Building iOS, Android and Web Apps that share a single Rust Core

2023-09-06

Stuart Harris

Founder & Chief Scientist

Red Badger

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

Stu

- Software engineer
- Founder of Red Badger

@stuartharris

RED BADGER

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. **Rust**, **WebAssembly**, and **Ports and Adapters**
3. **Crux** — experimental, open source tooling for building **headless** apps

Rust, WebAssembly, and Ports and Adapters

2 Better Tools and Better Architecture

- **Rust** is a **revolution**

  everyone can now build reliable, high quality software in almost any space — perfect for multi-platform app development

- **WebAssembly** is a **revolution**

  fast and portable — great for building apps in the languages we love

- **Ports and Adapters** is a **revolution**

  behaviour-centric architecture — leads to better testability

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 */ }
```

❷ 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 */ }
```

2 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 WebAssembly

WebAssembly helps us stay honest!

2 Behaviour-centric architecture

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

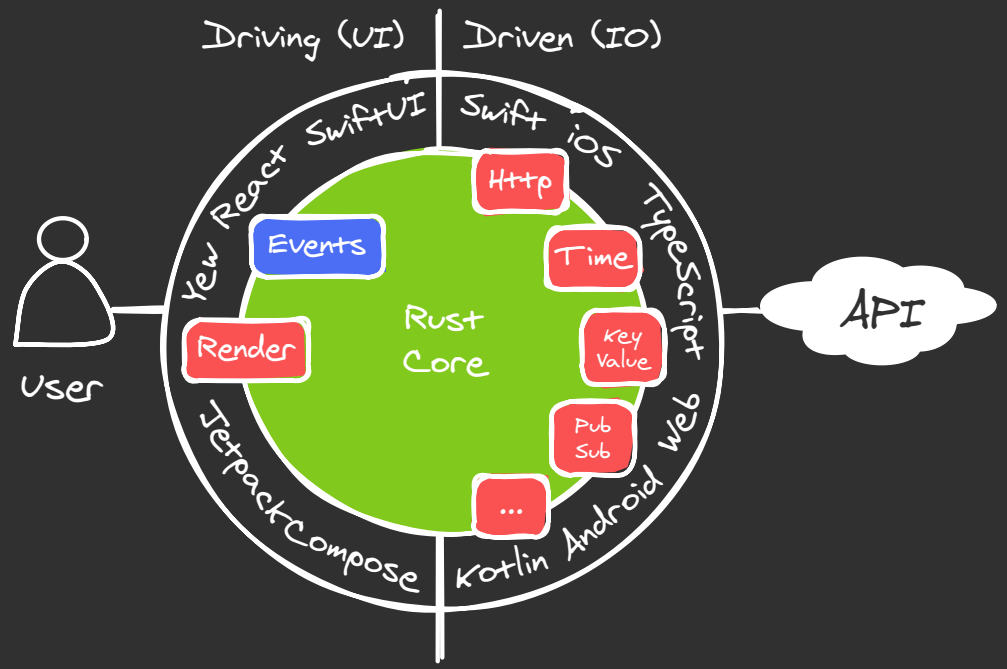
> "It does this... and looks like that!"

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



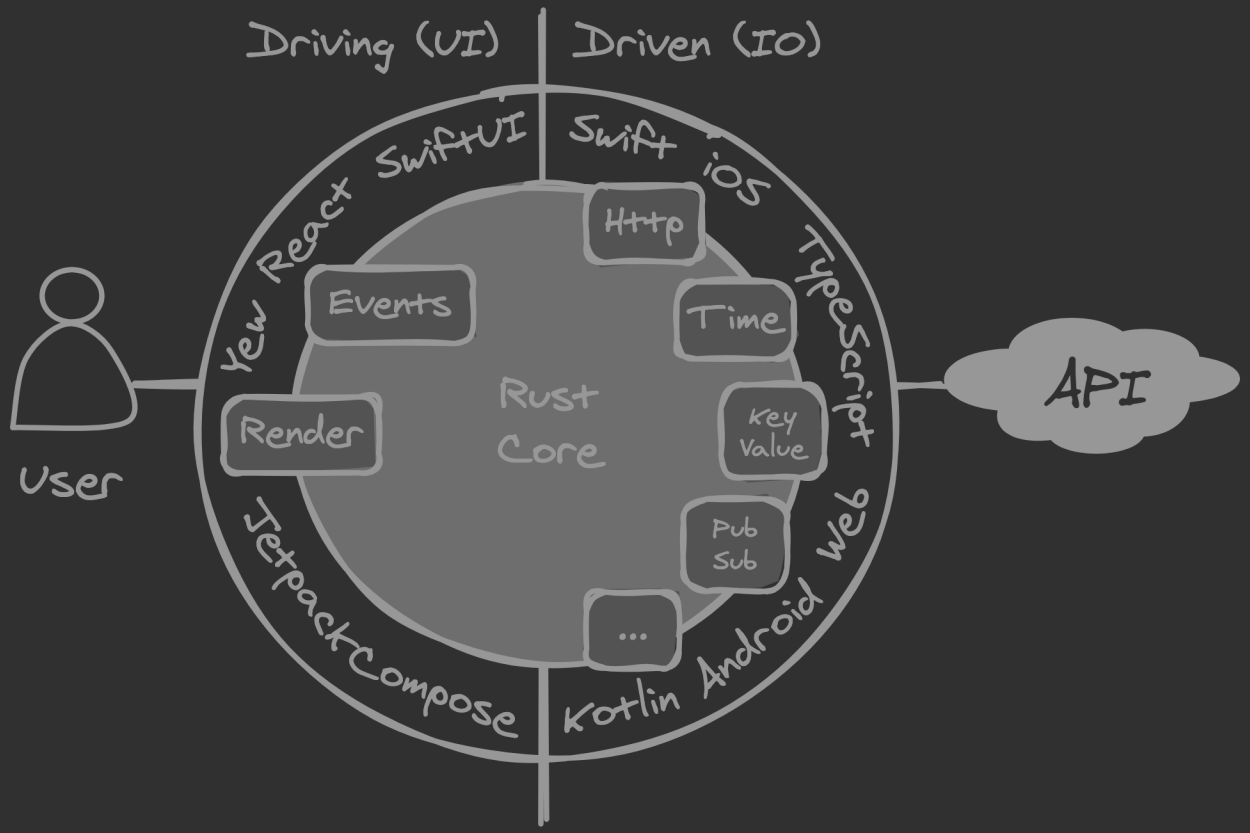
2 Ports and adapters

> 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

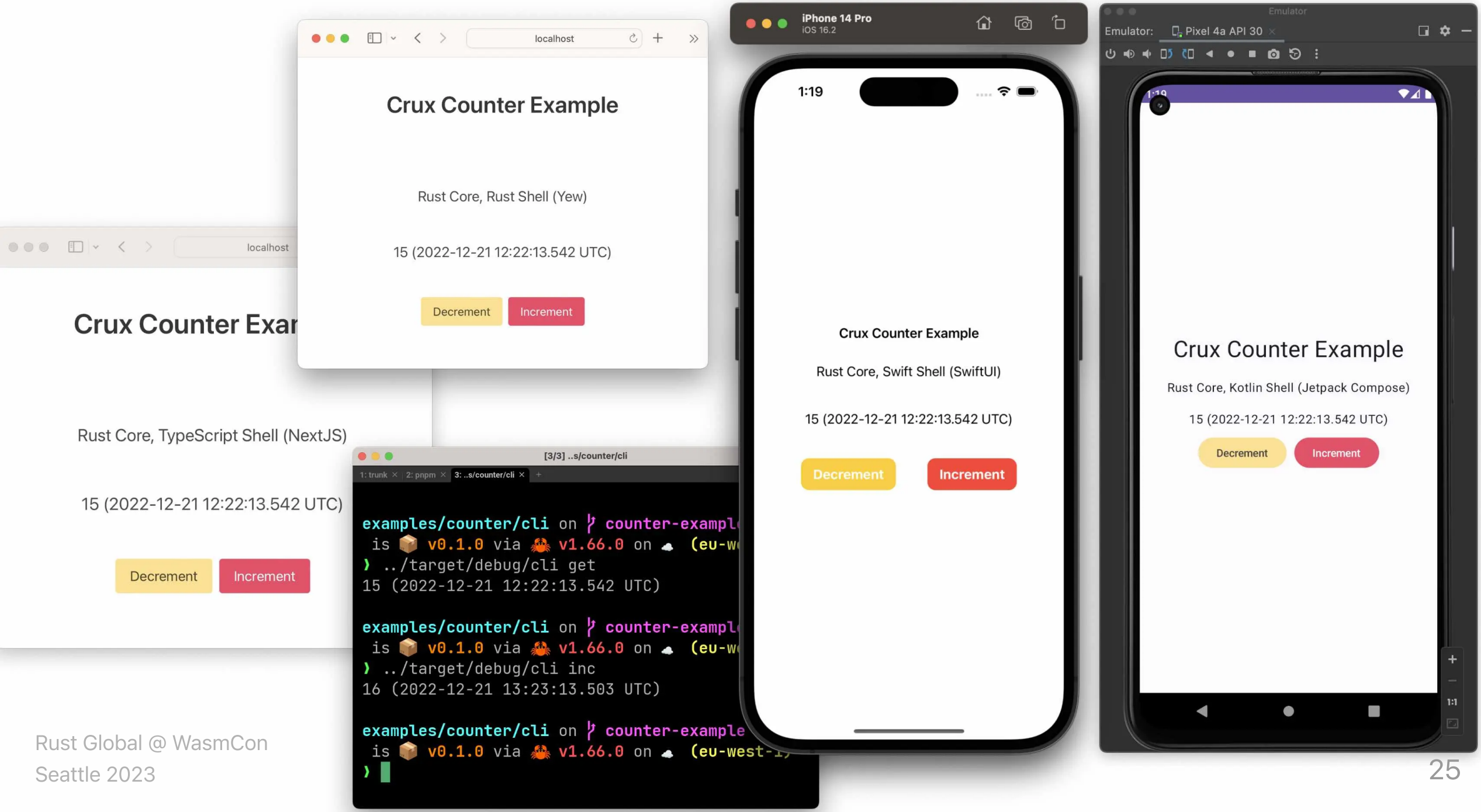
Hexagonal Architecture

1. **What** is the problem with multi-platform app development today?
2. **Rust**, **WebAssembly**, and **Ports and Adapters**
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

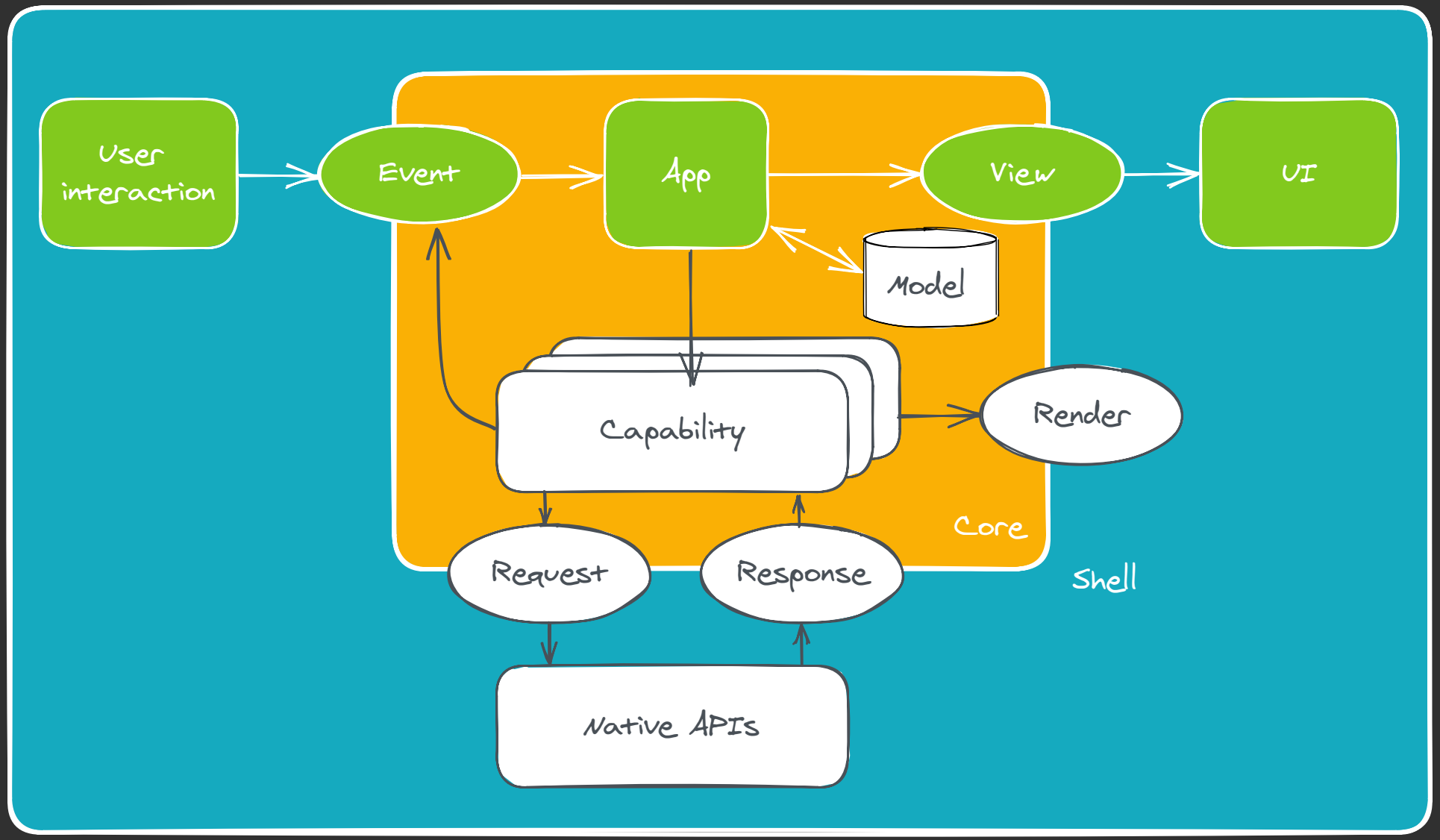
platform	language	UI	library	lib name	FFI
iOS	Swift	SwiftUI	static	`libshared.a`	`uniffi-bindgen`
Android	Kotlin	Compose	dynamic	`libshared.so`	`uniffi-bindgen`
Web	TypeScript	Remix	**wasm**	`shared.wasm`	`wasm-bindgen`
Web	Rust	Leptos	crate		
CLI	Rust	`println!()`	crate		

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::<Hello, _>::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 The crux of Crux

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

Thank you! 🙏

@stuartharris

Slides

Crux Book, Crux Github, Crux Docs, Crux Website

Rust Nation 2023 Talk