从零开始学iOS7开发系列3-我的地盘我做主-Cha14

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。

圣光将赐予我胜利。

欢迎回来~

虽然说看教程其实属于比较低效的学习方式，不过在目前的情况下只能勉强凑合着看吧。但别忘了游戏式互动、教别人学、参与实战和加入学习社区这些更加高效的学习方式。

这一课的内容还是关于Objective-C的理论知识。
只有很少的一点点，如果看得不过瘾，就攒着下次一起看吧。

id 类型

这个id可不是我们常说的用户编号之类的东西。当我们使用[]或objectAtIndex:方法从NSArray中获取一个对象时，就对对象使用了id类型。

在NSArray中可以存储任何类型的对象，但它不能假定这些对象都是NSObject的子类。事实上，在iOS中的确很极少数的对象（之前不是说所有吗？坑爹啊！！！）并非继承自NSObject。
对于此类对象，Objective-C语言提供了一个专属称号：id类型。

id 是Objective-C语言中一个特殊的关键字，它的意思就是“any object”（勉强翻译成任意对象）。id和NSObject有点类似，不过它没有假定自己可能是任何一种类型的对象-甚至没有说自己是NSObject对象。 id没有任何的方法，属性或实例变量，它是一个百分百的“裸奔”对象。

```
id o = @“Hello,world”;
NSLog(@“The text is: %@”,o);
```

我们经常使用id来描述遵从某个特殊协议的对象，即便对它所属的类一无所知。这也是之前对代理对象所做的事情：

```
@property(nonatomic,weak) id <ItemDetailViewControllerDelegate> delegate;
```

这里的id <ItemDetailViewControllerDelegate>意思是：“我只需要知道这个对象会实现ItemDetailViewControllerDelegate这个协议，至于它究竟是神马类，跟我无关。”

当然，我们也可以这样来写上面的这行代码：

```
@property(nonatomic,weak) NSObject <ItemDetailViewControllerDelegate> *delegate;
```

这样写的话也美问题，不过习惯上还是用id。

注意到我们在使用id时不需要在变量前面加上*星号，因为id类型的变量就暗含着它是个指针。

id还有其它的一些特殊规则。我们可以向id类型变量发送任何消息，但编译器不会对此发出警告。虽然这个功能看起来很强大，但同时也是有潜在危险的。因为没有对对象的类型进行严格限制，很容易在程序中向错误的对象发送错误的消息，然后iOS就会对此相当不满，然后应用就崩溃了。

比如说今天是情人节送礼物的好日子，结果你手一抖向前任发送了情人节快乐的微信，结果你的现任还不小心看到了，然后，然后就没有然后了。。。当然，根据弗洛伊德的潜意识论，这个所谓的手一抖其实是内心深处的一抖，只是你一直在压抑这份感情而已~ 你的现任显然是深知爱情心理学的，所以手抖需谨慎。

此外，对于id类型的对象我们无需进行cast（转换）。

```
NSString *s = [someArray objectAtIndex:3];

//or
NSString *s = someArray[3];
```

不过如果返回的是NSObject *类型而不是id类型，那么就必须这么写：

```
NSString *s = (NSString *)[someArray objectAtIndex:3];

//or
NSString *s = (NSString *)someArray[3];
```

但因为这里返回的是id类型，我们就可以省掉cast（转换）的工作，让代码看起来更简洁。

顺便提一下，在C++ 11中也有一个类型的概念，就是新引入的auto和decltype。
当然，auto是针对所有开发者的，而decltype则是提供给模板开发者的。

auto并不是说这个变量的类型不定，或者在运行时再确定，而是说这个变量在编译时可以由编译器推导出来，使用auto和decltype只是占位符的作用，告诉编译器这个变量的类型需要编译器推导，借以消除C++变量定义和使用时的类型冗余，从而解放了程序员打更多无意义的字符，避免了手动推导某个变量的类型，甚至有时候需要很多额外的开销才能绕过的类型声明。

更多知识可以参考这里：http://www.cprogramming.com/c++11/c++11-auto-decltype-return-value-after-function.html

之所以要提到C++，是因为C,Objective-C,C++都是最死硬的C系语言。特别是对于游戏开发者来说，掌握C,C++和OpenGL的知识是必不可少的。

虽然这个系列教程主要以Objective-C为基础，但还是那句话，在编程入门教程中学哪种语言不重要，重要的是在这个过程中知道如何学习，需要学习什么，学的这些东西有什么用，为什么要这么用。

好了，这部分的理论知识就到此结束了。
希望你现在对Objective-C中的对象和类有更深入的认识。

当然，掌握某种知识和技能的top3方式就是实战，所以从下一课开始让我们回到代码部分吧。

发送福利时间：