Hands-on introduction to Rust

Agenda (1/2)

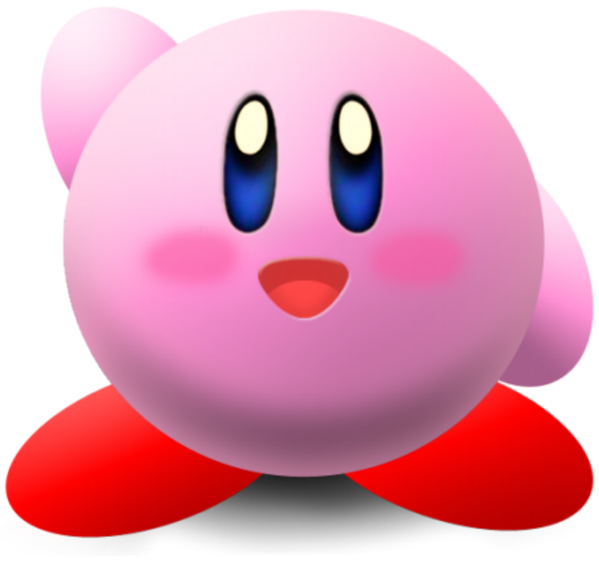
1. Cargo
2. Basics and documentation
3. Iterating
4. Making our own types
5. Strings and user input

Agenda (2/2)

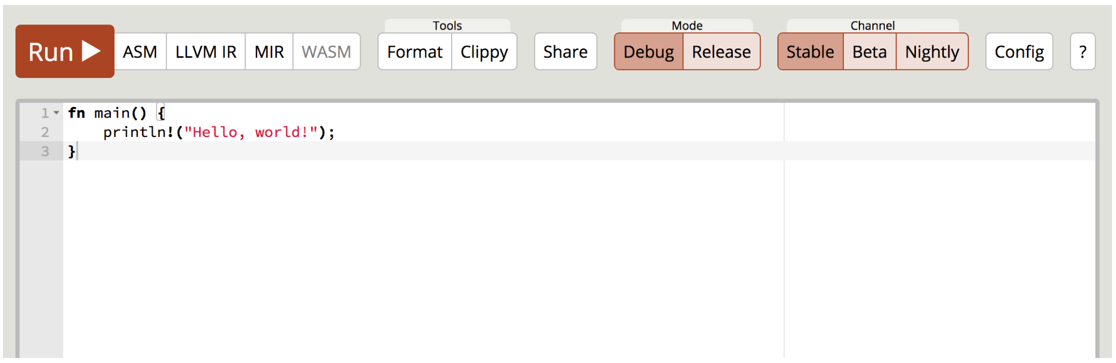
1. Error handling
2. Modules
3. FFI
4. More?

integer 32

Stack Overflow



Rust Playground



```
fn main() {
    println!("Hello, world!");
}
```

play.rust-lang.org

Jake Goulding

- Rust infrastructure team
- Working on a Rust video course for Manning
- A handful of crates
- Help out with AVR-Rust

Who are you?

Cargo

- Package manager
- Build tool
    - Code
    - Tests
    - Docs

Cargo

Cargo

Hello, world!

Printing values

Comments

API Documentation

- https://doc.rust-lang.org/
    - Click on "Standard Library API Reference"
-
    -

Functions

Functions

Variables

Variables

Variables are immutable by default

Types

- : unsigned 32-bit integer
- : signed 32-bit integer
- : floating point number
- and/or : more on these later
- : a boolean
- : a tuple

Type inference / explicit types

- Most of the time, you don't need to specify the type
- You can choose to if it helps you learn

Exercise

- Create a function
  -
  -
  -
- Print out the result of calling the function with

One answer

Another answer

Vectors

Iterating

Iterating

Iterators

Iterators

Iterators

Iterators

Iterators

Exercise

- Print out the values from 0 (inclusive) to 100 (exclusive)
- That are divisible by 3
- And divisible by 7

Exercise

- Print out the values from 0 (inclusive) to 100 (exclusive)
- That are divisible by 3
- And divisible by 7

- Instead of printing them out, try adding them up

One answer

Another answer

Structs

Enums

Exercise

- Create and structs
- Create a function that converts to
-

Exercise

- Create and structs
- Create a function that converts to
-

- Instead of a struct, do it with a single enum

One answer

Methods

Methods

Exercise

- Create a method that converts to
-

An answer

Strings

- Rust has two primary string types:
-
    - Owns the data
    - Can be extended or reduced
-
    - References existing data
    - Cannot change length

Strings

- Can convert from a to a via

- Can get a from a via

Exercise

- Create a function that prints a number
- Multiples of three print “Fizz” instead of the number
- Multiples of five print “Buzz” instead of the number
- Multiples of both three and five print “FizzBuzz”
- Call the function with the numbers from 1 to 100
- Change the function to return a string instead of printing

One answer

Reading user input

Parsing strings

Exercise

- Read user input of a temperature and convert it

Exercise

- Read user input of a temperature and convert it

- Ask if it's Celcius or Fahrenheit

Handling errors

- Rust does not have exceptions
- You can:
  - return an error
  - panic

Returning errors

- is an enum

- Can't currently be used in or in tests

Chained error returns

The operator is syntax sugar for returning an error or getting the success value.

Panicking

- Tears down the current thread
    - If it's the main thread, program exits
- Safe to do, in Rust terms
- When to panic:
    - Great for prototyping and "learning Rust" workshops
    - OK for an executable
    - Not good for a library
        - Unless there's an error from the library writer

Explicit panics

Implicit panics

- /
- /
- Indexing out of bounds ( )

Exercise

- Write a function that adds two values
- If either of the values are greater than , return an error
- If the sum is greater than , return an error
- Call the function and panic if it fails

Hints

- Write a helper function for the repeated logic and use
- Use as your returned error type and its value

One answer

Modules

Visibility

Modules in files

Modules in files

Exercise

Create a function which calls two others. The parent function should be called in

FFI

- Use C code from Rust
- There's a lot of battle-tested code out there

Target library

- Tracks a persons name and age
- Look at and

Scaffolding

Exercise

Add extern declarations for:

-
-
-
-

One answer

C strings

-
    - (capacity, length, data pointer)
    - Owns the data
-
    - (length, data pointer)
    - Borrows the data
-
    - (data pointer)
    - Owns borrows the data

Interoperating with C strings

-
  - counterpart to
  - : convert to
-
  - counterpart to
  - : convert to

The keyword

- When defining a function
- When calling unsafe functions
- When defining or implementing traits

functions

- The code cannot guarantee it is safe
- Often based on some choice of arguments
- Sometimes based on pre-existing state

blocks

- Calling this set of unsafe functions is always safe

Powers of the keyword

- Dereferencing a raw pointer
- Reading or writing a mutable static variable
- Calling an unsafe function
    - All foreign functions are unsafe
- Implementing an unsafe trait

Warning

- Not permitted to break any of Rust's guarantees
- Up to programmer to verify, not the compiler

Exercise

- Create a person via
- Print out result of
- Optional: clean up memory via

Hints

- Will use or
- Will use blocks

One answer

Exercise

Create a nicer Rust wrapper struct called .

Automatically freeing resources

- is a trait known to the compiler
- Called when a type goes out of scope

Exercise

- Convert the wrapper struct to use

Extra ideas

- Traits
- Generics