Introduction

You'll learn about the topics this chapter contains, which include using different collection types.

We'll cover the following

- What does this chapter include?

I'll admit I like organizing things. I change my garage constantly. I keep all the little screws in a box with drawers that I can easily pull out. I keep most hand tools on a pegboard, but I have a couple of small toolboxes stocked with the most common items for carrying around the house to do small repairs. And I keep a few cardboard boxes full of odds and ends.

The thing is that I don't need those containers. I can (and have) stuffed everything into one cardboard box, but it certainly doesn't make my life easier. I keep things separated because the container does matter. It changes how quickly I find what I need (drawers), how easily I can identify what I have or don't have (pegboard), and how seamlessly I can transport the whole group of things around the house (toolbox). I bet you see where this is going.

What does this chapter include?

This chapter is all about how to use *collections* to make your data easy to use and accessible. After all, the collections you use for your data can change how you work with the data.

The beauty of code is that, unlike my garage, you can switch back and forth between containers. That's great. You should always use the best collection for the job, and fortunately for you, the options in JavaScript have significantly increased.

What to do with all this new information? When choosing a variable declaration, you learned that the most important consideration was signaling intention to future developers. Similarly, when choosing a collection, you just have to ask yourself one question: *how can you maintain simplicity and flexibility?*

In this chapter, you'll look at different collection types and how they can give you

In this chapter, you'll look at different collection types and how they can give you flexibility and simplicity and when they might lead to confusing and buggy code.

You'll start off by looking at objects used as key-value collections and when they're an appropriate choice for data that won't be changed. From there, you'll see two new collections, `Map` and `Set`. You'll learn why those were introduced and how they create clear interfaces for working with data that you'll update or iterate over.

In addition, you'll learn how and when to switch over to another structure to take advantage of its methods before switching back to your original structure.

Even if you make a choice that ends up being wrong (and who among us hasn't made a few wrong choices when writing code?), you aren't going to be bound by it indefinitely. Take a look at the choices, select the one best suited to your task, but don't be afraid if you have to switch later. It's easy. The collection you use to hold your data does matter. But unlike my garage, you won't need to spend a Saturday rearranging things if you need to make a change.

In the next lesson, you'll see how to use objects for key-value lookups.