So what is the super secret strategy Launch Academy employs to make their graduates employable? Test Drive Development. I say super secret because it is something that wasn't talked about much, or really at all during the recruitment process and wasn't presented at all in the pre work either. Secret or omission, it’s here now. From a personal standpoint, I find it rather tedious. From a getting a job standpoint, I see it as massively valuable.

Before diving into the murky waters of TDD (sounds like a disease, right?) a quick rehash of what was introduced to our eager cohort this week. Object Oriented Programming (OOP) was the big topic at the start off the week. This was something that I had studied a fair amount in the pre-work and found it pretty straightforward, although to think it's straightforward likely means I know little to nothing. The use of classes feels a little closer to building real things and, as that's the ultimate goal, this was a welcome topic.

The real fun is that learning about classes and objects further confused me about Ruby's motto, 'everything is an object.' An instance of a class is an object (coming with it's own object ID!), but if everything is an object, why are there specific objects, and why are these objects objectively objectified obscurely, GAH! The word makes no sense to me anymore. Fun fact: in javascript a hash is called an object.

Now onto the serious business of TDD. I feel there are some in the cohort that find it an unnecessary hindrance. I'm leaning that way, not necessarily because it's a hindrance, but because it's just one more thing to get wrong when trying to finish a program. Also, because we're learning the strategy, we're implementing it on code that is oversimplified. If I have a method that multiplies a number by two, it seems rather frivolous to test that if I put 5 into that method it had better return 10. Yes it's more work, yes it's not as fun as writing the actual program, yes it more difficult than it should be.

So what's good about TDD? Well, I've been programming for 3 weeks. That is an astonishingly unremarkable amount of time. In 10 weeks I will have more than tripled that time, to an equally unremarkable span. Now, I may be the greatest programmer of all time (I'm the greatest programmer of all time), but I am still extremely green, and doing TDD means you're building code that your prospective employer or client can trust. I may be great at testing my code in my mind, or in my own way, but externally, I can't be trusted to write bug-free code with using testing. TDD is a way to put a stamp of approval on your work, and I can see why it will be important to master while moving forward into more complicated frameworks. TDD is broccoli, it's good for you, but very few people really like it.

One final note on TDD. It's not easy. Knowing what to test and how extensively is more of an art than a science, and it takes practice to make effective tests.

Next week is databases. My learning velocity is pretty solid right now, but I fear it will again be tested starting Monday. We've been instructed to take a full day off this weekend. So, that's a thing… that’s happening. Au revoir.