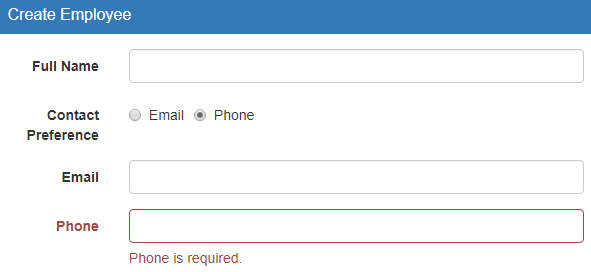
**Lesson14 Angular Dynamically adding or removing form control validators in reactive form**

**Notes:-**

**1-we apply addition or remove validation on the form control validator when write the phone control or remove control value**



**Expected Behavior:-**

**A-Add the "required" validator to the Phone form control when the user selects "Phone" as their contact preference**

**B-On the other hand, remove the "required" validator from the Phone form control, when the user selects "Email" as their contact preference**

**C-So on the "Phone" form control, we have to dynamically add or remove the required validator function**

**This can be very easily achieved using the following 3 functions   
(The three method are available in the abstract class 🡺 available Form Control)**

**1-setValidators()**

**2-clearValidators()**

**3-updateValueAndValidity()**

**Steps:-**

**1-In the employees-comp.component.html we reach to:-**

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

<div class="panel panel-primary">

<div class="panel-heading"><h3 class="panel-title">

Create Employee</h3></div>

<div class="panel-body">

<div class="form-group">

<label class="col-sm-2 control-label" for="fullName">Full Name</label>

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

<input id="fullName" type="text" class="form-control" formControlName="fullName" (blur)="logValidationErrors()">

<span class="help-block" \*ngIf="formErrors.fullName">

{{formErrors.fullName}}</span></div></div>

<div class="form-group">

<label class="col-md-2 control-label">Contact Preference</label>

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

//we see that we assign the same form Control Name to the 2 radio button

<label class="radio-inline"><input type="radio" value="email" formControlName="contactPreference">Email</label>

<label class="radio-inline">

<input type="radio" value="phone" formControlName="contactPreference">

Phone</label></div></div>

<!-- Email input element -->

<div class="form-group" [ngClass]="{'has-error': formErrors.email}">

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

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

<input id="email" type="text" class="form-control" formControlName="email" (blur)="logValidationErrors()">

<span class="help-block" \*ngIf="formErrors.email">

{{formErrors.email}}

</span>

</div>

</div>

<!-- Phone input element -->

<div class="form-group" [ngClass]="{'has-error': formErrors.phone}">

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

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

<input id="phone" type="text" class="form-control" formControlName="phone" (blur)="logValidationErrors()">

<span class="help-block" \*ngIf="formErrors.phone">

{{formErrors.phone}}

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

<div class="panel-footer">

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

<button class="btn btn-primary" type="button" (click)="onLoadDataClick()">Load Data</button>

</div>

</div>

</form>

**2-In the employees-comp.component.ts we apply the following code**

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

// Import FormGroup and FormControl classes

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

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

@Component({

selector: 'app-employees-comp',

templateUrl: './employees-comp.component.html',

styleUrls: ['./employees-comp.component.css']

})

export class EmployeesCompComponent implements OnInit {

// Include phone property

formErrors = {

'fullName': '',

'email': '',

'phone': '',

'skillName': '',

'experienceInYears': '',

'proficiency': ''

};

// Include required error message for phone form control

validationMessages = {

'fullName': {

'required': 'Full Name is required.',

'minlength': 'Full Name must be greater than 2 characters',

'maxlength': 'Full Name must be less than 10 characters.',},

'email': {

'required': 'Email is required.',

'emailDomain': 'Email domian should be prgaimtech.com'},

'phone': {

'required': 'Phone is required.'},

'skillName': {

'required': 'Skill Name is required.',},

'experienceInYears': {

'required': 'Experience is required.',},

'proficiency': {

'required': 'Proficiency is required.',},

};

employeeForm:FormGroup;

constructor(private fb:FormBuilder){}

ngOnInit() {

// Include FormControls for contactPreference, email & phone

// contactPreference has email as the default value

this.employeeForm = this.fb.group({

fullName: ['', Validators.compose([Validators.required, Validators.minLength(2), Validators.maxLength(10)])],

contactPreference: ['email'],

email: ['', Validators.required],

phone: [''],

skills: this.fb.group({

skillName: ['', Validators.required],

experienceInYears: ['', Validators.required],

proficiency: ['', Validators.required]}),});

this.employeeForm.valueChanges.subscribe((data) => {

this.logValidationErrors(this.employeeForm);});

//we apply the value changes event on the control name on the function

this.employeeForm.get('contactPreference').valueChanges.subscribe((data) =>{this.onContactPrefernceChange(data);})}

// If the Selected Radio Button value is "phone", then add the

// required validator function otherwise remove it

onContactPrefernceChange(frmGroup:string){

debugger;

const obj = this.employeeForm.get('phone');

if(frmGroup === 'phone'){

obj.setValidators(Validators.required);}

else{obj.clearValidators();}

obj.updateValueAndValidity();}

logValidationErrors(group: FormGroup = this.employeeForm): void {

Object.keys(group.controls).forEach((key: string) => {

const abstractControl = group.get(key);

if (abstractControl instanceof FormGroup) {

this.logValidationErrors(abstractControl);

} else {

this.formErrors[key] = '';

if (abstractControl && !abstractControl.valid

&& (abstractControl.touched || abstractControl.dirty)) {

const messages = this.validationMessages[key];

for (const errorKey in abstractControl.errors) {

if (errorKey) {

this.formErrors[key] += messages[errorKey] + ' ';

}}}}});}}