# Systemwide Previews on OSX and iOS

## Previewing with Quick Look

Session 106

Philippe Champeaux, Julien Robert

# What You'll Learn

- What is Quick Look
- Previewing with Quick Look in iOS and Mac OS X
- Thumbnailing with Quick Look in Mac OS X
- Producing thumbnails and previews in Mac OS X

# What Is Quick Look?

# What Is Quick Look?

## Displays thumbnails and previews of documents

- All basic document types
  - Image
  - PDF
  - Text , HTML, RTF
  - QuickTime movie
  - Audio files
  - iWork and Office documents
- Your own content in Mac OS X

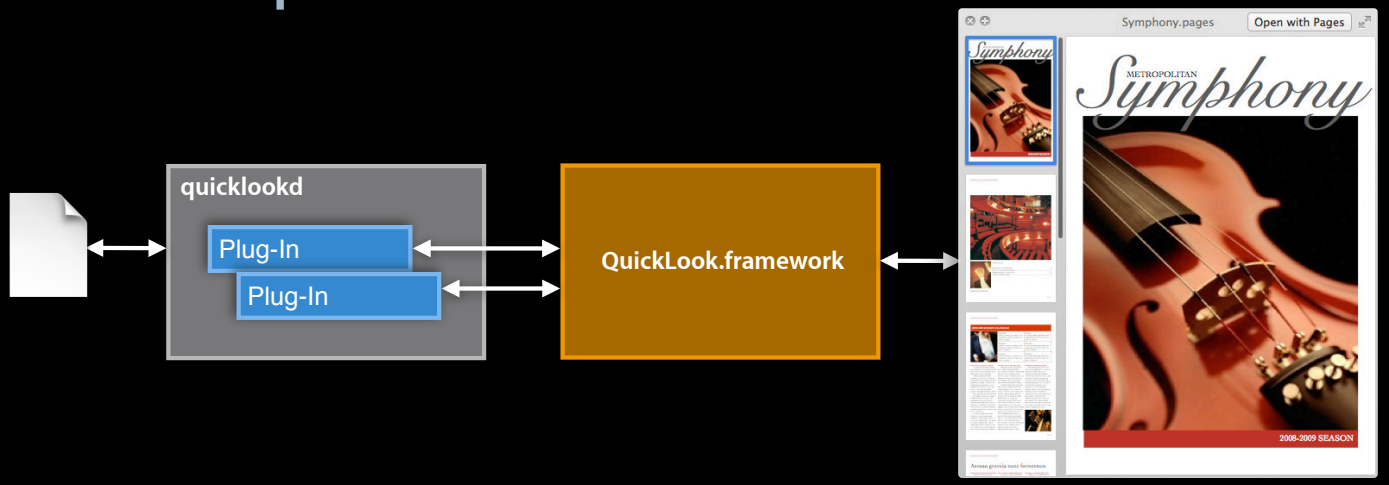
# What Is Quick Look?

## …and where can you see it?

- In Mail
- In Finder
- In Safari in iOS
- In Dock and Spotlight
- In your applications

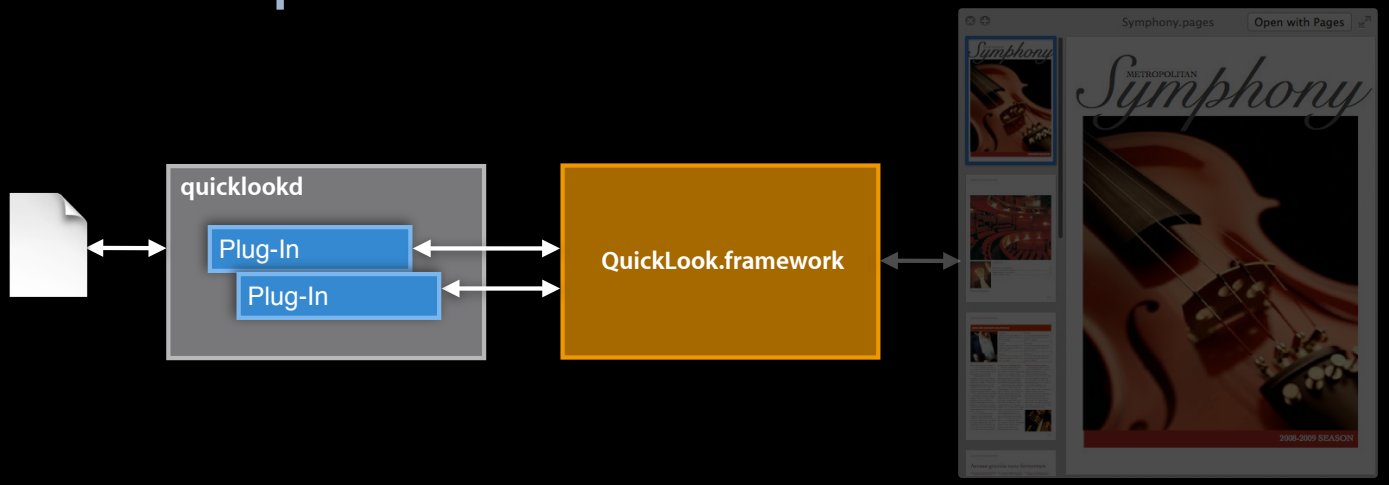
# How Does Quick Look Work?

## Two functional parts



# How Does Quick Look Work?

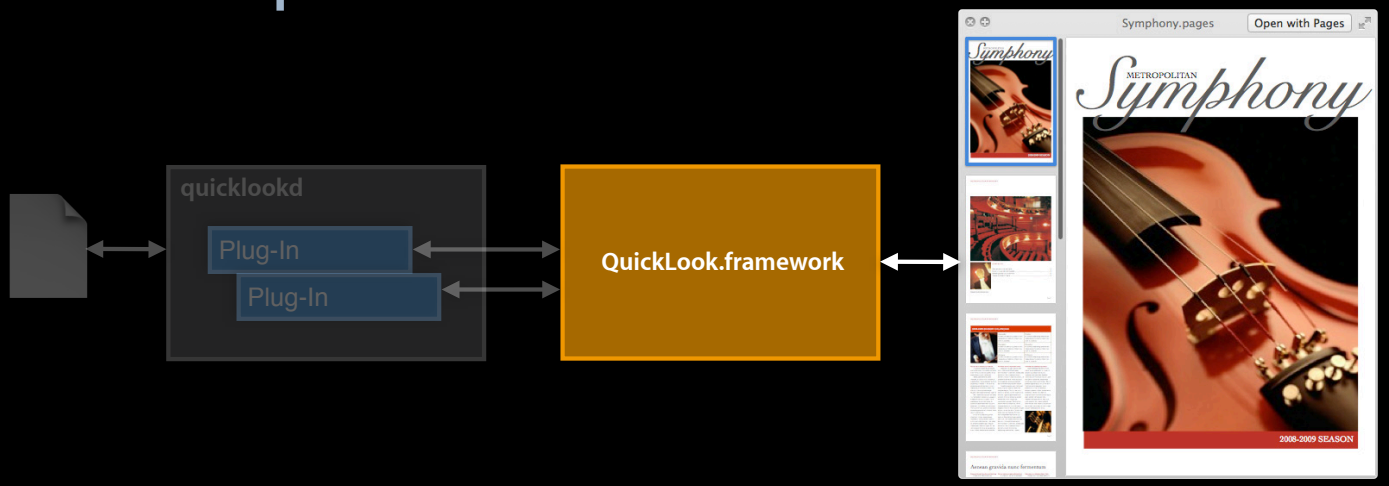
## Two functional parts




- Producer tells Quick Look how to read a file

# How Does Quick Look Work?

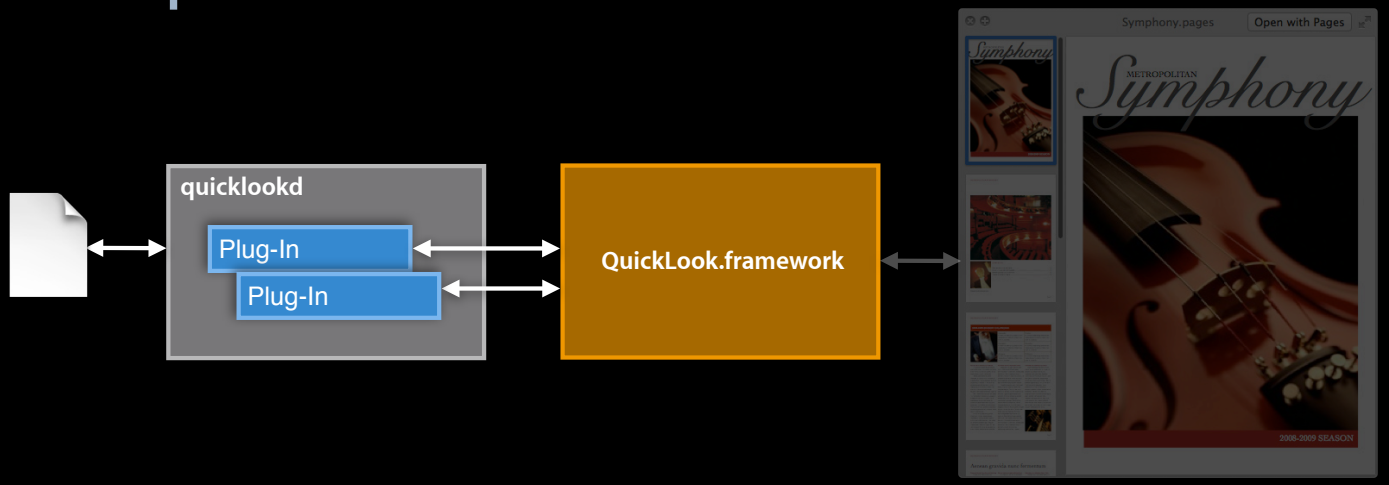
## Two functional parts




- Producer tells Quick Look how to read a file
- Consumer displays file content in your application

# How Does Quick Look Work?

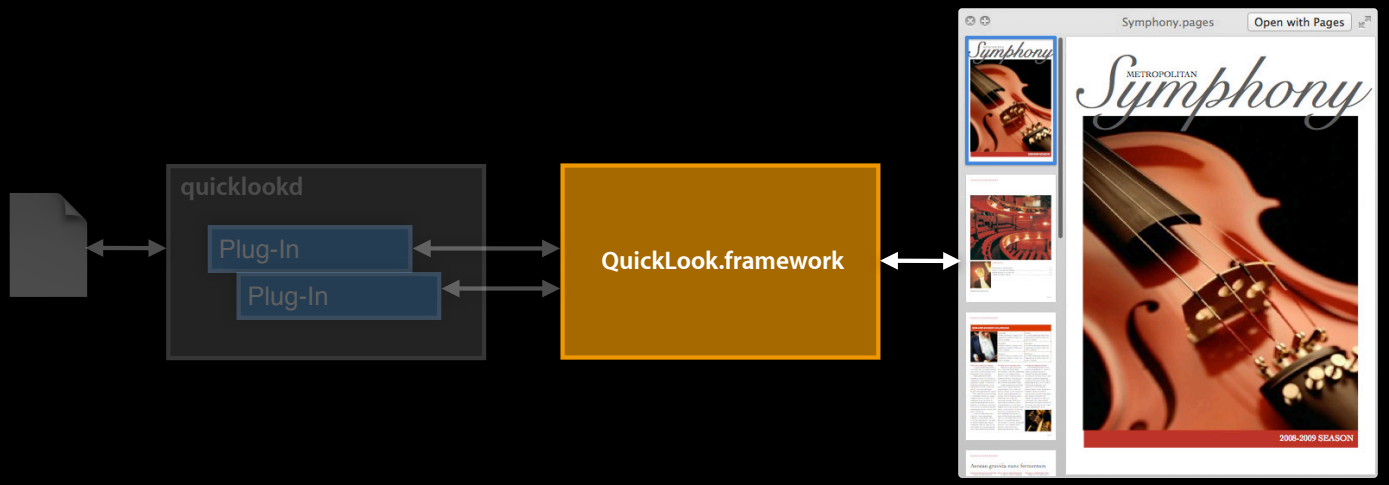
## Quick Look producer




- Translates complex document types into known formats
  - HTML, PDF, text, images, movies
- Uses a plug-in architecture
  - Plug-ins loaded on demand by quicklookd
  - Xcode includes a Quick Look plug-in template

# How Does Quick Look Work?

## Quick Look consumer




- Displays the content in your application
  - Preview panel/view in Mac OS X
  - Preview controller in iOS
  - Thumbnails in Mac OS X

# Quick Look Support

| | Previews | Thumbnails | Plug-ins |
|---|---|---|---|
| Mac OS X | ✓ | ✓ | ✓ |
| iOS | ✓ | | |

# Previewing with Quick Look

# Previewing with Quick Look

## In Mac OS X

# Previewing with Quick Look in Lion

## Use a panel or a view

- Preview panel
  - Subclass of NSPanel
  - One shared panel per application
  - Targets the current selection
- Preview view
  - Subclass of NSView
  - Multiple views per application
  - Directly control the previewed item

# Using the Preview Panel

## New user interface in Lion


- Does not hide when application is in background
- Swipe to navigate
- Prominent open button
- Previews in separate process

# Using the Preview Panel

## Getting started

- Link against the Quartz framework

```
#import <Quartz/Quartz.h>
[QLPreviewPanel sharedPreviewPanel]
```

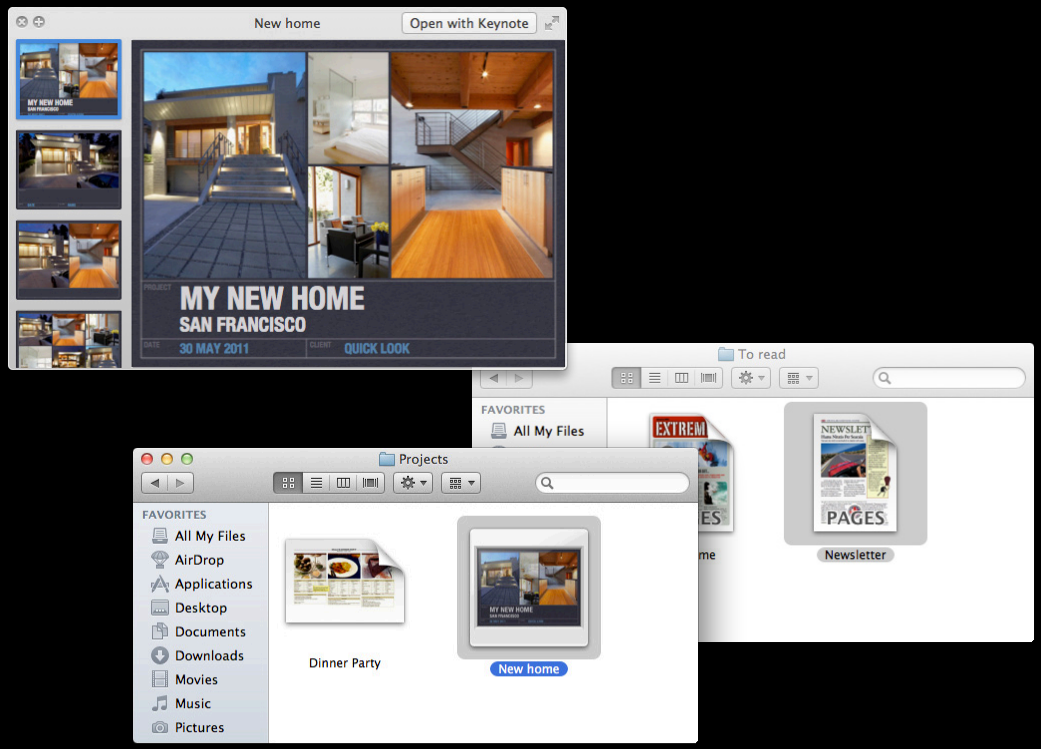
- Open/close as a regular panel

```
- (void)makeKeyAndOrderFront:(id)sender;
- (void)orderOut:(id)sender;
- (BOOL)enterFullScreenMode:(NSScreen *)screen withOptions:(NSDictionary *)
options;
- (void)exitFullScreenModeWithOptions:(NSDictionary *)options;
```

# Using the Preview Panel

## Tracking the selection

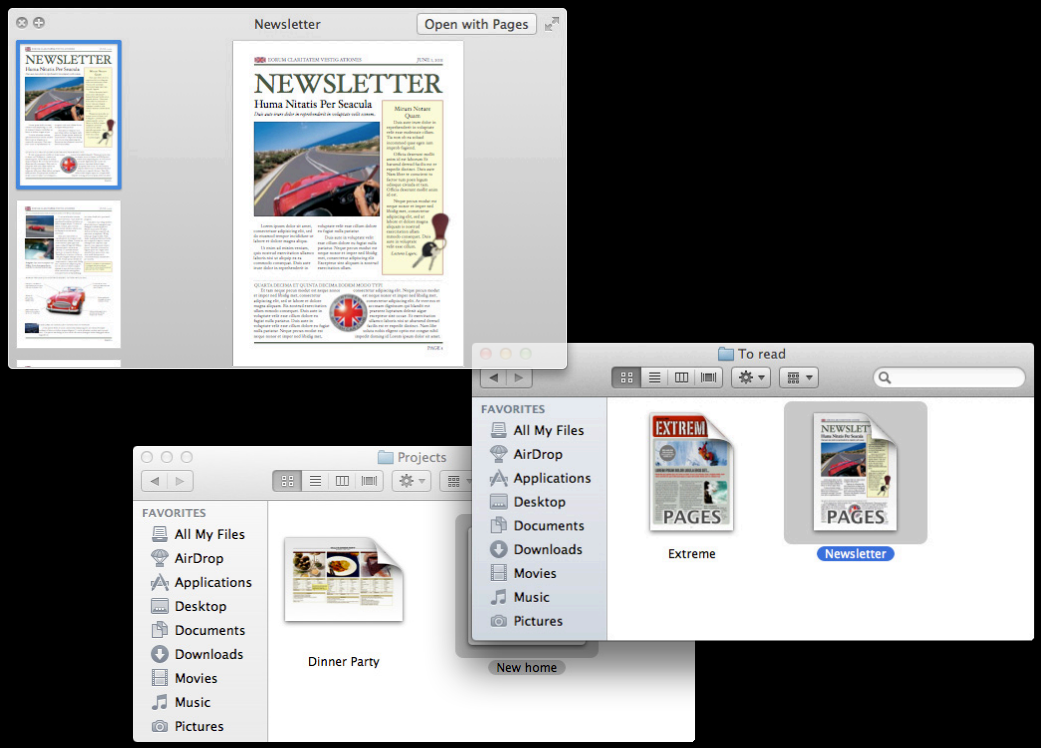
- Panel follows the selected item in the key window



# Using the Preview Panel

## Tracking the selection

- Panel follows the selected item in the key window



# Using the Preview Panel

## Tracking the selection

- Finds a controller through the responder chain
  - `- (BOOL)acceptsPreviewPanelControl:(QLPreviewPanel *)panel;`
- Give it the ownership of the preview panel
  - `- (void)beginPreviewPanelControl:(QLPreviewPanel *)panel;`
- Take it back
  - `- (void)endPreviewPanelControl:(QLPreviewPanel *)panel;`

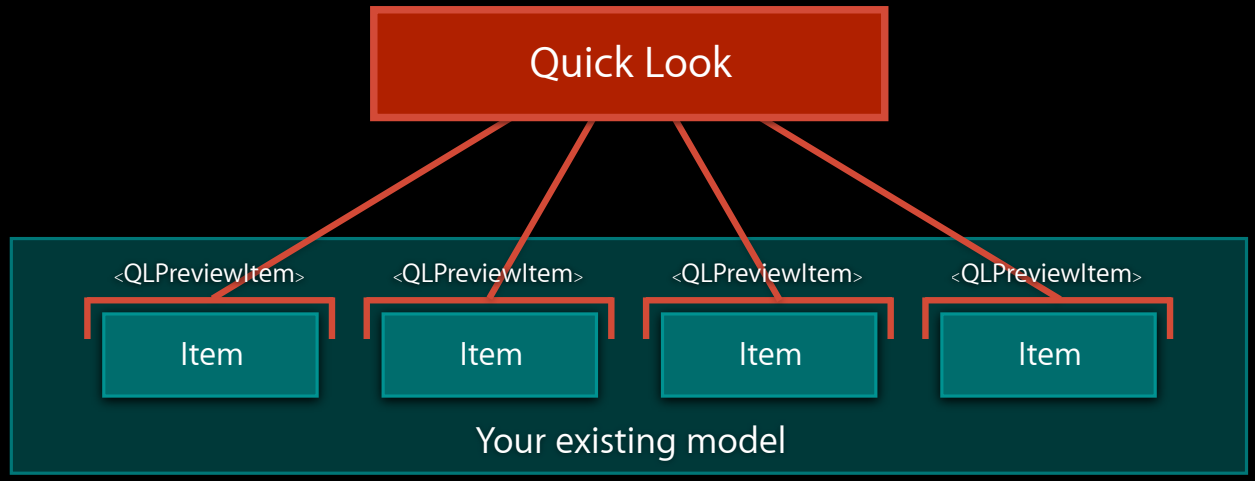
Never assume you own the preview panel

# Using the Preview Panel

## Providing content

- Datasource methods

```
- (NSInteger)numberOfPreviewItemsInPreviewPanel:(QLPreviewPanel *)panel;
- (id <QLPreviewItem>)previewPanel:(QLPreviewPanel *)panel
previewItemAtIndex:(NSInteger)index;
```



# Using the Preview Panel

## Providing content

- The QLPreviewItem protocol

```
@protocol QLPreviewItem <NSObject>
@property(readonly) NSURL *previewItemURL;
@property(readonly) NSString *previewItemTitle;
@end
```

- NSURL implements the QLPreviewItem protocol

```
@interface NSURL (QuickLook) <QLPreviewItem>
@end
```

# Using the Preview Panel

## Handling events

- Open/close shortcut

  `⌘-Y and/or spacebar`

- Getting events in your background view

  `- (BOOL)previewPanel:(QLPreviewPanel*)panel handleEvent:(NSEvent*)event`

  - Use for navigation in your view
  - Return YES if you handled the event, NO otherwise

# Using the Preview Panel

## Improving transition

- Delegate methods

```
- (NSRect)previewPanel:(QLPreviewPanel*)panel
sourceFrameOnScreenForPreviewItem:(id <QLPreviewItem>)item
```

```
- (id)previewPanel:(QLPreviewPanel*)panel transitionImageForPreviewItem:(id
<QLPreviewItem>)item contentRect:(NSRect*)contentRect
```

# Using the Preview View

## Getting started

- Link against the Quartz framework

```
#import <Quartz/Quartz.h>
[[QLPreviewView alloc] initWithFrame:frame style:style]
```

- Two styles
  - Normal `QLPreviewViewStyleNormal`
    - Use in full previews
  - Compact `QLPreviewViewStyleCompact`
    - Use in inspectors

# Using the Preview View

## Providing content

- Provide the preview item directly

```
@property (retain) id <QLPreviewItem> previewItem;
```

- Loads asynchronously
  - Do not assume its state
- Request reload if preview item modified

```
- (void)refreshPreviewItem;
```

# Using the Preview View

## Managing view lifecycle

Only on Mac OS

- Automatic with window (default mode)

```
previewView.shouldCloseWithWindow = YES;
// previewView will be closed when the window is closed
```

  - View killed when window closed

- Manual

```
previewView.shouldCloseWithWindow = NO;
[...]
// when the preview view won't be used anymore, ever
[previewView close];
```

# Seamless Opening

## Free for good citizens


- Seamless transition from icon or preview to full document
- Great integration with Quick Look
- No API
- Conditions
  - Use NSDocument correctly (a Kit lab)
  - Launch quickly (less than a se

# Demo

## Adopting the preview panel and preview view

# Summary

## Adopting the preview panel

1. Import and link against Quartz
2. Add calls to open/close the panel
3. Implement the controller methods
4. Implement the data source
5. Add code to reload data when the selection changes
6. Implement some delegate methods

# Previewing with Quick Look

## In iOS

# Previewing with Quick Look in iOS

## A subclass of UIViewController

- QLPreviewController
- Uses the standard API
- Integrates perfectly in the iOS workflow

# Previewing with Quick Look in iOS

## A subclass of UIViewController

- Displays all default document types
- Supports opening in other applications
- Supports printing

# Previewing with Quick Look in iOS

## Preview multiple documents

Only on iPad

# Previewing with Quick Look in iOS

## Advanced page navigation

5

Only on iPad

# Previewing with Quick Look in iOS

## Presenting the preview controller

- Setup

```
#import <QuickLook/QuickLook.h>
QLPreviewController *controller = [[QLPreviewController alloc] init];
controller.dataSource = self;
```

- Presentation

| presentModalViewController: animated: |
| --- |
| pushViewController: animated: |

# Previewing with Quick Look in iOS

## Providing content

- Data source

```
- (NSInteger)numberOfPreviewItemsInPreviewController:(QLPreviewController
*)controller;
```

```
- (id <QLPreviewItem>)previewController:(QLPreviewController *)controller
previewItemAtIndex:(NSInteger)index;
```

- Delegate methods

```
- (CGRect)previewController:(QLPreviewController *)controller
frameForPreviewItem:(id <QLPreviewItem>)item inSourceView:(UIView **)view
```

```
- (UIImage *)previewController:(QLPreviewController *)controller
transitionImageForPreviewItem:(id <QLPreviewItem>)item contentRect:(NSRect
*)contentRect
```

# Demo

## Adopting the preview controller

# Previewing with Quick Look in iOS

## Common pitfalls

- Make sure the delegate exists until the end
- Do not set the parent controller as preview item
- Leave the navigation bar to the preview controller

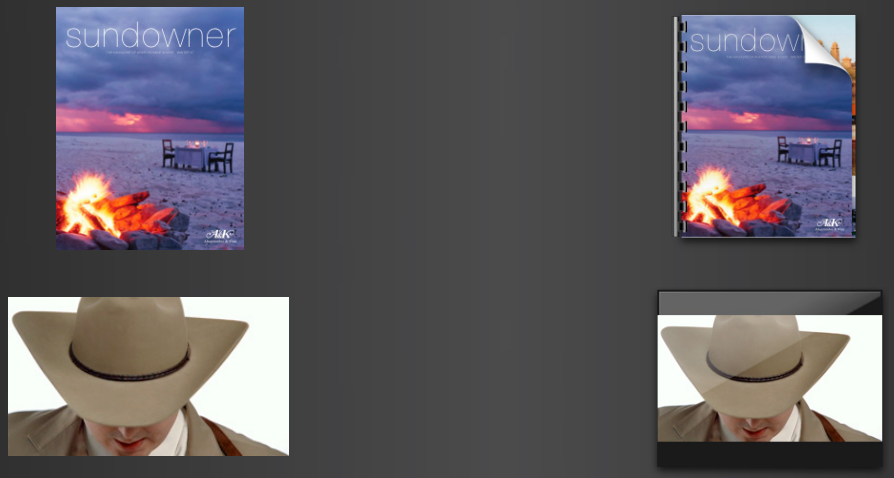
# Thumbnailing in Mac OS X

# Thumbnails

## Thumbnails are static images

Only on Mac OS

- Without or with icon decorations
  - Flat thumbnails: quick overview of content
  - Icons: emphasizes the file aspect

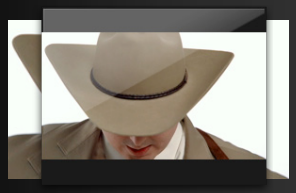


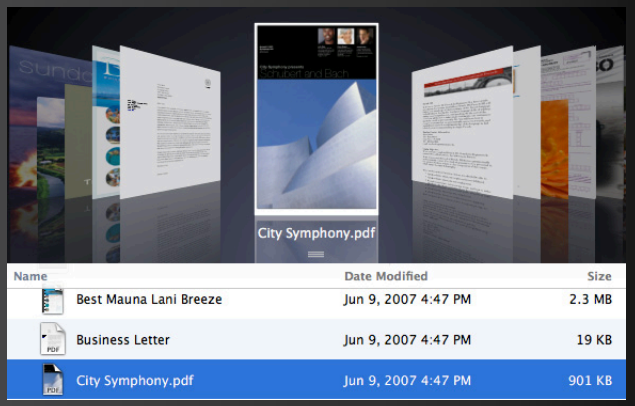
# Thumbnails

## Flat thumbnails

Only on Mac OS

- High resolution
- Inspectors, preview panes
- 3D views, other representations



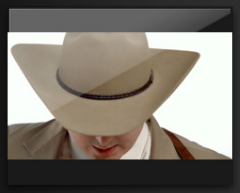



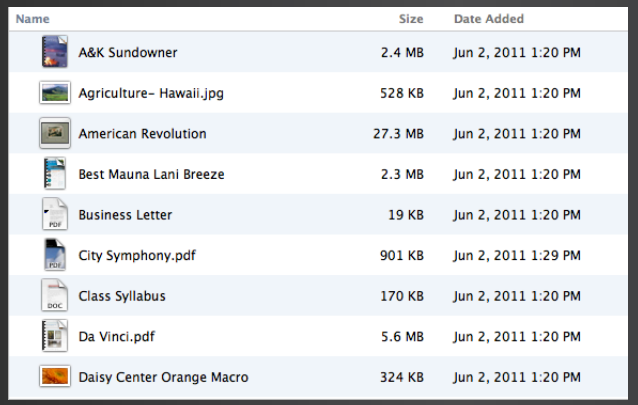
# Thumbnails

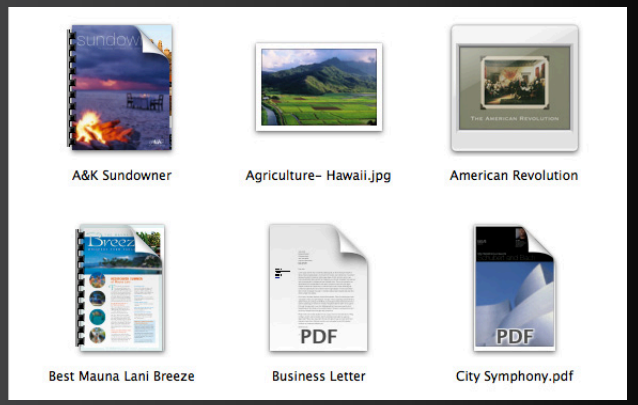
## Icons

Only on Mac OS

- Collection of items
  - Browser views, list views
- File handles





# Displaying Thumbnails

## Synchronous API

Only on Mac OS

- In the QuickLook framework

```
CGImageRef QLThumbnailImageCreate(CFURLRef url, CGSize maxThumbnailSize,
CFDictionaryRef options);
```

- Option to have icons

```
const CFStringRef kQLThumbnailOptionIconModeKey;
```

- May take time to return

Do not call this on your main thread

# Displaying Thumbnails

## Asynchronous API

Only on Mac OS

- Create CFType object

```
QLThumbnailRef thumbnail = QLThumbnailCreate(NULL, url, maxSize, options);
```

- Dispatch request using GCD

```
QLThumbnailDispatchAsync(thumbnail, dispatch_get_main_queue(), ^{
      CGImageRef image = QLThumbnailCopyImage(thumbnail);
      // use your image and release the thumbnail
});
```

- Cancelable

```
void QLThumbnailCancel(QLThumbnailRef thumbnail);
```

- Interaction with preview panel

```
CGRect QLThumbnailGetContentRect(QLThumbnailRef thumbnail)
```

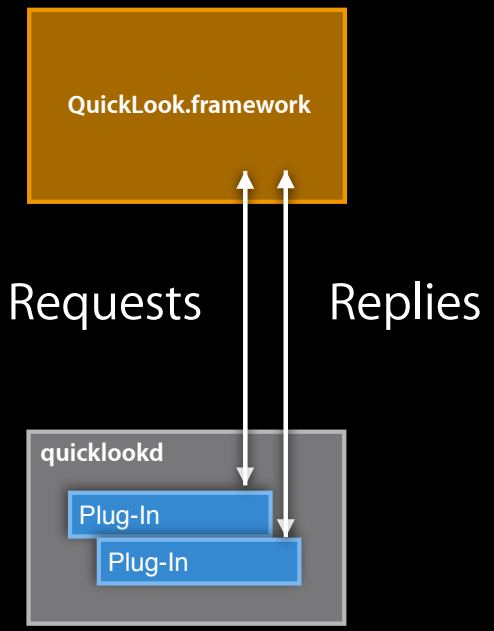
# Producing in Mac OS X

## Thumbnails and previews for your documents

# Quick Look Plug-Ins

## Architecture

- Extend support to more document types
- Sandboxed in quicklookd



# Quick Look Plug-Ins

## Implementing your plug-in

- Thumbnail request

```
GenerateThumbnailForURL(QLThumbnailRequestRef thumbnail, CFURLRef url) {
      // your code here
}
```

  - Give thumbnail image directly

```
QLThumbnailRequestSetImage(thumbnail, image)
QLThumbnailRequestSetImageWith[Data|URL](thumbnail, dataOrURL)
```

  - Draw it live

```
CGContextRef context = QLThumbnailRequestCreateContext(thumbnail)
// draw thumbnail in context
QLThumbnailRequestFlushContext(thumbnail)
```

# Quick Look Plug-Ins

## Implementing your plug-in

- Preview request

```
GeneratePreviewForURL(QLPreviewRequestRef preview, CFURLRef url) {
      // your code here
}
```

  - Give preview data directly

```
QLPreviewRequestSetDataRepresentation(preview, data, contentType)
QLPreviewRequestSetURLRepresentation(preview, url, contentType)
```

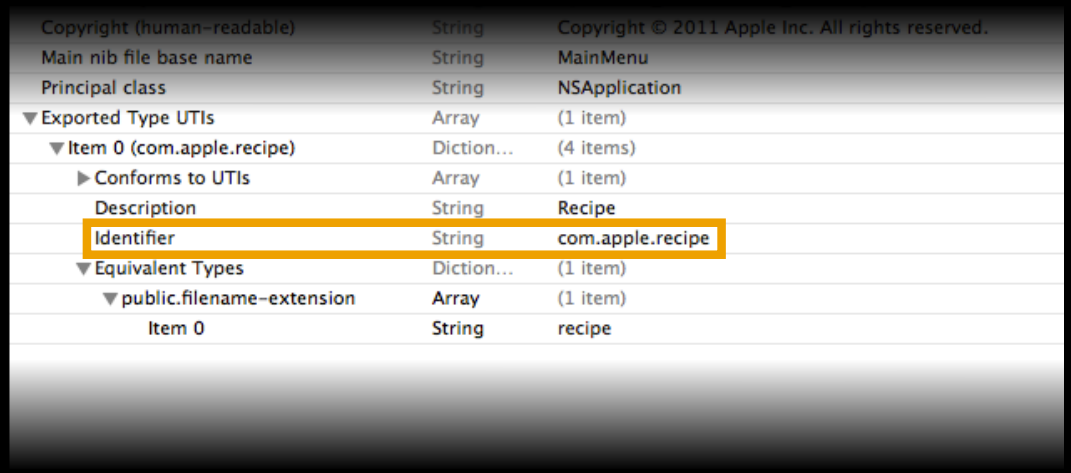
  - Draw it live

```
CGContextRef context = QLPreviewRequestCreate[PDF]Context(preview)
 // draw preview in context
QLPreviewRequestFlush[PDF]Context(preview)
```

# Quick Look Plug-Ins

## Document types (a.k.a. UTIs)

- Unique string
- Defined in Info.plist
  - Application
  - Plug-ins (Quick Look, Spotlight)
- Registered in LaunchServices



# Quick Look Plug-Ins

## Registering your plug-in

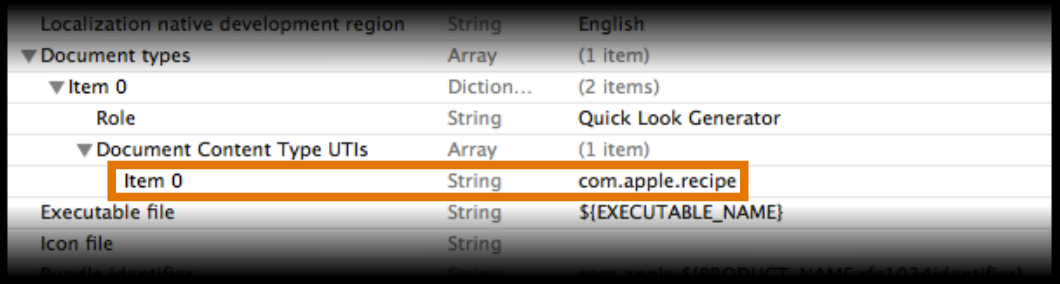
- Embedded in your application

  `MyApp.app/Contents/Library/QuickLook/PlugIn.qlgenerator`

- Standalone

  `[~]/Library/QuickLook/PlugIn.qlgenerator`

- Linked to your document type in Info.plist



# Quick Look Plug-Ins

## Specifying your plug-in attributes

- In Info.plist
- Static sizing attributes
  - `QLThumbnailMinimumSize`
    - Will not try to generate a smaller thumbnail
  - `QLPreviewWidth / QLPreviewHeight`
    - Default size for the preview
- Running attributes
  - `QLSupportsConcurrentRequests`
  - `QLNeedsToBeRunInMainThread`
    - Valid for both thumbnail and preview generation

# Quick Look Plug-Ins

## Using options and properties

- Options are given *to* you to adjust generation

  `kQLThumbnailOptionScaleFactorKey`

- Properties are given *by* you to customize display
  - Thumbnail properties
    - Badge and extension

      `kQLThumbnailPropertyExtensionKey / kQLThumbnailPropertyBadgeImageKey`
  - Preview properties
    - Sizing

      `kQLPreviewPropertyWidthKey / kQLPreviewPropertyHeightKey`
    - Custom title

      `kQLPreviewPropertyDisplayNameKey`

# Demo

## Creating a Quick Look plug-in

# Summary

## Creating a Quick Look plug-in

1. Create template from Xcode
2. Define your document type in Info.plist
3. Implement the thumbnail and preview prototypes
4. Define the plug-in attributes in Info.plist
5. Test your plug-in with qlmanage (man qlmanage)
6. Install it and test in Finder
7. Ship it!

# Where to Go from Here

- Use Quick Look in your applications
  - Show previews
  - Display thumbnails
- Write a Mac OS X Quick Look plug-in
  - Make your documents visible everywhere
  - Polish your thumbnails
  - Focus on performance

# More Information

**Bill Dudney**
iOS Apps & Frameworks Evangelist
dudney@apple.com

**Documentation**
Mac OS X Human Interface Guidelines
http://developer.apple.com/ue

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

# Labs

| | |
|---|---|
| Quick Look Lab | Application Frameworks Lab C<br>Today 4:30PM–6:00PM |
| Cocoa Lab | Application Frameworks Lab A<br>Today 2:00PM–6:00PM |
| UIViewController Lab | Application Frameworks Lab B<br>Today 2:00PM–4:15PM |


WWDC2011