# Exercise 4 – Using Props and State

## Objective

To be able to use props and state to display information.

## Overview

You have been asked to create an application that will provide a map of a specific location, using the Google Maps API. The specification for the application is as follows:

1. There should be a heading for the page containing the text “Your Google Map Locations”.
2. There should be a map displayed from a hard-coded location that is the address of the training location you are currently in.
3. There should be some text below the map that gives the address of the location that you are currently in.

To complete this exercise, you will need to create the following components:

1. An **App** component that will act as a parent for the other components.
2. A **Map** component that will create a **GMaps** object from the *Google Maps API* and *gmaps.js* library and a container to display the map in.
3. A **CurrentLocation** component that will display the address that has been provided to display a map of.

### Part 1 – Project Setup

1.1. This project has already been set up for you, so there is no need to run through the project set-up.

1.2. In a suitable text editor, navigate to the **EG04_UsingPropsAndState/starter** folder and examine the **index.html** file. You should notice that a stylesheet has already been provided, along with a link to a Bootstrap CDN. These are used for styling the page. You will also notice that there are 2 additional script sources added. Both are needed to implement the Google Maps part of the exercise.

1.3. Unlike other projects, there will be many JSX files by the time this application is complete and therefore you will need to create a **scripts** folder in the project root. Do this now.

1.4. You will also notice that the **webpack.config.js** file has been edited to reflect that the **main.js** file will be found in the **scripts** folder.

## Part 2 – The main.js file

2.1. In the **scripts** folder create a **main.js** file.

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

2.3. Add the `ReactDOM.render` method that takes the arguments of an `App` component and the element with the `id` of `content`.

## Part 3 – The App Component I – Rendering the Component

3.1. Create the **App.jsx** file in the scripts folder and add the `import` for `React`.

3.2. Add the `class` declaration, remembering its `export`.

3.3. Add the `render` method that should simply `return` the required title text in a `<h1>` element.

3.4. Save all files and fire-up the application, checking that the output works as expected.

## Part 4 – The App Component II – Constructing the Component

4.1. Add `state` to the component that sets:

- `currentAddress` to be a **string** containing the office location you are at (e.g Oxford Street, Manchester; St Katherine's Way, London; etc and addresses can be found at https://www.qa.com/training-locations ).
- `mapCoordinates` to an object that contains:
    - `lat` to be the latitude of the address
    - `lng` to be the longitude of the address

    *These can be found at http://www.latlong.net/*

4.2. In the `render` method, add a wrapping `<div>` tag and then prepare to display the address defined by adding a `CurrentLocation` component that has an **attribute** of `address` set to its **current state**.

4.3. Add an import statement for `CurrentLocation` and then save the file.

*Note: There will be a rendering error at this point as the `CurrentLocation` component does not yet exist!*

## Part 5 – The CurrentLocation Component – Informational Sub-component

5.1. Create a **CurrentLocation.jsx** file and add the class declaration.

5.2. The class should only contain the `render` function with a wrapping `<div>` that has the following classes (used by bootstrap for styling):

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

5.3. Add a `<h4>` tag with the `id` of `save-location`. The text in the tag should use props to display the address.

5.4. Save all files and check that the output is as expected.

## Part 6 – The Map Component I – Preparing the Map Display

6.1. Create a **Map.jsx** file and add the class declaration.

6.2. Add the `render` method which returns:

- A wrapping `<div>` with the `class` of `map-overlay`
- A paragraph that contains the text `Loading…`
- A `<div>` with the `id` of `map`

6.3. Save the file.

## Part 7 – The App Component III – Adding the Map Display to App

7.1. In the `render` method, in between the `<h1>` and `CurrentLocation` component, add a `Map` component that has `lat` and `lng` attributes that obtain their **values** from **state**.

7.2. Add the `import` statement to ensure that the `Map` component can be used.

7.3. Save the file and check that the output is as expected. There should be the word '*Loading…*' below the title and above the address supplied in the state.

## Part 8 – The Map Component II – Creating the Map

8.1. Add a `componentDidMount()` method to the class that will call the method `this.componentDidUpdate()` when it is invoked.

8.2. Add the `componentDidUpdate()` method that sets a variable called `maps` to be a `new GMaps` object that has the following **properties** passed into its **constructor** as an **object**:

- `el` set to the string `#map`
- `lat` set to the value passed in by props
- `lng` set to the value passed in by props

*The componentDidMount and componentDidUpdate methods are component lifecycle methods which will be discussed in more detail in the next chapter.*

8.3. Save the file and return to the browser. Your map should have appeared.

8.4. Try different locations to ensure all works correctly.

## Appendix

**Google Maps JavaScript API documentation:**

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

**gmaps.js documentation:**

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