

Rust Community Stuttgart

## Workshop

2019-05-15

# Ownership, Borrowing, and Lifetimes

By Romeo Disca





## RULES — OWNERSHIP & BORROWING

03.05.19

A reference has a  
copy trait.

A mutable reference  
doesn't have a copy trait

Stack:

Arrows (refs) always go up. ↑✓  
Never down. ✗⚡

Heap:

Data on the heap  
never gets relocated.

① All assignments are moves!

② Return values are always moved.  
temporary value / rhs value:

③ All assignments are first moved to the  
temporary memory. Then moved back to the stack  
stack:

④ Stack values are always dropped in  
reverse order.

The dot operator:

- ▷ no ref → no magic → method call
- ▷ ref → dereferences "\*" until no ref type found

## RULES - LIFETIMES

03.05.19

#1: fn f<sub>1 2</sub><sup>1 2</sup> <'a, 'b> (x: &'a i32, y: &'b i32) {}

#2: fn  $f \langle 'a \rangle (x: \&'a i32) \rightarrow \&'a i32 \{ \}$

(#3): fn f <'w> (&'use(f), x:&'i32, y:&'i32) → & T { ... }

type in inference  
is based on the  
function signature.

All functions implement Fn, FnMut, FnOnce, Copy, Clone, Send, and Sync.

function item type.

$$\text{let } f: fn() \rightarrow () = fn() \{ \}$$

An f variable.

## function pointer type

 $f_n \quad f() \quad \{ \}$ 

An  $\mathbb{F}$  declaration.

closure trait

$$f_n \leq F: F_n \text{Once}() \rightarrow () \times (g: F) \{$$
$$f(\underbrace{\{1, 2\}}_A)$$

A closure.

## RULES - LIFETIMES OF STRUCTS

03.05.19

struct A <'a, T> {  
 x &'a : T,  
}

older as

"x must be older as A"

impl <'a, T> A <'a, T> {  
 fn xxx(&self) → &'a T {  
 self.x  
 }  
}

younger as

"output values are always  
younger as struct A"