COMP 120: Web Programming and Engineering

Lab: The Ride-Hailing Service, Part 2 on Week 4

Revisit Part 1 on Week 3

Overview

Last week, you rendered a map with a few markers --all static content. This week will be much more involved: populate the map with dynamic content --information that changes over time. That is, you request a ride, and vehicles available will be displayed on map.

Instructions

This week, you are to implement four features onto your ride-hailing map from last week:

1. Determine your geolocation (geographic latitude and longitude) using the JavaScript `navigator.geolocation` object. Upon determining your geolocation, place a marker of where you are on the map. The icon of the marker must be different than icon used for each vehicle (it can be the default red pin Google uses).
2. Make a request to the ride-hailing API, providing your username (to be randomly generated for you, provided on Canvas), latitude, and longitude. Upon successful request, ride-hailing API will give to you a list of vehicles to mark onto map. This list of vehicles will change over time. As in part 1, each vehicle on the map shall be a marker with ths icon: .
3. Upon clicking on your marker on the map, display an information window (a.k.a., infowindow) noting the closest vehicle from where you are including the distance away in miles.
4. Render a polyline (any color) that connects "your" marker to the marker of the closest vehicle.

The Ride-Hailing API

The ride request API `https://jordan-marsh.herokuapp.com/rides` is HTTP POST only. It takes in three parameters and will return a string, JSON list, of the locations of vehicles --if the three input parameters are legitimate.

- `username` - Your username. NOTE: you will be assigned a randomly generated username for this lab, to be provided in Canvas.
- `lat` - Your latitude, a floating point number
- `lng` - Your longitude, a floating point number

If you successfully send the three inputs, you will receive a JSON string that looks like the following; a JSON string containing vehicle data. Example response:

```
[{"_id":"589bd30f8451126182dfbc62","username":"uOWuyLrd","lat":10.1,"lng":10.2,"created_at":"2019-
02-
09T02:25:19.575Z"},"_id":"589bd3258451126182dfbc63","username":"dwR3TbOH","lat":20.3,"lng":20.4,"c
02-09T02:25:41.166Z"}]
```

If you do not send the three inputs correctly, you will receive the following JSON: `{"error":"Whoops, something is wrong with your data!"}`

Important: Cross-Origin Resource Sharing (CORS) is enabled for this API!

Executing HTTP POST via `XMLHttpRequest`

In order to execute HTTP POST using the JavaScript `XMLHttpRequest` object, you need to add a parameter to the HTTP request header: `xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");` where `xhr` is an instance of `XMLHttpRequest`. See https://stackoverflow.com/questions/9713058/sending-post-data-with-a-xmlhttprequest for more information.

Also, the parameter to send via `xhr.send()` must the in the following format: `"username=YOUR_USERNAME&lat=YOUR_LATITUDE&lng=YOUR_LONGITUDE"` (with the double quotes). This is standard URI convention. Remember replace YOUR_LATITUDE and YOUR_LONGITUDE using string concatenation!

Performance Requirements

1. You can only call the ride-hailing API once. We're not working with a real-time schedule here.
2. You can only render the map once. That is, one single call of **`new google.maps.Map()`**.

Calulating the Distance Between Two Geopoints

There are two ways to calculate the distance between two geopoints, choose one:

1. Use the classic Haversine Formula. The formula is provided and analyzed at http://www.movable-type.co.uk/scripts/latlong.html. Also, you are encouraged to use the JavaScript implementation of the Haversine Formula on Stack Overflow but please do cite it in your **`README.md`** or in your code: http://stackoverflow.com/questions/14560999/using-the-haversine-formula-in-javascript.
2. **(Much referred and much easier)** Use **`google.maps.geometry.spherical.computeDistanceBetween(latLngA, latLngB);`**. See documentation at https://developers.google.com/maps/documentation/javascript/reference#spherical

Remember, the distance must be in **miles**, not meters.

Testing Your Work

For this part of the lab, you can open the **`index.html`** page on a web browser. However, it is bad practice as you will see next week. Another way to serve your page locally is to run Python's simple HTTP server in the folder of your work. That is:

```
% cd comp120-XXXXX/notuber; # where XXXXX is the name of your private GitHub repository
% python3 -m http.server
```

By default, a simple web server will open up on port 8000. Go to **`http://localhost:8000/`** on your favorite web browser to test your work.

Do not serve your work via the **`gh-pages`** branch of your private GitHub repository.

The `README` File

Each lab shall include a **`README`** file that describes the work. This description must:

1. Identify what aspects of the work have been correctly implemented and what have not.
2. Identify anyone with whom you have collaborated or discussed the lab.
3. Say approximately how many hours you have spent completing the lab.
4. Be written in either text format (**`README.txt`**) or in Markdown (**`README.md`**). Markdown is preferred. No other formats will be accepted. Please use all capital letters for **`README`**

This **`README.md`** file must be directly in the folder of the lab.

Going Beyond

- Upon determining your location, display the restaurants, bars, and coffee shops within one mile around you. Consider using the Google Places API.
- Upon clicking on a marker for a vehicle, show popup infowindow noting how far away the vehicle is, in miles, from you.

Submitting Part 2

Same instructions as for submitting part 1: push all your changes to the private repository in GitHub that I created for you in a folder named **`notuber`** under the **`master`** branch. Say that your private repository in GitHub is named **`comp120-mchow`**, make sure all the files are pushed to **`comp120-mchow/notuber`**.

Assessment

This lab is worth 10 points:

- (1 point) **`README`**
- (1 point) Determine and mark your location on the map
- (2 points) Make a successful request to the ride-hailing API, send your username, latitude, and longitude
- (3 points) Mark all the vehicles returned by the ride-hailing API on the map using the icon image
- (2 points) Note the closest vehicle from where you are (e.g., upon clicking on marker of where you are)
- (1 point) Polyline connecting your marker to the closest vehicle

- (-3 points) Errors exist in JavaScript console. That is, errors that are not Google Maps API related. Warnings are acceptable.
- (-3 points) You called either the ride-hailing API or Google Maps JavaScript API more than once.
- (BONUS 1 point) Accomplish at least one of the "Going Beyond" items