THE NON-BORING PARTS OF RUST

RUST TURKU MEETUP

Presented by Jani Anttonen

RUST TURKU MEETUP

WHO AM I?

- Software Developer
- Discovered Rust in 2017
- Working with distributed technologies in Equilibrium
- We need more people to run this community, so if you want to keep the next meetup or have any other ideas, hit me up: jani@equilibrium.co

EQUILIBRIUM

- Operating since 2017
- Distributed technologies (p2p, blockchain)
- Products and consulting

SCHEDULE

18:15 -> This

19:00 -> "From Conventional OOP to Rust"

19:45 -> "Afterbeers", open discussion, hanging out

We will all get our Rust love story

TRAITS

It is very easy in Rust to hide complexity by reimplementing basically any logic used by your struct

TRAITS

So... what if...

TRAITS

So... what if...

1 + 1 = 0

TRAITS

So... what if...

1 + 1 = 0

1 - 1 = 2

TRAITS

So... what if...

1 + 1 = 0

1 - 1 = 2

8 * 2 = 4

TRAITS

So... what if...

1 + 1 = 0

1 - 1 = 2

8 * 2 = 4

8 / 2 = 16

TRAITS

So... what if...

1 + 1 = 0

1 - 1 = 2

8 * 2 = 4

8 / 2 = 16

cats = dogs?

```rust
use std::fmt;
use std::fmt::Display;
use std::ops::{Add, Div, Mul, Sub};

/**
 * Because some men just want to see the world burn
 */
struct OpInt(i32);

impl Div for OpInt {
    type Output = Self;
    fn div(self, other: Self) -> Self {
        Self {
            0: self.0 * other.0,
        }
    }
}

fn main() {
    let opp1: OpInt = OpInt(4);
    let opp2: OpInt = OpInt(3);
    println!("{}", opp1 / opp2); // 12
}
```

TRAITS

Implementing traits like Display for your Struct is sort of like defining to_string in Java. Also good if you like to create concatenation logic for your Struct using the operator +, for example

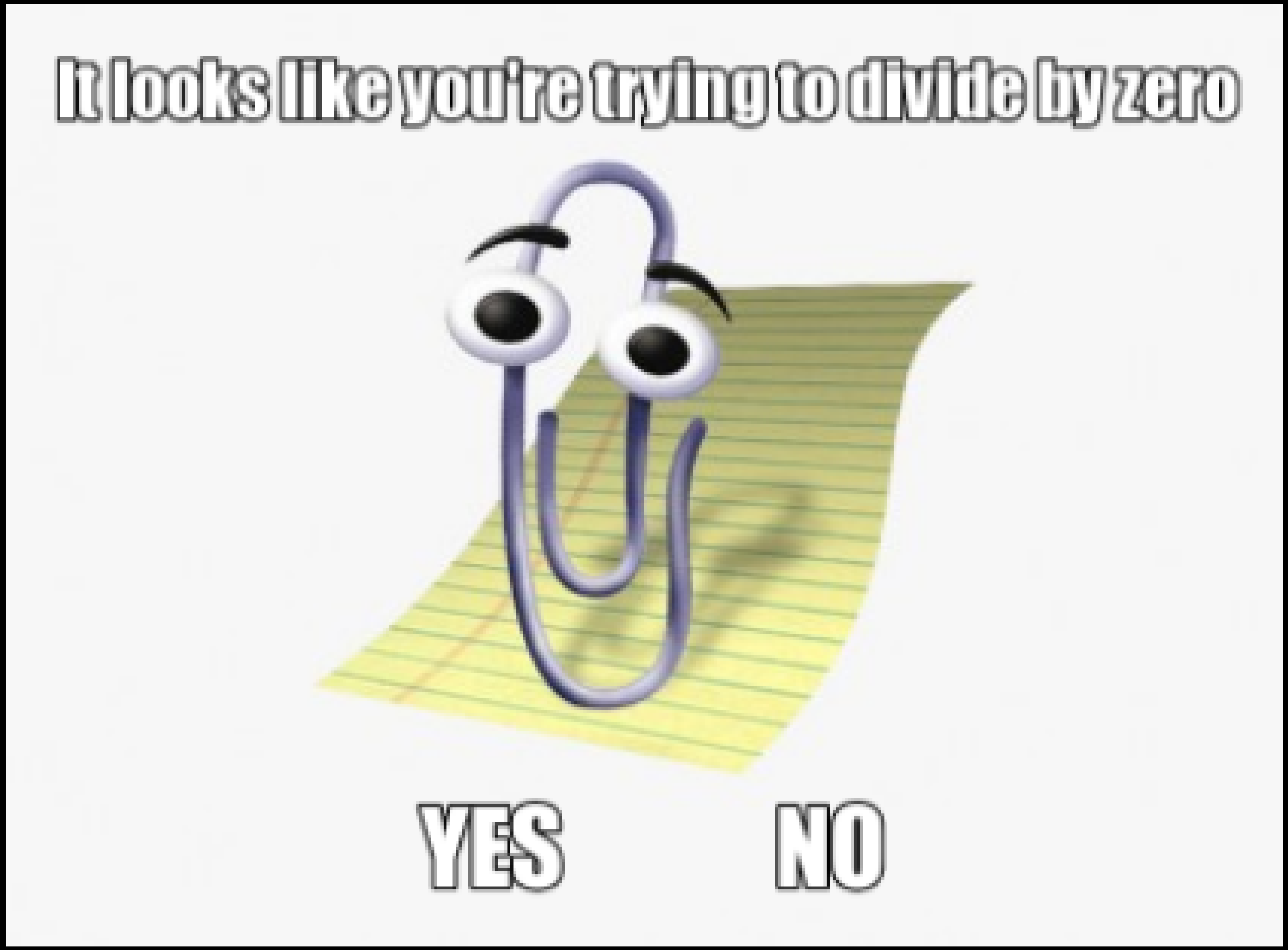
COMMENTS ARE A FIRST-CLASS CITIZEN

- Markdown
- Code examples must compile

COMMENTS ARE A FIRST-CLASS CITIZEN

No more outdated code samples in documentation!

THE COMPILER HINTS

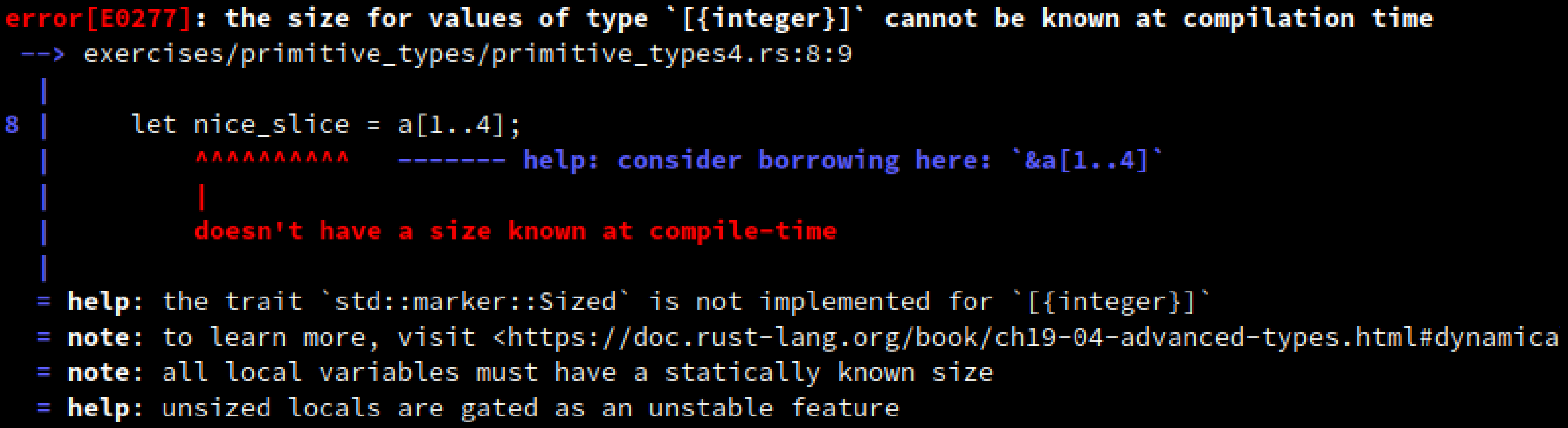


RUST TURKU MEETUP

THE COMPILER HINTS

No, but seriously.
Almost every time I've tried doing something stupid the compiler throws an error with step-by-step instructions on how not to be stupid.

THE COMPILER HINTS



```
error[E0277]: the size for values of type `[{integer}]` cannot be known at compilation time
 --> exercises/primitive_types/primitive_types4.rs:8:9
  |
8 |     let nice_slice = a[1..4];
  |         ^^^^^^^^^^   ------- help: consider borrowing here: `&a[1..4]`
  |         |
  |         doesn't have a size known at compile-time
  |
  = help: the trait `std::marker::Sized` is not implemented for `[{integer}]`
  = note: to learn more, visit <https://doc.rust-lang.org/book/ch19-04-advanced-types.html#dynamica
  = note: all local variables must have a statically known size
  = help: unsized locals are gated as an unstable feature
```

THE COMPILER HINTS

So yeah, that's awesome.

COOL PROJECTS

- **Ruma** https://github.com/ruma/ruma – A Matrix messaging server
- **Weld** https://www.weld.rs/ – A data analytics runtime
- **Amethyst** https://amethyst.rs/ – A game engine written in Rust
- **Sonic** https://github.com/valeriansaliou/sonic – A fast text search backend, alternative to Elasticsearch

RUST TURKU MEETUP

THANK YOU!