Project Rubric

Your project will be evaluated based on the project rubric file attached. Please review for detailed project requirements.

Development Strategy

It's very important that you plan your project before you start writing any code. Break your project down into *small* pieces of work and plan out your approach to each one. It's much easier to debug and fix an issue if you've only made a small change. It becomes much harder if you wait longer to test your code. You don't build a house all at once, but brick-by-brick.
Feel free to implement your own design workflow, but if you get stuck -- here are some quick tips to get started. The following verbiage reflects usage of the Google Maps API, but note that you may use another Map API system aside from Google Maps (e.g., Leaflet) for this project (please provide an explanation of which mapping system you used, and how you are using in in your project with comments or in your README).

1. Obtain a **Google Maps API key**.
- Be sure to include it as the value of the `key` parameter when loading the Google Maps API.
- You may have some initial concerns with placing your API key directly within your JavaScript source files, but rest assured this is perfectly safe. All client-side code must be downloaded by the client; therefore, the client must download this API key (it is not intended to be secret). Google has security measures in place to ensure your key is not abused. Note that it is *not technically possible* to make anything secret on the client-side.
2. Add a full-screen map to your page using the **Google Maps API**.
- For the sake of efficiency, the map API should be called only once
- With your Google Maps API key and a `props` object, how can you generate a valid URL to a static Google Maps image? Would writing a function work well for this? What would that function return?
3. Write code required to display map markers identifying at least 5 locations that you are interested in within this neighborhood.
- Your app should display those locations by default when the page is loaded.
4. Implement a list view of the set of locations you have defined.
5. Provide a filter option (e.g., a text field or dropdown menu) that uses an input field to filter both:
- The list view
- The map markers displayed by default on load. The list view and the markers should update accordingly in real time.
- **Note**: Providing a search function through a third-party API is not enough to meet specifications.
6. Add functionality using third-party APIs to provide information when a map marker or list view entry is clicked (e.g., Yelp reviews, Wikipedia, Flickr images, etc).
- StreetView and Places do not count as an additional 3rd party API because they are libraries included in the Google Maps API
- If you need a refresher on how to make Ajax requests to third-party servers, check out **Intro to Ajax**
- Please provide attribution to the data sources and/or APIs you use (e.g., if you are using Foursquare, indicate somewhere in your interface and in your README that you used Foursquare's API)

Overall, the application's interface should be intuitive to use. Here are a few examples of how to make the experience straightforward for your user:

- The input text area to filter locations should be easy to locate
- It should be simple to understand which set of locations is being filtered
- Selecting a location via list item or map marker should cause the map marker to bounce, or in some other way *animate*. This indicates that the location has been selected, and an associated info window should open above the map marker with additional information

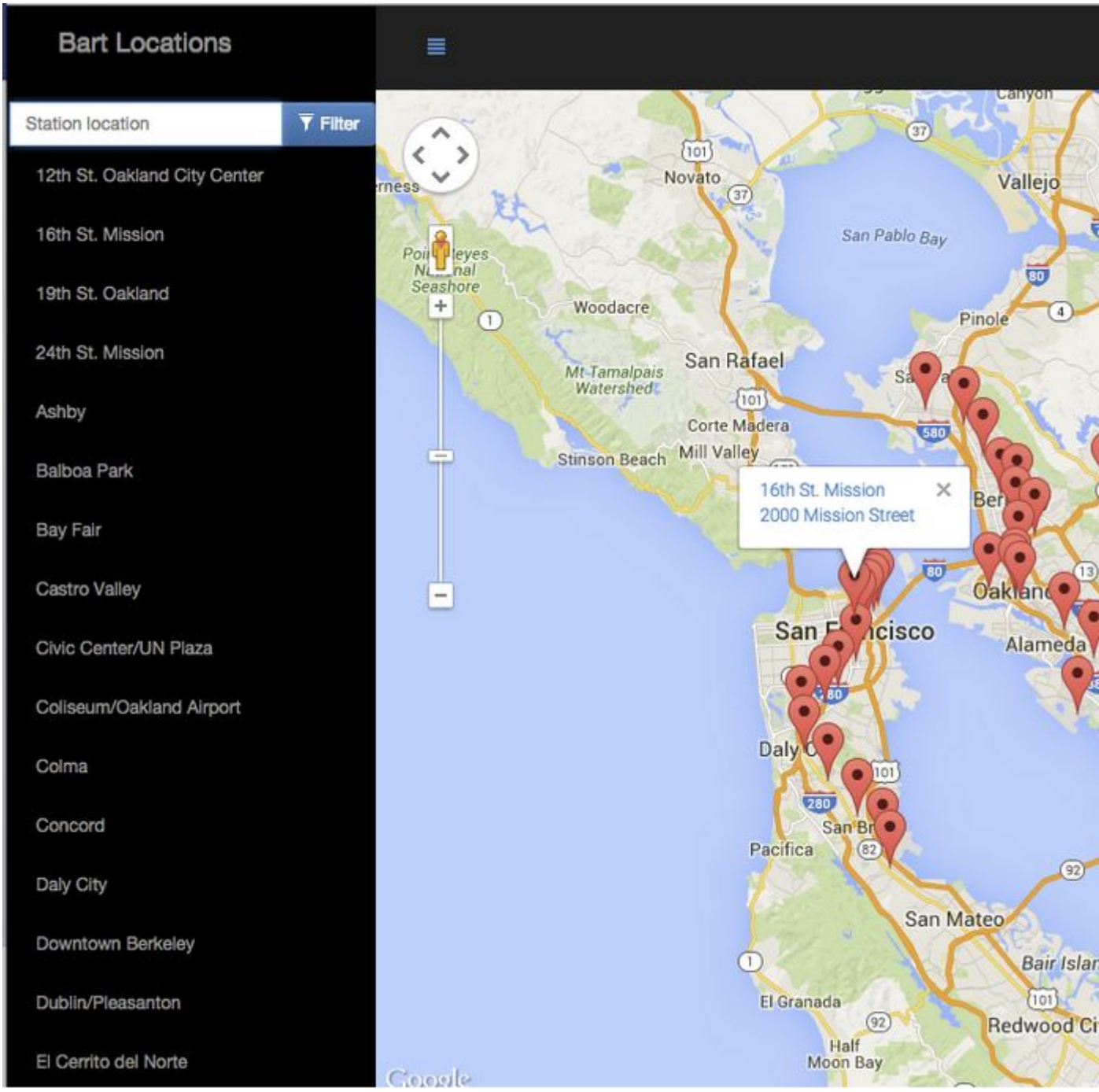
Asynchronicity and Error Handling

All data API's used in the application should load asynchronously, and errors should be handled gracefully. In case of error (e.g., a situation in which a third party API does not return the expected result), we expect your webpage to do one of the following:

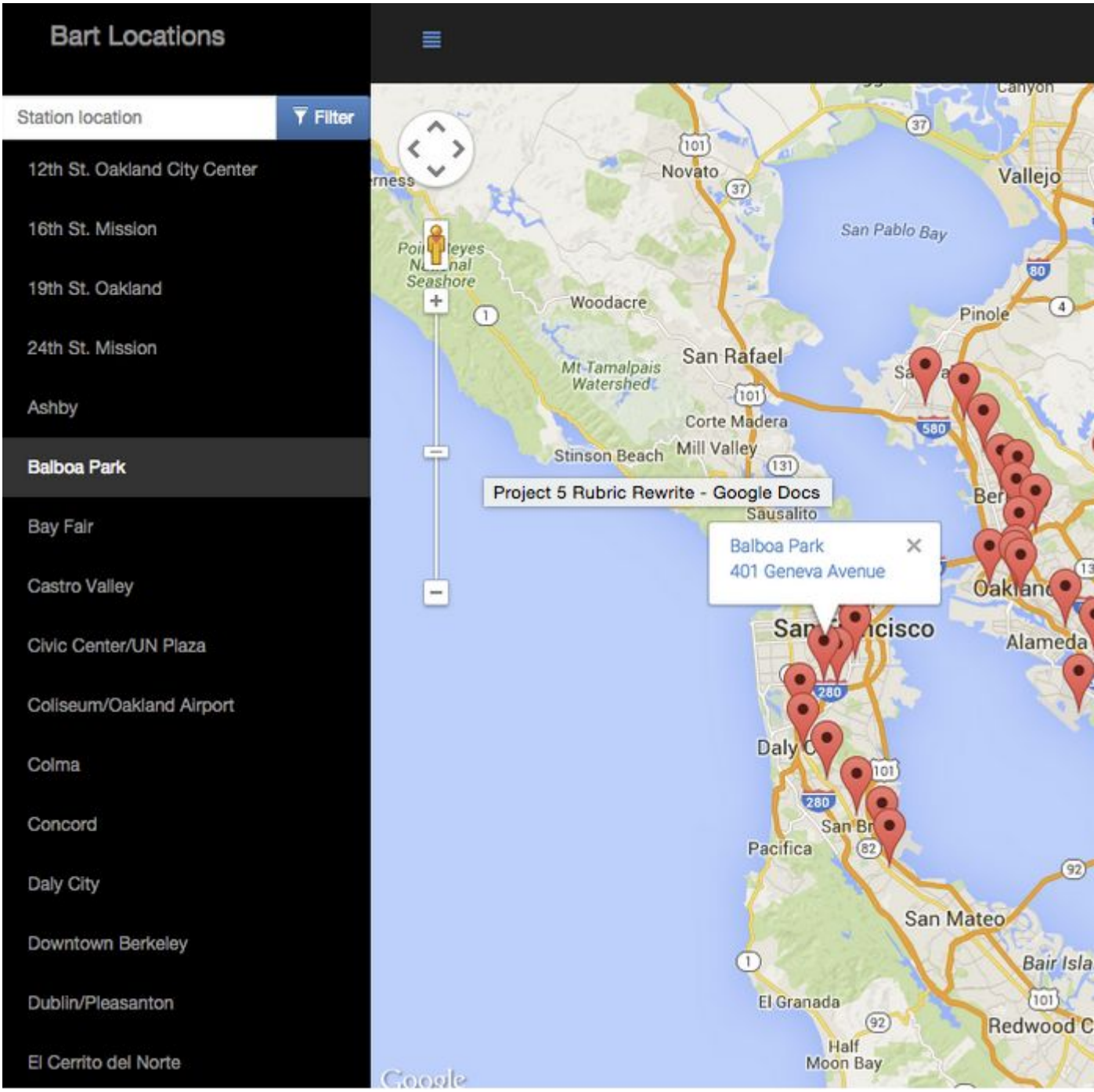
- A message is displayed that notifies the user that the data cannot be loaded
- There are no negative repercussions to the UI
  Note: You should handle errors if the browser has trouble initially reaching the third-party site as well. For example, consider a user using your Neighborhood Map, but the user's firewall prevents him/her from accessing the Instagram servers.

User Interface Examples

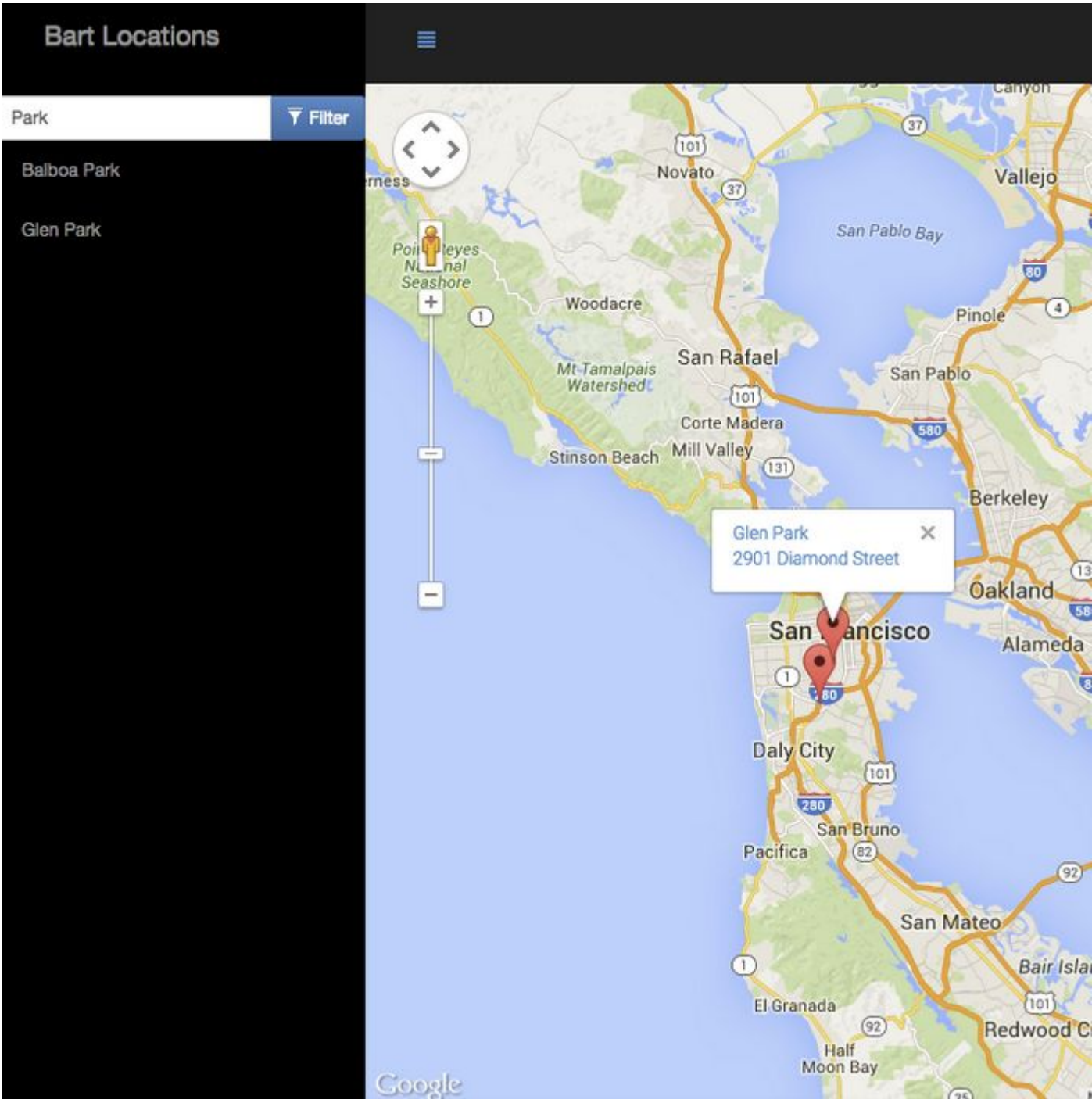
Here are some examples of the Neighborhood Map (React) project showing BART locations in the San Francisco Bay Area.



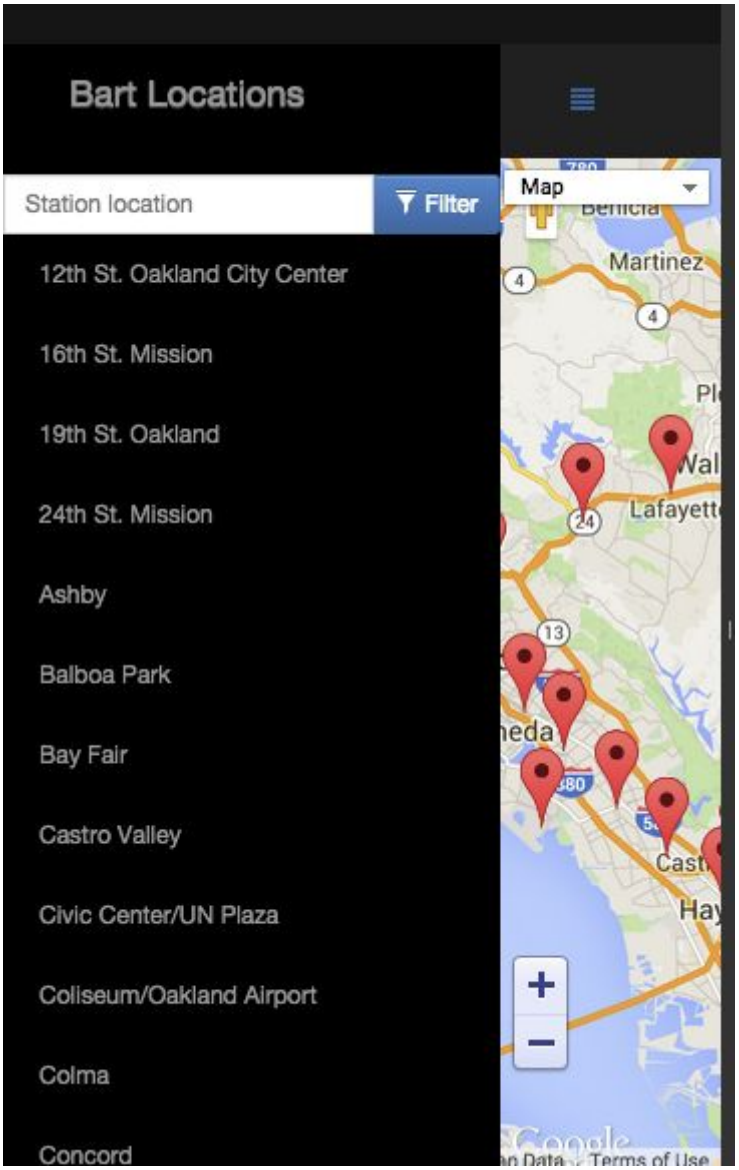
Clicking a marker on the map should open more information about that location.



Clicking a name in the list view should display the information for the associated marker.



The list of locations should be filterable with a text input or dropdown menu. Filtering the list also filters the markers on the map.



The web app should be responsive (i.e., to mobile devices). Note that a hamburger menu icon used to hide the list on small screens (this is just one possible mobile implementation).

Additional Resources

None of these are required, but they can help guide you along the way. Feel free to also check out this **Knowledge post** featuring student-curated resources for the Neighborhood Map (React) project as well.

- **Using Google Maps in a React component** (StackOverflow)
- **How to Write a Google Maps React Component**
- `react-google-maps` **package** (provides a set of React components wrapping underlying Google Maps API instances)

API-related resources:

- **Foursquare API**
- **Google Maps Street View Service**
- **ProgrammableWeb's API Directory**
- **MediaWikiAPI for Wikipedia**