How to declare a function

Functions allow you to package up lines of code that you can use (and often reuse) in your programs.

Sometimes they take **parameters** like the pizza button from the beginning of this lesson. `reheatPizza()` had one parameter: the number of slices.

```
function reheatPizza(numSlices) {
  // code that figures out reheat settings!
}
```

The `reverseString()` function that you saw also had one parameter: the string to be reversed.

```
function reverseString(reverseMe) {
  // code to reverse a string!
}
```

In both cases, the parameter is listed as a variable after the function name, inside the parentheses. And, if there were multiple parameters, you would just separate them with commas.

```
function doubleGreeting(name, otherName) {
  // code to greet two people!
}
```

But, you can also have functions that don't have any parameters. Instead, they just package up some code and perform some task. In this case, you would just leave the parentheses empty. Take this one for example. Here's a simple function that just prints out `"Hello!"`.

```
// accepts no parameters! parentheses are empty
function sayHello() {
  var message = "Hello!"
  console.log(message);
}
```

If you tried pasting any of the functions above into the JavaScript console, you probably didn't notice much happen. In fact, you probably saw `undefined` returned back to you. `undefined` is the default return value on the console when nothing is *explicitly* returned using the special `return` keyword.

Return statements

In the `sayHello()` function above, a value is **printed** to the console with `console.log`, but not explicitly returned with a **return statement**. You can write a return statement by using the `return` keyword followed by the expression or value that you want to return.

```
// declares the sayHello function
function sayHello() {
  var message = "Hello!"
  return message; // returns value instead of printing it
}
```

How to *run* a function

Now, to get your function to *do something*, you have to **invoke** or **call** the function using the function name, followed by parentheses with any **arguments** that are passed into it. Functions are like machines. You can build the machine, but it won't do anything unless you also turn it on. Here's how you would call the `sayHello()` function from before, and then use the return value to print to the console:

```
// declares the sayHello function
function sayHello() {
  var message = "Hello!"
  return message; // returns value instead of printing it
}

// function returns "Hello!" and console.log prints the return value
console.log(sayHello());
```

Parameters vs. Arguments

At first, it can be a bit tricky to know when something is either a parameter or an argument. The key difference is in where they show up in the code. A **parameter** is always going to be a *variable* name and appears in the function declaration. On the other hand, an **argument** is always going to be a *value* (i.e. any of the JavaScript data types - a number, a string, a boolean, etc.) and will always appear in the code when the function is called or invoked.

Try declaring and calling some functions on your own:

QUESTION 1 OF 2

Use the following function to answer this question.

```
function findAverage(x, y) {
  var answer = (x + y) / 2;
  return answer;
}

var avg = findAverage(5, 9);
```

What value will be stored in the variable `avg`?

- "answer"
- (x + y) / 2
- 7
- 14
- 4

SUBMIT

QUESTION 2 OF 2

```
function findAverage(x, y) {
  var answer = (x + y) / 2;
  return answer;
}

var avg = findAverage(5, 9);
```

Are `x` and `y` parameters or arguments for this function?

- Parameters
- Arguments

SUBMIT

NEXT