Leveraging Rust to build cross-platform mobile libraries

Rust Linz April 2021 - Jan-Erik / @badboy_

Leveraging Rust to build cross-platform ~~mobile~~ libraries

Rust Linz April 2021 - Jan-Erik / @badboy_

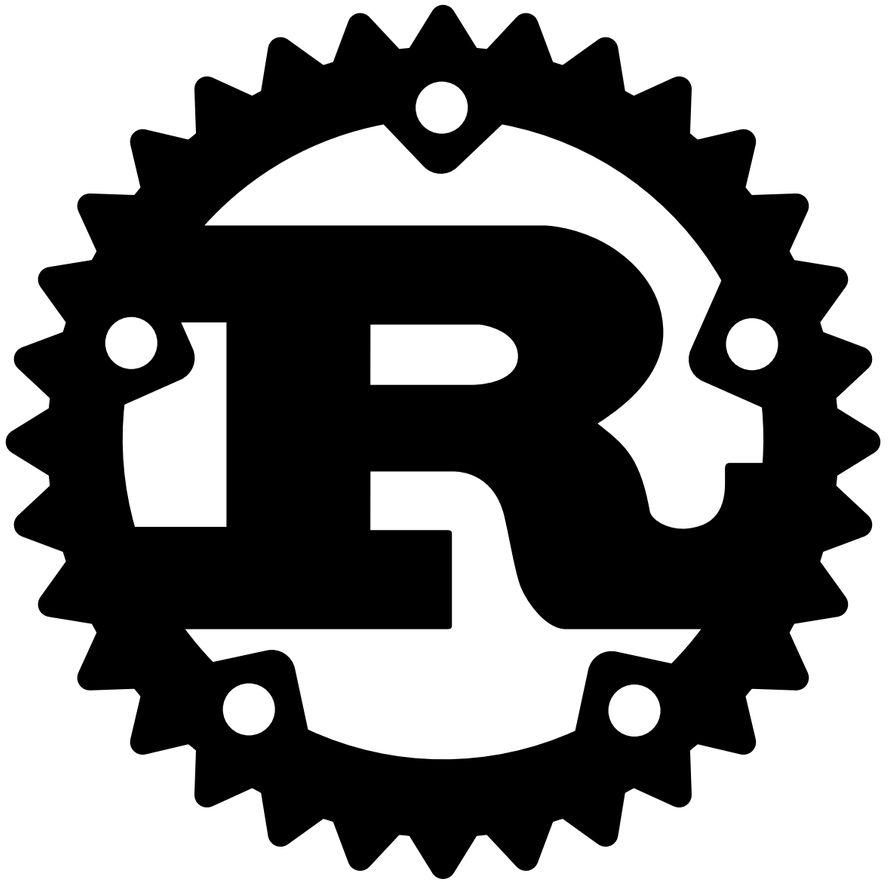
About me

- Firefox Telemetry engineer at Mozilla
- Rust Community Team member
- Scuba diver

Twitter: @badboy_
Blog: fnordig.de

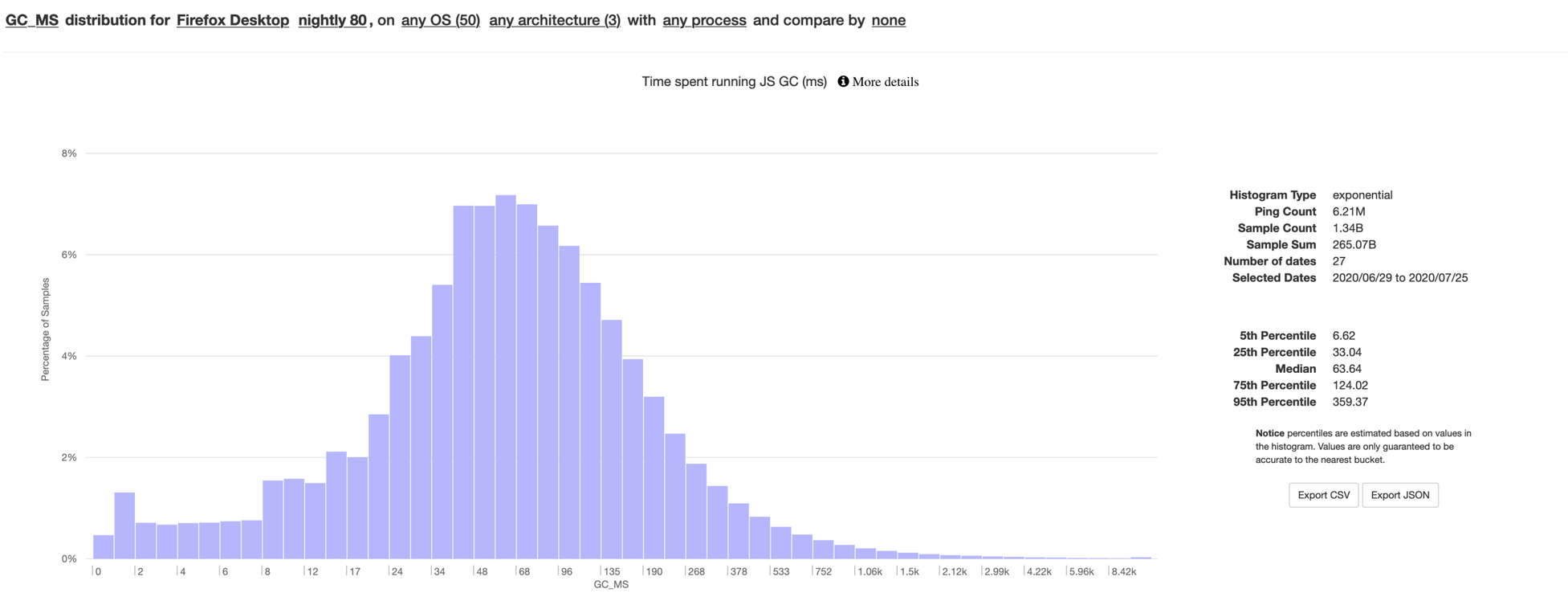
Slides:

fnordig.de/talks/2021/rustlinz/slides.pdf



https://github.com/mozilla/glean

Metric: Time spent running the JS GC[1]



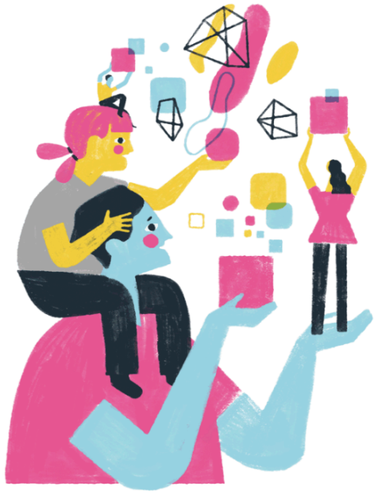
[1] Measurement Dashboard at https://telemetry.mozilla.org/new-pipeline/dist.html

Lean Data Practices

The Three Principles

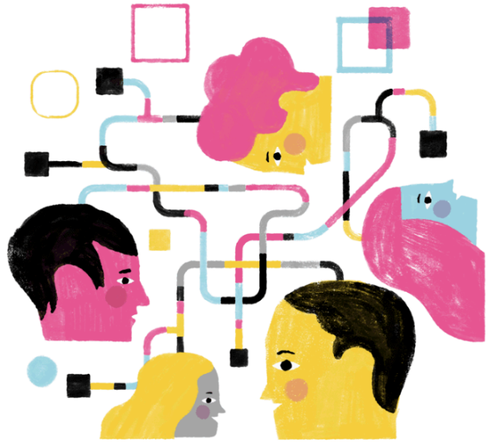
Stay Lean

Decide if all your data collection delivers value.



Build Security

Learn how to protect customer data.



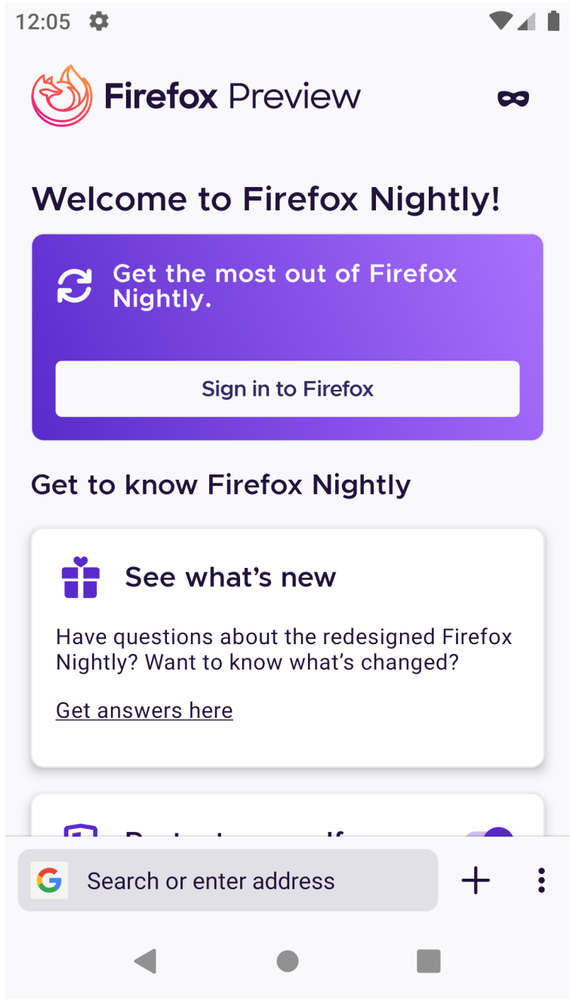
Engage Your Users

Keep customers informed and empowered.

Collecting Data Responsibly and at Scale [2]

[2] StarCon 2019 Talk by chutten.

Firefox for Android



The Glean SDK is a modern approach for a telemetry library and is part of the Glean project.

Introducing Glean — Telemetry for humans
by Georg Fritzsche.

What do we know?

Who's gonna use it?

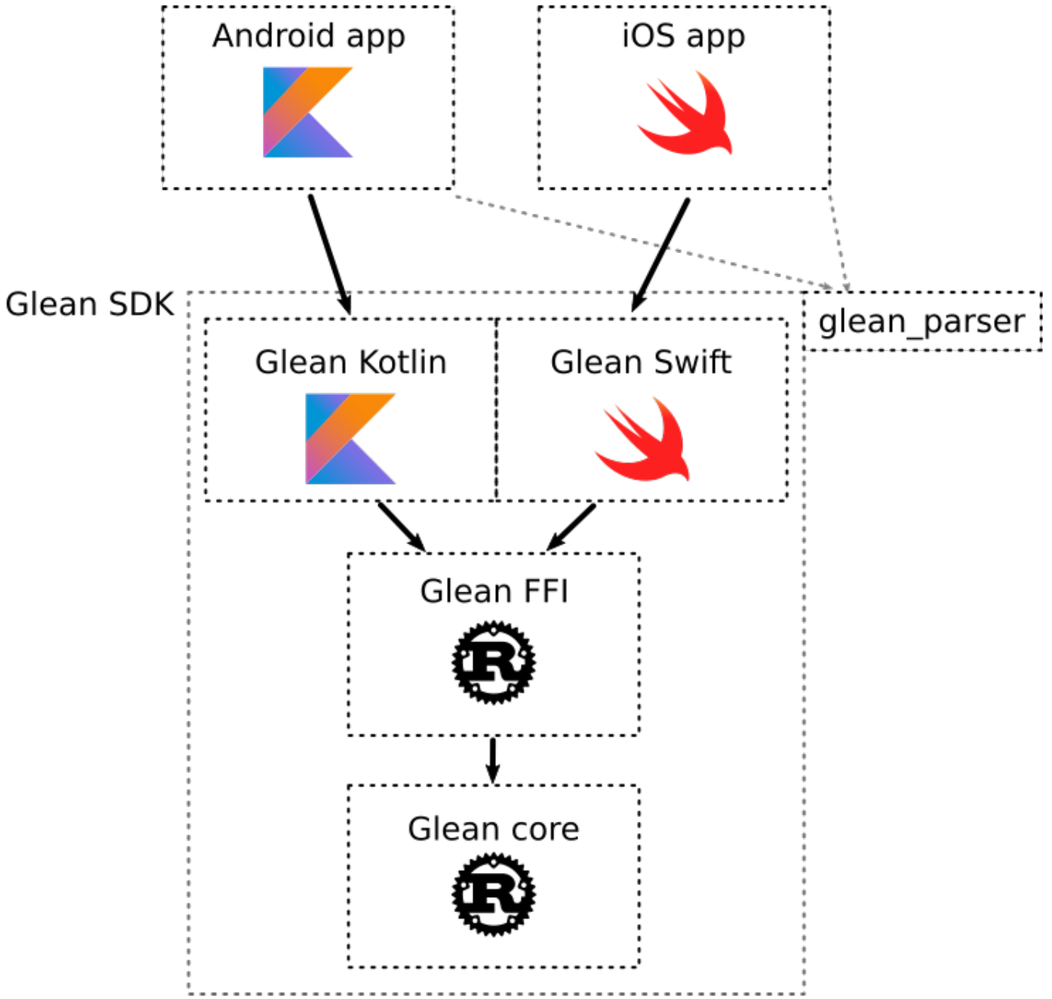
- Firefox for Android now
- Firefox for iOS after that
- Desktop eventually

What's used there?

- Kotlin
- Swift
- C++, JavaScript and Rust

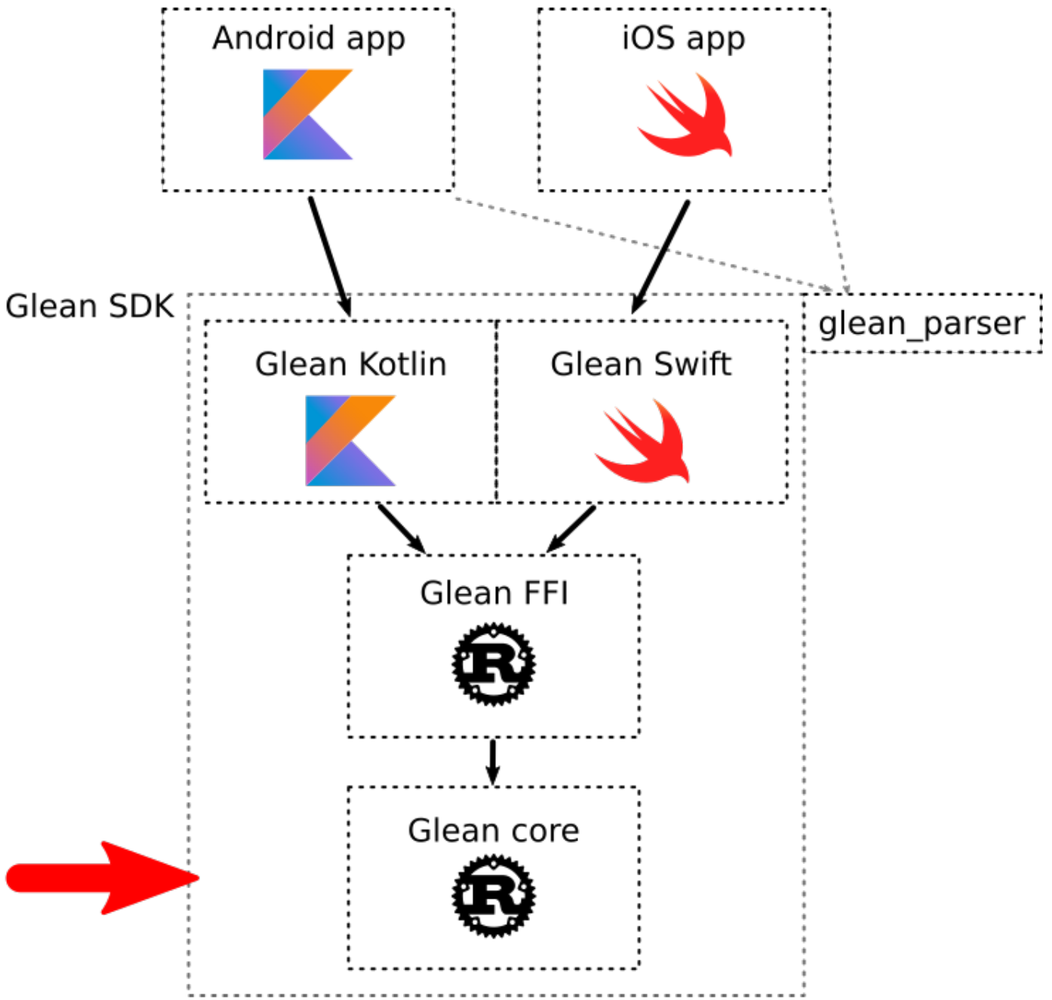
Cross-platform is a must!

Glean SDK stack



Glean Core

A Rust crate



Glean Core - a Rust crate

```
#[derive(Debug)]
struct Glean {
    data_path: PathBuf,
    upload_enabled: bool,
    data_store: Database,
    event_data_store: EventDatabase,
    core_metrics: CoreMetrics,
    // ...
}
```

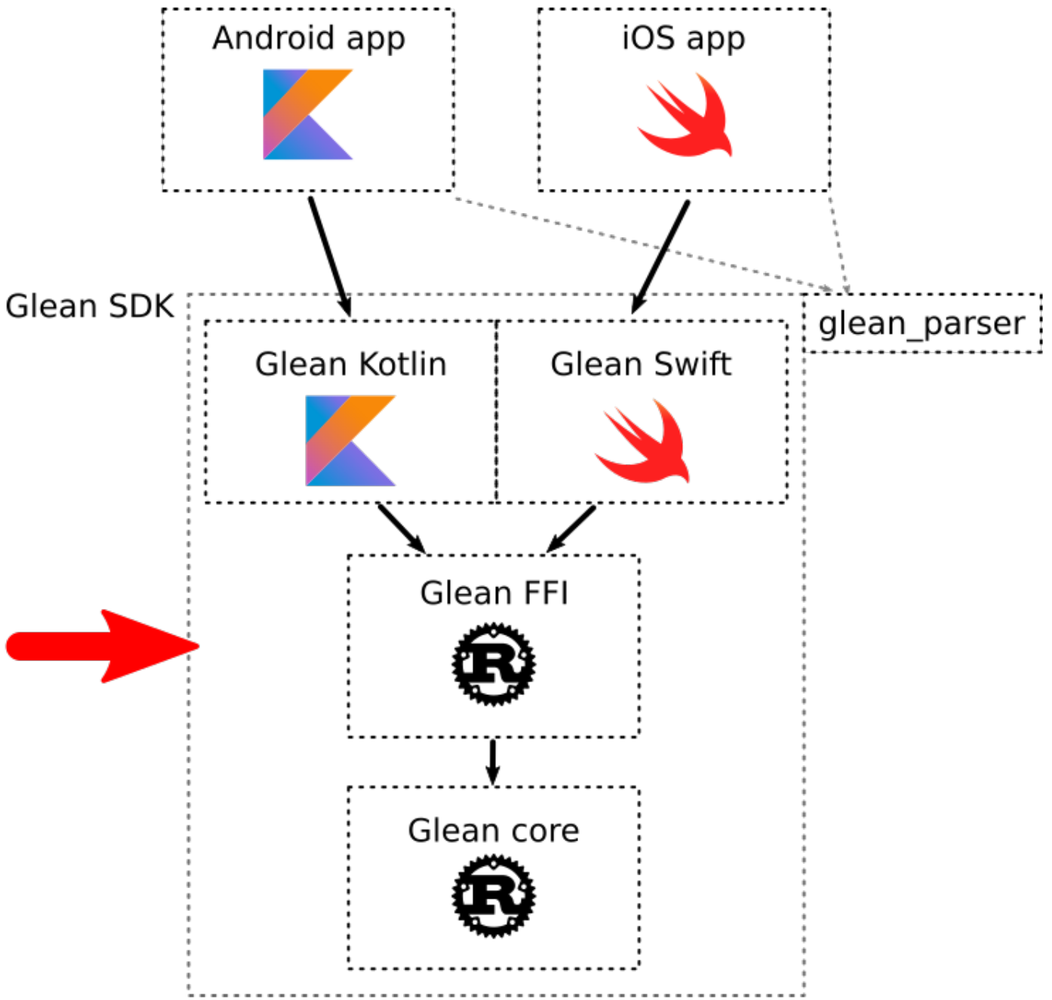
Metric types implemented as Rust types

```
struct CounterMetric {
    meta: CommonMetricData,
}

impl CounterMetric {
    fn add(&self, glean: &Glean, amount: i32) {
        glean
            .storage()
            .record_with(&self.meta, |old_value| old_value.add(amount))
    }
}
```

Glean FFI

the connection



Foreign **F**unction **I**nterface

Getting called from C

```
#[no_mangle]
extern "C" fn hello_rust_linz(data: c_int) -> *const u8 {
    "Rust Linz!\0".as_ptr()
}
```

CString to the rescue

```
let s = CString::new("Rust Linz!").unwrap();
assert!(s.into_bytes_with_nul() == b"Rust Linz!\0");
```

cbindgen

cbindgen creates C headers for Rust libraries which expose a public C API.

```rust
#[no_mangle]
extern "C" fn hello_rust_linz(data: c_int)
  -> *const u8 {
    "Rust Linz!\0".as_ptr()
}
```

```c
#include <stdint.h>
#include <stdlib.h>

const uint8_t *hello_rust_linz(int data);
```

ffi-support

Support library to simplify implementing FFI libraries*.

** as done by application-services[3] & Glean*

[3] https://github.com/mozilla/application-services

ffi-support: `IntoFFI`

Convert Rust types into FFI-compatible types

```
unsafe trait IntoFfi: Sized {
    type Value;
    fn ffi_default() -> Self::Value;
    fn into_ffi_value(self) -> Self::Value;
}

unsafe impl IntoFfi for String {
    type Value = *mut c_char;

    // ...
}
```

ffi-support: `FfiStr`

A safe wrapper around a null-terminated string.

```
struct FfiStr<'a> { /* fields omitted */ }

#[no_mangle]
extern "C" fn hello_rust_linz(data: FfiStr) {
    // Use of `data` after this function returns is impossible
}
```

Compile Targets

```
$ rustup target list
aarch64-apple-darwin (installed)
aarch64-apple-ios (installed)
aarch64-fuchsia
aarch64-linux-android (installed)
aarch64-pc-windows-msvc
aarch64-unknown-linux-gnu
aarch64-unknown-linux-musl
aarch64-unknown-none
[...]
x86_64-apple-ios (installed)
[...]
x86_64-unknown-redox
```

<arch><sub>-<vendor>-<sys>-<abi>

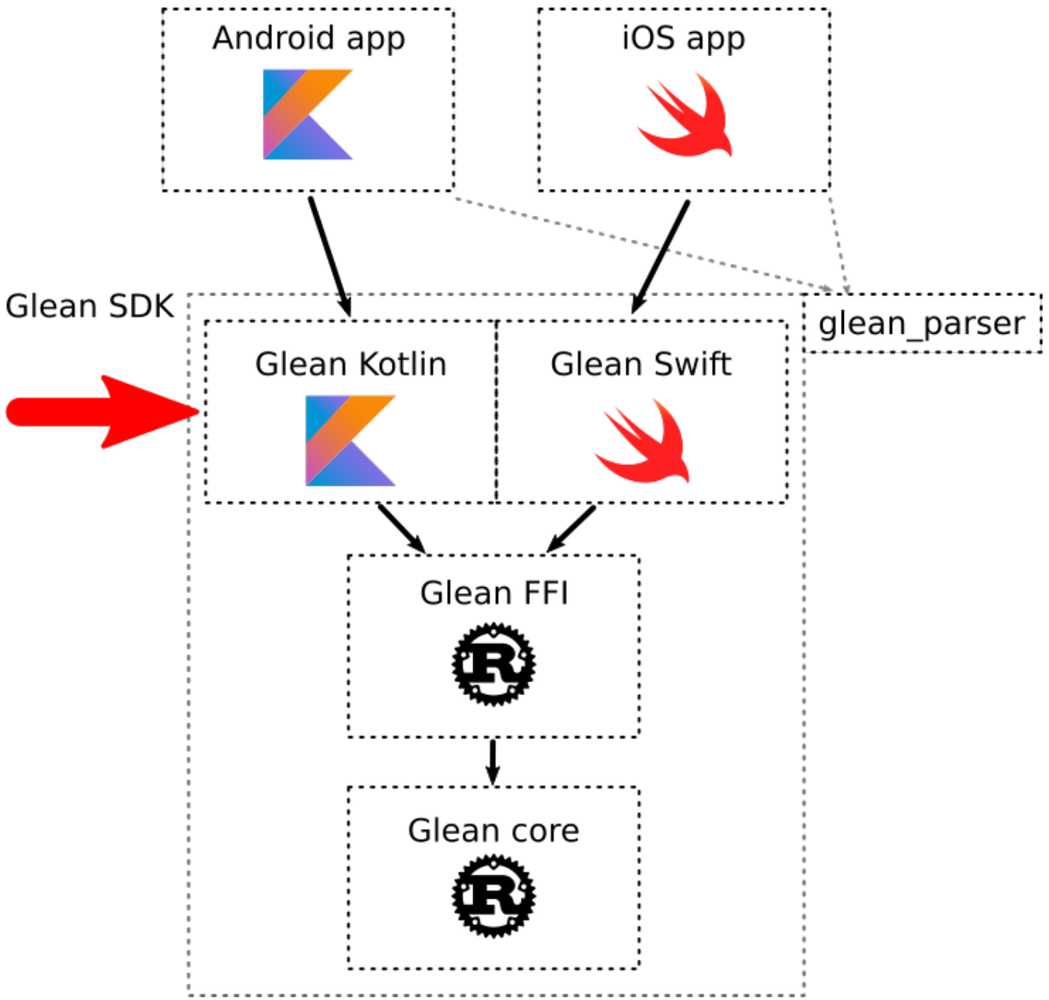
Glean targets[4]

```
rustup target add armv7-linux-androideabi   # for arm
rustup target add i686-linux-android        # for x86
rustup target add aarch64-linux-android     # for arm64
rustup target add x86_64-linux-android      # for x86_64 (& simulator)
rustup target add x86_64-unknown-linux-gnu  # for linux-x86-64
rustup target add x86_64-apple-darwin       # for macOS
rustup target add x86_64-pc-windows-gnu     # for win32-x86-64-gnu
rustup target add x86_64-pc-windows-msvc    # for win32-x86-64-msvc
rustup target add aarch64-apple-ios         # iOS (actual devices)
rustup target add x86_64-apple-ios          # iOS simulator
                  aarch64-apple-ios-sim     # iOS simulator (on an M1)
```

[4] This Week in Glean: rustc, iOS and an M1

Glean Kotlin

The Kotlin implementation



JNA provides Java programs easy access to native shared libraries without writing anything but Java code - no JNI or native code is required.

— *from github.com/java-native-access/jna*

Hello World with JNA

```
#[no_mangle]
extern "C" fn hello() -> *const c_char {
    "Rust Linz!\0".as_ptr()
}
```

JNA - loading a dynamic library

```
internal interface LibGleanFFI : Library {
    companion object {
        internal var INSTANCE: LibGleanFFI =
            Native.load("glean_ffi", LibGleanFFI::class.java)
    }

    // Need to read UTF-8 string from pointer
    fun hello(): Pointer
}

// ...

LibGleanFFI.INSTANCE.hello()
```

Building a Cargo project using Gradle[5]

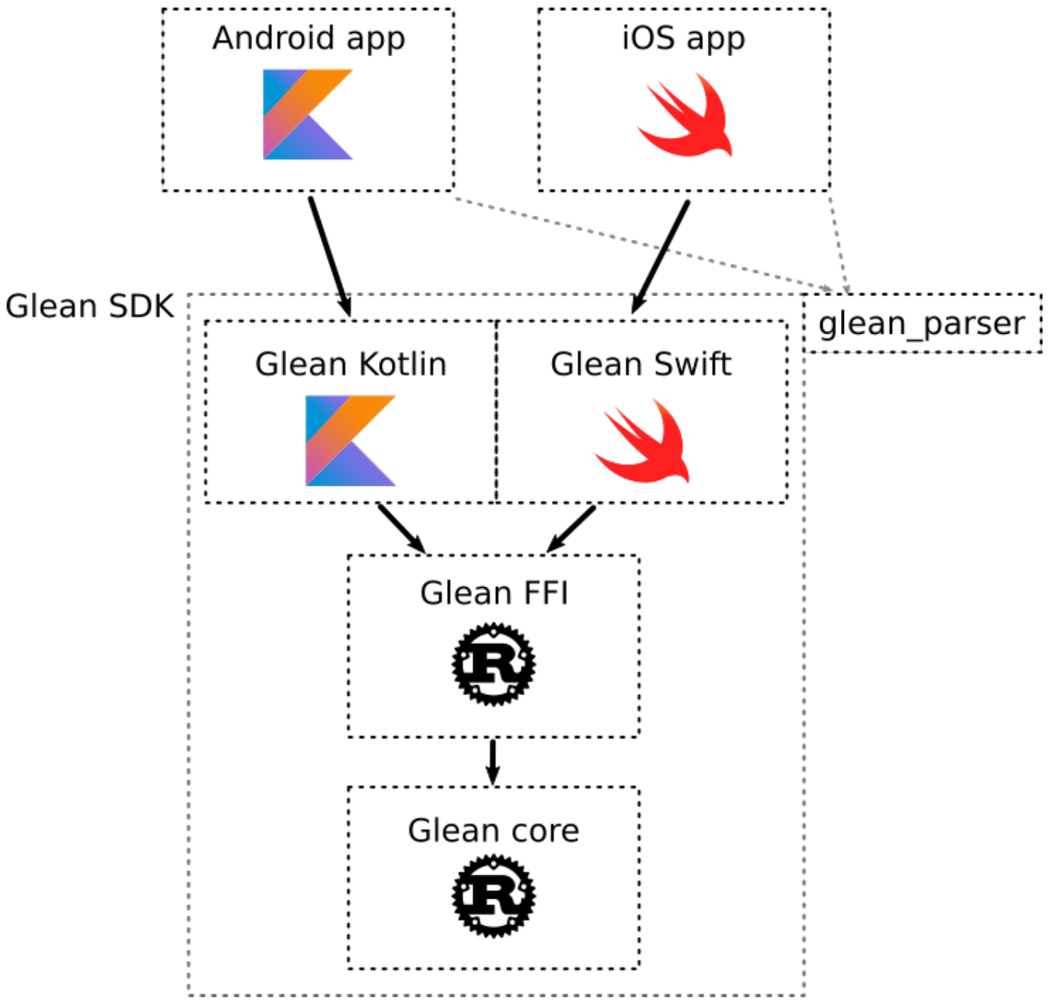
```
apply plugin: 'org.mozilla.rust-android-gradle.rust-android'

cargo {
    module  = "../ffi"
    libname = "glean_ffi"
    targets = ["arm", "x86"]
}
```

[5] github.com/mozilla/rust-android-gradle

Other Glean implementations

- Swift - it speaks C!
- Python - similar to Swift, using cffi
- C++ - for Firefox Desktop
- JavaScript - written in C++
- Rust



What if we don't need to write that much code?

uniffi[7]

a multi-language bindings generator

[7] github.com/mozilla/uniffi-rs

Define your interface

```
// WebIDL-like interface definition language
interface Rocket {
    constructor(string name);

    void lock_steering(string direction);

    [Throws=LaunchError]
    boolean launch();
};
```

Implement it in Rust[8]

```
struct Rocket {
  name: String,
  direction: String,
}

impl Rocket {
  fn new(name: String) -> Rocket {
    Rocket {
      name: name, steering: "".into()
    }
  }
```

```
fn lock_steering(&mut self, dir: String) {
  self.direction = dir;
}

fn launch(&self) -> Result<bool> {
  if self.direction != "up" {
    return Err(LaunchError::RocketLaunch);
  }
  Ok(true)
}
```

[8] Full example: github.com/badboy/rocketscience

✨ 🪄 MAGIC! 🪄 ✨

Use it in Swift

```
import rocketscience

let rocket = Rocket(name: "Orbiter")
rocket.lockSteering("up")
try! rocket.launch()
```

Use it in Kotlin

```
import rediger.uniffi.rocketscience.*

val rocket = Rocket(name = "Orbiter")
rocket.lockSteering("up")
rocket.launch()
```

Use it in Python

```
from rocketscience import *

rocket = Rocket(name = "Orbiter")
rocket.lock_steering("up")
rocket.launch()
```

uniffi ✨ 🪄 magic! 🪄 ✨

- Generates the FFI layer for you
- Takes care of type conversions on both sides
- Convenient APIs for the target language

uniffi under the hood (Rust)[9]

```
#[no_mangle]
pub extern "C" fn ffi_rocketscience_3a9e_Rocket_object_free(handle: u64) {
    let _ = UNIFFI_HANDLE_MAP_ROCKET.delete_u64(handle);
}

#[no_mangle]
pub extern "C" fn rocketscience_3a9e_Rocket_new(
    name: uniffi::RustBuffer,
) -> u64 {
    UNIFFI_HANDLE_MAP_ROCKET.insert_with_output(|| {
        Rocket::new(<String as uniffi::ViaFfi>::try_lift(name).unwrap())
    })
}
```

[9] rocketscience.uniffi.rs

uniffi under the hood (Swift)[10]

```
public class Rocket: RocketProtocol {
    public convenience init(name: String) {
        self.handle = rocketscience_3a9e_Rocket_new(name.lower())
    }
    deinit { ffi_rocketscience_3a9e_Rocket_object_free(handle) }

    public func add(part: Part) {
        try! rustCall(
            UniffiInternalError.unknown("rustCall")
        ) { err in
            rocketscience_3a9e_Rocket_add(self.handle, part.lower(), err)
        }
    }
}
```

[10] rocketscience.swift

We wrote code like that before!

We wrote code like that before!

 Computers are better at it.

uniffi for Glean?

This summer? Hopefully[11].

[11] Don't let my manager see this or he's gonna take my word for it.

Thanks to

the Glean team: Alessio, Bea, Chris, Travis and Mike.

the application-services & uniffi team.

Links

- Slides: fnordig.de/talks/2021/rustlinz/slides.pdf
- Rocket Science example: github.com/badboy/rocketscience
- Glean SDK repository: github.com/mozilla/glean
- Glean SDK docs: mozilla.github.io/glean
- Mozilla Data blog: blog.mozilla.org/data
- me on Twitter: @badboy_

Questions?