Introducing HealthKit

Session 203

Justin Rushing
iOS Software Engineer

Siji Rachel Tom
iOS Software Engineer









INSTANT HEART RATE
108 BPM

5K

WebMD




7

120
80

A1C
80-120
BG 8




UP

Existing Applications

Existing Applications

Statistical Analysis (Graphs, Trends)

Existing Applications

Statistical Analysis (Graphs, Trends)

Enter Information

Existing Applications

Statistical Analysis (Graphs, Trends)

Enter Information

Applications From Health Providers


9:41 AM
100%
Dashboard
Calories Burned
Daily Avg: 428
467cal
Today
743
136
May 5
12
19
26
June 2
Sleep
Daily Avg: 7h 9m
7h 29m
Last Night
9.5
0
May 5
12
19
26
June 2
Heart Rate
Min: 53 Max: 126
68bpm
Today
126
Dashboard
Health Data
Sources
Medical ID

Agenda

HealthKit API

- Creating data
- Saving data
- Asking for data

Demo

HealthKit Best Practices

Data in HealthKit

HKUnit

HKUnit

Represents a particular unit

HKUnit

Represents a particular unit

Simple or Complex

HKUnit

Represents a particular unit

Simple or Complex

Base units classified into types

- Mass, Length, Volume, …

mg/dL = Mass / Volume

HKUnit

```
HKUnit *g = [HKUnit gramUnit];
HKUnit *dL = [HKUnit literUnitWithMetricPrefix:HKMetricPrefixDeci];
HKUnit *gPerdL = [g unitDividedByUnit:dL];
```

HKUnit

```
HKUnit *g = [HKUnit gramUnit];
HKUnit *dL = [HKUnit literUnitWithMetricPrefix:HKMetricPrefixDeci];
HKUnit *gPerdL = [g unitDividedByUnit:dL];
```

HKUnit

```
HKUnit *g = [HKUnit gramUnit];
HKUnit *dL = [HKUnit literUnitWithMetricPrefix:HKMetricPrefixDeci];
HKUnit *gPerdL = [g unitDividedByUnit:dL];
```

HKUnit

```
HKUnit *g = [HKUnit gramUnit];
HKUnit *dL = [HKUnit literUnitWithMetricPrefix:HKMetricPrefixDeci];
HKUnit *gPerdL = [g unitDividedByUnit:dL];
```

HKUnit

```
HKUnit *g = [HKUnit gramUnit];
HKUnit *dL = [HKUnit literUnitWithMetricPrefix:HKMetricPrefixDeci];
HKUnit *gPerdL = [g unitDividedByUnit:dL];
```

or...

```
HKUnit *gPerdL = [HKUnit unitFromString:@"g/dL"];
```

HKQuantity

Double value relative to a unit

HKQuantity

Double value relative to a unit

Used for unit conversion

HKQuantity

```
HKUnit *gramUnit = [HKUnit gramUnit];
HKQuantity *grams = [HKQuantity quantityWithUnit:gramUnit doubleValue:20];
double kg = [grams doubleValueForUnit:[HKUnit unitFromString:@"kg"]];
```

HKQuantity

```
HKUnit *gramUnit = [HKUnit gramUnit];
HKQuantity *grams = [HKQuantity quantityWithUnit:gramUnit doubleValue:20];
double kg = [grams doubleValueForUnit:[HKUnit unitFromString:@"kg"]];
```

HKQuantity

```
HKUnit *gramUnit = [HKUnit gramUnit];
HKQuantity *grams = [HKQuantity quantityWithUnit:gramUnit doubleValue:20];
double kg = [grams doubleValueForUnit:[HKUnit unitFromString:@"kg"]];
```

HKQuantity

```
HKUnit *gramUnit = [HKUnit gramUnit];
HKQuantity *grams = [HKQuantity quantityWithUnit:gramUnit doubleValue:20];
double kg = [grams doubleValueForUnit:[HKUnit unitFromString:@“kg"]];
```

kg → .02

HKQuantity

Throws an exception if asked for an incompatible unit

```
BOOL kgCompatible = [grams isCompatibleWithUnit:[HKUnit unitFromString:@"kg"]];
BOOL kCalCompatible = [grams isCompatibleWithUnit:[HKUnit kilocalorieUnit]];
```

HKQuantity

Throws an exception if asked for an incompatible unit

```
BOOL kgCompatible = [grams isCompatibleWithUnit:[HKUnit unitFromString:@"kg"]];
BOOL kCalCompatible = [grams isCompatibleWithUnit:[HKUnit kilocalorieUnit]];
```

HKQuantity

Throws an exception if asked for an incompatible unit

```
BOOL kgCompatible = [grams isCompatibleWithUnit:[HKUnit unitFromString:@"kg"]];
BOOL kCalCompatible = [grams isCompatibleWithUnit:[HKUnit kilocalorieUnit]];
```

kgCompatible ⟶ YES

kcalCompatible ⟶ NO

HKObjectType

Steps

Calories

Nike Fuel

Vitamin C

Heart Rate

Oxygen Saturation

Body Temperature

Distance

Body Mass

BMI

Potassium

Vitamin B6

RR Interval

Blood Pressure

Vitamin B12

BAC

Body Fat Percentage

Height

Respiratory Rate

Perfusion Index

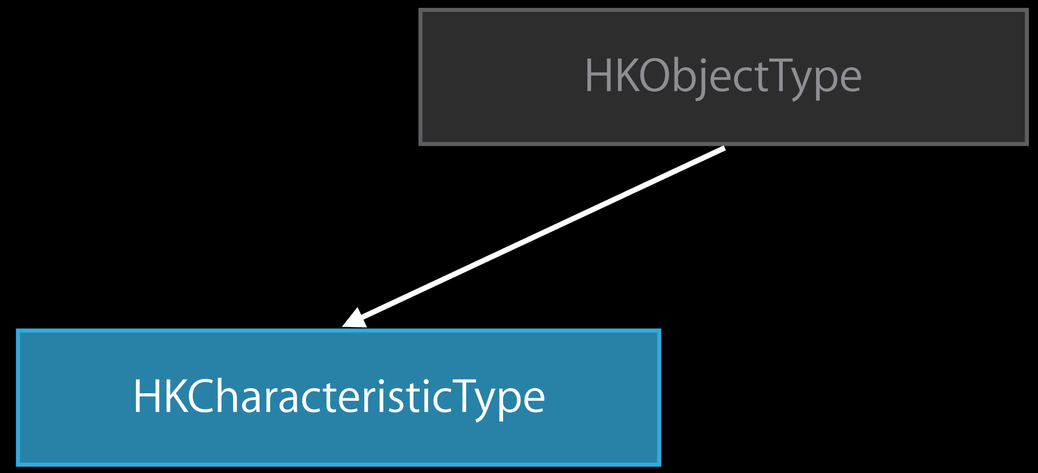
Vitamin A

Vitamin D

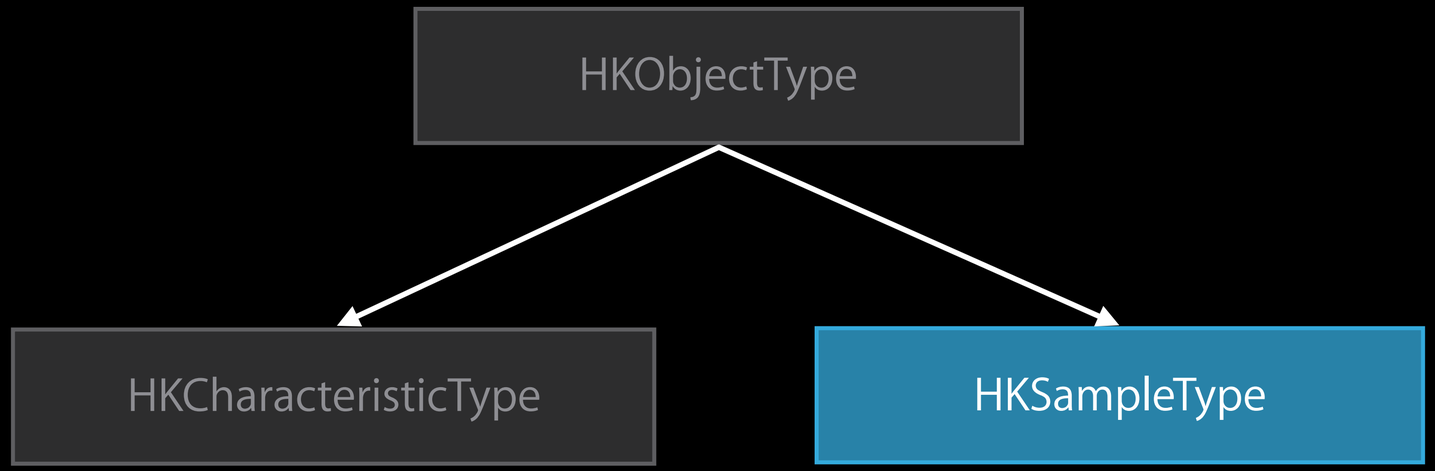
Blood Glucose

HKObjectType

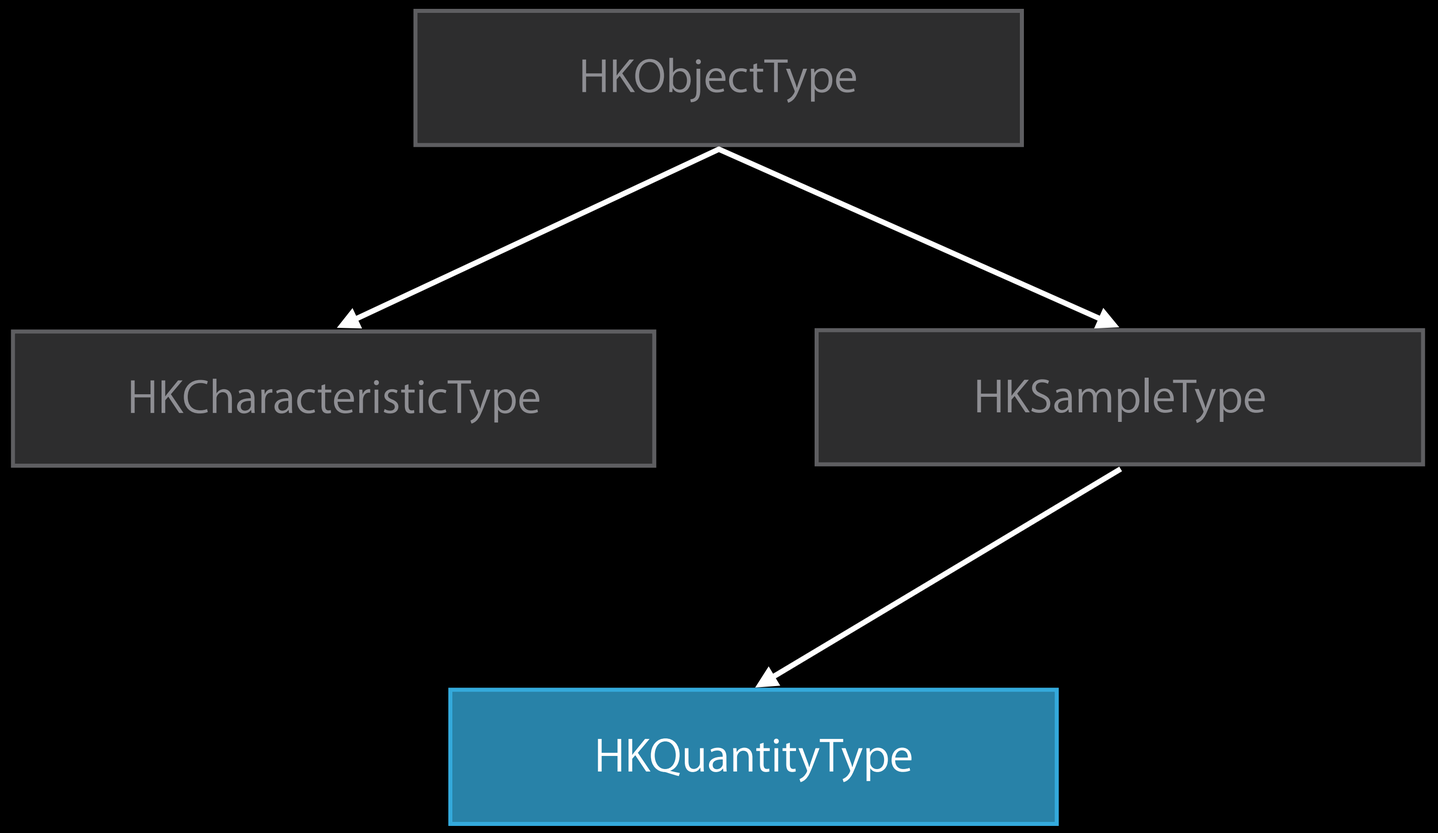
HKObjectType



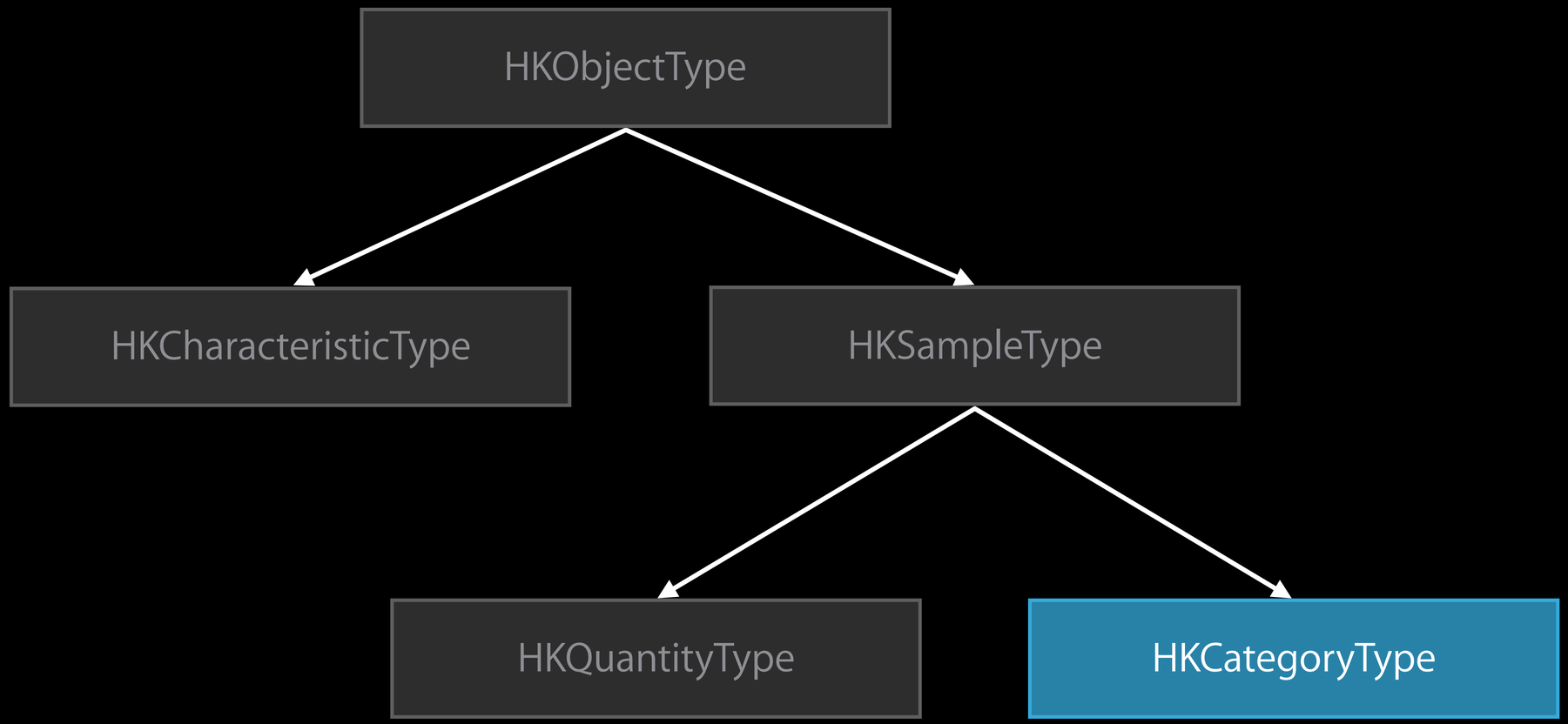
HKObjectType



HKObjectType



HKObjectType



HKObjectType

Type identifiers

HKObjectType

Type identifiers

HKQuantityTypeIdentifierHeartRate

HKObjectType

Type identifiers

HKQuantityTypeIdentifierHeartRate

HKObjectType Subclass

HKObjectType

Type identifiers

HKQuantityTypeIdentifierHeartRate

Type Name

HKObjectType

Creation

```
@interface HKObjectType : NSObject

+ (HKQuantityType *)quantityTypeForIdentifier:(NSString *)identifier;
+ (HKCategoryType *)categoryTypeForIdentifier:(NSString *)identifier;
+ (HKCharacteristicType *)characteristicTypeForIdentifier:(NSString
*)identifier;

@end
```

HKObjectType

Creation

```
@interface HKObjectType : NSObject

+ (HKQuantityType *)quantityTypeForIdentifier:(NSString *)identifier;
+ (HKCategoryType *)categoryTypeForIdentifier:(NSString *)identifier;
+ (HKCharacteristicType *)characteristicTypeForIdentifier:(NSString
*)identifier;

@end
```

HKObject

HKObject

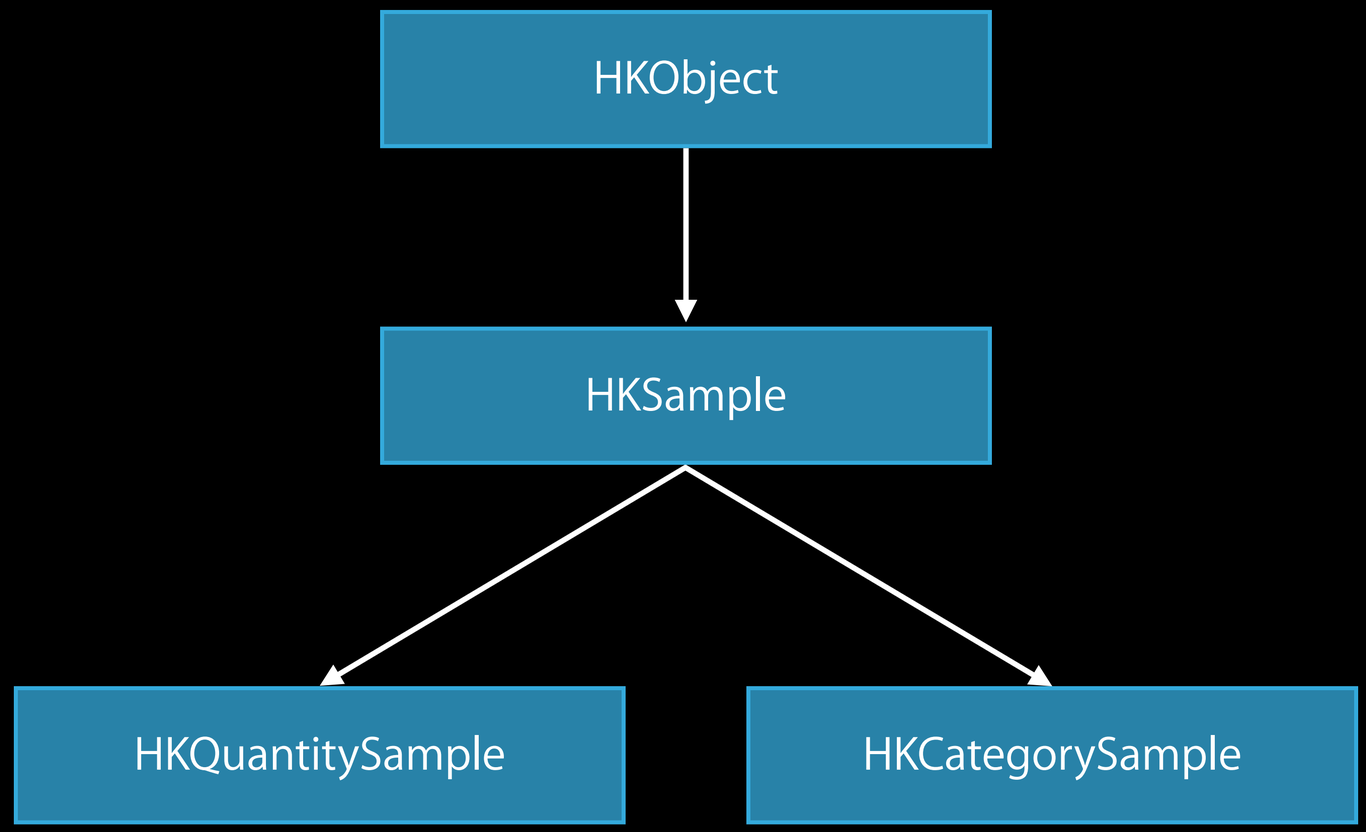
HKQuantity ✔

HKObject

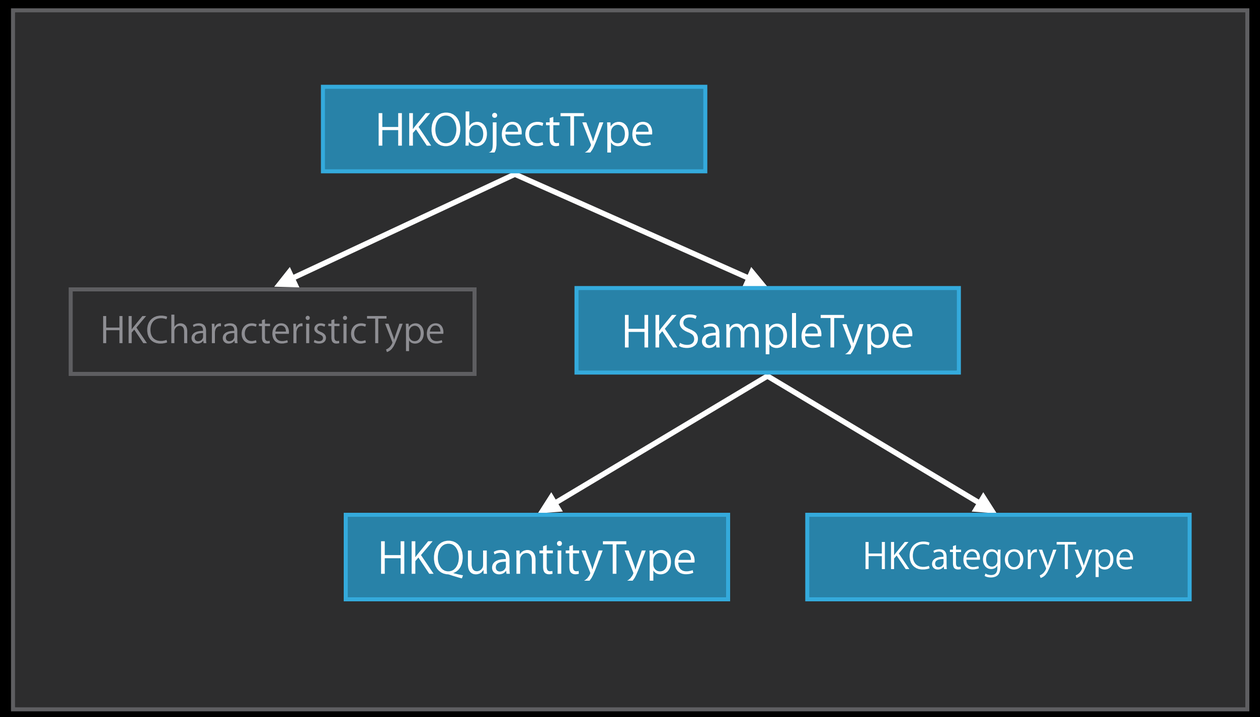
HKQuantity ✔

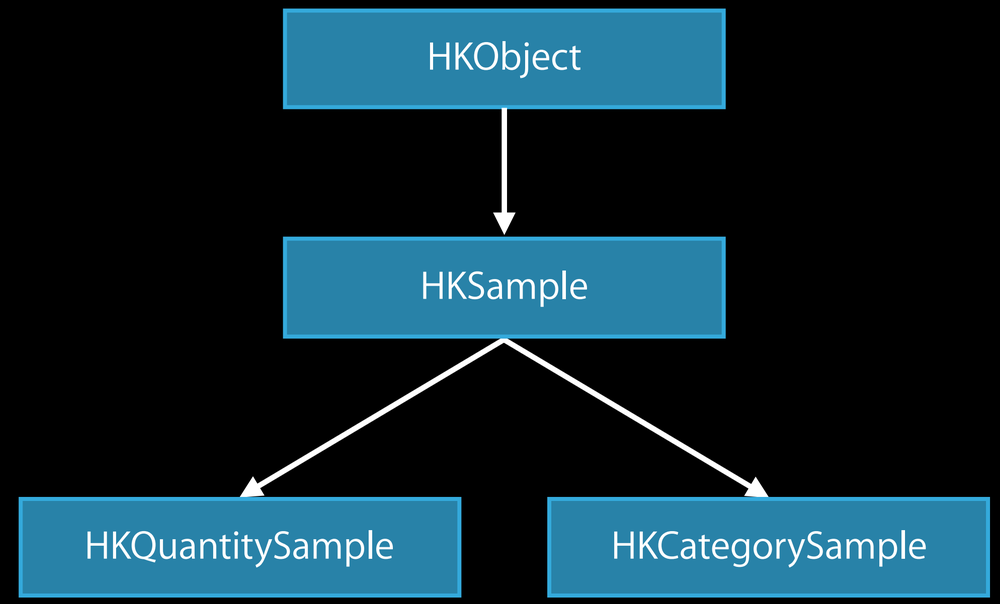
HKObjectType ✔

HKObject



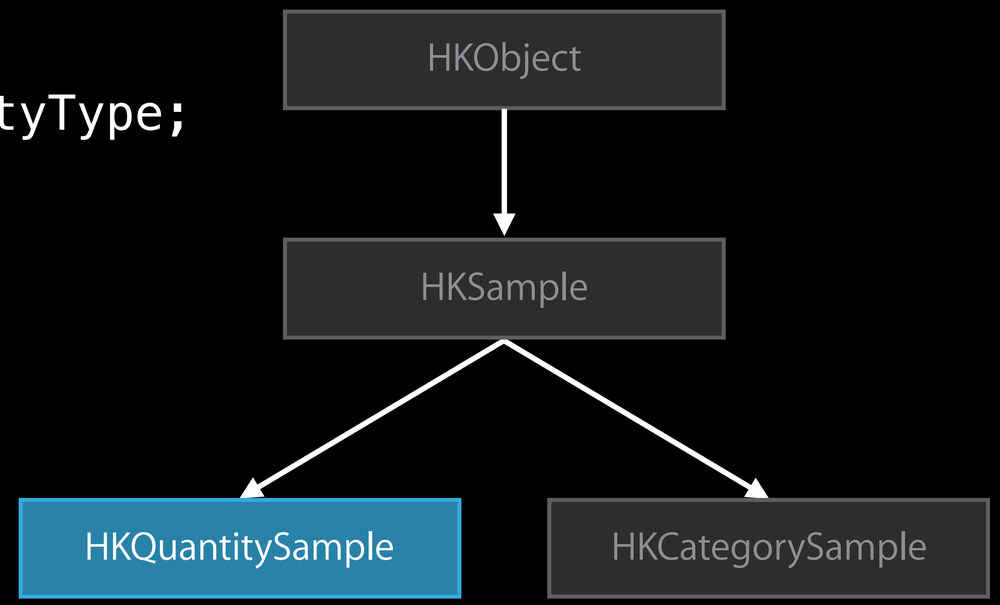
HKObject


HKObjectType
HKCharacteristicType
HKSampleType
HKQuantityType
HKCategoryType


HKObject
HKSample
HKQuantitySample
HKCategorySample

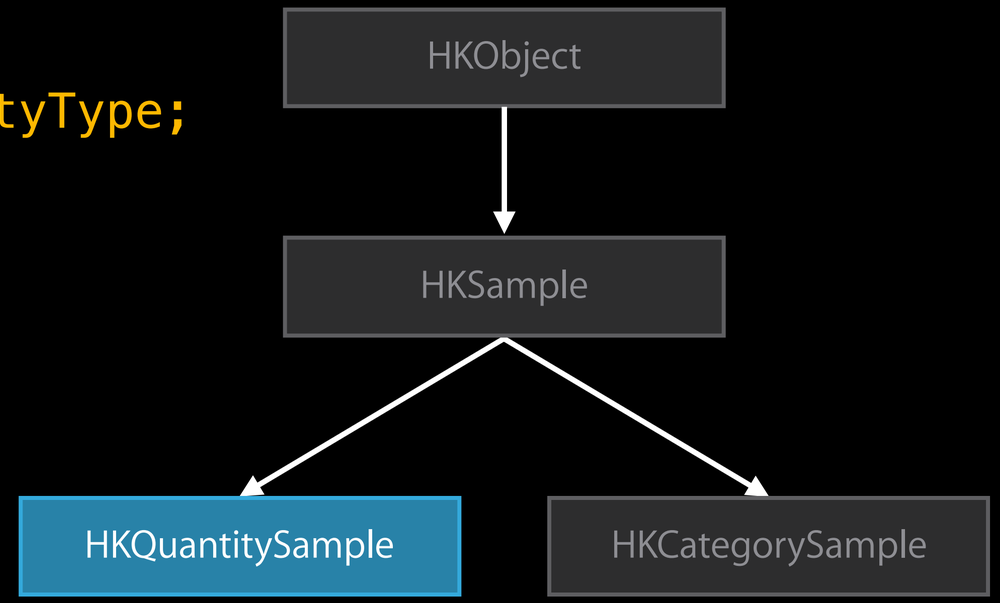
HKQuantitySample

```
@interface HKQuantitySample
@property (readonly) HKQuantityType *quantityType;
@property (readonly) HKQuantity *quantity;
@end
```



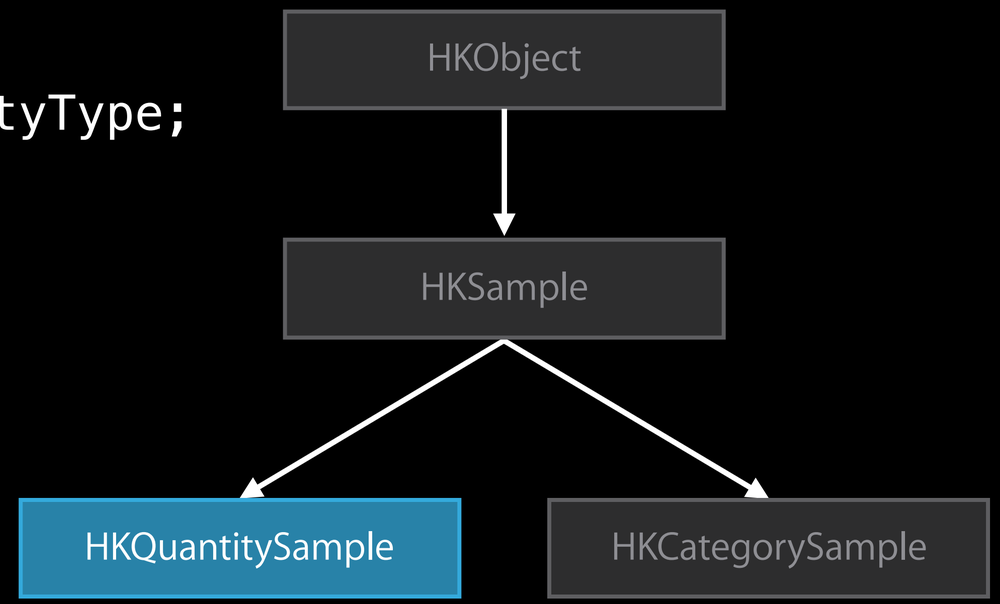
HKQuantitySample

```
@interface HKQuantitySample
@property (readonly) HKQuantityType *quantityType;
@property (readonly) HKQuantity *quantity;
@end
```



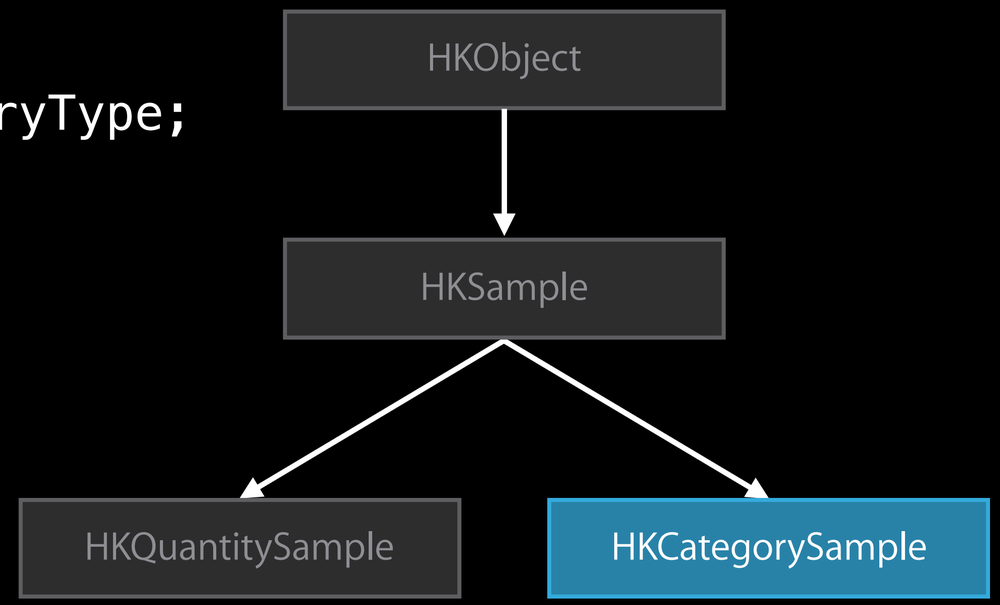
HKQuantitySample

```
@interface HKQuantitySample
@property (readonly) HKQuantityType *quantityType;
@property (readonly) HKQuantity *quantity;
@end
```



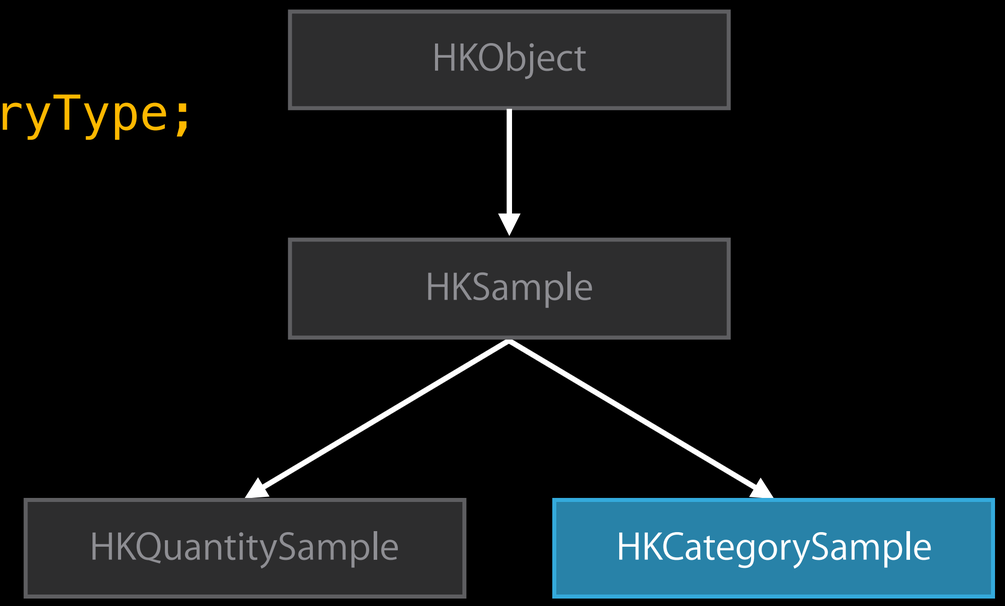
HKCategorySample

```
@interface HKCategorySample
@property (readonly) HKCategoryType *categoryType;
@property (readonly) NSInteger value;
@end
```



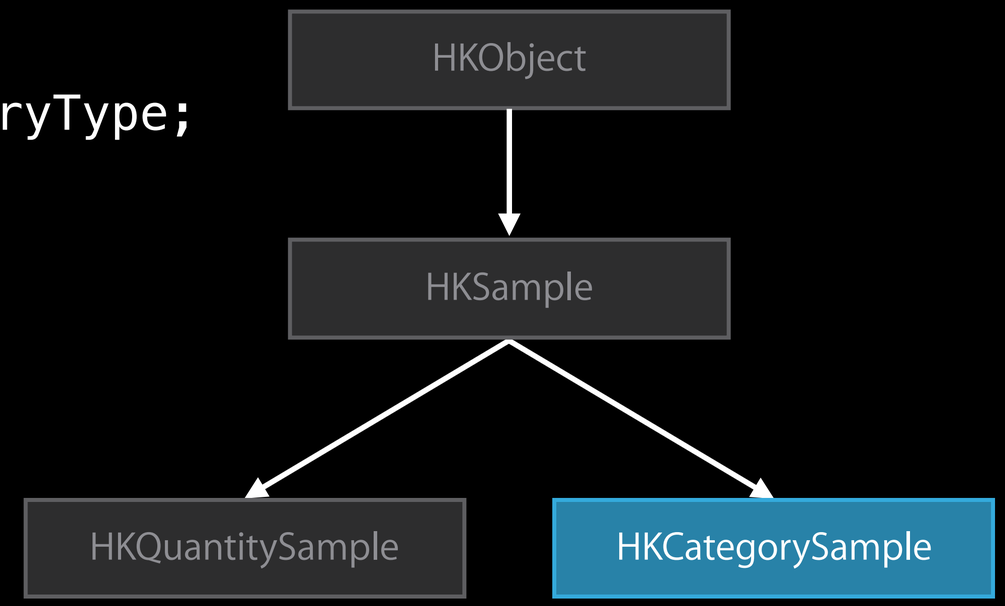
HKCategorySample

```
@interface HKCategorySample
@property (readonly) HKCategoryType *categoryType;
@property (readonly) NSInteger value;
@end
```



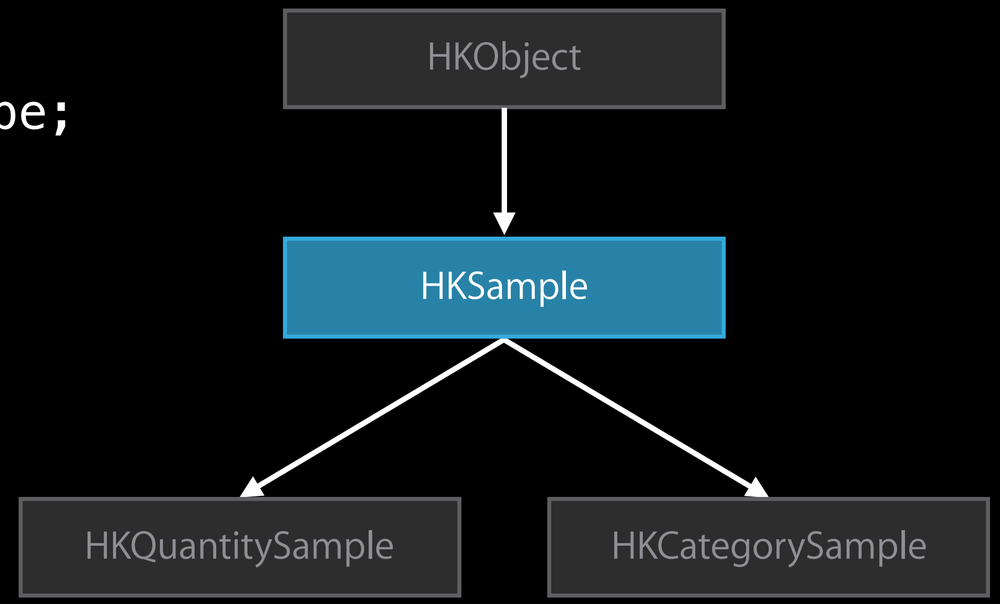
HKCategorySample

```
@interface HKCategorySample
@property (readonly) HKCategoryType *categoryType;
@property (readonly) NSInteger value;
@end
```



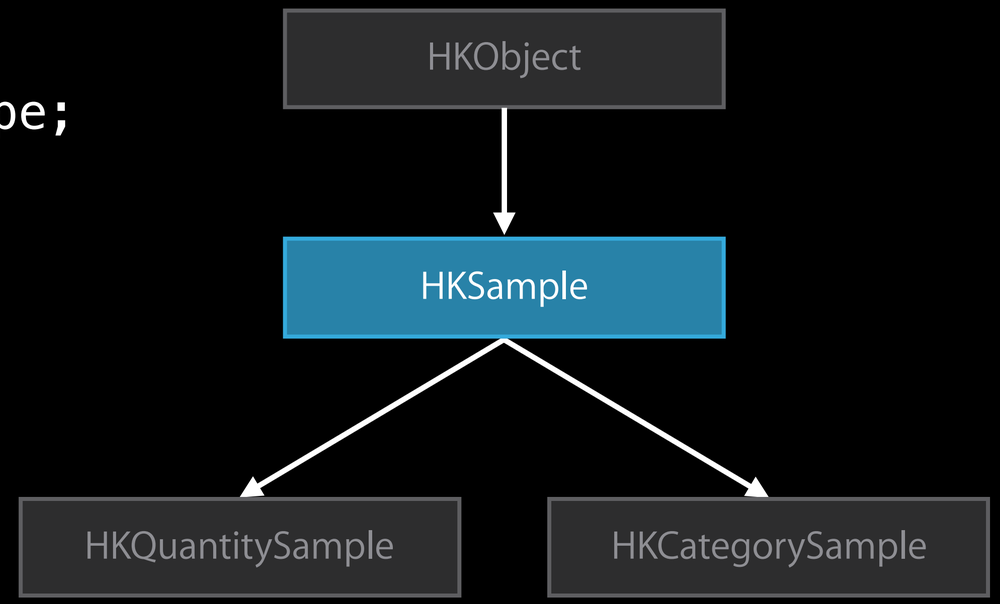
HKSample

```
@interface HKSample
@property (readonly) HKSampleType *sampleType;
@property (readonly) NSDate *startDate;
@property (readonly) NSDate *endDate;
@end
```



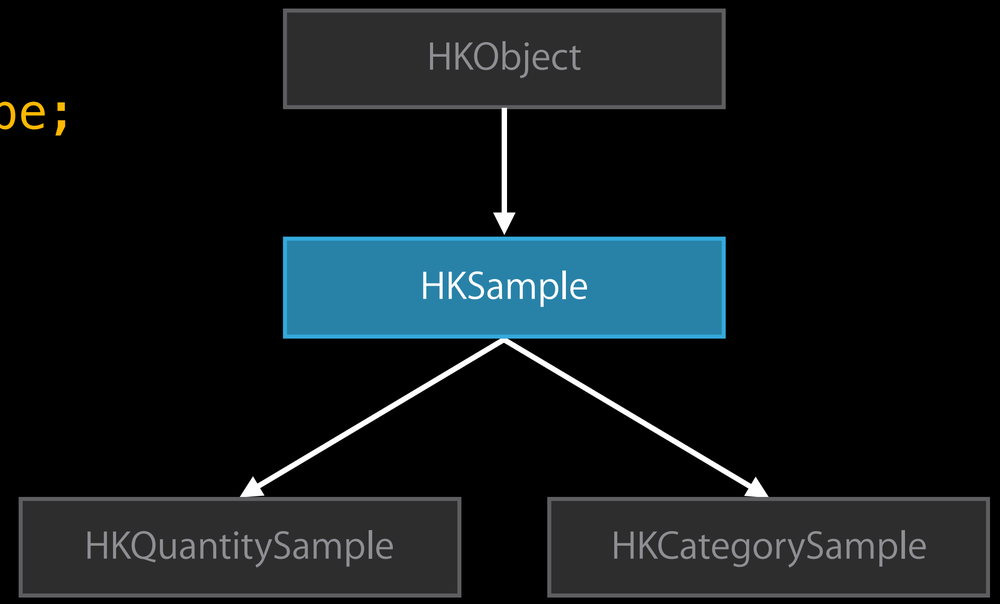
HKSample

```
@interface HKSample
@property (readonly) HKSampleType *sampleType;
@property (readonly) NSDate *startDate;
@property (readonly) NSDate *endDate;
@end
```



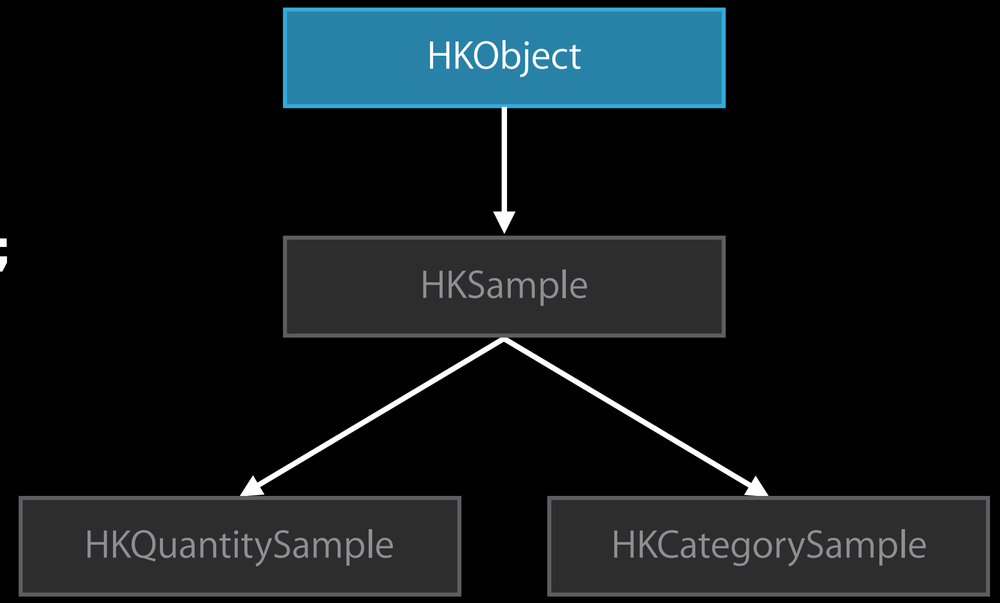
HKSample

```
@interface HKSample
@property (readonly) HKSampleType *sampleType;
@property (readonly) NSDate *startDate;
@property (readonly) NSDate *endDate;
@end
```



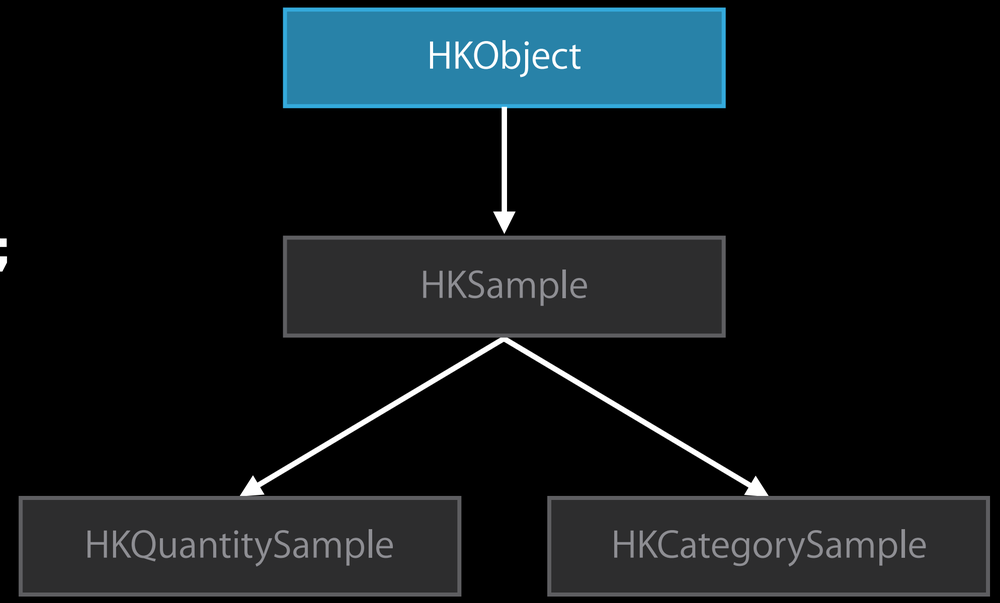
HKObject

```
@interface HKObject
@property (readonly) NSUUID *UUID;
@property (readonly) HKSource *source;
@property (readonly) NSDictionary *metadata;
@end
```



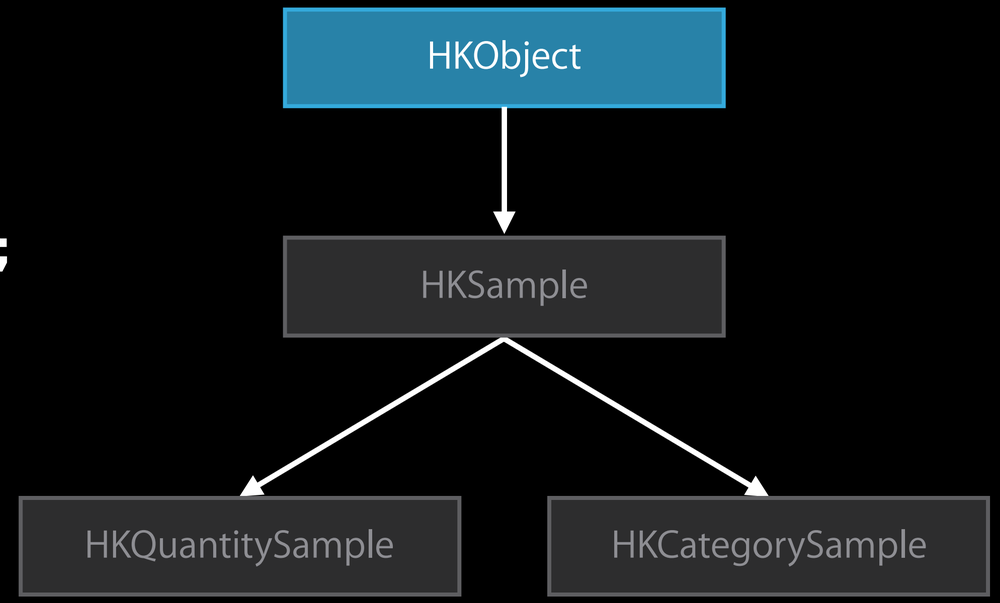
HKObject

```
@interface HKObject
@property (readonly) NSUUID *UUID;
@property (readonly) HKSource *source;
@property (readonly) NSDictionary *metadata;
@end
```



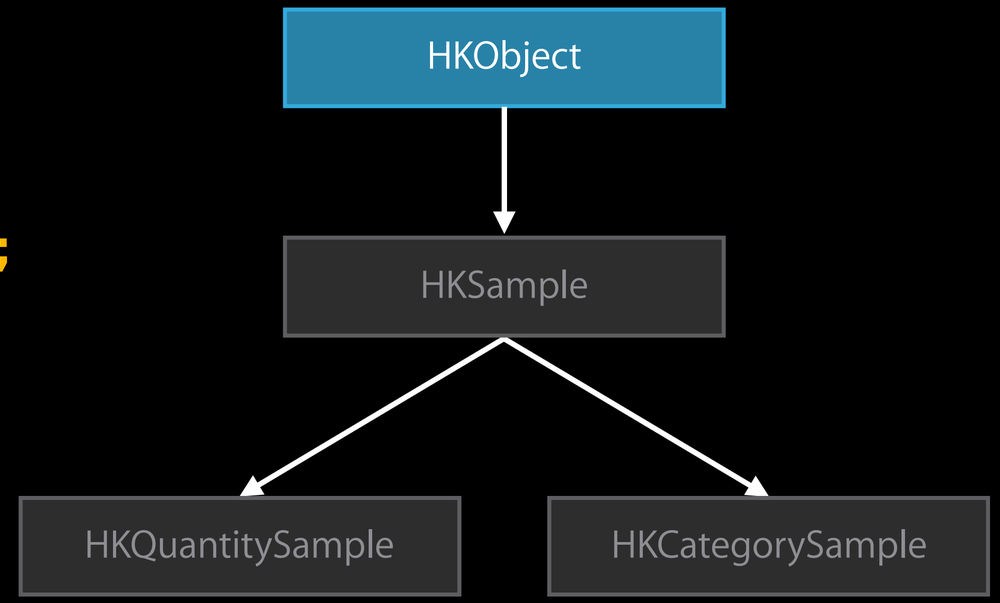
HKObject

```
@interface HKObject
@property (readonly) NSUUID *UUID;
@property (readonly) HKSource *source;
@property (readonly) NSDictionary *metadata;
@end
```



HKObject

```
@interface HKObject
@property (readonly) NSUUID *UUID;
@property (readonly) HKSource *source;
@property (readonly) NSDictionary *metadata;
@end
```



HKObject

Properties are readonly

HKObject

Properties are readonly

All HKObjects are immutable

HKObject

Properties are readonly

All HKObjects are immutable

Constructors on HKQuantitySample, HKCategorySample

Creating an HKObject

```
NSString *identifier = HKQuantityTypeIdentifierBodyTemperature;
HKQuantityType *tempType = [HKObjectType quantityTypeForIdentifier:identifier];

HKQuantity *myTemp = [HKQuantity quantityWithUnit:[HKUnit degreeFahrenheitUnit]
                                       doubleValue:98.6];

NSDictionary *meta = @{
            HKMetadataKeyBodyTemperatureSensorLocation : @(HKBodyTemperatureSensorLocationEar)
            };
HKQuantitySample *temperatureSample = [HKQuantitySample quantitySampleWithType:tempType
                                                                       quantity:myTemp
                                                                      startDate:[NSDate date]
                                                                        endDate:[NSDate date]
                                                                       metadata:meta];
```

Creating an HKObject

```
NSString *identifier = HKQuantityTypeIdentifierBodyTemperature;
HKQuantityType *tempType = [HKObjectType quantityTypeForIdentifier:identifier];

HKQuantity *myTemp = [HKQuantity quantityWithUnit:[HKUnit degreeFahrenheitUnit]
                                       doubleValue:98.6];

NSDictionary *meta = @{
            HKMetadataKeyBodyTemperatureSensorLocation : @(HKBodyTemperatureSensorLocationEar)
            };
HKQuantitySample *temperatureSample = [HKQuantitySample quantitySampleWithType:tempType
                                                                       quantity:myTemp
                                                                      startDate:[NSDate date]
                                                                        endDate:[NSDate date]
                                                                       metadata:meta];
```

Creating an HKObject

```
NSString *identifier = HKQuantityTypeIdentifierBodyTemperature;
HKQuantityType *tempType = [HKObjectType quantityTypeForIdentifier:identifier];

HKQuantity *myTemp = [HKQuantity quantityWithUnit:[HKUnit degreeFahrenheitUnit]
                                       doubleValue:98.6];

NSDictionary *meta = @{
            HKMetadataKeyBodyTemperatureSensorLocation : @(HKBodyTemperatureSensorLocationEar)
            };
HKQuantitySample *temperatureSample = [HKQuantitySample quantitySampleWithType:tempType
                                                                       quantity:myTemp
                                                                      startDate:[NSDate date]
                                                                        endDate:[NSDate date]
                                                                       metadata:meta];
```

Creating an HKObject

```
NSString *identifier = HKQuantityTypeIdentifierBodyTemperature;
HKQuantityType *tempType = [HKObjectType quantityTypeForIdentifier:identifier];

HKQuantity *myTemp = [HKQuantity quantityWithUnit:[HKUnit degreeFahrenheitUnit]
                                       doubleValue:98.6];

NSDictionary *meta = @{
            HKMetadataKeyBodyTemperatureSensorLocation : @(HKBodyTemperatureSensorLocationEar)
            };
HKQuantitySample *temperatureSample = [HKQuantitySample quantitySampleWithType:tempType
                                                                       quantity:myTemp
                                                                      startDate:[NSDate date]
                                                                        endDate:[NSDate date]
                                                                       metadata:meta];
```

Creating an HKObject

```
NSString *identifier = HKQuantityTypeIdentifierBodyTemperature;
HKQuantityType *tempType = [HKObjectType quantityTypeForIdentifier:identifier];

HKQuantity *myTemp = [HKQuantity quantityWithUnit:[HKUnit degreeFahrenheitUnit]
                                       doubleValue:98.6];

NSDictionary *meta = @{
                HKMetadataKeyBodyTemperatureSensorLocation : @(HKBodyTemperatureSensorLocationEar)
                };
    HKQuantitySample *temperatureSample = [HKQuantitySample quantitySampleWithType:tempType
                                                                          quantity:myTemp
                                                                         startDate:[NSDate date]
                                                                           endDate:[NSDate date]
                                                                          metadata:meta];
```

Saving Data

HKHealthStore

Link to the database

HKHealthStore

Link to the database

Lets you save objects and query for data

HKHealthStore

Link to the database

Lets you save objects and query for data

Should be long lived

HKHealthStore

Saving objects

```
self.store = [[HKHealthStore alloc] init];

...

HKQuantitySample *mySample = [self newSample];
[self.store saveObject:mySample withCompletion:^(BOOL success, NSError *error) {
    if (success) {
        NSLog(@"Object Saved!");
    }
}];
```

HKHealthStore

Saving objects

```
self.store = [[HKHealthStore alloc] init];

...

HKQuantitySample *mySample = [self newSample];
[self.store saveObject:mySample withCompletion:^(BOOL success, NSError *error) {
    if (success) {
        NSLog(@"Object Saved!");
    }
}];
```

HKHealthStore

Saving objects

```
self.store = [[HKHealthStore alloc] init];

...

HKQuantitySample *mySample = [self newSample];
[self.store saveObject:mySample withCompletion:^(BOOL success, NSError *error) {
    if (success) {
        NSLog(@"Object Saved!");
    }
}];
```

HKHealthStore

Saving objects

```
self.store = [[HKHealthStore alloc] init];

...

HKQuantitySample *mySample = [self newSample];
[self.store saveObject:mySample withCompletion:^(BOOL success, NSError *error) {
    if (success) {
        NSLog(@"Object Saved!");
    }
}];
```

HKHealthStore

Saving objects

```
self.store = [[HKHealthStore alloc] init];

...

HKQuantitySample *mySample = [self newSample];
[self.store saveObject:mySample withCompletion:^(BOOL success, NSError *error) {
    if (success) {
        NSLog(@"Object Saved!");
    }
}];
```

Asking for Data

HKHealthStore

Characteristics

For user characteristics, ask HKHealthStore directly

- Date of birth, blood type, biological sex

HKHealthStore

Characteristics

For user characteristics, ask HKHealthStore directly

- Date of birth, blood type, biological sex

```
NSError *error;
NSDate *dateOfBirth = [self.store dateOfBirthWithError:&error];
```

Queries

HKQuery

```
@interface HKQuery
@property (readonly) HKSampleType *sampleType;
@property (readonly) NSPredicate *predicate;
@end
```

HKQuery

```
@interface HKQuery
@property (readonly) HKSampleType *sampleType;
@property (readonly) NSPredicate *predicate;
@end
```

HKQuery

```
@interface HKQuery
@property (readonly) HKSampleType *sampleType;
@property (readonly) NSPredicate *predicate;
@end
```

HKQuery

Predicates

```
HKQuantity *weight = ...
[NSPredicate predicateWithFormat:@"%K > %@", HKPredicateKeyPathQuantity, weight];
```

HKQuery

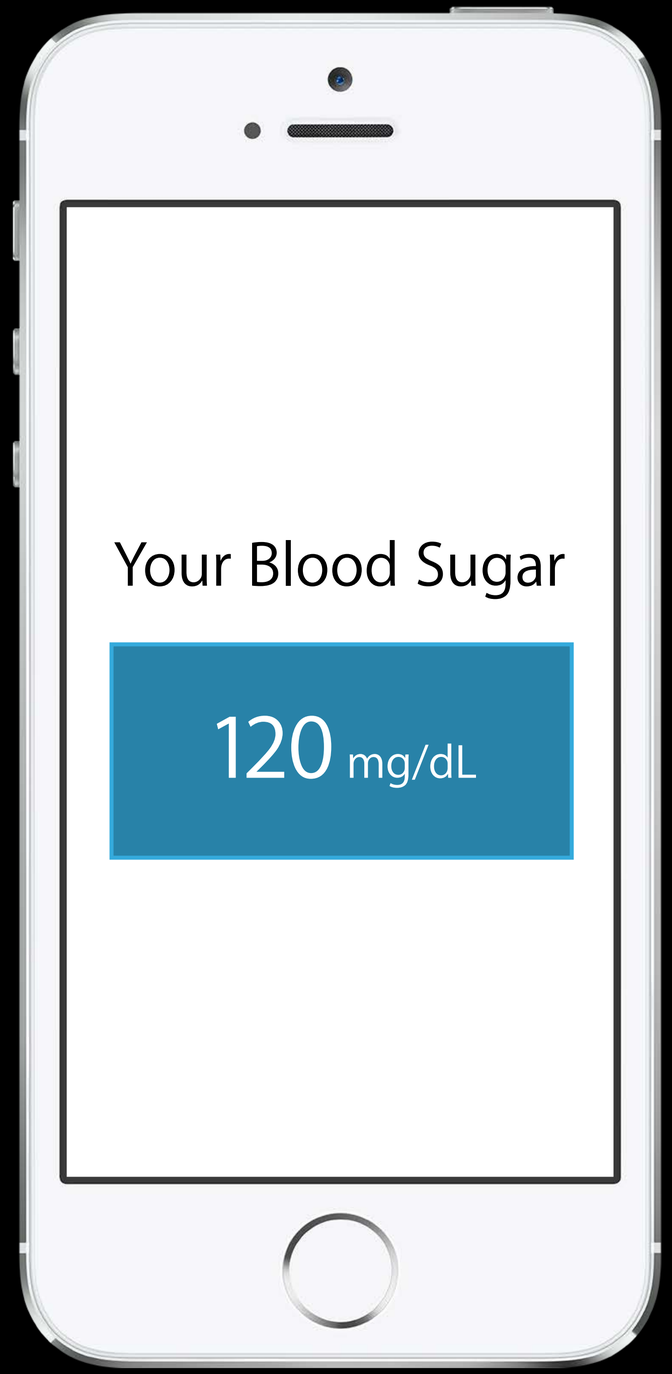
Predicates

```
HKQuantity *weight = ...
[NSPredicate predicateWithFormat:@"%K > %@", HKPredicateKeyPathQuantity, weight];
```

HKQuery

Predicates

```
HKQuantity *weight = ...
[NSPredicate predicateWithFormat:@"%K > %@", HKPredicateKeyPathQuantity, weight];


NSPredicateOperatorType greaterThan = NSGreaterThanPredicateOperatorType;
[HKQuery predicateForQuantitySamplesWithOperatorType:greaterThan
                                            quantity:weight];
```


Your Blood Sugar
120 mg/dL

HKSampleQuery

HKSampleQuery

Limit

- HKObjectQueryNoLimit

HKSampleQuery

Limit

- HKObjectQueryNoLimit

Sort Order

- Array of NSSortDescriptors

HKSampleQuery

Most recent query

```
HKQuantityType *bloodSugar = ...
NSString *endKey = HKSampleSortIdentifierEndDate;
NSSortDescriptor *endDate = [NSSortDescriptor sortDescriptorWithKey:endKey ascending:NO];
HKSampleQuery *query = [[HKSampleQuery alloc] initWithSampleType:bloodSugar
                                                       predicate:nil
                                                           limit:1
                                                 sortDescriptors:@[endDate]
                                                  resultsHandler:^(HKSampleQuery *query,
                                                                   NSArray *results,
                                                                   NSError *error)
{
    HKQuantitySample *sample = [results lastObject];
    NSLog(@"Sample: %@", sample);
}];
```

HKSampleQuery

Most recent query

```
HKQuantityType *bloodSugar = ...
NSString *endKey = HKSampleSortIdentifierEndDate;
NSSortDescriptor *endDate = [NSSortDescriptor sortDescriptorWithKey:endKey ascending:NO];
HKSampleQuery *query = [[HKSampleQuery alloc] initWithSampleType:bloodSugar
                                                       predicate:nil
                                                           limit:1
                                                 sortDescriptors:@[endDate]
                                                  resultsHandler:^(HKSampleQuery *query,
                                                                   NSArray *results,
                                                                   NSError *error)
{
   HKQuantitySample *sample = [results lastObject];
   NSLog(@"Sample: %@", sample);
}];
```

HKSampleQuery

Most recent query

```
HKQuantityType *bloodSugar = ...
NSString *endKey = HKSampleSortIdentifierEndDate;
NSSortDescriptor *endDate = [NSSortDescriptor sortDescriptorWithKey:endKey ascending:NO];
HKSampleQuery *query = [[HKSampleQuery alloc] initWithSampleType:bloodSugar
                                                       predicate:nil
                                                           limit:1
                                                 sortDescriptors:@[endDate]
                                                  resultsHandler:^(HKSampleQuery *query,
                                                                   NSArray *results,
                                                                   NSError *error)
{
   HKQuantitySample *sample = [results lastObject];
   NSLog(@"Sample: %@", sample);
}];
```

HKSampleQuery

Most recent query

```
HKQuantityType *bloodSugar = ...
NSString *endKey = HKSampleSortIdentifierEndDate;
NSSortDescriptor *endDate = [NSSortDescriptor sortDescriptorWithKey:endKey ascending:NO];
HKSampleQuery *query = [[HKSampleQuery alloc] initWithSampleType:bloodSugar
                                                       predicate:nil
                                                           limit:1
                                                 sortDescriptors:@[endDate]
                                                  resultsHandler:^(HKSampleQuery *query,
                                                                   NSArray *results,
                                                                   NSError *error)
{
    HKQuantitySample *sample = [results lastObject];
    NSLog(@"Sample: %@", sample);
}];
```

HKSampleQuery

Most recent query

```
HKQuantityType *bloodSugar = ...
NSString *endKey = HKSampleSortIdentifierEndDate;
NSSortDescriptor *endDate = [NSSortDescriptor sortDescriptorWithKey:endKey ascending:NO];
HKSampleQuery *query = [[HKSampleQuery alloc] initWithSampleType:bloodSugar
                                                       predicate:nil
                                                           limit:1
                                                 sortDescriptors:@[endDate]
                                                  resultsHandler:^(HKSampleQuery *query,
                                                                   NSArray *results,
                                                                   NSError *error)
{
    HKQuantitySample *sample = [results lastObject];
    NSLog(@"Sample: %@", sample);
}];
```

HKSampleQuery

Most recent query

```
HKQuantityType *bloodSugar = ...
NSString *endKey = HKSampleSortIdentifierEndDate;
NSSortDescriptor *endDate = [NSSortDescriptor sortDescriptorWithKey:endKey ascending:NO];
HKSampleQuery *query = [[HKSampleQuery alloc] initWithSampleType:bloodSugar
                                                       predicate:nil
                                                           limit:1
                                                 sortDescriptors:@[endDate]
                                                  resultsHandler:^(HKSampleQuery *query,
                                                                   NSArray *results,
                                                                   NSError *error)
{
    HKQuantitySample *sample = [results lastObject];
    NSLog(@"Sample: %@", sample);
}];
```

HKSampleQuery

Most recent query

```
HKQuantityType *bloodSugar = ...
NSString *endKey = HKSampleSortIdentifierEndDate;
NSSortDescriptor *endDate = [NSSortDescriptor sortDescriptorWithKey:endKey ascending:NO];
HKSampleQuery *query = [[HKSampleQuery alloc] initWithSampleType:bloodSugar
                                                       predicate:nil
                                                           limit:1
                                                 sortDescriptors:@[endDate]
                                                  resultsHandler:^(HKSampleQuery *query,
                                                                   NSArray *results,
                                                                   NSError *error)
{
    HKQuantitySample *sample = [results lastObject];
    NSLog(@"Sample: %@", sample);
}];
```

HKSampleQuery

Most recent query

```
HKQuantityType *bloodSugar = ...
NSString *endKey = HKSampleSortIdentifierEndDate;
NSSortDescriptor *endDate = [NSSortDescriptor sortDescriptorWithKey:endKey ascending:NO];
HKSampleQuery *query = [[HKSampleQuery alloc] initWithSampleType:bloodSugar
                                                       predicate:nil
                                                           limit:1
                                                 sortDescriptors:@[endDate]
                                                  resultsHandler:^(HKSampleQuery *query,
                                                                   NSArray *results,
                                                                   NSError *error)
{
    HKQuantitySample *sample = [results lastObject];
    NSLog(@"Sample: %@", sample);
}];
```

HKSampleQuery

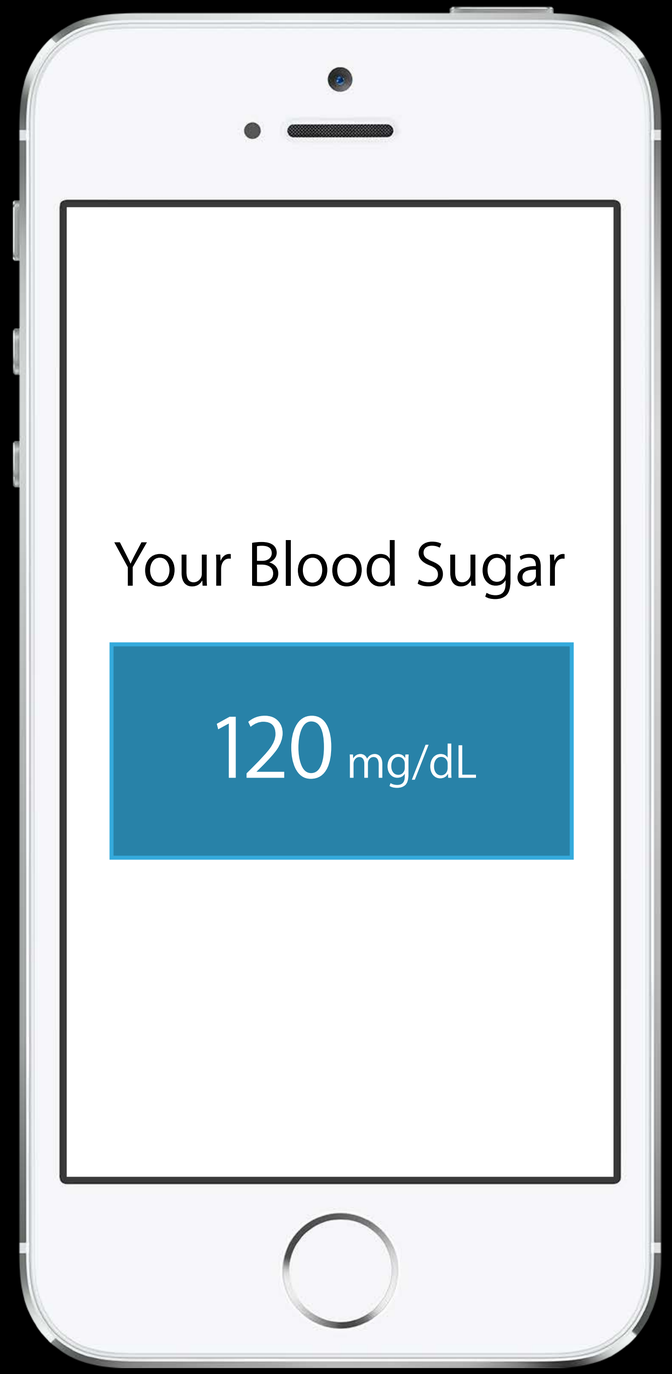
Most recent query

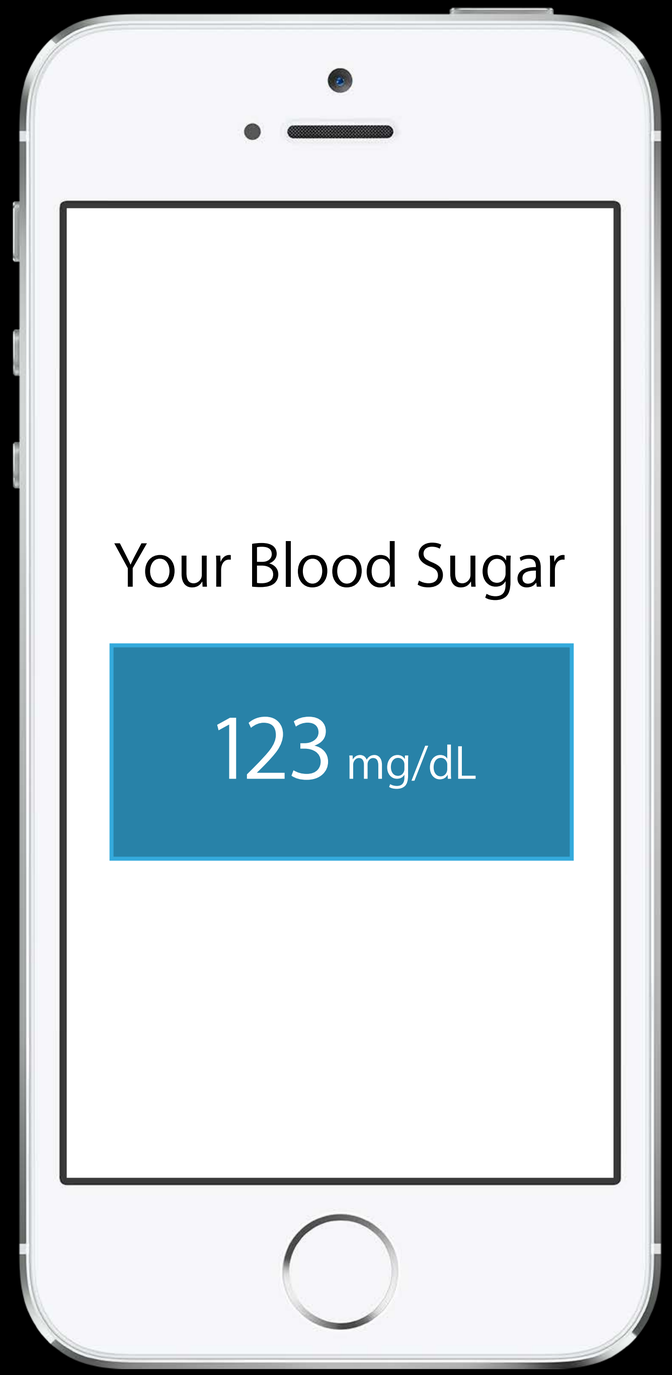
```
HKQuantityType *bloodSugar = ...
NSString *endKey = HKSampleSortIdentifierEndDate;
NSSortDescriptor *endDate = [NSSortDescriptor sortDescriptorWithKey:endKey ascending:NO];
HKSampleQuery *query = [[HKSampleQuery alloc] initWithSampleType:bloodSugar
                                                       predicate:nil
                                                           limit:1
                                                 sortDescriptors:@[endDate]
                                                  resultsHandler:^(HKSampleQuery *query,
                                                                   NSArray *results,
                                                                   NSError *error)
{
    HKQuantitySample *sample = [results lastObject];
    NSLog(@"Sample: %@", sample);
}];
```

HKSampleQuery

Most recent query

```
HKQuantityType *bloodSugar = ...
NSString *endKey = HKSampleSortIdentifierEndDate;
NSSortDescriptor *endDate = [NSSortDescriptor sortDescriptorWithKey:endKey ascending:NO];
HKSampleQuery *query = [[HKSampleQuery alloc] initWithSampleType:bloodSugar
                                                       predicate:nil
                                                           limit:1
                                                 sortDescriptors:@[endDate]
                                                  resultsHandler:^(HKSampleQuery *query,
                                                                   NSArray *results,
                                                                   NSError *error)
{
    HKQuantitySample *sample = [results lastObject];
    NSLog(@"Sample: %@", sample);
}];
```


Your Blood Sugar
120 mg/dL


Your Blood Sugar
123 mg/dL


Your Blood Sugar
122 mg/dL


Your Blood Sugar
124 mg/dL

HKObserverQuery

Watches for changes in the database

HKObserverQuery

Watches for changes in the database

Long running

- Update handler called every time something changes

HKObserverQuery

Watches for changes in the database

Long running

- Update handler called every time something changes

Completion handler only needed for background delivery

HKObserverQuery

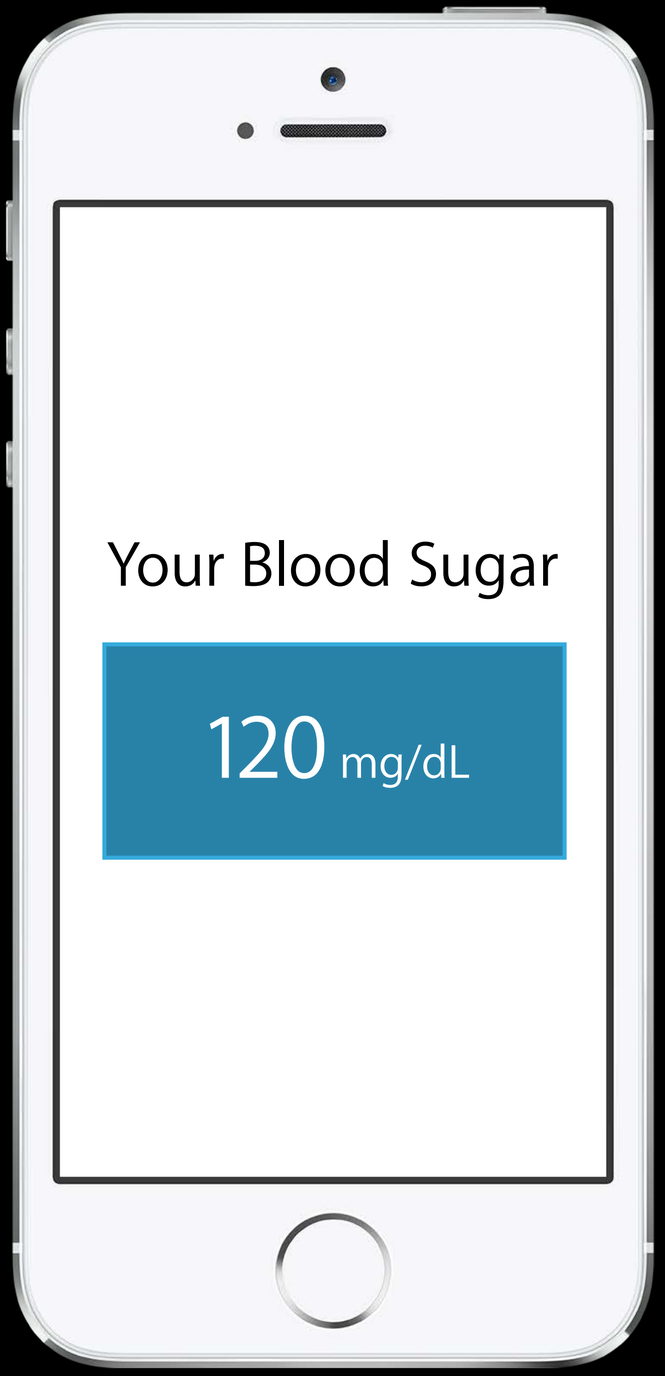
```
HKQuantityType *bloodSugar = ...
HKObserverQuery *query;
query = [[HKObserverQuery alloc] initWithSampleType:bloodSugar
                                          predicate:nil
                                      updateHandler:^(HKObserverQuery *query,
                                                      HKObserverQueryCompletionHandler handler,
                                                      NSError *error)
{
   NSLog(@“Updated!”);
}];
```

HKObserverQuery

```
HKQuantityType *bloodSugar = ...
HKObserverQuery *query;
query = [[HKObserverQuery alloc] initWithSampleType:bloodSugar
                                          predicate:nil
                                      updateHandler:^(HKObserverQuery *query,
                                                      HKObserverQueryCompletionHandler handler,
                                                      NSError *error)
{
   NSLog(@“Updated!”);
}];
```

HKObserverQuery

```
HKQuantityType *bloodSugar = ...
HKObserverQuery *query;
query = [[HKObserverQuery alloc] initWithSampleType:bloodSugar
                                          predicate:nil
                                      updateHandler:^(HKObserverQuery *query,
                                                      HKObserverQueryCompletionHandler handler,
                                                      NSError *error)
{
   NSLog(@“Updated!”);
}];
```


Your Blood Sugar
120 mg/dL

Your Blood Sugar

120 mg/dL

HKAnchoredObjectQuery

HKAnchoredObjectQuery

Limit

HKAnchoredObjectQuery

Limit

Anchor

anchor 1

anchor 2

anchor 3

anchor 4

anchor 5

anchor 6

anchor 1

anchor 2

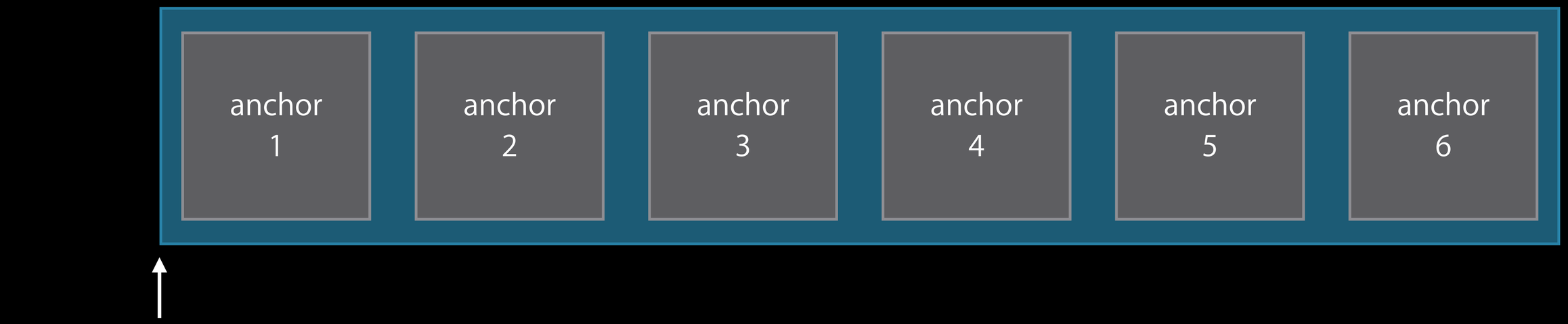
anchor 3

anchor 4

anchor 5

anchor 6

Query Anchor = 0


anchor 1
anchor 2
anchor 3
anchor 4
anchor 5
anchor 6
Query Anchor = 0

anchor 1

anchor 2

anchor 3

anchor 4

anchor 5

anchor 6

Query Anchor = 3

HKAnchoredObjectQuery

Anchors

Anchor is the last piece of data you've seen

HKAnchoredObjectQuery

Anchors

Anchor is the last piece of data you've seen

0 when you don't have an anchor

HKAnchoredObjectQuery

Anchors

Anchor is the last piece of data you've seen

0 when you don't have an anchor

New anchor with callback

HKAnchoredObjectQuery

Fetching new objects

```
self.lastAnchor = 0
...
HKQuantityType *bloodSugar = ...
HKAnchoredObjectQuery *query;
query = [[HKAnchoredObjectQuery alloc] initWithType:bloodSugar
                                          predicate:nil
                                             anchor:self.lastAnchor
                                              limit:HKObjectQueryNoLimit
                                  completionHandler:^(HKAnchoredObjectQuery *query,
                                                      NSArray *results,
                                                      NSUInteger newAnchor,
                                                      NSError *error) {
    self.lastAnchor = newAnchor;
    NSLog(@"Results: %@", results)
}];
```

HKAnchoredObjectQuery

Fetching new objects

```
self.lastAnchor = 0
...
HKQuantityType *bloodSugar = ...
HKAnchoredObjectQuery *query;
query = [[HKAnchoredObjectQuery alloc] initWithType:bloodSugar
                                          predicate:nil
                                             anchor:self.lastAnchor
                                              limit:HKObjectQueryNoLimit
                                  completionHandler:^(HKAnchoredObjectQuery *query,
                                                      NSArray *results,
                                                      NSUInteger newAnchor,
                                                      NSError *error) {
    self.lastAnchor = newAnchor;
    NSLog(@"Results: %@", results)
}];
```

HKAnchoredObjectQuery

Fetching new objects

```
self.lastAnchor = 0
...
HKQuantityType *bloodSugar = ...
HKAnchoredObjectQuery *query;
query = [[HKAnchoredObjectQuery alloc] initWithType:bloodSugar
                                          predicate:nil
                                             anchor:self.lastAnchor
                                              limit:HKObjectQueryNoLimit
                                  completionHandler:^(HKAnchoredObjectQuery *query,
                                                      NSArray *results,
                                                      NSUInteger newAnchor,
                                                      NSError *error) {
    self.lastAnchor = newAnchor;
    NSLog(@"Results: %@", results)
}];
```

HKAnchoredObjectQuery

Fetching new objects

```
self.lastAnchor = 0
...
HKQuantityType *bloodSugar = ...
HKAnchoredObjectQuery *query;
query = [[HKAnchoredObjectQuery alloc] initWithType:bloodSugar
                                          predicate:nil
                                             anchor:self.lastAnchor
                                              limit:HKObjectQueryNoLimit
                                  completionHandler:^(HKAnchoredObjectQuery *query,
                                                      NSArray *results,
                                                      NSUInteger newAnchor,
                                                      NSError *error) {

    self.lastAnchor = newAnchor;
    NSLog(@"Results: %@", results)
}];
```

HKAnchoredObjectQuery

Fetching new objects

```
self.lastAnchor = 0
...
HKQuantityType *bloodSugar = ...
HKAnchoredObjectQuery *query;
query = [[HKAnchoredObjectQuery alloc] initWithType:bloodSugar
                                          predicate:nil
                                             anchor:self.lastAnchor
                                              limit:HKObjectQueryNoLimit
                                  completionHandler:^(HKAnchoredObjectQuery *query,
                                                      NSArray *results,
                                                      NSUInteger newAnchor,
                                                      NSError *error) {
    self.lastAnchor = newAnchor;
    NSLog(@"Results: %@", results)
}];
```

HKAnchoredObjectQuery

Fetching new objects

```
self.lastAnchor = 0
...
HKQuantityType *bloodSugar = ...
HKAnchoredObjectQuery *query;
query = [[HKAnchoredObjectQuery alloc] initWithType:bloodSugar
                                          predicate:nil
                                             anchor:self.lastAnchor
                                              limit:HKObjectQueryNoLimit
                                  completionHandler:^(HKAnchoredObjectQuery *query,
                                                      NSArray *results,
                                                      NSUInteger newAnchor,
                                                      NSError *error) {

    self.lastAnchor = newAnchor;
    NSLog(@"Results: %@", results)
}];
```

anchor 1

anchor 2

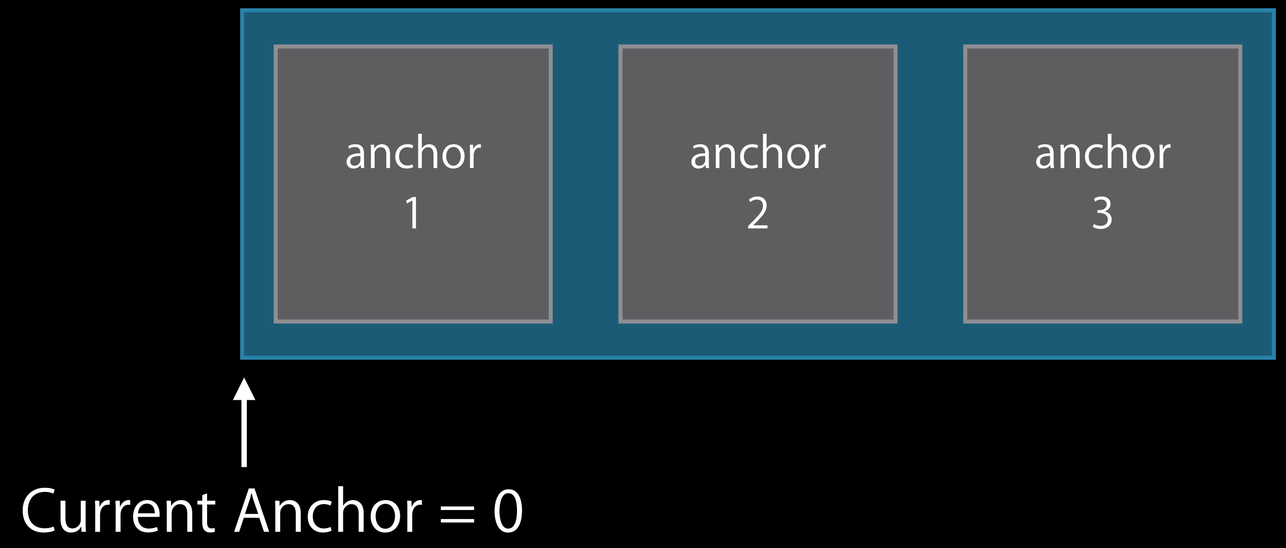
anchor 3

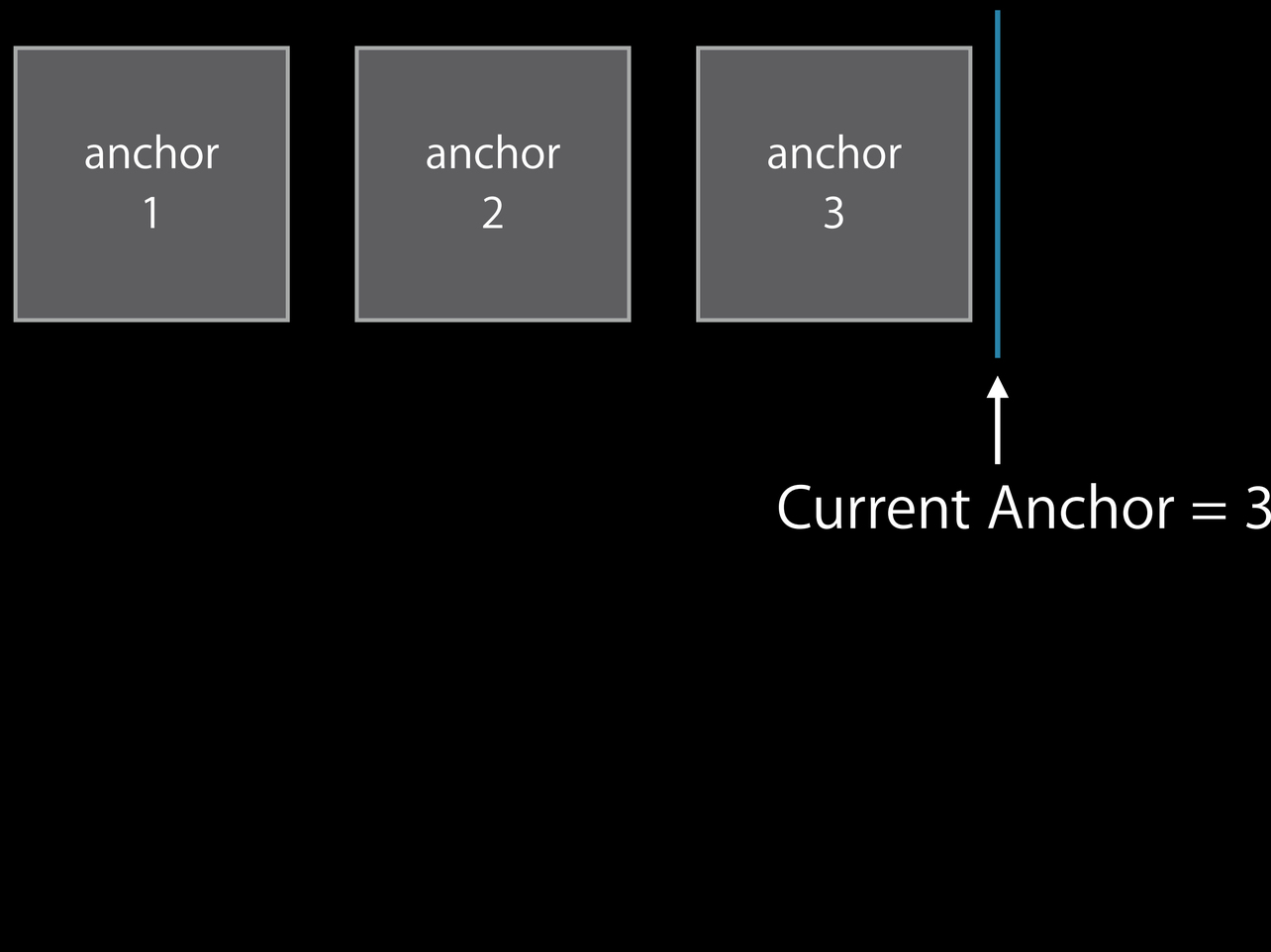
anchor 1

anchor 2

anchor 3

Current Anchor = 0


anchor 1
anchor 2
anchor 3
Current Anchor = 0

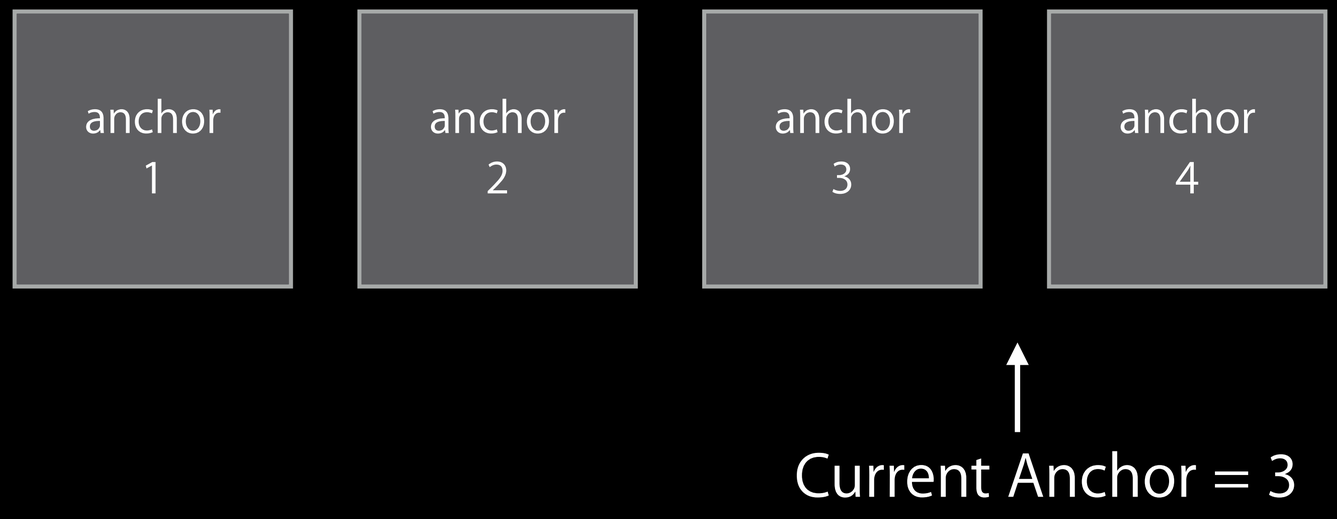

anchor 1
anchor 2
anchor 3
Current Anchor = 3

anchor 1

anchor 2

anchor 3

Current Anchor = 3


anchor 1
anchor 2
anchor 3
anchor 4
Current Anchor = 3

anchor 1

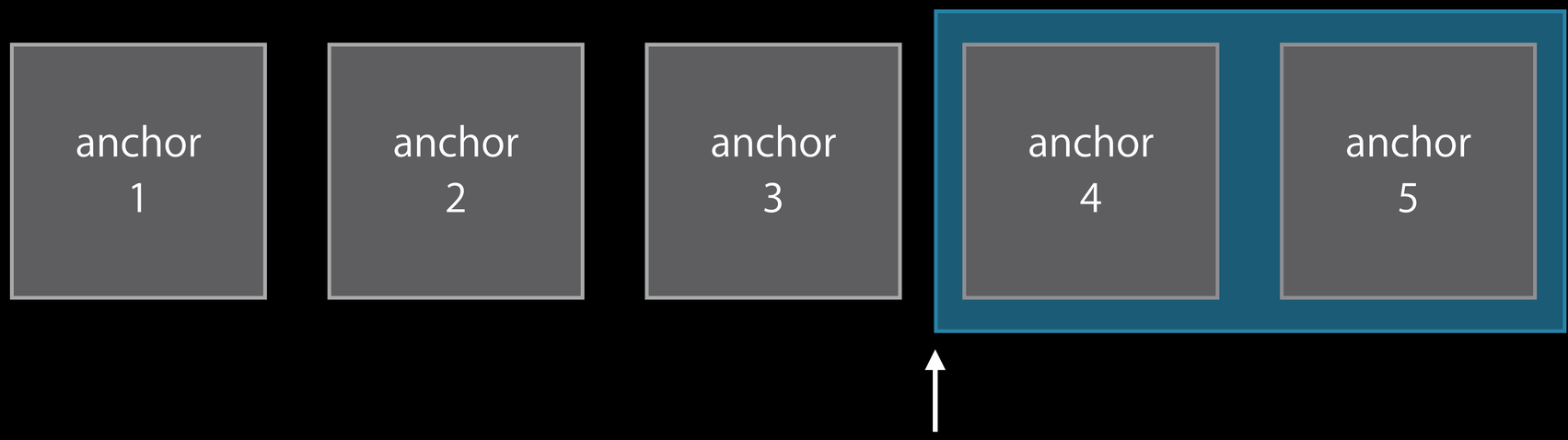
anchor 2

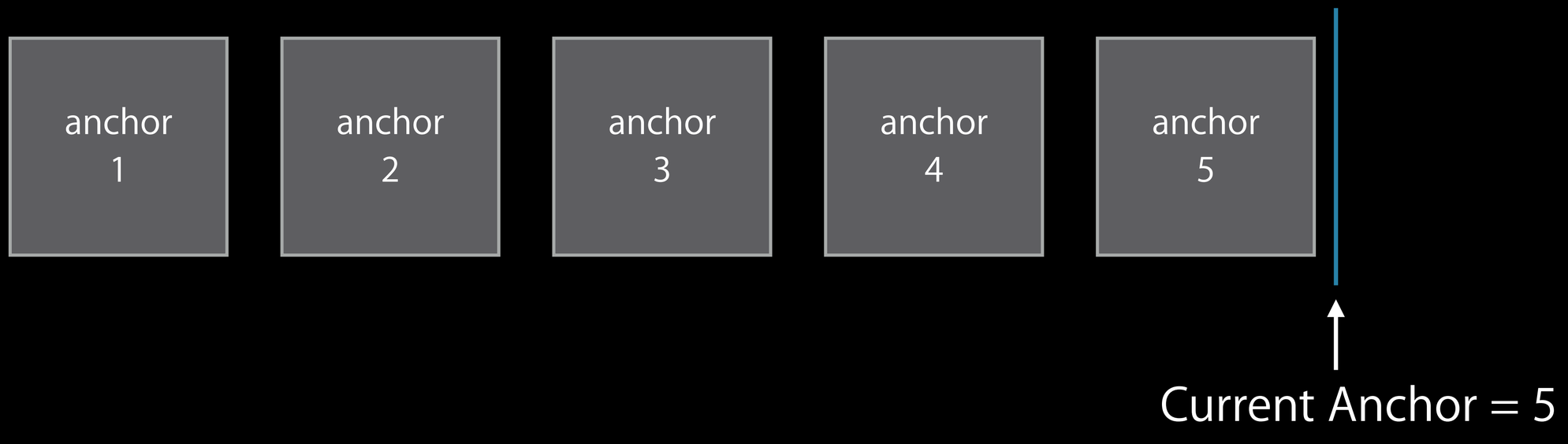
anchor 3

anchor 4

anchor 5

Current Anchor = 3


anchor 1
anchor 2
anchor 3
anchor 4
anchor 5
Current Anchor = 3


anchor 1
anchor 2
anchor 3
anchor 4
anchor 5
Current Anchor = 5

anchor 1

anchor 2

anchor 3

anchor 4

anchor 5

Current Anchor = 5

Executing Queries

HKHealthStore

Queries

```
@interface HKHealthStore : NSObject

- (void)executeQuery:(HKQuery *)query;
- (void)stopQuery:(HKQuery *)query;

@end
```

HKHealthStore

Queries

```
@interface HKHealthStore : NSObject

- (void)executeQuery:(HKQuery *)query;
- (void)stopQuery:(HKQuery *)query;

@end
```

HKHealthStore

Queries

```
@interface HKHealthStore : NSObject

- (void)executeQuery:(HKQuery *)query;
- (void)stopQuery:(HKQuery *)query;

@end
```

HKHealthStore

Queries

You can call stopQuery: multiple times

HKHealthStore

Queries

You can call stopQuery: multiple times

You can call executeQuery: only once

HKHealthStore

Queries

You can call stopQuery: multiple times

You can call executeQuery: only once

Callbacks are invalidated once a query is stopped

HKHealthStore

Queries

You can call stopQuery: multiple times

You can call executeQuery: only once

Callbacks are invalidated once a query is stopped

Only long running queries need to be stopped

Asking for Statistics

HKStatistics

HKStatistics

Sum, Min, Max, Average

HKStatistics

Sum, Min, Max, Average

For all data or for a particular HKSource

HKStatistics

Sum, Min, Max, Average

For all data or for a particular HKSource

Only for quantity types

Classifying Types

Not all types are the same!

Classifying Types

Not all types are the same!

Discrete

- Min, Max, Average

Classifying Types

Not all types are the same!

Discrete

- Min, Max, Average

Cumulative

- Sum

Discrete and Cumulative Types

Discrete and Cumulative Types

HKQuantityTypes are discrete or cumulative

Discrete and Cumulative Types

HKQuantityTypes are discrete or cumulative

Type identifiers are listed with aggregation style

Discrete and Cumulative Types

HKQuantityTypes are discrete or cumulative

Type identifiers are listed with aggregation style

HKQuantityTypes have an aggregationStyle property

```
typedef NS_ENUM(NSInteger, HKQuantityAggregationStyle) {
    HKQuantityAggregationStyleCumulative = 0,
    HKQuantityAggregationStyleDiscrete,
} NS_ENUM_AVAILABLE_IOS(8_0);
```

HKStatistics

Specifying statistics

HKStatistics

Specifying statistics

Calculations can be expensive

HKStatistics

Specifying statistics

Calculations can be expensive

Tell us what you want ahead of time

HKStatistics

Specifying statistics

Calculations can be expensive

Tell us what you want ahead of time

```
typedef NS_OPTIONS(NSUInteger, HKStatisticsOptions) {
    HKStatisticsOptionNone                  = 0,
    HKStatisticsOptionSeparateBySource      = 1 << 0,
    HKStatisticsOptionDiscreteAverage       = 1 << 1,
    HKStatisticsOptionDiscreteMin           = 1 << 2,
    HKStatisticsOptionDiscreteMax           = 1 << 3,
    HKStatisticsOptionCumulativeSum         = 1 << 4,
} NS_ENUM_AVAILABLE_IOS(8_0);
```

HKStatistics

HKStatisticsOptions

```
typedef NS_OPTIONS(NSUInteger, HKStatisticsOptions) {
  HKStatisticsOptionNone                = 0,
  HKStatisticsOptionSeparateBySource    = 1 << 0,
  HKStatisticsOptionDiscreteAverage     = 1 << 1,
  HKStatisticsOptionDiscreteMin         = 1 << 2,
  HKStatisticsOptionDiscreteMax         = 1 << 3,
  HKStatisticsOptionCumulativeSum       = 1 << 4,
} NS_ENUM_AVAILABLE_IOS(8_0);
```

HKStatistics

HKStatisticsOptions

```
typedef NS_OPTIONS(NSUInteger, HKStatisticsOptions) {
  HKStatisticsOptionNone                = 0,
  HKStatisticsOptionSeparateBySource    = 1 << 0,
  HKStatisticsOptionDiscreteAverage     = 1 << 1,
  HKStatisticsOptionDiscreteMin         = 1 << 2,
  HKStatisticsOptionDiscreteMax         = 1 << 3,
  HKStatisticsOptionCumulativeSum       = 1 << 4,
} NS_ENUM_AVAILABLE_IOS(8_0);
```

HKStatistics

Separate by source

	t1	t2	t3	t4	t5
Source A	5	5	2	2	8
Source B	6	5			8

HKStatistics

Separate by source

	t1	t2	t3	t4	t5
Source A	5	5	2	2	8
Source B	6	5			8

HKStatistics

Separate by source

	t1	t2	t3	t4	t5
Source A	5	5	2	2	8
Source B	6	5			8

HKStatistics

Separate by source

	t1	t2	t3	t4	t5
Source A	5	5	2	2	8
Source B	6	5			8

HKStatistics

Separate by source

	t1	t2	t3	t4	t5
Source A	5	5	2	2	8
Source B	6	5			8

Sum = 41

HKStatistics

Separate by source

	t1	t2	t3	t4	t5
Source A	5	5	2	2	8
Source B	6	5			8

HKStatistics

Separate by source

	t1	t2	t3	t4	t5
Source A	5	5	2	2	8
Source B	6	5			8

Sum = 41

HKStatistics

Separate by source

	t1	t2	t3	t4	t5
Source A	5	5	2	2	8
Source B	6	5			8

Sum = 23

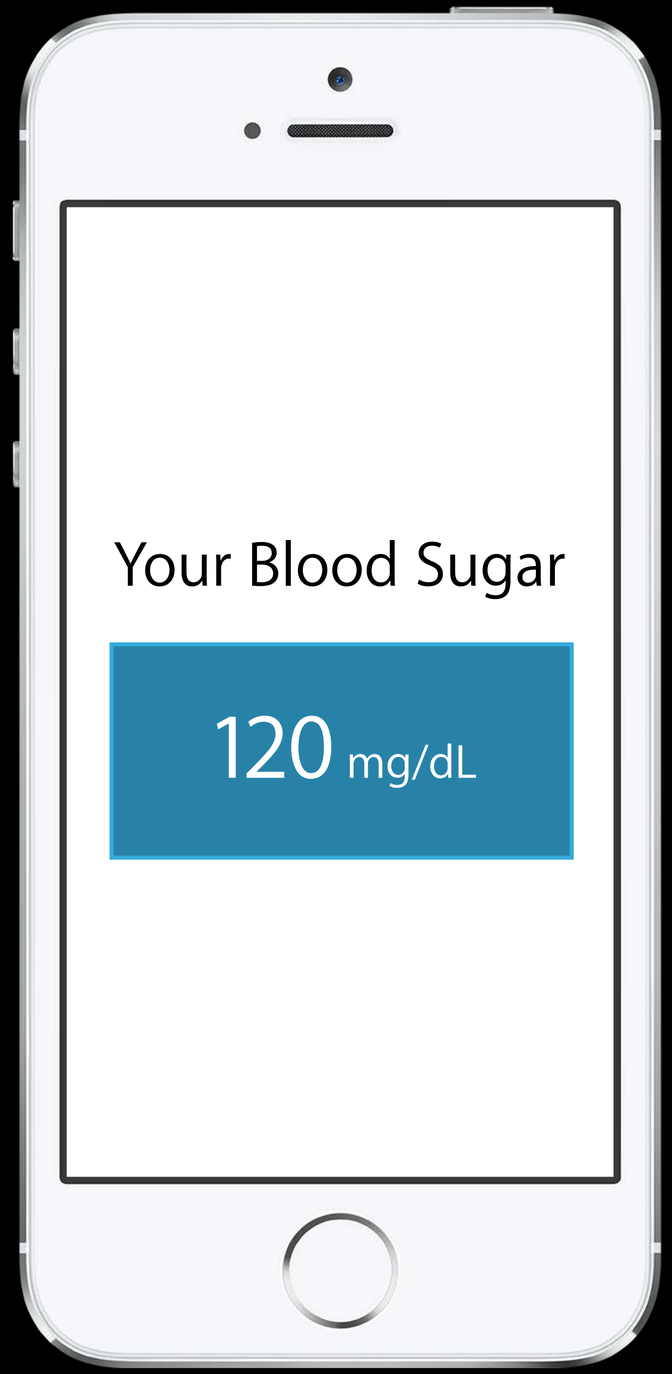
HKStatistics

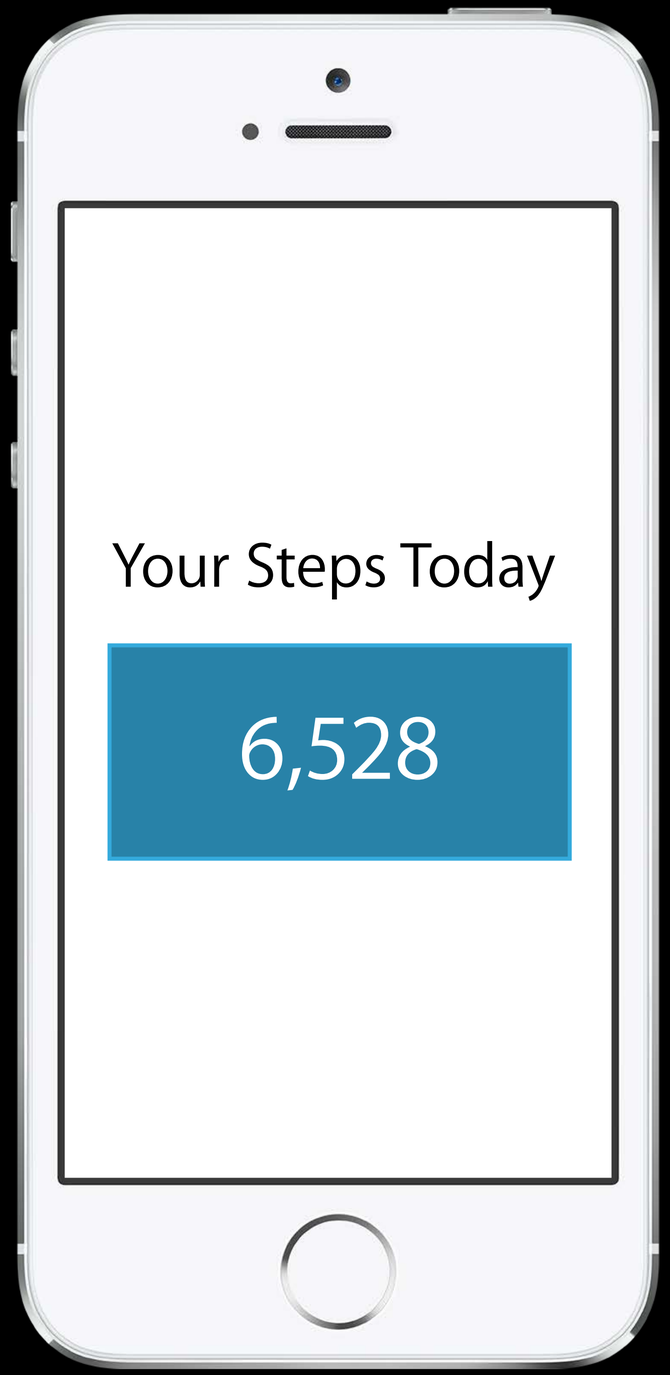
Separate by source

	t1	t2	t3	t4	t5
Source A	5	5	2	2	8
Source B	6	5			8

Source A = 22 steps

Source B = 19 steps


Your Blood Sugar
120 mg/dL


Your Steps Today
6,528

HKStatisticsQuery

HKStatisticsQuery

Takes HKStatisticsOptions

HKStatisticsQuery

Takes HKStatisticsOptions

Gives you back an HKStatistics object

HKStatisticsQuery

Total steps today

```
HKQuantityType *stepCount = ...
NSPredicate *today = ...
HKStatisticsOptions sumOptions = HKStatisticsOptionCumulativeSum;
HKStatisticsQuery *query;
query = [[HKStatisticsQuery alloc] initWithQuantityType:stepCount
                                  quantitySamplePredicate:today
                                                  options:sumOptions
                                        completionHandler:^(HKStatisticsQuery *query,
                                                            HKStatistics *result,
                                                            NSError *error)
{
   HKQuantity *sum = [result sumQuantity];
   NSLog(@"Steps: %lf", [sum doubleValueForUnit:[HKUnit countUnit]]);
}];
```

HKStatisticsQuery

Total steps today

```
HKQuantityType *stepCount = ...
NSPredicate *today = ...
HKStatisticsOptions sumOptions = HKStatisticsOptionCumulativeSum;
HKStatisticsQuery *query;
query = [[HKStatisticsQuery alloc] initWithQuantityType:stepCount
                                  quantitySamplePredicate:today
                                                  options:sumOptions
                                        completionHandler:^(HKStatisticsQuery *query,
                                                            HKStatistics *result,
                                                            NSError *error)
{
   HKQuantity *sum = [result sumQuantity];
   NSLog(@"Steps: %lf", [sum doubleValueForUnit:[HKUnit countUnit]]);
}];
```

HKStatisticsQuery

Total steps today

```
HKQuantityType *stepCount = ...
NSPredicate *today = ...
HKStatisticsOptions sumOptions = HKStatisticsOptionCumulativeSum;
HKStatisticsQuery *query;
query = [[HKStatisticsQuery alloc] initWithQuantityType:stepCount
                                  quantitySamplePredicate:today
                                                  options:sumOptions
                                        completionHandler:^(HKStatisticsQuery *query,
                                                            HKStatistics *result,
                                                            NSError *error)
{
   HKQuantity *sum = [result sumQuantity];
   NSLog(@"Steps: %lf", [sum doubleValueForUnit:[HKUnit countUnit]]);
}];
```

HKStatisticsQuery

Total steps today

```
HKQuantityType *stepCount = ...
NSPredicate *today = ...
HKStatisticsOptions sumOptions = HKStatisticsOptionCumulativeSum;
HKStatisticsQuery *query;
query = [[HKStatisticsQuery alloc] initWithQuantityType:stepCount
                                  quantitySamplePredicate:today
                                                  options:sumOptions
                                        completionHandler:^(HKStatisticsQuery *query,
                                                            HKStatistics *result,
                                                            NSError *error)
{
   HKQuantity *sum = [result sumQuantity];
   NSLog(@"Steps: %lf", [sum doubleValueForUnit:[HKUnit countUnit]]);
}];
```

HKStatisticsQuery

Total steps today

```
HKQuantityType *stepCount = ...
NSPredicate *today = ...
HKStatisticsOptions sumOptions = HKStatisticsOptionCumulativeSum;
HKStatisticsQuery *query;
query = [[HKStatisticsQuery alloc] initWithQuantityType:stepCount
                                  quantitySamplePredicate:today
                                                  options:sumOptions
                                        completionHandler:^(HKStatisticsQuery *query,
                                                            HKStatistics *result,
                                                            NSError *error)
{
   HKQuantity *sum = [result sumQuantity];
   NSLog(@"Steps: %lf", [sum doubleValueForUnit:[HKUnit countUnit]]);
}];
```

HKStatisticsQuery

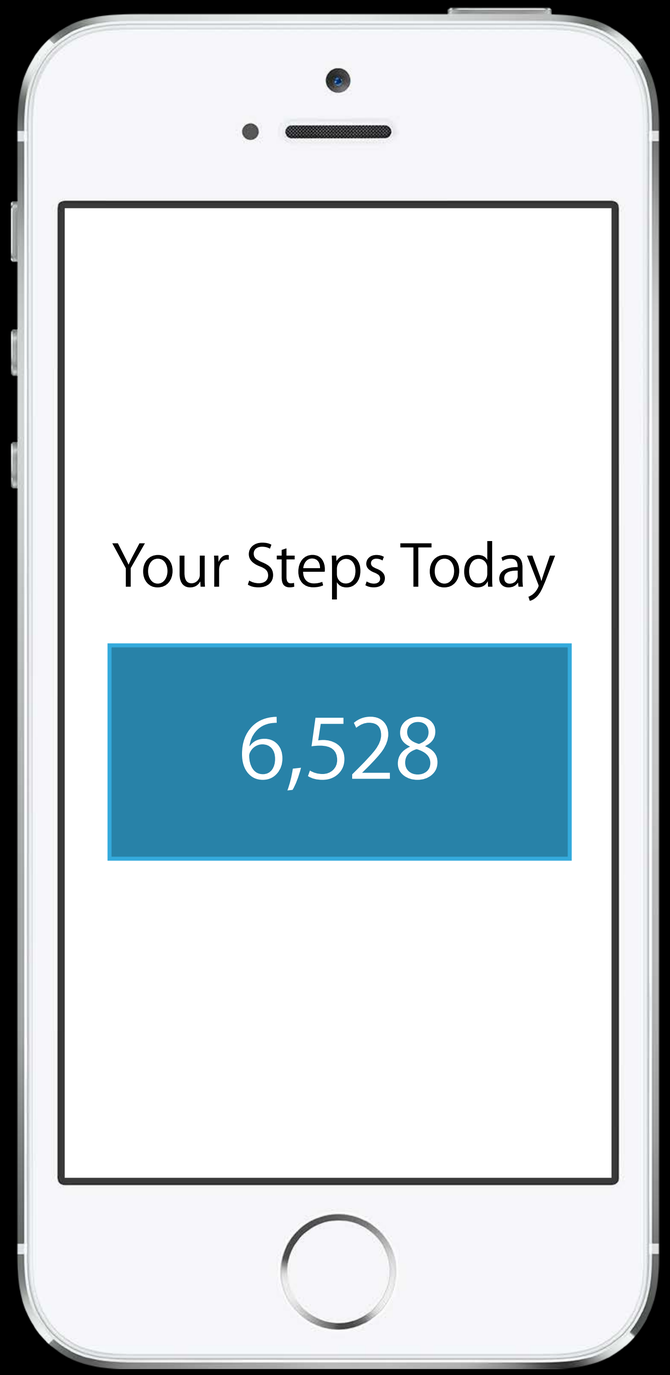
Total steps today

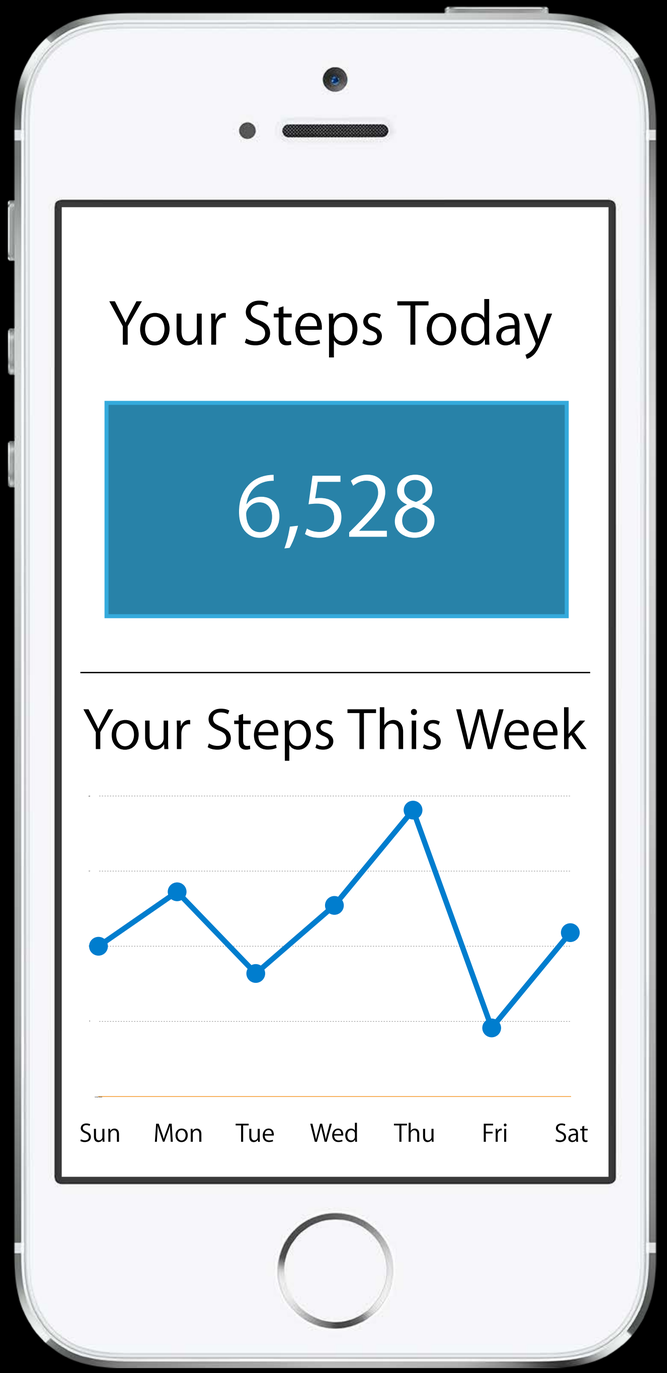
```
HKQuantityType *stepCount = ...
NSPredicate *today = ...
HKStatisticsOptions sumOptions = HKStatisticsOptionCumulativeSum;
HKStatisticsQuery *query;
query = [[HKStatisticsQuery alloc] initWithQuantityType:stepCount
                                  quantitySamplePredicate:today
                                                  options:sumOptions
                                        completionHandler:^(HKStatisticsQuery *query,
                                                            HKStatistics *result,
                                                            NSError *error)
{
   HKQuantity *sum = [result sumQuantity];
   NSLog(@"Steps: %lf", [sum doubleValueForUnit:[HKUnit countUnit]]);
}];
```

HKStatisticsQuery

Total steps today

```
HKQuantityType *stepCount = ...
NSPredicate *today = ...
HKStatisticsOptions sumOptions = HKStatisticsOptionCumulativeSum;
HKStatisticsQuery *query;
query = [[HKStatisticsQuery alloc] initWithQuantityType:stepCount
                                  quantitySamplePredicate:today
                                                  options:sumOptions
                                        completionHandler:^(HKStatisticsQuery *query,
                                                            HKStatistics *result,
                                                            NSError *error)
{
   HKQuantity *sum = [result sumQuantity];
   NSLog(@"Steps: %lf", [sum doubleValueForUnit:[HKUnit countUnit]]);
}];
```


Your Steps Today
6,528


Your Steps Today
6,528
Your Steps This Week
Sun
Mon
Tue
Wed
Thu
Fri
Sat

HKStatisticsCollection

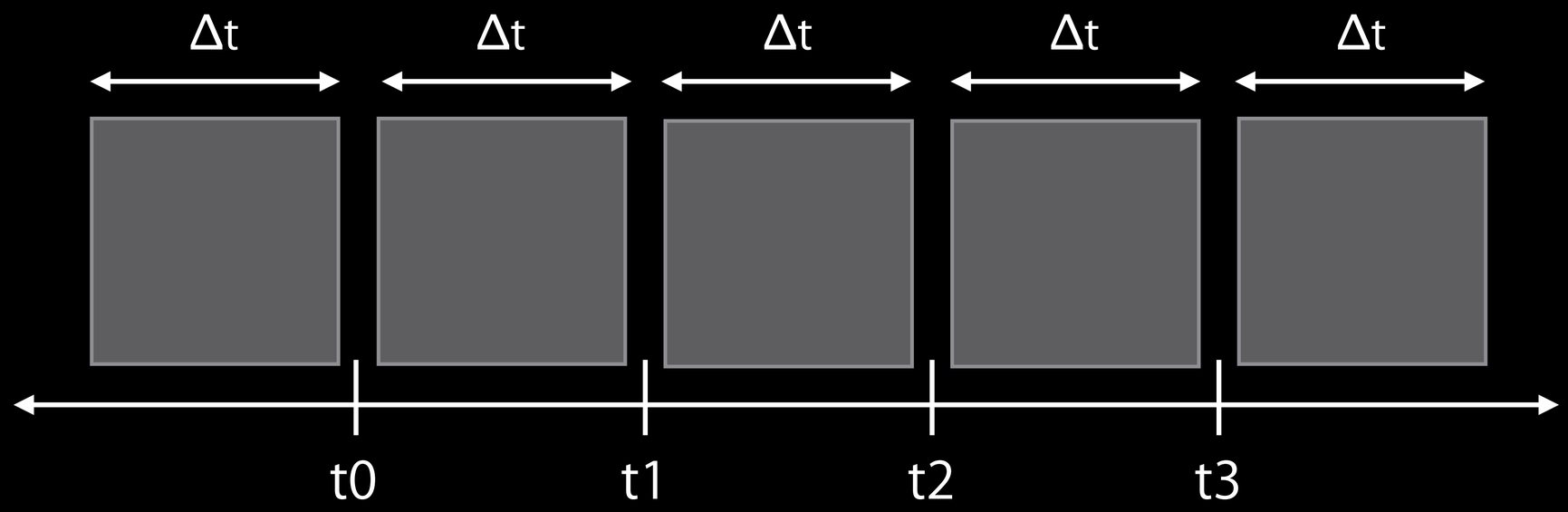
Contains multiple statistics objects

HKStatisticsCollection

Contains multiple statistics objects

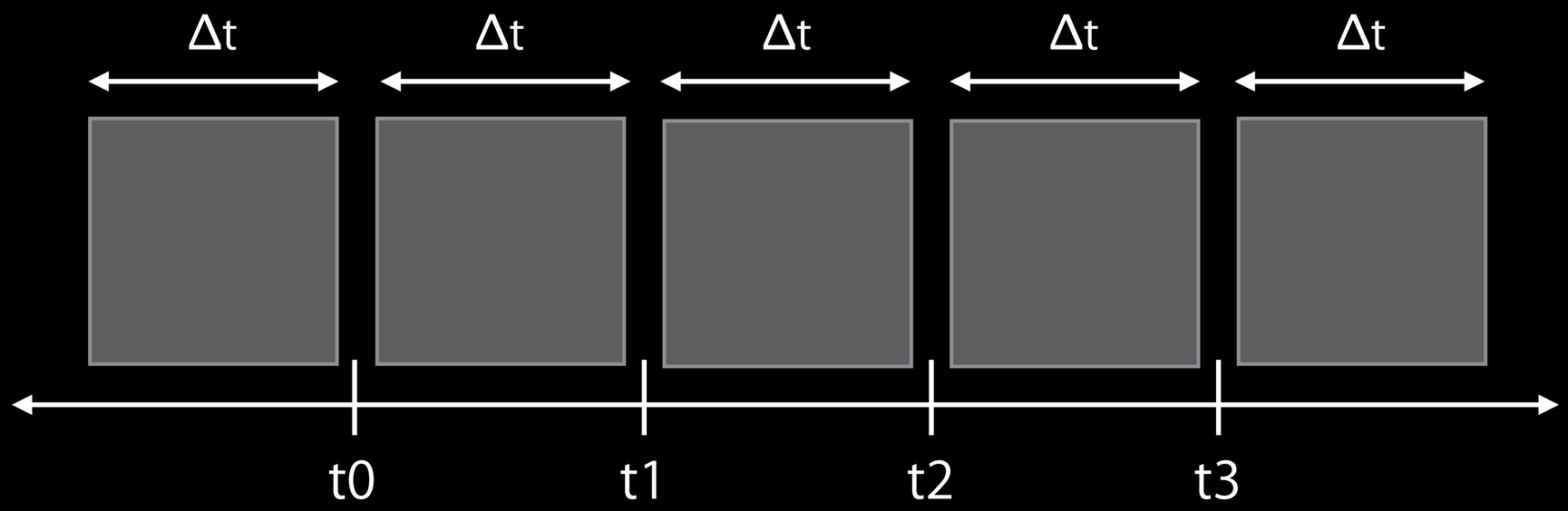
Splits time up into “intervals”

HKStatisticsCollection



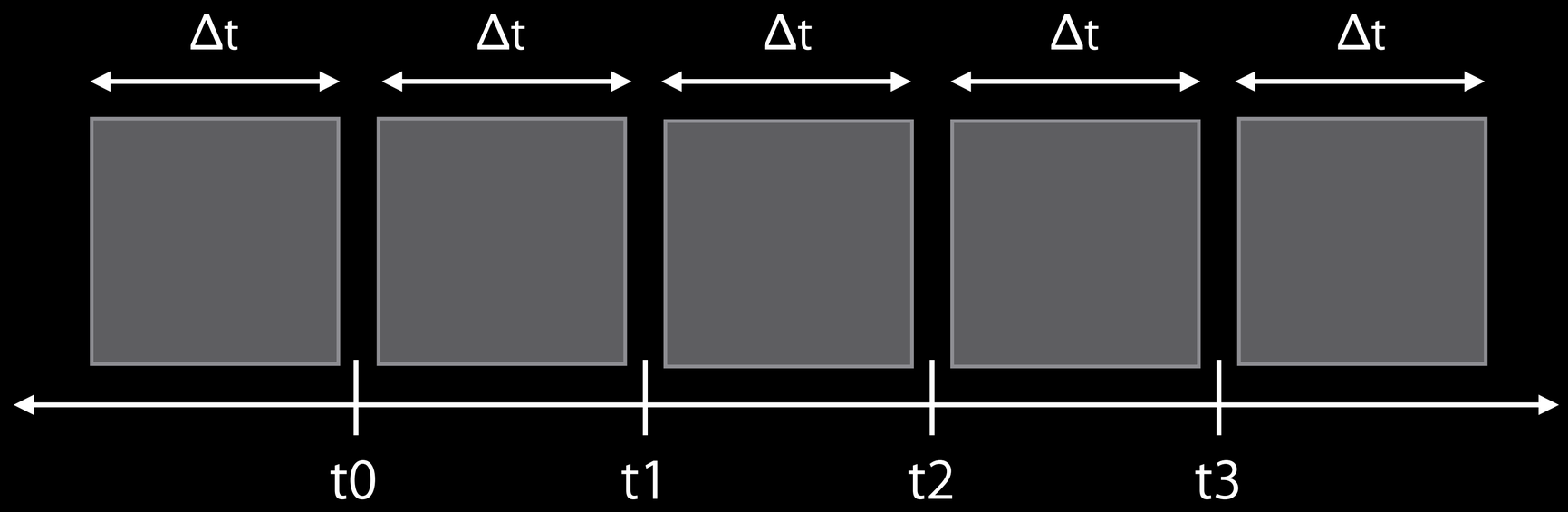
HKStatisticsCollection

Δt = Interval Components (NSDateComponents)



HKStatisticsCollection

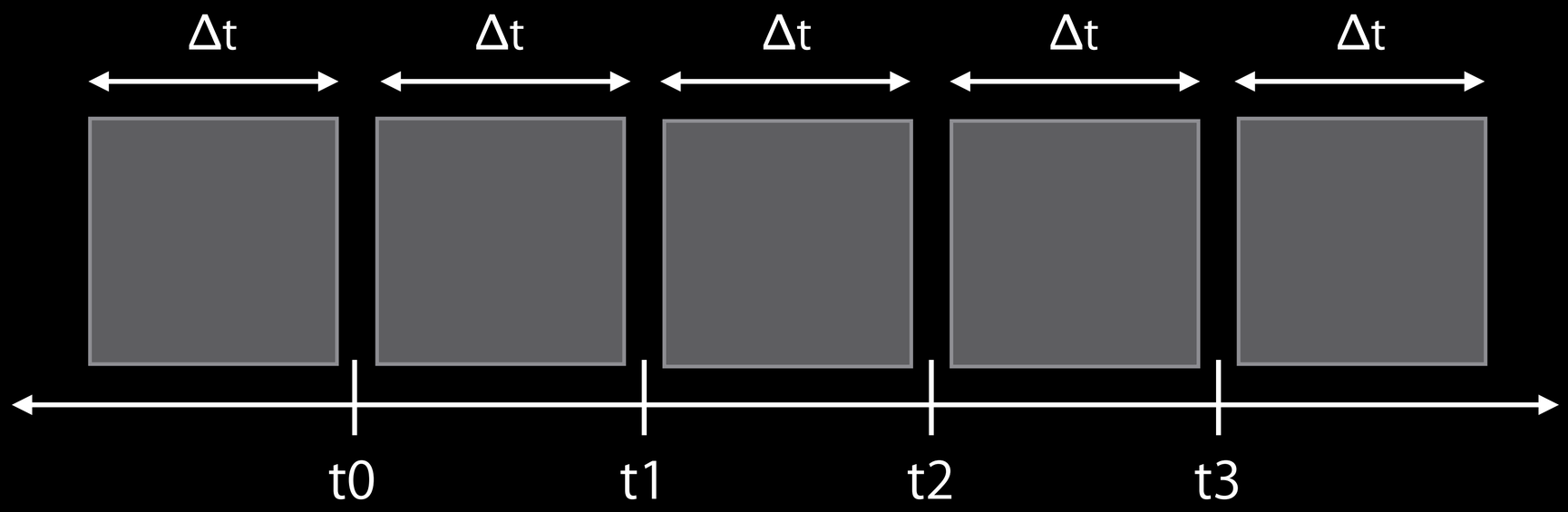
Δt = Interval Components (NSDateComponents)



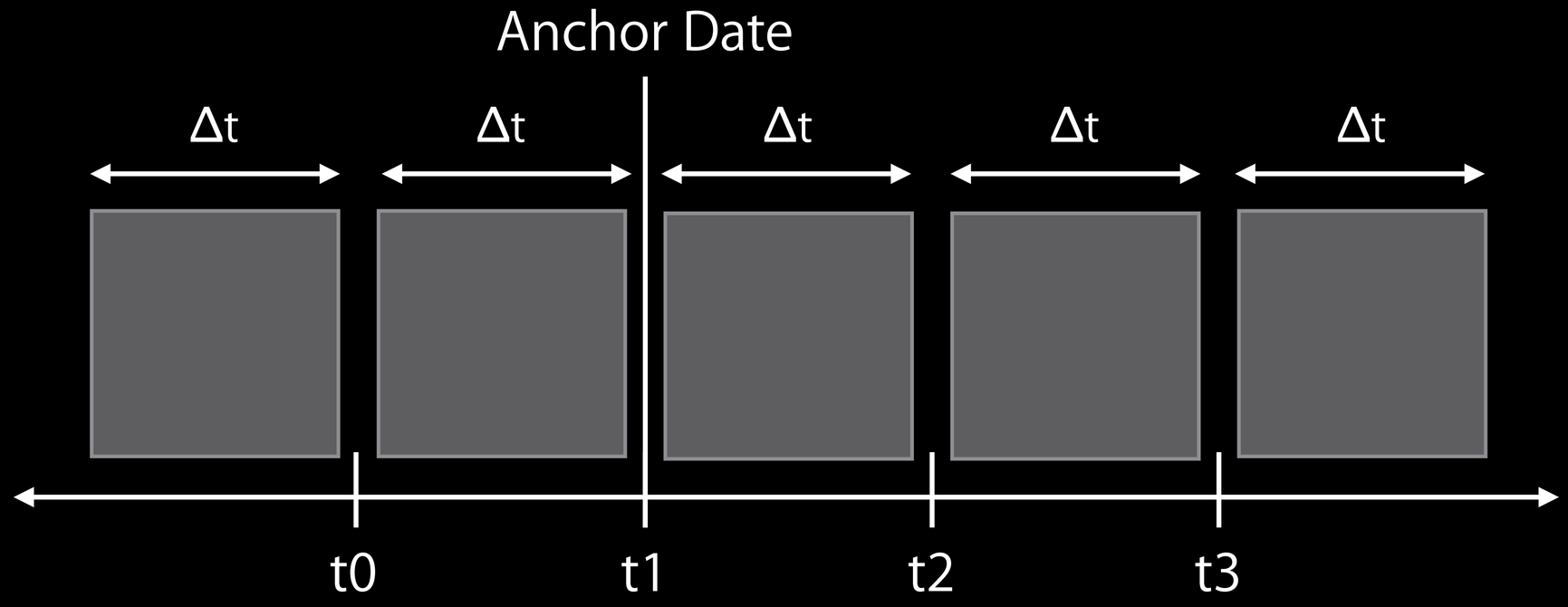
"Solutions to Common Date and Time Challenges" https://developer.apple.com/videos/wwdc/2013/

HKStatisticsCollection

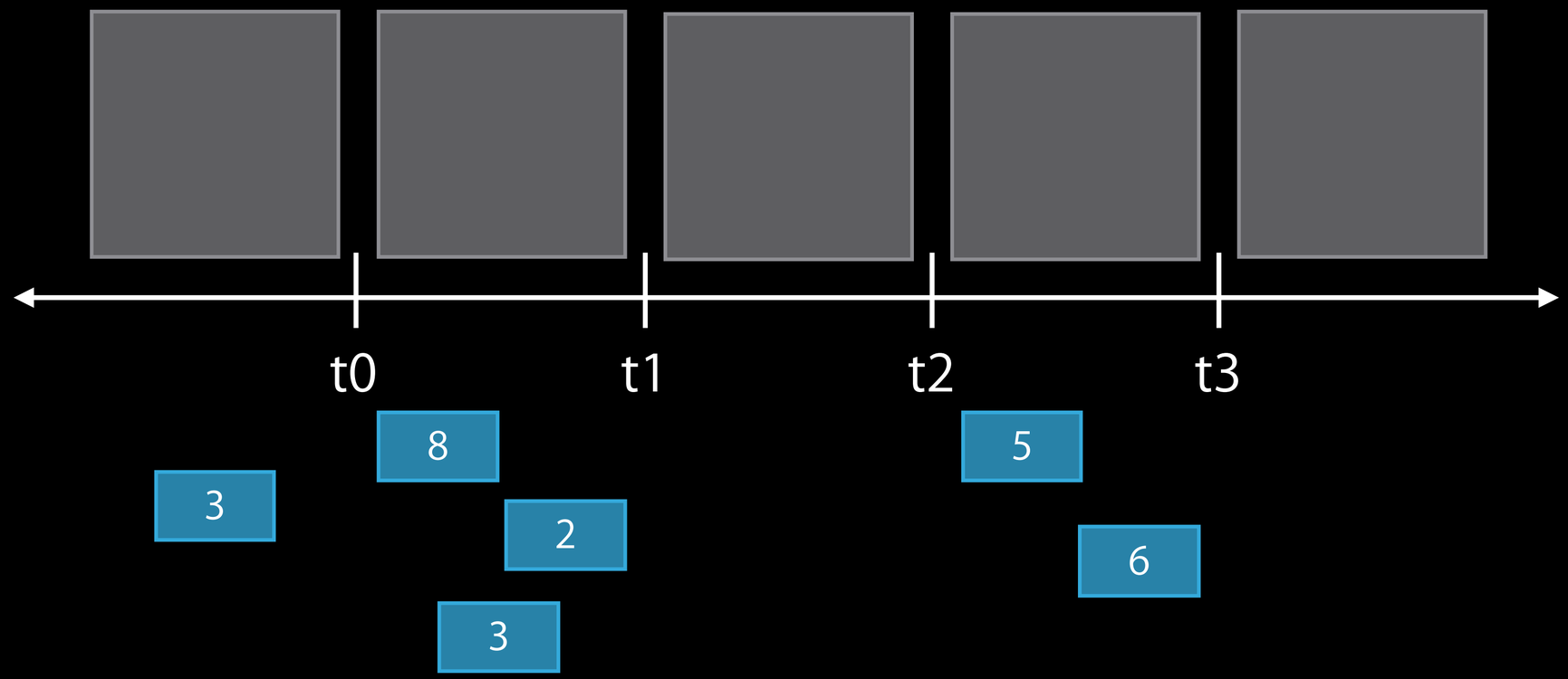
Δt = 1 day



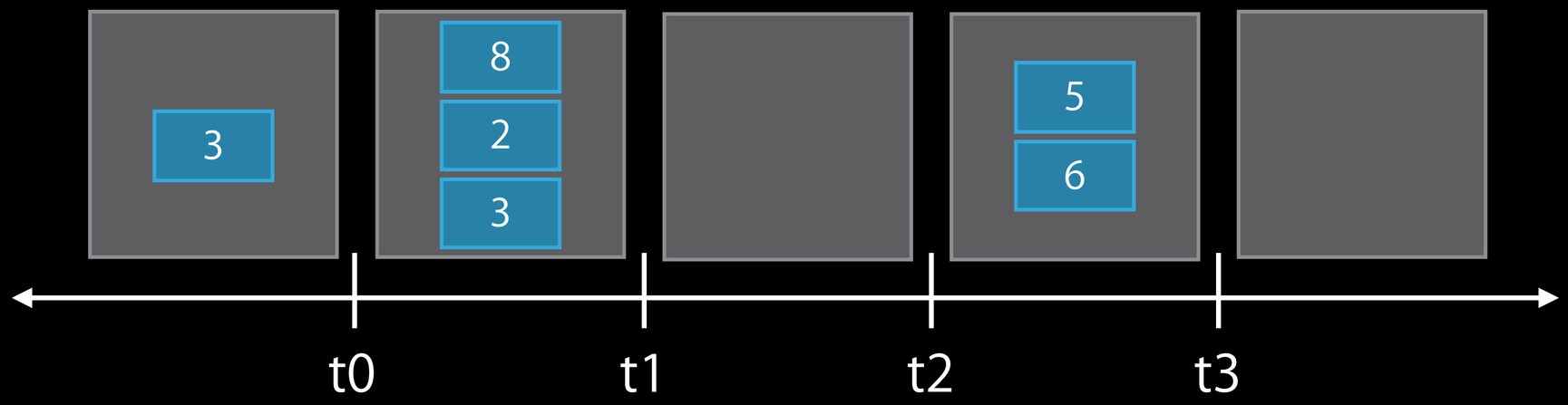
HKStatisticsCollection



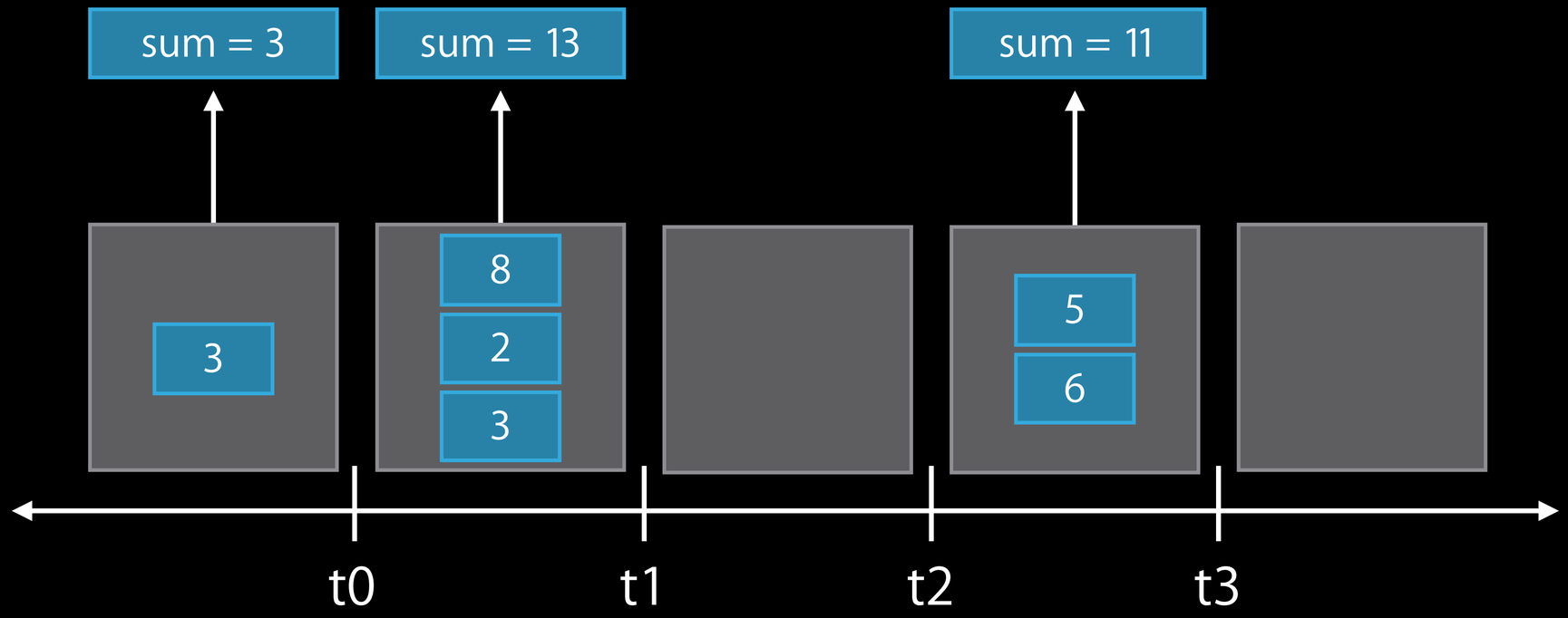
HKStatisticsCollection



HKStatisticsCollection



HKStatisticsCollection



HKStatisticsCollection

Usage

```
@interface HKStatisticsCollection : NSObject

- (NSArray *)statistics;
- (HKStatistics *)statisticsForDate:(NSDate *)date;
- (void)enumerateStatisticsFromDate:(NSDate *)startDate
                             toDate:(NSDate *)endDate
                              block:(void(^)(HKStatistics *stats, BOOL *stop))block;

@end
```

HKStatisticsCollection

Usage

```
@interface HKStatisticsCollection : NSObject

- (NSArray *)statistics;
- (HKStatistics *)statisticsForDate:(NSDate *)date;
- (void)enumerateStatisticsFromDate:(NSDate *)startDate
                             toDate:(NSDate *)endDate
                              block:(void(^)(HKStatistics *stats, BOOL *stop))block;

@end
```

HKStatisticsCollection

Usage

```
@interface HKStatisticsCollection : NSObject

- (NSArray *)statistics;
- (HKStatistics *)statisticsForDate:(NSDate *)date;
- (void)enumerateStatisticsFromDate:(NSDate *)startDate
                             toDate:(NSDate *)endDate
                              block:(void(^)(HKStatistics *stats, BOOL *stop))block;

@end
```

HKStatisticsCollection

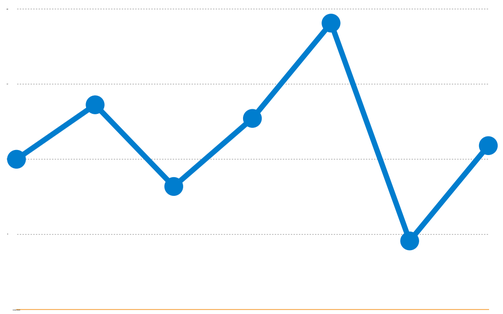
Usage

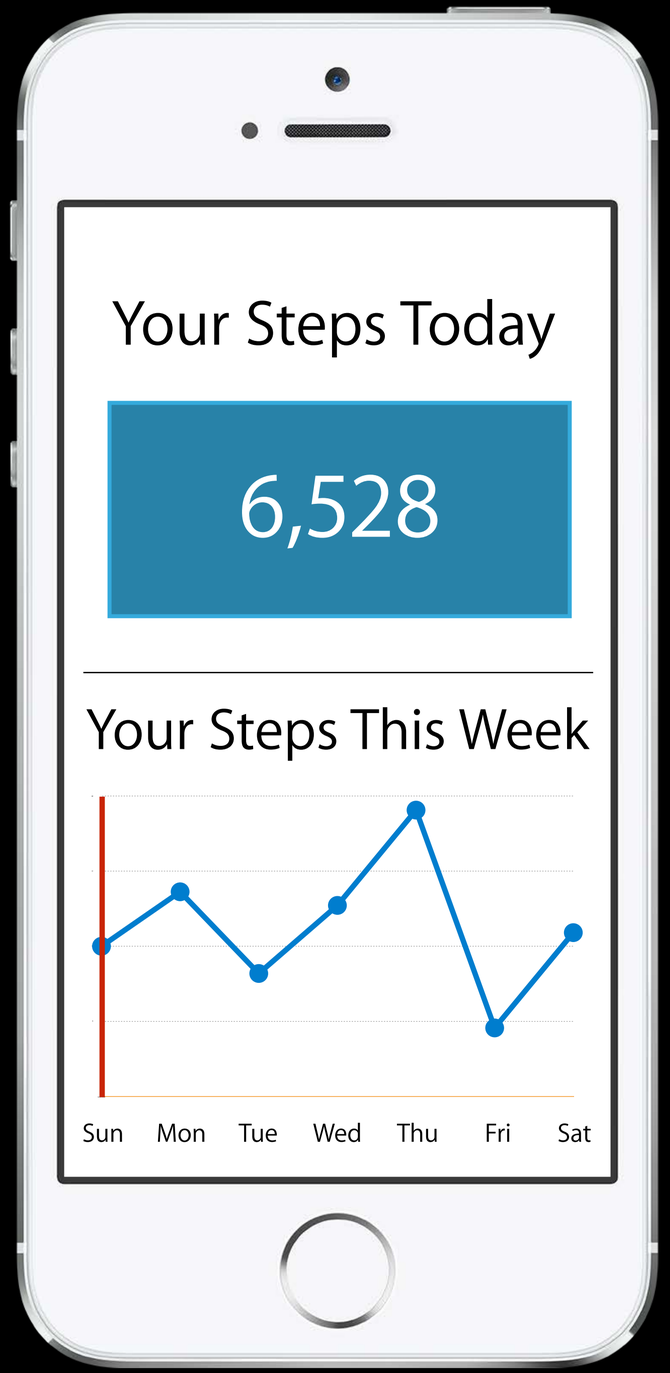
```
@interface HKStatisticsCollection : NSObject

- (NSArray *)statistics;
- (HKStatistics *)statisticsForDate:(NSDate *)date;
- (void)enumerateStatisticsFromDate:(NSDate *)startDate
                             toDate:(NSDate *)endDate
                              block:(void(^)(HKStatistics *stats, BOOL *stop))block;

@end
```

Your Steps Today

6,528

Your Steps This Week

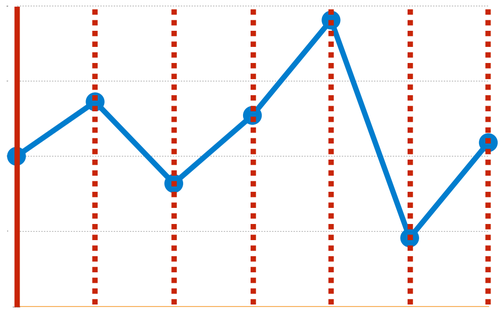



Your Steps Today
6,528
Your Steps This Week
Sun
Mon
Tue
Wed
Thu
Fri
Sat

Your Steps Today

6,528

Your Steps This Week



HKStatisticsCollectionQuery

HKStatisticsCollectionQuery

Statistics options

HKStatisticsCollectionQuery

Statistics options

Anchor date

HKStatisticsCollectionQuery

Statistics options

Anchor date

Interval components

HKStatisticsCollectionQuery

Statistics options

Anchor date

Interval components

Gives you back an HKStatisticsCollection

Demo

Fitness Tracker

Siji Rachel Tom
iOS Software Engineer

Demo

Agenda

Demo

Agenda

Surfacing data from other applications

Demo

Agenda

Surfacing data from other applications

Reading and writing your own data

Demo

Agenda

Surfacing data from other applications

Reading and writing your own data

Queries

Demo

Goal

Net Energy Burn = Total Active Energy Burned - Total Consumed Energy

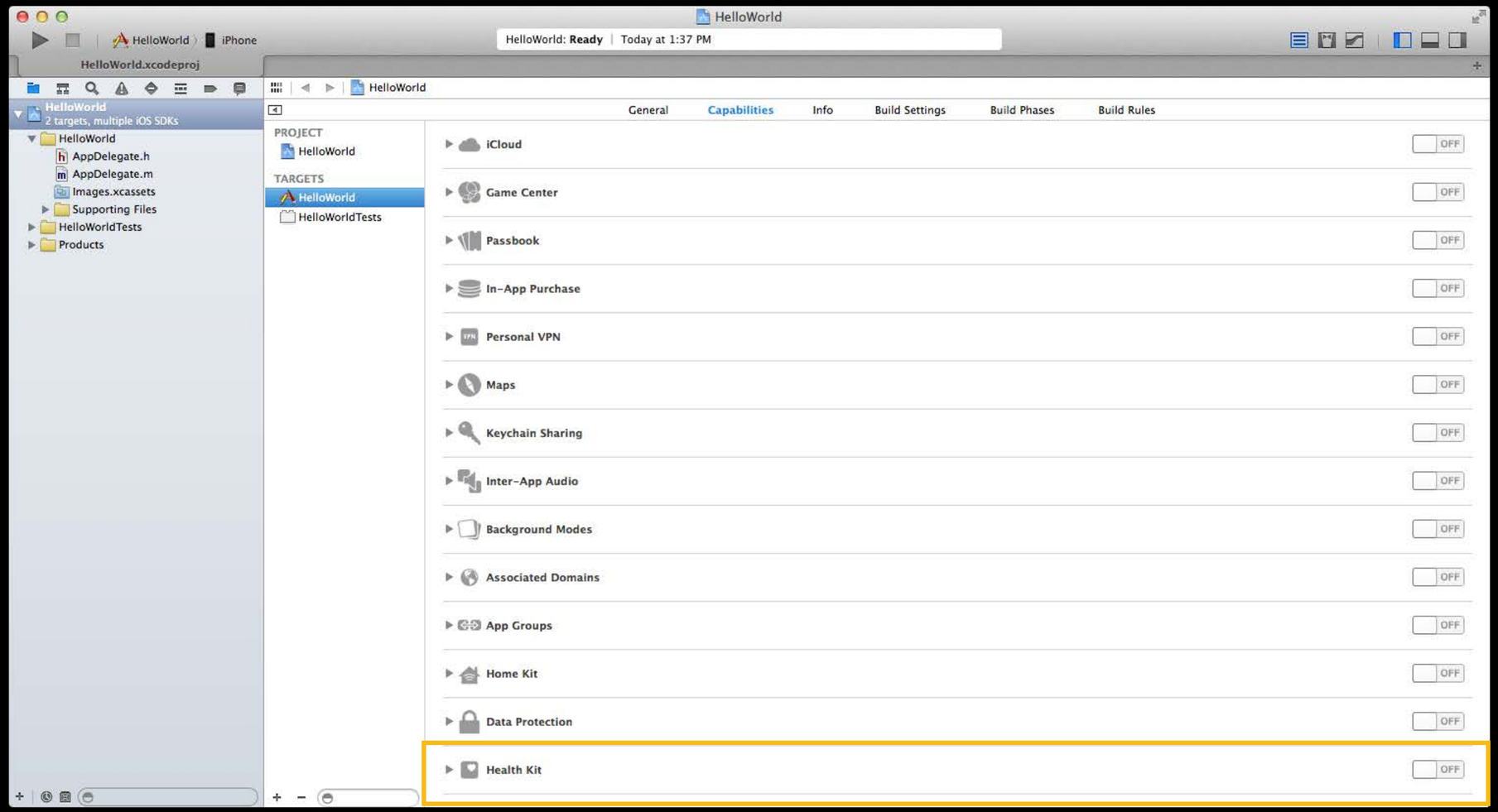
HealthKit Best Practices

Accessing HealthKit

Capabilities

Accessing HealthKit

Capabilities



Privacy and Permissions

Requesting authorization

Health data is sensitive!

Privacy and Permissions

Requesting authorization

Health data is sensitive!

Per object type

Privacy and Permissions

Requesting authorization

Health data is sensitive!

Per object type

Separate read and write

Privacy and Permissions

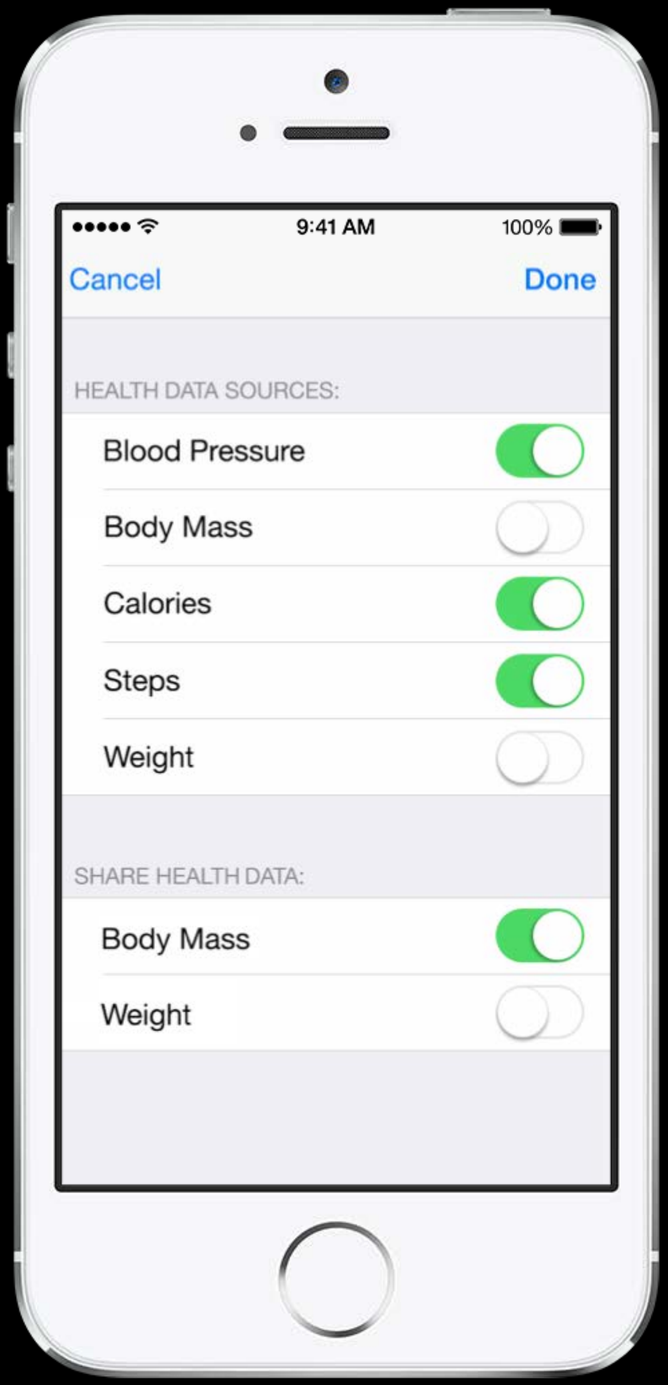
Requesting authorization

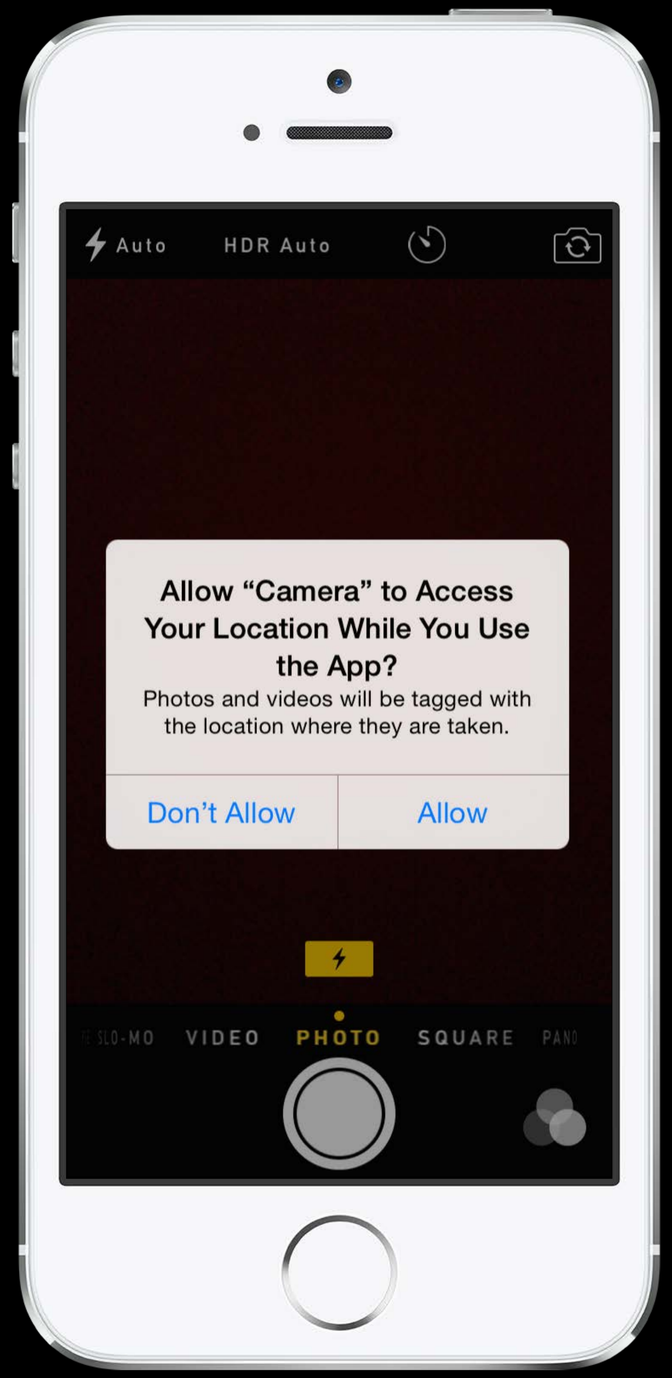
Health data is sensitive!

Per object type

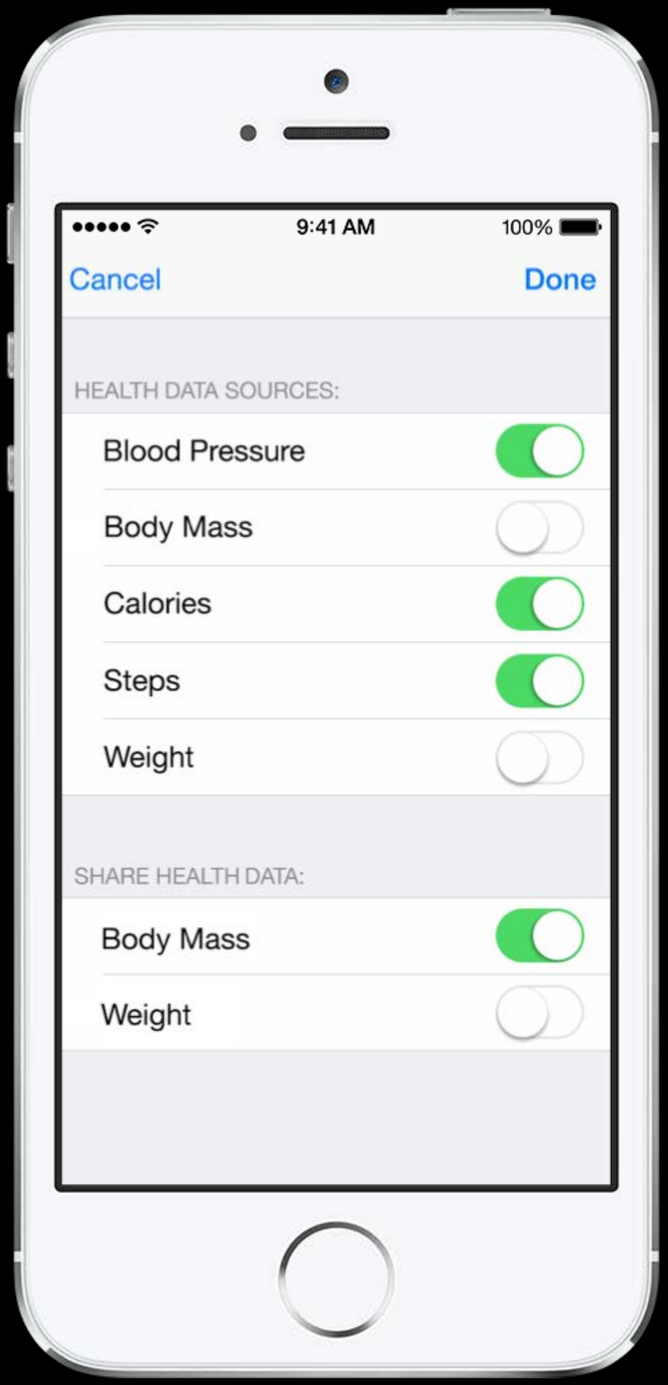
Separate read and write

```
- (void)requestAuthorizationToShareTypes:(NSSet *)typesToShare
                               readTypes:(NSSet *)typesToRead
                              completion:(void(^)(BOOL success, NSError *error))completion;
```


9:41 AM
100%
Cancel
Done
HEALTH DATA SOURCES:
Blood Pressure
Body Mass
Calories
Steps
Weight
SHARE HEALTH DATA:
Body Mass
Weight


Auto
HDR Auto
Allow “Camera” to Access Your Location While You Use the App?
Photos and videos will be tagged with the location where they are taken.
Don’t Allow
Allow
SLO-MO
VIDEO
PHOTO
SQUARE
PANO


9:41 AM
100%
Cancel
Done
HEALTH DATA SOURCES:
Blood Pressure
Body Mass
Calories
Steps
Weight
SHARE HEALTH DATA:
Body Mass
Weight


9:41 AM
100%
Cancel
Done
HEALTH DATA SOURCES:
Blood Pressure
Body Mass
Calories
Steps
Weight
SHARE HEALTH DATA:
Body Mass
Weight

Privacy and Permissions

Requesting authorization

Access to sharing authorization

No access to reading authorization

Privacy and Permissions

Requesting authorization

Access to sharing authorization

No access to reading authorization

```
typedef NS_ENUM(NSInteger, HKAuthorizationStatus) {
    HKAuthorizationStatusNotDetermined = 0,
    HKAuthorizationStatusSharingDenied,
    HKAuthorizationStatusSharingAuthorized,
}

- (HKAuthorizationStatus)authorizationStatusForType:(HKObjectType *)type;
```

Localization

NSFormatter APIs

Localization

NSFormatter APIs

NSNumberFormatter

NSDateFormatter

NSByteCountFormatter

Localization

NSFormatter APIs

NSNumberFormatter

NSDateFormatter

NSByteCountFormatter

NSMassFormatter

NSLengthFormatter

NSEnergyFormatter

Localization

NSFormatter APIs

Weight	NSMassFormatter
Energy	NSEnergyFormatter
Distance	NSLengthFormatter

NSMassFormatter

Formatting strings

```
NSMassFormatter *formatter = [[NSMassFormatter alloc] init];
formatter.forPersonMassUse = YES;

HKQuantity *weight = ...
double weightInKg = [weight doubleValueForUnit:[HKUnit unitFromString:@“kg”]];
NSString *localizedString = [formatter stringFromKilograms:weightInKg];
```

NSMassFormatter

Formatting strings

```
NSMassFormatter *formatter = [[NSMassFormatter alloc] init];
formatter.forPersonMassUse = YES;

HKQuantity *weight = ...
double weightInKg = [weight doubleValueForUnit:[HKUnit unitFromString:@“kg”]];
NSString *localizedString = [formatter stringFromKilograms:weightInKg];
```

NSMassFormatter

Formatting strings

```
NSMassFormatter *formatter = [[NSMassFormatter alloc] init];
formatter.forPersonMassUse = YES;

HKQuantity *weight = ...
double weightInKg = [weight doubleValueForUnit:[HKUnit unitFromString:@“kg”]];
NSString *localizedString = [formatter stringFromKilograms:weightInKg];
```

NSMassFormatter

Formatting strings

```
NSMassFormatter *formatter = [[NSMassFormatter alloc] init];
formatter.forPersonMassUse = YES;

HKQuantity *weight = ...
double weightInKg = [weight doubleValueForUnit:[HKUnit unitFromString:@“kg”]];
NSString *localizedString = [formatter stringFromKilograms:weightInKg];
```

NSMassFormatter

Formatting strings

```
NSMassFormatter *formatter = [[NSMassFormatter alloc] init];
formatter.forPersonMassUse = YES;

HKQuantity *weight = ...
double weightInKg = [weight doubleValueForUnit:[HKUnit unitFromString:@“kg”]];
NSString *localizedString = [formatter stringFromKilograms:weightInKg];
```

NSMassFormatter

Retrieving units

```
NSMassFormatter *formatter = [[NSMassFormatter alloc] init];
formatter.forPersonMassUse = YES;

NSMassFormatterUnit unit;
NSString *localizedString = [formatter unitStringFromKilograms:50
                                                      usedUnit:&unit];
```

Summary

Store and Share Health Data

Summary

Summary

Try out HealthKit

Summary

Try out HealthKit

Download sample code

Summary

Try out HealthKit

Download sample code

Come to our labs

More Information

Dave DeLong
App Frameworks Evangelist
delong@apple.com

David Harrington
Sr. Manager Hardware Evangelist
david@apple.com

Sample Code
Fitness Tracker
https://developer.apple.com/library/ios/fit_sample

Apple Developer Forums
http://devforums.apple.com

Related Sessions

• Designing Accessories for iOS and OS X	Nob Hill	Tuesday 9:00AM

Labs

• HealthKit Lab	Services Lab A	Tuesday 11:30AM
• HealthKit Lab	Services Lab B	Friday 9:00AM

WWDC14