从零开始学iOS7开发系列3-我的地盘我做主-Cha5

原文及示例代码来自raywenderlich store中的iOS Apprentice 系列3教程，经过翻译和改编。

小伙子们，看看谁来了？~那啥，大叔，这台词能不能不要这么重复啊？天天玩炉石也就听到这几句了。欢迎继续我们的学习。

好吧，这一课的主题依然还是——Objective-C的理论知识。还是那句友情提示，如果你对这些理论知识一点都没兴趣，也可以完全跳过去再说。

关于协议(protocols)

在Objective-C中，protocols(协议）是个很重要的概念。虽然我们在前一个系列的教程中反复接触了这个概念，不过鉴于它很重要又很让人头大，所以还是有必要再提一提。

protocol（协议）可以简单看做一堆方法的列表：

```
protocol MyProtocol <NSObject> - (void)requiredMethod; @optional
- (void)optionalMethod;
@end
```

如果你在一个对象的@interface语句后面看到<>尖括号，就代表该对象遵从某个协议：

```
@interface MyObject : NSObject <MyProtocol>
...
@end
```

此时MyObject就必须在自己的@implementation部分实现requiredMethod方法。虽然在协议中还有一个optionalMethod方法，但你也看到@optional这一行语句了。顾名思义，optional就是可选的意思，那么在@optional后面的协议方法可以根据情况看是否需要实现。

单纯的协议作用不大，很多时候我们用它来定义delegates（代理）。

关于对象Objects和指针pointers

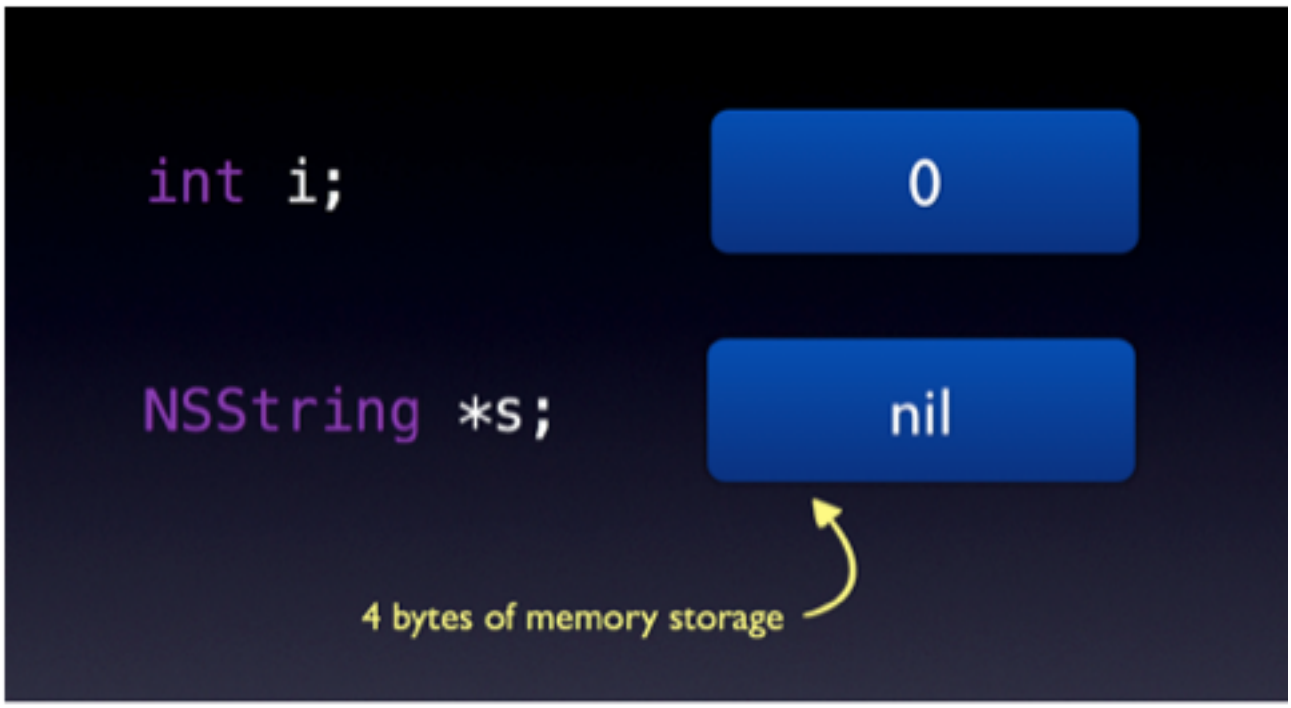
我承认最初在学C语言和C++语言的时候被指针难倒了，至今还心有余悸，看到指针就有一种要被人按倒在地菊花受虐的赶脚。
把pointer翻译成指针的那位砖家肯定也是在这个概念上被虐了千百遍，才选了这么个让人望而生畏的名词。

好吧，不过还记得我们的口号吗？Don’t panic. 虽然发明编程语言的人或许智商要比我们高那么一点点，但好歹也是爹妈生的，也是神经元的网状组合。对于同为碳基生物的发明，任何时候都要蛋定。

先别管这个词了，我们这里其实要讲的是基本数据类型变量和对象的本质区别。

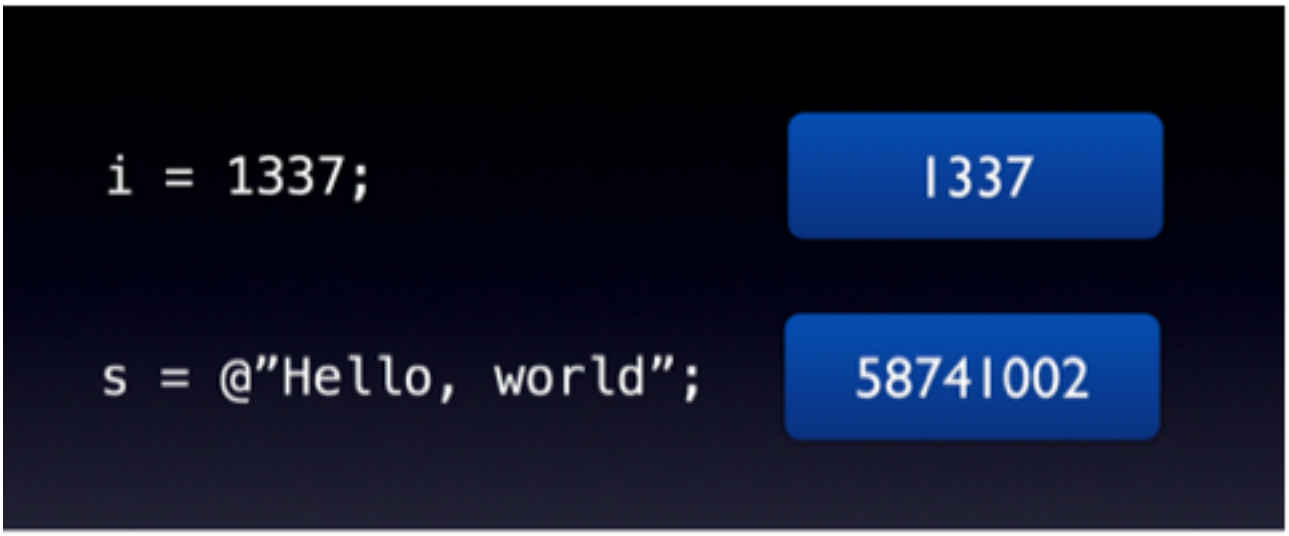
当我们声明一个int类型（或其它基本数据类型）的变量i时，编译器会在内存空间中预留一小块内存来保存我们可能要保存的数值（对于int类型通常是4bytes,4个字节）
而对于对象类型的变量也是类似的。当我们声明一个NSString类型的变量s时，编译器也会为它开辟一块4个字节的内存空间，和int类型一样。
这个貌似不科学啊，因为一个NSString对象中有可能会保存很多文本内容，4个字节的空间够吗？

默认情况下，变量i中保存的数值是0，而变量s的数值是nil，也就是不包含任何对象。



The memory that is reserved for two variables

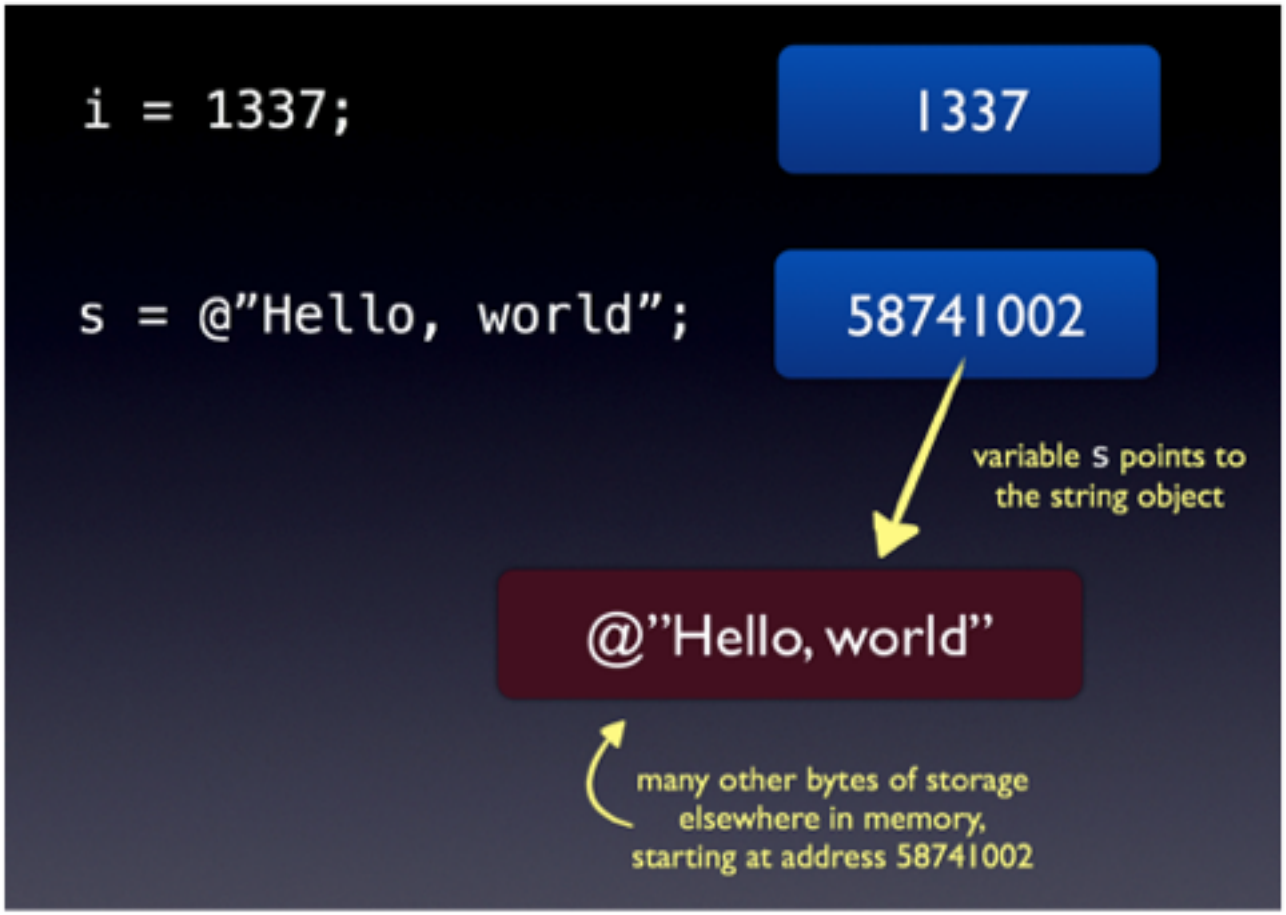
不过当我们在两个变量中分别保存了不同的数值时，它们所占用的内存就会如下所示：



The memory after the variables have values

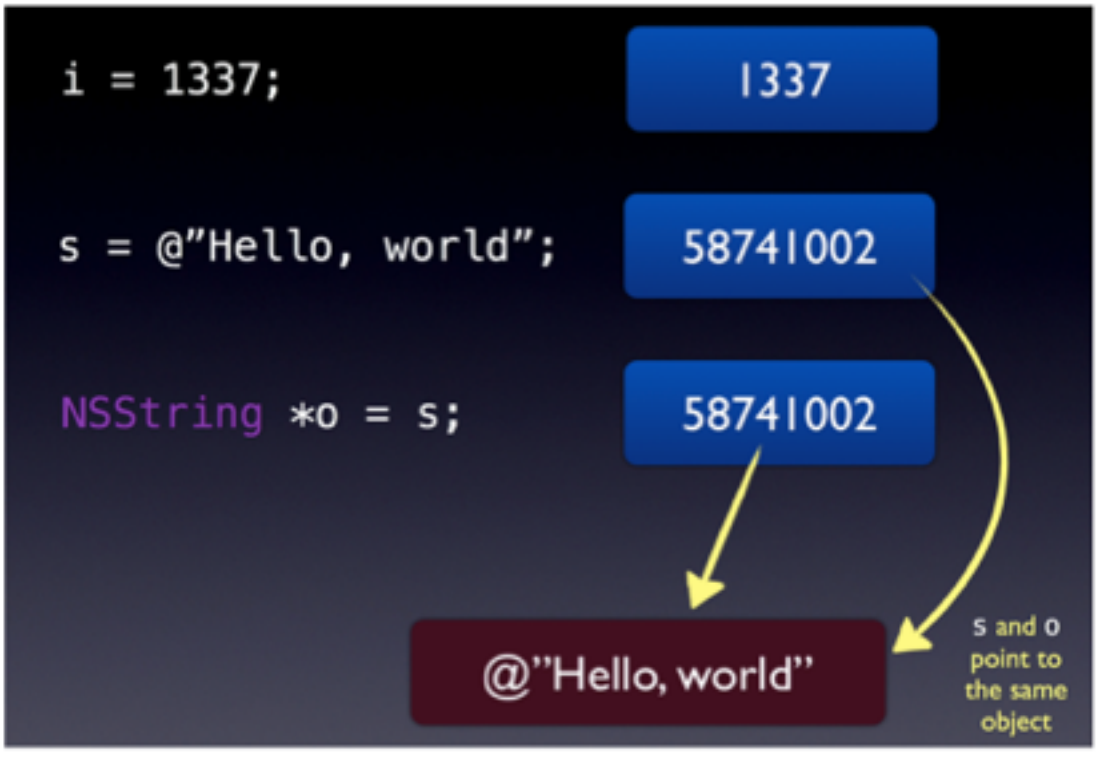
其中1337这个数值会直接保存到i所在的内存空间中，但变量s的值在内存空间中则用一个很奇怪的数字来表示。显然它看起来不像是文本。实际上，这里的数字并非NSString对象本身，而是NSString对象所在的内存地址。在计算机的世界中（智能手机、kinect,Oculus,智能手环、google glass从本质上都是计算机），内存中的每个字节都有一个独特的地址，而在这里地址58741002会指向字符串对象的首个字节的内存地址。

4个字节根本不可能提供足够的空间来保存s这个字符串变量，比如当其文本是@“Hello,world”时。因此，实际的情况是，当我们初始化一个NSString对象时，计算机会把它保存在内存中的某个位置，然后返回该内存空间的起始地址。而这里的58741002实际上就是字符串所在内存空间的地址。



Variable 's' contains the address of the string object

因此，这里我们所定义的变量s本质上是一个pointer（指针），因为在它里面保存的内容并非对象本身，而是对象所在的地址。通常可以读作"指向point to"某个对象。我们在定义对象的时候会在其名称前面添加一个*星号，就是为了区别包含了常规数值的变量本身和包含了指针地址的变量。
比如我们可以使用NSString *创建一个新的指针变量o，然后让它和s的数值相同。此时我们并没有拷贝这个对象，而仅仅拷贝了字符串对象所在的地址。
结果就是o和s都指向同一个字符串。



Variable 's' and 'o' both refer to the same object

指针对象的非直接性的确是C系编程语言（C,C++）让人头大的一个概念，当年Java的一大宣传语就是号称没有C和C++中让人头疼不已的指针。
幸运的是，Objective-C虽然也有指针的概念，但通常我们只需要知道怎么用就行了。不必像C或者C++那样需要频繁利用指针对底层进行操作。
对指针感兴趣的朋友可以参考：
http://www.lvtao.net/ios/c-pointer.html

注意：
虽然在Objective-C中*星号的主要是获取到对象的指针，但其实对基本数据类型其实也可以使用指针，比如int *。此时我们得到的是一个指向另一个int变量的变量-当然在Objective-C中很少这么做。
和C/C++一样，如果你想让自己玩玩智力游戏，还可以创建指针的指针。
不过对于入门者来说，这就好比一个在2维平面上的游戏人物想象3D场景中的人是怎么生活的。再想想看一个生活在3D场景中的人怎么看活在4维世界中的人。考虑到我们有3维空间和1维时空，貌似勉强算是4维世界的人。但如果让你想象一个生活在5维世界，6维世界甚至是更高维度世界中的人是如何生活的，你有何感想?

最简单的有时候反而是最有效的，虽然指针无比强大（C系语言开发者经常拿这个鄙视其它语言的开发者），但如果你理解不能，又有个P用?
当然，这里不是让你逃避指针这个概念，而是说学习有时候如烹小鲜，需徐徐图之。除了指针这个拦路虎，在学习iOS开发的过程中还有很多障碍。面对困难，首先是Don't panic，别害怕。其次是，根据需要去学，如果你的开发过程中基本用不到某个东西，就不必贪多求大。有了实际开发的需求再去学习，无论是理解的深刻度和实用性来说，都比你啃书本要强。

举个例子吧，如果你是个应用开发者，通常就不必要浪费时间去学SpriteKit,Game Controller，cocos2d和cocos2d-x,unity3d。

之所以在这里说这些废话，哥的意图其实已经很明显了。
是的~
恭喜你，猜对了，关于Objective-C的理论知识复习暂时到此结束了~

从下一课开始，将进入本系列的真正主题-我的地盘我做主。

既然是地盘，当然是移动开发特有的GPS定位先行了。

从下一课开始，你就可以在传统桌面开发人员和web开发人员跟前秀智商了。

听说你会C语言指针？不好意思，我就会搞搞GPS定位
听说你会struts+spring+hibernate？不好意思，我就会搞搞GPS定位
听说你会HTML5+CSS+javascript+JQuery?不好意思，我就会搞搞GPS定位
听说你会REST和JSON？不好意思，我就会搞搞GPS定位
听说你会TCP/IP和UDP协议？不好意思，我就会搞搞GPS定位
听说你会.NET？不好意思，我就会搞搞GPS定位
听说你会Unreal引擎？不好意思，我就会搞搞GPS定位
听说你会MFC？不好意思，我就会搞搞GPS定位
听说你会Node.js? 不好意思，我就会搞搞GPS定位
听说你会遗传算法和神经网络？不好意思，我就会搞搞GPS定位
听说你会hadoop? 不好意思，我就会搞搞GPS定位
听说你会LAMP？不好意思，我就会搞搞GPS定位
听说你会Ruby on Rails？不好意思，我就会搞搞GPS定位
听说你会Adobe Air开发？不好意思，我就会搞搞GPS定位
听说你会Core Location？不好意思，我就会搞搞GPS定位
。。。额，等等，貌似搞GPS定位必须会Core Location啊？一家人啊，误会误会~

好了，其实不管搞什么，要想精通都很难。不管是web前端开发，还是服务器端开发，不管是.NET开发还是SSH开发，不管是游戏引擎开发还是算法研究，不管是移动应用开发还是大数据，每个领域都不是那么简单。不管是哪种类型的开发人员都需要尊重。
最SB和2B的行为就是自己学了几天C++就去嘲笑用java的人，学了几天汇编和LISP就目中无人，学了几天ruby就觉得唯我独尊，学了几天unreal就瞧不起用unity和cocos2d-x的，学了TCP/IP和UDP

协议就看不上习惯用简单HTTP协议的，学了移动开发就瞧不起传统桌面开发和web开发的，学了iOS开发的就看不起Android开发的，如此种种几乎是业界常态，简而言之都是装B。

在《乔纳森传》中曾提到他在学习设计的时候采用的T字学习法。简单来说，就是精通一门，了解多门。作为开发者，最好是精通一到两门开发语言和框架，但同时需要多了解其它的技术。
了解和精通是两回事，经常看到有人在技术博客中号称自己精通多门开发语言和框架/工具，我只能说这些人情商很高。
编程分很多种，根据市场划分有游戏开发，有个人工具软件开发，有企业应用开发，有教育软件开发，有社交应用或网站开发。而根据平台类型，又分为传统桌面开发，游戏主机开发，移动开发，web开发，嵌入式开发。根据所涉及的语言、框架和工具，又有.NET,SSH/J2EE，LAMP，ROR，cocos2d-x等等。
即便是同一个语言、框架和工具下又有非常多的细节，不可能面面俱到。
究竟要学哪些，学到哪种程度，应该根据自己的喜好和工作需要，千万不要贪多求全。

世上的确有如乔布斯、盖茨、woz这样的天才和神级人物，但他们的数量毕竟太少，想学也学不来。
次而等之的如苹果首席设计师Jony Ive，他们极具天赋，但却依然会付出百倍于常人的努力，而且会focus在自己所喜爱的专长领域上。
作为寻常人，能不能做好两件事真的很重要。一件就是找到自己真正想要的事情（好吧，有时候是不得不做的事情，比如极度缺钱无法保证生存的时候），然后focus在上面。而另一件就是持续付出努力。所谓的持续不是一天两天，一个月两个月，甚至不是一年两年，而是十年二十年，直到尘归尘、土归土。目标、兴趣、方法、激励全用上，就只用在这一点上。即便是完全的SB，起码也可能有所收获吧。最可怕的事情不是拖延症，也不是放弃症，而是根本不知道自己究竟想要的是什么。
那么如果暂时找不到怎么办？乔帮主曾有云，keep looking,don't settle。向前进，别停住脚步。如果不知道自己想要的是什么，就先把手头该做和能做的事情做到极致吧，但别停止思考。边做边想，而不是一味空想，或者埋头苦干不思考。
这也是我需要努力的方向。大家一起努力吧。

今天废话太多，年终了毕竟感慨也多，还是送上福利吧。