academeez

@angular/forms

5. Validation - Angular's built in form validation

Validation

- Angular provides us with basic, form validation

```
<form>
    <input
        type="text"
        name="phone-number"
        [(ngModel)]=“phoneNumber”
        required
        minlength=“3”
        maxlength=“30”
        pattern=“[0-9]*”
    ↗
    <button type=“submit”>Submit↙button>
↙form>
```

```
firstName: FormControl = new FormControl("hello", [
    Validators.required,
    Validators.minLength(3)
]);
```

Validation Directives

- Inside @angular/forms there are validation directives that are available for us to use when we add the **FormsModule** to the **imports** array
- Those directives are added when you attach a **FormControl** (either with **NgModel** or **Reactive**) to an input and also add an HTML form validation attribute
- Let's examine those directives…

RequiredValidator

academeez

- Validate for required input.

```
@Directive({
  selector:
      ':not([type=checkbox])[required][formControlName],:not([type=checkbox])[required]
[formControl],:not([type=checkbox])[required][ngModel]',
  //...
})
export class RequiredValidator implements Validator {
  // ....
}
```

CheckboxRequiredValidator

- Require that a checkbox is checked.

```
@Directive({
  selector:
      'input[type=checkbox][required][formControlName],input[type=checkbox][required]
[formControl],input[type=checkbox][required][ngModel]',
  // ...
})
export class CheckboxRequiredValidator extends RequiredValidator {
  // ....
}
```

EmailValidator

academeez

- Validate that a text input is in a correct email format

```
@Directive({
  selector: '[email][formControlName],[email][formControl],[email][ngModel]',
  // ...
})
export class EmailValidator implements Validator {
  // ....
}
```

MinLengthValidator

academeez

- Validate the length of the input has more character than X

```
@Directive({
  selector: '[minlength][formControlName],[minlength][formControl],[minlength][ngModel]',
  // ...
})
export class MinLengthValidator implements Validator, OnChanges {
  // ...
}
```

MaxLengthValidator

academeez

- Validate the input is no longer than X

```
@Directive({
  selector: '[maxlength][formControlName],[maxlength][formControl],[maxlength]
[ngModel]',
  // ...
})
export class MaxLengthValidator implements Validator, OnChanges {
  // ...
}
```

PatternValidator

- Verify that an input match a certain pattern

```
@Directive({
  selector: '[pattern][formControlName],[pattern][formControl],[pattern][ngModel]',
  // ...
})
export class PatternValidator implements Validator, OnChanges {
// ...
}
```

Using the directives

- You can easily use those directives by placing them on a template along with **NgModel** or **FormControl**

```
<input type="text" required name="firstName" [formControl]="firstName" ↗
      {{ firstName.errors | json }}

<input
        type="text"
        minlength="3"
        name="lastName"
        [(ngModel)]="lastName"
        #lastNameNgModel="ngModel"
↗
{{ lastNameNgModel.errors | json }}
```

Validation Functions

academeez

- The same validations are also supplied as functions.
- Validation functions can be applied on Reactive forms

```
import { FormControl, Validators } from "@angular/forms";


firstName: FormControl = new FormControl(
    "hello",
    [Validators.required, Validators.minLength(3)]
);
```

Summary

- Angular is providing us with form validations inside **FormsModule**
- Those validations can be applied as Directives to **NgModel** and **FormControl**
- The validation functions are exposed to be applied directly in the **FormControl**

academeez

Thank You

Next Lesson: 6. Custom Validation