Breaking limits of Rust

Exploring Static Reflection for Powerful Code Generation

Rust Aarhus – April 2023 - Wojciech Polak

Introduction

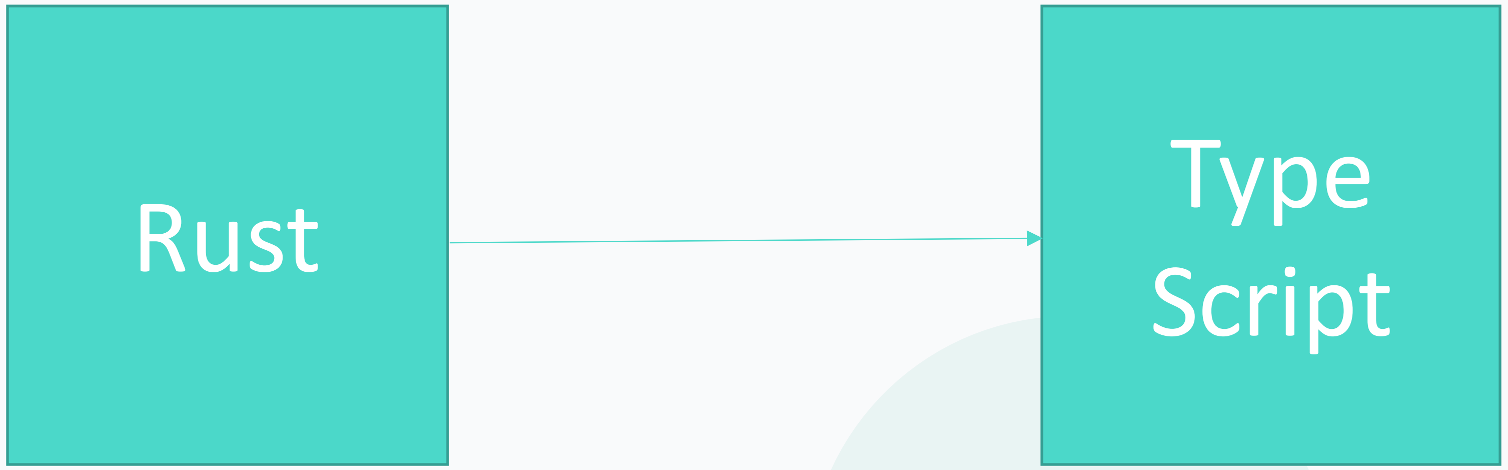
- Wojciech Polak (prefer Wojtek [v-oy-t-eh-c])
- @frondeus on Github
- CS graduate at Wrocław University of Science and Technology (Poland)
- Senior Full Stack Engineer at Impero A/S
- Writing Rust backend (and React frontend) to make Compliance Simplified

Problem

Rust Aarhus – April 2023 – Wojciech Polak

Problem

- We have a backend in Rust and Frontend in React (Typescript)
- How can we keep the Data Transfer Objects between those two realms in sync?



Possible solutions

- Procedural Macros
- Parsing code externally
- Compiler plugins
- **Static Reflection**

Procedural Macros

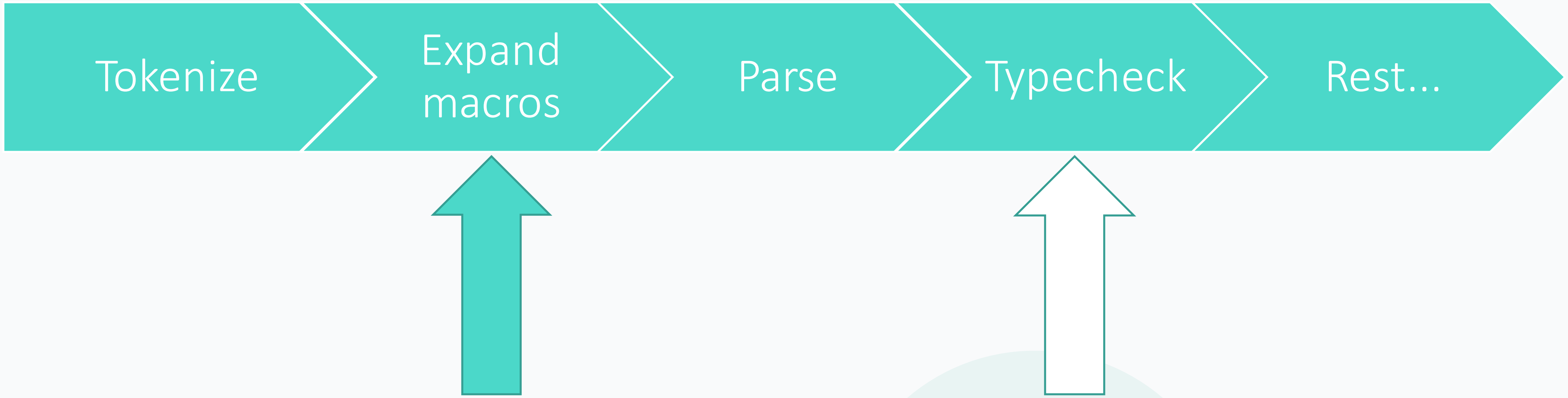
Rust Aarhus – April 2023 – Wojciech Polak

Procedural Macros

- #[derive(MyTrait)] or #[my_attribute]
- It operates on TokenStream
- struct MyDataType { foo: Foo }
- Ident(`struct`), Ident(`MyDataType`), Group('{', [Ident(`foo`), Punct(`:`), Ident(`Foo`)])
- We can then use `syn` crate to parse the TokenStream into data structure

Procedural Macros

- Limitation



- We have no information about types or where those are located

Procedural Macros

- There is a solution!
- Move the logic into the runtime by using traits :)
- That's exactly what for example `ts-rs` does.
- **But**
- You need to implement your trait for every type that is in the interface
- That's including the external types from your dependencies
- You need to "run" it – for example in #[test]

Possible solutions

- ~~Procedural Macros~~
- Parsing code externally
- Compiler plugins
- **Static Reflection**

Parsing code by yourself

Rust Aarhus – April 2023 – Wojciech Polak

Parsing code by yourself

- That's what we do currently in our production code
- Meet **Typebinder -** https://github.com/impero-com/typebinder
- We love it! It makes our life much, much easier.
- **But**
- We had to re-implement module resolution by ourselves.
- Which is prone to changes in Rust.
- Also it's not the easiest project to maintain
- You still lack the external type info

Possible solutions

- ~~Procedural Macros~~
- ~~Parsing code externally~~
- Compiler plugins
- **Static Reflection**

Compiler Plugin

Rust Aarhus – April 2023 – Wojciech Polak

Compiler Plugin

- Approach used by for example Cargo Clippy
- All info about the types right there
- Module resolution for free
- https://doc.rust-lang.org/1.16.0/book/compiler-plugins.html
- **BUT**
- Nightly only
- Very prone to changes in Rust compiler

Possible solutions

- ~~Procedural Macros~~
- ~~Parsing code externally~~
- ~~Compiler plugins~~
- **Static Reflection**

Credits: "Parsing Rust Considered Harmful" by Sasha Pourcelot @scrabsha

Rust Aarhus – April 2023 – Wojciech Polak

Also: "Reasoning about Rust: an introduction to Rustdoc's JSON format" by Luca Palmieri

Rust Aarhus – April 2023 – Wojciech Polak

Static Reflection

Rust Aarhus – April 2023 – Wojciech Polak

Static Reflection

- Build information about rust types, module structure via s**tatic analysis tool**
- Use that information to generate `&'static` data available in the Rust code
- Call that data in external exe, for example in `cargo xtask` to generate typebindings
- …
- But How?

Static Reflection

- Use **nightly** `cargo doc` that can generate JSON output instead of standard markdown files.
- The JSON is easy to deserialize :)
- There is even a crate that provides data types!
  https://crates.io/crates/rustdoc-types
- Therefore, introducing
  Erised – Static Reflection for Rust
  https://github.com/frondeus/erised
- Still work in progress

Live demo
& Short explanation

Erised – Static Reflection

- It's still experimental
- Static also can mean that it's zero-cost
  (Or the cost is amortized during the compilation)
- No runtime needed!
- It doesn't work with `build.rs` (unfortunately)
- You don't need a trait for every type used in the field.
- JSON format is unstable but it's closer to the stabilization than compiler plugin
- It also handles traits, methods (and soon modules, consts)

Q & A

Rust Aarhus – April 2023 – Wojciech Polak

We are hiring
career@impero.com

Rust Aarhus – April 2023 – Wojciech Polak

IMPERO

.COMPLIANCE .SIMPLIFIED