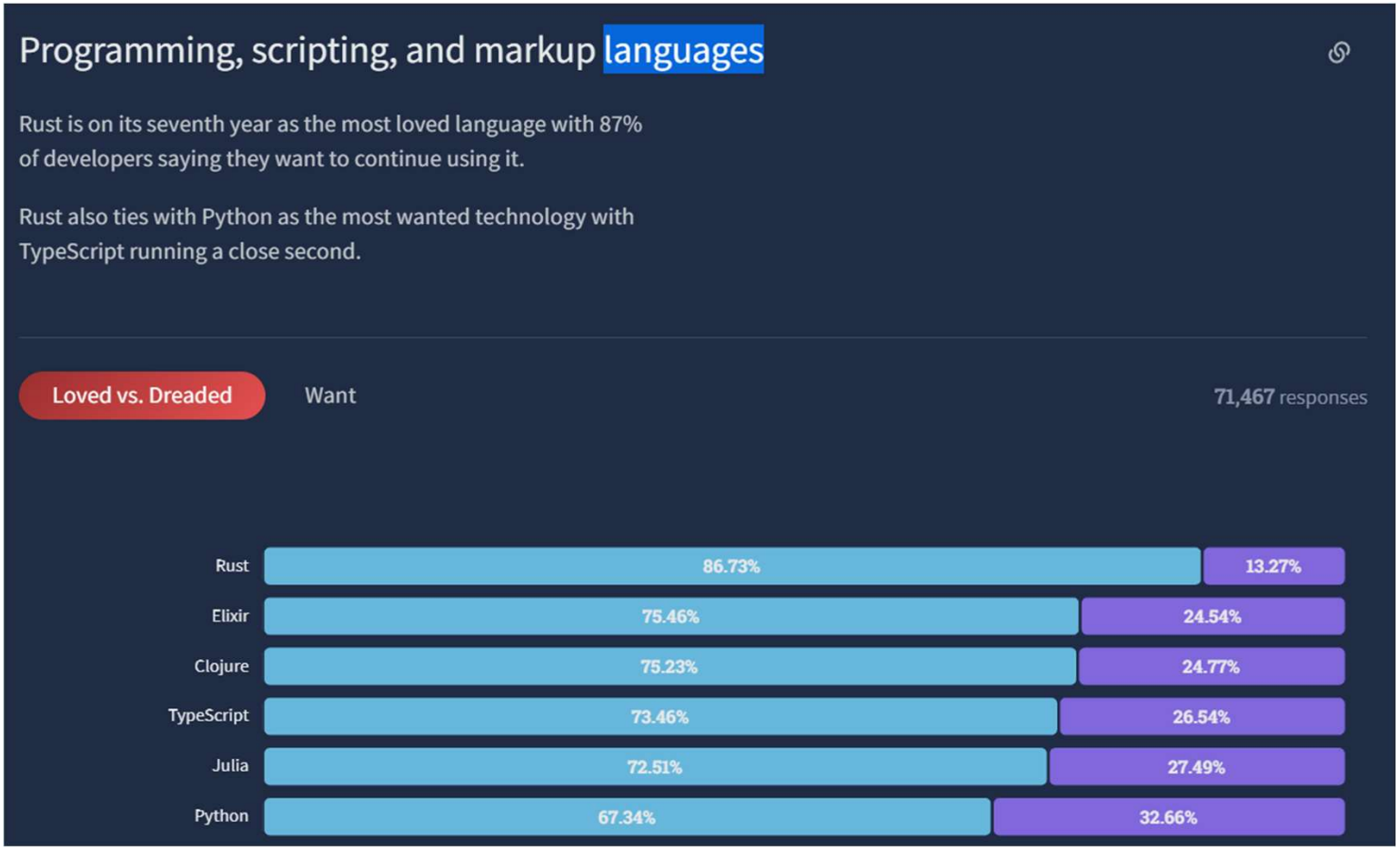
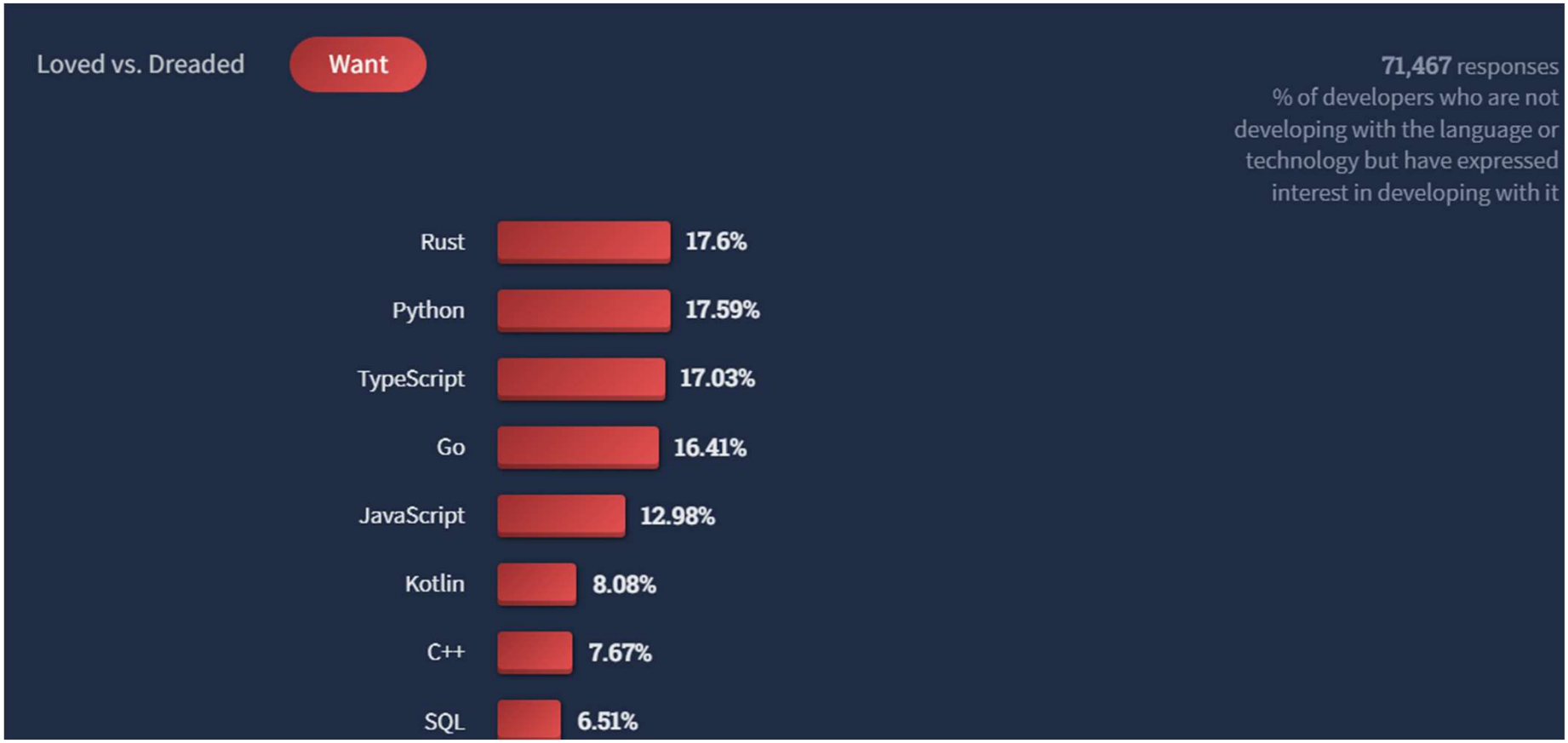
Rust 후기

방석현
- 프로그래밍 언어 찍먹 장인
- 사실 Rust 잘못함

그게 뭔데요


Programming, scripting, and markup languages
Rust is on its seventh year as the most loved language with 87% of developers saying they want to continue using it.
Rust also ties with Python as the most wanted technology with TypeScript running a close second.
Loved vs. Dreaded
Want
71,467 responses
Rust
86.73%
13.27%
Elixir
75.46%
24.54%
Clojure
75.23%
24.77%
TypeScript
73.46%
26.54%
Julia
72.51%
27.49%
Python
67.34%
32.66%

그게 뭔데요


Loved vs. Dreaded
Want
71,467 responses
% of developers who are not developing with the language or technology but have expressed interest in developing with it
Rust
17.6%
Python
17.59%
TypeScript
17.03%
Go
16.41%
JavaScript
12.98%
Kotlin
8.08%
C++
7.67%
SQL
6.51%

일단 당연히 Hello, World!

```
fn main() {
    println!("Hello, world!");
}
```

- 이 언어… 쉬운 것 같다!
- 근데… 함수가… fn으로 이니셜… 어라 리턴은?

Rust를 컴파일하려면?

- Cargo
  - Crate를 활용해 Dependency를 자동으로 세팅함 (npm보다 편하다!)
  - cargo build, cargo run, cargo test!

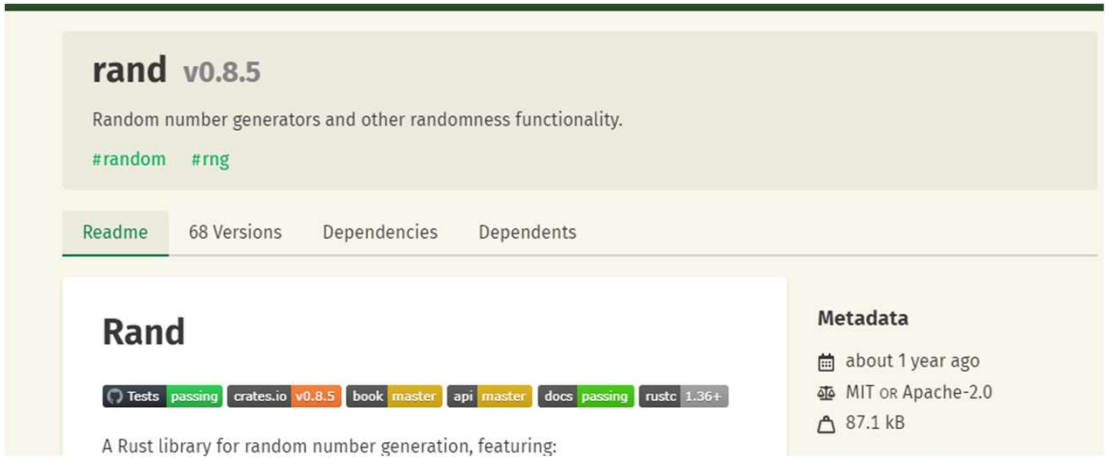
Rust를 컴파일하려면?

```
#[cfg(test)]
mod tests {
        use std::fs::OpenOptions;
        use std::io::Write
        #[test]
        fn test_file() {
                let mut file = OpenOptions::new()
                        .append(true)
                        .create(true)
                        .open("ferris.txt")
                        .expect("Failed to open ferris.txt");

                for _ in 0..5 {
                        file.write_all("Ferris\n".as_bytes())
                           .expect("Could not write to ferris.txt");
                }
}
```

Rust를 컴파일하려면?

- Cargo
  - Crate를 활용해 Dependency를 자동으로 세팅함 (npm보다 편하다!)
  - cargo build, cargo run, cargo test!
  - Library에 신조가 담겨있다
    - Backwards Compatibility



Rust를 컴파일하려면?

- Cargo
  - 그 무엇보다 친절하다… (하지만 이유가 있었다)
  - 그 전에 예시부터~

함수가 이상하다

```
fn fibo(n: u32) -> u32{
    match n{
            0| 1 | 2 => n,
            3.. => fibo(n - 2) + fibo(n - 1),
    }
}

fn main(){
    println!("{}", fibo(30));
}
```

함수가 이상하다

```
fn fibo(n: u32) -> u32{
    match n{
            0 | 2 => n,
            3.. => fibo(n - 2) + fibo(n - 1),
    }
}

fn main(){
    println!("{}", fibo(30));
}
```

```
   Compiling playground v0.0.1 (/playground)
error[E0004]: non-exhaustive patterns: `1_u32` not covered
 --> src/main.rs:2:11
  |
2 |     match n{
  |           ^ pattern `1_u32` not covered
  |
  = note: the matched value is of type `u32`
help: ensure that all possible cases are being handled by addi
  |
4 ~         3.. => fibo(n - 2) + fibo(n - 1),
5 ~         1_u32 => todo!(),
  |
```

난생 처음보는 것만 같은 이상한 것들

- Match
  - Switch이긴 한데… 하나라도 빠지면 오류 나서 안 돔
- Trait
  - Abstract class… 근데 하나라도 빠지면 오류 나서 안 돔 (사실 돌기도 함)
- Panic
  - Throws Error… 근데 컴파일 단계에서 안 돌아버림

그래도…

- 포인터는 없다!
- 오류나면 오류를 알려준다~
  - 심지어 상세하다!

그놈의 Ownership

```
let s1 = String::from("hello");
let s2 = s1;

println!("{}, world!", s1);
```

```
let mut s = String::from("hello");

let r1 = &mut s;
let r2 = &mut s;

println!("{}, {}", r1, r2);
```

```
let mut s = String::from("hello");

let r1 = &s; // no problem
let r2 = &s; // no problem
let r3 = &mut s; // BIG PROBLEM

println!("{}, {}, and {}", r1, r2, r3);
```

그놈의 Lifetime

```
fn main() {
    let reference_to_nothing = dangle();
}

fn dangle() -> &String {
    let s = String::from("hello");

    &s
}
```

```
fn no_dangle() -> String {
    let s = String::from("hello");

    s
}
```

장점

- Java같은 하급 언어와 다르게 Garbage Collector가 없다
- C같은 이상한 언어와 다르게 포인터로 터질 일 없다
- Java보다 빠르다
- C보다 빨리 쓸 수 있다 (포인터가 없어서 뇌가 절여질 일이 없음)
- Java마냥 Runtime에 컴파일을 하지 않는다
- C처럼 컴파일때 불친절하지 않다

단점

- 그게 없는 이유는 손으로 써야 해서다…
- 너무… 너무… 어… 어렵… 나?
  - 알려주긴 잘 알려주는데…
  - 대가리 박치기는 너무 어렵다!

찍먹 어케함?

- 새싹교실 Rust 오픈도 하고 신청도 받아요
  - 필수 능력: C / C++ or Java / 자료구조 / 알고리즘
  - 진행 예정: Rust로 알고리즘 풀기~
- https://doc.rust-lang.org/book/
- https://doc.rust-lang.org/rust-by-example/index.html
- Rustup docs --book 치면 나온다

Q&A

Thank You