Forms

At the end of Managing Data, the online store application has a product catalog and a shopping cart.

In this section, you'll finish the app by adding a form-based checkout feature. You'll create a form to collect user information as part of checkout.

Forms in Angular

Forms in Angular take the standard capabilities of the HTML based forms and add an orchestration layer to help with creating custom form controls, and to supply great validation experiences. There are two parts to an Angular Reactive form, the objects that live in the component to store and manage the form, and the visualization of the form that lives in the template.

Define the checkout form model

First, you'll set up the checkout form model. The form model is the source of truth for the status of the form and is defined in the component class.

1. Open `cart.component.ts`.
2. Angular's `FormBuilder` service provides convenient methods for generating controls. As with the other services you've used, you need to import and inject the service before you can use it:
   a. Import the `FormBuilder` service from the `@angular/forms` package.

src/app/cart/cart.component.ts

```
import { Component } from '@angular/core';
import { FormBuilder } from '@angular/forms';

import { CartService } from '../cart.service';
```

   The `FormBuilder` service is provided by the `ReactiveFormsModule`, which is already defined in the `AppModule` you modified previously (in `app.module.ts`).

   b. Inject the `FormBuilder` service.

src/app/cart/cart.component.ts

```
export class CartComponent {
  items:

  constructor(
    private cartService: CartService,
    private formBuilder: FormBuilder,
  ) {
  }
}
```

3. In the `CartComponent` class, define the `checkoutForm` property to store the form model.

src/app/cart/cart.component.ts

```
export class CartComponent
{
  items:
  checkoutForm;
}
```

4. During checkout, the app will prompt the user for a name and address. So that you can gather that information later, set the `checkoutForm` property with a form model containing `name` and `address` fields, using the `FormBuilder#group()` method.

src/app/cart/cart.component.ts

```
export class CartComponent {
  items:
  checkoutForm;

  constructor(
    private cartService: CartService,
    private formBuilder: FormBuilder,
  ) {
    this.items = this.cartService.getItems();

    this.checkoutForm = this.formBuilder.group({
      name: '',
      address: ''
    });
  }
}
```

5. For the checkout process, users need to be able to submit the form data (their name and address). When the order is submitted, the form should reset and the cart should clear.
In `cart.component.ts`, define an `onSubmit()` method to process the form. Use the `CartService#clearCart()` method to empty the cart items and reset the form after it is submitted. (In a real-world app, this method also would submit the data to an external server.)
The entire cart component is shown below:

src/app/cart/cart.component.ts

```
import { Component } from '@angular/core';
import { FormBuilder } from '@angular/forms';

import { CartService } from '../cart.service';

@Component({
  selector: 'app-cart',
  templateUrl: './cart.component.html',
  styleUrls: ['./cart.component.css']
})
export class CartComponent {
  items:
  checkoutForm;

  constructor(
    private cartService: CartService,
    private formBuilder: FormBuilder,
  ) {
    this.items = this.cartService.getItems();

    this.checkoutForm = this.formBuilder.group({
      name: '',
      address: ''
    });
  }

  onSubmit(customerData) {
    // Process checkout data here
    console.warn('Your order has been submitted', customerData);

    this.items = this.cartService.clearCart();
    this.checkoutForm.reset();
  }
}
```

The form model is defined in the component class. To reflect the model in the view, you'll need a checkout form.

Create the checkout form

Next, you'll add a checkout form at the bottom of the 'Cart' page.

1. Open `cart.component.html`.
2. At the bottom of the template, add an empty HTML form to capture user information.
3. Use a `FormGroup` property binding to bind the `checkoutForm` to the `form` tag in the template. Also include a "Purchase" button to submit the form.

src/app/cart/cart.component.html

```
<form [formGroup]="checkoutForm">

  <button class="button" type="submit">Purchase</button>

</form>
```

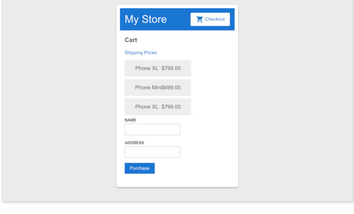
4. On the `form` tag, use an `ngSubmit` event binding to listen for the form submission and call the `onSubmit()` method with the `checkoutForm` value.

```
<form [formGroup]="checkoutForm" (ngSubmit)="onSubmit(checkoutForm.value)">
</form>
```

5. Add input fields for `name` and `address`. Use the `formControlName` attribute binding to bind the `checkoutForm` form controls for `name` and `address` to their input fields. The final complete component is shown below:

```
<h3>Cart</h3>

<p>
  <a routerLink="/shipping">Shipping Prices</a>
</p>

<div class="cart-item" *ngFor="let item of items">
  <span>{{ item.name }} </span>
  <span>{{ item.price | currency }}</span>
</div>

<form [formGroup]="checkoutForm" (ngSubmit)="onSubmit(checkoutForm.value)">

  <div>
    <label for="name">
      Name
    </label>
    <input id="name" type="text" formControlName="name">
  </div>

  <div>
    <label for="address">
      Address
    </label>
    <input id="address" type="text" formControlName="address">
  </div>

  <button class="button" type="submit">Purchase</button>

</form>
```

After putting a few items in the cart, users can now review their items, enter name and address, and submit their purchase:



Next steps

Congratulations! You have a complete online store application with a product catalog, a shopping cart, and a checkout function.

Continue to the "Deployment" section to move to local development, or deploy your app to Firebase or your own server.