Introduction to rust

Muharem Hrnjadovic

Rackspace International

@al_maisan

Yet another language .. WTH?

- Rust started by Graydon Hoare in 2006
- Mozilla (research) got involved in 2009
- Experimental browser (servo)
  - More parallelism
  - Less C++ bugs → less vulnerabilities
- “Opinionated version of C++”

good talk on rust (linux.conf.au 2014)

Status

- Current release: 0.9
- Fast moving, lots of (breaking) changes
- But: slowing down
- Stable candidate at some point this year

Best way to pick up rust?

- Language tutorial
- Standard library
- #rust on irc.mozilla.org

- Write some code!

Features

http://i.huffpost.com/gen/1539359/thumbs/n-VACUUM-CLEANER-570.jpg

rackspace® the open cloud company

OMG! Features!

- Ambition:

  safe, concurrent, practical, static systems language

- Big language, lots of influences
  - Haskell (declarations, pattern matching(?), parts of std lib)
  - Erlang (lightweight processes)
  - Golang? (channels, slices)
  - C++

Moar features!

- Static typing with type inference
- Higher-order functions
- Pattern matching and algebraic data types
- Polymorphism
- Syntax extensions (via macros)

Even moar features :-P

- Generics
- Traits (Java interfaces, haskell type classes)

```
// This does
fn head<T: Clone>(v: &[T]) -> T {
    v[0].clone()
}
```

- Futures (Python? Clojure?)

Eierlegende Wollmilchsau?

http://www.forexfactory.com/attachment.php?attachmentid=255957&stc=1&d=1244150501

rackspace® the open cloud company

Ownership & borrowing

- C++ facilitates harakiri in a million ways
- rust aims to enforce safe C++ patterns at compile time
- Memory always has single ownership
- Borrowing:
  - passing to a function
  - sending to another task
- Details here: talk on rust @ linux.conf.au

Crates/modules

- Crate:
  - Unit of compilation and linking
  - All sources that had to be compiled for a binary
  - Contains module hierarchy
- Pass crate root to `rustc` (it will find all the rest)
- functions are private by default
- struct members are public by default

example

```
mod farm {
    pub struct Farm {
        priv chickens: ~[Chicken],
        farmer: Human
    }

    impl Farm {
        fn feed_chickens(&self) { ... }
        pub fn add_chicken(&self, c: Chicken) { ... }
    }

    pub fn feed_animals(farm: &Farm) {
        farm.feed_chickens();
    }
}

fn main() {
    let f = make_me_a_farm();
    f.add_chicken(make_me_a_chicken());
    farm::feed_animals(&f);
```

Testing rust code

- Unit testing is easy
- Unit tests live in the same file as the code under test
- Recompilation (with `–test`) required

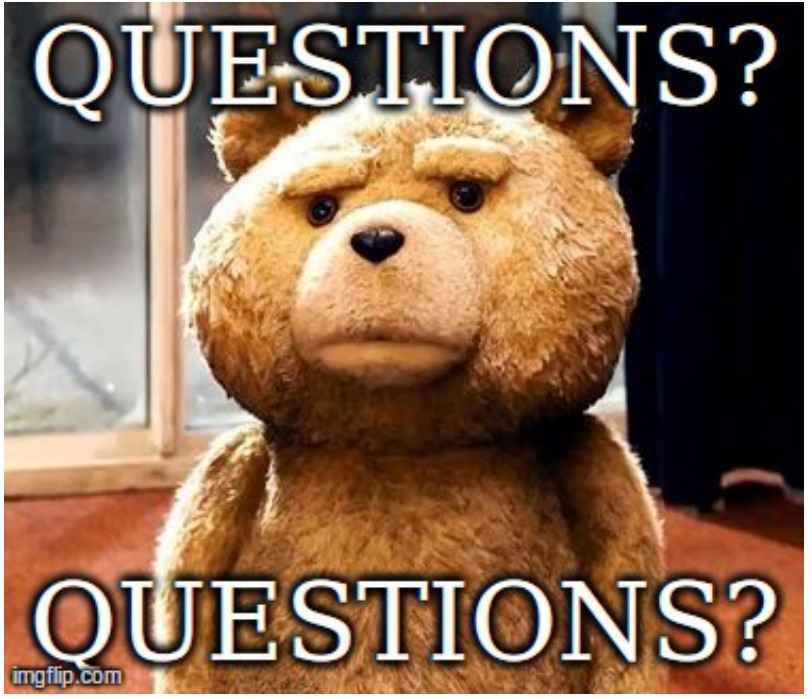
Documenting rust code

- Use rustdoc to generate docs from comments
- Not very sophisticated at this point
- e.g. no annotation for function params

demo time!

http://www.the-editing-room.com/img/Drive-Angry-Flying-Car-Explosion-1024x679.jpg

rackspace® the open cloud company



rackspace® the open cloud company