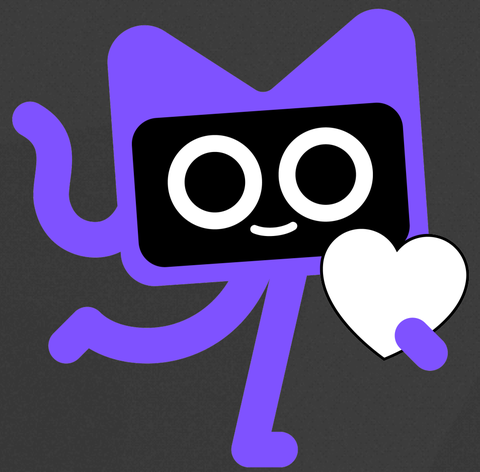
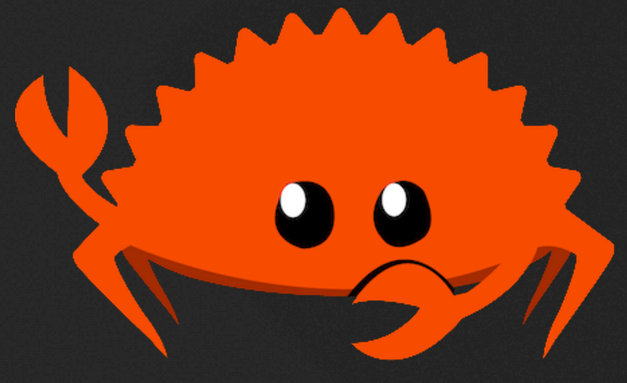
Seamless Multiplatform – Kotlin-Rust-Multiplatform?!

...or how to write a bug once to ship it everywhere*



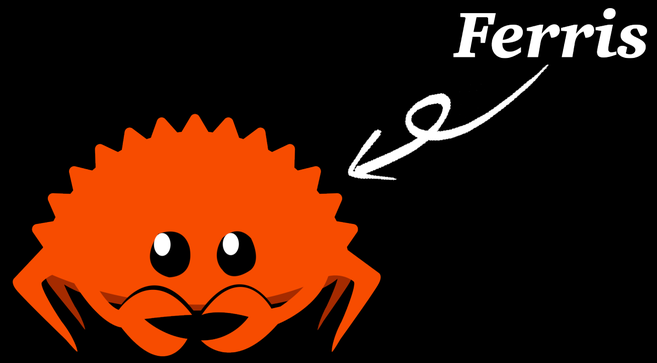


whoami

- Android engineer by day @ LichtBlick SE
- KMP believer by night
- Rust dilettante in my free time
- organizer for KUG & Rust & XTC Berlin

whoami

The Repo of the day...

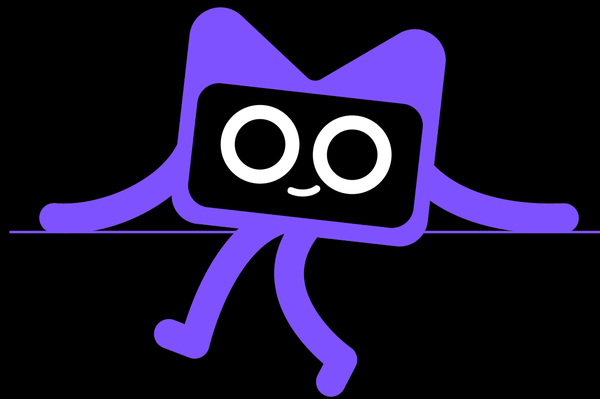


What this talk is about....

- why we want to integrate Kotlin/Rust
- how we integrate Kotlin/Rust
- The hypetrain in the unknown

...and what is not!

- in depth Rust Talk
- a ready to go solution
- a silverbullet
- arcane black magic



And before we start

tell me a joke about Android development 👩‍💻

🤖 Sure, here's a joke for you:

Why do Android developers prefer espresso?

Because it helps them keep perky during long coding sessions!

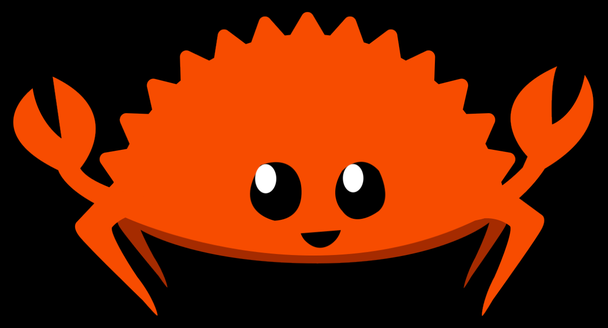
can you give me another one? 👩‍💻

🤖 Of course, here's another joke for you:

Why did the Android developer go broke?

Because he couldn't find a way to monetize his app!

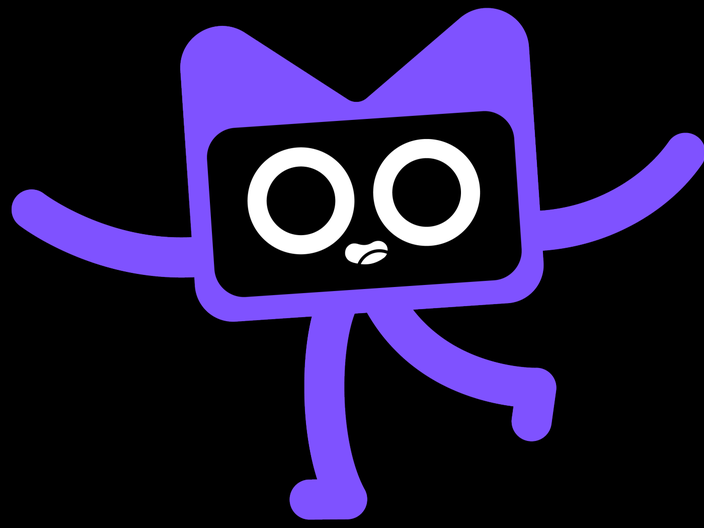
okay one more 👩‍💻



Why even consider Rust with Kotlin Multiplatform?

...and what does multiplatform even mean?

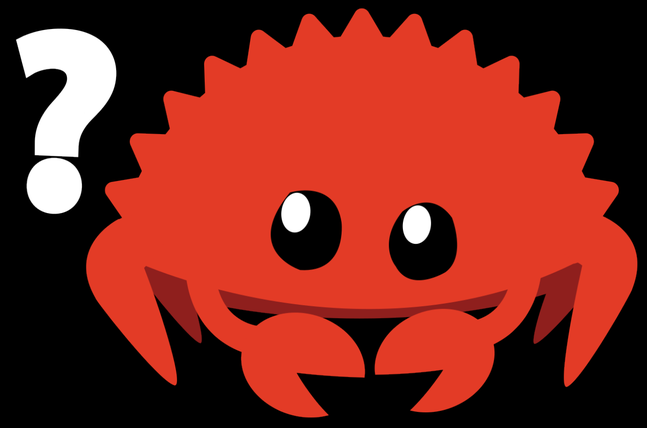
- native platform bindings instead virtual environment
- write code once and ship it to as many platforms as possible
- fallback to the platform to accomplish tasks, which are coupled with the platform



Why even consider Rust with Kotlin Multiplatform?

...but what is the goal?

- (ideally) one language to rule them all (and fits the needs of everyone)
- building things over an entire stack, different devices and os's
- sharing is caring and frees up space for other things

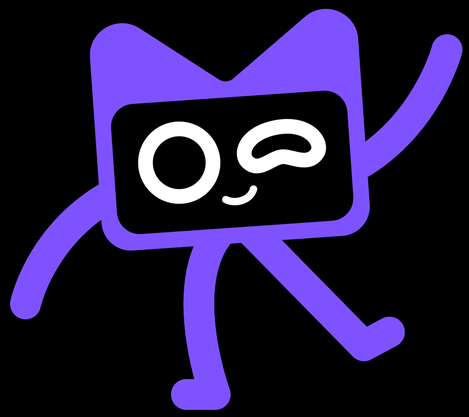


Why even consider Rust with Kotlin Multiplatform?

...is Kotlin alone not sufficient?

Kotlin is the best language in the world but:

- might leaks functionality you need (like crypto)
- you need comprehensive low-level bindings (like ML)
- you have rigorous resource constraints
- has no or not good enough support for your target



Why even consider Rust with Kotlin Multiplatform?

...can you tell more about this „Rust“?

Rust is awesome:

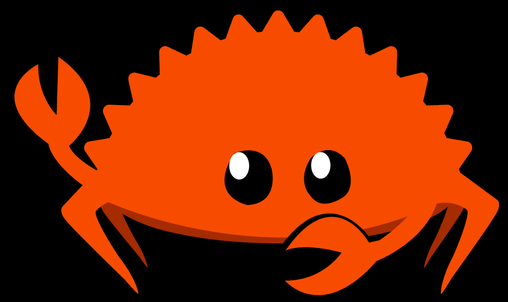
- memory safe*
- abstraction without overhead
- concurrency without fear
- feels hight-level while being low-level
- supreme WASM support

Why even consider Rust with Kotlin Multiplatform?

...so everything Rust from now on?

Rust is love but:

- it is a low-level language
- introduces complexity you do not need/want
- the tooling for (mobile) frontend is not there (yet)
- do you really want frontend Rust?

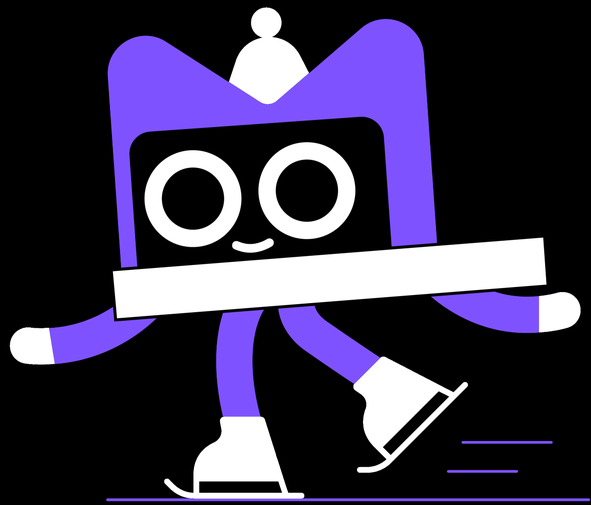


Why even consider Rust with Kotlin Multiplatform?

...is there more?

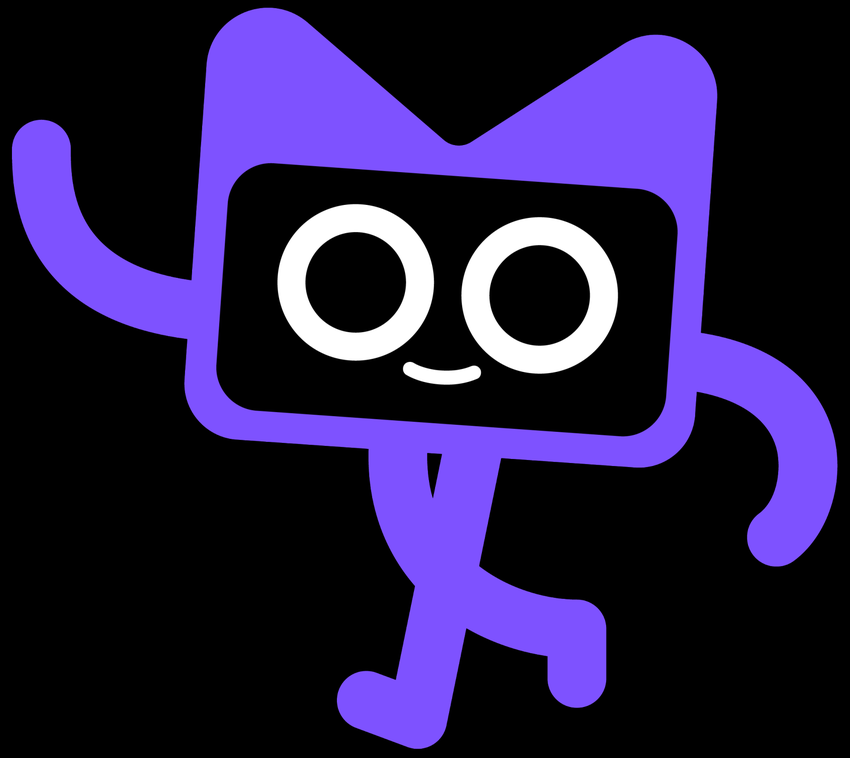
Business is Business:

- Rust is a requirement
- Kotlin is by far more widely adopted
- Biases & Goals & Team & Problems



Let’s look at some code

Hello World!



```
fun main() {
    println("Hello World!")
}
```

```rust
fn main() {
    println!("Hello World!");
}
```

Let’s look at some code

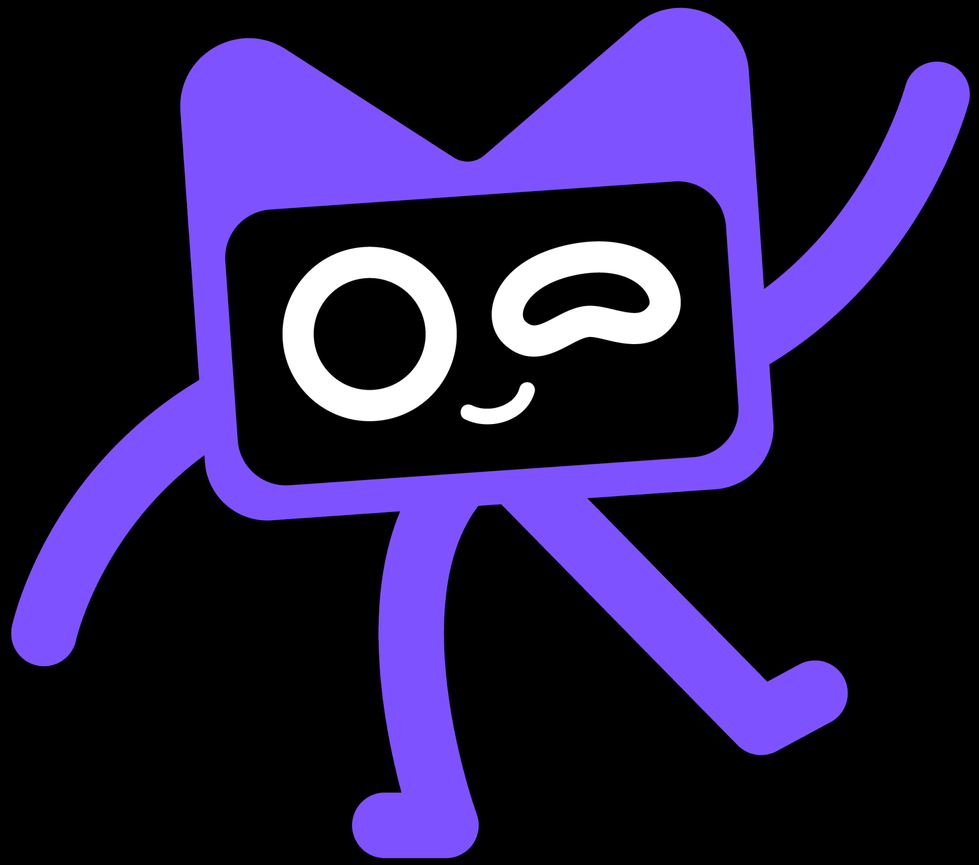
A bit more functions please!

```
fun main() {
    println(guess(23))
}

fun guess(number: Int): String {
    return if (number == 42) {
        "Of course!"
    } else {
        "Naaa, try again!"
    }
}
```

```rust
fn main() {
    println!("{}", guess(23));
}

pub fn guess(number: i32) -> String {
    if number == 42 {
        "Of course!"
    } else {
        "Naaa, try again!"
    }.to_string()
}
```

Let’s look at some code

And Annotations/Attributes.



```
fun main() {
    println(guess(23))
}

@HelloWorld
fun guess(number: Int): String {
    return if (number == 42) {
        "Of course!"
    } else {
        "Naaa, try again!"
    }
}
```

```rust
fn main() {
    println!("{}", guess(23));
}

#[HelloWorld]
pub fn guess(number: i32) -> String {
    if number == 42 {
        "Of course!"
    } else {
        "Naaa, try again!"
    }.to_string()
}
```

Let’s look at some code

That’s all we need for now, arrrrr!

How we glue now both together?

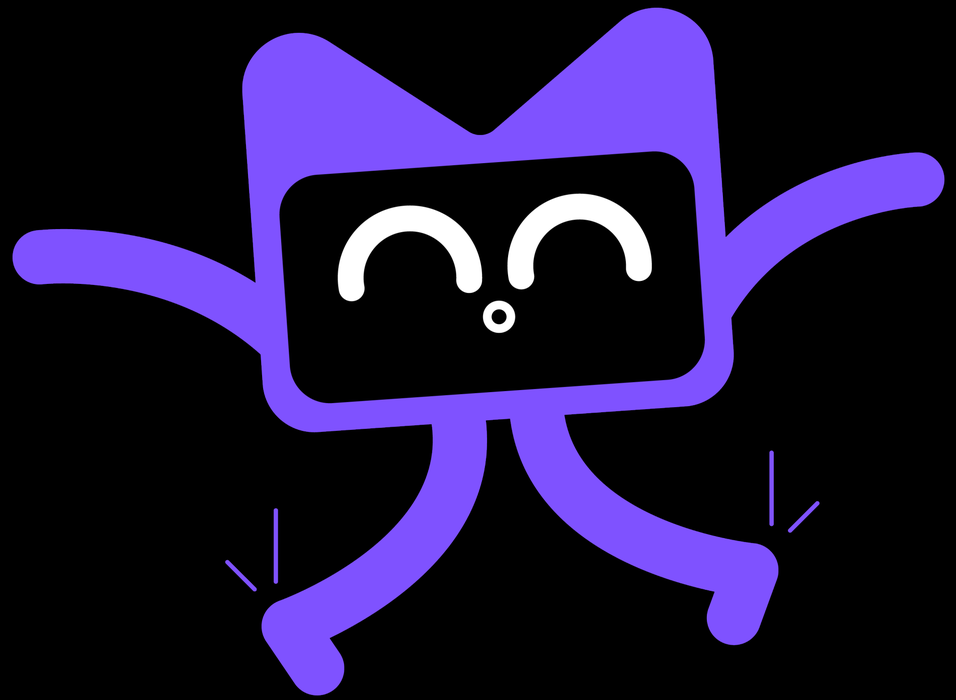
...and where do we start?

Let's „build“ an App:

- for iOS, Android, Web and Desktop
- which shows images of a cat which are coming from a backend
- which shows a pur factor which is a crazy long random number
- only (almost) Kotlin and Rust are allowed

How we glue now both together?

...and so it begins...



IDE Time

How we glue now both together?

JVM/Android

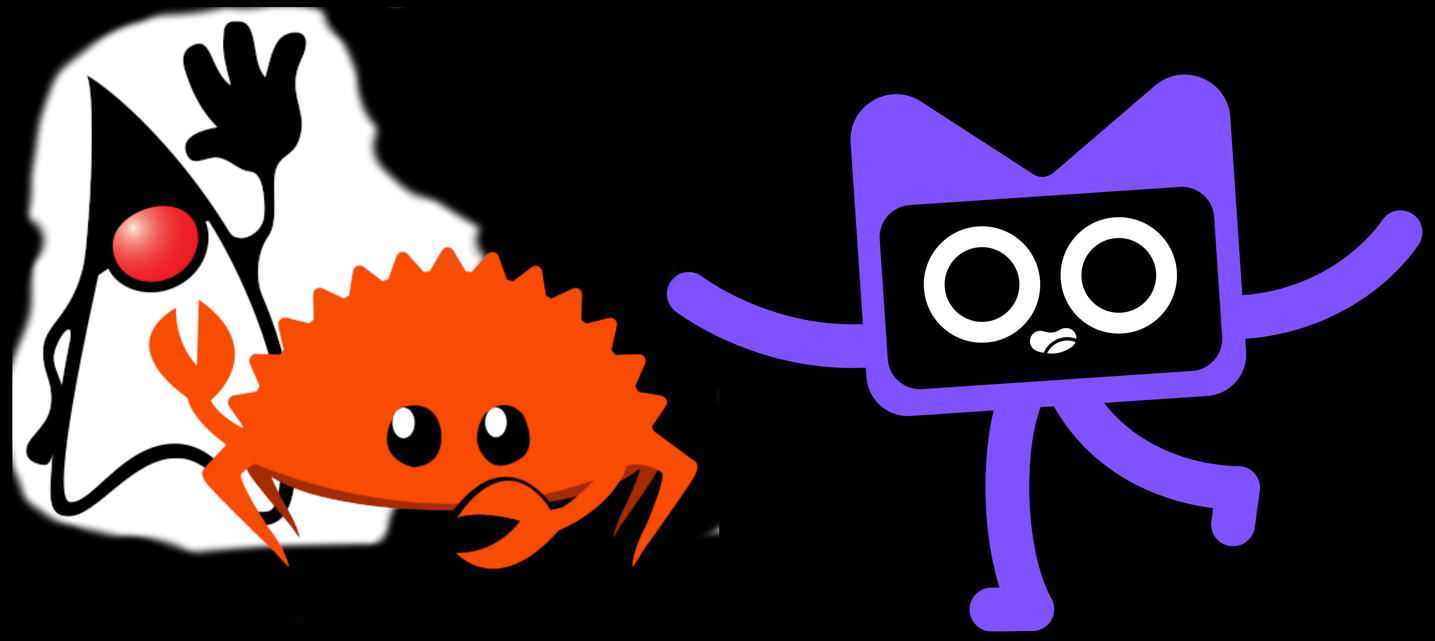
=> Flapigen

=> JNI

=> NDK Crate

=> Native Lib Loader

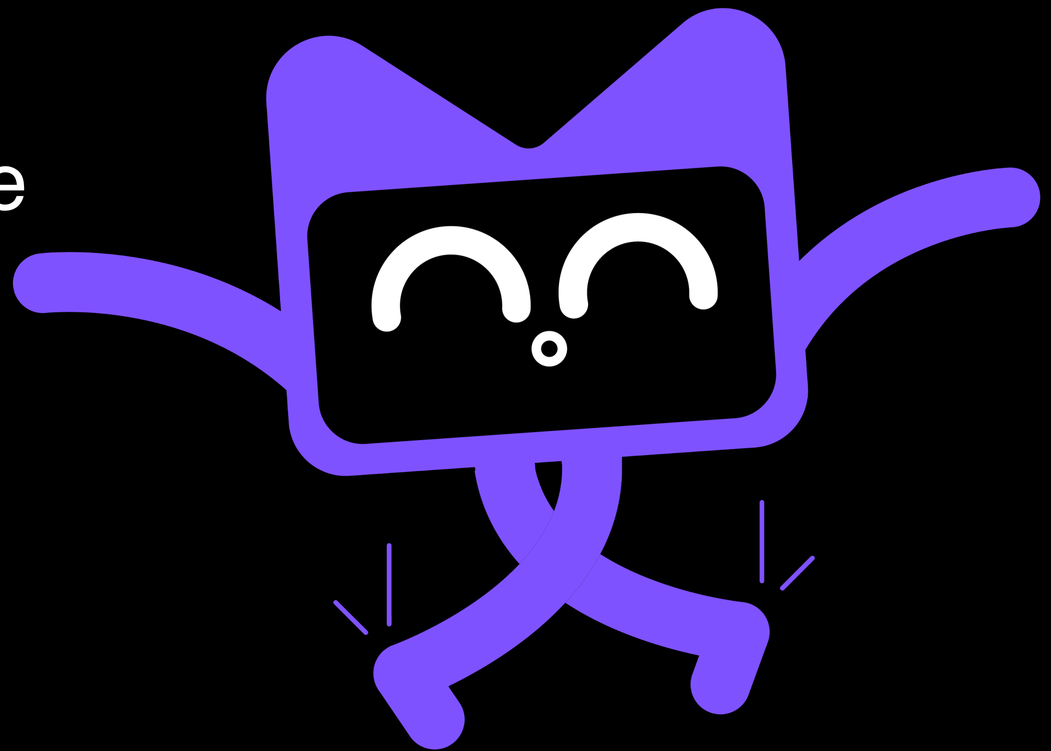
=> Mozilla's Rust Gradle Plugin



How we glue now both together?

...round 2

IDE Time



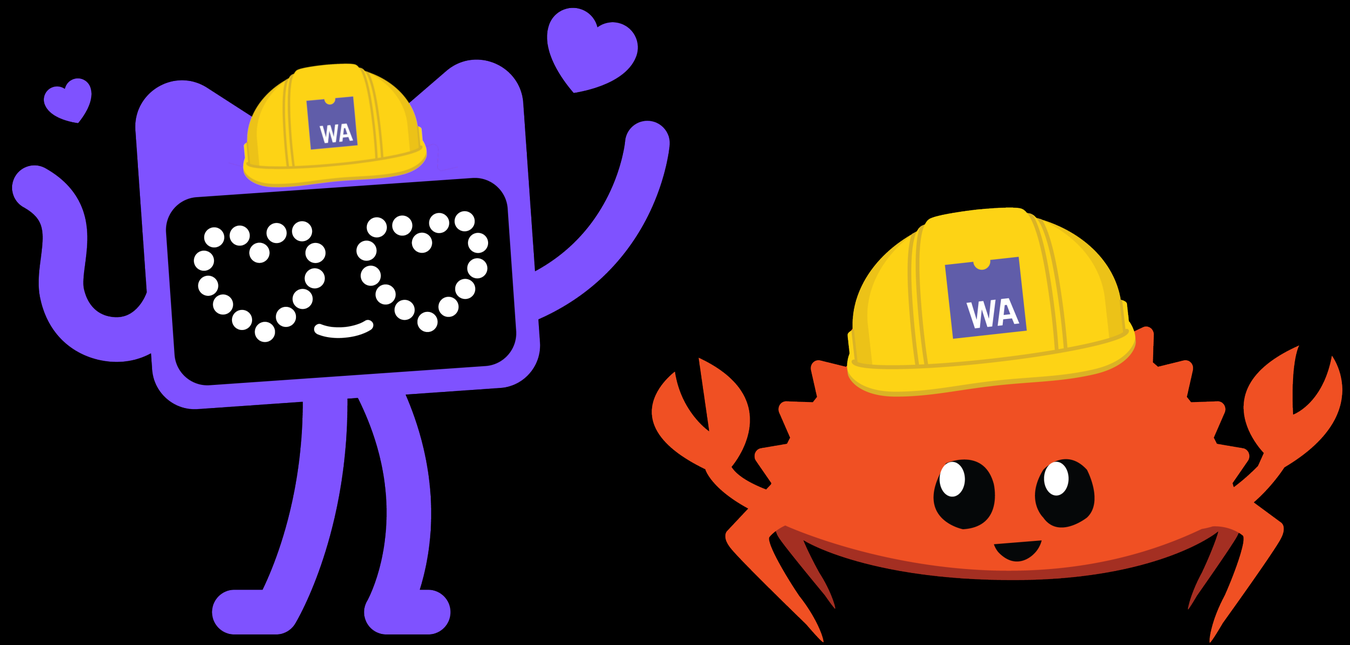
How we glue now both together?

JS/WASM

=> wasm-pack

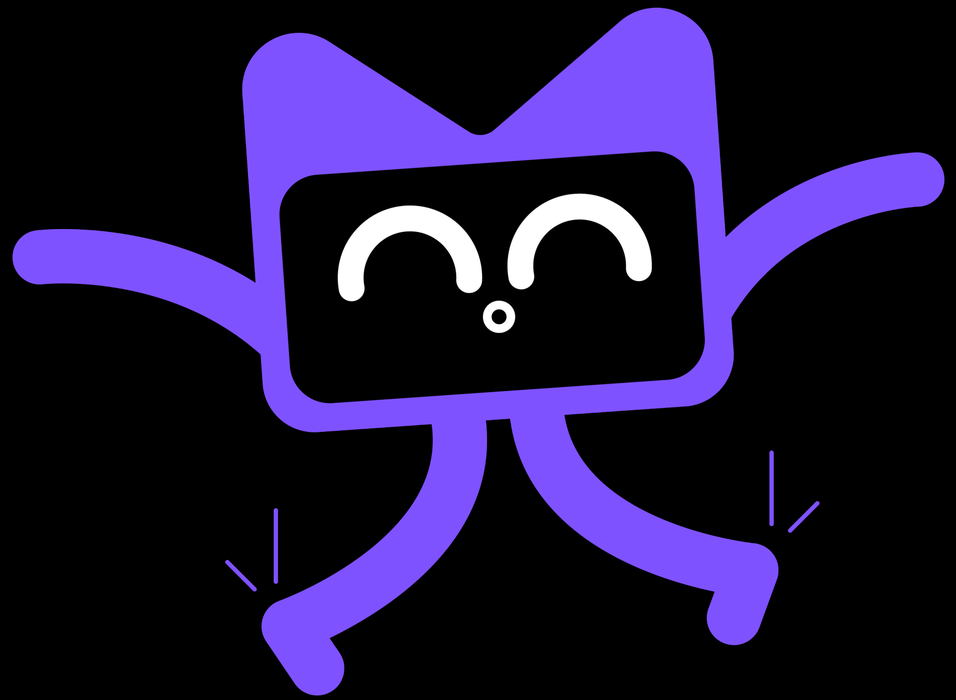
=> understanding of promises

=> a bit of webpack



How we glue now both together?

...round 3



IDE Time

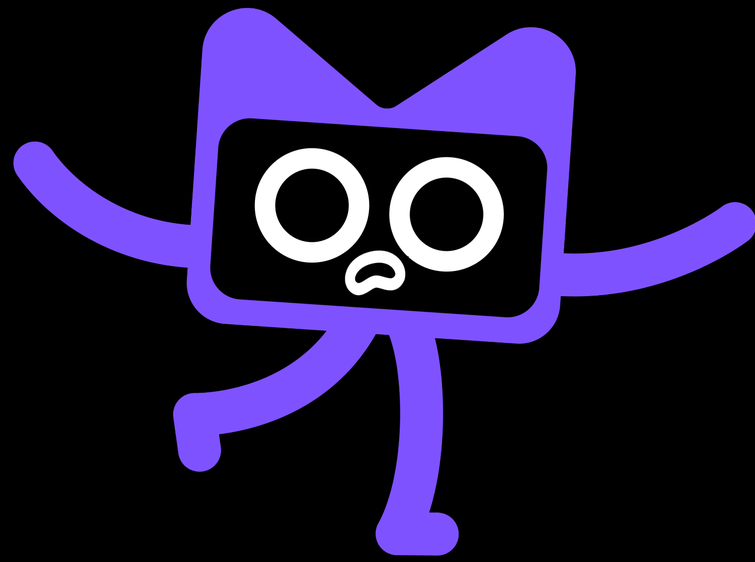
How we glue now both together?

Native

=> C-Bridge

=> Patients

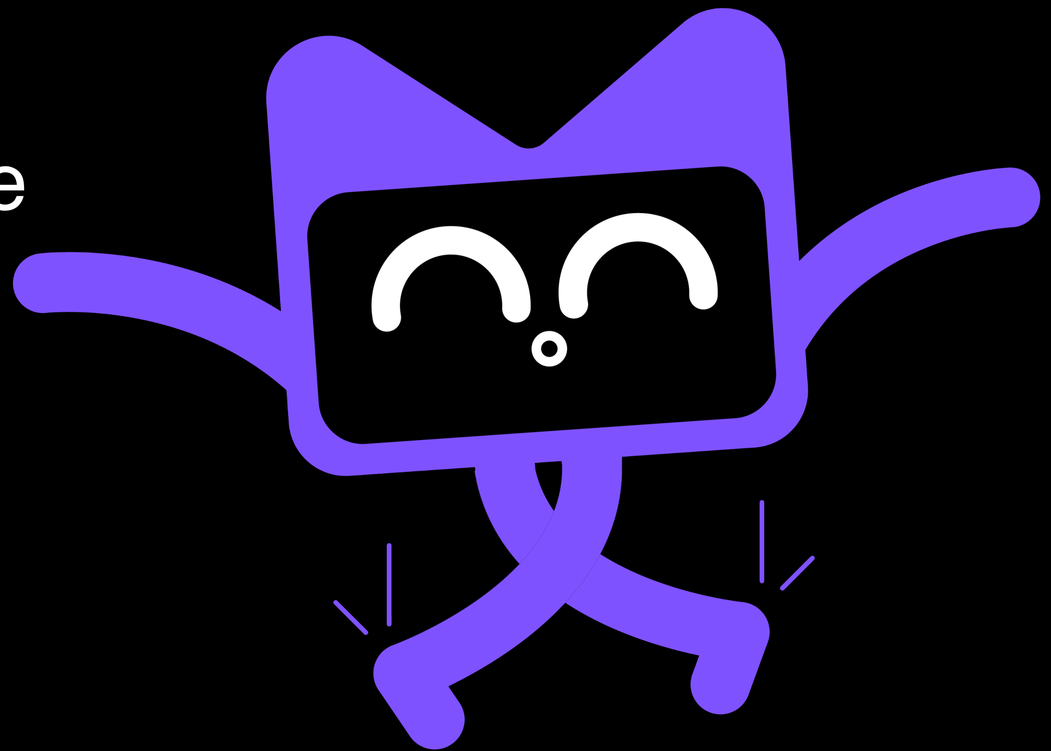
=> Time



How we glue now both together?

...round 4

IDE Time



Lessons learned

...there is an elephant in your build

We need:

- of course Gradle!!!!
- data handling Strategy (JSON, Protobuf...)
- concise interfaces
- tests!!!!

Lessons learned

...and traps, and pitfalls

- Debugging is tricky
- C – Bridge on Native + Handy work = Error prone
- The need for async invocations in JS
- Complex data is hard to deal with
- Mixing up Concurrency is a bad idea

Lessons learned

...and traps, and pitfalls

- it is almost a green field
- there is a learning curve and different biases (which have to be respected)
- ...

Helpful things

...for mostly Rust

- Googles Comprehensive Rust
- Mara Bos especially about Atomics and Locks
- Official Learning Resources & rustup
- No Boilerplate
- Join the your nearest KUG and Rust Community

Helpful things

...beyond this talk

- try UniFFI
- look at Trixnity
- TypeShare
- RUI
- find something on AreWeGUIYet
- Rewatch Supercharging Your Flutter Apps with Rust

Helpful things

...almost there but take this Demo!

Thank you! Questions?