Introduction

You'll learn about the topics this chapter contains, which include writing clear and concise conditionals to simplify code.

We'll cover the following

- Writing cleaner JavaScript code
  - Example
- What does this chapter include?

Have you ever been sucked into a round of spring cleaning? I don't mean the kind where you put "Scrub the floors" on your to-do list and then you actually do it. Rather, say it's a nice day so you open the window, but as soon as you do, a pile of papers blows off your desk. That's fine, you've been meaning to organize those anyway. So you start filing, but without that stack of papers hiding everything, now you notice that your computer cables look all tangled and sloppy, and how long has that coffee mug been hiding back there? Before you know it, you're taking your whole office apart. Once you start removing clutter, it's hard to stop.

Writing cleaner JavaScript code

By now, you've likely started getting a taste for clean and simple JavaScript. And that's wonderful. The new syntax allows you to do so much more with much less code. But you don't need to wait for new syntax before you make a positive change to your code.

Let's take a quick detour from new syntax to explore some older ideas, but with a new goal: *making clean and predictable JavaScript code*.

In this chapter, you're going to clean up conditional expressions. You'll revisit basic ideas, such as truthy and falsy values, ternary expressions, and short circuiting, with the goal of keeping everything simple and clean.

There's also a practical side: Now that you have more tools to assign and work with data, you can reuse old ideas to further leverage the new syntax.

Example

Here's a basic example: Let's say you wanted to set the *color* on a value. If the value is a *negative* number, you want the color to be *red* . If the value is *positive*, you want it to be *green*.

```
const transactions = [...spending, ...income];
const balance = calculateBalance(transactions);
let color;
if (balance > 0) {
    color = 'green';
} else {
    color = 'red';
}
```

The first two lines are like my newly cleaned desk. They're clear and expressive and are assigned with `const`, so you know they aren't changing. But as with spring cleaning, you look down and suddenly things seem awfully messy.

Where did that `let` come from? Oh, right. You need it to set the `color`, which will be mutated by the *conditional*. I guess it's okay, but it just doesn't feel as clean as the rest of the code.

All of a sudden, that block of code just doesn't look right. Fortunately, you don't need to leave it like that. You can rewrite it with the same simple syntax as before. No new syntax is necessary.

What does this chapter include?

To start off, you'll look at *truthy* and *falsy* values in JavaScript. Many techniques to simplify code involve truthy and falsy values, so you'll want a firm foundation. Next, you'll look at *ternaries*, a simple method for reducing `if`/`else` conditionals to a single line. Finally, you'll learn to write extremely *concise* conditionals and *variable assignment with short-circuiting*.

It's time to clean up the clutter in your conditionals. Between *truthy values, ternaries, and short-circuiting*, you'll be writing conditionals that fit with your modern JavaScript code. And as you move into array methods and functions, you'll see these ideas return over and over.

Let's get started putting your house in order.