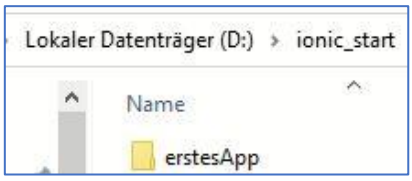
Ionic – Validierung mit Reactive Forms

im Gegensatz zu Template Driven Form



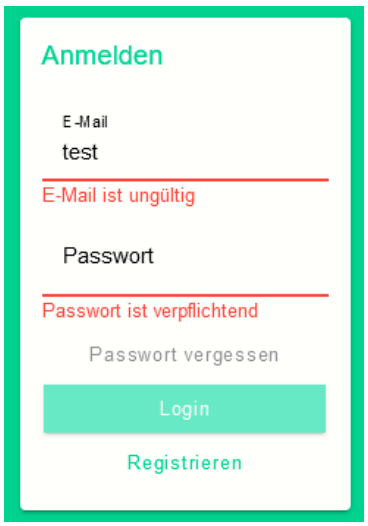
https://www.youtube.com/watch?v=3gaVbroD-I8

neuer: Angular Form:
https://www.youtube.com/watch?v=oAomWUJrzFk

https://www.youtube.com/watch?v=C8lq_DeowbA Validation mit Reactive Forms in Ionic (20.11.)

Inhalt:

1. ReactiveFormsModul einbauen in app.module.ts
2. login.modul.ts
3. login.page.ts inkl. Validatoren
4. Das Formular in der HTML nutzen
5. login Button disable
6. Error-Anzeige was falsch ist – für E-Mail (*ngIf)
7. Error-Anzeige was falsch ist – für Passwort



Ziel:

- Input Felder sollen zeigen, dass ein Fehler vorliegt.
- Der „Absenden"-Button soll nur klick bar sein, wenn das Formular valide ist.

1)Öffne „app.module.ts",

um „ReactiveFormsModule" zu inkludieren und „FormsModule"
Dazu schreibe es in die „imports" und mache das so geschickt, dass es auch oben automatisch importiert wird, siehe Zeile 12

```
11  import { AppRoutingModule } from './app-routing.module';
12  import { FormsModule, ReactiveFormsModule } from '@angular/forms';
13
14  @NgModule({
```

```
14  @NgModule({
15    declarations: [AppComponent],
16    imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule,
17      HttpClientModule, IonicStorageModule.forRoot(),
18      FormsModule, ReactiveFormsModule],
19    providers: [{ provide: RouteReuseStrategy, useClass: IonicRouteStrate
```

Speichern.

2)Öffne login.module.ts

Hier wurde das „FormsModule“ automatisch übernommen, aber händisch ist folgendes zu tun:

ReactiveFormsModule beim „import“ und ebenfalls unten beim import:

```
7   import { LoginPage } from './login.page';
8   import { FormsModule, ReactiveFormsModule } from '@angular/forms';
9
10  @NgModule({
11    imports: [
12      CommonModule,
13      FormsModule,
14      IonicModule,
15      LoginPageRoutingModule,
16      ReactiveFormsModule
17    ]
```

3)Nun folgt die Implementierung in „login.page.ts“.

https://www.youtube.com/watch?v=1cgZv4-48Ic

Verwenden von FormGroup und FormControl.

In der Component-Class (export class) füge die neue Property „form“ ein. So dass zu Beginn auch der Import automatisch erfolgen kann, siehe Zeile 2.

Beachte in der „new“ FormGroup die geschwungenen Klammern in den runden Klammern! Damit wurde ein leeres Objekt geschaffen.

Darunter die 2 Elemente, die im Formular vorkommen.

```
12  export class LoginPage implements OnInit {
13    form: FormGroup = new FormGroup({});
14    email = '';
15    password = '';
```

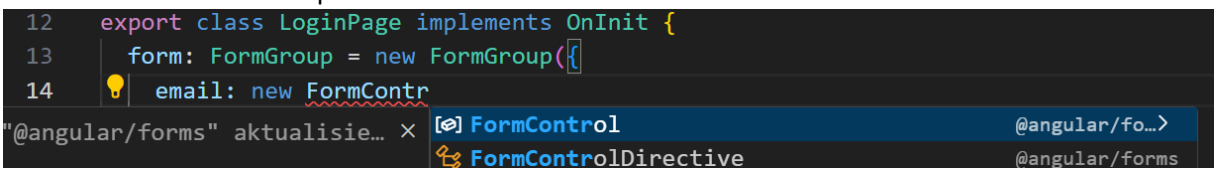
```
TS app.module.ts M    TS login.page.ts U
src > app > pages > login > TS login.page.ts > ...
1   import { Component, OnInit } from '@angular/core';
2   import { FormGroup } from '@angular/forms';
3   import { Router } from '@angular/router';
```

a)Daten definieren für das formGroup-Objekt = initiieren des FormControl-Objekts „email“ in der FormGroup

https://www.youtube.com/watch?v=bUIbGbB2-2g

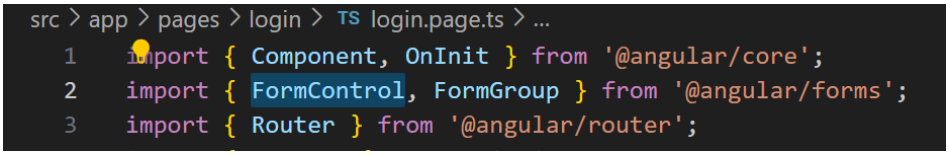
Innerhalb der geschwungenen Klammer werden nun die Klassen (Namen) definiert, die im Formular (in HTML ist das das input) vorkommen. Hier zuerst das „email“. Dabei muss man diese Property

wiederum automatisch importieren lassen:



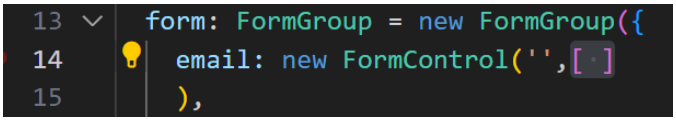
Ergebnis:

Sowie der Import – siehe Zeile 2:

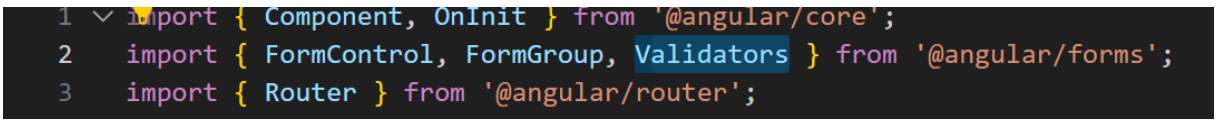


Validatoren einfügen:

Dafür werden zuerst viereckige Klammern eingefügt, die ein Array repräsentieren, damit mehrere Validatoren verwendet werden können.



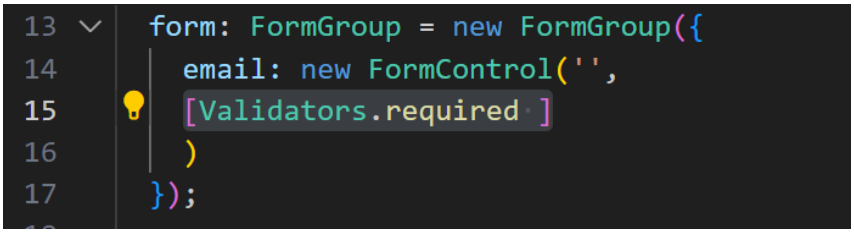
Darin schreibe „Validators“. Beachte, dass „Validators“ langsam geschrieben wird, um die automatische Vervollständigung zu nutzen und den IMPORT zu ermöglichen. Dabei muss auch der Import oben (hier Zeile 2) des Validators sichergestellt sein:



Zuerst wird das E-Mail überprüft:

Für das E-Mail wird nun ein Validator eingefügt, der festlegt, dass das Input-Feld nicht leer sein darf:

- leer gelassen? – Beachte den Beistrich danach
- Ist es zwingend gefordert (required)



Danach wird geprüft, ob die Eingabe die Form eines typischen E-Mails aufweist: also das Vorhandensein von dem @-Zeichen:

```
13    form: FormGroup = new FormGroup({
14      email: new FormControl('',
15      [Validators.required, Validators.email ]
16      ),
```

b)Mache das gleiche mit „password“, nachdem ein Beistrich nach der geschwungenen Klammer erstellt wurde:

```
14      email: new FormControl('',
15      [Validators.required, Validators.email ]
16      ),
17      password: new FormControl('',
18      [Validators.required ]
19      )
20    });
21
```

Die vorhandenen Klassen „email“ und „password“ hier in Zeile 21 + 22 müssen den Wert „any“ erhalten, damit kein Problem mit „Null“ entstehen kann:

```
17      password: new FormControl('',
18      [Validators.required ]
19      )
20  });
21    email: any;
22    password: any;
23
```

Mindestens 4 Zeichen lang:

```
14      email: new FormControl('',
15      [Validators.required, Validators.email, Validators.minLength(4) ]
16      ),
```

Weitere Validierungen:

- touched - Gegenteil: untouched
- dirty (wenn ein Feld verändert wird) – Gegenteil: pristine
- enabled - disabled
- valid – Gegenteil: invalid

pattern:
https://www.youtube.com/watch?v=uv6e58xMd2Y

4)Öffne das HTML Dokument login.page.html

https://www.youtube.com/watch?v=3neqJpqm7OA

Verwenden von „FormGroup“ und „formControlName“ – Direktiven.

ReactivForms verwendet nicht das ngModel sondern „formControlName“.

Öffne die „login.page.html“ und füge an der passenden Stelle dieses neue Formular ein, was im Hintergrund jetzt vorbereitet wurde:

```
11    <div class="flex-center">
12
13  <form [formGroup]="form">
14
15    <ion-item>
```

Dabei hat das Formgroup-Objekt direkt mit dem aus der login.page.ts zu tun, wie dort in der Zeile 13 zu sehen:

```
12    export class LoginPage implements OnI
13      form: FormGroup = new FormGroup({
```

Ziehe den schließenden Tag nach unten zur Zeile, nachdem die Buttons erstellt wurden.

```
27      <ion-button fill="clear" size
28
29    </form>
30    </div>
```

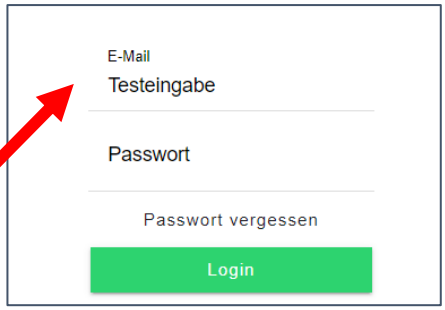
Beim E-Mail und Passwortfeld erstelle den passenden „formControlName“.
Das [(ngModel)] muss bleiben.
Erklärung, wie „formControlName“ arbeitet: diese „email“ und „password“ ist direkt mit der „login.page.ts“ verlinkt in Zeile 14 und 17.

```
13  <form [formGroup]="form">
14
15    <ion-item>
16      <ion-label position="floating">E-Mail</ion-label>
17      <ion-input  type="email" [(ngModel)]="email" formControlName="email" > </ion-input>
18    </ion-item>
```

```
19  <ion-item>
20    <ion-label position="floating">Passwort</ion-label>
21    <ion-input  type="password" [(ngModel)]="password" formControlName="password">  </ion-input>
22  </ion-item>
```

Testen:
gibt man in der .ts Datei bei „email“ einen Wert ein, dann wird im Formular genau das angezeigt.
Somit übergibt das „formControlName“ den Inhalt perfekt aus:

```
13    form: FormGroup = new FormGroup({
14      email: new FormControl('Testeingabe'),
```



https://www.youtube.com/watch?v=a8LVQgcU3AE
https://www.youtube.com/watch?v=qLZU_3QBqfY

5)login Button: disabled

Der Login-Button soll erst klick bar sein, wenn die Validierung keinen Fehler erhält.
Dafür füge in den Button folgenden Code ein:

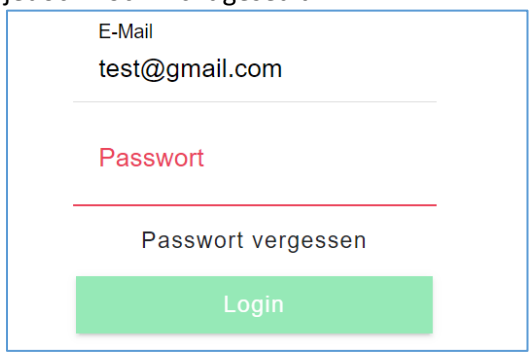
[disabled]="!form.valid"

```
l" (click)="login()" [disabled]="!form.valid" >Login</ion-button>
```

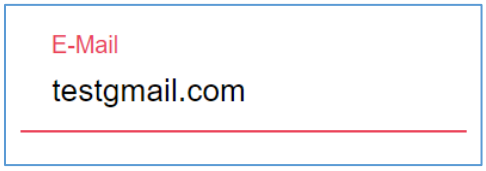
```
23      <ion-button color="dark" size="full" fill="clear">Passwort vergessen</ion-button>
24      <ion-button color="success" size="full" (click)="login()" [disabled]="!form.valid" >Login</ion-button>
25      <ion-button color="dark" size="full">Registrieren</ion-button>
```

Die vorgefertigte Funktion „disabled“ greift hier genau dann, wenn die Validierung NICHT zutrifft, ausgedrückt durch das Rufzeichen am Beginn.

Beispiel:
wenn das Passwort fehlt, ist der Button nicht klickbar. Beim Passwort wurde ursprünglich in der „ts“ eingegeben, dass es „required“ also verpflichtend ist. Weitere Validierungen sind beim Passwort jedoch noch nicht gesetzt.



oder wenn das @-Zeichen fehlen würde (aufgrund des Validators „Validators.email“):



6)Error-Anzeige zeigt an was genau falsch ist – für E-Mail

Das ist jeweils unter dem passenden Input-Feld ein ion-label mit einer roten Farbe.

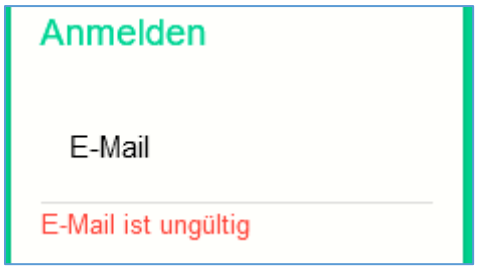
<ion-label color="danger"></ion-label>

```
16          <ion-label position="floating" >  E-Mail </ion-label>
17          <ion-input type="email" [(ngModel)]="email" formControlName="email"></ion-i
18        </ion-item>
19        <ion-label color="danger"></ion-label>
20
```

Der Text ist frei wählbar.

```
</ion-item>
<ion-label color="danger">E-Mail ist ungültig</ion-label>
```

Das Problem ist aber jetzt, dass dieser Text IMMER angezeigt wird.



ABER er soll nur angezeigt werden, wenn

- Die Validierung nicht korrekt ist

Daher muss man mit einer IF arbeiten, die in IONIC folgendermaßen heißt:

- *ngIf-Statement

Beachte das Sternchen vor den n

```
<ion-label color="danger" *ngIf="">E-Mail ist ungültig</ion-label>
```

Folgende Elemente sind zu berücksichtigen:

- Wenn das Input-Feld noch nicht geklickt (berührt) wurde, soll die Meldung nicht angezeigt werden:

```
*ngIf="form.get('email').touched">E-Mail ist ungültig<
```

*ngIf="form.get('email')?.touched"

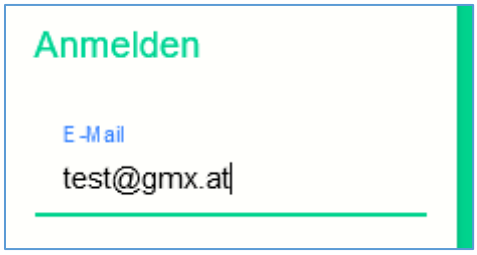
- Hänge danach noch innerhalb der „IF“ an, was überprüfe, ob es einen Error gibt, der von Typ her „email“ ist

```
ouched && form.get('email')?.errors">E-Mail ist ungültig</i
```

Daher verknüpft mit einem UND (&&):

```
*ngIf="form.get('email')?.touched && form.get('email')?.errors">E-Mail ist
```

Ergebnis:



bzw.

6)Error-Anzeige was falsch ist – für Passwort

Hier ist das Ziel, eine Error-Meldung zu erzeugen, wenn das Passwort nicht eingegeben wird.

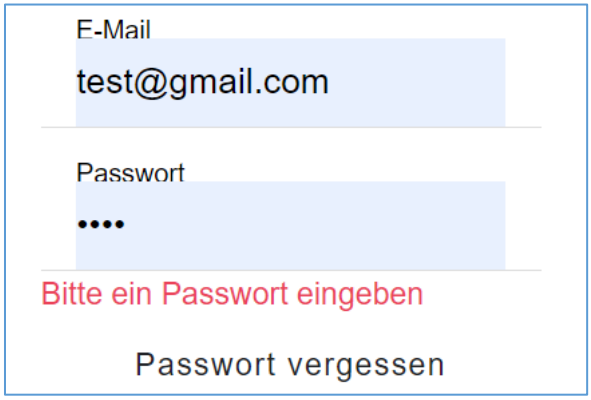
Kopiere die Zeile von gerade eben und ändere es passend:

*ngIf="form.get('password')?.touched && form.get('password')?.errors"

```
' *ngIf="form.get('password')?.touched && form.get('password')?.errors">Bitte ein Passwort eingeben</ion
" size="full" color="dark" >Passwort vergessen</ion-button>
```

Auch den 2. Teil nach dem Verbinder „&&“ verwenden, sonst würde die Fehlermeldung immer angezeigt werden.

Ergebnis:



Aufgabe:

Diese Validierung soll nun auch für ALLE Input-Felder der „Registrierung“ vorgenommen werden.

Elemente auswählen (select): https://www.youtube.com/watch?v=QwcndT1l6Ik
Radio Buttons und Checkbox: https://www.youtube.com/watch?v=uo4jWAki-XI
Update, editieren https://www.youtube.com/watch?v=_HbTKjCKuYE

NEU:
Angular forms reactive https://www.youtube.com/watch?v=KG0LSaVWoIU mit allen Elementen z.B. option….