Rust language

https://youtu.be/EI5FLIpaMSw

Rust laguage

- 모질라 리서치에서 개발한 범용 프로그래밍 언어
  - 안전하고, 병렬적이며, 실용적인 언어로 디자인
- 모질라 정책에 따라, Rust는 전적으로 오픈 소스로 개발되고 있으며, 커뮤니티로부터 피드백을 받고 있다.

최근 버전	1.54.0[1]
최근 버전 출시일	2021년 7월 29일 (12일 전)

- 카고(cargo)는 러스트의 빌드 시스템이다. 러스트는 cpu 병행처리 및 메모리 자원 누수 방지 등의 언어 철학 바탕 위에 설계된 취지에 맞게, 안전성으로 제작된 프로그램은 빌드 과정에서 사전에 불안정한 결과를 방지할 수 있다는 빌드시스템으로까지 이어지는 완전한 안정성을 실현하고 있다.

Rust laguage

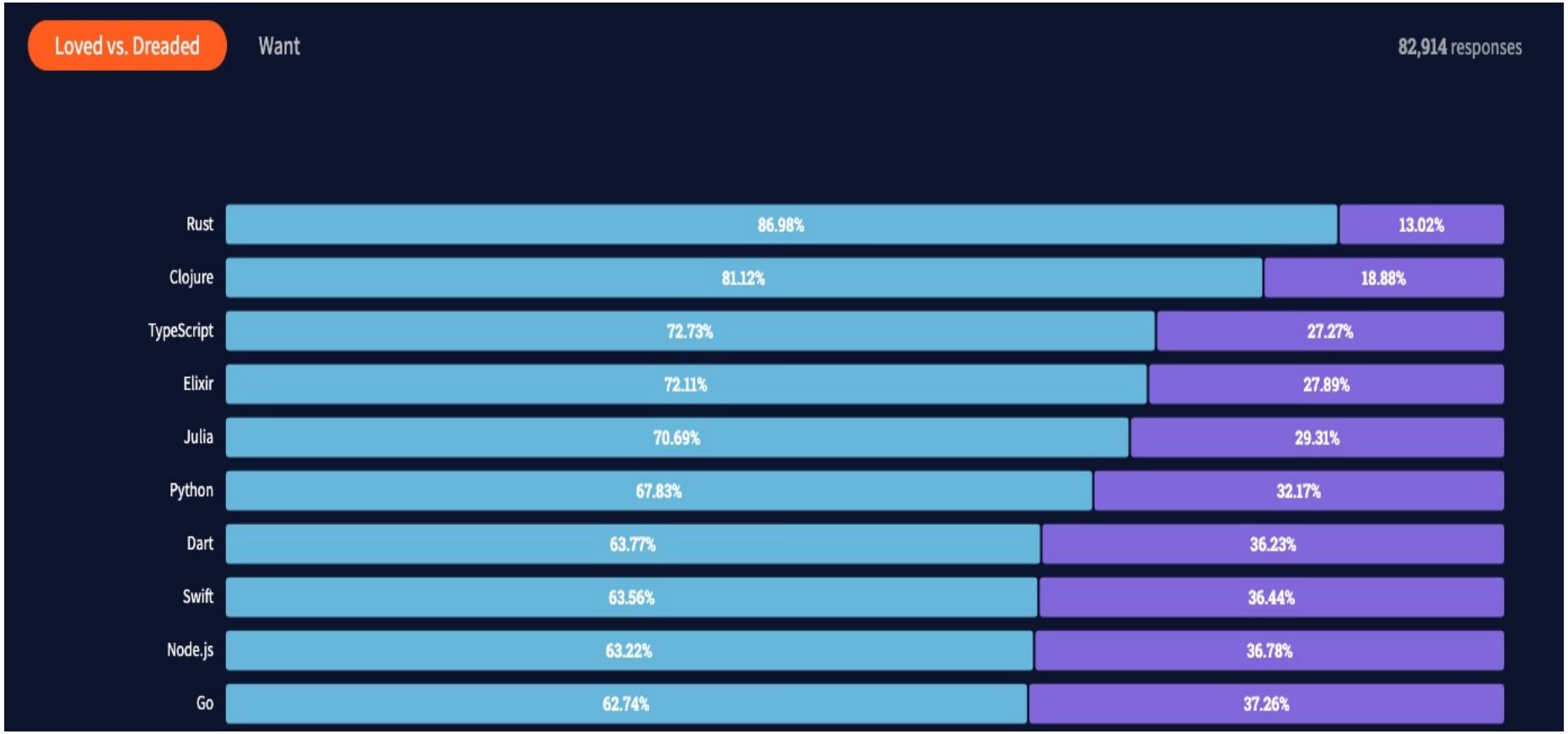
- 러스트는 인터넷에서 실행되는 서버 및 클라이언트 프로그램을 개발하는데 적합한 언어를 목표로 설계되었다. 이 목표에 따라 러스트는 안전성과 병행 프로그래밍, 그리고 메모리 관리의 직접 제어에 초점을 맞추고 있다. 또한 성능 면에서는 C++와 비슷한 수준을 목표로 하고 있다.

- 러스트의 문법은 중괄호로 코드 블록을 구분하고, if, else, while 등의 키워드를 사용하는 등 C 및 C++와 유사한 모양을 하고 있다. 그러나 러스트와 C/C++는 의미상으로는 크게 다른 문법을 갖고 있다.

- 러스트는 메모리 오류를 발생시키지 않도록 설계되었다. 러스트는 널 포인터나 초기화되지 않은 포인터가 존재하지 않도록 강제하고 있다. 모든 변수는 초기값을 가지고 할당되며, 해제된 포인터에 접근하는 코드는 컴파일러가 미리 감지하여 컴파일 오류를 일으킨다.

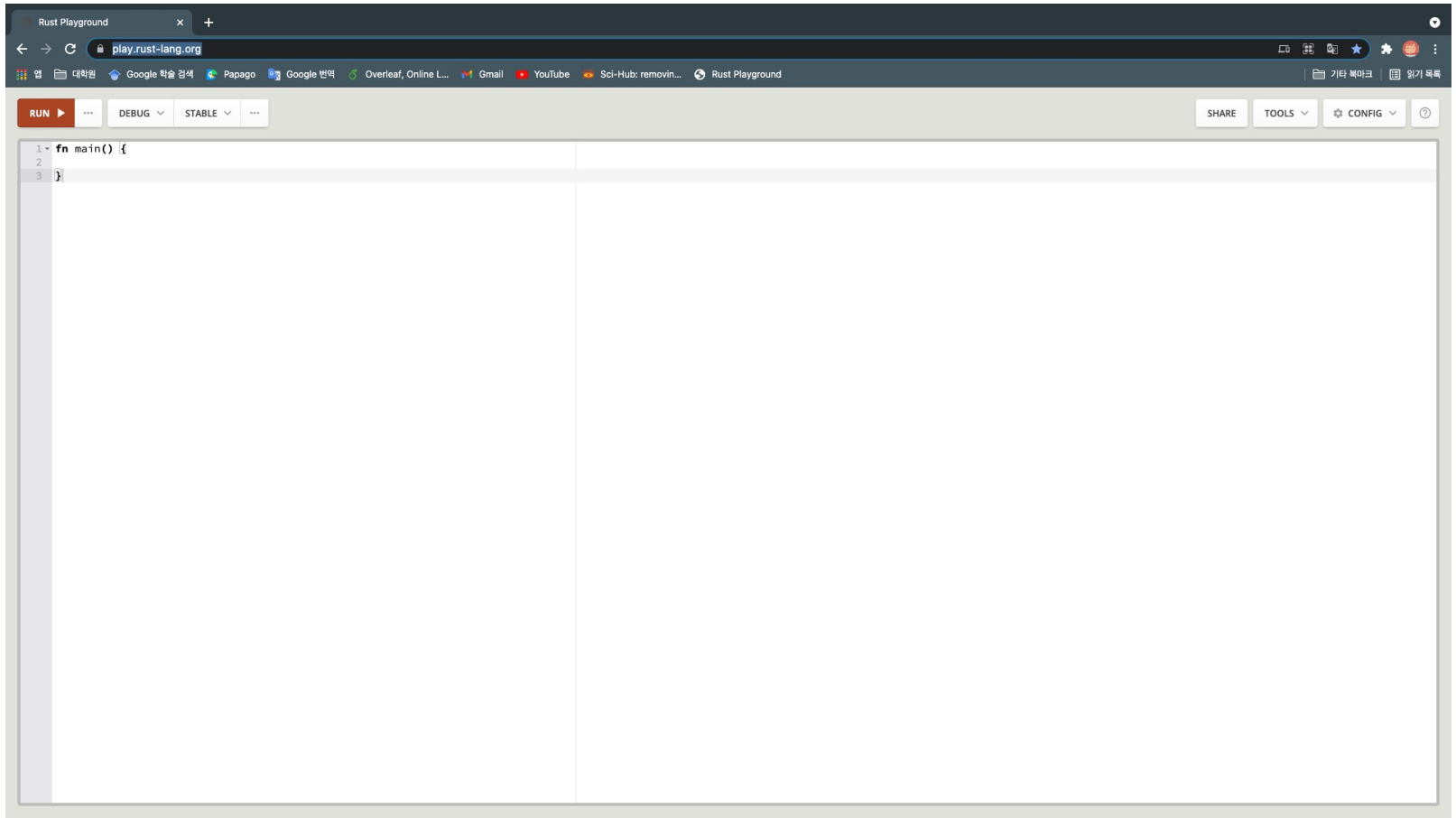
Rust language – stackoverflow 설문조사

https://insights.stackoverflow.com/survey/2021/#top-paying-technologies-programming-scripting-and-markup-languages



Rust playground

- https://play.rust-lang.org/



Rust 설치 및 사용

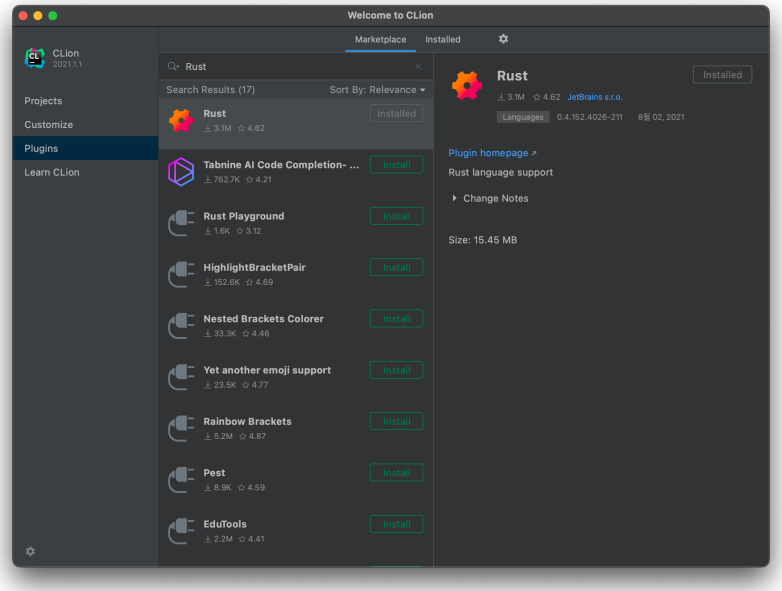
- Linux / MacOS
  - 터미널에서 $ curl https://sh.rustup.rs -sSf | sh
  - Rust is installed now. Great! 가 뜨면 설치 완료

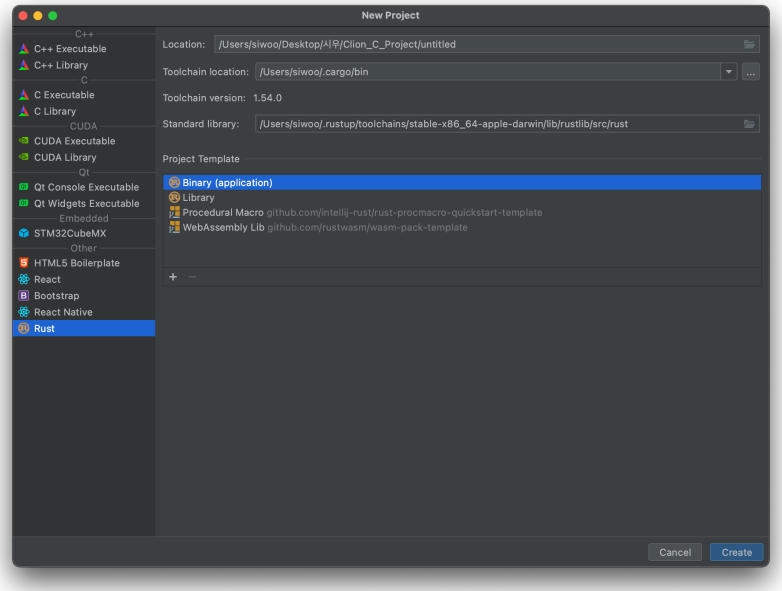
```
(base) siwoo@Siwooui-MacBookPro ~ % rustc --version

rustc 1.54.0 (a178d0322 2021-07-26)
```

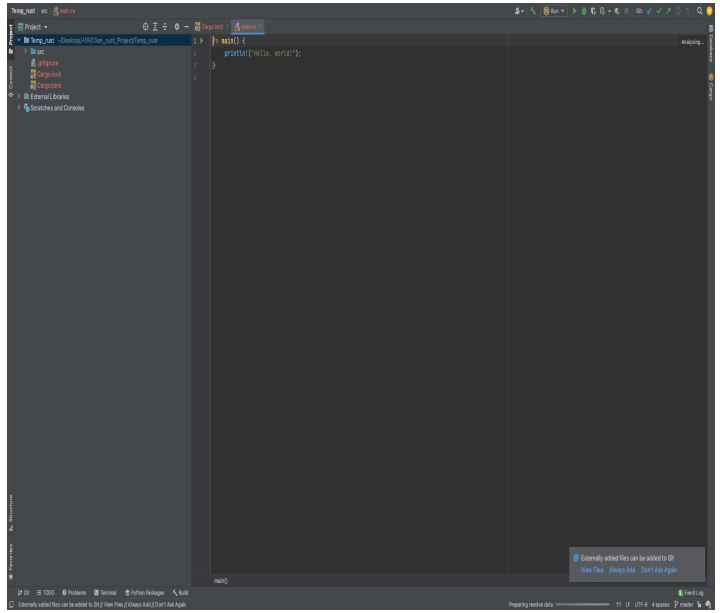
- Windows
  - https://forge.rust-lang.org/infra/other-installation-methods.html 해당 설치 지침에 따라서 설치 진행

Rust 설치 및 사용

- Clion 에서 Rust 프로젝트 생성
  - .rs 확장자 명을 사용







Rust language – main 함수 / 출력

- fn 키워드는 새로운 함수의 선언을 뜻함.
- println!() 는 화면에 출력하는 매크로
  - 러스트는 !를 통해 매크로를 호출하고 있음을 표시
  - println() 은 함수 호출 / println!() 매크로 호출
  - println!()은 출력 후 줄바꾸기를 함. 줄바꾸기를 안하려면 print!() 사용

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

Rust language - 변수

- let 키워드는 지역 변수를 나타냄.
  - let 키워드를 사용한 변수는 기본적으로 변하지 않음.

- mut를 붙여줌으로써 재할당할 수 있는 변수로 지정할 수 있음

```
fn main() {
    let x = 5;
    println!("The value of x is: {}", x);
    x = 6;
    println!("The value of x is: {}", x);
}
```

```
fn main() {
    let mut x = 5;
    println!("The value of x is: {}", x);
    x = 6;
    println!("The value of x is: {}", x);
}
```

Rust language – 데이터 타입

- 데이터 타입
  - 정수형

Length	Signed	Unsigned
8-bit	i8	u8
16-bit	i16	u16
32-bit	i32	u32
64-bit	i64	u64
arch	isize	usize

  - 부동 소수점 타입 : f32 / f64
  - Boolean 타입 : true / false
  - 문자 타입 : char 형은 ‘’ 작은 따옴표 String 은 “” 큰 따옴표를 사용함
    - Rust에서는 유니코드를 사용하여 아스키 코드와의 차이점이 있음.

Rust language – 튜플과 배열

튜플

```rust
fn main() {
    let tup: (i32, f64, u8) = (500, 6.4, 1);
}
```

```rust
fn main() {
    let tup = (500, 6.4, 1);

    let (x, y, z) = tup;

    println!("The value of y is: {}", y);
}
```

```rust
fn main() {
    let x: (i32, f64, u8) = (500, 6.4, 1);

    let five_hundred = x.0;

    let six_point_four = x.1;

    let one = x.2;
}
```

배열

```rust
fn main() {
    let a = [1, 2, 3, 4, 5];
}
```

```rust
let arr: [u8; 16] = [0; 16];
```

```rust
fn main() {
    let a = [1, 2, 3, 4, 5];

    let first = a[0];
    let second = a[1];
}
```

Rust language - 함수

void 함수 형태

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

    another_function();
}

fn another_function() {
    println!("Another function.");
}
```

반환 값을 갖는 함수

```
fn five() -> i32 {
    5
}

fn main() {
    let x = five();

    println!("The value of x is: {}", x);
}
```

함수 매개변수

```
fn main() {
    another_function(5, 6);
}

fn another_function(x: i32, y: i32) {
    println!("The value of x is: {}", x);
    println!("The value of y is: {}", y);
}
```

표현식

```
fn main() {
    let x = 5;

    let y = {
        let x = 3;
        x + 1
    };

    println!("The value of y is: {}", y);
}
```

Rust language – 조건문

```rust
fn main() {
    let number = 6;

    if number % 4 == 0 {
        println!("number is divisible by 4");
    } else if number % 3 == 0 {
        println!("number is divisible by 3");
    } else if number % 2 == 0 {
        println!("number is divisible by 2");
    } else {
        println!("number is not divisible by 4, 3, or 2");
    }
}
```

```
number is divisible by 3
```

```rust
fn main() {
    let condition = true;
    let number = if condition {
        5
    } else {
        6
    };

    println!("The value of number is: {}", number);
}
```

```
The value of number is: 5
```

Rust language - 반복문

- loop / while / for

loop

```
fn main() {
    loop {
        println!("again!");
    }
}
```

while

```
fn main() {
    let mut number = 3;

    while number != 0 {
        println!("{}!", number);

        number = number - 1;
    }

    println!("LIFTOFF!!!");
}
```

for

```
for i :i32 in 0..10{
    println!("{}", i);
}
```

```
fn main() {
    let a = [10, 20, 30, 40, 50];

    for element in a.iter() {
        println!("the value is: {}", element);
    }
}
```

Rust language – Simpira 구현 실습

- AES 알고리즘을 활용하여 Permutation 을 구현

Algorithm 1 `AESENC` (see [44])

1: **procedure** `AESENC`(state, key)
2: state ← `SubBytes`(state)
3: state ← `ShiftRows`(state)
4: state ← `MixColumns`(state)
5: state ← state ⊕ key
6: **return** state
7: **end procedure**

Algorithm 2 $F_{c,b}(x)$

1: **procedure** $F_{c,b}(x)$
2: $C \leftarrow$ `SETR_EPI32`(`0x00` $\oplus\, c \oplus b$,
3: `0x10` $\oplus\, c \oplus b$,
4: `0x20` $\oplus\, c \oplus b$,
5: `0x30` $\oplus\, c \oplus b$)
6: **return** `AESENC`(`AESENC`$(x, C), 0)$
7: **end procedure**

Algorithm 3 Simpira $(b = 1)$

1: **procedure** SIMPIRA(x_0)
2: $R \leftarrow 6$
3: **for** $c = 1, \ldots, R$ **do**
4: $x_0 \leftarrow F_{c,b}(x_0)$
5: **end for**
6: `InvMixColumns`(x_0)
7: **return** x_0
8: **end procedure**

Algorithm 4 Simpira^{-1} $(b = 1)$

1: **procedure** SIMPIRA(x_0)
2: $R \leftarrow 6$
3: `MixColumns`(x_0)
4: **for** $c = R, \ldots, 1$ **do**
5: $x_0 \leftarrow F^{-1}_{c,b}(x_0)$
6: **end for**
7: **return** x_0
8: **end procedure**

Q & A