Exercise 7 – Working with Datasets

Objective

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.

Part 1 – Project Setup

1.1. You may continue from the project used within EG06 for this exercise or you can use the **EG07_WorkingWithDatasets/starter** folder.

Part 2 – 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 class when it tries to render the component. This will be dealt with later in the exercise, but for now, the code to display an icon will be put into CurrentLocation.

2.1. In the **scripts** folder, open **CurrentLocation.jsx** file.

2.2. In `render()`, add a variable to store a class name and then 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 class name to `"glyphicon glyphicon-star"`
    - If `false`, set class to `"glyphicon glyphicon-star-empty"`

2.3. Under the `<h4>` tag, add a `<span>` that has:

- A `className` set by the variable declared above
- An `onClick` event linked to a `toggleFavourite` method that will be declared in `this` class
- An `aria-hidden` attribute set to `true`

2.4. Add a method called `toggleFavourite` to the class that has the statement:

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

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

Part 3 – Create the 'favourites' property in App

Now that the CurrentLocation class expects a favourite property, the App class must supply it and therefore must hold it in its state. The App class 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.

3.1. In the `constructor` for the class, add a global variable of type array to store the `favourites` in.

3.2. Next, check to see if `favourites` exists in `localStorage` and if it does, retrieve the details by parsing it into the array as JSON.

3.3. Add the `favourites` array to `state`.

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.

3.4. Add a favourite attribute to the `CurrentLocation` component and make it call a method (call it something like `isAddressInFavourites`) that will check to see if the value of `currentAddress` is in the `favourites` array.

3.5. Define the method `isAddressInFavourites` that:

- Takes a `currentAddress` value as an argument
- Sets a block level variable equal to the current `state` of the `favourites` array
- Loops through the array to check to see if the address supplied matches an address in the array, returning `true` if it does and `false` if it doesn't

To allow the `CurrentLocation` to be fully defined, the `onFavouriteToggle` event handling function needs to be defined in the App class. The flow of this section of code is as follows:

3.6. Add an `onFavouriteToggle` attribute to the `CurrentLocation` component and set it to call a handling method in the App class called something like `favouriteToggle`, passing in the current address

3.7. Define the handling method in the App class. It should:

- Check to see if the supplied address is already in `favourites`, calling a method to remove it from the favourites if it is and a method to add it to the favourites if it isn't

3.8. Define the method 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

3.9. Define the method to add the address to the favourites that takes the address as an argument. This method should perform the following actions:

- Set a block level variable to be equal to the `favourites` array
- Use `array.push()` to add the current address to the `favourites` array and a new property called `timestamp` (set to `Date.now()`)
- 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

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.

The final part of the **App.jsx** file to add this functionality is to prepare 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).

3.10. In `render()`, add another component called `FavouritesList`. Give it the following attributes:

- `favouriteLocations` set to the `favourites` array held in `state`
- `activeLocationAddress` set to the `currentAddress` value held in `state`
- `onClick` set to call `searchForAddress` in this class.

3.11. Add the `import` for the (yet to be created) `FavouritesList` component from a file of the same name.

The **App.jsx** file is now complete and can be saved and closed.

Part 4 – Create the 'FavouritesList' component

To display the locations that have be added to the favourites array, a new component will need to be created. This itself will be a parent component to a set of `FavouriteItem` components. This is because we want to be able to fire the `onClick` event for each of the items in the list.

4.1. In the **scripts** folder, create a new file called **FavouritesList.jsx**

4.2. Add an `import` for `React` and a yet undefined `FavouriteItem` class from a JSX file of the same name

4.3. Set the class up with an `export` and a `render()` method

4.4. Code the `render()` method using the following instructions:

- Declare a block level variable called `self` and set it to `this`
- Declare a block level variable called `favouriteLocations`
- This will be equal the `favouriteLocations` from props but it will be mapped to change the array with an anonymous function. This function will take `location` as an argument and in the function body:

- Set a block level variable called `active` to be the result of a Boolean equality evaluation of the of `self.props.activeLocationAddress` (the address given in the attribute in **App.jsx**) and `location.address` (the address from each element in the favourites array that has been passed in). This will decide the value of the active attribute in the component.
- Return a `FavouriteItem` component with the attributes:
  - `address` set to `location.address`
  - `key` set to `location.timestamp`
  - `timestamp` set to `location.timestamp`
  - `active` set to `active`
  - `onClick` set to call an click event by using `self.props.onClick.bind(this)`

- Use a selection statement to `return null` if the length of `favouriteLocations` does not exist (this should be outside of the anonymous function.

- Finally, have the `render()` method return the following HTML component:
  - A wrapping `<div>` with a `className` of:

    `"list-group col-xs-12 col-md-6 col-md-offset-3"`
  - An inner `<span>` with a `className` of

    `"list-group-item active"` and the content of '***Saved Locations***'
  - The result of the `favouriteLocations` execution.

Part 5 – Create the 'FavouriteItems' component

The final component to be created is a FavouriteItem component that will specify how the favourites will be configured for display in the list. It will use the help of a date library called ***moment*** to help display information about when the favourite was added and specify an `onClickevent` handler function to call the appropriate code when an individual item is clicked.

5.1. On the command line or terminal, use the command:

`npm install moment --save` to install the moment library

5.2. In the **scripts** folder, create a file called **FavouriteItems.jsx**

5.3. Create the `import` for `React` and `moment`

5.4. Set the class up with an `export` and `handleClick()` and `render()` methods

5.5. Make the body of the `handleClick()` method to set the `onClick` event from props to be the value of the props address:

```
this.props.onClick(this.props.address)
```

5.6. In the `render()` method:

- Set a block level variable `lgiClassName` to `"list-group-item"`
- Use a selection statement to add `" active-location"` to the `lgiClassName` if `active` from `props` is `true`
- Return a wrapping anchor `<a>` that has:
  - A `className` set to `lgiClassName`
  - Content of the `address` from `props`
  - An attribute `onClick` set to call the `handleClick()` in this class
  - An inner `<span>` with a `className` of `"createdAt"` and content of `{moment(this.props.timestamp).fromNow()}`
  - An second inner `<span>` with a `className` of:

    `"glyphicon glyphicon-menu-right"`

Saving all files should allow you to now add and retrieve maps of favourite locations. Try closing the browser and seeing if localStorage is working correctly.

Appendix

Google Maps JavaScript API documentation:

https://developers.google.com/maps/documentation/javascript/

gmaps.js documentation:

https://hpneo.github.io/gmaps/documentation.html

moment documentation:

http://momentjs.com/docs/