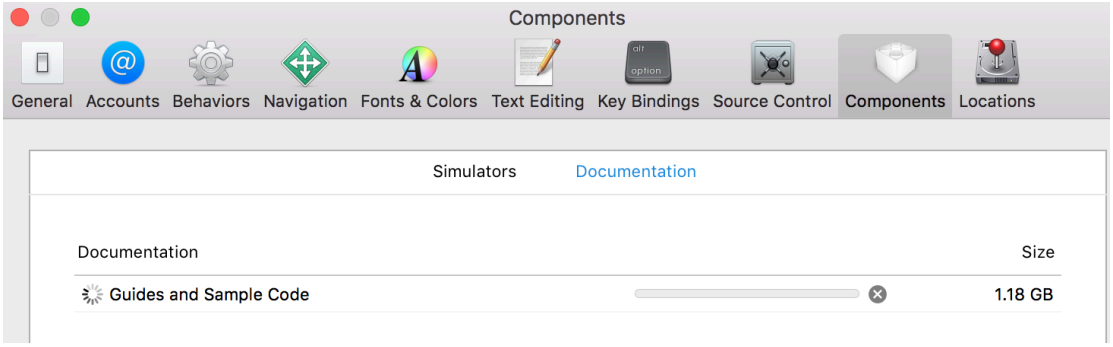
Apple

官网文档阅读指南

第一篇章　阅读途径

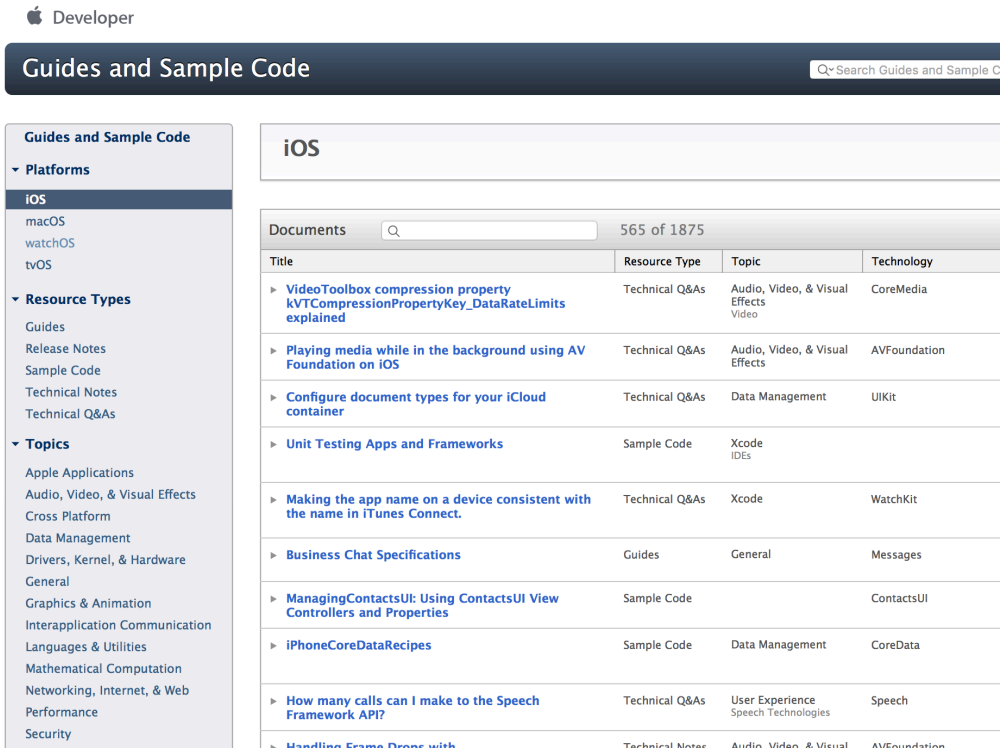
1、　在 Xcode 里【Command+,】出现 preferences，然后选中 Components 下的 Documentation 进行下载相关的官方文档。



2、　在 Xcode 里【Command+shift+0】documentation and API references 中可以打开官方文档

3、　官方文档网址：https://developer.apple.com/library/content/navigation/#section=Resource%20Types&topic=Guides

第二篇章 文档结构



介绍：

1.文档左侧的侧栏可以选择不同的文档库，可以选择 resource type、topic、和 framework

2.文档库子目录下的是文档的结构，里面包含我们浏览的所有信息内容的目录

3.右边的是目录下的文件名，点进去可以看到具体的文件信息

（1）Resource Types

先看 Resource Types，也就是资源类型，下面是这个类型的子目录介绍。

1．Guides——指南，指南就是一个一个问题的，从一个问题，一步一步的详细的介绍怎么使用 cocoa 库的文档，它能够帮助你梳理学习的脉络。

2. Release Notes——发布说明，说明 iOS 新版本带来的特性，就是相比于旧的 iOS 新出现的特性。

3.Sample code——示例代码，苹果官方提供用来帮助你学习某些技术的 API；建议在学习的时候使用，可以提高对程序实现的理解；同时这是苹果的工程师写的代码，从中你可以体会到苹果官方推荐的代码风格。

4.Technical Notes——技术说明 ， 一些技术主题文章，有空的时候可以浏览一下，可以提高对苹果技术的了解。

5.Technical Q&A——常见技术问答 ， 这是技术社区里面一些常见问题以及解答的整理

总结一下，这里的 **Guides** 是非看不可的，其他那些是用来备查的。

（2）Topics

我列举几个：

1、 Data Management ----- 数据管理

2、 General ----- 一般性的问题

3、 Graphics & Animation ----- 图形和动画

4、 Interapplication Communication-----应用间程序通信

5、 Language & Utilities ----- 语言和工具

6、 Mathematical Computation ----- 数学计算

7、 Networking，Internet &Web-----网络问题

8、 Performance ----- 性能

9、 Security -----安全

10、 User Experience ----- 用户体验

11、 Xcode ----- 介绍xcode的使用

（3）Technologies

学习的一些技术

第三篇章 文档入门导读

这里介绍的是你如何一步步的阅读开发文档，慢慢深入学习 iOS 编程

阅读顺序：

1.《start developing iOS apps today》

2.《your first iOS app》

3.《your second iOS app: storyboard》

4.《your third iOS app: icloud》

5.《iOS technology overview》

6.《iOS human interface guidelines》

7.《learning Objectvive-c :a primer》和《programing with objective-c》

8.《iOS app programming guide》

9.《view programming guide for iOS》 和《view controller programming guide for iOS》

10.《table view programming guide for iOS》

首先应该看的是 Getting Started 里面的《马上着手开发 iOS 应用程序 (Start Developing iOS Apps Today)》（中英文版本皆有，苹果官方的翻译）。这个文档讲的很浅，但是是建立概念的文档，你以后在开发里面经常遇到的概念，在这里都有包含，特别注意是，这个文档看起来简单，但是每页下面的相关文章，不是选读，都是必读。

即使是很多做了 iOS 开发很久的同学，其实也有很多概念上的误解，现代程序开发越来越简单，工具越来越强大，往往有些误解也可以继续前行，但是实际上不建立扎实的基础是很吃亏的，往往后面理解和解决一个不难解决小问题都要付出很多辛苦。

阅读这个文档的目的和检测标准是，以后你看到 iOS 开发中的基本概念，都大致所有了解。

读完《马上着手开发 iOS 应用程序 (Start Developing iOS Apps Today)》后，应该去看 Your XXX iOS App 系列这个系列不是什么很难的文章，你也不必着急先去学习 Objective-C，学什么 C 语言就更不要着急。我推荐的学习方法是有成就的逐步学习法。在学习系统体系架构、Objective-C 之前，你可以先按照文档写一个全天下最简单的 App，完成学习过程中第一个里程碑。在这个过程中不用担心有什么疑问，有什么不懂，先照着做就是。

阅读这三个文档的目的和检测标准是，把这三个 Demo App 做出来在模拟器上跑起来。

在这个过程中，你对开发工具的基本认识就建立起来了，也有了成就感，去了魅（就是消除了对 iOS 开发的神秘感）。

再往下，建议你去看《iOS Technology Overview》（iOS 技术概览），iOS 不是一个技术，而是一堆技术，前一篇讲到文档导航区域的分类，框架分类的时候，我说不细讲的原因就在于此，你要做一个动画应该用 Core Animation 还是 OpenGL？你要做一些文本相关操作应该用 Core Text 还是什么，就是看这里。

学习现代的程序开发，语言和框架并重。我们 Tiny4Cocoa 叫做这个名字的原因就是，iOS/Mac 开发者的代表往往就是这个 Cocoa 框架，就是这个 SDK。大多数你所需要的功能都躺在框架里面，你知道框架的结构，你才知道怎么去寻找相关的技术资料。

阅读这个文档的目的和检测标准是，遇到具体问题，知道应该去看哪方面的文档。

再下来，建议阅读的是《iOS Human Interface Guidelines》，Mac/iOS 平台虽然也是百花齐放各类程序、App 都有，但是总体看来，大多数优秀 App 的 UI 看起来都和整个系统很协调。这和 Windows 以及很多其他平台完全不同。这是为什么呢？

很大程度就归功于《Human Interface Guidelines》文化，所谓 Human Interface Guidelines 就是用户界面的规范，在苹果它还专门有一个缩写叫做 HIG，是天条一样的东西。所有的 App 都推荐遵循 HIG，遵循了 HIG，你做的东西用户看起来就会觉得和整个系统很协调。即使是你要做一些创新的设计，你势必会打破 HIG 的限制，但是你这个时候仍旧应该遵循 HIG 的精神。

此外，你阅读 HIG 的很重要一点是了解整个 UI 结构和 UE 行为的逻辑机理，这样才能在你设计界面的时候有所依据。

阅读这个文档的目的和检测标准是，看到任何一个 App，你可以知道它的任何一个 UI 是系统控件，还是自定义控件，它的层次关系等等。

《Learning Objective-C: A Primer》是非常初级和简单的入门，适合先阅读。《Programming with Objective-C》超微复杂一点点，适合后阅读。

一般人建议先学习语言，我反之建议先做了一个 App，然后再学习语言。原因有几个，首先现代开发工具，往往不是用来开发控制台程序的，上来就会有框架，光懂语言不会使用 IDE，甚至可能会更麻烦。再其次就是，其实现代语言发展到了面向对象以后，库往往比语言更复杂，更重要，或者说更多的时候，我们是在学习库，而不是语言，语言只是库的一个载体。

比如，Delegate 和 Block 等等都和 Cocoa 的 UI 异步机制关系紧密，光看代码，这些语言元素非常难以学习，也完全不知道其意义在哪里。

阅读这个文档的目的和检测标准是，看得懂基本的 Objective-C 代码，方便后面的学习和阅读各种示例代码。

《iOS App Programming Guide》基本上介绍的就是开发一个 App 的完整流程，包括 App 的生命周期、休眠、激活等等，里面介绍的细节颇多。正式开发第一个上线的 App 之前必看。或者开发了一个 App，临到提交前必看才文档。

阅读这个文档的目的和检测标准是，了解全部流程和很多细节问题。

《View Programming Guide for iOS》和《View Controller Programming Guide for iOS》非常重要。View 是整个图形界面里面最重要的概念。所有的图形、界面绘制都基于 View。你看到的一切复杂的用户界面，就是各种不同的 View 的组合堆叠。

View Controller 是 View 和某种逻辑在一起的组合，本质上这种组合不是必须的，但是是大大降低编程复杂度的一种设计。很多人不懂什么是 View 什么是 View Controller，这样写起代码来就很痛苦。

阅读这个文档的目的和检测标准是，深刻理解什么是 View，什么是 View Controller，理解什么情况用 View，什么情况用 View Controller。

UITableView 是最重要的一个控件，是最常用的 UI 界面元素。在 UICollectionView 出现之前，大量的内容列表展示的自定义控件都是基于 UITableView，比如很多书架、图片 Grid 其实都是 UITableView 做的。

所以《Table View Programming Guide for iOS》非常重要，一定要好好阅读。

阅读这个文档的目的和检测标准是，深刻理解 UITableView / UITableViewController 的理论和使用方法。

第四篇章 总结篇

我推荐的必读文档就这么多，仔细看的话，最多也就是今天就看完了。学习一个东西，如果有一点点耐心，有正确的方法其实不难，不是说脑子非要很聪明，大多数人都可以做到一个星期就学会 iOS 开发，其实就是读完这些文档，大多数人就会了。

就像我强调了无数次，阅读英文文档不难，我自己从当年看英文文档非常吃力，必须查词典开始，认真的看英文文档，不会就查词典，一个多月过去，读英文文档就完全不需要查词典了。

其实学习 iOS 也如此。当然我不是说你看懂这 10 组文档就再也不用看别的了。而是说，如果你看懂了这 10 组文档，你就从初学者，或者是虽然会写一些程序，但是对 iOS 其实还不懂的状态，变成了一个入门者。

我不希望这个文章可以一句一句的帮你学会 iOS 是什么，这个文章的目的是帮你快速入门。一旦你入门了，你再遇到问题该看什么，你就不需要我讲了，你自己就知道了。一旦入门了，你就会发现，Xcode 里面别的文档讲的内容虽然不同，但是结构你已经很清楚了，你学习起来很方便。

阅读本文的目的和检测标准是，遇到问题，知道看什么文档，想提升自己技术的时候，知道按照什么脉络自己组织阅读。