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()，用于适应不同手机屏幕字体大小的调整

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

14.为什么说OC是一门动态语言

a.动态类型，即运行时再决定类型，任何对象都可以被id指针所指，只有在运行时才能决定是什么类型。内置的int,NSString都属于静态类型，静态类型在编译的时候就能被识别出来，要注意区别编译时和运行时。

b.动态绑定，基于动态类型，在某个实例对象被确定后，其类型便被确定了。该对象对应的属性和相应的消息也被完全确定，这就是动态绑定。当发送respondsToSelector时，在OC消息转发机制被触发之前，对应类的resolveClassMethod将会被调用，可以在此动态添加方法

c.动态加载，根据不同机型动态加载不同的图片

15．项目中用过GCD吗

a.延时操作

b.dispatch\_group\_enter,dispatch\_group\_leave,dispatch\_group\_notify确保所有添加到group中的任务完成后再在notify中执行最后的操作

c.GCD使用信号量控制并发数

16.Category实现原理

通过runtime对category做了以下处理：1.将Category和它的主类注册到哈希表中；2.如果主类已实现，那么重建它的方法列表。在这里分了两种情况做处理：Category中的实例方法和属性被整合到主类中。而类方法则被整合到元类中。协议处理比较特殊，被同时整合到了主类和元类中。

最终都是通过调用static void remethodizeClass(Class cls)函数来重新整理类的数据的。通过源码可以看到真正处理Categoty方法的是attachMethodLists函数，主要作用是将类中的旧有方法和Category中新添加的方法合成一个新的方法列表，并赋值给method\_lists。

17.id可以指向任意对象的原因

Objective-C中的对象的定义是struct objc\_object,本质上是结构体对象，其中isa是它唯一的私有成员变量。Class和id分别是struct objc\_class和struct objc\_object类型的指针

18.实例方法和类方法的区别

实例方法是保存在类中的，而类方法是保存在元类中的。OC中类也是对象，也是某个类的实例，这个类就是元类（metaclass）。我们调用类方法时，给类对象发送消息。同样的，类对象能否响应这个消息也要通过isa找到类对象所属的类（元类）才能知道。而实例对象的元类就其所属的类。根元类的superClass不是nil而是根类，根类的superClass为nil

19.分类为什么可以添加方法但不能添加属性

因为Category在runtime中使用一个结构体表示的，它里边和NSObject的结构体不一样，分类中有instance\_methods,class\_methods,protocols,properties几个字段，没有ivarList，即成员变量，而表示ObjC类的结构体运行时并不能改变，所以可以添加方法但不能添加属性。

20.block为什么要用copy修饰

当block没有访问外界的变量时，是存储在代码区。当block访问外界变量时是存储在桟区，而此时的block处理作用域就会被释放，再去访问时就会出现野指针访问，所以要用copy修饰使block在堆中保留一份，相当于一只强引用着。因此如果block中用到self时，需要\_\_weak将其弱化。在ARC模式下，系统也会自动对block进行copy操作

\_\_block:block不允许修改外部变量的值，这里所说的外边变量的的值指的是桟中指针的内存地址。\_\_block所起到的作用就是将外部变量在桟中的内存地址放到了堆中

\_\_weak:避免block中发生循环引用，只能在ARC模式下使用，只能修饰对象，不能修饰基本数据类型

\_\_strong:保证代码在执行block的时候，修饰的对象不会被释放。当block执行完后，会自动释放该strongSelf

21.热更新技术有哪些

a.JSPatch:最根本的原因是OC是一门动态语言，OC上所有方法的调用都通过OC的Runtime在运行时进行，我们可以通过类名和方法名反射得到相应的类和方法

b.Weex

c.React Native

d.Hybrid

e.DynamicCocoa

22.事件传递流程

只有继承了UIResponder的对象才能接受并处理事件，我们称之为“响应者对象”，如UIApplication,UIViewController,UIView

传递流程是从父控件传递到子控件，也就是系统将改时间加到UIApplication管理的一个队列（FIFO）中，UIApplication->window->寻找处理事件最合适的View

事件响应链则是反过来传递，调用touches的一系列方法

23.main()函数之前的过程有哪些

动态链接库，用到的所有framework都是动态连接的，写代码前动态链接器需要完成准备工作。

整个事件有dyld（the dynamic link editor）主导，完成运行环境的初始化后，配合ImageLoader将二进制文件按格式加载到内存，动态链接依赖库，并由runtime负责加载成objc定义的结构，然后再执行load方法，所有初始化工作结束后，dyld才开始用真正的main函数。之后再去执行applicationDidFinishLaunch代理方法

24.load方法相关知识

+(void)load方法与其他方法有很大的不同。它只是一个在整个文件被加载到运行时，在main函数调用之前被objc运行时调用的钩子方法

25.项目内存优化

a.opaque不透明时设置为YES避免不必要的渲染

b.调整图片大小与imageView大小一样，临时调整大小会非常占用内存。比如WorkOS中有一个界面就是好多图片，然后程序崩溃。解决办法使用后台线程对图片进行一次缩放，然后在UIImageView中使用缩放后的图片

c.适当的使用lazy load

d.优化tableView---缓存行高(利用runloop)、减少subView的数量和视图层级、重用、缓存网络数据