

JavaScript

FULL CHEAT SHEET GUIDE



Tejas Talole

JavaScript Full Cheatsheet (2025 Edition)



1. Basics

1.1 Variables

```
// var - function-scoped, can be redeclared
var name = "Tejas";

// let - block-scoped, can be updated but not redeclared in the same scope
let age = 25;

// const - block-scoped, cannot be updated or redeclared
const country = "India";
```

-  `let` and `const` are preferred in modern JS.
-  `const` for values that don't change.

1.2 Data Types

Primitive: string, number, boolean, null, undefined, symbol, bigint

Non-primitive: object, array, function

```
let str = "Hello";           // String
let num = 42;                // Number
let bigIntNum = 9007199254740991n; // BigInt
let isTrue = true;          // Boolean
let nothing = null;         // Null
let notDefined;             // Undefined
let sym = Symbol("id");     // Symbol
```

1.3 Type Checking

```
typeof "Hello" // "string"
typeof 42       // "number"
typeof null    // "object" (historical quirk)
Array.isArray([]) // true
```

2. Operators

2.1 Arithmetic

```
+, -, *, /, %, ** // Addition, subtraction, multiplication, division, modulus,
exponent
```

2.2 Assignment

```
let x = 10;
x += 5; // 15
x -= 5; // 10
x *= 2; // 20
```

2.3 Comparison

```
5 == "5" // true (loose equality)
5 === "5" // false (strict equality)
5 != "5" // false
5 !== "5" // true
```

2.4 Logical

```
&& // AND
|| // OR
! // NOT
```

3. Control Flow

3.1 If / Else

```
if (age >= 18) {
  console.log("Adult");
} else {
  console.log("Minor");
}
```

3.2 Ternary

```
let msg = (age >= 18) ? "Adult" : "Minor";
```

3.3 Switch

```
switch(day) {  
  case "Monday":  
    console.log("Start of week");  
    break;  
  case "Friday":  
    console.log("Weekend soon!");  
    break;  
  default:  
    console.log("Another day");  
}
```

4. Loops

```
// For  
for (let i = 0; i < 5; i++) {  
  console.log(i);  
}  
  
// While  
let i = 0;  
while (i < 5) {  
  console.log(i);  
  i++;  
}  
  
// Do...while  
let j = 0;  
do {  
  console.log(j);  
  j++;  
} while (j < 5);  
  
// For...of (arrays, strings)  
for (let val of [10, 20, 30]) {  
  console.log(val);  
}  
  
// For...in (object keys)  
for (let key in {a:1, b:2}) {  
  console.log(key);  
}
```

5. Functions

5.1 Declaration

```
function greet(name) {  
  return `Hello ${name}`;  
}
```

```
}
```

5.2 Expression

```
const greet = function(name) {  
  return `Hello ${name}`;  
}
```

5.3 Arrow Functions

```
const greet = (name) => `Hello ${name}`;
```

5.4 Default Parameters

```
function multiply(a, b = 2) {  
  return a * b;  
}
```

5.5 Rest Parameters

```
function sum(...nums) {  
  return nums.reduce((a,b) => a+b, 0);  
}
```

6. Objects

```
const person = {  
  name: "Tejas",  
  age: 25,  
  greet() {  
    console.log(`Hi, I'm ${this.name}`);  
  }  
};  
person.greet();
```

- ◆ this refers to the object in regular methods, but behaves differently in arrow functions.

7. Arrays

```
let arr = [1, 2, 3];
arr.push(4);      // Add at end
arr.pop();        // Remove last
arr.unshift(0);   // Add at start
arr.shift();      // Remove first
arr.includes(2);  // true
arr.indexOf(2);   // 1
```

Common Array Methods*

```
[1, 2, 3].map(x => x * 2);           // [2, 4, 6]
[1, 2, 3].filter(x => x > 1);        // [2, 3]
[1, 2, 3].reduce((a,b) => a+b, 0);  // 6
[1, 2, 3].forEach(x => console.log(x));
```

ES6+ Features

- **Destructuring**

```
let [a, b] = [1, 2];
let {name, age} = {name: "Tejas", age: 25};
```

- **Spread Operator**

```
let nums = [1, 2, 3];
let copy = [...nums];
let merged = [...nums, 4, 5];
```

9. DOM Manipulation

```
document.getElementById("id");
document.querySelector(".class");
document.querySelectorAll("p");

let el = document.querySelector("#title");
el.textContent = "New Title";
el.style.color = "red";
```

10. Events

```
document.querySelector("button")
  .addEventListener("click", () => {
    alert("Button clicked!");
  });
```

11. JSON

```
let obj = {name: "Tejas"};
let jsonStr = JSON.stringify(obj); // object → JSON string
let parsed = JSON.parse(jsonStr);  // JSON string → object
```

12. Promises & Async

```
// Promise
let promise = new Promise((resolve, reject) => {
  setTimeout(() => resolve("Done!"), 1000);
});
promise.then(result => console.log(result));

// Async/Await
async function fetchData() {
  let data = await fetch("https://api.example.com");
  let json = await data.json();
  console.log(json);
}
```

13. Modules

```
// export.js
export const name = "Tejas";
export function greet() { console.log("Hi"); }

// import.js
import {name, greet} from './export.js';
```

14. Error Handling

```
try {
  throw new Error("Something went wrong");
} catch (err) {
  console.log(err.message);
} finally {
```

```
    console.log("Always runs");  
}
```

15. OOP in JS

```
class Person {  
  constructor(name) {  
    this.name = name;  
  }  
  greet() {  
    console.log(`Hi, I'm ${this.name}`);  
  }  
}  
  
let p = new Person("Tejas");  
p.greet();
```

16. Set & Map

```
let set = new Set([1, 2, 3, 3]); // Unique values  
set.add(4);  
set.has(2);  
  
let map = new Map();  
map.set("name", "Tejas");  
map.get("name");
```

17. Useful Built-in Methods

```
Number.parseInt("42");    // 42  
Number.parseFloat("42.5");// 42.5  
String.fromCharCode(65);  // "A"  
"Hello".toUpperCase();   // "HELLO"  
Math.max(1, 5, 3);        // 5  
Math.random();            // 0 - 1
```

18. Shortcuts

```
let val1 = obj && obj.key;    // Safe access  
let val2 = obj?.key;         // Optional chaining  
let val3 = obj?.key ?? "Default"; // Nullish coalescing
```