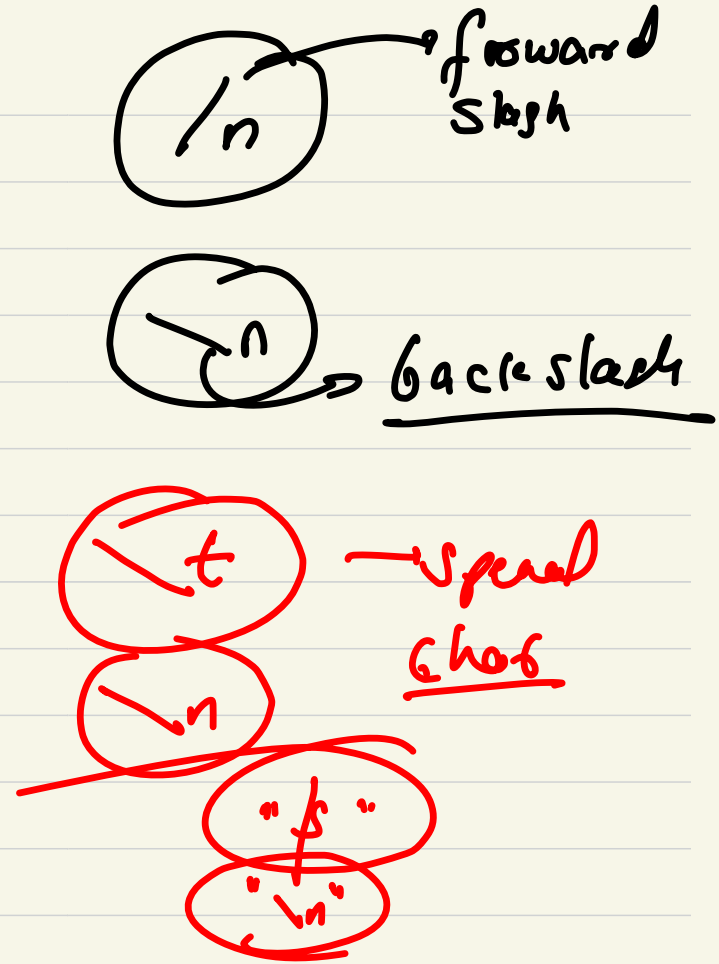
var x = 10;
var y = 2;

console.log(x + y)

x - y



Relational Operators

< , > , <= , >=

(10 < 12)

(13 > 5)

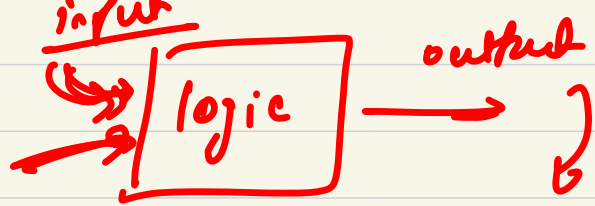
(5 < 2)

operand1 > < >= <= operand2

(10 < 12) → true

(5 < 2) → false

Logical Operators



true
false

Boolean

operand 1

operand 2

AND
OR
→ these need 2 operands

NOT → 1 operand

AN

AND GATE ⟶ &&

OR GATE ⟶ ||

NOT GATE ⟶ !

operand 1 → boolean

operand 2 → boolean

AND

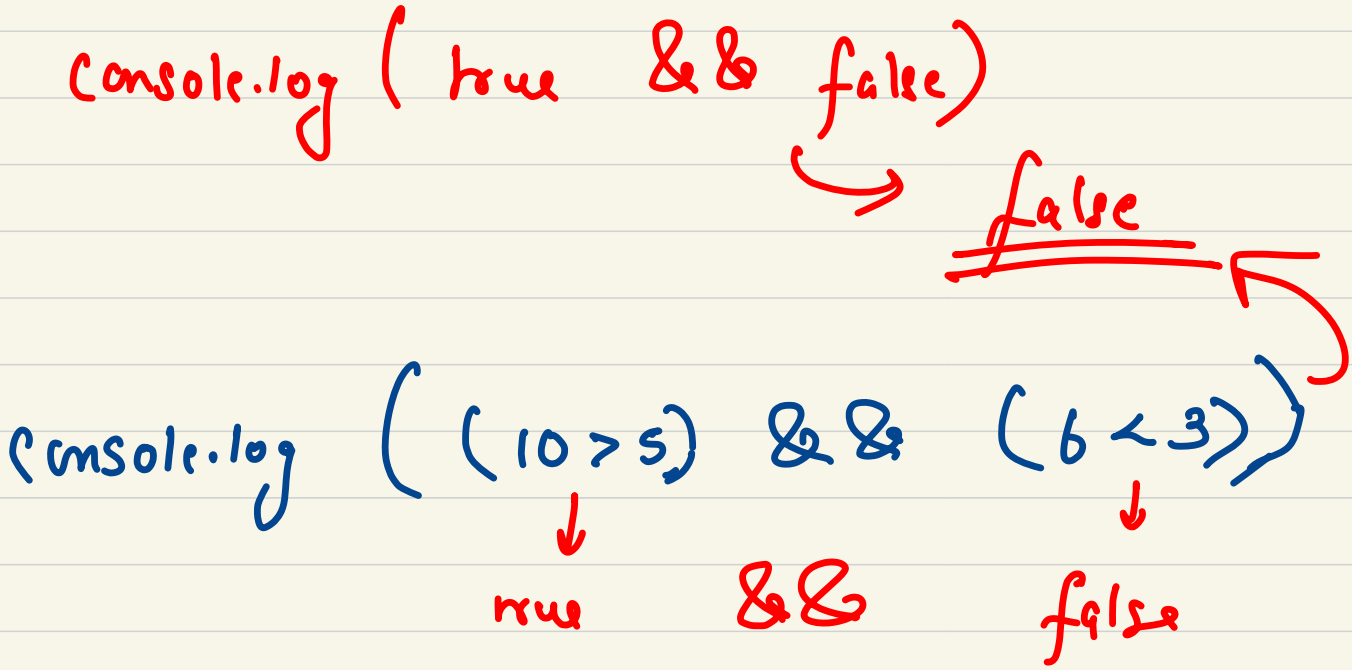
X	Y	X AND Y
false	false	false
true	false	false
false	true	false
true	true	true

OR

X	Y	X OR Y
false	false	false
false	true	true
true	false	true
true	true	true

NOT

X	output
true	false
false	true


console.log (true && false)
false
console.log ((10 > 5) && (6 < 3))
true && false

10 && 6

||

Q. what values are falsy in JS ??

null
undefined
""
+0, -0, NaN
false

→ empty string

} apart from these everything is truthy

→ Coercion (type Interconversion)

AND

X	Y	X AND Y
false	false	false
true	false	false
false	true	false
true	true	true

(0 && 6)

1

0

OR

X	Y	X OR Y
false	false	false
false	true	true
true	false	true
true	true	true

NOT

X	output
true	false
false	true

In a AND gate, if the first input is false, then, it doesn't evaluate the second input and immediately returns the first input.

as well as if first input is true, then the second input has to be evaluated & then second input is returned

In a OR gate, if the first input is true, then it doesn't evaluate the second input & immediately returns the first input.

whereas,

if the first input is false then it returns the second input.:

10 && ?

truthy

(10 > 6) && (6 < 7)

true

console.log(6 && 10)

10

Numbers →

directional

-0

NaN

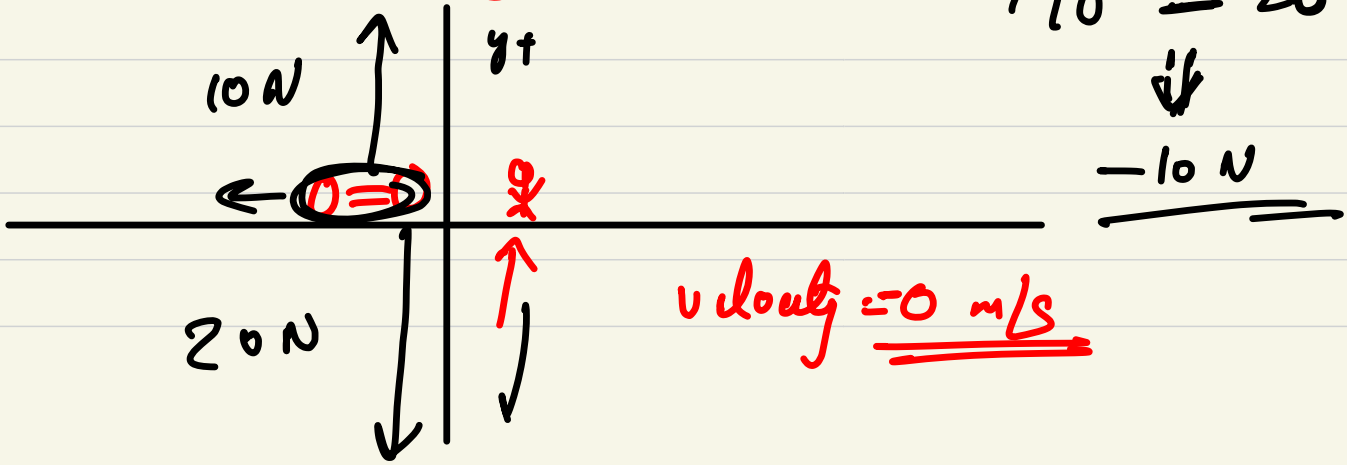
Infinity

-Infinity

} Numbers

magnitude

direction



↑ 10 — 20

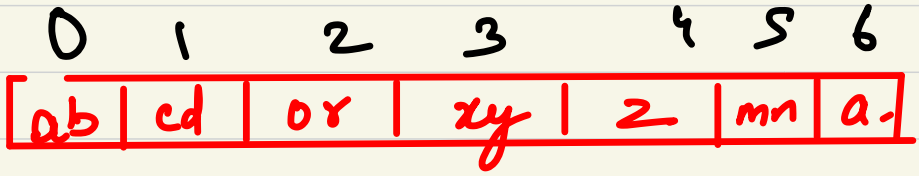
↓

-10 N

Net force → 5 Newton in -ve x axis

$-5\ N$

NaN → Not A Number



→ Return the Bucket Number in which df string is present.

if there is a situation where you're bound to return a number, but there is no valid possible No. to return,

then we do NaN

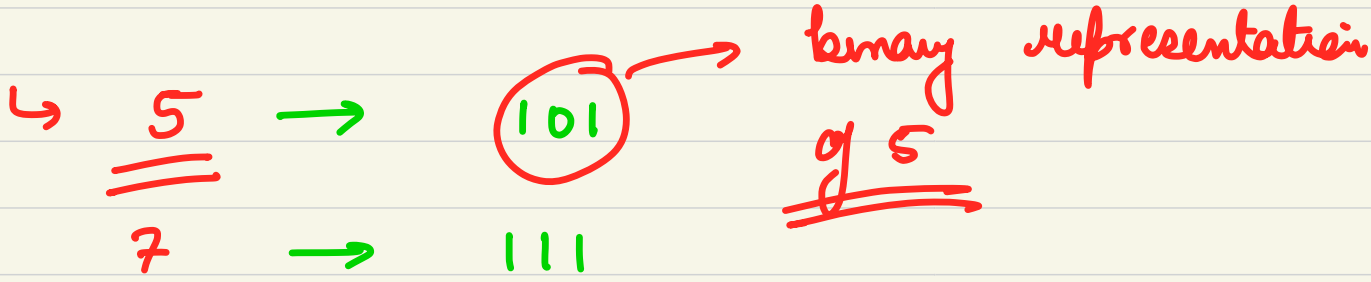
#! which is the only number in JS, which is not equal to itself 2.2

NaN

undefined / null

"Socket" / 2.

Bitwise Operators



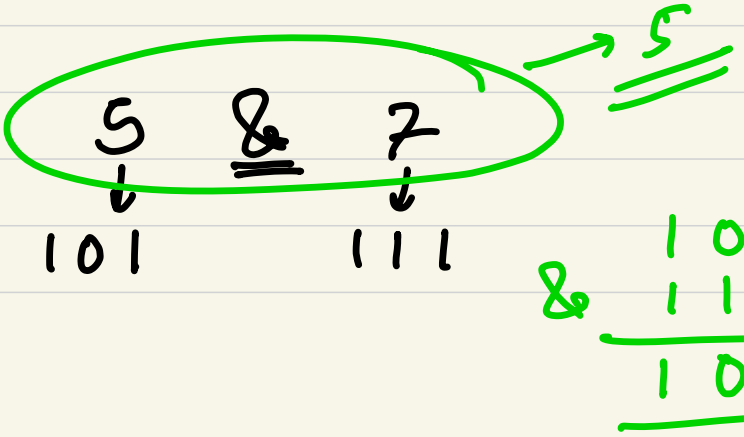
bitwise operators perform the corresponding operation bit by bit on the given operand.

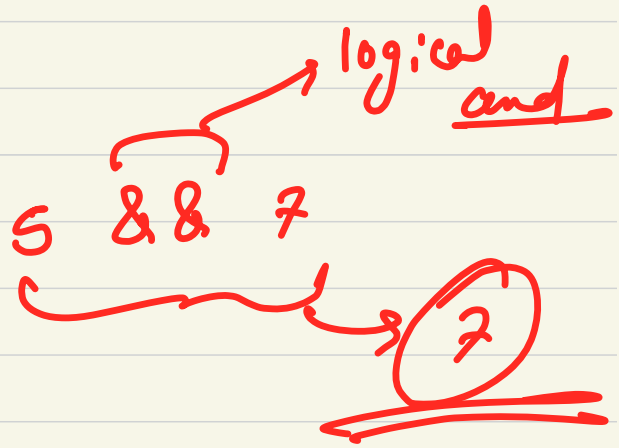
& → single ampersand → bitwise and

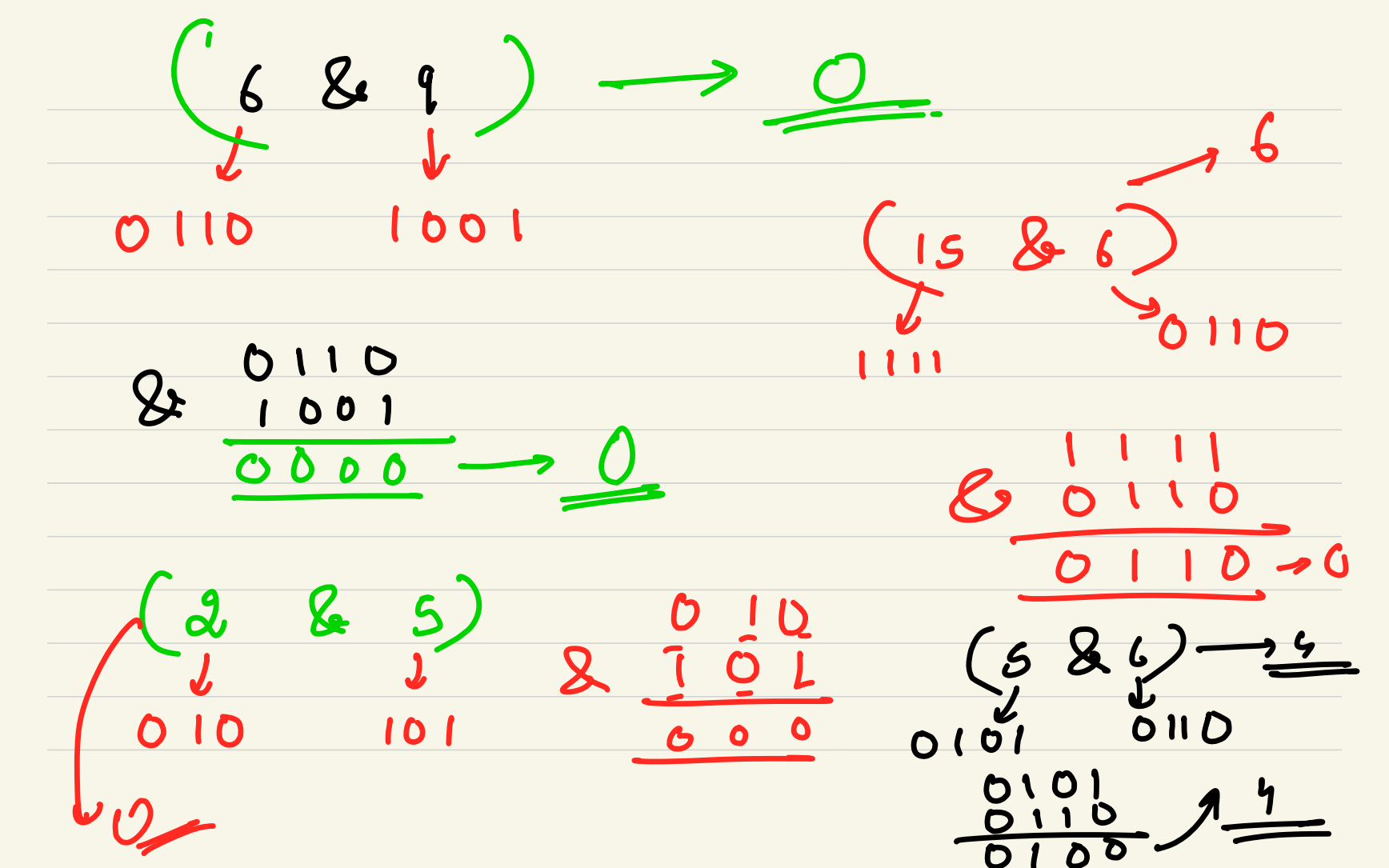
| → single pipe → bitwise or

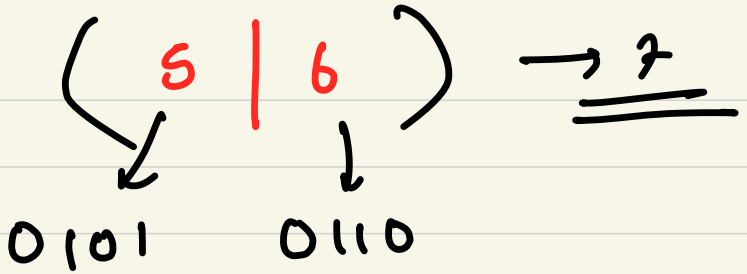
^ → bitwise xor

~ → bitwise not

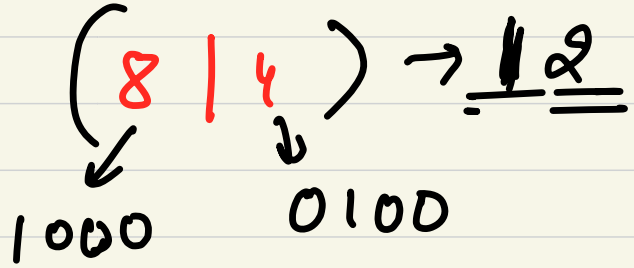





(6 & 9) → 0
0110
1001
(15 & 6) → 6
1111
0110
& 0110
1001
0000 → 0
& 1111
0110
0110 → 6
(2 & 5) → 0
010
101
& 010
101
000
(5 & 6) → 4
0101
0110
0101
0110
0100 → 4



→ 7

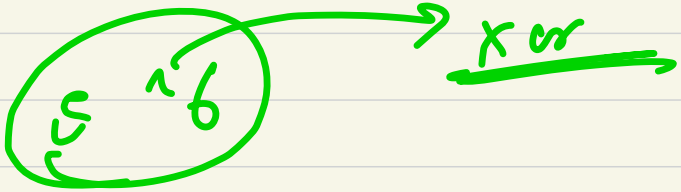


→ 12

```
  0101
| 0110
------
  0111
```

→ 7

```
  1000
  0100
------
  1100
```



XOR

A	B	A ^ B
0	0	0
0	1	1
1	0	1
1	1	0

(5 ^ 6) → 3

5 → 0101

6 → 0110

```
 0101
 0110
 ----
 0011  → 3
 ----
```

Equality Operators

== → abstract equality operator

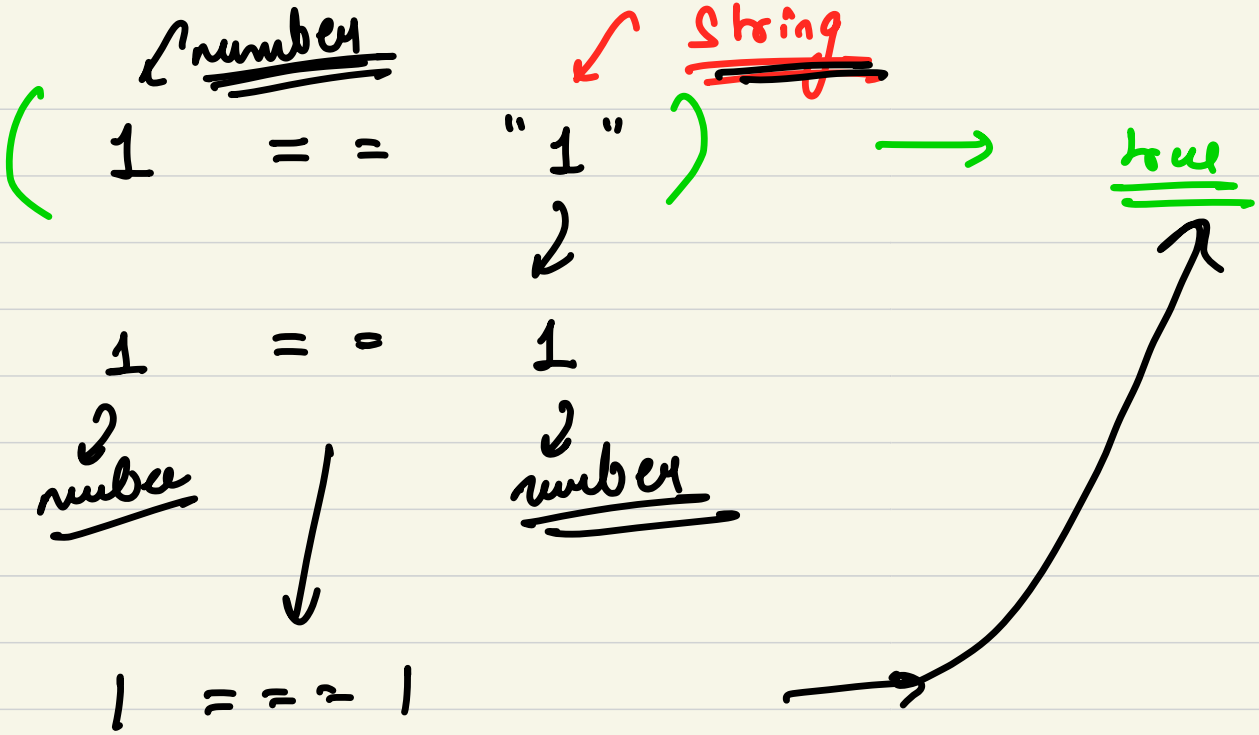
=== → strict equality operator

== → it checks the types of both operands,
- if type is same, then it calls ===
- if types are not same then type conversion occurs (coercion) & then comparison is done.

=== → it checks types of both the operands

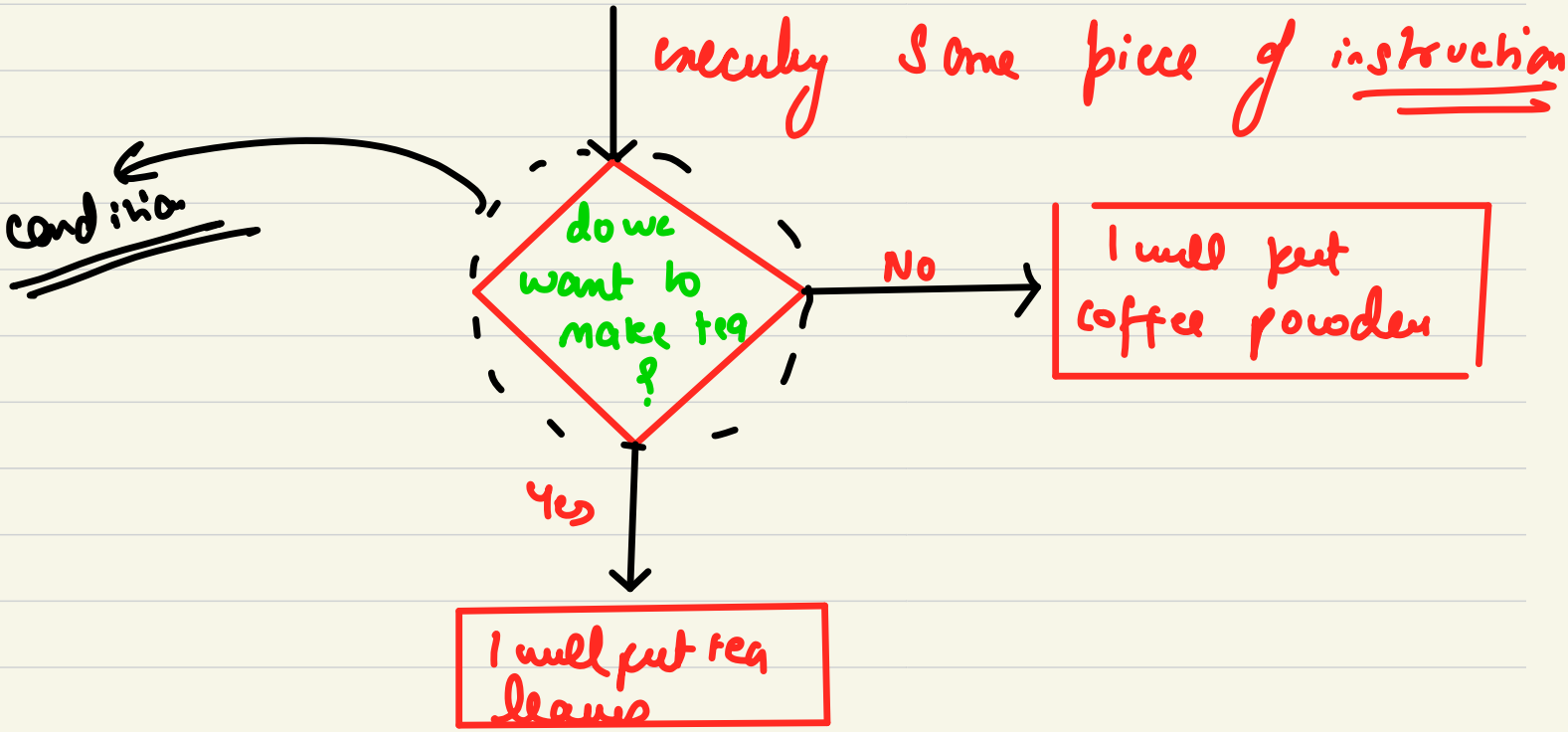
↳ if types are different it returns false

↳ if types are same then value comparison happens.


number
String
1 == "1"
true
1 == 1
number
number
1 === 1

Conditional Statements

we evaluate a condition so, using conditional statement, we can take decision and correspondingly change the actions we want to do.


executing some piece of instruction
condition
do we want to make tea ?
No
I will put coffee powder
Yes
I will put tea leaves

we have

↳ if and else statement

pair of curly braces create a block

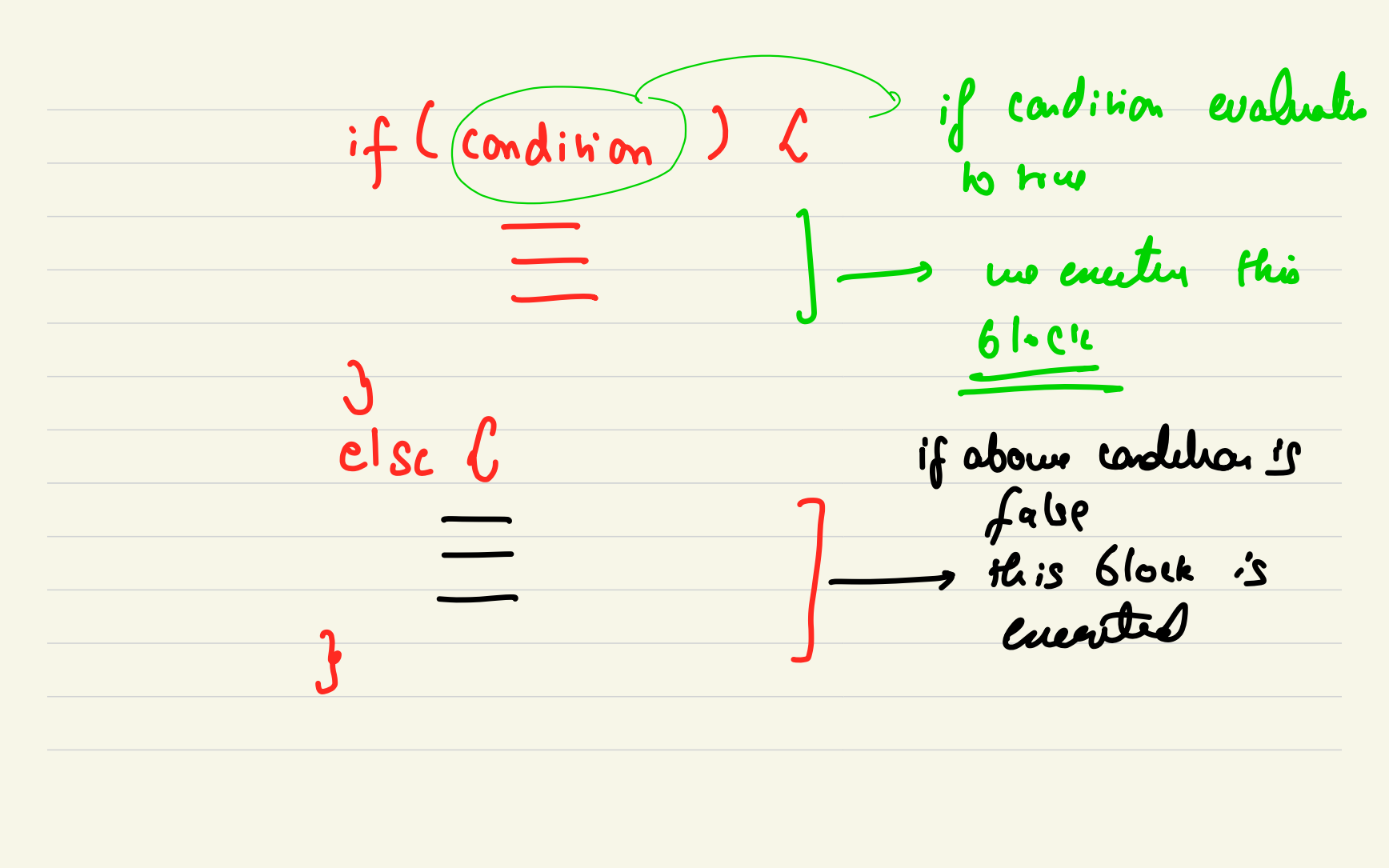
{ → block

}

//Syntax

```
if ( condition ) {
    // block
}
```

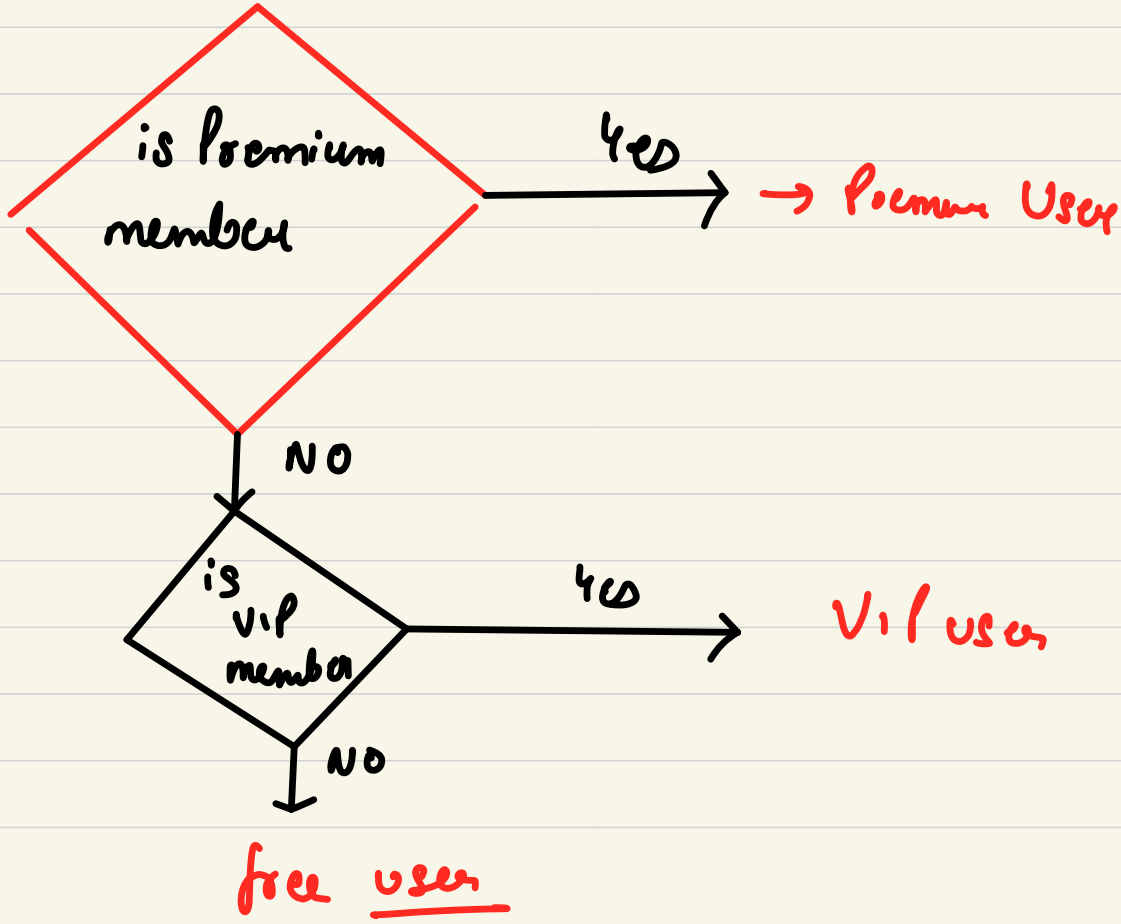
if this condition holds true then this region will be executed


if (condition) {
if condition evaluates to true
we execute this block
} else {
if above condition is false this block is executed
}

→ if the condition is true, only if block is executed and else block is completely avoided.

→ if condition is false, if block is completely avoided & only else block executes.

NOTE → if block can exist without else block but else block will not exist without if block.


is Premium member
YES
→ Premium User
NO
is VIP member
YES
VIP user
NO
free user

if (condition1) {

==

}

} → if condition1 is true only this part is execute

else if (condition2) {

} → if condition2 is true only this part is executed

} else {

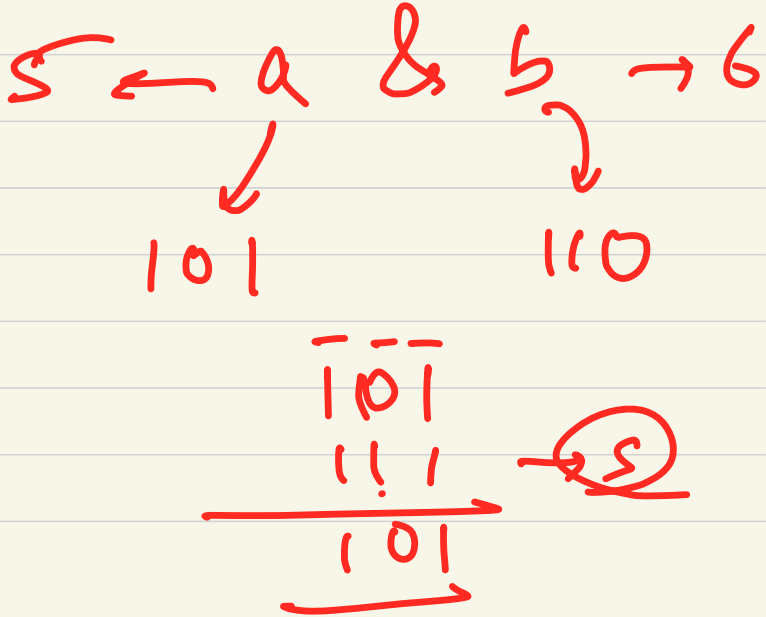
} → if everything above is false only this part is executed.

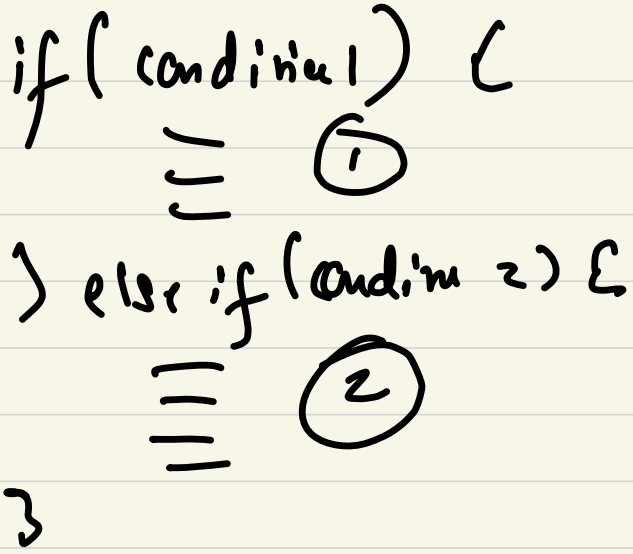
```
if (condition1) {
    ==
}
else if( condition 2) {
    ==
}
else if ( condition 3) {
    ==
}
.
.
.
else {
}
```

if multiple conditions are true, then the block when first true condition is written will be executed.

1. if can exist without else if & else
2. else cannot exist without if but can exist without else if
3. elseif cannot exist without if but can exist without else.

a && b




if (condition1) {
1
} else if (condition2) {
2
}

```
if(condition1){
    ≡ ①
}

if(condition2){
    ≡ ②
}
```

```
if ( isUserPrime && isUserDiscount ) {
        both content
} else if ( isUserPrime ) {
        only prime content
} else {
        buy something
}
```

Nested if else

```
if ( isUserPrime) {
    if (isUserDiscovery) {
        showboth
    } else {
        only prim
    }
} else {
    by something
}
```