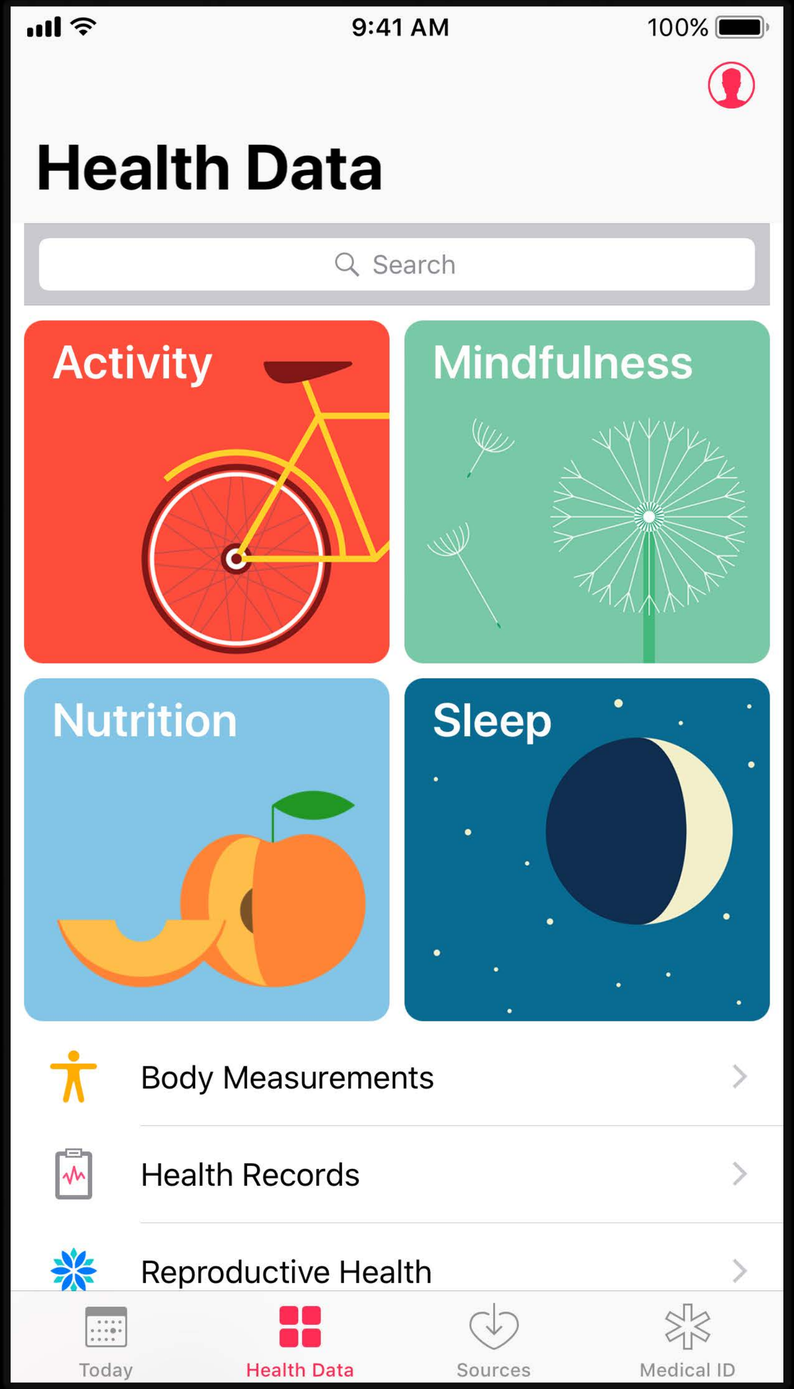
What's New in Health

Session 221

Alexa VanHattum, iOS Software Engineer
Michael Ozeryansky, iOS Software Engineer


3:45
8'11"
28:13


28-01 09:07
mmol/ L


9:41 AM
100%
Health Data
Search
Activity
Mindfulness
Nutrition
Sleep
Body Measurements
Health Records
Reproductive Health
Today
Health Data
Sources
Medical ID

New HealthKit types

Workout API updates

Sync identifiers

Supporting diabetes management

New HealthKit types

Workout API updates

Sync identifiers

Supporting diabetes management

New HealthKit types

Workout API updates

Sync identifiers

Supporting diabetes management

New HealthKit types

Workout API updates

Sync identifiers

Supporting diabetes management

New HealthKit types

Workout API updates

Sync identifiers

Supporting diabetes management

New HealthKit Types

Sample Types

NEW

Sample Types

NEW

Workout route

- `HKWorkoutRouteTypeIdentifier`

Sample Types

NEW

Workout route

- `HKWorkoutRouteTypeIdentifier`

Waist circumference

- `HKQuantityTypeIdentifierWaistCircumference`

Sample Types

NEW

Workout route

- `HKWorkoutRouteTypeIdentifier`

Waist circumference

- `HKQuantityTypeIdentifierWaistCircumference`

VO_2 max

- `HKQuantityTypeIdentifierVO2Max`

Sample Types

NEW

Workout route

- `HKWorkoutRouteTypeIdentifier`

Waist circumference

- `HKQuantityTypeIdentifierWaistCircumference`

VO_2 max

- `HKQuantityTypeIdentifierVO2Max`

Insulin delivery

- `HKQuantityTypeIdentifierInsulinDelivery`

Workout Activity Types

NEW

Workout Activity Types

NEW

Tai chi

- `HKWorkoutRouteType.Taichi`

Workout Activity Types

NEW

Tai chi

- `HKWorkoutRouteType.Taichi`

Mixed cardio

- `HKWorkoutActivityType.MixedCardio`

Workout Activity Types

NEW

Tai chi

- `HKWorkoutRouteType.Taichi`

Mixed cardio

- `HKWorkoutActivityType.MixedCardio`

Hand cycling

- `HKWorkoutActivityType.HandCycling`

Workout API Updates

Swimming, segments, and pause/resume


18:20.35
18:37
36 LAPS
900YD

Swimming

Tracking with Apple Watch

Support for pool and open water

Swimming

Tracking with Apple Watch

Support for pool and open water

Automatic swimming metrics

Swimming

Tracking with Apple Watch

Support for pool and open water

Automatic swimming metrics

- Swimming distance

Swimming

Tracking with Apple Watch

Support for pool and open water

Automatic swimming metrics

- Swimming distance
- Stroke count

Swimming

Tracking with Apple Watch

Support for pool and open water

Automatic swimming metrics

- Swimming distance
- Stroke count
- Individual lap detection

Swimming

Tracking with Apple Watch

Support for pool and open water

Automatic swimming metrics

- Swimming distance
- Stroke count
- Individual lap detection
- Per-lap stroke style detection

Swimming

Tracking with Apple Watch

NEW

Support for pool and open water

Automatic swimming metrics

- Swimming distance
- Stroke count
- Individual lap detection
- Per-lap stroke style detection
- Set detection

Swimming

Tracking with Apple Watch

NEW

Support for pool and open water

Automatic swimming metrics

- Swimming distance
- Stroke count
- Individual lap detection
- Per-lap stroke style detection
- Set detection

Apps can enable water lock

Swimming

Metadata keys and values

```
public let HKMetadataKeySwimmingLocationType: String
```

Swimming

Metadata keys and values

```
public let HKMetadataKeySwimmingLocationType: String
public enum HKWorkoutSwimmingLocationType : Int {
    case Unknown
    case Pool
    case OpenWater
}
```

Swimming

Metadata keys and values

```
public let HKMetadataKeySwimmingStrokeStyle: String
```

Swimming

Metadata keys and values

```
public let HKMetadataKeySwimmingStrokeStyle: String
public enum HKSwimmingStrokeStyle : Int {
    case Unknown
    case Mixed
    case Freestyle
    case Backstroke
    case Breaststroke
    case Butterfly
}
```

Swimming

Workout configuration

```
let workoutConfiguration = HKWorkoutConfiguration()
```

Swimming

Workout configuration

```
let workoutConfiguration = HKWorkoutConfiguration()

workoutConfiguration.activityType = HKWorkoutActivityType.swimming
```

Swimming

Workout configuration

```
let workoutConfiguration = HKWorkoutConfiguration()

workoutConfiguration.activityType = HKWorkoutActivityType.swimming
workoutConfiguration.swimmingLocationType = HKWorkoutSwimmingLocationType.pool
```

Swimming

Workout configuration

```
let workoutConfiguration = HKWorkoutConfiguration()

workoutConfiguration.activityType = HKWorkoutActivityType.swimming
workoutConfiguration.swimmingLocationType = HKWorkoutSwimmingLocationType.pool
workoutConfiguration.lapLength = HKQuantity(unit: .yard(), doubleValue: 25)
```

Swimming

Workout configuration

```
let workoutConfiguration = HKWorkoutConfiguration()

workoutConfiguration.activityType = HKWorkoutActivityType.swimming
workoutConfiguration.swimmingLocationType = HKWorkoutSwimmingLocationType.pool
workoutConfiguration.lapLength = HKQuantity(unit: .yard(), doubleValue: 25)

do {
    let workoutSession = try HKWorkoutSession(configuration: workoutConfiguration)



} catch let error {
    // Handle error...
}
```

Swimming

Workout configuration

```
let workoutConfiguration = HKWorkoutConfiguration()

workoutConfiguration.activityType = HKWorkoutActivityType.swimming
workoutConfiguration.swimmingLocationType = HKWorkoutSwimmingLocationType.pool
workoutConfiguration.lapLength = HKQuantity(unit: .yard(), doubleValue: 25)

do {
    let workoutSession = try HKWorkoutSession(configuration: workoutConfiguration)
    workoutSession.delegate = self



} catch let error {
    // Handle error...
}
```

Swimming

Workout configuration

```
let workoutConfiguration = HKWorkoutConfiguration()

workoutConfiguration.activityType = HKWorkoutActivityType.swimming
workoutConfiguration.swimmingLocationType = HKWorkoutSwimmingLocationType.pool
workoutConfiguration.lapLength = HKQuantity(unit: .yard(), doubleValue: 25)

do {
    let workoutSession = try HKWorkoutSession(configuration: workoutConfiguration)
    workoutSession.delegate = self
    healthStore.start(workoutSession)
    // ...
} catch let error {
    // Handle error...
}
```

Swimming

Enable water lock

```
func workoutSession(_ workoutSession: HKWorkoutSession,
                    didChangeTo toState: HKWorkoutSessionState,
                    from fromState: HKWorkoutSessionState,
                    date: Date) {









}
```

Swimming

Enable water lock

```
func workoutSession(_ workoutSession: HKWorkoutSession,
                    didChangeTo toState: HKWorkoutSessionState,
                    from fromState: HKWorkoutSessionState,
                    date: Date) {

    switch (fromState, toState) {




    }
}
```

Swimming

Enable water lock

```
func workoutSession(_ workoutSession: HKWorkoutSession,
                    didChangeTo toState: HKWorkoutSessionState,
                    from fromState: HKWorkoutSessionState,
                    date: Date) {

    switch (fromState, toState) {
    case (.notStarted, .running):




    }
}
```

Swimming

Enable water lock

```
func workoutSession(_ workoutSession: HKWorkoutSession,
                    didChangeTo toState: HKWorkoutSessionState,
                    from fromState: HKWorkoutSessionState,
                    date: Date) {

    switch (fromState, toState) {
    case (.notStarted, .running):
        let wkExtension = WKExtension.shared()
        wkExtension.enableWaterLock()
    // ...
    }
}
```

HKWorkoutEvent

Highlight a specific time in the workout

HKWorkoutEvent

Highlight a specific time in the workout

Used for pausing, resuming, laps, and markers

HKWorkoutEvent

Highlight a specific time in the workout

Used for pausing, resuming, laps, and markers

Created by HealthKit or your app

HKWorkoutEvent

Highlight a specific time in the workout

Used for pausing, resuming, laps, and markers

Created by HealthKit or your app

Save a list on HKWorkout

HKWorkoutEvent

Highlight a specific time in the workout

Used for pausing, resuming, laps, and markers

Created by HealthKit or your app

Save a list on HKWorkout

Affect the workout’s duration

Swimming

Observing lap events

```
// In your workout session's delegate
func workoutSession(_ workoutSession: HKWorkoutSession, didGenerate event: HKWorkoutEvent) {









}
```

Swimming

Observing lap events

```
// In your workout session's delegate
func workoutSession(_ workoutSession: HKWorkoutSession, didGenerate event: HKWorkoutEvent) {
    switch event.type {









    }
}
```

Swimming

Observing lap events

```
// In your workout session's delegate
func workoutSession(_ workoutSession: HKWorkoutSession, didGenerate event: HKWorkoutEvent) {
    switch event.type {
    case .lap:









    }
}
```

Swimming

Observing lap events

```
// In your workout session's delegate
func workoutSession(_ workoutSession: HKWorkoutSession, didGenerate event: HKWorkoutEvent) {
    switch event.type {
    case .lap:
        lapCount += 1









    }
}
```

Swimming

Observing lap events

```
// In your workout session's delegate
func workoutSession(_ workoutSession: HKWorkoutSession, didGenerate event: HKWorkoutEvent) {
    switch event.type {
    case .lap:
        lapCount += 1
        if let strokeStyle = event.metadata?[HKMetadataKeySwimmingStrokeStyle] {
            self.displayCurrentStrokeStyle(strokeStyle)
        }
    // ...
    }
}
```

New Workout Events

NEW

New Workout Events

NEW

```
// HKWorkout.h

public enum HKWorkoutEventType : Int {
    case pause
    case resume
    case lap
    case marker
    case motionPaused
    case motionResumed



}
```

New Workout Events

NEW

```
// HKWorkout.h

public enum HKWorkoutEventType : Int {
    case pause
    case resume
    case lap
    case marker
    case motionPaused
    case motionResumed

    case segment
    case pauseOrResumeRequest
}
```

New Workout Events

NEW

```
// HKWorkout.h

public enum HKWorkoutEventType : Int {
    case pause
    case resume
    case lap
    case marker
    case motionPaused
    case motionResumed

    case segment
    case pauseOrResumeRequest
}
```

HKWorkoutEvent.segment

NEW

```
open class HKWorkoutEvent : NSObject, NSSecureCoding, NSCopying {

    open var date: Date { get }


    public convenience init(type: HKWorkoutEventType, date: Date, metadata: [String : Any])


}
```

HKWorkoutEvent.segment

NEW

On HKWorkoutEvent: date → dateInterval

```
open class HKWorkoutEvent : NSObject, NSSecureCoding, NSCopying {

    open var date: Date { get }
    open var dateInterval: DateInterval { get }

    public convenience init(type: HKWorkoutEventType, date: Date, metadata: [String : Any])
    public convenience init(type: HKWorkoutEventType, dateInterval: DateInterval, metadata:
        [String : Any]?)
}
```

Start workout

.type

date interval

{metadata}

Start workout

.type

date interval

{metadata}

Start workout

.type

date interval

{metadata}

Start workout

.type

date interval

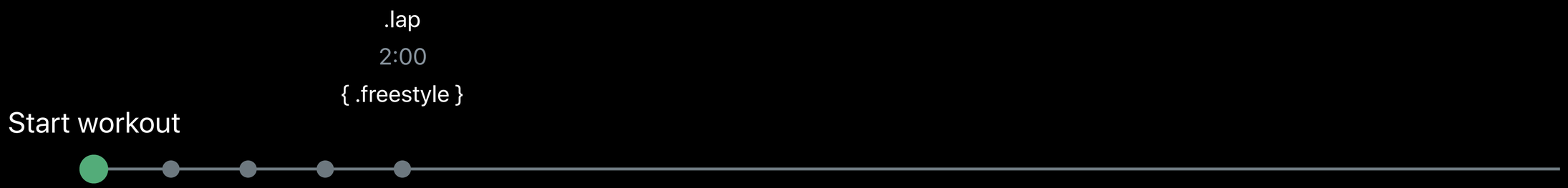
{metadata}

Start workout

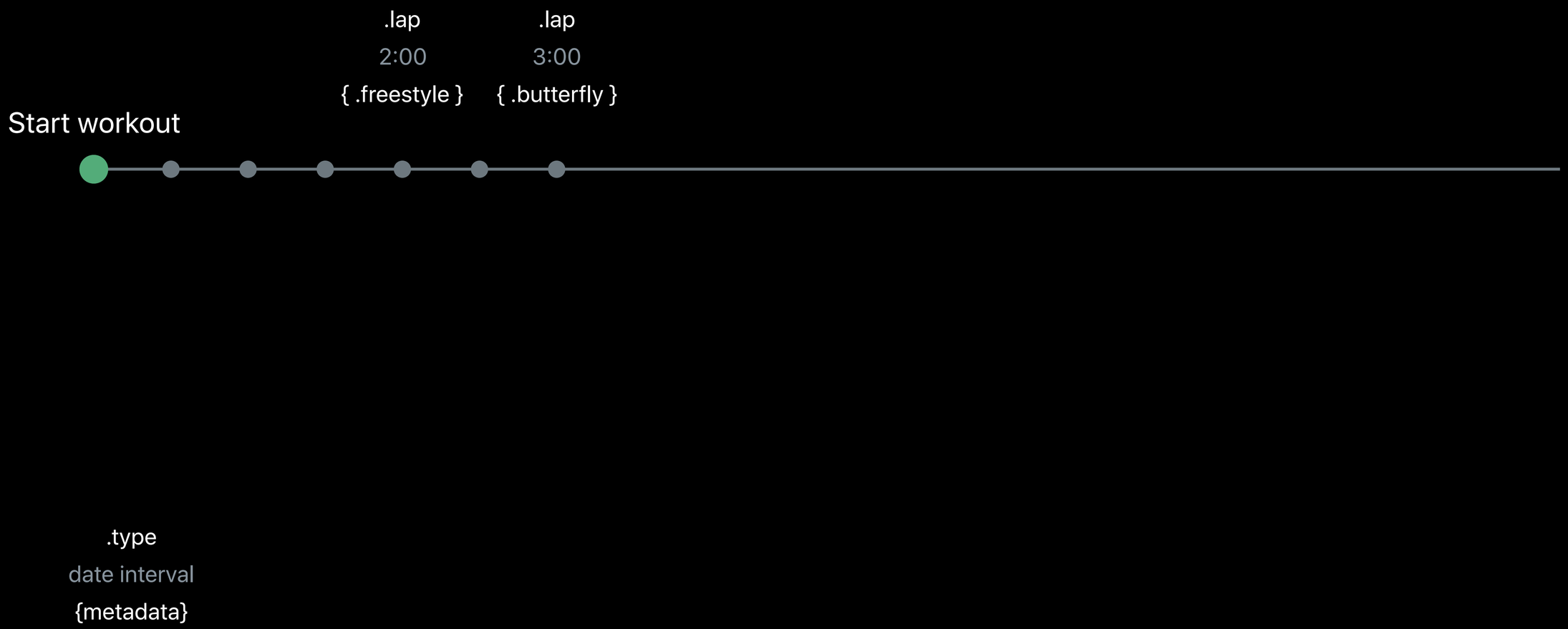
.type

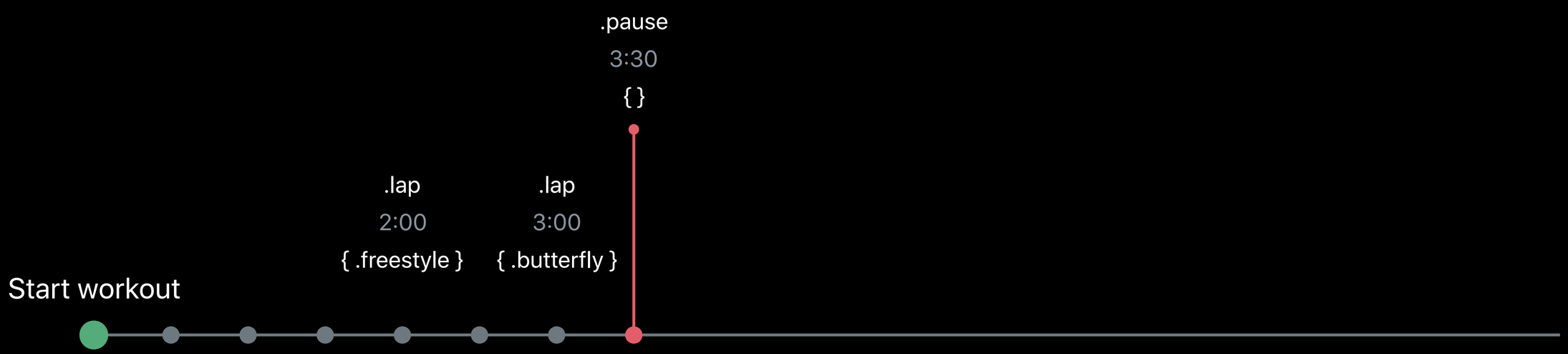
date interval

{metadata}

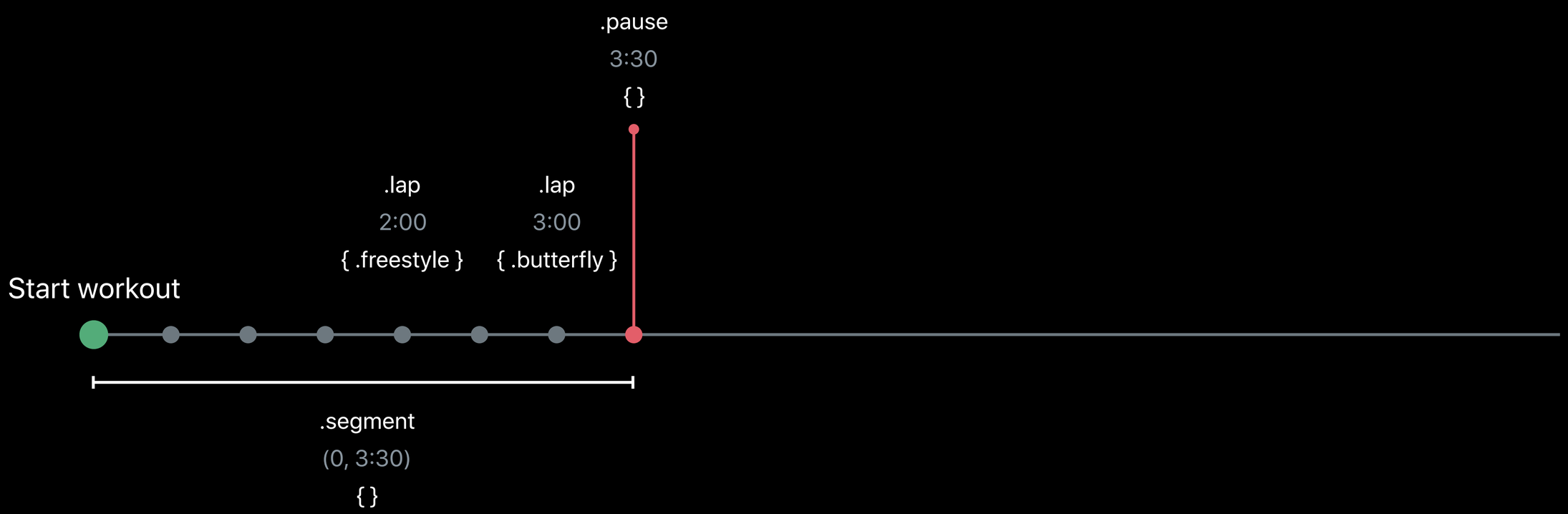


.type
date interval
{metadata}

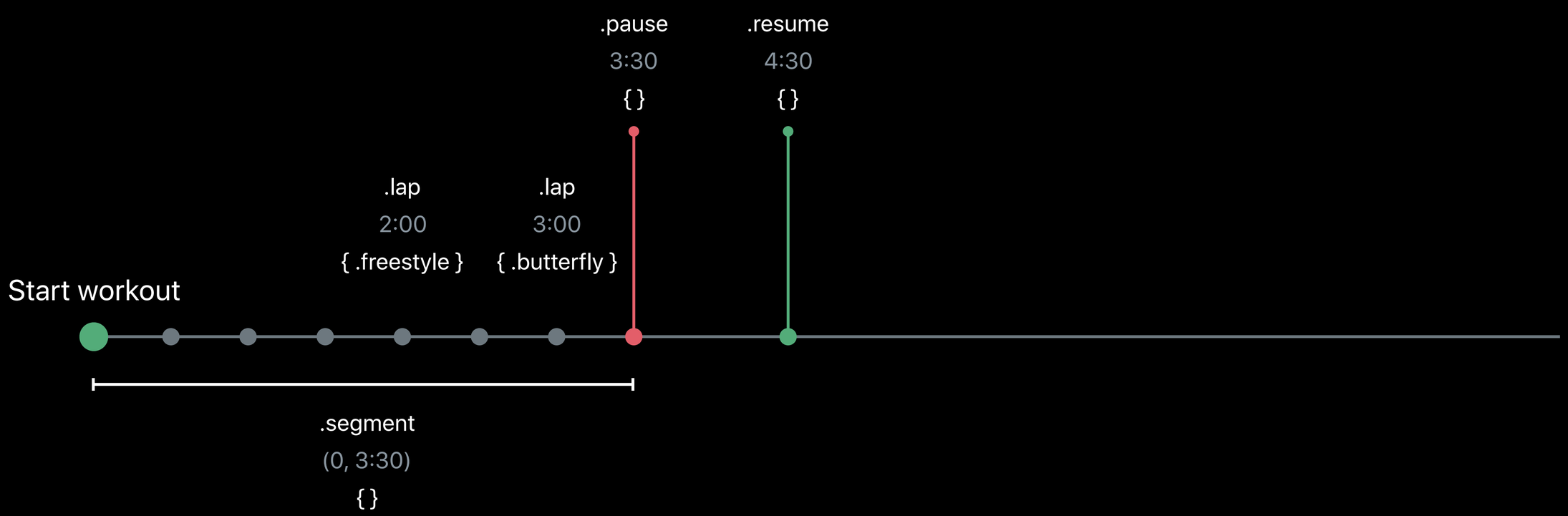

.lap
2:00
{ .freestyle }
.lap
3:00
{ .butterfly }
Start workout
.type
date interval
{metadata}



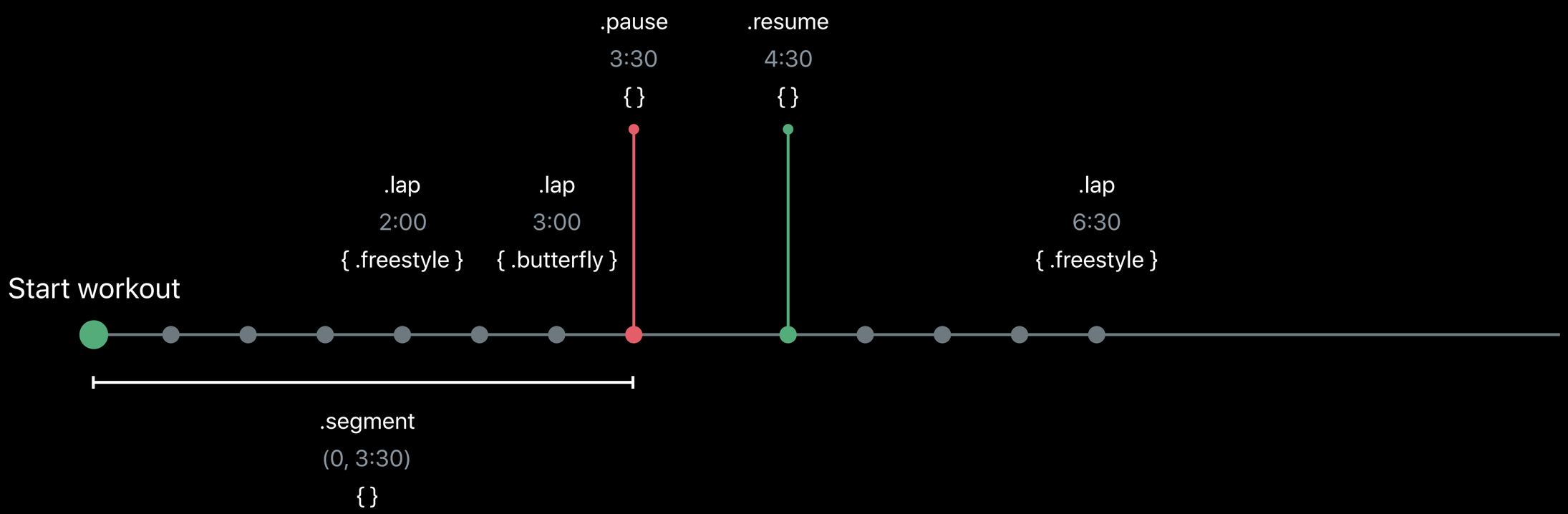
.type
date interval
{metadata}



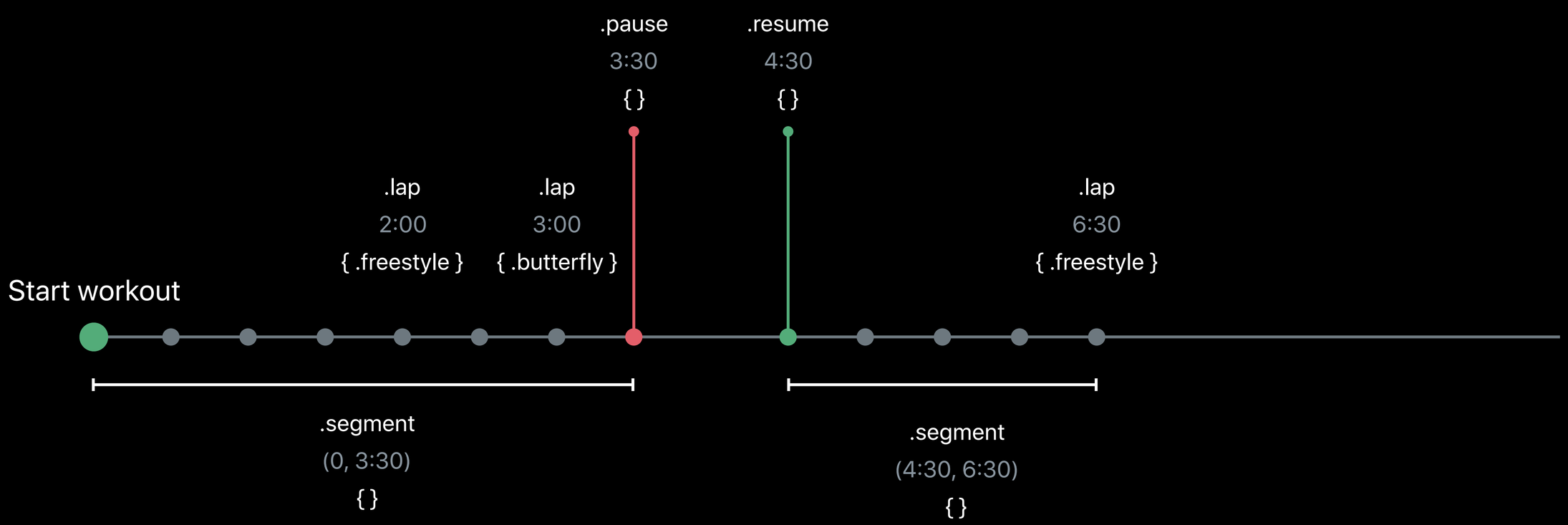
.type
date interval
{metadata}



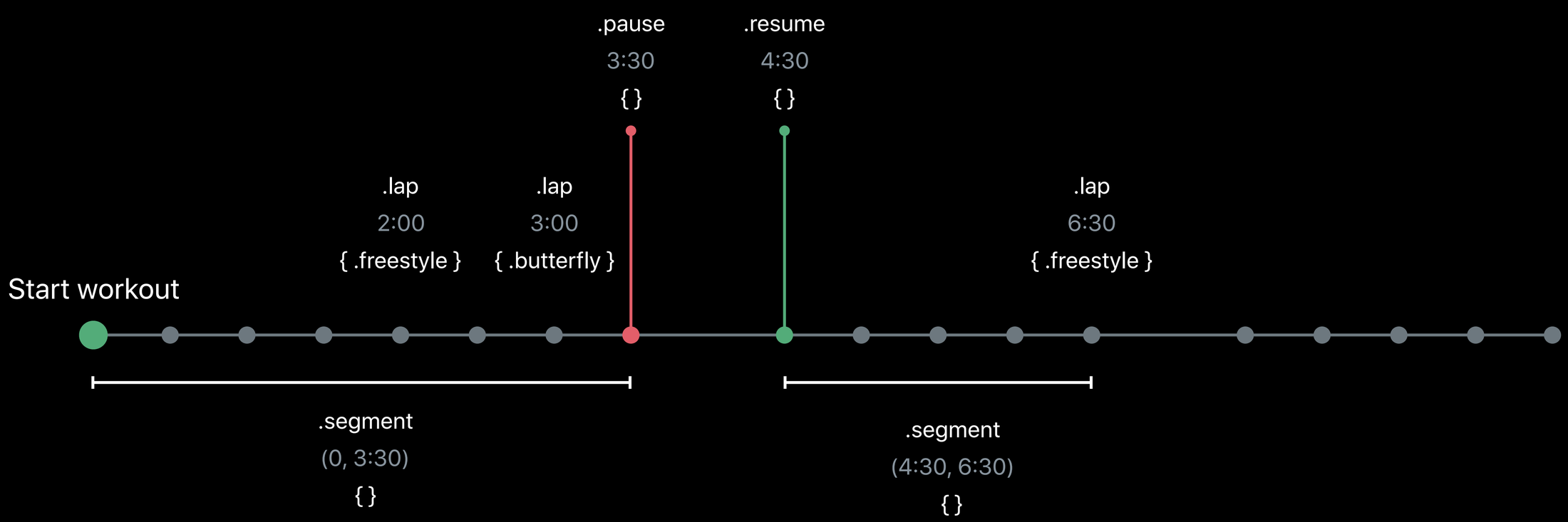
.type
date interval
{metadata}



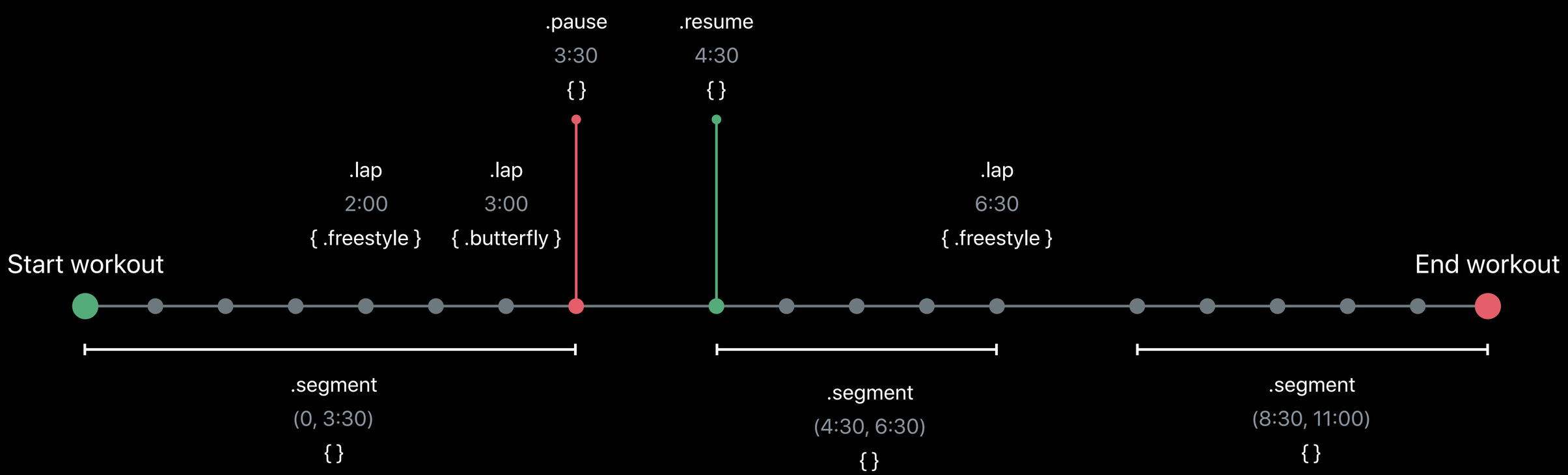
.type
date interval
{metadata}



.type
date interval
{metadata}



.type
date interval
{metadata}



.type
date interval
{metadata}

Workout Request Pause/Resume

NEW

New gesture for pausing and resuming workouts

Workout Request Pause/Resume

NEW

New gesture for pausing and resuming workouts

Quick press of the Digital Crown and side button

Workout Request Pause/Resume

NEW

New gesture for pausing and resuming workouts

Quick press of the Digital Crown and side button

Does work in water lock

Workout Request Pause/Resume

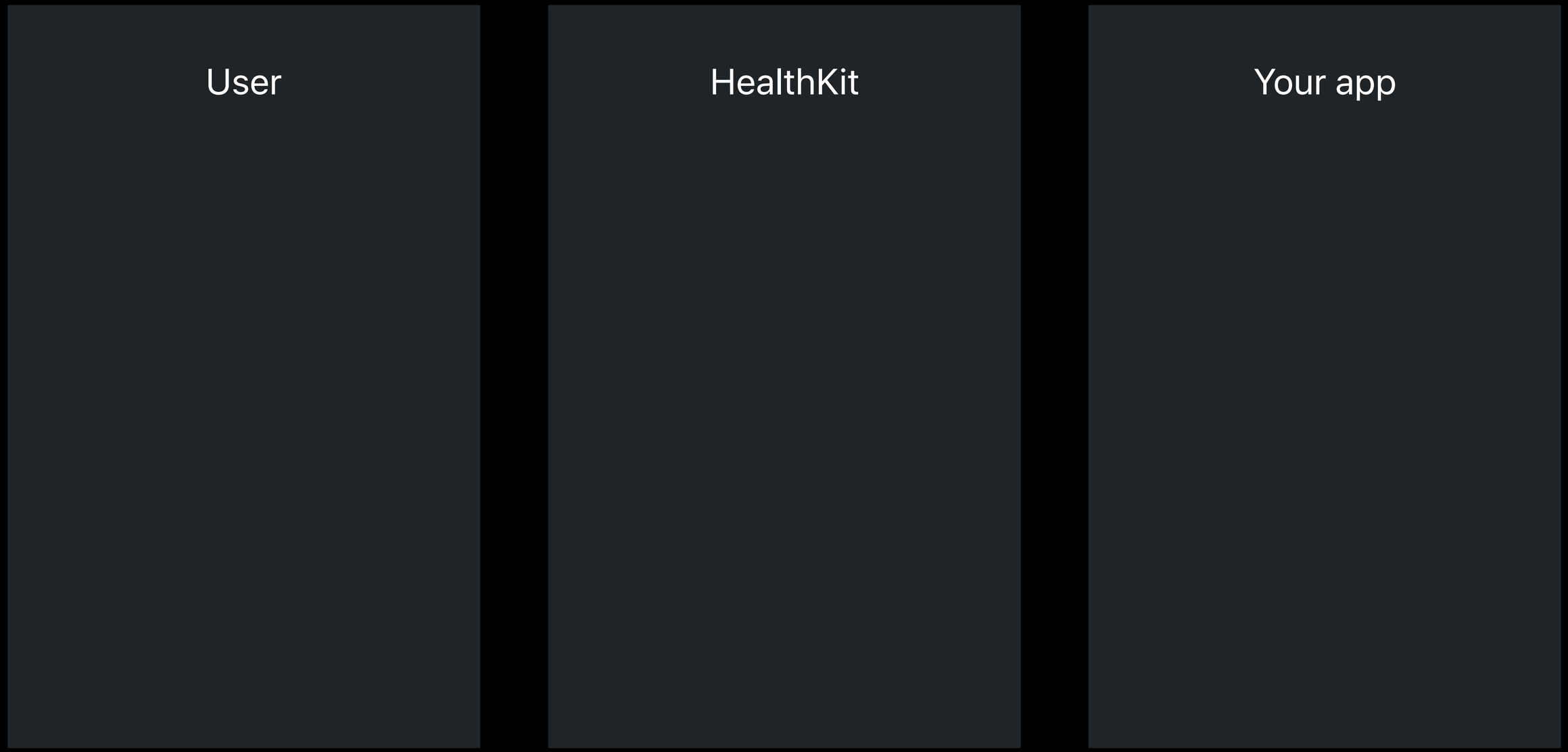
NEW

New gesture for pausing and resuming workouts

Quick press of the Digital Crown and side button

Does work in water lock

Handle in your workout session delegate


User
HealthKit
Your app

User

Presses Digital Crown
and side button

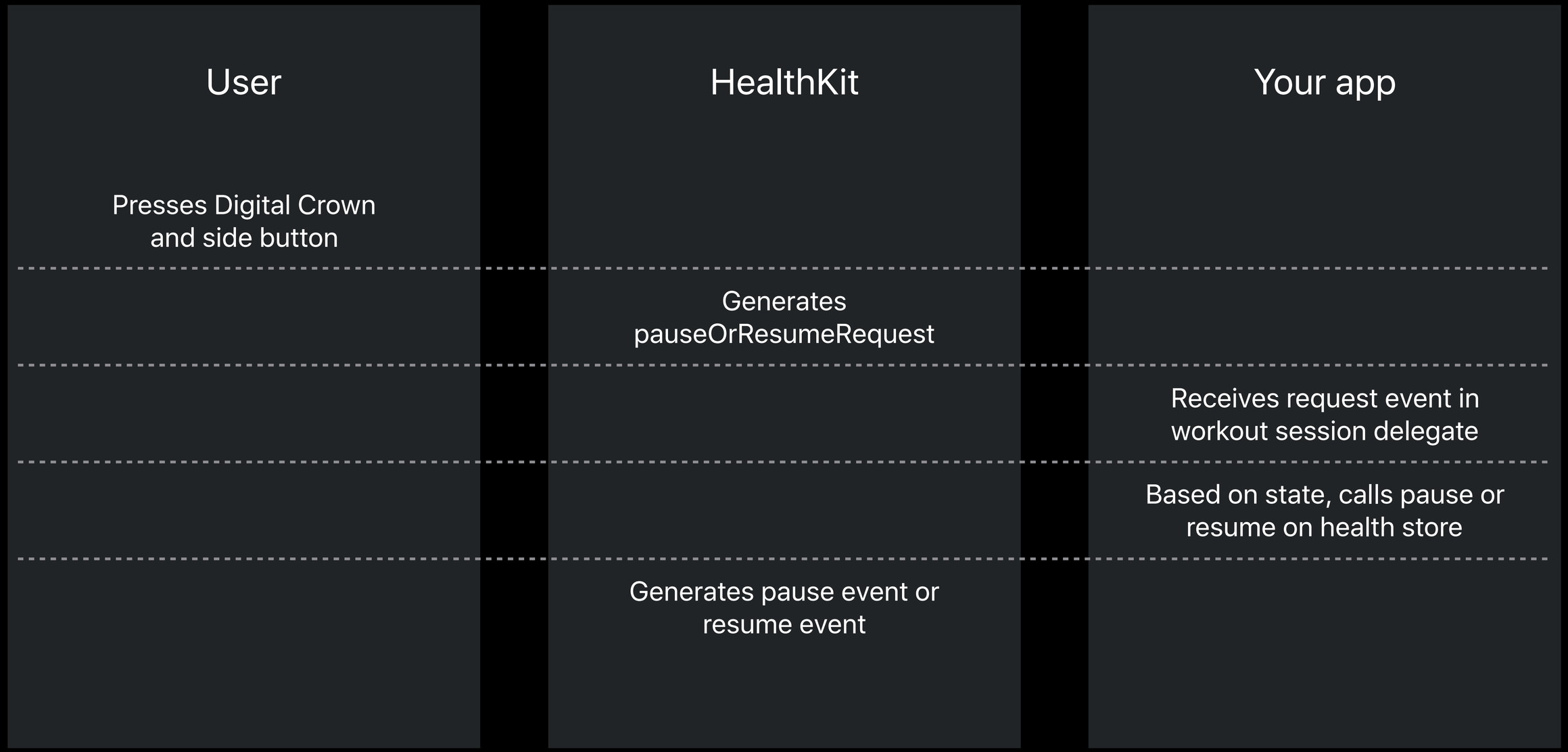
HealthKit

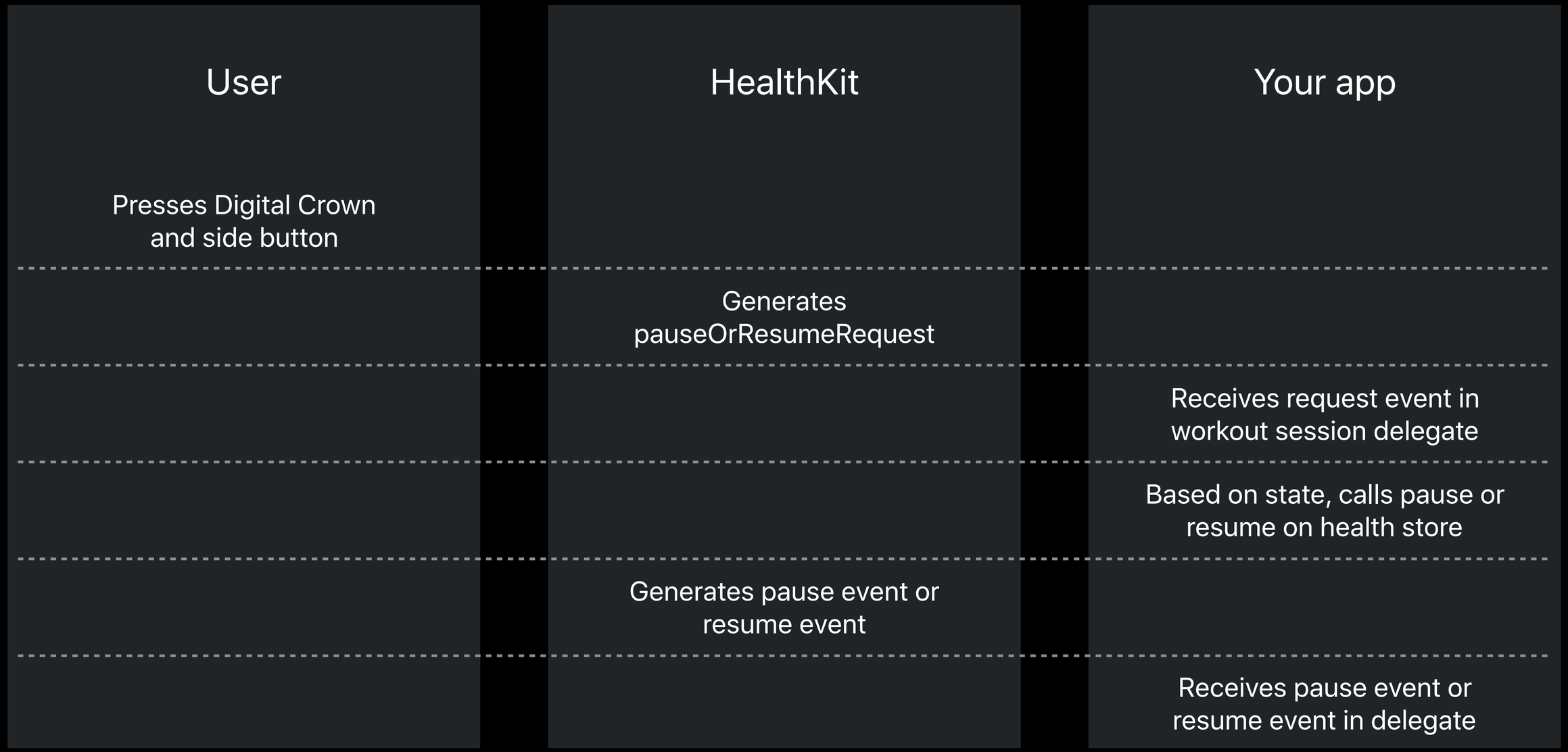
Your app


User
HealthKit
Your app
Presses Digital Crown
and side button
Generates
pauseOrResumeRequest

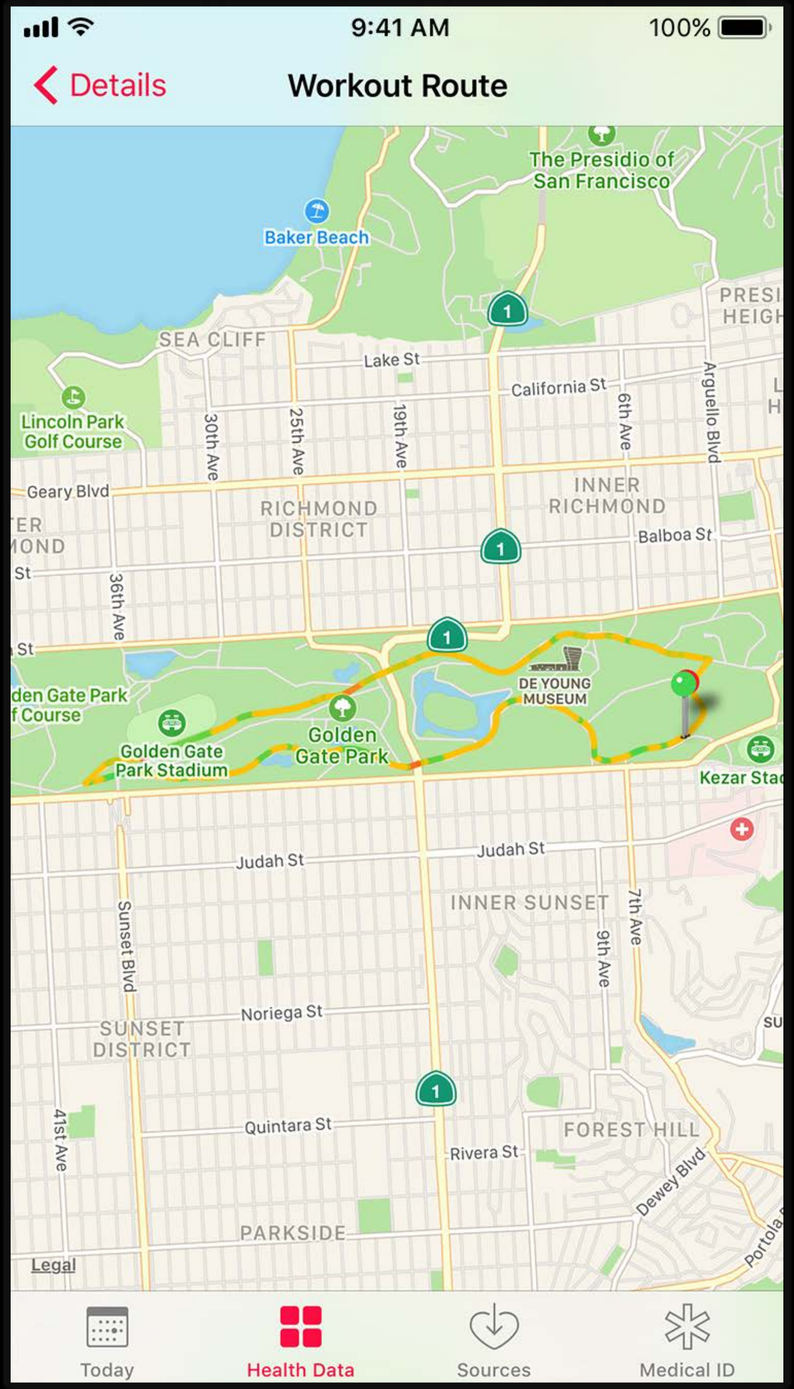

User
HealthKit
Your app
Presses Digital Crown
and side button
Generates
pauseOrResumeRequest
Receives request event in
workout session delegate


User
HealthKit
Your app
Presses Digital Crown and side button
Generates pauseOrResumeRequest
Receives request event in workout session delegate
Based on state, calls pause or resume on health store


User
HealthKit
Your app
Presses Digital Crown and side button
Generates pauseOrResumeRequest
Receives request event in workout session delegate
Based on state, calls pause or resume on health store
Generates pause event or resume event


User
HealthKit
Your app
Presses Digital Crown and side button
Generates pauseOrResumeRequest
Receives request event in workout session delegate
Based on state, calls pause or resume on health store
Generates pause event or resume event
Receives pause event or resume event in delegate

Workout Routes


9:41 AM
100%
Details
Workout Route
The Presidio of San Francisco
Baker Beach
SEA CLIFF
Lake St
California St
Lincoln Park Golf Course
30th Ave
25th Ave
19th Ave
6th Ave
Arguello Blvd
Geary Blvd
RICHMOND DISTRICT
INNER RICHMOND
Balboa St
36th Ave
DE YOUNG MUSEUM
Golden Gate Park Stadium
Golden Gate Park
Judah St
INNER SUNSET
7th Ave
9th Ave
Sunset Blvd
Noriega St
SUNSET DISTRICT
41st Ave
Quintara St
Rivera St
FOREST HILL
Dewey Blvd
PARKSIDE
Legal
Today
Health Data
Sources
Medical ID

Reading Workout Routes

New data type

NEW

`HKWorkoutRouteType`

Reading Workout Routes

New data type

NEW

`HKWorkoutRouteType`

Requires additional authorization

Reading Workout Routes

New data type

NEW

`HKWorkoutRouteType`

Requires additional authorization

Modeled as an array of CLLocations

Reading Workout Routes

New data type

NEW

`HKWorkoutRouteType`

Requires additional authorization

Modeled as an array of CLLocations

Datasets can be large

Reading Workout Routes

New data type

NEW

`HKWorkoutRouteType`

Requires additional authorization

Modeled as an array of CLLocations

Datasets can be large

- New `HKWorkoutRouteQuery`

Reading Workout Routes

New data type

NEW

`HKWorkoutRouteType`

Requires additional authorization

Modeled as an array of CLLocations

Datasets can be large

- New `HKWorkoutRouteQuery`
- Returns location data in batches

```
// Step 1: Query for samples of type HKWorkoutRoute associated to your workout
```

```
// Step 1: Query for samples of type HKWorkoutRoute associated to your workout
let workoutRouteType = HKSeriesType.workoutRoute()
let workoutPredicate = HKQuery.predicateForObjects(from: workout)
```

```
// Step 1: Query for samples of type HKWorkoutRoute associated to your workout
let workoutRouteType = HKSeriesType.workoutRoute()
let workoutPredicate = HKQuery.predicateForObjects(from: workout)

let workoutRoutesQuery = HKSampleQuery(sampleType: workoutRouteType,
    predicate: workoutPredicate, limit: HKObjectQueryNoLimit, sortDescriptors: nil)
    { (query, samples, error) in









}
```

```swift
// Step 1: Query for samples of type HKWorkoutRoute associated to your workout
let workoutRouteType = HKSeriesType.workoutRoute()
let workoutPredicate = HKQuery.predicateForObjects(from: workout)

let workoutRoutesQuery = HKSampleQuery(sampleType: workoutRouteType,
    predicate: workoutPredicate, limit: HKObjectQueryNoLimit, sortDescriptors: nil)
    { (query, samples, error) in
    guard let routeSamples = samples as? [HKWorkoutRoute] else { return }









}
```

```
// Step 1: Query for samples of type HKWorkoutRoute associated to your workout
let workoutRouteType = HKSeriesType.workoutRoute()
let workoutPredicate = HKQuery.predicateForObjects(from: workout)

let workoutRoutesQuery = HKSampleQuery(sampleType: workoutRouteType,
    predicate: workoutPredicate, limit: HKObjectQueryNoLimit, sortDescriptors: nil)
    { (query, samples, error) in
    guard let routeSamples = samples as? [HKWorkoutRoute] else { return }

    // Step 2: Query for location data from the routes
    for routeSample in routeSamples {









    }
}
```

```swift
// Step 1: Query for samples of type HKWorkoutRoute associated to your workout
let workoutRouteType = HKSeriesType.workoutRoute()
let workoutPredicate = HKQuery.predicateForObjects(from: workout)

let workoutRoutesQuery = HKSampleQuery(sampleType: workoutRouteType,
    predicate: workoutPredicate, limit: HKObjectQueryNoLimit, sortDescriptors: nil)
    { (query, samples, error) in
    guard let routeSamples = samples as? [HKWorkoutRoute] else { return }

    // Step 2: Query for location data from the routes
    for routeSample in routeSamples {
        let locationQuery = HKWorkoutRouteQuery(route: routeSample) {
            (routeQuery, locations, done, error) in

        }

    }
}
```

```swift
// Step 1: Query for samples of type HKWorkoutRoute associated to your workout
let workoutRouteType = HKSeriesType.workoutRoute()
let workoutPredicate = HKQuery.predicateForObjects(from: workout)

let workoutRoutesQuery = HKSampleQuery(sampleType: workoutRouteType,
    predicate: workoutPredicate, limit: HKObjectQueryNoLimit, sortDescriptors: nil)
    { (query, samples, error) in
    guard let routeSamples = samples as? [HKWorkoutRoute] else { return }

    // Step 2: Query for location data from the routes
    for routeSample in routeSamples {
        let locationQuery = HKWorkoutRouteQuery(route: routeSample) {
            (routeQuery, locations, done, error) in
            self.addLocationsToMapDisplay(locations)
        }

    }
}
```

```swift
// Step 1: Query for samples of type HKWorkoutRoute associated to your workout
let workoutRouteType = HKSeriesType.workoutRoute()
let workoutPredicate = HKQuery.predicateForObjects(from: workout)

let workoutRoutesQuery = HKSampleQuery(sampleType: workoutRouteType,
    predicate: workoutPredicate, limit: HKObjectQueryNoLimit, sortDescriptors: nil)
    { (query, samples, error) in
    guard let routeSamples = samples as? [HKWorkoutRoute] else { return }

    // Step 2: Query for location data from the routes
    for routeSample in routeSamples {
        let locationQuery = HKWorkoutRouteQuery(route: routeSample) {
            (routeQuery, locations, done, error) in
            self.addLocationsToMapDisplay(locations)
        }
        self.healthStore.execute(locationQuery)
    }
}
self.healthStore.execute(workoutRoutesQuery)
```

Building and Saving Workout Routes

NEW

Builder model—HKWorkoutRouteBuilder

Building and Saving Workout Routes

NEW

Builder model—HKWorkoutRouteBuilder

Location data is added asynchronously

Building and Saving Workout Routes

NEW

Builder model—HKWorkoutRouteBuilder

Location data is added asynchronously

Data is sorted by date when the series is finished

Building and Saving Workout Routes

NEW

Builder model—HKWorkoutRouteBuilder

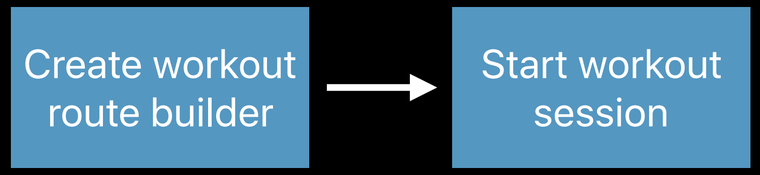
Location data is added asynchronously

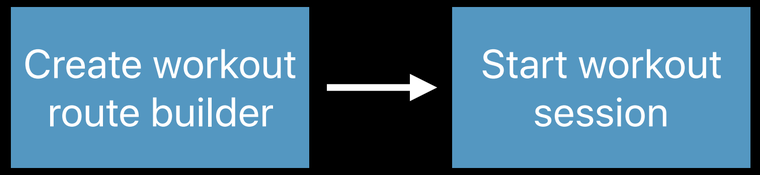
Data is sorted by date when the series is finished

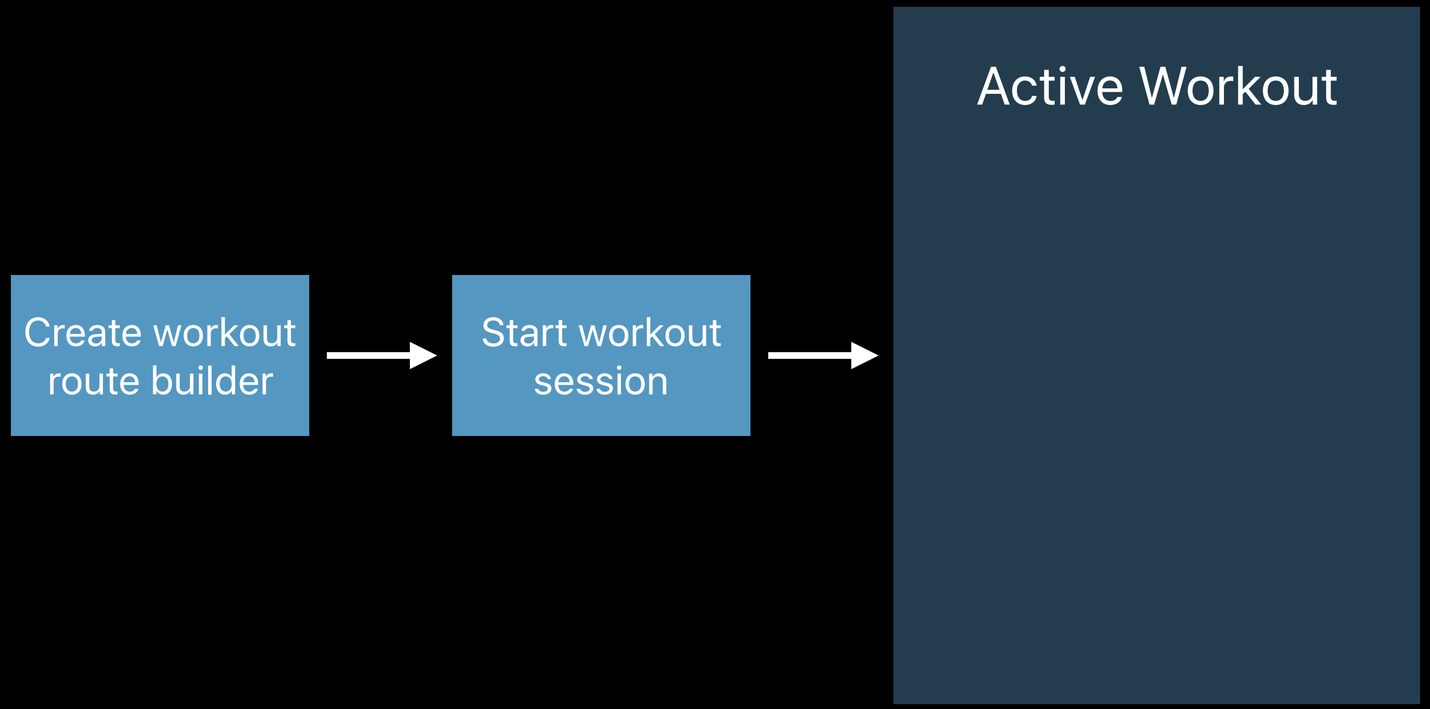
The workout must be saved before the route

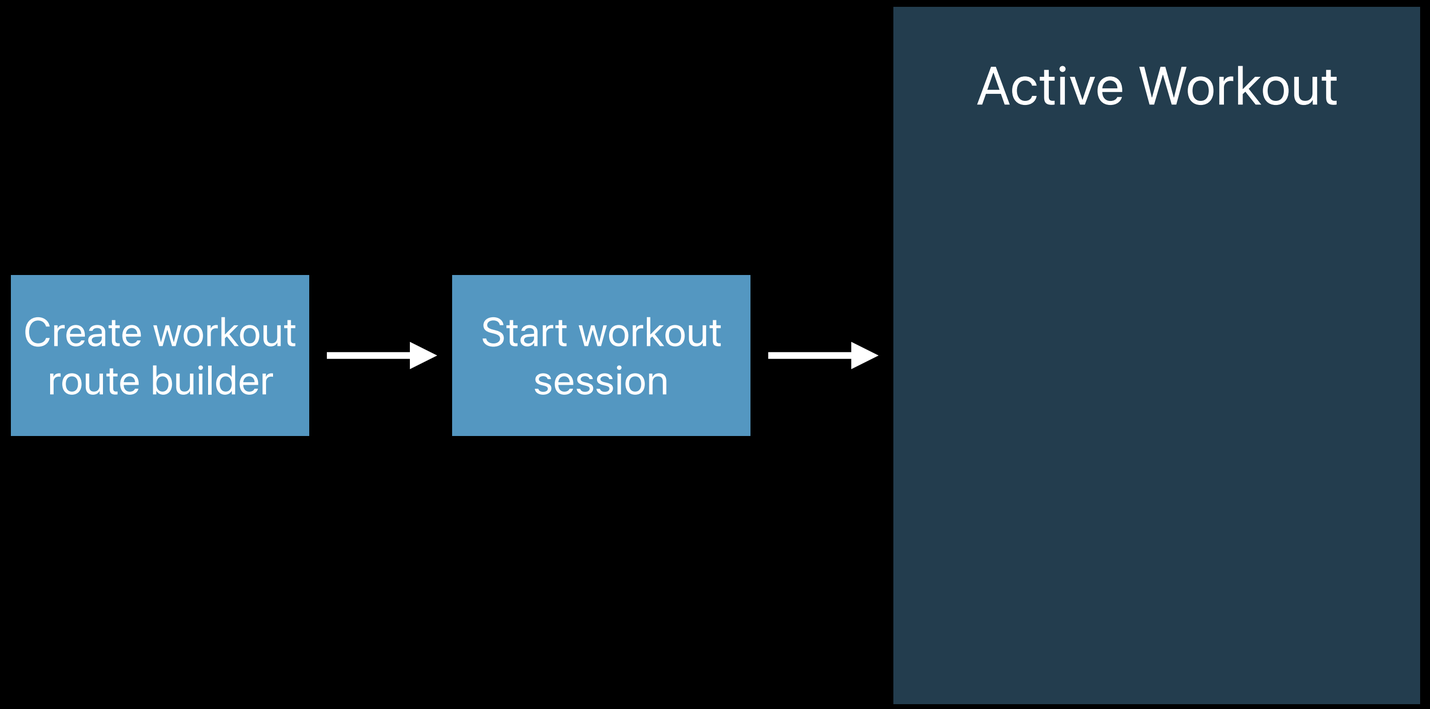
Create workout route builder

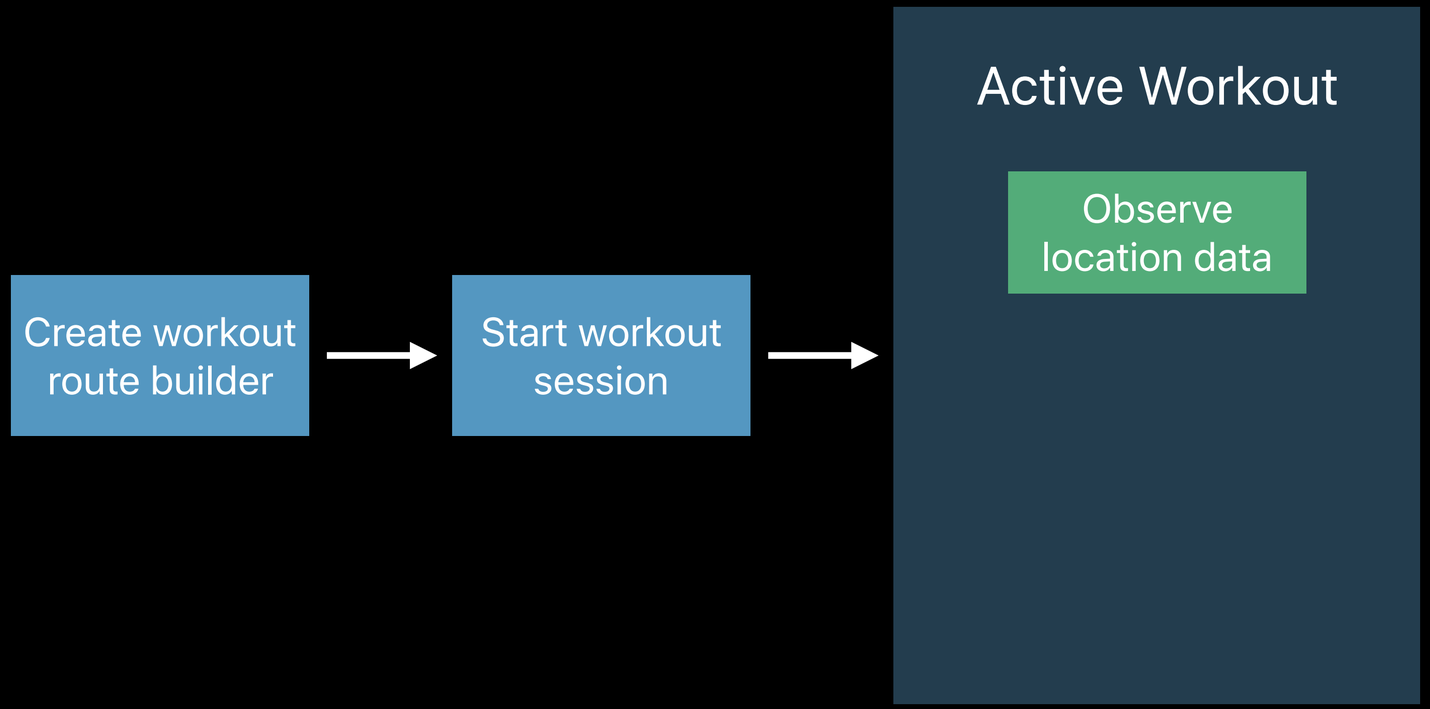
Create workout route builder

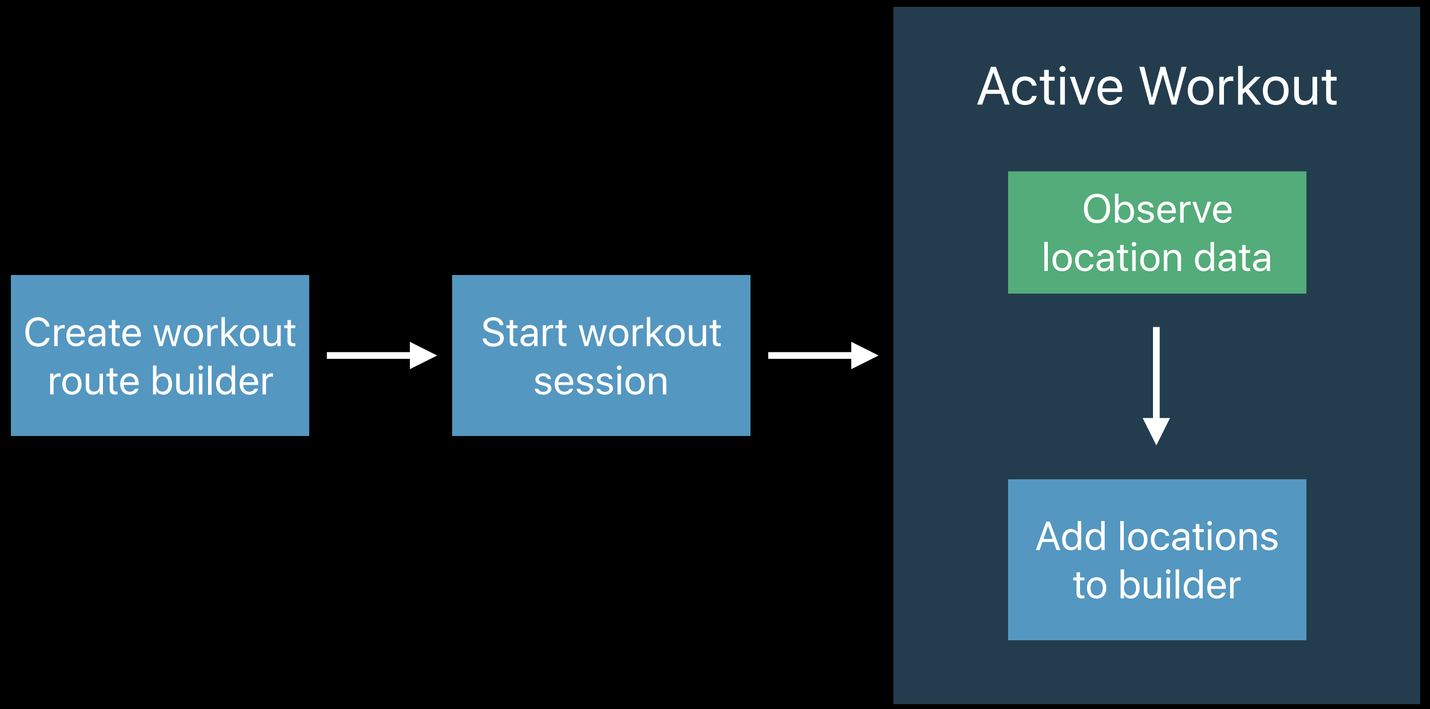

Create workout route builder
Start workout session

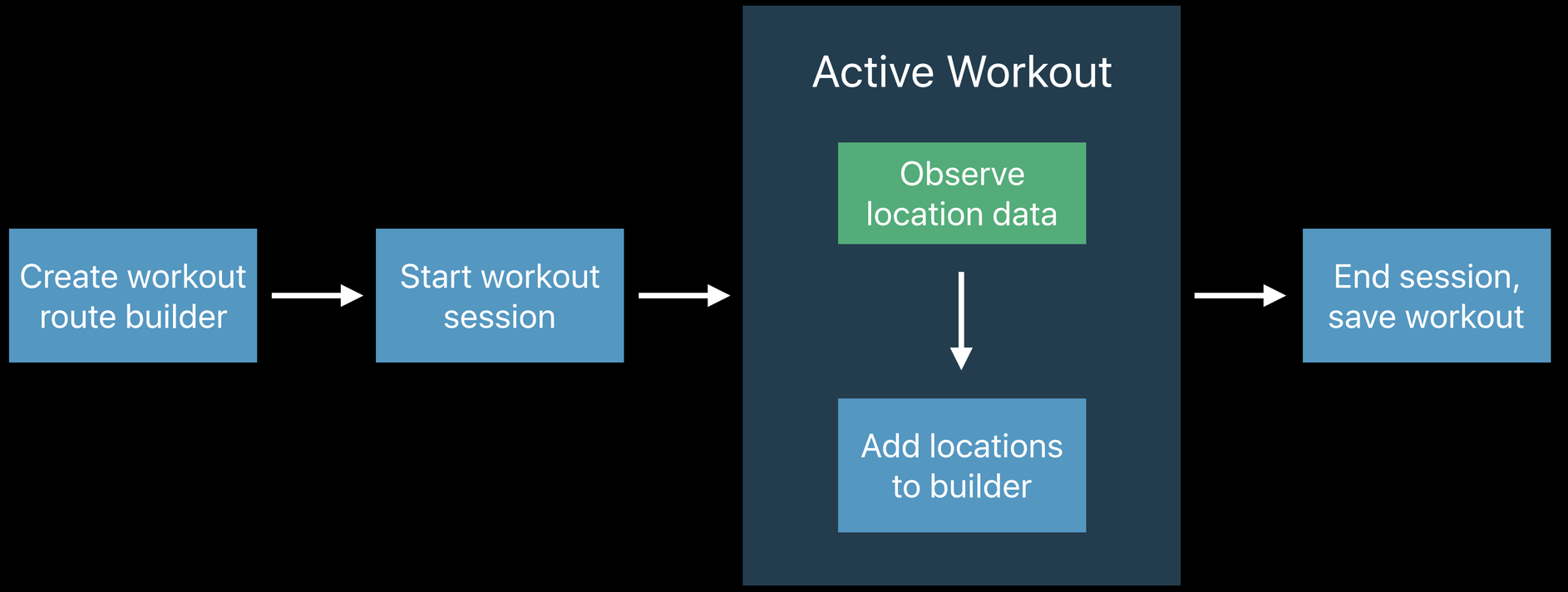

Create workout route builder
Start workout session

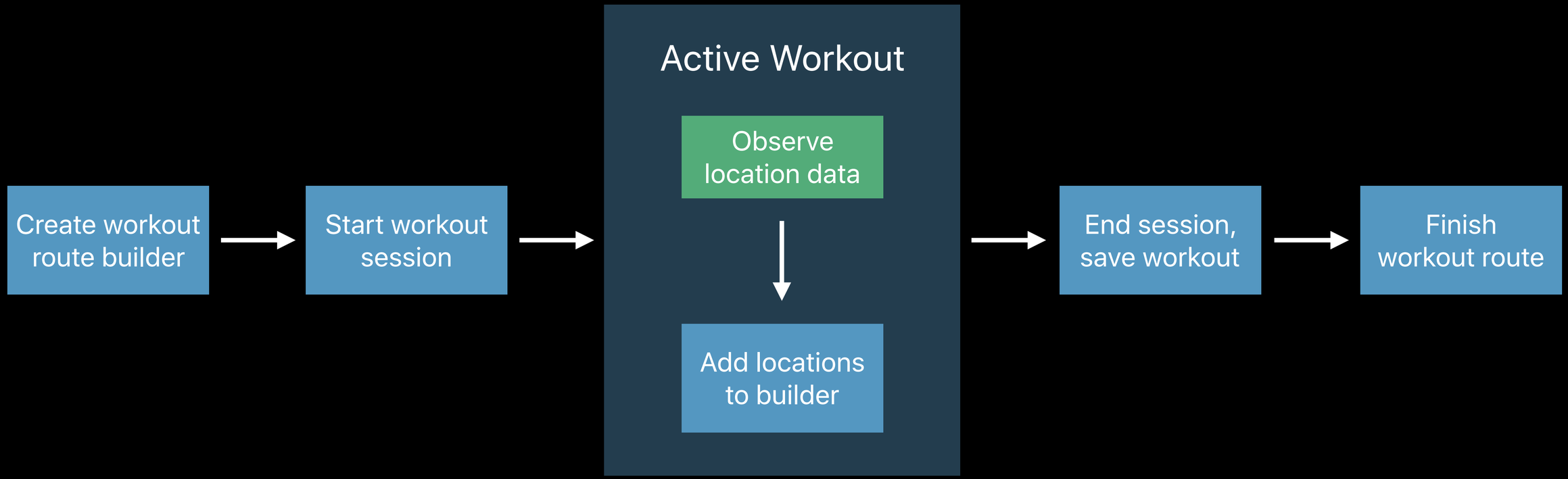

Create workout route builder
Start workout session
Active Workout


Create workout route builder
Start workout session
Active Workout


Create workout route builder
Start workout session
Active Workout
Observe location data


Create workout route builder
Start workout session
Active Workout
Observe location data
Add locations to builder


Create workout route builder
Start workout session
Active Workout
Observe location data
Add locations to builder
End session, save workout


Create workout route builder
Start workout session
Active Workout
Observe location data
Add locations to builder
End session, save workout
Finish workout route

```
// Step 1: Create a route builder and add locations
let builder = HKWorkoutRouteBuilder(healthStore: healthStore, device: nil)
```

```swift
// Step 1: Create a route builder and add locations
let builder = HKWorkoutRouteBuilder(healthStore: healthStore, device: nil)

// Step 2: Add locations as the workout is ongoing
let locations: [CLLocation] = self.fetchRecentLocations()
builder.insertRouteData(locations) { (success, error) in
    // Handle errors...
}
```

```swift
// Step 1: Create a route builder and add locations
let builder = HKWorkoutRouteBuilder(healthStore: healthStore, device: nil)

// Step 2: Add locations as the workout is ongoing
let locations: [CLLocation] = self.fetchRecentLocations()
builder.insertRouteData(locations) { (success, error) in
   // Handle errors...
}

// Step 3: After the workout is saved, save the route data
builder.finishRoute(with: workout, metadata: nil) { (workoutRoute, error) in
    // Handle errors...
}
```

Workout Route Demo

Incorporating routes into Speedy Sloth

HKObject Sync Identifiers

Michael Ozeryansky, iOS Software Engineer

HKObject Sync Identifiers

Identifiers and versioning

NEW

```
public let HKMetadataKeySyncIdentifier: String
public let HKMetadataKeySyncVersion: String
```

HKObject Sync Identifiers

Identifiers and versioning

NEW

```
public let HKMetadataKeySyncIdentifier: String
public let HKMetadataKeySyncVersion: String
```

Identifier can be any String

HKObject Sync Identifiers

Identifiers and versioning

NEW

```
public let HKMetadataKeySyncIdentifier: String
public let HKMetadataKeySyncVersion: String
```

Identifier can be any String

Version can be any Number

HKObject Sync Identifiers

Identifiers and versioning

NEW

```
public let HKMetadataKeySyncIdentifier: String
public let HKMetadataKeySyncVersion: String
```

Identifier can be any String

Version can be any Number

Use both keys together when saving an HKObject

HKObject Sync Identifiers

Identifiers and versioning

NEW

```
public let HKMetadataKeySyncIdentifier: String
public let HKMetadataKeySyncVersion: String
```

Identifier can be any String

Version can be any Number

Use both keys together when saving an HKObject

Restricted to your source

HKObject Sync Identifiers

Identifiers and versioning

NEW

HKObject Sync Identifiers

Identifiers and versioning

NEW

Sample uniqueness

HKObject Sync Identifiers

Identifiers and versioning

NEW

Sample uniqueness

Local versioning

HKObject Sync Identifiers

Identifiers and versioning

NEW

Sample uniqueness

Local versioning

Transaction safe

HKObject Sync Identifiers

Identifiers and versioning

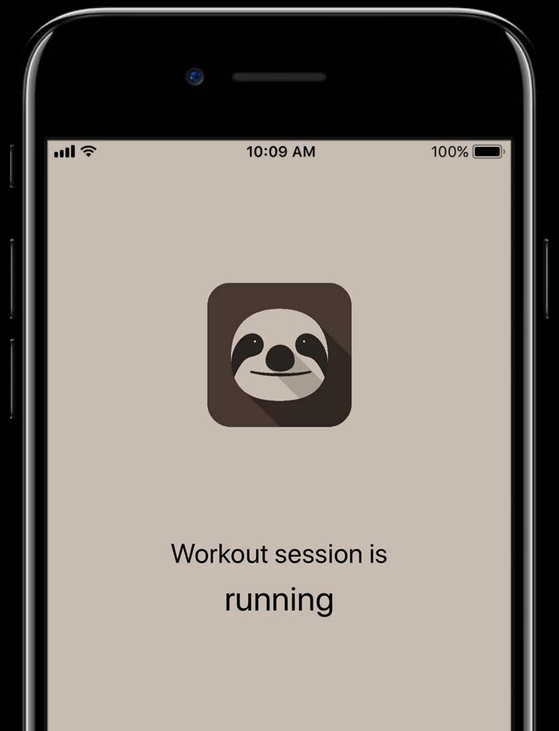
NEW

Sample uniqueness

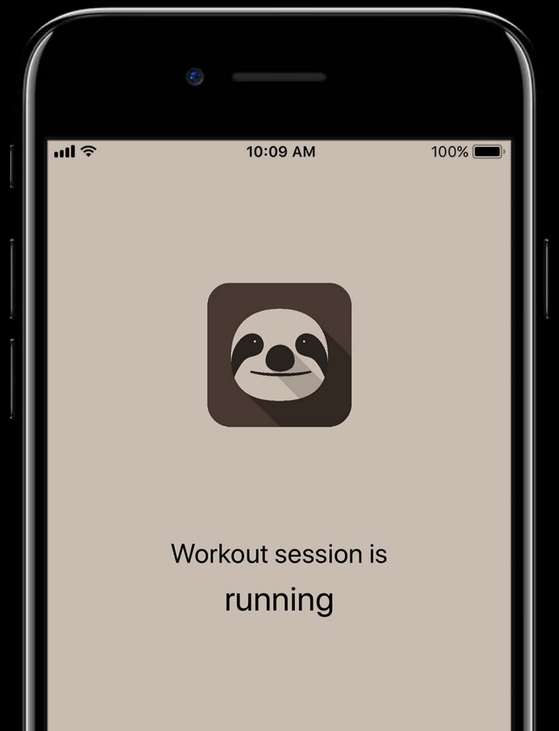
Local versioning

Transaction safe

Relationships are maintained

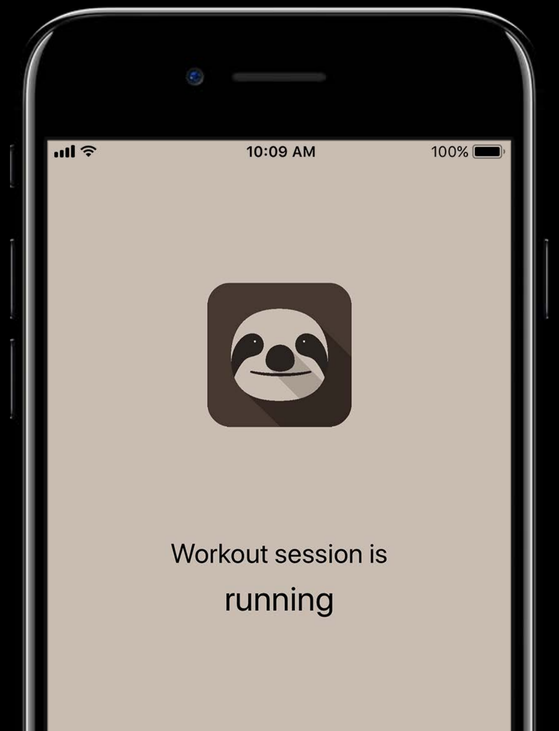

10:09 AM
100%
Workout session is
running


Walking
10:09
0:01:13
1.9 Calories
78.0 Meters
Pause
Marker
Stop


10:09 AM
100%
Workout session is
running

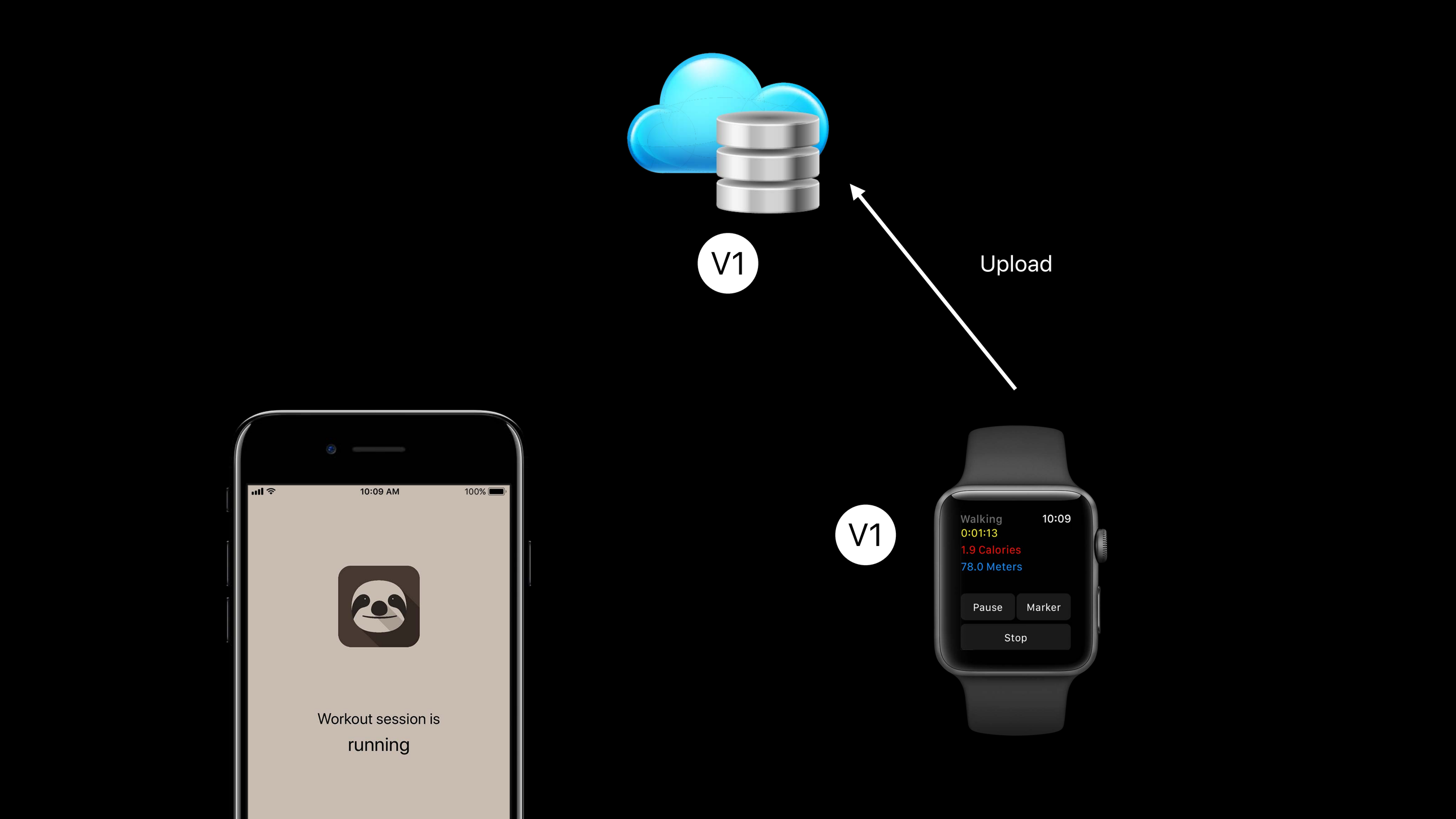
V1


Walking
10:09
0:01:13
1.9 Calories
78.0 Meters
Pause
Marker
Stop

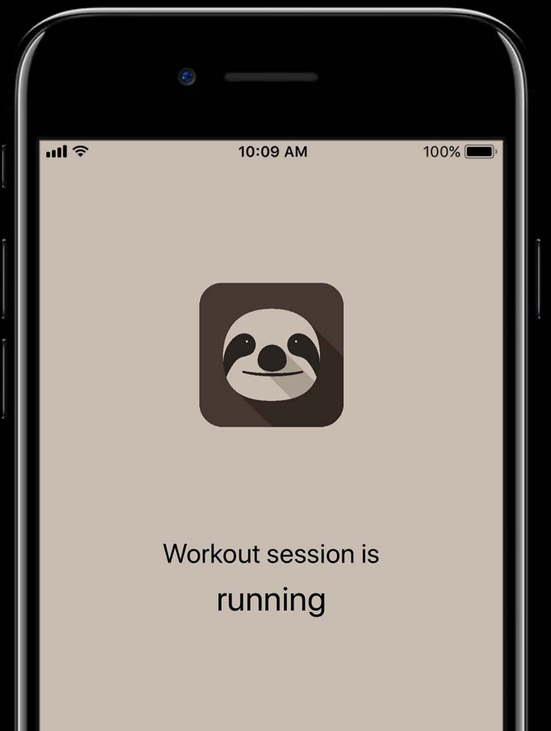

10:09 AM
100%
Workout session is
running

V1


Walking
10:09
0:01:13
1.9 Calories
78.0 Meters
Pause
Marker
Stop


V1
Upload
10:09 AM
100%
Workout session is
running
V1
Walking
10:09
0:01:13
1.9 Calories
78.0 Meters
Pause
Marker
Stop

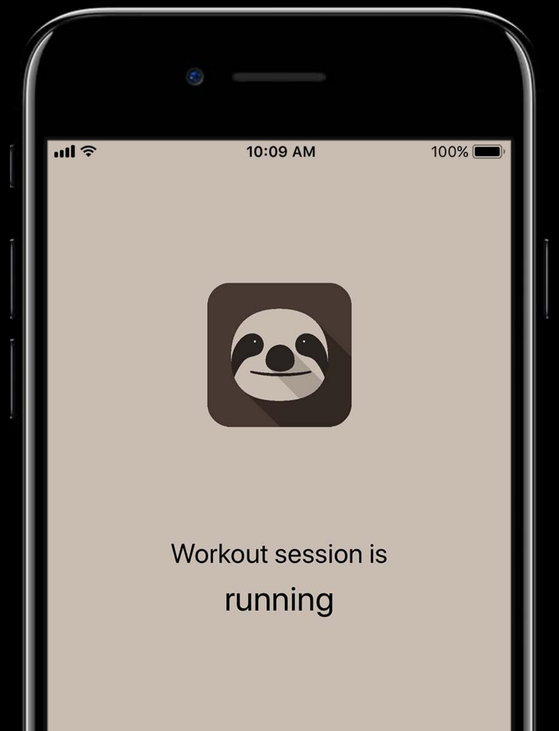
V1


10:09 AM
100%
Workout session is
running

V1


Walking
10:09
0:01:13
1.9 Calories
78.0 Meters
Pause
Marker
Stop

V1


10:09 AM
100%
Workout session is
running

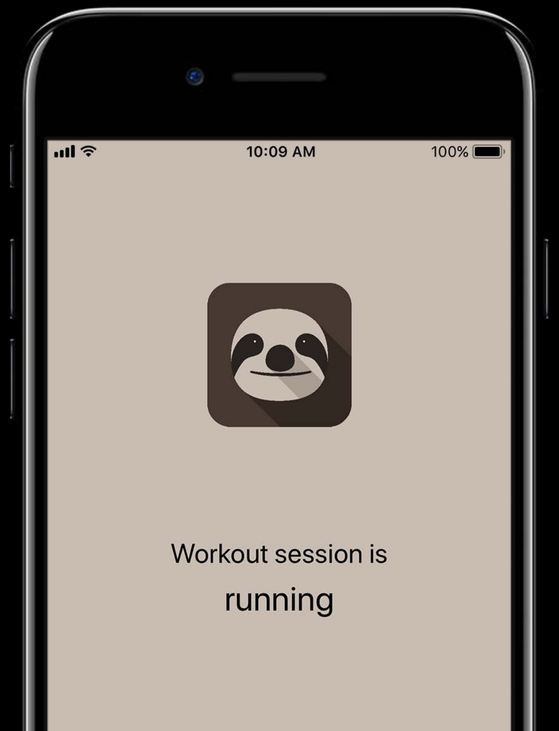
V1

Sync


Walking
10:09
0:01:13
1.9 Calories
78.0 Meters
Pause
Marker
Stop


V1
10:09 AM
100%
Workout session is
running
V1
V1
Sync
Walking
10:09
0:01:13
1.9 Calories
78.0 Meters
Pause
Marker
Stop

V1


10:09 AM
100%
Workout session is
running

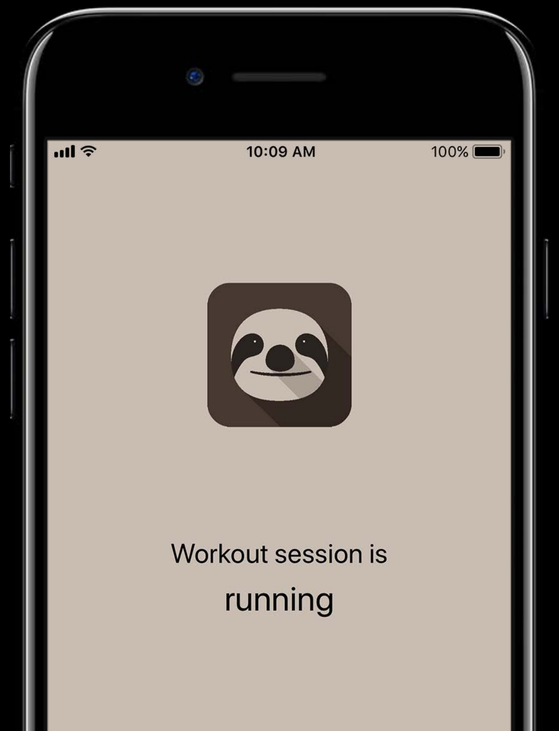
V1

V1


Walking
10:09
0:01:13
1.9 Calories
78.0 Meters
Pause
Marker
Stop

V2

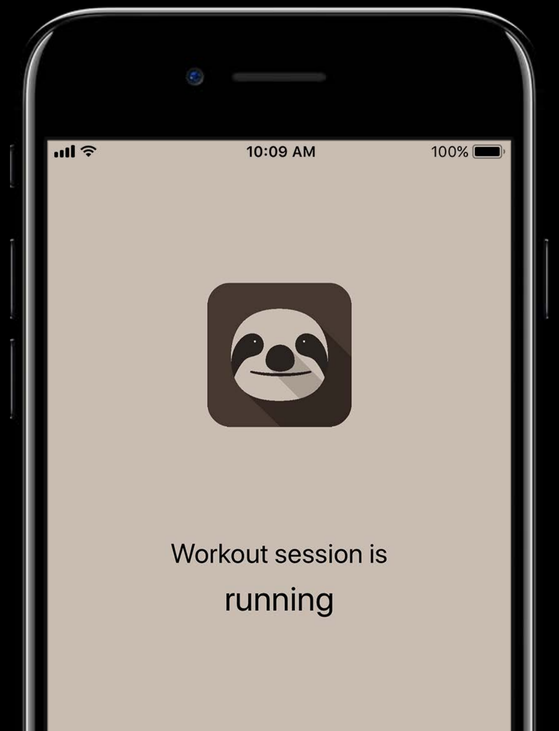
Processing



V1

V1

V2


10:09 AM
100%
Workout session is
running

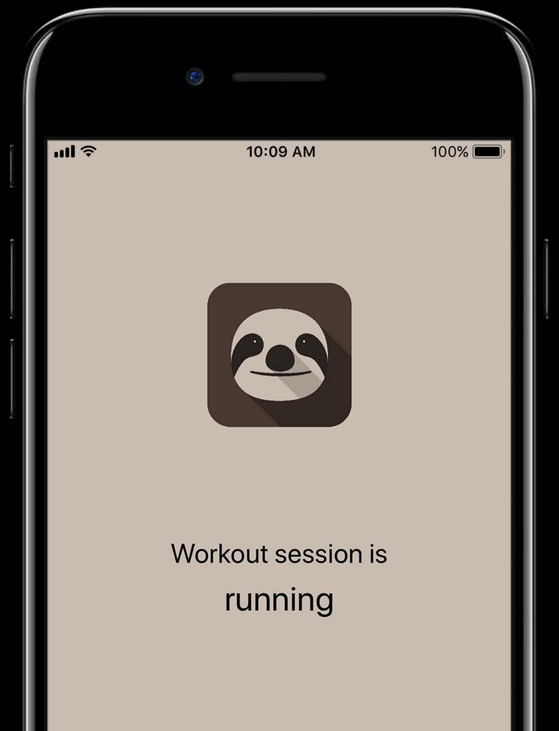
V1

V1


Walking
10:09
0:01:13
1.9 Calories
78.0 Meters
Pause
Marker
Stop


Download
V2
10:09 AM
100%
Workout session is
running
V1
V1
Walking
10:09
0:01:13
1.9 Calories
78.0 Meters
Pause
Marker
Stop

V2

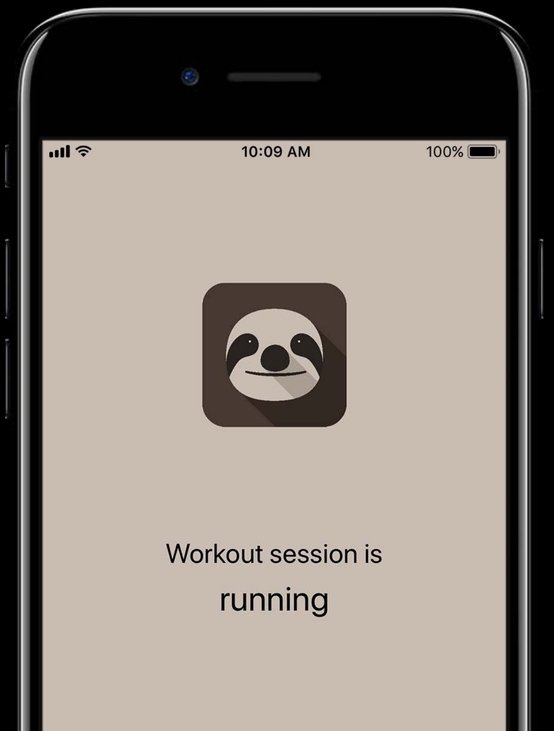

10:09 AM
100%
Workout session is
running

V1

V1


Walking
10:09
0:01:13
1.9 Calories
78.0 Meters
Pause
Marker
Stop

V2

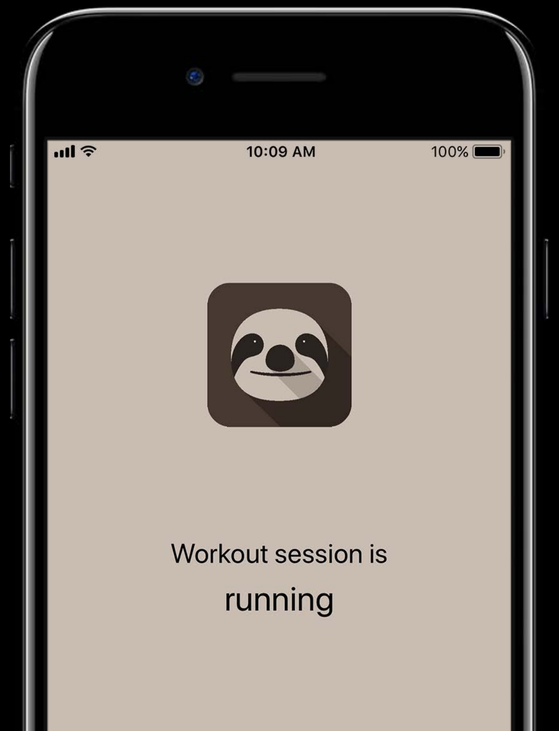


Delete
and add

V2

V1

V2


10:09 AM
100%
Workout session is
running

V2

V1

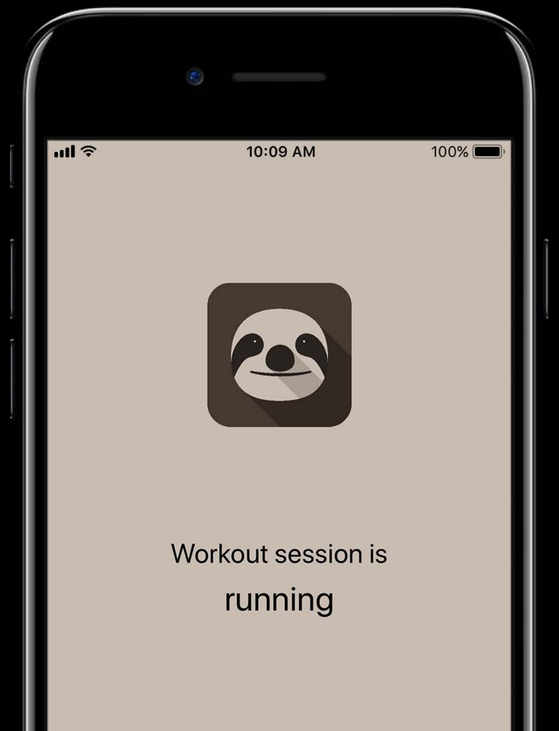

Walking
10:09
0:01:13
1.9 Calories
78.0 Meters
Pause
Marker
Stop


V2
10:09 AM
100%
Workout session is
running
V2
V1
Sync
Walking
10:09
0:01:13
1.9 Calories
78.0 Meters
Pause
Marker
Stop


V2
10:09 AM
100%
Workout session is
running
V2
V1
Sync
Walking
10:09
0:01:13
1.9 Calories
78.0 Meters
Pause
Marker
Stop


V2
10:09 AM
100%
Workout session is
running
V2
Sync
Delete
and add
V2
Walking
10:09
0:01:13
1.9 Calories
78.0 Meters
Pause
Marker
Stop

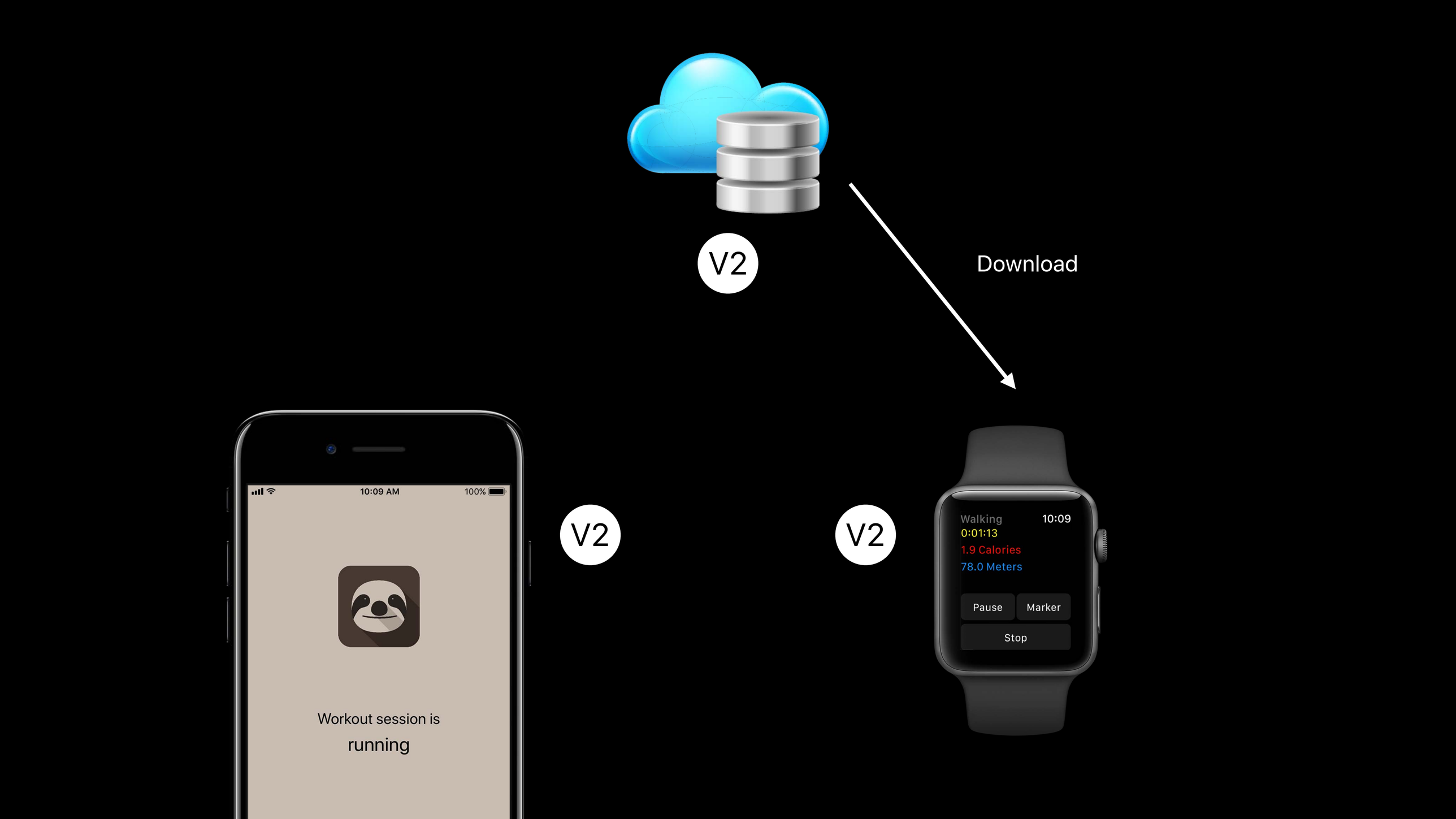
V2


10:09 AM
100%
Workout session is
running

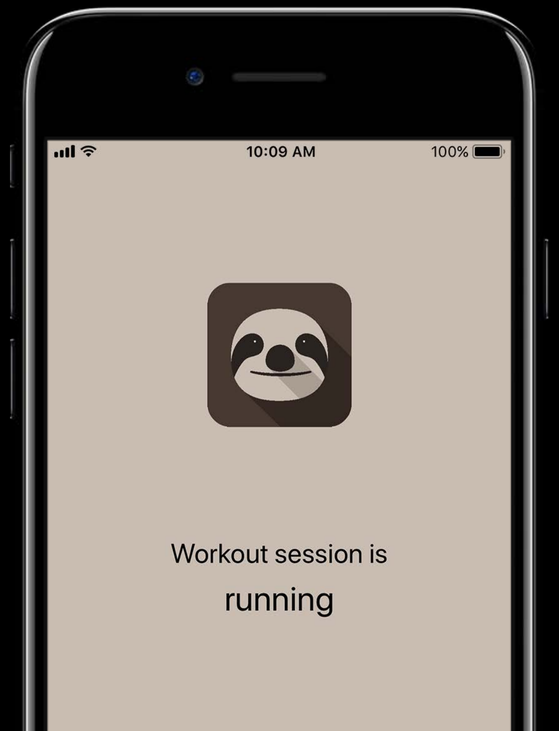
V2

V2


Walking
10:09
0:01:13
1.9 Calories
78.0 Meters
Pause
Marker
Stop


V2
Download
10:09 AM
100%
Workout session is
running
V2
V2
Walking
10:09
0:01:13
1.9 Calories
78.0 Meters
Pause
Marker
Stop

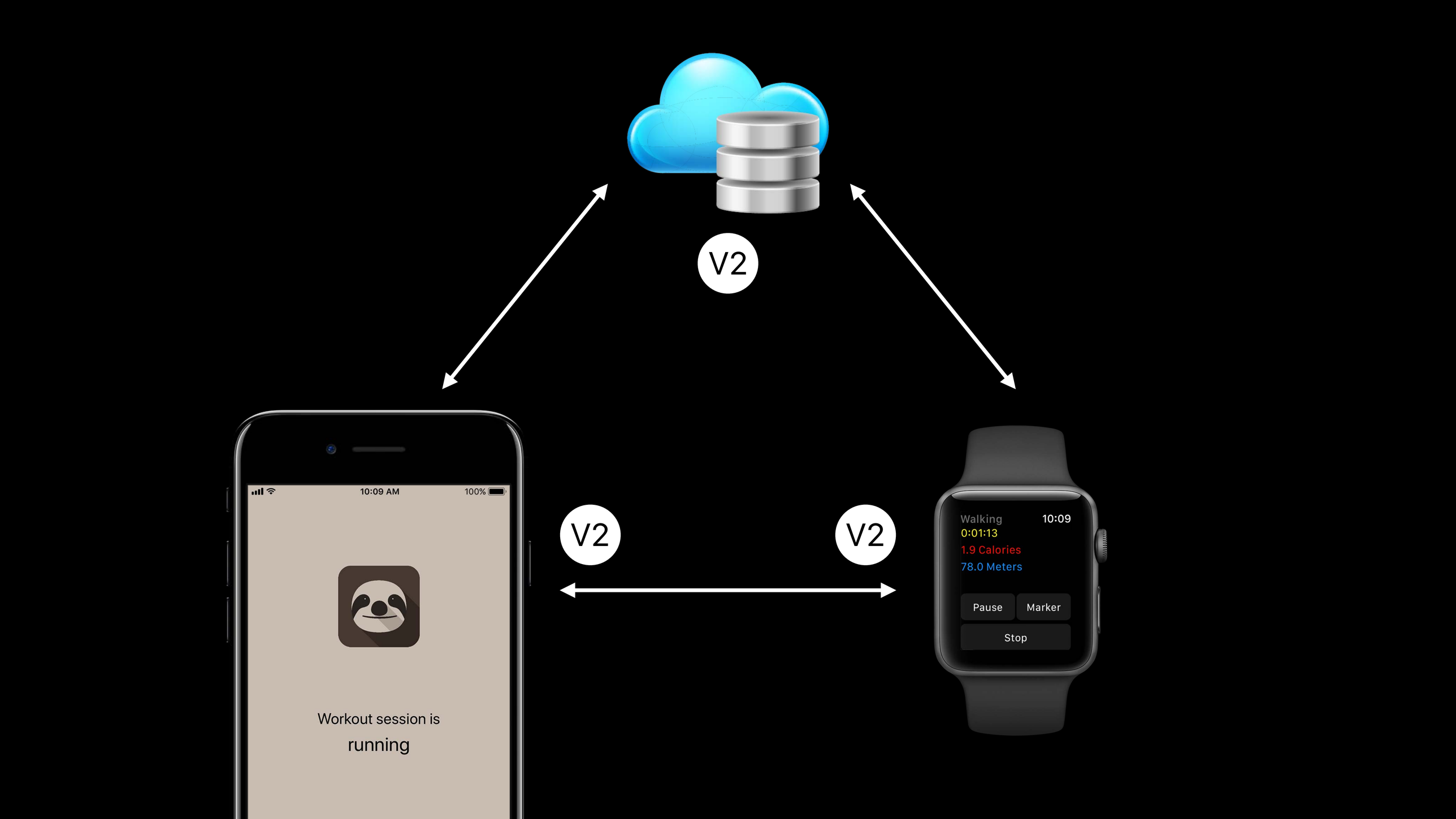
V2


10:09 AM
100%
Workout session is
running

V2

V2


Walking
10:09
0:01:13
1.9 Calories
78.0 Meters
Pause
Marker
Stop


V2
10:09 AM
100%
Workout session is
running
V2
V2
Walking
10:09
0:01:13
1.9 Calories
78.0 Meters
Pause
Marker
Stop

Demo

Updating samples using sync identifiers

Sample Source Information

Understanding the context and fidelity of HealthKit data

New Properties on HKSourceRevision

Operating system version and product type

```
open class HKSourceRevision : NSObject, NSSecureCoding, NSCopying {

    open var source: HKSource { get }
    open var version: NSString? { get }


}
```

New Properties on HKSourceRevision

Operating system version and product type

NEW

```
open class HKSourceRevision : NSObject, NSSecureCoding, NSCopying {

    open var source: HKSource { get }
    open var version: NSString? { get }
    open var productType: String? { get } // e.g. “watch2,4”

}
```

New Properties on HKSourceRevision

Operating system version and product type

NEW

```
open class HKSourceRevision : NSObject, NSSecureCoding, NSCopying {

    open var source: HKSource { get }
    open var version: NSString? { get }
    open var productType: String? { get } // e.g. “watch2,4”
    open var operatingSystemVersion: OperatingSystemVersion { get } // e.g. {4, 0, 0}
}
```

New Properties on HKSourceRevision

Operating system version and product type

NEW

```
open class HKSourceRevision : NSObject, NSSecureCoding, NSCopying {

    open var source: HKSource { get }
    open var version: NSString? { get }
    open var productType: String? { get } // e.g. “watch2,4”
    open var operatingSystemVersion: OperatingSystemVersion { get } // e.g. {4, 0, 0}
}

public let HKSourceRevisionAnyVersion: String
public let HKSourceRevisionAnyProductType: String
public let HKSourceRevisionAnyOperatingSystem: OperatingSystemVersion
```

Supporting Diabetes Management

Supporting Diabetes Management

NEW

What’s New in Core Bluetooth | Grand Ballroom B | Thursday 11:00AM

Supporting Diabetes Management

NEW

Blood glucose meal time

What’s New in Core Bluetooth | Grand Ballroom B | Thursday 11:00AM

Supporting Diabetes Management

NEW

Blood glucose meal time

Insulin support

Supporting Diabetes Management

NEW

Blood glucose meal time

Insulin support

CoreBluetooth in watchOS 4

What's New in Core Bluetooth — Grand Ballroom B — Thursday 11:00AM

Supporting Diabetes Management

Blood glucose meal time

NEW

Supporting Diabetes Management

Blood glucose meal time

NEW

```
public let HKMetadataKeyBloodGlucoseMealTime: String
```

Supporting Diabetes Management

Blood glucose meal time

NEW

```
public let HKMetadataKeyBloodGlucoseMealTime: String

public enum HKBloodGlucoseMealTime: Int {
    case preprandial
    case postprandial
}
```

Supporting Diabetes Management

Blood glucose meal time

NEW

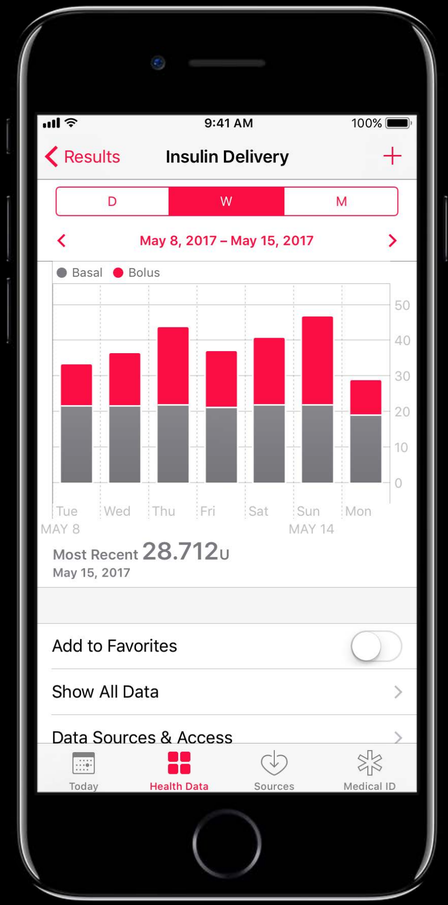
```
public let HKMetadataKeyBloodGlucoseMealTime: String

public enum HKBloodGlucoseMealTime: Int {
    case preprandial
    case postprandial
}
```

Time relative to a meal

Supporting Diabetes Management

Insulin delivery

NEW



Supporting Diabetes Management

Insulin delivery

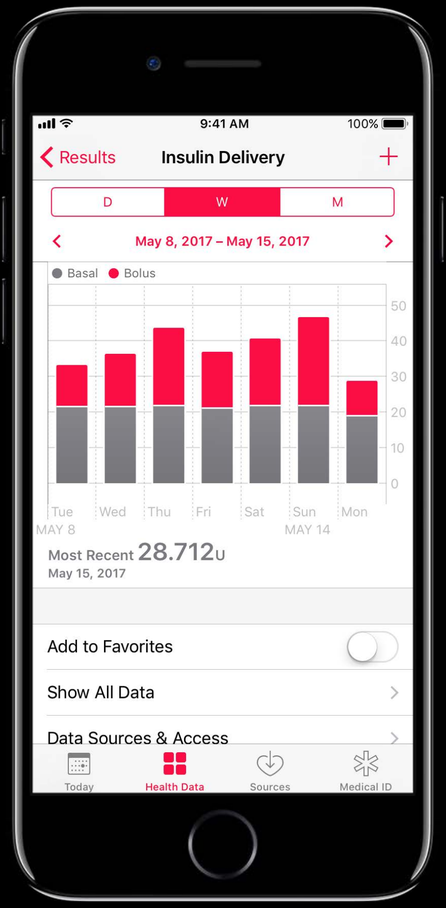
NEW

Supporting Diabetes Management

Insulin delivery

NEW

```
public static let insulinDelivery: HKQuantityTypeIdentifier
```



Supporting Diabetes Management

Insulin delivery

NEW

```
public static let insulinDelivery: HKQuantityTypeIdentifier

public let HKMetadataKeyInsulinDeliveryReason: String
```

Supporting Diabetes Management

Insulin delivery

NEW

```
public static let insulinDelivery: HKQuantityTypeIdentifier

public let HKMetadataKeyInsulinDeliveryReason: String

public enum HKInsulinDeliveryReason : Int {
    case basal
    case bolus
}
```

Supporting Diabetes Management

Insulin delivery

NEW

```
public static let insulinDelivery: HKQuantityTypeIdentifier

public let HKMetadataKeyInsulinDeliveryReason: String

public enum HKInsulinDeliveryReason : Int {
    case basal
    case bolus
}
```

Insulin that has been delivered

Supporting Diabetes Management

Insulin delivery

NEW

```
public static let insulinDelivery: HKQuantityTypeIdentifier

public let HKMetadataKeyInsulinDeliveryReason: String

public enum HKInsulinDeliveryReason : Int {
    case basal
    case bolus
}
```

Insulin that has been delivered

International unit

Supporting Diabetes Management

International unit

NEW

```
extension HKUnit {
    open class func internationalUnit() -> Self
}
```

Supporting Diabetes Management

International unit

NEW

```
extension HKUnit {
    open class func internationalUnit() -> Self
}
```

Biological effectiveness

Supporting Diabetes Management

International unit

NEW

```
extension HKUnit {
    open class func internationalUnit() -> Self
}
```

Biological effectiveness

Cannot be converted to other units

```
// Add Basal Insulin Sample From an Insulin Pump
```

```swift
// Add Basal Insulin Sample From an Insulin Pump

// Step 1: Create an insulin delivery quantity type
let quantityType = HKQuantityType.quantityType(forIdentifier: .insulinDelivery)!
```

```swift
// Add Basal Insulin Sample From an Insulin Pump

// Step 1: Create an insulin delivery quantity type
let quantityType = HKQuantityType.quantityType(forIdentifier: .insulinDelivery)!

// Step 2: Create a quantity of 0.825 units
let quantity = HKQuantity(unit: .internationalUnit(), doubleValue: 0.825)
```

```
// Add Basal Insulin Sample From an Insulin Pump

// Step 3: Create a quantity sample
let insulinSample = HKQuantitySample(
    type: quantityType,
    quantity: quantity,
    start: pumpDeliveryStartDate,
    end: pumpDeliveryEndDate,
    metadata: [

    ]
)
```

```
// Add Basal Insulin Sample From an Insulin Pump

// Step 3: Create a quantity sample
let insulinSample = HKQuantitySample(
    type: quantityType,
    quantity: quantity,
    start: pumpDeliveryStartDate,
    end: pumpDeliveryEndDate,
    metadata: [
        HKMetadataKeyInsulinDeliveryReason: HKInsulinDeliveryReason.basal.rawValue
    ]
)
```

```
// Add Basal Insulin Sample From an Insulin Pump

// Step 3: Create a quantity sample
let insulinSample = HKQuantitySample(
    type: quantityType,
    quantity: quantity,
    start: pumpDeliveryStartDate,
    end: pumpDeliveryEndDate,
    metadata: [
        HKMetadataKeyInsulinDeliveryReason: HKInsulinDeliveryReason.basal.rawValue
    ]
)

// Step 4: Save the new sample
healthStore.save(insulinSample) { success, error in }
```

```
// Statistics Query for Basal Samples
```

```
// Statistics Query for Basal Samples

// Step 1: Setup the query
let predicate = HKQuery.predicateForObjects(withMetadataKey: HKMetadataKeyInsulinDeliveryReason,
                                            allowedValues:
                                            [HKInsulinDeliveryReason.basal.rawValue])
```

```
// Statistics Query for Basal Samples

// Step 1: Setup the query
let predicate = HKQuery.predicateForObjects(withMetadataKey: HKMetadataKeyInsulinDeliveryReason,
                                            allowedValues:
                                            [HKInsulinDeliveryReason.basal.rawValue])
let quantityType = HKQuantityType.quantityType(forIdentifier: .insulinDelivery)!
```

```
// Statistics Query for Basal Samples

// Step 1: Setup the query
let predicate = HKQuery.predicateForObjects(withMetadataKey: HKMetadataKeyInsulinDeliveryReason,
                                            allowedValues:
                                            [HKInsulinDeliveryReason.basal.rawValue])
let quantityType = HKQuantityType.quantityType(forIdentifier: .insulinDelivery)!

let query = HKStatisticsCollectionQuery(quantityType: quantityType,
                                        quantitySamplePredicate: predicate,
                                        options:
                                        anchorDate:
                                        intervalComponents: )
```

```
// Statistics Query for Basal Samples

// Step 1: Setup the query
let predicate = HKQuery.predicateForObjects(withMetadataKey: HKMetadataKeyInsulinDeliveryReason,
                                            allowedValues:
                                            [HKInsulinDeliveryReason.basal.rawValue])
let quantityType = HKQuantityType.quantityType(forIdentifier: .insulinDelivery)!

let query = HKStatisticsCollectionQuery(quantityType: quantityType,
                                        quantitySamplePredicate: predicate,
                                        options: [.cumulativeSum, .separateBySource],
                                        anchorDate:
                                        intervalComponents: )
```

```
// Statistics Query for Basal Samples

// Step 1: Setup the query
let predicate = HKQuery.predicateForObjects(withMetadataKey: HKMetadataKeyInsulinDeliveryReason,
                                            allowedValues:
                                            [HKInsulinDeliveryReason.basal.rawValue])
let quantityType = HKQuantityType.quantityType(forIdentifier: .insulinDelivery)!

let query = HKStatisticsCollectionQuery(quantityType: quantityType,
                                        quantitySamplePredicate: predicate,
                                        options: [.cumulativeSum, .separateBySource],
                                        anchorDate: Date.distantPast,
                                        intervalComponents: )
```

```
// Statistics Query for Basal Samples

// Step 1: Setup the query
let predicate = HKQuery.predicateForObjects(withMetadataKey: HKMetadataKeyInsulinDeliveryReason,
                                            allowedValues:
                                            [HKInsulinDeliveryReason.basal.rawValue])
let quantityType = HKQuantityType.quantityType(forIdentifier: .insulinDelivery)!

let query = HKStatisticsCollectionQuery(quantityType: quantityType,
                                        quantitySamplePredicate: predicate,
                                        options: [.cumulativeSum, .separateBySource],
                                        anchorDate: Date.distantPast,
                                        intervalComponents: DateComponents(hour: 1))
```

```
// Statistics Query for Basal Samples

// Step 2: Set the results handler
query.initialResultsHandler = { query, results, error in
    // Process statistics
}
```

```
// Statistics Query for Basal Samples

// Step 2: Set the results handler
query.initialResultsHandler = { query, results, error in
    // Process statistics
}

// Step 3: Execute the query
healthStore.execute(query)
```

Summary

Expand reach with new data types

Summary

Expand reach with new data types

Build engaging workout experiences

Summary

Expand reach with new data types

Build engaging workout experiences

Prevent data duplication with sync identifiers

Summary

Expand reach with new data types

Build engaging workout experiences

Prevent data duplication with sync identifiers

Support users managing diabetes

More Information

https://developer.apple.com/wwdc17/221

Related Sessions

Creating Immersive Apps with Core Motion	Grand Ballroom B	Tuesday 4:10PM
What’s New in Core Bluetooth	Grand Ballroom B	Thursday 11:00AM
What’s New in Location Technologies	Executive Ballroom	Thursday 3:10PM
What’s New in CareKit and ResearchKit	Grand Ballroom A	Thursday 5:10PM

Labs

Health, Fitness, and Research Get-Together	Grand Ballroom A	Wed 6:30PM–7:45PM
HealthKit Lab	Technology Lab H	Thur 9:00AM–12:00PM
WatchConnectivity and WatchKit Lab	Technology Lab B	Fri 9:00AM–11:00AM
ResearchKit and CareKit Lab	Technology Lab H	Fri 11:00AM–1:00PM


WWDC17