ScopeTreasure Courses

JAVASCRIPT COURSES · LESSON 3.19 (SHORT) · CODE SCOPE SERIES

Timers — setTimeout + setInterval

Time as a feature: run code once later with setTimeout, again and again with setInterval, stopping with clear — and the 2 vs 2000 mistake.

Introduction

Normal JavaScript runs **now**. Line by line, at once. Timers let code run **later**. That one idea makes clocks, reminders, slideshows and auto-saving possible.

`setTimeout` runs a function **once** after a delay. `setInterval` runs it **again and again**. Both are **asynchronous** — the rest of your code does not wait for them. *(Roman Urdu: Timer code ko abhi nahi, baad mein chalata hai — aik baar ke liye setTimeout, baar baar ke liye setInterval.)*

Key points

- **`setTimeout(fn, ms)`** — run function `fn` **once**, after `ms` milliseconds. Example: `setTimeout(sayHello, 2000)` waits 2 seconds, then calls `sayHello` one time only.
- **The delay is in milliseconds** — `1000` is 1 second, `2000` is 2 seconds, `5000` is 5 seconds. This is the most common timer mistake for beginners.
  (Roman Urdu: Delay hamesha milliseconds mein hota hai — 2000 matlab 2 second, sirf 2 matlab pal bhar.)
- **`setInterval(fn, ms)`** — run `fn` again and again, every `ms` milliseconds, until you stop it. Perfect for a ticking clock or a counter that grows every second.
- **Every timer returns an id** — save it in a variable: `const id = setInterval(tick, 1000)`. Without the id, you cannot stop that timer later.
- **Stop a timer with clear** — `clearTimeout(id)` stops a `setTimeout` before it runs. `clearInterval(id)` stops a `setInterval` forever. A stop button on the page usually just calls `clearInterval(id)`.
  (Roman Urdu: Har timer ka aik id milta hai — usi id se clearTimeout ya clearInterval timer ko rokta hai.)
- **Timers are asynchronous** — code **after** the timer line runs **first**. The timer waits its turn, even if the delay is small. Never expect the lines below a timer to wait.
- **Pass the function, do not call it** — write `setTimeout(sayHello, 2000)`, not `setTimeout(sayHello(), 2000)`. With brackets, the function runs at once and its result is scheduled instead.

Code examples

Example 1 — Order of output: start, end, then the delayed line

```
console.log("start");

setTimeout(() => {
  console.log("This runs after 2 seconds");
}, 2000);

console.log("end");
```

What it does: It logs `start`, sets a 2-second timer, then logs `end` at once. The timer callback waits. Only after 2 seconds does its line run. This proves timers do not pause the lines below them.

What you will see: in the console, at once: `start` then `end`. After 2 seconds: `This runs after 2 seconds`. The order is start, end, delayed line — never start, delayed line, end.

Example 2 — The 2 vs 2000 mistake

```
setTimeout(() => {
  console.log("Almost instant - 2 milliseconds");
}, 2);

setTimeout(() => {
  console.log("After 2 seconds - 2000 milliseconds");
}, 2000);
```

What it does: It sets two timers side by side. The first delay is `2` — that is 2 thousandths of a second. The second delay is `2000` — that is a full 2 seconds. The numbers look close, the wait is not.

What you will see: `Almost instant - 2 milliseconds` appears almost immediately, so fast you cannot feel the wait. A full 2 seconds later, `After 2 seconds - 2000 milliseconds` appears. Writing `2` when you meant 2 seconds is the signature timer mistake.

Example 3 — A counter with setInterval and clearInterval

```
let count = 0;

const id = setInterval(() => {
  count = count + 1;
  console.log(count);

  if (count === 5) {
    clearInterval(id);
    console.log("Stopped");
  }
}, 1000);
```

What it does: Every 1000 milliseconds (1 second) the counter grows by 1 and logs itself. The timer id is saved in `id`. When the count reaches 5, `clearInterval(id)` stops the timer — the same job a stop button does when a user clicks it.

What you will see: after 1 second: `1`. After 2 seconds: `2`. After 3 seconds: `3`. After 4 seconds: `4`. After 5 seconds: `5` then `Stopped`. After that, nothing more — the interval is dead.

Quick recap

- **`setTimeout(fn, ms)`** runs once later; **`setInterval(fn, ms)`** runs again and again; the delay is always in **milliseconds**.
- Save the timer **id**, then stop it with **`clearTimeout(id)`** or **`clearInterval(id)`**.
- Timers are **asynchronous** — code after the timer line runs first, the callback runs when its time comes.

Common mistakes

1. **Writing 2 when you mean 2 seconds** — `setTimeout(fn, 2)` waits only 2 milliseconds, not 2 seconds. The fix: always think in milliseconds — 2 seconds is `2000`.
2. **Calling the function instead of passing it** — `setTimeout(sayHello(), 2000)` runs `sayHello` at once. The fix: pass the name or an arrow function without brackets — `setTimeout(sayHello, 2000)`.
3. **Forgetting to save the id** — if you write `setInterval(tick, 1000)` with no variable, no stop button in the world can stop it. The fix: `const id = setInterval(tick, 1000)`, then `clearInterval(id)`.

Practice

1. Write a program that logs `Get ready...` at once, then logs `Go!` after 3 seconds using `setTimeout`. Predict the order of the lines before you run it, then check.
2. Build a counter with `setInterval` that logs 1 to 10, one number every second, and stops itself at 10 with `clearInterval`. Then change the delay to `500` and feel the speed double.

Practice quiz

1. What is the difference between setTimeout and setInterval?

- There is none
- setTimeout runs a function once after a delay, setInterval runs it again and again every delay until stopped
- setTimeout repeats, setInterval runs once
- setTimeout is for seconds, setInterval for milliseconds

Answer: setTimeout runs a function once after a delay, setInterval runs it again and again every delay until stopped

2. How many milliseconds is a 2-second delay?

- 2
- 20
- 200
- 2000

Answer: 2000

3. How do you stop a timer?

- You cannot stop a timer once started
- Save the id it returns, then call clearTimeout(id) or clearInterval(id)
- Delete the function
- Refresh is the only way

Answer: Save the id it returns, then call clearTimeout(id) or clearInterval(id)

Watch the Code Scope channel: https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

Read online: https://scopetreasure.com/notes/js-timers · More free tools and notes at https://scopetreasure.com