What's New in Cocoa

Session 204

Ali Ozer
Director of Cocoa Frameworks

What's New in Cocoa

- Changes in AppKit and Foundation in Mountain Lion
- Pointers to labs and other talks

Topics

- iCloud
- High Resolution
- Gestures
- Layer-Backed Views
- Notification Center
- Cocoa XPC
- Objective-C
- And More!

iCloud

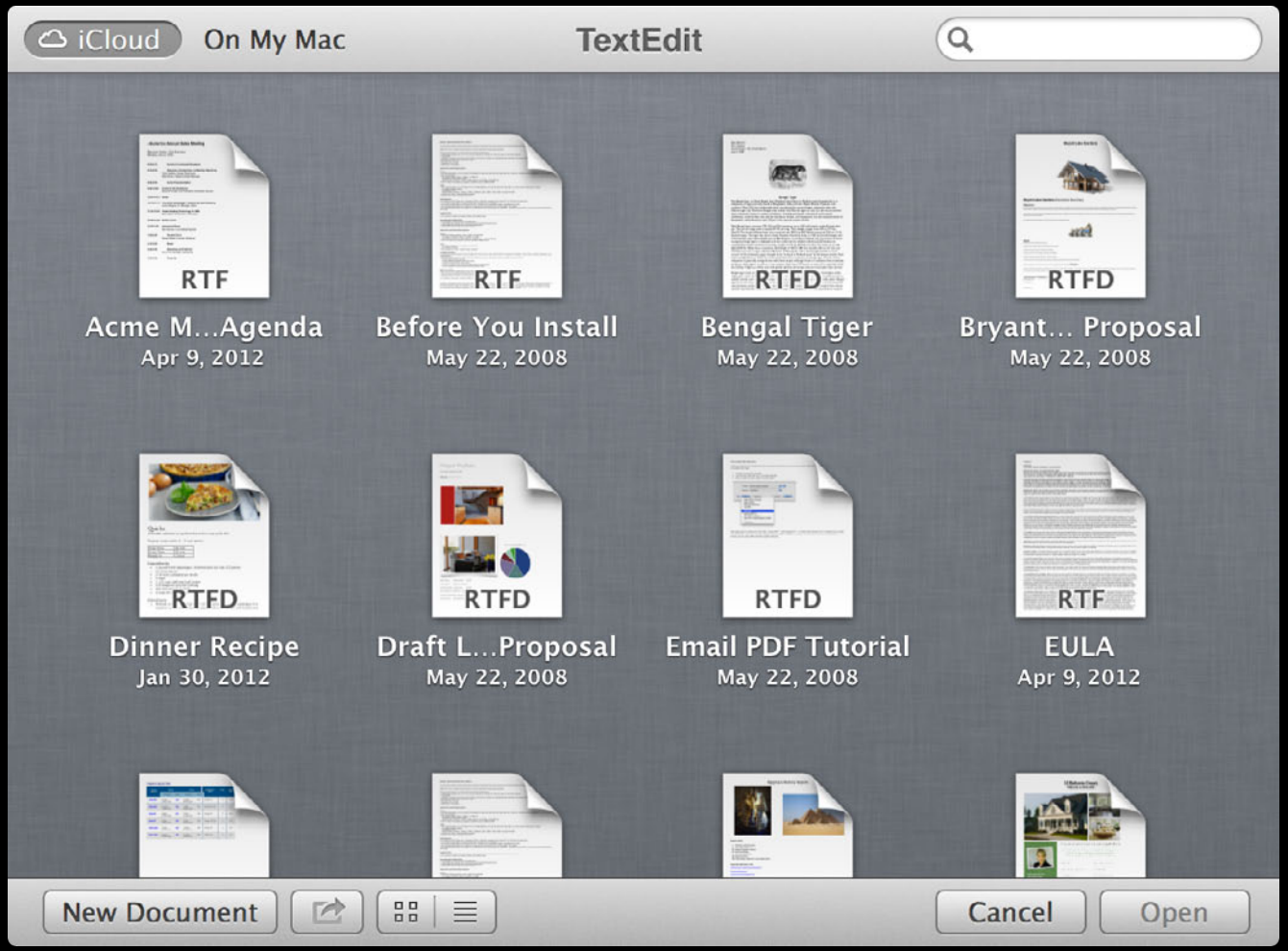
iCloud

- NSDocument support
  - Auto save
- NSFileManager

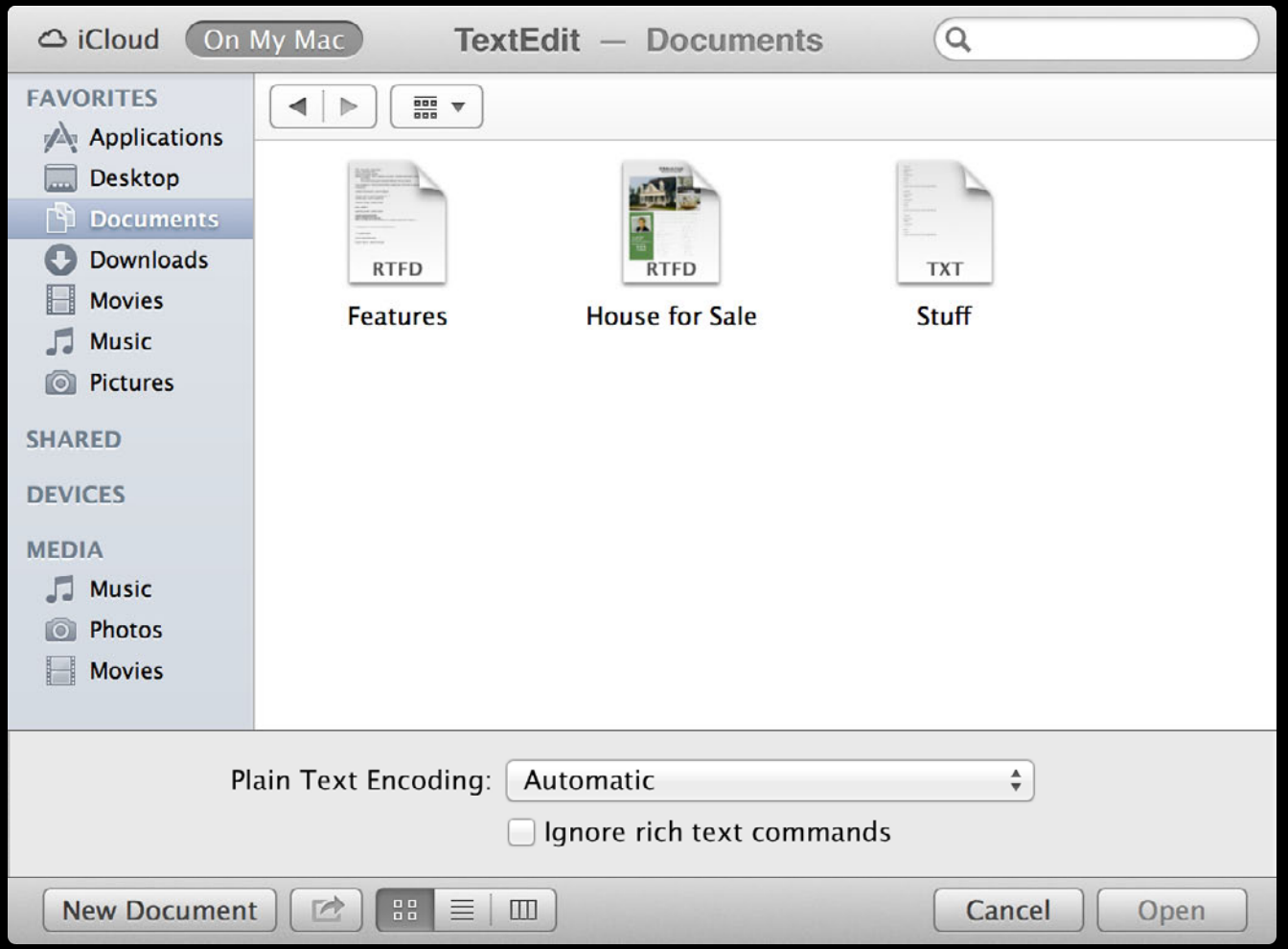
Demo

NSDocument iCloud support

iCloud Open Panel

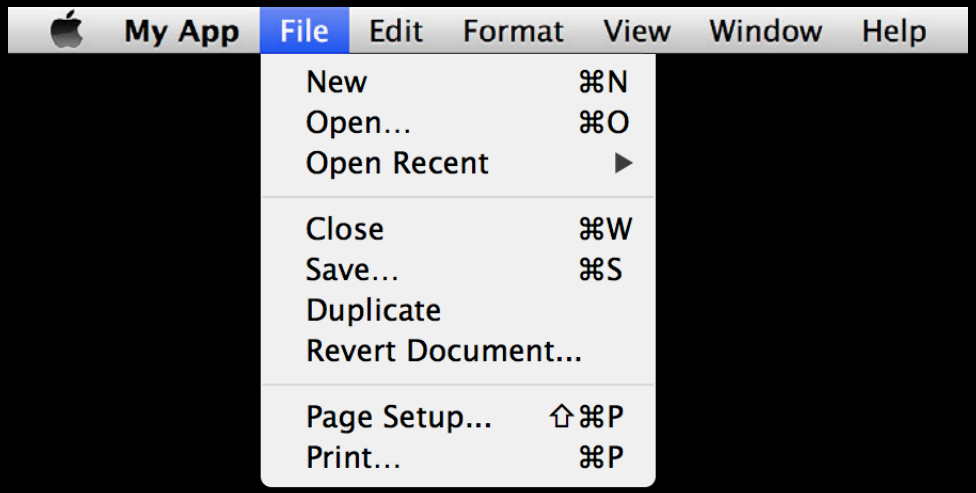


iCloud Open Panel

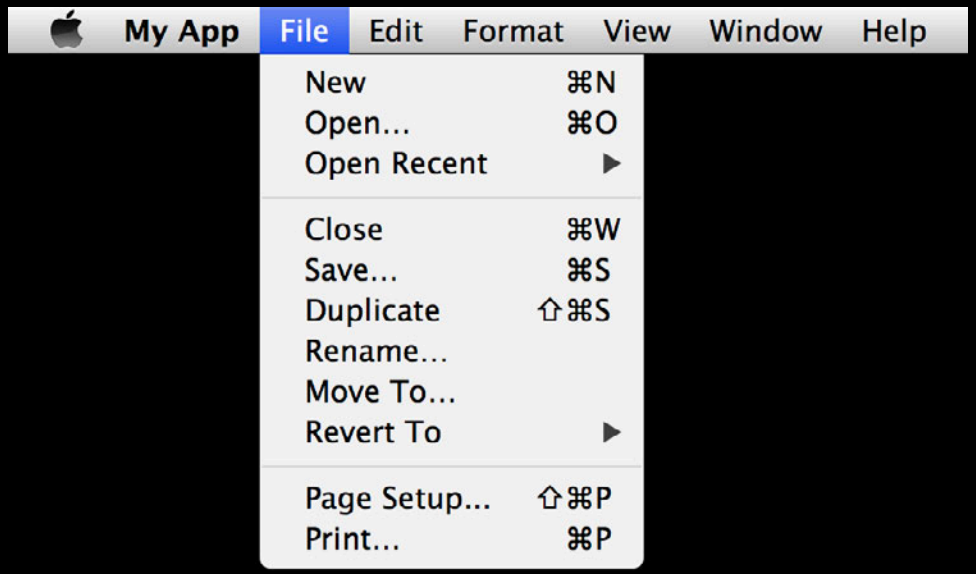


File Menu

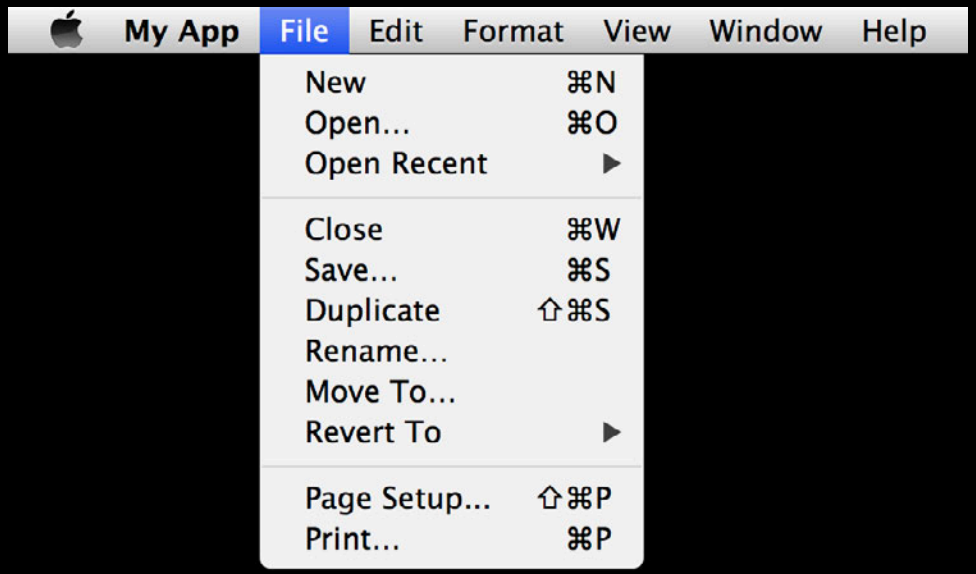
Lion



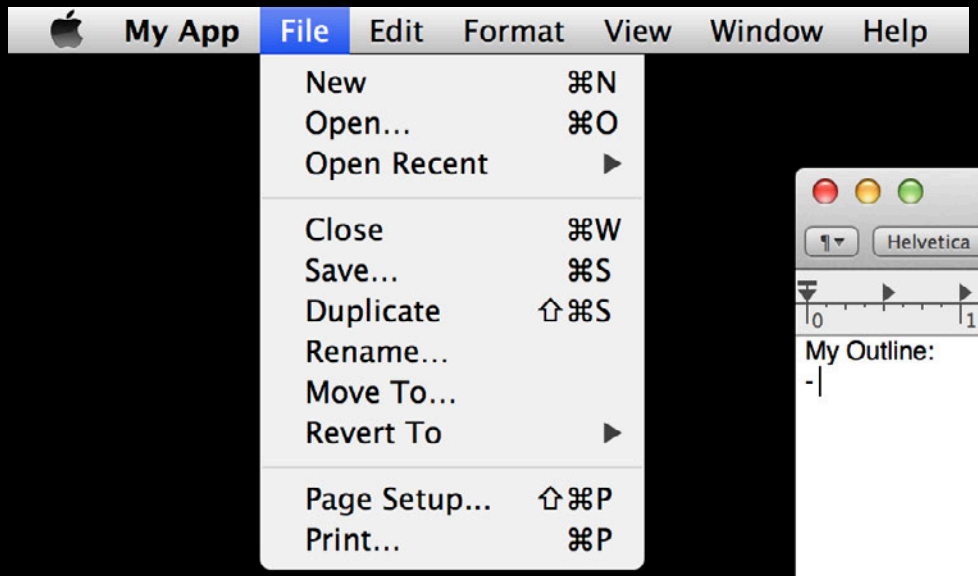
File Menu

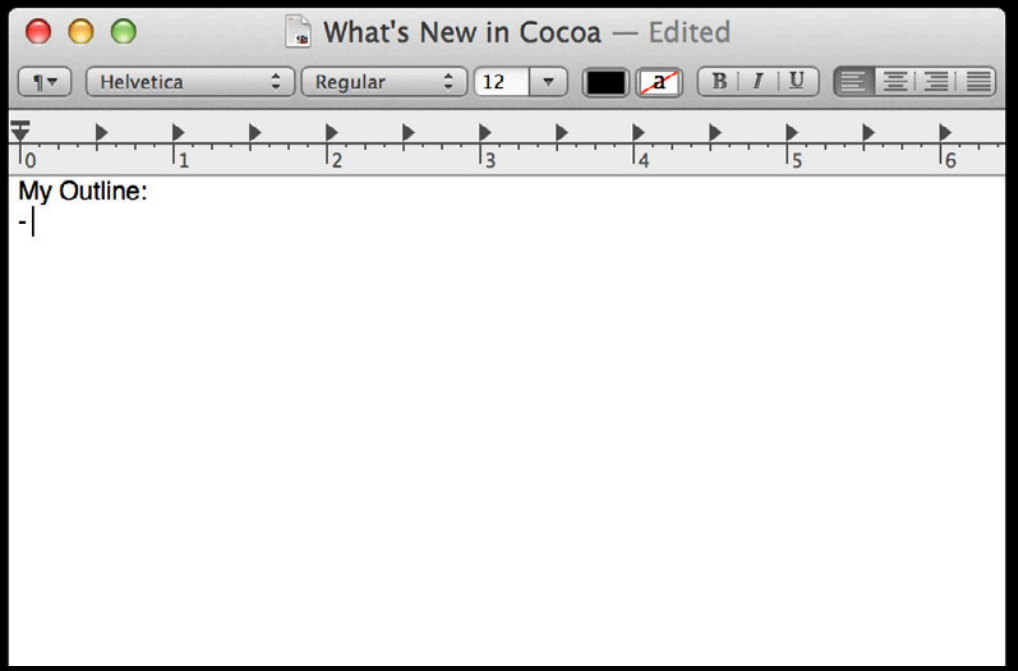

My App
File
Edit
Format
View
Window
Help
New ⌘N
Open... ⌘O
Open Recent ▶
Close ⌘W
Save... ⌘S
Duplicate ⇧⌘S
Rename...
Move To...
Revert To ▶
Page Setup... ⇧⌘P
Print... ⌘P

Titlebar Menu

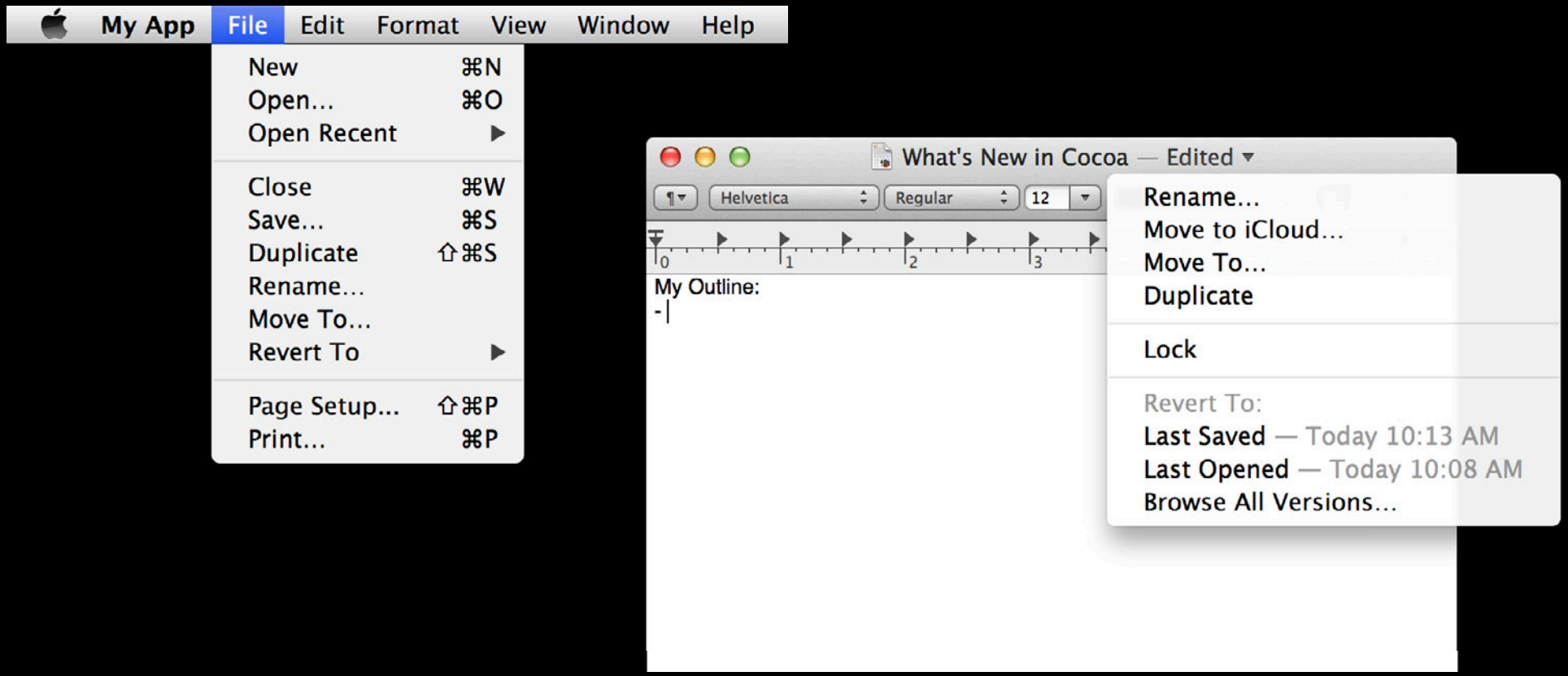


Titlebar Menu

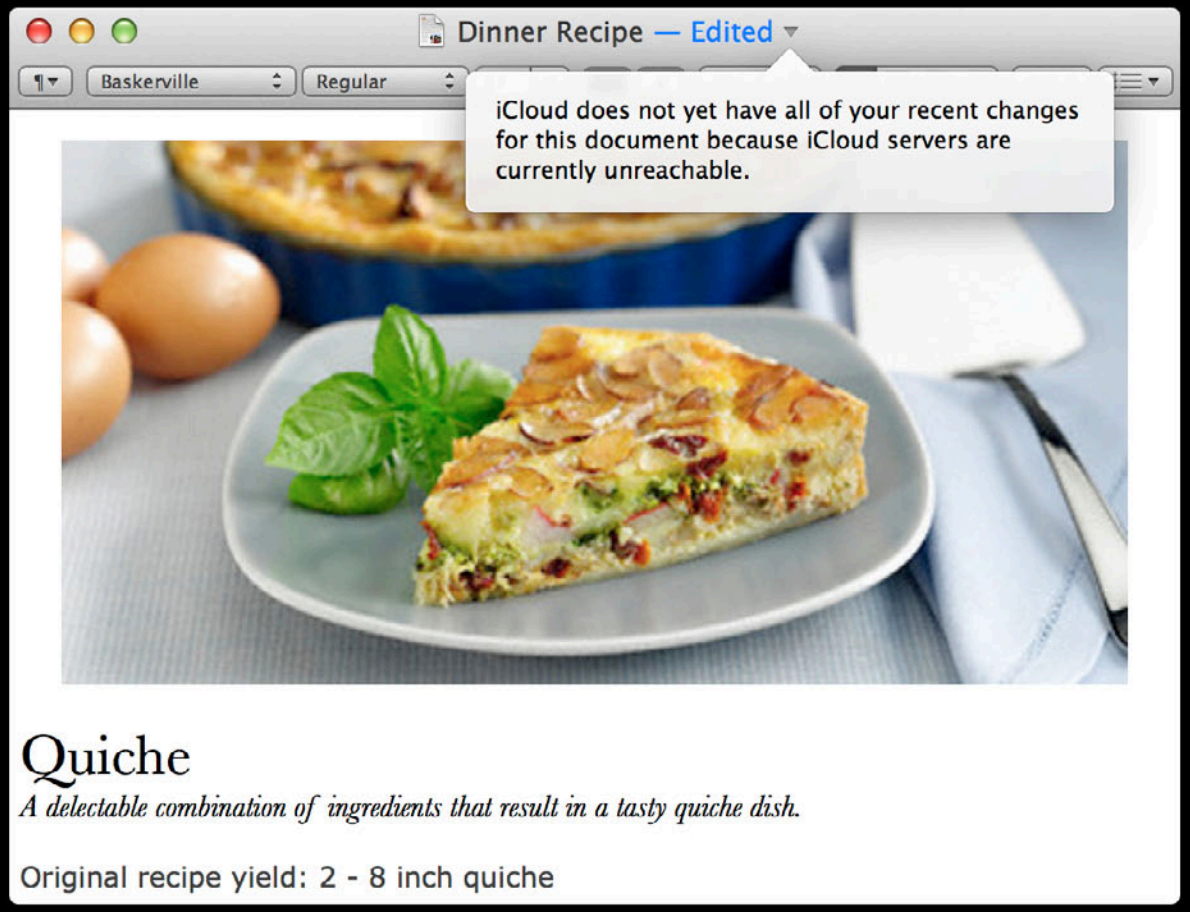




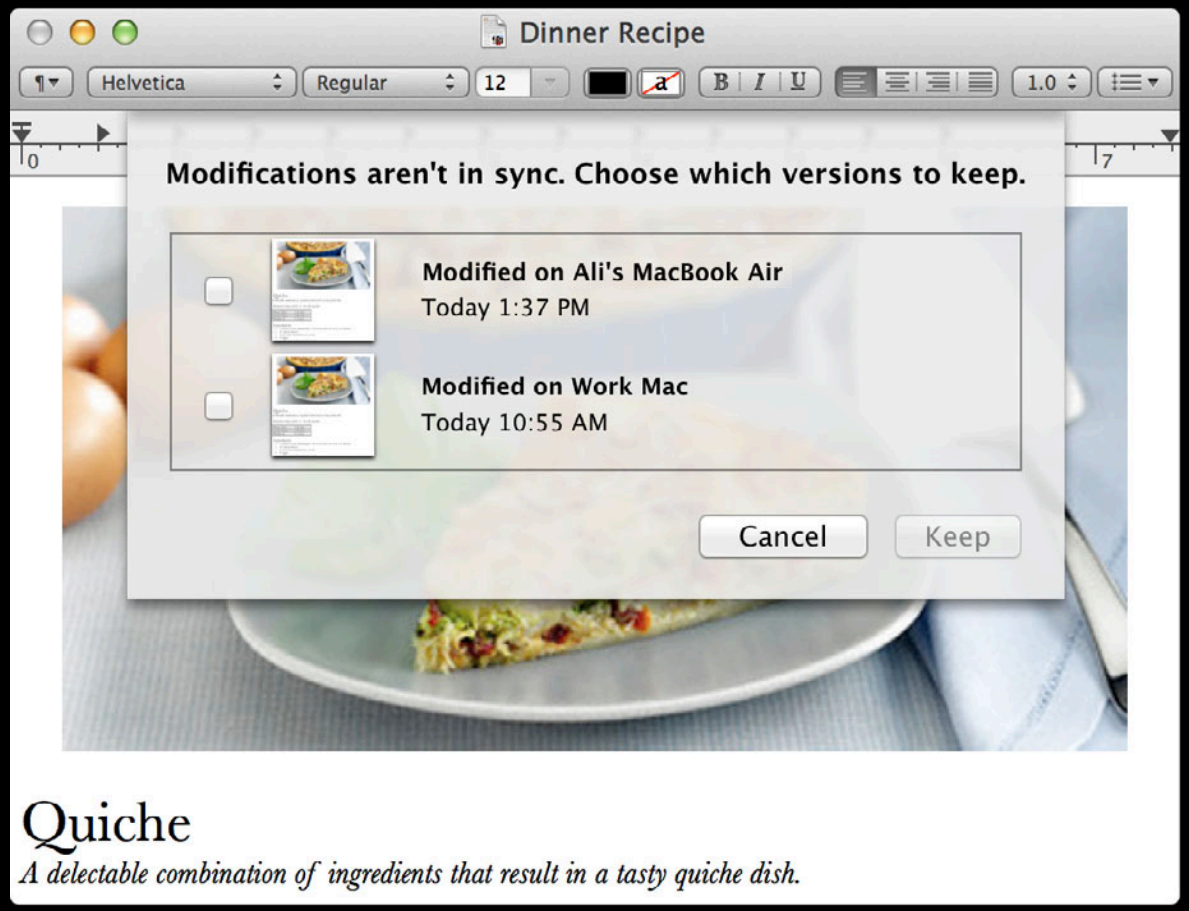
Titlebar Menu



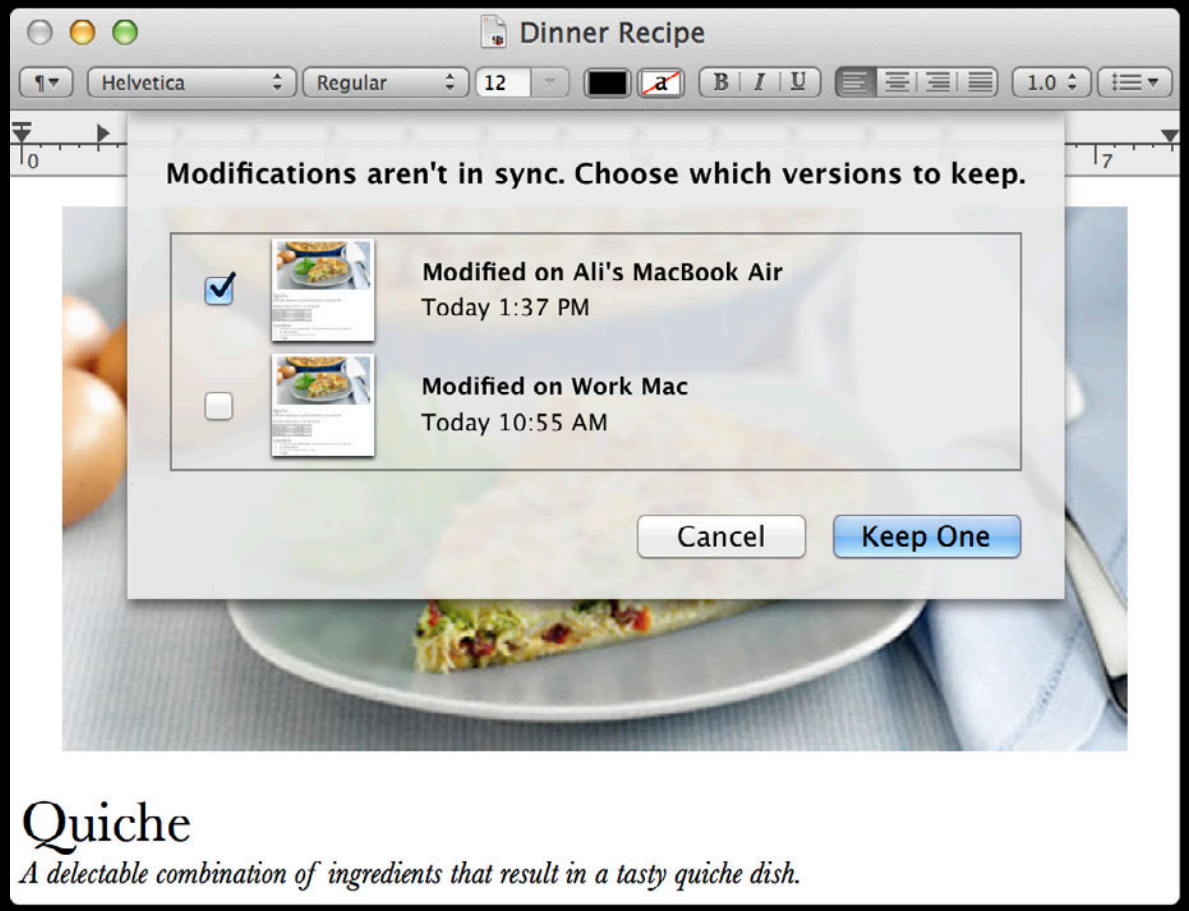
Non-Modal Warnings



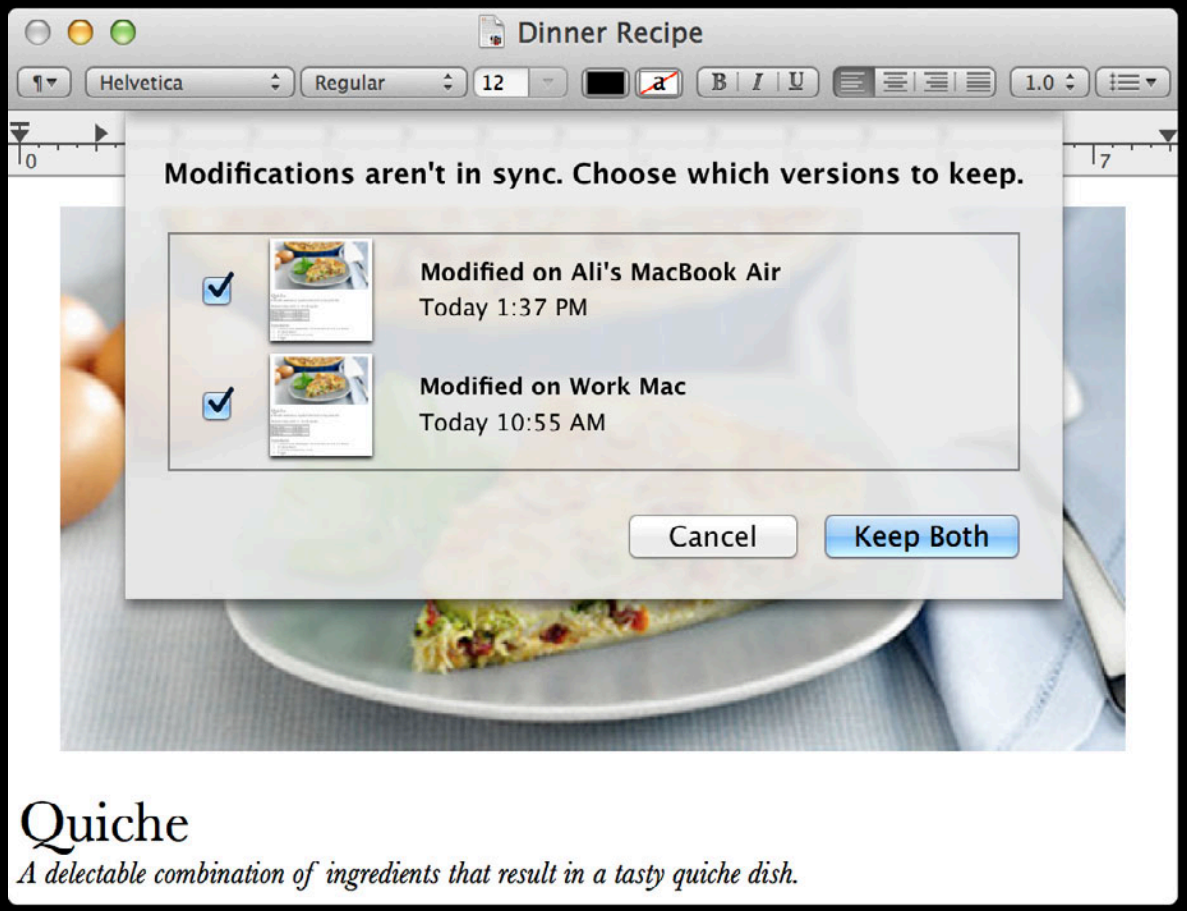
Conflict Resolution



Conflict Resolution



Conflict Resolution



iCloud Adoption with NSDocument

- How much new code do you need to write?

iCloud Adoption with NSDocument

- How much new code do you need to write?
  - Not much

iCloud Adoption with NSDocument

- How much new code do you need to write?
  - Not much
  - Automatic for Auto Save enabled NSDocument based apps with iCloud containers entitlement

Auto Save Recap

Auto Save Recap

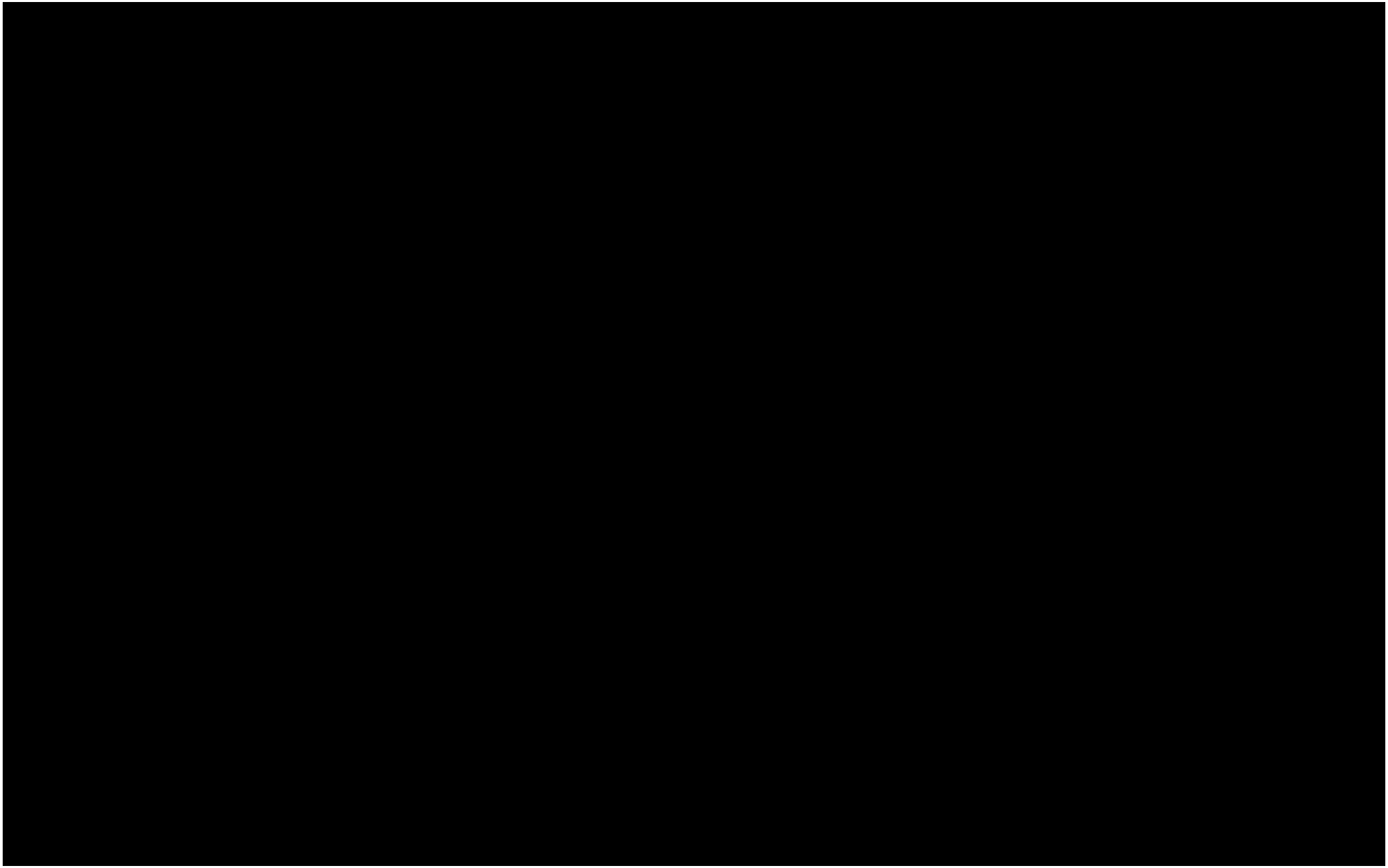
- Eliminate the need for command-s

Auto Save Recap

- Eliminate the need for command-s
- Allow quitting apps, restarting, etc without hassle

Auto Save Recap

- Eliminate the need for command-s
- Allow quitting apps, restarting, etc without hassle
- With iCloud, reduce chance of conflicts



Adopt Auto Save!

Auto Save Recap

- In your NSDocument subclass, implement

```
+ (BOOL)autosavesInPlace {
    return YES;
}
```

Auto Save Recap

- In your NSDocument subclass, implement

```
+ (BOOL)autosavesInPlace {
    return YES;
}
```

- Some other tasks to be aware of
  - Ensure serialization of UI and serialization of file access
  - Save more efficiently
  - Take versions into account

NSFileManager

NSFileManager

- Quick way to check current iCloud user

```
- (id <NSObject, NSCopying, NSCoding>)ubiquityIdentityToken;
```

NSFileManager

- Quick way to check current iCloud user

```
- (id <NSObject, NSCopying, NSCoding>)ubiquityIdentityToken;
```

- Hearing about changes to the iCloud user

```
NSString *NSUbiquityIdentityDidChangeNotification;
```

Related Sessions

Using iCloud with NSDocument	Marina Wednesday 3:15PM
iCloud Storage Overview	Pacific Heights Tuesday 4:30PM
Using iCloud with Core Data	Mission Wednesday 4:30PM
Advanced iCloud Document Storage	Marina Thursday 3:15PM

Labs

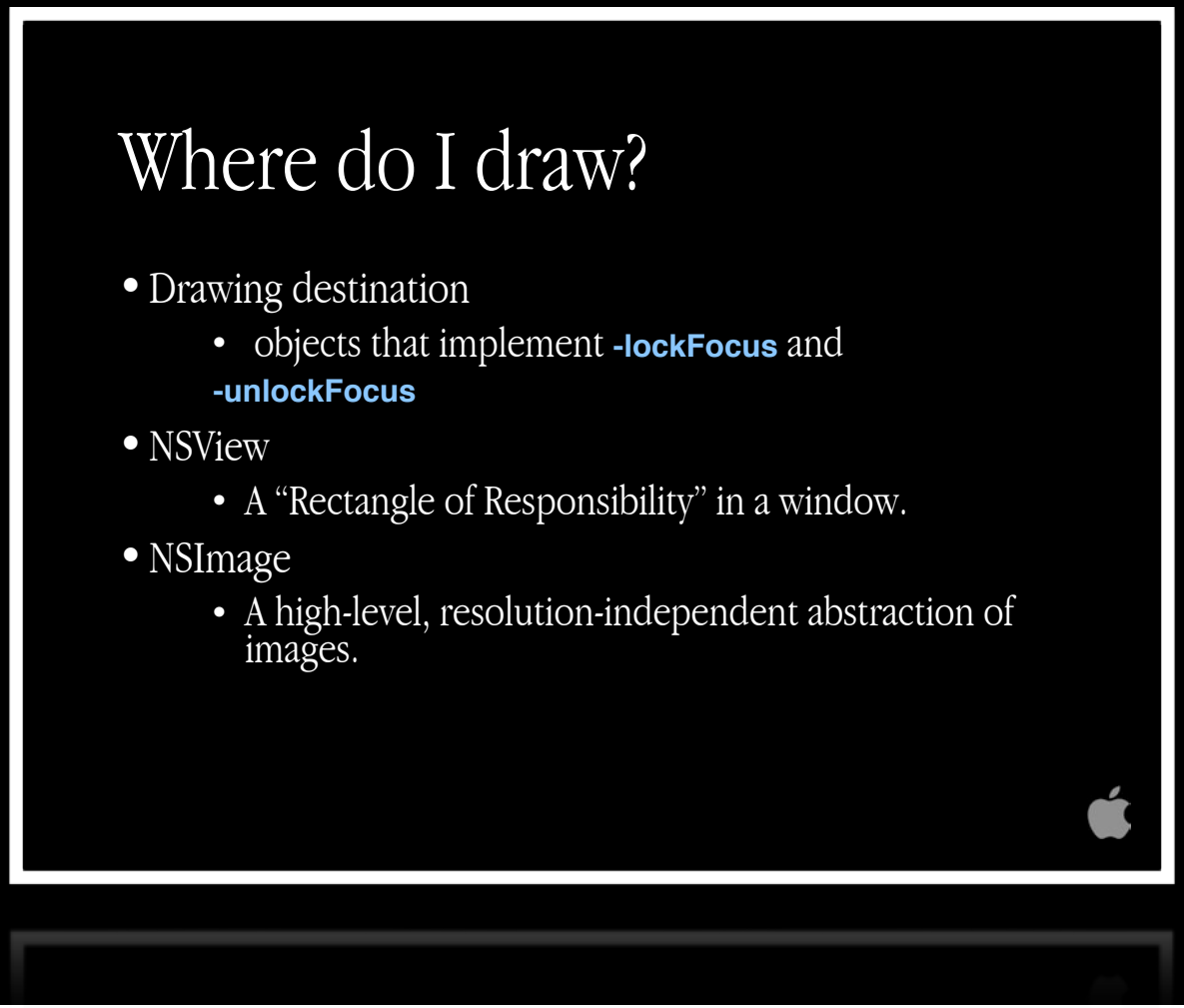
iCloud Storage Lab	Essentials Lab B Tuesday 11:30AM
iCloud Storage Lab	Essentials Lab B Thursday 4:30PM
iCloud Storage Lab	Essentials Lab B Friday 11:30AM

High Resolution

This time it's for real

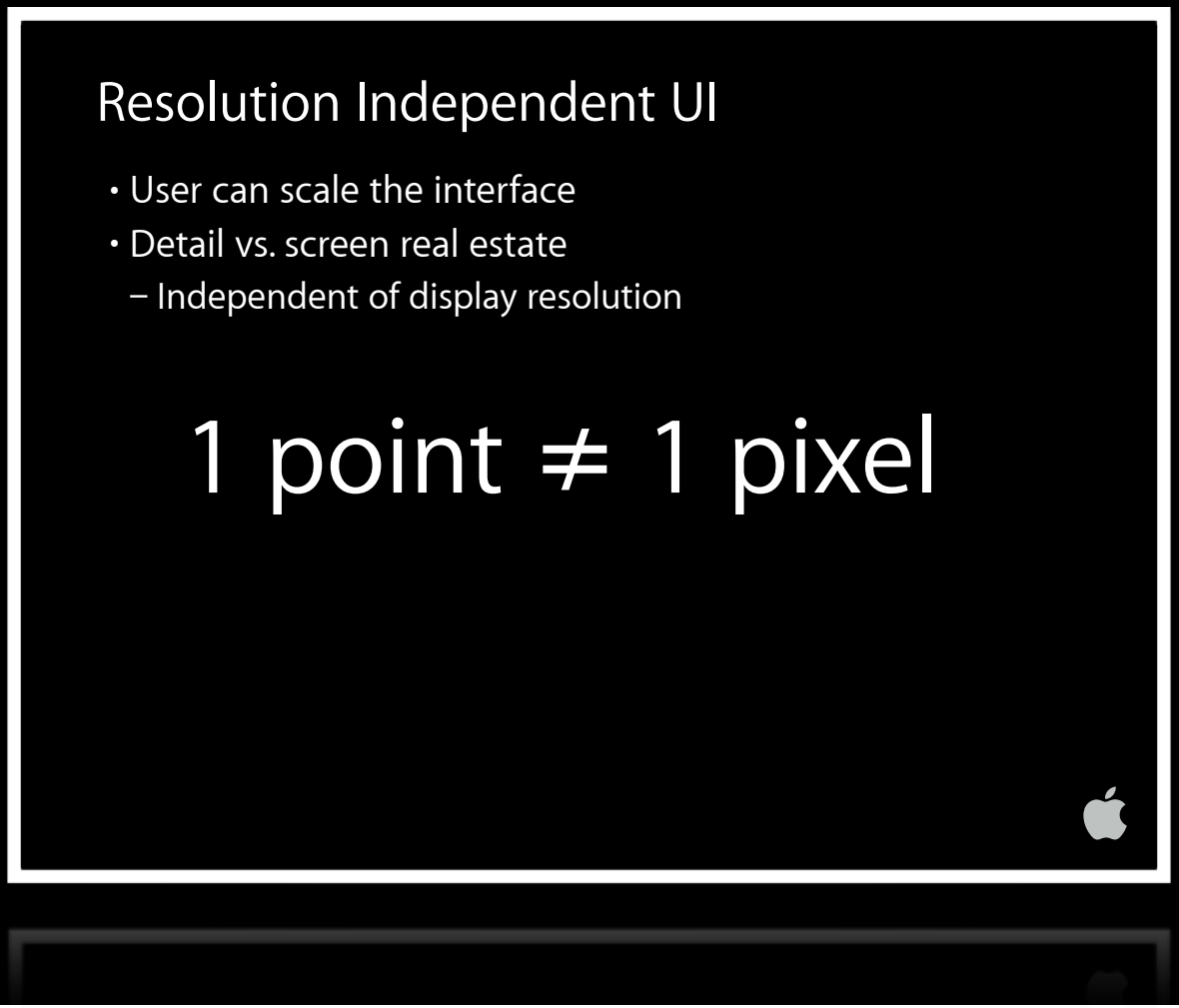
High Resolution “Time Machine”

High Resolution "Time Machine"



2002

High Resolution "Time Machine"

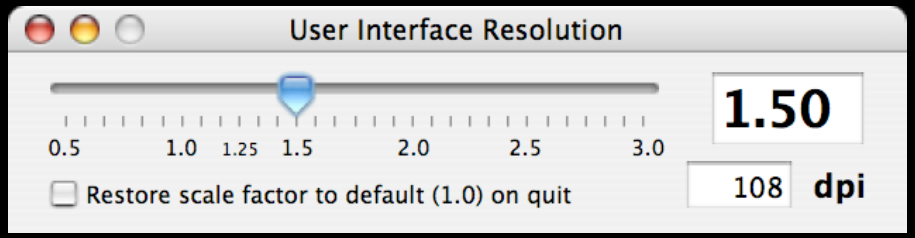


2004

High Resolution "Time Machine"

Resolution Independent UI

- Quartz Debug resolution window



- Scale factor is pixels-per-point
- Applies to applications launched after the setting change

2004

High Resolution "Time Machine"

Stuff You Should Not Be Doing

- Assuming sizeof(int) == sizeof(void *)
  - 64-bit
- Assuming 1 pixel == 1 point
  - Resolution independence
- Assuming APIs return things other than what they say
  - NSUserDefaults change
- Drawing faster than 60 fps
  - Coalesced updates
- Using private APIs

2005

High Resolution "Time Machine"

Resolution Independence and You

- You need high resolution artwork for a crisp interface

- Where possible, use vector based art
- Otherwise, supply images at 1x and 4x in multi-representation tiff
  - Set DPI appropriately so they are the same real size

```
NSImage 0x36b530 Name=Camera Size={100, 80} Reps=(
    NSBitmapImageRep 0x36acf0 Size={100, 80} Pixels=100x80
    NSBitmapImageRep 0x36be60 Size={100, 80} Pixels=400x320
)
```

2006

High Resolution “Time Machine”

2007

High Resolution "Time Machine"

When?

- Be ready by 2008
- ~ 130 DPI 17-inch MacBook Pro available today
- When else would you want resolution independence?
  - When you'd like a crisp look for larger UI
  - Use the full-screen resolution
  - Demos
  - Interact with your display more closely

2007

High Resolution “Time Machine”

2008

High Resolution “Time Machine”

2009

High Resolution “Time Machine”

2010

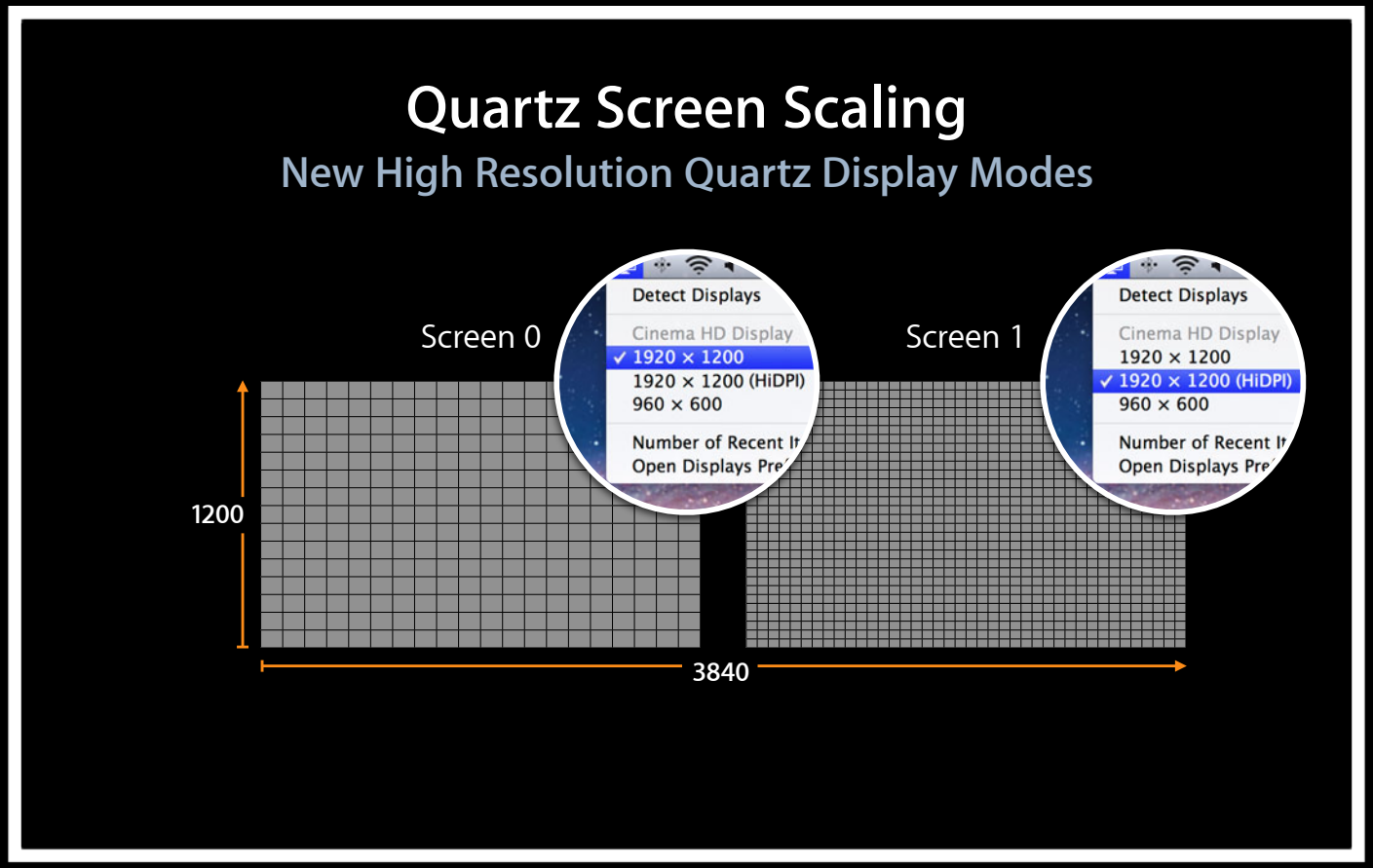
High Resolution “Time Machine”

Resolution Independence

Architecture Update for Mac OS X Lion

2011

High Resolution “Time Machine”



2011

High Resolution "Time Machine"

Resolution Independence

State of the Pixel Union

- LCD display technology is pervasive
- Benefiting from Moore's Law applied to displays
- The future of the pixel is small
- Challenge for software compatibility
- Back to the Mac!

2011

High Resolution “Time Machine”

2012

High Resolution “Time Machine”

2012

High Resolution

Integral 2x scale factor

High Resolution

Integral 2x scale factor

- By default existing Cocoa apps run at 2x
  - Text, paths, PDFs, etc all in high resolution
  - Apps need to provide 2x bitmap artwork

High Resolution

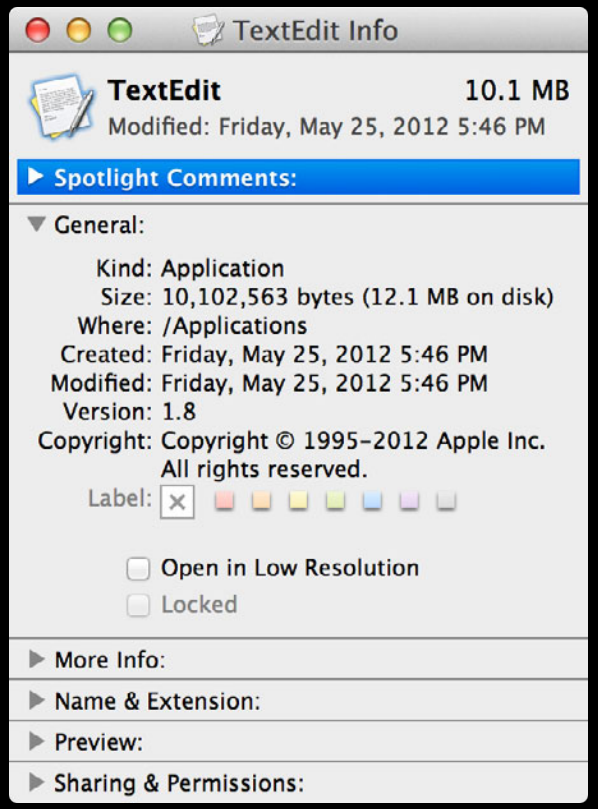
Integral 2x scale factor

- By default existing Cocoa apps run at 2x
  - Text, paths, PDFs, etc all in high resolution
  - Apps need to provide 2x bitmap artwork
- “Low Resolution” mode
  - Apps run at 1x
  - Magnified to 2x on screen
  - Default for Carbon and some Cocoa apps
  - User selectable

High Resolution

Integral 2x scale factor

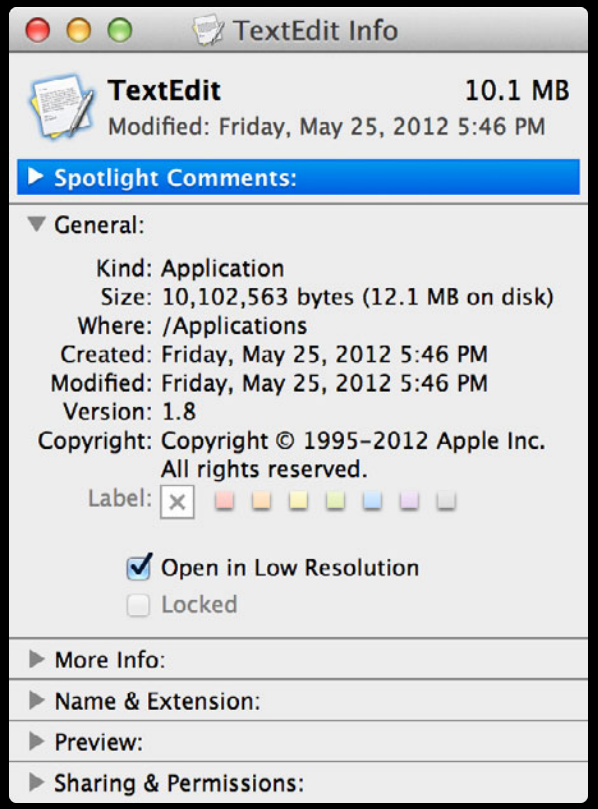
- By default existing Cocoa apps run at 2x
  - Text, paths, PDFs, etc all in high resolution
  - Apps need to provide 2x bitmap artwork
- "Low Resolution" mode
  - Apps run at 1x
  - Magnified to 2x on screen
  - Default for Carbon and some Cocoa apps
  - User selectable



High Resolution

Integral 2x scale factor

- By default existing Cocoa apps run at 2x
  - Text, paths, PDFs, etc all in high resolution
  - Apps need to provide 2x bitmap artwork
- "Low Resolution" mode
  - Apps run at 1x
  - Magnified to 2x on screen
  - Default for Carbon and some Cocoa apps
  - User selectable



High Resolution

Automatic behaviors of AppKit classes

High Resolution

Automatic behaviors of AppKit classes

- NSImage
  - Support for multiple representations

High Resolution

Automatic behaviors of AppKit classes

- NSImage
  - Support for multiple representations
- NSView
  - Automatic scaling of layer backing

High Resolution

Automatic behaviors of AppKit classes

- NSImage
  - Support for multiple representations
- NSView
  - Automatic scaling of layer backing
- NSTextView
  - Fractional advances by default for apps linked against 10.8 SDK

High Resolution

Automatic behaviors of AppKit classes

- NSImage
  - Support for multiple representations
- NSView
  - Automatic scaling of layer backing
- NSTextView
  - Fractional advances by default for apps linked against 10.8 SDK
- NSColor

High Resolution

Automatic behaviors of AppKit classes

- NSImage
  - Support for multiple representations
- NSView
  - Automatic scaling of layer backing
- NSTextView
  - Fractional advances by default for apps linked against 10.8 SDK
- NSColor
- NSShadow

High Resolution

Automatic behaviors of AppKit classes

- NSImage
  - Support for multiple representations
- NSView
  - Automatic scaling of layer backing
- NSTextView
  - Fractional advances by default for apps linked against 10.8 SDK
- NSColor
- NSShadow
- Auto Layout

Block-Backed NSImage

Block-Backed NSImage

- NSImage lockFocus draws in a single bitmap representation
  - Not appropriate for different contexts

Block-Backed NSImage

- NSImage lockFocus draws in a single bitmap representation
  - Not appropriate for different contexts
- Mountain Lion brings a dynamically callable version

```
@interface NSImage
+ (id)imageWithSize:(NSSize)size
            flipped:(BOOL)drawingHandlerShouldBeCalledWithFlippedContext
     drawingHandler:(BOOL (^)(NSRect dstRect))drawingHandler;
@end
```

Block-Backed NSImage

Block-Backed NSImage

- Lion

```
NSImage *myImage = [[NSImage alloc] initWithSize:rect.size];
[myImage lockFocus];
[[NSColor redColor] set];
[[NSBezierPath bezierPathWithOvalInRect:rect] fill];
[myImage unlockFocus];
```

Block-Backed NSImage

- Lion

```
NSImage *myImage = [[NSImage alloc] initWithSize:rect.size];
[myImage lockFocus];
[[NSColor redColor] set];
[[NSBezierPath bezierPathWithOvalInRect:rect] fill];
[myImage unlockFocus];
```

Block-Backed NSImage

- Lion

```
NSImage *myImage = [[NSImage alloc] initWithSize:rect.size];
[myImage lockFocus];
[[NSColor redColor] set];
[[NSBezierPath bezierPathWithOvalInRect:rect] fill];
[myImage unlockFocus];
```

Block-Backed NSImage

- Lion

```
NSImage *myImage = [[NSImage alloc] initWithSize:rect.size];
[myImage lockFocus];
[[NSColor redColor] set];
[[NSBezierPath bezierPathWithOvalInRect:rect] fill];
[myImage unlockFocus];
```

Block-Backed NSImage

- Lion

```
NSImage *myImage = [[NSImage alloc] initWithSize:rect.size];
[myImage lockFocus];
[[NSColor redColor] set];
[[NSBezierPath bezierPathWithOvalInRect:rect] fill];
[myImage unlockFocus];
```

Block-Backed NSImage

- Lion

```
NSImage *myImage = [[NSImage alloc] initWithSize:rect.size];
[myImage lockFocus];
[[NSColor redColor] set];
[[NSBezierPath bezierPathWithOvalInRect:rect] fill];
[myImage unlockFocus];
```

- Mountain Lion

```
NSImage *myImage = [NSImage imageWithSize:rect.size
                                  flipped:NO
                           drawingHandler:^(NSRect dstRect) {
                [[NSColor redColor] set];
                [[NSBezierPath bezierPathWithOvalInRect:dstRect] fill];
                return YES;
            }];
```

Block-Backed NSImage

- Lion

```
NSImage *myImage = [[NSImage alloc] initWithSize:rect.size];
[myImage lockFocus];
[[NSColor redColor] set];
[[NSBezierPath bezierPathWithOvalInRect:rect] fill];
[myImage unlockFocus];
```

- Mountain Lion

```
NSImage *myImage = [NSImage imageWithSize:rect.size
                                  flipped:NO
                           drawingHandler:^(NSRect dstRect) {
                [[NSColor redColor] set];
                [[NSBezierPath bezierPathWithOvalInRect:dstRect] fill];
                return YES;
            }];
```

Block-Backed NSImage

- Lion

```
NSImage *myImage = [[NSImage alloc] initWithSize:rect.size];
[myImage lockFocus];
[[NSColor redColor] set];
[[NSBezierPath bezierPathWithOvalInRect:rect] fill];
[myImage unlockFocus];
```

- Mountain Lion

```
NSImage *myImage = [NSImage imageWithSize:rect.size
                                  flipped:NO
                           drawingHandler:^(NSRect dstRect) {
                [[NSColor redColor] set];
                [[NSBezierPath bezierPathWithOvalInRect:dstRect] fill];
                return YES;
            }];
```

Block-Backed NSImage

- Lion

```
NSImage *myImage = [[NSImage alloc] initWithSize:rect.size];
[myImage lockFocus];
[[NSColor redColor] set];
[[NSBezierPath bezierPathWithOvalInRect:rect] fill];
[myImage unlockFocus];
```

- Mountain Lion

```
NSImage *myImage = [NSImage imageWithSize:rect.size
                                  flipped:NO
                           drawingHandler:^(NSRect dstRect) {
                [[NSColor redColor] set];
                [[NSBezierPath bezierPathWithOvalInRect:dstRect] fill];
                return YES;
            }];
```

High Resolution

- You can enable 2x modes on your Mac right now
  - Don’t need to wait for the delivery of your new MacBook Pro

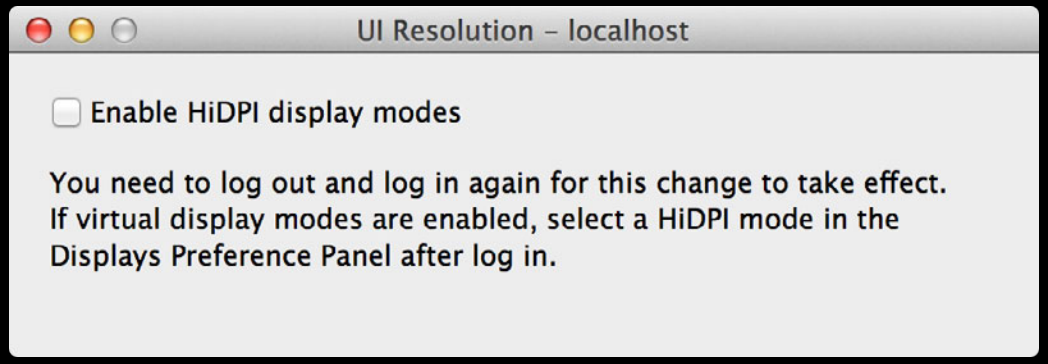
Enabling 2x Modes

- Download “Graphics Tools for Xcode”
- Open Quartz Debug
- Choose “UI Resolution” and enable HiDPI display modes

- Select a HiDPI mode in System Preferences

Enabling 2x Modes

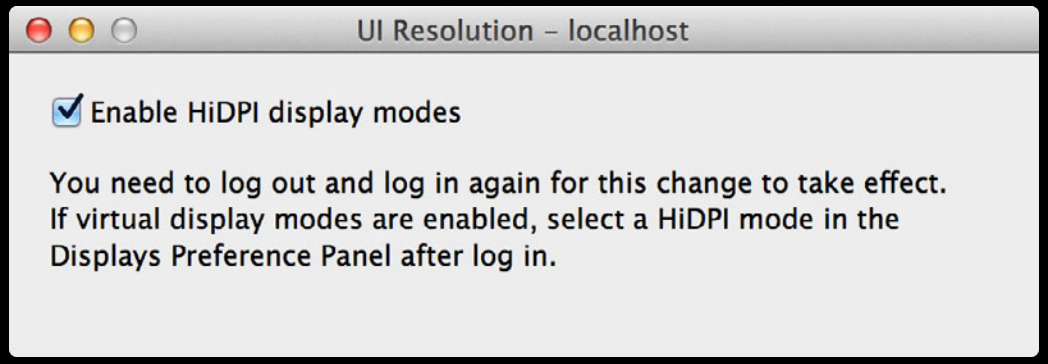
- Download "Graphics Tools for Xcode"
- Open Quartz Debug
- Choose "UI Resolution" and enable HiDPI display modes



- Select a HiDPI mode in System Preferences

Enabling 2x Modes

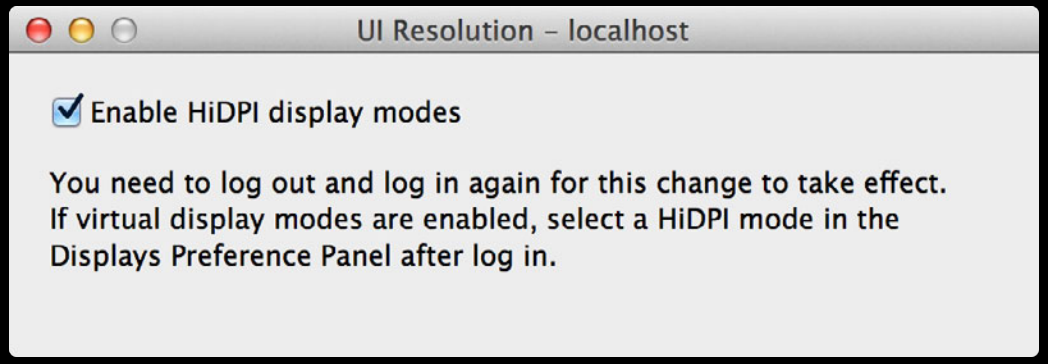
- Download "Graphics Tools for Xcode"
- Open Quartz Debug
- Choose "UI Resolution" and enable HiDPI display modes



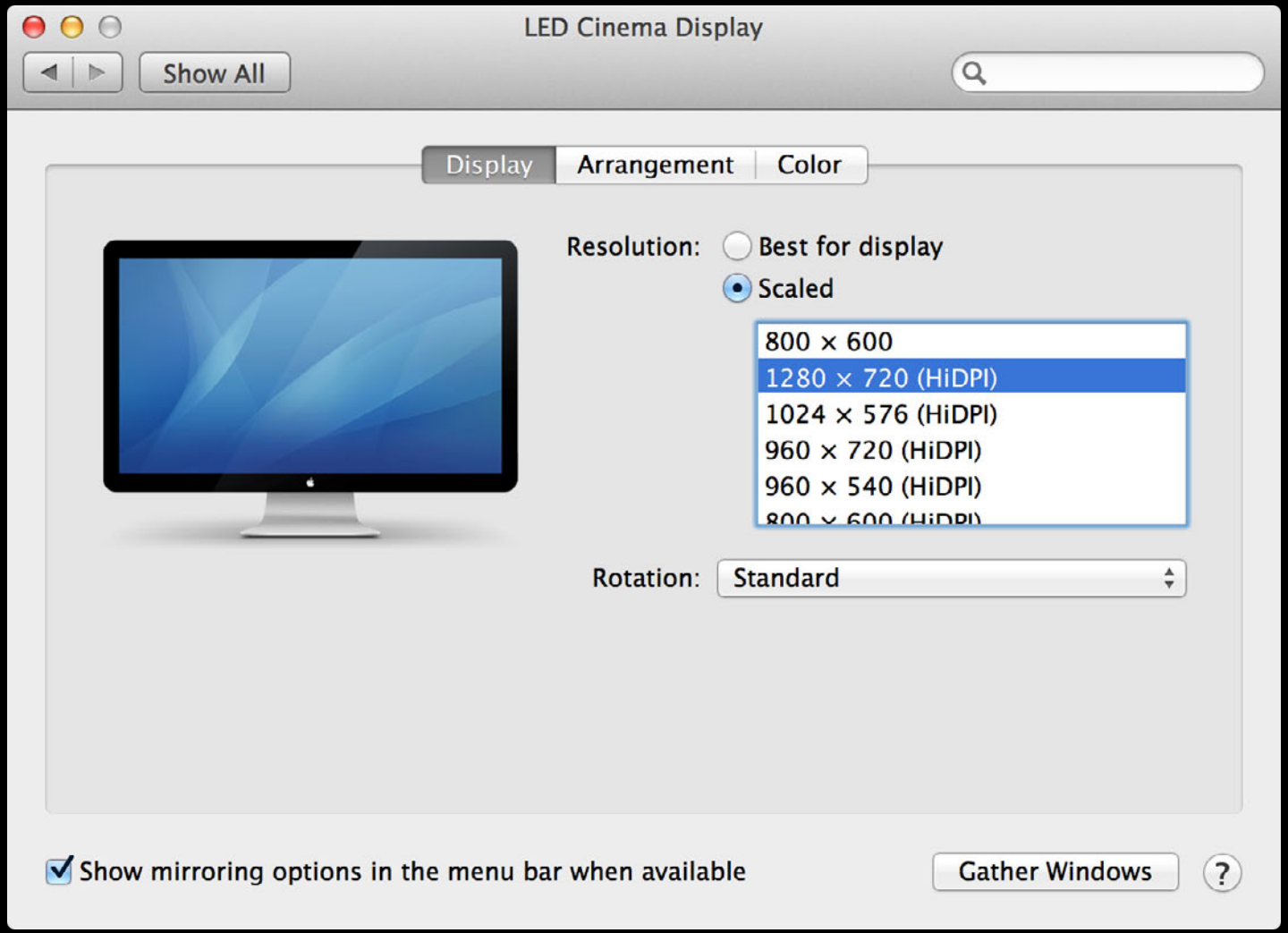
- Select a HiDPI mode in System Preferences

Enabling 2x Modes

- Download "Graphics Tools for Xcode"
- Open Quartz Debug
- Choose "UI Resolution" and enable HiDPI display modes



- Select a HiDPI mode in System Preferences


LED Cinema Display
Show All
Display
Arrangement
Color
Resolution:
Best for display
Scaled
800 × 600
1280 × 720 (HiDPI)
1024 × 576 (HiDPI)
960 × 720 (HiDPI)
960 × 540 (HiDPI)
Rotation:
Standard
Show mirroring options in the menu bar when available
Gather Windows
?

Related Sessions

Introduction to High Resolution on OS X	Presidio Wednesday 9:00AM
Advanced Tips & Tricks for High Resolution on OS X	Mission Friday 10:15AM

Labs

High Resolution on OS X Lab	Essentials Lab B Wednesday 11:30AM

Gestures

NSScrollView

Automatic behavior changes

NSScrollView

Automatic behavior changes

- Accelerated scrolling

NSScrollView

Automatic behavior changes

- Accelerated scrolling
- New overlay scroller look and behaviors

NSScrollView

Automatic behavior changes

- Accelerated scrolling
- New overlay scroller look and behaviors
- Smooth scrolling on by default

NSScrollView Magnification

Pinch in/out

NSScrollView Magnification

Pinch in/out

- Allows user to magnify the view

```
@property BOOL allowsMagnification;
```

NSScrollView Magnification

Pinch in/out

- Allows user to magnify the view

```
@property BOOL allowsMagnification;
```

- Parameters

```
@property CGFloat magnification;
@property CGFloat maxMagnification;
@property CGFloat minMagnification;
```

NSScrollView Magnification

Pinch in/out

- Allows user to magnify the view

```
@property BOOL allowsMagnification;
```

- Parameters

```
@property CGFloat magnification;
@property CGFloat maxMagnification;
@property CGFloat minMagnification;
```

- Conveniences

```
- (void)magnifyToFitRect:(NSRect)rect;
- (void)setMagnification:(CGFloat)mag centeredAtPoint:(NSPoint)point;
```

Animation Recap

Animation Recap

```
@protocol NSAnimatablePropertyContainer
- (id)animator;
...
@end
```

Animation Recap

```
@protocol NSAnimatablePropertyContainer
- (id)animator;
...
@end

@interface NSView : NSResponder <NSAnimatablePropertyContainer, ...>
...
@end
```

Animation Recap

- No animation

```
[myScrollView setMagnification:2.0];
```

Animation Recap

- No animation

```
[myScrollView setMagnification:2.0];
```

Animation Recap

- No animation

```
[myScrollView setMagnification:2.0];
```

- To animate talk to the animator

```
[[myScrollView animator] setMagnification:2.0];
```

Animation Recap

- No animation

```
[myScrollView setMagnification:2.0];
```

- To animate talk to the animator

```
[[myScrollView animator] setMagnification:2.0];

[[myScrollView animator] magnifyToFitRect:newRect];
```

Smart Magnification

Two-finger double-tap

Smart Magnification

Two-finger double-tap

- NSScrollView
  - If at 100%, magnify to 150%
  - Otherwise go to 100%

Smart Magnification

Two-finger double-tap

- NSScrollView
  - If at 100%, magnify to 150%
  - Otherwise go to 100%
- NSTextView
  - Smart magnify image attachments, table cells

Smart Magnification

Two-finger double-tap

- NSScrollView
  - If at 100%, magnify to 150%
  - Otherwise go to 100%
- NSTextView
  - Smart magnify image attachments, table cells
- Customize
  - Listen to event or NSResponder method
  - Or override NSView method

```
- (NSRect)rectForSmartMagnificationAtPoint:(NSPoint)location
                                   inRect:(NSRect)visibleRect;
```

Quick Look

Three-finger tap

Quick Look

Three-finger tap

- New responder methods

```
- (void)quickLookPreviewItems:(id)sender;
- (void)quickLookWithEvent:(NSEvent *)event;
```

Quick Look

Three-finger tap

- New responder methods

```
- (void)quickLookPreviewItems:(id)sender;
- (void)quickLookWithEvent:(NSEvent *)event;
```

- Implemented by NSTextView

Quick Look

Three-finger tap

- New responder methods

```
- (void)quickLookPreviewItems:(id)sender;
- (void)quickLookWithEvent:(NSEvent *)event;
```

- Implemented by NSTextView
- By default NSApplication performs a dictionary lookup

NSPageController

- New class for efficient swipe-navigation of views

NSPageController

- New class for efficient swipe-navigation of views
- Three transition styles

NSPageController

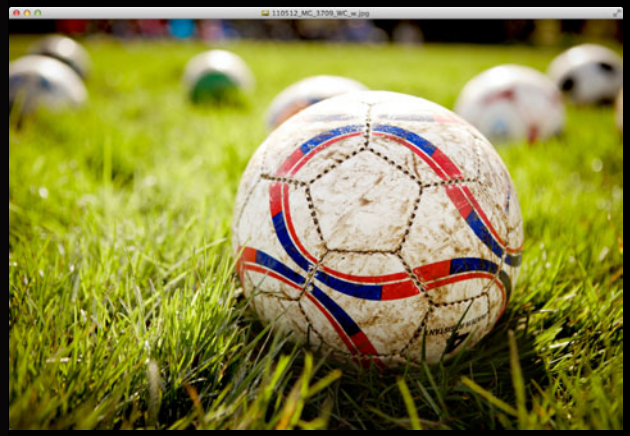
- New class for efficient swipe-navigation of views
- Three transition styles
  - History

NSPageController

- New class for efficient swipe-navigation of views
- Three transition styles
  - History

NSPageController

- New class for efficient swipe-navigation of views
- Three transition styles
  - History
  - Book



NSPageController

- New class for efficient swipe-navigation of views
- Three transition styles
  - History
  - Book

NSPageController

- New class for efficient swipe-navigation of views
- Three transition styles
  - History
  - Book
  - Strip

NSPageController

- New class for efficient swipe-navigation of views
- Three transition styles
  - History
  - Book
  - Strip

NSPageController

NSPageController

- Manages
  - A predefined array of content

```
@property(copy) NSArray *arrangedObjects;
@property NSInteger selectedIndex;
```

NSPageController

- Manages
  - A predefined array of content

```
@property(copy) NSArray *arrangedObjects;
@property NSInteger selectedIndex;
```

  - Or content created from navigation history

```
- (void)navigateForwardToObject:(id)object;
```

NSPageController

- Manages
  - A predefined array of content

```
@property(copy) NSArray *arrangedObjects;
@property NSInteger selectedIndex;
```

  - Or content created from navigation history

```
- (void)navigateForwardToObject:(id)object;
```

What's New with Gestures

Marina
Thursday 10:15AM

NSPageController

- Manages
  - A predefined array of content

```
@property(copy) NSArray *arrangedObjects;
@property NSInteger selectedIndex;
```

  - Or content created from navigation history

```
- (void)navigateForwardToObject:(id)object;
```

What's New with Gestures	Marina Thursday 10:15AM
OS X Gestures and Cocoa Lab	Essentials Lab B Thursday 2:00PM

Layer-Backed Views

Layer-Backed Views Recap

Layer-Backed Views Recap

- Make a view layer-backed with

```
[view setWantsLayer:YES];
```

Layer-Backed Views Recap

- Make a view layer-backed with

```
[view setWantsLayer:YES];
```

- This creates a Core Animation layer backing store for the view and subviews
  - Smoother animations
  - Better responsiveness

Layer-Backed Views Recap

- Make a view layer-backed with

```
[view setWantsLayer:YES];
```

- This creates a Core Animation layer backing store for the view and subviews
  - Smoother animations
  - Better responsiveness
- But drawRect: called as before
  - As controlled by layerContentsRedrawPolicy

Layer-Backed Views

Layer-Backed Views

- New APIs to update layer contents declaratively

Layer-Backed Views

- New APIs to update layer contents declaratively
- Override

```
- (BOOL)wantsUpdateLayer {
      return YES;
}
```

Layer-Backed Views

- New APIs to update layer contents declaratively
- Override

```
- (BOOL)wantsUpdateLayer {
    return YES;
}
```

- On setNeedsDisplay:, this causes -updateLayer to be invoked

```
- (void)updateLayer {
    ...
}
```

Layer-Backed Views

- New APIs to update layer contents declaratively
- Override

```
- (BOOL)wantsUpdateLayer {
    return YES;
}
```

- On setNeedsDisplay:, this causes -updateLayer to be invoked

```
- (void)updateLayer {
    ...
}
```

- Allows animations on background threads

Layer-Backed Views

- New APIs to update layer contents declaratively
- Override

```
- (BOOL)wantsUpdateLayer {
    return YES;
}
```

- On setNeedsDisplay:, this causes -updateLayer to be invoked

```
- (void)updateLayer {
    ...
}
```

- Allows animations on background threads
- Enables memory saving

Layer-Backed Views

- Traditional

```
- (void)drawRect:(NSRect)updateRect {
    [NSColor.redColor set];
    [NSBezierPath fillRect:updateRect];
}
```

Layer-Backed Views

- Traditional

```
- (void)drawRect:(NSRect)updateRect {
    [NSColor.redColor set];
    [NSBezierPath fillRect:updateRect];
}
```

- New

```
- (void)updateLayer {
    self.layer.backgroundColor = NSColor.redColor.CGColor;
}
```

Layer-Backed Views

- Traditional

```
- (void)drawRect:(NSRect)updateRect {
    [NSColor.redColor set];
    [NSBezierPath fillRect:updateRect];
}
```

- New

```
- (void)updateLayer {
    self.layer.backgroundColor = NSColor.redColor.CGColor;
}
```

- Or with image contents

```
- (void)updateLayer {
    self.layer.contents = [NSImage imageWith...];
}
```

```
NSColor.redColor.CGColor
```

```
NSColor.redColor.CGColor
```

```
NSColor.redColor.CGColor

[[NSColor redColor] CGColor]
```

NSColor

- New API to return an CGColor

```
- (CGColorRef)CGColor;
```

NSColor

- New API to return an CGColor

```
- (CGColorRef)CGColor;
```

- Also create NSColor from CGColor

```
+ (NSColor *)colorWithCGColor:(CGColorRef)cgColor;
```

Layer-Backed Views

Many other improvements

Layer-Backed Views

Many other improvements

- Improved sub-pixel anti-aliasing handling with layer-backed text

Layer-Backed Views

Many other improvements

- Improved sub-pixel anti-aliasing handling with layer-backed text
- Better management of flippedness

Layer-Backed Views

Many other improvements

- Improved sub-pixel anti-aliasing handling with layer-backed text
- Better management of flippedness
- Nested animations

Layer-Backed Views

Many other improvements

- Improved sub-pixel anti-aliasing handling with layer-backed text
- Better management of flippedness
- Nested animations
- And more

Layer-Backed Views

Many other improvements

- Improved sub-pixel anti-aliasing handling with layer-backed text
- Better management of flippedness
- Nested animations
- And more

Layer-Backed Views: AppKit + Core Animation	Nob Hill Wednesday 10:15AM

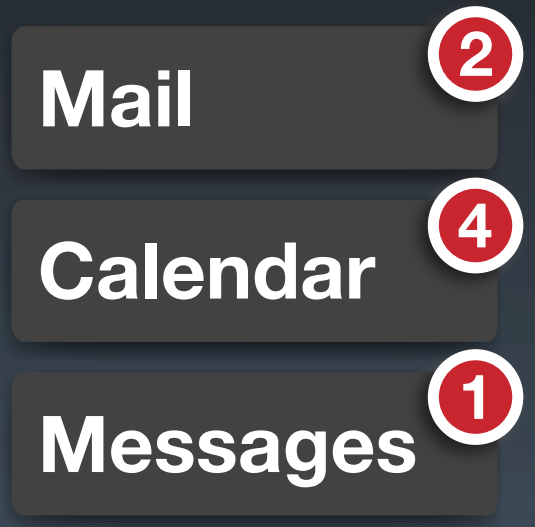
Layer-Backed Views

Many other improvements

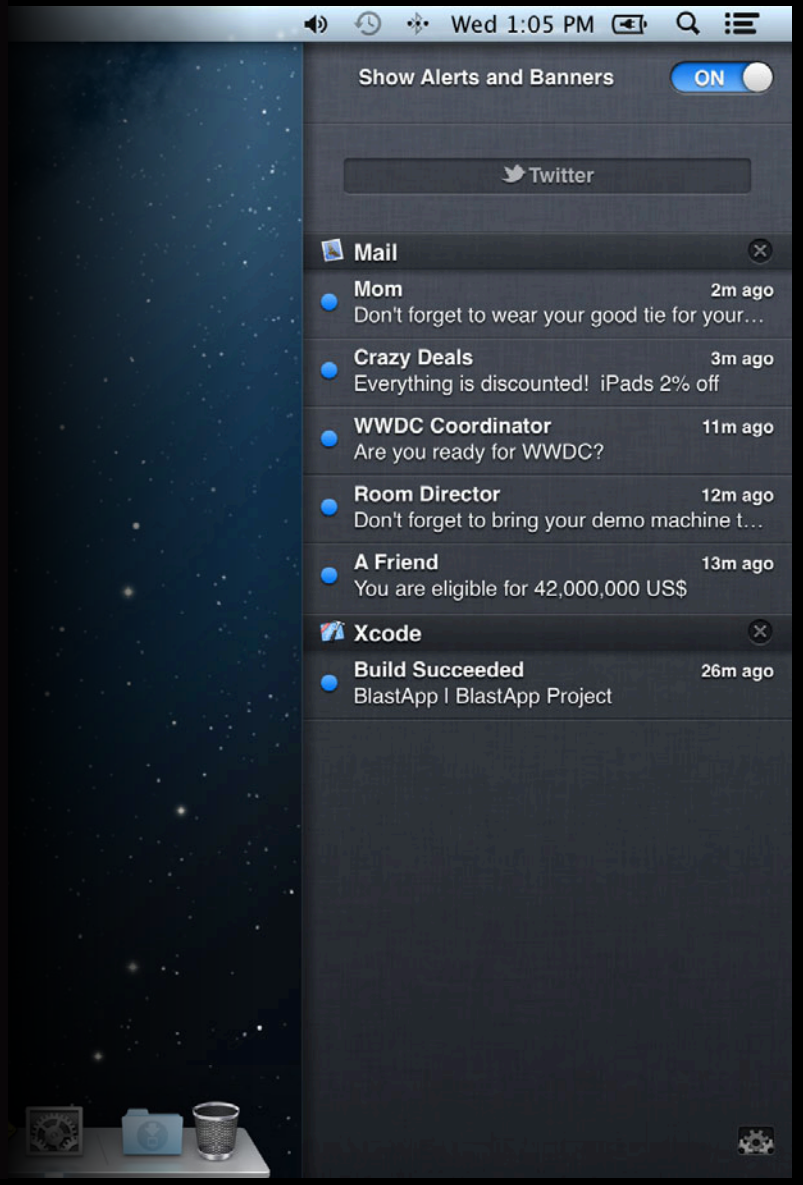
- Improved sub-pixel anti-aliasing handling with layer-backed text
- Better management of flippedness
- Nested animations
- And more

Layer-Backed Views: AppKit + Core Animation	Nob Hill Wednesday 10:15AM
Cocoa and Layer-Backed Views on OS X Lab	Essentials Lab B Wednesday 2:00PM

Notification Center

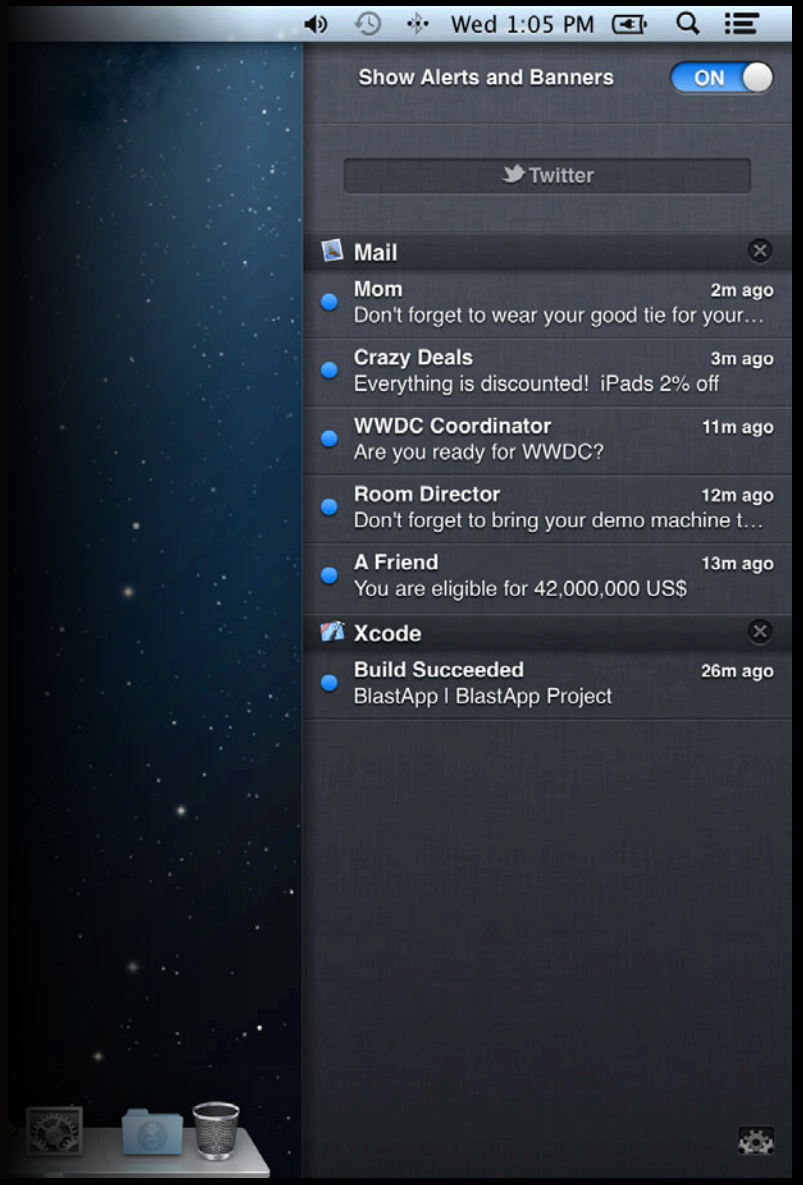


NSUserNotification



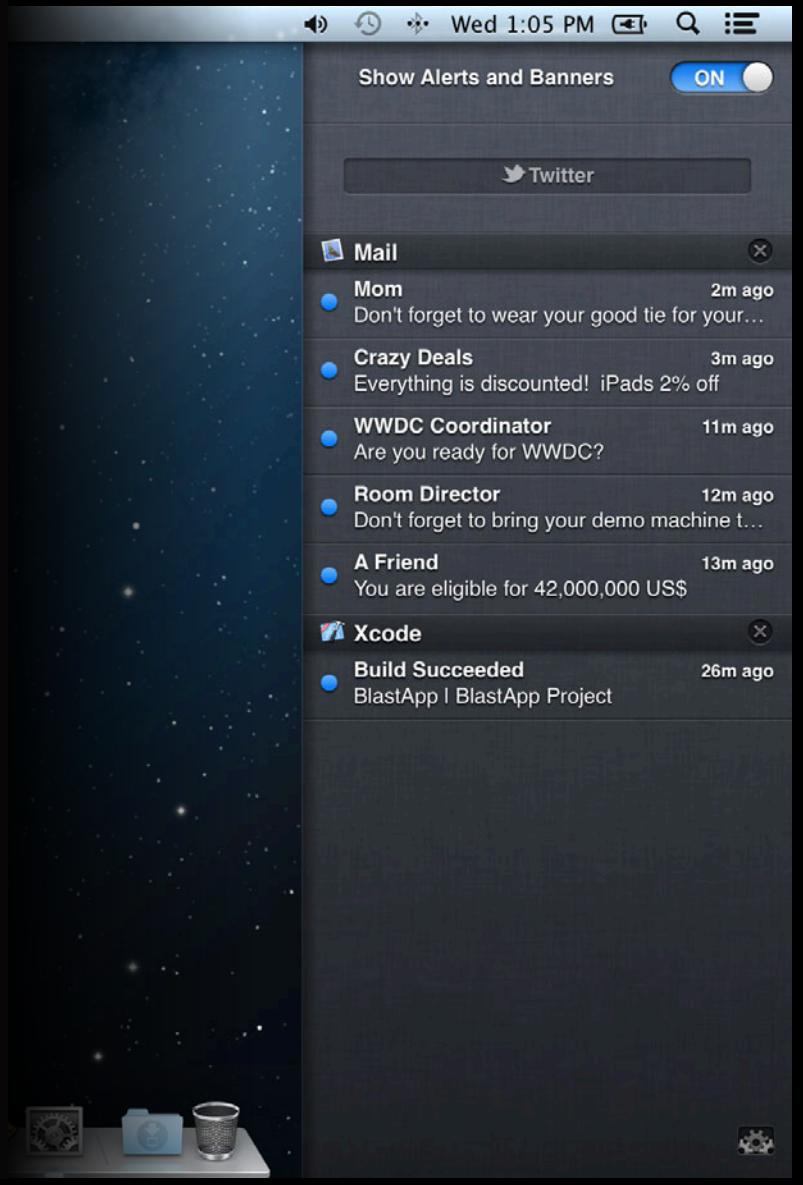
NSUserNotification

- API to the new notification center



NSUserNotification

- API to the new notification center
- Not to be confused with
  - NSNotification
  - CFNotification
  - Or CFUserNotification



NSUserNotification

- NSUserNotification
  - Has properties such as title, informativeText, deliveryTime, and more

NSUserNotification

- NSUserNotification
  - Has properties such as title, informativeText, deliveryTime, and more
- NSUserNotificationCenter
  - Allows management of scheduled or delivered notifications
  - Has a delegate who hears about delivery and activation of notifications
    - Your app may be launched if necessary

NSUserNotification

- Posting a notification

```
NSUserNotification *note = [[NSUserNotification alloc] init];
note.title = @"Hello";
[[NSUserNotificationCenter defaultUserNotificationCenter]
                                                deliverNotification:note];
```

NSUserNotification

- Posting a notification

```
NSUserNotification *note = [[NSUserNotification alloc] init];
note.title = @"Hello";
[[NSUserNotificationCenter defaultUserNotificationCenter]
                                                deliverNotification:note];
```

NSUserNotification

- Posting a notification

```
NSUserNotification *note = [[NSUserNotification alloc] init];
note.title = @"Hello";
[[NSUserNotificationCenter defaultUserNotificationCenter]
                                                deliverNotification:note];
```

NSUserNotification

- Posting a notification

```
NSUserNotification *note = [[NSUserNotification alloc] init];
note.title = @"Hello";
[[NSUserNotificationCenter defaultUserNotificationCenter]
                                                deliverNotification:note];
```

NSUserNotification

- Scheduling a notification to be posted in 30 seconds

- Scheduling a notification at the same time next day

```
note.deliveryDate = [NSDate dateWithTimeIntervalSinceNow:24 * 60 * 60];
```

- Take care how you set absolute dates

NSUserNotification

- Scheduling a notification to be posted in 30 seconds

```
NSUserNotification *note = [[NSUserNotification alloc] init];
note.title = @"Hello";
note.deliveryDate = [NSDate dateWithTimeIntervalSinceNow:30];
[[NSUserNotificationCenter defaultUserNotificationCenter]
                                                scheduleNotification:note];
```

- Scheduling a notification at the same time next day

```
note.deliveryDate = [NSDate dateWithTimeIntervalSinceNow:24 * 60 * 60];
```

- Take care how you set absolute dates

NSUserNotification

- Scheduling a notification to be posted in 30 seconds

```
NSUserNotification *note = [[NSUserNotification alloc] init];
note.title = @"Hello";
note.deliveryDate = [NSDate dateWithTimeIntervalSinceNow:30];
[[NSUserNotificationCenter defaultUserNotificationCenter]
                                                scheduleNotification:note];
```

- Scheduling a notification at the same time next day

```
note.deliveryDate = [NSDate dateWithTimeIntervalSinceNow:24 * 60 * 60];
```

- Take care how you set absolute dates

NSUserNotification

- Scheduling a notification to be posted in 30 seconds

```
NSUserNotification *note = [[NSUserNotification alloc] init];
note.title = @"Hello";
note.deliveryDate = [NSDate dateWithTimeIntervalSinceNow:30];
[[NSUserNotificationCenter defaultUserNotificationCenter]
                                                scheduleNotification:note];
```

- Scheduling a notification at the same time next day

```
note.deliveryDate = [NSDate dateWithTimeIntervalSinceNow:24 * 60 * 60];
```

- Take care how you set absolute dates

NSUserNotification

- Scheduling a notification to be posted in 30 seconds

```
NSUserNotification *note = [[NSUserNotification alloc] init];
note.title = @"Hello";
note.deliveryDate = [NSDate dateWithTimeIntervalSinceNow:30];
[[NSUserNotificationCenter defaultUserNotificationCenter]
                                                scheduleNotification:note];
```

- Scheduling a notification at the same time next day

```
note.deliveryDate = [NSDate dateWithTimeIntervalSinceNow:24 * 60 * 60];
```

NSUserNotification

- Scheduling a notification to be posted in 30 seconds

```
NSUserNotification *note = [[NSUserNotification alloc] init];
note.title = @"Hello";
note.deliveryDate = [NSDate dateWithTimeIntervalSinceNow:30];
[[NSUserNotificationCenter defaultUserNotificationCenter]
                                                scheduleNotification:note];
```

- Scheduling a notification at the same time next day

```
note.deliveryDate = [NSDate dateWithTimeIntervalSinceNow:24 * 60 * 60];
```

NSUserNotification

- Scheduling a notification to be posted in 30 seconds

```
NSUserNotification *note = [[NSUserNotification alloc] init];
note.title = @"Hello";
note.deliveryDate = [NSDate dateWithTimeIntervalSinceNow:30];
[[NSUserNotificationCenter defaultUserNotificationCenter]
                                                scheduleNotification:note];
```

- Scheduling a notification at the same time next day

```
note.deliveryDate = [NSDate dateWithTimeIntervalSinceNow:24 * 60 * 60];
```

- Take care how you set absolute dates

NSUserNotification

- Scheduling a notification to be posted in 30 seconds

```
NSUserNotification *note = [[NSUserNotification alloc] init];
note.title = @"Hello";
note.deliveryDate = [NSDate dateWithTimeIntervalSinceNow:30];
[[NSUserNotificationCenter defaultUserNotificationCenter]
                                                scheduleNotification:note];
```

- Scheduling a notification at the same time next day

```
note.deliveryDate = [NSDate dateWithTimeIntervalSinceNow:24 * 60 * 60];
```

- Take care how you set absolute dates

NSUserNotification

- Notifications presented only if app is not frontmost

NSUserNotification

- Notifications presented only if app is not frontmost
- User controls how notifications are ultimately presented

NSUserNotification

- Notifications presented only if app is not frontmost
- User controls how notifications are ultimately presented
- Guidelines
  - Use sparingly
  - Consider cleaning up

Cocoa XPC

Cocoa XPC

New interprocess communication mechanism for Cocoa

Cocoa XPC

New interprocess communication mechanism for Cocoa

- Secure

Cocoa XPC

New interprocess communication mechanism for Cocoa

- Secure
- Robust

Cocoa XPC

New interprocess communication mechanism for Cocoa

- Secure
- Robust
- Asynchronous

Cocoa XPC

New interprocess communication mechanism for Cocoa

- Secure
- Robust
- Asynchronous
- Cocoa

Classes

Classes

- NSXPCConnection
  - Holds
    - Exported object, its API
    - Proxy to the remote object, and its API
    - Error handlers

Classes

- NSXPCConnection
  - Holds
    - Exported object, its API
    - Proxy to the remote object, and its API
    - Error handlers
- NSXPCListener
  - Listens for connections and forwards to delegate

Classes

- NSXPCConnection
  - Holds
    - Exported object, its API
    - Proxy to the remote object, and its API
    - Error handlers
- NSXPCListener
  - Listens for connections and forwards to delegate
- NSXPCInterface
  - Defines the API the objects respond to

Interfaces

Interfaces

```
@protocol
```

Interfaces

```
@protocol CompressingServer
```

Interfaces

```
@protocol CompressingServer

- (void)compressContentsOfURL:(NSURL *)url;
```

Interfaces

```
@protocol CompressingServer

- (void)compressContentsOfURL:(NSURL *)url;

- (void)compressData:(NSData *)data
          withReply:(void (^)(NSData *compressedData))reply;

@end
```

Interfaces

```
@protocol CompressingServer

- (void)compressContentsOfURL:(NSURL *)url;

- (void)compressData:(NSData *)data
          withReply:(void (^)(NSData *compressedData))reply;

@end


id <CompressingServer> compressor = [connection remoteObjectProxy];
[compressor compressData:bigData withReply:^(NSData *compressedData) {
    [compressedData writeToURL:myURL atomically:YES];
}];
```

Interfaces

```
@protocol CompressingServer

- (void)compressContentsOfURL:(NSURL *)url;

- (void)compressData:(NSData *)data
          withReply:(void (^)(NSData *compressedData))reply;

@end
```

```
id <CompressingServer> compressor = [connection remoteObjectProxy];
[compressor compressData:bigData withReply:^(NSData *compressedData) {
    [compressedData writeToURL:myURL atomically:YES];
}];
```

Interfaces

```
@protocol CompressingServer

- (void)compressContentsOfURL:(NSURL *)url;

- (void)compressData:(NSData *)data
           withReply:(void (^)(NSData *compressedData))reply;

@end


id <CompressingServer> compressor = [connection remoteObjectProxy];
[compressor compressData:bigData withReply:^(NSData *compressedData) {
    [compressedData writeToURL:myURL atomically:YES];
}];
```

Interfaces

```
@protocol CompressingServer

- (void)compressContentsOfURL:(NSURL *)url;

- (void)compressData:(NSData *)data
          withReply:(void (^)(NSData *compressedData))reply;

@end


id <CompressingServer> compressor = [connection remoteObjectProxy];
[compressor compressData:bigData withReply:^(NSData *compressedData) {
    [compressedData writeToURL:myURL atomically:YES];
}];
```

Interfaces

```
@protocol CompressingServer

- (void)compressContentsOfURL:(NSURL *)url;

- (void)compressData:(NSData *)data
          withReply:(void (^)(NSData *compressedData))reply;

@end


id <CompressingServer> compressor = [connection remoteObjectProxy];
[compressor compressData:bigData withReply:^(NSData *compressedData) {
    [compressedData writeToURL:myURL atomically:YES];
}];
```

Secure Coding

- New protocol NSSecureCoding

Secure Coding

- New protocol NSSecureCoding
- Enables use of new object decoding methods

Secure Coding

- New protocol NSSecureCoding
- Enables use of new object decoding methods
  - Existing

```
- (id)decodeObjectForKey:(NSString *)key;
```

Secure Coding

- New protocol NSSecureCoding
- Enables use of new object decoding methods
  - Existing

```
- (id)decodeObjectForKey:(NSString *)key;
```

  - New

```
- (id)decodeObjectOfClass:(Class)aClass forKey:(NSString *)key;
```

Secure Coding

- New protocol NSSecureCoding
- Enables use of new object decoding methods
  - Existing

```
- (id)decodeObjectForKey:(NSString *)key;
```

  - New

```
- (id)decodeObjectOfClass:(Class)aClass forKey:(NSString *)key;
- (id)decodeObjectOfClasses:(NSSet *)classes forKey:(NSString *)key;
- (id)decodePropertyListForKey:(NSString *)key;
```

Secure Coding

- New protocol NSSecureCoding
- Enables use of new object decoding methods
  - Existing
    - `- (id)decodeObjectForKey:(NSString *)key;`
  - New
    - `- (id)decodeObjectOfClass:(Class)aClass forKey:(NSString *)key;`
    - `- (id)decodeObjectOfClasses:(NSSet *)classes forKey:(NSString *)key;`
    - `- (id)decodePropertyListForKey:(NSString *)key;`

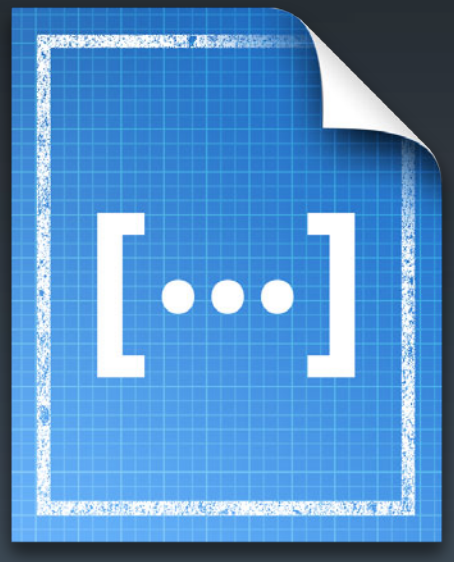
Cocoa Interprocess Communication with XPC	Russian Hill Thursday 4:30PM

Secure Coding

- New protocol NSSecureCoding
- Enables use of new object decoding methods
  - Existing
    - `- (id)decodeObjectForKey:(NSString *)key;`
  - New
    - `- (id)decodeObjectOfClass:(Class)aClass forKey:(NSString *)key;`
    - `- (id)decodeObjectOfClasses:(NSSet *)classes forKey:(NSString *)key;`
    - `- (id)decodePropertyListForKey:(NSString *)key;`

Cocoa Interprocess Communication with XPC	Russian Hill Thursday 4:30PM
Cocoa and XPC Lab	Essentials Lab A Friday 10:15AM

Objective-C



Objective-C

Objective-C

- Syntax for NSNumbers from constants

```
NSNumber *num = @42;
```

Objective-C

- Syntax for NSNumbers from constants

```
NSNumber *num = @42;          [NSNumber numberWithInteger:42];
```

Objective-C

- Syntax for NSNumbers from constants

```
NSNumber *num = @42;          [NSNumber numberWithInteger:42];
NSNumber *b   = @YES;
```

Objective-C

- Syntax for NSNumbers from constants

```
NSNumber *num = @42;          [NSNumber numberWithInteger:42];
NSNumber *b   = @YES;         [NSNumber numberWithBool:YES];
```

Objective-C

- Syntax for NSNumbers from constants

```
NSNumber *num = @42;
NSNumber *b   = @YES;
```

- Syntax for NSNumbers from expressions

```
NSNumber *num = @(40 + 2);
```

Objective-C

- Syntax for NSNumbers from constants

```
NSNumber *num = @42;
NSNumber *b   = @YES;
```

- Syntax for NSNumbers from expressions

```
NSNumber *num = @(40 + 2);
NSNumber *num = @(NSSomeEnumValue);
```

Objective-C

- Syntax for NSNumbers from constants

```
NSNumber *num = @42;
NSNumber *b   = @YES;
```

- Syntax for NSNumbers from expressions

```
NSNumber *num = @(40 + 2);
NSNumber *num = @(NSSomeEnumValue);
```

- And NSStrings from C-Strings

```
NSString *str = @(aFunctionReturningUTF8CString());
```

Objective-C

- Syntax for NSNumbers from constants

```
NSNumber *num = @42;
NSNumber *b   = @YES;
```

- Syntax for NSNumbers from expressions

```
NSNumber *num = @(40 + 2);
NSNumber *num = @(NSSomeEnumValue);
```

- And NSStrings from C-Strings

```
NSString *str = @(aFunctionReturningUTF8CString());
```

  - ASCII or UTF-8 only
  - Make sure the expression isn't NULL

Objective-C

- Syntax for creating NSArray

```
NSArray *array = @[object1, object2, object3];
```

Objective-C

- Syntax for creating NSArray

```
NSArray *array = @[object1, object2, object3];
```

- Syntax for creating NSDictionary

```
NSDictionary *dict = @{key1:object1, key2:object2, key3:object3};
```

Objective-C

- Syntax for accessing NSArray elements

```
NSArray *array = @[object1, object2, object3];
id result = array[1];
```

Objective-C

- Syntax for accessing NSArray elements

```
NSArray *array = @[object1, object2, object3];
id result = array[1];
```

- Syntax for accessing NSDictionary values

```
NSDictionary *dict = @{key1:object1, key2:object2, key3:object3};
id result = dict[key2];
```

Objective-C

- Syntax for accessing NSArray elements

```
NSArray *array = @[object1, object2, object3];
id result = array[1];
```

- Syntax for accessing NSDictionary values

```
NSDictionary *dict = @{key1:object1, key2:object2, key3:object3};
id result = dict[key2];
```

- Available for custom classes as well

Objective-C

- Ability to declare non-standard behaviors

Objective-C

- Ability to declare non-standard behaviors
- Exposed with macros in Foundation
  - NS_RELEASES_ARGUMENT
  - NS_REPLACES_RECEIVER
  - NS_RETURNS_INNER_POINTER
  - NS_VALID_UNTIL_END_OF_SCOPE
  - Etc

Objective-C

- Automatic property synthesis

Objective-C

- Automatic property synthesis
- Enums with fixed underlying types

Objective-C

- Automatic property synthesis
- Enums with fixed underlying types
- Deprecation of garbage collection

Objective-C

- Automatic property synthesis
- Enums with fixed underlying types
- Deprecation of garbage collection

Objective-C

- Automatic property synthesis
- Enums with fixed underlying types
- Deprecation of garbage collection

Modern Objective-C	Presidio Wednesday 10:15AM
Adopting Automatic Reference Counting	Marina Wednesday 11:30AM

Other Topics

NSSharingService

NSSharingService

NSSharingService

NSSharingService

NSSharingService

- Enables sharing of text, files, media, and more

NSSharingService

- Enables sharing of text, files, media, and more
- Allows customization of
  - Sharing service picker
  - UI for invoking the picker

NSSharingService

- Enables sharing of text, files, media, and more
- Allows customization of
  - Sharing service picker
  - UI for invoking the picker

Integrating With Facebook, Twitter and Sina Weibo

Presidio
Thursday 2:00PM

Application Sandbox

Application Sandbox

- Support for related items
  - Files/folders that need to be accessed with a user document

Application Sandbox

- Support for related items
  - Files/folders that need to be accessed with a user document
- Security-scoped bookmarks
  - Persistent access to user-chosen files and folders

Application Sandbox

- Support for related items
  - Files/folders that need to be accessed with a user document
- Security-scoped bookmarks
  - Persistent access to user-chosen files and folders

The OS X App Sandbox	Nob Hill Tuesday/Friday 10:15AM

Application Sandbox

- Support for related items
  - Files/folders that need to be accessed with a user document
- Security-scoped bookmarks
  - Persistent access to user-chosen files and folders

The OS X App Sandbox	Nob Hill Tuesday/Friday 10:15AM
Security Lab	Core OS Lab B Tuesday 3:15PM
Security Lab	Core OS Lab B Thursday 9:00AM

Application Sandbox

- Support for related items
  - Files/folders that need to be accessed with a user document
- Security-scoped bookmarks
  - Persistent access to user-chosen files and folders

The OS X App Sandbox	Nob Hill Tuesday/Friday 10:15AM
Security Lab	Core OS Lab B Tuesday 3:15PM
Security Lab	Core OS Lab B Thursday 9:00AM
Open and Save Panels Within an App Sandbox Q&A Lab	Essentials Lab B Wednesday 4:30PM

Auto Layout

Support for NSTextField wrapping

Auto Layout

Support for NSTextField wrapping

- Specify the preferred width

```
– (CGFloat)preferredMaxLayoutWidth;
– (void)setPreferredMaxLayoutWidth:(CGFloat)width;
```

Auto Layout

Improved support for NSSplitView

Auto Layout

Improved support for NSSplitView

- Now respects subview constraints

Auto Layout

Improved support for NSSplitView

- Now respects subview constraints
- Allows specifying holding priorities for individual subviews

```
- (NSLayoutPriority)holdingPriorityForSubviewAtIndex:(NSInteger)index;
- (void)setHoldingPriority:(NSLayoutPriority)priority
        forSubviewAtIndex:(NSInteger)subviewIndex;
```

Auto Layout

Automatic localization

Auto Layout

Automatic localization

- Constraints allow controls to resize and reposition based on content
  - Leading/trailing support enables right-to-left languages

Auto Layout

Automatic localization

- Constraints allow controls to resize and reposition based on content
  - Leading/trailing support enables right-to-left languages
- Use just one nib file

Auto Layout

Automatic localization

- Constraints allow controls to resize and reposition based on content
  - Leading/trailing support enables right-to-left languages
- Use just one nib file
- Apply strings for different languages

Auto Layout

Automatic localization

- Constraints allow controls to resize and reposition based on content
  - Leading/trailing support enables right-to-left languages
- Use just one nib file
- Apply strings for different languages
- Profit!

Auto Layout

- Now also available on iOS

Auto Layout

- Now also available on iOS

Introduction to Auto Layout for iOS and OS X	Mission Tuesday 10:15AM
Best Practices for Mastering Auto Layout	Mission Thursday 9:00AM
Auto Layout by Example	Mission Thursday 11:30AM

Auto Layout

- Now also available on iOS

Introduction to Auto Layout for iOS and OS X	Mission Tuesday 10:15AM
Best Practices for Mastering Auto Layout	Mission Thursday 9:00AM
Auto Layout by Example	Mission Thursday 11:30AM
Auto Layout Lab	Essentials Lab A Tuesday 2:00PM
Auto Layout Lab	App Services Lab B Thursday 2:00PM

NSByteCountFormatter

NSFormatter subclass

NSByteCountFormatter

NSFormatter subclass

- Displays byte counts

NSByteCountFormatter

NSFormatter subclass

- Displays byte counts
  - With appropriate units

NSByteCountFormatter

NSFormatter subclass

- Displays byte counts
  - With appropriate units
  - With thousands separators

NSByteCountFormatter

NSFormatter subclass

- Displays byte counts
  - With appropriate units
  - With thousands separators
  - With appropriate localization

NSByteCountFormatter

NSFormatter subclass

- Displays byte counts
  - With appropriate units
  - With thousands separators
  - With appropriate localization
  - Just like the rest of the system

NSByteCountFormatter

NSByteCountFormatter

Zero KB

NSByteCountFormatter

Zero KB 0 bytes

NSByteCountFormatter

Zero KB 0 bytes 0 GB

NSByteCountFormatter

Zero KB 0 bytes 0 GB

42 KB

NSByteCountFormatter

Zero KB 0 bytes 0 GB

42 KB 42

NSByteCountFormatter

Zero KB 0 bytes 0 GB

42 KB 42 KB

NSByteCountFormatter

Zero KB 0 bytes 0 GB

42 KB 42 KB 42 KB (42,109 bytes)

NSByteCountFormatter

Zero KB 0 bytes 0 GB

42 KB 42 KB 42 KB (42,109 bytes)

12.35 GB

NSByteCountFormatter

Zero KB 0 bytes 0 GB

42 KB 42 KB 42 KB (42,109 bytes)

12.35 GB 12,345.7 MB

NSByteCountFormatter

Zero KB 0 bytes 0 GB

42 KB 42 KB 42 KB (42,109 bytes)

12.35 GB 12,345.7 MB 12,345,679 KB

NSByteCountFormatter

Zero KB	0 bytes	0 GB	0 Go
42 KB	42 KB	42 KB (42,109 bytes)	42 Ko (42 109 octets)
12.35 GB	12,345.7 MB	12,345,679 KB	12 345 679 Ko

NSByteCountFormatter

Zero KB	0 bytes	0 GB	0 Go	٠ غ.ب.
42 KB	42 KB	42 KB (42,109 bytes)	42 Ko (42 109 octets)	٤٢ ك.ب. (٤٢١٠٩ بايت)
12.35 GB	12,345.7 MB	12,345,679 KB	12 345 679 Ko	١٢٣٤٥٦٧٩ ك.ب.

NSString Localized Number Formatting

NSString Localized Number Formatting

- Further proper localization of numbers in existing APIs

```
+ (id)localizedStringWithFormat:(NSString *)format, ...;
- (id)initWithFormat:(NSString *)format locale:(id)locale, ...;
```

NSString Localized Number Formatting

- Further proper localization of numbers in existing APIs

```
+ (id)localizedStringWithFormat:(NSString *)format, ...;
- (id)initWithFormat:(NSString *)format locale:(id)locale, ...;
```

- In addition to decimal point localization
  - Thousands separator
  - Digits

NSString Localized Number Formatting

- Further proper localization of numbers in existing APIs

```
+ (id)localizedStringWithFormat:(NSString *)format, ...;
- (id)initWithFormat:(NSString *)format locale:(id)locale, ...;
```

- In addition to decimal point localization
  - Thousands separator
  - Digits
- Automatic for apps linked against 10.8 SDK

Zeroing-Weak Support in Collections

- NSMapTable
- NSHashTable
- NSPointerArray

Zeroing-Weak Support in Collections

```
id obj = [NSObject new];
NSMapTable *table = [NSMapTable strongToWeakObjectsMapTable];
[table setObject:obj forKey:@"key"];
...
[obj release];
...
NSLog(@"%@", [table objectForKey:@"key"]);
```

Zeroing-Weak Support in Collections

```
id obj = [NSObject new];
NSMapTable *table = [NSMapTable strongToWeakObjectsMapTable];
[table setObject:obj forKey:@"key"];
...
[obj release];
...
NSLog(@"%@", [table objectForKey:@"key"]);
```

Zeroing-Weak Support in Collections

```
id obj = [NSObject new];
NSMapTable *table = [NSMapTable strongToWeakObjectsMapTable];
[table setObject:obj forKey:@"key"];
...
[obj release];
...
NSLog(@"%@", [table objectForKey:@"key"]);
```

Zeroing-Weak Support in Collections

```
id obj = [NSObject new];
NSMapTable *table = [NSMapTable strongToWeakObjectsMapTable];
[table setObject:obj forKey:@"key"];
...
[obj release];
...
NSLog(@"%@", [table objectForKey:@"key"]);
```

Zeroing-Weak Support in Collections

```
id obj = [NSObject new];
NSMapTable *table = [NSMapTable strongToWeakObjectsMapTable];
[table setObject:obj forKey:@"key"];
...
[obj release];
...
NSLog(@"%@", [table objectForKey:@"key"]);
```

Zeroing-Weak Support in Collections

```
id obj = [NSObject new];
NSMapTable *table = [NSMapTable strongToWeakObjectsMapTable];
[table setObject:obj forKey:@"key"];
...
[obj release];
...
NSLog(@"%@", [table objectForKey:@"key"]);
```

Zeroing-Weak Support in Collections

```
id obj = [NSObject new];
NSMapTable *table = [NSMapTable strongToWeakObjectsMapTable];
[table setObject:obj forKey:@"key"];
...
[obj release];
...
NSLog(@"%@", [table objectForKey:@"key"]);        // Logs “(null)”
```

Zeroing-Weak Support in Collections

```
id obj = [NSObject new];
NSMapTable *table = [NSMapTable strongToWeakObjectsMapTable];
[table setObject:obj forKey:@"key"];
...
[obj release];
...
NSLog(@"%@", [table objectForKey:@"key"]);        // Logs “(null)”
```

- Works under manual reference counting as well

Shared Key Dictionary

- Faster NSMutableDictionary for when the set of keys is known

Shared Key Dictionary

- Faster NSMutableDictionary for when the set of keys is known

```
@interface NSMutableDictionary
```

Shared Key Dictionary

- Faster NSMutableDictionary for when the set of keys is known

```
@interface NSMutableDictionary
...
+ (id)sharedKeySetForKeys:(NSArray *)keys;
```

Shared Key Dictionary

- Faster NSMutableDictionary for when the set of keys is known

```
@interface NSMutableDictionary
...
+ (id)sharedKeySetForKeys:(NSArray *)keys;
+ (id)dictionaryWithSharedKeySet:(id)keyset;

@end
```

Shared Key Dictionary

- Faster NSMutableDictionary for when the set of keys is known

```
@interface NSMutableDictionary
...
+ (id)sharedKeySetForKeys:(NSArray *)keys;
+ (id)dictionaryWithSharedKeySet:(id)keyset;

@end


id textKeys = [NSMutableDictionary sharedKeySetForKeys:
                                        @[@"Color", @"Font", @"Underline"]]];
...
```

Shared Key Dictionary

- Faster NSMutableDictionary for when the set of keys is known

```
@interface NSMutableDictionary
...
+ (id)sharedKeySetForKeys:(NSArray *)keys;
+ (id)dictionaryWithSharedKeySet:(id)keyset;

@end


id textKeys = [NSMutableDictionary sharedKeySetForKeys:
                                    @[@"Color", @"Font", @"Underline"]]];
...
NSMutableDictionary *dict =
              [NSMutableDictionary dictionaryWithSharedKeySet:textKeys];
```

Shared Key Dictionary

- Performance benefits

Shared Key Dictionary

- Performance benefits
  - Faster look-up due to use of minimal perfect hash

Shared Key Dictionary

- Performance benefits
  - Faster look-up due to use of minimal perfect hash
  - Memory savings due to sharing of keys

Shared Key Dictionary

- Performance benefits
  - Faster look-up due to use of minimal perfect hash
  - Memory savings due to sharing of keys
- Also allows for keys not in the keyset

Resume Behavior Change

Resume Behavior Change

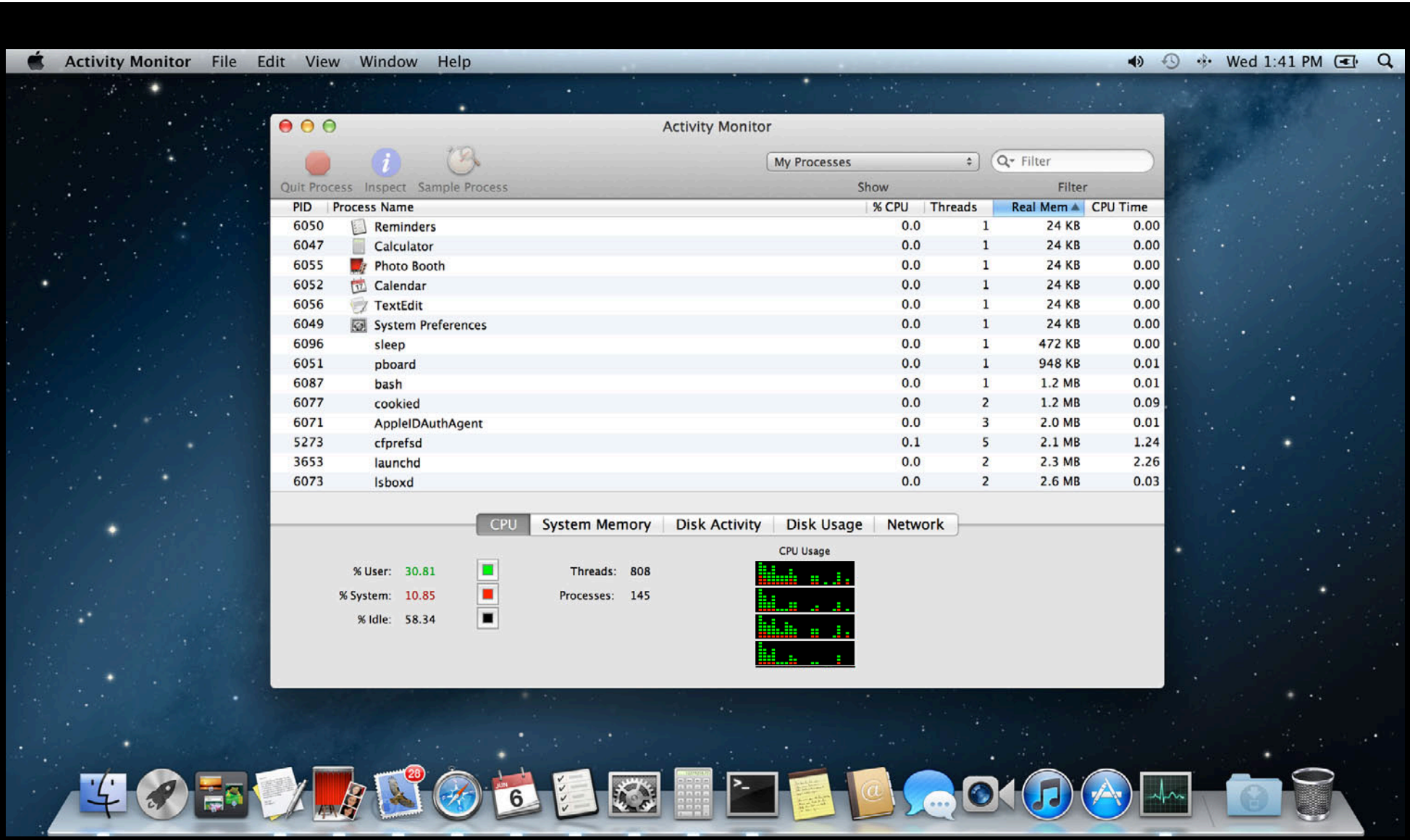
- On login, apps that were hidden at logout time are partially relaunched

Resume Behavior Change

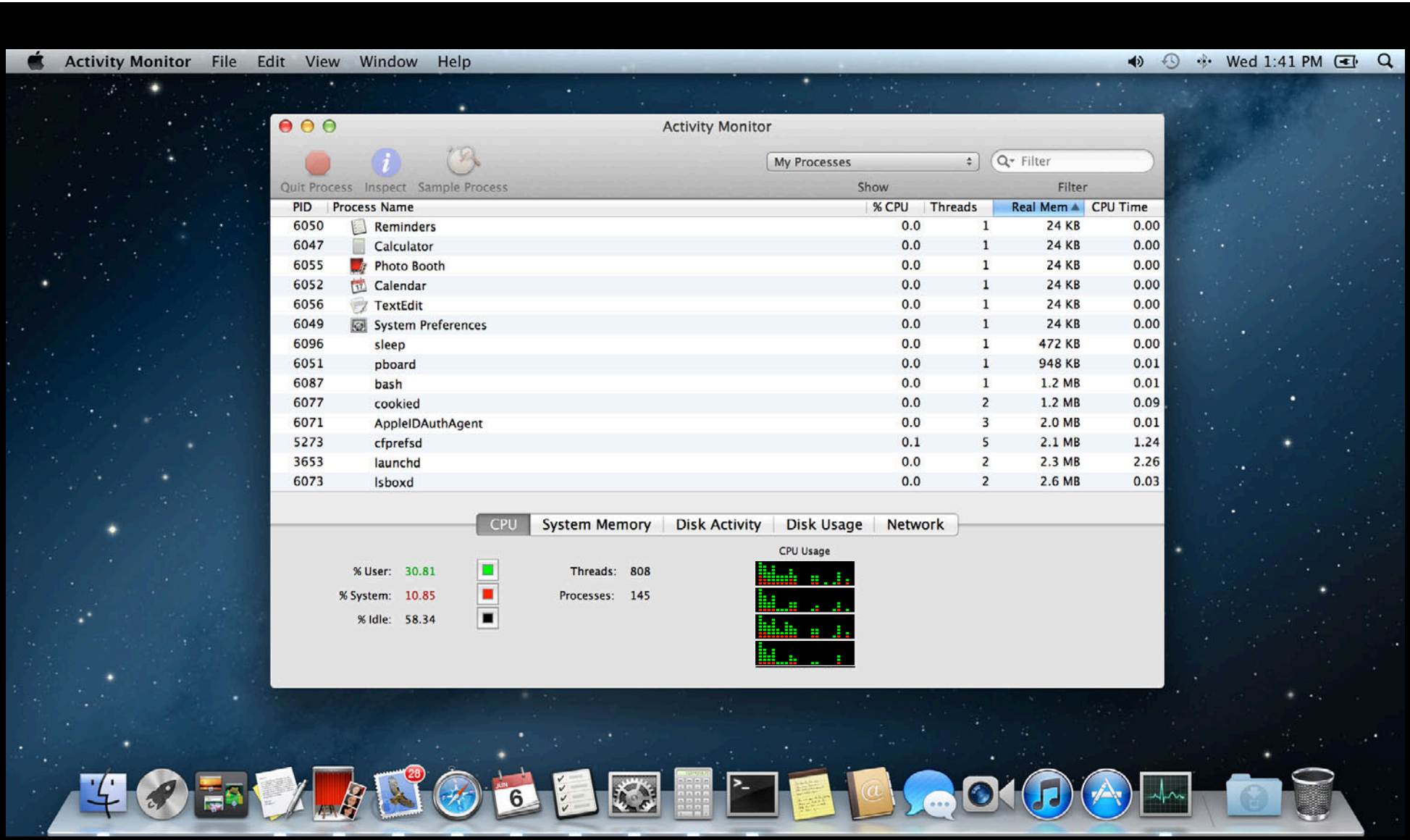
- On login, apps that were hidden at logout time are partially relaunched
  - Enough to make them look like they're launched

Resume Behavior Change

- On login, apps that were hidden at logout time are partially relaunched
  - Enough to make them look like they're launched
  - But they are not really running


Activity Monitor File Edit View Window Help
Wed 1:41 PM
Activity Monitor
Quit Process
Inspect
Sample Process
My Processes
Show
Filter
PID
Process Name
% CPU
Threads
Real Mem
CPU Time
6050 Reminders 0.0 1 24 KB 0.00
6047 Calculator 0.0 1 24 KB 0.00
6055 Photo Booth 0.0 1 24 KB 0.00
6052 Calendar 0.0 1 24 KB 0.00
6056 TextEdit 0.0 1 24 KB 0.00
6049 System Preferences 0.0 1 24 KB 0.00
6096 sleep 0.0 1 472 KB 0.00
6051 pboard 0.0 1 948 KB 0.01
6087 bash 0.0 1 1.2 MB 0.01
6077 cookied 0.0 2 1.2 MB 0.09
6071 AppleIDAuthAgent 0.0 3 2.0 MB 0.01
5273 cfprefsd 0.1 5 2.1 MB 1.24
3653 launchd 0.0 2 2.3 MB 2.26
6073 lsboxd 0.0 2 2.6 MB 0.03
CPU
System Memory
Disk Activity
Disk Usage
Network
% User: 30.81
% System: 10.85
% Idle: 58.34
Threads: 808
Processes: 145
CPU Usage


Activity Monitor File Edit View Window Help
Wed 1:41 PM
Activity Monitor
Quit Process
Inspect
Sample Process
My Processes
Filter
Show
Filter
PID
Process Name
% CPU
Threads
Real Mem
CPU Time
6050 Reminders 0.0 1 24 KB 0.00
6047 Calculator 0.0 1 24 KB 0.00
6055 Photo Booth 0.0 1 24 KB 0.00
6052 Calendar 0.0 1 24 KB 0.00
6056 TextEdit 0.0 1 24 KB 0.00
6049 System Preferences 0.0 1 24 KB 0.00
6103 sleep 0.0 1 472 KB 0.00
6051 pboard 0.0 1 948 KB 0.01
6087 bash 0.0 1 1.2 MB 0.01
Activity Monitor
CPU
System Memory
Disk Activity
Disk Usage
Network
% User: 1.00
% System: 1.26
% Idle: 97.74
Threads: 837
Processes: 146
CPU Usage


Activity Monitor File Edit View Window Help
Wed 1:41 PM
Activity Monitor
Quit Process
Inspect
Sample Process
My Processes
Show
Filter
PID
Process Name
% CPU
Threads
Real Mem
CPU Time
6050 Reminders 0.0 1 24 KB 0.00
6047 Calculator 0.0 1 24 KB 0.00
6055 Photo Booth 0.0 1 24 KB 0.00
6052 Calendar 0.0 1 24 KB 0.00
6056 TextEdit 0.0 1 24 KB 0.00
6049 System Preferences 0.0 1 24 KB 0.00
6096 sleep 0.0 1 472 KB 0.00
6051 pboard 0.0 1 948 KB 0.01
6087 bash 0.0 1 1.2 MB 0.01
6077 cookied 0.0 2 1.2 MB 0.09
6071 AppleIDAuthAgent 0.0 3 2.0 MB 0.01
5273 cfprefsd 0.1 5 2.1 MB 1.24
3653 launchd 0.0 2 2.3 MB 2.26
6073 lsboxd 0.0 2 2.6 MB 0.03
CPU
System Memory
Disk Activity
Disk Usage
Network
% User: 30.81
% System: 10.85
% Idle: 58.34
Threads: 808
Processes: 145
CPU Usage

Resume Behavior Change

What if your app needs to be relaunched, even if hidden?

Resume Behavior Change

What if your app needs to be relaunched, even if hidden?

- A hidden app will be relaunched on login if it

Resume Behavior Change

What if your app needs to be relaunched, even if hidden?

- A hidden app will be relaunched on login if it
  - Opts in to auto-termination

Resume Behavior Change

What if your app needs to be relaunched, even if hidden?

- A hidden app will be relaunched on login if it
  - Opts in to auto-termination

```
<key>NSSupportsAutomaticTermination</key>
<true/>
```

Resume Behavior Change

What if your app needs to be relaunched, even if hidden?

- A hidden app will be relaunched on login if it
  - Opts in to auto-termination

```
<key>NSSupportsAutomaticTermination</key>
<true/>
```

  - But is not auto-terminable at logout time

```
[[NSProcessInfo processInfo] disableAutomaticTermination:@"Reason"];
```

Resume Behavior Change

What if your app needs to be relaunched, even if hidden?

- A hidden app will be relaunched on login if it
  - Opts in to auto-termination

```
<key>NSSupportsAutomaticTermination</key>
<true/>
```

  - But is not auto-terminable at logout time

```
[[NSProcessInfo processInfo] disableAutomaticTermination:@"Reason"];
```

- Adopt auto-termination!

Resume Behavior Change

What if your app needs to be relaunched, even if hidden?

- A hidden app will be relaunched on login if it
  - Opts in to auto-termination

```
<key>NSSupportsAutomaticTermination</key>
<true/>
```

  - But is not auto-terminable at logout time

```
[[NSProcessInfo processInfo] disableAutomaticTermination:@"Reason"];
```

- Adopt auto-termination!

Other Enhancements

Other Enhancements

- New class NSUUID

Other Enhancements

- New class NSUUID
- New class NSUserScriptTask and subclasses

Other Enhancements

- New class NSUUID
- New class NSUserScriptTask and subclasses
- New class NSTextAlternatives

Other Enhancements

- New class NSUUID
- New class NSUserScriptTask and subclasses
- New class NSTextAlternatives
- Support for weak referencing of NSWindow, NSViewController, and more

Other Enhancements

- New class NSUUID
- New class NSUserScriptTask and subclasses
- New class NSTextAlternatives
- Support for weak referencing of NSWindow, NSViewController, and more
- Ability to register nibs in NSTableView / NSOutlineView

Other Enhancements

- New class NSUUID
- New class NSUserScriptTask and subclasses
- New class NSTextAlternatives
- Support for weak referencing of NSWindow, NSViewController, and more
- Ability to register nibs in NSTableView / NSOutlineView
- New NSColor API for areas revealed behind views

Other Enhancements

- New class NSUUID
- New class NSUserScriptTask and subclasses
- New class NSTextAlternatives
- Support for weak referencing of NSWindow, NSViewController, and more
- Ability to register nibs in NSTableView / NSOutlineView
- New NSColor API for areas revealed behind views
- ARC-compatible loading APIs in NSNib

Other Enhancements

- NSURL API to exclude files from being backed up

Other Enhancements

- NSURL API to exclude files from being backed up
- NSData API to avoid overwriting files

Other Enhancements

- NSURL API to exclude files from being backed up
- NSData API to avoid overwriting files
- NSFileManager APIs to trash items

Other Enhancements

- NSURL API to exclude files from being backed up
- NSData API to avoid overwriting files
- NSFileManager APIs to trash items
- Locale-parameterized case conversion APIs in NSString

Other Enhancements

- NSURL API to exclude files from being backed up
- NSData API to avoid overwriting files
- NSFileManager APIs to trash items
- Locale-parameterized case conversion APIs in NSString
- NSDateComponents leap month API

Summary

Exciting new user and developer features for you to adopt

- iCloud
- Auto Save
- High Resolution
- Gestures
- Layer-Backed Views
- Auto Layout and Auto Localization
- Cocoa XPC
- Notification Center
- Sharing Service
- And much more!

Resources

- Documentation

Resources

- Documentation
- Release notes
  - AppKit (“Application Kit Release Notes for OS X 10.8”)
  - Foundation (“Foundation Release Notes for OS X 10.8”)

Resources

- Documentation
- Release notes
  - AppKit (“Application Kit Release Notes for OS X 10.8”)
  - Foundation (“Foundation Release Notes for OS X 10.8”)
- Header files (Search for “10_8”)
  - MAC_OS_X_VERSION_10_8
  - NS_AVAILABLE_MAC(10_8), NS_AVAILABLE(10_8 ...)
  - NS_ENUM_AVAILABLE(10_8, ...), NS_CLASS_AVAILABLE(10_8, ...)
  - NS_DEPRECATED_MAC(..., 10_8), NS_DEPRECATED(..., 10_8, ..., ...)

Labs

Cocoa Lab	Essentials Lab B Tuesday 2:00PM
Cocoa and Layer-Backed Views on OS X Lab	Essentials Lab B Wednesday 2:00PM
OS X Gestures and Cocoa Lab	Essentials Lab B Thursday 2:00PM
Cocoa and XPC Lab	Essentials Lab A Friday 10:15AM

Labs

High Resolution on OS X Lab	Essentials Lab B Wednesday 11:30AM
Auto Layout Lab	Essentials Lab A Tuesday 2:00PM
Auto Layout Lab	App Services Lab B Thursday 2:00PM

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

Labs

Security Lab	Core OS Lab B Tuesday 3:15PM
Security Lab	Core OS Lab B Thursday 9:00AM
Open and Save Panels Within an App Sandbox Q&A Lab	Essentials Lab B Wednesday 4:30PM

More Information

Jake Behrens
UI Frameworks Evangelist
behrens@apple.com

Documentation
Mac Dev Center
http://developer.apple.com/mac
https://developer.apple.com/devcenter/mac/checklist/mountain-lion

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


WWDC2012