Keeping Your Watch App Up to Date

Session 218

Eric Lanz watchOS Engineer
Austen Green watchOS Engineer


Tuesday
10:09
21
12:00PM–1:00PM
Pick up Dad
San Francisco In-
ternational Airport


Wed
21
CALENDAR
In 60 minutes
Pick up Dad
Where
San Francisco International Airport


10:09
12:00–1:00PM
Pick up Dad
San Francisco Inter...
68°


Calendar
Tuesday
21
12:00PM–1:00PM
Pick up Dad
San Francisco In-
ternational Airport

Overview

Walkthrough

Scheduling

Best Practices

Case Study

Overview

User model

Overview

User model

Waiting for Coffee

Overview

User model

Waiting for Coffee

Check Weather

Overview

User model

Waiting for Coffee

Find a Lunch Spot

Check Weather

Overview

User model

Waiting for Coffee

Find a Lunch Spot

Check Weather

Receive Notification

Overview

User model

Waiting for Coffee

Find a Lunch Spot

Check Traffic

Check Weather

Receive Notification

Overview

User model

Waiting for Coffee

Find a Lunch Spot

Check Traffic

Check Weather

Receive Notification

Quick Reply

Overview

User model

Waiting for Coffee

Find a Lunch Spot

Check Traffic

Check Weather

Receive Notification

Quick Reply

Overview

Scheduling

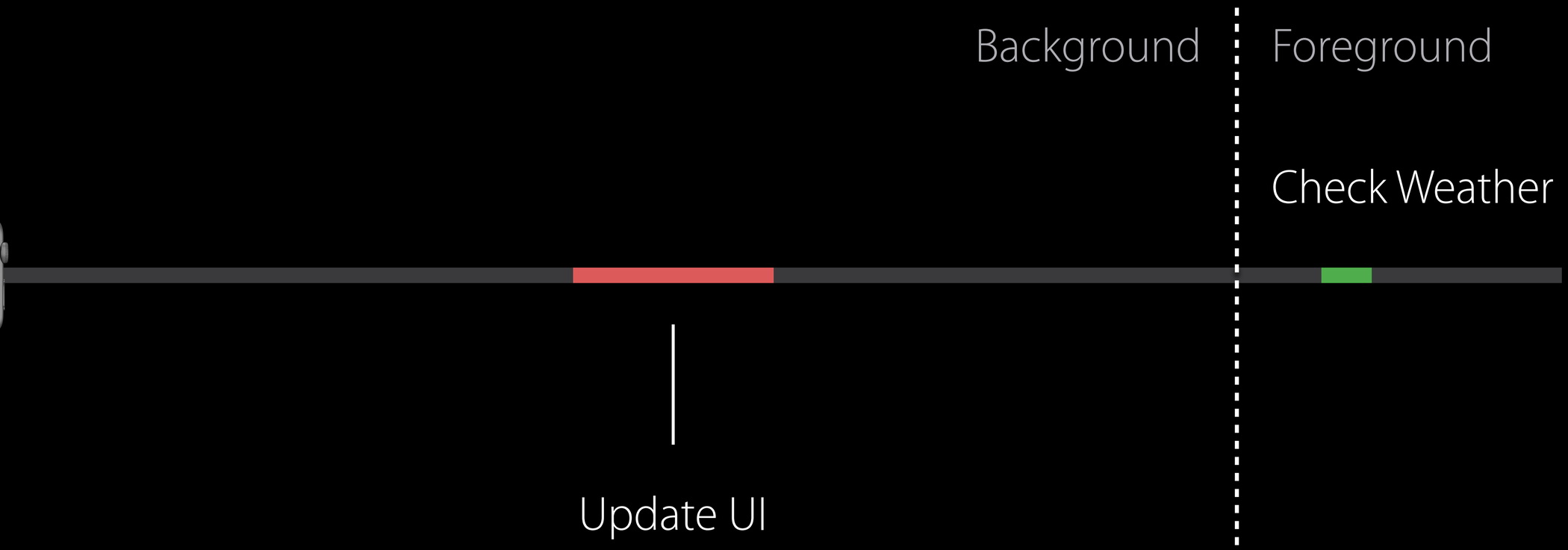
Background

Foreground

Check Weather

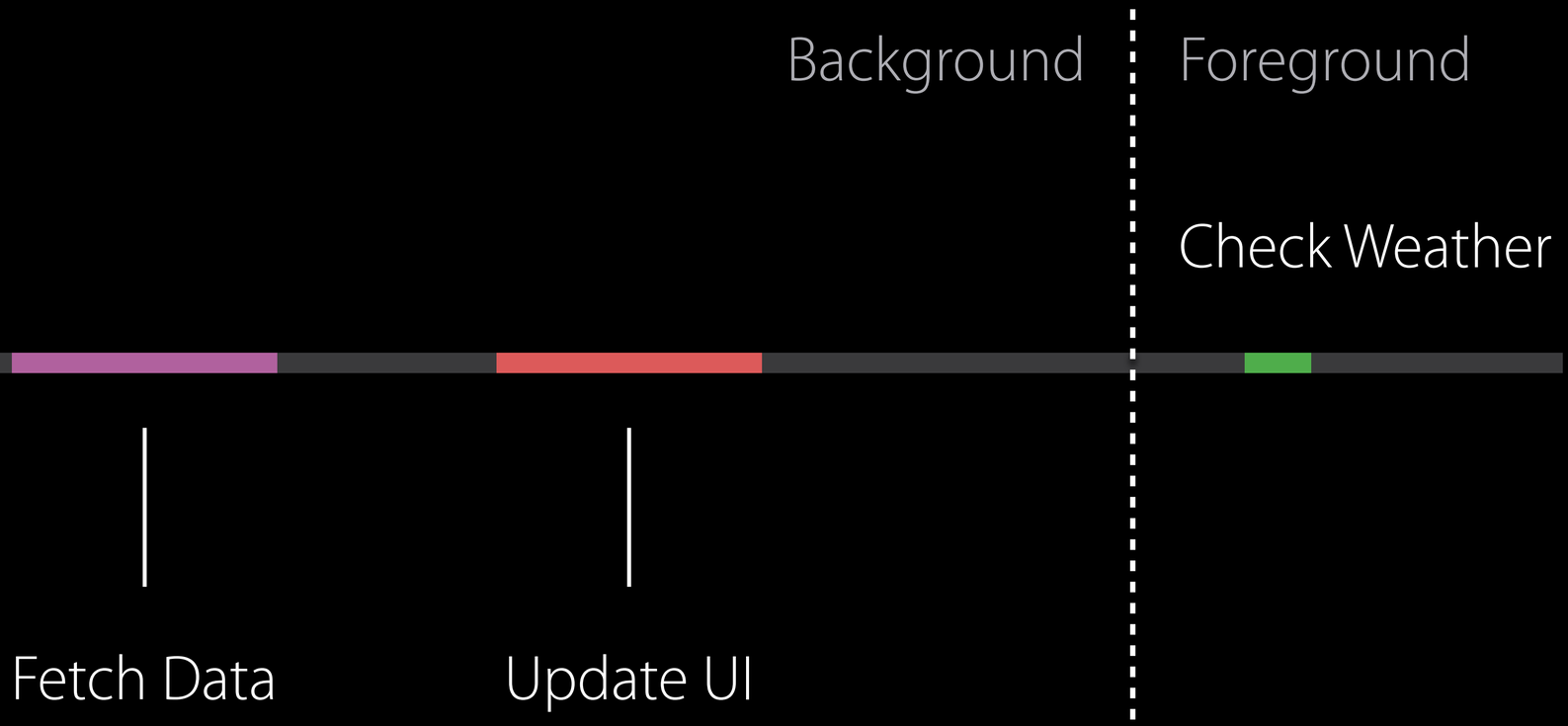
Overview

Scheduling



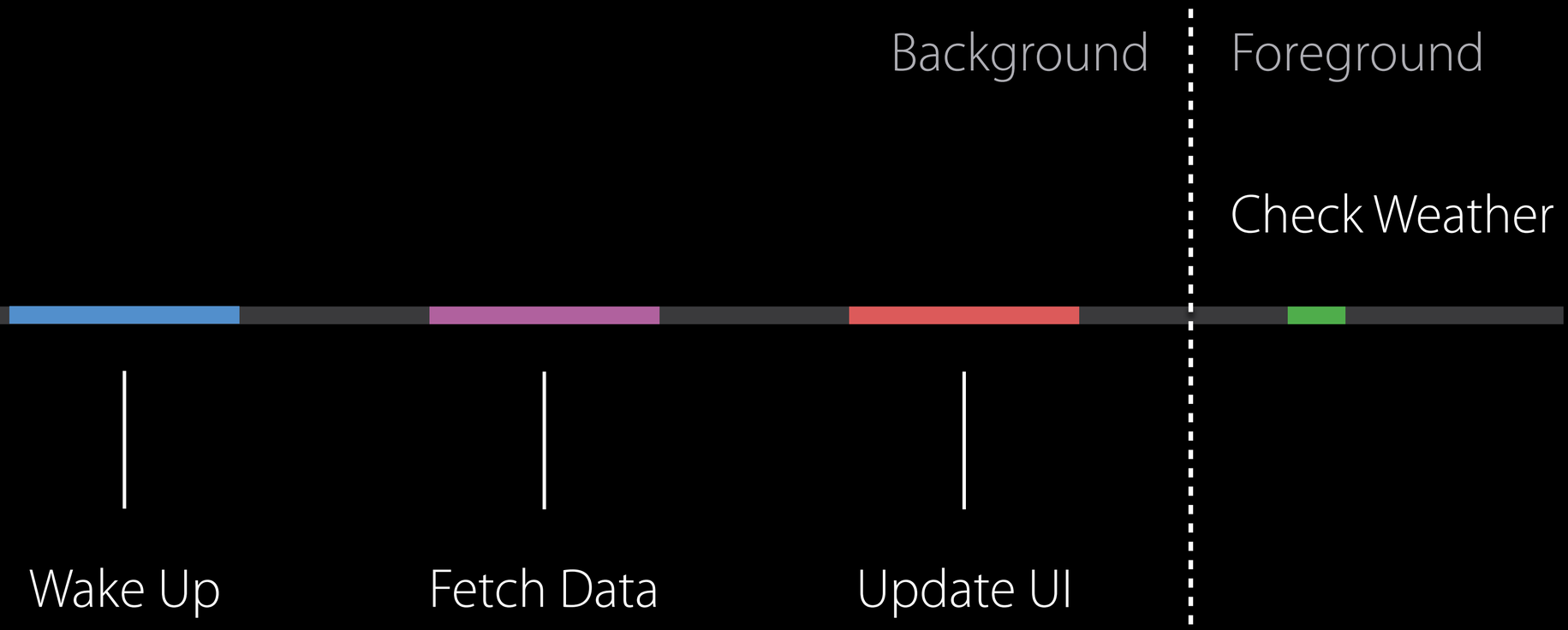
Overview

Scheduling



Overview

Scheduling



Tasks

watchOS

Task

Task

Task

Task

Task

Task

Task

Task

Applications

Suspended

Tasks

watchOS

Task

Task

Task

Task

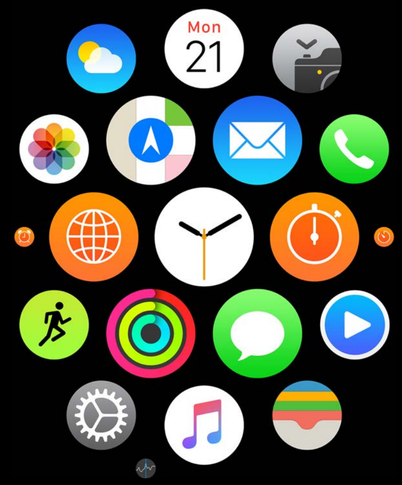
Task

Task

Task

Task

Applications



Active

Tasks

watchOS

Task

Task

Task

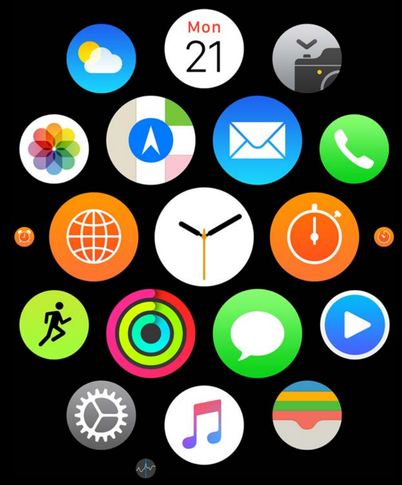
Task

Task

Task

Task

Applications



Active

Tasks

watchOS

Applications

```
func handle(_ backgroundTasks: Set<WKRefreshBackgroundTask>)
```

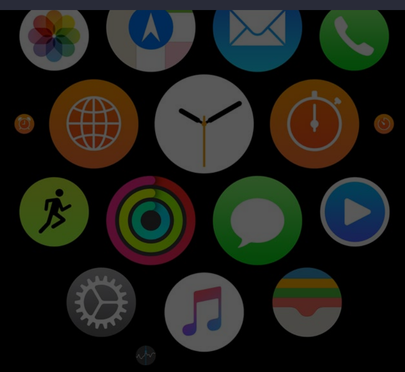
Task

Task

Task

Task

Task



Active

Tasks

watchOS

Task

Task

Task

Task

Task

Task

Task

Applications

Task

Suspended

Tasks

watchOS

Task

Task

Task

Task

Task

Task

Task

Task

Applications

Suspended

Tasks

Summary

`WKApplicationRefreshBackgroundTask`	WKExtension: `scheduleBackgroundRefresh`
`WKURLSessionRefreshBackgroundTask`	Schedule using URLSession
`WKSnapshotRefreshBackgroundTask`	WKExtension: `scheduleSnapshotRefresh`
`WKWatchConnectivityRefreshBackgroundTask`	Schedule using Watch Connectivity

Tasks

Summary

`WKApplicationRefreshBackgroundTask`	WKExtension: `scheduleBackgroundRefresh`
`WKURLSessionRefreshBackgroundTask`	Schedule using URLSession
`WKSnapshotRefreshBackgroundTask`	WKExtension: `scheduleSnapshotRefresh`
`WKWatchConnectivityRefreshBackgroundTask`	Schedule using Watch Connectivity

Tasks

Wake up

Background | Foreground

Check Weather

Wake Up

Application Task

Fetch Data

Update UI

Tasks

Summary

`WKApplicationRefreshBackgroundTask`	WKExtension: `scheduleBackgroundRefresh`
`WKURLSessionRefreshBackgroundTask`	Schedule using URLSession
`WKSnapshotRefreshBackgroundTask`	WKExtension: `scheduleSnapshotRefresh`
`WKWatchConnectivityRefreshBackgroundTask`	Schedule using Watch Connectivity

Tasks

Summary

`WKApplicationRefreshBackgroundTask`

WKExtension:
`scheduleBackgroundRefresh`

`WKURLSessionRefreshBackgroundTask`

Schedule using URLSession

`WKSnapshotRefreshBackgroundTask`

WKExtension:
`scheduleSnapshotRefresh`

`WKWatchConnectivityRefreshBackgroundTask`

Schedule using Watch Connectivity

Tasks

Fetch data

Background | Foreground

Check Weather

Wake Up

Fetch Data

URLSession Task

Update UI

Snapshots


Tuesday
10:09
21
12:00PM–1:00PM
Pick up Dad
San Francisco In-
ternational Airport

Snapshots


Calendar
Tuesday
21
12:00PM–1:00PM
Pick up Dad
San Francisco In-
ternational Airport

Tasks

Summary

`WKApplicationRefreshBackgroundTask`	WKExtension: `scheduleBackgroundRefresh`
`WKURLSessionRefreshBackgroundTask`	Schedule using URLSession
`WKSnapshotRefreshBackgroundTask`	WKExtension: `scheduleSnapshotRefresh`
`WKWatchConnectivityRefreshBackgroundTask`	Schedule using Watch Connectivity

Tasks

Summary

`WKApplicationRefreshBackgroundTask`	WKExtension: `scheduleBackgroundRefresh`
`WKURLSessionRefreshBackgroundTask`	Schedule using URLSession
`WKSnapshotRefreshBackgroundTask`	WKExtension: `scheduleSnapshotRefresh`
`WKWatchConnectivityRefreshBackgroundTask`	Schedule using Watch Connectivity

Tasks

Snapshot

Background | Foreground

Check Weather

Wake Up

Fetch Data

Update UI

Snapshot Task

Snapshots

Meet expectations

Snapshots

Optional default state

Snapshots

More information

Designing Great Apple Watch Experiences	Presidio	Wednesday 1:40PM

Tasks

Summary

`WKApplicationRefreshBackgroundTask`	WKExtension: `scheduleBackgroundRefresh`
`WKURLSessionRefreshBackgroundTask`	Schedule using URLSession
`WKSnapshotRefreshBackgroundTask`	WKExtension: `scheduleSnapshotRefresh`
`WKWatchConnectivityRefreshBackgroundTask`	Schedule using Watch Connectivity

Tasks

Summary

`WKApplicationRefreshBackgroundTask`	WKExtension: `scheduleBackgroundRefresh`
`WKURLSessionRefreshBackgroundTask`	Schedule using URLSession
`WKSnapshotRefreshBackgroundTask`	WKExtension: `scheduleSnapshotRefresh`
`WKWatchConnectivityRefreshBackgroundTask`	Schedule using Watch Connectivity

Tasks

Watch connectivity

Tasks

Watch connectivity

Complication

Context

File

User Info

Tasks

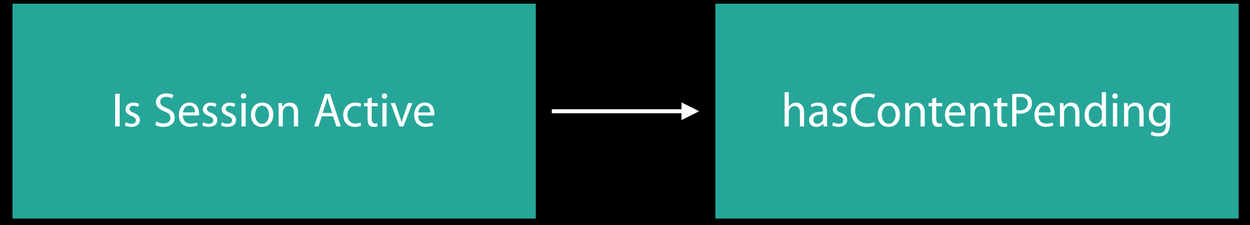
Watch connectivity

Is Session Active

Tasks

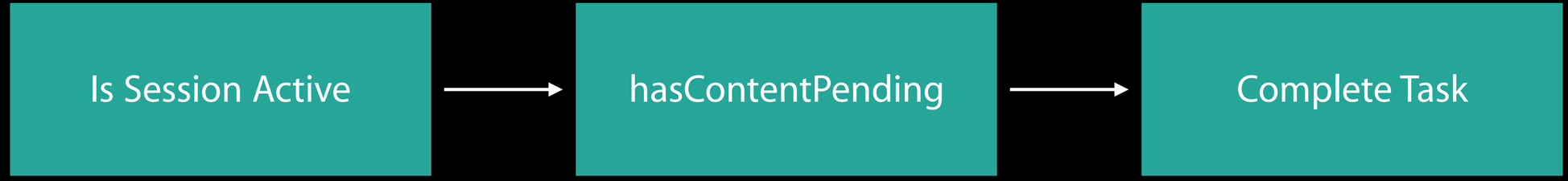
Watch connectivity

NEW



Tasks

Watch connectivity



Schedule
Background
Runtime

Schedule Background Runtime

Receive Tasks

Schedule Background Runtime

Receive Tasks

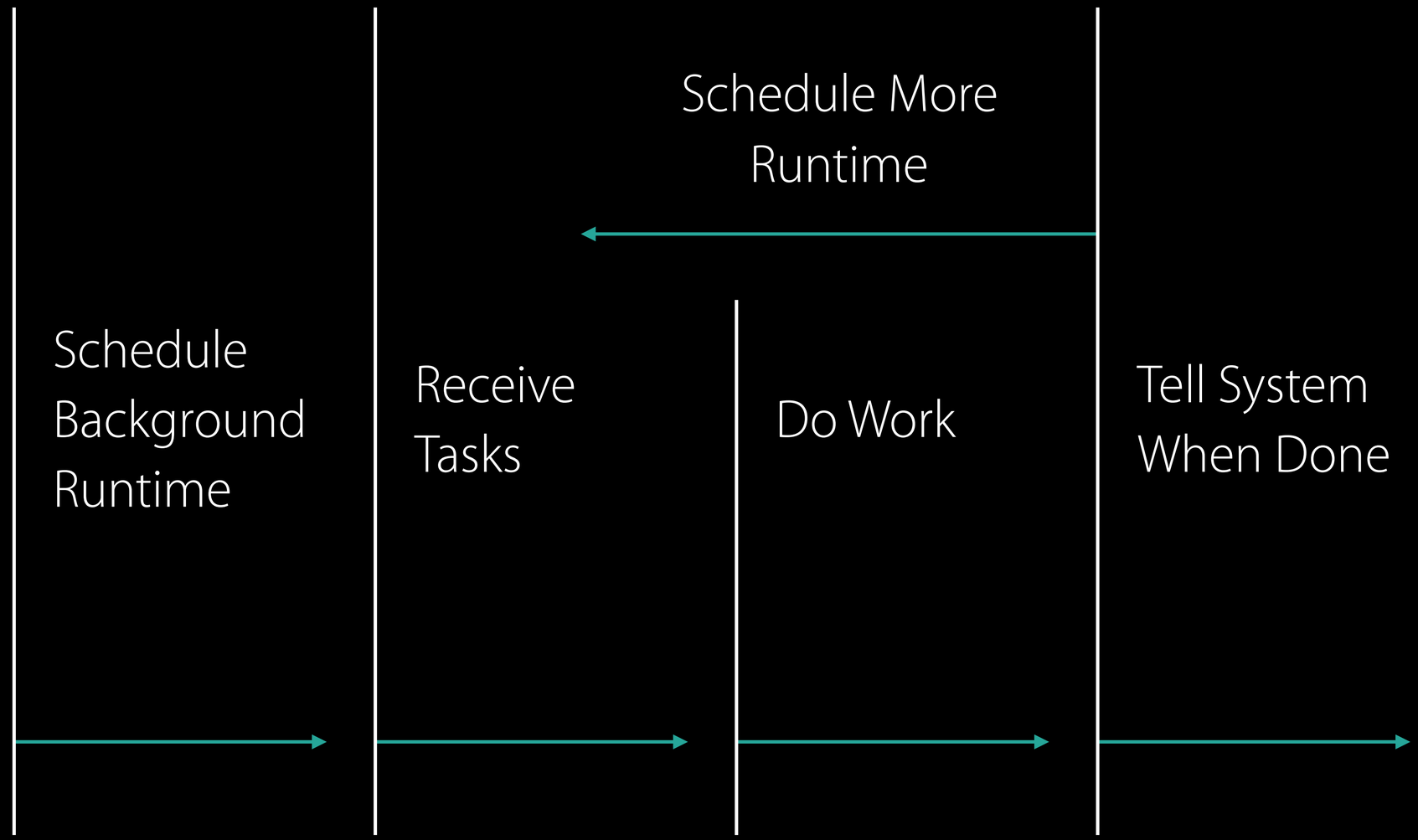
Do Work

Schedule More Runtime

Schedule Background Runtime

Receive Tasks

Do Work


Schedule More Runtime
Schedule Background Runtime
Receive Tasks
Do Work
Tell System When Done

Be a Good Citizen

Be a Good Citizen

User Launches App

3:00 PM

Be a Good Citizen

User Launches App ⟶ Wake Up

3:00 PM

4:00 PM

Be a Good Citizen

User Launches App Wake Up

3:50 PM 4:00 PM

Be a Good Citizen

User Launches App

Wake Up

3:50 PM

4:50 PM

Walkthrough

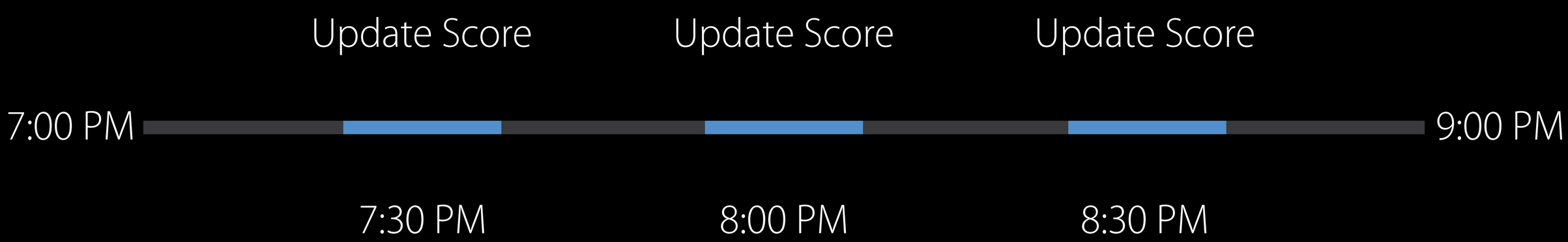
Walkthrough

Football scores

7:00 PM 9:00 PM

Walkthrough

Football scores



Walkthrough

Football scores

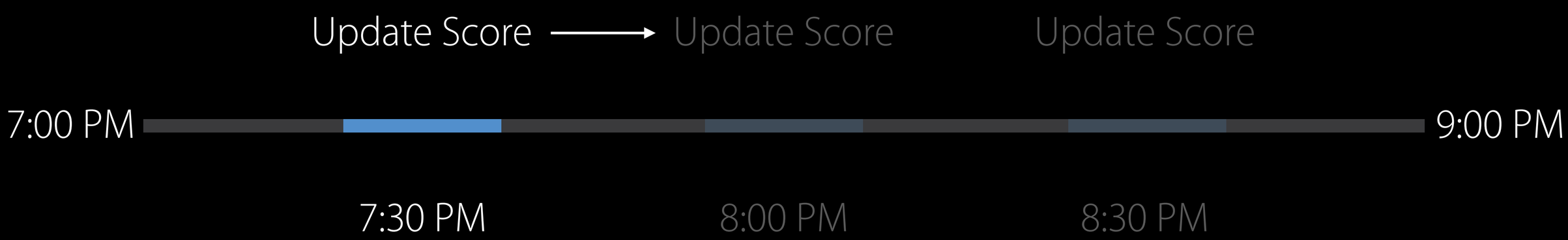
Update Score Update Score Update Score

7:00 PM 9:00 PM

7:30 PM 8:00 PM 8:30 PM

Walkthrough

Football scores



Walkthrough

Football scores

Update Score Update Score → Update Score

7:00 PM 9:00 PM

7:30 PM 8:00 PM 8:30 PM

```
// Scheduling Background Runtime

func myScheduleNextRefreshTask() {
    let fireDate = Date(timeIntervalSinceNow: 30 * 60) // 30 minutes from now

    let userInfo = ["lastActiveDate" : Date(),    // optional last active time
                    "reason" : “scoreUpdate”]     // optional reason

    WKExtension.shared().scheduleBackgroundRefresh(withPreferredDate: fireDate
    , userInfo: userInfo) { (error) in
        if error == nil {
            // successfully scheduled
        }
    }
}
```

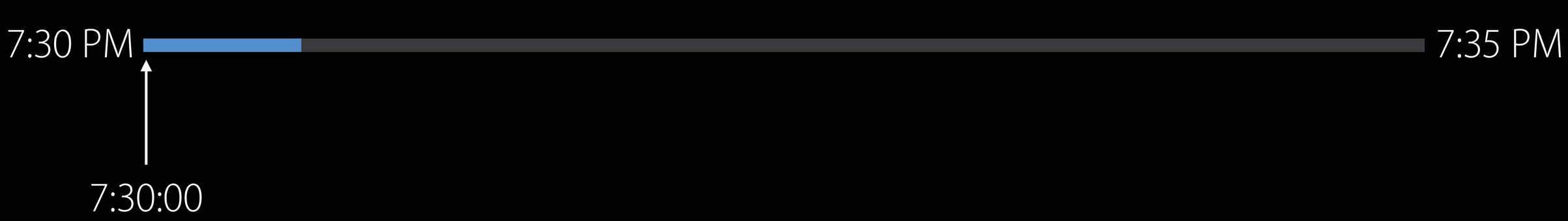
```
// Scheduling Background Runtime

func myScheduleNextRefreshTask() {
    let fireDate = Date(timeIntervalSinceNow: 30 * 60) // 30 minutes from now

    let userInfo = ["lastActiveDate" : Date(),     // optional last active time
                    "reason" : “scoreUpdate”]      // optional reason

    WKExtension.shared().scheduleBackgroundRefresh(withPreferredDate: fireDate
    , userInfo: userInfo) { (error) in
        if error == nil {
            // successfully scheduled
        }
    }
}
```

```
// Scheduling Background Runtime

func myScheduleNextRefreshTask() {
    let fireDate = Date(timeIntervalSinceNow: 30 * 60) // 30 minutes from now

    let userInfo = ["lastActiveDate" : Date(),    // optional last active time
                    "reason" : “scoreUpdate”]     // optional reason

    WKExtension.shared().scheduleBackgroundRefresh(withPreferredDate: fireDate
    , userInfo: userInfo) { (error) in
        if error == nil {
            // successfully scheduled
        }
    }
}
```

Walkthrough

Football scores

Wake Up



Walkthrough

Football scores

```
// Fetching Data in the Background

func myScheduleURLSession() {
    let backgroundConfigObject = URLSessionConfiguration.backgroundSessionConfiguration(
                                    withIdentifier: "com.example.urlsession")
    let backgroundSession = URLSession(configuration: backgroundConfigObject)

    let downloadTask = backgroundSession.downloadTask(
                    with: URL(string: "https://example.com/currentScores.json")!)
    downloadTask.resume()
}
```

```swift
// Fetching Data in the Background

func myScheduleURLSession() {
    let backgroundConfigObject = URLSessionConfiguration.backgroundSessionConfiguration(
                                        withIdentifier: "com.example.urlsession")
    let backgroundSession = URLSession(configuration: backgroundConfigObject)

    let downloadTask = backgroundSession.downloadTask(
                        with: URL(string: "https://example.com/currentScores.json")!)
    downloadTask.resume()
}
```

```
// Fetching Data in the Background

func myScheduleURLSession() {
    let backgroundConfigObject = URLSessionConfiguration.backgroundSessionConfiguration(
                                        withIdentifier: "com.example.urlsession")
    let backgroundSession = URLSession(configuration: backgroundConfigObject)

    let downloadTask = backgroundSession.downloadTask(
                    with: URL(string: "https://example.com/currentScores.json")!)
    downloadTask.resume()
}
```

```swift
// Fetching Data in the Background

func myScheduleURLSession() {
    let backgroundConfigObject = URLSessionConfiguration.backgroundSessionConfiguration(
                                        withIdentifier: "com.example.urlsession")
    let backgroundSession = URLSession(configuration: backgroundConfigObject)

    let downloadTask = backgroundSession.downloadTask(
                        with: URL(string: "https://example.com/currentScores.json")!)
    downloadTask.resume()
}
```

Walkthrough

Football scores

Wake Up

Fetch Data

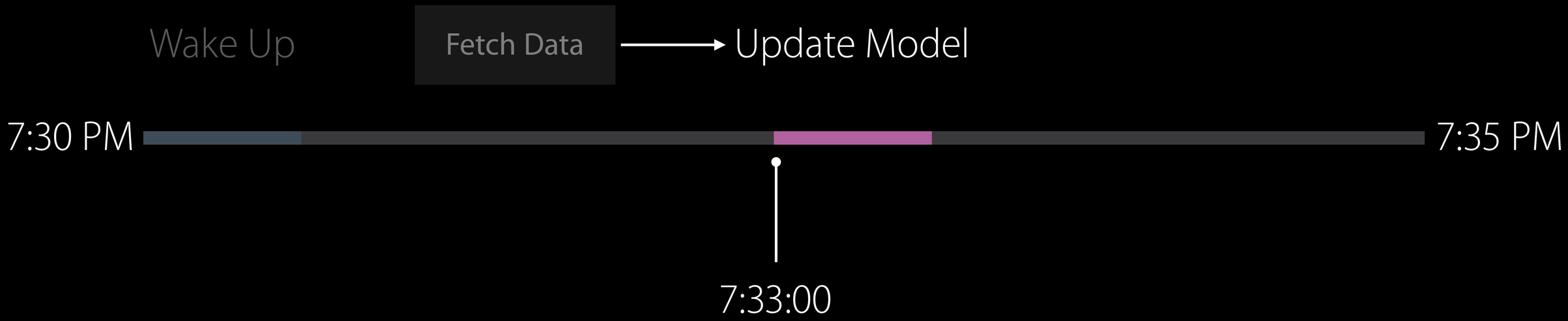
7:30 PM

7:35 PM

7:30:05

Walkthrough

Football scores



```swift
// Handling Background Tasks
// WKExtensionDelegate

func handle(_ backgroundTasks: Set<WKRefreshBackgroundTask>) {
    for task in backgroundTasks {
        if let urlTask = task as? WKURLSessionRefreshBackgroundTask
            let backgroundConfigObject
                URLSessionConfiguration.backgroundSessionConfiguration(
                    withIdentifier: urlTask.sessionIdentifier)
            let backgroundSession = URLSession(configuration: backgroundConfigObject,
                                               delegate: self,
                                          delegateQueue: nil)
            // receive data via URLSessionDownloadDelegate
            pendingBackgroundTasks.append(task)
        } else {
            task.setTaskCompleted()   // make sure to complete all tasks
        }
    }
}
```

```swift
// Handling Background Tasks
// WKExtensionDelegate

func handle(_ backgroundTasks: Set<WKRefreshBackgroundTask>) {
    for task in backgroundTasks {
        if let urlTask = task as? WKURLSessionRefreshBackgroundTask
            let backgroundConfigObject
                URLSessionConfiguration.backgroundSessionConfiguration(
                    withIdentifier: urlTask.sessionIdentifier)
            let backgroundSession = URLSession(configuration: backgroundConfigObject,
                                               delegate: self,
                                          delegateQueue: nil)
            // receive data via URLSessionDownloadDelegate
            pendingBackgroundTasks.append(task)
        } else {
            task.setTaskCompleted()   // make sure to complete all tasks
        }
    }
}
```

```swift
// Handling Background Tasks
// WKExtensionDelegate

func handle(_ backgroundTasks: Set<WKRefreshBackgroundTask>) {
    for task in backgroundTasks {
        if let urlTask = task as? WKURLSessionRefreshBackgroundTask
            let backgroundConfigObject
                URLSessionConfiguration.backgroundSessionConfiguration(
                    withIdentifier: urlTask.sessionIdentifier)
            let backgroundSession = URLSession(configuration: backgroundConfigObject,
                                               delegate: self,
                                          delegateQueue: nil)
            // receive data via URLSessionDownloadDelegate
            pendingBackgroundTasks.append(task)
        } else {
            task.setTaskCompleted()   // make sure to complete all tasks
        }
    }
}
```

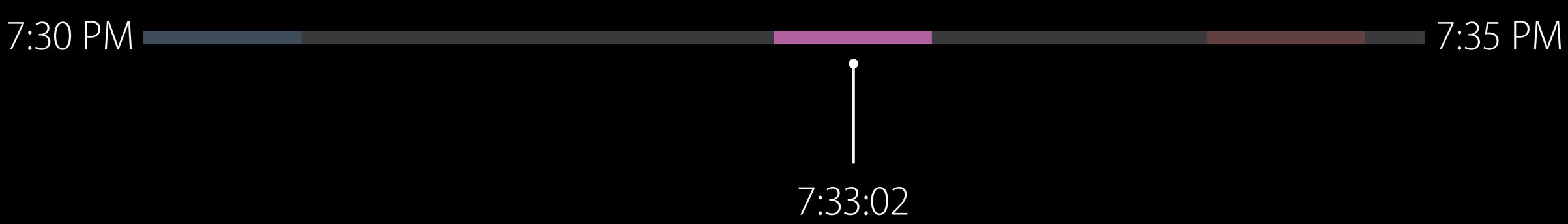
```swift
// Handling Background Tasks
// WKExtensionDelegate

func handle(_ backgroundTasks: Set<WKRefreshBackgroundTask>) {
    for task in backgroundTasks {
        if let urlTask = task as? WKURLSessionRefreshBackgroundTask
            let backgroundConfigObject
                URLSessionConfiguration.backgroundSessionConfiguration(
                    withIdentifier: urlTask.sessionIdentifier)
            let backgroundSession = URLSession(configuration: backgroundConfigObject,
                                               delegate: self,
                                          delegateQueue: nil)
            // receive data via URLSessionDownloadDelegate
            pendingBackgroundTasks.append(task)
        } else {
            task.setTaskCompleted()   // make sure to complete all tasks
        }
    }
}
```

```swift
// Handling Background Tasks
// WKExtensionDelegate

func handle(_ backgroundTasks: Set<WKRefreshBackgroundTask>) {
    for task in backgroundTasks {
        if let urlTask = task as? WKURLSessionRefreshBackgroundTask
            let backgroundConfigObject
                URLSessionConfiguration.backgroundSessionConfiguration(
                    withIdentifier: urlTask.sessionIdentifier)
            let backgroundSession = URLSession(configuration: backgroundConfigObject,
                                               delegate: self,
                                          delegateQueue: nil)
            // receive data via URLSessionDownloadDelegate
            pendingBackgroundTasks.append(task)
        } else {
            task.setTaskCompleted()   // make sure to complete all tasks
        }
    }
}
```

Walkthrough

Football scores

Wake Up | **Fetch Data** | Update Model → Update UI



Walkthrough

Football scores

Wake Up | Fetch Data | Update Model | Update UI

7:30 PM — 7:35 PM

7:30:05

Walkthrough

Football scores

Wake Up | **Fetch Data** | Update Model | Update UI

7:30 PM — 7:35 PM

7:34:00

Walkthrough

Football scores

Wake Up | Fetch Data | Update Model | Update UI

7:30 PM — 7:35 PM

7:34:05

```swift
// Completing Snapshot Task

func myCompleteSnapshotTask(snapTask : WKSnapshotRefreshBackgroundTask) {
    let expireDate = Date(timeIntervalSinceNow: 30 * 60) // 30 minutes
    let userInfo = ["lastActiveDate" : Date()]
    let restoredDefaultState = false

    snapTask.setTaskCompleted(restoredDefaultState: restoredDefaultState,
                      estimatedSnapshotExpiration: expireDate,
                                         userInfo: userInfo)
}
```

```swift
// Completing Snapshot Task

func myCompleteSnapshotTask(snapTask : WKSnapshotRefreshBackgroundTask) {
    let expireDate = Date(timeIntervalSinceNow: 30 * 60) // 30 minutes
    let userInfo = ["lastActiveDate" : Date()]
    let restoredDefaultState = false

    snapTask.setTaskCompleted(restoredDefaultState: restoredDefaultState,
                      estimatedSnapshotExpiration: expireDate,
                                         userInfo: userInfo)
}
```

```swift
// Completing Snapshot Task

func myCompleteSnapshotTask(snapTask : WKSnapshotRefreshBackgroundTask) {
    let expireDate = Date(timeIntervalSinceNow: 30 * 60) // 30 minutes
    let userInfo = ["lastActiveDate" : Date()]
    let restoredDefaultState = false

    snapTask.setTaskCompleted(restoredDefaultState: restoredDefaultState,
                              estimatedSnapshotExpiration: expireDate,
                                                  userInfo: userInfo)
}
```

Walkthrough

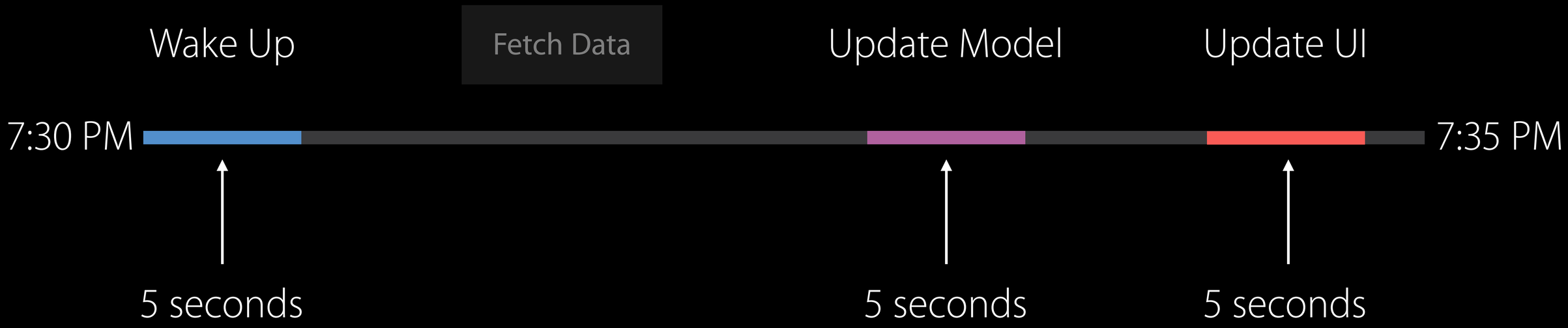
Football scores

Wake Up | Fetch Data | Update Model | Update UI

7:30 PM — 7:35 PM

Walkthrough

Football scores



Scheduling

Austen Green watchOS Engineer

Scheduling

Runtime

Scheduling

Runtime

Always scheduled in foreground

Scheduling

Runtime

Always scheduled in foreground

Usually suspended in background

Scheduling

Runtime

Always scheduled in foreground

Usually suspended in background

Targeted runtime for specific tasks

Scheduling

Runtime limits

Scheduling

Runtime limits

On order of seconds

Scheduling

Runtime limits

On order of seconds

- Time and CPU considered

Scheduling

Runtime limits

On order of seconds

- Time and CPU considered
- Apps killed for exceeding limits
  - CPU - 0xc51bad01
  - Time - 0xc51bad02

Scheduling

Runtime limits

On order of seconds

- Time and CPU considered
- Apps killed for exceeding limits
  - CPU - 0xc51bad01
  - Time - 0xc51bad02

Different allocations by task

```
WKApplicationRefreshBackgroundTask
```

```
WKURLSessionRefreshBackgroundTask
```

Scheduling

Complications

Scheduling

Complications

Multiple updates per hour

Scheduling

Complications

Multiple updates per hour

Request updates through `WKExtension`

Scheduling

Complications

Multiple updates per hour

Request updates through `WKExtension`

50 guaranteed pushes

```
// Scheduling
// Complications
```

NEW

```
func modelChangedOniPhone() {
    let session = WCSession.defaultSession()
    let transfers = session.remainingComplicationUserInfoTransfers
    let userInfo // Complication data

    switch transfers {
    case 0:
        // No transfers left.  Can still try to send
        session.transferCurrentComplicationUserInfo(userInfo)
        break;
    case  1...10:
        // Running low on transfers
        // Conditionally send if it's really important
        // Otherwise conserve the transfer for a more significant change
        break;
    default:
        // Send data immediately
        session.transferCurrentComplicationUserInfo(userInfo)
    }
}
```

```
// Scheduling
// Complications
```

NEW

```
func modelChangedOniPhone() {
    let session = WCSession.defaultSession()
    let transfers = session.remainingComplicationUserInfoTransfers
    let userInfo // Complication data

    switch transfers {
    case 0:
        // No transfers left.  Can still try to send
        session.transferCurrentComplicationUserInfo(userInfo)
        break;
    case  1...10:
        // Running low on transfers
        // Conditionally send if it's really important
        // Otherwise conserve the transfer for a more significant change
        break;
    default:
        // Send data immediately
        session.transferCurrentComplicationUserInfo(userInfo)
    }
}
```

// Scheduling

// Complications

NEW

```
func modelChangedOniPhone() {
    let session = WCSession.defaultSession()
    let transfers = session.remainingComplicationUserInfoTransfers
    let userInfo // Complication data

    switch transfers {
    case 0:
        // No transfers left.  Can still try to send
        session.transferCurrentComplicationUserInfo(userInfo)
        break;
    case  1...10:
        // Running low on transfers
        // Conditionally send if it's really important
        // Otherwise conserve the transfer for a more significant change
        break;
    default:
        // Send data immediately
        session.transferCurrentComplicationUserInfo(userInfo)
    }
}
```

// Scheduling

// Complications

NEW

```
func modelChangedOniPhone() {
    let session = WCSession.defaultSession()
    let transfers = session.remainingComplicationUserInfoTransfers
    let userInfo // Complication data

    switch transfers {
    case 0:
        // No transfers left.  Can still try to send
        session.transferCurrentComplicationUserInfo(userInfo)
        break;
    case  1...10:
        // Running low on transfers
        // Conditionally send if it's really important
        // Otherwise conserve the transfer for a more significant change
        break;
    default:
        // Send data immediately
        session.transferCurrentComplicationUserInfo(userInfo)
    }
}
```

// Scheduling

// Complications

NEW

```
func modelChangedOniPhone() {
    let session = WCSession.defaultSession()
    let transfers = session.remainingComplicationUserInfoTransfers
    let userInfo // Complication data

    switch transfers {
    case 0:
        // No transfers left.  Can still try to send
        session.transferCurrentComplicationUserInfo(userInfo)
        break;
    case  1...10:
        // Running low on transfers
        // Conditionally send if it's really important
        // Otherwise conserve the transfer for a more significant change
        break;
    default:
        // Send data immediately
        session.transferCurrentComplicationUserInfo(userInfo)
    }
}
```

```
// Scheduling
// CLKComplicationDataSource
```

```
optional public func getNextRequestedUpdateDate(handler handler: (NSDate?) -> Void)
```

```
// Scheduling
// WKExtension
```

```
public func scheduleBackgroundRefresh(withPreferredDate preferredFireDate: Date,
userInfo: NSSecureCoding?, scheduledCompletion: (NSError?) -> Swift.Void)
```

```
// Scheduling
// CLKComplicationDataSource


optional public func requestedUpdateDidBegin()
```

```
// Scheduling
// WKExtensionDelegate
```

```
optional public func handle(_ backgroundTasks: Set<WKRefreshBackgroundTask>)
```

Scheduling

Dock apps

Scheduling

Dock apps

Minimum one per hour

Scheduling

Dock apps

Minimum one per hour

Distributed budget

Scheduling

Dock apps

Minimum one per hour

Distributed budget

Scheduling

Dock apps

Minimum one per hour

Distributed budget

Kept in memory

Scheduling

Recent app

Scheduling

Recent app

Most Recently Used app in dock

Scheduling

Recent app

Most Recently Used app in dock

- Treated like a favorite app

Scheduling

Recent app

Most Recently Used app in dock

- Treated like a favorite app

Home screen apps should not expect regular scheduling

Scheduling

System snapshots

Scheduling

System snapshots

Does not count against budget

Scheduling

System snapshots

Does not count against budget

In addition to app-requested snapshot

Scheduling

System snapshots

Does not count against budget

In addition to app-requested snapshot

Triggers

Scheduling

System snapshots

Does not count against budget

In addition to app-requested snapshot

Triggers

- Complication timeline update

Scheduling

System snapshots

Does not count against budget

In addition to app-requested snapshot

Triggers

- Complication timeline update
- Notification interaction

Scheduling

System snapshots

Does not count against budget

In addition to app-requested snapshot

Triggers

- Complication timeline update
- Notification interaction
- Foreground –> Background

Scheduling

System snapshots

Does not count against budget

In addition to app-requested snapshot

Triggers

- Complication timeline update
- Notification interaction
- Foreground –> Background
- “Default state” one hour after backgrounding

Scheduling

System snapshots

Does not count against budget

In addition to app-requested snapshot

Triggers

- Complication timeline update
- Notification interaction
- Foreground –> Background
- “Default state” one hour after backgrounding
- Boot

Best Practices

Best Practices

Best Practices

Schedule as often as needed

Best Practices

Schedule as often as needed

Do not feel obligated to do work

- Finish ASAP
- Defer work

Best Practices

Schedule as often as needed

Do not feel obligated to do work

- Finish ASAP
- Defer work

Consider all runtime opportunities

- Dock and foreground activations
- Notifications
- Complication updates
- Background refresh

Best Practices

WKApplicationRefreshBackgroundTask

Best Practices

WKApplicationRefreshBackgroundTask

Use `scheduleBackgroundRefresh` for general-purpose runtime

Best Practices

WKApplicationRefreshBackgroundTask

Use `scheduleBackgroundRefresh` for general-purpose runtime

- Polling

Best Practices

WKApplicationRefreshBackgroundTask

Use `scheduleBackgroundRefresh` for general-purpose runtime

- Polling
- Scheduling future NSURLSessions

Best Practices

WKApplicationRefreshBackgroundTask

Use `scheduleBackgroundRefresh` for general-purpose runtime

- Polling
- Scheduling future NSURLSessions
- Known time transitions

Best Practices

WKApplicationRefreshBackgroundTask

Use `scheduleBackgroundRefresh` for general-purpose runtime

- Polling
- Scheduling future NSURLSessions
- Known time transitions
- Triggering complication updates

Best Practices

Snapshots

Best Practices

Snapshots

Invalidate snapshots appropriately

Best Practices

Snapshots

Invalidate snapshots appropriately

"Significant content change"

Best Practices

Snapshots

Invalidate snapshots appropriately

“Significant content change”

Avoid high-frequency invalidation

Best Practices

Data flow

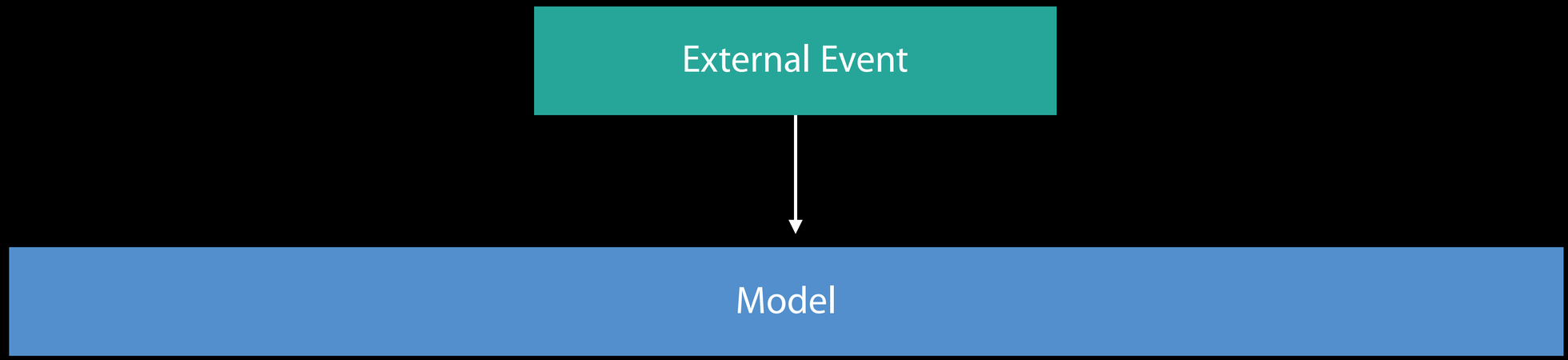
Best Practices

Data flow

External Event

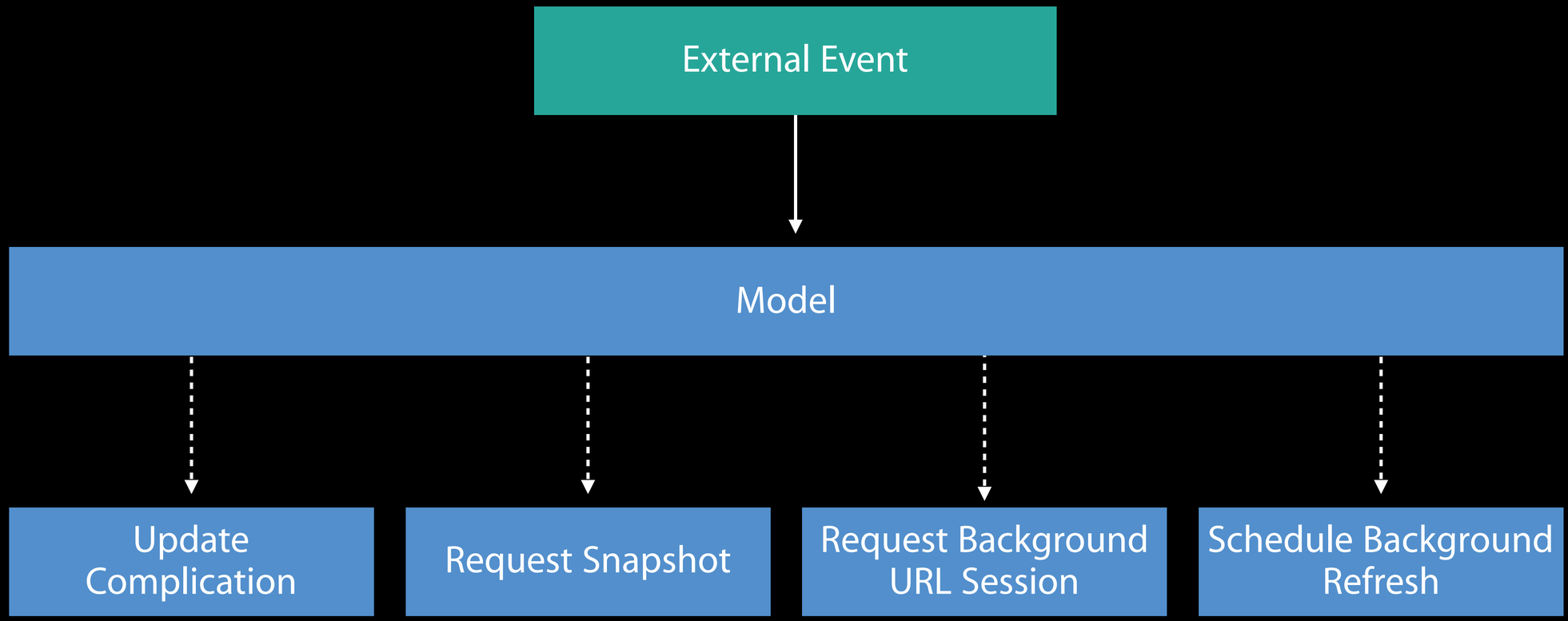
Best Practices

Data flow



Best Practices

Data flow



Best Practices

App lifecycle

Best Practices

App lifecycle

Finish background tasks ASAP on foreground activation

Best Practices

App lifecycle

Finish background tasks ASAP on foreground activation

Finish foreground work when entering background

Best Practices

App lifecycle

Finish background tasks ASAP on foreground activation

Finish foreground work when entering background

`NSProcessInfo.performExpiringActivity`

Best Practices

App lifecycle

Finish background tasks ASAP on foreground activation

Finish foreground work when entering background

`NSProcessInfo.performExpiringActivity`

Best Practices

App lifecycle

Finish background tasks ASAP on foreground activation

Finish foreground work when entering background

`NSProcessInfo.performExpiringActivity`

Data protection

Best Practices

Testing

Best Practices

Testing

Simulator for iterative development

Best Practices

Testing

Simulator for iterative development

Keep the device on charger

Best Practices

Testing

Simulator for iterative development

Keep the device on charger

Test the launch path

Best Practices

Testing

Simulator for iterative development

Keep the device on charger

Test the launch path

Verify tasks are being completed

Best Practices

Testing

Simulator for iterative development

Keep the device on charger

Test the launch path

Verify tasks are being completed

Live on it

Best Practices

Testing

Simulator for iterative development

Keep the device on charger

Test the launch path

Verify tasks are being completed

Live on it

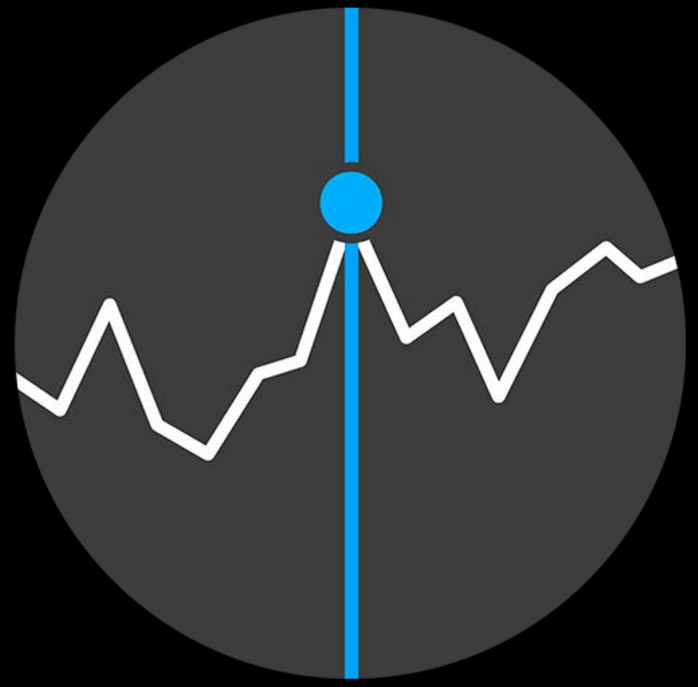
- Vary number of apps in the dock

Case Study

Stocks

Case Study

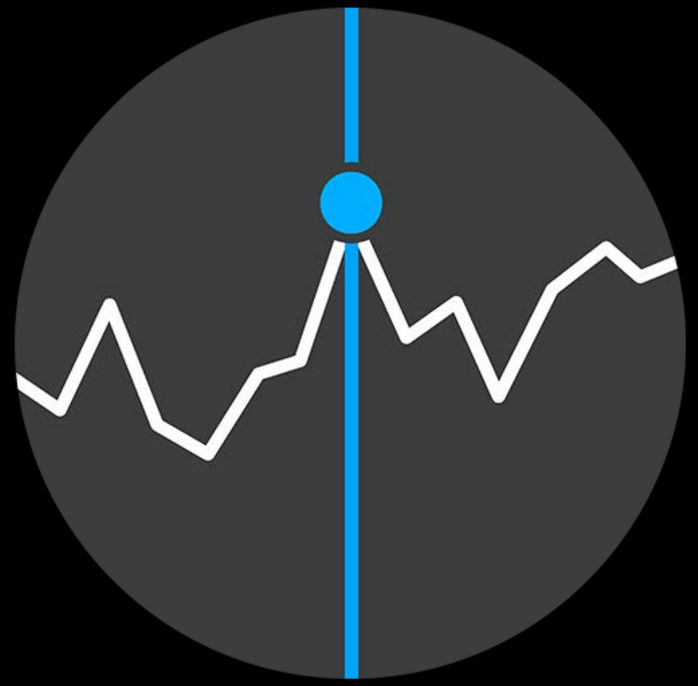
Stocks



Case Study

Stocks

Characteristics

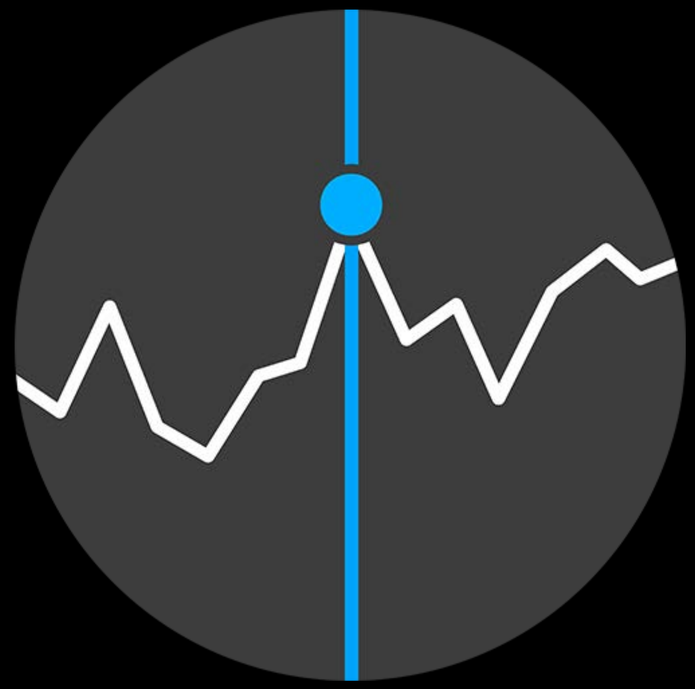


Case Study

Stocks

Characteristics

- `NSURLSession` to retrieve server data

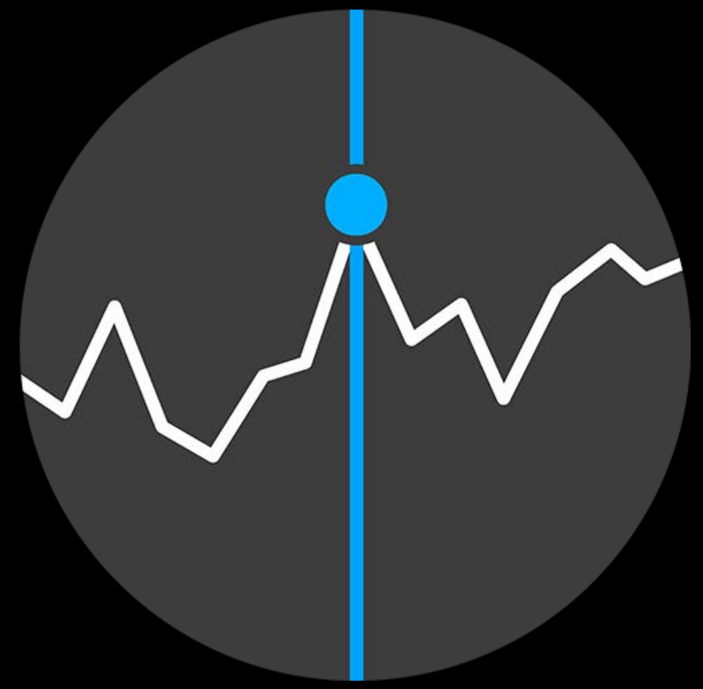


Case Study

Stocks

Characteristics

- `NSURLSession` to retrieve server data
- Has a complication

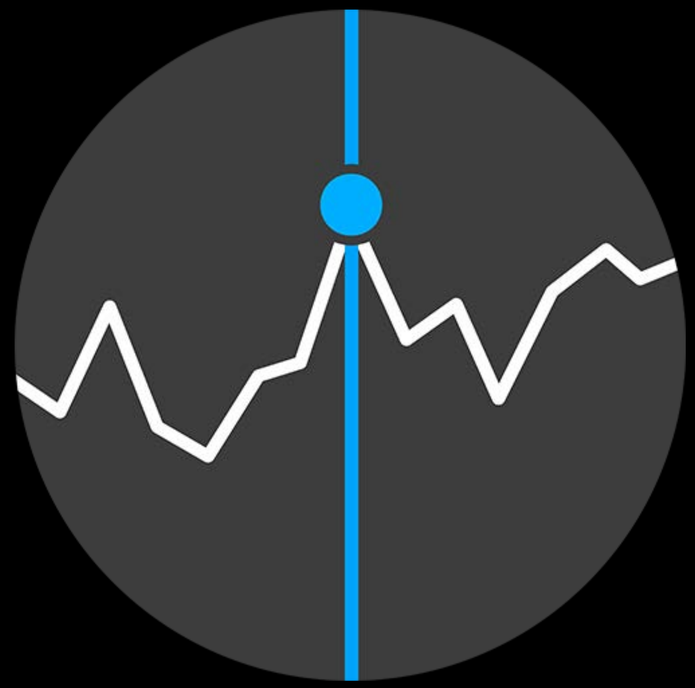


Case Study

Stocks

Characteristics

- `NSURLSession` to retrieve server data
- Has a complication
- Periodic cadence for part of the day

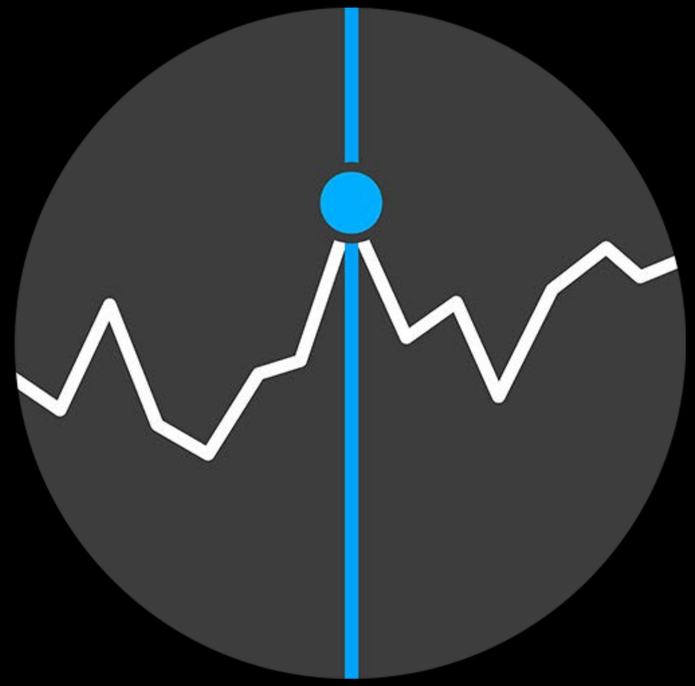


Case Study

Stocks

Characteristics

- `NSURLSession` to retrieve server data
- Has a complication
- Periodic cadence for part of the day
- No updates while markets are closed



Case Study

Boot

Case Study

Boot

- Load last data for snapshot

Case Study

Boot

- Load last data for snapshot
- Schedule a background `NSURLSession`

Case Study

Boot

- Load last data for snapshot
- Schedule a background `NSURLSession`
- Use `NSURLSessionDownloadTask`

Case Study

Boot

- Load last data for snapshot
- Schedule a background `NSURLSession`
- Use `NSURLSessionDownloadTask`
- `NSURLSessionDataTask` will fail for background session on app suspension

Case Study

`NSURLSession`

Case Study

`NSURLSession`

- Update model

Case Study

`NSURLSession`

- Update model
  - Trigger a complication update

Case Study

`NSURLSession`

- Update model
  - Trigger a complication update
  - Request a new snapshot for now

Case Study

`NSURLSession`

- Update model
  - Trigger a complication update
- Request a new snapshot for now
- Request a background refresh for next expected model update time

Case Study

Background refresh

Case Study

Background refresh

- Schedule the next `NSURLSession` download

Case Study

Stocks activated from the dock

Case Study

Stocks activated from the dock

- Use `NSURLSession` to update model

Case Study

Stocks activated from the dock

- Use `NSURLSession` to update model
  - Request complication update

Case Study

Stocks activated from the dock

- Use `NSURLSession` to update model
  - Request complication update
  - Request new snapshot

Case Study

Stocks activated from the dock

- Use `NSURLSession` to update model
  - Request complication update
  - Request new snapshot
  - Schedule background refresh for a later time

Case Study

Last update after market close

Case Study

Last update after market close

- Data has stopped changing for the day

Case Study

Last update after market close

- Data has stopped changing for the day
  - Complete update as normal

Case Study

Last update after market close

- Data has stopped changing for the day
  - Complete update as normal
  - Schedule next refresh for market open

Summary

Summary

Complete your tasks

Summary

Complete your tasks

Use runtime efficiently

Summary

Complete your tasks

Use runtime efficiently

Tell the system when data changes

Summary

Complete your tasks

Use runtime efficiently

Tell the system when data changes

Consider adoption strategies on case-by-case basis

More Information

https://developer.apple.com/wwdc16/218

Related Sessions

What's New in watchOS 3	Presidio	Tuesday 5:00PM
Quick Interaction Techniques for watchOS	Presidio	Wednesday 11:00AM
Designing Great Apple Watch Experiences	Presidio	Wednesday 1:40PM
Architecting for Performance on watchOS 3	Mission	Thursday 3:00PM

Labs

WatchKit and Background Tasks Lab	Frameworks Lab C Thursday 10:30AM
WatchKit and WatchConnectivity Lab	Frameworks Lab B Friday 2:00PM

WWDC16