React Information Flow Lab (CodeGrade)

(https://github.com/learn-co-curriculum/react-hooks-information-flow-lab) (https://github.com/learn-co-curriculum/react-hooks-information-flow-lab/issues/new)

Learning Goals

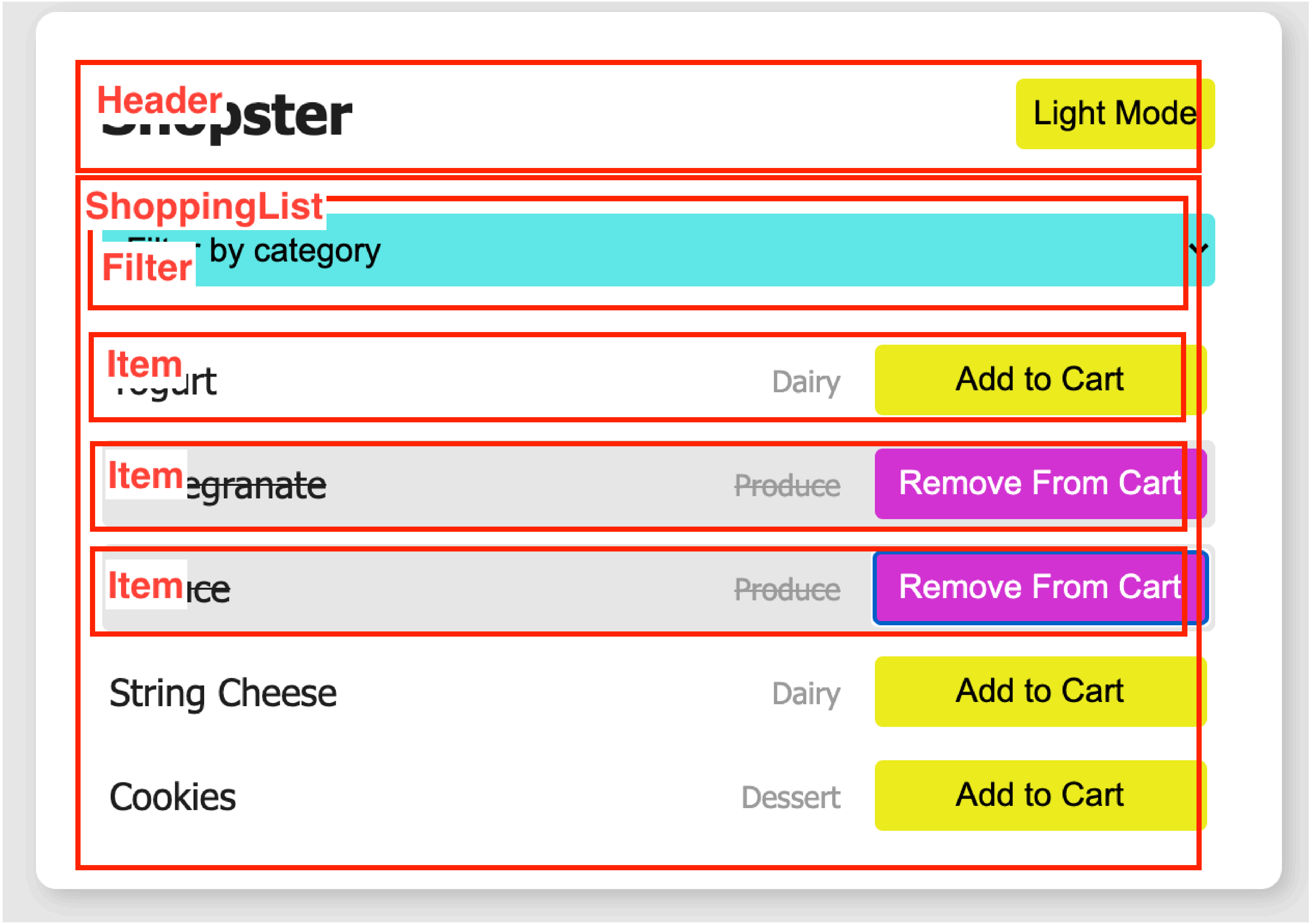
- Use callback functions to update state in a parent component
- Move state based on which components need access to it

Introduction

Last time we worked on the Shopping List app, we ended up with the following component hierarchy:

```
App
└── ShoppingList
    ├── Item
    ├── Item
    └── Item
```

However, it would probably make more sense to break the app down more like this:


Header
Shopster
Light Mode
ShoppingList
Filter
Filter by category
Item
Yogurt
Dairy
Add to Cart
Item
Pomegranate
~~Produce~~
Remove From Cart
Item
Lettuce
~~Produce~~
Remove From Cart
String Cheese
Dairy
Add to Cart
Cookies
Dessert
Add to Cart

```
App
├── Header
└── ShoppingList
    ├── Filter
    ├── Item
    ├── Item
    └── Item
```

Your goal is to refactor the code from the previous shopping list lab to match the component structure in the image above. As you work on refactoring the code, keep in mind:

- Which components should hold **state**? Which components need access to that state?
- How can we update state in one component from a child component?

Also, remember what we learned in the previous lesson:

- For information to propagate *down* the component tree, parents pass `props` to their children
- For information to propagate *up* the component tree, we typically invoke callbacks that were passed from parents to children as `props`
- Components of the same level (sibling components) cannot communicate directly! We can only communicate up and down the component tree. So if multiple components need to share the same information, that state should live in the parent component (or a more general ancestor).

Deliverables

Header

Create a `Header` component by refactoring the `<header>` element out of the `App` component. Clicking on the `<button>` inside of the `Header` component should still toggle dark mode on and off.

You will need to pass a *callback function* as a prop called `onDarkModeClick` to the `Header` component in order for the test to pass.

Filter

Create a `Filter` component by refactoring the `<select>` element out of the `ShoppingList` component. Changing the selection in the dropdown should still change which items are displayed in the shopping list.

You will need to pass a *callback function* as a prop called `onCategoryChange` to the `Filter` component in order for the test to pass.

Resources

- [React Docs on `useState`](https://reactjs.org/docs/hooks-state.html) (https://reactjs.org/docs/hooks-state.html)

This tool needs to be loaded in a new browser window

Load React Information Flow Lab (CodeGrade) in a new window