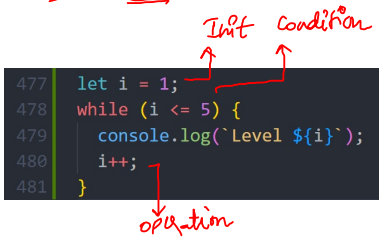
★ While Loop :



→ whenever you want to enter while loop condition must be True.

★ you dont have initialization, operation. These are handled separately.

→ you can use anything, But It depends problem.

① i = 1

② i<=5 ⇒ 1<=5 ⇒ True

③ cl ("Level 1");

④ i++ ⇒ 2

⑤ i<=5 ⇒ 2<=5 ⇒ True

⑥ cl ("Level 2");

⑦ i++ ⇒ 3

⑧ i<=5 ⇒ 3<=5 ⇒ True

⑨ cl ("Level 3");

⑩ i++ ⇒ 4

⑪ 4<=5 ⇒ True

⑫ cl ("Level 4");

⑬ i++ ⇒ 5

⑭ 5<=5 ⇒ True

⑮ cl ("Level 5");

⑯ i++ ⇒ 6

⑰ 6<=5 ⇒ False

★ Sum of digits :

Eg : 1325

Op : 1+3+2+5 = 11

Eg : 565423

Op : 5+6+5+4+2+3 = 25

★ to extract last digit,

digit = num % 10

★ To remove last digit

num = num/10

⇒ we repeat the above operations until num becomes 0.

let num = 1325;

⇒ a+b+c+d

★ Digit Extraction Technique

1325 % 10 = 5

1325/10 = 1325 = 132

132 % 10 = 2

132/10 = 13.2 = 13

13 % 10 = 3

13/10 = 1.3 = 1

1 % 10 = 1

1/10 = 0.1 = 0

0 % 10 = 0

1325 % 10 = 5

132 % 10 = 2

13 % 10 = 3

1 % 10 = 1

0 % 10 = 0

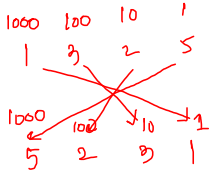
stop

* Reverse of a Number :

eg : 1 3 2 5

op : 5 2 3 1

* why this formula ?



rev = rev * 10 + digit

rev = ~~0~~ ~~5~~ ~~52~~ ~~523~~ 5231

1325 % 10 = 5

→ rev = rev * 10 + digit

= 0 * 10 + 5 = 5

132 % 10 = 2

→ rev = rev * 10 + digit

= 5 * 10 + 2 = 52

13 % 10 = 3

→ rev = rev * 10 + digit

= 52 * 10 + 3 = 523

1 % 10 = 1

→ rev = rev * 10 + digit

= 523 * 10 + 1 = 5231

0 stop

★ HCF of two numbers :

Eg : 75, 90
(a) (b)

highest common factor (HCF/GCD)

Greatest common divisor

→ X is a number such that,

75 % X == 0 and 90 % X == 0

and X is the largest Number

X = 1 to 75

75 to 1

75 % 1 == 0 and 90 % 1 == 0

75 % 2 == 0 and 90 % 2 == 0

75 % 3 == 0 and 90 % 3 == 0

⋮

~~76 77 78 79 ...~~ they will not give 0 remainder

75 % 76 = 75

75 % 77 = 75

75 % 78 = 75

⋮

75 % 3 == 0

90 % 3 == 0

---

75 % 5 == 0

90 % 5 == 0

---

75 % 15 == 0

90 % 15 == 0

layest

→ simple trick is,

Iterate from backwards the first X you encounter will be your GCD.

```
for (let num = min(a,b) ; num > 0, num--) {
    if (a % num == 0 && b % num == 0) {
        console.log(num);
        break;
    }
}
```

→ ~~1~~ 75

75 to 1

75 % 75, 90 % 75

75 % 74, 90 % 74

⋮

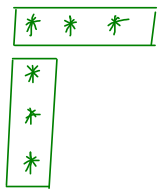
75 % 15, 90 % 15 == 0

stop (break)

★ N stars :

Eg: n = 3

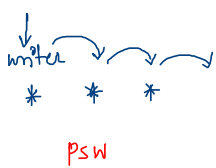
op:



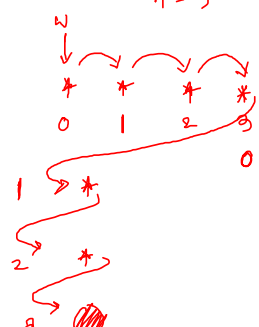
```
for(let i = 0; i < n; i++) {
    process.stdout.write("*");
}
Console.log(); → move to next line
for(let i = 0; i < n; i++) {
    console.log("*");
}
```

★ psw
→ print *
→ move to next pos

★ cl
→ print *
→ move to next line



n = 3



n = 3

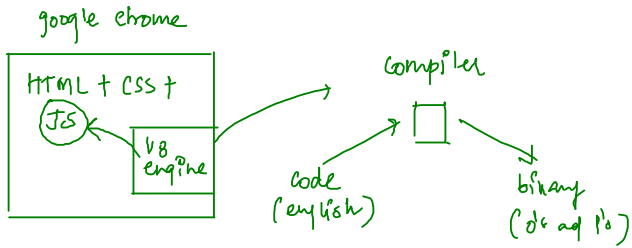
① i = 0 → *

② i = 1 → *

③ i = 2 → *

④ i = 3 (3 < n)
(3 < 3)

★ process.stdout.write :



C/C++ → mingw

Java → JVM

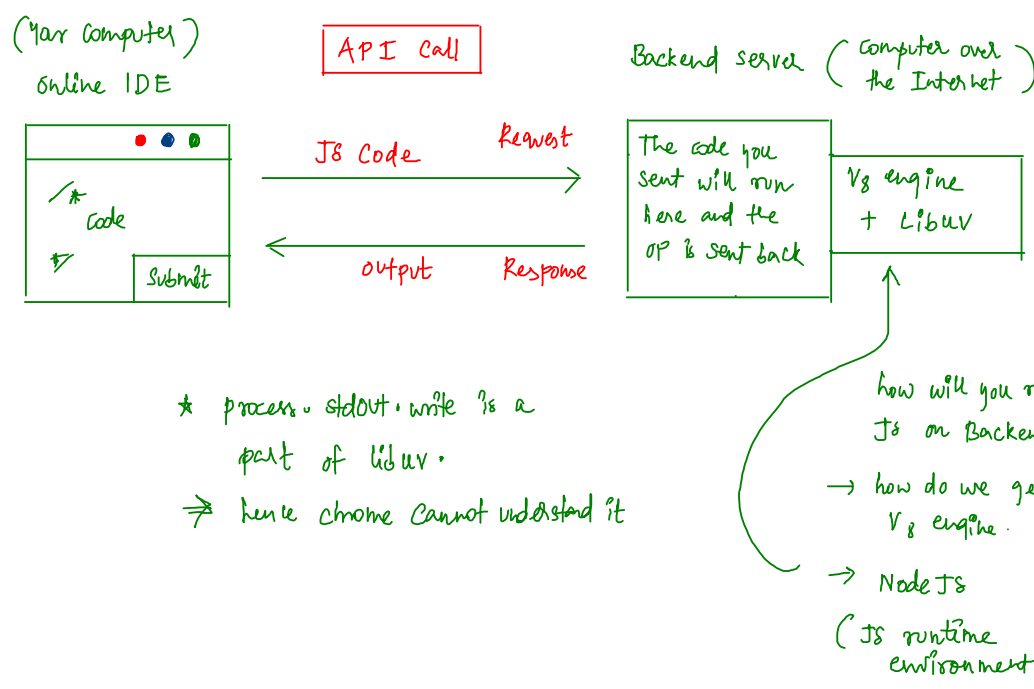
Python → python.exe

⇒ Installing a compiler

JS

→ running it on google chrome

→ google chrome is acting as a compiler


(Your computer)
Online IDE
API Call
Backend server (Computer over the Internet)
Code
Submit
JS Code
Request
output
Response
The code you sent will run here and the op is sent back
V8 engine + LibUV
* process.stdout.write is a part of libuv.
⇒ hence chrome cannot understand it
how will you run JS on Backend server?
→ how do we get V8 engine.
→ NodeJS
(JS runtime environment)