

Rust

In It for the Long Haul

Carol (Nichols || Goulding)

@carols10cents

is.gd/rustLH

THE RUST PROGRAMMING LANGUAGE

STEVE KLABNIK AND CAROL NICHOLS,
WITH CONTRIBUTIONS FROM THE RUST COMMUNITY



- Online
- Print

Manning liveVideo



Integer 32



Rust Core Team

(yep, I'm biased)



Plan

- Railroad industry
- C
- Rust
- What the software industry can learn from the railroad industry

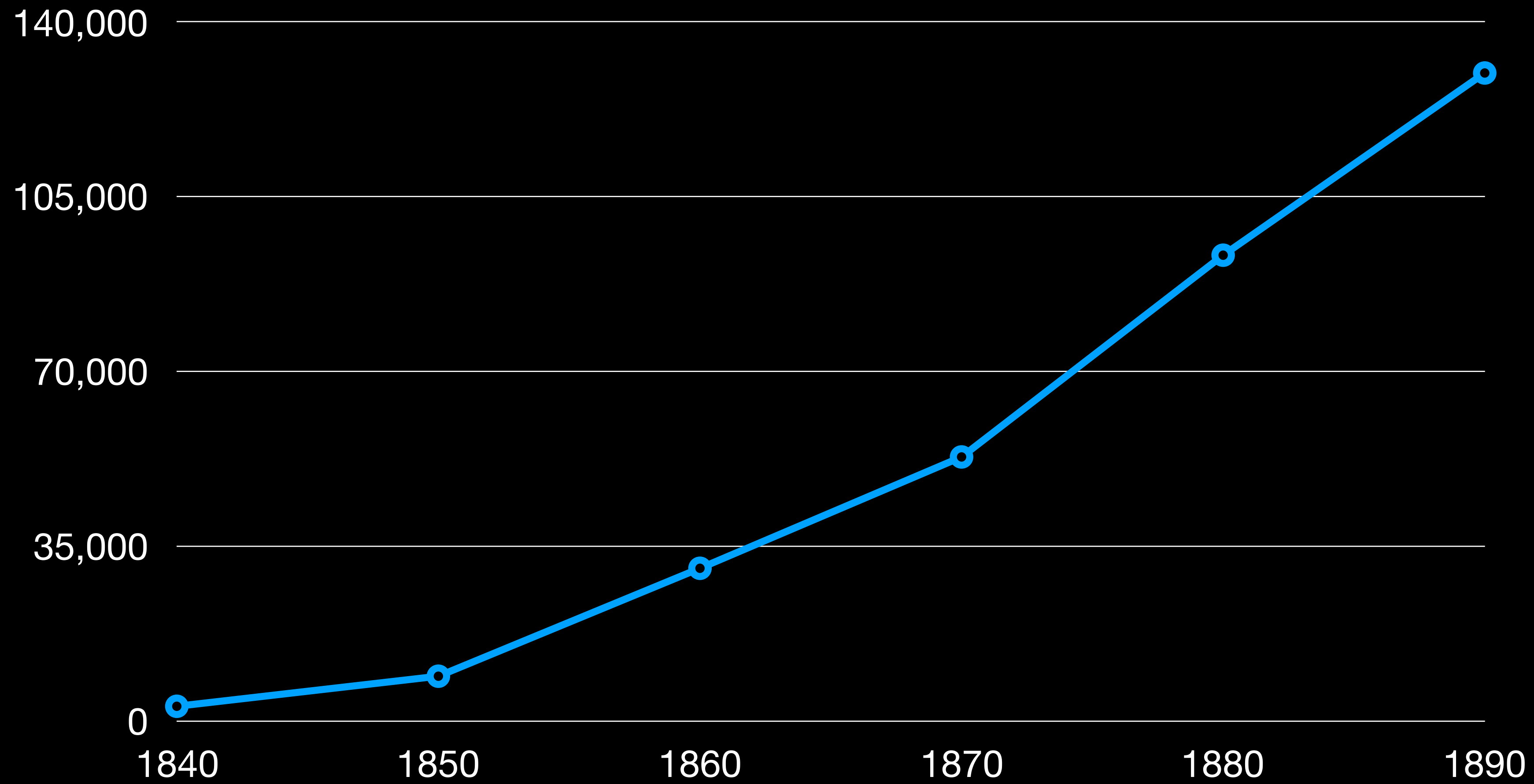
Plan

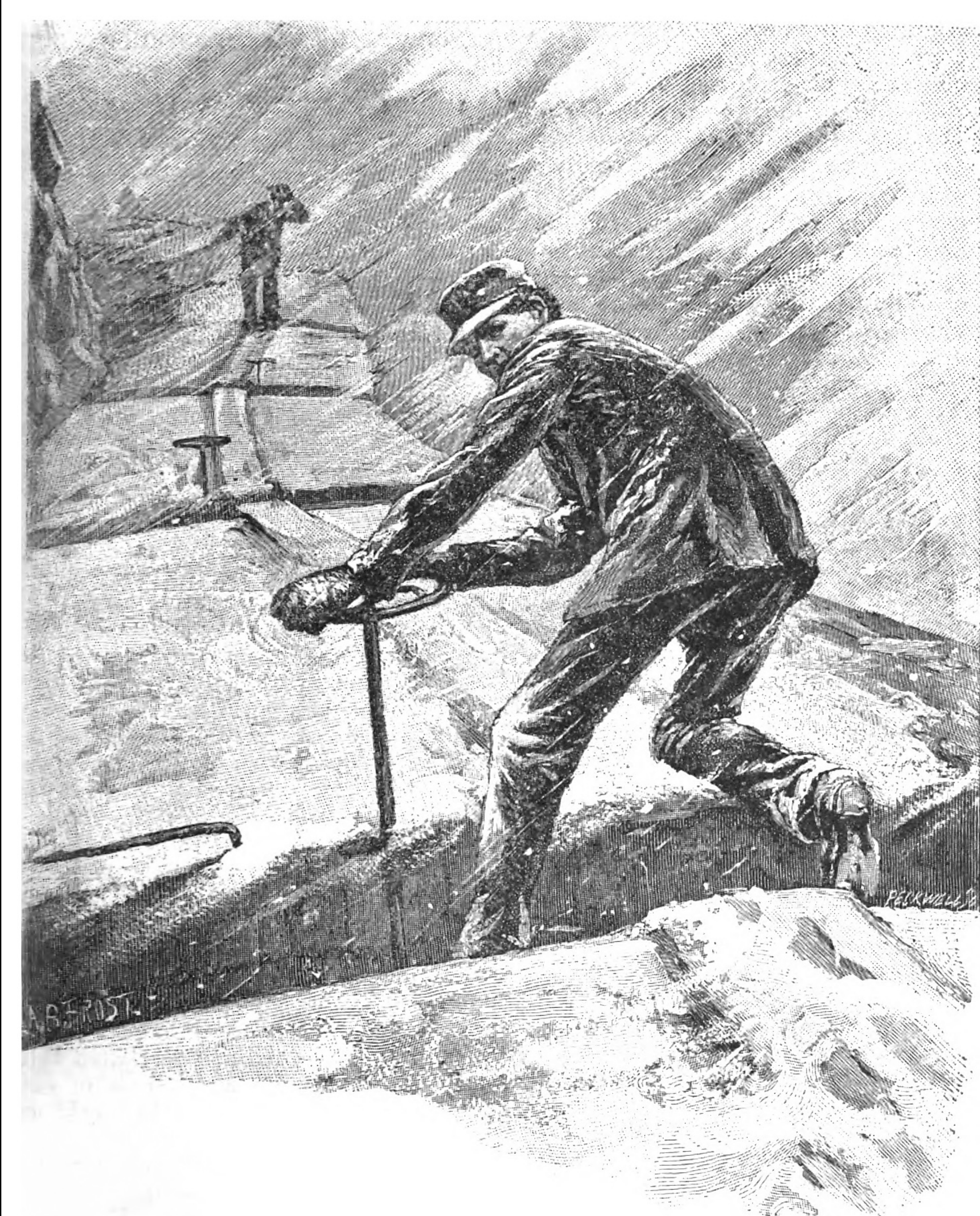
➔ Railroad industry

- C
- Rust
- What the software industry can learn from the railroad industry

1830

Miles of Rail in the US





A "PICNIC."

Brakeman

Engraving by Peckwell
Published 1890 in *The Railroad Conductor*
Public Domain in the US, [Wikipedia](#)

George Westinghouse

Photo: public domain in the US, [Wikipedia](#)



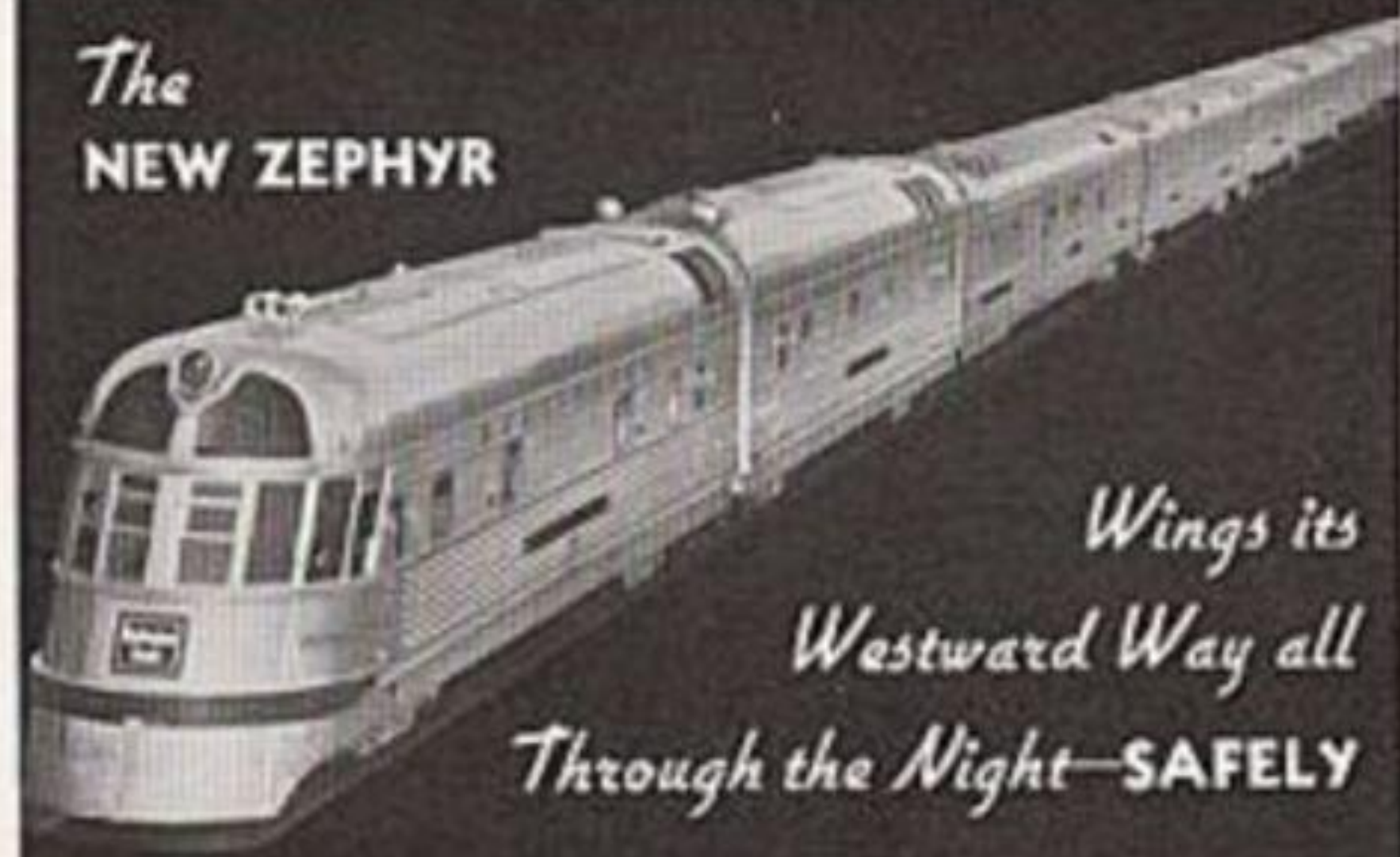
Air Brakes

- Compressed air
- Controls in the locomotive
- Air lines connecting all cars
- Apply brakes all at once
- Brakes on when there's no pressure

“Do you pretend to tell me
that you could stop trains
with air?”

– *Cornelius Vanderbilt, owner of the
New York Central Railroad*

The
NEW ZEPHYR



*Wings its
Westward Way all
Through the Night—SAFELY*

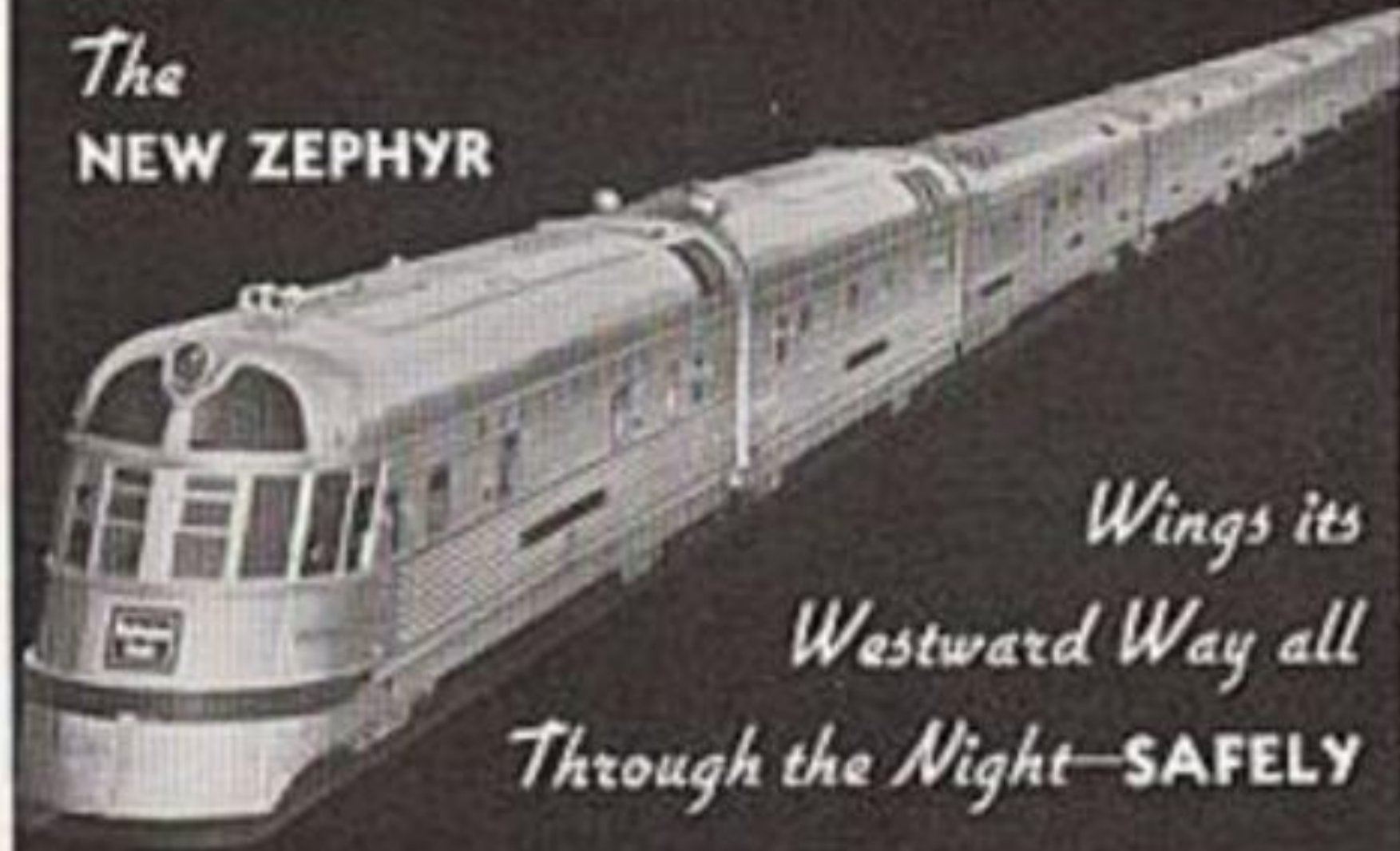
FLASHING through black darkness like a silver streak, the new Burlington Zephyr speeds over-night passengers from Chicago to Denver—SAFELY. Midway it passes its companion train on the return trip . . . Supplementing a fleet already rendering noteworthy service at various points on the Burlington Lines, these ten-car trains are of the latest streamlined construction, providing distinctive accommodations in parlor cars, coaches, and sleeping cars . . . The swift operation of these nightly carriers is safeguarded by Westinghouse Air Brakes, the improved HSC Equipment—highly efficient and effective.



WESTINGHOUSE AIR BRAKE CO.
GENERAL OFFICE AND WORKS . . . WILMERDING, PA.

Ad in 1936 *Railway Age*
Public Domain in the US, [Wikipedia](#)

The
NEW ZEPHYR



*Wings its
Westward Way all
Through the Night—SAFELY*

FLASHING through black darkness like a silver streak, the new Burlington Zephyr speeds over-night passengers from Chicago to Denver—SAFELY. Midway it passes its companion train on the return trip . . . Supplementing a fleet already rendering noteworthy service at various points on the Burlington Lines, these ten-car trains are of the latest streamlined construction, providing distinctive accommodations in parlor cars, coaches, and sleeping cars . . . **The swift operation of these nightly carriers is safeguarded by Westinghouse Air Brakes,** the improved HSC Equipment—highly efficient and effective.



WESTINGHOUSE AIR BRAKE CO.
GENERAL OFFICE AND WORKS . . . WILMERDING, PA.

“The swift operation of these nightly carriers is safeguarded by Westinghouse Air Brakes”

Ad in 1936 *Railway Age*
Public Domain in the US, [Wikipedia](#)

“They thought it was a necessity somehow,
that it occurred as a matter of course, that
some men had to be killed.”

–*L.S. Coffin, Iowa Railroad Commissioner,
Senate Hearing, 1890*

“If you are going to subject the railroad companies to this class of supervision, **then you might as well go into the character of bridges**, which is as serious a question as we have to deal with, and say that the bridges must conform to such and such standards.”

–Mr. Roberts, President of the Pennsylvania Railroad Company, Senate Hearing, 1890

1893

US Railroad

Safety Appliance Act

1900

Act fully enforced

Not perfect;

Vast improvement

Plan

- Railroad industry

➔ C

- Rust
- What the software industry can learn from the railroad industry

why C?

performance



portability



simplicity



legacy code



stability



memory

unsafety



Memory Safety Problems

- Use after free
- Double free
- Memory leaks
- Buffer overreads/overwrites
- Null pointers
- Data races

Memory Safety Problems

- Use after free
- Double free
- Memory leaks
- Buffer overreads/overwrites
- Null pointers
- Data races



“The best way to prevent these kinds of attacks is either to use a higher level language, which manages memory for you (albeit with less performance), or to be very, very, very, very careful when coding. **More careful than the entirety of the Android security team, for sure.**”

-Pulser_G2, A Demonstration of Stagefright-like Mistakes

“Around 70 percent of all the vulnerabilities in Microsoft products addressed through a security update each year are memory safety issues”

–Catalin Cimpanu reporting on a presentation by Matt Miller, MS security engineer. [ZDNet, 2019-02-11](#)

**Efforts to make
C safer**

valgrind

ASAN

UBSAN

IKOS

MISRA

Write code

THEN

make it safe

Safe-C, Checked C

C++

Plan

- Railroad industry
- C

➡ Rust

- What the software industry can learn from the railroad industry

Rust

**#1: Fixes common
memory safety
problems**


Ownership

Borrowing

```
fn main() {  
    let x = String::from("hi");  
    println!("{}", x);  
}
```

Allocates memory

```
fn main() {  
    let x = String::from("hi");  
    println!("{}", x);  
}
```



```
fn main() {  
  let x = String::from("hi");  
  println!("{}", x);  
}
```

Owner →

Allocates memory →


```
fn main() {  
  let x = String::from("hi");  
  println!("{}", x);  
}
```

Owner →

Allocates memory

Owner goes out of scope,
memory is cleaned up


```
fn main() {  
    let x = String::from("hi");  
    let y = x;  
    println!("{}", x);  
}
```

```
fn main() {  
    let x = String::from("hi");  
    let y = x;  Moves ownership  
    println!("{}", x);  
}
```

error[E0382]: borrow of moved value: `x`

```
fn main() {
```

```
    let x = String::from("hi");
```

```
    let y = x;
```

- value moved here

```
    println!("{}", x);
```

^ value borrowed

here after move

```
}
```

```
fn main() {  
    let x = String::from("hi");  
    let y = &x;  
    println!("{}", x);  
}
```

```
fn main() {  
    let x = String::from("hi");  
    let y = &x; ← Immutable borrow  
    println!("{}", x);  
}
```

```
fn main() {  
    let x = String::from("hi");  
    let y = &x;  
    println!("{}", x);  
    println!("{}", y);  
}
```



```
fn main() {  
    let y = {  
        let x = String::from("hi");  
        &x  
    };  
    println!("{}", y);  
}
```

```
fn main() {  
    let y = {  
        let x = String::from("hi");  
        &x ← Returning a reference  
    };  
    println!("{}", y);  
}
```

```
fn main() {  
    let y = {  
        let x = String::from("hi");  
        &x ← Returning a reference  
    }; ← x is cleaned up  
    println!("{}", y);  
}
```

error[E0597]: `x` does not live long enough

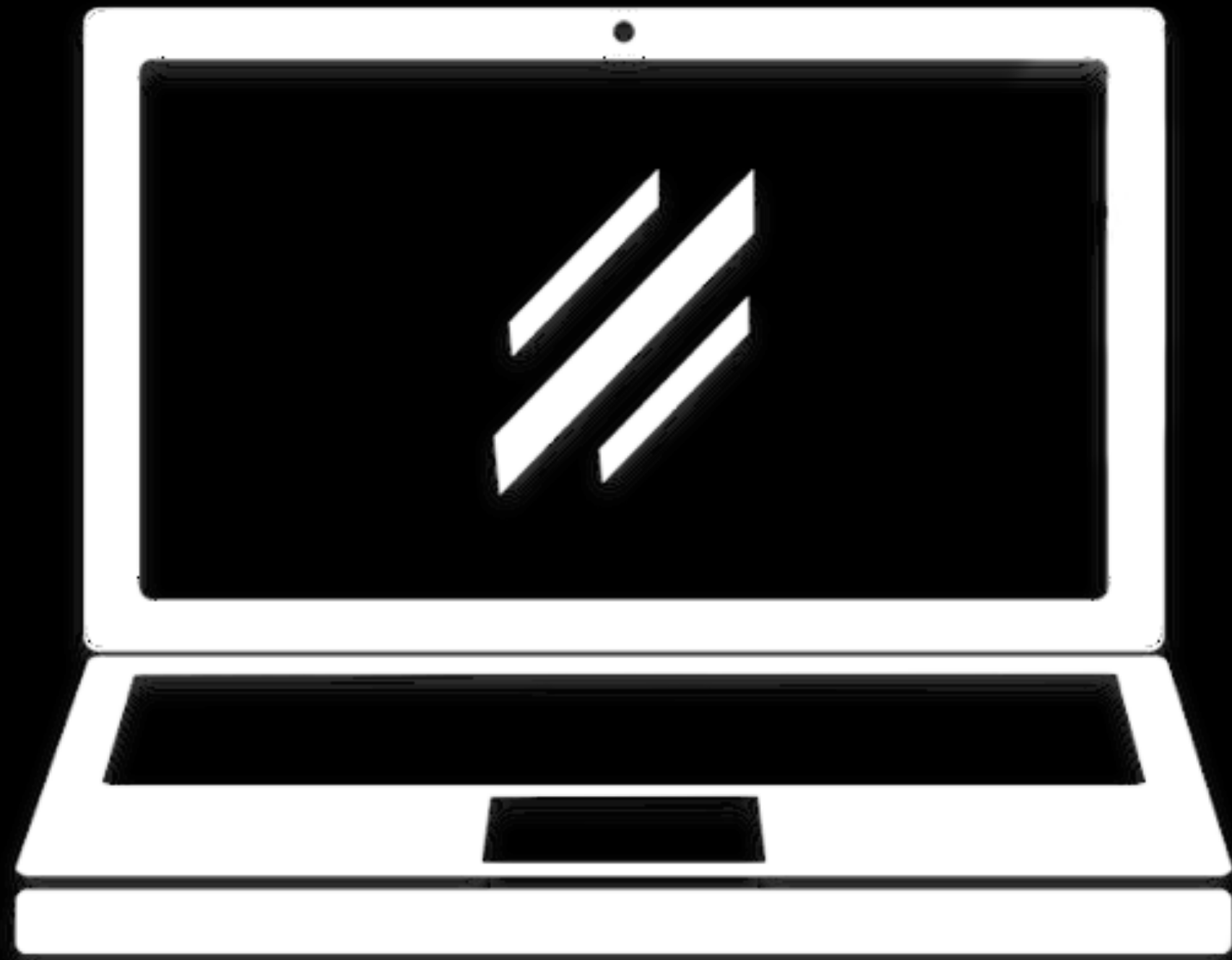
--> src/main.rs:4:9

```
|  
2 |     let y = {  
    |         - borrow later stored here  
3 |         let x = String::from("hi");  
4 |         &x  
    |         ^^ borrowed value does not live long enough  
5 |     };  
    |     - `x` dropped here while still borrowed
```

Rust Safety

- Either one mutable reference OR many immutable references
- No null, only Option
- Out-of-bounds access = at runtime, program stops
- Ownership rules apply across multiple threads

Computers are good at tedium.



⚠ Beep, boop. You forgot a semicolon in 23,982 places

Created by Dillon Arloff
from Noun Project

#2: Systems

programming is for

superhumans

everyone

unsafe

Unsafe code can...

- Dereference a raw pointer
- Call an unsafe function
- Implement unsafe traits
- Mutate global variables
- Access fields of unions



Look here for the

cause of memory

problems!



Opt OUT

Further unsafe Info

- Building on an Unsafe Foundation - Jason Orendorff, RBR 2018
- The Rustonomicon

Logic bugs

Tests

Fuzzers

memory



safety



performance



portability



simplicity



legacy code



legacy code

My "Rust out your C" Talk

stability

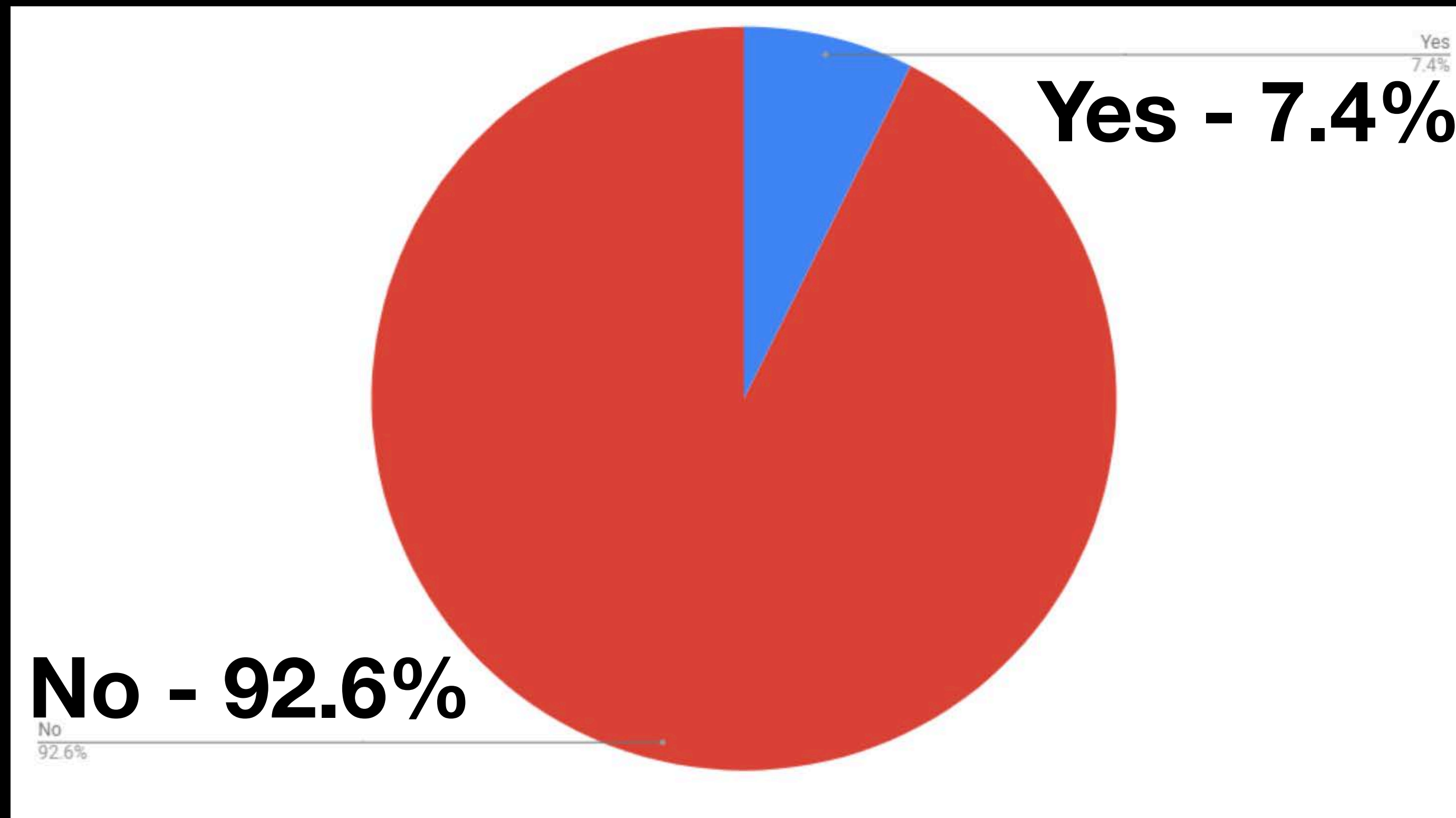


stability*



**We reserve the right to fix compiler bugs, patch safety holes, and change type inference in ways that may occasionally require new type annotations. We do not expect any of these changes to cause headaches when upgrading Rust. ([more detailed documentation](#))*

Has upgrading broken your code?



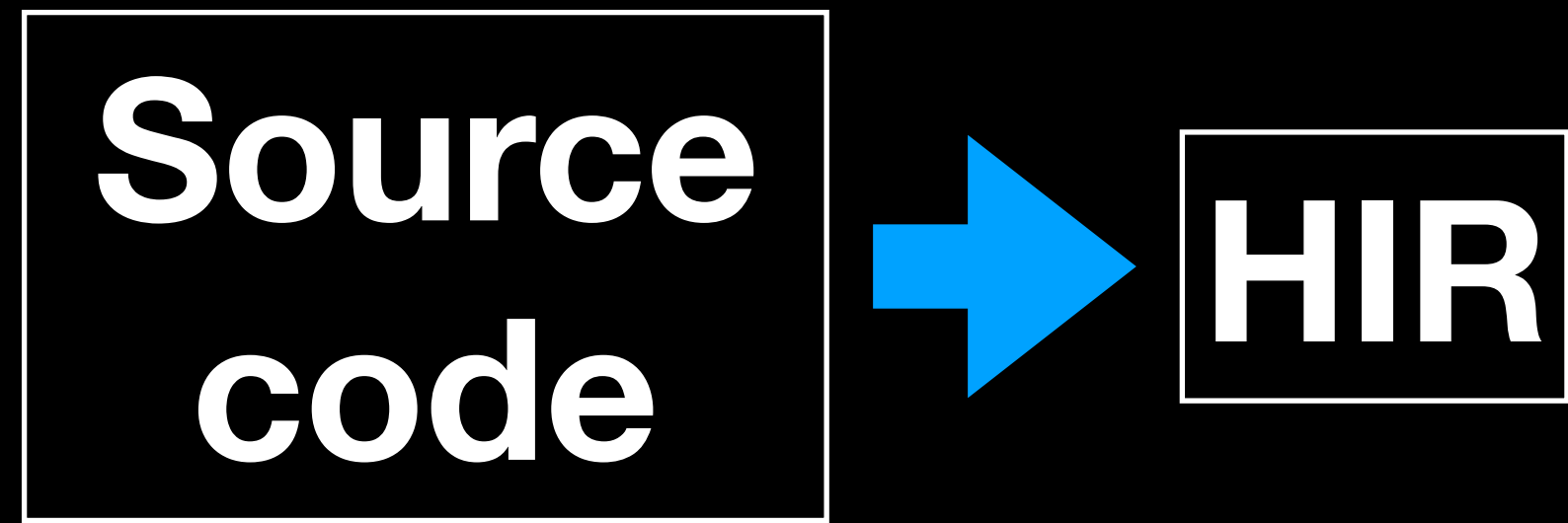
#3: stability

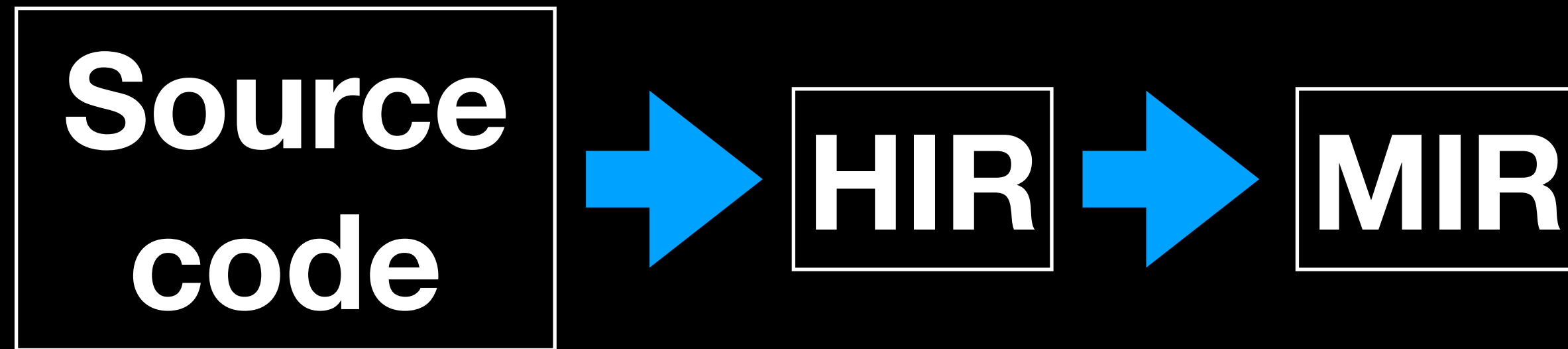
without

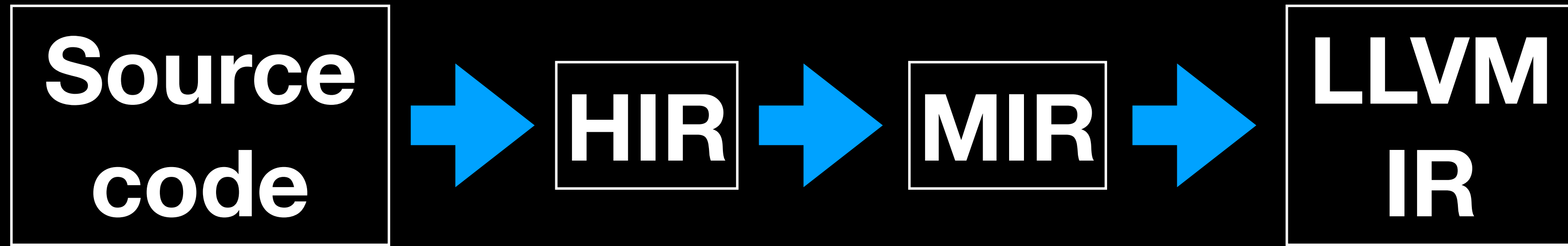
stagnation

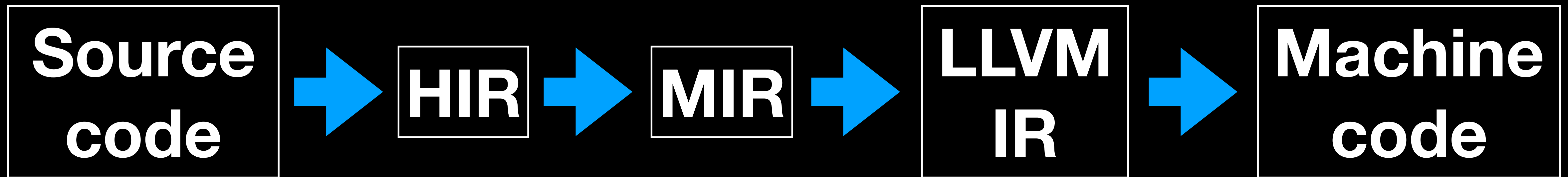
Editions

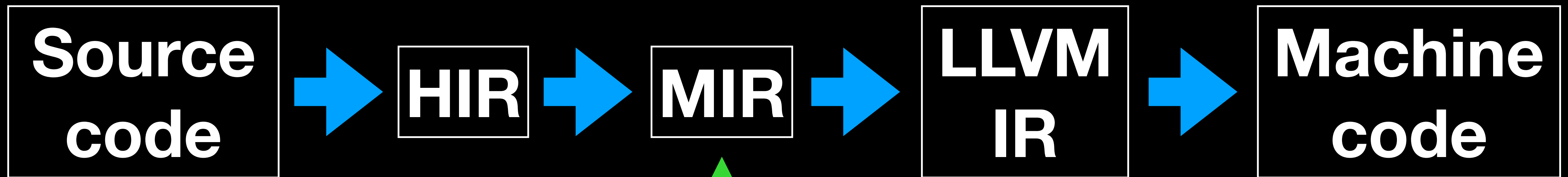
**Source
code**



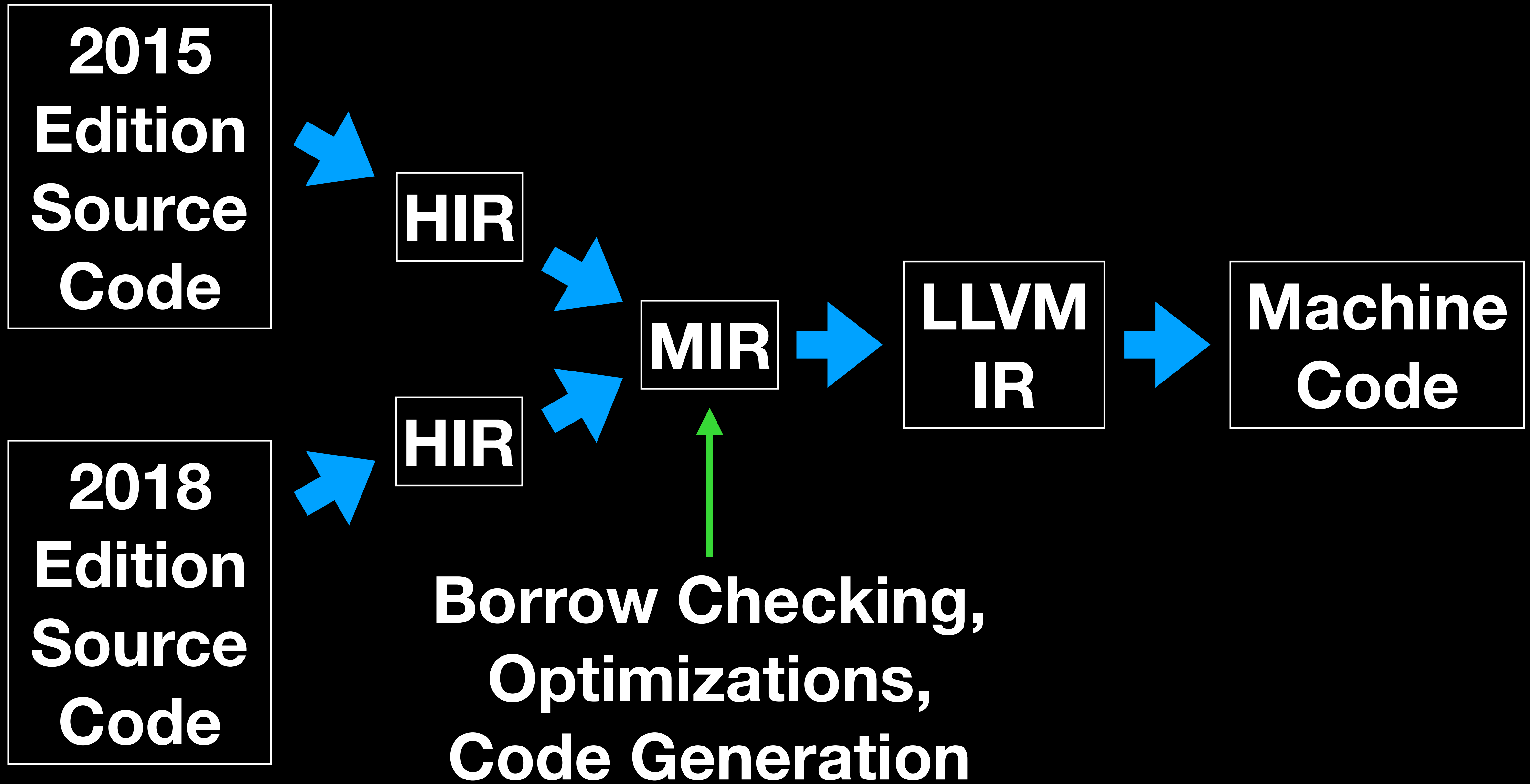








**Borrow checking,
Optimizations,
Code generation**



No ecosystem split!!!

Rust 2015
Project



Rust 2018
Library

Rust 2018
Project



Rust 2015
Library

**You pick when to
switch editions**

(never is totally fine!)

rustfix

Rust 2.0

A large, bold red 'X' is superimposed over the text 'Rust 2.0', indicating that this version of the Rust programming language is not the current or recommended one.

// TODO

- ISO/ECMA Standard
- Compiler certification
- LTS Release
- Better cargo/build system integration
- Private crate hosting
- Improved ecosystem

#4: Large

Enterprises are

using Rust

Mozilla



CSS Component

Implications of Rewriting a Browser Component in Rust

By Diane Hosfelt, 2019-02-28

CSS Component

- Security bugs since Firefox started: 69

Implications of Rewriting a Browser Component in Rust

By Diane Hosfelt, 2019-02-28

CSS Component

- Security bugs since Firefox started: 69
- Rust would have prevented: 51

Implications of Rewriting a Browser Component in Rust

By Diane Hosfelt, 2019-02-28

CSS Component

- Security bugs since Firefox started: 69
- Rust would have prevented: 51

73.9%

Implications of Rewriting a Browser Component in Rust

By Diane Hosfelt, 2019-02-28

AppAmaGooBookSoft

- ~~Apple~~
- Amazon - Firecracker
- Google - Fuchsia
- Facebook - Mononoke
- Microsoft - IoT Edge

#5: Rust

Governance

No BDFL

Teams and Working Groups

- Programming language theorists and designers
- Enterprise users
- Hobby users
- People from low-level languages
- People from high-level languages
- People from functional languages

**Decisions made
via public RFCs**

Code of Conduct

Rust's staying power

- Significant improvement in memory safety over the status quo
- More programmers can write and maintain it
- Editions enable stability without stagnation
- Large companies are depending on it
- Governance set up to endure

Plan

- Railroad industry
- C
- Rust

➔ What the software industry can learn from the railroad industry

Silicon Valley

arrogance

Juicero shows what's wrong with Silicon Valley thinking

–Christine Emba, Washington Post, 2017-04-24

**The Lyft Shuttle is pretty
much a glorified city bus —
with fewer poor people**

—Keith Spencer, Salon, 2017-06-19

**Elon Musk ‘can stick his
submarine where it hurts’
says British diver who helped
Thai cave rescue**

–Nick Allen, The Telegraph, 2018-07-14

**Trains not stopping =
people dying**

Memory unsafety???

= ???

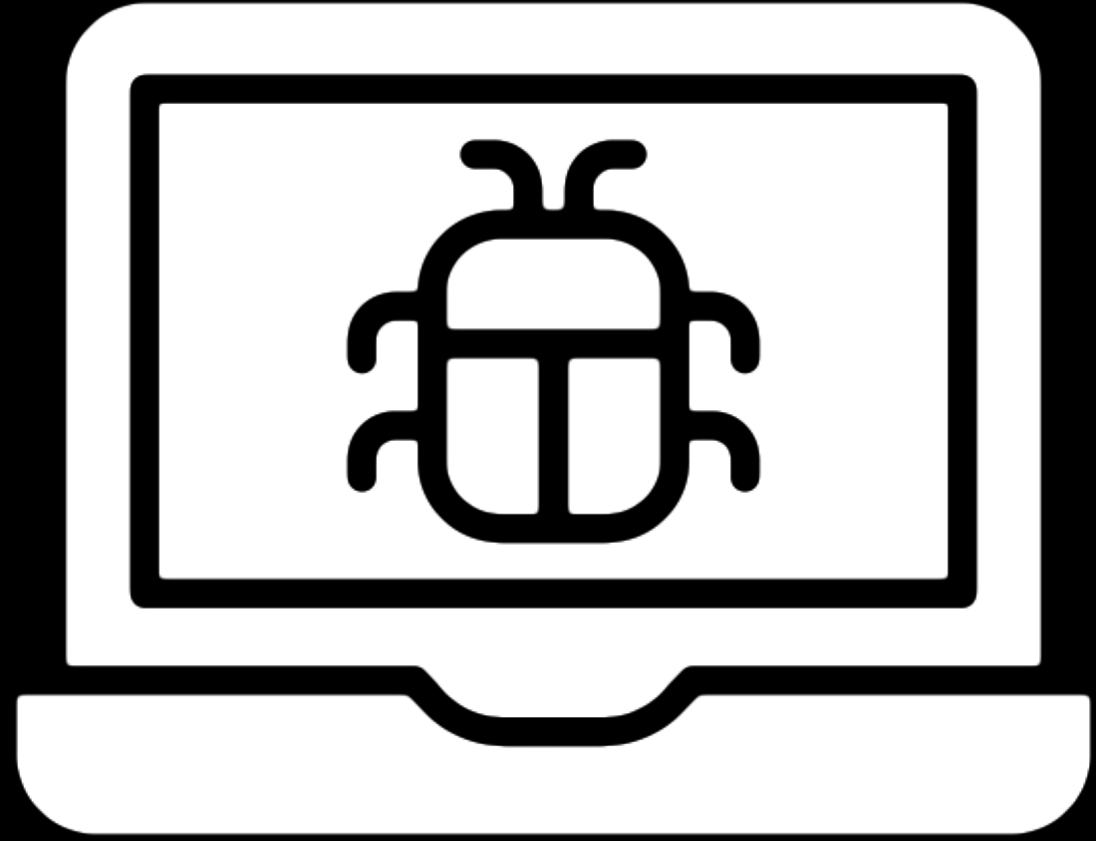
Actual Consequences

Actual Consequences

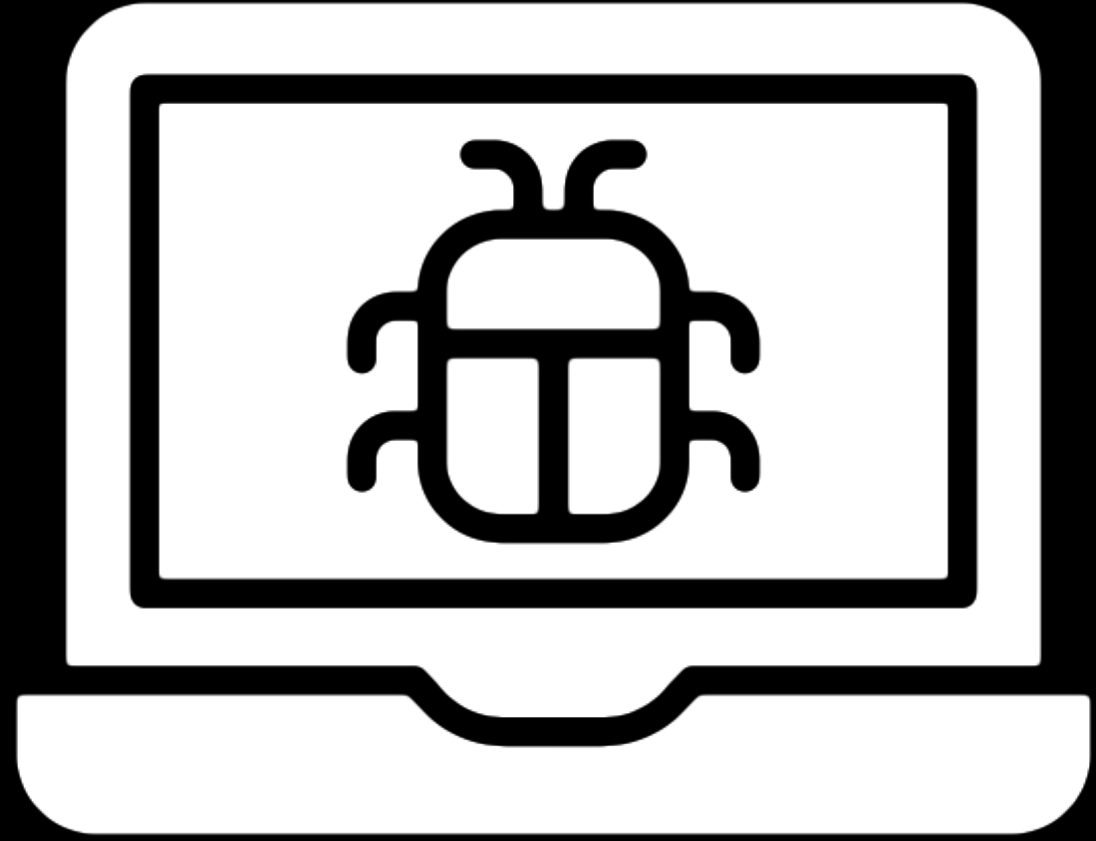
WannaCry Ransomware

May 2017

Actual Consequences



Actual Consequences

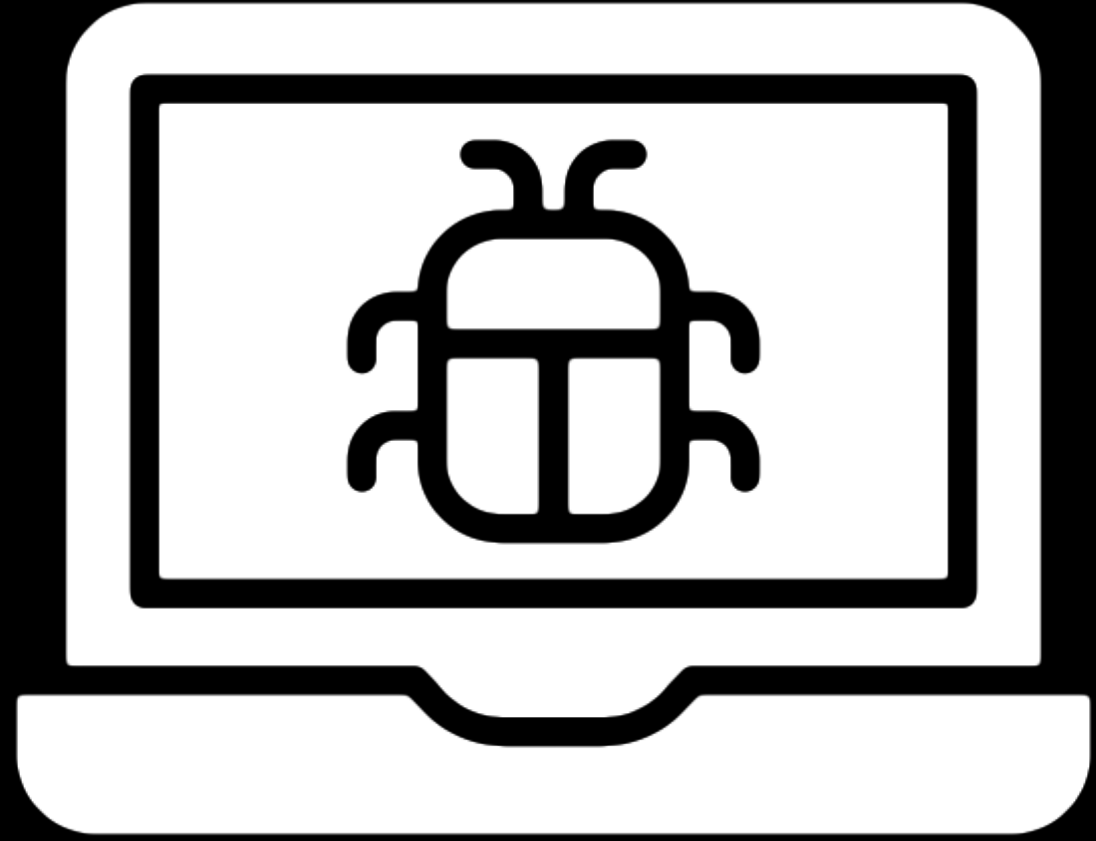


+



=

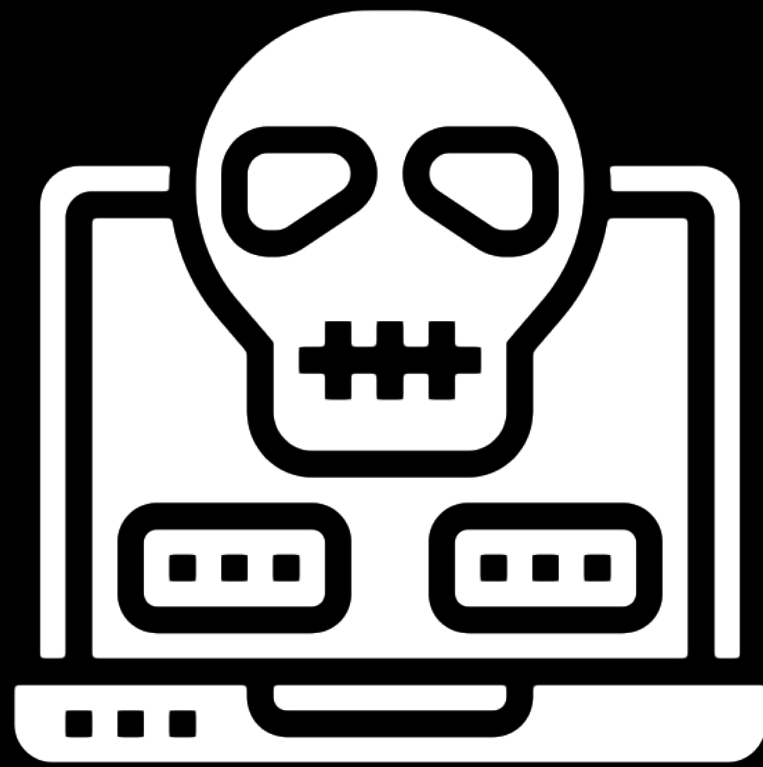
Actual Consequences



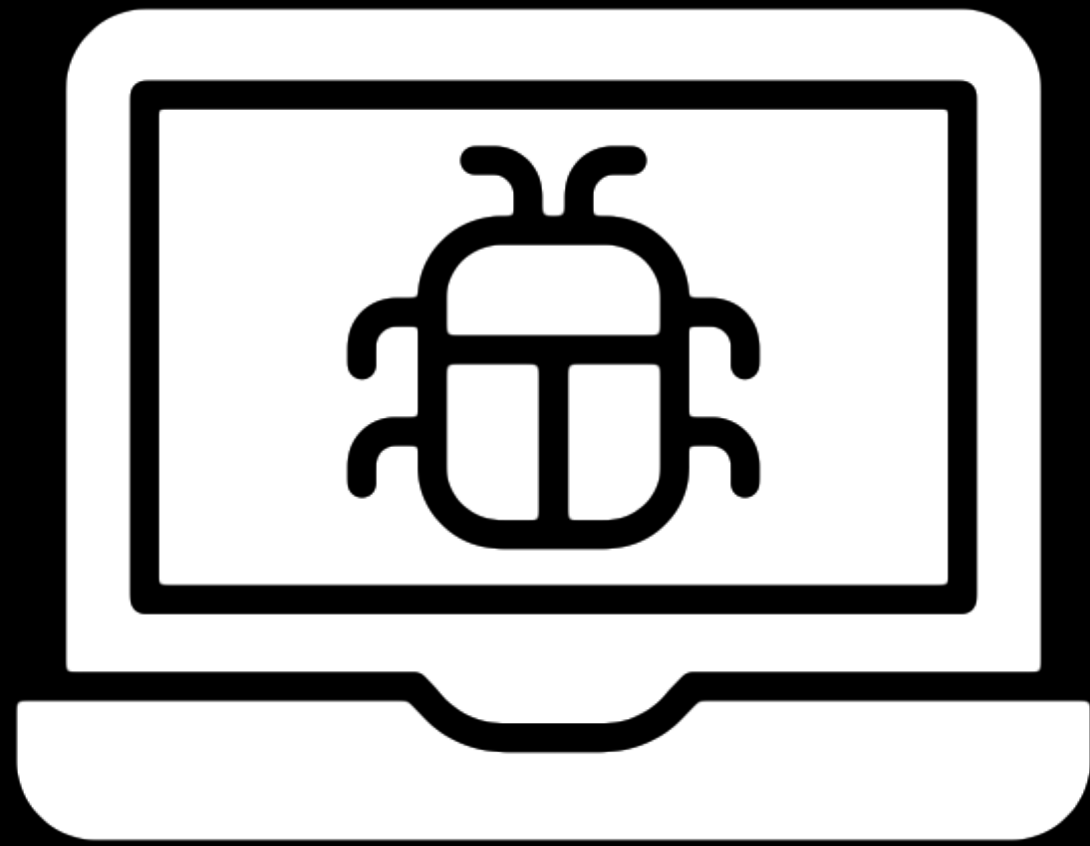
+



=



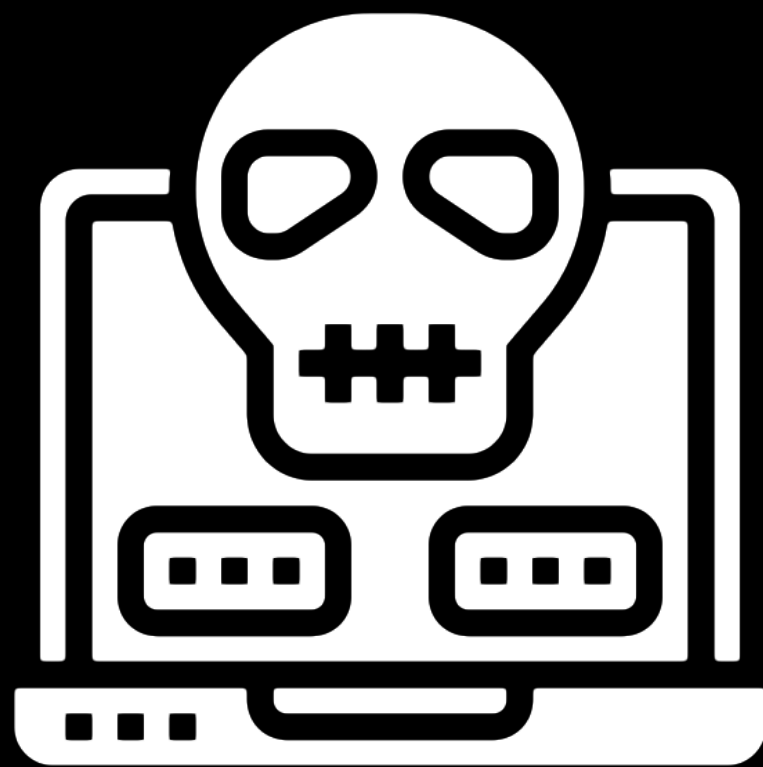
Actual Consequences



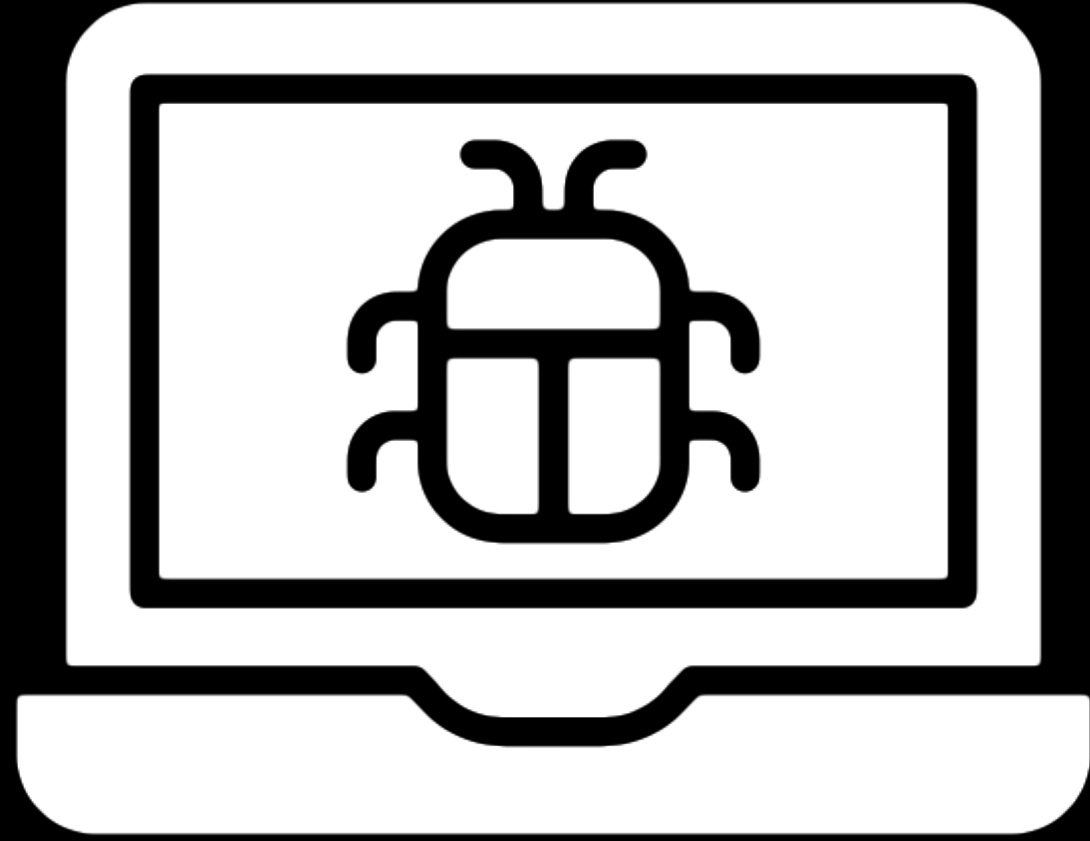
+



=



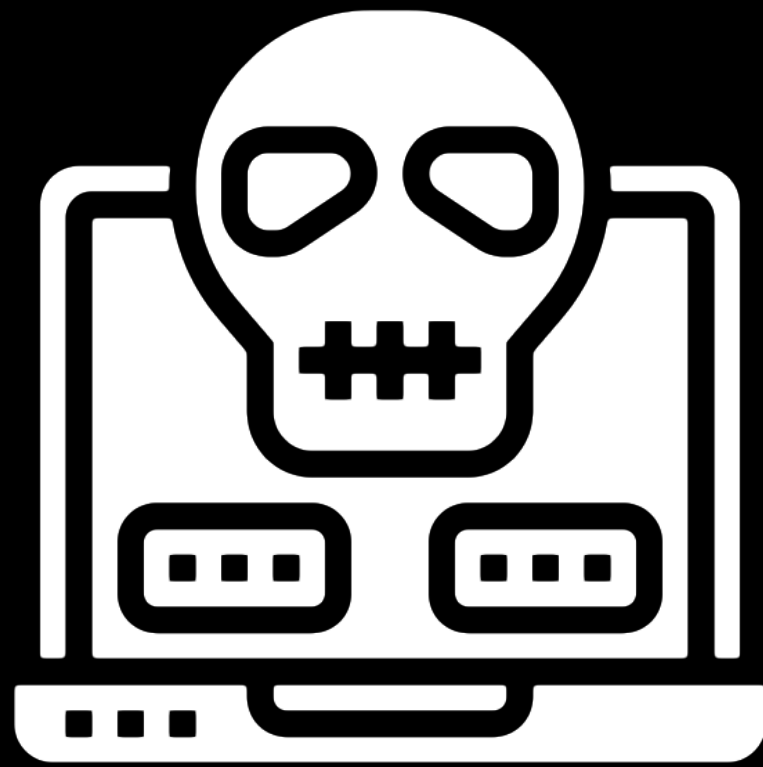
Actual Consequences



+



=



**people
dying**

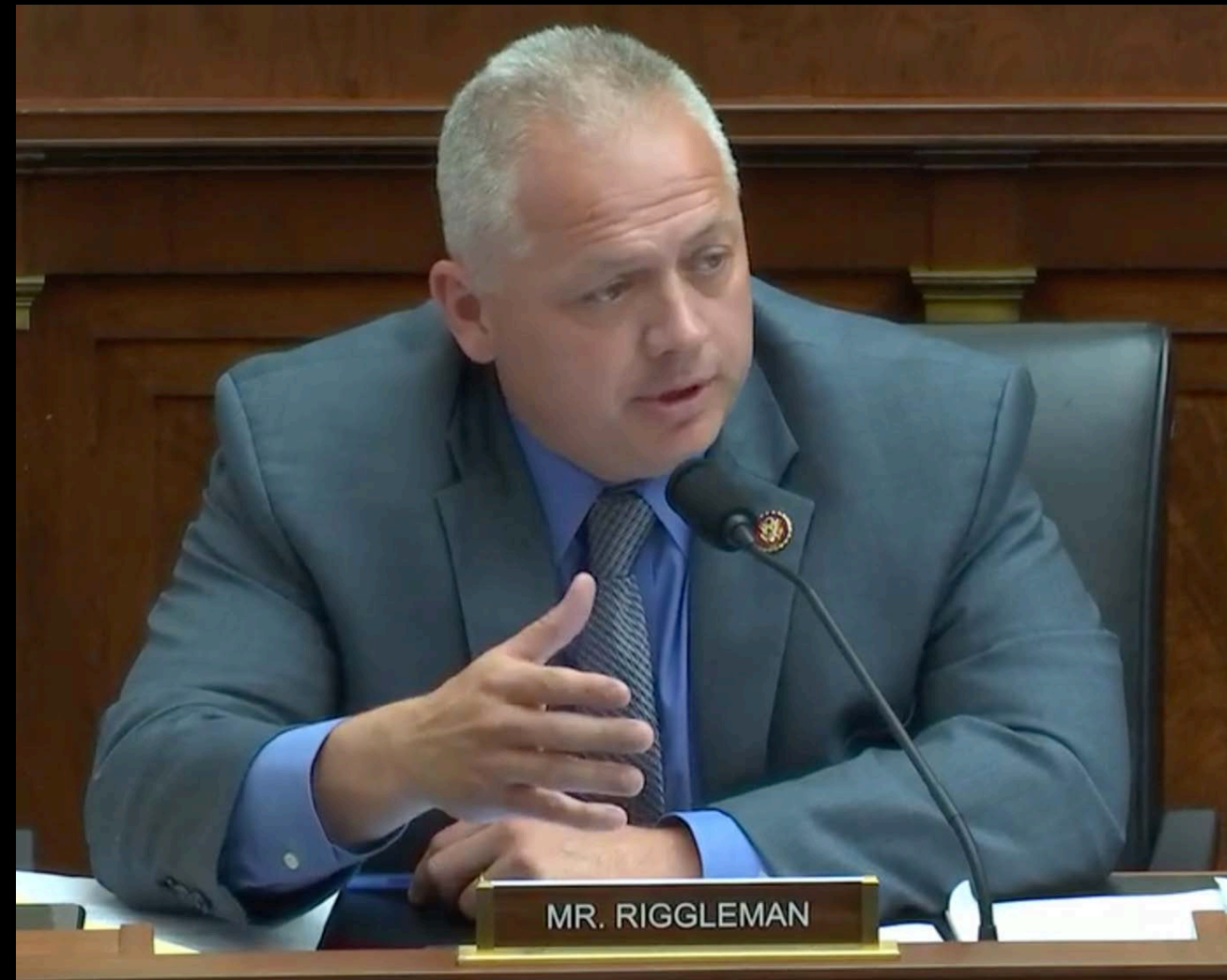
July 17, 2019

House Financial Services Committee

Hearing on Facebook's Libra Cryptocurrency

July 17, 2019

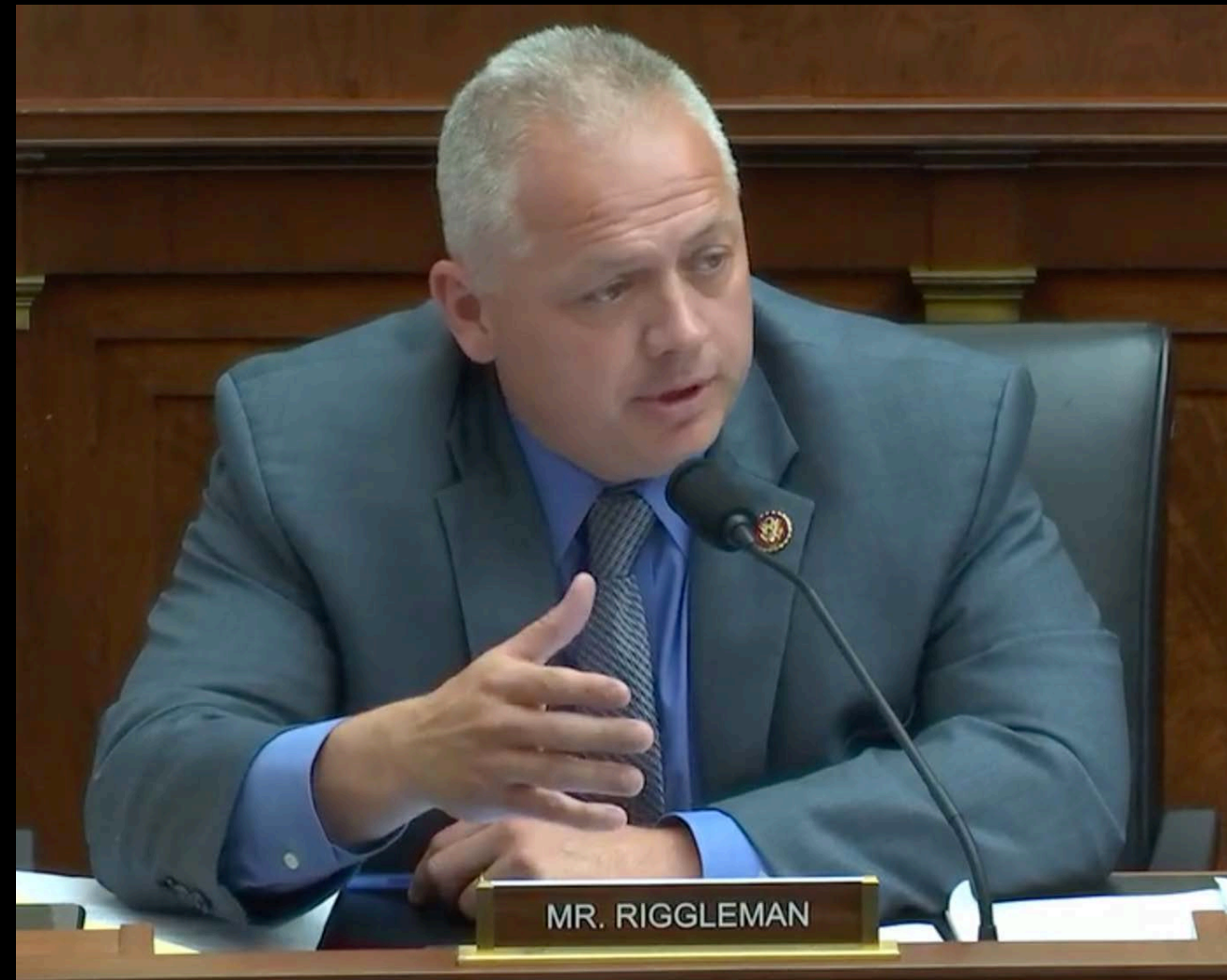
**House Financial Services Committee
Hearing on Facebook's Libra Cryptocurrency**



**Congressman Rigglesman
of Virginia**

July 17, 2019

House Financial Services Committee Hearing on Facebook's Libra Cryptocurrency



**Congressman Rigglesman
of Virginia**



**David Marcus
Head of Calibra at Facebook**

MR. BARR

MR. TIPTON

MR. RIGGLEMAN

Not the norm!!!

**We are our own
informed consumers**

1830 -> 1869 -> 1900

39 years

31 years

1973 -> 2015 -> ?????

42 years

?? years

Are we better?

Rust

In It for the Long Haul

Rust

In It for the Long Haul???

**I could be
wrong!**

GO



ZIG

Why not all of them?



new

mistakes!

is.gd/rustLH
@carols10cents
50% off Rust in
Motion course at
manning.com:
tsrust

