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?

Documentation comments (///): describes what the function does and how to use it — for someone reading the API, not the implementation.
Implementation comments (//): explains how the code works internally — for someone reading the logic. Break your code into **"paragraphs"** - logical chunks with blank lines and add an implementation comment on top.

```rust
/// Calculate the total price for a given order.
fn calculate_price(user: User, order: Order) -> u64 {

    // Sum the price of all items in the order.
    let mut price = order.items.iter().map(|i| i.price).sum::<u64>();

    // Apply discount if the user is eligible.
    let discount = calculate_discount(&user, price);
    price-=discount;

    // Add tax based on the user's tax rate.
    price += calculate_tax(&user, price);

    price
}
```

TIP💡

1. Remember- the code **doesn't** document itself, at least for the average human. Make sure you document your structs/enums, their members and functions.
2. Put effort into **naming** your variables and functions. Stuck? Ask your good-at-names teammate for inspiration or your favourite AI tool. If it feels impossibly hard - that might be a design problem in disguise.

Call For Action!📢💨 Next time you're frustrated by undocumented code? Don't just complain — add a doc and save someone else the pain.

#starkcode