COURSEWARE

Component Hierarchy

Contents

Overview

In this module, we will look at React's Component Hierarchy.

Thinking in React

React makes developers think about the applications that are being built as they build them.

Facebook's recommendation is to follow these steps when building apps using React:

1. Start with a mock
2. Break the UI into a component hierarchy
3. Build a static version in React
4. Identify the minimal (but complete) representation of UI state
5. Identify where your state should live
6. Add inverse data flow

All UIs to be built should have a mock up or a wireframe, and data should be available in a suitable format too, such as JSON.

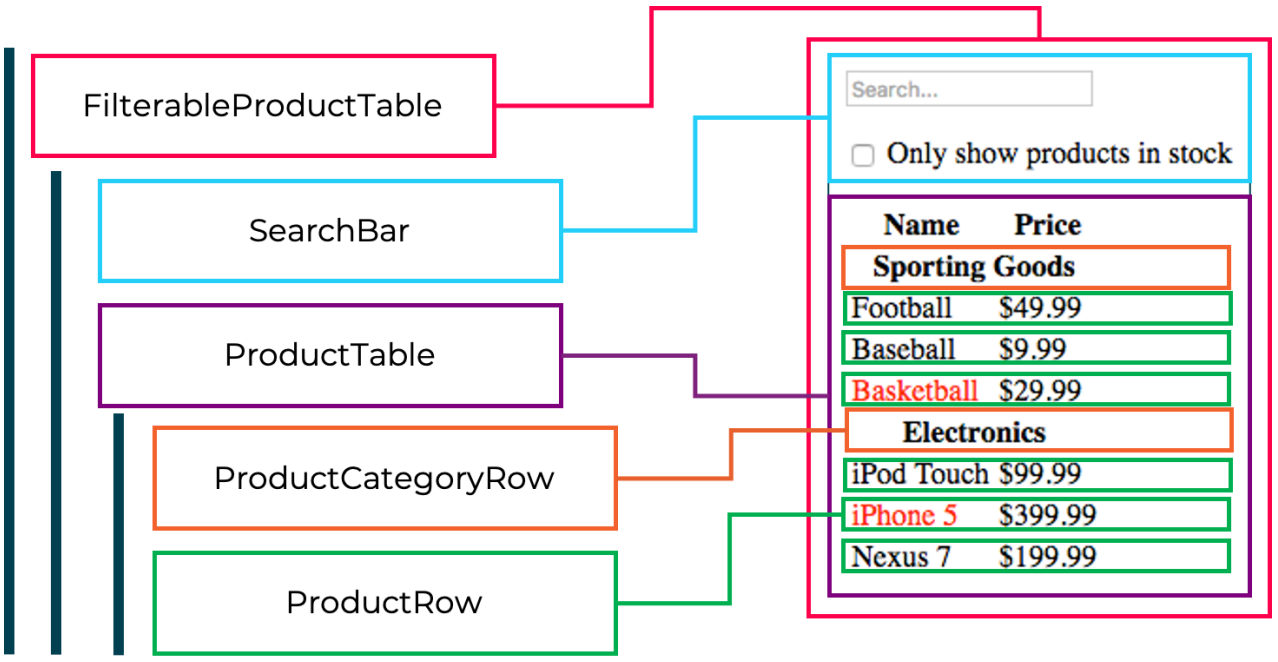
```
  [
    {category: "Sporting Goods", price: "$49.99", stocked: true, name:
"Football"},
      {category: "Sporting Goods", price: "$9.99", stocked: true, name:
"Baseball"},
      {category: "Sporting Goods", price: "$29.99", stocked: false, name:
"Basketball"},
      {category: "Electronics", price: "$99.99", stocked: true, name: "iPod
Touch"},
      {category: "Electronics", price: "$399.99", stocked: false, name: "iPhone
5"},
      {category: "Electronics", price: "$199.99", stocked: true, name: "Nexus
7"}
  ];
```

Hierarchy of Components

Consider the following very basic UI for the above JSON code;

Considering this is our UI, how would we break this down into components?

This is the core design principle of React; break down our UI for single-functional components that are in a hierarchy.



Building Static Versions

We should aim to build a version of our app that takes our data model and renders the UI without interactivity first.

This allows us to decouple the view and the interactivity, which is good as each require different amounts and methods of maintenance.

Best practice is to build components that reuse other components and pass data using props.

Building in this method helps establish a library of reusable components to render each data model.

Tutorial

We will be creating a header component for a website.

1. Create a new file called **Header.jsx**
2. Create a Function component called `Header` that has no parameters.
3. The return of the component should have wrapping <header> and <nav> elements:

    - <nav> should have classes navbar navbar-expand-sm;
    - A link to https://www.qa.com with a class of navbar-brand, a target of _blank and a rel of noopener noreferrer;
    - The link should contain an image whose src is any image, with a width of 100;
    - A sibling link to / with a class of navbar-brand and text of Todo App.
4. `export` Header as `default`.
5. Save the file.
6. Open your `App.js` file, and add an import for Header.
7. Within the outer , add a child of <Header /> as an older sibling of the inner .
8. Save the file.

▶ Header.jsx

Exercises

1. Create another component, this time for the footer of the page.
2. This should have a constant called `AnotherComponent` that takes no arguments, and returns 3-4 `<p>` elements with the use of React Fragments.
3. Ensure to export `MyComponent` as a default and add the component to your `MyComponent` file.
4. Wrap your `h1` tag and your `AnotherComponent` tag in a React Fragment (ensuring `AnotherComponent` is imported).
5. Save and run to see the result.