APPLE WATCH & IPHONE ARE GOOD FRIENDS

GUANSHAN LIU (@GUANSHANLIU)

WHO AM I

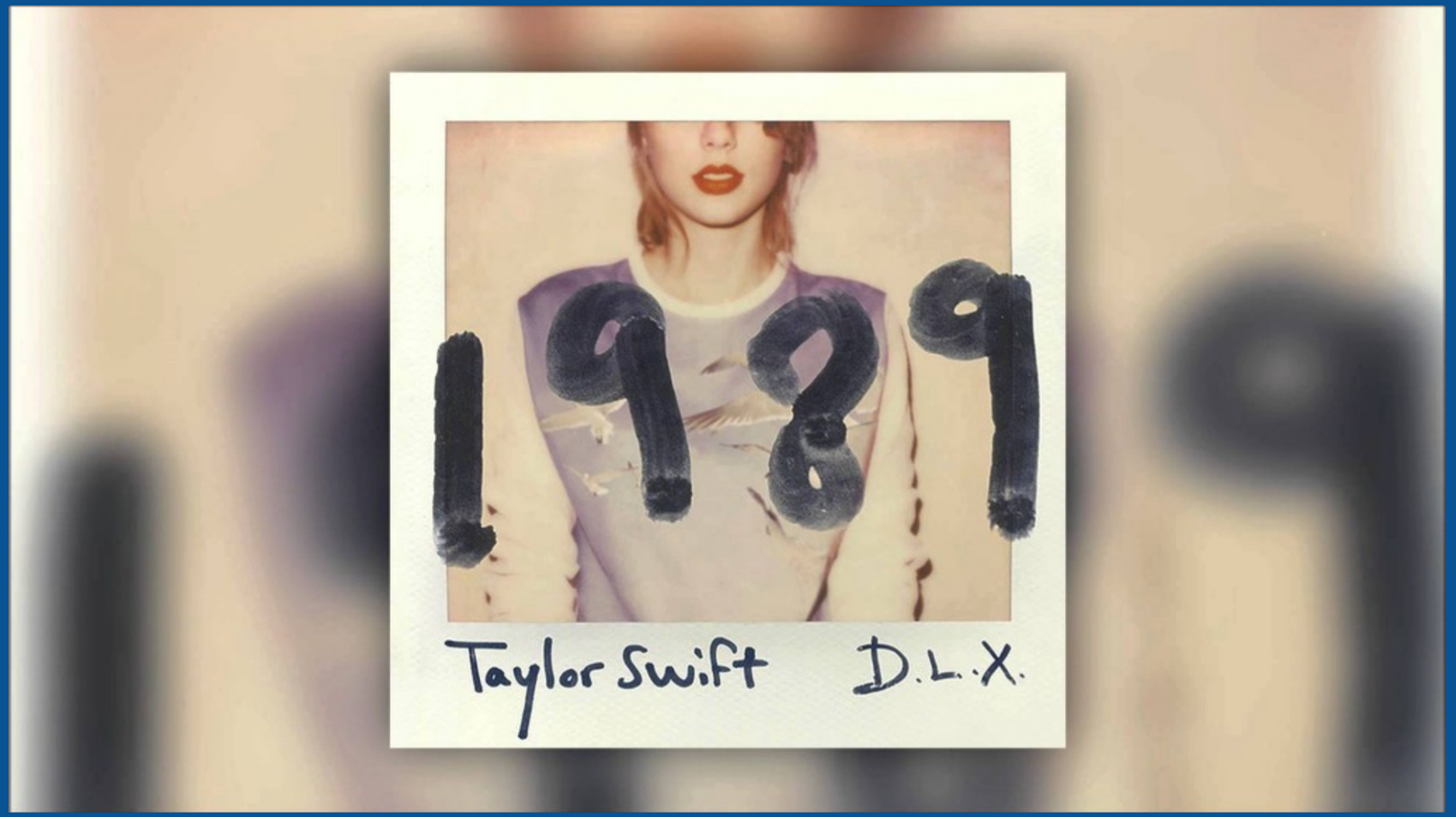
» An iOS developer on TTPod team, at Alibaba Inc.

» Twitter: @guanshanliu

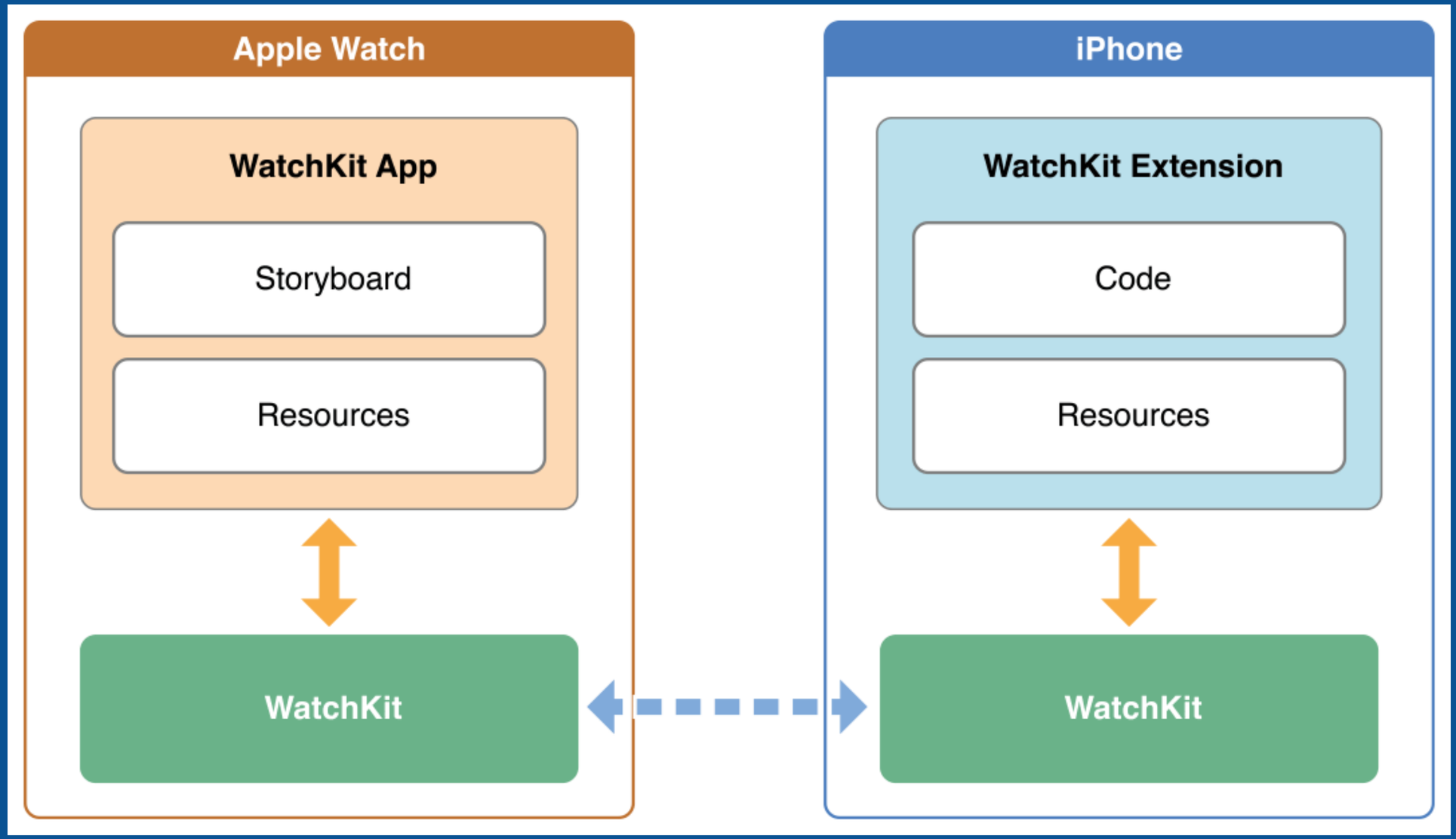
» I love coding, making things

» And I like Swift

AND THE HATERS GONNA HATE HATE HATE HATE HATE



From Apple Watch Programming Guide



SCENARIO 1:

OPEN WATCH APP

» The containing app may/may not be alive

» Needs to know what is/was data in the containing app

WatchKit: WKInterfaceController Class

```
class func openParentApplication(_ userInfo: [NSObject : AnyObject],
                                 reply reply: (([NSObject : AnyObject]!,
                                                NSError!) -> Void)?) -> Bool
```

It wakes up the containing app on iPhone via an option function of UIApplicationDelegate protocol

```
func application(_ application: UIApplication,
handleWatchKitExtensionRequest userInfo: [NSObject : AnyObject]?,
                               reply reply: (([NSObject : AnyObject]!) -> Void)!)
```

openParentApplication

LIMITATIONS:

» Initiate from the watch app

» It wakes up the containing app every time if the containing app is suspended or terminated.

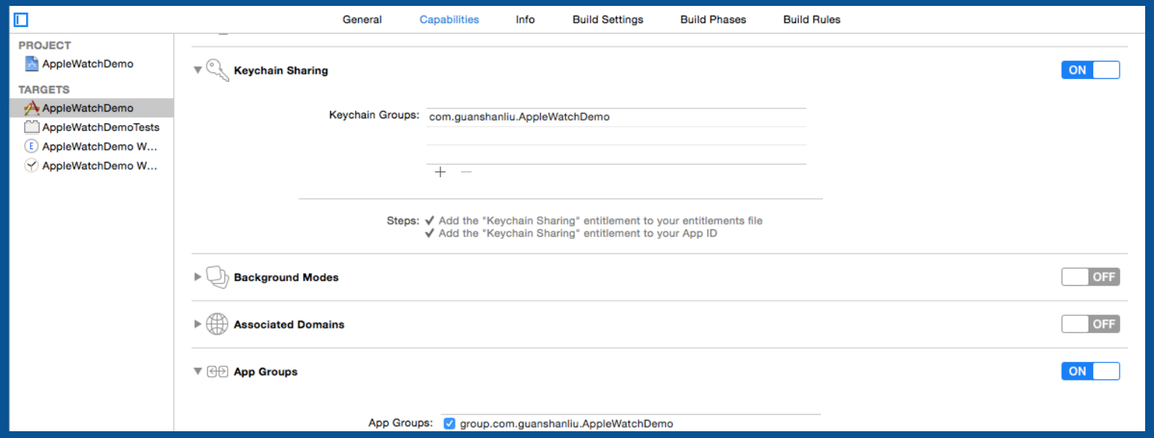
SCENARIO 2:

SHARE DATA

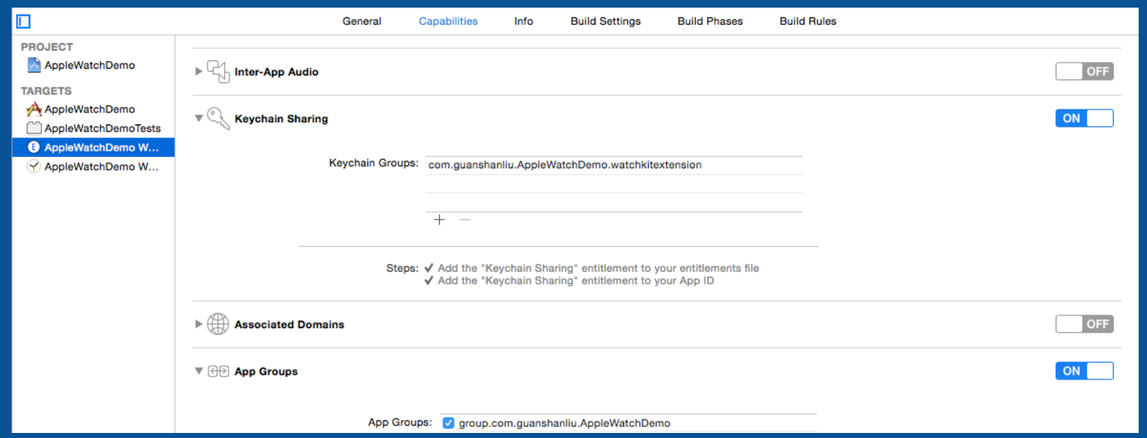
» The containing app needs to know changes made on the watch app, and vice versa

APP GROUPS

The containing app:



The WatchKit extension:



NSUSERDEFAULTS

```
class ViewController: UIViewController {
...
    let suiteName = "group.com.guanshanliu.AppleWatchDemo"
    let key = "value"

    @IBAction func onSegmentValueChange(sender: AnyObject) {
        if let defaults = NSUserDefaults(suiteName: suiteName) {
            defaults.setInteger(segment.selectedSegmentIndex, forKey: key)
            defaults.synchronize()
        }
    }

    func updateSegment() {
        if let defaults = NSUserDefaults(suiteName: suiteName) {
            segment.selectedSegmentIndex = defaults.integerForKey(key)
        }
    }

}
```

```
class InterfaceController: WKInterfaceController {
...
        @IBAction func onSliderValueChange(value: Float) {
         if let defaults = NSUserDefaults(suiteName: suiteName) {
             defaults.setInteger(Int(value), forKey: key)
             defaults.synchronize()
         }
    }
    let suiteName = "group.com.guanshanliu.AppleWatchDemo"
    let key = "value"

    func updateUIFromUserDefaults() {
         if let defaults = NSUserDefaults(suiteName: suiteName) {
             self.slider.setValue(Float(defaults.integerForKey(key)))
         }
    }
}
```

NSUserDefaults(suiteName:)

LIMITATIONS:

» Static, NSUserDefaultsDidChangeNotification doesn't apply to share user defaults atm

» Small amount of data

BUT, YOU CAN ALSO SHARE FILES, EVEN DATABASE VIA APP GROUPS.

SCENARIO 3:
GET NOTIFIED WHEN SHARED DATA CHANGES

The recommended way for reads and writes on iOS in a coordinated manner:

NSFILEPRESENTER AND NSFILECOORDINATOR

```swift
extension ViewController: NSFilePresenter {

    var presentedItemURL: NSURL? {
        let groupURL = NSFileManager.defaultManager().containerURLForSecurityApplicationGroupIdentifier(suiteName)
        let fileURL = groupURL?.URLByAppendingPathComponent("data")
        return fileURL
    }

    var presentedItemOperationQueue: NSOperationQueue {
        return NSOperationQueue.mainQueue()
    }

    func presentedItemDidChange() {
        readData()
    }

}

// Register to receive notification on changes
NSFileCoordinator.addFilePresenter(self)
```

```
class InterfaceController: WKInterfaceController {
...
    let suiteName = "group.com.guanshanliu.AppleWatchDemo"

    func readData() {
        let fileCoordinator = NSFileCoordinator()
        if let fileURL = presentedItemURL {
            fileCoordinator.coordinateReadingItemAtURL(fileURL, options: .allZeros, error: nil) { [unowned self] url in
                let data = NSDictionary(contentsOfURL: url)
                let value = (data?["value"] as? Int) ?? 0
                self.slider.setValue(Float(value))
            }
        }
    }

    func writeData(value: Int) {
        let fileCoordinator = NSFileCoordinator()
        if let fileURL = presentedItemURL {
            fileCoordinator.coordinateWritingItemAtURL(fileURL, options: .allZeros, error: nil) { url in
                let data = ["value": value]
                (data as NSDictionary).writeToURL(url, atomically: true)
            }
        }
    }

}
```

NSFilePresenter and NSFileCoordinator

Good:

» Changes are updated automatically

Bad:

» You may end up with deadlocks

» If a process is suspended mid coordinated I/O, it will never relinquish the ownership. Therefore, other processes get deadlocks.

» The containing app can observe UIApplicationDidEnterBackgroundNotification and cancel coordinated actions and remove file presenters.

» Extensions cannot do that.

For more, see Technical Note TN2408.

DON'T USE FILE COORDINATION APIS IN AN APP EXTENSION!

SCENARIO 4:
GET NOTIFIED WHEN SHARED DATA CHANGES, AND
NO DEADLOCKS

SAFEST WAYS FOR COORDINATED READS AND WRITES:

» Atomic save operations

```
(data as NSDictionary).writeToURL(url, atomically: true)
```

» SQLite, or Core Data

CFNOTIFICATIONCENTER
DARWIN NOTIFICATIONS

DARWIN NOTIFICATIONS

» C APIs

» Similar to NSNotification mechanism

» System-wide notifications

Get Darwin Notify Center

```
CFNotificationCenterRef center = CFNotificationCenterGetDarwinNotifyCenter();
```

Add an observer

```
CFNotificationCenterAddObserver(center, (__bridge const void *)(self), darwinNotificationCallback, (__bridge CFStringRef)name, NULL, CFNotificationSuspensionBehaviorDeliverImmediately);
```

Post a notification

```
CFNotificationCenterPostNotification(center, (__bridge CFStringRef)name, NULL, NULL, YES);
```

Remove an observer

```
CFNotificationCenterRemoveEveryObserver(center, (__bridge const void *)(self));
```

In the containing app:

In viewDidLoad()

1. Add an darwin notification observer. Update UI when receiving a notification.

```
DarwinNotificationWrapper.defaultCenter().addObserverForName(notificationName) { [unowned self] in
            self.updateUI()
 }
```

1. Initiate UI from data saved in the App Group.

updateUI()

in the containing app:

```
func updateUI() {
        var value = 0
        if let url = fileURL {
            let data = NSDictionary(contentsOfURL: url)
            value = data?[key] as? Int ?? 0
        }
        segment.selectedSegmentIndex = value;
 }

var fileURL: NSURL? {
        let groupURL = NSFileManager.defaultManager().containerURLForSecurityApplicationGroupIdentifier(suiteName)
        let fileURL = groupURL?.URLByAppendingPathComponent("data")
        return fileURL
}
```

In the containing app:

When selected segment index changes, update the data saved in the App Group, and post the darwin notification.

```
@IBAction func onSegmentValueChange(sender: AnyObject) {
        if let url = fileURL {
            let data = [key: segment.selectedSegmentIndex]
            (data as NSDictionary).writeToURL(url, atomically: true)
            DarwinNotificationWrapper.defaultCenter().postNotificationName(notificationName)
        }
}
```

In the WatchKit extension, almost exactly the same. One difference I made is that if file does not exist, wake up the containing app to get data.

```
func updateUI() {
        if let url = fileURL {
            let data = NSDictionary(contentsOfURL: url)
            let value = data?[key] as? Int ?? 0
            self.slider.setValue(Float(value))
        } else {
            updateUIFromParent()
        }
 }

func updateUIFromParent() {
        WKInterfaceController.openParentApplication([:]) { (info, error) in
            if let value = info[self.key] as? Int {
                self.slider.setValue(Float(value))
            }
        }
 }
```

MMWORMHOLE

» Created by Conrad Stoll

» Open-source framework available on GitHub

» A communication bridge between between an iOS or OS X extension and it's containing app

» Uses App Group and CFNotificationCenter Darwin Notifications

» pod 'MMWormhole'

OTHER DATA SHARING RELATED TOPICS:

» Keychain sharing

» Handoff

» Local & Remote Notifications

» iCloud

RESOURCES

» WatchKit documentations & sample codes from Apple

» WatchKit by Tutorials

» Architecting Your App
for the Apple Watch by @NatashaTheRobot

THANK YOU!

QUESTIONS?

Codes on GitHub

Guanshan Liu (@guanshanliu)