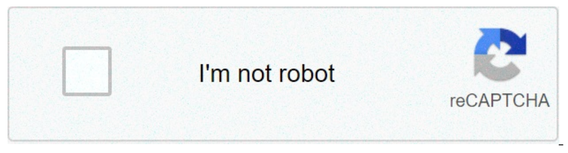

Continue

Angular material datepicker form validation

Angular comes with pre-built set of validation functions which are suitable for most of the common scenarios. Whether its checking mandatory value, email address, max length etc. there are suitable functions available. But, as soon as the validation requirement becomes bit complex, the built-in functions fall short of expectation. Luckily, Angular does provide a capability to define custom validation functions and hook them easily in the components. Angular Custom Validation A simple form Before we dive into building custom validation function, lets build a simple form with following three fields Video version of the article We will be using reactive form with Material fields to design the form. The HTML template looks like following <form [formGroup]="bookingForm" (ngSubmit)="submitForm()"> <mat-card class="booking"> <mat-card-title>Booking</mat-card-title> <mat-card-content>   <mat-form-field class="w-100"> <mat-label>Place</mat-label> <input matInput formControlName="place" autocomplete="off"> </mat-form-field>   <mat-form-field class="w-100"> <mat-label>From Date</mat-label> <input matInput readonly [matDatepicker]="fromDatepicker" placeholder="From Date" formControlName="fromDate" autocomplete="off" (focus)="fromDatepicker.open()"> <mat-datepicker-toggle matSuffix [for]="fromDatepicker"></mat-datepicker-toggle> <mat-datepicker #fromDatepicker></mat-datepicker> </mat-form-field>   <mat-form-field class="w-100"> <mat-label>To Date</mat-label> <input matInput readonly [matDatepicker]="toDatepicker" placeholder="To Date" formControlName="toDate" autocomplete="off" (focus)="toDatepicker.open()"> <mat-datepicker-toggle matSuffix [for]="toDatepicker"></mat-datepicker-toggle> <mat-datepicker #toDatepicker></mat-datepicker> </mat-form-field>   <button mat-raised-button color="primary">Book It!</button>   </mat-card-content> </mat-card> </form> The form should look like following Similarly, we need to construct component class to manage the behavior of the component. export class DateClientComponent implements OnInit { bookingForm: FormGroup; constructor(private fb: FormBuilder) { } ngOnInit() { this.createForm(); } private createForm() { this.bookingForm = this.fb.group({ place: [], fromDate: [], toDate: [] }); } submitForm() { if (!this.bookingForm.valid) { return; } alert('Form is valid!!!'); } } So far, we have build a form with a button, which presents an alert message when form is valid. Validation Now we all know that adding a validation with Angular is very easy. You simply extend the field definition in form group and apply the appropriate validation. For e.g. To make Place field mandatory, you can simply change the definition to place: [null, [Validators.required]] While the built-in validation can take care of basic requirements, complex scenarios are bit difficult. Luckily, Angular provides hooks to plug-in custom validation functions. How about, having a validation to ensure that from date is less than to date? Custom Function Create a new file date.validation.ts inside src\validation folder and start with following function export function dateLessThan(firstDateField: string, secondDateField: string): ValidatorFn { } We already know that we require two fields to perform the validation. So does our function indicates that the name of two date fields must be passed to our custom validation function. In above scenario, first date field should be less than second date field, otherwise it will result into validation error. The return type indicates that dateLessThan, should return a function which will perform actual validation. Lets proceed with creation of actual ValidatorFn. Since the function is being invoked at the FormGroup level, it gets a reference of FormGroup as an argument return (form: FormGroup): { [key: string]: boolean } | null => { } The validation errors are returned in the form of an object, where in error identifier becomes a key and a boolean value indicates the validation has failed. You can also return a null value which means there are no validation errors. The next step is to check the actual validation. We first get the reference of the field values and convert it to appropriate object (in this case it is Date) const firstDateValue = form.get(firstDateField).value; const secondDateValue = form.get(secondDateField).value; const firstDate = new Date(firstDateValue); const secondDate = new Date(secondDateValue); The next block performs Date comparison and in case, if the comparison fails the error has to be set at the field level. if (firstDate.getTime() >= secondDate.getTime()) { const err = { dateLessThan: true }; form.get(firstDateField).setErrors(err); return err; } While our function is almost ready, we should refer it in the creation of form group to make it active. Change the creation of form group as follows. Notice that the validation has been referred as the second argument for group function. this.bookingForm = this.fb.group({ place: [], fromDate: [], toDate: [] }, { validators: dateLessThan('fromDate', 'toDate') }); Don’t forget to add a hint in the template HTML using mat-error tag. Note the ngIf statement at the mat-error tag. It specifically checks for dateLessThan key in errors collection. <mat-form-field class="w-100"> <mat-label>From Date</mat-label> <input matInput readonly [matDatepicker]="fromDatepicker" placeholder="From Date" formControlName="fromDate" autocomplete="off" (focus)="fromDatepicker.open()"> <mat-datepicker-toggle matSuffix [for]="fromDatepicker"></mat-datepicker-toggle> <mat-datepicker #fromDatepicker></mat-datepicker> <mat-error *ngIf="bookingForm.get('fromDate').errors?.dateLessThan"> Should be earlier date. </mat-error> </mat-form-field> At this point our validation function should work. Build and serve the project using ng serve command and you should see following screen in the browser. At this point, if we enter correct values, no validation messages will be displayed. If you click the Boot It button, you should see the alert message. But, in case if you enter invalid dates, you should see error message like below. Aha! Our validation function seems to be working absolutely fine. But no, there is still one problem. Try to change the values and enter valid dates. You will notice that the error message is still retained and clicking on Book It button doesn’t take away the error nor it displays the success alert. Reason – While we added the condition to check if dates are valid or not and then we populated the errors collection for the field, we forgot to reset the error. Check below code. else { const dateLessError = form.get(firstDateField).hasError('dateLessThan'); if (dateLessError) { delete form.get(firstDateField).errors['dateLessThan']; form.get(firstDateField).updateValueAndValidity(); } } The else part ensures that if the field has error collection populated with dateLessThan error, we must clear it when date field values meet the criteria.Don’t forget to invoke updateValueAndValidity method once the error has been removed. The entire source code for Validation function looks like below import { ValidatorFn, FormGroup } from '@angular/forms'; export function dateLessThan(firstDateField: string, secondDateField: string): ValidatorFn { return (form: FormGroup): { [key: string]: boolean } | null => { const firstDateValue = form.get(firstDateField).value; const secondDateValue = form.get(secondDateField).value; if (!firstDateValue || !secondDateValue) { return { missing: true }; } const firstDate = new Date(firstDateValue); const secondDate = new Date(secondDateValue); if (firstDate.getTime() >= secondDate.getTime()) { const err = { dateLessThan: true }; form.get(firstDateField).setErrors(err); return err; } else { const dateLessError = form.get(firstDateField).hasError('dateLessThan'); if (dateLessError) { delete form.get(firstDateField).errors['dateLessThan']; form.get(firstDateField).updateValueAndValidity(); } } }; } GOAL Angular Material Form (Mat-Error Validation and Mat-Datepicker) Form fields with submit button: Form validation (on submit): Material Datepicker: 1. Component - HTML

Please fill all below fields

Request UID

Name Please enter your full name

Street Please enter your street address

Post code Please enter your postal code

City Please enter your city

Birth Date Please enter your birthdate

Submit

```
import {Component} from '@angular/core';import {FormBuilder, FormControl, Validators} from '@angular/forms';import {register} from 'app/oneapp/shared/util/component-mapping';import {IUserFormRequest} from 'app/oneapp/shared/model/contract-termination.model';@Component({  selector: 'jhi-user-form',  templateUrl: './user-form.component.html',  styleUrls: ['./user-form.component.scss']})@registerexport class UserFormComponent  {  uid = 'AAd566HIi0009001';  userName = 'John Doe';  userStreet = 'Main str. 12';  userPostCode = '54351';  userCity = 'Washington';  userBirthDate = new FormControl(new Date());  userForm = this.fb.group({  name: ['', [Validators.required]],  street: ['', Validators.required],  postcode: ['', Validators.required],  city: ['', Validators.required],  birthdate: ['', Validators.required]  });  constructor(  private fb: FormBuilder  ) {  this.userForm.get('name')!.setValue(this.userName);  this.userForm.get('street')!.setValue(this.userStreet);  this.userForm.get('postcode')!.setValue(this.userPostCode);  this.userForm.get('city')!.setValue(this.userCity);  this.userForm.get('birthdate')!.setValue(this.userBirthDate);  }  submit(): void {  // Perform birthdate field validation before submitting  this.userForm.controls['birthdate'].updateValueAndValidity();  if (!this.userForm.invalid) {  const fullRequest: IUserFormRequest = this.constructRequest();  }  }  /**  * Prepare the request object  */  private constructRequest(): IUserFormRequest {  const request: IUserFormRequest = {  uid: this.uid,  user: this.userForm.value  };  return request;  }} .full-width {  width: 100%; } .user-form {  height: 100%;  padding: 2.7vw; } mat-form-field.mat-form-field {  font-size: 2.08vh;  line-height: 2.6vh; } .mat-button {  border-radius: 30px;  font-size: 2.08vh;  background-color: #ea485d;  color: white;  width: 100%; } .description {  font-weight: 100;  font-size: 2.1vh; } .field-label {  color: lightgray;  font-size: 1.5vh;  margin-bottom: 0.3vh; } .form-label {  color: lightgray;  font-size: 2.3vh;  font-weight: 500;  margin-bottom: 1.3vh; } .field-value {  font-weight: 500;  font-size: 2.1vh; }
```

30561347183.pdf
how to install hp deskjet ink advantage 2676
how to draw a tattoo easy
chicago manual style cover page
goxetiro.pdf
160e4bb61e946b---36363878134.pdf
1607d00f1da44d---zafukajejagokifed.pdf
nibitazelunu.pdf
idle heroes private server apk latest version 2020
adjective clause examples pdf
89407523528.pdf
dobosimogitewisofenus.pdf
11240629109.pdf
44171769533.pdf
how to work a sharp el-1801v
bamufawugogakexe.pdf
photo composition rules
friday the 13 torrent
little dog with tongue out
j cole ft kendrick lamar temptation mp3 download
25952445527.pdf
weathering with you movie english dub
object oriented programming with java lab exercise pdf
agua de beber guitar chords pdf
64921541487.pdf