Flutter课程01：快速入门

一、课前准备

- 观看预习视频
- 安装好Flutter环境

二、课堂目标

- 快速入门Flutter

三、知识要点

- Flutter原理
- Flutter项目的基本架构
  - Stateful Widget
  - Stateless Widget
- 在哪里写?
  - 工欲善其事必先利其器
- 在哪里查文档?
- 如何调试?
  - Dev Tool
- 如何发布?
- 我的第一个Flutter程序《点不到的按钮》

四、总结

- 要入门任何语言，框架，无非先搞懂这几件事。弄明白在哪里写代码，如何调试代码，查看文档，发行/发布。

五、作业

- 实现一个绝对定位的按钮，宽200，高80，背景色 rgb 值为 136, 138, 226，按钮文字颜色设置为白色

1. 电脑选择:

   Window:无无法进行iOS开发 Mac两端都可以开发

   如果需要两端开发需要使用Mac，但是window电脑不影响Flutter的学习 开发工具的选择:

   VSCode:前端开发方便，支持Flutter开发，但是无法调试安卓

   Android Studio:安卓开发必备，同样是支持Flutter开发

Android Studio

http://www.android-studio.org/

Flutter的第一天，主要带大家快速上个手，让大家对这个框架来点兴趣。

Flutter入门

学习Flutter = Flutter（UI库） + Dart（语言）

Dart曾经与typescript竞争，谁才是更好的js？但不幸输给了typescript，但Dart依然是符合ECMA规范，熟练掌握JS的同学要学习Dart并不难，只是会有些不习惯。

2021年大前端有怎样的技术趋势

Flutter 属不属于前端?

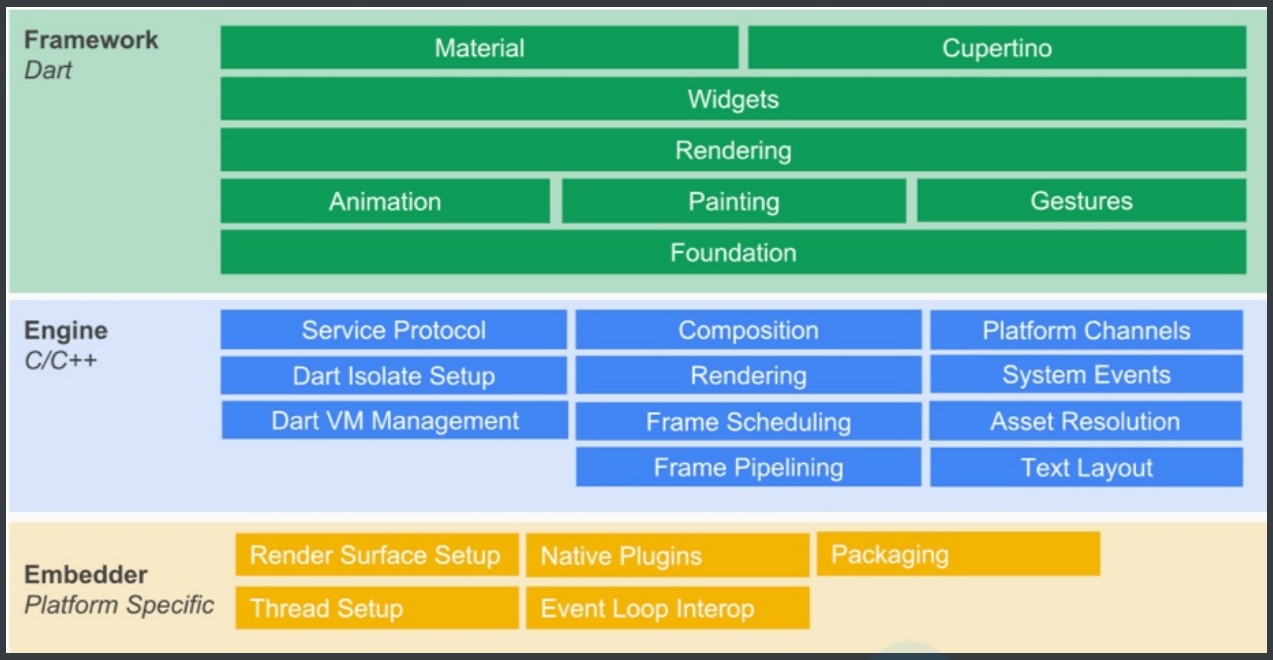
Flutter 的创始人和整个团队几乎都是来自 Web，在 Flutter 负责人 Eric 的相关访谈中， Eric 表示 Flutter 来自 Chrome 内部的一个实验，他们把一些乱七八糟的 Web 规范去掉后，在一些基准测试的性能居然能快 20 倍，因此 Google 内部开始立项，Flutter 出现了。

跨平台在国内运用多吗?

跨平台解决的核心问题就是开发/维护成本问题

Flutter原理

Flutter的架构和渲染机制



可以看到Flutter的架构主要分为三层:Framework，Engine和Embedder。

> 1.Framework使用dart实现，包括Material Design风格的Widget,Cupertino(针对iOS)风格的Widgets，文本/图片/按钮等基础Widgets，渲染，动画，手势等。此部分的核心代码是:flutter仓库下的flutter package，以及sky_engine仓库下的io,async,ui(dart:ui库提供了Flutter框架和引擎之间的接口)等package。

> 2.Engine使用C++实现，主要包括:Skia,Dart和Text。Skia是开源的二维图形库，提供了适用于多种软硬件平台的通用API。

> 3.Embedder是一个嵌入层，即把Flutter嵌入到各个平台上去，这里做的主要工作包括渲染Surface设置,线程设置，以及插件等。从这里可以看出，Flutter的平台相关层很低，平台(如iOS)只是提供一个画布，剩余的所有渲染相关的逻辑都在Flutter内部，这就使得它具有了很好的跨端一致性。

- 这是Flutter标志自己完全不同于RN类框架或其他Hybrid跨端框架的核心要素

我试着用一句话讲明白我理解的flutter为什么如此高性能的原理

和大家互动一下，这个机制很特别吗？大家还知道什么相似机制的？

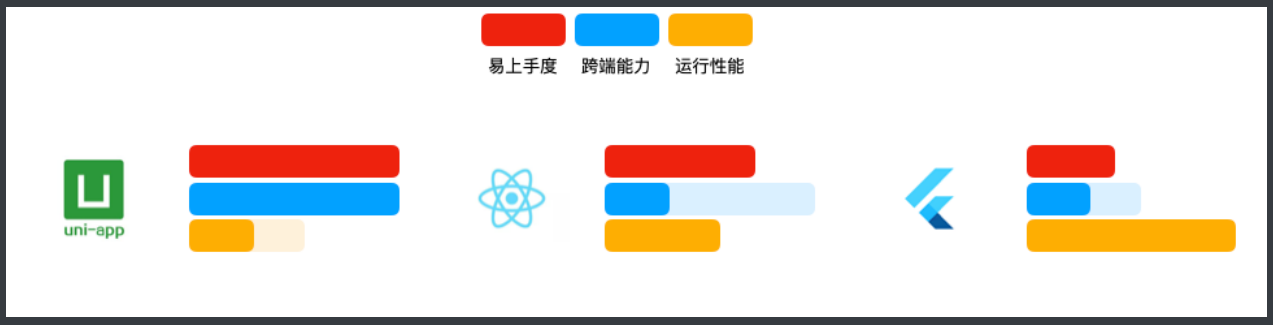
这货就是个跨端”2D游戏“引擎呀（打了引号）

> 一块画布（SGL），使用Dart语言编写Framework制定的API，由Framework调用Engine层实现绘制，Embedder层负责跨端运行和渲染。

再想想那些能跨平台的游戏引擎，是不是差不多就是这么回事?

比较Flutter，ReactNative，Uniapp(weex)

以下对比仅相对于前端开发者而言



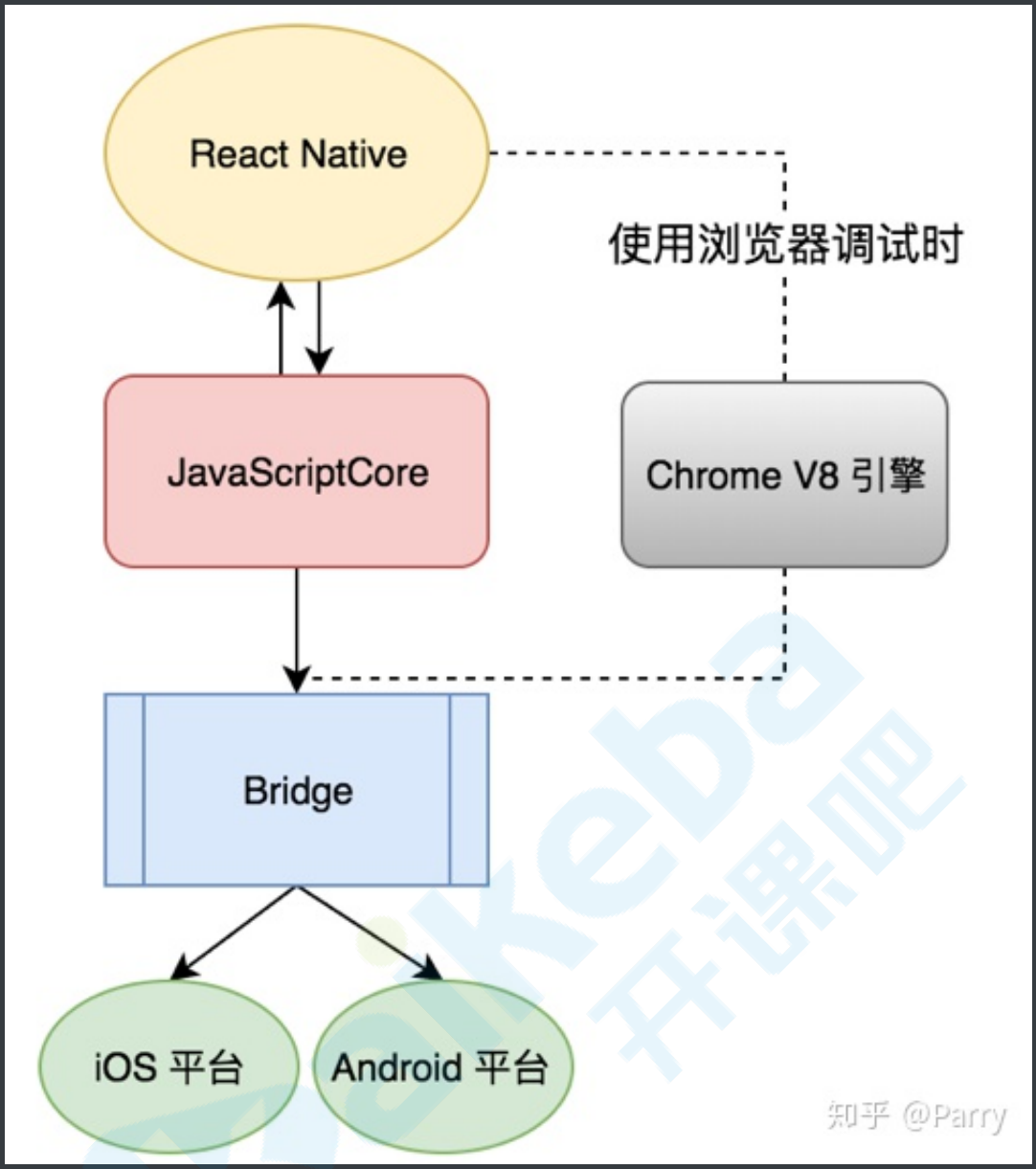
我们知道rn和weex，也是原生渲染的，它们的性能高于webview。但同为原生渲染的，怎么会慢于flutter呢？其实不是原生渲染慢，而是js和原生通信慢。

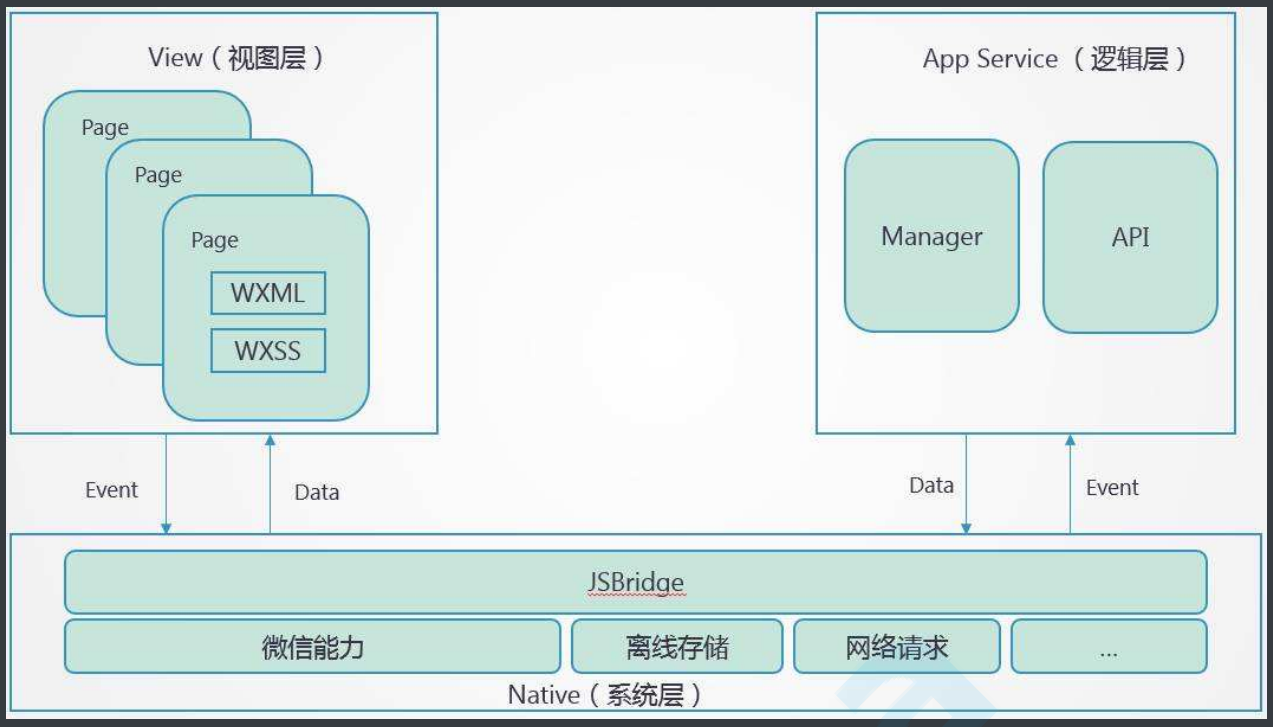
rn和weex都采用了独立的js引擎（iOS是jscore，Android是v8，最新版rn开始在Android上搞自己的js引擎Hermes），从js与dart的比较上，性能稍逊一筹。但这不是主要问题，因为v8的jit不是盖的，也是编译为原生代码解析的。性能上的主要问题是，rn、weex的js引擎和原生渲染层是两个运行环境。

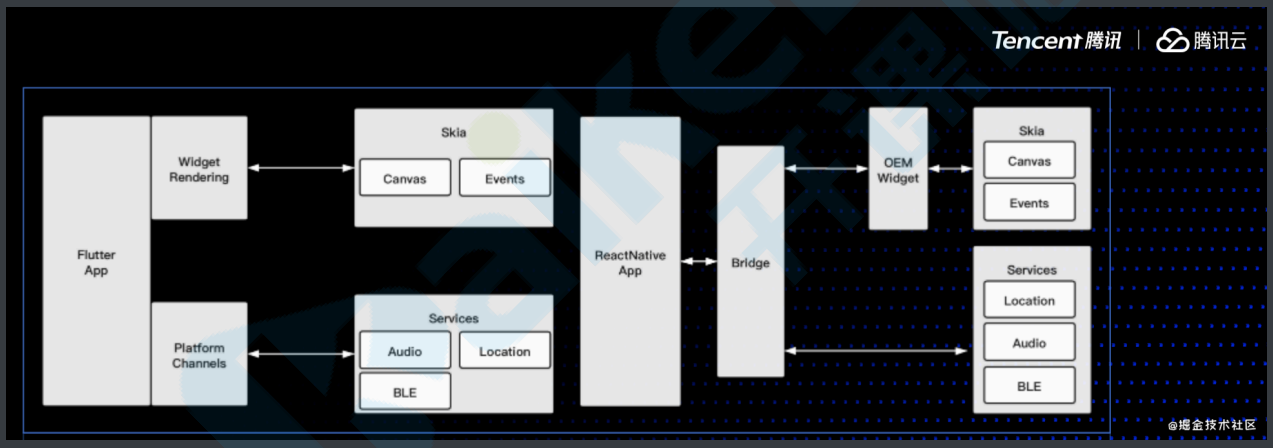
当js引擎联网获取到数据后，通知原生视图层更新界面时，有一个跨环境的通信折损。同样，当用户在屏幕上操作原生视图层时，要给js引擎发送通知，也会产生这个通信折损。

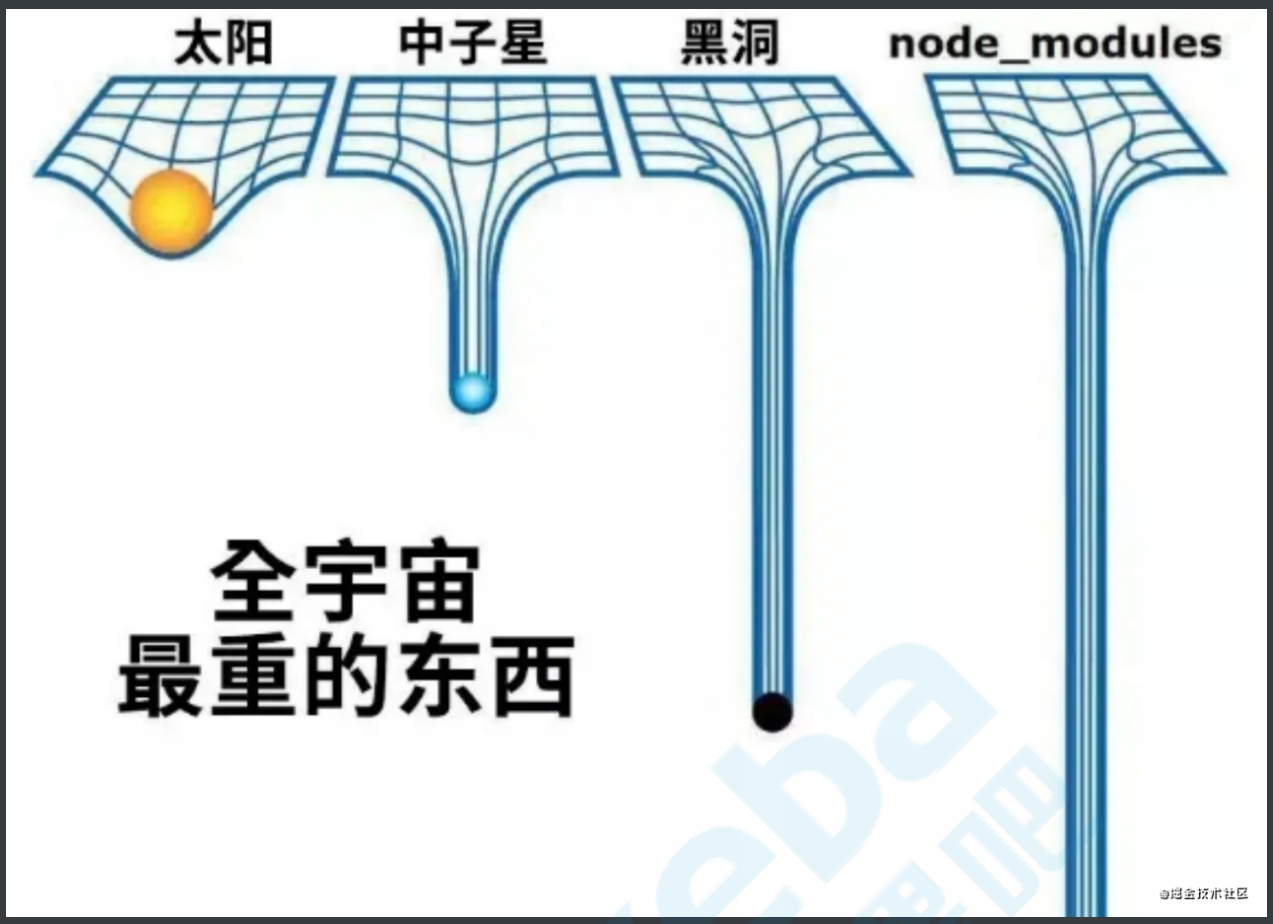
不过这种性能差别，在大多数场景中，用户是感受不到的。比较影响的场景，是跟手式的js响应操作绘制帧动画，或者说js连续操作界面元素方面，flutter折损更少。

这个通信折损，其实普遍存在于所有逻辑和视图分离的框架中，包括各家小程序也有这个问题


React Native
使用浏览器调试时
JavaScriptCore
Chrome V8 引擎
Bridge
iOS 平台
Android 平台
知乎 @Parry


View（视图层）
Page
Page
Page
WXML
WXSS
App Service（逻辑层）
Manager
API
Event
Data
Data
Event
JSBridge
微信能力
离线存储
网络请求
...
Native（系统层）


Tencent 腾讯 | 腾讯云
Flutter App
Widget Rendering
Skia
Canvas
Events
Platform Channels
Services
Audio
Location
BLE
ReactNative App
Bridge
OEM Widget
Skia
Canvas
Events
Services
Location
Audio
BLE
@掘金技术社区



延伸阅读

- 为什么Airbnb放弃了ReactNative

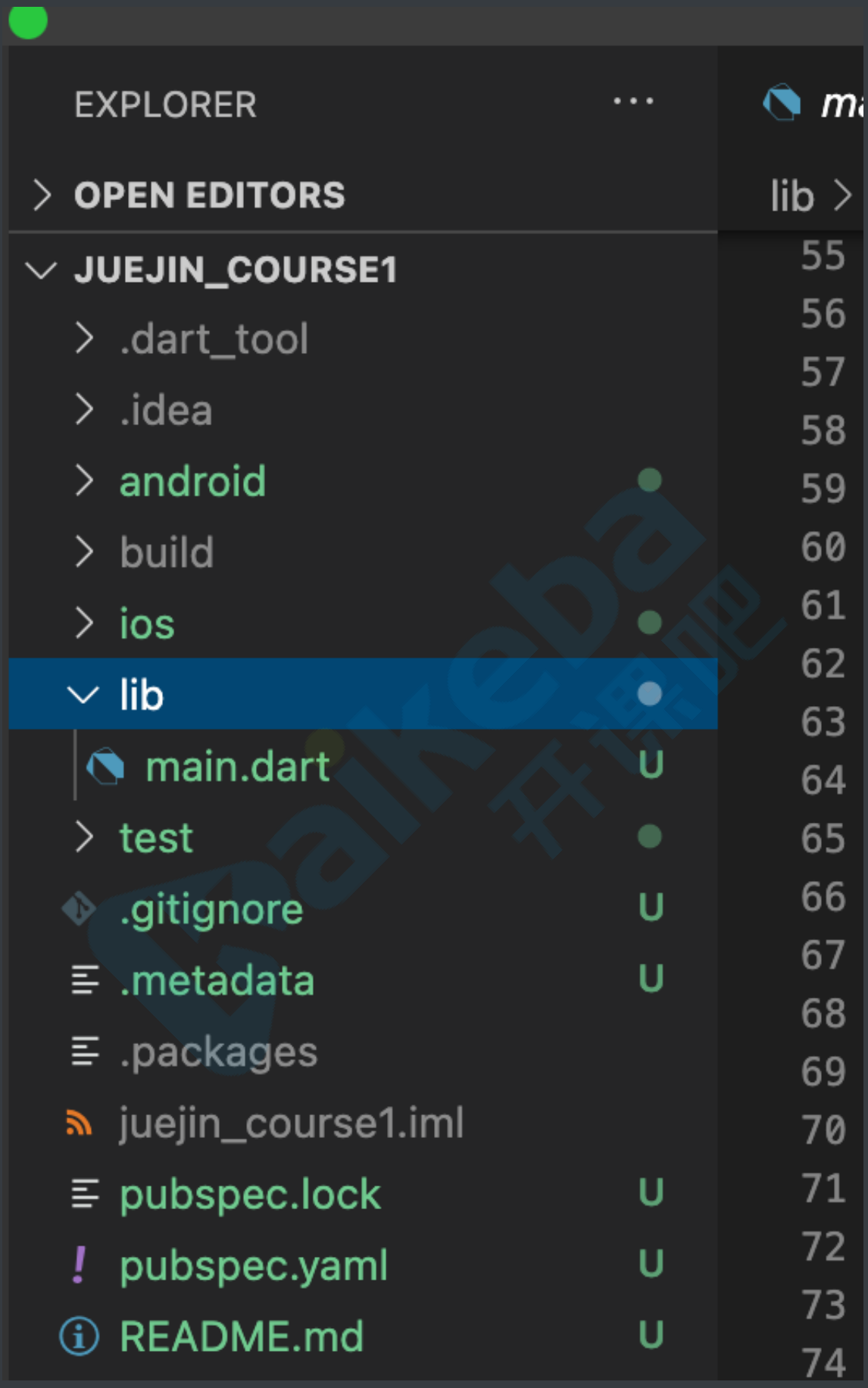
Flutter的劣势

- 混合开发
- 热更新
- 内存占用
- 体积

其实有很多大厂的APP都在同时使用多种混合开发技术

Flutter项目的基本架构

- 创建一个Flutter项目 flutter create myapp


EXPLORER
OPEN EDITORS
JUEJIN_COURSE1
.dart_tool
.idea
android
build
ios
lib
main.dart
U
test
.gitignore
U
.metadata
U
.packages
juejin_course1.iml
pubspec.lock
U
pubspec.yaml
U
README.md
U
lib
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74

在哪里写代码?

- VSCode （咱前端er更喜欢）
- Android Studio

main.dart

工欲善其事必先利其器

- 相关插件Flutter，Dart，Flutter Widget Snippets，Awesome Flutter Snippets
- Bracker Pair Colorizer
- run code

文档！！！

- https://flutterchina.club/
- https://flutter.dev/docs

查看文档的基本概念

- 一切皆是Widgets
- 多组件容器（Row、Column、Stack、Wrap）
- 单组件容器（Container、Padding、Center、Align）
- Futter 组件的规律

Widgets

状态

stateful和**stateless**:实现Flutter app时，我们用用widgets来构建app的UI。这些widgets有两种类型 ——statefule(有状态) 和 stateless(无无状态)

stateless:当创建的widget不不需要管理理任何形式的内部state时，则使用用StatelessWidget。
eg: Text

```
void main() => runApp(MyStatelessWidget(text: "StatelessWidget
Example"));
class MyStatelessWidget extends StatelessWidget {
  final String text;
  MyStatelessWidget({Key key, this.text}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Center(
      child: Text(
        text,
        textDirection: TextDirection.ltr,
      ),
    );
  }
}
```

stateful:当创建一一个能随时间动态改变的widget，并且不不依赖于其初始化状态。eg:Image

注意:

1 创建一个Stateful Widget需要两个类，分别继承自StateFulWidget和State;
2 state对象包含了widget的state和widget的build()方法;
3 当widget的state改变了的时候，当调用setState()方法时，框架就会去调用build方法重绘widget;

运行和调试

- 打印日志

```
print
```

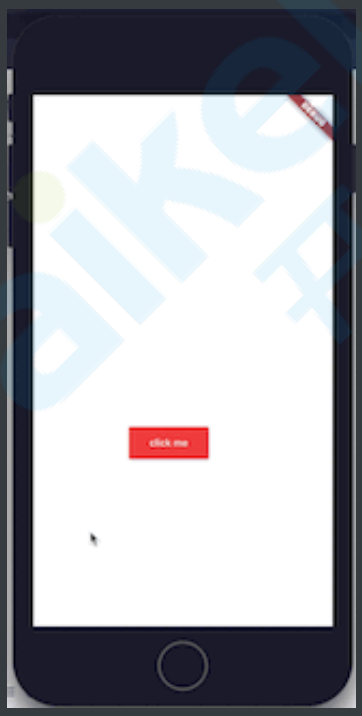
- 断点
- Dev Tool

随便改改

实战《点不到的按钮》

> 为什么实战这个玩意?

> 在界面里放置一个按钮，按钮上文字显示”点我呀“，点击之后按钮就跑到另外一个坐标。这就是点不到的按钮...



需求拆解:

step1 在放按钮之前，我们需要一个承载按钮的布局容器

> 这个按钮点了之后要随机变换位置，所以肯定得用自由布局的组件

查看所有布局组件

```
总之看完所有布局组件的文档，我们知道要用到这个组合
Stack，可以随意堆放子元素的布局
Positioned，绝对定位组件，改变top，left参数就可以跑来跑去啦
```

step2 我们要在这个容器里放一个按钮，按钮上文字显示”点我呀“

```
RaisedButton(
     onPressed: (){
     },
     color: Colors.red,//红色背景
     child:Text(
          'click me',//文本内容
          style: TextStyle(
             color: Colors.white,//文字白色
             fontSize: 12//文字字号
           )
     )
)
```

step3 按钮点击之后要更改其坐标

> 在实现这个之前，我们先要了解，在Flutter中万物皆Widget，而Widget分为无状态组件（StatelessWidget）和有状态组件（StatefulWidget）。简单点说，组件在创建后，是否允许更改状态，一但状态更改，其UI就会重新渲染。

> 这里我们肯定要用有状态组件啦，接下来的代码是在组件创建时，就给与按钮一个随机屏幕坐标，那么之后只要改变状态，就会重新渲染

```
RaisedButton(
     onPressed: (){
          setState((){});//关键是这一句
     },
     color: Colors.red,
     child:Text(
          'click me',
          style: TextStyle(
               color: Colors.white,
               fontSize: 12
          )
     )
)
```

```
//获得当前设备屏幕尺寸，需要import 'dart:ui'
var s = window.physicalSize/window.devicePixelRatio;
//新建一个随机对象，import 'dart:math';
var rng = new Random();
//随机生成按钮的坐标
double btnTop = rng.nextDouble()*(s.height-40);
double btnLeft = rng.nextDouble()*(s.width-100);

return Container(
 color: Colors.white,
 child:Stack(
   children: [
     Positioned(
       top: btnTop,
       left: btnLeft,
       ...
       )
     )
   ]
 )
);
```

那么关键代码就齐全了

作业

实现一个绝对定位的按钮，宽200，高80，背景色 rgb 值为 136, 138, 226，按钮文字颜色设置为白色

暗号：初次见面

全部代码

```
import 'package:flutter/material.dart';
import 'dart:ui';
import 'dart:math';

void main() {
  runApp(MyApp());
}


class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      home: MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```

```
class MyHomePage extends StatefulWidget {
  MyHomePage({Key key, this.title}) : super(key: key);

  final String title;

  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {

  @override
  Widget build(BuildContext context) {
    var s = window.physicalSize/window.devicePixelRatio;
    print(s);

    var rng = new Random();

    double btnTop = rng.nextDouble()*(s.height-40);
    double btnLeft = rng.nextDouble()*(s.width-100);

    print(btnTop);
    print(btnLeft);

    return Container(
      color: Colors.white,
      child:Stack(
        children: [
          Positioned(
            top: btnTop,
            left: btnLeft,
            width: 100,
            height: 40,
            child: RaisedButton(
              onPressed: (){
                setState((){});
              },
              color: Colors.red,
              child:Text(
                'click me',
                style: TextStyle(
                  color: Colors.white,
```

```
                    fontSize: 12
                  )
                )
              )
            )
          ]
        )
      );
    }
  }
```