La vie en Rust

Simon Chemouil
@simach

Toulouse Rust, 2019

Outline

Disclaimer

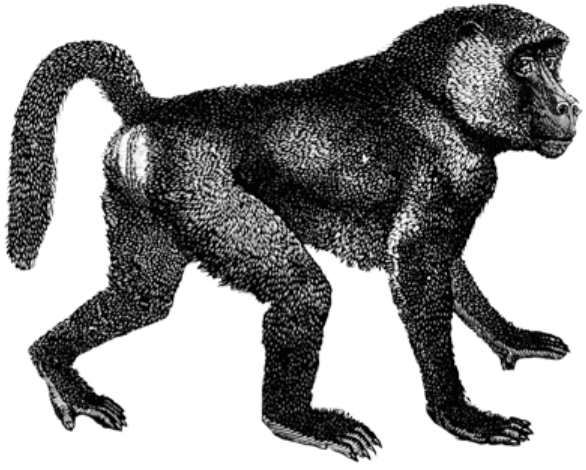

ATTENTION
CE FLIM N'EST PAS
UN FLIM SUR
LE CYCLIMSE
MERCI DE VOTRE
COMPREHENSION

Stereotypical Rust Programmer?

- C/C++: **control**, **precision (=> performance)**
- Ruby, Python, JavaScript: **ecosystem**, ***client-side Web***, fast prototyping
- Java, C#: **ecosystem**, ***tooling***, business support / entreprise friendliness
- Haskell, OCaml, F#: ***high-level constructs***, ***academically sound theory***, **safety & programming comfort**

To RiiR Or Not To RiiR...

Time for change.



Rewrite everything in Rust

10x Developer Guide

O RLY?

by Bootcamp Graduate

That Is The Question

Rust Genesis

- 2006: Graydon Hoare's personal project
- 2009: official support by Mozilla
- 2011: Rustc gets bootstrapped
  - Still an experimental language: optional GC, green threads, lower move semantics ergonomics...

Rust Genesis (Part II)

- 2014: Cargo announcement
  - Plan for crates.io
- May 2015: Rust 1.0
  - A commitment for stability
- August 2016: Zero-cost futures in Rust (blog)

Rust Is Not Ready

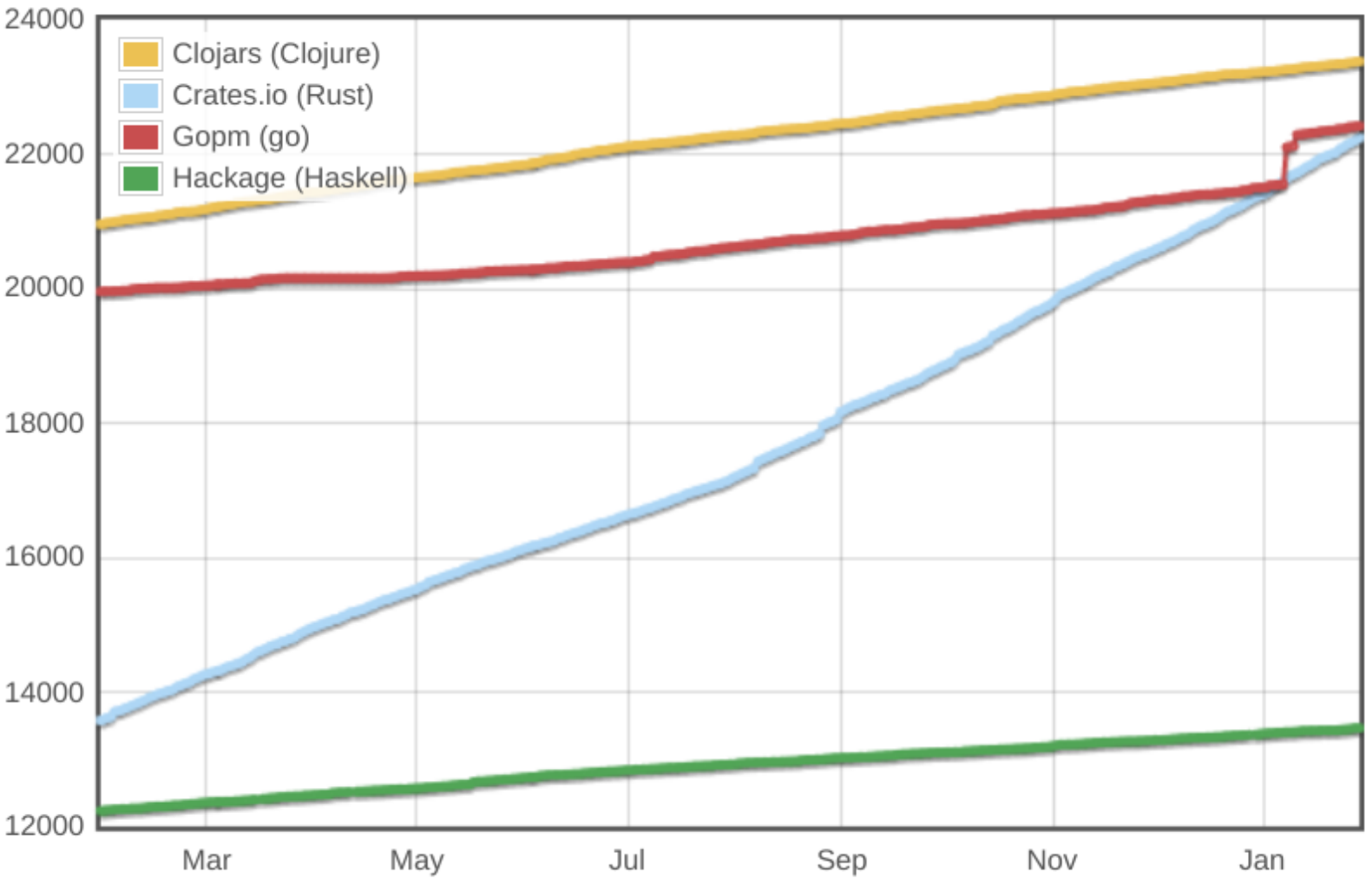
- Depends on your day job
  - Are We There Yet?
- My day job:
  - Data processing, async IO, common formats and protocols, ...

Time Skip

Revisiting Rust

- Summer 2018
  - Checking the state of Rust: language, ecosystem
  - Incredible development, a testament to Rust's productivity
- Edition 2018
  - Productivity improvements
- Interesting community dynamic

Rust dynamics



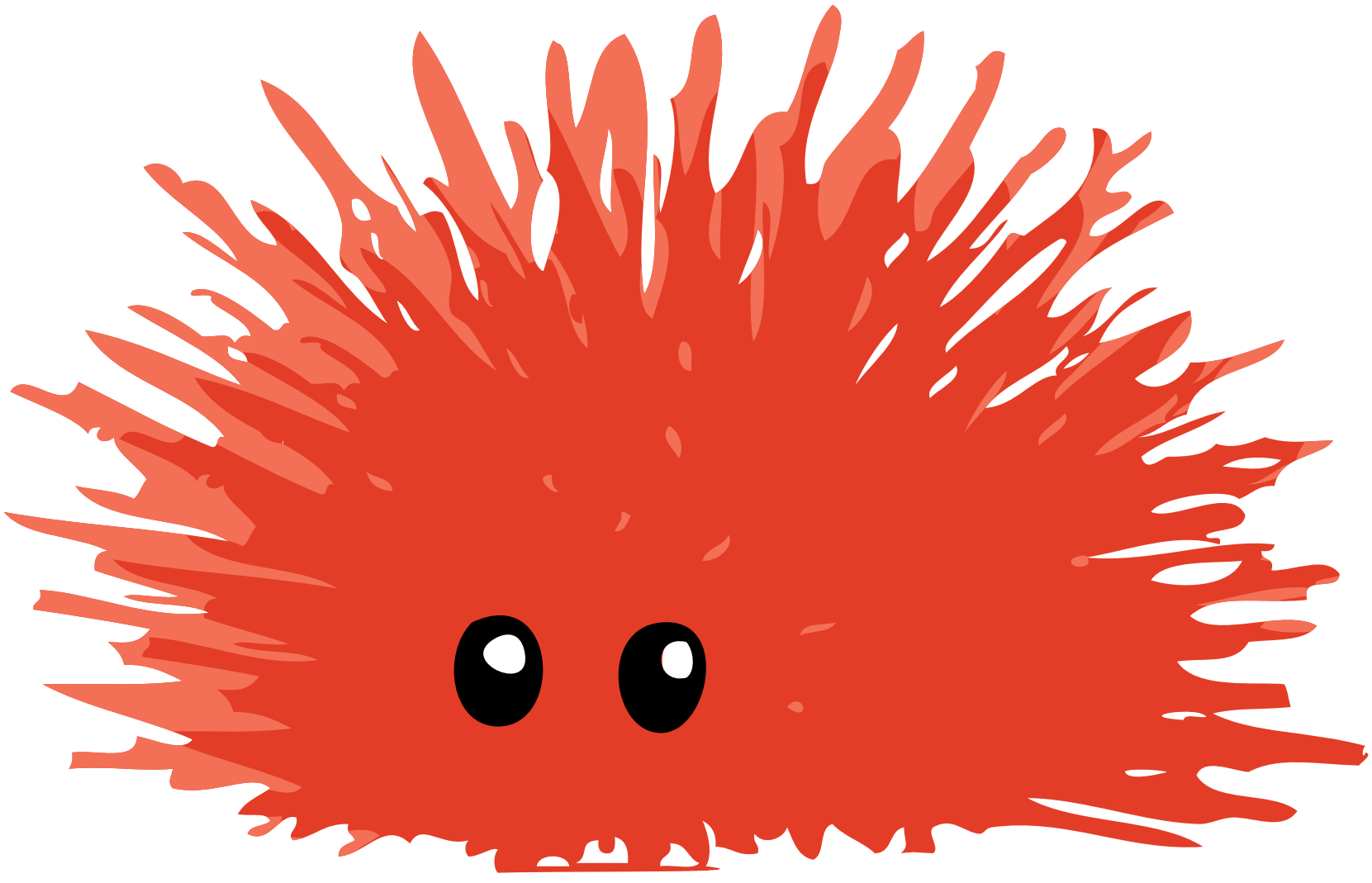
Beginning Rust

- The Rust Book!
- Finding a pet project
  - many Rust projects open to contributors

My First Rust Project

- A dynamic module runtime and service component framework inspired by OSGi
  - aka Pluginception
  - Still in progress
- Comfortable on day 2, covered a lot of ground in 5 days
  - Valuable help on IRC

The Rust Boogiemen



The Borrow Checker

- Ensures ownership and borrowing rules are respected
  - Applied *after* type checking
- Ever heard of "fighting the borrow checker"?
  - I'm the aggressor! :-)
  - How far can Rust go?
    - The Book is not enough, you *need* to fight or ask around.

Lifetimes!

- Give the borrow checker some hints!
  - Lifetimes are contagious – most of the time refer to the stack frame
- Not complicated:
  - but the rules are not clearly documented

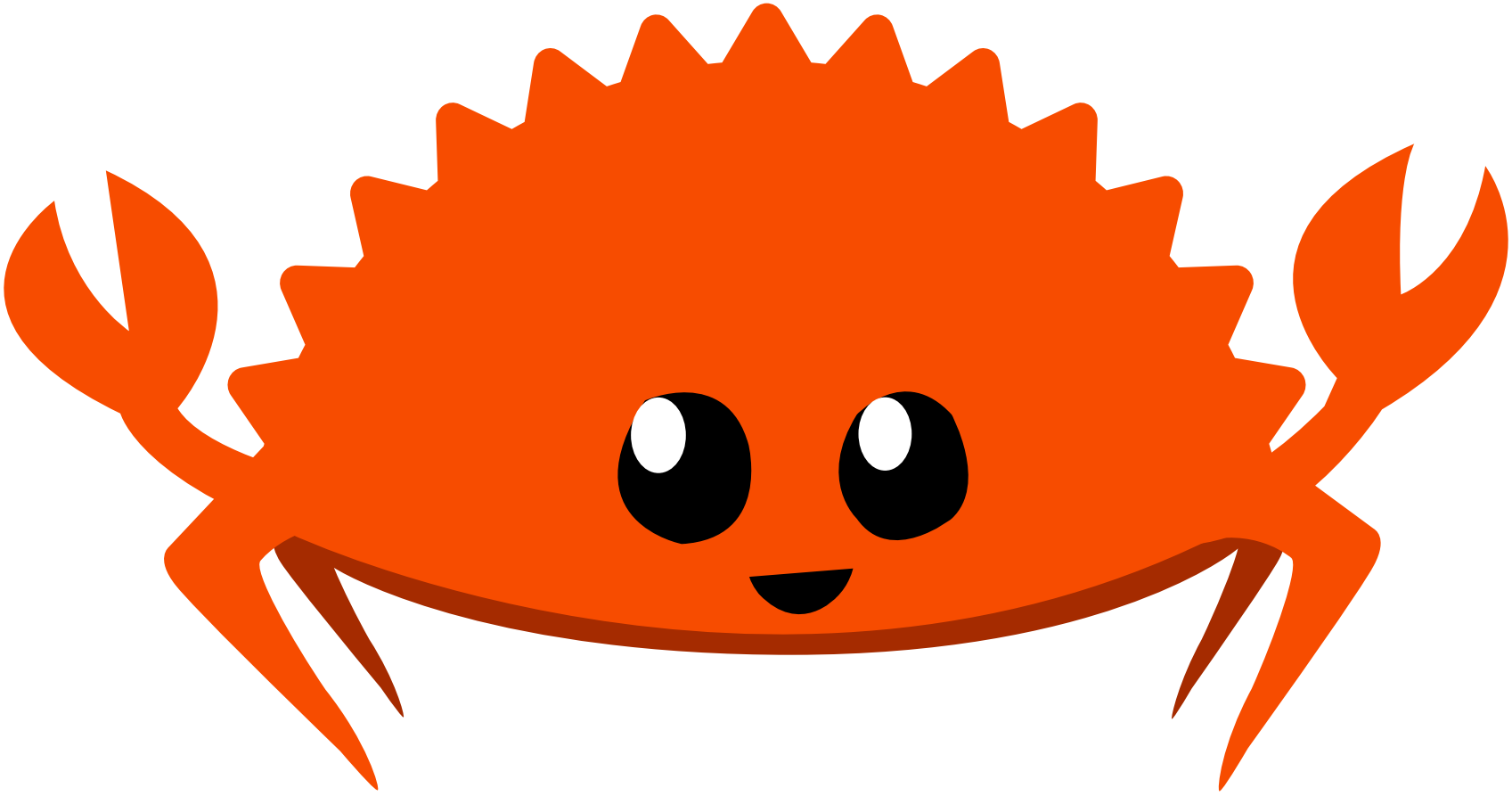
Static or Dynamic Borrowing, Lack of GC?

- Static references vs Rc and Arc
  - Or static vs dynamic lifetime!
  - Is the lifetime of my value known at compile-time?
- Life without GC is alright
  - Complex datastructures or algorithms need more thought
- These contraints result in high composability

Rust's Size

- Rust is a large language
  - Many subtleties
  - But after a few months you cover most of the ground
- You can be productive while knowing a fraction of Rust
  - Discover new parts as you need them

Strong Suits



No More Objects!

- Rust is not Object Oriented... and that's great!
  - Stronger design primitives instead: Struct and Enums that play well with generic types, and traits!
- Structs can have associated methods
  - It looks very much like Objects, but there is no subtyping in Rust (except for lifetimes)
- Abstraction through traits
  - Interfaces on (a lot of) steroids

Cargo, crates.io

- Project build, dependency management, etc is smooth
  - Very low learning curve
  - Heard of limitations, but haven't hit them
- Extensible through plugins
  - e.g cargo-watch for automatic refresh

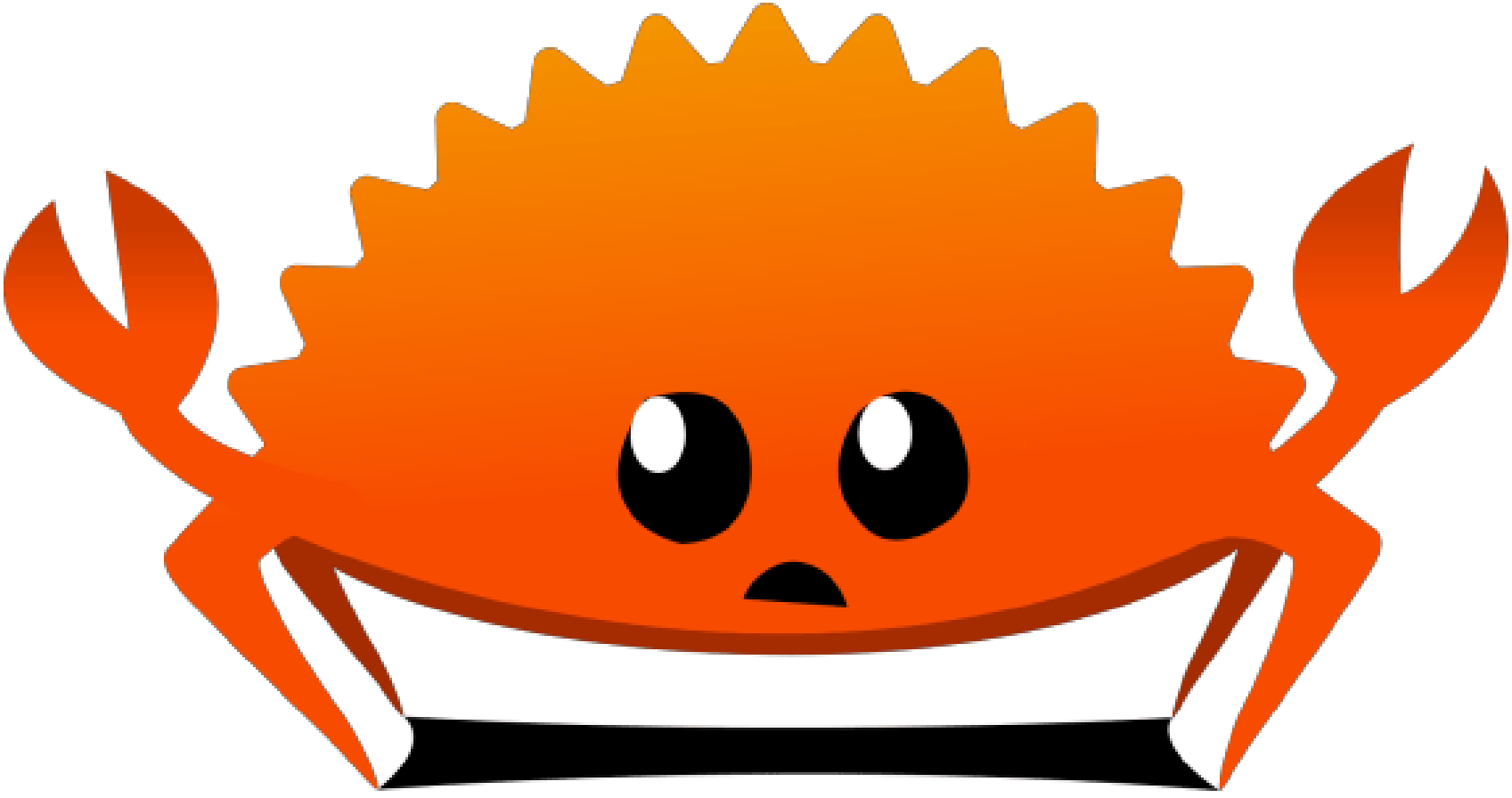
Thread-safety guarantees

- Rust statically prevents data races
  - When something is wrong, it shows you a full trace
- (Safe) Rust produces deterministic code
  - Deadlocks are easier to find, because they are reproducible

Community

- Very active, high-level community
  - Effective help on IRC
  - Follow and participate in Rust's development on GitHub
- Collaborative ecosystems (and WGs)
  - "Let's build X together"

Shortcomings



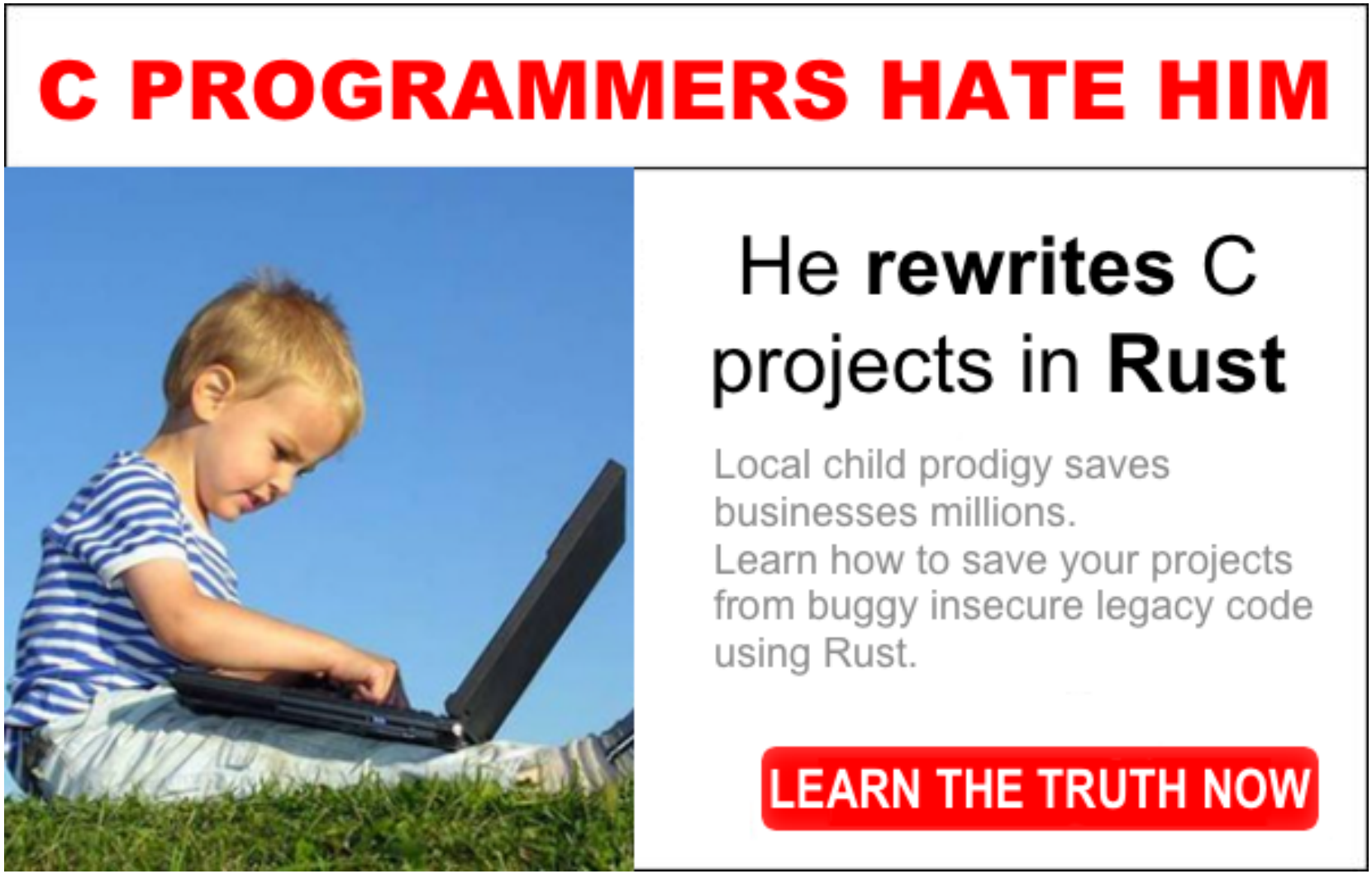
IDE and Debugging support

- Rust Language Server + VS Code
  - Both design issues and bugs
  - In progress: Rust Analyzer: a project for top-notch Rust IDE support
- Debugging support:
  - GDB/LLDB work, not perfect
  - A lot of "println" debugging + a strong compiler

An Unfinished Language

- Rust is unfinished
  - Many RFCs have been merged but not implemented
  - Documentation is scattered and not always complete
- But, very productive now, how much later? :-)
  - Productive as quality vs quantity

Let's Code


C PROGRAMMERS HATE HIM
He rewrites C projects in Rust
Local child prodigy saves businesses millions.
Learn how to save your projects from buggy insecure legacy code using Rust.
LEARN THE TRUTH NOW

Conclusion

- Much more to say:
  - Performance, type-level ergonomics, macros, ...
- Practice makes perfect
  - Find yourself a Rust project!

Questions...

Questions?

Twitter:

@simach