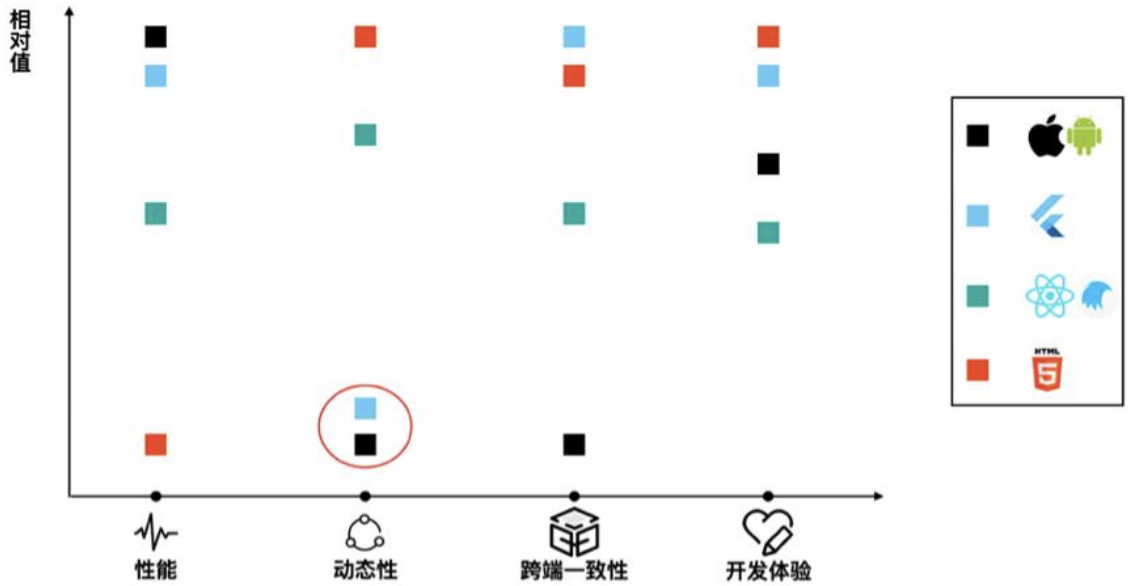
Flutter

Flutter是谷歌的移动UI框架，可以快速在iOS和Android上构建高质量的原生用户界面。Flutter可以与现有的代码一起工作。在全世界，Flutter正在被越来越多的开发者和组织使用，并且Flutter是完全**免费**、**开源**的。



[Flutter VS React Native VS Native，谁才是性能之王]

Flutter 优缺点

> 优点

1. **性能强大，流畅**

Flutter对比weex和react native相比，性能的强大是有目共睹的。基于dom树渲染原生组件，很难与直接在原生视图上绘图比肩性能，flutter直接在两个平台上重写了各自的UIKit，对接到平台底层，减少UI层的多层转换，UI性能可以比肩原生，这个优势在**滑动**和**播放动画**时尤为明显。

2. **路由设计优秀**

Flutter的路由传值非常方便，push一个路由，会返回一个Future对象（也就是Promise对象），使用await或者.then就可以在目标路由pop，回到当前页面时收到返回值。这个反向传值的设计基本是甩了微信小程序一条街了。弹出dialog等一些操作也是使用的路由方法，几乎不用担心出现传值困难

3. **UI跨平台稳定、可选静态的语言**

> 缺点

1. **糟糕的UI控件API**

2. 新语言Dart

Dart基础语法

1. `var` 一种声明变量而不指定其类型的方法。

```
var content = 'Dart 语法';
```

2. `dynamic` 数据类型是动态可变的，也可以定义任何变量，但是和var 不同的是，var 一旦赋值后，就不能改变数据类型了，例如以下用法就是错误的

```
dynamic example = 'example';

var content = 'Dart 语法';
content = 1; //❌ 错误的使用方法，content为String，不能赋值数字类型

dynamic example = 'example';
example = 1;//✅  这个使用方法正确，因为 dynamic 的类型是动态可变的
```

3. `Object` Dart 里所有东西都是对象，是因为 Dart 的所有东西都继承自 Object，因此 Object 可以定义任何变量，而且赋值后，类型也可以更改：

```
Object index = 100;
index = 'string';//✅ 因为  'String' 也是 Object
```

4. 常量：`final` 和 `const` 如果你不想更改变量的值，那么你可以用 final 和 const:

```
final content = 'Dart 语法';
static const bool switchOn = false;
```

> 使用 `final` 和 `const` 的时候可以把 `var` 省略
>
> `final` 和 `const` 变量只能赋值一次，而且只能在声明的时候就赋值
>
> `const` 是隐式的 `final`
>
> 在使用 `const` 的时候，如果变量是类里的变量，必须加 `static` ，是全局变量时不需要加

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

const demoConst = 'demo'; // 这里不用加 static

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {

    static content = 'Dart 语法'; // 这里必须加 static
    ...
}
```

> 那么final 和 const 有什么区别呢?
>
> **const 是编译时常量，在编译的时候就初始化了，但是 final 变量是当类创建的时候才初始化。**

5. `$variableName` 或 `${expression}` 字符串插值：将变量的值直接插入字符串中。
6. 函数:必选参数和可选参数

    1. 可选命名参数：使用 {} 包起来的参数是可选命名参数, 同时还可以给命名参数加 `@required`，意思是这个也是必填参数，例子如下：

```
 bool say(String msg , {@required String from, int clock}){
    print(msg+" from " + from + " at " + clock.toString());
    return true;
}

 say('Hello Flutter');//❌ 错误调用方式，因为 from 是必选参数，不填的话会报错

say('Hello Flutter',from: 'XiaoMing');//✅ 正确调用方式
say('Hello Flutter',from: 'XiaoMing',clock: 11);//✅ 这个调用方式也是正确的
```

    2. 可选位置参数：使用 [] 包起来的参数是可选位置参数

```
bool say(String msg , [String from , int clock]){
    print(msg+" from " + from + " at " + clock.toString());
    return true;
}

// 要给可选位置参数赋值时，必选按照顺序来赋值：
say('Hello Flutter');//✅ 因为 from 和 clock 是可选参数，所以可以不填

say('Hello Flutter','XiaoMing',1);//✅ 为可选位置参数赋值，只能一个参数一个参数对应的赋值，所以要全部赋值

say('Hello Flutter','XiaoMing')//✅
say('Hello Flutter',1)//❌ 因为  1 赋值给了 from,但是 from 是String，所以会报错
```

创建实例：不需要使用 new

创建类实例的时候，都要写 new，其实很麻烦的，而且也没有必要，所以 Dart 在创建实例的时候不在需要使用 new 。

类型判断操作符

下面是 Dart 支持的检查运行时类型的操作符：

操作符	含义	例子
`as`	类型转换	(emp as Person).firstName = 'Bob';

`is`	判断是否是某个类型,如果是的话，就返回 true	if (emp is Person) {
		// 如果 emp 是 Person 类型
		emp.firstName ='Bob';
		}

`is!`	判断是否不是某个类型，如果不是的话，就返回 true	if (emp is! Person) {
		// 如果 emp 不是 Person 类型
		}

> 上面的例子中，如果 emp 是 null 的话，as 的例子就会抛异常，is 和 isn't 的例子会返回 false.

条件运算符

Dart 有两个运算符，可以让您使用更简单的表达式来代替可能需要 if-else 语句的表达式：

1. `condition ? expr1 : expr2`

   如果 condition 是 true，返回 expr1，否则返回 expr2。

   当你需要根据一个 boolean 表达式来赋值时，可以使用 `? :`,例如：

   ```
   var visibility = isPublic ? 'public' : 'private';
   ```

2. `expr1 ?? expr2`

   如果 expr1 为 null，就返回 expr2 的值，否则返回 expr1 的值。

   如果需要根据一个 boolean 表达式是否为 null 来作为条件，可以使用 `??`，例如：

   ```
   String playerName(String name) => name ?? 'Guest';
   ```

一、UI框架

Flutter作为一门新**开源**的移动UI框架，在设计之初，谷歌的工程师就考虑到了开发者在学习Flutter的成本问题，Dart语言与Flutter的结合，获得泛型、class等强类型语言的特性保证了我们开发的应用安全可控，同时Flutter借鉴了FaceBook成熟开源框架React的单向数据绑定的特性，使我们在开发的过程中可以恰到好处的更新和控制我们的页面。

掌握Widget，在Flutter中，万物皆Widget！Widget作为我们搭建应用的组件，需要至少掌握我们常见的Widget

Widget	说明
Container	一个拥有绘制、定位、调整大小的 widget。
Row	在水平方向上排列子widget的列表。
Column	在垂直方向上排列子widget的列表。

Image	一个显示图片的widget
Text	单一格式的文本
Icon	A Material Design icon.
RaisedButton	Material Design中的button， 一个凸起的材质矩形按钮
Scaffold	Material Design布局结构的基本实现。此类提供了用于显示drawer、snackbar和底部sheet的API。
Appbar	一个Material Design应用程序栏，由工具栏和其他可能的widget（如TabBar和FlexibleSpaceBar）组成。
ListView	列表显示
Icon	图标
Switch	单选开关
Checkbox	复选框
TextField	输入框
Form	表单组件
Flex、Expanded	弹性布局
Wrap	流式布局
Stack、Positioned	层叠布局，用于页面定位，层叠摆放
Align	对齐与相对定位
GestureDetector	手势识别

> Widget拓展 进阶学习：flutter.cn/docs/

二.状态

StatelessWidget 和 StatefulWidget组件

自定义一个组件就需要继承 StatelessWidget 或者 StatefulWidget

StatelessWidget： 无状态组件，状态不可以发生改变的widget

StatefullWidget： 有状态组件，持有的状态可能在widget 生命周期改变 ，可以通过setState类更新UI在需要更新页面的时

候调用setState方法 组件内部的widget build 方法会重新执行达到刷新界面的效果

调用流程：setState() -- element.markNeedsBuild() -- BuildOwner 类 scheduleBuildFor() -- SchedulerBinding类ensureVisualUpdate() -- scheduleFrame() --

*调用setState()是至关重要的，因为这告诉框架，widget的状态已经改变，应该重绘

三.页面交互

1.GestureDetector

许多组件是没有onPressed 方法，还是实现交互可以使用这个类，上面中,onTap相当于onPressed方法,响应点击事件,而被GestureDetector包裹的可以是图片或者Column,或者其他的组件

onTapDown ： 在特定的位置轻触手势接触了屏¬幕

onTap：单击操作

onTapUp：在特定的位置产生了一个轻触手势并且停止接触屏幕

onTapCancel：触发了onTapDown，但是没有触发onTap

onDoubleTap：双击

onLongPress：长按事件

2.页面跳转路由设置

1.直接跳转通过 Navigator.push

```
eg：  Navigator.push(context, MaterialPageRoute(builder: (context) {
        return ParentWidget();
      }));
```

退出页面：Navigator.pop();

2.通过MaterialApp中“routes”设置页面路由，initRoute 设置默认加载页面。

eg：Navigator.pushNamed(context,"/test");

跳转传值：

```
eg：  Navigator.pushNamed(context, "/test3",arguments: {
                "data":"跳转带参数的界面,拿到了数据"
            });
```