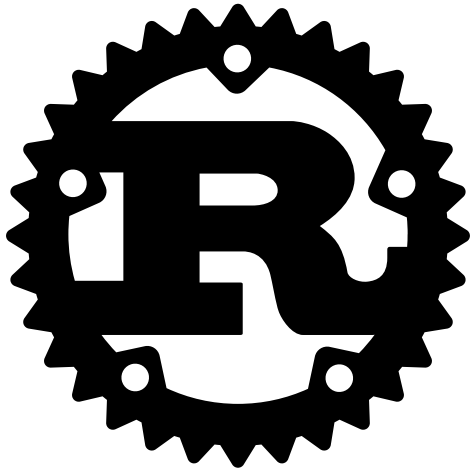

Rust in the Web

Eric Kunze

Inhalt

Entstehungsgeschichte

- 2006 persönliches Projekt von Graydon Hoare
- ab 2009 Projekt bei Mozilla
- 15. Mai 2015 Veröffentlichung der Version 1.0

Entstehungsgeschichte

- 2006 persönliches Projekt von Graydon Hoare
- ab 2009 Projekt bei Mozilla
- 15. Mai 2015 Veröffentlichung der Version 1.0

- Entwicklung einer neuen Browserenging $\rightarrow$ Servo

Entstehungsgeschichte

- 2006 persönliches Projekt von Graydon Hoare
- ab 2009 Projekt bei Mozilla
- 15. Mai 2015 Veröffentlichung der Version 1.0

- Entwicklung einer neuen Browserenging $\rightarrow$ Servo
  - Warum nicht C++ oder Java?

Ziele von Rust

Rust: Safe System Programming

Ziele von Rust

Rust: Safe System Programming

- Performance

Ziele von Rust

Rust: Safe System Programming

- Performance
- Kontrolle
  - minmale Runtime
  - keine Garbage Collection

Ziele von Rust

Rust: Safe System Programming

- Performance
- Kontrolle
  - minmale Runtime
  - keine Garbage Collection
- Sicherheit

Ziele von Rust

Rust: Safe System Programming

- Performance
- Kontrolle
  - minmale Runtime
  - keine Garbage Collection
- Sicherheit
- Features von höheren und funktionalen Programmiersprachen
  - Pattern Matching
  - Traits
  - Closures

Ownership and Borrowing

Ownership and Borrowing

- Ownership
  - jede Ressource hat zu einem Zeitpunkt **genau einen** Besitzer
  - Ressourcen können den Besitzer wechseln

Ownership and Borrowing

- Ownership
  - jede Ressource hat zu einem Zeitpunkt **genau einen** Besitzer
  - Ressourcen können den Besitzer wechseln

```
1 fn foo() {
2   let mut y: Vec<i32>
3       = Vec::new();
4
5   y.push(4);
6   bar(y);
7   y.push(5); // Compiler Error
8 }
```

```
1 fn bar(x: Vec<i32>) {
2   ...
3 }
```

Ownership and Borrowing

- Borrowing
  - Ownership kann verliehen werden

Ownership and Borrowing

- Borrowing
  - Ownership kann verliehen werden

- shared borrow
  - die Referenz ist **unveränderlich**
  - die Ressource kann mehrfach verliehen werden

Ownership and Borrowing

- Borrowing
  - Ownership kann verliehen werden
- shared borrow
  - die Referenz ist **unveränderlich**
  - die Ressource kann mehrfach verliehen werden

```rust
fn foo() {
    let mut y: Vec<i32>
        = Vec::new();

    y.push(4);
    bar(&y);
    y.push(5); // Ok
}
```

```rust
fn bar(x: &Vec<i32>){
    ...
    b = a + x[0];   // Ok

    x.push(1); // Compiler Error

}
```

Ownership and Borrowing

- mutable borrow
  - zu einem Zeitpunkt darf **nur eine** Referenz existieren
  - der Besitzer dieser Referenz kann die Ressource **verändern**

Ownership and Borrowing

- mutable borrow
  - zu einem Zeitpunkt darf **nur eine** Referenz existieren
  - der Besitzer dieser Referenz kann die Ressource **verändern**

```
1  fn foo() {
2    let mut y
3        = vec![1,2,3,4,5];
4    let mut x: Vec<i32>
5        = Vec::new();
6
7    bar(&y, &mut x); // Ok
8    bar(&y, &mut y);
9        // Compiler Error
10 }
```

```
1  fn bar(y: &Vec<i32>,
2         x: &mut Vec<i32>){
3
4    for v in y{
5      x.push(*v);
6    }
7  }
```

Vor- und Nachteile von Rust

Vor- und Nachteile von Rust

Vorteile

- Sicherheit und Geschwindigkeit
- Tools
  - Cargo
  - rustdoc
- Open Source Projekte crates.io

Vor- und Nachteile von Rust

Vorteile

- Sicherheit und Geschwindigkeit
- Tools
  - Cargo
  - rustdoc
- Open Source Projekte crates.io

Nachteile

- sehr junge Sprache
- häufige Änderungen
- Dokumentation
- (Kompilierzeit)

Rust in the Web

- HTTP-Server
  - Hyper
  - tiny-http
- HTTP client
  - Hyper
  - curl-rust
- Database drivers
  - rust-postgres
  - rusqlite
  - redis-rs
- Frameworks
  - Iron
  - rustful
  - Nickel

Iron

- Webframework
- basiert auf Hyper
- hochgradig Nebenläufig
- Basisframework ist leicht erweiterbar
- bietet Infrastruktur, um das Framework an individuelle Bedürfnisse anzupassen

Demo