Introduction

You'll learn about the topics this chapter contains, which include classes, prototypes, and generators.

We'll cover the following

An octopus has a brain. But in a way, it has many brains. Neurons run throughout an octopus's body, so in a sense, it can think with its arms. If its arm is removed from its body, it can still respond to stimulus, change color, and reach and grab for items.

This poses a challenge to researchers. An octopus can think, but not in the way mammals do. It responds similarly—it can remember specific people, prefer certain types of food, and even plan an escape—but the way its consciousness works is unusual. An octopus's mind is similar but different.

What does this chapter include?

In this chapter, we're going to explore concepts that are familiar, but they're also different. Developers have been complaining for years about these differences—they say JavaScript is broken because it doesn't behave like other object-oriented languages. The problem isn't that it's different. It's that it's just similar enough to create confusion.

In ES6+, things have gotten even more confusing because now JavaScript uses familiar syntax—*class, extend,static*—but the code doesn't always act as you'd expect if you come from another object-oriented language. In this chapter, you're going to recognize how classes in JavaScript are different even while they use many familiar concepts.

To start off, you'll build and extend a class. This will look similar to most object-oriented languages. After that, you'll see how classes work under the hood by combining class syntax and JavaScript prototypes. Next, you'll learn how to *mask complexity with getters, setters, and generators*. Finally, you will return to common *complexity with getters, setters, and generators*. Finally, you will return to common problems with the `this` keyword and techniques for solving them.

Remember that classes in JavaScript are a little different, but they still retain many of the benefits—and some of the problems—of class syntax. Classes will help you organize your code, build new instances of objects, and store local properties. Just keep in mind that JavaScript is its own language with a unique history and paradigm. Learn how you can use classes to fuse your existing knowledge of both JavaScript and other object-oriented languages.

In the next tip, you’ll learn how to build classes.