Function Basics

Foundations Course

Introduction

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.

Lesson overview

This section contains a general overview of topics that you will learn in this lesson.

- How to define and invoke different kinds of functions.
- How to use the return value.
- What function scope is.

Functions

Let's discuss parameters and arguments in the context of the following example function:

```
function favoriteAnimal(animal) {
    return animal + " is my favorite animal!"
}

const message = favoriteAnimal('Goat')
console.log(message)
```

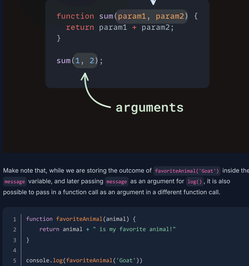
In JavaScript, parameters are the items listed between the parentheses `()` in the function declaration. Function arguments are the actual values we decide to pass to the function.

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 declaration, 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?

In the fifth line, `favoriteAnimal('Goat')` is where we are calling our `favoriteAnimal` function and passing the value `'Goat'` inside that function call. 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.

Here is a diagram to help you visualize how parameters are passed to a function, and how values get returned from it.



Make note that, while we are storing the outcome of `favoriteAnimal('Goat')` inside the `message` variable, and later passing `message` as an argument for `log()`, it is also possible to pass in a function call as an argument in a different function call.

```
function favoriteAnimal(animal) {
    return animal + " is my favorite animal!"
}

console.log(favoriteAnimal('Goat'))
```

In the example above, we pass in `favoriteAnimal('Goat')` as an argument in `log()`. When doing so, the return value of the `favoriteAnimal('Goat')` function call (the string `'Goat is my favorite animal!'`) is used as the argument for `log()`, without first storing it in a separate variable.

Keep this possibility in mind because you'll be passing in function calls as arguments somewhat often. If we just called `favoriteAnimal('Goat')` on its own without passing it as an argument for `console.log` to print it's return value, it would still return that string... to nowhere. Nothing would appear in the console, nor would the string get stored in a variable.

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?

Assignment

> **MDN exercises**
>
> Note that the assigned MDN articles also have their own exercises attached, which you should **not** do, as they involve knowledge we haven't touched yet.

1. Read the function basics article 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 "Functions == Comments" section, as well as the last "task" at the end of this lesson uses loops, which you will learn about in the next lesson. Don't worry about them.)
2. Read this MDN article on functions. Don't worry as there may be some functions that can be beyond the reach of this particular lesson, but do 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 upfront.
3. Read this article about return values.
4. Now, read the function expressions article about functions in JavaScript to give you a little more context, and read the article on arrow functions for an introduction to arrow functions. 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.
5. Finally, learn about the JavaScript Call Stack. The article goes in-depth 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 `return`ed values are going. This doubles as a bit of early computer science as well.

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 our instructions on how to run JavaScript code.

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.
   - `add7(10)` should return `17`
2. Write a function called `multiply` that takes 2 numbers and returns their product.
   - `multiply(3, 2)` should return `6`
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.
   - `capitalize("abcd")` should return `"Abcd"`
   - `capitalize("ABCD")` should return `"Abcd"`
   - `capitalize("aBcD")` should return `"Abcd"`
4. Write a function called `lastLetter` that takes a string and returns the very last letter of that string.
   - `lastLetter("abcd")` should return `"d"`

Knowledge check

The following questions are an opportunity to reflect on key topics in this lesson. If you can't answer a question, click on it to review the material, but keep in mind you are not expected to memorize or master this knowledge.

- What are functions useful for?
- How do you invoke a function?
- What are anonymous functions?
- What is function scope?
- What are return values?
- What are arrow functions?
- What is the difference between a function declaration and a function expression?

Additional resources

This section contains helpful links to related content. It isn't required, so consider it supplemental.

- What's the difference between using "let" and "var"? - stackoverflow
- How JavaScript Code is executed?