面试笔试都是必考语法知识的。请认真复习和深入研究OC。

Objective-C

方法和选择器有何不同？(Difference between method and selector?)

- selector是一个方法的名字，method是一个组合体，包含了名字和实现.

Core Foundation的内存管理

- 凡是带有Create、Copy、Retain等字眼的函数，创建出来的对象，都需要在最后做一次release
- 比如CFRunLoopObserverCreate release函数：CFRelease(对象);

malloc和New的区别

- new 是c++中的操作符，malloc是c 中的一个函数
- new 不止是分配内存，而且会调用类的构造函数，同理delete会调用类的析构函数，而malloc则只分配内存，不会进行初始化类成员的工作，同样free也不会调用析构函数
- 内存泄漏对于malloc或者new都可以检查出来的，区别在于new可以指明是那个文件的那一行，而malloc没有这些信息。
- new 和 malloc效率比较
- new可以认为是malloc加构造函数的执行。
- new出来的指针是直接带类型信息的。

你是否接触过OC中的反射机制？简单聊一下概念和使用

- class反射
- 通过类名的字符串形式实例化对象
  Class class NSClassFromString@(@"student");
  Student *stu = [[class alloc]init];
- 将类名变为字符串
  Class class =[Student class];
  NSString *className = NSStringFromClass(class);
- SEL的反射

- 通过方法的字符串形式实例化方法
  SEL selector = NSSelectorFromClass(@"setName");
  [stu performSelector:selector withObject:@"Mike"];
- 将方法变成字符串
  NSStringFomrSelector(@selector*(setName:))

什么是SEL?如何声明一个SEL?通过那些方法能够,调用SEL包装起来的方法?

- SEL就是对方法的一种包装。包装的SEL类型数据它对应相应的方法地址，找到方法地址就可以调用方法。在内存中每个类的方法都存储在类对象中，每个方法都有一个与之对应的SEL类型的数据，根据一个SEL数据就可以找到对应的方法地址，进而调用方法。

- SEL s1 = @selector(test1); // 将test1方法包装成SEL对象

- SEL s2 = NSSelectorFromString(@"test1"); // 将一个字符串方法转换成为SEL对象

- 调用方法有两种方式：

- 1.直接通过方法名来调用 [person text]

- 2.间接的通过SEL数据来调用 SEL aaa=@selector(text); [person performSelector:aaa];

协议中<NSObject>是什么意思?子类继承了父类,那么子类会遵守父类中遵守的协议吗?协议中能够定义成员变量?如何约束一个对象类型的变量要存储的地址是遵守一个协议对象?

- 遵守NSObject协议
- 会
- 能，但是只在头文件中声明，编译器是不会自动生成实例变量的。需要自己处理getter和setter方法
- id<xxx>

NS/CF/CG/CA/UI这些前缀分别是什么含义

- 函数归属于属于cocoa Fundation框架
- 函数归属于属于core Fundation框架
- 函数归属于属于CoreGraphics.frameworks框架

- 函数归属于属于CoreAnimation.frameworks框架
- 函数归属于属于UIkit框架

面向对象都有哪些特征以及你对这些特征的理解。

- 继承：继承是从已有类得到继承信息创建新类的过程。提供继承信息的类被称为父类（超类、基类）；得到继承信息的类被称为子类（派生类）。 继承让变化中的软件系统有了一定的延续性，同时继承也是封装程序中可变因素的重要手段。

- 封装：封装是把数据和操作数据的方法绑定起来，对数据的访问只能通过已定义的接口。我们在类中编写的方法就是对实现细节的一种封装；我们编写一个类就是对数据和数据操作的封装。可以说，封装就是隐藏一切可隐藏的东西，只向外界提供最简单的编程接口。

- 多态性：多态性是指允许不同子类型的对象对同一消息作出不同的响应。简单的说就是用同样的对象引用调用同样的方法但是做了不同的事情。多态性分为编译时的多态性和运行时的多态性。方法重载（overload）实现的是编译时的多态性（也称为前绑定），而方法重写（override）实现的是运行时的多态性（也称为后绑定）。运行时的多态是面向对象最精髓的东西，要实现多态需要做两件事：1. 方法重写（子类继承父类并重写父类中已有的或抽象的方法）；2. 对象造型（用父类型引用引用子类型对象，这样同样的引用调用同样的方法就会根据子类对象的不同而表现出不同的行为）。

- 抽象：抽象是将一类对象的共同特征总结出来构造类的过程，包括数据抽象和行为抽象两方面。抽象只关注对象有哪些属性和行为，并不关注这些行为的细节是什么。

我们说的Objective-C是动态运行时语言是什么意思？(When we call objective c is runtime language what does it mean?)

- 主要是将数据类型的确定由编译时,推迟到了运行时。这个问题其实浅涉及到两个概念,运行时和多态。

- 简单来说, 运行时机制使我们直到运行时才去决定一个对象的类别,以及调用该类别对象指定方法。

- 多态:不同对象以自己的方式响应相同的消息的能力叫做多态。

- 意思就是假设生物类(life)都拥有一个相同的方法-eat;那人类属于生物,猪也属于生物,都继承了life后,实现各自的eat,但是调用是我们只需调用各自的eat方法。也就是不同的对象以自己的方式响应

了相同的消 息(响应了eat这个选择器)。因此也可以说,运行时机制是多态的基础.

readwrite, readonly, assign, retain, copy, nonatomic属性的作用?

- readwrite 是可读可写特性;需要生成getter方法和setter方法;
- readonly 是只读特性 只会生成getter方法 不会生成setter方法 ,不希望属性在类外改变;
- assign 是赋值特性,setter方法将传入参数赋值给实例变量;仅设置变量时; assign用于简单数据类型,如NSInteger,double,bool;
- retain 表示持有特性,setter方法将传入参数先保留,再赋值,传入参数的引用计数retaincount会+1;
- copy 表示赋值特性,setter方法将传入对象复制一份;需要完全一份新的变量时;
- nonatomic 非原子操作,决定编译器生成的setter getter是否是原子操作;
- atomic表示多线程安全,一般使用 nonatomic。

简述NotificationCenter、KVC、KVO、Delegate?并说明它们之间的区别?(重点)

- KVO（Key-Value- Observing）：一对多, 观察者模式,键值观察机制，它提供了观察某一属性变化的方法，极大简化了代码。
- KVC(Key-Value-Coding)：是键值编码, 一个对象在调用setValue的时候，
  - 检查是否存在相应key的set方法，存在就调用set方法。
  - set方法不存在，就查找_key的成员变量是否存在，存在就直接赋值。
  - 如果_key没找到，就查找相同名称的key，存在就赋值。
  - 如果没有就调用valueForUndefinedkey和setValue：forUndefinedKey。
- Delegate: 通常发送者和接收者的关系是直接的一对一的关系。
  - 代理的目的是改变或传递控制链。允许一个类在某些特定时刻通知到其他类，而不需要获取到那些类的指针。

- 可以减少框架复杂度。消息的发送者(sender)告知接收者(receiver)某个事件将要发生，delegate同意然然后发送者响应事件，delegate机制使得接收者可以改变发送者的行为。

- Notification: 观察者模式, 通常发送者和接收者的关系是间接的多对多关系。 消息的发送者告知接收者事件已经发生或者将要发送，仅此而已，接收者并不能反过来影响发送者的行为。

- 区别

  - 效率肯定是delegate比NSNotification高。
  - delegate方法比notification更加直接，需要关注返回值，所以delegate方法往往包含should这个很传神的词。相反的，notification最大的特色就是不关心结果。所以notification往往用did这个词汇。
  - 两个模块之间联系不是很紧密，就用notification传值，例如多线程之间传值用notificaiton。
  - delegate只是一种较为简单的回调，且主要用在一个模块中，例如底层功能完成了，需要把一些值传到上层去，就事先把上层的函数通过delegate传到底层，然后在底层call这个delegate，它们都在一个模块中，完成一个功能，例如说 NavgationController 从 B 界面到A 点返回按钮(调用popViewController方法) 可以用delegate比较好。

懒加载(What is lazy loading ?)

- 就是懒加载,只在用到的时候才去初始化。也可以理解成延时加载。我觉得最好也最简单的一个列子就是tableView中图片的加载显示了, 一个延时加载, 避免内存过高,一个异步加载,避免线程堵塞提高用户体验

OC有多继承吗?没有的话可以用什么方法替代?

- 多继承即一个子类可以有多个父类,它继承了多个父类的特性。
- Object-c的类没有多继承,只支持单继承,如果要实现多继承的话,可以通过类别和协议的方式来实现。
- protocol（协议）可以实现多个接口,通过实现多个接口可以完成多继承;
- Category（类别）一般使用分类,用Category去重写类的方法,仅对本Category有效,不会影响到其他类与原有类的关系。

分别描述类别(categories)和延展(extensions)是什么?以及两者的区别?继承和类别在实现中有何区别?为什么Category只能为对象添加方法,却不能添加成员变量?

- 类别: 在没有原类.m文件的基础上,给该类添加方法;
- 延展:一种特殊形式的类别,主要在一个类的.m文件里声明和实现延展的作用,就是给某类添加私有方法或是私有变量。
- 两个的区别:
    - 延展可以添加属性并且它添加的方法是必须要实现的。延展可以认为是一个私有的类目。
    - 类别可以在不知道,不改变原来代码的情况下往里面添加新的方法,只能添加,不能删除修改。
    - 并且如果类别和原来类中的方法产生名称冲突,则类别将覆盖原来的方法,因为类别具有更高的优先级。
- 继承可以增加，修改删除方法，添加属性。
- Category只能为对象添加方法,却不能添加成员变量的原因:如果可以添加成员变量,添加的成员变量没有办法初始化

Objective-C有私有方法么?私有变量呢?如多没有的话,有没有什么代替的方法?

- objective-c类里面的方法只有两种,静态方法和实例方法.但是可以通过把方法的声明和定义都放在.m文件中来实现一个表面上的私有方法。有私有变量,可以通过@private来修饰,或者把声明放到.m文件中。在Objective-C中,所有实例变量默认都是私有的, 所有实例方法默认都是公有的

include与#import的区别? #import与@class的区别?

- **import指令是Object-C针对#include的改进版本，#import确保引用的文件只会被引用一次，这样你就不会陷入递归包含的问题中。**
- **import与@class二者的区别在于：**
    - **import会链入该头文件的全部信息，包括实例变量和方法等；而@class只是告诉编译器，其后面声明的名称是类的名称，至于这些类是如何定义的，暂时不用考虑。**
    - 在头文件中一般使用@class来声明这个名称是类的名称,不需要知道其内部的实体变量和方法.
    - 而在实现类里面，因为会用到这个引用类的内部的实体变量和方法，所以需要使用#import来包含这个被引用类的头文件。

- 在编译效率方面，如果你有100个头文件都#import了同一个头文件，或者这些文件是依次引用的，如A–>B, B–>C, C–>D这样的引用关系。当最开始的那个头文件有变化的话，后面所有引用它的类都需要重新编译，如果你的类有很多的话，这将耗费大量的时间。而是用@class则不会。
- 如果有循环依赖关系，如:A–>B, B–>A这样的相互依赖关系，如果使用#import来相互包含，那么就会出现编译错误，如果使用@class在两个类的头文件中相互声明，则不会有编译错误出现。

浅复制(拷贝)和深复制的区别? (Difference between shallow copy and deep copy?)

- 浅复制(copy)：只复制指向对象的指针，而不复制引用对象本身。

- 深复制(mutableCopy)：复制引用对象本身。深复制就好理解了,内存中存在了两份独立对象本身,当修改A时,A_copy不变。

类变量的@protected,@private,@public,@package声明各有什么含义?

变量的作用域不同。

- @protected 该类和子类中访问，是默认的;
- @private 只能在本类中访问;
- @public 任何地方都能访问;
- @package 本包内使用，跨包不可以

Objective-C与C、C++之间的联系和区别?

- Objective-C和C++都是C的面向对象的超集。
- Object与C++的区别主要点：Objective-C是完全动态的，支持在运行时动态类型决议(dynamic typing)，动态绑定(dynamic binding)以及动态装载(dynamic loading)；而C++是部分动态的，编译时静态绑定，通过嵌入类(多重继承)和虚函数(虚表)来模拟实现。
- Objective-C 在语言层次上支持动态消息转发，其消息发送语法为 [object function]； 而且C++ 为 object->function()。 两者的语义也不同，在 Objective-C 里是说发送消息到一个对象上，至于这个对象能不能响应消息以及是响应还是转发消息都不会 crash； 而在 C++ 里是说对象进行了某个操作，如果对象没有这个操作的话，要么编译会报错(静态绑定)，要么程序会 crash 掉的(动态绑定)。

目标-动作机制

- 目标是动作消息的接收者。一个控件，或者更为常见的是它的单元，以插座变量的形式保有其动作消息的目标。
- 动作是控件发送给目标的消息，或者从目标的角度看，它是目标为了响应动作而实现的方法. 程序需要某些机制来进行事件和指令的翻译。这个机制就是目标-动作机制。

Objective-C优点和缺点

- 优点:1.Cateogies 2.Posing 3.动态识别 4.指标计算 5.弹性讯息传递 6.不是一个过度复杂的C衍生语言 7.Objective-C与C++可混合编程

- 缺点:1.不支持命名空間 2.不支持运算符重载 3.不支持多重继承 4.使用动态运行时类型,所有的方法都是函数调用,所以很多编译时优化方法都用不到。(如内联函数等),性能低劣。

C语言的函数调用和oc的消息机制有什么区别？

- 对于C语言，函数的调用在编译的时候会决定调用哪个函数。编译完成之后直接顺序执行。
- OC的函数调用成为消息发送。属于动态调用过程。在编译的时候并不能决定真正调用哪个函数（事实证明，在编译阶段，OC可以调用任何函数，即使这个函数并未实现，只要申明过就不会报错。而C语言在编译阶段就会报错）。只有在真正运行的时候才会根据函数的名称找到对应的函数来调用。

什么是谓词

谓词就是通过NSPredicate给定的逻辑条件作为约束条件，完成对数据的筛选。

```
//定义谓词对象，谓词对象中包含了过滤条件

NSPredicate *predicate = [NSPredicate predicateWithFormat:@"age<%d",30];

//使用谓词条件过滤数组中的元素，过滤之后返回查询的结果

NSArray *array = [persons filteredArrayUsingPredicate:predicate];

//可以使用&&进行多条件过滤

predicate = [NSPredicate predicateWithFormat:@"name='1' && age>40"];

array = [persons filteredArrayUsingPredicate:predicate];

//包含语句的使用

predicate = [NSPredicate predicateWithFormat:@"self.name IN {'1','2','4'} || self.age IN{30,40}"];
```

```
//指定字符开头和指定字符结尾，是否包含指定字符

//name以a开头的

predicate = [NSPredicate predicateWithFormat:@"name BEGINSWITH 'a'"];

//name以ba结尾的

predicate = [NSPredicate predicateWithFormat:@"name ENDSWITH 'ba'"];

//name中包含字符a的

predicate = [NSPredicate predicateWithFormat:@"name CONTAINS 'a'"];

//like进行匹配多个字符

//name中只要有s字符就满足条件

predicate = [NSPredicate predicateWithFormat:@"name like '*s*'"];

//?代表一个字符，下面的查询条件是：name中第二个字符是s的

predicate = [NSPredicate predicateWithFormat:@"name like '?s'"];
```

C与OC混用

处理.m可以识别c和oc，.mm可以识别c c++ oc 但是cpp只能用c/c++

atomic和nonatomic的区别

- atomic提供多线程安全，防止读写未完成的时候被另外一个线程读写，造成数据错误。
- nonatomic在自己管理内存的环境中，解析的访问器保留并自动释放返回值，若指定了nonatomic，那么访问器只是简单的返回这个值。

常见的oc数据类型哪些，和c的基本类型有啥区别

- 常见的：NSInteger CGFloat NSString NSNumber NSArray NSDate
- NSInteger根据32或者64位系统决定本身是int还是long
- CGFloat根据32或者64位系统决定本身是float还是double
- NSString NSNumber NSArray NSDate都是指针类型的对象，在堆中分配内存，c语言中的char int 等都是在栈中分配空间

id和nil代表什么

- id类型的指针可以指向任何OC对象
- nil代表空值（空指针的值，0）

nil和NULL的区别?

- 从oc的官方语法上看，nil表示对象的指针 即对象的引用为空
- null表示指向基础数据类型变量 即c语言变量的指针为空
- 在非arc中 两个空可以互换，但是在arc中 普通指针和对象引用被严格限制，不能互换

nil、Nil、NULL和NSNull区别

- nil和C语言的NULL相同，在objc/objc.h中定义。nil表示Objective-C对象的值为空。在C语言中，指针的空值用NULL表示。在Objective-C中，nil对象调用任何方法表示什么也不执行，也不会崩溃。

- Nil:那么对于我们Objective-C开发来说，Nil也就代表((void *)0)。但是它是用于代表空类的. 比如：Class myClass = Nil;

- NULL: 在C语言中，NULL是无类型的，只是一个宏，它代表空. 这就是在C/C++中的空指针。对于我们Objective-C开发来说，NULL就表示((void*)0).

- NSNull:NSNull是继承于NSObject的类型。它是很特殊的类，它表示是空，什么也不存储，但是它却是对象，只是一个占位对象。使用场景就不一样了，比如说服务端接口中让我们在值为空时，传空。NSDictionry *parameters = @{@"arg1" : @"value1",@"arg2" : arg2.isEmpty ? [NSNull null] : arg2};

- NULL、nil、Nil这三者对于Objective-C中值是一样的，都是(void *)0，那么为什么要区分呢？又与NSNull之间有什么区别：

- NULL是宏，是对于C语言指针而使用的，表示空指针

- nil是宏，是对于Objective-C中的对象而使用的，表示对象为空

- Nil是宏，是对于Objective-C中的类而使用的，表示类指向空

- NSNull是类类型，是用于表示空的占位对象，与JS或者服务端的null类似的含意

向一个nil对象发送消息会发生什么?

- 向nil发送消息是完全有效的——只是在运行时不会有任何作用。
- 如果一个方法返回值是一个对象，那么发送给nil的消息将返回0(nil)
- 如果方法返回值为指针类型，其指针大小为小于或者等于sizeof(void*)，float，double，long double 或者long long的整型标量，发送给nil的消息将返回0。

-如果方法返回值为结构体，正如在《Mac OS X ABI 函数调用指南》，发送给nil的消息将返回0。结构体中各个字段的值将都是0。其他的结构体数据类型将不是用0填充的。

- 如果方法的返回值不是上述提到的几种情况，那么发送给nil的消息的返回值将是未定义的。

self.和self->的区别

- self.是调用get或者set方法
- self是当前本身，是一个指向当前对象的指针
- self->是直接访问成员变量

类方法和实例方法的本质区别和联系

类方法	实例方法
属于类对象	属于实例对象
只能类对象调用	实例对象调用
self是类对象	self是实例对象
类方法可以调用其他类方法	实例方法可以调用实例方法
类方法不能访问成员变量	实例方法可以访问成员变量
类方法不能直接调用对象方法	实例方法可以调用类方法

_block/weak修饰符区别

- _block在arc和mrc环境下都能用，可以修饰对象，也能修饰基本数据类型
- _weak只能在arc环境下使用，只能修饰对象(NSString)，不能修饰基本数据类型(int)
- _block对象可以在block中重新赋值，_weak不行。

写一个NSString类的实现

NSString *str = [[NSString alloc]initWithCString:nullTerminatedCString encoding:encoding];

为什么标准头文件都有类似以下的结构?

ifndef __INCvxWorksh

define __INCvxWorksh

ifdef __cplusplus

extern "C" {

endif

ifdef __cplusplus

}

endif

endif

显然，头文件中的编译宏“#ifndef __INCvxWorksh、#define __INCvxWorksh、#endif” 的作用是防止该头文件被重复引用

init和initwithobject区别（语法）？

- 后者给属性赋值

@property的本质是什么？ivar、getter、setter是如何生成并添加到这个类中的?

```
@property的本质：
@property = ivar（实例变量） + getter（取方法） + setter（存方法）
“属性” (property)有两大概念：ivar（实例变量）、存取方法（access method = getter + setter）
ivar、getter、setter如何生成并添加到类中：
这是编译器自动合成的，通过@synthesize关键字指定，若不指定，默认为@synthesize propertyName = _propertyName;若手
现在编译器已经默认为我们添加@synthesize propertyName = _propertyName;因此不再需要手动添加了，除非你真的要改成员
生成getter方法时，会判断当前属性名是否有_，比如声明属性为@property (nonatomic, copy) NSString *_name;那么所生
不过，命名都要有规范，是不允许声明属性是使用_开头的，不规范的命名，在使用runtime时，会带来很多的不方便的。
```

这个写法会出什么问题：@property (copy) NSMutableArray *array;

- 没有指明为nonatomic，因此就是atomic原子操作，会影响性能。该属性使用了同步锁，会在创建时生成一些额外的代码用于帮助编写多线程程序，这会带来性能问题，通过声明nonatomic可以节省这些虽然很小但是不必要额外开销。在我们的应用程序中，几乎都是使用nonatomic来声明的，因为使用atomic并不能保证绝对的线程安全，对于要绝对保证线程安全的操作，还需要使用更高级的方式来处理，比如NSSpinLock、@syncronized等
- 因为使用的是copy，所得到的实际是NSArray类型，它是不可变的，若在使用中使用了增、删、改操作，则会crash

@protocol和category中如何使用 @property

```
在protocol中使用@property只会生成setter和getter方法声明，我们使用属性的目的是希望遵守我协议的对象能实现该属
category使用@property也是只会生成setter和getter方法的声明，如果我们真的需要给category增加属性的实现，需要借
objc_setAssociatedObject
objc_getAssociatedObject
```

@property中有哪些属性关键字?

1. 原子性 （atomic，nonatomic）

2. 读写（readwrite, readonly）

3. 内存管理（assign, strong, weak, unsafe_unretained,copy）

4. getter、setter

isa指针问题

- isa：是一个Class 类型的指针. 每个实例对象有个isa的指针,他指向对象的类，而Class里也有个isa的指针, 指向meteClass(元类)。元类保存了类方法的列表。当类方法被调用时，先会从本身查找类方法的实现，如果没有，元类会向他父类查找该方法。同时注意的是：元类（meteClass）也是类，它也是对象。元类也有isa指针,它的isa指针最终指向的是一个根元类(root meteClass).根元类的isa指针指向本身，这样形成了一个封闭的内循环。

如何访问并修改一个类的私有属性?

- 一种是通过KVC获取
- 通过runtime访问并修改私有属性

如何为 Class 定义一个对外只读对内可读写的属性?

在头文件中将属性定义为readonly,在.m文件中将属性重新定义为readwrite

Objective-C 中，meta-class 指的是什么?

meta-class 是 Class 对象的类,为这个Class类存储类方法,当一个类发送消息时,就去这个类对应的meta-class中查找那个消息,每个Class都有不同的meta-class,所有的meta-class都使用基类的meta-class(假如类继承NSObject,那么他所对应的meta-class也是NSObject)作为他们的类

Objective-C 的class是如何实现的？Selector是如何被转化为 C 语言的函数调用的?

- 当一个类被正确的编译过后，在这个编译成功的类里面，存在一个变量用于保存这个类的信息。我们可以通过[NSClassFromString]或[obj class]。这样的机制允许我们在程序执行的过程当中，可以Class来得到对象的类，也可以在程序执行的阶段动态的生成一个在编译阶段无法确定的一个对象。 （isa指针）

- @selector()基本可以等同C语言的中函数指针,只不过C语言中，可以把函数名直接赋给一个函数指针，而Object-C的类不能直接应用函数指针，这样只能做一个@selector语法来取.

```
@interface foo
-(int)add:int val;
@end

SEL class_func ; //定义一个类方法指针
class_func = @selector(add:int);
```

- @selector是查找当前类的方法，而[object @selector(方法名:方法参数..)] ;是取object对应类的相应方法.

- 查找类方法时，除了方法名,方法参数也查询条件之一.

- 可以用字符串来找方法 SEL　变量名　=　NSSelectorFromString(方法名字的字符串);

- 可以运行中用SEL变量反向查出方法名字字符串。NSString　*变量名　=　NSStringFromSelector(SEL参数);

- 取到selector的值以后，执行seletor。 SEL变量的执行.用performSelecor方法来执行.
[对象　performSelector:SEL变量　withObject:参数1　withObject:参数2];

对于语句NSString *obj = [[NSData alloc] init];，编译时和运行时obj分别是什么类型?

- 编译时是NSString类型，运行时是NSData类型.

@synthesize和@dynamic分别有什么作用?

```
- @property有两个对应的词，一个是 @synthesize，一个是 @dynamic。如果 @synthesize和 @dynamic都没写，那么默认的
- @synthesize 的语义是如果你没有手动实现 setter 方法和 getter 方法，那么编译器会自动为你加上这两个方法。
- @dynamic 告诉编译器：属性的 setter 与 getter 方法由用户自己实现，不自动生成。（当然对于 readonly 的属性只需提
```

NSString 的时候用copy和strong的区别?

```
OC中NSString为不可变字符串的时候，用copy和strong都是只分配一次内存，但是如果用copy的时候，需要先判断字符串是否是
```

NSArray、NSSet、NSDictionary与NSMutableArray、NSMutableSet、NSMutableDictionary的特性和作用（遇到copy修饰产生的变化）

- 特性:
- NSArray表示不可变数组，是有序元素集，只能存储对象类型，可通过索引直接访问元素，而且元素类型可以不一样，但是不能进行增、删、改操作；NSMutableArray是可变数组，能进行增、删、改操作。通过索引查询值很快，但是插入、删除等效率很低。
- NSSet表示不可变集合，具有确定性、互异性、无序性的特点，只能访问而不能修改集合；NSMutableSet表示可变集合，可以对集合进行增、删、改操作。集合通过值查询很快，插入、删除操作极快。
- NSDictionary表示不可变字典，具有无序性的特点，每个key对应的值是唯一的，可通过key直接获取值；NSMutableDictionary表示可变字典，能对字典进行增、删、改操作。通过key查询值、插入、删除值都很快。
- 作用:
- 数组用于处理一组有序的数据集，比如常用的列表的dataSource要求有序，可通过索引直接访问，效率高。
- 集合要求具有确定性、互异性、无序性，在iOS开发中是比较少使用到的，笔者也不清楚如何说明其作用
- 字典是键值对数据集，操作字典效率极高，时间复杂度为常量，但是值是无序的。在ios中，常见的JSON转字典，字典转模型就是其中一种应用。

请把字符串2015-04-10格式化日期转为NSDate类型

```
NSString *timeStr = @"2015-04-10";
NSDateFormatter *formatter = [[NSDateFormatter alloc] init];
formatter.dateFormat = @"yyyy-MM-dd";
formatter.timeZone = [NSTimeZone defaultTimeZone];
NSDate *date = [formatter dateFromString:timeStr];
// 2015-04-09 16:00:00 +0000
NSLog(@"%@", date);
```

在一个对象的方法里：self.name=@object;和name=@object有什么不同

- 这是老生常谈的话题了，实质上就是问setter方法赋值与成员变量赋值有什么不同。通过点语法self.name实质上就是[self setName:@object];。而name这里是成员变量，直接赋值。
  一般来说，在对象的方法里成员变量和方法都是可以访问的，我们通常会重写Setter方法来执行某些额外的工作。比如说，外部传一个模型过来，那么我会直接重写Setter方法，当模型传过来时，也就是意味着数据发生了变化，那么视图也需要更新显示，则在赋值新模型的同时也去刷新UI。这样也不用再额外提供其他方法了。

怎样使用performSelector传入3个以上参数，其中一个为结构体

```
- (id)performSelector:(SEL)aSelector;
- (id)performSelector:(SEL)aSelector withObject:(id)object;
- (id)performSelector:(SEL)aSelector withObject:(id)object1 withObject:(id)object2;
```

因为系统提供的performSelector的api中，并没有提供三个参数。因此，我们只能传数组或者字典，但是数组或者字典只有存入对象类型，而结构体并不是对象类型，那么怎么办呢？
没有办法，我们只能通过对象放入结构作为属性来传过去了：

```
ypedef struct HYBStruct {
  int a;
  int b;
} *my_struct;

@interface HYBObject : NSObject

@property (nonatomic, assign) my_struct arg3;
@property (nonatomic, copy)  NSString *arg1;
@property (nonatomic, copy) NSString *arg2;

@end

@implementation HYBObject

// 在堆上分配的内存，我们要手动释放掉
- (void)dealloc {
```

```
    free(self.arg3);
}

@end
```

测试：

```
my_struct str = (my_struct)(malloc(sizeof(my_struct)));
str->a = 1;
str->b = 2;
HYBObject *obj = [[HYBObject alloc] init];
obj.arg1 = @"arg1";
obj.arg2 = @"arg2";
obj.arg3 = str;

[self performSelector:@selector(call:) withObject:obj];

// 在回调时得到正确的数据的
- (void)call:(HYBObject *)obj {
    NSLog(@"%d %d", obj.arg3->a, obj.arg3->b);
}
```

objc中向一个对象发送消息[obj foo]和objc_msgSend()函数之间有什么关系?

```
实际上，编译器在编译时会转换成objc_msgSend，大概会像这样：
((void (*)(id, SEL))(void)objc_msgSend)((id)obj, sel_registerName("foo"));
也就是说，[obj foo];在objc动态编译时，会被转换为：objc_msgSend(obj, @selector(foo));这样的形式，但是需要根据具
```

下面的代码输出什么?

```
@implementation Son : Father

- (id)init {
    self = [super init];
    if (self) {
        NSLog(@"%@", NSStringFromClass([self class]));
        NSLog(@"%@", NSStringFromClass([super class]));
    }
    return self;
}

@end

// 输出
NSStringFromClass([self class]) = Son
NSStringFromClass([super class]) = Son
这个题目主要是考察关于Objective-C中对self和super的理解。我们都知道：self是类的隐藏参数，指向当前调用方法的这个类
很多人会想当然的认为“super和self类似，应该是指向父类的指针吧！”。这是很普遍的一个误区。其实 super是一个 Magic Ke
上面的例子不管调用[self class]还是[super class]，接受消息的对象都是当前 Son *xxx 这个对象。
当使用self调用方法时，会从当前类的方法列表中开始找，如果没有，就从父类中再找；而当使用super时，则从父类的方法列表
```

若一个类有实例变量NSString *_foo，调用setValue:forKey:时，可以以foo还是_foo作为key?

- 两者都可以。

什么时候使用NSMutableArray，什么时候使用NSArray?

- 当数组在程序运行时，需要不断变化的，使用NSMutableArray，当数组在初始化后，便不再改变的，使用NSArray。需要指出的是，使用NSArray只表明的是该数组在运行时不发生改变，即不能往NSAarry的数组里新增和删除元素，但不表明其数组内的元素的内容不能发生改变。NSArray是线程安全的，NSMutableArray不是线程安全的，多线程使用到NSMutableArray需要注意。

类NSObject的那些方法经常被使用?

- NSObject是Objetive-C的基类，其由NSObject类及一系列协议构成。
- 其中类方法alloc、class、 description 对象方法init、dealloc、–performSelector:withObject:afterDelay:等经常被使用

什么是简便构造方法?

- 简便构造方法一般由CocoaTouch框架提供，如NSNumber的 + numberWithBool: + numberWithChar: + numberWithDouble: + numberWithFloat: + numberWithInt:
- Foundation下大部分类均有简便构造方法，我们可以通过简便构造方法，获得系统给我们创建好的对象，并且不需要手动释放。

什么是构造方法，使用构造方法有什么注意点。

什么是构造方法：构造方法是对象初始化并一个实例的方法。

构造方法有什么用： 一般在构造方法里 对类进行一些初始化操作
注意点：方法开头必须以init开头，接下来名称要大写 例如 initWithName ， initLayout

创建一个对象需要经过那三个步骤?

- 开辟内存空间
- 初始化参数
- 返回内存地址值

Get方法的作用是什么?

Get方法的作用：为调用者返回对象内部的成员变量

Set方法的作用是什么？Set方法的好处?

- Set方法的作用：为外界提供一个设置成员变量值的方法。
- Set方法的好处：
  - 不让数据暴露在外，保证了数据的安全性
  - 对设置的数据进行过滤

结构体当中能定义oc对象吗?

不能, 因为结构体当中只能是类型的声明不能进行分配空间

点语法本质是什么,写一个点语法的例子,并写上注释

- 点语法的本质是方法的调用，而不是访问成员变量，当使用点语法时，编译器会自动展开成相应的方法。切记点语法的本质是转换成相应的set和get方法，如果没有set和get方法，则不能使用点语法。
- 例如有一个Person类 通过@property定义了name和age属性,再提供了一个run方法。
- Person *person = [Person new];
- person.name=@”itcast”;//调用了person的setName方法
- int age = person.age; // 调用了person的age方法
- person.run //调用了person的run方法

id类型是什么，instancetype是什么，有什么区别?

- id类型：万能指针，能作为参数，方法的返回类型。
- instancetype：只能作为方法的范围类型，并且返回的类型是当前定义类的类类型。

成员变量名的命名以下划线开头的好处?

- 与get方法的方法名区分开来;
- 可以和一些其他的局部变量区分开来，下划线开头的变量，通常都是类的成员变量。

这段代码有什么问题吗:

```
 @implementation Person
- (void)setAge:(int)newAge {
self.age = newAge; }
@end
会死循环,会重复调用自己!self.age 改为_age即可;
```

并且书写不规范:setter方法中的newAge应该为age

截取字符串”20 | http://www.baidu.com”中,”|”字符前面和后面的数据,分别输出它们。

```
   NSString * str = @"20 | http://www.baidu.com";
NSArray *array = [str componentsSeparatedByString:@"|"]; //这是分别输出的截取后的字符串
for (int i = 0; i<[array count]; ++i) { NSLog(@"%d=%@",i,[array objectAtIndex:i]);
}
```

写一个完整的代理,包括声明,实现

```
//创建
@protocol MyDelagate
@required
-(void)eat:(NSString *)foodName;
@optional
-(void)run;
@end

//声明
@interface person: NSObject< MyDelagate>

//实现
@implementation person
-(void)eat:(NSString *)foodName;
{ NSLog(@"吃:%@!",foodName);}
-(void)run
{ NSLog(@"run!");}
@end
```

isKindOfClass、isMemberOfClass、selector作用分别是什么

- isKindOfClass,作用是,某个对象属于某个类型或者继承自某类型
- isMemberOfClass:某个对象确切属于某个类型
- selector:通过方法名,获取在内存中的函数的入口地址

请分别写出SEL、id、@的意思?

- SEL是“selector”的一个类型,表示一个方法的名字-------就是一个方法的入口地址

- id是一个指向任何一个继承了Object(或者NSObject)类的对象。需要注意的是id是一个指针,所以在使用id 的时候不需要加*。

- @:OC中的指令符

unsigned int 和int 有什么区别。假设int长度为65535，请写出unsigned int与int的取值范围

int:基本整型，当字节数为2时 取值范围为-32768~32767，当字节数为4时 取值范围
负的2的31次方 到 2的31次方减1
unsigned int：无符号基本整型，当字节数为2时 取值范围为0~65535，当字节数为4时 取值范围为0到2的32次方减1

Foundation对象与Core Foundation对象有什么区别

- Foundation对象是OC的，Core Foundation对象是C对象
- 数据类型之间的转换
    - ARC:__bridge_retained（持有对象所有权,F->CF）、__bridge_transfer（释放对象有所权CF->F）
    - 非ARC: __bridge

编写一个函数，实现递归删除指定路径下的所有文件。

```
+ (void)deleteFiles:(NSString *)path;{
    // 1.判断文件还是目录
    NSFileManager * fileManger = [NSFileManager defaultManager];
    BOOL isDir = NO;
    BOOL isExist = [fileManger fileExistsAtPath:path isDirectory:&isDir];
    if (isExist) {
        // 2. 判断是不是目录
        if (isDir) {
        NSArray * dirArray = [fileManger contentsOfDirectoryAtPath:path error:nil];
            NSString * subPath = nil;
            for (NSString * str in dirArray) {
                subPath  = [path stringByAppendingPathComponent:str];
                BOOL issubDir = NO;
                 [fileManger fileExistsAtPath:subPath isDirectory:&issubDir];
                [self deleteFiles:subPath];
            }


        }else{
            NSLog(@"%@",path);
            [manager removeItemAtPath:filePath error:nil];
        }
    }else{
        NSLog(@"你打印的是目录或者不存在");
    }
```

```
}
```