Rust - обзор экосистемы

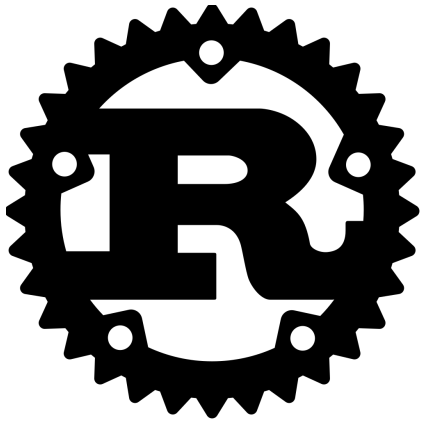
BITFURY

Илья Богданов
@vitvakatu

План

- Rust — краткое введение
- Инструменты для разработчика
- Библиотеки
- Взаимодействие с C/C++

Часть 1 - Rust

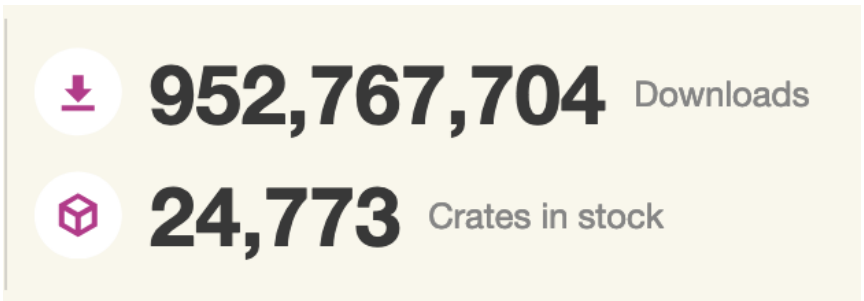


Rust Programming Language

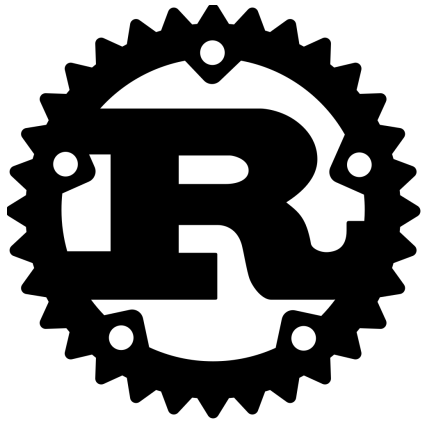
Скорость **или** Безопасность?

Announcing Rust 1.0 Alpha

Jan. 09, 2015 · The Rust Core Team



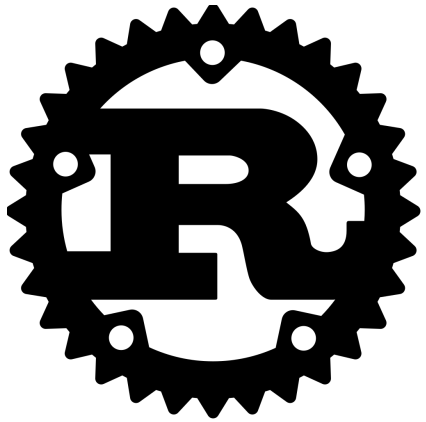
Rust Voted Most Loved Language for the 3rd Year in a Row in Stack Overflow Developer Survey



Rust Programming Language

Скорость *и* Безопасность

... и Выразительность



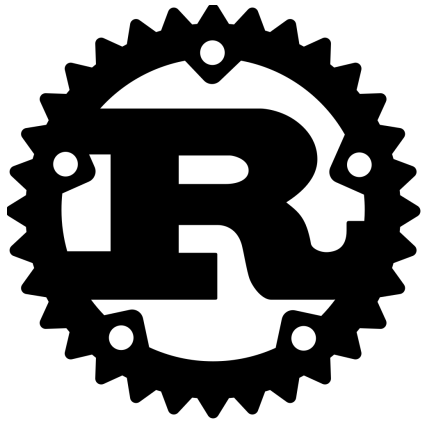
Rust Programming Language

Скорость = **zero-cost abstractions**

Скорость = **LLVM**

Скорость = нет сборщика мусора

Скорость = минималистичный рантайм

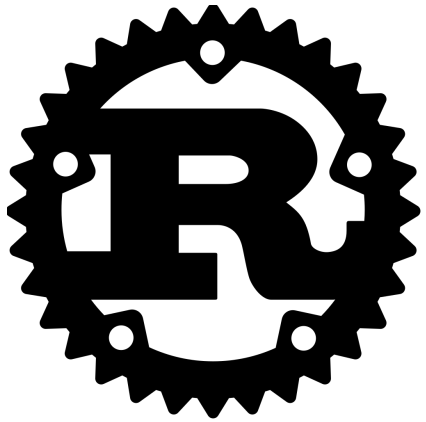


Rust Programming Language

Безопасность = нет null

Безопасность = нет гонок данных

Безопасность = нет UB*



Rust Programming Language

Выразительность = развитая система типов

Выразительность = **pattern-matching**

Выразительность = макросы

Выразительность = итераторы

Скомпилируется?

```
let a = 1;
let b = 2;
a = a + b;
```

Скомпилируется?

```
let a: i32 = 1;
let b: i32 = 2;
a = a + b;
```

Скомпилируется?

```
let mut a = 1;
let b = 2;
a = a + b;
```

Скомпилируется?

```
let mut some_value;
for i in 0..array.len() {
    if some_condition(array[i]) {
        some_value = array[i];
    }
}

println!("Found! {}", some_value);
```

Скомпилируется?

```
let a = 1;
let b = a;

println!("A = {}", a);
println!("B = {}", b);
```

Скомпилируется?

```
let a = 10;
let b = &a;
let c = &a;
println!("B = {}", b);
println!("C = {}", c);

// Выведет
// B = 10
// C = 10
```

Скомпилируется?

```
let mut a = 10;
let b = &a;
let c = &mut a;
*c = 20;
println!("B = {}", b);
println!("C = {}", c);
```

Скомпилируется?

```rust
let mut a = 10;
let b = &a;
let c = &mut a;
println!("B = {}", b);
println!("C = {}", c);
```

```
error[E0502]: cannot borrow `a` as mutable because it is also borrowed as immutable
 --> src/main.rs:4:13
  |
3 |     let b = &a;
  |             -- immutable borrow occurs here
4 |     let c = &mut a;
  |             ^^^^^^ mutable borrow occurs here
5 |     println!("B = {}", b);
  |                        - immutable borrow later used here
```

Borrow Checker

```
struct S {}

fn get_first_element(array: &[S]) -> &S {
    let element = &array[0];
    return element;
}
```

Borrow Checker

```
struct S {}

fn get_first_element<'a>(array: &'a [S]) -> &'a S {
    let element: &'a S = &array[0];
    return element;
}
```

Borrow Checker

```
fn select_str(s1: &str, s2: &str) -> &str {
    if s1.len() > s2.len() {
        return s1;
    } else {
        return s2;
    }
}
```

```
error[E0106]: missing lifetime specifier
 --> src/main.rs:1:38
  |
1 | fn select_str(s1: &str, s2: &str) -> &str {
  |                                      ^ expected lifetime parameter
  |
  = help: this function's return type contains a borrowed value,
 but the signature does not say whether it is borrowed from `s1` or `s2`
```

Borrow Checker

```
fn select_str<'a, 'b: 'a>(s1: &'a str, s2: &'b str) -> &'a str {
    if s1.len() > s2.len() {
        return s1;
    } else {
        return s2;
    }
}
```

Безопасная многопоточность

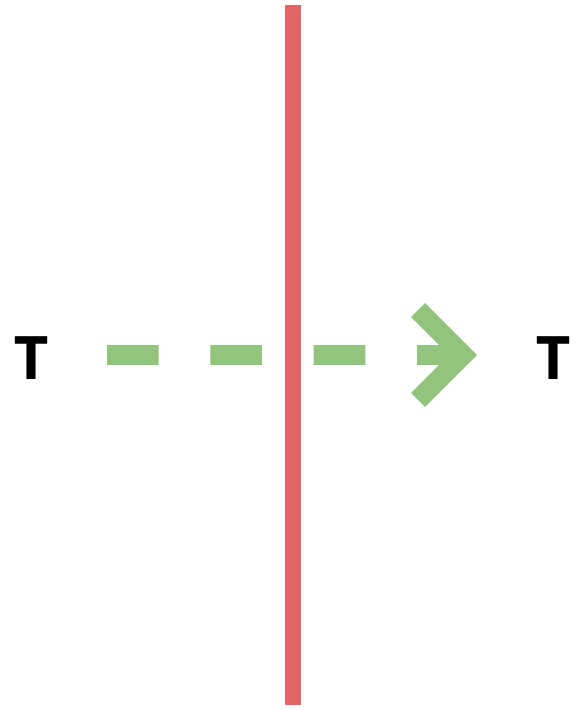
```
trait A {
    fn method1();
    fn method2();
}

struct S {}

impl A for S {
    fn method1() { /* implementation */ }

    fn method2() { /* implementation */ }
}
```

Безопасная многопоточность

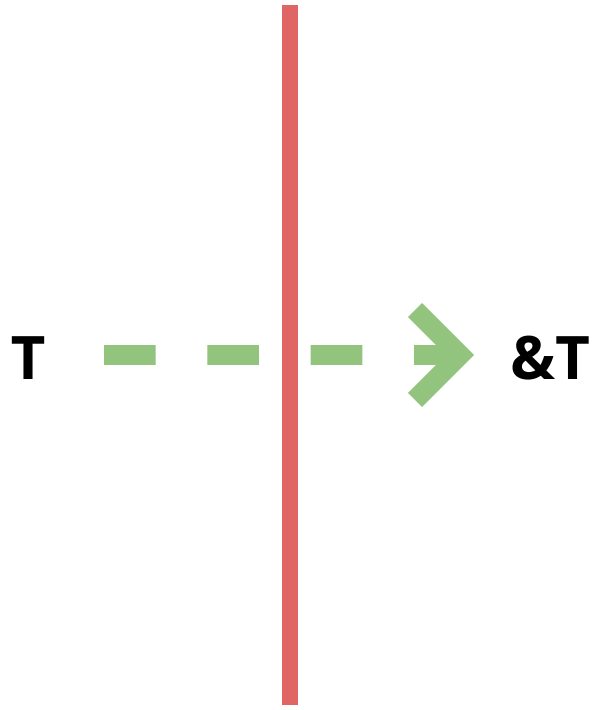
```
trait Marker {}

struct S {}

impl Marker for S {}

fn accept_only_markers<T: Marker>(t: T) {
    // ...
}
```

Безопасная многопоточность

Send

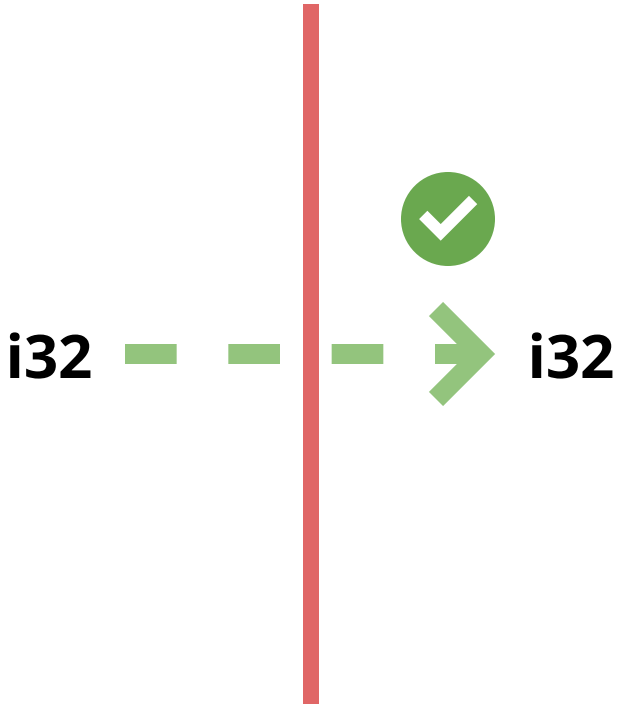


Sync

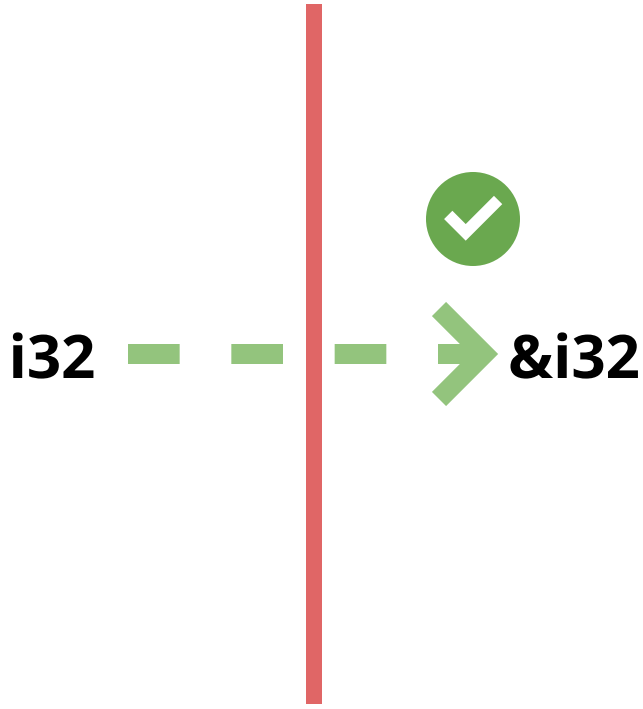


Безопасная многопоточность

Send

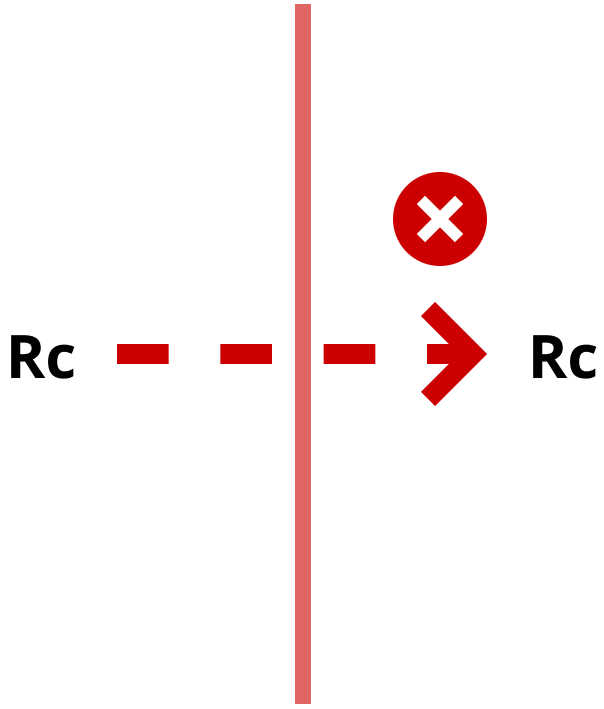


Sync

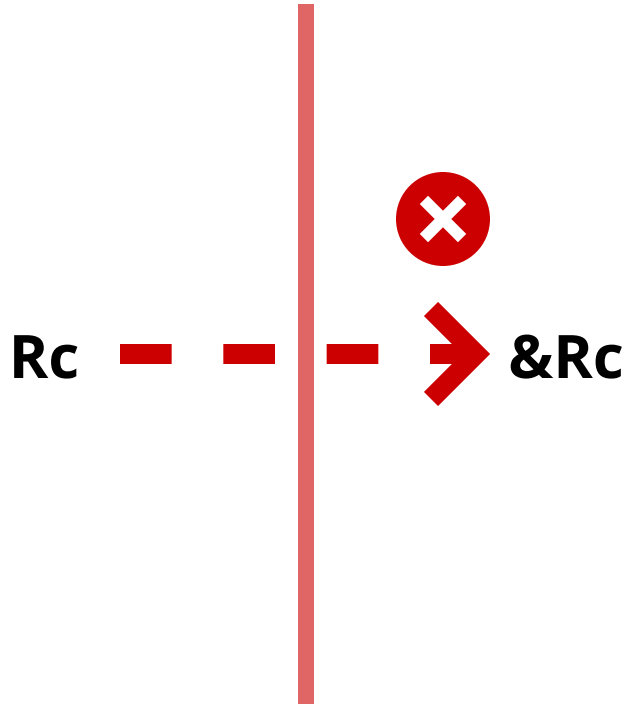


Безопасная многопоточность

Send



Sync



Итераторы

```
let max = array.iter().max();
```

```
fn shoes_in_my_size(
    shoes: Vec<Shoe>,
    shoe_size: u32
) -> Vec<Shoe> {
    shoes.into_iter()
        .filter(|s| s.size == shoe_size)
        .collect()
}
```

ADT

```
enum Option<T> {
    Some(T),
    None
}

enum Result<T, E> {
    Ok(T),
    Err(E),
}

let result = some_operation();
match result {
    Ok(_) => {},
    Err(e) => panic!("Oh, no, not again! {}", e),
}
```

Макросы

```
macro_rules! map {
    ($( $key:expr => $value:expr),*) => {{
        let mut hm = HashMap::new();
        $( hm.insert($key, $value); )*
        hm
    }};
}
```

Макросы

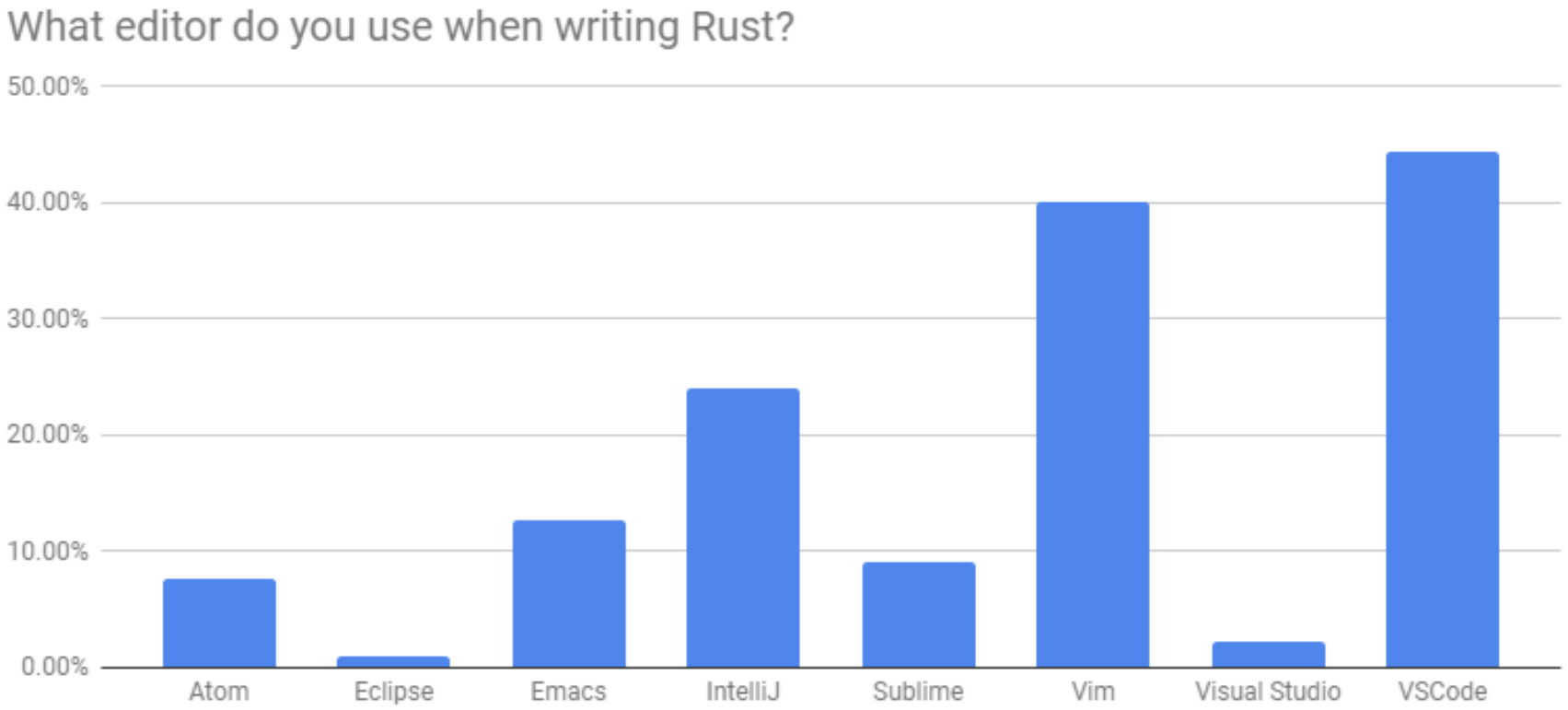
```
let user = map!(
    "name" => "John",
    "gender" => "Boy"
);
```

Макросы

```
#[derive(Debug, Clone)]
#[derive(PartialEq, Eq, Hash)]
struct MyStruct {
    field1: i32,
    field2: i32,
}
```

Часть 2 - Инструменты

Где писать код?



https://areweideyet.com

One Cargo to rule the world

```
⇒  cargo new my_awesome_project
     Created binary (application) `my_awesome_project` package


⇒  cd my_awesome_project
⇒  ls -la
total 16
drwxr-xr-x   6 ilyabogdanov  staff   192 Apr 15 21:07 .
drwxr-xr-x  57 ilyabogdanov  staff  1824 Apr 15 21:07 ..
drwxr-xr-x   9 ilyabogdanov  staff   288 Apr 15 21:07 .git
-rw-r--r--   1 ilyabogdanov  staff    19 Apr 15 21:07 .gitignore
-rw-r--r--   1 ilyabogdanov  staff   141 Apr 15 21:07 Cargo.toml
drwxr-xr-x   3 ilyabogdanov  staff    96 Apr 15 21:07 src


⇒  cat src/main.rs
fn main() {
    println!("Hello, world!");
}
```

One Cargo to rule the world

```
while isDayTime() {
    cargo build      // Компилировать
    cargo test       // Протестировать
    cargo bench      // Замерить производительность
    cargo fmt        // Форматировать код
    git commit -am "One more piece of work"
}
```

Часть 3 - Библиотеки

Serde - универсальная (де)сериализация

Serde

```rust
#[derive(Serialize, Deserialize, Debug)]
struct Point {
    x: i32,
    y: i32,
}

fn main() {
    let point = Point { x: 1, y: 2 };

    // Convert the Point to a JSON string.
    let serialized = serde_json::to_string(&point).unwrap();

    // Prints serialized = {"x":1,"y":2}
    println!("serialized = {}", serialized);

    // Convert the JSON string back to a Point.
    let deserialized: Point = serde_json::from_str(&serialized).unwrap();

    // Prints deserialized = Point { x: 1, y: 2 }
    println!("deserialized = {:?}", deserialized);
}
```

Serde

```rust
#[derive(Serialize, Deserialize, Debug)]
struct Point {
    x: i32,
    y: i32,
}

fn main() {
    let point = Point { x: 1, y: 2 };

    // Convert the Point to a JSON string.
    let serialized = serde_json::to_string(&point).unwrap();

    // Prints serialized = {"x":1,"y":2}
    println!("serialized = {}", serialized);

    // Convert the JSON string back to a Point.
    let deserialized: Point = serde_json::from_str(&serialized).unwrap();

    // Prints deserialized = Point { x: 1, y: 2 }
    println!("deserialized = {:?}", deserialized);
}
```

Serde

```rust
#[derive(Serialize, Deserialize, Debug)]
struct Point {
    x: i32,
    y: i32,
}

fn main() {
    let point = Point { x: 1, y: 2 };

    // Convert the Point to a JSON string.
    let serialized = serde_json::to_string(&point).unwrap();

    // Prints serialized = {"x":1,"y":2}
    println!("serialized = {}", serialized);

    // Convert the JSON string back to a Point.
    let deserialized: Point = serde_json::from_str(&serialized).unwrap();

    // Prints deserialized = Point { x: 1, y: 2 }
    println!("deserialized = {:?}", deserialized);
}
```

Serde

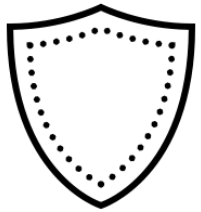
```rust
#[derive(Serialize, Deserialize, Debug)]
struct Point {
    x: i32,
    y: i32,
}

fn main() {
    let point = Point { x: 1, y: 2 };

    // Convert the Point to a JSON string.
    let serialized = serde_json::to_string(&point).unwrap();

    // Prints serialized = {"x":1,"y":2}
    println!("serialized = {}", serialized);

    // Convert the JSON string back to a Point.
    let deserialized: Point = serde_json::from_str(&serialized).unwrap();

    // Prints deserialized = Point { x: 1, y: 2 }
    println!("deserialized = {:?}", deserialized);
}
```

ACTIX

Web-framework на Акторах

Быстрый

Безопасный



Мощный

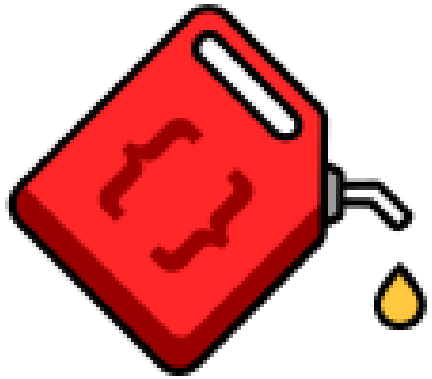
HTTP/2

SSL/TLS

WebSocket

Rnk	Framework	Best performance (higher is better)	
1	actix-raw	7,040,642	100.0%
2	fasthttp	7,026,716	99.8%
3	ulib-plaintext_fit	7,016,745	99.7%
4	wizzardo-http	7,014,982	99.6%
5	libreactor	7,011,500	99.6%
6	ulib	7,008,767	99.5%
7	tokio-minihttp	7,007,335	99.5%
8	may-minihttp	7,006,770	99.5%
9	rapidoid	6,999,211	99.4%
10	rapidoid-http-fast	6,995,006	99.4%
11	aspcore	6,970,937	99.0%
12	actix	6,715,989	95.4%
13	hyper	6,254,812	88.8%
14	cpoll_cppsp	5,734,965	81.5%
15	h2o	5,326,290	75.7%
16	proteus	4,841,612	68.8%
17	netty	4,560,356	64.8%

https://www.techempower.com/benchmarks



DIESEL

ORM & Query builder с верификацией **SQL** во время компиляции и высокой производительностью.

Diesel

```
#[derive(Queryable)]
pub struct Download {
    id: i32,
    version_id: i32,
    downloads: i32,
    counted: i32,
    date: SystemTime,
}
```

Diesel

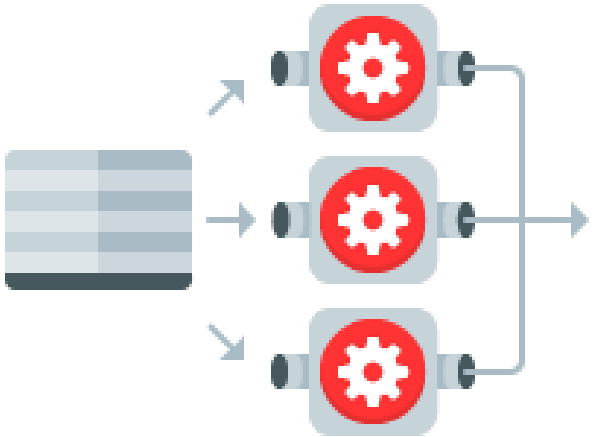
```
let versions = Version::belonging_to(krate)
    .select(id)
    .order(num.desc())
    .limit(5);
let downloads = version_downloads
    .filter(date.gt(now - 90.days()))
    .filter(version_id.eq(any(versions)))
    .order(date)
    .load::<Download>(&conn)?;
```

Diesel

```
SELECT version_downloads.*
  WHERE date > (NOW() - '90 days')
    AND version_id = ANY(
      SELECT id FROM versions
        WHERE crate_id = 1
        ORDER BY num DESC
        LIMIT 5
    )
  ORDER BY date
```

Rayon

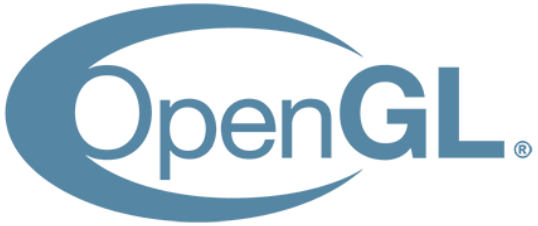
Data Parallelism for Everyone



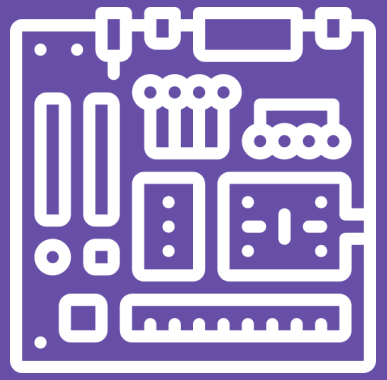
Rayon

```
stores.iter()
    .map(|store| store.compute_price(&list))
    .sum();


stores.par_iter()  // <- this line changed!
    .map(|store| store.compute_price(&list))
    .sum();
```

Современная универсальная
графическая библиотека



https://github.com/gfx-rs/gfx



Встраиваемые системы

Отключаемая стандартная библиотека

Простое взаимодействие с C

Гибкое управление памятью

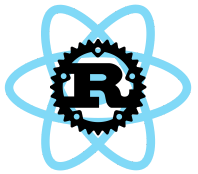
```rust
#![no_std]
#![no_main]

#[entry]
fn main() -> ! {
    let p = hal::Peripherals::take().unwrap();
    let cp = hal::CorePeripherals::take().unwrap();

    let mut sc = p.SYSCTL.constrain();
    // Pick our oscillation settings
    sc.clock_setup.oscillator = sysctl::Oscillator::Main(
        sysctl::CrystalFrequency::_16mhz,
        sysctl::SystemClock::UsePll(sysctl::PllOutputFrequency::_80_00mhz),
    );
    let clocks = sc.clock_setup.freeze();
    let mut porta = p.GPIO_PORTA.split(&sc.power_control);
    // ....
}
```

```rust
// ...
let uart = Serial::uart0(
    p.UART0,
    porta
        .pa1
        .into_af_push_pull::<AF1>(&mut porta.control),
    porta
        .pa0
        .into_af_push_pull::<AF1>(&mut porta.control),
    (),
    (),
    115200.bps(),
    NewlineMode::SwapLFtoCRLF,
    &clocks,
    &sc.power_control,
);

loop {
    writeln!(uart, "Hello, World!\r\n").unwrap();
}
```

WEBASSEMBLY

stdweb

```
$ cargo install -f cargo-web

$ cargo web start

# Open http://localhost:8000
```



Yew

React-like Frontend Framework in Rust

Yew

```
html! {
    <section class="todoapp",>
        <header class="header",>
            <h1>{ "todos" }</h1>
            { view_input(&model) }
        </header>
        <section class="main",>
            <input class="toggle-all",
                   type="checkbox",
                   checked=model.is_all_completed(),
                   onclick=|_| Msg::ToggleAll, />
            { view_entries(&model) }
        </section>
    </section>
}
```

Yew

```
let message = "Hello, 世界!";
let result = js! {
    alert( @{message} );
    return 2 + 2 * 2;
};

println!( "2 + 2 * 2 = {:?}", result );
```

https://github.com/deniskolodin/yew

Часть 4 - C/C++

Rust FFI

```
#[cfg(all(target_os = "win32", target_arch = "x86"))]
#[link(name = "kernel32")]
#[allow(non_snake_case)]
extern "stdcall" {
    fn SetEnvironmentVariableA(n: *const u8, v: *const u8) -> libc::c_int;
}
```

Rust FFI

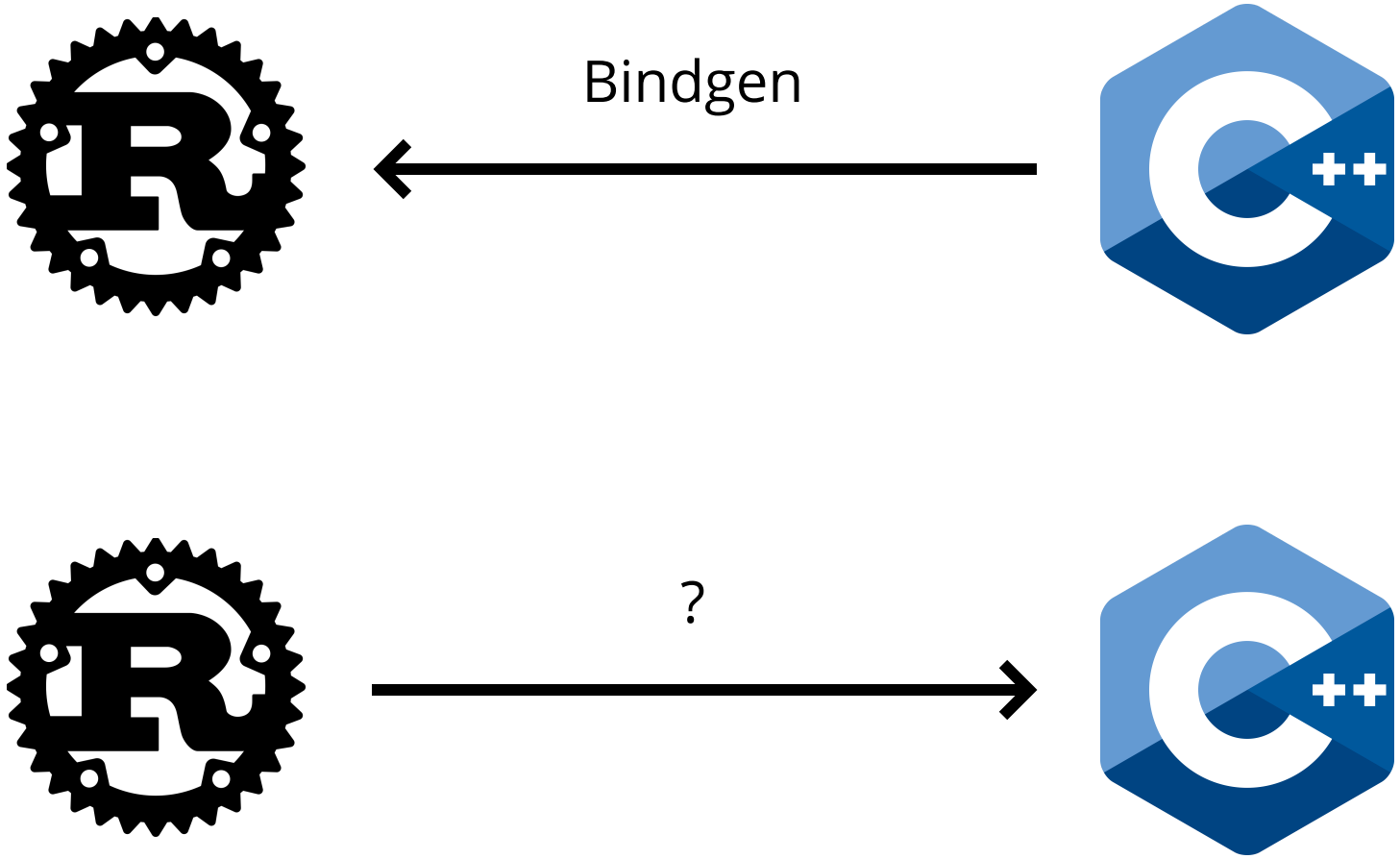
```
#[link(name = "snappy")]
extern {
    fn snappy_compress(input: *const u8,
                       input_length: size_t,
                       compressed: *mut u8,
                       compressed_length: *mut size_t) -> c_int;
    fn snappy_uncompress(compressed: *const u8,
                         compressed_length: size_t,
                         uncompressed: *mut u8,
                         uncompressed_length: *mut size_t) -> c_int;
    fn snappy_max_compressed_length(source_length: size_t) -> size_t;
    fn snappy_uncompressed_length(compressed: *const u8,
                                  compressed_length: size_t,
                                  result: *mut size_t) -> c_int;
    fn snappy_validate_compressed_buffer(compressed: *const u8,
                                         compressed_length: size_t) -> c_int;
}
```

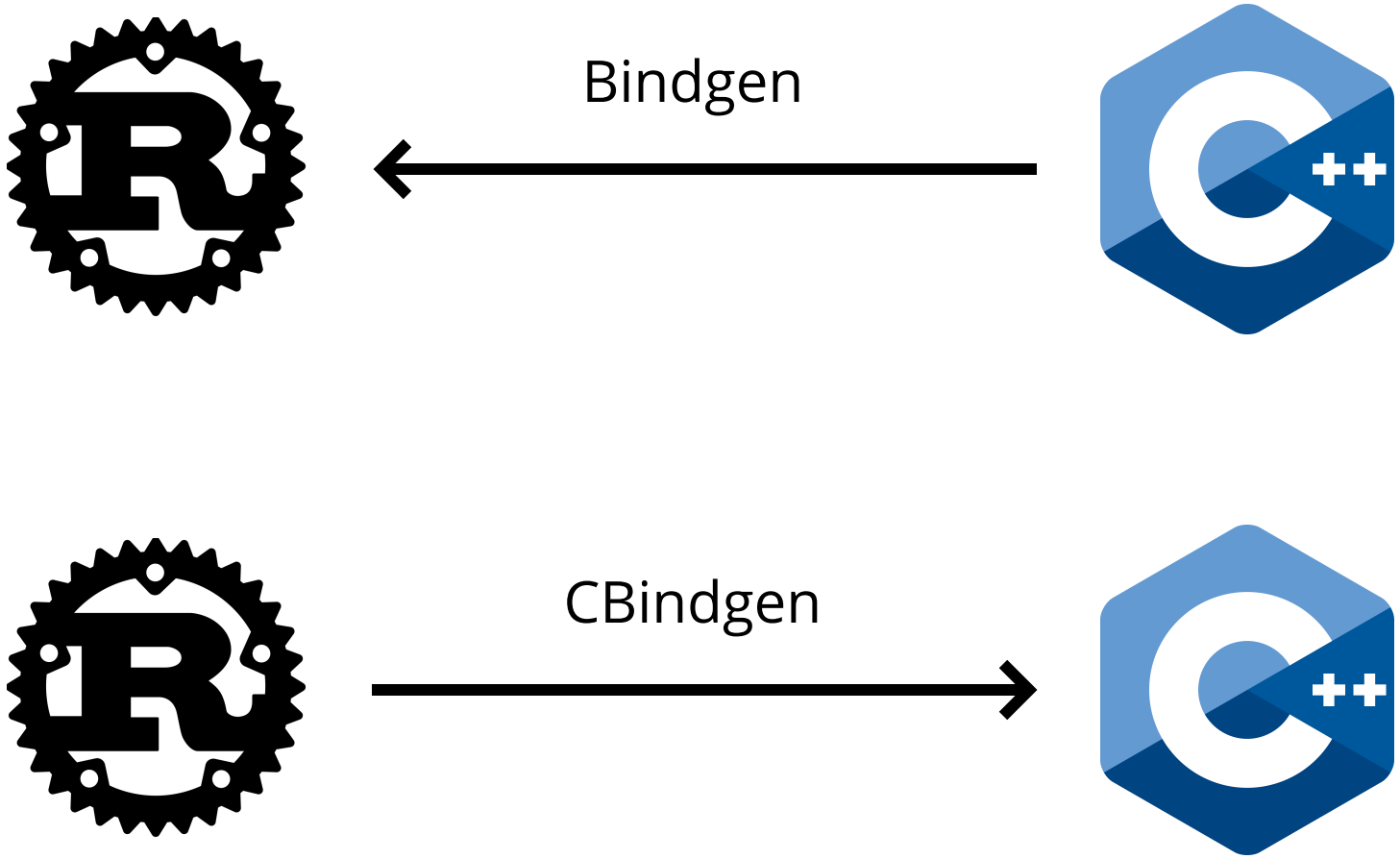
Rust Bindgen

```
// cool.h

typedef struct CoolStruct {
    int x;
    int y;
} CoolStruct;

void cool_function(int i, char c, CoolStruct* cs);
```

Rust Bindgen

```
/* automatically generated by rust-bindgen */

#[repr(C)]
pub struct CoolStruct {
    pub x: std::os::raw::c_int,
    pub y: std::os::raw::c_int,
}

extern "C" {
    pub fn cool_function(i: std::os::raw::c_int,
                         c: std::os::raw::c_char,
                         cs: *mut CoolStruct);
}
```


Bindgen
?


Bindgen
CBindgen

CBindgen

```
#[repr(C)]
enum Foo {
    A([f32; 20])
}

#[no_mangle]
pub extern "C" fn root(a: Foo) {}
```

CBindgen

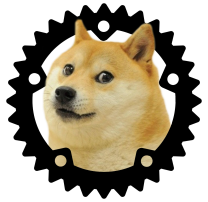
```
typedef enum {
  A,
} Foo_Tag;

typedef struct {
  float _0[20];
} A_Body;

typedef struct {
  Foo_Tag tag;
  union {
    A_Body a;
  };
} Foo;

void root(Foo a);
```

WebGPU

```
let instance = wgpu::Instance::new();
let adapter = instance.get_adapter(&wgpu::AdapterDescriptor {
    power_preference: wgpu::PowerPreference::LowPower,
});
let mut device = adapter.create_device(&wgpu::DeviceDescriptor {
    extensions: wgpu::Extensions {
        anisotropic_filtering: false,
    },
});
```

WebGPU

```
WGPUInstanceId instance = wgpu_create_instance();

WGPUAdapterId adapter = wgpu_instance_get_adapter(instance,
    &(WGPUAdapterDescriptor){
        .power_preference = WGPUPowerPreference_LowPower,
});

WGPUDeviceId device = wgpu_adapter_create_device(adapter,
    &(WGPUDeviceDescriptor){
        .extensions =
            {
                .anisotropic_filtering = false,
            },
});
```



Вопросы?

@vitvakatu

meetup.com/spbrust

@rustlang_ru