2.component.ts > ...  
1  import { Component, OnInit } from '@angular/core';  
2  import { FormControl, FormGroup } from '@angular/forms';  
3  
4  @Component({  
5    selector: 'app-form2',  
6    templateUrl: './form2.component.html',  
7    styleUrls: ['./form2.component.css']  
8  })  
9  export class Form2Component implements OnInit {  
10   formdata:FormGroup;  
11   emailid: any;  
12   password:any;  
13   constructor() { }  
14  
15   ngOnInit(): void {  
16     this.formdata=new FormGroup({  
17       emailid:new FormControl("yannam.shiva@gmail.com"),  
18       password:new FormControl("rani")  
19     })  
20   }  
21   onClickSubmit(data){  
22     this.emailid=data.emailid;  
23     this.password=data.password;  
24   }  
25  
26  
27 }  
28
```

- Step 4 : Connect this form to your HTML form:

Now, our Form is ready to be bind with our HTML. So moving to HTML file, we can bind the form like below:

```
<form [formGroup]="formdata" (ngSubmit)="onClickSubmit(formdata.value)" >
```

formGroup: Binds the Form to the FormGroup we created in component i.e. with the *signupform*.

ngSubmit: This event will be triggered when we submit the form.

formControlName: Each form fields (inputs) should have formControlName, which is exactly the same as given in TS file.

For example,

```
<input type="text" class="form-control" name="email" placeholder="emailid" formControlName="emailid" >
```

Form2.component.html

```
src > app > form2 > form2.component.html > ...
1  <div class="container">
2    <div class="row">
3
4
5    <div class="form-bg">
6      <form [formGroup]="formdata" (ngSubmit)="onClickSubmit(formdata.value)" >
7        <h1 >Login page</h1>
8        <br>
9        <div class="form-group">
10
11          <input type="text" class="form-control" name="email" placeholder="emailid" formControlName="emailid" >
12
13        </div>
14
15        <div class="form-group">
16
17          <input type="text" class="form-control" name="pass" placeholder="password" formControlName="password" >
18
19        </div>
20        <div class="form-group">
21          <button type="submit" [disabled]="!formdata.valid" class="btn btn-primary" >Log In</button>
22        </div>
23
24        <hr>
25        <h3>Email Entered : {{emailid}}</h3>
26      </form>
27    </div>
28  </div>
29 </div>
30 </div>
31 |
```

- Step 5: And lastly onSubmit() method in our TS file will just submit the form and the value of email id is printed on the browser.

← → localhost:4200

Login page

Email Entered : yannam.shiva@gmail.com

➤ Example 2- Reactive Forms

- Step 1 :- Create a new component named form3

ng g component form3

- Step 2:- Define the form (formGroup) in a component class:

Form3.Component.html

```
export class Form3Component implements OnInit {  
  firstname:string='';  
  lastname:string='';  
  email:string='';  
  signupform:FormGroup;  
  constructor(private formBuilder:FormBuilder) {  
    this.signupform=formBuilder.group({
```



```

    fname:"shiva",
    lname: "yannam",
    email:"yannam.shiva@gmail.com"
  });
}
register(signupform){
  this.firstname=signupform.controls.fname.value;
  this.lastname=signupform.controls.lname.value,
  this.email=signupform.controls.email.value;
  console.log(this.firstname);
  console.log(signupform.controls);
}

ngOnInit(): void {
}

}

```

In above code we have initialized formGroup's object with different form fields(formControl's objects). A constructor of the class is defined using FormBuilder which allows us to forgo of all the **newing** of form group and form controls:

```
TS form3.component.ts X
src > app > form3 > TS form3.component.ts > Form3Component > constructor
1  import { Component, OnInit } from '@angular/core';
2  import { FormGroup, FormControl, FormBuilder } from '@angular/forms';
3
4  @Component({
5    selector: 'app-form3',
6    templateUrl: './form3.component.html',
7    styleUrls: ['./form3.component.css']
8  })
9  export class Form3Component implements OnInit {
10   firstname:string="";
11   lastname:string="";
12   email:string="";
13   signupform:FormGroup;
14   constructor(private formBuilder:FormBuilder) {
15     this.signupform=formBuilder.group({
16       fname:"shiva",
17       lname: "yannam",
18       email:"yannam.shiva@gmail.com"
19     });
20   }
21   register(signupform){
22     this.firstname=signupform.controls.fname.value;
23     this.lastname=signupform.controls.lname.value,
24     this.email=signupform.controls.email.value;
25     console.log(this.firstname);
26     console.log(signupform.controls);
27   }
28
29   ngOnInit(): void {
30   }
31
32 }
33
```

- Step 3 : Connect this form to your HTML form:

Now, our Form is ready to be bind with our HTML. So moving to HTML file, we can bind the form like below:

<form [formGroup]='signupform' (ngSubmit)='register(signupform)' >

```

form3.component.html X
src > app > form3 > form3.component.html > ...
1  <div class="container">
2    <div class="row">
3      <div class="form-bg">
4        <form [formGroup]='signupform' (ngSubmit)="register(signupform)" >
5          <h1 class="text-center">Registration Page</h1>
6          <br>
7          <div class="form-group">
8            <input type="text" class="form-control" placeholder="First Name" name="firstname" formControlName='fname'>
9
10         </div>
11
12         <div class="form-group">
13           <input type="text" class="form-control" placeholder="Last Name" name="lastname" formControlName='lname'>
14
15         </div>
16
17         <div class="form-group">
18           <input type="text" class="form-control" placeholder="Email Id" name="email" formControlName='email'>
19           </div>
20
21         <div class="form-group">
22           <button type="submit" class="btn btn-primary" id="Register">Register</button>
23
24         </div>
25       <br>
26       <br>
27
28       FirstName : {{firstname}}<br>
29       Lastname : {{lastname}}<br>
30       Emal    : {{email}}<br>
31
32     </form>
33
34   </div>
35
36 </div>
37
38 </div>
39
40

```

- Step 4: And lastly register() method in our TS file will just submit the form and the value of firstname, lastname, email id is printed on the browser. Also the value of the form fields and first name can be viewed on the console.

[WDS] Live Reloading enabled.	client:52
shiva	form3.component.ts:25
	form3.component.ts:26
▶ {fname: FormControl, lname: FormControl, email: FormControl}	
>	

```
▼ {fname: FormControl, lname: FormControl, email: FormControl} ⓘ  
  ▼ email: FormControl  
    asyncValidator: null  
    errors: null  
    pristine: true  
    status: "VALID"  
    ► statusChanges: EventEmitter {_isScalar: false, observers: Array(0), ...}  
    touched: false  
    validator: null  
    value: "yannam.shiva@gmail.com"
```

```
▼ fname: FormControl  
  asyncValidator: null  
  errors: null  
  pristine: true  
  status: "VALID"  
  ► statusChanges: EventEmitter {_isScalar: false, observers: Array(0), ...}  
  touched: false  
  validator: null  
  value: "shiva"
```

```
▼ lname: FormControl  
  asyncValidator: null  
  errors: null  
  pristine: true  
  status: "VALID"  
  ► statusChanges: EventEmitter {_isScalar: false, observers: Array(0), ...}  
  touched: false  
  validator: null  
  value: "yannam"
```

Registration Page

FirstName : shiva

Lastname : yannam

Emai : yannam.shiva@gmail.com