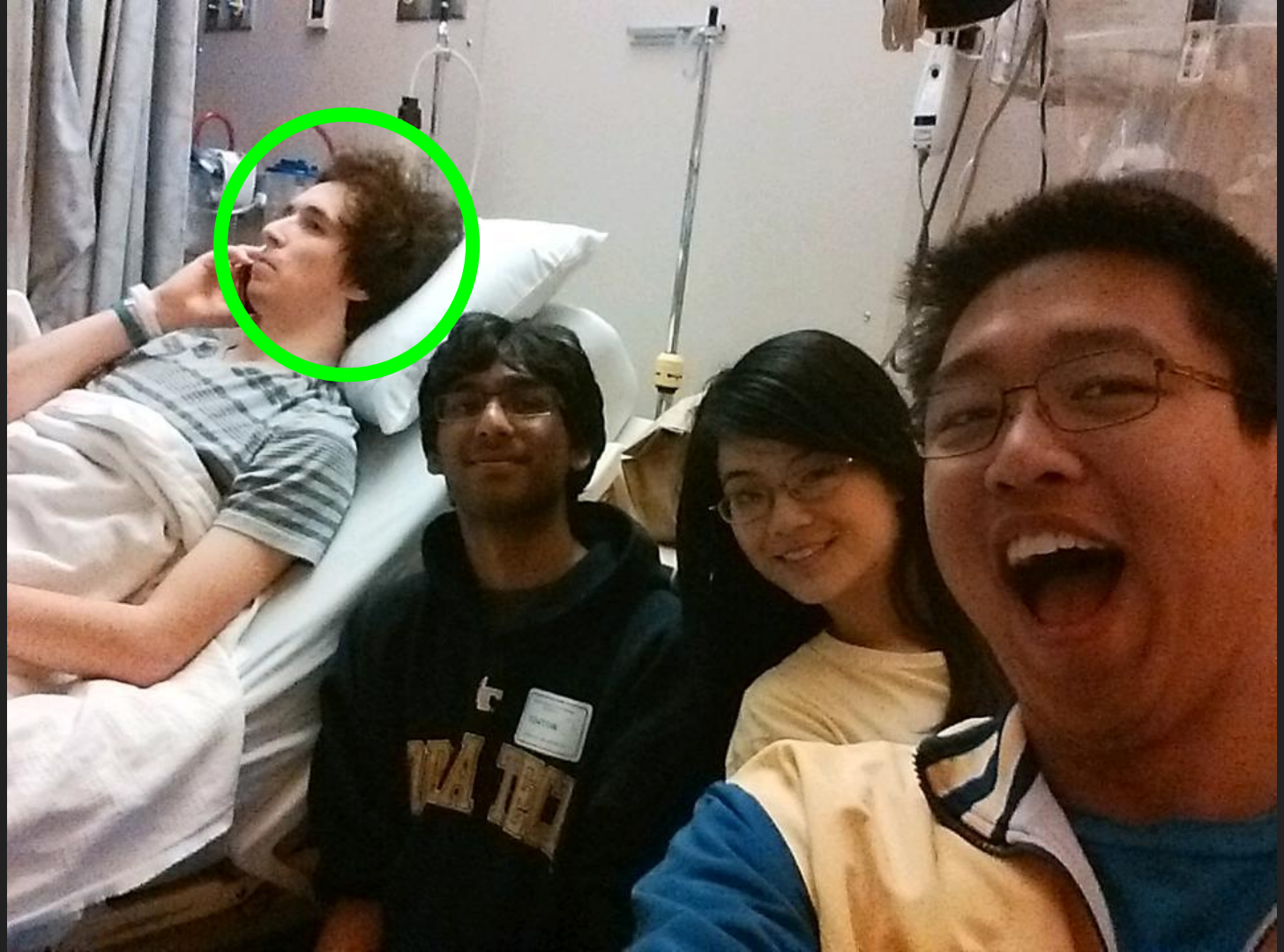
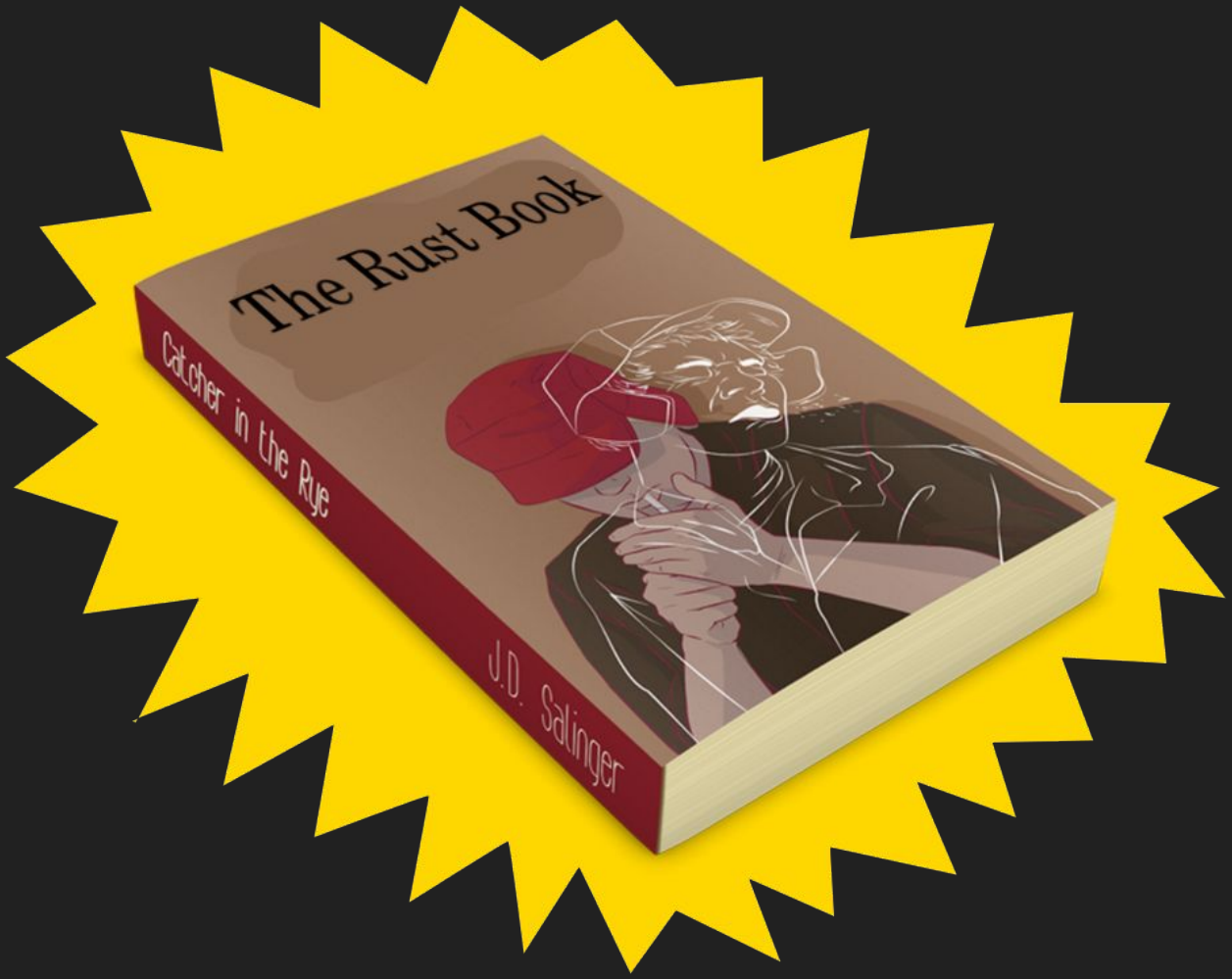
Rust Lang

the lang to end all langs

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




The Rust Book
Catcher in the Rye
J.D. Salinger

Rust is influence by

- Alef
- C#
- C++
- Cyclone
- Erlang
- Haskell
- Hermes
- Limbo
- Newsqueak
- NIL
- OCaml
- Ruby
- Scheme
- Standard
- ML
- Swift

… and has already influenced

- C# 7
- Elm
- Idris
- Swift

Swift is *heavily* influenced by Rust,
if you know Swift, Rust will go by *swift*-er

16 Good Things = 1 Great Thing Right?

Yes.

Rust is expression-based

```rust
fn gcd(a: u32, b: u32) -> u32 {
    if b == 0 {
        a
    } else {
        gcd(b, a % b)
    }
}
```

Rust has algebraic types

```rust
enum People {
    Alien,
    Lonely(String),
    Together(String, String),
}
struct World {
    people: People,
}
```

Rust has monads

```rust
fn divide(num: f64, denom: f64) -> Option<f64> {
    if denom == 0.0 {
        None
    } else {
        Some(num/denom)
    }
}

fn main() {
    match divide(5.0, 0.0) {
        None => println!("Divide by zero!"),
        Some(result) => println!("{}", result),
    };
}
```

Rust has has pattern matching

```rust
fn main() {
    let x = 'ϰ';

    match x {
        'a' ... 'j' => println!("early letter"),
        'k' ... 'z' => println!("late letter"),
        _ => println!("something else"),
    }
}
```

Rust has closures

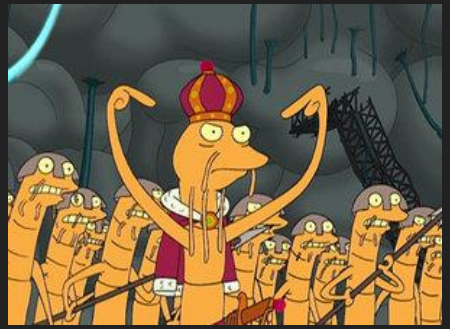
```rust
fn main() {
    let compare_os = |a, b| {
        format!("Gentoo > {} + {}", a, b)
    };

    println!("{}", compare_os("Arch", "Mint"));
}
```

...and so much more! But Rust **doesn't** have:

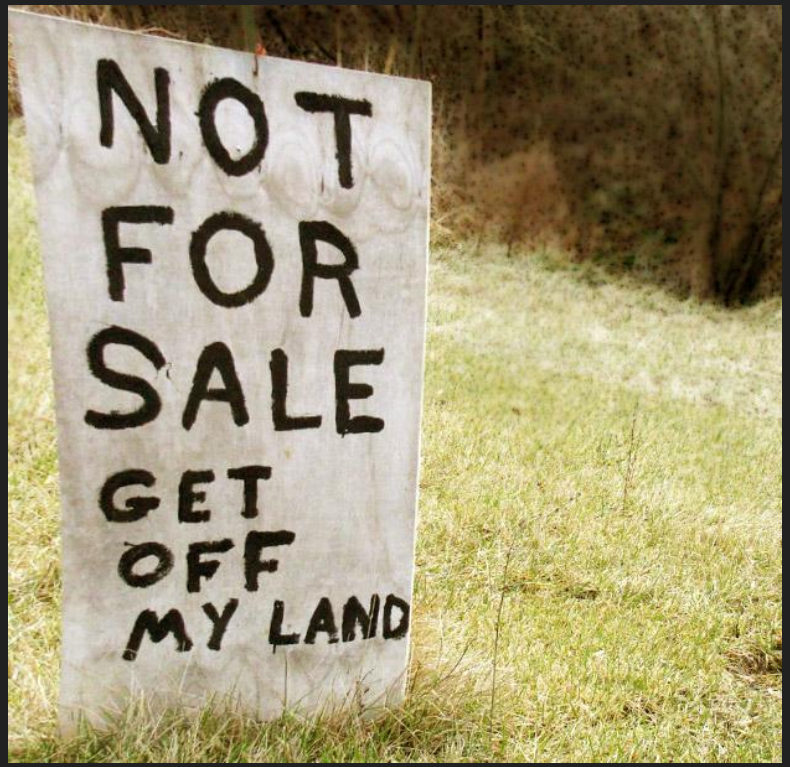
- a shared root namespace
- global (before main) mutable variables
- "accidental octal" from leading zeros
- `goto` (not even a reserved word)
- `case` fallthrough
- a == operator you can typo as = and still compile
- silent coercions between boolean and anything else
- silent coercions between enums and integers
- implementation-dependent sign for % with negative
- compilation based on textual inclusion (`#include`) or elision (`#ifdef`)

Taken from @graydon2

Where’s the Special Sauce™?



Ownership & Borrowing



The Problem

```
#include <vector>
#include <iostream>
using namespace std;
typedef vector<int>::iterator stuff_it;

int main(int, char**) {
    vector<int> stuff = vector<int>(1);

    stuff[0] = 1;

    int* yes_i_have_a_value = &stuff[0];

    stuff.push_back(2);

    for (stuff_it it = stuff.begin(); it != stuff.end(); it++) {
        cout << *it << endl;
    }

    cout << endl << *yes_i_have_a_value << endl;
}
```

The Solution: Ownership & Borrowing

```rust
fn main() {
    let mut stuff = Vec::with_capacity(1);

    stuff.push(1);

    let yes_i_have_value = &stuff[0];

    stuff.push(2);

    println!("Vector: {:?}", &stuff);
    println!("Reference: {}", &yes_i_have_value);
}
```

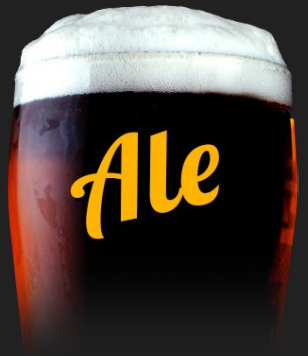
The lifetime of each reference/object is in the type!

```rust
struct VecRef<'v, T>
where T: 'v
{
    inner: Option<&'v Vec<T>>,
}

fn main() {
    let mut vec_ref: VecRef<i32> = VecRef {
        inner: None,
    };

    let vec = Vec::new();
    vec_ref.inner = Some(&vec);

    println!("{:?}", vec_ref.inner);
}
```

But Michael, does it *scale*?

S-

SC-



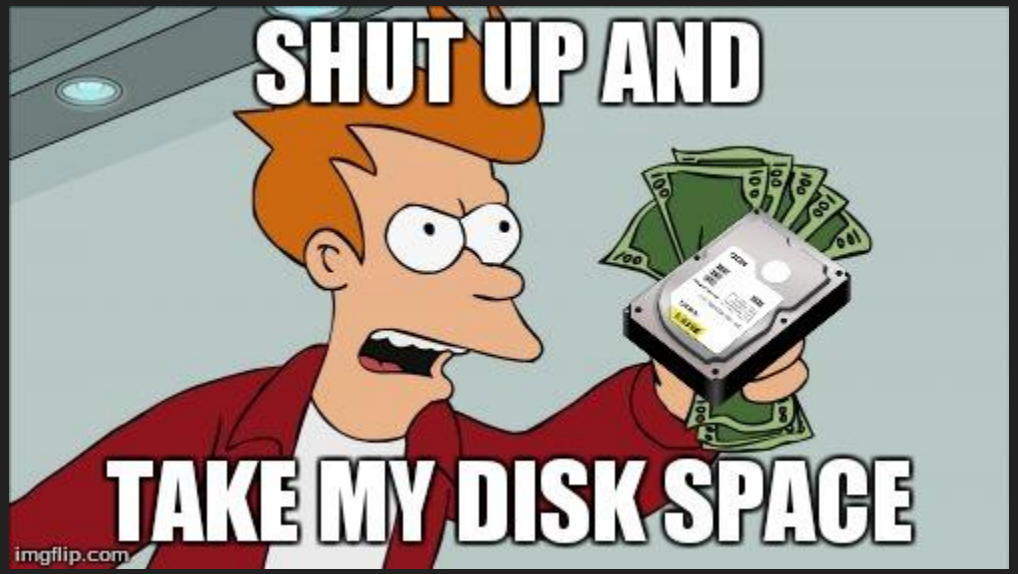
Yes!

1. Servo
2. Redox OS
3. Glium
4. Piston
5. Coreutils
6. Rustation
7. Playform
8. Iron

^ so close

Becoming a Rustacean

so when can I get the new one?



Rust 1.6.0 comes out tomorrow!