Civil Discourse Map Guide

The civil discourse map is react-leaflet map that uses the world.json file in the 'src/geoJson/' directory to create an interactive map. The JavaScript files used for creating and displaying the map are: ColorMap.js, MapWrapper.js, Legend.js, and App.js. In this guide, there will be in depth descriptions of each file used to create and display the map for this application.

ColorMap.js

Map component which adds appropriate colors, popup information, and hover options to each country displayed on the map.

Parameter – *allCountries*

allCountries is a list of all countries provided by the getCountry() function in 'src/actions/index.js' called on line 12 in 'src/components/MapWrapper.js' to get country information we wish to display from AWS via an Axios request.

Functions:

getColor(rank) – gets active color scheme and the country's associated color based on the country's discourse rank and color scheme selected.

getCountryColor(name) – returns the color for that country based on the *rank* of the provided name.

getMatchingCountries(name) – returns a country if the name provided matches the country's name.

getRank(name) – returns the discourse ranking of the country name provided to display in popup.

getPopulation(name) – returns the population of the country name provided to display in popup.

getInternetPercent(name) – returns the internet accessibility percentage of the country name provided to display in popup.

getCensorshipLevel(name) – returns censorship level of the country name provided to display in popup.

geoJsonStyle(country) – sets the fill color of the country to the color assigned from *getCountryColor(name)*

Return –

The return for ColorMap is the map and its components wrapped in a *MapContainer* element which is responsible for creating the leaflet map instance and providing the map to its child elements.

<*Pane*> - helps with ordering of layers for the map. Since our pane element has a *TileLayer* element, it creates a layer for the *TileLayer.*

<*TileLayer*> - provides the map with the country names loaded from the url provided

<*Legend /*> - is the legend component imported into ColorMap.js

<*GeoJSON*> - allows GeoJson data from our world.json to be parsed and displayed on our map. In our GeoJSON element, the world.json file is parsed to display basic country information in a popup.

onEachFeature() – gives each *feature*(a.k.a. country) event handlers in this application. layer.bindpopup will bind a popup element to the GeoJson layer in that features coordinates, which the user interacts with on the map. This is also where most of the functions are used to get the data needed to display in the popup.

ColorMap.js exports the ColorMap with its functions to be wrapped in MapWrapper.js with the header for the app.

MapWrapper.js

The MapWrapper file wraps the Header and ColorMap components to be displayed on the main page. The Header and ColorMap components are wrapped in a div for ease of display. The <*Suspense*> element has a ternary operator *world.countries.length >= 171 ?* which returns the ColorMap if true, or a circular progress indicator if all countries have not been fetched from AWS. The Suspense element is used to determine if all async data has been fetched, if not it will notify React that it is still waiting.

useEffect(() => {

getCountry() - tells the application to fetch the country data from AWS after rendering

}, [])

Legend.js

The legend component in the lower right-hand corner of the map that gives the user information about how the color relates to the country's civil discourse ranking. The legend also has a menu which allows the user to change to the color scheme they desire.

App.js

The App.js file is the whole application wrapped in a <*React.Fragment*> element, in this case just "<>", and a <*BrowserRouter*> to specifiy the different pages of the application.

The first <*route*> element holds the MapWrapper component and is the default page when the url loads.

Moving Forward

Map libraries were researched to determine if there was a simpler option to create an interactive map for this application. Upon conclusion, React-leaflet is the strongest map library with many examples and great documentation about how each component works. However, there are a couple options that should be considered going forward with this application.

Map Libraries

Current: React-leaflet

Potential:

- **Pigeon-maps**
  - Small library
  - Easy to get up to speed
  - Can be used with React-leaflet
  - No external dependencies
- **React-mapbox-gl**
  - Fully customizable map making with mapbox
  - Large library - comparable to leaflet

General Map Changes

With the map being the most prominent part of this application, it is important that user feedback is considered to make necessary changes to the map. Currently, the popup location is in the process of being moved to the lower left-hand corner to allow increased visibility of the countries surrounding the country selected. Another option to be considered for displaying basic country information is the use of React-leaflet-sidebarv2 which is a third party library that will allow a sidebar that would be filled with the popup information on a country click. This is an example of how that could look: