JavaScript String Methods Cheat Sheet

Core methods, == vs === behavior, and five string-handling defect patterns for QA, BA, and IT professionals.

1. Core String Methods

Method	Does	Watch For
`.trim()`	Removes leading/trailing whitespace	Result must be reassigned — strings are immutable
`.includes(text)`	Boolean substring check	Case-sensitive by default
`.split(delim)`	Breaks a string into an array	Breaks on inconsistent delimiters in source data
`.slice(start,end)`	Extracts a substring by index	Off-by-one errors common when masking/truncating IDs
`.replace(old,new)`	Replaces matches	No /g flag = only the first match is replaced
`.toUpperCase()/.toLowerCase()`	Changes case	Missing call is a common cause of case-sensitivity defects
`.length`	Character count (property)	Counts UTF-16 code units — breaks on emoji/some scripts

2. == vs === Quick Reference

Comparison	Result	Why
`"5" == 5`	true	Loose equality coerces the string to a number
`"5" === 5`	false	Strict equality — different types, no coercion
`"" == false`	true	Empty string coerces to falsy
`"5" + 3`	"53"	+ triggers concatenation the moment either side is a string

3. Five Defect Patterns to Recognize on Sight

Discarded method return values
userInput.trim() alone does nothing — the result must be reassigned. No error is thrown, so this survives code review easily.

Case-sensitive comparisons without normalization
"Reyes" === "reyes" is false. Any externally sourced text needs an explicit case-sensitivity decision.

Numeric-looking strings cast to Number
Account numbers, ZIP codes, and IDs with leading zeros lose data when converted to a JavaScript Number: "00458213" becomes 458213.

Missing global flag on regex replace
str.replace(/,/, "") removes only the first match. Partially-cleaned data often traces back to a missing /g flag.

Silent type coercion in math operations
"5" + 3 produces "53", not 8. Forms pulling raw input values without explicit conversion are a common source of broken totals.

Edge case: String primitives and String objects (new String()) look identical but are never === equal to each other when wrapped in an object, since object comparison checks reference, not value.

Rule of thumb: if a value will never be added, subtracted, or averaged — account numbers, ZIP codes, medical record numbers — keep it a string through the entire pipeline.