The JavaScript Cheat Sheet

What is JavaScript (and modern JS)?

JavaScript makes web pages interactive. HTML gives structure, CSS handles styling, and JavaScript responds to clicks, validates forms, fetches data, and updates the screen. It runs inside the browser. Modern JavaScript (ES2015 and later, up through ES2024) added `let`, `const`, arrow functions, modules, and async/await. This sheet uses current syntax, not old `var` and ES5 patterns.

Variables and types

Use `const` by default, `let` when a value changes. Both are block-scoped. Avoid `var` (function-scoped and hoisted).

```
const name = "Ada";   // no reassignment
let count = 0;        // reassignable
typeof name;          // "string"
```

Type	Example
string	`"hello"`
number	`42`, `3.14`
boolean	`true` / `false`
null	`null` (deliberate empty)
undefined	`undefined` (not set)
object	`{}`, `[]`
symbol	`Symbol("id")`
bigint	`10n`

Quirk: `typeof null` returns `"object"`. A historical bug, kept for compatibility.

Operators

Prefer strict `===` and `!==` over `==` and `!=`, which coerce types.

Operator	Meaning
`+ - * / % **`	Arithmetic (incl. remainder, power)
`=== !==`	Strict equal / not equal
`&& || !`	Logical and, or, not
`cond ? a : b`	Ternary (inline if/else)
`??`	Nullish (fallback for null/undefined)
`obj?.prop`	Optional chaining

```
const label = isAdult ? "Adult" : "Minor";
const port = config.port ?? 3000;
```

Strings and template literals

```
const greeting = `Hi ${name}, ${3 + 2} messages`;
"JavaScript".length;            // 10
"JavaScript".slice(0, 3);       // "Jav"
"cat,dog".includes("dog");      // true
"a-b-c".replace("-", "+");      // "a+b-c"
"a-b-c".replaceAll("-", "+");   // "a+b+c"
"a,b,c".split(",");             // ["a","b","c"]
"  hi  ".trim();                // "hi"
"loud".toUpperCase();           // "LOUD"
"5".padStart(2, "0");           // "05"
```

Numbers and Math

```
parseInt("10", 10);      // 10
parseFloat("3.14");      // 3.14
Number("42");            // 42
Math.round(4.6);         // 5
Math.floor(4.9);         // 4
Math.ceil(4.1);          // 5
Math.random();           // 0 to <1
(3.14159).toFixed(2);    // "3.14"
```

Arrays

Method	What it does
`.map(fn)`	New array, each item transformed
`.filter(fn)`	New array, items that pass a test
`.reduce((a,c)=>a+c,0)`	Fold to one value
`.forEach(fn)`	Run code per item
`.find / .some / .every`	First match / any / all
`.includes(x)`	Is a value present
`.push .pop .shift .unshift`	Add/remove ends (mutates)
`.sort .splice .reverse`	Reorder/edit in place (mutates)
`.slice .flat .concat`	Return new array (safe)

```
const doubled = nums.map(n => n * 2);
const big = nums.filter(n => n > 1);
const sum = nums.reduce((a, c) => a + c, 0);
const merged = [...a, ...b];         // spread
const [first, second] = arr;         // destructuring
arr.at(-1);                          // last item
```

Objects

```
const user = { name: "Rae", age: 30 };
user.name;  user["age"];
const { name, age } = user;            // destructuring
const updated = { ...user, age: 31 };  // spread + override
const record = { name };               // shorthand
Object.keys(user);    // ["name","age"]
Object.values(user);  // ["Rae", 30]
Object.entries(user); // [["name","Rae"], ...]
Object.hasOwn(user, "name");  // true
```

Functions and arrow functions

```
function greet(name) { return "Hi " + name; }
const double = (a) => a * 2;           // arrow
const add = (a = 1, b = 1) => a + b;   // default params
const sumAll = (...nums) => nums.reduce((a, c) => a + c, 0);
(() => { /* runs now */ })();           // IIFE
```

Arrow functions do not bind their own `this`. Do not use them as object methods or constructors.

Conditionals and loops

```
if (score > 90) { grade = "A"; } else { grade = "B"; }

switch (day) {
  case 1: label = "Mon"; break;
  default: label = "Other";
}

for (let i = 0; i < 3; i++) { }
for (const item of list) { }   // values
for (const key in obj) { }     // keys
while (running) { }
do { } while (false);
// break exits; continue skips a pass
```

The DOM

Prefer `.textContent` over `.innerHTML` for untrusted text.

```
const box = document.querySelector(".card");
const all = document.querySelectorAll("li");  // NodeList
const el = document.getElementById("main");
box.textContent = "Safe text";
box.innerHTML = "<b>careful</b>";
box.classList.add("active");
box.classList.toggle("open");
el.setAttribute("href", "/home");
el.style.color = "crimson";
const div = document.createElement("div");
box.append(div);
```

Events

```
btn.addEventListener("click", (e) => {
  console.log("clicked", e.target);
});
form.addEventListener("submit", (e) => {
  e.preventDefault();   // stop reload
});
// click, input, submit, keydown, change, mouseover
```

Promises and async/await

```
const p = new Promise((resolve, reject) => resolve("done"));
p.then(v => console.log(v)).catch(e => console.error(e));

async function load() {
  try {
    const res = await fetch("/api/data");
    if (!res.ok) throw new Error("HTTP " + res.status);
    return await res.json();
  } catch (err) { console.error(err); }
}
```

`fetch` does not reject on 404 or 500, only on network failure. Check `response.ok`.

ES Modules

```
// math.js
export const PI = 3.14;
export function square(n) { return n * n; }
export default function cube(n) { return n * n * n; }

// app.js
import cube, { PI, square } from "./math.js";
import * as math from "./math.js";
```

Handy ES2020+ features

```
user?.address?.city;          // optional chaining
value ?? "default";           // nullish fallback
count ??= 5;                  // nullish assign
arr.at(-1);                   // last element
const copy = structuredClone(obj);   // deep copy
Object.hasOwn(obj, "key");
await Promise.allSettled([a, b]);
```

Mini-recipes

```
// Fetch JSON safely
async function getUsers() {
  const res = await fetch("/api/users");
  if (!res.ok) throw new Error("Failed: " + res.status);
  return res.json();
}

// Filter then map
const names = users.filter(u => u.active).map(u => u.name);

// Debounce
function debounce(fn, ms) {
  let timer;
  return (...args) => {
    clearTimeout(timer);
    timer = setTimeout(() => fn(...args), ms);
  };
}

// Guard
const city = user?.address?.city ?? "Unknown";

// Toggle a class on click
btn.addEventListener("click", () => box.classList.toggle("active"));
```

Common gotchas

- Use `===`, not `==`. Loose equality coerces: `0 == ""` is `true`.
- Prefer `let` and `const` over `var`.
- Arrow functions have no own `this`; avoid for object methods/constructors.
- `NaN !== NaN`. Test with `Number.isNaN(x)`.
- Mutating (`sort`, `splice`, `push`, `reverse`) vs new-array (`map`, `filter`, `slice`).
- Every `async` function returns a Promise.
- `fetch` only rejects on network failure; check `response.ok`.
