Personal Reflection

1. Before I finished foundations the tech field seemed like an incredibly advanced and complicated system that took years of study and dedication to even begin to understand. Now, I have a slightly differently look at it. While the field is vast and deep, understanding the basics are not as difficult as I had expected. Before Devmountain I felt like what I was learning was always a tool that was deprecated in the tech field, and I was just using it as a steppingstone. Here I feel like I finally learned useful and relevant information to the real tech field, and I can actually start contributing even if it's just a beginning.

2. CSS was by far the hardest thing I've had to do. I know it's a broad topic with multiple answers, but I still wish there was more emphasis placed on teaching it. I really feel like I learned every part of that and html on my own and honestly, I learned bad habits that I'm still trying to break. My feedback would be to rework the lectures to communicate the importance of good structure and organized html/css. It's very hard for me to learn when there are a seemingly infinite number of ways to structure and do something and I have no basis for what a good way and bad way is.

3. I learn by re-watching lectures multiple times at 2x speed. I lose focus and attention if it's at normal speed but having the other foundation recordings available helped immensely. I could pause and re-watch parts I didn't understand. I watched every single lecture twice, once with foundations 1 and then again, the next day with my own class. It helped me review the content and come up with questions beforehand. A cool example of how this worked was with the Jest test driven development lectures. Jest was somewhat complicated, but the assignments recognized that and were very much simplified. It's something we could say we experienced, but not necessarily something that we fully needed to grasp. Because I watched those lectures and did the assignments twice, I can honestly say I have a good grasp on jest, and I have applied some of it to my own projects already. I see the benefit and I could tell an employer about it in some depth.

4. As soon as I have access to the content (which I really wish I had before specializations started) I will get ahead and watch and do everything twice just like before. I'll search the documentation for questions and record every note to refer to later.

5. Pair programming is good, but not flawless. There were a couple times that I came a little unprepared and immediately I struggled to comprehend the assignment and get past the social aspect. It's hard to have no knowledge of something and then understand it while at the same time talking to someone. What did help me were 2 things. First, if I knew the subject well, teaching it was very beneficially for me and helped solidify my understanding along with opening my eyes to misunderstandings

I had about the content. Second, after I had tried my best at something and had time to document my questions and failures, it was also beneficially to then talk to someone who may have the answer or could point me in the right direction. Bottom line, pair programming works as long as both parties understand and have read through the assignment and know what they need to accomplish so they aren't wasting time.

6. I loved the diversity of what we were doing. Originally, I was afraid this would be 8 solid hours of lecture and then a homework assignment every day. It's not like that at all. We have a warm-up, a good solid concise lecture and then hands on practical practice. I enjoy the layout. My favorite labs were the deeper javaScript ones. Coding like that is something I could do everyday for the rest of my life. I love the logic puzzles so so so much more than css.