

## 开篇词 | 锚定一个点，然后在这个点上深耕

笔记本： iOS开发高手课

创建时间： 2019/3/11 19:54

更新时间： 2019/3/11 20:01

作者： 8193514@qq.com

URL： <https://time.geekbang.org/column/article/0?cid=161>

---

## 开篇词 | 锚定一个点，然后在这个点上深耕

戴铭 2019-03-11



你好，我是戴铭，欢迎你加入我的 iOS 开发专栏。

说起 iOS 开发，自然是绕不开 iPhone 和 App Store 这两个词。多少年过去了，我依然记得 2007 年乔布斯发布第一代 iPhone 时的场景，可以说，那款 iPhone 重新定义了很多人对手机的认知。那一天，也是移动互联网时代的开端。紧接着，在 2008 年 7 月的 WWDC 苹果全球开发者大会上，苹果宣布 App Store 正式对外开放，这也意味着属于开发者的移动互联网时代真正开始了。

一转眼，十多年过去了，移动开发早已飞入寻常百姓家，移动 App 基本成为了大众互联网产品的标配。从技术角度看，这些年来移动开发领域相继诞生了组件化、热修复、动态化、大前端、Flutter、小程序等热门技术，给人一种“乱花渐欲迷人眼”的感觉，它们争先恐后地成为行业焦点，然后又又在不知不觉中被后来者替代，最后再逐步淡出开发者的视野。

**在我看来，这些技术看似过眼云烟，实则是递进式推动着移动技术的演进。**

这些技术看似“纷繁杂乱”，实则是殊途同归，它们背后都是一些通用的底层技术和创新的设计思想。比如，热修复、动态化和大前端的底层技术，都是 JavaScriptCore 这样的 JavaScript 虚拟机技术；再比如，大前端和 Flutter 的渲染，使用的都是 WebCore 中 Skia 这样的渲染引擎。所以，每当我串起来整体看移动开发领域这些年的这些“新”技术时，总是会感慨说“万变不离其宗”。我就觉得如果我们能深入进去把某一门技术研究透彻，那再拿下其他的技术时就会变得轻车熟路。

**以组件化为例**，它是顺应着 App 从单一业务到多业务汇聚的演进而出现的一门技术。比如微信刚发布时业务单一，就只有聊天的功能，后来又加上了支付、朋友圈、游戏，再等到小程序功能上线后更是打车、电影票、购物等只要你能想到的需求它都有，俨然成为了一个超级平台。

从本质上讲，组件化是将上层业务隔离开，下层提供通用能力的一种架构模式，这样上层业务团队可以分开从而减少团队沟通成本，下层能力的通用性又反过来提高了各个业务团队的开发效率。为了达到不同业务隔离的结果，解耦手段不断被引入到 iOS 开发中，比如使用协议或者中间者模式在运行时统调等方式。

听到这里，你还会觉得组件化技术陌生吗？并不陌生，它的核心思想就是解耦。只要你把这块研究透彻了，那再理解与解耦相关的其他技术架构也就水到渠成了。

**我们再以热修复为例**，如果你看过相关的热修复技术源代码或者架构图的话，应该不难发现热修复技术的核心引擎主要就是 JavaScriptCore，它要求原生开发者使用 JavaScript 来编写代码。而为了方便原生开发者，热修复引擎最大卖点就是将原生 Objective-C 代码转成 JavaScript 代码然后让 JavaScriptCore 去解释。这样一个代码转换过程其实就是通过编译技术来实现的。

同时，在目前正流行的这波大前端和小程序浪潮中，各种大前端技术都对 Web 标准做了大量限制，定义了自己的规范模板。这些自定义的模板最终都会使用 Babel 这样的前端编译技术将其编译成 JavaScript 代码，然后再交给 JavaScriptCore 解释调用原生渲染。

所以，你看，只要掌握了热修复中的核心技术，就不难理解小程序的[实现](#)原理。一切看起来就是那么自然。

最近圈子里又开始流行 Flutter 了，在 Flutter 这波浪潮还没有全面落地铺开前，很多人就开始关心：下一个热点会是什么？其实我觉得大可不必在乎，你只要静下心来好好消化掉这几年浪潮留下的关键技术，在这个基础上再去理解各种“新技术”，必然会驾轻就熟。

最后，再说个你最能切身体会的企业招聘对于 iOS 开发者的要求。以前对工作经历只要求有过完整独立完成的 App 上架就够了，而现在如果你缺少大型项目经验，团队规模小，没有好的提质增效开发经验，在应聘时的竞争力会大打折扣。

**但，这并不是 iOS 领域的个体问题，任何一个领域其实都和移动领域一样，从小型到大型，从个人到团队，从低效到高效，从凑够用到高要求。**

比如说，后端开发领域伴随着互联网的发展，也有过同样的经历。你会发现，在一些公司从后端晋升到更高级别的开发者会更多些。这就是因为后端开发领域很早就从小规模开发转变成了大规模开发，在这个过程中已经整体经过了大量的演进，对于开发人员的要求也在逐步提高，特别是对系统架构的稳定和灵活设计能力的要求，还有对工程质量和规范效率方面的高要求。

我有幸深度经历了移动技术和后端技术的演进过程，并在工作中进行了深度的调研和研究，最终将成果落地到各个项目中。在 iOS 技术发展的过程中，我的知识也得到了递进式提升，也最终被运用到了实际工作中，比如组件化方案落地、大前端建设、应用开发阶段效率的提升、上线后各种难点问题的解决等。

我热爱分享，喜欢将平时学习和工作中的经验分享到[我的博客](#)和微博上，也会将一些技术总结通过代码发到 [我的 GitHub](#)上。

在这个专栏里，我会针对移动开发这些年演进过程中沉淀下来的那些技术，那些支撑着 iOS 迈向更稳健、更成熟的技术进行详细而系统的输出。同时，我也会提出自己的一些思考，包括对于各种技术后面发展的方向和可能性的想法。

**接下来，我跟你说下专栏大致内容构成和写作思路。**

移动开发面对的也是计算设备，和后端一样也要监控和解决设备的内存和线程等性能问题，编程的本质就是要解决问题，无论是需求、开发、调试、线上问题都需要编程来解决，而代码是开发者的唯一武器。

所以，在**第一部分的基础篇**，我会围绕着如何解决 iOS 开发各个阶段的问题展开。这是编程的基础，没有这个基础其他都免谈。同时，解决问题的扎实程度，也决定了你在面试中的竞争力。

iOS 开发者更多的是面向用户界面和交互的开发，而在界面、交互以及数据通信处理过程中存在大量的重复工作，因此我会**在第二部分的应用开发篇里**，给你推荐一些经典好用的第三方库。

用好这些库，能够帮助你大幅提高应用开发的效率。同时，我也会带你去探究这些优秀库的背后原理和**实现**思路，当你面对千奇百怪的需求时，也能够开发出适合特定需求的库。说不定下一个经典的第三方库就是由你开发的呢？

如果你希望自己能在技术能力和职级上得到晋升，在碰到问题时不再被动地见招拆招，而是按照自己的套路主动出击化险为夷，那么对于底层原理的深入学习就非常必要了。在这个过程中，你还能学到前辈解决问题的思路，这将让你收获颇丰。

所以，在**第三部分的原理篇**，我会专门针对一些底层原理，比如 XNU、内存管理、编译等进行分享，期待能够激发起你的学习兴趣，让你内力大增。

所谓良药苦口，底层知识的最初学习过程一定是辛苦的，只有目标和意志非常坚定的那群人才能坚持下来。但是，当你利用这些知识造出更好的轮子时，那种成就是你在舒适区获得的愉悦感无法比拟的。

iOS 开发技术的演进和前端是齐头并进的，前端从开始的 H5 Hybrid 容器“陪跑姿态”转变为以 React Native 这样的技术为支撑的“助跑角色”，还有 Flutter 这种新原生技术期待能够主导 iOS 的开发，最后小程序这种产品形态主打生态牌，而技术上返璞归真采用 Hybrid 技术又将前端技术重新拉回舞台。

面对现在这种原生与前端共舞的情景，我会**在第四部分**帮你拆解各种技术细节，以及它们之间的内在联系，以便帮助你站在更高的视角去判断未来的技术走向和趋势。晋升到更高位置的你，对未来技术走向的判断将尤为重要。

**最后，希望你能认真动手完成每篇文章后面的课后作业。**

对于咱们手艺人来说，不动手都是空谈，就像绘画教程，光看不练，是不会有进步的。这就如同九阴真经的口诀，铭记于心后还需要常年累月的修炼才能精进。动手就会碰到问题，就会思考，这个主动过程会加深你的记忆，这样后面再碰到问题时，你会更容易将相关知识串联起来，形成创新式的思考。

好了，今天的内容就到这里，如果可以的话，还请你在留言区中做个自我介绍，和我聊聊你目前的工作、学习情况，以及你在学习 iOS 开发时的痛点，这样我们可以彼此了解，也方便我在后面针对性地给你讲解。

加油，让我们一起静下心，沉到具体的技术里，潜心研究。