luv2code

Checkout Form Validation

Form Validation

- Before we submit the form to the backend ... let's perform validation

- Check for required fields, min length etc ...

Angular Validation

- Angular has a set of built-in validation rules

Name	Description
required	Must be a non-empty value
min	Must be a number >= value
max	Must be a number <= value
minLength	Length must be >= value
maxLength	Length must be <= value
pattern	Must match a regular expression pattern
email	Match email against common regular expression patterns
others …	

https://angular.io/api/forms/Validators

Additional Validation Features

- Define custom validators
- Cross-field validation
- Asynchronous validators

https://angular.io/guide/form-validation

Checkout Form Validation

From: The Boss

- Our requirements
  - All fields are required
  - Email address: has proper email format
  - Credit card field: only numbers allowed (16 digits)
  - CVC number: only numbers allowed (3 digits)

Development Process

1. Specify validation rules for the form controls
2. Define Getter methods to access form controls
3. Update HTML template to display error messages
4. Add event handler to check validation status when submit button clicked

Step 1: Specify validation rules for form controls

- Recall that a form field is represented by FormControl object
- Create the FormControl and pass in initial value and validators

Basic Syntax

```
new FormControl(initialValue, validators, ...)
```

array of validators

Example

```
new FormControl('', [Validators.required, Validators.minLength(2)])
```

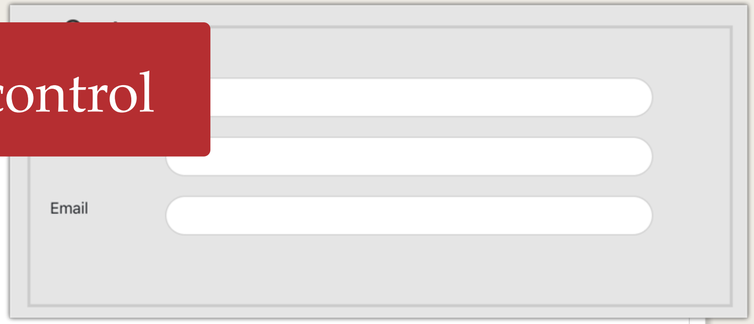
Step 1: Specify validation rules for form controls

File: checkout.component.ts

```
    this.checkoutFormGroup = this.formBuilder.group({

      customer: this.formBuilder.group({
        'firstName': new FormControl('',
          [Validators.required, Validators.minLength(2)]
        ),

        'lastName': new FormControl('',
          [Validators.required, Validators.minLength(2)]
        ),

        'email': new FormControl('',
          [Validators.required,
          Validators.pattern('^[a-z0-9._%+-]+@[a-z0-9.-]+\\.[a-z]{2,4}$')] // works for anuva@gmail.com
        )
      }),
  ...
  });
```

firstName form control

Regular Expression



Validators.email vs Validators.pattern

- You may wonder why we aren't using Angular: `Validators.email` ???
- `Validators.email` only checks for: `<someText>@<moreText>`
  - Based on this ... the following is valid email format:
    - `anil@luv2code`
    - `anil@gmail`

Too lenient :-(

- It does not check for domain name format: `anil@luv2code.com` or `anil@gmail.com`
- As a result, we use a regular expression with `Validators.pattern`

Validators.email vs Validators.pattern

- You may wonder why we aren't using Angular: `Validators.email` ???
- `Valid` … `ext>`
  - Base…
    - `an`…
    - `an`…

Validators.email and Validators.pattern

Only check the FORMAT

Does not verify if email address is real

- It does not check for domain name format: `anil@luv2code.com` or `anil@gmail.com`
- As a result, we use a regular expression with `Validators.pattern`

Step 2: Define Getter methods to access form controls

File: checkout.component.ts

```
  get firstName() {
    return this.checkoutFormGroup.get('customer.firstName');
  }

  get lastName() {
    return this.checkoutFormGr
  }

  get email() {
    return this.checkoutFormGr
  }

...
```

```
    this.checkoutFormGroup = this.formBuilder.group({

      customer: this.formBuilder.group({
        'firstName': new FormControl('', [
          Validators.required, Validators.minLength(2)
        ]),

        'lastName': new FormControl('', [
          Validators.required, Validators.minLength(2)
        ]),

        'email': new FormControl('', [
          Validators.required,
          Validators.pattern('^[a-z0-9._%+-]+@[a-z0-9.-]+\\.[a
        ])
      }),
    ...
    });
```

Step 3: Update HTML template to display error messages

File: checkout.component.html

If validation fails
then display error messages

```
<label>First Name</label>
<input formControlName="firstName" type="text">

<div *ngIf="firstName.invalid && (firstName.dirty || firstName.touched)" class="alert alert-danger">

 <div *ngIf="firstName.errors.required">
  First Name is required
 </div>

 <div *ngIf="firstName.errors.minlength">
  First Name must be at least 2 characters long
 </div>

</div>
```

Step 3: Update HTML template to display error messages

File: checkout.component.html

invalid: did validations fail?

```
<label>First Name</label>
<input formControlName="firstName" type="text">

<div *ngIf="firstName.invalid && (firstName.dirty || firstName.touched)" class="alert alert-danger">

 <div *ngIf="firstName.errors.required">
  First Name is required
 </div>

 <div *ngIf="firstName.errors.minlength">
  First Name must be at least 2 characters long
 </div>

</div>
```

Step 3: Update HTML template to display error messages

File: checkout.component.html

dirty: did user change field value?

```
<label>First Name</label>
<input formControlName="firstName" type="text">

<div *ngIf="firstName.invalid && (firstName.dirty || firstName.touched)" class="alert alert-danger">

</div>
```

Only display validation errors if user has interacted with the form
- When the user changes field value, the control is marked as "dirty"
- When the field loses focus, the control is marked as "touched"

Step 3: Update HTML template to display error messages

File: checkout.component.html

touched: did field lose focus?

```
<label>First Name</label>
<input formControlName="firstName" type="text">

<div *ngIf="firstName.invalid && (firstName.dirty || firstName.touched)" class="alert alert-danger">

</div>
```

Only display validation errors if user has interacted with the form
- When the user changes field value, the control is marked as "dirty"
- When the field loses focus, the control is marked as "touched"

Step 3: Update HTML template to display error messages

File: checkout.component.html

```
<label>First Name</label>
<input formControlName="firstName" type="text">

<div *ngIf="firstName.invalid && (firstName.dirty || firstName.touched)" class="alert alert-danger">

 <div *ngIf="firstName.errors.required">
  First Name is required
 </div>

 <div *ngIf="firstName.errors
  First Name must be at leas
 </div>

</div>
```

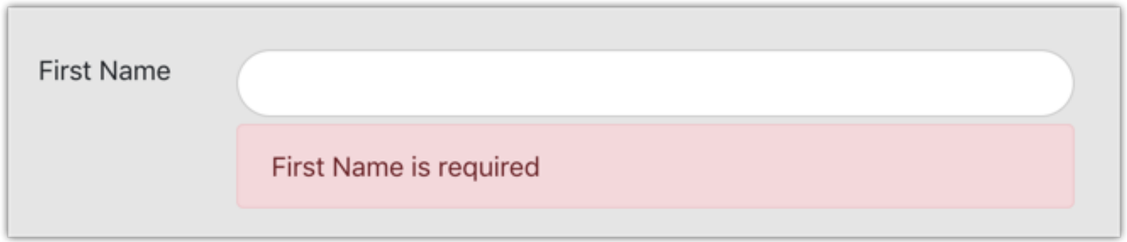
errors object contains list of validations that failed for this form control

Step 3: Update HTML template to display error messages

File: checkout.component.html

```
<label>First Name</label>
<input formControlName="firstName" type="text">

<div *ngIf="firstName.invalid && (firstName.dirty || firstName.touched)" class="alert alert-danger">

 <div *ngIf="firstName.errors.required">
  First Name is required
 </div>

 <div *ngIf="firstName.errors.minlength">
  First Name must be at least 2 characters long
 </div>

</div>
```

Display specific error message

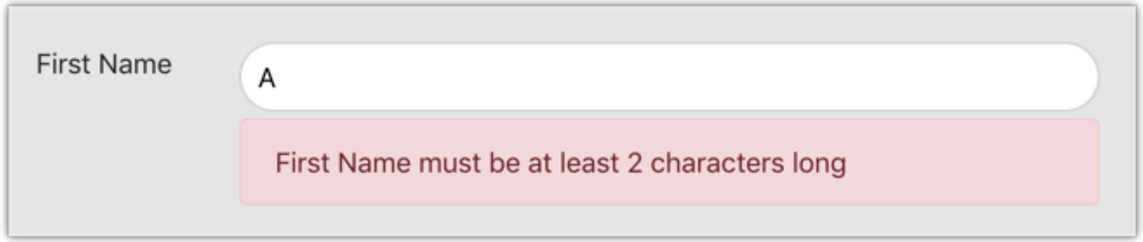


Step 3: Update HTML template to display error messages

File: checkout.component.html

```
<label>First Name</label>
<input formControlName="firstName" type="text">

<div *ngIf="firstName.invalid && (firstName.dirty || firstName.touched)" class="alert alert-danger">

 <div *ngIf="firstName.errors.required">
  First Name is required
 </div>

 <div *ngIf="firstName.errors.minlength">
  First Name must be at least 2 characters long
 </div>

</div>
```

Display specific error message



Step 3: Update HTML template to display error messages

File: checkout.component.html

Email address field

```
<label>Email</label>
<input formControlName="email" type="text">

<div *ngIf="email.invalid && (email.dirty || email.touched)" class="alert alert-danger">

 <div *ngIf="email.errors.required">
  Email is required
 </div>

 <div *ngIf="email.errors.pattern">
  Email must be a valid email address
 </div>

</div>
```

Display error messages

Step 4: Add event handler to check validation on submit

File: checkout.component.ts

```
onSubmit() {
  console.log("Handling the submit button");
  ...

  if (this.checkoutFormGroup.invalid) {
    this.checkoutFormGroup.markAllAsTouched();
  }

  console.log("CheckoutFormGroup is valid: " + this.checkoutFormGroup.valid);
}
```

Touching all fields triggers the display of the error messages