JavaScript Values & Type Coercion Cheat Sheet

The falsy-value list, == vs === behavior, and the value-related defect patterns QA and BA teams run into most.

---

1. The Complete Falsy Value List

```
false   0   -0   ""   null   undefined   NaN
```

That's the entire list — six distinct values plus -0. Everything else, including `[]` and `{}`, is truthy.

2. The Seven Primitive Types + Object

Type	Example	Where You'll See It
String	"claim-4521"	IDs, names, free-text fields
Number	1042.50	Amounts, counts, timestamps
Boolean	true / false	Eligibility flags, active status
Undefined	undefined	Never assigned; missing property
Null	null	Deliberately set to nothing
Symbol	Symbol("id")	Rare — unique object keys
BigInt	9007199254740993n	Large IDs beyond safe Number range
Object	{ id: 1 }	Compared by reference, not content

3. == vs === — Same Comparison, Different Answer

Comparison	== (loose)	=== (strict)
"100" == 100	true — coerced	false
0 == false	true	false
null == undefined	true	false
"" == 0	true	false
NaN == NaN	false (always)	false (always)

Rule of thumb: if you see == in production conditional logic or a test assertion, treat it as a flag worth asking about — not a style choice.

4. Five Value-Related Defect Patterns

Falsy 0 treated as missing
if (fieldValue) treats a legitimate 0 the same as empty. Common in copay amounts, counts, lab results.

Object comparison by reference
Two objects with identical content are never === equal. Use deep-equality (e.g. Jest's toEqual) instead.

Empty-array truthy bug
[] is truthy, so if (resultsArray) passes even with zero results. Check .length > 0 instead.

NaN never equals itself
NaN === NaN is false. Use Number.isNaN(value), never a direct equality check.

String + number concatenation
"5" + 3 produces "53", not 8, whenever either operand is a string in a + expression.