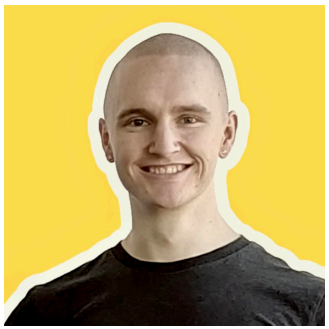
Rust Workshop Light

Etwas über mich



- Wissenschaftlicher Assistent und
  Masterstudent in Informatik an der ZHAW
- Nerd Stuff: Rust, Linux, Tastaturen, Versionskontrolle etc.
- Rust-Erfahrung in verschiedenen Anwendungsbereichen:
  - Server / web APIs
  - Compiler Backend (Cranelift)
  - Frontend mit WebAssembly (Leptos)
  - Linux Kernel Development
  - (wenig) Bare-Metal-Embedded

Rust Weiterbildung an der ZHAW

- Über sechs Kurseinheiten: Grundlagen bis zu Fortgeschrittenes, Tooling und diverse Abschlussprojekte
- Nächstes Startdatum: September 2025
- Nicht nur für “Embedded Developers”, aber C/C++ Verständnis (Speicherverwaltung) wird vorausgesetzt.

- Nehmt ein paar Arbeitskollegen mit, alleine ist es schwierig dem Chef eine neue Sprache zu verkaufen 😉

Rust at a Glance

- Memory-safe systems programming language without garbage collection
- Conceived at Mozilla, version 1.0 released in 2015
- Open-source, developed by a community and supported by the Rust Foundation
- Strong backwards-compatibility guarantees

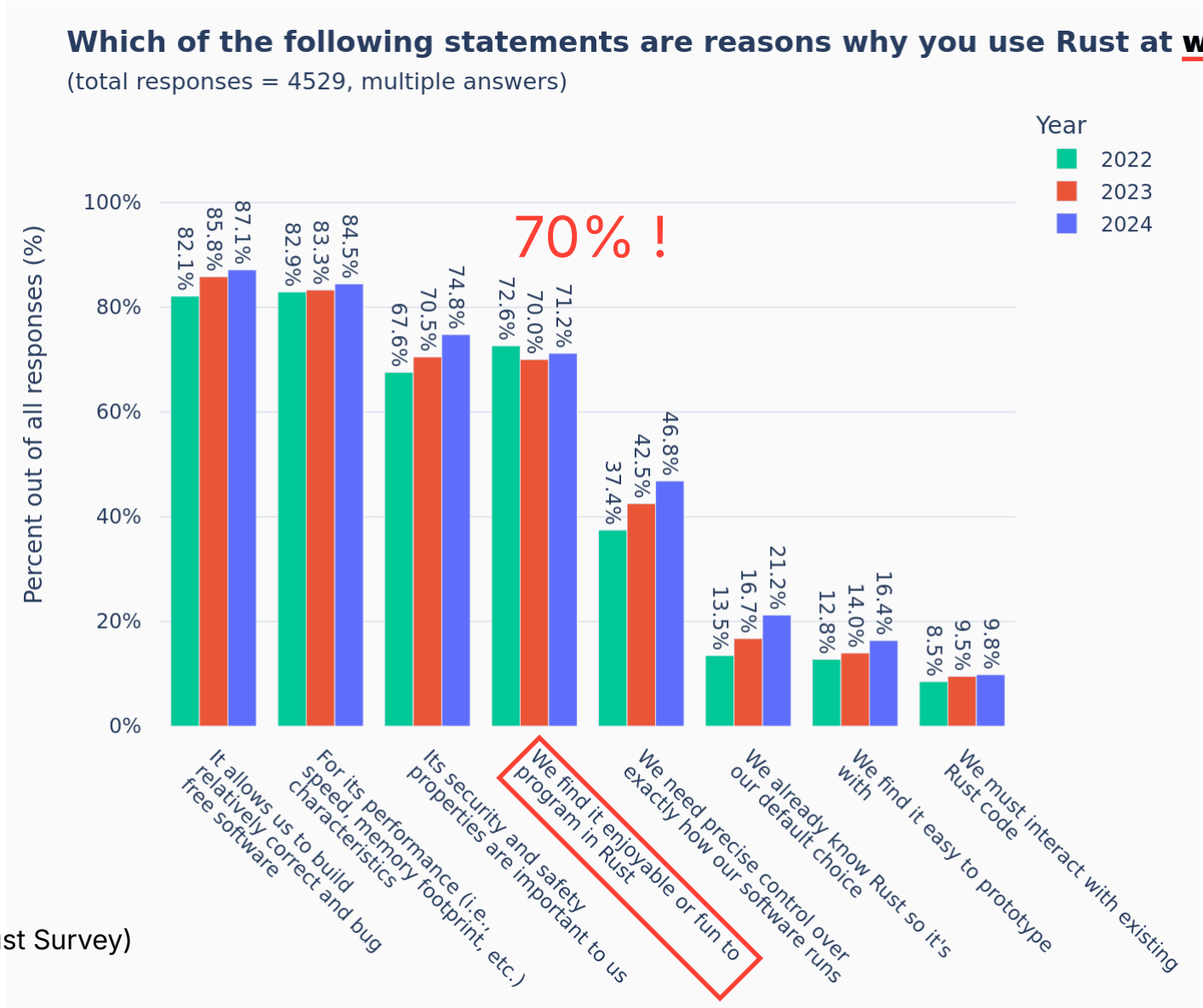
Motto: “A language empowering everyone to build reliable and efficient software.”

First-class Tooling

A typical Rust installation includes the official:

- compiler (rustc)
- package manager (cargo)
- formatter (rustfmt)
- linter (clippy)
- documentation generator (rustdoc)
- toolchain manager (rustup)
- LSP (rust-analyzer) `rustup component add rust-analyzer`

Fun!



(2024 State of Rust Survey)

Microsoft

→ 70% of security vulnerabilities are memory related

> It’s time to halt starting any new projects in C/C++ and use Rust for those scenarios where a non-GC language is required. For the sake of security and reliability. The industry should declare those languages as deprecated.
>
> — Mark Russinovich, CTO Microsoft Azure (2022)

Google

→ 70% of security vulnerabilities are memory related

> Rust teams at Google are as productive as ones using Go, and more than twice as productive as teams using C++.
>
> — Lars Bergström, Director of Engineering, Google Android (2024)

Reasons not to use Rust

Learning curve 🧠

→ temporary

Specific libraries 📚

→ machine learning, GUI programming etc.

Compile times ⏳

→ no problem in small code bases

→ in big code bases, you *really want*
all the help you can get from a compiler

Ownership

How does Rust achieve safety with the performance of C?

→ Ownership and lifetimes.
The validity of each pointer (aka. *reference*) is tracked at compile time.

```
let x;
{
    let y = 42;     // 👶 lifetime of y starts
    x = &y;         // 🔗 x is associated with lifetime of y
}                   // 💀 lifetime of y ends
println!("{}", x); // ❌ error: y does not live long enough
```

Differentiating Language Features

What are some programming paradigms that shape the way you code in Rust every day?

Enums (Algebraic Data Types)

```
enum IP {
    V4([4; u8]),
    V6([16; u8]),
}
```

inspired by functional programming, often used instead of inheritance

Traits (Type Classes)

```
trait Comparable {
    fn compare(&self, other: &Self) -> i32;
}

fn sort<T: Comparable>(slice: &mut [T]);
```

inspired by functional programming, very flexible interfaces

Macros

```
// so-called "derive macro"
#[derive(cool_library::CoolTrait)]
struct MyCustomType { /* fields... */ }

fn main() {
    MyCustomType::generated_function();
}
```

inspired by LISP, they operate on token streams

Async

```rust
async fn do_io_heavy_thing() { /**/ }

#[tokio::main(flavor = "current_thread")]
async fn main() {
    tokio::spawn(do_io_heavy_thing());
    do_io_heavy_thing().await;
}
```

runtime agnostic (tokio, embassy) best performance for IO-bound tasks

Jetzt seid Ihr dran!

Es gibt mehrere Tutorials zur Auswahl
mit verschiedenen Schwerpunkten.

Experimentiert, stellt Fragen, tauscht Euch aus!

rwl.buenzli.dev

“Rust Workshop Light dot Bünzli dot dev”

- `#macros` `#enums` Erstelle ein Kommandozeilenprogramm
- `#macros` Hantiere mit Datenformaten wie JSON oder TOML
- `#traits` Schreibe eine Middleware für einen Webserver
- `#async` Erledige IO intensive Arbeit effizient
- `#enums` Erweitere Typen der Standardbibliothek