# Computing, Year 3 Rich Web Applications

# Lab8 Dependency Injection

## Overview

In this lab, you'll refactor the Films'R'Us application so that it takes advantage of Angular dependency injection.

## Roadmap

There are 4 exercises in this lab.  
Here is a brief summary of the tasks you will perform in each exercise; detailed instructions follow later:

1. Injecting the film service into components
2. Defining a new component to input film reviews
3. Implementing a logger for film reviews
4. Configuring the logger to write brief or verbose messages

## Exercise 1: Injecting the film service into components

Refactor the application so that the film service is injected into component(s) where needed. Think about which component should define the provider for this service – the placement of the provider determines the scope of the injected service.

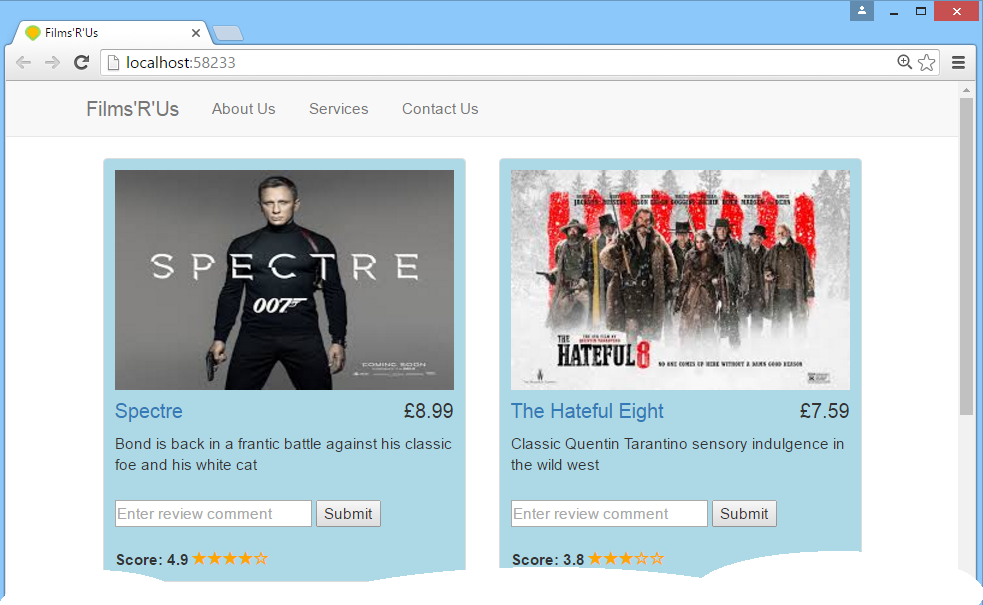
Build and run the application as usual, i.e. open a Command Prompt window and run the following command:

npm start

If all is well, your code should compile without any errors and the browser should display the films as normal.

## Exercise 2: Defining a new component to input film reviews

Enhance the application so that it allows the user to enter a review comment for a film. For example, display a text box and a button for each film as follows:



I suggest you define a new component, e.g. named ReviewComponent, to encapsulate this new UI and behaviour. The component should keep a record of all the review comments for a film. Also, display the number of reviews so far (unless it's zero), with a nice little tool tip for extra merit:



**Before you try to test your work, see the important notes on the next page…**

## Important notes:

## In order to make the text boxes work, you'll need to make use of two-way data binding. As you might recall from the "Data Binding" chapter earlier in the course, two-way data binding requires you to add FormsModule to your application module. This will entail code changes in the following 3 files:

## package.json Add @angular/forms to the dependencies property. Then re-run npm install so this new package is downloaded.

## systemjs.config.js Add @angular/forms to the map property. Make sure you specify the correct location for this module.

## app/module.ts Add FormsModule to the imports property in the @NgModule decorator.

## Now you can run npm start again, to test your work.

## Exercise 3: Implementing a logger for film reviews

Define a new service, e.g. named ReviewTrackerService, which logs all film reviews (e.g. to the browser console window). Should this be a singleton service? Should it be stateless? Where should it be injected? You decide…

## Exercise 4: Configuring the logger to write brief or verbose messages

Enhance the way the global logger is instantiated, so that it writes either verbose messages or brief messages to the console window. You should use a factory provider to govern the mode of the global logger.

Run the application twice:

* The first time, configure the logger to display verbose messages.
* Then edit the configuration to display brief messages, and re-run the application.

Verify that the logger writes the correct type of messages in each case.