goto; conference

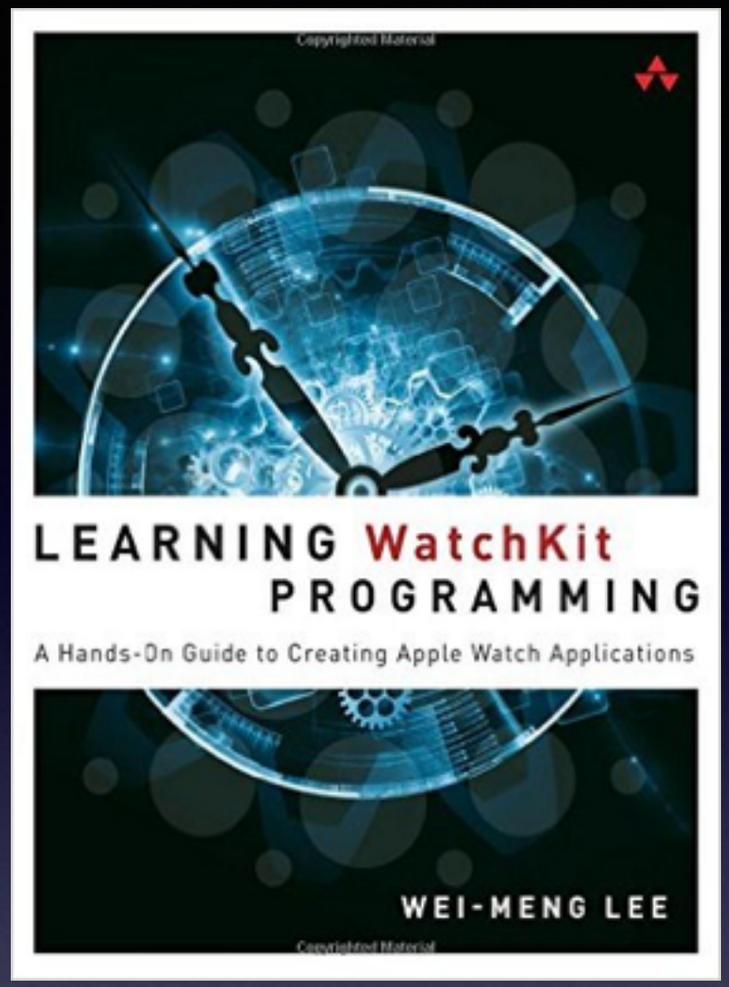
Developing Killer Apple Watch Apps

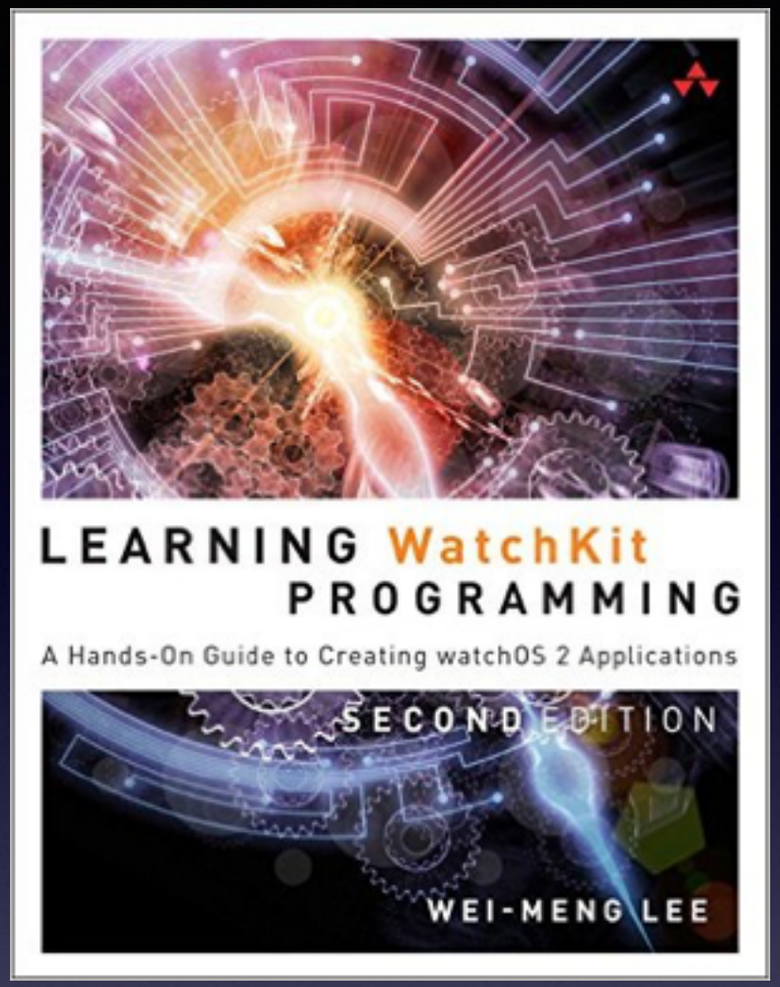
Wei-Meng Lee

goto; conference

Click ‘engage’ to rate sessions and ask questions

Join the conversation #gotocph





Wei-Meng Lee
weimenglee@learn2develop.net
Developer Learning Solutions, http://www.learn2develop.net

Agenda

- Getting started with Apple Watch development
- Cool stuff in watchOS 2.0
  - Complications
  - Watch Connectivity Framework
- Download all demos from:
  - `http://goo.gl/DtTOCc`

Smart Watches

Who will be the winner?

Whoever controls the smartphone platforms,
controls the smartwatch platform

And the winner(s) are…

Tools

- Xcode 7
  - Familiarity with iOS development
  - Swift (Objective-C is also supported)

Apple Watch Specifications

- Powered using a custom chip (S1) from Apple
- Sensors
  - Heart rate sensor on back of watch
  - Accelerometer, WiFi, and Bluetooth LE
- Charged wirelessly using a magnetic charger

Interacting with the Apple Watch

- **Digital Crown** - scroll through list of items, zooms in or out of images, etc. Also act as a Home button

- **Force Touch** - pressure sensitive screen; differentiates between tap and press; a press is like a right-click on a computer

- **Taptic Engine** - haptic feedback system; taps on your wrist to inform of notifications; vibrates when you rotate the digital crown

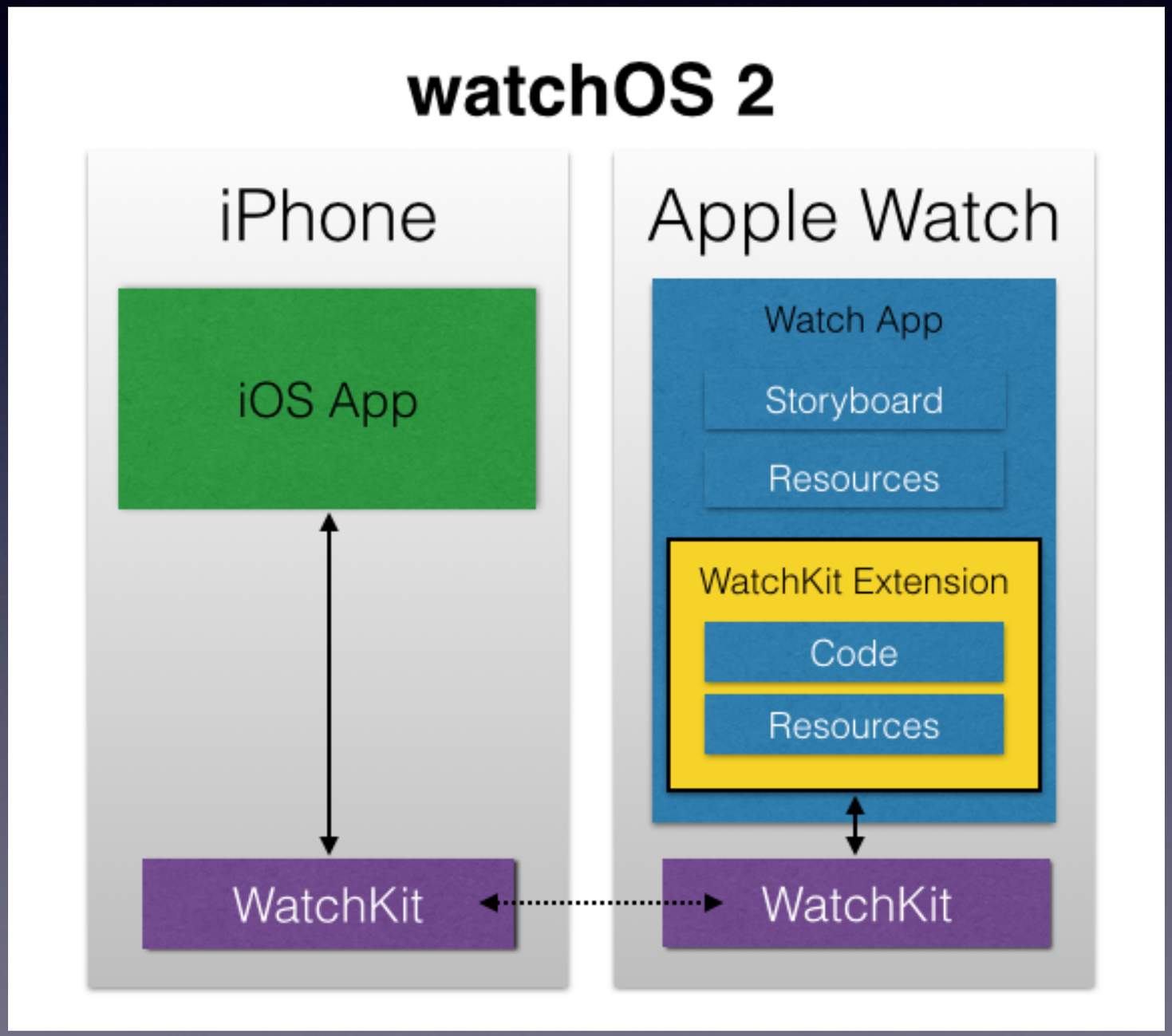
Apple Watch Sizes

- 38mm (height)
- 42mm (height)

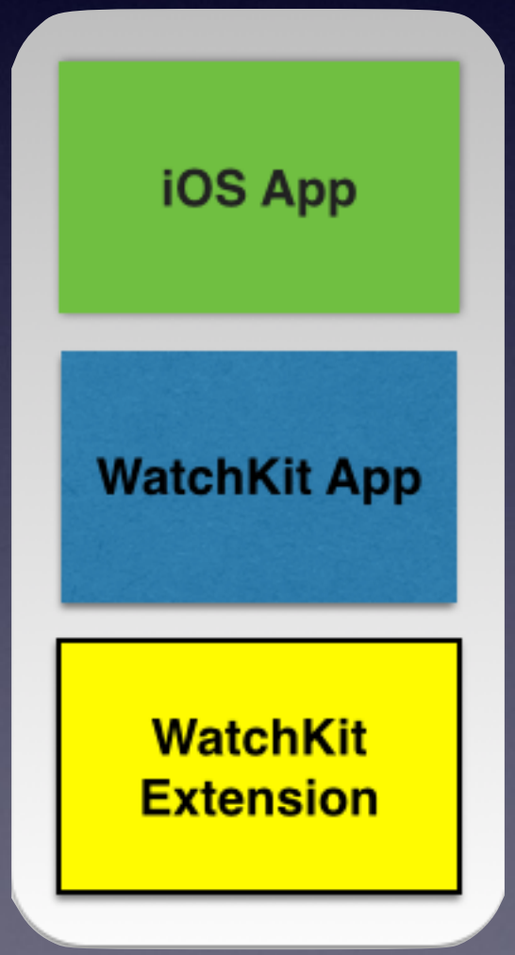
Good news! No AutoLayout!
Yet

WatchKit App Architecture

iOS App and Watch App



Deploying Apple Watch Apps



Types of Apple Watch Apps

- **Watch Apps**
  - native apps that run on the Apple Watch and interacts with the application logic running on the iPhone
- **Complications**
  - apps that supplies data to be displayed on the watch faces
- **Notifications**
  - displays notifications received by the iPhone (either local or remote notifications); apps can customise the notification interface
- **Glances**
  - a supplemental way for the user to view important information from your app. Glances does not support interactions with users – tapping on a glance will launch the Watch app

Hello, Apple Watch!

Complications

What are Complications?

- A complication is a function on a timepiece that does more than just tell the time

Quick Look at the Various Watch Faces

Complications Not Supported on the Following Watch Faces

- Motion
- X-Large
- Timelapse
- Astronomy
- Photo Album
- Photo
- Solar

Time Travel

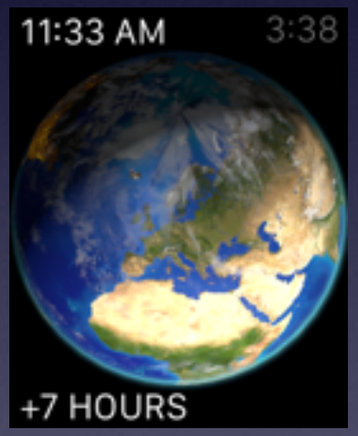
- Time Travel is a new feature in watchOS 2.0 that allows your application to display time-sensitive information on watch faces with complications. Turning the Digital Crown back and forth displays events past and future
- Lets you turn back time (or forward) to display information related to that specific time
- Examples:
  - Weather forecast for the next 2 days
  - Stock prices
    - Too bad you can’t display stock prices for the future!

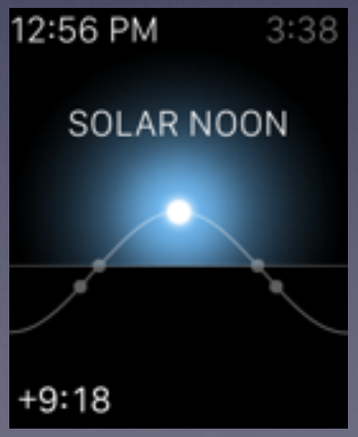
Watch Faces Not Compatible with Time Travel

- Motion
- X-Large
- Timelapse
- Photo Album
- Photo

Special Watch Faces that supports their own Time Travel

- Astronomy
- Solar





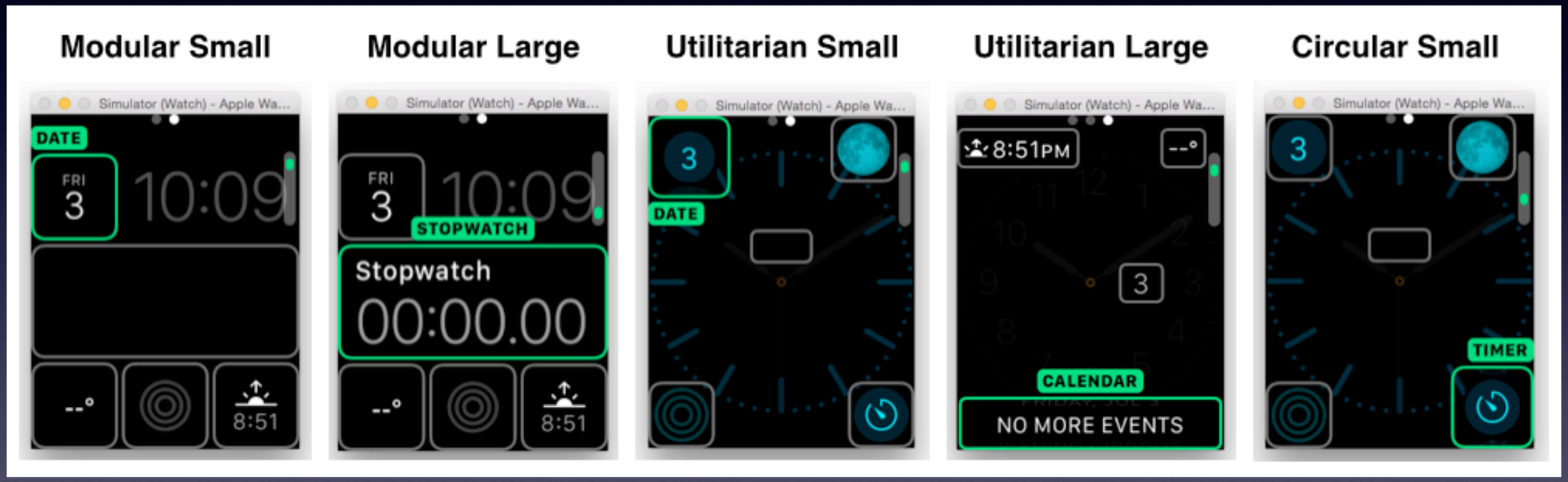
Supporting Complications in your Apple Watch App

- Use the ClockKit Framework
- Implement the **CLKComplicationDataSource** protocol

Implementing the CLKComplicationDataSource Protocol

- `getSupportedTimeTravelDirectionsForComplication:withHandler:`
  - Time travel directions
- `getTimelineStartDateForComplication:withHandler:`
  - Earliest date for complication data
- `getTimelineEndDateForComplication:withHandler:`
  - Latest date for complication data
- `getPrivacyBehaviorForComplication:withHandler:`
  - Privacy behavior - show or hide data when watch is locked
- `getCurrentTimelineEntryForComplication:withHandler:`
  - Current data to display
- `getTimelineEntriesForComplication:beforeDate:limit:withHandler:`
  - Data to display for past timeline
- `getTimelineEntriesForComplication:afterDate:limit:withHandler:`
  - Data to display for future timeline
- `getNextRequestedUpdateDateWithHandler:`
  - Next update time for timeline data
- `getPlaceholderTemplateForComplication:withHandler:`
  - Static display for complication

Placement for Complications

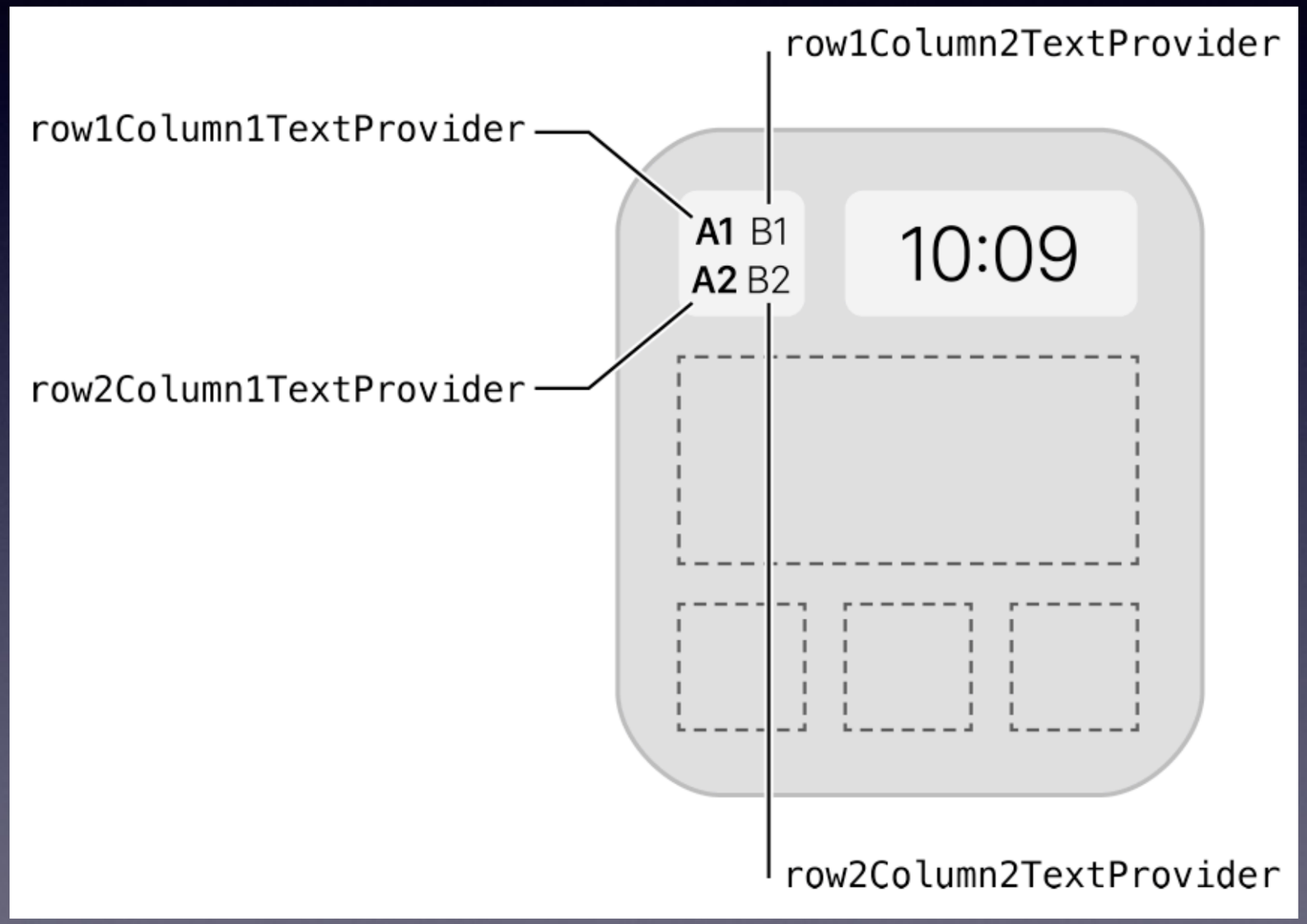


Complication Templates

```
CLKComplicationTemplateModularSmallColumnsText
CLKComplicationTemplateModularSmallRingImage
CLKComplicationTemplateModularSmallRingText
CLKComplicationTemplateModularSmallSimpleImage
CLKComplicationTemplateModularSmallSimpleText
CLKComplicationTemplateModularSmallStackImage
CLKComplicationTemplateModularSmallStackText

CLKComplicationTemplateModularLargeColumns
CLKComplicationTemplateModularLargeStandardBody
CLKComplicationTemplateModularLargeTable
CLKComplicationTemplateModularLargeTallBody

CLKComplicationTemplateCircularSmallRingImage
CLKComplicationTemplateCircularSmallRingText
CLKComplicationTemplateCircularSmallSimpleImage
CLKComplicationTemplateCircularSmallSimpleText
CLKComplicationTemplateCircularSmallStackImage
CLKComplicationTemplateCircularSmallStackText

CLKComplicationTemplateUtilitarianSmallFlat
CLKComplicationTemplateUtilitarianSmallRingImage
CLKComplicationTemplateUtilitarianSmallRingText
CLKComplicationTemplateUtilitarianSmallSquare

CLKComplicationTemplateUtilitarianLargeFlat
```

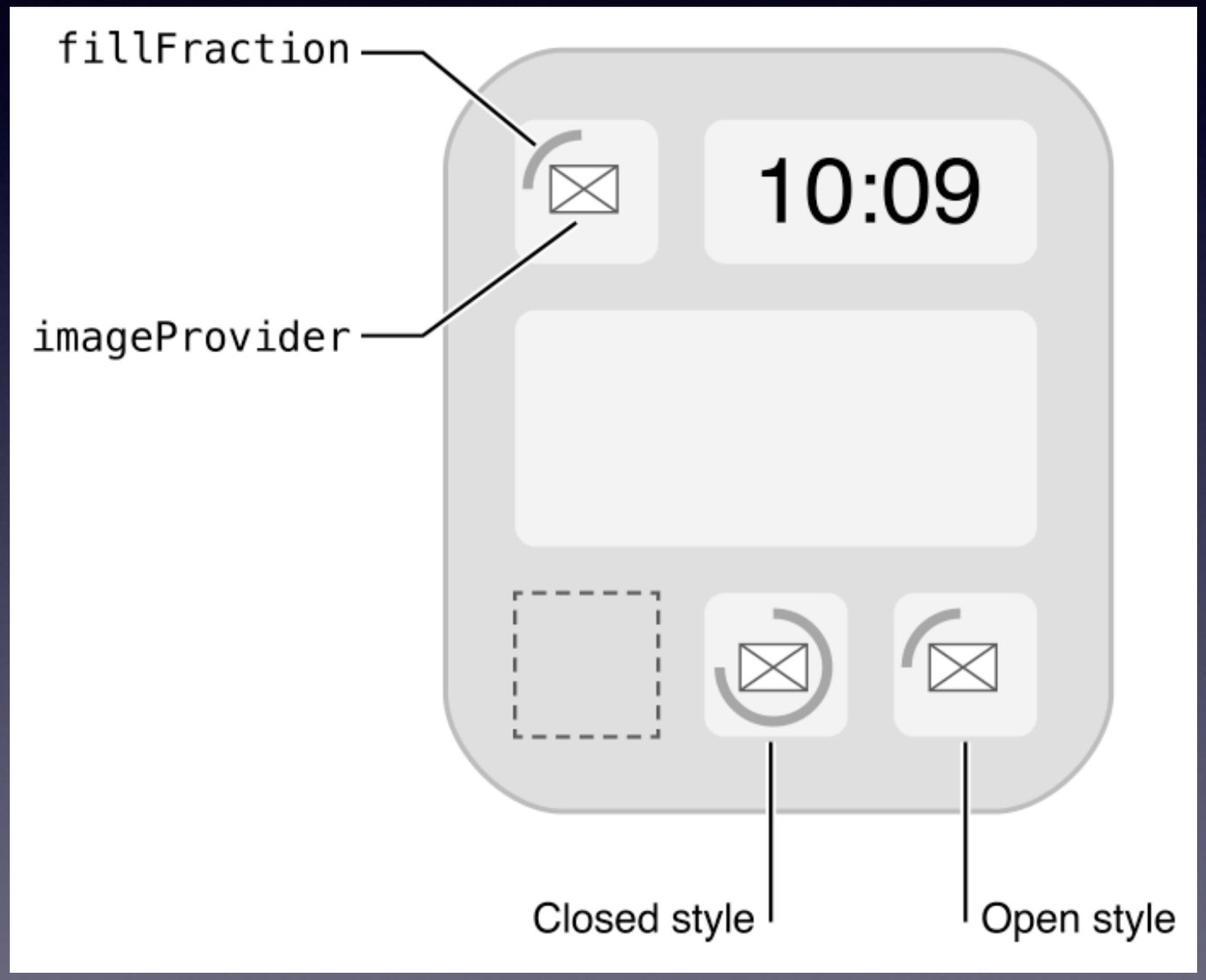
Complication Templates

CLKComplicationTemplateModularSmallColumnsText



Complication Templates

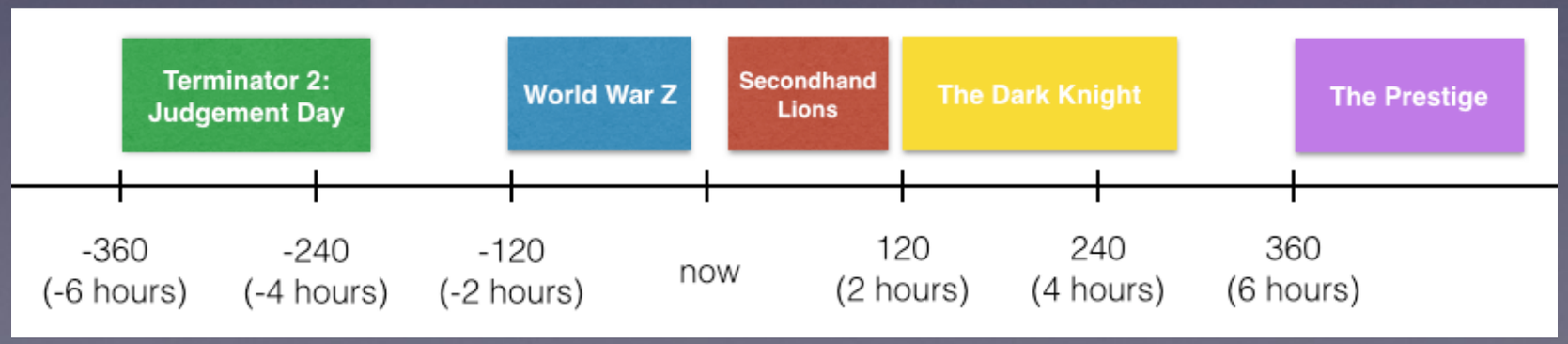
`CLKComplicationTemplateModularSmallRingImage`



Complications Demo

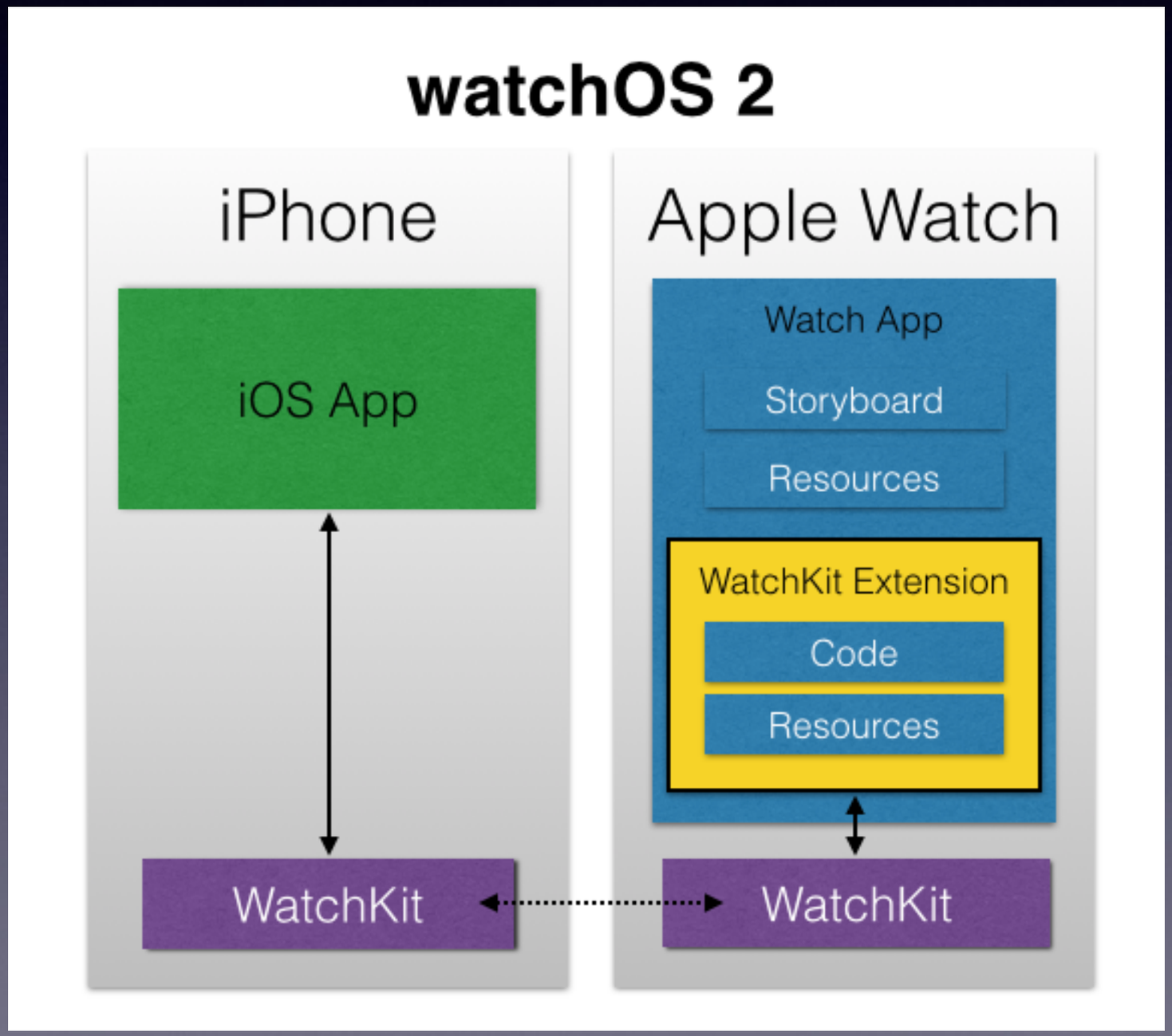
Movie Data

```
let movies = [
    Movie(movieName: "Terminator 2: Judgment Day",
        runningTime: 137 * MINUTE,
        runningDate: NSDate(timeIntervalSinceNow: -360 * MINUTE),
        rating:8),
    Movie(movieName: "World War Z",
        runningTime: 116 * MINUTE,
        runningDate: NSDate(timeIntervalSinceNow: -120 * MINUTE),
        rating:7),
    Movie(movieName: "Secondhand Lions",
        runningTime: 90 * MINUTE,
        runningDate: NSDate(timeIntervalSinceNow: 10 * MINUTE),
        rating:8),
    Movie(movieName: "The Dark Knight",
        runningTime: 152 * MINUTE,
        runningDate: NSDate(timeIntervalSinceNow: 120 * MINUTE),
        rating:9),
    Movie(movieName: "The Prestige",
        runningTime: 130 * MINUTE,
        runningDate: NSDate(timeIntervalSinceNow: 360 * MINUTE),
        rating:8),
]
```



Communicating between the iOS App and the Watch App

iOS and Watch Apps are on Separate Islands

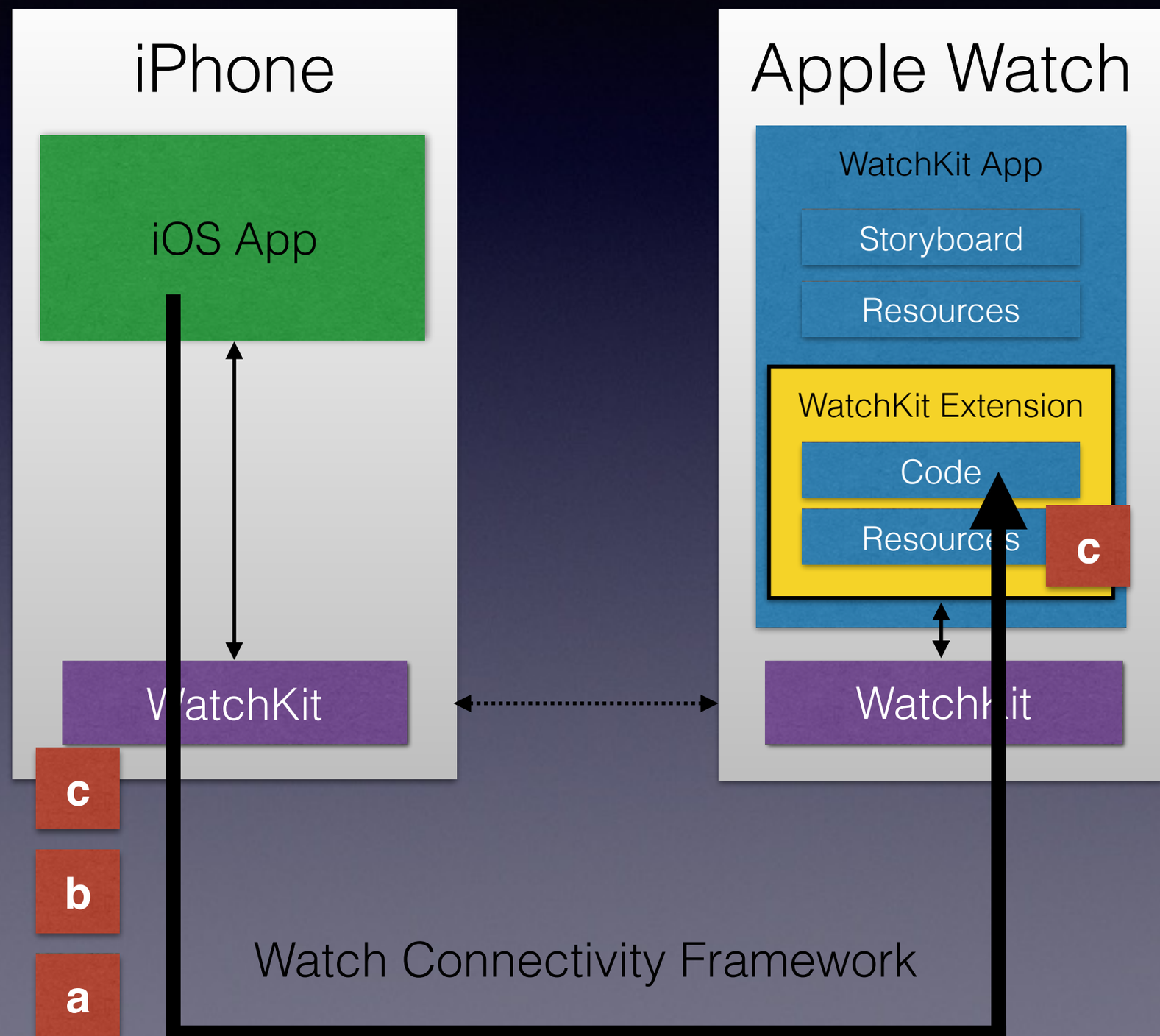


Watch Connectivity Framework

- **Background Transfers**
  - Application Context
  - User Info
  - File Transfer
- **Interactive Messaging**
  - Send Message

Background Transfers

Application Context

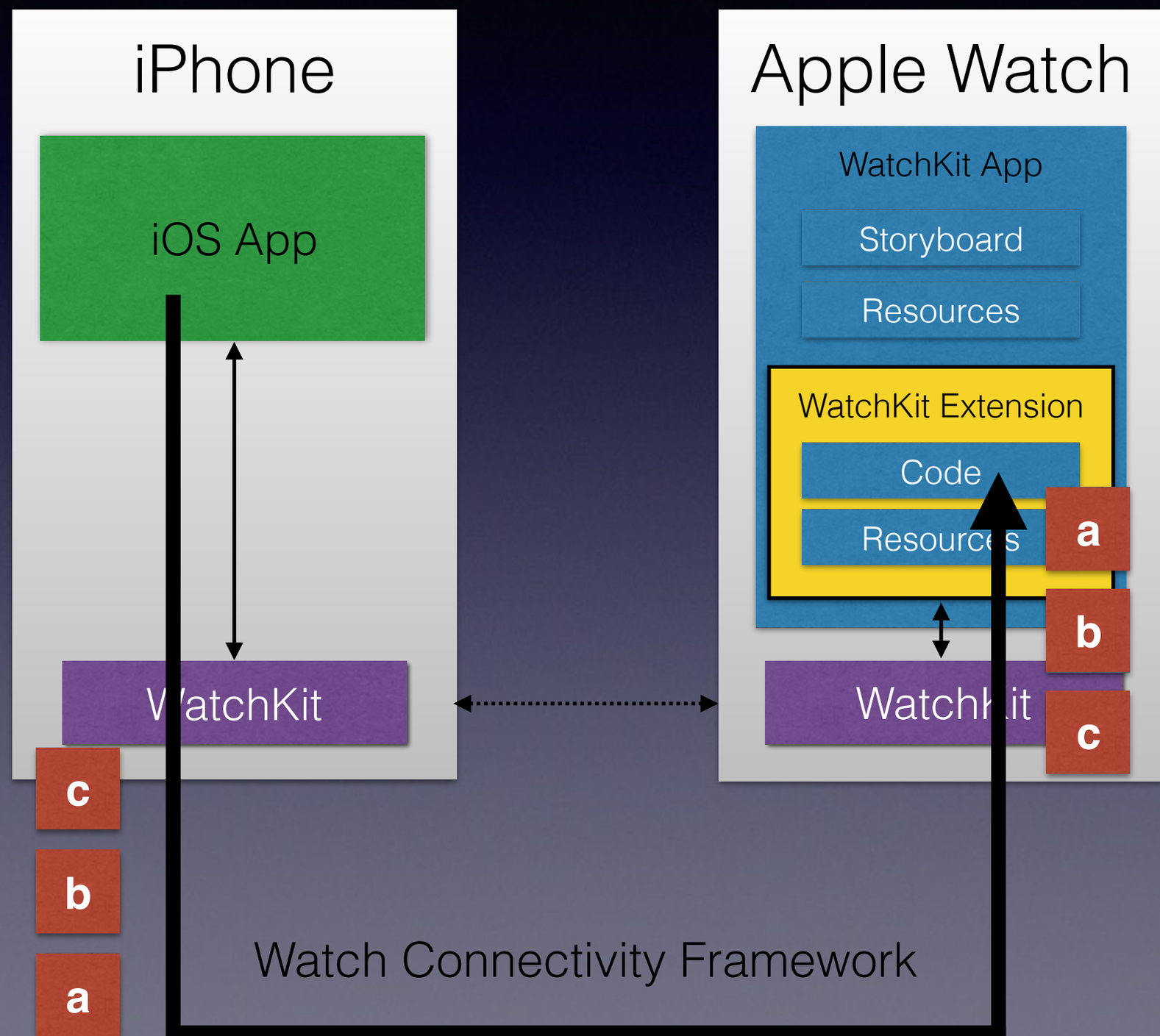


Use Case

- Useful for updating application states, such as glances on the Apple Watch

Background Transfers

User Info

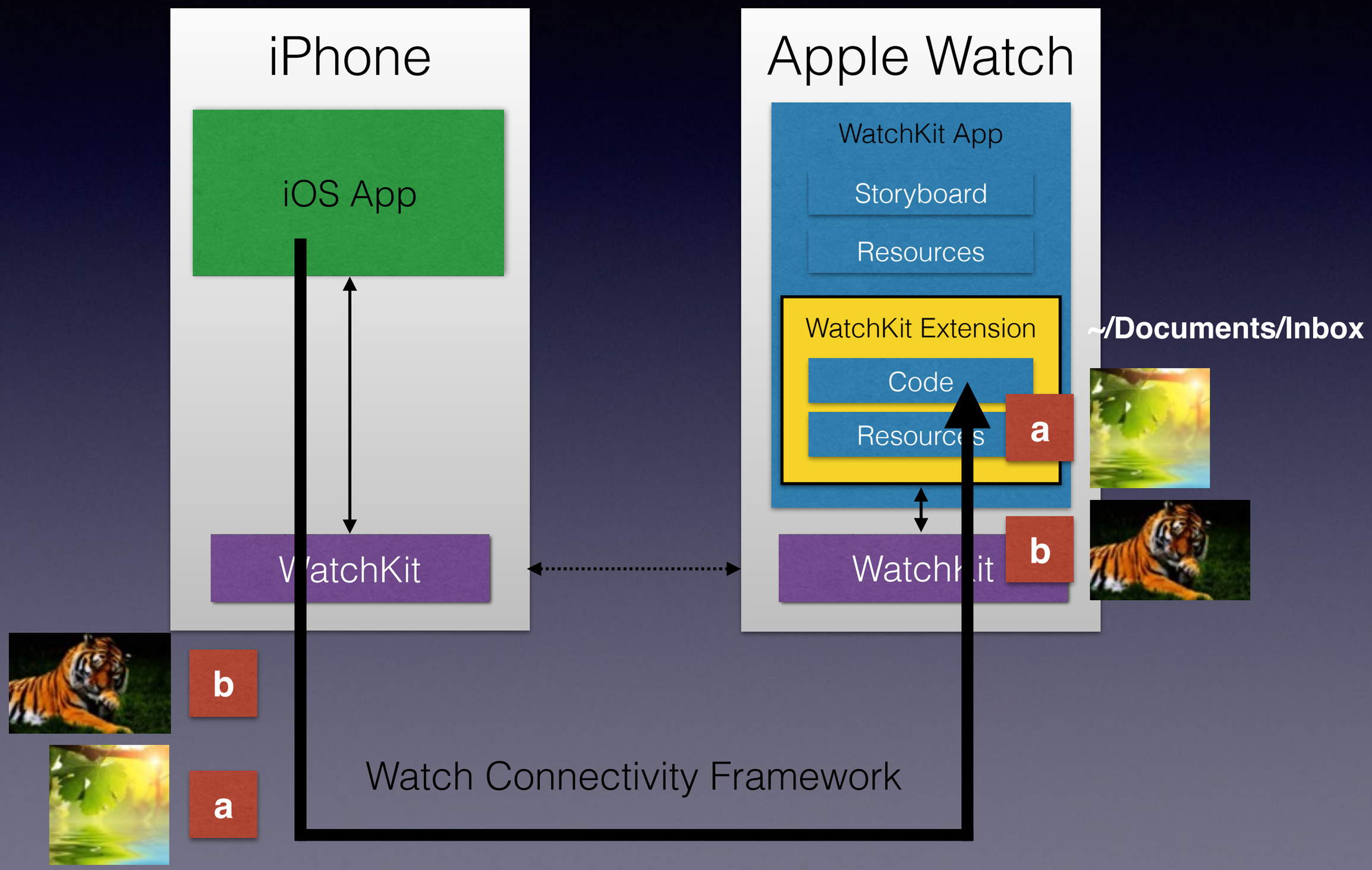


Use Case

- Useful for games where changes on one device must be synchronized on the other device

Background Transfers

File Transfer

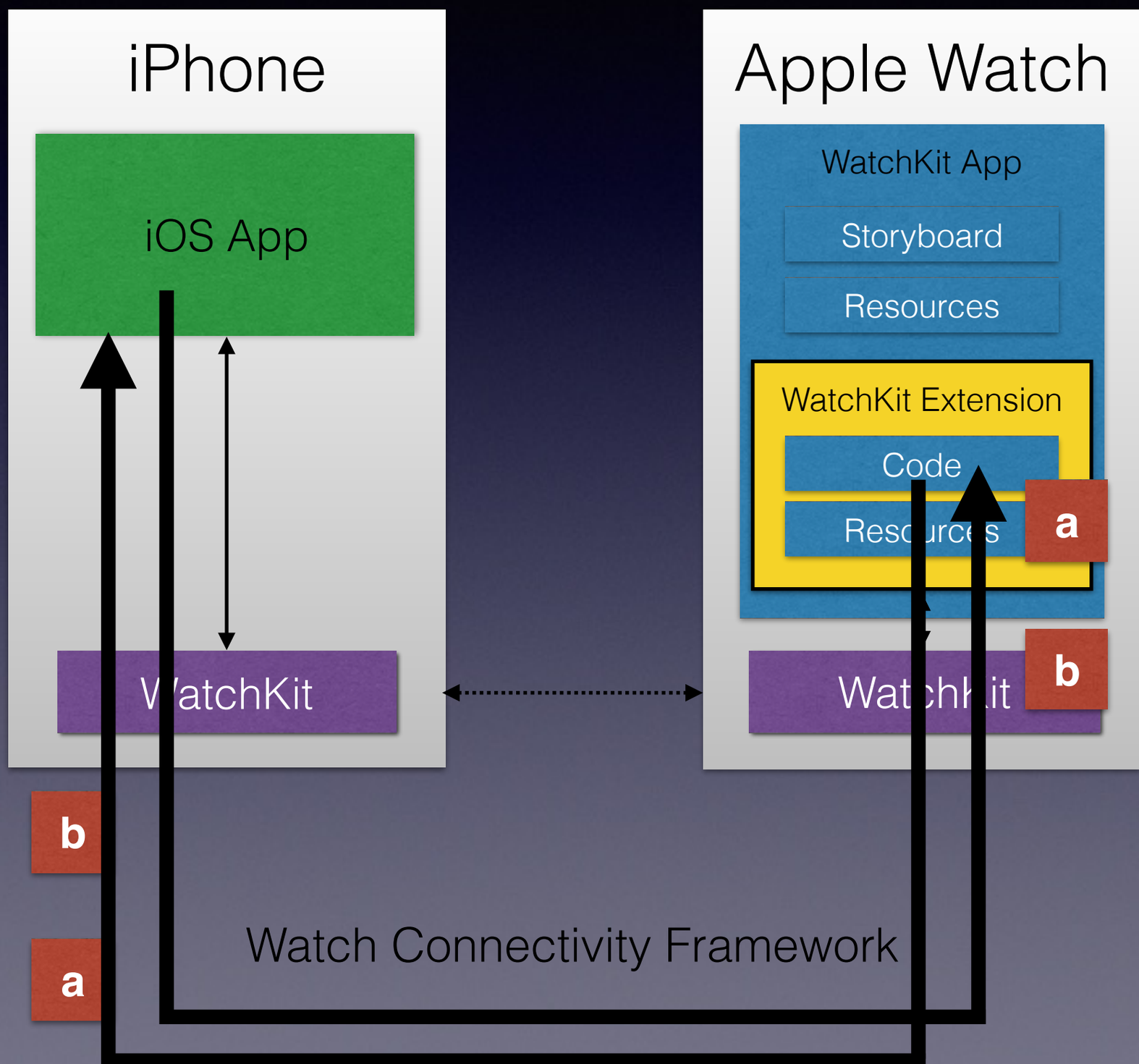


Use Case

- Useful for transferring files between devices

Interactive Communications

Send Message



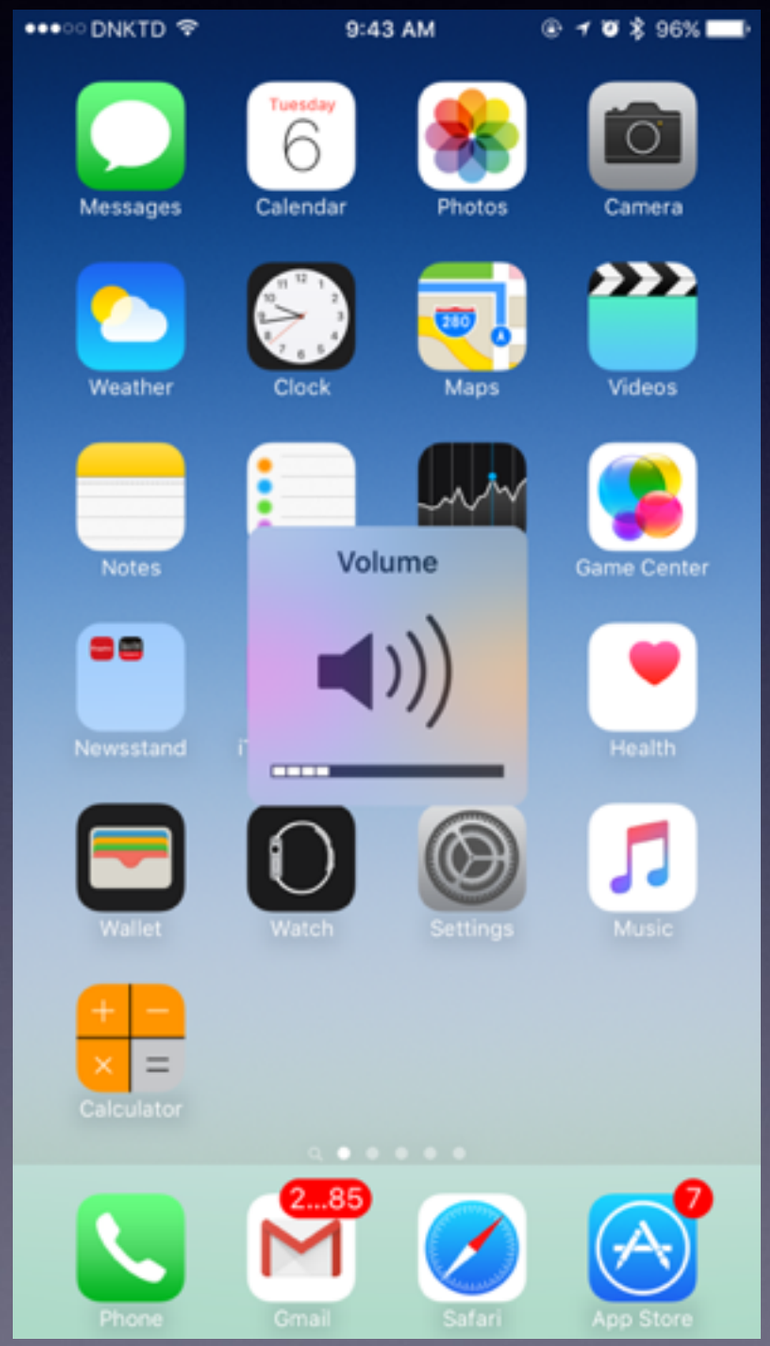
- When Apple Watch sends a message to the iPhone, it wakes up the containing iPhone app in the background (if it is not in the foreground)
- When the iPhone sends a message to the watch, the watch app must be in the foreground, else it fails

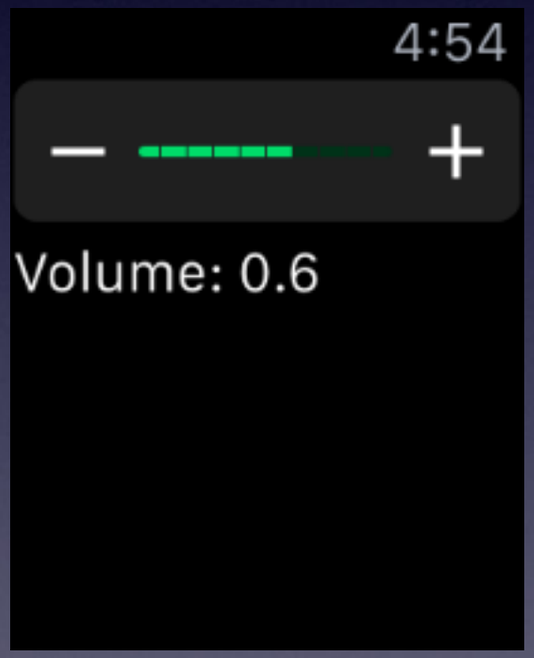
Use Case

- Useful for cases where you need to control your iPhone through the Apple Watch. For example, you can create an app on the Apple Watch to adjust the volume on the iPhone, or perform some home automation tasks.

Demos

Volume Controller





Killer Watch Apps

- Don’t cram your existing apps onto the watch!
- Think of the watch as an extension of your phone
  - Use it to enhance the experience of the mobile app
- Home automation
  - “Hey Siri”
  - “Launch Garage app”
  - Tap the “**Unlock Garage Door**”

goto; conference

Please

Remember to rate this session

Thank you!

Join the conversation #gotocph