#WWDC18

New Ways to Work with Workouts

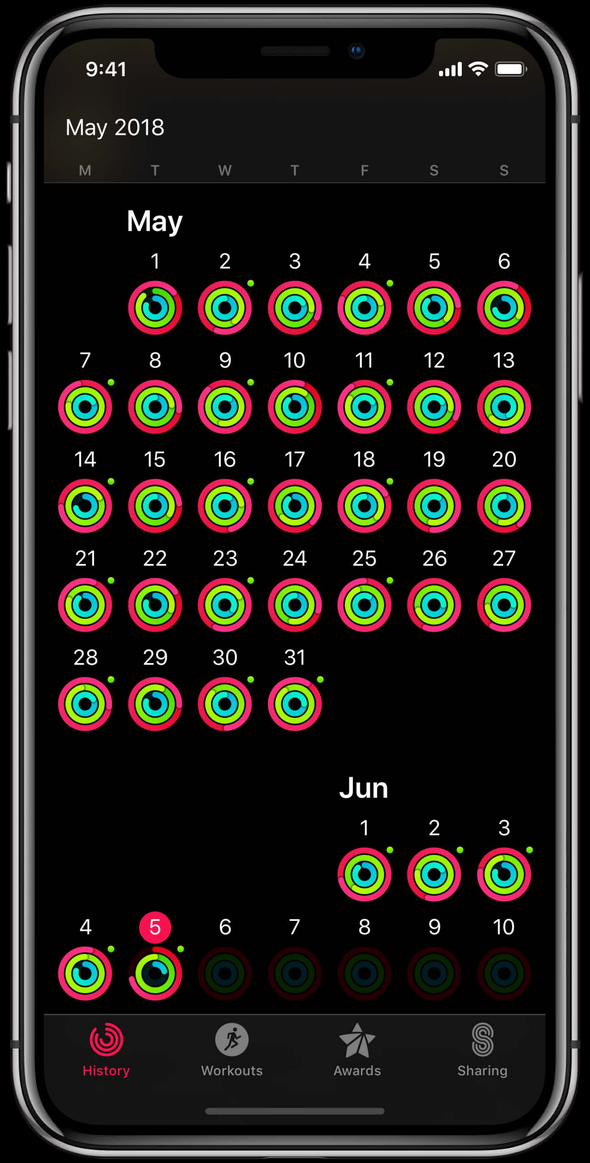
Session 707

Niharika Bedekar, Fitness Software Engineer
Karim Benhmida, Health Software Engineer

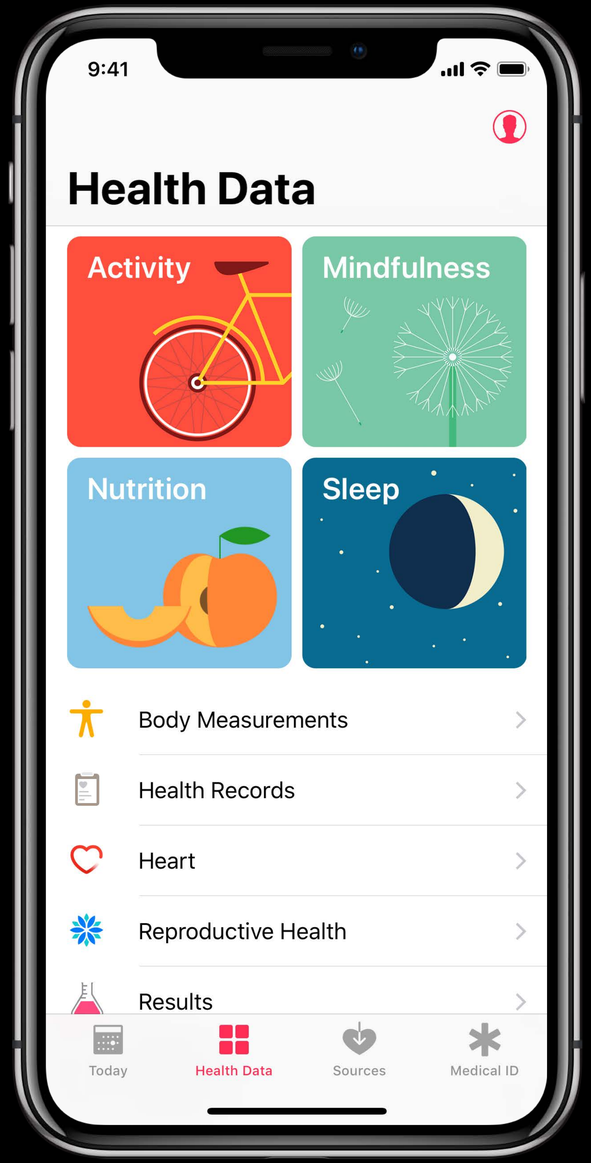

Workout
10:09
Indoor Walk
OPEN GOAL
Outdoor Run
OPEN GOAL

200 Million

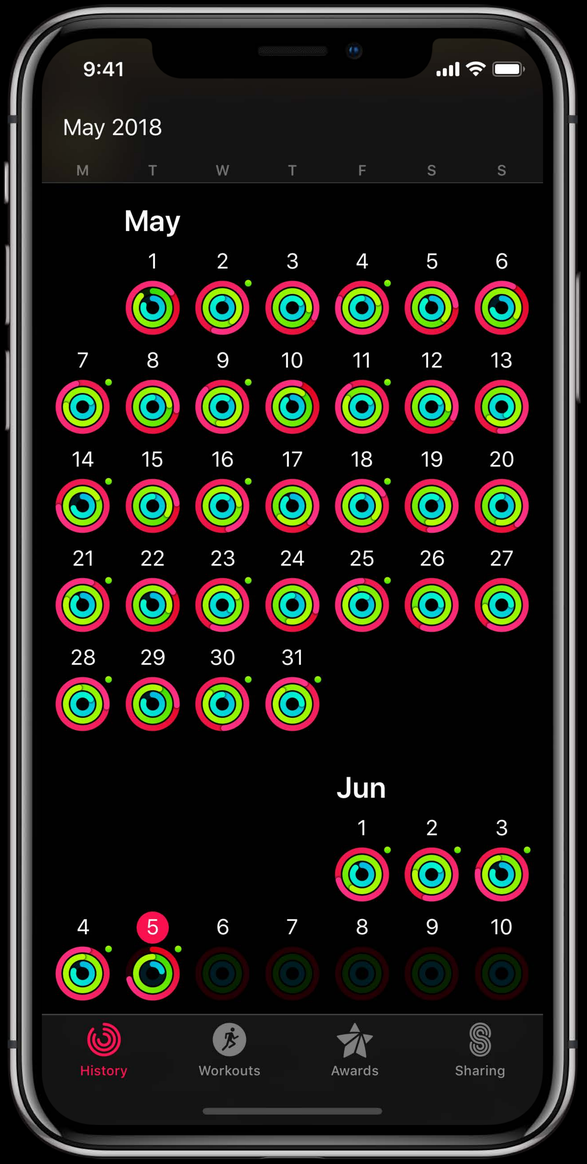
Health and Fitness App Store Category Downloads



Activity

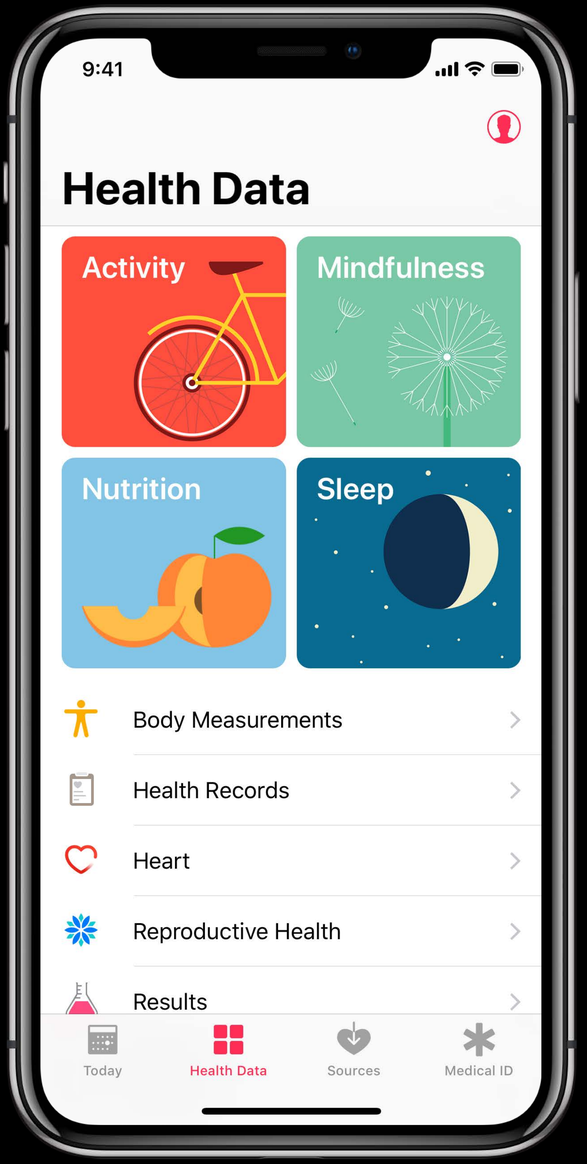


Health



Activity

HealthKit



Health

Agenda

Agenda

Privacy and Authorization

Agenda

Privacy and Authorization

New Workout API

Agenda

Privacy and Authorization

New Workout API

New Quantity Series API

Privacy and Authorization

Privacy and Authorization

Privacy and Authorization

Privacy and Authorization

Privacy and Authorization

Proportional Collection

HealthKit authorization can change

Privacy and Authorization

Ask for...

Only what you need

Only when you need it

Every time you need it

Privacy and Authorization

Privacy and Authorization

```
let typesToShare: Set = [
    HKSampleType.workoutType()
]
```

Privacy and Authorization

```
let typesToShare: Set = [
    HKSampleType.workoutType()
]
let typesToRead: Set = [
    HKQuantityType.quantityType(forIdentifier: .heartRate)!,
    HKQuantityType.quantityType(forIdentifier: .activeEnergyBurned)!,
    HKQuantityType.quantityType(forIdentifier: .distanceWalkingRunning)!
]
```

Privacy and Authorization

```
let typesToShare: Set = [
    HKSampleType.workoutType()
]
let typesToRead: Set = [
    HKQuantityType.quantityType(forIdentifier: .heartRate)!,
    HKQuantityType.quantityType(forIdentifier: .activeEnergyBurned)!,
    HKQuantityType.quantityType(forIdentifier: .distanceWalkingRunning)!
]

healthStore.requestAuthorization(toShare: typesToShare,
                                 read: typesToRead) { (success, error) in
    // Handle errors
}
```

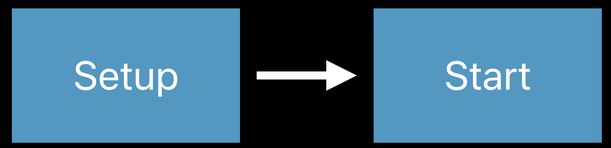
New Workout API

Karim Benhmida, Health Software Engineer

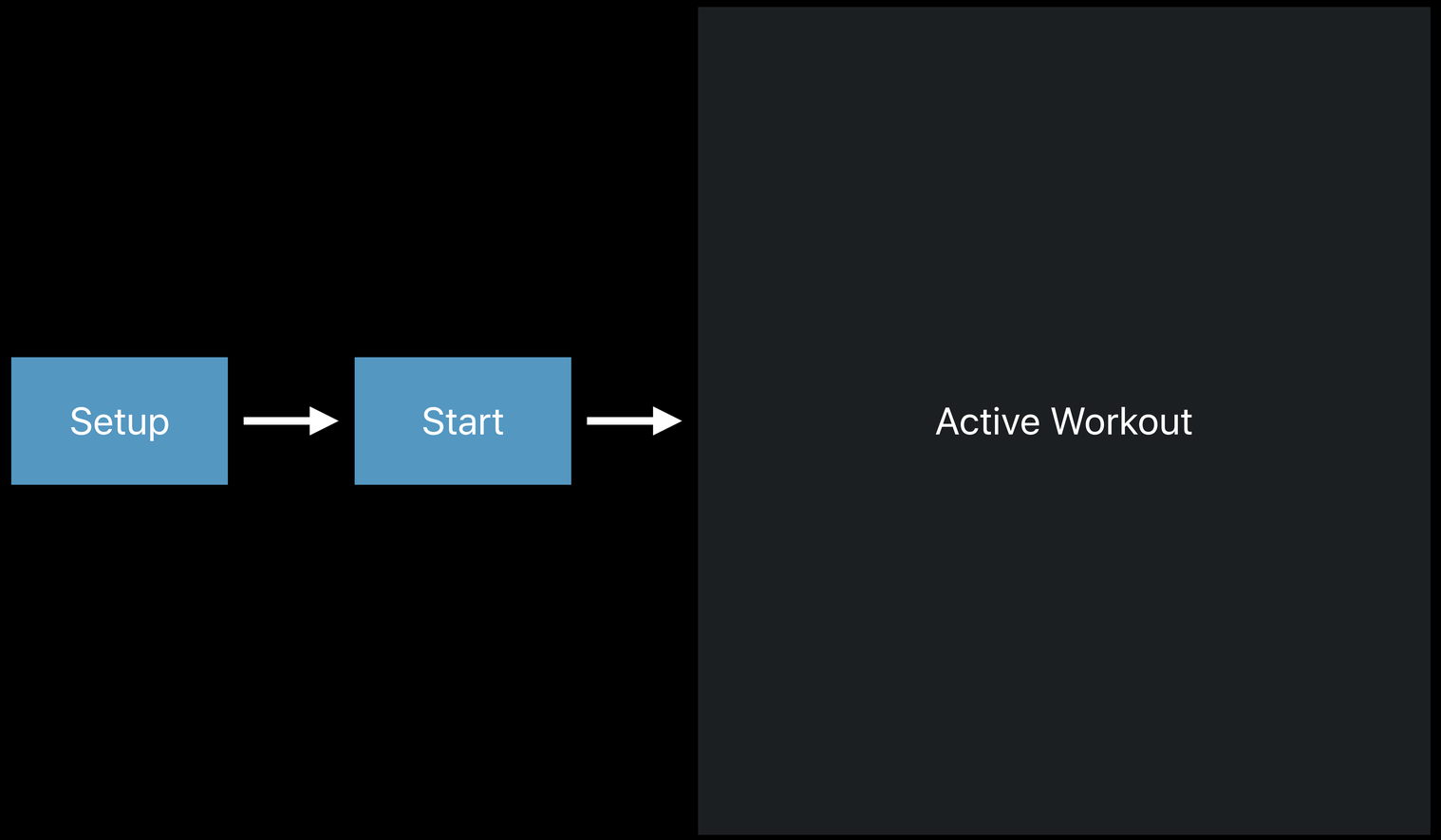
Workout App Lifecycle

Workout App Lifecycle

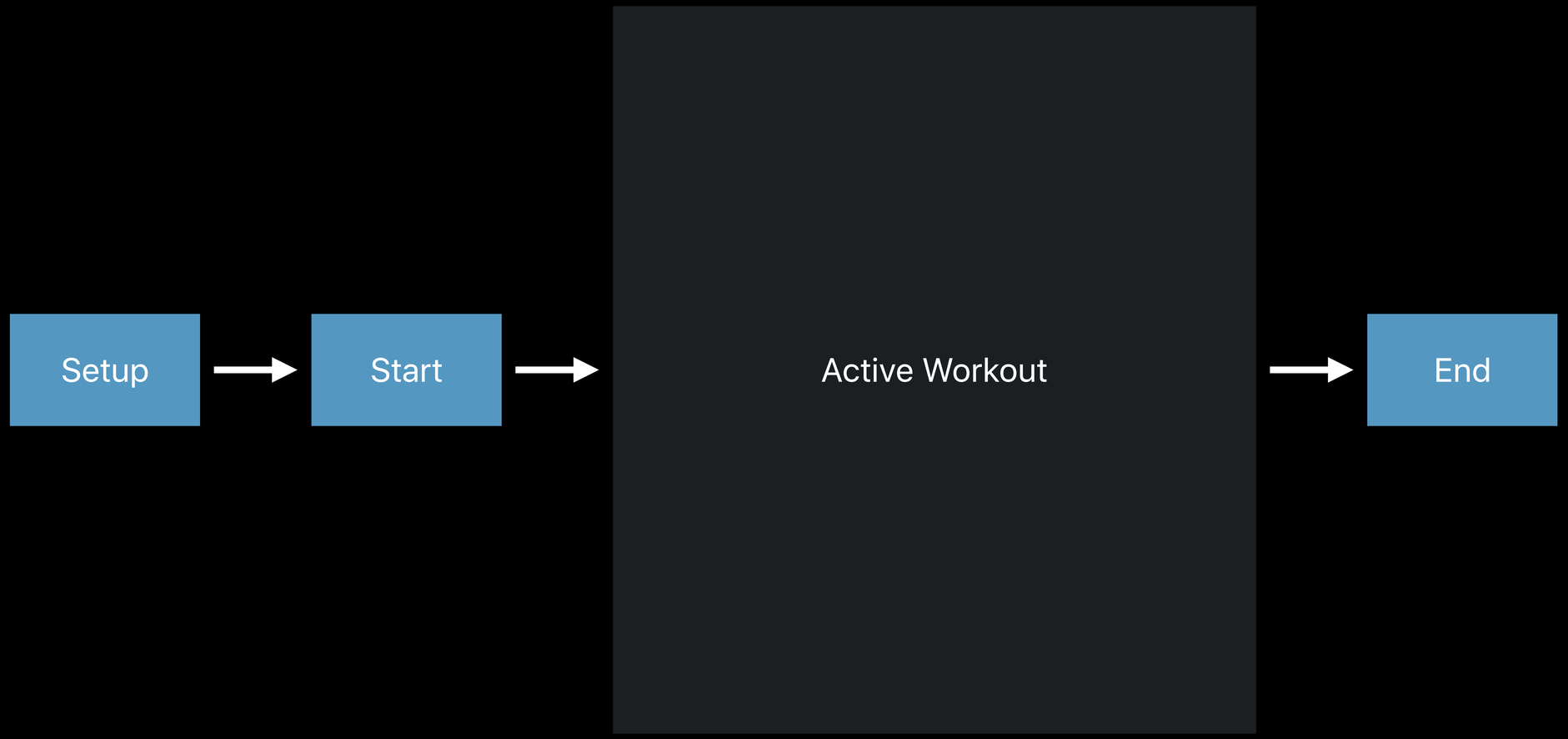
Workout App Lifecycle



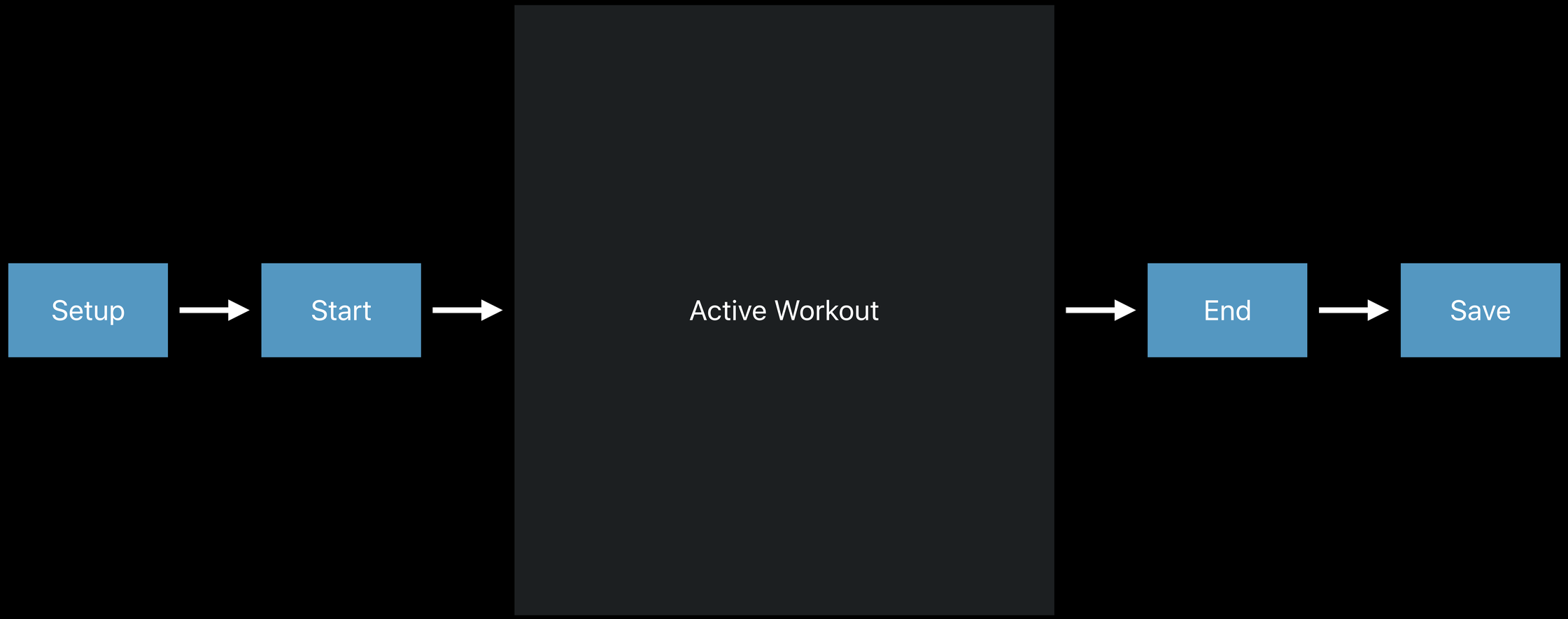
Workout App Lifecycle



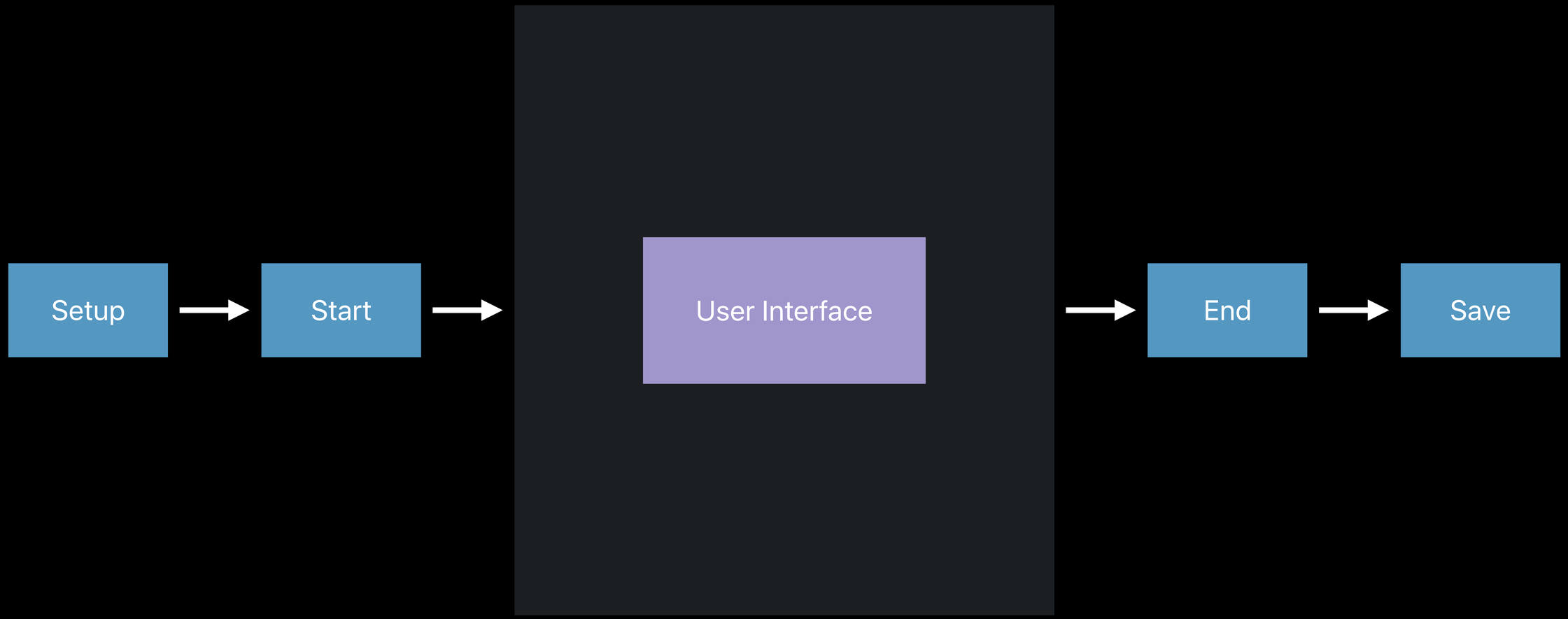
Workout App Lifecycle



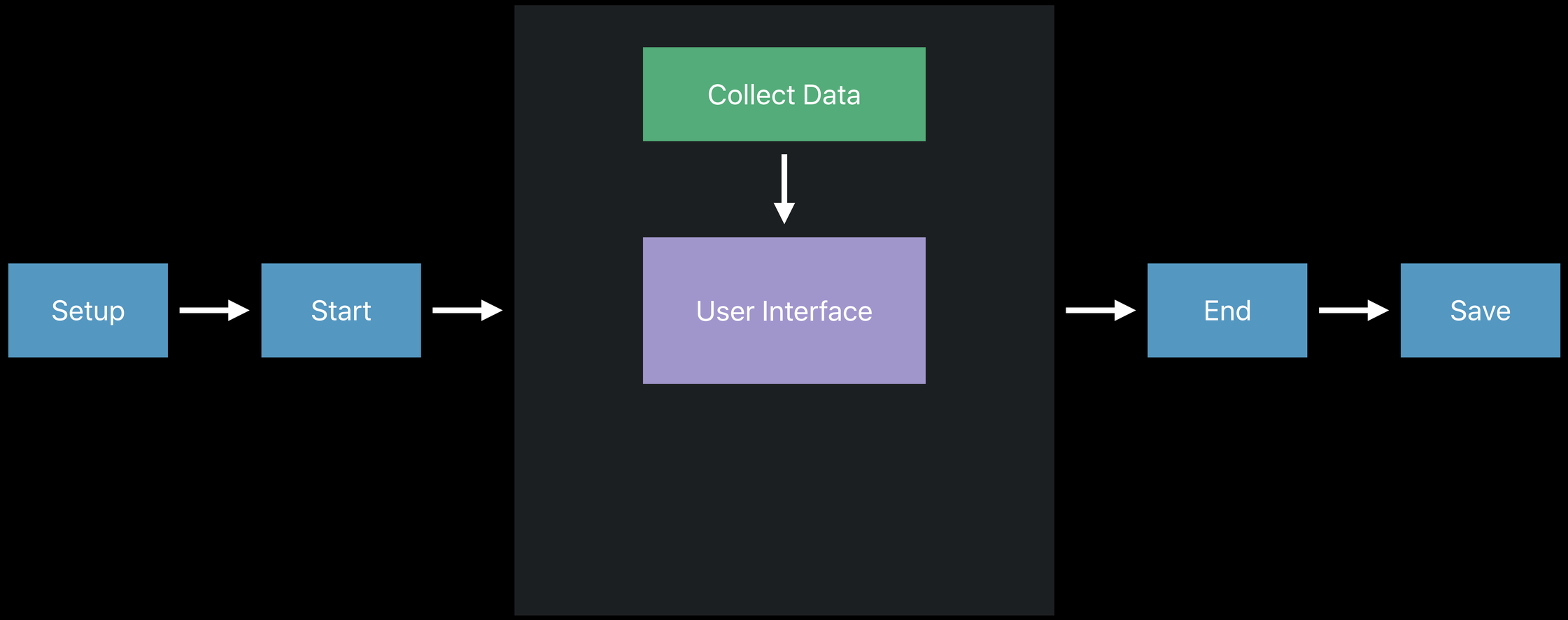
Workout App Lifecycle



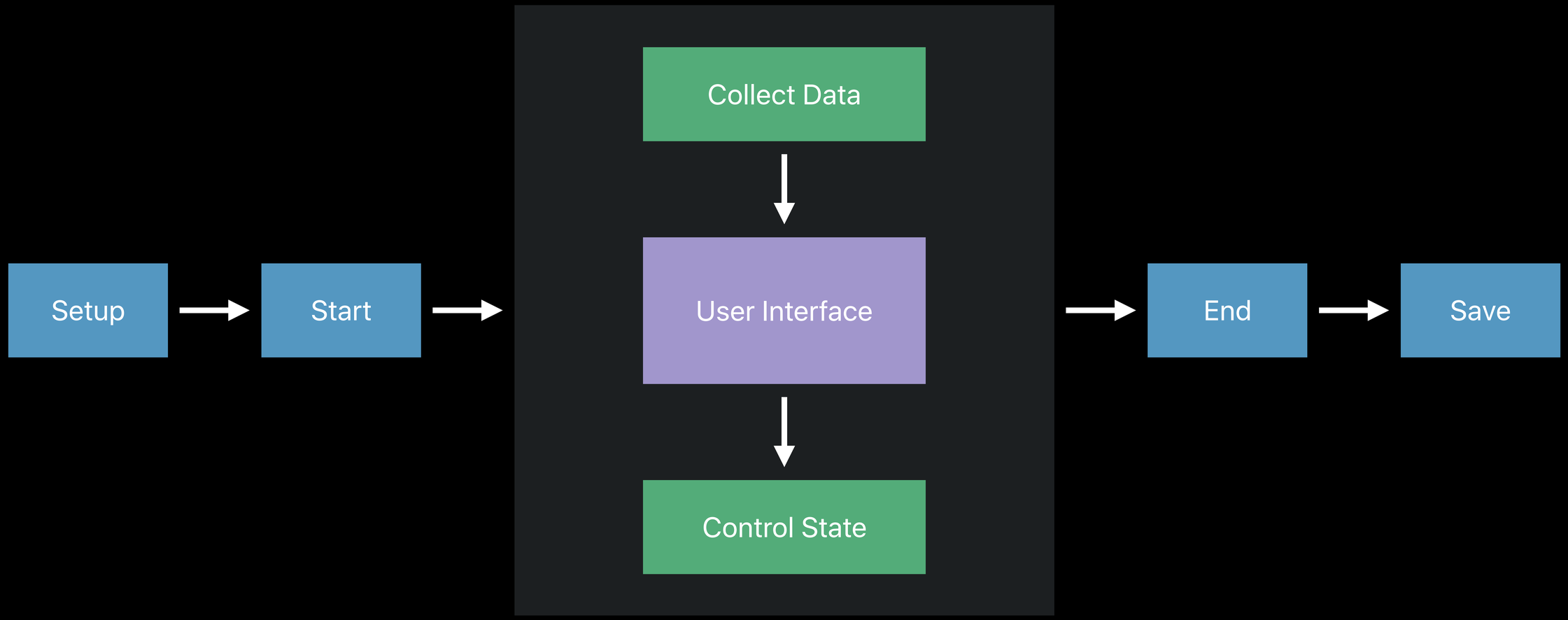
Workout App Lifecycle



Workout App Lifecycle



Workout App Lifecycle



Workout App Lifecycle

HKWorkoutSession (watchOS)

Workout App Lifecycle

HKWorkoutSession (watchOS)

Prepares sensors for data collection

Workout App Lifecycle

HKWorkoutSession (watchOS)

Prepares sensors for data collection

Allows background running

Workout App Lifecycle

HKWorkoutSession (watchOS)

Prepares sensors for data collection

Allows background running

Controls workout state

Workout App Lifecycle

HKWorkoutSession (watchOS)

Prepares sensors for data collection

Allows background running

Controls workout state

Generates events

Workout App Lifecycle

Workout App Lifecycle

Collect data generated by the device

Workout App Lifecycle

Collect data generated by the device

Save a workout

Collect Data

NEW

Collect Data

NEW

`HKWorkoutBuilder`

Collect Data

NEW

`HKWorkoutBuilder`

Creates and saves an `HKWorkout`

Collect Data

NEW

`HKWorkoutBuilder`

Creates and saves an `HKWorkout`

Add samples, events, and metadata

Collect Data

NEW

`HKWorkoutBuilder`

Creates and saves an `HKWorkout`

Add samples, events, and metadata

Collect Data

NEW

`HKWorkoutBuilder`

Creates and saves an `HKWorkout`

Add samples, events, and metadata

`HKLiveWorkoutBuilder`

Collect Data

NEW

`HKWorkoutBuilder`

Creates and saves an `HKWorkout`

Add samples, events, and metadata

`HKLiveWorkoutBuilder`

watchOS only

Collect Data

NEW

`HKWorkoutBuilder`

Creates and saves an `HKWorkout`

Add samples, events, and metadata

`HKLiveWorkoutBuilder`

watchOS only

Subclass of `HKWorkoutBuilder`

Collect Data

NEW

`HKWorkoutBuilder`

Creates and saves an `HKWorkout`

Add samples, events, and metadata

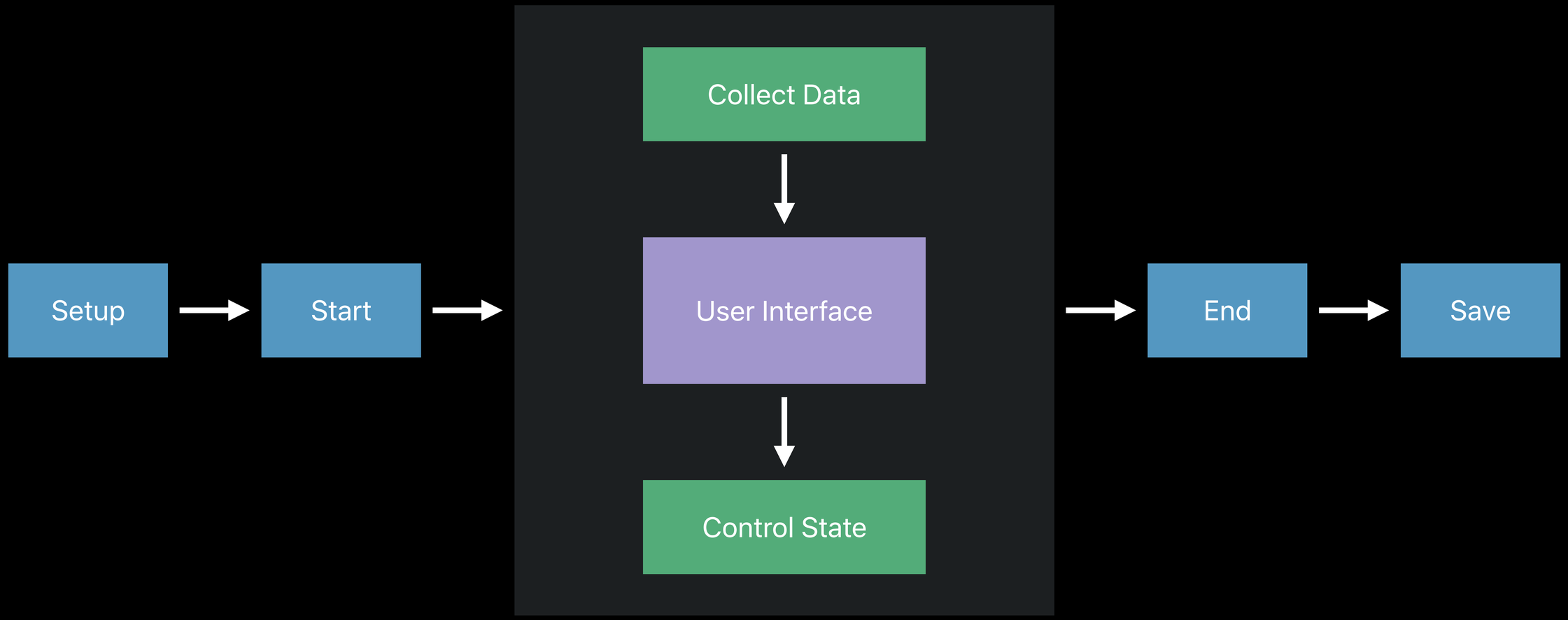
`HKLiveWorkoutBuilder`

watchOS only

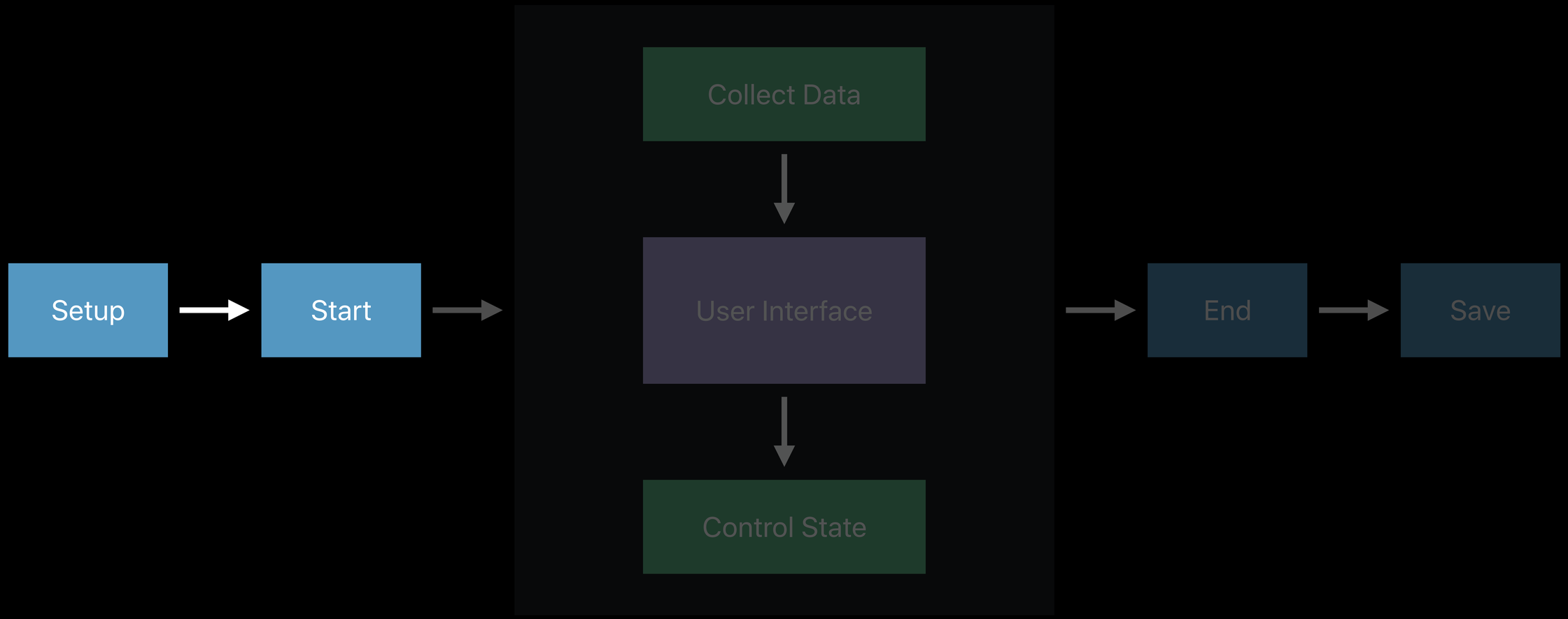
Subclass of `HKWorkoutBuilder`

Automatic sample and event collection

Workout App Lifecycle



Workout App Lifecycle



Setup and Start

HKWorkoutBuilder

NEW

Setup and Start

HKWorkoutBuilder

NEW

```
// HKWorkoutBuilder
let builder = HKWorkoutBuilder(healthStore: healthStore,
                               configuration: workoutConfiguration,
                               device: nil)
```

Setup and Start

HKWorkoutBuilder

NEW

```
// HKWorkoutBuilder
let builder = HKWorkoutBuilder(healthStore: healthStore,
                               configuration: workoutConfiguration,
                               device: nil)
// Start
builder.beginCollection(withStart: Date(), completion: { (success, error) in
    // Handle error
})
```

Setup and Start

HKLiveWorkoutBuilder (watchOS)

NEW

Setup and Start

HKLiveWorkoutBuilder (watchOS)

NEW

```
// Create session
let session = try HKWorkoutSession(healthStore: healthStore,
                                   configuration: workoutConfiguration)
```

Setup and Start

HKLiveWorkoutBuilder (watchOS)

NEW

```
// Create session
let session = try HKWorkoutSession(healthStore: healthStore,
                                   configuration: workoutConfiguration)

// Retrieve builder
let builder = session.associatedWorkoutBuilder()
```

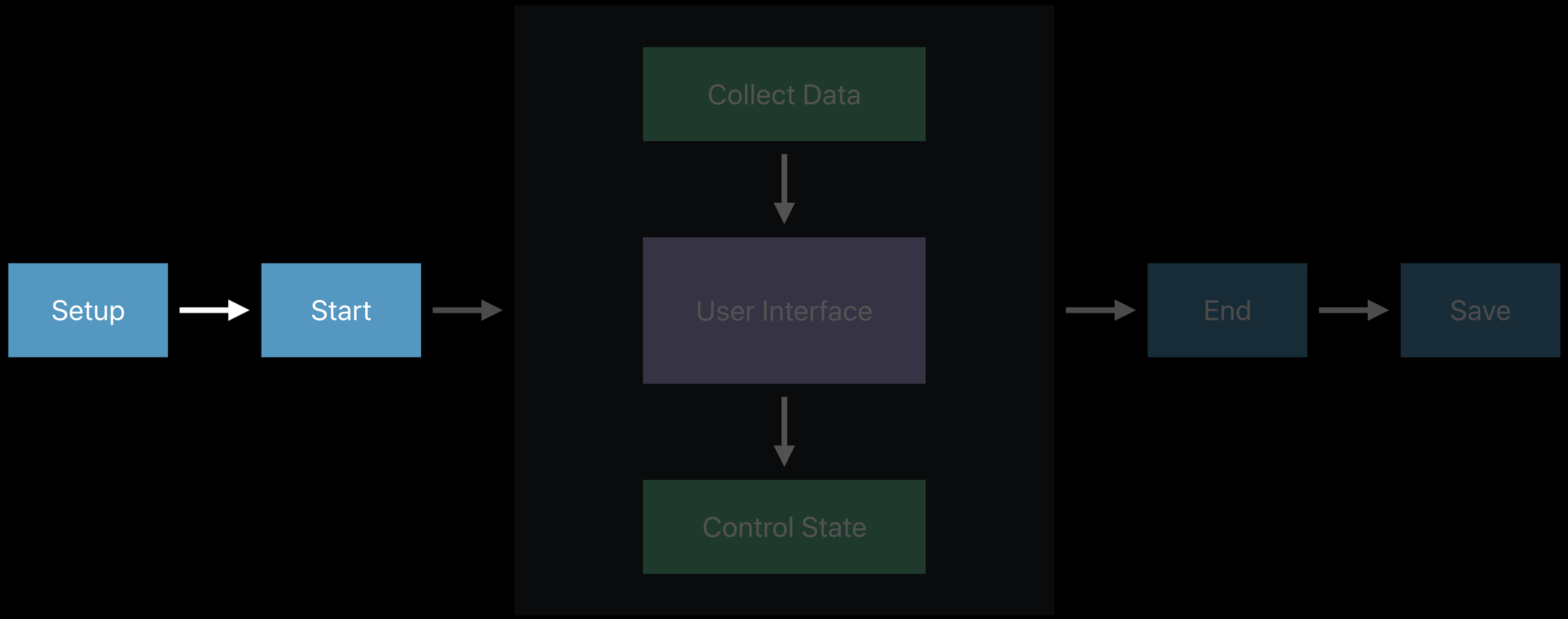
Setup and Start

HKLiveWorkoutBuilder (watchOS)

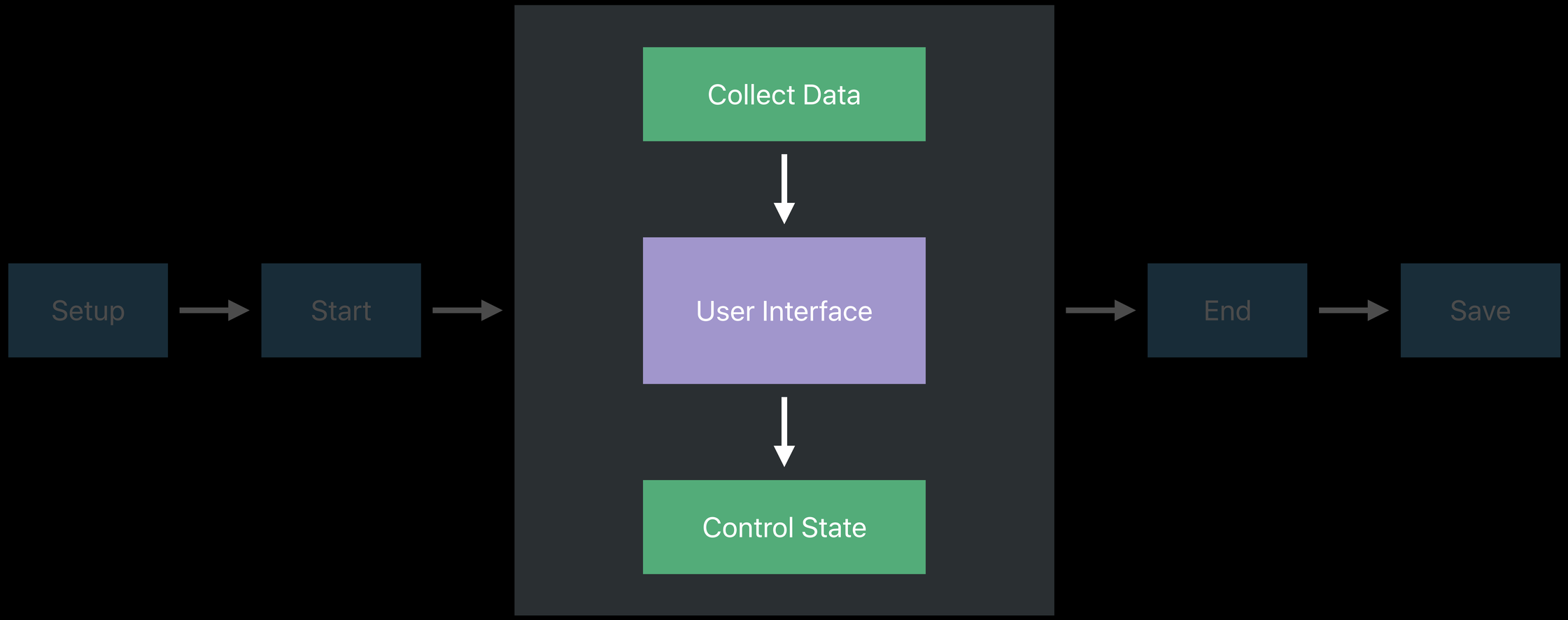
NEW

```
// Create session
let session = try HKWorkoutSession(healthStore: healthStore,
                                   configuration: workoutConfiguration)

// Retrieve builder
let builder = session.associatedWorkoutBuilder()

// Start session and builder
session.startActivity()

builder.beginCollection(withStart: Date(), completion: { (success, error) in
    // Handle error
})
```

Workout App Lifecycle



Workout App Lifecycle



Collect Data

Samples

NEW

Collect Data

Samples

NEW

```
let samples: [HKSample] = ...

builder.add(samples) { (success, error) in
    // Handle error
}
```

Collect Data

Events

NEW

Collect Data

Events

NEW

```
let events: [HKWorkoutEvent] = ...

builder.addWorkoutEvents(events) { (success, error) in
    // Handle error
}
```

Collect Data

Metadata

NEW

Collect Data

Metadata

NEW

```
let metadata: [String: Any] = ...

builder.addMetadata(metadata) { (success, error) in
    // Handle error
}
```

Collect Data

HKLiveWorkoutDataSource (watchOS)

NEW

Collect Data

HKLiveWorkoutDataSource (watchOS)

NEW

Automatically collects data specific to the workout

Collect Data

HKLiveWorkoutDataSource (watchOS)

NEW

Automatically collects data specific to the workout

Data types to collect can be customized

Collect Data

HKLiveWorkoutDataSource (watchOS)

NEW

Collect Data

HKLiveWorkoutDataSource (watchOS)

NEW

```
// Create a data source
let dataSource = HKLiveWorkoutDataSource(healthStore: healthStore,
                                         workoutConfiguration: workoutConfiguration)
```

Collect Data

HKLiveWorkoutDataSource (watchOS)

NEW

```
// Create a data source
let dataSource = HKLiveWorkoutDataSource(healthStore: healthStore,
                                         workoutConfiguration: workoutConfiguration);

// Set data source
workoutBuilder.dataSource = dataSource
```

Collect Data

HKLiveWorkoutDataSource (watchOS)

NEW

```
// Create a data source
let dataSource = HKLiveWorkoutDataSource(healthStore: healthStore,
                                         workoutConfiguration: workoutConfiguration);

// Set data source
workoutBuilder.dataSource = dataSource

// Optionally add other types
let quantityType: HKQuantityType = ...
let optionalPredicate: NSPredicate = ...

dataSource.collectStatistics(for: quantityType, predicate: optionalPredicate)
```

User Interface

Update displayed statistics

NEW

User Interface

Update displayed statistics

NEW

```
//MARK: HKLiveWorkoutBuilderDelegate
func workoutBuilder(_ workoutBuilder: HKLiveWorkoutBuilder,
                    didCollectDataOf collectedTypes: Set<HKSampleType>) {
    let heartRateType = HKQuantityType.quantityType(forIdentifier: .heartRate)!

    if collectedTypes.contains(heartRateType) {
        let updatedStatistics = workoutBuilder.statistics(for: heartRateType)
        updateHeartRateLabel(updatedStatistics)
    }
}
```

User Interface

Update displayed statistics

NEW

```
//MARK: HKLiveWorkoutBuilderDelegate
func workoutBuilder(_ workoutBuilder: HKLiveWorkoutBuilder,
                    didCollectDataOf collectedTypes: Set<HKSampleType>) {
    let heartRateType = HKQuantityType.quantityType(forIdentifier: .heartRate)!

    if collectedTypes.contains(heartRateType) {
        let updatedStatistics = workoutBuilder.statistics(for: heartRateType)
        updateHeartRateLabel(updatedStatistics)
    }
}
```

User Interface

Update displayed statistics

NEW

```
//MARK: HKLiveWorkoutBuilderDelegate
func workoutBuilder(_ workoutBuilder: HKLiveWorkoutBuilder,
                    didCollectDataOf collectedTypes: Set<HKSampleType>) {
    let heartRateType = HKQuantityType.quantityType(forIdentifier: .heartRate)!

    if collectedTypes.contains(heartRateType) {
        let updatedStatistics = workoutBuilder.statistics(for: heartRateType)
        updateHeartRateLabel(updatedStatistics)
    }
}
```

User Interface

Update displayed statistics

NEW

```
//MARK: HKLiveWorkoutBuilderDelegate
func workoutBuilder(_ workoutBuilder: HKLiveWorkoutBuilder,
                    didCollectDataOf collectedTypes: Set<HKSampleType>) {
    let heartRateType = HKQuantityType.quantityType(forIdentifier: .heartRate)!

    if collectedTypes.contains(heartRateType) {
        let updatedStatistics = workoutBuilder.statistics(for: heartRateType)
        updateHeartRateLabel(updatedStatistics)
    }
}
```

User Interface

Track elapsed time

NEW

User Interface

Track elapsed time

NEW

```
//MARK: HKLiveWorkoutBuilderDelegate
func workoutBuilderDidCollectEvents(_ workoutBuilder: HKLiveWorkoutBuilder) {
    let elapsedTime = builder.elapsedTime
    updateTimer(with: elapsedTime)
}
```

User Interface

Track elapsed time

NEW

```
//MARK: HKLiveWorkoutBuilderDelegate
func workoutBuilderDidCollectEvents(_ workoutBuilder: HKLiveWorkoutBuilder) {
    let elapsedTime = builder.elapsedTime
    updateTimer(with: elapsedTime)
}
```

Control State

HKWorkoutSession (watchOS)

NEW

Control State

HKWorkoutSession (watchOS)

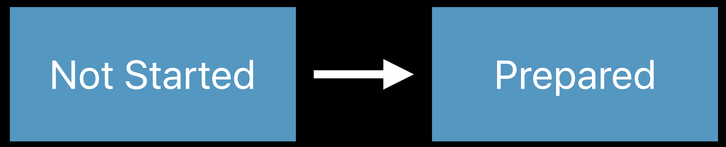
NEW

Not Started

Control State

HKWorkoutSession (watchOS)

NEW



Control State

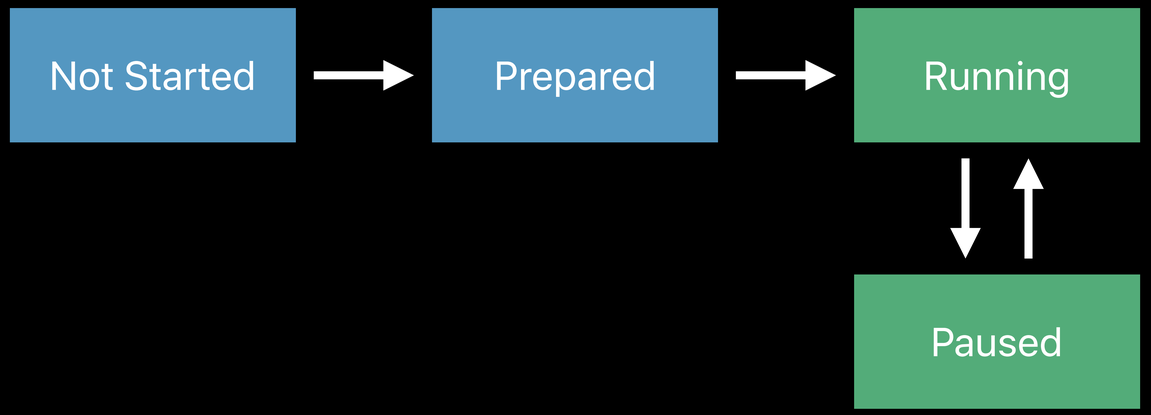
HKWorkoutSession (watchOS)

NEW

Control State

HKWorkoutSession (watchOS)

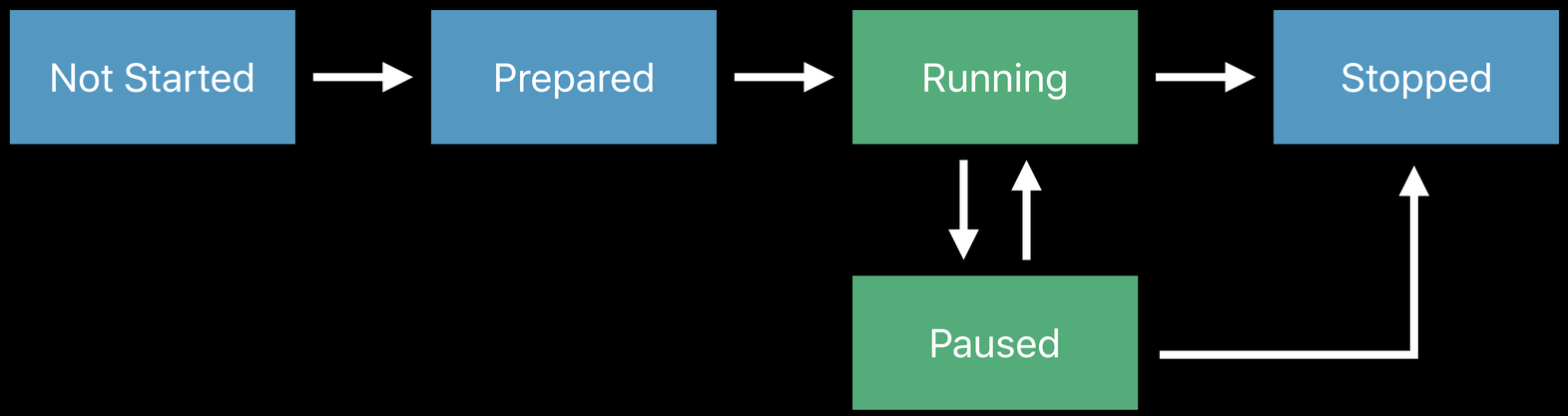
NEW



Control State

HKWorkoutSession (watchOS)

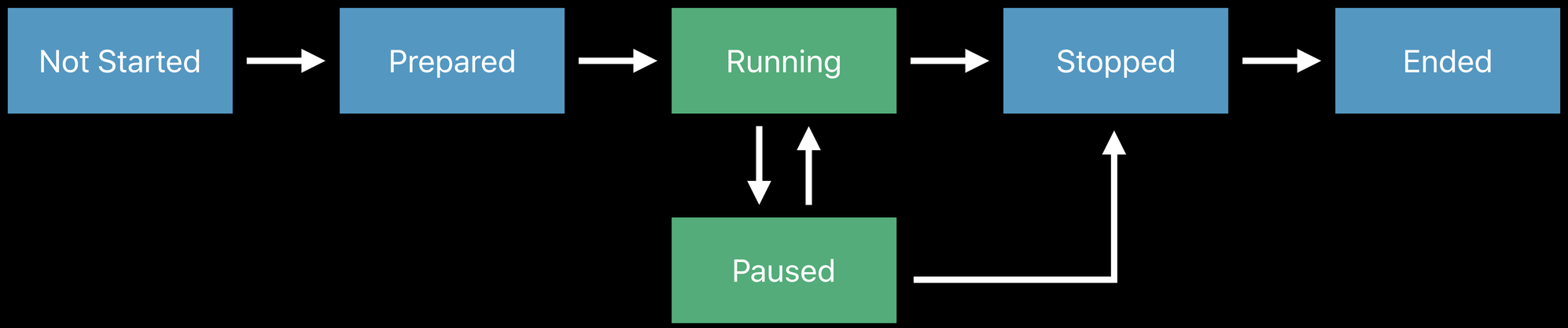
NEW



Control State

HKWorkoutSession (watchOS)

NEW



Control State

HKWorkoutSession (watchOS)

NEW

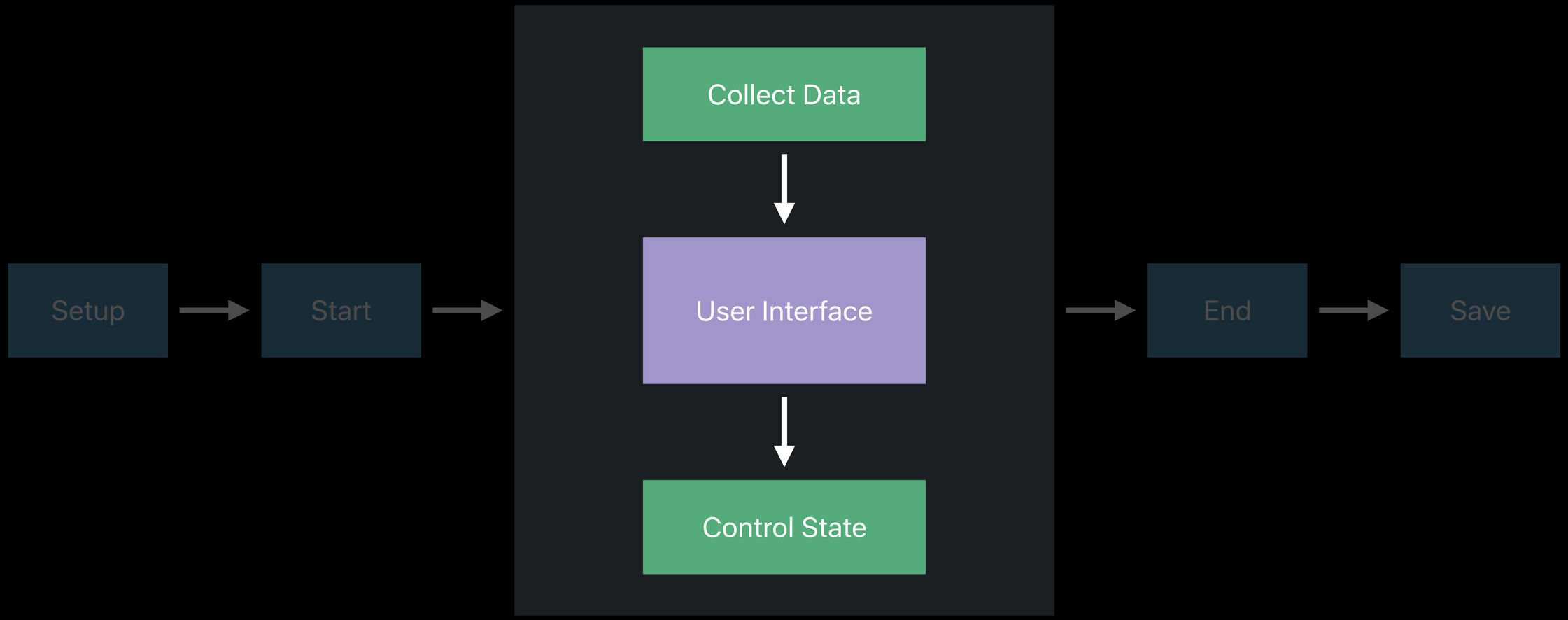
Control State

HKWorkoutSession (watchOS)

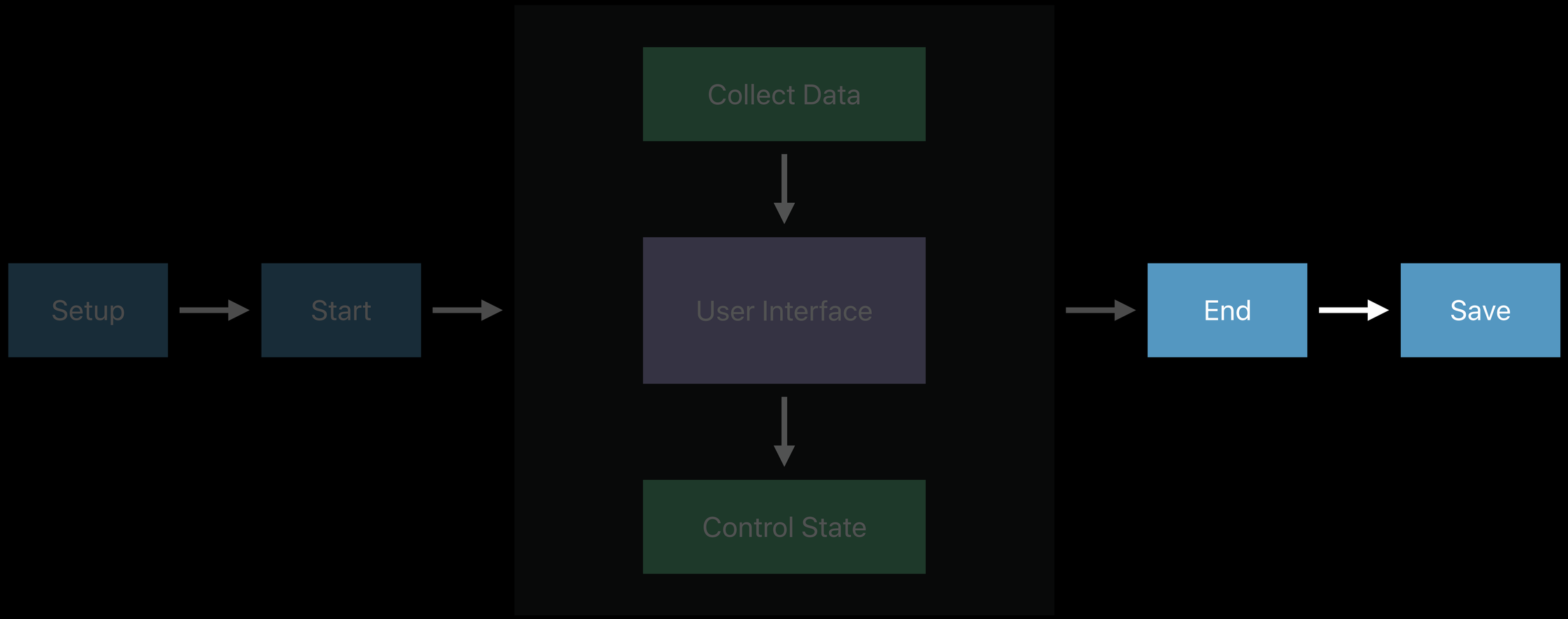
NEW

```
session.prepare()
session.startActivity()
session.pause()
session.resume()
session.stopActivity()
session.end()
```

Workout App Lifecycle



Workout App Lifecycle



End a Workout

NEW

End a Workout

NEW

```
// End session
session.end()
```

End a Workout

NEW

```
// End session
session.end()

// End builder
builder.endCollection(withEnd: Date(), completion: { (success, error) in
    // Handle error
})
```

Save a Workout

NEW

Save a Workout

NEW

```
// Save
builder.finishWorkout { (workout, error) in
    // Handle error
})
```

Demo

Building a Workout App for Apple Watch

Workout Recovery (watchOS)

NEW

Workout Recovery (watchOS)

NEW

App automatically relaunched if it crashes during a workout

Workout Recovery (watchOS)

NEW

App automatically relaunched if it crashes during a workout

Session and builder restored in their previous state

Workout Recovery (watchOS)

NEW

App automatically relaunched if it crashes during a workout

Session and builder restored in their previous state

Data source must be setup again

Workout Recovery (watchOS)

NEW

Workout Recovery (watchOS)

NEW

```
class ExtensionDelegate: NSObject, WKExtensionDelegate {

    func handleActiveWorkoutRecovery() {
        let healthStore = HKHealthStore()
        healthStore.recoverActiveWorkoutSession { (session, error) in
            // Handle error
        }
    }

}
```

Workout Recovery (watchOS)

NEW

```
class ExtensionDelegate: NSObject, WKExtensionDelegate {

    func handleActiveWorkoutRecovery() {
        let healthStore = HKHealthStore()
        healthStore.recoverActiveWorkoutSession { (session, error) in
            // Handle error
        }
    }

}
```

Workout Recovery (watchOS)

NEW

```
class ExtensionDelegate: NSObject, WKExtensionDelegate {

    func handleActiveWorkoutRecovery() {
        let healthStore = HKHealthStore()
        healthStore.recoverActiveWorkoutSession { (session, error) in
            // Handle error
        }
    }

}
```

Workout Recovery (watchOS)

NEW

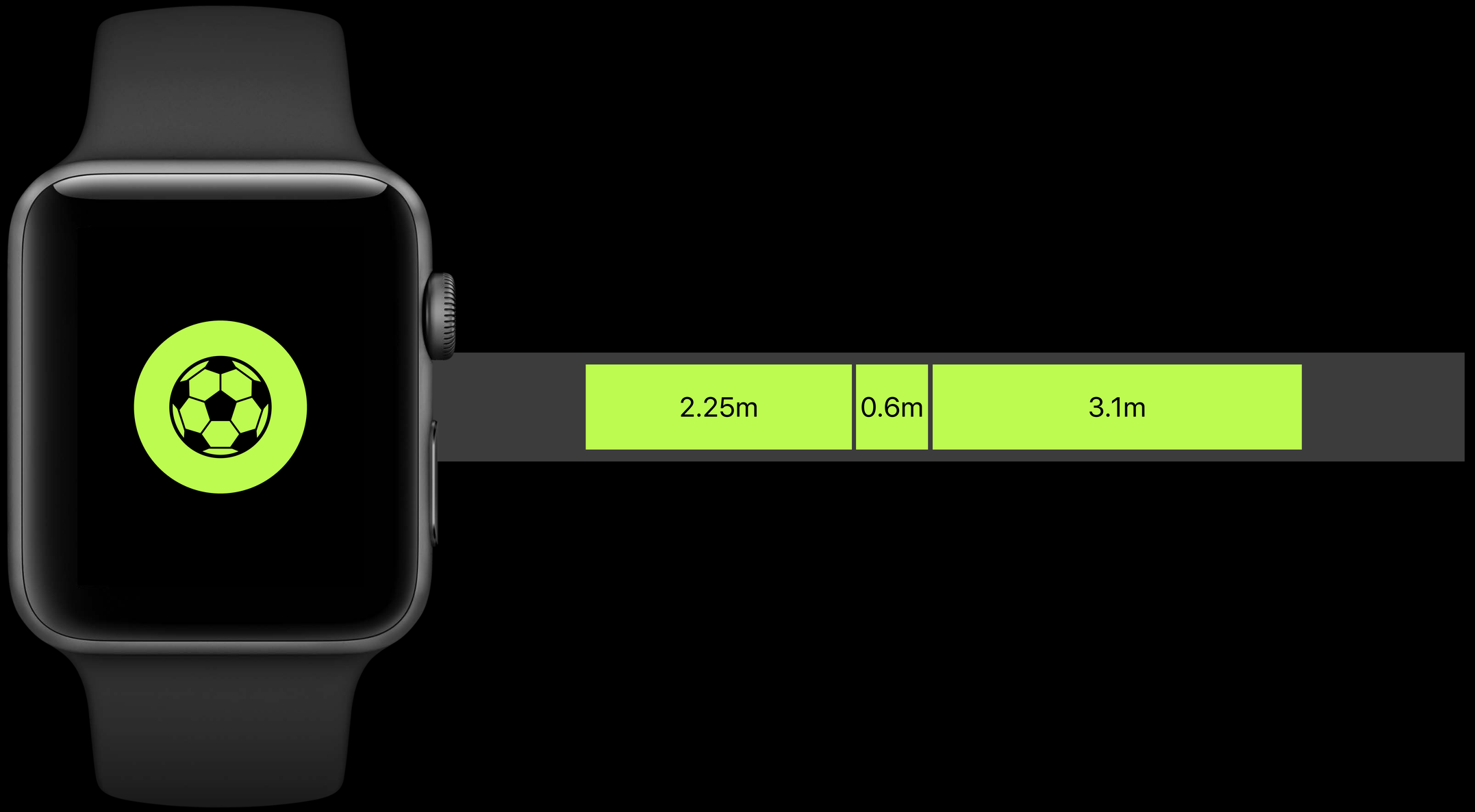
```
class ExtensionDelegate: NSObject, WKExtensionDelegate {

    func handleActiveWorkoutRecovery() {
        let healthStore = HKHealthStore()
        healthStore.recoverActiveWorkoutSession { (session, error) in
            // Handle error
        }
    }

}
```

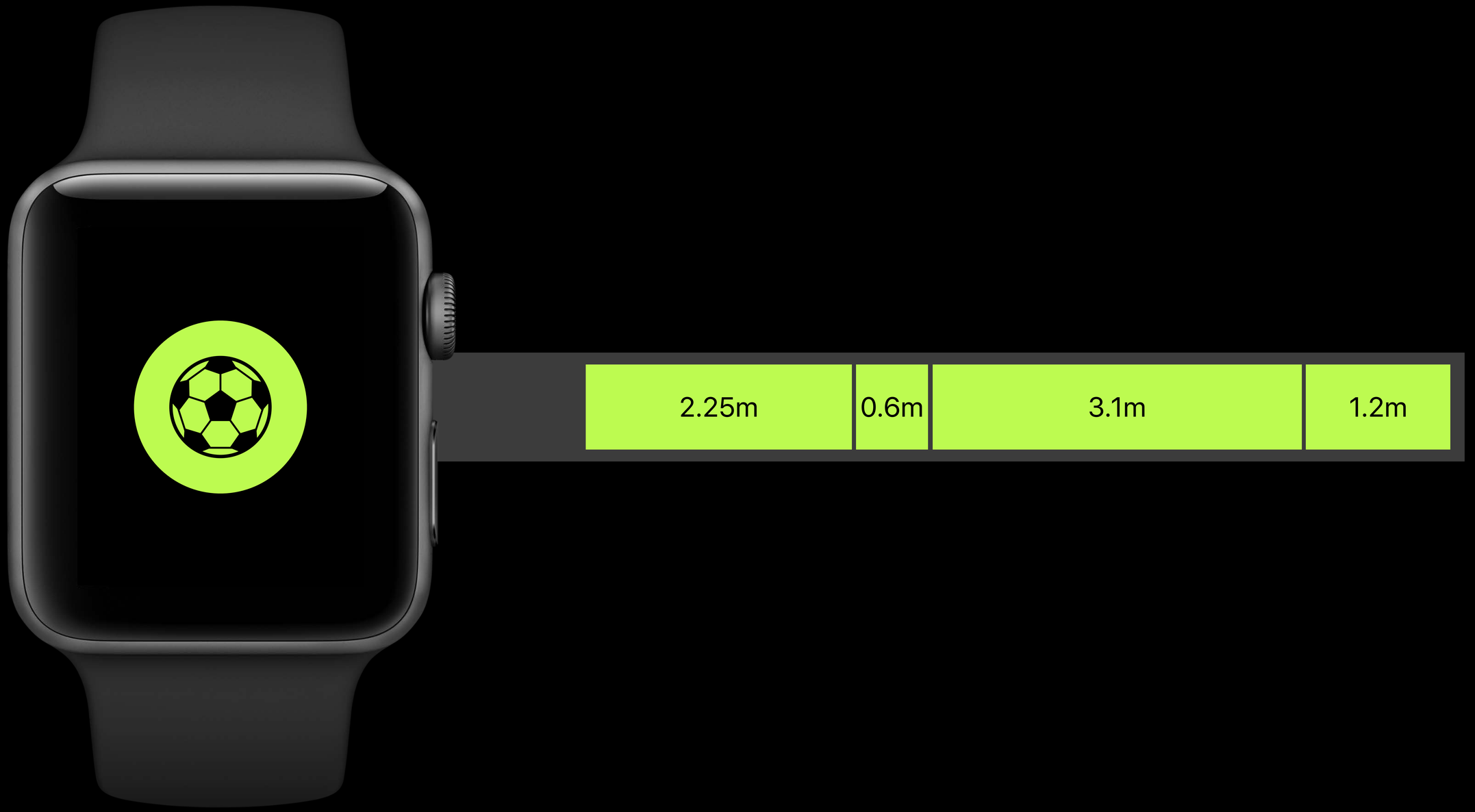
New Quantity Series API

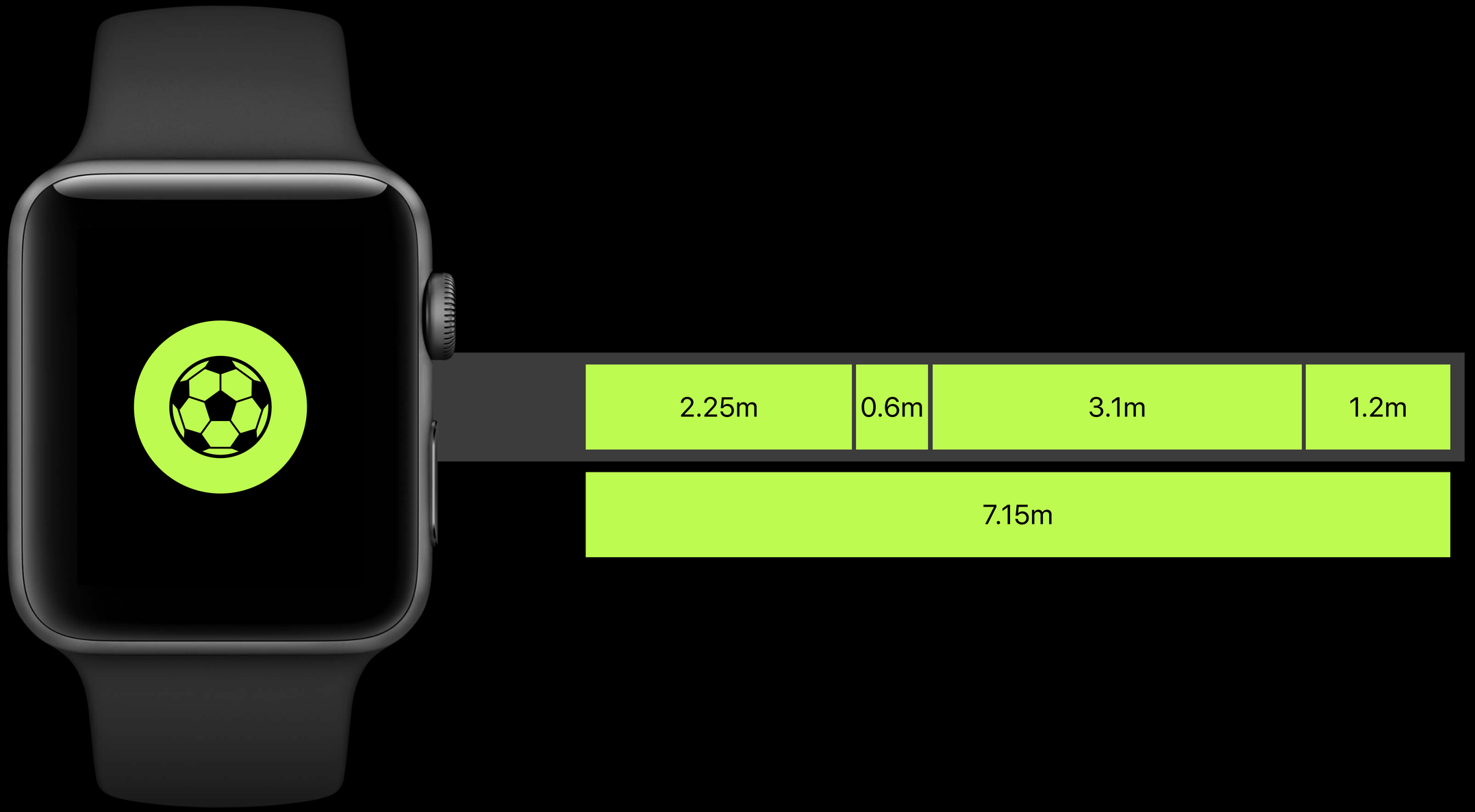
Niharika Bedekar, Fitness Software Engineer


2.25m


2.25m
0.6m


2.25m
0.6m
3.1m


2.25m
0.6m
3.1m
1.2m


2.25m
0.6m
3.1m
1.2m
7.15m

NEW

HKCumulativeQuantitySeriesSample

NEW

HKQuantitySample

HKCumulativeQuantitySeriesSample

Quantity Series API

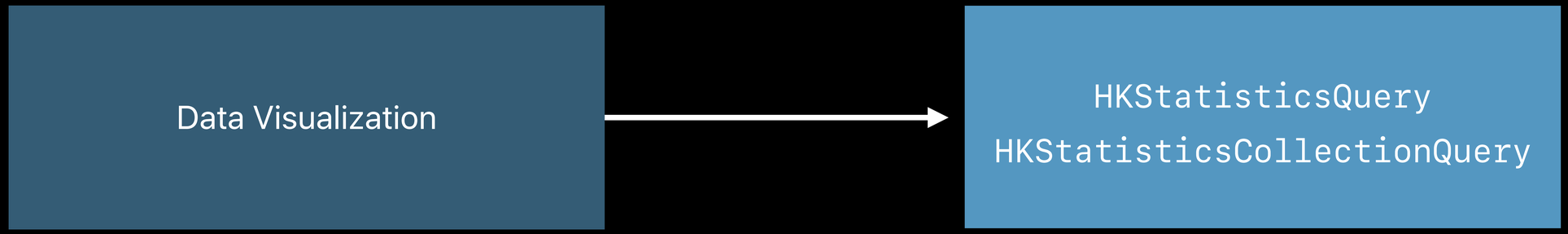
Use cases

Quantity Series API

Use cases

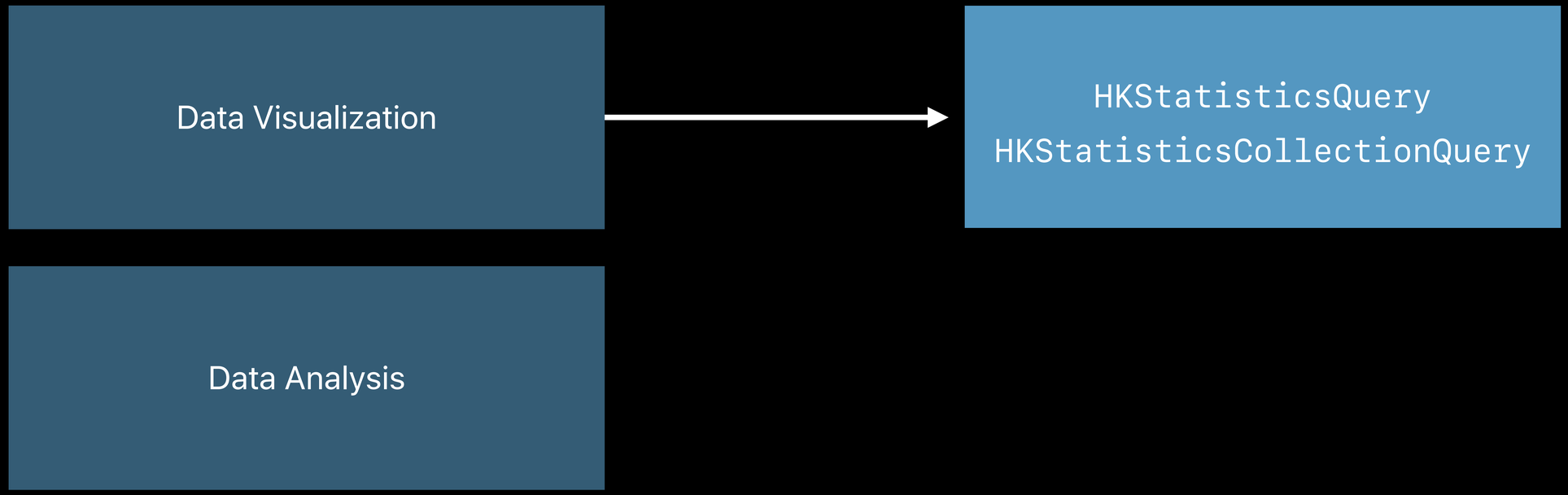
Quantity Series API

Use cases



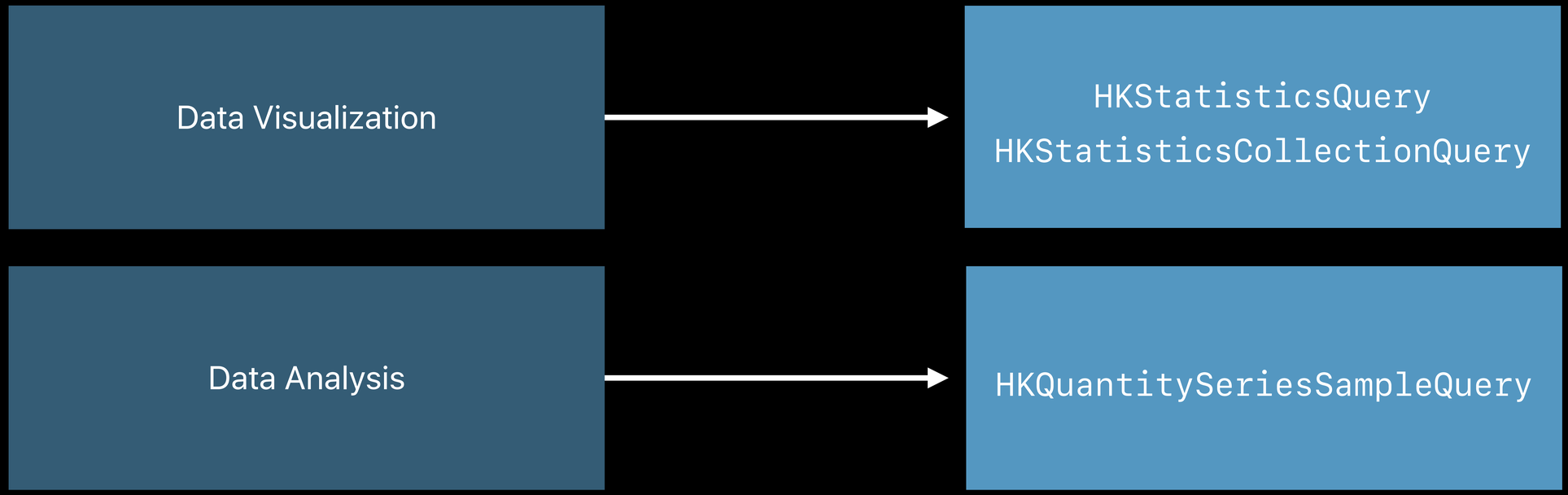
Quantity Series API

Use cases



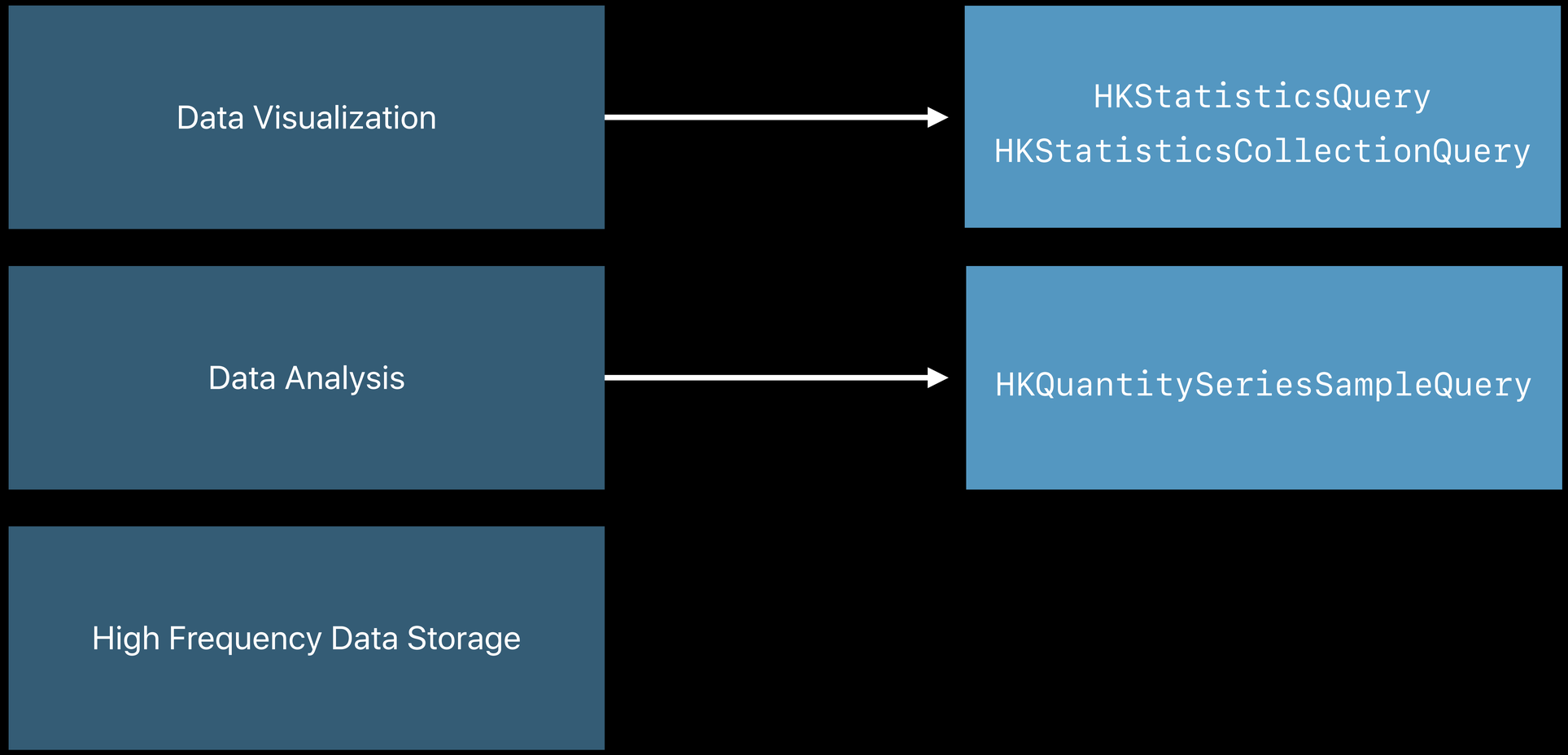
Quantity Series API

Use cases



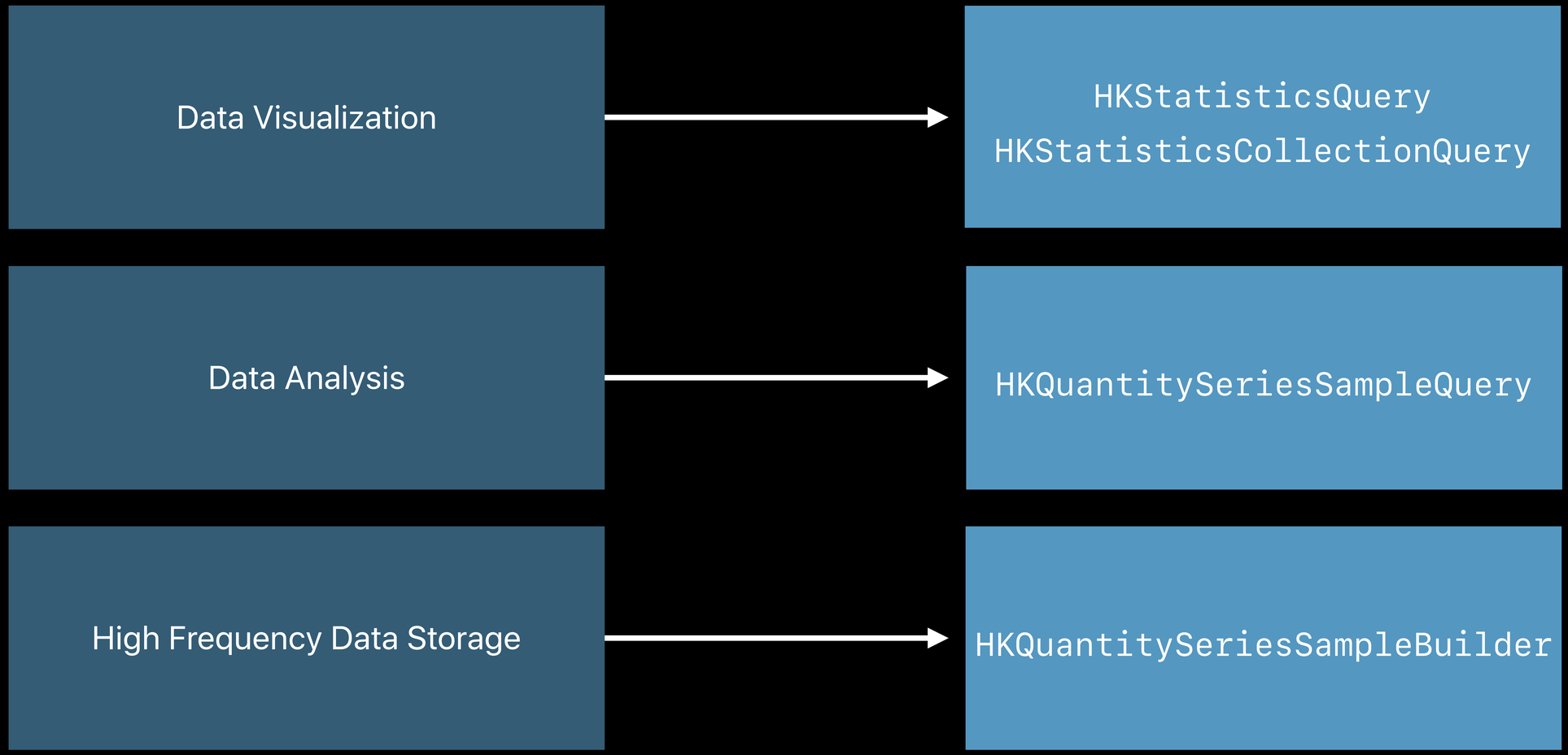
Quantity Series API

Use cases



Quantity Series API

Use cases



HKQuantitySeriesSampleQuery

HKQuantitySeriesSampleQuery

Implementation

NEW

HKQuantitySeriesSampleQuery

Implementation

NEW

```
// Step 1: Create array to hold HKQuantity objects
var quantities = [HKQuantity]()
```

HKQuantitySeriesSampleQuery

Implementation

NEW

```
// Step 1: Create array to hold HKQuantity objects
var quantities = [HKQuantity]()

// Step 2: Initialize your query and gather data in handler
let seriesQuery = HKQuantitySeriesSampleQuery(quantitySample: seriesSample) {
     (query, quantity, date, success, error) in
     analyzeQuantity(quantity: quantity)
}
```

HKQuantitySeriesSampleQuery

Implementation

NEW

```
// Step 1: Create array to hold HKQuantity objects
var quantities = [HKQuantity]()

// Step 2: Initialize your query and gather data in handler
let seriesQuery = HKQuantitySeriesSampleQuery(quantitySample: seriesSample) {
     (query, quantity, date, success, error) in
     analyzeQuantity(quantity: quantity)
}

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

HKQuantitySeriesSampleBuilder

HKQuantitySeriesSampleBuilder

Implementation

NEW

HKQuantitySeriesSampleBuilder

Implementation

NEW

```
// Step 1: Create an HKQuantitySeriesSampleBuilder
let seriesBuilder = HKQuantitySeriesSampleBuilder(healthStore: healthStore, quantityType:
quantityType, startDate: Date(), device: nil)
```

HKQuantitySeriesSampleBuilder

Implementation

NEW

```
// Step 1: Create an HKQuantitySeriesSampleBuilder
let seriesBuilder = HKQuantitySeriesSampleBuilder(healthStore: healthStore, quantityType:
quantityType, startDate: Date(), device: nil)
// Step 2: Insert HKQuantity objects
while shouldInsert {
    let quantity = HKQuantity(unit: .meter(), doubleValue: lastDistance)
    seriesBuilder.insert(quantity) { (success, error) in
        // Handle errors…
    }
}
```

HKQuantitySeriesSampleBuilder

Implementation

NEW

```
// Step 1: Create an HKQuantitySeriesSampleBuilder
let seriesBuilder = HKQuantitySeriesSampleBuilder(healthStore: healthStore, quantityType:
quantityType, startDate: Date(), device: nil)
// Step 2: Insert HKQuantity objects
while shouldInsert {
    let quantity = HKQuantity(unit: .meter(), doubleValue: lastDistance)
    seriesBuilder.insert(quantity) { (success, error) in
        // Handle errors…
    }
}
// Step 3: Complete the series
seriesBuilder.finishSeries(metadata: nil) { (quantity, error) in
    // Handle errors…
}
```

Summary

Summary

Respect User Privacy

Summary

Respect User Privacy

Quickly Build Robust Workout Apps

Summary

Respect User Privacy

Quickly Build Robust Workout Apps

Save Data Efficiently

More Information

https://developer.apple.com/wwdc2018/707

Health and Fitness Technologies Lab	Technology Lab 4	Wednesday, 1:00PM
Health, Fitness, and Research Get-Together	Market Terrace	Wednesday, 6:15PM
Accessing Health Records with HealthKit		WWDC 2018


WWDC18