academeez

# @angular/forms

## 6. Custom Validation

# Custom Validation

- We can create our own custom validation

```
<!– Validate correct phone –>
<input type="text" name="phone-number" [(ngModel)]=“phoneNumber”
    required
    phone
↗
<!– Validate correct social security format –>
<input type="text" name=“social-security” [(ngModel)]=“scNumber”
    required
    socialSecurity
↗
```

# Custom Validation as Directive / Function

- Our custom validation can be applied with a directive on the template
- Our custom Validation can be applied as a function to Reactive

```
<input
        type="text"
        required
        phoneNumber
        name="phone"
        [formControl]="phone" ↗
```

```
phone: FormControl = new FormControl(
    "",
    [Validators.required, Validators.phone]
);
```

# FormControl Validation directive communication

- For the validation to work, the FormControl needs to grab the instance of the validation directive
- after grabbing the instance the FormControl needs to call a validation method on the directive class.
- Angular supplies us with the **Validator** interface. The Validation directive needs to implement that interface so that the **FormControl** can call the proper method.

# Validator interface

- Validation directive needs to implement this interface
- Validation directive can on: FormGroup, FormArray, FormControl

```
export interface Validator {
  validate(control: AbstractControl): ValidationErrors|null;
  registerOnValidatorChange?(fn: () => void): void;
}
```

# EX. Validation Directive PhoneValidator

academeez

- Create a validation directive to verify a phone number

```
<input
        name="phone"
        type="text"
        phone
        [(ngModel)]="phone"
        #phoneNgModel="ngModel"
      ↗
      {{ phoneNgModel.errors | json }}
```

# EX. Validation Directive Password/Repeat

academeez

- Create a validation directive that will get a FormGroup and will validate that the password and repeat password match

```
<form passwordRepeat #repeatNgForm="ngForm">
    …
</form>
```

# ValidatorFn

academeez

- Custom Validation that can be applied only to Reactive can be created using a function
- Angular provided an interface for that Function

```
export interface ValidatorFn {
  (control: AbstractControl): ValidationErrors|null;
}
```

# Ex. ValidatorFn

- Create the Validators we created in the previous ex. but as functions and use them in a reactive form

# Summary

academeez

- Angular provides us tools to create our own reusable for validation
- We can create a validation directive
- We can create a validation function that we can use on Reactive

academeez

# Thank You

## Next Lesson: 7. ControlValueAccessor