Block By Block.

BLOCK NONCE #7: Rust enums are great!

What?

Everyone knows what enums are — but knowing when to use them isn't always obvious. Here's the rule of thumb: if your type can only take on a few known values, you should enum it.

Why?

Enums make your code self-documenting, more readable, and give the compiler a chance to catch your bugs before your teammate does.

How?

```
struct LoggingConfig {
    enabled: bool,
    // If not enabled, we don't know the level.
    level: Option<LogLevel>,
}
impl LoggingConfig {
    fn log(&self, message: &str) {
        if self.enabled {
            // Why taking the risk of panicking here?
            let level = self.level.unwrap();
            log(level, message);
        }
    }
}
```

Let's just enum it:

```
enum LoggingConfig {
    Disabled,
    Enabled(LogLevel),
}
impl LoggingConfig {
    fn log(&self, message: &str) {
          if let Self::Enabled(level) = self { // No risk of panicking.
              log(level, message);
        }
    }
}
```

TIP💡 Planning to (de)serialize? Reach for a struct, not an enum. Enums tend to confuse other languages.