

CS573 Data Visualization Final Project Process Book

By Christina Aiello

Table of Contents

- [Overview and Motivation](#)
- [Week 1: Data Preprocessing and Implementing the Visualizations](#)
- [Week 2: Post-Proposal-Feedback Work](#)

Overview and Motivation

My family has a fascination with trivia. They greatly enjoy watching Jeopardy and other trivia shows, they play Trivial Pursuit to show off their skills, and they'll even just end up browsing Wikipedia pages for hours to read about different random topics. Not only is trivia something that family members of mine enjoy, but my friends and acquaintances (and really, many people in general) have a love for trivia games. Many bars and other venues will have trivia nights (We even have one here at WPI every week) where participants form teams and try to correctly answer as many trivia questions as they can. With this love for trivia questions in mind, I searched for data sets related to trivia and stumbled upon this fantastic data set of 216,930 Jeopardy questions.

I would like to give users the ability to view the frequency of rounds, values, and air dates in this data set. In addition, I want to let the user see which questions end up being in the Double Jeopardy and Final Jeopardy rounds most often, potentially helping a user target his or her knowledge to categories that end up in these rounds. I plan to offer three different graphs (first a breakdown by round, then by value, then by air date), and finally I want to create a Word Cloud for the user to see. I would also like to create a special feature that lets a user search for a word in all categories, questions, and answers in the entire data set.

Week 1: Data Preprocessing and 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.



Bar graphs and pie charts?

Preprocessing the Data

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

- Changing all instances of the ampersand (&) in the "Question" column to be the word "and."
- Removing any quotation marks from all values in the "Question" column.
- Removing the dollar signs from the values of questions to allow d3 to sort the values in numerical order.
- 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)
- Removing the exclamation points from the "Round" attribute (no real need, just done for aesthetics).

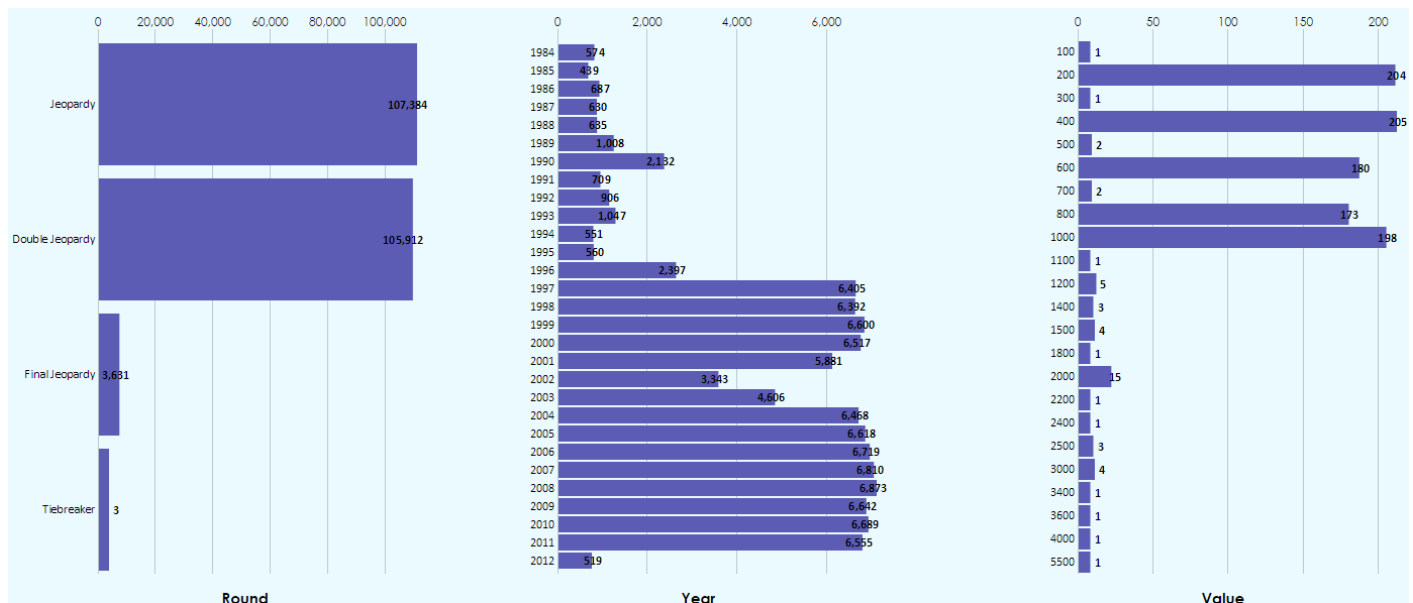
Bar Graphs

Exploratory Data Analysis: 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.



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.

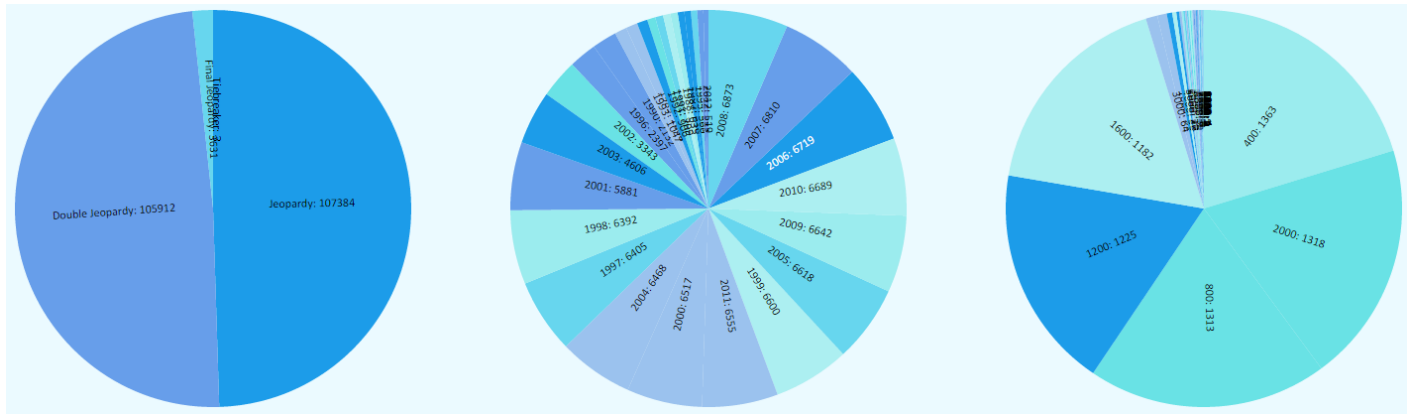
Below I show the three graphs in closer detail.



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.

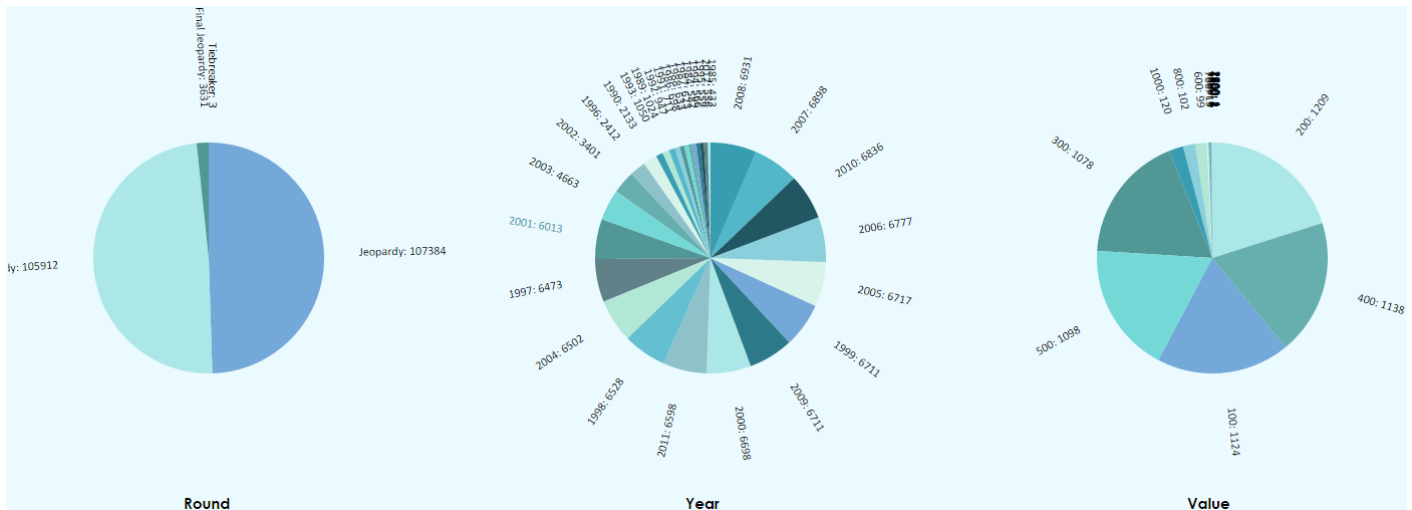
Pie Charts

Exploratory Data Analysis: 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.



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.

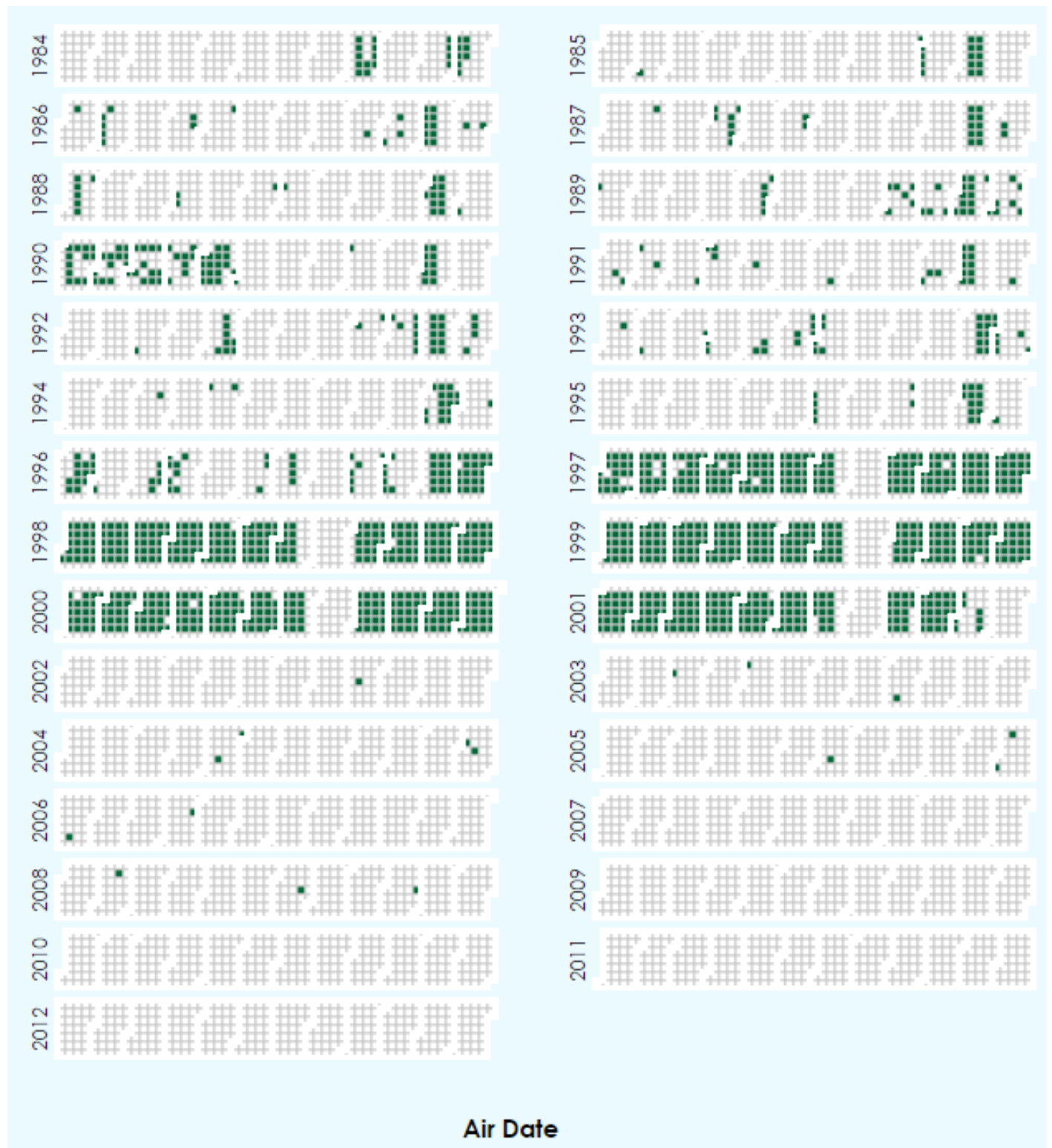


Even after moving the labels outside of the pie slices, the labels were unfortunately still unreadable.

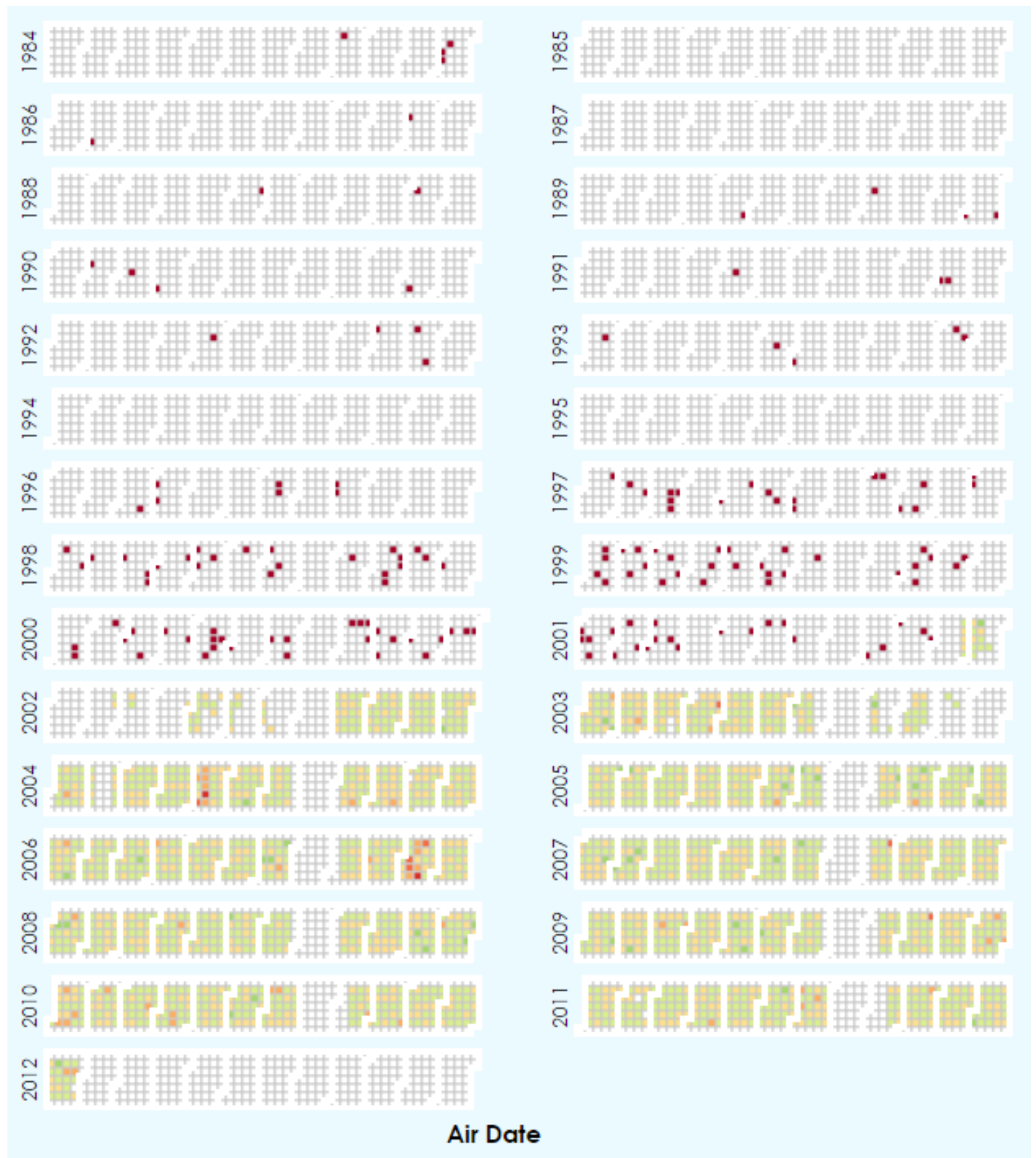
Calendar Charts

Related Work: I had seen calendar charts on [this](#) website and liked the way they arranged/displayed data, so I chose to include them in my project. 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, since that sometimes happens when d3 code isn't properly done, 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.

Exploratory Data Analysis: Once I got the calendar chart working, I realized not only that splitting the data by air date (rather than aggregating by month or year) was possible, but I also enjoyed the benefit the user gained from being able to see questions per day rather than grouped into an entire month or year.



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:



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

Design Evolution: 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:



Shown above is my revised calendar graph. It only uses shades of blue.

Word Clouds

Exploratory Data Analysis: 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. [Click to see his website.](#)

How the Word Cloud Generator Works

The layout algorithm for positioning words without overlap is available on GitHub under an open source license as [d3-cloud](#). Note that this is the only the layout algorithm and any code for converting text into words and rendering the final output requires additional development.

As word placement can be quite slow for more than a few hundred words, the layout algorithm can be run *asynchronously*, with a configurable time step size. This makes it possible to animate words as they are placed without stuttering. It is recommended to always use a time step even without animations as it prevents the browser's event loop from blocking while placing the words.

The layout algorithm itself is incredibly simple. For each word, starting with the most "important":

1. Attempt to place the word at some starting point: usually near the middle, or somewhere on a central horizontal line.
2. If the word intersects with any previously-placed words, move it one step along an increasing spiral. Repeat until no intersections are found.

The hard part is making it perform efficiently! According to Jonathan Feinberg, [Wordle](#) uses a combination of hierarchical bounding boxes and quadrees to achieve reasonable speeds.

Glyphs in JavaScript

There isn't a way to retrieve precise glyph shapes via the DOM, except perhaps for SVG fonts. Instead, we draw each word to a hidden canvas element, and retrieve the pixel data.

Retrieving the pixel data separately for each word is expensive, so we draw as many words as possible and then retrieve their pixels in a batch operation.

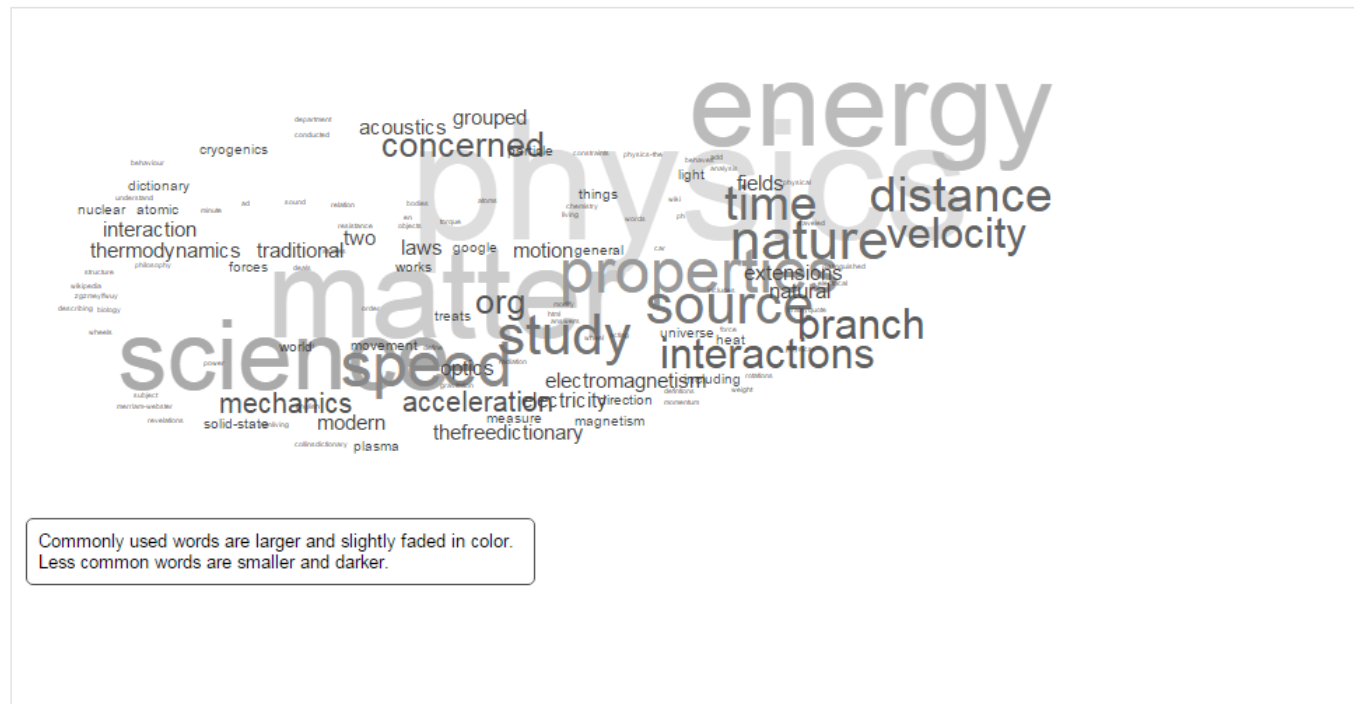
Sprites and Masks

My initial implementation performed collision detection using sprite masks. Once a word is placed, it doesn't move, so we can copy it to the appropriate position in a larger sprite representing the whole

[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](#).

D3 Word Cloud implementation



Word cloud implementation.

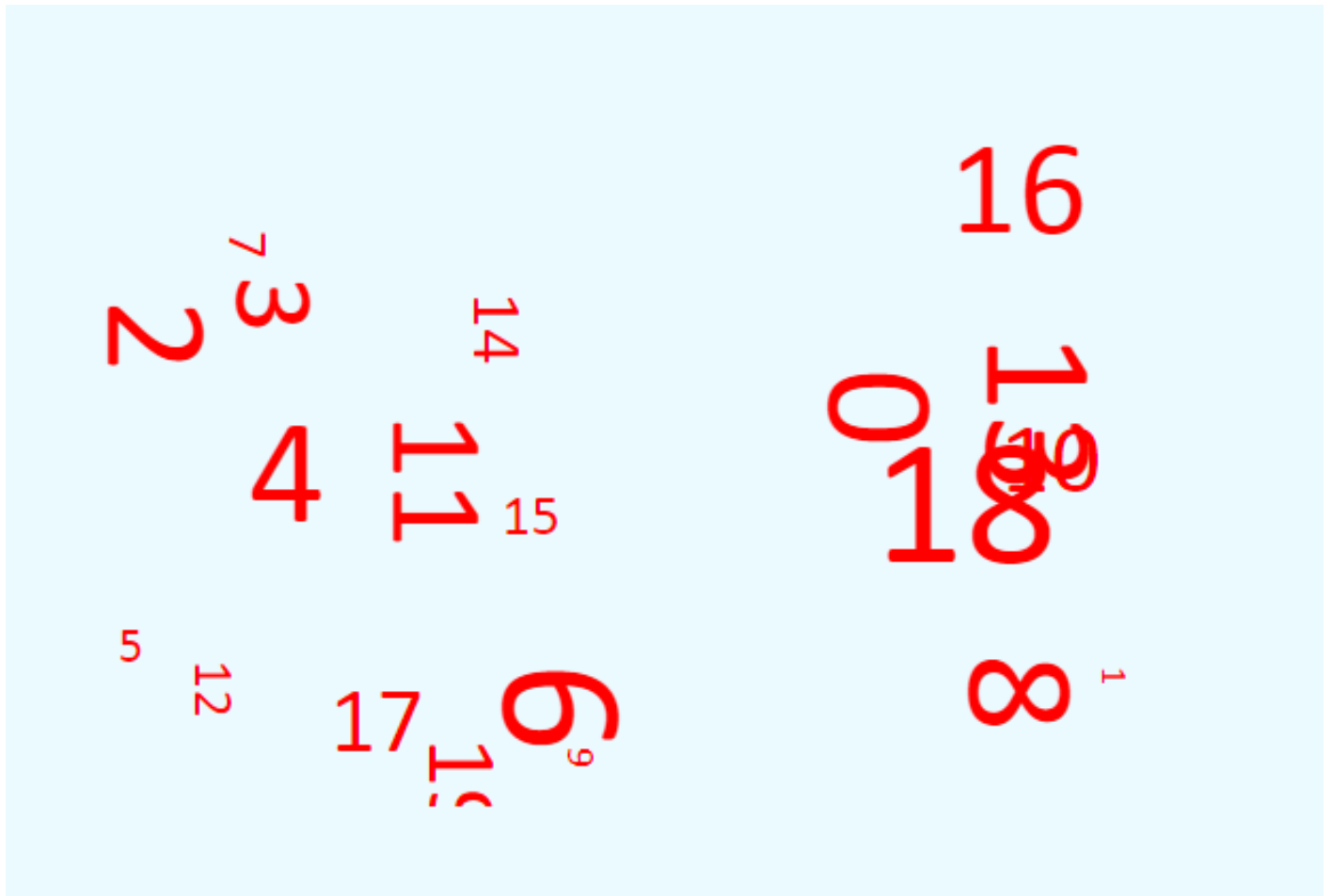
[Open in a new window.](#)

Example of how to

1. Change style from default. Uses linear scale

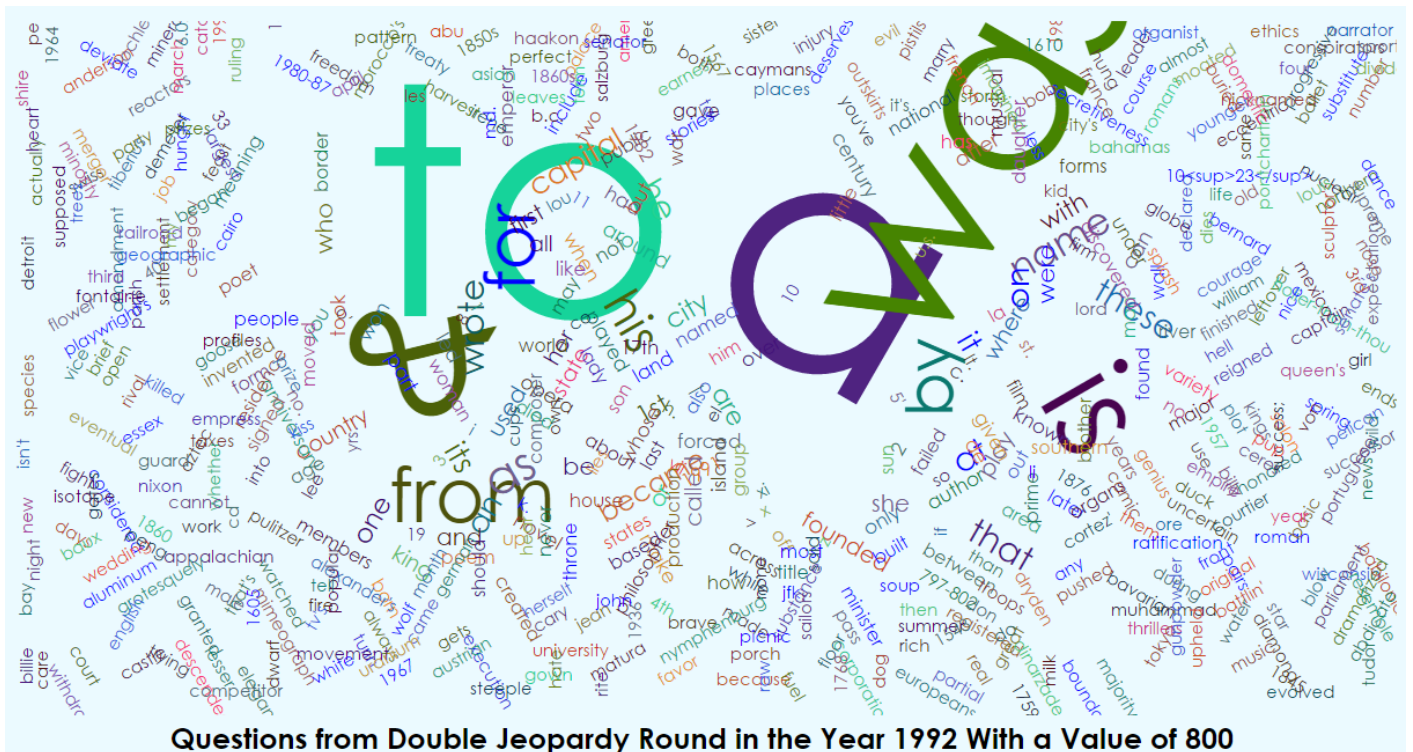
This example Word Cloud was the most straightforward and understandable for me, so I used this as my reference.

My first attempt at getting the Word Cloud working was a bit... sad. Something was showing up in cloud form, but not words...



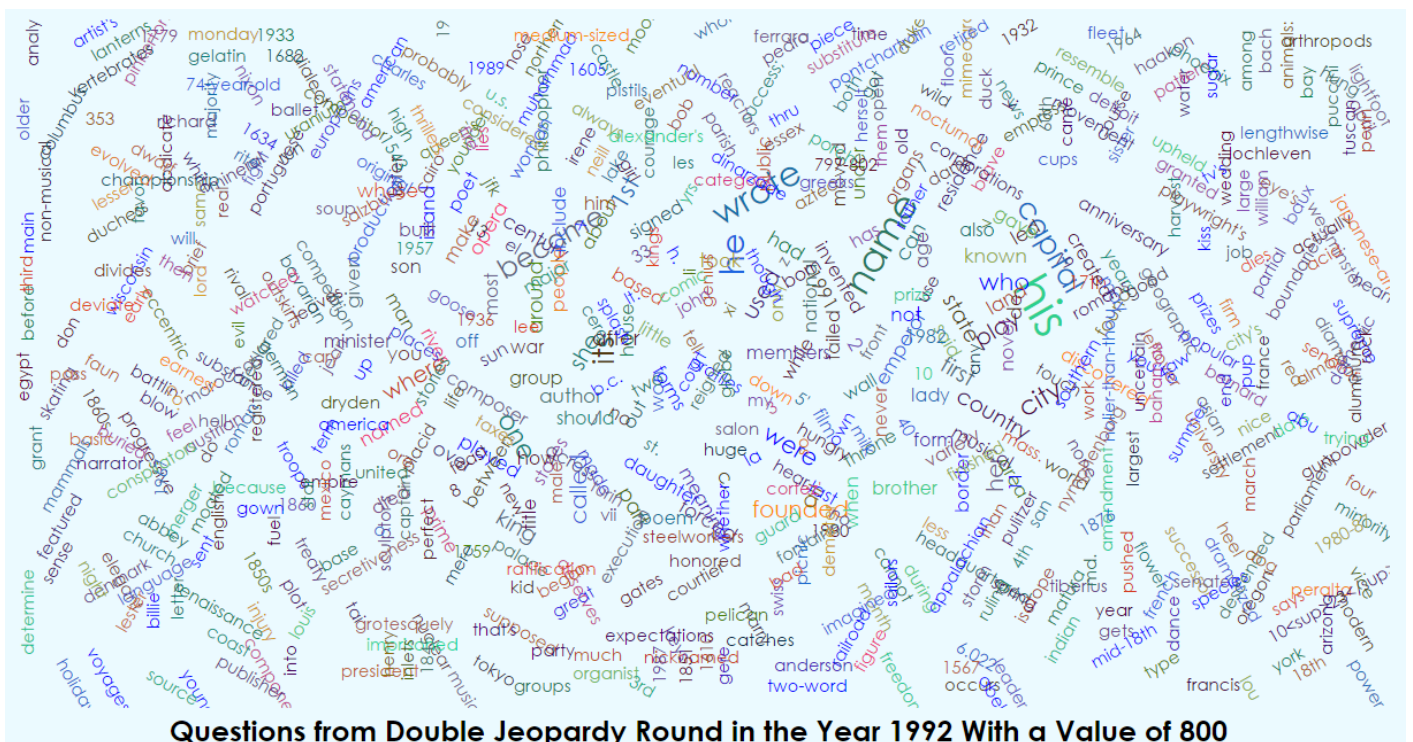
This should have had actual words in it. Well... it's something...

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.



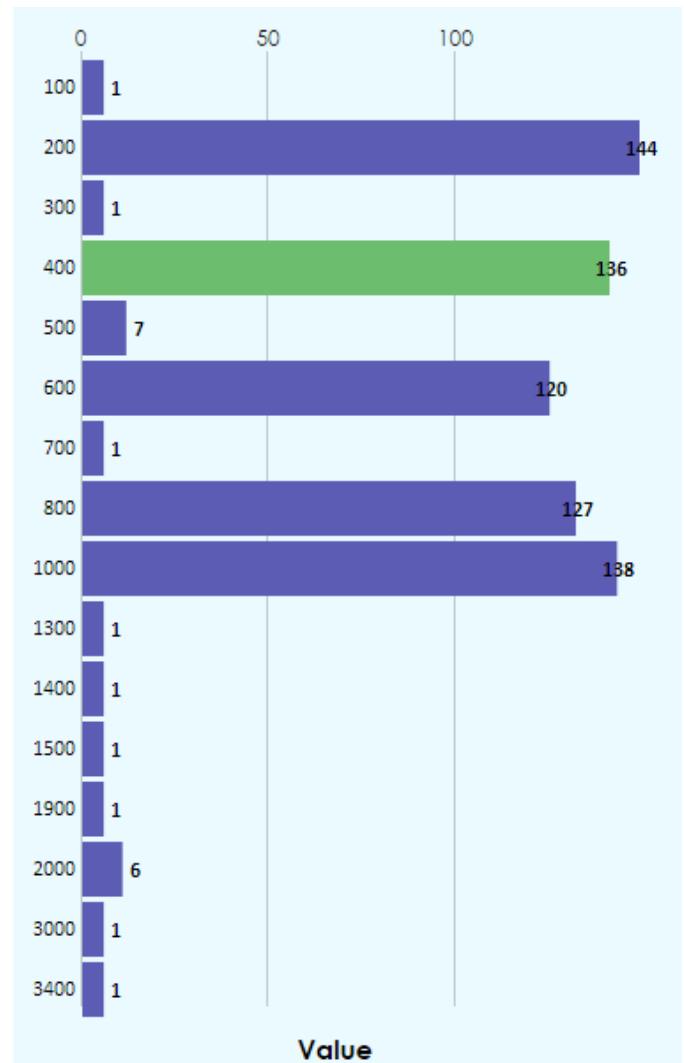
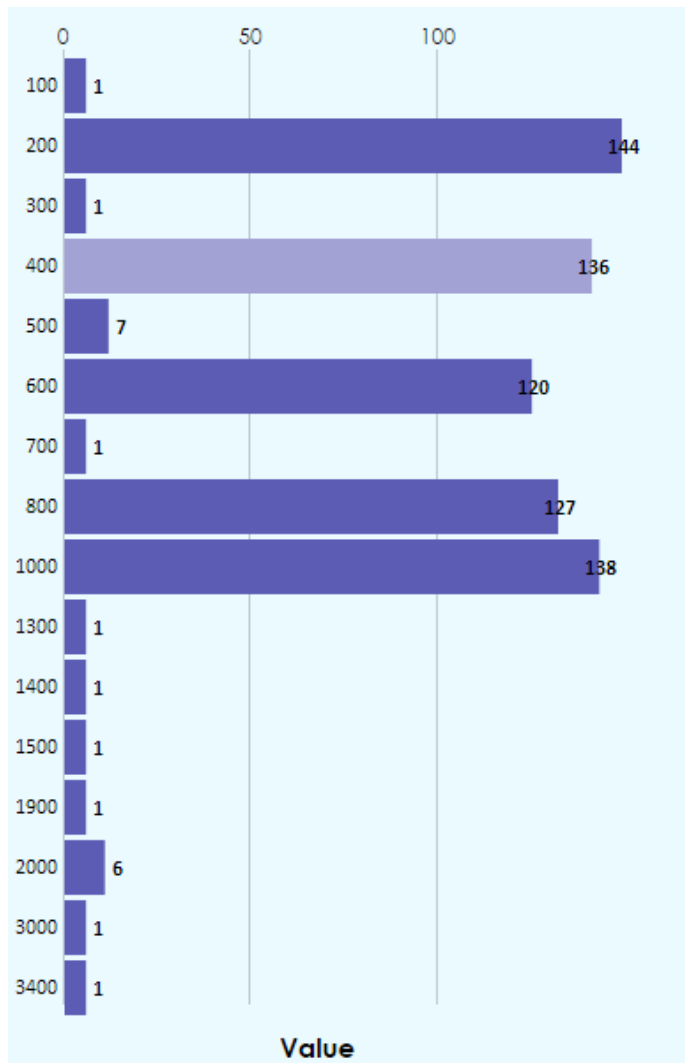
I was just so curious about how many times the word "a" appeared in questions, right?... Probably not.

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.



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:



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-Proposal-Feedback Work

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

Implementation: 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 calendar 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:

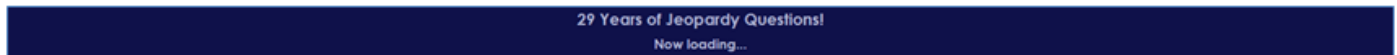
```
var data = d3.nest().key(function(d) { return d.Date; }).map(data[0].values);
```

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.

Adding Loading Message

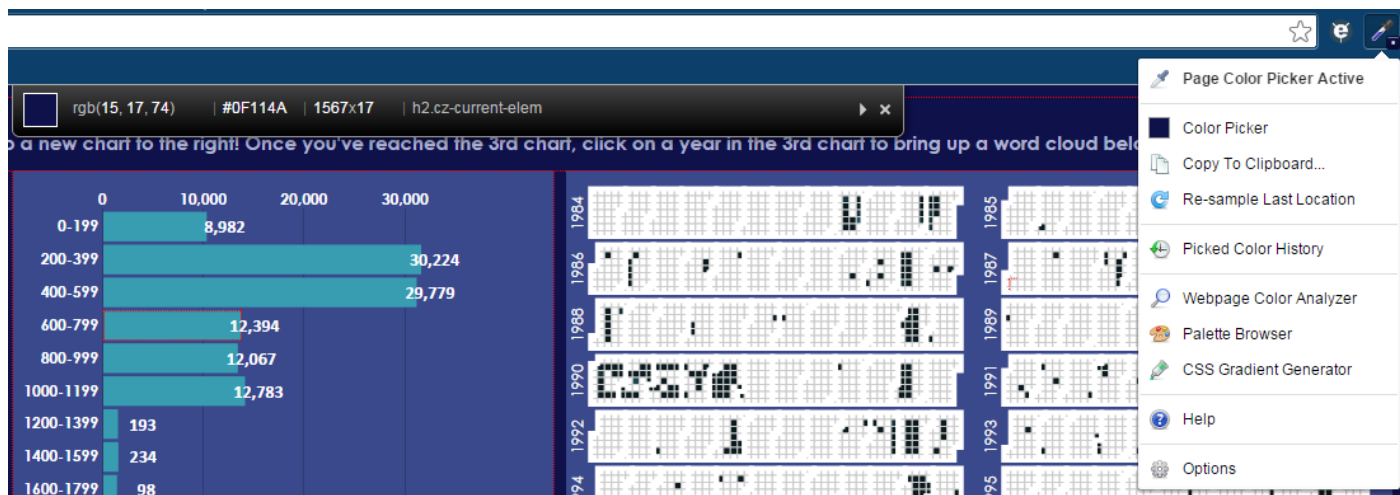
Due to the slow load count for the Word Cloud (and some instances of the Calendar Chart), I added a loading message at the top of the screen where the directions/explanation for the page is. When a data point is clicked, the loading message replaces the directions. Once the visualization has loaded, the directions reappear.



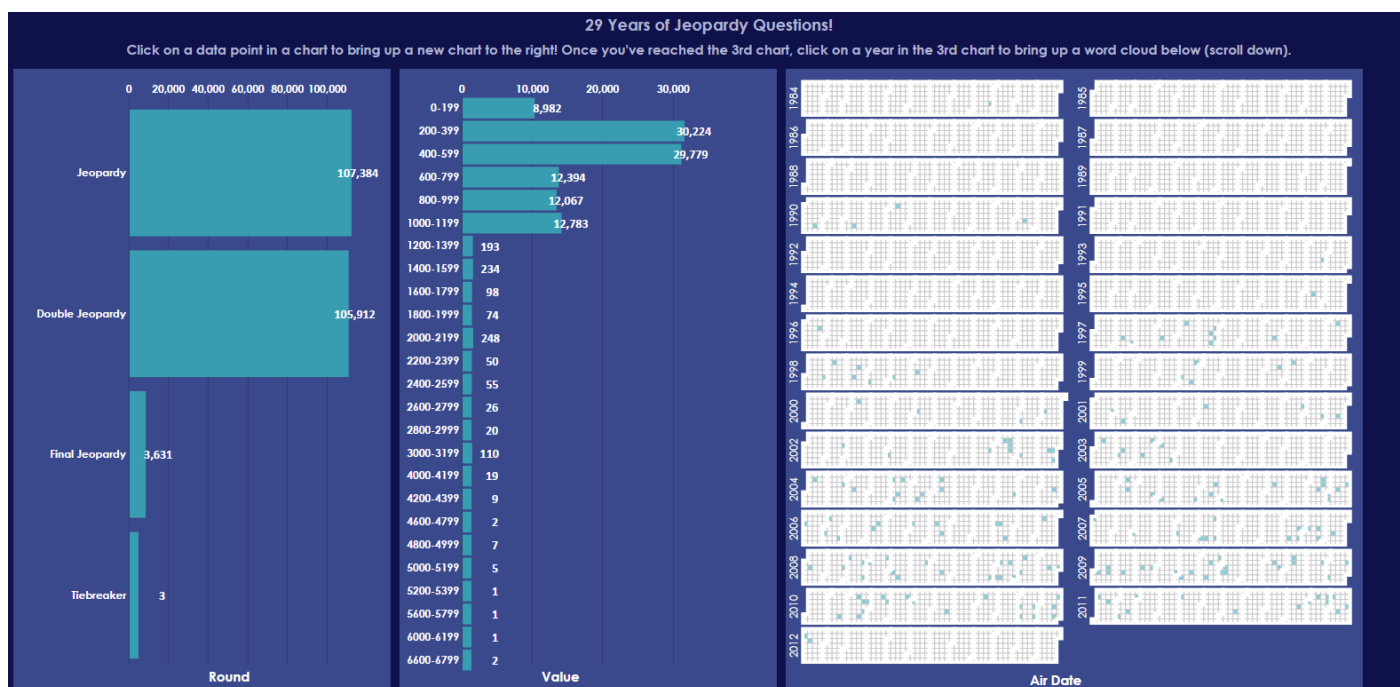
The top image shows the directions/explanation for the web application, and the bottom image shows the loading image for the page.

Updating Color Scheme

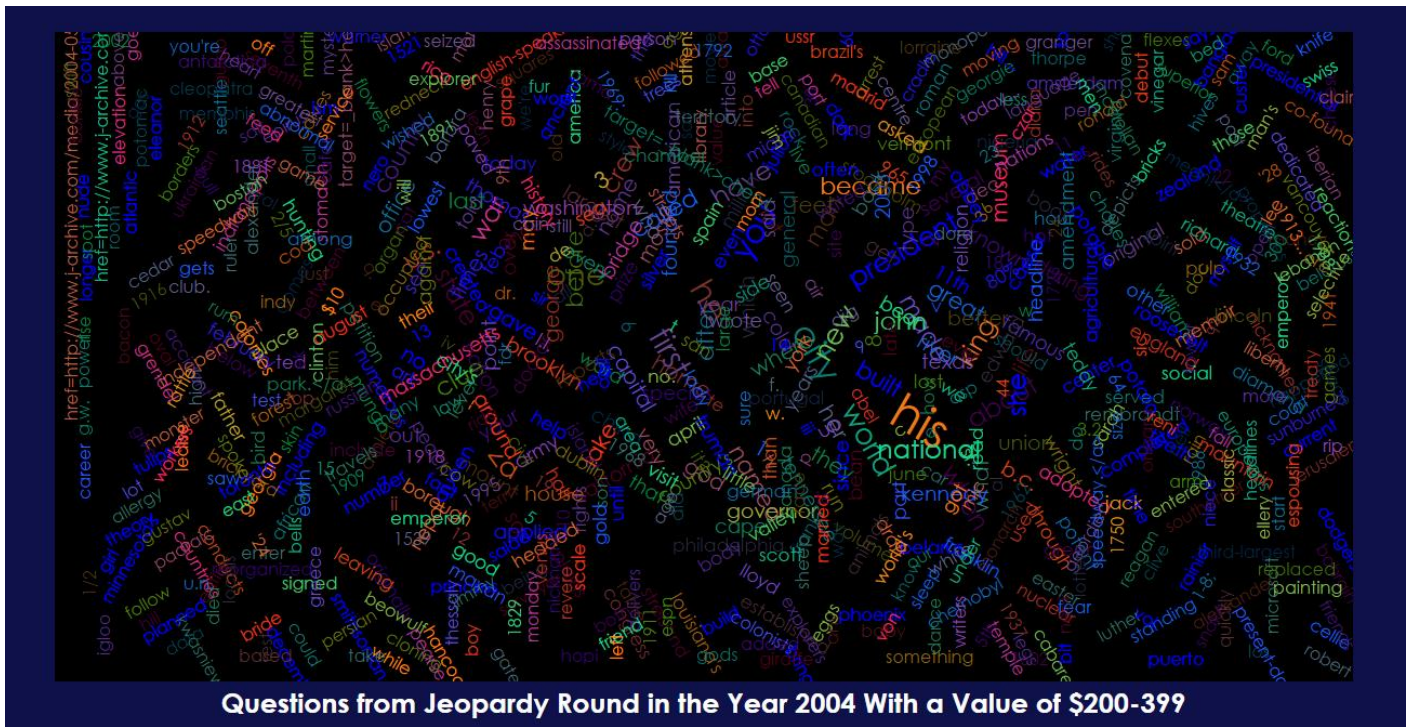
Design Evolution: Once I had developed the basic features for this application, I wanted to tackle the color scheme. While I liked the blueish color I had chosen previously, I felt that the color scheme was... boring. I decided to do a Google search for "Jeopardy" and used my "Colorzilla" eyedropper extension in Chrome to pick colors out of images from the Jeopardy show/logo. I then used those hex colors in my color scheme.



This is the "Colorzilla" eyedropper application, which was a very useful tool in developing this project.



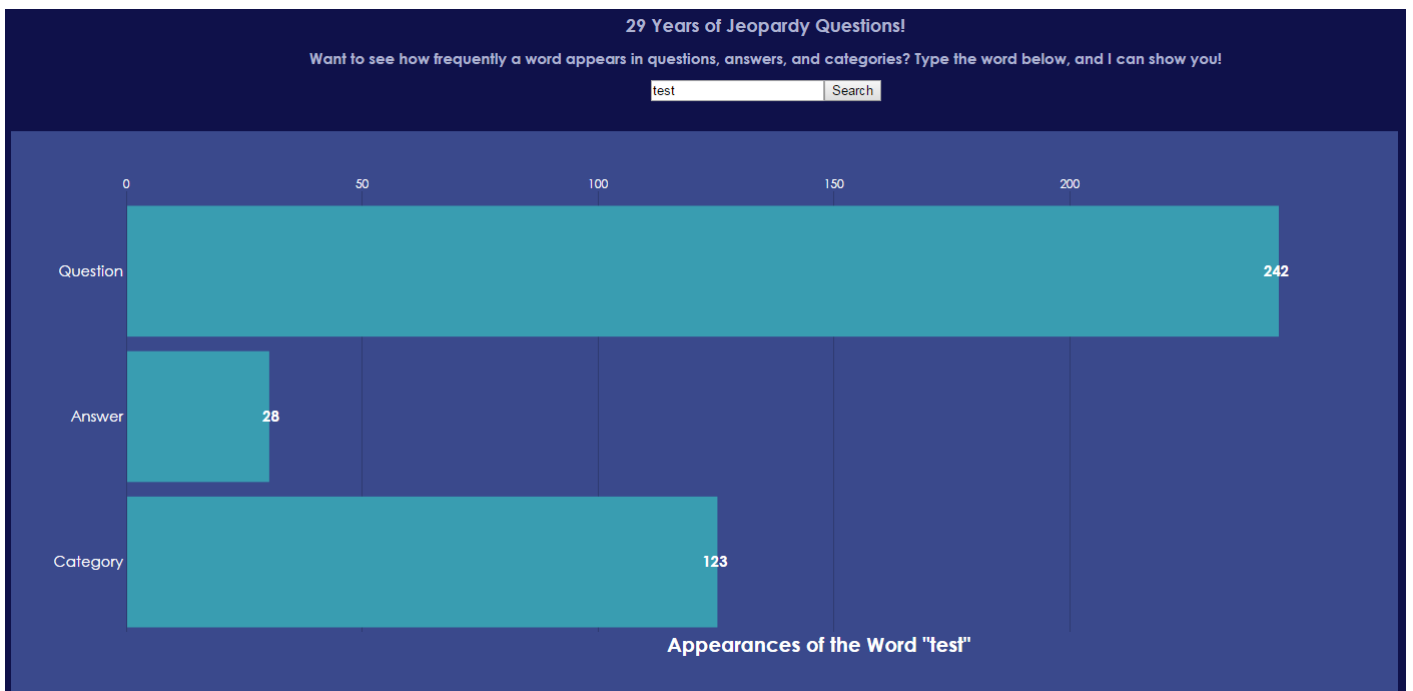
Shown above is my updated color scheme, meant to be closer to Jeopardy's actual color scheme.



The image above shows my updated color scheme for the Word Cloud.

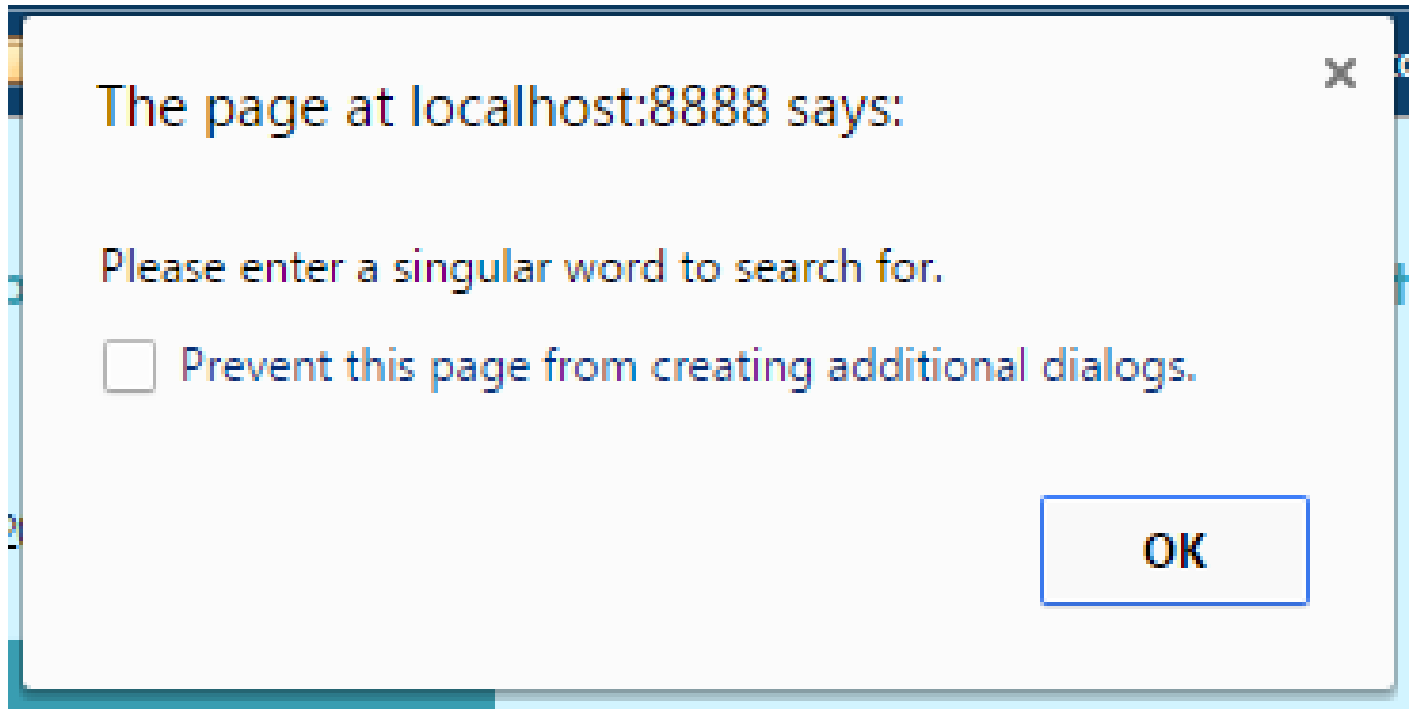
Extra Feature: Search for a Word

Implementation: 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.



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:



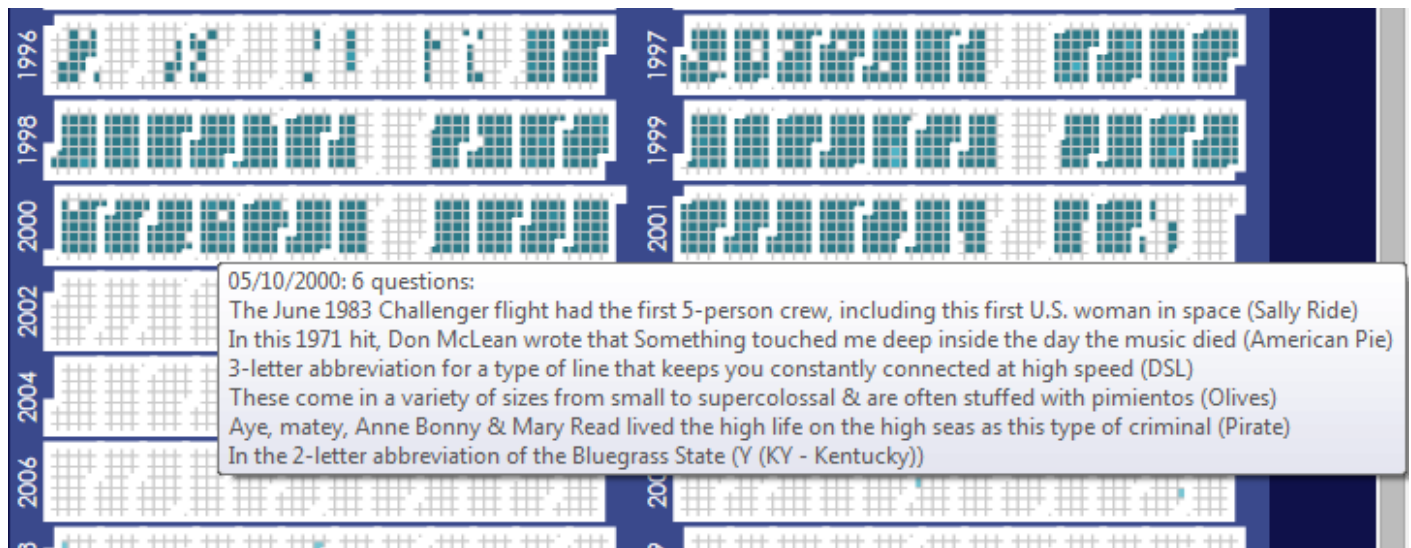
The error message that is shown when a user tries to search for what I've deemed to be "invalid" input.

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.

Adding Content for Tooltips

Data: While I find the two bar graphs and the calendar chart to be interesting, even with the addition of the Word Cloud I felt as though I wasn't using the actual question and answer data enough. Not having the questions and answers available for reading made it feel as though I had "detached" these attributes from the other attributes (round, value, and date), which just didn't feel "right." As I mentioned before, I feel as though much of the value of this project is in the content of the questions and answers, and not using them enough made me unsatisfied with the project as it was.

Because of this, I chose to create a tooltip (for now, just the standard text box that appears when you hover over something in a browser) that contained all questions and answers for any specific date (MM/DD/YYYY) in the Calendar Chart. The tooltip contains the date, the number of questions from that round -> value -> date drilldown, and then the content of each question and answer. This required writing a very short helper function that would take in an array of data points and construct a string with each question and answer from that array. Adding in this feature lets the user see the actual content of the questions and answers. The image below depicts this feature.



It's not pretty yet, but the content is there, and that is what I was concerned about at this point in time.

Getting More Use Out Of Word Cloud

In thinking of ways to improve this project, I realized that I wasn't getting as much as I was hoping to get out of the word cloud. The number of data points that were used to construct the word cloud was smaller than I'd liked, and this left the user seeing semi-boring visualizations. I wanted there to be dramatic differences in word counts (and therefore font sizes) of the words displayed. I then decided to change the functionality of the project so that if you hover over any bars in the bar graphs, the next graph (bar or calendar) is generated, and if you CLICK either of the bar graphs, a word cloud is made using that bar's data.



This is closer to what I was hoping for.

One noticeable issue with this newly-added functionality is its speed. Going through all of these data points (and there were only around 3,600 of them) took a minute or two to complete. This left the user waiting longer than I'd like them to have to wait to see their data. I may either put a size limit on word clouds, or I may try to preprocess the data and create counts of words ahead of time (for example, count the frequency of all words from Jeopardy rounds, from Final Jeopardy rounds, from

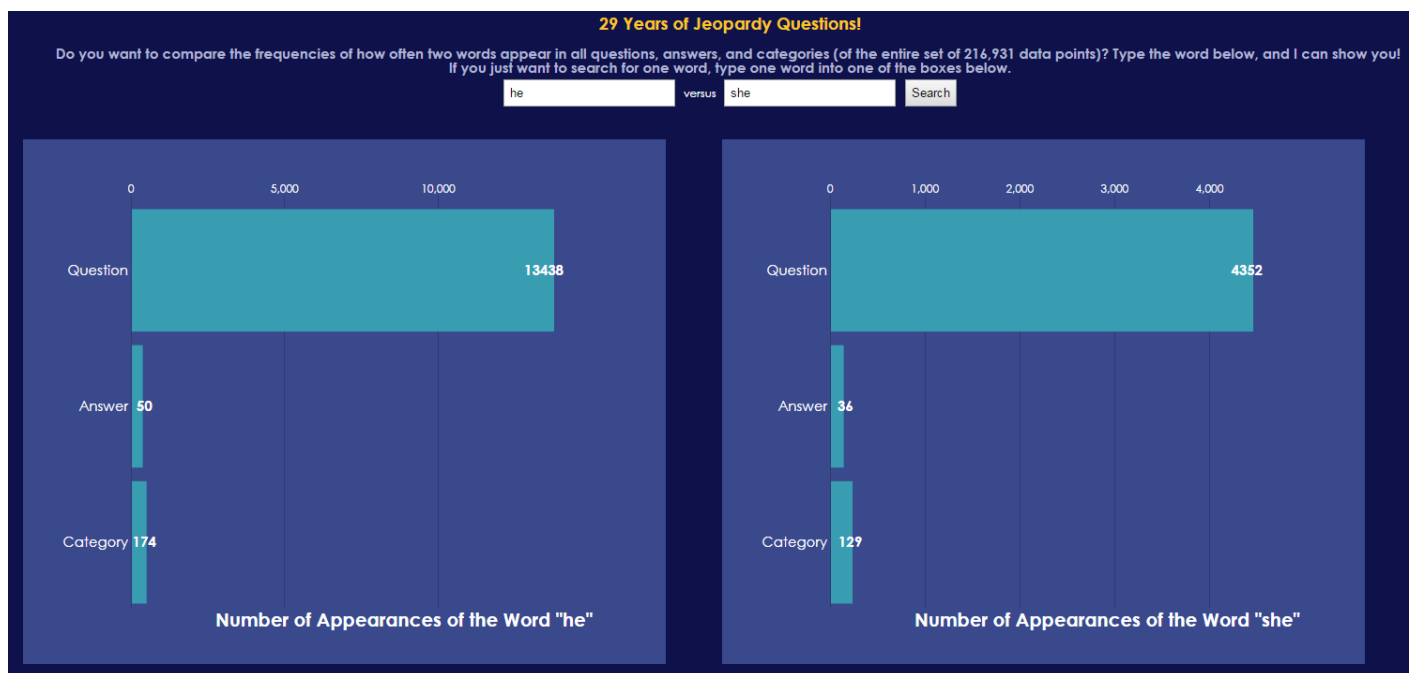
Double Jeopardy rounds, and from Tiebreaker rounds). I could do this for every data point in the set (for both bar graphs), which would completely eliminate the time needed to construct the mapping of words and their counts.

Feedback from Project Presentation

Design Evolution and Implementation: The feedback that I received from the project presentations in class last night (12/1) was wonderful. The suggestions I received were:

- Let the user compare the frequencies of two words, so allow for side-by-side searching.
- Show trends over time, meaning show how often words appear in questions as time has passed.

The first suggestion was made after someone asked me what interesting things I noticed in the data, and I noted how the word "he" appears in many more questions than the word "she." This lead to the idea of comparing side by side, in addition to searching for just one word. I really liked both of these suggestions and have already implemented one (you can still search for just one word, but now you have the option to compare the results for two words as well):



You can now see how often a word appears in the data set and compare it to another word's number of appearances.

I plan on implementing the second suggested feature as soon as I can.

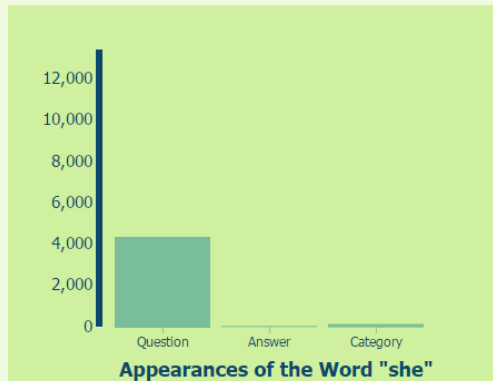
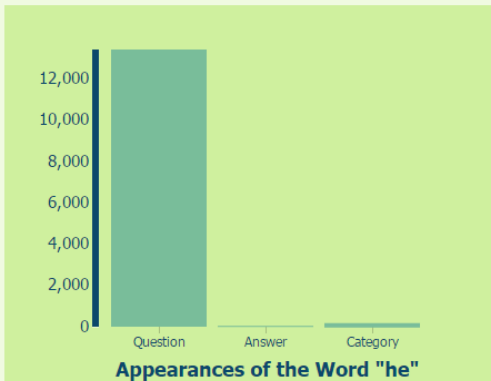
Changing the Focus to Comparisons and Time

Design Evolution and Implementation: It was suggested to me to try to focus the project more on comparing the appearances of words (comparing the number of times one word appears versus another), and then also showing a calendar chart of when those appearances took place. This lets the user gain an insight into what was going on in the world at different points in time: What were some of the news headlines? What was the "gossip"? What events were currently unfolding and worthy of being mentioned on a show like Jeopardy?

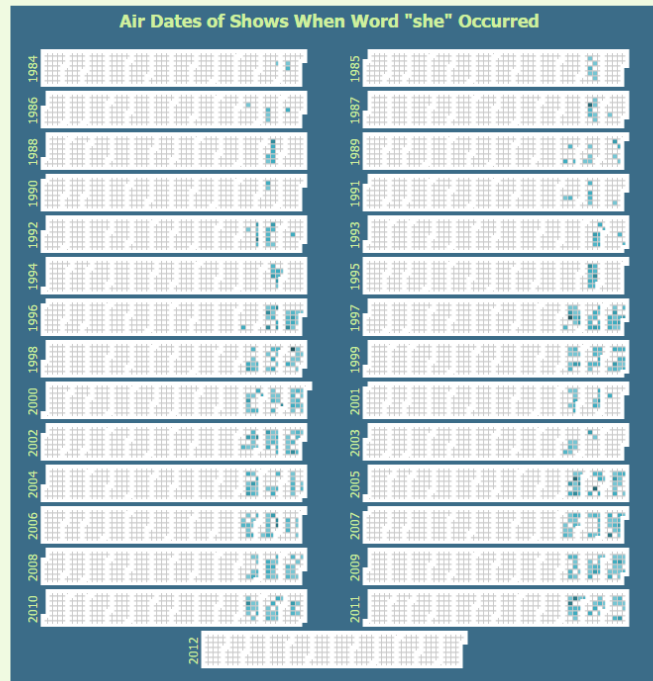
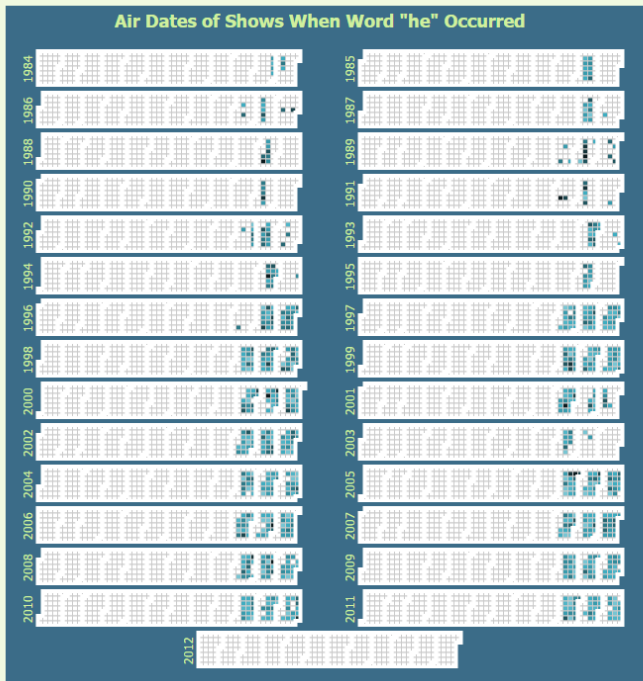
29 Years of Jeopardy Questions!

Type two words below to compare the frequencies of how often two words appear in all questions, answers, and categories (of the entire set of 216,931 data points)! If you just want to search for one word, type one word into one of the boxes below.

he versus she Search



At the top of the screen you can type in two words. You are then shown two bar graphs with these words' frequencies, and calendar charts for the words are shown below.



Here we can see the frequencies of the two words we searched for, over time.

Prettier and More Detailed ToolTips

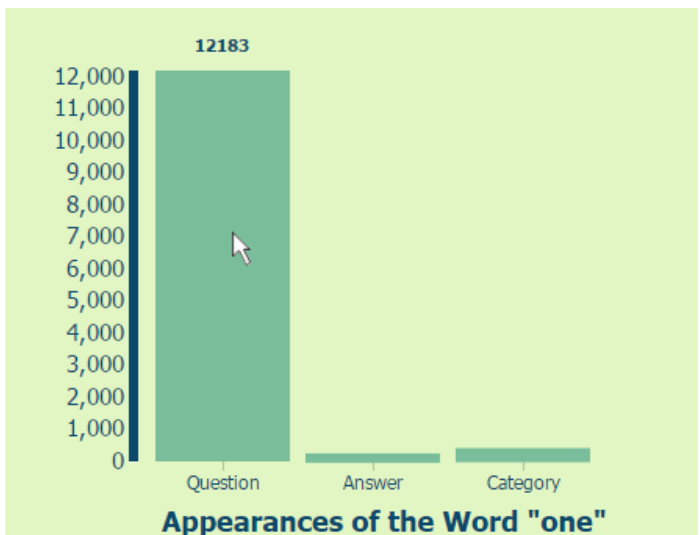
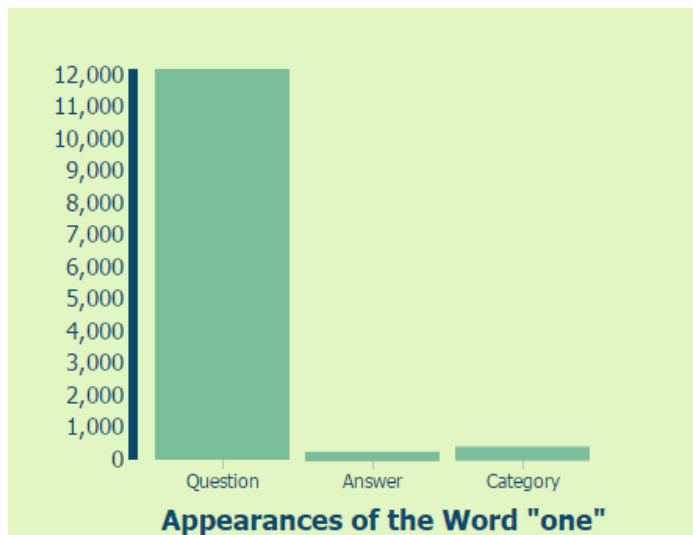
Design Evolution and Implementation and Data: Previously these calendar charts only let the user see the number of times a word occurred on a particular day. I rewrote my code to (on mouseover) display the questions/categories/answers that this word was found in, which I feel improves the project by giving the user more access to the data. In seeing what questions a certain word was found in, one can get a better understand of what was going on in the world at that point in time.



The tooltip shows the date, the number of questions, and finally lists each question.

Extra Interactivity: Displaying Actual Numbers in Bar Chart

Design Evolution and Implementation and Data: Originally I wanted my bar charts to always display the actual values that the bars are, however this felt like too much was shown on the graph at one time. I decided to use the tool tip library I used for the calendar view to make a very small, simple tool tip for the bar charts that would show the actual value of the bar when you hover over it.



The left image shows before hovering, and the right shows what happens while hovering.