Evan DePosit
6/5/19
Front End Web Technologies
Final Project

# Corona Virus Dashboard Application

Coronavirus Dashboard app displays graphs to compare the daily number of cases/deaths, average number of cases/deaths, growth rates, and number of cases and deaths relative to the population for all states in the U.S.

## Project URL

https://ejdeposit.github.io/frontend-project/

## Github Repo

https://github.com/ejdeposit/frontend-project

## Journal

1. I started by making a function to make a single API call using fetch to get the historical number of cases and deaths for a single state. Next I needed to figure out how to make multiple api calls while waiting on the results of each call before preceding to graph the results. After reading a little bit about promises and not totally understanding how to make it work I ended up using the async/await functions to make the ajax call.

2. Next I implemented functions to graph the data for each state. I started by just graphing the numbers from the API call directly, then I figured out how to graph dates which required parsing the strings from the api call and figuring out how to graph dates with charts.js. I had to tinker around with the code for a while to get it to actually graph by date, and in the end I wan't not sure what i did to fix the problem. Then the dates weren't correct because months ware zero indexed, so I had to fix that.

3. Added event to check boxes for states so that a new graph is made each time the user selects a new state or unselects a state.

4. Added data structure to save results from each API call to avoid duplicate api call. This required refactoring code and working through bugs that occurred as a result.

5. Added radio input to graph deaths API call. Just required tinkering with existing code and adding some parameters to existing functions calls.

6. The graph scale would get all messed up after making a graph multiple times from checking and rechecking different states. The solution was to use chart destroy() I had to add some additional parameters and return values to some functions so that I cold catch the chart object variable, but the chart was a promise so I was still having trouble calling destroy() on it. The solution was to call Then() on the promise and then use destroy within then().

7. Created a python script to make all input checkbox elements for each state.

8. I put the check boxes inside a flex container to make them look more neat. I wanted them to be in arranged in multiple columns so first I laid them out in a column but this made one long column unless I set a max height for the container, but I didn't want to set a max height because I wanted the checkboxes to take up more room vertically if the user

resizes the window. The solution was was to arrange the checkboxes in rows, set the width of the checkbox flex item, and use space between to arrange them in resizable columns.

9. I decided to make graph as soon as page loads to give better visual clue about what to do for user. After reading the Design of Everyday Things, I decided it would be confusing to present the user with a bunch of check boxes with no indication of what checking them wold actually do.

10. I added a flex container that contained the two input containers and the graph so that I could center everything horizontally and add margins to the sides of the page.

11. Added fonts by importing them from google in the css. Added event listener for graph type radio inputs so new graph is made each time new graph option is selected

12. Added rolling seven day average graph. debugging took some time.

13. Added graph for two week cumulative number of cases and deaths. Tried to splice the list of days, but but there seemed to be some issue with the splice method where the length property didn't update properly when you attempt to splice more than is in the list then it takes the remaining and leaves the list length unchanged leading to an infinite loop. The solution was to use a for loop instead of a while loop.

14. modified first python script to create new script that makes object with populations of each state. Used this to divide two week totals by population to get number of cases per 1000 people in the state.

15. When refactoring the code, I accidentally deleted a variable. but code kept runnings as if the variable were still accessible in the function, despite the fact that it wasn't a global variable or declared in the outer block of the function. This led to really confusing error that took a while to pinpoint.

16. Added growth rate graph

17. Added object to save the color of each state, so the lines don't change colors each time you add a new state to the graph

18. Fixed accessibility. Added fieldset and legends, page sections, got rid of skipped heading level.

19. Added button to change colors in case the colors on the line graph are too similar.