PROJECT

Neighborhood Map

A part of the Front-End Web Developer Nanodegree Program

PROJECT REVIEW | CODE REVIEW 2 | NOTES

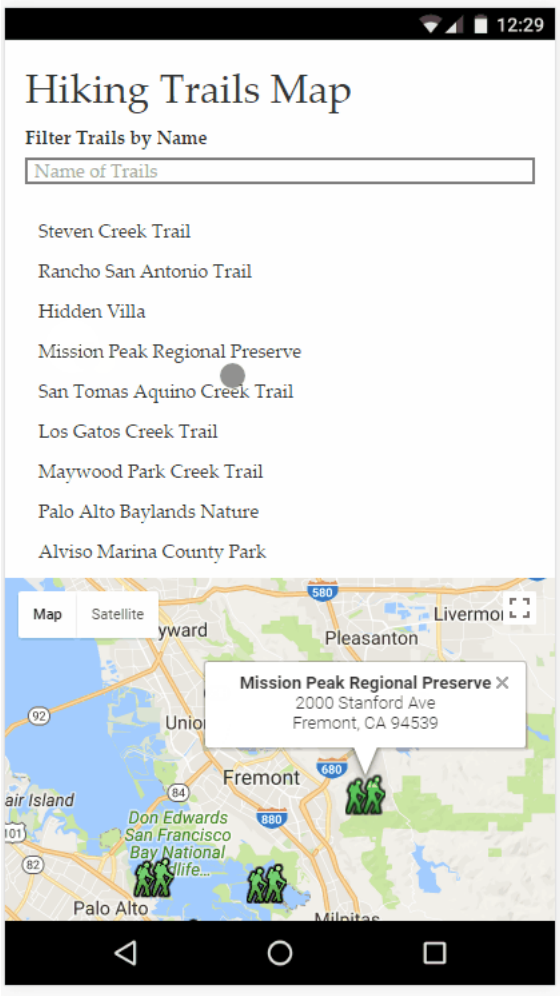
Meets Specifications

SHARE YOUR ACCOMPLISHMENT

Interface Design

All application components render on-screen in a responsive manner.

All application components are usable across modern desktop, tablet, and phone browsers.



App Functionality

Includes a text input field or dropdown menu that filters the map markers and list items to locations matching the text input or selection. Filter function runs error-free.

A list-view of location names is provided which displays all locations by default, and displays the filtered subset of locations when a filter is applied.
Clicking a location on the list displays unique information about the location, and animates its associated map marker (e.g. bouncing, color change.)
List functionality is responsive and runs error free.

Map displays all location markers by default, and displays the filtered subset of location markers when a filter is applied.

Clicking a marker displays unique information about a location in either an `infoWindow` or `DOM` element.

Markers should animate when clicked (e.g. bouncing, color change.)

Any additional custom functionality provided in the app functions error-free.

App Architecture

Code is properly separated based upon `Knockout` best practices (follow an `MVVM` pattern, avoid updating the `DOM` manually with `jQuery` or `JS`, use `observables` rather than forcing refreshes manually, etc). `Knockout` should not be used to handle the `Google Map API`.

There are at least 5 locations in the model. These may be hard-coded or retrieved from a data API.

Asynchronous Data Usage

Application utilizes the `Google Maps API` and at least one non-Google third-party `API`.
Refer to this documentation

All data requests are retrieved in an asynchronous manner.

Data requests that fail are handled gracefully using common fallback techniques (i.e. `AJAX` error or fail methods). 'Gracefully' means the user isn't left wondering why a component isn't working. If an `API` doesn't load there should be some visible indication on the page (an alert box is ok) that it didn't load. *Note*: You do not need to handle cases where the user goes offline.

Location Details Functionality

Functionality providing additional data about a location is provided and sourced from a 3rd party API. Information can be provided either in the marker's `infoWindow`, or in an `HTML` element in the `DOM` (a sidebar, the list view, etc.)

Provide attribution for the source of additional data. For example, if using Foursquare, indicate somewhere in your UI and in your README that you are using Foursquare data.

Application runs without errors.

Functionality is presented in a usable and responsive manner.

Documentation

A `README` file is included detailing all steps required to successfully run the application.

Comments are present and effectively explain longer code procedures.

Code is formatted with consistent, logical, and easy-to-read formatting as described in the Udacity JavaScript Style Guide.

If build tools (such as Gulp or Grunt) are used, both source and production code are submitted in the same repository in separate directories. These directories are usually named `src` and `dist` respectively.

DOWNLOAD PROJECT

2 CODE REVIEW COMMENTS ›

Have a question about your review? Email us at review-support@udacity.com and include the link to this review.

RETURN TO PATH

Student FAQ