

# Why you should take a look at



*Antonin Carette - FOSDEM 2018 - Rust devroom*

Slides and resources available @ [github.com/k0pernicus/fosdem\\_rust\\_talk](https://github.com/k0pernicus/fosdem_rust_talk)

# Hello !

- Interested in system programming, optimizations and performances
- 2018's programming languages = *Rust* + *Nim* + *Elixir*
- Open Source contributor
- Mozilla ♥



*k0pernicus*



*k0pernicus*

At the beginning...  
a need

*Since 2000, for consumers,*

***big changes:***

- from 32bit to 64bit architectures,
- from mono-core to multi-core architectures,
- from mono-thread to multi-threaded applications,
- more powerful hardware,
- a lot of new softwares,
- etc...

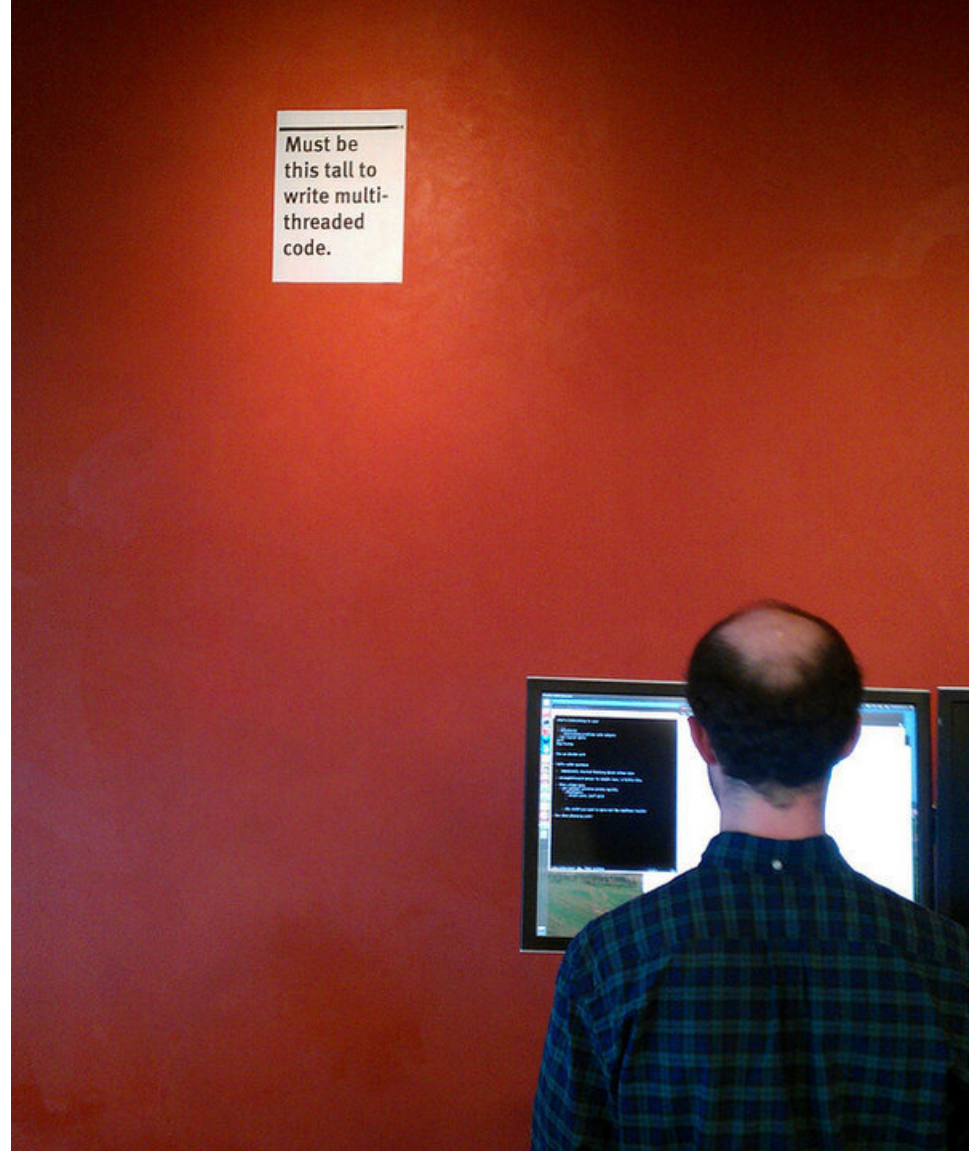
*Since 2000, for developers,*

***big troubles:***

- from sequential code to multi-threaded/multi-core support applications,
- data race issues,
- big memory leaks problems,
- big RAM consumption,
- the software race,
- etc...

# "Must be This Tall to Write Multi-Threaded Code"

*<http://bholley.net/blog/2015/must-be-this-tall-to-write-multi-threaded-code.html>*



We need a memory and threads safe programming language, with the same performance than C++.





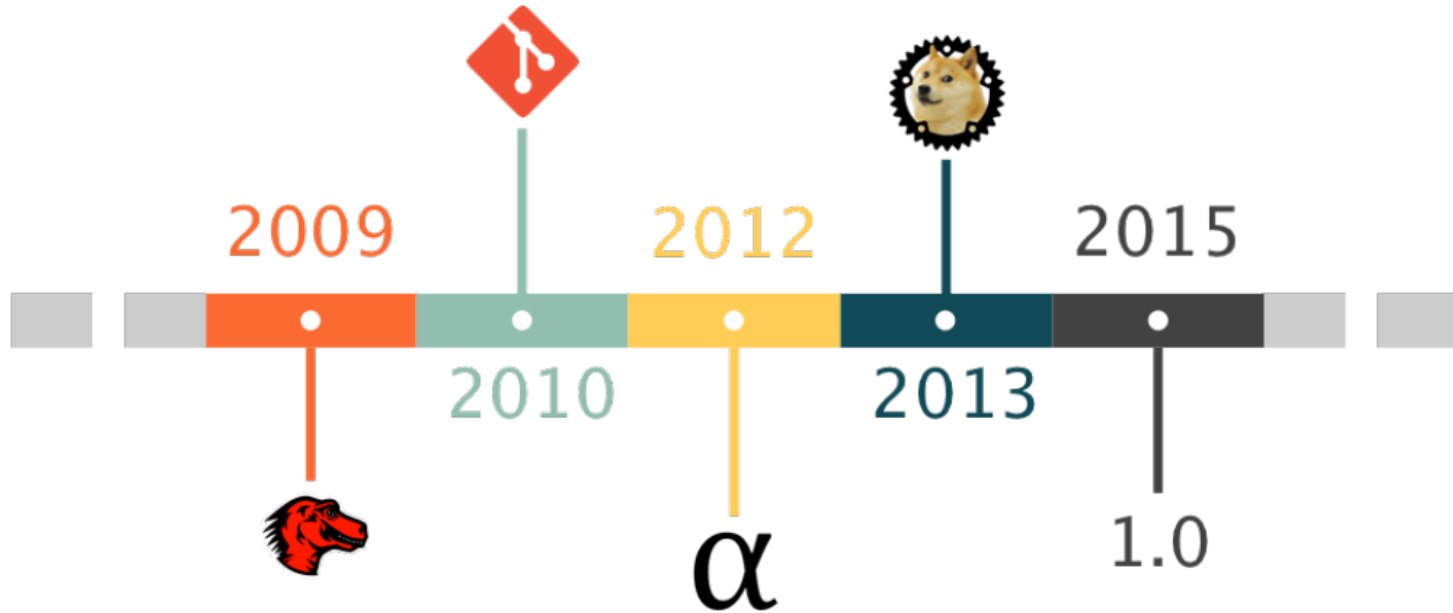
= **Rust**, a *modern, safe, fast,*  
and *concurrent* Open Source  
systems programming language.



# Content

1. Quick history
2. Uniqueness by concepts
3. What we want? Productivity!
4. Awesome companies, awesome projects
5. Open Source is **not only** code
6. #Rust2018
7. Conclusion

# Quick history



# Beyond the best features

- Immutability (default)
- **Memory leaks and data race safety, raised at compilation time**
- **Zero-cost abstraction**
- Define type behaviours with *traits*
- Rich build tool (*cargo*)
- Generics
- Multiple metaprogramming levels
- FFI (C, Ruby, Python, Haskell, etc.)
- WASM
- Rich error handling
- etc...

Once upon a time...  
the DVD seller,  
and the customer.

Vector author: macrovector (Freepik)

# Feature::MemSafety



I would like to  
buy this DVD!



# Feature::MemSafety



Sorry sir, but the  
box is empty...

# Feature::MemSafety

```
struct DVD{
    title: String,
}

fn take (dvd: DVD) {
    println!("Owner >> Thanks for the DVD!")
}

fn main () {
    // Null pointer
    let dvd : DVD;
    // COMPILE TIME ERROR <- use of possibly uninitialized variable: `dvd`
    take(dvd);
}
```

***No null pointer dereference situation***

# Feature::MemSafety



Sir, we have the  
DVD you requested!



# Feature::MemSafety



Thanks!



# Feature::MemSafety



This DVD is not  
mine anymore!

# Feature::MemSafety

```
struct DVD{
    title: String,
}

fn take (dvd: DVD) {
    println!("Owner >> I bought {} - it seems awesome!", dvd.title);
}

fn main () {
    let dvd = DVD{title: String::from("Blade Runner")};
    // `dvd` will belongs to `take`
    take(dvd);
    // `dvd` does not exists anymore, as `take` does not exists too, so I can't use it...
    // COMPILE TIME ERROR <- use of moved value: `dvd`
    println!("Me >> I still have {}!", dvd.title);
}
```

***Ownership situation***

# Feature::MemSafety



I would like to  
rent this DVD!



# Feature::MemSafety



Sure! Please **return to us**  
**this DVD before the end**  
**of the FOSDEM!**

# Feature::MemSafety

```
struct DVD{
    title: String,
}

fn borrow (dvd: &DVD) {
    // Access without modifications
    println!("Borrower >> {} is awesome!", dvd.title);
}

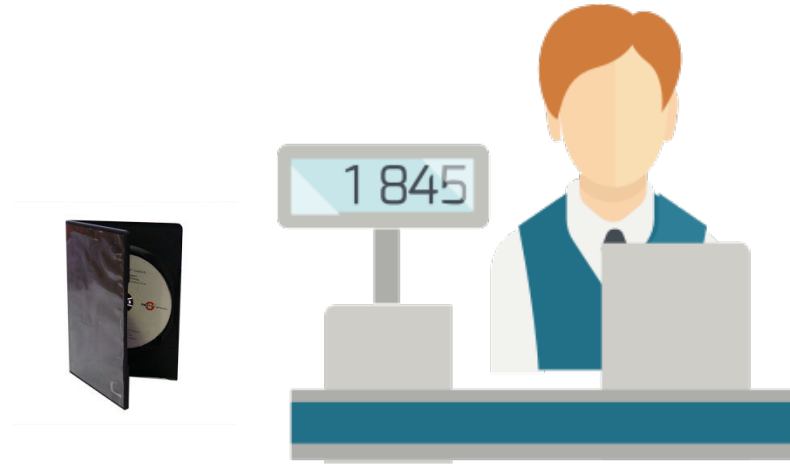
fn main () {
    let dvd = DVD{title: String::from("Blade Runner")};
    // `main` is still the owner of `dvd`
    borrow(&dvd);
    println!("Me >> I still have {}!", dvd.title);
}
```

***Borrowing situation***

# Feature::MemSafety



I couldn't read the  
DVD, due to the  
protection copy...



# Feature::MemSafety



Sorry for that. This is a  
DVD copy of the movie.



# Feature::MemSafety



Cool, a RW disk - let's  
try to modify it...



# Feature::MemSafety

```
struct DVD{
    title: String,
}

fn mut_borrow (dvd: &mut DVD) {
    dvd.title = String::from("Bienvenue chez les Ch'tis");
    println!("Borrower >> Nyark nyark!");
}

fn main () {
    let mut dvd = DVD{title: String::from("Blade Runner")};
    // `main` is still the owner of `dvd`
    mut_borrow(&mut dvd);
    println!("Me >> I still have... WHAT!? WHAT IS {}!?", dvd.title);
}
```

***Mutable Borrowing Situation***

# Feature::MemSafety

Using Rust, you can't:

- attempt to dereference a null pointer,
- attempt to use already-freed memory (ex. *dangling* pointer),
- forget to free memory,
- and attempt to free already-freed memory.

# Feature::MemSafety

But there is rules to respect:

1. the borrower's scope **must not outlast** the owner,
2. you can have **at least** one reference to a resource,
3. you can have **one** mutable reference to a resource,
4. you can't have the last two rules **at the same time**.

# Feature::ThreadSafety

When does a data race happens?

- at least two pointers to the same ressource,
- at least one writing pointer,
- un-synchronized operations.

# Feature::ThreadSafety

How can Rust answers to this problem ?

**Ownership** (again) because...

- if you have multiple references, you don't have any write pointer,
- if you have one pointer, you don't have any other references,
- synchronized operations by default.

# Feature::ThreadSafety

Using Rust, you can't:

- read and write the same variable from multiple threads at the same time (without wrapping it in a lock or other concurrency primitive),
- forget to acquire a lock before accessing the variable it protects.

# Feature::ZeroCostAbst

**Objective:** to combine low-level control with high-level programming concepts.



# Feature::ZeroCostAbst

*Developers:* "Features are good, abstraction is great, and we need security - but we care about overhead..."

*Rust maintainers:* "With Rust, you only pay for the features you actually use! Rust does not contains a GC, and performs safety checks at compile time!"

# Be productive



*Cargo*

+



*Rustup*

# Be productive



**clementd**



@clementd

Abonné



rustup + cargo is by far my fav toolchain  
when it comes to build + dep management

🌐 À l'origine en anglais

13:53 - 24 janv. 2018

Clément Delafargue, *Clever Cloud* CTO

# Cargo

Awesome features in one command, ... ..one configuration file !

- ***compile*** the program,
- ***check*** the program,
- ***build*** the doc,
- ***init*** the project,
- ***run*** the program,
- run ***unit tests***,
- run ***benchmarks***,
- ***publish*** your crate,
- ***install/uninstall*** crate(s),
- etc...

```
# The release profile, used for `cargo build --release`.
[profile.release]
opt-level = 3
debug = false
rpath = false
lto = false
debug-assertions = false
codegen-units = 1
panic = 'unwind'

# The testing profile, used for `cargo test`.
[profile.test]
opt-level = 0
debug = true
rpath = false
lto = false
debug-assertions = true
codegen-units = 1
panic = 'unwind'

# The benchmarking profile, used for `cargo bench`.
[profile.bench]
opt-level = 3
debug = false
rpath = false
lto = false
debug-assertions = false
codegen-units = 1
panic = 'unwind'
```

# Rustup

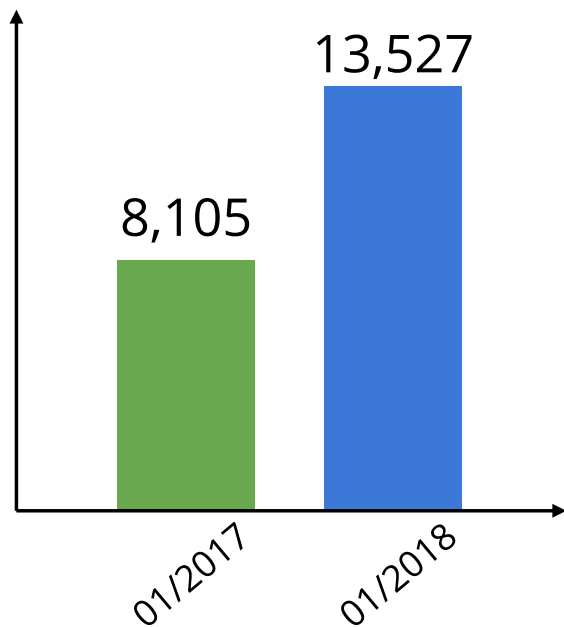
## Objectives:

- installs Rust from the official release channels,
- enabling you to easily switch between stable, beta, and nightly compilers,
- keep the compilers updated,
- making cross-compiling simpler.

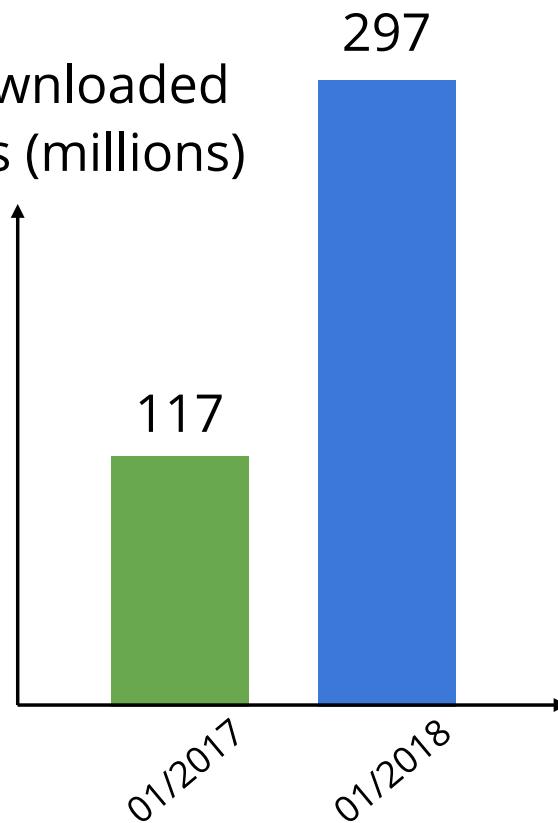
*<https://rustup.rs/>*

# Be productive

Crates on stock



# downloaded crates (millions)



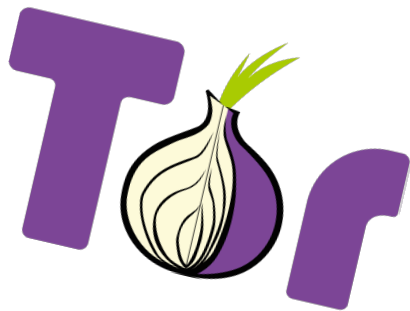
# Be productive

IDE's friendly: *<https://github.com/rust-lang-nursery/rls>*

# Rust in production



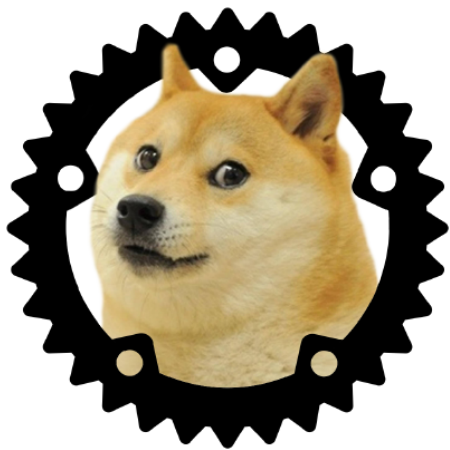
OVH.com



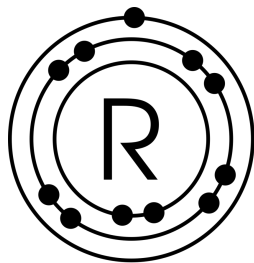
<https://www.rust-lang.org/en-US/friends.html>



# Rust in production



piston



Redox

Xi



DIESEL

*<https://github.com/rust-unofficial/awesome-rust>*



# Community 🥰

The Rust compiler, for 50 releases...

- 4,700 forks,
- 74,000 commits,
- 2,000 contributors.

The community is open to RFCs here: <http://rust-lang.github.io/rfcs>

More than 90 Rust User Groups worldwide, in over 35 countries.

Big events in US/Canada (*Rust Belt Rust*), Europe (*Rust Fest*), etc...

# Community 🥰

Search a meetup/conference or help here: [\*https://community.rs/\*](https://community.rs/)

What's everyone working on this week:  
[\*https://users.rust-lang.org/c/community\*](https://users.rust-lang.org/c/community)

Search/find whatever you want about community here:  
[\*https://www.rust-lang.org/en-US/community.html\*](https://www.rust-lang.org/en-US/community.html)

# Community

Developer Survey 2015  
stackoverflow.com



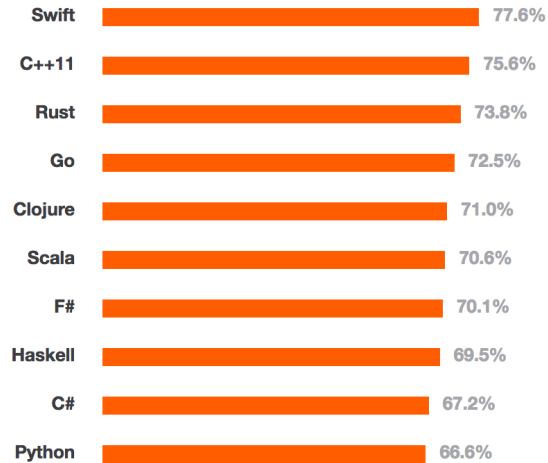
3rd position

## II. MOST LOVED, DREADED, AND WANTED

Most Loved

Most Dreaded

Most Wanted



# Community

Developer Survey 2016

stackoverflow.com



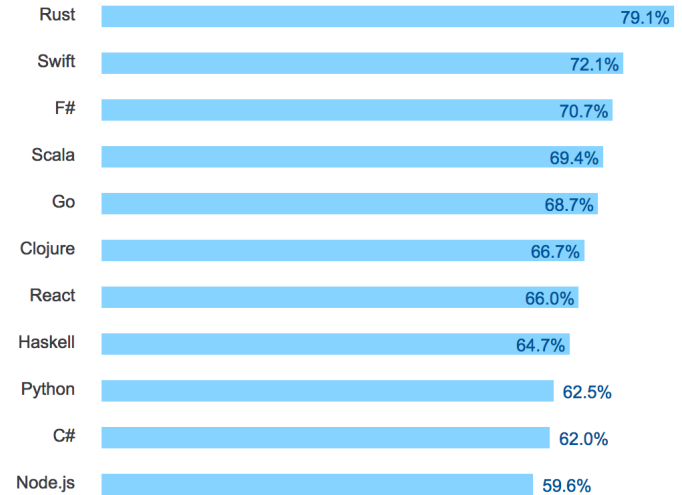
1st position

## II. Most Loved, Dreaded, and Wanted

Loved

Dreaded

Wanted



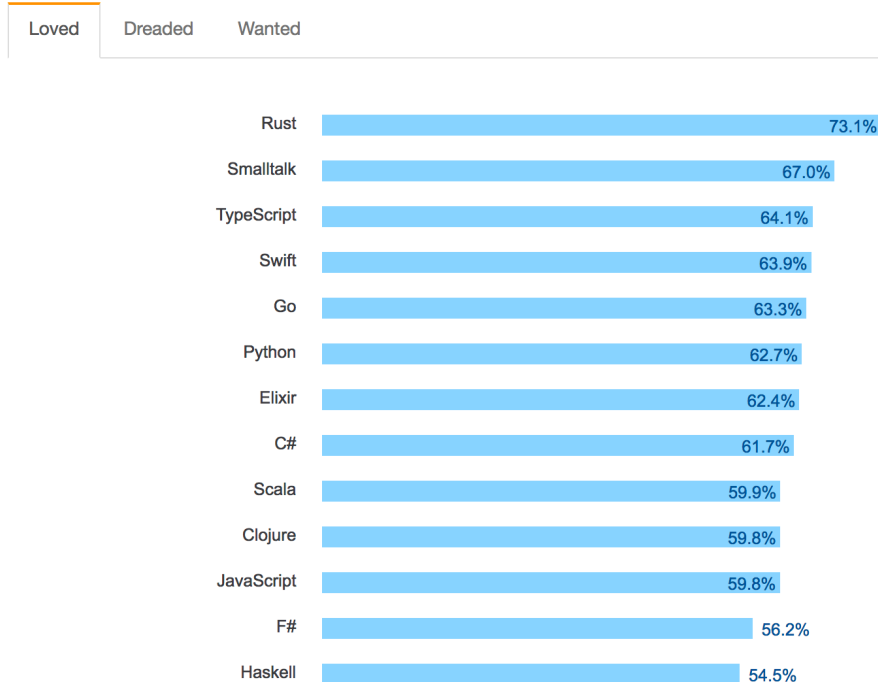
# Community

Developer Survey 2017  
stackoverflow.com



1st position

## Most Loved, Dreaded, and Wanted Languages







# #Rust2018


*"We care about your requests."*

## RFC: Rust 2018 Roadmap #2314

 **Open** aturon wants to merge 9 commits into [rust-lang:master](#) from [aturon:roadmap-2018](#)

 Conversation 50

 Commits 9

 Files changed 1



aturon commented a day ago • edited ▾

Contributor



This RFC sets the *Rust 2018 Roadmap*, in accordance with [RFC 1728](#). This year's goals are:

- Ship an epoch release: Rust 2018.
- Build resources for intermediate Rustaceans.
- Connect and empower Rust's global community.
- Grow Rust's teams and new leaders within them.

In pursuing these goals, we will focus particularly on four target domains for Rust:

- Web services.
- WebAssembly.
- CLI apps.
- Embedded devices.

*A hearty thank you to the 100-some people who wrote blog posts to help drive this process!*

[Rendered](#)



66



3



53



50

<https://github.com/rust-lang/rfcs/pull/2314>



So, interested ?



# So, interested ?

**Survival guide:**    [github.com/k0pernicus/fosdem\\_rust\\_talk](https://github.com/k0pernicus/fosdem_rust_talk)

**Rust official:**    <https://rust-lang.org>

**Rust book:**    <https://doc.rust-lang.org/book>