**What are functions?**

When first learning how to calculate the area of a rectangle, there’s a sequence of steps to calculate the correct answer:

1. Measure the width of the rectangle.
2. Measure the height of the rectangle.
3. Multiply the width and height of the rectangle.

With practice, you can calculate the area of the rectangle without being instructed with these three steps every time.

We can calculate the area of one rectangle with the following code:

const width = 10; const height = 6; const area = width \* height; console.log(area); // Output: 60

Imagine being asked to calculate the area of three different rectangles:

// Area of the first rectangle const width1 = 10; const height1 = 6; const area1 = width1 \* height1; // Area of the second rectangle const width2 = 4; const height2 = 9; const area2 = width2 \* height2; // Area of the third rectangle const width3 = 10; const height3 = 10; const area3 = width3 \* height3;

In programming, we often use code to perform a specific task multiple times. Instead of rewriting the same code, we can group a block of code together and associate it with one task, then we can reuse that block of code whenever we need to perform the task again. We achieve this by creating a *function*. A function is a reusable block of code that groups together a sequence of statements to perform a specific task.

In this lesson, you will learn how to create and use functions, and how they can be used to create clearer and more concise code.

**Function Declarations**

In JavaScript, there are many ways to create a function. One way to create a function is by using a *function declaration*. Just like how a variable declaration binds a value to a variable name, a function declaration binds a function to a name, or an *identifier*. Take a look at the anatomy of a function declaration below:

A function declaration consists of:

* The function keyword.
* The name of the function, or its identifier, followed by parentheses.
* A function body, or the block of statements required to perform a specific task, enclosed in the function’s curly brackets, { }.

A function declaration is a function that is bound to an identifier, or name. In the next exercise we'll go over how to run the code inside the function body.

We should also be aware of the *hoisting* feature in JavaScript which allows access to function declarations before they're defined.

Take a look at example of hoisting:

console.log(greetWorld()); // Output: Hello, World! function greetWorld() { console.log('Hello, World!'); }

Notice how hoisting allowed greetWorld() to be called before the greetWorld() function was defined! Since hoisting isn't considered good practice, we simply want you to be aware of this feature.

If you want to read more about hoisting, check out [MDN documentation on hoisting](https://developer.mozilla.org/en-US/docs/Glossary/Hoisting).

# Calling a Function

As we saw in previous exercises, a function declaration binds a function to an identifier.

However, a function declaration does not ask the code inside the function body to run, it just declares the existence of the function. The code inside a function body runs, or executes, only when the function is called. To call a function in your code, you type the function name followed by parentheses.

This function call executes the function body, or all of the statements between the curly braces in the function declaration.

We can call the same function as many times as needed.

Let’s practice calling functions in our code.

# Parameters and Arguments

So far, the functions we've created execute a task without an input. However, some functions can take inputs and use the inputs to perform a task. When declaring a function, we can specify its parameters. Parameters allow functions to accept input(s) and perform a task using the input(s). We use parameters as placeholders for information that will be passed to the function when it is called.

Let’s observe how to specify parameters in our function declaration:

In the diagram above, calculateArea(), computes the area of a rectangle, based on two inputs, width and height. The parameters are specified between the parenthesis as widthand height, and inside the function body, they act just like regular variables. width and heightact as placeholders for values that will be multiplied together.

When calling a function that has parameters, we specify the values in the parentheses that follow the function name. The values that are passed to the function when it is called are called arguments. Arguments can be passed to the function as values or variables.

In the function call above, the number 10 is passed as the width and 6 is passed as height. Notice that the order in which arguments are passed and assigned follows the order that the parameters are declared.

The variables rectWidth and rectHeight are initialized with the values for the height and width of a rectangle before being used in the function call.

By using parameters, calculateArea() can be reused to compute the area of any rectangle! Functions are a powerful tool in computer programming so let’s practice creating and calling functions with parameters.

# Default Parameters

One of the features added in ES6 is the ability to use default parameters. Default parameters allow parameters to have a predetermined value in case there is no argument passed into the function or if the argument is undefined when called.

Take a look at the code snippet below that uses a default parameter below:

function greeting (name = 'stranger') { console.log(`Hello, ${name}!`) } greeting('Nick') // Output: Hello, Nick! greeting() // Output: Hello, stranger!

* In the example above, we used the =operator to assign the parameter name a default value of 'stranger'. This is useful to have in case we ever want to include a non-personalized default greeting!
* When the code calls greeting('Nick') the value of the argument is passed in and, 'Nick', will override the default parameter of 'stranger' to log 'Hello, Nick!' to the console.
* When there isn't an argument passed into greeting(), the default value of 'stranger'is used, and 'Hello, stranger!' is logged to the console.

By using a default parameter, we account for situations when an argument isn't passed into a function that is expecting an argument.

Let’s practice creating functions that use default parameters.