Exercise 08 – Working with Datasets

Objectives

To be able to add to and remove from datasets, updating the display in real-time.

Overview

Continuing the Your Google Map Locations application that was created earlier, the next functionality to add to the application is to enable a user to save their favourite locations. The following specifications exist:

1. The location display should have something that a user can modify to indicate that the current location displayed should be classed as a favourite.
2. Favourite locations should appear in a list underneath the location display, with the following information recorded:
   a) The name of the location
   b) A timestamp of when the location was added
3. Favourite locations should be available whilst the browser cache has not been emptied and clicking on a favourite in the list displays its map.

Exercise Instructions

Check that the starter project runs correctly

1. In the command line, navigate into the **starter** folder for **EG08**.
2. Run the command:

```
npm install
```

3. Now run the command:

```
npm start
```

4. Verify that the browser outputs the **Your Google Map Locations** as it did at the end of the previous exercise.

Adding to and removing from favourites

Prepare CurrentLocation to accept a 'favourites' property

To display an icon next to the location, **CurrentLocation**'s **render** method will need to be able to recognise whether the **address** is a favourite or not. This will be passed in from the **App** component when it tries to **render** this component. It will be dealt with later in the exercise, but for now, the code to display an icon will be put into **CurrentLocation**.

1. Open **CurrentLocation.jsx** from the **src** folder.
2. In the **render** method, add a variable called **favouriteClassName** as an empty string.
3. Use conditional statements to:
   - Check if the current address is *NOT* **"Location not found..."**;

- Check to see if the component's **favourite** prop is **true** and set the variable for the **favouriteClassName** to **glyphicon glyphicon-star**;
- If **false**, set facouriteClassName to **glyphicon glyphicon-star-empty**.

4. Under the **h4**, add a **span** that has:
   - A **className** set to **favouriteClassName**;
   - An **onClick** event linked to a **toggleFavourite** method that will be declared in **this** class;
   - An **aria-hidden** attribute set to **true**.
5. Add a method called **toggleFavourite** to the class that has the statement:

```
this.props.onFavouriteToggle(this.props.address);
```

6. Save the **CurrentLocation.jsx** file and close it.

Create the favourites property in App

Now that the **CurrentLocation** component expects a **favourite** property, the **App** component must supply it and therefore must hold it in its **state**. The **App** component will deal with storing the favourite's details using **localStorage**, retrieving favourites information and removing the favourite from the list. It will also call a child component to display the list of favourites, supplying that with the information it needs.

1. Open **App.jsx**.
2. Before the class, add a **const** of type **array** to store the **favourites** in.
3. Next, check to see if **favourites** exists in **localStorage** and if it does, retrieve the details by parsing it into the array as JSON.
4. In the constructor, add the **favourites** array to **state**, using the value obtained before the class.

The next stage is to deal with passing the **favourite** property to the **CurrentLocation** component. To set the **favourite** property, the application needs to know if the *current address already exists* in the **favourites** array.

5. Add a function to the **App** class called **isAddressInFavourites** that:
   - Takes a **currentAddress** value as an argument;
   - Wraps the rest of the function body in an **if** statement that *only executes* the instructions below if **currentAddress** is *not* **Location not found...**
   - Sets a block level variable called **favourites** equal to the *current state of the favourites array*;
   - Sets a block level variable called **found**, inialising it to **false**;
   - Loops through the local array to check to see if the **address** supplied **matches an address in the array**, returning **found** as **true** if it does and **false** if it doesn't;
   - Returns **found**.
6. In render, add a **favourite** attribute to the **CurrentLocation** component and make it call a i**sAddressInFavourites** with an argument of this.state.currentAddress.

We are still missing the **onFavouriteToggle** property in the **CurrentLocation**. This will be a method that will add or remove the **currentAddress** from the **favourites** state (and update **localStorage**).

7. Define the handling method in the **App** class called **onFavouriteToggle**, passing in **currentAddress** as an argument. It should:
   - Check to see if the **currentAddress** is already in **favourites**:
     - Call a method called **removeFavourite** to remove it from the **favourites** if it is;
     - Call a method called **addFavourite** to add it to the **favourites** if it isn't.
8. Define the method **removeFavourite** to remove the address from favourites that takes the **address** as an argument. This method should execute the following instructions:
   - Set a block level variable to be equal to the **favourites** array;
   - Set a block level variable to store an **index** value from the array and initialise it to be **-1**;
   - Loop through the array, checking if the address supplied to the method matches an address in the array:
     - If it does, set the **index** value to be the current value of the loop counter and breaking out of the loop.
   - After the loop has completed or been broken out of, check to see if the **index** value is now *positive* and use the **array.splice()** function to remove the element at the **index**;
   - Set the class's state for **favourites** to be the value of the *spliced array*;
   - Update the **favourites** in **localStorage** by using the **JSON.stringify** function;
9. Define the method **addFavourite** to add the address to the favourites that takes the **address** as an argument. This method should perform the following actions:
   - Only execute instructions below **if address** is *not* **Location not found...**
   - Set a block level variable to be equal to the **favourites** array;
   - Use **array.push()** to add an object containing the current address and a new property called **timestamp** (set to **Date.now()**) to the **favourites** array;
   - Set the class's state for **favourites** to be the value of the method's **favourites** array;
   - Update the **favourites** in **localStorage** by using the **JSON.stringify** function.
10. In the **render** method, add an **onFavouriteToggle** attribute to the **CurrentLocation** component and set it to call the same handling method.

One final thing here, we don't want the *favourite star* to be rendered if the location was not found.

11. Open **CurrentLocation.jsx**.
12. Turn the **span** into a *ternery statement* that checks to see if **this.props.address** is **Location not found...**
    - If it is, return **false**, if it isn't, return the *span*.

13. Save the files and check that your code works.

Saving all files and running the application should allow you to see that a star symbol has now been placed next to the location name. Clicking on this star will toggle its appearance from being unfilled to filled. This action will also edit the favourites array in localStorage. You can see this by opening the Developer Tools in Chrome, clicking the Application tab and opening localStorage.

Displaying the list of favourites

The final part of this iteration to the application is to add this functionality to display the list of favourites. This will be done by adding a new component and then supplying it with the list of **favourites** and the current address. Clicking a favourite will fire the **searchForAddress()** method defined earlier (in the previous exercise).

The FavouritesList and FavouriteItem Components

1. Create a new component in the **src** folder called **FavouritesList**.
2. Create another new component file called **FavouriteItem**.

We are going to create an array of **FavouriteItem**s and populate the **FavouritesList** component with it.

FavouriteItem Component

The **FavouriteItem** will show:

- If the item is the currently displayed map;
- The address that was added;
- When the favourite was added (using a package called moment that is already installed);
- A glyphicon for menu-right.

Clicking on the address will trigger the application to use this address and set the state of **currentAddress** to it.

3. In **FavouriteItem**, **import React**.
4. Import **moment** from **'moment'**.
5. Create a functional component that takes **props**.
6. Inside the function, declare an arrow function called **handleClick** that has no arguments and calls **props.onClick** with **props.address**.
7. Declare a variable called **lgiClassName** and initialise it to **"list-group-item"**.
8. If **props.active** is **true**, append **" active-location"** to **lgiClassName**.
9. Return the following:
    - An **a** with a **className** of **lgiClassName** and an **onClick** of **handleClick** and content of:

```
this.props.address
```

    - A **span** with a **className** of **createdAt** with content as shown below:

```
{moment(this.props.timestamp).fromNow()}
```

- A **span** with a **className** of **"glyphicon glyphicon-menu-right"**.

10. Save the file.

FavouritesList Component

1. In **FavouritesList**, add the **imports** for **React** and **FavouriteItem**.
2. Create a functional component called **FavouritesList** that takes **props** and:
   - Sets a variable **favouriteLocations** and *maps* this against **props.favouriteLocations**, with the callback being an arrow function that:
     - Takes **location** as an argument;
     - Sets **active** to be the result of a boolean check of equality for **props.activeLocationAddress** and **location.address**;
     - **Returns** a **FavouriteItem** component with the following attributes:
       - **address** set to **location.address**;
       - **key** set to **location.timestamp**;
       - **timestamp** set to **location.timestamp**;
       - **active** set to **active**;
       - **onClick** set to **props.onClick**.
3. Using a *ternary* inside a wrapping **React.Fragment**, check to see if **favouriteLocations** has a **length** and **return** this mark up if it has:
     - A wrapping **div** with this **className**:

```
"list-group col-xs-12 col-md-6 col-md-offset-3"
```

     - A **span** with a **className** of **"list-group-item-active"** and content of **Saved Locations**;
     - The **favouriteLocations** array.
   - OR **null**

Make App render FavouritesList and supply props

1. Open **App.jsx**.
2. **Import** the **FavouritesList** component.
3. In **render**, before the closing **div**, add a **FavouritesList** component setting:
   - **favouriteLocations** to the **favourites** array held in **state**;
   - **activeLocationAddress** to the **currentAddress** value held in **state**;
   - **onClick** set to call **searchForAddress**.
4. Save the file and check the application is fully working.

You should now be allowed to now add and retrieve maps of favourite locations. Try closing the browser and seeing if localStorage is working correctly.

Appendix

moment documentation:

http://momentjs.com/docs/