Assignment 4. Creating a Fashion Weather App Utilizing Streaming Open Data

Due November 17th, 2022 @ 11:59 PM Worth 25 points

In teams, you will build an app that interprets various streams of live weather data and makes colloquial suggestions about how to dress and what to bring to deal with current conditions. The user will click on a map, indicating their location, and the application will capture these coordinates. The coordinates will be passed to a weather service API, which will return various live data such as temperature, precipitation, air quality, and pollen count. Finally, the app will interpret these data and compile plain text recommendations that inform the user of what they need to be comfortable outside in such conditions.

Goals

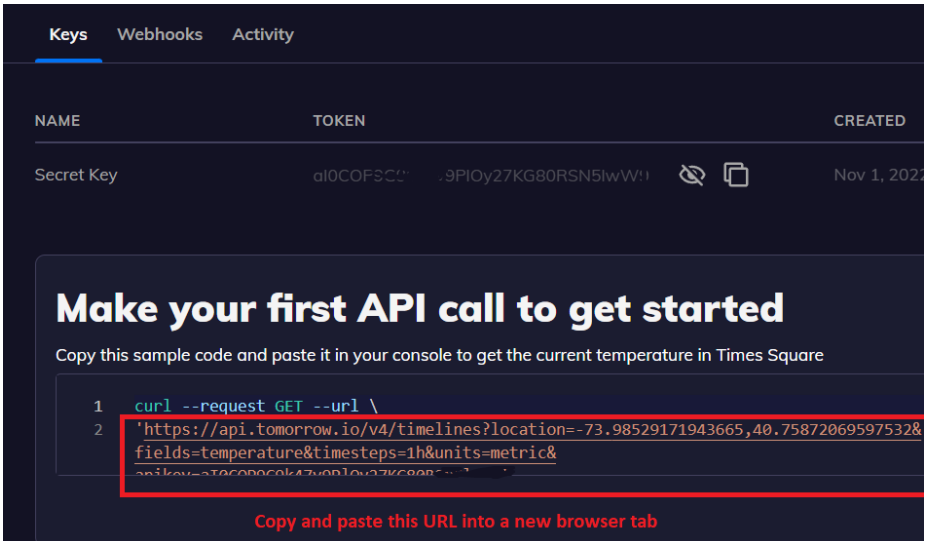
After you've completed this assignment, you should better understand how to:

- Listen for DOM events to pass information to your code (i.e., clicking the page)
- Work with an API
- Use Javascript to do API "calls" and parse the returned JSON data
- Create a response matrix that translates quantitative data into actionable qualitative observations (Basically what an AI does at a much higher level!)
- Finally, gain experience in working with streaming data.

Tasks

1. This assignment requires that you use the Tomorrow.io (https://www.tomorrow.io/) weather API. One member of your team will need to create an account (https://app.tomorrow.io/signup). Please use your McGill email and answer any questions you may be asked to suggest that you are a student, using it for educational purposes.

2. Create an API key to use for your application (https://app.tomorrow.io/development/keys). On this page you will also find a URL that is ready for you to test. Copy and paste the URL into your browser and see what you get in return. The JSON data you see is what you are eventually loading into your script.



a. Look at the structure of this URL. It has parameters that can be changed to return different information. By default, this URL has “timesteps=1h” if we change this to “timesteps=current” we get the current (live) temperature.

b. **Note:** At the time of writing this assignment, we have determined that the latitude and longitude coordinates on the default example are reversed. Change around the numbers to use this correctly. Please make sure you always have LATITUDE first followed by LONGITUDE.

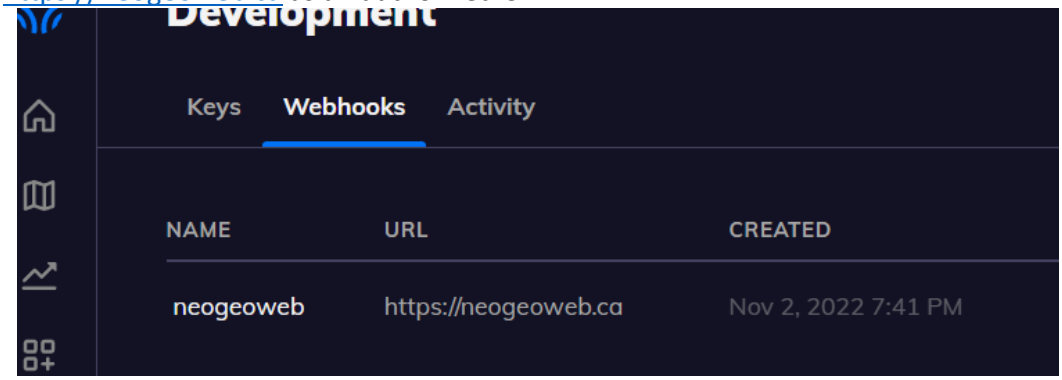
As provided from the website the default the example URL contained:

“location=-73.98529171943665,40.75872069597532,”

Whereas, it should have been:

“location=40.75872069597532,-73.98529171943665”

c. Before you leave this page, be sure to go to the ‘webhooks’ tab and add https://neogeoweb.ca as an authorized URL.



3. Conceptualize and create an app (in HTML) that translates Tomorrow.io data into practical, actionable suggestions for a user. For example: *Our app focuses on European weather conditions. The map will load with Europe in focus and the style customized to have any relevant labels and country boundaries visible. The point file we load will feature European capitals. Our data requests will focus on temperature, humidity, precipitation, cloud cover and wind gust speed. We will suggest the correct kinds of outerwear (e.g., Jacket, windbreakers, none) for Chesterton (i.e., the nearest city/town), whether or not the user needs an umbrella/galoshes, should they should put on sunscreen and whether or not they can expect it to be a humid or windy day.*

a. To maximize the utility of your application you will need to start with an embedded world map from Mapbox GL JS. Choose an area on which to focus. You are given free rein to zoom into a specific area of the world and style the map as you desire, but you should establish a clear use case and style and stick to that (see example).

Remember that weather conditions will vary by your chosen area so you will need to customize your fashion advice (i.e., no -40° C coats for Egypt right now).

Keep in mind: The expected behavior for this API is that when you submit any reasonable latitude or longitude (within the bounds of where there are instruments that measure weather) you will get some data returned to you.

b. Find a dataset of latitude and longitude coordinates for relevant cities/towns within the area of your app and add this as a point layer to Mapbox (minimum requirement is 50 points).

c. Make sure to have a title for your applications and some relevant explanatory text in the HTML body of your app.

4. Write JS code that waits for and then captures the coordinates of where a user will click on the map.
    a. Add this click to the map as a Mapbox GL JS point (drop a pin). Your code should allow for the user to click somewhere else on the map and drop a new pin, removing the old one.

    b. Snap your pin to the nearest city/town point. That will be the location you send to Tomorrow.io and what you send to the end user.

5. Use the coordinate information to construct the GET request URL that you will send to the Tomorrow.io API. The request will follow the format you saw in Step 2. **'var url = https://api.tomorrow.io/v4/timelines?location=x,y …'**
    a. You will need to ask the API for minimum 5 different kinds of weather data. This will require some trial and error and investigation to understand the kind of data you get back for each kind of weather information (i.e., temperature will return integers of degrees; pollen count will return 0-5 values that represent various conditions.)

    b. You will find the breakdown of all the parameters that you can include in your GET request here: https://docs.tomorrow.io/reference/data-layers-overview

    c. Scroll down the page and pay special attention to the section called "field descriptions". Here you will see all the possible datasets that you can ask the Tomorrow.io API for and the kind of data that you will get in return. Try replacing the "fields=temperature" section of the sample URL you saw in Step 2 (e.g., "fields=cloudCover").

    d. By the end of this step, your code should concatenate the parts of the single URL you will be sending to the Tomorrow.io API. You will add in the snapped user x,y into the location section and as well as all of the comma separated datasets you want into the "fields=temperature,cloudCover … " section.

6. To import the JSON response you get from the Tomorrow.io API into your JS code you will need to add the JQuery library to your HTML. Once you have done this, you can use the getJSON () function.
    a. In the `<head>` section of your HTML, add the following script tag:

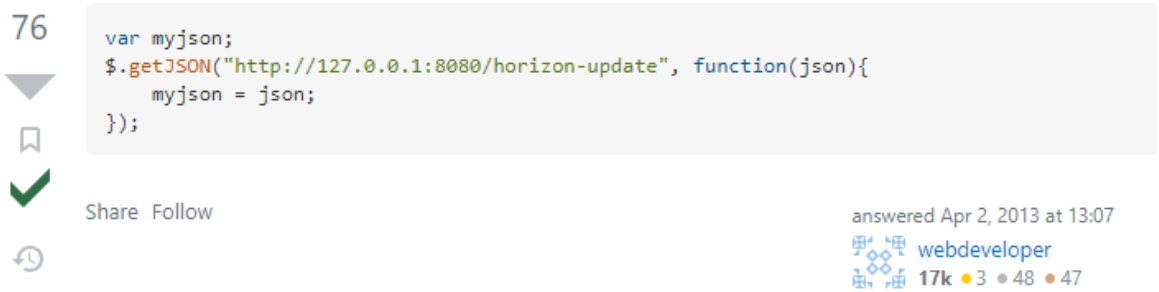
    ```
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    ```

    This adds the JQuery library to your page and allows you to use its functions in when you write your `<script>` section.

    b. Use the getJSON() function call to send your GET request URL to the tomorrow.io API. This function can be used to store the response JSON as an object variable.

i. Your code must not call the getJSON() function until you have captured the user input and added the x,y to the GET request URL. One thing must trigger the other to happen. Pay attention to what is in scope and when.

c. You will be working with "callbacks", which look straightforward but are beyond the ability of most students in class. For instance, sometimes your API call will take longer to return your JSON data than it takes to execute the rest of the code in your <script> section. In this instance your code will do its analysis on a null value and not work.
   i. To combat this, you need to put all your analysis section inside the getJSON function. This is the heretofore mentioned callback in JS.
   ii. In the example above, this would mean replacing the "`myjson = json`" line with all your analysis code or calls to functions written outside this section.
   iii. Take a look at how callbacks work on StackOverflow https://stackoverflow.com/questions/15764844/jquery-getjson-save-result-into-variable , specifically this answer:



7. The final step for your app is to return fashion advice to your user. This advice should include things like what they need to bring with them; what they need to wear; and what to expect when they go outside. This will require some critical thinking and creativity on your part. Think about how you take the quantitative data and translate it into some plain text advice. You may offer whatever practical advice you can think of. We are not expecting you to be an expert here or to do any scientific research, but you do need to use common sense (e.g., cold temperature = coat). The more effort you put into the section, the better the advice for your user.
   a. Your application's code should use conditional statements and string concatenation to create a single block of text that you display somewhere on your page or as an alert for the user.

   b. Do not forget to mention the nearest location to the location the user selected on the map.

   c. You must write at least one logic condition that to combines two or more variables (e.g., cold temp + high wind speed = expect windchill).

8. As usual, document your code. Also structure your code. When you are working with features like user clicks and URLs represented by strings, these should be stored as

global variables. As much as possible you should structure your code by creating functions.

9. Write a report.
    a. Your group report should
        i. Describe the purpose of your app.
        ii. Justify the style choices and map options you chose.
        iii. Document the structure of your app. Build a flowchart (think of it as a cartographic model) to explain the step-by-step logic of your app.
        iv. Create a response matrix that translates your "gotten" various weather observations at the specified x,y,t into qualitative actions (i.e., the fashion advice for any particular observation)

    b. For each of you in the team, reflect on the assignment (approximately one page each)
        i. Find two-three blog posts on streaming data (cite as well). What are their main points? Anything we missed in class? Coordinate with others in class so you're not relying on the same posts.
        ii. What information sources did you use to complete your portion of the assignment? For example, did you view any Youtube videos? Which ones? Did you utilize stackoverflow? How?
        iii. Make sure that all the individual reflections for your team are appended to the bottom of your report.

Notes

You can do development on your local machine but to test your getJSON() requests you will need to publish your HTML to the server and run the webpage from there, which is why we add https://neogeoweb.ca as an authorized site in Step 2c.

You will probably find Tomorrow.io rejects any getJSON() request they receive if you try to load the page saved to your local machine.

Submission

Email the code, the URL to your working app and your report describing the process (including individual reflections) to renee.sieber@mcgill.ca and Nicolas.dossantos@mail.mcgill.ca.