Introduction

You'll learn about the topics this chapter contains, which include series of loops and array methods used for looping.

We'll cover the following

My auto mechanics teacher in high school once started off class by holding up a large adjustable wrench. "Some people think this is the only tool they need," he said. He demonstrated how someone could use the wrench to unscrew a bolt, take off a screw, and pry off a gasket like a set of pliers. "They even use it as a hammer," he said as he swung the wrench in the air.

"But if all you use is a wrench, you'll round off your bolts. You'll snap off fan blades, dent your car, and generally make things worse." As he finished, he placed the wrench on the bench. "And that's why we have specialized tools," he said as he showed us how to use ratchet wrenches, torque wrenches, spark plug gap gauges, and all the other tools a good mechanic keeps at hand.

Looping in JavaScript

You're about to get your specialized set of tools for looping through data in JavaScript. When you use the right tool, your code will be more clear, and you'll signal your intentions to other developers.

You've almost certainly used a `for` loop before. If you wanted to convert an array of integers to strings, you could easily use a `for` loop. You simply need to create an empty array, loop through the values one at a time using the *index* from the `for` loop, and push the new values into new array.

But you know that will just create problems. You'll have more variables to keep track of. You'll have mutations. And other developers will have no clue what you're trying to accomplish. When you use one type of loop, other developers won't know if you're returning all results of an original array or just a subset. They

won't know at a glance if you're converting data types or changing the collection from an array to an object.

If you try to solve every problem with one tool, your code will be complex and hard to use. You need better tools.

What does this chapter include?

We're going to start off by jumping ahead a bit to see how to use arrow functions. **Arrow functions** turn simple loops into one-liners. Next, you'll get an in-depth look at how array methods simplify loops and how you can know what to expect based on the specialized loop. From there, we'll explore several different array methods one at a time. We'll start with `map()`, which pulls the same information out from each collection. We'll look at a series of specialized loops that do one task well, finishing with the most flexible loop, `reduce()`, which can do pretty much anything. We'll wrap up by exploring the updated `for...of` and `for...in` loops, which use clear variable names instead of indexes.

The best mechanics can always grab the right tool for the job. Coding is no different. If you look at the best code, you'll notice the developers always use the loop that fits their intentions. These new loops take practice and can be confusing. But before you know it, you'll be able to use them with ease, and you'll love how clear and expressive you've made your code. Special tools exist for a reason. Time to learn how to use them.

To start, take a look at one of the most beloved new features in JavaScript: *arrow functions*.