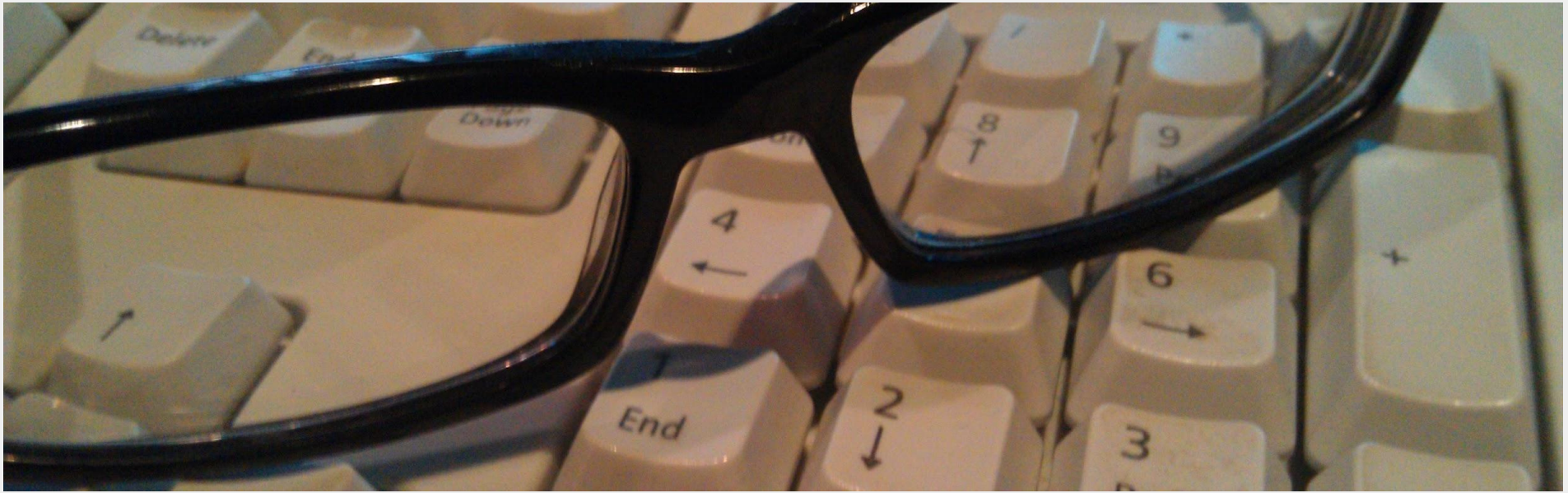
Systems Programming for Web Developers



Herman J. Radtke III

VP of Engineering Nordstromrack.com|HauteLook

When It Matters

Rust:

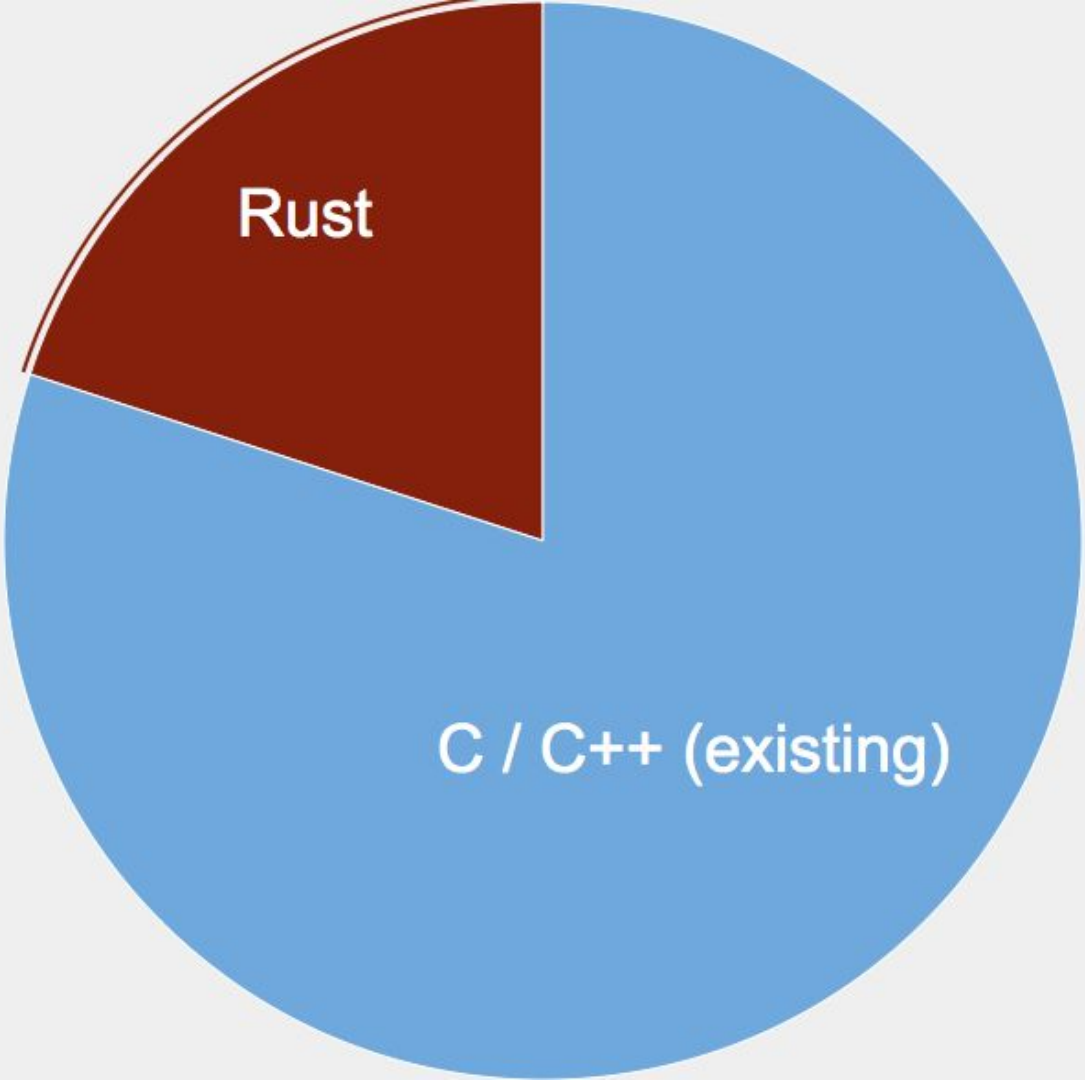
1. Memory safety without GC
2. Concurrency without data races
3. Abstraction without overhead

areweweb yet .com?

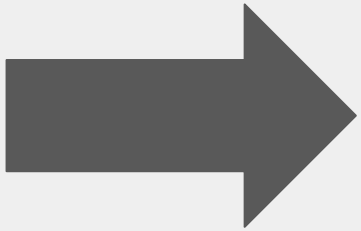
Rust + Web Developer

=

Systems Programmer


Rust
C / C++ (existing)

Ruby
Developers



Dev Ops

capistrano

puppet, chef

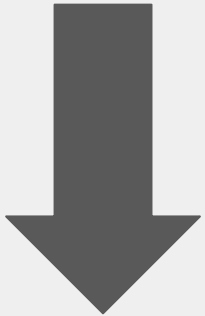
rvm, rbenv, chruby

Vagrant

Sensu


UI Developer
Node
Node
Node
Node
=
Full Stack Developer

Web Developer



Systems Programmer

Modern Web Stacks

Application Caching

HTTP Caching

Message/Job Queues

Relational, NoSQL, Search Databases

Load Balancers

VM / Containers

Knowledge

Gap

Web Developer

Barriers

Memory Management

Thread Safety

Expressiveness

Why Rust?

1. Memory safety without GC
2. Concurrency without data races
3. Abstraction without overhead

Ownership

```
fn take(v: Vec<i32>) { }

let v = vec![1, 2, 3];

take(v);

println!("v[0] is: {}", v[0]);
```

```
error: use of moved value: `v`
println!("v[0] is: {}", v[0]);
                        ^
```

```
fn take(v: Vec<i32>) { }

let v = vec![1, 2, 3];

take(v);

println!("v[0] is: {}", v[0]);
```

~~Segmentations Faults~~

~~Use after free~~

~~Memory leaks~~

Compiler as Teacher

Contribute Contribute Contribute Contribute Contribute
Contribute Contribute Contribute Contribute Contribute
Contribute Contribute Contribute Contribute Contribute
Contribute Contribute Contribute Contribute

with

confidence

Type Inference

```
let foo: i32 = 5;
```

~~let foo: i32 = 5;~~

```
let foo = 5;
```

```
let f = |x: i32| -> i32 { x };
```

~~let f = |x: i32| -> i32 { x };~~

```
let f = |x| { x };
```

```
fn silly<'a> (s: &'a str) -> &'a str { }
```

~~fn silly<'a> (s: &'a str) -> &'a str { }~~

```
fn silly (s: &str) -> &str { }
```

Optional

Correctness

```
let r = some_func();
if r.is_err() {
    return false;
}

// happy path code
```

Option<T>

```
let foo = "bar";
let c = match foo.find('b') {
    Some(c) => c,
    None => panic!("Not found")
};
```

```
let foo = "nope";
let c = match foo.find('b') {
    Some(c) => c,
    None => panic!("Not found")
};
```

```
// yolo
let foo = "bar";
let c = foo.find('b').unwrap();
```

Programmer Knows

Best

```
let sparkle_heart = unsafe {
    String::from_utf8_unchecked(
        vec![240, 159, 146, 150]
    );
};
```

Zero Cost Abstractions

Mullet Driven Dev

Ruby in the front

C in the back

```
let n = (1..10)
    .map(|i| i * 2)
    .filter(|&i| i > 5)
    .fold(0, |acc, i| acc + i);
```

```
loop {
  prompt()
    .and_then(read)
    .and_then(evaluate)
    .and_then(print)
    .unwrap_or_else( /* error */);
}
```

Generics

```
fn first<A, B>(pair: (A, B)) -> A {
    let (a, _) = pair;
    return a;
}
```

```
first((1, 2)); // returns 1
first(("a", "b")); // returns "a"
```

Monomorphization

```
first((1, 2)); // returns 1
first(("a", "b")); // returns "a"
```

```
fn first_i32(pair: (i32, i32)) -> i32 {
    let (a, _) = pair;
    return a;
}
```

```
fn first_str(pair: (&str, &str)) -> &str {
    let (a, _) = pair;
    return a;
}
```

```
fn first<A, B>(pair: (A, B)) -> A {
    let (a, _) = pair;
    return a;
}
```

Foreign Function Interface

C

Lingua Franca

Further Learning

- http://doc.rust-lang.org/book
- http://chrismorgan.info/blog/rust-ownership-the-hard-way.html
- http://rustbyexample.com
- http://www.reddit.com/r/rust
- https://users.rust-lang.org
- http://exercism.io/languages/rust
- carols10cents/rustlings

Rust LA Meetup

http://www.meetup.com/Rust-Los-Angeles/

Herman J. Radtke III

@hermanradtke

github.com/hjr3

hermanradtke.com/tags/rustlang.html

Questions?