



## BLOCK NONCE #11: Document Your Code

### What?

DOCUMENTATION = TL;DR.

### Why?

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

### How?

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

**Implementation comments** (//): explain how the code works internally — for someone reading the logic. Break your code into “**paragraphs**” - logical chunks separated by blank lines and add an implementation comment on top.



```
/// Calculates 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();

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

    // Add VAT according to the user's profile.
    price + calculate_vat(&user, price)
}
```

#### TIP

1. Remember - the code usually **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 for inspiration. 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.