Jacob Wronski
Coding the Law
Client Letter
December 10, 2023

For my final Coding the Law project, I have been developing a general-use data scraping bookmarklet. The project has taken several turns since its origin; however, the end result is satisfying — not only because it works but also because I learned how it works.

One of the many ideas for the final project was to create a tool that could comb a webpage and extract all the names, downloading them to a CSV so I could create an alumni database from mastheads online. After discussing the idea with Professor Colarusso, we decided to expand the functionality of a pre-existing REGEX to CSV bookmarklet. The goal was to take the already existing code, which required the user to manually input REGEX strings, groups, and delimiters, and partially automate the population of those fields by creating a drop-down selection menu with patterns that had pre-assigned values. Essentially, the final product would be a simple data-scraping tool that is accessible to all legal practitioners, regardless of technical prowess.

Serving as my own client, I sought input and advice from Professor Colarusso, as well as Chat GPT and Github Copilot. Since I had very limited experience with JavaScript, I started my process by familiarizing myself with the code as was written by asking Chat GPT to explain it line-by-line and function-by-function. The process was not smooth, and I faced several blockers, particularly while trying to format the data correctly into the CSV. One of the goals set by Professor Colarusso was to get the code to separate different capturing groups, thereby increasing the functionality of the tool. Ensuring the REGEX string and capturing groups correctly split into different columns required careful experimentation. The final product successfully separates out the different capturing groups, allowing for a finer level of data extraction. This is particularly evident in the Federal Case Name pattern, which can successfully match and extract federal case names and then separate them in the CSV so the first column shows the entire name, and columns after that are specific captured groups, like which circuit court and year.

Two very small but impactful features I added to the code were not initially part of the plan but were implemented while I was testing the tool. With the lofty goal of aiding in legal research by finding and organizing certain data points, the tool must have some degree of organization that will allow the user to further refine the data. Each time the code was adjusted, I had to recreate the bookmarklet and try it out. This resulted in my downloads box becoming inundated with the generated CSVs. I realized that "file.csv(1)-file.csv(850)" did not provide much organization to help the user track their data. First, I implemented a dynamic filename, which would automatically name the CSV after the pattern used and the date it was generated. Next, I made the bookmarklet log the webpage that was scraped and add it to the bottom of the CSV after all matches. While far from a perfect solution, these two additions allow the user to keep track of their data and even have a back reference if they needed to refind the source they used.

The final product performs well in a controlled environment. I created a test HTML with different variations of the data that represented the different ways people might write out phone numbers, emails, case names, and statutes. This allowed for easy testing of the string and code variations, without worrying about different style changes or other issues persistent across webpages. The code executes well on various "live" websites; however, its consistency may be improved by overriding a webpage's style or first cleaning the HTML into an RTF format before running the REGEX strings. Either of these additions would increase the bookmarklet’s consistency and reduce the number of times it matches HTML text or fails to recognize the text at all.