**CS573 Data Visualization   
Final Project Process Book (Draft)**  
*By Christina Aiello*

# Week 1: Just Implementing the Visualizations

During week 1 I wanted to just implement the basics for the visualizations I would be using (and not add in any extra features yet). Some of my code that I reused from previous assignments already had some extra features in it (ones I created previously), so I chose to keep those features in my code but didn’t add any new features to start. The charts I’ve chosen to implement are bar graphs, pie charts, calendar charts, and word clouds.

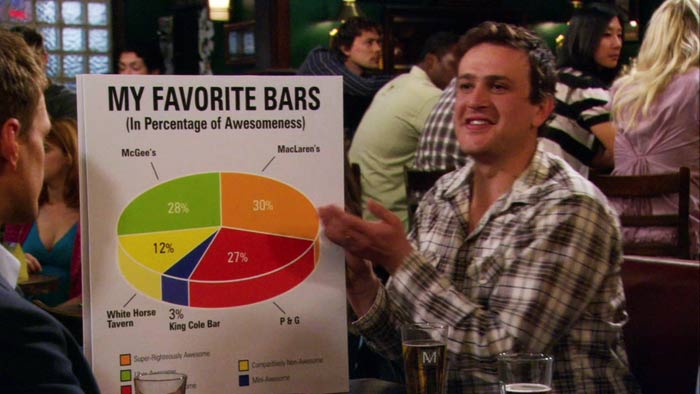


Figure 1: Bar graphs and pie charts?

## Preprocessing the Data

The data I have had to be preprocessed to some extent. Changes I made were:

1. Changing all instances of the ampersand (&) in the “Question” column to be the word “and.”
2. Removing any quotation marks from all values in the “Question” column.
3. Removing the dollar signs from the values of questions to allow d3 to sort the values in numerical order.
4. Changing the “Air Date” to be specifically in MM/DD/YYYY format (some were in M/DD/YYYY if the month only had one digit)
5. Removing the exclamation points from the “Round” attribute (no real need, just done for aesthetics).

## Bar Graph

I’ve worked with Bar Graphs this entire term, so getting these to work did not take much time thankfully. The bar graphs I had at this point would filter the data based on certain attributes (first by round, then by year, and lastly by value). If you hover over one of the bars in the first bar graph, the middle bar graph appears with a further breakdown of the data. The same occurs if you hover over the middle bar graph: A final bar graph appears to the right with even more specific data.

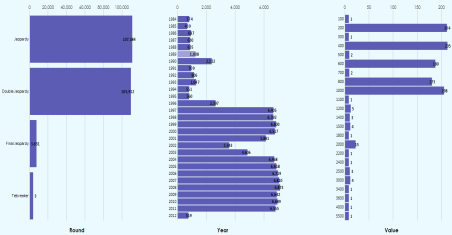
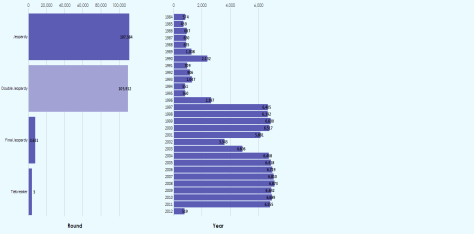
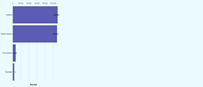


Figure 2: These three images show the process of making the various graphs appear on the screen. The leftmost image shows just the first graph that appears. The middle and rightmost images show how to make the second and third graphs appear on the screen.

Below I show the three graphs in closer detail.

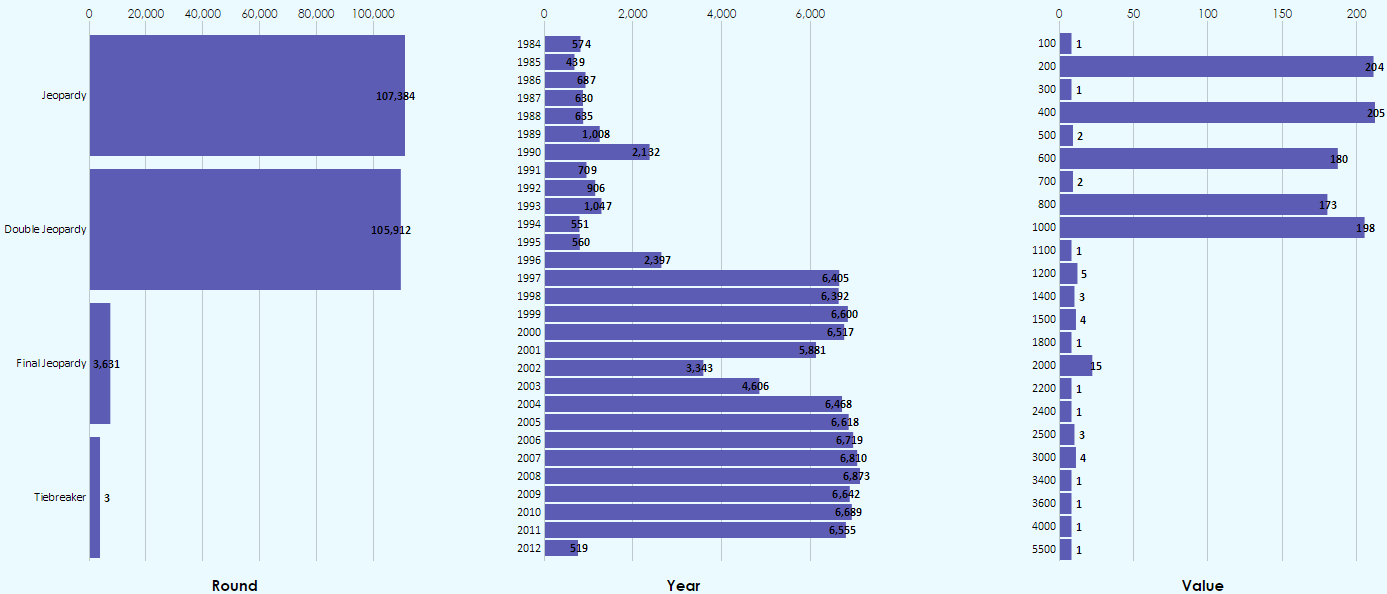


Figure 3: The leftmost graph is broken down by the "round" attribute. The middle graph is broken down by year. The rightmost graph is broken down by the question's value.

## Pie Charts

I have also used pie charts in previous assignments, so getting these two work was also not too complicated. One change I did have to make was to, rather than change the color of a pie slice that is hovered over, change the color of that pie slice’s label. (Changing the colors of a pie slice can get messy since not every pie slice is the same color to start with, which the way bar charts can be.) Another change I needed to make was in regards to the labels on my data in the pie charts. With some pie charts having dramatically different slice sizes (one value may be 400 and the other may be 1 or 2), the labels for the pie charts became very hard to read. The image below depicts this.

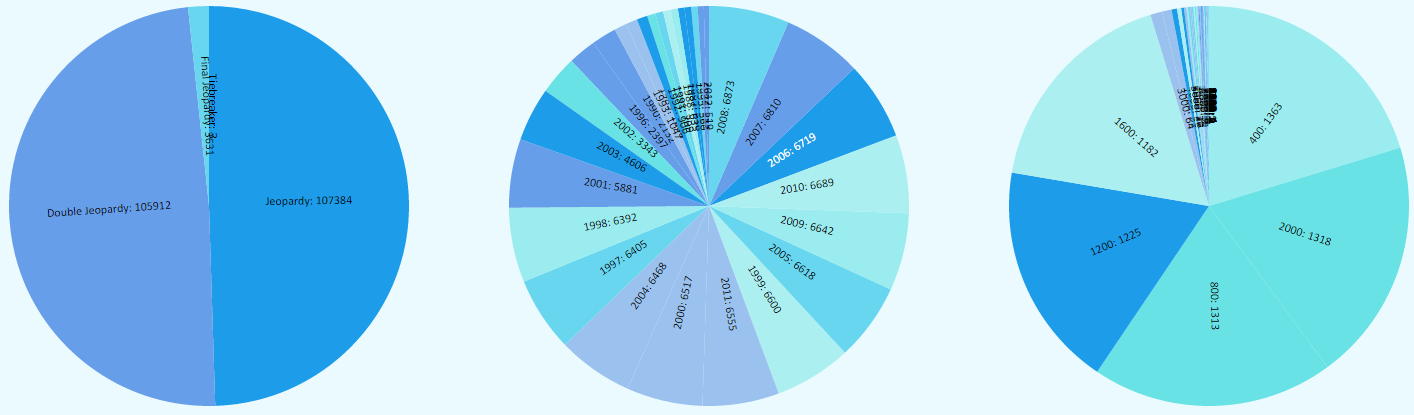


Figure 4: Unfortunately the labels are no longer readable. They need to be moved outside of the pie charts.

I tried a workaround to get the label for the pie chart to show up outside of the pie slices, however that didn’t end well. The image below depicts the result. While I could create a legend for the data, this would require having a set of over 30 or so colors, and I feel as though legends with that many colors are overwhelming. I think I’ll scrap the pie chart idea. Boo pie charts.

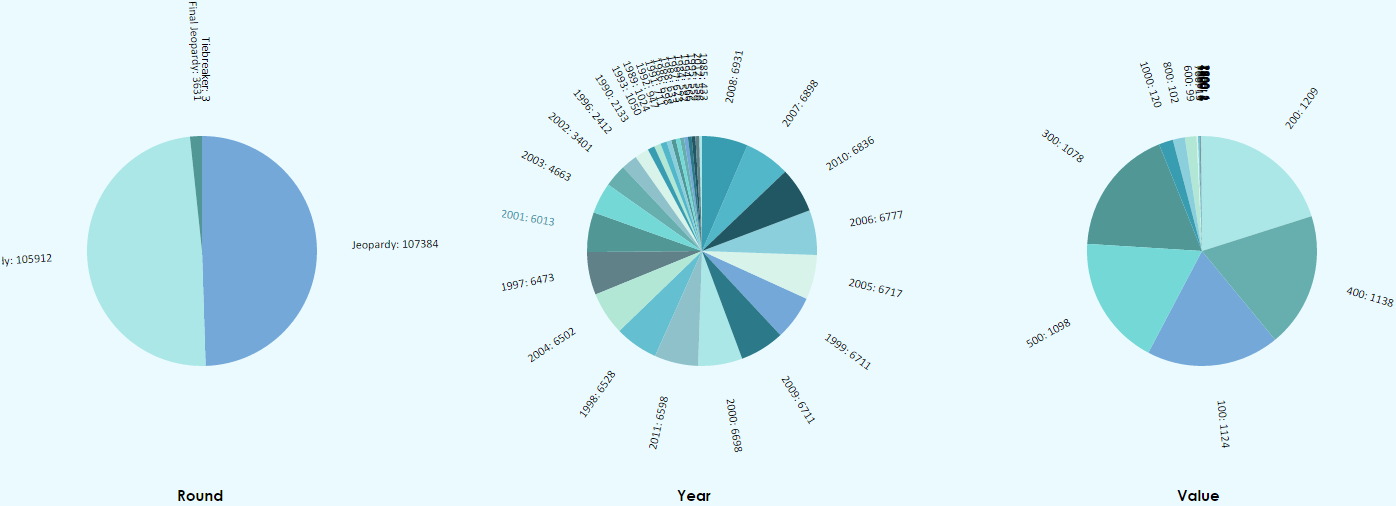


Figure 5: Even after moving the labels outside of the pie slices, the labels were unfortunately still unreadable.

## Calendar Chart

After a bit of battling with d3, I was finally able to get my calendar chart to display properly (Protip: When you want to color something in, use CSS to color it… I thought d3 wasn’t working and that was why my chart displayed in all black, but I was just missing some CSS). I realized that organizing the calendar view in a certain way (put two years next to each other and making them small) I was able to avoid having to aggregate the data by year and could keep the entire MM/DD/YYYY set of information. I chose to have the third chart in my set of 3 be a calendar chart.

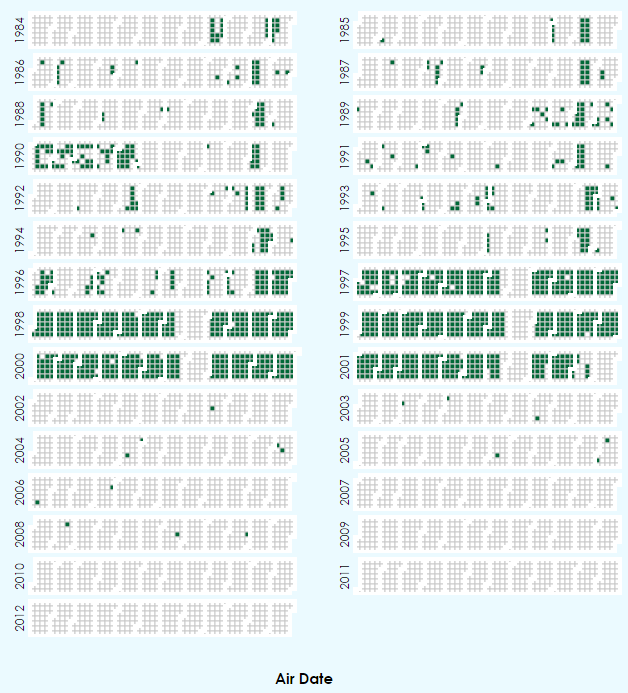


Figure 6: The figure above shows the resulting calendar chart that I have created.

Once I got the Calendar Chart working, I needed to do some adjusting of the code to get it to correctly display the color scheme in regards to the values. In the above image, all squares were being assigned the darkest color green rather than being given a correct value. My changes to actually incorporate the number of occurrences of each date ended up looking like the following image:

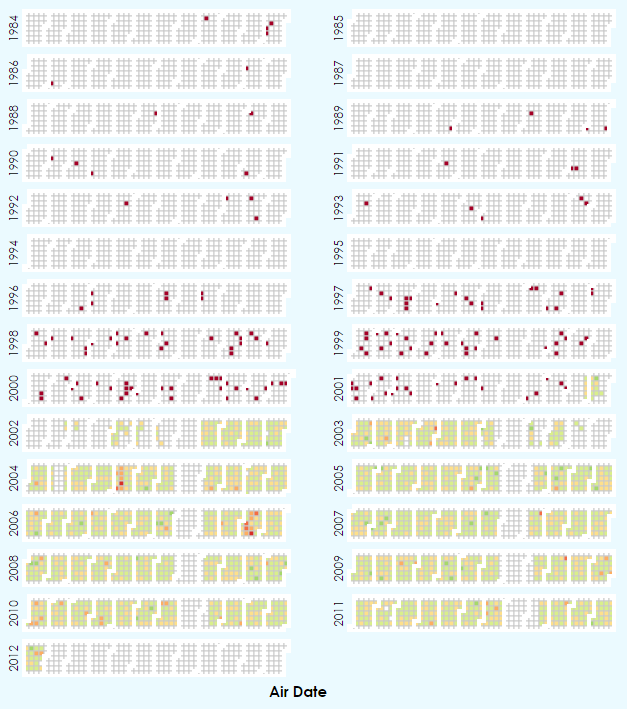


Figure 7: Red represents dates with smaller values, and green represents dates with bigger values. The scale goes red to orange to yellow to green.

While I was happy with how this looked at first, we were taught in class that using a range of red -> orange -> yellow -> green colors as a scale is not the most effective method for coloring values, so I chose to make all colors blue instead. Lighter blue meant a smaller value, and darker blue meant a larger value. The resulting chart can be seen below:



Figure 8: Shown above is my revised calendar graph.

## Word Cloud

With words being such a valuable part of my data set, I knew I wanted to include a Word Cloud in my project in some way. I first found this Word Cloud instance, which was created by Jason Davies (<https://www.jasondavies.com/wordcloud/#%2F%2Fwww.jasondavies.com%2Fwordcloud%2Fabout%2F>).



Figure 9: I really liked how this looked, and I thought it could be an interesting way to show which words most frequently appear in Jeopardy questions.

Which lead me to the creator’s website, which explained how the Word Cloud library he had created worked in more detail (<https://www.jasondavies.com/wordcloud/about/>).

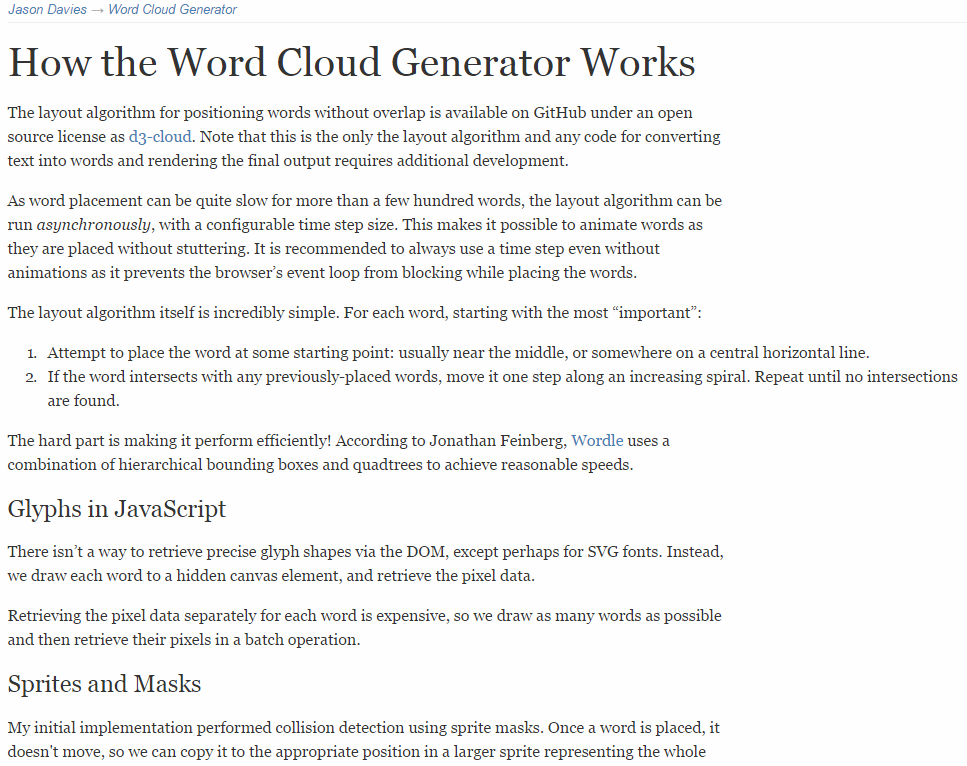


Figure 10: Jason’s website about the Word Cloud he had created.

Jason did a wonderful job of explaining his d3 Word Cloud library, however I tend to work best by example, so I kept searching. What I then found to be the most helpful was this implementation of their library (<http://bl.ocks.org/ericcoopey/6382449>).

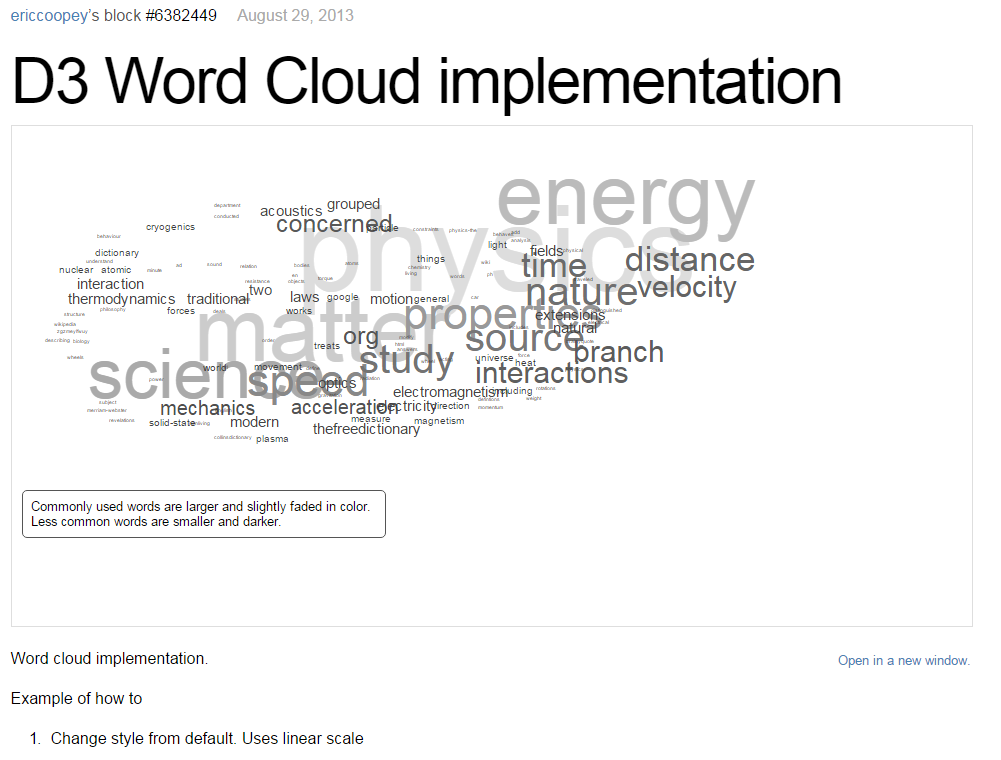


Figure 11: This example Word Cloud was the most straightfoward and understandable for me, so I used this as my reference.

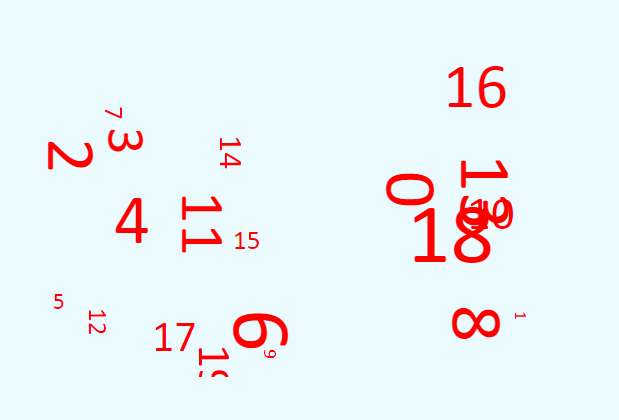


Figure 12: My first attempt at getting the Word Cloud working. This should have had actual words in it. Well... it’s something...

Eventually I did get the Word Cloud working, and it looked something like the image below. While I was happy with it, I realized that a lot of words I deemed “boring” (prepositions, for example) ended up being the most frequently seen words, and I feel as though these aren’t words people are curious about. Because of this, I chose to create an array of filtered words. If any of the words in any questions are in the filtered word list, they were skipped over and weren’t added to the Word Cloud.

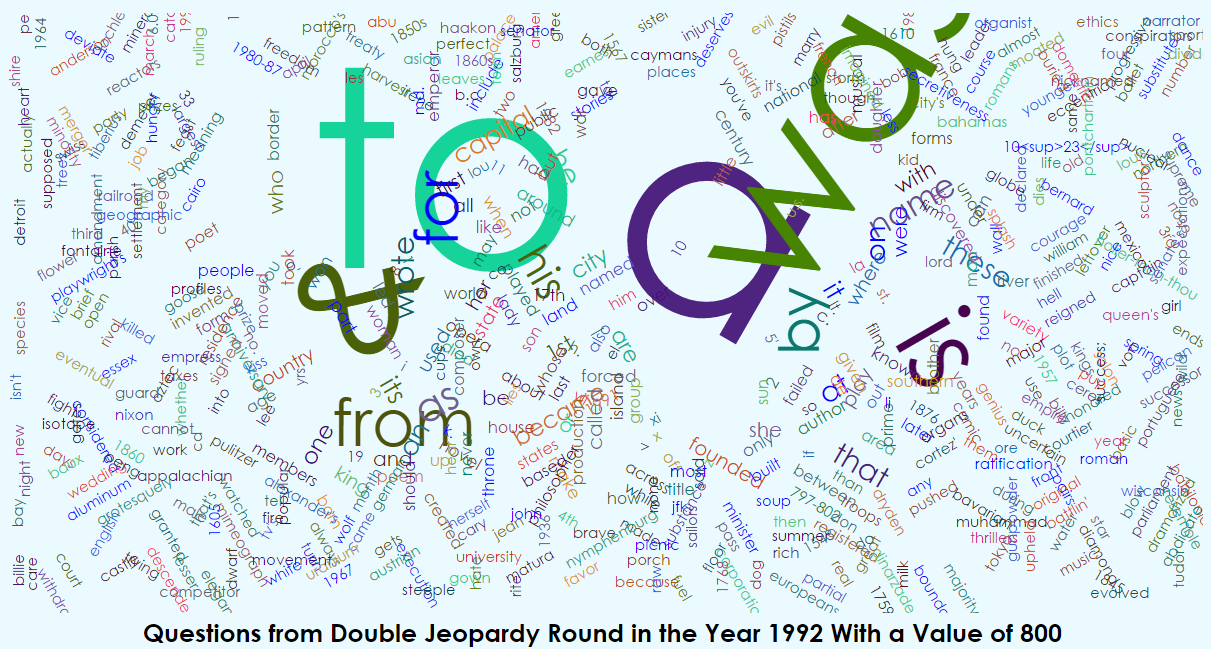


Figure 13: I was just so curious about how many times the word "a" appeared in questions, right?... Probably not.

In addition to filtering out words, I also had to turn all words to lowercase using JavaScript’s .toLowerCase() method due to the same word showing up in the Word Cloud in different cases (Example: “He” versus “he”). Next I wanted to give the Word Cloud some sort of title, so I chose to construct a title in the format “Questions from ROUND in the Year YEAR With a Value of VALUE,” filling in “ROUND,” “YEAR,” and “VALUE” with their respective values from the data set. Eventually I had created the visualization below.

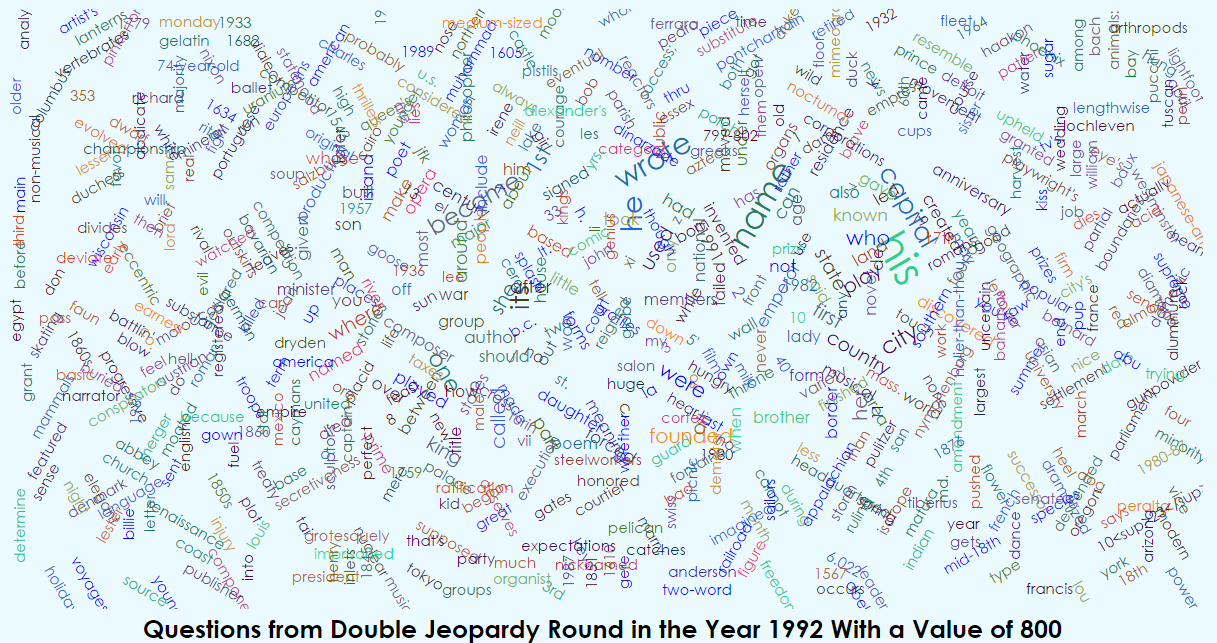


Figure 14: My eventual Word Cloud, which I felt was much more interesting after removing prepositions and other more "plain" (in my opinion) words. This is the same data set as the previous image, just with the filtered words removed.

I then had to hook together the Word Cloud and the bar graphs I’d created. Hovering over a data point in the rightmost bar graph still changes its color to a lighter purple (which happens with all three graphs). However, when a user clicks (not hovers over, but actually clicks on) a data point (a bar) in the rightmost graph, a Word Cloud would be generated below. Once the Word Cloud had loaded, the selected bar will turn green:



Figure 15: The left image shows what the bar graph looks like when the data is loading for the selected data point. The right image shows that the chosen data point turns green to confirm that the Word Cloud finished loading below.

Originally my plan wasn’t to add extra features, just implement the graphs, however this was a very trivial change that I was able to make in a few minutes due to previous code that I’d written.

# Week 2: Post Feedback from Proposal

Now that I have received feedback on my proposal, I have decided to not aggregate my “Air Date” data by year and instead keep the MM/DD/YYYY data in its entirety. In addition, I have chosen to not allow users to choose which visualization they want to see, and instead I will just offer bar graphs, calendar charts, and word clouds. I would also like to incorporate my extra feature (searching questions and answers) as well.

## Adding More Value to Calendar Chart – Word Cloud Incorporation

The calendar chart code that I referenced was originally rolling up the data that was aggregated by date, meaning that for every date in the data it would count up the number of occurrences rather than preserving the array of objects representing each question that had a matching day. Rolling up removed the extra data (question and answer) that I wanted to use in my ca lendar chart, so I changed a few lines to preserve this data. The line below is the line I changed:

var data = d3.nest() .key(function(d) { return d.Date; }) .rollup(function(d) { return d.length; }).map(data[0].values);

The line above became:

var data = d3.nest() .key(function(d) { return d.Date; }).map(data[0].values);

Making this change matched up an array of data points (which contain the question, answer, air date, etc.) with a particular date in the calendar, rather than just counting the number of matching data points per date in the calendar.

Now, when users click anywhere in a year in the calendar chart, the same word cloud from before appears below. The downside to adding the word cloud in this way means that I now have to loop through all data points (from the entire group of points that was plotted in the calendar graph) to see which of them are of the correct year. I then have to do the same procedure as before to find word frequency: I have to look at each word in each question and count its occurrences in all questions from a particular year (well, round -> value -> year). This still gives me data sets of size 9,000-12,000 or so in some places, meaning that my algorithm will be slow to create the word cloud. This means I will need to optimize my word counting to speed up this process. I also should add in a loading message of some sort.

## Extra Feature: Search for a Word

What I feel provides the most value to users is the content of questions, answers, and categories. While graphing those is particularly challenging (other than using a Word Cloud), I wanted some way to let users know how often a word appears in the entire data set and not just in a particular drilldown of round -> value -> year. I created an extra feature (a word searcher) which will take in a singular word and search for that word in all questions, answers, and categories in the entire data set. While a word cloud would be overwhelming for the entire data set in my opinion, I feel that a bar graph that shows the number of occurrences of a word in all questions, all answers, and all categories would be useful. Seen below is my implementation of this feature.

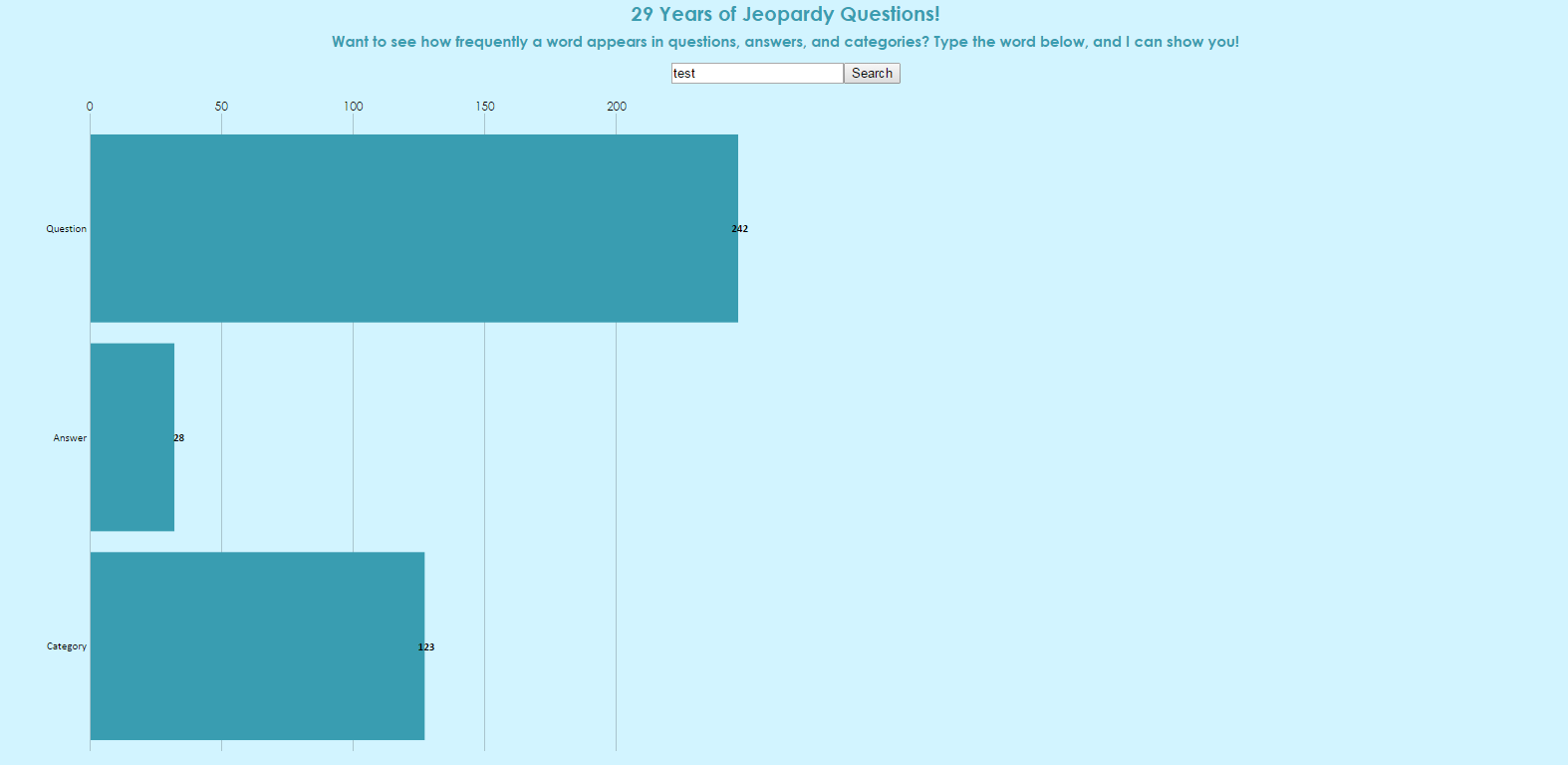


Figure : The resulting bar graph from searching for the word "test."

I added JavaScript validation to this tool to prevent the user from searching for empty words or strings with spaces. When a user tries to search for one of the aforementioned items, they are greeted with this message:

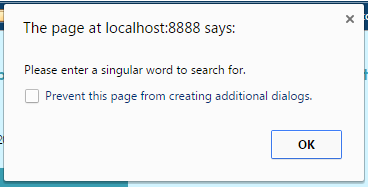


Figure : The error message that is shown when a user tries to search for what I've deemed to be "invalid" input.

For ease of use, I also included JavaScript code to let the user press “Enter” on his or her keyboard to submit the query, in addition to letting them use the “submit” button that I created.