



Cocos2d-x  
高薪就业班

你和大牛的距离，就差这一步

开放式多业务平台TOMP发布会  
奔跑吧云计算

您所在的位置：移动开发 > 热门推荐 > iOS面试攻略上篇:Objective-C关键字和概念

# iOS面试攻略上篇:Objective-C关键字和概念

2013-07-10 11:31 佚名 oschina 字号: T | T

收藏 +

还在面试的时候感觉自己像一只无头苍蝇么? 本文为大家整理了一系列iOS面试题，其中包括一些Objective-C的关键字和概念，少编也祝各位马到功成。

AD: 干货来了,不要等! WOT2015 北京站演讲PPT开放下载!

欢迎查看下篇: [iOS面试攻略下篇:Objective-C面试题和基本概念](#)

@

看到这个关键字，我们就应该想到，这是Object-C对C 语言的扩展，例如@interface XXX。

@interface

声明类

@implementation

实现类

@protocol

声明协议

@optional

与@protocol配合使用，说明协议中的某个或者某几个方法可以不实现

@required

与@protocol配合使用，说明协议中的某个方法或者某几个方法必须实现

@end

与@interface ,@implementation,@protocol配合使用，代表声明或者实现结束

@encode

@encode为编译器宏，它可以将类型转换为相应的字符串。

id

id是指向Objective-C类对象的指针，它可以声明为任何类对象的指针，当在Objective-C中使用id时，编译器会假定你知道，id指向哪个类的对象。与void\*是不同的是，void\*编译器不知道也不假定指向任何类型的指针。

nil

定义为一个常量，如果一个指针的值为nil,代表这个指针没有指向任何对象。

self

在Objective-C中，关键字self与c++中this是同一概念，就是类对象自身的地址，通过self可以调用自己的实例变量和方法



51CTO微信公众号  
weixin51cto



扫描下载学院iPad端  
缓存视频离线看

菜鸟变大师  
大牛帮你来规划!

51CTO学院学习路线图上线了

● 学习思路明确有计划 ● 记录学习进度效率高



专题 Android Studio上手攻略

专题 开发环境 Android Studio 上手攻略 51CTO 移动开发

既然强大的Android Studio来了，有什么理由不去用呢?

iOS新语言swift上手指南

MIUI 6 测评：细节的量变

文章排行	24小时	本周	本月
21个免费的UI界面设计工具、资源及网站			
在Eclipse下搭建Android开发环境教程			
人人都是开发者：5款傻瓜式APP开发工具			
三大移动Web开发框架哪个适合你?			
哪个市场最挣钱，腾讯告诉你！			
阿里：这是全世界最大的打假团队			
Android应当从iOS处窃取的五大最佳功能			
iOS 8出色的跨应用通信效果：解读Actio			
看库克如何干掉Android			
为什么说2015年将是微服务架构元年?			

热点专题

更多>>



iOS开发之常见疑难问题

在iOS开发过程中，尤其是对于新手来说，都会遇到或多



Web App开发最佳实践

Web App开发中会面临越来越“重”的问题，如果在开始

Android开发常见“疑

Super

当子类需要调用父类的方法时，会用到Super关键字。Super指向的是父类的指针，子类重写父类的方法时，调用父类的方法是一个比较好的习惯。因为当我们不知道父类在该方法中实现的功能时，如果不调用父类的方法，有可能我们重写的方法会失去该功能，这是我们所不愿意看到的情况。

NSNull

NSNull是没有的意思，如果一个字典的值为NSNull,那说明与该值对应的Key是没有值的，例如Key为address，说明与address对应的是值是没有。

self super class public protected private id

[self class] [super class] selector

objective-c runtime reference

标准用法

self = [super init]

new

1 Objective-C有一个特性，就是可以把类当成对象来发送消息，这种用法通常用于新建对象时，例如 XXX \*object = [XXX new];

类方法 +

如果想声明属于类而不属于类对象的方法，用+。+用来修饰类的方法，使用+修饰的类方法，是整个类的方法，不属于哪一个类对象，这与C++中的static在类中使用的概念一样。

%@

在NSLog中，使用%@表示要调用对象的description方法。

概念

类

是一种结构，它表示对象的类型，就像int与 char 一样，也可以声明类的变量(对象)

实例化

为类的对象分配内存和初始化，达到可以使用该类对象的目的。

对象(实例)

类的实例化后的产物

消息

在Object-C中，类的对象执行的操作，是通过给该类或者该类对象发送消息实现，如：  
[object func]；就是给object对象发送 func消息，类似C++中的方法调用。给object对象发送func消息后，object对象查询所属类的func方法执行。

方法调度

当向一个对象发送消息时(调用方法)，这个方法是怎么被调用的呢？这就依赖于方法高度程序，方法调度程序查找的方法如下：

在本类的方法中，找被调用的方法，如果找到了，就调用，如果找不到被沿着继承路径去查找，从哪个类找到，就调用哪个类的方法，如果到最根上的类还是没有找到，那编译就会出错。

继承与复合

在Objective-C中支持继承，但只是支持单一继承(有且只有一个父类有)，如果想使用多继承的特性，可以使用分类和协议技术。



作为Android开发者，最头疼是什么？相信大家会异口同

热点标签

iOS开发 Android开发 Symbian开发 MeeGo开发  
BlackBerry开发 Windows Phone开发 Phone Club  
Android 4.0 webOS 3.0 智能手机 软件下载

[点击这里查看样刊](#)

[立即订阅](#)

全站热点



“棱镜”惊醒中国信息安全最弱神经



云计算时代 需要怎样的加密保护

Essential C++中文版  
iOS教程之ASIHttpRequest完全攻略  
独家提供:最具有价值的Hadoop Summit  
开发频道2013年7月第4周重点内容推荐  
微软Azure助推移动互联网云适配

读书



精通JavaScript动态网页编程(实例版)  
本书通过大量实例代码，以ECMA-262版本3为基础，结合JavaScript 1.5和JavaScript 5.5，由浅入深、循序渐进地介绍了JavaScript知

基于Project2003的项目管理  
Tomcat与Java Web开发技术详解  
网管第一课——计算机网络原理  
网管第一课——网络组建与管理

博文推荐

[更多>>](#)



nowpaper  
Windows  
Phone专家



himi  
Android开  
发专家

自制节目树立视频行业“去竞争化”优  
屈服？Google中国问题似乎解决有望  
C#正则表达式编程（三）： Match类和G  
Windows-Server2008下加强系统安全性

最新热帖

[更多>>](#)

面试之自我救赎  
30岁之前的目标，51CTO的Fans，共勉  
CCNA重点知识点  
大公司里的草还是小公司里的宝？  
价值上万的SQL学习视频【初学者的福

继承是is-a,复合是has-a。复合是通过包含指向对象的指针实现的,严格意义上讲,复合是针对对象间来说,对于基本数据类型来说,它们被认为是对象的一部分。

### 装箱与拆箱

由于NSArray,NSDictionary等类不能直接存储基本数据类型,所以要想在NSArray \ NSDictionary中使用基本数据类型,就得使用装箱与拆箱。

在Objective-C中,可以使用NSNumber和NSValue来实现对数据类型的包装,NSNumber可以实现对基本数据类型的包装,NSValue可以实现对任意类型数据的包装。

将基本类型封装成对象叫装箱,从封装的对象中提取基本类型叫拆箱(取消装箱),其它语言如Java原生支持装箱与拆箱,Objective-C不支持自动装箱与拆箱,如果需要得需要自己来实现装箱与拆箱。

### 存取方法

在使用类对象的实例变量(成员数据)时,不要直接使用对象中的实例,要使用存以方法来获取或者修改实例,既setter和getter,在Cocoa中,存取方法有命名习惯,我们得符合这种习惯,以便于与其它团队成员合作.setter方法是修改或者设置实例值,命名习惯为set+实例名,例有一个类有path实例变量,那setter命名为setPath,getter命名为Path,为什么不是getPath,因为get在Cocoa中有特殊的含义,这个含义就是带有get的方法就意味着这个方法通过形参指针(传入函数的参数指针)来返回值。我们要遵守这个命名习惯或者说规则。

在Objective-C 2.0中加入了@property和@synthesize来代替setter和getter,这两个关键字为编译器指令。还有点表达式,存取类成员的值时,可以使用点表达式。

Object.attribute,当点表达式在=号左边时,调用的是setter方法,在=号右边时,调用的是getter方法。

@property 语法为:@property (参数) 类型 变量名.

在这里主要说明一下参数.

参数分为三种:

第一种:读写属性包括(readonly/readwrite/)

第二种:setter属性(assign,copy,retain),assign是简单的赋值,copy是释放旧成员变量,并新分配内存地址给成员变量,将传入参数内容复制一份,给成员变量。retain是将传入 参数引用计数加1,然后将原有的成员变量释放,在将成员变量指向该传入参数。

第三种:与多线程有关(atomic, nonatomic).当使用多线程时,使用atomic,在不使用多线程时使用nonatomic

### 对象创建与初始化

在Objective-C中创建对象有两种方法,一种是[类 new];另一种是[[类 alloc] init].这两种方法是等价的,但按惯例来讲,使用[[类 alloc] init];

alloc操作是为对象分配内存空间,并将对象的数据成员都初始, int 为0, BOOL 为NO, float 为0.0等。

初始化,默认的初始化函数为init,init返回值为id,为什么回返回id呢,因为要实现链式表达式,在Objective-C中叫嵌套调用。

为什么要嵌套调用? ? 因为初始化方法init返回值可能与alloc返回的对象不是同一个? 为什么会发生这种情况? 基于类簇的初始化,因为init可以接受参数,在init内部有可能根据不同的参数来返回不同种类型的对象,所以最会发生上面说的情况。

在初始化时,建议使用if (self = [super init])

### 便利初始化

当一个类需要根据不同的情况来初始化数据成员时，就需要便利初始化函数，与init初始化不同的是，便利初始化函数有参数，参数个数可以有1到N个，N是类数据成员个数。

指定初始化函数：什么是指定初始化函数？在类中，某个初始化函数会被指定为指定的初始化函数，确定指定初始化函数的规则是初始化函数中，参数最多的为指定初始化函数，

其它未被指定为指定初始化函数的初始化函数要调用指定初始化函数来实现。对于该类的子类也是一样，只要重写或者直接使用父类的指定初始化函数。上述文字有些绕，来个例子吧

```
@interface A{

int x;

int y;

}

-(id) init;

-(id) initWithX:(int) xValue;

-(id) initWithY:(int) yValue;

-(id) initWithXY:(int) xValue

yVal:(int) yValue;

@end
```

这里initWithXY被确定为指定初始化函数。

```
-(id) initWithXY:(int) xValue

yVal:(int) yValue{

if (self = [super init]){

x = xValue;

y = yValue;

}

return self;

}

-(id) init{

if (self = self initWithXY:10

yVal:20){

}

return self;

}

.....

@interface B: A{

int z;

}

-(jd) initWithXY .....;

@end
```

```
@implementation B

-(id) initWithXY:(int) xValue

yVal:(int) yValue{

if (self = [super initWithXY:10

yVal=20]){

z= 40;

}

return self;

}

@end
```

### 自动释放池

内存管理是软件代码中的重中之重，内存管理的好坏，直接影响着软件的稳定性。在Cocoa中，有自动释放池，这类似于C++中的智能指针。

NSObject有一个方法是autorelease，当一个对象调用这个方法时，就会将这个对象放入到自动释放池中。

drain,该方法是清空自动释放池，不是销毁它。drain方法只适用于Mac OS X 10.4以上的版本，在我们写的代码中要使用release，release适用于所有版本。

自动释放池是以栈的方式实现，当创建一个自动释放池A时，A被压入栈顶，这时将接入autorelease消息的对象放入A自动释放池，这时创建一个新的B自动释放池，B被压入栈顶，创建完成后删除B,这个接收autorelease消息的对象依然存在，因为A自动释放池依然存在。

### 引用计数

每个对象都有一个与之相应的整数，称它为引用计数，当该引用计数为0时，Objective-C自动向该对象发送dealloc,以销毁该对象，与该引用计数相关的方法(消息)有下面几个

1 增加引用计数：通过alloc,new,copy创建一个对象时，该对象的引用计数加1(其实就是1，因为之前为0)

2 增加引用计数: retain

3 减少引用计数: release

局部分配内存(临时对象):

1 如果使用alloc,new,copy创建对象，则需要主动调用对象的release方法

2 如果使用非alloc,new,copy创建对象，我们认为该对象引用计数为1，并已经加入了自动释放池，我们不需要主动的调用对象的release方法。

拥有对象(在类中以成员的方法存在):

1 如果使用alloc,new,copy创建对象，则需要 dealloc方法中，释放该对象

2 如果使用非alloc,new,copy创建对象，则在拥有该对象时，保留该对象(执行retain方法)，在dealloc方法中，释放该对象。

### dealloc

当对象的引用计数为0时，Objective-C会自动发送对象的dealloc消息(自动调用对象的dealloc方法，类似于C++的析构函数)，所以我们可以自己重写dealloc方法，来实现类里的对其它使用资源的释放工作。

注意：不要直接在代码中显示调用dealloc方法。

## 垃圾回收

在Objective-C 2.0中引入了垃圾回收机制(自动管理内存)，在工程设置里设置Objective-C Garbage Collection为Required[-fobjc-gc-only]就可以使用垃圾回收机制。

启用垃圾回收机制后，通常的内存管理命令都变成了空操作指令，不执行任何操作。

Objective-C的垃圾回收机制是一种继承性的垃圾回收器，垃圾回收器定期检查变量和对象以及他们之间的指针，当发现没有任何变量指向对象时，就将该对象视为被丢弃的垃圾。所以在不在使用一个对象时，将指针他的指针设置为nil,这时垃圾回收器就会清理该对象。

注意：如果开发iPhone软件，则不能使用垃圾回收。在编写iPhone软件时，Apple公司建议不要在自己的代码中使用autorelease方法，并且不要使用创建自动释放对象的函数。

## 类别

什么是类别？类别是一种为现有类添加新方法的方式。

为什么使用类别或者说使用类别的目的是什么？有以下三点：

第一，可以将类的实现分散到多个不同的文件或多个不同的框架中。

如果一个类需要实现很多个方法，我们可以将方法分类，把分好的类形成类别，可以有效的管理和驾驭代码。

第二，创建对私有方法的前向引用。

第三，向对象添加非正式协议。

## 委托

委托的意思就是你自己想做某事，你自己不做，你委托给别人做。

在Objective-C中，实现委托是通过类别(或非正式协议)或者协议来实现。

举个例子：Apple要生产iPhone,Apple自己不生产(种种原因，其中之一就是在中国生产成本低，他们赚的银子多),Apple委托富士康来生产,本来富士康原来不生产iPhone,现在要生产了,所以他得自己加一个生产iPhone的生产线(类别，增加生产iPhone方法)，这就是通过类别来实现委托。下面用代码来说明这个例子。

.....

```
Apple *apple = [[Apple alloc] init];
```

```
Foxconn *fox = [[Foxconn alloc] init];
```

```
[apple setDelegate:fox];
```

```
[apple produceiPhone];
```

.....

```
@implementation Apple
```

```
-(...) setDelegate:(id) x{
```

```
delegate = x; //! 将委托的生产对象指定为x
```

```
}
```

```
-(...) produceiPhone{
```

```
[delegate produceiPhone]; //! 委托对象生产iPhone
```

```
}
```

```
@interface Foxconn : NSObject
```

```
...
```



```
@end
```

```
@interface NSObject(ProduceIPhone) //! Foxconn之前就可以生产其它产品，有过声明和定义
```

```
-(...) produceIPhone //! 增加生产iPhone能力
```

```
@end
```

```
@implementation NSObject(ProduceIPhone)
```

```
//! 生产iPhone
```

```
-(...) produceIPhone{
```

```
.....
```

```
}
```

```
@end
```

别走开，下页更精彩

## 非正式协议

创建一个NSObject的类别，称为创建一个非正式协议。为什么叫非正式协议呢？

也就是说可以实现，也可以不实现被委托的任务。

拿上面的例子来说，Apple要求Foxconn除了能生产iPhone外，还有一个要求是在一定时间内完成.由于双方没有签合同，所以时间要求和生产要求规格都是非正式协议

## 选择器

选择器就是一个方法的名称。选择器是在Objective-C运行时使用的编码方式，以实现快速查找。可以使用@selector预编译指令，获取 选择器 @selector(方法名)。NSObject提供了一个方法respondsToSelector:的方法，来访问对象是否有该方法(响应该消息)。

拿上面的Apple请Foxconn生产iPhone为例，Apple怎么知道Foxconn有没有生产iPhone的能力呢?Apple就通过 respondsToSelector方法询问Foxconn，是否可以生产iPhone(是否可以响应produceIPhone)，询问结果是可以，那Apple就委托Foxconn生产，Foxconn就生产出来了人们比较喜欢的iPhone产品。

## 正式协议

与非正式协议比较而言，在Objective-C中，正式协议规定的所有方法必须实现。在Objective-C2.0中，Apple又增加了两个关键字，协议中的方法也可以不完全实现，是哪个关键字见关键字部份的@optional,@required。

正式协议声明如下：

```
@protocol XXX
```

```
-(...) func1;
```

```
-(...) func2;
```

```
@end
```

使用协议：

```
@interface Object : NSObject //! Object从NSObject派生，并遵循XXX协议，要实现func1,func2函数。
```

```
...
```

```
@end
```

## 习惯用法

分配内存和初始化

```
self = [super init];
```

对象间交互

在Objective-C中，所有对象间的交互都是通过指针实现。

快速枚举

```
for (Type *p in array)
```

注意:

Objective-C不支持多继承

objective-c只不过是拥有一些附加特性的C语言。本质上就是C语言

**1.C语言使用#include通知编译器应在头文件中查询定义。objective-c也可以使用#include来实现这个目的，但你永远不可能这么做，你会用#import，它是GCC编译器提供的，#import可以保证头文件只被包含一次。**

xcode会使用预编译头文件（一种经过压缩的，摘要形式的头文件），在通过#import导入这种文件时，加载速度会非常快。

## 2.什么是框架

框架是一种聚集在一个单元的部件集合，包含头文件，库，图像，声音文件等。苹果公司将cocoa，Carbon，QuickTime和OpenGL 等技术 作为框架集提供。cocoa的组成部分有Foundation和Application Kit框架。还有一个支持框架的套件，包含 Core Animation和Core Image，这为Cocoa增添了多种精彩的功能。

每个框架都是一个重要的技术集合，通常包含数十个甚至上百个头文件。每个框架都有一个主头文件，它包含了所有框架的各个头文件。通过使用#import导入主头文件，可以使用所有框架的特性。

**3.Foundation框架处理的是用户界面之下的层（layer）中得特性，例如数据结构和通信机制。**

## 4.NS前缀

NS这个前缀告诉你函数来自cocoa而不是其他工具包。

两个不同事物使用相同标示符时会导致名称冲突，而前缀可以预防这个大问题。

## 5.BOOL类型

objective-c中得BOOL实际上是一种对带符号的字符类型（signed char）的定义。它使用8位存储空间，YES为1，NO为0。

## 6.间接

不在代码中直接使用某个值，而是使用指向该值的指针。另一个意思是，让别的类来完成本类的工作。

例子：

1.循环次数的变量。变量与间接

2.使用从文件读取。文件与间接

在OOP（面向对象编程）中，间接十分重要。OOP使用间接来获取数据，OOP真正的革命性就是它在调用代码中使用间接。比如在调用函数时，不是直接调用，而是间接调用。

## 7.过程式程序与OOP的区别

过程式程序建立在函数之上，数据为函数服务。面向对象编程从相反的角度来看待问题。它



以程序的数据为中心，函数为数据服务。在OOP中，不在重点关注程序中得函数，而是专注与数据。

## 8.id

id是一种泛型，用于表示任何种类的对象。

## 9.OOP中得一些术语

类：类是一种结构，它表示对象的类型。对象引用类来获取和本身有关的各种信息，特别是运行什么代码来处理每种操作。

对象：对象是一种结构，它包含值和指向其类的隐藏指针。

实例：实例是“对象”的另一种称呼。

消息：消息是对象可以执行的操作，用于通知对象去做什么。

方法：方法是响应消息而运行的代码。根据对象的类，消息可以调用不同的方法。

方法调度程序：是objective-c使用的一种机制，用于推测执行什么方法以响应某个特定的消息。

接口：接口是对象的类应该提供的特性的描述。接口不提供实现细节。

实现：实现是使接口正常工作的代码。

## 10.中缀符

objective-c有一种名为中缀符的语法技术。方法的名称及其参数都是合在一起的。例如：

[trxtThing setValue:@"Hello there" color:kBlueColor]; 中 setValue: 和 color:实际上是参数的名称（实际上是方法名称的一部分）。使代码可读性更强，更容易理解参数的用途。

## 11.先行短线

```
-(void)draw;
```

前面的短线表明这是objective-c方法的生命。这是一种区分函数原型与方法声明的方式，函数原型中没有先行短线。-代表是实例方法。+代表是类方法。

## 12.@interface

创建某个特定类的对象之前，objective-c编译器需要一些有关该类的信息。他必须知道对象的数据成员和它提供的特性可以使用@interface指令把这种信息传递给编译器。用于定义类的公共接口。

## 13.@implementation

是一个编译器指令，表明你将为某个类提供代码。类名出现在@implementation之后。该行的结尾处没有分号。因为在objective-c编译器指令后不必使用分号。

@interface和@implementation间的参数名不同是正确的。

在@interface中没有声明却在@implementation中实现的方法是私有方法。

## 14.实例化

创建对象的过程叫做实例化。实例化对象时，需要分配内存，然后这些内存被初始化并保存一些有用的默认值，这些值不同于你在获得新分配内存时得到的随机值。内存分配和初始化完成后，就创建了一个新的对象实例。

## 15.继承

创建一个新类时，通常需要定义新类以区别于其他类以及现有类。使用继承可以定义一个具有父类所有功能的新类，它继承了父类的这些功能。

objective-c没有多继承。

创建一个新类时，其对象首先从自身的超类中继承实例变量，然后添加他们自己的实例变量。

超类

父类

子类

孩子类

重写

方法调度:objective-c的方法调度程序将子当前类中搜索响应的方法。如果调度程序无法在接受消息的对象类中找到响应的方法，它就会在该类的超类中进行查找。顺着继承链找到下一个超类进行查找，直到NSObject类中也没有该方法，则会出现运行时错误。

## 16.复合

对象引用其他对象时，可以利用其他对象提供的特性，这就是复合。

## 17.UML

UML是一种用图表表示类，类的内容以及他们之间关系的常见方法。

## 18.多态

使用更具体种类的对象（子类对象）代替一般类型（父类），这种能力称为多态性。

## 19.self

是一个指向接收消息的对象的指针。指向第一个实例变量isa。因为objective-c编译器已经看到了所有这些类的@interface声明，因此，它能直到对象中实例变量的布局，通过这些重要的信息，编译器可以产生代码并查找任何需要的实例变量。

基地址加偏移：编译器使用“基地址加偏移”机制实现奇妙的功能。给定的对象基地址，是指第一个实例变量的首个字节在内存中得位置，通过在该地址加上偏移地址，编译器就可以查到其他实例变量的位置。

## 20.间接寻址方式，直接寻址方式

## 21.super

objective-c提供某种方式来重写方法，并且仍然调用超类的实现方式。当需要超类实现自身的功能，同时在前面或者后面执行某些额外的工作时，这种机制非常有用。为了调用继承方法的实现，需要使用super作为方法调用的目标。

## 22.cocoa

cocoa实际上是由2个不同的框架组成的：Foundation Kit和 Application Kit。Application Kit包含了所有的用户接口对象和高级类。

Foundation Kit

## 23.NSAutoreleasePool

mian()函数创建了(通过alloc)并初始化(通过init)了一个NSAutoreleasePool实例。在mian()函数结尾，这个池被排空。这就是Cocoa内存管理的预览。

## 24.NSRange

```
typedef struct _NSRange {  
  
    unsigned int location;  
  
    unsigned int length;  
  
}NSRange;
```

这个结构体用来表示相关事务的范围，通常是字符串里的字符范围或者数组里的元素范围。

## 25.三种赋值方式

1.NSRange range;

range.location = 17;

range.length = 4;

2.C语言的聚合结构赋值机制

NSRange range = {17, 4};

3.Cocoa提供的一个快捷函数NSMakeRange()

NSRange range = NSMakeRange(17, 4);

使用NSMakeRange()的好处是你可以在任何能够使用函数的地方使用它，例如在方法调用中将其当成参数传递。

## 26.几何数据类型

1.NSPoint 代表笛卡尔平面中得一个点(x, y).

```
typedef struct _NSPoint {
```

```
float x;
```

```
float y;
```

```
}NSPoint;
```

2.NSSize 用来存储长度和宽度

```
typedef struct NSSize {
```

```
float width;
```

```
float height;
```

```
}NSSize;
```

3.NSRect 矩形数据类型，它是由点和大小复合而成

```
typedef struct _NSRect {
```

```
NSPoint origin;
```

```
NSSize size;
```

```
}NSRect;
```

## 27.字符串NSString

stringWithFormat:就是一个工厂方法，它根据你提供的参数创建新对象。

length:长度

isEqualToString:比较字符串内容是否相同

compare:将接受对象和传递来的字符串逐个字符的进行比较。返回一个enum数据

NSCaseInsensitiveSearch:不区分大小写字符。

NSLiteralSearch:进行完全比较，区分大小写

NSNumericSearch:比较字符串的字符个数，而不是字符值。

-(NSRange)rangeOfString:(NSString \*)aString;

返回的range.start为开始位置， range.length为长度。

## 28.NSMutableString可变字符串。

stringWithCapacity:创建一个新的NSMutableString

字符串的大小并不仅限于所提供的容量，这个容量仅是个最优值。如果要创建一个40mb的字符串。

```
NSMutableString *str = [NSMutableString stringWithCapacity: 42];
```

appendString接受参数aString，然后将其复制到接收对象的末尾。

appendFormat与stringWithFormat：类似，但它将格式化的字符串附加在接收字符串的末尾，而不是创建新的字符串对象。

## 29.集合家族

1.NSArray:是一个Cocoa类，用来存储对象的有序列表。

两个限制：1.只能存储objective-c的对象，不能存储C语言中得基本数据类型。

2.也不能存储nil。

## 30.枚举器，快速枚举

## 31.NSDictionary字典

关键字及其定义的集合。

## 32.NSNumber包装(以对象形式实现)基本数据类型

装箱：将一个基本类型的数据包装成对象。

取消装箱：从对象中提取基本类型的数据。

objective-c不支持自动装箱。

## 33.NSValue是NSNumber的父类。

```
+(NSValue *)valueWithBytes:(const void *)value objCType:(const char *)type;
```

传递的参数是你想要包装的数值的地址(如一个NSUInteger或你自己的struct)。通常，得到的是你想要存储的变量的地址（在C语言中使用操作符&）。你也可以提供一个用来描述这个数据类型的字符串，通常用来说明struct中实体的类型和大小。你不用自己写代码来生成这个字符串，@encode编译器指令可以接受数据类型的名称并为你生成合适的字符串。

```
NSRect rect = NSMakeRect(1, 2, 30, 40);
```

```
NSValue *value;
```

```
value = [NSValue valueWithBytes:&rect objCType:@encode(NSRect)];
```

```
[array addObject:value];
```

## 34.NSNull只有一个方法[NSNull null];

[NSNull null]总是返回一样的数值，所以你可以使用运算符==将该值与其他值进行比较。

## 35.单实例架构：只需要一个实例。

查找文件：

例如：NSFileManager \*manager;

```
manager = [NSFileManager defaultManager];
```

defaultManager可以为我们创建一个属于我们自己的NSFileManager对象。

```
NSString *home = [@"~" stringByExpandingTildeInPath];将~替换成当前用户的主目录。
```

```
NSDirectoryEnumerator *direnum = [manager enumeratorAtPath:home];返回一个
```

NSDictionaryEnumerator，它是NSEnumerator的子类。每次在这个枚举器对象中调用nextObject时，都会返回该目录中得一个文件的另一个路径。

### 36.内存管理

#### 1)对象生命周期

对象的生命周期包括诞生(alloc或者new方法实现)，生存(接受消息和执行操作)，交友(借助方法的组合和参数)以及当他们的生命结束时最终死去(被释放)。当对象的生命周期结束时，他们的原材料(内存)将被回收以供新的对象使用。

#### 2)引用计数

cocoa采用了一种称为引用计数的技术，有时候也叫保留计数。每个对象有一个与之相关联的整数，称作它的引用计数器或保留计数器。当某段代码需要访问一个对象时，该代码将该对象的保留计数器值+1，表示“我要访问该对象”。当这段代码结束对象访问时，将对象的保留计数器值-1，表示它不再访问该对象，当保留计数器值为0时，表示不再有代码访问该对象了，因此该对象被销毁，其占用的内存被系统回收以便重用。

alloc, new, copy 1

retain +1

release -1

#### 3)对象所有权

如果一个对象具有指向其他对象的实例变量，则称该对象拥有这些对象。

在类A中 B对象拥有其指向的C对象，则B对象拥有C对象。

如果一个函数创建了一个对象，则称该函数拥有它创建的这个对象。

main()函数创建了对象a 称main()函数拥有a对象

当多个实体拥有某个特定的对象时，对象的所有权关系就更加复杂了，这也是保留计数器值可能大于1的原因。

例子：

```
main() {  
  
    Engine *engine = [Engine new];  
  
    [car setEngine:engine];  
  
}
```

现在哪个实体拥有engine对象？是main函数还是car类？哪个实体负责确保当engine对象不再被使用时能够收到release消息？因为 car类正在使用engine对象，所以不可能是main函数。因为main函数随后还可能会使用engine对象，所以也不可能是car类。

解决方法是让car类保留engine对象，将engine对象的保留计数器值增加到2。这是因为car类和main函数这2个实体都正在使用 engine对象。car类应该在setEngine：方法中保留engine对象，而main函数应该释放engine对象。然后，当car类完成其任务时再释放engine对象(在其dealloc方法中)，最后engine对象占用的资源被回收。

如果您使用名字以“alloc”或“new”开头或名字中包含“copy”的方法（例如alloc，newObject或mutableCopy）创建了一个对象，则您将获得该对象的所有权；或者如果您向一个对象发送了一条retain消息，则您也会获得该对象的所有权。

#### 4)访问方法中得保留和释放

#### 5)自动释放池NSAutoreleasePool

是一个存放实体的池(集合)。你可以用NSMutableArray来编写自己的自动释放池，以容纳对象

并在dealloc方法中向池中得所有对象发送release消息。

autorelease

当给一个对象发送autorelease消息时，实际上是将该对象添加到NSAutoreleasePool中。当自动释放池被销毁时，会向该池中得所有对象发送release消息。

#### 6)自动释放池的销毁时间

在我们一直使用的Foudation库工具中，创建和销毁自动释放池的方法非常明确：

```
NSAutoreleasePool *pool;

pool = [[NSAutoreleasePool alloc] init];

...

[pool release];
```

创建一个自动释放池时，该池自动成为活动的池。释放该池时，其保留计数器值归0，然后该池被销毁。在销毁的过程中，该池释放其包含的所有对象。当使用 Application Kit时，cocoa定期自动为你创建和销毁自动释放池。通常是在程序处理完当前事件以后执行这些操作。你可以使用任意多得自动 释放对象，当不再使用它们时，自动释放池将自动为你清理这些对象。

你可能已经在xcode自动生成代码中遇见过另一种销毁自动释放池中对象的方式：-drain方法。该方法只是清空自动释放池而不是销毁它。并且只适用于mac os x10.4以上的版本。

#### 7)自动释放池的工作过程

我们在任何时候向一个对象发送autorelease消息，该对象都会呗添加到这个自动释放池中。被加入到自动释放池的对象的引用计数值不会变化。当自动释放池被销毁时(向自动释放池发送release消息，自动释放池的引用计数值变为0，调用自身的dealloc函数)，会调用自身的dealloc函数，会向池中得对象发送release消息。

别走开，下页更精彩

### 37.cocoa内存管理规则

1)当你使用new，alloc或copy方法创建一个对象时，该对象的保留计数器值为1。当不再使用该对象时，你要负责向该对象发送一条release或autorelease消息。这样，该对象将在其使用寿命结束时被销毁。

2)当你通过任何其他方法获得一个对象时，则假设该对象的保留计数器值为1，而且已经被设置为自动释放，你不需要执行任何操作来确保该对象被清理。如果你打算在一段时间内拥有该对象，则需要保留它并确保在操作完成时释放它。

3)如果你保留了某个对象，你需要(最终)释放或自动释放该对象。必须保持retain方法和release方法的使用次数相同。

### 38.清理自动释放池

由于自动释放池的销毁时间是完全确立的，所以它在循环执行过程中不会被销毁。在迭代中或者循环中，需要建立自己的自动释放池。

### 39.垃圾回收gc

自动内存管理机制。objective-c的垃圾回收器是一种继承性的垃圾回收器。与那些已经存在了一段时间的对象相比，新创建的对象更可能被当成 垃圾。垃圾回收器定期检查变量和对象以及他们之间的指针，当发现没有任何变量指向某个对象时，就将该对象视为应该被丢弃的垃圾。如果你再一个实例变量中指向某个 对象，一定要在某个时候使该实例变量赋值为nil，以取消对该对象的引用并使垃圾回收器知道该对象可以被清理了。

与自动释放池一样，垃圾回收器也是在时间循环结束时才触发。



ARC是什么?

ARC是iOS 5推出的新功能, 全名叫 ARC(Automatic Reference Counting)。简单地说, 就是代码中自动加入了retain/release, 原先需要手动添加的用来处理内存管理的引用计数的代码可以自动地由编译器完成了。

该功能在 iOS 5/ Mac OS X 10.7 开始导入, 利用 Xcode4.2 可以使用该功能。简单地理解 ARC, 就是通过指定的语法, 让编译器(LLVM 3.0)在编译代码时, 自动生成实例的引用计数管理部分代码。有一点, ARC并不是GC, 它只是一种代码静态分析 (Static Analyzer) 工具。

#### 40.分配

是一个新对象诞生的过程。是从操作系统获得一块内存并将其指定为存放对象的实例变量的位置。

#### 40.初始化

与分配对应的操作是初始化。初始化从操作系统取得一块内存。准备用于存储对象。

嵌套调用技术非常重要, 因为初始化方法返回的对象可能与分配的对象不同。

##### 1)初始化时做什么?

给实例变量赋值并创建你得对象完成任务所需要的其他对象。

##### 2)便利初始化函数

许多类包含便利初始化函数。用来完成某些额外的工作的初始化方法, 可以减少你自己完成这些工作的麻烦。

例如: NSString类中-(id)initWithFormat:(NSString \*)format,...;

##### 3)指定初始化函数

类中的某个初始化方法被指派为指定初始化函数。该类所有初始化方法使用指定初始化函数执行初始化操作。子类使用其超类的指定初始化函数实现超类的初始化。

例如: 其他初始化函数通过指定初始化函数实现。

#### 41.初始化函数规则

不需要为你自己的类创建初始化函数方法。如果不需要设置任何状态, 或者只需要alloc方法将内存清零的默认行为, 则不需要担心init。

如果构造了一个初始化函数, 则一定要在你自己的指定初始化函数中调用超类的指定初始化函数。一定要将超类的初始化函数的值赋给self对象, 并返回你自己的初始化方法的值。因为超类可能决定返回一个完全不同的对象。

如果初始化函数不止一个, 则选择一个座位指定初始化函数。被选定的方法应该调用超类指定初始化函数。要按照指定初始化函数的形式实现所有其他初始化函数, 就像我们在前面的实现一样。

#### 42.特性@property

objective-c2.0的特性只适用于mac os x 10.5或更高版本。特性主要应用于cocoa的新组件(尤其是华丽夺目的core Animation效果)。

##### 1)简化接口

@property预编译指令的作用是自动生命属性的setter和getter方法。

##### 2)@synthesize也是一种新的编译器功能, 表示“创建该属性的访问器”。

##### 3)点表达式

如果点表达式出现在=左边, 该属性名称的setter方法(set方法)将被调用。如果点表达式出现在

对象变量右边，则该属性名称的getter方法(get方法)将被调用。

#### 4)特性扩展

@property()括号里面的东西,是对应在set方法中要添加的语句。比如我在括号里写retain，就相当于在它的set方法里添加了一句 [xx retain]。

@property属性

属性分为3类：

1.读写属性（Writability）包含：readwrite / readonly

2.setter语义（Setter Semantics）包含：assign / retain / copy

3.原子性（Atomicity）包含：nonatomic

下面具体说明各个属性的含义

readwrite / readonly：

决定是否生成set访问器，readwrite是默认属性，生成getter和setter方法；readonly只生成getter方法，不生成setter方法。

readonly关键字代表setter不会被生成，所以它不可以和 copy/retain/assign组合使用。

assign / retain / copy:

这些属性用于指定set访问器的语义，也就是说，这些属性决定了以何种方式对数据成员赋予新值。

assign:

直接赋值，索引计数不改变，适用于简单数据类型，例如：NSNumber、CGFloat、int、char等。

retain：

指针的拷贝，使用的是原来的内存空间。

对象的索引计数加1。

此属性只能用于Objective-C对象类型，而不能用于Core Foundation对象。（原因很明显，retain会增加对象的引用计数，而基本数据类型或者Core Foundation对象都没有引用计数）。

copy：

对象的拷贝，新申请一块内存空间，并把原始内容复制到那片空间。

新对象的索引计数为1。

此属性只对那些实行了NSCopying协议的对象类型有效。

很多Objective-C中的object最好使用retain，一些特别的object（例如：string）使用copy。

nonatomic：

非原子性访问，不加同步，多线程并发访问会提高性能。如果不加此属性，则默认是两个访问方法都为原子型事务访问。默认值是atomic，为原子操作。

（atomic是Objc使用的一种线程保护技术，基本上讲，是防止在写未完成的时候被另外一个线程读取，造成数据错误。而这种机制是耗费系统资源的，所以在iPhone这种小型设备上，如果没有使用多线程间的通讯编程，那么nonatomic是一个非常好的选择。）

#### 5)保留周期retain cycle

引用计数器在该周期中归零。

#### 6)什么是属性访问器

属性访问器（Property Accessor），包括 get 访问器和 set 访问器分别用于字段的读写操作

其设计目的主要是为了实现面向对象（OO）中的封装思想。根据该思想，字段最好设为 private，一个精巧的类最好不要直接把字段设为公有提供给客户调用端直接访问

另外要注意属性本身并不一定和字段相联系

#### 7)self.a与a的区别

self.a使编译器知道我们期望使用访问器访问a。如果只使用裸名a，编译器将假设我们直接修改了实例变量。

#### 8)self.a = nil

这行代码表示使用nil参数调用setName:方法。生成的访问器方法将自动释放以前的name对象，并使用nil替代a。该方法完成了释放 name对象所占内存的操作。当然，也可以只释放name对象以清理其占用的内存。如果你再dealloc方法以外的地方清除特性，那么使用"将nil赋值给对象"的方法可以将特性设置为nil，同时可以使我们避免对已释放内存的悬空引用问题。

#### 9)特性不是万能的

有些方法不适合特性所能涵盖的方法的相当狭小的范围。特性不支持那些需要接受额外参数的方法。

### 43.类别@category

#### 1)声明类别

```
@interface NSString (NumberConvenience)
```

```
-(NSNumber *)lengthAsNumber;
```

```
@end
```

该声明具有2个特点。首先，现有类位于@interface关键字之后，其后是位于圆括号中的一个新名称。该声明表示，类别的名称是 NumberConvenience，而且该类别将向NSString类中添加方法。只要保证类别名称的唯一性，你可以向一个类中添加任意多得类别。

其次，你可以指定希望向其添加类别的类以及类别的名称，而且你还可以列出添加的方法，最后以@end结束。由于不能添加新实现变量，因此与类声明不同的是，类别的声明中没有实例变量部分。

#### 2)实现类别

#### 3)类别的局限性

第一，无法向类中添加新的实例变量。类别没有位置容纳实例变量。

第二，名称冲突，即类别中得方法与现有的方法重名。当发生名称冲突时，类别具有更高的优先级。你得类别方法将完全取代初始方法，从而无法再使用初始方法。有些编程人员在自己的类别方法中增加一个前缀，以确保不发生名称冲突。

有一些技术可以克服类别无法增加新实例变量的局限。例如，可以使用全局字典存储对象与你想要关联的额外变量之间的映射。但此时你可能需要认真考虑一下，类别是否是完成当前任务的最佳选择。

#### 4)类别的作用

cocoa中得类别主要用于3个目的：第一，将类的实现分散到不同文件或者不同框架中。第二，创建对私有方法的前向引用。第三，向对象添加非正式协议。

**44.run**循环是一种cocoa构造，它一直处于阻塞状态(即不执行任何处理)，知道某些事件发生为止。

### 45.响应选择器

一个类别如何知道其委托对象是否能够处理那些发送给它的消息?

类别首先检查对象, 询问其能否响应该选择器。如果该对象能够响应该选择器,

1)选择器@selector()

选择器只是一个方法名称, 但它以objective-c运行时使用的特殊方式编码, 以快速执行查询。

你可以使用@selector()预编译指令选择器, 其中方法名位于圆括号中。

#### 46.委托 非正式协议

#### 47.正式协议

与非正式协议一样, 正式协议是一个命名的方法列表。但与非正式协议不同的是, 正式协议要求显式的采用协议。采用协议的办法是在类的@interface声明中列出协议名称。采用协议意味着你承诺实现该协议的所有方法。

1)声明协议

```
@protocol NSCopying
```

```
-(id)copyWithZone:(NSZone *)zone;
```

```
@end
```

2)采用协议

```
@interface Car: NSObject <NSCopying>
```

```
{
```

```
}
```

```
@end
```

3)协议和数据类型

如果一个用尖括号括起来的协议名称跟随在id之后, 则编译器将知道你期望任意类型的对象, 只要其遵守该协议。

4)objective-c2.0的新特性@optional @required

@optional可选择实现的方法

@required必须实现的方法

因此cocoa中得非正式协议正被带有@optional的正式协议所取代。

#### 48.Application Kit

1)IBOutlet与IBAction

他们实际上只是APPKit提供的#define。IBOutlet的定义没有任何作用, 因此将不会对它进行编译。IBAction定义为 void, 这意味着在AppController中声明的方法的返回类型将使void。

IBOutlet和IBAction不执行任何操作, 他们并不是用于编译的, 实际上他们是为Interface Builder以及阅读代码的人提供的标记。通过查找IBOutlet和 IBAction, Interface Builder知道AppController对象具有两个能够连接的实例变量。

2)IBOutlet是如何工作的

当加载nib文件时(MainMenu.nib会在应用程序启动时自动加载, 可以创建你自己的nib文件并自行加载), 存储在nib文件中得任何对象都会被重新创建。这意味着会再后台执行alloc和init方法。所以, 当应用程序启动时, 会分配并初始化一个AppController实例。在执行 init方法期间, 所有IBOutlet实例变量都为nil。只有创建了nib文件中得所有对象(这包括窗口和文本域和按钮), 所有连接才算完成。

一旦建立了所有连接(也就是将NSTextField对象的地址添加到AppController的实例变量中),

会向创建的每个对象发送消息 `awakeFromNib`。一个非常常见的错误是试图在`init`方法中使用 `IBOutlet`执行一些操作。由于所有实例变量都为`nil`，发送给他们的所有 消息不执行任何操作，所以在`init`中得任何尝试都会发生无提示失败。(这是Cocoa导致效率低和占用大量调试时间的一个方面)。如果你想知道为什么这 些操作不起作用，可以使用`NSLog`输出实例变量的值，并查看他们是否都为`nil`。对于创建的对象和发送的`awakeFromNib`消息，都不存在预定义 顺序。

文件加载与保存

#### 49.属性列表

##### 1)自动释放对象

`NSDate`

`NSData` `NSData`对象是不可改变的。他们被创建后就不能改变。可以使用他们，但不能改变其中的内容。

##### 2)编码对象 编码和解码

cocoa具备一种机制来将对象自身转换为某种格式并保存到磁盘中。对象可以将他们的实例变量和其他数据块编码为数据块，然后保存到磁盘中。以后将这些数据块读回到内存中，并且还能基于保存的数据创建新对象。这个过程称为编码和解码。或称为序列化和反序列化。

#### 50.键/值编码 KVC

是一种间接改变对象状态的方式，其实现方法是使用字符串描述要更改的对象状态部分。

##### 1)valueForKey与setValue:forKey:

这两种方法的工作方式相同。他们首先查找名称的setter(getter)方法，如果不存在setter(getter)方法，他们将在类中查找名为名称或\_名称的实例变量。然后给它赋值(取值)。无需通过对象指针直接访问实例变量。

##### 2)路径

键路径的深度是任意的，具体取决于对象图。

键路径不仅能引用对象值，还可以引用一些运算符来进行一些运算，例如获取一组值的平均值或返回这组值中得最小值和最大值。

例如：`NSNumber *count;`

```
count = [garage valueForKeyPath:@"cars.@count"];
```

```
NSLog(@"We have %@ cars", count);
```

我们将路径“cars.@count”拆开，cars用于获取cars属性，它是来自garage的NSArray类型的值。接下来的部分是@count，其中@符号意味着后面将进行一些运算。

和 `cars@sun.mileage`

最大值 `cars@min.mileage`

最小值 `cars@max.mileage`

##### 3)整体操作

KVC非常棒的一点是，如果向NSArray请求一个键值，它实际上会查询数组中得每个对象来查找这个键值，然后将查询结果打包到另一个数组中并返回给你。这种方法也适用于通过键路径访问的对象内部的数组。

##### 4)批处理

KVC包含两个调用，可以使用他们对对象进行批量更改。第一个调用是`dictionaryWithValuesForKeys:`。它接受一个字符串数组。该调用获取一些键，对每个键使用`valueForKey:`，然后为键字符串和刚才获取的值构建一个字典。

【编辑推荐】

- 1. [iOS开发:block的探究](#)
- 2. [iOS开发:20个帮你简化移动app开发流程的工具](#)
- 3. [中国iOS开发者两年增9倍 均用户3万](#)
- 4. [iOS开发:创建独立的警告视图](#)
- 5. [iOS开发 UITableView中的单元格背景渐变](#)

【责任编辑：milk TEL：（010）68476606】

原文：iOS面试攻略上篇:Objective-C关键字和概念

返回移动开发首页



### 微信扫一扫

#### 优质技术内容尽收掌中

同时，您也可以在移动端浏览器上输入“[www.51cto.com](http://www.51cto.com)”随时随地浏览和分享最具价值的技术内容。

#### 微博推荐



51CTO移动开  
51CTO移  
动开发频



51CTO官方微  
51cto官方  
微博



51CTO技术博  
51CTO技  
术博客官方



51CTO  
技术社区  
51CTO技  
术社区(ho



51CTO熊平  
51CTO传  
媒总裁熊平



小林51CTO  
北京无忧创  
想信息技术

一键关注

分享到：

1

收藏 | 打印 | 复制



给力  
(0票)



动心  
(0票)



废话  
(0票)




专业  
(0票)



标题党  
(0票)




路过  
(0票)




共有0人喜欢 共0条评论

社区 登录



请输入你的评论

140



或游客留言

发 布

按时间排序

还没有评论内容

友言[?]

关于 [iOS面试题](#) [Objective-C概念](#) [iOS开发](#) 的更多文章

开发者：你真的认为用户在乎的是手机操作系  
你还认为Android开发者收入低于iOS开发吗？  
苹果是否应为开发者增加一个付费升级的权限  
iOS 6 vs iOS 7--如何平衡两者设计风格的转

iOS开发各类优化解决方案集锦  
开发者在代码开发和维护中，往往会遇到很多代码优  
化和内存优化的[详细]



iOS开发OpenGL ES教程(2)绘制矩形



栏目热门

更多>>

100分的输家：一个146年历史的诺基亚为何4  
李开复：移动互联网时代该如何创业  
开源免费天气预报接口API以及全国所有地区  
三星或将推"土豪金"Galaxy S4 死磕iPhone 5  
微软收购诺基亚：移动开发者应该看到什么

同期最新

更多>>

渠道失灵？手机预装软件面临“三输”风险  
是时候纠正了！对Android与iOS游戏的那些错  
App Store五周年:盘点历年最佳应用  
你真的认为iPhone只是一部手机？透露苹果惊  
优秀程序员必须知道的32个算法，提高你的开

移动开发

频道导航

平台

移动Web | Android | iOS | Windows Phone

应用

移动应用 | 移动团队 | 应用商店 | 专题汇总 | Phone Club

观察

业界观察 | 调查数据 | 移动信息化

Android

热点 | 资讯 | 基础 | 多媒体 | 数据库 | 设计 | 工具 | 编译

热点推荐



Android开发应用详解



那些性感的让人尖叫的程序员



HTML5 下一代Web开发标准详解



高性能WEB开发应用指南



Ubuntu开源技术交流频道

热门标签：[windows频道](#) [移动开发](#) [云计算](#) [objective-c](#) [tp-link路由器设置图解](#) [html5](#)

51CTO旗下网站

领先的IT技术网站 51CTO

领先的中文存储媒体 WatchStor

中国首个CIO网站 CIOAge

中国首家数字医疗网站 HC3i