Exercise 10 – Thinking In React

Objective

To be able to create a React application effectively.

Overview

The purpose of this exercise is to take a static version of an application and make it interactive by adding state, using the "Thinking In React" steps provided by Facebook. The first 2 steps have been completed (i.e. breaking UI into component hierarch and building a static version). The rest of the exercise is to complete steps 3 to 5. Solutions at the end of step 4 and 5 have been provided. To remind you, the application should:

1. Allow a user to enter text into a search box to filter a list of available products
2. Decide whether they wish to only display products that are 'in stock'

Note: the search will remain case sensitive unless you further modify the search to ignore case.

Part 1 – Project Setup

1.1. Navigate to **EG10_ThinkingInReact/starter** folder. This contains the static version of the application.

Part 2 – View the static application

2.1. Explore the project files, ensuring that you are clear as to why each component has been created and what it represents. If you are unsure, please refer to DG10_ThinkingInReact.

2.2. View the application in your browser.

Part 3 – Identify The Minimal (but complete) Respresentation of UI State

Remember to ask the following questions about state:

- Is it passed in from a parent via props?
- Does it remain unchanged over time?
- Can you compute it based on any other state or props in the component?

No actual coding is needed in this part but it would be good to make a note of what you decide.

Part 4 – Identify Where State Should Live

If you have successfully negotiated Part 3, you should have concluded that states in the application are:

- The search text that a user enters
- The value of the 'in stock only' checkbox

The next stage is to decide where in the application this state should be held. Run through the following process and then add state to the relevant components in the application.

- Identify every component that renders something based on state
- Find common owner component
- Either common component or component even higher up should own state
- If no component makes sense, create new component to hold state and add it into the hierarchy above the common owner component

Part 5 – Add Inverse Data Flow

You have probably added state to the `FilterableProductTable` component, including passing this state via props to the `SearchBar` and `ProductTable` components. You should also have modified these two components to include the values in the search form and changed the return of the `forEach` loop to return a blank row if the product name does not provide a match for the filter text OR the 'in stock only' condition is not met. If you unsure of what the code should look like, consult the solution for Step 4 and your instructor if you need to.

The final part of the application is to affect a change when the user changes the form. Although this is relatively little code, it does take more thinking about. The code already passes the values down the component hierarchy, we need to be able to retrieve what has been entered in the `SearchBar` component so that the `FilterableProductTable` can change the state of the application.

5.1. `FilterableProductTable` needs to pass a callback to `SearchBar` that will be fired whenever state should be updated

- Add a function `handleUserInput` to `FilterableProductTable` to set state
- Add `onUserInput` as a property of `SearchBar` that passes the `handleUserInput` function

5.2. `SearchBar` needs to handle an `onChange` event to answer the callback made in the previous step

- Add a constructor to `SearchBar`
- Define a function `handleChange` that provides arguments for the callback (i.e. the value of `filterTextInput` and the status of the `inStockOnlyInput`)
- Add a `ref` attribute to the text input that returns the result of an anonymous function that sets the `filterInputText` to `input`
- Add an `onChange` attribute to the text input that calls `handleChange`
- Add a `ref` attribute to the checkbox that returns the result of an anonymous function that sets the `inStockOnlyInput` to input
- Add an `onChange` attribute to the checkbox that calls `handleChange`

Saving the files and running the application should result in a working application.

Appendix

Thinking In React documentation:

https://facebook.github.io/react/docs/thinking-in-react.html