THE FOLLOWING TALK IS RATED

OBJ-C

WHAT YOU ARE ABOUT TO WATCH CONTAINS EXPLICIT SYNTAX AND MAY NOT BE SUITABLE FOR ALL AUDIENCES.

VIEWER DISCRETION IS ADVISED.

Intro

Current Life

Jeffrey Bergier

iOS Developer at Mercari

New to Tokyo from San Francisco

🌉 ➡️ 🗼

@jeffburg on Twitter

tokyojeff.com on Web

Past Life

Jeffrey Bergier

iOS Developer at Topology Eyewear

UX Designer at Riverbed Technology

BS Industrial Design from SFSU

@jeffburg on Twitter

tokyojeff.com on Web

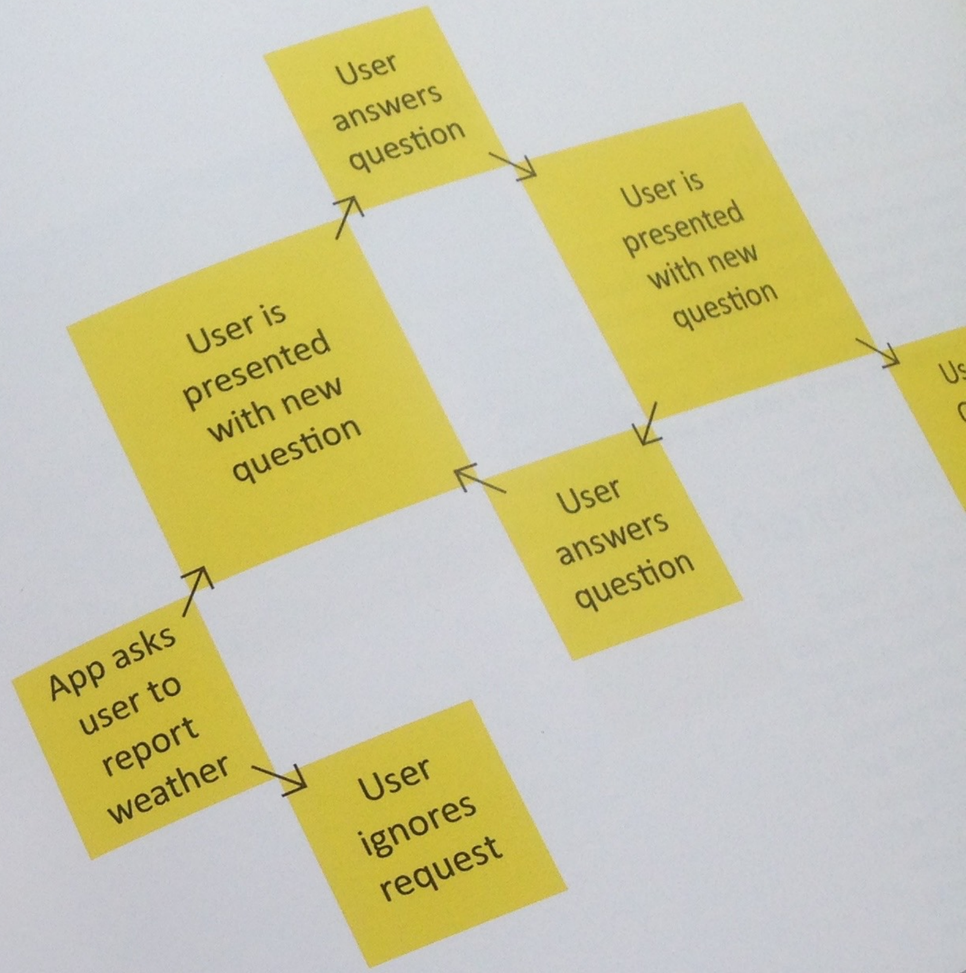
Function Diagram

Reporting the Weather

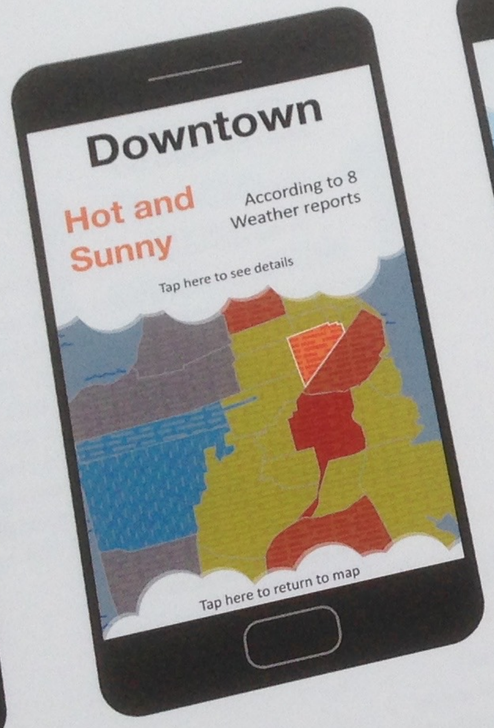
The simplest way to have the user report the weather would be to add a button around the map that allows the user to do so. However, I thought that if the application were smarter than that and could "ask" the user to report the weather then they would be more likely to do so. So my idea was to have the application discover when the user has moved from one district to another whilst it was closed and ask the user if they would like to report the weather in their new location.

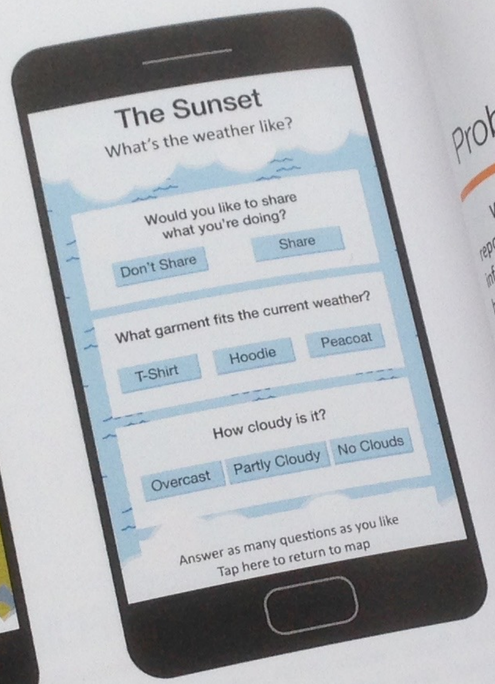
Continuous Questions

Some users have more time to spend on social networks than others. I thought that a great way to take advantage of this was to have a continuous stream of questions ready when the user decided they wanted to report the weather. That way, users with no time would answer only one question and then quit. Whereas users with lots of time could continue to answer questions for as long as they wanted. This would encourage the most usage while still getting a lot of data.



Execution Summary





Problem Solved

Well, mostly. The current solution requires that users actively report weather information. Only then can other users read the weather information presented by the application. The best case scenario would have made the weather information gathering automatic. Specifically, Twitter would have been great for this. Twitter users would post weather information and then this application would secretly analyze the information and show it on the map. Alas, not enough Twitter users use the geolocation feature. So while this social network adds a dynamics feature to the application, it is truly a means to an end.

Pros

The pros of the solution involve an innovative way to plot weather on a map. The weather bar is better than pins on a map. Weather is more spread out than a specific address, which is what the pins were designed for. The weather bar also allows easier combining of similar information into one graphical asset to then present to the user. This simplifies the display for the user. The color system used is easy to understand because its used by the standard of standards in weather, the National Weather Service. The solution also allows users to get deep into the application to report as much information as they want or to read as much information as they want. It also allows users to get information quickly and to report information quickly.

Cons

The social networking aspect of the solu
said before is a means to an end. Finding ano
to collect weather information within the
be ideal. The social network aspect pres
such as the chicken and egg problem. It
that users are interested enough in the
regular basis to report the weather oft


iOS Conf SG| 2019
17th to 19th, January, 2019
BUY TICKETS

Fear Not,

for the Filesystem is with You

Outline

- State of Persistence in iOS and macOS
- Proposed Persistence Idea
- Review Foundation API
- Details of Solution
- Q&A
- Bonus: Syncing (time permitting)

Outline

- State of Persistence in iOS and macOS
- Proposed Persistence Idea
- Review Foundation API
- Details of Solution
- Q&A
- Bonus: Syncing (time permitting)

Persistence is Important

- Our apps can be paused or killed at any time
- Network connections are often poor
- Users assume their data is always saved
- The architecture of the app is strongly tied to the persistence layer
- The UX of the app is strongly tied to the quality of persistence

Desired Features

1. Complex models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
3. Ability to sync user data with a server
4. Data change API to enable:
    1. Lightweight view controllers
    2. Animated tableview updates
5. Support App Extensions (Interprocess safety)
6. Speed

No Solution

- NSUserDefaults
  - No queries, not good for complex models
- NSSecureCoding / Codable
  - No queries, no change API, not good for large arrays, or complex models
- Core Data
  - No syncing
- Cloud Kit
  - No persistence

but…

Fear Not,

for the Filesystem is with You

Outline

- State of Persistence in iOS and macOS
- Proposed Persistence Idea
- Review Foundation API
- Details of Solution
- Q&A
- Bonus: Syncing (time permitting)

Outline

- State of Persistence in iOS and macOS
- Proposed Persistence Idea
- Review Foundation API
- Details of Solution
- Q&A
- Bonus: Syncing (time permitting)

Proposed Persistence Idea

- Use 100% Foundation API
- Support all features mentioned before

Whats the name of the pod?

- I **don’t** want to push a library
- I **don’t** want to pretend that my solution is the only solution
- I **hope** this talk encourages you to think a little bit about persistence
- I **hope** that you also consider using the Filesystem
  - Its powerful and flexible.

- But to help explain everything, I have a repo on Github
  - Its the code I'll be showing in these slides
  - Its MIT licensed, so do as you wish
- https://www.github.com/jeffreybergier/JSBFilesystem

Desired Features

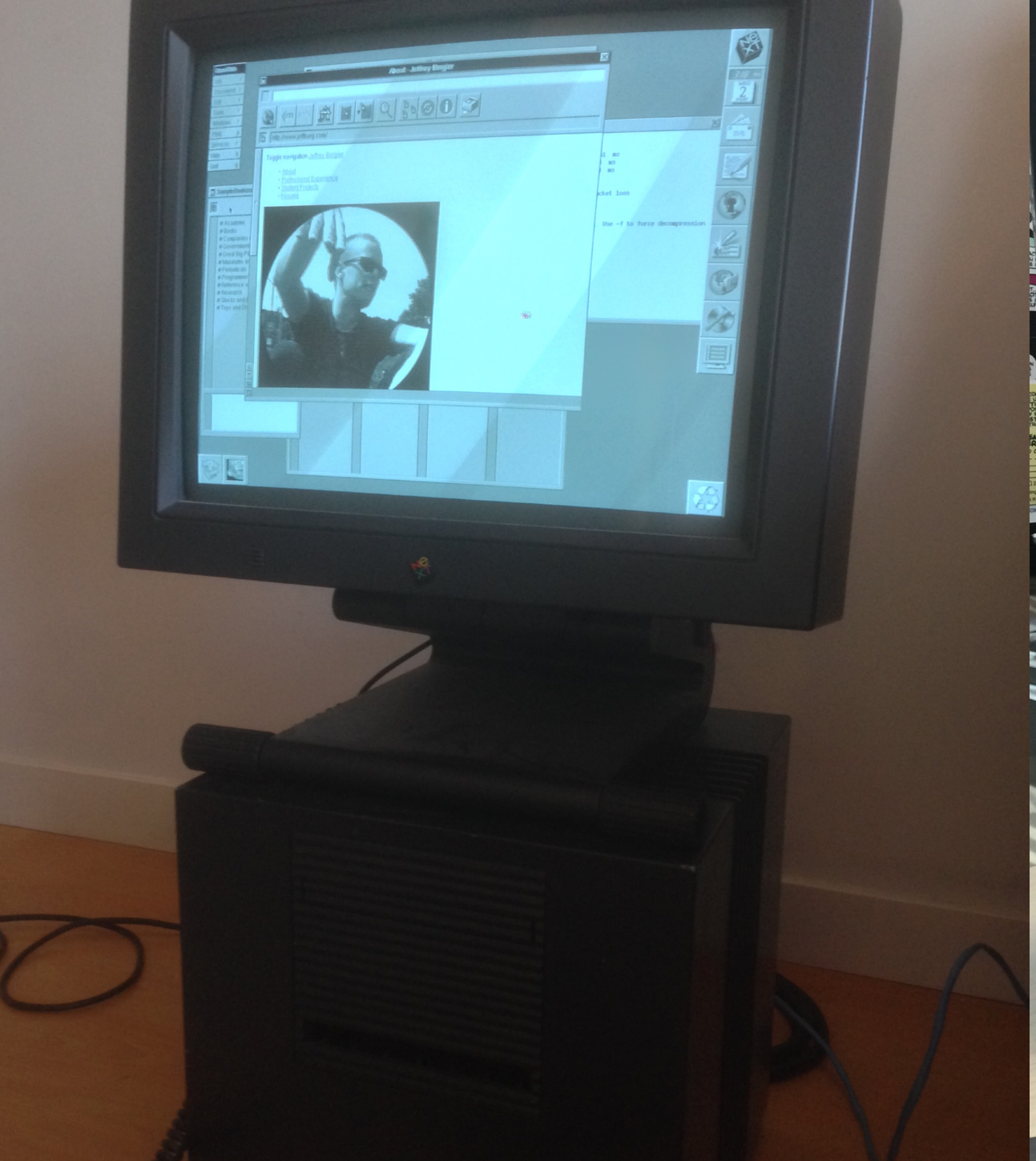
1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. Lightweight view controllers
    2. Animated tableview updates
5. Support App Extensions (Interprocess safety)
6. Speed

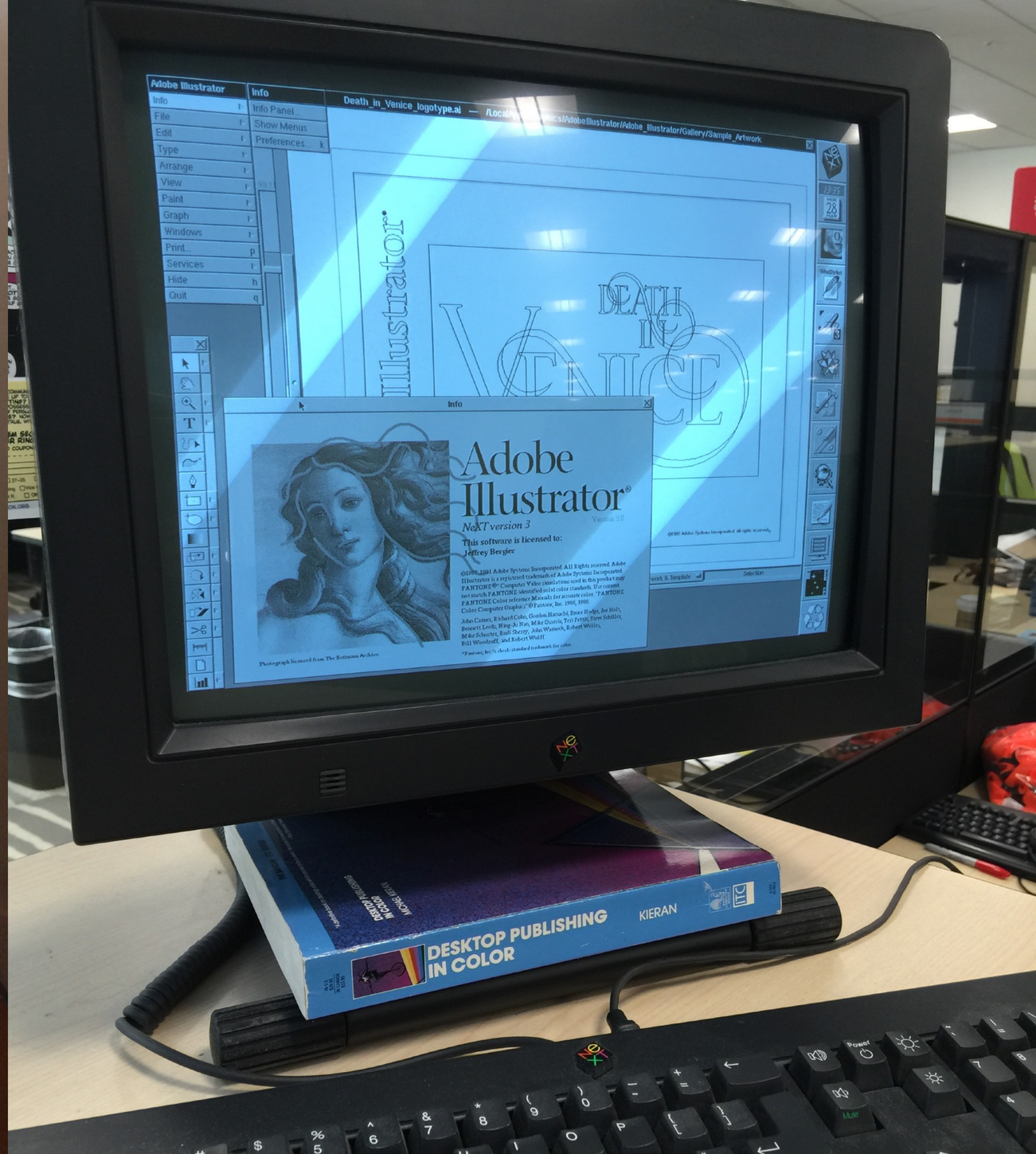
Desired Features

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. Lightweight view controllers
    2. Animated tableview updates
5. Support App Extensions (Interprocess safety)
6. Speed

Complex Models

- NSSecureCoding and Codable
- NSFileWrapper
  - Old API that allows complex models to be stored in packages
- Using these two together we can represent our complex models




Adobe Illustrator
Info
Info Panel...
Show Menus
Preferences...
File
Edit
Type
Arrange
View
Paint
Graph
Windows
Print...
Services
Hide
Quit
Death_in_Venice_logotype.ai
Info
Adobe Illustrator®
NeXT version 3
This software is licensed to:
Jeffrey Bergier
DESKTOP PUBLISHING IN COLOR
KIERAN

PB
Info
Project
File
Edit
Format
Tools
Windows
Print... p
Services
Hide h
Quit q

Contacts_Live — ~/Projects

NSFileWrapper.h

Headers
Documentation

NSDataLinkManager.h
NSDataLinkPanel.h
NSDPSContext.h
NSDPSServerContext.h
NSDragging.h
NSEPSImageRep.h
NSErrors.h
NSEvent.h
NSFileWrapper.h

@NSFileWrapper

-initDirectoryWithFileWrappers:
-initRegularFileWithContents:
-initSymbolicLinkWithDestination
-initWithPath:
-initWithSerializedRepresentation
-writeToFile:atomically:updateFil
-serializedRepresentation
-setFilename:
-filename

```
/*
        NSFileWrapper.h
        Application Kit
        Copyright (c) 1995-1996, NeXT Software, Inc.
        All rights reserved.
*/

#import <Foundation/Foundation.h>

@class NSImage;

@interface NSFileWrapper : NSObject <NSCoding> {
  @private
    id _impl;
    NSString *_fileName;
    NSString *_preferredFileName;
    NSDictionary *_fileAttributes;
    NSImage *_image;
    id _subdocuments;
}

////////////////////////////// Initialization ////////////////////////////////

- (id)initDirectoryWithFileWrappers:(NSDictionary *)docs;
    // Designated initializer. Inits new directory type instances.

- (id)initRegularFileWithContents:(NSData *)data;
    // Designated initializer. Inits new regular file type instances.

- (id)initSymbolicLinkWithDestination:(NSString *)path;
    // Designated initializer. Inits new symbolic link type instances.

- (id)initWithPath:(NSString *)path;
    // Designated initializer. Inits the instance from the file, directory, or symbolic link at path.  This can end up creating a
whole tree of instances with a directory type instance at the root.
```

16:07
SAT 5 JAN

Desired Features

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. Lightweight view controllers
    2. Animated tableview updates
5. Support App Extensions (Interprocess safety)
6. Speed

Desired Features

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. Lightweight view controllers
    2. Animated tableview updates
5. Support App Extensions (Interprocess safety)
6. Speed

Queries

- Fast Sorting with NSFileManager, NSURL, and NSArray
  - File Name, Creation Date, Modification Date
- Filtering with an array of closures with signature (URL) -> Bool
  - First failure causes the file to be ignored

Desired Features

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. Lightweight view controllers
    2. Animated tableview updates
5. Support App Extensions (Interprocess safety)
6. Speed

Desired Features

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. Lightweight view controllers
    2. Animated tableview updates
5. Support App Extensions (Interprocess safety)
6. Speed

Sync

Because all of our data is a folder full of packages we have many options

- iCloud Drive
- Dropbox
- Git (with LibGit2 / Objective-Git)
- Heck, use RSYNC

Desired Features

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. Lightweight view controllers
    2. Animated tableview updates
5. Support App Extensions (Interprocess safety)
6. Speed

Desired Features

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. Lightweight view controllers
    2. Animated tableview updates
5. Support App Extensions (Interprocess safety)
6. Speed

Data Changes

- NSFilePresenter
  - Notifications from the system when a our files change
- Basic [(URL, Date)] diffing
  - Roll your own diffing
  - IGListKit 👉👉
  - Proposed Swift 5 Diffing Solution
- A ChangesObserved closure Provides the needed information to our UI

Desired Features

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. Lightweight view controllers
    2. Animated tableview updates
5. Support App Extensions (Interprocess safety)
6. Speed

Desired Features

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. Lightweight view controllers
    2. Animated tableview updates
5. Support App Extensions (Interprocess safety)
6. Speed

Extension Support

- Shared Document Containers
- NSFileCoordinator
  - API that automatically manages file access
    - Both within a single process and between processes
  - NSFileCoordinator and NSFilePresenter work hand in hand.

Desired Features

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. Lightweight view controllers
    2. Animated tableview updates
5. Support App Extensions (Interprocess safety)
6. Speed

Desired Features

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. Lightweight view controllers
    2. Animated tableview updates
5. Support App Extensions (Interprocess safety)
6. Speed

Speed

- NSFileManager only loads NSArray<NSURL> into memory
- Sorting uses resources built into NSURL
- We don’t load our complex models until we need them
  - Usually in `-tableView:cellForRowAtIndexPath:`

Desired Features

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. Lightweight view controllers
    2. Animated tableview updates
5. Support App Extensions (Interprocess safety)
6. Speed

Desired Features

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. Lightweight view controllers
    2. Animated tableview updates
5. Support App Extensions (Interprocess safety)
6. Speed

Demo

```
46.2 MB
Zero KB/s
Zero KB/s
m.a...-thread (serial)
ontroller.viewDidLo...
Main
tfetch-thread (7)
```

```
36      class func newVC(completion: ((UIViewController) -> Void)?) -> UIViewController {
37          let vc = ListTableViewController(style: .plain)
38          let navVC = UINavigationController(rootViewController: vc)
39          return navVC
40      }
41
42      private let directory: JSBFSObservedDirectory =
43          try! JSBFSObservedDirectory(base: .documentDirectory,
44                                      appendingPathComponent: "MyFiles_Deleted",
45                                      createIfNeeded: true,
46                                      sortedBy: .modificationNewestFirst,
47                                      filteredBy: nil,
48                                      changeKind: .modificationsAsInsertionsDeletions)
49
50      override func viewDidLoad() {
51          super.viewDidLoad()                                                    Thread 1:
52
53          // configure the change closure
54          self.directory.changesObserved = { [unowned self] changes in
55              // update the tableview
56              self.tableView.beginUpdates()
57              self.tableView.insertRows(at: changes.insertions.map({ IndexPath(item: $0, section: 0) }), with: .right)
58              changes.moves.forEach() { move in
59                  self.tableView.moveRow(at: IndexPath(item: move.from, section: 0), to: IndexPath(item: move.to, section: 0))
60              }
61              self.tableView.deleteRows(at: changes.deletions.map({ IndexPath(item: $0, section: 0) }), with: .left)
62              self.tableView.endUpdates()
63              // update the title
64              let fileCount = (try? self.directory.contentsCount()) ?? 0
65              self.title = "Showing \(fileCount) Items"
66              NSLog("%@", changes)
67          }
68          self.tableView.reloadData()
69          // set up a timer to modify the underlying data for display
```

JSBFilesystem_sample_app 〉 Thread 1 〉 0 ListTableViewController.viewDidLoad()

```
NIB Init: <JSBFilesystem_sample_app.ListTableViewController: 0x7fe5af50c380>

NIB Init: <UINavigationController: 0x7fe5af835400>

(lldb) po self.directory.url
▿ file:///Users/jeffreybergier/Library/Developer/CoreSimulator/Devices/A61F01FC-5FBA-4BB3-B6D8-DA78728D448A/data/Containers/Data/Application/C5E51B81-26AD
```

Outline

- State of Persistence in iOS and macOS
- Proposed Persistence Idea
- Review Foundation API
- Details of Solution
- Q&A
- Bonus: Syncing (time permitting)

Outline

- State of Persistence in iOS and macOS
- Proposed Persistence Idea
- Review Foundation API
- Details of Solution
- Q&A
- Bonus: Syncing (time permitting)

Review Foundation API

1. Complex Models (Containing simple as well as binary data)
    1. **NSFileWrapper**
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. **NSFilePresenter**
5. Support App Extensions (Interprocess safety)
    1. **NSFileCoordinator**
6. Speed

Review Foundation API

1. Complex Models (Containing simple as well as binary data)
    1. **NSFileWrapper**
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. **NSFilePresenter**
5. Support App Extensions (Interprocess safety)
    1. **NSFileCoordinator**
6. Speed

NSFileWrapper

```
do {
    let payloadData = try self.model.plistData()
    let payloadWrapper = FileWrapper(regularFileWithContents: payloadData)
    payloadWrapper.preferredFilename = "payload.plist"
    let imageData = try self.model.image.pngData()
    let imageWrapper = FileWrapper(regularFileWithContents: imageData)
    imageWrapper.preferredFilename = "image.png"
    let parentWrapper = FileWrapper(directoryWithFileWrappers: [
        "payload.plist": payloadWrapper,
        "image.png": imageWrapper
        ])
    let fileURL = self.directory.appendingPathComponent(self.fileName)
    try NSFileCoordinator.JSBFS_write(parentWrapper, to: fileURL, filePresenter: nil)
} catch {
    let e = String(describing: error)
    NSLog(e)
}
```

Review Foundation API

1. Complex Models (Containing simple as well as binary data)
    1. **NSFileWrapper**
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. **NSFilePresenter**
5. Support App Extensions (Interprocess safety)
    1. **NSFileCoordinator**
6. Speed

Review Foundation API

1. Complex Models (Containing simple as well as binary data)
    1. **NSFileWrapper**
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. **NSFilePresenter**
5. Support App Extensions (Interprocess safety)
    1. **NSFileCoordinator**
6. Speed

NSFilePresenter

Create an Object That Conforms to the Protocol

```
public protocol NSFilePresenter : NSObjectProtocol {

    public var presentedItemURL: URL? { get }

    optional public func presentedItemDidChange()

    public var presentedItemOperationQueue: OperationQueue { get }

}

NSFileCoordinator.addFilePresenter(myPresenter)
```

Review Foundation API

1. Complex Models (Containing simple as well as binary data)
    1. **NSFileWrapper**
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. **NSFilePresenter**
5. Support App Extensions (Interprocess safety)
    1. **NSFileCoordinator**
6. Speed

Review Foundation API

1. Complex Models (Containing simple as well as binary data)
    1. **NSFileWrapper**
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. **NSFilePresenter**
5. Support App Extensions (Interprocess safety)
    1. **NSFileCoordinator**
6. Speed

NSFileCoordinator

- Protects us from writing to a file while its being written to
- Protects us from reading a file when its being written to

NSFileCoordinator

- Apple’s First Block Based API
- The block is called synchronously, its not asynchronous
- Horrible Swift Interface

NSFileCoordinator

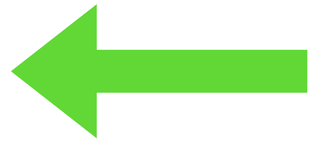
```
@interface NSFileCoordinator;

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

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


@end
```

NSFileCoordinator

```
open class NSFileCoordinator : NSObject {

    open func coordinate(readingItemAt url: URL,
                         options: NSFileCoordinator.ReadingOptions = [],
                         error outError: NSErrorPointer,
                         byAccessor reader: (URL) -> Void)

    open func coordinate(writingItemAt url: URL,
                         options: NSFileCoordinator.WritingOptions = [],
                         error outError: NSErrorPointer,
                         byAccessor writer: (URL) -> Void)

}
```

NSFileCoordinator

```
__block NSData* data = nil;
NSFileCoordinator* c = [[NSFileCoordinator alloc] initWithFilePresenter:nil];
[c coordinateReadingItemAtURL:url
                      options:NSFileCoordinatorReadingResolvesSymbolicLink
                        error:nil
                   byAccessor:^(NSURL*_Nonnull newURL)
{
    data = [NSData dataWithContentsOfURL:newURL options:0 error:nil];
}];
return data;
```



NSFileCoordinator

Bonus: Did you know that foundation can zip up files for you?

```
    /* Whether reading of an item is being done for the purpose of uploading. When using this
option, NSFileCoordinator will create a temporary snapshot of the item being read and will
relinquish its claim on the file once that snapshot is made to avoid blocking other coordinated
writes during a potentially long upload. If the item at the URL being read is a directory (such
as a document package), then the snapshot will be a new file that contains the zipped contents
of that directory, and the URL passed to the accessor block will locate that file.

    When using this option, you may upload the document outside of the accessor block. However,
you should open a file descriptor to the file or relocate the file within the accessor block
before you do so, because NSFileCoordinator will unlink the file after the block returns,
rendering it inaccessible via the URL.
    */
    NSFileCoordinatorReadingForUploading
```

Review Foundation API

1. Complex Models (Containing simple as well as binary data)
    1. **NSFileWrapper**
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. **NSFilePresenter**
5. Support App Extensions (Interprocess safety)
    1. **NSFileCoordinator**
6. Speed

Review Foundation API

1. Complex Models (Containing simple as well as binary data)
    1. **NSFileWrapper**
2. Queries for sorting and filtering data
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. **NSFilePresenter**
5. Support App Extensions (Interprocess safety)
    1. **NSFileCoordinator**
6. Speed

Outline

- State of Persistence in iOS and macOS
- Proposed Persistence Idea
- Review Foundation API
- Details of Solution
- Q&A
- Bonus: Syncing (time permitting)

Outline

- State of Persistence in iOS and macOS
- Proposed Persistence Idea
- Review Foundation API
- Details of Solution
- Q&A
- Bonus: Syncing (time permitting)

Details of Solution

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
    1. **JSBFSDirectory**
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. **JSBFSObservedDirectory**
5. Support App Extensions (Interprocess safety)
    1. **NSFileCoordinator Categories**
6. Speed

Details of Solution

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
    1. **JSBFSDirectory**
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. **JSBFSObservedDirectory**
5. Support App Extensions (Interprocess safety)
    1. **NSFileCoordinator Categories**
6. Speed

JSBFSDirectory

Gives a TableView Friendly API to a Directory

```
open class JSBFSDirectory : NSObject {

  open var url: URL { get }

  open var filteredBy: [@convention(block) (URL) -> Bool]?
  open var sortedBy: JSBFSDirectorySort

  open func contentsCount() throws -> Int
  open func url(at index: Int) throws -> URL

}
```

Sorting with [URL]

```
NSFileManager *fileManager = [NSFileManager defaultManager];
NSMutableArray *files = [[fileManager contentsOfDirectoryAtURL:
                                                          toryURL
                                    includingPropertiesForKeys:@[NSURLCreationDateKey]
                                                       options:0
                                                         error:nil] mutableCopy];
[files sortUsingComparator:^(NSURL *lURL, NSURL *rURL) {
    NSDate *lDate, *rDate;
    [lURL getResourceValue:&lDate forKey:NSURLCreationDateKey error:nil];
    [rURL getResourceValue:&rDate forKey:NSURLCreationDateKey error:nil];
    return [lDate compare:rDate];
}];
```

https://stackoverflow.com/a/48347245

Details of Solution

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
    1. **JSBFSDirectory**
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. **JSBFSObservedDirectory**
5. Support App Extensions (Interprocess safety)
    1. **NSFileCoordinator Categories**
6. Speed

Details of Solution

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
    1. **JSBFSDirectory**
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. **JSBFSObservedDirectory**
5. Support App Extensions (Interprocess safety)
    1. **NSFileCoordinator Categories**
6. Speed

JSBFSObservedDirectory

Uses NSFilePresenter to Observe and Notify of Changes to Its Directory

```
open class JSBFSObservedDirectory : JSBFSDirectory, NSFilePresenter {

  open var changesObserved: ((JSBFSDirectoryChanges) -> Void)?
  open func forceUpdate()

}
```

JSBFSDirectoryChanges

```
open class JSBFSDirectoryChanges : NSObject {

    open var insertions: IndexSet { get }
    open var deletions: IndexSet { get }
    open var moves: [JSBFSDirectoryChangesMove] { get }

}

open class JSBFSDirectoryChangesMove : NSObject {

    open var from: Int { get }
    open var to: Int { get }

}
```

JSBFSObservedDirectory

Diffing with IGListKit

```
- (void)presentedItemDidChange;
{
    NSError* error = nil;
    JSBFSObservedDirectoryChangeBlock block = [self changesObserved];
    JSBFSObservedDirectoryChangeKind changeKind = [self changeKind];
    NSArray<JSBFSFileComparison*>* lhs = [self internalState];
    NSArray<JSBFSFileComparison*>* rhs =
        [NSFileCoordinator JSBFS_comparableContentsOfDirectoryAtURL:[self url]
                                                           sortedBy:[self sortedBy]
                                                         filteredBy:[self filteredBy]
                                                      filePresenter:nil
                                                              error:&error];

    if (error) {
        NSLog(@"%@", error);
        rhs = [[NSArray alloc] init];
    }
    IGListIndexSetResult* result = IGListDiff(lhs, rhs, IGListDiffEquality);
    JSBFSDirectoryChanges* changes = [result changeObjectForChangeKind:changeKind];
    [self setInternalState:rhs];
    if (changes && block) {
        dispatch_async(dispatch_get_main_queue(), ^{
            block(changes);
        });
```

Details of Solution

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
    1. **JSBFSDirectory**
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. **JSBFSObservedDirectory**
5. Support App Extensions (Interprocess safety)
    1. **NSFileCoordinator Categories**
6. Speed

Details of Solution

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
    1. **JSBFSDirectory**
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. **JSBFSObservedDirectory**
5. Support App Extensions (Interprocess safety)
    1. **NSFileCoordinator Categories**
6. Speed

NSFileCoordinator

Swift Friendly Wrapper

```
extension NSFileCoordinator {

  open class func JSBFS_write(_ data: Data, to url: URL) throws
  open class func JSBFS_write(_ fileWrapper: FileWrapper, to url: URL) throws
  open class func JSBFS_readData(from url: URL) throws -> Data
  open class func JSBFS_readFileWrapper(from url: URL) throws -> FileWrapper

}
```

Details of Solution

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
    1. **JSBFSDirectory**
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. **JSBFSObservedDirectory**
5. Support App Extensions (Interprocess safety)
    1. **NSFileCoordinator Categories**
6. Speed

Details of Solution

1. Complex Models (Containing simple as well as binary data)
2. Queries for sorting and filtering data
    1. **JSBFSDirectory**
3. Ability to Sync User Data With a Server
4. Data Change API to Enable:
    1. **JSBFSObservedDirectory**
5. Support App Extensions (Interprocess safety)
    1. **NSFileCoordinator Categories**
6. Speed

Usage

Usage

Create a Data Source

```
class ListTableViewController: UITableViewController {

    private let directory: JSBFSObservedDirectory =
        try! JSBFSObservedDirectory(base: .documentDirectory,
                                    appendingPathComponent: "MyFiles",
                                    createIfNeeded: true,
                                    sortedBy: .modificationNewestFirst,
                                    filteredBy: nil,
                                    changeKind: .modificationsAsInsertionsDeletions)

}
```

Usage

Implement UITableViewDataSource

```
class ListTableViewController: UITableViewController {

    override func tableView(_ tableView: UITableView, numberOfRowsInSection s
                                                                        Int {
        do {
            return try self.directory.contentsCount()
        } catch {
            NSLog(String(describing: error))
            return 0
        }
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let id = "MyCell"
        let cell = tableView.dequeueReusableCell(withIdentifier: id) ??
            UITableViewCell(style: UITableViewCell.CellStyle.subtitle, reuseIdentifier: id)
        cell.textLabel?.text = ""
        cell.detailTextLabel?.text = ""
        do {
            let url = try self.directory.url(at: indexPath.row)
            let data = try NSFileCoordinator.JSBFS_readData(from: url, filePresenter: nil)
            let fileContents = String(data: data, encoding: .utf8) ?? "Invalid Data"
            cell.textLabel?.text = fileContents
            cell.detailTextLabel?.text = url.lastPathComponent
        } catch {
            NSLog(String(describing:error))
        }
        return cell
    }
}
```

Usage

✨ Do the Magic ✨

```
class ListTableViewController: UITableViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        self.directory.changesObserved                    self] changes in
            self.tableView.beginUpdates()
            let ins = changes.insertions.map({ IndexPath(item: $0, section: 0) })
            self.tableView.insertRows(at: ins, with: .right)
            changes.moves.forEach() { move in
                let from = IndexPath(item: move.from, section: 0)
                let to = IndexPath(item: move.to, section: 0)
                self.tableView.moveRow(at: from, to: to)
            }
            let dels = changes.deletions.map({ IndexPath(item: $0, section: 0) })
            self.tableView.deleteRows(at: dels, with: .left)
            self.tableView.endUpdates()
        }
    }

}
```

More Complex Demo

Hipstapaper File Edit View Window Help

Search

Reading List
Unread Items
All Items
Tags
Tag Number 4 10
Tag Number 0 11
Tag Number 1 8
Tag Number 2 11
Tag Number 3 11
aaaTag 2

Recently Modified on Top

Unarchived

Fake Data 2
2019/01/14 16:56

Google: The Epic Search for the number: 0
2019/01/14 16:55

Google: The Epic Search for the number: 3
2019/01/14 16:26

Google: The Epic Search for the number: 4
2019/01/14 16:26

Google: The Epic Search for the number: 5
2019/01/14 16:26

Google: The Epic Search for the number: 6
2019/01/14 16:26

Google: The Epic Search for the number: 7
2019/01/14 16:26

Google: The Epic Search for the number: 8
2019/01/14 16:26

Google: The Epic Search for the number: 9
2019/01/14 16:26

Google: The Epic Search for the number: 10
2019/01/14 16:26

Google: The Epic Search for the number: 11
2019/01/14 16:26

Google: The Epic Search for the number: 12
2019/01/14 16:26

HPData

Search

Name	Date Modified
7F04827B-83FD-4C31...243E4851D0.hpwebsite	Today 20:46
3A955270-C966-4731...8D699114CB.hpwebsite	January 14, 2019 16:56
79119E55-C2F1-48BD-...729C8EE789.hpwebsite	January 14, 2019 16:55
656E19AD-F880-4307...437F9B030D.hpwebsite	January 14, 2019 16:26
04F0261B-65E0-441D-...BACF6C67E.hpwebsite	January 14, 2019 16:26
68D0DDA4-719E-4B28...9A38039879.hpwebsite	January 14, 2019 16:26
F175AD3E-4798-429C...DA3C32EF75.hpwebsite	January 14, 2019 16:26
E20E2361-A5E8-4473-...28C7A4BA5.hpwebsite	January 14, 2019 16:26
FCFAA68E-EE89-4937...F5389B03A7.hpwebsite	January 14, 2019 16:26
4539CA20-9EA1-4B29...E46820DC59.hpwebsite	January 14, 2019 16:26
2E5A4568-C341-4C0B...49B08E3E9A.hpwebsite	January 14, 2019 16:26
2533A1CB-0A3F-4682...55083F4C49.hpwebsite	January 14, 2019 16:26
757F5BFA-E792-4658-...A22542EDD.hpwebsite	January 14, 2019 16:26
F33268E9-9996-452F...CA1265F57A.hpwebsite	January 14, 2019 16:26
6EDCEC96-FCBB-4E11...5C24E19C6A.hpwebsite	January 14, 2019 16:26
8A517C23-BAE3-4365...6E9771A0A8.hpwebsite	January 14, 2019 16:26
43D1A826-96B7-425C...ED6F7A6AB0.hpwebsite	January 14, 2019 16:26
38E86723-19B8-48F9-...2D0B36DF3.hpwebsite	January 14, 2019 16:26
6EB60FE9-C16B-4D9B...E63C489107.hpwebsite	January 14, 2019 16:26
F9A30108-1CBD-46D3...7554EBB32F.hpwebsite	January 14, 2019 16:26
BC183F50-E332-4662...4B0548C95D.hpwebsite	January 14, 2019 16:26
B6552AF2-7E4E-43DB...06FBC8BFE0.hpwebsite	January 14, 2019 16:26
1221A8CD-A480-476A...D5CB6CC05.hpwebsite	January 14, 2019 16:26
14BBE9B2-FFAE-4B28-...C7BB0DEFD.hpwebsite	January 14, 2019 16:26
339F5061-BF98-4E6C...E03786DFCE.hpwebsite	January 14, 2019 16:26
1AD12307-AAAF-4FF6-...4D0D1EF9E5.hpwebsite	January 14, 2019 16:26
AB02468C-CD07-4B78...74EBE47102.hpwebsite	January 14, 2019 16:26
7F0EA34E-4E68-43E5...ED82752E26.hpwebsite	January 14, 2019 16:26
6B765396-A551-44F2-...710C2AC7D.hpwebsite	January 14, 2019 16:26
535FCDFF-2C4B-4FE8...B15746156A.hpwebsite	January 14, 2019 16:26
934ED4C6-6F9B-4923...B2D60B16E.hpwebsite	January 14, 2019 16:26
8ECEB7F0-6068-452C...EA70491B50.hpwebsite	January 14, 2019 16:26

Outline

- State of Persistence in iOS and macOS
- Proposed Persistence Idea
- Review Foundation API
- Details of Solution
- Q&A
- Bonus: Syncing (time permitting)

Outline

- State of Persistence in iOS and macOS
- Proposed Persistence Idea
- Review Foundation API
- Details of Solution
- Q&A
- Bonus: Syncing (time permitting)

Q&A

Jeffrey Bergier

github.com/
jeffreybergier/
JSBFilesystem/ on Github

@jeffburg on Twitter
tokyojeff.com on Web

jeffburgdemo / HPData

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Demo Data

Edit

Manage topics

43 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

94497F6A-9D8E-5E19-B5CA-CD5F24810718 and jeffreybergier <URLItem> Unarchived Latest commit 02cb5b0 21 seconds ago

04F0261B-65E0-441D-9DDA-EB5BACF6C67E.hpwebsite	<URLItem> Tag Assigned	4 days ago
1221A8CD-A480-476A-A3DC-0BBD5CB6CC05.hpwebsite	<URLItem> Tag Assigned	4 days ago
14BBE9B2-FFAE-4B28-9806-946C7BB0DEFD.hpwebsite	<URLItem> Tag Assigned	4 days ago
1AD12307-AAAF-4FF6-838B-7E4D0D1EF9E5.hpwebsite	<URLItem> Tag Assigned	4 days ago
2533A1CB-0A3F-4682-AFEB-1955083F4C49.hpwebsite	<URLItem> Tag Assigned	4 days ago
2E5A4568-C341-4C0B-8BB8-5149B08E3E9A.hpwebsite	<URLItem> Tag Assigned	4 days ago
339F5061-BF98-4E6C-A5FD-00E03786DFCE.hpwebsite	<URLItem> Tag Assigned	4 days ago
38E86723-19B8-48F9-9C93-AC12D0B36DF3.hpwebsite	<URLItem> Tag Assigned	4 days ago