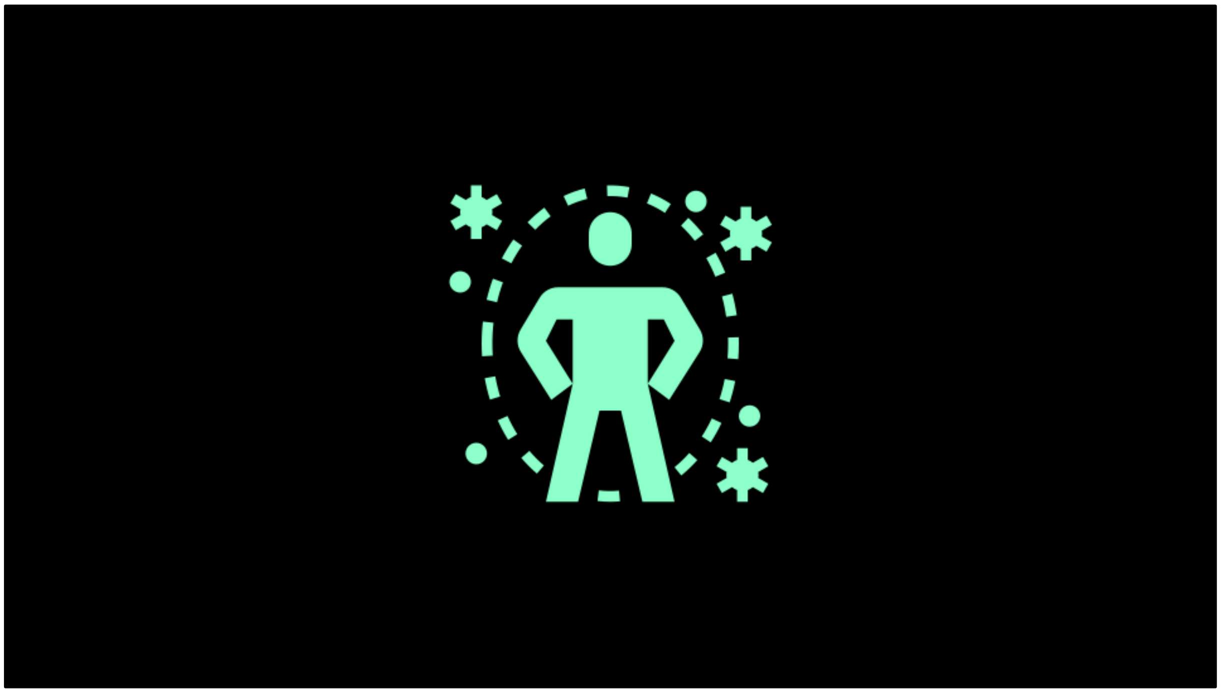
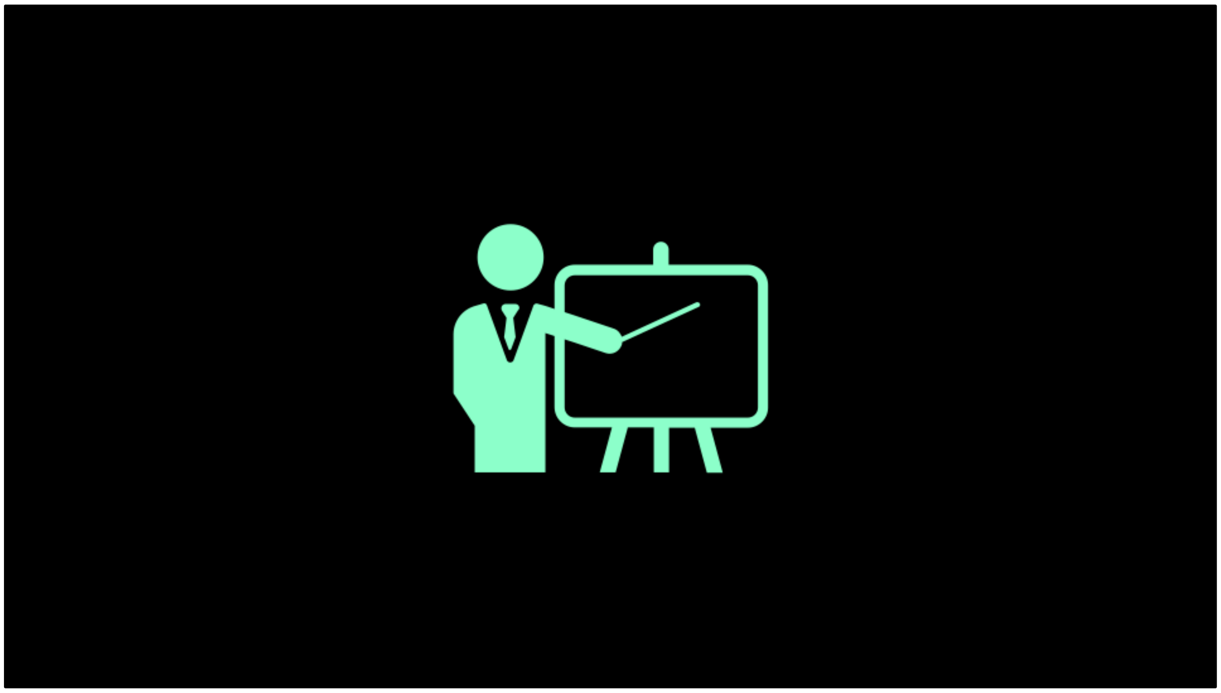
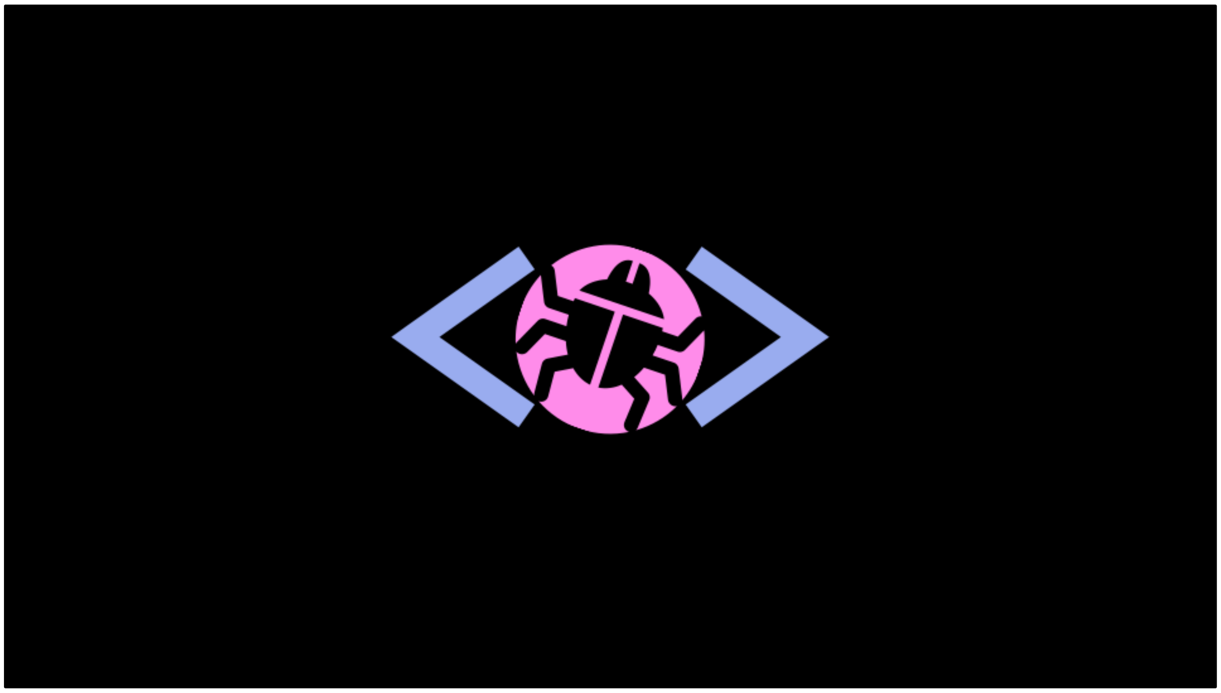


Show of hands. Who remembers something about our immune system from middle school?

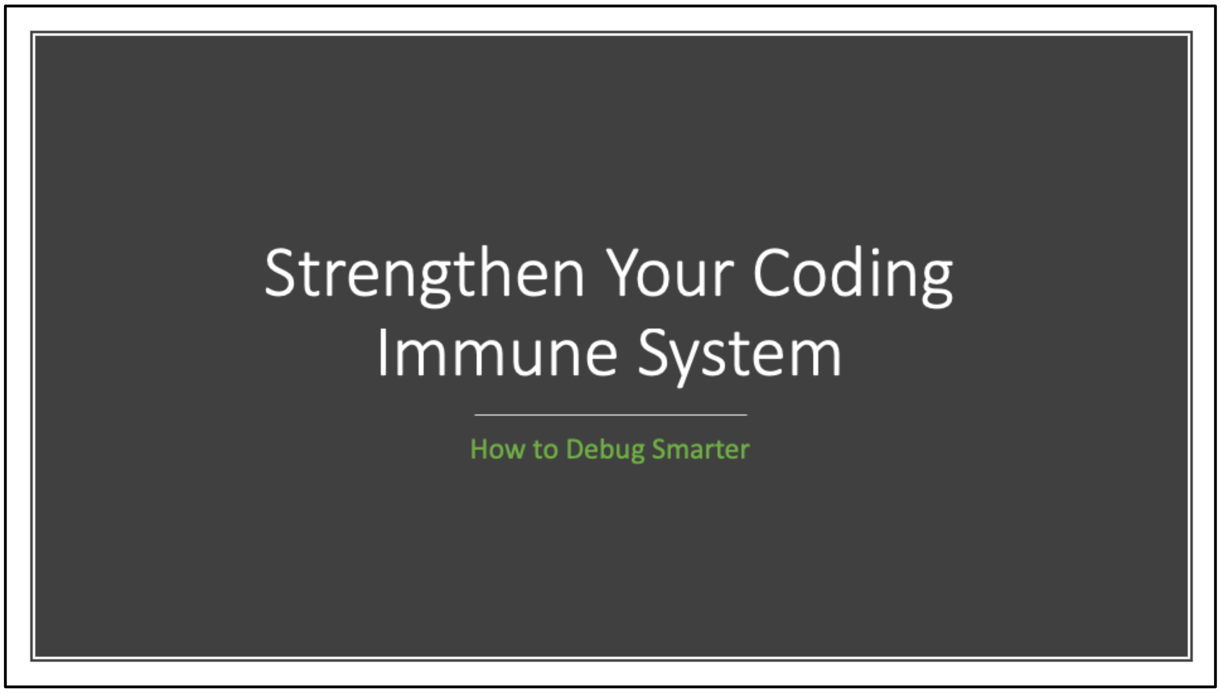
I found out, as I prepared for this speech, that our understanding of immune systems has advanced some since I was in the fifth grade, but don't worry, we're sticking to the basics



To protect us from illnesses, or bugs, our immune system needs to be trained. It does so by encountering bugs and building defenses against them.



It turns out, our code base also has an immune system, an entity that will find and remove bugs, but that entity needs to be trained, as well.



Strengthen Your Coding Immune System

How to Debug Smarter

Marlena Baker

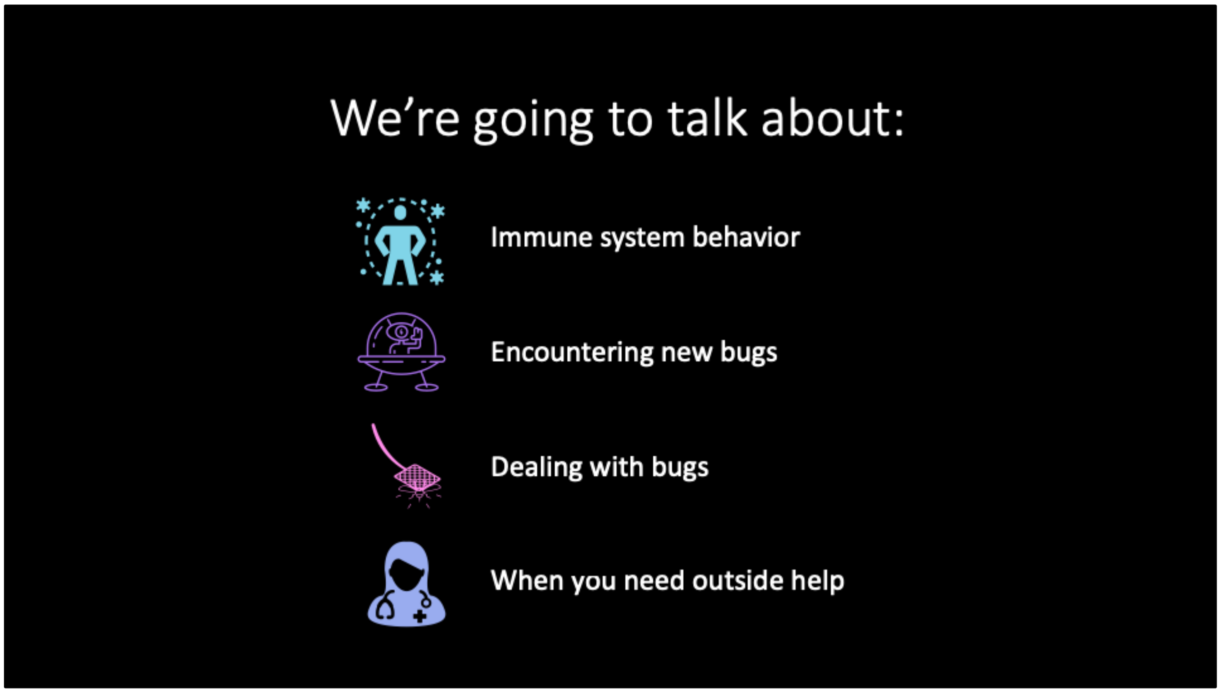
Front End Developer @ HomeAdvisor

Education Enthusiast

Snake and Cat Mom

Distractible Crafter

meetup.com/Denver-Vue-js-Meetup

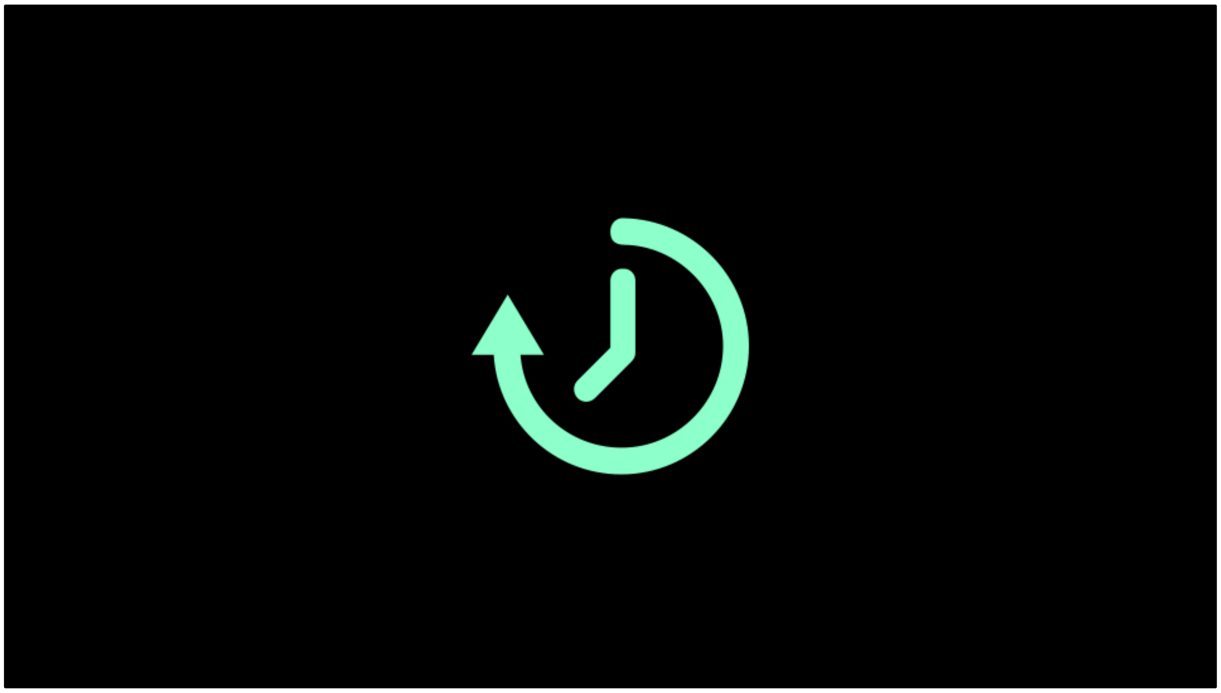


We’re going to talk about:

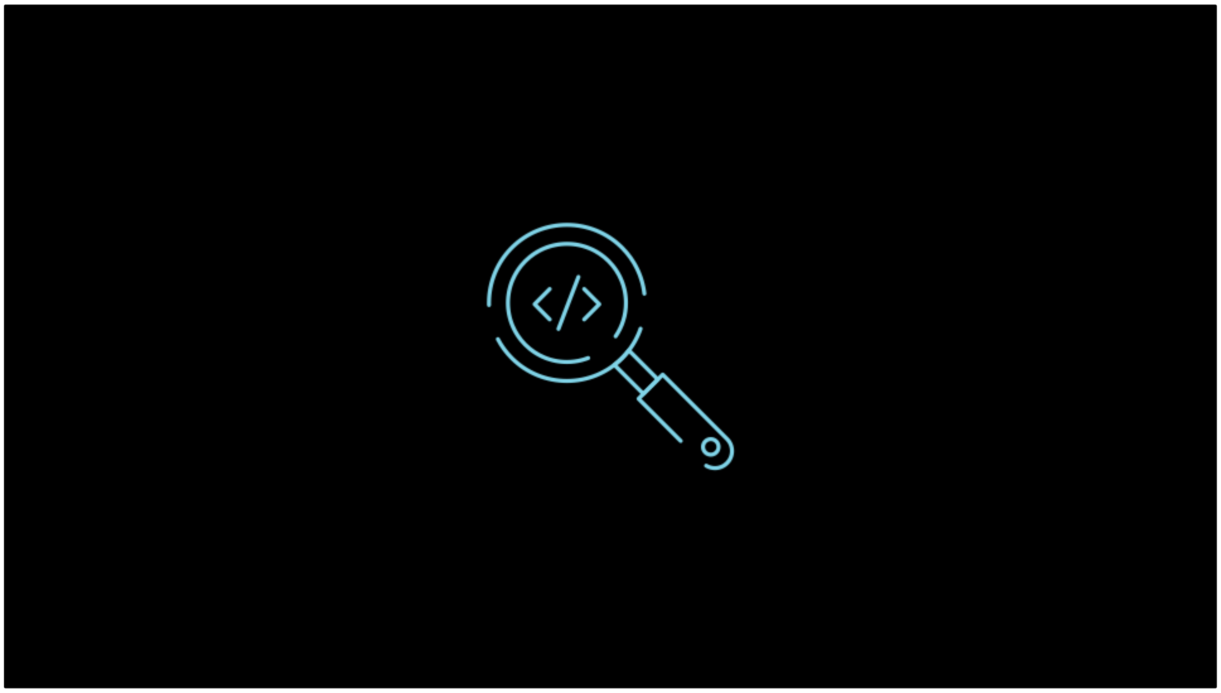
- Immune system behavior
- Encountering new bugs
- Dealing with bugs
- When you need outside help

IMMUNE SYSTEM BEHAVIOR

Fully functional immune system is decent at knowing when something isn’t supposed to be there

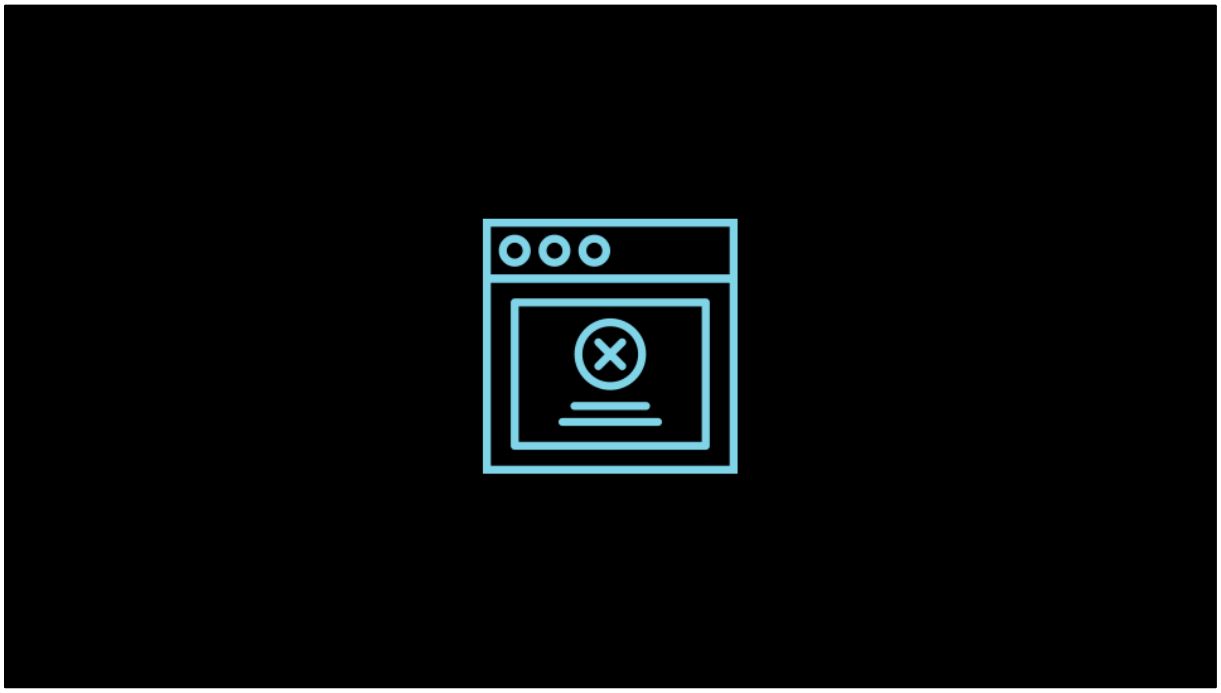


Learns the difference over time



First and foremost, you need to know your code. Even if you're helping debug someone else's code, take a moment and evaluate what it's supposed to be doing. Yes, it takes time. The better your understanding, the easier it will be to spot bugs.

Being able to evaluate code as you read it is a wildly valuable skill. Even if you aren't very familiar with the language, there's often enough similarities to give you some clues. And, of course, there's always the developer's best friend, Google.



Pay attention to the warnings your code gives you! They're red so that you notice them, not

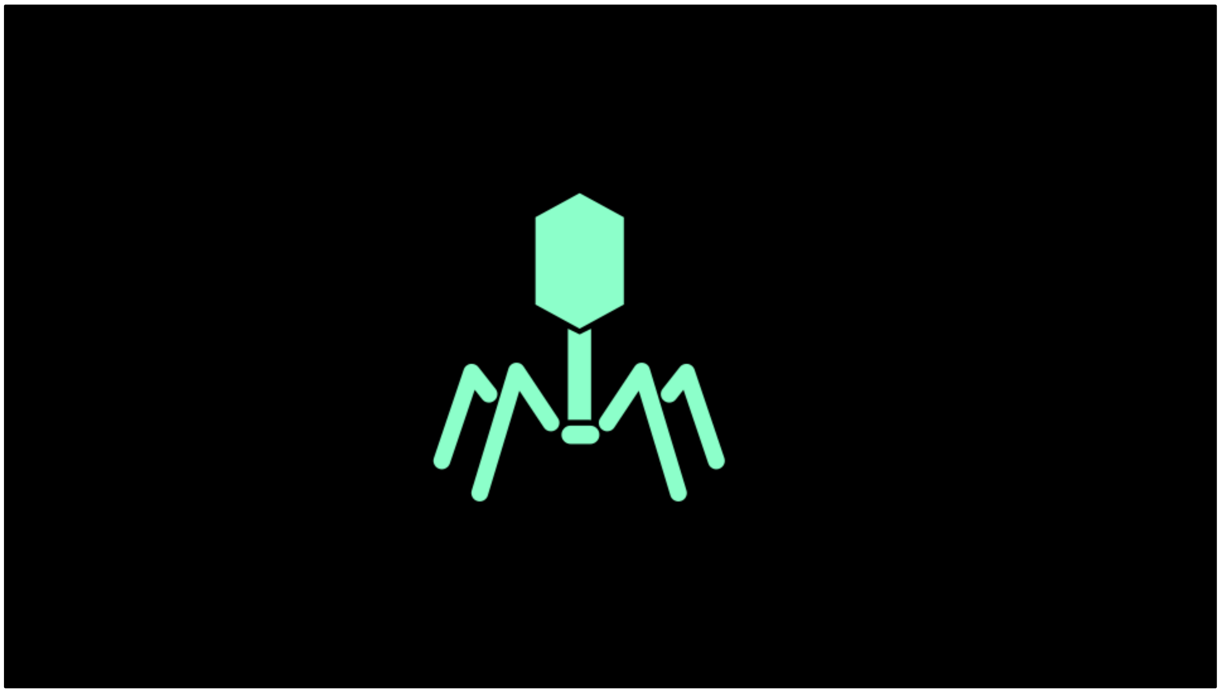
Errors are there to help you. They will give you useful information like what type of bug and where it might be? File and line, potentially a stack trace that will let you work your way to the problem area.

Strange behavior from the application

So many times I’ve thought that my code was ready to go, only to find that user text inputs aren’t being captured or my HTTP request isn’t firing. Code can fail ‘silently’, especially Javascript, so it’s very important to make sure it runs correctly.

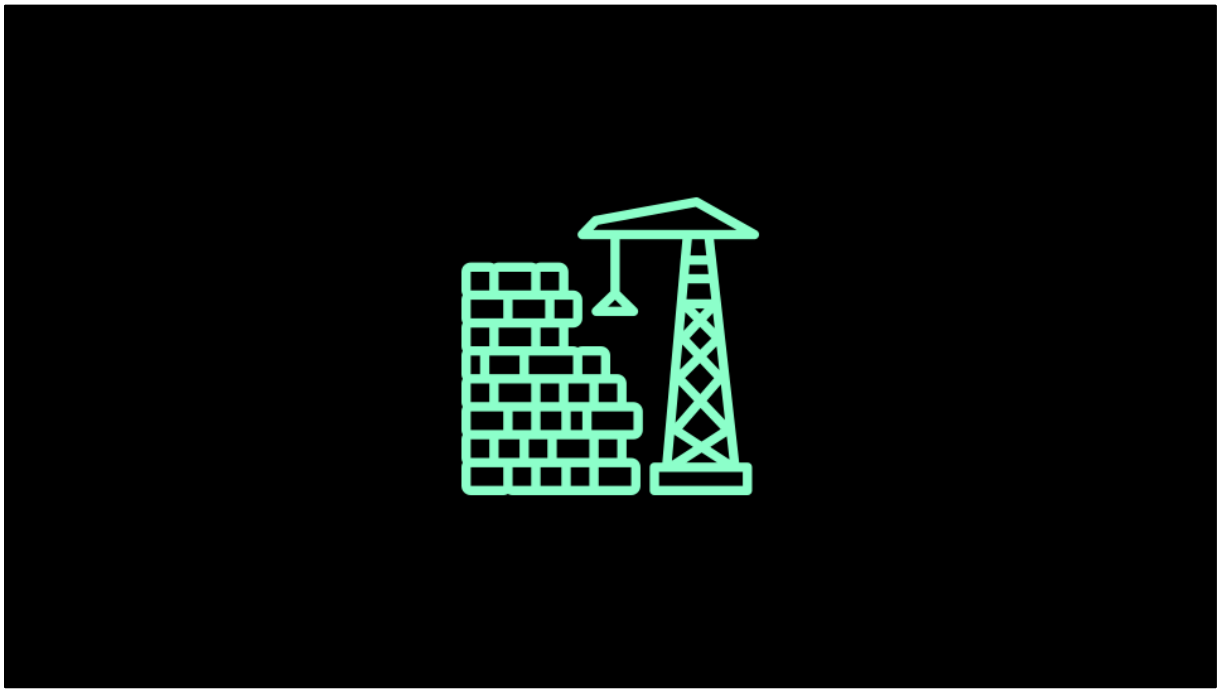
ENCOUNTERING NEW BUGS

Who here has encountered a bug in their code?



Our immune system doesn’t just immediately know how to get rid of everything that makes it into our bodies. It has no idea that bug exists until it shows up. (In the adaptive immune response)

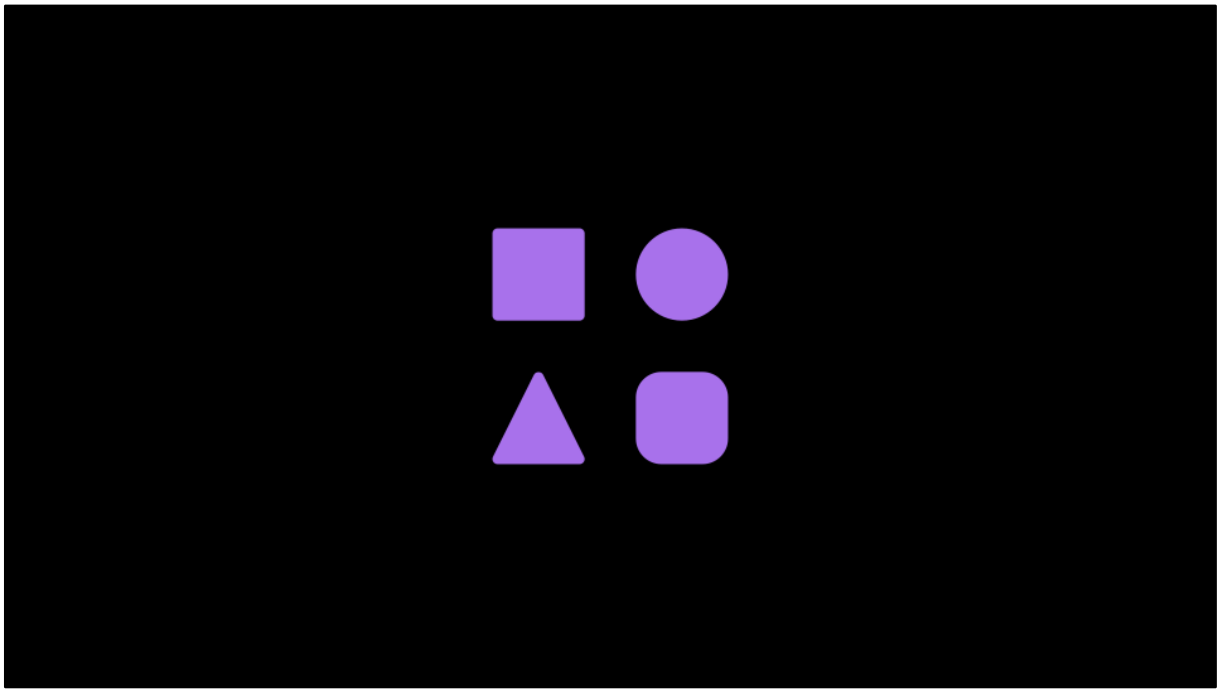
When it runs into a new bug, it begins to develop antibodies for that bug



Those antibodies will protect us from the bug if we encounter it again. The solution for that bug has already been found.

To effectively debug, you have to pay attention to bugs you encounter and learn about them.

Bugs can cause a lot of stress and frustration, especially if they keep cropping up, don't seem to have a solution, or you're working on a tight deadline. The most important thing you can do when you hit one is to stop and think about how you're going to approach it and what you can learn from it.

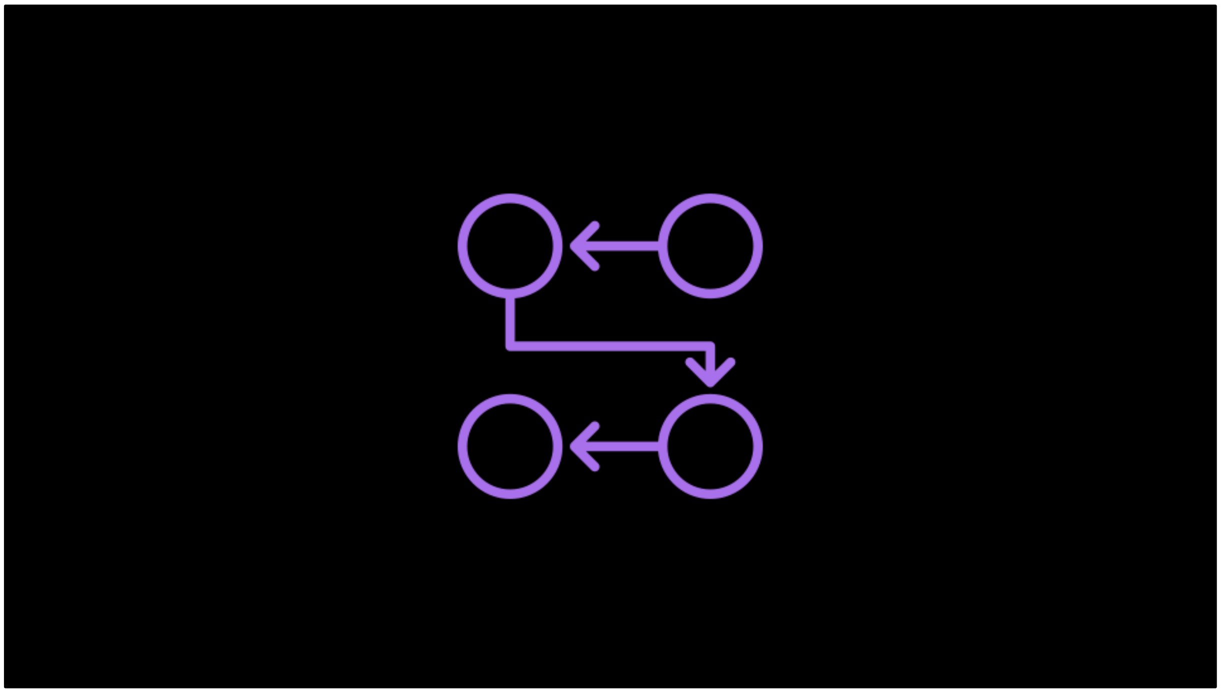


Note the type of the error. That’s one of those helpful hints that error messages will give you.

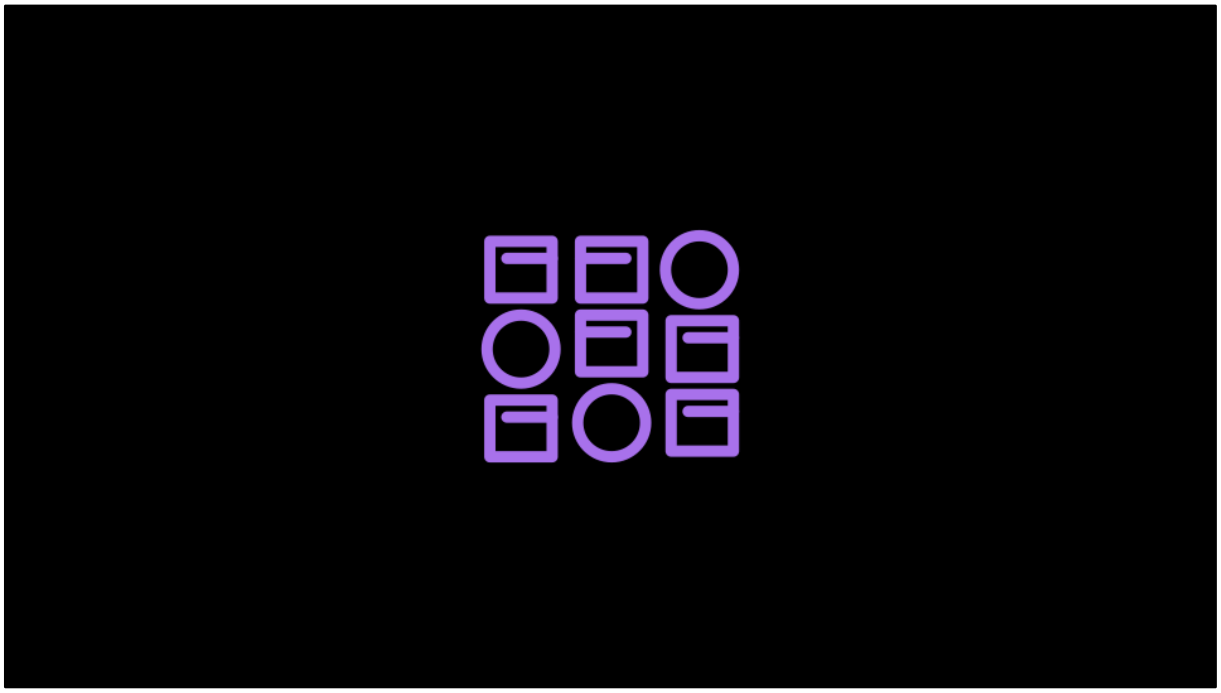
Reference errors and Type errors have distinctly different causes. Keeping the type of error in mind as you look for the bug will help you evaluate if a block of code is a likely culprit. My favorite is when I get a reference error because I’ve misspelled my variable name in two completely different ways between declaring it and reading it.

Note the symptoms of the bug. What behavior does it cause? That could be a valuable clue if your error messages haven't given you enough information, or if there's no error message.

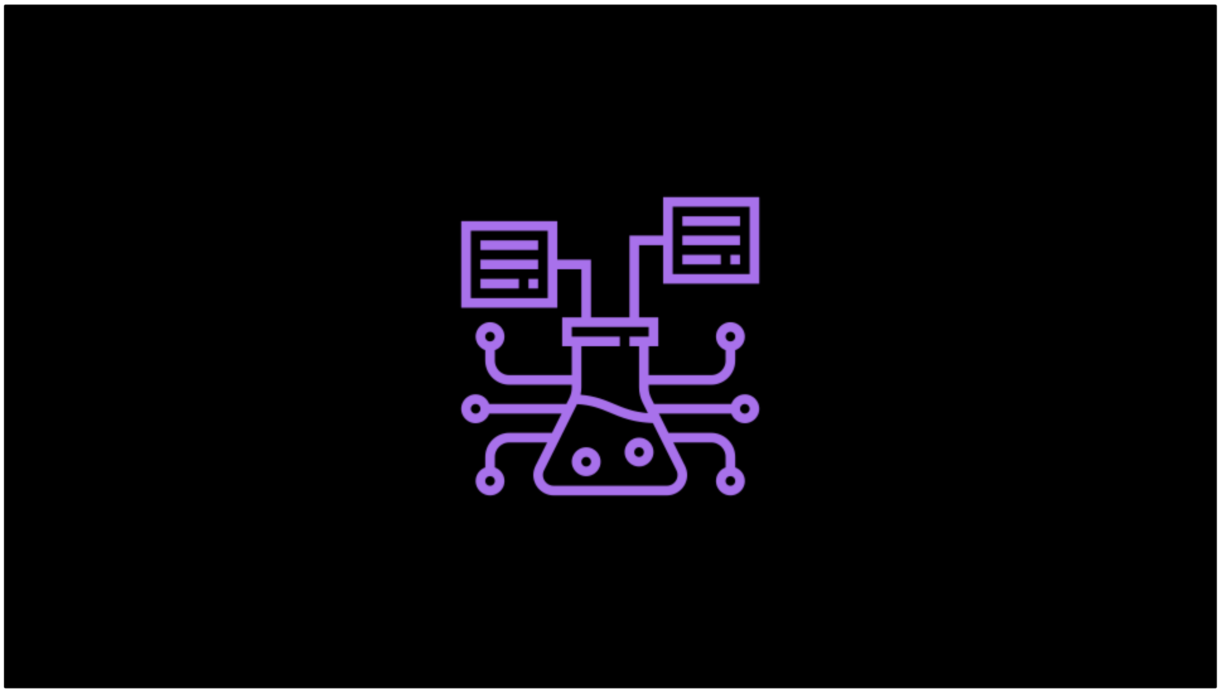
I had a very weird bug when I was learning to code where a timer I was writing kept zeroing out almost immediately when it was started. I had a `setInterval`, and I was bringing in the value the user entered just fine. It would tick down one second, and then, bam, it was zero. I agonized over that behavior, because I could not figure out where I was doing that. I could clearly see that my function was lowering the value by a second with every interval. I was ready to close my computer and walk away. Coding didn't make sense. With some help, I discovered that I had failed to put enough equal signs in the conditional that would clear my interval. Instead of checking the seconds remaining against 0, I was setting them to 0. Now I know that if a value is being set to something I don't expect, it's a good idea to scan through my conditionals, though I've also gotten more careful about writing them.



Note the cause. In my code, that often means I'm looking for a misspelling. Knowing what caused the error you fix now will help you find, or even avoid, future errors faster. I watch my conditional expressions a little more carefully now. Another bug that comes to mind is messing up your binding of 'this' with function declarations. Once you've run into a few of those, that becomes a cause to look for, if your references are coming up 'undefined'.

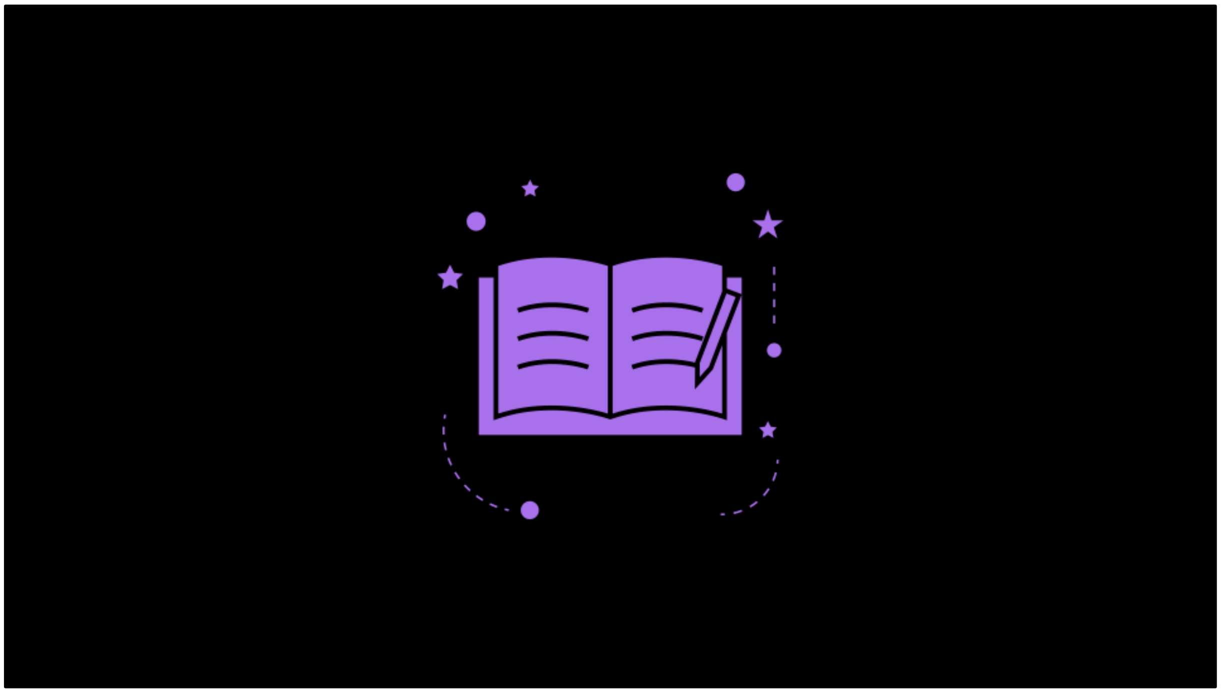


Pay attention to similar bugs. There's a few errors in JavaScript that I can immediately tell you don't typically point at the right line in their error messages. Missing brackets can do this. That gives me valuable information, because I've left off one often enough to know that an error pointing to line 54 when there's only 30 lines in my file is likely due to a missing bracket or parenthesis.



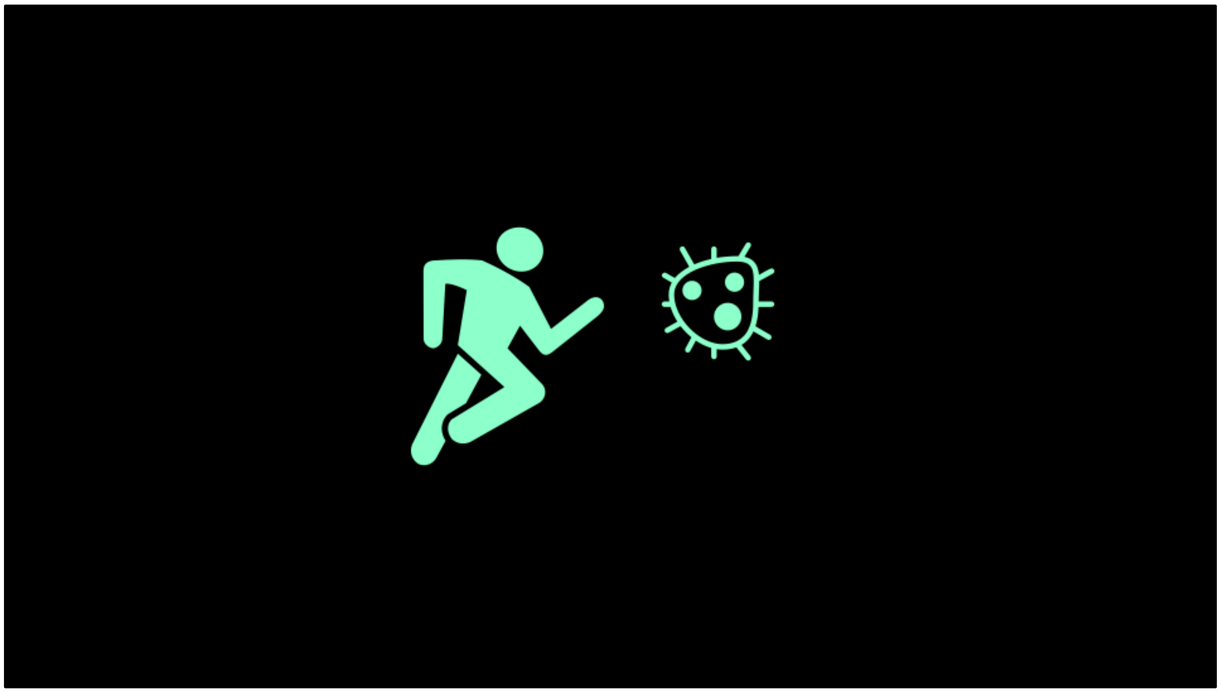
When you're dealing with a bug you've never seen before, unless there's an obvious cause, like a typo or a missed bracket, you're probably going to need to experiment. Seeing how small changes affect the error or the behavior of your app is valuable information. If you make small, incremental changes, you can isolate behavior. I know I'm spoiled in Javascript land, it takes me about 2 seconds to refresh my entire app, but no matter the language, it's a very good way to learn about odd behaviors.

If I'm writing an endpoint and I'm getting a 200, but the response body is empty when it shouldn't be, I'm going to start at the query to the database and add logging and breakpoints to see what my app is getting back from the db. It's not getting a response, but I've failed to capture the error correctly. Now I'm putting in my error handling and moving to the next part to see where my data is getting lost. And so on.



Keep a record. It’s not that uncommon to note down how issues were solved. That’s basically what Stack Overflow is. Keeping track of bugs common to your code base in a small notebook or your favorite note program can save you a lot of time. Those notes are your code’s antibodies.

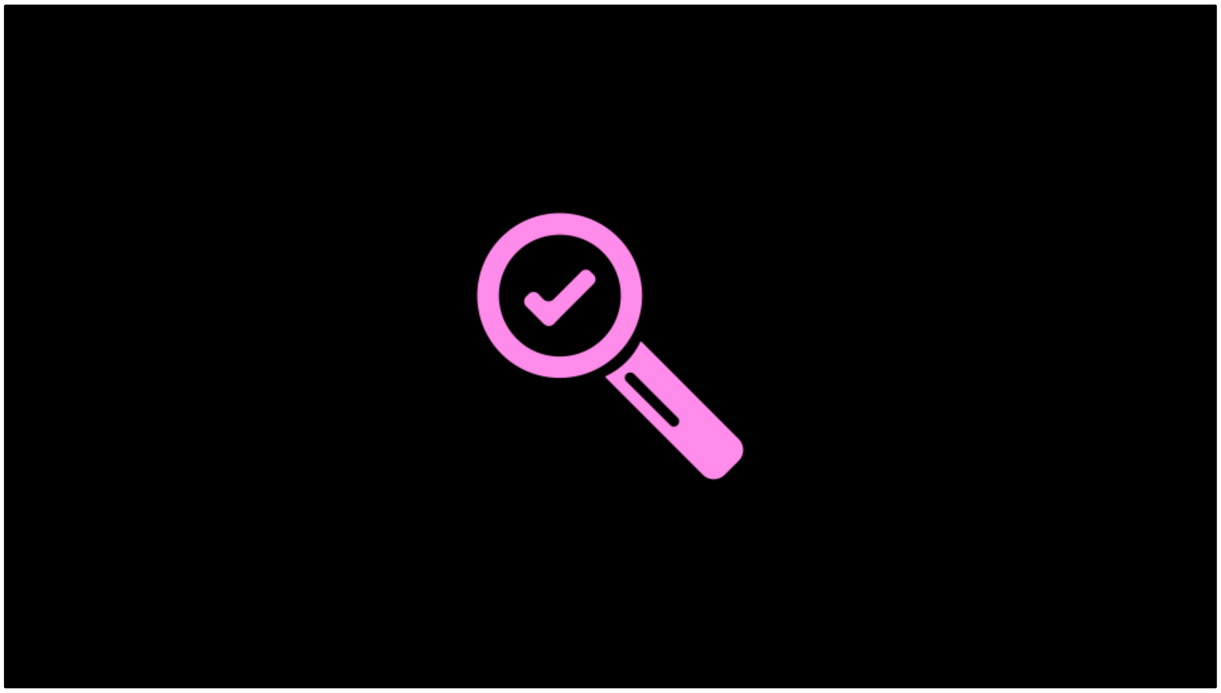
DEALING WITH BUGS



Once your immune system has antibodies for a bug, it can take care of it much more quickly in the future, and you may not even get sick to start with. Once you are familiar with a bug in your code you:

will be less likely to write that bug into your code

I’m typically watching for typos, poorly formed conditionals, and missing commas.

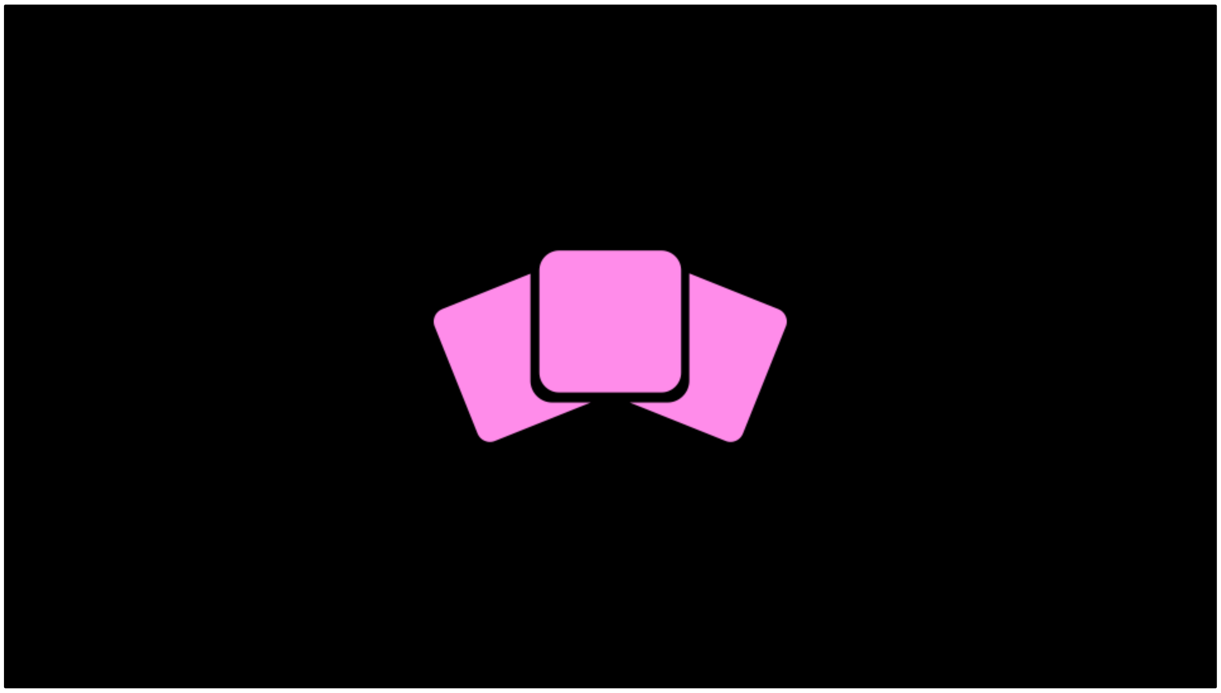


will be able to identify what is causing an error much more quickly

Sometimes I see an error and know exactly where to look. Benefit of messing up my code so often.

will be able to hone in on the solution

Once you find the error, your antibodies, or records, will tell you how to fix it.



will be able to solve similar bugs more easily

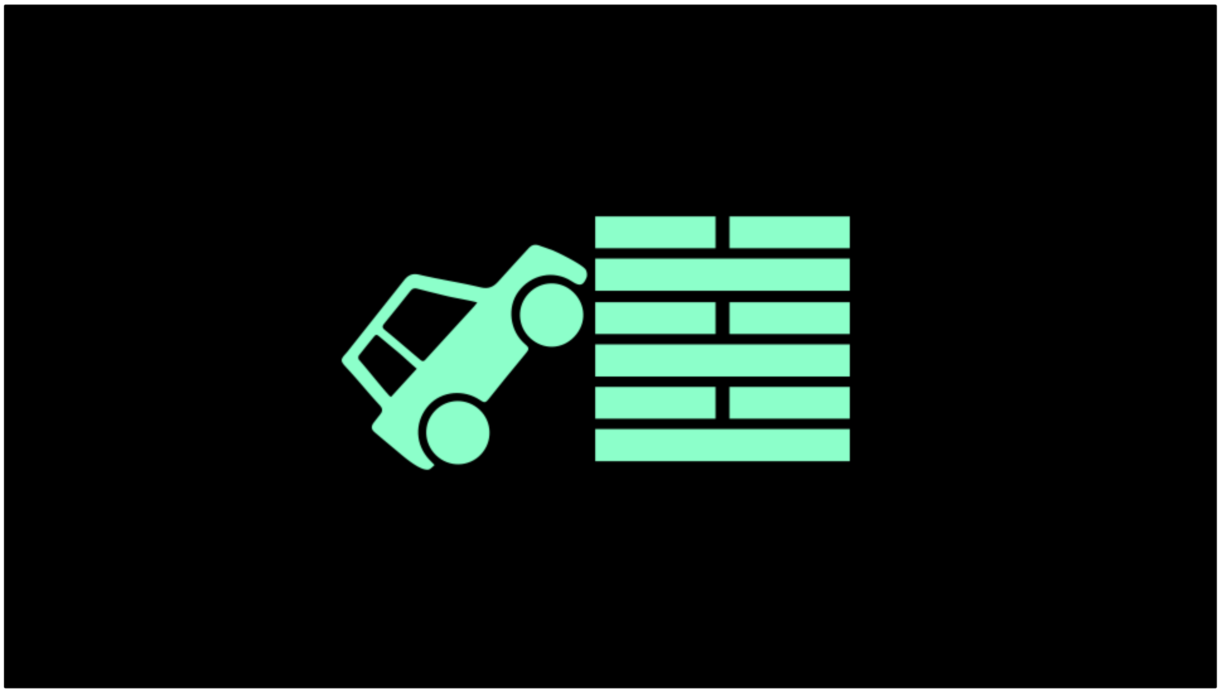
So now you’ve got a great knowledge base of how to deal with reference errors. Even if the bug is new, you’re going to have a great jumping off point to figuring it out.

will be able to help others with bugs

Pausing here. Helping other people debug can be the most rewarding and most frustrating thing you'll ever do. I personally love it. Not only can I help them get unstuck and hopefully remove some frustration from their day, you have the opportunity to teach them about the bug. You can give them the knowledge to beat the bug on their own next time. The important thing is to guide them to figuring out the bug for themselves, not just pointing out the issue. It's this process we've been talking about, the experimentation and discovery, that really builds up your skill at debugging. It's hard to do. It takes longer, and we're all eager to solve problems. But it's so worth it.

WHEN YOU NEED OUTSIDE HELP

Sometimes the knowledge you have isn’t enough to solve the issue.



Your immune system has its limitations.

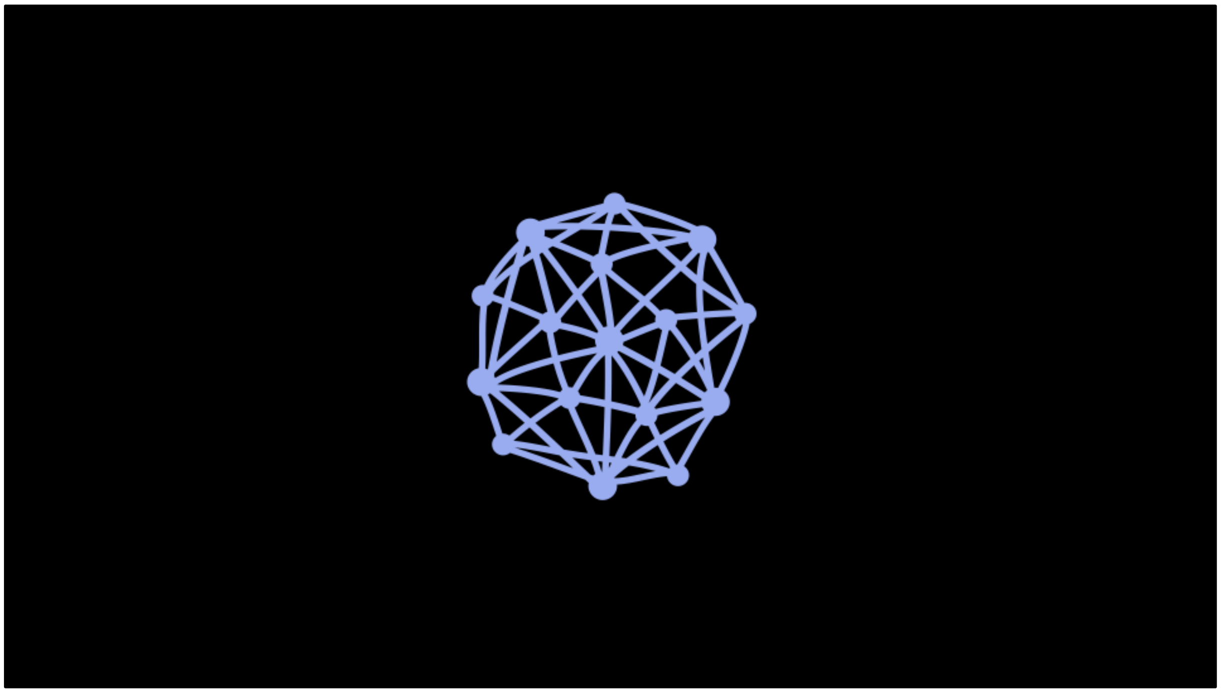
Sometimes the bug is just too strong for it, or has countermeasures against the immune system.

It can only go after malicious organisms in your body, so external irritants are outside it's reach.

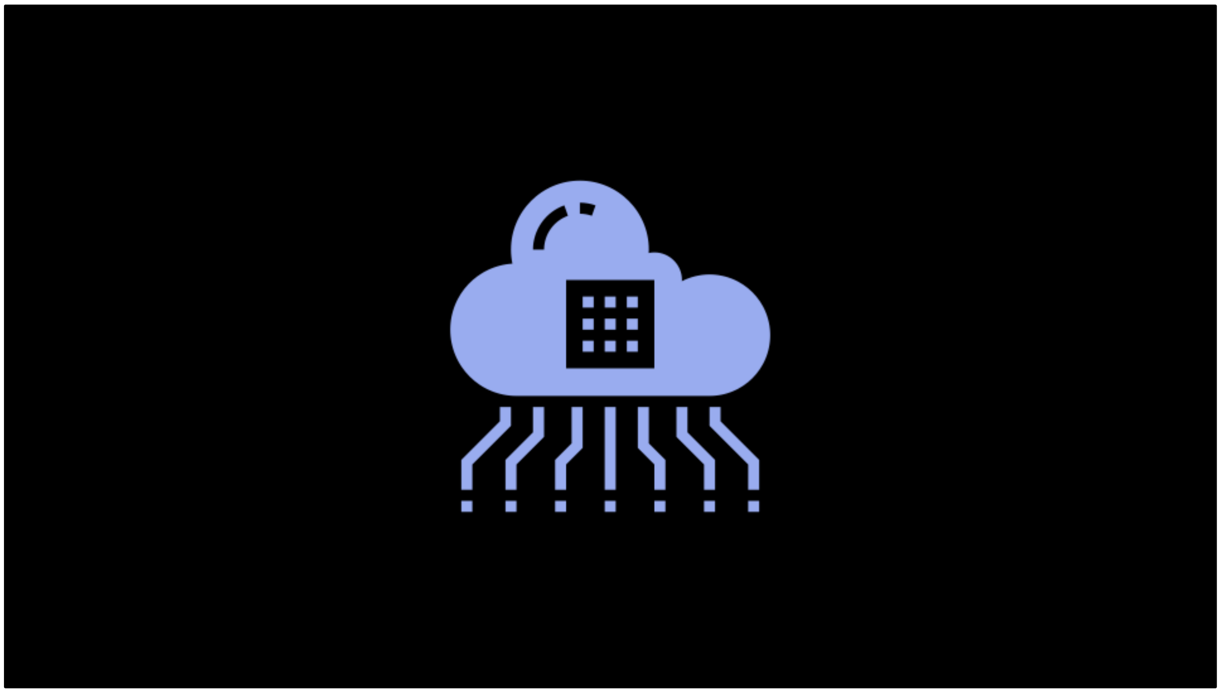
You might rarely need to seek outside help, if your debugging knowledge is well developed and you have a lot of control over your environment, but that's certainly not true of a lot of us. You might need this frequently, because your immune system isn't as developed, or you haven't been doing Java that long. Learning this stuff takes time! It's okay to get help, but pay attention when you do and start building up those antibodies.

Then, of course, there's everyone's favorite, environmental issues.

The database could be acting up.

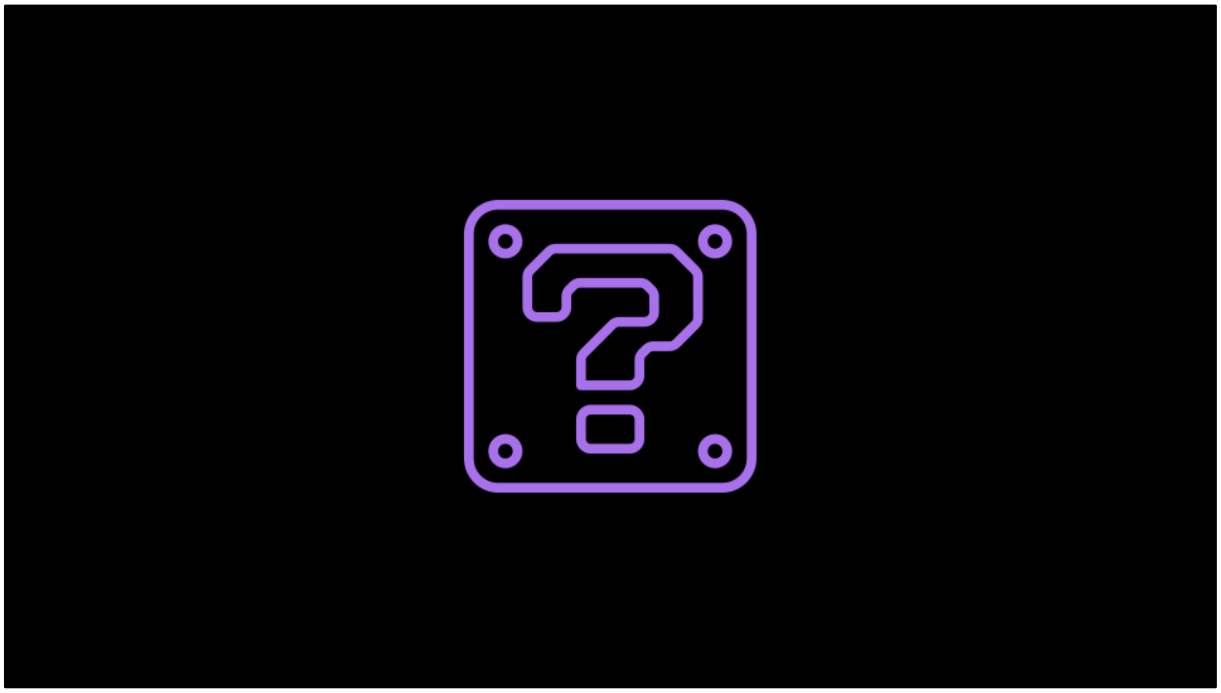


Other services or apis that you rely on could be having their own issues.

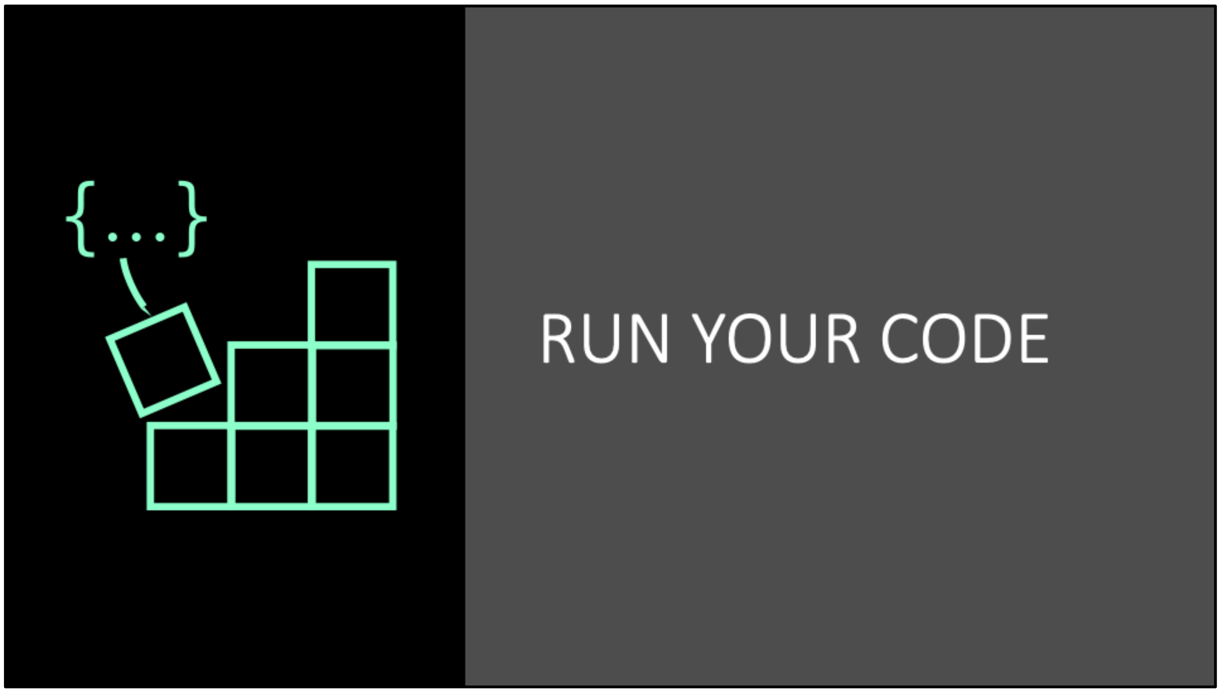


The network or the cloud could be under the weather.

Even if you don't have the permission or resources to fix these things, it's important to note the symptoms and causes, because then you'll be able to pin down a db issue much more quickly the next time it happens.

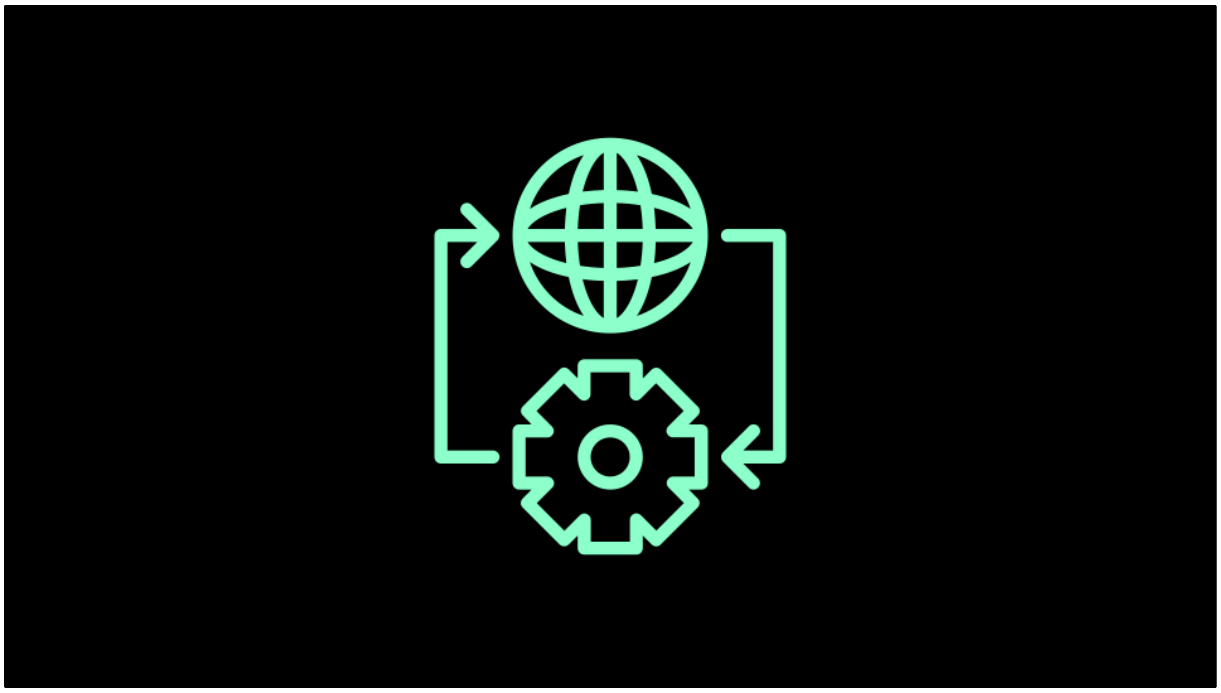


There's a key step to making all of this happen, though. I mentioned it in a previous section, but I can't state its importance enough.



RUN YOUR CODE

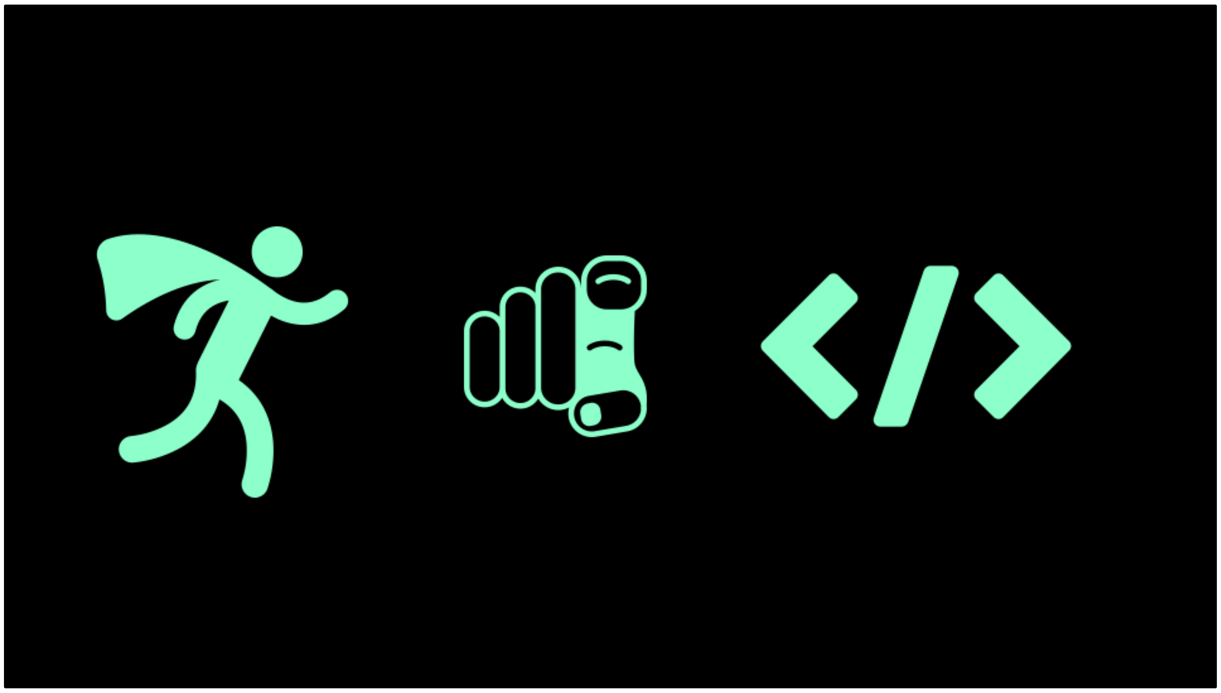
Run your code.



Run code you think is right to make sure it works

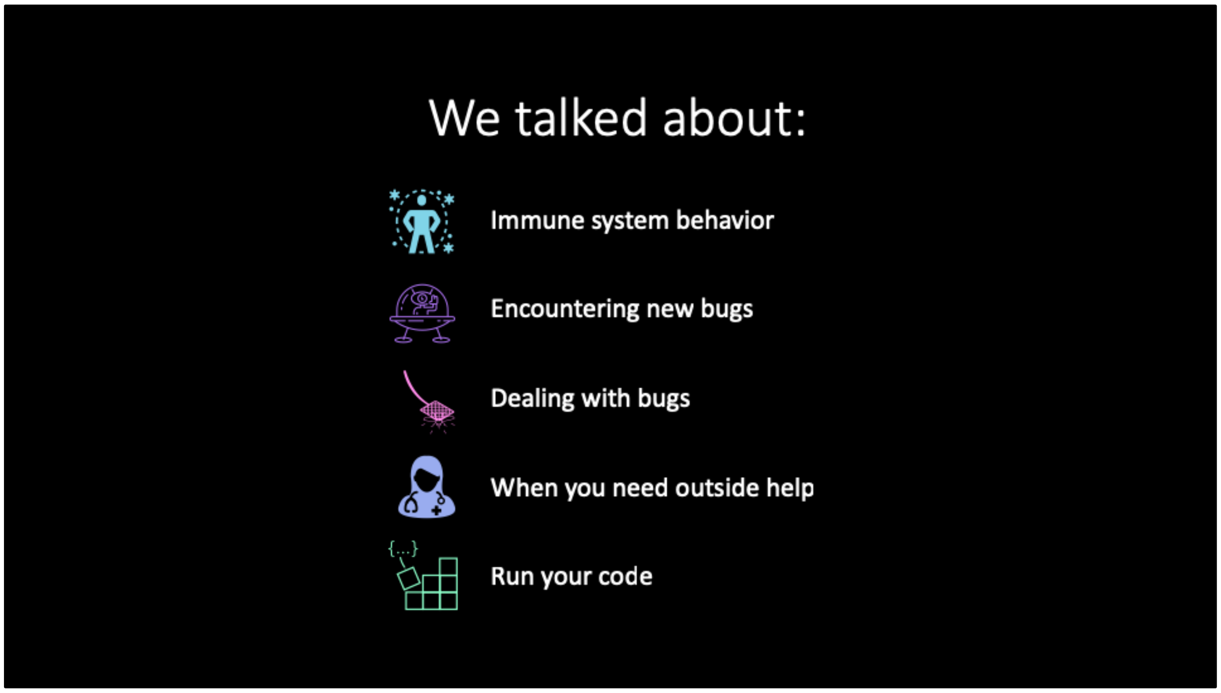
Run code you feel uncertain about to see what it does

Run code often so that you can isolate changes that might break your app



Run. Your. Code.

You will learn so much from it.



We talked about:

- Immune system behavior
- Encountering new bugs
- Dealing with bugs
- When you need outside help
- Run your code

These tips will help you debug anything that might come your way and write better code. Keep building. Keep messing up. Keep learning.

@marlena_baker2 baker-marlena /in/marlenabaker DenverDevs: marlena.baker

THANK YOU

I’m terrible at social media, but I try. I exist on Twitter, Github, LinkedIn, and the Denver Devs Slack space. I’d be happy to chat after the talk if you have any questions. Thank you very much.