How many of you've encountered == or === operator in JS.?

1) == ⟶ loose equality check where we don't consider types ✗

2) === ⟶ Strict equality check where we consider types also ✗

Not true

So acc. to the docs, for both == and === type checks are done.

x == y
x === y
} → Type(x) Type(y)

this is always checked.

Difference is both of them does something diff with this type info.

== → after type checks, if both args have same type performs ===

=== → after type check, if both args have diff type, it returns false.

The "==" operator

- abstract equality comparator
- coercive equality:

So for == operator, if the types are not equal they may coerce.

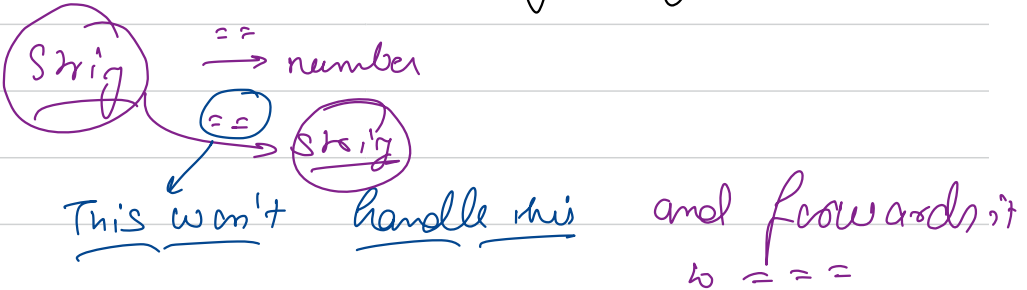
a == b → whether coercion is helpful or not.

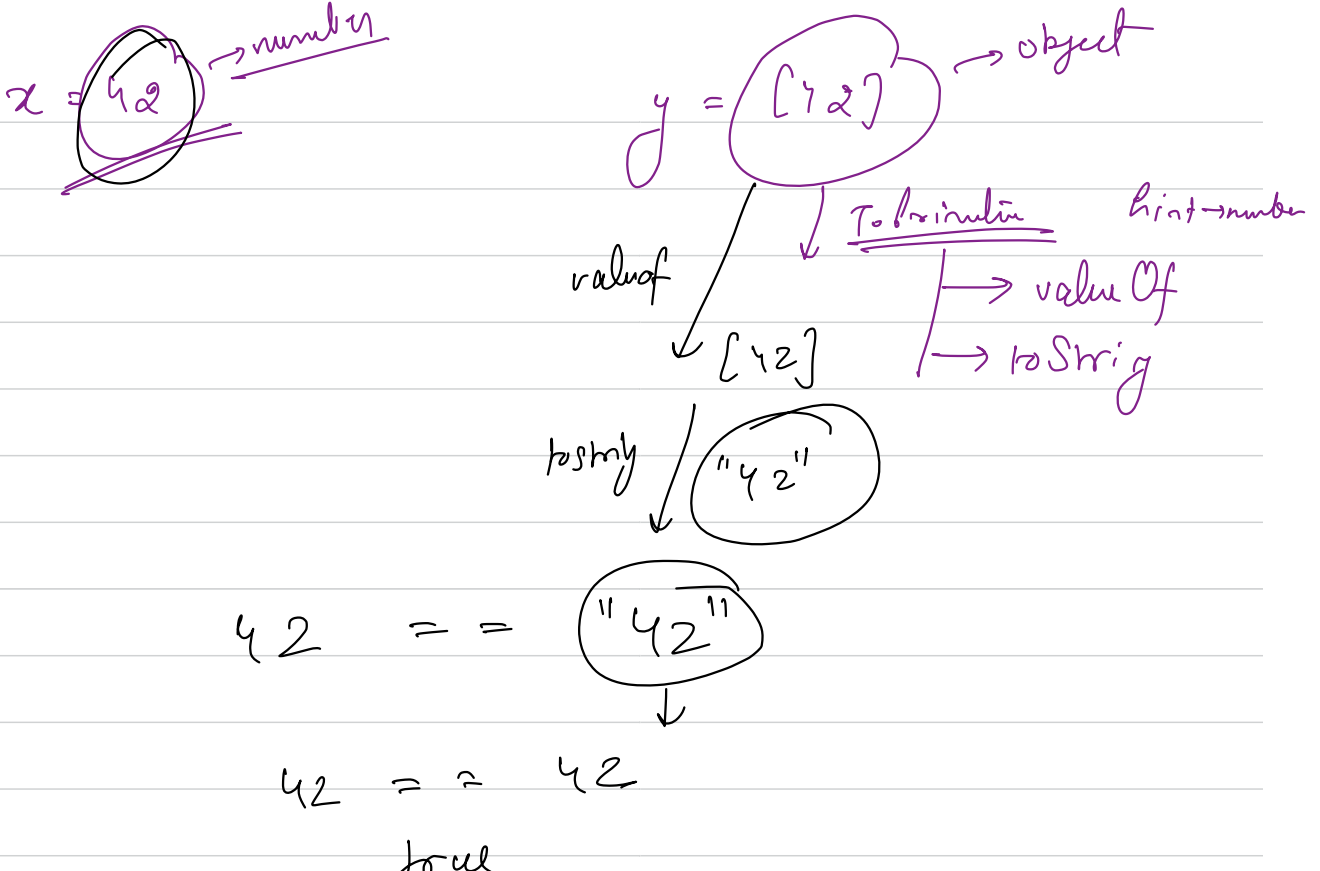
1) treats null and undefined as same.

2) == op, prefers numeric comparison wherever it can.

↳ When strings and booleans are compared with no., it tries to convert them to a no.

3) == only compares primitive, if any obj is given it will call ToPrimitive on it.




x = 42
number
y = [42]
object
ToPrimitive
hint → number
valueOf
→ valueOf
→ toString
[42]
toString
"42"
42 == "42"
42 == 42
true

Corner cases

true

a) `[] == ![]` → checks if array is equal to one array's negation with coercive comparison.

true

b) `[] != []` → not a real life example → whether the 2 arrays are not coercively equal.

[] == 

To Boolean

falsy

to cast

[] == !true

[] == false

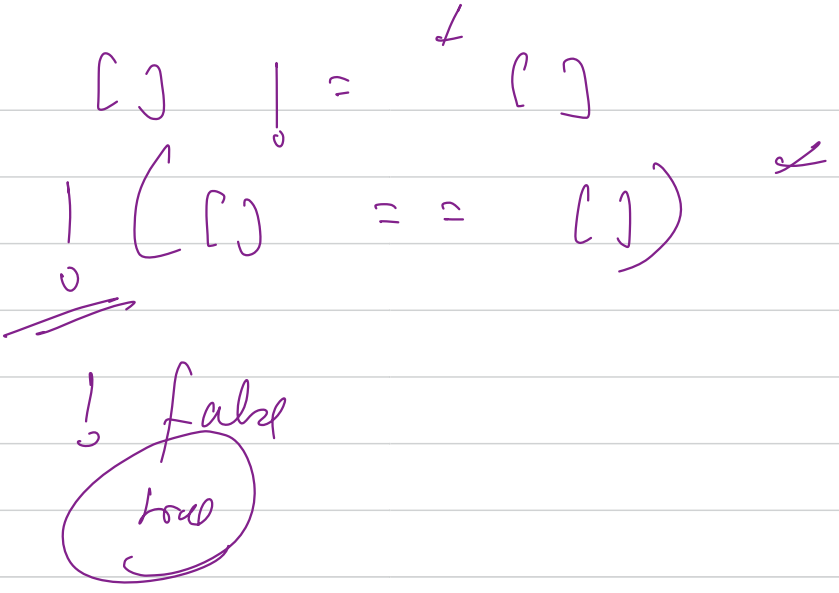
"" == false

0 == false

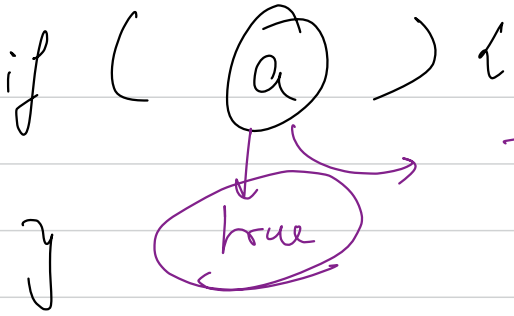
0 == 0

true

To Primitive


[] != []
!([] == [])
!
false
true

a = []



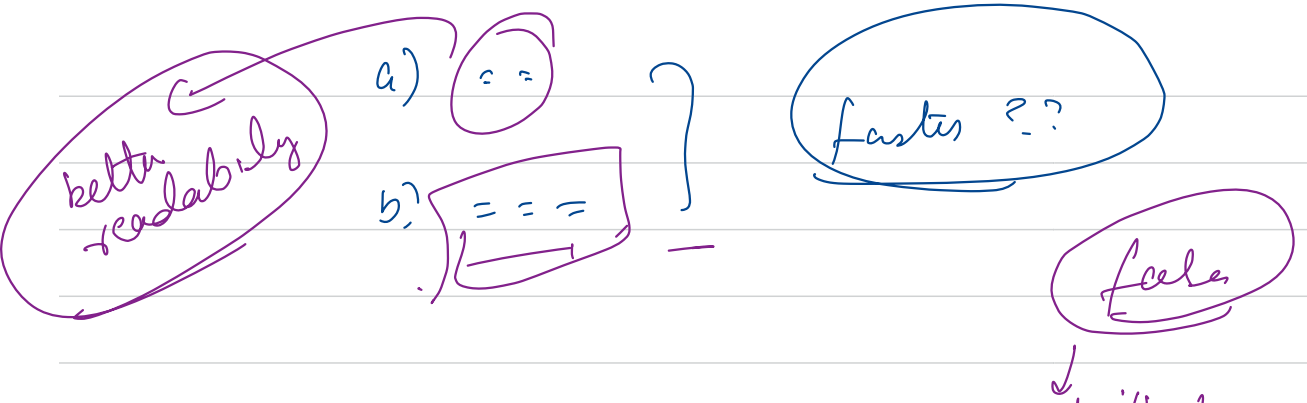
[ ]
↓
Truthy

if (a == true) →

a
↓
1
↓
0 == 1 → false

if (a == false) →

a
↓
1
↓
0 == 0 → true


a) ==
b) ===
better readability
faster ??
false
will do no execute
no entra algo