ASYNCHRONOUS PROGRAMMING IN SWIFT

异步

@GUANSHANLIU SWIFTCON CHINA 2016

ABOUT ME

- 刘冠杉 GUANSHAN LIU
- SENIOR IOS DEVELOPER AT ALIBABA MUSIC
- TWITTER: @GUANSHANLIU
- MEDIUM: @GUANSHANLIU
- ORGANIZER OF COCOAHEADS SHANGHAI MEETUP

HTTP://WWW.MEETUP.COM/COCOAHEADS-SHANGHAI/

SCHEDULE

1. 现实中 REALITY
2. DEMO
3. 传统式 TRADITIONAL / 响应式 REACTIVE
4. 近未来 THE FUTURE

现实中

REALITY

APPLE

SWIFT 3.0 RELIES ENTIRELY ON PLATFORM CONCURRENCY PRIMITIVES (LIBDISPATCH, FOUNDATION, PTHREADS, ETC.) FOR CONCURRENCY. LANGUAGE SUPPORT FOR CONCURRENCY IS AN OFTEN-REQUESTED AND POTENTIALLY HIGH-VALUE FEATURE, BUT IS TOO LARGE TO BE IN SCOPE FOR SWIFT 3.0.

SWIFT THREAD SAFETY

GRAND CENTRAL DISPATCH

GCD FOR SHORT. A LOW-LEVEL C API

有人在想 SOMEONE MAY THINK ASYNCHRONOUS PROGRAMMING WITH GCD IS EASY

很简单

```
dispatch_async(utilityQueue) {
    // Download image
    dispatch_async(mainQueue, {
        // Update UI
    })
}
```

THE CALLBACK HELL 地狱

```
dispatch_async(utilityQueue) {
    // Download image
    dispatch_async(mainQueue, {
        // Update UI
        dispatch_async(utilityQueue) {
            // Cache image
        }
    })
}
```

THE CALLBACK HELL 地狱

- DIFFICULT TO READ
- DIFFICULT TO MAINTAIN
- SYNCHRONIZATION IS PAINFUL

WHAT IS HARD IN ASYNCHRONOUS PROGRAMMING?

同步 难

SYNCHRONIZATION

同步 难

```
// Bad solution
dispatch_async(firstQueue) {
    dispatch_sync(secondQueue) {
        // Code requiring both queues, may risk dead-lock
    }
}
```

EXAMPLE FROM JUSTIN SPAHR-SUMMERS

同步 难

```
// Good solution
let concurrentQueue = dispatch_queue_create("concurrent",
                                     DISPATCH_QUEUE_CONCURRENT)
dispatch_set_target_queue(firstQueue, concurrentQueue)
dispatch_set_target_queue(secondQueue, concurrentQueue)
dispatch_barrier_async(concurrentQueue) {
    // Code requiring both queues
}
```

EXAMPLE FROM JUSTIN SPAHR-SUMMERS

NSOperation
NSOperationQueue

AN OBJECTIVE-C API ON TOP OF GRAND CENTRAL DISPATCH

NSOperation & NSOperationQueue

- 依赖 DEPENDENCIES
- 状态监控 OBSERVE THE STATE USING KVO
- 控制 MORE CONTROLS:
  `maxConcurrentOperationCount`

错误处理 难

ERRORS HANDLING IN ASYNCHRONOUS SCENARIOS

错误处理 难

1. APPLE USES COMPLETION HANDLERS TO HANDLE ERRORS IN ASYNCHRONOUS SCENARIOS.
2. APPLE'S USE OF COMPLETION HANDLERS IS THEY ARE ALWAYS CALLED.
3. COMPLETION HANDLERS ARE CALLED EITHER WITH A RESULT OR AN ERROR.

错误处理 难

```
enum Result<T> {
    case Success(T)
    case Failure(ErrorType)
}
```

错误处理 难

1. NO GUARANTEE THAT AN ASYNCHRONOUS FUNCTION ALWAYS CALLS A CALLBACK
2. NO GUARANTEE THAT AN ASYNCHRONOUS FUNCTION ONLY CALLS A CALLBACK ONCE
3. DO NOT KNOW ON WHICH QUEUE THAT A CALLBACK WILL BE CALLED

状態管理 難

STATE MANAGEMENT

THE LESS STATE WE HAVE TO MANAGE, AND THE MORE DECLARATIVE CODE WE CAN WRITE, THE BETTER.

> BRENT SIMMONS

第三方

3RD PARTY FRAMEWORKS

BRIGHTFUTURE

FUTURES / PROMISES

REACTIVE

1. RXSWIFT
2. REACTIVECOCOA
3. BOND
4. VINCERP
5. INTERSTELLAR

异步编程难

- 同步难
- 错误处理难
- 状态管理难

DEMO

DEMO

- 搜索词发生，如果文字长度4个以上，发起新的请求，上一个请求被取消
- 0.3秒内搜索词多次变化，只有最后一次会发起请求
- 请求返回，界面需要更新
- 有一个刷新BUTTON，点击会立即发起请求

DEMO

近未来

THE FUTURE

ASYN - AWAIT

```
func getAvatar() -> async UIImage
do {
    let image = await getAvatar()
    // Do something with the image
} catch {
    // Handle error
}
// Or
imageView.image <~ getAvatar()
```

讲义 SLIDES + 例子 DEMO ARE ON GITHUB

谢谢

THANK YOU

欢迎提问

QUESTIONS?