App Frameworks

#WWDC15

What's New in Cocoa

Session 202

Ali Ozer Director of Cocoa Frameworks

Agenda

Swiftification

AppKit

Foundation

Swiftification

API enhancements in support of Swift

APIs in Swift

Swift provides features to express APIs more precisely

APIs in Swift

Swift provides features to express APIs more precisely

```
var subviews: [NSView]
```

APIs in Swift

Swift provides features to express APIs more precisely

```
var subviews: [NSView]
class func systemFontOfSize(CGFloat) -> NSFont
```

APIs in Swift

Swift provides features to express APIs more precisely

```
var subviews: [NSView]

class func systemFontOfSize(CGFloat) -> NSFont

func imageForResource(String) -> NSImage?
```

Swiftification

We enabled Objective-C to express these Swift facilities

- Nullability
- Generics

Swiftification

We enabled Objective-C to express these Swift facilities

- Nullability
- Generics

Applied these to APIs of many of our frameworks

- Better exposure in Swift
- Clearer API expression
- Compile time type checking

Nullability

Whether values can or cannot be nil

Nullability

Whether values can or cannot be nil

10.10 and iOS 8 SDKs had nullability declarations for Swift only

Nullability

Whether values can or cannot be nil

10.10 and iOS 8 SDKs had nullability declarations for Swift only

10.11 and iOS 9 bring this ability to Objective-C

Nullability

Whether values can or cannot be nil

10.10 and iOS 8 SDKs had nullability declarations for Swift only

10.11 and iOS 9 bring this ability to Objective-C

`nonnull` never nil

Nullability

Whether values can or cannot be nil

10.10 and iOS 8 SDKs had nullability declarations for Swift only

10.11 and iOS 9 bring this ability to Objective-C

`nonnull`	never nil
`nullable`	can be nil

Nullability

Whether values can or cannot be nil

10.10 and iOS 8 SDKs had nullability declarations for Swift only

10.11 and iOS 9 bring this ability to Objective-C

`nonnull`	never nil
`nullable`	can be nil
`null_resettable`	property can be set to nil, but won't return nil

Nullability

Whether values can or cannot be nil

10.10 and iOS 8 SDKs had nullability declarations for Swift only

10.11 and iOS 9 bring this ability to Objective-C

`nonnull`	never nil
`nullable`	can be nil
`null_resettable`	property can be set to nil, but won't return nil
`null_unspecified`	not specified

Nullability

Since nonnull is the majority case, we also have

```
NS_ASSUME_NONNULL_BEGIN
NS_ASSUME_NONNULL_END
```

Nullability

Since nonnull is the majority case, we also have

```
NS_ASSUME_NONNULL_BEGIN
NS_ASSUME_NONNULL_END
```

With these, nonnull is unnecessary

Nullability

NSColorWell

```
@property (copy) NSColor *color;
```

NSImageView

```
@property (nullable, strong) NSImage *image;
```

NSMenu

```
@property (null_resettable, strong) NSFont *font;
```

Nullability

NSColorWell

```
@property (copy) NSColor *color;
```

⟶

```
var color: NSColor
```

NSImageView

```
@property (nullable, strong) NSImage *image;
```

NSMenu

```
@property (null_resettable, strong) NSFont *font;
```

Nullability

NSColorWell

```
@property (copy) NSColor *color;
```

⟶

```
var color: NSColor
```

NSImageView

```
@property (nullable, strong) NSImage *image;
```

⟶

```
var image: NSImage?
```

NSMenu

```
@property (null_resettable, strong) NSFont *font;
```

Nullability

NSColorWell

```
@property (copy) NSColor *color;                    ⟶  var color: NSColor
```

NSImageView

```
@property (nullable, strong) NSImage *image;        ⟶  var image: NSImage?
```

NSMenu

```
@property (null_resettable, strong) NSFont *font;   ⟶  var font: NSFont!
```

Nullability

You may see build time warnings

Nullability

You may see build time warnings

```
colorWell.color = nil;
```

Nullability

You may see build time warnings

```
colorWell.color = nil;
```

⚠ Null passed to a callee that requires a non-null argument

Nullability Guidelines

In general, nil is not a valid object value

Nullability Guidelines

In general, nil is not a valid object value

NSString, NSArray, NSDictionary properties are rarely nil

```
@"", @[], @{}
```

Nullability Guidelines

In general, nil is not a valid object value

NSString, NSArray, NSDictionary properties are rarely nil

```
@"", @[], @{}
```

APIs that accept or return nil should document what nil means

Nullability Guidelines

In general, nil is not a valid object value

NSString, NSArray, NSDictionary properties are rarely nil

```
@"", @[], @{}
```

APIs that accept or return nil should document what nil means

- A nil backgroundColor in NSText means “don’t draw background”

Nullability Guidelines

In general, nil is not a valid object value

NSString, NSArray, NSDictionary properties are rarely nil

```
@"", @[], @{}
```

APIs that accept or return nil should document what nil means

- A nil backgroundColor in NSText means “don’t draw background”
- A nil locale in many of our APIs means “non-localized”

Generics

Generics

Lightweight type parameterization

Generics

Lightweight type parameterization

Great for specifying types of elements in collections

Generics

Lightweight type parameterization

Great for specifying types of elements in collections

```
@property (copy) NSArray *recentSearches;                    // 10.10
```

Generics

Lightweight type parameterization

Great for specifying types of elements in collections

```
@property (copy) NSArray *recentSearches;                    // 10.10

@property (copy) NSArray<NSString *> *recentSearches;        // 10.11
```

Generics

Lightweight type parameterization

Great for specifying types of elements in collections

```
@property (copy) NSArray *recentSearches;                    // 10.10

@property (copy) NSArray<NSString *> *recentSearches;        // 10.11

var recentSearches: [AnyObject]                              // 10.10
```

Generics

Lightweight type parameterization

Great for specifying types of elements in collections

```
@property (copy) NSArray *recentSearches;                    // 10.10

@property (copy) NSArray<NSString *> *recentSearches;        // 10.11

var recentSearches: [AnyObject]                              // 10.10

var recentSearches: [String]                                 // 10.11
```

Generics

```
@interface NSArray : NSObject
```

Generics

```
@interface NSArray<ObjectType> : NSObject
```

Generics

```
@interface NSArray<ObjectType> : NSObject

- (ObjectType)objectAtIndex:(NSUInteger)index;
```

Generics

```
@interface NSArray<ObjectType> : NSObject

- (ObjectType)objectAtIndex:(NSUInteger)index;
- (BOOL)containsObject:(ObjectType)anObject;
```

Generics

```
@interface NSArray<ObjectType> : NSObject

- (ObjectType)objectAtIndex:(NSUInteger)index;
- (BOOL)containsObject:(ObjectType)anObject;
- (NSArray<ObjectType> *)arrayByAddingObject:(ObjectType)anObject;
```

Generics

```
@interface NSArray<ObjectType> : NSObject

- (ObjectType)objectAtIndex:(NSUInteger)index;
- (BOOL)containsObject:(ObjectType)anObject;
- (NSArray<ObjectType> *)arrayByAddingObject:(ObjectType)anObject;
...
@end
```

Generics

```
@interface NSArray<ObjectType> : NSObject

- (ObjectType)objectAtIndex:(NSUInteger)index;
- (BOOL)containsObject:(ObjectType)anObject;
- (NSArray<ObjectType> *)arrayByAddingObject:(ObjectType)anObject;
...
@end

@property (copy) NSArray<NSString *> *recentSearches;
```

Generics

```
@interface NSArray<ObjectType> : NSObject

- (ObjectType)objectAtIndex:(NSUInteger)index;
- (BOOL)containsObject:(ObjectType)anObject;
- (NSArray<ObjectType> *)arrayByAddingObject:(ObjectType)anObject;
...
@end

@property (copy) NSArray<NSString *> *recentSearches;
...
if ([searchField.recentSearches containsObject:someURL]) ...
```

Generics

```
@interface NSArray<ObjectType> : NSObject

- (ObjectType)objectAtIndex:(NSUInteger)index;
- (BOOL)containsObject:(ObjectType)anObject;
- (NSArray<ObjectType> *)arrayByAddingObject:(ObjectType)anObject;
...
@end

@property (copy) NSArray<NSString *> *recentSearches;
...
if ([searchField.recentSearches containsObject:someURL]) ...
```

⚠ Incompatible pointer types sending 'NSURL *' to parameter of type 'NSString * __nonnull'

Generics Support in Foundation Collections

NSArray

NSDictionary

NSSet

NSOrderedSet

NSHashTable

NSMapTable

NSCache

NSEnumerator

Generics Guidelines

Generics Guidelines

Use generics in variable declarations in your call sites

Generics Guidelines

Use generics in variable declarations in your call sites

```
NSArray *recents = searchField.recentSearches;
```

Generics Guidelines

Use generics in variable declarations in your call sites

```
NSArray<NSString *> *recents = searchField.recentSearches;
```

Generics Guidelines

Use generics in variable declarations in your call sites

```
NSArray<NSString *> *recents = searchField.recentSearches;
```

Generics Guidelines

Use generics in variable declarations in your call sites

Decorate your own properties and APIs

Generics Guidelines

Use generics in variable declarations in your call sites

Decorate your own properties and APIs

```
@property (copy) NSArray *files;
```

Generics Guidelines

Use generics in variable declarations in your call sites

Decorate your own properties and APIs

```
@property (copy) NSArray<NSURL *> *files;
```

Generics Guidelines

Use generics in variable declarations in your call sites

Decorate your own properties and APIs

```
@property (copy) NSArray<NSURL *> *files;
```

Generics Guidelines

Use generics in variable declarations in your call sites

Decorate your own properties and APIs

Apply generics to your custom collection classes and your custom categories on Foundation collections

Generics Guidelines

Use generics in variable declarations in your call sites

Decorate your own properties and APIs

Apply generics to your custom collection classes and your custom categories on Foundation collections

```
@interface NSArray(MySpecialSortExtensions)
@property (copy) NSArray *mySortedArrayByColor;
@end
```

Generics Guidelines

Use generics in variable declarations in your call sites

Decorate your own properties and APIs

Apply generics to your custom collection classes and your custom categories on Foundation collections

```
@interface NSArray<ObjectType> (MySpecialSortExtensions)
@property (copy) NSArray<ObjectType> *mySortedArrayByColor;
@end
```

Generics Guidelines

Use generics in variable declarations in your call sites

Decorate your own properties and APIs

Apply generics to your custom collection classes and your custom categories on Foundation collections

```
@interface NSArray<ObjectType> (MySpecialSortExtensions)
@property (copy) NSArray<ObjectType> *mySortedArrayByColor;
@end
```

kindof

kindof

NSView's subviews property in 10.10

```
@property(copy) NSArray *subviews;
```

kindof

NSView's subviews property in 10.10

```
@property(copy) NSArray *subviews;
```

First attempt at applying generics

```
@property(copy) NSArray<NSView *> *subviews;
```

kindof

NSView's subviews property in 10.10

```
@property(copy) NSArray *subviews;
```

First attempt at applying generics

```
@property(copy) NSArray<NSView *> *subviews;
```

Unintended result

```
NSButton *button = myContainerView.subviews[0];
```

kindof

NSView's subviews property in 10.10

```
@property(copy) NSArray *subviews;
```

First attempt at applying generics

```
@property(copy) NSArray<NSView *> *subviews;
```

Unintended result

```
NSButton *button = myContainerView.subviews[0];
```

⚠ Incompatible pointer types initializing 'NSButton *' with an expression of type 'NSView *'

kindof

NSView's subviews property in 10.10

```
@property(copy) NSArray *subviews;
```

First attempt at applying generics

```
@property(copy) NSArray<NSView *> *subviews;
```

With "kindof"

kindof

NSView's subviews property in 10.10

```
@property(copy) NSArray *subviews;
```

First attempt at applying generics

```
@property(copy) NSArray<NSView *> *subviews;
```

With "kindof"

```
@property(copy) NSArray<__kindof NSView *> *subviews;
```

kindof

NSView's subviews property in 10.10

```
@property(copy) NSArray *subviews;
```

First attempt at applying generics

```
@property(copy) NSArray<NSView *> *subviews;
```

With "kindof"

```
@property(copy) NSArray<__kindof NSView *> *subviews;
```

```
NSButton *button = myContainerView.subviews[0];
```

kindof Guidelines

kindof Guidelines

"kindof" does not imply an automatic runtime type check

kindof Guidelines

"kindof" does not imply an automatic runtime type check

Use sparingly

- Where it should be safe for the caller to make assumption about the class of the object

kindof Guidelines

"kindof" does not imply an automatic runtime type check

Use sparingly

- Where it should be safe for the caller to make assumption about the class of the object
- Avoid in cases where the caller should do a runtime type query
  - NSImage

```
@property (readonly, copy) NSArray<NSImageRep *> *representations;
```

Error Handling

NSError returning methods in Swift

Error Handling

NSError returning methods in Swift

```
- (BOOL)writeToURL:(NSURL *)url
           options:(NSDataWritingOptions)opts
             error:(NSError **)errorPtr;
```

Error Handling

NSError returning methods in Swift

```
- (BOOL)writeToURL:(NSURL *)url
           options:(NSDataWritingOptions)opts
             error:(NSError **)errorPtr;

func writeToURL(NSURL, options: NSDataWritingOptions) throws
```

Error Handling

NSError returning methods in Swift

```
- (BOOL)writeToURL:(NSURL *)url
           options:(NSDataWritingOptions)opts
             error:(NSError **)errorPtr;

func writeToURL(NSURL, options: NSDataWritingOptions) throws
```

```
do {
    try data.writeToURL(url, options:[])
} catch {
    self.window.presentError(error as NSError)
}
```

Error Guidelines

NSError guidelines apply

Error Guidelines

NSError guidelines apply

Use NSError and Swift error handling for runtime errors

- File not found
- Out of disk space
- Unreachable network

Error Guidelines

NSError guidelines apply

Use NSError and Swift error handling for runtime errors

- File not found
- Out of disk space
- Unreachable network

Exceptions are still used for programming errors

- Array index out of bounds
- Assertion failures

Naming Cleanup

Naming Cleanup

```
typedef enum {
   NSLeftTextAlignment,
   NSRightTextAlignment,
   NSCenterTextAlignment,
   NSJustifiedTextAlignment,
   NSNaturalTextAlignment
} NSTextAlignment;
```

Naming Cleanup

```
typedef enum {
   NSLeftTextAlignment,
   NSRightTextAlignment,
   NSCenterTextAlignment,
   NSJustifiedTextAlignment,
   NSNaturalTextAlignment
} NSTextAlignment;
```

```
NSTextAlignmentLeft
NSTextAlignmentRight
NSTextAlignmentCenter
NSTextAlignmentJustified
NSTextAlignmentNatural
```

Naming Cleanup

```
typedef enum {
   NSLeftTextAlignment,
   NSRightTextAlignment,
   NSCenterTextAlignment,
   NSJustifiedTextAlignment,
   NSNaturalTextAlignment
} NSTextAlignment;
```

→

```
NSTextAlignmentLeft
NSTextAlignmentRight
NSTextAlignmentCenter
NSTextAlignmentJustified
NSTextAlignmentNatural
```

```
NSTextAlignment.LeftTextAlignment
```

Naming Cleanup

```
typedef enum {
   NSLeftTextAlignment,                 NSTextAlignmentLeft
   NSRightTextAlignment,                NSTextAlignmentRight
   NSCenterTextAlignment,          ⟶    NSTextAlignmentCenter
   NSJustifiedTextAlignment,            NSTextAlignmentJustified
   NSNaturalTextAlignment               NSTextAlignmentNatural
} NSTextAlignment;

NSTextAlignment.LeftTextAlignment  ⟶  NSTextAlignment.Left
```

Naming Cleanup

```
typedef enum {
   NSLeftTextAlignment,                 NSTextAlignmentLeft
   NSRightTextAlignment,                NSTextAlignmentRight
   NSCenterTextAlignment,          ⟶    NSTextAlignmentCenter
   NSJustifiedTextAlignment,            NSTextAlignmentJustified
   NSNaturalTextAlignment               NSTextAlignmentNatural
} NSTextAlignment;

NSTextAlignment.LeftTextAlignment  ⟶  NSTextAlignment.Left
```

And many more!

Swiftification

What's New in Swift	Presidio	Tuesday 11:00AM
Swift and Objective-C Interoperability	Mission	Tuesday 1:30PM

AppKit

AppKit

Trackpad

Full Screen

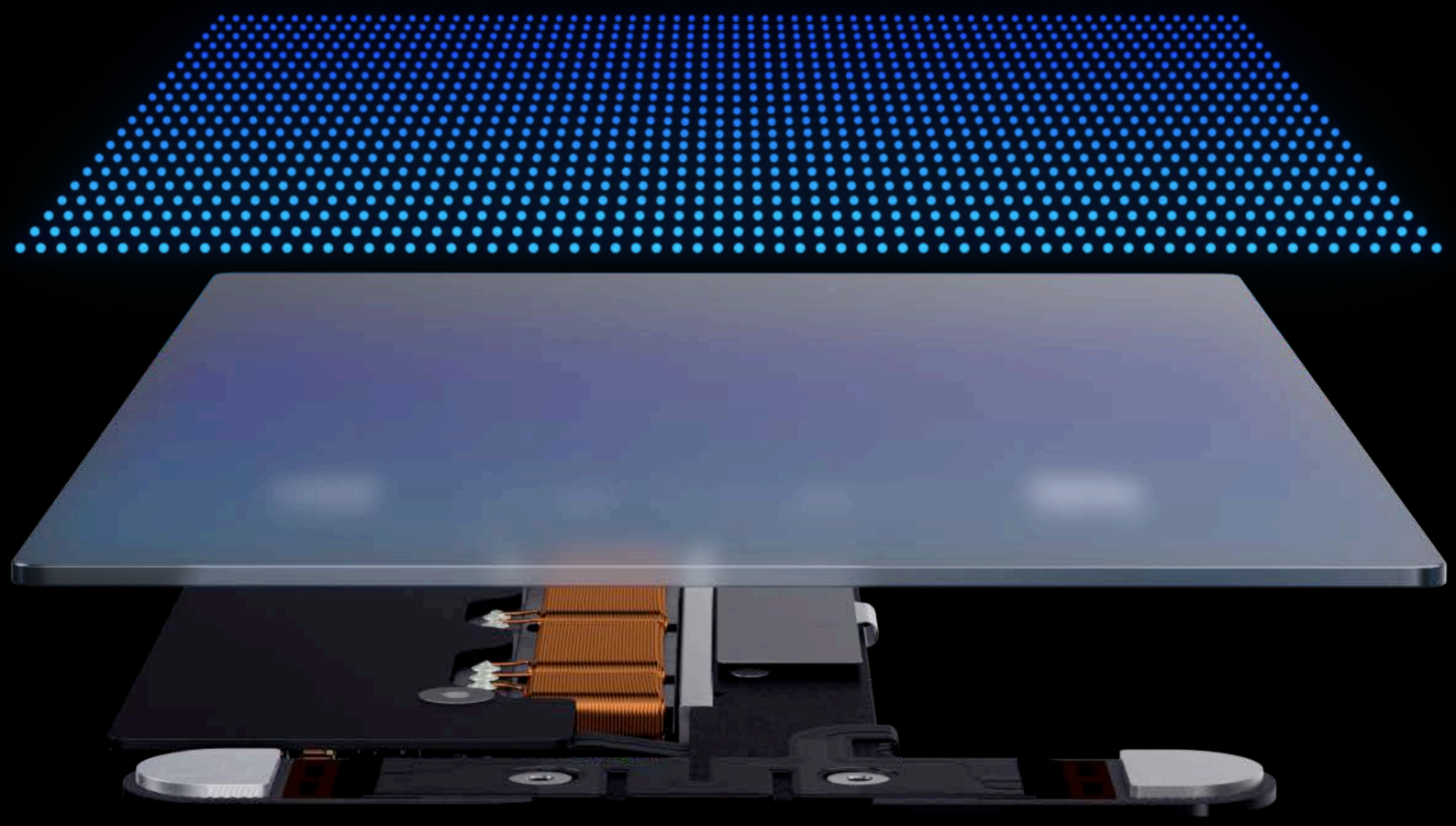
Auto Layout

Collection View

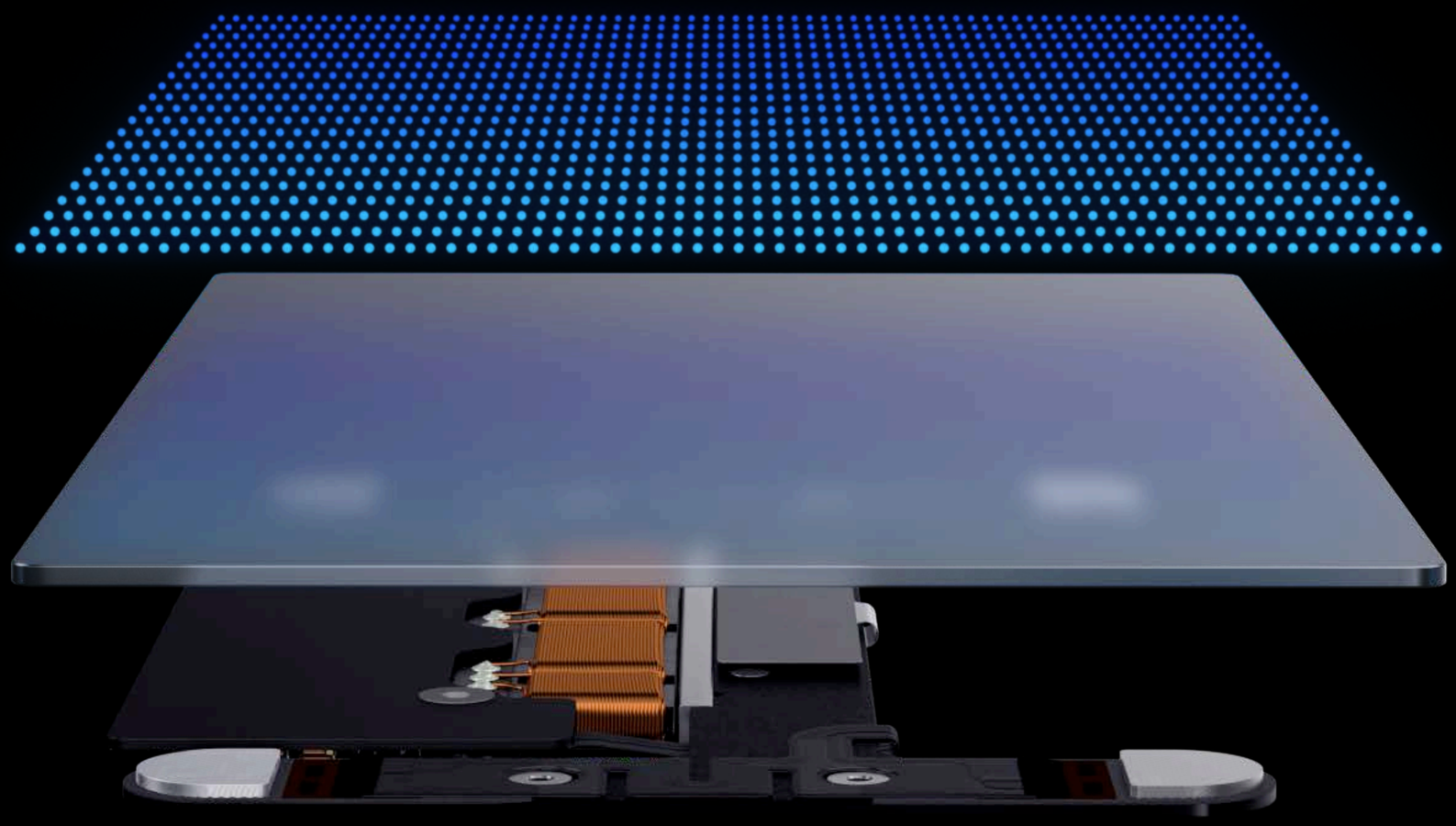
Text

Visual Atomicity

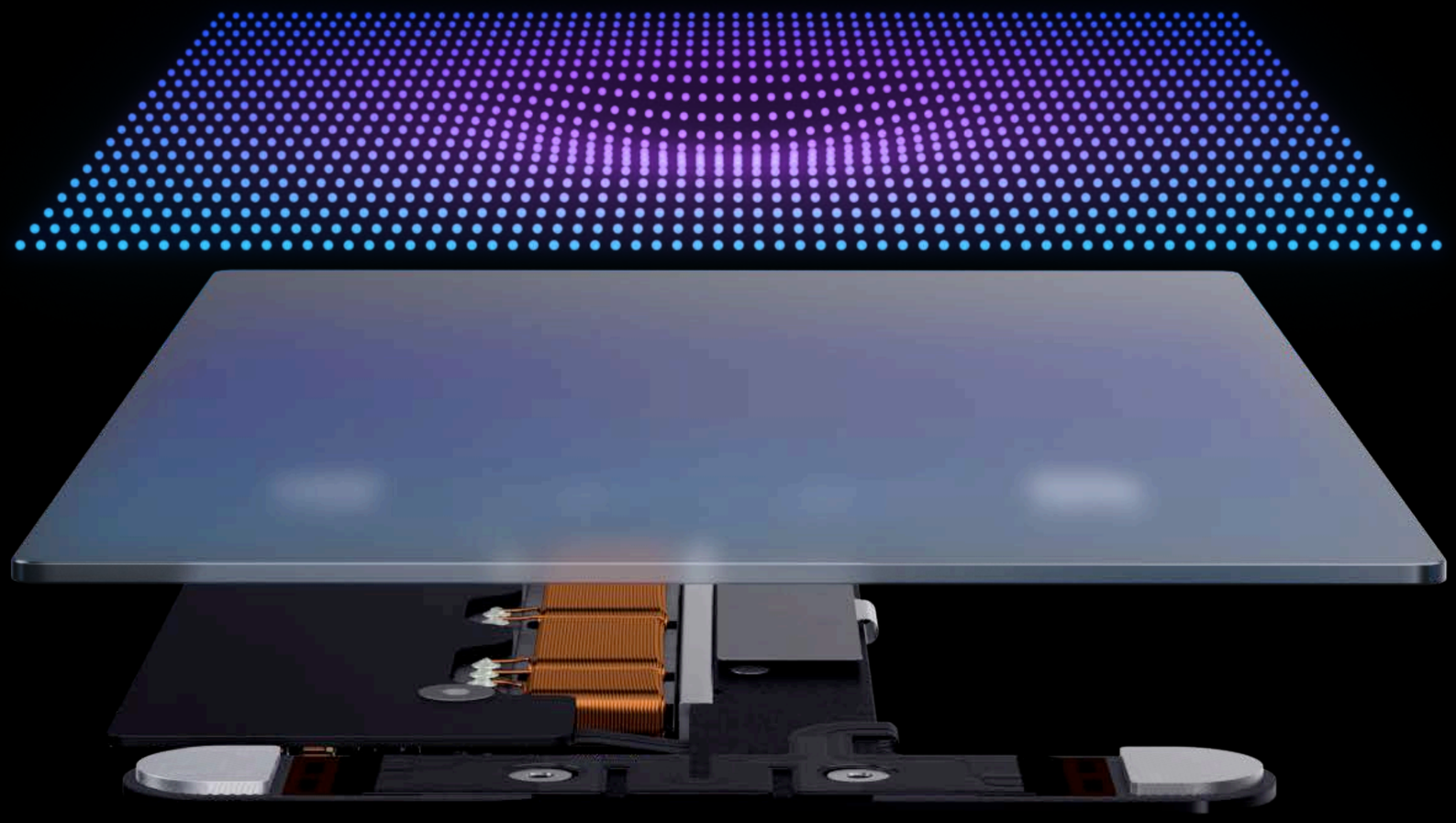
Trackpad



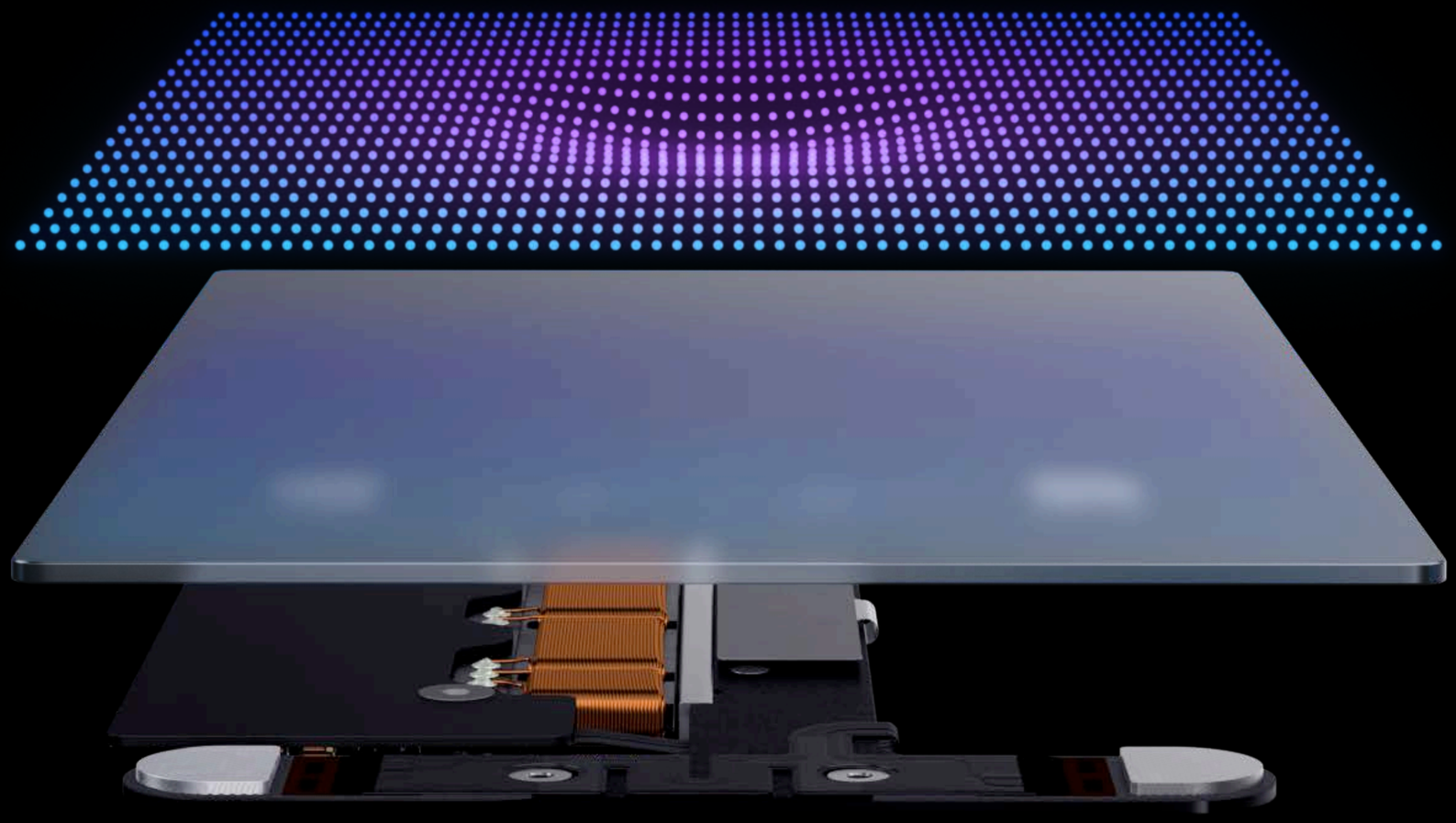
Trackpad



Trackpad



Trackpad




Messages File Edit View Buddies Window Help
Thu 9:41 AM
Search
Julie Harris 9:41 AM
This Saturday, 9pm
The gang Yesterday
Meet us outside the theater at 9. We have tickets already so…
The team Yesterday
Summer Friday tomorrow, lets do something after work!
Tasha Kelter Yesterday
I got it on sale, I think there are still some left. It was the loc…
Jennifer Wright Yesterday
I think I am staying in tonight. I've got an early flight tomor…
Steve Lemay Yesterday
Does anyone have a power drill I can borrow? Trying to mou…
Jon Negroni Yesterday
Did you catch that game last night!? What an amazing ca…
Sue Zeifman 10/14/14
Where did you go on vacation? I saw the pictures, and they…
To: Julie Harris
Details
Today, 9:35 AM
So I have some big news…
Tim proposed!
I know, he caught me
house.
We're having a little g
to celebrate.
Can you make it?
This Saturday, 9pm
iMessage
New Event
Location
Saturday, October 18, 2014
all-day
8 PM
9 PM
New Event
10 PM
11 PM
AM
Details
Cancel
Add to Calendar
MacBook


10×
01:00
-00:36
MacBook

Preview File Edit View Go Tools Window Help

Thu 9:41 AM

School of Rock consent form — Edited

Search

Trackpad Camera

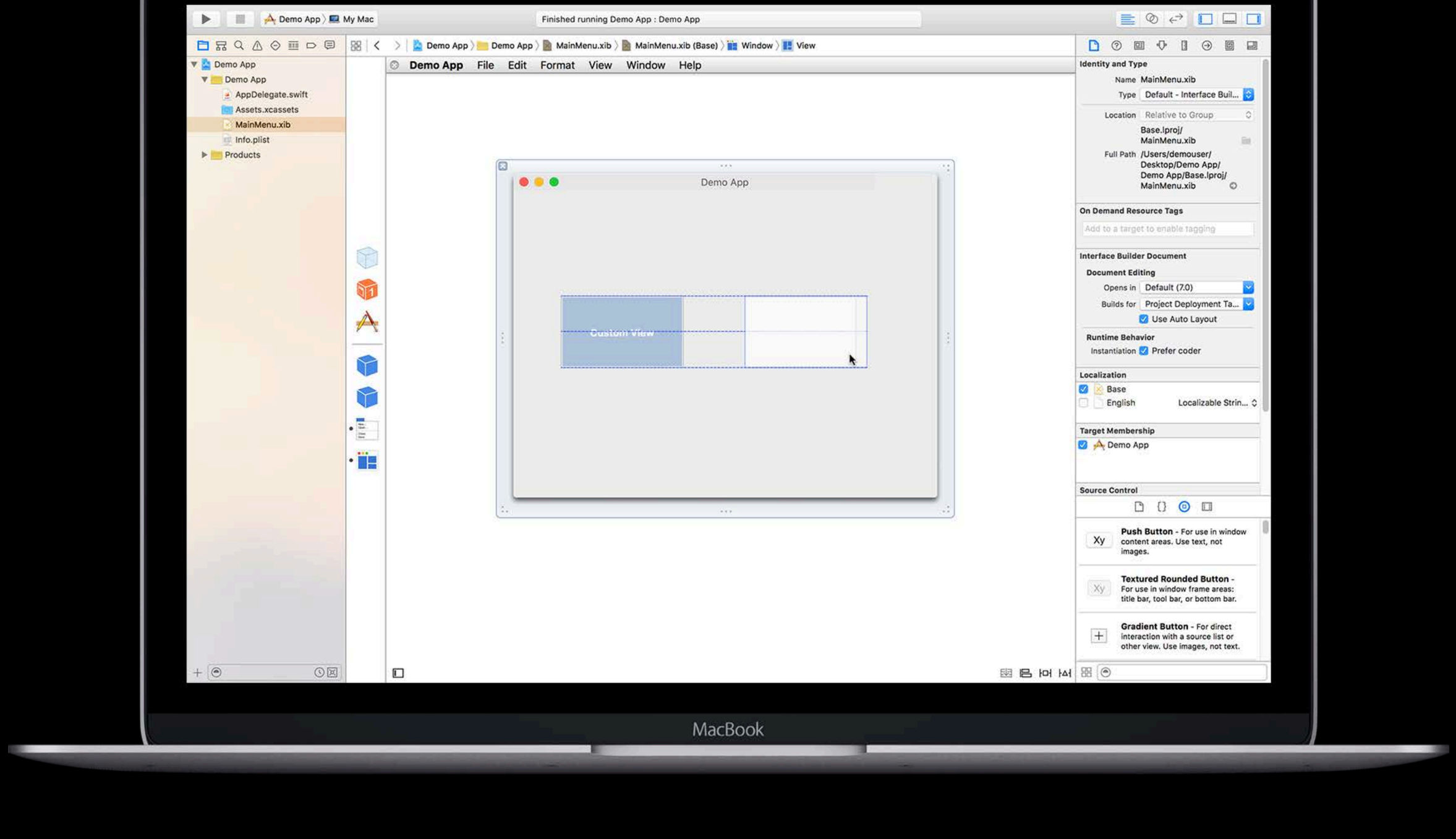
Sign your name on the trackpad.

Clear Cancel Done

mley **on behalf of** Miles Bromley.

dge receipt and agreement with the School of Rock
ivers.

MacBook


Demo App
My Mac
Finished running Demo App : Demo App
Demo App
Demo App
MainMenu.xib
MainMenu.xib (Base)
Window
View
Demo App
Demo App
AppDelegate.swift
Assets.xcassets
MainMenu.xib
Info.plist
Products
Demo App File Edit Format View Window Help
Demo App
Custom View
Identity and Type
Name MainMenu.xib
Type Default - Interface Buil...
Location Relative to Group
Base.lproj/
MainMenu.xib
Full Path /Users/demouser/
Desktop/Demo App/
Demo App/Base.lproj/
MainMenu.xib
On Demand Resource Tags
Add to a target to enable tagging
Interface Builder Document
Document Editing
Opens in Default (7.0)
Builds for Project Deployment Ta...
Use Auto Layout
Runtime Behavior
Instantiation Prefer coder
Localization
Base
English
Localizable Strin...
Target Membership
Demo App
Source Control
Xy
Push Button - For use in window content areas. Use text, not images.
Xy
Textured Rounded Button - For use in window frame areas: title bar, tool bar, or bottom bar.
Gradient Button - For direct interaction with a source list or other view. Use images, not text.
MacBook

Force Touch

APIs introduced in 10.10.3

Force Touch

APIs introduced in 10.10.3

Accelerator controls

```
NSButtonType.AcceleratorButton, .MultiLevelAcceleratorButton
NSSegmentSwitchTracking.MomentaryAccelerator
```

Force Touch

APIs introduced in 10.10.3

Accelerator controls

```
NSButtonType.AcceleratorButton, .MultiLevelAcceleratorButton
NSSegmentSwitchTracking.MomentaryAccelerator
```

New event type for pressure

```
NSEventType.EventTypePressure
```

Force Touch

APIs introduced in 10.10.3

Accelerator controls

```
NSButtonType.AcceleratorButton, .MultiLevelAcceleratorButton
NSSegmentSwitchTracking.MomentaryAccelerator
```

New event type for pressure

```
NSEventType.EventTypePressure
```

NSResponder and NSGestureRecognizer

```
func pressureChangeWithEvent(NSEvent)
```

Force Touch

APIs introduced in 10.11

Class for customizing pressure configuration

```
NSPressureConfiguration
```

Force Touch

APIs introduced in 10.11

Class for customizing pressure configuration

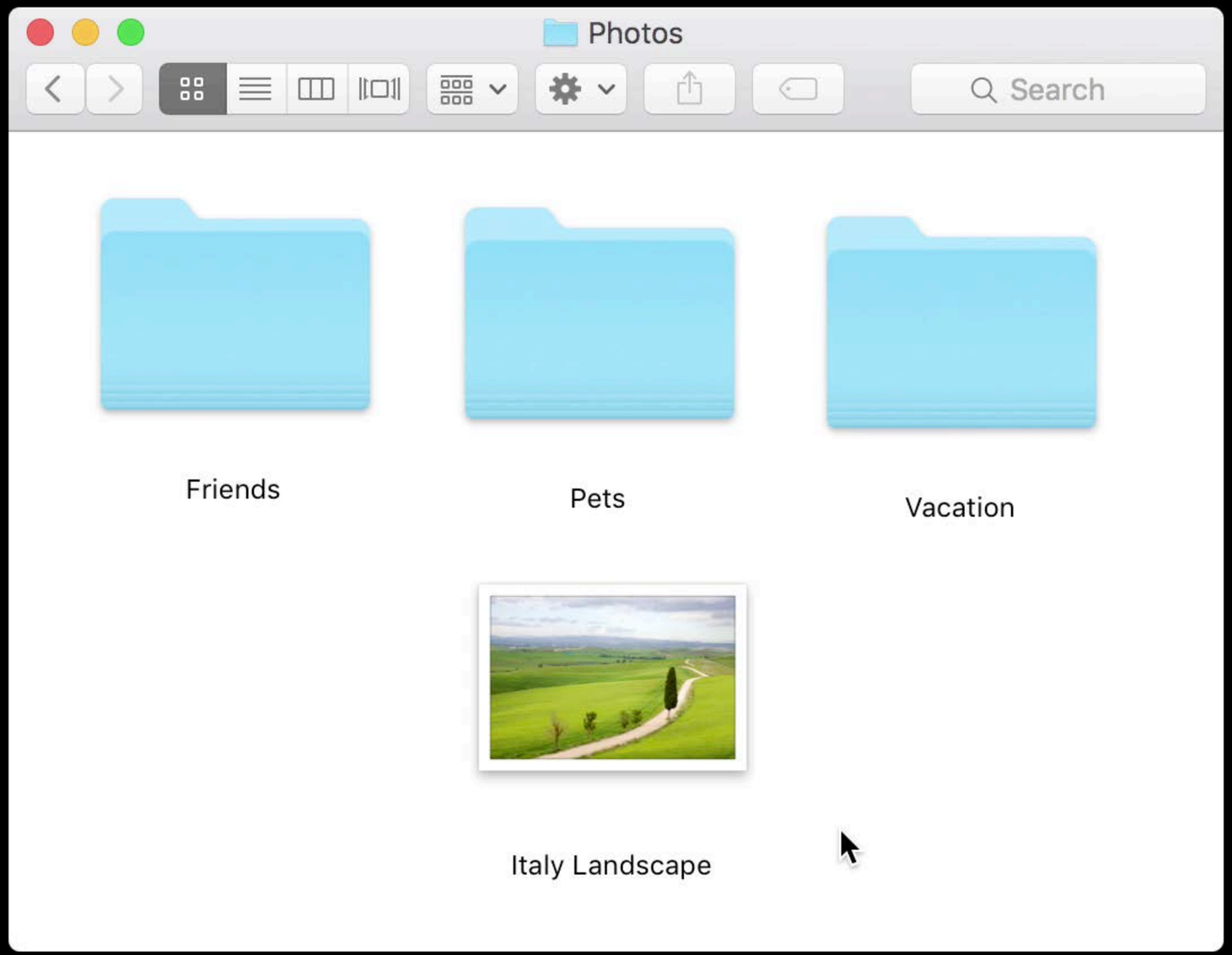
`NSPressureConfiguration`

Classes for providing haptic feedback

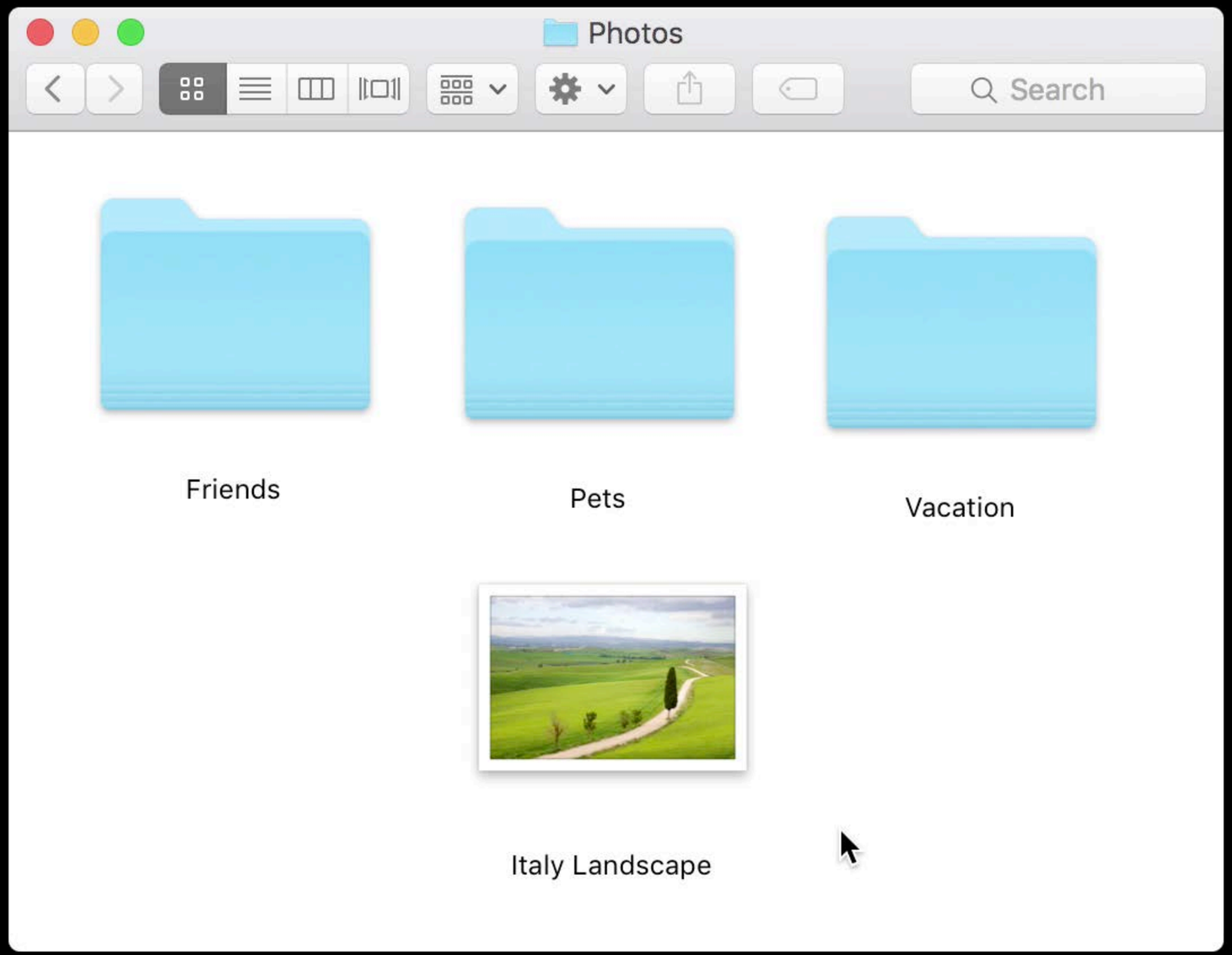
`NSHapticFeedbackManager`

`NSAlignmentFeedbackFilter`

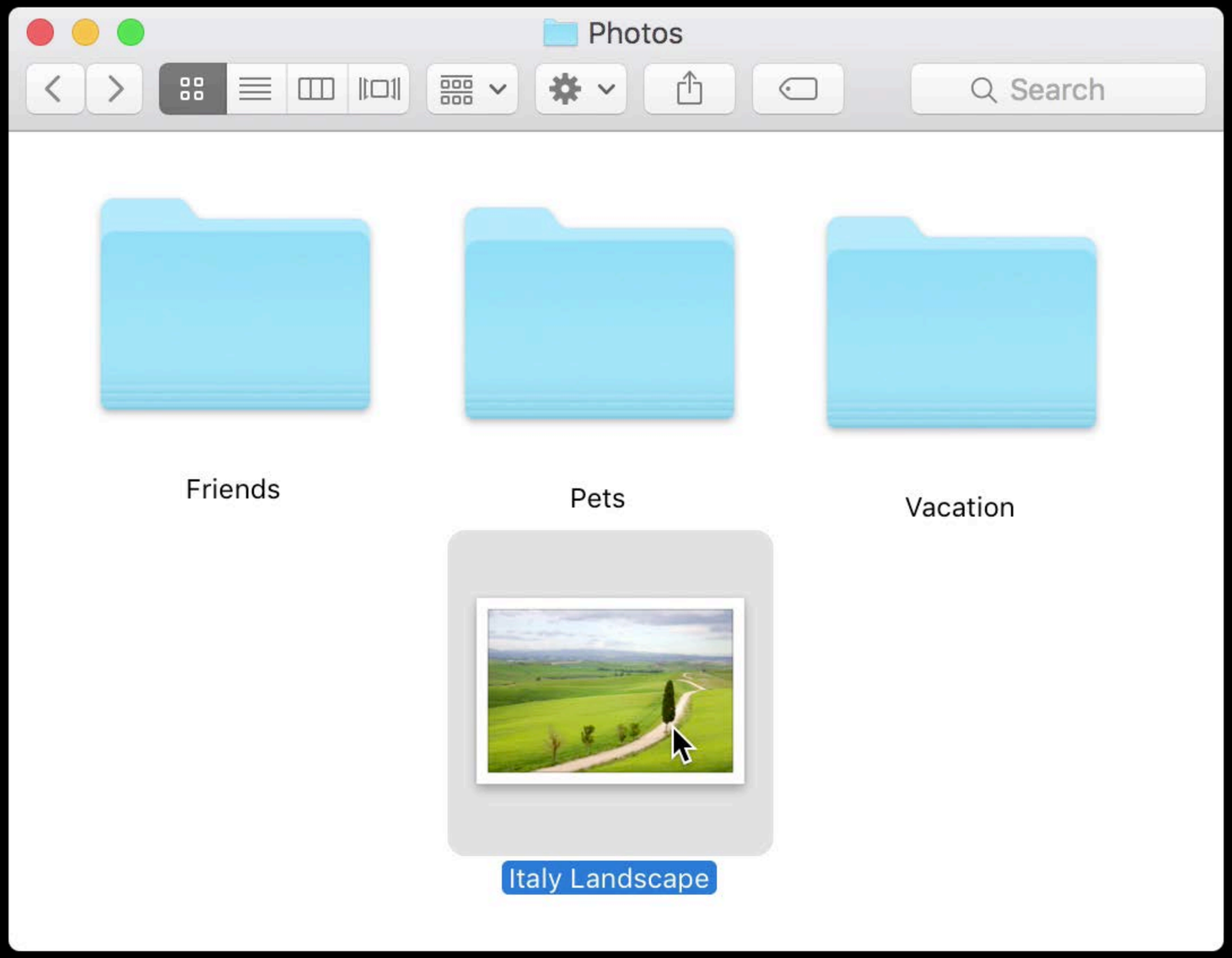
Spring Loading



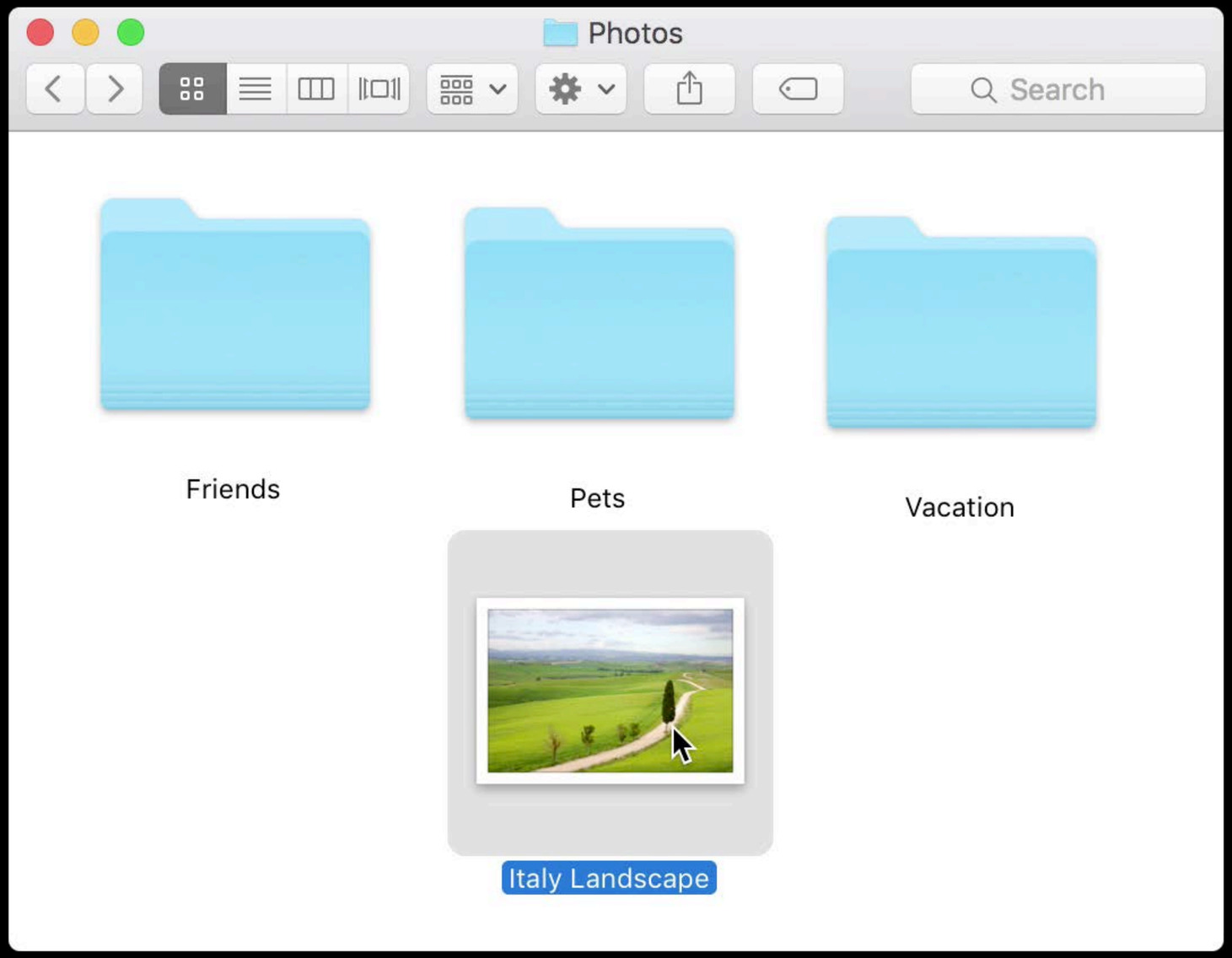
Spring Loading



Spring Loading on Force Click



Spring Loading on Force Click



Spring Loading

Spring Loading

Simple spring loading on NSButton, NSSegmentedControl

```
var springLoaded: Bool
```

Spring Loading

Simple spring loading on NSButton, NSSegmentedControl

```
var springLoaded: Bool
```

- Action sent on hover or force click

Spring Loading

Simple spring loading on NSButton, NSSegmentedControl

```
var springLoaded: Bool
```

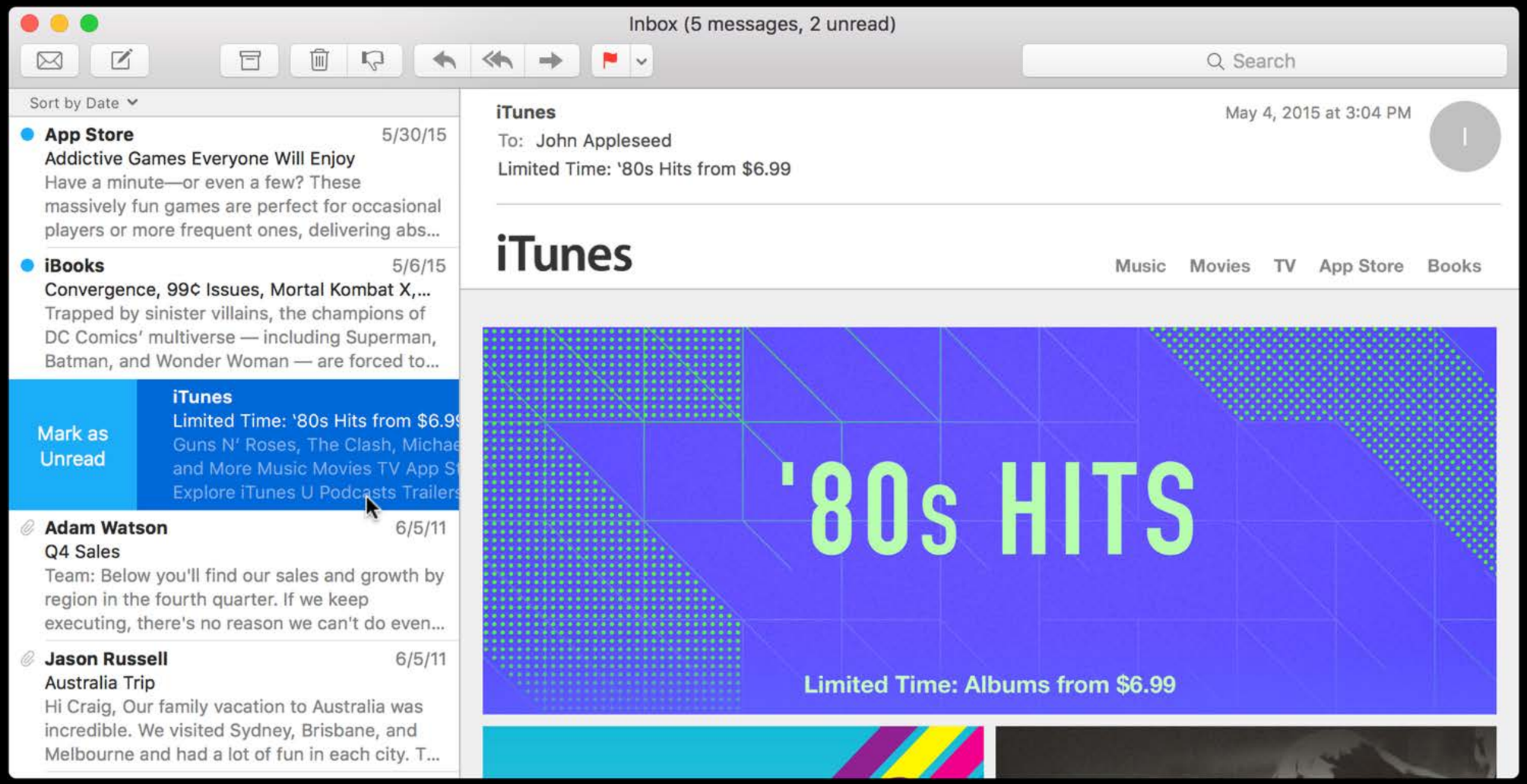
- Action sent on hover or force click

Spring loading on arbitrary destinations

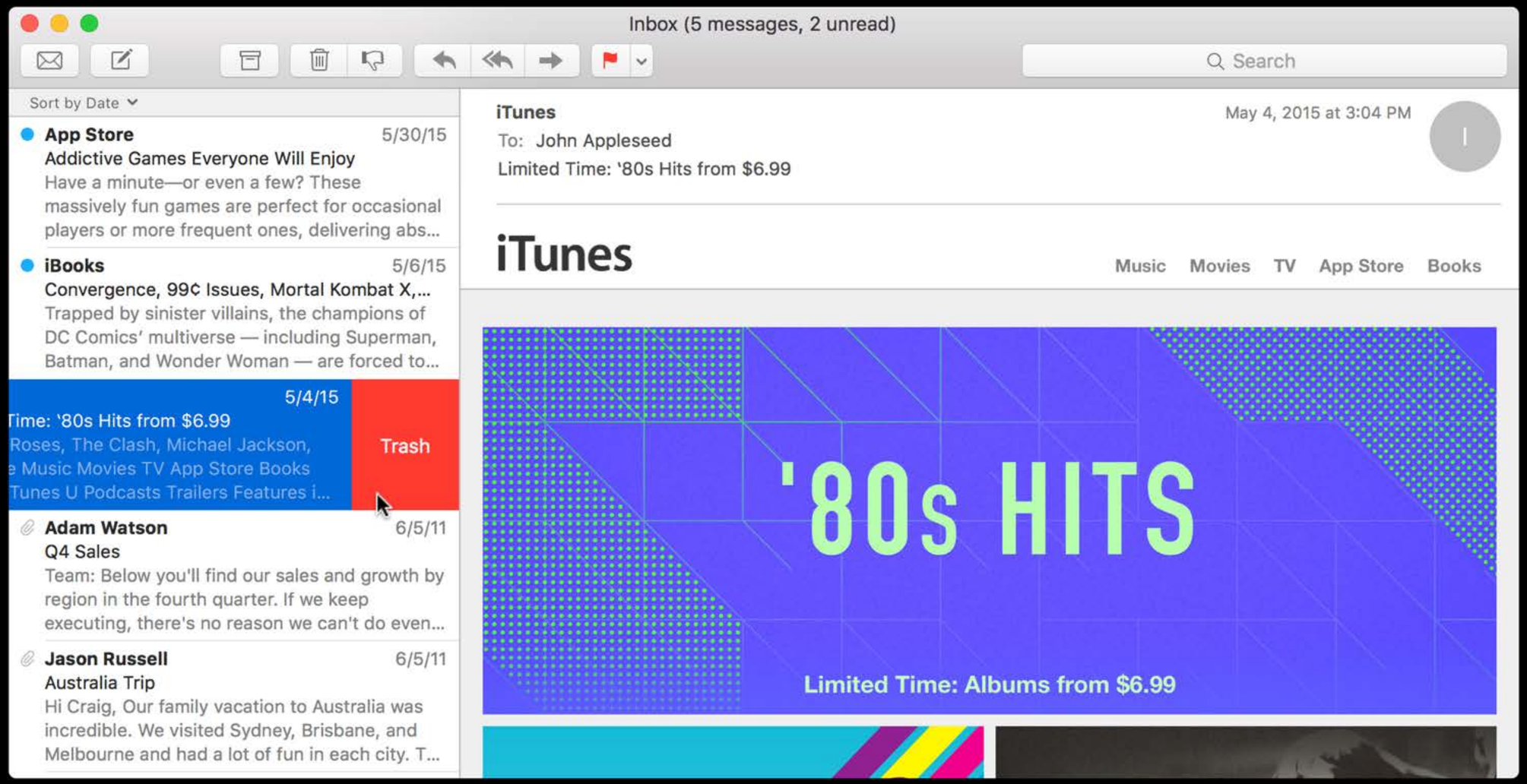
```
protocol NSSpringLoadingDestination
```

Swipe-to-Delete

Swipe-to-Delete



Swipe-to-Delete



Swipe-to-Delete

```
protocol NSTableViewDelegate {
  ...
  optional func tableView(NSTableView,
                  rowActionsForRow: Int,
                  edge: NSTableRowActionEdge) -> [NSTableViewRowAction]
}
```

Swipe-to-Delete

```
protocol NSTableViewDelegate {
  ...
  optional func tableView(NSTableView,
                  rowActionsForRow: Int,
                  edge: NSTableRowActionEdge) -> [NSTableViewRowAction]
}
```

Swipe-to-Delete

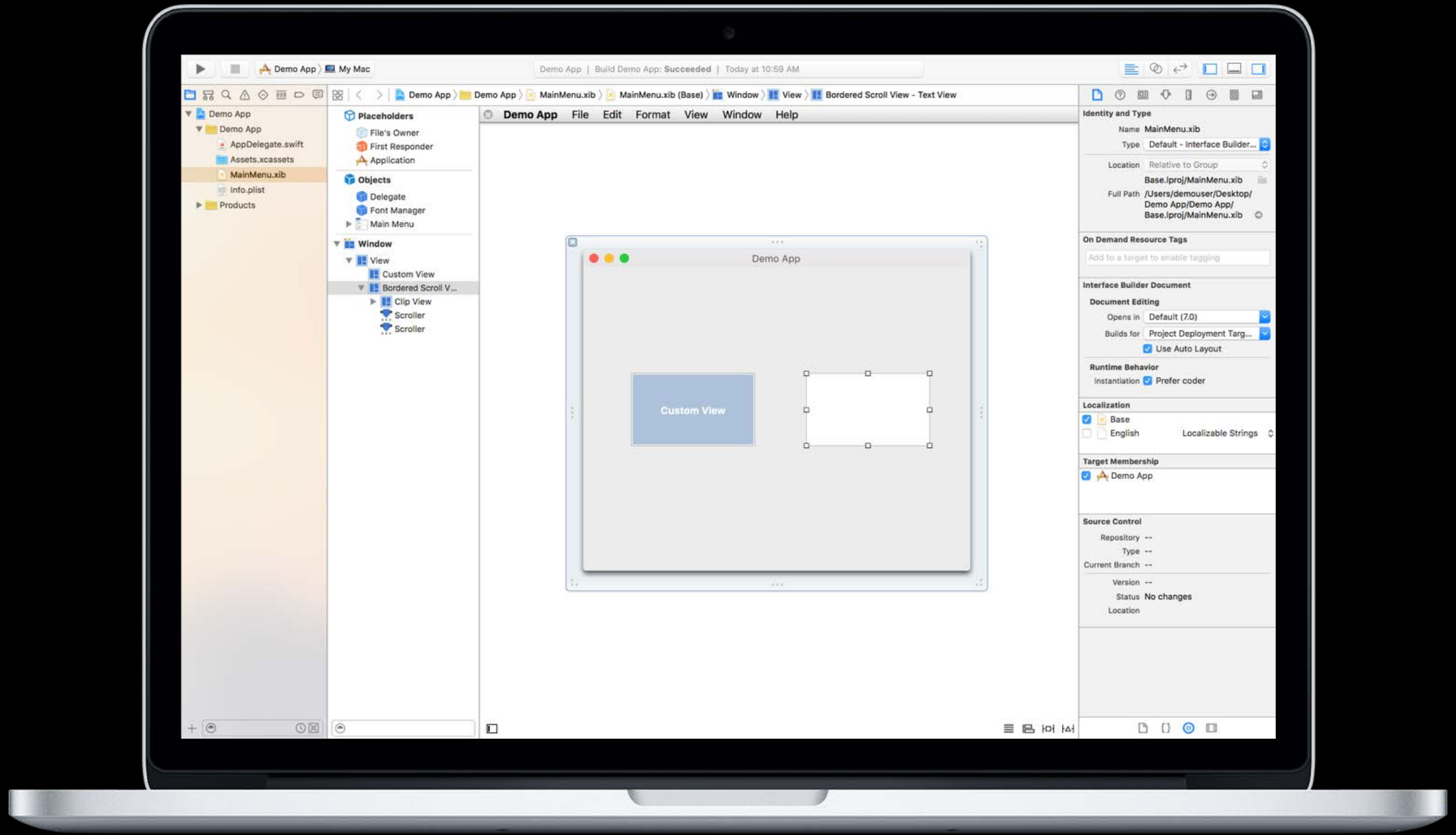
```
protocol NSTableViewDelegate {
  ...
  optional func tableView(NSTableView,
                rowActionsForRow: Int,
                edge: NSTableRowActionEdge) -> [NSTableViewRowAction]
}

class NSTableViewRowAction : NSObject {
  init(style: NSTableViewRowActionStyle,
       title: String,
       handler: ((NSTableViewRowAction, Int) -> Void))
  ...
}
```

Trackpad

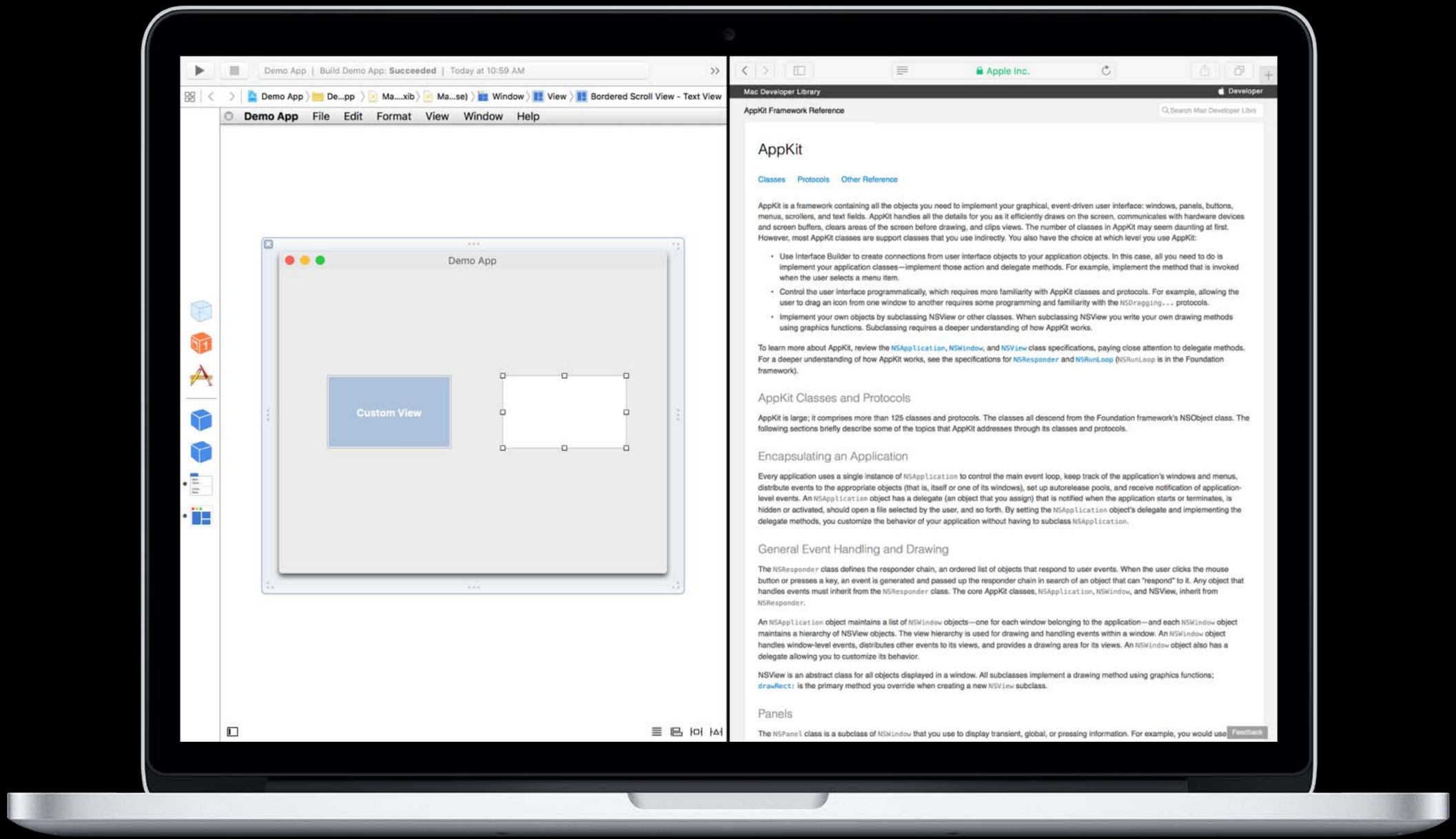
Adopting New Trackpad Features	Mission	Thursday 10:00AM

Full Screen

Full Screen


Demo App
Custom View
Identity and Type
Name MainMenu.xib
Type Default - Interface Builder...
Location Relative to Group
Base.lproj/MainMenu.xib
Full Path /Users/demouser/Desktop/Demo App/Demo App/Base.lproj/MainMenu.xib
On Demand Resource Tags
Interface Builder Document
Document Editing
Opens in Default (7.0)
Builds for Project Deployment Targ...
Use Auto Layout
Runtime Behavior
Instantiation Prefer coder
Localization
Base
English
Localizable Strings
Target Membership
Demo App
Source Control
Repository --
Type --
Current Branch --
Version --
Status No changes
Location

Full Screen Split View



Full Screen

Tiling is automatic for many windows

Full Screen

Tiling is automatic for many windows

Resizable windows are eligible for tiling

- Whether they are full-screen capable or not

Full Screen

Tiling is automatic for many windows

Resizable windows are eligible for tiling

- Whether they are full-screen capable or not

API for opting windows in or out of tiling

```
struct NSWindowCollectionBehavior : OptionSetType {
  ...
```

Full Screen

Tiling is automatic for many windows

Resizable windows are eligible for tiling

- Whether they are full-screen capable or not

API for opting windows in or out of tiling

```
struct NSWindowCollectionBehavior : OptionSetType {
  ...
  static var FullScreenAllowsTiling: NSWindowCollectionBehavior { get }
  static var FullScreenDisallowsTiling: NSWindowCollectionBehavior { get }
}
```

Full Screen

Tiling is automatic for many windows

Resizable windows are eligible for tiling

- Whether they are full-screen capable or not

API for opting windows in or out of tiling

```
struct NSWindowCollectionBehavior : OptionSetType {
  ...
  static var FullScreenAllowsTiling: NSWindowCollectionBehavior { get }
  static var FullScreenDisallowsTiling: NSWindowCollectionBehavior { get }
  static var FullScreenPrimary: NSWindowCollectionBehavior { get } // 10.7
}
```

Window

Title: Demo Window

Autosave: Autosave Name

Appearance
- [x] Title Bar
- [ ] Unified Title And Toolbar
- [ ] Full Size Content View
- [x] Shadow
- [ ] Textured

Controls
- [x] Close
- [x] Minimize
- [x] Resize
- [x] Toolbar Button

Behavior
- [x] Restorable
- [x] Visible At Launch
- [ ] Hide On Deactivate
- [ ] Release When Closed
- [ ] Always Display Tooltips
- [ ] Recalculates View Loop

Spaces: Inferred Behavior

Exposé: Inferred Behavior

Cycling: Inferred Behavior

Full Screen: Primary Window

Tiling: Allows Tiling

Animation: Inferred Behavior

Appearance: Default (Aqua)

Memory
- [x] Deferred
- [x] One Shot

Buffered

Window

Title: Demo Window

Autosave: Autosave Name

Appearance
- [x] Title Bar
- [ ] Unified Title And Toolbar
- [ ] Full Size Content View
- [x] Shadow
- [ ] Textured

Controls
- [x] Close
- [x] Minimize
- [x] Resize
- [x] Toolbar Button

Behavior
- [x] Restorable
- [x] Visible At Launch
- [ ] Hide On Deactivate
- [ ] Release When Closed
- [ ] Always Display Tooltips
- [ ] Recalculates View Loop

Spaces: Inferred Behavior

Exposé: Inferred Behavior

Full Screen: Primary Window

Tiling: Allows Tiling

Appearance: Default (Aqua)

Memory
- [x] Deferred
- [x] One Shot

Buffered

Full Screen

Tiling is automatic for many windows

Full Screen

Tiling is automatic for many windows

But AppKit checks to see that windows can coexist in the same screen

Full Screen

Tiling is automatic for many windows

But AppKit checks to see that windows can coexist in the same screen

Full Screen

Tiling is automatic for many windows

But AppKit checks to see that windows can coexist in the same screen

Full Screen

Tiling is automatic for many windows

But AppKit checks to see that windows can coexist in the same screen

Full Screen

Tiling is automatic for many windows

But AppKit checks to see that windows can coexist in the same screen

Full Screen

Facilities to enable your windows to resize gracefully

Full Screen

Facilities to enable your windows to resize gracefully

NSSplitViewItem with sidebar behavior

Full Screen

Facilities to enable your windows to resize gracefully

NSSplitViewItem with sidebar behavior

NSStackView automatic detaching of hidden views

Full Screen

Facilities to enable your windows to resize gracefully

NSSplitViewItem with sidebar behavior

NSStackView automatic detaching of hidden views

And more!

Full Screen

Facilities to enable your windows to resize gracefully

NSSplitViewItem with sidebar behavior

NSStackView automatic detaching of hidden views

And more!

Improving the Full Screen Window Experience Pacific Heights Thursday 2:30PM

Auto Layout

NSStackView

NSLayoutAnchor

NSLayoutGuide

NSStackView

NSStackView

Now on iOS as well!

NSStackView

New options for view distribution

```
var distribution: NSStackViewDistribution
```

NSStackView

New options for view distribution

```
var distribution: NSStackViewDistribution

enum NSStackViewDistribution : Int {
  case GravityAreas
```

NSStackView

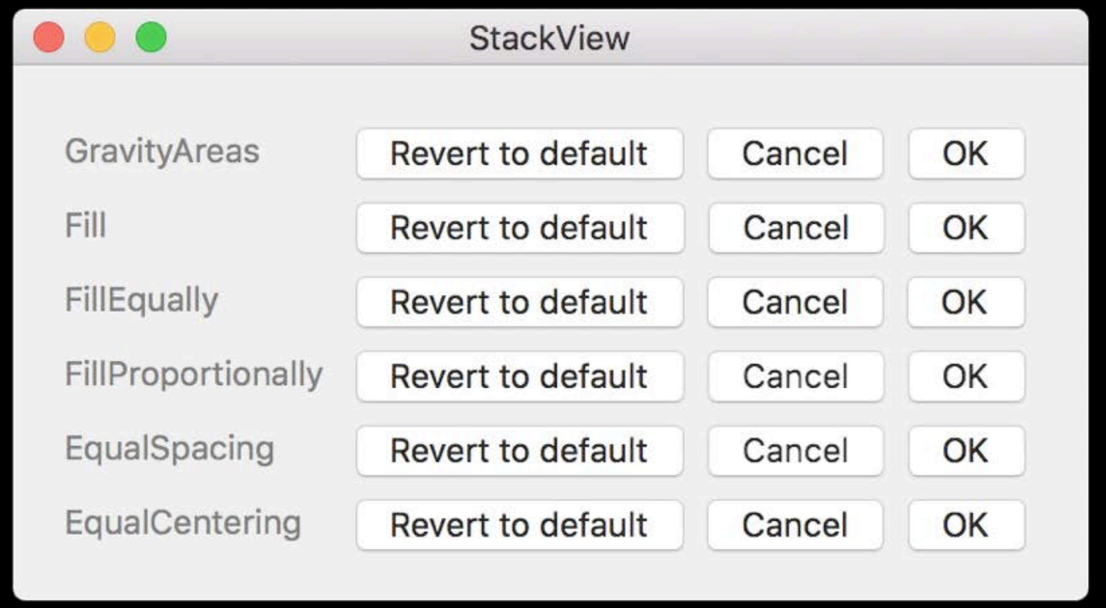
New options for view distribution

```
var distribution: NSStackViewDistribution

enum NSStackViewDistribution : Int {
  case GravityAreas
  case Fill
  case FillEqually
  case FillProportionally
  case EqualSpacing
  case EqualCentering
}
```

NSStackView

New options for view distribution

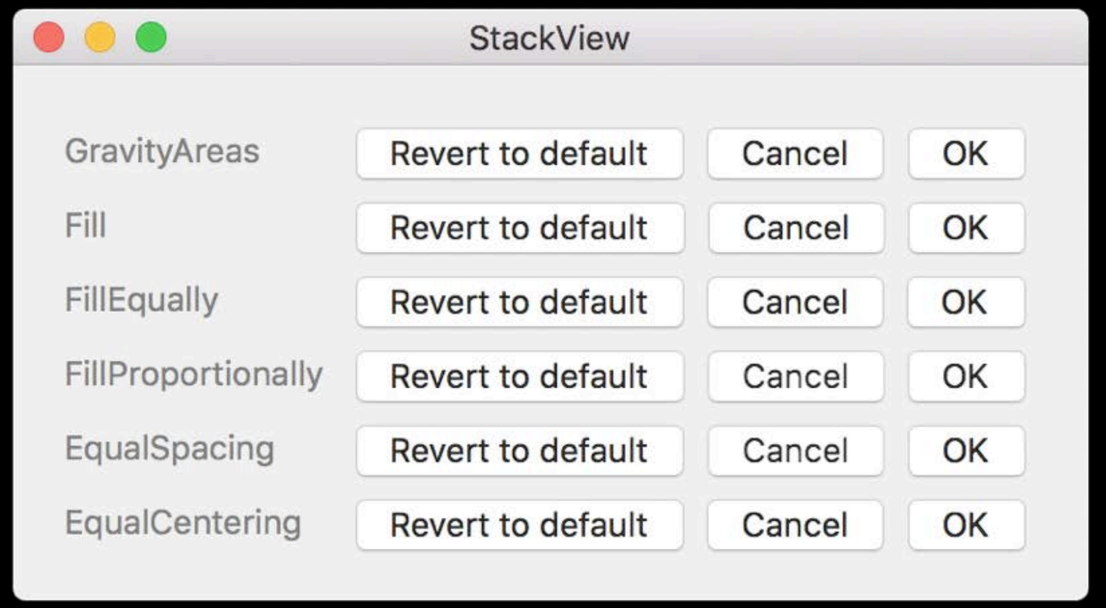
```
var distribution: NSStackViewDistribution

enum NSStackViewDistribution : Int {
  case GravityAreas
  case Fill
  case FillEqually
  case FillProportionally
  case EqualSpacing
  case EqualCentering
}
```



NSStackView

New options for view distribution

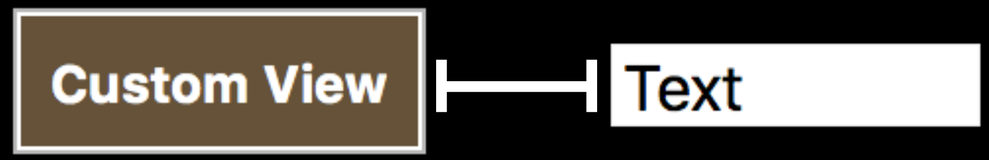
```
var distribution: NSStackViewDistribution

enum NSStackViewDistribution : Int {
  case GravityAreas
  case Fill
  case FillEqually
  case FillProportionally
  case EqualSpacing
  case EqualCentering
}
```



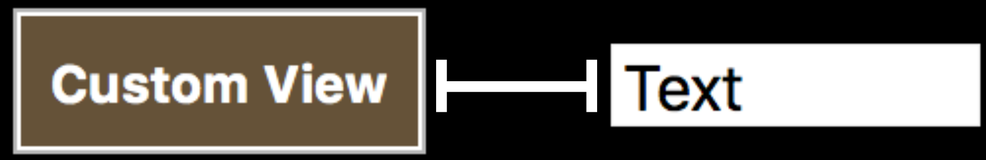
NSLayoutAnchor

NSLayoutAnchor

NSLayoutAnchor



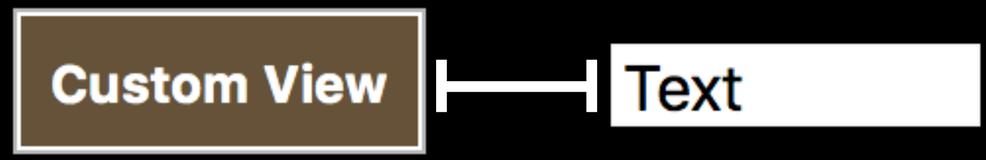
NSLayoutAnchor



Instead of

```
var constraint = NSLayoutConstraint(item: text,
                                    attribute: .Leading,
                                    relatedBy: .Equal,
                                    toItem: view,
                                    attribute: .Trailing,
                                    multiplier: 1.0,
                                    constant: padding)
```

NSLayoutAnchor



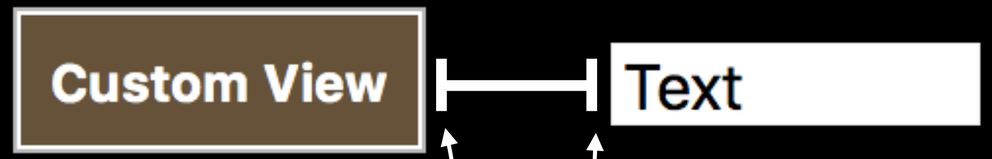
Instead of

```
var constraint = NSLayoutConstraint(item: text,
                                    attribute: .Leading,
                                    relatedBy: .Equal,
                                    toItem: view,
                                    attribute: .Trailing,
                                    multiplier: 1.0,
                                    constant: padding)
```

Can now write

```
var constraint =
        view.trailingAnchor.constraintEqualToAnchor(text.leadingAnchor,
                                                    constant: padding)
```

NSLayoutAnchor



Instead of

```
var constraint = NSLayoutConstraint(item: text,
                                    attribute: .Leading,
                                    relatedBy: .Equal,
                                    toItem: view,
                                    attribute: .Trailing,
                                    multiplier: 1.0,
                                    constant: padding)
```

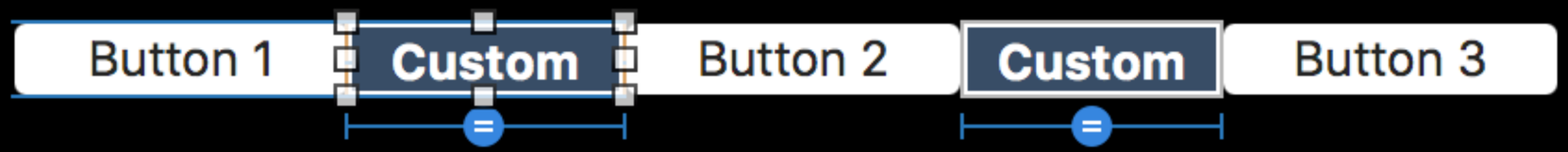
Can now write

```
var constraint =
        view.trailingAnchor.constraintEqualToAnchor(text.leadingAnchor,
                                                    constant: padding)
```

NSLayoutGuide

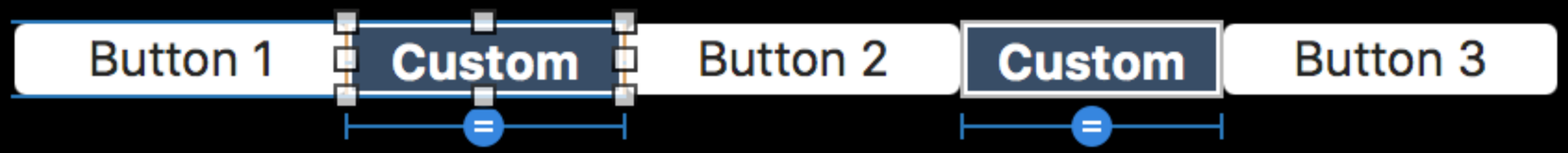
NSLayoutGuide

If you find yourself creating placeholder views for auto layout purposes:



NSLayoutGuide

If you find yourself creating placeholder views for auto layout purposes:



You can now instead use NSLayoutGuide

- A lightweight object that can participate in auto layout much like a view

Auto Layout

Mysteries of Auto Layout, Part 1	Presidio	Thursday 11:00AM
Mysteries of Auto Layout, Part 2	Presidio	Thursday 1:30PM
Improving the Full Screen Window Experience	Pacific Heights	Thursday 2:30PM

NSCollectionView

NSCollectionView

More scalable

Data source-based loading

Heterogeneous items

Optional grouping with headers/footers

Customizable layout

NSCollectionView

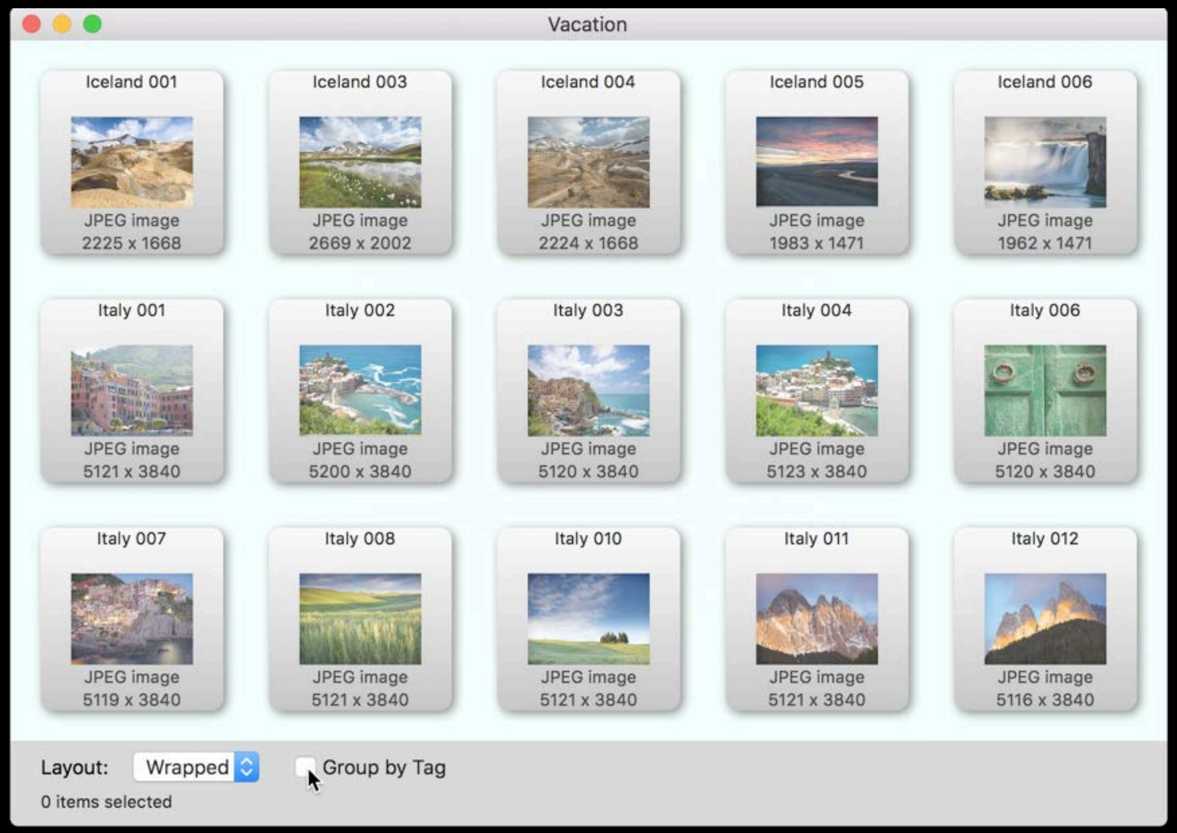
More scalable

Data source-based loading

Heterogeneous items

Optional grouping with headers/footers

Customizable layout



NSCollectionView

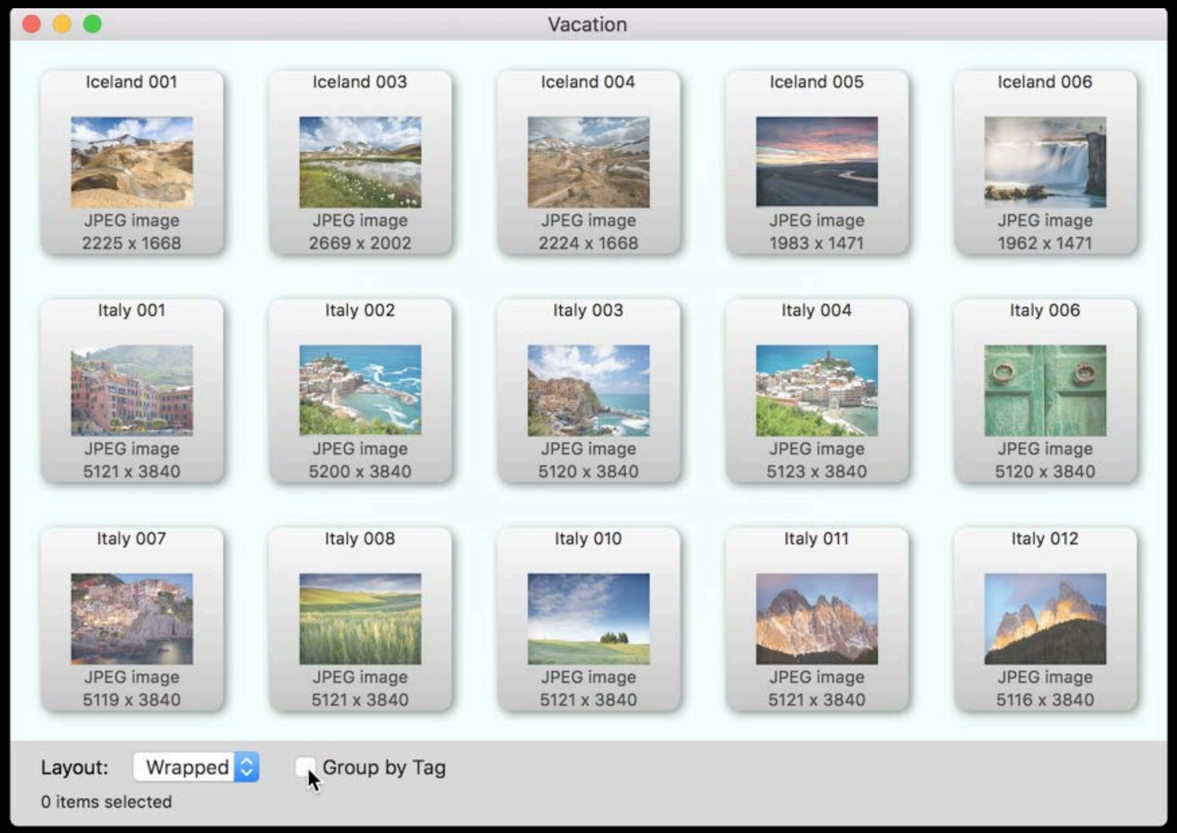
More scalable

Data source-based loading

Heterogeneous items

Optional grouping with headers/footers

Customizable layout



NSCollectionView

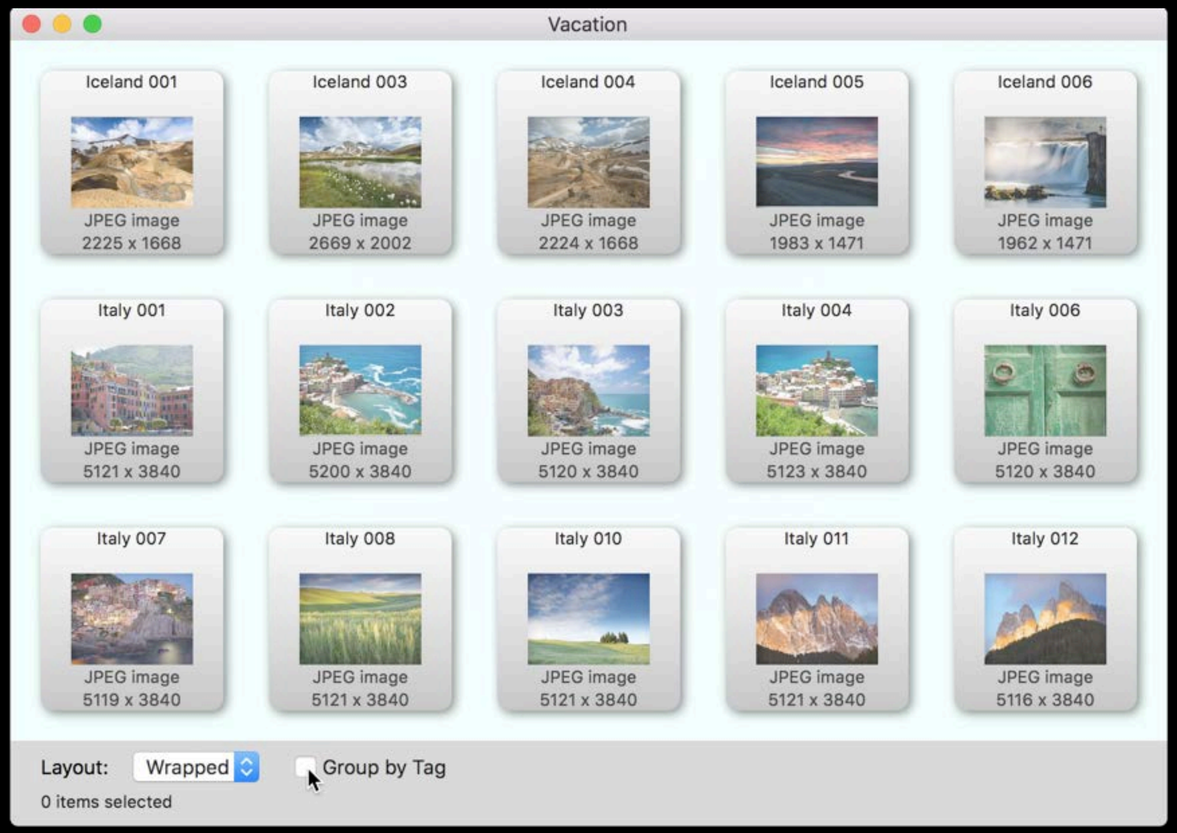
More scalable

Data source-based loading

Heterogeneous items

Optional grouping with headers/footers

Customizable layout



What's New in NSCollectionView Mission Thursday 4:30PM

Text

New system UI font

New APIs

Sample Text

10.11

Sample Text 10.11

Sample Text 10.10

Sample Text 10.11

Sample Text 10.10

Sample Text 10.0

Sample Text 10.11

Sample Text 10.10

Sample Text 10.0

Sample Text Mac OS 8

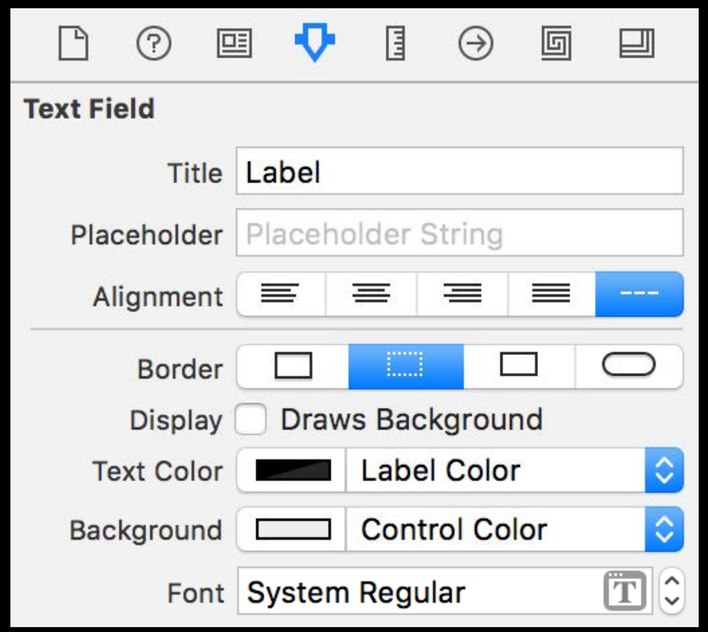
NSFont

Use appropriate "meta" font APIs, available since 10.0

```
class func systemFontOfSize(CGFloat) -> NSFont
class func boldSystemFontOfSize(CGFloat) -> NSFont
class func labelFontOfSize(CGFloat) -> NSFont
class func menuFontOfSize(CGFloat) -> NSFont
...
```

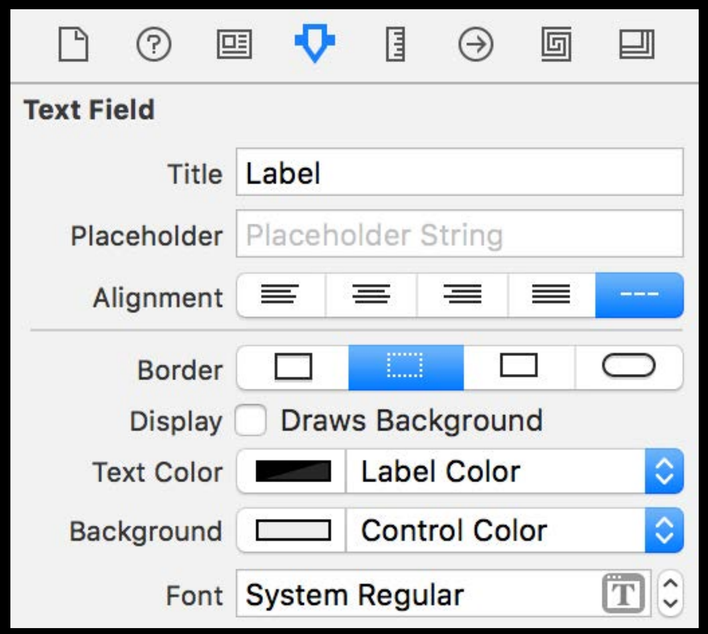
NSFont

Or use the “system” fonts in Xcode attributes inspector



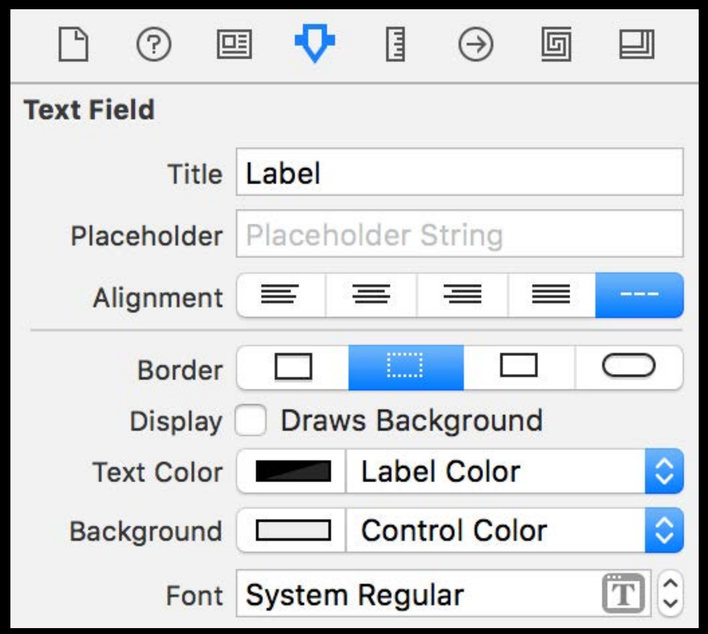
NSFont

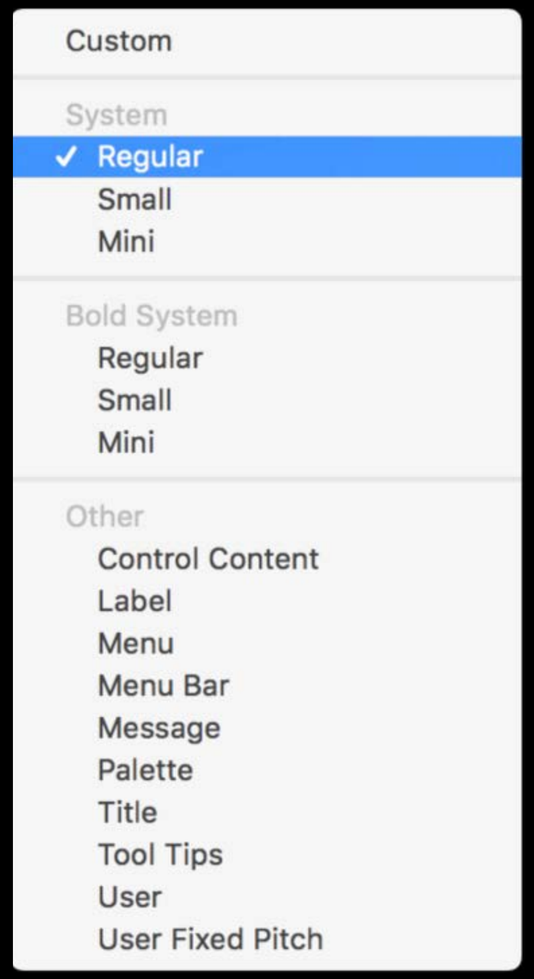
Or use the “system” fonts in Xcode attributes inspector



NSFont

Or use the "system" fonts in Xcode attributes inspector





NSFont

New API for system font at different weights

NSFont

New API for system font at different weights

```
class NSFont {
  ...
  class func systemFontOfSize(CGFloat, weight: CGFloat) -> NSFont
}
```

NSFont

New API for system font at different weights

```
class NSFont {
  ...
  class func systemFontOfSize(CGFloat, weight: CGFloat) -> NSFont
}
```

```
let NSFontWeightUltraLight: CGFloat
...
let NSFontWeightRegular: CGFloat
...
let NSFontWeightBlack: CGFloat
```

NSFont

New API for system font at different weights

```
class NSFont {
  ...
  class func systemFontOfSize(CGFloat, weight: CGFloat) -> NSFont
  class func monospacedDigitSystemFontOfSize(CGFloat, weight: CGFloat) ->
                                                                  NSFont
}
```

NSFont

New API for system font at different weights

```
class NSFont {
  ...
  class func systemFontOfSize(CGFloat, weight: CGFloat) -> NSFont
  class func monospacedDigitSystemFontOfSize(CGFloat, weight: CGFloat) ->
                                                                  NSFont
}

NSFont.systemFontOfSize(48,
                weight: NSFontWeightRegular)
```

NSFont

New API for system font at different weights

```
class NSFont {
  ...
  class func systemFontOfSize(CGFloat, weight: CGFloat) -> NSFont
  class func monospacedDigitSystemFontOfSize(CGFloat, weight: CGFloat) ->
                                                                  NSFont
}

NSFont.systemFontOfSize(48,
                weight: NSFontWeightRegular)
```

0123456789

NSFont

New API for system font at different weights

```
class NSFont {
  ...
  class func systemFontOfSize(CGFloat, weight: CGFloat) -> NSFont
  class func monospacedDigitSystemFontOfSize(CGFloat, weight: CGFloat) ->
                                                                  NSFont
}

NSFont.systemFontOfSize(48,
                weight: NSFontWeightRegular)

NSFont.monospacedDigitSystemFontOfSize(48,
                weight: NSFontWeightRegular)
```

0123456789

NSFont

New API for system font at different weights

```
class NSFont {
  ...
  class func systemFontOfSize(CGFloat, weight: CGFloat) -> NSFont
  class func monospacedDigitSystemFontOfSize(CGFloat, weight: CGFloat) ->
                                                                  NSFont
}

NSFont.systemFontOfSize(48,
                weight: NSFontWeightRegular)

NSFont.monospacedDigitSystemFontOfSize(48,
                weight: NSFontWeightRegular)
```

0123456789

0123456789

0000000000
1111111111
2222222222
3333333333
4444444444
5555555555
6666666666
7777777777
8888888888
9999999999

0000000000
1111111111
2222222222
3333333333
4444444444
5555555555
6666666666
7777777777
8888888888
9999999999

0000000000
1111111111
2222222222
3333333333
4444444444
5555555555
6666666666
7777777777
8888888888
9999999999

New System UI Font

Introducing the New System Fonts	Presidio	Friday 2:30PM

New APIs

New functionality and parity with iOS

New APIs

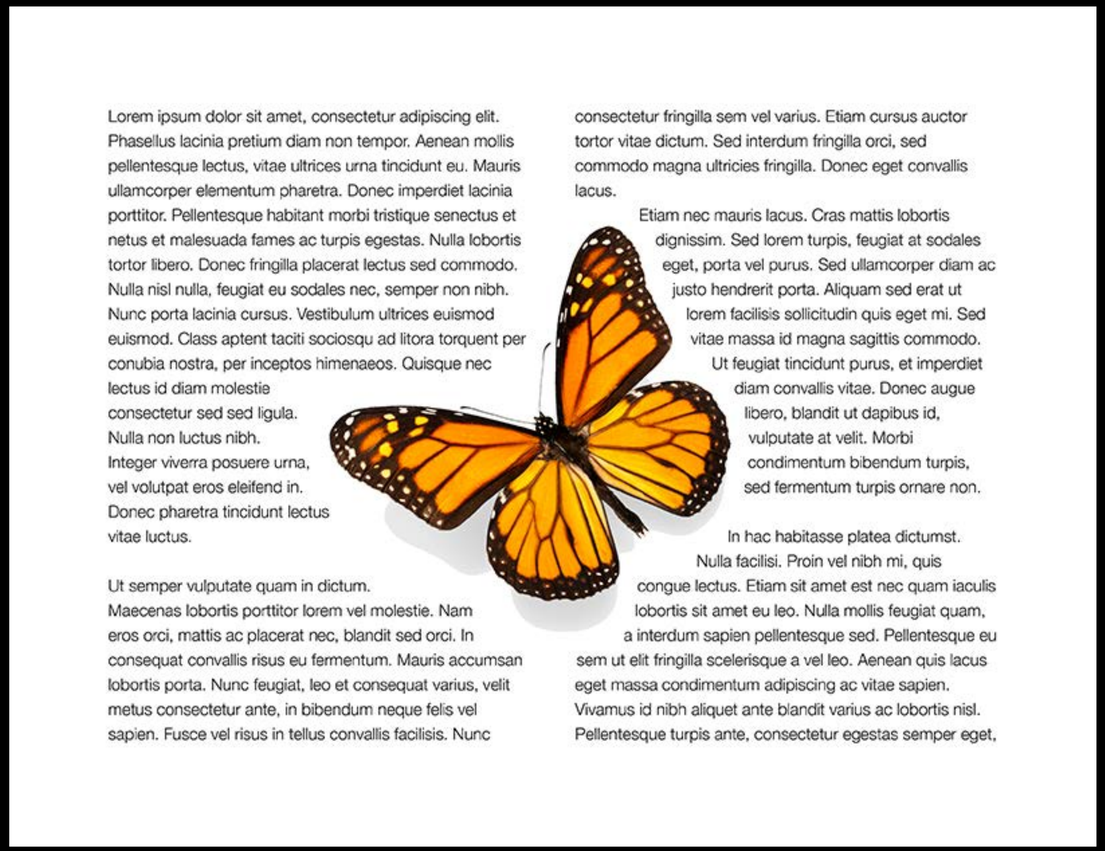
New functionality and parity with iOS

```
class NSTextContainer {
  ...
  var exclusionPaths: [NSBezierPath]
}
```

New APIs

New functionality and parity with iOS

```
class NSTextContainer {
  ...
  var exclusionPaths: [NSBezierPath]
}
```



New APIs

New functionality and parity with iOS

```
class NSTextContainer {
  ...
  var exclusionPaths: [NSBezierPath]
}
```

```
class NSTextField {
  ...
  var maximumNumberOfLines: Int
  var allowsDefaultTighteningForTruncation: Bool
}
```

New APIs

New functionality and parity with iOS

```
class NSTextContainer {
  ...
  var exclusionPaths: [NSBezierPath]
}
```

```
class NSTextField {
  ...
  var maximumNumberOfLines: Int
  var allowsDefaultTighteningForTruncation: Bool
}
```

And many more!

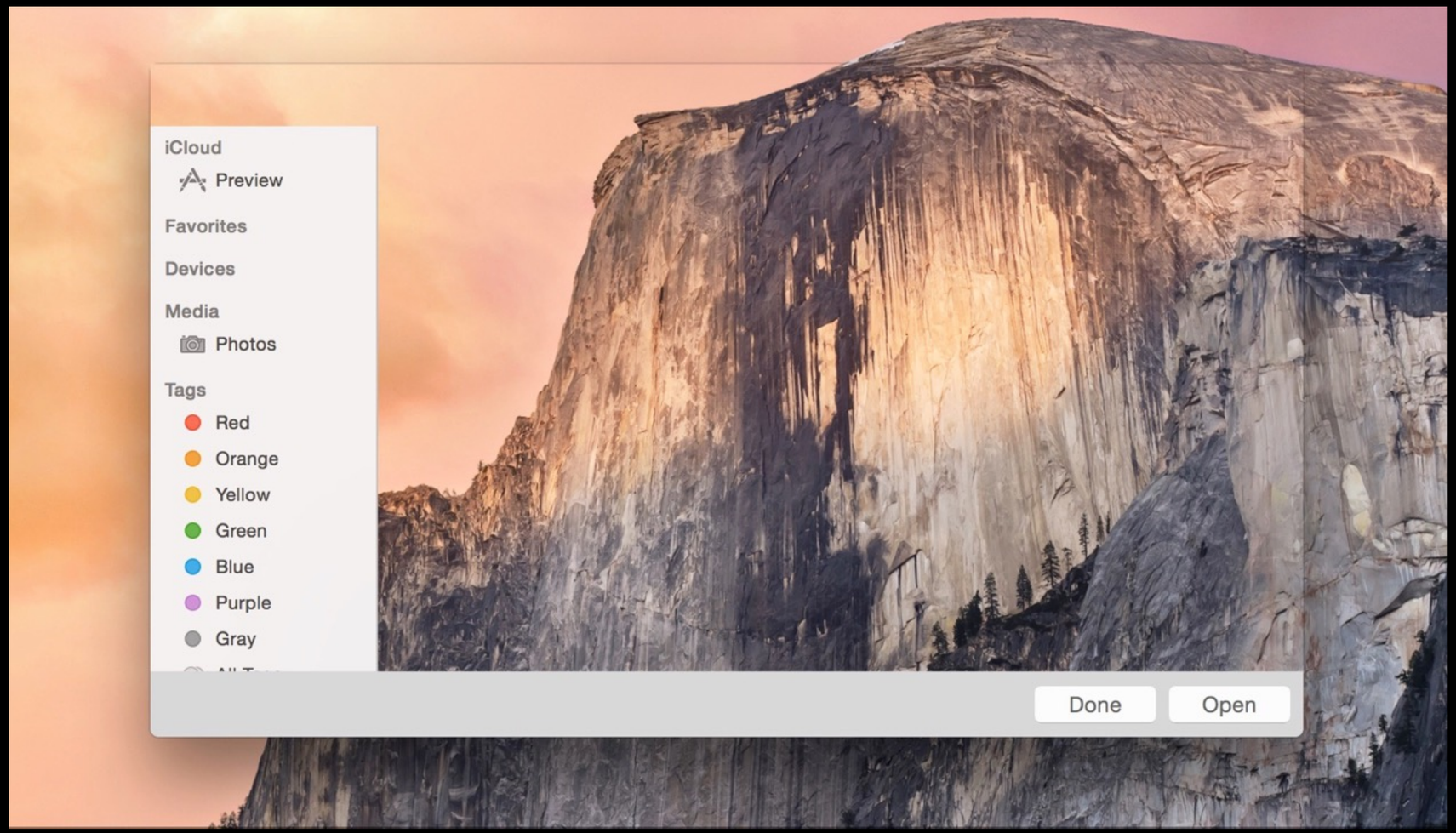
Visual Atomicity

Visual Atomicity

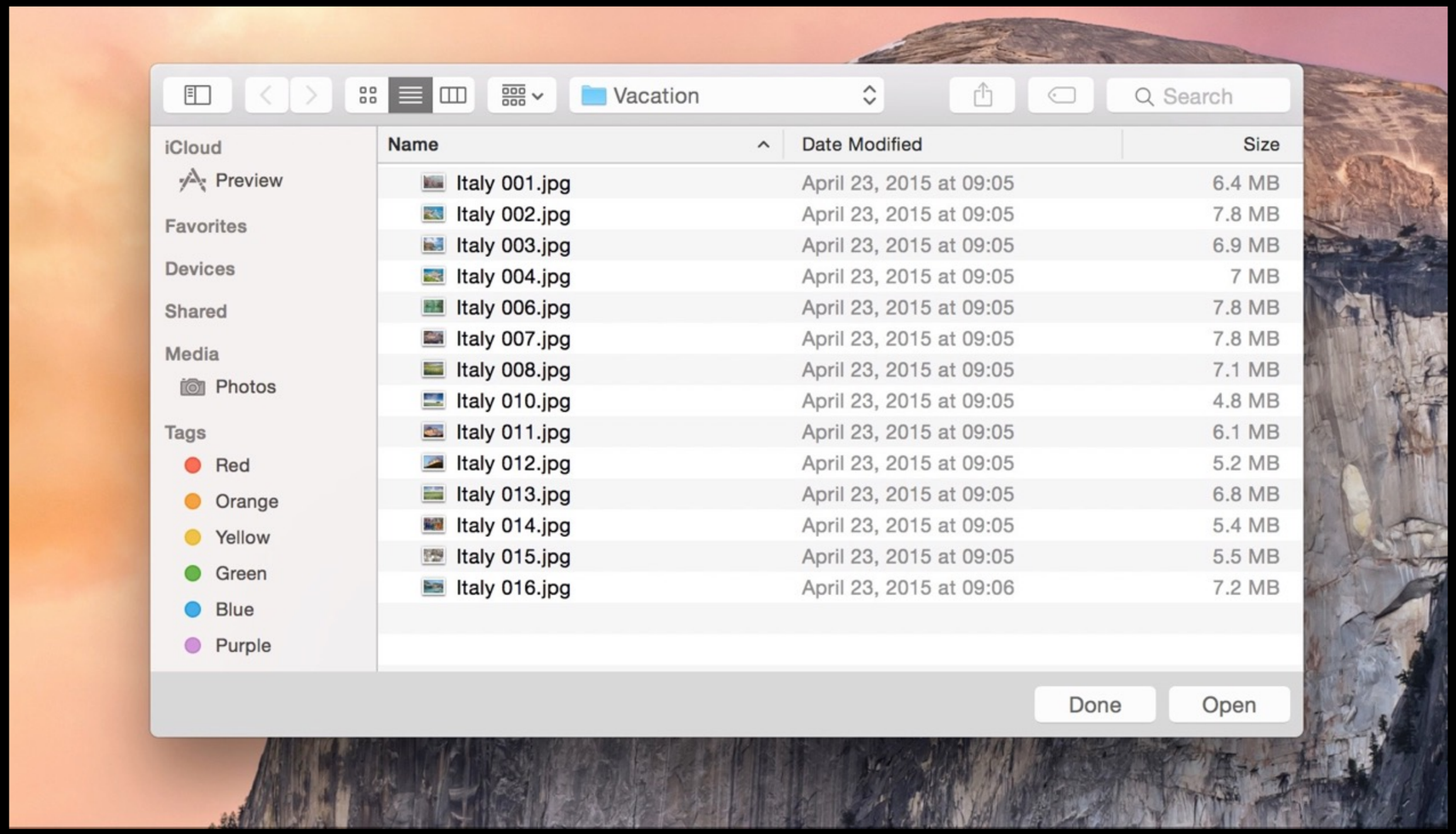
Visual Atomicity

Visual Atomicity

Visual Atomicity



Visual Atomicity



Visual Atomicity

Too many tools

Visual Atomicity

Too many tools

```
NSDisableScreenUpdates() / NSEnableScreenUpdates()
NSWindow.disableFlushWindow() / enableFlushWindow()
NSWindow.disableScreenUpdatesUntilFlush()
NSWindow.flushWindow()
NSWindow.displayIfNeeded() / display()
CATransaction.begin() / commit() / flush()
NSAnimationContext.beginGrouping() / endGrouping()
NSAnimationContext.runAnimationGroup(_:, completionHandler:)
```

Visual Atomicity

Now achieved with NSAnimationContext

```
NSDisableScreenUpdates() / NSEnableScreenUpdates()
NSWindow.disableFlushWindow() / enableFlushWindow()
NSWindow.disableScreenUpdatesUntilFlush()
NSWindow.flushWindow()
NSWindow.displayIfNeeded() / display()
CATransaction.begin() / commit() / flush()
NSAnimationContext.beginGrouping() / endGrouping()
NSAnimationContext.runAnimationGroup(_:, completionHandler:)
```

Visual Atomicity

Now achieved with NSAnimationContext

```
NSAnimationContext.beginGrouping() / endGrouping()
NSAnimationContext.runAnimationGroup(_:, completionHandler:)
```

Visual Atomicity

Now achieved with NSAnimationContext

```
NSAnimationContext.beginGrouping() / endGrouping()
NSAnimationContext.runAnimationGroup(_:, completionHandler:)

NSAnimationContext.beginGrouping()
window.setContentSize(newSize)
otherWindow.setFrameOrigin(newOtherOrigin)
view.frame = newViewFrame
...
NSAnimationContext.endGrouping()
```

Visual Atomicity

Now achieved with NSAnimationContext

```
NSAnimationContext.beginGrouping() / endGrouping()
NSAnimationContext.runAnimationGroup(_:, completionHandler:)

NSAnimationContext.beginGrouping()
window.setContentSize(newSize)
otherWindow.setFrameOrigin(newOtherOrigin)
view.frame = newViewFrame
...
NSAnimationContext.endGrouping()
```

Visual Atomicity

Now achieved with NSAnimationContext

```
NSAnimationContext.beginGrouping() / endGrouping()
NSAnimationContext.runAnimationGroup(_:, completionHandler:)

NSAnimationContext.beginGrouping()
window.setContentSize(newSize)
otherWindow.setFrameOrigin(newOtherOrigin)
view.frame = newViewFrame
...
NSAnimationContext.endGrouping()

NSDisableScreenUpdates() / NSEnableScreenUpdates()
NSWindow.disableScreenUpdatesUntilFlush()
```

Visual Atomicity

Now achieved with NSAnimationContext

```
NSAnimationContext.beginGrouping() / endGrouping()
NSAnimationContext.runAnimationGroup(_:, completionHandler:)

NSAnimationContext.beginGrouping()
window.setContentSize(newSize)
otherWindow.setFrameOrigin(newOtherOrigin)
view.frame = newViewFrame
...
NSAnimationContext.endGrouping()

NSDisableScreenUpdates() / NSEnableScreenUpdates()
NSWindow.disableScreenUpdatesUntilFlush()
```

Foundation

Foundation

NSUndoManager

NSCoder

NSError

NSProgress

NSNotificationCenter

NSPersonNameComponentsFormatter

Thermal state

NSUndoManager

Existing NSUndoManager API not a perfect fit in Swift

NSUndoManager

Existing NSUndoManager API not a perfect fit in Swift

```
- (void)registerUndoWithTarget:(id)target
                      selector:(SEL)selector
                        object:(id)anObject;

- (id)prepareWithInvocationTarget:(id)target;
```

Block-Based Undo

Block-Based Undo

```
- (void)registerUndoWithTarget:(id)target
                       handler:(void (^)(id target))undoHandler;
```

Block-Based Undo

```
- (void)registerUndoWithTarget:(id)target
                       handler:(void (^)(id target))undoHandler;

func registerUndoWithTarget<TargetType>(TargetType,
                                handler: TargetType -> ())
```

Block-Based Undo

```
- (void)registerUndoWithTarget:(id)target
                       handler:(void (^)(id target))undoHandler;

func registerUndoWithTarget<TargetType>(TargetType,
                                handler: TargetType -> ())
```

Separate argument for the target

Block-Based Undo

```
- (void)registerUndoWithTarget:(id)target
                       handler:(void (^)(id target))undoHandler;

func registerUndoWithTarget<TargetType>(TargetType,
                                handler: TargetType -> ())
```

Separate argument for the target

Use of generic type

Block-Based Undo

```
class ColorfulShape {
    var undoManager : NSUndoManager?

    var color = NSColor.blackColor() {
        didSet {
            undoManager?.registerUndoWithTarget(self) {target in
                target.color = oldValue
            }
        }
    }
}
```

Block-Based Undo

```
class ColorfulShape {
    var undoManager : NSUndoManager?

    var color = NSColor.blackColor() {
        didSet {
            undoManager?.registerUndoWithTarget(self) {target in
                target.color = oldValue
            }
        }
    }
}
```

Block-Based Undo

```
class ColorfulShape {
    var undoManager : NSUndoManager?

    var color = NSColor.blackColor() {
        didSet {
            undoManager?.registerUndoWithTarget(self) {target in
                target.color = oldValue
            }
        }
    }
}
```

Block-Based Undo

```
class ColorfulShape {
    var undoManager : NSUndoManager?

    var color = NSColor.blackColor() {
        didSet {
            undoManager?.registerUndoWithTarget(self) {target in
                target.color = oldValue
            }
        }
    }
}
```

Block-Based Undo

```
class ColorfulShape {
    var undoManager : NSUndoManager?

    var color = NSColor.blackColor() {
        didSet {
            undoManager?.registerUndoWithTarget(self) {target in
                target.color = oldValue
            }
        }
    }
}
```

Block-Based Undo

```
class ColorfulShape {
    var undoManager : NSUndoManager?

    var color = NSColor.blackColor() {
        didSet {
            undoManager?.registerUndoWithTarget(self) {target in
                target.color = oldValue
            }
        }
    }
}
```

```
func registerUndoWithTarget<TargetType>(TargetType,
                                        handler: TargetType -> ())
```

NSCoder Error Handling

Explicit handling of decoding errors

NSCoder Error Handling

Explicit handling of decoding errors

```
class NSCoder {
  func decodeObjectForKey(String) -> AnyObject?                  // 10.0, iOS 2
  func decodeObjectOfClasses(NSSet?,
                      forKey: String) -> AnyObject?              // 10.8, iOS 6
  ...
}
```

NSCoder Error Handling

Explicit handling of decoding errors

```
class NSCoder {
  func decodeObjectForKey(String) -> AnyObject?                  // 10.0, iOS 2
  func decodeObjectOfClasses(NSSet?,
                     forKey: String) -> AnyObject?               // 10.8, iOS 6
  ...
  func decodeTopLevelObjectForKey(String) throws -> AnyObject?
  func decodeTopLevelObjectOfClasses(NSSet?,
                                forKey: String) throws -> AnyObject?
  ...
}
```

NSCoder Error Handling

Explicit handling of decoding errors

```
class NSCoder {
  func decodeObjectForKey(String) -> AnyObject?                  // 10.0, iOS 2
  func decodeObjectOfClasses(NSSet?,
                     forKey: String) -> AnyObject?               // 10.8, iOS 6
  ...
  func decodeTopLevelObjectForKey(String) throws -> AnyObject?
  func decodeTopLevelObjectOfClasses(NSSet?,
                                forKey: String) throws -> AnyObject?
  ...
}
```

NSCoder Error Handling

Explicit handling of decoding errors

```
class NSCoder {
  func decodeObjectForKey(String) -> AnyObject?                  // 10.0, iOS 2
  func decodeObjectOfClasses(NSSet?,
                    forKey: String) -> AnyObject?                // 10.8, iOS 6
  ...
  func decodeTopLevelObjectForKey(String) throws -> AnyObject?
  func decodeTopLevelObjectOfClasses(NSSet?,
                              forKey: String) throws -> AnyObject?
  ...
}
```

NSError Value Provider

NSError Value Provider

```
var err = NSError(
    domain: MyGameErrorDomain,
    code: MyGameErrorLowFunds,
    userInfo: nil)
```

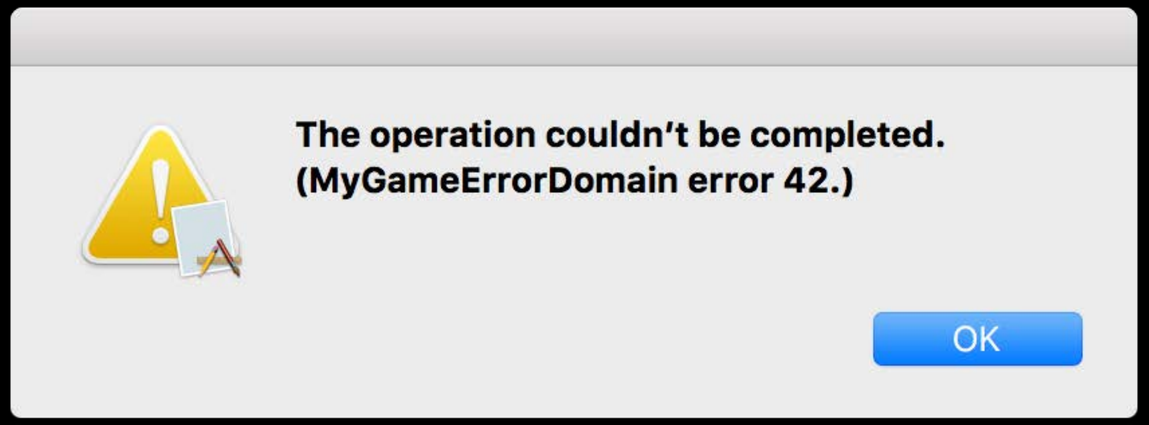
NSError Value Provider

```
var err = NSError(
    domain: MyGameErrorDomain,
    code: MyGameErrorLowFunds,
    userInfo: nil)
```

Simple, but not user-presentable

NSError Value Provider

```
var err = NSError(
    domain: MyGameErrorDomain,
    code: MyGameErrorLowFunds,
    userInfo: nil)
```

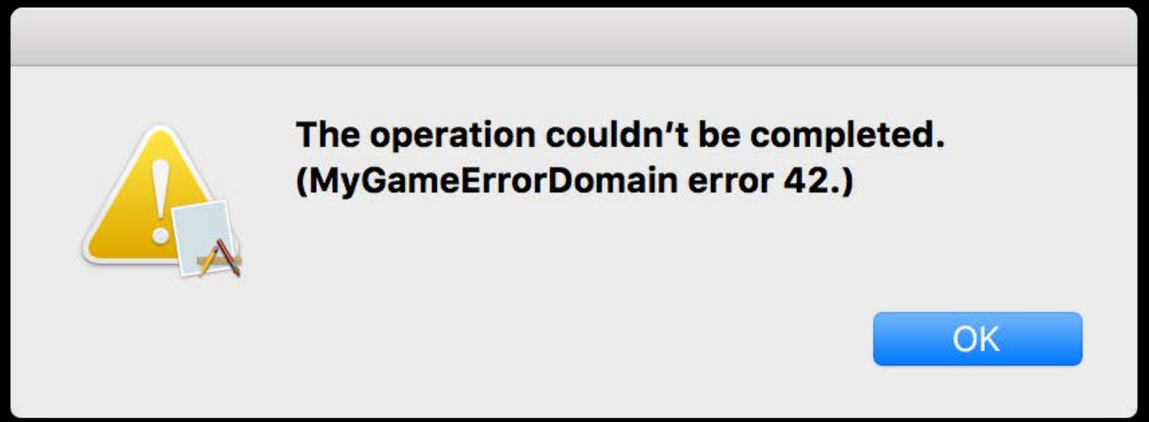


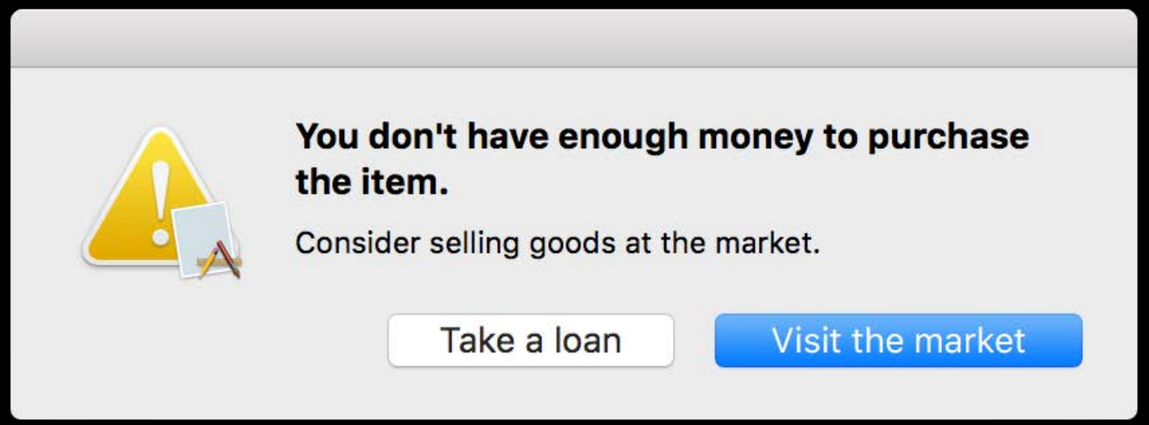
Simple, but not user-presentable

NSError Value Provider

```
var err = NSError(
    domain: MyGameErrorDomain,
    code: MyGameErrorLowFunds,
    userInfo: nil)
```

Simple, but not user-presentable





NSError Value Provider

```
var err = NSError(
    domain: MyGameErrorDomain,
    code: MyGameErrorLowFunds,
    userInfo: [
        NSLocalizedDescriptionKey: NSLocalizedString("You don't have enough
                                    money to purchase the item.", comment:"..."),
        NSLocalizedRecoverySuggestionErrorKey: NSLocalizedString("Consider
                                   selling goods at the market.", comment:"..."),
        NSLocalizedRecoveryOptionsErrorKey: [
                        NSLocalizedString("Visit the market", comment:"..."),
                        NSLocalizedString("Take a loan", comment:"...")],
        NSRecoveryAttempterErrorKey: self])
```

NSError Value Provider

Create and return desired values on demand

NSError Value Provider

Create and return desired values on demand

Register NSError domain-specific user info value provider

```
class func setUserInfoValueProviderForDomain(String,
                                provider: ((NSError, String) -> AnyObject)?)
```

NSError Value Provider

Create and return desired values on demand

Register NSError domain-specific user info value provider

```
class func setUserInfoValueProviderForDomain(String,
                                provider: ((NSError, String) -> AnyObject)?)
```

It will be invoked with any keys missing in the userInfo dictionary

NSError Value Provider

Create and return desired values on demand

Register NSError domain-specific user info value provider

```
class func setUserInfoValueProviderForDomain(String,
                                provider: ((NSError, String) -> AnyObject)?)
```

It will be invoked with any keys missing in the userInfo dictionary

```
var err = NSError(
    domain: MyGameErrorDomain,
    code: MyGameErrorLowFunds,
    userInfo: nil)
```

NSProgress

NSProgress

More explicit management of progress reporting

NSProgress

More explicit management of progress reporting

- API to add child progress objects directly

NSProgress

More explicit management of progress reporting

- API to add child progress objects directly
- Protocol for classes which can report progress

```
protocol NSProgressReporting {
    var progress: NSProgress { get }
}
```

NSProgress

More explicit management of progress reporting

- API to add child progress objects directly
- Protocol for classes which can report progress

```
protocol NSProgressReporting {
    var progress: NSProgress { get }
}
```

Ability to resume

NSProgress

More explicit management of progress reporting

- API to add child progress objects directly
- Protocol for classes which can report progress

```
protocol NSProgressReporting {
    var progress: NSProgress { get }
}
```

Ability to resume

Best Practices for Progress Reporting Pacific Heights Friday 1:30PM

NSNotificationCenter

NSNotificationCenter

Deallocated observers are automatically unregistered

NSNotificationCenter

Deallocated observers are automatically unregistered

```
let center = NSNotificationCenter.defaultCenter()
center.defaultCenter().addObserver(self,
                                   selector: "localeChanged:",
name: NSCurrentLocaleDidChangeNotification,
object: nil)
```

NSNotificationCenter

Deallocated observers are automatically unregistered

```
let center = NSNotificationCenter.defaultCenter()
center.defaultCenter().addObserver(self,
                                   selector: "localeChanged:",
                                   name: NSCurrentLocaleDidChangeNotification,
                                   object: nil)
```

No need to call

```
center.removeObserver(self,
                      name: NSCurrentLocaleDidChangeNotification,
                      object: nil)
```

NSPersonNameComponentsFormatter

NSPersonNameComponentsFormatter

Enables proper localized formatting of names

NSPersonNameComponentsFormatter

Enables proper localized formatting of names

Provides styles for different forms

NSPersonNameComponentsFormatter

```
let components = NSPersonNameComponents()
```

NSPersonNameComponentsFormatter

```
let components = NSPersonNameComponents()
components.givenName = "Grace"
components.middleName = "Murray"
components.familyName = "Hopper"
```

NSPersonNameComponentsFormatter

```
let components = NSPersonNameComponents()
components.givenName = "Grace"
components.middleName = "Murray"
components.familyName = "Hopper"

let formatter = NSPersonNameComponentsFormatter()
let result = formatter.stringFromPersonNameComponents(components)
```

NSPersonNameComponentsFormatter

```
let components = NSPersonNameComponents()
components.givenName = "Grace"
components.middleName = "Murray"
components.familyName = "Hopper"

let formatter = NSPersonNameComponentsFormatter()
let result = formatter.stringFromPersonNameComponents(components)

formatter.style = .Long              // “Grace Murray Hopper”
```

NSPersonNameComponentsFormatter

```
let components = NSPersonNameComponents()
components.givenName = "Grace"
components.middleName = "Murray"
components.familyName = "Hopper"

let formatter = NSPersonNameComponentsFormatter()
let result = formatter.stringFromPersonNameComponents(components)

formatter.style = .Long           // “Grace Murray Hopper”
formatter.style = .Default        // “Grace Hopper”
```

NSPersonNameComponentsFormatter

```
let components = NSPersonNameComponents()
components.givenName = "Grace"
components.middleName = "Murray"
components.familyName = "Hopper"

let formatter = NSPersonNameComponentsFormatter()
let result = formatter.stringFromPersonNameComponents(components)

formatter.style = .Long          // “Grace Murray Hopper”
formatter.style = .Default       // “Grace Hopper”
formatter.style = .Short         // “Grace”
```

NSPersonNameComponentsFormatter

```
let components = NSPersonNameComponents()
components.givenName = "Grace"
components.middleName = "Murray"
components.familyName = "Hopper"

let formatter = NSPersonNameComponentsFormatter()
let result = formatter.stringFromPersonNameComponents(components)

formatter.style = .Long          // “Grace Murray Hopper”
formatter.style = .Default       // “Grace Hopper”
formatter.style = .Short         // “Grace”
formatter.style = .Short         // “G Hopper” in Russian
```

NSString

Conditional quotation

Simpler localized case changing and searching

Transliteration

Adaptive (variable width) strings for UI presentation

NSString

Conditional quotation

Simpler localized case changing and searching

Transliteration

Adaptive (variable width) strings for UI presentation

What's New in Internationalization Pacific Heights Friday 9:00AM

Thermal State

API introduced in 10.10.3

Thermal State

API introduced in 10.10.3

Thermal State

API introduced in 10.10.3

```
class NSProcessInfo {
  var thermalState: NSProcessInfoThermalState { get }
}
```

Thermal State

API introduced in 10.10.3

```
class NSProcessInfo {
  var thermalState: NSProcessInfoThermalState { get }
}

enum NSProcessInfoThermalState : Int {
  case Nominal
  case Fair         // Fans may become audible
  case Serious      // Fans at maximum speed
  case Critical     // System needs to cool down
}
```

Thermal State

API introduced in 10.10.3

```
class NSProcessInfo {
  var thermalState: NSProcessInfoThermalState { get }
}

enum NSProcessInfoThermalState : Int {
  case Nominal
  case Fair         // Fans may become audible
  case Serious      // Fans at maximum speed
  case Critical     // System needs to cool down
}

let NSProcessInfoThermalStateDidChangeNotification: String
```

Quick Energy Efficient Coding Recap

Whenever possible, let the system manage activities

Specify tolerance on NSTimers

Wrap user-level operations with NSProcessInfo activity APIs

Schedule non-interactive tasks with NSBackgroundActivityScheduler

Perform background networking with NSURLSession

Set quality of service on NSOperations, NSOperationQueues, etc.

Core Data

Core Data

Unique constraints

Batch deletion

Many other API enhancements

Core Data

Unique constraints

Batch deletion

Many other API enhancements

What's New in Core Data	Mission	Thursday 2:30PM

More Stuff

NSBundle on-demand resources

NSDataAsset for arbitrary content in asset catalogs

NSUserActivity app search support

NSURL, NSURLComponents enhancements

NSFileManager unmount and eject functionality

“dictionary[key] = nil;” now does removeObjectForKey:

Accessibility APIs no longer raise

NSSearchField notifications and centered look

NSStringDrawingContext for more powerful string drawing

Summary

Summary

General API improvements

Summary

General API improvements

Many features and enhancements

- Force Touch
- Full screen split view
- Auto Layout
- NSCollectionView
- Text
- Foundation

More Information

Documentation and Videos

Cocoa and OS X Documentation, AppKit, and Foundation release notes

http://developer.apple.com/osx

Swift Language Documentation

http://developer.apple.com/swift

Technical Support

Apple Developer Forums

http://developer.apple.com/forums

General Inquiries

Paul Marcos, App Frameworks Evangelist

pmarcos@apple.com

Related Sessions

Adopting New Trackpad Features	Mission	Thursday 10:00AM
Mysteries of Auto Layout, Part 1	Presidio	Thursday 11:00AM
Mysteries of Auto Layout, Part 2	Presidio	Thursday 1:30PM
Improving the Full Screen Window Experience	Pacific Heights	Thursday 2:30PM
What's New in NSCollectionView	Mission	Thursday 4:30PM
What's New in Core Data	Mission	Thursday 2:30PM
Best Practices for Progress Reporting	Pacific Heights	Friday 1:30PM

Related Sessions

What's New in Swift	Presidio	Tuesday 11:00AM
Swift and Objective-C Interoperability	Mission	Tuesday 1:30PM
Advanced NSOperations	Presidio	Friday 9:00AM
What's New in Internationalization	Pacific Heights	Friday 9:00AM
Introducing the New System Fonts	Presidio	Friday 2:30PM

Labs

Interface Builder and Auto Layout Lab	Developer Tools Lab B	Tuesday 1:30–6:00PM
Cocoa Lab	Frameworks Lab B	Tuesday 2:30–6:00PM
Foundation Lab	Frameworks Lab A	Wednesday 9:00–10:40AM
Cocoa, Force Touch, and Gestures Lab	Frameworks Lab A	Thursday 11:00AM–1:10PM
Interface Builder and Auto Layout Lab	Developer Tools Lab C	Thursday 2:30–6:00PM
Cocoa and Full Screen Support Lab	Frameworks Lab D	Thursday 3:30–6:00PM
Cocoa and NSCollectionView Lab	Frameworks Lab B	Friday 9:00–10:40AM
Text and Fonts Lab	Frameworks Lab D	Friday 3:30–4:30PM


WWDC15