**Form Validator & getting Data without Form**

In app.module.ts

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

…

imports: [

...

FormsModule,

ReactiveFormsModule

...

],

**First Way to validate and get data two way binding only in Html Or template driven form**

**<form #f="ngForm" (ngSubmit)="onSubmitt(f.value)" >**

**<input type="text" placeholder="Enter Email" name="email" [(ngModel)] ="ValueEmail" #validateEmail="ngModel" email required >**

**<input type="password" name="password" placeholder="Enter Password" [(ngModel)] ="ValuePass" #validatePass="ngModel" required minlength="6" >**

**<input type="url" placeholder="Enter site URL" name="site" pattern="https?://.+" [(ngModel)] ="modelsite" #site="ngModel" required url minlength="11">**

**<!-- \*\*\*\*\*\*\*\* Validataion mesg \*\*\*\*\*\*\*\*\*\*\*\*\*\* -->**

**<div style="color: red;**

**text-align: center;" \*ngIf="validateEmail.errors?.required && validateEmail.touched">\* Email is required</div>**

**<div style="color: red;**

**text-align: center;" \*ngIf="validateEmail.errors?.email && validateEmail.touched">\* Email is Must be a valid</div>**

**<div \*ngIf="validatePass.errors?.required && validatePass.touched ">Password is required </div>**

**<div \*ngIf="validatePass.errors?.minlength ">Password Must be a 6 Digit</div>**

**<div \*ngIf="site.errors?.required && site.touched || site.errors?.pattern ">Site is required and Must be a valid</div>**

**{{modelsite}}{{ValueEmail}}**

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

**</form>**

**\*\*note**

**1.#validateEmail="ngModel" only for validaton**

**2.[(ngModel)] ="ValueEmail" get data only using two way binding**

**Form validation using Reactive form/ model driven method**

**First using form builder**

**<form [formGroup]="validations\_form" (ngSubmit)="onSubmit()" class="form\_section">**

**Email: <input formControlName="email" /> <br />**

**<div class="error\_msgs">**

**<p**

**\*ngIf="validations\_form.controls.email.invalid && validations\_form.controls.email.touched && validations\_form.controls.email.hasError('titleblank') ">**

**Please**

**enter job title</p>**

**</div>**

**{{validations\_form.controls.email.value}}**

**Mobile: <input formControlName="mobile" /> <br />**

**<div class="error\_msgs">**

**<p \*ngIf="validations\_form.controls.mobile.invalid && validations\_form.controls.mobile.touched ">**

**Please**

**enter Mobile no</p>**

**</div>**

**{{validations\_form.controls.mobile.value}}**

**<button type="submit" [disabled]="!validations\_form.valid">Save</button>**

**<button type="reset" (click)="revert()">Revert</button>**

**</form>**

**And in ts**

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

**validations\_form;**

**constructor(**

**public navCtrl: NavController,**

**public formBuilder: FormBuilder,**

**public navParams: NavParams) {**

**this.validations\_form = this.formBuilder.group({**

**mobile: ['434', Validators.required],**

**email: ['', ,this.validateEmail]**

**});**

**validateEmail(control: FormControl) {**

**return new Promise(resolve => {**

**if (control.value.trim() == "") {**

**resolve({ titleblank: true });**

**} else {**

**resolve(null);**

**}**

**});**

**}**

In App module

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

…

imports: [

ReactiveFormsModule,

IonicModule.forRoot(MyApp)

],

Using Form Group

<form [formGroup]="validations\_form" (ngSubmit)="onSubmit()" class="form\_section">

Email: <input formControlName="email" /> <br />

<div class="error\_msgs">

<p

\*ngIf="validations\_form.controls.email.invalid && validations\_form.controls.email.touched ">

Please

enter job title</p>

</div>

{{validations\_form.controls.email.value}}

Mobile: <input formControlName="mobile" /> <br />

<div class="error\_msgs">

<p \*ngIf="validations\_form.controls.mobile.invalid && validations\_form.controls.mobile.touched ">

Please

enter Mobile no</p>

</div>

{{validations\_form.controls.mobile.value}}

<button type="submit" [disabled]="!validations\_form.valid">Save</button>

<button type="reset" (click)="revert()">Revert</button>

</form>

In ts

validations\_form;

constructor(

public navCtrl: NavController,

public formBuilder: FormBuilder,

public navParams: NavParams) {

this.validations\_form = new FormGroup({

mobile: new FormControl('434', Validators.required),

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

});

}

* All import same for reactive form

**Just Getting data from form**

In html

<form #userlogin = "ngForm" (ngSubmit)="onSubmitt(userlogin.value)">

<input type="text" name="emailid" placeholder="Enter Email Id" ngModel><br><br>

<input type="password" name="passid" placeholder="Enter password" ngModel><br><br>

<button type="submit" >Submit</button>

</form>

\*\* ngForm is Directory and userlogin.value passing data by onSubmitt method and ngModel getting data by name

In ts File

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

…

onSubmitt(value){

console.log("form Data0",value);

console.log("form Data",value.emailid);

}

**Getting Data and validate type**

In html file

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

<input type="text" name="emailid" placeholder="Enter Email Id" formControlName="emailid"><br><br>

<input type="password" name="passid" placeholder="Enter password" formControlName="passwd"><br><br>

<button type="submit" [disabled] = "!formdata.valid" >Submit</button>

</form>

\*\* [formGroup] is class binding and formControlName binding data

In Ts File

…

formdata;

…

ngOnInit() {

this.formdata = new FormGroup({

emailid: new FormControl("", Validators.compose([

Validators.required,

Validators.pattern("[^ @]\*@[^ @]\*")

])),

passwd: new FormControl("")

});

}

**Getting Data Without Using Form (two way Binding)**

In html

<input [(ngModel)]="testInput" placeholder="Enter a Value Name">

<button (click)="consoleClass()"> click to call</button>

{{testInput}}

In TS

consoleClass(){

console.log(this.testInput);

}