Creating the SearchBar.js

The main feature of this app is the ability to search for businesses using specific criteria. To create this functionality, we will be adding a searchComponent and linking up our components to work together.

1. In the `SearchBar/` directory (inside the components directory) create two new `files`
    - SearchBar.js
    - SearchBar.css
2. Add the CSS the from the `searchBar-style.txt` project file
3. Open `SearchBar.js` and import the `React Library` & `SearchBar.css` (in that order)

The search bar will communicate with the Yelp API, although we will implement that part of the setup later, we still need to create the structure needed to be able to communicate with the API.

Specifically, requests to the Yelp API must follow formatting and naming conventions set by the API. For example, the search bar should allow users to search businesses by: `Best Match, Highest Rated, & Most Reviewed`

To achieve this, you'll create an object with keys and values that conform to what the API expects to receive.

4. Create an object `sortByOptions` with three keys - remember the order is important and each key should be a string

    - Best Match
    - Highest Rated
    - Most Reviewed

5. Using the [documentation] (https://www.yelp.com/developers/documentation/v3/business_search), set the values to each of the keys. The `sort_by` entry in the table of the `"Parameters"` section will be helpful. The values should be strings.

Creating the SearchBar Component

We need to dynamically create the list items needed to display the sort options (Best Match, Highest Rated, Most Reviewed). This is to help future proof against potential changes to the Yelp API. We will do this using a *renderSortByOptions()* method

6. Use the React library to create a component called *SearchBar*. Don't add a render() method
7. Inside the body of the component declaration, create a method called `renderSortByOptions()`
8. Add a *return* statement(no parentheses)
9. To iterate through the object, you'll need to start by accessing the keys of the `sortByOptionsobject`. Call the keys() method on the JavaScript Object library. Pass in `sortByOptions` as the argument. - This will give us access to the keys
10. Now that you have access to the keys, you'll iterate through them using the `map()` method. Call the `map()` method by chaining it to the end of the line you just wrote. It should look something like this.

```
```

```
renderSortByOptions() {
return Object.keys(sortByOptions).map();
}
```

11. Pass a callback function to the *map()* method as an argument. The callback function should have one parameter called *sortByOption*. The callback function should also use arrow syntax.

```
renderSortByOptions() {
return Object.keys(sortByOptions).map(sortByOption => {

});
}
```

12. Inside of the callback function, access the *sortByOptions* values using the *sortByOption* parameter of the callback function. Store values in variable called `sortByOptionValue`

13. On the next line, return a `<li>` element. The list item should have an attribute called key set to *sortByOptionValue* (don't forget to use curly braces to inject JavaScript). The content of the list item should be `sortByOption`. Again, use curly braces to achieve the JavaScript injection.

```
return <li key=?> ? </li>;
```

Rendering the SearchBar Component

14. In the `<SearchBar/>` component, add the *render()* method

15. Inside of the .render() method, add a return statement with JSX that renders the HTML inside of the `search-bar.txt` project file.

    - Follow the guidelines below when you write the HTML (linked above) as JSX:
      - Change all `class` attributes to `className`.
      - Do not change the `class` values, as we will use them in the next step to add style to the search bar component.
      - Replace the comment with a call to `.renderSortByOptions()`

16. At the bottom of the file, export *SearchBar*.

Save and close all current files - This is less of a step & more of a *tip*

---

17. Open the `App.js` file in the `App/` directory. At the top of the page import each of the following:

    - The React App
    - App.css - Be sure to add the CSS from the `app-styles.txt` project file
    - Import the `<BusinessList/>` component

    - Import the `<SearchBar/>` component *Check and double check the paths to these components, VScode will help.*

18. The create-react-app command creates a default App component for you. It includes a `render()` method along with a return statement. We won't need the default App component, so let's make some modifications. Delete everything inside of the `return` statement.

19. Inside of the `App.js .render() method`, return JSX that renders the HTML from `app.txt` in moodle.

    - Follow the guidelines below when you write the HTML (linked above) as JSX:

        - Change the `class` attribute to `className`.
        - Do not change the `class` values, as we will use them to add style to the website.
        - Replace the comments with their corresponding components.

20. By default, the create-react-app command adds a default export in the class declaration of the App component, making it look like the following:

```
export default class App extends Component {
// Code
}
```

To be consistent across with how components are exported in the project, remove the `export default` from the beginning of the class declaration. Instead, export App at the bottom of the file (you've done this for all components so far).