EDUFY

Functions, part 2: buttons, arrow functions and default values

Maximilian Bahrami · 10 November 2026

Run a function when a button is clicked, write short arrow functions with =>, give parameters default values, and learn which variables a function can see.

In the last lesson you made functions, gave them parameters and got answers back with return. Now you'll use them where frontend developers use them every day: on a web page. By the end of this lesson you'll be able to run a function when a button is clicked, write the short arrow => version of a function, give parameters default values, and know which variables a function can see.

A function for a button

To run a function when something is clicked, give its **name** to addEventListener. Press **Run**, then click the button:

```
<!DOCTYPE html>
<html lang="en">
<body style="padding: 16px; font-family: sans-serif">
<button id="hello">Say hello</button>
<p id="message"></p>
<script>
const button = document.getElementById("hello");
const message = document.getElementById("message");

function showHello() {
message.textContent = "Hello!";
console.log("The button was clicked");
}

button.addEventListener("click", showHello);
</script>
</body>
</html>
```

Look closely: it's showHello, **without** (). You're not running the function now. You're handing it to the button and saying: run this later, when someone clicks.

Which line runs save only when the button is clicked?

- button.addEventListener("click", save());
- ✓ button.addEventListener("click", save);

> • button.addEventListener("click", "save");

> save without brackets hands the function to the button for later. save() runs it **right away**, once, when the page loads. "save" is just text.

Arrow functions: the short way

You've already seen () => { ... } in earlier lessons. It's an **arrow function**: a shorter way to write a function. These three do exactly the same:

```
// 1. The normal way
function double(n) {
return n * 2;
}

// 2. Arrow function
const doubleArrow = (n) => {
return n * 2;
};

// 3. Arrow function on one line
const doubleShort = (n) => n * 2;

console.log(double(5)); // 10
console.log(doubleArrow(5)); // 10
console.log(doubleShort(5)); // 10
```

In the one-line version there are no { } and no return: the value after => is returned automatically.

Most often you'll see arrow functions **inside** addEventListener, when the code is short and only used there:

```
button.addEventListener("click", () => {
console.log("Clicked!");
});
```

Which one should I use?

> Both are fine. A simple habit: use function name() { ... } for functions you call by name, and an arrow () => { ... } for short code you hand to something else, like a click.

Default values

You can give a parameter a value to use when nothing is passed in. Write it with =:

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

```
greet("Sara"); // Hello, Sara!
greet(); // Hello, friend!
```

Without the default, greet() would print Hello, undefined!.

What can a function see?

A variable made **inside** a function only exists inside it. A variable made **outside** can be used inside:

```
const siteName = "EDUFY"; // outside

function welcome() {
const user = "Sara"; // inside
console.log(user + ", welcome to " + siteName);
}

welcome();
console.log(siteName); // works
console.log(user); // error: user is not defined
```

This is a good thing: every function can have its own user, total or result without mixing them up.

Putting it together: a counter

One function, changeBy, does the work for all three buttons. Press **Run** and try them:

```
<!DOCTYPE html>
<html lang="en">
<body style="padding: 16px; font-family: sans-serif">
<p id="count" style="font-size: 40px; margin: 0 0 8px">0</p>
<button id="minus">−1</button>
<button id="plus">+1</button>
<button id="reset">Reset</button>
<script>
const text = document.getElementById("count");
let count = 0;

function changeBy(amount) {
count = count + amount;
text.textContent = count;
console.log("count:", count);
}

document.getElementById("plus")
.addEventListener("click", () => changeBy(1));
document.getElementById("minus")
.addEventListener("click", () => changeBy(-1));
```

```
document.getElementById("reset")
.addEventListener("click", () => changeBy(-count));
</script>
</body>
</html>
```

changeBy needs a value, so the button can't take changeBy directly. The small arrow () => changeBy(1) is a function that calls changeBy with the right number when it's clicked.

Common mistakes

Calling the function instead of handing it over

Wrong

```
button.addEventListener("click", showHello());
```

Right

```
button.addEventListener("click", showHello);
```

With (), showHello runs once when the page loads, and clicks do nothing. If you need to pass a value, wrap it in an arrow: () => changeBy(1).

Braces without return

Wrong

```
const double = (n) => { n * 2 };
console.log(double(5)); // undefined
```

Right

```
const double = (n) => n * 2;
console.log(double(5)); // 10
```

With { }, an arrow function needs return, just like a normal one. Without braces, it returns automatically.

Using a variable from inside a function outside it

Wrong

```
function getTotal() {
const total = 50;
}
getTotal();
console.log(total); // error
```

Right

```
function getTotal() {
return 50;
}
const total = getTotal();
console.log(total); // 50
```

Variables inside a function stay there. To get a value out, return it.

Exercise

A temperature converter

Make a page with a number box, a **Convert** button and a paragraph for the answer.

1. Write toFahrenheit(celsius) from the last lesson, as an arrow function.
2. Write a function showResult() that reads the number box, calls toFahrenheit, and shows the answer in the paragraph.
3. Hand showResult to the button with addEventListener.

Tip: the value in an input box is text. Number(input.value) turns it into a number.

Answer

```
<!DOCTYPE html>
<html lang="en">
<body style="padding: 16px; font-family: sans-serif">
<input id="celsius" type="number" value="20" />
<button id="convert">Convert</button>
<p id="answer"></p>
<script>
const input = document.getElementById("celsius");
const answer = document.getElementById("answer");

const toFahrenheit = (celsius) => celsius * 1.8 + 32;

function showResult() {
const celsius = Number(input.value);
const result = toFahrenheit(celsius);
answer.textContent = celsius + " °C = " + result + " °F";
console.log(result);
}

document.getElementById("convert")
.addEventListener("click", showResult);
</script>
</body>
</html>
```

> toFahrenheit only does the maths and returns it. showResult deals with the page. Keeping those two jobs apart makes each function short and easy to reuse.

Key points

- Hand a function to a button by its name, without brackets: addEventListener("click", showHello).
- To pass a value, wrap the call in an arrow: () => changeBy(1).
- Arrow functions are a short way to write functions: (n) => n * 2 returns automatically; with { } you need return.
- Default values: function greet(name = "friend").
- Variables made inside a function stay inside it. Use return to get a value out.