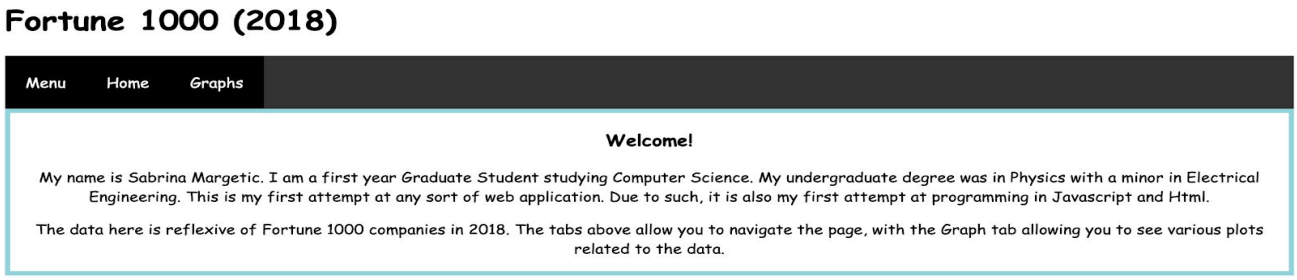
CSE 564 - Project 1

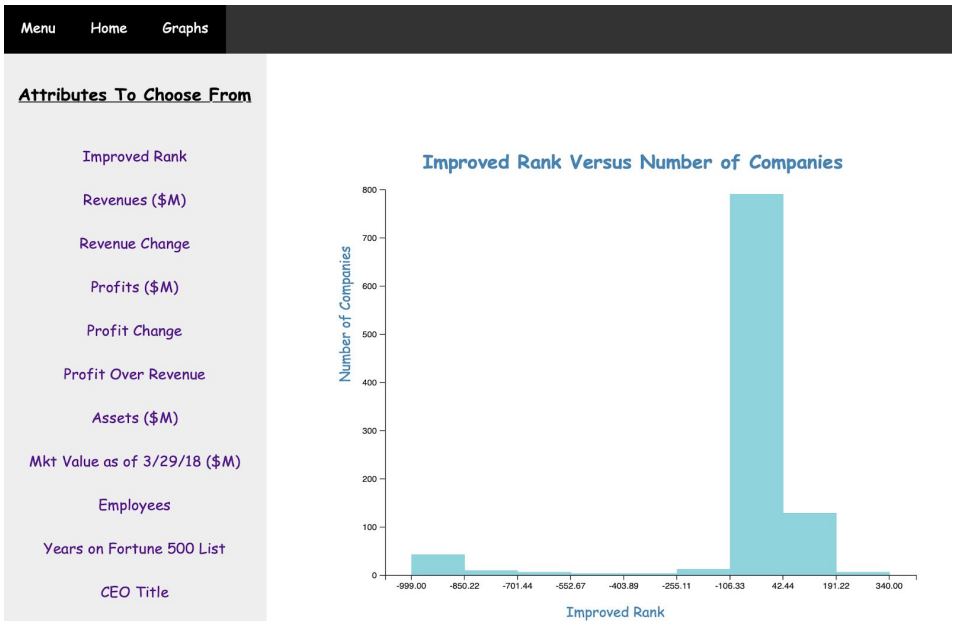
By: Sabrina Margetic

My project focuses on the data of companies that made the Fortune 1000 list in 2018. To start off, I created 3 files titled “index.html,” “index.js,” and “index.css.” The index.html file links the other files, presents all major text, and links menu items to corresponding functions that need to be executed on click. The index.css file is used for appearance aspects.

There are two main pages (navigable by the menu bar located at the top), one is the “Home” section and one is the “Graph’s” section. The home section presents a small prompt (written in the html file) explaining who I am and my background (shown below).



The Graphs section presents created graphs as well as a menu corresponding to the data of the graphs, as shown below:



As you can see, the first presented graph is the “Improved Rank Versus Number of Companies” graph.

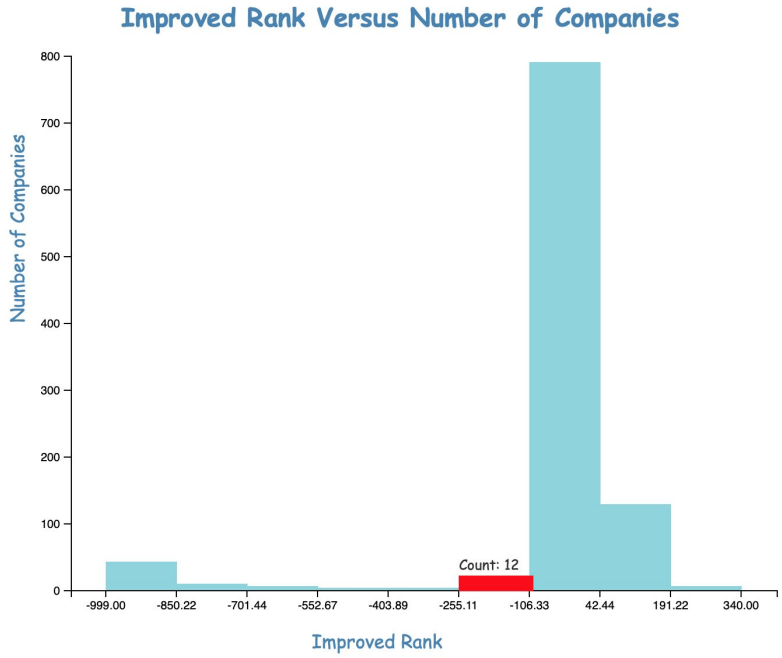
The code written for the graphs section is written in the index.js file. The main functions of the index.js file are: createContinuousGraph(), createDiscreteGraph(),

getDataToMakeDiscreteGraph(), getDataToMakeDiscreteRegionGraph(), getDataToMakeContinuousGraph(), getDataForProfitOverRevenue(), mySliderFunction(), and getDataToMakeDiscreteCityGraph(), excluding input variables.

The getDataToMakeContinuousGraph(), takes in the particular attribute that the graph is being created for and reads in the corresponding data. If necessary, it parses the data, in order to generate a pure numerical value. It then obtains the max and min values, used later for scaling purposes of the x-axis. Finally, it creates an array of counts, the length of the number of ticks requested (discussed further later), and counts how many companies fall below the tick. Tick mark locations are calculated by subtracting the min from the max and dividing by the number of requested ticks. They are implemented in successive increments.

There are 10 attributes for which bar graphs from continuous (numerical) data are created: “Improved Rank,” “Revenues ($M),” “Revenue Change,” “Profits ($M),” “Profit Change,” “Profit Over Revenue,” “Assets ($M),” “Mkt Value as of 3/29/18 ($M),” “Employees,” and “Years on Fortune 500 List.” Data is collected for all using the getDataToMakeContinuousGraph(), except for Profit over Revenue due to the need for additional calculations (instead the getDataForProfitOverRevenue() function is used).

At the end, the getDataToMakeContinuousGraph() calls upon the createContinuousGraph(), which ultimately generates the respective graph. It creates both a x-axis and y-axis scale in order to generate the respective axes with labels. It creates blue bars to depict the number of counts in each category. Additionally, the bars of the graph turn red, expand width and height, and write the number of counts above the bar, when hovering. An image depicting this is shown below:



Similarly, the getDataToMakeDiscreteRegionGraph() takes in a categorical attribute, and produces unique x values, and number of companies for each unique x value. The function to the right produces the unique x values:

```
function uniqueValues(value, index, self){
  return self.indexOf(value) === index;
}
```

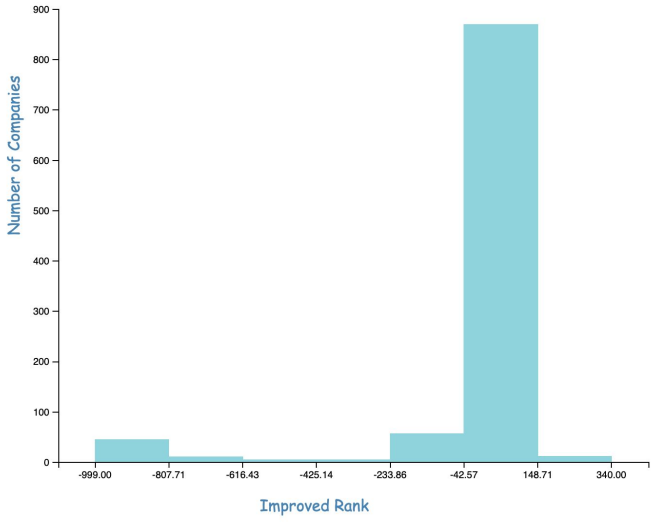
There are six categorical attributes: “CEO Title,” “Sector,” “Industry,” “City,” “State,” and “Region.” However, the Region and City ones require separate functions (getDataToMakeDiscreteRegionGraph() and getDataToMakeDiscreteCityGraph(), respectively). Region data is not provided in the given data, but is calculated through the state attribute. The city graph has a minimum requirement of a count of 3 companies, due to the massive amount of cities with small counts, making it clustered.

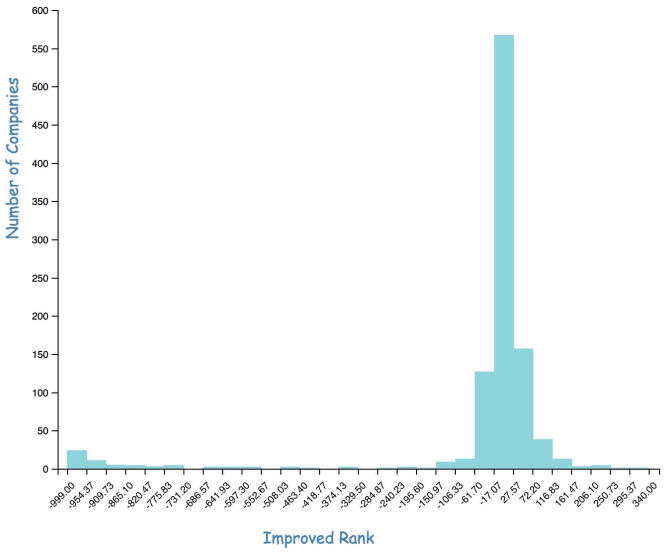
At the end, the createDiscreteGraph() is called upon in these functions. It produces a similar graph to the one created by the createContinuousGraph() function, except, the x-axis has categorical values.

The slider is initially declared in the html file:

```
<div class="slidecontainer" id = "mySlideContainer" >
  <input  type="range" value = "10" max = "41" min="3" step = "1" class="slider" id="mySlider" onchange = "mySliderFunction()">
  <!-- <range id ="myRange"></range> -->
</div>
```

On change, it calls the mySliderFunction(). The function changes the number of ticks (also referred to as “numberOfBoxes” in the code), to the value that the slider provides. A global variable holds the attribute currently working with, and calls on getDataToMakeDiscreteGraph() and getDataToMakeContinousGraph(), accordingly (since the functions use numberOfBoxes to generate data accordingly). An example of a decrease in





number of ticks, and an example of an increase in number of ticks are located, below in their respectively.