ScopeTreasure Courses

JAVASCRIPT COURSES · LESSON 3.12 · CODE SCOPE SERIES

Strings & Template Literals

Text done right: quotes vs backticks, ${} placeholders, calculations inside strings, length and indexes, case, search, slice and split.

Introduction

Joining text with `+` quickly becomes quote chaos: open quote, close quote, plus, variable, plus, quote again — one missing piece and the whole line breaks. **Template literals** fix this: write the text inside **backticks** (the key above Tab), and drop values straight into their slots with `${ }`. The string becomes a *living* string — it can even do maths inside itself. *(Roman Urdu: Backtick wali string zinda hoti hai — value aur hisaab seedha is ke andar aa jata hai.)*

Key points

- **Three ways to write text:** single quotes `'...'`, double quotes `"..."`, and backticks `` `...` ``. Only backticks allow ${ } placeholders and multi-line text.
- **Placeholders:** `${name}` is a slot — JavaScript puts the value of `name` in that exact place when the string is built. Any expression works, not just variables.
- **Calculations live inside:** `${price * 2}` does the maths first, then places the answer. No extra variable, no `+` signs.

  (Roman Urdu: ${ } ke andar sirf naam nahi, pura hisaab bhi likh sakte ho.)
- **Multi-line text:** inside backticks, pressing Enter is allowed — the line break becomes part of the string. No error, and no `\n` needed.
- **Strings have boxes too:** `text.length` counts the characters, and each character has an **index starting from zero** — `"Chai"[0]` is `"C"`, `"Chai"[3]` is `"i"`.
- **Case tools:** `toUpperCase()` and `toLowerCase()` return the text in one case. The search trick: lowercase **both sides** before comparing, or `"Chai"` will never match `"chai"`.
- **Search and cut:** `includes("chai")` answers `true` or `false`; `slice(0, 4)` cuts out a piece; `split(" ")` chops text into an **array** of pieces.

  (Roman Urdu: includes dhoondta hai, slice kaat kar tukra deta hai, split tukron ki list bana deta hai.)

Code examples

Example 1 — A living string with a calculation inside

```
const name = "Ahmed";
const item = "Masala Chai";
const price = 250;

const message = `Hello ${name}, your ${item} costs Rs ${price}. Two cups will cost Rs ${price
* 2}.`;
console.log(message);

const note = `Order ready.
Please collect it from the counter.`;
console.log(note);
```

What it does: Every `${ }` slot is filled when the string is built: the name, the item, the price — and the last slot even multiplies `price * 2` on the spot. The second string simply runs over two lines inside the backticks; the Enter key press is part of the text, with no `\n` anywhere.

What you will see: the console prints `Hello Ahmed, your Masala Chai costs Rs 250. Two cups will cost Rs 500.` and then the note on two separate lines: `Order ready.` and `Please collect it from the counter.`

Example 2 — Length, index and a password check

```
const drink = "Chai";
console.log(drink.length);
console.log(drink[0]);
console.log(drink[3]);

const password = "abc123";
if (password.length < 8) {
  console.log("Password too short");
} else {
  console.log("Password length OK");
}

const typed = "CHAI";
const saved = "chai";
console.log(typed.toLowerCase() === saved.toLowerCase());
console.log(saved.includes("ha"));
```

What it does: `length` counts characters and the index reads one box at a time, starting from zero — this is exactly how a real sign-up page checks a password: shorter than 8 characters means too short. The last lines

show the case trick in action: `typed` and `saved` only match after **both** are lowercased, and `includes` checks whether one piece of text sits inside another.

What you will see: the console prints `4`, then `C`, then `i`, then `Password too short` (because `abc123` has only 6 characters), then `true`, then `true`.

Example 3 — Slice, split, and a receipt from one template

```
const text = "Masala Chai";
console.log(text.slice(0, 6));
console.log(text.split(" ")[1]);

const product = { name: "Masala Chai", price: 250, qty: 2 };
const receipt = `===== RECEIPT =====
Item: ${product.name}
Qty: ${product.qty}
Total: Rs ${product.price * product.qty}`;
console.log(receipt);
```

What it does: `slice(0, 6)` cuts out the first six characters, and `split(" ")` chops the text at the space into an array, so index `1` is the second word. Then the finale: one template literal turns a product object into a full receipt card — the multi-line layout, the values and the total calculation all live inside a single string.

What you will see: the console prints `Masala`, then `Chai`, then the receipt exactly like this: a line `===== RECEIPT =====`, then `Item: Masala Chai`, then `Qty: 2`, then `Total: Rs 500`.

Quick recap

- Backticks make living strings: `${ }` holds values **and** calculations, and Enter gives you real line breaks.
- `length` counts, indexes start at **zero**, and you compare text fairly only after matching the case on both sides.
- `includes` searches, `slice` cuts a piece out, `split` turns text into an array.

Common mistakes

1. **Using normal quotes with ${ }** — `'Hello ${name}'` prints the characters `${name}` literally, because placeholders only work inside backticks. The fix: use the backtick key (above Tab) whenever a string has a slot.
2. **Expecting index 1 to be the first character** — string indexes start at **zero**, so `"Chai"[1]` is `"h"`, not `"C"`. The fix: count from 0, and remember the last index is always `length - 1`.
3. **Comparing text without matching case** — `"Chai" === "chai"` is `false`, so searches and logins fail for no visible reason. The fix: run `toLowerCase()` on **both** sides before you compare or search.

Practice

1. Make variables `student = "Fatima"`, `course = "JavaScript"` and `fees = 1500`. Build one template literal that prints a full admission slip over three lines, with the total for 3 months calculated inside a `${ }` slot.
2. Take the text `"Learning JavaScript is fun"`. Print its length, print its first and last characters, cut out the word `JavaScript` with `slice`, and use `split` to print how many words the sentence has.

Practice quiz

1. let msg = 'It's ok'; breaks on the quote — what is the fix?

- Use double quotes: "It's ok"
- Remove the apostrophe word
- Add a second single quote
- Change let to const

Answer: Use double quotes: "It's ok"

2. What does this print?

```
let s = "Hi";
console.log(s + "!");
```

- Hi!
- Hi
- !
- Error

Answer: Hi!

Watch the video lesson: https://youtu.be/7Z_LBfE3pkE

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