Big O Notation - What does it mean?

> Can you write a program to sum from 1 to n? (n is the input your type from your keyboard)

This is simple question, I think you can do it. Or if you can't do it, you should learn basic programming before reading this section.

```
function addUpTo(n) {
  let total = 0;
  for (let i = 1; i <= n; i++) total += i;
  return total;
}
```

Okayyy, you can do it like me. Now, let's test our code.

```
addUpTo(10);
// 55
addUpTo(100);
// 550
```

Yea, you see, we got the result. But, Is it better? "How long" does it take? Now we have a simple measuring on it. We use **performance.now()** - a javascript feature. Now let's do it.

```
const t1 = performance.now();
addUpTo(1000000000); // 1 000 000 000
const t2 = performance.now();
console.log((t2 - t1) / 1000); // We got 0.7 seconds.
```

From 1 to 1 billion, we need 0.7 seconds to do it. In better way, with formula, you can do it better.

```
addUpTo(n) = 1 + 2 + 3 + .... + n - 1 + n
addUpTo(n) = n + n - 1 + n - 2 + ... + 2 + 1 // reverse it
=> 2*addUpTo(n) = (n + 1) + (n + 1) + ....
                = n * (n+1)
=> addUpTo(n) = n * (n + 1) / 2
```

To test it, we can write a function to check how long this formula take. I think it's very simple to do it, right?

```
function addUpToEnhanced(n) {
  return (n * (n + 1)) / 2;
}
```

```
const t3 = performance.now();
addUpToEnhanced(1000000000); // 1 000 000 000
const t4 = performance.now();
console.log((t4 - t3) / 1000); // We got 0 seconds 🤣🤣.
```

It's too short to measure =))). And you see, the first one take 0.7 seconds calc a sum from 1 to 1 000 000 000, but the other only takes 0.000000 (too small to know). So, anytime we want to measure our code, we need write a function, calc it 2 times and measure it by using **performance.now()**. NO, we NEVER do it.

> In computer architecture, we have many properties (components) to make it. But we should know that: ***the same hardware, not the same properties, not the same speed, not the same engineer did it!***