HOW DO I OCAML?!?!?

"How do I do well at Ocaml?"

This is a question that quite a few people have posed to me over the past few weeks, and understandably so. OCaml doesn't seem to be something that will just click, and from what I can tell, it could click at any time. I'll try and answer this question in stages.

"But it hasn't clicked, and I feel like it never will. What should I do?"

I still don't fully understand OCaml, but there are definitely a few things that have helped me on my way to really beginning to grasp the small bit of OCaml that we're learning as part of the module.

The most useful thing that any of you can do is this... Instead of sitting there like I think absolutely everyone did for the first few weeks saying "this is horrible, I hate OCaml, I give up", take a step back and think about it this way. You're producing an elegant and (hopefully) efficient solution to a problem that in many other languages, would take a much more complex solution to solve.

As well as this, practice really does make perfect. In your spare time (not all of it, of course), try to work your way through the 99 OCaml problems here:

`https://ocaml.org/learn/tutorials/99problems.html`

For the most part, a lot of these problems should be accessible to you, and as for the ones that currently aren't, work through ones that you know you can tackle, or at least have a good stab at, and then come back to the harder ones.

Finishing a program in OCaml seems like climbing a mo ntain - you get to the summit (or in this case, you sit in your chair and collapse from the exhausting mental trauma you put yourself through every friday night)... and then suddenly you realise that everything you just worked for has culminated itself into one final product. If you haven't cheated, you'll sit there and hopefully feel some sense of achievement (i know I do), and I think it's this that will really spur you on to developing your skills with OCaml.

Another tip I'd have is talking through the problem that you've been set with other people. Chat to your friends about it... hell, make new friends who are struggling too, and work through it together. Don't neccesarily sit there infront of your computer screen, gawping at it and doing nothing at all though - grab a pen and paper, or a whiteboard and split the problem down into smaller chunks.

If you feel like you've hit a brick wall, and your code isn't passing the tests, take the test case that your code failed on, and trace through it by hand. Work out **exactly** what your code is doing, and then ascertain why this is producing the wrong result. Keep doing this until your eyes bleed (or alternatively, until you solve the bugs)...

Talking it through with other people not only gives you both the chance to realise things that the others might not have, it also gives you a chance to undergo something called "Rubber Ducking". This is a legitimate technique, and the idea is that you explain what you're trying to do to someone else (or a rubber duck...), and in doing so, you'll realise exactly what you need to do, or what has gone wrong. I can't stress enough how worthwhile this is, even if it sounds stupid. Don't just sit there and say "I don't think this would help me" - try it out and maybe you'll surprise yourself.

Finally, the other day my friend introduced me to Richard Feynman's algorithm for problem solving. I'll say it in advance, when you first see this, you'll think I'm joking. Just give it a couple of minutes, and look back at the algorithm - maybe even try it out. It sounds obvious, but sometimes you have to just take a step back and assess exactly what you need to do.
Here it is,

1. **Write down the problem.**
2. **Think very hard.**
3. **Write down the answer.**

I hope this has been helpful, and if you have any questions about what I've said, just ask.