Introduction to Regular Expressions

This lesson explains and introduces Regular Expressions.

Time to apply specific rules so that each field only errors if the user input breaks the rules for that particular field.

Regular Expressions

Our rules are for strings, and the *only* robust method of string validation is through **regular expressions**. If you haven't heard of it, a regular expression (shortened to regex) is a powerful tool in many programming languages that let us define patterns to match strings. They can look a bit unfriendly at first, but I'm certain you'll find them extremely useful and worth familiarizing yourself with.

The basic concept is you have two delimiters that mark the beginning and end of your regular expression, `/`. In the middle, you can use groups and symbols to specify rules. Let's just learn through example.

There's one thing to keep in mind as we're going through these examples: while regex is very powerful, we want to use everything at our disposal to make things **human readable**.

Readability

Tasks in JavaScript are often not expensive enough for frontend developers to concern themselves with performance metrics, or at least not as often as backend developers. We're writing code for one browser and one user interacting with that browser. Of course, this isn't true as a blanket statement across all frontend tasks and doesn't mean you shouldn't care to learn about that aspect of programming, but it does mean that when you're thinking about tradeoffs, readability should be weighted heavily. Your inputs are probably not big enough to make your hand-written binary search make a noticeable performance difference compared to for-looping search, but it will be less readable. Pure regex can be more concise, but not as readable as other methods.

We'll learn how to put these Regular Expressions into practice for user input validation in the next lesson.