Exercise 6 – Using Form Elements and Events

Objective

To be able to use a form elements in a sub-component, creating a search box that updates other components.

Overview

Continuing the Your Google Map Locations application that was created earlier, there is now a requirement to have a search box within the app that will allow a user to search for maps of locations. The following specifications exist:

1. The search box should allow text to be input and a search icon to activate the search
2. When the search is run, the application will either:
    a. Update the map display with a map of the location searched for and update the current location
    b. Display "Location not found…" in the current location component

This increment to the application will require the use of the gmaps.js library and the code for this will be given as part of the exercise instructions.

Part 1 – Project Setup

1.1. You may continue from the project used within EG04 for this exercise or you can use the **EG06_UsingFormsAndEvents/starter** folder.

Part 2 – The Search Component

2.1. In the **scripts** folder create a **Search.jsx** file.

2.2. Add `imports` for `React` and `ReactDOM`.

2.3. Define the `Search` class and set it up so that it has a `state` component of `value` set to an *empty string*.

2.4. Define the `render` method so that it returns:

- A `form` with the `id` of `geocoding_form`, a `className` of `form-horizontal` and an `onSubmit` attribute the binds a function called `handleSubmit` to the component.
- A first inner `<div>` with a `className` of `form-group`.
- A nested `<div>` with a `className` of:

`col-xs-12 col-md-6 col-md-offset-3`

- A nested inner `<div>` with the `className` of `input-group`.
- A nested `input` element of `type` of `text` with:
    - an `id` of `address`
    - a `className` of `form-control`
    - a `placeholder` of `Find a location…`
    - a `value` of the current `state` `value`
    - an `onChange` event bound to a `handleClick` method
- A `span` element in the same `div` as the `input` with a `className` of `input-group-btn`
- A nested `span` with:
    - a `className` of `glyphicon glyphicon-search`
    - an `aria-hidden` attribute set to `true`

2.5. Add a `handleChange` function to the class that takes an `event` as an argument. The function should set the `state` of `value` to the value supplied in the input on the form

2.6. Add a `handleSubmit` function to the class that takes an `event` as an argument. The function should:

- Stop the event firing its usual action
- Pass the `value` of `state` to `onSearch` through `props`
- Run the `blur()` function on the `input` by using a `ReactDOM` method and an element selector to find it.

Part 3 – The App Component – Executing the Search

3.1. In the App.jsx file, add a function called `searchForAddress` that takes `address` as an argument.

3.2. A GMaps object is required to find the location and its map co-ordinates. The code for the function should be as shown on the next page:

```
searchForAddress(address) {
     let self = this;
  GMaps.geocode({
    address: address,
    callback: function(results, status) {
   if(status !== OK) {
    self.setState({
         currentAddress : Location not found…
    });
    return;
   }
   let latlng = results[0].geometry.location;
   self.setState({
    currentAddress: results[0].formatted_address,
    mapCoordinates: {
         lat: latlng.lat(),
         lng: latlng.lng()
    }
   });
     }
     });
}
```

The `GMaps.geocode()` function essentially does the leg-work of taking the textual address and searching for a corresponding latitude and longitude for it, presenting the results in a way that the map can be generated from its result.

3.3. In the `render()` method add a `Search` component under the `<h1>`. It should have an `onSearch` attribute that binds the `searchForAddress` method to the component.

3.4. Add the `import` for the `Search` component at the appropriate place in the code.

Appendix

Google Maps JavaScript API documentation:

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

gmaps.js documentation:

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