Advanced iCloud Document Storage

Session 237

Mark Piccirelli
Cocoa Frameworks Engineer

What We Will Talk About

- iCloud Document Storage

What We Will Talk About

- iCloud Document Storage
  - Not key-value storage

What We Will Talk About

- iCloud Document Storage
  - Not key-value storage
- Not documents

What We Will Talk About

- iCloud Document Storage
  - Not key-value storage
- Not documents
- Shoebox apps

What Is a Shoebox App?

What Is a Shoebox App?

- Many apps do not deal in documents

What Is a Shoebox App?

- Many apps do not deal in documents
- Just show the user their data, not files

What Is a Shoebox App?

- Many apps do not deal in documents
- Just show the user their data, not files
  - iPhoto

What Is a Shoebox App?

- Many apps do not deal in documents
- Just show the user their data, not files
  - iPhoto
  - iTunes

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

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
- NSDocument or UIDocument are not appropriate

Today's Example

A picture-viewing app

What We Will Talk About

What We Will Talk About

- Using Foundation APIs directly
  - NSFileCoordinator
  - NSFilePresenter
  - NSFileVersion
  - NSFileManager

What We Will Talk About

- Things that happen in iCloud apps

What We Will Talk About

- Things that happen in iCloud apps
  - Changes to your app's files

What We Will Talk About

- Things that happen in iCloud apps
  - Changes to your app's files
  - Conflicts

What We Will Talk About

- Things that happen in iCloud apps
  - Changes to your app's files
  - Conflicts
- Tips and Advice

File Coordination

iCloud Document Storage

Multiple processes accessing the same file

iCloud Document Storage

Multiple processes accessing the same file

- One process writing while another is reading is bad

iCloud Document Storage

Multiple processes accessing the same file

- One process writing while another is reading is bad
  - How does a process know when it is safe?

iCloud Document Storage

Multiple processes accessing the same file

- One process writing while another is reading is bad
  - How does a process know when it is safe?
- iCloud changes files and then your app must read them

iCloud Document Storage

Multiple processes accessing the same file

- One process writing while another is reading is bad
  - How does a process know when it is safe?
- iCloud changes files and then your app must read them
  - How does a process know when it must read?

iCloud Document Storage

Multiple processes accessing the same file

- One process writing while another is reading is bad
  - How does a process know when it is 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

iCloud Document Storage

Multiple processes accessing the same file

- One process writing while another is reading is bad
  - How does a process know when it is 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?

File Coordination

File Coordination

- It is a locking mechanism

File Coordination

- It is a locking mechanism
  - Prevents your app from reading while iCloud writes

File Coordination

- It is a locking mechanism
  - Prevents your app from reading while iCloud writes
  - And vice versa

File Coordination

- It is a locking mechanism
  - Prevents your app from reading while iCloud writes
  - And vice versa
- It is a notification mechanism

File Coordination

- It is a locking mechanism
  - Prevents your app from reading while iCloud writes
  - And vice versa
- It is a notification mechanism
  - Tells your app when iCloud changes have happened

File Coordination

- It is a locking mechanism
  - Prevents your app from reading while iCloud writes
  - And vice versa
- It is a notification mechanism
  - Tells your app when iCloud changes have happened
- It is a *triggering* mechanism

File Coordination

- It is a locking mechanism
  - Prevents your app from reading while iCloud writes
  - And vice versa
- It is a notification mechanism
  - Tells your app when iCloud changes have happened
- It is a *triggering* mechanism
  - When iCloud reads or writes, your app gets a chance to do things first

File Coordination

File Coordination

- NSFileCoordinator

File Coordination

- NSFileCoordinator
  - The class you use to do *coordinated* file access

File Coordination

- NSFileCoordinator
  - The class you use to do *coordinated* file access
- NSFilePresenter

File Coordination

- NSFileCoordinator
  - The class you use to do *coordinated* file access
- NSFilePresenter
  - The protocol you implement to hear about coordinated file access

File Coordination

- NSFileCoordinator
  - The class you use to do *coordinated* file access
- NSFilePresenter
  - The protocol you implement to hear about coordinated file access
  - NSDocument and UIDocument conform to it

File Coordination

- NSFileCoordinator
  - The class you use to do *coordinated* file access
- NSFilePresenter
  - The protocol you implement to hear about coordinated file access
  - NSDocument and UIDocument conform to it
- OS X 10.7 and iOS 5

File Coordination

- NSFileCoordinator
  - The class you use to do *coordinated* file access
- NSFilePresenter
  - The protocol you implement to hear about coordinated file access
  - NSDocument and UIDocument conform to it
- OS X 10.7 and iOS 5
- Used by more than just iCloud

NSFileCoordinator

Tell us what you are doing, we will 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 are doing, we will 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

Two ways to use it

NSFilePresenter

Two ways to use it

- Register a file presenter of an individual file

NSFilePresenter

Two ways to use it

- Register a file presenter of an individual file
  - Hear about the file changing and moving

NSFilePresenter

Two ways to use it

- Register a file presenter of an individual file
  - Hear about the file changing and moving
  - Get asked to save changes

NSFilePresenter

Two ways to use it

- Register a file presenter of an individual file
  - Hear about the file changing and moving
  - Get asked to save changes
  - Get asked to accommodate deletion

NSFilePresenter

Two ways to use it

- Register a file presenter of an individual file
  - Hear about the file changing and moving
  - Get asked to save changes
  - Get asked to accommodate deletion
  - Get asked to *relinquish* to readers and writers

NSFilePresenter

Two ways to use it

- Register a file presenter of an individual file
  - Hear about the file changing and moving
  - Get asked to save changes
  - Get asked to accommodate deletion
  - Get asked to *relinquish* to readers and writers
- Register a file presenter of an entire directory tree

NSFilePresenter

Two ways to use it

- Register a file presenter of an individual file
  - Hear about the file changing and moving
  - Get asked to save changes
  - Get asked to accommodate deletion
  - Get asked to *relinquish* to readers and writers
- Register a file presenter of an entire directory tree
  - Hear about files changing and moving

NSFilePresenter

Two ways to use it

- Register a file presenter of an individual file
  - Hear about the file changing and moving
  - Get asked to save changes
  - Get asked to accommodate deletion
  - Get asked to *relinquish* to readers and writers
- Register a file presenter of an entire directory tree
  - Hear about files changing and moving
- You will probably use it both ways

NSFilePresenter

Get notified about an individual file

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

NSFilePresenter

Get notified about an individual file

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

NSFilePresenter

Get notified about an individual file

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

NSFilePresenter

Get told to do important things

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

NSFilePresenter

Get told to do important things

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

- You register when you first *present* the corresponding item in the UI

NSFilePresenter

Get told to do important things

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

- You register when you first *present* the corresponding item in the UI
- Stay registered until you are done letting the user view and edit it

NSFilePresenter

Get told to do important things

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

- When iCloud needs to write, you get a chance to write first

NSFilePresenter

Get told to do important things

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

- When iCloud needs to delete, you get a chance to stop presenting first

NSFilePresenter

Get told to do important things

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

- When iCloud needs to delete, you get a chance to stop presenting first
- Should deregister your file presenter too

NSFilePresenter

Get asked to relinquish a file

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

NSFilePresenter

Get asked to relinquish a file

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

- Your first and last notification that something is happening

NSFilePresenter

Get asked to relinquish a file

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

- Your first and last notification that something is happening
- Delineate batches of the other messages

NSFilePresenter

Get notified about things in your directory

```
- (void)presentedSubitemDidChangeAtURL:(NSURL *)url;
- (void)presentedSubitemAtURL:(NSURL *)oldURL didMoveToURL:(NSURL *)newURL;
```

NSFilePresenter

Get notified about things in your directory

```
- (void)presentedSubitemDidChangeAtURL:(NSURL *)url;
- (void)presentedSubitemAtURL:(NSURL *)oldURL didMoveToURL:(NSURL *)newURL;
```

NSFilePresenter

Get notified about things in your directory

```
- (void)presentedSubitemDidChangeAtURL:(NSURL *)url;
- (void)presentedSubitemAtURL:(NSURL *)oldURL didMoveToURL:(NSURL *)newURL;
```

NSFilePresenter

Get notified about things in your directory

```
- (void)presentedSubitemDidChangeAtURL:(NSURL *)url;
- (void)presentedSubitemAtURL:(NSURL *)oldURL didMoveToURL:(NSURL *)newURL;
```

- (Ignore those other "subitem" methods you see in the header file)

A Shoebox App

User Adds an Item

User Adds an Item

App Writes a New File Using NSFileCoordinator

App Writes a New File Using NSFileCoordinator

Writing a New File with NSFileCoordinator

```
- (BOOL)saveAndReturnError:(NSError **)outError {
    NSURL *url = [self url];
    __block BOOL didWrite = NO;
    NSFileCoordinator* fc = [[NSFileCoordinator alloc]
                                initWithFilePresenter:self];
    [fc coordinateWritingItemAtURL:url
                           options:NSFileCoordinatorWritingForReplacing
                             error:outError
                        byAccessor:^(NSURL *updatedURL) {
        NSFileWrapper *fw = [self fileWrapper];
        didWrite = [fw writeToURL:updatedURL options:0
              originalContentsURL:nil error:outError];
    }];
    return didWrite;
}
```

Writing a New File with NSFileCoordinator

```
- (BOOL)saveAndReturnError:(NSError **)outError {
    NSURL *url = [self url];
    __block BOOL didWrite = NO;
    NSFileCoordinator* fc = [[NSFileCoordinator alloc]
                                initWithFilePresenter:self];
    [fc coordinateWritingItemAtURL:url
                           options:NSFileCoordinatorWritingForReplacing
                             error:outError
                        byAccessor:^(NSURL *updatedURL) {
        NSFileWrapper *fw = [self fileWrapper];
        didWrite = [fw writeToURL:updatedURL options:0
              originalContentsURL:nil error:outError];
    }];
    return didWrite;
}
```

Writing a New File with NSFileCoordinator

```
- (BOOL)saveAndReturnError:(NSError **)outError {
    NSURL *url = [self url];
    __block BOOL didWrite = NO;
    NSFileCoordinator* fc = [[NSFileCoordinator alloc]
                                initWithFilePresenter:self];
    [fc coordinateWritingItemAtURL:url
                           options:NSFileCoordinatorWritingForReplacing
                             error:outError
                        byAccessor:^(NSURL *updatedURL) {
        NSFileWrapper *fw = [self fileWrapper];
        didWrite = [fw writeToURL:updatedURL options:0
              originalContentsURL:nil error:outError];
    }];
    return didWrite;
}
```

iCloud Uploads the File

iCloud Uploads the File

Same App Running on iPad

iCloud Downloads File Metadata

iCloud Downloads File Metadata

App's File Presenter Receives a Message

Responding to a New File

```
- (void)presentedSubitemDidChangeAtURL:(NSURL *)url {
    NSFileCoordinator* fc = [[NSFileCoordinator alloc]
                                initWithFilePresenter:self];
    [fc coordinateReadingItemAtURL:url options:0 error:outError
                        byAccessor:^(NSURL *updatedURL) {
        Picture *picture = [self pictureAtURL:updatedURL];
        if (picture) {
            [picture loadFromURL:updatedURL];
        } else {
            [self addPictureWithURL:updatedURL];
        }
    }];
}
```

Responding to a New File

```
- (void)presentedSubitemDidChangeAtURL:(NSURL *)url {
    NSFileCoordinator* fc = [[NSFileCoordinator alloc]
                                initWithFilePresenter:self];
    [fc coordinateReadingItemAtURL:url options:0 error:outError
                        byAccessor:^(NSURL *updatedURL) {
        Picture *picture = [self pictureAtURL:updatedURL];
        if (picture) {
            [picture loadFromURL:updatedURL];
        } else {
            [self addPictureWithURL:updatedURL];
        }
    }];
}
```

Responding to a New File

```
- (void)presentedSubitemDidChangeAtURL:(NSURL *)url {
    NSFileCoordinator* fc = [[NSFileCoordinator alloc]
                                initWithFilePresenter:self];
    [fc coordinateReadingItemAtURL:url options:0 error:outError
                        byAccessor:^(NSURL *updatedURL) {
        Picture *picture = [self pictureAtURL:updatedURL];
        if (picture) {
            [picture loadFromURL:updatedURL];
        } else {
            [self addPictureWithURL:updatedURL];
        }
    }];
}
```

App's Coordinated Read Triggers Downloading

App's Coordinated Read Triggers Downloading

Reading the New File

```
- (void)presentedSubitemDidChangeAtURL:(NSURL *)url {
    NSFileCoordinator* fc = [[NSFileCoordinator alloc]
                                initWithFilePresenter:self];
    [fc coordinateReadingItemAtURL:url options:0 error:outError
                        byAccessor:^(NSURL *updatedURL) {
        Picture *picture = [self pictureAtURL:updatedURL];
        if (picture) {
            [picture loadFromURL:updatedURL];
        } else {
            [self addPictureWithURL:updatedURL];
        }
    }];
}
```

App Displays the New Item

App Displays the New Item

iPad and Mac Show the Same Items

File Versions

File Versions

File Versions

- Users can edit on multiple devices at once

File Versions

- Users can edit on multiple devices at once
- Conflicts!

File Versions

- Users can edit on multiple devices at once
- Conflicts!
- iCloud *senses* conflicts
  - Picks a winner
  - Puts the winning contents in the file
  - Even when your app is not running
    - Every file always has something decent in it

File Versions

- Users can edit on multiple devices at once
- Conflicts!
- iCloud *senses* conflicts
  - Picks a winner
  - Puts the winning contents in the file
  - Even when your app is not running
    - Every file always has something decent in it
- iCloud does not *resolve* conflicts

File Versions

- Your app must resolve conflicts

File Versions

- Your app must resolve conflicts
- Might have to look at losers

File Versions

- Your app must resolve conflicts
- Might have to look at losers
- Where did iCloud leave them?

NSFileVersion

It is what conflict losers become

```
+ (NSFileVersion *)currentVersionOfItemAtURL:(NSURL *)url;
+ (NSArray *)otherVersionsOfItemAtURL:(NSURL *)url;
+ (NSArray *)unresolvedConflictVersionsOfItemAtURL:(NSURL *)url;
```

NSFileVersion

It is what conflict losers become

```
+ (NSFileVersion *)currentVersionOfItemAtURL:(NSURL *)url;
+ (NSArray *)otherVersionsOfItemAtURL:(NSURL *)url;
+ (NSArray *)unresolvedConflictVersionsOfItemAtURL:(NSURL *)url;
```

NSFileVersion

It is what conflict losers become

```
+ (NSFileVersion *)currentVersionOfItemAtURL:(NSURL *)url;
+ (NSArray *)otherVersionsOfItemAtURL:(NSURL *)url;
+ (NSArray *)unresolvedConflictVersionsOfItemAtURL:(NSURL *)url;
```

NSFileVersion

You can present them to the user

- Properties you can use

```
URL
localizedName
localizedNameOfSavingComputer
modificationDate
```

NSFileVersion

It is what you use to resolve conflicts

```
- (NSURL *)replaceItemAtURL:(NSURL *)url
                    options:(NSFileVersionReplacingOptions)options
                      error:(NSError **)error;
```

- Make a real file out of a version

NSFileVersion

It is what you use to resolve conflicts

```
- (NSURL *)replaceItemAtURL:(NSURL *)url
                    options:(NSFileVersionReplacingOptions)options
                      error:(NSError **)error;
```

- Make a real file out of a version
- Maybe replace the file that contains the current version

NSFileVersion

It is what you use to resolve conflicts

```
- (NSURL *)replaceItemAtURL:(NSURL *)url
                    options:(NSFileVersionReplacingOptions)options
                      error:(NSError **)error;
```

- Make a real file out of a version
- Maybe replace the file that contains the current version
- Maybe make a new file off to the side

NSFileVersion

It is what you use to resolve conflicts

- Another property you can use

  `resolved`

NSFileVersion

It is what you use to resolve conflicts

- Another property you can use

  `resolved`

- This one is not read-only

NSFileVersion

It is what you use to resolve conflicts

- Another property you can use

  `resolved`

- This one is not read-only
- Setting it to YES tells iCloud it can discard the conflict loser lazily

NSFilePresenter Messages About Versions

Conflicts can be sensed at any time

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

NSFilePresenter Messages About Versions

Conflicts can be sensed at any time

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

- One more property you can use

```
conflict
```

NSFilePresenter Messages About Versions

Conflicts can be resolved at any time

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

NSFilePresenter Messages About Versions

Conflicts can be resolved at any time

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

- The user might have resolved the conflict on another device

iPad in Airplane Mode

User Changes an Item on Mac

User Changes an Item on Mac

App Writes Changed File

App Writes Changed File

iCloud Uploads the Change

iCloud Uploads the Change

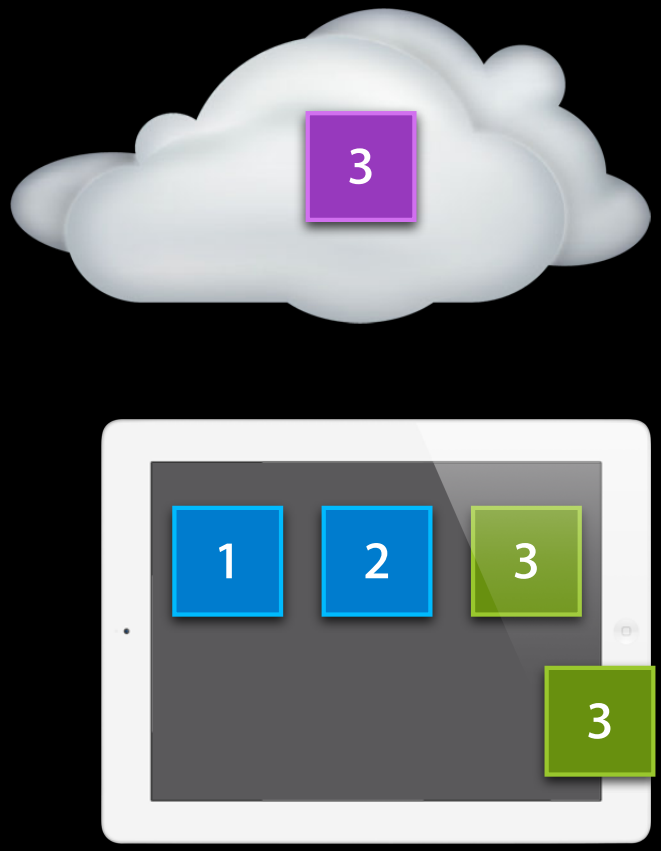
User Changes the Same Item on iPad

User Changes the Same Item on iPad

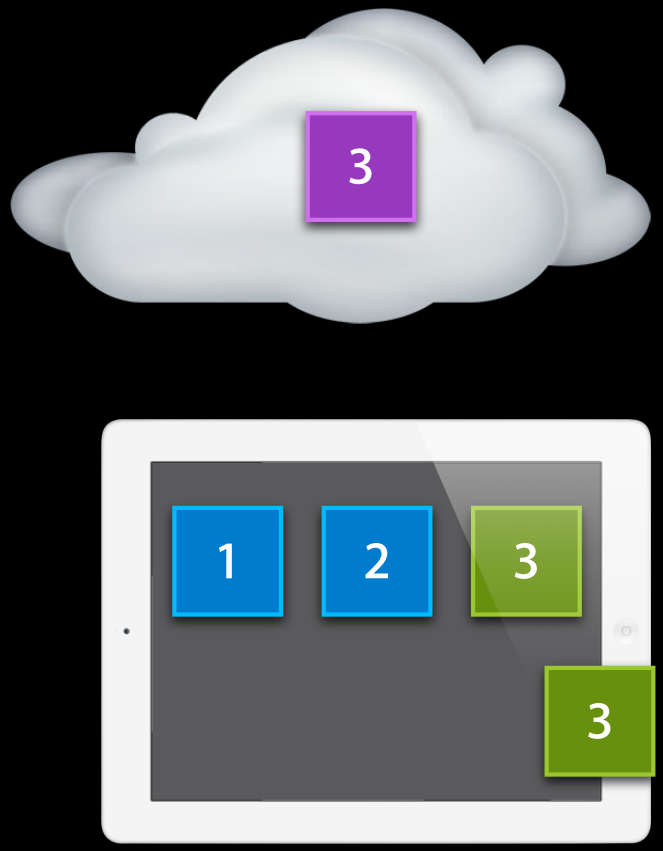
App Writes Changed File

App Writes Changed File

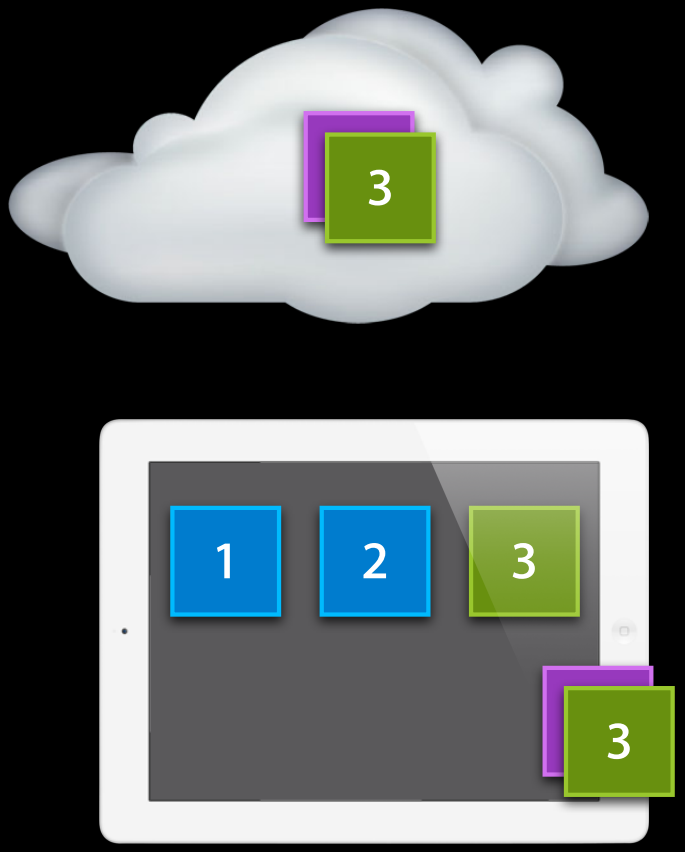
User Takes iPad out of Airplane Mode



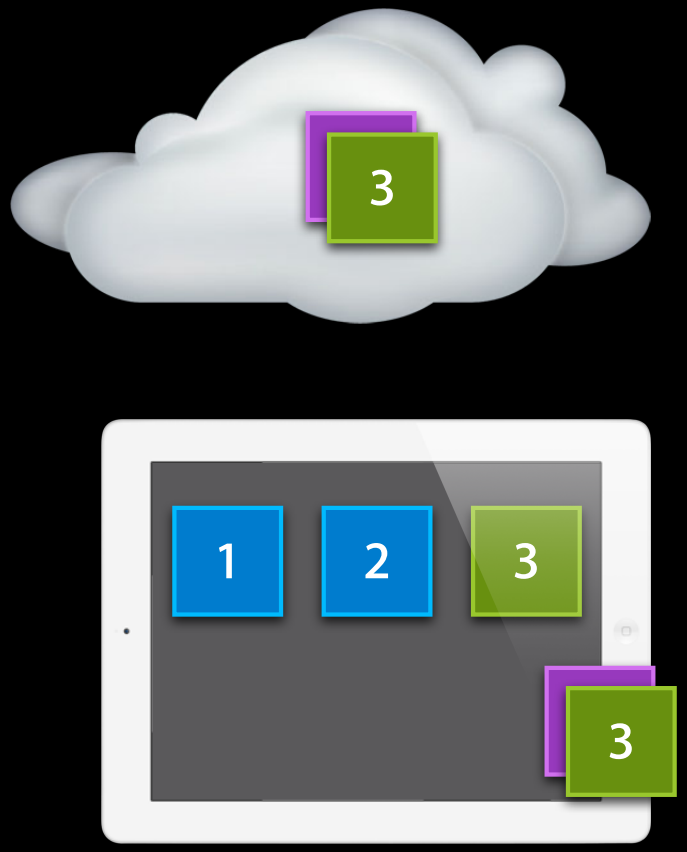
Versions Are Uploaded and Downloaded



Versions Are Uploaded and Downloaded



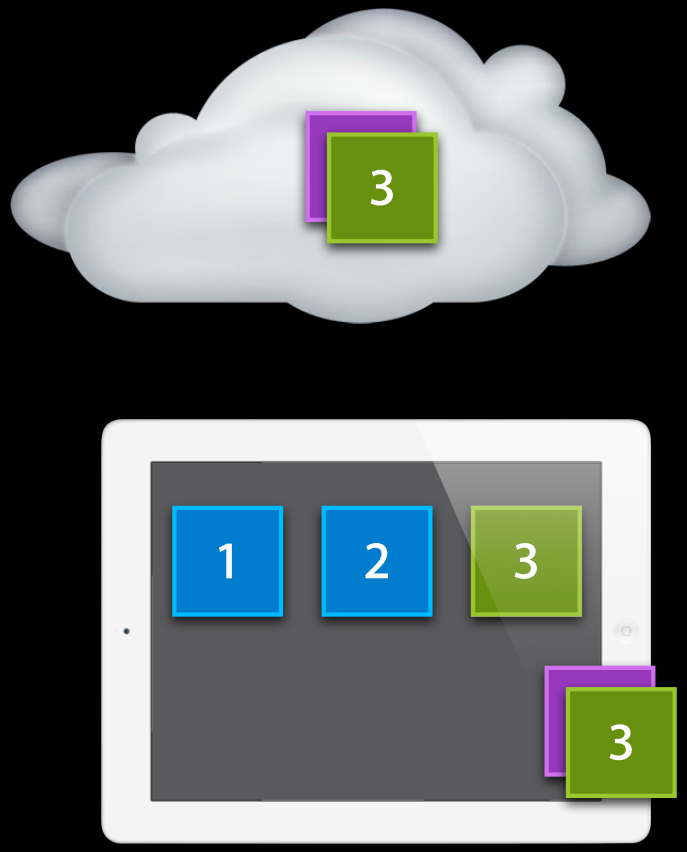
App's File Presenter Receives a Message



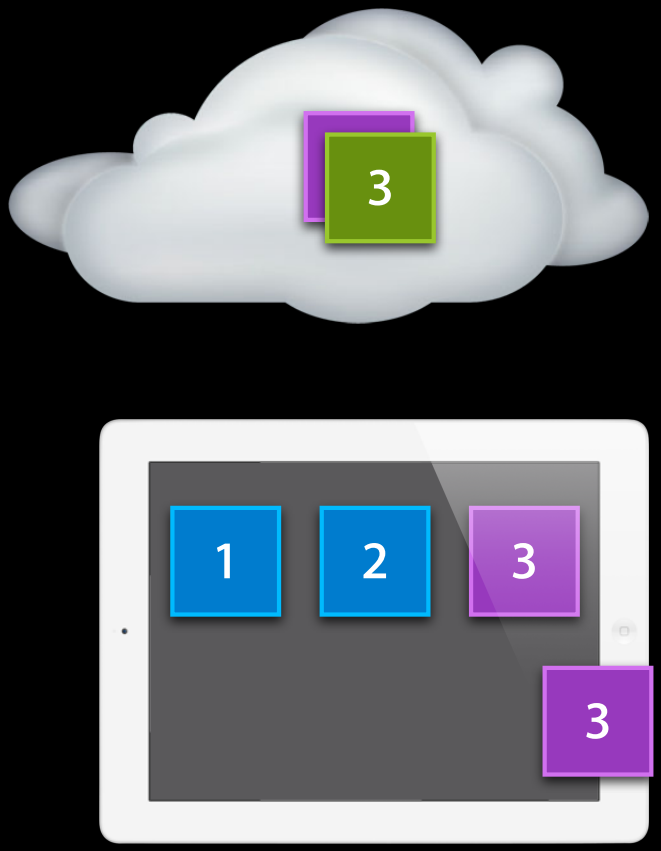
Responding to a Conflict

```
- (void)presentedSubitemAtURL:(NSURL *)url
                didGainVersion:(NSFileVersion *)version {
    if (version.isConflict) {
        Picture *picture = [self pictureAtURL:url];
        if (picture) {
            [self presentConflictVersion:version forPicture:picture];
        }
    }
}
```

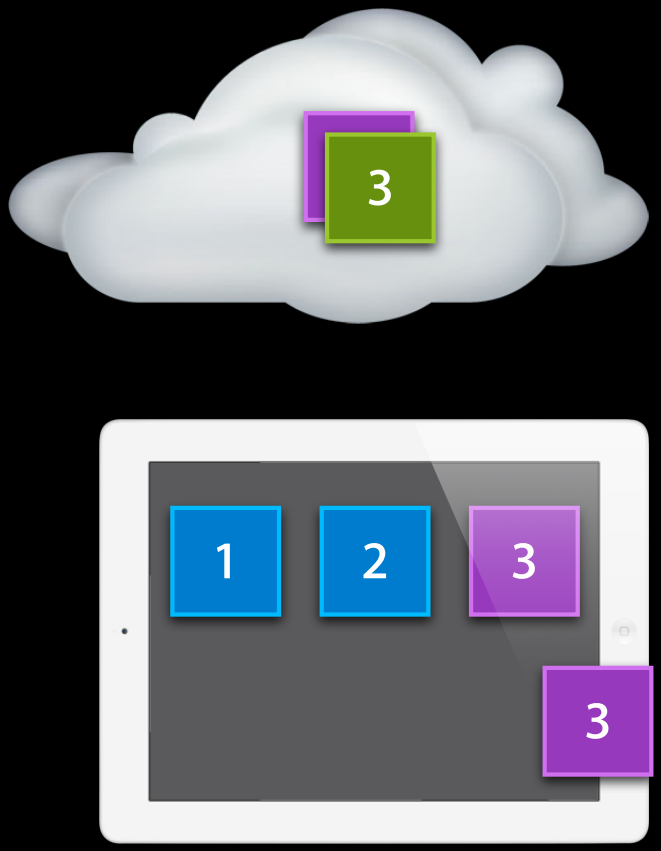
App Resolved the Conflict



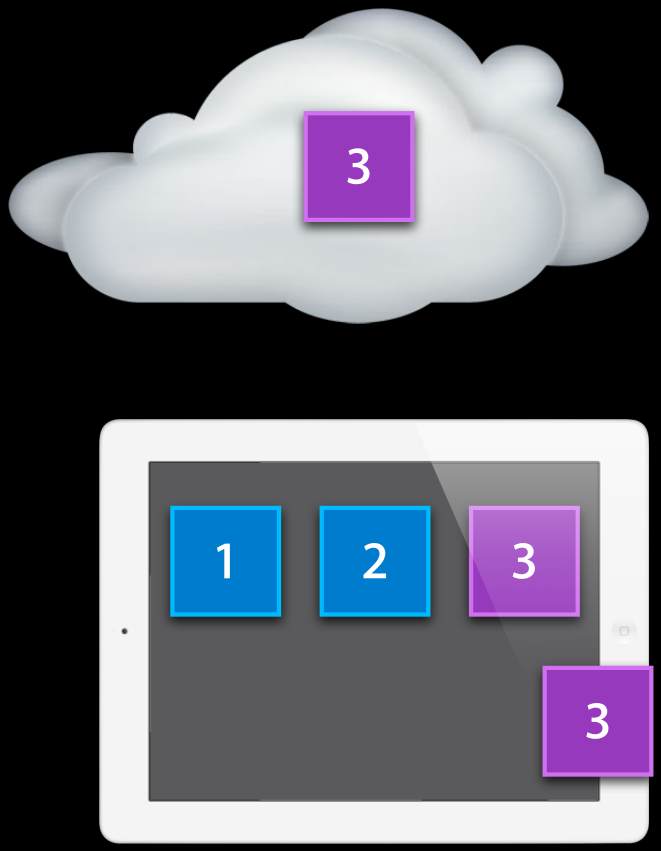
App Resolved the Conflict



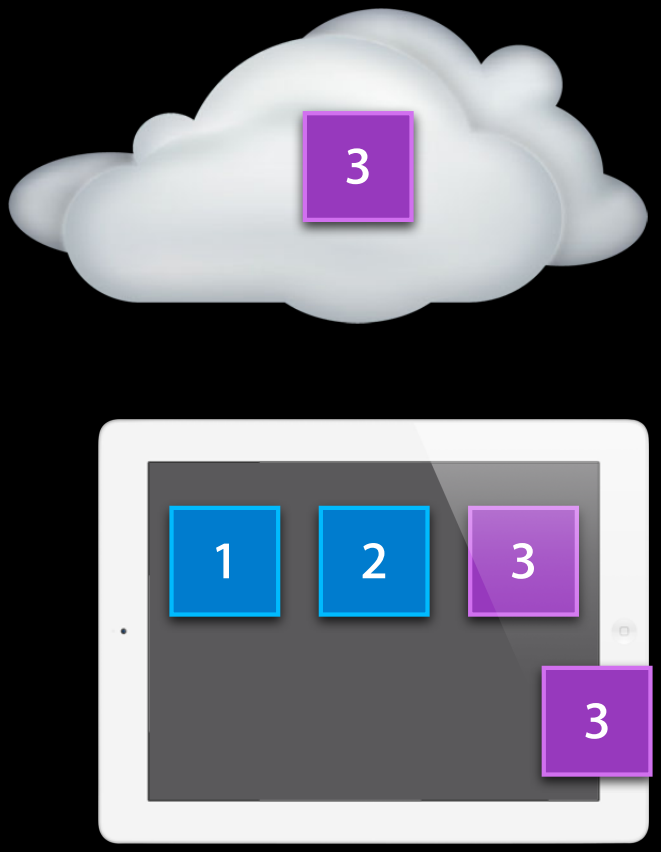
Conflict Resolution Propagates to the Cloud



Conflict Resolution Propagates to the Cloud



Conflict Resolution Propagates to the Mac



Conflict Resolution Propagates to the Mac

Conflict Resolution Propagates to the Mac

Conflict Resolution Propagates to the Mac

iPad and Mac Show the Same Items

Tips and Advice

App Startup

- Use -[NSFileManager ubiquityIdentityToken] to see if iCloud is on

App Startup

- Use -[NSFileManager ubiquityIdentityToken] to see if iCloud is on
  - Fast enough to use on the main thread

App Startup

- Use -[NSFileManager ubiquityIdentityToken] to see if iCloud is on
  - Fast enough to use on the main thread
  - -URLForUbiquityContainerIdentifier: is not

App Startup

- Use -[NSFileManager ubiquityIdentityToken] to see if iCloud is on
  - Fast enough to use on the main thread
  - -URLForUbiquityContainerIdentifier: is not
- Listen for NSUbiquityIdentityDidChangeNotification

App Startup

- Use -[NSFileManager ubiquityIdentityToken] to see if iCloud is on
  - Fast enough to use on the main thread
  - -URLForUbiquityContainerIdentifier: is not
- Listen for NSUbiquityIdentityDidChangeNotification
- New in OS X 10.8 and iOS 6

Reading Files

- Coordinated reading can trigger downloading

Reading Files

- Coordinated reading can trigger downloading
  - Can take a while

Reading Files

- Coordinated reading can trigger downloading
  - Can take a while
  - Do not do it on the main thread

Reading Files

- Coordinated reading can trigger downloading
  - Can take a while
  - Do not do it on the main thread
- Do good error checking

Reading Files

- Coordinated reading can trigger downloading
  - Can take a while
  - Do not do it on the main thread
- Do good error checking
  - The file might be deleted while you are waiting to read

Reading Files

- Coordinated reading can trigger downloading
  - Can take a while
  - Do not do it on the main thread
- Do good error checking
  - The file might be deleted while you are waiting to read
  - Your app is about to get a notification about it

Resolving Conflicts

- Accessing a file's versions is like accessing its contents

Resolving Conflicts

- Accessing a file's versions is like accessing its contents
- Do coordinated reading when enumerating or reading versions

Resolving Conflicts

- Accessing a file's versions is like accessing its contents
- Do coordinated reading when enumerating or reading versions
- Do coordinated writing when adding, removing, or resolving versions

When iCloud Deletes Files

- Your NSFilePresenter might be sent
  -accommodatePresentedItemDeletionWithCompletionHandler:

When iCloud Deletes Files

- Your NSFilePresenter might be sent
  -accommodatePresentedItemDeletionWithCompletionHandler:
- And then the file actually gets moved
  - But you only notice if you keep watching the file

When iCloud Deletes Files

- Your NSFilePresenter might be sent

  -accommodatePresentedItemDeletionWithCompletionHandler:
- And then the file actually gets moved
  - But you only notice if you keep watching the file
- Do not take advantage of implementation details you will see if you keep watching

Save Settings in the Right Place

- Preferences
  - Scroll bar and window positions
  - All of our devices support different screen sizes

Save Settings in the Right Place

- Preferences
  - Scroll bar and window positions
  - All of our devices support different screen sizes
- Some discardable settings are OK
  - For example, Keynote's current slide
  - Saving when you're saving for real changes is OK

Save Settings in the Right Place

- Some things are never OK
  - Time stamps

Save Settings in the Right Place

- Some things are never OK
  - Time stamps
- Keep them out of the cloud
  - False conflicts
  - Sync loops

File Format Compatibility

- Shoeboxes stay around forever

File Format Compatibility

- Shoeboxes stay around forever
- Do things that work on both platforms

File Format Compatibility

- Shoeboxes stay around forever
- Do things that work on both platforms
- Different versions of your app running at the same time
  - Editing with old versions of your app

Summary

- Use NSFileCoordinator when you read and write files
- Use NSFilePresenter to hear about changes that happened
- Use NSFileVersion to deal with conflicts

More Information

Mike Jurewitz
Developer Tools and Frameworks Evangelist
jurewitz@apple.com

iCloud Design Guide
http://developer.apple.com

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

Related Sessions

Using iCloud with UIDocument	Marina Wednesday 10:15AM
Using iCloud with NSDocument	Marina Wednesday 3:15PM
Using iCloud with Core Data	Mission Wednesday 4:30PM

Labs

iCloud Storage Lab	Essentials Lab B Thursday 4:30PM
iCloud Storage Lab	Essentials Lab B Friday 11:30AM


WWDC2012