1.xib为什么用weak修饰？

因为拖拽的时候回自动加到subviews数组中，进行了强引用

2.进程：进程是系统进行资源分配和调度的一个独立单位

线程是进程的一个实体，是CPU调度和分派的基本单位

3.为什么默认情况下无法修改被block捕获的值？—block都做了哪些？

默认情况下，block里边的变量，拷贝进去的是变量的值。—block修饰后，拷贝进去的是指向变量的指针，所以我们就可以修改变量的值

4.视图的混合中opaque和alpha对性能的影响

当视图混合图层比较多时，应该表明opaque属性以避免无用的alpha通道合成。opaque设置为YES时，系统就会认为时不透明的，就不会再去进行alpha合成。反之如果alpha为小于1的值的话，opaque就应该设置为NO

5.单元测试

6.SDWebImage实现机制

7.KVC/KVO

KVC：KVC使用场景：利用KVC可以修改类的私有变量，可以修改IOS隐藏一些属性，如UITextField的placeHolderText默认style在需求中达不到要求，我们可以直接通过KVC快速定义自己的style,代码如下：

[textFieldsetValue:[UIColorredColor]forKeyPath:@"placeholderLabel.textColor"];

KVO：在当对象(B)被监听时，那么系统就会在运行期动态的创建该对象类的一个子类，类名就是在该类的前面加上NSKVONotifying\_的前缀，子类并重写了任何被监听属性的setter方法，并使用willChangeValueForKey和didChangeValueForKey即手动触发方式来实现，这么做是基于设置属性会调用setter方法(KVC协议)。

8.正则相关知识

9.什么是堆和栈？他们在哪儿？

10.转场动画知识

11.Swift和Objective-C异同点

Swift的安全性在于：声明的变量默认都是非空的，于是声明的时候如果不给初始化赋值，就会报错；如果声明为可选类型的变量，使用的时候就会涉及到解包的流程，就能对非空进行必要的判断，于是乎在代码编译阶段就加强了对空值的判断，大大增加了代码的稳定性

a.oc是一门动态语言，而Swift是一门静态语言,告诉机器的越多，运行就越快，提高性能，减少运行时的bug，便于后期维护扩展

b.Swift更加安全，可选类型增加非空判断

c.oc中的数组，字典等引用类型在Swift中都是值类型，栈上提升性能

d.Swift更加简洁，即是面相对象也是函数式编程

12.Runtime用法

a.methodSwizzling class\_getInstanceMethod & class\_exchangeImplementation()

b.查找所有成员变量，动态添加方法和属性

找不到selector时就会执行+(BOOL)resolveInstanceMethod:(SEL)sel方法，在其中执行class\_addMethod方法即可

c.关联对象，可用于封装alertView，在分类中动态添加属性

objc\_setAssociatedObject和objc\_getAssociatedObject

13.RunLoop相关知识

Runloop就是为了保证进程的存活，每一个线程都对应一个runloop，实时的监听时间和Timer。由五部分组成：

CFRunLoopSource0:自定义方法触发

CFRunLoopSource1:port方法触发，即系统内部触发

CFRunLoopTimer:用户监听定时器的触发

CFRunLoopMode:Default、Initialization、Tracking、Common、EventRecieve（通常用不到，用于监听系统内部）

CFRunLoopObserver:可以利用这个机制进行性能优化，例如监听runloop运行状态进行tableView的行高度缓存