JavaScript cheat sheet · Bringing web pages to life

Include Variables Text Maths Flow Functions Arrays Help

Holger Sebastião · www.sebastiao.org
Version 1.0 · 2026-10-25

Including JavaScript — external

```
<head>
  <link rel="stylesheet"
        href="css/style.css">
  <script src="js/script.js" defer></script>
</head>
```

Just like CSS: a separate file in the `js` folder. `defer` means the script only runs once the whole page has loaded. Otherwise it can't find the elements yet.

```
website/
├── index.html
├── css/style.css
└── js/script.js
```

Output and console

```
console.log("Hello world");  // console
console.log("Sum:", sum);
alert("Watch out!");         // pop-up
```

Open the **console**: **F12** or Cmd+Alt+I → **Console** tab. It also shows every error in red, with file and line.

What's different from Java?

JAVA	JAVASCRIPT
`int num = 5;`	`let num = 5;` no data type
`final`	`const`
`System.out.println`	`console.log`
`public int sum(…)`	`function sum(…)`
`==` for numbers	`===` always
`a.equals(b)`	`a === b` for text too
`int[]` fixed length	array grows with `push`
`7 / 2` → `3`	`7 / 2` → `3.5`

Same as in Java: `if`, `else`, `for`, `while`, `switch`, `{ }`, `;`, `//`, `&&`, `||`, `!`. Case matters.

Maths — Math

`Math.round(2.5)`	round → `3`
`Math.floor(2.9)`	round down → `2`
`Math.random()`	random from `0` to below `1`
`10 % 3`	remainder → `1`

Variables — let · const

```
let points = 12;        // can change
points = points + 1;
const MAX_POINTS = 15;  // fixed
const name = "Ali";
```

`let` for values that change, `const` for fixed values. JavaScript keeps track of the type itself. Old programs use `var`: don't use it any more.

Data types

TYPE	EXAMPLE
`number`	`42`, `3.5` (whole and decimal numbers)
`string`	`"Ali"` or `'Ali'`
`boolean`	`true`, `false`
`undefined`	variable without a value
`null`	deliberately "nothing"
Array	`[3, 8, 12]`
Object	`{ name: "Ali", age: 17 }`

`typeof points` → `"number"`

Text — string

```
let name = "Ali";
let greeting = "Hello " + name + "!";
let greeting2 = `Hello ${name}!`;  // same
```

`greeting.length`	number of characters
`greeting.toUpperCase()`	all capitals
`greeting.includes("Ali")`	contains? → `true`
`greeting.trim()`	remove outer spaces

Inside **backticks** `` `…` ``, `${ }` inserts the value (placeholder).

Number or text?

```
"5" + 3           // "53"  text!
Number("5") + 3   // 8
Number("abc")     // NaN: not a number
(7.456).toFixed(2)  // "7.46"
```

! Input from forms is **always text**. Convert it with `Number(…)` before calculating.

Decisions — if · else

```
if (points >= 13) {
  console.log("very good");
} else if (points >= 5) {
  console.log("passed");
} else {
  console.log("failed");
}
```

`===` `!==`	equal, not equal (value **and** type)
`<` `>` `<=` `>=`	less, greater …
`&&` `||` `!`	and, or, not

! `==` converts: `"5" == 5` is `true`. So always use `===`.

Multiple choice — switch

```
switch (day) {
  case "Sat":
  case "Sun":
    console.log("Weekend");
    break;
  default:
    console.log("School");
}
```

Loops

COUNTING LOOP

```
for (let i = 1; i <= 10; i++) {
  console.log(i);
}
```

PRE-TEST LOOP

```
let sum = 0;
while (sum < 100) {
  sum = sum + 7;
}
```

POST-TEST LOOP

```
let roll;
do {
  roll = Math.floor(Math.random() * 6) + 1;
} while (roll !== 6);
```

Functions

```
function calculateGrade(points) {
  let grade = (17 - points) / 3;
  return Math.round(grade * 10) / 10;
}

let myGrade = calculateGrade(11);  // 2
```

No return type, no parameter types. Without `return` the function returns `undefined`.

FOR READING: ARROW FUNCTION

```
const double = (num) => num * 2;
```

Short form, common on the internet. We write `function`.

Arrays

```
let grades = [12, 8, 15];
grades.push(10);          // add at the end
console.log(grades[0]);   // 12
console.log(grades.length);  // 4

for (let i = 0; i < grades.length; i++) {
  console.log(grades[i]);
}
for (let grade of grades) { // each one
  console.log(grade);
}
```

`grades.pop()`	remove the last one
`grades.includes(8)`	contained? → `true`
`grades.indexOf(15)`	position → `2`, otherwise `-1`
`grades.join(", ")`	`"12, 8, 15, 10"`

Objects

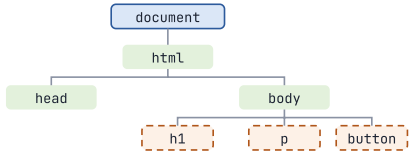
```
let student = {
  firstName: "Ali",
  points: 12
};
console.log(student.firstName);  // Ali
student.points = 13;
```

JavaScript cheat sheet · Bringing web pages to life

DOM · Change · Events · Forms · Timing · Help

Holger Sebastião · www.sebastiao.org
Version 1.0 · 2026-10-25

The DOM — the page as a tree



The browser turns the HTML into a tree: the **DOM**. JavaScript finds elements in it and changes them, and the page shows the change straight away.

Finding elements

```
<h1 id="title">Grade calculator</h1>
<p class="hint">…</p>
```

```
let title =
  document.getElementById("title");
let hint =
  document.querySelector(".hint");
```

`getElementById("title")`	the element with this id, **without** `#`
`querySelector(".hint")`	the **first** match, using a CSS selector

If nothing is found, you get `null`. Then the error `Cannot read properties of null`: check the id and add `defer`.

Saving in the browser

```
localStorage.setItem("name", "Ali");
let name = localStorage.getItem("name");
localStorage.removeItem("name");
```

Survives a reload, but only on this computer and only as text. Don't store anything secret.

New elements

```
let list =
  document.getElementById("list");
let item = document.createElement("li");
item.textContent = "Milk";
list.append(item);    // add at the end
item.remove();        // delete again
```

Changing content

```
title.textContent = "New title";
hint.innerHTML = "Now <b>bold</b>";
```

`textContent`	text only, tags are shown as text
`innerHTML`	text with HTML tags
`value`	content of an input field

Never output user input with `innerHTML`, use `textContent`: otherwise someone can inject foreign HTML and scripts.

Changing the look

```
/* style.css */
.error {
  color: #b3261e;
  font-weight: bold;
}
```

```
hint.classList.add("error");
hint.classList.remove("error");
// on if off, off if on:
hint.classList.toggle("error");
```

Better like this: the look lives in the CSS, JavaScript only switches the class.

```
title.style.color = "red";    // also works
title.style.backgroundColor = "yellow";
```

With `style`, `background-color` becomes `backgroundColor`.

Working on many elements

```
let links =
  document.querySelectorAll("nav a");
for (let link of links) {
  link.classList.add("menu");
}
```

`querySelectorAll` returns a list. `for … of` goes through every element one by one.

Reacting to clicks — events

```
<button id="button">Click me</button>
<p id="output"></p>
```

```
let button =
  document.getElementById("button");
let output =
  document.getElementById("output");
let clicks = 0;

function count() {
  clicks = clicks + 1;
  output.textContent = clicks + " clicks";
}

button.addEventListener("click", count);
```

Pass `count` **without** brackets. With `count()` the function runs once straight away and never on a click.

EVENT	HAPPENS WHEN …
`"click"`	something is clicked
`"input"`	someone types in a field
`"change"`	a selection has changed
`"submit"`	a form is submitted
`"keydown"`	a key is pressed
`"mouseover"`	the mouse moves over it

FOR READING: IN THE HTML

```
<button onclick="count()">Click</button>
```

Also works, but mixes HTML and JavaScript. We keep them apart: `addEventListener`.

Live calculation — input

```
let field =
  document.getElementById("points");

function show() {
  let points = Number(field.value);
  output.textContent =
    "Grade: " + calculateGrade(points);
}

field.addEventListener("input", show);
```

Recalculated on every key press, no button needed.

Handling a form — grade calculator

```
<form id="calculator">
  <label for="points">Points</label>
  <input type="number" id="points"
         min="0" max="15" required>
  <button type="submit">Calculate</button>
</form>
<p id="result"></p>
```

```
let calculator =
  document.getElementById("calculator");
function evaluate(event) {
  event.preventDefault();
  let field =
    document.getElementById("points");
  let points = Number(field.value);
  let grade = calculateGrade(points);
  document.getElementById("result")
    .textContent = "Grade: " + grade;
}

calculator.addEventListener("submit",
                            evaluate);
```

`preventDefault()` stops the browser from sending the form and reloading the page, so the field and the result stay visible.

Timing

```
// once after 2000 ms = 2 seconds
setTimeout(showHint, 2000);
// every second, until stopped
let clock = setInterval(tick, 1000);
clearInterval(clock);
```

Here too: function name **without** brackets.

Typical mistakes

- `… is not defined`: typo in the name or wrong case.
- `"5" + 3` gives `"53"`: forgot `Number()`.
- `=` instead of `===` in the `if`: assigns instead of comparing.
- Script in the head without `defer`: elements are `null`.
- Nothing happens: check the console (F12), print values with `console.log`.
- Breakpoint: **Sources** tab → click the line number. The program stops there.

Reference: developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model