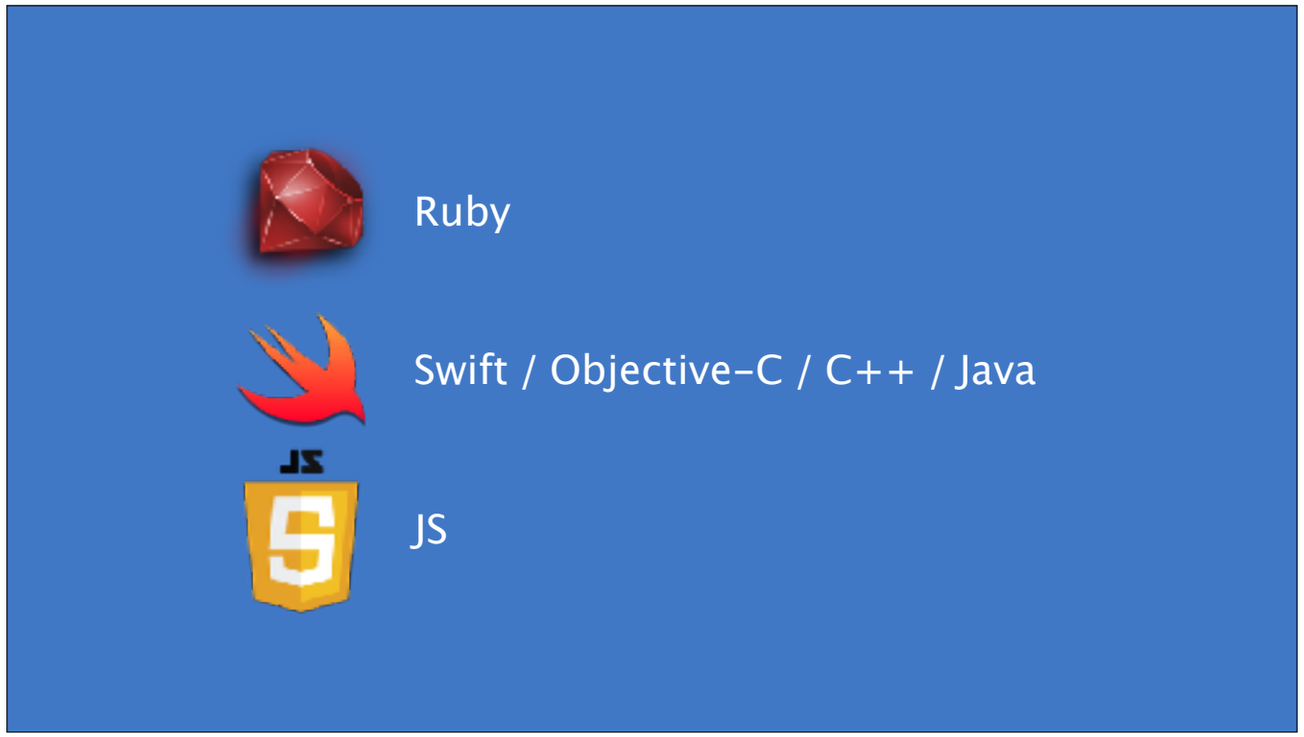
WHY I LOVE RUST

Stanko Krtalic Rusendic

github.com/Stankec
@monorkin

01 BACKGROUND



I have used these languages the most during my career. And I love each of them for different reasons

But each language has a problem associated with it

Ruby is SLOW

Swift / Objective-C / C++ / Java is HIGH RITUAL AND NOT REALLY SAFE

JS is WHAT THE F*!$K

Ruby programs versus Go

all other Ruby programs & measurements

by benchmark task performance

binary-trees

source	secs	KB	gz	cpu	cpu load
Ruby	58.72	192,132	1123	166.36	67% 61% 68% 90%
Go	39.88	361,208	688	152.12	96% 95% 96% 96%

regex-dna

source	secs	KB	gz	cpu	cpu load
Ruby	7.98	108,480	529	23.28	95% 69% 64% 64%
Go	3.89	369,380	1229	8.29	43% 53% 61% 82%

k-nucleotide

Benchmark of Ruby vs Go

```java
public class DataRaces {
    public static void main(String[] args) {
        UseCounter c = new UseCounter();
        Thread t1 = new Thread(c);
        Thread t2 = new Thread(c);
        Thread t3 = new Thread(c);
        t1.start();
        t2.start();
        t3.start();

        Counter.count = 0;

        SynchronizedUseCounter sc = new SynchronizedUseCounter();
        Thread t4 = new Thread(sc);
        Thread t5 = new Thread(sc);
        Thread t6 = new Thread(sc);
        t4.start();
        t5.start();
        t6.start();
    }
}
```

```java
class Counter {
    public static long count = 0;
}

class UseCounter implements Runnable {
    public static void increment() {
        Counter.count++;
        System.out.print(Counter.count + " ");
    }
    public void run() {
        increment();
        increment();
        increment();
    }
}

class SynchronizedUseCounter implements Runnable {
    public static synchronized void increment() {
        Counter.count++;
        System.out.print(Counter.count + " ");
    }
    public void run() {
        increment();
        increment();
        increment();
    }
}
```

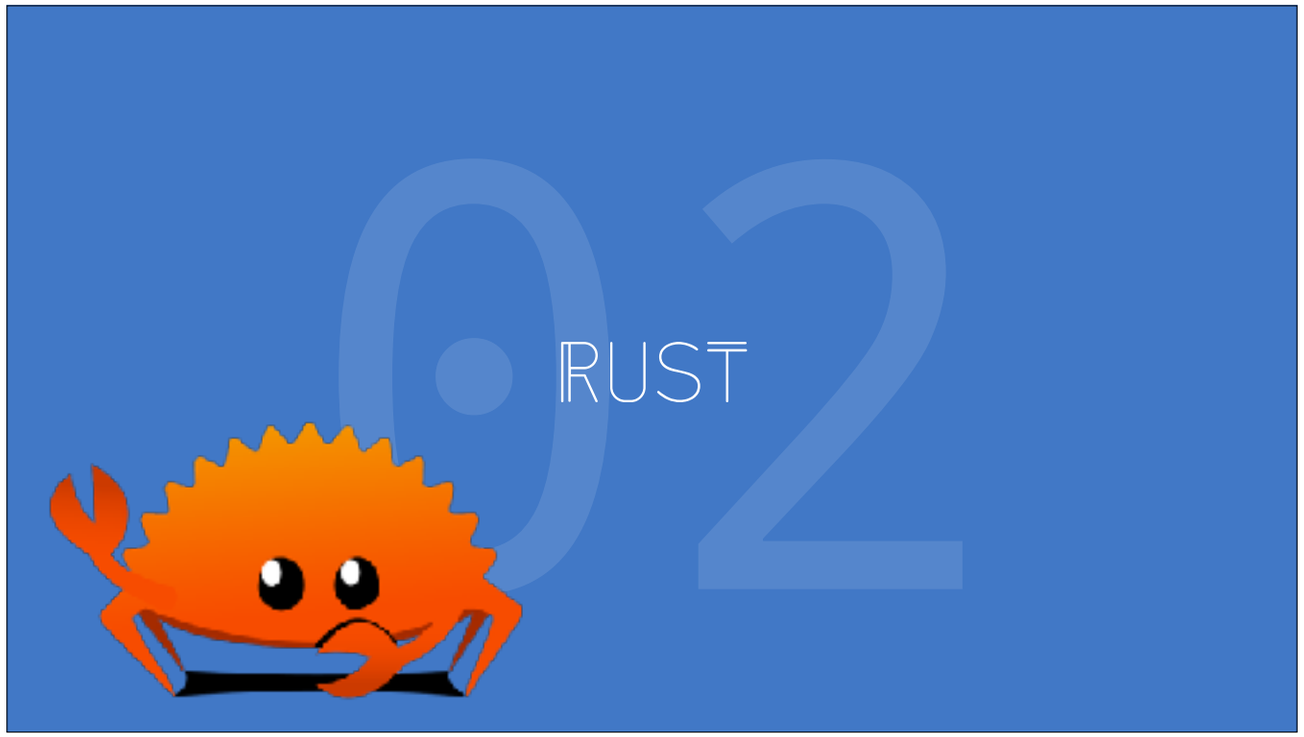
It prints something like this:

```
1 2 3 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

Data race conditions in multithreaded applications

```
>>> '3' + 2

32
>>> '3' - 2
1
>>> [] + {}
[object Object]
>>> {} + []
0
>>> NaN
NaN
>>> NaN === NaN
false
>>> typeof NaN
number
>>>
```

Fucking JavaScript!?!?!??!?!?!?!??!?!?!??!?!



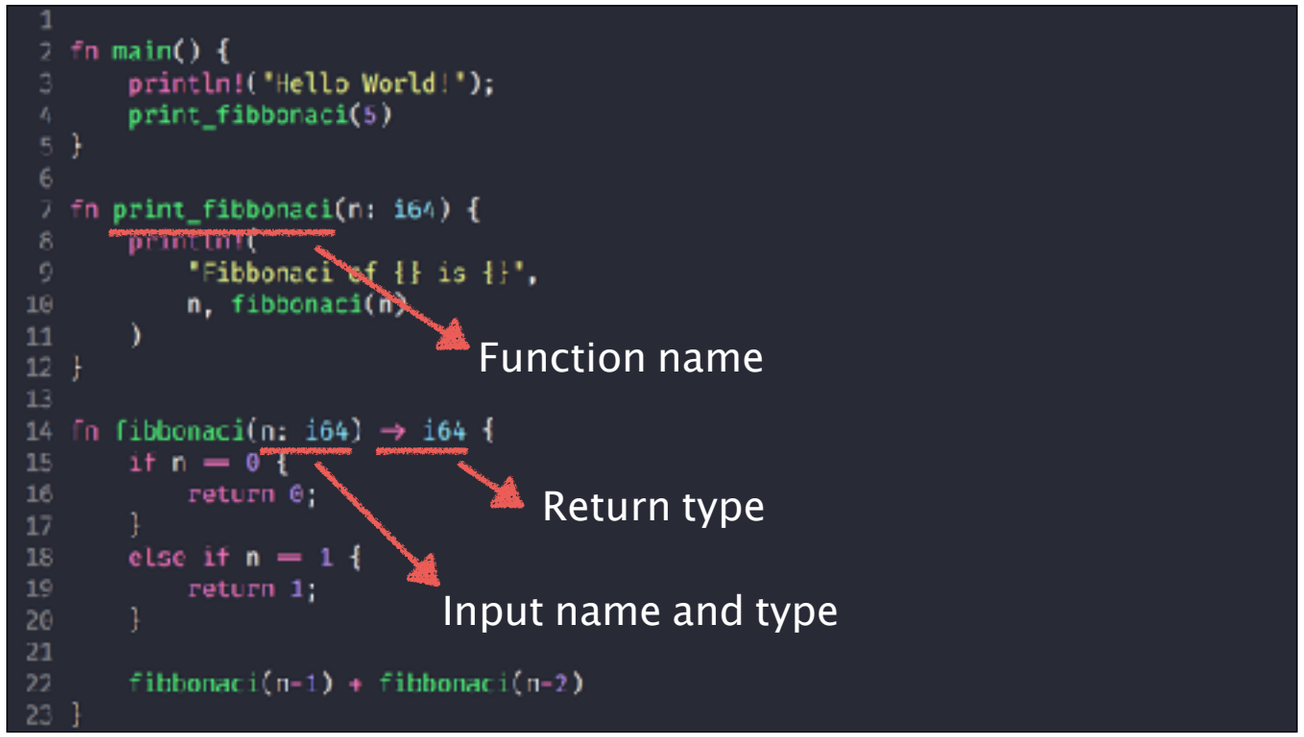
Then, one day, while I was researchng code isolation problems in VMs. I ran across a fascinating article about Rust. Rust was developed as a language that ensures safety. A program written in Rust can’t fail unexpectedly, it can’t cause a buffer overflow, it can’t try to mutate the same location in memory simultaneously.

No data races
No unexpected mutations
Safe threading
Fast

It promises all that, and more

```rust

fn main() {
    println!("Hello World!");
    print_fibbonaci(5)
}

fn print_fibbonaci(n: i64) {
    println!(
        "Fibbonaci of {} is {}",
        n, fibbonaci(n)
    )
}

fn fibbonaci(n: i64) -> i64 {
    if n == 0 {
        return 0;
    }
    else if n == 1 {
        return 1;
    }

    fibbonaci(n-1) + fibbonaci(n-2)
}
```

It has a simple and readable syntax


fn main() {
println!("Hello World!");
print_fibbonaci(5)
}
fn print_fibbonaci(n: i64) {
println!(
"Fibbonaci of {} is {}",
n, fibbonaci(n)
)
}
fn fibbonaci(n: i64) -> i64 {
if n == 0 {
return 0;
}
else if n == 1 {
return 1;
}
fibbonaci(n-1) + fibbonaci(n-2)
}
Function name
Return type
Input name and type

Thinks about developer satisfaction.

```rust

match i {
    1 => println!("Yay"),
    2 => println!("Noooooo!"),
    _ => println!("What is this!?")
}

loop {
    println!("Hi!");
}

for greting in greetings {
    println!(greeting);
}
```

There are no switch cases, no while loops and even the for loops is different.
Switch cases can cause unexpected fall throught.
While loops will potentially run indefinitely.
For loops share issues with while loops.
There fore Rust only allows infinite loops, and iteration loops.

```rust
fn print_name(user: Option<User>) {
    match user {
        Some(user) => println!("My name is {}", user),
        None => println!("No user given")
    }
}
```

There is no NULL pointer.
Null pointers are evil, they can cause unexpected states since they can be accessed as other normal values.

```
38
39
40
41
42
43
44 fn handle_book(book: Book) {
45     return_book(book);
46     borrow_book(book);
47 }
~
~
~
~
~
~
```

Now we come to the most confusing part of Rust. Ownership.

This example won’t compile. Since we passed the variable book to the `return_book` function we also gave it ownership of the book object. This is analogous to the real world. If you give a book to your friend you can’t give it to someone else.

```
39
40
41
42
43
44 fn handle_book(book: Book) {
45     let book = return_book(book);
46     borrow_book(book);
47 }
48
49
50
51
52
53
54
```

If your friend returns you the book. The you can give it further to somebody else.

```
38
39
40
41
42
43
44 fn handle_book(book: Book) {
45     return_book(&book);
46     borrow_book(&book);
47 }
48
49
50
51
52
53
```

This has no analogy in the real world, but Rust allows you to borrow a book to multiple friends if you know that they are not going to write to it.

```
38
39
40
41
42
43
44 fn harcle_book(book: Book) {
45     return_book(&book);
46     borrow_book(&book);
47 }
48
49
50
51
52
53
```

& indicates that we are passing a reference

```rust

fn print_name(user: Option<User>) -> Error<NoUserError, User> {
    match user {
        Some(user) => {
            println!("My name is {}", user);
            return Ok(user);
        },
        None => {
            println!("No user given");
            return Err(NoUserError {})
        }
    }
}

```

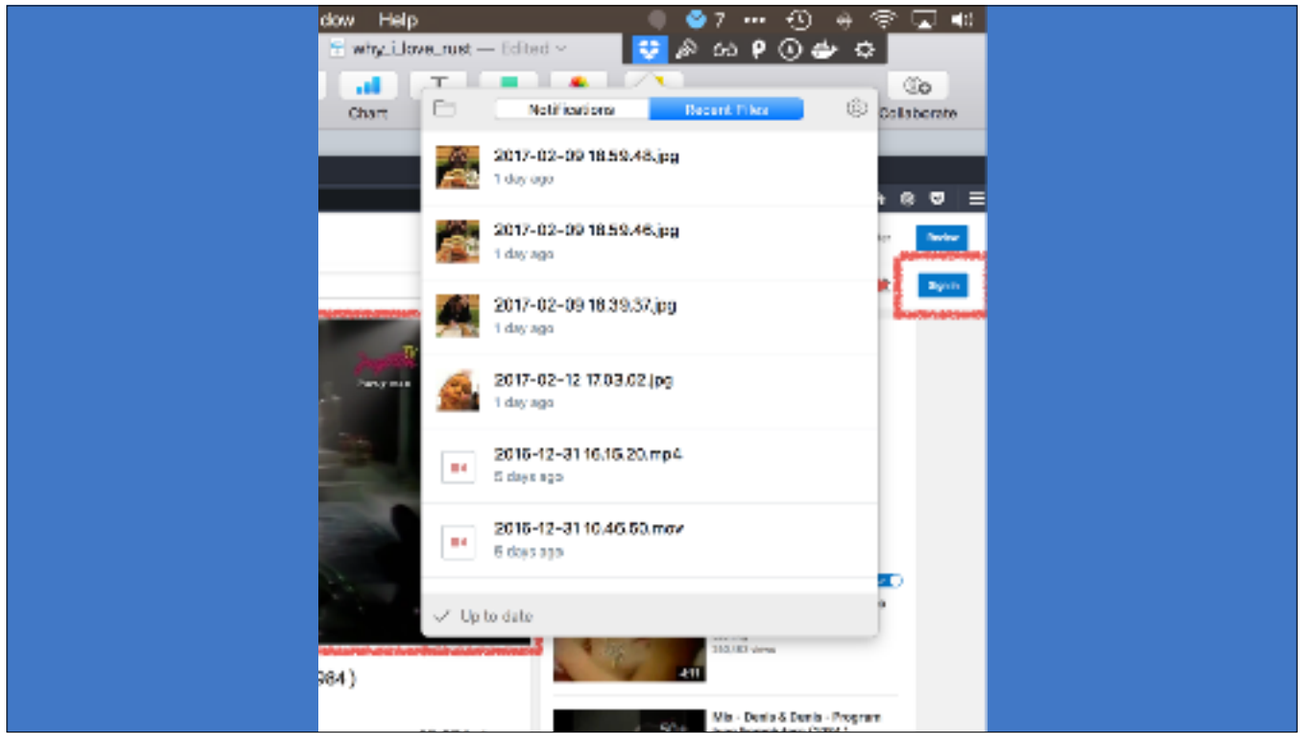
Since it’s Valentine’s day. I have to say that Rust is just like a significant other, it will force you to talk about every issue, no matter how big or small.

If a function can error, you have to explicitly state that it can fail, and what error it will return.

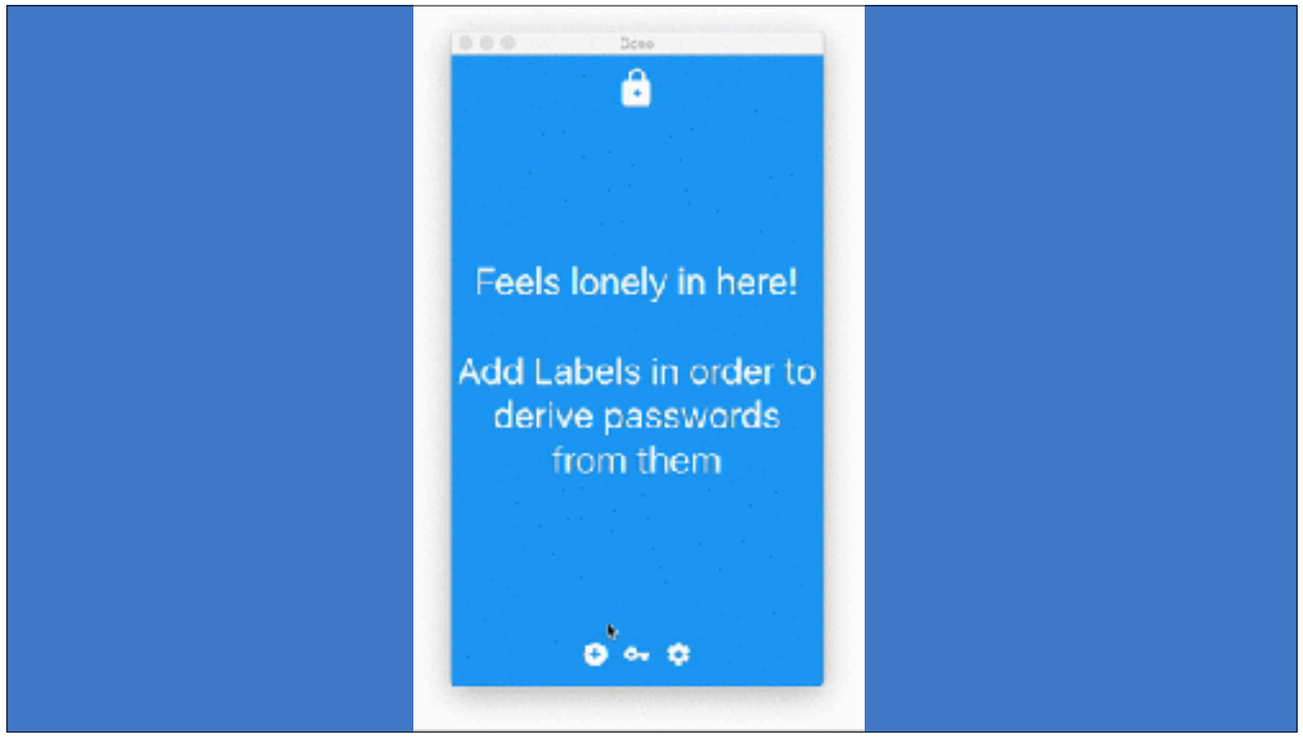
03

IN ACTION

Every time you open Firefox and go to Youtube you are using Rust.
Firefox's URL parser is written in Rust. The media player is written in Rust.
Even RedHat started adopting Rust in the development of LDAP.



Dropbox uses Rust as it’s filesystem driver - Pocket universe



I developed a few apps my self

04

LEARN IT

https://doc.rust-lang.org/stable/book/

The best resource for learning Rust at the moment is the book by Steven Klabnik