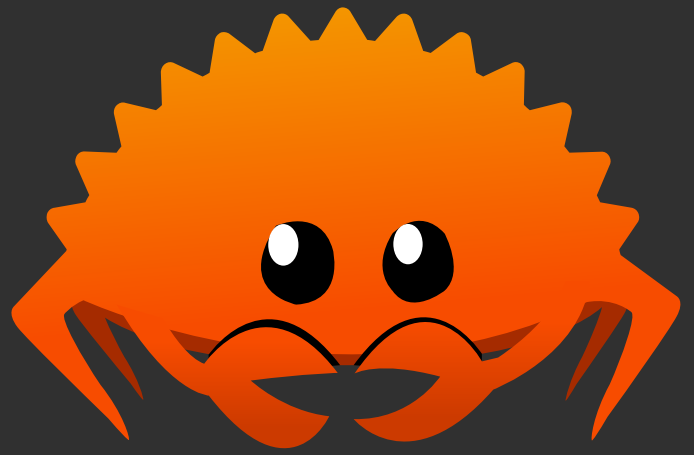
RED BADGER



Rust in the Enterprise

2023-09-23

Stuart Harris and Viktor Charypar

1. How does Rust help solve the problems that Enterprises care about?
2. Where does the resistance to adopting Rust come from?
3. How do we go about introducing Rust into an Enterprise?
4. Beer and Pizza
5. Crux — how Rust can help enterprises build multi-platform apps

Stu

- Software engineer
- Founder of Red Badger

Viktor

- Software engineer
- Tech Director at Red Badger

RED BADGER

What do enterprises care about?

1. **Quality** software that works
2. **Sustainability** — green software that lasts
3. **Security** and reputational damage
4. **Cost** and speed of delivery
5. **Control, Risk, Compliance** operate safely in regulated environments
6. **Innovation, Talent, Culture** be a destination employer

1

Quality software that works

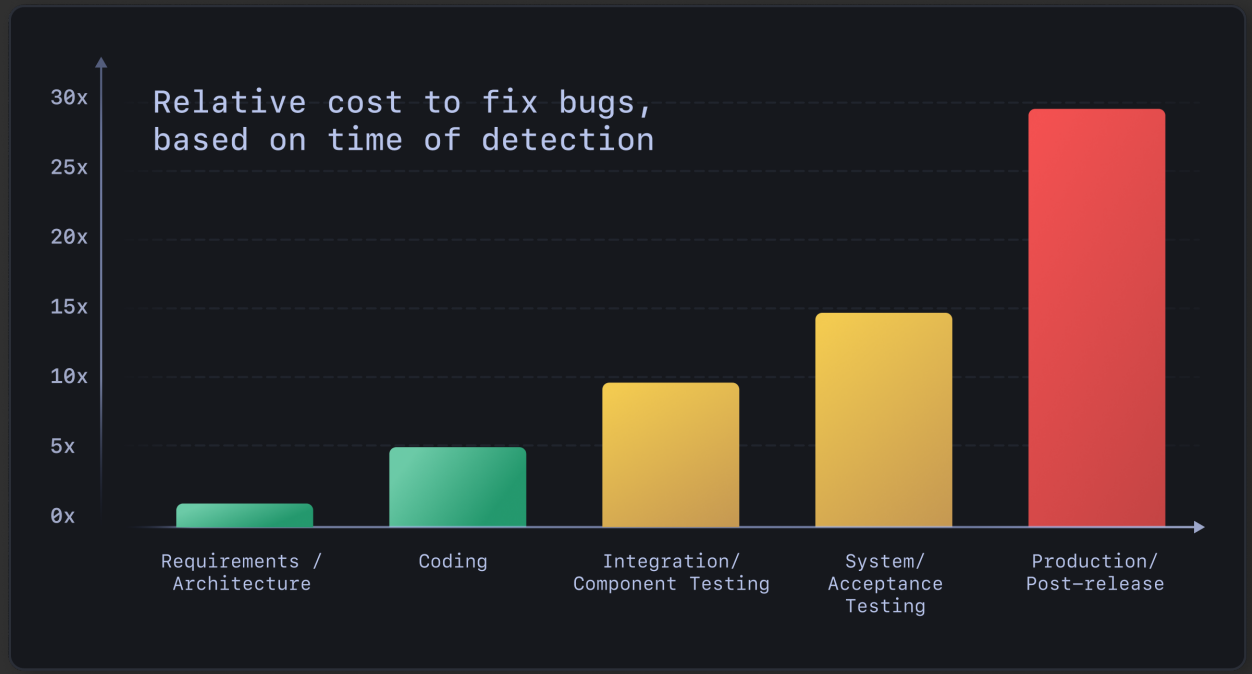
... and delights our users

- easy to use, accessible for everyone, fast and reliable
- maintainable, extensible and **testable**

1 Shifting left

Remove the long tail of bugs that are expensive to fix

> "Rust is the language where you have the hangover first!"



https://deepsource.com/blog/exponential-cost-of-fixing-bugs

What do enterprises care about?

1. **Quality** software that works
2. **Sustainability** — green software that lasts
3. **Security** and reputational damage
4. **Cost** and speed of delivery
5. **Control, Risk, Compliance** operate safely in regulated environments
6. **Innovation, Talent, Culture** be a destination employer

Sustainability — green software that lasts

… and is easy to maintain and extend

> Must be easy to understand, test, deploy, change

2 Where does the effort go?

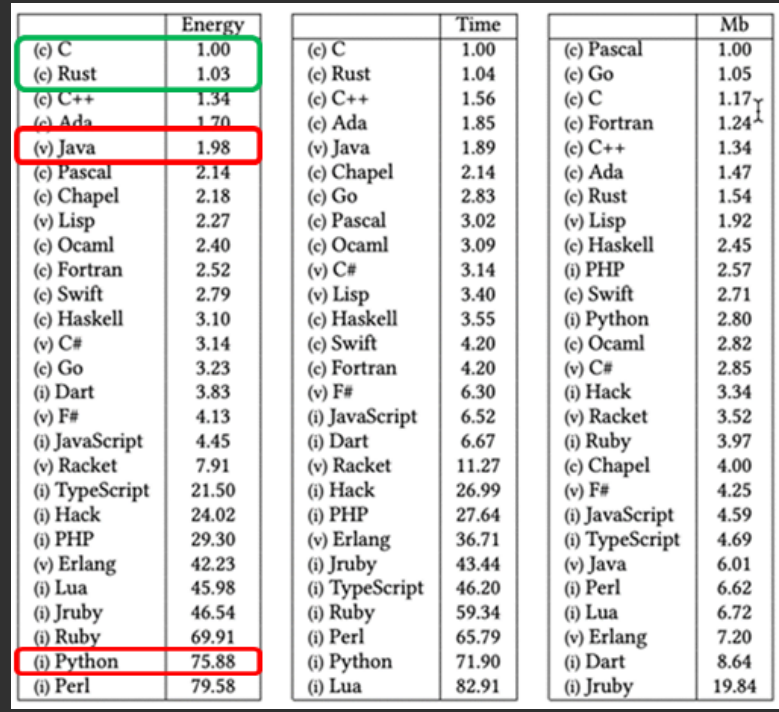
> "Free up your teams to work on the most important problems — those that are **unique** to your business"

- Rust's guardrails prevent us wasting time on bugs
- Rust's toolchain is the best there is
  - removes bike-shedding
  - reduces yak-shaving
- A large codebase won't descend into chaos
- "It just works"™
  - no long tail — means you can leave it alone

2 Sustainable Software Engineering

> "Everyone has a part to play." "Sustainability is enough, all by itself, to justify our work."

- Find as many "Carbon Proxies" as you can, e.g.
  - Time-to-interactive and Page weight
  - Average server response time
  - Cost of your services
  - The utilisation of your servers



	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

	Time
(c) C	1.00
(c) Rust	1.04
(c) C++	1.56
(c) Ada	1.85
(v) Java	1.89
(c) Chapel	2.14
(c) Go	2.83
(c) Pascal	3.02
(c) Ocaml	3.09
(v) C#	3.14
(v) Lisp	3.40
(c) Haskell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	11.27
(i) Hack	26.99
(i) PHP	27.64
(v) Erlang	36.71
(i) Jruby	43.44
(i) TypeScript	46.20
(i) Ruby	59.34
(i) Perl	65.79
(i) Python	71.90
(i) Lua	82.91

	Mb
(c) Pascal	1.00
(c) Go	1.05
(c) C	1.17
(c) Fortran	1.24
(c) C++	1.34
(c) Ada	1.47
(c) Rust	1.54
(v) Lisp	1.92
(c) Haskell	2.45
(i) PHP	2.57
(c) Swift	2.71
(i) Python	2.80
(c) Ocaml	2.82
(v) C#	2.85
(i) Hack	3.34
(v) Racket	3.52
(i) Ruby	3.97
(c) Chapel	4.00
(v) F#	4.25
(i) JavaScript	4.59
(i) TypeScript	4.69
(v) Java	6.01
(i) Perl	6.62
(i) Lua	6.72
(v) Erlang	7.20
(i) Dart	8.64
(i) Jruby	19.84

sustainability with rust sustainable software engineering overview how to measure and reduce the carbon footprint of your application

What do enterprises care about?

1. **Quality** software that works
2. **Sustainability** — green software that lasts
3. **Security** and reputational damage
4. **Cost** and speed of delivery
5. **Control, Risk, Compliance** operate safely in regulated environments
6. **Innovation, Talent, Culture** be a destination employer

Security and reputational damage

Secure by design

> Must be easy to **reason** about, solid and **reliable**, built on an **open** and **secure** foundation

What do enterprises care about?

1. **Quality** software that works
2. **Sustainability** — green software that lasts
3. **Security** and reputational damage
4. **Cost** and speed of delivery
5. **Control, Risk, Compliance** operate safely in regulated environments
6. **Innovation, Talent, Culture** be a destination employer

Cost and speed of delivery

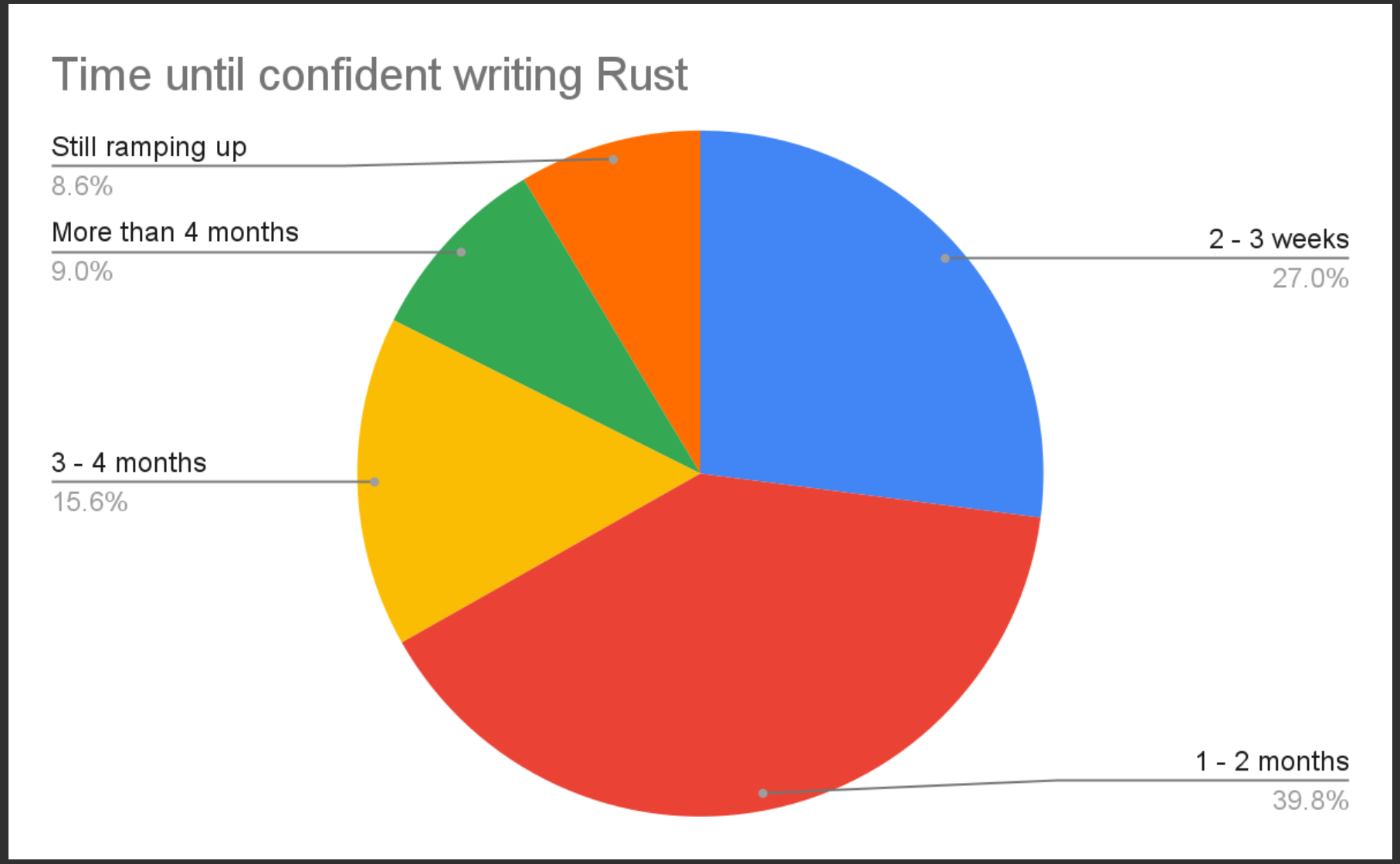
> We think that, overall, Rust speeds up development and reduces costs

4 There are rumours...

- Rust takes more than 6 months to learn – Debunked! 😇
- The Rust compiler is not as fast as people would like – Confirmed! 😅
- Unsafe code and interop are always the biggest challenges – Debunked! 😌
- Rust has amazing compiler error messages – Confirmed! 😍
- Rust code is high quality – Confirmed! 🤩

from over 1,000 Google developers

https://opensource.googleblog.com/2023/06/rust-fact-vs-fiction-5-insights-from-googles-rust-journey-2022.html


Time until confident writing Rust
Still ramping up
8.6%
More than 4 months
9.0%
3 - 4 months
15.6%
2 - 3 weeks
27.0%
1 - 2 months
39.8%

What do enterprises care about?

1. **Quality** software that works
2. **Sustainability** — green software that lasts
3. **Security** and reputational damage
4. **Cost** and speed of delivery
5. **Control, Risk, Compliance** — operate safely in a regulated environment
6. **Innovation, Talent, Culture** be a destination employer

Control, Risk, Compliance — operate safely in a regulated environment

- trusted and open source
- safe from memory related bugs
- safe from data races and concurrency bugs
- easy to **test** and **maintain** — good for regulated industries

5 Rust and WebAssembly

- Rust and Wasm are a perfect match
- Wasm is becoming a game-changer for cloud native
- The sandboxed execution environment is a great fit for enterprise

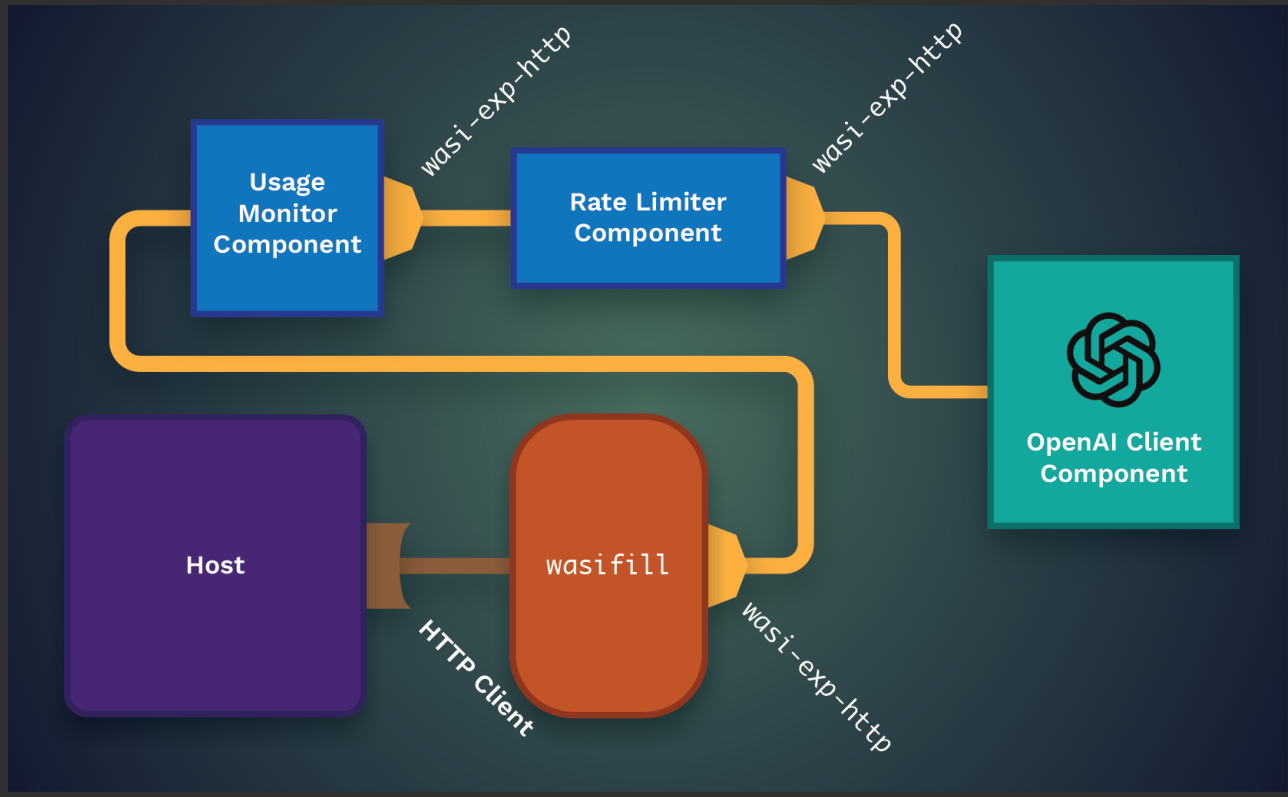
5

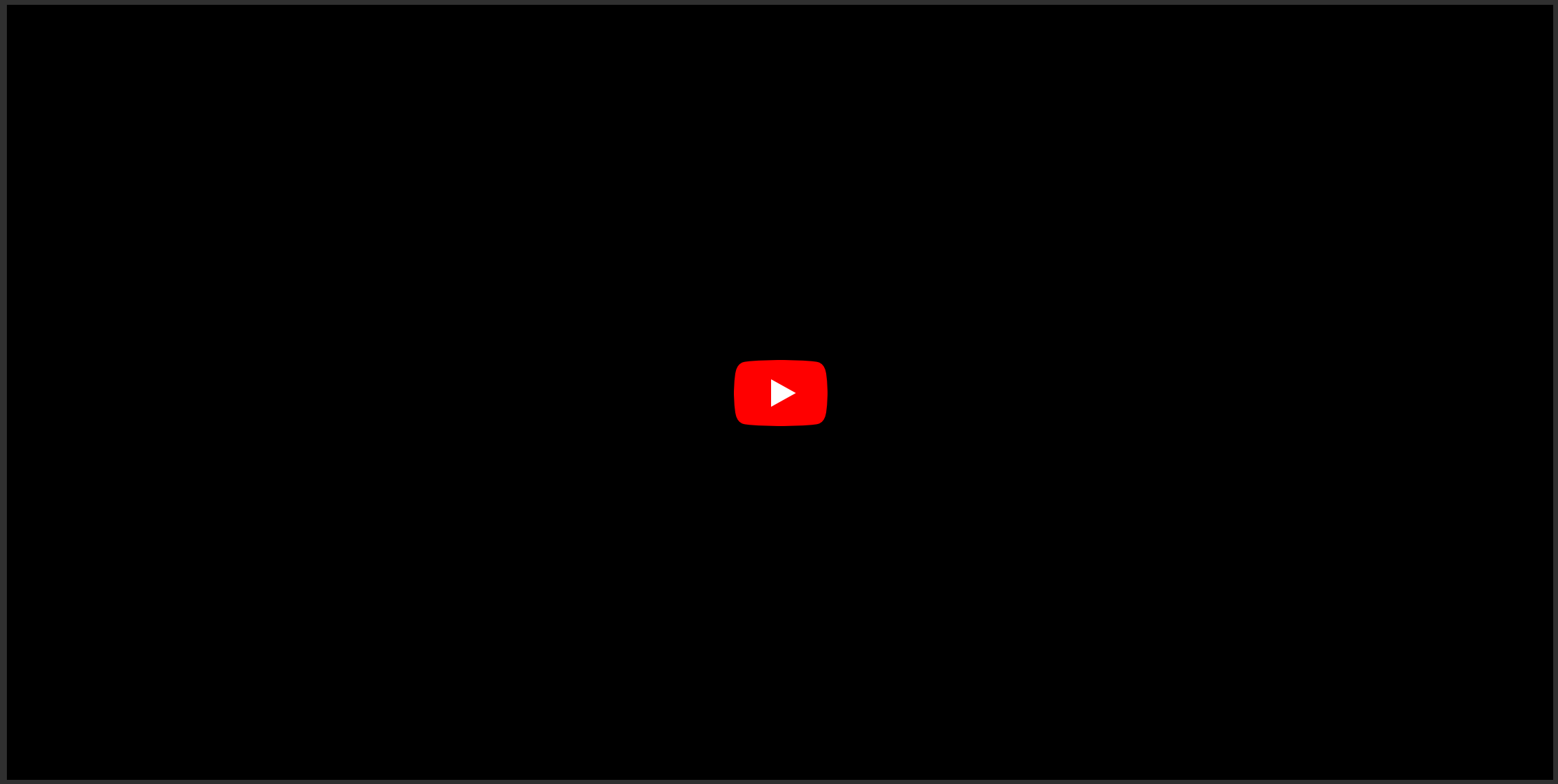
WebAssembly is a revolution

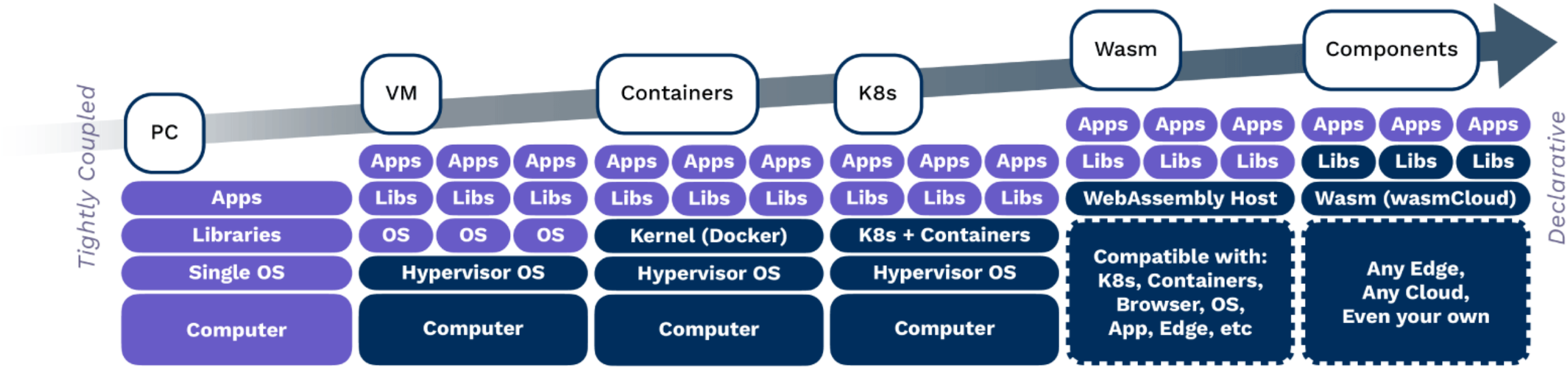
WebAssembly Component Model

WIT

https://cosmonic.com/blog/engineering/gap-bridging-with-wasifills







FORMAT	PC	CLOUD	CONTAINER	K8S	WASM	COSMONIC
EXECUTION	*Image (Datacenter)*	*VM (Public Cloud)*	*Container (Docker)*	*Containers (K8s / Cloud)*	*WASM (Everywhere)*	*Distributed WASM (Everywhere)*
Dev Responsibility	Full	OS, App, Lib	App, Lib	App, Lib	Wasm	Business Logic
Abstraction	-	CPU	Linux Kernel	K8s	Secure Sandbox	Sandbox + Capabilities
Compatibility	All	Most	Most	Most	Most	Most
Size	Large	Med	Small	Small	Tiny	Minuscule
Portability	-	Low	Med (CPU, Linux)	Med (CPU, Linux)	High	Highest
Security	System	OS	Process Boundary	Process Boundary	Capability	Component
Location	On Prem & Co-location	Proprietary Cloud & Edge	Dev, Edge, Cloud, K8s	Dev, Edge, Cloud, K8s	Dev, App, Edge, Cloud, K8s, Browser, Devices	Dev, App, Edge, Cloud, K8s, Browser, Devices

Legend: **Developer Provided** **Service Provided** **Flexible**

Developer writes app logic

```rust
use wasmbus_rpc::actor::prelude::*;
use wasmcloud_interface_httpserver::{HttpRequest, HttpResponse, HttpServer, HttpServerReceiver};

#[derive(Debug, Default, Actor, HealthResponder)]
#[services(Actor, HttpServer)]
struct HelloActor {}

/// Implementation of the HTTP server capability
#[async_trait]
impl HttpServer for HelloActor {
    async fn handle_request(&self, _ctx: &Context, _req: &HttpRequest) -> RpcResult<HttpResponse>
{       Ok(HttpResponse::ok("Hello, World!"))
    }
}
```

Platform provides everything else

Developer code

RUNTIME

Wasmtime

CONTRACT

wasmcloud:keyvalue

CONTRACT

wasmcloud:sqldb

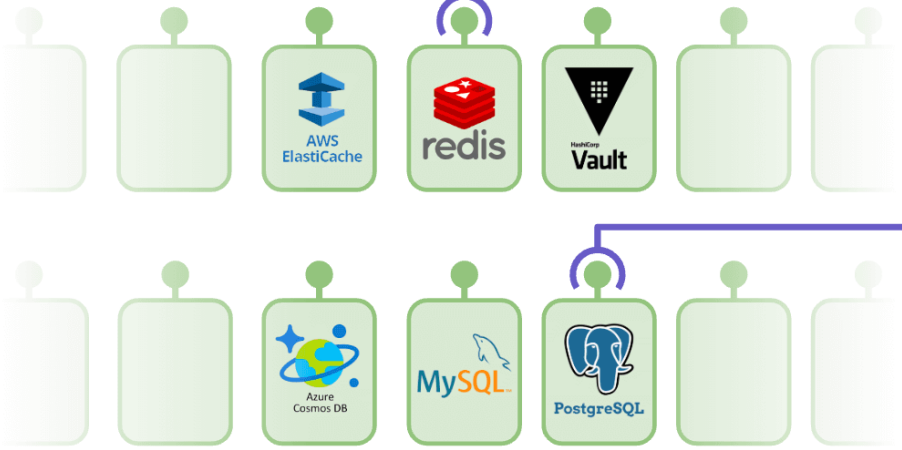
cosmonic

Define application with OAM Schema

```yaml
apiVersion: core.oam.dev/v1beta1
kind: ApplicationConfiguration
metadata:
  name: myapp
  annotations:
    version: v1.23.0
    description: Sample application for the demo
spec:
  components:
    - name: ui
      type: actor
      properties:
        image: wasmcloud.azurecr.io/ui:0.3.2
      traits:
        - type: spreadscaler
          properties:
            replicas: 1
            spread:
              - name: uimyapp
                requirements:
                  app: myapp

    - name: customers
      type: actor
      properties:
        image: wasmcloud.azurecr.io/customers:0.3.1
      traits:
        - type: linkdef
          properties:
            target: postgres
```

Swap providers *at runtime*



What do enterprises care about?

1. **Quality** software that works
2. **Sustainability** — green software that lasts
3. **Security** and reputational damage
4. **Cost** and speed of delivery
5. **Control, Risk, Compliance** operate safely in regulated environments
6. **Innovation, Talent, Culture** be a destination employer

6

Innovation, Talent, Culture be a destination employer

> "Managers need to know that their hard-won employees are gainfully employed and are not frustrated, bored, or flight risks"

6 Innovation, Talent, Culture

- attract and retain talent — become a "destination"
- build a culture of engineering excellence
- Rust helps you become a better software engineer
- "Most loved/desired" on Stack Overflow for 8 years in a row

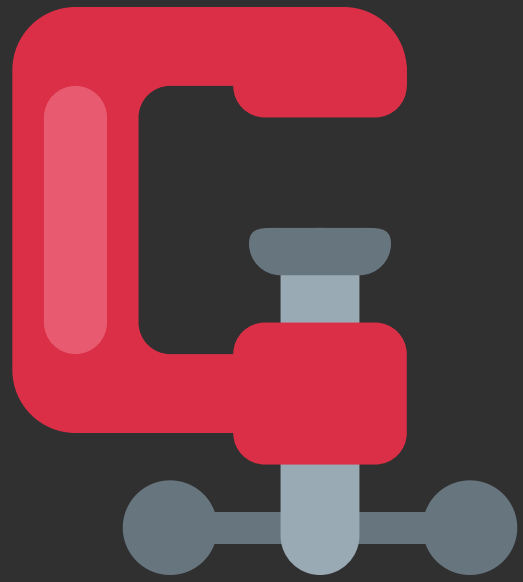
Where does the resistance come from?

- **Risk** — "We don't want to be the first to do this"
- **Cost** — "We don't have the budget"
- **Speed** — "We don't have the time"
- **Culture** — "We don't have the talent"

> This is all FEAR talking — FUD!

Pincer movement!

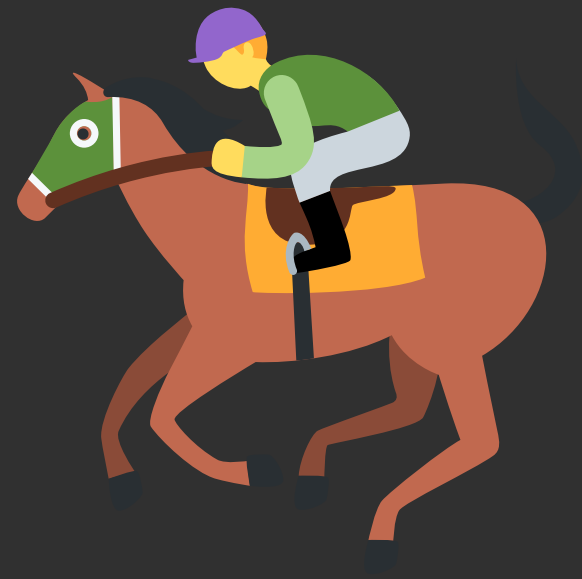
- **Top down** — "We need to do this"
- **Bottom up** — "We want to do this"



How do we get there?

Top down

- Talk to the CIO. Find a buddy to help you
- Find the open source owner (OSPO)
- Find a champion (and a project goal)
    - Might not be a technology champion
    - Would be great to have a product sponsor
- Run a 2-horse race



Skunk Works

> a group within an organization given a high degree of autonomy and unhampered by bureaucracy, with the task of working on advanced or secret projects

Now is the time

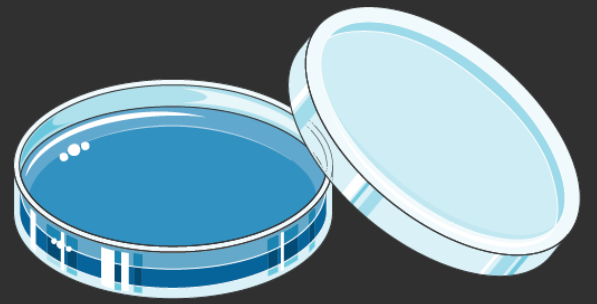
> “We don’t want to be first, but we can’t be third!”

But if you’re not starting now, you’ll already be behind when it matters

How do we get there?

Bottom up

- Learn it yourself
- Write some tools
- Start small and stay small
- Build a mini community, hold meetups
- Infect your organisation



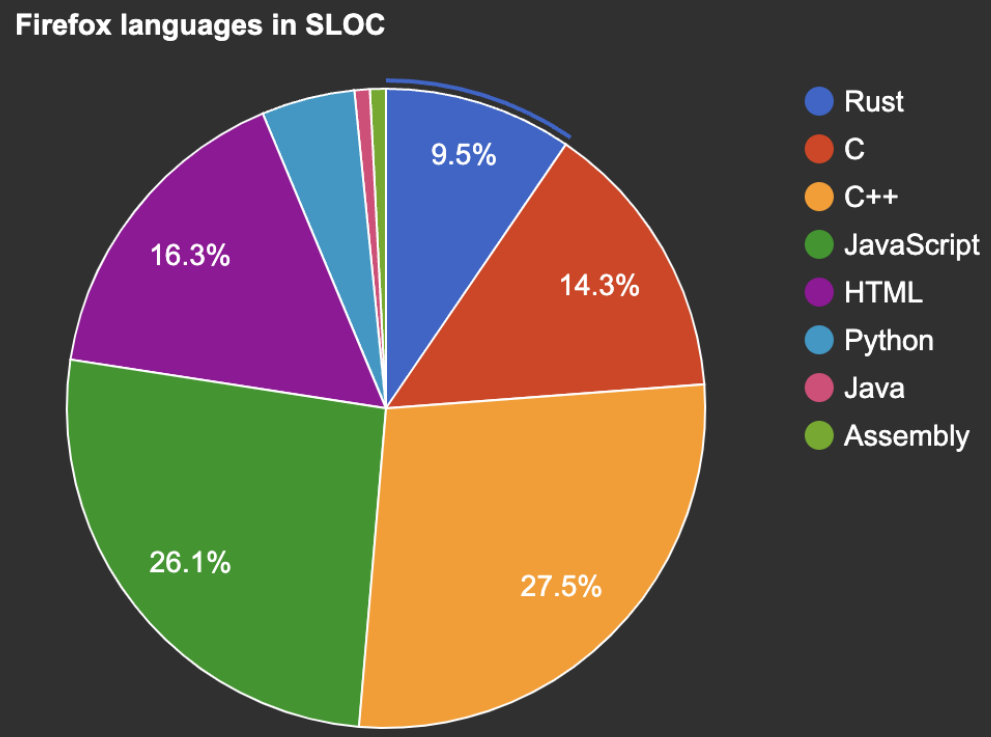
> Think of a petri dish — a culture pops up in a few places and then eventually it's everywhere

Also...

> Don't “Rewrite it in Rust”™!

Rust was born at Mozilla in 2009...

...but in 2023, Firefox is still less than 10% Rust



Let's have a break!

Let's talk Crux!

> Crux is a great example of how Rust can help the enterprise build better quality software with less effort (cost) and more Joy™

1. **What** is the problem with multi-platform app development today?
2. **Rust**, **WebAssembly**, and **Ports and Adapters**
3. **Crux** — experimental, open source tooling for building **headless** apps

What is the problem with multi-platform app development today?

1 Building a multi-platform app (don't @ me!)

	Platform Native	Kotlin MM	React Native	Capacitor Ionic	Flutter
Native UX	✅	✅	😐	❌	❌
Web?	❌	😐	😐	✅	✅
Development	😐	✅	😐	✅	✅
Testing	😐	😐	🤯	🤯	😐
Maintenance	😐	✅	😡	😡	✅
Effort	3x	2x	2x	1.5x	1.4x

1 Tooling and Architecture

Building quality apps across all platforms is too hard

- In order to reuse code we end up **compromising** on UX and/or DX
- UI-centric **architectures** make applications **hard to test** and maintain

1 UI-centric architecture

- UI layout is the **primary** organising principle
- Behaviour and interaction with the outside world are **secondary**

> "It looks like this… and does that"

1. **What** is the problem with multi-platform app development today?
2. **Behaviour-centric** architecture
3. **Crux** — experimental, open source tooling for building **headless** apps

Behaviour-centric architecture

2 What if we start with behaviour?

But how do we model behaviour?

- update a **model** in response to **events**
- emit **effects** — intent to perform side-effects

2 Behaviour

Update state when an Event is raised

```
fn update(event: Event, state: Model) -> Model {
  // perform HTTP request
}
```

2 Behaviour

A pure update function (cf. Elm, Redux, etc.)

```
fn update(event: Event, state: Model) -> (Model, Vec<Effect>)
```

A dirty function with side-effects

```
fn http(effect: Effect) { /* perform HTTP request */ }
```

2 UI

Imagine the UI as a projection of state (cf. early React)

```
fn view(state: Model) { /* update UI */ }
```

❷ UI

A pure view function

```
fn view(state: Model) -> ViewModel
```

A dirty function — UI is a side-effect

```
fn render(view: ViewModel) { /* update UI */ }
```

2 Before

Behavior

```
fn update(event: Event, state: Model) -> (Model, Vec<Effect>)
```

```
fn http(effect: Effect) { /* perform HTTP request */ }
```

UI

```
fn view(state: Model) -> ViewModel
```

```
fn render(view: ViewModel) { /* update UI */ }
```

❷ After

Core (pure)

```
fn update(event: Event, state: Model) -> (Model, Vec<Effect>)
```

```
fn view(state: Model) -> ViewModel
```

Shell (dirty)

```
fn http(effect: Effect) { /* perform HTTP request */ }
```

```
fn render(view: ViewModel) { /* update UI */ }
```

2 Behaviour-centric architecture

- Behaviour is the **primary** organising principle
- Interaction with the outside world is **secondary**
- UI is also a side-effect

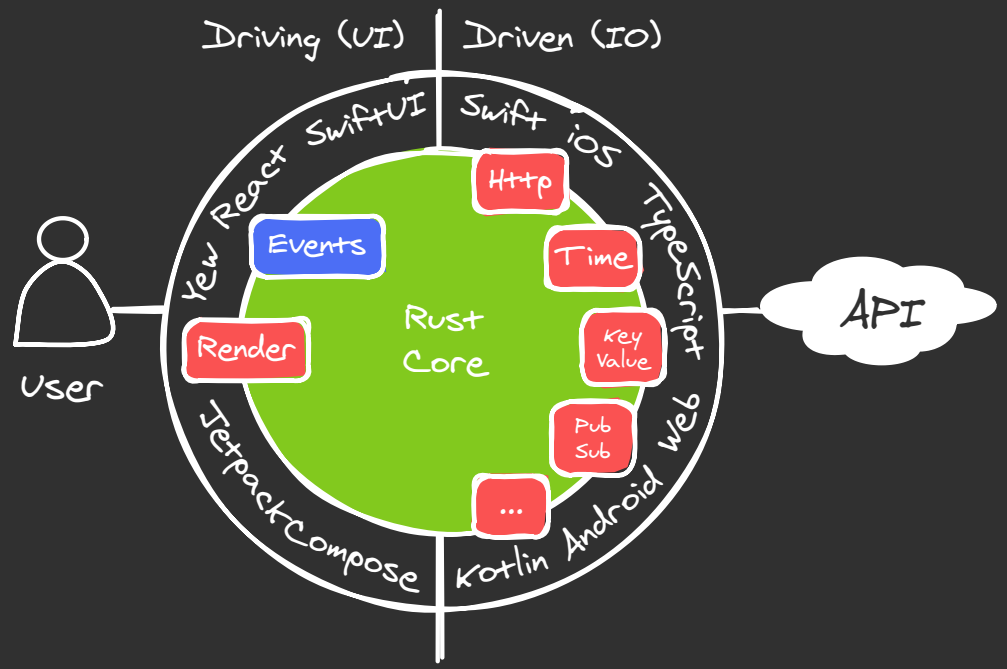
> "It does this... and looks like that!"

2 Ports and adapters

> Allow an application to equally be driven by users, programs, automated test or batch scripts, and to be developed and tested in isolation from its eventual run-time devices and databases.

Alistair Cockburn, 2005

Hexagonal Architecture



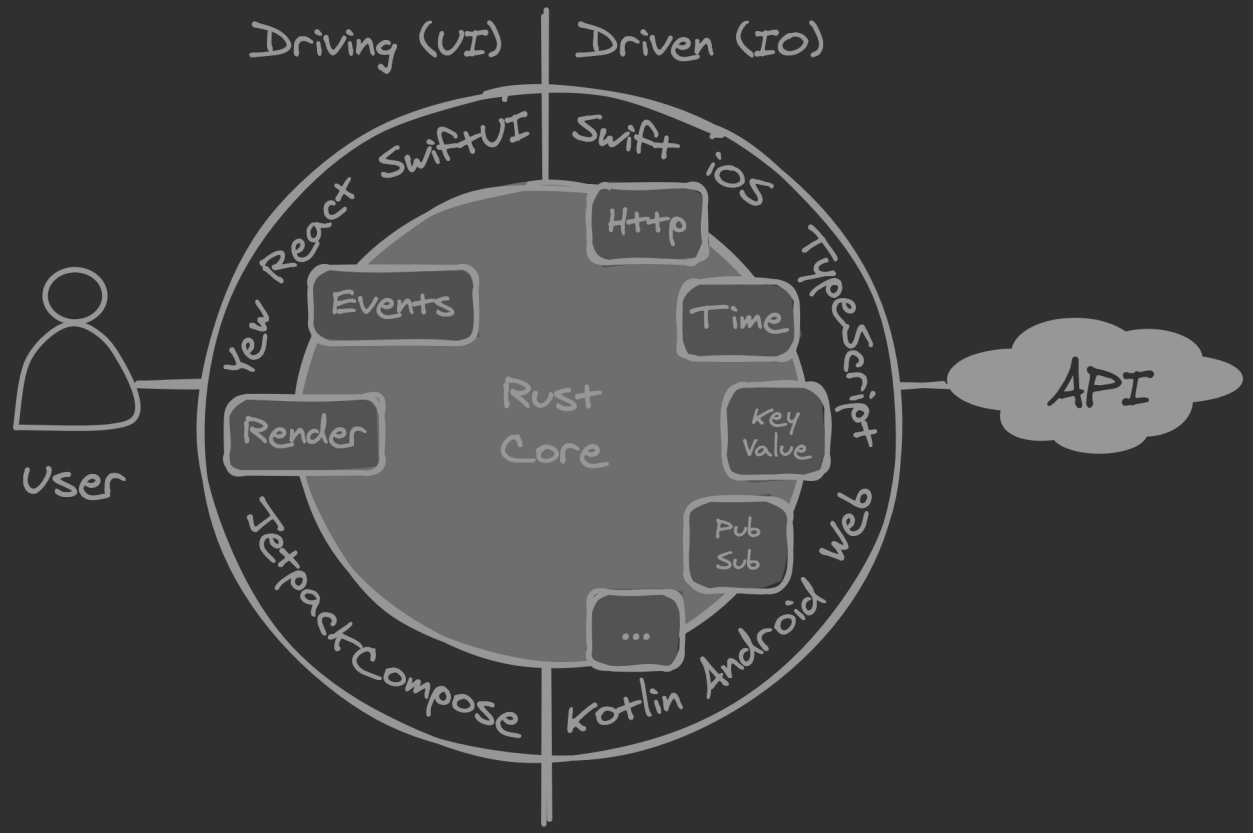
2 Ports and adapters

> The application can be deployed in **headless** mode, so only the API is available, and other programs can make use of its functionality

Alistair Cockburn, 2005

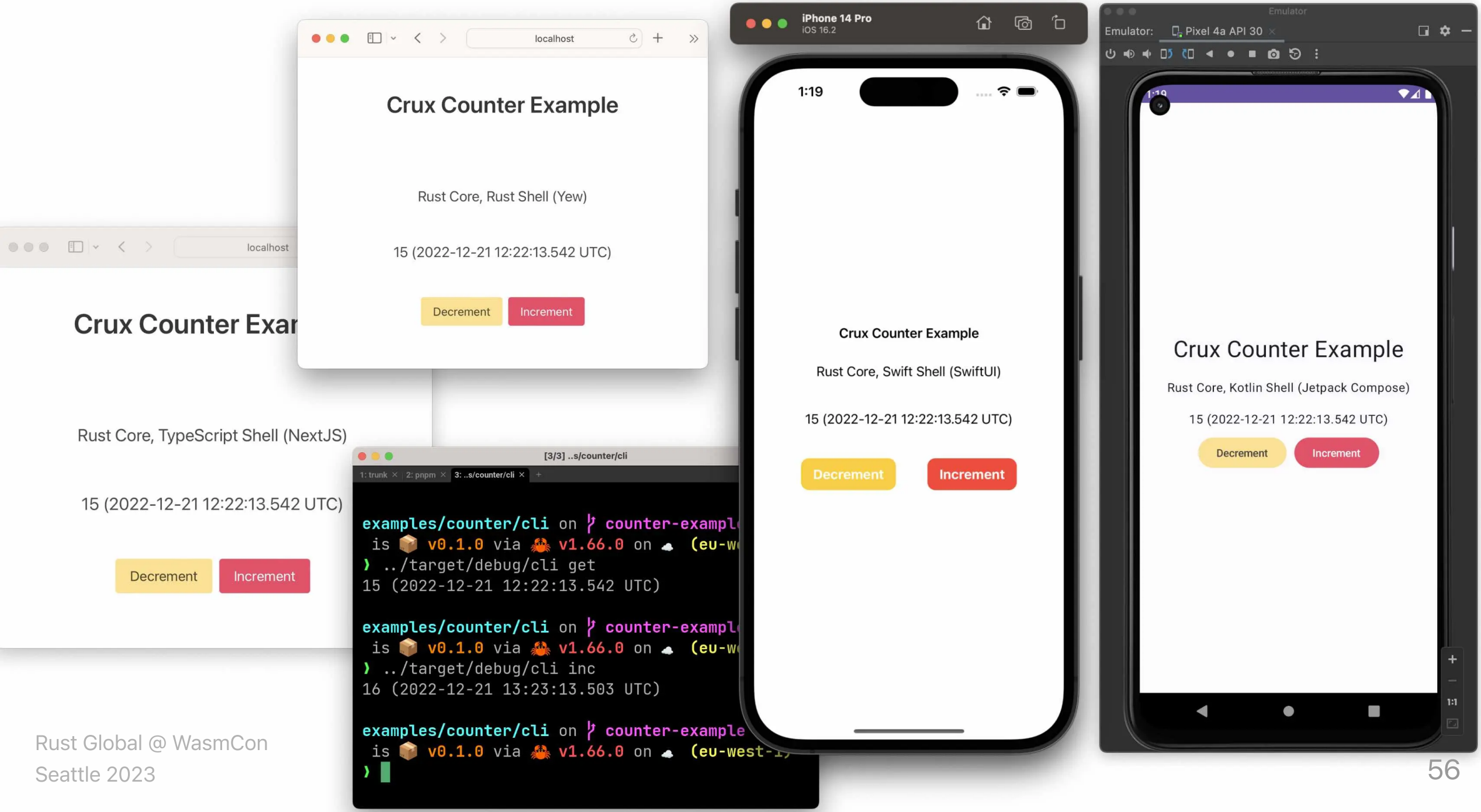
Hexagonal Architecture

1. **What** is the problem with multi-platform app development today?
2. **Behaviour-centric** architecture
3. **Crux** — experimental, open source tooling for building **headless** apps



Crux

- Shared **behaviour**
- in **Rust**
- Platform **native** UX


localhost
Crux Counter Example
Rust Core, Rust Shell (Yew)
15 (2022-12-21 12:22:13.542 UTC)
Decrement
Increment
localhost
Crux Counter Exar
Rust Core, TypeScript Shell (NextJS)
15 (2022-12-21 12:22:13.542 UTC)
Decrement
Increment
[3/3] ..s/counter/cli
1: trunk
2: pnpm
3: ..s/counter/cli
examples/counter/cli on counter-example
is v0.1.0 via v1.66.0 on (eu-w
../target/debug/cli get
15 (2022-12-21 12:22:13.542 UTC)
examples/counter/cli on counter-example
is v0.1.0 via v1.66.0 on (eu-w
../target/debug/cli inc
16 (2022-12-21 13:23:13.503 UTC)
examples/counter/cli on counter-example
is v0.1.0 via v1.66.0 on (eu-west-1)
iPhone 14 Pro
iOS 16.2
1:19
Crux Counter Example
Rust Core, Swift Shell (SwiftUI)
15 (2022-12-21 12:22:13.542 UTC)
Decrement
Increment
Emulator
Emulator: Pixel 4a API 30
Crux Counter Example
Rust Core, Kotlin Shell (Jetpack Compose)
15 (2022-12-21 12:22:13.542 UTC)
Decrement
Increment

3 Any platform

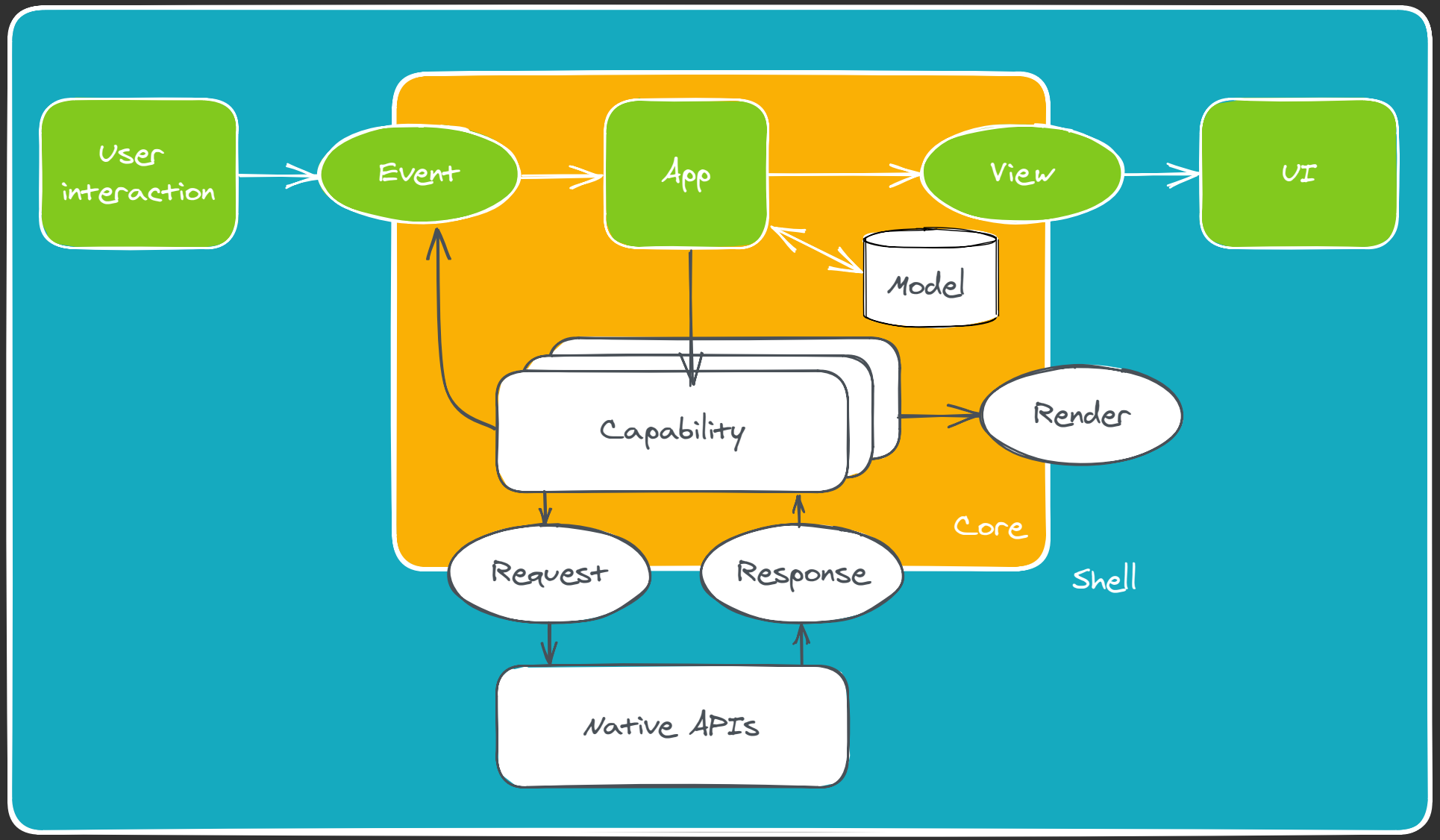
platform	language	UI	library	lib name	FFI
iOS	Swift	SwiftUI	static	`libshared.a`	`uniffi-bindgen`
Android	Kotlin	Compose	dynamic	`libshared.so`	`uniffi-bindgen`
Web	TypeScript	Remix	**wasm**	`shared.wasm`	`wasm-bindgen`
Web	Rust	Leptos	crate		
CLI	Rust	`println!()`	crate		

3 FFI

```
namespace shared {
  bytes process_event([ByRef] bytes msg);
  bytes handle_response([ByRef] bytes uuid, [ByRef] bytes res);
  bytes view();
};
```

Type generation with `serde-generate`

3


User interaction
Event
App
View
UI
Model
Capability
Render
Core
Request
Response
Shell
Native APIs

3 Capabilities

```
// fire and forget
caps.render
  .render();

// request/response
caps.http
  .post(API_URL)
  .header("Authorization", token)
  .body_json(json)
  .expect("could not serialize body")
  .expect_json()
  .send(Event::Created);

// streamed responses
caps.sse
  .get_json(API_URL, Event::Update);
```

3 Capabilities

- Built-in (`Render`)
- `crux_*` crates (`Http` , `KeyValue` , `Platform` , `Time`)
- Custom
  - `ServerSentEvents` in the Counter example
  - `Delay` example in the book
  - `Timer` and `PubSub` in the Notes example
- Community contributed

3 What does a Crux app look like?

```rust
#[derive(Default)]
pub struct App;

impl crux_core::App for App {
    type Event = Event;
    type Model = Model;
    type ViewModel = ViewModel;
    type Capabilities = Capabilities;

    fn update(&self, event: Self::Event, model: &mut Self::Model, caps: &Self::Capabilities) {
        match event {
            Event::Increment => model.count += 1,
            Event::Decrement => model.count -= 1,
            Event::Reset => model.count = 0,
        };

        caps.render.render();
    }

    fn view(&self, model: &Self::Model) -> Self::ViewModel {
        ViewModel {
            count: format!("Count is: {}", model.count),
        }
    }
}
```

3 What does testing look like?

3 What does a test look like?

```rust
#[cfg(test)]
mod test {
    use super::*;
    use crux_core::testing::AppTester;

    #[test]
    fn increments_count() {
        let app = AppTester::<App, _>::default();
        let mut model = Model::default();

        let update = app.update(Event::Increment, &mut model);

        // Check the app asked us to `Render`
        assert_effect!(update, Effect::Render(_));

        // Check view model is correct
        let actual_view = app.view(&model).count;
        let expected_view = "Count is: 1";
        assert_eq!(actual_view, expected_view);
    }
}
```

3 17ms

```
crux/examples/notes on  remix [!] via 🦀 v1.71.0 on ☁️  (eu-west-1)
❯ cargo nextest run
    Finished test [unoptimized + debuginfo] target(s) in 0.06s
    Starting 25 tests across 3 binaries
        PASS [   0.005s] shared app::editing_tests::removes_selection_on_backspace
        PASS [   0.006s] shared app::editing_tests::inserts_text_at_cursor_and_renders
        PASS [   0.006s] shared app::editing_tests::changes_selection
        PASS [   0.006s] shared app::editing_tests::handles_emoji
        PASS [   0.005s] shared app::editing_tests::renders_text_and_cursor
        PASS [   0.006s] shared app::editing_tests::removes_character_after_cursor
        PASS [   0.006s] shared app::editing_tests::removes_selection_on_delete
        PASS [   0.006s] shared app::editing_tests::removes_character_before_cursor
        PASS [   0.006s] shared app::editing_tests::moves_cursor
        PASS [   0.007s] shared app::editing_tests::replaces_empty_range_and_renders
        PASS [   0.005s] shared app::editing_tests::replaces_range_and_renders
        PASS [   0.005s] shared app::note::test::splices_text
        PASS [   0.005s] shared app::save_load_tests::creates_a_document_if_it_cant_open_one
        PASS [   0.006s] shared app::editing_tests::replaces_selection_and_renders
        PASS [   0.005s] shared app::note::test::inserts_text
        PASS [   0.005s] shared app::save_load_tests::opens_a_document
        PASS [   0.006s] shared app::save_load_tests::starts_a_timer_after_an_edit
        PASS [   0.007s] shared app::save_load_tests::saves_document_when_typing_stops
        PASS [   0.007s] shared app::sync_tests::concurrent_clean_edits
        PASS [   0.007s] shared app::sync_tests::concurrent_conflicting_edits
        PASS [   0.005s] shared app::sync_tests::one_way_sync
        PASS [   0.005s] shared app::sync_tests::receiving_own_edits
        PASS [   0.005s] shared app::sync_tests::two_way_sync
        PASS [   0.005s] shared app::sync_tests::remote_insert_behind_cursor
        PASS [   0.005s] shared app::sync_tests::remote_delete_moves_cursor
------------
     Summary [   0.017s] 25 tests run: 25 passed, 0 skipped
```

3 Demo

3 The crux of Crux

github.com/redbadger/**crux**

- **headless**, multi-platform, composable apps with shared **behaviour**
- better **testability**
- higher **quality** apps
- more **joy** from better tools

Why Rust?

- https://aws.amazon.com/blogs/opensource/sustainability-with-rust/
- https://www.wired.com/story/rust-secure-programming-language-memory-safe/
- https://content.red-badger.com/resources/how-to-use-rust-to-build-networked-services
- https://www.technologyreview.com/2023/02/14/1067869/rust-worlds-fastest-growing-programming-language
- https://www.techspot.com/news/97654-how-broken-elevator-led-one-most-loved-programming.html

Rust adoption in the enterprise

- https://www.infoq.com/presentations/rust-adoption-journey/
- https://rustmagazine.org/issue-1/2022-review-the-adoption-of-rust-in-business/
- https://vercel.com/blog/turborepo-migration-go-rust
- https://opensource.googleblog.com/2023/06/rust-fact-vs-fiction-5-insights-from-googles-rust-journey-2022.html
- https://www.infoq.com/articles/rust-ecosystem-review-2023

Crux

- https://github.com/redbadger/crux
- https://redbadger.github.io/crux/
- https://red-badger.com/crux
- https://docs.rs/crux_core/latest/crux_core/
- https://www.youtube.com/watch?v=cWCZms92-1g&t=5s

Thank you!

🙏

@stuartharris @charypar