UDACITY

EN-US ZH-CN

Neighborhood Map Project

Criteria	Meets Specifications
Interface Design	
Responsiveness	All application components render on-screen in a responsive manner.
Usability	All application components are usable across modern desktop, tablet, and phone browsers.
App Functionality	
Filter Locations via Text Input	Includes a text input field that filters the map markers and list items to locations matching the text input. Filter function runs error-free.
List View	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 and Markers	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.)
App Architecture	

Proper Use of `Knockout`	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 hard-coded in the model.
Asynchronous Data Usage	
Asynchronous API Requests	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.
Error Handling	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	
Additional Location Data	Functionality providing additional data about a location is provided. 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.
Error Free	Application runs errors free.
Usability	Functionality is presented in a usable and responsive manner.
Documentation	
README	A `README` file is included detailing all steps required to successfully run the application.
Comments	Comments are present and effectively explain longer code procedures.
Code Quality	Code is formatted with consistent, logical, and easy-to-read formatting as described in the Udacity JavaScript Style Guide.

Suggestions to Make Your Project Stand Out!

- Add unique functionality beyond the minimum requirements (i.e. the ability to “favorite” a location, etc.).
- Incorporate a build process allowing for production quality, minified code, to be delivered to the client.
- Data persists when the app is closed and reopened, either through localStorage or an external database (e.g. Firebase).
- Include additional third-party data sources beyond the minimum required.
- Style different markers in different (and functionally-useful) ways, depending on the data set.
- Implement additional optimizations that improve the performance and user experience of the filter functionality (keyboard shortcuts, autocomplete functionality, filtering of multiple fields, etc).
- Integrate all application components into a cohesive and enjoyable user experience.