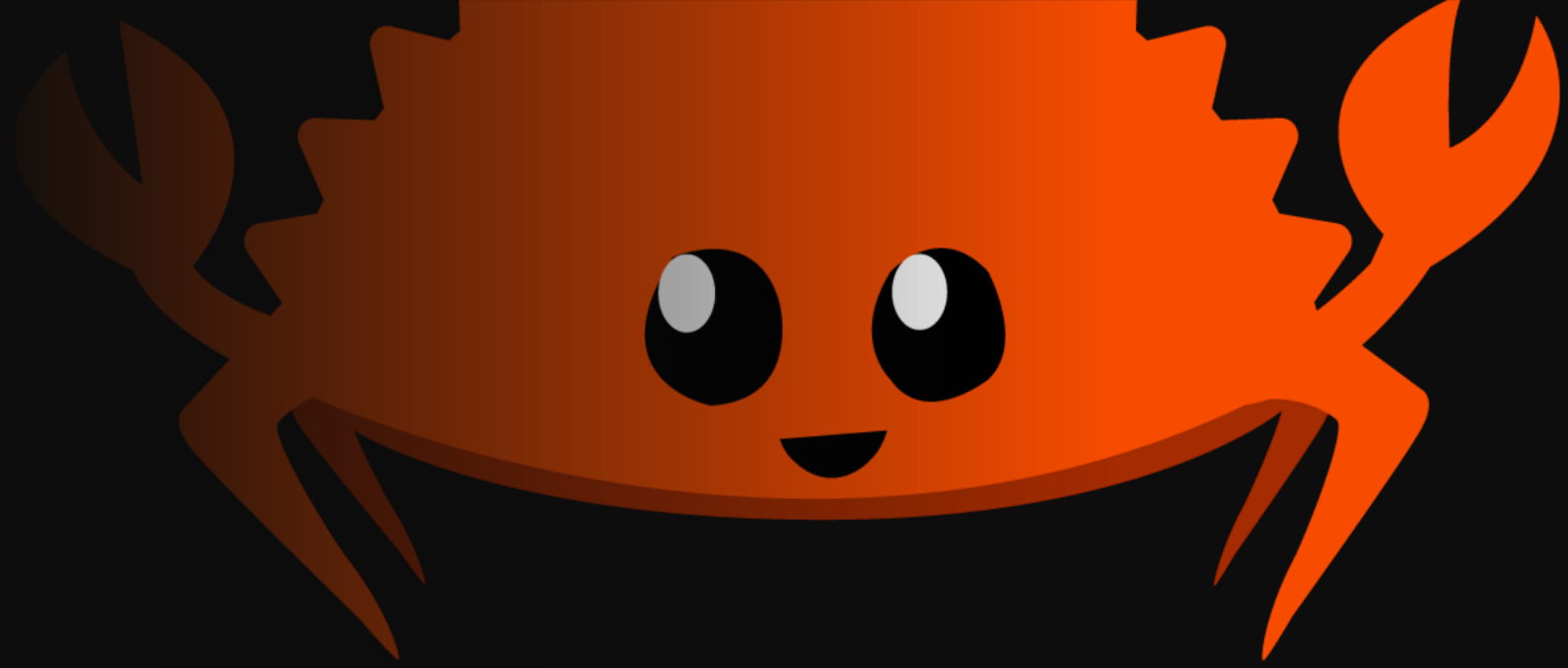


Rust

Vortragender: Finn Rodenberg

Seminar: Effiziente Programmierung

Wintersemester 2020/21

insights.stackoverflow.com/survey/2020

[5]

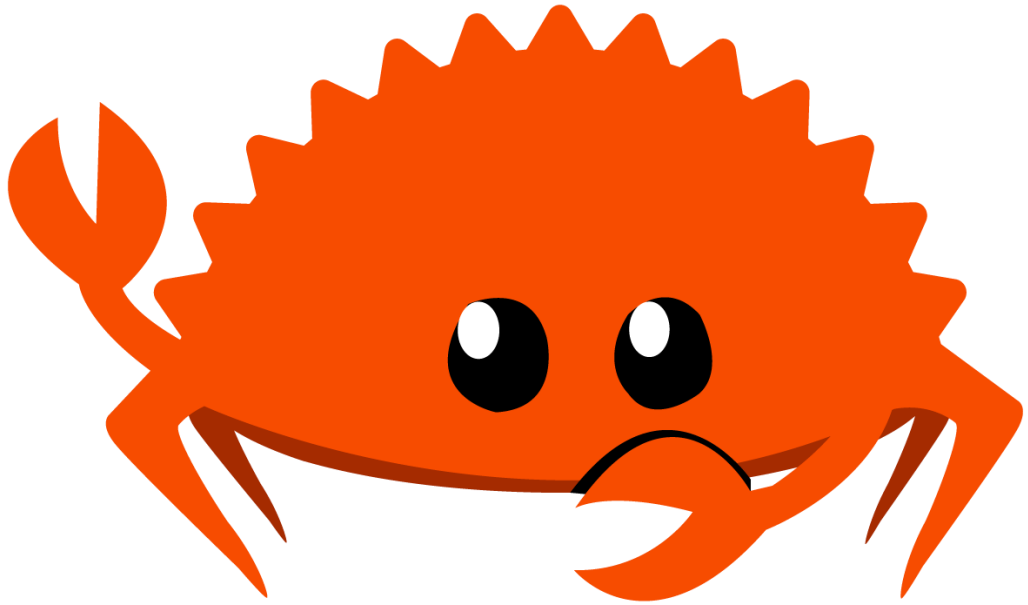
Rust Anwender

- mozilla.org
- atlassian.com
- dropbox.com
- threema.ch
- … [6]

insights.stackoverflow.com/survey/2020

Inhalt

- Hintergrund
- Installation
  - Rustup
  - Cargo
- Funktionsweise
  - Ownership Modell
  - Shared borrows
  - Mutable borrows
  - Mutability XOR Aliasing
  - Unsafe Rust
- Konklusion



rustacean.net

Hintergrund

& die Geschichte von Rust

Hintergrund

- 2006 Start als persönliches Projekt
- 2009 Mozilla Research wird Sponsor
- 2015 erster stable Release
- Aug. 2020 Entlassungen bei Mozilla Research
- Nov. 2020 AWS stellt Rust Entwickler ein

Spider-Man (2002). USA: Sony Pictures

„Hack without fear!“

- Niko Matsakis (C++ Now 2017) über Rust

Rust Versprechungen

- Ambitionierte Sicherheitsgarantien
- …ohne Performancekompromisse

Installation

rustup (+ Toolchains)

rustup

```
C:\Users\finn\Documents\Uni\EP Seminar\Rust>rustup
rustup 1.23.1 (3df2264a9 2020-11-30)
The Rust toolchain installer

USAGE:
    rustup [FLAGS] [+toolchain] <SUBCOMMAND>

FLAGS:
    -v, --verbose    Enable verbose output
    -q, --quiet      Disable progress output
    -h, --help       Prints help information
    -V, --version    Prints version information

ARGS:
    <+toolchain>    release channel (e.g. +stable) or custom toolchain to set override

SUBCOMMANDS:
    show           Show the active and installed toolchains or profiles
    update         Update Rust toolchains and rustup
    check          Check for updates to Rust toolchains
    default        Set the default toolchain
    toolchain      Modify or query the installed toolchains
    target         Modify a toolchain's supported targets
    component      Modify a toolchain's installed components
    override       Modify directory toolchain overrides
    run            Run a command with an environment configured for a given toolchain
    which          Display which binary will be run for a given command
    doc            Open the documentation for the current toolchain
```

Installation

Cargo

Rust Funktionsweise

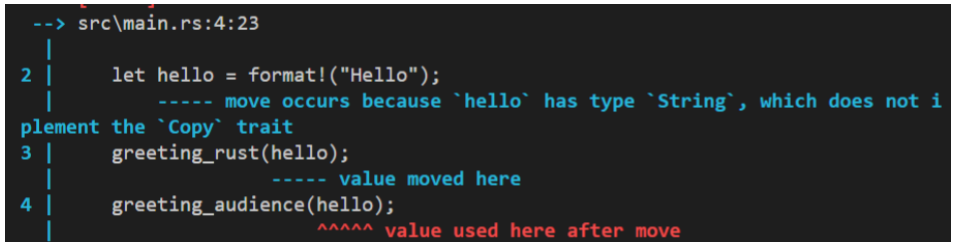
Ownership

Ownership Einstieg: Java Beispiel

```java
public class MyClass {
    public static void main(String args[]) {
      String hello = "Hello";
      greetingJava(hello);
      greetingAudience(hello);
    }

    static void greetingJava(String greet) {
        System.out.println(greet + " Java!");
    }

    static void greetingAudience(String greet) {
        System.out.println(greet + " everyone!");
    }
}
```

```
Hello Java!
Hello everyone!
```

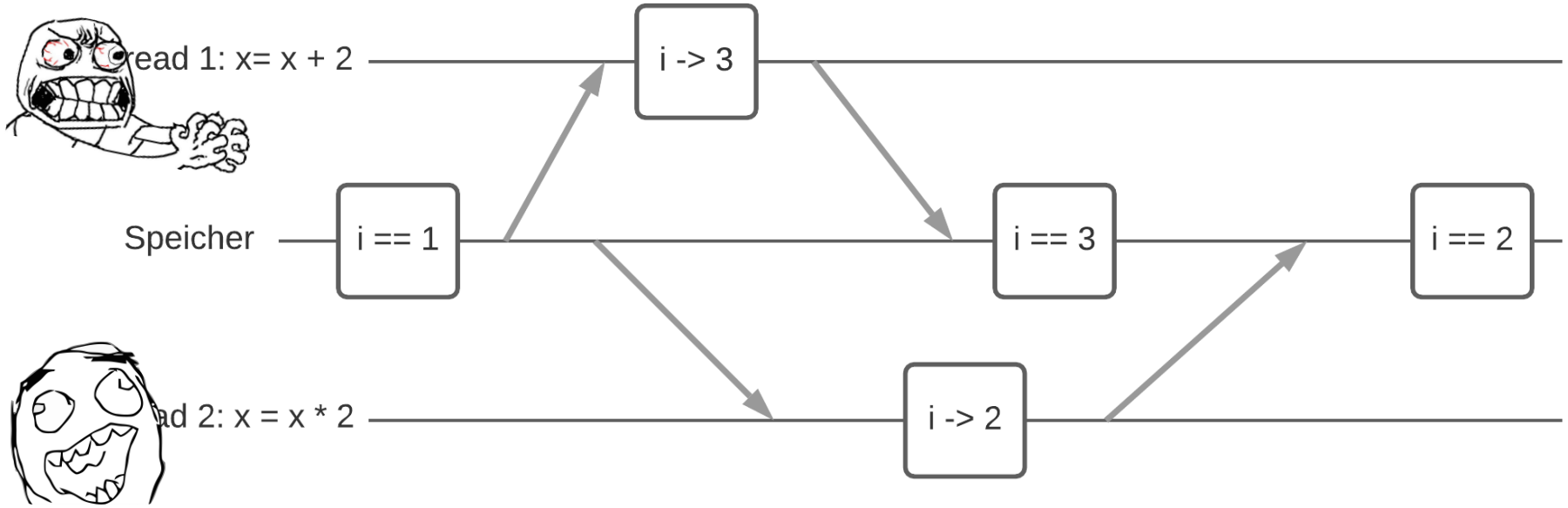
```rust
fn main() {
    let hello = String::from("Hello");
    greeting_rust(hello);
    greeting_audience(hello);
}

fn greeting_rust(greet: String) {
    println!("{0} Rust!", greet);
}

fn greeting_audience(greet: String) {
    println!("{0} everyone!", greet);
}
```



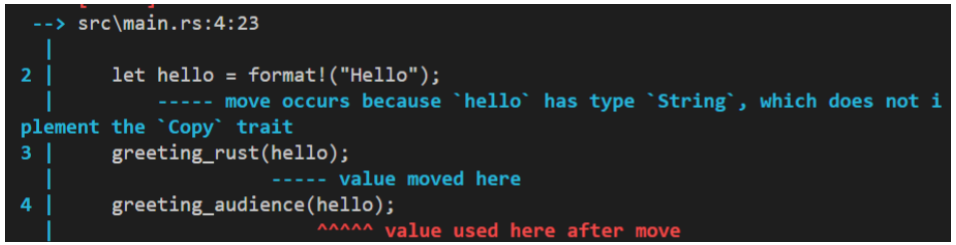
```
error[E0382]: use of moved value: `hello`
 --> src\main.rs:4:23
  |
2 |     let hello = format!("Hello");
  |         ----- move occurs because `hello` has type `String`, which does not i
mplement the `Copy` trait
3 |     greeting_rust(hello);
  |                   ----- value moved here
4 |     greeting_audience(hello);
  |                       ^^^^^ value used here after move
```

```java
public class MyClass {
    public static void main(String args[]) {
        SomeImportantObj essential = ...;
        doOperations(essential);
        displayInformation(essential);
    }

    static void doOperations(SomeImportantObj mainThing) {
        Thread t = new Thread(...);
        t.start();
    }

    static void displayInformation(SomeImportantObj mainThing) {
        String msg = "We handle the essential information:\n";
        System.out.println(msg + essential.display());
    }
}
```

Einschub: Data Races



```rust
fn main() {
    let hello = String::from("Hello");
    greeting_rust(hello);
    greeting_audience(hello);
}

fn greeting_rust(greet: String) {
    println!("{0} Rust!", greet);
}

fn greeting_audience(greet: String) {
    println!("{0} everyone!", greet);
}
```



```rust
fn main() {
    let hello = String::from("Hello");
    greeting_rust(hello.clone());
    greeting_audience(hello);
}

fn greeting_rust(greet: String) {
    println!("{0} Rust!", greet);
}

fn greeting_audience(greet: String) {
    println!("{0} everyone!", greet);
}
```

```
Hello Rust!
Hello everyone!
```

```rust
fn main() {
    let number = 1;
    count_first(number);
    count_second(number);
}

fn count_first(num: i8){
    println!("Look at me counting! {0}", num);
}

fn count_second(num: i8){
    println!("I am amazing at this! {0}", num + 1);
}
```

```
Look at me counting! 1
I am amazing at this! 2
```

Typen mit autocopy [2]

- Bool
- Char
- Signed/unsigned Integer (i8, i16, .., u8, …)
- Float (f32, f64)
- Weitere

Rust Funktionsweise

Shared Borrows

```rust
fn main() {
    let language = format!("Rust");
    let l = &language;
    mission_one(l);
    mission_two(l);
}

fn mission_one(lang: &String) {
    println!("Learn {0}'s Ownership.", lang);
}

fn mission_two(lang: &String) {
    println!("Become a fellow {0}acean!", lang);
}
```

```
Learn Rust's Ownership.
Become a fellow Rustacean!
```

Rust Funktionsweise

Werte bearbeiten

Werte bearbeiten: Rust Beispiel

```rust
fn main() {
    let counter = 1;

    while counter <= 3 {
        println!("{0} sheep", counter);
        counter += 1;
    }
}
```

```
error[E0384]: cannot assign twice to immutable variable `counter`
 --> src\main.rs:6:9
  |
2 |     let counter = 1;
  |         -------
  |         |
  |         first assignment to `counter`
  |         help: make this binding mutable: `mut counter`
...
6 |         counter += 1;
  |         ^^^^^^^^^^^^ cannot assign twice to immutable variable
```

Werte bearbeiten: Rust Fehlermeldung

```
error[E0384]: cannot assign twice to immutable variable `counter`
 --> src\main.rs:6:9
  |
2 |     let counter = 1;
  |         -------
  |         |
  |         first assignment to `counter`
  |         help: make this binding mutable: `mut counter`
...
6 |         counter += 1;
  |         ^^^^^^^^^^^^ cannot assign twice to immutable variable
```

```rust
fn main() {
    let mut counter = 1;

    while counter <= 3 {
        println!("{0} sheep", counter);
        counter += 1;
    }
}
```

```
1 sheep
2 sheep
3 sheep
```

Rust Funktionsweise

Mutable borrows

Mutable Borrows: Rust Beispiel

```rust
fn main() {
    let mut x = 10;
    println!("x: {0}", x);
    let xm = &mut x;
    helper(xm);
    println!("x: {0}", x);

}

fn helper(x: &mut i32){
    *x += 1;
}
```

```
x: 10
x: 11
```

Referenzen für Werte

- & referencing
- * dereferencing

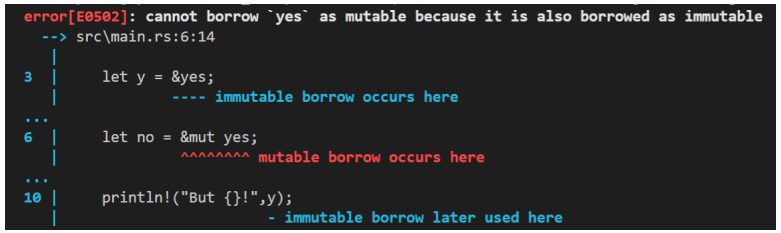
```
1 fn main() {
2     let mut x = 10;
3     let xm = &mut x;
4     *xm += 1;
5 }
```

Rust Funktionsweise

Mutability XOR Aliasing

Mutability XOR Aliasing: Rust Beispiel

```rust
fn main() {
    let mut yes = String::from("Yes");
    let y = &yes;
    println!("{}",yes);

    let no = &mut yes;
    *no = String::from("No");
    println!("{}", no);

    println!("But {}!",y);
}
```



Mutability XOR Aliasing: Rust Fehlermeldung

```
error[E0502]: cannot borrow `yes` as mutable because it is also borrowed as immutable
  --> src\main.rs:6:14
   |
3  |     let y = &yes;
   |             ---- immutable borrow occurs here
...
6  |     let no = &mut yes;
   |              ^^^^^^^^ mutable borrow occurs here
...
10 |     println!("But {}!",y);
   |                         - immutable borrow later used here
```

Rust Funktionsweise

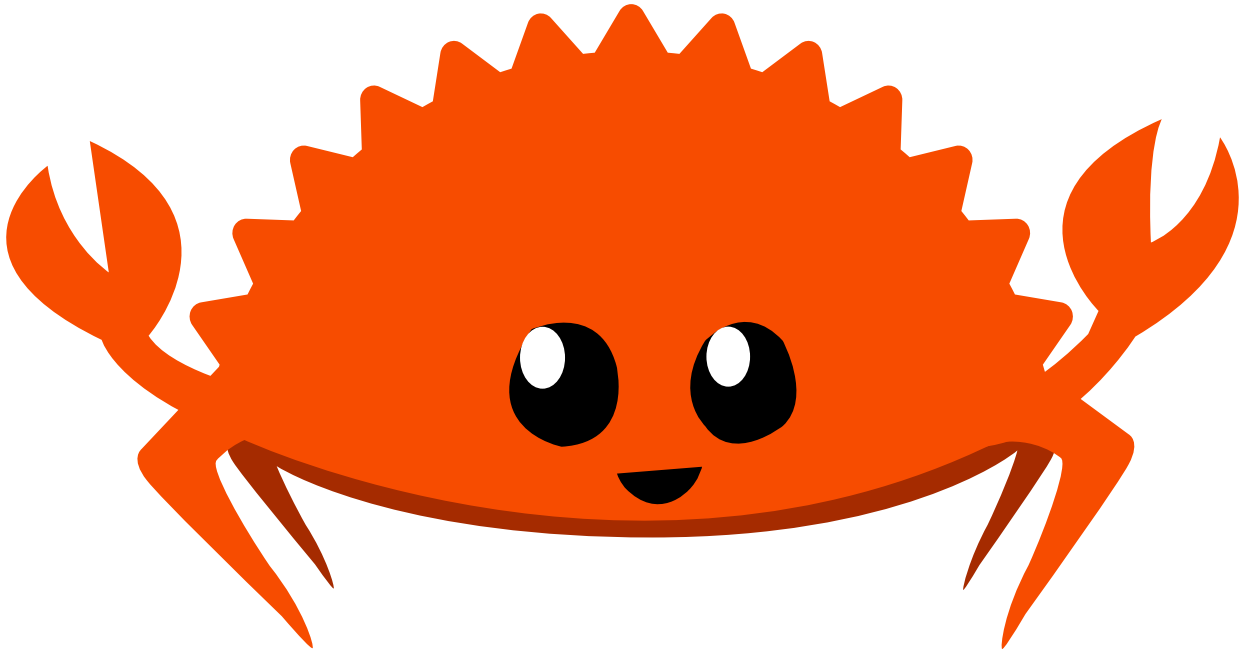
Unsafe Rust

Konklusion

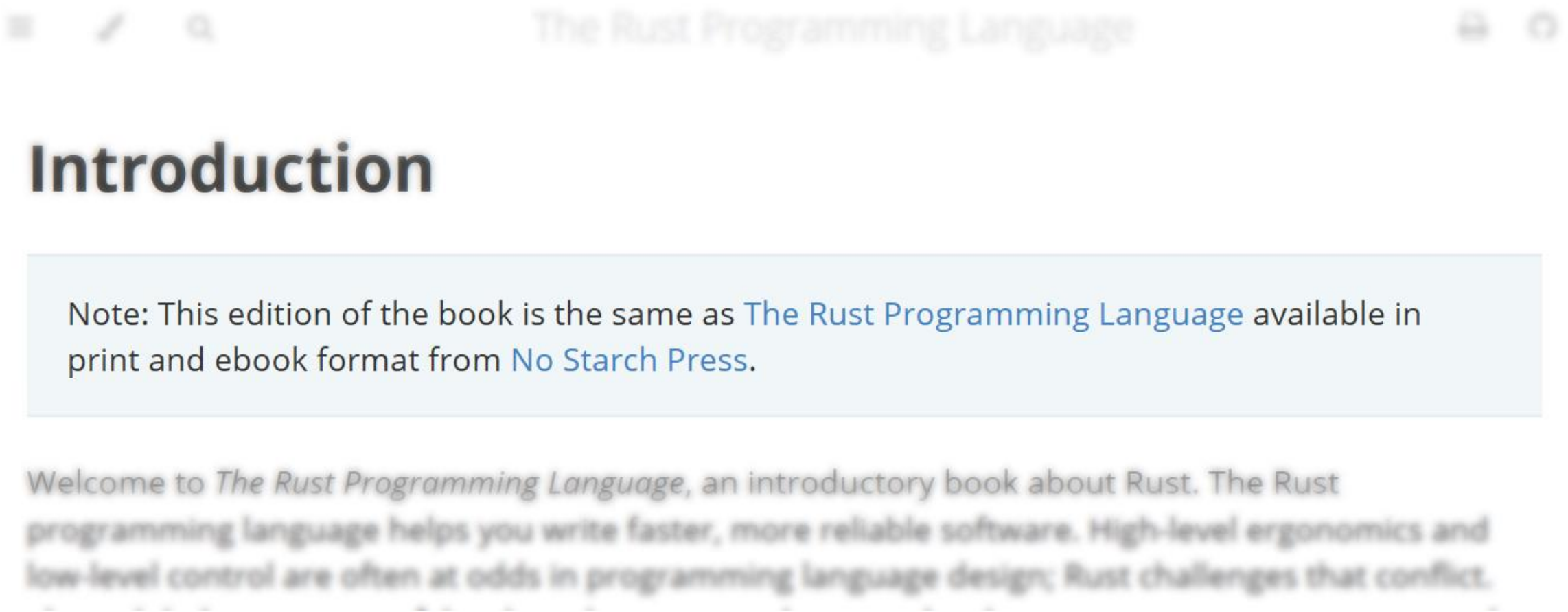
Was wissen wir nun?

Konklusion zu Rust

- Ownership Modell
  - Mutability XOR Aliasing
- Rustup, Cargo



rustacean.net


The Rust Programming Language
Introduction
Note: This edition of the book is the same as The Rust Programming Language available in print and ebook format from No Starch Press.
Welcome to The Rust Programming Language, an introductory book about Rust. The Rust programming language helps you write faster, more reliable software. High-level ergonomics and low-level control are often at odds in programming language design; Rust challenges that conflict.

[1]

Literaturverzeichnis

- [1] https://doc.rust-lang.org/book/ (18.01.21)
- [2] https://rust-lang.org/ (01.02.21)
- [3] https://doc.rust-lang.org/std/marker/trait.Copy.html (20.01.21)
- [4] https://www.heise.de/developer/artikel/Boesartige-Race-Conditions-und-Data-Races-3721793.html (25.01.21)
- [5] https://insights.dice.com/2019/11/11/10-github-programming-languages/ (10.12.20)
- [6] https://www.rust-lang.org/production/users