JavaScript Master Reference

## 🧠 BASICS

* `let`, `const`, `var` — Declare variables. Prefer `const` for constants, `let` for block-scoped values.
* `typeof value` — Returns the data type as a string (e.g., "number").
* `===` vs `==` — `===` checks value and type (strict equality), `==` allows type coercion.
* `null` — An intentional absence of value.
* `undefined` — A variable declared but not assigned a value.
* `NaN` — "Not a Number", result of invalid numeric operations.

## 🔁 CONTROL FLOW

* `if`, `else if`, `else` — Conditional branching.
* `switch` — Multi-branch conditional matching a value.
* `for (init; cond; inc)` — Standard loop.
* `for...of` — Loop through iterable values (arrays, strings).
* `for...in` — Loop through keys of an object.
* `while`, `do...while` — Loops based on a condition.
* `break`, `continue` — Exit loop early or skip to next iteration.
* `try { } catch (e) { } finally { }` — Error handling.

## 📦 OBJECTS & ARRAYS

* `{ key: value }` — Object literal.
* `obj.key`, `obj["key"]` — Access properties.
* `Object.keys(obj)` — Array of keys.
* `Object.values(obj)` — Array of values.
* `Object.entries(obj)` — Array of `[key, value]` pairs.
* `Array.isArray(val)` — Checks if value is an array.
* `[1, 2, 3]` — Array literal.
* `arr.length` — Number of items.

## 🧱 ARRAY METHODS

* `arr.push(val)` — Add to end.
* `arr.pop()` — Remove from end.
* `arr.shift()` — Remove from start.
* `arr.unshift(val)` — Add to start.
* `arr.concat(arr2)` — Join arrays.
* `arr.slice(start, end)` — Copy part of array.
* `arr.splice(index, count, ...newItems)` — Remove/replace items.
* `arr.map(fn)` — Transform items.
* `arr.filter(fn)` — Keep items matching condition.
* `arr.reduce(fn, init)` — Combine items into one value.
* `arr.find(fn)` — First matching item.
* `arr.findIndex(fn)` — Index of first match.
* `arr.includes(val)` — Check for presence.
* `arr.sort([compareFn])` — Sort items.
* `arr.reverse()` — Reverse in-place.
* `arr.flat(depth)` — Flatten nested arrays.
* `arr.every(fn)` — All items match?
* `arr.some(fn)` — Any items match?

## 🔡 STRING METHODS

* `str.length` — Length of string.
* `str.charAt(i)` — Character at position.
* `str.includes(substr)` — Contains substring?
* `str.indexOf(substr)` — First index of substring.
* `str.lastIndexOf(substr)` — Last index of substring.
* `str.slice(start, end)` — Substring.
* `str.split(delimiter)` — Split into array.
* `str.replace(find, replace)` — Replace content.
* `str.trim()` — Remove leading/trailing spaces.
* `str.toUpperCase()` — Capitalize.
* `str.toLowerCase()` — Lowercase.
* `str.startsWith(str)` — Begins with?
* `str.endsWith(str)` — Ends with?

## 🔢 MATH & NUMBERS

* `Math.round(num)` — Round to nearest.
* `Math.floor(num)` — Round down.
* `Math.ceil(num)` — Round up.
* `Math.max(...nums)` — Highest value.
* `Math.min(...nums)` — Lowest value.
* `Math.random()` — Random [0, 1).
* `Math.abs(val)` — Absolute value.
* `Math.pow(base, exp)` — Exponentiation.
* `Math.sqrt(num)` — Square root.

## 📆 DATE METHODS

* `new Date()` — Current date/time.
* `Date.now()` — Timestamp in ms.
* `date.getFullYear()` — Year.
* `date.getMonth()` — Month (0-indexed).
* `date.getDate()` — Day of month.
* `date.getDay()` — Day of week (0–6).
* `date.getHours()` — Hour.
* `date.toISOString()` — ISO date string.
* `date.toLocaleDateString()` — Locale date.

## 🔧 UTILITY FUNCTIONS

* `debounce(fn, ms)` — Wait X ms after last call.
* `throttle(fn, ms)` — Limit calls to every X ms.
* `range(start, end)` — Create number array.
* `chunk(arr, size)` — Break array into chunks.
* `flattenDeep(arr)` — Fully flatten nested array.
* `groupBy(arr, fn)` — Group items by key.
* `unique(arr)` — Remove duplicates.
* `splitIntoRows(arr, rowLen)` — Break into 2D rows.

## 🧠 FUNCTION BASICS

* `function name(args) {}` — Named function.
* `const fn = (args) => {}` — Arrow function.
* `return value;` — Return from function.
* Functions are first-class: can be passed, returned, stored.