# BLOCK NONCE #11: Document Your Code

## What?

**DOCUMENTATION = TL;DR.**

## Why?

It gives people the option to use your code without digging through the details.

## How?

**Documnetation comments (`///`)**: describes what the function does and how to use it — for someone reading the API, not the implementation.

```
/// Returns the factorial of a non-negative integer.
///
/// # Arguments
/// * `n` - The number to compute the factorial for.
///
/// # Panics
/// Panics if `n` is greater than 20 to avoid overflow.
fn factorial(n: u64) -> u64 {
```

**Implementation comments (`//`)**: explains how the code works internally — for someone reading the logic.

```
fn calculate_discount(price: u64, is_vip: bool) -> u64 {
    if is_vip { price * 0.8 } // VIP users get a 20% discount.
    else { price }
}
```

**TIP**💡

1. Remember- the code **doesn't** document itself, at least for the averge human.
2. Put effort into **naming** your variables and functions. Stuck? Ask your good-at-names teammate for inspiration. If it feels impossibly hard - that might be a design problem in disguise.
3. Break your code into **"paragraphs"** - logical chunks with blank lines. If it makes sense, add a short comment on top to explain what that "paragraph" does.

**Call For Action!**📢💨 Pick a random file from your codebase. Can you understand what it does without reading all of it? If not, add documentation according to the key points above!