The Life of a watchOS App

Session 216

Neil Desai, WatchKit Framework Engineer


Wed
7

Unified process runtime

Frontmost app state

Background app refresh

Background modes

Unified process runtime

Frontmost app state

Background app refresh

Background modes

Unified process runtime

Frontmost app state

Background app refresh

Background modes

Unified process runtime

Frontmost app state

Background app refresh

Background modes

Unified process runtime

Frontmost app state

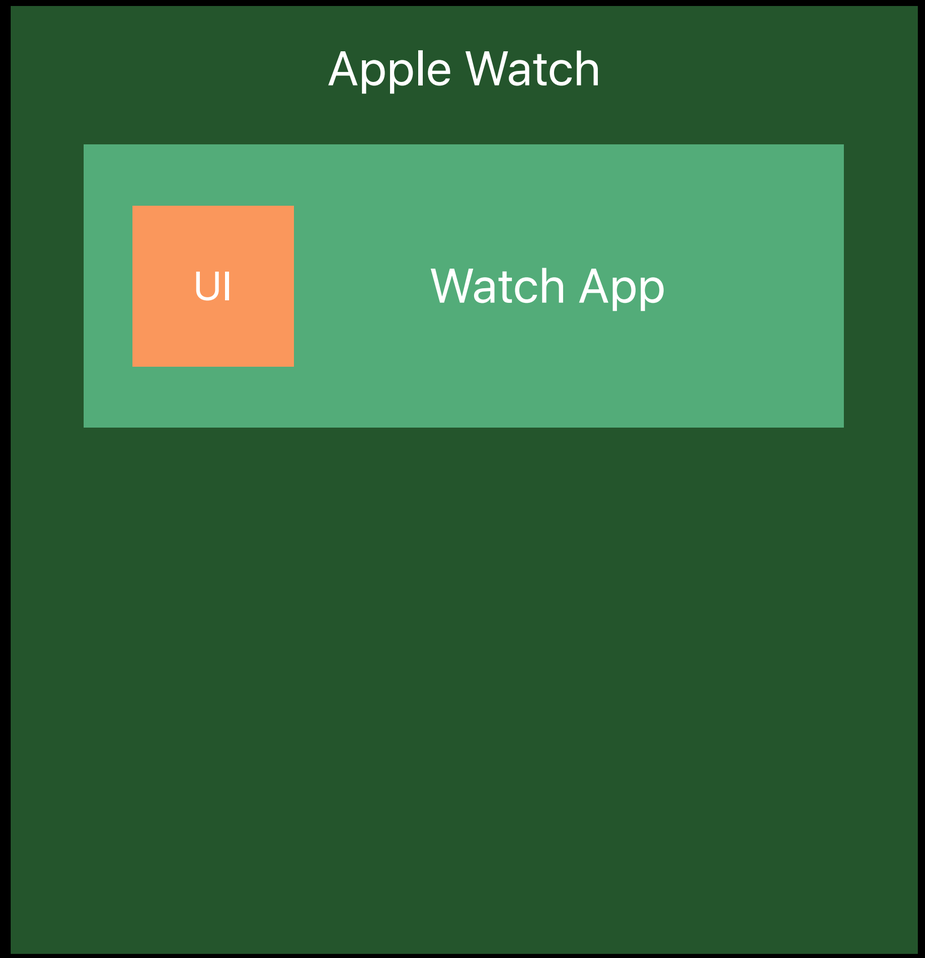
Background app refresh

Background modes

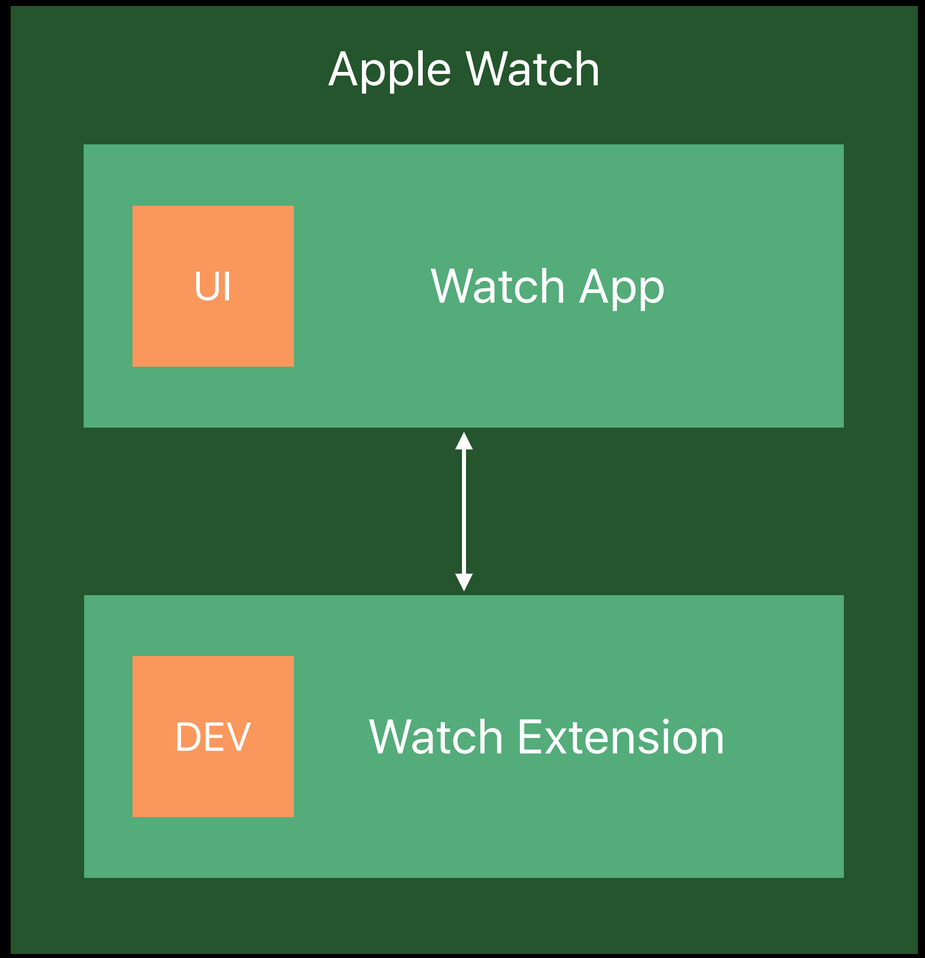
Unified Process Runtime

Architecture

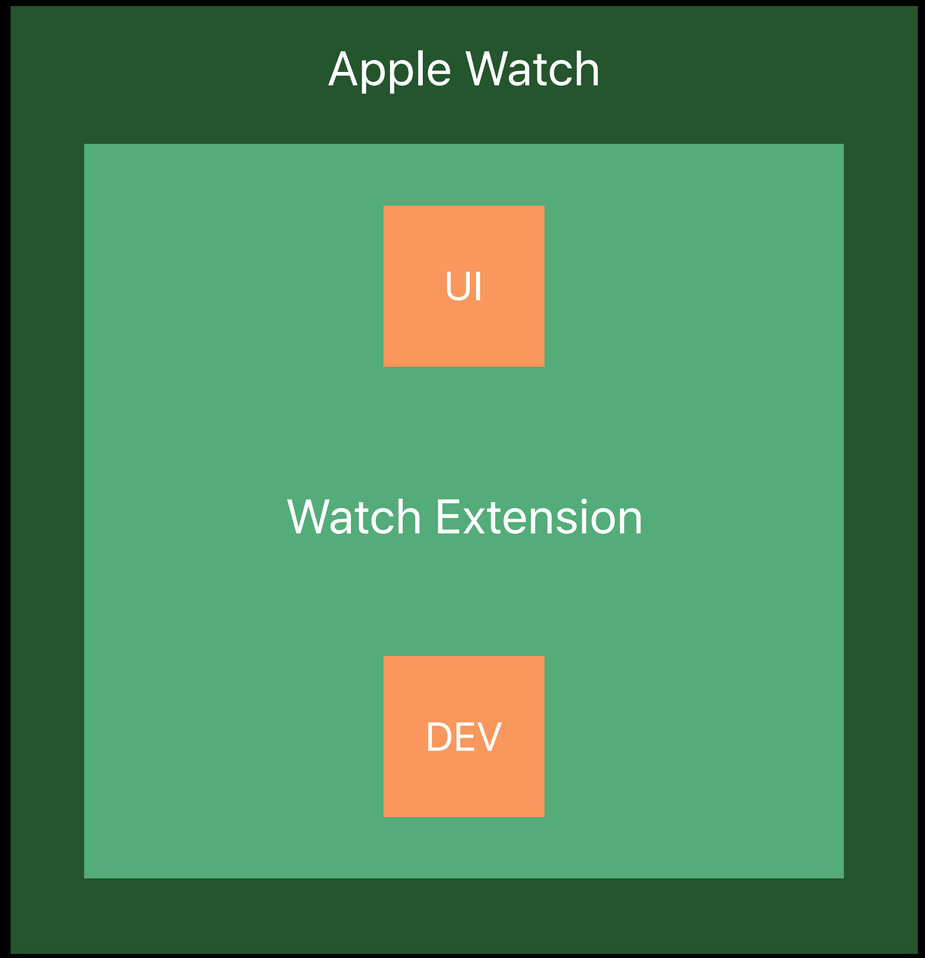
Architecture



Architecture



Architecture



Unified Process Runtime

Performance win

Unified Process Runtime

Performance win

- Touch latency improvements

Unified Process Runtime

Performance win

- Touch latency improvements
- Pan performance

Unified Process Runtime

Performance win

- Touch latency improvements
- Pan performance
- Launch performance

Unified Process Runtime

Performance win

- Touch latency improvements
- Pan performance
- Launch performance

Upped-memory limits accordingly

No changes required.

Frontmost App State

Apple Pie Me


10:09
Order a Pie!

Frontmost App State

Frontmost App State

Application state

Frontmost App State

Application state

Frontmost

Frontmost App State

Application state

Frontmost

Enhanced capabilities

Frontmost App State

Application state

Frontmost

Enhanced capabilities

Architecture


10:09
Order a Pie!

Active


10:09
2:00–3:30PM
Coffee with Forum
Central garden

Background

Active

Frontmost

Frontmost

Background


Timer
10:09
01:59
Reset
Resume


Timer
10:09
02:00
Reset
Pause


Timer
10:09
01:59
Reset
Resume

```
// Enable Extended Frontmost


override func willActivate() {
  WKExtension.shared().isFrontmostTimeoutExtended = true
}
```

```
// Enable Extended Frontmost


override func willActivate() {
  WKExtension.shared().isFrontmostTimeoutExtended = true
}
```

Active

Great, but what does this mean?

Enhanced Capabilities

Enhanced Capabilities

WatchConnectivity resumes

Enhanced Capabilities

WatchConnectivity resumes

NSURLSession resumes

Enhanced Capabilities

WatchConnectivity resumes

NSURLSession resumes

Task completion

Enhanced Capabilities

WatchConnectivity resumes

NSURLSession resumes

Task completion

Haptics

Enhanced Capabilities

WatchConnectivity resumes

NSURLSession resumes

Task completion

Haptics

Frontmost notification

Benefits of Being Frontmost

WatchConnectivity background transfer improvements

Benefits of Being Frontmost

WatchConnectivity background transfer improvements

NSURLSession transfer improvements

Benefits of Being Frontmost

WatchConnectivity background transfer improvements

NSURLSession transfer improvements

- Initiation of a download occurs immediately most times

Benefits of Being Frontmost

Task completions

Benefits of Being Frontmost

Task completions

- Prioritized

Benefits of Being Frontmost

Benefits of Being Frontmost

Haptics can be used to tap your user while frontmost

Benefits of Being Frontmost

Haptics can be used to tap your user while frontmost

Speaker conditions

Benefits of Being Frontmost

Haptics can be used to tap your user while frontmost

Speaker conditions

Bluetooth Headphone conditions

Benefits of Being Frontmost

Benefits of Being Frontmost

Receive remote and local

Benefits of Being Frontmost

Receive remote and local

Choose the right experience for your user

Benefits of Being Frontmost

Receive remote and local

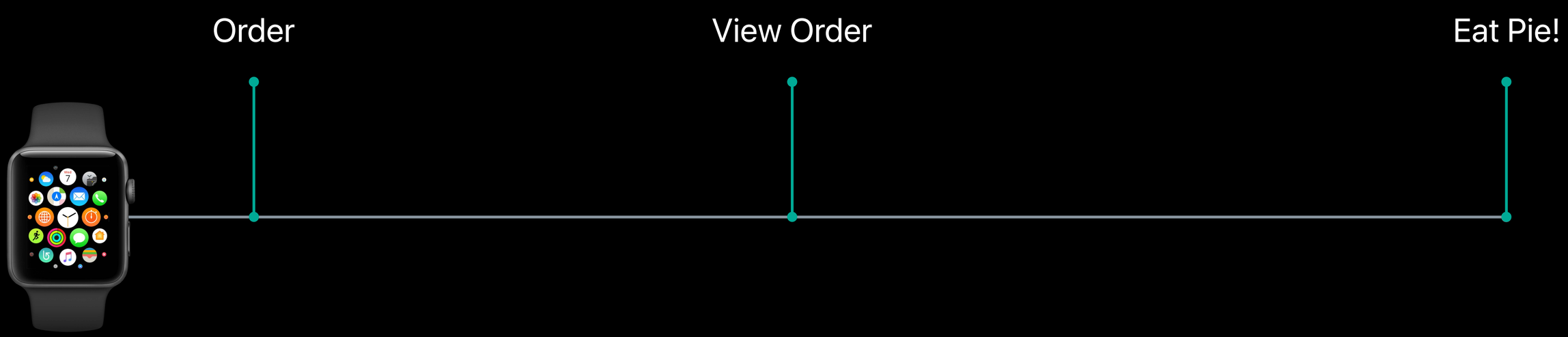
Choose the right experience for your user

```
func userNotificationCenter(_ center: UNUserNotificationCenter, willPresent
notification: UNNotification, withCompletionHandler completionHandler:
@escaping (UNNotificationPresentationOptions) -> Void)
```

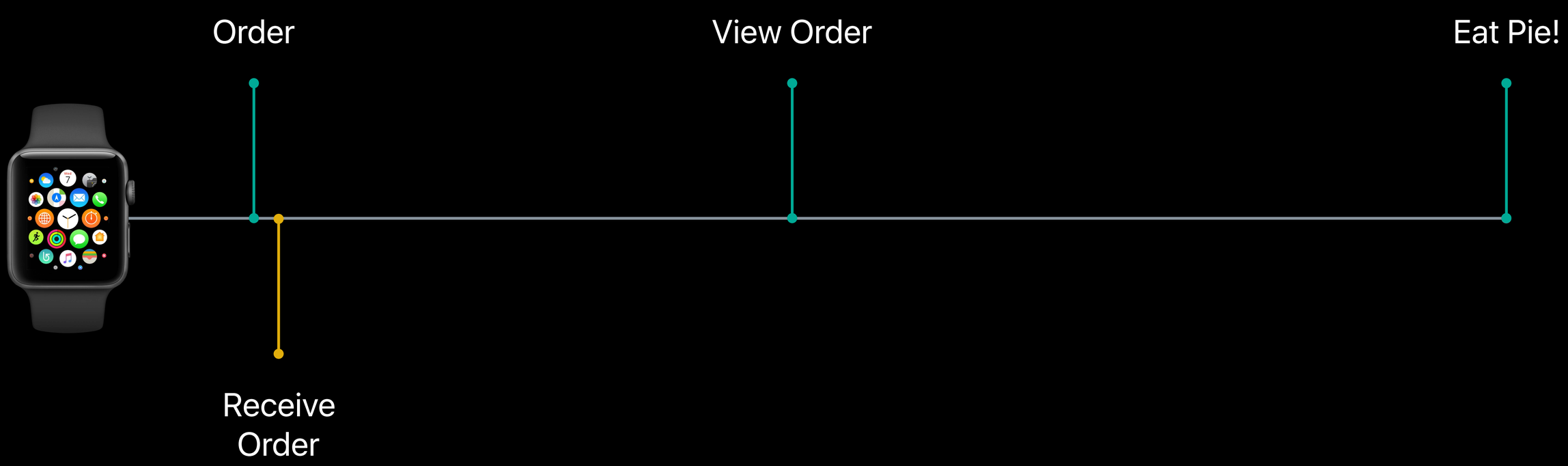
All of this is free.

Architect your apps for this experience.

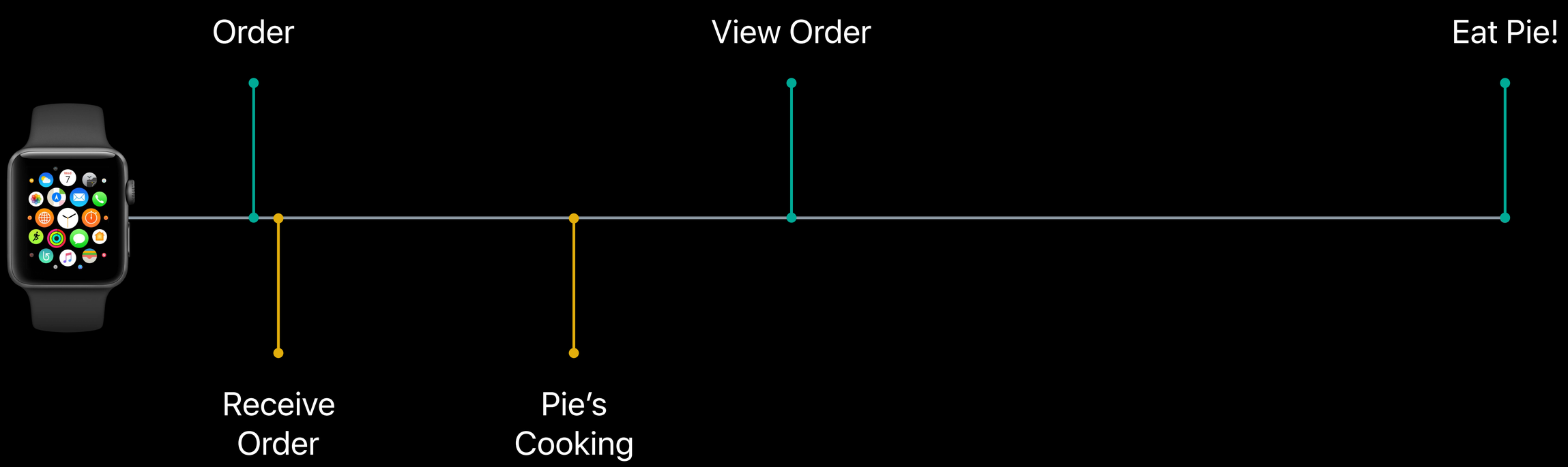
User Model



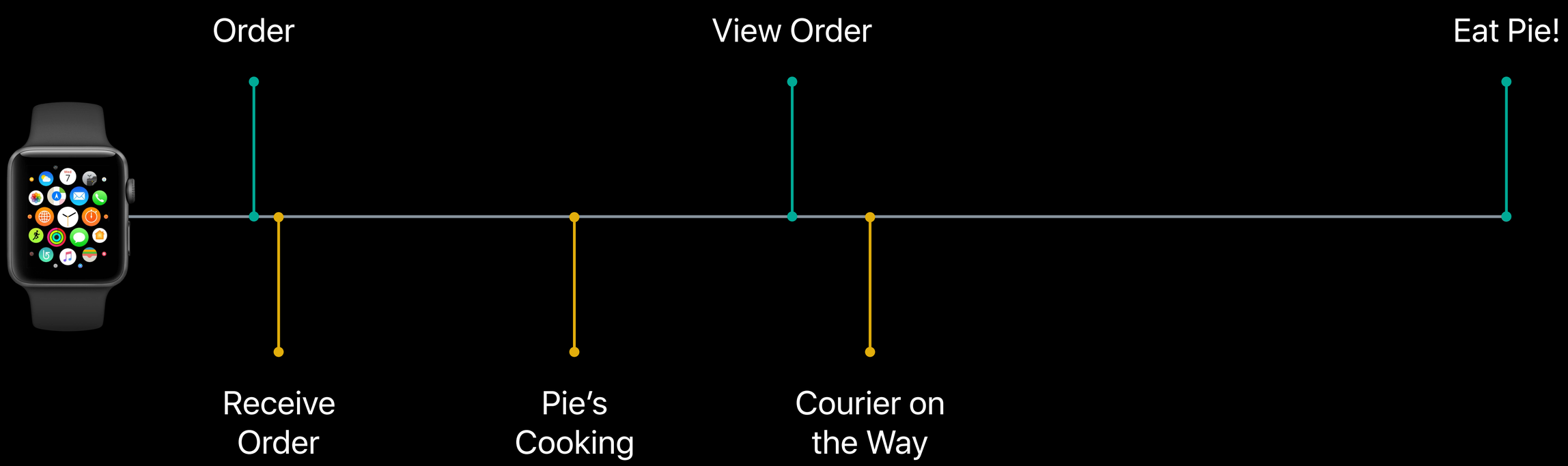
User/App Model



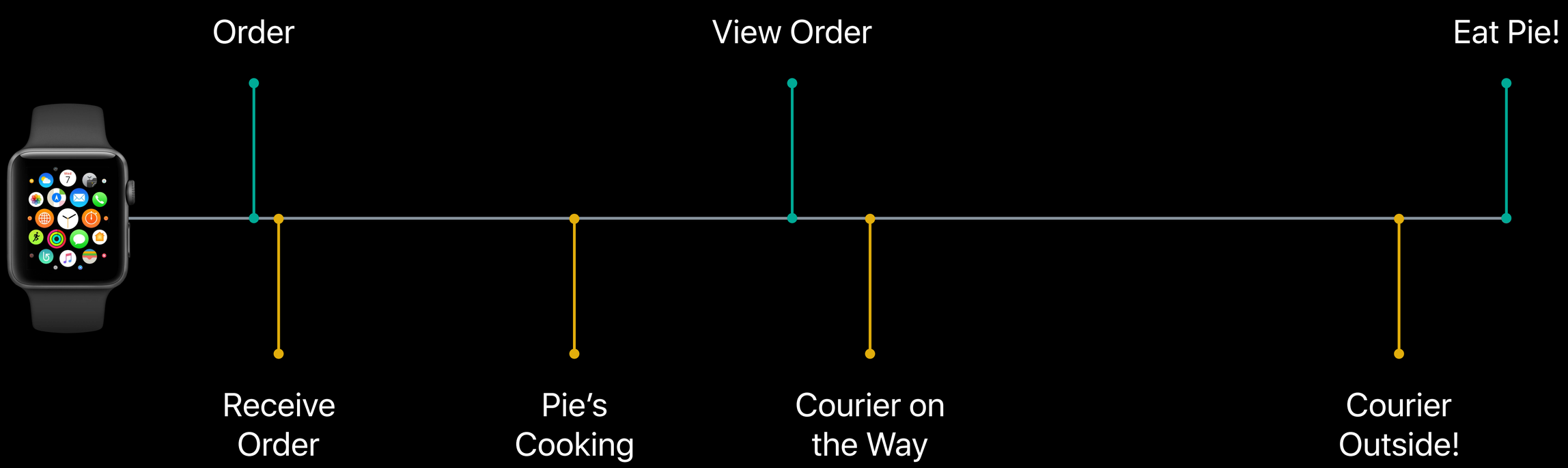
User/App Model



User/App Model



User/App Model



Receive Order

Receive Order


Apple Pie Me
10:09


Apple Pie Me
10:09
We'll tap you when
your pie is ready!

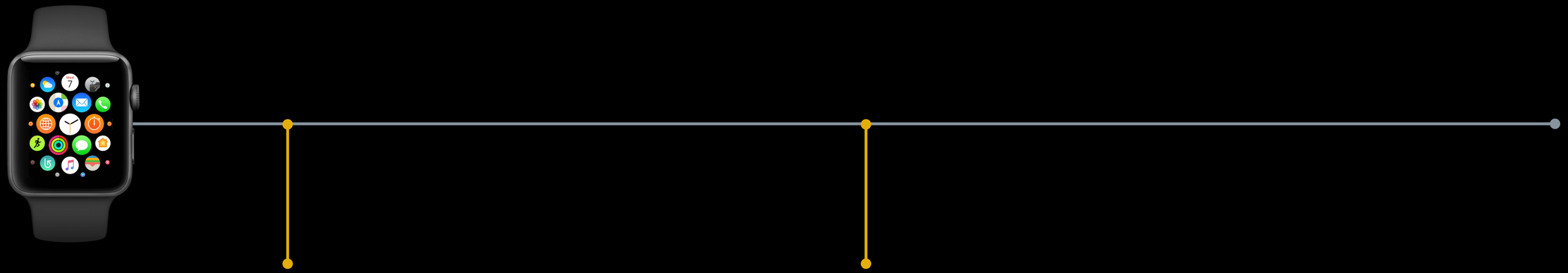
Receive Order

Receive Order

Timeline

Post Order

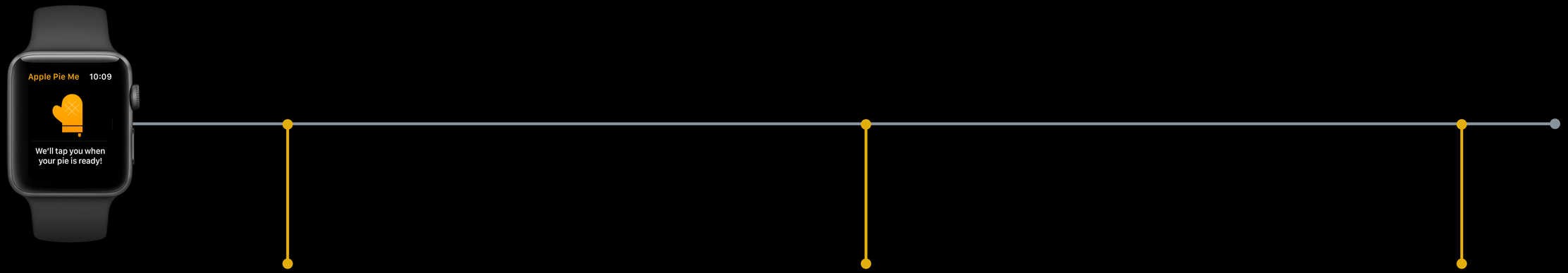
Timeline



Post Order

Enable
Timeout API

Timeline



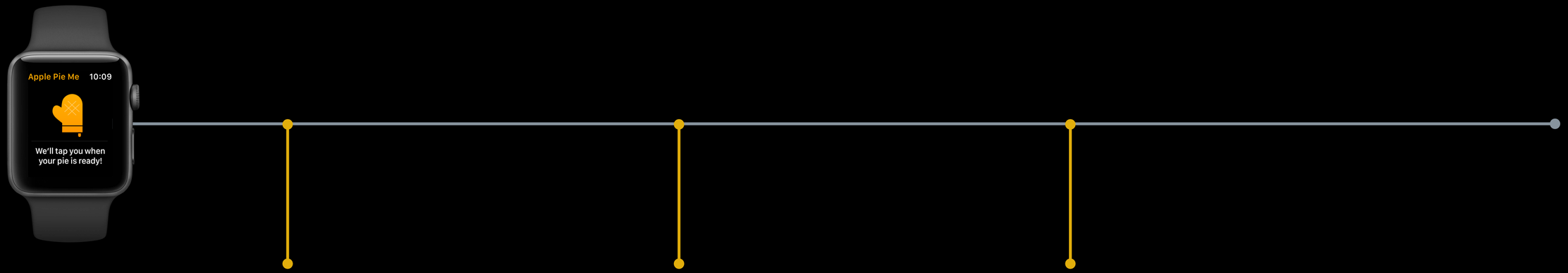
Post Order

Enable Timeout API

Update UI


10:09
APPLE PIE ME
Oh no! Something went wrong with your order
😞😞😞😞😞
Reorder

Timeline

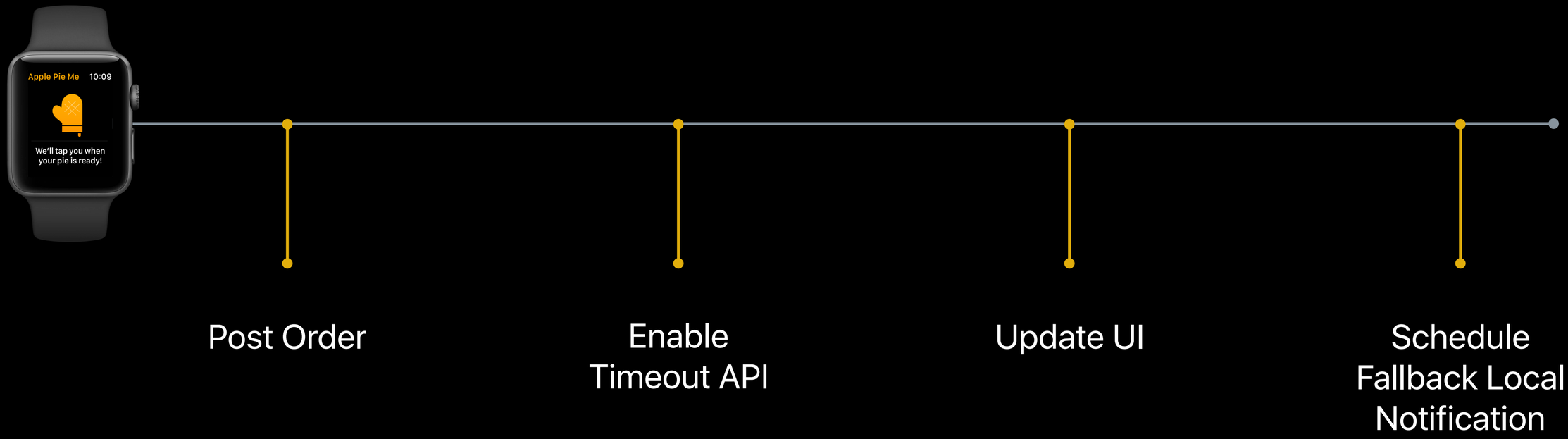


Post Order

Enable Timeout API

Update UI

Timeline


Apple Pie Me 10:09
We'll tap you when your pie is ready!
Post Order
Enable Timeout API
Update UI
Schedule Fallback Local Notification

```
@IBAction func orderButtonPressed() {
    let session = MyNSURLSessionDelegate.backgroundSession
    let task = session.downloadTask(with: orderPostURL)
    WKExtension.shared().isFrontmostTimeoutExtended = true
    triggerFallbackLocalNotification()
    WKInterfaceController.reloadRootPageControllers(withNames: ["orderController"],
        contexts: nil,
        orientation: .horizontal,
        pageIndex: 0)
}
```

```
@IBAction func orderButtonPressed() {
    let session = MyNSURLSessionDelegate.backgroundSession
    let task = session.downloadTask(with: orderPostURL)
    WKExtension.shared().isFrontmostTimeoutExtended = true
    triggerFallbackLocalNotification()
    WKInterfaceController.reloadRootPageControllers(withNames: ["orderController"],
        contexts: nil,
        orientation: .horizontal,
        pageIndex: 0)
}
```

```
@IBAction func orderButtonPressed() {
    let session = MyNSURLSessionDelegate.backgroundSession
    let task = session.downloadTask(with: orderPostURL)
    WKExtension.shared().isFrontmostTimeoutExtended = true
    triggerFallbackLocalNotification()
    WKInterfaceController.reloadRootPageControllers(withNames: ["orderController"],
        contexts: nil,
        orientation: .horizontal,
        pageIndex: 0)
}
```

```
@IBAction func orderButtonPressed() {
    let session = MyNSURLSessionDelegate.backgroundSession
    let task = session.downloadTask(with: orderPostURL)
    WKExtension.shared().isFrontmostTimeoutExtended = true
    triggerFallbackLocalNotification()
    WKInterfaceController.reloadRootPageControllers(withNames: ["orderController"],
        contexts: nil,
        orientation: .horizontal,
        pageIndex: 0)
}
```

```
@IBAction func orderButtonPressed() {
    let session = MyNSURLSessionDelegate.backgroundSession
    let task = session.downloadTask(with: orderPostURL)
    WKExtension.shared().isFrontmostTimeoutExtended = true
    triggerFallbackLocalNotification()
    WKInterfaceController.reloadRootPageControllers(withNames: ["orderController"],
        contexts: nil,
        orientation: .horizontal,
        pageIndex: 0)
}
```

```
@IBAction func orderButtonPressed() {
    let session = MyNSURLSessionDelegate.backgroundSession
    let task = session.downloadTask(with: orderPostURL)
    WKExtension.shared().isFrontmostTimeoutExtended = true
    triggerFallbackLocalNotification()
    WKInterfaceController.reloadRootPageControllers(withNames: ["orderController"],
        contexts: nil,
        orientation: .horizontal,
        pageIndex: 0)
}
```

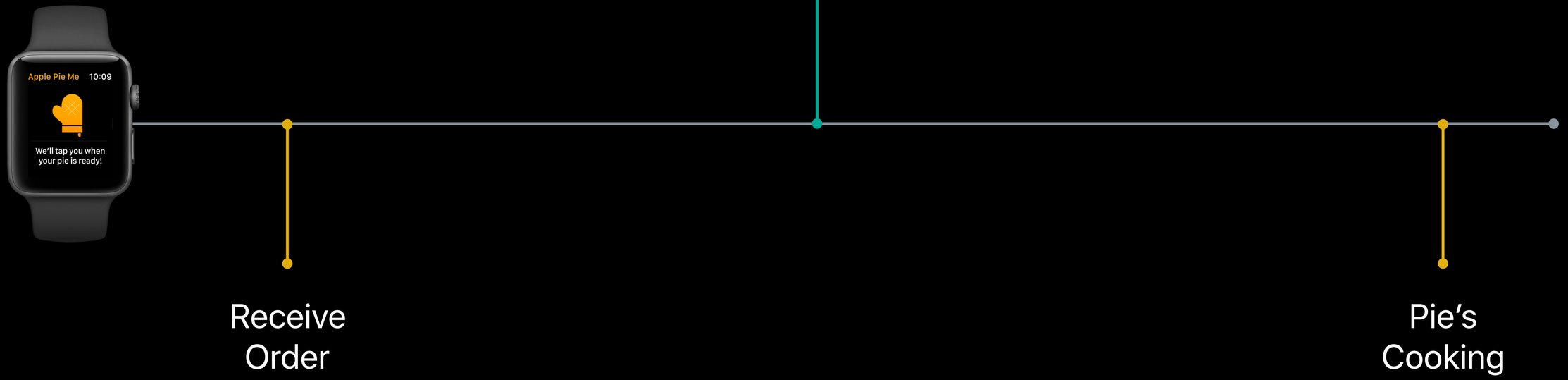
User Model

Receive Order

User Model


Apple Pie Me 10:09
We'll tap you when your pie is ready!
Receive Order
Pie's Cooking

User Model


Apple Pie Me 10:09
We'll tap you when your pie is ready!
User Puts Wrist Down
Receive Order
Pie's Cooking

Pie’s Cooking

Pie’s Cooking

```
override func userNotificationCenter(_ center: UNUserNotificationCenter,
                                     willPresent notification: UNNotification,
                                     withCompletionHandler completionHandler: @escaping
                                        (UNNotificationPresentationOptions) -> Void) {

    cancelFallbackNotifications()
     WKInterfaceController.reloadRootPageControllers(withNames: ["cookingController"],
         contexts: nil,
         orientation: .horizontal,
         pageIndex: 0)

     WKInterfaceDevice.current().play(.success)
     completionHandler()
}
```

```
override func userNotificationCenter(_ center: UNUserNotificationCenter,
                                     willPresent notification: UNNotification,
                                     withCompletionHandler completionHandler: @escaping
                                        (UNNotificationPresentationOptions) -> Void) {

    cancelFallbackNotifications()
     WKInterfaceController.reloadRootPageControllers(withNames: ["cookingController"],
         contexts: nil,
         orientation: .horizontal,
         pageIndex: 0)

     WKInterfaceDevice.current().play(.success)
     completionHandler()
}
```

```
override func userNotificationCenter(_ center: UNUserNotificationCenter,
                                     willPresent notification: UNNotification,
                                     withCompletionHandler completionHandler: @escaping
                                        (UNNotificationPresentationOptions) -> Void) {

    cancelFallbackNotifications()
    WKInterfaceController.reloadRootPageControllers(withNames: ["cookingController"],
        contexts: nil,
        orientation: .horizontal,
        pageIndex: 0)

    WKInterfaceDevice.current().play(.success)
    completionHandler()
}
```

```
override func userNotificationCenter(_ center: UNUserNotificationCenter,
                                     willPresent notification: UNNotification,
                                     withCompletionHandler completionHandler: @escaping
                                        (UNNotificationPresentationOptions) -> Void) {

    cancelFallbackNotifications()
    WKInterfaceController.reloadRootPageControllers(withNames: ["cookingController"],
        contexts: nil,
        orientation: .horizontal,
        pageIndex: 0)

    WKInterfaceDevice.current().play(.success)
    completionHandler()
}
```

```
override func userNotificationCenter(_ center: UNUserNotificationCenter,
                                     willPresent notification: UNNotification,
                                     withCompletionHandler completionHandler: @escaping
                                        (UNNotificationPresentationOptions) -> Void) {

    cancelFallbackNotifications()
     WKInterfaceController.reloadRootPageControllers(withNames: ["cookingController"],
         contexts: nil,
         orientation: .horizontal,
         pageIndex: 0)

     WKInterfaceDevice.current().play(.success)
     completionHandler()
}
```

```
override func userNotificationCenter(_ center: UNUserNotificationCenter,
                                     willPresent notification: UNNotification,
                                     withCompletionHandler completionHandler: @escaping
                                        (UNNotificationPresentationOptions) -> Void) {

    cancelFallbackNotifications()
     WKInterfaceController.reloadRootPageControllers(withNames: ["cookingController"],
         contexts: nil,
         orientation: .horizontal,
         pageIndex: 0)

     WKInterfaceDevice.current().play(.success)
     completionHandler()
}
```


10:09
12
Minutes
Your pie is cooking!

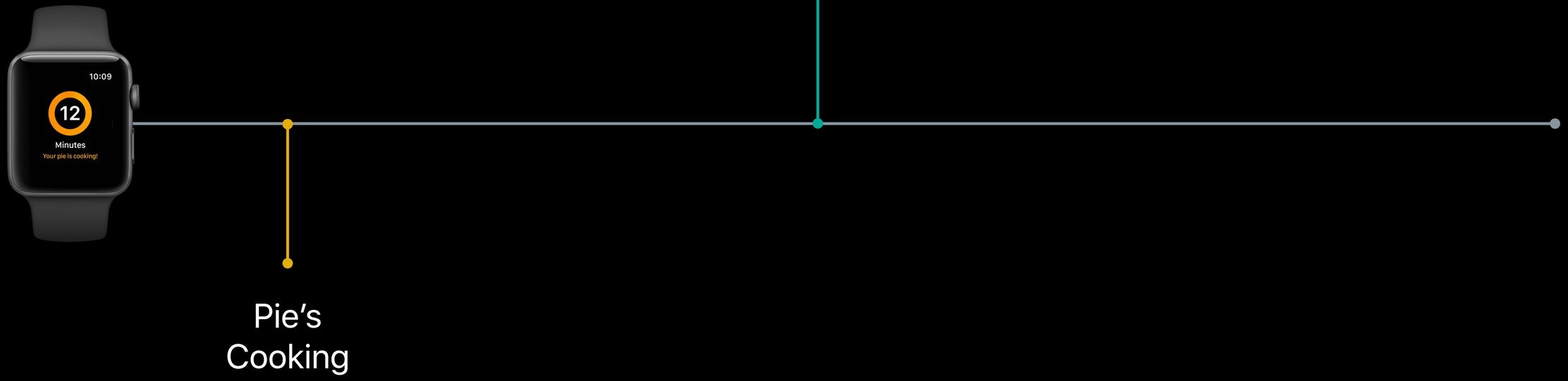
User Model

Pie’s Cooking

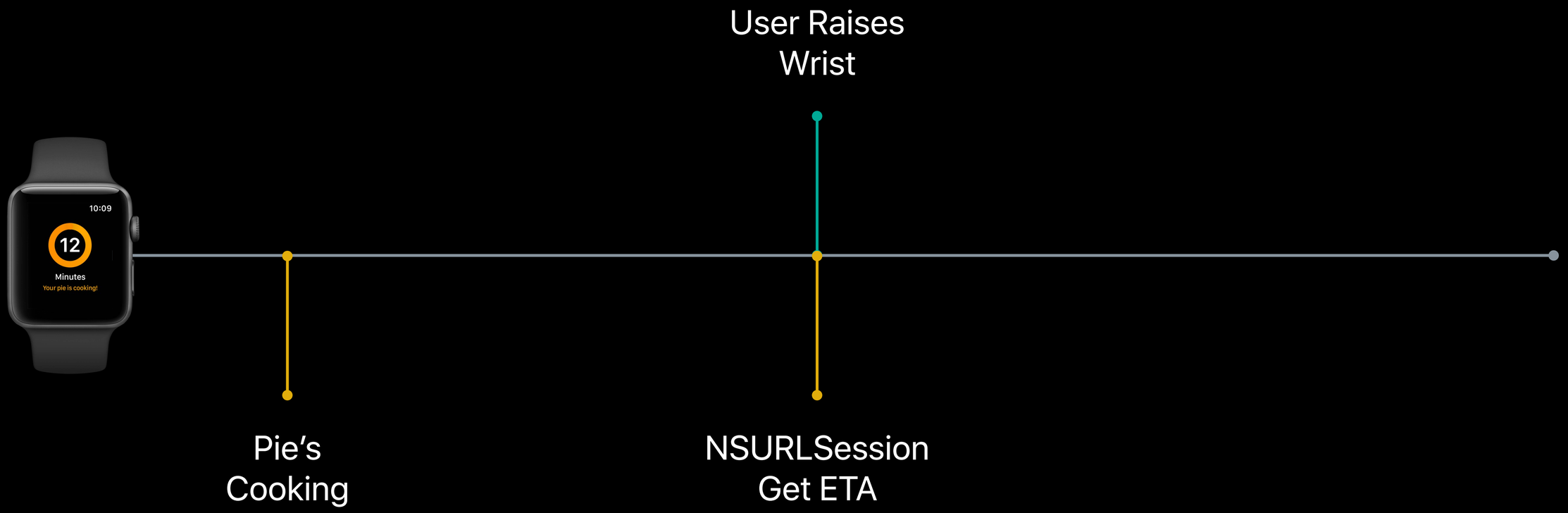
User Model

Pie’s
Cooking

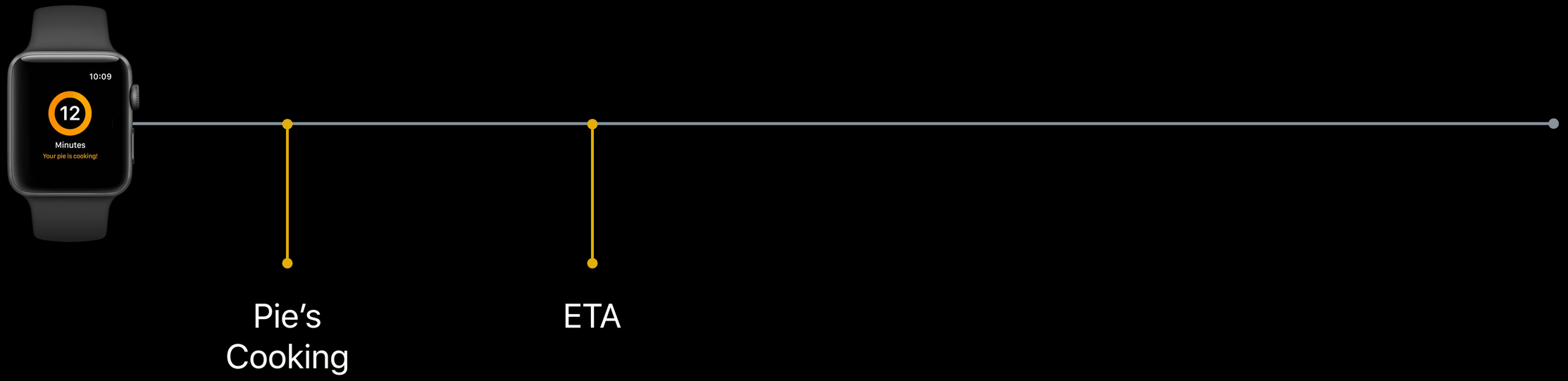
User Model


10:09
12
Minutes
Your pie is cooking!
User Raises Wrist
Pie's Cooking

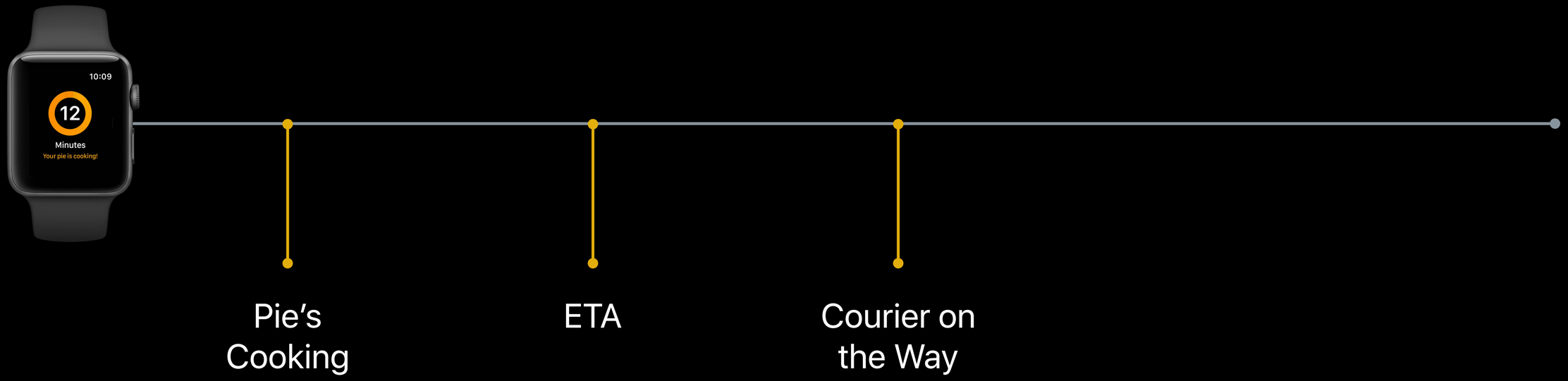
User Model


User Raises
Wrist
10:09
12
Minutes
Your pie is cooking!
Pie's
Cooking
NSURLSession
Get ETA

App Model



App Model

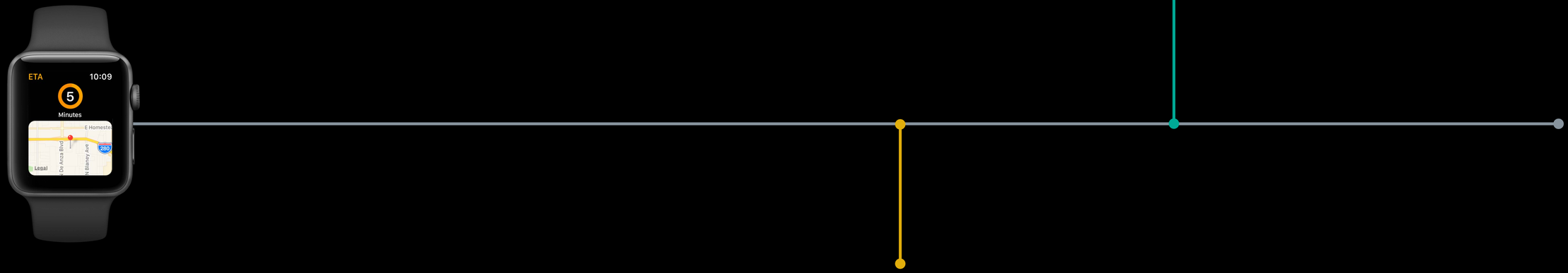

10:09
12
Minutes
Your pie is cooking!
Pie’s Cooking
ETA
Courier on the Way


ETA
10:09
5
Minutes
E Homestea
280
N De Anza Blvd
N Blaney Ave
Legal

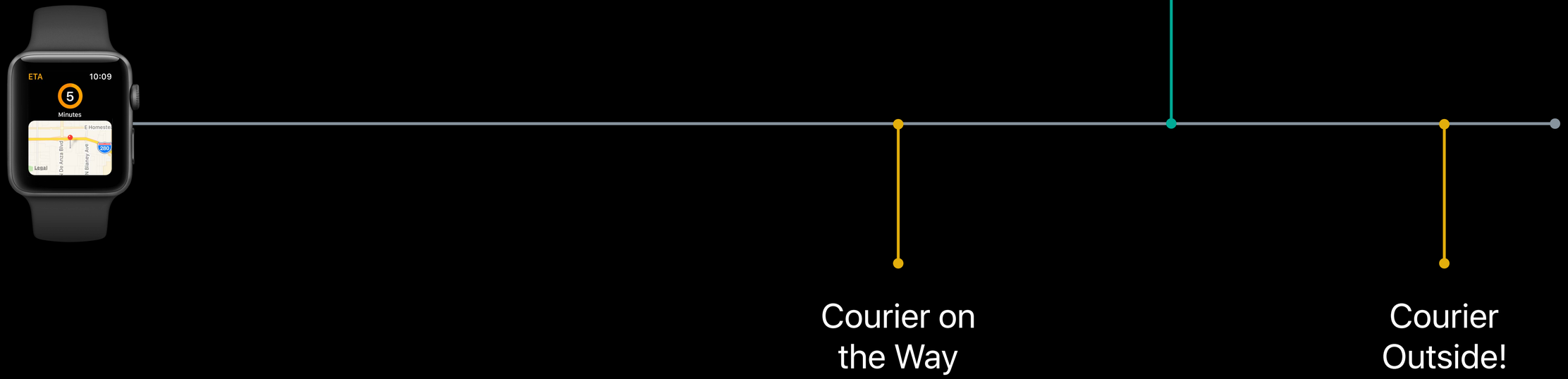
User Model

Courier on
the Way

User Model

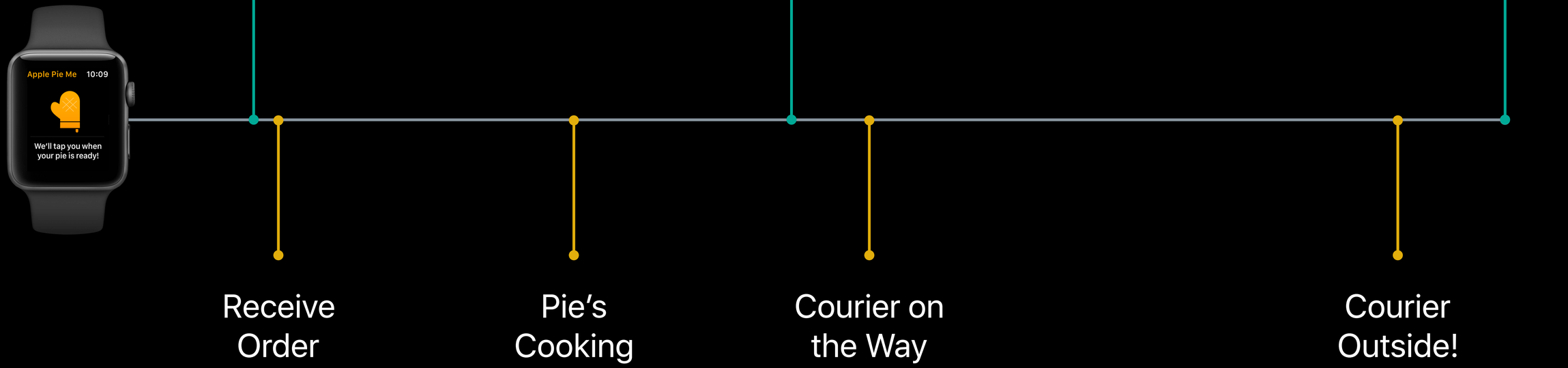

ETA
10:09
5
Minutes
E Homestead
De Anza Blvd
N Blaney Ave
280
Legal
User Raises Wrist
Courier on the Way

User Model

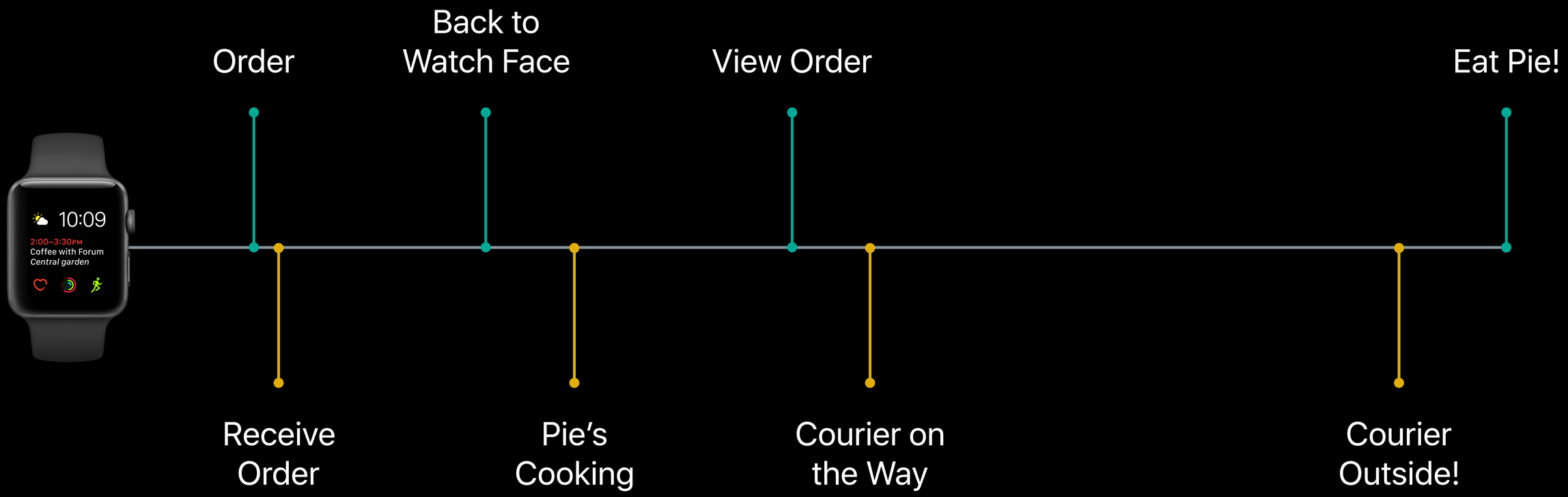

ETA
10:09
5
Minutes
User Raises Wrist
Courier on the Way
Courier Outside!


Eat!
10:09
Your pie is here!
E Homestea
280
N De Anza Blvd
N Blaney Ave
Legal

User Model


Order
View Order
Eat Pie!
Apple Pie Me 10:09
We'll tap you when your pie is ready!
Receive Order
Pie's Cooking
Courier on the Way
Courier Outside!

User Model


Order
Back to Watch Face
View Order
Eat Pie!
10:09
2:00–3:30PM
Coffee with Forum
Central garden
Receive Order
Pie's Cooking
Courier on the Way
Courier Outside!

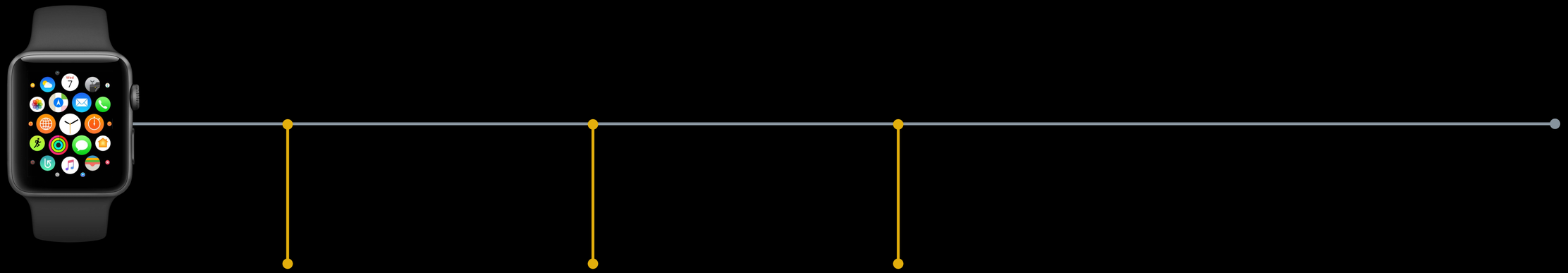
Timeline

Timeline

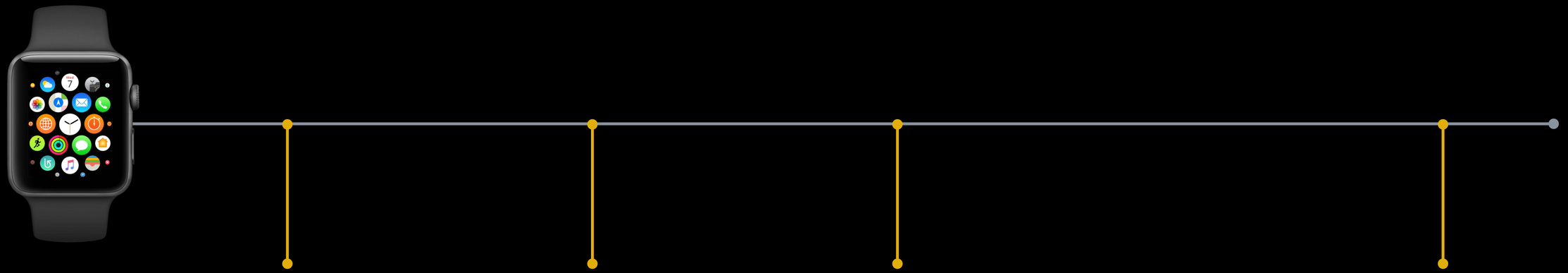
Receive
Order

Timeline

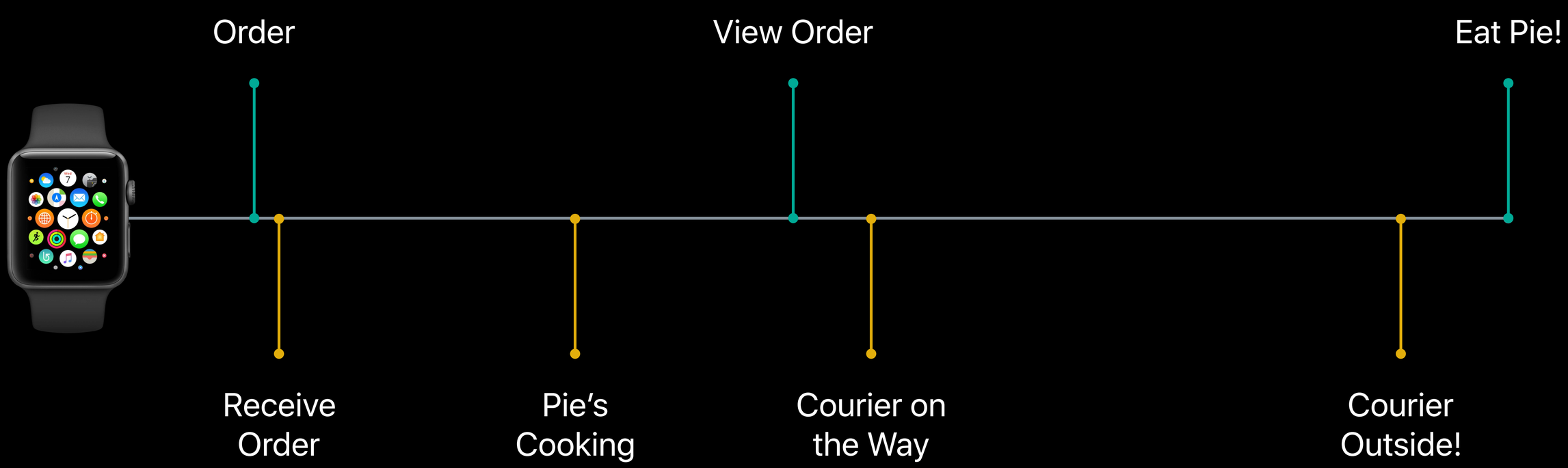
Timeline



Timeline


Receive Order
Pie's Cooking
Courier on the Way
Courier Outside!

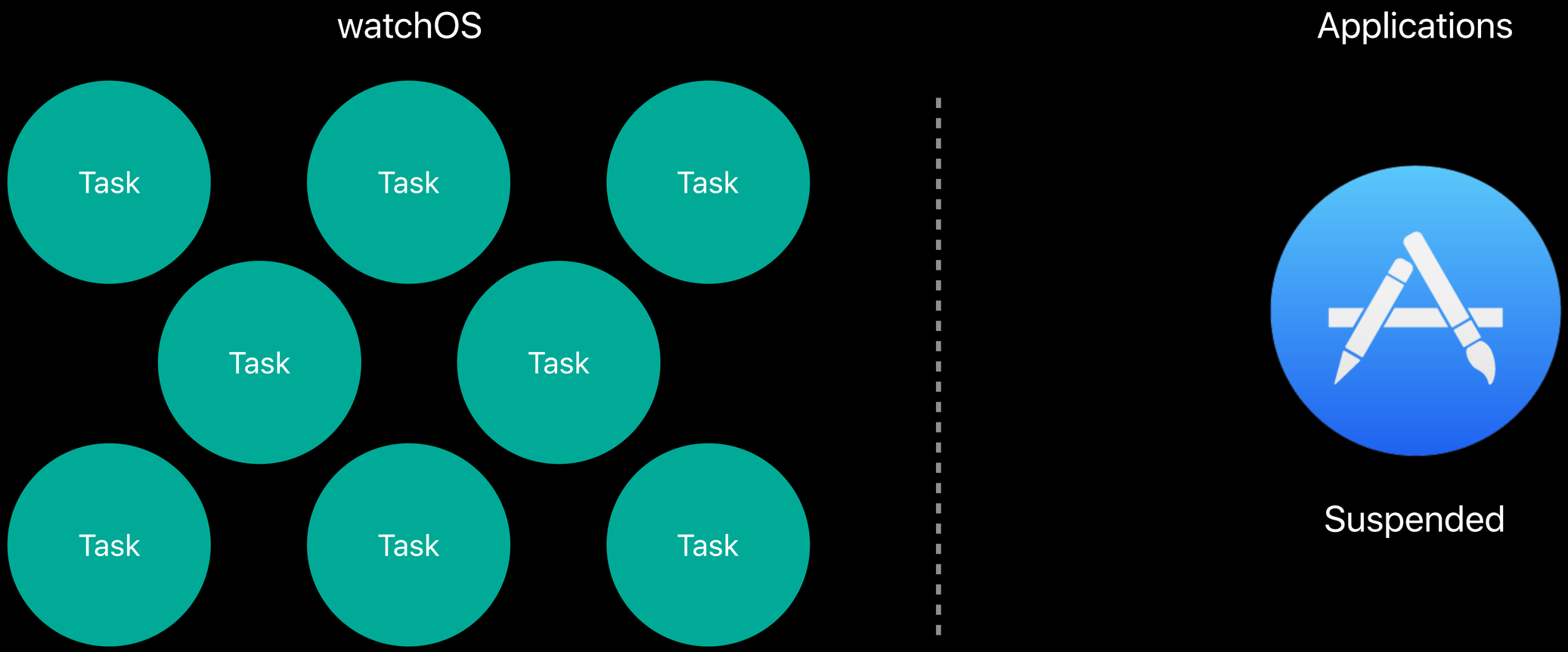
Review


Order
View Order
Eat Pie!
Receive Order
Pie’s Cooking
Courier on the Way
Courier Outside!

Background App Refresh

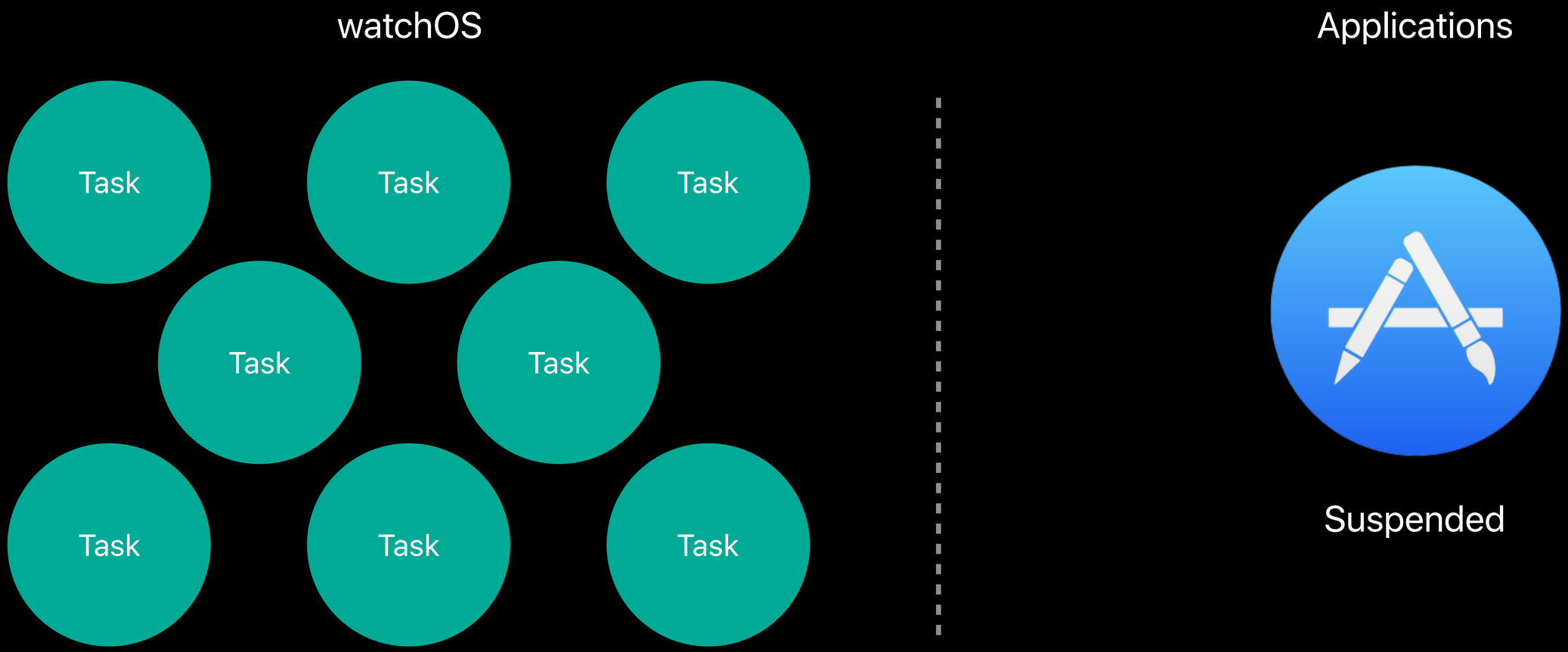
Background App Refresh

Overview



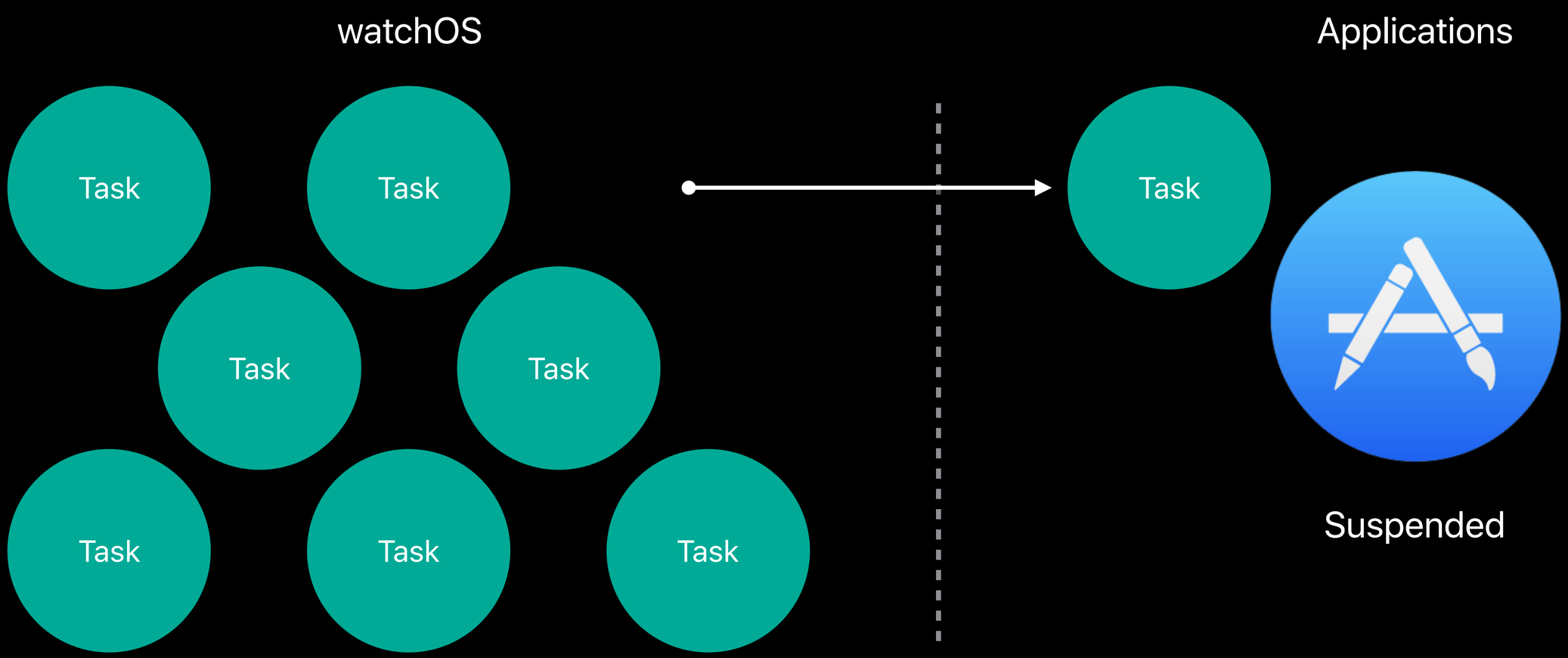
Background App Refresh

Overview



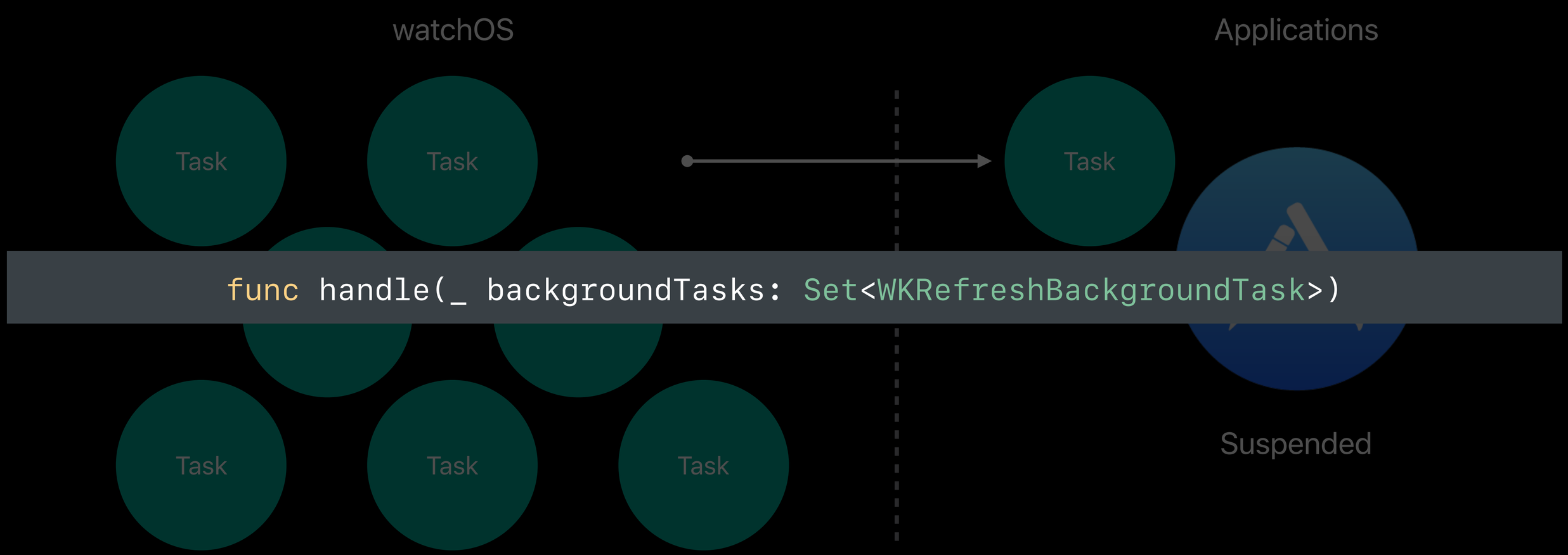
Background App Refresh

Overview



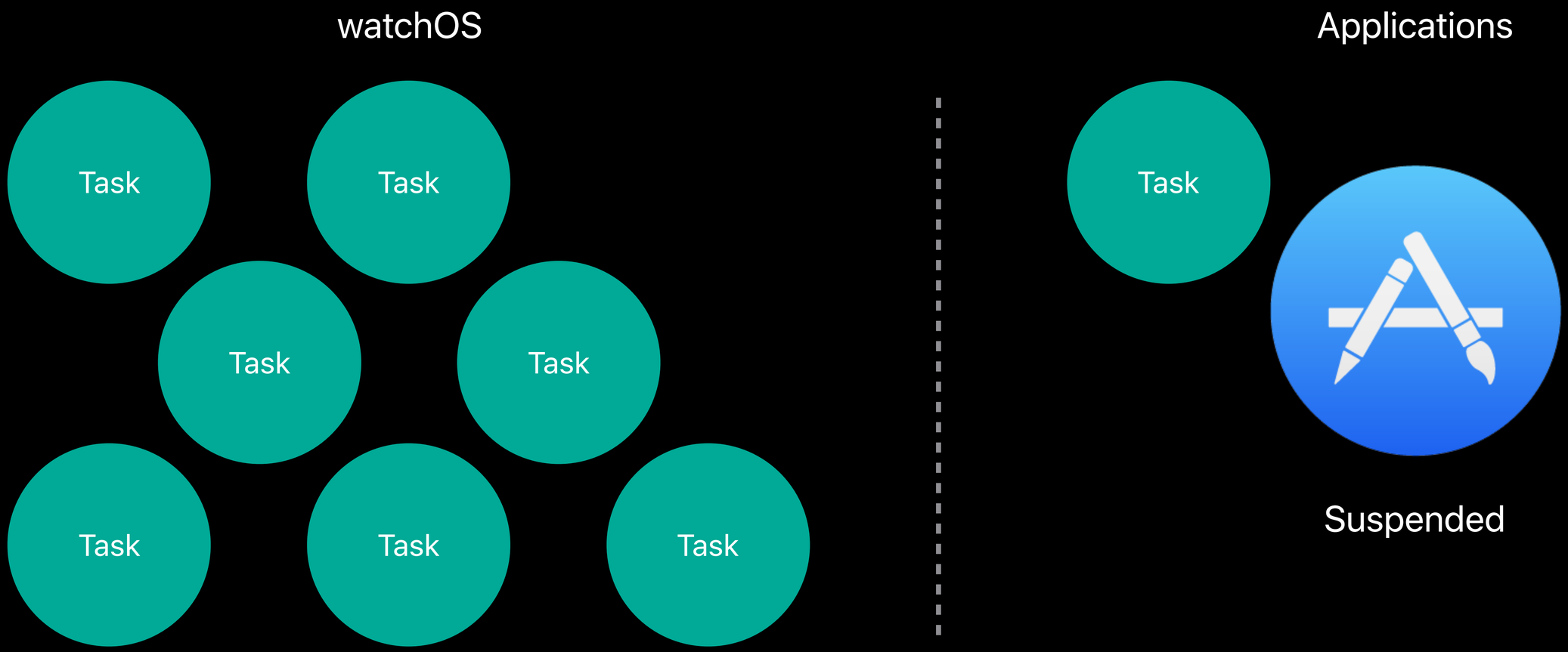
Background App Refresh

Overview



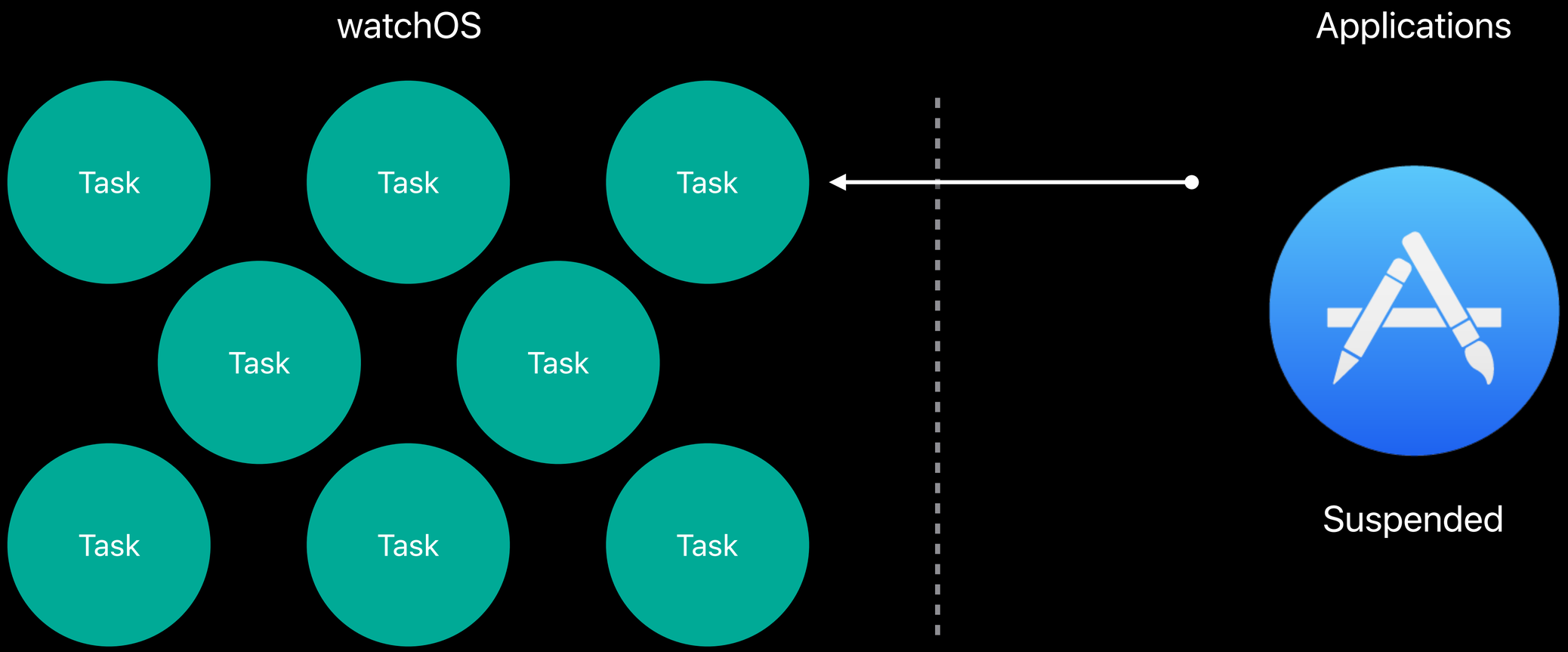
Background App Refresh

Overview



Background App Refresh

Overview



API Changes

```
func handle(_ backgroundTasks: Set<WKRefreshBackgroundTask>) {
    for task in backgroundTasks {
        switch task {
        case let backgroundTask as WKApplicationRefreshBackgroundTask:
            // Perform functions as needed
            WKExtension.shared().scheduleSnapshotRefresh(withPreferredDate: fireDate,
                                 userInfo: nil) { error in
                                      // code
                                 }
            backgroundTask.setTaskCompleted()
        case ...// handle other task types
    }
}
```

```
func handle(_ backgroundTasks: Set<WKRefreshBackgroundTask>) {
    for task in backgroundTasks {
        switch task {
        case let backgroundTask as WKApplicationRefreshBackgroundTask:
            // Perform functions as needed
            WKExtension.shared().scheduleSnapshotRefresh(withPreferredDate: fireDate,
                                 userInfo: nil) { error in
                                      // code
                                 }
            backgroundTask.setTaskCompleted()
        case ...// handle other task types
    }
}
```

```
func handle(_ backgroundTasks: Set<WKRefreshBackgroundTask>) {
    for task in backgroundTasks {
        switch task {
        case let backgroundTask as WKApplicationRefreshBackgroundTask:
            // Perform functions as needed
            WKExtension.shared().scheduleSnapshotRefresh(withPreferredDate: fireDate,
                                 userInfo: nil) { error in
                                      // code
                                 }
            backgroundTask.setTaskCompleted()
        case ...// handle other task types
    }
}
```

```
func handle(_ backgroundTasks: Set<WKRefreshBackgroundTask>) {
    for task in backgroundTasks {
        switch task {
        case let backgroundTask as WKApplicationRefreshBackgroundTask:
            // Perform functions as needed
            backgroundTask.setTaskCompletedWithSnapshot(true)
        case ...// handle other task types
    }
}
```

```
func handle(_ backgroundTasks: Set<WKRefreshBackgroundTask>) {
    for task in backgroundTasks {
        switch task {
        case let backgroundTask as WKApplicationRefreshBackgroundTask:
            // Perform functions as needed
            backgroundTask.setTaskCompletedWithSnapshot(true)
        case ...// handle other task types
    }
}
```

```
func handle(_ backgroundTasks: Set<WKRefreshBackgroundTask>) {
    for task in backgroundTasks {
        switch task {
        case let snapshotTask as WKSnapshotRefreshBackgroundTask:
            // Perform functions as needed
            // Snapshot tasks have a unique completion call, make sure to set
            // your expiration date
            snapshotTask.setTaskCompleted(restoredDefaultState: true,
                estimatedSnapshotExpiration: Date.distantFuture,
                userInfo: nil)
        case ...// handle other task types
    }
}
```

```
func handle(_ backgroundTasks: Set<WKRefreshBackgroundTask>) {
    for task in backgroundTasks {
        switch task {
        case let snapshotTask as WKSnapshotRefreshBackgroundTask:
            // Perform functions as needed
            // Snapshot tasks have a unique completion call, make sure to set
            // your expiration date
            snapshotTask.setTaskCompleted(restoredDefaultState: true,
                estimatedSnapshotExpiration: Date.distantFuture,
                userInfo: nil)
        case ...// handle other task types
    }
}
```

```swift
func handle(_ backgroundTasks: Set<WKRefreshBackgroundTask>) {
    for task in backgroundTasks {
        switch task {
        case let snapshotTask as WKSnapshotRefreshBackgroundTask:
            // Perform functions as needed
            snapshotTask.setTaskCompletedWithSnapshot(true)
        case ...// handle other task types
    }
}
```

```
func handle(_ backgroundTasks: Set<WKRefreshBackgroundTask>) {
    for task in backgroundTasks {
        switch task {
        case let snapshotTask as WKSnapshotRefreshBackgroundTask:
            // Perform functions as needed
            snapshotTask.setTaskCompletedWithSnapshot(true)
        case ...// handle other task types
    }
}
```

```
// Deprecated in watchOS 4.0
// Not called in watchOS 3, if you implement handle(_ backgroundTasks:)

public protocol CLKComplicationDataSource: NSObjectProtocol {
    optional public func getNextRequestedUpdateDate(handler: @escaping (Date?) -> Swift.Void)
    optional public func requestedUpdateDidBegin()
    optional public func requestedUpdateBudgetExhausted()
}
```

```
// Use for Scheduling Complication Updates in watchOS 3 and Greater


public protocol WKExtensionDelegate: NSObjectProtocol {
     optional public func handle(_ backgroundTasks: Set<WKRefreshBackgroundTask>)
}

extension WKExtension {
     open func scheduleBackgroundRefresh(withPreferredDate preferredFireDate: Date,
                                         userInfo: NSSecureCoding?,
                        scheduledCompletion: @escaping (Error?) -> Swift.Void)
}
```

// Use for Scheduling Complication Updates in watchOS 3 and Greater

```
public protocol WKExtensionDelegate: NSObjectProtocol {
    optional public func handle(_ backgroundTasks: Set<WKRefreshBackgroundTask>)
}

extension WKExtension {
    open func scheduleBackgroundRefresh(withPreferredDate preferredFireDate: Date,
                                        userInfo: NSSecureCoding?,
                                        scheduledCompletion: @escaping (Error?) -> Swift.Void)
}
```

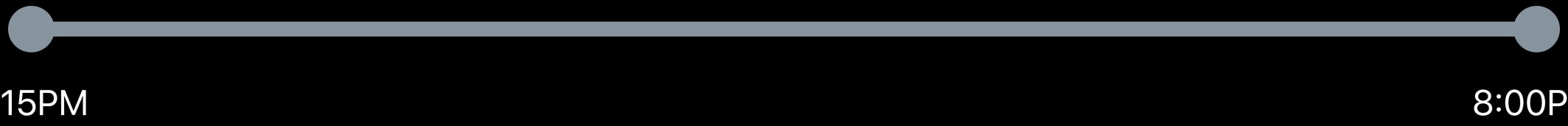
```
// Use for Scheduling Complication Updates in watchOS 3 and Greater
```

```
public protocol WKExtensionDelegate: NSObjectProtocol {
    optional public func handle(_ backgroundTasks: Set<WKRefreshBackgroundTask>)
}
```

```
extension WKExtension {
    open func scheduleBackgroundRefresh(withPreferredDate preferredFireDate: Date,
                                        userInfo: NSSecureCoding?,
                                        scheduledCompletion: @escaping (Error?) -> Swift.Void)
}
```

Background App Refresh

Tips

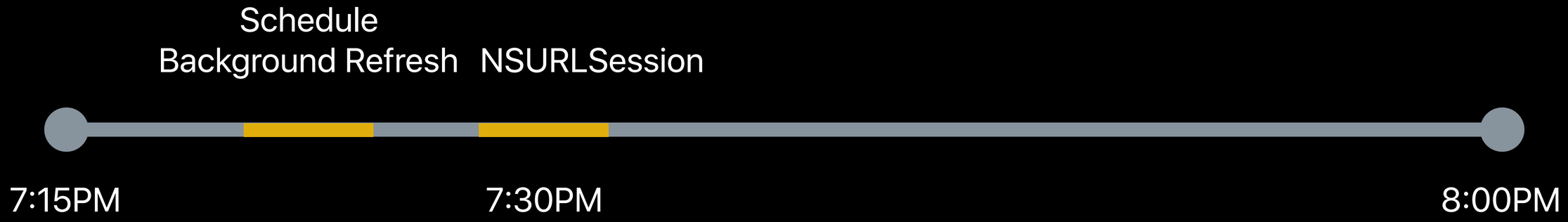


Background App Refresh

Tips

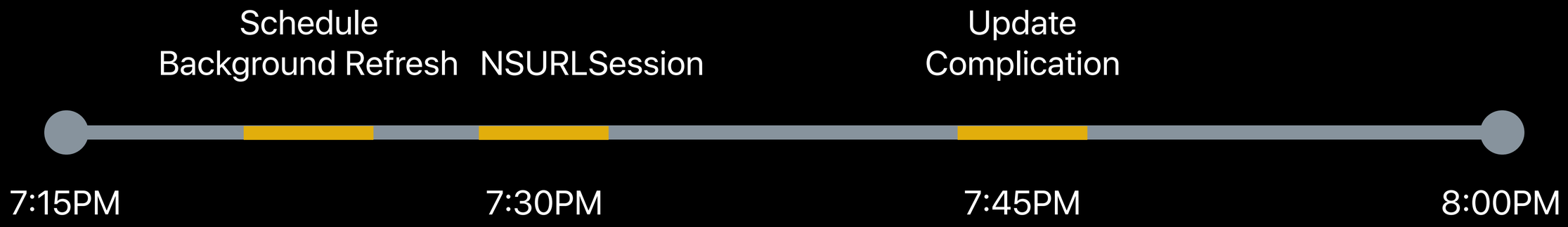
Background App Refresh

Tips



Background App Refresh

Tips

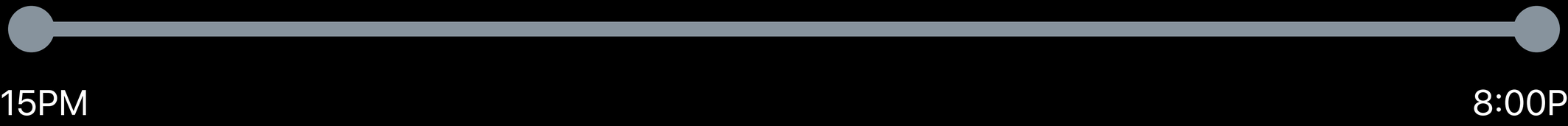


```
// New NSURLSessionTask API

class URLSessionTask: NSObject, NSCopying, NSProgressReporting  {
     var earliestBeginDate : Date?
}
```

```
// New NSURLSessionTask API

class URLSessionTask: NSObject, NSCopying, NSProgressReporting  {
     var earliestBeginDate : Date?
}
```

Background App Refresh

Tips



Background App Refresh

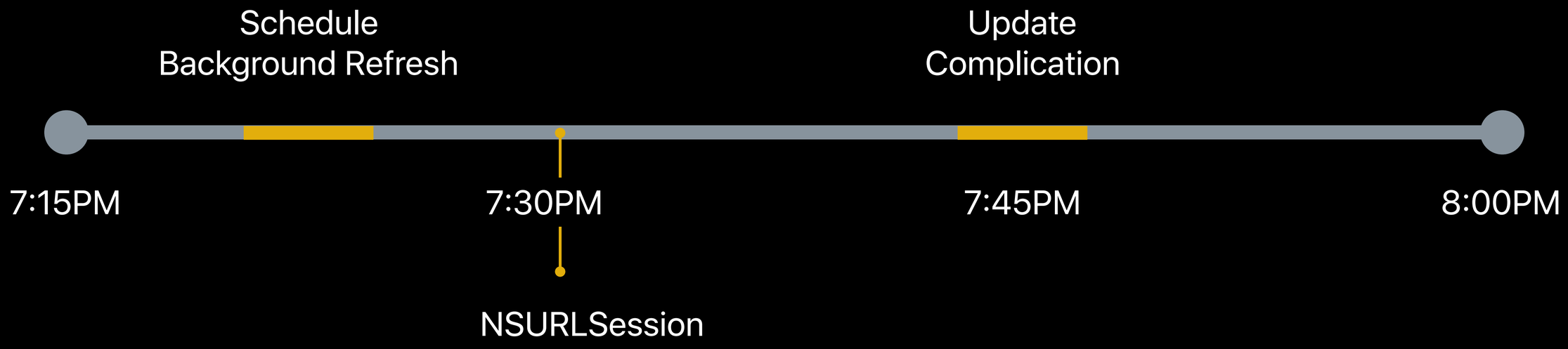
Tips

Background App Refresh

Tips

Background App Refresh

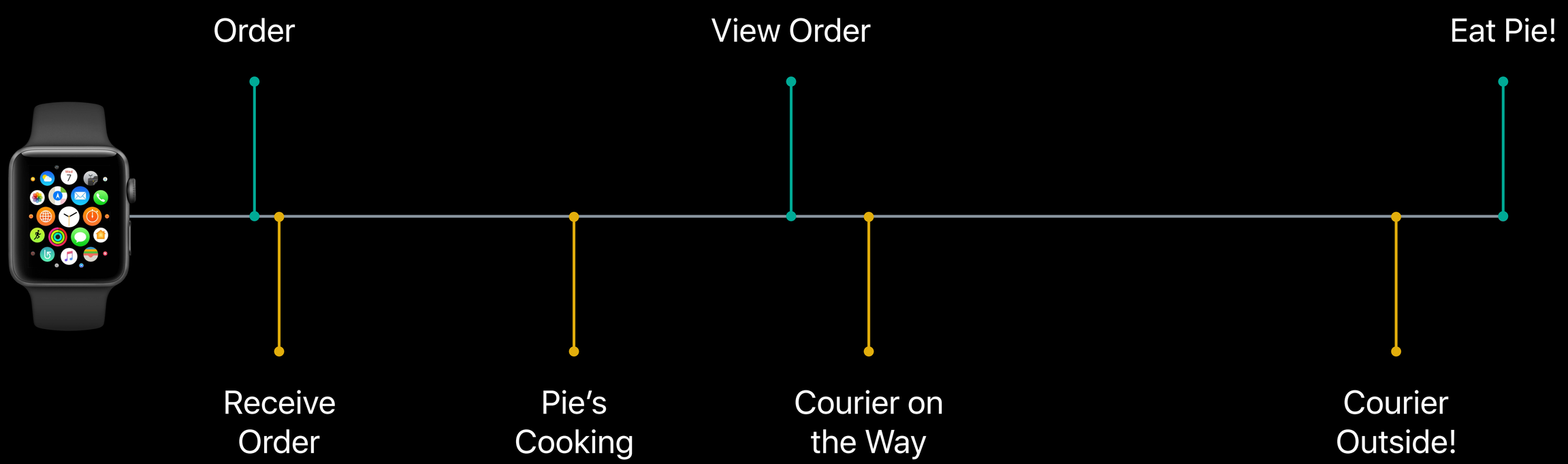
Tips



Apple Pie Me

Background App Refresh

Apple Pie Me



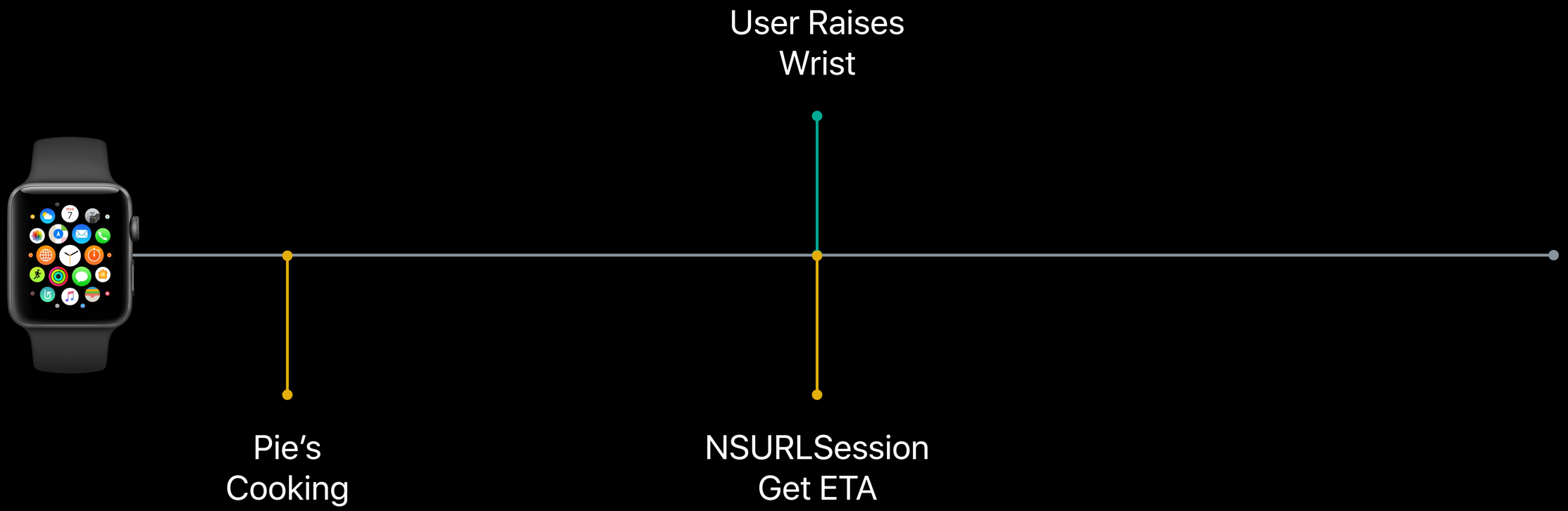
Background App Refresh

Apple Pie Me

```
override func userNotificationCenter(_ center: UNUserNotificationCenter,
                                     willPresent notification: UNNotification,
                                     withCompletionHandler completionHandler: @escaping
                                        (UNNotificationPresentationOptions) -> Void) {

    cancelFallbackNotifications()
     WKInterfaceController.reloadRootPageControllers(withNames: ["cookingController"],
         contexts: nil,
         orientation: .horizontal,
         pageIndex: 0)

     WKInterfaceDevice.current().play(.success)
     completionHandler()
}
```

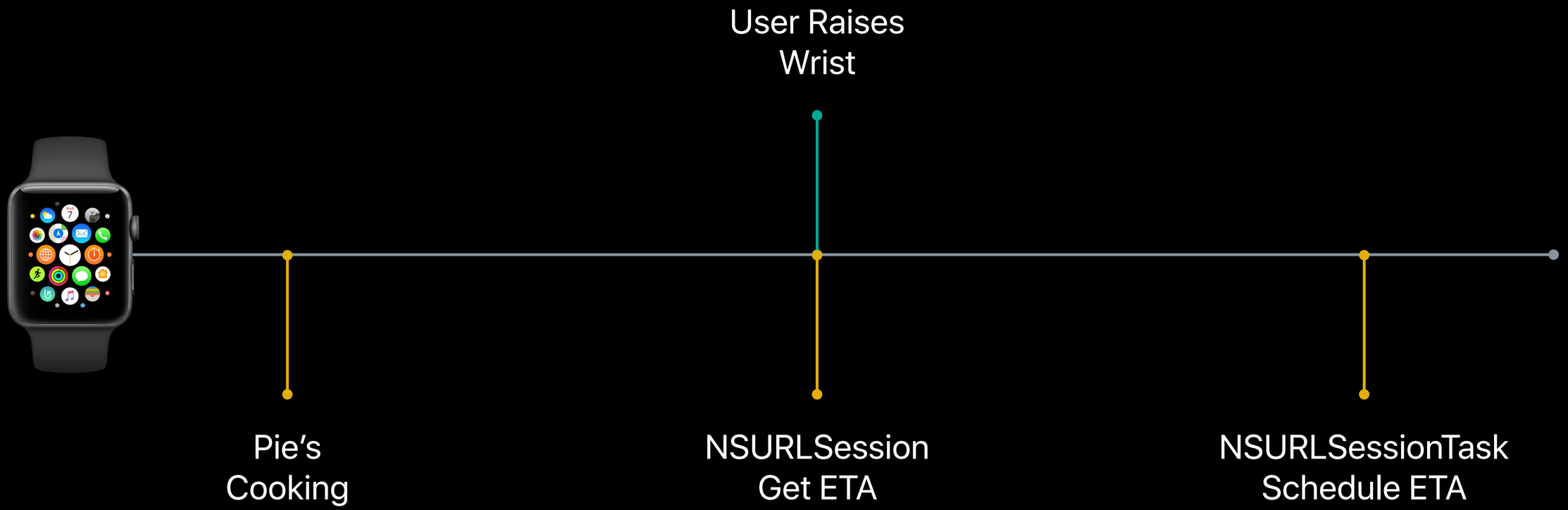
Background App Refresh

Apple Pie Me



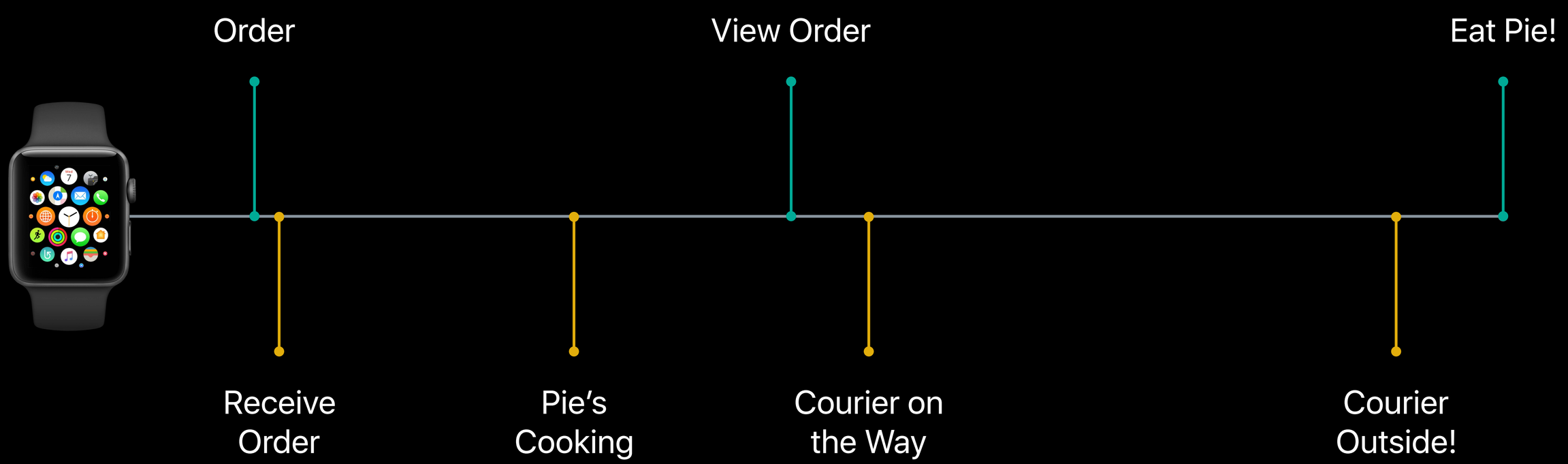
Background App Refresh

Apple Pie Me



Background App Refresh

Apple Pie Me



Apple Pie Me

Apple Pie Me

NSURLSession resumes

Apple Pie Me

NSURLSession resumes

Haptics

Apple Pie Me

NSURLSession resumes

Haptics

Notifications

Apple Pie Me

NSURLSession resumes

Haptics

Notifications

Background app refresh

Apple Pie Me

NSURLSession resumes

Haptics

Notifications

Background app refresh

Schedule NSURLSession ETA requests

Background Modes

Background Modes

Background Modes

Workout

Background Modes

Workout

Audio recording

Background Modes

Workout

Audio recording

Navigation

Audio Recording Background Mode

Audio Recording Background Mode

Your UI

Audio Recording Background Mode

Your UI

Foreground initiated, background running

Audio Recording Background Mode

Your UI

Foreground initiated, background running

Frontmost while recording

Audio Recording Background Mode

Your UI

Foreground initiated, background running

Frontmost while recording

Ability to play haptics

Playback and Recording

Playback and Recording

Playback

- AVAudioPlayer (watchOS 3.1 SDK)

Playback and Recording

Playback

- AVAudioPlayer (watchOS 3.1 SDK)

Recording

- AVAudioInputNode (AVAudioEngine)
- AVAudioRecorder
- AVAudioSession recording permissions

Playback and Recording

Playback

- AVAudioPlayer (watchOS 3.1 SDK)

Recording

- AVAudioInputNode (AVAudioEngine)
- AVAudioRecorder
- AVAudioSession recording permissions

Formats supported

- AAC-LC, AAC-ELD, HE-AAC, HE-AACv2, MP3 (decoding only), Opus

Location Background Mode

Location Background Mode

Foreground initiated, background running

Location Background Mode

Foreground initiated, background running

Frontmost while in session

Location Background Mode

Location Background Mode

startUpdatingLocation

Location Background Mode

startUpdatingLocation

Set allowsBackgroundLocationUpdates

Apple Pie Me

Apple Pie Me

Driver app

Apple Pie Me

Driver app

Play haptics while in session

Summary

Summary

Unified process runtime

Summary

Unified process runtime

Frontmost app state

Summary

Unified process runtime

Frontmost app state

Background app refresh

Summary

Unified process runtime

Frontmost app state

Background app refresh

Background modes

Rethink Your Apps

Rethink Your Apps

Design apps for Frontmost App State

Rethink Your Apps

Design apps for Frontmost App State

Think through the enhanced capabilities

Rethink Your Apps

Design apps for Frontmost App State

Think through the enhanced capabilities

Build two second experiences

More Information

https://developer.apple.com/wwdc17/216

Related Sessions

What's New in watchOS	Hall 2	Wednesday 9:00AM
Planning a Great Apple Watch Experience	Grand Ballroom B	Thursday 5:10PM
What's New in Audio	Grand Ballroom B	Tuesday 1:50PM
Keeping Your Watch App Up to Date		WWDC 2016
Architecting for Performance on watchOS 3		WWDC 2016

Labs

Background Updates and WatchKit Lab	Technology Lab H	Wed 3:10PM–6:00PM
WatchKit Lab	Technology Lab C	Thurs 1:00PM–4:00PM
WatchConnectivity and WatchKit Lab	Technology Lab B	Fri 9:00AM–11:00AM


WWDC17