React Challenge

1 Purpose

The purpose of this challenge is to access your skill in several languages and technologies used in modern web design, such as HTML5, CSS3, JavaScript, React, Redux, and Immutable.js.

2 Challenge

The challenge is to create a simple single page application (SPA) for managing a local "to-do list". The application should allow the user to;

- List the items on the to-do list, optionally filtered by state and sorted;
- Add an item to the list;
- Edit an item on the list (i.e. change its description or mark it as complete);
- Remove an item from the list.

2.1 Requirements and Recommendations

The application must be programmed in JavaScript (ES6/7), and you are encouraged to make use of modern JavaScript functionality such as **async/await**.

The use of the '*var*' keyword is forbidden.

You can use any public package in the NPM repository.

The use of libraries such as **lodash** or **underscore** is discouraged.

The use of Promise libraries, such as **bluebird** is forbidden.

The application must use the **React** library (https://facebook.github.io/react) for rendering its components.

For your convenience, we strongly recommend you scaffolded the application using the **create-react-app** tool (https://github.com/facebookincubator/create-react-app).

The application must use the **Redux** library (http://redux.js.org) for managing its global state (*i.e.* the "to-do list").

The application's data store must make use of the collections provided by the **Immutable.js** library (https://facebook.github.io/immutable-js).

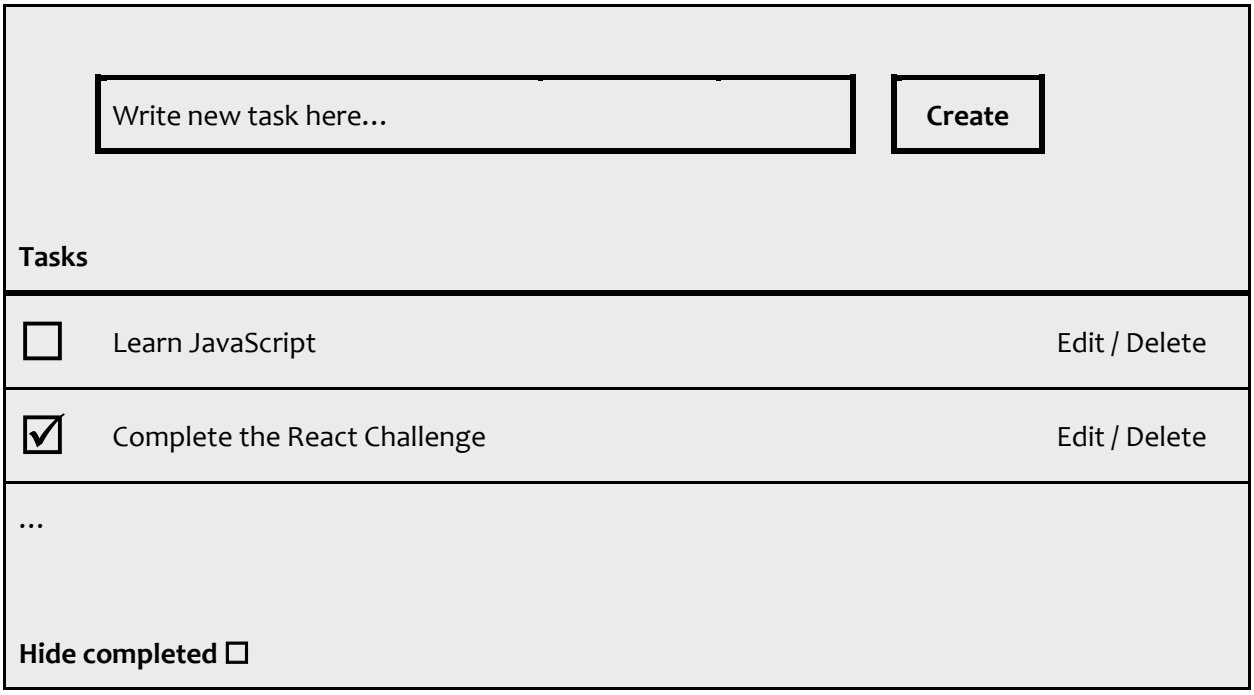
To compute derived data, such as sorted or filtered tasks, you must use the **Reselect** library (https://github.com/reactjs/reselect).

You are encouraged to use type annotations on your code, such as those provided by **Flow** (https://flow.org).

Having the application synchronize its global state with the browser's local storage, in order to allow the user to close and re-open the browser without losing information will be valued. You can investigate the **redux-storage** package (https://github.com/react-stack/redux-storage) if you which to pursue this endeavor.

2.2 Layout

The application should have a two sections. Both sections should be displayed in the same view, with a task creation form atop the "to-do list", as shown below:



2.3 User Interaction

The following user interactions must be implemented:

- **Creating a task** – The user writes the task on an input text field and clicks the "*Create*" button. The task is then added to the list and marked as incomplete.
- **Marking a task as complete/incomplete** – The user clicks the checkbox on the left of the list entry, which toggles its state.
- **Editing a task** – Clicking the "*Edit*" link on the right side of the list entry should allow the task name to be edited. You can display an input text field in-place, or re-use the task creation form.

- **Deleting a task** – Clicking the *"Delete"* link on the right side of the list entry should delete the entry.
- **Filtering the list** – Toggling the *"Hide completed"* checkbox on the bottom of the list should filter the completed entries from the list.
- **Sorting the list** – Clicking the *"Tasks"* list header should sort the list alphabetically from A to Z, clicking it again should change the order from Z to A, and a third time return to the default sorting (by creation date).

2.4 Styling

The styling of the elements is up to you, but you should show proficiency in positioning and sizing elements, styling text, colors, and borders.

Have fun!