## **JAVASCRIPT FUNDAMENTALS PART 3**

### **Functions**

Things are about to get *really* exciting. So far you have been writing an impressive amount of code to solve various problems, but that code has not been as useful as it could be. Imagine taking one of your scripts and bundling it into a little package that you could use over and over again without having to rewrite or change the code. That’s the power of functions, and they’re used *constantly* in JavaScript.

1. [This lengthy MDN article](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Functions) is a good place to start. Pay special attention to the sections on ‘Function Scope’. Scope is a topic that commonly trips up both beginner and intermediate coders, so it pays to spend some time with it up front.
2. Read this article about [return values](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Return_values).
3. Let’s discuss parameters and arguments in the context of the following example function:  
   function favoriteAnimal(animal) { console.log(animal + " is my favorite animal!") } favoriteAnimal('Goat')
4. In JavaScript, parameters are the items listed between the parentheses in the function declaration. Function arguments are the actual values we decide to pass to the function. In the example above, the function definition is written on the first line: function favoriteAnimal(animal). The parameter, animal, is found inside the parentheses. We could just as easily replace animal with pet, x, or blah. But in this case, naming the parameter animal gives someone reading our code a bit of context so that they don’t have to guess what animal may eventually contain. By putting animal inside the parentheses of the favoriteAnimal() function, we are telling JavaScript that we will send *some* value to our favoriteAnimal function. This means that animal is just a placeholder for some future value. But what value are we sending? The last line, favoriteAnimal('Goat'), is where we are calling our favoriteAnimal function and passing the value Goat inside that function. Here, Goat is our argument. We are telling the favoriteAnimal function, “Please send ‘Goat’ to the favoriteAnimal function and use ‘Goat’ wherever the ‘animal’ placeholder is.” Because of the flexibility that using a parameter provides, we can declare any animal to be our favorite. Feel free to experiment with the code on your own and replace Goat with your favorite animal. Notice how we can change the argument to anything we like? Try changing animal in the function declaration and in the function body, too. What happens when you do?
5. Next, read [this article](http://javascript.info/function-basics) from Javascript.info. We’ve mentioned this before, but JavaScript has changed a bit over the years and functions have recently received some innovation. This article covers one of the more useful new abilities: ‘default parameters’. (NOTE: The last “task” at the end of this lesson uses loops, which you will learn about in the next lesson. Don’t worry about that one.)
6. Now, read [this article](http://javascript.info/function-expressions) about functions in JavaScript to give you a little more context, and read [this article](http://javascript.info/arrow-functions-basics) for an introduction to a relatively new feature in modern JavaScript called the arrow function. Arrow functions are useful but not crucial, so don’t worry about them too much just yet. We include them here because you are likely to encounter them as you move forward, and it’s better that you have at least *some* idea of what you’re looking at whenever they crop up.
7. Finally, read [this article](https://medium.com/javascript-in-plain-english/node-call-stack-explained-fd9df1c49d2e) about call stacks and how return works in the context of chained function calls. Don’t worry if you don’t fully understand this yet, but it’s important to keep in mind where your returned values are going. This doubles as a bit of early computer science as well.

### **Practice**

Let’s write some functions! Write these in the script tag of a skeleton HTML file. If you’ve forgotten how to set it up, review the instructions from fundamentals 1.

For now, just write each function and test the output with console.log.

1. Write a function called add7 that takes one number and returns that number + 7.
2. Write a function called multiply that takes 2 numbers and returns their product.
3. Write a function called capitalize that takes a string and returns that string with *only* the first letter capitalized. Make sure that it can take strings that are lowercase, UPPERCASE or BoTh.
4. Write a function called lastLetter that takes a string and returns the very last letter of that string:
   1. lastLetter("abcd") should return "d"