Five Years on the Rust Core Team

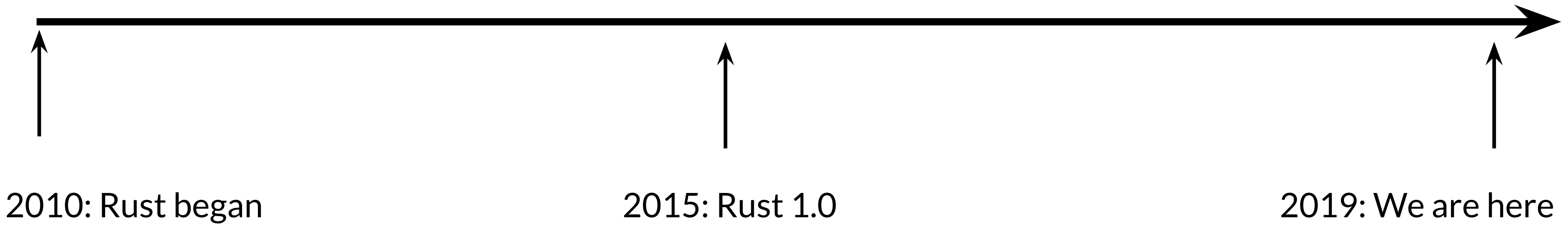
A retrospective

@steveklabnik

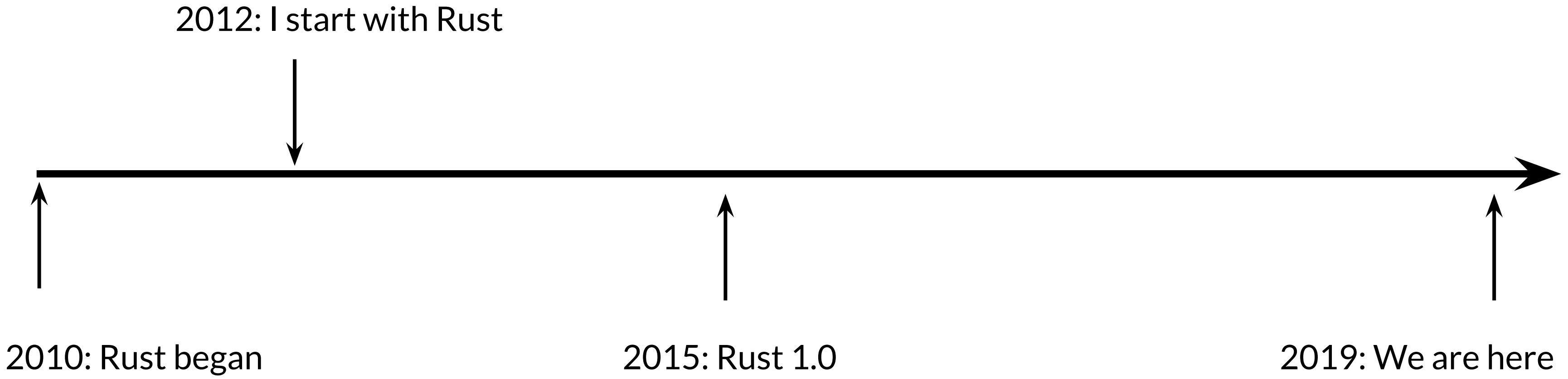
What is Rust?

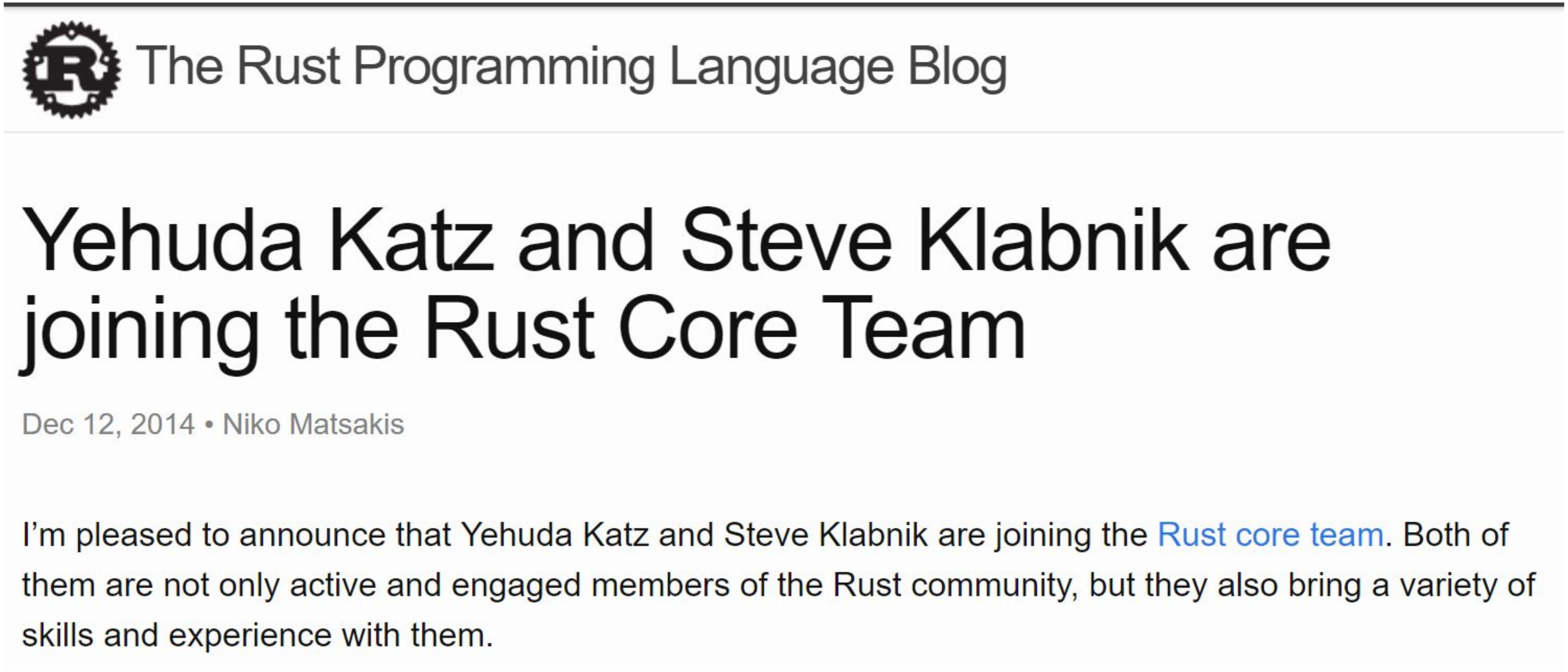
Rust

A language empowering everyone
to build reliable and efficient software.


2010: Rust began
2015: Rust 1.0
2019: We are here

I wrote my first Rust code in December 2012


2012: I start with Rust
2010: Rust began
2015: Rust 1.0
2019: We are here

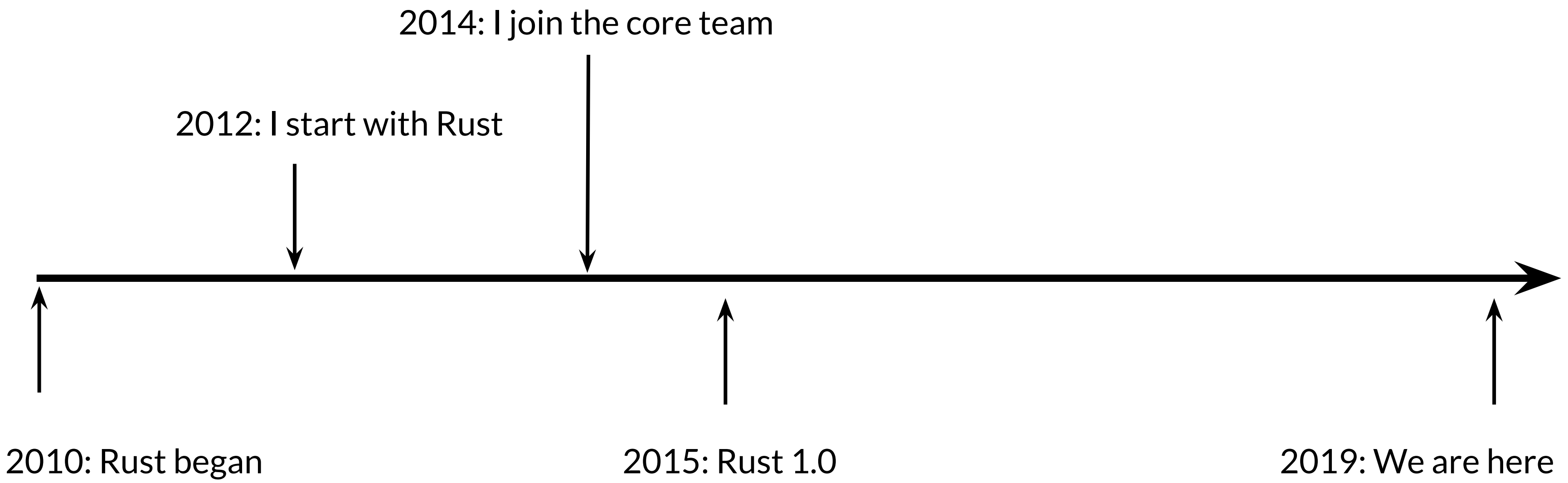


The Rust Programming Language Blog

Yehuda Katz and Steve Klabnik are joining the Rust Core Team

Dec 12, 2014 • Niko Matsakis

I’m pleased to announce that Yehuda Katz and Steve Klabnik are joining the Rust core team. Both of them are not only active and engaged members of the Rust community, but they also bring a variety of skills and experience with them.


2014: I join the core team
2012: I start with Rust
2010: Rust began
2015: Rust 1.0
2019: We are here

But this talk isn’t really about me; it’s about Rust

It’s about Rust and its **governance**

Governance: who makes decisions?

In the beginning:

informal

Rust governance formations

Over time

- 2010: informal

As projects grow, “who makes decisions” is more important

Problem:
Lack of
consistency

Two solutions

BDFL

Core Team

BDFL

“Benevolent Dictator For Life”

Pros:

- Consistent: only one person!
- Simple: only one person!

Cons:

- What happens if the BDFL quits?
- What happens if they’re not benevolent?

Core Team

Pros:

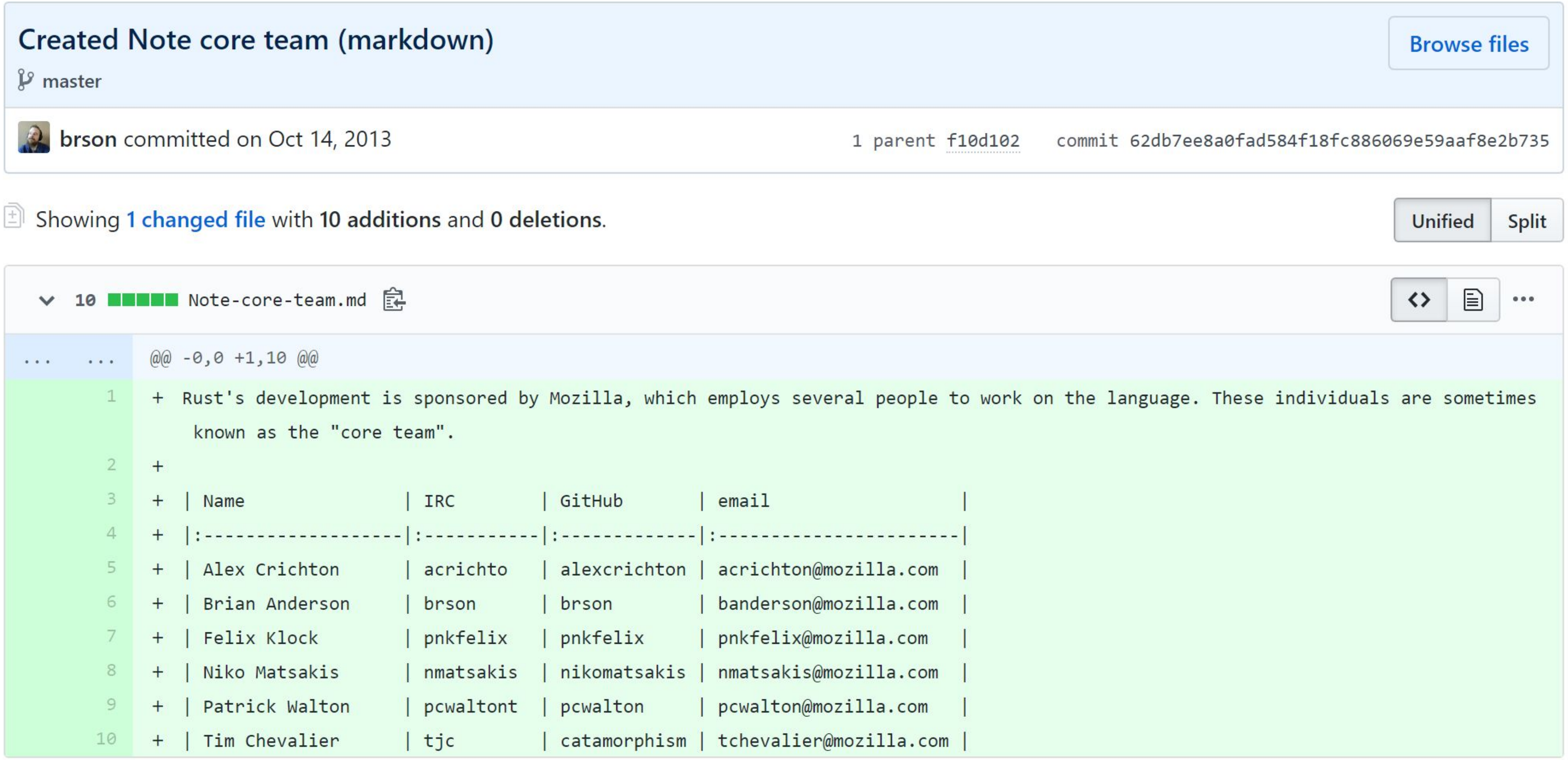
- Consistent: group agrees
- Resilient: people can be added or removed

Cons:

- Not simple: now multiple people have to agree
- Not easy: what if we disagree on something important?

Two solutions

- 🚫 ~~BDFL~~
- 🥇 Core Team



Created Note core team (markdown)

master

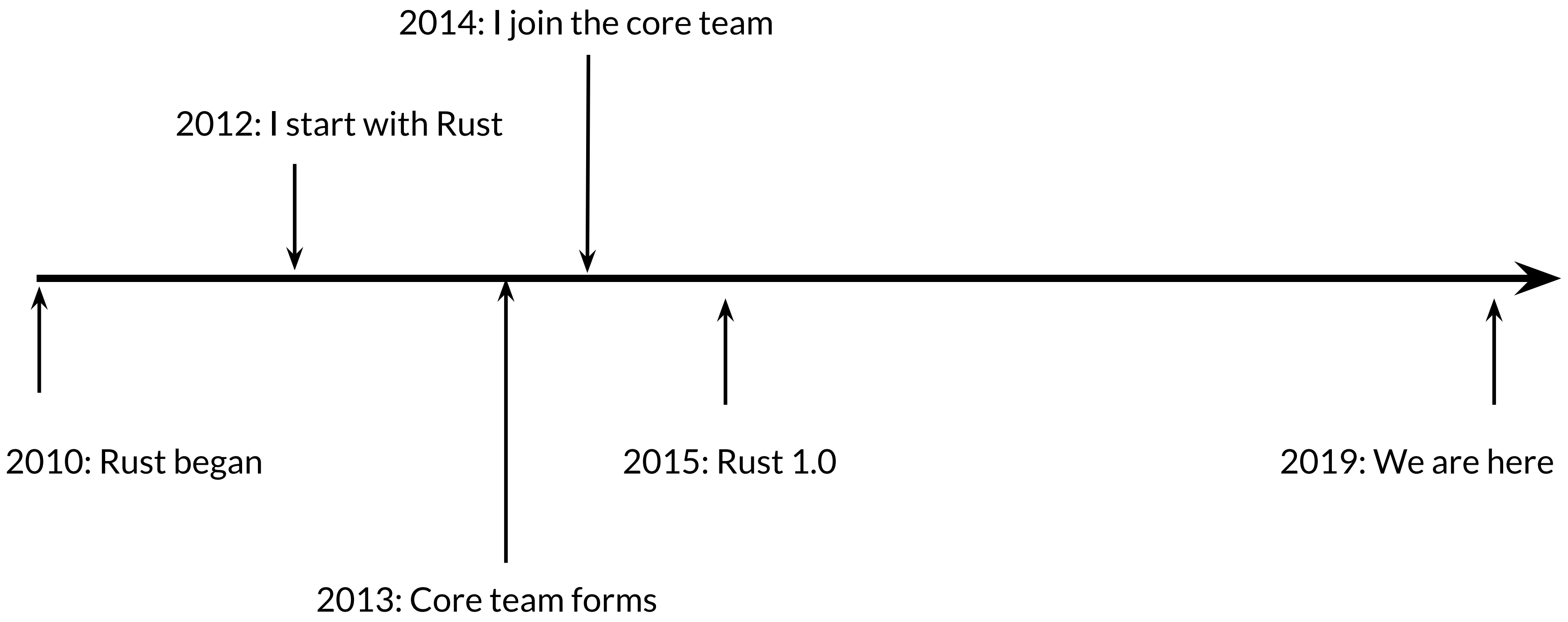
brson committed on Oct 14, 2013

1 parent f10d102 commit 62db7ee8a0fad584f18fc886069e59aaf8e2b735

Showing 1 changed file with 10 additions and 0 deletions.

10 Note-core-team.md

```
@@ -0,0 +1,10 @@
+ Rust's development is sponsored by Mozilla, which employs several people to work on the language. These individuals are sometimes known as the "core team".
+
+ | Name              | IRC        | GitHub       | email                 |
+ |:------------------|:-----------|:-------------|:----------------------|
+ | Alex Crichton     | acrichto   | alexcrichton | acrichton@mozilla.com |
+ | Brian Anderson    | brson      | brson        | banderson@mozilla.com |
+ | Felix Klock       | pnkfelix   | pnkfelix     | pnkfelix@mozilla.com  |
+ | Niko Matsakis     | nmatsakis  | nikomatsakis | nmatsakis@mozilla.com |
+ | Patrick Walton    | pcwaltont  | pcwalton     | pcwalton@mozilla.com  |
+ | Tim Chevalier     | tjc        | catamorphism | tchevalier@mozilla.com |
```


2014: I join the core team
2012: I start with Rust
2010: Rust began
2013: Core team forms
2015: Rust 1.0
2019: We are here

Rust governance formations

Over time

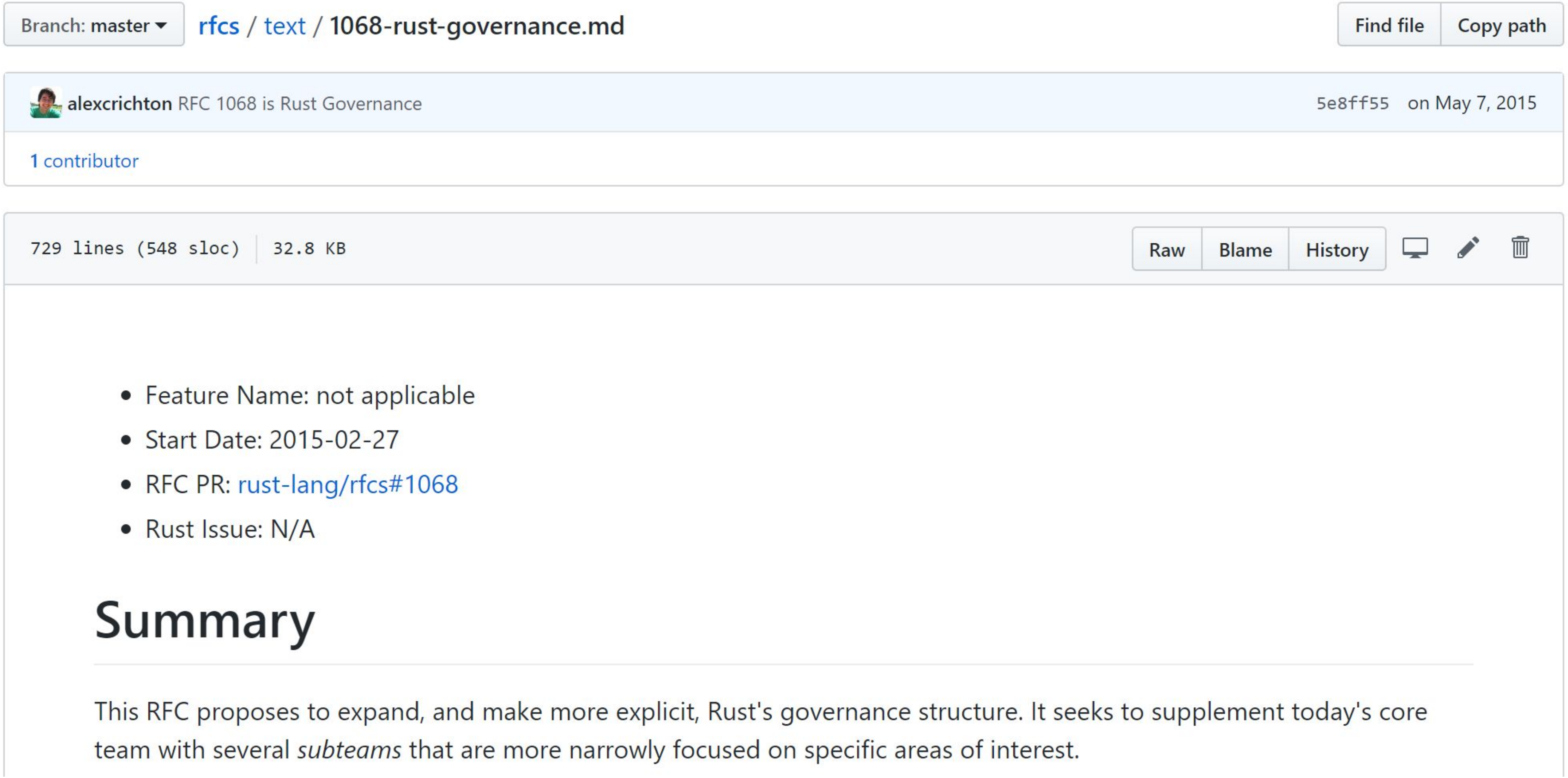
- 2010: informal
- 2013: Core Team

Problem:

This doesn’t scale

Solution:

More teams!



Branch: master rfcs / text / 1068-rust-governance.md Find file Copy path

alexcrichton RFC 1068 is Rust Governance 5e8ff55 on May 7, 2015

1 contributor

729 lines (548 sloc) 32.8 KB Raw Blame History

- Feature Name: not applicable
- Start Date: 2015-02-27
- RFC PR: rust-lang/rfcs#1068
- Rust Issue: N/A

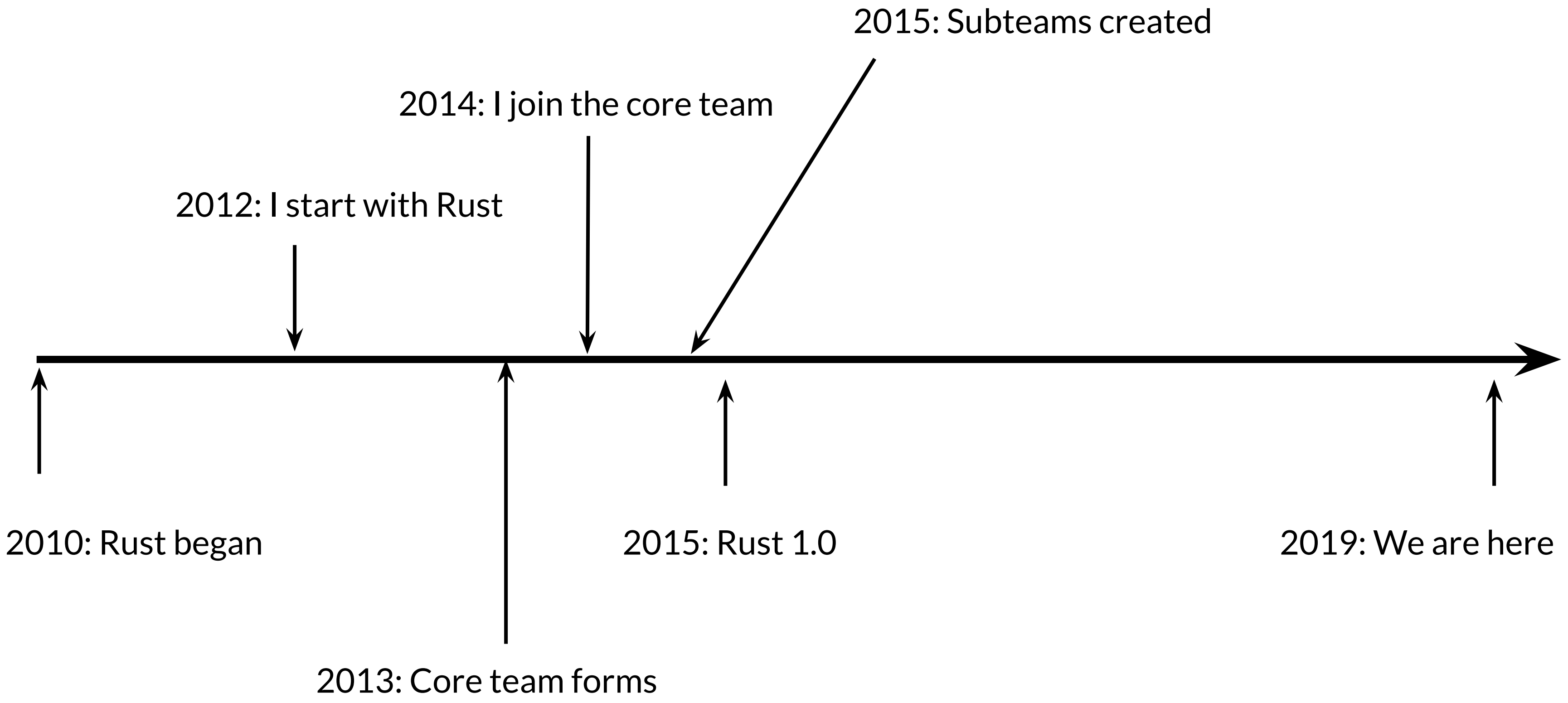
Summary

This RFC proposes to expand, and make more explicit, Rust's governance structure. It seeks to supplement today's core team with several *subteams* that are more narrowly focused on specific areas of interest.

The teams

With all of that out of the way, what subteams should we start with? This RFC proposes the following initial set:

- Language design
- Libraries
- Compiler
- Tooling and infrastructure
- Moderation


2015: Subteams created
2014: I join the core team
2012: I start with Rust
2010: Rust began
2015: Rust 1.0
2019: We are here
2013: Core team forms

Rust governance formations

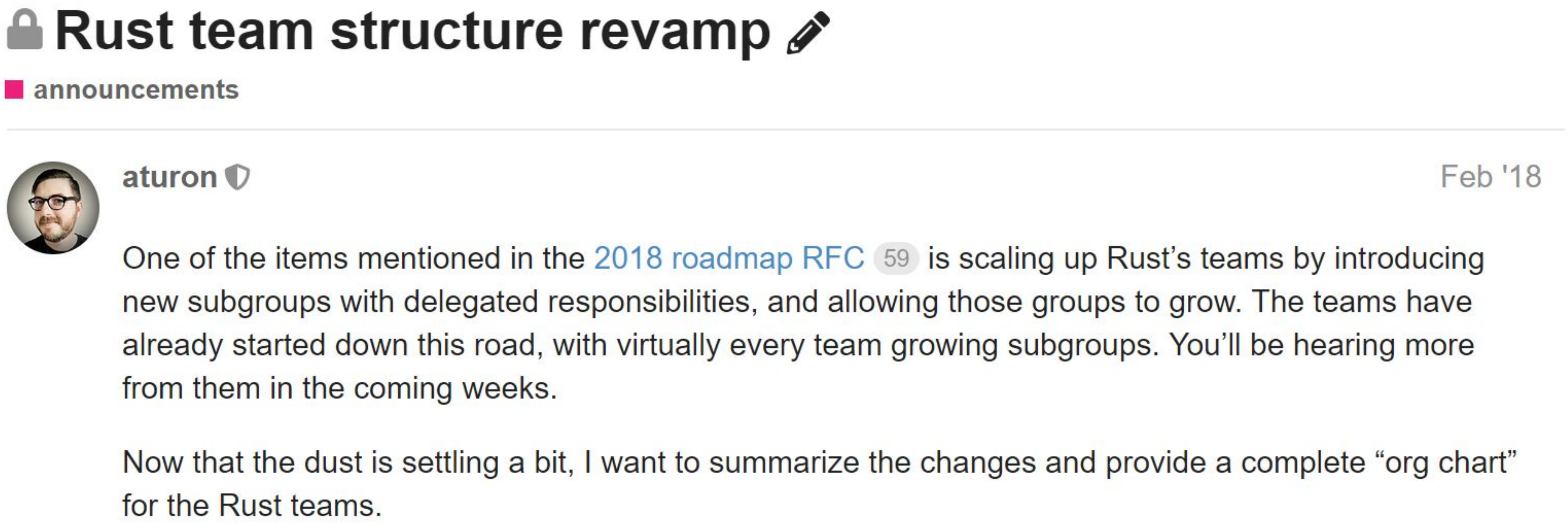
Over time

- 2010: informal
- 2013: Core Team
- 2015: Core team + subteams

Problem:
Inflexible, and not every team wants RFCs

Solution:

Even more teams + Working Groups



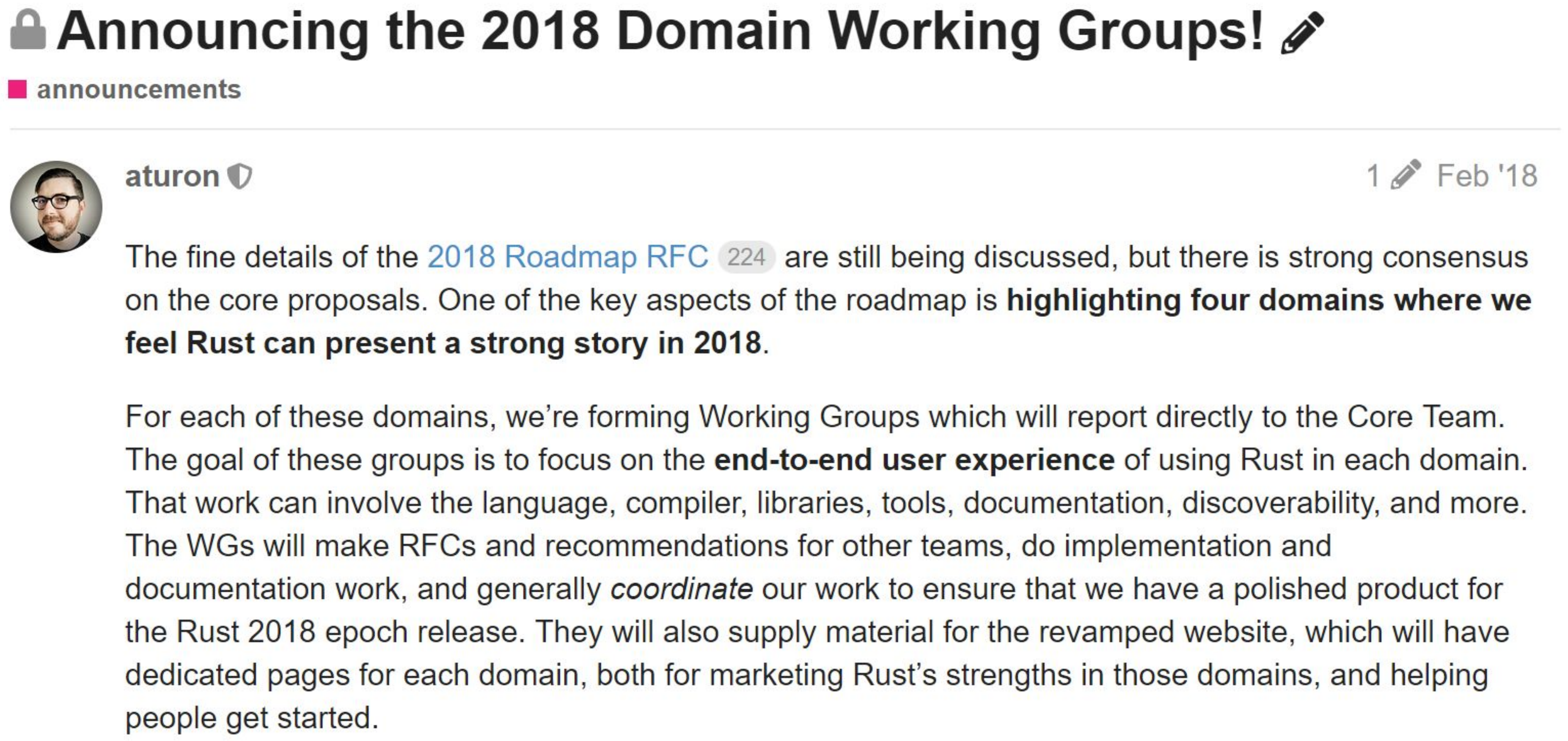
Rust team structure revamp

announcements

aturon Feb '18

One of the items mentioned in the 2018 roadmap RFC 59 is scaling up Rust's teams by introducing new subgroups with delegated responsibilities, and allowing those groups to grow. The teams have already started down this road, with virtually every team growing subgroups. You'll be hearing more from them in the coming weeks.

Now that the dust is settling a bit, I want to summarize the changes and provide a complete "org chart" for the Rust teams.



Announcing the 2018 Domain Working Groups!

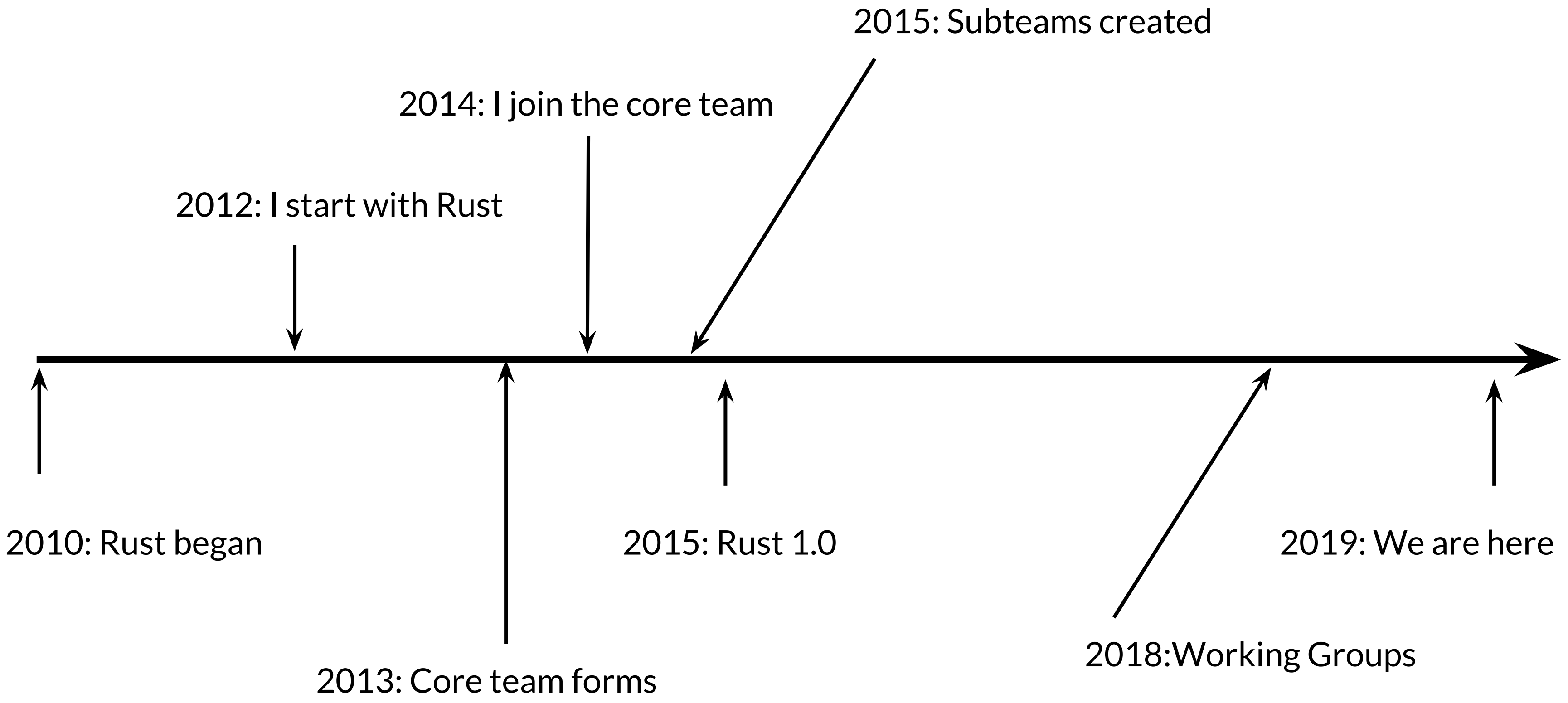
announcements

aturon

1 Feb '18

The fine details of the 2018 Roadmap RFC 224 are still being discussed, but there is strong consensus on the core proposals. One of the key aspects of the roadmap is **highlighting four domains where we feel Rust can present a strong story in 2018**.

For each of these domains, we're forming Working Groups which will report directly to the Core Team. The goal of these groups is to focus on the **end-to-end user experience** of using Rust in each domain. That work can involve the language, compiler, libraries, tools, documentation, discoverability, and more. The WGs will make RFCs and recommendations for other teams, do implementation and documentation work, and generally *coordinate* our work to ensure that we have a polished product for the Rust 2018 epoch release. They will also supply material for the revamped website, which will have dedicated pages for each domain, both for marketing Rust's strengths in those domains, and helping people get started.


2015: Subteams created
2014: I join the core team
2012: I start with Rust
2010: Rust began
2013: Core team forms
2015: Rust 1.0
2018:Working Groups
2019: We are here

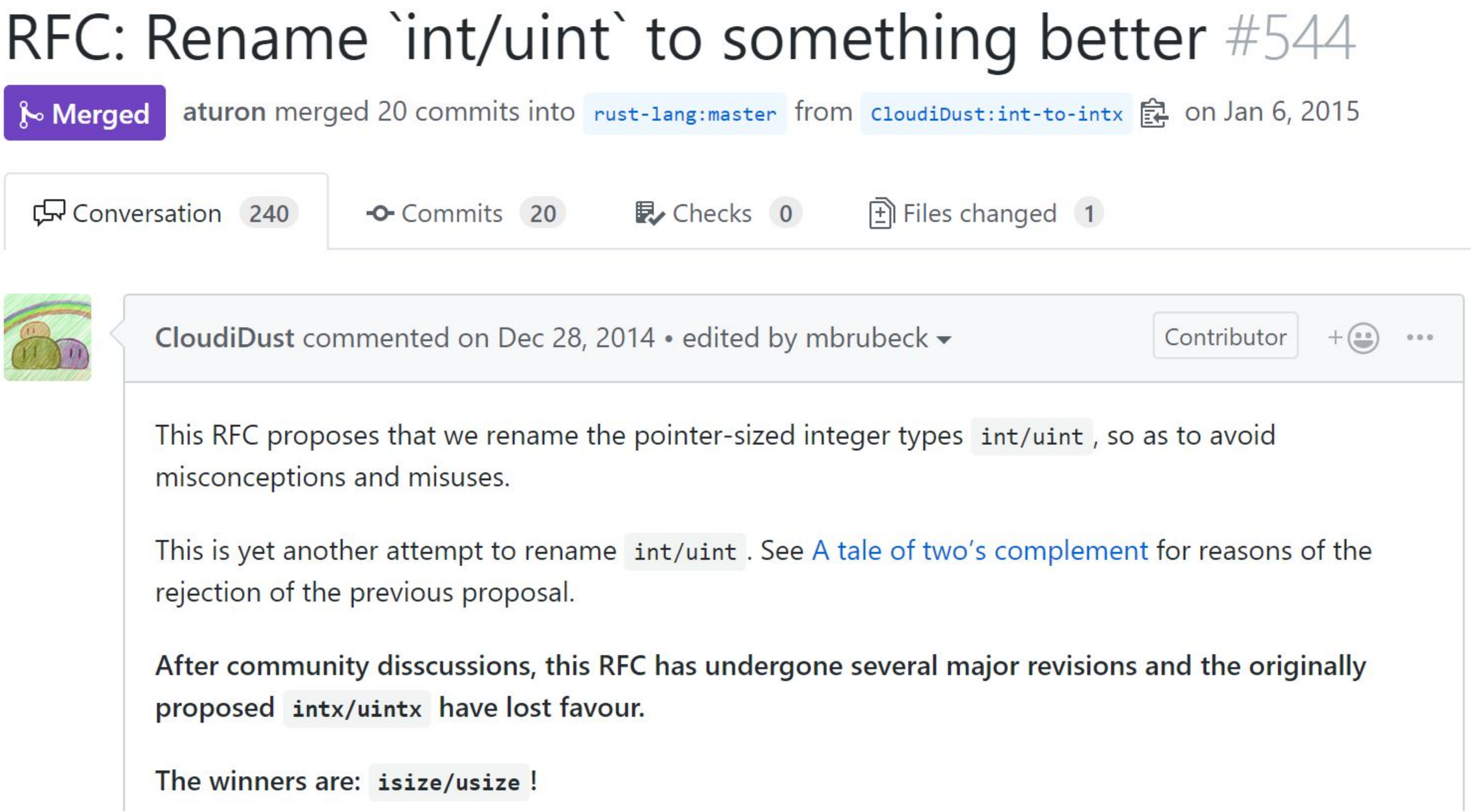
Rust governance formations

Over time

- 2010: informal
- 2013: Core Team
- 2015: Core team + subteams
- 2018: Core team + subteams + working groups

Σ(￣。￣ノ)

We’ve also learned how to make better decisions



RFC: Rename `int/uint` to something better #544

Merged aturon merged 20 commits into `rust-lang:master` from `CloudiDust:int-to-intx` on Jan 6, 2015

CloudiDust commented on Dec 28, 2014 • edited by mbrubeck

This RFC proposes that we rename the pointer-sized integer types `int/uint`, so as to avoid misconceptions and misuses.

This is yet another attempt to rename `int/uint`. See A tale of two's complement for reasons of the rejection of the previous proposal.

After community disscussions, this RFC has undergone several major revisions and the originally proposed `intx/uintx` have lost favour.

The winners are: `isize/usize`!

The “no new rationale” rule

Making decisions in public

Decisions must be made only from the basis of rationale already debated in public.

A final proposal for await syntax

Posted on May 6, 2019

This is an announcement regarding the resolution of the syntax for the await operator in Rust. This is one of the last major unresolved questions blocking the stabilization of the async/await feature, a feature which will enable many more people to write non-blocking network services in Rust. This post contains information about the timeline for the final decision, a proposal from the language team which is the most likely syntax to be adopted, and the justification for this decision.

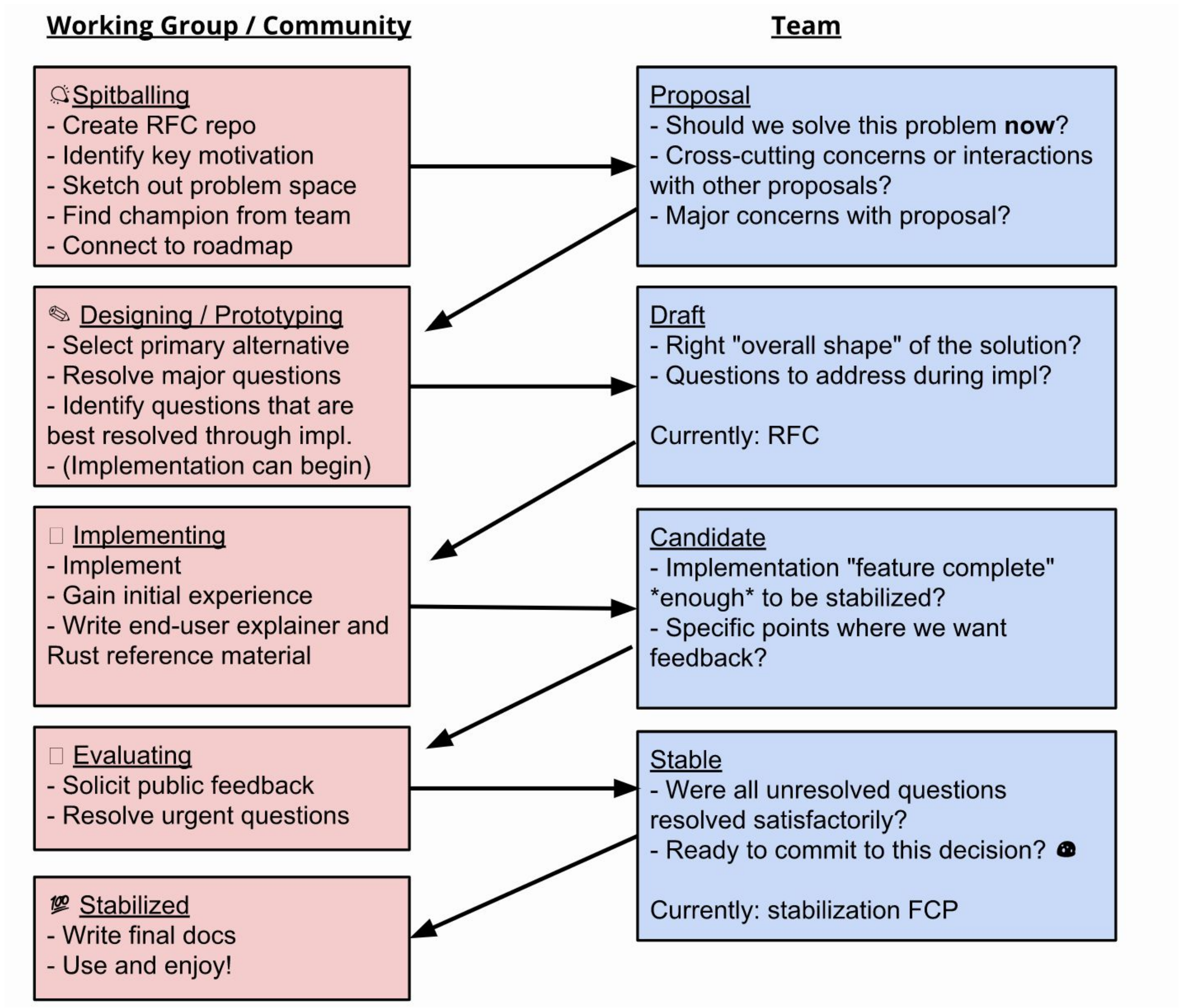
In brief: we intend to make a final decision on **May 23**, and we currently favor adopting the "dot await" postfix syntax. All of this is elaborated further in this document.

We still have much to improve

Are things too open?

Anyone can comment on any RFC at any time, but only one person has actually authored the RFC.

Responding to hundreds of comments is only possible if Rust is your job.



Working Group / Community

💭 Spitballing
- Create RFC repo
- Identify key motivation
- Sketch out problem space
- Find champion from team
- Connect to roadmap

✏️ Designing / Prototyping
- Select primary alternative
- Resolve major questions
- Identify questions that are best resolved through impl.
- (Implementation can begin)

Implementing
- Implement
- Gain initial experience
- Write end-user explainer and Rust reference material

Evaluating
- Solicit public feedback
- Resolve urgent questions

💯 Stabilized
- Write final docs
- Use and enjoy!

Team

Proposal
- Should we solve this problem **now**?
- Cross-cutting concerns or interactions with other proposals?
- Major concerns with proposal?

Draft
- Right "overall shape" of the solution?
- Questions to address during impl?

Currently: RFC

Candidate
- Implementation "feature complete" *enough* to be stabilized?
- Specific points where we want feedback?

Stable
- Were all unresolved questions resolved satisfactorily?
- Ready to commit to this decision? 😬

Currently: stabilization FCP

Do we need a “Rust Foundation”?

There’s no way to donate money to Rust development right now.

A foundation would let us do that. But it also could be a huge distraction.

Leadership in more places

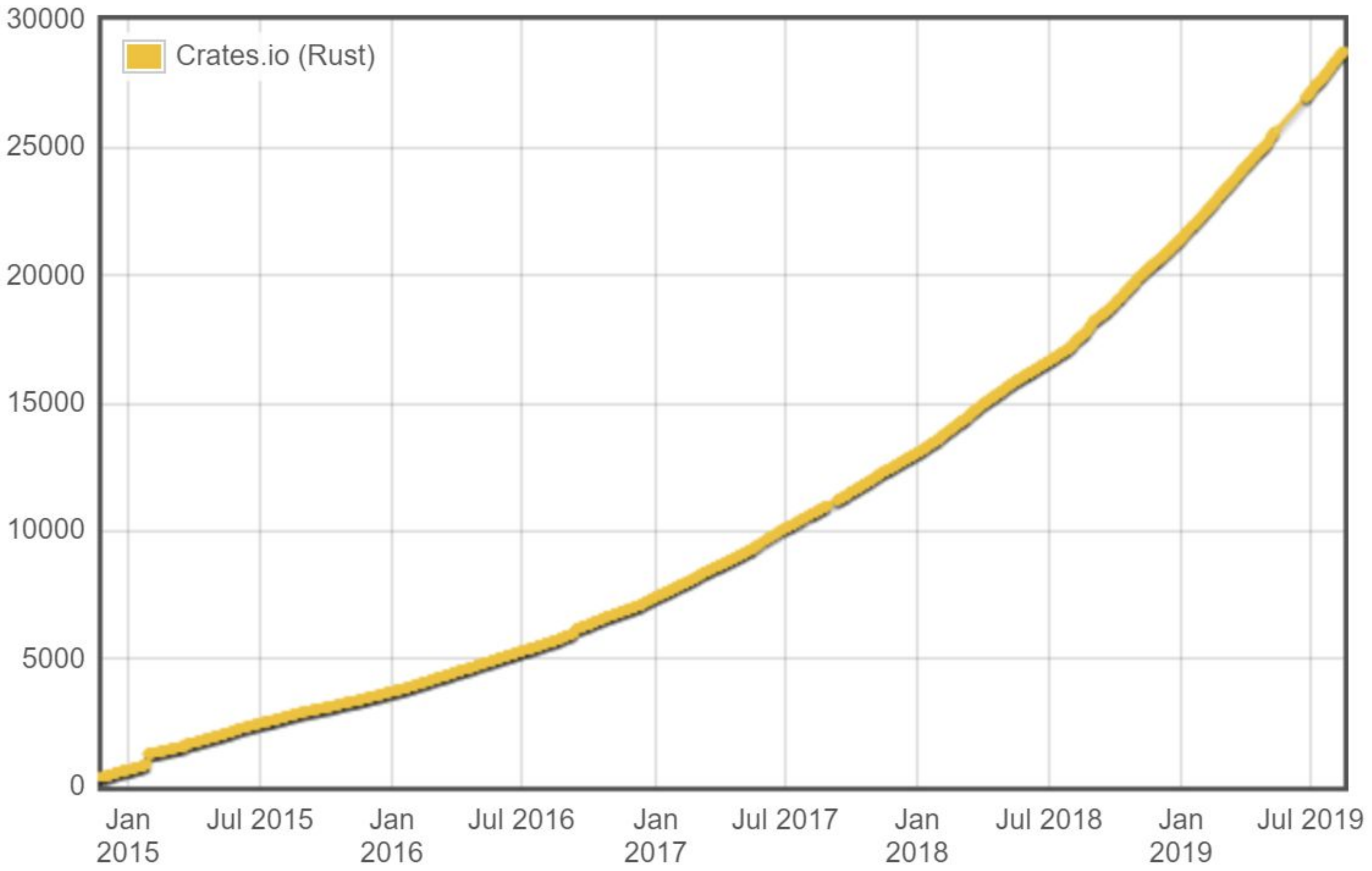
Leadership is mostly based in the US and Europe.

We'd like to get more people involved from more places.

Scheduling meetings becomes even harder.

Rust is succeeding!

Module Counts



New sponsors of Rust infrastructure

We'd like to thank two new sponsors of Rust's infrastructure who provided the resources needed to make Rust 1.37.0 happen: Amazon Web Services (AWS) and Microsoft Azure.

- AWS has provided hosting for release artifacts (compilers, libraries, tools, and source code), serving those artifacts to users through CloudFront, preventing regressions with Crater on EC2, and managing other Rust-related infrastructure hosted on AWS.
- Microsoft Azure has sponsored builders for Rust’s CI infrastructure, notably the extremely resource intensive rust-lang/rust repository.

Thank you! <3