Unit 8: React: Lab 8 – Order (Context)

Assigned: 02/08/23

Due: 02/09/23

Completed: 02/09/23

**LMS:** <https://lms.grandcircus.co/mod/assign/view.php?id=22825>

**Google Doc:** <https://docs.google.com/document/d/12XYx_E-CEnTOjMYpJE3DBV9oh3bfDcNU_9M886TCko8/preview>

**GitHub:** <https://github.com/clintmsmith/GrandCircusLabs/tree/main/Order/orderlab>

**Task:** Create a React app [like this example](https://context-exercise.surge.sh/) where a user can add food to their order. Use Context to give various components access to the order array and the functions that facilitate the order array.

**Build Specifications:**

1. Components:
   1. Header
   2. MenuList
   3. MenuItem
   4. Sidebar
2. Create an interface called Item which consists of:
   1. id: string
   2. name: string
   3. description: string
   4. calories: number
   5. price: number
   6. vegetarian: Boolean
3. The App component will contain the Header, MenuList, and the Sidebar components.
4. The MenuList component will contain an array of menu items (see below).
5. The MenuList will contain the MenuItem component. Use the map method to produce as many MenuItem components as there are items in the menu.
6. Make a context folder that contains:
   1. OrderContext.ts
   2. OrderContextProvider.tsx
7. OrderContext.ts creates context.
   1. Hint: Model your context by implementing an interface with these properties.
      1. order: Item[];
      2. addItem: (item: Item) => void;
      3. removeItem: (id: string) => void;
8. OrderContextProvider.tsx should provide context to all its children. This is where the order state and the functions to update the state will live.
   1. Don't forget to add OrderContextProvider to your index.tsx!
9. The useContext hook should be used in the Header, MenuList, and Sidebar components.

**Menu Array:**

Text

Description automatically generated

Text

Description automatically generated

After creating our React App, we then added some folders in our src folder for the various parts of our app.

Graphical user interface

Description automatically generated with medium confidence

Components included four (4) files and we started with basic **function** skeletons

Graphical user interface, text, application

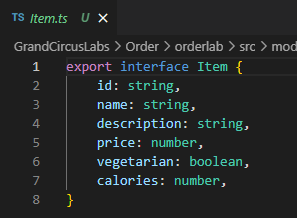
Description automatically generated

We then copied the data above and put it into the data folder

Text

Description automatically generated

We then used this data to complete our **interface** in the models folder.

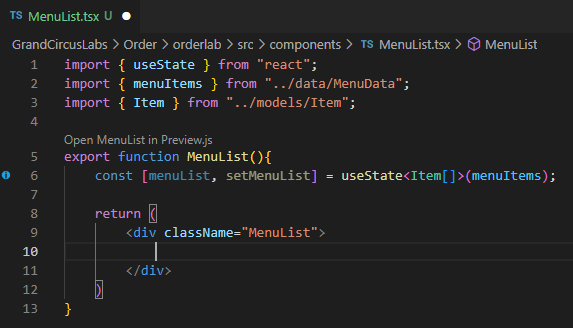


We then added our components to App.tsx and used the Quick Fix to import them.

Text

Description automatically generated

We then added a state to MenuList.tsx



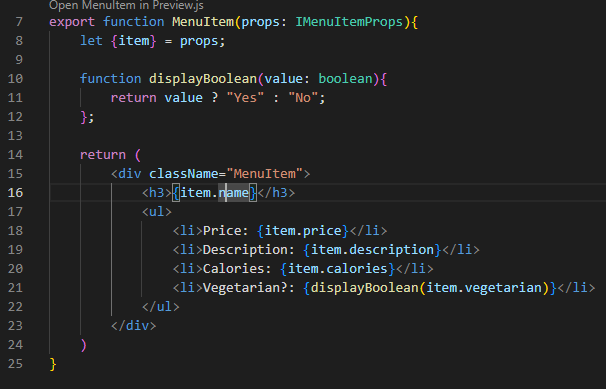
We then moved over to our MenuItem.tsx

In the Social Posts lab we used {props.post.title}, but we can use destructuring to dig down a little bit and save ourself some typing.

Text

Description automatically generated

So in this lab we did this by creating the variable “let {item} = props”. In our return we can then cut out the “props” portion.



We then wanted to use .map method for our MenuItem(s). Jonathan mentioned that you could use the index for the key, but generally you shouldn’t as it can change. The id is more stable and unique. Something to keep in mind for interview questions.

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Here is the structure of our App. You can see that Sidebar and Header are not part of the same hierarchy as MenuList and MenuItem. But we want an action taken in MenuItem to influence Sidebar, so how do we do that? We can do that with **context**. Otherwise we would have to pass info up from MenuItem to MenuList to App and then finally down to Sidebar and that’s a lot of steps.

Diagram

Description automatically generated

So all of the children essentially will fall into a zone of Order Context, which allows them to communicate directly. You could add a component later that falls outside of the Order Context, much like we see with scope in functions.

Diagram

Description automatically generated

Within our ‘context’ folder we added three (3) files. Notice the extension of the files.

Text

Description automatically generated with medium confidence

Within OrderContextModel.ts we created an interface. Here this takes an order, which is an array of Item[], and then calls functions with the function signature (name, what it takes in, and then what it returns).

A screenshot of a computer

Description automatically generated with medium confidence

In the OrderContext.ts we then used the OrderContextModel’s model and gave them their default value. React has a built-in **function** ‘**createContext’** that we can import.

Text

Description automatically generated

We then built our **provider**.

Text

Description automatically generated

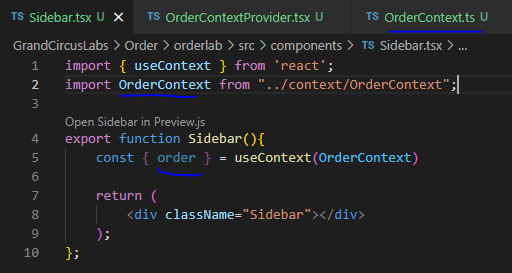
The reason we use OrderContext.Provider is consumer / provider. The consumers are the components that use this provider.

So, we’ve built the **provider**, but we still need to build the **zone** that can use this. To do that we add this **component** to our App.tsx and put the **components** we want to make use of this by putting them inside this **component** as **children**.

Text

Description automatically generated

Within our Sidebar.tsx we want to make use of our **context** and we want to grab the **order**.



A picture containing text

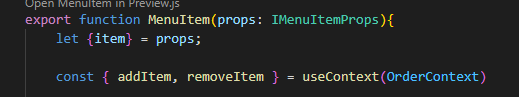
Description automatically generated

We want to return an ordered (numbered) list



We want to do the same thing in MenuItem.tsx, importing useContext and OrderContext, then we add a const that allows us to make use of those. Finally, we added buttons that call on those with onClick.





Text

Description automatically generated

Remember that in OrderContextProvider addItem takes in an item, but removeItem takes in the item.id. James noticed that if you add multiple of the same item, then removeItem, our code will remove all items that share that id. We would have to add code to get rid of just the last instance of the id.

A screenshot of a computer

Description automatically generated with medium confidence

To calculate the order total, we went back into the Sidebar.tsx and made use of .reduce() method. This makes use of a **variable** “prev” that holds a value. It starts at 0, then finds the first item.price and adds it, and repeats until it has gone through the entire array.



Lastly, we want to add the Order Count element to our Header.tsx. We do the typical imports and create the const, but then can simply use .length because order is an array, so the .length just counts the number of objects in the array.

Text

Description automatically generated

In the example one they added some functionality to the button so that only after you clicked ‘Add Item’ did the ‘Remove Button’ become visible.