

JavaScript Essentials — Cheat Sheet

FREE CHEAT SHEET

One page from the complete **JavaScript Essentials Companion Guide**. [Get the full guide →](#)

Core Syntax

TASK	SYNTAX
Variable	<code>const x = 5</code> / <code>let x = 5</code>
Template literal	<code>`\${a}`</code> and <code>\${b}`</code>
Conditional expr.	<code>x ? "yes" : "no"</code>
Loop values / keys	<code>for (x of arr)</code> / <code>for (k in obj)</code>
Arrow function	<code>(x) => x * 2</code>
Class	<code>class Name { constructor() {} }</code>
Try/catch	<code>try {} catch (e) {}</code>
Async function	<code>async function f() {}</code>
Await	<code>const x = await f()</code>

Array Methods

METHOD	PURPOSE
<code>.map(fn)</code>	Transform each item
<code>.filter(fn)</code>	Keep matching items
<code>.reduce(fn, init)</code>	Combine into one value
<code>.forEach(fn)</code>	Run code per item
<code>.find(fn)</code>	First match
<code>.includes(x)</code>	Membership check

Async and Promises

TASK	SYNTAX
Handle a promise	<code>p.then(fn).catch(fn)</code>
Run in parallel	<code>Promise.all([p1, p2])</code>
Wrap in try/catch	<code>try { await p } catch (e) {}</code>

String Methods

METHOD	PURPOSE
<code>.toUpperCase()</code> / <code>.toLowerCase()</code>	Change case
<code>.trim()</code>	Remove whitespace
<code>.split(sep)</code>	Split into array
<code>.replace(old, new)</code>	Replace first match
<code>.includes(x)</code>	Substring check

Common Errors

ERROR	MEANING
<code>TypeError</code>	Wrong type / property on <code>undefined</code>
<code>ReferenceError</code>	Variable not declared
<code>SyntaxError</code>	Invalid JS or JSON
<code>RangeError</code>	Number outside allowed range

npm Commands

TASK	COMMAND
Init project	<code>npm init -y</code>
Install a package	<code>npm install name</code>
Install dev dependency	<code>npm install --save-dev name</code>
Run a script	<code>npm run script-name</code>

Golden Rules

RULE

Use `===`, not `==`, essentially always.

RULE

`.map()` / `.filter()` return new arrays — they never change the original.

RULE

Arrow functions keep `this` from their surrounding scope. Plain functions don't.

RULE

Every `async` function returns a promise — `await` it, don't use it directly.

RULE

Trace `Cannot read properties of undefined` back to where the value became `undefined`.