Unit 8: React: Lab 7 – Weather API

Assigned: 02/06/23

Due: 02/08/23

Completed: 02/08/23

**LMS:** <https://lms.grandcircus.co/mod/assign/view.php?id=22824>

**Google Doc:** <https://docs.google.com/document/d/1dC-OVF1dl_Ig8HGLeS-Z7qS9VQqaquFVI0U6PYhnnSM/preview>

**GitHub:**

**Task:** Display information from a weather API in a React application.

**Build Specifications:** create a **WeatherForecast** component and display it in the **App** component.

* Use this endpoint for the forecast in Detroit: <https://api.weather.gov/gridpoints/DTX/65,33/forecast>
* Pay careful attention to the objects and arrays in the JSON. The information you need is in the "periods" array.
* Display at least the **name**, **temperature**, **icon**, and **detailedForecast** for each period.
* It could look something like this, but you can add more information and customize the look if you like.

Graphical user interface, text, application, email

Description automatically generated

**Extended Challenge:** Use the [Open Weather Map API](https://openweathermap.org/api). You will have to sign up for an API key, but you will be able to specify the location for the forecast.

**WORK DONE IN CLASS:**

Created a new app in the Labs folder:

npx create-react-app weatherapp --template typescript

Jonathan recommends starting the process by looking at the JSON file to identify what it contains. Here we can see data that exists, but we don’t necessarily need.

Text

Description automatically generated

Whereas all the info in the Properties, drilled down by Periods is the parent / child relationship we need and is where we need to drill down to for the info our API will use.

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

We have worked with **State Hooks** (which utilize useState). Today we are going to use the **useEffect Hook**, which has 2 parameters (function, optional).

The empty array “**[ ]”** are important because without them, this **component** will re-run every time it **renders**. This helps with speed because each call to the API could add time. Additionally, calling APIs can cost money for each time you call them. Think back to our ContactForm where the **function** was re-run every time someone typed a letter because we used **onChange**. Think if you did an API call every time someone typed a single letter.

**Components** have a **lifecycle**:

* A component **mounts** when it first comes into being.

Text

Description automatically generated

In the JSON file we opened we copied the entire code. Then, in VS Code we created a model Weather.ts and in that file we used **CTRL + SHIFT + P** to open the command palette search bar. We start typing ‘**Paste JSON as Code**’ and then give it a **Top Level Name**, being the name of our file (Weather). Once we hit enter, then the code we copied from the JSON file is pasted here.

Text

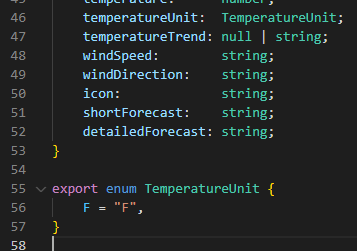
Description automatically generated

The info we need is found in the **interface** Period

Graphical user interface, text

Description automatically generated

Enum is a set list of values that don’t change. You can use it for months of the year, days of the week, etc. They are using it here to say that this cannot be changed. If you ever have to add anything to this, then add the new thing to the bottom.



Then, in the Terminal we typed in the code below



We then created a new folder in our **src** folder and called it “services”. The files in here are based on the need of the user. We can create functionality that can then be called upon in your App. By putting it into a separate file and then exporting it, then we can use it anywhere in our App without having to repeat the code (DRY – don’t repeat yourself) again.

Graphical user interface

Description automatically generated with medium confidence

We create a function for our API call and make a Promise that will return Weather, which is the base / root of our JSON.

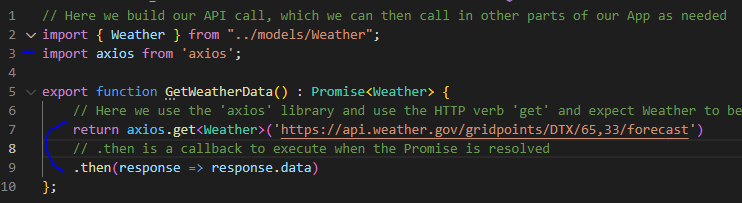
Text

Description automatically generated

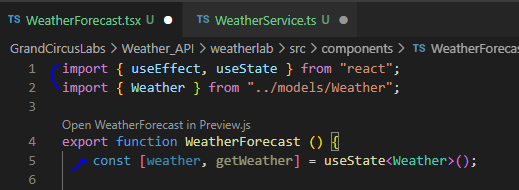
Text

Description automatically generated

We use the **axios library**, which is kinda the default library for API calls in TypeScript, whereas we used **fetch** when doing JavaScript API calls.



We then went back into WeatherForecast.tsx and added a **state hook**, using the Quick Fix to import both **useState** and the Weather interface. We didn’t have to add **[ ]** with <Weather> because there is only a single Weather as seen in the Weather.ts file.



We then make our API call by calling our GetWeatherData **function** in WeatherService.ts, then take that data and pass it along to our **state hook function** setWeather.

Text

Description automatically generated

Text

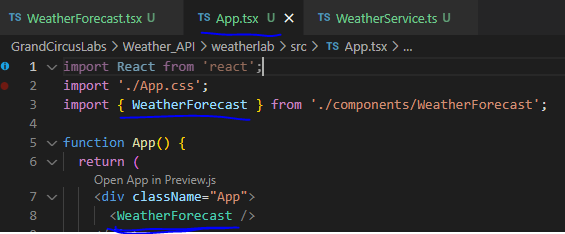
Description automatically generated

We then created an additional **useEffect** to console.log the results of our API call into the console.

Text

Description automatically generated

After adding the **component** to our App.tsx and then using **npm start** we can see this in the console.



Text

Description automatically generated with low confidence

The question mark (?) here means it is **nullable** because when it is first called it is **undefined** due to the little bit of time that it takes to get the response from the API.



But we only want certain parts of these **objects** and to **map** them as individual list items. Again, we use the ? with “periods?” because of the small amount of time between the API call and response where it wouldn’t technically exist. Because the icon references a URL, then we have to add this as an IMG tag with that as the source.

Text

Description automatically generated

Finally, we use this in our JSX. For the brief period where ‘weather’ would be undefined we don’t want to do anything. Once it is defined, then displayPeriods. You could technically put the stuff above in the JSX as well, but for some people the way we did it is preferred because it keeps the JSX cleaner.

Graphical user interface, text

Description automatically generated with medium confidence

Once we save, then our App updates with the API data. From here, we just have to add styling with CSS.

Graphical user interface, text, application, chat or text message

Description automatically generated