History of Rust

Say hi to Ferris!

Outline

Chapter 1
History at Firefox (Mozilla)

Browser Wars

After Netscape Navigator vs. IE, we had Firefox vs. Chrome

- Browsers are gigantic pieces of technology
- Practically replicating **OS features**
- Firefox's mission: make it fast

2006: The birth of Rust

Speeding up the browser needed a new systems language

- **Graydon Hoare** starts Rust in 2006
- Rust is inspired by the old technologies: Newsqueak, Limbo, Cyclone

2006: The birth of Rust

Speeding up the browser needed a new systems language

- **Graydon Hoare** starts Rust in 2006
- Rust is inspired by the old technologies: Newsqueak, Limbo, Cyclone
- Graydon also authored Monotone — inspired Git
- Graydon later worked on **Swift** at Apple

2006: The birth of Rust

Speeding up the browser needed a new systems language

- **Graydon Hoare** starts Rust in 2006
- Rust is inspired by the old technologies: Newsqueak, Limbo, Cyclone
- Graydon also authored Monotone — inspired Git
- Graydon later worked on **Swift** at Apple
- 2009: Mozilla offically adopts Rust

Servo, Quantum, independence

- **Servo** — Mozilla new engine in Rust
- Servo itself never ships — but its pieces (Stylo) land in Quantum (2017)

Servo, Quantum, independence

- **Servo** — Mozilla new engine in Rust
- Servo itself never ships — but its pieces (Stylo) land in Quantum (2017)
- **2020**: Pandemic strikes and Mozilla terminates entire Rust team
- Rust becomes independent — the **Rust Foundation** (2021)

Chapter 2
Systems Languages

Everything is built on C/C++

Decades of infrastructure, decades of memory bugs

- Manual memory management — powerful, but error-prone

Everything is built on C/C++

Decades of infrastructure, decades of memory bugs

- Manual memory management — powerful, but error-prone
- Memory-safety issues common enough that **governments issue advisories** (NSA, CISA) recommending memory-safe languages

What makes Rust special

- Zero-cost abstractions — high-level code, no runtime tax

What makes Rust special

- Zero-cost abstractions — high-level code, no runtime tax
- Memory safety without a garbage collector (ownership & borrowing)

What makes Rust special

- Zero-cost abstractions — high-level code, no runtime tax
- Memory safety without a garbage collector (ownership & borrowing)
- High-level ergonomics with low-level control

What makes Rust special

- Zero-cost abstractions — high-level code, no runtime tax
- Memory safety without a garbage collector (ownership & borrowing)
- High-level ergonomics with low-level control
- Compiles via **LLVM**

What makes Rust special

- Zero-cost abstractions — high-level code, no runtime tax
- Memory safety without a garbage collector (ownership & borrowing)
- High-level ergonomics with low-level control
- Compiles via **LLVM**

One asterisk: `async`/`await` is not fully zero-cost

The internet's reaction

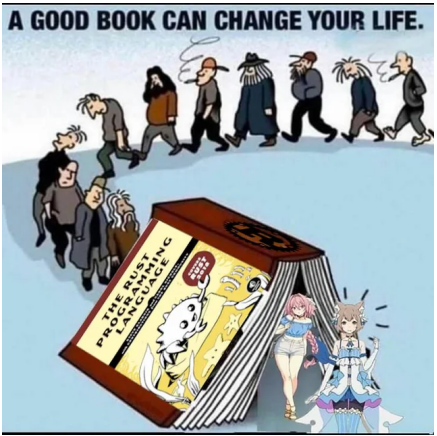
Chapter 3
The Rise of Rust

Community culture

Rust's growth is as much *cultural* as technical

- Explicit error message written for humans
- A strict **Code of Conduct** from early on
- Rustaceans, Ferris the crab, and yes — programming socks

A good book can change your life



“Rewrite it in Rust”

- Evangelism becomes a meme: **“rewrite it in Rust”** (aka. *crabbing*)
- Every popular tool eventually gets a Rust rewrite proposed
- Evangelism cuts both ways — enthusiasm vs. fatigue

Every. Single. Time.

PROGRAM

PROGRAM WRITTEN IN ©RUST™

Growing pains

- **Actix drama** (2020) — maintainer backlash over `unsafe` code criticism

Growing pains

- **Actix drama** (2020) — maintainer backlash over `unsafe` code criticism
- **Trademark drama** (2023) — Rust Foundation's policy proposal sparks community revolt

Growing pains

- **Actix drama** (2020) — maintainer backlash over `unsafe` code criticism
- **Trademark drama** (2023) — Rust Foundation's policy proposal sparks community revolt

Growth exposes governance — not just code quality

Chapter 4
Alternatives: Zig

Zig: the challenger

- Started by **Andrew Kelley** (2016) as simpler alternative to C
- No hidden control flow, no hidden allocations
- `comptime` instead of macros — explicit over implicit

Meanwhile, poor Ada

Bun & the AI rewrite

- **Bun** — a JS runtime written in Zig, meteoric rise since 2022
- Its put Zig on the map outside systems programming

Bun & the AI rewrite

- **Bun** — a JS runtime written in Zig, meteoric rise since 2022
- Its put Zig on the map outside systems programming
- Then: Anthropic acquirement and an AI-driven rewrite — evangelism meets automation

The pinnacle of Rust evangelism

Even **alternative** innitiatives get converted back to Rust

Rust won the systems-language conversation — every challenger is now measured against it

Thank You

References

- *The Rust Programming Language*, Steve Klabnik & Carol Nichols.
- Mozilla, *Servo: a parallel browser engine*, servo.org.
- Mozilla, *Quantum Project*, 2017.
- *Rust Foundation*, rust-foundation.org.
- CISA/NSA, *The Case for Memory Safe Roadmaps*, 2023.
- Andrew Kelley et al., *The Zig Programming Language*, ziglang.org.