**FORMS IN ANGULAR**

Forms are crucial part of web applications which is used to get data input from users.

There are two ways to create forms in angular

1. Template Driven forms

* Used to create small to medium sized form
* Entire form validation is done in HTML Templates

2. Model Driven Forms or Reactive forms

* Used to create large size forms
* Entire form validation is done in component class using FormBuilder and Validator classes

# AngularJS Validation CSS Classes

AngularJS includes following CSS classes to allow styling of form and input controls based on the state of form field.

| CSS Class | Description |
| --- | --- |
| ng-valid | Angular will set this CSS class if the input field is valid without errors. |
| ng-invalid | Angular will set this CSS class if the input does not pass validations. |
| ng-pristine | Angular will set this CSS class if a user has not interacted with the control yet. |
| ng-dirty | Angular will set this CSS class if the value of form field has been changed. |
| ng-touched | Angular will set this CSS class if a user tabbed out from the input control. |
| ng-untouched | Angular will set this CSS class if a user has not tabbed out from the input control. |
| ng-submitted | Angular will set this CSS class if the form has been submitted. |

## Difference between Reactive Forms and Template-driven Forms

|  |  |  |
| --- | --- | --- |
| **Comparison Index** | **Reactive Forms** | **Template-driven Forms** |
| Setup (form model) | Reactive forms are more explicit. They are created in component class. | Template-driven forms are less explicit. They are created by directives. |
| Data model | Structured | Unstructured |
| Predictability | Synchronous | Asynchronous |
| Form validation | Functions | Directives |
| Mutability | Immutable | Mutable |
| Scalability | Low-level API access | Abstraction on top of APIs |

**Template Driven Forms**

**1. Build a template-driven form**

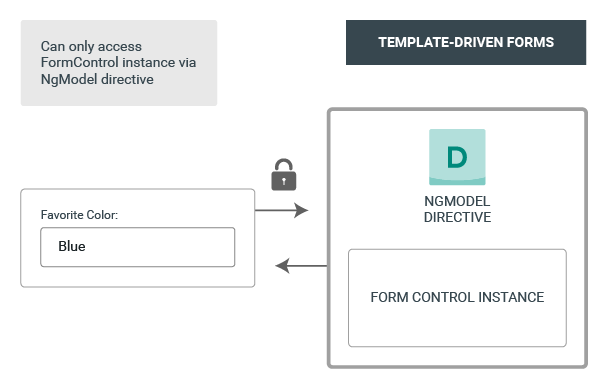
Template-driven forms rely on directives defined in the [FormsModule](https://angular.io/api/forms/FormsModule).

* The [NgModel](https://angular.io/api/forms/NgModel) directive reconciles value changes in the attached form element with changes in the data model, allowing you to respond to user input with input validation and error handling.

What is NgModel?

**ngModel** is a directive which binds input, select and textarea, and stores the required user value in a variable and we can use that variable whenever we require that value.  
It also is used during validations in a form.

* The [NgForm](https://angular.io/api/forms/NgForm) directive creates a top-level [FormGroup](https://angular.io/api/forms/FormGroup) instance and binds it to a <form> element to track aggregated form value and validation status. As soon as you import [FormsModule](https://angular.io/api/forms/FormsModule), this directive becomes active by default on all <form> tags. You don't need to add a special selector.
* Instead of issuing a standard POST we want to call a function on our component instead, to do that we use the ngSubmitdirective and add it to the form element



**Example:**

**Template Driven-Form(app.component.html-normal ngmodel html)**

<form (ngSubmit)="login(form1)" #form1="ngForm">

    <div class="form-row">

      <div class="form-group col-md-6">

        <label for="inputEmail4">Email</label>

        <input type="email" class="form-control" id="inputEmail4" placeholder="Email" ngModel>

      </div>

      <div class="form-group col-md-6">

        <label for="inputPassword4">Password</label>

        <input type="password" class="form-control" id="inputPassword4" ngModel placeholder="Password">

      </div>

    </div>

    <div class="form-group">

      <label for="inputAddress">Address</label>

      <input type="text" class="form-control" id="inputAddress" placeholder="1234 Main St"ngModel>

    </div>

    <div class="form-group">

      <label for="inputAddress2">Address 2</label>

      <input type="text" class="form-control" id="inputAddress2" placeholder="Apartment, studio, or floor" ngModel>

    </div>

    <div class="form-row">

      <div class="form-group col-md-6">

        <label for="inputCity">City</label>

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

      </div>

      <div class="form-group col-md-2">

        <label for="inputZip">Zip</label>

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

      </div>

    </div>

        <button type="submit" class="btn btn-primary">Sign in</button>

        </form>

**Template Driven-Form(app.component.html-normal ngmodelhtml+validation)**

<form (ngSubmit)="login(form1)" #form1="ngForm">

    <div class="form-row">

      <div class="form-group col-md-6">

        <label for="inputEmail4">Email</label>

        <input type="email" #propemail="ngModel" class="form-control" id="inputEmail4" placeholder="Email" ngModel name="email" required minlength="5">

      <span \*ngIf="!propemail.valid && propemail.touched" class="error-block">

         <span \*ngIf="propemail.errors.required">Please provide email id</span>

         <span \*ngIf="propemail.errors.minlength">name must be atleast 5 char long</span>

      </span>

      </div>

      <div class="form-group col-md-6">

        <label for="inputPassword4">Password</label>

        <input type="password" class="form-control" id="inputPassword4" ngModel name="pwd" placeholder="Password">

      </div>

    </div>

    <div class="form-group">

      <label for="inputAddress">Address</label>

      <input type="text" class="form-control" id="inputAddress" placeholder="1234 Main St"ngModel name="address">

    </div>

    <div class="form-row">

      <div class="form-group col-md-6">

        <label for="inputCity">City</label>

        <input type="text" class="form-control" id="inputCity" ngModel name="city">

      </div>

      <div class="form-group col-md-2">

        <label for="inputZip">Zip</label>

        <input type="text" class="form-control" id="inputZip" ngModel name="zip">

      </div>

    </div>

<!--     <button [disabled]="!form1.valid" type="submit" class="btn btn-primary">Sign in</button>-->

        <button type="submit" class="btn btn-primary">Sign in</button>

        </form>

**App.component.ts**

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

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

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

      login(myform:NgForm)

      {

        alert("success")

      }

}

**Appmodule.ts**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

@NgModule({

  declarations: [

    AppComponent

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    FormsModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

**app.component.css**

.ng-invalid .ng-touched

{

    border-color:red;

}

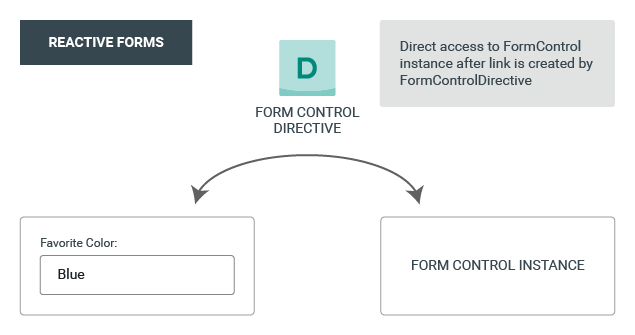
.error-block

{

    color:red;

}

**2. Model Driven Forms or Reactive Forms**



1. Go into the project
2. Install bootstrap (npm install --save bootstrap)
3. Paste the following line in angular.json inside styles

 "node\_modules/bootstrap/dist/css/bootstrap.min.css",

**appcomponent.html**

<form [formGroup]="registrationForm" (ngSubmit)="onSubmit()">

  <div class="form-row">

    <div class="form-group col-md-6">

      <label for="inputEmail4">Email</label>

      <input type="email" formControlName="uname" class="form-control" id="inputEmail4" >

      <span class="error-block"

      \*ngIf="!registrationForm.get('uname').valid && registrationForm.get('uname').touched" >Please valid email</span>

    </div>

    <div class="form-group col-md-6">

      <label for="inputPassword4">Password</label>

      <input type="password"formControlName="pwd" class="form-control" id="inputPassword4" >

    </div>

    <div class="form-group col-md-6">

      <label for="inputPassword4">ConfirmPassword</label>

      <input type="cpassword" formControlName="cpwd"class="form-control" id="inputPassword4" >

    </div>

    <div class="form-group col-md-2">

      <label for="inputZip">Zip</label>

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

    </div>

  </div>

      <button type="submit" class="btn btn-primary" >Sign in</button>

      <br>

      FORMNAME:{{registrationForm.value|json}}<br>

        </form>

**appcomponent.css**

.ng-invalid .ng-touched

{

    border-color:red;

}

.error-block

{

    color:red;

}

**appcomponent.ts**

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

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

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'reactiveform';

  registrationForm:FormGroup;

 constructor(){}

  ngOnInit()

  {

  this.registrationForm=new FormGroup({

  uname:new FormControl('',[Validators.required,Validators.email]),

  zip:new FormControl('',Validators.required),

  pwd:new FormControl('',Validators.required),

  cpwd:new FormControl('',Validators.required)

  },this.passwordmatch);

  }

  passwordmatch(fg:FormGroup):Validators{

    return fg.get('pwd').value===fg.get('cpwd').value? null:

    {notmatched:true};

  }

  onSubmit()

  {

    console.log(this.registrationForm)

   }

}

**Appmoduel.ts**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

**import { FormsModule, ReactiveFormsModule } from '@angular/forms';**

**@NgModule({**

**declarations: [**

**AppComponent**

**],**

**imports: [**

**BrowserModule,**

**AppRoutingModule,**

**FormsModule,**

**ReactiveFormsModule**

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }