

猿辅导公司的

Swift 迁移之路

唐巧

公司简介

- K12 领域的独角兽公司。
- 拥有 2 亿用户，月活几千万。
- 互联网女皇报告中的在人工智能领域崛起的中国公司。



旗下产品



猿辅导

中小学全科在线直播课



猿题库

自适应智能练习和测评系统



小猿搜题

中小学拍照搜题应用



粉笔公考

公务员在线学习第一平台



斑马英语

英语启蒙

关于我

- 唐巧，小猿搜题产品技术负责人
- 长期分享技术，拥有共计 10 万的微博和微信公众号粉丝
- 《iOS 开发进阶》、《iOS 面试之道》作者



大纲

- 猿辅导 App 的 Swift 迁移之路
- 猿辅导老师端 App 的 Swift 迁移之路
- 小猿搜题 App 的 Swift 迁移之路

Date	Version
2014-09-09	Swift 1.0
2014-10-22	Swift 1.1
2015-04-08	Swift 1.2
2015-09-21	Swift 2.0
2016-09-13	Swift 3.0
2017-09-19	Swift 4.0
2018-03-29	Swift 4.1
2018-07-05	Swift 4.2

猿辅导

的

Swift 迁移之路

决策回顾

- 背景
 - 时间：2016 年 6 月
 - Swift 版本：Swift 2
 - 依赖很重的 C++ 直播库
 - 历史 Objective-C 代码行数：8 万行
 - 团队 iOS 成员人数：2 人
 - 大家都对使用 Swift 抱有极高的兴趣

```
commit e8529c341bbfe6ff22af79b787f16b438b10be02
Author: huangling <huangling@fenbi.com>
Date:   Fri Jun 17 17:34:58 2016 +0800

    ADD: swift startup

Change-Id: I8e02820bbd3ecb6962ed616f29c6aaf0316f429c
```

决策方案

混编，从此之后不再写 Objective-C !

混编进展

- 优势：
 - 面向协议编程, Lazy Load, type infer, guard
- 问题：
 - Swift 升级
 - IDE 补全慢/挂掉, Xcode 无故白屏, Internal Error, 系统的方法补全无故失败
 - 混编的编译速度奇慢（最慢出现过打包一次 40 分钟的情况）

调整

- 升级打包机和开发机
- 当前项目代码量：
 - Objective-C 25万行
 - Swift 5万行
- 2018 年1月开始，新的项目代码用 Objective-C 编写

猿辅导老师版

的

Swift 迁移之路

决策回顾

- 背景
 - 时间：2016 年 7 月
 - 全新的项目，历史 Objective-C 代码行数：0 行
 - 需要依赖 C++ 直播组件
 - 团队 iOS 成员人数：2 人

决策方案

纯 Swift 写一个新项目

遇到的问题

- 编译器性能问题，编译一次 5 分钟
- IDE 无误编译，卡顿严重。
- Swift 和 C++ 底层直播库的调用麻烦，调试时常出问题。
- 系统补全时常挂掉
- pod 1.5 对动态库支持不好，很多问题

解决方案

- 黑苹果
- 用 injected 来 run 起来调整界面
- 用 OC 做 Swift 与 C++ 的媒介，在 OC 代码处进行对象的 po



现状

- 当前 Swift 代码量：6 万行
- 继续用 Swift 在编写，但是感觉并不太特别舒服

小猿搜题

的

Swift 迁移之路

决策回顾

- 背景
 - 时间：2016 年 10 月
 - Swift 版本：3.0 版本
 - 历史 Objective-C 代码行数：8 万行
 - 团队 iOS 成员人数：3 人
- 挑战
 - 产品迭代需求无法保证团队有充足时间换到 Swift
 - 兄弟产品混编 Swift 遇到了很多问题

决策方案

开启一个新的 Swift 项目，利用偶尔的需求空闲时间，完全重写小猿搜题

决策调整

- 背景
 - 时间：2018 年 7 月
 - Swift 版本：4.2 版本
 - 历史 Objective-C 代码行数：12 万行 (+4万行)
 - Swift 项目代码行数：5 万行
 - 团队 iOS 成员人数：3 人
- 挑战
 - 需求增加太快，Swift 重写赶不上新需求的增长

- 决策方案：模块化 Objective-C 版本，希望未来能够复用 OC 版本的模块

组件化构想

公司共用的
Common 组件

项目共用的
Common 组件

组件化构想

项目的底层组件

公司共用的
Common 组件

项目共用的
Common 组件

组件化构想

业务功能组件

OC

业务功能组件

OC

项目的底层组件

公司共用的
Common 组件

项目共用的
Common 组件

组件化构想

业务功能组件

OC

业务功能组件

OC

业务功能组件

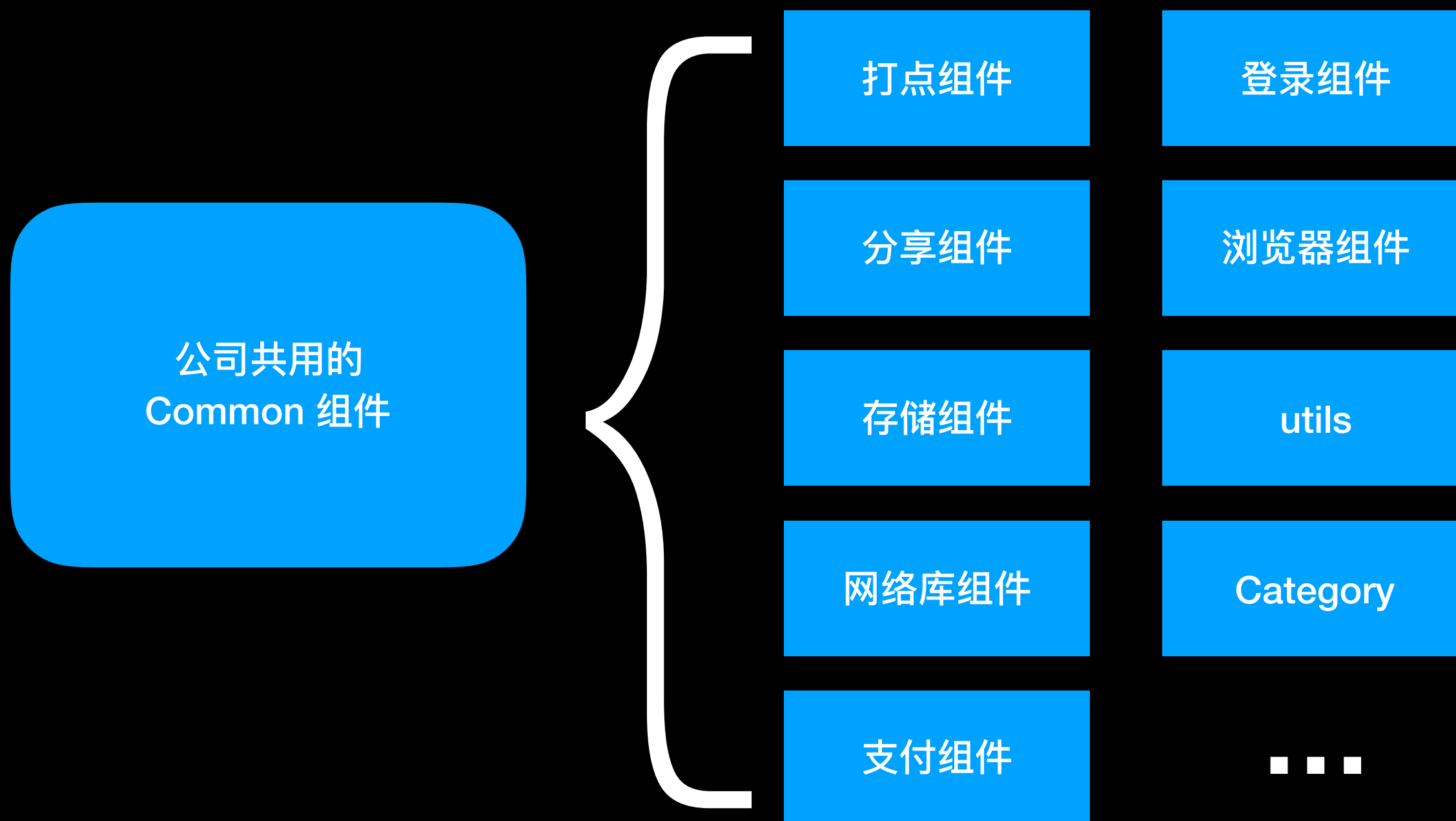
Swift

项目的底层组件

公司共用的
Common 组件

项目共用的
Common 组件

组件化构想



项目的底层组件

网络组件

视频播放

...

资源升级

项目共用的
Common 组件

公共 UI 样式

...

公用的参数

业务功能组件

OC

小猿商城

视频播放

24 点游戏

收藏功能

奥数闯关

导出PDF

英语随身听

智囊团

小猿日报

■ ■ ■

组件化进展

- 已经完成约 30% 的功能模块组件化
- 功能模块耦合太严重，是组件化的挑战

组件化进度			
组件	子组件	开发人员	完成状态
common			done
商城	客服		done
24点			done
奥数	练习		done
作文批改	相机		done
作文、英语作文			done
随身听			done
发现页			done
日报			
登录和账号			
拍照裁剪			
搜题历史			
收藏列表			
导出PDF			
智囊团			

小结

- 用 Swift 搭建核心项目代码
- 以功能为单元粒度，组件化迁移原来的 OC 代码
- 相比混编，该方案 Swift 程度会更高，随时可以将功能部分重写成 Swift 版本，偏于最后完全 Swift 化
- 相比重写，该方案在时间成本上较为可控

反思

- 效率没感到明显提升

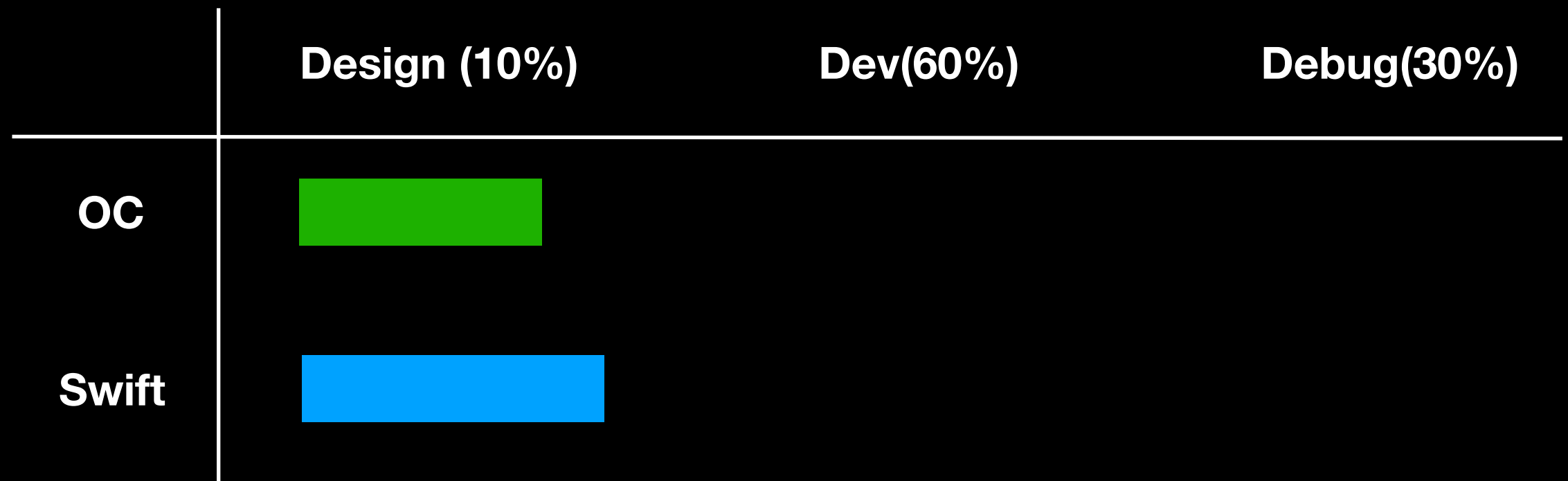
写代码最花时间的是什么？

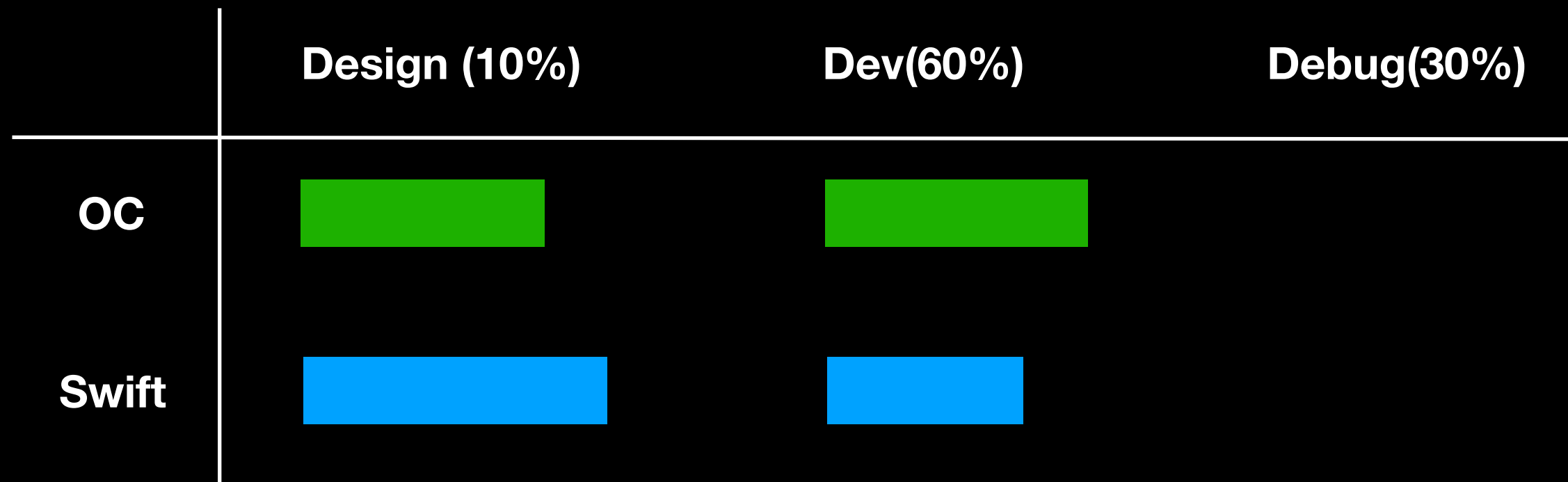
Design + Dev + Debug







Design + Dev + Debug







10% + 60% + 30%

	Design (10%)	Dev(60%)	Debug(30%)
OC			
Swift			





	Design (10%)	Dev(60%)	Debug(30%)
OC			
Swift			

	Design (10%)	Dev(60%)	Debug(30%)
OC			
Swift			

感觉用 Swift ， 效率并没有什么提升。

我其实不是来唱衰 Swift 的

未来一定会变好

在 OC 当保守派其实很难

不肯学习
不愿担风险
守旧不够乐观保守
用不流行的技术
不够积极
不肯努力
不愿改变

希望大家能够客观评判引入新技术的好处和风险

谢谢~