深入浅出安卓，怎样从零学好移动开发

转载

weixin_30478923 于 2017-04-20 08:13:00 发布 103 收藏
文章标签： 移动开发 java 操作系统
原文链接：http://www.cnblogs.com/lxjshuju/p/6736804.html

因为近几年来互联网的飞速发展，安卓和iOS平台的大量普及推广。移动开发在当前是很热门的一个方向。

有不少同学问我怎样学习安卓。要学些什么，难不难学。之前一直没有想好应该怎么回答这个问题，仅仅是简单的说安卓自身门槛不高。并不难学。由于我认为准确回答一个类似这种问题往往须要灵感。如今依据我的学习体验，做个大概的总结。

1、我为什么学安卓

我从刚開始接触安卓开发到如今也有两三年的时间了，原本我是打算做硬件方向的。对安卓仅仅是感兴趣。全然没有考虑过工作的问题。

后来慢慢感受到硬件难度偏大，成就感比較低，也不太想做那种技术性非常强的东西。纯技术思维，基本不是必需和大众用户交流，由于嵌入式、硬件方面一般都是比較底层的。直接用户还是技术人员。

个人感觉长期这样easy减少情商，作为一个情商本来就偏低的技术宅，与非技术人员交流会更加困难（导致找对象都比較困难）。

于是后来慢慢改成了安卓开发。安卓属于大众型产品，非常多时候须要从大众用户的角度去考虑问题，技术性相对弱化了。

前段时间面试，顺利的进了美团的安卓研发岗位。

我之所以学安卓，也是由于碰巧大一时学校有一门安卓选修课，就去学了下。有打算转做iOS，主要是iOS开发条件比較高，最好要苹果笔记本和苹果设备，还有每年100美元的开发人员账号（尽管能够用所谓的黑苹果，可是据说开发起来easy出现故障）；加上之前一直没时间，所以也还没開始去学。

2、安卓要学些什么。难度怎样（重要）

在我看来安卓开发有两层意思，第一层意思就是安卓自身的开发知识。而第二层意思是安卓、移动应用乃至各种软件开发的编程思想。这两者的关系，就像文字和写作的关系一样。

小学的时候我们就在学习识字写字。要说这件事难不难，显然仅仅要肯花时间就不那么难。毕竟我们小时候都是这么过来的，你能看懂我写的这个，说明你也是认识字的。

我们识字的目的是干什么呢？目的是看懂别人写的东西，以及自己写东西表达想法传达给别人。会识字而没有太多思想的人。写不出来好文章；而有思想的人，不须要学会非常多汉字就能写出不错的文章，还能够随时查字典。

相同，我认为安卓学习也是如此。

2.1 安卓自身开发知识

第一层含义，安卓自身开发知识。仅仅要肯花时间，理解力略微好一些的人都能学会。

安卓开发首先最好有Java基础，没学过能够先简单的去学习一下。

推荐两本书《Head First Java》和《Java编程思想》（英文名《Thinking in Java》）。Head First系列的书，特点是简单好懂，适合入门（最好有其它语言编程的基础），但解说不是非常深入。对于理解力够强的人。看起来比較费时间。而后者是Java语言公认的权威经典书籍，假设想全面的学习Java。能够看后面这本书，可是难度偏大，尤其是对于没有学习过面向对象编程语言的人来说。

安卓开发本身的技术知识，无非就是各种封装好的API接口函数（API=Application Programming Interface 应用程序编程接口），你仅仅要依照它的规则去调用即可了。安卓的接口有官方给出的完整说明文档，安装了安卓开发包后。也有自带docs目录，里面就是说明文档。

对于英文水平欠缺的人可能略微有点难度，另外，因为谷歌訪问不了，网页版的说明文档有些可能会打不开。假设你不想自己看官方英文文档，你能够直接买本安卓开发的书，或者在网上找安卓学习资料。

这些资料也是别人依据官方文档和自己学习经验总结出来的。有时候假设须要用到一些别人非常少用到的东西，或者网上说的比較含糊，这时官方文档是最权威、准确的參考资料。

2.2 安卓项目实战

第一层意思非常easy掌握，也就是安卓开发的基础知识，相当于学习怎么识字和写字。而第二层含义，安卓项目的实际开发。难度就要大一些了。

有些人一辈子就那么过去了。从来非常少思考复杂的问题，也没有什么思想。

而写作是须要灵感和思想的，仅仅是会写字，写不出来好的文章。相同的道理。有些人学安卓，仅仅是掌握了主要的API，却怎么也写不出好的程序来。

然后他们不知道应该怎么做，可是又不甘落后，索性把iOS开发、Windows开发，各种程序API都学一遍。而始终仅仅能做出来一些非常easy的东西。

如今问题来了，编程究竟须要学习哪些思想呢？应该怎么学呢？这里的编程思想，并非专门针对安卓而言，而是针对全部软件开发而言。找工作的时候。你会发现。有些公司的软件研发岗位命名招聘的是iOS。可是并不一定要求掌握iOS，有安卓、Windows或其它上层软件开发经验也能够。这就是由于，仅仅要从一种编程语言、一种开发环境学到了软件编程思想，再去学其它环境和语言下的编程就easy得多。

2.3 理论和project

软件分为两部分。理论和project。

理论方面。国内的发展不是非常好。

比方人脸识别的程序。须要用到一些数学理论模型，并以此做出算法来解决这个问题。在一些有实力的公司。会有这种理论研究部门。最典型的像苹果、谷歌、微软这种巨头，研究自然语言处理、图像处理、大数据分析、人工智能等等诸多问题。苹果的Siri。谷歌的安卓内置的语音识别引擎。微软小冰等等。而在国内。百度的搜索引擎对自然语言的处理、科大讯飞的中英文语音处理方案等，也都是须要大量的理论知识。

由于理论研究非常大的一个特点是不确定性。非常可能研究了非常久也没有成果。而实力不足的公司非常难有这种资源进行理论研究，所以在国内主要是有实力的大公司，以及国家提供经费的研究所、一些大学的实验室，才干有条件进行这方面的深入研究了（顺便提一句，也正是由于这种不确定性，国内学术腐败比較严重，各种抄袭，另外即使研究不出来成果也有经费）。

通常假设一个公司软件的研发须要用到深厚的理论。会专门给这个设置一些岗位。比方算法project师、图像算法project师等。而安卓研发、iOS开发这类。则更偏向于project应用。当然有时候。也会涉及到一些简单的算法问题，那些更像是小学奥数题。关键看解决这个问题的思路是否灵活。往往不须要非常强的理论知识。比如我在网上看到一道历年的百度笔试题：百度地图其中，每一个地点的左侧或右侧须要放置地名，地名的文字是矩形区域；设计一个算法，使得尽可能多的显示地名，同一时候要考虑地图的缩放。这个问题没有什么标准答案，出题者也许也想知道最好的答案；而实际实现时，就要看有没有足够聪明的程序猿，能给出一个尽可能好的方法了。

project的特点就是把理论应用到实际上来，并且要考虑到开发成本、时间、安全性等实际问题。

专门研究这些project的东西组成了《软件project》的学科，可是仅仅有软件相关学院才会去上这样的课程。并且这样的课程太抽象了，理论性太强，往往忽视了实践的环节。

2.4 软件project的重要思想：模块化和代码复用

软件project思想有非常多，模块化、代码复用是当中非常基本、非常重要的编程思想。所谓模块化，就是把一个完整的东西拆分成非常多个小的模块，每一个模块完毕一定的功能，分工协作，然后依照合适的规则则合成一个完整的系统。拿整个人类社会来说，每一个人都须要衣食住行，可是实际上，有的人专门从事服装制造，有的人专门做食品，有的人负责建筑……最后整个人类社会分工协作，效率大大提高，构成一个总体。拿计算机来说。我们的电脑由主板、内存、硬盘、屏幕、各种外设组成，每一个模块被独立设计制造出来。而且仅仅要接口吻合，能够任意进行组合。买电脑的时候我不一定须要内置蓝牙模块。可是在须要用的时候。我能够非常轻松的买一个USB蓝牙模块装在电脑上。

嫌内存太小，我也能够自己给电脑换内存，而不须要更换整个电脑。

程序也是这种，我能够写一个模块专门用于网络连接的相关控制。以后无论做什么应用，仅仅要用到网络，直接把这个模块放进去调用。积累的模块多了，后面就能像搭积木一样搭建不少的代码。大大减轻了程序开发的负担，提高了效率，节省了成本。

而模块化开发也有利于分工合作。一个庞大的程序一个人不一定能做完，比方我们用的Windows操作系统，代码量可能达到几千万行甚至很多其它。这时候就须要非常多人共同完毕。

每一个人或一个小团队完毕一个小的模块，而且不同的模块之间规定好接口，然后同一时候进行开发。模块化编程实现了代码复用、提高了开发效率、有利于分工协作。等等长处。是软件开发的核心思想之中的一个。

为了实现模块化，不同的模块之间要尽可能减小耦合度。也就是说。一个模块对于外部相当于一个黑盒子。我们仅仅能看到对外的接口，而模块内部的详细实现，与其它模块之间的关联应该尽可能小。

这样在改动一个模块的时候，仅仅要保持接口不变，对于整个软件来说就没有影响。

2.5 软件研发相关的学科知识

软件开发须要的一些公共的知识，也是面试常常会问的学科知识有《数据结构和基本算法》《数据库》《操作系统》《计算机网络》《设计模式》等。

首先《数据结构和基本算法》差点儿是全部软件相关技术岗位必会的。数据结构能够简单理解成数据是怎样进行组织并保存在电脑的内存中的，而基本算法则是研究怎样高效的对这些数据进行读取和处理。比方查找、排序，比較考验智商。

数据结构和算法原本是两种知识，可是因为他们之间的关系非常密切。所以这两者常会作为一个学科。一起学习。通常假设你从事软件研发。要求掌握基本算法就能够了。也就是数据结构课程中介绍的算法。

假设你的算法更强，能够考虑专门从事算法研究，那也非常不错（假设算法学的非常好，能够去谷歌总部，顺便就出国了）。数据结构推荐书籍《大话数据结构》。

《操作系统》看上去似乎和应用软件没有密切联系。可是有非常多时候，软件设计须要用到多线程等知识，这个时候，对操作系统的原理有所了解，会做的更好。

毕竟应用软件是执行在操作系统之上的。

《计算机网络》在应用软件中使用非常广泛，我们用的大多数应用都须要用到网络，所以这门课必定是非常重要的。推荐书籍《现代操作系统》、《计算机网络》。

所谓《数据库》，就是最经常使用的一种数据的保存手段。我们用QQ给被人发送消息，一条一条的消息并非简单的用文本文件保存在手机里的，而是通过数据库进行保存的。

对于应用软件开发来说。我们所要学习的是数据库的使用。一般不须要深入了解数据库的实现原理，所以学起来不会太难。数据库最经常使用的是SQL和SQLite，两者语法非常接近。SQL语言号称是第四代编程语言，而C语言这样的是第三代，越是上层的语言越接近自然语言。所以SQL语法也非常好理解。有些时候用到一些不太好理解的语句，主要是由于语句包括的逻辑比較难理解。倒不是SQL自身的问题。

举个样例，在一个表格mytable里保存了全班学生的信息。有number和name两列分别表示学号和名字。

这时我想知道小明同学的学号。我仅仅须要用以下的语句选择他的学号就能够了，差点儿和英语一样：

SELECT number FROM mytable WHERE name='小明';

编程有非常多优化思想，除了提高开发效率、分工协作。还会考虑到安全问题等。这些编程思想的大量研究，人们积累了非常多技巧，《设计模式》这一课程就是对一些使用频繁、经过了非常多人考验、而且非常有借鉴价值的程序设计思想进行的总结。

而设计模式的精髓并不仅仅是照搬那些模式，很多其它的是曾经人的经验积累作为灵感和素材，依据实际需求，创造出很多其它好的编程技巧和思想。

推荐书籍：《设计模式》（机械工业出版社）。《Head First Design Patterns》（中文名《深入浅出设计模式》）。《大话设计模式》。

2.6 移动开发独有的特点

除了上面这些以外，移动应用开发与传统桌面应用开发相比，另一些特别的东西。移动开发。也就是针对移动平台进行的应用开发，手机、平板等产品。受限于有限的屏幕、CPU速度、内存、电源供应、能够随便移动、网络费用可能比較高等特点，移动开发就有一些比較值得注意的东西了。移动应用的界面应该简洁、方便。button文字等设置的大一些。方便操作。充分利用手势进行操作。还有针对安卓和iOS等不同平台进行优化，符合用户使用习惯（比如安卓有返回键。可是iOS没有）。然后在程序的数据处理方面，要充分考虑移动设备自身性能。进行各种调整。这类的问题有非常多。假设须要深入了解，能够看一些相关的书籍。

2.7 安卓开发可能用到的知识，或研究方向

在Facebook等一些公司。流行一种概念，叫做全栈project师。所谓全栈project师。说的直白一点，就是一个人独立完毕整个项目，包含client开发、前台网页设计、后台server搭建等。

这对于project师的要求很高，不仅要知识面广，并且各个方面的学习都得有一定的深度。

这里我不打算讨论什么全栈project师，我自己对server那些也没什么概念。我仅仅是依据自己的经验和了解，总结下我所觉得的、安卓开发还有可能要涉及的知识。

首先是平面设计、交互设计、用户体验。

移动应用作为大众型产品，用户体验相当重要。

假设有过个同类产品，功能接近。用户肯定更喜欢用户体验更好的产品。用户界面须要用到设计方面的知识。当然在大部分公司，通常会有专门的交互设计相关职位。所以对于应用开发人员来说，设计方面不须要掌握的太深入，做一点简单了解当然是没错的。

然后上面已经说了非常多软件project方面的问题，数据结构、设计模式、操作系统、计算机网络等等。不再反复。

安卓NDK环境和JNI开发。

安卓基于Linux操作系统。主要由Java编程。可是有些时候须要用到C++。比如核心代码须要保密，而Java保密性有所欠缺；有些程序仅仅有C++环境才干实现；要用到一些高性能的算法等的支持，而Java运行效率偏低。这是我们能够使用Java的JNI，调用C++开发的程序库完毕功能。

C++的开发基本上就和在Linux上编程几乎相同，差别在于安卓系统中有一定的权限限制。而安卓NDK就是官方给出的、用于高速开发安卓JNI程序的开发环境。

安卓系统的实现、系统级开发。

安卓系统有个非常大的特点是开源免费，因此我们非常easy就能获取安卓系统的源代码进行学习。了解安卓系统的设计。了解安卓系统设计，后来我们就能够从事偏底层的安卓开发。系统订制。乃至安卓驱动开发、操作系统开发方面的工作。安卓是个优秀的操作系统（比如小米手机系统就进行了深度订制）。

游戏开发。移动游戏眼下是非常火的行业，非常多公司从游戏产品中获得了大量的收入，游戏开发自然是一个不错的选择。大型3D游戏往往会使用各种游戏引擎来进行开发。因为我眼下差点儿没有做过安卓游戏，所以也没有太多的了解。不做过多讨论了。

另外还有server方面的研发。我们手机上必备的软件有QQ、微信、支付宝等。这些软件非常重要的特点，不是在于软件自身，而是由于强大的后台网络服务支持。于是server方面的研发也不错，只是这已经不直接属于安卓应用开发的范畴了。另外还有推广运营、管理等。和安卓开发有直接关联，我并不太了解。也不做过多讨论。

整体来说。学会安卓门槛非常低；可是学好安卓绝非易事。

3、安卓学习方法和技巧

3.1 整体学习思想

回到一開始文字和文章的类比上来。我们大部分人两三岁的时候就能学会用汉语说话，小学的时候就能认识非常多汉字。可是为什么学英语似乎比汉语难非常多呢？至少从初中就開始学英语，各种语法、单词，一直到大学，考四六级。直到大学毕业，非常多人的英语还是远不如小学毕业时的汉语水平。

我也是当中之中的一个。大学不仅没进步，反而曾经学的单词都快忘记了。

原因非常easy，缺乏实践。我们学习语言的目的是为了应用。可是在大部分人的生活环境中，根本没有太多须要用英语来交流的地方，然后自然连单词也慢慢忘了。假设我们想写文章。没有好的想法。能够多去看看别人的文章和书籍学习学习。而有一些个人的想法，仅仅需掌握主要的文字，就能够写了。

遇到不会写的字。翻翻字典就好了，不是必需把各种生僻字都记住。一样能写出好文章。而背单词，死记硬背记不牢；多练习多实践，不熟悉的东西也慢慢熟悉。自然记住了。

另外实践多了，我们还能积累非常多好的句子，比方各种诗词之类，写作时就能够充分利用。

相同的道理，安卓的学习，首先是应该知道最主要的一些东西。

我们能够大致的看一本安卓入门的书籍，依照书上说明，搭建开发环境，把常见的基本接口简单的实践一下，有个总体了解（认识经常使用字）。然后我们就能够实际运用了，也就是做项目（写作文）。

遇到问题。我们须要自己多动脑思考，多在网上找解决方法。实在解决不了再考虑请教别人（查字典）。

假设对编程的思想掌握起来比較吃力，能够去看看网上的一些开源程序源代码（看别人的文章）。久而久之，不仅对常见的API有了了解，也慢慢学会了编程的思想和技巧（自己会写文章了）。

编程时。要注意模块化，把经常使用的一些自己写好的模块封装起来。做好凝视。以便以后使用（好词好句记录）。另外，学习的东西记得及时做一些笔记和总结，假设整理的比較好，也能够发表到网上。对别人也许也会有帮助（做笔记和分享）。

软件开发还有个非常重要的过程就是程序的调试。在安卓中，因为用的是Java，程序调试手段非常多，也非常方便。

安卓提供了一个Log接口。能够在关键的地方打印日志，然后在Eclipse的LogCat窗体中查看，对于程序调试会有非常大帮助。

Java应用在执行出错时。能直接显示出错的代码位置和错误类型，也是在LogCat窗体中显示，然后就非常easy找到错误的原因所在了。

安卓开发时，能够电脑连接手机在线调试，设置断点，查看变量的值、执行的进程和线程、内存消耗、文件等信息。详细方法请自行搜索。

3.2 基本知识学习

说的详细一点，安卓学习的过程大致是这种。

首先是搭建开发环境，通经常使用Java+Eclipse+ADT插件+Android SDK，也能够用Android Studio，详细方法网上有非常多参考资料。开发环境中集成了安卓虚拟机，假设你没有安卓设备，能够在虚拟机上执行程序。可是速度较慢，不支持一些传感器等硬件设备；假设有安卓设备。最好在实际设备上执行程序。

然后是掌握安卓四大组件。尤其是Activity和Service及其生命周期（BroadcastReceiver和ContentProvider能够后面再学）。Intent实现界面的跳转。Menu菜单。然后是安卓的经常使用控件、XML布局（Layout）等。这些是安卓最基础的东西，能够通过编写Demo程序的形式去学习。

网上有个文档《深入浅出Google Android》。里面就通过一个简单的安卓程序实例。介绍了这些知识。

3.3 进一步学习

到此安卓最主要的基本API就算是学习完毕了。

然后还有SQLite数据库、各种传感器、动画控制、多媒体、网络通信、GPS定位、电源管理等API。这些API能够先仅仅作简单了解，直接去写实际项目。你能够试着写一些简单应用。比如计算器、音乐播放器、小游戏。或者你所感兴趣的简单应用（一開始难度不要太大）。须要用到的API再去具体的学习，逐步锻炼编程能力。代码要规范，尽可能符合Java命名标准，程序代码尽可能写成模块化的。提高代码复用。

记住，完毕相同的功能。在保证程序结构清晰、模块化、规范化的基础上。代码量越少越好。

3.4 深入学习，并开发高质量应用

而再到后来。你可能须要更深入的去学习安卓，这个时候可能须要了解一些安卓系统Java层的源代码（安卓底层用的C和C++，上层开发包中API用Java编写）。能够在网上下载到，然后在Eclipse中设置关联源代码。须要查看源代码时。直接用Eclipse转到函数定义。就能看到安卓系统的Java层源代码了。另外你可能须要学习《操作系统》《设计模式》《软件project》等前面提到的课程知识，加深对软件开发相关知识的理解。

比方你能够自己独立完毕或者与别人合作做一些项目。可能涉及到多线程、大量数据的处理、JNI的使用、自己定义控件和界面布局，识别特殊的用户手势，游戏引擎等等（能够參考网上的开源项目。以及平时我们用到的各种手机应用）。

我的建议是。后期做安卓应用的时候。直接做功能完整的应用，而且要经过重复测试调整；尤其是要注重用户体验。还有程序的规范性、稳定可靠性（比如Java中空指针的推断、try...catch的使用、线程通信等）。这样才干非常好的学习移动开发的精髓。假设你仅仅是为了学习那些API，做出来一些体验非常不好的Demo级别应用，仅仅能说是学会了安卓，却没有学好。

也许你会认为有些应用功能很easy，要不了多少时间就能做好，实际上远不是那么简单。一个优秀的安卓应用，不仅用户界面和体验很好，并且程序规范、稳定可靠、运行效率高。可扩展性强，想做到这一点。很的不easy。一个优秀的商业安卓应用。基本的代码实现阶段。可能仅仅占了整个应用开发时间的1/3甚至更少。在开发之前，有不少的时间是在进行应用的策划安排；而在开发完之后。又须要大量的时间，相应用进行重复的测试调整更新，最后才干被公布，从而安装到我们的手机上。

这里顺便一提，安卓应用开发相比iOS的一个难点来自安卓系统碎片化问题。安卓系统是开源免费的。这是一大优势，也因此对非常多国产和国外手机制造商带来了非常大的优点（假设没有安卓。非常多手机厂商恐怕都深陷危机之中了。也许移动互联网也不会发展的这么快。不知道如今是不是iOS要称霸天下。或者WP大受欢迎，又或者塞班还会屹立不倒）。

可是安卓的这样的特点，导致同一款安卓应用至少要同一时候兼容各种主流手机型号。各种配置，各种屏幕尺寸，各种系统环境。而这也是迄今为止安卓开发人员心中永远的痛（ノ﹏ヽ）。

4、附上我的学习经历和部分作品

这里简介下我的安卓学习经历。假设读者能从中得到一些启示那最好只是了。

4.1 初学安卓

当初学单片机的时候。编程至少是底层到C语言，有时甚至是汇编指令。然后再究竟层寄存器、数字电路、模拟电路的理解，差点儿全部的东西都得自己实现。因此一開始接触Java和安卓。我非常不习惯抽象的上层编程，总感觉那些封装好的函数调用非常难理解，由于不知道那些函数做了些什么。只是后来慢慢就习惯了。而且越来越感觉到Java这种上层语言非常高效好用。

我是从大一下学期開始上选修课学习安卓的。

当时仅仅了解C语言，Java并不了解，而安卓主要是由Java开发。Java代码尤其是像接口之类，对于没有学习过面向对象编程语言的人来说不太好理解。

比方我当时就一直记不住这种代码。认为函数里面又有函数真心非常奇妙：

```
button_ok.setOnClickListener(new OnClickListener() {
void onClick(View v) {
// ...
}
});
```

只是如今熟悉了Java以后，这样的代码再常见只是了，内部的newkeyword在括号里实例化了匿名的接口类实例，并对其抽象方法进行了实现。

当时上选修课我的感觉就是全然听不懂。依照一贯的作风。我没有认真的去听课，而是跟着老师的课程节奏。搭建环境，借了參考书，然后上网找资料。自己学。上课给我带来的优点是学习氛围。还有主要的学习方向。为了交大作业。我用一周的时间做了一个拼图游戏。在当时应该是全部上选修课的人中做的非常好的，拿到了95分的高分。

只是在如今看来。那个简单的应用仅仅能说是能用，而应用的质量太低，不是由于功能太简单没有新意。而是用户体验不好，兼容性也太差。

4.2 参加谷歌安卓大赛

大一暑假为了參加谷歌大学生安卓大赛（详情百度一下就能找到），借了一本安卓书。花了几乎相同两个月的时间，差点儿每天从早到晚。一边看书一边上网找资料一边写代码。

最后做了一个安卓多媒体备忘录应用GoodMemo，获得了谷歌安卓大赛西北赛区优秀奖。当时这个程序算是让我学会了安卓中的各种基本API，同一时候学到了非常多编程思想方面的东西（对于我来说，当时非常多东西是靠自己想出来的。过程比較慢，可是锻炼效果非常好；当然你能够去參考《设计模式》之类的书籍，或者看网上的开源项目）。要说代码量。Java代码写了两万行，XML没有统计。可是假设如今再让我重写一个一样的程序。肯定要不了那么多代码。如今再看，感觉那个应用的界面还是不够好，对不同手机的兼容性也不够。

这是当时參赛的演示视频 http://v.youku.com/v_show/id_XNDUyMDI5MTE2.html

因为谷歌有段时间没法訪问，今年谷歌安卓大赛的主页在这里 http://miac.buu.edu.cn/

4.3 再次参加谷歌安卓大赛

然后就到了今年年初，我决定又一次參加一次谷歌安卓大赛，并严格依照软件设计规范。编写一个全新的高质量安卓应用。去年年末设想作品方案，想到要做一个闹铃应用。至于有什么特点，我公布到安智市场了，看介绍就知道（从我一開始做多媒体备忘录GoodMemo以来，就和闹铃应用结下了不解之缘，一个好的创意真的非常难 o(╯□╰)o）。

从寒假在家開始进行界面的概念图和交互设计，到实际编程（实际上闹铃数据部分的编程仅仅用了几天就完毕了，绝大多数时间用在了界面设计上），再重复的调整和修复各种BUG，包含应用的名称都想了非常长时间。最后还是姐姐给我的灵感。就叫做《It's the time》。断断续续直到8月末，才算是基本完毕了。

然后又找同学帮忙体验和提建议，对界面进行了不少调整。

面试时就演示了我的这个作品。感觉还不错。

面试结束后又再做了一些调整改进，最后最终公布到安智市场了，能够点击以下的链接安装感受一下。在9月底的时候，提交到了谷歌安卓大赛站点。临时还没有開始评选。这也是我眼下能拿得出手的，充分考虑了用户体验的基本的一个应用。

可是兼容性还是不够，貌似在有些同学的小米手机上没法用……深度订制的小米手机系统真折腾人。。。至于功能，没什么好说的……实在想不出什么新意。

http://www.anzhi.com/soft_1856039.html

4.4 黑客编程马拉松

今年早些的时候，四月份的时候，我和另外两个队友參加了一次黑客马拉松编程大赛。我负责全部编程工作。

连续30小时。5000行代码。三等奖、最佳设计奖、最努力编程奖。我们的作品Ding，是一个语音控制的闹铃（又是闹铃+_+），事实上程序BUG非常多，也没有再去改动了，创意一般。实际价值不太大。

可是队友设计的界面风格和宣传视频受到了评委一致好评。而我编程的坚持不懈也受到了奖励。事实上虽说应用有5000行代码，实际上就和我前面所说，并不是全都是现场完毕的，而是有一部分之前就做过相关的模块。直接放进去用了。30小时的主要时间都用在了界面的编写和已有代码的移植、改动和完好上了。

4.5 实习经历

还是在今年，从三月份開始到七月份，我在一家公司实习，做的是安卓开发，学了不少之前没用到的东西，多线程、网络、语音识别、各种第三方API（比如天气查询）等等。主要是学了不少技术，项目不再细说。

4.6 最简单的手电筒应用

在一開始打算往安卓公布应用之前，我有一些东西不明确，比方应用签名之类的。个人开发的应用一旦公布出去，成为商业应用，会有什么须要注意的东西呢？当时纠结了非常久，也找了不少资料，决定先做个简单的应用试一下。然后在8月份的时候，我做了一个简易的手电筒应用SimpleLamp，公布到市场了。当时感受到安卓碎片化问题确实挺严重，即使是小小一个手电筒，本来简单的一个API就能完毕的事。可是因为要兼容不同型号设备，我看网上一些代码。还得对一些特别的手机型号进行推断、特殊对待，相当麻烦，而市场上那些比較受欢迎的安卓手电筒应用，必强调的一个点就是兼容多少种手机型号。所以这里我也来掺和一下，我的手电筒应用经过了更新以后，能兼容非常多型号的安卓手机（详细多少种我没法统计，反正我认识的同学下载使用的，都没有出现过无法打开手电筒的现象）。点击以下的链接能够下载体验。

http://www.anzhi.com/soft_1795050.html

以上就是至今为止我的安卓学习经历了。也欢迎大家支持我的作品SimpleLamp和It's the time。再发一遍下载地址……

It's the time http://www.anzhi.com/soft_1856039.html

SimpleLamp http://www.anzhi.com/soft_1795050.html

另外，APP开发的完整流程能够参考知乎上的这个问题和回答：

http://www.zhihu.com/question/19957949

至此，花了将近一天时间写的安卓相关总结就算是写完了，希望对大家有所帮助，也希望大家多多支持。

个人水平有限，有不论什么问题和不正确的地方，欢迎指出，或和我交流讨论。也欢迎大家关注我的个人主页 http://www.hainter.com 。

大家的支持会是我最大的动力，有了动力。我会认真写很多其它的文章来分享我的学习经验。