React Lab Assignment (30 points)

Lab Setup

Please complete lab setup before attempting the lab exercises. The reference material slides can be found on Blackboard.

For this lab, you'll need Google Chrome and a **text editing tool** such as Sublime, Atom, VSCode, etc.

First, we want to ensure that we have Node.js and React.js on our computer:

1. Check that you have Node and npm installed by running
2. `node --version` and `npm --version` in your terminal. You should have:
    a. node version >= 14.0.0
    b. npm version >= 5.6.0
    c. If your terminal prints that it doesn't recognize "node", you can install it here; make sure to get the appropriate version for your computer

What is React?

React is a UI library developed at Facebook to facilitate the creation of interactive, stateful & reusable UI components. As Wikipedia puts it, React allows developers to "create large web-applications that use data and can change over time without reloading the page."

When JavaScript interacts with an HTML document, it is interacting with the **Document Object Model**, or **DOM**, which is a tree containing all of the HTML elements of that document. React uses a concept called the Virtual DOM to selectively render nodes on this tree. In essence, this allows React to interact with as little of the DOM as possible while still adequately making changes to the state of a webpage. (If this doesn't make sense to you, read on!)

As an example, pretend your website renders the full body of a person. The DOM structures the information like this:

- Person
    - Head
        - Face

- Eyes
- Nose
- Ears

If you wanted to change a small feature on this site, such as give them Clippy's eyes, the website would normally re-render completely -- even the features that didn't change.

React, on the other hand, renders this change differently. First, it runs a "diffing" algorithm that identifies what features have changed. Then, it reconciles and updates the parts of the DOM that have changed.

If the idea of the Virtual DOM is confusing, don't worry! It's difficult to wrap one's head around, and knowing exactly how the DOM and Virtual DOM function aren't necessary for this lab. Just know that React is used because it optimizes DOM interaction.

If you're interested in learning more about what the DOM is and how it functions, this link is very helpful.

Goal of this lab

This lab aims to provide you with a simple introduction to creating your own React app! You will learn all about **JSX, components, props, state, events, and how files work in React to create a running app.** This lab can be **extremely** useful for the UI implementation part of project #2, so make sure to save your code to reference later!

Creating an app

Once you have React.js installed, we can use the create-react-app command to create the files necessary to build an app:

1. In your terminal, move to the directory you want your app to be created (ie. `cd Desktop`).
2. Run `npx create-react-app my-app` to create your app.
3. Then run `cd my-app` to go into that directory
4. Run `npm start` to start your app cc

   a. **You do not have to rerun npm start when you make changes to the app's .js or .html files -- the app will automatically update when you save your files!**
5. In your browser, go to http://localhost:3000/ to see your app live!
6. Now open up any text editor of your choice and open the `my-app` folder
7. All of the changes that we will be making will be in the `src` directory
8. Reference tutorial to create a new react app: here.

Stencil code

Stencil code for the files you are required to add/change for this lab can be found here.

In the my-app folder created before, open the src folder. Download the 5 files (you can disregard the README file) from the link above, and move them into the src directory. If it asks whether you want to overwrite duplicate files, click 'replace'.

Now, you should have the following files in your src directory:

App.css	FilteredList.jsx	index.js
App.js	HelloWorld.jsx	
App.test.js	List.jsx	
Counter.jsx	index.css	

JSX

You may notice that there are files with the extension '.jsx'. You also may notice that the syntax seems to be a blend of Javascript and HTML. This syntax is called **JSX**, and it is a Javascript XML syntax transform. This lets you write HTML-ish tags in your Javascript.

Note that this is not exact HTML—you are really just writing XML-based object representations. For regular html tags, the `class` attribute is `className` and the `for` attribute is `htmlFor` in JSX because these are reserved words in Javascript. While you can certainly use React without JSX, we highly recommend that you use JSX.

A more in-depth explanation of JSX can be found here.

Components

React's basic building blocks are called **components**. Components are independent, reusable classes that compose different parts of your UI. React applications are made up of components, many of which are rendered within other components.

It is generally a good idea to write each component in its own file. For example, the HelloWorld component is implemented in **HelloWorld.jsx**. To use it elsewhere, simply import the component at the top of the file that you wish to use it in, as we did at the top of **App.js**. Once the component is imported, we can add it to the UI using:

```
<HelloWorld />
```

in the `render()` method. Note that you must export the component at the bottom of its own file in order to be able to import it elsewhere.

Props

When we use our defined components, we can add attributes called props that are passed from parent components. These attributes are available in our components as this.props and can be used to render dynamic data.

Let's say that we want to add a name prop to the HelloWorld component, which we passed to the component in **App.js**. When we add the HelloWorld component into the UI, we can pass the name prop:

```
<HelloWorld name={'Gabby'} />
```

We can later access this name prop in **HelloWorld.jsx**, using `this.props.name`.

Lifecycle Methods

The `render()` method is the only required method for creating a component, but there are several lifecycle methods and specs we can use that can be helpful which you can read about here.

State

Every component has a state object and a props object. Initial state should be set in the `constructor()`, but can be set or reset elsewhere using the `setState()` method. Calling `setState()` triggers UI updates and is the bread and butter of React's interactivity.

For example, in the constructor of a component, we can initialize the state object with the key 'color' using:

```
this.state = { color: "blue" };
```

Then, when we want to change the value of 'color' to yellow, we can use:

```
this.setState({ color: "yellow" });
```

Events

React also has a built in cross browser events system. The events are attached as properties of components and can trigger methods.

For example, let's say we have a function called carryOutAction() that is defined in the component, that should be called when a user presses a button. In the render() method of a component that will display this button, we can add the following:

```
<button onClick={this.carryOutAction}></button>
```

Note: When calling a React function inside HTML code, the function must be wrapped with curly brackets (i.e. { }), unlike in JavaScript.

With this, when a button click is detected, it will call the `carryOutAction()` function.

Lab Exercises (30 points)

Exercise 1: Hello World

Ensure that you have all the stencil code downloaded from the link provided above.

For this exercise, you'll be able to get familiar with working with the different files in a React app. Follow the **HelloWorld TODOs** in **App.js** and **HelloWorld.jsx**. The goal is to get 'Welcome to cereal counter webpage by [student first and last name]' displayed on your webpage!

Exercise 2: Counter

For this exercise, we are going to create a counter, that will increment when users click a button. Follow the **Counter TODOs** in **App.js** and **Counter.jsx**.

Exercise 3: Filtered List

For this exercise, we are going to create a webpage that filters cereals based on their shape and features searching cereals by name.

You will be using a Dropdown menu for this exercise, so we need to install Bootstrap. From the my-app folder in your terminal, run

```
npm install react-bootstrap --save
```

Add the following css file right after `<meta charset="utf-8" />` in **my-app/public/index.html** for styling:

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.m
in.css"/>
```

Now, add the following script tags right after `<title>React App</title>` in **my-app/public/index.html** for styling:

```
<script
```

```
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popp
er.min.js"></script>

<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min
.js"></script>
```

Your task will be to add in a dropdown menu that will **filter out cereal by shape** (circular and square). When circular is selected, only circular cereal should show and when square is selected, only square cereal should show on the list. You will also have to make sure that the dropdown will work with the **search filter** so that the list will only show cereal that fulfills **both the search and dropdown filters**. We have provided you with an updated **FilteredList.jsx** with TODOs on how to implement. Additional resources might be helpful for this exercise.

Follow the **FilteredList TODOs** in **App.js** and **FilteredList.jsx**. Make sure all three of your filter selections work - **including the "All" dropdown** - in order to fully complete your Dropdown Button Filter.

Additional Resources

Facebook's official React tutorial
React video tutorial
React documentation
Facebook talk explaining the rationale behind using React
React Developer Tools (Chrome, Firefox)

Deliverables

1. Create a new directory (named: react-lab-[lastname]). Copy and paste the **src** and **public** folders from your my-app project directory into the new directory. Also copy and paste the **README** file with reference resources in the new directory. Submit a zipped version of the new directory via Blackboard by **03/08, 6 PM**.
2. Submit a **video** showing a live demo of all three exercises. Demo for Exercise 3 should include both search and dropdown filtering. Include a voice over in the video, if necessary to help viewers understand your actions.

Rubric

(5 points) Exercise 1: ‘Welcome to cereal counter webpage by [student first and last name]’ displayed on your webpage as a header element.
(10 points) Exercise 2: The counter successfully increments on a click of a button.
(15 points) Exercise 3: All the dropdown filter selections are working successfully to filter out cereal by shape. Dropdown works successfully with the search filter so that the list will only show cereal that fulfills both the search and dropdown filters.

Course Policies

All students should follow standards of conduct as discussed in UIC’s disciplinary policies (see syllabus for more details). As per the course policies, you should be doing this assignment independently without discussing it with other students in the class or elsewhere. Do not copy and paste code from online sources.