# Rust 101

Deploy, Write, and Improve Rust

E. Dunham
@qedunham

2017-01-19
http://talks.edunham.net/lca2017/rust101.pdf

Rust 101

E. Dunham
@qedunham

Rust 101

E. Dunham
@qedunham

Welcome!

- 100 minutes
- "101" is an intro class
  - Learn what Rust is & isn't
  - Run Rust code
  - Meet Rust's special features
  - Improve Rust
- Not:
  - Hack time for you
  - Unsafe or advanced Rust
  - Exhaustive Q&A [1]

---

[1]Come to the BoF at lunch!

Rust 101

E. Dunham
@qedunham

You'll learn the key concepts
necessary for successful Rust
programming, as well as how to
continue exploring the language after
LCA.

*The Abstract*

Rust 101

E. Dunham
@qedunham

About Me:

- DevOps for Mozilla Research [2]
- Rust Community Team member
- FOSS & robotics background
- NOT a compiler wizard

---

[2]Infra is mostly Python

Rust 101

E. Dunham
@qedunham

Have you...

- Heard of Rust?
- Used Rust?
- Contributed to Rust?

Rust 101

E. Dunham
@qedunham

Have you...

- Written C, C++, Assembly, etc?
- Written Java, Python, Ruby, JS, etc?
- Written Haskell, Erlang, Ocaml, etc?

Rust 101

E. Dunham
@qedunham

Have you...

- Used version control?
- Used GitHub?
- Contributed to a FOSS project?

What's Rust?

- Systems language [3]
- Safety + Performance
- Community... (Thriving but controversial)

---

[3]Contrast to Go, a language for sysadmins

Rust 101

E. Dunham
@qedunham

Rust's Buzzwords

- Safety, Speed, Concurrency
- Memory safety without garbage collection
- Zero-cost abstractions
- Hack Without Fear

Rust 101

E. Dunham
@qedunham

Intro
 About Me
 About You
 About Rust

Run Rust

Write Rust

Improve Rust

Aside: Safety & GC

- Memory must be reused
- C: "Just follow these rules perfectly, you're smart"
- Java, JS, etc: "Wait a minute, I'll take care of it"
- Rust: "I'll prove correctness at compile time"

History

- Since ~2010
- 1.0 Stable in May 2015
- Currently [4] version 1.14.0
- Mozilla sponsorship & support

---

[4]until January 26th

Rust 101

E. Dunham
@qedunham

Notable Projects

- servo.org Browser Engine
- habitat.sh Infrastructure Tooling
- Dropbox (internal use)
- https://www.rust-lang.org/en-US/friends.html

Where is Rust a good tool?

- Speed + Safety essential
- LLVM-supported architecture
- Team ♡ new technology

Where might Rust be a bad tool?

- Timeframe prohibits new learning
- Need code reuse [5]
- Can't handle CoC

---

[5]Corrode can translate C to unsafe Rust

Rust 101

E. Dunham
@qedunham

Questions about Rust's place in the world?

Rust 101

E. Dunham
@qedunham

Intro

Run Rust

Channels
Installation
Options

Write Rust

Improve Rust

Rust's channels:

- Nightly: Trying neat ideas
- Beta: Release candidates
- Stable: Always backwards-compatible [6]

---

[6]https://blog.rust-lang.org/2015/05/15/Rust-1.0.html

Rust 101

E. Dunham
@qedunham

Which channel to use?

- Stable code should run anywhere
- Switch to nightly for dependencies
- New project? Pick stable [7]

---

[7]Unless you need an unstable feature

Aside: Crater

- Compile all published libraries
- Diff results from stable and candidate
- https://github.com/brson/taskcluster-crater

Rust 101

E. Dunham
@qedunham

Questions about channels & installation?

Installation:

- Online: `play.integer32.com` or `play.rust-lang.org`
- Many rusts: `rustup.rs`
- System package manager [8]
- Tinfoil hat: Compile from source

---

[8]Or add Rust to your favorite!

Playpens:

- Source at https://github.com/rust-lang/rust-playpen
- Choose output, LLVM IR, or ASM
- Gist your progress
- Config alters editor settings

Rustup:

- Docs at https://github.com/rust-lang-nursery/rustup.rs
- ~/.cargo/bin
- *rustup install nightly*
- *rustup run nightly cargo new*

Security:

- All releases GPG signed
- Key only held by Rust Core Team members
- `keybase.io/rust` (proved by local signing)
- Rustup checks signatures for you
- `github.com/rust-lang/rust`

Deployment

- Check README for system dependencies
- *git clone projecturl*
- *cd project*
- *cargo run*
    - Downloads any dependencies
    - Compiles deps & code
    - Executes src/main.rs

Rust 101

E. Dunham
@qedunham

Questions about installation?

Rust 101

E. Dunham
@qedunham

Intro

Run Rust

Write Rust
Errors
Syntax
Types & Traits
Safety
Using Libraries

Improve Rust

1. Intro
   - About Me
   - About You
   - About Rust

2. Run Rust
   - Channels
   - Installation Options

3. Write Rust
   - Errors
   - Syntax
   - Types & Traits
   - Safety
   - Using Libraries

4. Improve Rust
   - Level up
   - Find a project
   - Get involved

IDE support:

- https://areweideyet.com/

- http://www.jonathanturner.org/2017/01/rls-alpha-release.html

REPL equivalents:

- Use the playpen
- playbot on IRC
- https://github.com/murarth/rusti worked briefly on nightly

Rust wants you to succeed.

- Rules catch things that look unsafe
- "Unsafe" directive is an override
- Errors deserve helpful docs
- Click error number in playpen!

Erroneous Errors?

- Search the web!
- Gist your code
- Ask on IRC #rust-beginners
- File a bug

Aside: Other helpful tools

- https://github.com/nrc/rustfmt
- Rustfmt standardizes style for you
- https://github.com/manishearth/rust-clippy
- Clippy gives helpful suggestions

Rust 101

E. Dunham
@qedunham

Questions about errors?

Scope Syntax:

- Everything between matched {}
- Scopes can nest
- {Outer Scope {Inner Scope}}
- Pay attention to a value's scope!

Function Syntax:

- *fn myfunction { ... }*
- *fn myfunction (arg: type, arg: type) – ⟩ resulttype { ... }*
- Type signatures are like Mad Libs
- Function has at least name and scope

Rust 101

E. Dunham
@qedunham

Macro Syntax:

- Shorthand for functions with variable number of arguments
- *macroname!(foo, bar, baz)*
- `doc.rust-lang.org/beta/book/macros.html`
- You'll see "println!"

A Function:

```rust
fn halve(x: i32) -> i32 {
    return x / 2;
}

fn main(){
    println!("{}", halve(4));
}
```

Punctuation Matters:

- Expressions end with a semicolon
- Exception: bare expression on last line of function returns result
- Spaces separate tokens: *i32* is not i 32
- Whitespace is mostly irrelevant

Abusing Whitespace:

```
fn halve
(x                    :
i32)->i32
{
return
x
        /
                  2
; } fn main
()
{ println!("{}", halve
(
4
));}
```

Control Flow Syntax:

- Conditionals and loops are familiar
  - *if x { ... }*
  - *loop { ... }*
  - *while x { ... }*
  - *for x in 1..100 { ... }*
- Match statements combine conditionals
- https://doc.rust-lang.org/book/if.html
- https://doc.rust-lang.org/book/loops.html
- https://doc.rust-lang.org/book/match.html

Rust 101

E. Dunham
@qedunham

Intro

Run Rust

Write Rust
Errors
Syntax
Types & Traits
Safety
Using Libraries

Improve Rust

Matching on a variable:

```
fn main() {
    let day = 19;
    println!("January {} 2017 is:", day);
    match day {
        15 => println!("Travel to Hobart"),
        16 | 17 => println!("Miniconf Time"),
        18...20 => println!("The Conference"),
        _ => println!("not LCA at all"),
    }
}
```

Questions about basic syntax?

Rust 101

E. Dunham
@qedunham

Why Types & Traits?

- Describe characteristics of inputs and outputs

- Avoid allocating unneeded memory

- Remind humans how code works [9]

---

[9] Even type signatures that the compiler could infer must be spelled out.

Built-in types

- https://doc.rust-lang.org/book/primitive-types.html
- Primitives, arrays, strings, tuples

Custom types

- http://rustbyexample.com/custom_types.html
- Structs & Enums
- Use types from your dependencies (example in a few slides)

Traits

- https://doc.rust-lang.org/book/traits.html
- Traits describe a type's abilities
- You can tell Rust how type has trait with "impl"
- Generalize function's input and output

Rust 101

E. Dunham
@qedunham

Questions about types & traits?

Ownership:

- *let myint = 42;*
- "myint" is a variable binding
- "myint" **owns** the value 42
- Every value has exactly one owner
- See https://doc.rust-lang.org/book/ownership.html

Mutability:

- Owner can only change value if it's **mutable**
- *let mut myint = 42;*

Rust 101

E. Dunham
@qedunham

Changing the owner:

```
fn main() {
    let first = 42;
    println!("{}", first);

    let second = first;
    println!("{}", second);

    // this would be an error:
    // println!("{}", first);
}
```

Rust 101

E. Dunham
@qedunham

Borrowing

- Grant temporary access to a value
- 1 mutable borrow XOR unlimited immutable borrows
- Syntax: &myvar
- https://doc.rust-lang.org/book/references-and-borrowing.html

Rust 101

E. Dunham
@qedunham

Lifetimes

- Remember {scopes}?
- Variables disappear when their scope ends!
- No borrow may outlive its value's owner.

Rust 101

E. Dunham
@qedunham

Questions about safety?

Rust 101

E. Dunham
@qedunham

Package Management

- Package manager: Cargo
- Libraries: Crates
- Package index: crates.io
- See "Package Managers All The Way Down", Tasman B, 3:40pm today
- See http://doc.crates.io/guide.html

Create a binary or library

- *cargo new –bin*
    - Binary
    - You run main.rs
    - cargo.lock stores state of last good build
- *cargo new*
    - Library
    - src/lib.rs, no main function

Rust 101

E. Dunham
@qedunham

'cargo new' creates:

- Cargo.toml
- src/main.rs or src/lib.rs
- .git if absent

Rust 101

E. Dunham
@qedunham

Depending on a crate

- Search crates.io
- Search the web, check recent blogs
- Check docs, license, & project policies
- Add it to 'dependencies' section of cargo.toml
  - *name = "0.1"*
  - *name = { git = "https://github.com/org/repo.git" rev = "123abcd" }*
- *extern crate rand; use rand::Rng;*

Rust 101

E. Dunham
@qedunham

Questions about libraries?

Rust 101

E. Dunham
@qedunham

1. Intro
   - About Me
   - About You
   - About Rust

2. Run Rust
   - Channels
   - Installation Options

3. Write Rust
   - Errors
   - Syntax
   - Types & Traits
   - Safety
   - Using Libraries

4. Improve Rust
   - Level up
   - Find a project
   - Get involved

Always...

- Respect others' licenses
- License your own code
- Document & share what you learn!

Read a Book

- https://doc.rust-lang.org/stable/book/
- https://doc.rust-lang.org/nomicon/
- O'Reilly Book coming soon

Follow the News

- http://www.newrustacean.com/ podcast
- https://soundcloud.com/posix4e/sets/rustyradio interviews
- https://this-week-in-rust.org/ Weekly newsletter
- https://blog.rust-lang.org/ Official Blog

Practice

- https://github.com/carols10cents/rustlings
- http://rustbyexample.com/

Rust 101

E. Dunham
@qedunham

Questions about learning more Rust?

Join a project

- https://crates.io/, find popular crates
- Search GitHub "is:issue label:easy language:rust"

Port something

- https://github.com/jameysharp/corrode
- linux.conf.au/schedule/presentation/51/ Friday 1:20pm
- https:
  //blog.rust-lang.org/2015/04/24/Rust-Once-Run-Everywhere.html

Rust 101

E. Dunham
@qedunham

Questions about finding a project?

File or Fix issues

- github.com/rust-lang/rust
- We triage regularly
- If in doubt, ask on IRC first

Chat Online

- IRC: #rust, #rust-beginners on irc.mozilla.org
- `users.rust-lang.org` Users Forum
- Reddit, StackOverflow, etc.

Rust 101

E. Dunham
@qedunham

Find or join a meetup

- Search for your area + Rust meetup
- Community Team Calendar, goo.gl/EJ2iRb
- #rust-community on Mozilla IRC

Attend a conference

- `rustconf.com` Oregon, September
- `www.rust-belt-rust.com`, Pennsylvania, October
- `www.rustfest.eu`, Europe, September

Rust 101

E. Dunham
@qedunham

Questions about getting involved?

Rust 101

E. Dunham
@qedunham

Attend the LCA Rustlang BoF!

- Right here, lunch today
- Start Hacking

Rust 101

E. Dunham
@qedunham

Thank you!