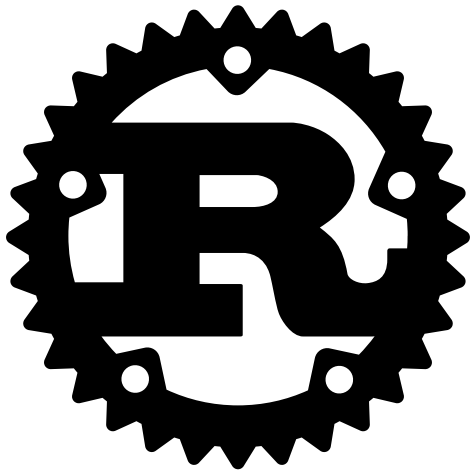

Rust in the Web

Eric Kunze

Rust - Historie

-
-
-
-

Warum Rust?

- Entwicklung einer neuen Browserenging → Servo
  - bessere Performance als alle anderen
  - legacy code
- Warum nicht C++ oder Java?
- Neue Sprache
  - schnell
  - sicher
  - Kontrolle über das System
  - Features von höheren- und funktionalen Programmiersprachen

→ Rust

Was macht Rust sicher?

- Ownership and Borrowing
- Ownership
  - jede Ressource hat genau einen Besitzer

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

```
fn foo(x: Vec<i32>) {
  ...
}
```

Was macht Rust sicher?

- Borrowing
  - jede Ressource kann verliehen werden
- shared borrow

```
fn foo() {
  let mut y: Vec<i32>
    = Vec::new();
  y.push(4);
  bar(&y);
  y.push(5);
}
```

```
fn bar(x: &Vec<i32>){
  println!("{}",x[0]);   // Ok
  x.push(1); // Compiler Error
}
```

Was macht Rust sicher?

- mutabl borrow
  - nur eine Referenz auf eine Ressource und diese hat nur einen Besitzer

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

```
fn bar(x: &Vec<i32>,
    y: &mut Vec<i32>){
  for v in x{
    y.push(*v);
  }
}
```

Was macht Rust sicher?

- Aliasing und Mutation zur gleichen Zeit wird verhindert
- Bsp. C++

```
void foo(){
  int *y = new int[10];
  for(int i=0;i<10;i++)
    y[i] = i;

  int *x = &y[9];
  y=bar(y);
  delete[] y;
  cout<<*x<<endl;
}
```

```
int* bar(int *v){
  delete[] v;
  v = new int[5]
  for(int i=0;i<10;i++){
    v[i] = i*2;
  }
  return v;
}
```