UDACITY Logout

‹ Return to "Front-End Web Developer Nanodegree" in the classroom

Neighborhood Map (React)

REVIEW | CODE REVIEW 7 | HISTORY

Meets Specifications

🎉🎊Congratulations🎊🎉

Awesome job finishing this project💪
You learned a lot as I can see and did a really good job on this project🎉

Have a look at the comments I placed in your code and project review.
There are some additional resources to make this project even greater💪

Now it's time to celebrate🎊

Good 🍀 and stay Udacious👍

Interface Design

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

Awesome

nice work adding the search box👍

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

Great Job

The components are all usable on small, middle and large screens.
I only suggest to make your side menu off the screen on smaller devices and show an open/close button to hide/show the sidebar.
This will increase the user experience.
If your interested in this have a read through this post

Application 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.**

Perfect

Nice work handling the search field updating the markers and the list💪

✓ 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 somewhere on the page (modal, separate div, inside an infoWindow).

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

Awesome

The map displays a list by default.
When a filter is added the markers update.💪
When the marker or list is clicked an info modal pops up showing additional information about the location👍

Asynchronous Data Usage

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

All data requests are retrieved in an asynchronous manner using either the Fetch API or XMLHttpRequest.

Great Job

Good job adding `script.async` in your `App.js` file.
This will prevent that other scripts stop loading when something goes wrong with the Google API💪

✓ 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 that it didn't load.

Documentation

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

Nice Job

Your `README.md` helped me set up this app easily.👍

suggestion
See previous reviewers comment about writing documentation about the production mode.
If you want to learn more about how to write **Awesome** readmes have a look at this free Udacity Course

✓ Comments are present and effectively explain longer code procedures.

Comments are to help you and other developers understand your code better.
If you want to learn more about writing good comments have a look at the Udacity Style Guide

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, a modal, 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 console errors.

Awesome

I really love how clean the console is 💪

✓ Functionality is presented in a usable and responsive manner.

Accessibility

✓ Focus is appropriately managed allowing users to noticeably tab through each of the important elements of the page. Modal or interstitial windows appropriately lock focus.

✓ Elements on the page use the appropriate semantic elements. For those elements in which a semantic element is not available, appropriate `ARIA roles` are defined.

Awesome

Great job adding the roles 💪

suggestion
Add also a role to the map

✓ All content-related images include appropriate alternate text that clearly describes the content of the image.

Offline Use

✓ When available in the browser, the site uses a service worker to cache responses to requests for site assets. Visited pages are rendered when there is no network access.

Nice job

The service worker is activated in production mode 💪

suggestion
See other reviewer's comment about adding this to your readme.
Also in your package.json you added a `"homepage"` key this you should remove.
The build process assumes that your project is hosted there and will append that URL to all your links.

Application Architecture

✓ React code follows a reasonable component structure.

State control is managed appropriately: event handlers are passed as props to child components, and state is managed by parent component functions when appropriate.

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

There are no errors. There are no warnings that resulted from not following the best practices listed in the documentation, such as using `key` for list items. All code is functional and formatted properly.

DOWNLOAD PROJECT

7 CODE REVIEW COMMENTS ›

RETURN TO PATH