# 1.2 初识Flutter

## 1.2.1 Flutter简介

Flutter 是 Google推出并开源的移动应用开发框架，主打跨平台、高保真、高性能。开发者可以通过 Dart语言开发 App，一套代码同时运行在 iOS 和 Android平台。 Flutter提供了丰富的组件、接口，开发者可以很快地为 Flutter添加 native扩展。同时 Flutter还使用 Native引擎渲染视图，这无疑能为用户提供良好的体验。

#### 跨平台自绘引擎

Flutter与用于构建移动应用程序的其它大多数框架不同，因为Flutter既不使用WebView，也不使用操作系统的原生控件。 相反，Flutter使用自己的高性能渲染引擎来绘制widget。这样不仅可以保证在Android和iOS上UI的一致性，而且也可以避免对原生控件依赖而带来的限制及高昂的维护成本。

Flutter使用Skia作为其2D渲染引擎，Skia是Google的一个2D图形处理函数库，包含字型、坐标转换，以及点阵图都有高效能且简洁的表现，Skia是跨平台的，并提供了非常友好的API，目前Google Chrome浏览器和Android均采用Skia作为其绘图引擎。

目前Flutter默认支持iOS、Android、Fuchsia（Google新的自研操作系统）三个移动平台。但Flutter亦可支持Web开发（Flutter for web）和PC开发，本书的示例和介绍主要是基于iOS和Android平台的，其它平台读者可以自行了解。

#### 高性能

Flutter高性能主要靠两点来保证，首先，Flutter APP采用Dart语言开发。Dart在 JIT（即时编译）模式下，速度与 JavaScript基本持平。但是 Dart支持 AOT，当以 AOT模式运行时，JavaScript便远远追不上了。速度的提升对高帧率下的视图数据计算很有帮助。其次，Flutter使用自己的渲染引擎来绘制UI，布局数据等由Dart语言直接控制，所以在布局过程中不需要像RN那样要在JavaScript和Native之间通信，这在一些滑动和拖动的场景下具有明显优势，因为在滑动和拖动过程往往都会引起布局发生变化，所以JavaScript需要和Native之间不停的同步布局信息，这和在浏览器中要JavaScript频繁操作DOM所带来的问题是相同的，都会带来比较可观的性能开销。

#### 采用Dart语言开发

这是一个很有意思，但也很有争议的问题，在了解Flutter为什么选择了 Dart而不是 JavaScript之前我们先来介绍两个概念：JIT和AOT。

目前，程序主要有两种运行方式：静态编译与动态解释。静态编译的程序在执行前全部被翻译为机器码，通常将这种类型称为**AOT** （Ahead of time）即 “提前编译”；而解释执行的则是一句一句边翻译边运行，通常将这种类型称为**JIT**（Just-in-time）即“即时编译”。AOT程序的典型代表是用C/C++开发的应用，它们必须在执行前编译成机器码，而JIT的代表则非常多，如JavaScript、python等，事实上，所有脚本语言都支持JIT模式。但需要注意的是JIT和AOT指的是程序运行方式，和编程语言并非强关联的，有些语言既可以以JIT方式运行也可以以AOT方式运行，如Java、Python，它们可以在第一次执行时编译成中间字节码、然后在之后执行时可以直接执行字节码，也许有人会说，中间字节码并非机器码，在程序执行时仍然需要动态将字节码转为机器码，是的，这没有错，不过通常我们区分是否为AOT的标准就是看代码在执行之前是否需要编译，只要需要编译，无论其编译产物是字节码还是机器码，都属于AOT。在此，读者不必纠结于概念，概念就是为了传达精神而发明的，只要读者能够理解其原理即可，得其神忘其形。

现在我们看看Flutter为什么选择Dart语言？笔者根据官方解释以及自己对Flutter的理解总结了以下几条（由于其它跨平台框架都将JavaScript作为其开发语言，所以主要将Dart和JavaScript做一个对比）：

1. **开发效率高**

* Dart运行时和编译器支持Flutter的两个关键特性的组合：
* **基于JIT的快速开发周期**：Flutter在开发阶段采用，采用JIT模式，这样就避免了每次改动都要进行编译，极大的节省了开发时间；
* **基于AOT的发布包**: Flutter在发布时可以通过AOT生成高效的ARM代码以保证应用性能。而JavaScript则不具有这个能力。

1. **高性能**

* Flutter旨在提供流畅、高保真的的UI体验。为了实现这一点，Flutter中需要能够在每个动画帧中运行大量的代码。这意味着需要一种既能提供高性能的语言，而不会出现会丢帧的周期性暂停，而Dart支持AOT，在这一点上可以做的比JavaScript更好。

1. **快速内存分配**

* Flutter框架使用函数式流，这使得它在很大程度上依赖于底层的内存分配器。因此，拥有一个能够有效地处理琐碎任务的内存分配器将显得十分重要，在缺乏此功能的语言中，Flutter将无法有效地工作。当然Chrome V8的JavaScript引擎在内存分配上也已经做的很好，事实上Dart开发团队的很多成员都是来自Chrome团队的，所以在内存分配上Dart并不能作为超越JavaScript的优势，而对于Flutter来说，它需要这样的特性，而Dart也正好满足而已。

1. **类型安全**

* 由于Dart是类型安全的语言，支持静态类型检测，所以可以在编译前发现一些类型的错误，并排除潜在问题，这一点对于前端开发者来说可能会更具有吸引力。与之不同的，JavaScript是一个弱类型语言，也因此前端社区出现了很多给JavaScript代码添加静态类型检测的扩展语言和工具，如：微软的TypeScript以及Facebook的Flow。相比之下，Dart本身就支持静态类型，这是它的一个重要优势。

1. **Dart团队就在你身边**

* 看似不起眼，实则举足轻重。由于有Dart团队的积极投入，Flutter团队可以获得更多、更方便的支持，正如Flutter官网所述“我们正与Dart社区进行密切合作，以改进Dart在Flutter中的使用。例如，当我们最初采用Dart时，该语言并没有提供生成原生二进制文件的工具链（这对于实现可预测的高性能具有很大的帮助），但是现在它实现了，因为Dart团队专门为Flutter构建了它。同样，Dart VM之前已经针对吞吐量进行了优化，但团队现在正在优化VM的延迟时间，这对于Flutter的工作负载更为重要。”

#### 总结

本节主要介绍了一下Flutter的特点，如果你感到有些点还不是很好理解，不用着急，随着日后对Flutter细节的了解，再回过头来看，相信你会有更深的体会。

## 1.2.2 Flutter框架结构

本节我们先对Flutter的框架做一个整体介绍，旨在让读者心中有一个整体的印象，这对初学者来说非常重要。如果一下子便深入到Flutter中，就会像是一个在沙漠中没有地图的人，即使可以找到一个绿洲，但是他也不会知道下一个绿洲在哪。因此，无论学什么技术，都要现有一张清晰的“地图”，而我们的学习过程就是“按图索骥”，这样我们才不会陷于细节而“目无全牛”。言归正传，我们看一下Flutter官方提供的Flutter框架图，如图1-1所示：

图1-1

### Flutter Framework

这是一个纯 Dart实现的 SDK，它实现了一套基础库，自底向上，我们来简单介绍一下：

* 底下两层（Foundation和Animation、Painting、Gestures）在Google的一些视频中被合并为一个dart UI层，对应的是Flutter中的dart:ui包，它是Flutter引擎暴露的底层UI库，提供动画、手势及绘制能力。
* Rendering层，这一层是一个抽象的布局层，它依赖于dart UI层，Rendering层会构建一个UI树，当UI树有变化时，会计算出有变化的部分，然后更新UI树，最终将UI树绘制到屏幕上，这个过程类似于React中的虚拟DOM。Rendering层可以说是Flutter UI框架最核心的部分，它除了确定每个UI元素的位置、大小之外还要进行坐标变换、绘制(调用底层dart:ui)。
* Widgets层是Flutter提供的的一套基础组件库，在基础组件库之上，Flutter还提供了 Material 和Cupertino两种视觉风格的组件库。而**我们Flutter开发的大多数场景，只是和这两层打交道**。

### Flutter Engine

这是一个纯 C++实现的 SDK，其中包括了 Skia引擎、Dart运行时、文字排版引擎等。在代码调用 dart:ui库时，调用最终会走到Engine层，然后实现真正的绘制逻辑。

### 总结

Flutter框架本身有着良好的分层设计，本节旨在让读者对Flutter整体框架有个大概的印象，相信到现在为止，读者已经对Flutter有一个初始印象，在我们正式动手之前，我们还需要了解一下Flutter的开发语言Dart。

## 1.2.3 如何学习Flutter

本节给大家一些学习建议，分享一下笔者在学习Flutter中的一些心得，希望可以帮助你提高学习效率，避免不必要的坑。

### 资源

* **官网**：阅读Flutter官网的资源是快速入门的最佳方式，同时官网也是了解最新Flutter发展动态的地方，由于目前Flutter仍然处于快速发展阶段，所以建议读者还是时不时的去官网看看有没有新的动态。
* **源码及注释**：源码注释应作为学习Flutter的第一文档，Flutter SDK的源码是开源的，并且注释非常详细，也有很多示例，实际上，Flutter官方的SDK文档就是通过注释生成的。源码结合注释可以帮你解决大多数问题。
* **Github**：如果遇到的问题在StackOverflow上也没有找到答案，可以去github flutter 项目下提issue。
* **Gallery源码**：Gallery是Flutter官方示例APP，里面有丰富的示例，读者可以在网上下载安装。Gallery的源码在Flutter源码“examples”目录下。

### 社区

* **StackOverflow**：如果你还没听过StackOverflow，这是目前全球最大的程序员问答社区，现在也是活跃度最高的Flutter问答社区。StackOverflow上面除了世界各地的Flutter使用者会在上面交流之外，Flutter开发团队的成员也经常会在上面回答问题。
* **Flutter中文网社区**：Flutter中文网(https://flutterchina.club)是笔者维护中文网站，目前也是最大的中文资源社区，上面提供了Flutter官网的文档翻译、开源项目、及案例，还有申请加入组织的入口哦。
* **博客**：随着Flutter技术的推广，相信很快网上将会有很多Flutter相关的文章、博客，读者可以多去浏览、阅读。

### 总结

有了资料和社区后，对于我们学习者自身来说，最重要的还是要多动手、多实践，在本书后面的章节中，希望读者能够亲自动手写一下示例。准备好了吗，下一章中，我们将正式进入Flutter的世界！