内存管理

1. iOS5.0之前，OC内存管理遵循“谁创建，谁释放；谁引用，谁管理”的原则。
2. 当创建或引用一个对象的时候，需向她发送alloc、copy、new、retain消息；当释放对象时需发送release消息；当对象引用计数为0时，系统将释放该对象，这是OC的手动管理机制MRC
3. Ios5.0之后，引用自动管理机制——自动引用计数(ARC)。管理原则一样，知识不再需要手动调用retain、release、autorelease
4. ARC编译特性：在适当位置自动插入release和autorelease
5. ARC引入strong和weak关键字：strong是强引用，只要引用存在，对象就不能被销毁。weak是弱引用，当被引用对象消失时，弱引用会自动设置为nil。二者区别是，当一个对象不再有强引用指针指向时候它就会释放，即使还有weak的指针指向
6. 谁创建，谁释放：如果使用alloc、copy(mutableCopy)或retain一个对象时，你就有义务，向它发送一条release或autorelease消息。其他方法创建的对象，不需由你来管理内存
7. 向一个对象发送一条autorelease消息，这个对象不会立即被销毁，而是将这个对象放入自动释放池，待池子释放时，它会向池中每个对象发一个release消息，以此来释放对象
8. 向一个对象发送release消息，并不意味这个对象被销毁了，而是当这个对象的引用计数为0时系统才会调用对象的dealloc方法，释放该对象和对象本身所拥有的实例。
9. 如果一个对象有一个strong类型的指针指向着，这个对象就不会被释放(强引用某对象)引用计数＋1
10. 如果一个指针指向超出了它的作用域，就会被指向nil
11. 如果一个指针被指向nil，那么它原来指向的对象就被释放了
12. 当一个视图控制器被释放时，它内部的全局指针都会被指向nil
13. 局部变量：出了作用域，指针会被置为nil
14. 方法内部创建对象，外部需要添加autorelease
15. 连线时用weak描述
16. block中为了避免循环引用问题，使用weak描述
17. 如果需使用自动释放池，用@autoreleasepool{};
18. 在非ARC工程中才用ARC去编译某些类:在compile sources双击，-fobjc-arc
19. 在ARC工程中才用MRC去编译某些类:在compile sources双击，-fno-objc-arc
20. iOS定义@propery属性时需遵循内存管理语义，添加修饰符：assign,weak,retain,strong,copy，以下分别进行介绍：
21. assign：用于基本数据类型进行赋值操作，不更改引用计数，也可以修饰对象，但assign修饰对象在释放后指针地址还在，指针并没有置为nil，指向被销毁的内存，称为野指针。后续往该对象发消息会crash基础类型一般分配在栈上，栈内存由系统自动处理，不会造成野指针
22. weak：修饰object类型，对象释放后，指针会置为nil，是一种弱引用。与strong不同的是，当一个对象不再有强类型指针指向它时，就会被释放，即使有其他weak指针指向，也会被清除
23. strong：相当于retain，但有些类型如NSString，使用strong相当于使用copy
24. copy：建立一个索引计数为1的对象后释放旧对象，赋值的是一份深拷贝，即内容拷贝。注：若不可变对象，则为浅拷贝，指针拷贝。
25. retain：释放旧值，新值retainCount＋1，只指针拷贝(浅拷贝)
26. block用copy属性修饰符：block分配在栈上随时会被回收，copy一份到堆上面，就不会了。在ARC下无修饰也会自动进行copy的
27. \_\_block：在ARC、MRC下均可用，修饰对象或基本数据类型。\_\_block修饰对象可以在block中重新被赋值，但\_\_weak不行
28. \_\_weak只在ARC中使用，只能修饰对象，不能修饰基本数据类型
29. ARC下，为避免block出现循环引用，经常会\_\_weak typeof(self) weakSelf = self;
30. Parent-child相互持有、委托模式引起循环引用：让child只有parent对象为weak类型
31. 类property用strong、copy等持有了block对象，如果block再持有self引起循环引用；用9)中方法可以避免
32. NSTimer会持有对象，若类property持有了nstimer，timer又持有了self引起循环引用：在删除timer前调[timer invalidate]; \_timer = nil;