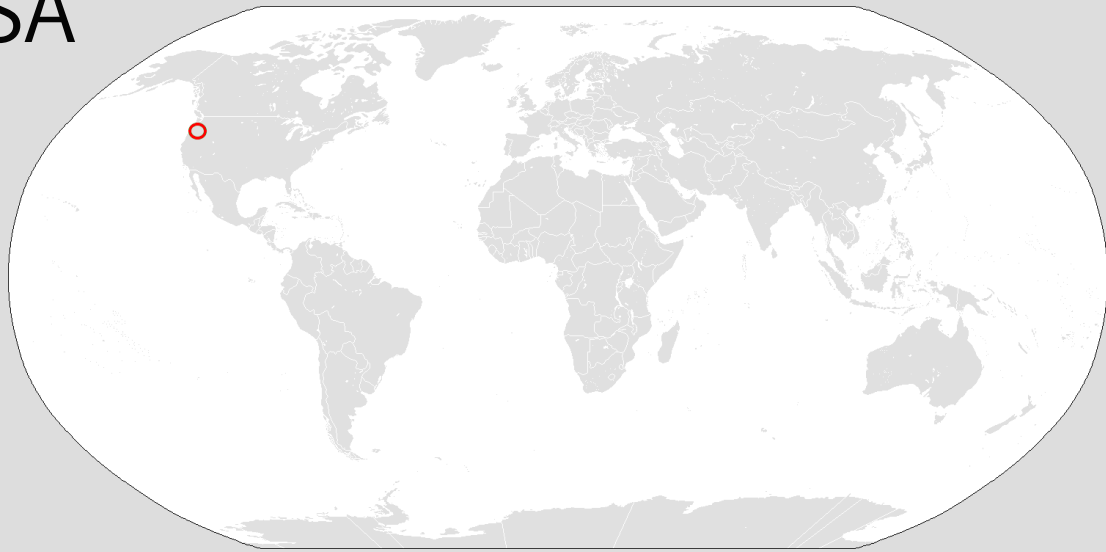
Rust

E. Dunham

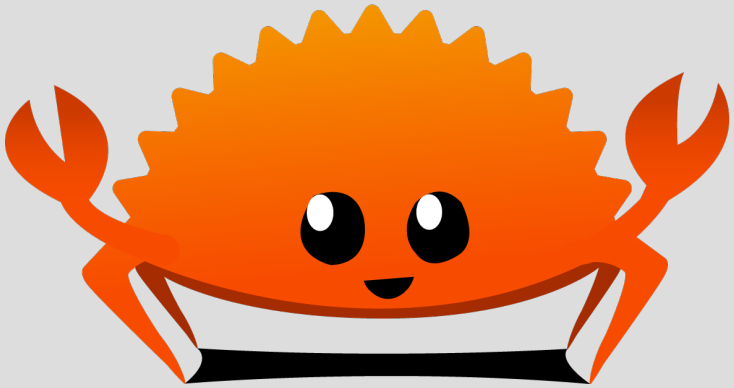
- DevOps Engineer, Mozilla Research
  - Rust, Servo CI & Infrastructure
- Rust Community Team member
- Portland, Oregon, USA
- Open Source nerd



The Rust Programming Language

- Started ~2010
- 1.0 Stable May 15, 2015
- 54380 commits, 1582 authors
- 79.1% “loved”, https://stackoverflow.com/research/developer-survey-2016
- Safe, Concurrent, Fast.

- Unofficial mascot: Ferris the Rustacean



Why Rust?

- Memory safety
- Easier concurrency
- Performance of C, C++ with 0-cost abstraction
- Hot code and resource-constrained systems
- Open development process & community

Why not Rust?

- Web pages
- Teams with expertise elsewhere
- Esoteric systems (PRs welcome!)
- Libraries/dependencies in other languages lose Rust's benefits
- When slow & buggy is ok

Community Links

- https://www.rust-lang.org/
- https://github.com/rust-lang/
- https://www.reddit.com/r/rust/
- https://users.rust-lang.org/
- http://stackoverflow.com/questions/tagged/rust
- irc.mozilla.org #rust #rust-beginners
- http://rustaceans.org/
- https://twitter.com/rustlang

Rust's Social Processes

Setting Expectations

"...most impressive aspect of Rust is the **welcoming community** that supports it. This community could become Rust's **not-so-secret weapon**."

- http://www.infoworld.com/article/2947214/open-source-tools/two-reasons-the-rust-language-will-succeed.html

"The Rust community seems to be **populated entirely by human beings.** I have no idea how this was done."

- http://scattered-thoughts.net/blog/2015/06/04/three-months-of-rust/

The Rust Code of Conduct

Conduct

Contact: rust-mods@rust-lang.org

- We are committed to providing a friendly, safe and welcoming environment for all, regardless of level of experience, gender, gender identity and expression, sexual orientation, disability, personal appearance, body size, race, ethnicity, age, religion, nationality, or other similar characteristic.
- On IRC, please avoid using overtly sexual nicknames or other nicknames that might detract from a friendly, safe and welcoming environment for all.
- Please be kind and courteous. There's no need to be mean or rude.
- Respect that people have differences of opinion and that every design or implementation choice carries a trade-off and numerous costs. There is seldom a right answer.
- Please keep unstructured critique to a minimum. If you have solid ideas you want to experiment with, make a fork and see how it works.
- We will exclude you from interaction if you insult, demean or harass anyone. That is not welcome behaviour. We interpret the term "harassment" as including the definition in the Citizen Code of Conduct; if you have any lack of clarity about what might be included in that concept, please read their definition. In particular, we don't tolerate behavior that excludes people in socially marginalized groups.
- Private harassment is also unacceptable. No matter who you are, if you feel you have been or are being harassed or made uncomfortable by a community member, please contact one of the channel ops or any of the Rust moderation team immediately. Whether you're a regular contributor or a newcomer, we care about making this community a safe place for you and we've got your back.
- Likewise any spamming, trolling, flaming, baiting or other attention-stealing behaviour is not welcome.

Moderation

These are the policies for upholding our community's standards of conduct in our communication channels, most notably in Rust-related IRC channels.

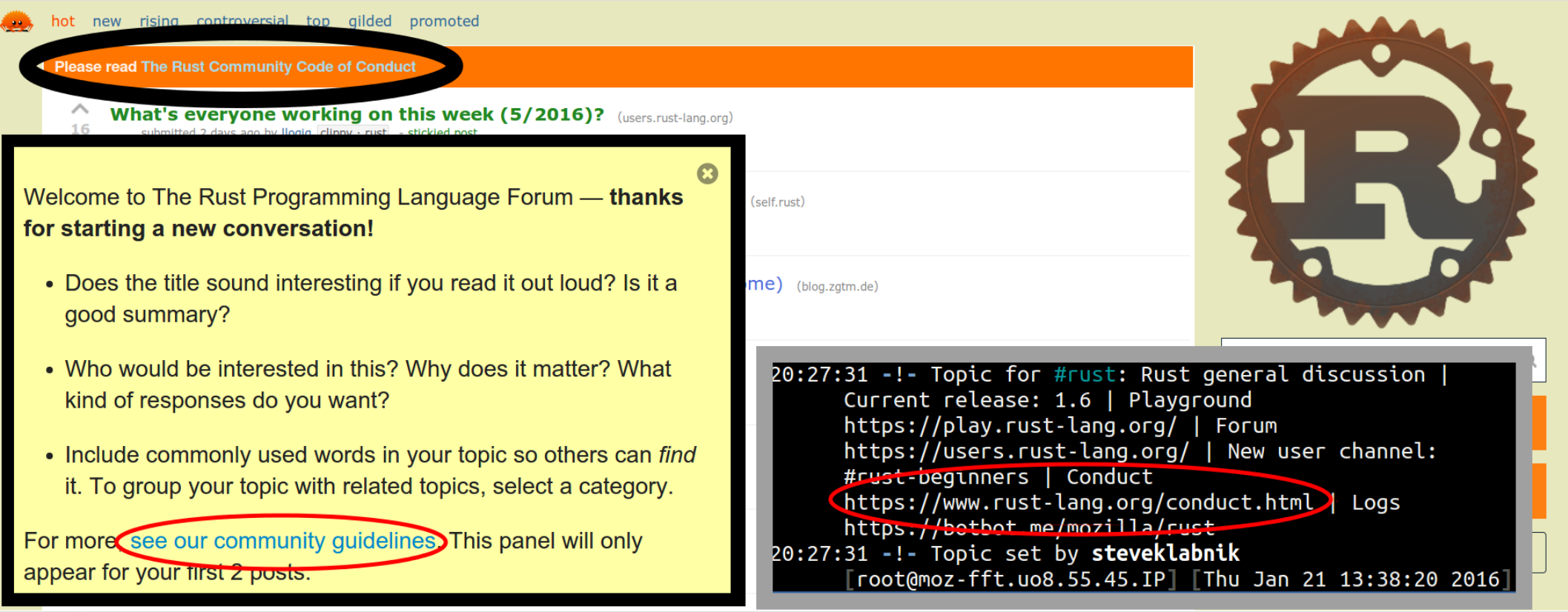
1. Remarks that violate the Rust standards of conduct, including hateful, hurtful, oppressive, or exclusionary remarks, are not allowed. (Cursing is allowed, but never targeting another user, and never in a hateful manner.)
2. Remarks that moderators find inappropriate, whether listed in the code of conduct or not, are also not allowed.
3. Moderators will first respond to such remarks with a warning.
4. If the warning is unheeded, the user will be "kicked," i.e., kicked out of the communication channel to cool off.
5. If the user comes back and continues to make trouble, they will be banned, i.e., indefinitely excluded.
6. Moderators may choose at their discretion to un-ban the user if it was a first offense and they offer the offended party a genuine apology.
7. If a moderator bans someone and you think it was unjustified, please take it up with that moderator, or with a different moderator, **in private**. Complaints about bans in-channel are not allowed.
8. Moderators are held to a higher standard than other community members. If a moderator creates an inappropriate situation, they should expect less leeway than others.

In the Rust community we strive to go the extra step to look out for each other. Don't just aim to be technically unimpeachable, try to be your best self. In particular, avoid flirting with offensive or sensitive issues, particularly if they're off-topic; this all too often leads to unnecessary fights, hurt feelings, and damaged trust; worse, it can drive people away from the community entirely.

And if someone takes issue with something you said or did, resist the urge to be defensive. Just stop doing what it was they complained about and apologize. Even if you feel you were misinterpreted or unfairly accused, chances are good there was something you could've communicated better — remember that it's your responsibility to make your fellow Rustaceans comfortable. Everyone wants to get along and we are all here first and foremost because we want to talk about cool technology. You will find that people will be eager to assume good intent and forgive as long as you earn their trust.

Adapted from the Node.js Policy on Trolling as well as the Contributor Covenant v1.3.0.

Automatic Reminders



Exclusion

“The Rust community gives me a particularly **bad feeling**. They're rather **tyrannical** about enforcing their code of conduct. They even have a **moderation attack squad** to go after anyone they deem to be an enemy! ... This sets off **warning alarms** for me.”

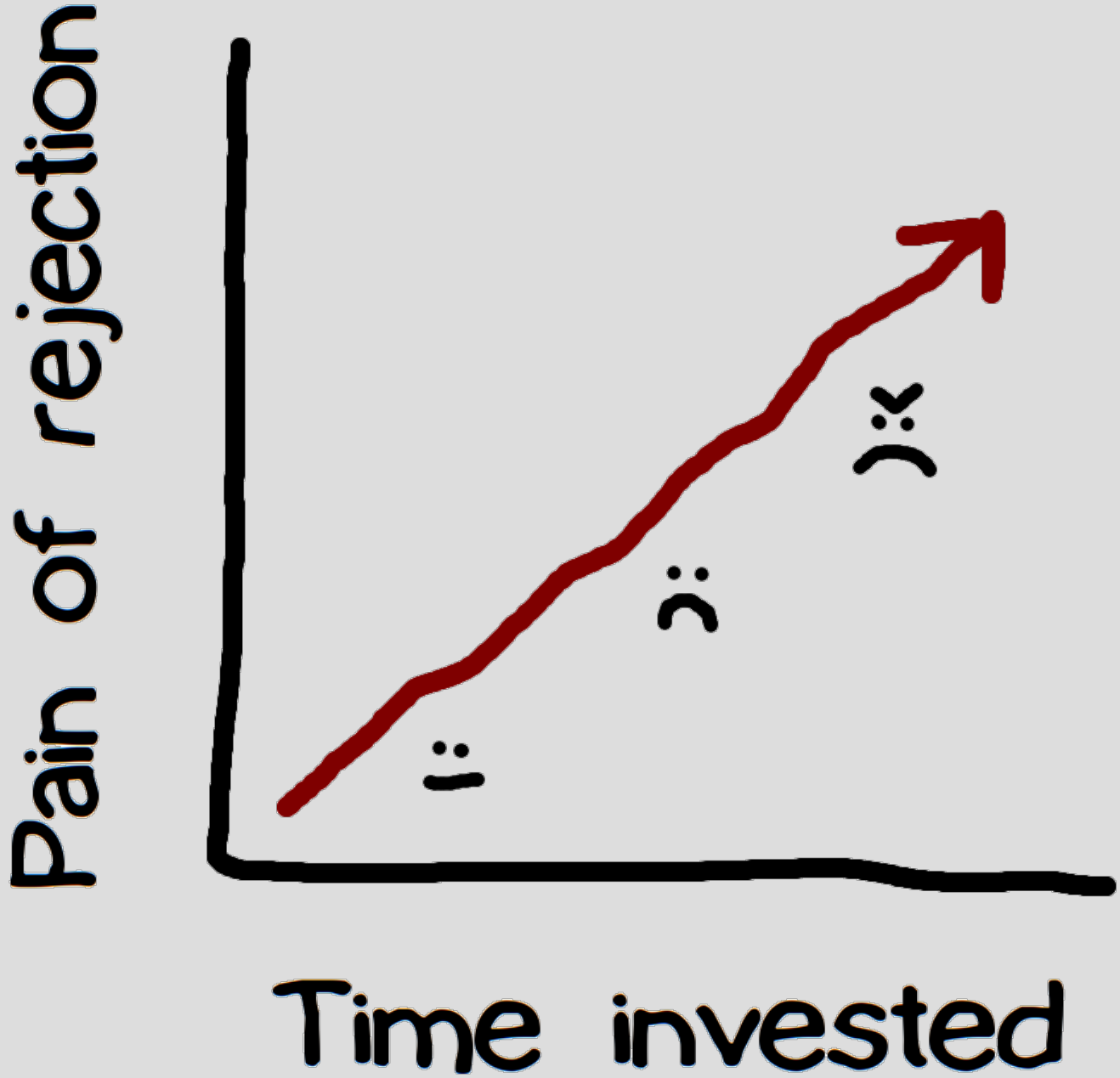
- https://developers.slashdot.org/comments.pl?sid=8652809&cid=51352141

Communication

"We ask that [larger changes] be put through a bit of a design process and produce a consensus

… so that all stakeholders can be confident about the direction the language is evolving in."

- https://github.com/rust-lang/rfcs/

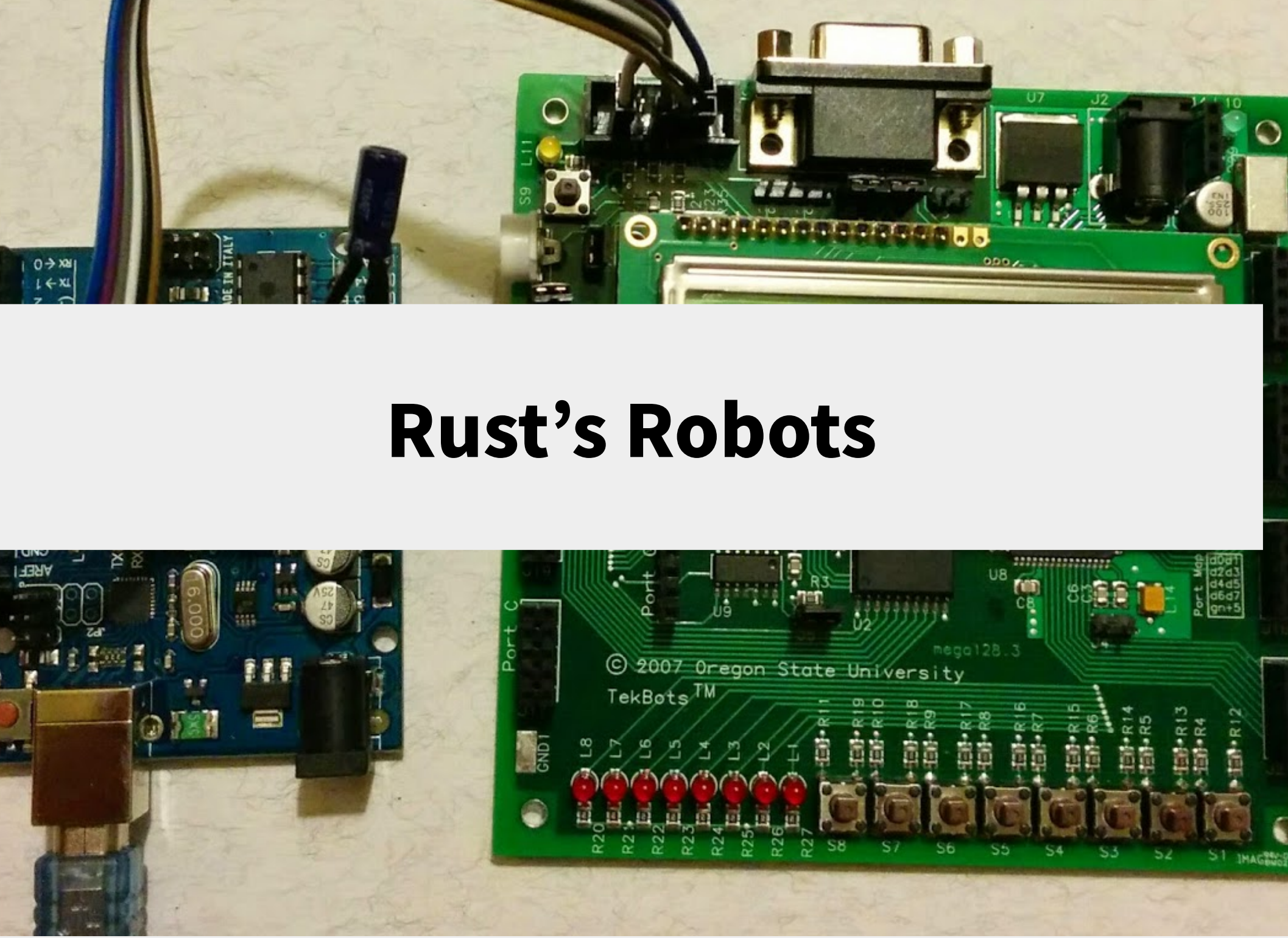

Pain of rejection
Time invested

Appreciation

- https://this-week-in-rust.org/
- http://blog.rust-lang.org/
- “Friends of the Tree”

Processes Recap

- High expectations
- Exclude the immoderate
- Require & simplify communication
- Show appreciation



Rust's Robots

AUTOMATICALLY MAINTAIN A REPOSITORY OF CODE THAT ALWAYS PASSES ALL THE TESTS

http://graydon.livejournal.com/186550.html

Rust: Bors & Homu

- https://github.com/servo/homu
- Test tree's state after landing r+'d PR

Welcoming Newbies

- https://github.com/nrc/highfive
- Offer guidance & assign reviewer

rust-highfive commented 4 hours ago — The Rust Programming Language member

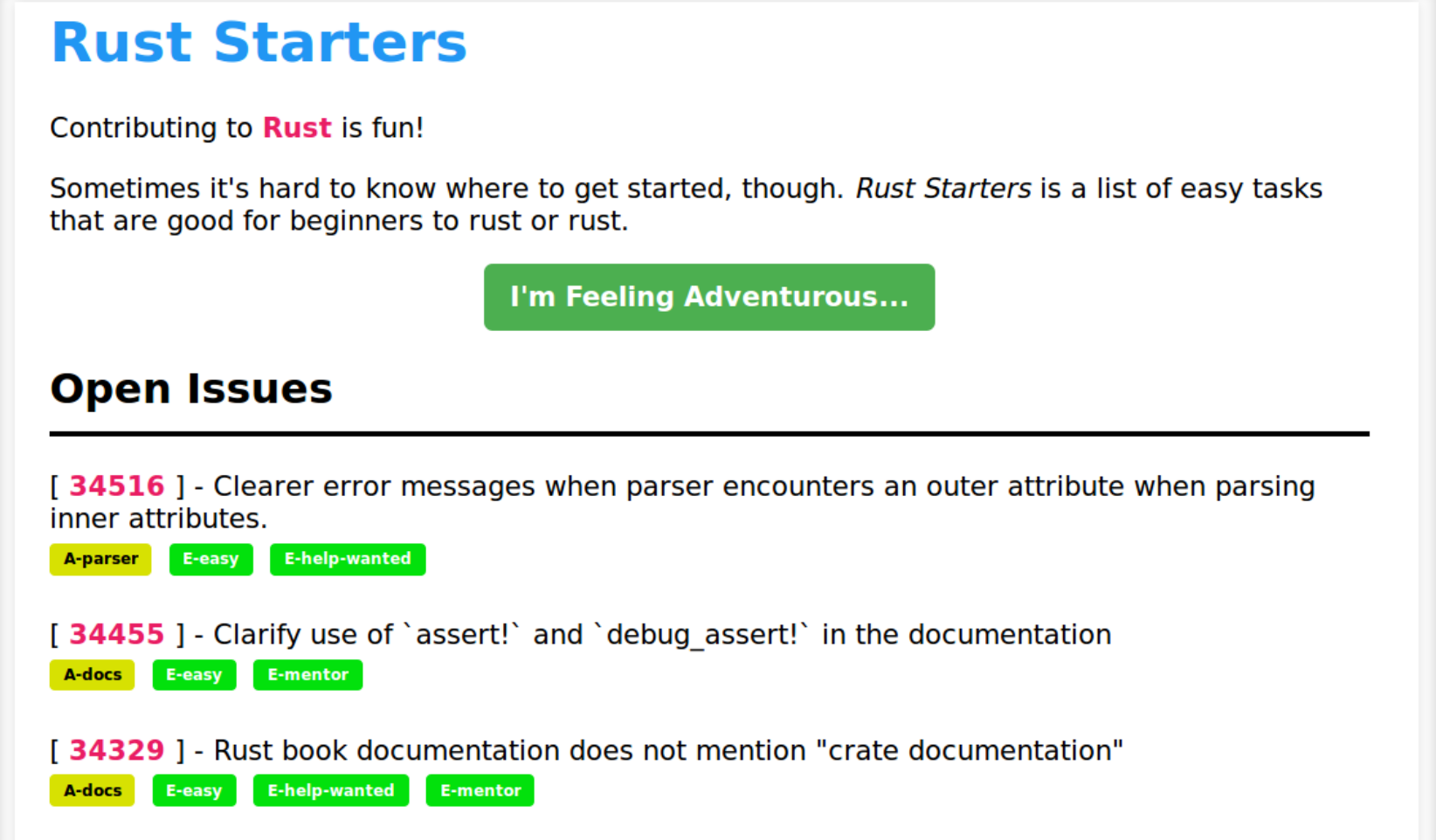
Thanks for the pull request, and welcome! The Rust team is excited to review your changes, and you should hear from **@alexcrichton** (or someone else) soon.

If any changes to this PR are deemed necessary, please add them as extra commits. This ensures that the reviewer can see what has changed since they last reviewed the code. Due to the way GitHub handles out-of-date commits, this should also make it reasonably obvious what issues have or haven't been addressed. Large or tricky changes may require several passes of review and changes.

Please see the contribution instructions for more information.

Mentorship & Guidance

- http://edunham.github.io/rust-starters/, https://starters.servo.org/



Where do good first bugs go?

- CONTRIBUTING.txt
- https://www.codemontage.com/
- http://www.codetriage.com/
- http://issuehub.io/
- http://up-for-grabs.net/
- http://yourfirstpr.github.io/
- https://openhatch.org

Robots Recap

- Automatically maintain a repository of code that always passes all the tests
- Welcome & guide new contributors
- Mark & share introductory bugs

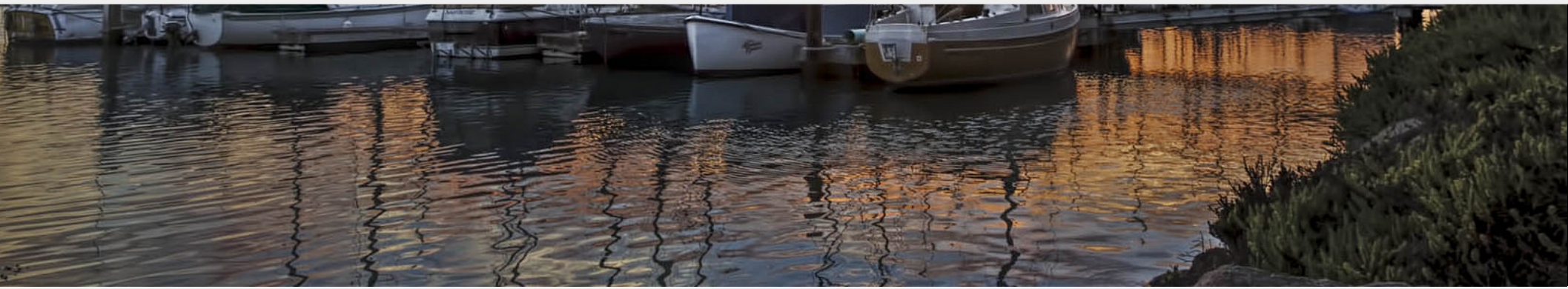
Rust Learning Resources

Language Docs

doc.rust-lang.org

Compare Similar Languages

github.com/ctjhoa/rust-learning/

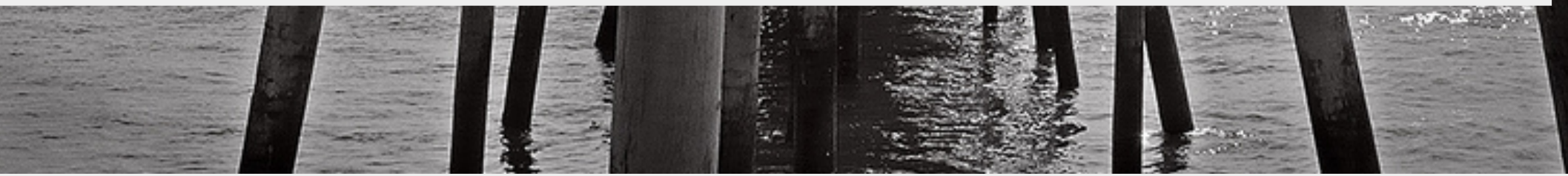


Community Support

www.rust-lang.org/community.html

users.rust-lang.org

#rust-beginners on irc.mozilla.org



Write Real Code

GitHub search “is:open is:issue language:rust”

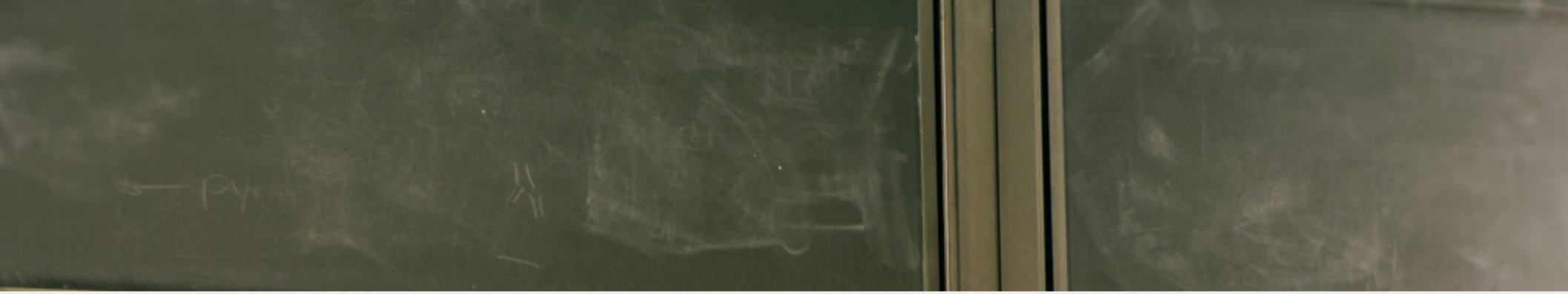
starters.servo.org

Read The Books

doc.rust-lang.org/book

O'Reilly, Programming Rust

Packt Publishing, Rust Essentials



Examine Examples

rustbyexample.com

Carol's rustlings

Rosetta Code

Find Good Tools

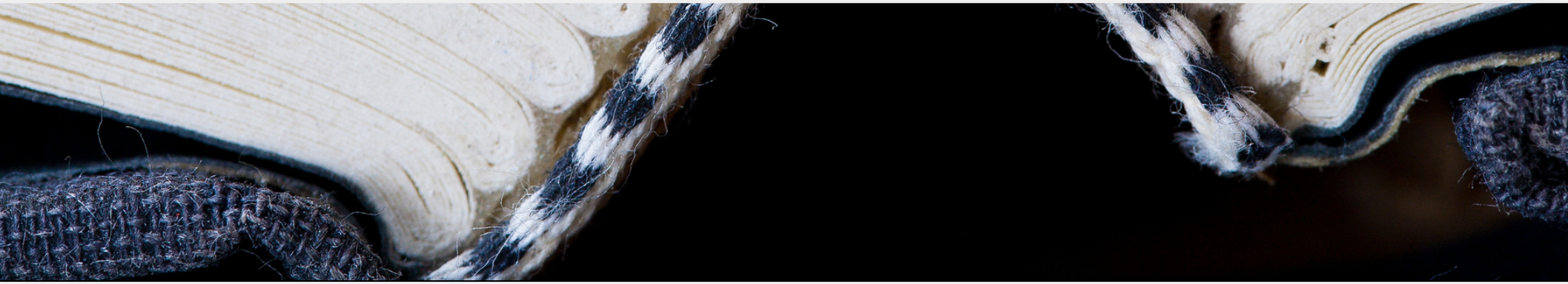
play.rust-lang.org

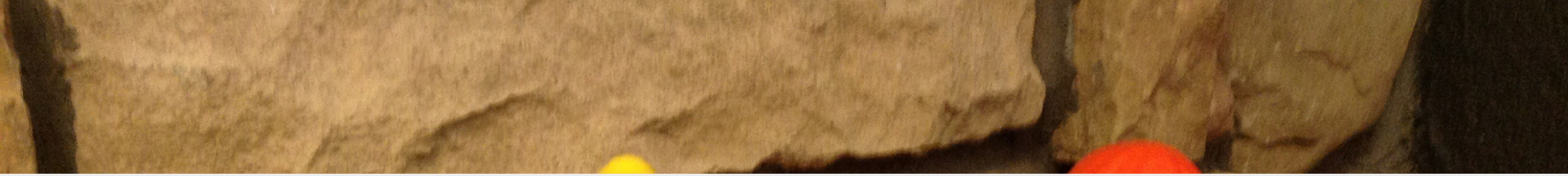
Clippy and Rustfmt

IDE support

Read Real Code

crates.io popular libraries
The Rust Compiler
Rust in Production





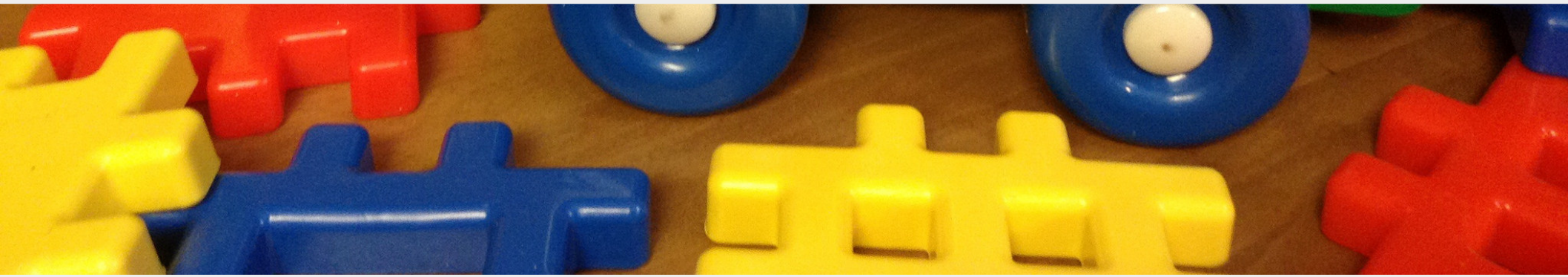
Write Toy Programs

Rust by Example

exercism.io

Project Euler

Hackerrank

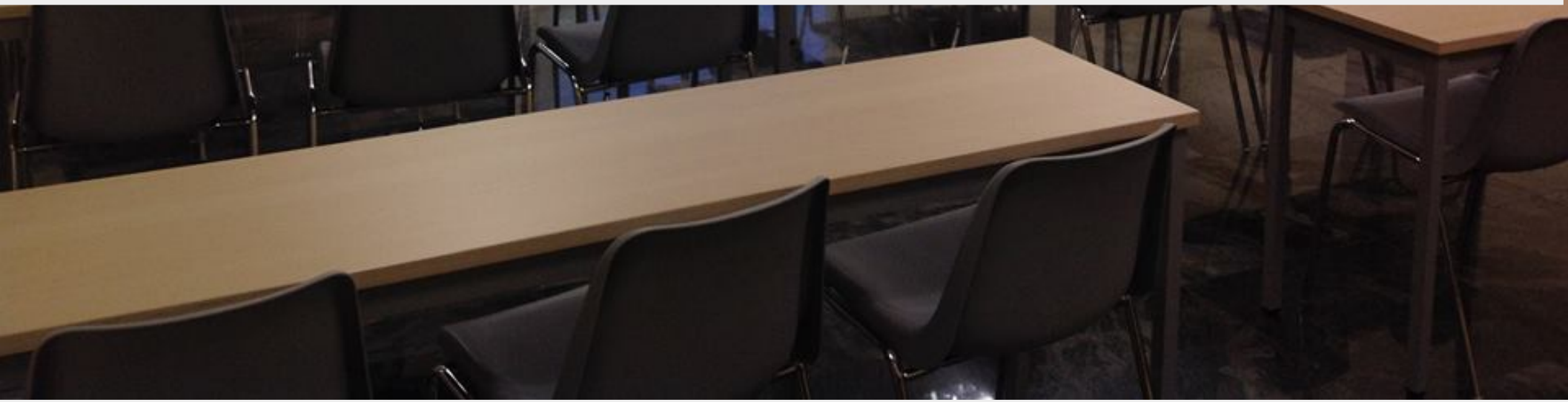


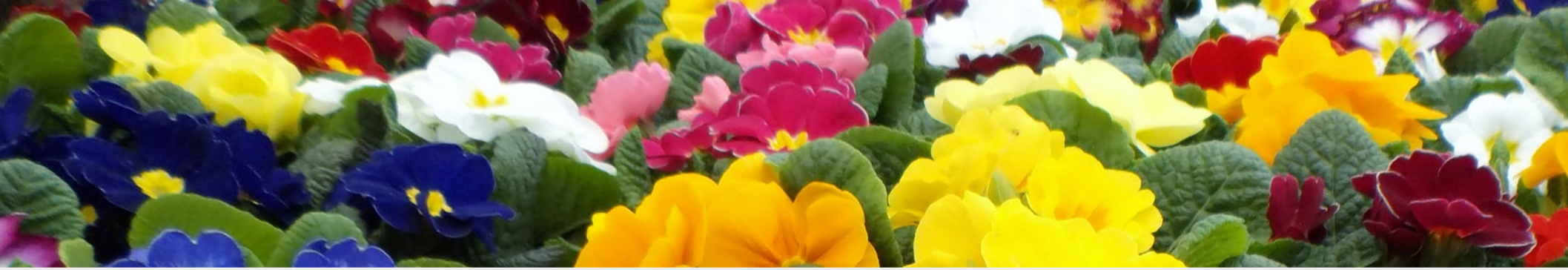
Google & Stackoverflow

Stackoverflow Rust tag

Watch Lectures & Courses

rust-learning list





Participate!

Rust subreddit

This Week In Rust

Errors are here to help

Getting Rust

- https://github.com/rust-lang-nursery/rustup.rs (manage multiple versions)
- http://rust-lang.org/downloads.html (single version)
- http://play.rust-lang.org (official, no libraries)
- http://play.integer32.com/ (has some popular crates)

Basic Syntax

```
1  // Main takes no arguments and returns nothing
2  fn main(){
3      // The function body is the *scope* inside these curly brace
4      // Create a variable. It owns a string.
5      let what_to_say = "Hello World";
6      // Meet print syntax
7      println!("This program says {}", what_to_say);
8  }
```

```
This program says Hello World

Program ended.
```

Scope Errors!

```
1  fn not_main(){
2      let what_to_say = "Hello World";
3  }
4  fn main(){
5      println!("This program says {}", what_to_say);
6  }
```

```
<anon>:5:38: 5:49 error: unresolved name `what_to_say` [E0425]
<anon>:5     println!("This program says {}", what_to_say);
                                              ^~~~~~~~~~~
```

Punctuation Errors

```
1 fn main(){
2     let what_to_say = "Hello World"
3     println!("This program says {}", what_to_say);
4 }
```

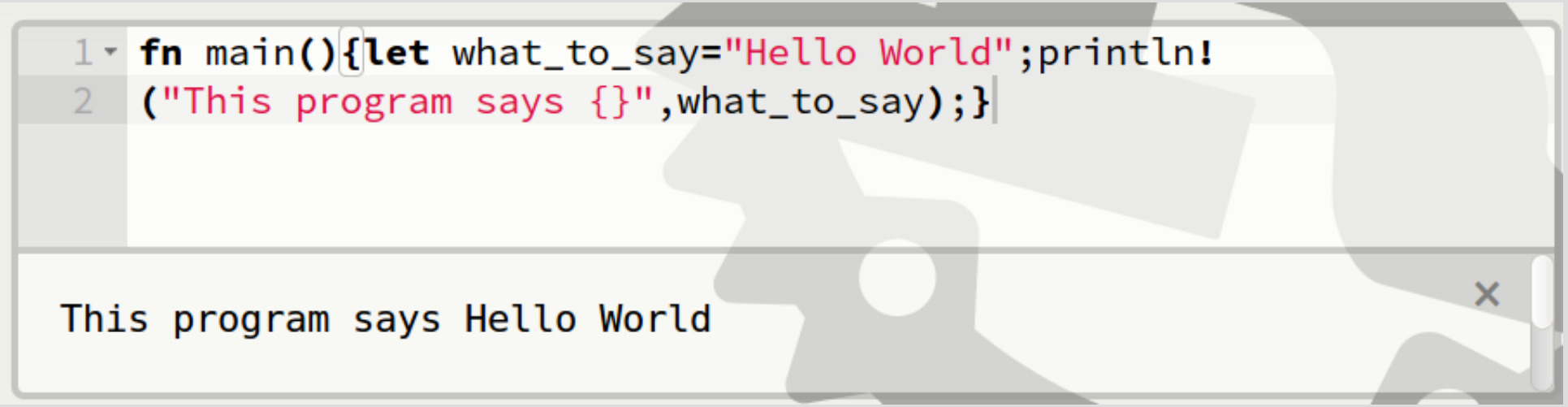
```
<anon>:3:5: 3:12 error: expected one of `.`, `;`, or an
operator, found `println`
<anon>:3     println!("This program says {}", what_to_say);
```

Unused Variables

```
1 fn main(){
2     let what_to_say = "Hello World";
3     println!("Hello");
4 }
```

```
<anon>:2:9: 2:20 warning: unused variable: `what_to_say`,
#[warn(unused_variables)] on by default
<anon>:2     let what_to_say = "Hello World";
```

Hey, Pythonistas!



Hey, Pythonistas!

```
1  fn
2  main
3  (
4      ){
5  let what_to_say
6      =
7  "Hello World"
8  ;println
9  !  (
10 "This program says {}"
11     , what_to_say
12 ) ;           }
```

```
This program says Hello World

Program ended.
```

Primitive types (built-in)

- `bool`
- `char`
- `i8, i16, i32, i64`
- `u8, u16, u32, u64`
- `f32, f64`
- `isize, usize`
- `str`
- `tuple`

https://doc.rust-lang.org/book/primitive-types.html

Function Type Signatures

```
fn  function name
      ( name : type ,
        name : type )
     -> result type {
     ...
}
```

Function with a type signature

```
1  fn sum(a: i32, b: i32, c: i32) -> i32{
2      a + b + c
3  }
4
5  fn main(){
6      println!("A sum is {}", sum(1, 2, 3))
7  }
```

```
A sum is 6

Program ended.
```

Synonymous return

```
1  fn sum(a: i32, b: i32, c: i32) -> i32{
2      return a + b + c;
3  }
4
5  fn main(){
6      println!("A sum is {}", sum(1, 2, 3))
7  }
```

```
A sum is 6

Program ended.
```

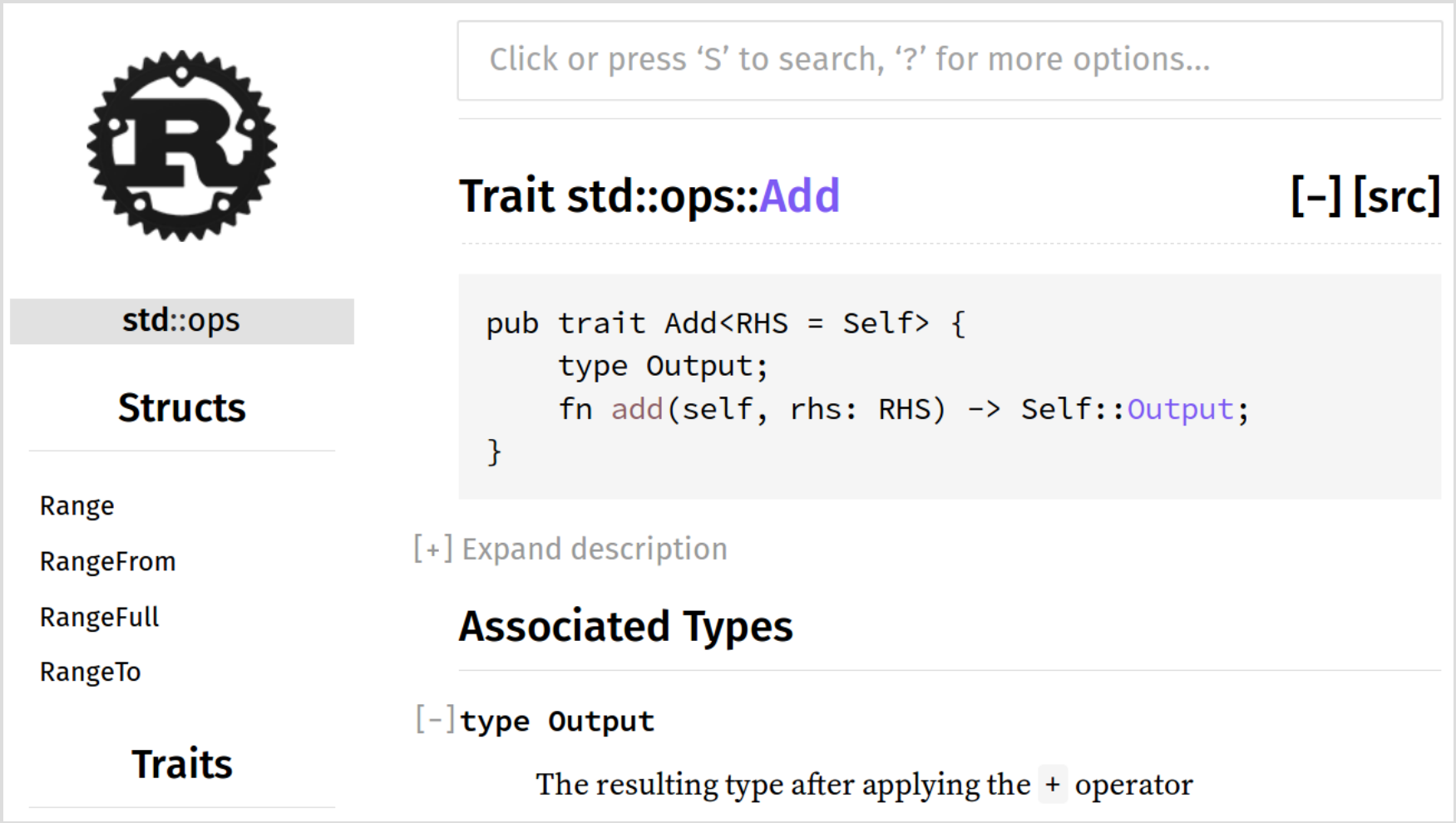
Type Errors

```
1  fn sum(a: i32, b: i32, c: i32) -> i32{
2      return a + b + c;
3  }
4
5  fn main(){
6      println!("A sum is {}", sum(1.1, 2.3, 3.2))
7  }
```

```
<anon>:6:33: 6:36 error: mismatched types:
 expected `i32`,
    found `_`
(expected i32,
    found floating-point variable) [E0308]
```

Anything you can add...

https://doc.rust-lang.org/std/ops/trait.Add.html



Traits

```
1  use std::ops::Add;
2
3  fn sum<T: Add<Output=T>>(a: T, b: T, c: T) -> T{
4      return a + b + c;
5  }
6
7  fn main(){
8      println!("A sum is {}", sum(1.1, 2.3, 3.2))
9  }
```

```
A sum is 6.6

Program ended.
```

Additional Resources

- https://doc.rust-lang.org/book/traits.html
- http://blog.rust-lang.org/2015/05/11/traits.html
- http://rustbyexample.com/trait.html
- http://pcwalton.github.io/blog/2012/08/08/a-gentle-introduction-to-traits-in-rust/

Ownership Rules

- No borrow may outlive value's owner
- Exactly 1 mutable reference (&mut T)
  - OR
- As many read-only references (&T) as you want

Giving Away a Value

```
1  fn main() {
2      let mystring = "I'm immutable!".to_string();
3      println!("{}", mystring);
4      let mut yarn = mystring; //yarn now owns the value
5      println!("{}", yarn);
6      yarn = "I have been mutated".to_string();
7      println!("{}", yarn);
8  }
9
```

```
I'm immutable!
I'm immutable!
I have been mutated

Program ended.
```

Given away means gone

```
1  fn main() {
2      let mystring = "I'm immutable!".to_string();
3      println!("{}", mystring);
4      let mut yarn = mystring; //yarn now owns the value
5      println!("{}", mystring);
6  }
7
```

```
<anon>:5:20: 5:28 error: use of moved value: `mystring` [E0382]
<anon>:5      println!("{}", mystring);
                             ^~~~~~~~
<std macros>:2:25: 2:56 note: in this expansion of format_args!
<std macros>:3:1: 3:54 note: in this expansion of print!
```

Borrow the value

```
1  fn main() {
2      let mystring = "I'm immutable!".to_string();
3      println!("{}", mystring);
4      let yarn = &mystring;
5      println!("{}", mystring);
6      println!("{}", yarn);
7  }
8
```

```
I'm immutable!
I'm immutable!
I'm immutable!

Program ended.
```

Make a mutable copy

```
1  fn main() {
2      let mystring = "I'm immutable!".to_string();
3      println!("{}", mystring);
4      let mut yarn = mystring.clone();
5      println!("{}", mystring);
6      println!("{}", yarn);
7      yarn = "I have been mutated".to_string();
8      println!("{}", yarn);
9  }
```

```
I'm immutable!
I'm immutable!
I'm immutable!
I have been mutated
```

Review

- Only owner can access value
- 1 mutable reference or unlimited read-only references to each value
- Borrow may not outlive owner

Additional Resources

- http://doc.rust-lang.org/stable/book/ownership.html
- http://doc.rust-lang.org/stable/book/references-and-borrowing.html
- Why the to_string()? http://hermanradtke.com/2015/05/03/string-vs-str-in-rust-functions.html
- http://rustbyexample.com/trait/clone.html
- https://www.reddit.com/r/rust/comments/2xxjda/when_should_my_type_be_copy/

How can we help?

- Email community-team@rust-lang.org
- IRC #rust-community on irc.mozilla.org
- Email me: edunham@mozilla.com or mozcommunity@edunham.net

Thank You

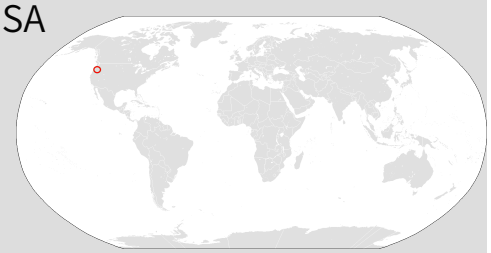
talks.edunham.net/mozcommunity-rust

Photos at https://www.flickr.com/photos/143305168@N08/favorites

Rust

E. Dunham

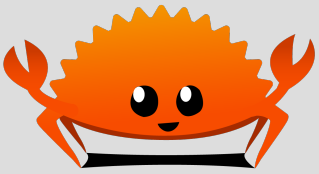
- DevOps Engineer, Mozilla Research
  - Rust, Servo CI & Infrastructure
- Rust Community Team member
- Portland, Oregon, USA
- Open Source nerd



The Rust Programming Language

- Started ~2010
- 1.0 Stable May 15, 2015
- 54380 commits, 1582 authors
- 79.1% “loved”, https://stackoverflow.com/research/developer-survey-2016
- Safe, Concurrent, Fast.

- Unofficial mascot: Ferris the Rustacean



Why Rust?

- Memory safety
- Easier concurrency
- Performance of C, C++ with 0-cost abstraction
- Hot code and resource-constrained systems
- Open development process & community

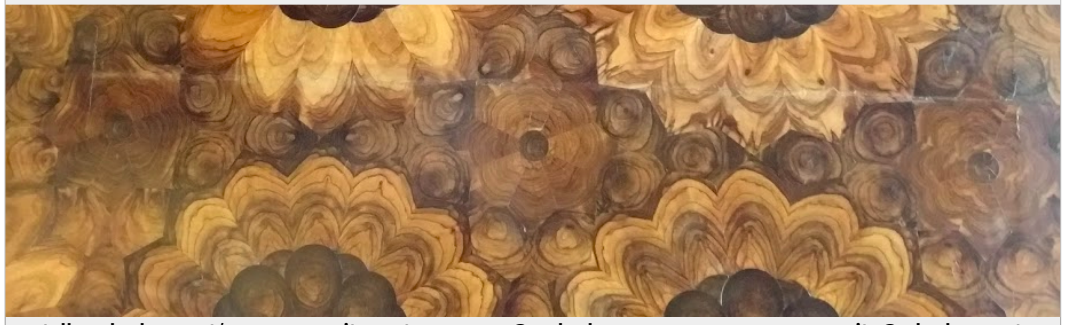
Why not Rust?

- Web pages
- Teams with expertise elsewhere
- Esoteric systems (PRs welcome!)
- Libraries/dependencies in other languages lose Rust's benefits
- When slow & buggy is ok

Community Links

- https://www.rust-lang.org/
- https://github.com/rust-lang/
- https://www.reddit.com/r/rust/
- https://users.rust-lang.org/
- http://stackoverflow.com/questions/tagged/rust
- irc.mozilla.org #rust #rust-beginners
- http://rustaceans.org/
- https://twitter.com/rustlang

Rust's Social Processes



Setting Expectations

"...most impressive aspect of Rust is the **welcoming community** that supports it. This community could become Rust's **not-so-secret weapon**."

- http://www.infoworld.com/article/2947214/open-source-tools/two-reasons-the-rust-language-will-succeed.html

"The Rust community seems to be **populated entirely by human beings.** I have no idea how this was done."

- http://scattered-thoughts.net/blog/2015/06/04/three-months-of-rust/

The Rust Code of Conduct

Conduct

Contact: rust-mods@rust-lang.org

- We are committed to providing a friendly, safe and welcoming environment for all, regardless of level of experience, gender, gender identity and expression, sexual orientation, disability, personal appearance, body size, race, ethnicity, age, religion, nationality, or other similar characteristic.
- On IRC, please avoid using overtly sexual nicknames or other nicknames that might detract from a friendly, safe and welcoming environment for all.
- Please be kind and courteous. There's no need to be mean or rude.
- Respect that people have differences of opinion and that every design or implementation choice carries a trade-off and numerous costs. There is seldom a right answer.
- Please keep unstructured critique to a minimum. If you have solid ideas you want to experiment with, make a fork and see how it works.
- We will exclude you from interaction if you insult, demean or harass anyone. That is not welcome behaviour. We interpret the term "harassment" as including the definition in the Citizen Code of Conduct; if you have any lack of clarity about what might be included in that concept, please read their definition. In particular, we don't tolerate behavior that excludes people in socially marginalized groups.
- Private harassment is also unacceptable. No matter who you are, if you feel you have been or are being harassed or made uncomfortable by a community member, please contact one of the channel ops or any of the Rust moderation team immediately. Whether you're a regular contributor or a newcomer, we care about making this community a safe place for you and we've got your back.
- Likewise any spamming, trolling, flaming, baiting or other attention-stealing behaviour is not welcome.

Moderation

These are the policies for upholding our community's standards of conduct in our communication channels, most notably in Rust-related IRC channels.

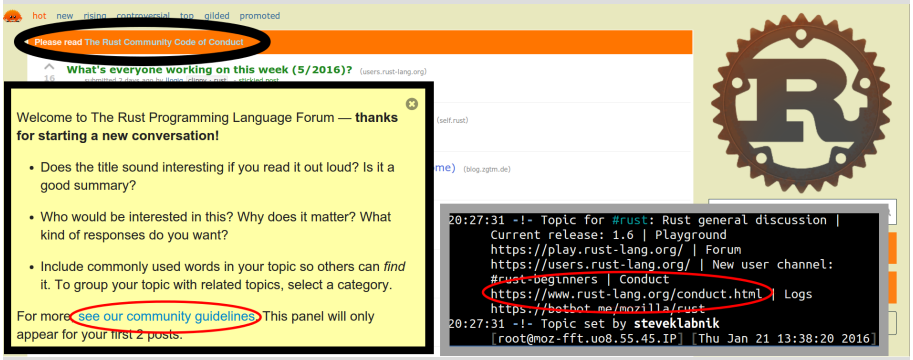
1. Remarks that violate the Rust standards of conduct, including hateful, hurtful, oppressive, or exclusionary remarks, are not allowed. (Cursing is allowed, but never targeting another user, and never in a hateful manner.)
2. Remarks that moderators find inappropriate, whether listed in the code of conduct or not, are also not allowed.
3. Moderators will first respond to such remarks with a warning.
4. If the warning is unheeded, the user will be "kicked," i.e., kicked out of the communication channel to cool off.
5. If the user comes back and continues to make trouble, they will be banned, i.e., indefinitely excluded.
6. Moderators may choose at their discretion to un-ban the user if it was a first offense and they offer the offended party a genuine apology.
7. If a moderator bans someone and you think it was unjustified, please take it up with that moderator, or with a different moderator, **in private**. Complaints about bans in-channel are not allowed.
8. Moderators are held to a higher standard than other community members. If a moderator creates an inappropriate situation, they should expect less leeway than others.

In the Rust community we strive to go the extra step to look out for each other. Don't just aim to be technically unimpeachable, try to be your best self. In particular, avoid flirting with offensive or sensitive issues, particularly if they're off-topic; this all too often leads to unnecessary fights, hurt feelings, and damaged trust; worse, it can drive people away from the community entirely.

And if someone takes issue with something you said or did, resist the urge to be defensive. Just stop doing what it was they complained about and apologize. Even if you feel you were misinterpreted or unfairly accused, chances are good there was something you could've communicated better — remember that it's your responsibility to make your fellow Rustaceans comfortable. Everyone wants to get along and we are all here first and foremost because we want to talk about cool technology. You will find that people will be eager to assume good intent and forgive as long as you earn their trust.

Adapted from the Node.js Policy on Trolling as well as the Contributor Covenant v1.3.0.

Automatic Reminders



Exclusion

“The Rust community gives me a particularly **bad feeling**. They're rather **tyrannical** about enforcing their code of conduct. They even have a **moderation attack squad** to go after anyone they deem to be an enemy! ... This sets off **warning alarms** for me.”

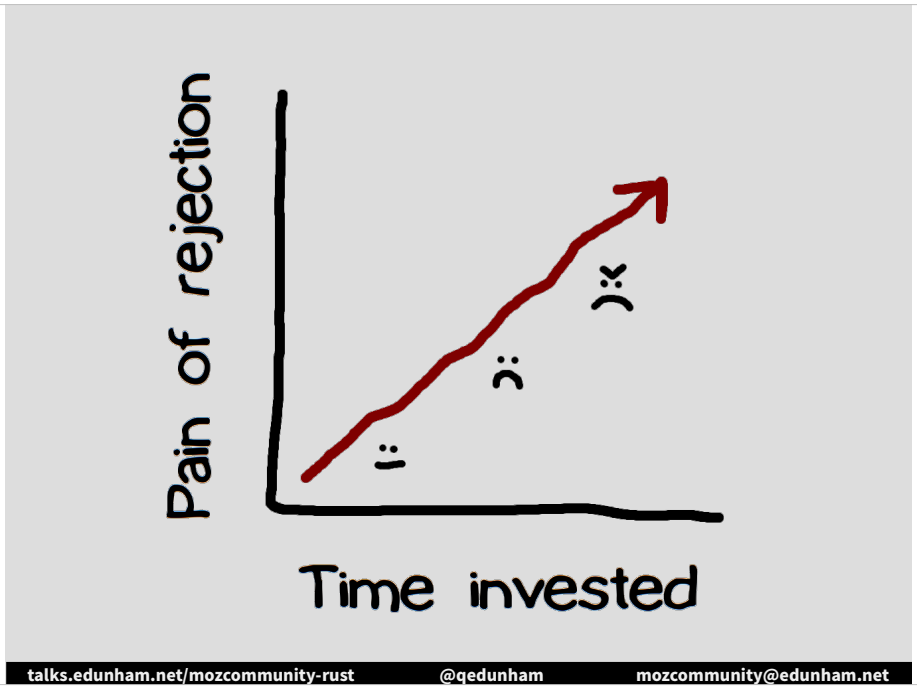
- https://developers.slashdot.org/comments.pl?sid=8652809&cid=51352141

Communication

"We ask that [larger changes] be put through a bit of a design process and produce a consensus

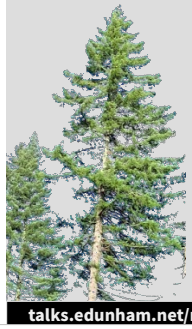
… so that all stakeholders can be confident about the direction the language is evolving in."

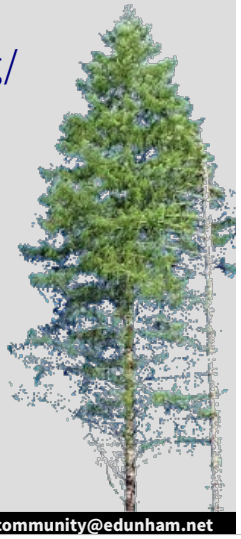
- https://github.com/rust-lang/rfcs/


Pain of rejection
Time invested

Appreciation

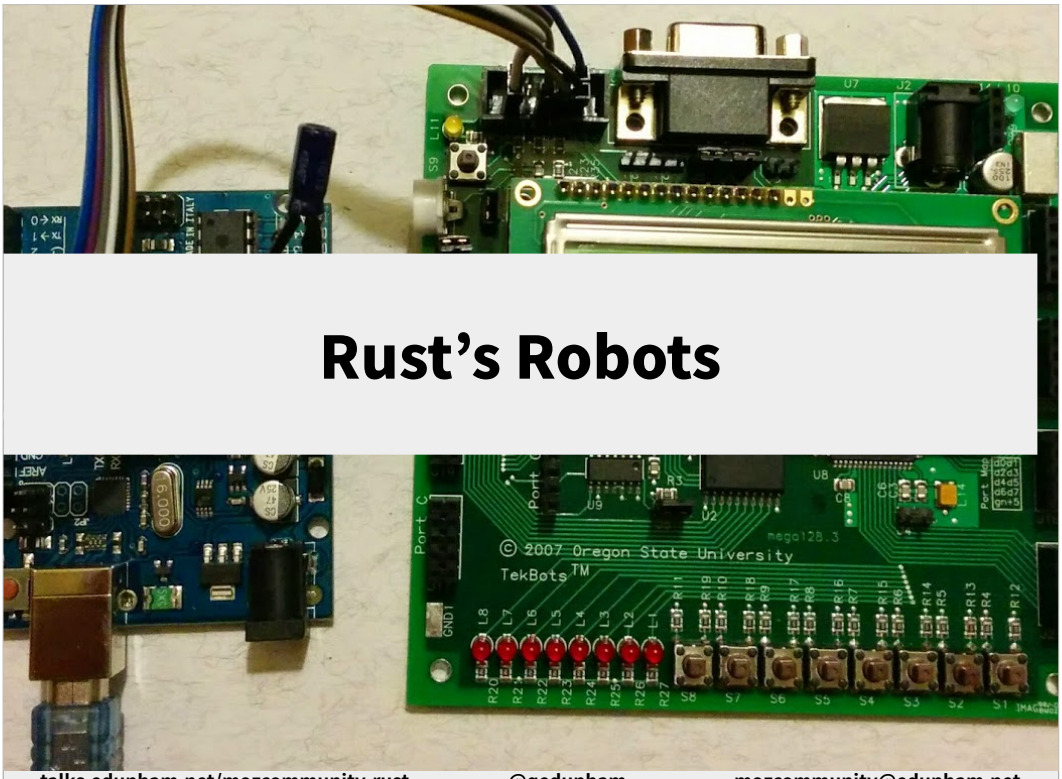
- https://this-week-in-rust.org/
- http://blog.rust-lang.org/
- “Friends of the Tree”





Processes Recap

- High expectations
- Exclude the immoderate
- Require & simplify communication
- Show appreciation



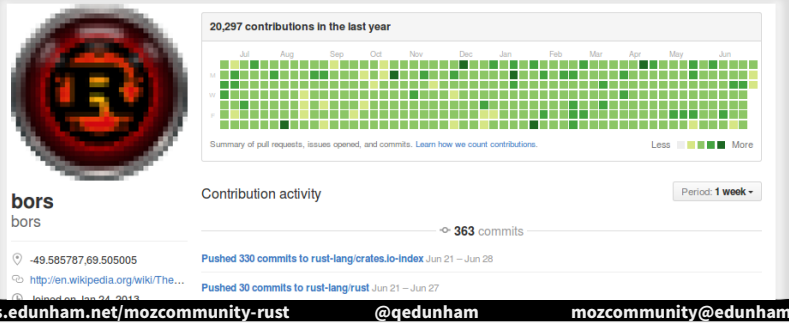
Rust's Robots

AUTOMATICALLY MAINTAIN A REPOSITORY OF CODE THAT ALWAYS PASSES ALL THE TESTS

http://graydon.livejournal.com/186550.html

Rust: Bors & Homu

- https://github.com/servo/homu
- Test tree's state after landing r+'d PR



Welcoming Newbies

- https://github.com/nrc/highfive
- Offer guidance & assign reviewer

rust-highfive commented 4 hours ago — The Rust Programming Language member

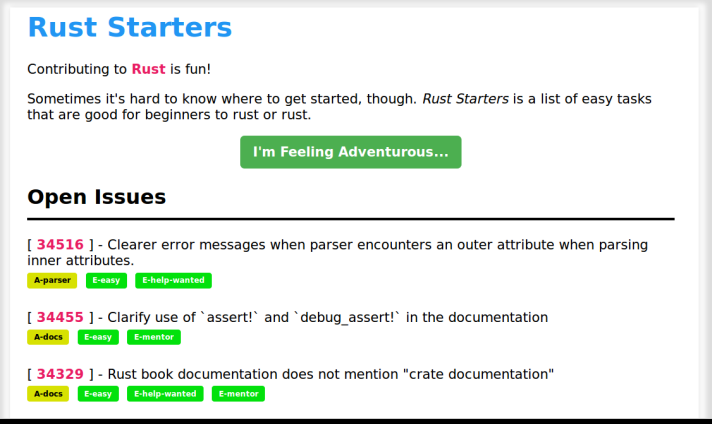
Thanks for the pull request, and welcome! The Rust team is excited to review your changes, and you should hear from **@alexcrichton** (or someone else) soon.

If any changes to this PR are deemed necessary, please add them as extra commits. This ensures that the reviewer can see what has changed since they last reviewed the code. Due to the way GitHub handles out-of-date commits, this should also make it reasonably obvious what issues have or haven't been addressed. Large or tricky changes may require several passes of review and changes.

Please see the contribution instructions for more information.

Mentorship & Guidance

- http://edunham.github.io/rust-starters/, https://starters.servo.org/



Where do good first bugs go?

- CONTRIBUTING.txt
- https://www.codemontage.com/
- http://www.codetriage.com/
- http://issuehub.io/
- http://up-for-grabs.net/
- http://yourfirstpr.github.io/
- https://openhatch.org

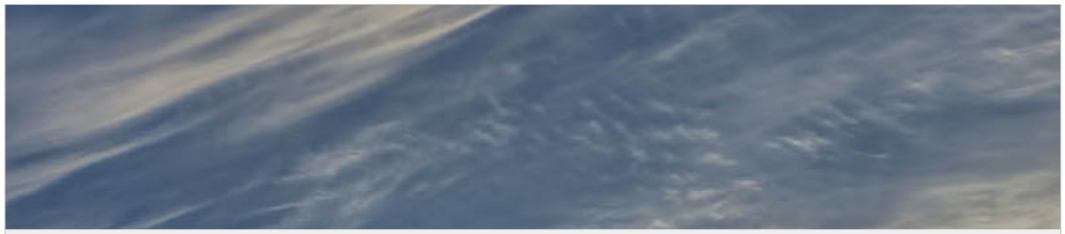
Robots Recap

- Automatically maintain a repository of code that always passes all the tests
- Welcome & guide new contributors
- Mark & share introductory bugs

Rust Learning Resources

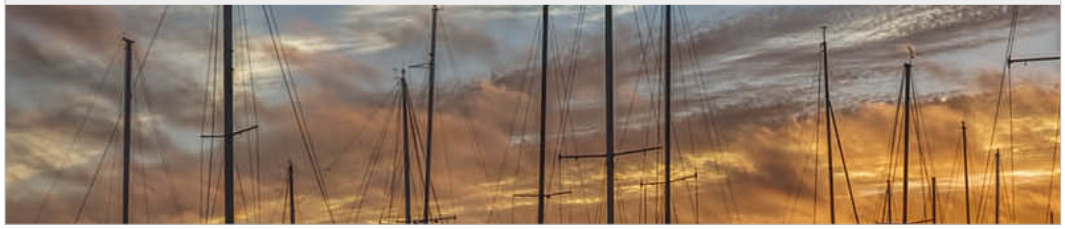
Language Docs

doc.rust-lang.org



Compare Similar Languages

github.com/ctjhoa/rust-learning/



Community Support

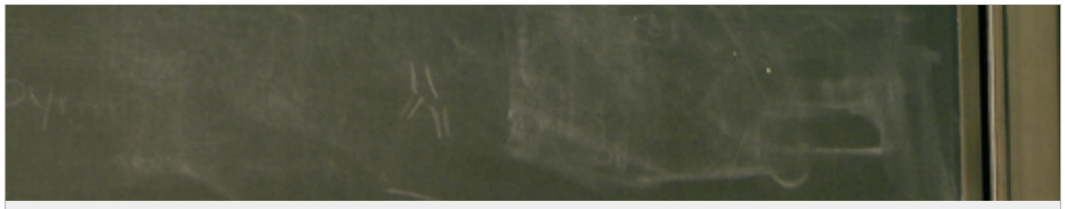
www.rust-lang.org/community.html
users.rust-lang.org
#rust-beginners on irc.mozilla.org

Write Real Code

GitHub search “is:open is:issue language:rust”
starters.servo.org

Read The Books

doc.rust-lang.org/book
O’Reilly, Programming Rust
Packt Publishing, Rust Essentials

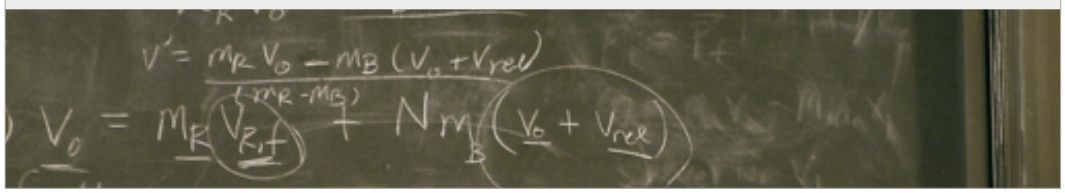


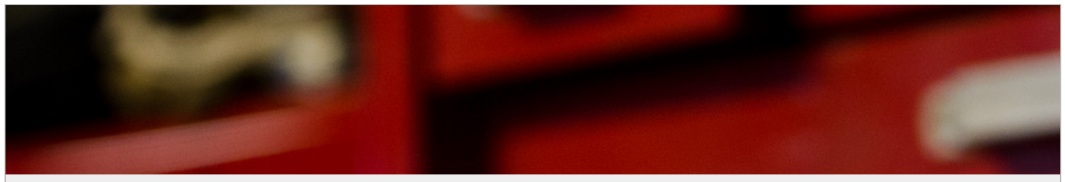
Examine Examples

rustbyexample.com

Carol's rustlings

Rosetta Code



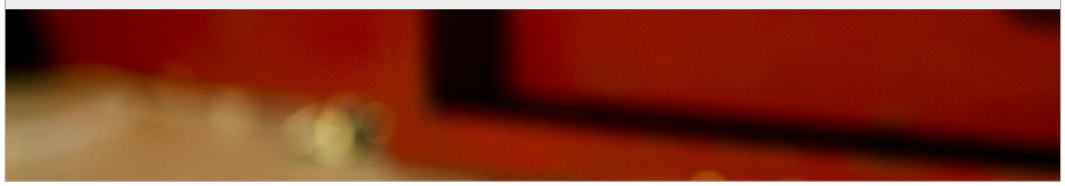


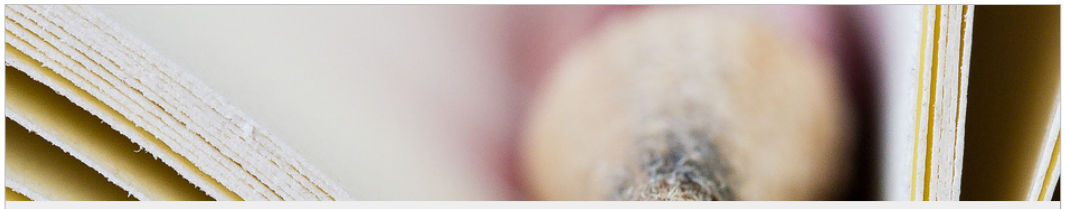
Find Good Tools

play.rust-lang.org

Clippy and Rustfmt

IDE support



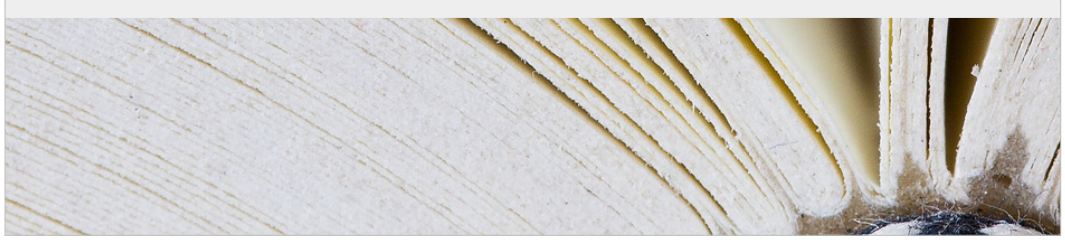


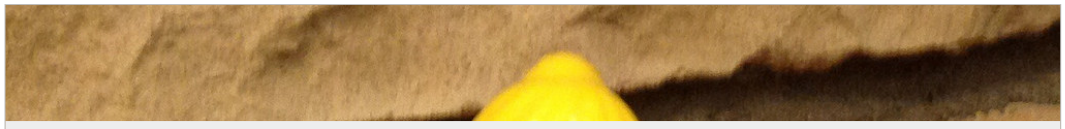
Read Real Code

crates.io popular libraries

The Rust Compiler

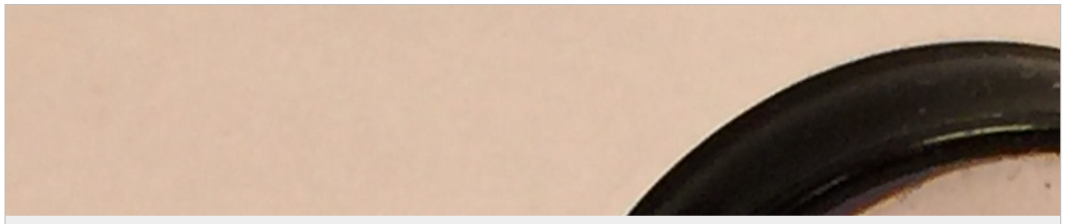
Rust in Production





Write Toy Programs

Rust by Example
exercism.io
Project Euler
Hackerrank



Google & Stackoverflow

Stackoverflow Rust tag

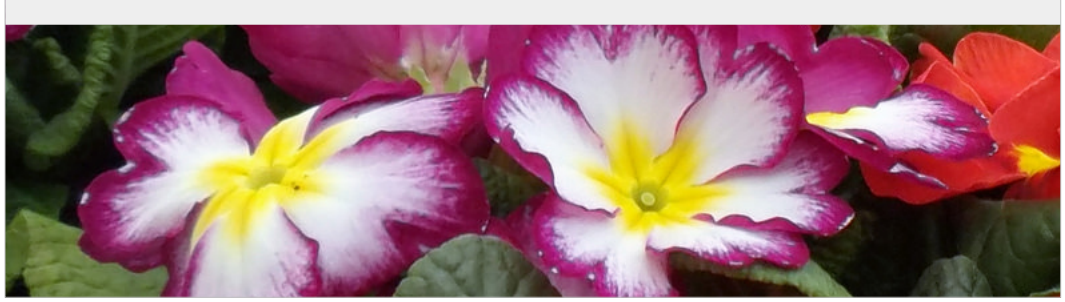
Watch Lectures & Courses

rust-learning list

Participate!

Rust subreddit

This Week In Rust



Errors are here to help

Getting Rust

- https://github.com/rust-lang-nursery/rustup.rs (manage multiple versions)
- http://rust-lang.org/downloads.html (single version)
- http://play.rust-lang.org (official, no libraries)
- http://play.integer32.com/ (has some popular crates)

Basic Syntax

```
1  // Main takes no arguments and returns nothing
2  fn main(){
3      // The function body is the *scope* inside these curly brace
4      // Create a variable. It owns a string.
5      let what_to_say = "Hello World";
6      // Meet print syntax
7      println!("This program says {}", what_to_say);
8  }
```

```
This program says Hello World

Program ended.
```

Scope Errors!

```
1  fn not_main(){
2      let what_to_say = "Hello World";
3  }
4  fn main(){
5      println!("This program says {}", what_to_say);
6  }
```

```
<anon>:5:38: 5:49 error: unresolved name `what_to_say` [E0425]
<anon>:5      println!("This program says {}", what_to_say);
                                              ^~~~~~~~~~~
```

Punctuation Errors

```
1 fn main(){
2     let what_to_say = "Hello World"
3     println!("This program says {}", what_to_say);
4 }
```

```
<anon>:3:5: 3:12 error: expected one of `.`, `;`, or an
operator, found `println`
<anon>:3     println!("This program says {}", what_to_say);
```

Unused Variables

```
1 fn main(){
2     let what_to_say = "Hello World";
3     println!("Hello");
4 }
```

```
<anon>:2:9: 2:20 warning: unused variable: `what_to_say`,
#[warn(unused_variables)] on by default
<anon>:2     let what_to_say = "Hello World";
```

Hey, Pythonistas!

```
1 fn main(){let what_to_say="Hello World";println!
2 ("This program says {}",what_to_say);}
```

```
This program says Hello World
```

Hey, Pythonistas!

```
1  fn
2  main
3  (
4      ){
5  let what_to_say
6      =
7  "Hello World"
8  ;println
9  !  (
10 "This program says {}"
11     , what_to_say
12 ) ;           }
```

```
This program says Hello World

Program ended.
```

Primitive types (built-in)

- `bool`
- `char`
- `i8, i16, i32, i64`
- `u8, u16, u32, u64`
- `f32, f64`
- `isize, usize`
- `str`
- `tuple`

https://doc.rust-lang.org/book/primitive-types.html

Function Type Signatures

```
fn function_name
    (name: type,
     name: type)
    -> result_type {
    ...
}
```

Function with a type signature

```
1  fn sum(a: i32, b: i32, c: i32) -> i32{
2      a + b + c
3  }
4
5  fn main(){
6      println!("A sum is {}", sum(1, 2, 3))
7  }
```

```
A sum is 6

Program ended.
```

Synonymous return

```
1  fn sum(a: i32, b: i32, c: i32) -> i32{
2      return a + b + c;
3  }
4
5  fn main(){
6      println!("A sum is {}", sum(1, 2, 3))
7  }
```

```
A sum is 6

Program ended.
```

Type Errors

```
1  fn sum(a: i32, b: i32, c: i32) -> i32{
2      return a + b + c;
3  }
4
5  fn main(){
6      println!("A sum is {}", sum(1.1, 2.3, 3.2))
7  }
```

```
<anon>:6:33: 6:36 error: mismatched types:
 expected `i32`,
    found `_`
(expected i32,
    found floating-point variable) [E0308]
```

Anything you can add...

https://doc.rust-lang.org/std/ops/trait.Add.html

Traits

```
1  use std::ops::Add;
2
3  fn sum<T: Add<Output=T>>(a: T, b: T, c: T) -> T{
4      return a + b + c;
5  }
6
7  fn main(){
8      println!("A sum is {}", sum(1.1, 2.3, 3.2))
9  }
```

```
A sum is 6.6

Program ended.
```

Additional Resources

- https://doc.rust-lang.org/book/traits.html
- http://blog.rust-lang.org/2015/05/11/traits.html
- http://rustbyexample.com/trait.html
- http://pcwalton.github.io/blog/2012/08/08/a-gentle-introduction-to-traits-in-rust/

Ownership Rules

- No borrow may outlive value's owner
- Exactly 1 mutable reference (&mut T)
  - OR
- As many read-only references (&T) as you want

Giving Away a Value

```
1  fn main() {
2      let mystring = "I'm immutable!".to_string();
3      println!("{}", mystring);
4      let mut yarn = mystring; //yarn now owns the value
5      println!("{}", yarn);
6      yarn = "I have been mutated".to_string();
7      println!("{}", yarn);
8  }
9
```

```
I'm immutable!
I'm immutable!
I have been mutated

Program ended.
```

Given away means gone

```
1  fn main() {
2      let mystring = "I'm immutable!".to_string();
3      println!("{}", mystring);
4      let mut yarn = mystring; //yarn now owns the value
5      println!("{}", mystring);
6  }
7
```

```
<anon>:5:20: 5:28 error: use of moved value: `mystring` [E0382]
<anon>:5      println!("{}", mystring);
                             ^~~~~~~~
<std macros>:2:25: 2:56 note: in this expansion of format_args!
<std macros>:3:1: 3:54 note: in this expansion of print!
```

Borrow the value

```
1  fn main() {
2      let mystring = "I'm immutable!".to_string();
3      println!("{}", mystring);
4      let yarn = &mystring;
5      println!("{}", mystring);
6      println!("{}", yarn);
7  }
8
```

```
I'm immutable!
I'm immutable!
I'm immutable!

Program ended.
```

Make a mutable copy

```
1  fn main() {
2      let mystring = "I'm immutable!".to_string();
3      println!("{}", mystring);
4      let mut yarn = mystring.clone();
5      println!("{}", mystring);
6      println!("{}", yarn);
7      yarn = "I have been mutated".to_string();
8      println!("{}", yarn);
9  }
```

```
I'm immutable!
I'm immutable!
I'm immutable!
I have been mutated
```

Review

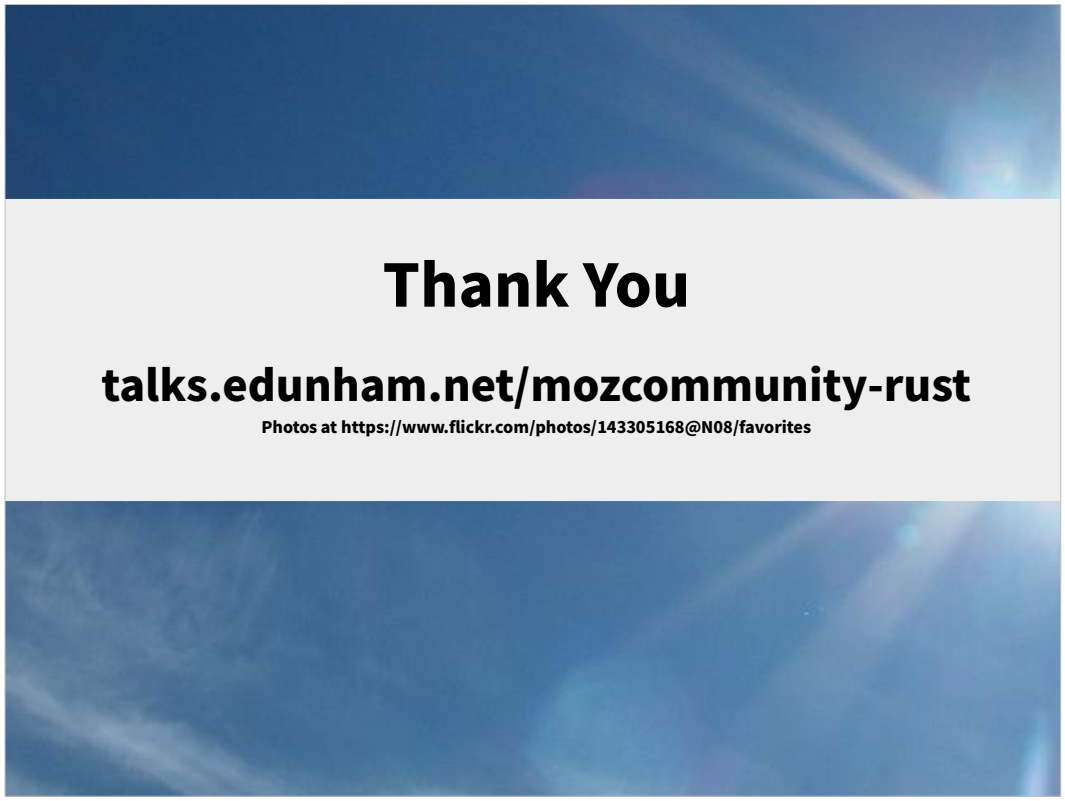
- Only owner can access value
- 1 mutable reference or unlimited read-only references to each value
- Borrow may not outlive owner

Additional Resources

- http://doc.rust-lang.org/stable/book/ownership.html
- http://doc.rust-lang.org/stable/book/references-and-borrowing.html
- Why the to_string()? http://hermanradtke.com/2015/05/03/string-vs-str-in-rust-functions.html
- http://rustbyexample.com/trait/clone.html
- https://www.reddit.com/r/rust/comments/2xxjda/when_should_my_type_be_copy/

How can we help?

- Email community-team@rust-lang.org
- IRC #rust-community on irc.mozilla.org
- Email me: edunham@mozilla.com or mozcommunity@edunham.net


Thank You
talks.edunham.net/mozcommunity-rust
Photos at https://www.flickr.com/photos/143305168@N08/favorites