Swift第八节课：闭包（下）

OC Block 和 Swift 闭包相互调用

我们在OC中定义的Block，在Swift中是如何调用的那？我们来看一下

```
typedef void(^ResultBlock)(NSError *error);

@interface LGTest : NSObject

+ (void)testBlockCall:(ResultBlock)block;

@end
```

在 Swift 中我们可以这么使用

```
LGTest.testBlockCall{ error in
    let errorcast = error as NSError
    print(errorcast)
}

func test(_ block: ResultBlock){
       let error = NSError.init(domain: NSURLErrorDomain, code: 400, userIn
       block(error)
}
```

比如我们在 Swift里这么定义，在OC中也是可以使用的

```
class LGTeacher: NSObject{
   @objc static var closure: (() -> ())?
}

+ (void)test{
//    LGTeacher.test{}

    LGTeacher.closure = ^{
         NSLog(@"end");
    };
//    LGTeacher.closure = ^{
//
//    }
}

LGTest.test()

```

```
LGTeacher.closure!()
```

@convention： 用于修饰函数类型

- 修饰Swift中的函数类型（调用 C 函数的时候）
- 调用 OC 方法是，修饰 Swift 函数类型

defer

定义：

`defer` {} 里的代码会在当前*代码块*返回的时候执行，无论当前*代码块*是从哪个分支 return 的，即使程序抛出错误，也会执行。

如果多个 defer 语句出现在同一作用域中，则它们出现的顺序与它们执行的顺序相反，也就是先出现的后执行。

这里我们看一个简单的例子：

```
func f() {
    defer { print("First defer") }
    defer { print("Second defer") }
    print("End of function")
}
f()
```

defer能做什么?

```
func append(string: String, terminator: String = "\n", toFileAt url: URL) th
    // The data to be added to the file
    let data = (string + terminator).data(using: .utf8)!

    // If file doesn't exist, create it
    guard FileManager.default.fileExists(atPath: url.path) else {
        try data.write(to: url)
        return
    }


    // If file already exists, append to it
    let fileHandle = try FileHandle(forUpdating: url)
    fileHandle.seekToEndOfFile()
    fileHandle.write(data)
    fileHandle.closeFile()
}

let url = URL(fileURLWithPath: NSHomeDirectory() + "/Desktop/swift.txt")
try append(string: "iOS面试突击", toFileAt: url)
try append(string: "Swift", toFileAt: url)
```

这里有时候如果当前方法中多次出现 `closeFile` ,那么我们就可以使用 `defer`

```
 func append(string: String, terminator: String = "\n", toFileAt url: URL) t
    // The data to be added to the file
    let data = (string + terminator).data(using: .utf8)!

     defer{
        fileHandle.closeFile()
     }

    // If file doesn't exist, create it
    guard FileManager.default.fileExists(atPath: url.path) else {
        try data.write(to: url)
        return
    }

    // If file already exists, append to it
    let fileHandle = try FileHandle(forUpdating: url)
    fileHandle.seekToEndOfFile()
    fileHandle.write(data)

}

let url = URL(fileURLWithPath: NSHomeDirectory() + "/Desktop/swift.txt")
try append(string: "iOS面试突击", toFileAt: url)
try append(string: "Swift", toFileAt: url)try append(string: "Line 1", toFil
try append(string: "Line 2", toFileAt: url)
```

比如我们在使用指针的时候

```
let count = 2
let pointer = UnsafeMutablePointer<Int>.allocate(capacity: count)
pointer.initialize(repeating: 0, count: count)

defer {
    pointer.deinitialize(count: count)
    pointer.deallocate()
}
```

比如我们在进行网络请求的时候，可能有不同的分支进行回调函数的执行

```
func netRquest(completion: () -> Void) {
  defer {
    self.isLoading = false
    completion()
  }
  guard error == nil else { return }
}
```

全局变量和局部变量捕获的区别

捕获引用类型

捕获多个值的时候

逃逸闭包

逃逸闭包的定义：当闭包作为一个实际参数传递给一个函数的时候，并且是在函数返回之后调用，我们就说这个闭包逃逸了。当我们声明一个接受闭包作为形式参数的函数时，你可以在形式参数前写 @escaping 来明确闭包是允许逃逸的。

- 作为函数的参数传递
- 当前闭包在函数内部异步执行或者被存储
- 函数结束，闭包被调用，生命周期结束

- 不会产生循环引用，函数作用域内释放
- 编译器更多性能优化 （retain， relsase）
- 上下文的内存保存再栈上，不是堆上

注意：可选类型默认是逃逸闭包！！！

自动闭包

是一种用来把实际参数传递给函数表达式打包的闭包，不接受任何实际参数，当其调用是，返回内部表达式的值。

好处：用普通表达式代替闭包的写法，语法糖的一种

```
func debugOutPrint(_ condition: Bool , _ message: String){
    if condition {
        print("lg_debug:\(message)")
    }
}

debugOutPrint(true, "Application Error Occured")
```