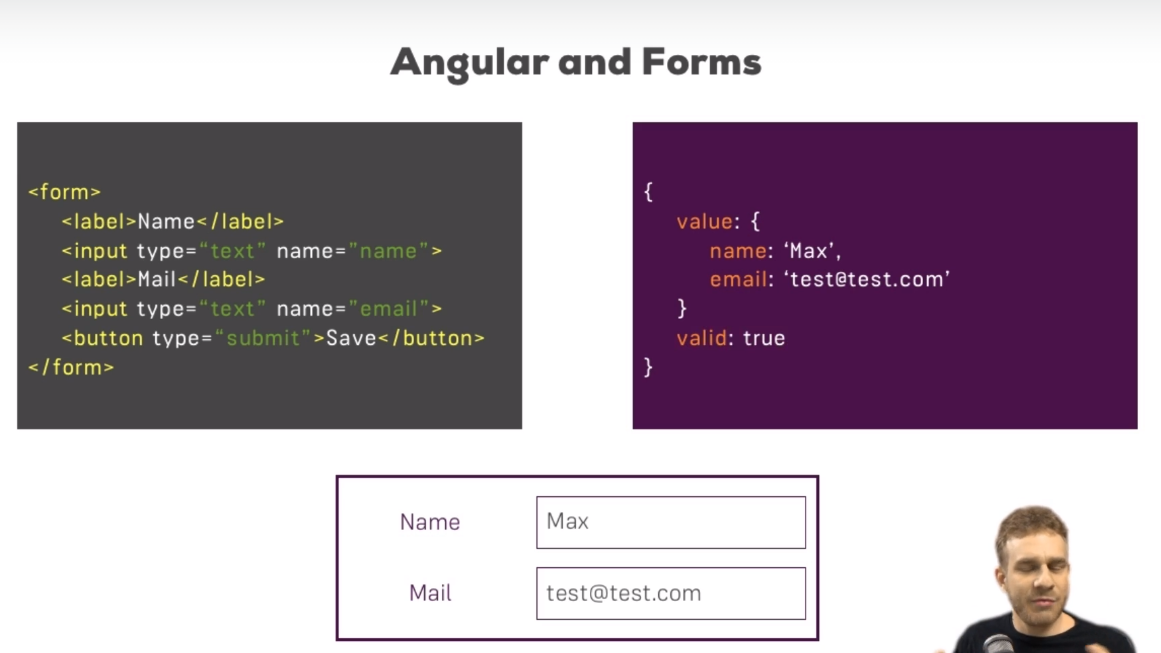
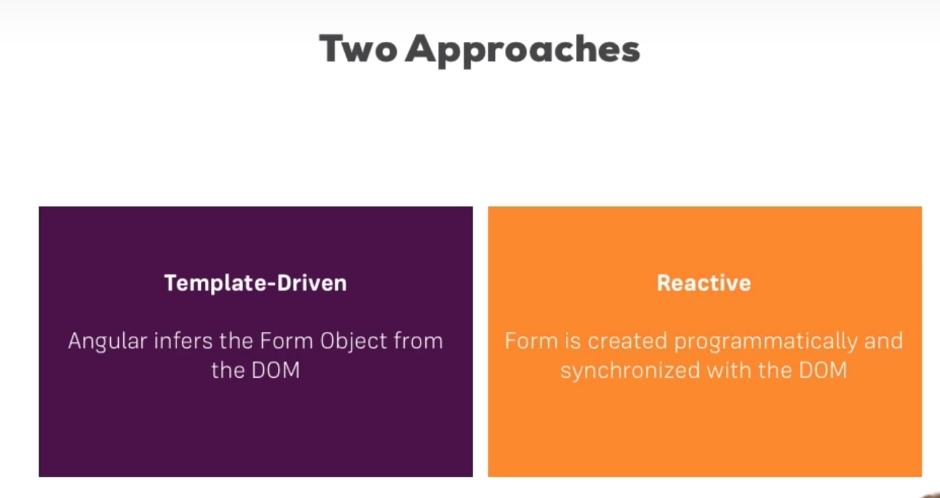
**Section09 Handling Forms in Angular**

**Notes: -**

**1-since the Angular is SPA, there no reload page, so we use form creation in Angular to validate and send our form request to code behind then to http request**

**(With angular it will return the form values in JSON format with sent form status)**





**Lesson02 Creation of Form Template in Registering the Controls**

**Notes: -**

**1-we will using Form Module in the app.module which is responsible to handle template form and registering and submit form values**

**(You can form server as selector for Angular directive, which then creates JavaScript representation of the form for you)**

**(You have to add directive selector to each control in the form template to inform Angular to take this control value when form submit)**

**(By using ng Model that is part in the Forms Module which is directive selector)**

**2-we have to add ngSubmit on the form that refer to the local reference to the form itself and pass it as parameter on the method as NgForm which contain the form state with values property as below**



**We see a lot of property such as controls which see each control with its state such as dirty which indicate that the control is type inside , errors to show validation erros , touched that is clicked but not typed and so on.**

<div class="container">

**<div class="row">**

**<div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">**

**<form (ngSubmit)="onSubmit(f)" #f="ngForm">**

**<div id="user-data">**

**<div class="form-group">**

**<label for="username">Username</label>**

**<input type="text" id="username" class="form-control" ngModel name="username">**

**</div>**

**<button class="btn btn-default" type="button">Suggest an Username</button>**

**<div class="form-group">**

**<label for="email">Mail</label>**

**<input type="email" id="email" class="form-control" ngModel name="email">**

**</div></div>**

**<div class="form-group">**

**<label for="secret">Secret Questions</label>**

**<select id="secret" class="form-control" ngModel name="secret">**

**<option value="pet">Your first Pet?</option>**

**<option value="teacher">Your first teacher?</option>**

**</select></div>**

**<button class="btn btn-primary" type="submit">Submit</button>**

**</form>**

**</div></div></div>**

**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 {**

**suggestUserName() {const suggestedName = 'Superuser';}**

**onSubmit(form:NgForm){**

**console.log(form);}}**

**Lesson03 Pass @ViewChild in Angular**

**Notes:-**

**1-you can use View child to indicate to the local reference in similar way as below**

**<div class="container">**

**<div class="row">**

**<div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">**

**<form (ngSubmit)="onSubmit()" #f="ngForm">**

**<div id="user-data">**

**<div class="form-group">**

**<label for="username">Username</label>**

**<input type="text" id="username" class="form-control" ngModel name="username">**

**</div>**

**<button class="btn btn-default" type="button">Suggest an Username</button>**

**<div class="form-group">**

**<label for="email">Mail</label>**

**<input type="email" id="email" class="form-control" ngModel name="email">**

**</div></div>**

**<div class="form-group">**

**<label for="secret">Secret Questions</label>**

**<select id="secret" class="form-control" ngModel name="secret">**

**<option value="pet">Your first Pet?</option>**

**<option value="teacher">Your first teacher?</option>**

**</select>**

**</div>**

**<button class="btn btn-primary" type="submit">Submit</button>**

**</form>**

**</div></div></div>**

**On code behind**

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

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

**@Component({**

**selector: 'app-root',**

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

**styleUrls: ['./app.component.css']})**

**export class AppComponent {**

**@ViewChild('f') signForm:NgForm;**

**suggestUserName() {**

**const suggestedName = 'Superuser';}**

**onSubmit(){**

**console.log(this.signForm);}}**

**Lesson04 Adding Validation to check User Input**

**Notes:-**

**1-required is default html attribute , but angular detect it and use it to use as validation**

**2-Angular keeps track not only on the form level but also on the form controls level**

**(it will add some class ng-touched ng-valid ng-dirty and when the form is invalid it will ng-invalid if the form control is invlaid)**

**(under this link** [**https://angular.io/api/forms/Validators**](https://angular.io/api/forms/Validators) **, you will see all directivive form validator that can be used on the form)**

**<div class="form-group">**

**<label for="username">Username</label>**

**<input type="text" id="username" class="form-control" ngModel name="username" required>**

**</div>**

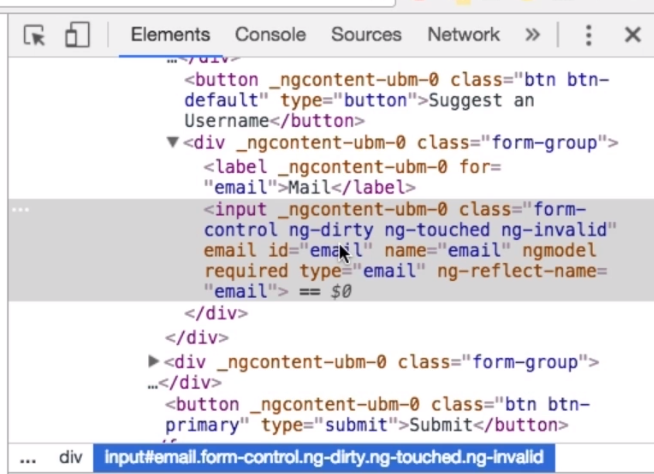
**<button class="btn btn-default" type="button">Suggest an Username</button>**

**<div class="form-group">**

**<label for="email">Mail</label>**

**<input type="email" id="email" class="form-control" ngModel name="email" required email>**

**</div>**



**Lesson05 using the Form State And Set Default Value**

**Notes:-**

**1-we can disable the submit form when the form is invalid**

**2-we can apply style on the form control when its invalid or touched , becouse the angualr template form template automatically adding these classes**

**3-in order to apply validation message to the form control we have to add local reference that refer to ngModel and using this local reference on check isValid and isTouched**

**.container {margin-top: 30px;}**

**/\* it will show when the control is touched and invalid its added these class automatically in Angular Form template \*/**

**input.ng-invalid.ng-touched{**

**border: 1px solid red;}**

**<div class="container">**

**<div class="row">**

**<div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">**

**<form (ngSubmit)="onSubmit()" #f="ngForm">**

**<div id="user-data">**

**<div class="form-group">**

**<label for="username">Username</label>**

**<input type="text" id="username" class="form-control" ngModel name="username" required #userName="ngModel">**

**//this span message will not show unless its invalid and touched**

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

**Please enter valid email</span>**

**</div>**

**<button class="btn btn-default" type="button">Suggest an Username</button>**

**<div class="form-group">**

**<label for="email">Mail</label>**

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

**//this span message will not show unless its invalid and touched**

**<span class="help-block" \*ngIf="!email.valid && email.touched">Please enter valid email</span>**

**</div></div>**

**<div class="form-group">**

**<label for="secret">Secret Questions</label>**

**//we can set default value on the select element**

**<select id="secret" class="form-control" ngModel name="secret" [ngModel]="defaultValue">**

**<option value="pet">Your first Pet?</option>**

**<option value="teacher">Your first teacher?</option>**

**</select>**

**</div>**

**//it will activate once the form is valid**

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

**</form>**

**</div></div></div>**

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

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

**@Component({**

**selector: 'app-root',**

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

**styleUrls: ['./app.component.css']})**

**export class AppComponent {**

**@ViewChild('f') signForm:NgForm;**

**defaultValue:string = "pet";**

**suggestUserName() {**

**const suggestedName = 'Superuser';}**

**onSubmit(){**

**console.log(this.signForm);}}**

**Lesson06 Using Two Way Binding**

**Notes:-**

**1-you can apply two way binding in order to refelect the values in two ways as below**

**<!-- we can apply two way binding from back to view and vice versa -->**

**<div class="form-group">**

**<textarea name="questionAnswer" class="form-control" rows="10" [(ngModel)]="answer"></textarea>**

**</div>**

**<p>You reply: {{answer}}</p>**

**2-the form passing values will be the same**

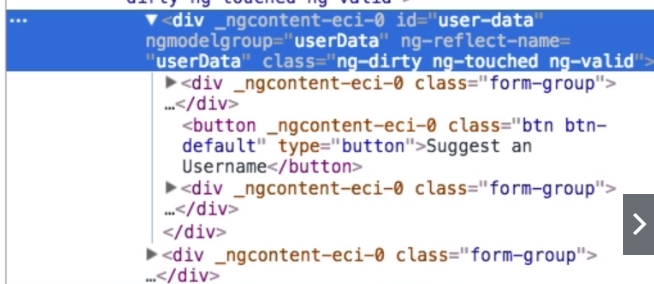
**Lesson07 Grouping Form Controls**

**Notes:-**

**1-we can apply form group as ngModelGroup as Form Group in Reactive form**

**2-we can see that the form ngModelGroup is also include in the user controls and has its own valid which indicate if all the form controls inside it is valid or not**

**3-we can see that the form control in DOM element has its attribute ndmodelgroup=”userData”**



**<div class="container">**

**<div class="row">**

**<div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">**

**<form (ngSubmit)="onSubmit()" #f="ngForm">**

**<!-- we create form group with apply local reference & use validation check for the form group-->**

**<div id="user-data" ngModelGroup="userData" #userData="ngModelGroup">**

**<div class="form-group" >**

**<label for="username">Username</label>**

**<input type="text" id="username" class="form-control" ngModel name="username" required #userName="ngModel">**

**<span class="help-block" \*ngIf="!userName.valid && userName.touched">Please enter valid email</span>**

**</div>**

**<button class="btn btn-default" type="button">Suggest an Username</button>**

**<div class="form-group">**

**<label for="email">Mail</label>**

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

**<span class="help-block" \*ngIf="!email.valid && email.touched">Please enter valid email</span>**

**</div></div>**

**//we check that the NgFormGroup is valid or not**

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

**User Data Group is invalid</span>**

**<div class="form-group">**

**<label for="secret">Secret Questions</label>**

**<select id="secret" class="form-control" ngModel name="secret" [(ngModel)]="defaultValue">**

**<option value="pet">Your first Pet?</option>**

**<option value="teacher">Your first teacher?</option>**

**</select></div>**

**<!-- we can apply two way binding from back to view and vice versa -->**

**<div class="form-group">**

**<textarea name="questionAnswer" class="form-control" rows="10" [(ngModel)]="answer"></textarea>**

**</div>**

**<p>You reply: {{answer}}</p>**

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

**</form>**

**</div></div></div>**

**Lesson08 Handling Radio Buttons**

**Notes:-**

**1-in order to apply gender radio button list**

**<div class="radio" \*ngFor="let gender of genders">**

**<label>**

**<input type="radio" name="gender" ngModel  [value]="gender" required>**

**{{gender}}**

**</label>**

**</div>**

**//on the component code behind we declare array of list as below**

**genders = ['male','female'];**

**Lesson09 Setting and Patching Form Values**

**Notes:-**

**1-there are two way of set values on the forms in tempalte form**

**SetValue : you have to set value for all form controls**

**PatchValue : you have to set value for cutom form control**

**suggestUserName() {**

**const suggestedName = 'Superuser';**

**//you have to set value to all form controls on the form**

**this.signForm.setValue({**

**userData:{**

**username: suggestedName,**

**email:''},**

**secret:'pet',**

**questionAnswer:'',**

**gender:'male'});**

**//you have to set value to specific form control on the form**

**this.signForm.form.patchValue({userData:{**

**username:suggestedName**

**}});}**

**Lesson10 Using Form Data and Reset Form**

**<div class="container">**

**<div class="row">**

**<div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">**

**<form (ngSubmit)="onSubmit()" #f="ngForm">**

**<!-- we create form group with apply local reference  and use validation check for the form group>**

**<div id="user-data" ngModelGroup="userData" #userData="ngModelGroup">**

**<div class="form-group" >**

**<label for="username">Username</label>**

**<input type="text" id="username" class="form-control" ngModel name="username" required #userName="ngModel">**

**<span class="help-block" \*ngIf="!userName.valid && userName.touched">Please enter valid email</span></div>**

**<button class="btn btn-default" type="button" (click)="suggestUserName()">Suggest an Username</button>**

**<div class="form-group">**

**<label for="email">Mail</label>**

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

**<span class="help-block" \*ngIf="!email.valid && email.touched">Please enter valid email</span>**

**</div></div>**

**<span class="help-block" \*ngIf="!userData.valid && userData.touched">User Data Group is invalid</span>**

**<div class="form-group">**

**<label for="secret">Secret Questions</label>**

**<select id="secret" class="form-control" ngModel name="secret" [(ngModel)]="defaultValue">**

**<option value="pet">Your first Pet?</option>**

**<option value="teacher">Your first teacher?</option>**

**</select>**

**</div>**

**<!-- we can apply two way binding from back to view and vice versa -->**

**<div class="form-group">**

**<textarea name="questionAnswer" class="form-control" rows="10" [(ngModel)]="answer">**

**</textarea></div>**

**<div class="radio" \*ngFor="let gender of genders" >**

**<label>**

**<input type="radio" name="gender"  #genderEle="ngModel" ngModel  [value]="gender" required>**

**{{gender}}</label></div>**

**<p>You reply: {{answer}}</p>**

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

**<button class="btn btn-secondary" type="button" (click)="resetForm()">Reset</button>**

**</form></div>**

**<hr>**

**<div class="row" \*ngIf="submitted">**

**<div class="col-xs-12">**

**<h3>Your Data</h3>**

**<p>Username : {{user.username}}</p>**

**<p>Mail :  {{user.email}}</p>**

**<p>Secrete Question: Your fist {{user.answer}}</p>**

**<p>Answer:  {{user.answer}}</p>**

**<p>Gender: {{user.gender}} </p>**

**</div></div></div></div>**

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

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

**@Component({**

**selector: 'app-root',**

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

**styleUrls: ['./app.component.css'],})**

**export class AppComponent {**

**@ViewChild('f') signForm: NgForm;**

**defaultValue: string = 'pet';**

**answer: string = '';**

**genders = ['male', 'female'];**

**user = {**

**username: '',**

**email: '',**

**secretQuestion: '',**

**answer: '',**

**gender: '',};**

**submitted:boolean = false;**

**suggestUserName() {**

**const suggestedName = 'Superuser';**

**this.signForm.setValue({**

**userData: {**

**username: suggestedName,**

**email: '',},**

**secret: 'pet',**

**questionAnswer: '',**

**gender: 'male',});**

**this.signForm.form.patchValue({**

**userData: {username: suggestedName,},});}**

**onSubmit() {**

**console.log(this.signForm);**

**this.submitted = true;**

**this.user.username = this.signForm.value.userData.username;**

**this.user.email = this.signForm.value.userData.email;**

**this.user.secretQuestion = this.signForm.value.secret;**

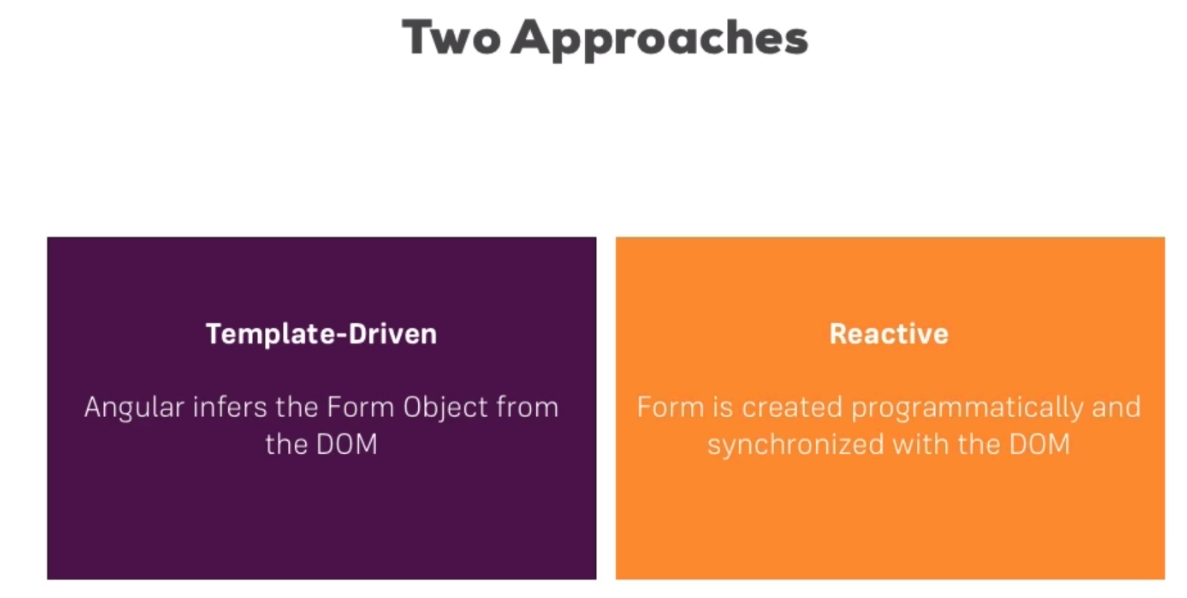
**this.user.answer = this.signForm.value.questionAnswer;**

**this.user.gender = this.signForm.value.gender;}**

**resetForm(){this.signForm.reset();}}**

**Reactive Form Approach**

**Notes:-**



**1-just like the NgForm , there is form group which hold group of form controls**

**2-we have to import ReactiveFormModule in the app.module.ts**

**3-there are 4 major types in reactive form**

**A-Form Builder**

**B-Form Group**

**C-Form Array**

**D-Form Control**

**4-can contain form gorup with inner form group or form array and vice versa**

**5-all validation checks happen on the backend except the show of span error messages**

**Lesson11 Configure Reactive Form and use set validation and user inner form group**

**Steps:-**

**1-on the app.module we inject of the ReactiveFormModule**

**2-on the ReactiveForm.html**

**<div class="container">**

**<div class="row">**

**<div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">**

**//we apply ngSubmit to hit to target function on component**

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

**<div formGroupName="userData">**

**<div class="form-group" >**

**<label for="username">Username</label>**

**<input type="text" class="form-control" formControlName="username">**

**<!- we make check that from the getter get inner form group then get form control username**

**and check status -->**

**<span \*ngIf="!getControls.userData.get('username').valid && getControls.userData.get('username').touched">username is required</span></div>**

**<div class="form-group">**

**<label for="email">Mail</label>**

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

**<!-- we make check that from the getter get inner form group then get form control email**

**and check status -->**

**<span \*ngIf="!getControls.userData.get('email').valid && getControls.userData.get('email').touched">email is required</span>**

**</div></div>**

**<div class="radio" \*ngFor="let gender of genders" >**

**<label>**

**<input type="radio" name="gender"  [value]="gender" formControlName="gender">**

**{{gender}}</label></div>**

**<button class="btn btn-primary" type="submit">Submit</button>**

**<!-- <button class="btn btn-secondary" type="button" (click)="resetForm()">Reset</button> -->**

**</form>**

**</div><hr></div></div>**

**3-on the ReactiveForm.ts**

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

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

**@Component({**

**selector: 'app-reactive-form',**

**templateUrl: './reactive-form.component.html',**

**styleUrls: ['./reactive-form.component.css'],})**

**export class ReactiveFormComponent implements OnInit {**

**genders = ['male', 'female'];**

**signForm: FormGroup;**

**ngOnInit(): void {**

**this.signForm = new FormGroup({**

**//first param is initial value , second param is vvalidator**

**userData: new FormGroup({**

**username: new FormControl(null, [Validators.required]),**

**email: new FormControl(null, [Validators.required, Validators.email]),}),**

**gender: new FormControl('male'),});}**

**onSubmit(){console.log(this.signForm);}**

**//this getter get the control that inside the form group and inner form groups**

**get getControls(): any {return this.signForm['controls'];}}**

**Lesson12 Form Array in Reactive Form**

**Notes:-**

**1-you can declare form array which can hold array of form controls or inner form group as below**

**<div class="container">**

**<div class="row">**

**<div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">**

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

**<div formGroupName="userData">**

**<div class="form-group" >**

**<label for="username">Username</label>**

**<input type="text" class="form-control" formControlName="username">**

**<!-- we make check that from the getter get inner form group then get form control username**

**and check status -->**

**<span \*ngIf="!getControls.userData.get('username').valid &&**

**getControls.userData.get('username').touched">username is required</span></div>**

**<div class="form-group">**

**<label for="email">Mail</label>**

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

**<!-- we make check that from the getter get inner form group then get form control**

**email and check status -->**

**<span \*ngIf="!getControls.userData.get('email').valid &&**

**getControls.userData.get('email').touched">email is required</span></div></div>**

**<div class="radio" \*ngFor="let gender of genders" >**

**<label><input type="radio" name="gender"  [value]="gender" formControlName="gender">**

**{{gender}}</label></div>**

**<div formArrayName="hoppies">**

**<h4>You Hobbies</h4>**

**<button class="btn btn-default" type="button" (click)="onAddHobby()">Addd Hobby</button>**

**//we need index in order to use it in declare form control name as below**

**<div class="form-group" \*ngFor="let hoppy of getControls.hoppies.controls;let i=index">**

**<input type="text" class="form-control" [formControlName]="i" />**

**</div></div>**

**<button class="btn btn-primary" type="submit">Submit</button>**

**</form>**

**</div><hr></div></div>**

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

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

**@Component({**

**selector: 'app-reactive-form',**

**templateUrl: './reactive-form.component.html',**

**styleUrls: ['./reactive-form.component.css'],})**

**export class ReactiveFormComponent implements OnInit {**

**genders = ['male', 'female'];**

**signForm: FormGroup;**

**ngOnInit(): void {**

**this.signForm = new FormGroup({**

**//first param is initial value , second param is vvalidator**

**userData: new FormGroup({**

**username: new FormControl(null, [Validators.required]),**

**email: new FormControl(null, [Validators.required, Validators.email]),}),**

**gender: new FormControl('male'),**

**hoppies: new FormArray([]),});}**

**onSubmit() {console.log(this.signForm);}**

**get getControls(): any {return this.signForm['controls'];}**

**onAddHobby() {**

**//we declare form control and using it to push to the form group > form array as below**

**const ctrl = new FormControl(null,[Validators.required]);**

**(<FormArray>this.signForm.get('hoppies')).push(ctrl);}}**

**Lesson13 Custom Validator in Angular**

**Notes:-**

**1-Validator is just function which get executed by angular automatically when it chekcs the validaity of the form control and checks the validity whenever you change the form control**

**2-we see if the black list found on custom validator it will add to the errors proeprty called nameIsForbidden:true**

**forbiddenUserName = ['Chris', 'Ana'];**

**ngOnInit(): void {**

**this.signForm = new FormGroup({**

**//first param is initial value , second param is vvalidator**

**userData: new FormGroup({**

**//we have to bind the validator the the form control as below**

**username: new FormControl(null, [Validators.required,this.forbiddenNames.bind(this)]),**

**email: new FormControl(null, [Validators.required, Validators.email]),}),**

**gender: new FormControl('male'),**

**hoppies: new FormArray([]),});}**

**//this custom validator take form control as parameter and we return object of boolean value / null**

**//and using on the validator of the form control**

**forbiddenNames(control: FormControl): { [s: string]: boolean } | null {**

**if (this.forbiddenUserName.indexOf(control.value) !== -1) {**

**return { nameIsForbidden: true };}**

**return null;}**

3-we can check for the error message display on the view as below

**<div class="form-group" >**

**<label for="username">Username</label>**

**<input type="text" class="form-control" formControlName="username">**

**<!-- we make check that from the getter get inner form group then get form control**

**username and check status -->**

**<span \*ngIf="!getControls.userData.get('username').valid &&**

**getControls.userData.get('username').touched">**

**<span \*ngIf="getControls.userData.get('username').errors['required']">**

**username is required**

**</span>**

**<span \*ngIf="getControls.userData.get('username').errors['nameIsForbidden']">**

**username is forbidden**

**</span>**

**</span>**

**</div>**

**Lesson14 Creating Custom Async Validator**

**Notes:-**

**1-to make async validator to make check validator to the server side**

**2-on the app > validator > create the interface Asyncvalidator and MyEmailValdiator**

**import { AbstractControl, ValidationErrors, Validator } from "@angular/forms";**

**import { Observable } from "rxjs";**

**interface AsyncValiator extends Validator {**

**validate(control: AbstractControl): Promise<ValidationErrors | null> | Observable<ValidationErrors | null>**

**// inherited from forms/Validator**

**validate(control: AbstractControl): ValidationErrors | null}**

**import { Injectable } from '@angular/core';**

**import { AbstractControl, AsyncValidator, ValidationErrors } from '@angular/forms';**

**import { Observable, of } from 'rxjs';**

**import { catchError, map } from 'rxjs/operators';**

**@Injectable({ providedIn: 'root' })**

**export class MyEmailValidator implements AsyncValidator{**

**constructor() {}**

**validate(control: AbstractControl): Promise<ValidationErrors | null> | Observable<ValidationErrors | null> {**

**const promise = new Promise<any>((resolve,reject) => {**

**setTimeout(() => {**

**if(control.value == "test@test.com"){**

**resolve({'emailIsForbidden':true});}**

**else{resolve(null);}},1500);});**

**return promise;}}**

**on the recive-form.component.ts we call the**

**constructor(private myEmailValidator:MyEmailValidator){}**

**ngOnInit(): void {**

**this.signForm = new FormGroup({**

**//first param is initial value , second param is validator , third is async validator**

**userData: new FormGroup({**

**//we have to bind the validator the the form control as below**

**username: new FormControl(null, [Validators.required,this.forbiddenNames.bind(this)]),**

**email: new FormControl(null, [Validators.required, Validators.email],this.myEmailValidator.validate.bind(this.myEmailValidator)),}),**

**gender: new FormControl('male'),**

**hoppies: new FormArray([]),});}**

**on the receive-form.html we set the following error message as below**

**<div class="form-group">**

**<label for="email">Mail</label>**

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

**<!-- we make check that from the getter get inner form group then get form control email**

**and check status -->**

**<span \*ngIf="!getControls.userData.get('email').valid**

**&& getControls.userData.get('email').touched">**

**<span \*ngIf="getControls.userData.get('email').errors['required']">email is required</span>**

**<span \*ngIf="getControls.userData.get('email').errors['emailIsForbidden']">email is forbidden**

**</span></span></div>**

**Lesson15 Reacting to status changes or value changes and SetValue and patchValue and ResetForm**

**<div class="container">**

**<div class="row">**

**<div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">**

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

**<div formGroupName="userData">**

**<div class="form-group" >**

**<label for="username">Username</label>**

**<input type="text" class="form-control" formControlName="username">**

**<!-- we make check that from the getter get inner form group then get form control username**

**and check status -->**

**<span \*ngIf="!getControls.userData.get('username').valid &&**

**getControls.userData.get('username').touched">**

**<span \*ngIf="getControls.userData.get('username').errors['required']">**

**username is required**

**</span>**

**<span \*ngIf="getControls.userData.get('username').errors['nameIsForbidden']">**

**username is forbidden**

**</span>**

**</span></div>**

**<button class="btn btn-default" type="button" (click)="suggestUserName()">Suggest an Username</button>**

**<div class="form-group">**

**<label for="email">Mail</label>**

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

**<!-- we make check that from the getter get inner form group then get form control**

**email and check status -->**

**<span \*ngIf="!getControls.userData.get('email').valid &&**

**getControls.userData.get('email').touched">**

**<span \*ngIf="getControls.userData.get('email').errors['required']">**

**email is required**

**</span>**

**<span \*ngIf="getControls.userData.get('email').errors['emailIsForbidden']">**

**email is forbidden</span></span></div></div>**

**<div class="radio" \*ngFor="let gender of genders" >**

**<label>**

**<input type="radio" name="gender"  [value]="gender" formControlName="gender">**

**{{gender}}**

**</label>**

**</div>**

**<div formArrayName="hoppies">**

**<h4>You Hobbies</h4>**

**<button class="btn btn-default" type="button" (click)="onAddHobby()">Addd Hobby</button>**

**<div class="form-group" \*ngFor="let hoppy of getControls.hoppies.controls;let i=index">**

**<input type="text" class="form-control" [formControlName]="i" />**

**</div></div>**

**<button class="btn btn-primary" type="submit">Submit</button>**

**<button class="btn btn-secondary" type="button" (click)="resetForm()">Reset</button>**

**</form></div>**

**<hr></div></div>**

**On the code behind**

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

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

**import { MyEmailValidator } from 'src/app/validators/my-email-validator';**

**@Component({**

**selector: 'app-reactive-form',**

**templateUrl: './reactive-form.component.html',**

**styleUrls: ['./reactive-form.component.css'],})**

**export class ReactiveFormComponent implements OnInit {**

**genders = ['male', 'female'];**

**signForm: FormGroup;**

**forbiddenUserName = ['Chris', 'Ana'];**

**constructor(private myEmailValidator:MyEmailValidator){}**

**ngOnInit(): void {**

**this.signForm = new FormGroup({**

**//first param is initial value , second param is validator , third is async validator**

**userData: new FormGroup({**

**//we have to bind the validator the the form control as below**

**username: new FormControl(null, [Validators.required,this.forbiddenNames.bind(this)]),**

**email: new FormControl(null, [Validators.required, Validators.email],this.myEmailValidator.validate.bind(this.myEmailValidator)),}),**

**gender: new FormControl('male'),**

**hoppies: new FormArray([]),});**

**//you can register event listener for the status changes or value changes**

**this.signForm.statusChanges.subscribe((val) => {**

**console.log(`status changes to ${val}`);});**

**this.signForm.valueChanges.subscribe((val) => {**

**console.log(`value changes to ${val}`);});}**

**onSubmit() {console.log(this.signForm);}**

**get getControls(): any {return this.signForm['controls'];}**

**onAddHobby() {**

**const ctrl = new FormControl(null, [Validators.required]);**

**(<FormArray>this.signForm.get('hoppies')).push(ctrl);}**

**//this custom validator take form control as parameter and we return object of boolean value or nu**

**//and using on the validator of the form control**

**forbiddenNames(control: FormControl): { [s: string]: boolean } | null {**

**if (this.forbiddenUserName.indexOf(control.value) !== -1) {**

**return { nameIsForbidden: true };}**

**return null;}**

**suggestUserName() {**

**const suggestedName = 'Superuser';**

**this.signForm.setValue({**

**userData:{**

**username:suggestedName,**

**email:'max@test.com'},**

**gender:'male',**

**hoppies:[]});**

**this.signForm.patchValue({**

**userData:{**

**username:suggestedName,**

**email:'max@test.com'}});}**

**resetForm(){this.signForm.reset();}}**