**iOS自动化测试**

**iOS官方自动化测试框架**

**XCTest**

这个没什么好说的，苹果在iOS7引入的测试框架，主要用于接口测试

* 优点：与Xcode深度集成，OC语言编写，易上手
* 缺点：受限于官方的测试API，功能不是特别丰富

**UIAutomation**

苹果提供的UI自动化测试框架，使用JavaScript编写。基于UIAutomation有扩展型的工具框架和驱动型的框架。扩展型框架以JavaScript扩展库方法提供了很多好用js工具，注入式的框架通常会提供一些Lib或者是Framework，要求测试人员在待测应用的代码工程中导入这些内容，框架可以通过他们完成对app的驱动。驱动型UI Automation 在自动化测试底层使用了UI Automation库，通过TCP通信的方式驱动UI Automation来完成自动化测试，通过这种方式，编辑脚本的语言不再局限于JavaScript（比如Appium）

* 优点：原生支持，可用模拟器
* 缺点：JS编写，上手成本较高，

**UITesting**

适用于iOS9.0及以上的设备，和XCTest类似

* 优点：原生支持，不用配置环境，可用Swift，OC，上手简单，支持Web View测试
* 缺点：录制的插入代码不准确，中文支持不佳

**基于UI Automation第三方框架**

很多第三方框架的测试是魔改官方的UI Automation的，可以分为扩展型、驱动型、私有API型和注入编译框架型。

**扩展型框架**

扩展型 UI Automation采用JavaScript扩展库的方法提高UI Automation的易用性，常用的框架有TuneupJS, ynm3k等

**驱动型框架**

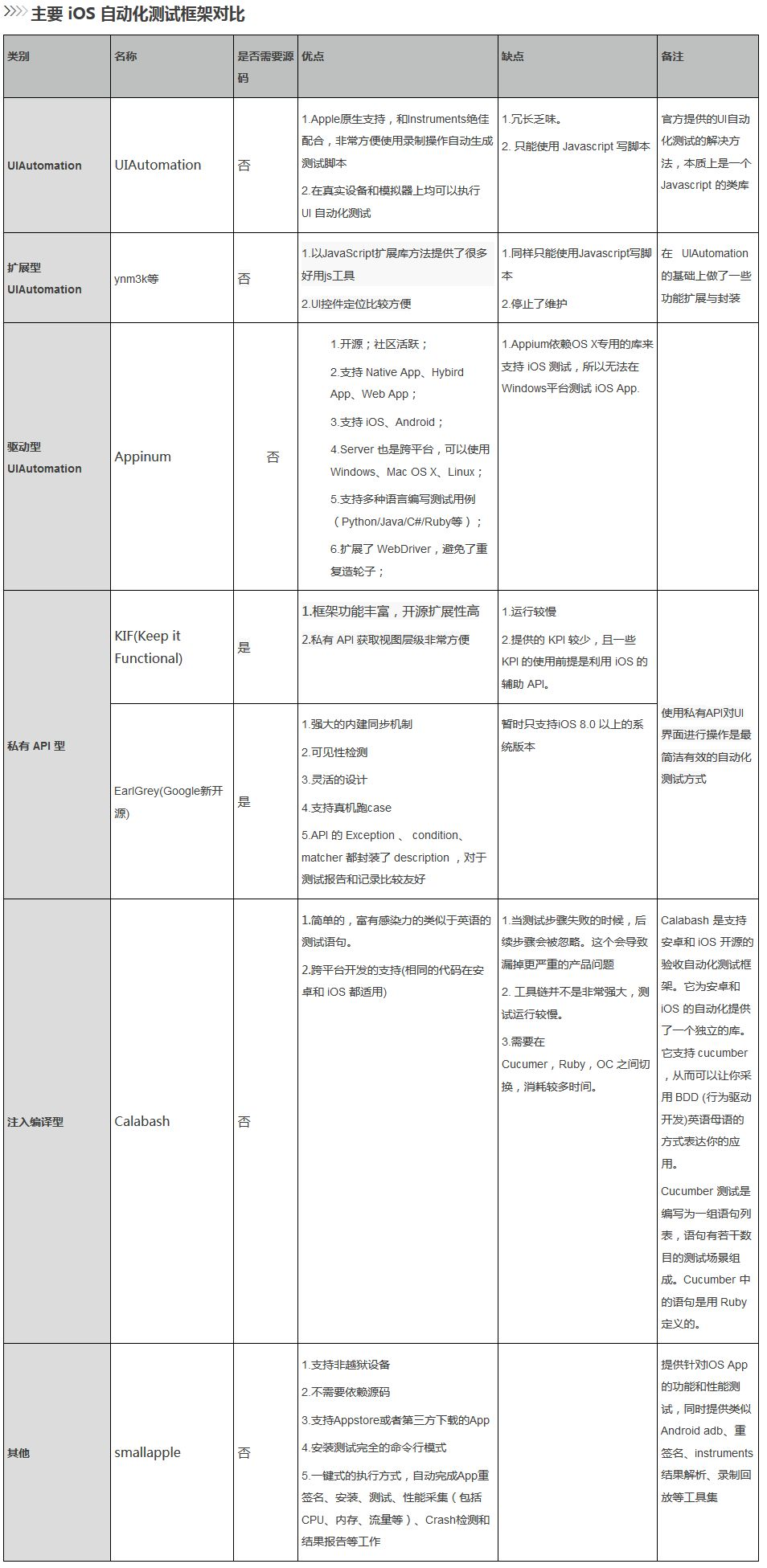
驱动型 UI Automation 在自动化测试底层使用了 UI Automation 库，通过 TCP 等通信方式驱动 UI Automation 来完成自动化测试。这种方式下，编辑脚本的语言不再局限于 JavaScript 。常见的框架有 iOSDriver、Appium等

**私有API型框架**

私有 API 型框架直接使用 Apple 私有 API 对 UI 界面进行操作。常见的框架主要是 KIF

**注入编译型框架**

注入编译型框架在编译时注入一个Server到App内部，通过Server和外界通信来完成UI指令操作，常见的框架主要是Frank、Calabash



**基于UI Test的第三方框架**

Xcode 7发布后，Apple 提供了一种新的 UI 自动化测试解决方法——UI Testing，它基于 XCTest 测试框架，通过控件的可访问性来定位和获取控件，并提供了多种 UI 操作 API，使用源码语言，能方便地进行调试。 我们在以上分类中挑选具有代表性的自动化框架：UI Automation、Appium、KIF、Frank、UI Testing 进行对比，下表是这几种测试框架的特点对比：



（图源https://tech.meituan.com/2016/09/02/ios-uitest-kif.html，美团技术团队的文章）

**业界常用自动化测试框架**

**KIF**

美团在用的框架，属于私有API型框架，使用XCTest测试框架，利用苹果提供给辅助功能的辅助属性accessibility attributes，注入自己的API进行测试并生成测试报告。利用KIF的私有API，获取页面元素的属性值，编写脚本并插入，利用断言来判断页面的点击事件是否符合我们的预期。

* 优点：使用OC||Swift，上手学习成本低，添加了私有API相比于官方的UI Test框架提供了更加全面易用的支持

**Appium**