Conceptos básicos de funciones

Curso de Fundamentos

Introducción

Las cosas están a punto de ponerse *realmente* interesantes. Hasta ahora has escrito una cantidad impresionante de código para resolver varios problemas, pero ese código no ha sido tan útil como podría ser.

Imagina tomar uno de tus scripts y agruparlo en un pequeño paquete que puedas usar una y otra vez sin tener que reescribir ni cambiar el código. Ese es el poder de las funciones, que se usan *constantemente* en JavaScript.

Resumen de la lección

Esta sección contiene una descripción general de los temas que aprenderá en esta lección.

- Cómo definir e invocar diferentes tipos de funciones.
- Cómo utilizar el valor de retorno.
- ¿Qué es el ámbito de función?

Funciones

Analicemos los parámetros y argumentos en el contexto de la siguiente función de ejemplo:

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

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

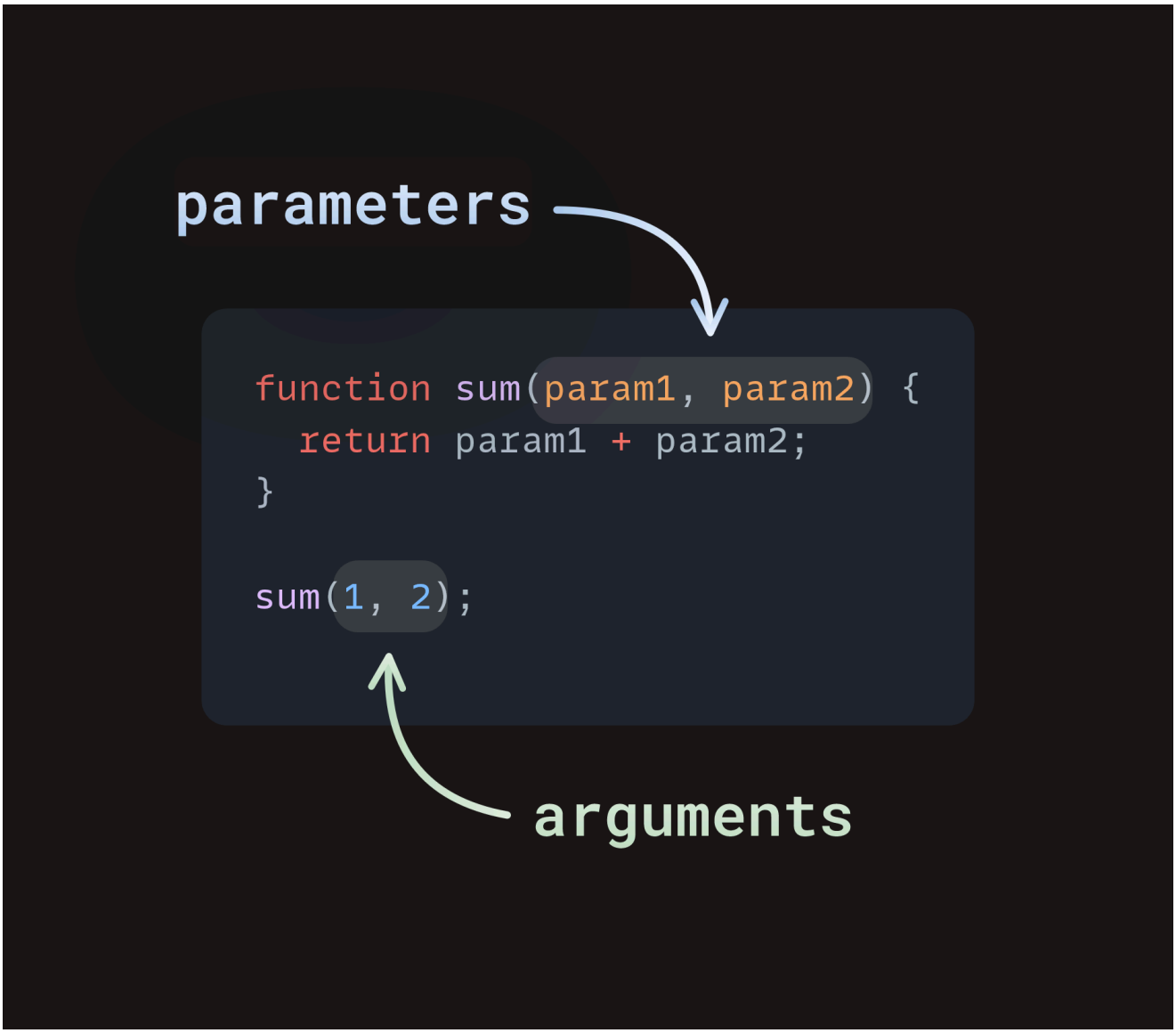
En JavaScript, los parámetros son los elementos que se enumeran entre paréntesis `()` en la declaración de la función. Los argumentos de la función son los valores reales que decidimos pasar a la función.

En el ejemplo anterior, la definición de la función se escribe en la primera línea: `function favoriteAnimal(animal)` . El parámetro, `animal` , se encuentra dentro de los paréntesis. Podríamos reemplazarlo fácilmente `animal` por `pet` , `x` o `blah` . Pero en este caso, nombrar el parámetro `animal` le da a alguien que lee nuestro código un poco de contexto para que no tenga que adivinar qué `animal` puede contener eventualmente.

Al poner `animal` dentro de los paréntesis de la `favoriteAnimal()` función, le estamos diciendo a JavaScript que le enviaremos *un* valor a nuestra `favoriteAnimal` función. Esto significa que `animal` es solo un **marcador de posición** para un valor futuro. Pero ¿qué valor estamos enviando?

La última línea, `favoriteAnimal('Goat')` , es donde llamamos a nuestra `favoriteAnimal` función y pasamos el valor `'Goat'` dentro de esa función. Aquí, `'Goat'` está nuestro argumento. Le decimos a la `favoriteAnimal` función, “Envía `'Goat'` a la función favoriteAnimal y úsalo `'Goat'` donde esté el marcador de posición 'animal'”. Debido a la flexibilidad que proporciona el uso de un parámetro, podemos declarar cualquier animal como nuestro favorito.

Aquí hay un diagrama para ayudarle a visualizar cómo se pasan los parámetros a una función y cómo se devuelven los valores desde ella.



Tenga en cuenta que al llamar `favoriteAnimal()` dentro de of `console.log()` con el argumento `'Goat'` obtenemos el valor de retorno de la función, cadena of `"Goat is my favorite animal!"`, impreso en la consola. Estamos pasando una llamada de función `favoriteAnimal('Goat')` como argumento en una llamada de función diferente - `log()`.

Tenga en cuenta esta posibilidad porque pasará llamadas de función como argumentos con cierta frecuencia. Si simplemente llamamos a la función sin usar `console.log` para imprimir su valor de retorno, no aparecería nada en la consola , pero aun así la función devolvería esa cadena.

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

> Note that articles #1 and #2 also have their own exercises attached, which you should **not** do, as they involve knowledge we haven't touched yet.

1. This MDN article on functions is a good place to start. 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.
2. Read this article about return values.
3. Next, 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.)
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.
2. Write a function called `multiply` that takes 2 numbers and returns their product.
3. Escriba una función llamada `capitalize` que tome una cadena y la devuelva con *solo* la primera letra en mayúscula. Asegúrese de que pueda aceptar cadenas en minúscula, MAYÚSCULAS o ambas.
4. Escriba una función llamada `lastLetter` que tome una cadena y devuelva la última letra de esa cadena:
    - `lastLetter("abcd")` Debería regresar `"d"`

Comprobación de conocimientos

Las siguientes preguntas son una oportunidad para reflexionar sobre temas clave de esta lección. Si no puede responder una pregunta, haga clic en ella para revisar el material, pero tenga en cuenta que no se espera que memorice o domine este conocimiento.

- ¿Para qué sirven las funciones?
- ¿Cómo se invoca una función?
- ¿Qué son las funciones anónimas?
- ¿Qué es el alcance de la función?
- ¿Qué son los valores de retorno?
- ¿Qué son las funciones de flecha?
- ¿Cuál es la diferencia entre una declaración de función y una expresión de función?

Recursos adicionales

Esta sección contiene enlaces útiles a contenido relacionado. No es obligatoria, por lo que se la puede considerar complementaria.

- ¿Cuál es la diferencia entre usar “let” y “var”? - stackoverflow
- ¿Cómo se ejecuta el código JavaScript?

Mejorar en GitHub Informar un problema Ver registro de cambios de la lección

Ver curso

Marcar como completado

Próxima lección

¡Apóyanos!

El Proyecto Odin está financiado por la comunidad. ¡Únase a nosotros para ayudar a estudiantes de todo el mundo apoyando el Proyecto Odin!

Más información Dona ahora

THE ODIN PROJECT

Educación en codificación de alta calidad mantenida por una comunidad de código abierto.

Sobre nosotros Guías

Acerca de

Equipo

Blog

Casos de éxito

Apoyo

Preguntas frecuentes

Contribuir

Contáctenos

Guías de la comunidad

Guías de instalación

Legal

Términos

Privacidad