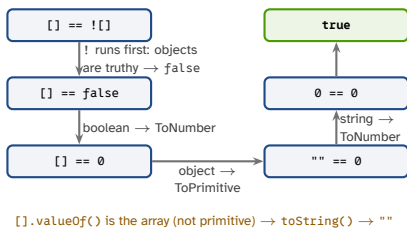
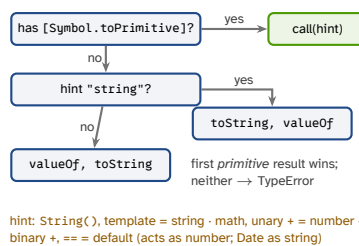


In one line: Seven primitives (immutable, by value) + **object** (by reference). Coercion runs a few spec algorithms — **ToPrimitive**, **ToNumber**, **IsLooselyEqual** — so every “WAT” is traceable. ESM imports are **live bindings**; require returns an object.

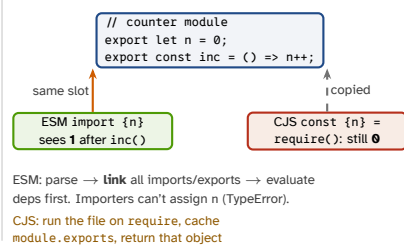
1 [] == ![] step by step



2 ToPrimitive(obj, hint)



3 live binding vs copy



Types and equality

- Primitives: string number bigint boolean undefined symbol null; everything else is an object (arrays, functions, dates). "a".length works by temporary boxing.
- typeof: null -> "object" (a 1995 tag bug, kept forever) · functions -> "function" · arrays -> "object" (Array.isArray) · undeclared -> "undefined" (a TDZ name throws).
- ===: no coercion. == (IsLooselyEqual): same type -> ===; null == undefined and nothing else; number vs string -> ToNumber; boolean -> ToNumber first; object vs primitive -> ToPrimitive; then repeat.
- NaN !== NaN, +0 === -0. Object.is fixes both; **SameValueZero** (includes, Map/Set keys): NaN equals NaN, ±0 equal. indexOf(NaN) is -1, includes(NaN) true.
- Falsy**: false 0 -0 0n "" null undefined NaN — nothing else. "0", "false", [], {} are truthy.

The puzzles — and why

expression	result	why
[] + {}	"[object Object]"	both -> strings("", "[object Object]"), concat
{ } + []	0 (statement)	leading { } is a block; +[] -> 0
[] + []	""	"" + ""
[1, 2] + [3]	"1,23"	arrays join with commas
"5" - 2 or "5" + 2	3 or "52"	- is numeric only; + prefers strings
+" " · + "12 " · + "12px"	0 · 12 · NaN	ToNumber trims; whole string must parse
null + 1 · undefined + 1	1 · NaN	null -> 0, undefined -> NaN
null == 0 · null >= 0	false · true	== special-cases null; >= does ToNumber
[10, 1, 2].sort()	[1, 10, 2]	default sort compares strings
"b"+"a"+"a"+"a"	"baNaNa"	+"a" is NaN
0.1 + 0.2 === 0.3	false	0.30000000000000004

Numbers

- One number: IEEE-754 double. Integers exact up to $\text{Number.MAX_SAFE_INTEGER} = 2^{53} - 1$; $2^{53} + 1 === 2^{53}$. Compare floats with a tolerance ($\text{Number.EPSILON} = 2^{-52}$ is relative).
- BigInt (10n, ES2020): arbitrary precision; mixing with number in arithmetic throws TypeError; JSON.stringify throws. IDs from a 64-bit backend -> send as strings.
- Bitwise ops go through Int32: $2^{31} \mid 0$ is negative.

??, ||, ?.

a || d takes d for any falsy a (so 0, "" are lost); a ?? d only for null/undefined (ES2020). Mixing ?? with ||/&& needs parens (SyntaxError). a?.b.c short-circuits the whole chain to undefined; f?().o?.[k]; not assignable. ??=, ||=, &&= (ES2021).

Value vs reference, copying

Primitives copy the value; objects copy the *reference* (“call by sharing”: mutating a param is visible to the caller, reassigning it is not). **Swift**: no structs here — every object behaves like a class, so value semantics are by convention (spread, freeze, immutable updates). { ... o }, [... a], Object.assign are **shallow**. structuredClone is deep: cycles, Map/Set, Date, typed arrays; throws on functions; class instances come back as plain objects. JSON round-trip drops undefined/functions, turns Dates into strings, throws on cycles and BigInt.

Modules — ESM vs CommonJS

	ESM import/export	CJS require/module.exports
when	static, top level, linked before run; hoisted	a runtime call; anywhere, conditional
bindings	live , read-only	a plain object; destructuring copies
loading	async-capable; top-level await	synchronous
this, mode	undefined, always strict	module.exports, sloppy
tree-shaking	yes (static shape)	hard

- Circular**: CJS gets the *partially filled* exports (missing keys undefined); ESM: a let/const/class read before its module ran throws ReferenceError (TDZ); function declarations work (hoisted).
- Interop**: import x from 'cjs' = module.exports; CJS named imports are guessed. require(esm) was ERR_REQUIRE_ESM -> import(); bundlers/TS fake ESM via __esModule — the .default mess.
- import() (both systems) -> promise of the namespace: lazy/split loading.

Interview traps

- ! count || 10 with count = 0 gives 10 — use ??.
- typeof null is "object"; x != null checks both nullish.
- Top-level await in a shared module blocks every importer's evaluation.

Remember

=== by default, ?? for defaults · ToPrimitive then ToNumber · doubles lie after 2^{53} · spread is shallow · ESM links, CJS runs.

Likely questions

- == vs === vs Object.is? — coerce · strict · strict + NaN, ±0.
- Why is [] == ![] true? — drawing 1.
- ESM vs CJS? — static + live + async-capable vs runtime + copied + sync.