Stories

Step 1

* Building first project with html2canvas
* Switching from locally hosted to Heroku javascript bug in URL (and the reverse)
* Making sure id being passed around remains a number
* Float versus flex box
* Using a reset.css
* Linking files properly to front end JS files
* Importance of arrow functions and higher order methods to simplify code
* Remembering return values of methods
* Importance of console.logs and testing
* Being thorough with reading and understanding documentation
* Necessity of mental motivation and persistence

Step 2

For my first project I chose to use html2canvas for the download feature. I had never attempted to have downloadable content before, and html2canvas is advertised as a really simple and easy to use module for creating faux screenshots than can then be downloaded. Although easy to install and write the initial function, I found out that html2canvas will not export images if the source is outside the origin. The documentation recommends a proxy server. However, I discovered that simply changing the “allow taint” attribute from the default setting of false to true corrected the problem.

One of my greatest lessons in bug fixing is when the axiom of never assuming a block of code is correct came full circle. My first attempt at an interactive list with delete buttons was a challenge to get working, but when it finally began working, I noticed that a button was being added to the end of my list. Each time I deleted an item, a new button would appear at the bottom of my list. I spent hours working through and rewriting the axios calls and my server file. I even went line by line through my css file. I struggled with css, was only starting to get the hang of JavaScript, so I assumed the error was there. My html file was just a basic template with a few lines added. I was very comfortable with html at the time, so for hours I assumed my JavaScript was the problem. Finally, I just began playing with my list and observing the behavior. I had an aha moment where I realized that the JavaScript was not adding an extra button. I had placed my list and its corresponding delete button inside of my form and had an event listener on the entire form instead of just the submit button. So, it was not an extra button at all, but the form adding a list item with an empty string attached to it. I fully understood in that moment why its so dangerous to assume that any piece of code in a nonfunctional app is right. The code I was most comfortable with and most sure of had housed the problem the entire time.

An especially important nontechnical concept I learned during my bootcamp time is the importance of feeding yourself positive encouragement. The very first time I had to write an app that passed information from the front end to the back end and back again I couldn’t seem to get anything to work. I poured through notes and old code, but kept hitting a brick wall with each version of code. The devil on my shoulder was telling me that I wasn’t understanding and wasn’t cut out for this and after an hour of not getting a single item to display, I wanted to cry and had to step away. It was then I remembered the words of Ford when he said, “Whether you think you can or you can’t, either way you are right.” I looked back on myself and thought, “I didn’t even know what axios was a few weeks ago. I’m not behind, I’m growing.” I took a deep breath, went back to my computer and wrote the words “Make it work” into my code and began collecting other motivational sayings and adding them in too. Step by step I attempted to track in my mind what information was being passed and to what. Once I had my mind back on my side, inside of the next 30 minutes I had my first axios call working that I had written entirely on my own. I wasn’t unfamiliar with self-talk before then, but I got my first experience with just how powerful it can be. If something is difficult, self-talk can be the difference between crossing the finish line and not.

Step 3

I love creating picture quotes on social media. So, for my first project I decided to make an app that would streamline the process by allowing user to input and image and some text and then generating a picture quote without the need for photo editing software. Because I only had roughly one day to get the project up and running, I needed a simple way to add upload and download functionality to my basic vanilla JS web app. I wound up choosing the html2canvas module to implement the download feature. I had never attempted to have downloadable content before, and html2canvas is advertised as a simple and easy to use module for creating faux screenshots than can then be downloaded. Although easy to install and write the initial function, I found out the hard way that html2canvas will not export images if the source is outside the origin. However, I discovered that simply changing the “allow taint” attribute from the default configuration settings from false to true corrected the problem.

One of my greatest lessons in fixing bugs was when the axiom of never assuming a block of code is correct came full circle. After learning about axios and DOM manipulation, I set out to create my own interactive list that included corresponding delete buttons for each item in the list. It was a challenge to get the delete buttons working, but when they finally did I noticed that when the list item was removed, a button was also being added to the end of my list. Each time I deleted an item, a new button would appear at the bottom of my list. I spent hours working through and rewriting the axios calls and my server file. I even went line by line through my css file. At the time, I struggled with CSS, was only starting to get the hang of JavaScript, so I assumed the error was there. My html file was just a basic template with a few lines added. I was very comfortable with html at the time, so for hours I assumed my JavaScript was the problem. Finally, in frustration, I stopped coding and just began playing with my list and observing the behavior. I had an aha moment where I realized that the JavaScript was not adding an extra button. I had placed my list and its corresponding delete button inside of my form within the html file, and had an event listener on the entire form instead of targeting the submit button. So, my JavaScript was not an extra button at all, but the form was adding a list item with an empty string attached to it each time I tried to delete something. I fully understood in that moment why it’s so dangerous to assume that any piece of code in a nonfunctional app is right. The code I was most comfortable with and most sure of had housed the problem the entire time.