Exercise 04 – Creating Components with JSX

Objectives

To be able to create some components to create a static version of the FilterableProductTable.

Overview

This exercise will take a basic React application and make some components to start to put together a static version of the FilterableProductTable discussed in the Thinking in React chapter. We will not worry about the data in the table and how many rows it has as long as we have at least one of each type.

Check that the starter project runs correctly

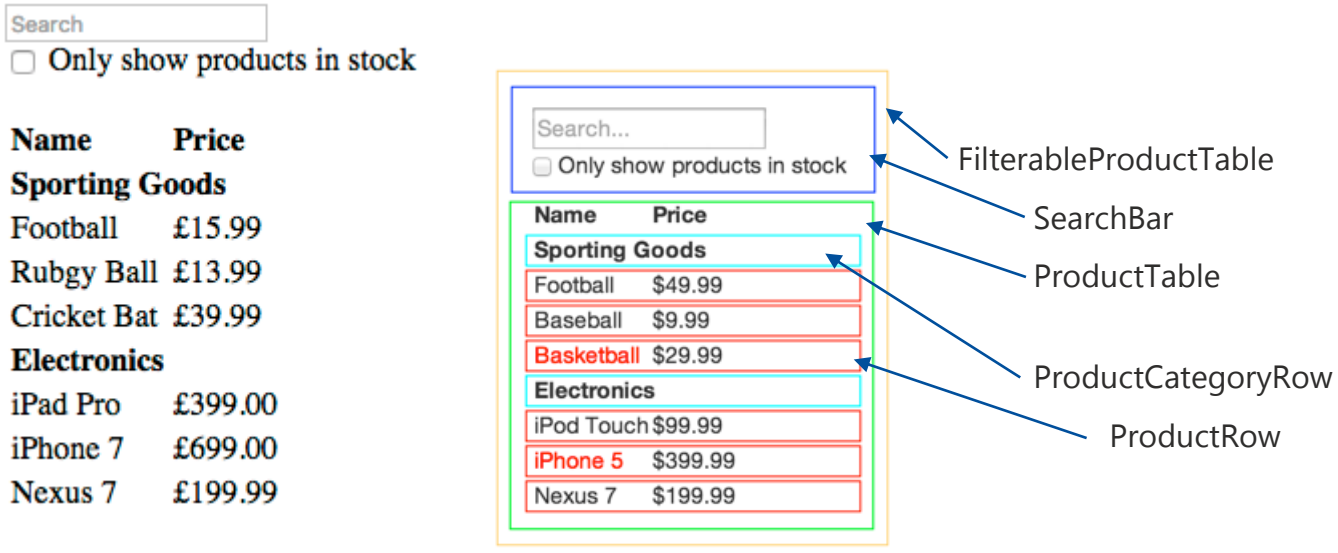
1. In the command line, navigate into the **starter** folder for **EG04**.
2. Run the command:

```
npm start
```

3. Verify that the browser outputs a heading **Filterable Product Table** with a line underneath it.

Wireframe and Mock Data

The wireframe for the table is on the left and component hierarchy on the right.



The data that is to be used in the table is as follows:

```
[
    {category: "Sporting Goods", price: "£49.99", stocked: true,
name: "Football"},
    {category: "Sporting Goods", price: "£9.99", stocked: true,
name: "Baseball"},
    {category: "Sporting Goods", price: "£29.99", stocked: false,
name: "Basketball"},
    {category: "Electronics", price: "£99.99", stocked: true,
name: "iPod Touch"},
    {category: "Electronics", price: "£399.99", stocked: false,
name: "iPhone 6"},
    {category: "Electronics", price: "£199.99", stocked: true,
name: "Nexus 7"}
];
```

This data will be used in the application once we have examined how to get data into applications. For now, we are satisfied that we know what the data is!

Part 1 - The FilterableProductTable Component

We have a hierarchy of components defined through the analysis of the wireframe. The outermost component is the **FilterableProductTable**. This is going to be created as a class component (for reasons that will become clear later in the course!) It will be placed inside the return of the render method in the App component and will contain the other components needed to create the table.

1. In the **src** folder, create a new file called **FilterableProductTable.jsx**.
2. Add an **import** for **React** to the top of the file.
3. Define a *default exported* **class** called **FilterableProductTable** that **extends React.Component**.
4. In the class, define a **render** method. It should **return** (for the time being) an empty **div** element.
5. Save this file.

Now we have a **FilterableProductTable** component, this needs to be placed in the **App** component. There are 2 stages to this - making the **FilterableProductTable** component available in the file via an import and then actually asking **App** to render the component.

6. Open **App.jsx** and add an import statement for **FilterableProductTable**.
7. Under the **<hr>** element in the render method, add the component:

```
…
    <hr>
    <FilterableProductTable />
…
```

8. Save the file and check the output in the browser.

Inspection of the page in the **elements** tab should reveal that there is an empty **div** in the **div** with an **id** of **root**. Inspection of the **React** tab should show an **App** component with the **h1**, **hr** and a **FilterableProductTable** component.

Part 2 - The SearchBar component

The **SearchBar** component will contain a text box input and a checkbox. This will be created as a class component (again, for reasons which will become clear as we go through the course). Once created, we will import and use the component inside the **FitlerableProductTable**.

1. In the **src** folder, create a new file called **SearchBar.jsx**.
2. Add an **import** for **React**.
3. Create an **export default class** called **SearchBar** with a **render** function that returns:
   a) A **form** element (with no action) containing:
      i) An **input** of **type text** and a **placeholder** set to **"Search..."**;
      ii) A **p** with an **input** inside of **type checkbox** and a text of **"Only show products in stock"**.
4. Save the file.

To use this in the **FilterableProductTable**, we have to import it into the component and ask its render method to use it.

5. Open **FilterableProductTable.jsx** and add an import for **SearchBar**.
6. Inside the empty div add the **SearchBar** component.

```
<div>
    <SearchBar />
</div>
```

7. Save the file and check that the **SearchBar** appears on the page.

Part 3 - The ProductTable

The **ProductTable** component is going to be the placeholder for the table. It needs to contain the main heading row and then the relevant **ProductCategoryRow** or **ProductRow**. The ProductTable will be a functional component.

1. In the **src** folder, create a new file called **ProductTable.jsx**.
2. Add an **import** for **React**.
3. Create a **const** called **ProductTable** that is set to an arrow function (with no parameters) that **returns**:
   a) A **table** element containing:
      i) A **thead** element containing a **tr** that has:
         (1) A **th** element with the content **Name**;
         (2) A **th** element with the content **Price**;
      ii) A **tbody** element
4. Export **ProductTable** as a default.
5. Save the file.
6. Import **ProductTable** into **FilterableProductTable** and place it below the **SearchBar**.
7. Check the output of the browser.

Part 4 - The ProductCategoryRow Component

To keep things simple for the time being, we are going to have a single product category in our table. To get our output to look like the mock, we need to know more about how React deals with data - that will come later!

1. In the **src** folder, create a new file called **ProductCategoryRow.jsx**.
2. Add an **import** for **React**.
3. Create a **const** called **ProductCategoryRow** that is set to an arrow function (with no parameters) that **returns**:
    a) A **tr** element with:
        i) A **th** element that *spans 2 columns* and has the content **"Electronics"**.
4. Export **ProductCategoryRow** as default.
5. Save the file.
6. Import the **ProductCategoryRow** into **ProductTable** and render them inside the **tbody**.
7. Save the file.
8. Check the output of the browser is as expected.

Part 5 - The Product Row.

Fly solo on this one! The **ProductRow** needs to be a table row with a cell that has the name of one of the electronics products and another cell for its price.