Dealing with Debugging

While it's tempting to ask others for help anytime you struggle with your code, it's actually a pretty bad idea. We love how you're working in teams, and we love teaching you, but we also want to empower you to use every tool at your disposal to become better, more self-reliant programmers. Therefore, we present you with a simple challenge: learn to debug your code.

#An Error?!

The first step in mastering debugging is to check in with yourself about **how you respond to errors**. You can get upset, or you can remember that computers only do what we ask them to do. An error, as much as it hurts to say it, is almost always the dev's own fault.

An average dev reviewing their code after an error message.

Rather than getting upset, **read the error calmly**. Put it into Google, or even just click on it. Eclipse usually shows you where things are going wrong, and you can usually trace an error all the way back to where it came from. Your goal should be to see an error and **think of the possibilities you have for solving it**, rather than believing that, somehow, your code should be infallible the first time you write it out.

And when you get another error right after solving the first–that's progress.

Not all progress is as easy as one and done. A new error is still progress!

So what tools do you have?

#Debugging in Eclipse

Rather than having a wall of text here, we're going to let John explain it to you. Follow along with his examples, maybe even make them happen in your own IDE, and see what Eclipse can do to make your life easier:

Open in new tab

https://youtu.be/aqcJsKdjjvU

Outside of Eclipse, you also have access to the code visualizer we've been using throughout the camp so far. There are plenty of sandbox IDEs here on Coding Rooms (the ones that are unscored) for you to play around in, too. Another good place to play around is W3schools. Search for the new topic you're learning, and play around with their examples.

Anytime you learn something new, test the limits of your understanding. Throw in new commands, new methods–why not? It can feel like this is all a big waste of time, but checking your assumptions is one of the best ways to learn how to code properly–especially since most errors are born out of our assumptions in the first place.

While we'd love to just pump your brain full of knowledge and ensure that it all stays there, that's just not how learning code works. Even seasoned devs will read documentation, throw in a few print statements, and just see what's going on from time to time. Learn to love this process and there won't be anything in the tech world that you can't master. Use the tools above, ask each other, ask us, but don't be afraid to experiment!