

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe



Rust

Florian “Florob” Zeitz

2015-10-28

Gliederung

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

- 1 Meta
- 2 Ownership
 - Einführung
 - Beispiel
- 3 Features
 - Algebraische Datentypen
 - Pattern Matching
 - Iterators
 - Unsafe

Gliederung

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

1 Meta

2 Ownership

- Einführung
- Beispiel

3 Features

- Algebraische Datentypen
- Pattern Matching
- Iterators
- Unsafe

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

- Systemprogrammiersprache
- kompiliert (LLVM backend)
- statisch typisiert
- stark typisiert
- affines Typsystem
- low-level, mit high-level Funktionalität
- große Community
- inspiriert von: C++, Erlang, Haskell, OCaml, Swift, ...

Geschichte

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

ab 2006 persönliches Projekt von Graydon Hoare
Compiler in OCaml

seit 2009 Entwicklung unterstützt von Mozilla, als Teil von Mozilla
Research

seit 2011 selbst compilierender Compiler

seit 2014 Sprachänderungen durch RFC Prozess

Mai 2015 Veröffentlichung von Rust 1.0

Philosophie

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

- Speichersicher
 - kein use-after-free
 - keine out-of-bounds Zugriffe
- keine data races
- minimale Laufzeitumgebung
- kein obligatorischer Garbage Collector
- keine versteckten Kosten
- immutable by default
- zero-cost abstractions

Gliederung

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

1 Meta

2 Ownership

- Einführung
- Beispiel

3 Features

- Algebraische Datentypen
- Pattern Matching
- Iterators
- Unsafe

Gliederung

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

1 Meta

2 Ownership

- Einführung
- Beispiel

3 Features

- Algebraische Datentypen
- Pattern Matching
- Iterators
- Unsafe

C++: Realloc

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1  #include <iostream>
2  #include <vector>
3  #include <string>
4
5  int main() {
6      std::vector<std::string> v;
7
8      v.emplace_back("Foo");
9      auto const &x = v[0];
10     v.emplace_back("Bar");
11     std::cout << x << '\n';
12
13     return 0;
14 }
```

C++: Realloc

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 $ clang++ -std=c++14 -Wall vector.cc -o vector-cc
2 $ ./vector-cc
3 zsh: segmentation fault (core dumped) ./vector-cc
```

C++: Iterator Invalidation

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1  #include <iostream>
2  #include <string>
3  #include <vector>
4
5  int main() {
6      std::vector<std::string> v = { "F", "o", "o" };
7
8      for (auto const &it : v) {
9          v.push_back(it + it);
10     }
11     for (auto const &it : v) {
12         std::cout << it << '\n';
13     }
14
15     return 0;
16 }
```

C++: Iterator Invalidation

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 $ clang++ -std=c++14 -Wall iter.cc -O -o iter-cc
2 $ ./iter-cc
3 F
4 O
5 O
6 FF
7
8
9 $
```

Beobachtung

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

Probleme entstehen bei der Kombination von:

Veränderlichkeit

+

Aliasing

```
v.emplace_back(...)  
v.push_back(...)
```

```
auto const &x = v  
auto const &it : v
```

Rusts Lösung

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

- Ownership: `fn f(x: Type) {...}`
 - ein Besitzer
 - veränderlich[†]
 - lesbar
 - kann ausgeliehen werden
- Shared borrow: `fn f(x: &Type) {...}`
 - beliebig teilbar (aliasing)
 - unveränderlich
 - lesbar
- Mutable borrow: `fn f(x: &mut Type) {...}`
 - nur eines zur Zeit
 - veränderlich
 - lesbar

Rusts Lösung

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

- Ownership: `fn f(x: Type) {...}`
 - ein Besitzer
 - veränderlich[†]
 - lesbar
 - kann ausgeliehen werden
- Shared borrow: `fn f(x: &Type) {...}`
 - beliebig teilbar (aliasing)
 - unveränderlich
 - lesbar
- Mutable borrow: `fn f(x: &mut Type) {...}`
 - nur eines zur Zeit
 - veränderlich
 - lesbar

Rusts Lösung

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

- Ownership: `fn f(x: Type) {...}`
 - ein Besitzer
 - veränderlich[†]
 - lesbar
 - kann ausgeliehen werden
- Shared borrow: `fn f(x: &Type) {...}`
 - beliebig teilbar (aliasing)
 - unveränderlich
 - lesbar
- Mutable borrow: `fn f(x: &mut Type) {...}`
 - nur eines zur Zeit
 - veränderlich
 - lesbar

Ownership

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 struct Crop;
2 struct Flour;
3
4 fn grind(_c: Crop) -> Flour {
5     Flour
6     // _c is freed here
7 }
8
9 fn main() {
10     let c = Crop;
11
12     grind(c); // c moves into `grind()`
13     // grind(c); // error: use of moved value: `c`
14 }
```

Shared Borrow

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 struct Book { page: u32 }
2
3 fn read(b: &Book) {
4     println!("I read page {}", b.page);
5 }
6
7 fn main() {
8     let b = Book { page: 1 };
9     let l = &b;
10
11     read(&b);
12     read(l);
13     read(&b);
14 }
```

Mutable Borrow

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
7 fn turn_page(b: &mut Book) { b.page += 1; }
8
9 fn main() {
10     let mut b = Book { page: 1 };
11
12     read(&b);
13     turn_page(&mut b);
14     read(&b);
15
16     let l = &b;
17     // turn_page(&mut b); // error: cannot borrow `b` as
18                           // mutable because it is also
19                           // borrowed as immutable
20     read(l);
21 }
```

Rust: Realloc

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 fn main() {  
2     let mut v: Vec<String> = Vec::new();  
3  
4     v.push("Foo".to_string());  
5     let x: &String = &v[0];  
6     v.push("Bar".to_string());  
7  
8     println!("{}", x);  
9 }
```

Rust: Realloc

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 $ rustc vector.rs
2 vector.rs:6:5: 6:6 error: cannot borrow `v` as mutable because it is
   ↳ also borrowed as immutable
3 vector.rs:6      v.push("Bar".to_string());
4                 ^
5 vector.rs:5:23: 5:24 note: previous borrow of `v` occurs here; the
   ↳ immutable borrow prevents subsequent moves or mutable borrows
   ↳ of `v` until the borrow ends
6 vector.rs:5      let x: &String = &v[0];
7                 ^
8 vector.rs:9:2: 9:2 note: previous borrow ends here
9 vector.rs:1 fn main() {
10 ...
11 vector.rs:9 }
12           ^
13 error: aborting due to previous error
```

Rust: Iterator Invalidation

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 fn main() {
2     let mut v: Vec<String> = vec!["F".into(), "o".into(),
3                                     "o".into()];
4
5     for it in &v {
6         v.push(it.clone() + &it);
7     }
8     for it in &v {
9         println!("{}", it);
10    }
11 }
```

Rust: Iterator Invalidation

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 $ rustc iter.rs
2 iter.rs:5:9: 5:10 error: cannot borrow `v` as mutable because it is
   ↳ also borrowed as immutable
3 iter.rs:5          v.push(it.clone() + &it);
4                   ^
5 iter.rs:4:16: 4:17 note: previous borrow of `v` occurs here; the
   ↳ immutable borrow prevents subsequent moves or mutable borrows
   ↳ of `v` until the borrow ends
6 iter.rs:4          for it in &v {
7                   ^
8 iter.rs:6:6: 6:6 note: previous borrow ends here
9 iter.rs:4          for it in &v {
10 iter.rs:5          v.push(it.clone() + &it);
11 iter.rs:6          }
12                   ^
13 error: aborting due to previous error
```

Gliederung

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

1 Meta

2 Ownership

- Einführung
- **Beispiel**

3 Features

- Algebraische Datentypen
- Pattern Matching
- Iterators
- Unsafe

Beispiel: Pi Berechnung

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

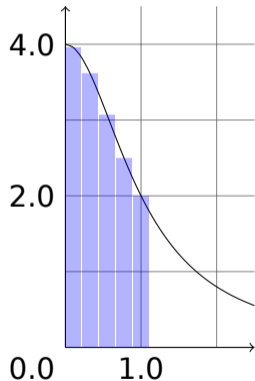
Patterns

Iterators

Unsafe

$$\begin{aligned}\pi &= 4 \arctan(1) \\ &= \int_0^1 \frac{4}{1+x^2} dx\end{aligned}$$

- Berechne π durch Riemann-Integration
- Näherung des Flächeninhalts durch schmale Rechtecke
- Gut parallelisierbar



C++ Lösung

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1  #include <cstdint>
2  #include <iostream>
3  #include <thread>
4  #include <vector>
5
6  int main() {
7      constexpr uint64_t NUM_THREADS = 4;
8      constexpr uint64_t NUM_STEPS = 100000;
9      constexpr uint64_t THREAD_STEPS = NUM_STEPS / NUM_THREADS;
10     constexpr double STEP = 1.0 / NUM_STEPS;
11
```

C++ Lösung

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
12  double pi = 0;
13
14  std::vector<std::thread> threads;
15
16  for (int i = 0; i < NUM_THREADS; ++i) {
17      uint64_t lower = THREAD_STEPS * i;
18      uint64_t upper = THREAD_STEPS * (i+1);
19      threads.emplace_back([=, &pi]() {
20          for (uint64_t j = lower; j < upper; ++j) {
21              double x = (j + 0.5) * STEP;
22              pi += 4.0/(1.0 + x*x) * STEP;
23          }
24      });
25 }
```

C++ Lösung

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
26
27     for (auto &t : threads) t.join();
28
29     std::cout.precision(10);
30     std::cout << "Pi = " << pi << '\n';
31
32     return 0;
33 }
```

C++ Lösung

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 $ clang++ -std=c++14 -lpthread -Wall pi.cc -o pi-cc
2 $ ./pi-cc
3 Pi = 1.156130797
4 $ ./pi-cc
5 Pi = 1.099799814
```

- Typisches data race
- Thread A liest $\text{pi} = 0.1423$
- Thread B liest $\text{pi} = 0.1423$
- Thread A schreibt $\text{pi} = 0.7609$
- Thread B schreibt $\text{pi} = 0.5768$
- $\text{pi} = 0.5768$, Thread As Berechnung ist verloren

Rust Lösung A

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 use std::thread;
2
3 fn main() {
4     const NUM_THREADS: u64 = 6;
5     const NUM_STEPS: u64 = 100_000;
6     const THREAD_STEPS: u64 = NUM_STEPS / NUM_THREADS;
7     const STEP: f64 = 1.0 / NUM_STEPS as f64;
8 }
```

Rust Lösung A

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
9     let mut pi: f64 = 0.0;
10
11     let guards: Vec<_> = (0..NUM_THREADS).map(|i| {
12         let lower: u64 = THREAD_STEPS * i;
13         let upper: u64 = THREAD_STEPS * (i+1);
14         let pi_ref: &mut f64 = &mut pi;
15         thread::spawn(move || {
16             for j in lower..upper {
17                 let x: f64 = (j as f64 + 0.5) * STEP;
18                 *pi_ref += 4.0 / (1.0 + x*x) * STEP;
19             }
20         })
21     }).collect();
22
```

Rust Lösung A

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
23     for g in guards { g.join().unwrap(); }  
24  
25     println!("Pi = {:.10}", pi);  
26 }
```


Rust Lösung A

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1  $ rustc -0 pi.rs
2  pi.rs:14:32: 14:39 error: cannot infer an appropriate lifetime for borrow expression due to
   ↪ conflicting requirements
3  pi.rs:14      let pi_ref: &mut f64 = &mut pi;
4                ^~~~~~
5  pi.rs:15:9: 20:11 note: first, the lifetime cannot outlive the call at 15:8...
6  pi.rs:15      thread::spawn(move || {
7  pi.rs:16          for j in lower..upper {
8  pi.rs:17              let x: f64 = (j as f64 + 0.5) * STEP;
9  pi.rs:18              *pi_ref += 4.0/(1.0 + x*x) * STEP;
10 pi.rs:19          }
11 pi.rs:20      })
12 pi.rs:15:28: 20:10 note: ...so that captured variable `pi_ref` does not outlive the enclosing closure
13 pi.rs:15      thread::spawn(move || {
14 pi.rs:16          for j in lower..upper {
15 pi.rs:17              let x: f64 = (j as f64 + 0.5) * STEP;
16 pi.rs:18              *pi_ref += 4.0/(1.0 + x*x) * STEP;
17 pi.rs:19          }
18 pi.rs:20      })
19 pi.rs:14:32: 14:39 note: but, the lifetime must be valid for the expression at 14:31...
20 pi.rs:14      let pi_ref: &mut f64 = &mut pi;
21                ^~~~~~
22 pi.rs:14:32: 14:39 note: ...so that reference is valid at the time of borrow
23 pi.rs:14      let pi_ref: &mut f64 = &mut pi;
24                ^~~~~~
25 error: aborting due to previous error
```

Rust Lösung B

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 use std::sync::Mutex;
2 use std::thread;
3
4 fn main() {
5     const NUM_THREADS: u64 = 6;
6     const NUM_STEPS: u64 = 100_000;
7     const THREAD_STEPS: u64 = NUM_STEPS / NUM_THREADS;
8     const STEP: f64 = 1.0 / NUM_STEPS as f64;
9 }
```

Rust Lösung B

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
10     let pi: Mutex<f64> = Mutex::new(0.0f64);
11
12     let guards: Vec<_> = (0..NUM_THREADS).map(|i| {
13         let lower: u64 = THREAD_STEPS * i;
14         let upper: u64 = THREAD_STEPS * (i+1);
15         let pi_ref = &pi;
16         thread::spawn(move || {
17             for j in lower..upper {
18                 let x: f64 = (j as f64 + 0.5) * STEP;
19                 let mut lock = pi_ref.lock().unwrap();
20                 *lock += 4.0/(1.0 + x*x) * STEP;
21             }
22         })
23     }).collect();
```

Rust Lösung B

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

24

```
for g in guards { g.join().unwrap(); }
```

25

26

27

```
println!("Pi = {:.10}", *pi.lock().unwrap());
```

28

```
}
```

Rust Lösung B

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 $ rustc pi_mutex.rs
2 pi_mutex.rs:12:47: 23:6 error: closure may outlive the current function, but it borrows `pi`, which is
   ↳ owned by the current function [E0373]
3 pi_mutex.rs:12      let guards: Vec<_> = (0..NUM_THREADS).map(|i| {
4 pi_mutex.rs:13      let lower: u64 = THREAD_STEPS * i;
5 pi_mutex.rs:14      let upper: u64 = THREAD_STEPS * (i+1);
6 pi_mutex.rs:15      let pi_ref = &pi;
7 pi_mutex.rs:16      thread::spawn(move || {
8 pi_mutex.rs:17      for j in lower..upper {
9
10 pi_mutex.rs:15:23: 15:25 note: `pi` is borrowed here
11 pi_mutex.rs:15      let pi_ref = &pi;
12                          ^~
13 pi_mutex.rs:12:47: 23:6 help: to force the closure to take ownership of `pi` (and any other referenced
   ↳ variables), use the `move` keyword, as shown:
14 pi_mutex.rs:      let guards: Vec<_> = (0..NUM_THREADS).map(move |i| {
15 pi_mutex.rs:      let lower: u64 = THREAD_STEPS * i;
16 pi_mutex.rs:      let upper: u64 = THREAD_STEPS * (i+1);
17 pi_mutex.rs:      let pi_ref = &pi;
18 pi_mutex.rs:      thread::spawn(move || {
19 pi_mutex.rs:      for j in lower..upper {
20
21 error: aborting due to previous error
```

Rust Lösung C

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 use std::sync::{Arc, Mutex};
2 use std::thread;
3
4 fn main() {
5     const NUM_THREADS: u64 = 6;
6     const NUM_STEPS: u64 = 100_000;
7     const THREAD_STEPS: u64 = NUM_STEPS / NUM_THREADS;
8     const STEP: f64 = 1.0 / NUM_STEPS as f64;
9
```

Rust Lösung C

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
10     let pi: Arc<Mutex<f64>> = Arc::new(Mutex::new(0.0f64));
11
12     let guards: Vec<_> = (0..NUM_THREADS).map(|i| {
13         let lower: u64 = THREAD_STEPS * i;
14         let upper: u64 = THREAD_STEPS * (i+1);
15         let pi_ref = pi.clone();
16         thread::spawn(move || {
17             for j in lower..upper {
18                 let x: f64 = (j as f64 + 0.5) * STEP;
19                 let mut lock = pi_ref.lock().unwrap();
20                 *lock += 4.0/(1.0 + x*x) * STEP;
21             }
22         })
23     }).collect();
```

Rust Lösung C

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

24

```
for g in guards { g.join().unwrap(); }
```

25

26

27

```
println!("Pi = {:.10}", *pi.lock().unwrap());
```

28

```
}
```


Rust Lösung C

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 $ rustc -O pi_arc.rs
2 $ ./pi_arc
3 Pi = 3.1415126520
```

Gliederung

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

1 Meta

2 Ownership

- Einführung
- Beispiel

3 Features

- Algebraische Datentypen
- Pattern Matching
- Iterators
- Unsafe

Gliederung

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

1 Meta

2 Ownership

- Einführung
- Beispiel

3 Features

- Algebraische Datentypen
- Pattern Matching
- Iterators
- Unsafe

Tuple

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 // u32 × &str
2 let x: (u32, &str) = (42, "Marvin");
3 println!("{}", x.0);
4 println!("{}", x.1);
```

- Struktureller Produkttyp
- Methoden implementiert für bis zu 12-stellige Tupel

Struct

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 // u32 × &str
2 struct Person {
3     age: u32,
4     name: &'static str
5 }
6 let x: Person = Person { age: 42, name: "Marvin" };
7 println!("{}", x.name, x.age);
```

- Produkttyp mit Feldnamen

Enum

Rust

Florob

Meta

Ownership

Einführung
Beispiel

Features

ADTs
Patterns
Iterators
Unsafe

```
1 // C-like
2 enum Dir {
3     North,
4     East,
5     South,
6     West
7 }
8 let d: Dir = Dir::East;
```

- Summentyp
- ähnlich zu tagged Unions

```
1 // with associated data
2 enum Shape {
3     Rect { x: f32, y: f32 },
4     Circ { r: f32 }
5 }
6 let c: Shape = Shape::Circ {
7     r: 23.0
8 };
```

Beispiel Option

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 enum Option<T> {  
2     None,  
3     Some(T)  
4 }
```

- äquivalent zu Haskell's Maybe Monade (return = Some, bind = and_then)
- statt NULL-Zeigern, nil-Objekten, etc.

Beispiel Option

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 let v: Vec<u32> = ...;  
2  
3 v.last().and_then(|l| if l < 3 { Some(l+4) } else { None })  
4     .and_then(|l| if l > 12 { Some(l-4) } else { None })  
5     .unwrap_or(42);
```


Gliederung

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

1 Meta

2 Ownership

- Einführung
- Beispiel

3 Features

- Algebraische Datentypen
- **Pattern Matching**
- Iterators
- Unsafe

match

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 let x = 2u32;  
2  
3 match x {  
4     1 => "One",  
5     2 | 3 => "Twree",  
6     5..9 => "Large small number",  
7     _ => "Fallthrough"  
8 }
```

match

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 let d = Dir::East;
2
3 match d {
4     Dir::North => println!("Northwards!")
5     Dir::East  => println!("Go Anti-West!")
6     Dir::South => println!("Southwards!")
7     Dir::West  => println!("Go West!")
8 }
```

match

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 let c = Shape::Circ { r: 1.0 };
2
3 match c {
4     Shape::Rect { x, y } => println!("{}", x, y),
5     Shape::Circ { r }   => println!("{}", r)
6 }
```

let

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 let Person { age, name } = marv;  
2 let (x, y) = point;  
3 let Person { age: alter, name: vorname } = marv;
```

if let

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 if let Ok(dir) = std::env::var("HOME") {  
2     println!("Home dir is {}", dir);  
3 }
```

Matching function parameters

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 fn print_person(&Person { age, name }: &Person) {  
2     println!("{}", name, age);  
3 }
```

Gliederung

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

1 Meta

2 Ownership

- Einführung
- Beispiel

3 Features

- Algebraische Datentypen
- Pattern Matching
- **Iterators**
- Unsafe

Iterators

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

- Objekte die eine `.next()` Methode besitzen
- **for**-Schleifen sind syntactic sugar für wiederholtes Aufrufen von `.next()` bis es `None` zurück gibt

Adaptors

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 fn main() {  
2     let mult = (1..).filter(|x| (1..11).all(|y| x % y == 0))  
3         .next().unwrap();  
4     println!("{}", mult);  
5 }
```

- Finde die kleinste Zahl die von jeder Zahl von 1 bis 10 geteilt wird

Gliederung

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

1 Meta

2 Ownership

- Einführung
- Beispiel

3 Features

- Algebraische Datentypen
- Pattern Matching
- Iterators
- **Unsafe**

Unsafe

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

```
1 pub fn u64_le_array(x: u64) -> [u8; 8] {  
2     unsafe { std::mem::transmute(x.to_le()) }  
3 }
```

- Nötig um einige sichere Abstraktionen zu erstellen
- ändert **nicht** die Sprachsemantik
- erlaubt das Aufrufen von als **unsafe** markierten Funktionen
- z. B. Pointerarithmetik, Dereferenzieren von raw Pointern

Rust

Florob

Meta

Ownership

Einführung

Beispiel

Features

ADTs

Patterns

Iterators

Unsafe

Danke für die Aufmerksamkeit.
Fragen?



<http://babelmonkeys.de/~florob/talks/RUG-2015-10-28-Rust.pdf>