# Debugging Overview

Any programmer will tell you that it is incredibly common to be making great progress working through a coding problem when all of a sudden an error like this gets thrown at you:

/home/ccuser/workspace/learn-javascript-debugging-code/app.js:1

"my\_name".capitalize();

          ^

TypeError: "my\_name".capitalize is not a function

...

Alternatively, you might have finally finished implementing a complex function only to find it doesn’t work at all as you expected:

console.log('The square root of 4 is ' + squareRoot(4));

=> 'The square root of 4 is 1.87878787'

If any of these things happen to you, don’t fret! In our article [Thinking About Errors Differently in Your Code](https://news.codecademy.com/errors-in-code-think-differently/), you discovered that these experiences are normal for all developers, no matter how experienced they are. A key skill developers possess is knowing how to debug their code when issues occur.

This lesson will build on that article, demonstrating how to implement those skills in real-life JavaScript situations. If you haven’t had a chance to read the article yet, we highly encourage you to do so right now.

Alright, are you ready to begin your transition from frustrated coder to bug squasher extraordinaire? Let’s get started!

Error Stack Traces

We’ll start this lesson by taking a closer look at the most straightforward way to know your code isn’t working as expected: errors!

You might recognize errors as the scary red text that appears on your screen when you try to run broken code. A piece of software, called a compiler, is trying to translate your code so that your computer can understand and run it. However, the compiler is coming across a piece of code that it can’t interpret. As a result, it throws an error back to you to let you know that it has to stop and why.

This information is logged as an **error stack trace** — a printed message containing information about where the error occurred, what type of error was thrown, and a description of the error.

Take a look at the diagram to the right to see where you can find all of this information.

As a new programmer, it’s easy to see an error and be overcome with a sense of dread and self-doubt. However, you’ll soon realize, errors are the best kind of bug you can hope for, since they tell you exactly what went wrong and where.