title: iOS基础知识点总结(一)
date: 2018-02-05 22:22:22
categories: iOS基础

tags: iOS

以下是一些自己收集的比较基础的面试问题（大神可以忽略）附上答案，毕竟个人水平有限难免会有些错误或回答的局限性，如果发现错误也欢迎大家指出，一起探讨进步。

1、#import和#include的区别,@class代表什么

- `import`会包含这个类的所有信息，包含实体变量和方法(.h文件中)，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，后面会再告诉你。
- `#import`比起`#include`的好处是不会引起交叉编译，也就是常说的重复包含。

注：`#import`就是把被引用类的头文件都走一遍，即把`.h`文件里的变量和方法都包含进来一次，且仅一次，而`@class`不用，所以后者编译效率更高。

2、控制器的生命周期

- initWithCoder:(NSCoder *)aDecoder：（如果使用storyboard或者xib）
- loadView：加载view
- viewDidLoad：view加载完毕
- viewWillAppear：控制器的view将要显示
- viewWillLayoutSubviews：控制器的view将要布局子控件
- viewDidLayoutSubviews：控制器的view布局子控件完成`这期间系统可能会多次调用viewWillLayoutSubviews 、 viewDidLayoutSubviews`
- viewDidAppear：控制器的view完全显示
- viewWillDisappear：控制器的view即将消失
- viewDidDisappear：控制器的view完全消失

loadView方法:

> 当我们用到控制器view的时候，就会调用控制器view的get方法，在get方法内部，首先判断view是否已经被创建，如果已经存在，则直接返回存在的view，如果不存在，则会调用控制器的loa

dView方法，在控制器没有被销毁的情况下，loadView也可能会被执行多次

小结：控制器view的生命周期：loadView -> viewDidLoad -> viewWillAppear -> viewWillLayoutSubviews -> viewDidLayoutSubviews
-> viewDidAppear -> viewWillDisappear -> viewDidDisappear

3、导航push做了哪些事情

- 当调用push方法时, 会把要push的控制器添加到导航控制器管理的栈中,把之前导航控制器中栈顶控制器View给移除,把当前栈顶控制器添加上去

4、导航pop做了哪些事情

- 当调用pop方法的时候，会把要pop的控制器从栈里移除，把之前导航控制器中栈顶控制器的view移除，把当前栈顶控制器添加上去

5、frame 和 bounds 的区别是什么?

- frame相对于父视图,是父视图坐标系下的位置和大小。bounds相对于自身,是自身坐标系下的位置和大小。
- frame以父控件的左上角为坐标原点，bounds以自身的左上角为坐标原点

6、+load 和 +initialize 的区别是什么

- +(void)load; 当类对象被引入到项目时，runtime会向每个一个类对象发送load消息，load方法会在每一个类甚至分类被引入时仅调用一次，调用的顺序：父类优先于子类，子类优先于分类
- +(void)initialize; 第一次使用这个类的时候会调用这个方法

7、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`

8、strong / weak / unsafe_unretained的区别

- weak只能修饰OC对象，使用weak不会使引用计数加1，对象销毁时修饰的对象会指向nil
- strong等价于retain，能使计数器加1，且不能用来修饰数据类型
- unsafe_unretained等价于assign，可以用来修饰数据类型和OC对象，但是不会使计数器加1，

且对象销毁时也不会将对象指向nil，容易造成野指针错误。

9、iOS 的沙盒目录结构是怎样的

- Application:存放程序源文件，上架前经过数字签名，上架后不可修改
- Documents:常用目录，iTunes同步该应用时会同步此文件夹中的内容，适合存储重要数据。
- Library
  - Caches：用于存放应用程序专用的支持文件，保存应用程序再次启动过程中需要的信息。
  - Preference：iTunes同步该应用时会同步此文件夹中的内容，通常保存应用的设置信息。您不应该直接创建偏好设置文件，而是应该使用NSUserDefaults类来取得和设置应用程序的偏好.
- tmp 目录：这个目录用于存放临时文件，保存应用程序再次启动过程中不需要的信息。该路径下的文件不会被iTunes备份。

10、通过代码如何自定义控件? 并且简单的描述下每一个步骤的理由?

- 新建一个继承`UIView`的子类(所谓自定义控件就是继承系统自带的控件写一个适合自己项目特殊需求的控件)
- 在`initWithFrame`方法中添加子控件(保证别人在其他类不管是通过`init`还是通过`initWithFrame`创建都能够添加子控件，因为init方法内部会调用initWithFrame)
- 在`layoutSubViews`方法中设置子控件的frame(在`initWithFrame`方法中当前的控件可能没值，所以计算不了子控件的位置和尺寸，而在`layoutSubViews`方法中，能够拿到当前控件的尺寸)
- 提供一个模型属性，重写模型属性的`set`方法

11、简单的描述下类扩展(extension)和分类(category)的区别?

- 类扩展(也被称作匿名分类)没有名字，分类有名字
- 类扩展中新添加的方法必须要实现，分类可以不实现
- 类扩展是分类的一个特例，可以为一个类添加一些私有成员变量和方法，分类可以在不修改原来类的基础上，为一个类添加类扩展方法，一般用于给系统自带的类扩展方法，不能添加成员变量，如果一定要添加，可以通过`runtime`实现

12、Objective-C 如何对已有的方法，添加自己的功能代码以实现类似记录日志这样的功能

- 主要考察的是`runtime`如何交换方法

实现：

1，先在分类中添加一个方法，注意不能重写系统方法，否则会被覆盖

```
+ (NSString *)p_printLog {
    // 这里写打印行号,什么方法,哪个类调用等等
}
```

2，交换方法

```
objc
// 加载分类到内存到时候调用
+ (void)load {
// 获取系统的描述方法
Method description = class_getClassMethod(self, @selector(description));
// 自定义的打印日志方法
Method myLog = class_getClassMethod(self, @selector(p_printLog));
// 交换方法
method_exchangeImplementations(description, myLog);
}
```

13、如何让 Category 支持属性

- 使用 `runtime` 实现

```
// .h
@interface NSObject (LJTest)
@property (nonatomic, copy) NSString *testName;
@end

// .m
static const char *kIdentifier = "testName";

@implementation NSObject (LJTest)

- (NSString *)testName {
    // 根据关联的 kIdentifier, 获取关联的值
    return objc_getAssociatedObject(self, kIdentifier);
}

- (void)setTestName:(NSString *)testName {
    // 第一个参数：给哪个对象添加关联
```

```
    // 第二个参数：关联的key，通过这个key获取
    // 第三个参数：关联的value
    // 第四个参数:关联的策略
    objc_setAssociatedObject(self, kIdentifier, testName, OBJC_ASSOCIATION_RETAIN_NONA
TOMIC);
}
```

14、iOS assign，weak，strong，copy， atomic， nonatomic 使用场景

assign 与 weak区别

- `assign` 适用于基本数据类型， `weak` 适用于 `NSObject` 对象，并且是一个弱引用， `assign` 其实也可以用来修饰对象，那么我们为什么不用它修饰对象呢？因为被 `assign` 修饰的对象(一般编译的时候会产生警告：Assigning retained object to unsafe property; object will be released after assignment)在释放之后，指针的地址还是存在的，也就是说指针并没有被置为 `nil` ，造成野指针，对象一般分配在堆上的某块内存内存上，如果在后续的内存分配中，恰巧分到了这块地址，程序就会崩掉
- 为什么可以用 `assign` 修饰基本数据类型呢？因为基础数据类型一般分配在栈上，栈的内存会由系统自动处理，不会造成野指针。 `weak` 修饰的对象在释放之后，指针地址会被置为 `nil` ，所以在一般弱引用使用weak修饰
- `weak` 使用场景：在ARC，在有可能出现循环引用的时候，往往要通过让其中的一端使用weak来解决，常见的比如：delegate属性，通常就会声明为weak。自身已经对它进行一次强引用，没有必要再强引用一次时也会使用weak。

strong 与 copy区别

- `strong` 与 `copy` 都会使引用计数(retain)加1，但 `strong` 是两个指针指向同一块地址，copy会在内存里拷贝一份对象，两个指针指向不同的内存地址

__block 与 __weak区别

- `__block` 用来修饰一个变量，这个变量在block代码块中，如果想被修改，则需要使用 `__block` 修饰，使用使用 `__block` 修饰的变量在block代码块中会被retain（ARC下会retain，MRC下不会retain）__weak：使用__weak修饰的变量不会在block代码块中被retain，同时，在ARC下，要避免block出现循环引用 __weak typedof(self)weakSelf = self;

nonatomic 与 atomic区别

- `atomic` 的意思就是setter/getter这两个函数的一个原语操作。如果有多个线程同时调用setter的话，不会出现某一个线程执行setter全部语句之前，另一个线程开始执行setter情况，相当于函数头尾加了锁一样。 nonatomic不保证setter/getter的原语行，所以你可能会取到不完整的东西。 比如setter函数里面改变两个成员变量，如果你用nonatomic的话，getter可能会取到只更改了其中一个

变量时候的状态。 atomic是线程安全的,nonatomic是线程不安全的。如果只是单线程操作的话用nonatomic最好,因为后者效率高一些

15、对NSUserDefaults的理解

- `NSUserDefaults`是系统提供的一种存储数据的方式，主要用于保存少量的数据，默认存储到library下的Preferences文件夹