Better React Components

Goal

The goal of this lab is to create better React components.

The Mission

In this lab you will be updating a movie editor build with React. The application is already functional and will let you browse through the list of known movies and select one to edit. However the complete application with all its logic is coded into a single component.

Your job is to split this single App component with all logic into separate components. Each component should have a clear responsibility and delegate other responsibilities to other components.

You can start the application by running the start.bat file in the root folder. When you do a small Node.js server will start. Next the default browser will start to display the index.html page. Please note that the first time you start this will take some time as number of NPM packages will download. This requires an internet connection where the NPM registry is not blocked by a firewall.

Note: The **FINISHED** folder contains a finished version of this lab as a reference.

Type it out by hand?

Typing it drills it into your brain much better than simply copying and pasting it. You're forming new neuron pathways. Those pathways are going to help you in the future. Help them out now.

Assignment 1

In the first part of this exercise you will split the App component into different components responsible for rendering different parts of the user interface. We will leave all state management in the App component for now. This is not optimal and something we will fix in the second part of this exercise.

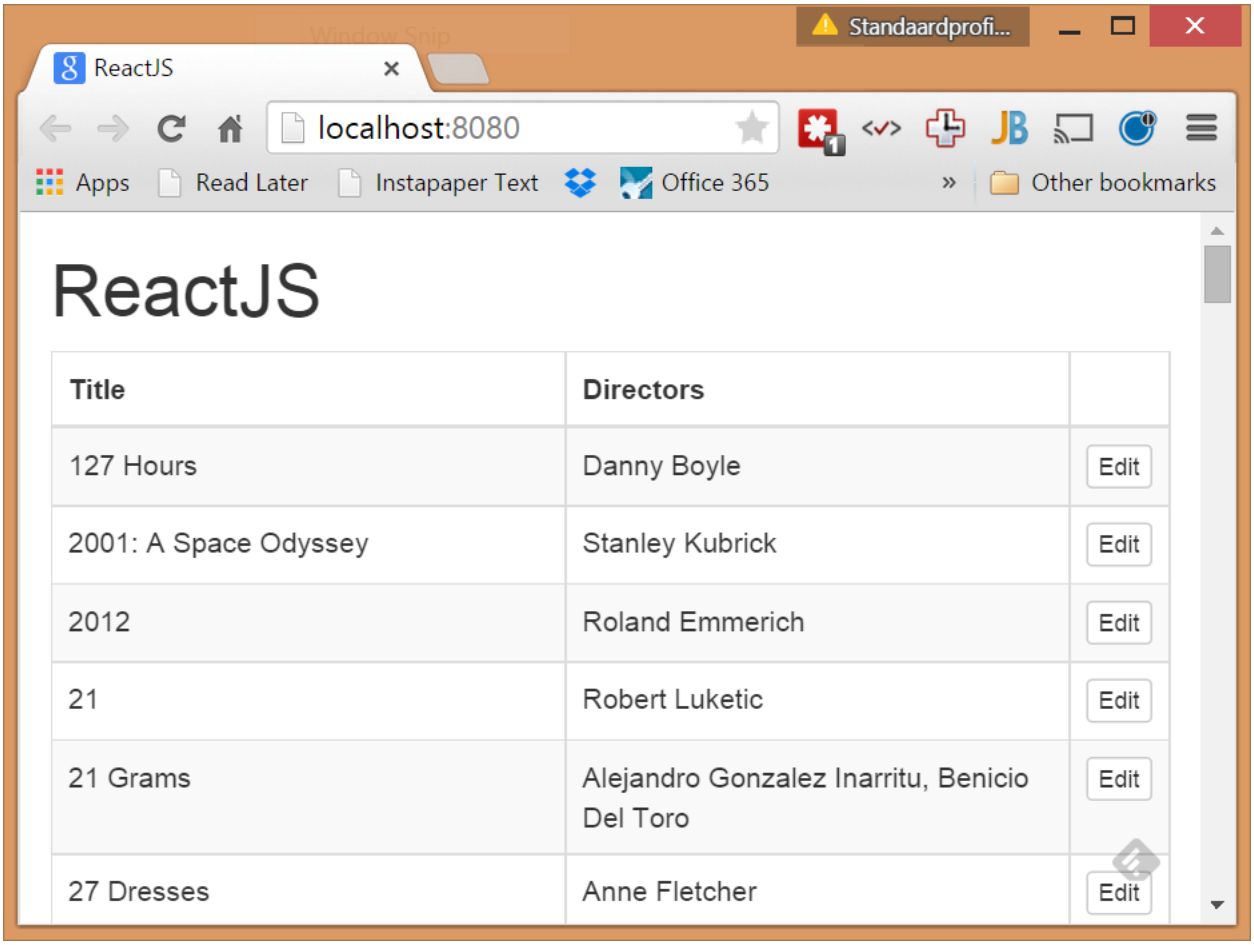
Open **APP.JS** in the **BEGIN/APP** folder and observe that the *render()* function really consists of two parts of which only one will execute at a time. The first part of the *render()* is concerned with rendering the data entry form. The second part is concerned with rendering the table of movies.

The first step is to refactor the movies list.

Create a new file named **MOVIESLIST.JS** and add a component *MoviesList*. This component will be responsible for rendering the table of movies. As this will be a really simple component you can use a pure stateless function here. Move the code to render the html table of movies from *App* into the new *MoviesList* component. Make sure to add *proptype* definitions for the required array of movies and the *toEditMode()* function. Update the *App* component *render()* function to use this new component.

The next step is to further simplify the *MoviesList* component by separating the rendering of each movie row. Create a new **MOVIESLISTROW.JS** and add a class *MoviesListRow* to it. This component can also be a pure stateless function. Implement the function by moving the relevant code from the *MoviesList render()* function. Add the *proptype* definition for the require props. Update the *MoviesList* component *render()* function to use this new component.

This should take care of the list of movies shown. Start the application and make sure it is completely functional.



The next step is to do the same with the movies editor.

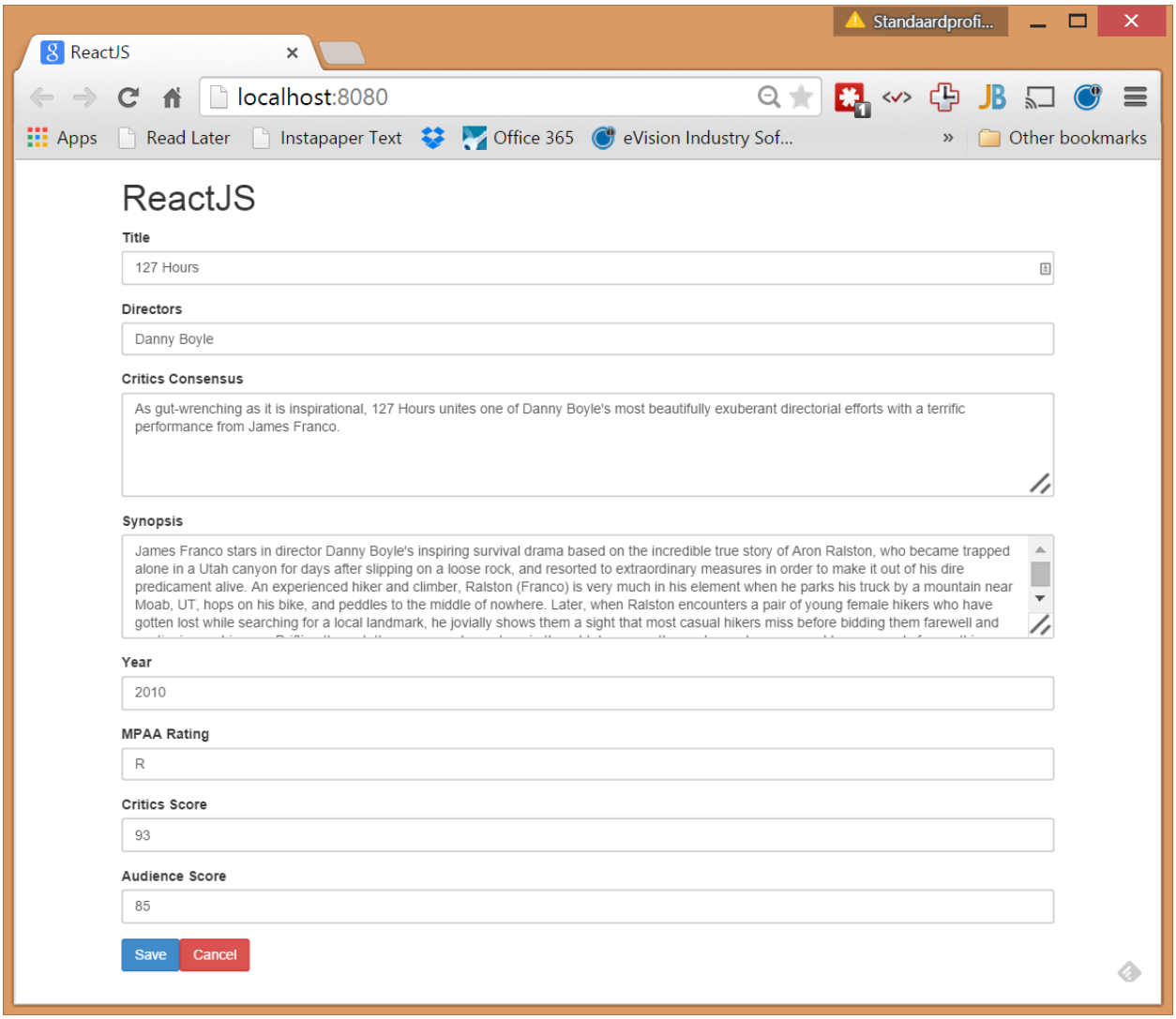
Create new file **MOVIEEDITOR.JS** and add a component *MovieEditor*. The *MovieEditor* will be more complex and needs to be a class. Move the code to render the movie edit form from *App* into the new *MovieEditor* component. Make sure to add *proptype* definitions for the required movie object and the *toEditMode()* and *onChange()* functions. Update the *App* component *render()* function to use this new component.

This movie editor still has a lot of duplication as there are multiple input and text area elements.

Create a new file **INPUTTEXT.JS** and add a component named *InputText* as a pure function. Move the markup for an *<input>* element plus its caption to this component. Make sure to define any required *proptypes* like before. Update the *MovieEditor* component to use this *InputText* component instead of the original *<input>* elements.

Create a new file **TEXTAREA.JS** and add a component *TextArea*. Repeat the actions you did for the input elements here to replace the original *<textarea>* elements.

This should take care of the movie editor. Start the application and make sure it is completely functional. You should be able to edit and save movies or cancel any changes made.



Assignment 2

You have now successfully split the rendering of the application into different components. However the App component is still responsible for all state management. In this section you will add two additional components to manage the movie state that is returned from the server. The *App* component will only manage the state need to determine if the list or edit form should be shown. In addition the *App* component will manage the id of the movie being edited in the *MovieEditor* component.

Create a new file **MOVIESLISTSTATE.JS** and add a component named *MoviesListState* to it. This component will be responsible for managing the array of movies to be show. It will use the previously created *MoviesList* component for the actual rendering. Move all code related to the movie list from *App* to *MoviesListState*. Replace the *MoviesList* with the *MoviesListState* component in the App component.

This should take care of the list of movies shown. Start the application and make sure it is completely functional.

Create a new file **MOVIEEDITORSTATE.JS** and add a component named *MovieEditorState*. Just like the before move all code related to a single movie instance from the *App* component to the *MovieEditorState* component. Replace *MovieEditor* with the *MovieEditorState* component in the App component. Move the AJAX request for sending the movie to the server during a save action from the *MovieEditor* component to the *MovieEditorState* component.

This should take care of the movie editor. Start the application and make sure it is completely functional. You should be able to edit and save movies or cancel any changes made.

With these changes we should have achieved the following: The *App* component should be only concerned with choosing between a list or editor view and the id of the movie in the editor. The *MoviesListState* component should only be concerned about loading the collection of movies and passing that onto *MoviesList*. The *MoviesList* component should only be concerned with rendering a list of movies. The *MovieEditorState* component should only be concerned with loading and saving the movie object. The *MovieEditor* component should only be concerned with rendering the movie editor form. Only the *MoviesListState* and *MovieEditorState* components should be doing any AJAX requests.

Optional assignment 3

The *proptype* definitions for the *MovieEditor* and *MoviesList* so far only require an object and an array. This means that passing an object with any shape would satisfy the definition even though the component would not be functioning correct. Your task is to improve these two *proptype* definitions so the actual shape of the object is validated.