MERN Day - 6

JavaScript

C → gcc

C++ → gdb

Java → JVM

python → Python3

C# → Microsoft.

JS → Node / ~~It~~ can also run without node env but not in local machine.

→ Node

① Printing

console.log(" ") → Print → ~~It~~ JS no. 1 a printing statement.

Terminal   string saying

C/C++ → Procedure/Structure & OOPS

2S

Java → Fully OOPS

4S

Python → Structure, object, Script, Web Prog/

10S

JS

↳ To integrate MERN / Javascript Stack

↳ Scripting.

→ normal → vanilla JS

↳ Programming ⟨ oops → $\boxed{\text{ES6}}$

delarations/datatype

long
integer
~~byte~~
~~float~~
double
short
→ number ✓

sting
char
→ string ✓

boolean → bbolean

→ Primitive datatypes

Secondary datatypes ⟷

Arrays
objects
sets

Arrays => arry = [10, 20, 30, 40, "abc", true]

console.log(array). // o/p : [10, 20, 30, 40, 'abc', true] objects.

1st way:

```
obj1 = {
    firstname: "St. Joseph",
    "lastname": "Institute of Technology",

}
clg(obj1) //o/p {
                    firstname: 'St. Joseph',
                    lastname: 'Institute of Techology',

                }
```

2nd way:

```
obj2 = {}
obj2["firstname"] = "hello"
obj2[        ] = "SJIT"
```

```
obj2["lastname"]=
console.log(obj2)   //o/p: {
                        firstname: 'Rello',
                        lastname: 'SJIT'
                    }
```

3rd way:

```
obj3= new Object()
obj3["firstname"]="Ri",
obj3["last name"]="SJIT"
obj3[
clg(obj3) // {
                firstname: 'Ri',
                lastname: 'SJIT'
          }
```

Operators

① Assignment operator (=)

② Unary operator (++, --)

tic operator (+, -, *, /, %)

② Arithmetic op

③ Logical operator (&&, ||, !)

④ Relational / Conditional operator (>, <, >=, <=, ==, !=, ===, !==)

⑤ Bitwise operator (&, |, ~, ^, <<, >>, >>>)

⑥ Combination of operators (+=, -=) Assignment+ Arithmetic, Assignment + Bitwise

⑦ dot operator (.)

```
    console.log(5&&10) //o/p: ~~0~~ 10 ✗
①  console.log(true &&10) //op: 10 ✓
②  console.log(false&&10) //op: ~~0~~ false ✗
③  console.log(false || 10) //op: 10 ✓
④  console.log(true || 15) //op: ~~25~~ true ✗
⑤  console.log(5 || 10) //op: ~~15~~ 5 ✗
⑥  console.log(false&&true) //o/p false ✓
⑦  console.log(true && false) //o/p. false ✓
⑧  console.log(false||true) //o/p: true ✓
⑨  console.log(true || false) //o/p: true ✓
⑩
```

check for the First condition if

2nd condition

And(&&) → Itwill che goingto checkthe
it is true it going to terminate its execution
else it will

OR(||) → It will check for the first condition
if it is true then it will not check
the 2nd condition but if the 1st condition
is false it will check for the 2nd
condition.

Relational operator

a = '5'

b = 5

clg(a==b) true

clg(a===b) false

'5' == 5 => true

type: string   type: number

== operator will check only its
content

'5' === 5 => false

type string === type number

Indirectly it
using logical
&& operator

ks its

=== operator will check its content as well as its type

Bitwise

Byte ⇒ 1 Byte = how many bit
8 bit

0000 0000

$2^4\ 2^3\ 2^2\ 2^1\ 2^0$

& 2 → 0000 0010
2 → 0000 0010

0000 0010 ⇒ 2

ε4

& 70 → 0100 0110
35 → 0010 0011

0000 0010 = 2

AND

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

64 + 63 = 127
127 − 32 = 95 >
95 − 16 = 79
79 − 8 = 71
~~71 − 4 = 67~~
~~71 − 2 = 69~~
71 − 1 = 70

Ans XOR : 0 1 1 0 0 1 0 1
4 + 1 = 101

An

64 + 32 + ...

-128

1 ⇒ 0000 0001

~1 ⇒ 1111 1110 ⇒ -2

-128 + 64 + 32 + 16 + 8 + 4 + 2 = 254

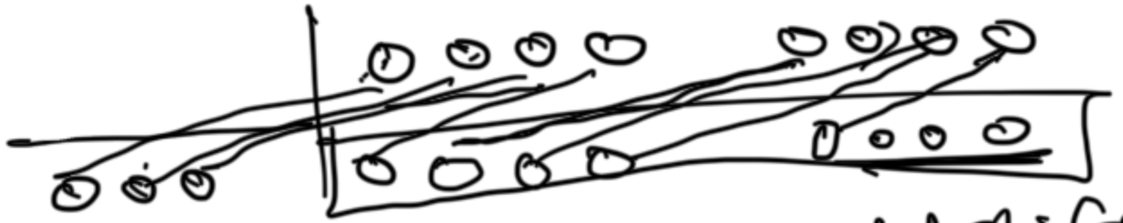
- 64 + 32 + 16 + 8 + 4 + 2

-32 + 16 + 8 + 4 + 2

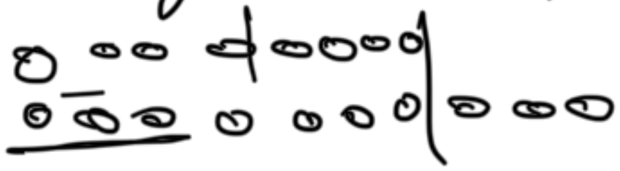
$= -16 + 14 = \boxed{-2}$

left shift operator.

55 << 3

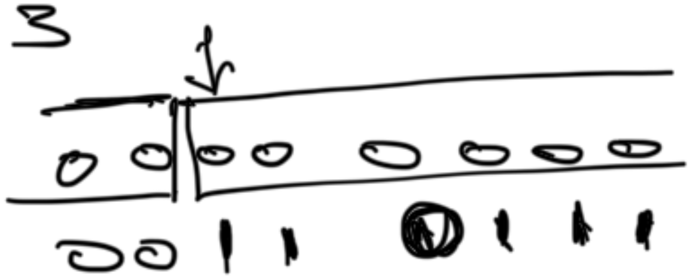


Right shift operator 55 >> 3



L.S.O

55 << 3



~~63 - 32 =~~

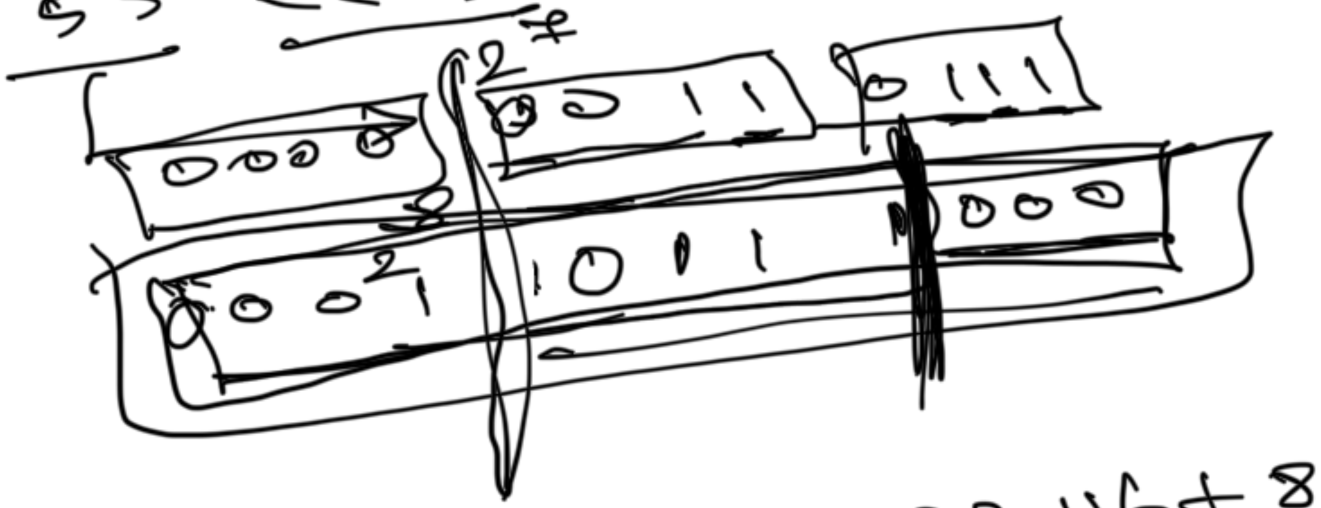
~~63 - 16~~

63 - 8 = 55

32+16+4+2+1

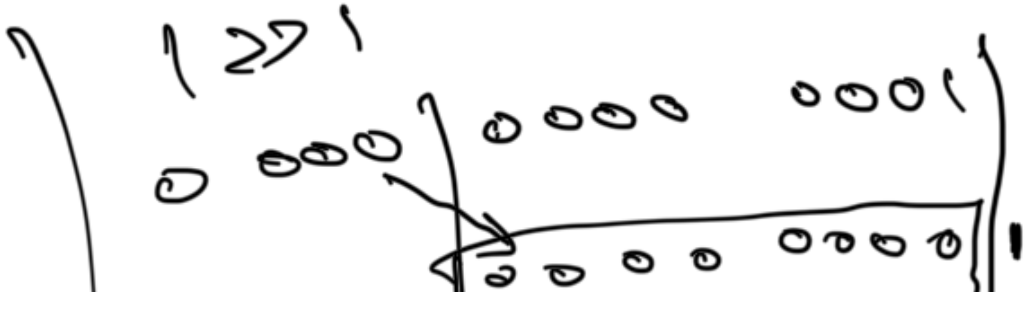
$48 + 7 = \boxed{55}$

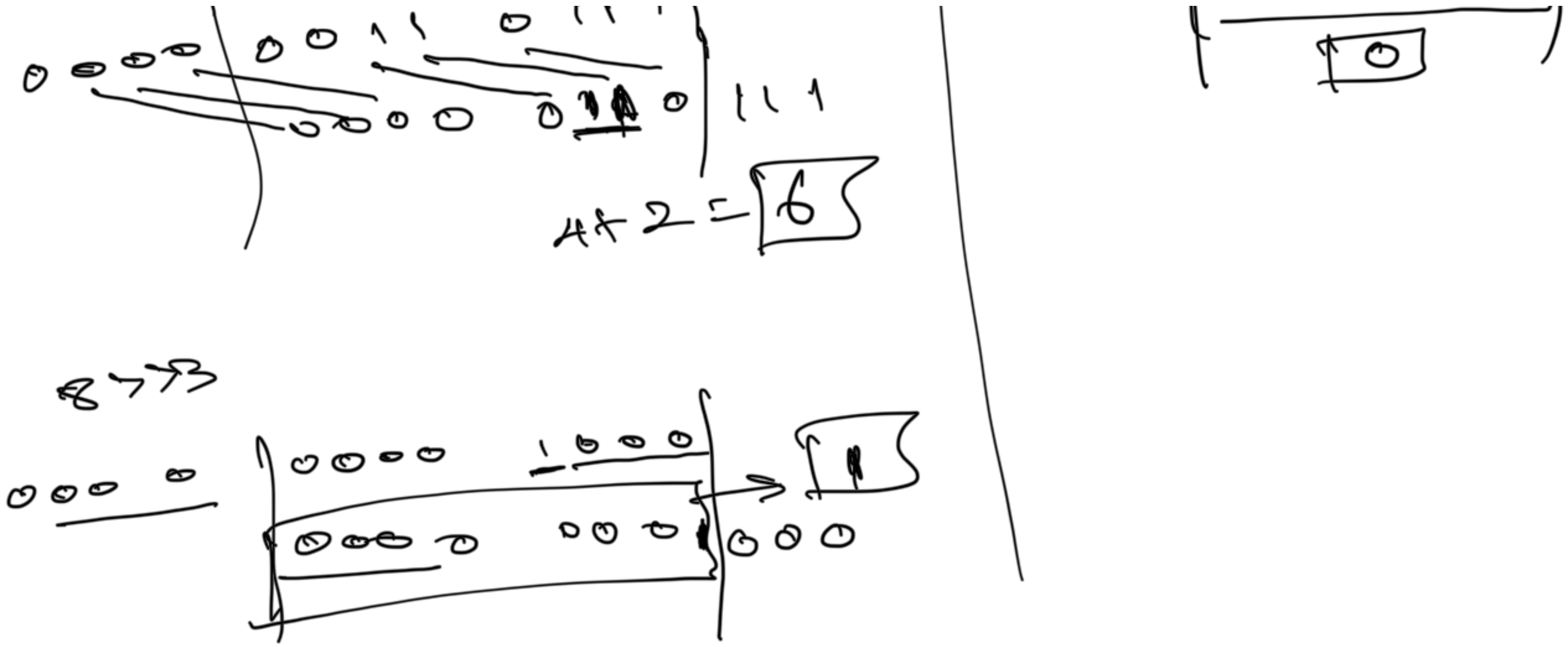
55 << 3



256+128+32+16+8 = 440

55 >> 3





Looping

① for - in

② for - of

③ forEach.

```
array=[10, 20, 30]          → index / key
for (i in array){           o/p: 0
    clg(i)                       1
                                 2
}
```

the value present in

for of loop is going to print the

the array.

* If you want to utilize for-of loop for objects then you are dependent on

1). object.keys( )

2). object.values( )

3). object.entries( )

Task: { } if you utilize this brackets why undefined is coming as your output.

for Each

```
array=[10,20,30]
array.forEach((ele)=>{
  clg(ele)
})
```

o/p: 10
20
30.

X Basics of JS completed X

Intermediate

ES6

① variables → refer to variables.

② Arrow function =>

```
var functionName = () =>{
    clg("This is test arrow fun")
}
functionName()
```