Storing Documents in iCloud Using iOS 5

An introduction to UIDocument

Session 116

Luke the Hiesterman
iOS Frameworks Engineer

Agenda

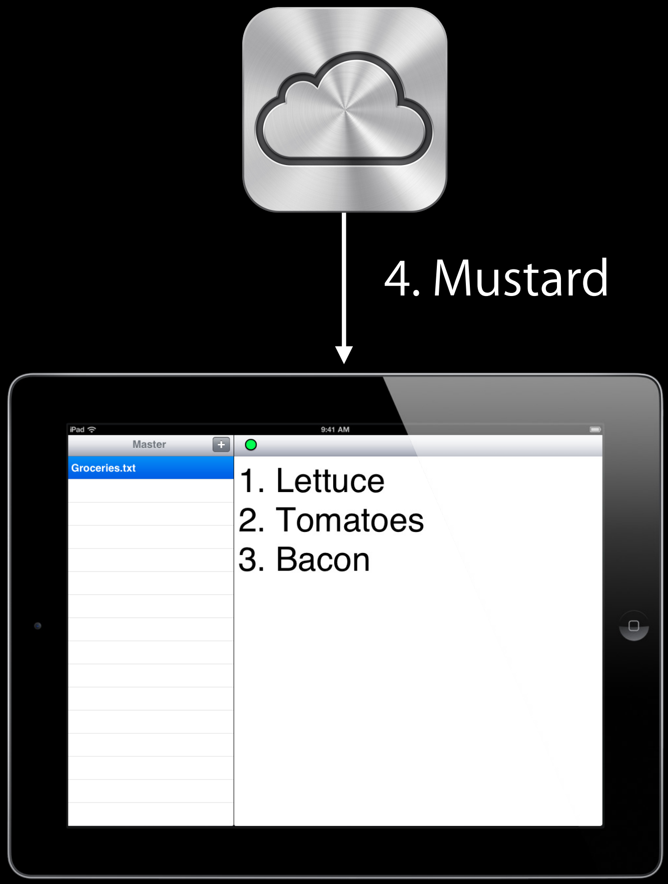
- What we’re trying to solve
- Basic hooks
- Handling edge cases
- Advanced concepts

Building Document Applications

Document-Handling Challenges

- Handling saving and opening
  - Block the main thread?
  - Deal with concurrent threads?
  - When should I save?

iCloud Challenges

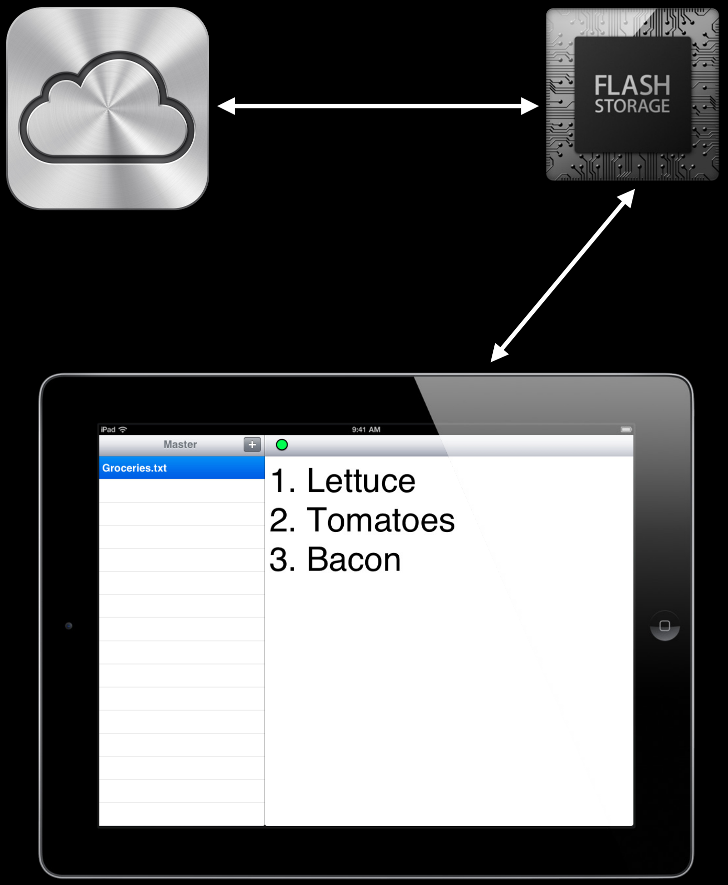


4. Mustard

1. Lettuce
2. Tomatoes
3. Bacon

- Respond to updates from iCloud

iCloud Challenges

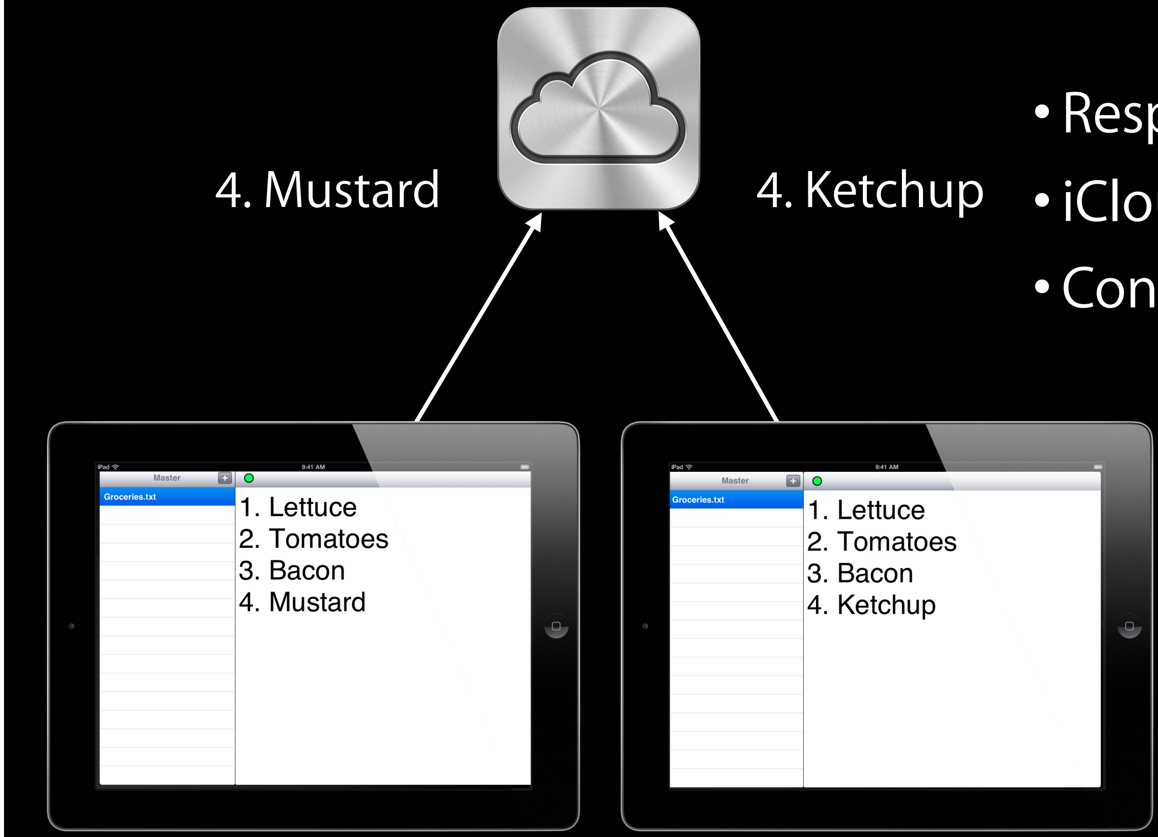


Groceries.txt

1. Lettuce
2. Tomatoes
3. Bacon

- Respond to updates from iCloud
- iCloud-safe reading and writing

iCloud Challenges



4. Mustard

4. Ketchup

1. Lettuce
2. Tomatoes
3. Bacon
4. Mustard

1. Lettuce
2. Tomatoes
3. Bacon
4. Ketchup

- Respond to updates from iCloud
- iCloud-safe reading and writing
- Conflict discovery

The Solution

- UIDocument solves all of these problems!

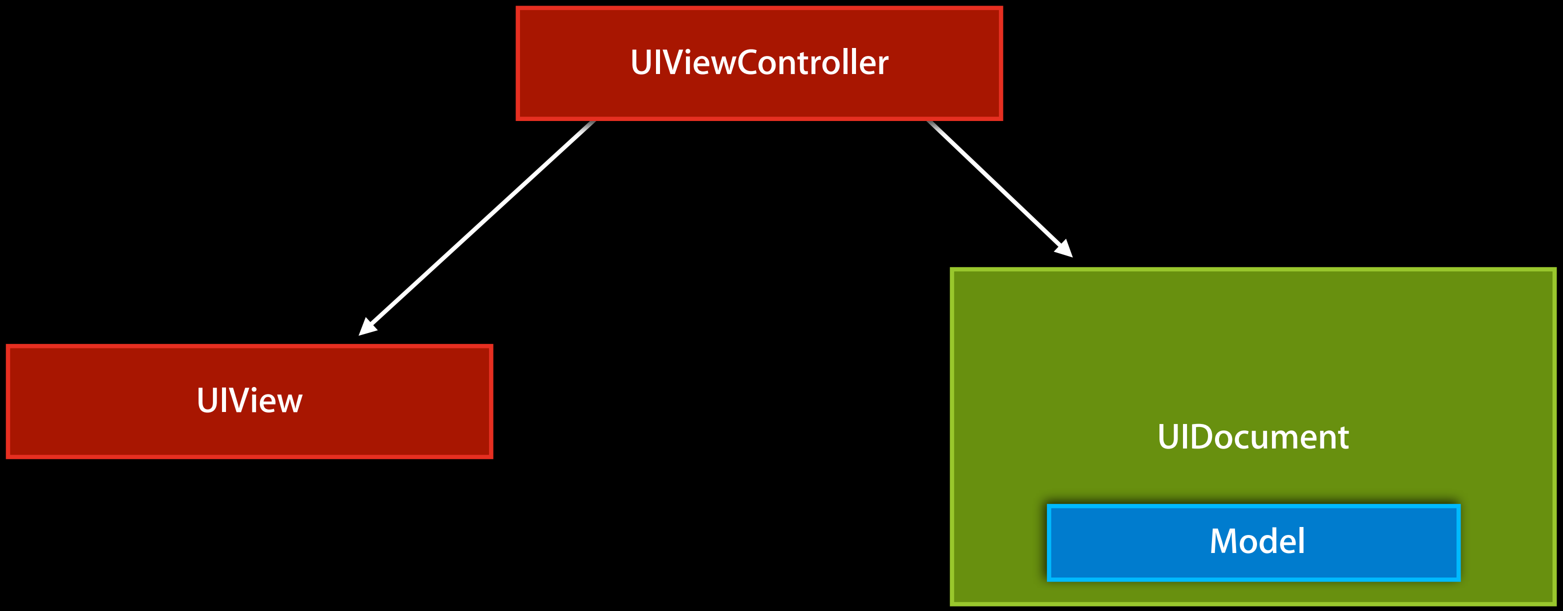
UIDocument Basics

What UIDocument Provides

- Document model base class
- Key document-handling features
    - Concurrent background reading and writing
    - Saveless user model
    - iCloud integration
- UIManagedDocument
    - CoreData integration
    - Optimized for iCloud syncing
    - Automatic conflict resolution

Document Application Architecture

Using UIDocument as a model controller



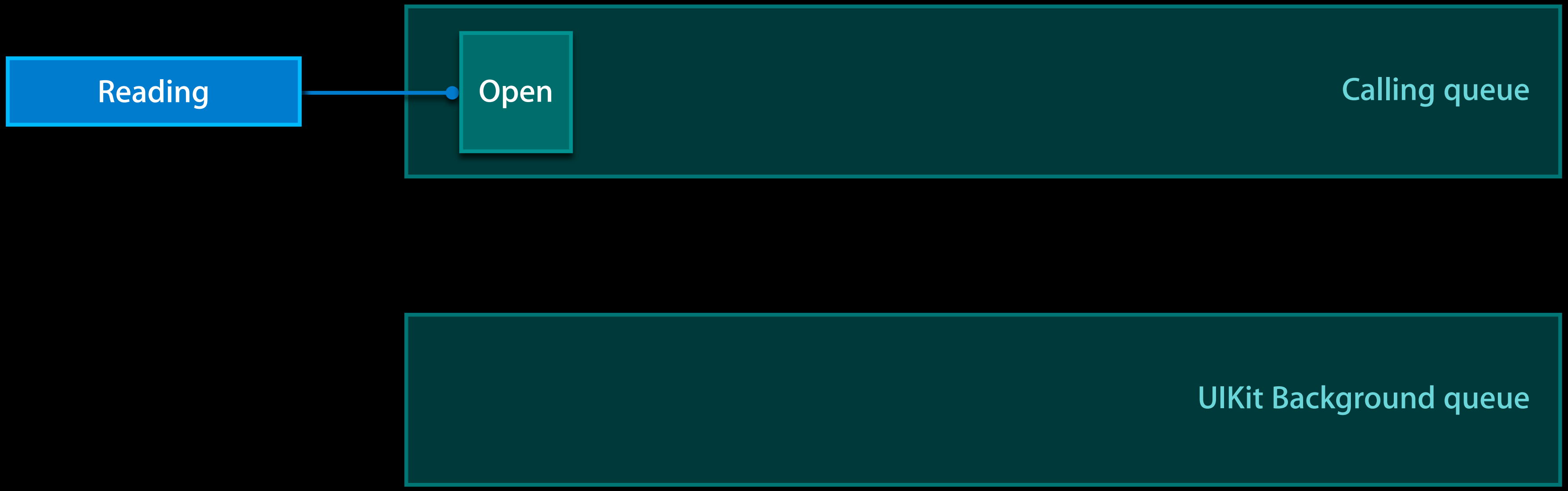
Reading and Writing

The simple story

- Background queue
- Reading triggered by opening or reverting
- Writing triggered by saving
- Automatic handling for NSData and NSFileWrapper

Reading

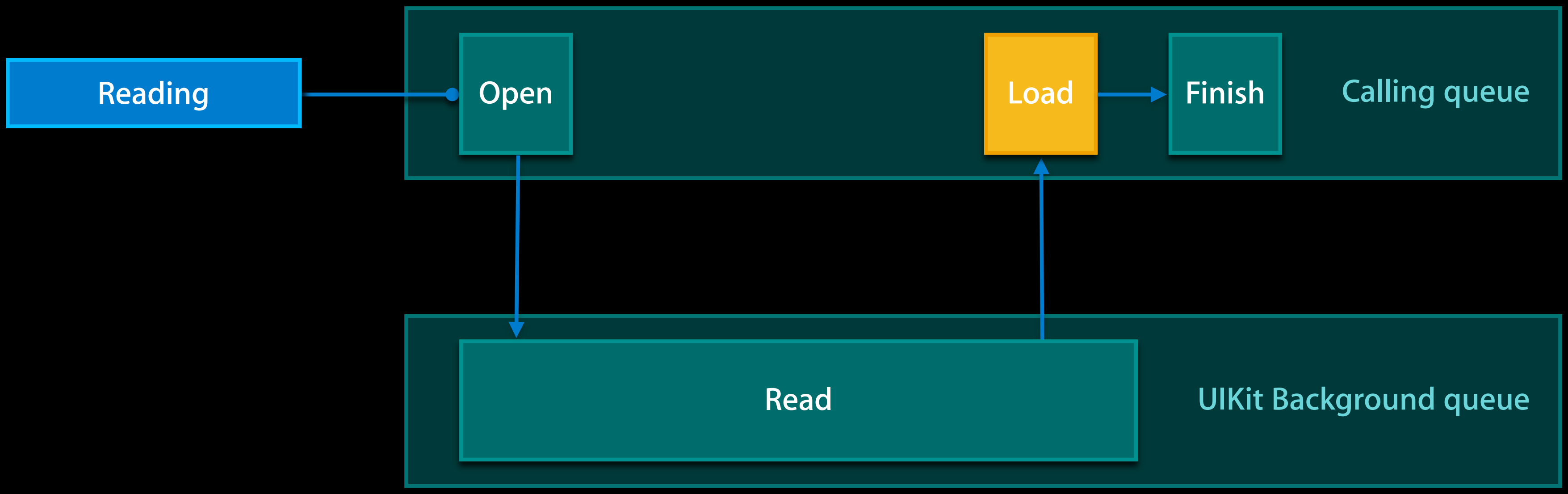
The simple story



```
- (void)openWithCompletionHandler:(void (^)(BOOL success))
completionHandler;
```

Reading

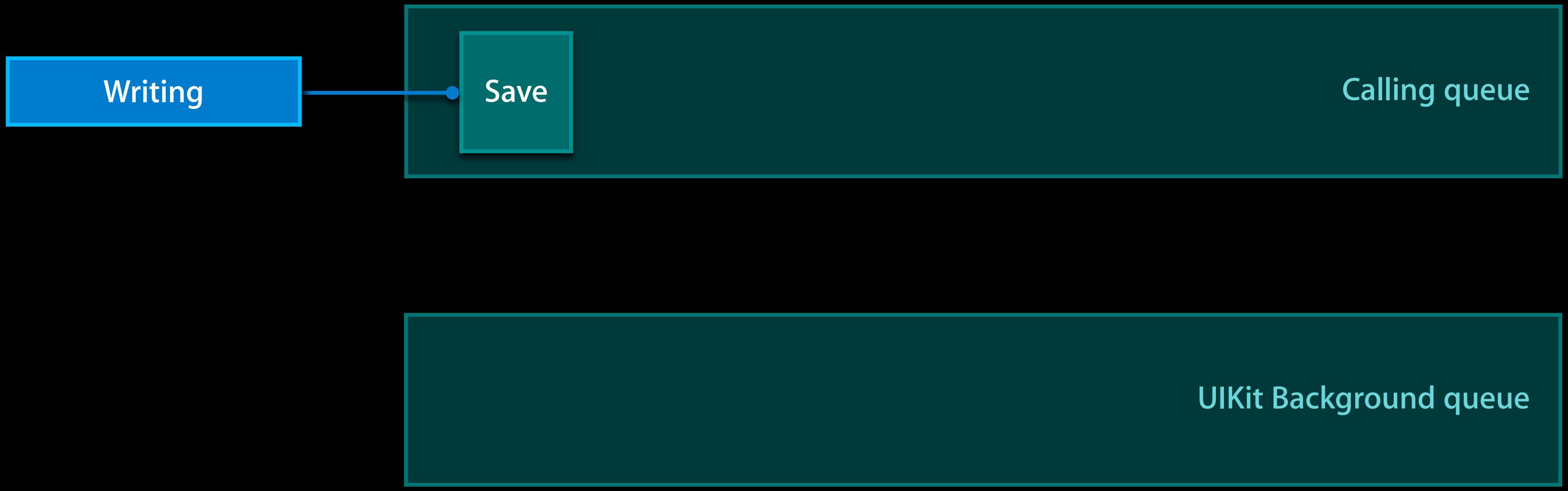
The simple story



```
- (BOOL)loadFromContents:(id)contents
                  ofType:(NSString *)typeName
                   error:(NSError **)outError;
```

Writing

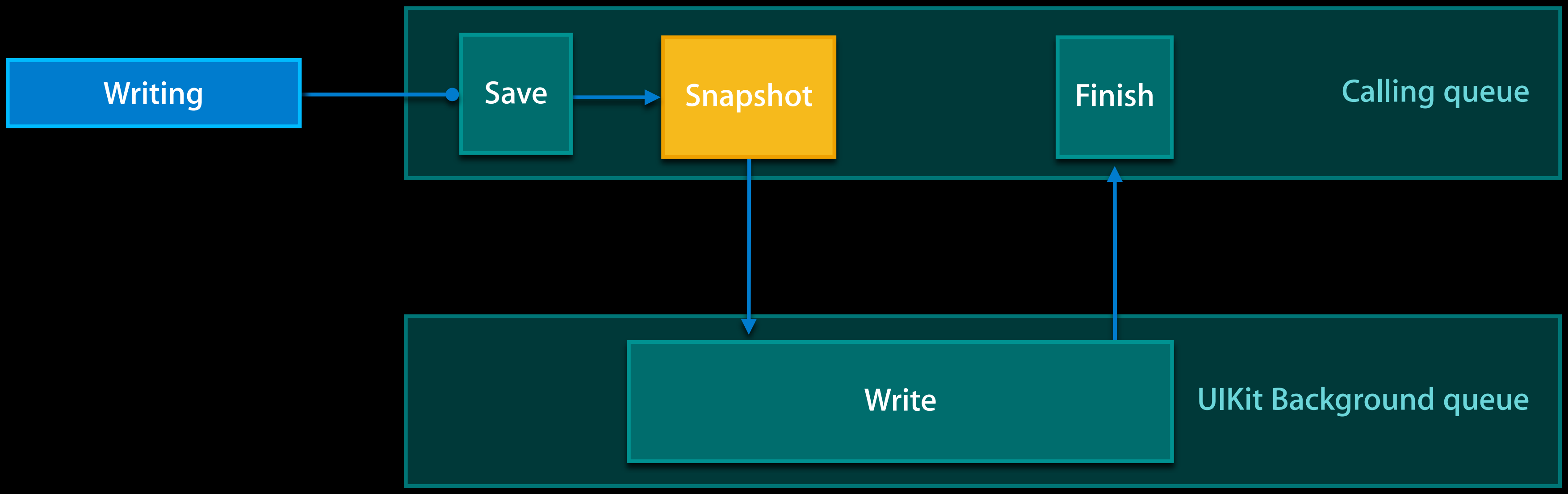
The simple story



```
- (BOOL)saveToURL:(NSURL *)url
 forSaveOperation:(UIDocumentSaveOperation)saveOperation
completionHandler:(void (^)(BOOL success))completionHandler;
```

Writing

The simple story

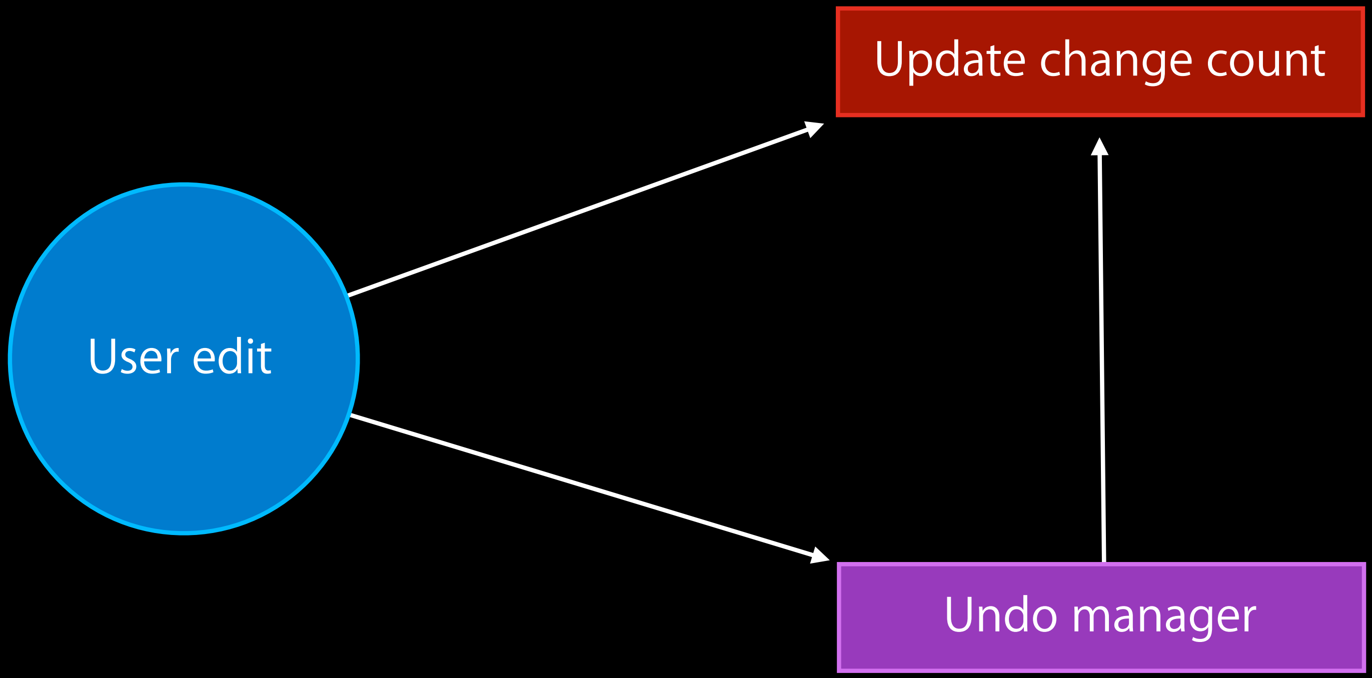


```
- (id)contentsForType:(NSString *)type
                error:(NSError *)error;
```

User Edits and Autosaving

Handling Changes

The simple story



Demo

Handling Conflicts and Errors

Observing Document State

- `documentState` provides information related to when user edits might not be safe
- Observe state changes with `UIDocumentStateChangedNotification`

The Document States

- `UIDocumentStateNormal` = everything is fine
- Nonnormal states

  `UIDocumentStateClosed`

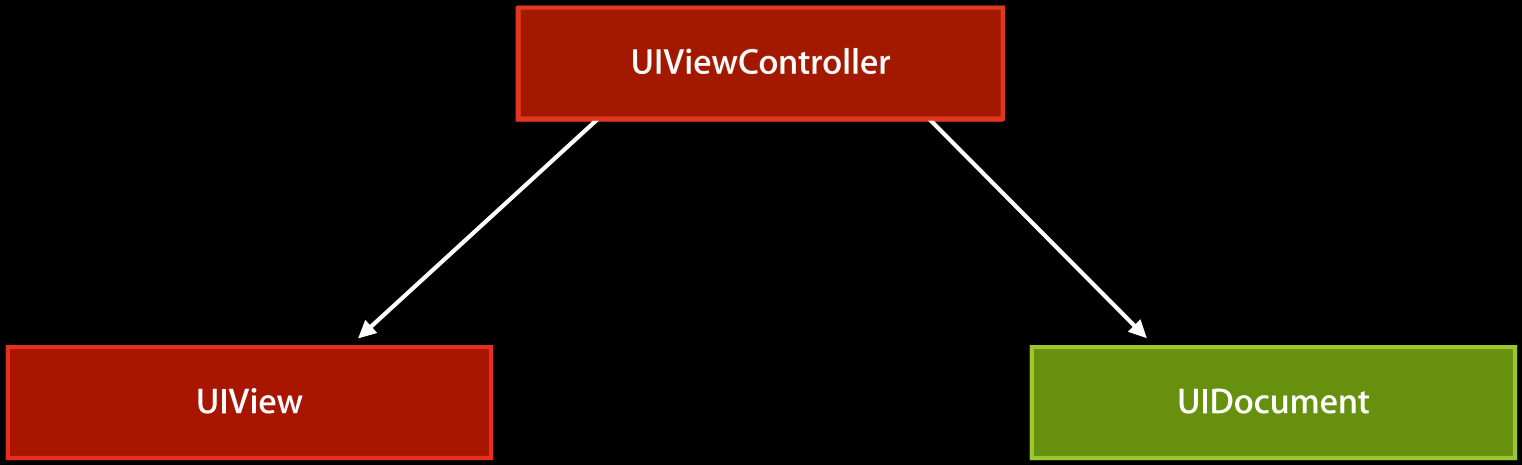
  `UIDocumentStateInConflict`

  `UIDocumentStateSavingError`

  `UIDocumentStateEditingDisabled`

Document Application Architecture

Responding to errors and conflicts



UIViewController

UIView

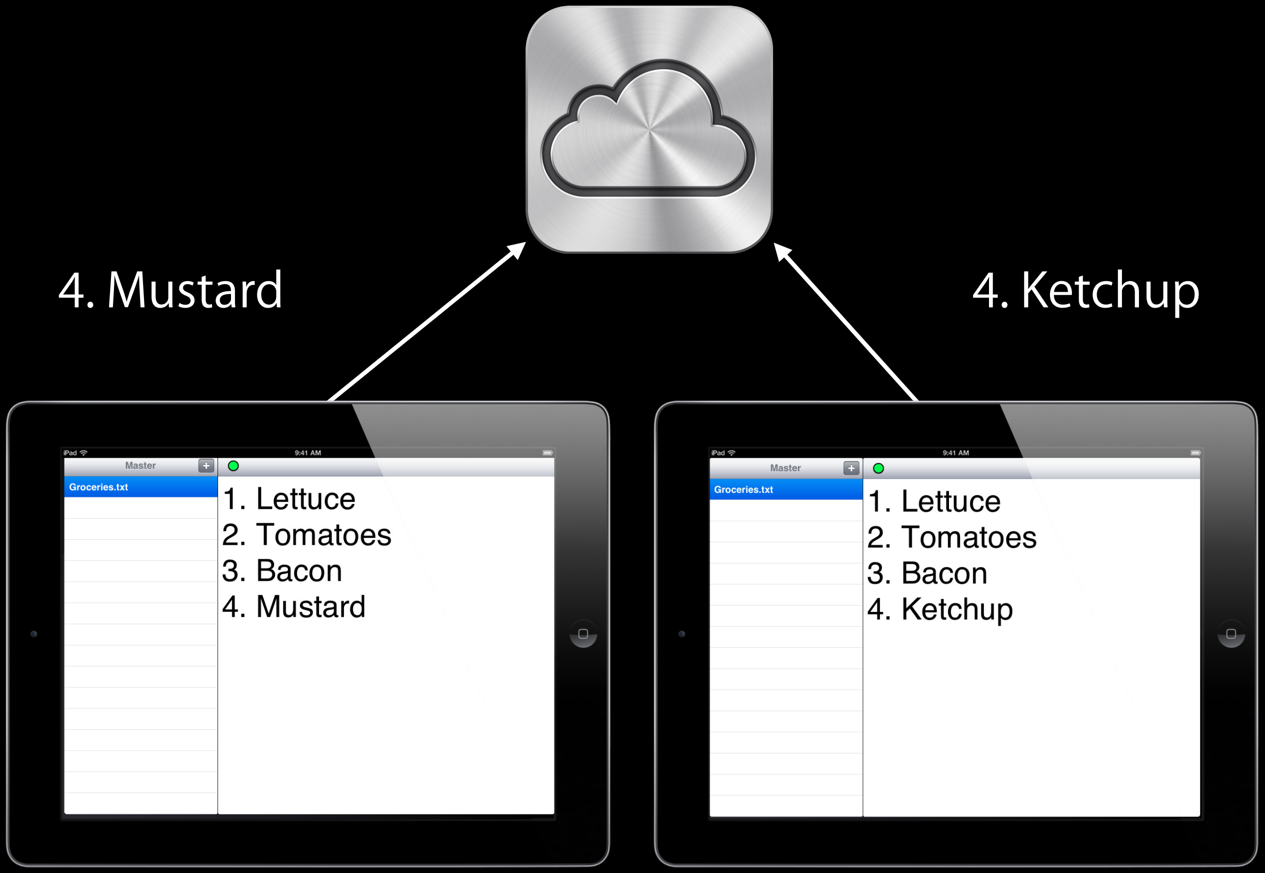
UIDocument

Responding to Saving Errors

- Register for `UIDocumentStateChangedNotification`
- `UIDocumentStateSavingError` will be set
- Show nonmodal UI

Conflict Resolution

Conflict Resolution



4. Mustard

4. Ketchup

1. Lettuce
2. Tomatoes
3. Bacon
4. Mustard

1. Lettuce
2. Tomatoes
3. Bacon
4. Ketchup

Resolving Conflicts

- What applications need to worry about conflicts?
    - All that use iCloud!
- Discover conflicts with `UIDocumentStateInConflict`
- Use NSFileVersion APIs
- Revert the document if necessary

Conflict-Resolution Strategies

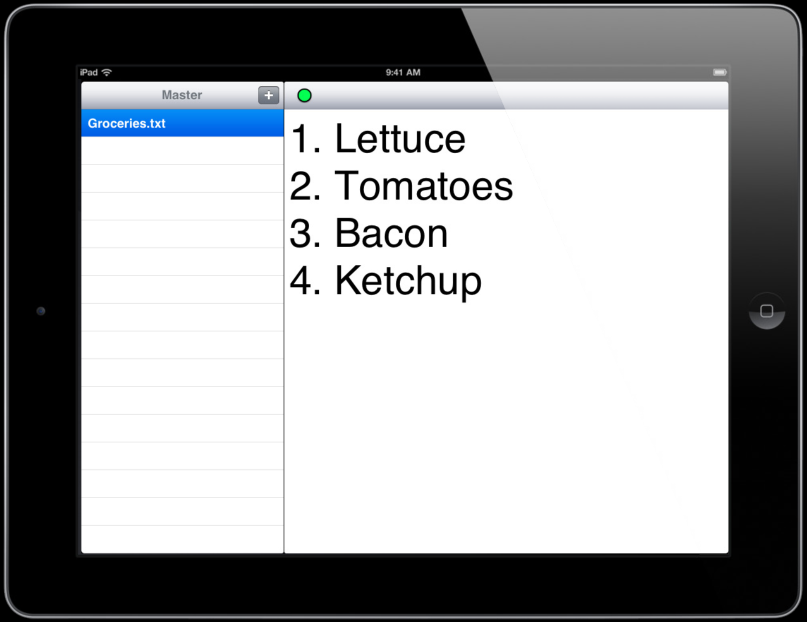
- Automatic merging
- Take the latest
- Prompt user
- Use UIManagedDocument

What’s New in CoreData on iOS

Mission
Tuesday 3:15–4:15PM

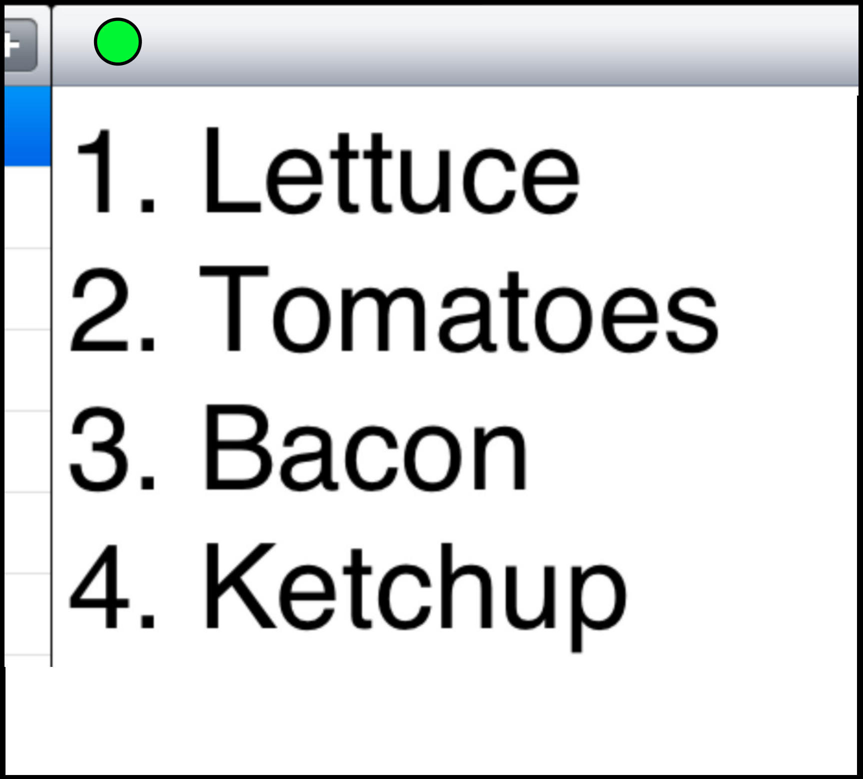
Extending CloudNotes

Adding State UI



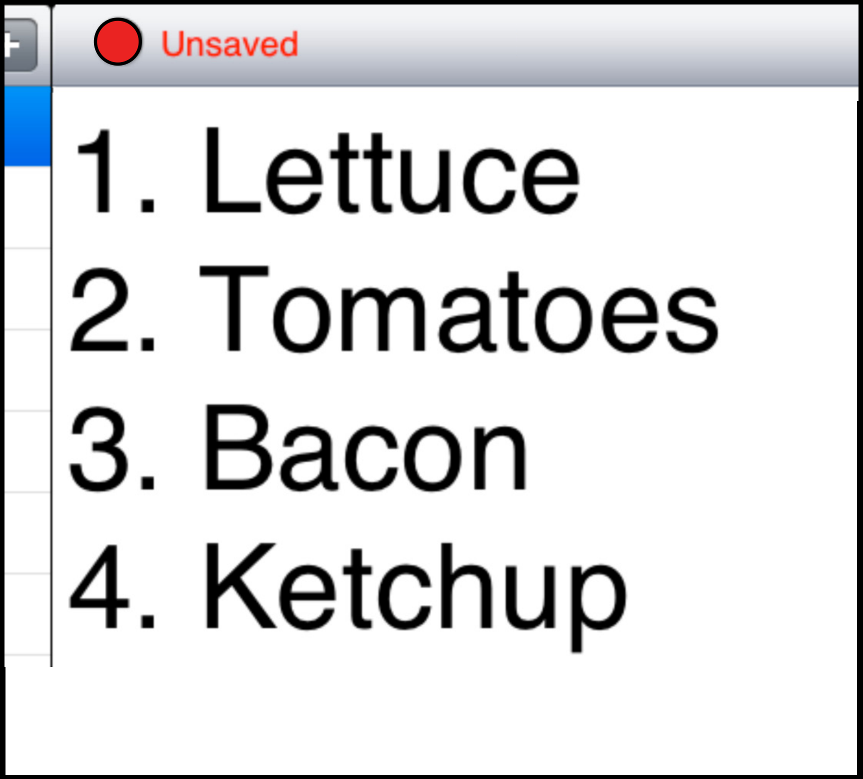
1. Lettuce
2. Tomatoes
3. Bacon
4. Ketchup

Normal State



1. Lettuce
2. Tomatoes
3. Bacon
4. Ketchup

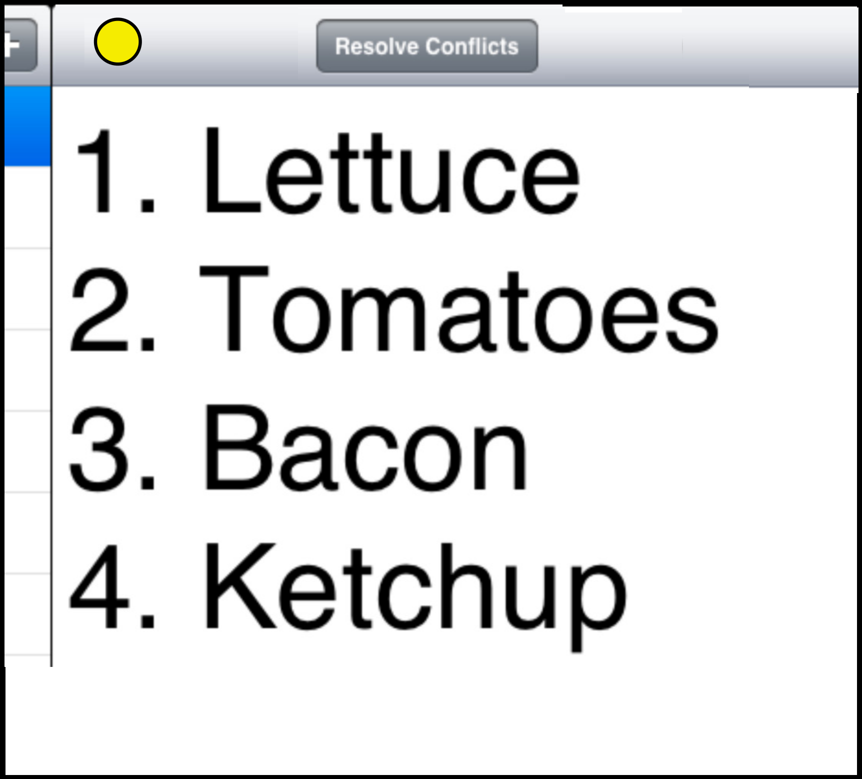
Saving Errors



● Unsaved

1. Lettuce
2. Tomatoes
3. Bacon
4. Ketchup

Conflicts

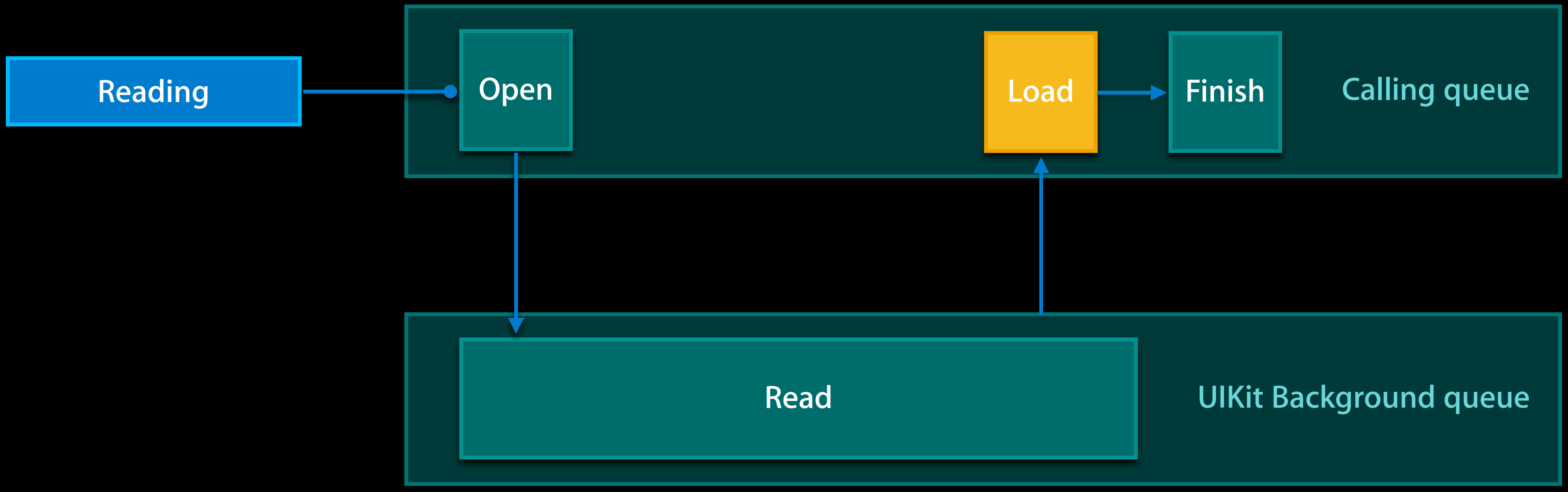


1. Lettuce
2. Tomatoes
3. Bacon
4. Ketchup

Advanced Concepts

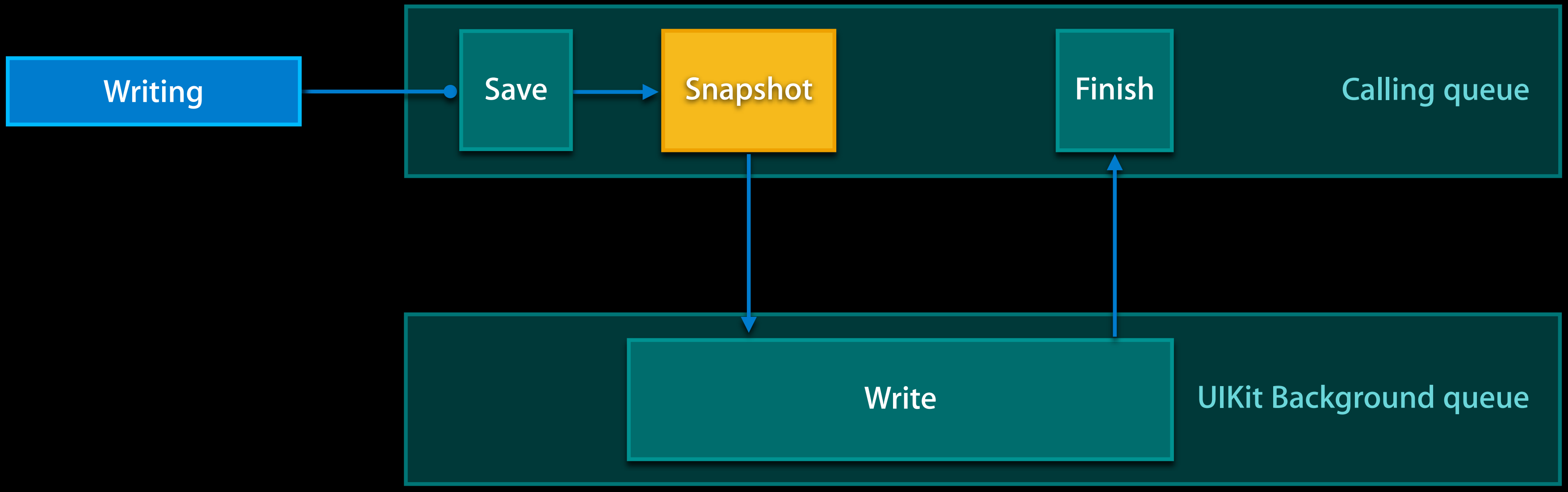
Reading

Revisiting the queue model



Writing

Revisiting the queue model



Calling into the Background Queue

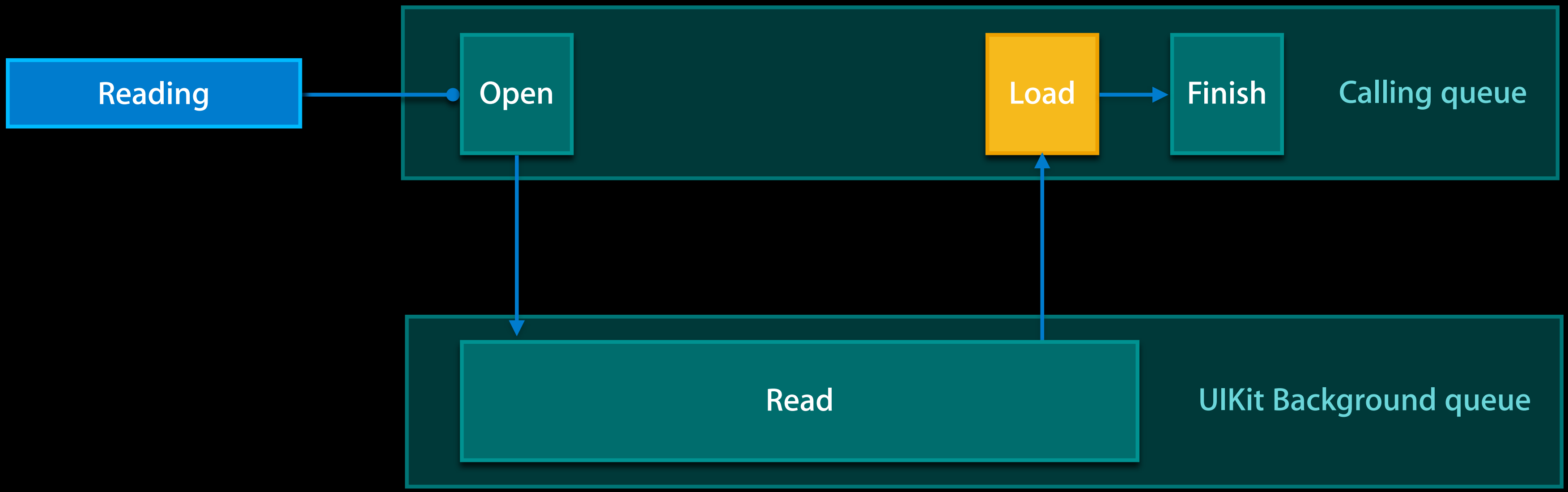
```
- (void)performAsynchronousFileAccessUsingBlock:(void (^)(void))block;
```

- NSFileCoordinator

Taking Advantage of File Coordination

Pacific Heights
Tuesday 4:30–5:30PM

Incremental Reading



Reading

Open

Load

Finish

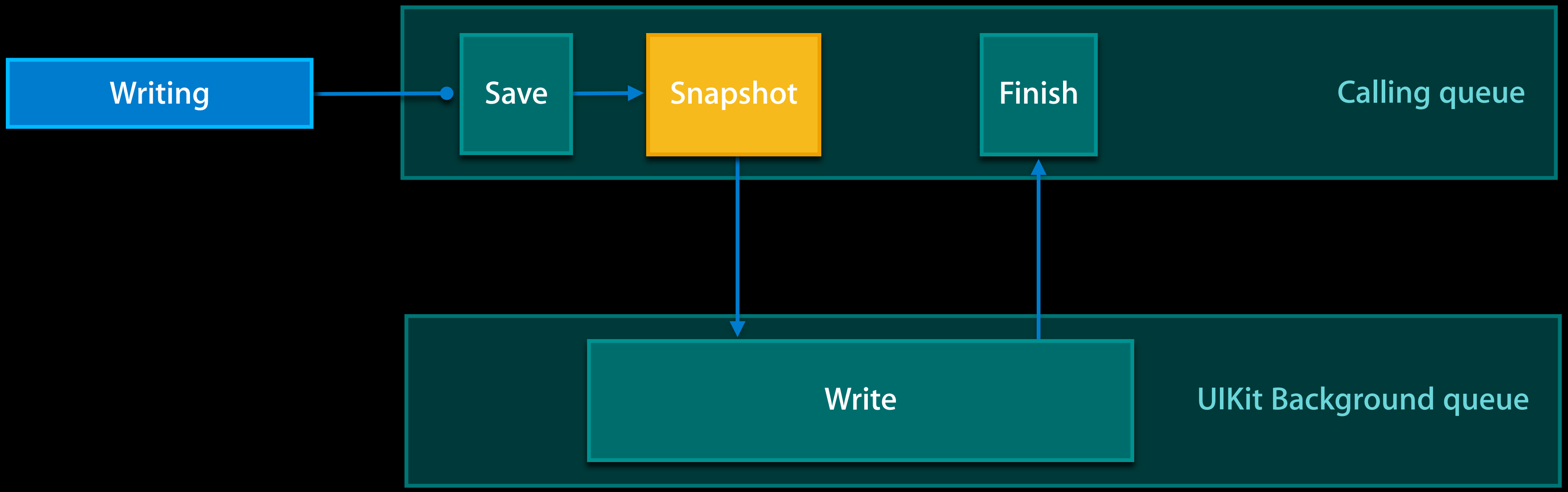
Calling queue

Read

UIKit Background queue

```
- (BOOL)readFromURL:(NSURL *)url
              error:(NSError **)error;
```

Safe Writing



Writing

Save

Snapshot

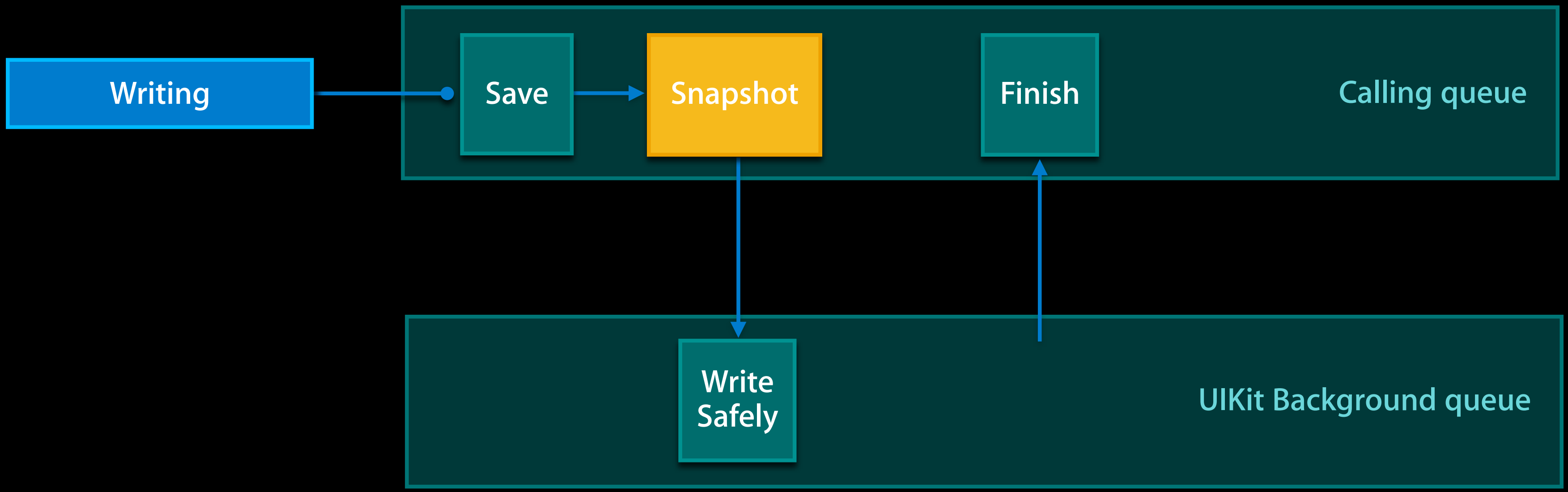
Finish

Calling queue

Write

UIKit Background queue

Safe Writing



Writing

Save

Snapshot

Finish

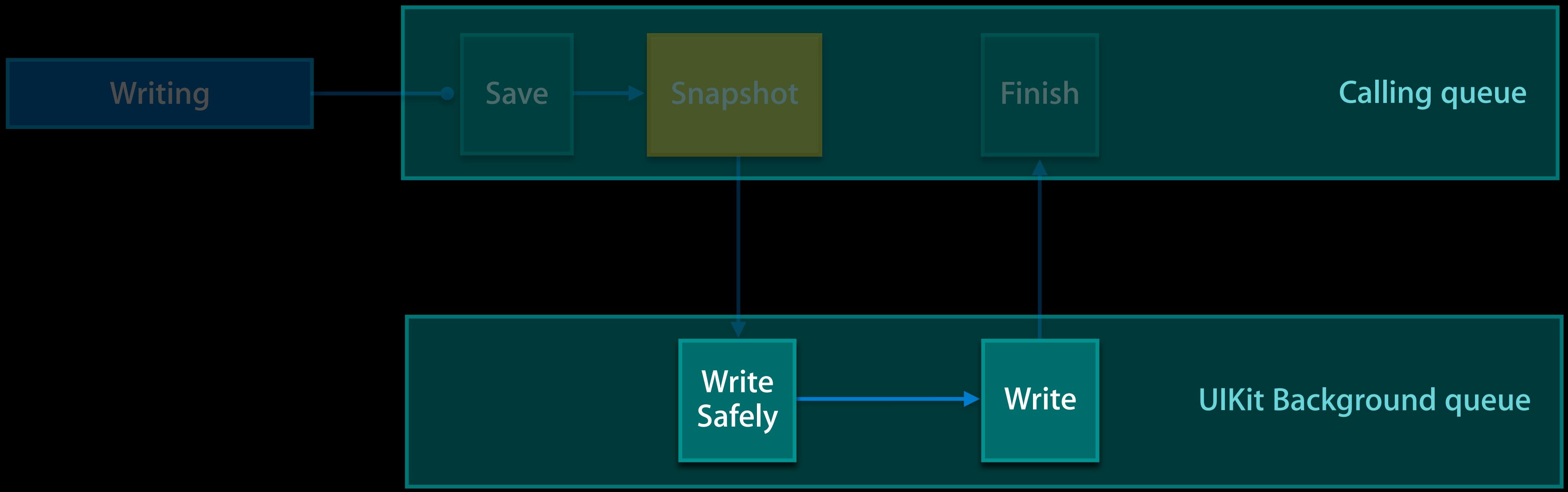
Calling queue

Write Safely

UIKit Background queue

```
- (BOOL)writeContents:(id)contents
        andAttributes:(NSDictionary *)additionalFileAttributes
          safelyToURL:(NSURL *)url
     forSaveOperation:(UIDocumentSaveOperation)saveOperation
                error:(NSError **)outError;
```

Writing a Custom Snapshot



```
- (BOOL)writeContents:(id)contents
        andAttributes:(NSDictionary *)additionalFileAttributes
          safelyToURL:(NSURL *)url
     forSaveOperation:(UIDocumentSaveOperation)saveOperation
                error:(NSError **)outError;
```

Changing the File Type

```
- (NSString *)savingFileType;

- (NSString *)fileNameExtensionForType:(NSString *)typeName
                        saveOperation:(UIDocumentSaveOperation)saveOperation;
```

Conclusion

Conclusion

- Be simple
- Register changes
- Use iCloud
- Resolve conflicts

Other Sessions and Lab

Taking Advantage of File Coordination	Pacific Heights Tuesday 4:30–5:30PM
What’s New in CoreData on iOS	Mission Tuesday 3:15–4:15PM
iCloud Storage Overview	Presidio Tuesday 11:30–12:30PM
UIDocument Lab	Application Frameworks Lab D Thursday 9:00–11:15AM

WWDC2011