

London Software
Craftsmanship Community

Introduction to Rust

Mattsi Jansky



Introduction

The Rust Programming Language

Rust is a compiled systems programming language focusing on performance, security and safety. It is:

- Multi-paradigm
 - Not strictly functional but has functional features, not strictly OO but has OO features.
- Not garbage collected, but it is memory-safe (mostly).
- Think of it as a competitor to C/C++, rather than a competitor to Java/C#.



Higher-level

- Control
- Performance,
- + Confidence
- + Safety



Lower-level

- + Control
- + Performance
- Confidence
- Safety

Introduction

Applications

Anything where performance and safety are key considerations, such as...

- Game engines
- Operating systems
- File systems
- Web browsers
- Simulations
- Embedded devices/firmware
- Drivers
- Safety-critical software



Installing Rust

<https://www.rust-lang.org/tools/install>



```
1 curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
2 source ~/.bashrc
```

See "[Other Installation Methods](#)" if you are on Windows.



- On Windows, download and run `rustup-init.exe`.



Installing Rust

<https://play.rust-lang.org/>

If you cannot install Rust for any reason, the Rust Playground linked above can help. Everything we're doing can be executed in one file via the Playground. In the case of Tic-Tac-Toe, just copy the contents of the `lib.rs` file into Playground.



Hello World!



```
1 cargo new hello_world  
2 cd hello_world
```



Hello World!



```
1 .  
2 |— Cargo.toml  
3 |— src  
4   |— main.rs  
5  
6 1 directory, 2 files
```



Hello World!

Cargo.toml

```
1 [package]
2 name = "hello-world"
3 version = "0.1.0"
4 edition = "2021"
5
6 # See more keys and their definitions at https://doc.rust-lang.org/cargo/reference/manifest.html
7
8 [dependencies]
```



Hello World!



Cargo.toml

```
1 fn main() {  
2     println!("Hello, world!");  
3 }
```



Hello World!



```
1 cargo run
```



FizzBuzz



```
1 for i in 1..11 {  
2   //Your code here  
3 }
```

Loop through the numbers 1-20

- If the number is divisible by 3, print “Fizz”
- If the number is divisible by 5, print “Buzz”
- If it is divisible by both, print “FizzBuzz”
- If none of the above is true, print nothing.



```
1 if n < 0 {  
2   print!("{}", is negative", n);  
3 } else if n > 0 {  
4   print!("{}", is positive", n);  
5 } else {  
6   print!("{}", is zero", n);  
7 }
```



```
1 let add = 1 + 1; //2  
2 let sub = 6 - 2; //4  
3 let div = 10 / 2; //5  
4 let mult = 5 * 2; //10  
5 let rem = 10 % 8; //2
```



Enums

- Algebraic datatypes, similar to F# or Haskell
- An *enum* has *variants*, each of which can store different variables
- Eg, Rust enums can implement the command pattern like so:

```
enum Message {  
    Quit,  
    Move { x: i32, y: i32 },  
    Write(String),  
    ChangeColor(i32, i32, i32),  
}
```



Pattern Matching

- Rust has some really cool pattern-matching
- A *match* statement must be exhaustive- if there are any cases not accounted for, you get a compile error.

```
let topping = "pineapple"

match food {
  "pineapple" | "strawberry" => println!("Does not belong on Pizza"),
  "Cheese" | "Tomatoes" | "Jalapenos" => println!("Does belong on pizza"),
  _ => "Unknowable"
}
```

```
let number_of_garlic_cloves = 30;

match number_of_garlic_cloves {
  1..=5 => "Appropriate",
  6..=10 => "Going a bit overboard",
  11..=15 => "Going a bit overboard",
  16..=20 => "STOP",
  _ => "WOAH nelly"
}
```

```
let first_letter_of_name = 'm';

match first_letter_of_name {
  'a'..'l' => println!("not bad"),
  'n'..'z' => println!("pretty cool"),
  'm' => println!("The coolest 🤩")
}
```



Ownership Model

- This is Rust's most unique feature, and really makes it stand out
- Ownership rules
 - Every value in memory has an *owner*. An owner is a variable.
 - There can only be one owner at one time.
 - When the owner goes out of scope, the value will be dropped.
- **For example...**



Traits

- Rust has no inheritance, but instead has *traits*
- Defines an interface that a struct can meet
- To meet that interface, declare a new *impl* block for that struct matching the trait

```
pub trait Summary {  
    fn summarize_author(&self) -> String;  
  
    fn summarize(&self) -> String {  
        format!("(Read more from {})...", self.summarize_author())  
    }  
}  
  
pub struct Tweet {  
    pub username: String,  
    pub content: String,  
    pub reply: bool,  
    pub retweet: bool,  
}  
  
impl Summary for Tweet {  
    fn summarize_author(&self) -> String {  
        format!("@{}", self.username)  
    }  
}
```

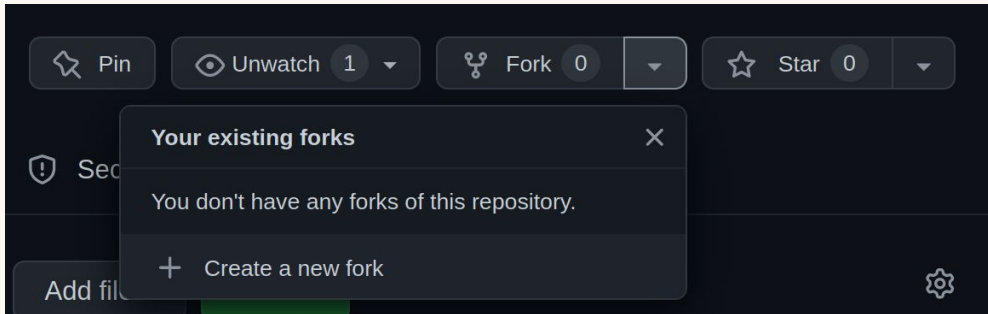


Let's write Tic-Tac-Toe (Norts and Crosses)

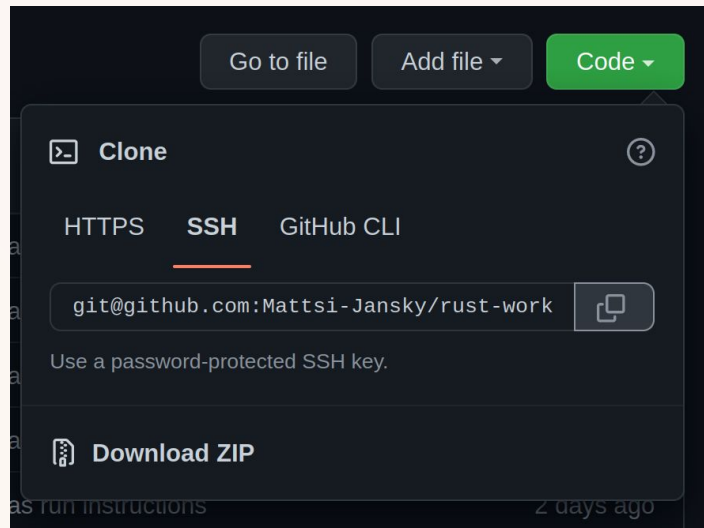


<https://github.com/Mattsi-Jansky/rust-workshop>

**Have a GitHub account?
Fork the repo, and clone
your fork.**



**Otherwise, clone the repo
with git or download a zip.**



Conclusion

The Future is
Rust. Maybe?

- Rust has been the #1 most loved programming language in the StackOverflow Developer Survey every year since 2016
 - Actual adoption from companies is much lower, however
- Rust now has its own foundation, independent from Mozilla
- An ongoing project supported by Google is trying to add support for Rust to the Linux Kernel project- the only one other than C.
- Ed Snowden endorsement



Disadvantages

- Young community
 - Has not built up the extensive libraries of software that Java et al have
 - Many fewer developers in the job market than other more established communities
- More complex than other languages
 - Faster and easier to train new developers in simpler languages



Resources for Studying

- Start with **The Rust Book**
 - Available free online, or paperback as “The Rust Programming Language”
 - Written by Rust language designers, frequently updated, well written
- Use **Exercism** to get practice and mentoring
- Start a study group
 - See blog post “Want to learn a new skill? Start a study group, here's how”
- After The Rust Book, try **Rust for Rustaceans**
- YouTube - **Let's Get Rusty** (beginner), **Crust of Rust** (intermediate)
- Discord Rust groups







Rust London

<https://www.meetup.com/rust-london-user-group/>



Fin

Thank You for listening, LSCC!

 www.matt.si
 [@mattsijansky](https://twitter.com/mattsijansky)
 [mattsi-jansky](https://github.com/mattsi-jansky)
 [mjjansky](https://www.linkedin.com/in/mjjansky)

