

*Flutter*设计、实现与演进

一个开发者的观察、实践与思考

自我介绍



赵 裕

- 腾讯 - 客户端开发工程师
- 著有《Flutter内核源码剖析》，博客: vimerzhao.top
- 腾讯学堂讲师: 《Flutter源码导读》、《软件Bug的排查与解决》

分享大纲

- Part1 开场介绍
- Part2 设计思考
- Part3 实现细节
- Part4 演进梳理
- Part5 QA交流



跨平台的历史 - 技术视角

HTML 1.0

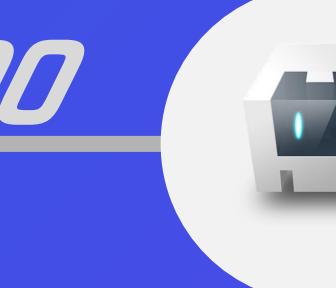
在1993年6月作为互联网工程工作小组(IETF)工作草案发布



1993



2000



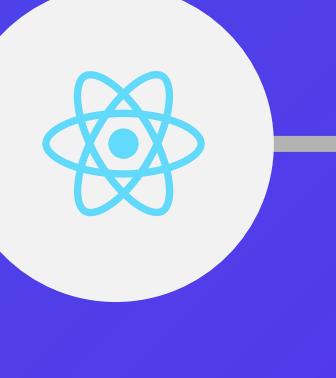
2008



2013



2016



2018



QT 1.0

On 20 May 1995 Troll Tech publicly released Qt 0.90 for X11/Linux with the source code under the Qt Free Edition License

SWT 1.0

AWT->Swing->SWT(Standard Widget Toolkit), 2001

React Native

2015年, Facebook 正式推出

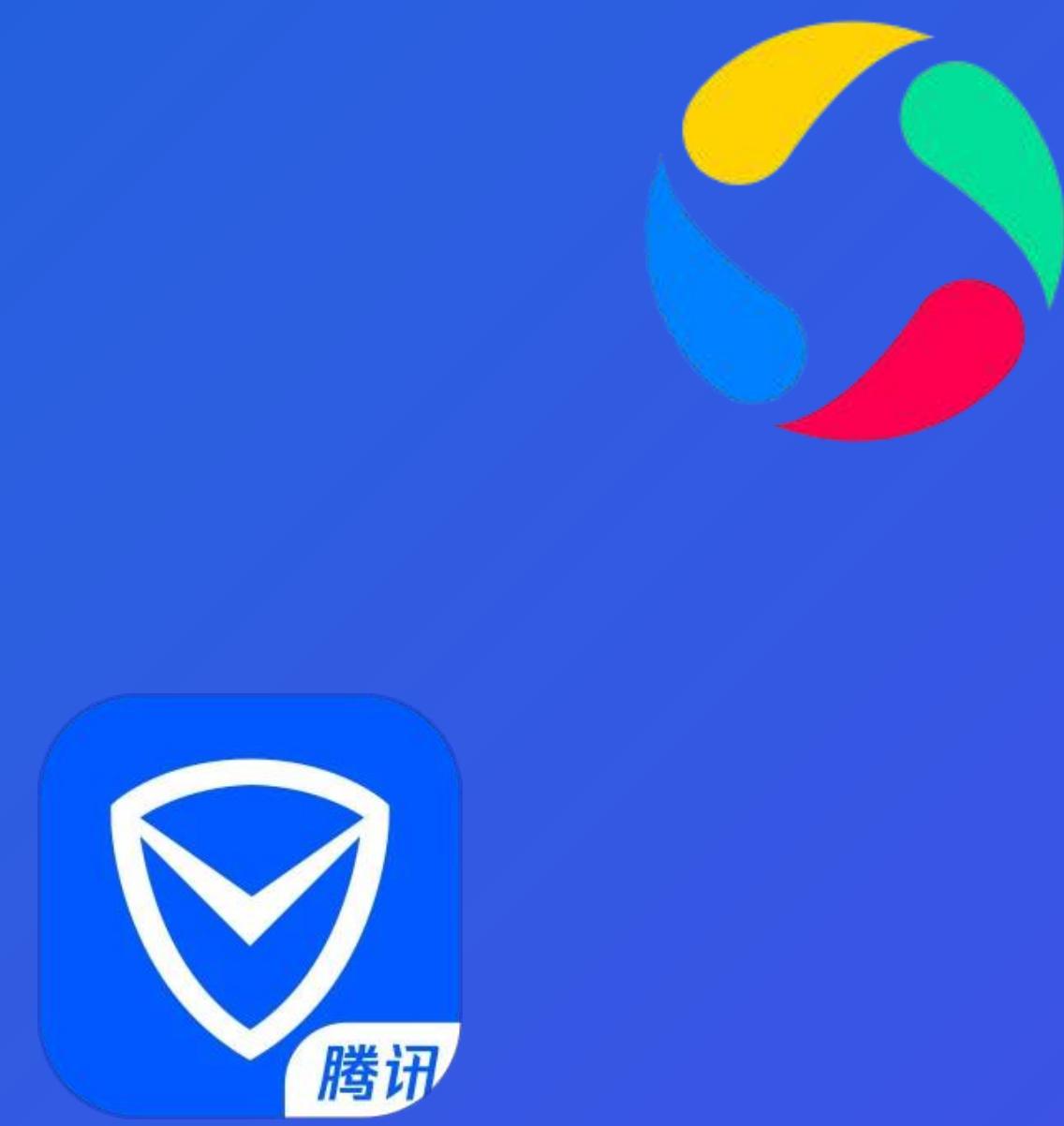
PhoneGap

2009年前后,
最早出现于iOS

Flutter

2018年, Flutter 1.0发布

跨平台的历史 - 产品视角

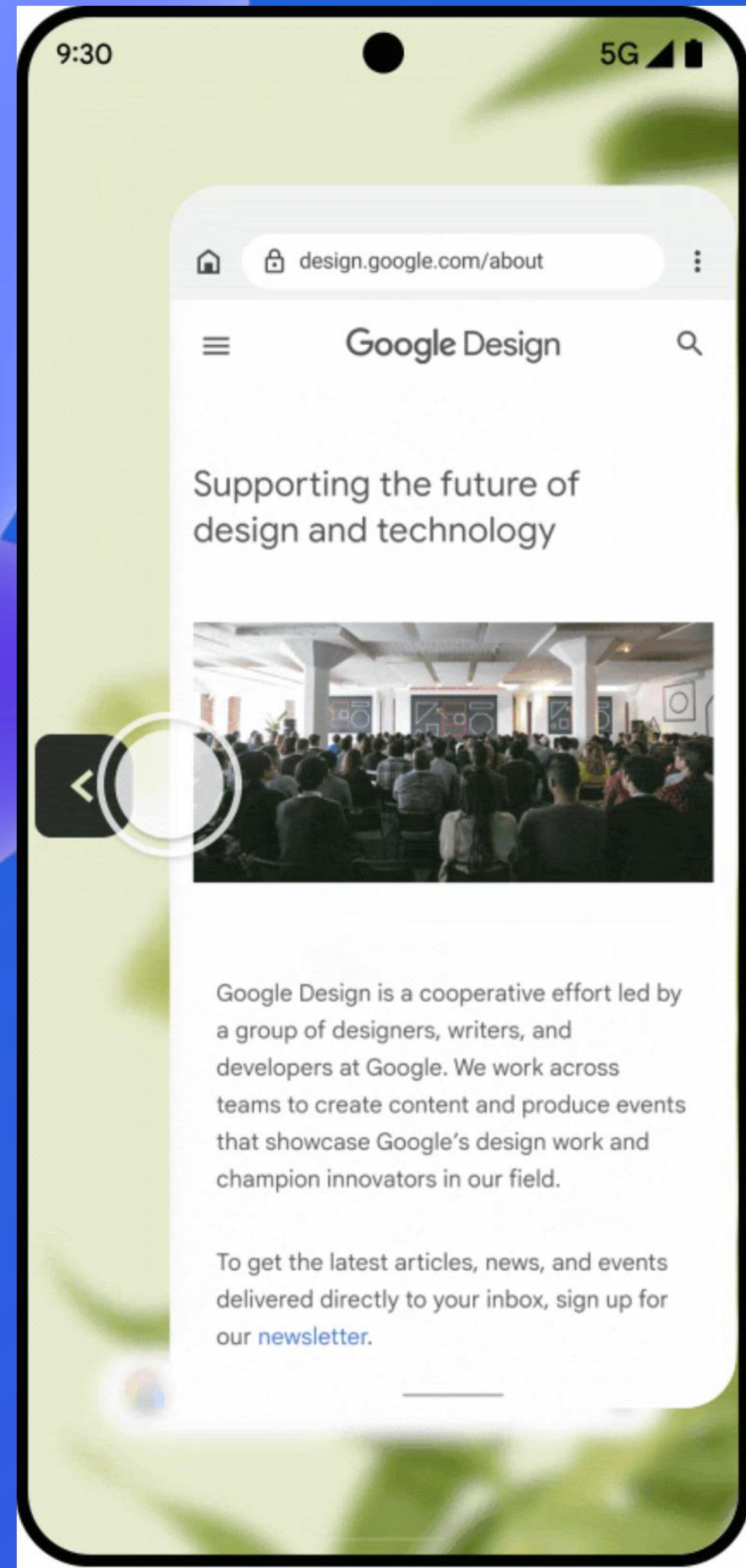


移动时代



移动+PC+Web

跨平台的思考：是否存在彻底的跨平台



Predictive Back Gesture



Android QQ

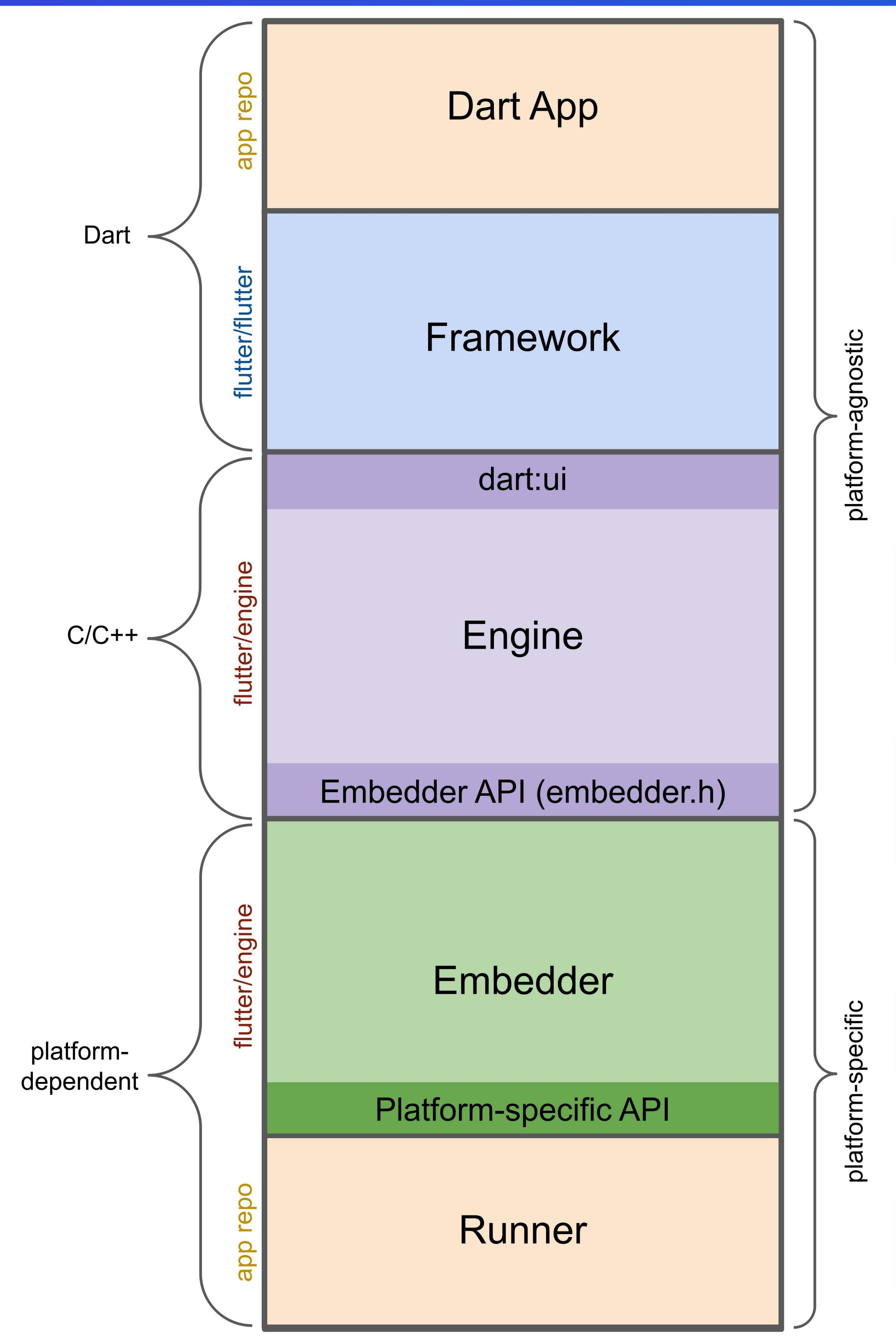
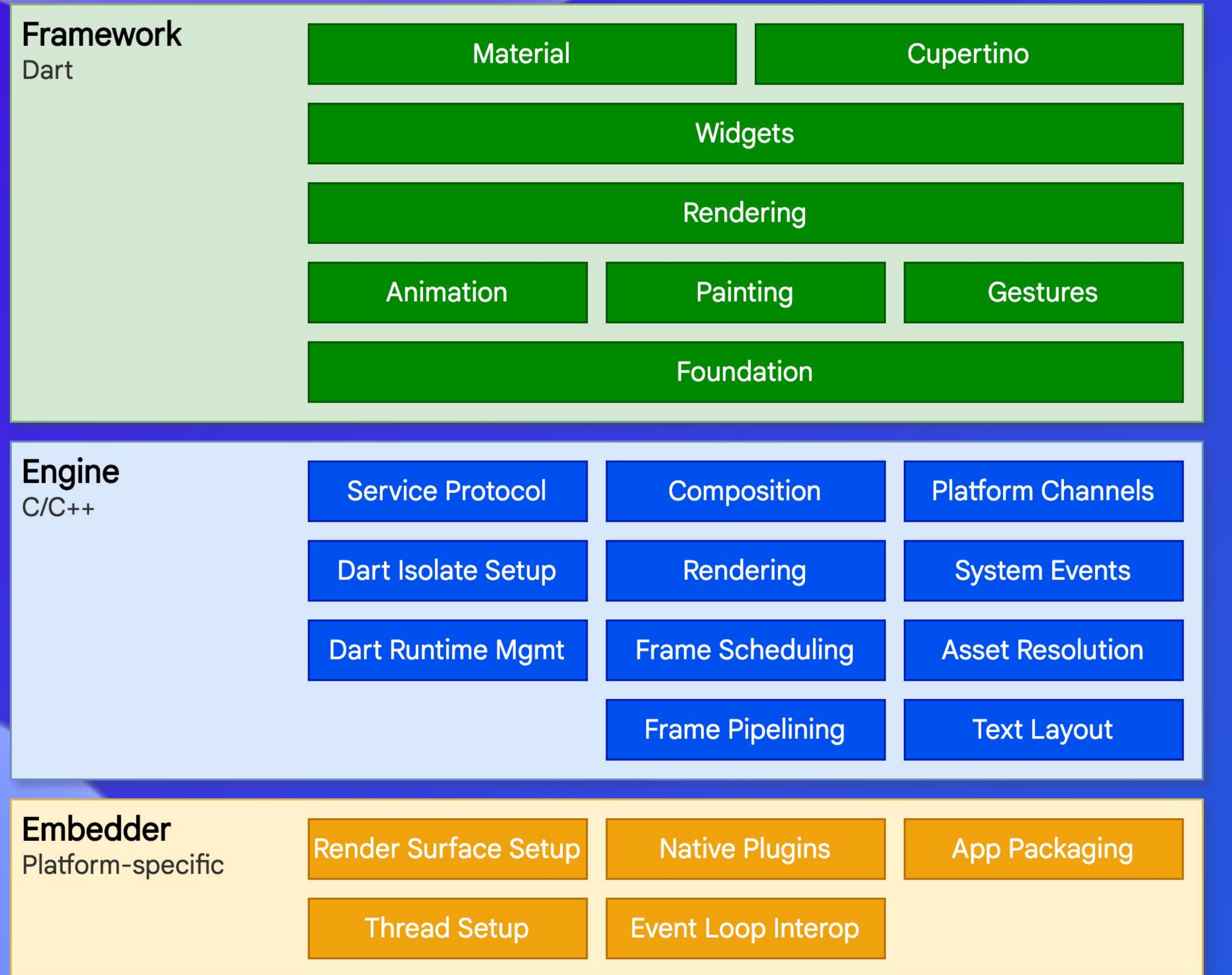


PC QQ

Android & iOS

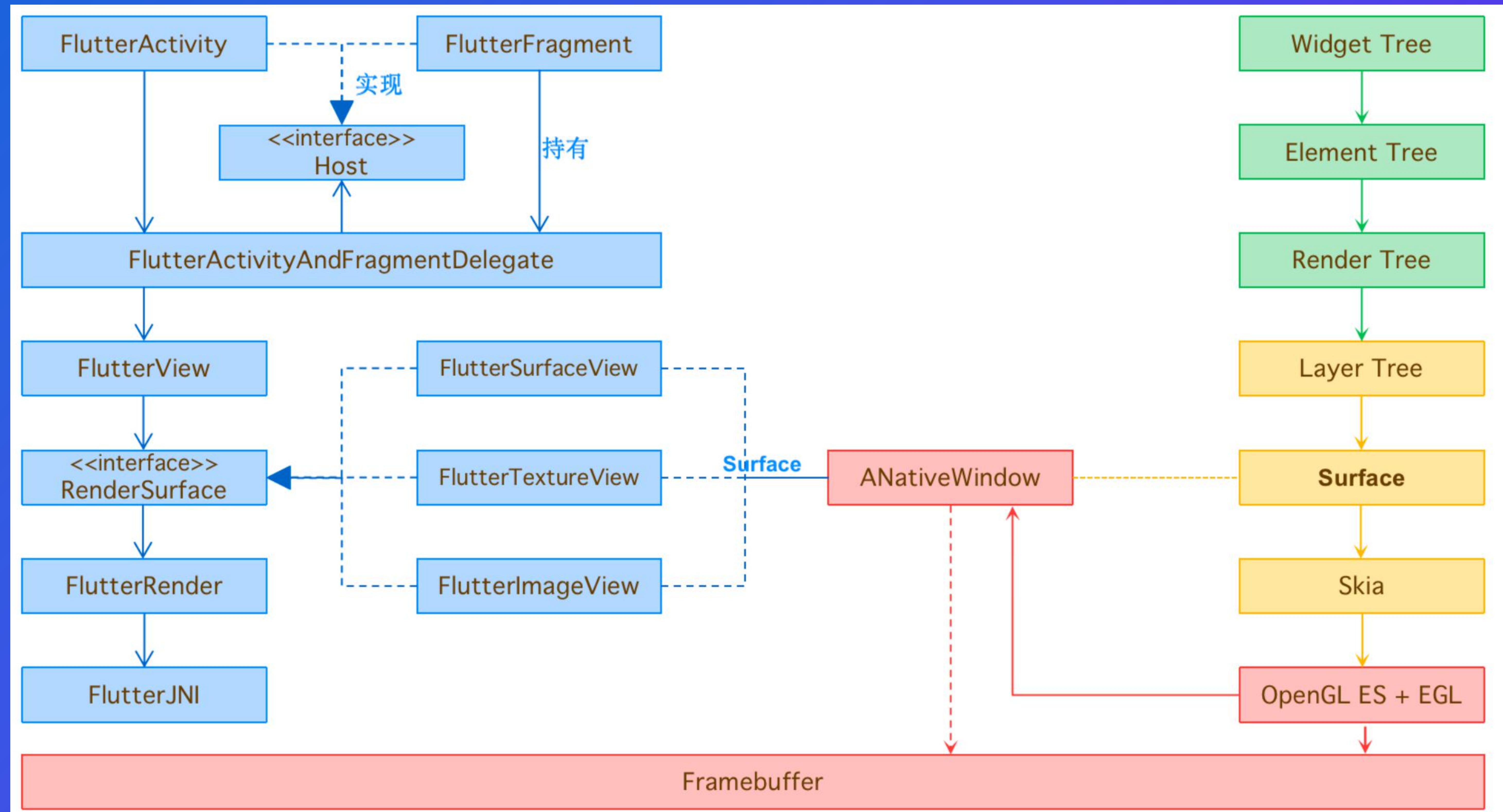
Mobile & PC

Flutter设计浅析



设计 → 实现

Overview



+ Surface



Android

+ CALayer



iOS

+ HWND



Windows

+ GdkWindow



Linux

+ NSView



Mac

+ Canvas



Web

+ Vulkan

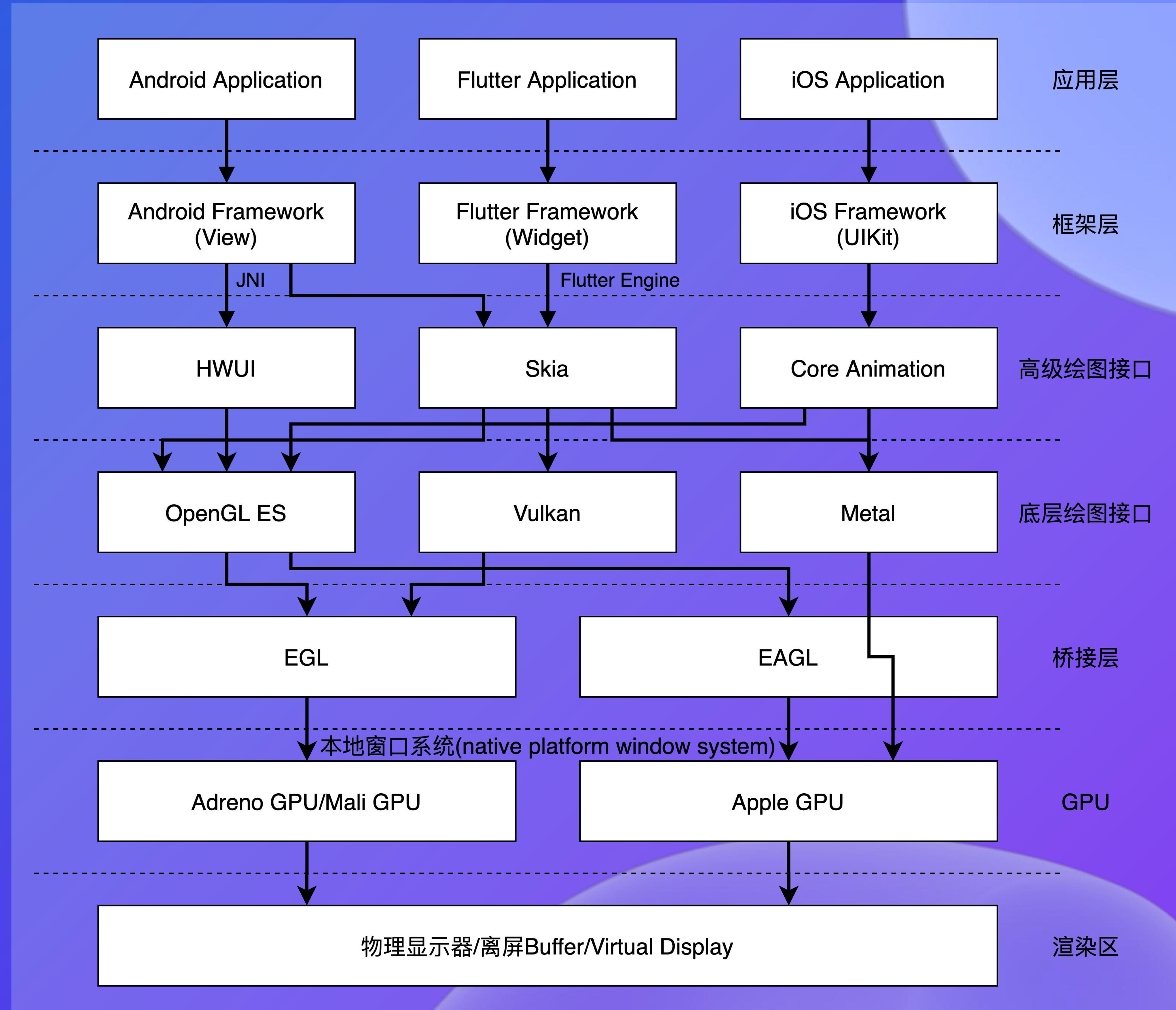
+ Metal

+ DirectX

+ OpenGL

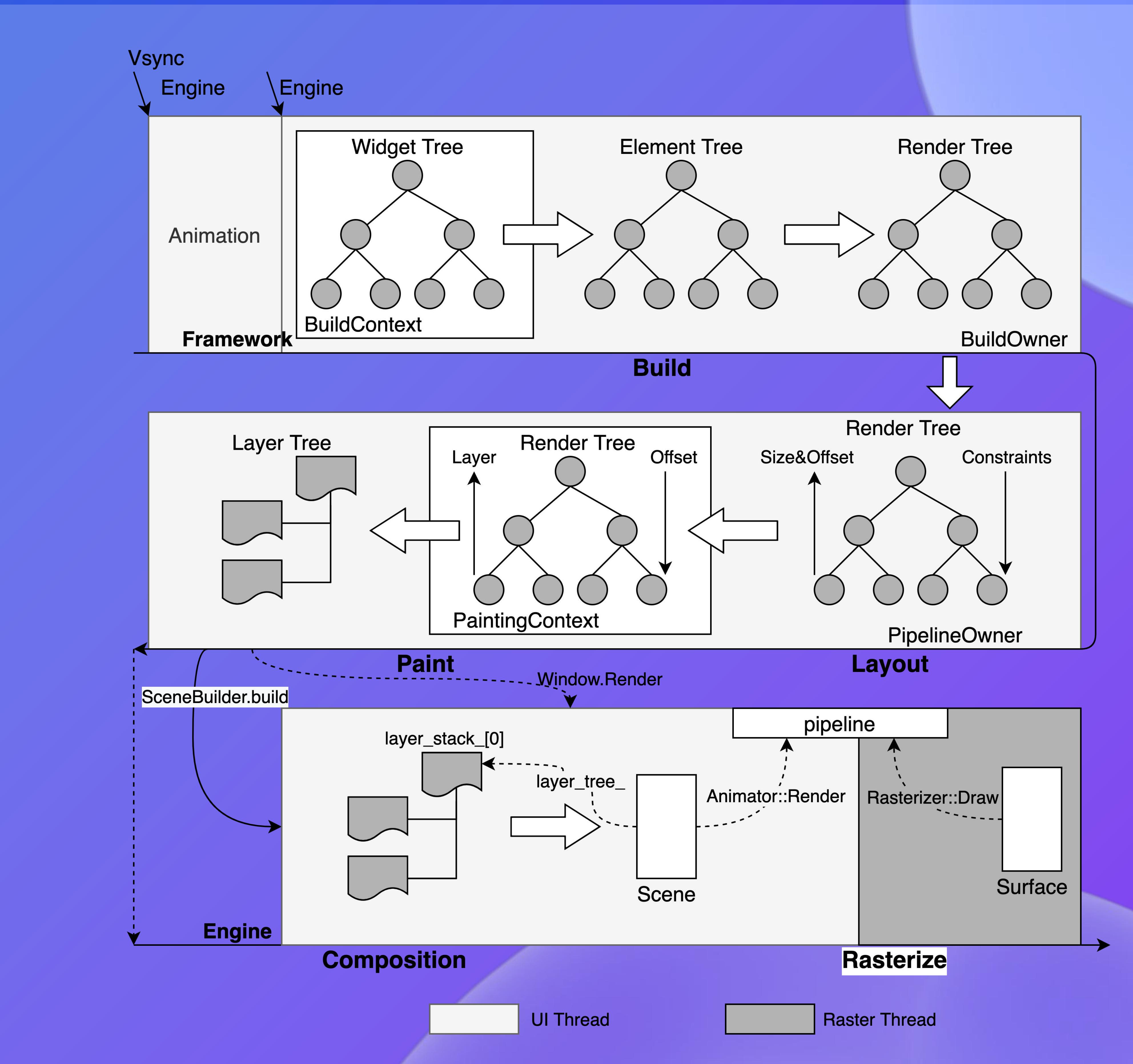
渲染体系

- 用户面向哪一层？
- 业务开发者面向哪一层？
- 系统厂商有何影响？



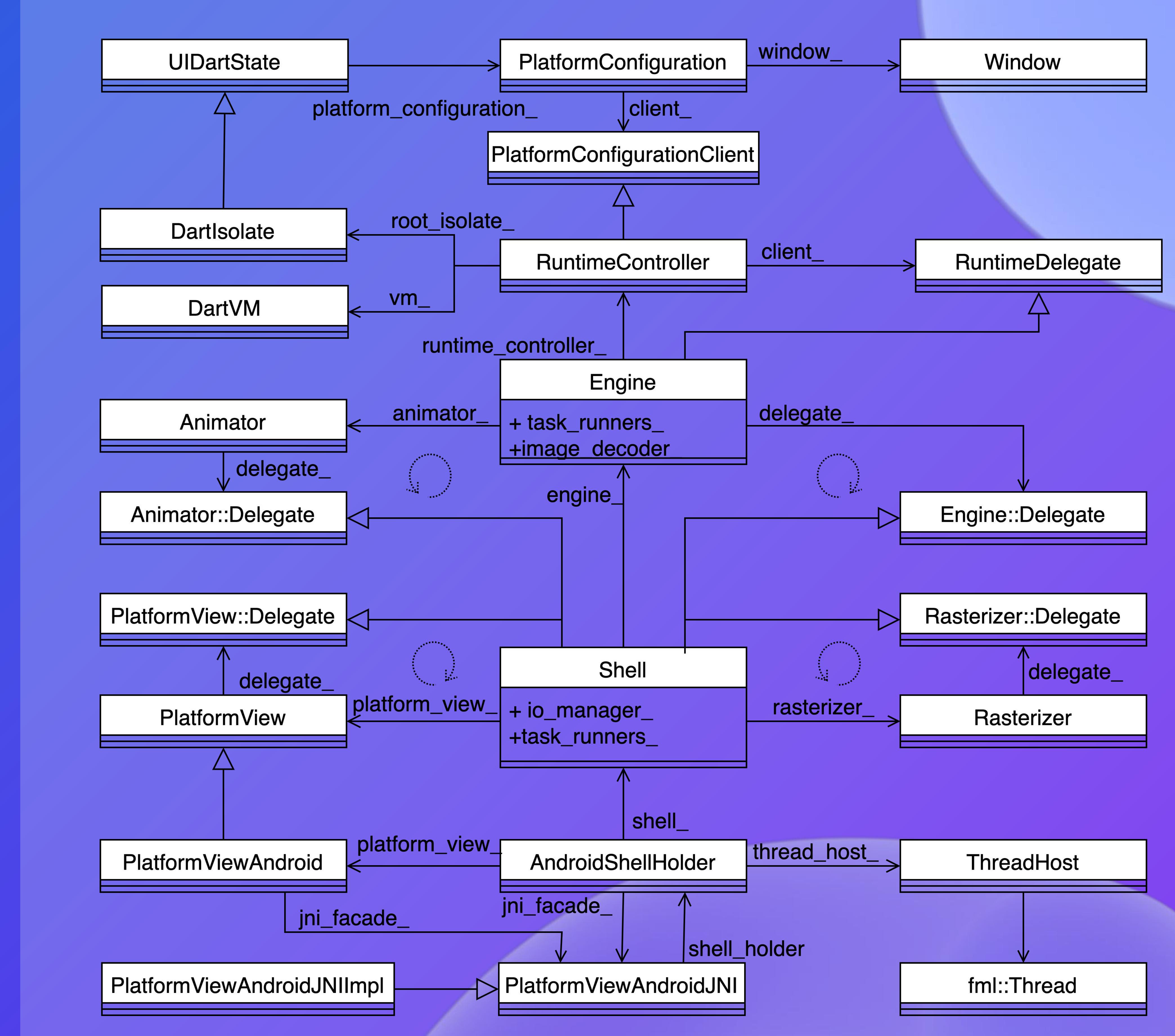
渲染管线

- 渲染管线的作用是什么？
- 生产者、消费者分别是谁？
- Flutter的渲染有哪些优化？



*Engine*的具体实现

- *Engine*的主要职责是什么？
- 涉及哪些线程？
- *Framework*如何与平台通信？

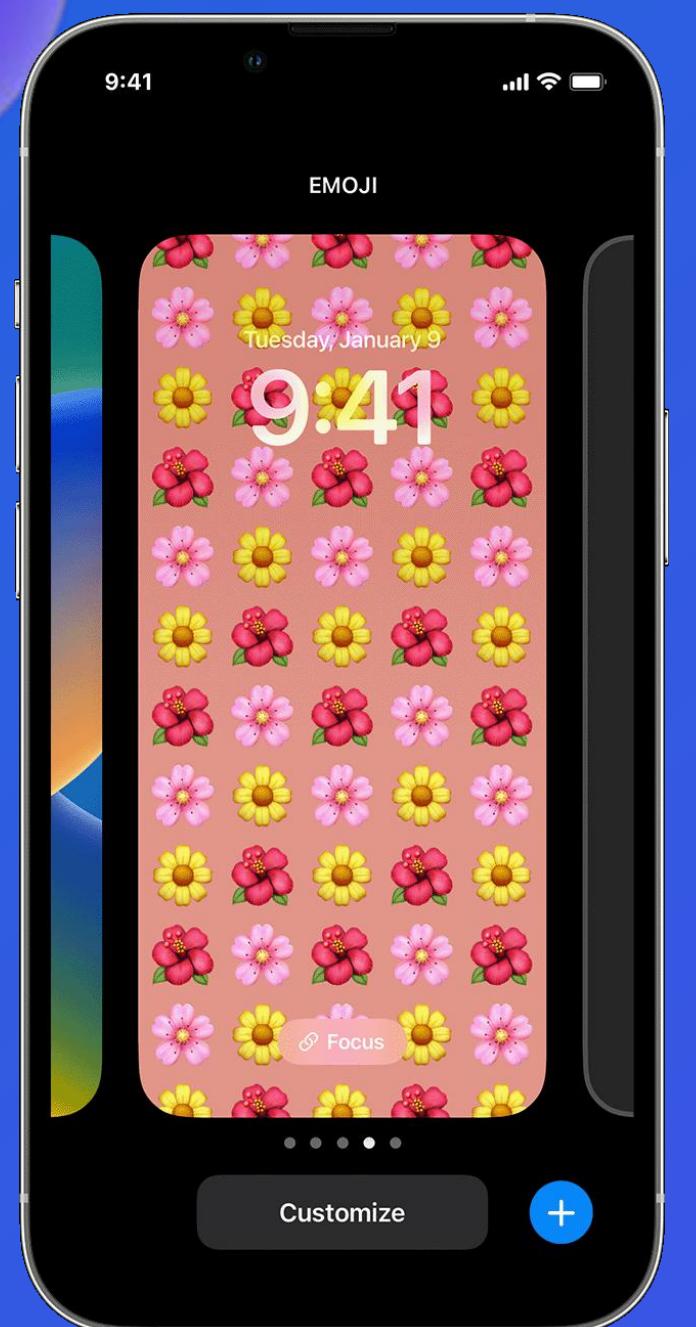
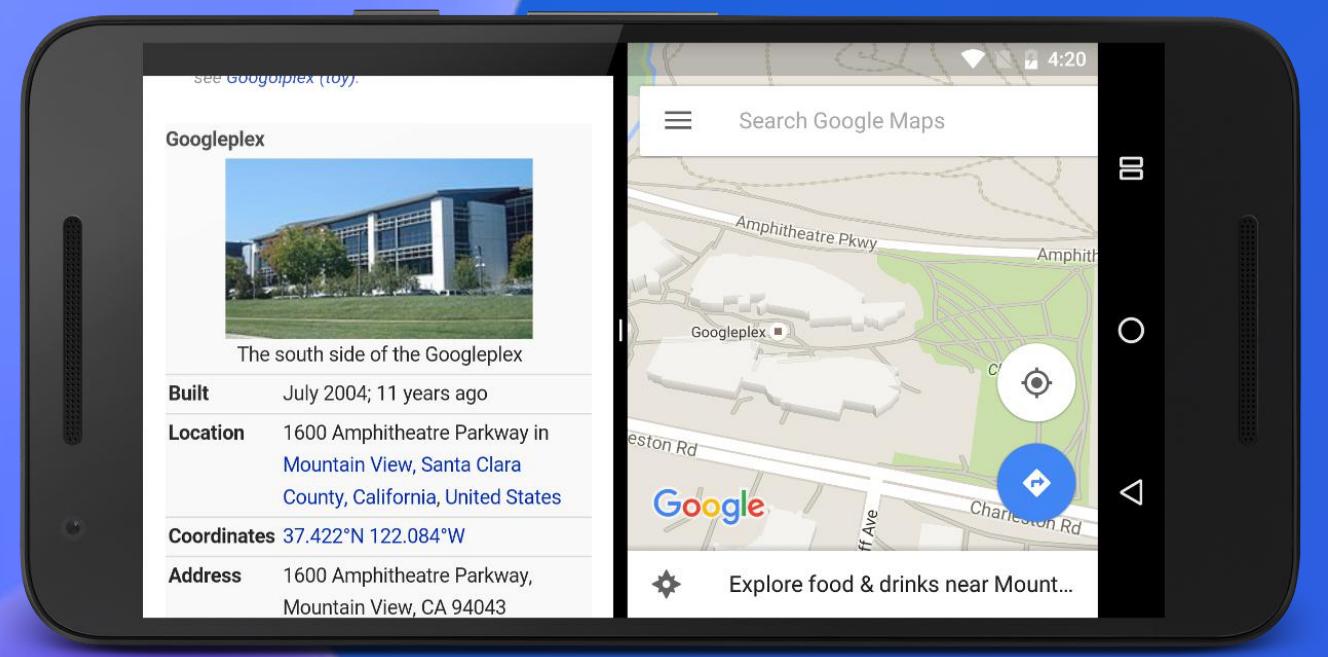


实现 → 演进

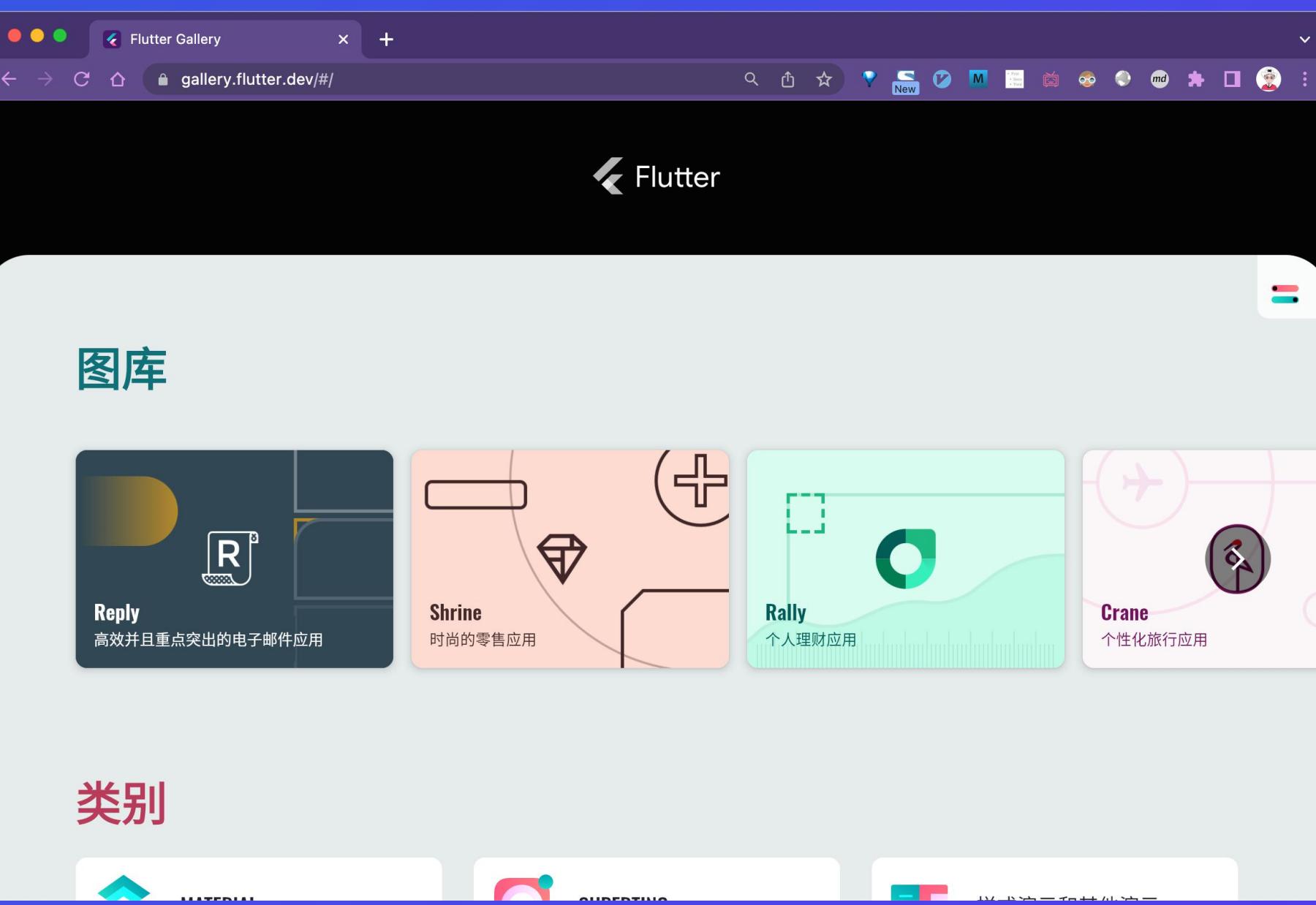
Roadmap

- + 2018: 1.0发布, *Android&iOS*持续完善
- + 2020: 专注质量提升, *Web*和桌面平台的支持
- + 2021: 2.0发布, 正式支持*Web*。
- + 2022: 3.0发布, 支持*Windows/Mac/OSLinux*。开始彻底解决*Jank*。
*DisplayList + Impeller*逐步替代旧有渲染体系。
- + 2023: 全平台 & 渲染 & 开发者诉求

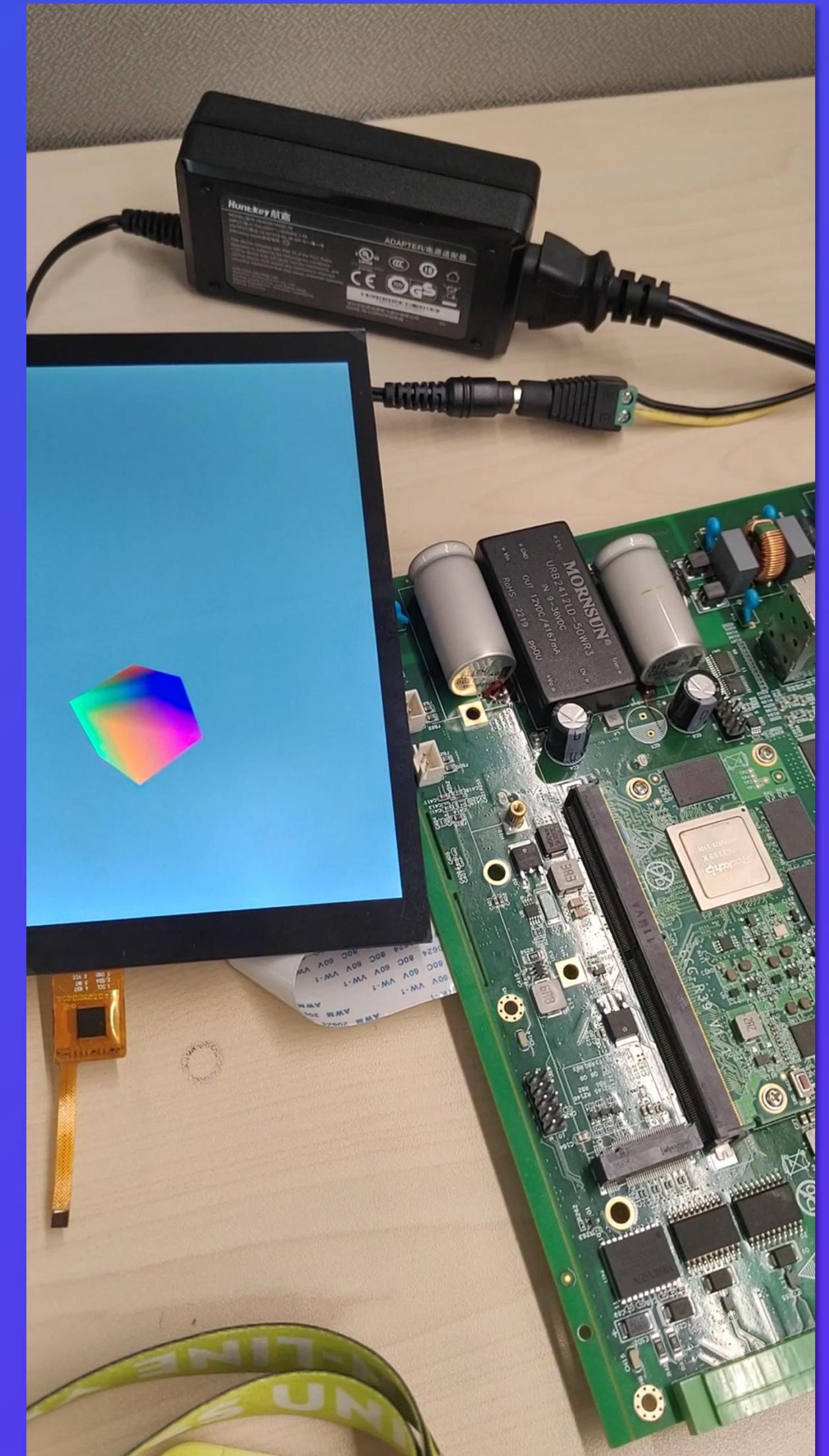
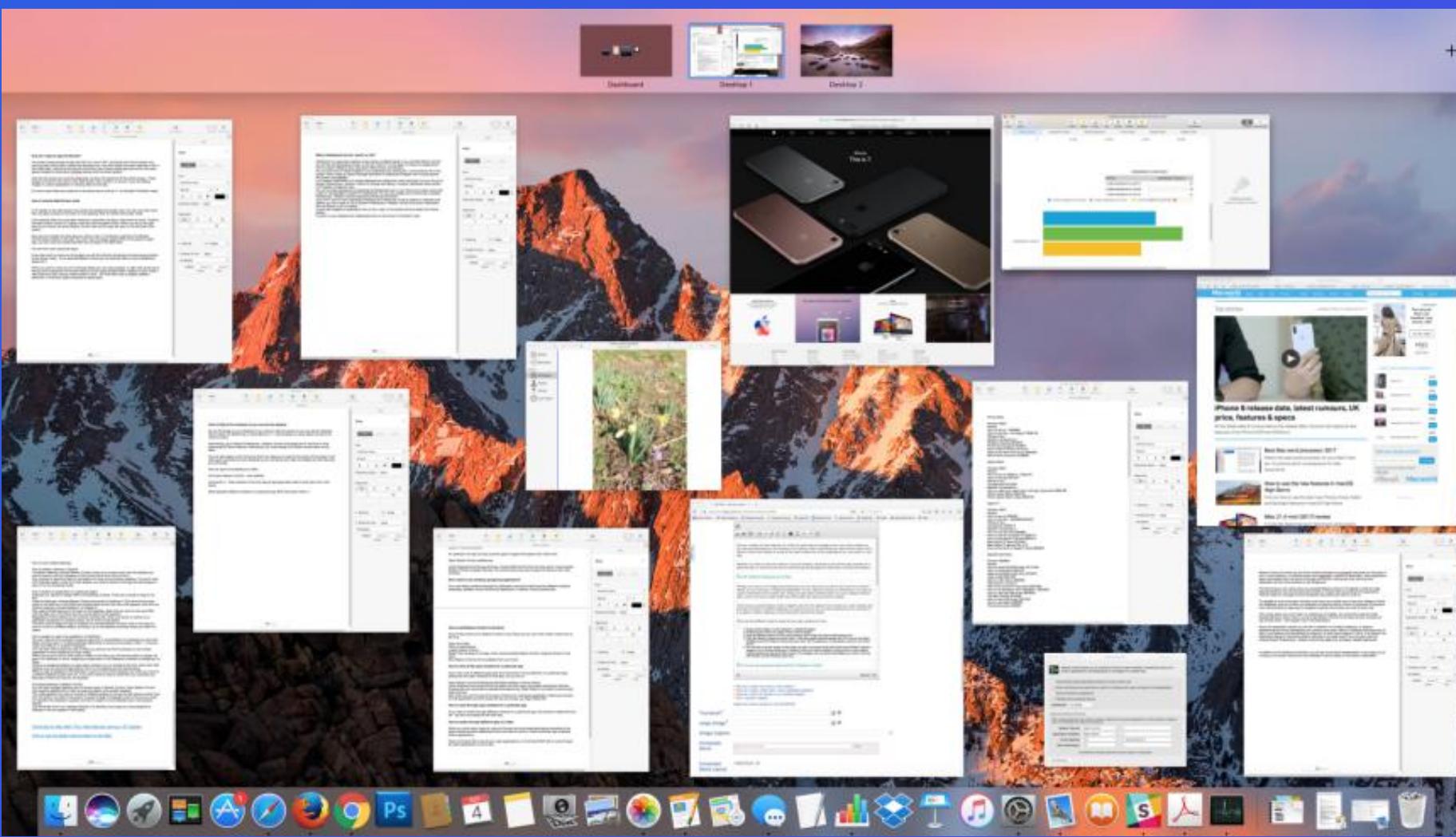
一些挑战



Android&iOS



PC & Web & Embedded



- 更复杂的渲染宿主 [窗口]
- 更复杂的输入系统 [触摸屏、鼠标、键盘、触摸板、手写笔.....]
- 定制的能力，如PC的多窗口、Web的SEO支持
- 更多样化的渲染后端和环境

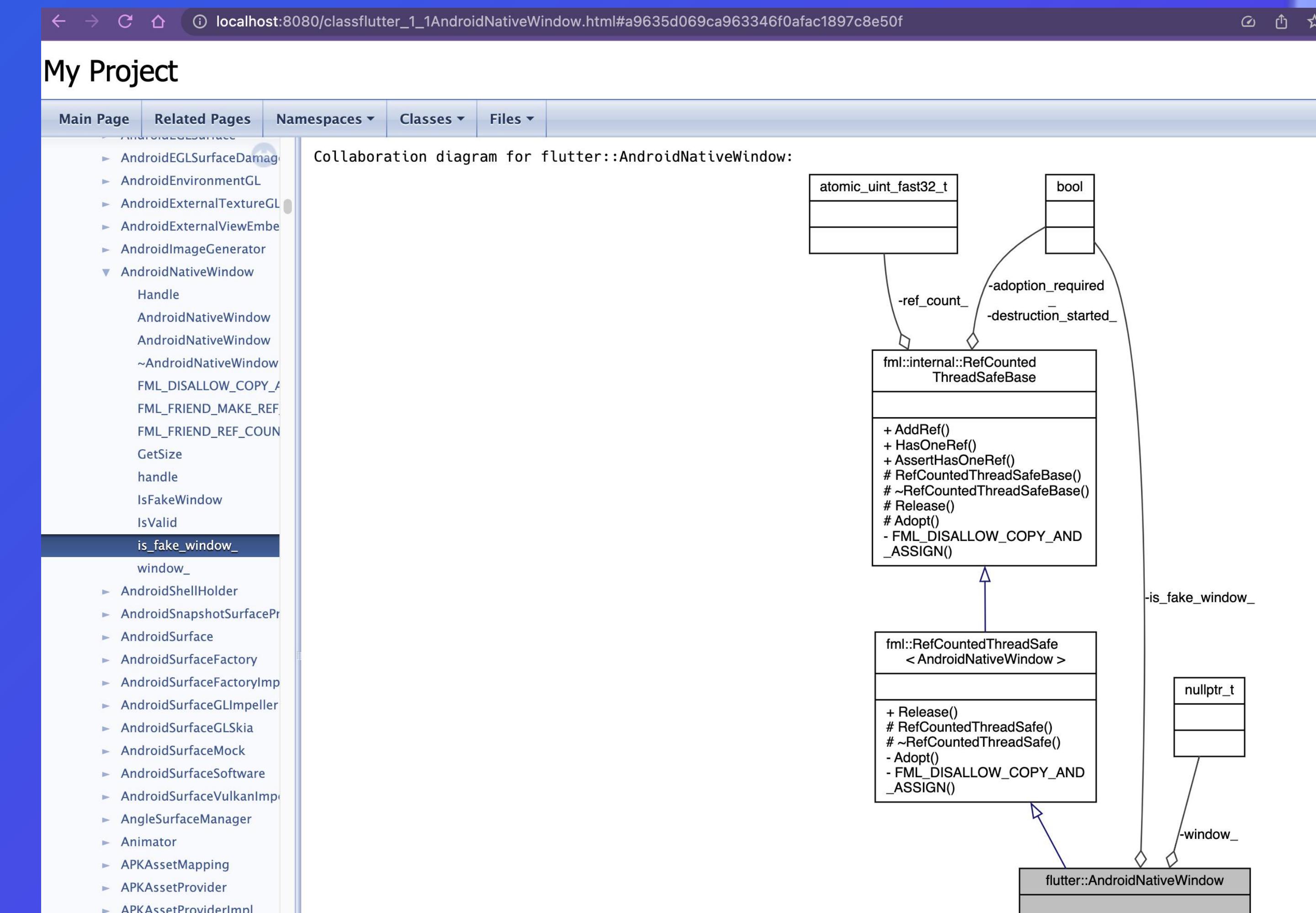
番外

Flutter源码学习技巧

实践

领域

工具

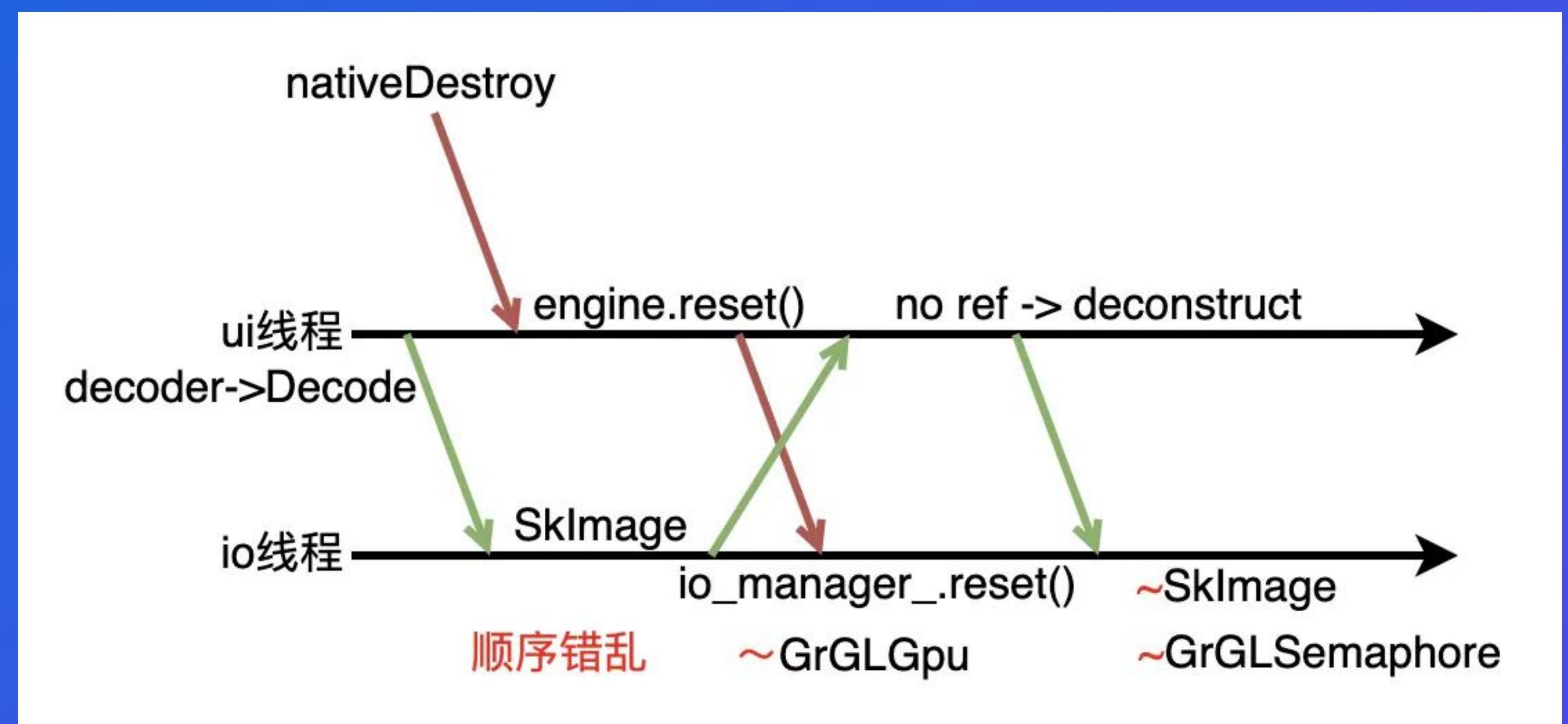


doxygen

```
tmux attach -t 1
17 // |GPUSurfaceGLDelegate|
16 std::unique_ptr<GLContextResult> GLContextMakeCurrent() override;
15
14 // |GPUSurfaceGLDelegate|
13 bool GLContextClearCurrent() override;
12
11 // |GPUSurfaceGLDelegate|
10 bool GLContextPresent(uint32_t fbo_id) override;
19 // |GPUSurfaceGLDelegate|
20 sk_sp<const GrGLInterface> GetGLInterface() const override;
21
22 private:
23     fml::RefPtr<AndroidNativeWindow> native_window_;
24     std::unique_ptr<AndroidEGLSurface> onscreen_surface_;
25     std::unique_ptr<AndroidEGLSurface> offscreen_surface_;
26
27     //-----
28     /// @brief      Takes the super class AndroidSurface's AndroidContext and
29     ///             return a raw pointer to an AndroidContextGL.
30
31 android_surface_gl.h [68 Col 38]     std::unique_ptr<AndroidEGLSurface> onscreen_surface_;
32 android_surface_gl.cc [27 Col 7]     onscreen_surface_(nullptr),
33 android_surface_gl.cc [43 Col 3]     onscreen_surface_ = nullptr;
34 android_surface_gl.cc [67 Col 14]     FML_DCHECK(onscreen_surface_);
35 android_surface_gl.cc [70 Col 15]     if (size == onscreen_surface_->GetSize()) {
36 android_surface_gl.cc [78 Col 3]     onscreen_surface_ = nullptr;
37 android_surface_gl.cc [79 Col 3]     onscreen_surface_ = GLContextPtr()->CreateOnscreenSurface(native_window_);
38 android_surface_gl.cc [80 Col 8]     if (!onscreen_surface_->IsValid()) {
39 android_surface_gl.cc [84 Col 3]     onscreen_surface_->MakeCurrent();
40 android_surface_gl.cc [105 Col 3]    onscreen_surface_ = nullptr;
41
42 -- NORMAL -- --normal --auto-preview location (16/16) /data/research/flutter2.0/src/flutter
```

code index

跨平台开发的Bug斗争案例



<https://github.com/flutter/flutter/issues/87895>



<https://github.com/flutter/flutter/issues/35078>

THANK YOU!

