* For Loop :

* In order to enter the for loop, condition must be true

* After creating the code it will perform operation

* operation → condition cycle repeats until the condition becomes false.
  ⇒ i<=5 ⇒ 6<=5 ⇒ false

① Initialisation, i = 1

② i<=5 ⇒ 1<=5 ⇒ true

③ console.log(`Level ${i}`);
⇓
"Level 1"

④ i++ ⇒ i = 2

⑤ i<=5 ⇒ 2<=5 ⇒ true

⑥ console.log(`Level ${i}`);
⇓
"Level 2"

⑦ i++ ⇒ i = 3

⑧ i<=5 ⇒ 3<=5 ⇒ true

⑨ console.log(`Level ${i}`);
"Level 3"

⑩ i++ ⇒ i = 4

⑪ i<=5 ⇒ 4<=5 ⇒ true

⑫ console.log(`Level ${i}`);
"Level 4";

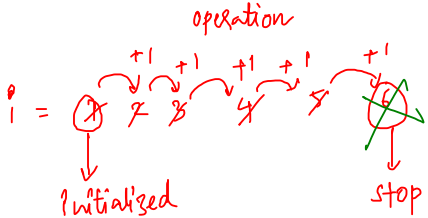
⑬ i++ ⇒ i = 5

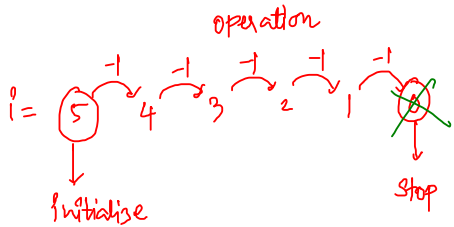
⑭ i<=5 ⇒ 5<=5 ⇒ true

⑮ console.log(`Level ${i}`);
⇓
"Level 5"

⑯ i++ ⇒ i = 6

⑰ i<=5 ⇒ 6<=5 ⇒ false
STOP the Loop






```
i < 6
(or)
for(let i=1; i<=5; i++) {
    cl(Level ${i});
}

i >= 1
(or)
for(let i=5; i>0 ; i--) {
    cl(Level ${i});
}
```

"Level i" → ✗ JS doesn't know that i is a variable

dynamic strings (Template literals)

`Level ${i}` → JS understand that i is a variable

* Sum of 'N' natural numbers :

num = (1) ✗ ✗ ✗ ✗ ..... ✗ (N+1)

(1) → init; +1 +1 +1 → operation; (N+1) → stop

= ✗ ✗ ✗ ✗ ✗ (6) when N = 5

(6) → stop

Eg : n = 5

Op : 1 + 2 + 3 + 4 + 5 = 15

Eg : n = 3

Op : 1 + 2 + 3 = 6 ⇒ Add number form 1 to N

```
let sum = 0;
for( let num = 1 ; num <= N ; num++) {
    sum = sum + num ;
}
```

N = 5, sum = ~~0~~ ~~1~~ ~~3~~ ~~6~~ ~~10~~ 15

(1) num = 1, 1 <= 5
→ sum = sum + num
= 0 + 1 = 1

(2) num++ ⇒ 2 <= 5,
→ sum = sum + num
= 1 + 2 = 3

(3) num++ ⇒ 3 <= 5
→ sum = sum + num
= 3 + 3 = 6

(4) num++ ⇒ 4 <= 5
→ sum = sum + num
= 6 + 4
= 10

(5) num++ ⇒ 5 <= 5
→ sum = sum + num
= 10 + 5
= 15

(6) num++ ⇒ 6 <= 5 ✗

Efficient way :

- The previous approach takes 'N' iterations to calculate the sum.

- Sum of 1st N natural number = $\frac{N(N+1)}{2}$

Eg : $N = 5 \Rightarrow \frac{5(5+1)}{2}$

$\Rightarrow \frac{5 \times 6^{3}}{2_{1}} \Rightarrow 15$

* Even Sum :

Eg: n = 12

op: 2 + 4 + 6 + 8 + 10 + 12 = 42

+2 +2 +2 +2 +2

2 — init

operation

condition

N = 8

num = ~~1~~ 2 ~~3~~ 4 ~~5~~ 6 ~~7~~ ~~8~~ ~~9~~ (9 <= 8) ✗

(8)

Sum = 0 + 2 + 4 + 6 + 8 = 20

num = ~~1~~ 2 ~~3~~ 4 ~~5~~ 6 ~~7~~ ~~8~~ ~~9~~ ~~10~~

(4)

#1:

```
let sum = 0
for (let num = 2; num <= N; num += 2) {
    sum += num
}
```

num = num + 2

num += 2

num++

num = num + 1

⇒ iterations = N/2 (30 mins)

#2:

```
let sum = 0;
for (let num = 1; num <= N; num++) {
    if (num % 2 == 0) {
        sum += num;
    }
}
```

⇒ iterations = N (1 hr)

#3:

Sum of even Numbers until 'N' = $\frac{N(N+2)}{4}$

⇒ not working when N is odd

⇒ when N = 13, answer of N = 12

Eg: N = 12 $\quad \frac{12(12+2)}{4}$

$\Rightarrow \frac{3}{\cancel{12} \times 14}{\cancel{4}\,1} = 42$

2 + 4 + 6 + 8 + 10 + 12

$\Longleftrightarrow$ Same

★ Answer for N-1

Eg: ✗ N = 13 $\quad \frac{13(13+2)}{4}$

$= \frac{13 \times 15}{4}$

$= 48.75$ ✗

2 + 4 + 6 + 8 + 10 + 12

Eg: N = 20 $\quad \frac{20(20+2)}{4}$

$\Rightarrow \frac{20 \times 22}{4} = 110$

2 + 4 + 6 + 8 + 10 + 12 + 14 + 16 + 18 + 20

$\Longleftrightarrow$ Same

Eg: ✗ N = 21 $\quad \frac{21(21+2)}{4}$

N = N − 1

= 20 $\Rightarrow \frac{21 \times 23}{4} \neq 5.25 \times 23$

⇒ Some fraction ✗

2 + 4 + 6 + 8 + 10 + 12 + 14 + 16 + 18 + 20

break :

⇒ terminate / come out of the current loop in execution.

```
452  for (let i = 1; i <= 5; i++) {
453    if (i == 3) {
454      console.log("Sorry you cannot move forward");
455      break;
456    }
457
458    console.log(`Level ${i}`);
459  }
```

① i = 1, 1 < = 5
→ cl (Level ${i})

② i++, 2 < = 5
→ cl (Level ${i})

③ i++, 3 < = 5
→ cl ("Sorry you cannot move forward");
→ break ⇒ forget everything
Just come out.

continue :

⇒ move to the next iteration of current loop in execution by ignoring all the code below it.

```
463  for (let i = 1; i <= 5; i++) {
464    if (i == 3) {
465      console.log("You can skip this level");
466      continue;
467    }
468    console.log(`Level ${i}`);
469  }
```

① i = 1, 1 <= 5
→ cl (Level ${i})

② i++, 2 <= 5
→ cl (Level ${i})

③ i++, 3 <= 5
→ cl ("You can skip this Level");
→ continue ;

④ i++, 4 <= 5
→ cl (Level ${i})

⑤ i++, 5 <= 5
→ cl (Level ${i})

⑥ i++, 6 <= 5 (stop)

* Check Prime : (which is divisible only by 1 and itself)

Eg: n = 5

op: true

Eg: n = 13

op: true

Q: Check 13 is prime or not ?

[2, 12] → [2, N-1]

↓

Is there any number that divides 13 ?

13 % 2 == 0 ✗
13 % 3 == 0 ✗
13 % 4 == 0 ✗
13 % 5 == 0 ✗
13 % 6 == 0 ✗
13 % 7 == 0 ✗
13 % 8 == 0 ✗
13 % 9 == 0 ✗
13 % 10 == 0 ✗
13 % 11 == 0 ✗
13 % 12 == 0 ✗

→ I have tried and failed to a number which divides 13.

Prime ↑

Q: check 49 is prime or not ?

[2, 48]

49 % 2 == 0 ✗
49 % 3 == 0 ✗
49 % 4 == 0 ✗
49 % 5 == 0 ✗
49 % 6 == 0 ✗
49 % 7 == 0 ✓

→ I found a number that divides 49 ⇒ not prime

⇒ Convert to the code; (flag variable Technique)

↳ I did not find any num which divides N

```
let isFound = false;          num < N
                              (or)
for(let num = 2;  num <= N-1 ;  num++) {

    if(N % num == 0) {
        isFound = true;
        break;
    }
}
if(isFound == true) {
    cl("not prime");
}
else {
    cl("a prime");
}
```

// here how do you know that you found a number which divides N ?

* you can come out using break or normal termination, how can you differentiate between them ?