Variables: let, const, var

Labelled boxes for your data: naming rules, let for changing values, const with its lock, and why old var is avoided — const first.

Introduction

Your code needs to remember things: a price, a name, a count of cups. JavaScript remembers values in **variables**. Think of a variable as a **labelled box**: the label is written outside the box, and the value sits inside it. When your code says the label, JavaScript looks inside the box.

JavaScript gives you three words to make a box: `let`, `const`, and the old `var`. They are not equal. In this lesson you will learn the naming rules, see how `let` values change, feel the lock on `const`, and learn the simple decision rule professionals use: **const first**.

(Roman Urdu: Variable aik label wala dabba hai — bahar naam, andar value; const wala dabba lock ho jata hai.)

Key points

- **Declaring a variable:** `let price = 250;` makes a box labelled `price` and puts `250` inside. The `=` here means **store this value** in the box.
- **Naming rules:** a name can use letters, numbers, `_` and `$` — but it **cannot start with a number** and **cannot contain spaces**. `chaiPrice` is fine; `2price` and `chai price` are errors.
  (Roman Urdu: Naam hamesha letter se shuru hoga, number se nahi, aur beech mein space nahi aayegi.)
- **Use camelCase and meaningful names:** write `totalBill`, not `totalbill` or `tb`. A good name tells the reader what is inside the box without opening it.
- **`let` is for values that change:** declare with `let` when the value must move — like a price that rises, or a counter that grows.
- **`const` is a locked box:** declare with `const` when the value must **never change**. Trying to store a new value in a `const` box throws a **TypeError** — JavaScript stops and tells you honestly.
  (Roman Urdu: const ka matlab hai ye value ab change nahi ho sakti — koshish karoge to error milega.)
- **`var` is the old way — avoid it:** `var` leaks out of blocks and allows silent re-declaring, which creates surprising bugs. Modern code does not use it.
- **The decision rule: const first.** Use `let` only when you know the value must change. Use `var` never.
- **Name is not the value:** the label `price` is not the number `250`. The box can hold a new value later (with `let`), but the label stays the same.

Code examples

Example 1 — let: a value that changes

```
let price = 250;
console.log(price);

price = 300;
console.log(price);

let cups = 0;
cups = cups + 1;
cups = cups + 1;
cups = cups + 1;
console.log(cups);
```

What it does: The box `price` starts with `250`, then we store `300` in the same box — the old value is gone. The box `cups` starts at `0` and grows by one, three times, like a counter counting cups of chai.

What you will see: three lines in the Console: `250`, then `300`, then `3`.

Example 2 — const: the locked box and its honest error

```
const shopName = "ChaiPoint";
console.log(shopName);

shopName = "Tea House";
console.log(shopName);
```

What it does: `shopName` is declared with `const`, so its value is locked at `"ChaiPoint"`. The next line tries to store a new value — and JavaScript refuses, loudly and honestly, instead of silently breaking your program later.

What you will see: the Console prints `ChaiPoint`, then the script stops with a red error: `TypeError: Assignment to constant variable.` The last line never runs. That error is `const` protecting you.

Example 3 — Naming rules in real code

```
let chaiPrice = 250;
let numberOfCups = 3;
let totalBill = chaiPrice * numberOfCups;

console.log(totalBill);
console.log("Total: " + totalBill + " rupees");
```

What it does: Three variables with clear camelCase names. `totalBill` is calculated from the other two boxes — `250 * 3` — and printed two ways: as a pure number, and inside a sentence built with `+`.

What you will see: two lines in the Console: `750` and then `Total: 750 rupees`.

Quick recap

- A variable is a **labelled box**: name outside, value inside.
- **const first**, `let` when the value must change, `var` never.
- Names use camelCase, start with a letter, and carry no spaces.

Common mistakes

1. **Changing a const value** — storing a new value in a `const` box throws `TypeError: Assignment to constant variable.` Fix: if the value must change, declare it with `let` from the start.
2. **Bad names like `2price` or `chai price`** — names cannot start with a number or contain spaces, and the script fails. Fix: start with a letter and use camelCase, like `chaiPrice`.
3. **Using var out of habit** — `var` leaks and re-declares silently, creating bugs that are hard to find. Fix: follow the rule — `const` first, then `let`, never `var`.

Practice

1. Declare a `const` with your shop or city name, a `let` for a price, and a `let` counter starting at `0`. Increase the counter two times, change the price once, and print all three values.
2. Deliberately try to change your `const` value and read the error in the Console. Write one line in your notebook: what does the error say, and why is it helpful?

Practice quiz

1. const price = 250; then later price = 300; — what is the fix?

- Change const to let, because the value must change
- Change const to var
- Delete the second line only
- Add quotes around 300

Answer: Change const to let, because the value must change

2. 2price = 50; gives an error — what is the fix?

- Rename it to price2, because a name cannot start with a number
- Add quotes: '2price'
- Change = to ==

- Add a semicolon

Answer: Rename it to price2, because a name cannot start with a number

3. let PI = 3.14; is used for a value that never changes — what is the better fix?

- Change let to const, because the value must never change
- Change let to var
- Remove the declaration
- Change 3.14 to 3

Answer: Change let to const, because the value must never change

4. let x = 1; let x = 2; throws an error — what is the fix?

- Remove the second let and just write x = 2;
- Change both to var
- Add quotes
- Move the second line up

Answer: Remove the second let and just write x = 2;

5. What does this print?

```
let x = 10;
x = x - 4;
console.log(x);
```

- 6
- 10
- 4
- 14

Answer: 6

Watch the video lesson: https://youtu.be/hMRm-u93d0w

Read online: https://scopetreasure.com/notes/js-variables-let-const-var · More free tools and notes at https://scopetreasure.com