Flutter知识点

江河_ios

1 Flutter是什么?

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。

优点

- 热重载（Hot Reload），利用Android Studio直接一个ctrl+s就可以保存并重载，模拟器立马就可以看见效果，相比原生慢长的编译过程强很多；
- 一切皆为Widget的理念，对于Flutter来说，手机应用里的所有东西都是Widget，通过可组合的空间集合、丰富的动画库以及分层扩展的架构实现了富有感染力的灵活界面设计；
- 借助可移植的GPU加速的渲染引擎以及高性能本地代码运行时以达到跨平台设备的高质量用户体验。 简单来说就是：最终结果就是利用Flutter构建的应用在运行效率上会和原生应用差不多。

缺点

- 不支持热更新；
- 三方库有限，需要自己造轮子；

- Dart语言编写，增加了学习难度，并且学习了Dart之后无其他用处，相比JS和Java来说。

2 Flutter特性有那些?

1 跨平台

2 热重载。

3 Flutter 中的生命周期

1. initState：插入渲染树时调用，只调用一次，widget创建执行的第一个方法，可以再里面初始化一些数据，以及绑定控制器。

2. didChangeDependencies：在 initState() 后调用。当State对象的依赖发生变化时会被调用；例如：在之前build() 中包含了一个InheritedWidget，然后在之后的build() 中InheritedWidget发生了变化，那么此时InheritedWidget的子widget的didChangeDependencies()回调都会被调用。InheritedWidget这个widget可以由父控件向子控件共享数据，案例可以参考 scoped_model开源库。

3. build： 它主要是用于构建Widget子树的，调用次数：多次，初始化之后开始绘制界面，当setState触发的时候会再次被调用

4. didUpdateWidget：组件状态改变时候调用，可能会调用多次

5. deactivate:当 State 对象从树中被移除时，会调用此回调。

6. dispose():当 State 对象从树中被永久移除时调用；通常在此回调中释放资源。

7. reassemble：此回调是专门为了开发调试而提供的，在热重载 (hot reload) 时会被调用，此回调在 Release 模式下永远不会被调用。

4 flutter widget、Element、RenderObject 关系

· Widget 实际上就是 Element 的配置数据，Widget 树实际上是一个配置树，而真正的 UI 渲染树是由 Element 构成；不过，由于 Element 是通过 Widget 生成，所以它们之间有对应关系，所以在大多数场景，我们可以宽泛地认为 Widget 树就是指 UI 控件树或 UI 渲染树。
· 一个 Widget 对象可以对应多个 Element 对象。这很好理

解，根据同一份配置（Widget），可以创建多个实例（Element）。

· 从创建到渲染的大体流程是：根据 Widget 生成 Element，然后创建相应的 RenderObject 并关联到 Element.renderObject 属性上，最后再通过 RenderObject 来完成布局排列和绘制。

flutter 使用 RenderObjects 管理传统 UI 对象的许多职责（例如维护布局的状态）。RenderObjects 在帧之间保持不变，flutter 的轻量级 Widgets 告诉框架在状态之间改变 RenderObjects。

· Widget 仅用于存储渲染所需要的信息。

· RenderObject 负责管理布局、绘制等操作。

· Element 才是这颗巨大的控件树上的实体。

Layer

iOS 的每一个 UIView 都有一个 layer，flutter 的 render object 不一定存在 layer，一般情况下一个 renderObject 子树都渲染在一个 layer 上，那么什么 renderObject 具有 layer，子 renderObject 怎么渲染到这个 layer?

- 当一个 renderObject 的 alwaysNeedsCompositing == true 或者 isRepaintBoundary == true，renderOject 会有对应的 compositing layer
- 子 renderObject 会对目标 layer 返回对应的 offsetLayer,

目标compositing layer再根据offset合成一个渲染的纹理buffer

5 didChangeDependencies有两种情况会被调用。

1 创建时候在initState 之后被调用

2 在依赖的InheritedWidget发生变化的时候会被调

6 Flutter跟ReactNative、weex 等有什么不同?

1 WebViews 最早的跨平台方案是基于 JaveScript 和WebView的，像PhoneGap、Cordova、Ionic等。UI通过WebView来显示html代码，系统服务则通过一个中间层桥接到JaveScript中去。

2 原生 App

苹果2008年发布iOS，Google 2009年发布Android，它们的SDK是基于两种不同的编程语言Objective-C 和 Jave.现在又有了Swift和Kotlin

3 React Native RN不仅桥接系统服务，也将系统UI也桥接到了JaveScript中，这样写出来的UI最终也会渲染成原生的控件。

4 Flutter Flutter使用Dart语言开发，Dart可以被编译

（AOT）成不同平台的本地代码，让Flutter可以直接和平台通讯而不需要一个中间的桥接过程，从而提高了性能。

7 Widget的两种类型是什么?

1 StatelessWidget: 一旦创建就不关心任何变化，在下次构建之前都不会改变。它们除了依赖于自身的配置信息（在父节点构建时提供）外不再依赖于任何其他信息。比如典型的Text、Row、Column、Container等，都是StatelessWidget。它的生命周期相当简单：初始化、通过build()渲染。

2 StatefulWidget: 在生命周期内，该类Widget所持有的数据可能会发生变化，这样的数据被称为State，这些拥有动态内部数据的Widget被称为StatefulWidget。比如复选框、Button等。State会与Context相关联，并且此关联是永久性的，State对象将永远不会改变其Context，即使可以在树结构周围移动，也仍将与该context相关联。当state与context关联时，state被视为已挂载。StatefulWidget由两部分组成，在初始化时必须要在createState()时初始化一个与之相关的State对象

8 Flutter 和 Dart的关系是什么?

Flutter是一个使用Dart语言开发的跨平台移动UI框架，通过自建绘制引擎，能提高性能、高保真地进行移动开发。Dart囊括了多数编程语言的优点，它

更符合Flutter构建界面的方式

9 mixin extends implement 之间的关系?

继承（关键字 extends）、混入 mixins （关键字 with）、接口实现（关键字 implements）。这三者可以同时存在，前后顺序是 extends -> mixins -> implements。

Flutter 中的继承是单继承，子类重写超类的方法要用 @Override，子类调用超类的方法要用 super。

在 Flutter 中，Mixins 是一种在多个类层次结构中复用类代码的方法。mixins 的对象是类，mixins 绝不是继承，也不是接口，而是一种全新的特性，可以 mixins 多个类，mixins 的使用需要满足一定条件。

10 什么是 Navigator? MaterialApp 做了什么?

Navigator 是在 Flutter 中负责管理维护页面堆栈的导航器。

MaterialApp 在需要的时候，会自动为我们创建 Navigator。

Navigator.of(context)，会使用 context 来向上遍历 Element 树，找到 MaterialApp 提供的_NavigatorState 再调用其 push/pop 方法完成导航操作。

11 Widget 唯一标识 Key 有那几种?

在 flutter 中，每个 widget 都是被唯一标识的。这个唯一标识在 build 或 rendering 阶段由框架定义。该标识对应于可选的 Key 参数，如果省略，Flutter 将会自动生成一个。

在 flutter 中，主要有 4 种类型的 Key：GlobalKey（确保生成的 Key 在整个应用中唯一，是很昂贵的，允许 element 在树周围移动或变更父节点而不会丢失状态）、LocalKey、UniqueKey、ObjectKey。

12 Future 和 Isolate 有什么区别?

future 是异步编程，调用本身立即返回，并在稍后的某个时候执行完成时再获得返回结果。在普通代码中可以使用 await 等待一个异步调用结束。

isolate 是并发编程，Dartm 有并发时的共享状态，所有 Dart 代码都在 isolate 中运行，包括最初的 main()。每个 isolate 都有它自己的堆内存，意味着其中所有内存数据，包括全局数据，都仅对该 isolate 可见，它们之间的通信只能通过传递消息的机制完成，消息则通过端口 (port) 收发。isolate 只是一个概念，具体取决于如何实现，比如在 Dart VM 中一个 isolate 可能会是一个线程，在 Web 中可能会是一个 Web Worker。

13 Stream 与 Future 是什么关系?

Stream 和 Future 是 Dart 异步处理的核心 API。Future 表示稍后获得的一个数据，所有异步的操作的返回值都用 Future 来表示。但是 Future 只能表示一次异步获得的数据。而 Stream 表示多次异步获得的数据。比如界面上的按钮可能会被用户点击多次，所以按钮上的点击事件（onClick）就是一个 Stream 。简单地说，Future 将返回一个值，而 Stream 将返回多次值。Dart 中统一使用 Stream 处理异步事件流。Stream 和一般的集合类似，都是一组数据，只不过一个是异步推送，一个是同步拉取。

14 await for 如何使用?

await for 是不断获取 stream 流中的数据，然后执行循环体中的操作。它一般用在直到 stream 什么时候完成，并且必须等待传递完成之后才能使用，不然就会一直阻塞。

15 Flutter 的架构主要分成三层:Framework，Engine 和 Embedder。

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。
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。其已作为Google Chrome，Chrome OS，Android, Mozilla Firefox, Firefox OS等其他众多产品的图形引擎，支持平台还包括Windows7+,macOS 10.10.5+,iOS8+,Android4.1+,Ubuntu14.04+等。
Dart部分主要包括:Dart Runtime，Garbage Collection(GC)，如果是Debug模式的话，还包括JIT(Just In Time)支持。Release和Profile模式下，是AOT(Ahead Of Time)编译成了原生的arm代码，并不存在JIT部分。Text即文本渲染，其渲染层次如下：衍生自minikin的libtxt库(用于字体选择，分隔行)。HartBuzz用于字形选择和成型。Skia作为渲染/GPU后端，在Android和Fuchsia上使用FreeType渲染，在iOS上使用CoreGraphics来渲染字体。
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。

16 Flutter 如何与 Android iOS 通信?

Flutter与原生系统主要有三种通信形式：MethodChannel和EventChannel及BasicMessageChannel。

1 BasicMessageChannel：用于传递字符串和半结构化的信息，持续通信，收到消息后可以回复此次消息，如：Native将遍历到的文件信息陆续传递到Dart，在比如：Flutter将从服务端陆续获取到信息交给Native加工，Native处理完返回等；

2 MethodChannel：用于传递方法调用。（method invocation）一次性通信：如Flutter调用Native拍照；

3 EventChannel: 用于数据流（event streams）的通信，持续通信，收到消息后无法回复此次消息，通常用于Native向Dart的通信，如：手机电量变化，网络连接变化，陀螺仪，传感器等；

这三种类型的类型的Channel都是全双工通信，即A <=> B，Dart可以主动发送消息给platform端，并且platform接收到消息后可以做出回应，同样，platform端可以主动发送消息给Dart端，dart端接收数后返回给platform端。

17 Flutter中的Widget、State、Context的核心概念？是为了解决什么问题?

1 Widget: 在Flutter中，几乎所有东西都是Widget。将一个

Widget想象为一个可视化的组件（或与应用可视化方面交互的组件），当你需要构建与布局直接或间接相关的任何内容时，你正在使用 Widget。
2 Widget树: Widget以树结构进行组织。包含其他 Widget的widget被称为父 Widget(或 widget容器)。包含在父 widget中的widget被称为子 Widget。
3 Context: 仅仅是已创建的所有 Widget树结构中的某个 Widget的位置引用。简而言之，将context作为 widget树的一部分，其中context所对应的widget被添加到此树中。一个context只从属于一个 widget，它和 widget一样是链接在一起的，并且会形成一个context树。
4 State: 定义了 StatefulWidget实例的行为，它包含了用于"交互/干预"Widget信息的行为和布局。应用于 State的任何更改都会强制重建 Widget。
5 这些状态的引入，主要是为了解决多个部件之间的交互和部件自身状态的维护。

18 ****Dart****部分**

19 Dart是值传递还是引用传递

dart是**值传递**。

20 Dart 当中的 「..」表示什么意思?

Dart 当中的 「..」意思是 「级联操作符」，为了方便配置而

使用。
「..」和「.」不同的是 调用「..」后返回的相当于是 this，而「.」返回的则是该方法返回的值 。

21 Dart 的作用域

Dart 没有 「public」「private」等关键字，默认就是公开的，私有变量使用 下划线 _开头

22 Dart 是不是单线程模型？是如何运行的?

Dart 是单线程模型，如何运行的看这张图：
[图片上传失败...(image-cff6d3-1598350597924)]
：
Dart 在单线程中是以消息循环机制来运行的，其中包含两个任务队列，一个是“微任务队列” microtask queue，另一个叫做“事件队列” event queue。
入口函数 main() 执行完后，消息循环机制便启动了。首先会按照先进先出的顺序逐个执行微任务队列中的任务，当所有微任务队列执行完后便开始执行事件队列中的任务，事件任务执行完毕后再去执行微任务，如此循环往复，生生不息。

23 final 和 const 区别

const 的值在编译期确定，final 的值要到运行时才确定

24 说一下Dart异步编程中的Stream数据流?

在Dart中，Stream 和 Future 一样，都是用来处理异步编程的工具。它们的区别在于，Stream 可以接收多个异步结果，而Future 只有一个。Stream 的创建可以使用 Stream.fromFuture，也可以使用 StreamController 来创建和控制。还有一个注意点是：普通的Stream 只可以有一个订阅者，如果想要多订阅的话，要使用 asBroadcastStream()。

25 App 打包上传

在flutter项目中分别打开 Android和IOS 文件，和原生开发方式的打包上架形式是一样的。

命令打包

cd 到文件目录

iOS：Flutter Build ios –-release

Android： flutter build apk –release

26 启动页和图标，添加方式和iOS，android原有的方式一样。