EDUFY

Functions, part 1: make one, call it, get an answer back

Maximilian Bahrami · 10 November 2026

Make a function and call it, give it values with parameters, get an answer back with return, and know when to use console.log and when to use return.

A function is a piece of code with a name, that you can run whenever you want, as many times as you want. Instead of writing the same lines again and again, you write them once and call them by name. By the end of this lesson you'll be able to make a function, run it, give it information to work with, and get an answer back.

Make a function, then call it

Think of a function as a recipe. Writing the recipe down doesn't cook anything. You have to **make** it and then **call** it:

```
// 1. Make the function
function sayHello() {
console.log("Hello!");
console.log("Welcome to EDUFY");
}

// 2. Call it (run it)
sayHello();
sayHello();
```

- function tells JavaScript you're making a function.
- sayHello is its name.
- The code between { and } is what it does.
- sayHello() runs it. The brackets () are what make it run.

Making the function prints nothing. Each call runs the code once, so you see the two lines twice.

Give it information: parameters

Most functions need some information to work with. You put a name for it between the brackets. That name is called a **parameter**:

```
function greet(name) {
console.log("Hello, " + name + "!");
```

```
}

greet("Sara");
greet("Omar");
greet("Lina");
```

name works like a variable that only exists inside the function. Each time you call greet, the value you put in the brackets ("Sara", "Omar"...) goes into name.

A function can take more than one value. Separate them with commas, and give them **in the same order**:

```
function introduce(name, age) {
console.log(name + " is " + age + " years old");
}

introduce("Sara", 16);
introduce("Omar", 17);
```

Get an answer back: return

console.log only **shows** a value. Often you want the function to **give back** an answer that the rest of your code can use. That's what return does:

```
function add(a, b) {
return a + b;
}

const result = add(2, 3);
console.log(result); // 5
console.log(add(10, 20)); // 30
console.log(add(1, 1) * 2); // 4
```

When the function reaches return, it stops and hands the value back to where it was called. You can store the answer in a variable, print it, or use it in more maths.

console.log or return?

> console.log is for **you**, to see what's happening. return is for **your code**, to use the answer. A function that calculates something should almost always return it.

double(n) returns n * 2. What does console.log(double(4) + 1) print?

> • 8
> • ✓ 9
> • 10
> • nothing

> double(4) returns 8. Then 8 + 1 is 9, and that's what gets printed.

A real example: prices

Here is a function you could find in a real shop. It works out the total for one product:

```
function getTotal(price, quantity) {
return price * quantity;
}

const shirts = getTotal(20, 3);
const socks = getTotal(5, 4);

console.log("Shirts: " + shirts + " €"); // 60 €
console.log("Socks: " + socks + " €"); // 20 €
console.log("All: " + (shirts + socks) + " €");
```

If the price rule ever changes (a discount, for example), you change it **in one place**, and every call uses the new rule.

Give functions good names

A function **does** something, so start its name with a verb: getTotal, showMessage, addItem, isValid. When you read getTotal(20, 3), you already know what happens without opening the function.

Common mistakes

Forgetting the brackets

Wrong

```
sayHello; // nothing happens
```

Right

```
sayHello(); // runs it
```

Without (), you only mention the function. You don't run it.

Printing instead of returning

Wrong

```
function add(a, b) {
console.log(a + b);
}
const result = add(2, 3);
console.log(result); // undefined
```

Right

```
function add(a, b) {
return a + b;
}
const result = add(2, 3);
console.log(result); // 5
```

A function without return gives back undefined: "nothing". If you need the answer later, return it.

Writing code after return

Wrong

```
function add(a, b) {
return a + b;
console.log("Done"); // never runs
}
```

Right

```
function add(a, b) {
console.log("Done");
return a + b;
}
```

return ends the function right away. Anything below it never runs.

Exercise

Celsius to Fahrenheit

Write a function toFahrenheit(celsius) that **returns** the temperature in Fahrenheit. The formula is celsius * 1.8 + 32.

Then print the result for 0, 20 and 100. You should get 32, 68 and 212.

Answer

```
function toFahrenheit(celsius) {
return celsius * 1.8 + 32;
}

console.log(toFahrenheit(0)); // 32
console.log(toFahrenheit(20)); // 68
console.log(toFahrenheit(100)); // 212
```

The function gets the temperature in celsius, works out the answer and returns it. console.log outside the function shows it.

Key points

- A function is code with a name. Make it with function name() { ... }, run it with name().
- Parameters are the values a function gets, written between the brackets: greet(name).
- return gives an answer back and ends the function. Without it you get undefined.
- console.log shows a value; return lets your code use it.
- Name functions with a verb: getTotal, showMessage.

In the next lesson you'll use functions on a real web page: with buttons, with the short arrow => way of writing them, and with default values.