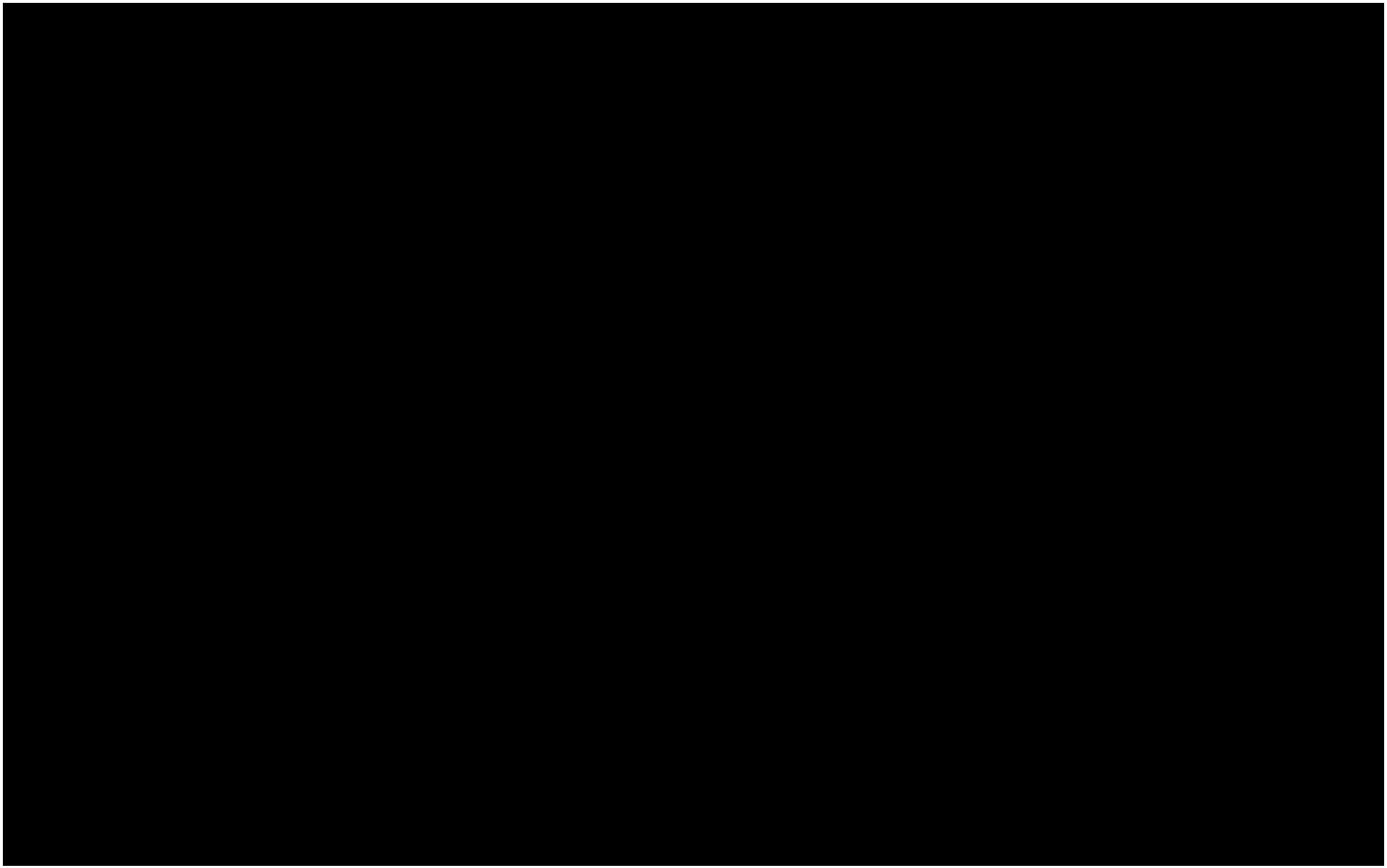
iCloud Storage Overview

Session 209

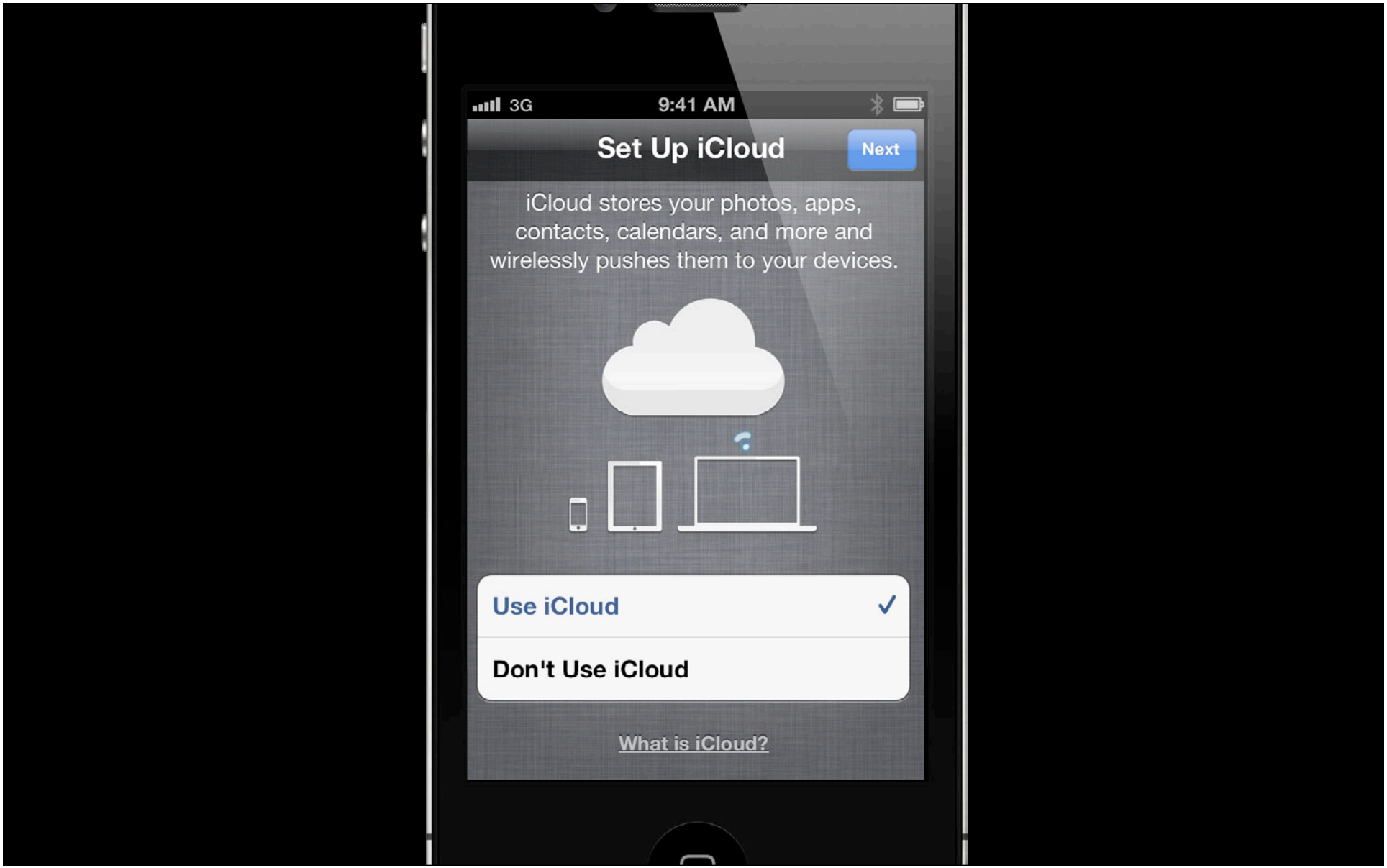
Eric Krugler
iCloud Engineering

iCloud



125 Million

iCloud Accounts


3G
9:41 AM
Set Up iCloud
Next
iCloud stores your photos, apps, contacts, calendars, and more and wirelessly pushes them to your devices.
Use iCloud
Don't Use iCloud
What is iCloud?


9:42 AM
Set Up iCloud
iCloud stores your content in the cloud and pushes it to your devices. Learn more.
Set up iCloud on this Mac
Back
Continue
MacBook Air

iCloud Storage API

Key Value Storage

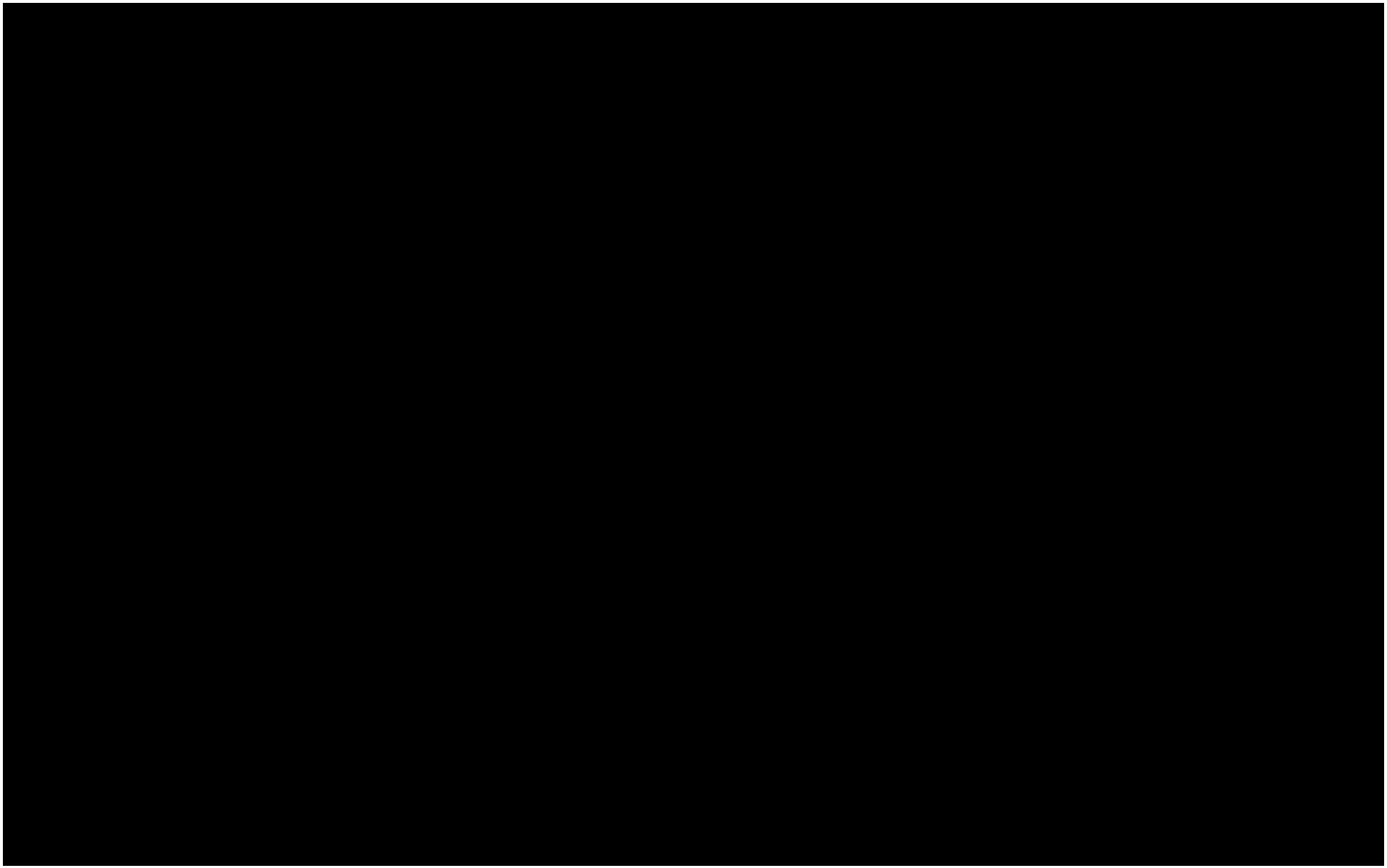
Key Value Storage

Document Storage

Key Value Storage

Document Storage

Core Data Storage




SoundHound
PRO
jamie
oliver
HD
NBA 2K12
PRO


N
W
E
S
Q4 2011
73°
Dictionary / Thesaurus
Aa

iCloud is integrated into everything we do

iCloud Checklist

iCloud Checklist

- ✓ Accounts Setup

iCloud Checklist

- ✓ Accounts Setup
- ✓ Client APIs

iCloud Checklist

- ✓ Accounts Setup
- ✓ Client APIs
- ✓ Server Code

iCloud Checklist

- Accounts Setup
- Client APIs
- Server Code
- Operations

Stress Free iCloud Storage

Stress Free iCloud Storage

iCloud Storage API

iCloud Storage API

Dallas De Atley
Manager, Platform Services

Using iCloud Storage API

What you'll learn

Using iCloud Storage API

What you'll learn

- How it works

Using iCloud Storage API

What you'll learn

- How it works
- How to use it

Using iCloud Storage API

What you'll learn

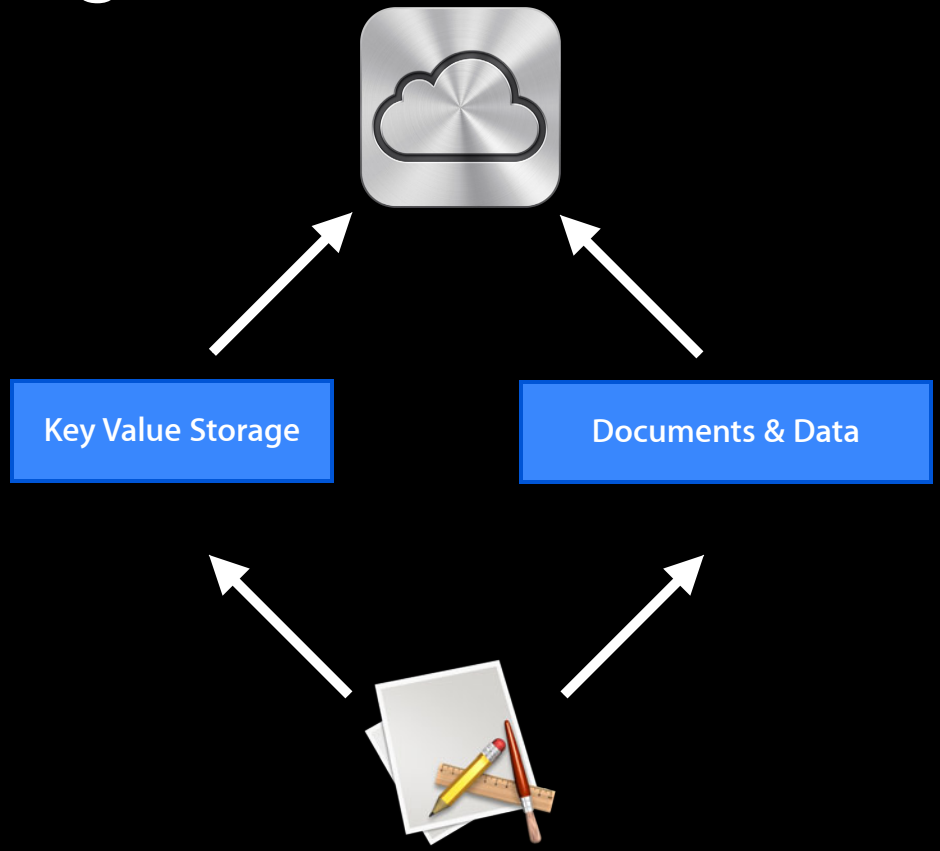
- How it works
- How to use it
- Best practices

Using iCloud Storage API

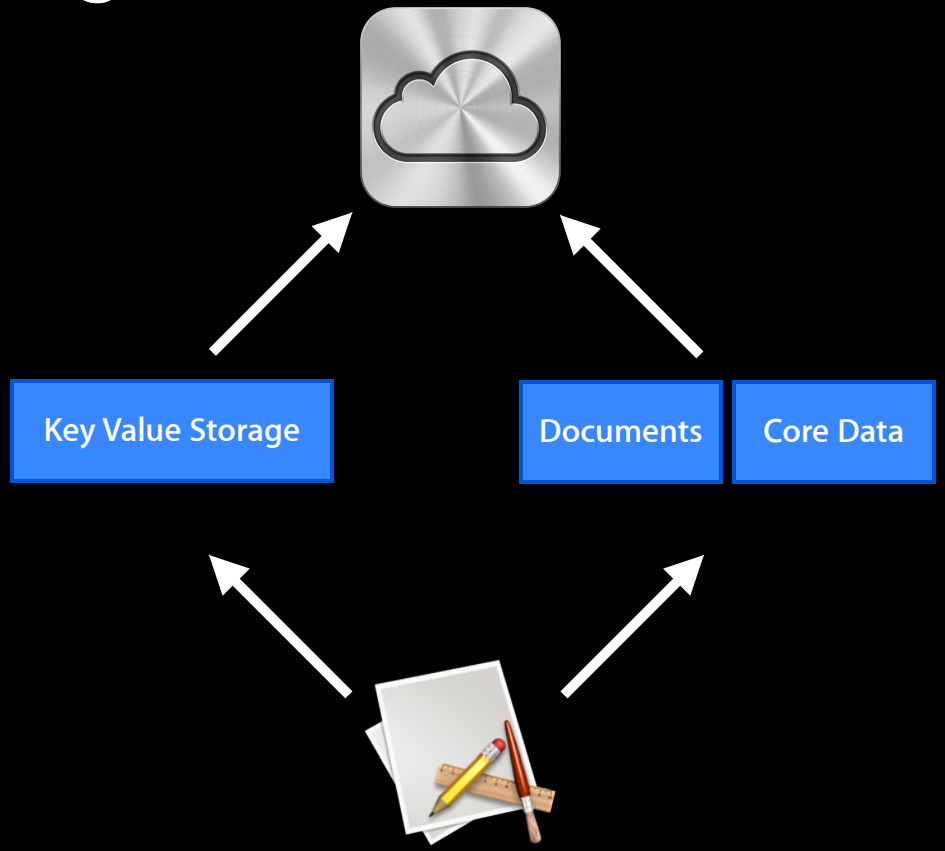
What you'll learn

- How it works
- How to use it
- Best practices
- Debugging tools

iCloud Storage



iCloud Storage



iCloud Storage

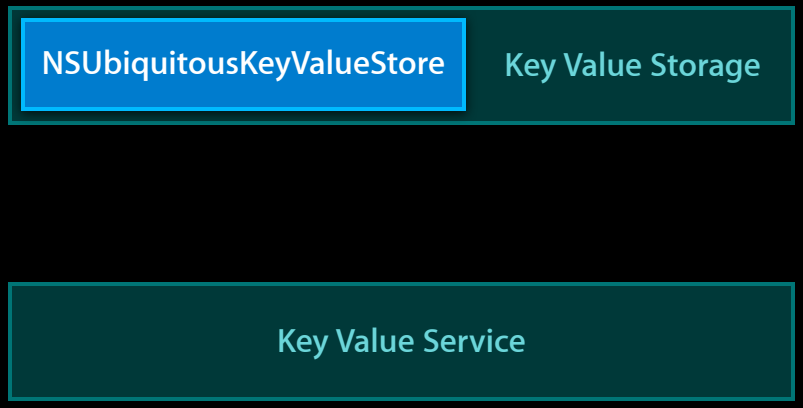
Key Value Storage

iCloud Storage

Key Value Storage

iCloud Storage

Key Value Storage

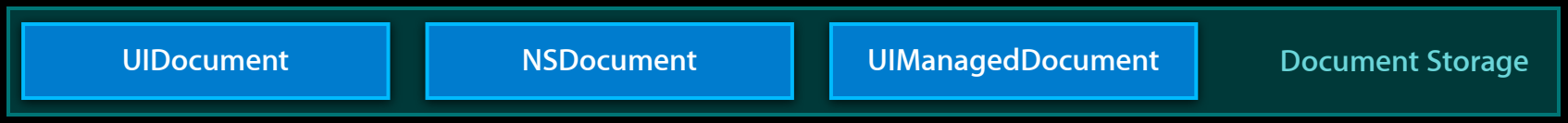


iCloud Storage

Document Storage

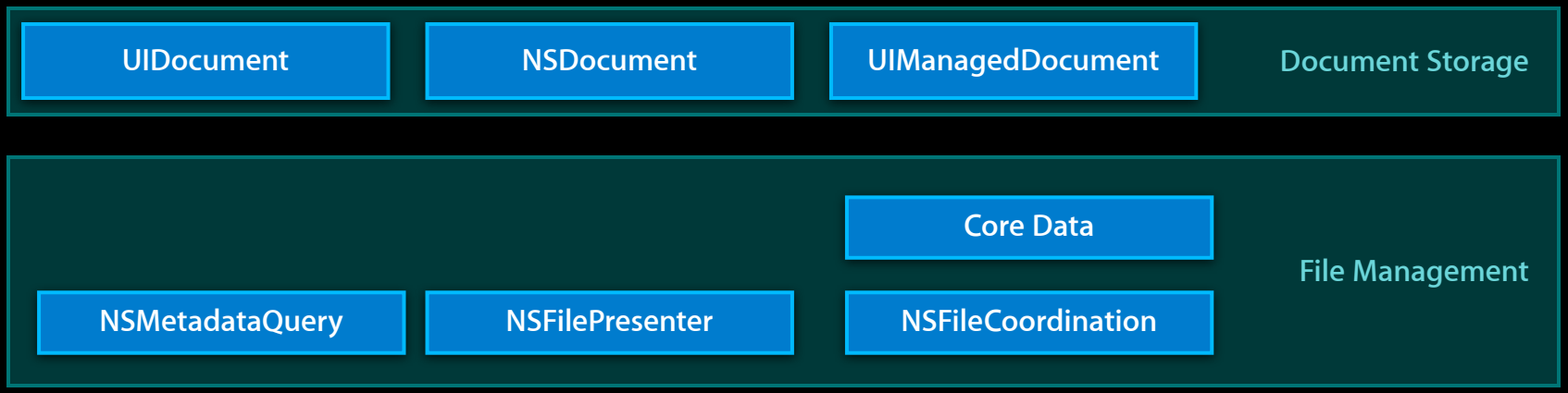
iCloud Storage

Document Storage



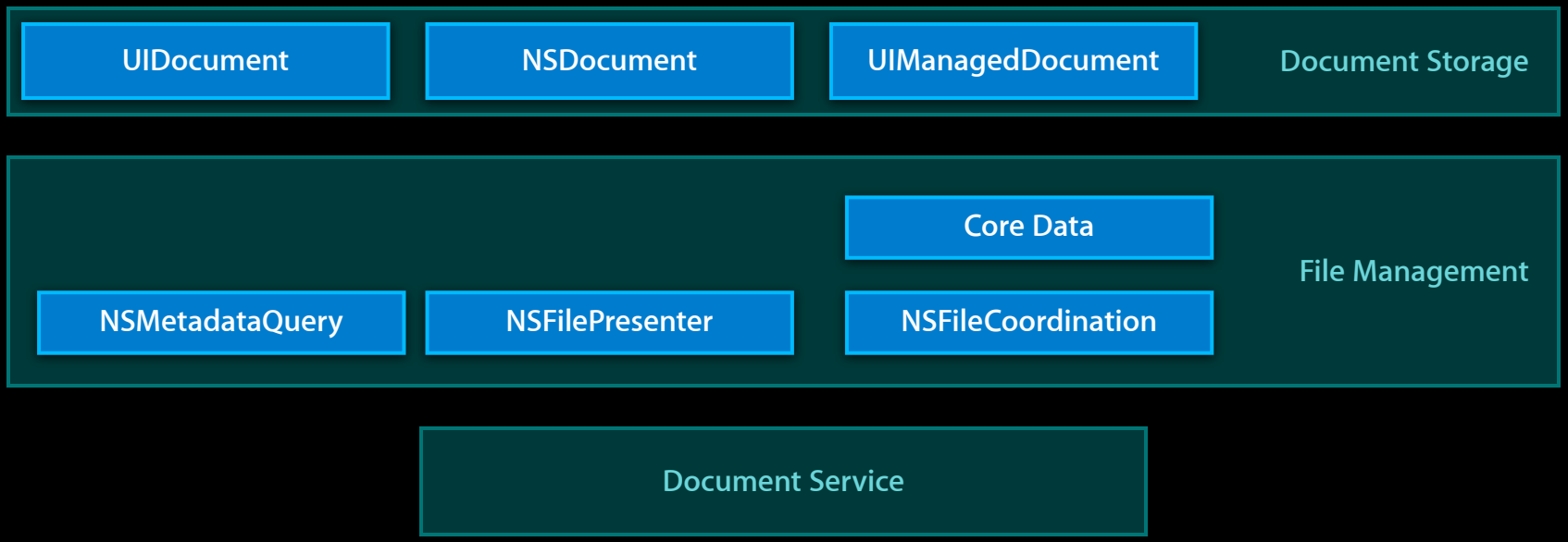
iCloud Storage

Document Storage



iCloud Storage

Document Storage



Enabling iCloud in Your Project

Enabling iCloud in Your Project

- App Store only

Enabling iCloud in Your Project

- App Store only
- Provision your iOS devices

Enabling iCloud in Your Project

- App Store only
- Provision your iOS devices
- Set the entitlements in your project

Enabling iCloud in Your Project

Provisioning your iOS devices

Enabling iCloud in Your Project

Provisioning your iOS devices

- What is provisioning?

Enabling iCloud in Your Project

Provisioning your iOS devices

- What is provisioning?
- Register your iOS device

Enabling iCloud in Your Project

Provisioning your iOS devices

- What is provisioning?
- Register your iOS device
  - Developer Portal

Enabling iCloud in Your Project

Provisioning your iOS devices

- What is provisioning?
- Register your iOS device
  - Developer Portal
  - Xcode Organizer

Enabling iCloud in Your Project

Provisioning your iOS devices

- What is provisioning?
- Register your iOS device
  - Developer Portal
  - Xcode Organizer
- Provisioning Profile

Enabling iCloud in Your Project

Provisioning your iOS devices

- What is provisioning?
- Register your iOS device
  - Developer Portal
  - Xcode Organizer
- Provisioning Profile
  - Defines your devices

Enabling iCloud in Your Project

Provisioning your iOS devices

- What is provisioning?
- Register your iOS device
  - Developer Portal
  - Xcode Organizer
- Provisioning Profile
  - Defines your devices
  - Defines your developer certificate

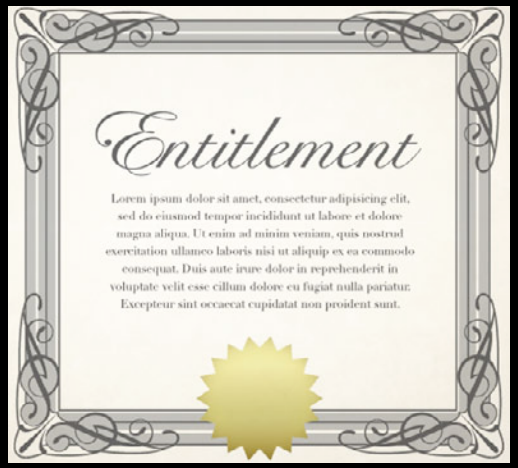
Enabling iCloud in Your Project

Provisioning your iOS devices

- What is provisioning?
- Register your iOS device
  - Developer Portal
  - Xcode Organizer
- Provisioning Profile
  - Defines your devices
  - Defines your developer certificate
  - Grants iCloud support

Enabling iCloud in Your Project

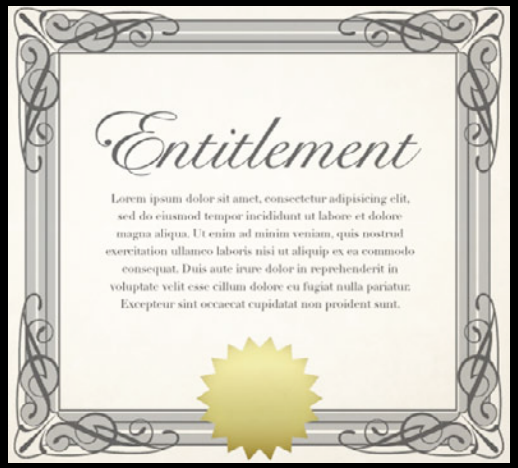
Entitlements



Enabling iCloud in Your Project

Entitlements

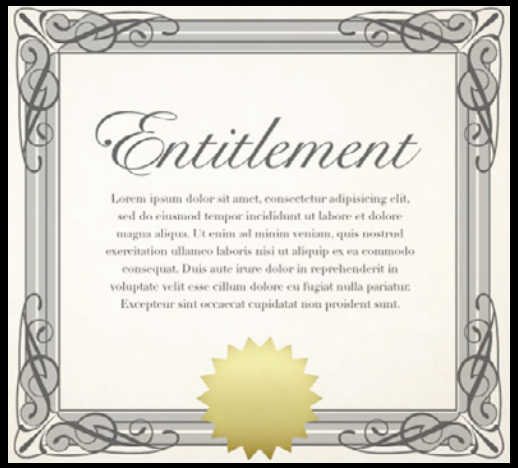
- What is an entitlement?



Enabling iCloud in Your Project

Entitlements

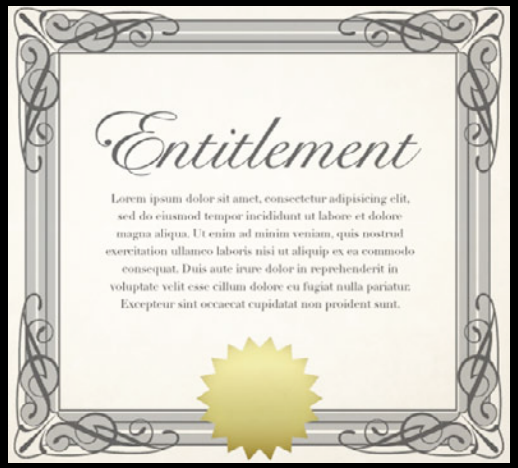
- What is an entitlement?
  - Simple access control



Enabling iCloud in Your Project

Entitlements

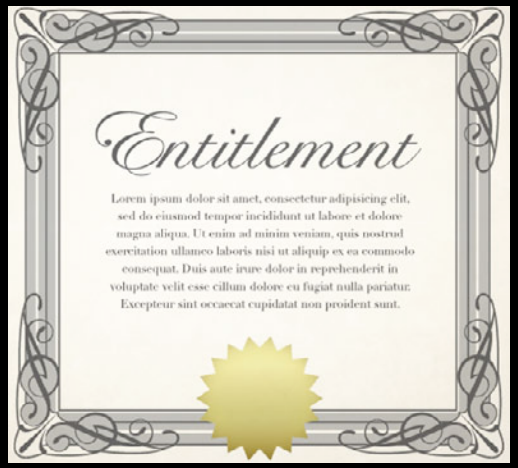
- What is an entitlement?
  - Simple access control
  - Included in the code signature



Enabling iCloud in Your Project

Entitlements

- What is an entitlement?
  - Simple access control
  - Included in the code signature
- Set the entitlements in your project

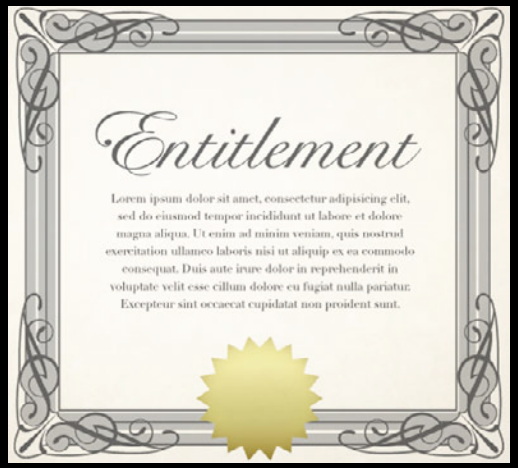


Enabling iCloud in Your Project

Entitlements

- What is an entitlement?
  - Simple access control
  - Included in the code signature
- Set the entitlements in your project

```
com.apple.developer.ubiquity-
kvstore-identifier
```



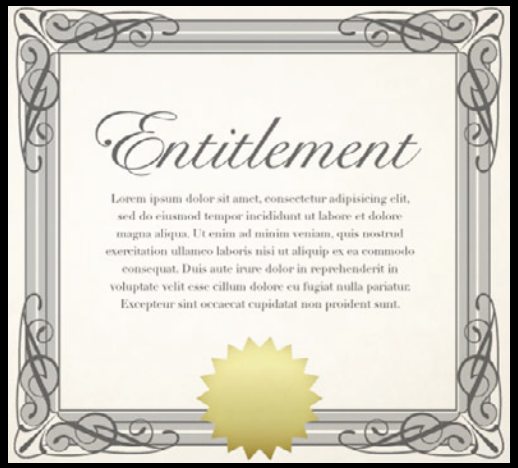
Enabling iCloud in Your Project

Entitlements

- What is an entitlement?
  - Simple access control
  - Included in the code signature
- Set the entitlements in your project

```
com.apple.developer.ubiquity-
kvstore-identifier

com.apple.developer.ubiquity-
container-identifiers
```



Key Value Storage

Storing simple data

Key Value Storage

Storing simple data

`NSUbiquitousKeyValueStore`

Key Value Storage

Storing simple data

`NSUbiquitousKeyValueStore`

- Store simple plist values

Key Value Storage

Storing simple data

`NSUbiquitousKeyValueStore`

- Store simple plist values
- Simple conflict resolution

Key Value Storage

Storing simple data

`NSUbiquitousKeyValueStore`

- Store simple plist values
- Simple conflict resolution
- Usable without iCloud account

Key Value Storage

Storing simple data

`NSUbiquitousKeyValueStore`

- Store simple plist values
- Simple conflict resolution
- Usable without iCloud account
- Key value service improvements

Key Value Storage

Storing simple data

`NSUbiquitousKeyValueStore`

- Store simple plist values
- Simple conflict resolution
- Usable without iCloud account
- Key value service improvements
  - Increased capacity

Key Value Storage

Storing simple data

`NSUbiquitousKeyValueStore`

- Store simple plist values
- Simple conflict resolution
- Usable without iCloud account
- Key value service improvements
  - Increased capacity
    - Maximum of 1024 keys

Key Value Storage

Storing simple data

`NSUbiquitousKeyValueStore`

- Store simple plist values
- Simple conflict resolution
- Usable without iCloud account
- Key value service improvements
  - Increased capacity
    - Maximum of 1024 keys
    - 1 MB per application

Key Value Storage

Storing simple data

`NSUbiquitousKeyValueStore`

- Store simple plist values
- Simple conflict resolution
- Usable without iCloud account
- Key value service improvements
  - Increased capacity
    - Maximum of 1024 keys
    - 1 MB per application
  - Greater responsiveness

Key Value Storage

Storing simple data

`NSUbiquitousKeyValueStore`

- Store simple plist values
- Simple conflict resolution
- Usable without iCloud account
- Key value service improvements
  - Increased capacity
    - Maximum of 1024 keys
    - 1 MB per application
  - Greater responsiveness
    - 15 requests every 90 seconds

Observing Changes

```
// get application's default store
kvStore = [NSUbiquitousKeyValueStore defaultStore];

// observe changes
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(kvStoreDidChange:)
name:NSUbiquitousKeyValueStoreDidChangeExternallyNotification object:nil];

// get any change since last launch
[kvStore synchronize];
```

Observing Changes

```
// get application's default store
kvStore = [NSUbiquitousKeyValueStore defaultStore];

// observe changes
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(kvStoreDidChange:)
name:NSUbiquitousKeyValueStoreDidChangeExternallyNotification object:nil];

// get any change since last launch
[kvStore synchronize];
```

Observing Changes

```
// get application's default store
kvStore = [NSUbiquitousKeyValueStore defaultStore];

// observe changes
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(kvStoreDidChange:)
name:NSUbiquitousKeyValueStoreDidChangeExternallyNotification object:nil];

// get any change since last launch
[kvStore synchronize];
```

Observing Changes

```
// get application's default store
kvStore = [NSUbiquitousKeyValueStore defaultStore];

// observe changes
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(kvStoreDidChange:)
name:NSUbiquitousKeyValueStoreDidChangeExternallyNotification object:nil];

// get any change since last launch
[kvStore synchronize];
```

Making Changes

```
// store values
[kvStore setObject:someObject forKey:@"someKey"];
[kvStore setBool:YES forKey:@"someOtherKey"];

// store values locally

[kvStore synchronize];
```

Making Changes

```
// store values
[kvStore setObject:someObject forKey:@"someKey"];
[kvStore setBool:YES forKey:@"someOtherKey"];

// store values locally

[kvStore synchronize];
```

Making Changes

```
// store values
[kvStore setObject:someObject forKey:@"someKey"];
[kvStore setBool:YES forKey:@"someOtherKey"];

// store values locally

[kvStore synchronize];
```

Making Changes

```
// store values
[kvStore setObject:someObject forKey:@"someKey"];
[kvStore setBool:YES forKey:@"someOtherKey"];

// store values locally

[kvStore synchronize];
```

Getting the Notification

```
- (void)kvStoreDidChange:(NSNotification *)notification
{
    NSDictionary* userInfo = [notification userInfo];
    // get change reason (initial download, external change or quota
violation change)
    int reason = [[userInfo
objectForKey:NSUbiquitousKeyValueStoreChangeReasonKey] intValue];
    // get the affected keys
    NSArray* changedKeys = [userInfo
objectForKey:NSUbiquitousKeyValueStoreChangedKeysKey];
    // store the values locally
}
```

Getting the Notification

```
- (void)kvStoreDidChange:(NSNotification *)notification
{
    NSDictionary* userInfo = [notification userInfo];
    // get change reason (initial download, external change or quota
violation change)
    int reason = [[userInfo
objectForKey:NSUbiquitousKeyValueStoreChangeReasonKey] intValue];
    // get the affected keys
    NSArray* changedKeys = [userInfo
objectForKey:NSUbiquitousKeyValueStoreChangedKeysKey];
    // store the values locally
}
```

Getting the Notification

```
- (void)kvStoreDidChange:(NSNotification *)notification
{
    NSDictionary* userInfo = [notification userInfo];
    // get change reason (initial download, external change or quota
violation change)
    int reason = [[userInfo
objectForKey:NSUbiquitousKeyValueStoreChangeReasonKey] intValue];
    // get the affected keys
    NSArray* changedKeys = [userInfo
objectForKey:NSUbiquitousKeyValueStoreChangedKeysKey];
    // store the values locally
}
```

Best Practices

Key Value Storage

Best Practices

Key Value Storage

- Take advantage of KVS!

Best Practices

Key Value Storage

- Take advantage of KVS!
  - Configuration

Best Practices

Key Value Storage

- Take advantage of KVS!
  - Configuration
  - State

Best Practices

Key Value Storage

- Take advantage of KVS!
  - Configuration
  - State
  - Consistent experience across devices

Best Practices

Key Value Storage

- Take advantage of KVS!
  - Configuration
  - State
  - Consistent experience across devices
- Don’t store passwords

Best Practices

Key Value Storage

- Take advantage of KVS!
  - Configuration
  - State
  - Consistent experience across devices
- Don’t store passwords
  - Use SecItem to place in the keychain

Best Practices

Key Value Storage

- Take advantage of KVS!
  - Configuration
  - State
  - Consistent experience across devices
- Don’t store passwords
  - Use SecItem to place in the keychain
- Keep a local cache

Best Practices

Key Value Storage

- Take advantage of KVS!
  - Configuration
  - State
  - Consistent experience across devices
- Don’t store passwords
  - Use SecItem to place in the keychain
- Keep a local cache
- Watch for quota violations

Document Storage

How it works

Document Storage

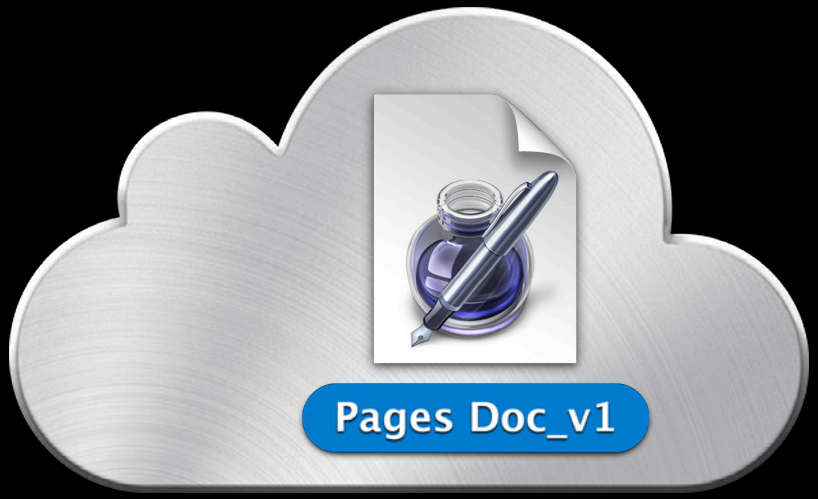
How it works

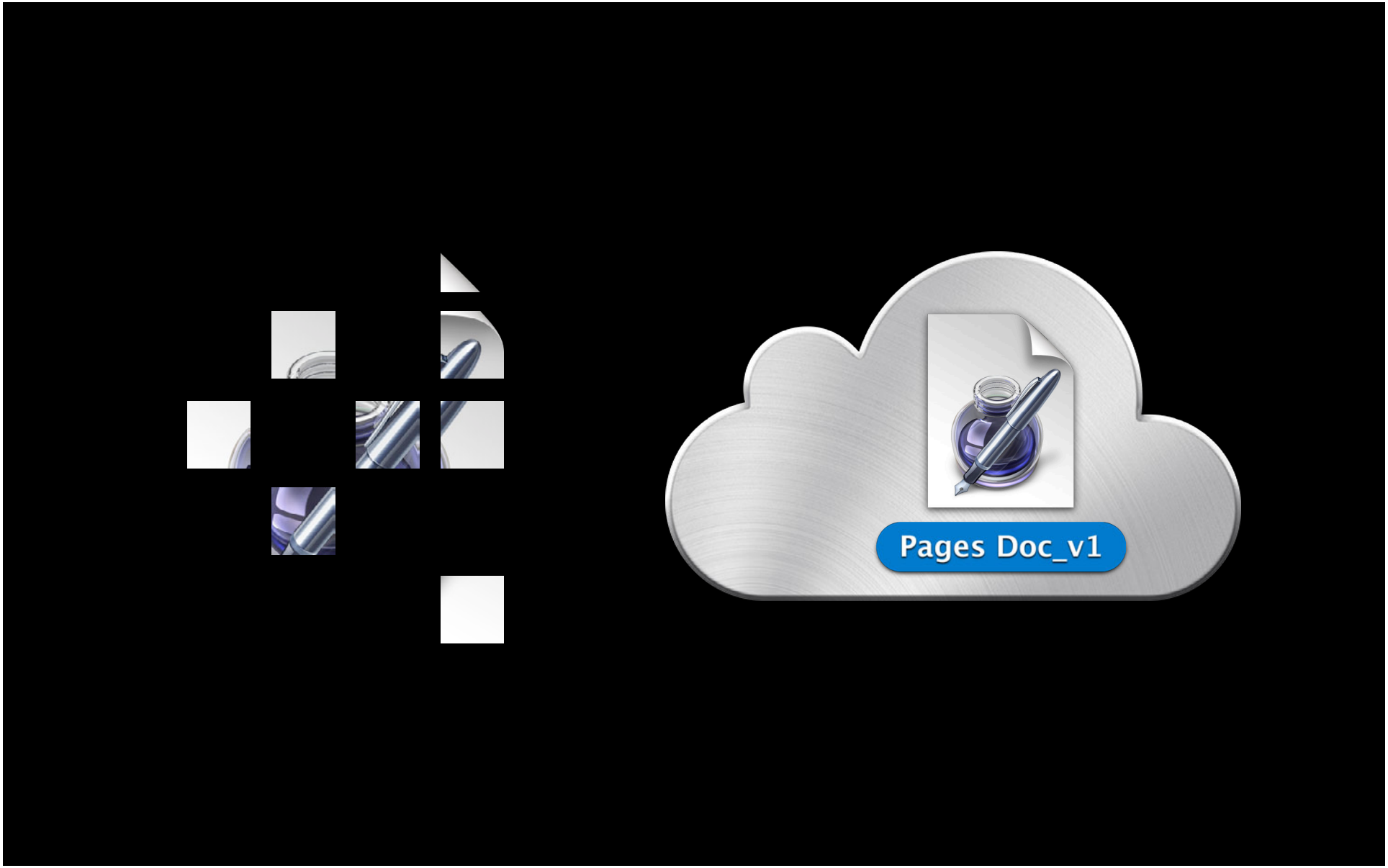
- Application container

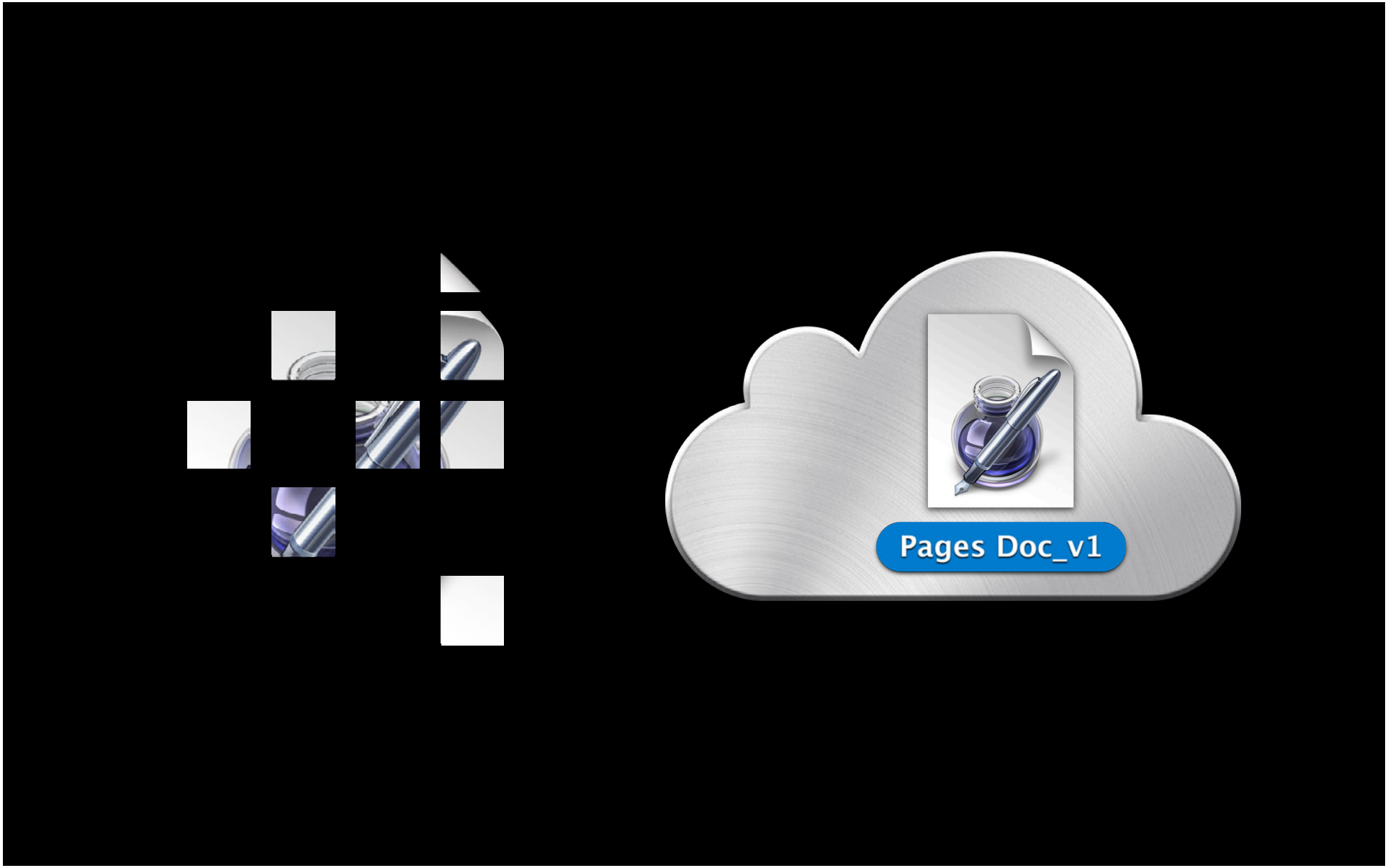
Document Storage

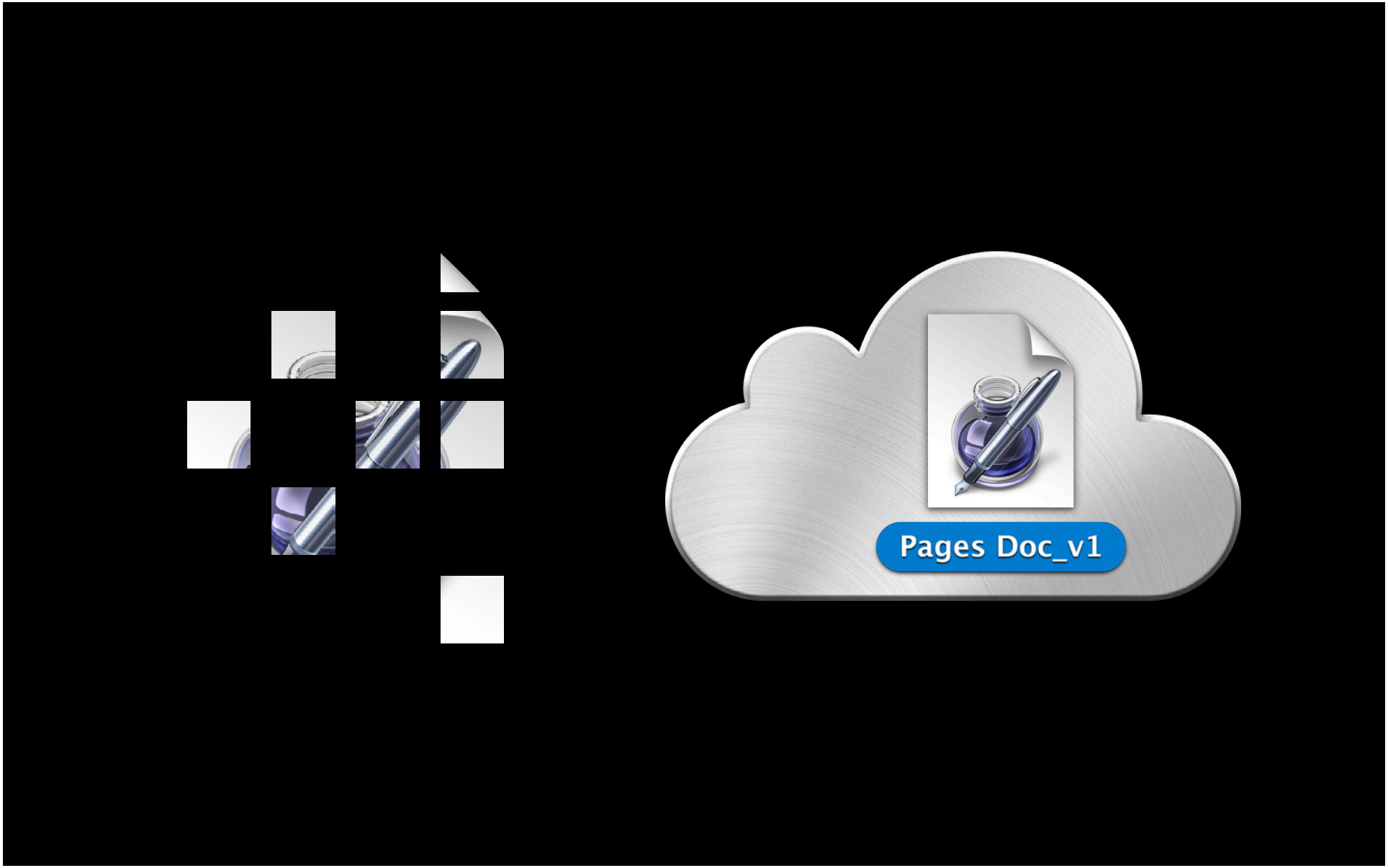
How it works

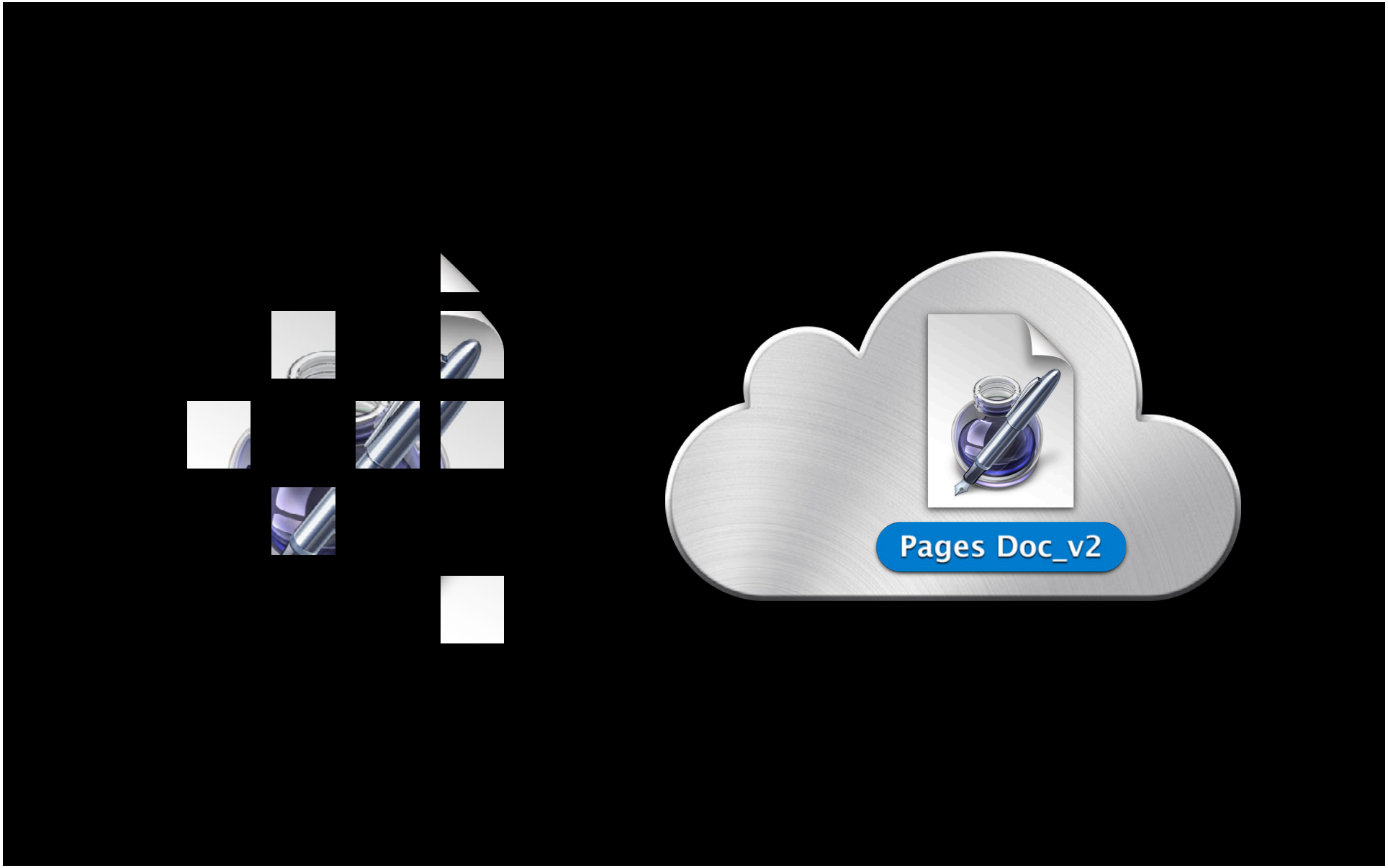
- Application container
- Ubiquity container


Pages Doc_v1


Pages Doc_v1


Pages Doc_v1


Pages Doc_v1


Pages Doc_v2

Document Storage

How it works

Document Storage

How it works

- Metadata is always pushed

Document Storage

How it works

- Metadata is always pushed
- Device pulls when appropriate

Document Storage

How it works

- Metadata is always pushed
- Device pulls when appropriate
  - OS X always downloads

Document Storage

How it works

- Metadata is always pushed
- Device pulls when appropriate
  - OS X always downloads
  - iOS requires app to request

Document Storage

How it works

- Metadata is always pushed
- Device pulls when appropriate
  - OS X always downloads
  - iOS requires app to request
  - All subsequent changes are reflected

Document Storage

How it works

- Metadata is always pushed
- Device pulls when appropriate
  - OS X always downloads
  - iOS requires app to request
  - All subsequent changes are reflected
- Peer to peer when possible

Document Storage

How it works

- Metadata is always pushed
- Device pulls when appropriate
  - OS X always downloads
  - iOS requires app to request
  - All subsequent changes are reflected
- Peer to peer when possible
- Automatic conflict resolution

Document Storage

How it works

- Metadata is always pushed
- Device pulls when appropriate
  - OS X always downloads
  - iOS requires app to request
  - All subsequent changes are reflected
- Peer to peer when possible
- Automatic conflict resolution
- URL publishing

Document Storage

How it works

- Metadata is always pushed
- Device pulls when appropriate
  - OS X always downloads
  - iOS requires app to request
  - All subsequent changes are reflected
- Peer to peer when possible
- Automatic conflict resolution
- URL publishing
  - Alternative to email attachments

Detecting the iCloud Account

Detecting the iCloud Account

- Requires an iCloud account

Detecting the iCloud Account

- Requires an iCloud account
- Ubiquity Identity Token

Detecting the iCloud Account

- Requires an iCloud account
- Ubiquity Identity Token
  - Anonymous

Detecting the iCloud Account

- Requires an iCloud account
- Ubiquity Identity Token
  - Anonymous
  - Specific to your app and device

Detecting the iCloud Account

- Requires an iCloud account
- Ubiquity Identity Token
  - Anonymous
  - Specific to your app and device
- iCloud Account Notification

Detecting the iCloud Account

- Requires an iCloud account
- Ubiquity Identity Token
  - Anonymous
  - Specific to your app and device
- iCloud Account Notification
- Container URL

Detecting the iCloud Account

- Requires an iCloud account
- Ubiquity Identity Token
  - Anonymous
  - Specific to your app and device
- iCloud Account Notification
- Container URL
  - Avoid calling this on the main thread

When Your App Launches

1. Check for iCloud availability

```
// get the user's ubiquity token
id token = [[NSFileManager defaultManager] ubiquityIdentityToken];
if (token) {

    // cache the token
}
```

When Your App Launches

2. Register for the iCloud availability change notification

```
// Register for the identity changed notification
center = [NSNotificationCenter defaultCenter];
[center addObserver:self
        selector:@selector(_handleUbiquityIdentityChanged:)
        name:NSUbiquityIdentityDidChangeNotification object:nil];
```

When Your App Launches

3. Make your Ubiquity container available

```
dispatch_async (dispatch_get_global_queue (DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
// get the ubiquity container URL
     containerURL = [fileManager URLForUbiquityContainerIdentifier:nil];
});
```

Best Practices

iCloud account availability

Best Practices

iCloud account availability

- Listen for the account notification

Best Practices

iCloud account availability

- Listen for the account notification
- If the account token changes

Best Practices

iCloud account availability

- Listen for the account notification
- If the account token changes
  - Clear any caches

Best Practices

iCloud account availability

- Listen for the account notification
- If the account token changes
  - Clear any caches
  - Refresh your UI

Best Practices

iCloud account availability

- Listen for the account notification
- If the account token changes
  - Clear any caches
  - Refresh your UI
- Setup your ubiquity container on a separate thread

Document Storage

Types of documents

Document Storage

Types of documents

- File

Document Storage

Types of documents

- File
- Package

Document Storage

Types of documents

- File
- Package
- Core Data

Document Types

File

Document Types

File

- Unix file types

Document Types

File

- Unix file types
  - Regular files

Document Types

File

- Unix file types
  - Regular files
  - Symlinks

Document Types

File

- Unix file types
  - Regular files
  - Symlinks
  - Directories

Document Types

File

- Unix file types
  - Regular files
  - Symlinks
  - Directories
  - Extended attributes

Document Types

File

- Unix file types
  - Regular files
  - Symlinks
  - Directories
  - Extended attributes
- Be aware of case sensitivity

Document Types

File

- Unix file types
  - Regular files
  - Symlinks
  - Directories
  - Extended attributes
- Be aware of case sensitivity
  - Case Sensitive Filesystem

Document Types

File

- Unix file types
  - Regular files
  - Symlinks
  - Directories
  - Extended attributes
- Be aware of case sensitivity
  - Case Sensitive Filesystem
    - FOO != foo

Document Types

File

- Unix file types
  - Regular files
  - Symlinks
  - Directories
  - Extended attributes
- Be aware of case sensitivity
  - Case Sensitive Filesystem
    - FOO != foo
  - Case Insensitive Filesystem

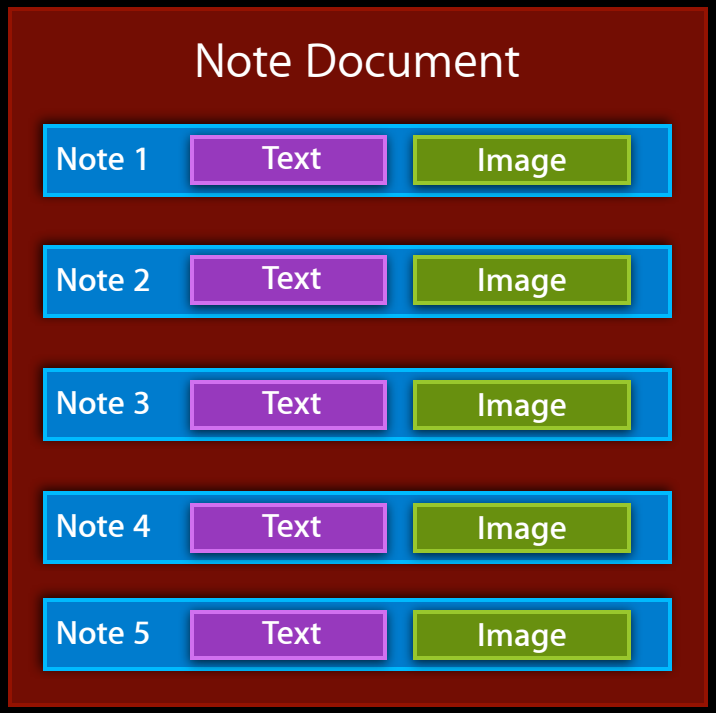
Document Types

File

- Unix file types
  - Regular files
  - Symlinks
  - Directories
  - Extended attributes
- Be aware of case sensitivity
  - Case Sensitive Filesystem
    - FOO != foo
  - Case Insensitive Filesystem
    - FOO == foo

Document Types

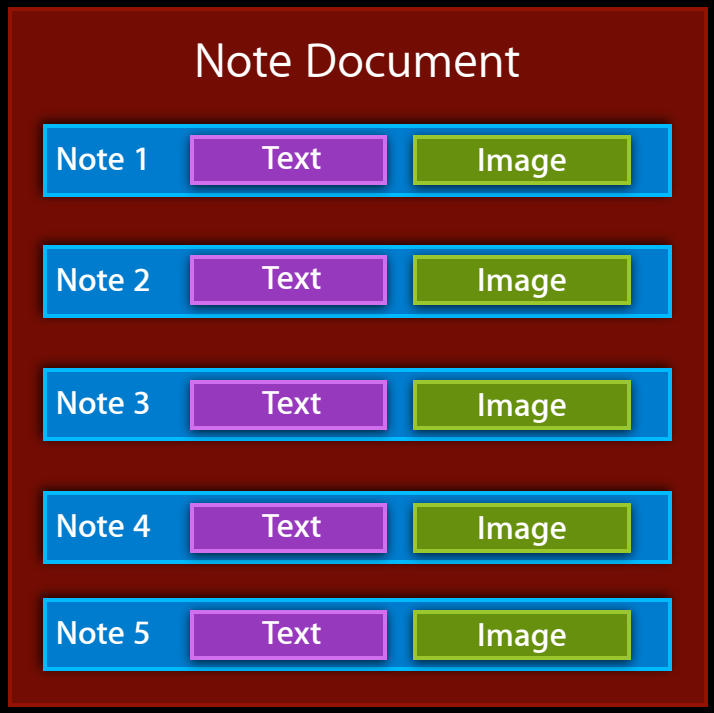
Package



Document Types

Package

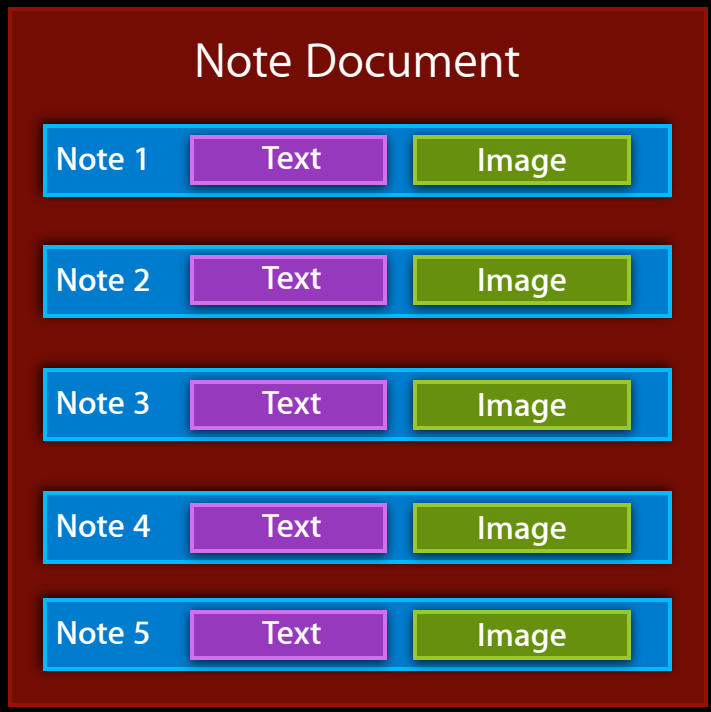
- Multiple files in a single directory



Document Types

Package

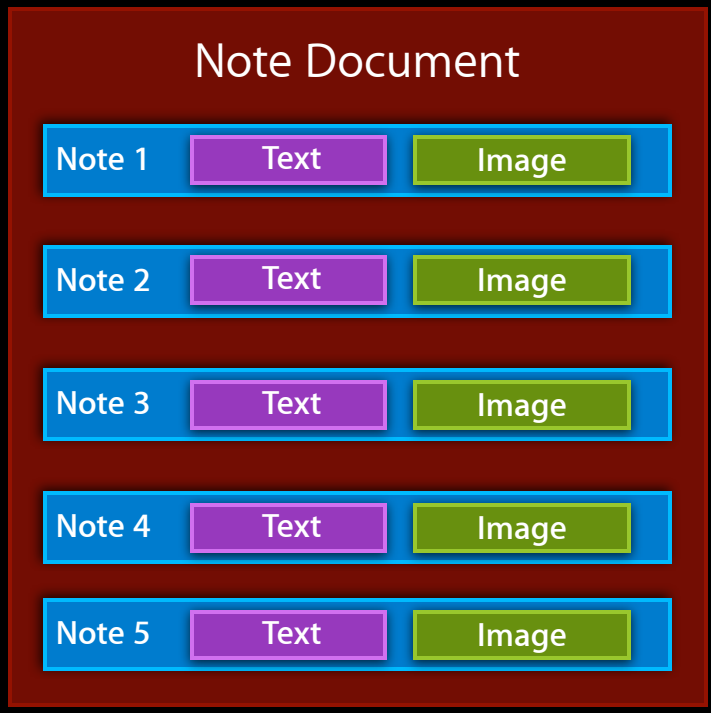
- Multiple files in a single directory
- Assets of a document broken out



Document Types

Package

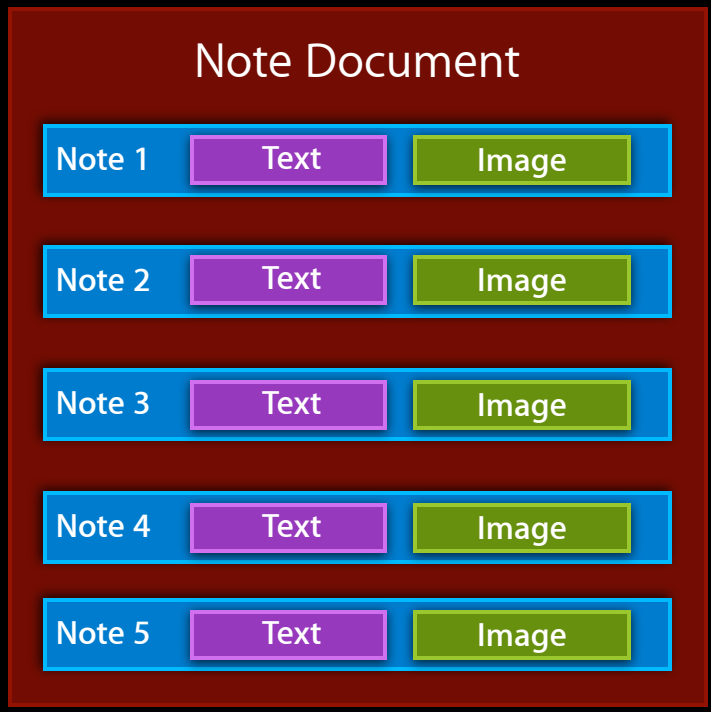
- Multiple files in a single directory
- Assets of a document broken out
  - Lots of small files



Document Types

Package

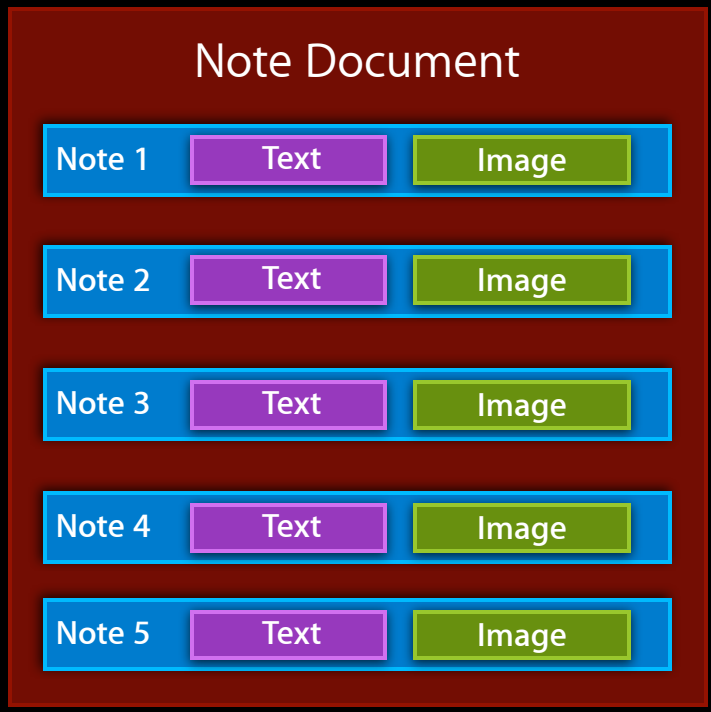
- Multiple files in a single directory
- Assets of a document broken out
  - Lots of small files
  - Fewer updates to iCloud



Document Types

Package

- Multiple files in a single directory
- Assets of a document broken out
  - Lots of small files
  - Fewer updates to iCloud
- iCloud handles updates to the package atomically



Related Sessions

Using iCloud with UIDocument	Marina Wednesday 10:15AM
Using iCloud with NSDocument	Marina Wednesday 3:15PM

Document Type

Core Data

Document Type

Core Data

- Shoebox style application

Document Type

Core Data

- Shoebox style application
  - Database

Document Type

Core Data

- Shoebox style application
  - Database
- Core Data store remains local

Document Type

Core Data

- Shoebox style application
  - Database
- Core Data store remains local
- Change logs upload to iCloud

Document Type

Core Data

- Shoebox style application
  - Database
- Core Data store remains local
- Change logs upload to iCloud
- XML and binary stores

Document Type

Core Data

- Shoebox style application
  - Database
- Core Data store remains local
- Change logs upload to iCloud
- XML and binary stores
  - Each change is a full transfer

Document Type

Core Data

- Shoebox style application
  - Database
- Core Data store remains local
- Change logs upload to iCloud
- XML and binary stores
  - Each change is a full transfer
  - Good for small data sets that don't change often

Document Type

Core Data

- Shoebox style application
  - Database
- Core Data store remains local
- Change logs upload to iCloud
- XML and binary stores
  - Each change is a full transfer
  - Good for small data sets that don't change often

`UIManagedDocument`

Document Type

Core Data

- Shoebox style application
  - Database
- Core Data store remains local
- Change logs upload to iCloud
- XML and binary stores
  - Each change is a full transfer
  - Good for small data sets that don't change often

`UIManagedDocument`

`NSPersistentDocument` doesn't support iCloud

Related Session

Using iCloud with Core Data	Marina Wednesday 4:30PM

Your Document Format

Planning for the future

Your Document Format

Planning for the future

- Design for network efficiency

Your Document Format

Planning for the future

- Design for network efficiency
- Design for cross platform

Your Document Format

Planning for the future

- Design for network efficiency
- Design for cross platform
  - Platform incompatibilities

Your Document Format

Planning for the future

- Design for network efficiency
- Design for cross platform
  - Platform incompatibilities
    - Keyed archivers

Your Document Format

Planning for the future

- Design for network efficiency
- Design for cross platform
  - Platform incompatibilities
    - Keyed archivers
  - Cloud representation

Your Document Format

Planning for the future

Your Document Format

Planning for the future

- Design for app upgrades

Your Document Format

Planning for the future

- Design for app upgrades
  - Version your document format

Your Document Format

Planning for the future

- Design for app upgrades
  - Version your document format
  - Version compatibility

Your Document Format

Planning for the future

- Design for app upgrades
  - Version your document format
  - Version compatibility
    - Read-Write

Your Document Format

Planning for the future

- Design for app upgrades
  - Version your document format
  - Version compatibility
    - Read-Write
    - Read only

Your Document Format

Planning for the future

- Design for app upgrades
  - Version your document format
  - Version compatibility
    - Read-Write
    - Read only
    - Unsupported

Your Document Format

Performance considerations

Your Document Format

Performance considerations

- Beware of sync loops

Your Document Format

Performance considerations

- Beware of sync loops
- Avoid rapid changes

Your Document Format

Performance considerations

- Beware of sync loops
- Avoid rapid changes
  - Scroll position

Your Document Format

Performance considerations

- Beware of sync loops
- Avoid rapid changes
  - Scroll position
  - Modification dates

Your Document Format

Privacy considerations

Your Document Format

Privacy considerations

- Use iCloud for user data only

Your Document Format

Privacy considerations

- Use iCloud for user data only
  - No caches

Your Document Format

Privacy considerations

- Use iCloud for user data only
  - No caches
  - No temporary files

Your Document Format

Privacy considerations

- Use iCloud for user data only
  - No caches
  - No temporary files
  - No autogenerated content

Your Document Format

Privacy considerations

- Use iCloud for user data only
  - No caches
  - No temporary files
  - No autogenerated content
- Don’t publish sensitive information

Your Document Format

Privacy considerations

- Use iCloud for user data only
  - No caches
  - No temporary files
  - No autogenerated content
- Don’t publish sensitive information
  - Undo history

iCloud Document Storage

API

iCloud Document Storage

API

- NSFileManager

iCloud Document Storage

API

- NSFileManager
  - Account token

iCloud Document Storage

API

- NSFileManager
  - Account token
  - Container URL

iCloud Document Storage

API

- NSFileManager
  - Account token
  - Container URL
  - Move, Rename, Delete

iCloud Document Storage

API

- NSFileManager
  - Account token
  - Container URL
  - Move, Rename, Delete
- NSFileCoordinator

iCloud Document Storage

API

- NSFileManager
  - Account token
  - Container URL
  - Move, Rename, Delete
- NSFileCoordinator
  - Cooperating with the system

iCloud Document Storage

API

- NSFileManager
  - Account token
  - Container URL
  - Move, Rename, Delete
- NSFileCoordinator
  - Cooperating with the system
- NSMetadataQuery

iCloud Document Storage

API

- NSFileManager
  - Account token
  - Container URL
  - Move, Rename, Delete
- NSFileCoordinator
  - Cooperating with the system
- NSMetadataQuery
  - Discovering your files

iCloud Document Storage

API

- NSFileManager
  - Account token
  - Container URL
  - Move, Rename, Delete
- NSFileCoordinator
  - Cooperating with the system
- NSMetadataQuery
  - Discovering your files
- NSDocument

iCloud Document Storage

API

- NSFileManager
  - Account token
  - Container URL
  - Move, Rename, Delete
- NSFileCoordinator
  - Cooperating with the system
- NSMetadataQuery
  - Discovering your files
- NSDocument
- UIDocument

iCloud Document Storage

API

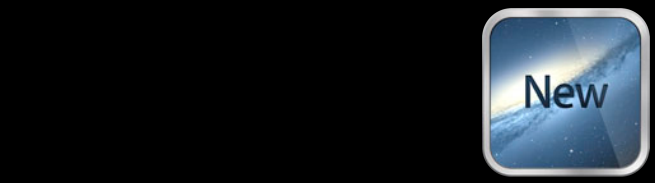
- NSFileManager
  - Account token
  - Container URL
  - Move, Rename, Delete
- NSFileCoordinator
  - Cooperating with the system
- NSMetadataQuery
  - Discovering your files
- NSDocument
- UIDocument
- UIManagedDocument

NSDocument

OS X

NSDocument

OS X



- Fully integrated with iCloud

NSDocument

OS X

- Fully integrated with iCloud
  - Enables the Ubiquity container

NSDocument

OS X

- Fully integrated with iCloud
  - Enables the Ubiquity container
  - Coordinates with the OS

NSDocument

OS X

- Fully integrated with iCloud
  - Enables the Ubiquity container
  - Coordinates with the OS
  - Tracks files

NSDocument

OS X

- Fully integrated with iCloud
  - Enables the Ubiquity container
  - Coordinates with the OS
  - Tracks files
  - Versions

NSDocument

OS X

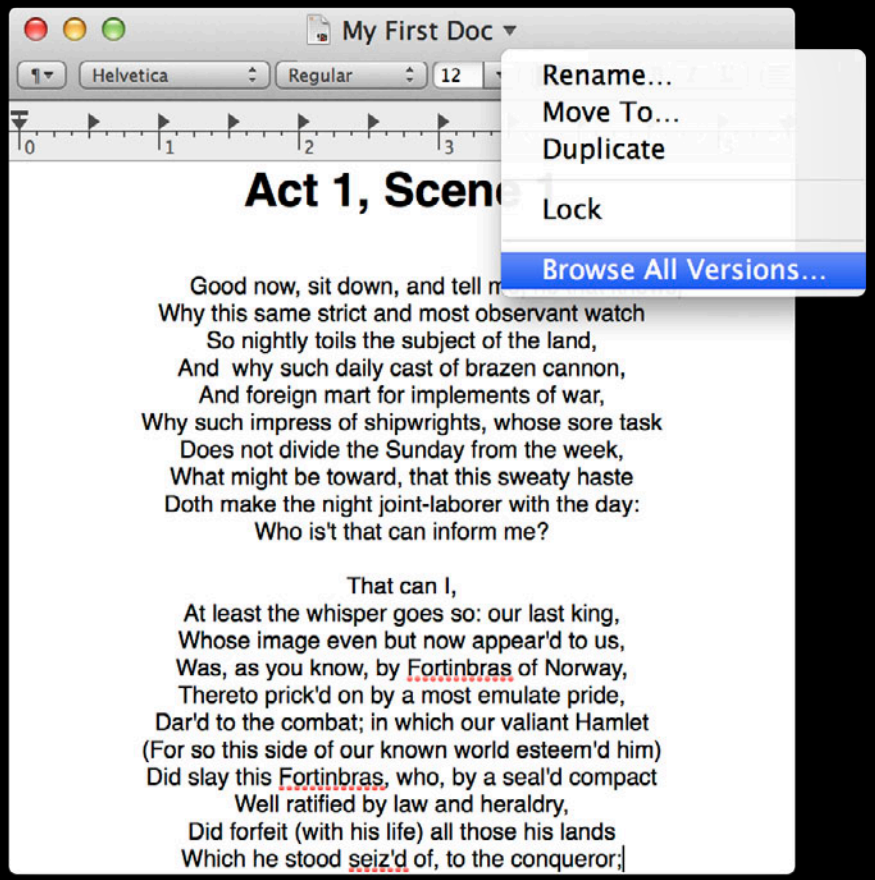
- Fully integrated with iCloud
  - Enables the Ubiquity container
  - Coordinates with the OS
  - Tracks files
  - Versions
  - Conflicts

NSDocument

Open Panel

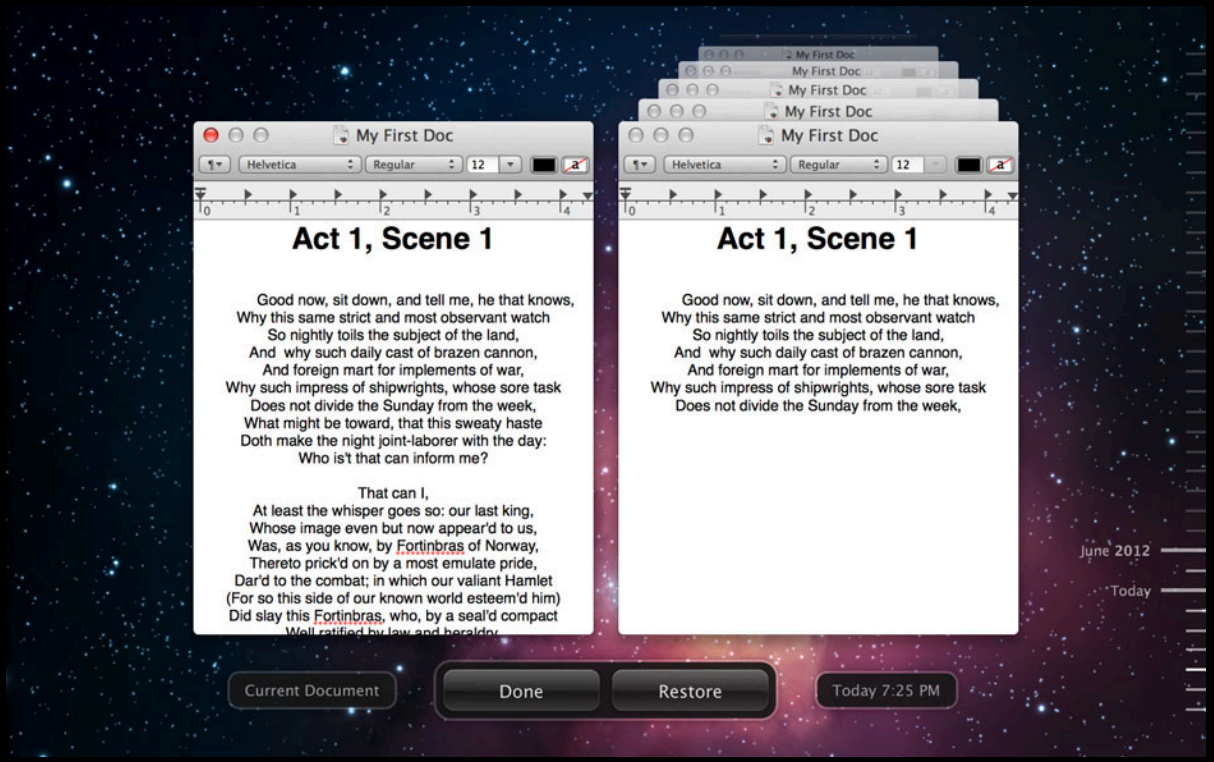
NSDocument

Versions



NSDocument

Conflicts



Related Session

Using iCloud with NSDocument	Marina Wednesday 3:15PM

Document Classes

Typical workflow

	iOS	OS X
Create a new document	UIDocument	NSDocument
Create a new Core Data document	UIManagedDocument	NSPersistentDocument
Document discovery in iCloud	NSMetadataQuery	NSDocument's open panel
Handle version conflicts	UIDocument, NSFileVersion	User managed
Move, rename and delete	NSFileCoordinator, NSFileManager	NSDocument's open panel

Document Classes

Typical workflow

	iOS	OS X
Create a new document	UIDocument	NSDocument
Create a new Core Data document	UIManagedDocument	NSPersistentDocument
Document discovery in iCloud	NSMetadataQuery	NSDocument's open panel
Handle version conflicts	UIDocument, NSFileVersion	User managed
Move, rename and delete	NSFileCoordinator, NSFileManager	NSDocument's open panel

Document Classes

Typical workflow

	iOS	OS X
Create a new document	UIDocument	NSDocument
Create a new Core Data document	UIManagedDocument	NSPersistentDocument
Document discovery in iCloud	NSMetadataQuery	NSDocument's open panel
Handle version conflicts	UIDocument, NSFileVersion	User managed
Move, rename and delete	NSFileCoordinator, NSFileManager	NSDocument's open panel

Document Classes

Typical workflow

	iOS	OS X
Create a new document	UIDocument	NSDocument
Create a new Core Data document	UIManagedDocument	NSPersistentDocument
Document discovery in iCloud	NSMetadataQuery	NSDocument's open panel
Handle version conflicts	UIDocument, NSFileVersion	User managed
Move, rename and delete	NSFileCoordinator, NSFileManager	NSDocument's open panel

Document Classes

Typical workflow

	iOS	OS X
Create a new document	UIDocument	NSDocument
Create a new Core Data document	UIManagedDocument	NSPersistentDocument
Document discovery in iCloud	NSMetadataQuery	NSDocument's open panel
Handle version conflicts	UIDocument, NSFileVersion	User managed
Move, rename and delete	NSFileCoordinator, NSFileManager	NSDocument's open panel

Document Classes

Typical workflow

	iOS	OS X
Create a new document	UIDocument	NSDocument
Create a new Core Data document	UIManagedDocument	NSPersistentDocument
Document discovery in iCloud	NSMetadataQuery	NSDocument's open panel
Handle version conflicts	UIDocument, NSFileVersion	User managed
Move, rename and delete	NSFileCoordinator, NSFileManager	NSDocument's open panel

Best Practices

Documents

Best Practices

Documents

- Subclass native document classes

Best Practices

Documents

- Subclass native document classes

  `NSDocument`

Best Practices

Documents

- Subclass native document classes

  `NSDocument`

  `UIDocument`

Best Practices

Documents

- Subclass native document classes
  - `NSDocument`
  - `UIDocument`
  - `UIManagedDocument`

Best Practices

Documents

- Subclass native document classes
  - `NSDocument`
  - `UIDocument`
  - `UIManagedDocument`
- Use the auto save behavior

Best Practices

Documents

- Subclass native document classes
  - `NSDocument`
  - `UIDocument`
  - `UIManagedDocument`
- Use the auto save behavior
- Core Data

Best Practices

Documents

- Subclass native document classes
  - `NSDocument`
  - `UIDocument`
  - `UIManagedDocument`
- Use the auto save behavior
- Core Data
  - Keeps the Core Data store local

Best Practices

Documents

- Subclass native document classes
  - `NSDocument`
  - `UIDocument`
  - `UIManagedDocument`
- Use the auto save behavior
- Core Data
  - Keeps the Core Data store local
  - Uploads change logs into iCloud

Best Practices

Documents

- Subclass native document classes

  `NSDocument`

  `UIDocument`

  `UIManagedDocument`

- Use the auto save behavior
- Core Data
  - Keeps the Core Data store local
  - Uploads change logs into iCloud
  - Use migration to prepopulate a store

Best Practices

iOS documents

Best Practices

iOS documents

- Actively track documents

Best Practices

iOS documents

- Actively track documents

  `NSMetadataQuery`

Best Practices

iOS documents

- Actively track documents

  `NSMetadataQuery`

- Download needed files

Best Practices

iOS documents

- Actively track documents

  `NSMetadataQuery`

- Download needed files
- Support conflict resolution

Best Practices

iOS documents

- Actively track documents

  `NSMetadataQuery`

- Download needed files
- Support conflict resolution

  `UIDocumentStateInConflict`

Best Practices

iOS documents

- Actively track documents

  `NSMetadataQuery`

- Download needed files
- Support conflict resolution

  `UIDocumentStateInConflict`

  - Avoid user involvement if possible

Best Practices

OS X documents

Best Practices

OS X documents

- Avoid deadlocks with modal UI

Best Practices

OS X documents

- Avoid deadlocks with modal UI
- OS manages conflict resolution

Best Practices

OS X documents

- Avoid deadlocks with modal UI
- OS manages conflict resolution
- iCloud aware open dialogs

iCloud Storage

Debugging the cloud

iCloud Storage

Debugging the cloud

- Test with multiple devices

iCloud Storage

Debugging the cloud

- Test with multiple devices
- Monitor network traffic

iCloud Storage

Debugging the cloud

- Test with multiple devices
- Monitor network traffic
- Use airplane mode to induce conflicts

iCloud Storage

Debugging the cloud

- Test with multiple devices
- Monitor network traffic
- Use airplane mode to induce conflicts
- Configuration profile

iCloud Storage

Debugging the cloud

- Test with multiple devices
- Monitor network traffic
- Use airplane mode to induce conflicts
- Configuration profile
- File bug reports!

iCloud Storage

Debugging the cloud

- Test with multiple devices
- Monitor network traffic
- Use airplane mode to induce conflicts
- Configuration profile
- File bug reports!
  - bugreport.apple.com

iCloud Storage

Debugging the cloud

- Test with multiple devices
- Monitor network traffic
- Use airplane mode to induce conflicts
- Configuration profile
- File bug reports!
  - bugreport.apple.com
  - Developer forum

iCloud Storage

Testing your app

iCloud Storage

Testing your app

- developer.icloud.com

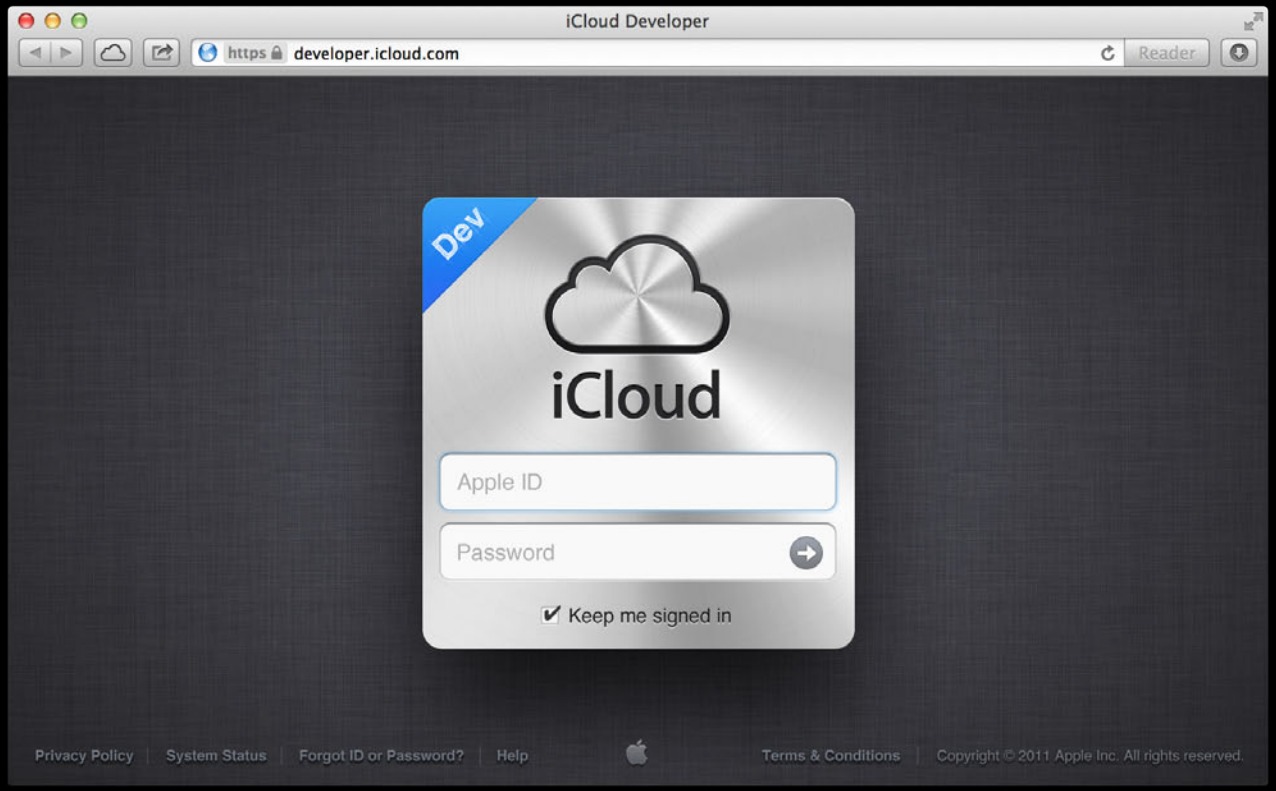
iCloud Storage

Testing your app

- developer.icloud.com
  - Requires activating iOS 6 device

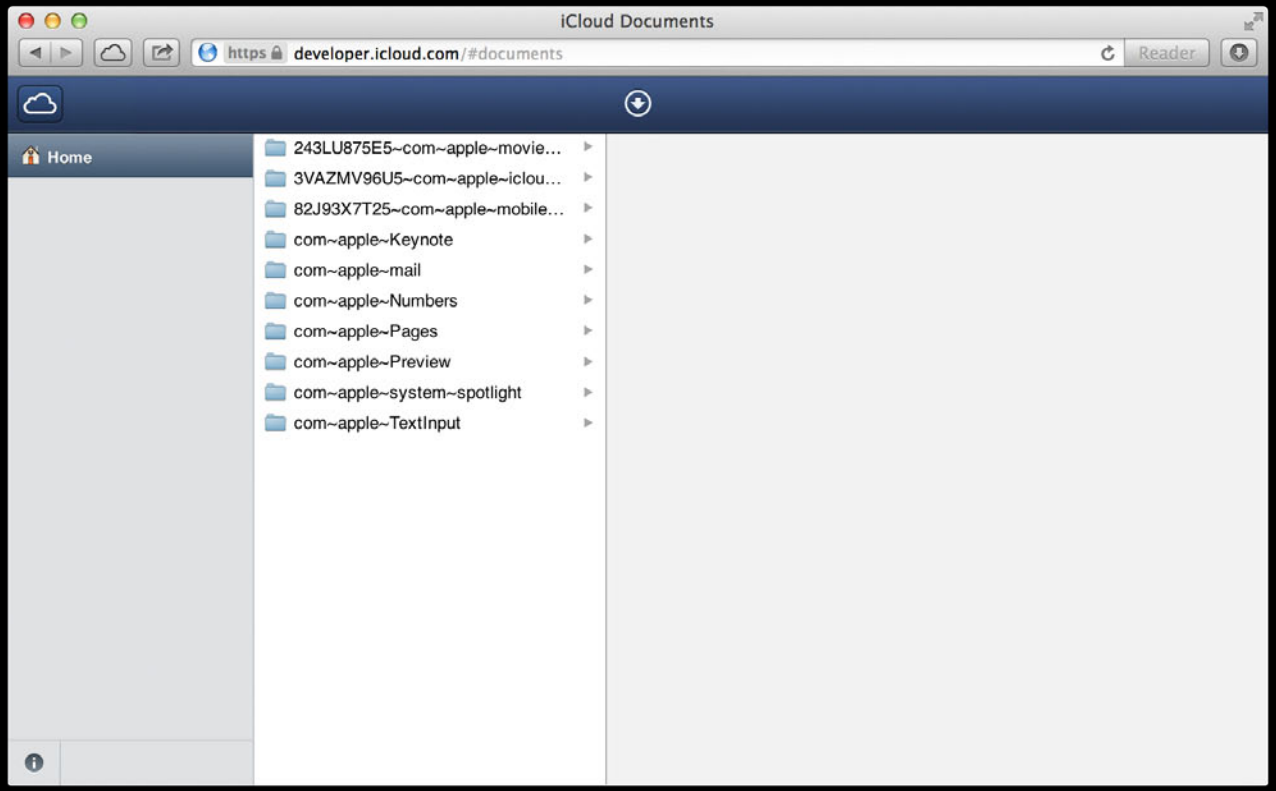
Debugging the Cloud

developer.icloud.com



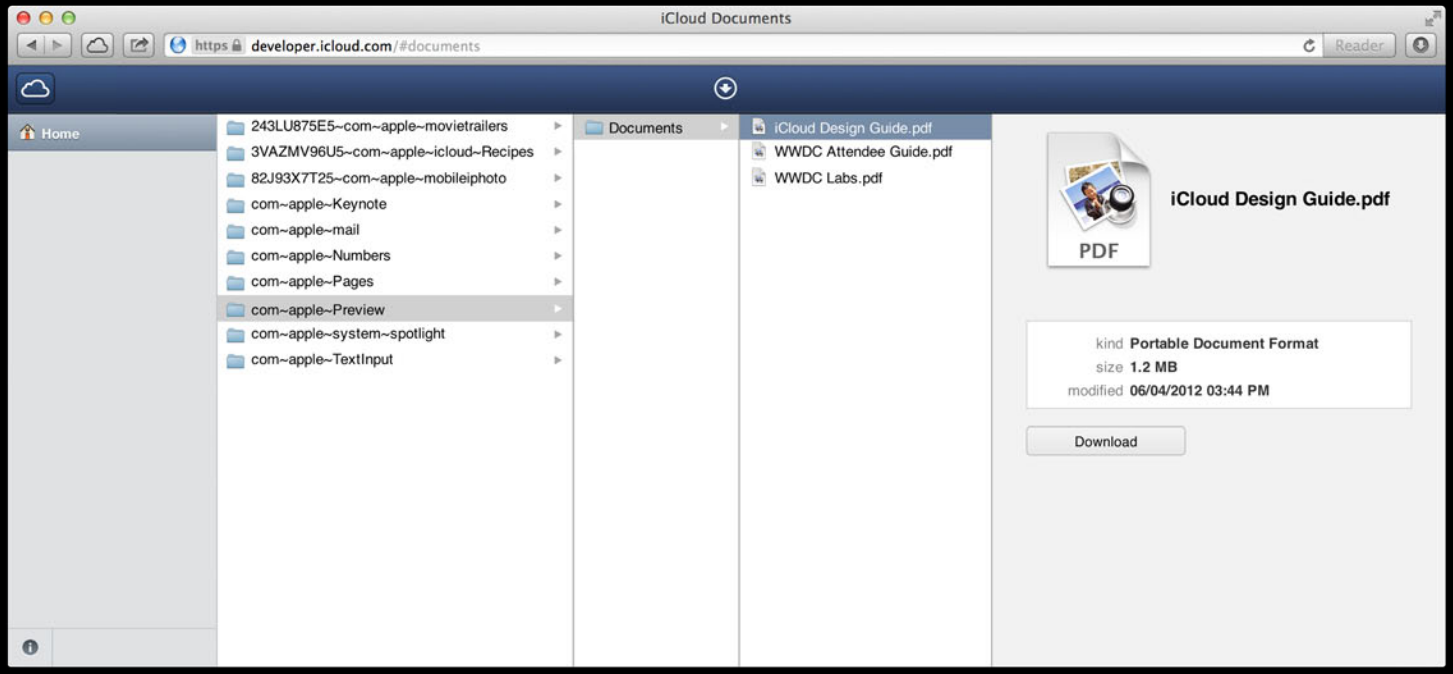
Debugging the Cloud

developer.icloud.com



Debugging the Cloud

developer.icloud.com



More Information

Michael Jurewitz
Developer Tools Evangelist
jury@apple.com

Documentation
iCloud Design Guide

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

iOS Dev Center
http://developer.apple.com/devcenter/ios/

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
Advanced iCloud Document Storage	Marina Thursday 3:15PM

Labs

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

Summary

- iCloud makes for a great user experience
- Key Value Storage
  - Provides a consistent experience between devices
- iCloud Account
  - New account token and notification
  - Container URL
- Document Storage
  - Document types
  - Best practices


WWDC2012