11.3.2 Simple JavaScript Functions

Even though Python and JavaScript are logically similar, it's becoming more and more apparent to Dana that they have many syntactical differences. It's a lot to take in, so Dana wants more practice creating and calling functions. She wants to know if arguments are passed in the same way, as well as how functions are called.

Arguments can be passed into both Python and JavaScript functions. Let's take a look at another Python function as an example. Look at the following code:

```
# Takes two numbers and adds them
def addition(a, b):
    return a + b
```

In this function, we've added the ability to input two numbers and add them. Let's convert this same function to JavaScript in the DevTools console. Type the following on a new line in your console:

```
// Takes two numbers and adds them
function addition(a, b) {
  return a + b;
}
```

To test the new function, type `console.log(addition(4, 5));`. This is the equivalent of using a print statement in Python to print the function. Like Python, we can condense the code even further by typing only `addition(4, 5);` to execute the function as well.

IMPORTANT

> In the "addition" function created above, the items within the parentheses are referred to as **parameters**. For example:
>
> `function addition(a, b) { return a + b; }`
>
> In this function, data points `a` and `b` are the parameters. Think of them as placeholders for the values we will add later, such as 4 and 5.
>
> Functions in JavaScript can have any number of parameters. However, from a practical standpoint, it's not a good idea to have more than two parameters per function. Too many arguments can significantly slow down and even crash your code.

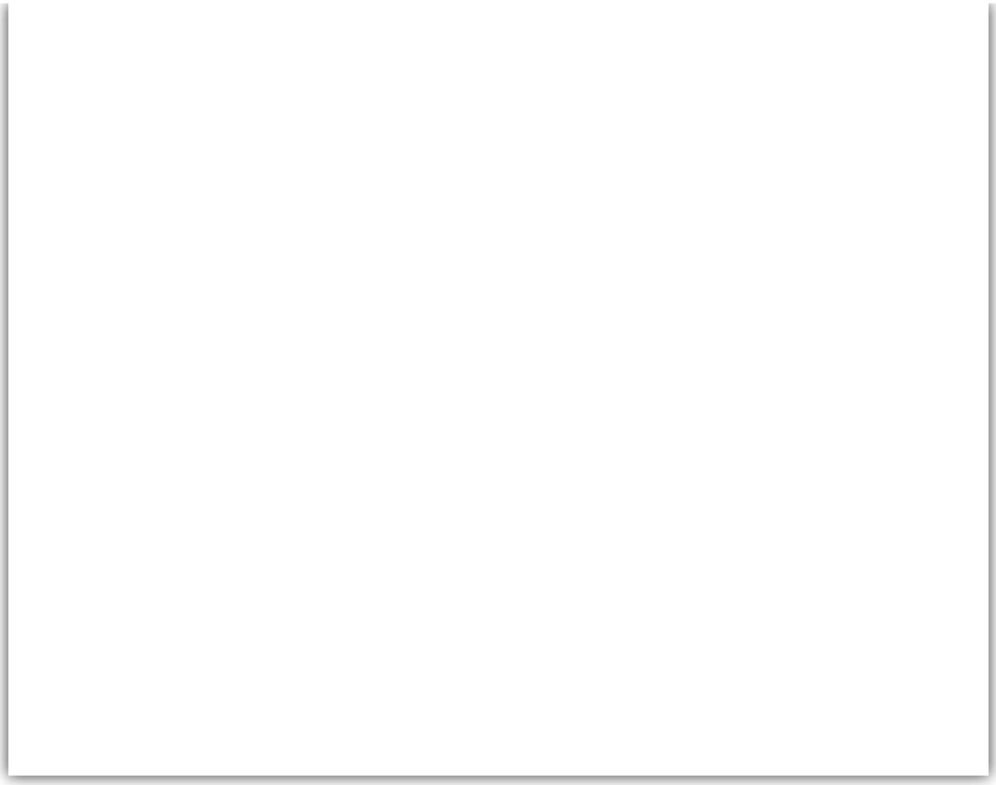
Now practice functions in the following Skill Drill.

SKILL DRILL

Practice executing the addition function in your console. Try switching up the numbers and printing it with and without the use of `console.log();`.

Functions are a versatile tool in any coding language, and JavaScript is no different. Functions can also call other functions. The code below creates a new function that includes our simple function within it. In your console, type the `doubleAddition` function below:

```
// Functions can call other functions
function doubleAddition(c, d) {
  var total = addition(c, d) * 2;
  return total;
}
```



Let's run the new function, `doubleAddition`, with the same figures we used earlier: 4 and 5. Within this function, we're calling our original function (`addition`) and multiplying the sum of 4 and 5 by 2. We've assigned a variable to the function we've already created, so that we can print the total using a return statement.

So far, we've created a function that performs simple addition and a second function that calls our original function, which is a great introduction to JavaScript functions.

NOTE

> If the code and output in your console is getting cluttered, type `clear()` and press Enter to clear the working area of your console.
>
> Once cleared, you won't be able to see the code anymore, but you can still access what you've written by using the up arrow key on your keyboard. This allows you to cycle through the different lines of code you've already executed.

Like functions in Python, JavaScript functions are very versatile and can incorporate many other actions, such as incorporating `for` loops, which we will explore shortly.

First, let's explore one of the key improvements to JavaScript functions introduced by ES6: arrow functions.