Introduction

You'll learn about the topics this chapter contains, which include using npm, component architecture pattern, and using CSS.

We'll cover the following

In chess, there's a distinction between tactics and strategy. Tactics are a series or combination of moves to achieve a goal. It's short term and fairly easy to define and teach. Strategy, by contrast, is an abstract conception of which side has more influence in a particular part of the board. Tactics solve problems at hand. Strategy allocates resources in anticipation of future problems.

Most coding books, are about tactics. *How do you transform an array? How do you write a function that consumes asynchronous data?*

What does this chapter include?

Now you're going to learn strategy. In the software development world, organizing code in a way that makes it *extendable, reusable, and manageable* is called **architecture**. Instead of solving clear problems, you're going to learn to split and organize code to make future problems—*extending classes, handling edge cases, finding bugs—easier* to solve. You got a taste of code architecture when you learned about *dependency injection* in Tip 32, Write Functions for Testability. That was the first time you saw how code could be split to keep one function focused and flexible.

Most books avoid discussing architecture because it's messy. The solutions aren't clear, and you can usually only recognize a mistake months later when it's too late and too expensive to change the structure of a codebase. In this chapter, you'll see how to structure a project from the ground up. You'll learn how to incorporate modern JavaScript tooling to pull the pieces together into a final product.

You'll start off by learning how you can separate code into different files using *import* and *export*. Next, you'll learn how to incorporate third-party code with `npm` *import* and *export*. Next, you'll learn how to incorporate third-party code with `npm`. Then, you'll learn how to break an application into well-designed pieces using the

component architecture pattern. This isn't the only architecture pattern, but it's the most popular one now, and it's very different from most server-side patterns.

Next, you'll combine the pieces into a final, usable asset with build tools. And you'll finish up by learning how to use CSS to handle animations that used to be the responsibility of JavaScript.

This chapter will be a little more difficult. There are more moving pieces, and the examples are more complex, even as they're still extremely simplified compared to anything you'll see in production. But if you come away understanding that clean architecture is just as important—and just as achievable—*as clean code*, I guarantee your projects will benefit.

The first step in creating a clean architecture is breaking code into reusable and shareable pieces with *import* and *export*. Let's explore that in the next lesson.