Introduction

You'll learn about the topics this chapter contains, which include handling parameters and destructuring

We'll cover the following

I'm famous for assuming I can always find a shortcut. If I'm driving down the highway and hit construction, I'll take the first off ramp, determined to find a quick way around the delay. Exiting the highway requires that I ignore the protests of my wife, who thinks it would just be easier to slow down and follow the orange cones.

Well, my wife is usually right. I pull off and take a side road that suddenly veers even further off course. I don't mind. I grew up in the middle of nowhere, so I have an intuition for county roads. At least I think I do, until the paved road turns into a dirt road before coming to a dead end at a wheat field. Giving in, I take out my phone to turn on the GPS. Oh wait. I'm in the middle of a wheat field. There's no signal.

Simple actions can spiral out of control quickly. This happens all the time with function arguments. You start with the best of intentions. The function will take two arguments and return a simple value. Suddenly, edge cases pop up. Data inconsistencies creep in. Before you know it, you need eight different parameters to cover dozens of situations. You'd like to give up, but by now, you're too afraid of breaking all the code downstream that depends on this function.

What does this chapter include?

In this chapter, you'll learn how to plan for changing function arguments and how to create parameters that will be clean and give you flexibility.

First, you'll see how to add *default parameters* to cover situations where information may not be available. Next, you'll learn how to *pull* information out of objects using *destructuring* and how destructuring can be combined with function objects using *destructuring* and how destructuring can be combined with function parameters to accommodate a range of options. Using that knowledge, you'll

combine information back into new objects, *creating return statements* that share plenty of information in usable bundles. After that, you'll return to parameters to see how you can *create* functions without even knowing the number of arguments to expect.

There's no problem with being adventurous as long as you have a plan for the inevitable contingencies. In this chapter, you'll see how functions can be built to handle unexpected changes. Learn the lesson I never did as a driver: Leap into the unknown, but plan for the unforeseen.

In the next tip, you'll look at default parameters.