

Flutter における FFI

FFI ?

Foreign **f**unction **i**nterface

今回は C++/C の呼び出しの話

話すこと

- **利用者目線の Flutter/Dart における FFI**
- **提供者目線の Flutter/Dart における FFI**

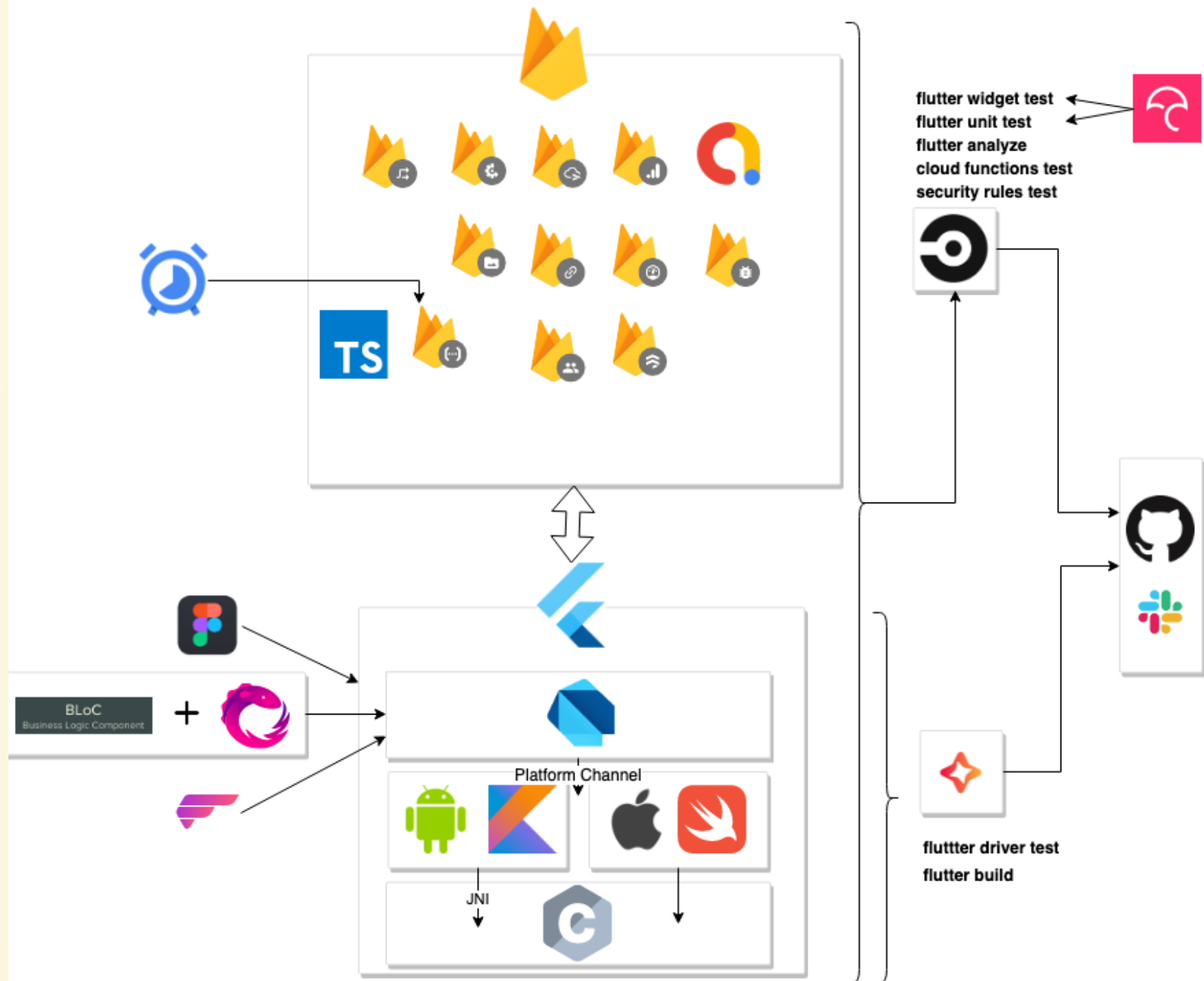
自己紹介

しみず なおき





お家で作ってるモノ



オセロには常に C が必要

各言語の C 呼び出し

代表的なもの

言語	実装方法
C++	<code>extern "C"</code> で C++ の名前マングリングを無効にできる。
Java	JNI や JNA , SWIG を使う
Python	ctypes や cffi を使う
Rust	extern キーワード で容易に呼べる
Ruby	Ruby-FFI を使う
Javascript	WebAssembly を使う
Swift	そのままいける し、 カスタム も可能

```
package main

/*
#include <stdlib.h>
#include <stdio.h>

void hello() {
    printf("Hello\n");
}
*/
import "C"

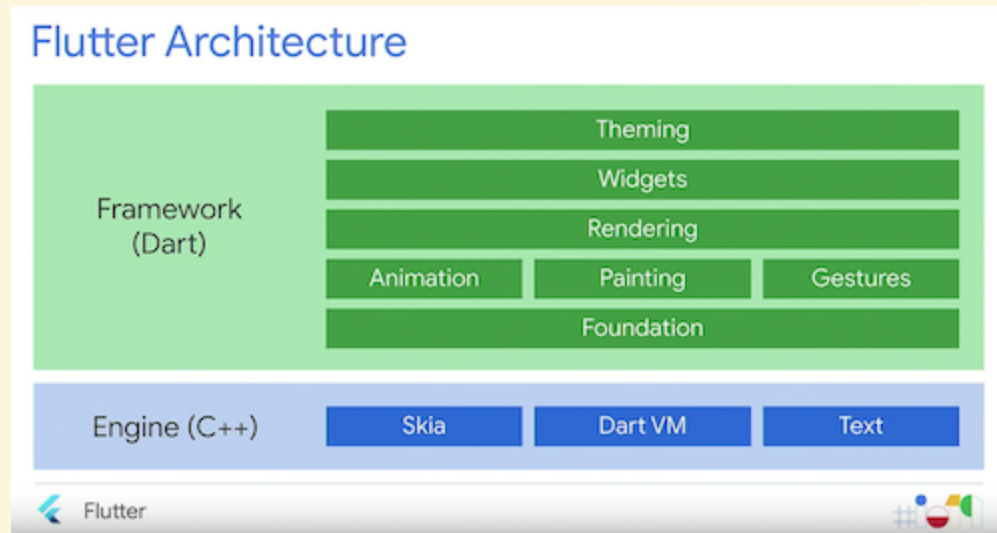
func main() {
    C.hello()
}
```

Dart は？

これから本題

利用者目線の Flutter/Dart における FFI

Google I/O'19 でも言及あり



“

We are working on a new foreign function interface. This should help you reuse existing C and C++ code, which is important for some critical stuff

”

① Native Extension

② dart : ffi

① Native Extension

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include "include/dart_api.h"
#include "include/dart_native_api.h"

Dart_NativeFunction ResolveName(Dart_Handle name, int argc, bool* auto_setup_scope);

DART_EXPORT Dart_Handle sample_hello_Init(Dart_Handle parent_library) {
    if (Dart_IsError(parent_library)) return parent_library;
    Dart_Handle result_code = Dart_SetNativeResolver(parent_library, ResolveName, NULL);
    if (Dart_IsError(result_code)) return result_code;
    return Dart_Null();
}

void hello(Dart_NativeArguments arguments) {
    Dart_EnterScope();
    printf("Hello\n");
    Dart_ExitScope();
}

Dart_NativeFunction ResolveName(Dart_Handle name, int argc, bool* auto_setup_scope) {
    if (!Dart_IsString(name) || auto_setup_scope == NULL) return NULL;
    Dart_EnterScope();
    const char *cname;
    Dart_StringToCString(name, &cname);
    Dart_NativeFunction result = NULL;
    if (strcmp(cname, "Hello") == 0) result = Hello;
    Dart_ExitScope();
    return result;
}
```

Dart 側

```
library sample_hello;  
import 'dart-ext:sample_hello';  
void hello() native "Hello";
```

参考: [dart-lang sample extension](#)

② dart:ffi

“ The extension mechanism discussed in this page is for deep integration of the VM.
If you just need to call existing code written in C or C++, see [C & C++ interop using FFI](#). ”

引用元: [Native extensions for the standalone Dart VM](#)

```
import "dart:ffi" as ffi;
import 'dart:io' show Platform;

void main() {
  final libHelloWorld = ffi.DynamicLibrary.open(
    "./libHelloWorld.dylib");
  final helloWorld = libHelloWorld.lookupFunction
    <ffi.Void Function(), void Function()>("helloWorld");

  helloWorld();
}
```

https://github.com/sensuikan1973/Dart_FFI_Hello_World

さて、Flutter では？

Support integrating with C/C++ in plugin framework #7053



jtrunick opened this issue on 29 Nov 2016 · 141 comments



jtrunick commented on 29 Nov 2016 • edited by mit-mit ▾



It would be nice to have an example of calling C/C++ code, or at least how to build native code along with a Flutter app. This may purely a Gradle question, but its not clear to someone that's not an expert on Gradle (for example, me), how to pull this off.

Admin comment: Please see [dart-lang/sdk#34452](#) for current status and additional information



553



52



68



14



117



21

Ass



Lab

de

eng

p: 1

plu

sev

たくさんの 👍 の思いは？

① 既存ソフトをより統合しやすくしてほしい

○ **大量のグルーコードがっらい**

○ **低オーバーヘッドがいい**

SQLite

Realm

OpenCV

crypto, ssh ... libraries

などが具体例として挙げられている

② 大量のデータを効率よく出し入れしたい

なお、Dart 2.4 から TransferableTypedData が使用できるようになったので、ある程度はそれで間に合いそう

どうするか？

提供者目線の Flutter/Dart における FFI

① C++ でメソッドチャンネルを提供する？



**メソッドチャンネルがオーバーヘッド
高いので、目的に合わない**

② Native Exstention でサポートできるようにする？



【理由 1】

名前ベースの API

```
DART_EXPORT DART_WARN_UNUSED_RESULT Dart_Handle  
Dart_SetField(Dart_Handle container, Dart_Handle name, Dart_Handle value);
```

👉 AOT に不親切

👉 名前解決がキャッシュされない

dart-lang/sdk/runtime/include/dart_api.h より引用

【理由 2】

Reflective Marshaling は効率良くない

```
void isEmailAddress(Dart_NativeArguments arguments)
```

`void` `arguments` 👁️ 返り値も引数も型は決まってるけど...

⇒ 引数/返り値が静的に型付けされた上での Marshaling の方が効率良い

⇒ FFI 🙌

Flutter/Dart チームが採った方法は？

dart : ffi 👍

ちなみに


“ we expect that moving Flutter Engine from C API to FFI should significantly reduce overheads associated with crossing the boundary between Dart and native code ”

結果どう使えるのか？

利用者目線の Flutter/Dart における FFI

に話を戻す


利用者目線の Flutter/Dart における FFI

 dart-lang / sdk

Unwatch releases 231 Unstar 3,935 Fork 496

<> Code Issues 4,719 Pull requests 2 Projects 6 Wiki Security Insights

Tag: 2.4.0 sdk / samples / ffi / sqlite / Create new file Upload files Find file History

 dcharkes and commit-bot@chromium.org [doc/ffi] Fix CString implementation in dart:ffi sample ... Latest commit e7f7984 on 21 May

..

docs	[doc/ffi] Fix CString implementation in dart:ffi sample	2 months ago
example	[vm/ffi] Add copyright headers to sqlite example and incorporate Mich...	3 months ago
lib	[doc/ffi] Fix CString implementation in dart:ffi sample	2 months ago
test	[vm/ffi] Add copyright headers to sqlite example and incorporate Mich...	3 months ago
.gitignore	[doc] dart:ffi SQLite sample	4 months ago
README.md	[vm/ffi] Add copyright headers to sqlite example and incorporate Mich...	3 months ago
pubspec.yaml	[doc] dart:ffi SQLite sample	4 months ago

README.md

Sample code dart:ffi

This is an illustrative sample for how to use `dart:ffi`.

2.4 にて Preview 版提供開始！

(Flutter/Android での試験的サポートも始まっている)

どんな感じの構成になるのか

App Developer	Package Developer			Dart VM Team	Package Developer	Native Library Developer
Flutter App (Imports package)	Package API (Does not expose dart:ffi)	Package Implementation (Code which converts C++ abstractions into Dart abstractions)	Bindings	dart:ffi	Glue code (Code which takes care of things such as C++ exceptions)	Native Library
Dart					C / C++	

今後も Flutter/Dart に期待大

意欲的な方は、
ぜひ dart:ffi のレビュー版 FB を送り
ましょう 🍷

おわり？



Flutter/Dart の FFI 実装の難しさに触れてみたい

提供者目線の Flutter/Dart における FFI

の話を時間の限りします

あああ

ありがとうございました

リンク一覧

- [Dart VM FFI Vision](#)
 - [Design and implement Dart VM FFI](#)
 - [Flutter Support integrating with C/C++ in plugin framework](#)
 - [Native extensions for the standalone Dart VM](#)
 - [Support for Dart Extensions](#)
- [C & C++ interop using FFI](#)
 - [Dart Native platform](#)
 - [dart:ffi sqlite sample](#)
- [The Engine architecture](#)
 - [Writing custom platform-specific code](#)
 - [Custom Flutter Engine Embedders](#)
- [Language features for FFI](#)
- [sensuikan1973/flutter-ffi-slide](#)
- [sensuikan1973/Dart FFI Hello World](#)