

This repository

Search


Pull requests

Issues

Marketplace

Gist

+ ▾



▾

udacity / fend-office-hours

Watch ▾

50

Star

105

Fork

104

Code

Issues 0

Pull requests 1

Projects 0

Wiki

Insights ▾

Branch: master ▾




fend-office-hours / Javascript Design Patterns / P5 Project Overview /

Create new file

Upload files

Find file

History

 JohnUdacity	no longer recommend twitter	Latest commit 348dd68 on Jun 25, 2015
..		
 images	Organize JSDP folder, added JSTest folder, wrote articles for 2/19 OH	3 years ago
 README.md	no longer recommend twitter	2 years ago

 README.md

Office Hours Link

Link to the Office Hours: [P5 and P6](#)
[Link to P6 Overview](#)

Project5 Neighborhood Map Example

Let's talk a little bit about the basic functionality of our map! First and foremost we need to make sure we include the 3 main components A Search bar, a list view, and a map. Notice our map has markers and the list view contains the names of each of the locations on our map.

Search

Key Locations

Little Manuels

Hazels Drive-in

Tao San Jin

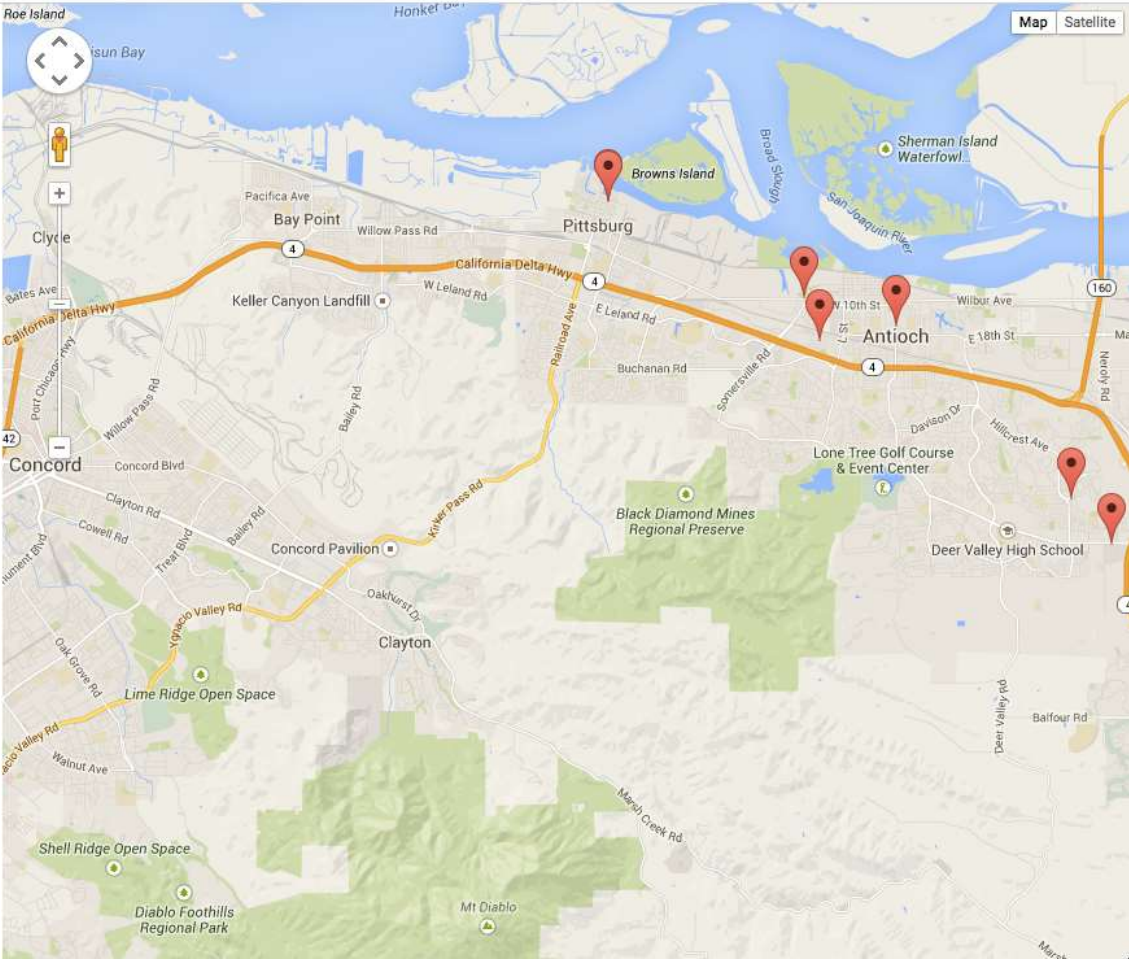
Johnny Garlics

In-N-Out Burger

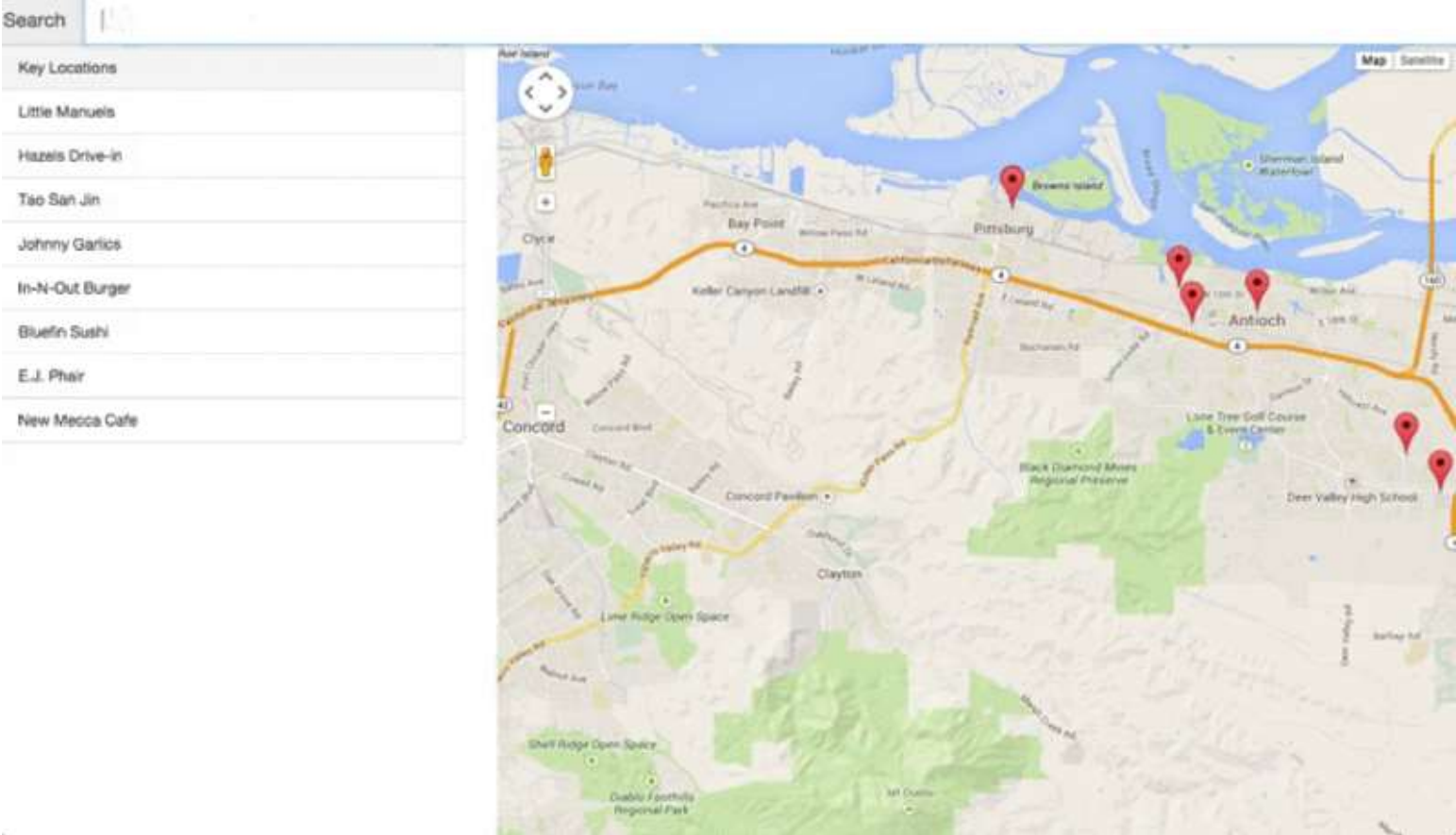
Bluefin Sushi

E.J. Phair

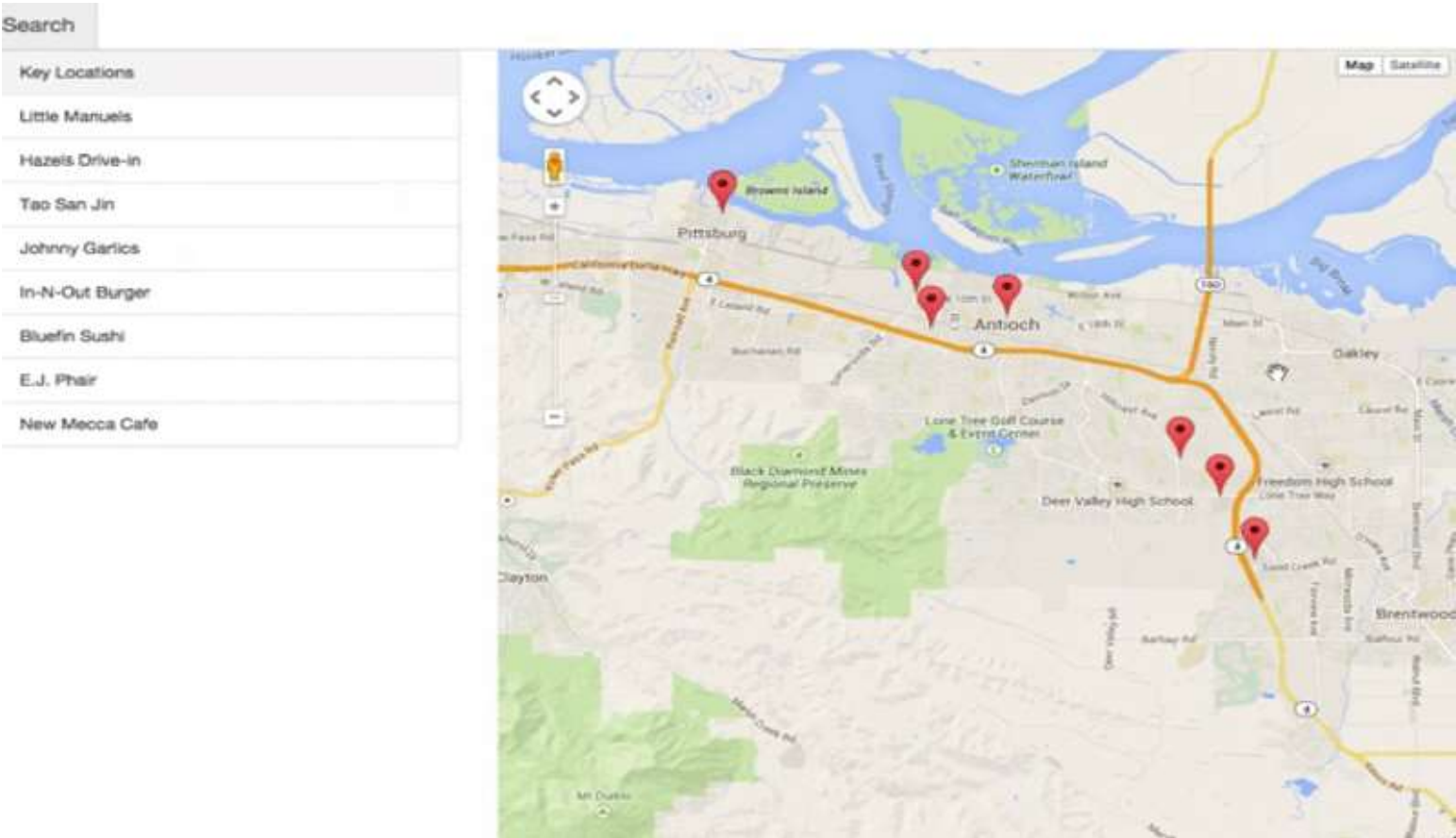
New Mecca Cafe



Now for basic functionality we need to make sure that our search filters BOTH our list view and our map markers. We are only looking for a basic search functionality, so it only needs to be able to search by name. Feel free though to modify the filter to search for location type or other information.



Along with our list and markers filtering based on our search bar, they should also both be clickable. When we click a list item it should move towards the marker and open an info window. Clicking on a marker should also open an info window. The content of the info window is up to you but it would be a good chance to include an API! Notice as well that our markers animate to draw our users attention to that marker.



The Rubric

Interface Design

All application components render on-screen in a responsive manner, usable across modern desktop, tablet, and phone browsers

By this point in the Nanodegree you should be familiar with what it means for something to be responsive. The best way to begin making the interface responsive is that you design for mobile first. Build it small then make it big! If you have time take a look at the [Responsive Web Design Course](#) that we released that will teach you a lot of great ways to make things responsive.

App Functionality

All application components function appropriately without error.

This is what is shown above, make sure that the search bar, list view, and map. If you do add additional components these also **must** function as well. Adding half-built additional functionality will lead to not meeting specifications.

App Architecture

Code is properly separated based upon Knockout's best practices.

Knockout is a bit of a shift in thinking. If you don't know where to start go through [Knockouts Documentation](#), which reads like a series of articles. Follow along and build some of the basic examples from Knockout and try to apply them to your project.

Really think about the Model-View-ViewModel paradigm. Remember that Google Maps API handles the View and the Model you are mostly taking of the ViewModel component. You never really touch the HTML code that takes care of the View.

Finally be sure to think of infowindow's content as either a view or a model. The InfoWindow itself is the ViewModel. We need to separate concerns and make sure we make it easy for us to go back and set the content for the InfoWindow

Asynchronous Data Usage

Application utilizes Google's Map API and at least one additional third-party "data API". All data requests are retrieved in an asynchronous manner. In the event of a failed data retrieval errors are handled gracefully.

So we require that you utilize Google's Map API. The main thing we want to point out here that you must make sure that if the data retrieval fails you indicate to the user that such a problem has ocured. The user should not be left guessing why something is not working. To test the functionality of failed data requests you can simply disconnect from the internet.

API's We Recommend

[Yelp's API](#) [Instagram's API](#) [Foursquare's API](#) [Wikipedia's API](#)

Questions

Can you elaborate on the search/filter functionality? should I be able to search by name of the place or something else as well?

To meet specifications the only functionality we are looking for that it filters by name. You can add additional functionality to the filter if you want for example utilizing search terms or being able to filter by star rating.

The code I write to set up data for my Model such as load it from foursquare using getJSON(...) should it be inside my Model itself or in the ViewModel?

Depends on where you think that data should go. If you think it makes sense as part of your Model put it there. If you think it makes more sense as part of the ViewModel put it their. Just stay consistent and make sure that you will be able to come back to the project in the future and easily know where things are located and whats going on.

Any recommendations for APIs that require OAuth? Utilizing client-side JavaScript appears to be frowned upon for these instances.

Check out <http://oauth.net/code/>. For a list OAuth library. I utilized Yelp API that just uses OAuth. Most others tend to be pretty simple.

I've run into several APIs that do not support CORS or JSONP. I'm guessing these are expected to be accessed by back-end systems, and our client-side scripts would access the data there. Do you have any recommendations for utilizing these at all?

Besides trying to set up a back-end (i.e. using something like NodeJS). With JSONP it's a bit of a hack that goes around CORS. JSONP gives us the ability to get the response as a function. You send a JSONP request and you get the data back and that data gets called on an anonymous function. Really JSONP is a bit of a hack to get the raw data from a server. Assuming you're using JQuery with AJAX to make your JSONP request.

When I am hiding/displaying markers with marker.setMap(null)... is this following knockout best practices because even though I am not touching the DOM, I am also not using knockout observables etc..

Not using an observable isn't going against knockout's best practices. Since the marker object is a part of Google Maps ViewModel and not necesarilly the one you are building you don't necessarily need to use knockout for that.

What are your favorite APIs other than Google maps?

We really like the yelp API, Last.fm API, Twitter's API, Instagrams API, Foursquares API. For other maps we haven't really tried any other mapping API's than googles API's. If there is a website you go to regularly look out for a developer or API link as most sites these days have these!

Does this mean using open street maps api is ok as well?

Not for this project we ask that you please use only the google maps API, but in your own project feel free to!