**ScopeTreasure Courses**

JAVASCRIPT COURSES · LESSON 3.10 · CODE SCOPE SERIES

# Destructuring & Spread

Unpack values in one line: object and array destructuring, defaults, renaming, swaps, the ... spread copy, joining arrays and rest.

## Introduction

Pulling values out of an object used to take one line per value: `chai.name`, `chai.price`, `chai.stock` — three lines for three values. **Destructuring** does the same job in **one line**: you write the shape you want, and JavaScript fills it from the object or array. In this lesson you will also meet its partner, the **spread** operator `...`, which pours the contents of one box into a brand-new box. *(Roman Urdu: Destructuring matlab dabba khol kar cheezein aik hi line mein nikaal lena.)*

## Key points

- **Object destructuring:** `const { name, price } = chai;` creates variables called `name` and `price`, filled from the keys with the **same names**. The rule is simple: **variable name = key name**.
  *(Roman Urdu: Variable ka naam wahi rakho jo object ki key hai, warna value nahi milegi.)*
- **Renaming:** if that name is already taken, rename while you unpack: `const { price: rate } = chai;` — the value of key `price` now lives in a variable called `rate`.
- **Default value = safety net:** `const { sugar = 10 } = chai;` — if the key is missing, you get the default `10` instead of `undefined`. If the key exists, the real value wins.
- **Array destructuring works by position**, not by name: `const [first, second] = menu;`. Skip an item with an empty comma: `const [first, , third] = menu;`.
- **The famous swap:** `[a, b] = [b, a];` swaps two values with **no third box** (no temporary variable) — remember this trick, it comes up everywhere.
- **Spread copies:** `const copy = { ...chai };` pours every key of `chai` into a **new** object. Change the copy and the original stays safe. Spread also joins arrays: `[...hot, ...cold]`.
- **Rest gathers the leftovers:** `const [first, ...rest] = menu;` — `rest` is a new array with everything that is left. In objects, `const { name, ...rest } = chai;` works the same way.
  *(Roman Urdu: Spread phailata hai, rest samet-ta hai — dono ke nishan teen dot hain.)*
- **React glimpse:** in the React series you will see functions that receive their props already unpacked, like `function Card({ name, price })` — that is just object destructuring in the parameter list.

## Code examples

### Example 1 — Object destructuring in one line

```
const chai = { name: "Masala Chai", price: 250, stock: 12 };

const { name, price, stock } = chai;
console.log(name);
console.log(price);
console.log(stock);

const { price: rate, discount = 10 } = chai;
console.log(rate);
console.log(discount);
```

What it does: The first destructuring pulls `name`, `price` and `stock` out in one line, because each variable name matches a key. The second line renames `price` to `rate` (in case the name `price` is already used), and asks for `discount` with a default of `10` — `chai` has no `discount` key, so the safety net catches it.

**What you will see:** the console prints, in order: `Masala Chai`, then `250`, then `12`, then `250` (the renamed `rate`), then `10` (the default, not `undefined`).

### Example 2 — Array destructuring and the no-third-box swap

```
const menu = ["Chai", "Coffee", "Lassi"];

const [first, , third] = menu;
console.log(first);
console.log(third);

let cupA = "green tea";
let cupB = "masala chai";
[cupA, cupB] = [cupB, cupA];
console.log(cupA);
console.log(cupB);
```

What it does: Arrays unpack by **position**: `first` gets index 0, the empty space skips index 1, and `third` gets index 2. Then the swap trick trades the contents of `cupA` and `cupB` in a single line — no temporary third variable is created anywhere.

**What you will see:** the console prints `Chai`, then `Lassi`, then `masala chai`, then `green tea` — the two cups have changed places.

**Example 3 — Spread: a safe copy, joined arrays and rest**

```
const chai = { name: "Masala Chai", price: 250, stock: 12 };

const copy = { ...chai, price: 300 };
console.log(copy.price);
console.log(chai.price);

const hot = ["Chai", "Coffee"];
const cold = ["Lassi", "Juice"];
const all = [...hot, ...cold];
console.log(all.join(", "));

const [firstItem, ...restItems] = all;
console.log(firstItem);
console.log(restItems.join(", "));
```

What it does: `{ ...chai, price: 300 }` builds a **new** object with all of `chai` inside it, then replaces its `price` — the original is never touched. Array spread pours `hot` and `cold` into one new array, in order. At the end, rest does the opposite job: `firstItem` takes the first drink and `...restItems` gathers everything left over.

**What you will see:** the console prints `300` (the copy's new price), then `250` (the original is safe), then `Chai, Coffee, Lassi, Juice`, then `Chai`, then `Coffee, Lassi, Juice`.

## Quick recap

- Destructuring unpacks in one line: objects by **key name**, arrays by **position**.
- Rename with `{ price: rate }`, protect missing keys with a **default** like `{ sugar = 10 }`, and swap with `[a, b] = [b, a]`.
- Spread `...` **copies and joins** safely; rest `...` **gathers what is left**.

## Common mistakes

1. **Variable name does not match the key** — `const { cost } = chai;` gives `undefined`, because `chai` has no key called `cost`. The fix: use the exact key name, or rename on purpose with `{ price: cost }`.
2. **Editing the original when you wanted a copy** — `const copy = chai;` does not copy an object, it just gives the same object a second name, so changing `copy.price` changes `chai.price` too. The fix: spread it — `const copy = { ...chai };`.
3. **Using object destructuring on an array (or the reverse)** — `{ }` reads keys, `[ ]` reads positions. Mixing them up gives `undefined` values or errors. Match the brackets to the data: curly for objects, square for arrays.

## Practice

1. Make an object `student` with keys `name`, `course` and `fees`. Unpack all three in one line and print them. Then unpack `fees` again under the new name `monthlyFees`, and add a default `discount = 0` for a key the object does not have.
2. Make two arrays, `fruits = ["Apple", "Banana"]` and `vegetables = ["Potato", "Onion"]`. Join them into one array with spread, take the first item out with destructuring, and gather the rest with `...rest`. Print all three results.

## Practice quiz

**1. What does const { name, price } = chai; do?**

- It copies the whole object under one name
- It creates variables name and price, filled from the keys with the same names
- It deletes those keys from chai
- It renames the object to name

**Answer:** It creates variables name and price, filled from the keys with the same names

**2. How do you rename while unpacking, so price lands in a variable called rate?**

- const { price = rate } = chai;
- const { price: rate } = chai;
- const { rate: price } = chai;
- const [price, rate] = chai;

**Answer:** const { price: rate } = chai;

**3. What is the difference between const copy = chai; and const copy = { ...chai };?**

- There is no difference
- copy = chai gives the same object a second name, so changing copy changes chai; spread pours every key into a new object, so the original stays safe
- Spread copies only the first key
- copy = chai is the safe copy

**Answer:** copy = chai gives the same object a second name, so changing copy changes chai; spread pours every key into a new object, so the original stays safe

---

**Watch the video lesson:** https://youtu.be/8vESIJk0BTQ

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