# Taking Advantage of File Coordination

Session 109

**Mark Piccirelli**
Cocoa Frameworks Engineer

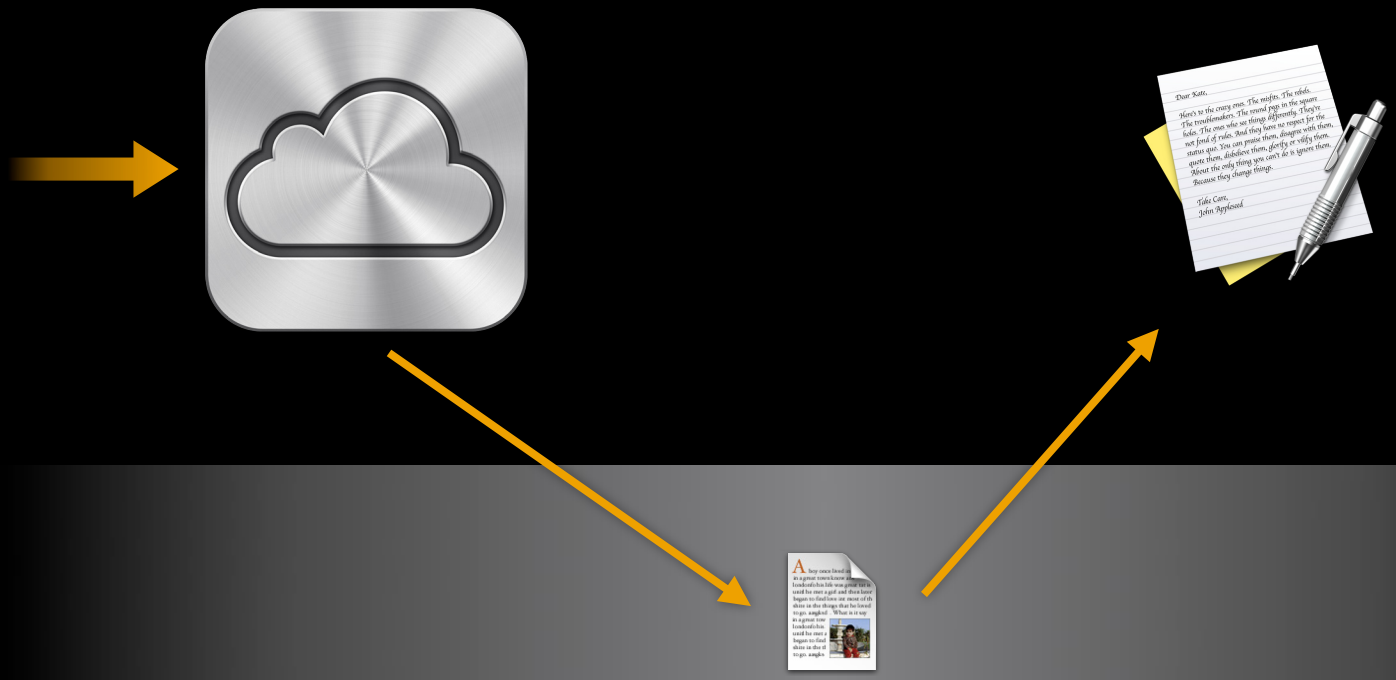
# What We'll Talk About

- What it is
- Why it's important
- How to use it

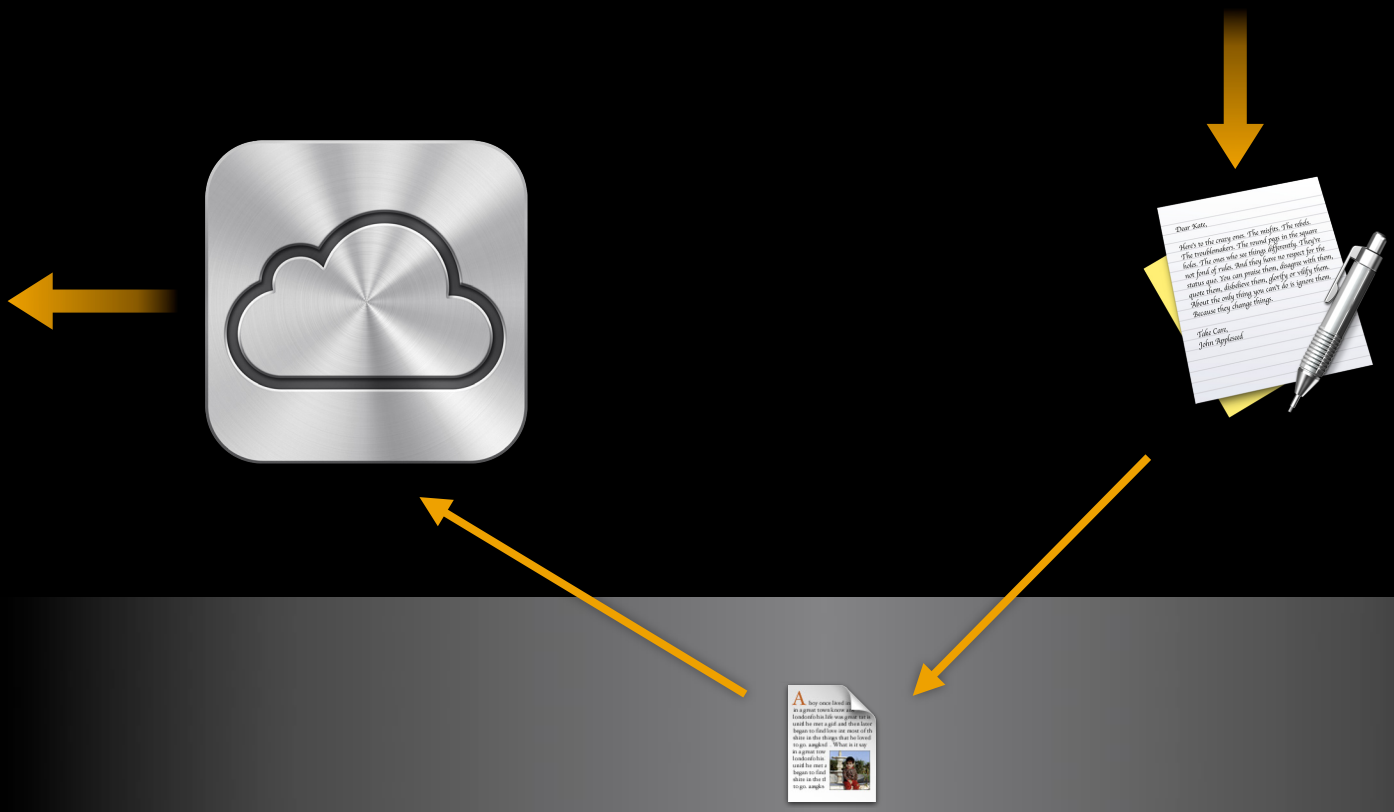
# iCloud

# iCloud Storage


# iCloud Storage



# iCloud Storage



# iCloud Storage

## Multiple processes accessing the same file

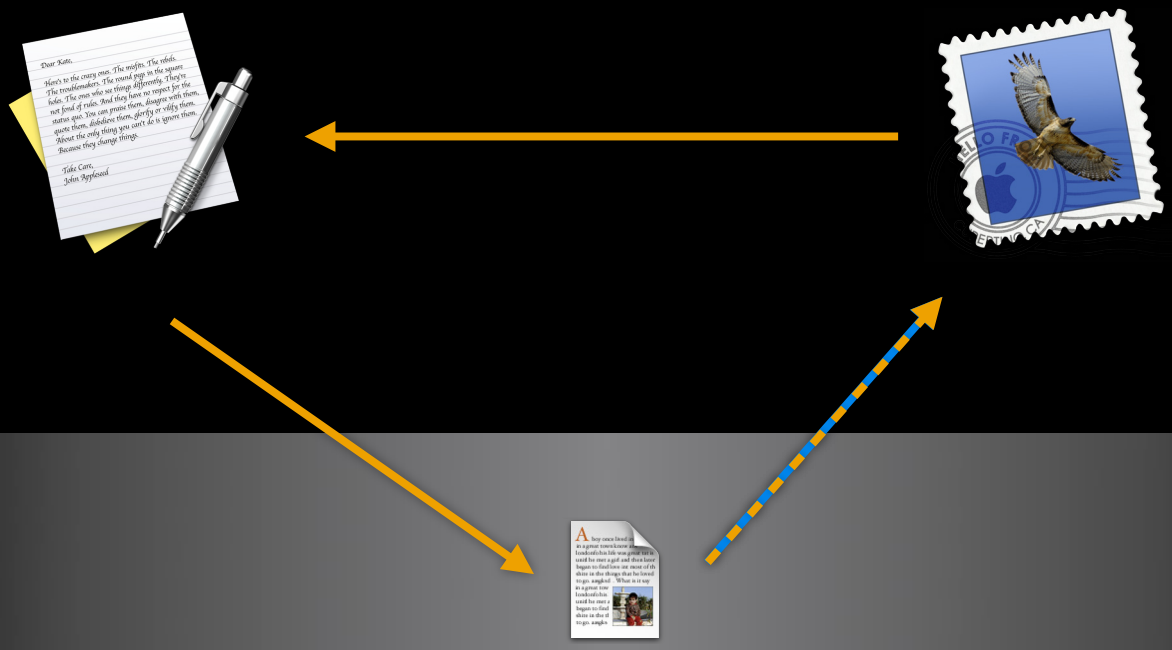
- One process writing while another is reading is bad
  - How does a process know when it's safe?
- iCloud changes files and then your app must read them
  - How does a process know when it must read?
- iCloud needs your files up-to-date to do conflict detection
  - How does a process know when it must write?

# Auto Save

# Auto Save

- A new document model for Mac OS X
- The user never has to save
- The window on the screen *is* the file

# Auto Save


Dear Katie,
Here's to the crazy ones. The misfits. The rebels.
The troublemakers. The round pegs in the square
holes. The ones who see things differently. They're
not fond of rules. And they have no respect for the
status quo. You can quote them, disagree with them,
glorify or vilify them.
About the only thing you can't do is ignore them.
Because they change things.
Take Care,
John Appleseed

# Auto Save

## Simple for the user, not us

- Processes reading out-of-date files would be bad
  - How does a process know when it must write?

# File Coordination

# File Coordination

- It's a locking mechanism
  - *File coordination* prevents one process from reading while another writes
- It's a notification mechanism
  - Your writing might cause some other process to read
  - Your reading might cause some other process to write
  - And vice versa

# File Coordination

- NSFileCoordinator
  - The class you use to do *coordinated* file access
- NSFilePresenter
  - The protocol you implement to hear about coordinated file access
- Mac OS 10.7 and iOS 5

# NSFileCoordinator

## Tell us what you're doing, we'll tell you when to do it

```
- (void)coordinateReadingItemAtURL:(NSURL *)url
                           options:(NSFileCoordinatorReadingOptions)options
                             error:(NSError **)outError
                        byAccessor:(void (^)(NSURL *newURL))reader;
- (void)coordinateWritingItemAtURL:(NSURL *)url
                           options:(NSFileCoordinatorWritingOptions)options
                             error:(NSError **)outError
                        byAccessor:(void (^)(NSURL *newURL))writer;
```

# NSFileCoordinator

## Tell us what you're doing, we'll tell you when to do it

```
- (void)coordinateReadingItemAtURL:(NSURL *)url
                           options:(NSFileCoordinatorReadingOptions)options
                             error:(NSError **)outError
                        byAccessor:(void (^)(NSURL *newURL))reader;
- (void)coordinateWritingItemAtURL:(NSURL *)url
                           options:(NSFileCoordinatorWritingOptions)options
                             error:(NSError **)outError
                        byAccessor:(void (^)(NSURL *newURL))writer;
```

- You pass in a block, we invoke the block

# NSFilePresenter

## Tell us who you are, we'll tell you what to do

```
- (void)presentedItemDidChange;
- (void)presentedItemDidMoveToURL:(NSURL *)newURL;
- (void)savePresentedItemChangesWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
- (void)accommodatePresentedItemDeletionWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
```

# NSFilePresenter

## Tell us who you are, we'll tell you what to do

```
- (void)presentedItemDidChange;
- (void)presentedItemDidMoveToURL:(NSURL *)newURL;
- (void)savePresentedItemChangesWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
- (void)accommodatePresentedItemDeletionWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
```

# NSFilePresenter

## Tell us who you are, we'll tell you what to do

```
- (void)presentedItemDidChange;
- (void)presentedItemDidMoveToURL:(NSURL *)newURL;
- (void)savePresentedItemChangesWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
- (void)accommodatePresentedItemDeletionWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
```

# NSFilePresenter

## Tell us who you are, we'll tell you what to do

```
- (void)presentedItemDidChange;
- (void)presentedItemDidMoveToURL:(NSURL *)newURL;
- (void)savePresentedItemChangesWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
- (void)accommodatePresentedItemDeletionWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
```

# NSFilePresenter

## Tell us who you are, we'll tell you what to do

```
- (void)presentedItemDidChange;
- (void)presentedItemDidMoveToURL:(NSURL *)newURL;
- (void)savePresentedItemChangesWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
- (void)accommodatePresentedItemDeletionWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
```

# NSFilePresenter

## Tell us who you are, we'll tell you what to do

```
- (void)presentedItemDidChange;
- (void)presentedItemDidMoveToURL:(NSURL *)newURL;
- (void)savePresentedItemChangesWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
- (void)accommodatePresentedItemDeletionWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
```

- Register an NSFilePresenter when you're presenting a file to the user
- Hear about other processes' coordinated file access

# NSFilePresenter

## Tell us who you are, we'll tell you what to do

```
- (void)presentedItemDidChange;
- (void)presentedItemDidMoveToURL:(NSURL *)newURL;
- (void)savePresentedItemChangesWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
- (void)accommodatePresentedItemDeletionWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
```

- Register an NSFilePresenter when you're presenting a file to the user
- Hear about other processes' coordinated file access
- *Affect* other processes' coordinated file access

# When to Use NSFileCoordinator

- Files that the user thinks of as files
  - On Mac OS X
  - Coordinate with Finder
  - Coordinate with other apps
- iCloud documents
  - On both Mac OS X and iOS
  - Coordinate with iCloud daemons

# When to Use NSFileCoordinator

- Your goal is to play well with other processes
- You don’t know what other processes might do

# When to Use NSFileCoordinator

- Reading
- Creating
  - Exporting
- Updating
- Copying
- Moving and renaming
  - Directories, too
- NSDocument and UIDocument do this for you

# When Not to Use NSFileCoordinator

- Temporary files
- Cache files
- Anything private enough

# When to Be an NSFilePresenter

- Present a file to the user for viewing or editing
  - File packages, too
- Reload when other processes change it
- Close when other processes will delete it
- *Relinquish* to other processes
- NSDocument and UIDocument do this for you

# NSFileCoordinator

# The Locking Rules

- Readers don't block readers
- Readers block writers
- Writers block readers
- Writers block writers
- File packages are treated atomically
- NSFileCoordinators don't block themselves

# Creating an NSFileCoordinator

```
- (id)initWithFilePresenter:(id<NSFilePresenter>)filePresenterOrNil;
```

- One per operation
  - NSDocument opening or saving
  - Finder copying

# Creating an NSFileCoordinator

```
- (id)initWithFilePresenter:(id<NSFilePresenter>)filePresenterOrNil;
```

- You can pass an NSFilePresenter
  - For an operation done on behalf of it
  - Filters out messages about your own file access
  - Helps us break deadlocks

# Simple Reading and Writing

```
- (void)coordinateReadingItemAtURL:(NSURL *)url
                           options:(NSFileCoordinatorReadingOptions)options
                             error:(NSError **)outError
                        byAccessor:(void (^)(NSURL *newURL))reader;

- (void)coordinateWritingItemAtURL:(NSURL *)url
                           options:(NSFileCoordinatorWritingOptions)options
                             error:(NSError **)outError
                        byAccessor:(void (^)(NSURL *newURL))writer;
```

# Simple Reading and Writing

```
- (void)coordinateReadingItemAtURL:(NSURL *)url
                           options:(NSFileCoordinatorReadingOptions)options
                             error:(NSError **)outError
                        byAccessor:(void (^)(NSURL *newURL))reader;

- (void)coordinateWritingItemAtURL:(NSURL *)url
                           options:(NSFileCoordinatorWritingOptions)options
                             error:(NSError **)outError
                        byAccessor:(void (^)(NSURL *newURL))writer;
```

- Pass in a block, NSFileCoordinator invokes the block

# Simple Reading and Writing

```
- (void)coordinateReadingItemAtURL:(NSURL *)url
                           options:(NSFileCoordinatorReadingOptions)options
                             error:(NSError **)outError
                        byAccessor:(void (^)(NSURL *newURL))reader;

- (void)coordinateWritingItemAtURL:(NSURL *)url
                           options:(NSFileCoordinatorWritingOptions)options
                             error:(NSError **)outError
                        byAccessor:(void (^)(NSURL *newURL))writer;
```

- You get passed a new URL
  - Moving and renaming
  - NSURL cache invalidation

# Simple Reading and Writing

```
- (void)coordinateReadingItemAtURL:(NSURL *)url
                           options:(NSFileCoordinatorReadingOptions)options
                             error:(NSError **)outError
                        byAccessor:(void (^)(NSURL *newURL))reader;

- (void)coordinateWritingItemAtURL:(NSURL *)url
                           options:(NSFileCoordinatorWritingOptions)options
                             error:(NSError **)outError
                        byAccessor:(void (^)(NSURL *newURL))writer;
```

- Synchronous

# Simple Reading

## Example

```
- (NSData *)readFromURL:(NSURL *)url error:(NSError **)outError {
    __block NSData *data = nil;
    NSFileCoordinator *fileCoordinator =
        [[NSFileCoordinator alloc] initWithFilePresenter:self];
    [fileCoordinator coordinateReadingItemAtURL:url options:0
        error:outError byAccessor:^(NSURL *newURL) {
        data = [[NSData alloc] initWithContentsOfURL:newURL options:0
            error:outError];
    }];
    [fileCoordinator release];
    return [data autorelease];
}
```

# Simple Reading

## Example

```
- (NSData *)readFromURL:(NSURL *)url error:(NSError **)outError {
    __block NSData *data = nil;
    NSFileCoordinator *fileCoordinator =
        [[NSFileCoordinator alloc] initWithFilePresenter:self];
    [fileCoordinator coordinateReadingItemAtURL:url options:0
        error:outError byAccessor:^(NSURL *newURL) {
        data = [[NSData alloc] initWithContentsOfURL:newURL options:0
            error:outError];
    }];
    [fileCoordinator release];
    return [data autorelease];
}
```

# Simple Reading

## Example

```
- (NSData *)readFromURL:(NSURL *)url error:(NSError **)outError {
    __block NSData *data = nil;
    NSFileCoordinator *fileCoordinator =
        [[NSFileCoordinator alloc] initWithFilePresenter:self];
    [fileCoordinator coordinateReadingItemAtURL:url options:0
        error:outError byAccessor:^(NSURL *newURL) {
        data = [[NSData alloc] initWithContentsOfURL:newURL options:0
            error:outError];
    }];
    [fileCoordinator release];
    return [data autorelease];
}
```

# Simple Reading

## Example

```
- (NSData *)readFromURL:(NSURL *)url error:(NSError **)outError {
    __block NSData *data = nil;
    NSFileCoordinator *fileCoordinator =
        [[NSFileCoordinator alloc] initWithFilePresenter:self];
    [fileCoordinator coordinateReadingItemAtURL:url options:0
        error:outError byAccessor:^(NSURL *newURL) {
        data = [[NSData alloc] initWithContentsOfURL:newURL options:0
            error:outError];
    }];
    [fileCoordinator release];
    return [data autorelease];
}
```

# Simple Reading

## Example

```
- (NSData *)readFromURL:(NSURL *)url error:(NSError **)outError {
    __block NSData *data = nil;
    NSFileCoordinator *fileCoordinator =
        [[NSFileCoordinator alloc] initWithFilePresenter:self];
    [fileCoordinator coordinateReadingItemAtURL:url options:0
        error:outError byAccessor:^(NSURL *newURL) {
        data = [[NSData alloc] initWithContentsOfURL:newURL options:0
            error:outError];
    }];
    [fileCoordinator release];
    return [data autorelease];
}
```

# Simple Reading

## Example

```
- (NSData *)readFromURL:(NSURL *)url error:(NSError **)outError {
    __block NSData *data = nil;
    NSFileCoordinator *fileCoordinator =
        [[NSFileCoordinator alloc] initWithFilePresenter:self];
    [fileCoordinator coordinateReadingItemAtURL:url options:0
        error:outError byAccessor:^(NSURL *newURL) {
        data = [[NSData alloc] initWithContentsOfURL:newURL options:0
            error:outError];
    }];
    [fileCoordinator release];
    return [data autorelease];
}
```

# Simple Reading

## What might have just happened?

- Waiting for writers
  - Other users of NSFileCoordinator
  - File might be moved or deleted
- NSFilePresenter messaging
  - File might be updated
- Errors
  - An NSFilePresenter might have had trouble updating

# More Than One File at a Time

## Reading and writing

```
- (void)coordinateReadingItemAtURL:(NSURL *)readingURL
                           options:(NSFileCoordinatorReadingOptions)readingOptions
                  writingItemAtURL:(NSURL *)writingURL
                           options:(NSFileCoordinatorWritingOptions)writingOptions
                             error:(NSError **)outError
                        byAccessor:
                (void (^)(NSURL *newReadingURL, NSURL *newWritingURL))readerWriter;
```

# More Than One File at a Time

## Reading and writing

```
- (void)coordinateReadingItemAtURL:(NSURL *)readingURL
                           options:(NSFileCoordinatorReadingOptions)readingOptions
                  writingItemAtURL:(NSURL *)writingURL
                           options:(NSFileCoordinatorWritingOptions)writingOptions
                             error:(NSError **)outError
                        byAccessor:
                (void (^)(NSURL *newReadingURL, NSURL *newWritingURL))readerWriter;
```

- Use it to avoid deadlocks
- Copying

# More Than One File at a Time

## Writing, and still more writing

```
- (void)coordinateWritingItemAtURL:(NSURL *)url1
                           options:(NSFileCoordinatorWritingOptions)options1
                  writingItemAtURL:(NSURL *)url2
                           options:(NSFileCoordinatorWritingOptions)options2
                             error:(NSError **)outError
                        byAccessor:(void (^)(NSURL *newURL1, NSURL *newURL2))writer;
```

# More Than One File at a Time

## Writing, and still more writing

```
- (void)coordinateWritingItemAtURL:(NSURL *)url1
                           options:(NSFileCoordinatorWritingOptions)options1
                  writingItemAtURL:(NSURL *)url2
                           options:(NSFileCoordinatorWritingOptions)options2
                             error:(NSError **)outError
                        byAccessor:(void (^)(NSURL *newURL1, NSURL *newURL2))writer;
```

- Moving

# NSFileCoordinator Options

## Reading

```
NSFileCoordinatorReadingResolvesSymbolicLink
```

- Difficult to get right on your own
- Only applies to the item itself

# NSFileCoordinator Options

## Reading

`NSFileCoordinatorReadingWithoutChanges`

- By default your reader gives NSFilePresenters a chance to write
- Sometimes you are not interested
- iCloud uses it

# NSFileCoordinator Options

## Writing

- No options means you are *updating*
- Even when it's more complicated than opening, writing, then closing
- Safe saving
- We deal in what the user thinks of as files
  - Not always actual files

# NSFileCoordinator Options

## Writing

`NSFileCoordinatorWritingForReplacing`

- Creating a brand new item
- Moving an existing item
- Checking for a file to replace does not work
  - Race conditions
  - Don’t bother
- Tells NSFilePresenters ahead of time
- NSDocument Save As and Save To operations

# NSFileCoordinator Options

## Writing

`NSFileCoordinatorWritingForDeleting`

- Special handling of directories
  - Waits for access of contained items
  - Makes access of contained items wait
- Tells NSFilePresenters ahead of time
- (WritingForReplacing does the same things)

# NSFileCoordinator Options

## Writing

`NSFileCoordinatorWritingForMoving`

- Same waiting rules as WritingForReplacing and WritingForDeleting
  - Operations on giant directories can be one coordinated write
- No additional NSFilePresenter messaging

# NSFileCoordinator Options

## Writing, and then some

`NSFileCoordinatorWritingForMerging`

- Tells any NSFilePresenter to write before you write!
  - Then you read before you write
- iCloud uses it
  - Conflict detection

# Renaming While Saving

```
- (void)itemAtURL:(NSURL *)oldURL didMoveToURL:(NSURL *)newURL;
```

- We deal in what the user thinks of as files
- Sometimes you have to tell us what just happened
- Invoke it during coordinated writing
- NSDocument uses it
  - TextEdit's switching between .rtf and .rtfd

# NSFilePresenter

# NSFilePresenter

- Your app says what it's showing to the user
- Participates in other processes' coordinated file access
- Gets notified of uncoordinated file access, too

# Registering An NSFilePresenter

- A protocol, not a class
  - NSDocument and UIDocument implement it
- NSFileCoordinator methods

```
+ (void)addFilePresenter:(id<NSFilePresenter>)filePresenter;
+ (void)removeFilePresenter:(id<NSFilePresenter>)filePresenter;
```

- You always have to invoke the removing method
  - -dealloc is often too late
  - NSDocument does it in -close

# Saying What's Presented

```
@property (readonly) NSURL *presentedItemURL;
```

- You don't tell file coordination, file coordination asks you
- On any thread, at any time
  - Atomic property
- You don't tell file coordination when it's moved, file coordination tells you
  - Might mean you have to use file coordination

# Getting Messages on a Queue

```
@property (readonly) NSOperationQueue *presentedItemOperationQueue;
```

- The queue for other messages
- [NSOperationQueue mainQueue] is often OK
  - Watch out for deadlocks

# Relinquishing to Other Readers and Writers

```
- (void)relinquishPresentedItemToReader:
                                (void (^)(void (^reacquirer)(void)))reader;
- (void)relinquishPresentedItemToWriter:
                                (void (^)(void (^reacquirer)(void)))writer;
```

- Someone wants to read or write

# Relinquishing to Other Readers and Writers

```
- (void)relinquishPresentedItemToReader:
                                (void (^)(void (^reacquirer)(void)))reader;
- (void)relinquishPresentedItemToWriter:
                                (void (^)(void (^reacquirer)(void)))writer;
```

- Someone wants to read or write
- You *must* invoke the passed-in block
  - Quickly!

# Relinquishing to Other Readers and Writers

```
- (void)relinquishPresentedItemToReader:
                                (void (^)(void (^reacquirer)(void)))reader;
- (void)relinquishPresentedItemToWriter:
                                (void (^)(void (^reacquirer)(void)))writer;
```

- Someone wants to read or write
- You *must* invoke the passed-in block
  - Quickly!
- You can pass a block
  - It gets invoked when the reading or writing is done

# Relinquishing to a Writer

## Example

```
- (void)relinquishPresentedItemToWriter:
                            (void (^)(void (^reacquirer)(void)))writer {
    [self waitForWriter:^(void (^stopWaiting)(void)) {
        writer(^(void) {
            stopWaiting();
        });
    }];
}
```

# Relinquishing to a Writer

## Example

```
- (void)relinquishPresentedItemToWriter:
                            (void (^)(void (^reacquirer)(void)))writer {
    [self waitForWriter:^(void (^stopWaiting)(void)) {
        writer(^(void) {
            stopWaiting();
        });
    }];
}
```

- Stop looking at the file
- Stop looking at ivars that relate to the file
- Like -[NSDocument performAsynchronousFileAccessUsingBlock:]

# Relinquishing to a Writer

## Example

```
- (void)relinquishPresentedItemToWriter:
                            (void (^)(void (^reacquirer)(void)))writer {
    [self waitForWriter:^(void (^stopWaiting)(void)) {
        writer(^(void) {
            stopWaiting();
        });
    }];
}
```

# Relinquishing to a Writer

## Example

```
- (void)relinquishPresentedItemToWriter:
                            (void (^)(void (^reacquirer)(void)))writer {
    [self waitForWriter:^(void (^stopWaiting)(void)) {
        writer(^(void) {
            stopWaiting();
        });
    }];
}
```

- Let the other process go

# Relinquishing to a Writer

## Example

```
- (void)relinquishPresentedItemToWriter:
                            (void (^)(void (^reacquirer)(void)))writer {
    [self waitForWriter:^(void (^stopWaiting)(void)) {
        writer(^(void) {
            stopWaiting();
        });
    }];
}
```

- The other process is done

# Relinquishing to a Writer

## Example

```
- (void)relinquishPresentedItemToWriter:
                            (void (^)(void (^reacquirer)(void)))writer {
    [self waitForWriter:^(void (^stopWaiting)(void)) {
        writer(^(void) {
            stopWaiting();
        });
    }];
}
```

# Hearing About Changes

```
- (void)presentedItemDidChange;
- (void)presentedItemDidMoveToURL:(NSURL *)newURL;
```

- A change happened
- Catch up
- Usually while relinquished to a writer
- Not always
  - Not everyone uses file coordination
  - Do the best you can

# Saving on Demand

```
- (void)savePresentedItemChangesWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
```

- Something wants to read the most up-to-date contents
  - Autosaving?
- Save the user's changes
- Quickly!
  - This is no time to present UI
- Always invoke the completion handler
- What NSFileCoordinatorReadingWithoutChanges prevents

# Getting Deleted

```
- (void)accommodatePresentedItemDeletionWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
```

- Something wants to delete the file
- A good time to close a document
- Not a good time to present UI
- Don’t forget that completion handler
- What NSFileCoordinatorWritingForDeleting and NSFileCoordinatorWritingForReplacing cause

# Versions

## Auto Save and iCloud

```
- (void)presentedItemDidGainVersion:(NSFileVersion *)version;
- (void)presentedItemDidLoseVersion:(NSFileVersion *)version;
- (void)presentedItemDidResolveConflictVersion:(NSFileVersion *)version;
```

- Something added, removed, or resolved a version
- iCloud does this when it detects conflicts
- NSDocument takes care of this for you

# Supporting iCloud

## Not all apps are document-based

# What Is a Shoebox App?

- Many apps do not deal in documents
- Just show the user their data, not files
  - iPhoto
  - iTunes
- Like a *shoebox* of pictures or tapes

# NSFilePresenter in Shoebox Apps

- Presenting a directory full of files, not just one file
- iCloud *weak packages*
  - .weakpkg file name extension
  - iCloud reads multiple files per coordinated read
  - iCloud changes multiple files per coordinated write
  - Your consistency rules are our consistency rules
- File coordination treats them like regular file packages

# Presenting a Package

## Many of the regular methods apply

```
- (void)relinquishPresentedItemToReader:
                        (void (^)(void (^reacquirer)(void)))reader;
- (void)relinquishPresentedItemToWriter:
                        (void (^)(void (^reacquirer)(void)))writer;
- (void)savePresentedItemChangesWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
```

# Presenting a Package

## Many of the regular methods apply

```
- (void)relinquishPresentedItemToReader:
                        (void (^)(void (^reacquirer)(void)))reader;
- (void)relinquishPresentedItemToWriter:
                        (void (^)(void (^reacquirer)(void)))writer;
- (void)savePresentedItemChangesWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
```

# Presenting a Package

## Many of the regular methods apply

```
- (void)relinquishPresentedItemToReader:
                        (void (^)(void (^reacquirer)(void)))reader;
- (void)relinquishPresentedItemToWriter:
                        (void (^)(void (^reacquirer)(void)))writer;
- (void)savePresentedItemChangesWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
```

# Presenting a Package

## Many of the regular methods apply

```
- (void)relinquishPresentedItemToReader:
                        (void (^)(void (^reacquirer)(void)))reader;
- (void)relinquishPresentedItemToWriter:
                        (void (^)(void (^reacquirer)(void)))writer;
- (void)savePresentedItemChangesWithCompletionHandler:
                        (void (^)(NSError *errorOrNil))completionHandler;
```

# Presenting Package Subitems

```
- (void)presentedSubitemDidAppearAtURL:(NSURL *)url;
- (void)presentedSubitemDidChangeAtURL:(NSURL *)url;
- (void)presentedSubitemAtURL:(NSURL *)oldURL didMoveToURL:(NSURL *)newURL;
- (void)accommodatePresentedSubitemDeletionAtURL:(NSURL *)url
    completionHandler:(void (^)(NSError *errorOrNil))completionHandler;
```

# Presenting Package Subitems

```
- (void)presentedSubitemDidAppearAtURL:(NSURL *)url;
- (void)presentedSubitemDidChangeAtURL:(NSURL *)url;
- (void)presentedSubitemAtURL:(NSURL *)oldURL didMoveToURL:(NSURL *)newURL;
- (void)accommodatePresentedSubitemDeletionAtURL:(NSURL *)url
    completionHandler:(void (^)(NSError *errorOrNil))completionHandler;
```

- Messages about individual files
- Coming soon

# Versions of Package Subitems

```
- (void)presentedSubitemAtURL:(NSURL *)url
                  didGainVersion:(NSFileVersion *)version;
- (void)presentedSubitemAtURL:(NSURL *)url
                  didLoseVersion:(NSFileVersion *)version;
- (void)presentedSubitemAtURL:(NSURL *)url
    didResolveConflictVersion:(NSFileVersion *)version;
```

# Versions of Package Subitems

```
- (void)presentedSubitemAtURL:(NSURL *)url
                didGainVersion:(NSFileVersion *)version;
- (void)presentedSubitemAtURL:(NSURL *)url
                didLoseVersion:(NSFileVersion *)version;
- (void)presentedSubitemAtURL:(NSURL *)url
    didResolveConflictVersion:(NSFileVersion *)version;
```

- Individual files have their own versions
  - Only in iCloud weak packages

# Summary

- New features
  - iCloud
  - Auto Save
- New mechanisms
  - Locking
  - Notification
- Use NSFileCoordinator when changing files
- Use NSFilePresenter when showing files to the user
  - Even when the user doesn't know they are files

# More Information

**Bill Dudney**
Application Frameworks Evangelist
dudney@apple.com

**Documentation**
Mac OS X Dev Center
http://developer.apple.com/devcenter/mac

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

# More Information

## Header files

<Foundation/NSFileCoordinator.h>

<Foundation/NSFilePresenter.h>

- There are some comments there

# Related Sessions

| | |
|---|---|
| What's New in Cocoa | Presidio<br>Tuesday 10:15AM |
| iCloud Storage Overview | Presidio<br>Tuesday 11:30AM |
| Auto Save and Versions in Mac OS X Lion | Pacific Heights<br>Tuesday 3:15PM |
| Storing Documents in iCloud Using iOS 5 | Presidio<br>Wednesday 3:15 PM |
| What's New in Core Data on Mac OS X | Nob Hill<br>Thursday 11:30AM |

# Labs

| | |
|---|---|
| Cocoa Lab | App Frameworks Lab A<br>Tuesday 2:00-6:00PM |
| Cocoa and Automatic Layout Lab | App Frameworks Lab A<br>Wednesday 9:00-11:15AM |
| Cocoa, Full Screen, and Aqua Lab | App Frameworks Lab A<br>Wednesday 2:00-6:00PM |
| Cocoa, Auto Save, File Coordination, and Resume Lab | App Frameworks Lab A<br>Thursday 2:00-4:15PM |


WWDC2011