Exercise 13 - Pipes

Objective

To use Angular's built in pipes to format data available in the view and to create our own pipe.

Overview

Initially in this exercise we will use 2 of Angular's built in pipes to present data in the format we want. These are very common transformations and hence why they are included as standard.

We will then create our own pipe which when used will shorten any lengthy text to the number of characters we specify, replacing the trimmed text with ellipses.

Instructions

Part 1 - Controlling how data is displayed

Course codes should be displayed in all uppercase. At the moment, this happens because they are stored in uppercase - but there is no control in place to ensure this happens going forward, nor would we want to add such control.

1. Add the uppercase pipe to the expression in the course-browse template that displays the course code.
2. Change the course service so that some of the course codes are in lower or mixed case and then check the application is still displaying them in uppercase as desired.

We also have some boiler plate async code in our components that we can simplify through the use of pipes.

3. Amend the **course-browse** component, specifically the making `courses` equal to the return of the `courseService.getCourses()` call in `ngOnInit` and removing the `.subscribe()` callback and replacing it with a `.pipe()`
    a. For **error handling**, as done in the subscribe method, this requires a call to `catchError`
        i. `catchError` takes a variable (called `err`) as the *argument to an anonymous function* and the body uses the **if...else chain** as before
        ii. `catchError` should **return** an *empty Observable*
    b. **Import** `Observable` from `rxjs` and `catchError` from `rxjs/operators`

4. Change the `Course[]` type from the `courses` declaration *at the top of the class* into an `Observable` of `Course[]`
5. The `ngFor` directive in our **course-browse** template cannot handle a Promise/Observable and so throws an error - add the `async` pipe to unwrap the value first

Feel free to change the InstructorGalleryComponent too, the solution shows the code needed (but it is extremely similar to that which you have just done for the course-browse)

Part 2 - Creating a custom pipe

6. Each course has a description which is in a text file within the assets folder. Add the descriptions for each course to the course service.
7. Add a new list item for description to the course-browse template, and a text-area for the course-editor template. How does the course-browse look now? Not good I imagine.
8. We're going to create a custom pipe that we can use to trim long text in situations like this.
9. Create a pipe called ellipses in our shared module. The transform function only needs to receive one argument, length.
10. Import the shared module into the courses module.
11. The EllipsesPipe transform should return the original string, cut off at the length specified and appended with ellipses (...) - it should also return the original value if the value is not a string or if the length is not specified.
12. Now add the EllipsesPipe into the template for the description, giving it a suitably small length (85 is a good starting point)