App Frameworks

#WWDC15

WatchKit In-Depth, Part 1

Session 207

Andrew Platzer WatchKit Engineer
Forest Hill WatchKit Engineer

WatchKit In-Depth, Part 1

WatchKit In-Depth, Part 1

Architecture

Resources and Data

Migration

Enhancements

Architecture

Watch App for watchOS 2

Architecture

Watch App for watchOS 2

Components

- iOS application
- watchOS WatchKit extension
- watchOS Watch application

Architecture

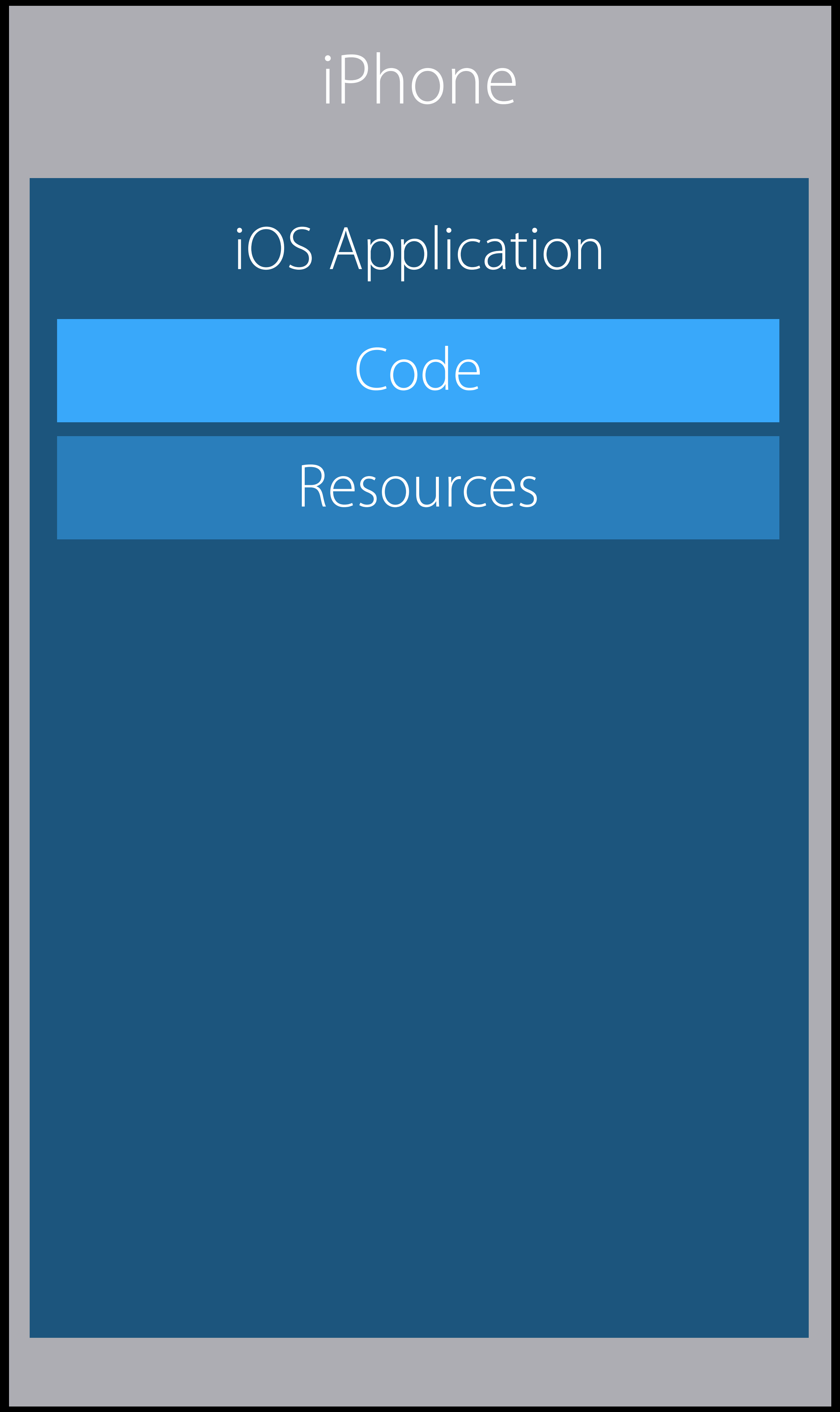
Watch App for watchOS 2

Architecture

Watch App for watchOS 2

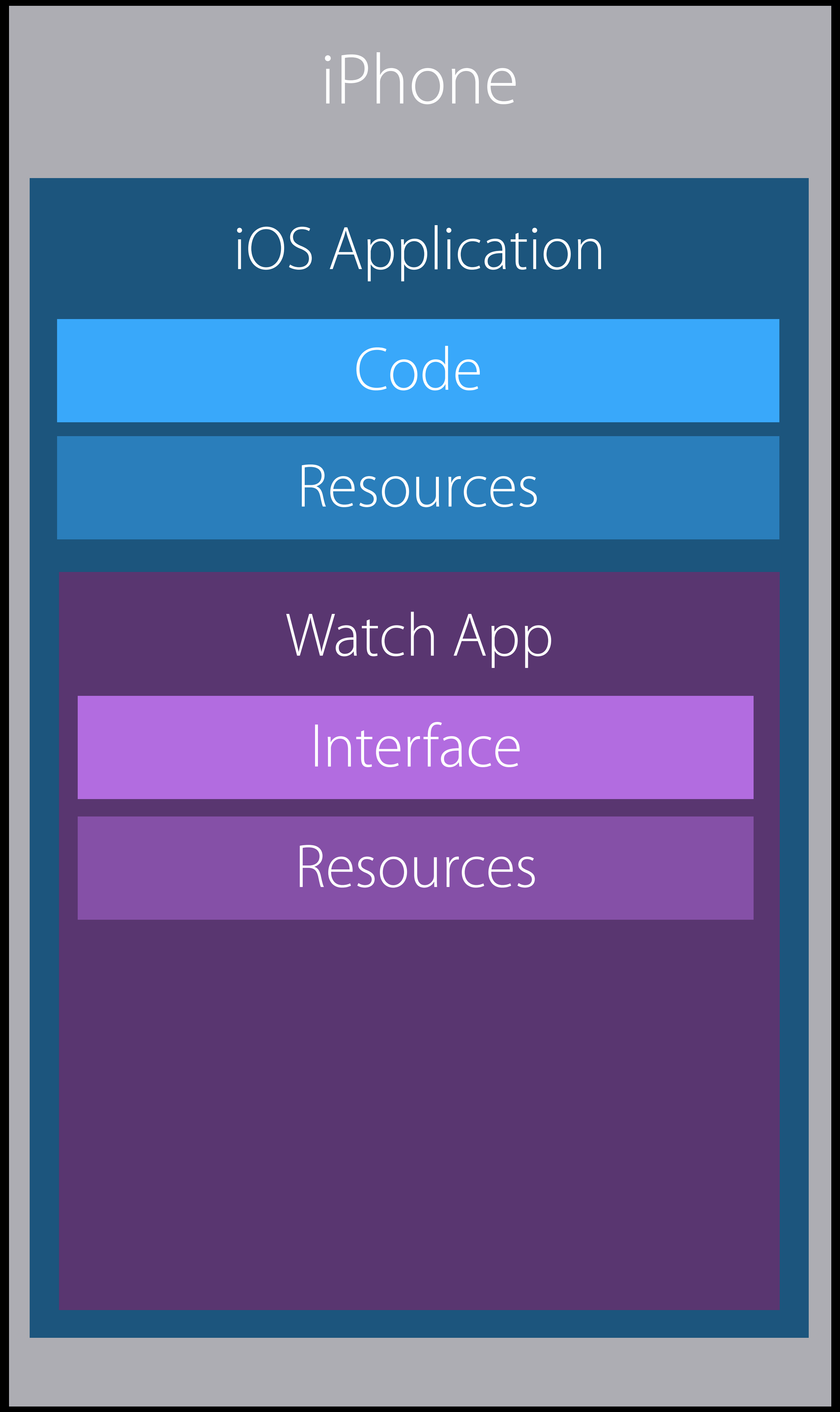
Architecture

Watch App for watchOS 2



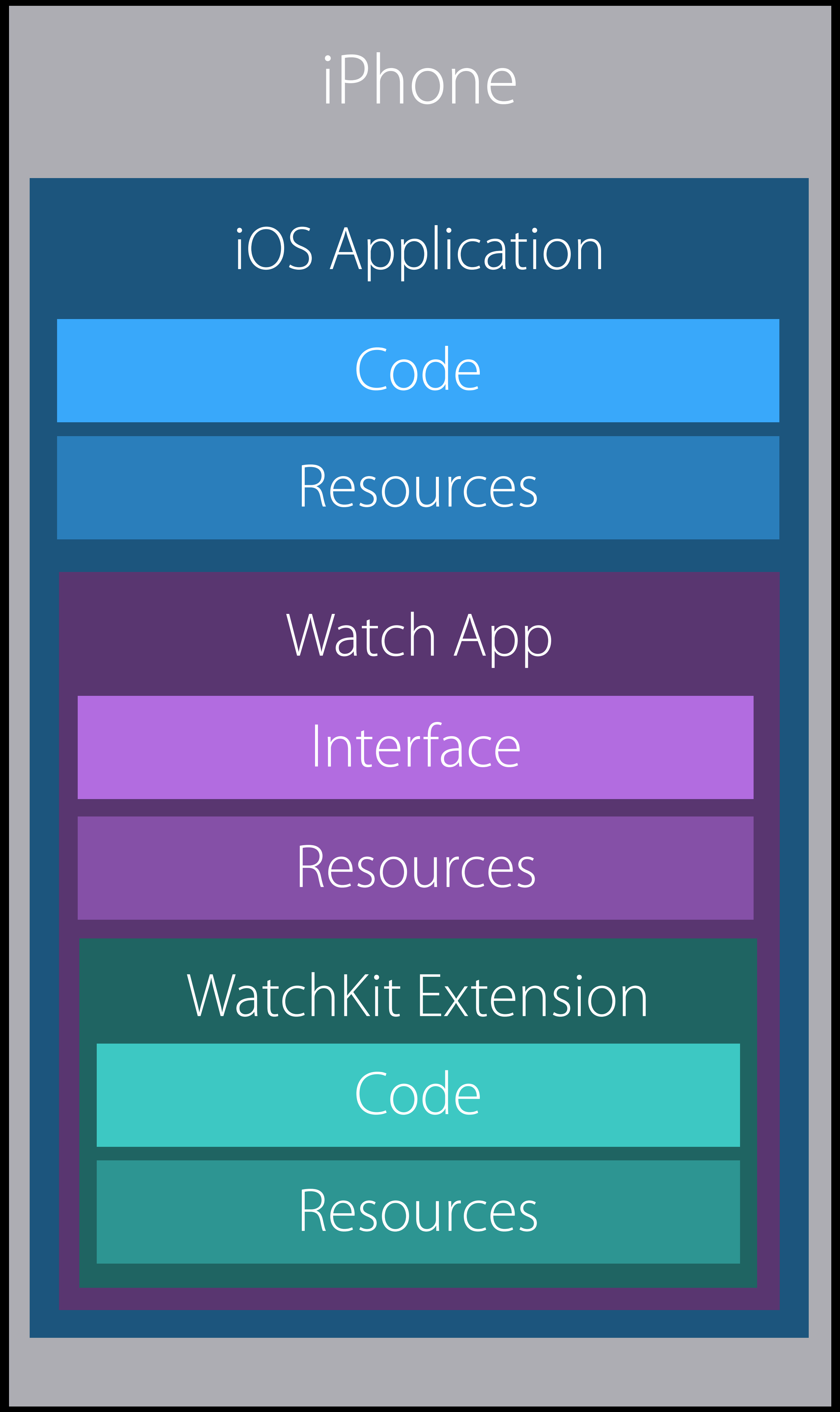
Architecture

Watch App for watchOS 2



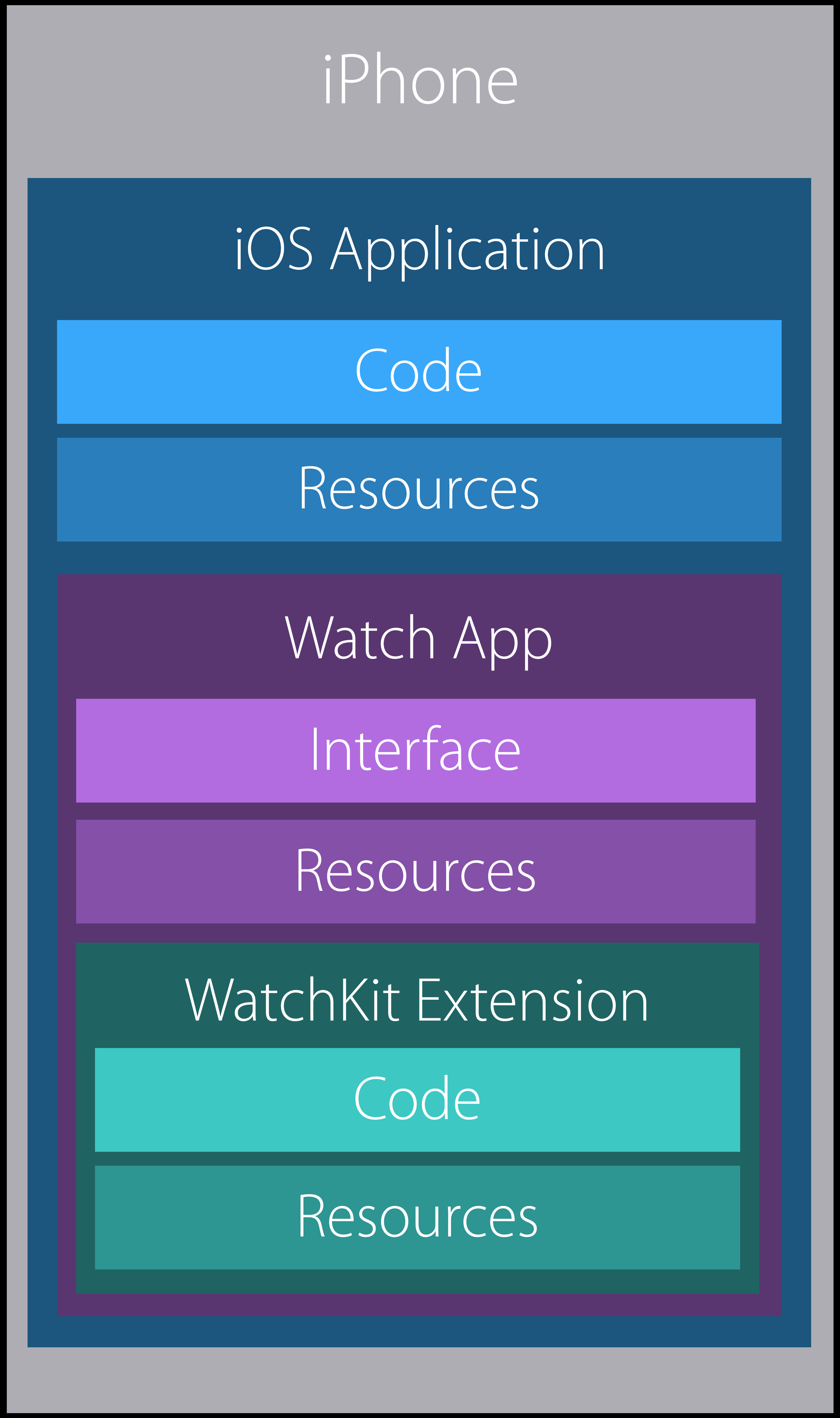
Architecture

Watch App for watchOS 2



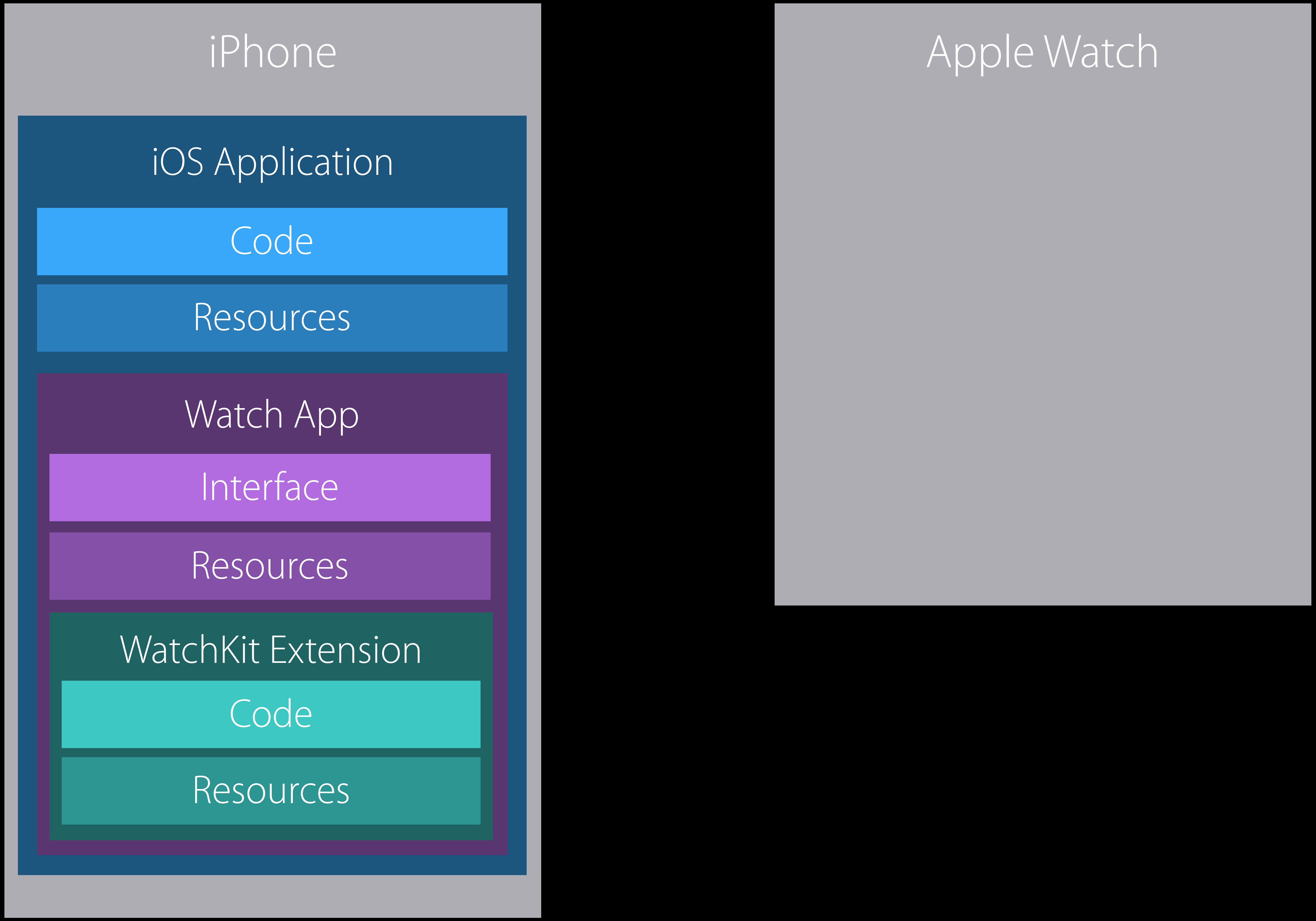
Architecture

Watch App for watchOS 2



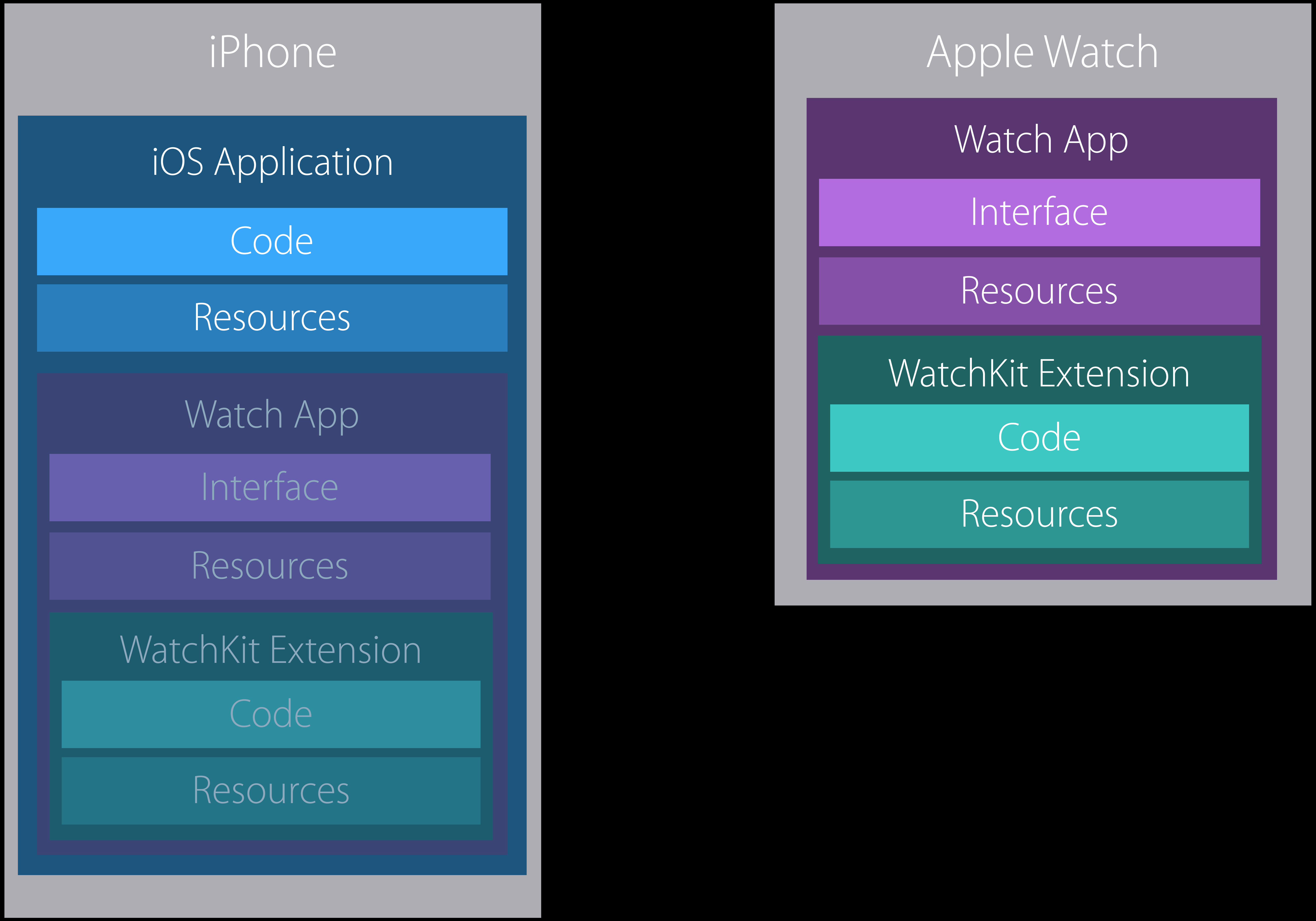
Architecture

Watch App for watchOS 2



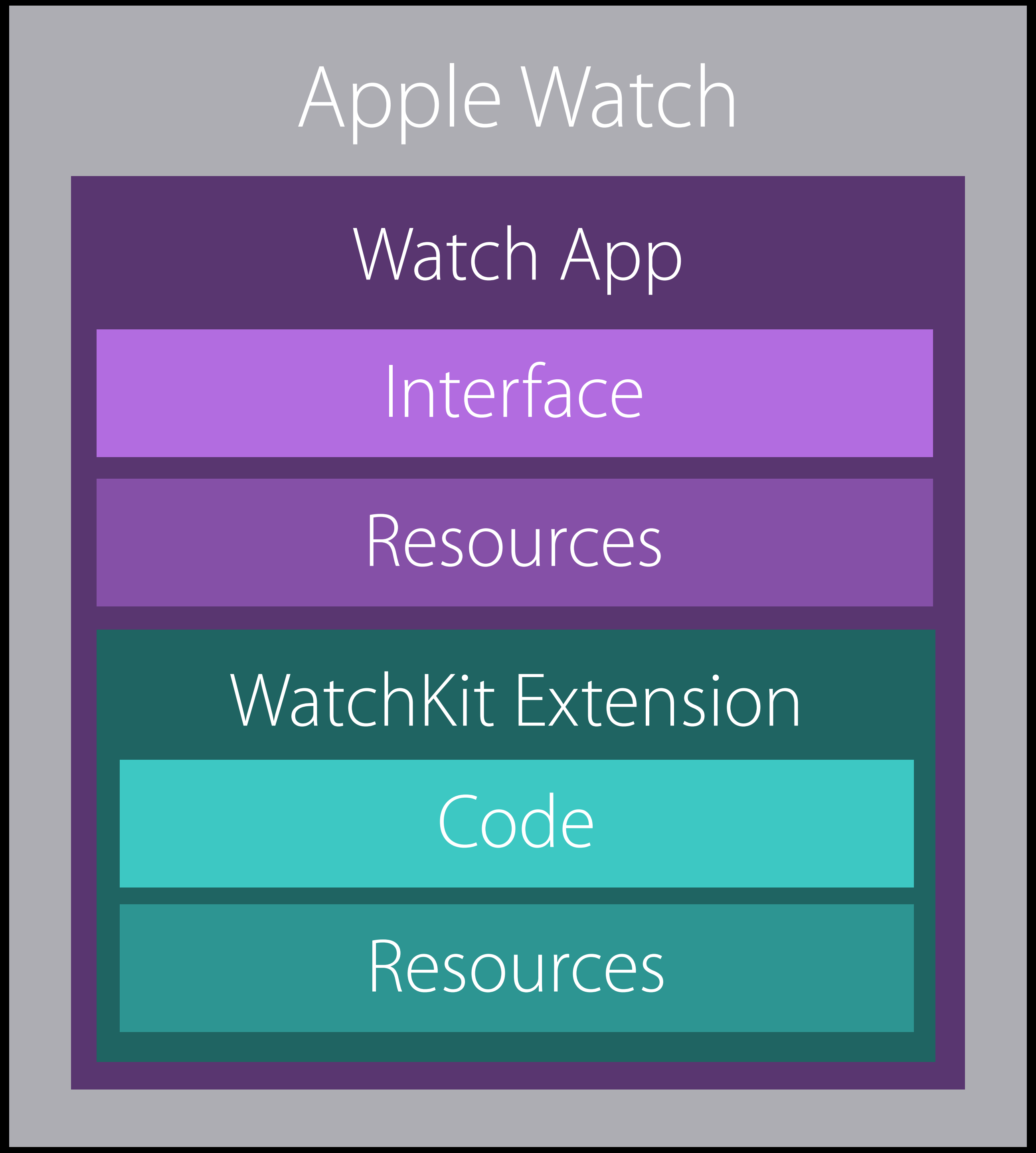
Architecture

Watch App for watchOS 2



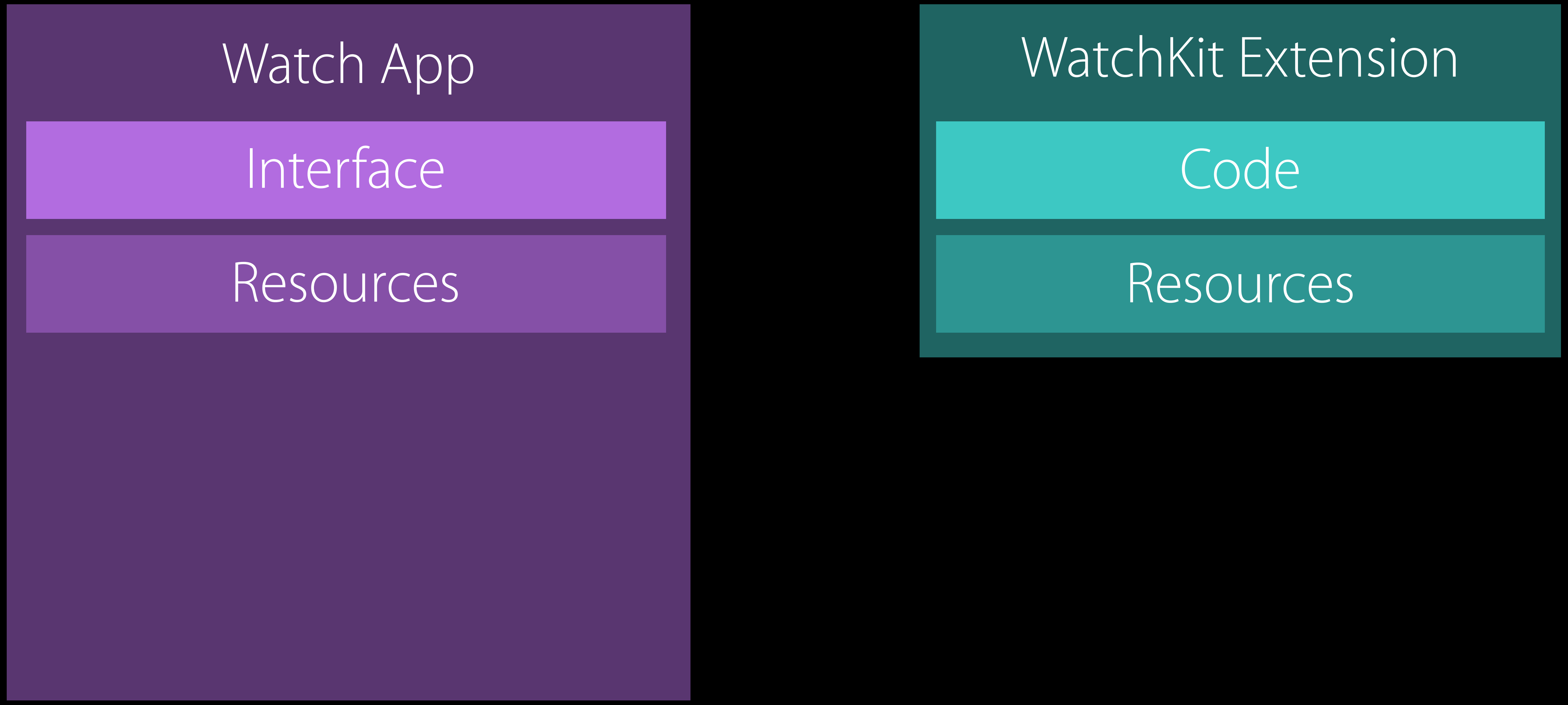
Architecture

Watch App for watchOS 2



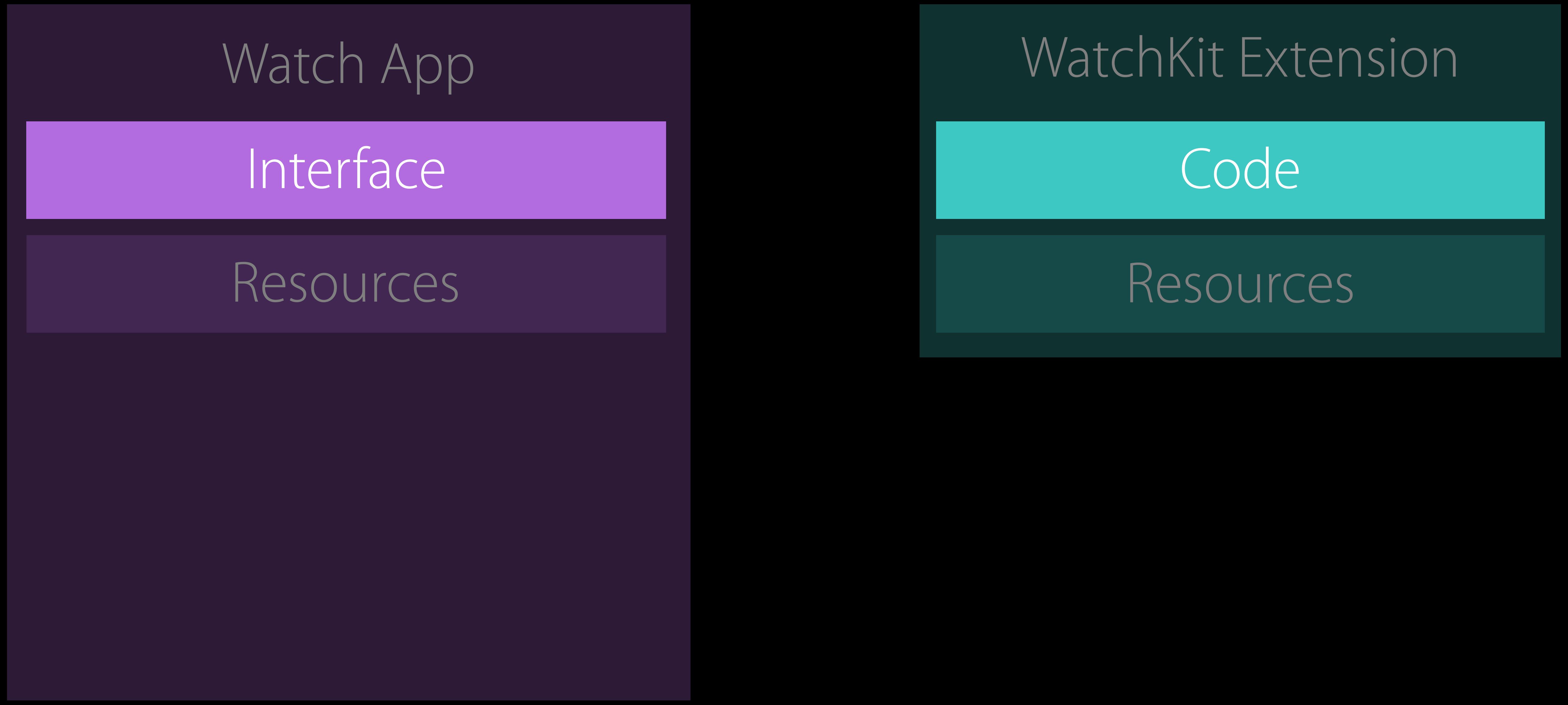
Architecture

Watch App for watchOS 2



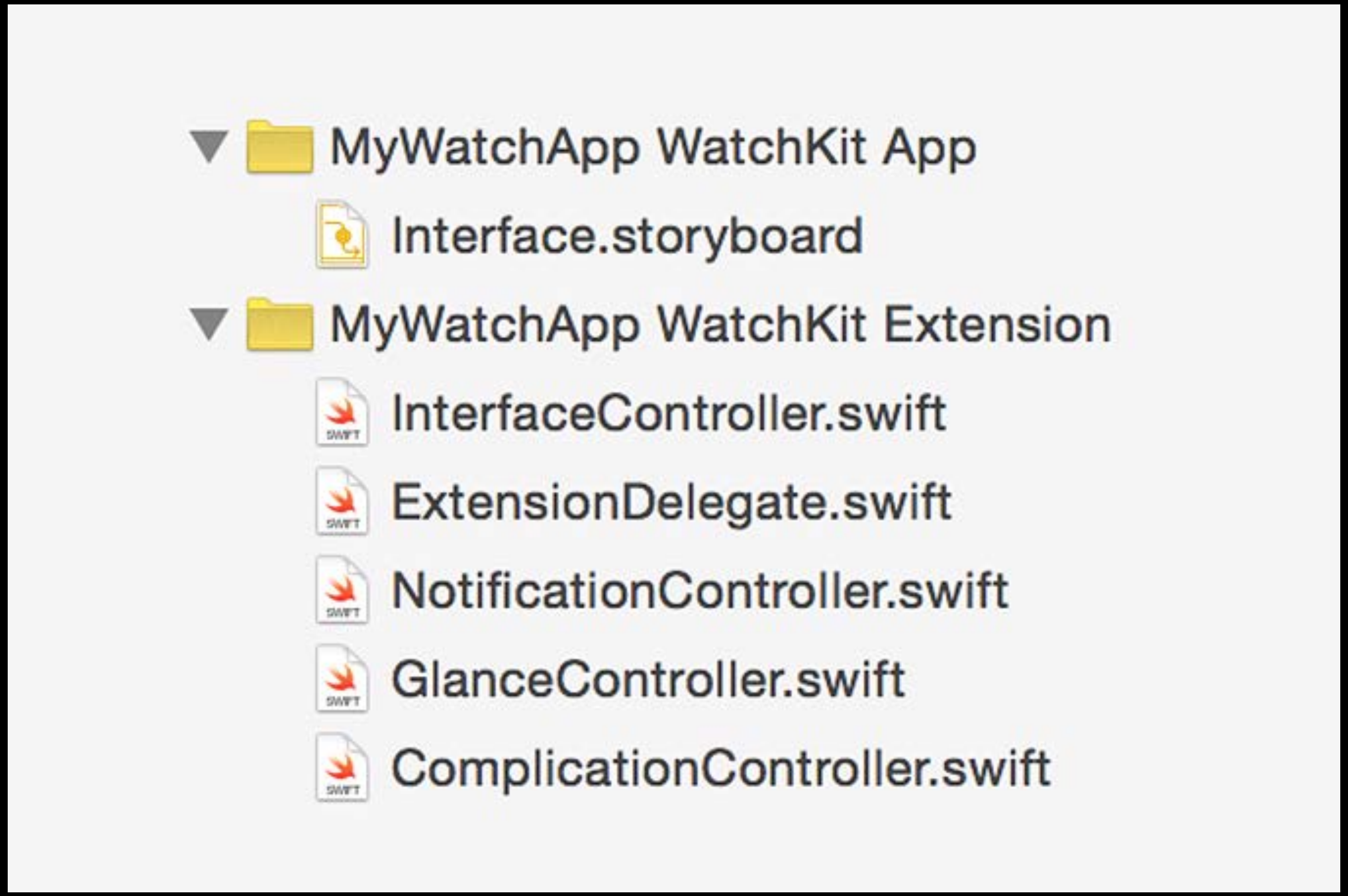
Architecture

Watch App for watchOS 2



Architecture

WatchKit App



```
▼ MyWatchApp WatchKit App
    Interface.storyboard
▼ MyWatchApp WatchKit Extension
    InterfaceController.swift
    ExtensionDelegate.swift
    NotificationController.swift
    GlanceController.swift
    ComplicationController.swift
```

Architecture

Interface.storyboard

Architecture

Interface.storyboard

Edit with Interface Builder

Architecture

Interface.storyboard

Edit with Interface Builder

Interface Elements

- Label
- Image
- Button
- Switch
- Slider
- Date Label
- Timer Label
- Separator
- Group
- Table
- Menus
- Map

Architecture

NEW

Interface.storyboard

Edit with Interface Builder

Interface Elements

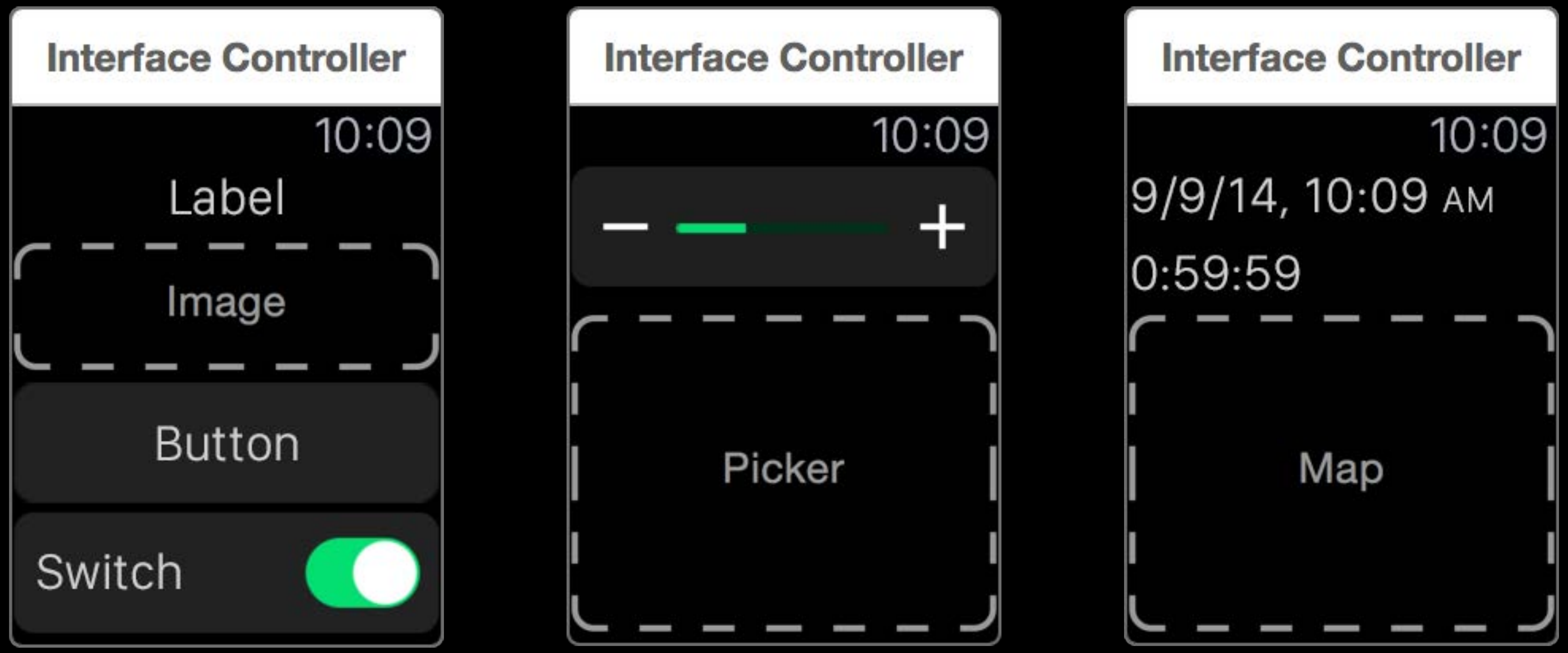
- Label
- Image
- Button
- Switch
- Slider
- Date Label
- Picker
- Timer Label
- Separator
- Group
- Table
- Menus
- Map
- Movie

Architecture

Interface.storyboard

Architecture

Interface.storyboard

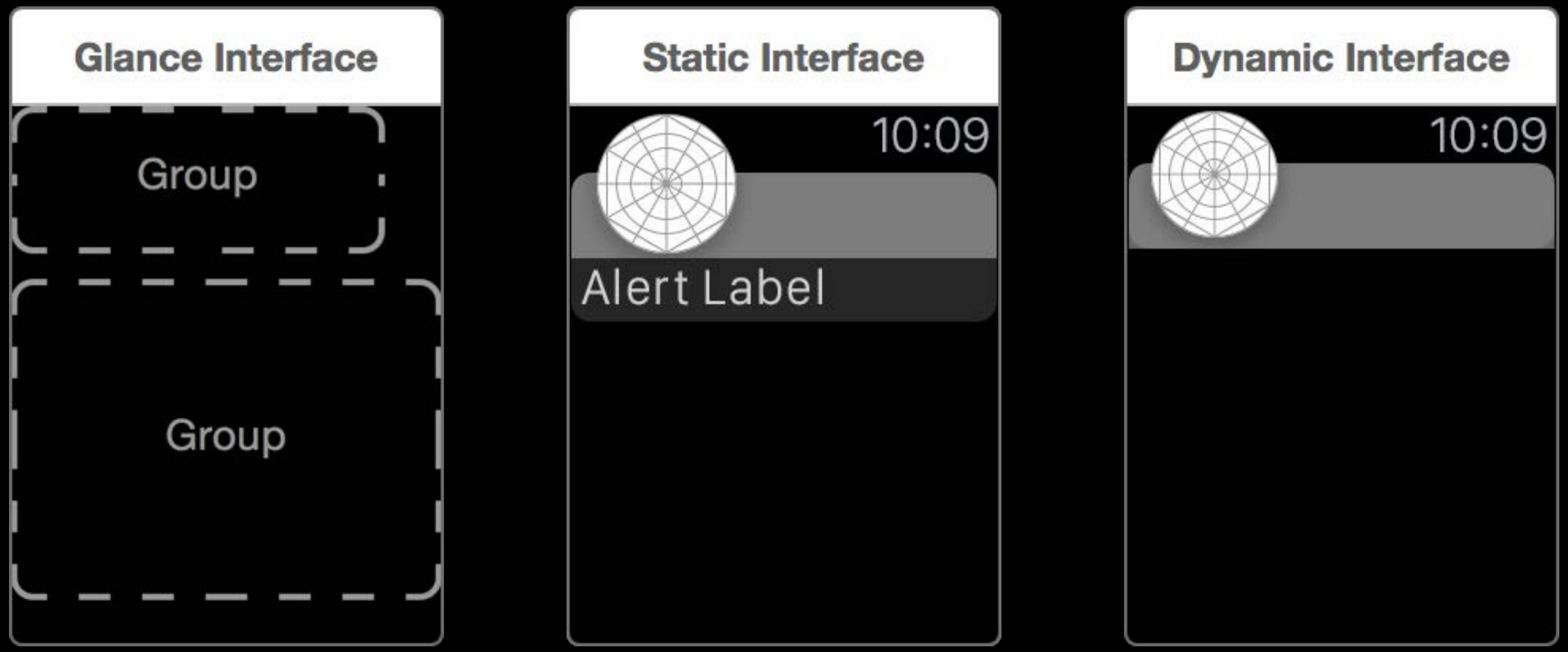


Architecture

Interface.storyboard

Architecture

Interface.storyboard



Architecture

WatchKit Extension

Roles

- Application
- Glance
- Notification
- Complication

Architecture

WatchKit Extension

Roles

• Application	`WKInterfaceController`
• Glance	`WKInterfaceController`
• Notification	`WKUserNotificationInterfaceController`
• Complication	`CLKComplicationDataSource`

Architecture

WatchKit Extension

Roles

Application	`WKInterfaceController`
Glance	`WKInterfaceController`
Notification	`WKUserNotificationInterfaceController`
Complication	`CLKComplicationDataSource`

Architecture

WatchKit Extension

Architecture

WatchKit Extension

WKInterfaceController

Architecture

WatchKit Extension

WKInterfaceController

- Interface properties

Architecture

WatchKit Extension

WKInterfaceController

- Interface properties
- Menu handling

Architecture

WatchKit Extension

WKInterfaceController

- Interface properties
- Menu handling
- Controller navigation and paging

Architecture

WatchKit Extension

WKInterfaceController

- Interface properties
- Menu handling
- Controller navigation and paging
- Controller modal presentation

Architecture

WatchKit Extension

WKInterfaceController

- Interface properties
- Menu handling
- Controller navigation and paging
- Controller modal presentation
- Alert and action sheets

Architecture

WatchKit Extension

WKInterfaceController

- Interface properties
- Menu handling
- Controller navigation and paging
- Controller modal presentation
- Alert and action sheets
- System UI — text input, video, audio

Architecture

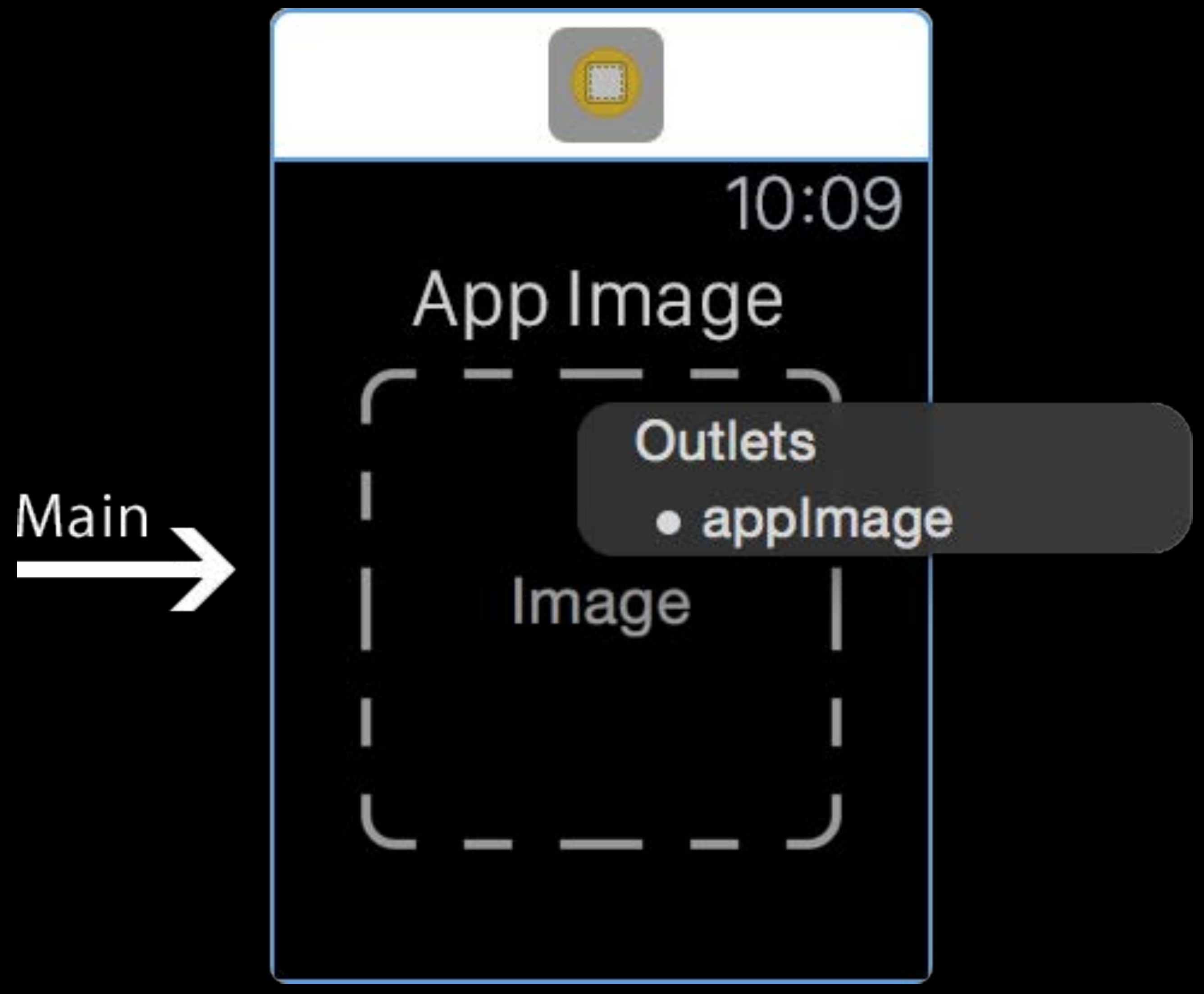
Interface properties

```
class InterfaceController: WKInterfaceController {
    @IBOutlet weak var appImage: WKInterfaceImage!
    override func willActivate() {
        super.willActivate()
        appImage.setImageNamed("ApplicationImage")
    }
```

Architecture

Interface properties

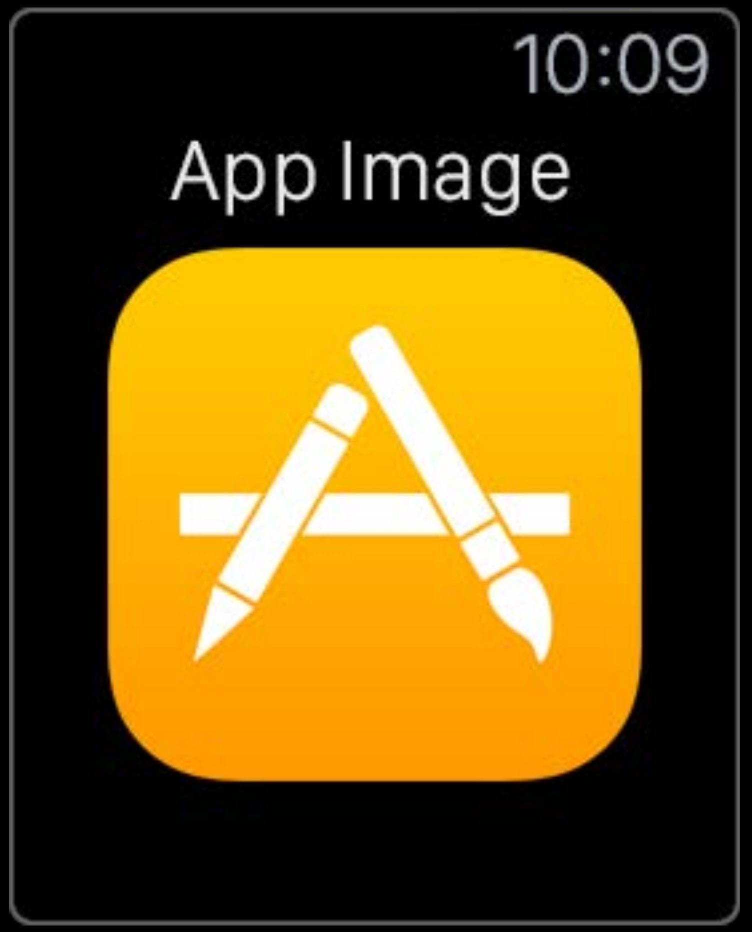
```
class InterfaceController: WKInterfaceController {
    @IBOutlet weak var appImage: WKInterfaceImage!
    override func willActivate() {
        super.willActivate()
        appImage.setImageNamed("ApplicationImage")
    }
```



Architecture

Interface properties

```
class InterfaceController: WKInterfaceController {
    @IBOutlet weak var appImage: WKInterfaceImage!
    override func willActivate() {
        super.willActivate()
        appImage.setImageNamed("ApplicationImage")
    }
```



Resources and Data

Application and Extension

Resources

Static Resources

Multiple locations

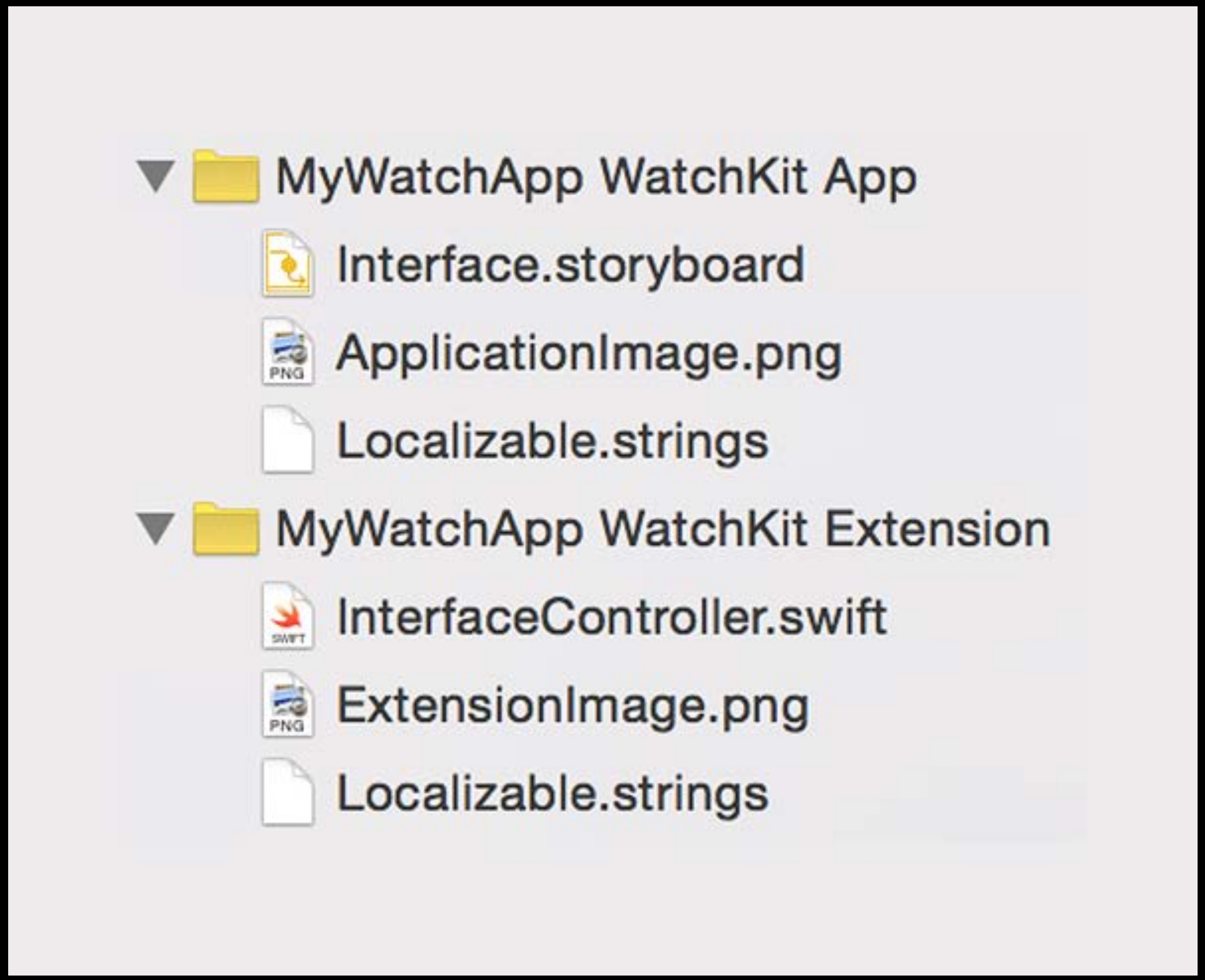
- Watch App bundle
- WatchKit Extension bundle

Resources

Static Resources

Multiple locations

- Watch App bundle
- WatchKit Extension bundle



MyWatchApp WatchKit App
Interface.storyboard
ApplicationImage.png
Localizable.strings
MyWatchApp WatchKit Extension
InterfaceController.swift
ExtensionImage.png
Localizable.strings

Resources

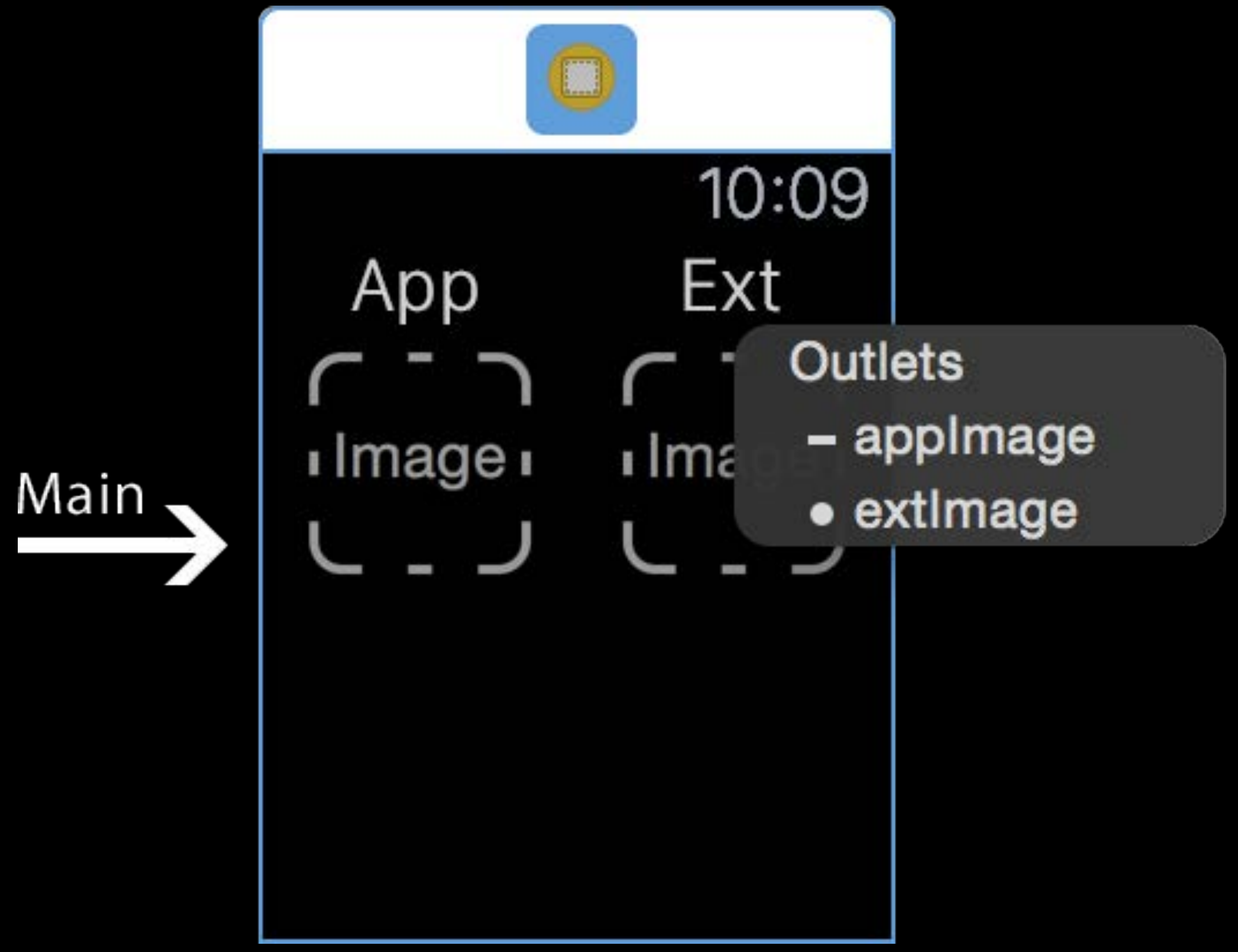
Static Resources

```
class InterfaceController: WKInterfaceController {
    @IBOutlet weak var appImage: WKInterfaceImage!
    @IBOutlet weak var extImage: WKInterfaceImage!
    override func willActivate() {
        super.willActivate()
        appImage.setImageNamed("ApplicationImage")
        extImage.setImageNamed("ExtensionImage")
    }
```

Resources

Static Resources

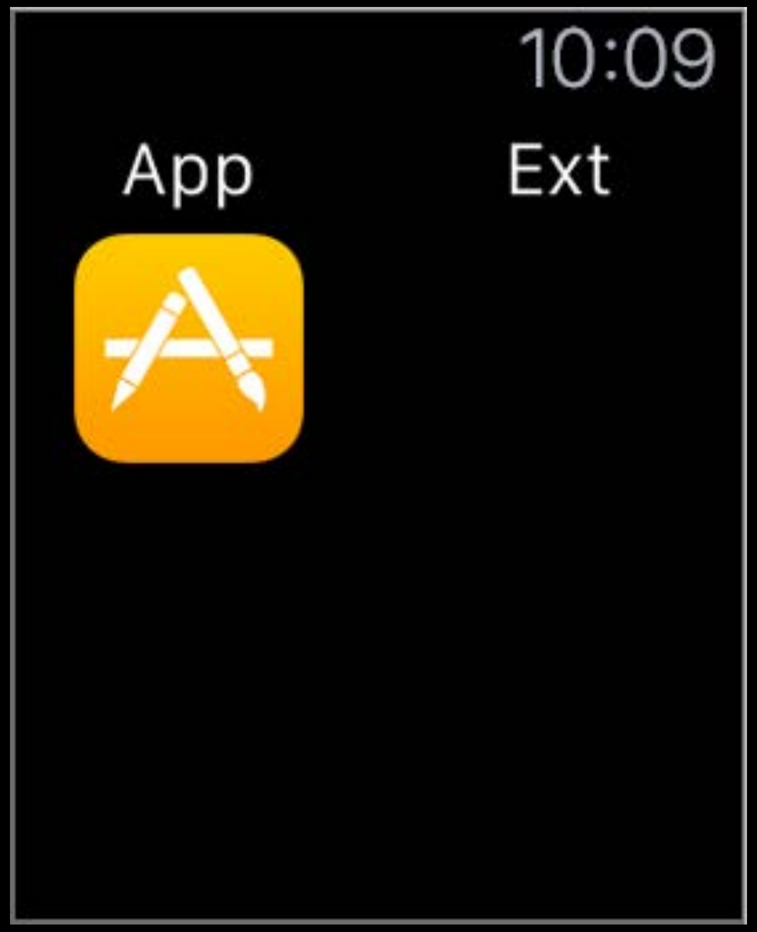
```
class InterfaceController: WKInterfaceController {
    @IBOutlet weak var appImage: WKInterfaceImage!
    @IBOutlet weak var extImage: WKInterfaceImage!
    override func willActivate() {
        super.willActivate()
        appImage.setImageNamed("ApplicationImage")
        extImage.setImageNamed("ExtensionImage")
    }
```



Resources

Static Resources

```
class InterfaceController: WKInterfaceController {
    @IBOutlet weak var appImage: WKInterfaceImage!
    @IBOutlet weak var extImage: WKInterfaceImage!
    override func willActivate() {
        super.willActivate()
        appImage.setImageNamed("ApplicationImage")
        extImage.setImageNamed("ExtensionImage")
    }
```



Resources

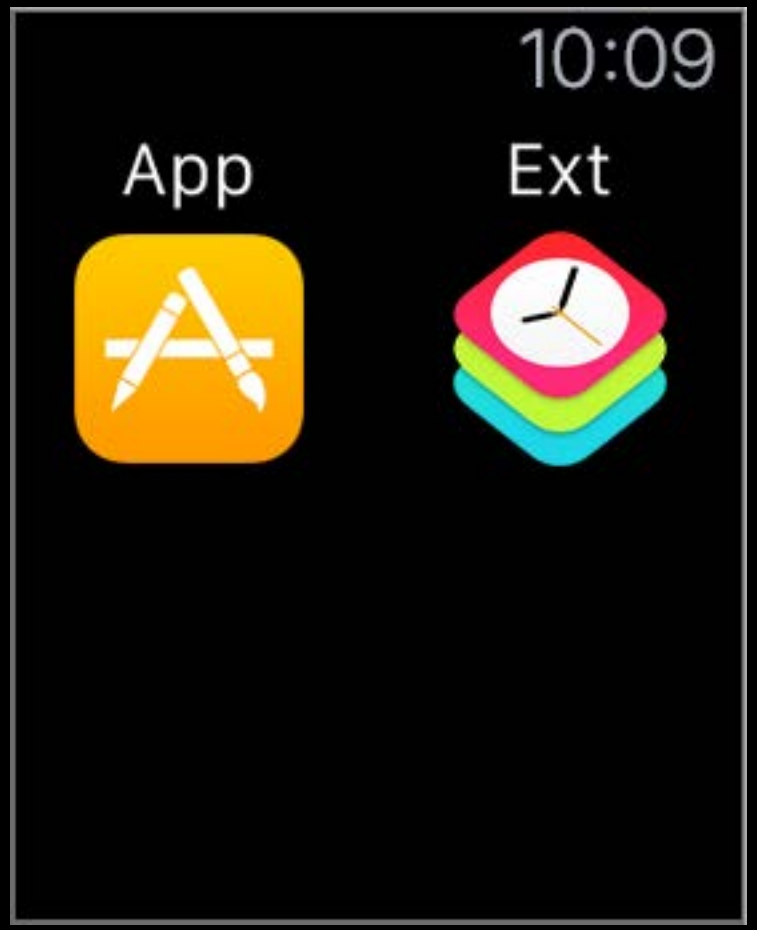
Static Resources

```
class InterfaceController: WKInterfaceController {
    …
        appImage.setImageNamed("ApplicationImage")
        let image = UIImage(named: "ExtensionImage")
        extImage.setImage(image)
    }
```

Resources

Static Resources

```
class InterfaceController: WKInterfaceController {
    …
        appImage.setImageNamed("ApplicationImage")
        let image = UIImage(named: "ExtensionImage")
        extImage.setImage(image)
    }
```



Data

Local data storage

Data

Local data storage

Document folder

- Non-purgeable
- Not restored

Data

Local data storage

Document folder

- Non-purgeable
- Not restored

Caches folder

- Purgeable

Data

Local data storage

```
func saveData(data: NSData, fileName: String) {
   let fileManager = NSFileManager.defaultManager()
   guard let documentDir = fileManager.URLsForDirectory(.DocumentDirectory,
                                inDomains: .UserDomainMask).first else { return }
   let url = documentDirectory.URLByAppendingPathComponent(fileName)
   data.writeToURL(url, atomically: true)
}
```

Data

Local data storage

```
func saveData(data: NSData, fileName: String) {
   let fileManager = NSFileManager.defaultManager()
   guard let documentDir = fileManager.URLsForDirectory(.DocumentDirectory,
                                     inDomains: .UserDomainMask).first else { return }
   let url = documentDirectory.URLByAppendingPathComponent(fileName)
   data.writeToURL(url, atomically: true)
}
```

Data

Local data storage

```
func saveData(data: NSData, fileName: String) {
   let fileManager = NSFileManager.defaultManager()
   guard let documentDir = fileManager.URLsForDirectory(.DocumentDirectory,
                                inDomains: .UserDomainMask).first else { return }
   let url = documentDirectory.URLByAppendingPathComponent(fileName)
   data.writeToURL(url, atomically: true)
}
```

Data

Media

Data

Media

Application

- Plays media files
- Records audio to a file

Data

Media

Application

- Plays media files
- Records audio to a file

Extension

- Downloads media files
- Reads recorded audio files

Data

Media

Data

Media

Must use a shared container

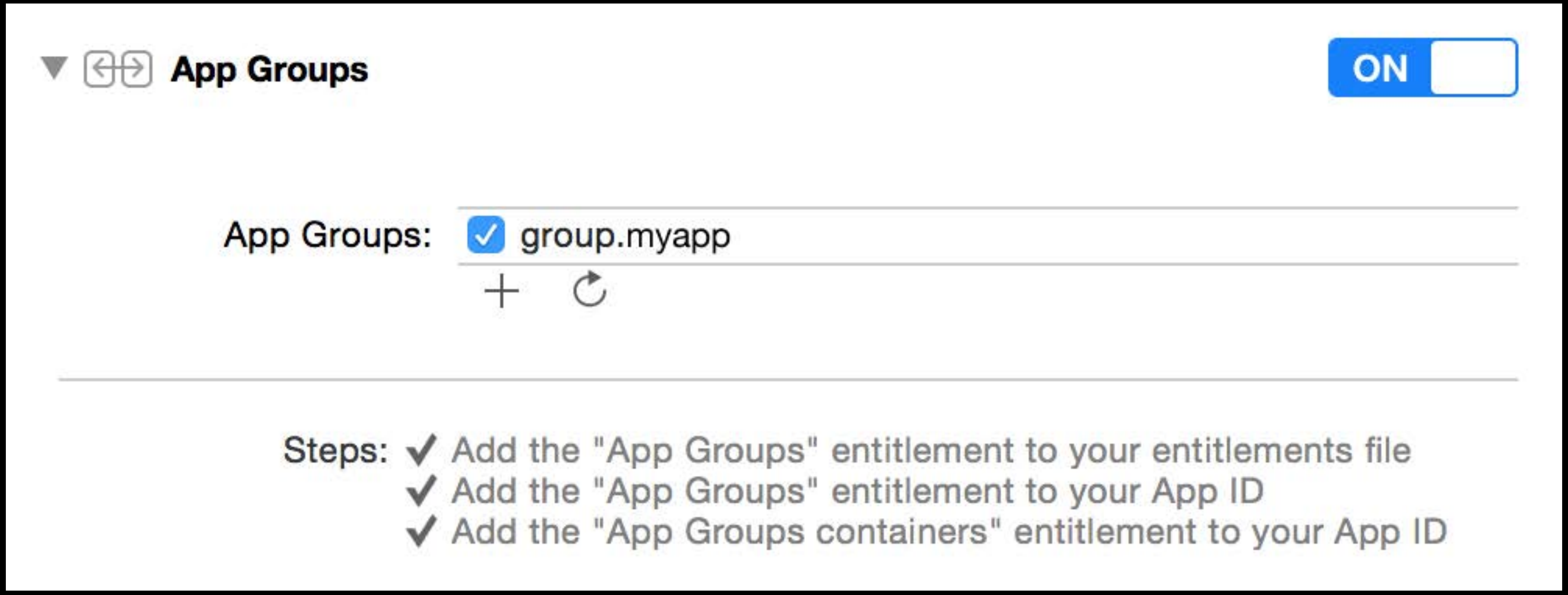
Data

Media

Must use a shared container

Enable App Groups in Xcode

- Extension
- Application



App Groups ON

App Groups: ✓ group.myapp

Steps: ✓ Add the "App Groups" entitlement to your entitlements file
✓ Add the "App Groups" entitlement to your App ID
✓ Add the "App Groups containers" entitlement to your App ID

Data

Media

```
func recordAudio(name: String) {

    let fileManager = NSFileManager.defaultManager()
    let container =
fileManager.containerURLForSecurityApplicationGroupIdentifier("group.myapp")!

    let fileName = name.stringByAppendingPathExtension("mp4")!
    let audioFileURL = container.URLByAppendingPathComponent(fileName)

    self.presentAudioRecordingControllerWithOutputURL(audioFileURL, ...
```

Data

Media

```
func recordAudio(name: String) {

    let fileManager = NSFileManager.defaultManager()
    let container =
fileManager.containerURLForSecurityApplicationGroupIdentifier("group.myapp")!

    let fileName = name.stringByAppendingPathExtension("mp4")!
    let audioFileURL = container.URLByAppendingPathComponent(fileName)

    self.presentAudioRecordingControllerWithOutputURL(audioFileURL, ...
```

Data

Transferring data

Data

Transferring data

NSURLSession

Data

Transferring data

NSURLSession

WatchConnectivity

Data

NSURLSession

Data

NSURLSession

Direct access to internet

- http:// and https://

Data

NSURLSession

Direct access to internet

- http:// and https://

Background uploads and downloads

- Extension may not be running
- Downloaded files must be copied

Data

NSURLSession

```
class Downloader : NSObject, NSURLSessionDownloadDelegate {

    lazy var session: NSURLSession = self.createDownloadSession()

    func createDownloadSession() -> NSURLSession {
        let config = NSURLSessionConfiguration
                    .backgroundSessionConfigurationWithIdentifier("Downloader")
        return NSURLSession(configuration: config, delegate: self,
                            delegateQueue: nil)
    }

    func download(url: NSURL) {
        let task = session.downloadTaskWithURL(url)!
        task.resume()
    }
```

Data

NSURLSession

```
class Downloader : NSObject, NSURLSessionDownloadDelegate {

    lazy var session: NSURLSession = self.createDownloadSession()

    func createDownloadSession() -> NSURLSession {
        let config = NSURLSessionConfiguration
                  .backgroundSessionConfigurationWithIdentifier("Downloader")
        return NSURLSession(configuration: config, delegate: self,
                            delegateQueue: nil)
    }

    func download(url: NSURL) {
        let task = session.downloadTaskWithURL(url)!
        task.resume()
    }
```

Data

NSURLSession

```
class Downloader : NSObject, NSURLSessionDownloadDelegate {

    lazy var session: NSURLSession = self.createDownloadSession()

    func createDownloadSession() -> NSURLSession {
        let config = NSURLSessionConfiguration
                  .backgroundSessionConfigurationWithIdentifier("Downloader")
        return NSURLSession(configuration: config, delegate: self,
                            delegateQueue: nil)
    }

    func download(url: NSURL) {
        let task = session.downloadTaskWithURL(url)!
        task.resume()
    }
```

Data

NSURLSession

```
class Downloader : NSObject, NSURLSessionDownloadDelegate {

    lazy var session: NSURLSession = self.createDownloadSession()

    func createDownloadSession() -> NSURLSession {
        let config = NSURLSessionConfiguration
                   .backgroundSessionConfigurationWithIdentifier("Downloader")
        return NSURLSession(configuration: config, delegate: self,
                                      delegateQueue: nil)
    }

    func download(url: NSURL) {
        let task = session.downloadTaskWithURL(url)!
        task.resume()
    }
```

Data

NSURLSession

```
func restart() {
     session = createDownloadSession()
}

func URLSession(NSURLSession, downloadTask: NSURLSessionDownloadTask,
                              didFinishDownloadingToURL location: NSURL) {
    let fileManager = NSFileManager.defaultManager()
    guard let cacheDir = fileManager.URLsForDirectory(.CachesDirectory,
                             inDomains: .UserDomainMask).first else { return }
    let cacheURL =
        cacheDir.URLByAppendingPathComponent(location.lastPathComponent!)

    do { try fileManager.copyItemAtURL(location, toURL: cacheURL) }
    catch { print(error) }
}
```

Data

NSURLSession

```
func restart() {
    session = createDownloadSession()
}

func URLSession(NSURLSession, downloadTask: NSURLSessionDownloadTask,
                              didFinishDownloadingToURL location: NSURL) {
    let fileManager = NSFileManager.defaultManager()
    guard let cacheDir = fileManager.URLsForDirectory(.CachesDirectory,
                            inDomains: .UserDomainMask).first else { return }
    let cacheURL =
        cacheDir.URLByAppendingPathComponent(location.lastPathComponent!)

    do { try fileManager.copyItemAtURL(location, toURL: cacheURL) }
    catch { print(error) }
}
```

Data

NSURLSession

```
func restart() {
     session = createDownloadSession()
}

func URLSession(NSURLSession, downloadTask: NSURLSessionDownloadTask,
                                didFinishDownloadingToURL location: NSURL) {
    let fileManager = NSFileManager.defaultManager()
    guard let cacheDir = fileManager.URLsForDirectory(.CachesDirectory,
                                inDomains: .UserDomainMask).first else { return }
    let cacheURL =
        cacheDir.URLByAppendingPathComponent(location.lastPathComponent!)

    do { try fileManager.copyItemAtURL(location, toURL: cacheURL) }
    catch { print(error) }
}
```

Data

NSURLSession

```
func restart() {
     session = createDownloadSession()
}

func URLSession(NSURLSession, downloadTask: NSURLSessionDownloadTask,
                              didFinishDownloadingToURL location: NSURL) {
    let fileManager = NSFileManager.defaultManager()
    guard let cacheDir = fileManager.URLsForDirectory(.CachesDirectory,
                             inDomains: .UserDomainMask).first else { return }
    let cacheURL =
        cacheDir.URLByAppendingPathComponent(location.lastPathComponent!)

    do { try fileManager.copyItemAtURL(location, toURL: cacheURL) }
    catch { print(error) }
}
```

Data

WatchConnectivity framework

NEW

Watch ↔ Phone

- Share data
- Transfer files
- Talk to counterpart

Data

WatchConnectivity framework

NEW

Watch ↔ Phone

- Share data
- Transfer files
- Talk to counterpart

Migration

watchOS 1 ➟ watchOS 2

Migration

watchOS 1

Migration

watchOS 1

WatchKit extension for watchOS 1

Migration

watchOS 1

WatchKit extension for watchOS 1

- Uses iOS Platform and SDK

Migration

watchOS 1

WatchKit extension for watchOS 1

- Uses iOS Platform and SDK
- Runs on iPhone

Migration
watchOS 1

WatchKit extension for watchOS 1

- Uses iOS Platform and SDK
- Runs on iPhone
- Share framework with iOS application

Migration

watchOS 1

WatchKit extension for watchOS 1

- Uses iOS Platform and SDK
- Runs on iPhone
- Share framework with iOS application
- Image caching

Migration

watchOS 1

WatchKit extension for watchOS 1

- Uses iOS Platform and SDK
- Runs on iPhone
- Share framework with iOS application
- Image caching
- openParentApplication()

Migration

watchOS 2

Migration

watchOS 2

New watchOS platform and SDK

Migration

watchOS 2

New watchOS platform and SDK

Subset of iOS frameworks available

Migration

watchOS 2

New watchOS platform and SDK

Subset of iOS frameworks available

Include project frameworks

Migration

Reusability

Migration

Reusability

Same API

- Copy code
- Copy resources

Migration

Reusability

Same API

- Copy code
- Copy resources

Improvements

- UI responsiveness
- Independent operation
- New UI elements
- Animation

Migration

Reusability

Same API

- Copy code
- Copy resources

Improvements

- UI responsiveness
- Independent operation
- New UI elements
- Animation

Migration

Controllers

Migration
Controllers

Same

- Interface controller
- Glance controller
- Notification controller

Migration

Controllers

NEW

Same

- Interface controller
- Glance controller
- Notification controller

New

- Extension delegate
- Complication data source

Migration

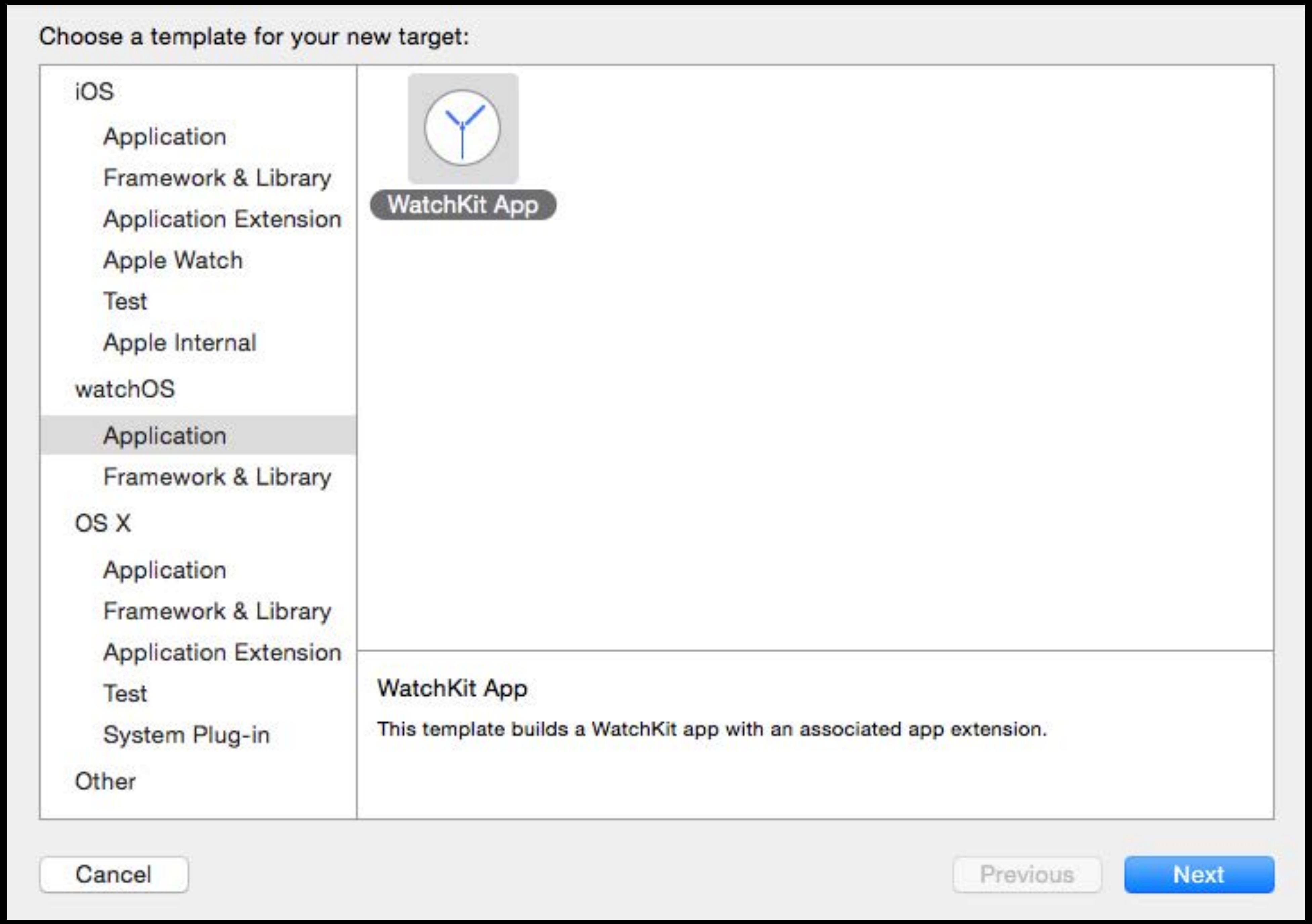
Xcode

Migration

Xcode

Existing project

- Add watchOS Application target


Choose a template for your new target:

iOS
- Application
- Framework & Library
- Application Extension
- Apple Watch
- Test
- Apple Internal

watchOS
- Application
- Framework & Library

OS X
- Application
- Framework & Library
- Application Extension
- Test
- System Plug-in

Other

WatchKit App

WatchKit App

This template builds a WatchKit app with an associated app extension.

Migration

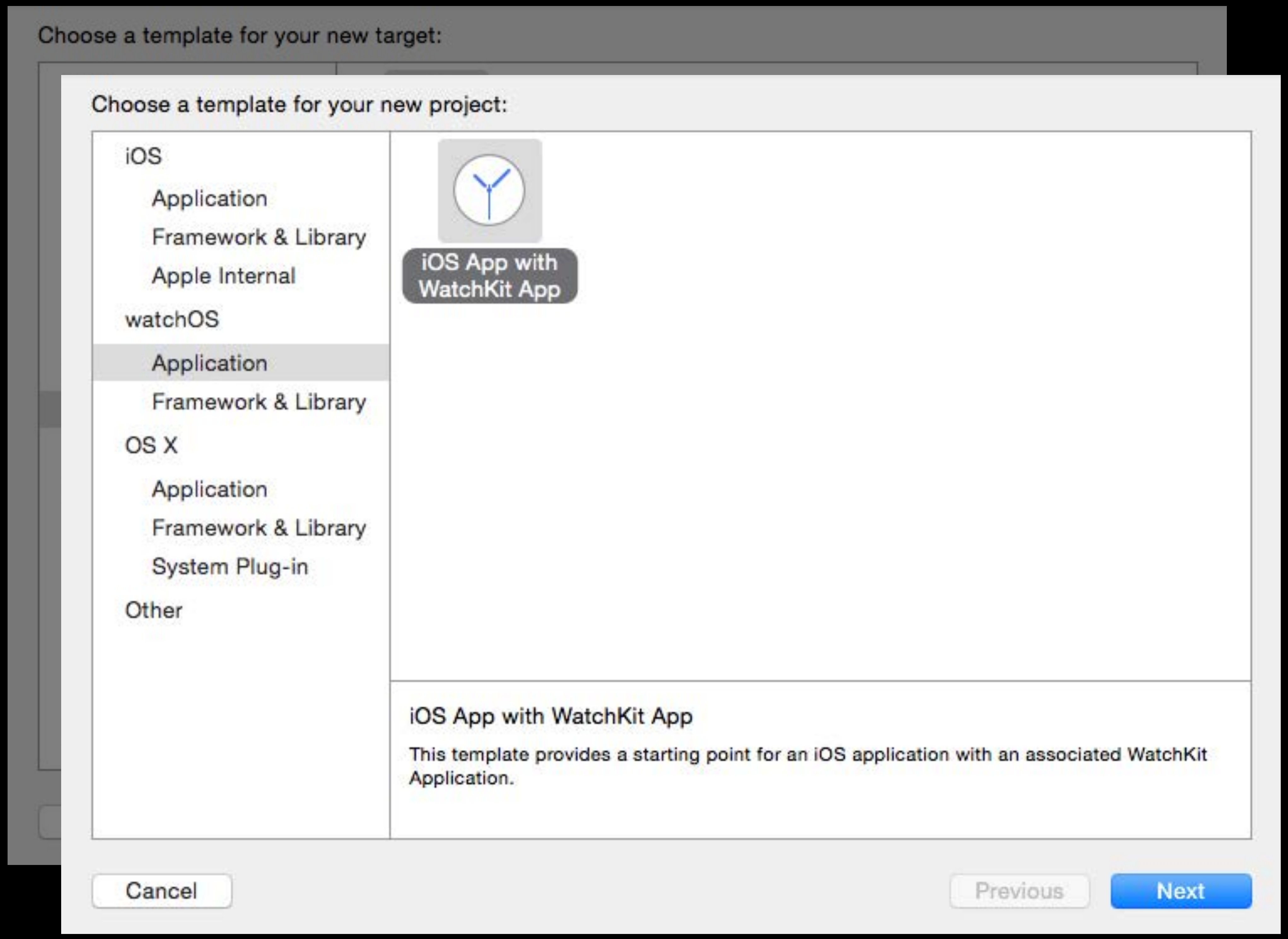
Xcode

Existing project

- Add watchOS Application target

New project

- Create iOS App with WatchKit App



Choose a template for your new target:

Choose a template for your new project:

iOS App with WatchKit App

This template provides a starting point for an iOS application with an associated WatchKit Application.

Migration

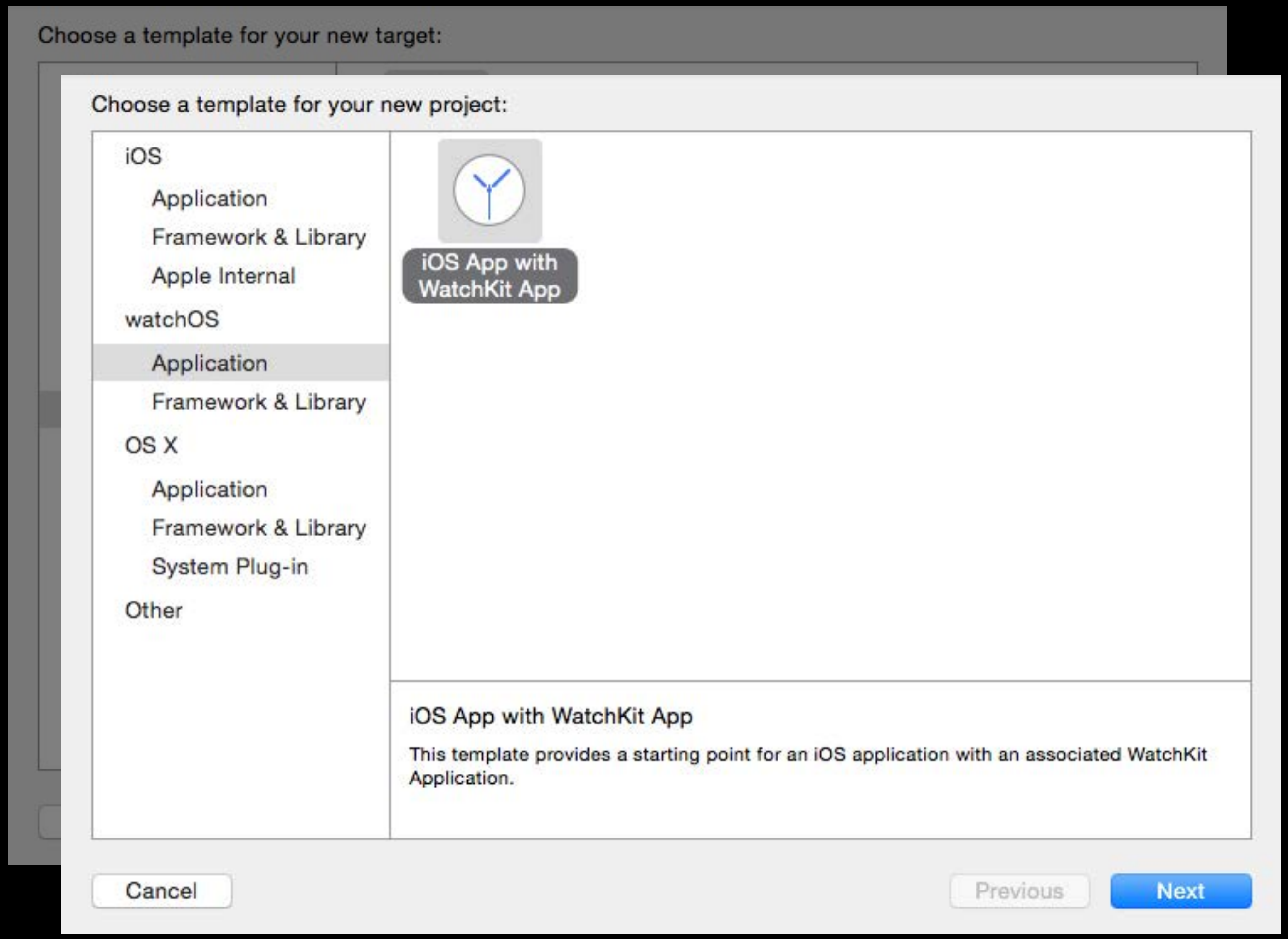
Xcode

Existing project

- Add watchOS Application target

New project

- Create iOS App with WatchKit App



Choose a template for your new target:

Choose a template for your new project:

iOS App with WatchKit App

This template provides a starting point for an iOS application with an associated WatchKit Application.

New APIs in WatchKit for watchOS 2

Forest Hill WatchKit Engineer

WKExtensionDelegate

Callback methods for app lifecycle

WKExtensionDelegate

Callback methods for app lifecycle

- iOS
    - UIApplicationDelegate

WKExtensionDelegate

Callback methods for app lifecycle

iOS

- UIApplicationDelegate

watchOS 2

- WKExtensionDelegate

WKExtensionDelegate

applicationDidFinishLaunching

Called once on launch

Perform app initialization

Setup notification observers

Warmup services

NOTE: App is not active yet

WKExtensionDelegate

applicationDidBecomeActive

Each time app becomes visually active

Activate timers

Update any state

WKExtensionDelegate

applicationWillResignActive

Only call before going to background

Prepare to be inactive

Save state

Disable running services, timers, etc

WKExtensionDelegate

App lifecycle

```
func applicationDidFinishLaunching()
func applicationDidBecomeActive()
func applicationWillResignActive()
```

Callbacks on app lifecycle only

WKExtensionDelegate

User Activity

```
func handleUserActivity(userInfo: [NSObject : AnyObject])
```

watchOS 1

    method on root WKInterfaceController

watchOS 2

    method on WKExtensionDelegate

User Activity

NEW

User Activity

NEW

*This property on WKExtension:

```
var rootInterfaceController: WKInterfaceController
```

will be coming in a future seed of watchOS 2

User Activity

NEW

*This property on WKExtension:

```
var rootInterfaceController: WKInterfaceController
```

will be coming in a future seed of watchOS 2

```
func handleUserActivity(userInfo: [NSObject : AnyObject]) {
    let rootController = WKExtension.sharedExtension.rootInterfaceController
    rootController.popToRootController()
    rootController.doStuffForUserActivity(userInfo)
}
```

WKExtension

Analogous to UIApplication in iOS

Singleton

Encapsulates the running application

WKExtension

Analogous to UIApplication in iOS

Singleton

Encapsulates the running application.

In watchOS 2, we have WKExtension

WKExtension

Open URLs

```
WKExtension.sharedExtension().openSystemURL(systemURL)
```

Valid Schemes

- Phone
- SMS
- PassKit

Notifications

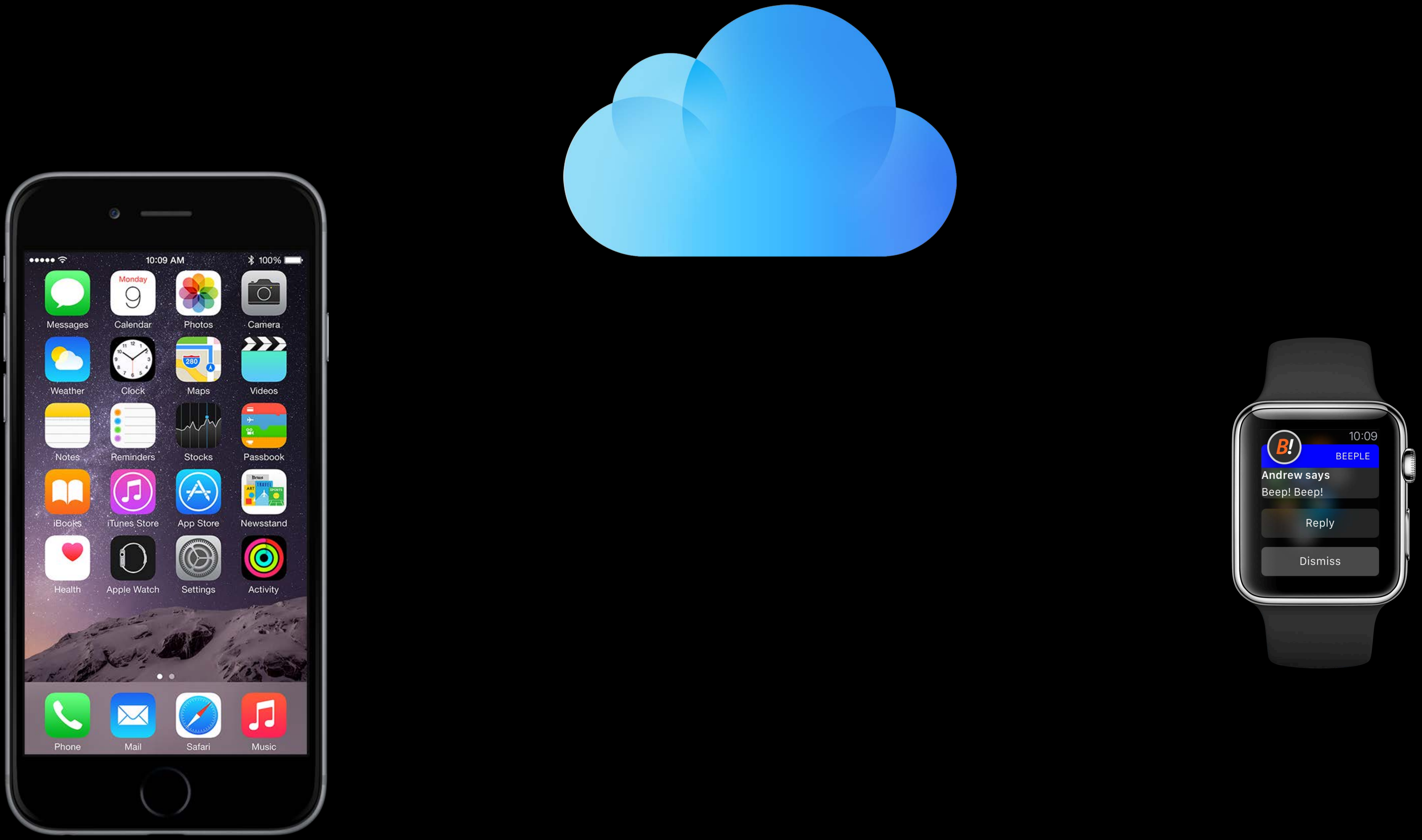
Remote Notifications

10:09 AM
100%
Monday
9
Messages
Calendar
Photos
Camera
Weather
Clock
Maps
Videos
Notes
Reminders
Stocks
Passbook
iBooks
iTunes Store
App Store
Newsstand
Health
Apple Watch
Settings
Activity
Phone
Mail
Safari
Music

Remote Notifications

Remote Notification

Remote Notifications



Notification Routing Rules

Criteria include

Notification Routing Rules

Criteria include

iPhone screen is locked

Notification Routing Rules

Criteria include

iPhone screen is locked

 Watch

- on wrist
- unlocked

Notifications

Notification handling when app is not active

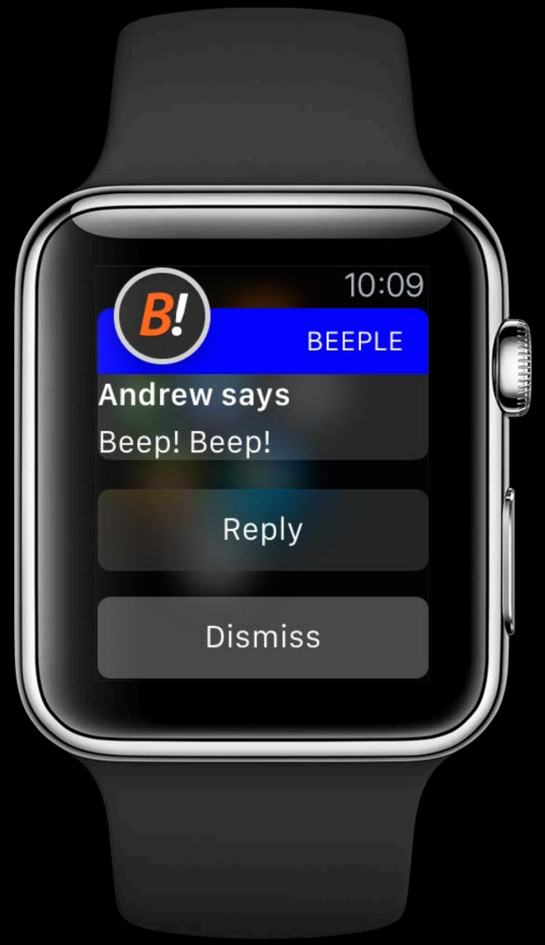
Called on WKUserNotificationInterfaceController

```
func didReceiveRemoteNotification(remoteNotification: [NSObject : AnyObject],
                         withCompletion: WKUserNotificationInterfaceType -> Void)
func didReceiveLocalNotification(localNotification: UILocalNotification,
                         withCompletion: WKUserNotificationInterfaceType -> Void)
```

Local Notifications

Local Notifications

Local Notifications



Scheduling Local Notification

Sending message to phone

Scheduling Local Notification

Sending message to phone

```
let message = ["request" : "fireLocalNotification"]
WCSession.defaultSession().sendMessage(message,
    replyHandler: nil,
    errorHandler: { error in
       print(error.localizedDescription)
    }
)
```

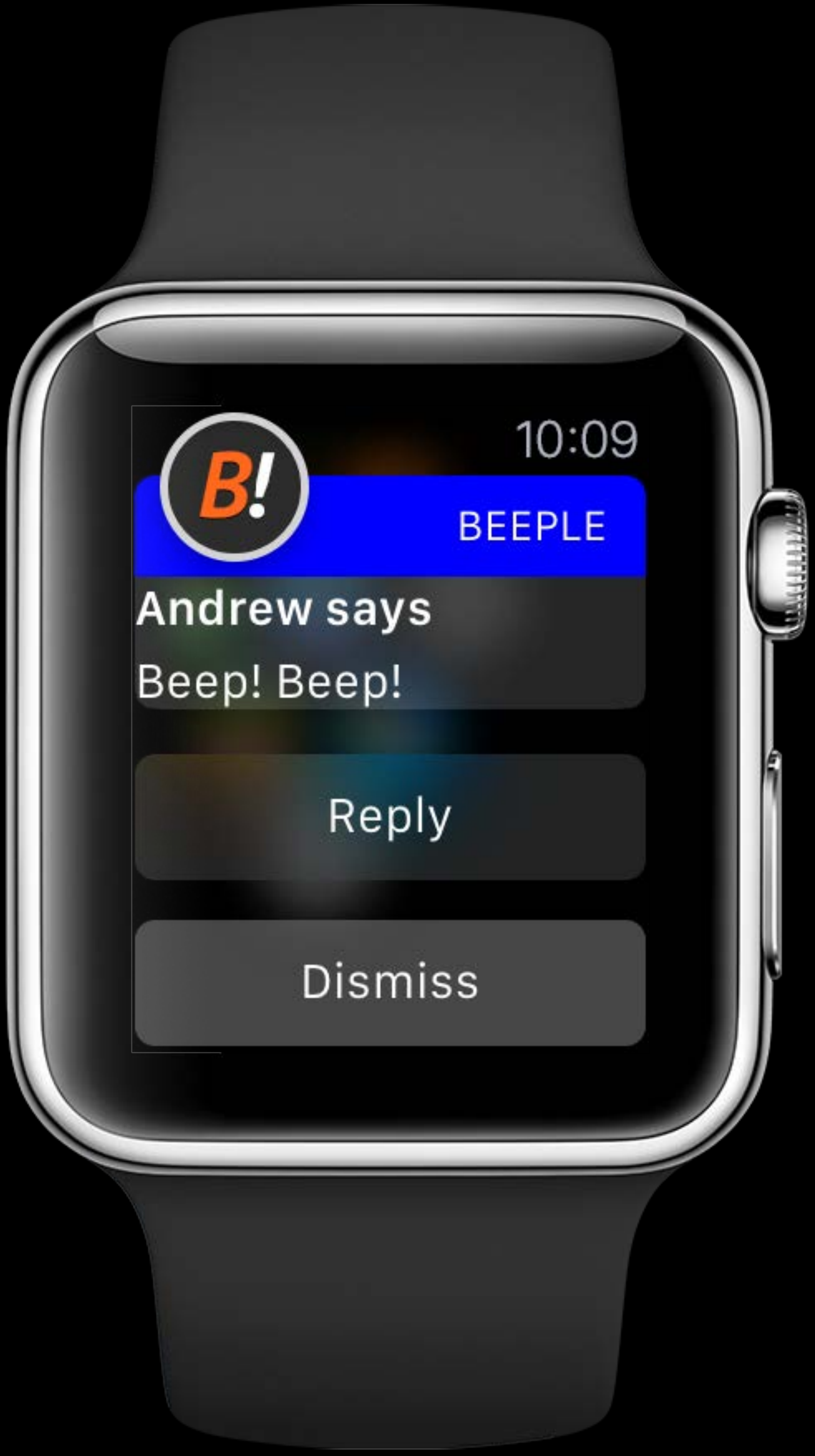
Scheduling Local Notification

Sending message to phone

```
let message = ["request" : "fireLocalNotification"]
WCSession.defaultSession().sendMessage(message,
    replyHandler: nil,
    errorHandler: { error in
       print(error.localizedDescription)
    }
)
```

```
func session(session: WCSession, didReceiveMessage message:
[String : AnyObject]) {
    guard message["request"] as? String == “fireLocalNotification” else {
       return
    }

    let localNotification = buildLocalNotification()
    UIApplication.sharedApplication().scheduleLocalNotification(
      localNotification)
}
```

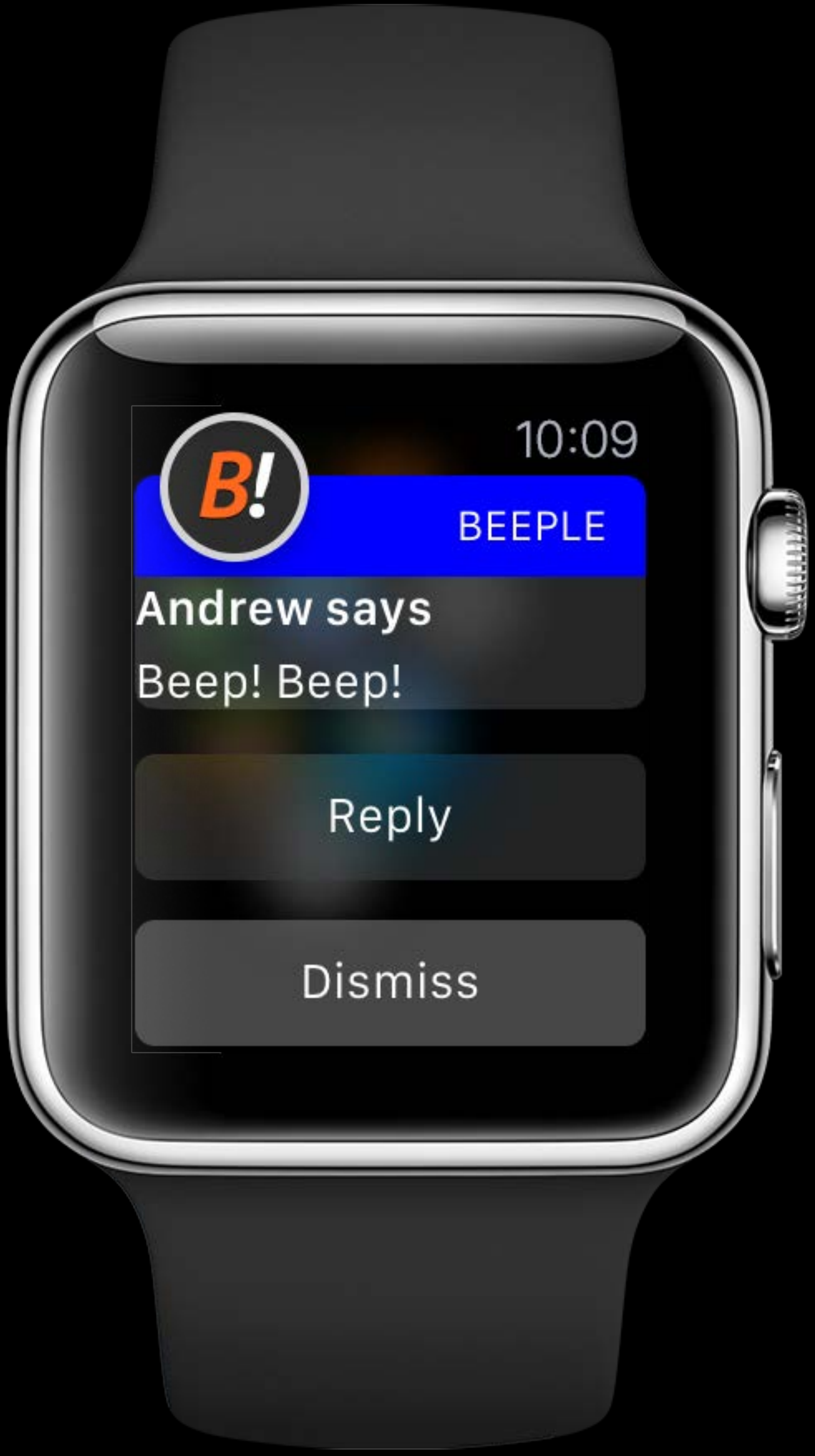

10:09
B!
BEEPLE
Andrew says
Beep! Beep!
Reply
Dismiss

Notification Actions

Launching from a notification action

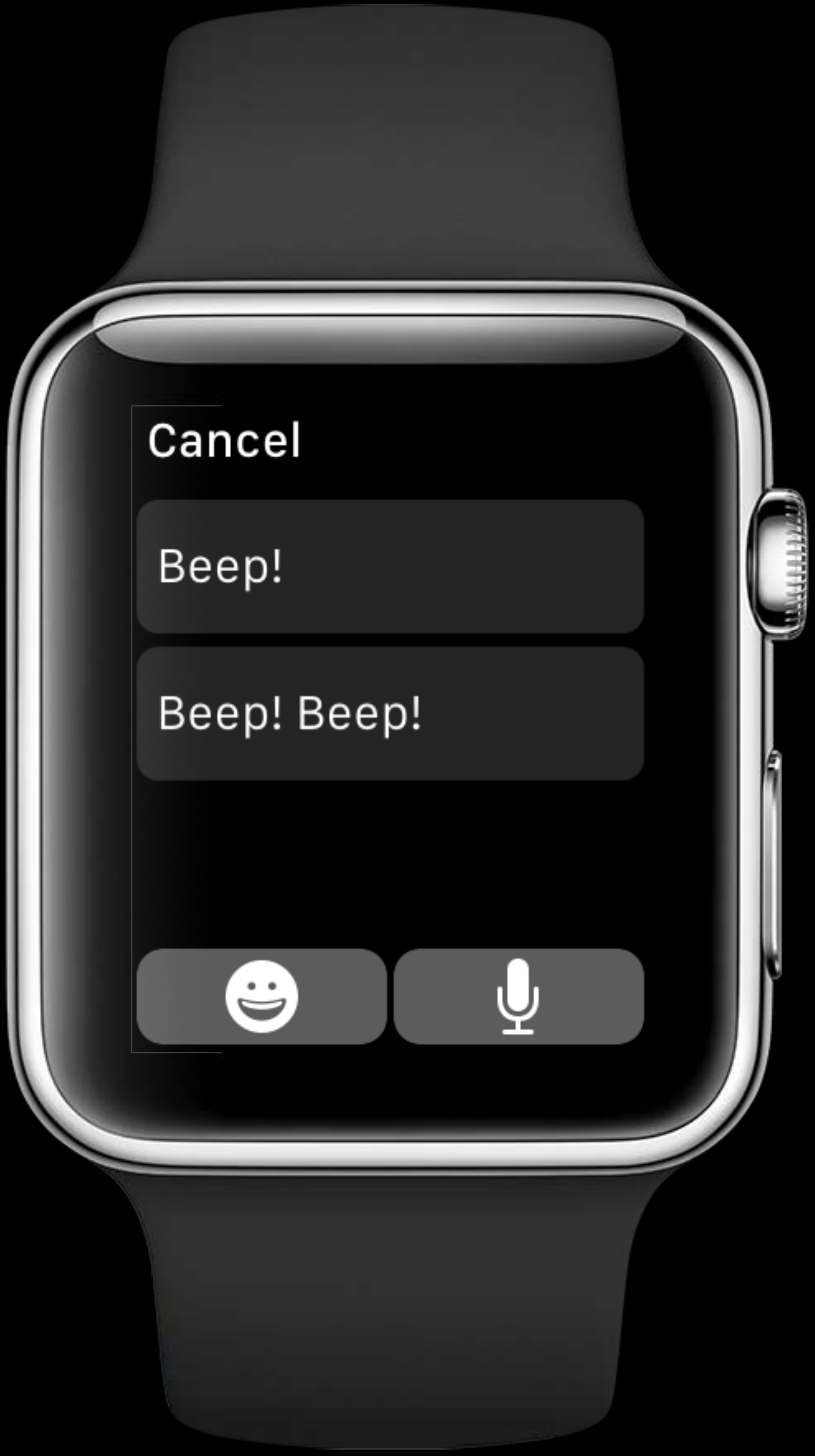
```
func handleActionWithIdentifier(identifier: String,
                                forRemoteNotification: [NSObject : AnyObject])
func handleActionWithIdentifier(identifier: String,
                                forLocalNotification: UILocalNotification)
```

Inline notification text replies



Andrew says

Beep! Beep!


Cancel

Beep!

Beep! Beep!

Notification Actions

Providing suggestions for inline text replies

```
func suggestionsForResponseToActionWithIdentifier(identifier: String,
remoteNotification: [String : AnyObject]) -> [String] {
    return ["Beep!", "Beep! Beep!"]
}
```

Notification Actions

Launching from a notification action with inline text input

```
func handleActionWithIdentifier(identifier: String,
                                forRemoteNotification: [NSObject : AnyObject],
                                withResponseInfo: [NSObject : AnyObject])

func handleActionWithIdentifier(identifier: String,
                                forLocalNotification: UILocalNotification,
                                withResponseInfo: [NSObject : AnyObject])
```

Notification Actions

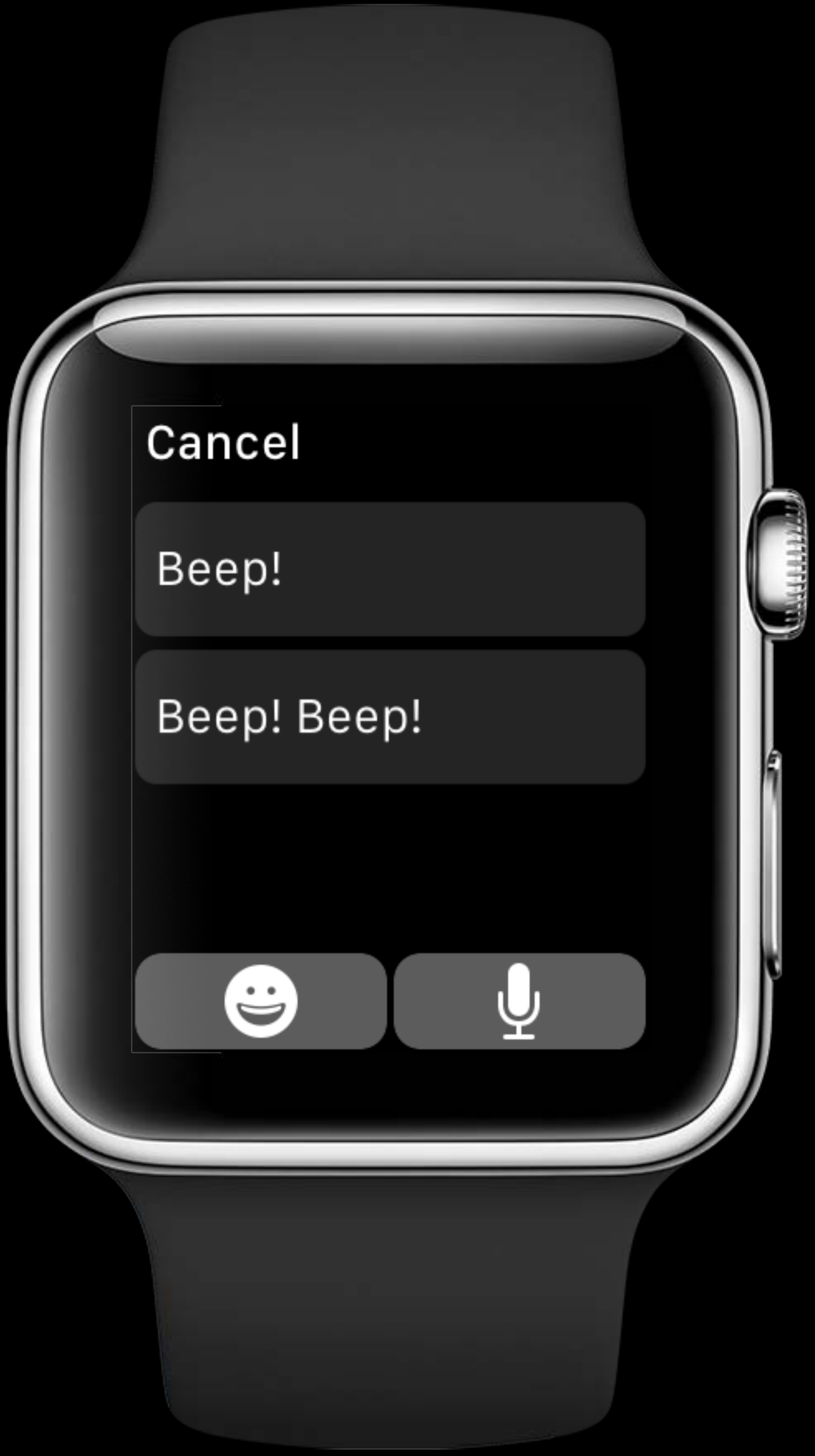
Launching from a notification action with inline text input

```
func handleActionWithIdentifier(identifier: String,
                        forRemoteNotification: [NSObject : AnyObject],
                        withResponseInfo: [NSObject : AnyObject])

func handleActionWithIdentifier(identifier: String,
                        forLocalNotification: UILocalNotification,
                        withResponseInfo: [NSObject : AnyObject])
```

Inline text stored in

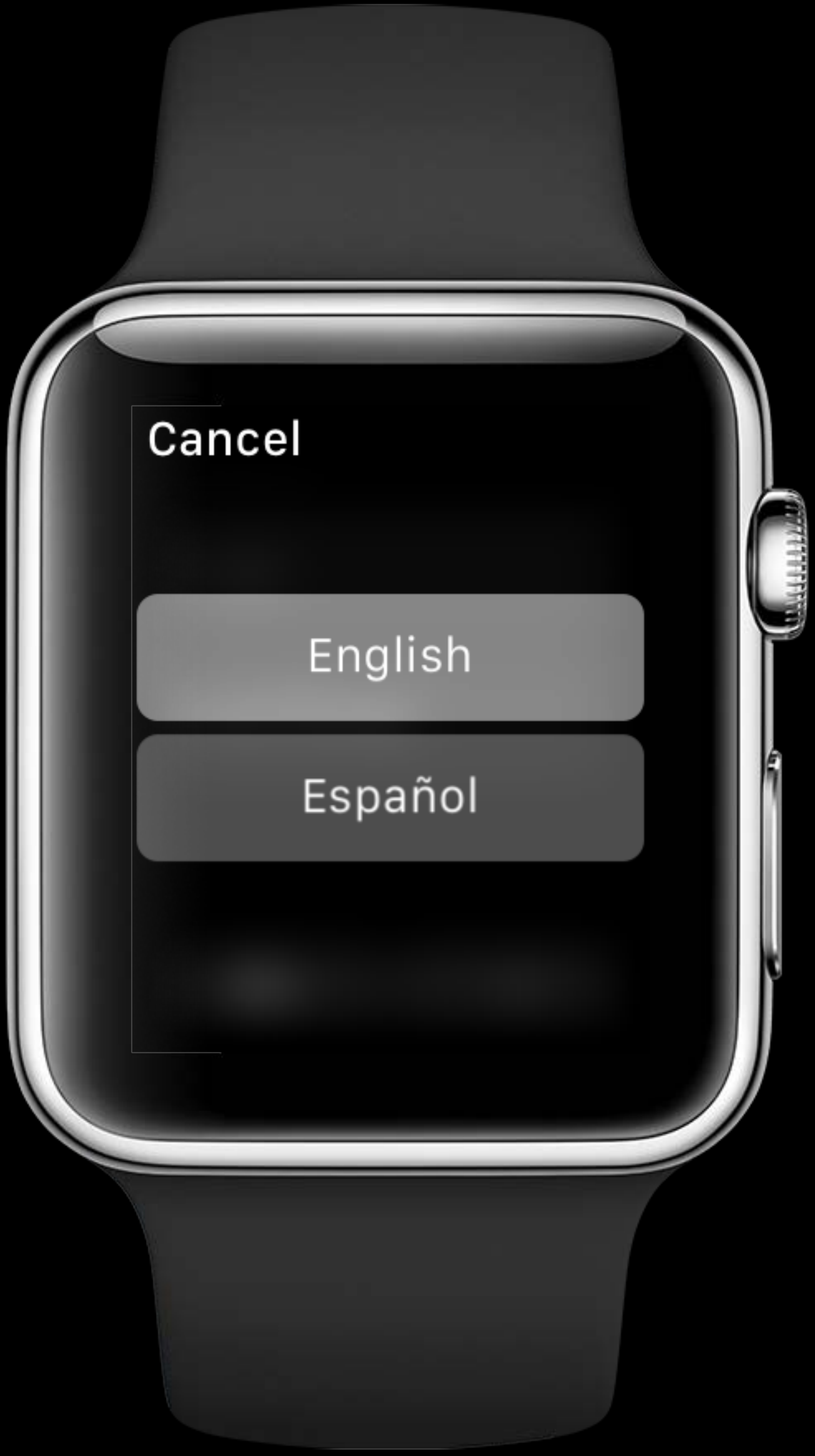
`UIUserNotificationActionResponseTypedTextKey`

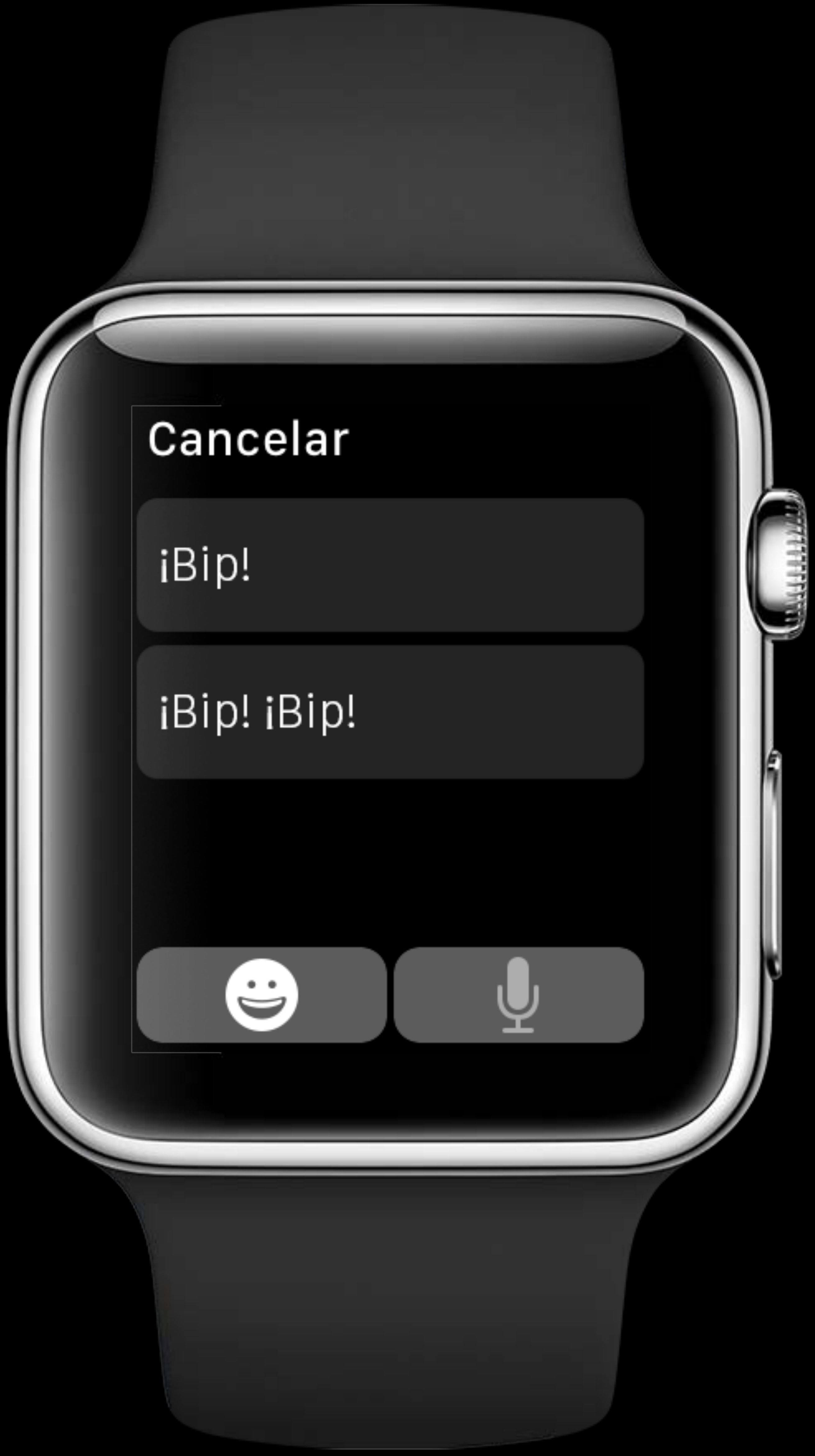

Cancel

Beep!

Beep! Beep!

Choose Language


Cancel
English
Español



Cancelar

¡Bip!

¡Bip! ¡Bip!

Notifications

Notification handling when app is active

On your WKExtensionDelegate:

```
func didReceiveRemoteNotification(userInfo: [NSObject : AnyObject])
func didReceiveLocalNotification(notification: UILocalNotification)
```

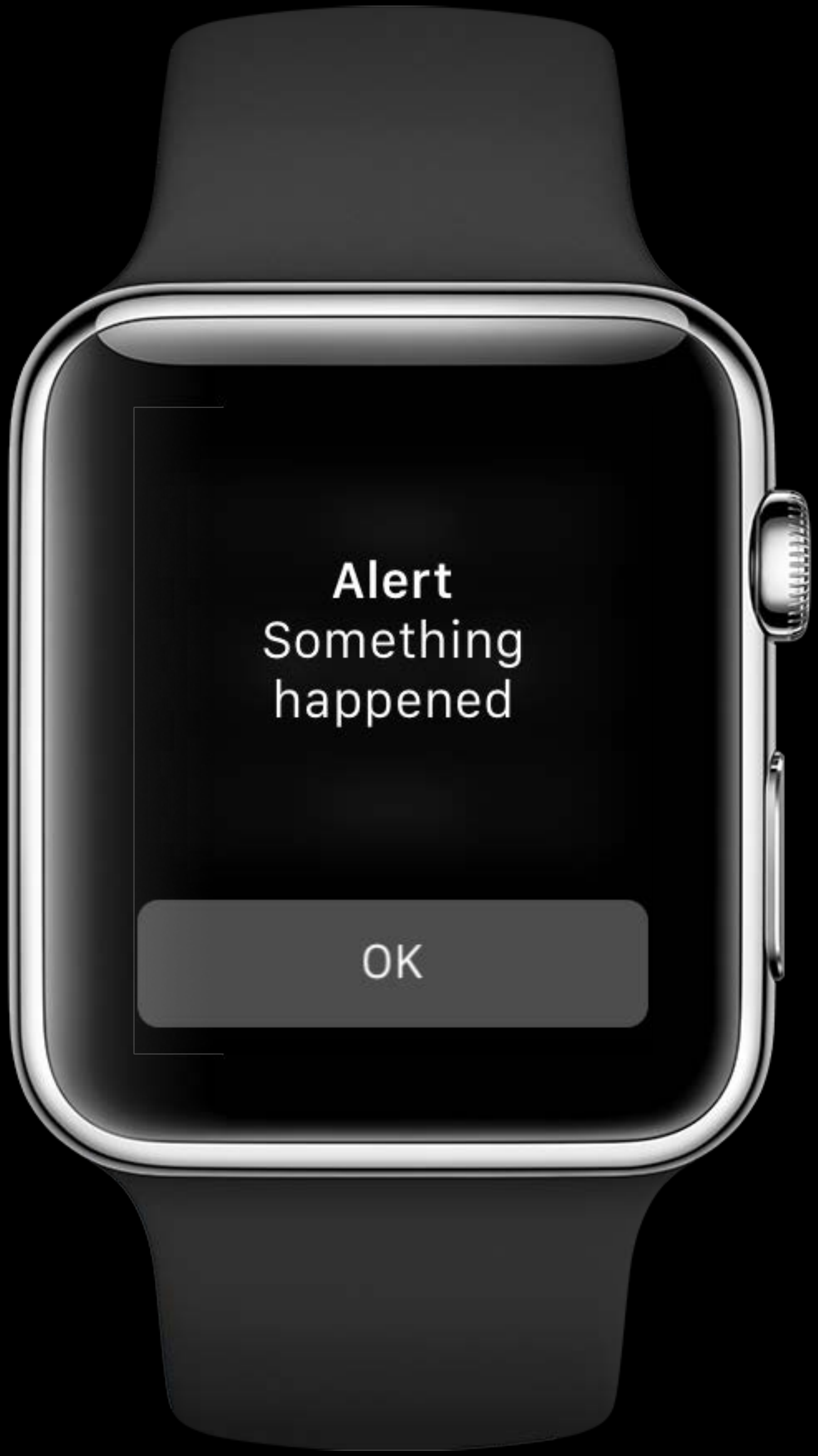
Modal Alerts

Modal Alerts

```
func presentAlertControllerWithTitle(title: String?,
                                     message: String?,
                                     preferredStyle: WKAlertControllerStyle,
                                     actions: [WKAlertAction])
```

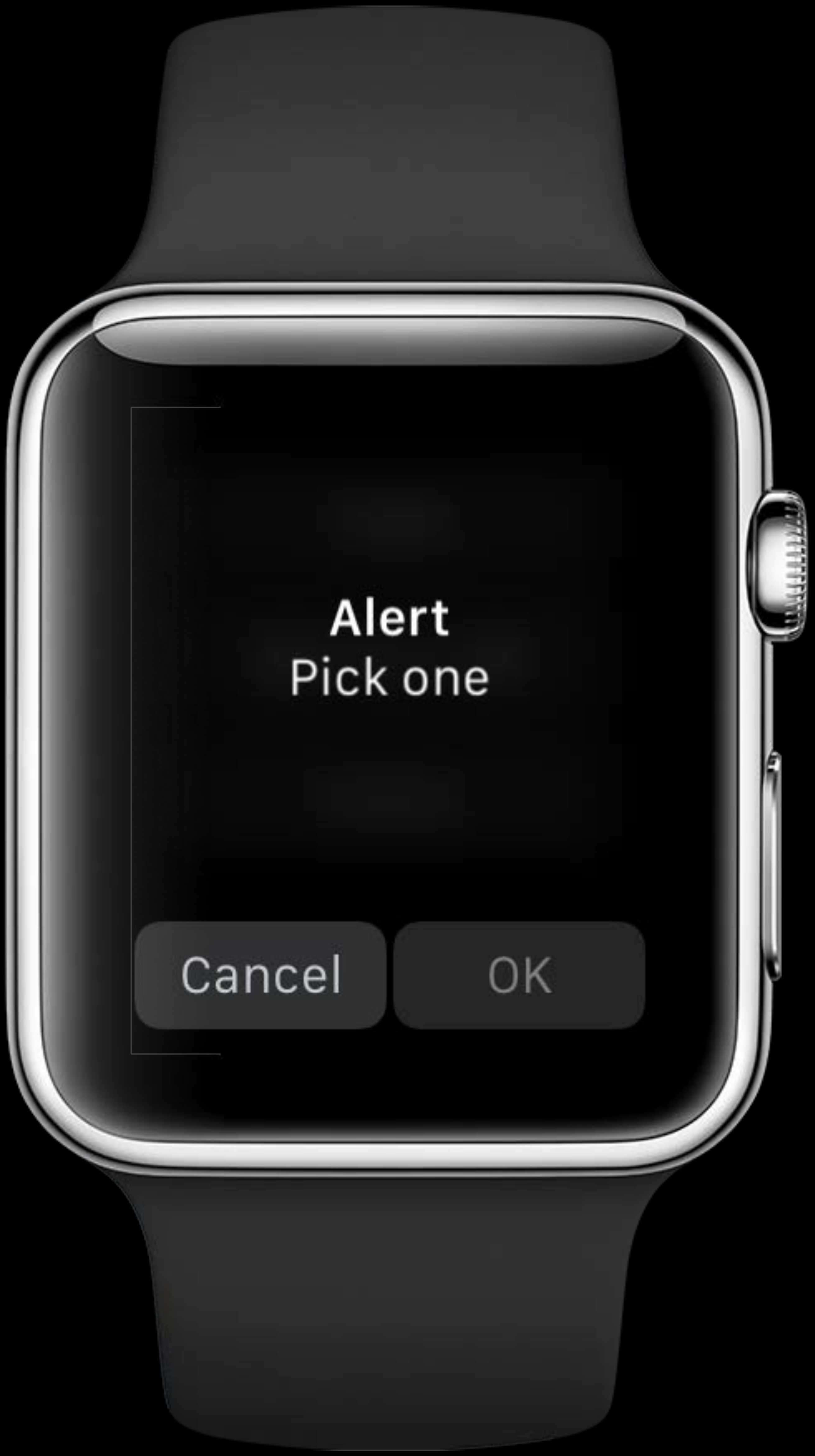
Modal Alert

.Alert



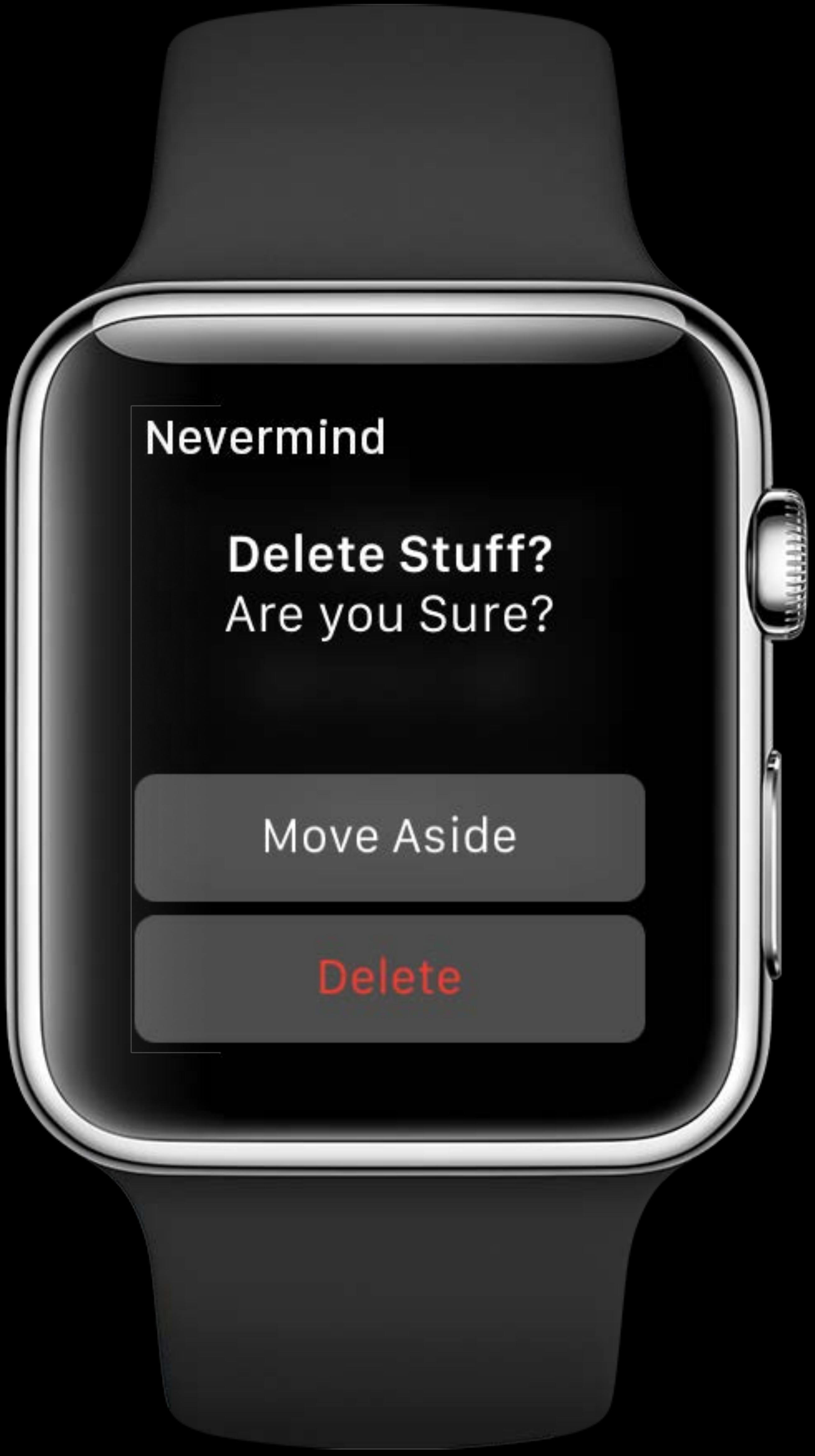
Modal Alerts

.SideBySideButtonsAlert



Modal Alerts

.ActionSheet



Summary

Summary

New architecture in watchOS 2

Summary

New architecture in watchOS 2

WKExtensionDelegate

Summary

New architecture in watchOS 2

WKExtensionDelegate

New API

Summary

New architecture in watchOS 2

WKExtensionDelegate

New API

More to come…

More Information

Documentation

watchOS 2 Transition Guide
WatchKit Programming
developer.apple.com/library

Sample Code

Lister
WatchKit Catalogue
developer.apple.com/watchOS

Technical Support

Apple Developer Forums
Developer Technical Support

General Inquiries

Jake Behrens, watchOS
Frameworks Evangelist
behrens@apple.com

Related Sessions

Introducing WatchKit for watchOS 2	Presidio	Tuesday 10:00AM
Building Watch Apps	Pacific Heights	Tuesday 4:30PM
WatchKit In-Depth, Part 2	Pacific Heights	Wednesday 10:00AM
Creating Complications with ClockKit	Pacific Heights	Wednesday 11:00AM
WatchKit Tips and Tricks	Presidio	Friday 10:00AM

WWDC15