Challenge 0: Create Project

Objectives

- Create a new project

Steps

1. Use what you learned in the setup part of the course to create a new project named **Challenges**
2. Run the project in debug mode.
3. Install the Bootstrap CSS library.
   a. https://getbootstrap.com/docs/4.3/getting-started/introduction/
   b. Use npm install to pull it into your project as a dependency.
   c. Link just the stylesheet not the JavaScript files including jQuery.

Challenge 1: List Component

Objectives

- Create a list component

Requirements

1. Create a component that lists the instances of **contract-type.model.ts** in **mock-contract-types.ts**.

   a. The **files** mentioned above can be found in **code/labs/snippets/challenges**.

   b. Follow the Angular Style Guide when naming files and classes as well as to determine folder structure.

      - https://angular.io/guide/styleguide

   c. Use the Bootstrap list group to format the list.

      - https://getbootstrap.com/docs/3.3/components/#list-group

Challenge 2: Directives

Objectives

- Allow inline editing of list items

Requirements

1. When a user clicks a list item it should become editable, i.e. become an input element with a type of text.

   ☞ Wrap the contractType name in a label tag so it can be shown or hidden separately from the input.

   ☞ Add an editing property to the model to keep track of whether a given list item is being edited.

   ⚠ For now the input element can be blank when it appears. In later challenges, we will populate this value and save the edits.

   ⚠ When you click on an item after the first item in the list the input that becomes visible but it will not get focus until you click in it. We will fix this issue in Challenge 7 are learn more about directives.

2. The input element should become a static list item again when the input has focus and when…

   a. The escape key is pressed.

   b. The input loses focus (blur) event.

   c. The enter key is pressed.

   ☞ You can bind to a specific key using the event binding syntax: (keyup.escape)="myMethod()".

Challenge 3: Components

Objectives

- Refactor the list component code to become three components: list, item and input components (each with a specific purpose).

Requirements

1. Create item and input components.
2. Have them communicate with the list component via input properties.
3. Refactor the list code to use the components.

Challenge 4: Forms

Objectives

- Display an initial value for an item
- Edit the item and then validate the new value
- Log the changes

Requirements

1. Display the item's value in the input when it first displays.
2. Validate that the input element is required.
3. Display a message and change the input style if it is invalid.
4. When valid, obtain the edited data and log it to the console.

Challenge 5: Services & Http

Objectives

- Implement data access logic with an API

Requirements

1. Create a service to load and save data from an API
   a. Use *json-server* and the data provided in **code/labs/snippets/challenges/db.json**.
2. Inject the service into the list component and use it to load and save data
   a. Use Observables to return the data
   b. If you want too return the mock-contract-types data initially you will need to:
      1. import { of } from 'rxjs';
      2. And use it to return the data then map it as shown below.

```
return  of(CONTRACT_TYPES)
```

Challenge 6: Router

Objectives

- Use the Component Router

Requirements

1. Create two routes:s
   a. The path **projects** should load the project list.
   b. The path **contracttypes** should load the contract-type-list.
   c. Create navigation links to move between the two components.
2. Create a contract types module to hold all contract type related code and lazy load it only after you've visited the contract list page.

Challenge 7: Custom Directive

Objectives

- Create a custom attribute directive

Requirements

1. Notice that when you click on the first contract type and it becomes editable — the input has focus. If you click any other contract type (beyond the first) it becomes editable but doesn't have focus.
2. Create a custom directive that is applied to all inputs on a page and sets focus to that input when it is initialized.

   ☞ Hints: You will need to use an ElementRef to get a reference to the native element that you want to give focus. You will need to use a Renderer to invoke the native focus method on a DOM element.