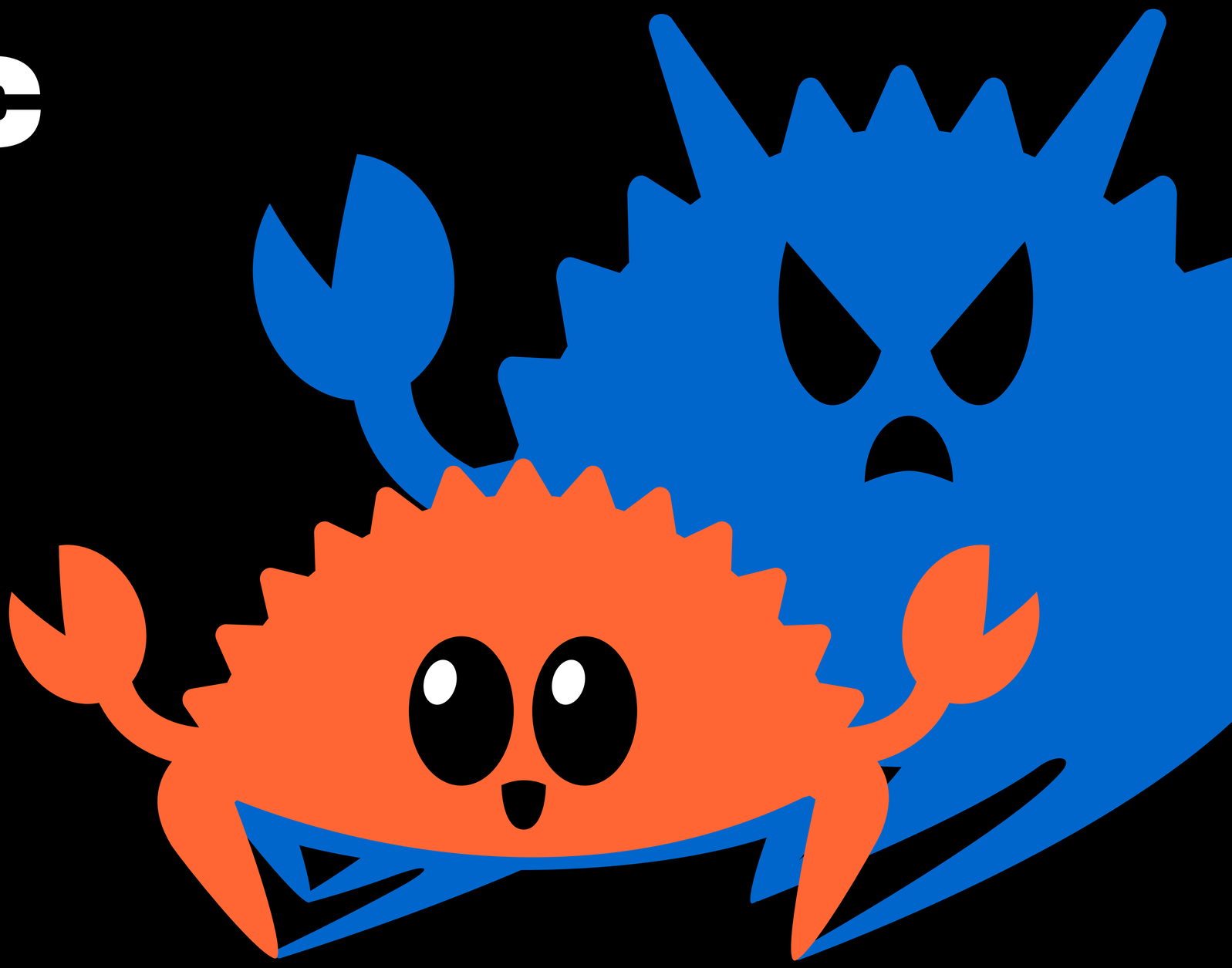
# Экскурс в Rust



@mgrachev
evrone

# Михаил Грачев

**Back-end**
**Team lead**
**DevOps**

**GitHub/Twitter**
**@mgrachev**

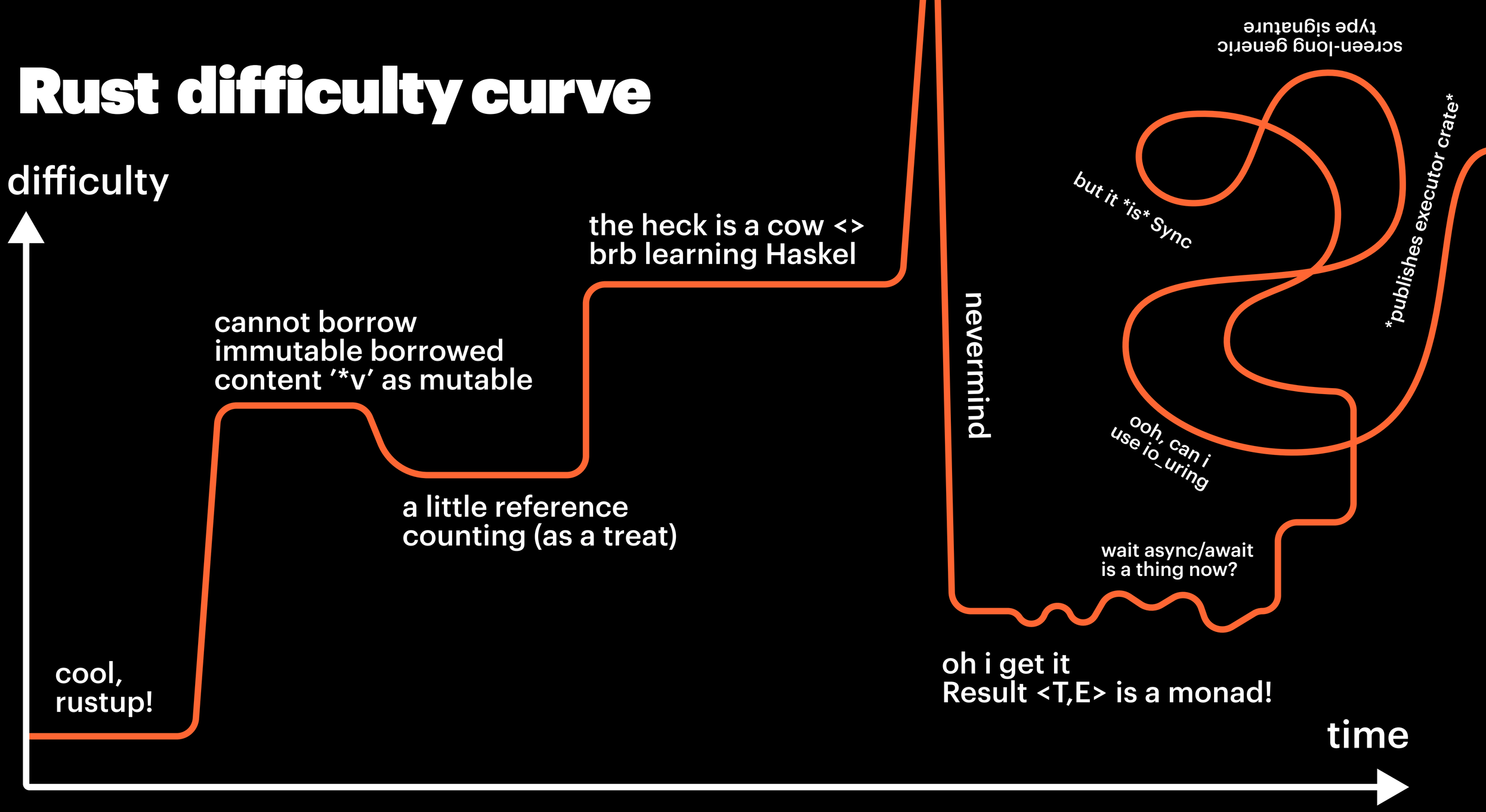
# Мы ❤ Open Source

@mgrachev
evrone

github.com/dotenv-linter

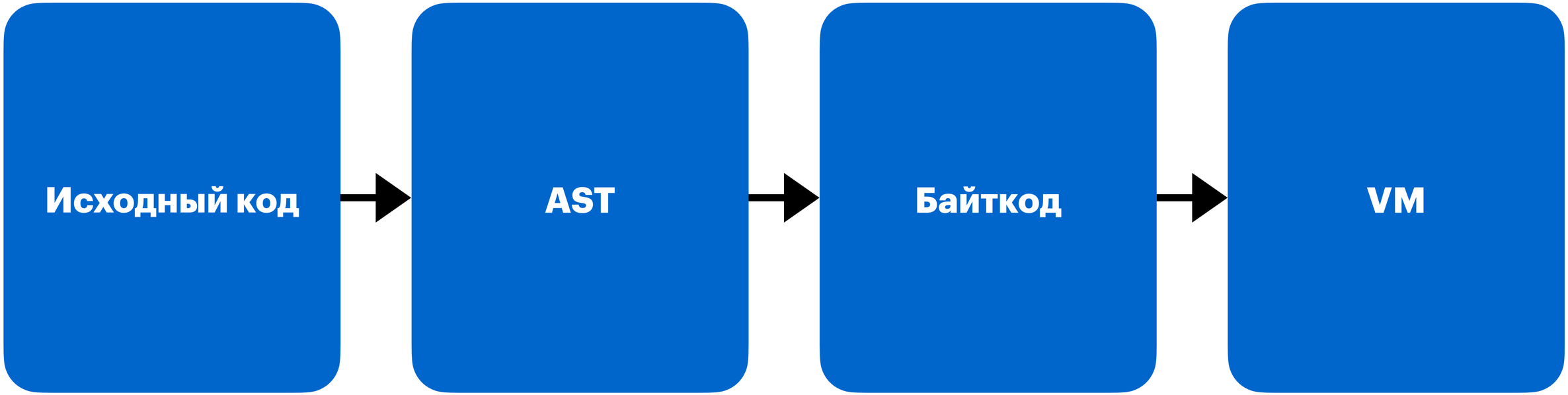
github.com/datanymizer

# Rust difficulty curve



# Работа с памятью: Ruby, Python

@mgrachev
evrone



# Работа с памятью: Ruby, Python

@mgrachev
evrone

ПЛЮС:

**VM/GC отвечают за работу с памятью**

# Работа с памятью: Ruby, Python

@mgrachev
evrone

ПЛЮС:

**VM/GC отвечают за работу с памятью**

МИНУС:

**Скорость работы**

# Работа с памятью: Rust

@mgrachev
evrone

# Работа с памятью: Rust

@mgrachev
evrone

Нет VM/GC

Исходный код → AST → Машинный код

# Работа с памятью: Rust

@mgrachev
evrone

Нет VM/GC

Нет ручного управления памятью (malloc/free)

Исходный код → AST → Машинный код

# Работа с памятью: Rust

@mgrachev
evrone

ПЛЮС:

**Скорость работы**

# Работа с памятью: Rust

@mgrachev
evrone

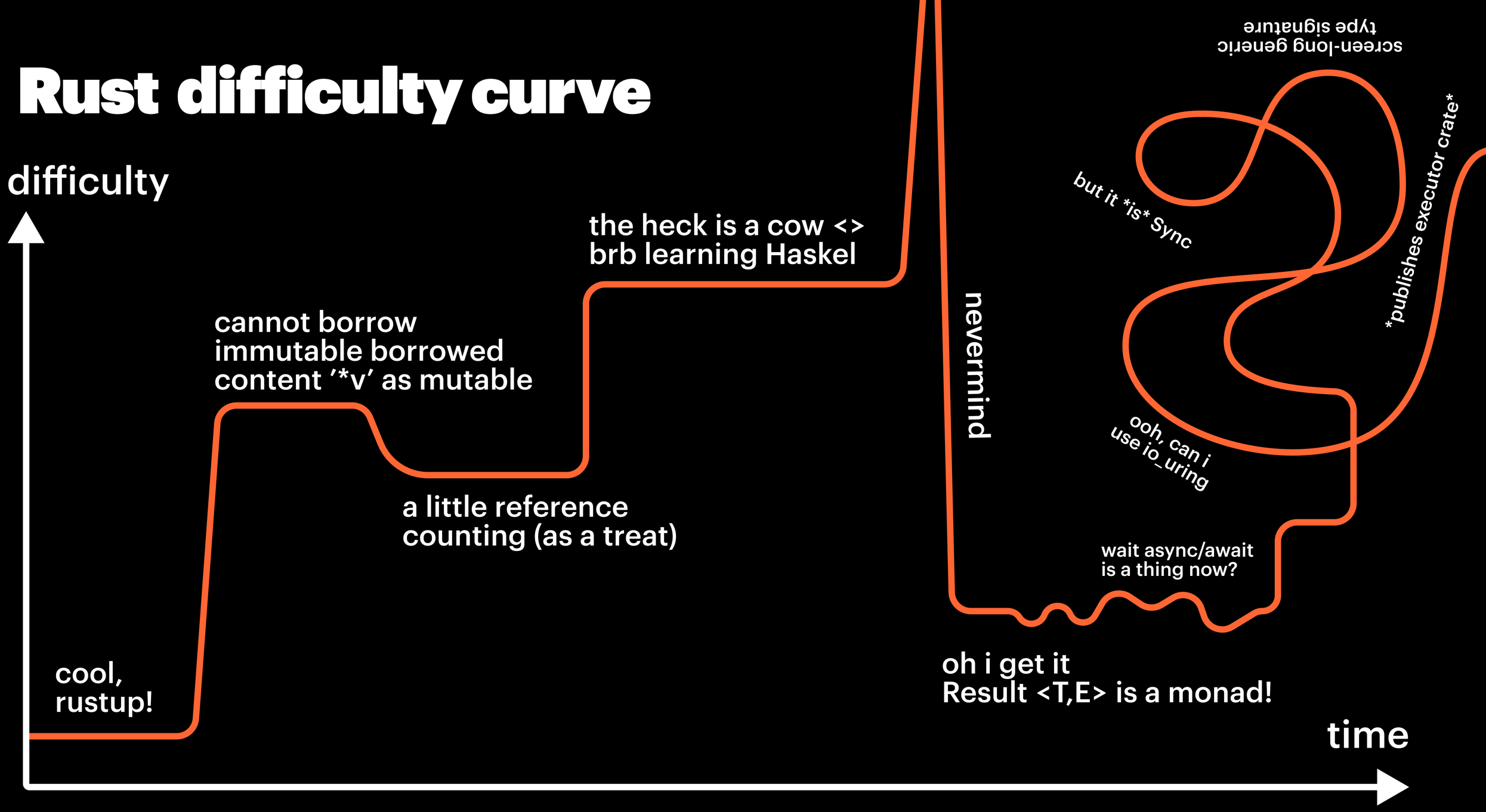
ПЛЮС:

**Скорость работы**

МИНУС:

**?**

# Rust difficulty curve

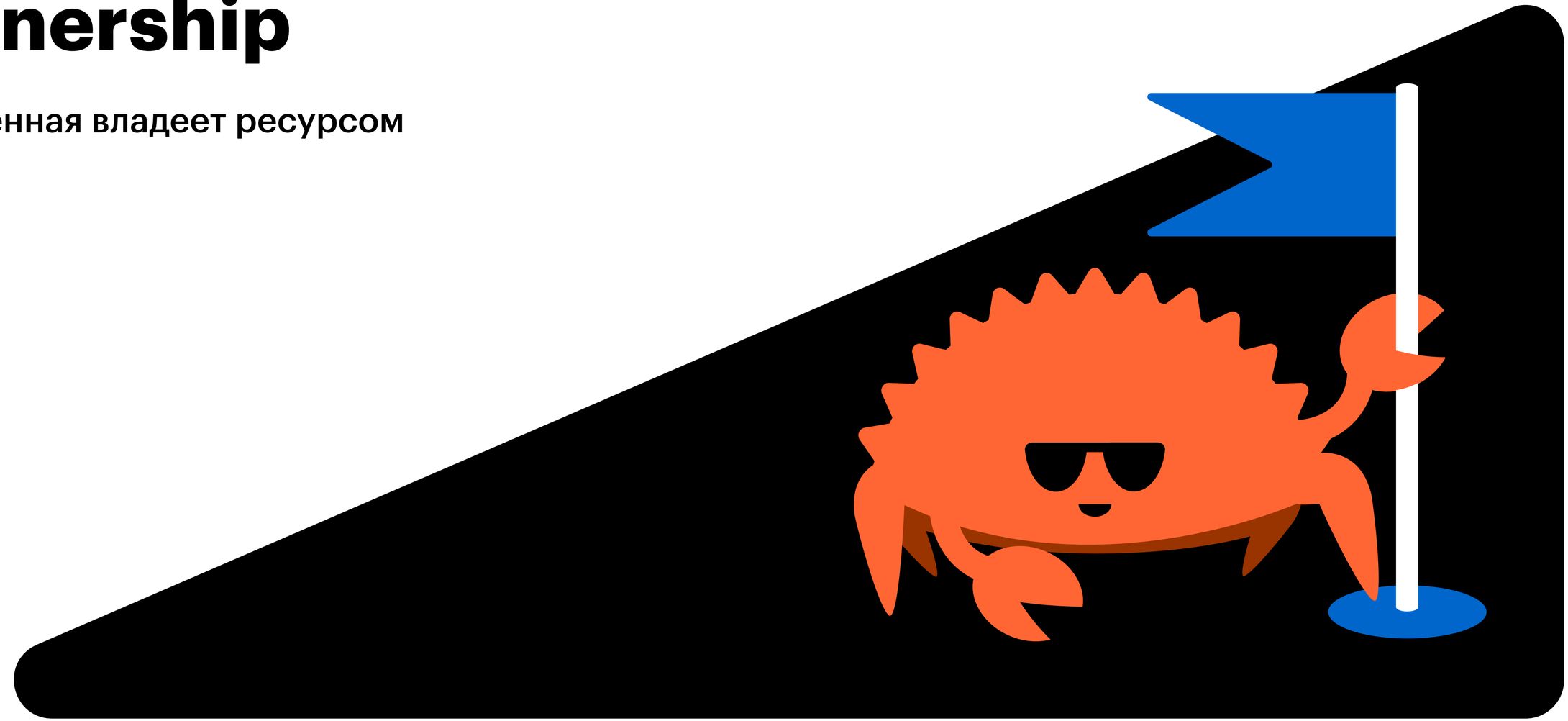


# Концепции Rust

@mgrachev
evrone

## Ownership

Переменная владеет ресурсом



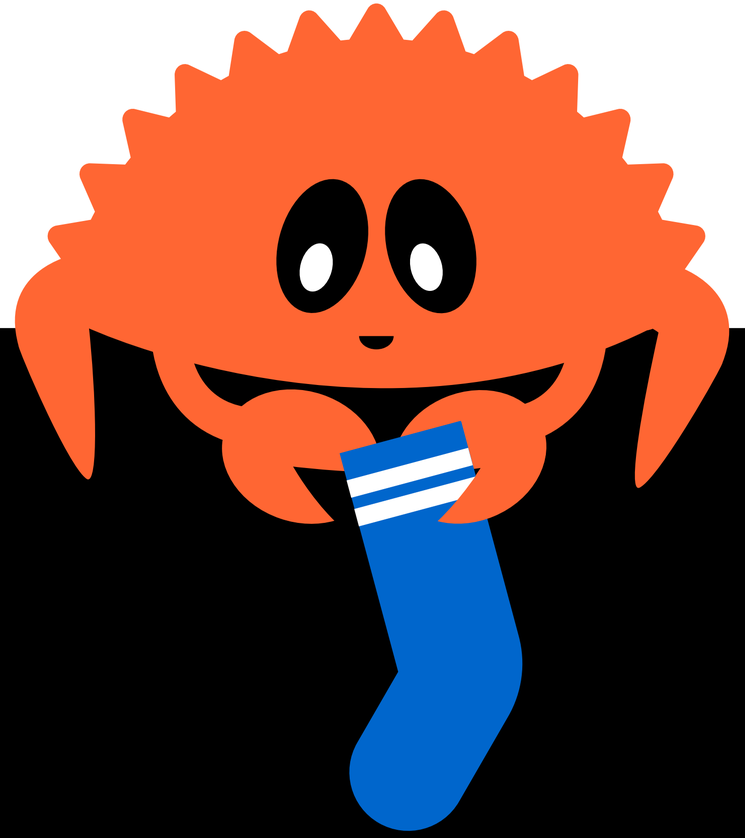
# Концепции Rust

@mgrachev
evrone

## Resource Acquisition Is Initialization (RAII)

Автоматическое освобождение ресурса

```
fn main() { // <- start of scope
    let _a = [0; 10];
} // <- end of scope
```



# Концепции Rust

@mgrachev
evrone

## Деструкторы

Освобождают системные ресурсы

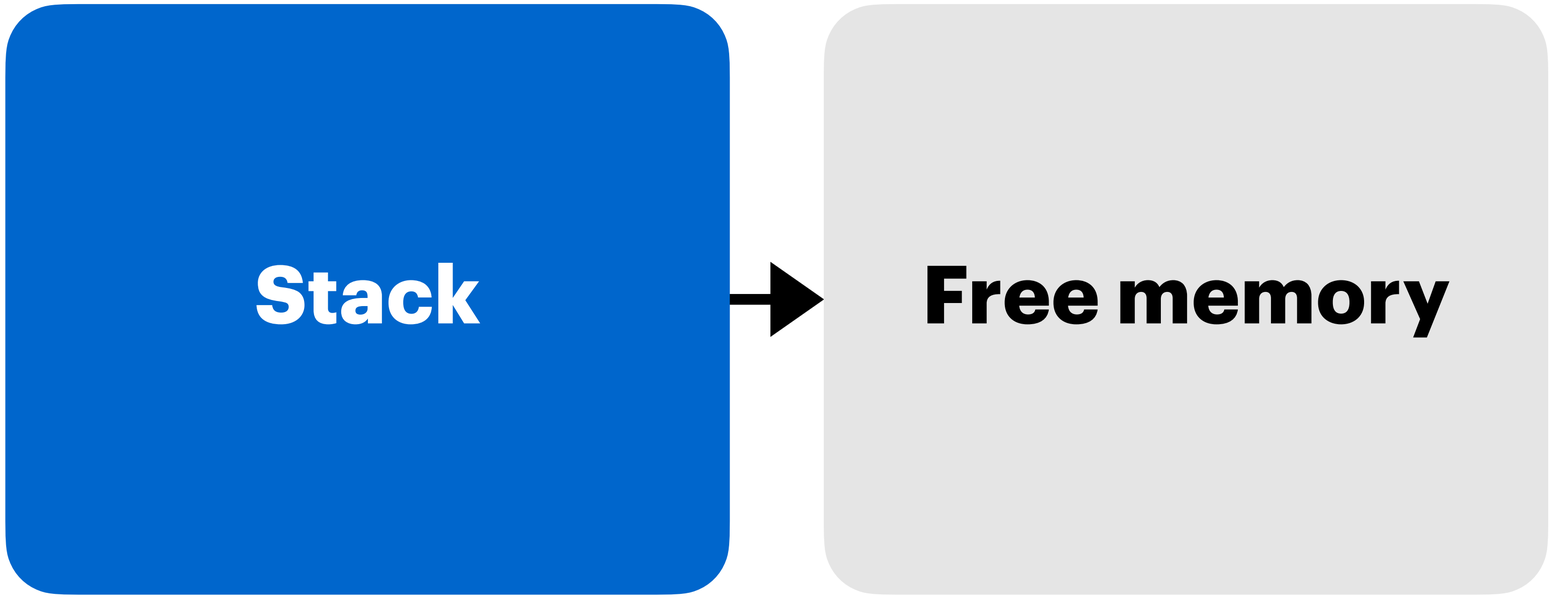
```rust
struct HasDrop;

impl Drop for HasDrop {
    fn drop(&mut self) {
        println!("Dropping HasDrop!");
    }
}
```

# Концепции Rust

@mgrachev
evrone

## Stack

Хранение данных по умолчанию

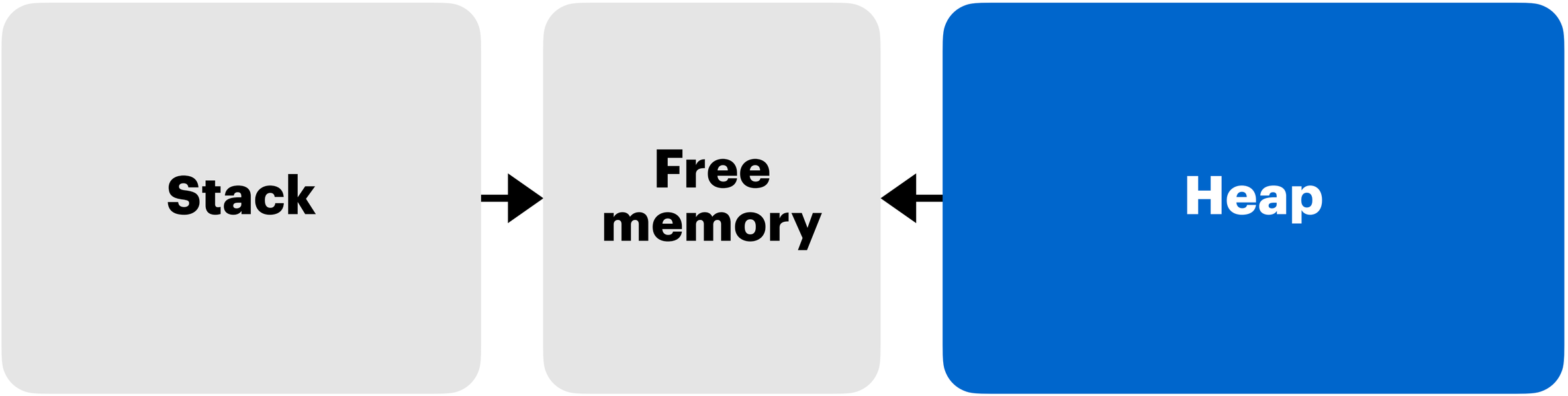


# Концепции Rust

@mgrachev
evrone

## Heap

Для динамически расширяемых объектов



# Переменные в Ruby/Python

@mgrachev
evrone

```
1 a = []
2 b = a
3 b.push(1)
4 a == [1] # => true
```

# Переменные в Rust

@mgrachev
evrone

```rust
fn main() {
    let a = vec![];
    let mut b = a;
    b.push(1);
    assert_eq!(a, b); // error
}
```

# Переменные в Rust

@mgrachev
evrone

```
error[E0382]: borrow of moved value: `a`
 --> src/main.rs:5:5
2 |     let a = vec![];
  |         - move occurs because `a` has type `Vec<i32>`,
            which does not implement the `Copy` trait
3 |     let mut b = a;
  |                 - value moved here
5 |     assert_eq!(a, b);
  |     ^^^^^^^^^^^^^^^^ value borrowed here after move
```

# Move семантика

@mgrachev
evrone

```rust
fn foo(a: Vec<i32>) {}

fn main() {
    let a = vec![1];
    let b = a;
    println!("{:?}", a); // error
    // Or
    let a = vec![];
    foo(a);
    println!("{:?}", a); // error
}
```

# Copy семантика

@mgrachev
evrone

```rust
fn foo(a: i32) {}

fn main() {
    let a = 1;
    let b = a;
    println!("{:?}", a); // ok
    // Or
    foo(a);
    println!("{:?}", a); // ok
}
```

# Заимствование

@mgrachev
evrone

```
a = []
b = a
b.push(1)
a == [1] # => true
```

# Ограничения заимствования

@mgrachev
evrone

Сколько угодно immutable ссылок

```
1 fn main() {
2     let a = vec![1, 2, 3];
3     let b = &a;
4     let c = &a;
5
6     assert_eq!(a, *b);
7     assert_eq!(a, *c);
8 }
```

# Ограничения заимствования

@mgrachev
evrone

Сколько угодно immutable ссылок

```
1 fn main() {
2     let a = vec![1, 2, 3];
3     let b = &a;
4     let c = &a;
5
6     assert_eq!(a, *b);
7     assert_eq!(a, *c);
8 }
```

Либо только одна mutable ссылка

```
1 fn main() {
2     let mut a = vec![];
3     let b = &mut a;
4     b.push(1);
5     assert_eq!(a, vec![1]);
6 }
```

# Массивы в Ruby/Python

@mgrachev
evrone

```ruby
a = [1]
a.push(2)
a == [1, 2] # true

a = [1] * 10_000_000 # ok
```

# Массивы в Rust

@mgrachev
evrone

```rust
fn main() {
    let mut a = [1, 2];
    let b = a; // copy semantics
    a[1] = 1;
    assert_eq!(a, [1, 1]);
    assert_eq!(b, [1, 2]);

    let c = [1; 10_000_000]; // fatal error: stack overflow
}
```

# Динамический массив в Rust

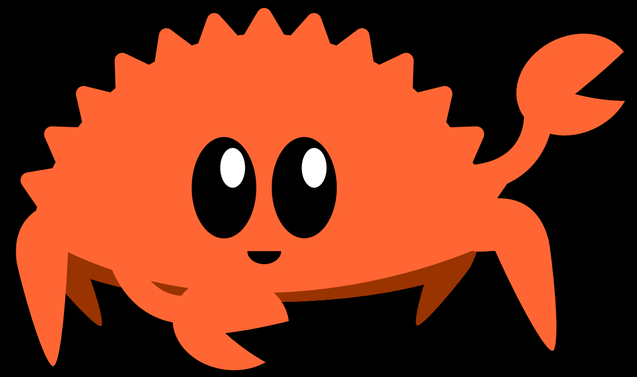
@mgrachev
evrone

```rust
fn main() {
    let a = vec![1; 10_000_000]; // ok
    let mut b = a; // move semantics
    b.push(1);
    assert_eq!(b.len(), 10_000_001);

    assert_eq!(a.len(), 10_000_001); // error
}
```

# Строки

@mgrachev
evrone

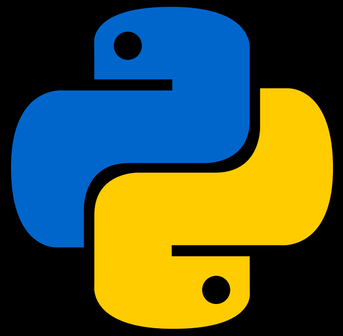
Rust



&str
String
&[u8]
&[u8; N]
Vec<u8>
&u8
OsStr
OsString
Path
PathBuf
CStr
CString
&’ static str

String

Ruby
Python



# Строковые литералы &str

@mgrachev
evrone

```rust
fn main() {
    let s = "foo";
    let s2 = s; // copy semantics
    assert_eq!(s, s2);
}
```

# String

@mgrachev
evrone

```rust
fn main() {
    let s = String::from("foo");
    let s2 = s; // move semantics
    assert_eq!(s, s2); //error
}
```

# ООП: Классы

@mgrachev
evrone

```python
class Person:
  def __init__(self, name):
    self.name = name

  def greeting(self):
    print(f"Hello, my name is {self.name}")

p1 = Person("John")
p1.greeting()
```

# ООП: Structs

@mgrachev
evrone

```rust
struct Person {
    name: String,
}

impl Person {
    fn new(name: String) -> Self {
        Self { name }
    }
    fn greeting(&self) {
        println!("Hello my name is {}", self.name)
    }
}

let p = Person::new("John".to_string());
p.greeting();
```

# ООП: Композиция

@mgrachev
evrone

```rust
struct Person {
    name: String,
    fields: Fields,
}

struct Fields {
    age: u8,
}

impl Person {
    fn new(name: String, age: u8) -> Self {
        Self { name, fields: Fields { age }}
    }
}
```

# ООП: Traits

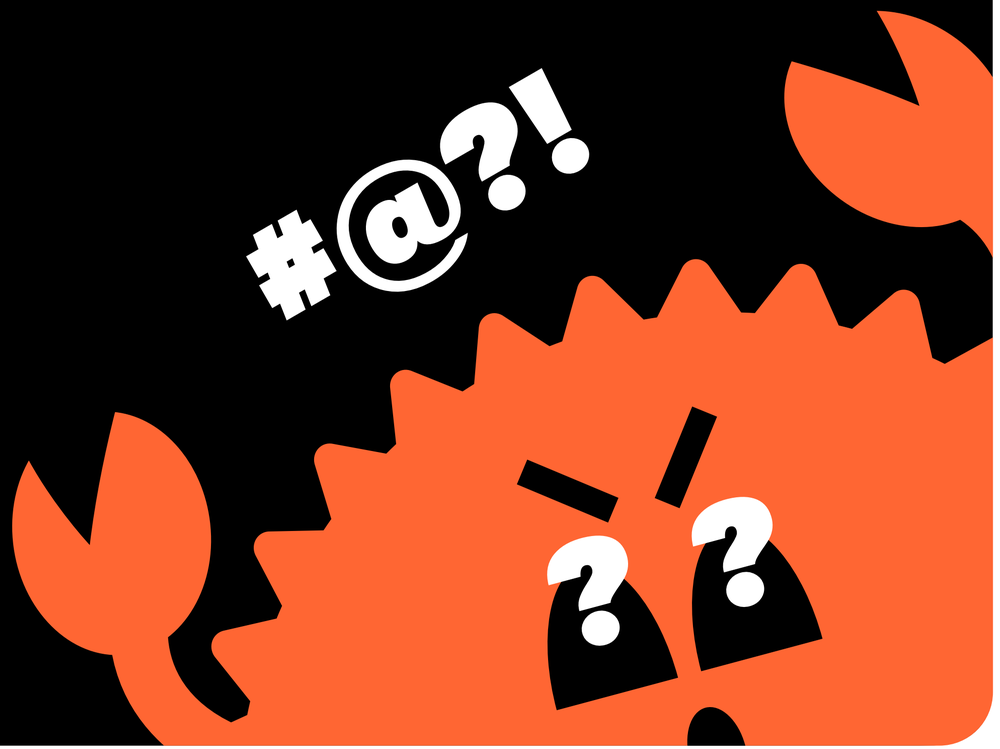
@mgrachev
evrone

```
1 trait Greeting: Display {
2     fn greeting(&self) {
3         println!("Hello")
4     }
5 }
6
7 impl Greeting for Person {
8     fn greeting(&self) {
9         println!("Hello my name is {}", self.name)
10    }
11 }
```

# ООП: Generics

@mgrachev
evrone

```rust
trait Greeting: Display {
    fn greeting(&self) {
        println!("Hello")
    }
}

impl Greeting for Person {
    fn greeting(&self) {
        println!("Hello my name is {}", self.name)
    }
}

fn greeting<T: Greeting>(p: T) {
    p.greeting()
}
```



# ООП: Trait Objects

@mgrachev
evrone

```
1 fn greeting(p: &dyn Greeting) {
2     p.greeting()
3 }
```

# ООП: Trait Objects

@mgrachev
evrone

```
1 struct DynStruct {
2     fields: Vec<Box<dyn Greeting>>
3 }
```

# Работа с ошибками

@mgrachev
evrone

```
1 try:
2     somelib.download(url)
3 except:
4     print("network error")
```

# Отсутствуют nil, null

@mgrachev
evrone

```
1 enum Option<T> {
2   Some(T),
3   None,
4 }
5
6 let value: Option<i8> = Some(5);
7 // Or
8 let value: Option<i8> = None;
```

# Отсутствуют исключения

@mgrachev
evrone

```
1 enum Result<T, E> {
2   Ok(T),
3   Err(E),
4 }
5
6 let value: Result<i8, &str> = Ok(8);
7 // Or
8 let value: Result<i8, &str> = Err("message");
```

# Pattern matching и работа с ошибками

@mgrachev
evrone

```
1 match download(url) {
2     Ok(data) => handle(data),
3     Err(e) => match e.kind() {
4         NotFound => {}
5         InvalidData => {}
6         _ => {}
7     },
8 }
```

# Concurrency

@mgrachev
evrone

Потоки без GIL, Mutex,
каналы с сообщениями

# Concurrency

@mgrachev
evrone

Потоки без GIL, Mutex, каналы с сообщениями

Async/await из коробки, но без runtime

# Метапрограммирование через макросы

@mgrachev
evrone

## Декларативные

```
1 macro_rules! say_hello {
2     () => {
3         println!("Hello!");
4     };
5 }
```

# Метапрограммирование через макросы

@mgrachev
evrone

## Процедурные

```
1 #[proc_macro]
2 pub fn make_answer(_item: TokenStream) -> TokenStream {
3     "fn answer() -> u32 { 42 }".parse().unwrap()
4 }
```

# Инструменты

@mgrachev
evrone

¯_(ツ)_/¯

Батарейки
не включены

# Инструменты

@mgrachev
evrone

¯_(ツ)_/¯

Батарейки
не включены

**cargo**

Пакетный
менеджер
и не только

# Инструменты

@mgrachev
evrone

¯_(ツ)_/¯

Батарейки
не включены

**cargo**

Пакетный
менеджер
и не только

**rustfmt**

Конец войны
за стили

# Инструменты

@mgrachev
evrone

**¯_(ツ)_/¯**

Батарейки
не включены

**cargo**

Пакетный
менеджер
и не только

**rustfmt**

Конец войны
за стили

**clippy**

Набор
линтеров

# Заключение

@mgrachev
evrone

Круто,
безопасно,
но сложно

# Заключение

@mgrachev
evrone

**Круто, безопасно, но сложно**

**Компилятор — ваш друг**

(возможно, не сразу)

# Заключение

@mgrachev
evrone

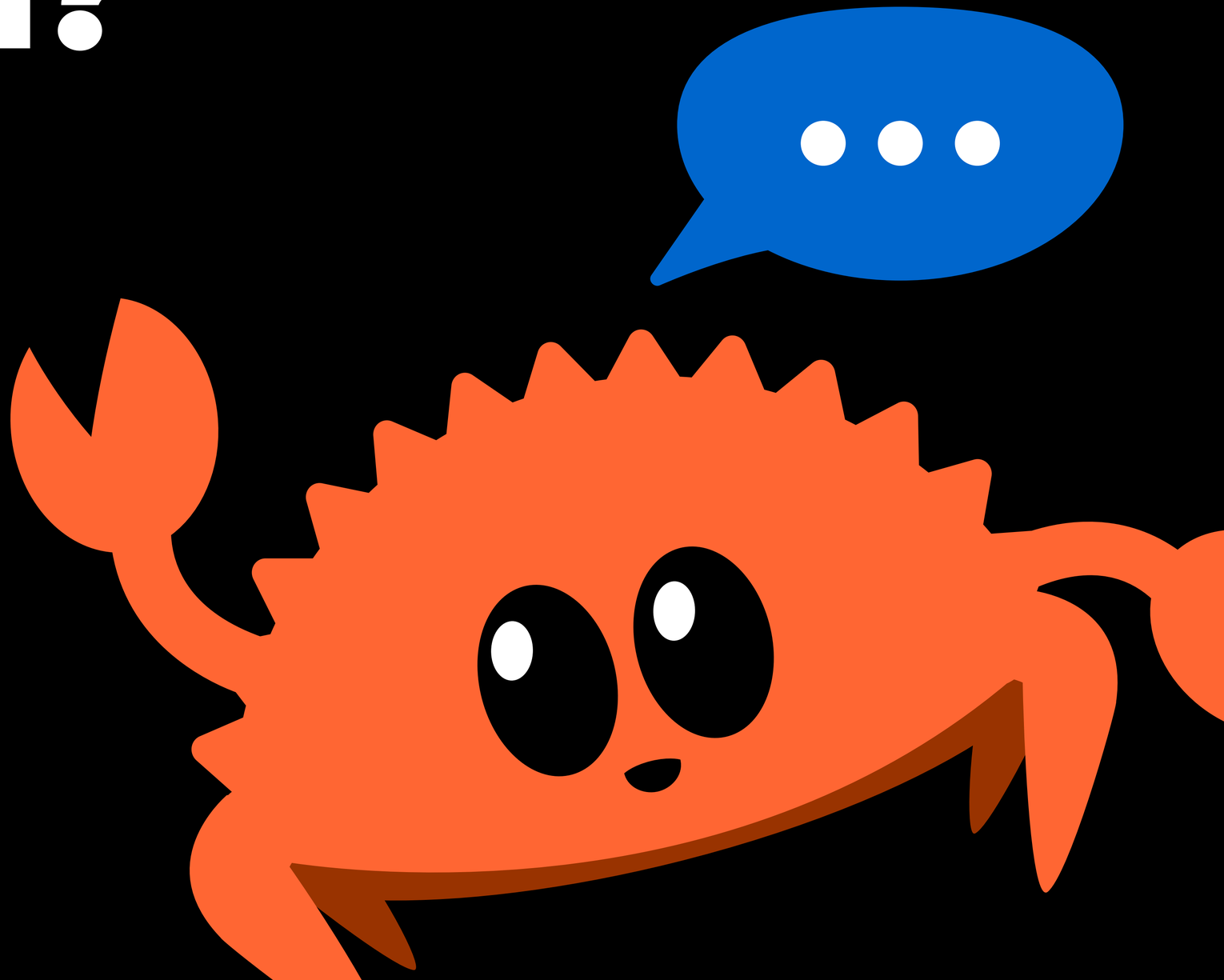
**Круто, безопасно, но сложно**

**Компилятор — ваш друг**

(возможно, не сразу)

**Обработка ошибок на высоте**

evrone.com

# Вопросы?

GitHub/Twitter
@mgrachev