FLUTTER

1. 跨端技术介绍
2. flutter架构设计
3. flutter通信原理

-- 刘爱霞

01

跨端技术介绍

1 跨端

PhoneGap

小程序

React Native

Flutter

Weex

PWA

快应用

...

基于webView，js + html 在一个线程

基于webView，JS + HTML 两个线程

渲染交给原生系统，类HTML+JS的UI创建逻辑

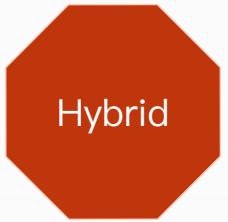
跳过原生系统渲染，直接和更底层的Skia图形库对接渲染 + Dart开发逻辑

1
跨端

如何才能实现跨端?

1. UI如何绘制

2. 逻辑（包括用户交互的逻辑和与宿主系统通讯的逻辑）如何响应



Ionic/Cordova/PhoneGap

Rn/Weex

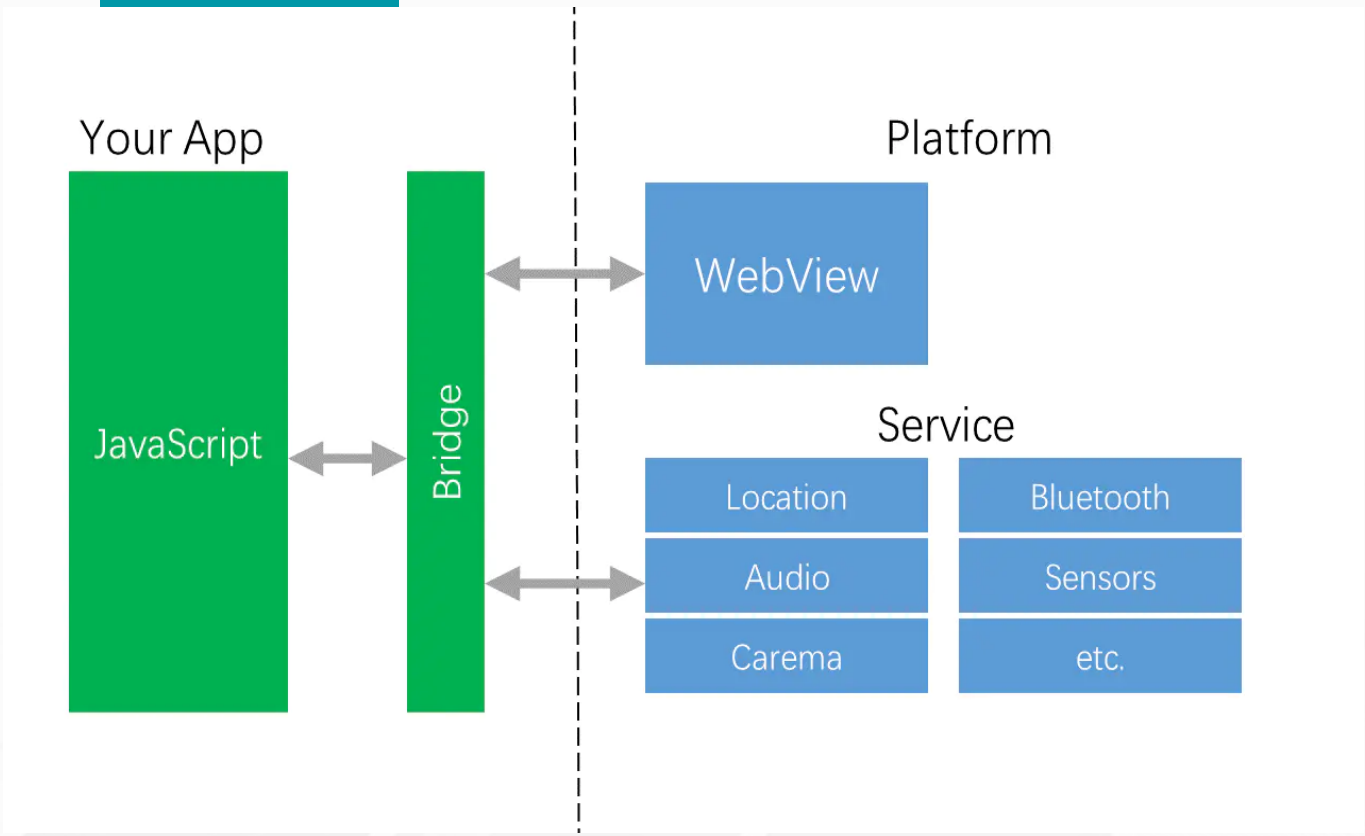
Flutter

Hybrid给出的答案是webView + JS,
OEM Wrapper给出的是 VirtualDOM 转 Native 组件 + JS,
自渲染（Flutter）给出的答案是Skia+Dart

1

跨端

Hybrid



原理：

ui：运行在系统自带的 WebView中，WebView也可以理解为一个OEM组件

逻辑：js + 将原生的接口封装后暴露给 JavaScript，

1. 开发效率

对前端开发者友好，背靠前端庞大的JavaScript生态

涉及到Native调用的部分不可避免要熟悉Android/iOS

能力受限于桥接层，扩展性弱

在移动端开发，调试和错误日志并不是很友好

2. 动态化

Web天生自带动态能力

3. 多端一致性

浏览器内核的渲染独立于系统组件，无法保证原生体验

涉及宿主的问题，需要开发者处理，做不到完全屏蔽

4. 性能

受限于网络环境，比Native更加消耗流量

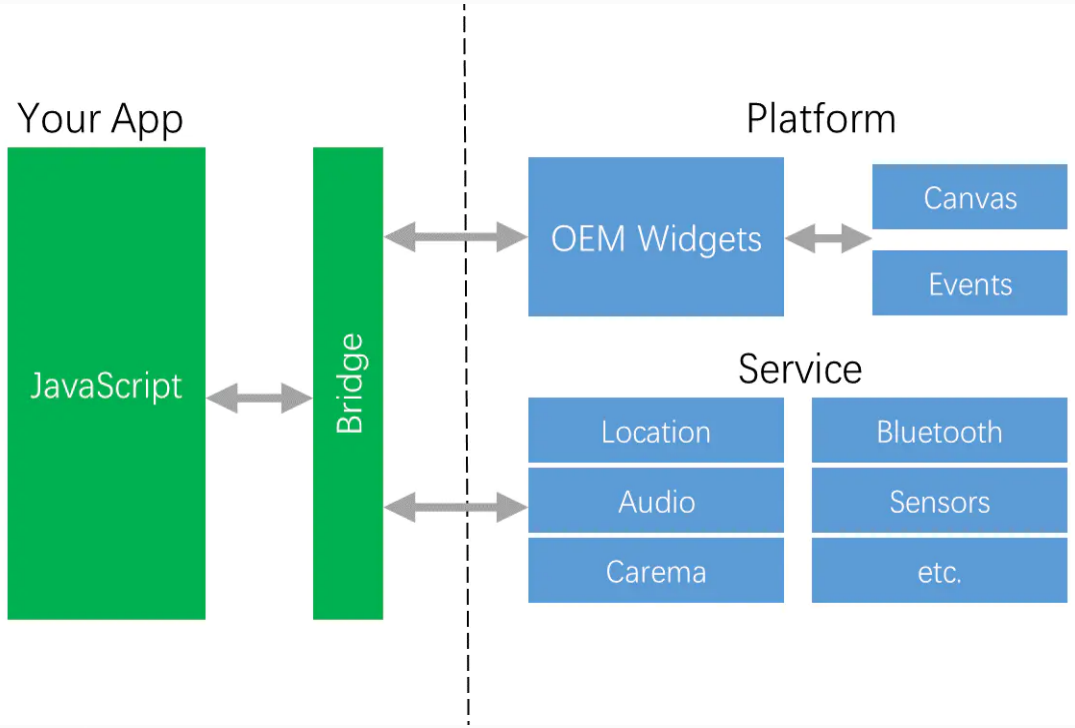
受限于浏览器、系统平台特性

渲染性能 ， Webview性能差

1

跨端

OEM Wrapper



原理：

ui：使用类前端的语法来描述OEM，从而转成Native控件，交由系统绘制。

逻辑：ios/android Bridge + js

版本变动频繁，需要开发者自己优化，工作量大

与Native交互需要开发者自己支持，维护成本高

文档不完善、调试信息、错误日志提示不够友好

2. 动态化

可以支持

3. 多端一致性

渲染成各自平台的组件，可以保证Native的体验

由于渲染依赖原生控件，不同平台的控件需要单独维护，并且当系统更新时，社区控件可能会滞后

其控件系统也会受到原生UI系统限制，例如，在Android中，手势冲突消歧规则是固定的，这在使用不同人写的控件嵌套时，手势冲突问题将会变得非常棘手

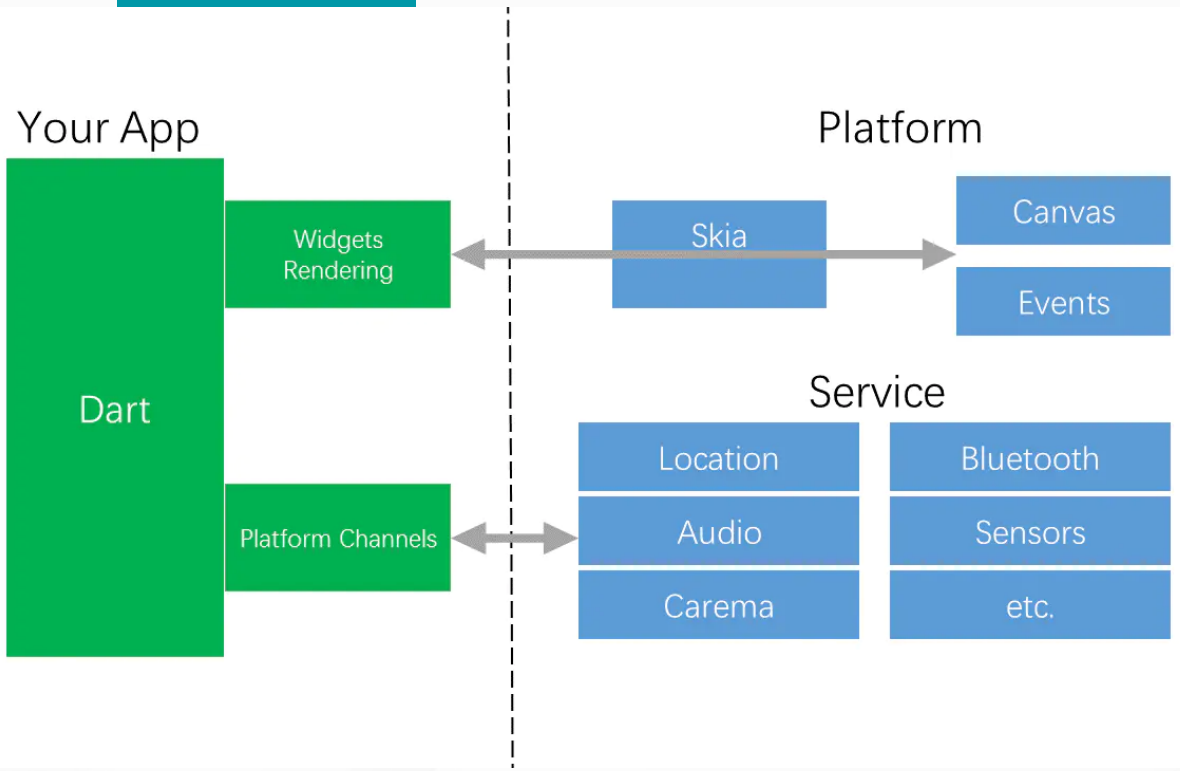
4. 性能

稍差于Native，但远好于Hybrid

渲染时需要JavaScript和原生之间通信，在有些场景如拖动可能会因为通信频繁导致卡顿

1 跨端

自渲染



原理：

ui：基于 skia 图像引擎完成的，不依赖任何一个系统平台

逻辑：Platform channel 通信模式 + Dart

1. 开发效率

开发工具完备，提供了VS Code（最流行的编辑器），Intellij IDEA 插件

Google背书，文档完备，社区较完备

Dart语言本身有上手成本，没有前端的生态

2. 动态化

动态性不足，为了保证UI绘制性能，自绘UI系统一般都会采用AOT模式编译其发布包

3. 多端一致性

自绘制UI，提供了Material Design和Cupertino两种风格的Widget

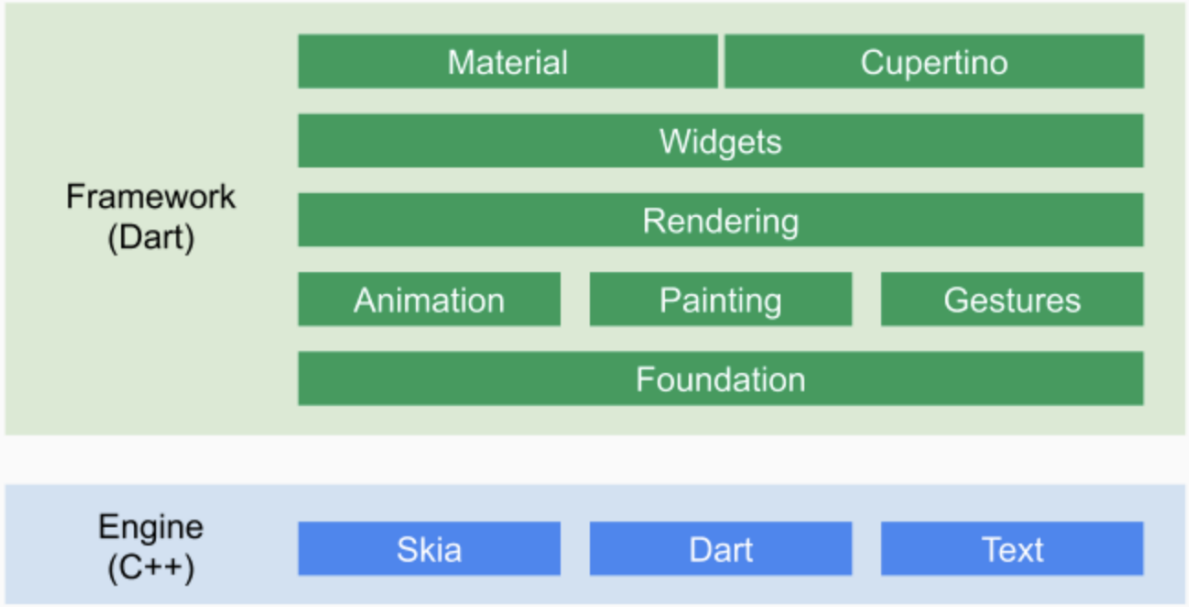
4. 性能

性能和Native绘制一样

02

Flutter 架构

2
flutter 架构



1. 有封装好的 UI 组件 Material 和 Cupertino 【相 当于前端的 Ant-Design / Element / iview 等 UI 框架】

2. 封装好的 UI 组件 Material 和 Cupertino 由更基础的 Widget 组件拼装而成。 【这里的 Widget 相当于 前端的 HTML div, h1, span 等标签元素】

3. 继续往下， Widget 层是由 Animation(动画), Painting(绘制), Gesture(手势) 共同通过 Rendering 构成的 对象。【这里和前端稍稍有点区别，前端 UI 的结构(html)，样式(CSS)，事件交互(JS 是分开的，而在 Flutter ， 都是 Widget】

- - 元素 widget。 如 button，menu，list
- - 样式 widget。如 font，color
- - 布局 widget。 如 padding，margin

Framework 中的 UI 交互都是有 Engine 来进行绘制渲染的。Engine 层内部会通过 Skia 图形引擎画出 UI 组件，Skia 是 Google 开源的 2D 图形引擎，适用于多个平台系统，这也是 flutter 能跨平台的核心元素之一。这也是为什么前面说 flutter 能不局限系统 OEM 组件的限制。 也就是说，如果你想要自己封装一个 ant-design 画风的 flutter UI 框架，你可以直接通过基础的 Widget 搭建出自己的 UI 框架。如果底层基础 UI 满足不了你的需求。你可以直接用 dart 调用 Skia 图像引擎的 API，画出自己的 UI，没有任何的限制。

03

Flutter 通信

3

flutter 通信

Platform Channel简介

不同于之前几种用 bridge 的方式来调用系统服务，flutter 用 Platform channel机制的形式去调用系统api。Platform Channel 不依赖代码生成，而是建立在消息传递方式上。

在Flutter中,提供了三种Platform Channel用来支持和平台之间数据的传递:

- **BasicMessageChannel**: 支持字符串和半结构化的数据传递
- **MethodChannel**: 支持方法调用,既可以从Flutter发平台发起方法调用,也可以从平台代码向Flutter发起调用
- **EventChannel**: 支持数据流通信

MethodChannel Demo -- 获取电池电量 -- Flutter端

Flutter端：

```
// 创建一个方法通道，通道名称是samples.flutter.io/battery
static const platform = const MethodChannel('samples.flutter.io/battery');
// 获取电池电量方法

Future<void>_getBatteryLevel()async {
      String batteryLevel;
      try {
          // 调用该通道的方法getBatteryLevel，在对应的端中也有这个方法
           final int result = awaitplatform.invokeMethod('getBatteryLevel');
           batteryLevel = 'Battery level at $result % .';
      } on PlatformException catch (e) {
           batteryLevel = "Failed to get battery level: '${e.message}'.";
      }
      setState((){_batteryLevel=batteryLevel;});
 }
```

MethodChannel Demo -- 获取电池电量

安卓端（java）：

```
importandroidx.annotation.NonNull;

importio.flutter.embedding.android.FlutterActivity;
importio.flutter.embedding.engine.FlutterEngine;
importio.flutter.plugin.common.MethodChannel;

publicclassMainActivityextendsFlutterActivity{

    privatestaticfinalStringCHANNEL="samples.flutter.dev/battery";
     @Override
     publicvoidconfigureFlutterEngine(@NonNullFlutterEngineflutterEngine){
          super.configureFlutterEngine(flutterEngine);
          newMethodChannel(flutterEngine.getDartExecutor().getBinaryMessenger(),CHANNEL).setMethodCallHandler((call,result)->{
               if(call.method.equals("getBatteryLevel")){
                    intbatteryLevel=getBatteryLevel(); // 调用java方法，返回数据
                    result.success(batteryLevel);
               }

          }
     );
}
```

THANKS

多多指教