Block By Block.

BLOCK NONCE #9: Use Early Return

Why?

Because your main logic shouldn't be wrapped in layers of `if`:

```rust
fn watch_movie(user: User) -> Result<(), WatchMovieError> {
    if user.is_authenticated() {
        if user.is_premium_plan() {
            if user.pay().is_ok() {
                user.watch_movie()
            }
        }
    }
    Err(WatchMovieError::AccessDenied)
```

How?

Just flip the condition, add an early return and let the main logic breath.

```rust
fn watch_movie(user: User) -> Result<(), WatchMovieError> {
    if !user.is_authenticated() {
        return Err(WatchMovieError::NotAuthenticated);
    }
    if !user.is_premium_plan() {
        return Err(WatchMovieError::NonPremiumUser);
    }
    user.pay()?;
    user.watch_movie()
```

This code is easier to understand - a simple list of checks followed by the main logic.

TIP💡

1. The `?` operator is another form of early return. It's a shortcut for:

   ```rust
       match user.pay() => { Ok(x) => x, Err(e) => return Err(e); }
   ```

2. **Note:** `continue` = early return for loops!
3. If your function need to handle one **enum** variant, use the let-else syntax:

```rust
let AuthStatus::Authenticated(user) = auth else {
    return Err(AccessError::NotAllowed);
};
```

4. Do not early return in the middle of a big function! It'll complicate the flow instead of simplifying it. Returning in the middle of a function means most of the function's logic is relevant — refactor it instead of returning early.