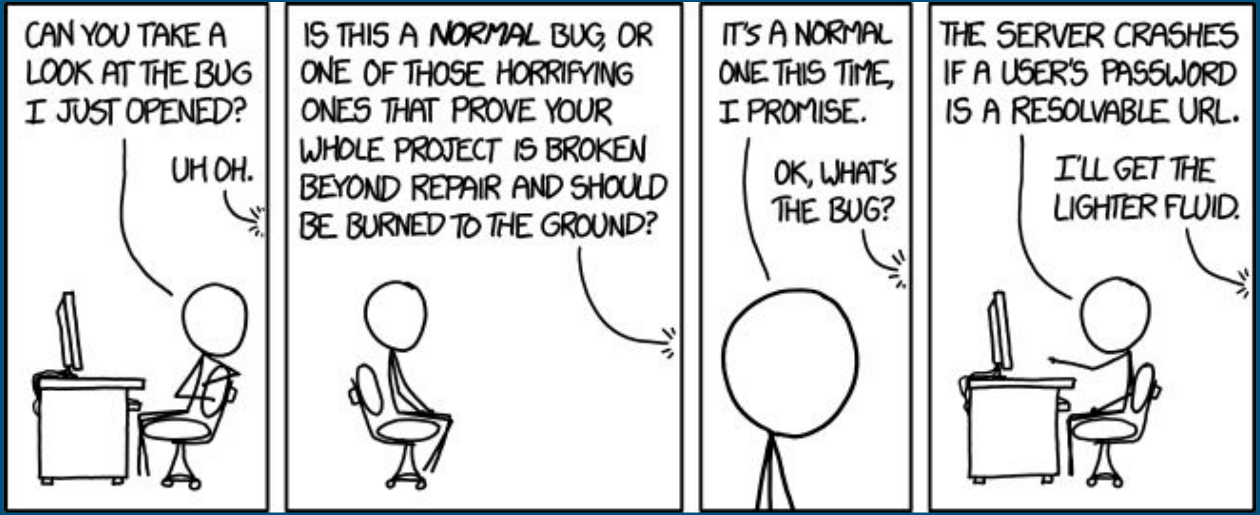
Intro to Rust

Learning to Write Safe Programs with Less Code

First: do we need another language?

- Language evolution is important
- New ideas can improve old languages (python, java, php)
- New communities can be built from the ground up

~70% of security bugs are from memory safety [1]


CAN YOU TAKE A LOOK AT THE BUG I JUST OPENED?
UH OH.
IS THIS A *NORMAL* BUG, OR ONE OF THOSE HORRIFYING ONES THAT PROVE YOUR WHOLE PROJECT IS BROKEN BEYOND REPAIR AND SHOULD BE BURNED TO THE GROUND?
IT'S A NORMAL ONE THIS TIME, I PROMISE.
OK, WHAT'S THE BUG?
THE SERVER CRASHES IF A USER'S PASSWORD IS A RESOLVABLE URL.
I'LL GET THE LIGHTER FLUID.

from https://xkcd.com/1700/

Challenges for system languages

How can we:

- Describe low level programs with high level abstractions
- Avoid a costly runtime (no garbage collection)
- Make safe memory guarantees at compile time

The answer: a better compiler

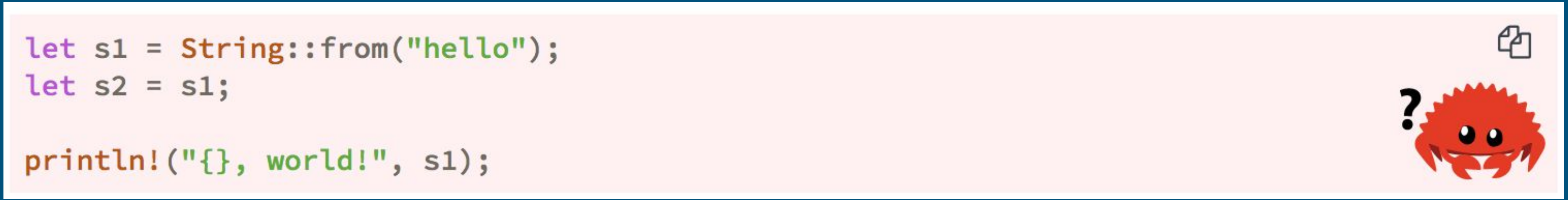
The rust compiler knows a lot about your program:

- Whenever values are assigned to variables
- Which code accesses and/or attempts to change values
- When values go out of scope

Introducing rust’s ownership model

- Every value has an owner (in foo = 5, foo is the owner)
- There can only be one
- The value can be moved to a new owner (let x = 5; let y = x;)
- Values are dropped when the owner goes out of scope

What happens next?

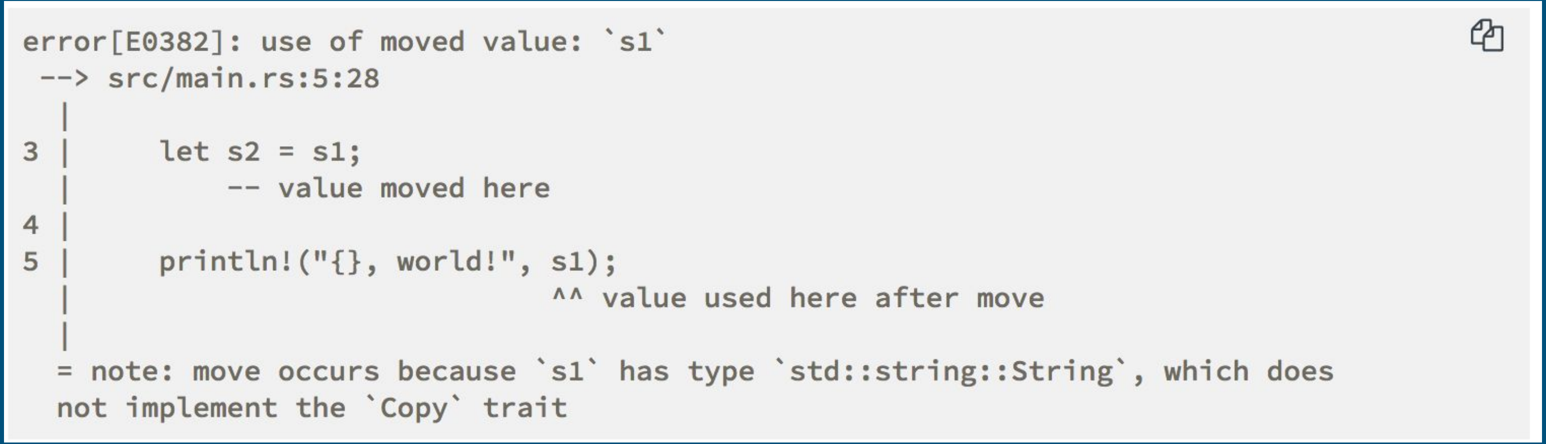


```
let s1 = String::from("hello");
let s2 = s1;

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

from https://doc.rust-lang.org/book/ch04-01-what-is-ownership.html

s1 is no longer the owner



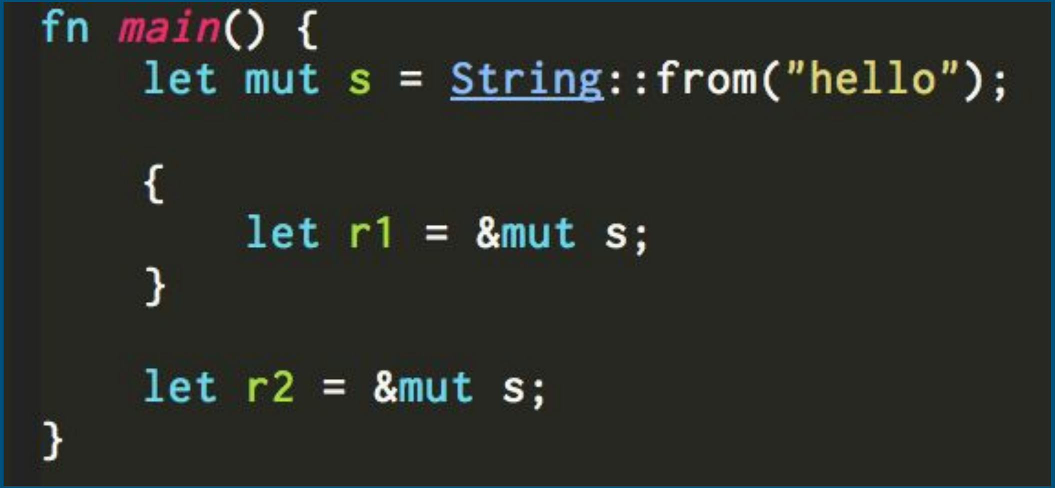
```
error[E0382]: use of moved value: `s1`
 --> src/main.rs:5:28
  |
3 |     let s2 = s1;
  |         -- value moved here
4 |
5 |     println!("{}, world!", s1);
  |                            ^^ value used here after move
  |
  = note: move occurs because `s1` has type `std::string::String`, which does
  not implement the `Copy` trait
```

from https://doc.rust-lang.org/book/ch04-01-what-is-ownership.html

Borrowing

- You can have as many immutable (read-only) references as you want
- Or you can have one mutable (read/write) reference

What happens next?



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

    {
        let r1 = &mut s;
    }

    let r2 = &mut s;
}
```

from https://doc.rust-lang.org/book/ch04-02-references-and-borrowing.html

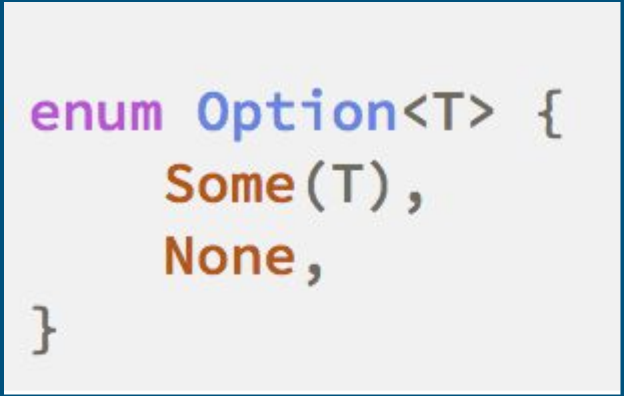
Goodbye to null

- An option type is an enum that can have Some(<value>) or None
- Any code that uses it needs to handle both possible cases
- No null references or missing code paths

"I call it my billion-dollar mistake."

-Tony Hoare, inventor of the null reference

The Option type defined

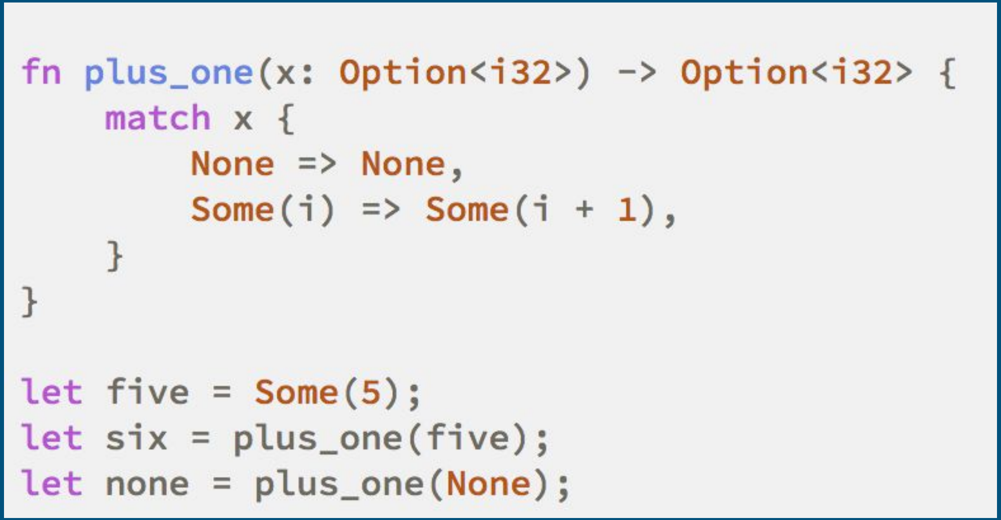


```
enum Option<T> {
    Some(T),
    None,
}
```

The type variable T can be substituted with any type to make a new type.

e.g. Option<String>

Pattern matching

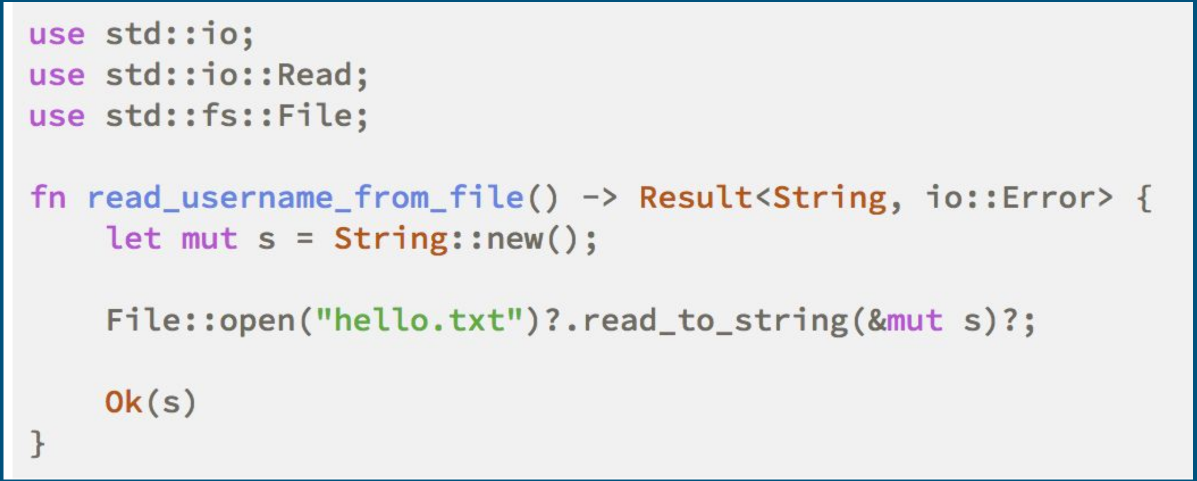


```rust
fn plus_one(x: Option<i32>) -> Option<i32> {
    match x {
        None => None,
        Some(i) => Some(i + 1),
    }
}

let five = Some(5);
let six = plus_one(five);
let none = plus_one(None);
```

from https://doc.rust-lang.org/book/ch06-02-match.html

Error propagation with ?



```
use std::io;
use std::io::Read;
use std::fs::File;

fn read_username_from_file() -> Result<String, io::Error> {
    let mut s = String::new();

    File::open("hello.txt")?.read_to_string(&mut s)?;

    Ok(s)
}
```

from https://doc.rust-lang.org/book/ch09-02-recoverable-errors-with-result.html

Zero cost abstractions

The compiler removes any overhead or unnecessary runtime lookups for features like:

- Pattern matching
- Generics
- Traits
- Iterators

Fearless concurrency in rust

- Uses channels to communicate
- Threads are OS-level threads (not green threads)
- Ownership/borrowing prevents data races and common bugs
- There's nothing* special!

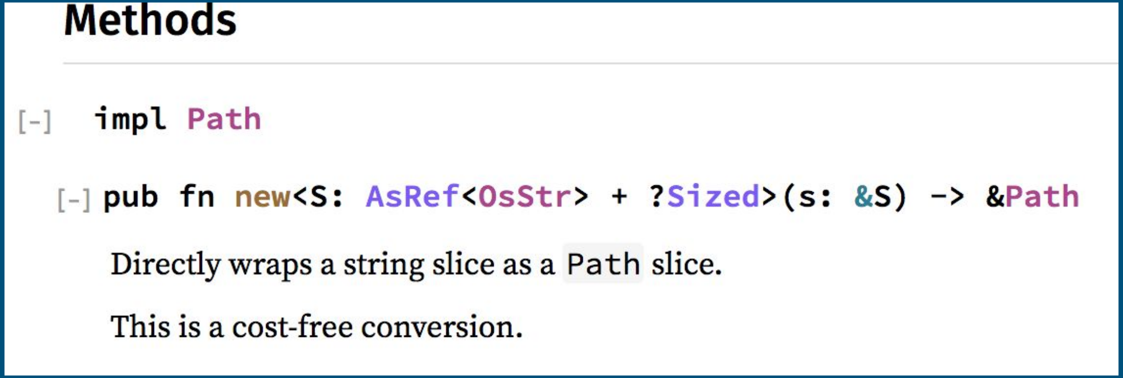
* aside from the sync/send traits and threading

API docs

- Building docs compiles your code
- API docs can’t get out of date
- Types are linked to their definitions and sources
- Provides links to dependency’s documentation
- You can even use compiler-checked doc tests!

API docs: example

How would you work with the `Path` library?



Methods

[-] impl Path

```
[-] pub fn new<S: AsRef<OsStr> + ?Sized>(s: &S) -> &Path
```

Directly wraps a string slice as a `Path` slice.

This is a cost-free conversion.

from https://doc.rust-lang.org/std/path/struct.Path.html

WebAssembly (WASM)

- No garbage collection makes it easy to compile to new targets
- Rust .wasm files are small (no runtime)
- Supported by firefox, chrome, IE, safari
- Deploy your apps straight to the web
- It’s not JavaScript

When to use rust

- Greenfield projects
- Anywhere you’d use a low level language
- Compiled extensions for python
- WASM

When not to use rust

- Great abstractions! ...still not suited to scripting
- Not the most common skillset
- Doesn’t have the libraries of java and python

How do I learn more?

- https://doc.rust-lang.org/stable/book/
- https://github.com/ericrasmussen/rust-exercises/

Links

[1] https://www.zdnet.com/article/microsoft-70-percent-of-all-security-bugs-are-memory-safety-issues/