Unit Testing

Goal

The goal of this lab is to unit test 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.

Your job is to create unit tests for the various React 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 create a basic setup to unit test components using Jest and Enzyme and create some unit tests for the *InputText* component.

Install Enzyme and its dependencies using:
npm install enzyme enzyme-adapter-react-16 react-test-renderer --save-dev

Next Create a **__TESTS__** folder in the **SRC** folder to contain the tests.

Create a file named **SETUPTESTS.JS** in the src folder. Add the required configuration code for Enzyme to it.

```
import { configure } from 'enzyme';
import Adapter from 'enzyme-adapter-react-16';

configure({ adapter: new Adapter() });
```

Create a new test named **INPUTTEXT.TEST.JS**. Import the *mount()* function from Enzyme and use it for a basic test to assert that the *InputText* component renders a HTML input element.

```
import React from "react";
import { mount } from "enzyme";

import InputText from "../InputText";

describe("The InputText", () => {
  it("renders a input", () => {
    const component = mount(<InputText>Label</InputText>);

    const inputs = component.find("input");

    expect(inputs.length).toBe(1);
  });
});
```

Run *npm test* from the command line and make sure your test passes.

Add two more tests for the *InputText* component. The first should assert that the *onChange* handler is called when the *value* is changed. The second should assert that the HTML input is updated when the *value* prop is updated.

Run *npm test* from the command line and make sure your test passes.

Assignment 2

In the first part of this exercise you will test the complete movie editor form.

Add a new file **MOVIEEDITOR.TEST.JS** and create a basic test to make sure the *MovieEditor* component renders an HTML form element.

Add test for the following functionality:

- The MovieEditor renders an HTML form element
- The MovieEditor should call onChange when the movie title is changed
- The MovieEditor should go to the list view when the cancel button is clicked

- The MovieEditor should save the movie when the save button is clicked

Run *npm test* from the command line and make sure your test passes.

Assignment 3

In the first part of this exercise you will add snapshot testing to prevent accidental changes of component rendering.

Add a new file named **RENDER.TEST.JS** and import the *renderer* from *react-test-renderer*. Use it to take a snapshot of the various components.

```
it("empty MoviesList", () => {
  const tree = renderer
    .create(<MoviesList movies={[]} toEditMode={() => {}} />)
    .toJSON();

  expect(tree).toMatchSnapshot();
});
```

Include tests for:

- An empty *MoviesList*
- The non-empty *MoviesList*
- An empty *MovieEditor*
- The non-empty *MovieEditor*
- The *InputText*
- The *TextArea*

Run *npm test* from the command line and make sure your test passes.

Optional assignment 4

In the first part of this exercise you will add an asynchronous test to assert that the *MovieListState* component loads movies using the *fetch()* API.

Add a new file **MOVIESLISTSTATE.TEST.JS** and add a test to make sure the *MoviesListState* component loads movies. Make sure to mock the global *fetch()* API using a Jest mock function.