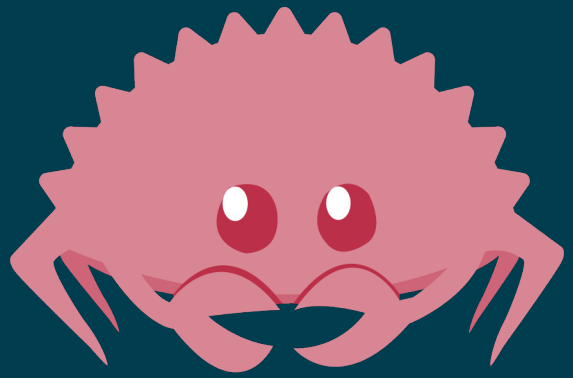
Ready for Rust

Erik Dörnenburg
erik@thoughtworks.com | @erikdoe



/thoughtworks

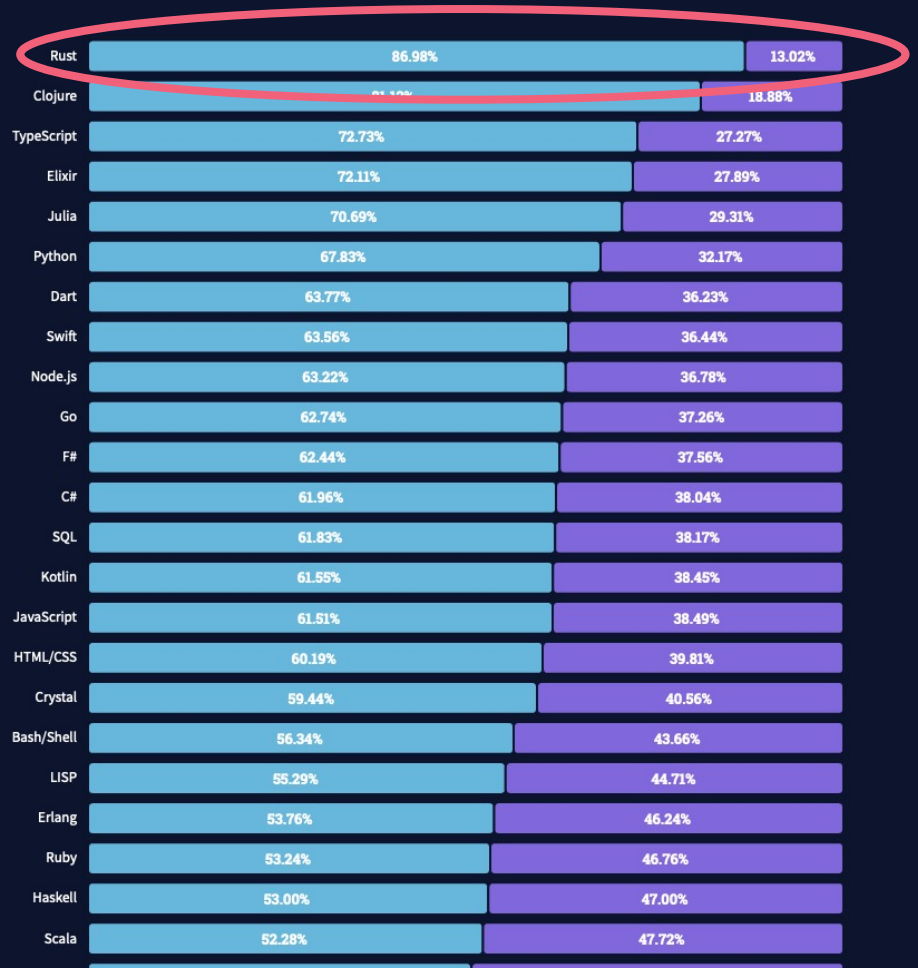
“It's brilliant”

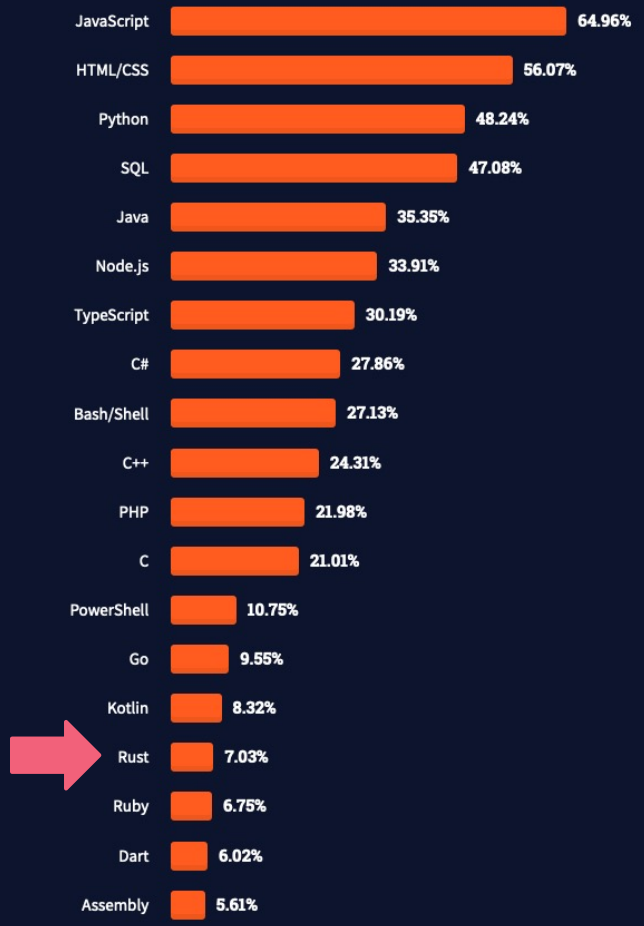
Mark Rendle, GOTO Copenhagen party keynote

Loved vs. Dreaded | Want

82,914 responses

All Respondents | Professional Developers





2021 Developer Survey

Google

moz://a


Swift


GO

About the security content of iOS 14.8.1 and iPadOS 14.8.1

This document describes the security content of iOS 14.8.1 and iPadOS 14.8.1.

About Apple security updates

For our customers' protection, Apple doesn't disclose, discuss, or confirm security issues until an investigation has occurred and patches or releases are available. Recent releases are listed on the Apple security updates page.

Apple security documents reference vulnerabilities by CVE-ID when possible.

For more information about security, see the Apple Product Security page.

---

iOS 14.8.1 and iPadOS 14.8.1

Released October 26, 2021

Audio

Available for: iPhone 6s and later, iPad Pro (all models), iPad Air 2 and later, iPad 5th generation and later, iPad mini 4 and later, and iPod touch (7th generation)

Impact: A malicious application may be able to elevate privileges

Description: An integer overflow was addressed through improved input validation.

CVE-2021-30907: Zweig of Kunlun Lab

ColorSync

Available for: iPhone 6s and later, iPad Pro (all models), iPad Air 2 and later, iPad 5th generation and later, iPad mini 4 and later, and iPod touch (7th generation)

Impact: Processing a maliciously crafted image may lead to arbitrary code execution

Description: A memory corruption issue existed in the processing of ICC profiles. This issue was addressed with improved input validation.

CVE-2021-30917: Alexandru-Vlad Niculae and Mateusz Jurczyk of Google Project Zero

Continuity Camera

About the security content of iOS 14.8.1 and iPadOS 14.8.1

This document describes the security content of iOS 14.8.1 and iPadOS 14.8.1.

About Apple security updates

For our customers' protection, Apple doesn't disclose, discuss, or confirm security issues until an investigation has occurred and patches or releases are available. Recent releases are listed on the Apple security updates page.

Apple security documents reference vulnerabilities by CVE-ID when possible.

For more information about security, see the Apple Product Security page.

iOS 14.8.1 and iPadOS 14.8.1

Released October 28, 2021

Audio

Available for: iPhone 6s and later, iPad Pro (all models), iPad Air 2 and later, iPad 5th generation and later, iPad mini 4 and later, and iPod touch (7th generation)

Impact: A malicious application may be able to elevate privileges

Description: An integer overflow was addressed through improved input validation.

CVE-2021-30667: Zweig of Kunlun Lab

ColorSync

Available for: iPhone 6s and later, iPad Pro (all models), iPad Air 2 and later, iPad 5th generation and later, iPad mini 4 and later, and iPod touch (7th generation)

Impact: Processing a maliciously crafted image may lead to arbitrary code execution

Description: A memory corruption issue existed in the processing of ICC profiles. This issue was addressed with improved input validation.

CVE-2021-30677: Alexandru-Vlad Niculae and Mateusz Jurczyk of Google Project Zero

Continuity Camera

Available for: iPhone 6s and later, iPad Pro (all models), iPad Air 2 and later, iPad 5th generation and later, iPad mini 4 and later, and iPod touch (7th generation)

Impact: A local attacker may be able to cause unexpected application termination or arbitrary code execution

Description: This issue was addressed with improved checks.

CVE-2021-30663: an anonymous researcher

CoreGraphics

Available for: iPhone 6s and later, iPad Pro (all models), iPad Air 2 and later, iPad 5th generation and later, iPad mini 4 and later, and iPod touch (7th generation)

Impact: Processing a maliciously crafted PDF may lead to arbitrary code execution

Description: An out-of-bounds write was addressed with improved input validation.

CVE-2021-30689

GPU Drivers

Available for: iPhone 6s and later, iPad Pro (all models), iPad Air 2 and later, iPad 5th generation and later, iPad mini 4 and later, and iPod touch (7th generation)

Impact: A malicious application may be able to execute arbitrary code with kernel privileges

Description: An out-of-bounds write issue was addressed with improved bounds checking.

CVE-2021-30660: Yinyi Wu (@3nds1) of Ant Security Light-Year Lab

IOMobileFrameBuffer

Available for: iPhone 6s and later, iPad Pro (all models), iPad Air 2 and later, iPad 5th generation and later, iPad mini 4 and later, and iPod touch (7th generation)

Impact: An application may be able to execute arbitrary code with kernel privileges. Apple is aware of a report that this issue may have been actively exploited.

Description: A memory corruption issue was addressed with improved memory handling.

CVE-2021-30683: an anonymous researcher

Kernel

Available for: iPhone 6s and later, iPad Pro (all models), iPad Air 2 and later, iPad 5th generation and later, iPad mini 4 and later, and iPod touch (7th generation)

Impact: An application may be able to execute arbitrary code with kernel privileges

Description: A memory corruption issue was addressed with improved memory handling.

CVE-2021-30668: Zweig of Kunlun Lab

Kernel

Available for: iPhone 6s and later, iPad Pro (all models), iPad Air 2 and later, iPad 5th generation and later, iPad mini 4 and later, and iPod touch (7th generation)

Impact: A malicious application may be able to execute arbitrary code with kernel privileges

Description: A memory corruption issue was addressed with improved memory handling.

CVE-2021-30690: Zweig of Kunlun Lab

Sidecar

Available for: iPhone 6s and later, iPad Pro (all models), iPad Air 2 and later, iPad 5th generation and later, iPad mini 4 and later, and iPod touch (7th generation)

Impact: A local attacker may be able to cause unexpected application termination or arbitrary code execution

Description: This issue was addressed with improved checks.

CVE-2021-30663: an anonymous researcher

Status Bar

Available for: iPhone 6s and later, iPad Pro (all models), iPad Air 2 and later, iPad 5th generation and later, iPad mini 4 and later, and iPod touch (7th generation)

Impact: A user may be able to view restricted content from the Lock Screen

Description: A Lock Screen issue was addressed with improved state management.

CVE-2021-30678: videosdebarraquito

Voice Control

Available for: iPhone 6s and later, iPad Pro (all models), iPad Air 2 and later, iPad 5th generation and later, iPad mini 4 and later, and iPod touch (7th generation)

Impact: A local attacker may be able to cause unexpected application termination or arbitrary code execution

Description: A use after free issue was addressed with improved memory management.

CVE-2021-30662: 08To3w88 of ZecOps Mobile EDR Team

WebKit

Available for: iPhone 6s and later, iPad Pro (all models), iPad Air 2 and later, iPad 5th generation and later, iPad mini 4 and later, and iPod touch (7th generation)

Impact: A malicious website using Content Security Policy reports may be able to leak information via redirect behavior

Description: An information leakage issue was addressed.

CVE-2021-30688: Prakash (@1337hr3ath)

Additional recognition

WebKit

We would like to acknowledge Ivan Fratric of Google Project Zero for their assistance.

Published Date: October 28, 2021

NEWS

C++ Memory Bugs Prompt Microsoft to Eye Rust Instead

By David Ramel ▪ 07/18/2019

Microsoft is eyeing the Rust programming language as a safer replacement of C/C++ code after discovering just how many security problems are caused by memory corruption bugs.

That news comes in a new blog post by the Microsoft Security Response Center (MSRC), which in triaging every reported Microsoft vulnerability since 2004 found that "one astonishing fact sticks out."

That astonishing fact? "The majority of vulnerabilities fixed and with a CVE

Most Popular Articles

What Are gRPC Web Services and When Should I Use Them?

How to Integrate Blazor Components

“The majority of vulnerabilities fixed and with a CVE assigned are caused by developers inadvertently inserting memory corruption bugs into their C and C++ code”

– *Microsoft Security Response Center: A proactive approach to more secure code (July 2019)*

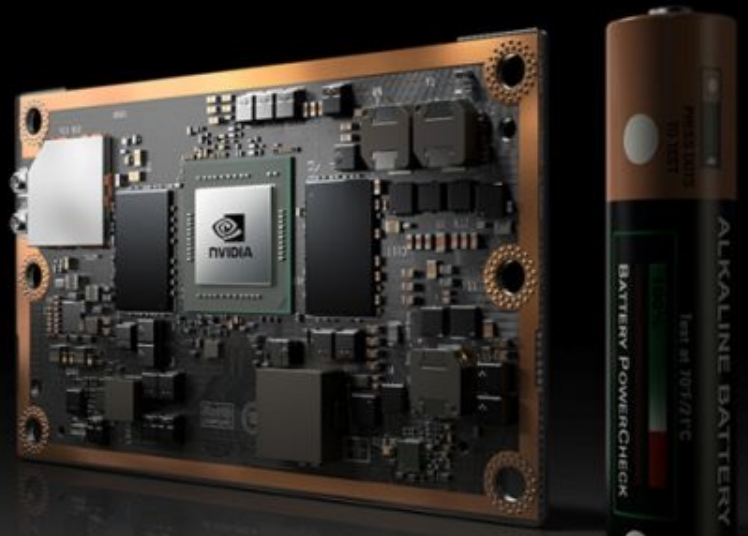
JETSON TX2

EMBEDDED AI SUPERCOMPUTER

2 Core i7 PCs in <10W

256 CUDA cores

>1 TFLOPS



cuDNN, TensorRT

CUDA

Linux or ROS

Getting ready

erik — -zsh

```
➜ ~ rustup update
info: syncing channel updates for 'stable-x86_64-apple-darwin'
info: latest update on 2019-08-15, rust version 1.37.0 (eae3437df 2019-08-13)
info: downloading component 'rustc'
 78.7 MiB /  78.7 MiB (100 %)  11.6 MiB/s ETA:   0 s
info: downloading component 'rust-std'
 56.3 MiB /  56.3 MiB (100 %)  11.0 MiB/s ETA:   0 s
info: downloading component 'cargo'
info: downloading component 'rust-docs'
info: downloading component 'rls-preview'
info: downloading component 'rust-src'
info: downloading component 'rust-analysis'
info: removing component 'rustc'
info: removing component 'rust-std'
info: removing component 'cargo'
info: removing component 'rust-docs'
info: removing component 'rls-preview'
info: removing component 'rust-src'
info: removing component 'rust-analysis'
info: installing component 'rustc'
info: installing component 'rust-std'
info: installing component 'cargo'
info: installing component 'rust-docs'
info: installing component 'rls-preview'
info: installing component 'rust-src'
```

read-for-rust — 80×24

```
➜ read-for-rust cargo new hello-rust
    Created binary (application) `hello-rust` package
➜ read-for-rust
```

Name

- src
  - main.rs
- Cargo.toml

crellinor-rust 〉 Cargo.toml

Project

- crellinor-rust [crellionor] ~/Projects/GA/cre
  - .idea
  - scripts
  - src
    - creature.rs
    - genetics.rs
    - lib.rs
    - loader.rs
    - log.rs
    - main.rs
    - multiverse.rs
    - params.rs
    - plant.rs
    - program.rs
    - random.rs
    - terrain.rs
    - utils.rs
    - world.rs
  - tests
  - .gitignore
  - bitbucket-pipelines.yml
  - Cargo.lock
  - Cargo.toml

Cargo.toml | program.rs | creature.rs | terrain.rs | genetics.rs | ra

```toml
[package]
name = "crellinor"
version = "0.1.0"
authors = ["Erik Doernenburg <erik@doernenburg.com>"]
edition = "2018"

[dependencies]
chrono = "0.4"
maplit = "1.0.1"
rand = "0.5.3"
serde = "1.0.76"
serde_json = "1.0.26"
serde_derive = "1.0.76"
uuid = { version = "0.6", features = ["v4", "serde"] }
```

crellinor-rust — -zsh

```
➜ crellinor-rust git:(master) ✗ cargo build
```

crellinor-rust — -zsh

```
➜ crellinor-rust git:(master) ✗ cargo test
```

Code

```rust
15 pub struct World {
16     pub name: Option<String>,
17     pub params: Params,
18     pub random: RNG,
19     pub terrain: Terrain,
20     pub cycle: u64,
21     pub log: Log,
22 }
23
24 impl World {
25     pub fn new(name: &str, params: Params) -> World {
26         let terrain = Terrain::with_size(params.world_size);
27         World {
28             name: Some(name.to_owned()),
29             params,
30             random: RNG::new(),
31             terrain,
32             cycle: 0,
33             log: Log::new(),
34         }
35     }
```

```rust
name = Some("My first world");
name = None;
```

```rust
terrain: terrain,
```

```rust
    pub fn do_cycles(&mut self, num: u64) {
        for _ in 0..num {
            self.do_one_cycle();
        }
    }
```

```rust
pub fn cycle_count(params: &Params, prog: &[Instr]) -> u64 {
    prog.iter().fold(0, |acc, instr| acc + params.instr_cycles(instr))
}
```

```rust
    pub fn all_instructions() -> HashMap<Instr, u64> {
        hashmap! {
            EAT => 10,
            MOV =>  5,
            TUR =>  3,
            TUL =>  3,
            NOP =>  1,
            JMP =>  1,
            JRE =>  1,
            BFH =>  1,
            BFA =>  1,
        }
    }
```

```rust
pub fn cycle_count(params: &Params, prog: &[Instr]) -> u64 {
    prog.iter().fold(0, |acc, instr| acc + params.instr_cycles(instr))
}
```

```rust
pub fn cycle_count(params: &Params, prog: &[Instr]) -> u64 {
    prog.iter().map(|instr| params.instr_cycles(instr)).sum()
}
```

Memory management

```rust
{                           // s is not valid here, it's not yet declared
    let s = "hello";        // s is valid from this point forward

    // do stuff with s
}                           // this scope is now over, and s is no longer valid
```

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

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

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

let len = calculate_length(&s1);
```

```rust
fn calculate_length(s: &String) -> usize { // s is a reference to a String
    s.len()
} // Here, s goes out of scope. But because it does not have ownership of what
  // it refers to, nothing happens.
```

```
let message = get_message();
```

```
6962 FFFF 6821 0800 00E9 44FC FFFF 6835 0800 00E9 3AFC FFFF 6845 0800 00E9  ..h!    .D...h5    .:...hE    .
6963 30FC FFFF 6855 0800 00E9 26FC FFFF 6865 0800 00E9 1CFC FFFF 6874 0800  0...hU    .&...he    . ...ht
69658 00E9 12FC FFFF 6884 0800 00E9 08FC FFFF 6894 0800 00E9 FFFB FFFF 0000        h          h
69674 446F 6E65 2E20 5265 6163 6865 6420 7468 6520 656E 6420 6F66 2061 6C6C  Done. Reached the end of all
69690 2077 6F72 6C64 732E 0A00 0000 0054 7269 6564 2074 6F20 7368 7269 6E6B   worlds.      Tried to shrink
696AC 2074 6F20 6120 6C61 7267 6572 2063 6170 6163 6974 796C 6962 616C 6C6F  to a larger capacityliballo
696C8 632F 7261 775F 7665 632E 7273 6372 6561 7475 7265 5F6D 6178 5F61 6765  c/raw_vec.rscreature_max_age
696E4 6361 6C6C 6564 2060 4F70 7469 6F6E 3A3A 756E 7772 6170 2829 6020 6F6E  called `Option::unwrap()` on
69700 2061 2060 4E6F 6E65 6020 7661 6C75 656C 6962 636F 7265 2F6F 7074 696F   a `None` valuelibcore/optio
6971C 6E2E 7273 0000 0000 0000 0000 0006 9674 0000 0100 0000 0000 0000 0000  n.rs          .t
69738 0000 0000 0000 0000 0000 0000 0000 0200 0000 0000 0000 0300 0000 0000
69754 0000 0000 0000 0000 0000 2000 0000 0000 0000 0300 0000 0000 0000 5061                                Pa
```

```
let messageRef = &message;
```

Illustration only! Not an accurate description of Rust, or C, or common CPUs.

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

let len = calculate_length(&s1);
```

```rust
fn calculate_length(s: &String) -> usize { // s is a reference to a String
    s.len()
} // Here, s goes out of scope. But because it does not have ownership of what
  // it refers to, nothing happens.
```

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

    change(&s);
}

fn change(some_string: &String) {
    some_string.push_str(", world");
}
```

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

    change(&mut s);
}

fn change(some_string: &mut String) {
    some_string.push_str(", world");
}
```

```rust
let mut s = String::from("hello");

let r1 = &mut s;
let r2 = &mut s;
```

```rust
let mut s = String::from("hello");

let r1 = &s; // no problem
let r2 = &s; // no problem
let r3 = &mut s; // BIG PROBLEM
```

```rust
fn main() {
    let reference_to_nothing = dangle();
}

fn dangle() -> &String {
    let s = String::from("hello");

    &s
}
```

```rust
    pub fn do_with_creatures_mut<F>(&mut self, mut func: F)
        where F: FnMut(&mut Terrain, &mut Creature, (u32, u32)) -> Option<(u32, u32)> {
```

```rust
    fn process_all_creatures(&mut self) {
        self.terrain.do_with_creatures_mut(|terrain, creature, pos|
            {
                creature.ep -= 1;
                if (creature.age() >= self.params.creature_max_age) || (creature.ep == 0) {
                    return None;
                }
                let mut ctx = PContext::new(&self.params, terrain, self.cycle, pos);
                return Some(creature.do_cycle(&mut ctx));
            });
    }
```

```
/Users/erik/.cargo/bin/cargo build --color=always --all --all-targets
   Compiling crellinor v0.1.0 (/Users/erik/Projects/GA/crellinor-rust)
error[E0501]: cannot borrow `self.terrain` as mutable because previous closure requires unique access
   --> src/world.rs:137:9
    |
137 |           self.terrain.do_with_creatures_mut(|terrain, creature, pos|
    |           ^            -------------------- ----------------------- closure construction occurs here
    |           |            |
    |  ________|            first borrow later used by call
    | |
138 | |             {
139 | |                 creature.ep -= 1;
140 | |                 if (creature.age() >= self.params.creature_max_age) || (creature.ep == 0) {
    | |                                       ---- first borrow occurs due to use of `self` in closure
...   |
144 | |                 return Some(creature.do_cycle(&mut ctx));
145 | |             });
    | |_____________^ second borrow occurs here

error[E0500]: closure requires unique access to `self` but it is already borrowed
   --> src/world.rs:137:44
    |
137 |         self.terrain.do_with_creatures_mut(|terrain, creature, pos|
    |         ------------ -------------------- ^^^^^^^^^^^^^^^^^^^^^^^^ closure construction occurs here
    |         |            |
    |         |            first borrow later used by call
    |         borrow occurs here
...
140 |                 if (creature.age() >= self.params.creature_max_age) || (creature.ep == 0) {
    |                                       ---- second borrow occurs due to use of `self` in closure

error: aborting due to 2 previous errors

Some errors have detailed explanations: E0500, E0501.
For more information about an error, try `rustc --explain E0500`.
```

rust-lang / rust #25957

```
➜  crellinor-rust git:(master) ✗ cargo build
   Compiling crellinor v0.1.0 (/Users/edoernen/Projects/GA/crellinor-rust)
error: unknown start of token: \u{37e}
  --> src/main.rs:13:30
   |
13 |     crellinor::run(worldfile);
   |                              ^
   |
help: Unicode character ';' (Greek Question Mark) looks like ';' (Semicolon), but it is not
   |
13 |     crellinor::run(worldfile);
   |                              ~

error: could not compile `crellinor` due to previous error
```

```rust
fn process_all_creatures(&mut self) {
    self.terrain.do_with_creatures_mut(|terrain, creature, pos|
        {
            creature.ep -= 1;
            if (creature.age() >= self.params.creature_max_age) || (creature.ep == 0) {
                return None;
            }
            let mut ctx = PContext::new(&self.params, terrain, self.cycle, pos);
```

E0500: closure requires unique access to self but it is already borrowed

```rust
fn process_all_creatures(&mut self) {
    let cycle = self.cycle;
    let params = &self.params;
    self.terrain.do_with_creatures_mut(|terrain, creature, pos|
        {
            creature.ep -= 1;
            if (creature.age() >= params.creature_max_age) || (creature.ep == 0) {
                return None;
            }
            let mut ctx = PContext::new(params, terrain, cycle, pos);
```

Parallelism

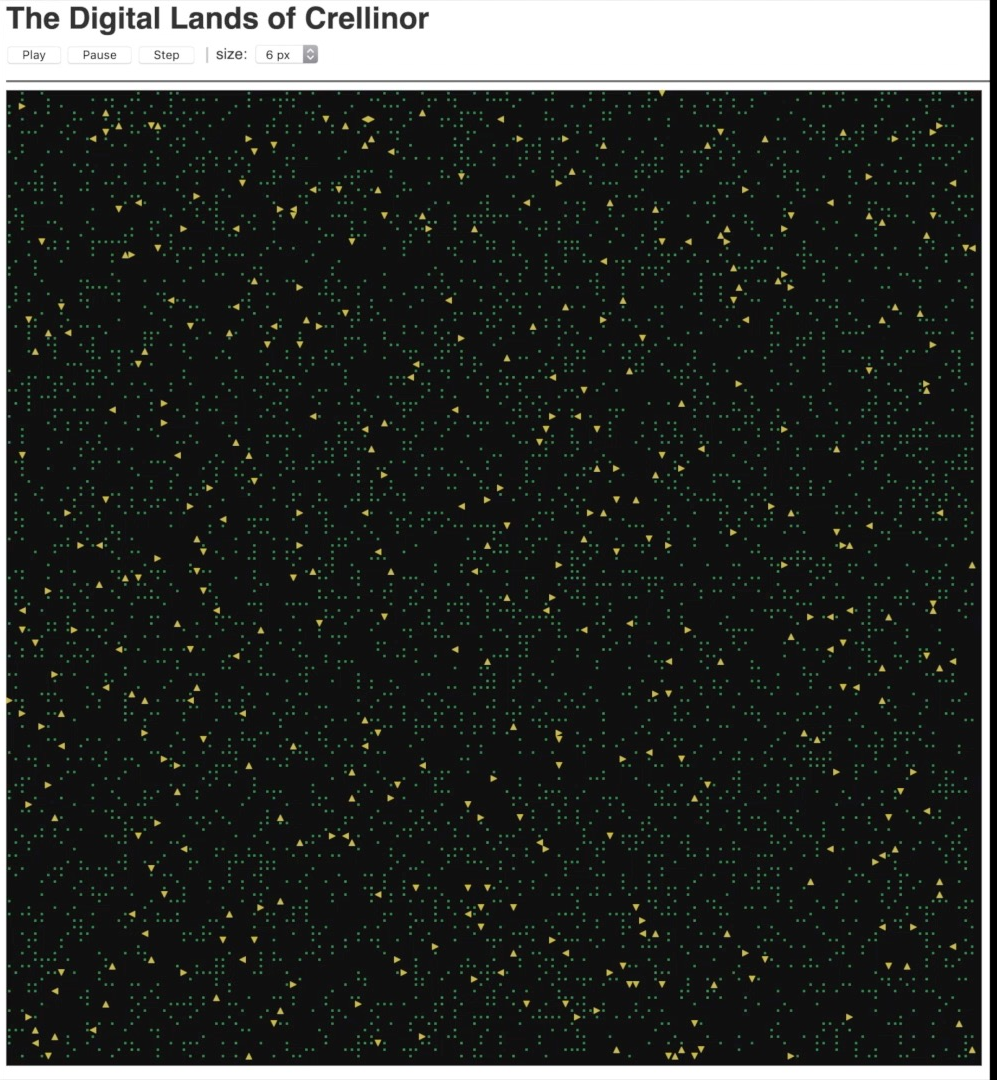
```rust
fn run_multiverse(worldfn: fn() -> World) {
    let mut handles = Vec::new();

    for tnum in 0..NUM_THREADS {
        let h = thread::spawn(move || {
            for snum in 0..(NUM_SIMS / NUM_THREADS) {
                run_world(tnum, snum, worldfn());
            }
        });
        handles.push(h);
    }

    while let Some(h) = handles.pop() {
        h.join().unwrap();
    }
}
```

Performance

Clojure:
110,000 cycles/s



Rust:
~~3,500,000 cycles/s~~
25,000,000 cycles/s

“I had experienced some frustrations trying to implement in Rust the same structure I had had in C. So I mentally gave up on performance, resolving to just get something working first.”

– Bryan Cantrill, on his blog (September 2018)

Time to generate a statemap for a "modest" trace (~4 million state transitions)

Node.js	**83.1s**
Node/C hybrid	**11.8s**
Rust	**8.1s**

Thank you for listening, now it's time for questions

Erik Dörnenburg
Head of Technology
erik@thoughtworks.com | @erikdoe

/thoughtworks