

Strings

Ref Javascript

Characters & length

A string is text inside quotes. With simple letters, each character has one position, its index; the first index is 0. JavaScript counts UTF-16 code units: an emoji may take two positions.

```
const word = "Script";
console.log(word.length);      // 6
console.log(word[0]);          // S
console.log(word[2]);          // r
console.log(word.at(-1));      // t (counts back from the end)
console.log(word[word.length - 1]); // t
```

- The last index is `length - 1`. `word[6]` is undefined, not an error.
- Strings are immutable: `word[0] = "s"` cannot change the string and throws a `TypeError` in strict mode. Build a new string instead.

Join strings with `+`, which is called concatenation:

```
const first = "Ada";
const last = "Lovelace";
console.log(first + " " + last); // Ada Lovelace
```

String methods

A method is a function that belongs to a value. Call it with a dot. A string method gives back a **new** value and leaves the original alone.

```
const city = " Hong Kong ";
console.log(city.trim());           // Hong Kong
console.log(city.trim().toUpperCase()); // HONG KONG
console.log(city.length);           // 12 (the spaces are still
↪ there)
```

Method	Meaning	Example	Result
<code>.toUpperCase()</code> / <code>.toLowerCase()</code>	change case	<code>"Hi".toLowerCase()</code>	<code>"hi"</code>
<code>.trim()</code>	remove spaces at both ends	<code>" hi ".trim()</code>	<code>"hi"</code>
<code>.includes(t)</code>	does it contain t?	<code>"cat".includes("a")</code>	<code>true</code>
<code>.indexOf(t)</code>	first position of t, or -1	<code>"cat".indexOf("t")</code>	<code>2</code>
<code>.slice(a, b)</code>	characters from a up to (not including) b	<code>"Script".slice(0, 3)</code>	<code>"Scr"</code>
<code>.split(sep)</code>	break into an array	<code>"a,b,c".split(",")</code>	<code>["a","b","c"]</code>
<code>.replace(a, b)</code>	swap the first a for b	<code>"cat".replace("c", "b")</code>	<code>"bat"</code>

```
const email = "mei@school.org";
const at = email.indexOf("@");
console.log(email.slice(0, at));           // mei
console.log(email.slice(at + 1));         // school.org
console.log(email.split("@"));            // ["mei","school.org"]
```

Template literals

A template literal is a string written between backticks (on most keyboards, the key left of 1). Put any expression inside `${ }` and its value is placed into the text:

```
const name = "Mei";
const age = 17;
console.log(`${name} is ${age} years old.`); // Mei is 17 years old.
console.log(`Next year: ${age + 1}`);       // Next year: 18
```

- Backticks can also hold several lines of text.
- This is easier to read than many + signs: compare `"Hi " + name + "!"` with ``Hi ${name}!``.

```
const price = 4.5;
console.log(`Two cost ${ (price * 2).toFixed(2) }`); // Two cost $9.00
```

Common mistakes

- `${ }` only works inside backticks. In `"Hi ${name}"` the text is printed exactly as typed.
- Methods do not change the string: `city.trim()` on its own line does nothing. Store the result: `const clean = city.trim();`.
- `slice(a, b)` stops **before** `b`, just like the last index is `length - 1`.