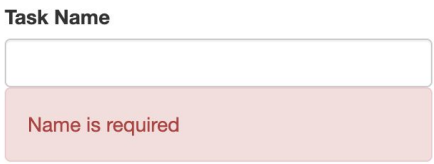
Template-driven form: Input Validation

Pada template-driven form, Kita bisa menambahkan validasi input, sebagaimana menambahkan validasi html form, seperti **required**, **minlength, maxlength**, dll. (lebih lanjut macam-macam validasi HTML: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Constraint_validation)

```
<div class="form-group">
  <label for="name">Task Name</label>
  <input
    type="text"
    class="form-control"
    id="taskName"
    name="taskName"
    ngModel
    required
    #taskName="ngModel"
  >
  <div [hidden]="taskName.valid || taskName.pristine" class="alert alert-danger">
    Name is required
  </div>
</div>
```



Contoh disamping adalah cara menambahkan validasi **required** pada elemen input.
Tambahkan pula variable template misal, **taskName** yang berisikan **ngModel**.

Karena **taskName** mereferensikan ngModel di input, maka kita bisa menggunakan properti-properti yang disediakan, seperti pengecekan valid tidaknya input dengan **.valid**, pengecekan apakah value input telah diubah dengan **.pristine**.

Bila input tidak valid, (contoh disini input dihapus hingga kosong), maka munculkan alert mengenai validasi input yang harus dipenuhi.

Referensi properti dalam NgModel: https://angular.io/api/forms/NgModel

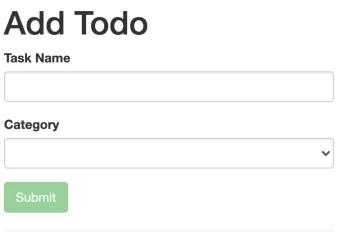
Template-driven form: Form Validation

Selain validasi input yang menggunakan ngModel untuk pengecekan validitas, kita juga bisa menambahkan validasi form, yang menggunakan ngForm untuk pengecekan.

```
<form (ngSubmit)="onSubmit(todoForm)" #todoForm="ngForm">
  <div class="form-group">…
  </div>

  <div class="form-group">…
  </div>

  <button
    type="submit"
    class="btn btn-success"
    [disabled]="!todoForm.form.valid"
  >
    Submit
  </button>
</form>
```



Contoh disamping, membuat button submit ter-disable ketika form (berikut elemen value didalamnya) belum valid.

Referensi properti dalam NgModel: https://angular.io/api/forms/NgModel

Reactive Form

Reactive Form, atau yang dikenal juga dengan model-driven form, membutuhkan module **ReactiveFormsModule**.

app.module.ts

```ts
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { ReactiveFormsModule } from '@angular/forms';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [
    BrowserModule,
    // other imports ...
    ReactiveFormsModule,
  ],
  providers: [],
  bootstrap: [AppComponent],
})
export class AppModule {}
```

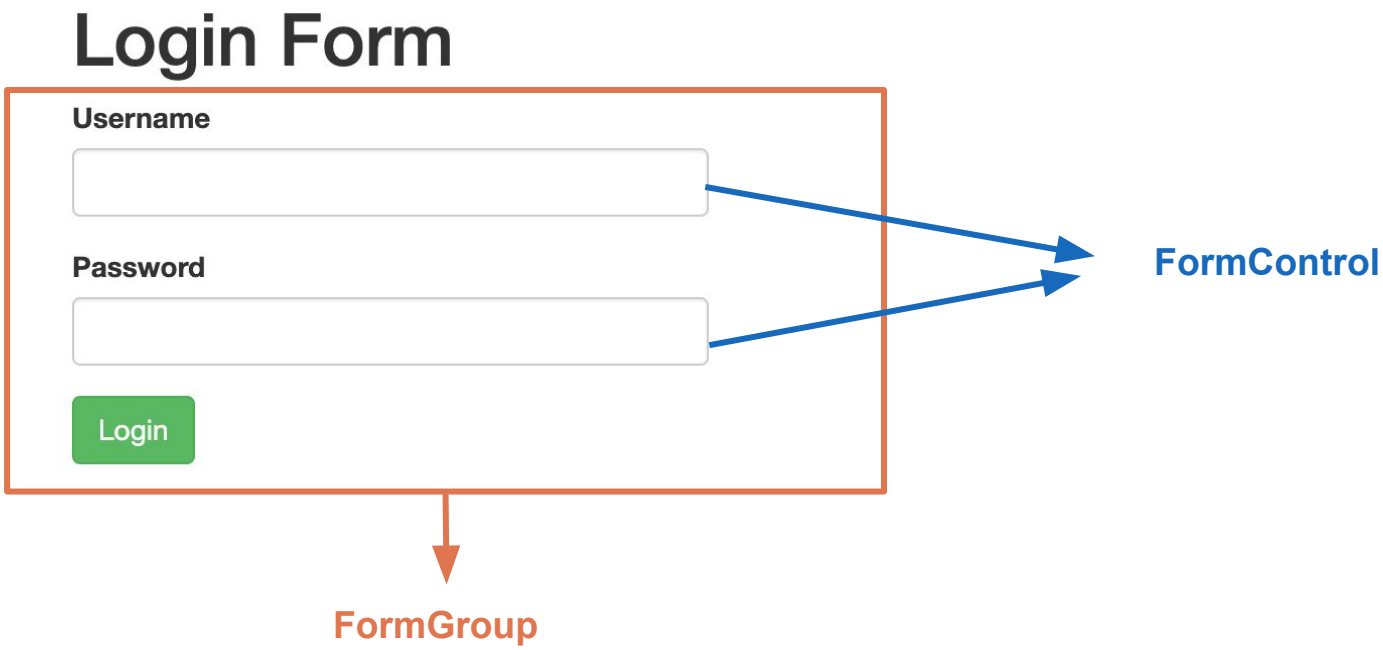
Referensi: https://angular.io/guide/reactive-forms

Class-class dalam Form

- **FormControl**: men-track value dan validasi dari satu elemen dalam form. tracks the value and validation status of an individual form control.
- **FormGroup**: men-track value dan validasi dari form group.
- **FormArray** tracks the same values and status for an array of form controls.
- **ControlValueAccessor** creates a bridge between Angular FormControl instances and built-in DOM elements.

Referensi: https://angular.io/guide/forms-overview

Reactive Form: Contoh login



Referensi: https://angular.io/guide/reactive-forms

Reactive Form: Contoh login

```ts
import { Component } from '@angular/core';
import {FormGroup, FormControl} from '@angular/forms';

@Component({
  selector: 'app-reactive-form',
  templateUrl: './reactive-form.component.html',
  styleUrls: ['./reactive-form.component.css']
})
export class ReactiveFormComponent {
  currentUser:{
    isLogin: boolean;
    username: string,
    password: string
  } = { isLogin:false, username: '', password: '' }


  loginForm = new FormGroup({
    username: new FormControl(''),
    password: new FormControl('')
  })

  onLogin( ) {
    console.log(this.loginForm); // berisi object Form

    this.currentUser = {
      isLogin: true,
      username: this.loginForm.value.username,
      password: this.loginForm.value.password
    }
  }

}
```

.component.ts

```html
<h1>Login Form</h1>

<form [formGroup]="loginForm" (ngSubmit)="onLogin()">
  <div class="form-group">
    <label for="name"> Username</label>
    <input
      type="text"
      class="form-control"
      id="username"
      name="username"
      formControlName="username"
    >
  </div>

  <div class="form-group">
    <label for="name"> Password</label>
    <input
      type="password"
      class="form-control"
      id="password"
      name="password"
      formControlName="password"
    >
  </div>

  <button
    type="submit"
    class="btn btn-success"
  >
    Login
  </button>
</form>

<div *ngIf="currentUser.isLogin">
  Successfully login. Hello, {{currentUser.username}}!
</div>
```

.component.html

Referensi: https://angular.io/guide/reactive-forms

Reactive Form: Input Validation

1. Gunakan class **Validators** untuk memvalidasi input.
2. **Getter** diperlukan agar dapat melakukan pengecekan dan memunculkan alert di template.

```ts
import { Component } from '@angular/core';
import {FormGroup, FormControl, Validators} from '@angular/forms';

@Component({…
})
export class ReactiveFormComponent {
  currentUser:{…
  } = { isLogin:false, username: '', password: '' }


  loginForm = new FormGroup({
    username: new FormControl('', [
      Validators.required,
      Validators.minLength(5),
    ]),
    password: new FormControl('')
  })

  get username() {
    return this.loginForm.get('username')
  }

  onLogin( ) {…
  }

}
```

```html
<div class="form-group">
  <label for="name"> Username</label>
  <input
    type="text"
    class="form-control"
    id="username"
    name="username"
    formControlName="username"
  >
  <span *ngIf="username && username.invalid"
    style="color: red">
    Input Username is invalid
  </span>
</div>
```

Referensi macam-macam validasi: https://angular.io/api/forms/Validators