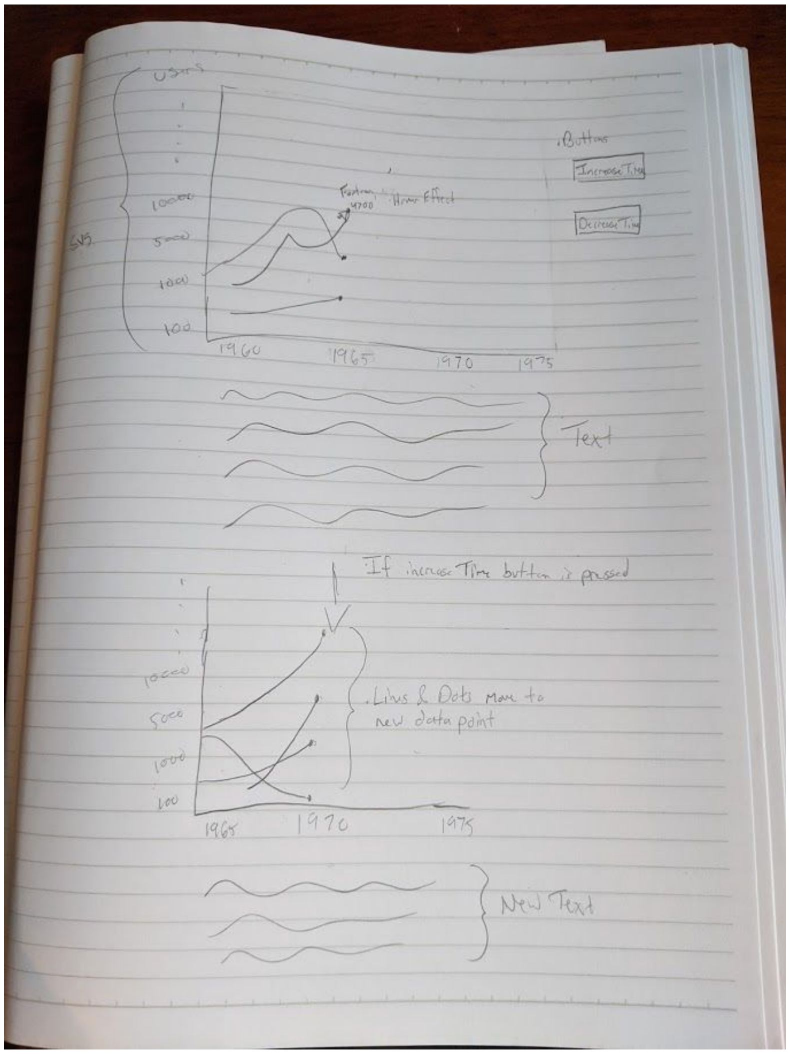
Matt Blanco
Final Project
ARTG 2260
04/19/2020

1. My project is a data visualization of the popular programming languages used from 1965 to 2019. There are buttons that allow the user to increment or decrement the years and the data will animate to reflect the changes of the time. The JavaScript adds the functionality and the D3 script for drawing the graph and data, the HTML has the skeleton of the application with all the text, titles, and tags, and the CSS styles the whole application while creating the style for the timeline. As for other assets a dataset was used for the programming language data (a link of which can be found on the application).



2.
   a. Overall the application works almost exactly how I envisioned it. The whole project came together very well and several of the problems I knew I would run into were solved fairly easily. I did deviate from my original vision by having the timeline not change, only having broad information from the decade, and how I displayed each piece of data was modified from how I originally planned.

3. Overall the application is very smooth and works well. Animations are elegant with a fairly appealing aesthetic. The overlapping text of the graph's data can make it difficult to read and sometimes a piece of data does not have a color associated with it, only a black circle.
4. Yes, I believe the general code construction of my project makes sense. A sequential pattern was followed starting with variable definitions, SVG initialization, then onto button presses and animations. The code patterns used were also helpful allowing for the project to be more efficient.
5. I'll be reflecting on how I created the paths (line 138) for each programming language each year. Since we didn't discuss paths in class I had to research this on my own and do my best to understand it. I opted for a function that manually created the string path even though I learned of the lineDraw() function that can be used to just input data and have the string created automatically. Personally I found it easier to understand creating the path string manually which is why I did not change it.
6. Using the same pattern as the previous question I could implement the lineDraw() function which would make the creation of each path more efficient and easier to understand for someone who doesn't know about SVG paths. A function would still have to be made so there would be about the same amount of code.
7. There are several times either in creating the various SVG elements to animating each element where a similar piece of code is reused. Perhaps these duplicate pieces of code could be put inside a function allowing the code to be more efficient where the same part isn't copied several times. This method would be better since again the code would be less clunky. Although in a smaller scale project like this one, performance isn't terribly affected by this.
8. At first I was worried how I would animate the changing axis although that was one of the first hurdles I overcame when it came to animation. What I didn't know was how similar it was to animating any other SVG element.
9. There are a few things that I believe can be improved about my project. First off the general code layout. For some reason it feels clunky when I look at it, and I'm not sure why but that probably means it can be improved in some way. That would just take more practice with JavaScript to fix. I could also improve how I display the name of each programming language and data. Currently lots of the data is close together leading to overlapping text making parts of the graph difficult to read. I would need to change how I display the data to fix this problem.
10. Originally I wanted to have the timeline animate with each decade; however, I realized this (while not too complex) would simply take a while to implement and have a lot of moving parts so I decided to scrap the idea with what I currently have. I also tried several methods of displaying the programming language name and respective data with a mouse hover. Although when comparing the mouse position to the position of SVG elements they were not the same so the hover effect displayed incorrect data. Problems like these simply take time and practice with JavaScript and D3 to learn what is and isn't possible.