App Frameworks

#WWDC16

# What's New in Cocoa

Session 203

Ali Ozer Director, Cocoa Frameworks
Raleigh Ledet Event Wrangler, Cocoa Frameworks
Taylor Kelly Engineer, Cocoa Frameworks

# Agenda

API Updates

AppKit

Foundation

# API Updates

# API Updates

Swift API Guidelines

General API Refinements

# Swift API Guidelines

## Existing Cocoa guidelines remain valid

# Swift API Guidelines

## Existing Cocoa guidelines remain valid

Use clear and consistent naming

# Swift API Guidelines

## Existing Cocoa guidelines remain valid

Use clear and consistent naming

Strive for fluent usage

# Swift API Guidelines

## Existing Cocoa guidelines remain valid

Use clear and consistent naming

Strive for fluent usage

Name mutating and non-mutating method pairs consistently

# Swift API Guidelines

## Existing Cocoa guidelines remain valid

Use clear and consistent naming

Strive for fluent usage

Name mutating and non-mutating method pairs consistently

Avoid abbreviations

# Swift API Guidelines

## Existing Cocoa guidelines remain valid

Use clear and consistent naming

Strive for fluent usage

Name mutating and non-mutating method pairs consistently

Avoid abbreviations

---

API Design for Cocoa and Cocoa Touch — WWDC 2010

# Swift API Guidelines

## Some key changes

Eliminate repeated and needless words

# Swift API Guidelines

## Some key changes

### Eliminate repeated and needless words

```
contacts.arrayByAddingObject(person)
```

# Swift API Guidelines

## Some key changes

Eliminate repeated and needless words

```
contacts.arrayByAddingObject(person) → contacts.adding(person)
```

# Swift API Guidelines

## Some key changes

### Eliminate repeated and needless words

```
contacts.arrayByAddingObject(person) → contacts.adding(person)
NSColor.blueColor()
```

# Swift API Guidelines

## Some key changes

### Eliminate repeated and needless words

```
contacts.arrayByAddingObject(person) → contacts.adding(person)
NSColor.blueColor() → NSColor.blue()
```

# Swift API Guidelines

## Some key changes

Eliminate repeated and needless words

```
contacts.arrayByAddingObject(person) → contacts.adding(person)
NSColor.blueColor() → NSColor.blue()
```

But still strive for clarity

# Swift API Guidelines

## Some key changes

### Eliminate repeated and needless words

```
contacts.arrayByAddingObject(person) → contacts.adding(person)
NSColor.blueColor() → NSColor.blue()
```

### But still strive for clarity

```
NSColor.textColor()
```

# Swift API Guidelines

## Some key changes

Make use of types

# Swift API Guidelines

## Some key changes

### Make use of types

```
document.readFromURL(docURL, ofType: “rtf”)
```

# Swift API Guidelines

## Some key changes

### Make use of types

```
document.readFromURL(docURL, ofType: “rtf”) → document.read(from: docURL, ofType: ”rtf”)
```

# Swift API Guidelines

## Some key changes

### Make use of types

```
document.readFromURL(docURL, ofType: “rtf”) → document.read(from: docURL, ofType: ”rtf”)

func read(from: URL, ofType: String) throws
```

# Swift API Guidelines

## Some key changes

### Make use of types

```
document.readFromURL(docURL, ofType: “rtf”) → document.read(from: docURL, ofType: ”rtf”)

func read(from: URL, ofType: String) throws
func read(from: Data, ofType: String) throws
```

# Swift API Guidelines

## Some key changes

### Make use of types

```
document.readFromURL(docURL, ofType: “rtf”) → document.read(from: docURL, ofType: ”rtf”)

func read(from: URL, ofType: String) throws
func read(from: Data, ofType: String) throws
func read(from: FileWrapper, ofType: String) throws
```

# Swift API Guidelines

## Some key changes

### Make use of types

```
document.readFromURL(docURL, ofType: “rtf”) → document.read(from: docURL, ofType: ”rtf”)

func read(from: URL, ofType: String) throws
func read(from: Data, ofType: String) throws
func read(from: FileWrapper, ofType: String) throws
```

### Make use of label on first argument

# Swift API Guidelines

## Some key changes

### Make use of types

```
document.readFromURL(docURL, ofType: “rtf”) → document.read(from: docURL, ofType: ”rtf”)

func read(from: URL, ofType: String) throws
func read(from: Data, ofType: String) throws
func read(from: FileWrapper, ofType: String) throws
```

### Make use of label on first argument

### Compensate for weak type information

# Swift API Guidelines

## Some key changes

### Make use of types

```
document.readFromURL(docURL, ofType: “rtf”) → document.read(from: docURL, ofType: ”rtf”)

func read(from: URL, ofType: String) throws
func read(from: Data, ofType: String) throws
func read(from: FileWrapper, ofType: String) throws
```

### Make use of label on first argument

### Compensate for weak type information

# Swift API Guidelines

Many framework and standard library APIs have changed

# Swift API Guidelines

Many framework and standard library APIs have changed

Importer maps Objective-C APIs to Swift using the updated guidelines

- Some APIs require further tuning
  - Use NS_SWIFT_NAME

# Swift API Guidelines

Many framework and standard library APIs have changed

Importer maps Objective-C APIs to Swift using the updated guidelines

- Some APIs require further tuning
  - Use NS_SWIFT_NAME

Migrator converts existing code

# Swift API Guidelines

Many framework and standard library APIs have changed

Importer maps Objective-C APIs to Swift using the updated guidelines

- Some APIs require further tuning
  - Use NS_SWIFT_NAME

Migrator converts existing code

| Swift API Design Guidelines | Presidio | Tuesday 10:00AM |
| --- | --- | --- |

# General API Refinements

Nullability

Properties

Generics

Enumerations

Nested enumerations and options

String enumerations

@noescape

# Nullability

Bulk of this effort done last year

Continued refinements this year

# Nullability

Bulk of this effort done last year

Continued refinements this year

```
// 10.11
func addItem(withTitle: String,
             action: Selector?,
             keyEquivalent: String) -> NSMenuItem?
```

# Nullability

Bulk of this effort done last year

Continued refinements this year

```
// 10.11
func addItem(withTitle: String,
             action: Selector?,
             keyEquivalent: String) -> NSMenuItem?
```

```
// 10.12
func addItem(withTitle: String,
             action: Selector?,
             keyEquivalent: String) -> NSMenuItem
```

# Properties

## Class properties

# Properties

## Class properties

### NSWindow

```
// 10.12
class var allowsAutomaticWindowTabbing: Bool
```

# Properties

## Class properties

### NSWindow

```
// 10.12
class var allowsAutomaticWindowTabbing: Bool
@property (class) BOOL allowsAutomaticWindowTabbing;
```

# Properties

## Class properties

### NSWindow

```
// 10.12
class var allowsAutomaticWindowTabbing: Bool
@property (class) BOOL allowsAutomaticWindowTabbing;
```

### NSPersistentStoreCoordinator

```
// 10.11
class func registeredStoreTypes() -> [String : NSValue]
```

# Properties

## Class properties

### NSWindow

```
// 10.12
class var allowsAutomaticWindowTabbing: Bool
@property (class) BOOL allowsAutomaticWindowTabbing;
```

### NSPersistentStoreCoordinator

```
// 10.11
class func registeredStoreTypes() -> [String : NSValue]
```

```
// 10.12
class var registeredStoreTypes: [String : NSValue] { get }
```

# Properties

## Class properties

### NSWindow

```
// 10.12
class var allowsAutomaticWindowTabbing: Bool
@property (class) BOOL allowsAutomaticWindowTabbing;
```

### NSPersistentStoreCoordinator

```
// 10.11
class func registeredStoreTypes() -> [String : NSValue]
```

```
// 10.12
class var registeredStoreTypes: [String : NSValue] { get }
@property(class, readonly, strong) NSDictionary<NSString *, NSValue *> *registeredStoreTypes;
```

# Generics

## Not just for collections!

# Generics

## Not just for collections!

NSFetchRequest

# Generics

## Not just for collections!

NSFetchRequest

```
// 10.11
class NSFetchRequest : NSPersistentStoreRequest
```

# Generics

## Not just for collections!

### NSFetchRequest

```
// 10.11
class NSFetchRequest : NSPersistentStoreRequest
```

```
// 10.12
class NSFetchRequest<ResultType : NSFetchRequestResult> : NSPersistentStoreRequest
```

# Generics

## Not just for collections!

### NSFetchRequest

```
// 10.11
class NSFetchRequest : NSPersistentStoreRequest
```

```
// 10.12
class NSFetchRequest<ResultType : NSFetchRequestResult> : NSPersistentStoreRequest
   func execute() throws -> [ResultType]
```

# Generics

## Not just for collections!

### NSFetchRequest

```
// 10.11
class NSFetchRequest : NSPersistentStoreRequest
```

```
// 10.12
class NSFetchRequest<ResultType : NSFetchRequestResult> : NSPersistentStoreRequest
   func execute() throws -> [ResultType]
```

```
let request : NSFetchRequest<Employee> = Employee.fetchRequest()
```

# Generics

## Not just for collections!

### NSFetchRequest

```
// 10.11
class NSFetchRequest : NSPersistentStoreRequest
```

```
// 10.12
class NSFetchRequest<ResultType : NSFetchRequestResult> : NSPersistentStoreRequest
   func execute() throws -> [ResultType]
```

```
let request : NSFetchRequest<Employee> = Employee.fetchRequest()
if let employees = try? request.execute() { … }      // employees is [Employee]
```

# Generics

## Not just for collections!

### NSFetchRequest

```
// 10.11
class NSFetchRequest : NSPersistentStoreRequest
```

```
// 10.12
class NSFetchRequest<ResultType : NSFetchRequestResult> : NSPersistentStoreRequest
   func execute() throws -> [ResultType]
```

```
let request = Employee.fetchRequest()
if let employees = try? request.execute() { … }      // employees is [Employee]
```

# Enumerations

Modernized names for enumerations and option sets

# Enumerations

Modernized names for enumerations and option sets

```
// Swift 2
enum NSColorPanelMode : Int {
    case NSNoModeColorPanel
    case NSColorListModeColorPanel
    case NSWheelModeColorPanel
    case NSCrayonModeColorPanel
    …
}
```

# Enumerations

Modernized names for enumerations and option sets

```
// Swift 2
enum NSColorPanelMode : Int {
    case NSNoModeColorPanel
    case NSColorListModeColorPanel
    case NSWheelModeColorPanel
    case NSCrayonModeColorPanel
    …
}
```

```
NSColorPanel.sharedColorPanel().mode = .NSCrayonModeColorPanel
```

# Enumerations

Modernized names for enumerations and option sets

```
// Swift 3
enum NSColorPanelMode : Int {
    case none
    case colorList
    case wheel
    case crayon
    …
}
```

```
NSColorPanel.shared().mode = .crayon
```

# Enumerations

Modernized names for enumerations and option sets

```
// Swift 2
enum NSColorPanelMode : Int {
    case NSNoModeColorPanel
    case NSColorListModeColorPanel
    case NSWheelModeColorPanel
    case NSCrayonModeColorPanel
    …
}
```

```
NSColorPanel.sharedColorPanel().mode = .NSCrayonModeColorPanel
```

# Enumerations

Modernized names for enumerations and option sets

```
// Swift 3
enum NSColorPanelMode : Int {
    case none
    case colorList
    case wheel
    case crayon
    …
}
```

```
NSColorPanel.shared().mode = .crayon
```

# String Enumerations

Formalize string-valued enumerations in APIs

```
// Swift 2
```

```swift
// Swift 2
let NSStringTransformLatinToGreek: String
let NSStringTransformStripDiacritics: String
...
```

```swift
// Swift 2
let NSStringTransformLatinToGreek: String
let NSStringTransformStripDiacritics: String
...
func stringByApplyingTransform(transform: String, reverse: Bool) -> String?
```

```swift
// Swift 2
let NSStringTransformLatinToGreek: String
let NSStringTransformStripDiacritics: String
...
func stringByApplyingTransform(transform: String, reverse: Bool) -> String?

// Swift 3
struct StringTransform { ... }
```

```swift
// Swift 2
let NSStringTransformLatinToGreek: String
let NSStringTransformStripDiacritics: String
...
func stringByApplyingTransform(transform: String, reverse: Bool) -> String?

// Swift 3
struct StringTransform { ... }
extension StringTransform {
    static let latinToGreek: StringTransform
    static let stripDiacritics: StringTransform
    ...
}
```

```swift
// Swift 2
let NSStringTransformLatinToGreek: String
let NSStringTransformStripDiacritics: String
...
func stringByApplyingTransform(transform: String, reverse: Bool) -> String?

// Swift 3
struct StringTransform { ... }
extension StringTransform {
   static let latinToGreek: StringTransform
   static let stripDiacritics: StringTransform
   ...
}
func applyingTransform(_ transform: StringTransform, reverse: Bool) -> String?
```

```swift
// Swift 2
let NSStringTransformLatinToGreek: String
let NSStringTransformStripDiacritics: String
...
func stringByApplyingTransform(transform: String, reverse: Bool) -> String?

// Swift 3
struct StringTransform { ... }
extension StringTransform {
   static let latinToGreek: StringTransform
   static let stripDiacritics: StringTransform
   ...
}
func applyingTransform(_ transform: StringTransform, reverse: Bool) -> String?
```

```swift
// Swift 2
let NSStringTransformLatinToGreek: String
let NSStringTransformStripDiacritics: String
...
func stringByApplyingTransform(transform: String, reverse: Bool) -> String?

// Swift 3
struct StringTransform { ... }
extension StringTransform {
   static let latinToGreek: StringTransform
   static let stripDiacritics: StringTransform
   ...
}
func applyingTransform(_ transform: StringTransform, reverse: Bool) -> String?
```

```swift
// Swift 2
let NSStringTransformLatinToGreek: String
let NSStringTransformStripDiacritics: String
...
func stringByApplyingTransform(transform: String, reverse: Bool) -> String?

// Swift 3
struct StringTransform { ... }
extension StringTransform {
   static let latinToGreek: StringTransform
   static let stripDiacritics: StringTransform
   ...
}
func applyingTransform(_ transform: StringTransform, reverse: Bool) -> String?
```

```
// Adding your own
```

```
// Adding your own
extension StringTransform {
    static let publishing = StringTransform("Any-Publishing")
}
```

```swift
// Adding your own
extension StringTransform {
    static let publishing = StringTransform("Any-Publishing")
}

let fancyUserInput = userInput.applyingTransform(.publishing, reverse: false)
```

# String Enumerations

Use NS_EXTENSIBLE_STRING_ENUM  or NS_STRING_ENUM in Objective-C

# String Enumerations

Use NS_EXTENSIBLE_STRING_ENUM  or NS_STRING_ENUM in Objective-C

```
// 10.11
NSString * const NSStringTransformLatinToGreek;
NSString * const NSStringTransformStripDiacritics;
...
```

# String Enumerations

Use NS_EXTENSIBLE_STRING_ENUM  or NS_STRING_ENUM in Objective-C

```
// 10.11
NSString * const NSStringTransformLatinToGreek;
NSString * const NSStringTransformStripDiacritics;
…
```

```
// 10.12
typedef NSString *NSStringTransform NS_EXTENSIBLE_STRING_ENUM;
NSStringTransform const NSStringTransformLatinToGreek;
NSStringTransform const NSStringTransformStripDiacritics;
…
```

# Nested Enumerations, Options, Globals

Pull declarations related to a class into the class

# Nested Enumerations, Options, Globals

Pull declarations related to a class into the class

```
// Swift 2
struct NSDataWritingOptions : OptionSetType { … }
```

# Nested Enumerations, Options, Globals

Pull declarations related to a class into the class

```
// Swift 2
struct NSDataWritingOptions : OptionSetType { … }
class NSData … {
  func writeToURL(url: NSURL, options: NSDataWritingOptions) throws
  …
}
```

# Nested Enumerations, Options, Globals

Pull declarations related to a class into the class

```
// Swift 2
struct NSDataWritingOptions : OptionSetType { … }
class NSData … {
  func writeToURL(url: NSURL, options: NSDataWritingOptions) throws
  …
}
```

```
// Swift 3
class NSData … {
```

# Nested Enumerations, Options, Globals

Pull declarations related to a class into the class

```
// Swift 2
struct NSDataWritingOptions : OptionSetType { … }
class NSData … {
  func writeToURL(url: NSURL, options: NSDataWritingOptions) throws
  …
}
```

```
// Swift 3
class NSData … {
  struct WritingOptions : OptionSet { … }
```

# Nested Enumerations, Options, Globals

Pull declarations related to a class into the class

```
// Swift 2
struct NSDataWritingOptions : OptionSetType { … }
class NSData … {
  func writeToURL(url: NSURL, options: NSDataWritingOptions) throws
  …
}
```

```
// Swift 3
class NSData … {
  struct WritingOptions : OptionSet { … }
  func write(to: URL, options: WritingOptions = []) throws
}
```

# Nested Enumerations, Options, Globals

Pull declarations related to a class into the class

```
// Swift 2
struct NSDataWritingOptions : OptionSetType { … }
class NSData … {
  func writeToURL(url: NSURL, options: NSDataWritingOptions) throws
  …
}
```

```
// Swift 3
class NSData … {
  struct WritingOptions : OptionSet { … }
  func write(to: URL, options: WritingOptions = []) throws
}
```

# Nested Enumerations, Options, Globals

Pull declarations related to a class into the class

```
// Swift 2
struct NSDataWritingOptions : OptionSetType { … }
class NSData … {
  func writeToURL(url: NSURL, options: NSDataWritingOptions) throws
  …
}
```

```
// Swift 3
class NSData … {
  struct WritingOptions : OptionSet { … }
  func write(to: URL, options: WritingOptions = []) throws
}
```

# Nested Enumerations, Options, Globals

Pull declarations related to a class into the class

```
// Swift 2
struct NSDataWritingOptions : OptionSetType { … }
class NSData … {
  func writeToURL(url: NSURL, options: NSDataWritingOptions) throws
  …
}
```

```
// Swift 3
class NSData … {
  struct WritingOptions : OptionSet { … }
  func write(to: URL, options: WritingOptions = []) throws
}
```

# Nested Enumerations, Options, Globals

Pull declarations related to a class into the class

```
// Swift 2
struct NSDataWritingOptions : OptionSetType { … }
class NSData … {
  func writeToURL(url: NSURL, options: NSDataWritingOptions) throws
  …
}
```

```
// Swift 3
class NSData … {
  struct WritingOptions : OptionSet { … }
  func write(to: URL, options: WritingOptions = []) throws
}
```

# @noescape

# @noescape

Indicates that a closure's execution won't escape the function call

# @noescape

Indicates that a closure's execution won't escape the function call

```
func performBatchUpdates(_ updates: (@noescape () -> Void)?,
                         completionHandler: ((Bool) -> Void)? = nil)
```

# @noescape

Indicates that a closure's execution won't escape the function call

```
func performBatchUpdates(_ updates: (@noescape () -> Void)?,
                         completionHandler: ((Bool) -> Void)? = nil)
```

# @noescape

Indicates that a closure's execution won't escape the function call

```
func performBatchUpdates(_ updates: (@noescape () -> Void)?,
                         completionHandler: ((Bool) -> Void)? = nil)
```

# @noescape

Indicates that a closure's execution won't escape the function call

```
func performBatchUpdates(_ updates: (@noescape () -> Void)?,
                         completionHandler: ((Bool) -> Void)? = nil)
```

```
- (void)performBatchUpdates:(void (NS_NOESCAPE ^__nullable)(void))updates
        completionHandler:(void (^__nullable)(BOOL finished))completionHandler;
```

# AppKit

# AppKit

Window snapping and tabbing

Right-to-left

Promise drags

Container views

Grid view and auto layout

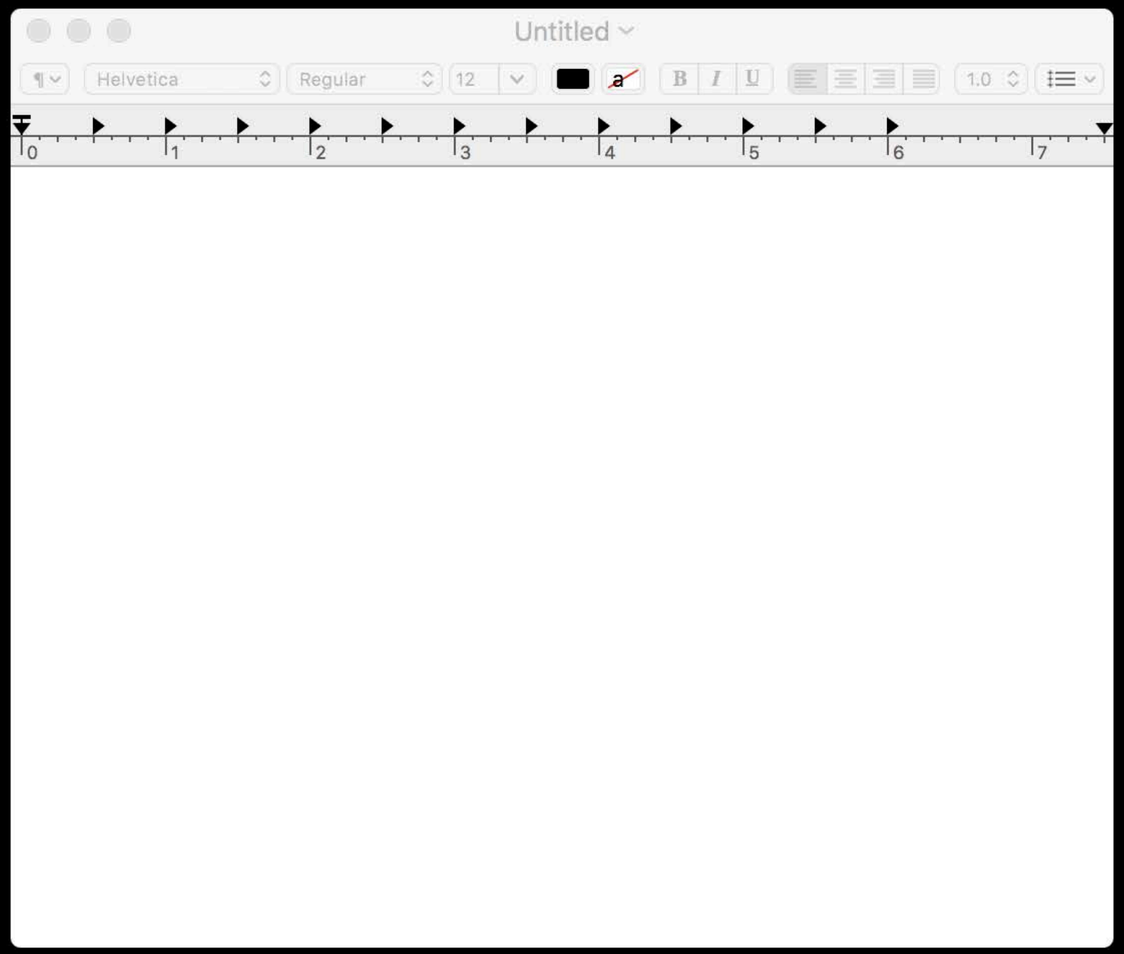
Wide color

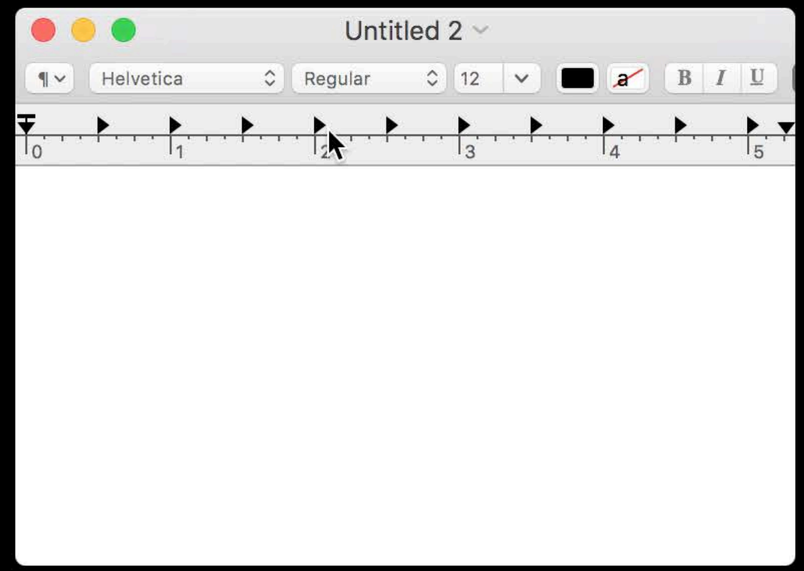
Status items

Control constructors

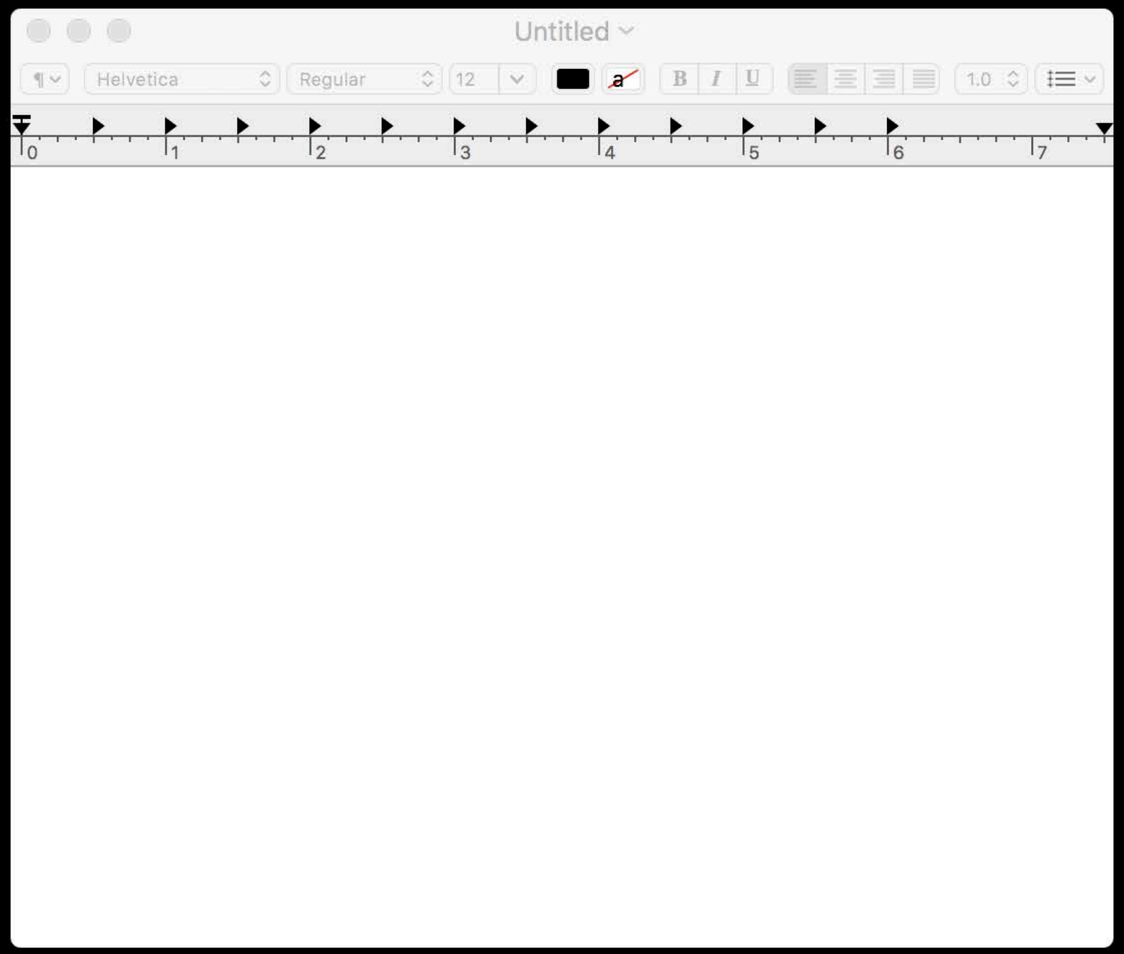
API refinements

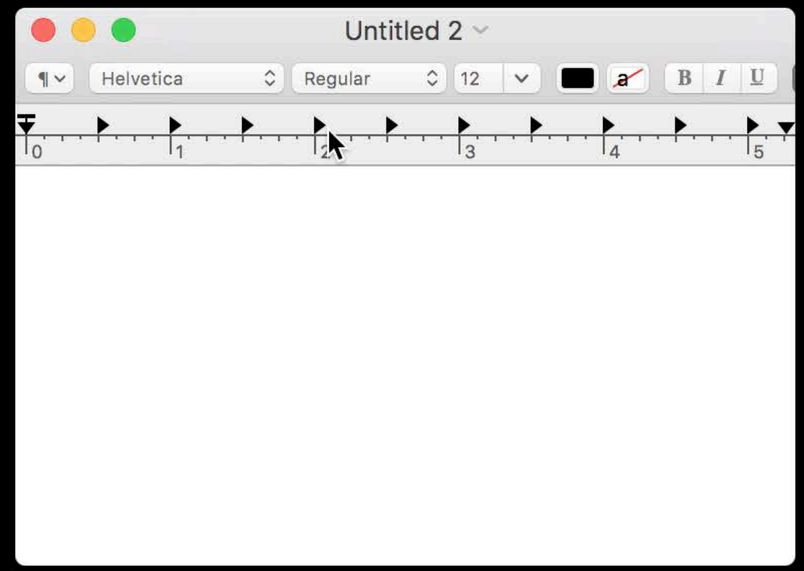
# Window Snapping



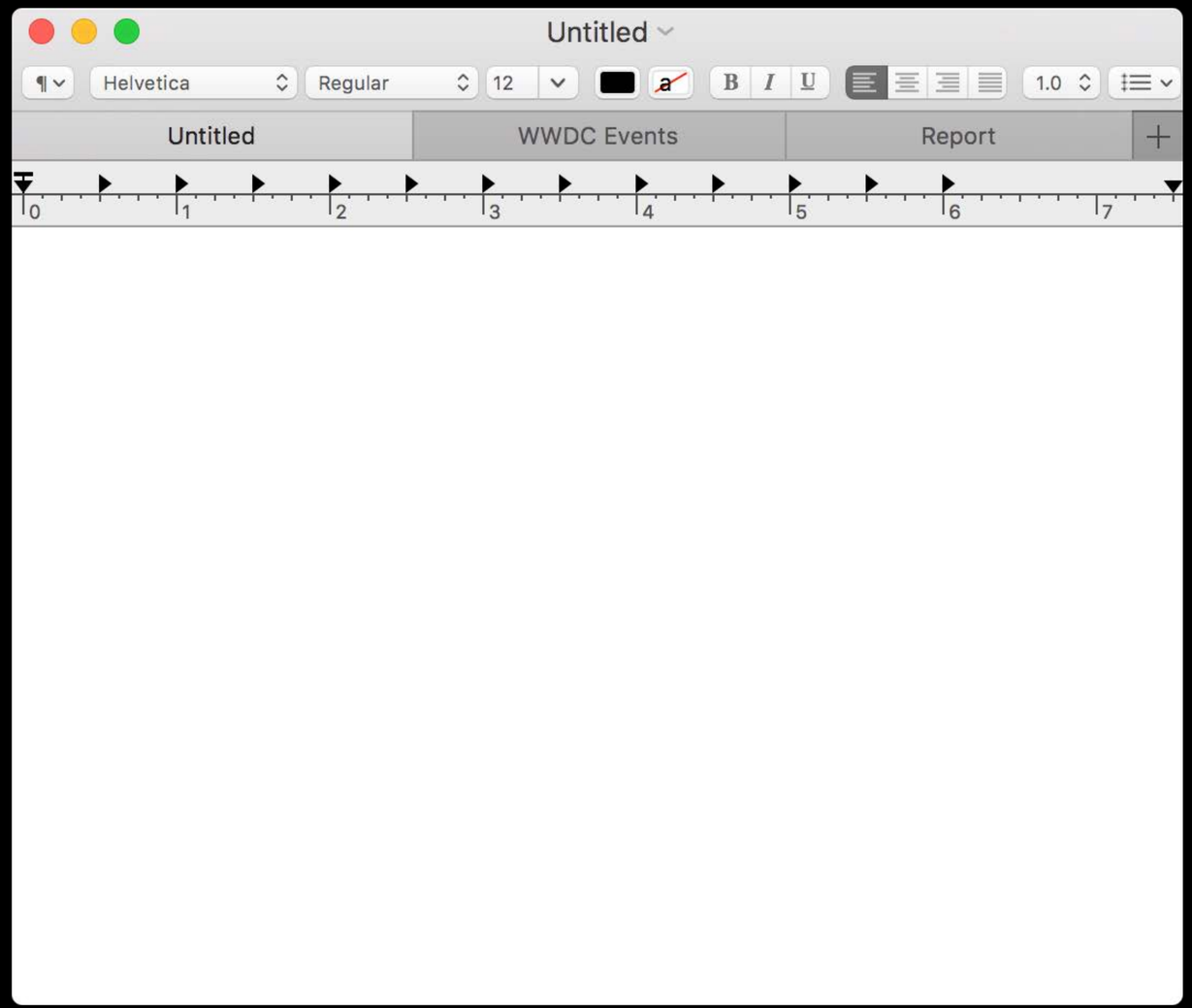



# Window Snapping



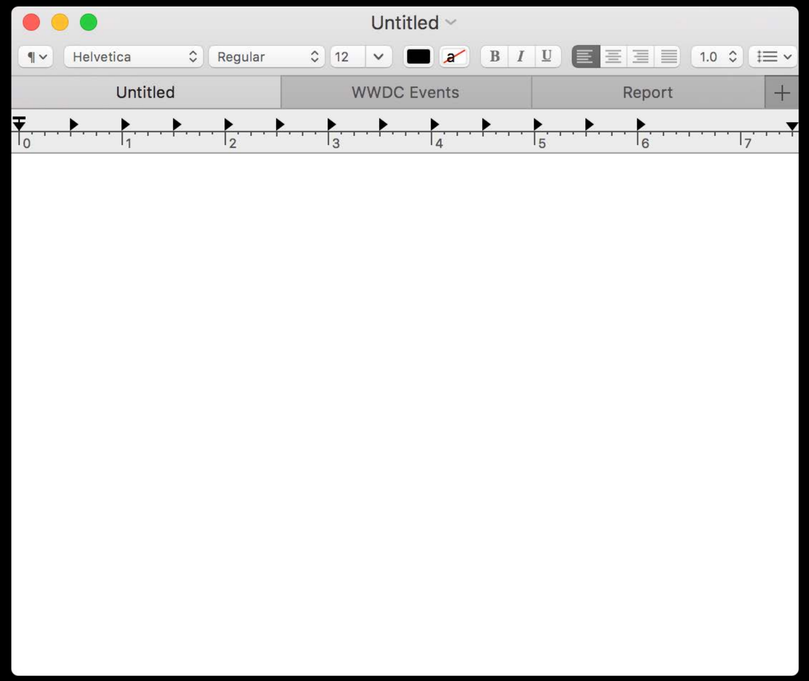



# Window Tabs


Untitled
Helvetica
Regular
12
1.0
Untitled
WWDC Events
Report
0
1
2
3
4
5
6
7

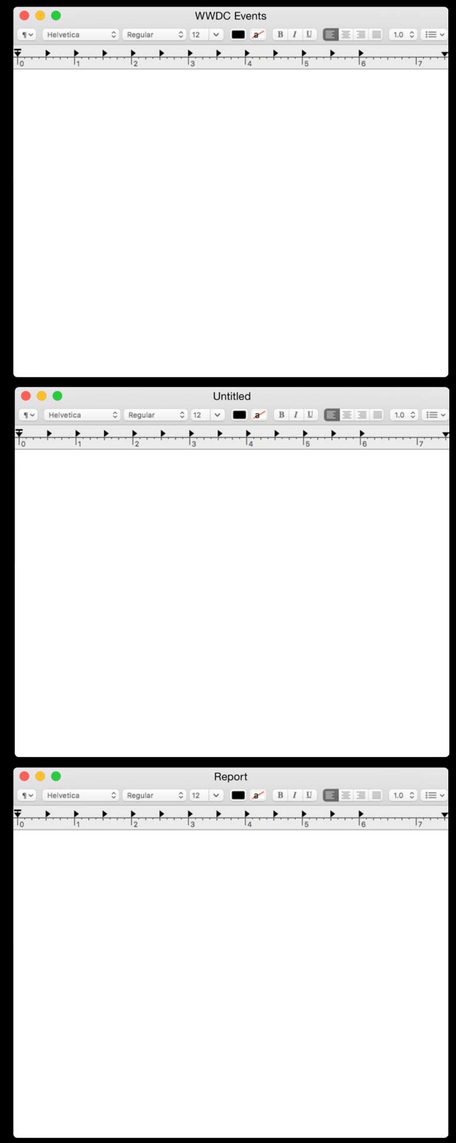
# Window Tabs

Just windows



# Window Tabs

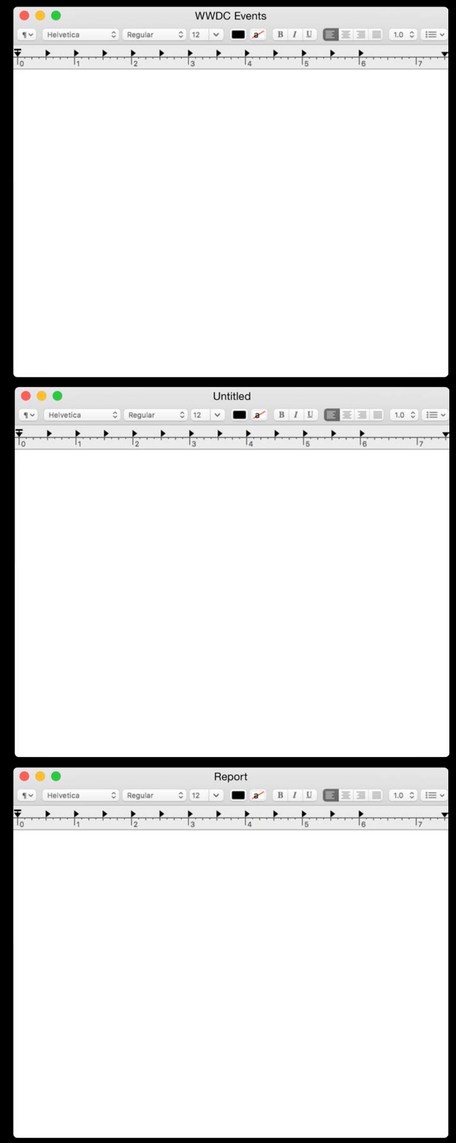
Just windows



# Window Tabs

Just windows

Every window is visible

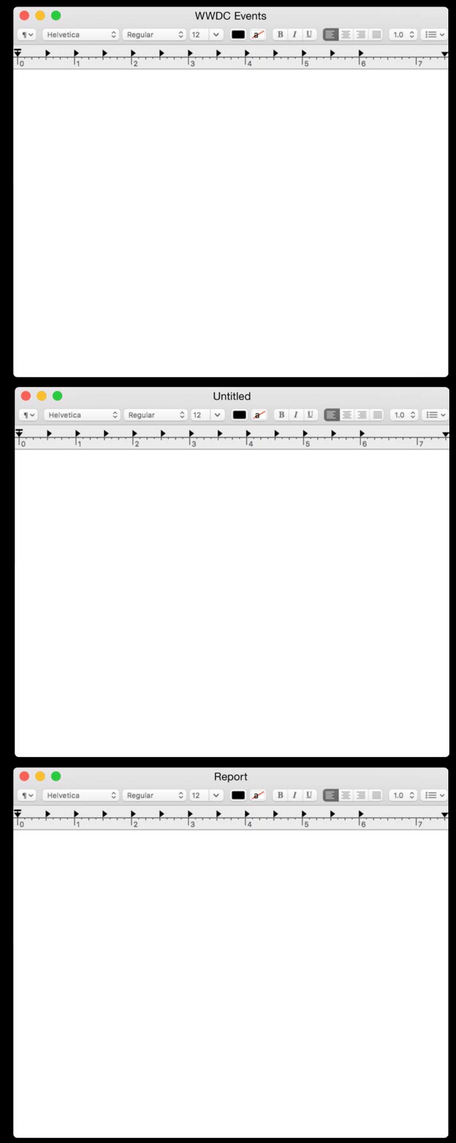


# Window Tabs

Just windows

Every window is visible

Hidden at system level

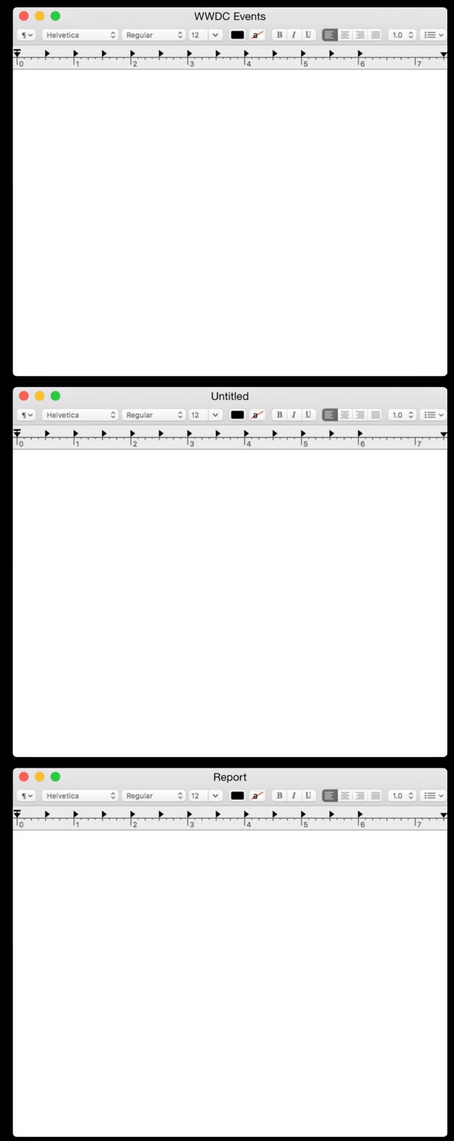


# Window Tabs

Just windows

Every window is visible

Hidden at system level

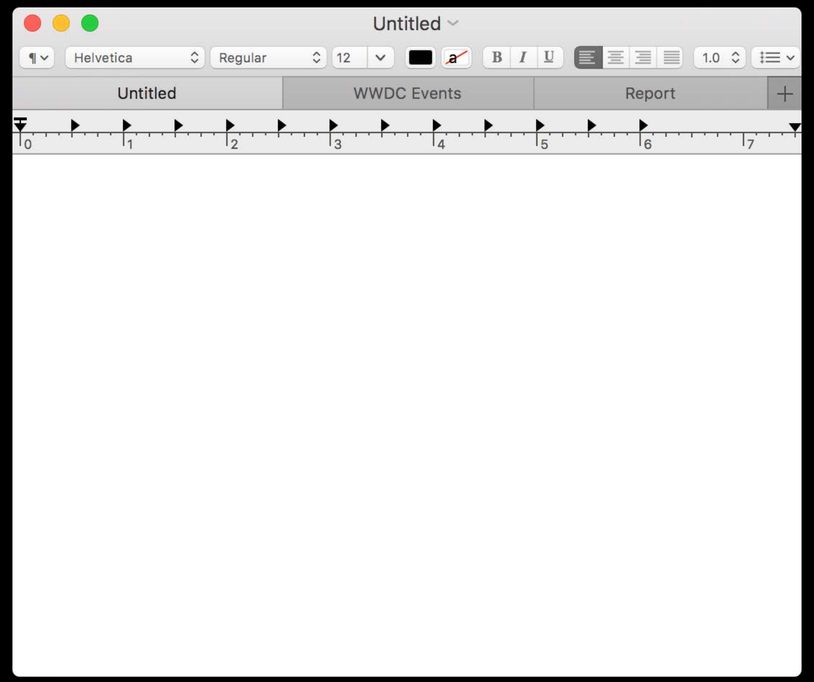


# Window Tabs

Just windows

Every window is visible

Hidden at system level



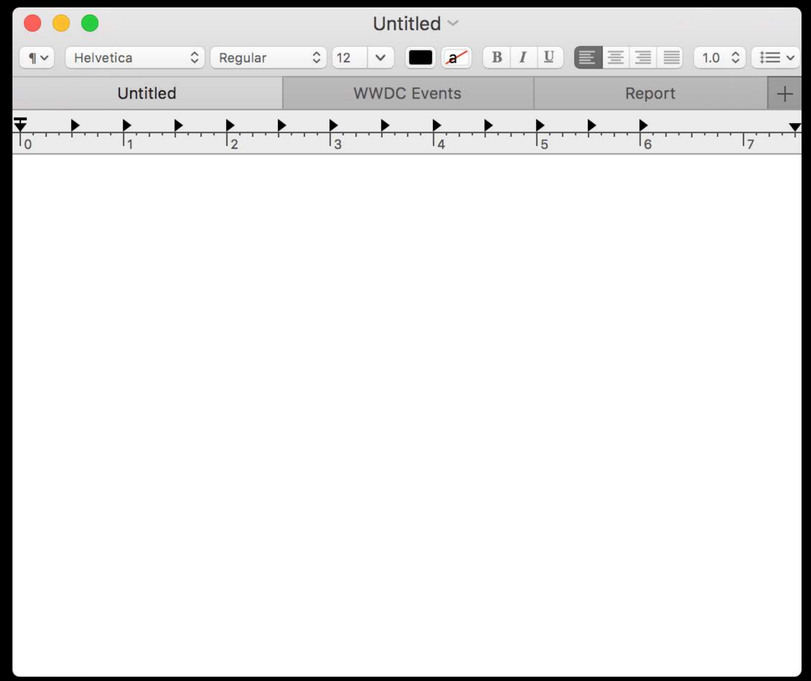
# Window Tabs

Just windows

Every window is visible

Hidden at system level

Automatic



# Window Tabs

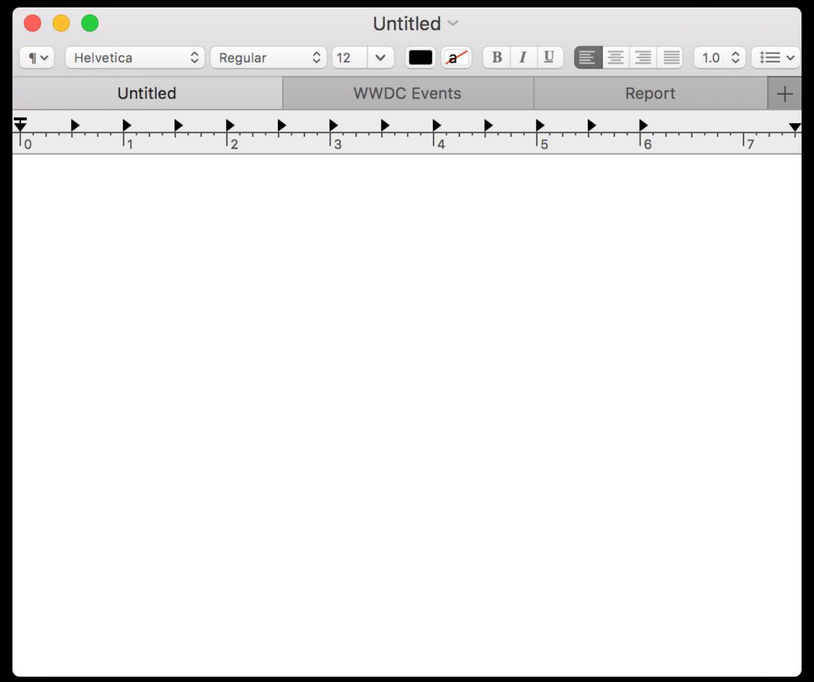
Just windows

Every window is visible

Hidden at system level

Automatic

Create tab: orderFront()



# Window Tabs

Just windows

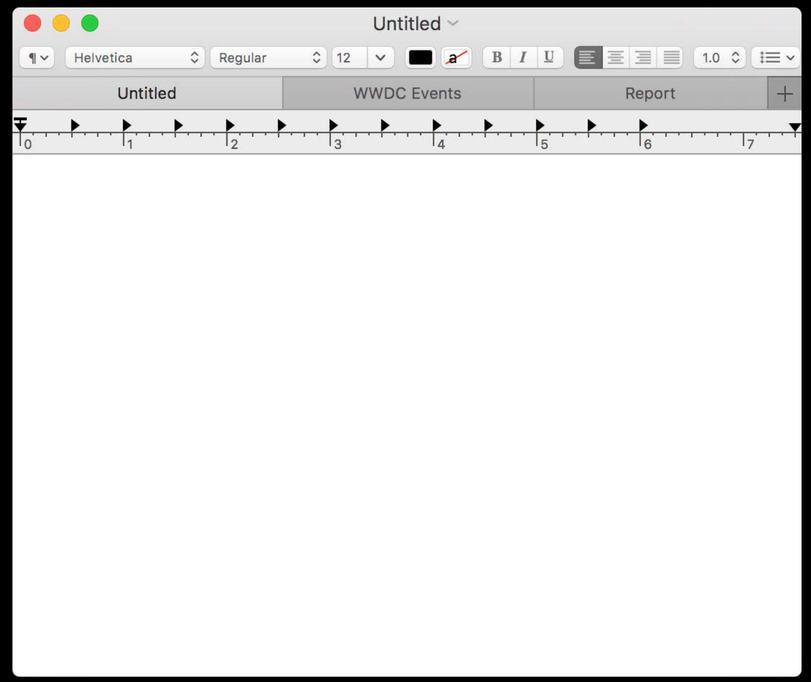
Every window is visible

Hidden at system level

Automatic

Create tab: orderFront()

Remove tab: orderOut()



# Window Tabs

Just windows

Every window is visible

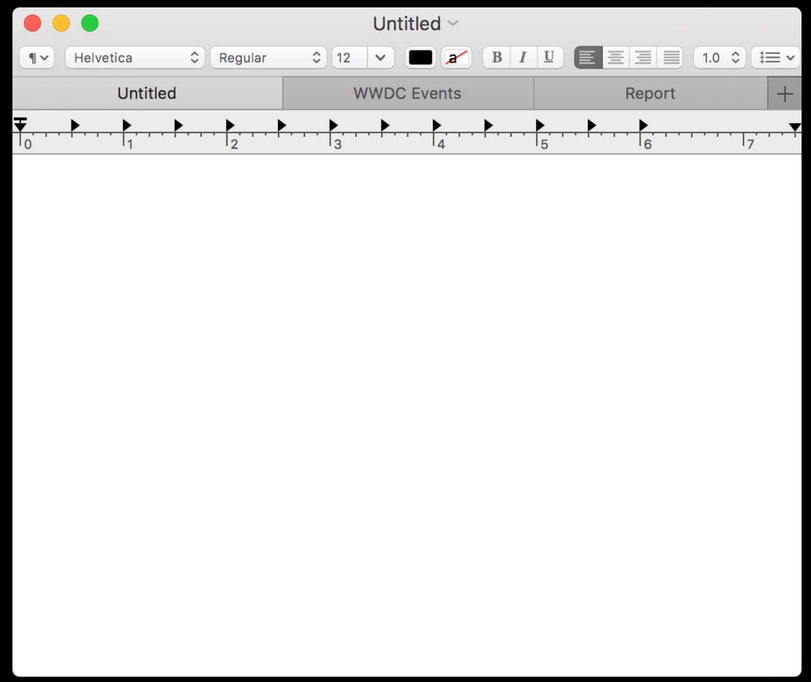
Hidden at system level

Automatic

Create tab: orderFront()

Remove tab: orderOut()

Window resized on tab switch



# Window Tabs

NSDocument based app

Non-NSDocument based app

Existing tab implementation

Customization

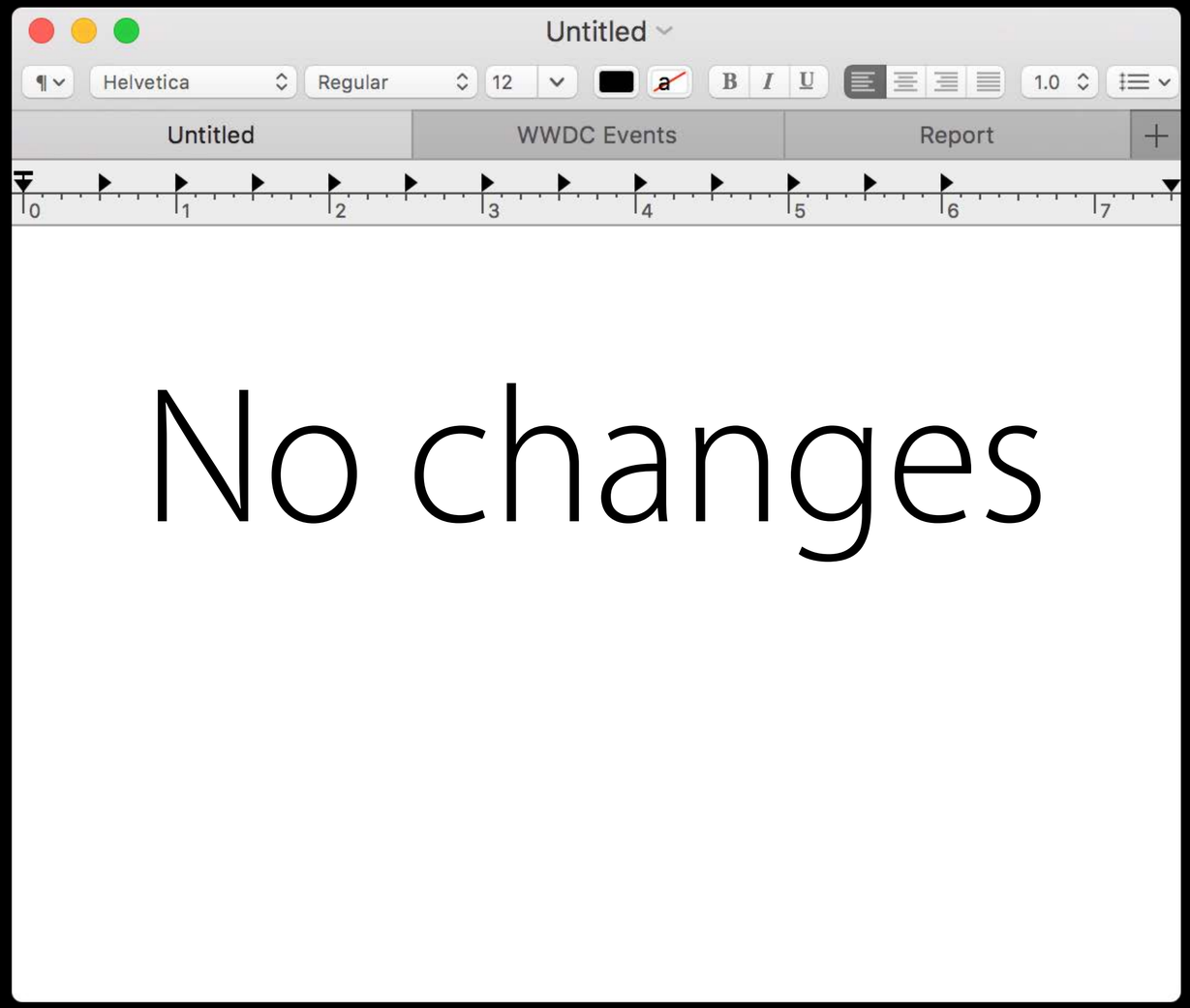
# Window Tabs

## NSDocument based app

Fully automatic

# Window Tabs

## NSDocument based app



# Window Tabs

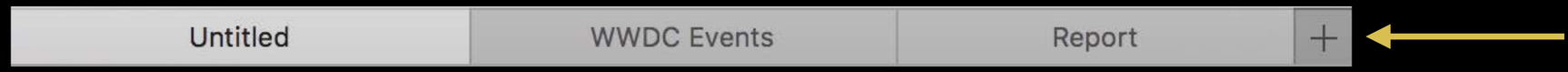
## Non-NSDocument based app

Enable the “New tab” button

# Window Tabs

## Non-NSDocument based app

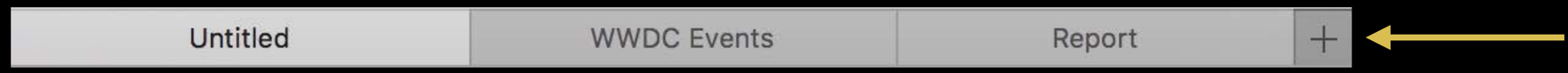
Enable the “New tab” button



# Window Tabs

## Non-NSDocument based app

Enable the “New tab” button




```
extension NSResponder
@IBAction public func newWindowForTab(_ sender: AnyObject?)
```

# Window Tabs

## Existing tab implementation

Disable automatic window tabbing

# Window Tabs

## Existing tab implementation

Disable automatic window tabbing

```
extension NSWindow
public class var allowsAutomaticWindowTabbing: Bool
```

# Window Tabs

## Customization

```
extension NSWindow

public class var userTabbingPreference: NSWindowUserTabbingPreference { get }
public var tabbingMode: NSWindowTabbingMode
public var tabbingIdentifier: String
public var tabbedWindows: [NSWindow]? { get }
public func addTabbedWindow(window: NSWindow, ordered: NSWindowOrderingMode)
```

# Window Tabs

## Customization

```
extension NSWindow

public class var userTabbingPreference: NSWindowUserTabbingPreference { get }
public var tabbingMode: NSWindowTabbingMode
public var tabbingIdentifier: String
public var tabbedWindows: [NSWindow]? { get }
public func addTabbedWindow(window: NSWindow, ordered: NSWindowOrderingMode)
```

```
.manual
.always
.fullScreen
```

# Window Tabs

## Customization

```
extension NSWindow

public class var userTabbingPreference: NSWindowUserTabbingPreference { get }
public var tabbingMode: NSWindowTabbingMode
public var tabbingIdentifier: String
public var tabbedWindows: [NSWindow]? { get }
public func addTabbedWindow(window: NSWindow, ordered: NSWindowOrderingMode)
```

# Window Tabs

## Customization

```
extension NSWindow

public class var userTabbingPreference: NSWindowUserTabbingPreference { get }
public var tabbingMode: NSWindowTabbingMode
public var tabbingIdentifier: String
public var tabbedWindows: [NSWindow]? { get }
public func addTabbedWindow(window: NSWindow, ordered: NSWindowOrderingMode)
```

```
.automatic
.preferred
.disallowed
```

# Window Tabs

## Customization

```
extension NSWindow

public class var userTabbingPreference: NSWindowUserTabbingPreference { get }
public var tabbingMode: NSWindowTabbingMode
public var tabbingIdentifier: String
public var tabbedWindows: [NSWindow]? { get }
public func addTabbedWindow(window: NSWindow, ordered: NSWindowOrderingMode)
```

# Window Tabs

## Customization

```
extension NSWindow

public class var userTabbingPreference: NSWindowUserTabbingPreference { get }
public var tabbingMode: NSWindowTabbingMode
public var tabbingIdentifier: String
public var tabbedWindows: [NSWindow]? { get }
public func addTabbedWindow(window: NSWindow, ordered: NSWindowOrderingMode)
```

# Window Tabs

## Customization

```
extension NSWindow

public class var userTabbingPreference: NSWindowUserTabbingPreference { get }
public var tabbingMode: NSWindowTabbingMode
public var tabbingIdentifier: String
public var tabbedWindows: [NSWindow]? { get }
public func addTabbedWindow(window: NSWindow, ordered: NSWindowOrderingMode)
```

# Window Tabs

## Customization

```
extension NSWindow

@IBAction public func selectNextTab(_ sender: AnyObject?)
@IBAction public func selectPreviousTab(_ sender: AnyObject?)
@IBAction public func moveTabToNewWindow(_ sender: AnyObject?)
@IBAction public func mergeAllWindows(_ sender: AnyObject?)
@IBAction public func toggleTabBar(_ sender: AnyObject?)
```

# AppKit

Window snapping and tabbing

Right-to-left

Promise drags

Container views

Grid view and auto layout

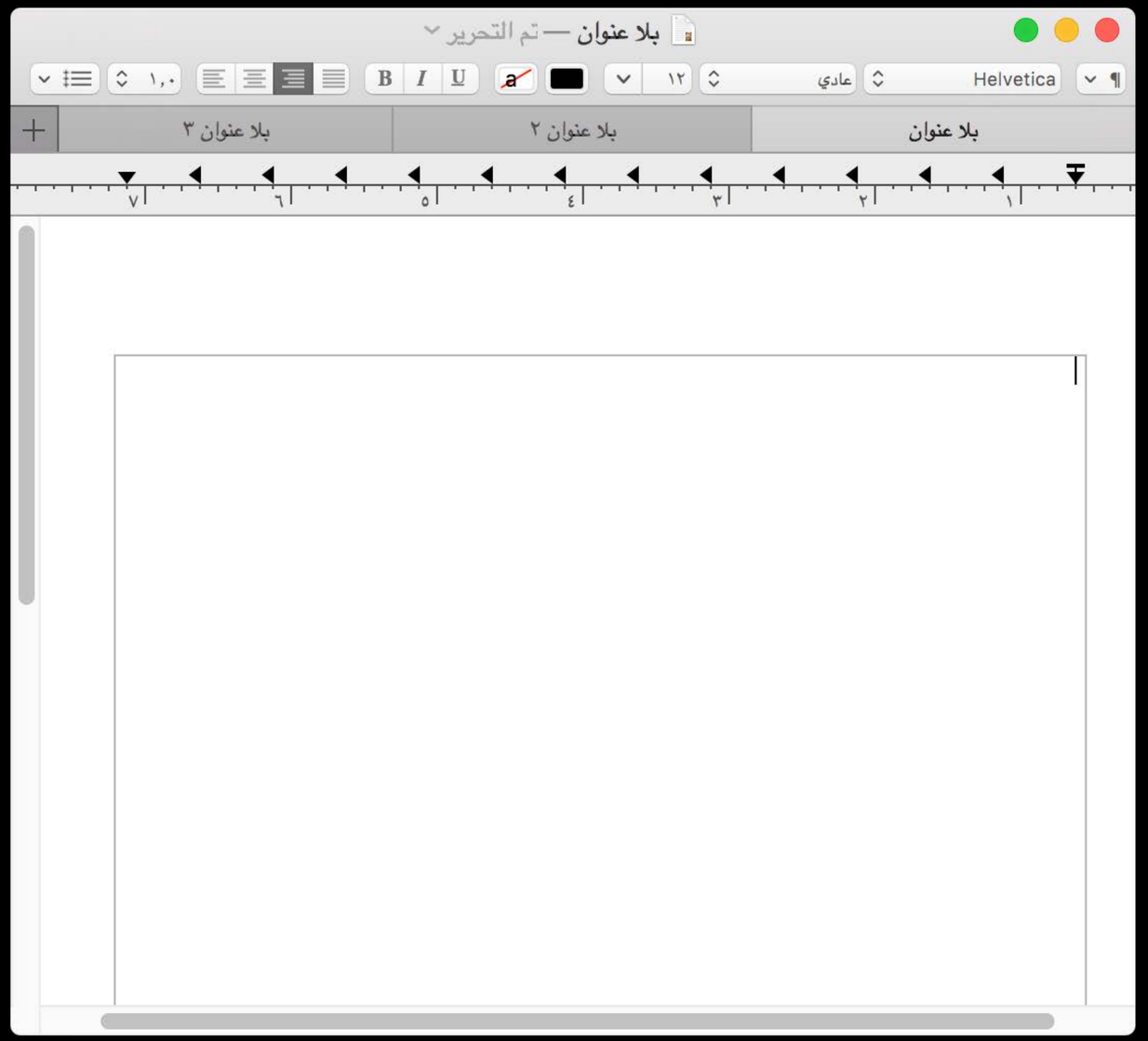
Wide color

Status items

Control constructors

API refinements

# Right-to-Left Support



# Right-to-Left Support

System level

Application level

Content level

# Right-to-Left Support

System level

Application level

Content level

# Right-to-Left Support

System level

Application level

Content level

Development tip

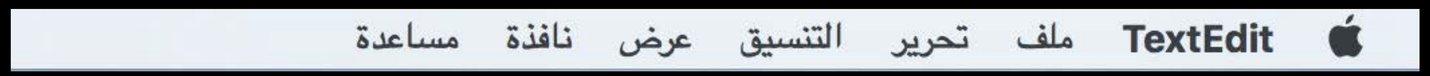
# Right-to-Left Support

## System level

# Right-to-Left Support

## System level

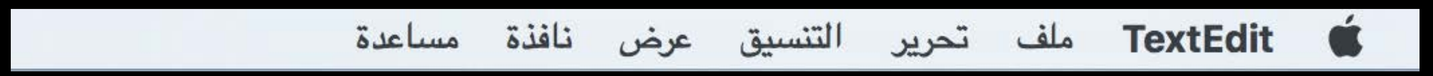
Menu bar



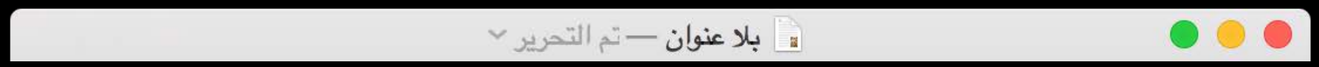
# Right-to-Left Support

## System level

Menu bar




Window title bar



# Right-to-Left Support

## Application level

NSScrollView

- Vertical Scroller
- Vertical Ruler

NSBrowser

# Right-to-Left Support

## Content level

Auto Layout

NSTableView

NSOutlineView

NSCollectionView

NSStackView

NSSplitView

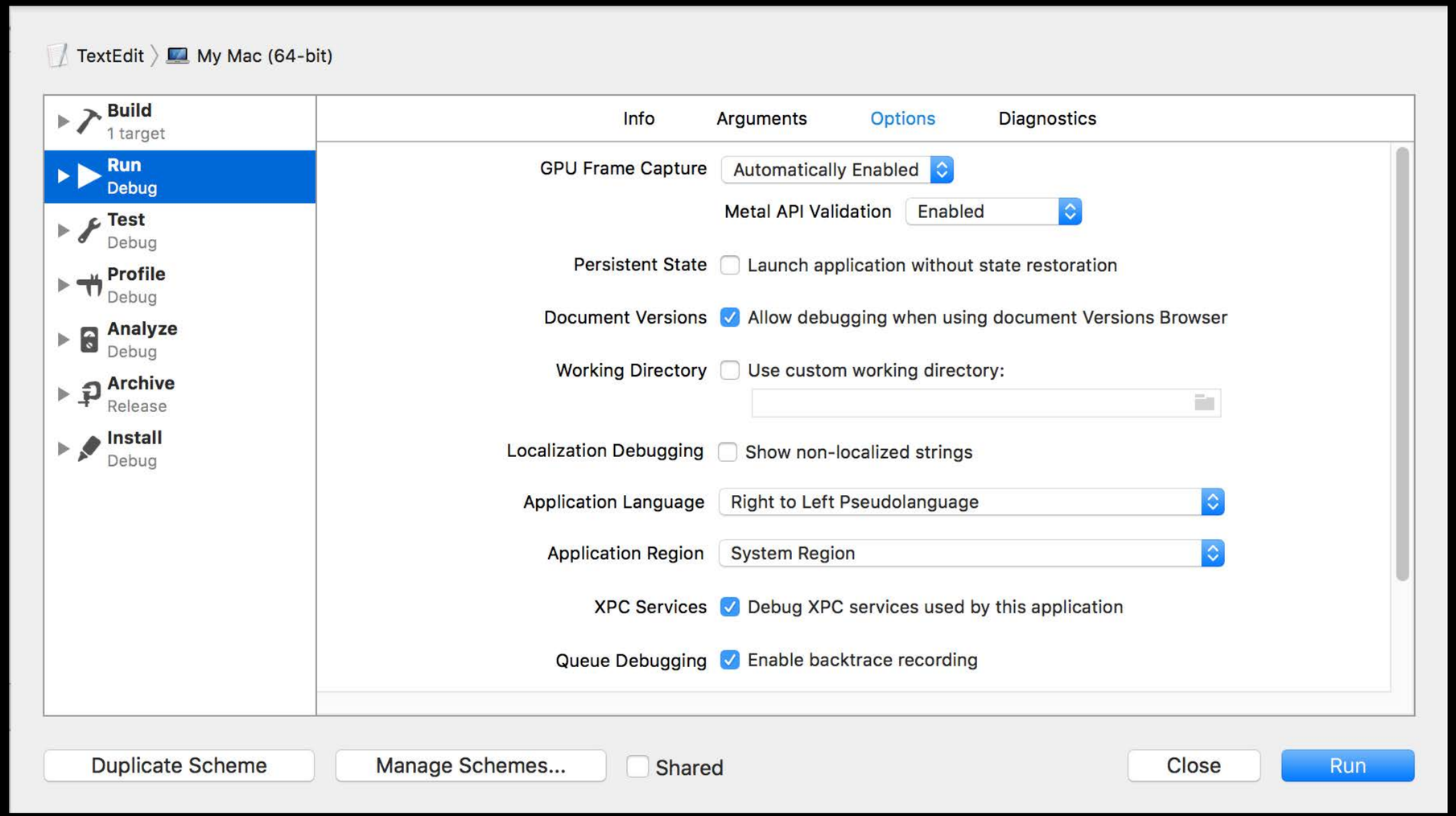
NSPageController

NSButton

....

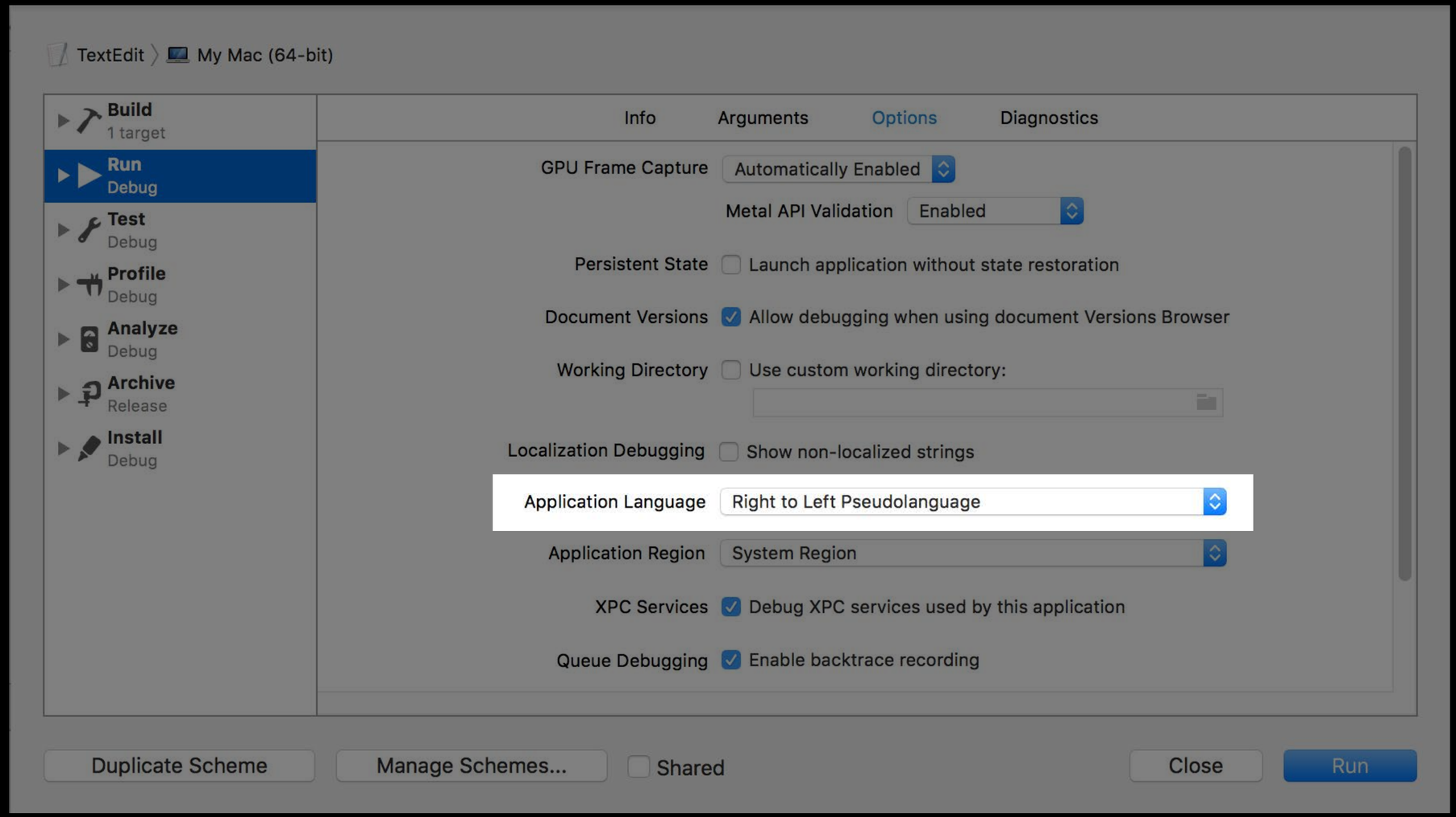
# Right-to-Left Support

## Development tip



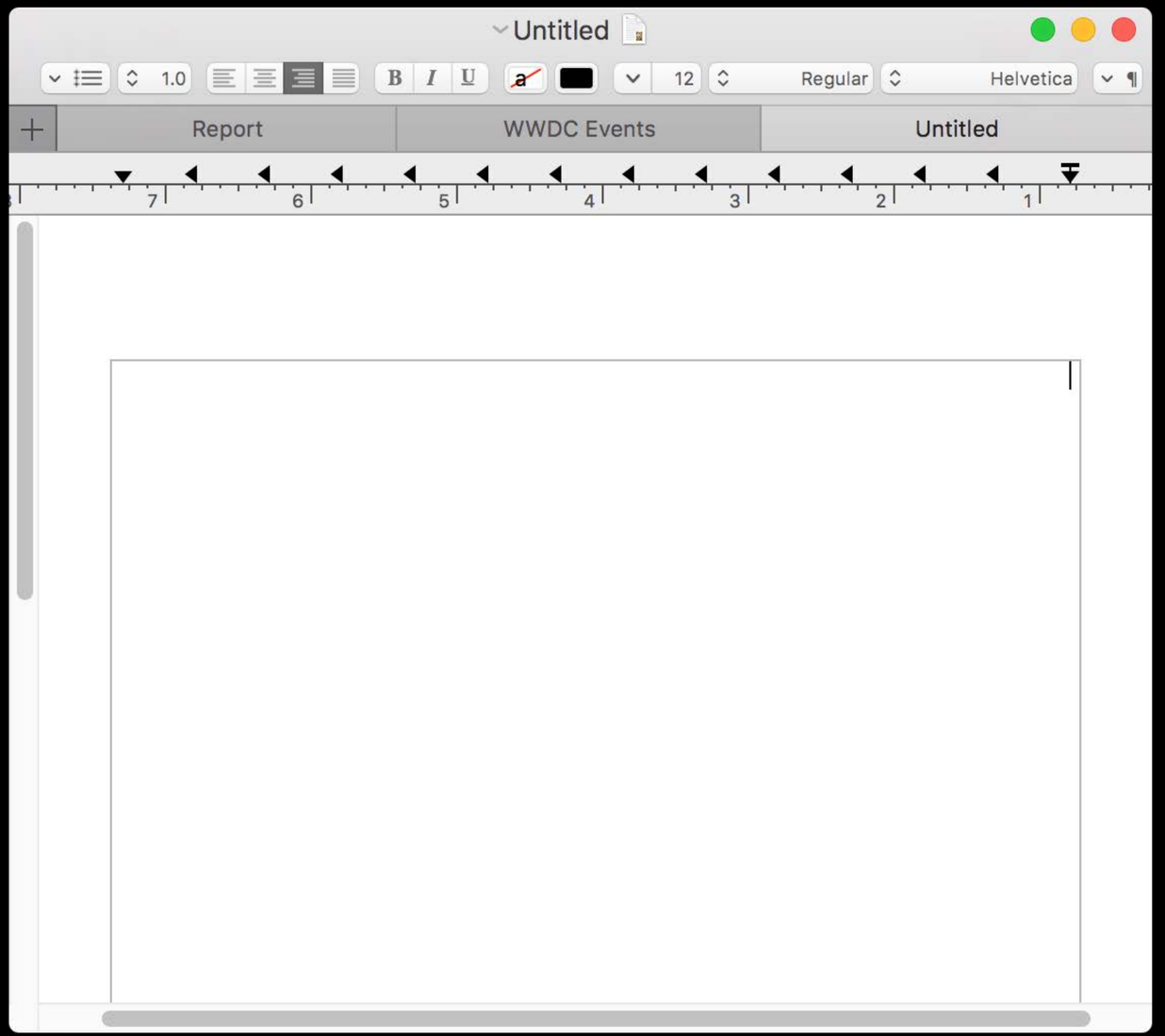
# Right-to-Left Support

## Development tip



# Right-to-Left Support

## Development tip



# Related Sessions

| | | |
|---|---|---|
| What's New in International User Interfaces | Nob Hill | Friday 9:00AM |

# AppKit

Window snapping and tabbing

Right-to-left

Promise drags

Container views

Grid view and auto layout

Wide color

Status items

Control constructors

API refinements

# File Promise Drags

NSFilePromiseProvider and NSFilePromiseReceiver

Supports drag flocking

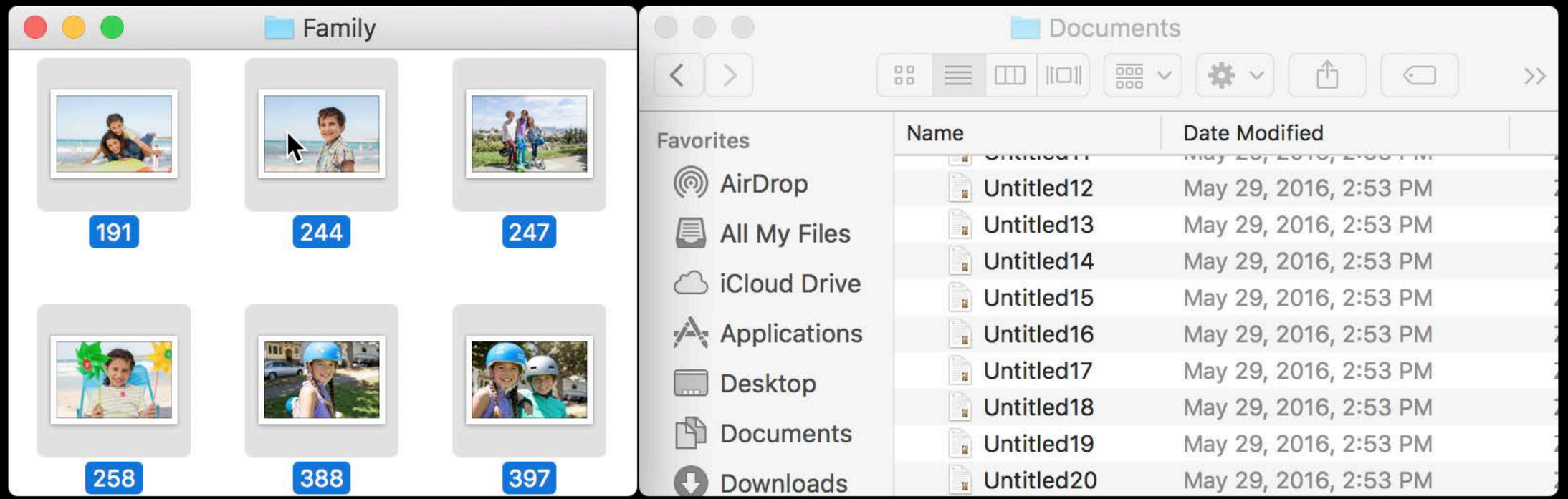
UTI based

Pasteboard Writer/Reader compliant

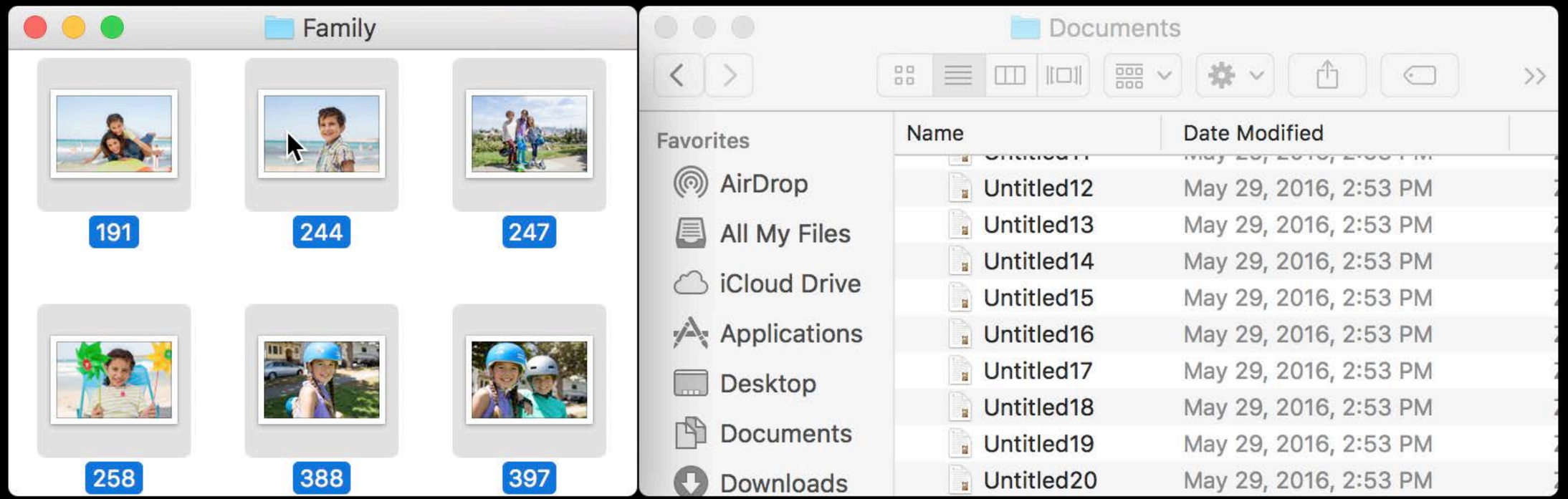
File coordinated when possible

Backwards compatible

# File Promise Drags



# File Promise Drags



# File Promise Drags

NSFilePromiseProvider & NSFilePromiseReceiver

Supports drag flocking

UTI based

Pasteboard writer/Reader compliant

File coordinated when possible

Backwards compatible

# File Promise Drags


```
public class NSFilePromiseProvider {

   public convenience init(fileType: String, delegate: NSFilePromiseProviderDelegate)

}
```

# File Promise Drags


```
public class NSFilePromiseProviderDelegate {
    public func filePromiseProvider(_ filePromiseProvider: NSFilePromiseProvider,
        fileNameForDestination destinationURL: URL) -> String

    public func filePromiseProvider(_ filePromiseProvider: NSFilePromiseProvider,
        writePromiseTo url: URL, completionHandler: (NSError) -> Void)

    optional public func promiseOperationQueue(for filePromiseProvider: NSFilePromiseProvider)
        -> NSOperationQueue
}
```

# File Promise Drags


```
public class NSFilePromiseProviderDelegate {
    public func filePromiseProvider(_ filePromiseProvider: NSFilePromiseProvider,
        fileNameForDestination destinationURL: URL) -> String

    public func filePromiseProvider(_ filePromiseProvider: NSFilePromiseProvider,
        writePromiseTo url: URL, completionHandler: (NSError) -> Void)

    optional public func promiseOperationQueue(for filePromiseProvider: NSFilePromiseProvider)
        -> NSOperationQueue
}
```

# File Promise Drags


```
// Register to receive file promise drags
view.register(forDraggedTypes: NSFilePromiseReceiver.readableDraggedTypes())
```

# File Promise Drags


```
// Register to receive file promise drags
view.register(forDraggedTypes: NSFilePromiseReceiver.readableDraggedTypes())
```

```
// Get file promises from pasteboard
let filePromises = pasteboard.readObjects(forClasses:[NSFilePromiseReceiver.self], options:nil)
```

# File Promise Drags


```
public class NSFilePromiseReceiver {

    public var fileTypes: [String] { get }

    public func receivePromisedFiles(atDestination destinationDir: URL,
        options: [NSObject : AnyObject] = [:], operationQueue: OperationQueue,
        reader: (URL, NSError?) -> Void)

    public var fileNames: [String] { get }
}
```

# File Promise Drags


```
public class NSFilePromiseReceiver {

    public var fileTypes: [String] { get }

    public func receivePromisedFiles(atDestination destinationDir: URL,
        options: [NSObject : AnyObject] = [:], operationQueue: OperationQueue,
        reader: (URL, NSError?) -> Void)

    public var fileNames: [String] { get }
}
```

# AppKit

Window snapping and tabbing

Right-to-left

Promise drags

Container views

Grid view and auto layout

Wide color

Status items

Control constructors

API refinements

# NSCollectionView

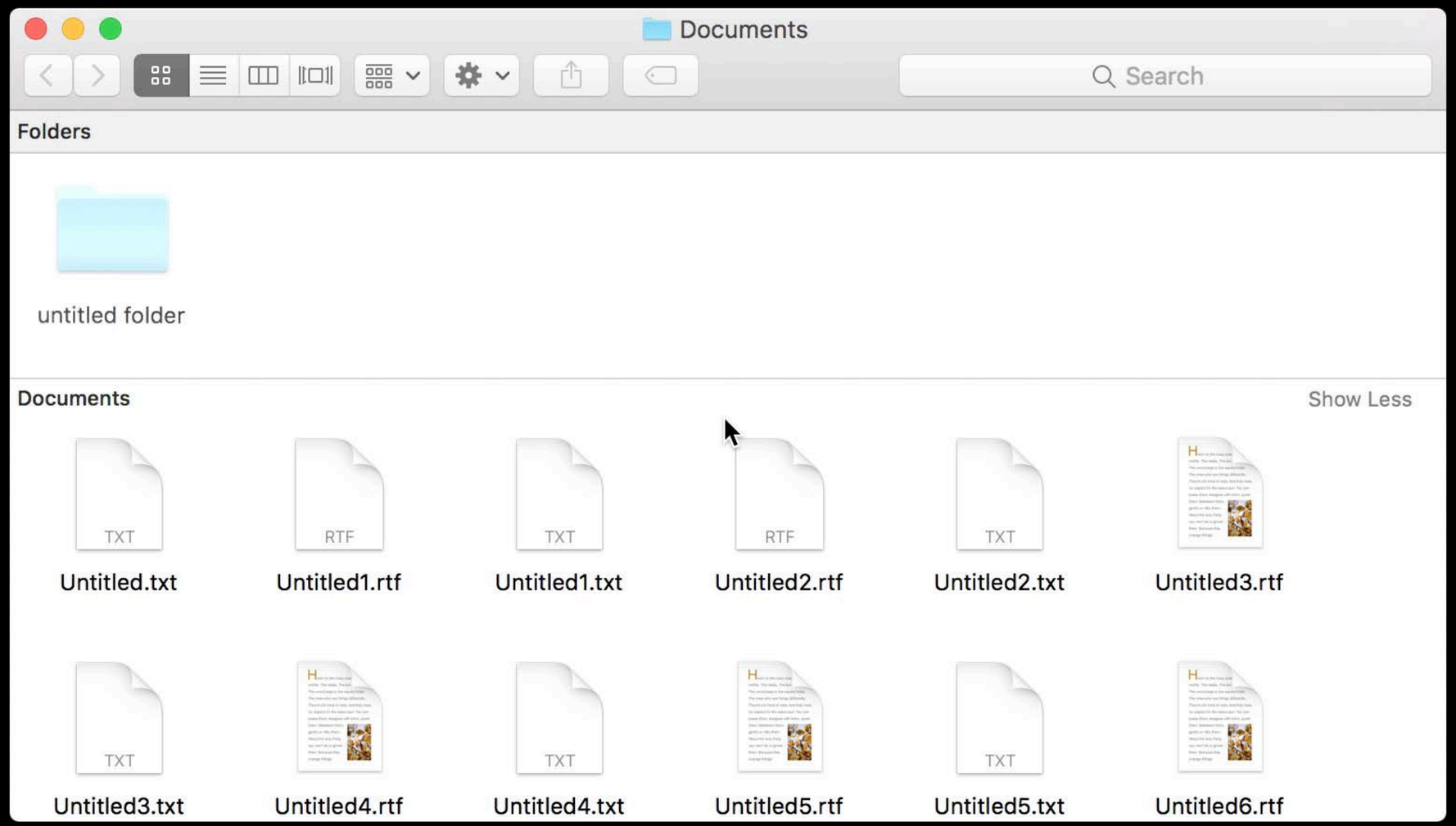
# NSCollectionView

Background view can be configured to scroll

```
extension NSCollectionView
public var backgroundViewScrollsWithContent: Bool
```

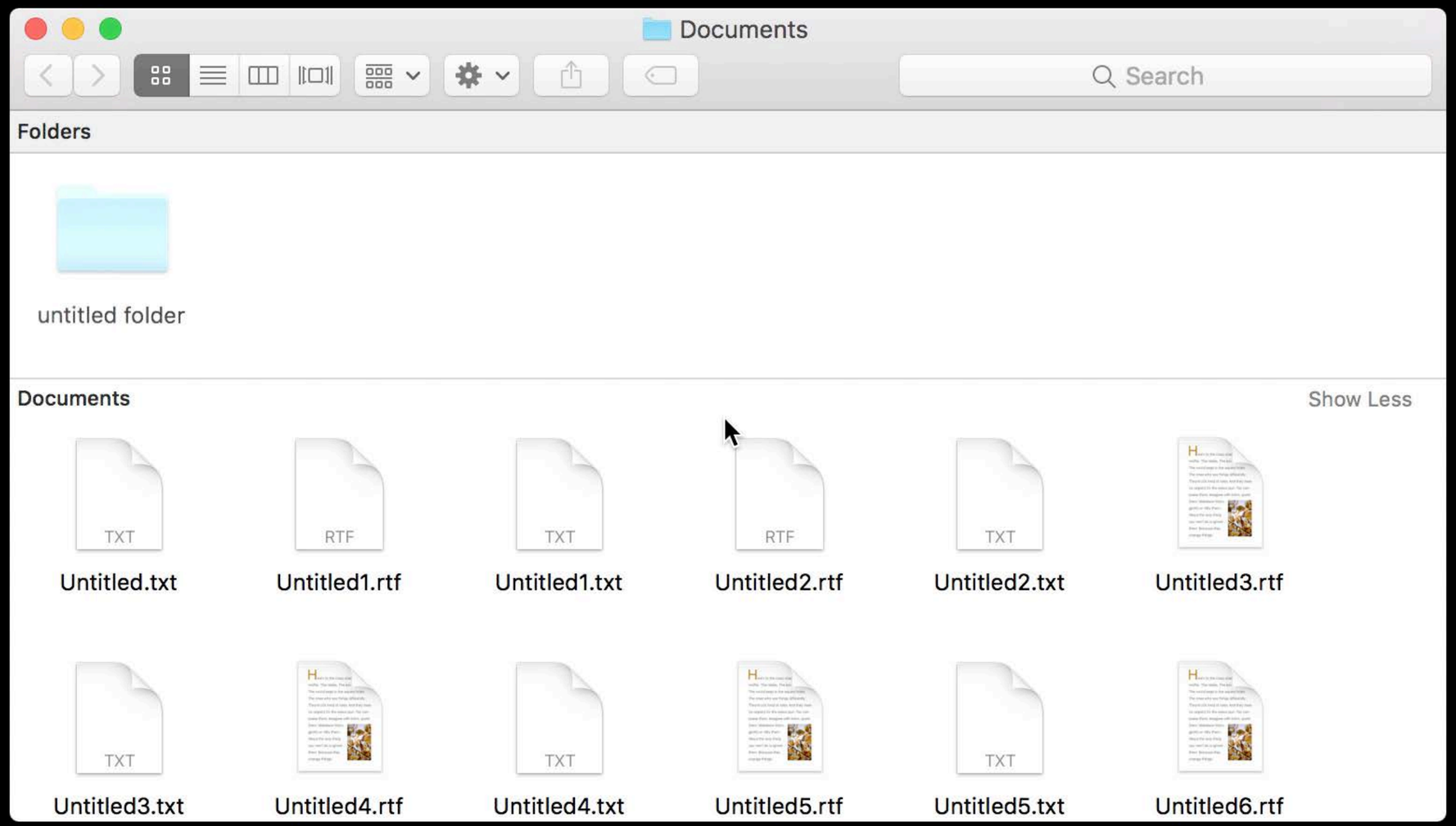
# NSCollectionView

## Optional "floating" headers and footers



# NSCollectionView

## Optional "floating" headers and footers



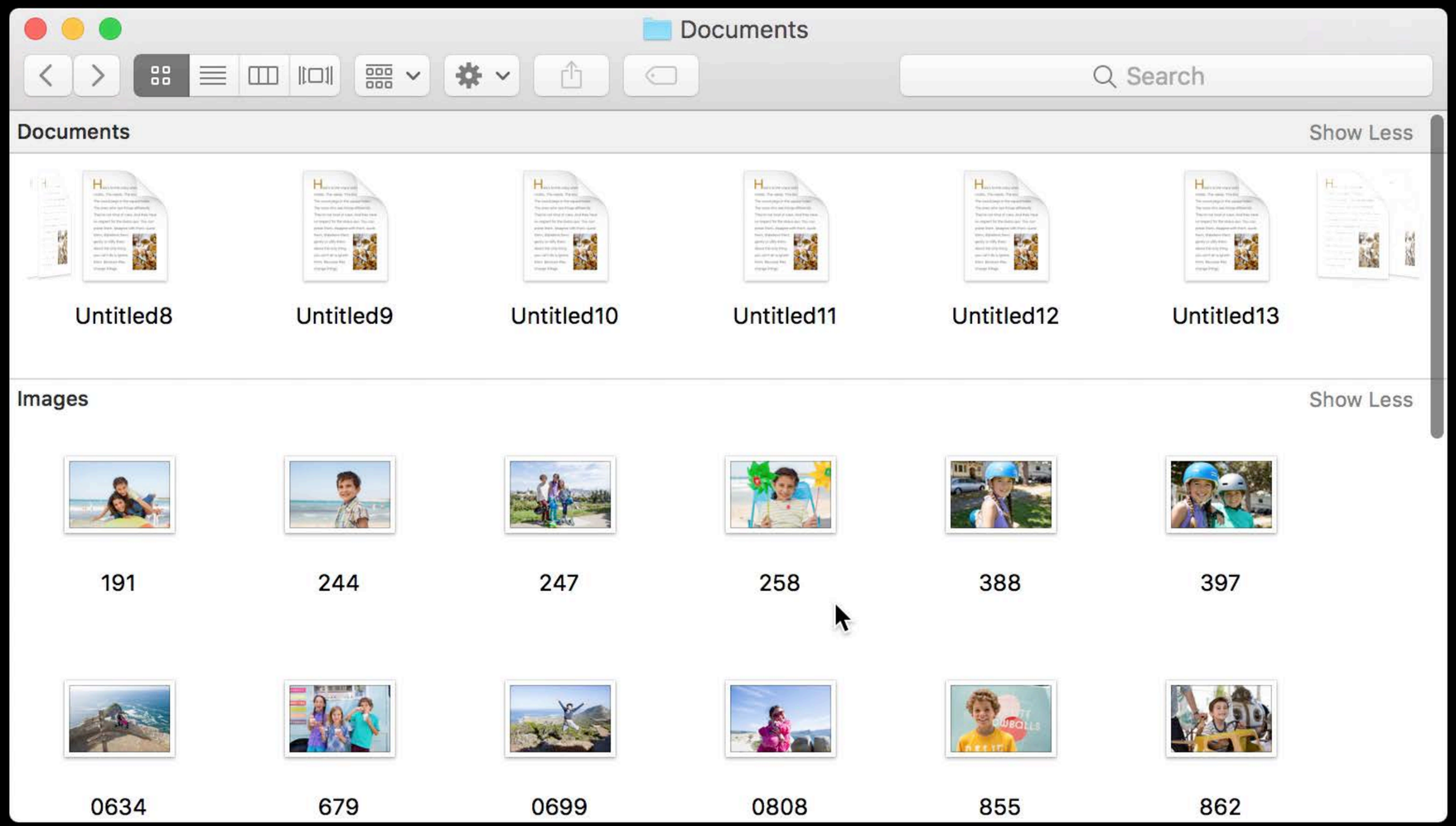
# NSCollectionView

Optional “floating” headers and footers

```
extension NSCollectionViewFlowLayout
public var sectionHeadersPinToVisibleBounds: Bool
public var sectionFootersPinToVisibleBounds: Bool
```

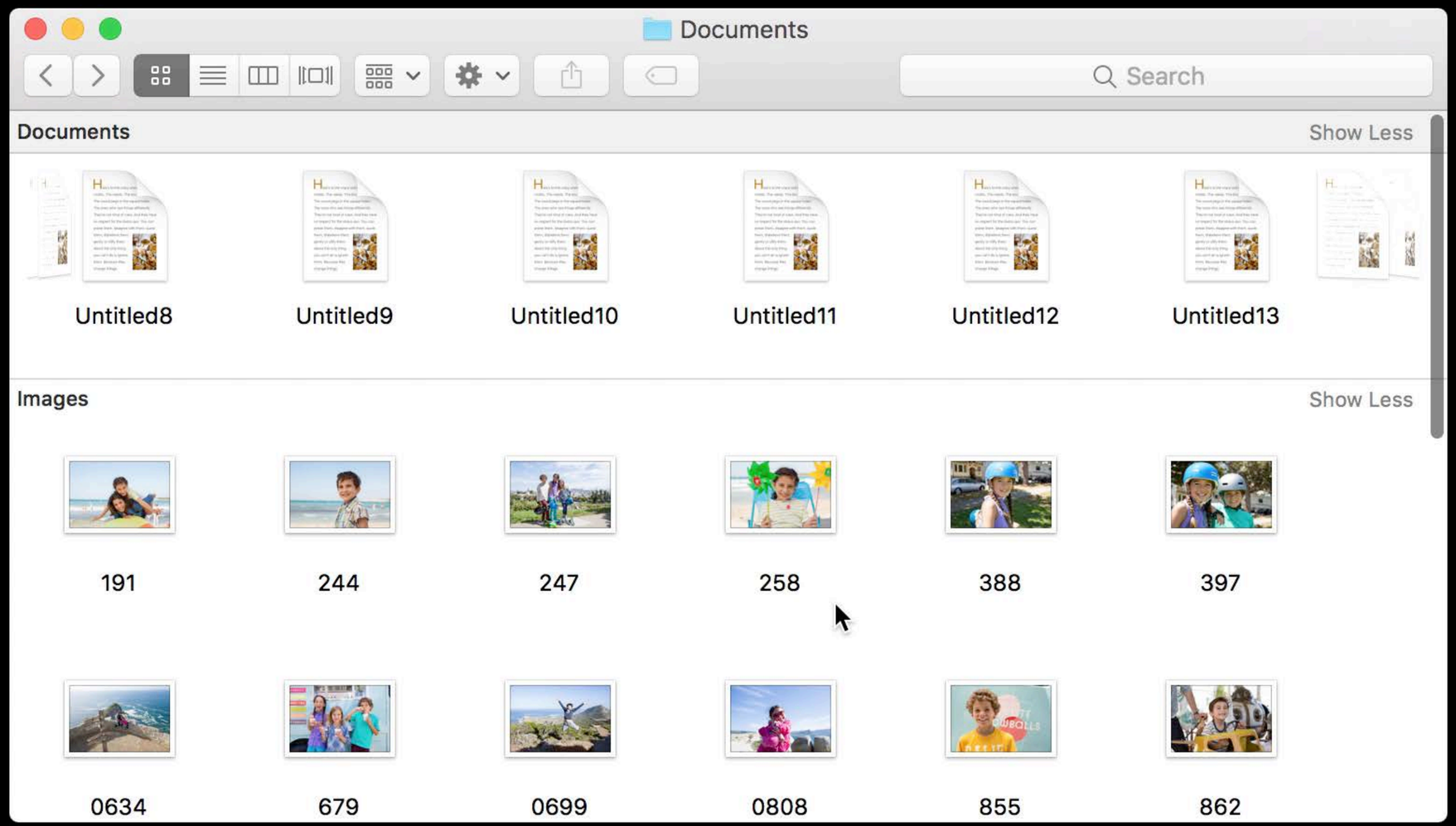
# NSCollectionView

Optionally collapse any section to a single, horizontally scrollable row



# NSCollectionView

Optionally collapse any section to a single, horizontally scrollable row



# NSCollectionView

Optionally collapse any section to a single, horizontally scrollable row

```
extension NSCollectionView
@IBAction public func toggleSectionCollapse(_ sender: AnyObject)
```

# NSCollectionView

Optionally collapse any section to a single, horizontally scrollable row

```
extension NSCollectionView
@IBAction public func toggleSectionCollapse(_ sender: AnyObject)
```

```
public protocol NSCollectionViewSectionHeaderView {
   @IBOutlet optional public var sectionCollapseButton: NSButton? { get set }
}
```

# NSTableView

Reload “full width” cell views when column index set is -1

```
tableView.reloadData(forRowIndexes:, columnIndexes:)
```

# NSOutlineView

Reloads cell views associated with ‘item’ when reloadItem() is called

# NSOutlineView

Reloads cell views associated with ‘item’ when reloadItem() is called

Strongly references items

```
public var stronglyReferencesItems: Bool
```

# AppKit

Window snapping and tabbing

Right-to-left

Promise drags

Container views

Grid view and auto layout

Wide color

Status items

Control constructors

API refinements

# NSGridView

# NSGridView

Auto Layout container view

# NSGridView

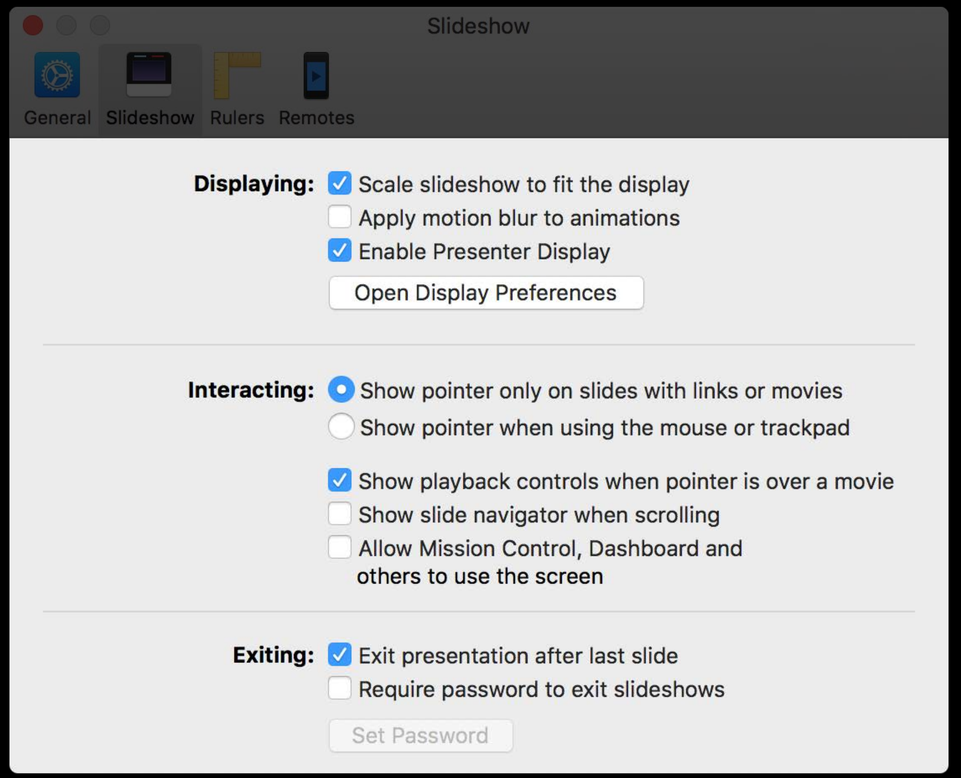
Auto Layout container view

Intersecting rows and columns

# NSGridView

Auto Layout container view

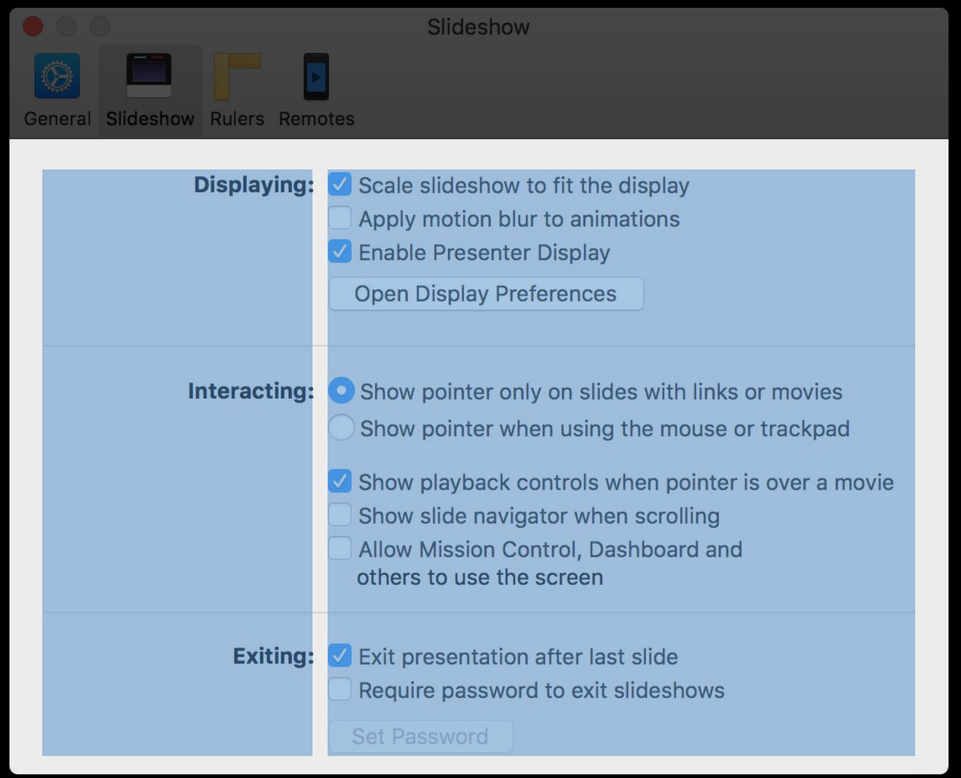
Intersecting rows and columns



# NSGridView

Auto Layout container view

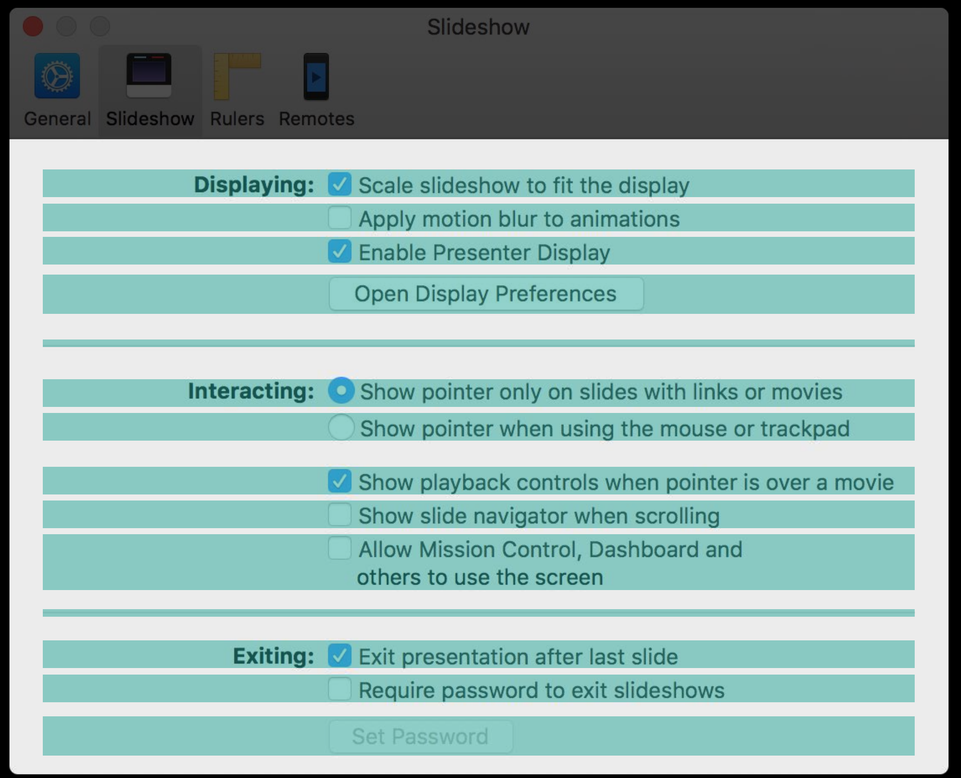
Intersecting rows and columns



# NSGridView

Auto Layout container view

Intersecting rows and columns

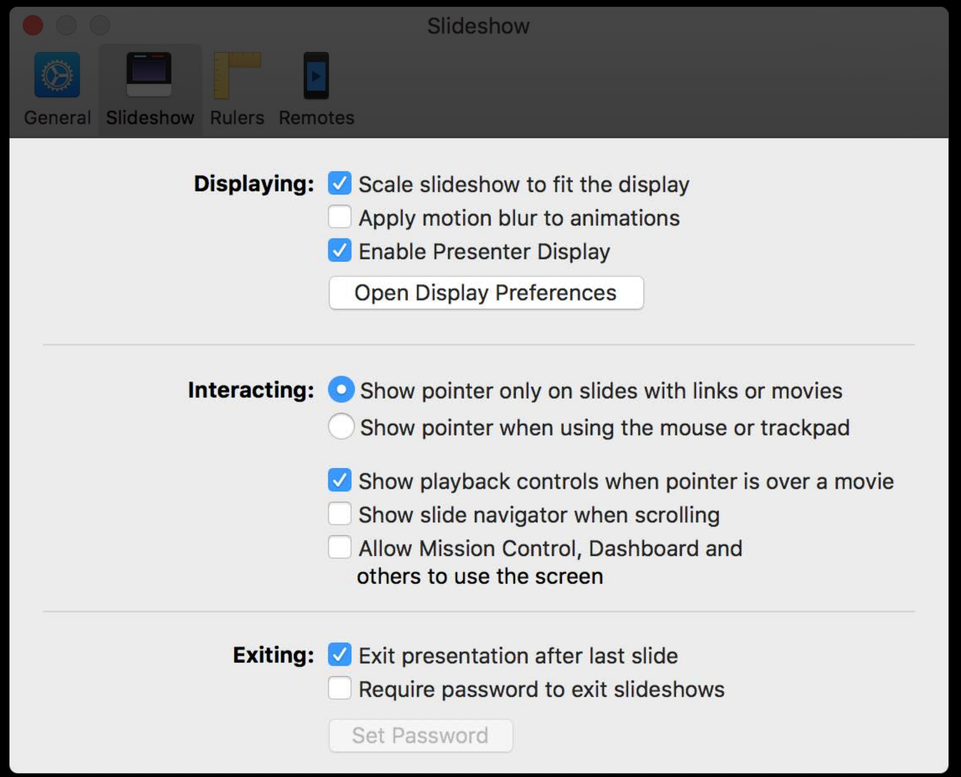


# NSGridView

Auto Layout container view

Intersecting rows and columns

- Alignment

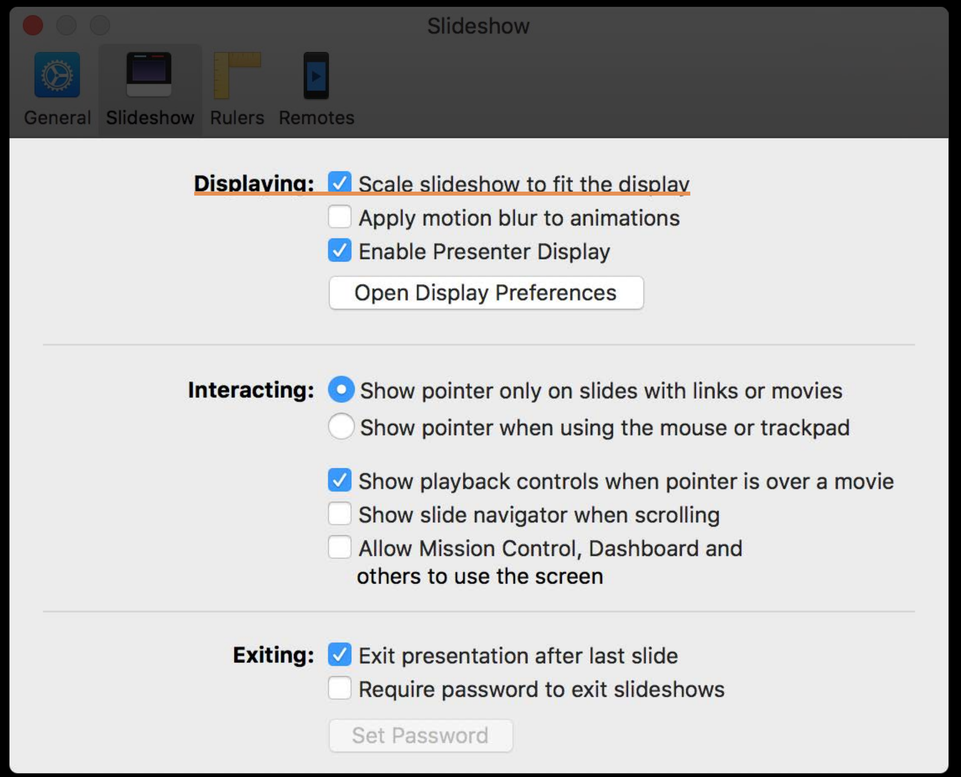


# NSGridView

Auto Layout container view

Intersecting rows and columns

- Alignment

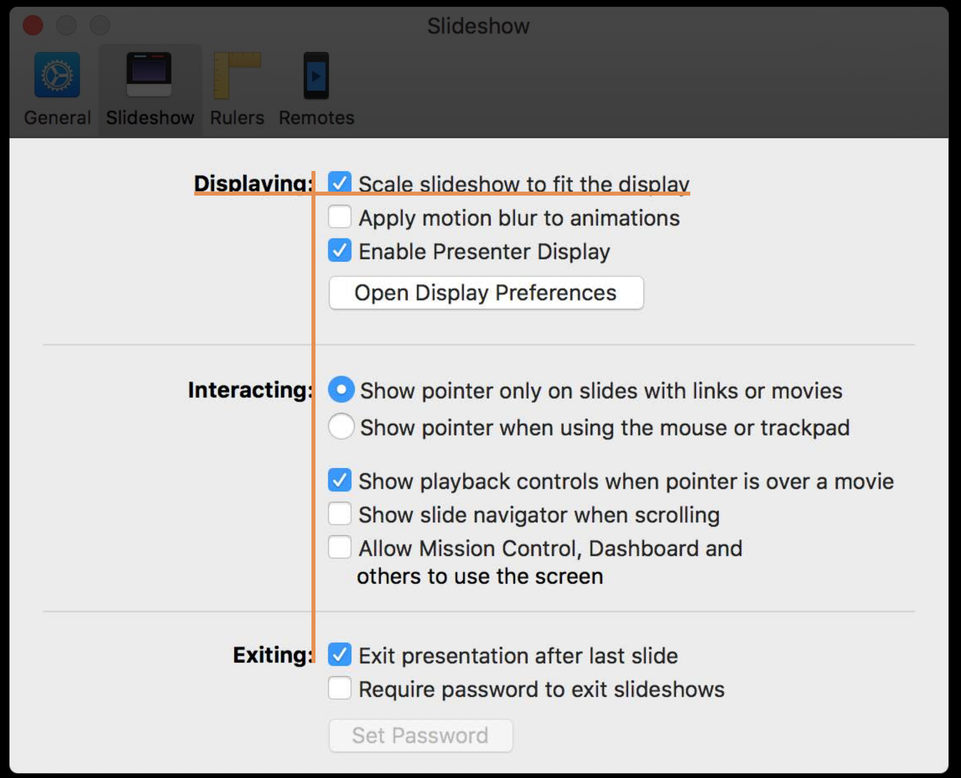


# NSGridView

Auto Layout container view

Intersecting rows and columns

- Alignment

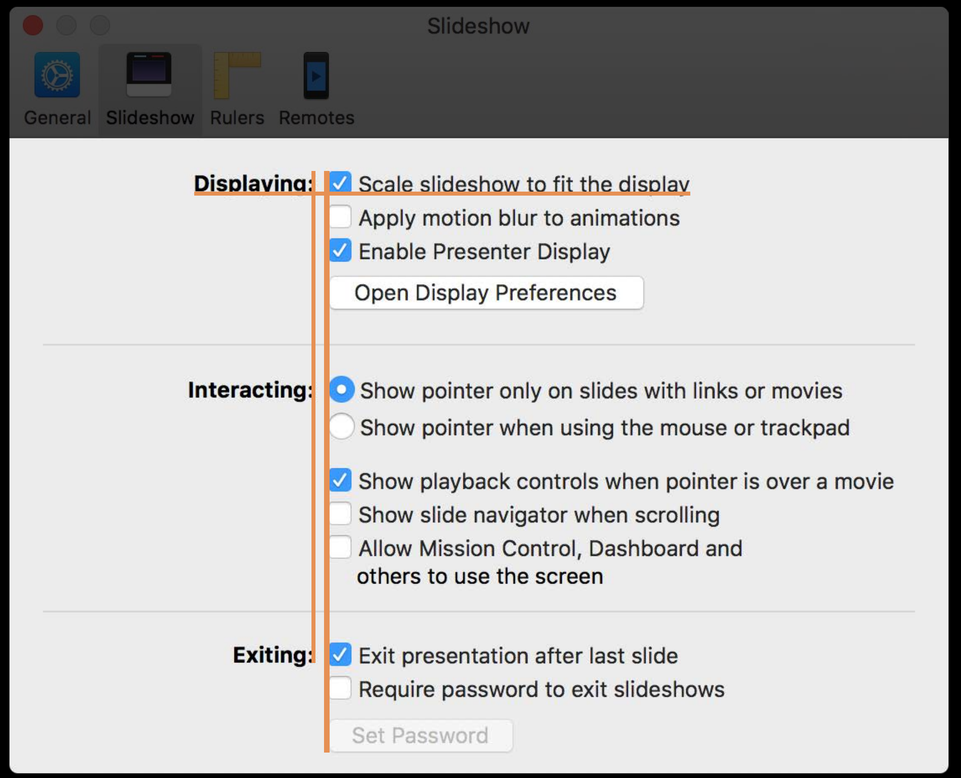


# NSGridView

Auto Layout container view

Intersecting rows and columns

- Alignment

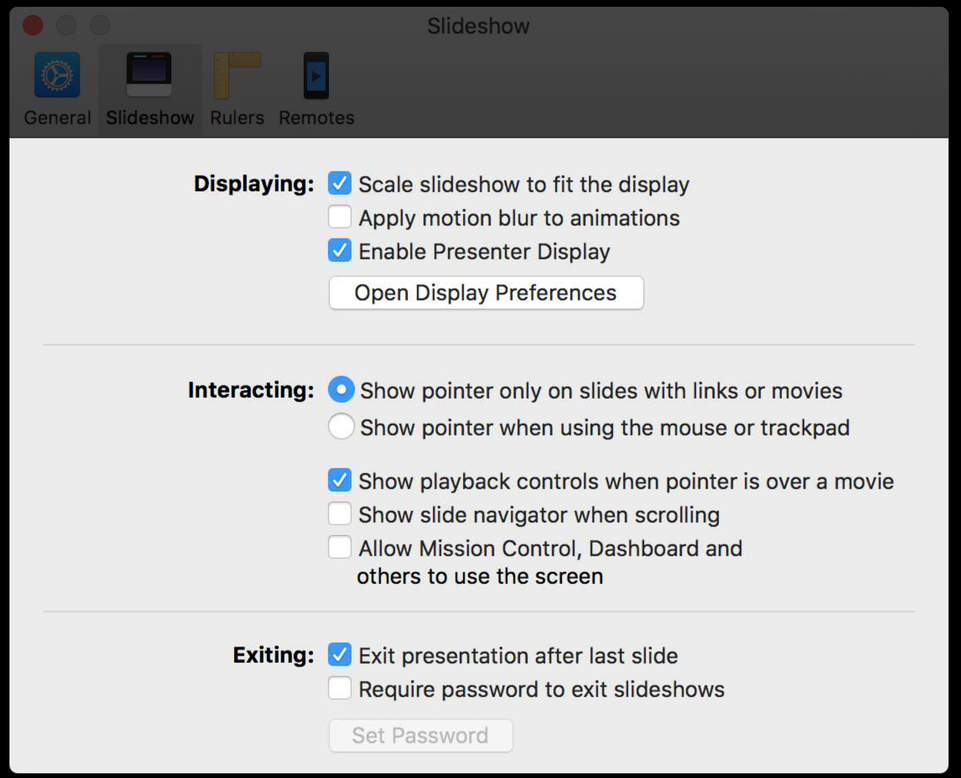


# NSGridView

Auto Layout container view

Intersecting rows and columns

- Alignment
- Spacing and padding

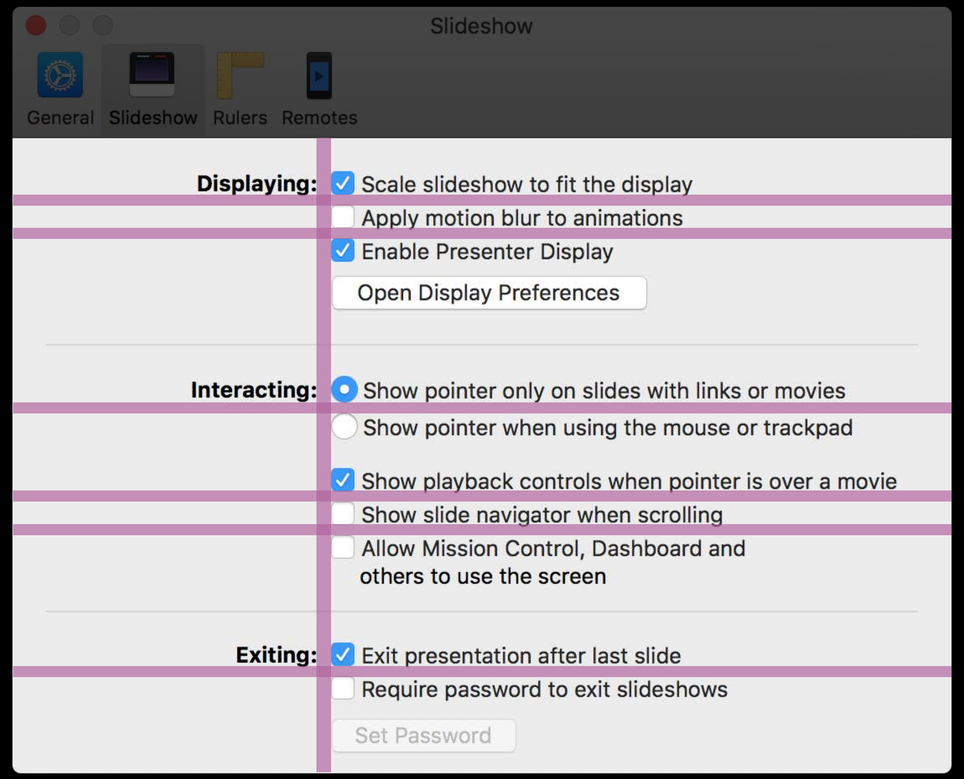


# NSGridView

Auto Layout container view

Intersecting rows and columns

- Alignment
- Spacing and padding

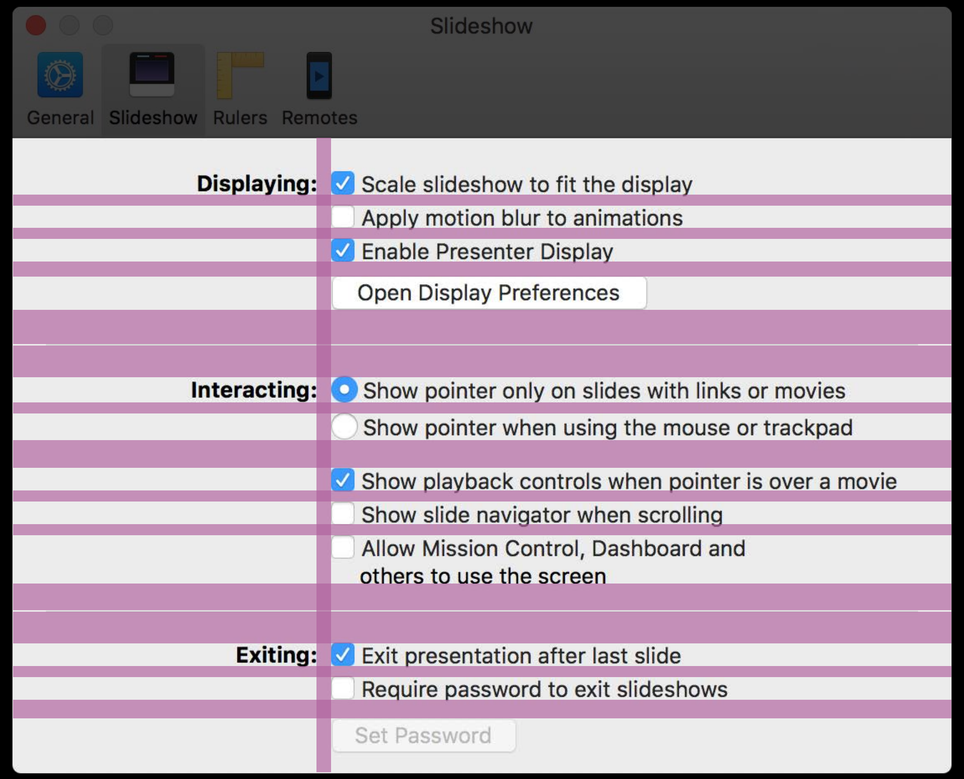


# NSGridView

Auto Layout container view

Intersecting rows and columns

- Alignment
- Spacing and padding

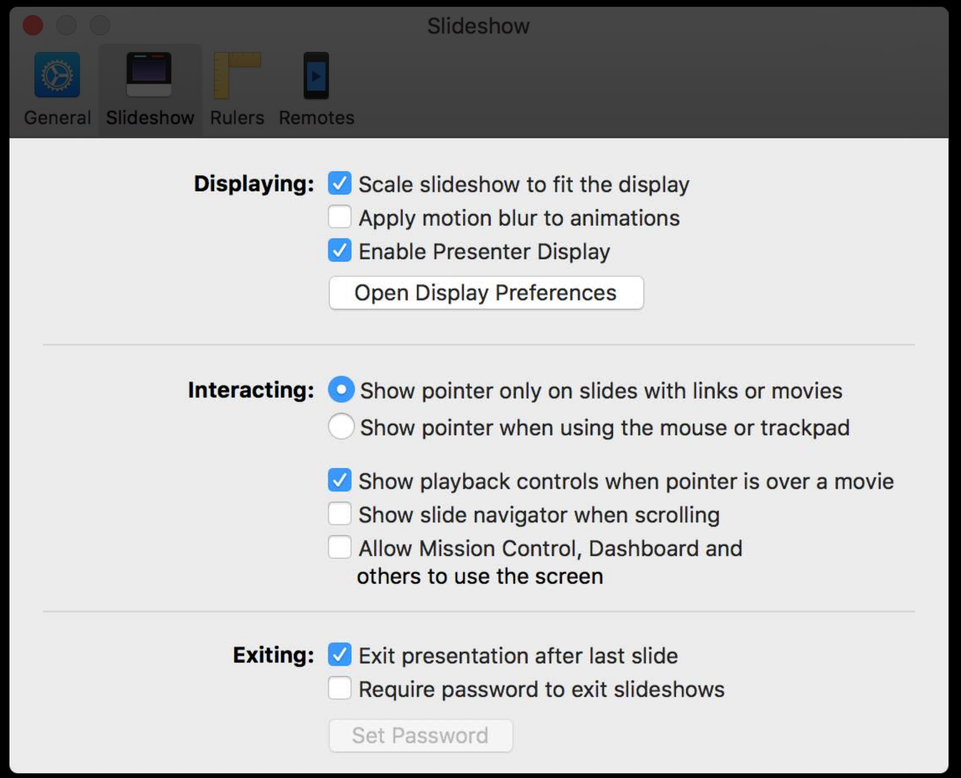


# NSGridView

Auto Layout container view

Intersecting rows and columns

- Alignment
- Spacing and padding
- Cell merging

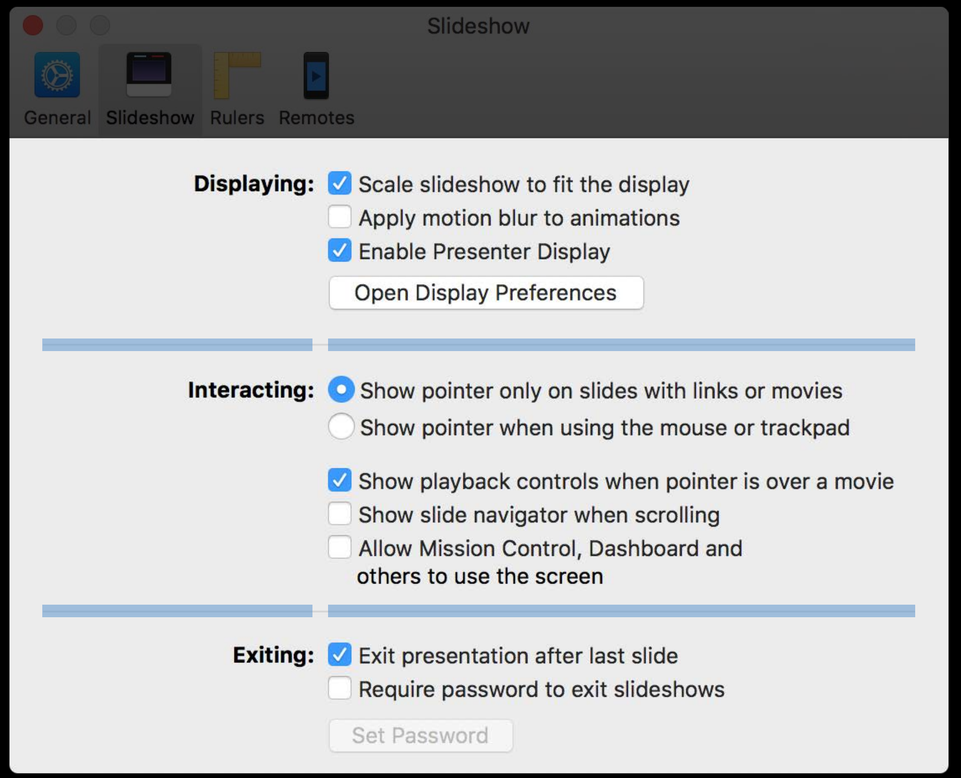


# NSGridView

Auto Layout container view

Intersecting rows and columns

- Alignment
- Spacing and padding
- Cell merging

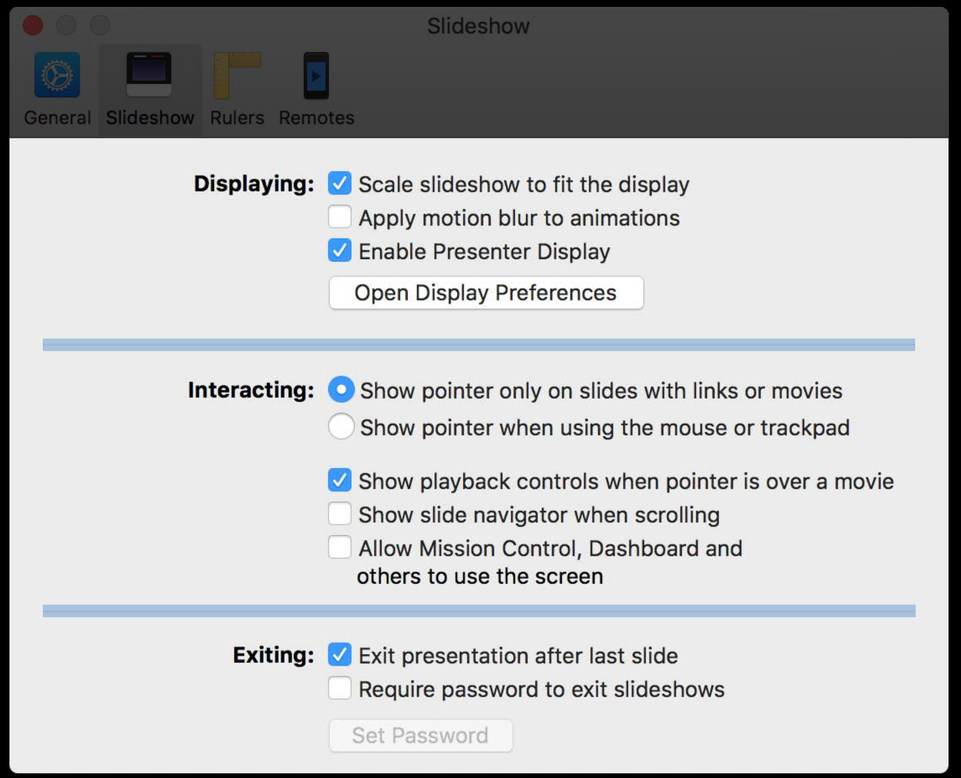


# NSGridView

Auto Layout container view

Intersecting rows and columns

- Alignment
- Spacing and padding
- Cell merging



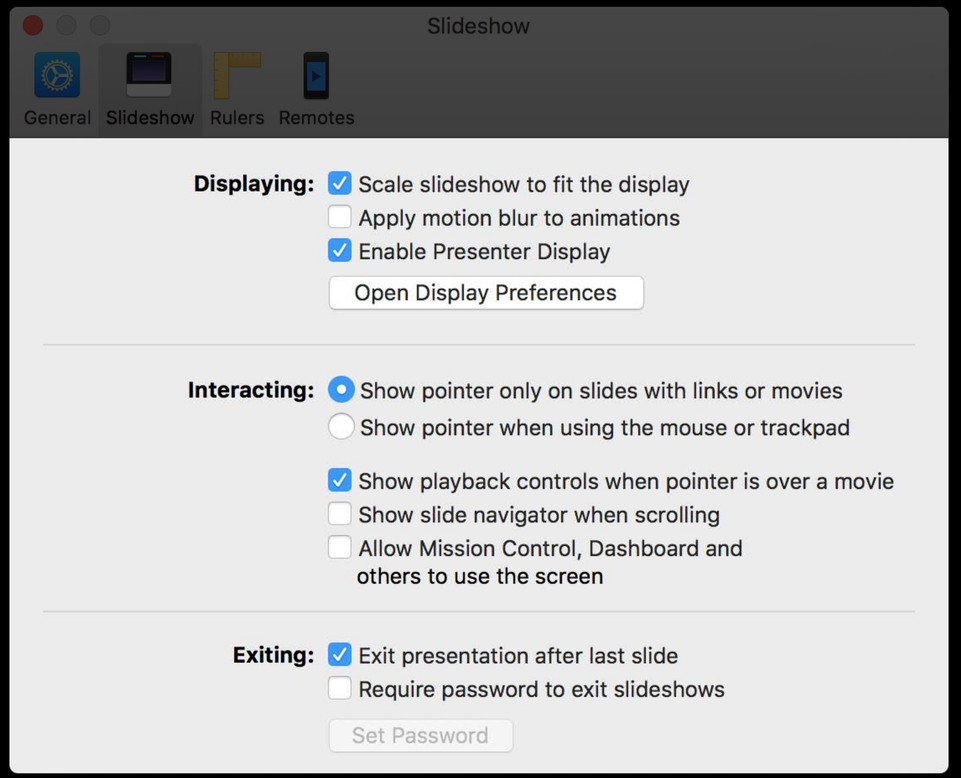
# NSGridView

Auto Layout container view

Intersecting rows and columns

- Alignment
- Spacing and padding
- Cell merging

Dynamic hiding of rows/columns



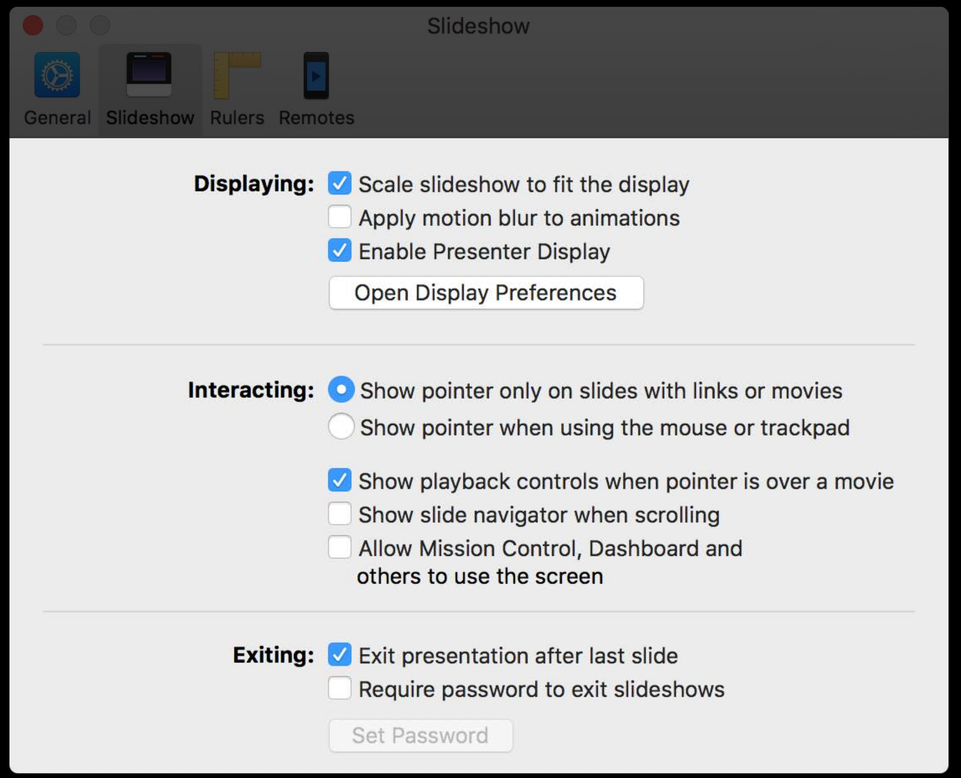
# NSGridView

Auto Layout container view

Intersecting rows and columns

- Alignment
- Spacing and padding
- Cell merging

Dynamic hiding of rows/columns



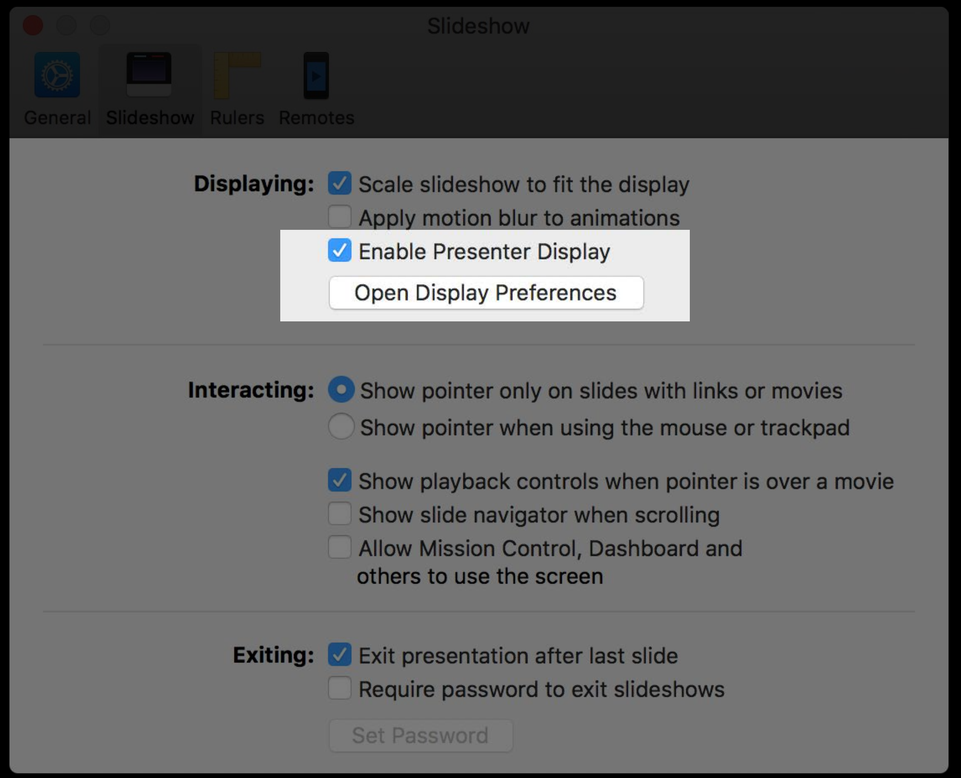
# NSGridView

Auto Layout container view

Intersecting rows and columns

- Alignment
- Spacing and padding
- Cell merging

Dynamic hiding of rows/columns



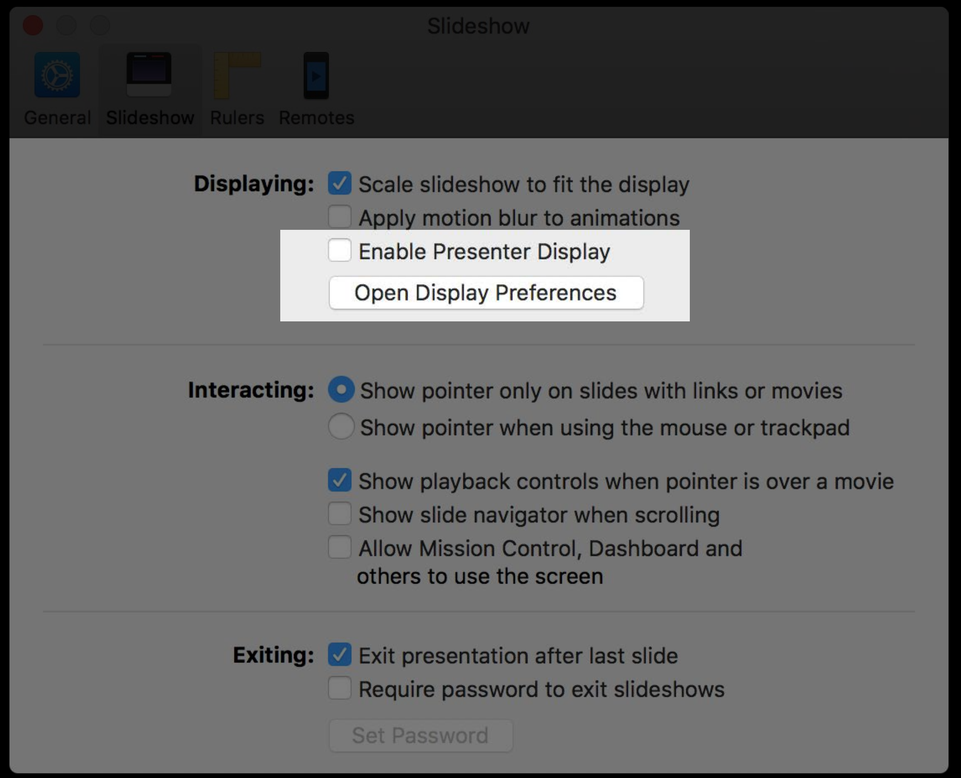
# NSGridView

Auto Layout container view

Intersecting rows and columns

- Alignment
- Spacing and padding
- Cell merging

Dynamic hiding of rows/columns



# NSGridView

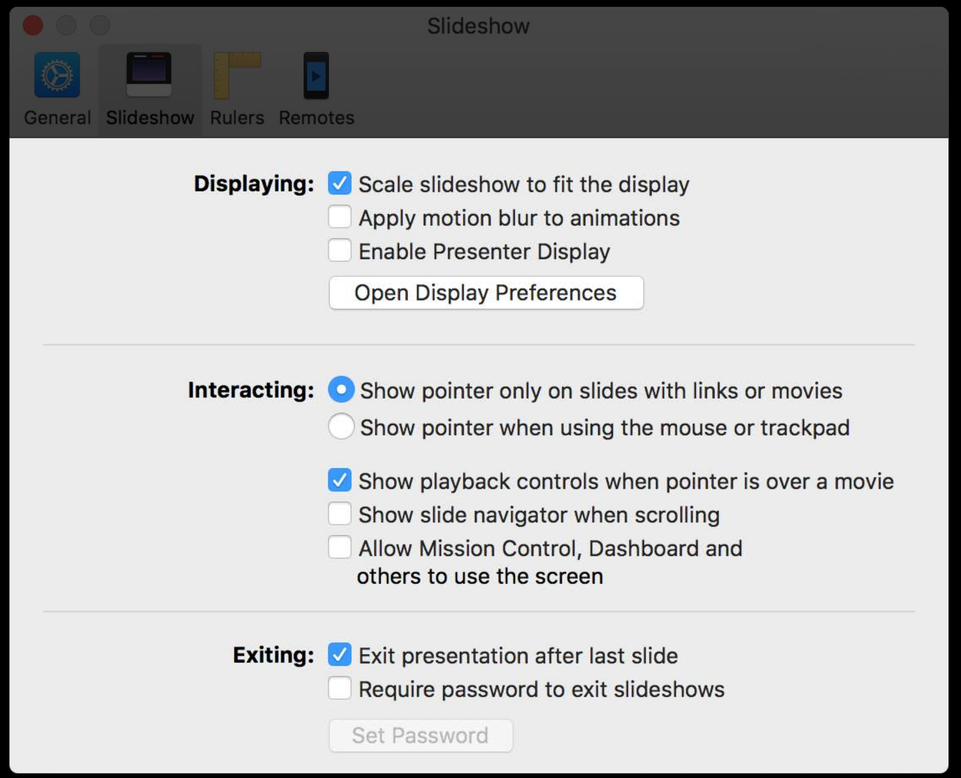
Auto Layout container view

Intersecting rows and columns

- Alignment
- Spacing and padding
- Cell merging

Dynamic hiding of rows/columns

```
displayPrefButtonRow.hidden = true
```



# NSGridView

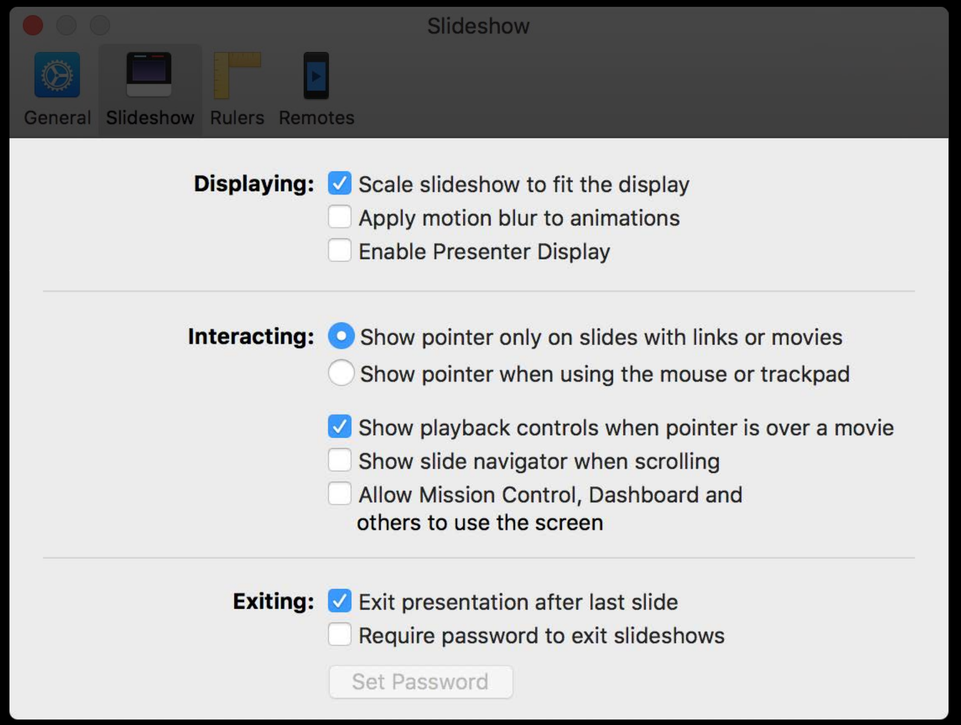
Auto Layout container view

Intersecting rows and columns

- Alignment
- Spacing and padding
- Cell merging

Dynamic hiding of rows/columns

```
displayPrefButtonRow.hidden = true
```



# Auto Layout Improvements

## AppKit layout cycle cleanup

# Auto Layout Improvements

## AppKit layout cycle cleanup

General deferred layout pass

# Auto Layout Improvements

## AppKit layout cycle cleanup

General deferred layout pass

- needsLayout = true

# Auto Layout Improvements

## AppKit layout cycle cleanup

General deferred layout pass

- needsLayout = true
- layout() no longer called twice for layer-backed views

# Auto Layout Improvements

## AppKit layout cycle cleanup

General deferred layout pass

- needsLayout = true
- layout() no longer called twice for layer-backed views
- Implicit dirtying is less frequent

# Auto Layout Improvements

## AppKit layout cycle cleanup

General deferred layout pass

- needsLayout = true
- layout() no longer called twice for layer-backed views
- Implicit dirtying is less frequent

Manual layout

# Auto Layout Improvements

## AppKit layout cycle cleanup

General deferred layout pass

- needsLayout = true
- layout() no longer called twice for layer-backed views
- Implicit dirtying is less frequent

Manual layout

- Calling super.layout() is no longer necessary

# Auto Layout Improvements

## AppKit layout cycle cleanup

General deferred layout pass

- needsLayout = true
- layout() no longer called twice for layer-backed views
- Implicit dirtying is less frequent

Manual layout

- Calling super.layout() is no longer necessary
- Dirtying of layout will cause additional passes

# Auto Layout Improvements

# Auto Layout Improvements

Layout loop debugging on macOS 10.12 and iOS 10

# Auto Layout Improvements

Layout loop debugging on macOS 10.12 and iOS 10

NSLayoutConstraint API

- New firstAnchor and secondAnchor properties
- firstItem is now nullable, firstAttribute can be .notAnAttribute

# Auto Layout Improvements

Layout loop debugging on macOS 10.12 and iOS 10

NSLayoutConstraint API

- New firstAnchor and secondAnchor properties
- firstItem is now nullable, firstAttribute can be .notAnAttribute

Incremental auto layout adoption in Interface Builder

# Auto Layout Improvements

Layout loop debugging on macOS 10.12 and iOS 10

NSLayoutConstraint API

- New firstAnchor and secondAnchor properties
- firstItem is now nullable, firstAttribute can be .notAnAttribute

Incremental auto layout adoption in Interface Builder

What's New in Auto Layout Presidio Friday 3:00PM

# Wide Gamut Color

P3 color space

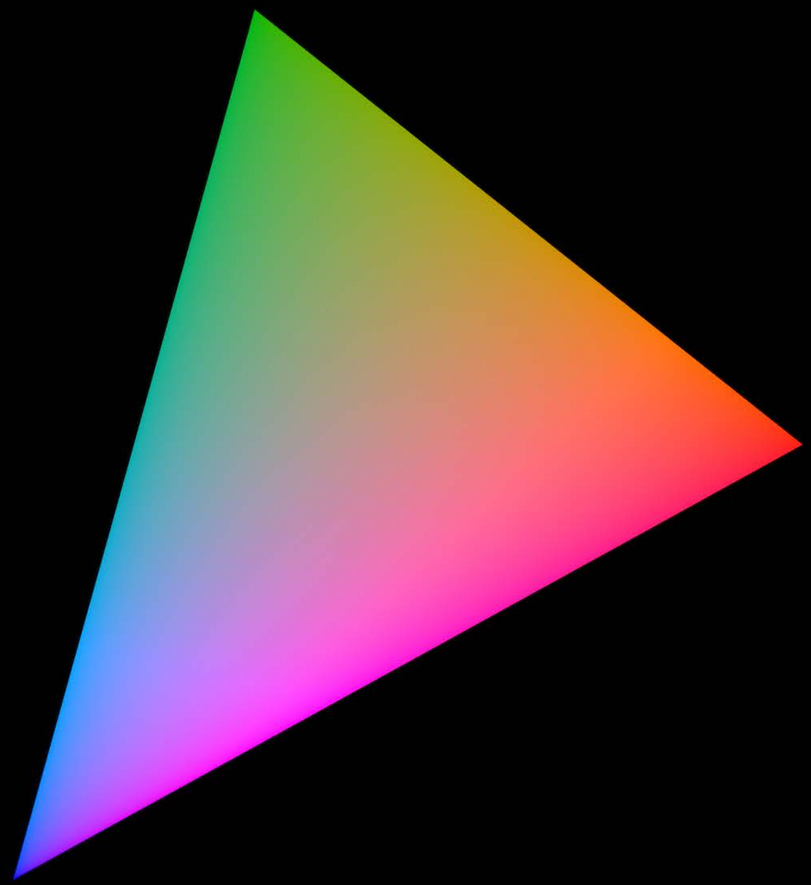
Extended range sRGB

Bit depth

Color panel

# Wide Gamut

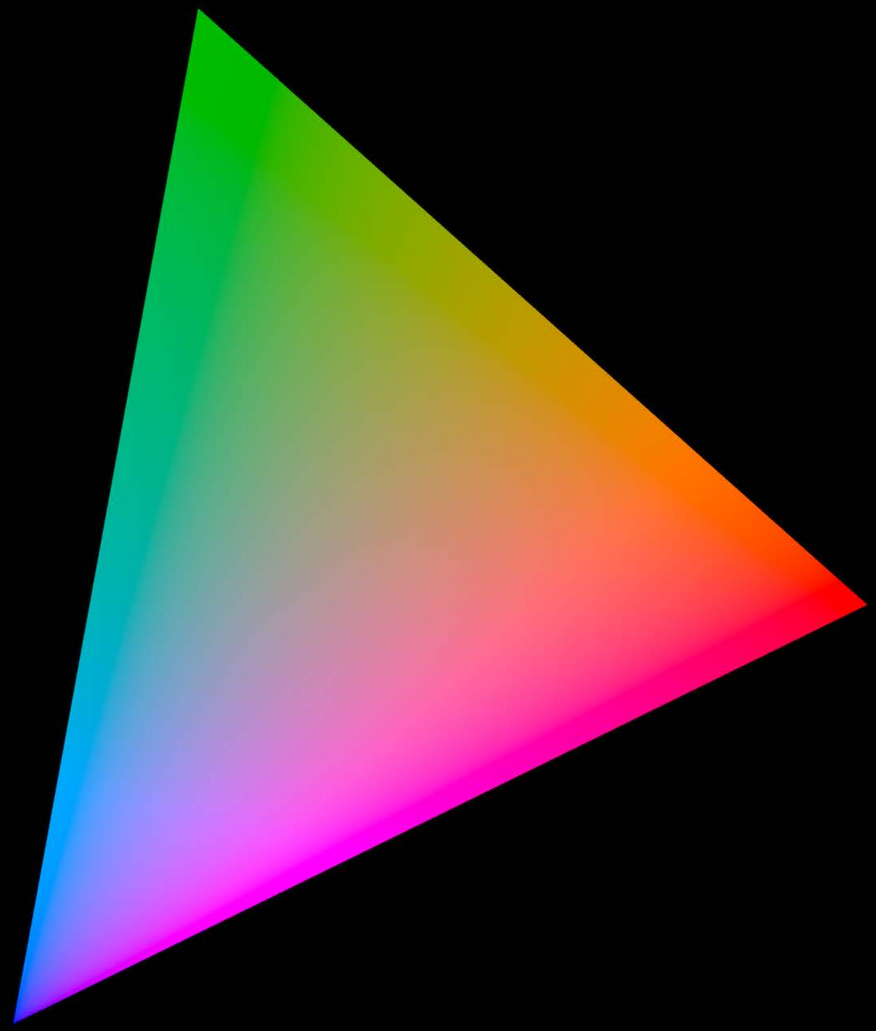
## sRGB



# Wide Gamut

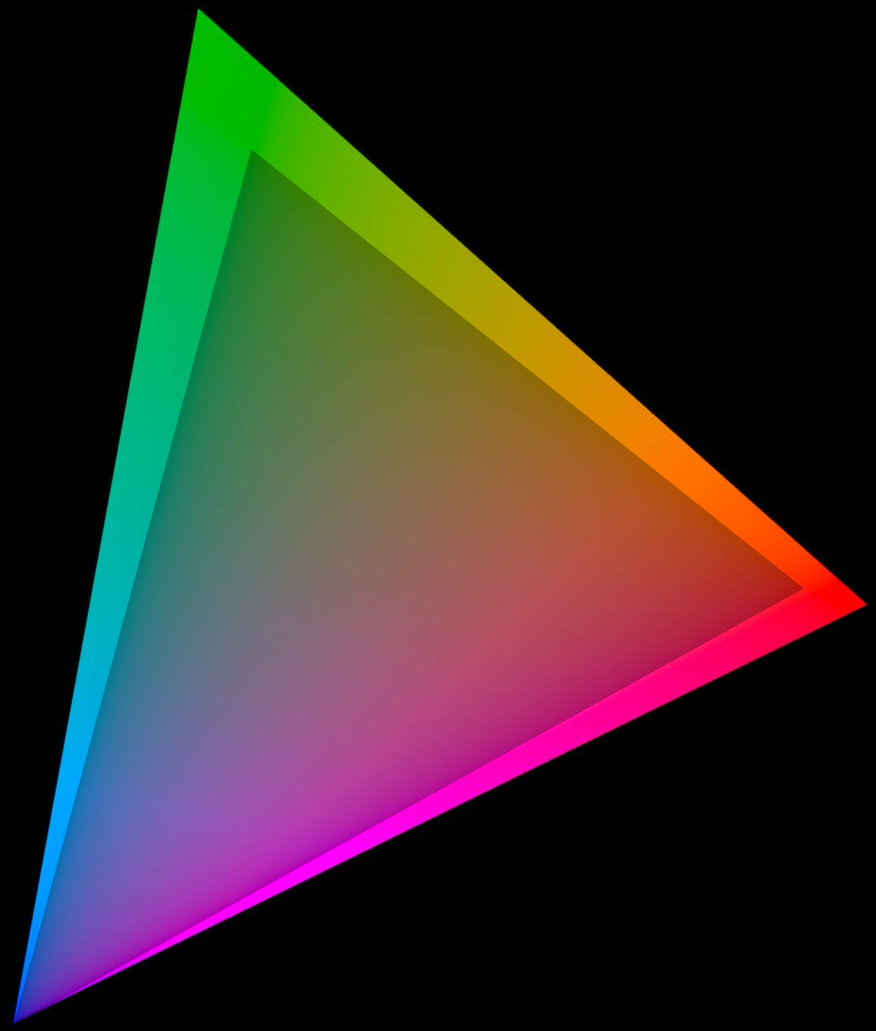
# Wide Gamut

## P3



# Wide Gamut

## P3



**EMERALD**

17-5641TCX

2013

Color of the Year

# EMERALD

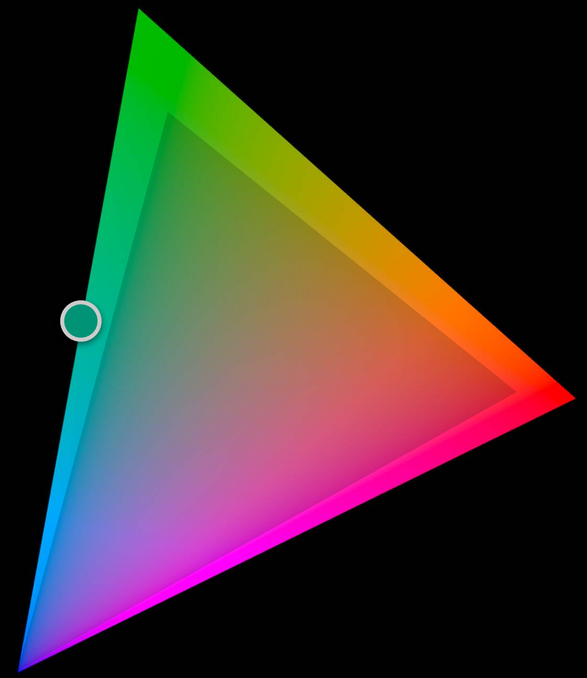
17-5641TCX

2013

Color of the Year

# EMERALD

17-5641TCX



**sRGB** -0.26, 0.46, 0.57

**P3** 0.06, 0.46, 0.56

```
// Wide Color API: sRGB & P3

extension NSColorSpace {
   public class func sRGB() -> NSColorSpace
   public class func displayP3() -> NSColorSpace
}

extension NSColor {
   public init(sRGBRed red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)
   public init(displayP3Red red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)
}

extension UIColor {
   public init(displayP3Red red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)
}
```

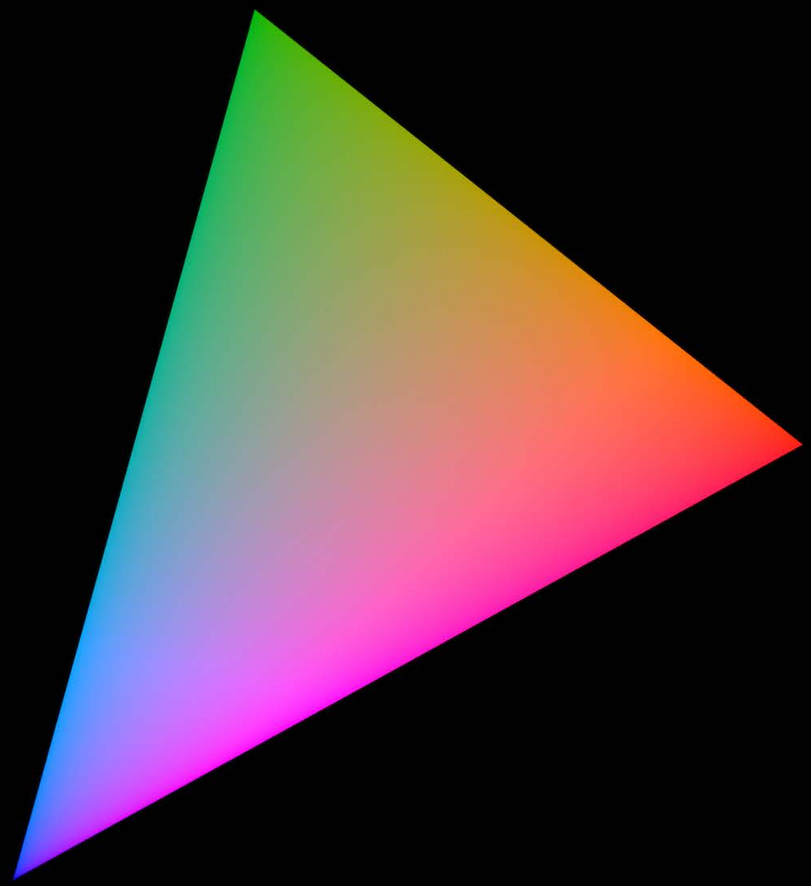
```
// Wide Color API: sRGB & P3

extension NSColorSpace {
    public class func sRGB() -> NSColorSpace
    public class func displayP3() -> NSColorSpace
}

extension NSColor {
    public init(sRGBRed red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)
    public init(displayP3Red red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)
}

extension UIColor {
    public init(displayP3Red red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)
}
```

```
// Wide Color API: sRGB & P3

extension NSColorSpace {
    public class func sRGB() -> NSColorSpace
    public class func displayP3() -> NSColorSpace
}

extension NSColor {
    public init(sRGBRed red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)
    public init(displayP3Red red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)
}

extension UIColor {
    public init(displayP3Red red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)
}
```

```
// Wide Color API: sRGB & P3

extension NSColorSpace {
    public class func sRGB() -> NSColorSpace
    public class func displayP3() -> NSColorSpace
}

extension NSColor {
    public init(sRGBRed red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)
    public init(displayP3Red red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)
}

extension UIColor {
    public init(displayP3Red red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)
}
```

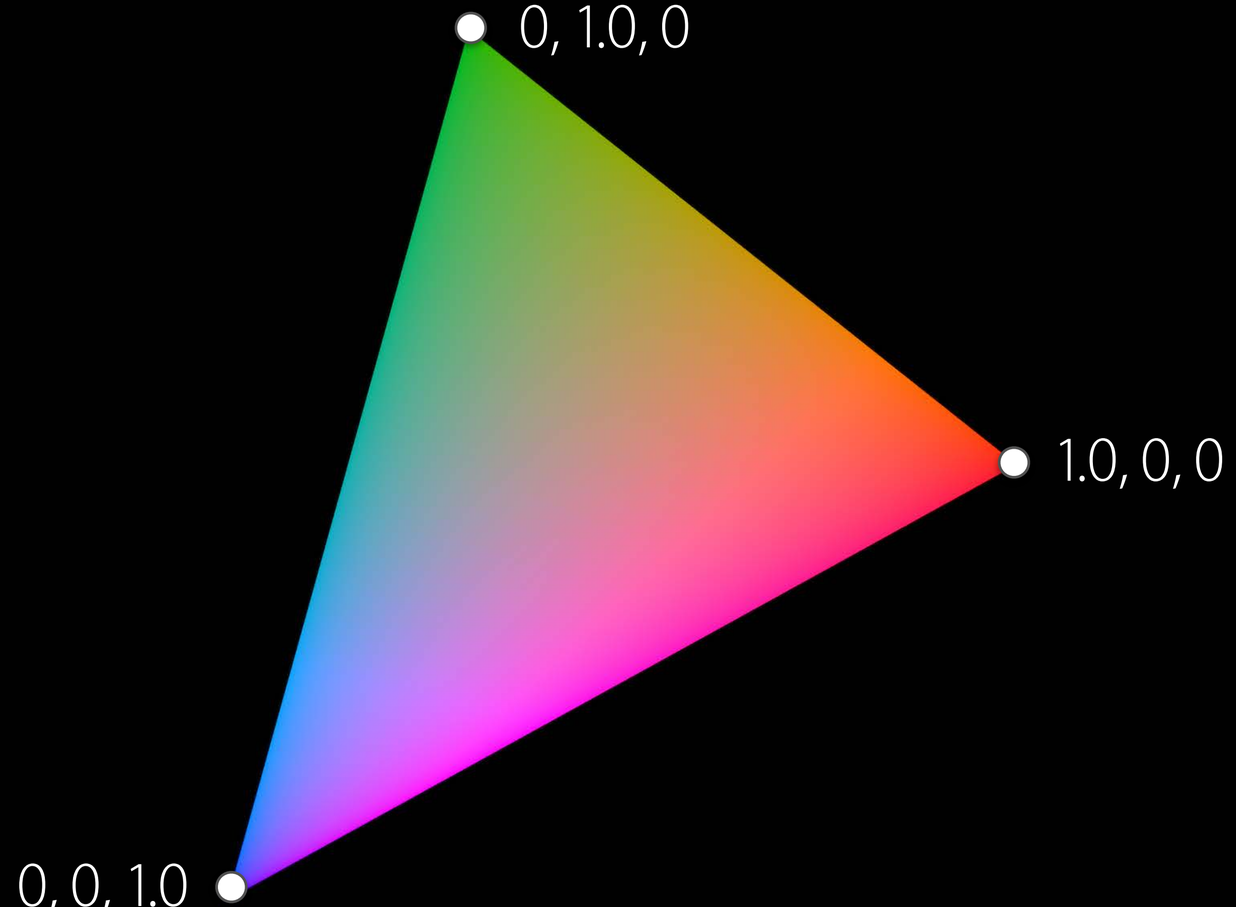
# Wide Gamut

## Extended range sRGB



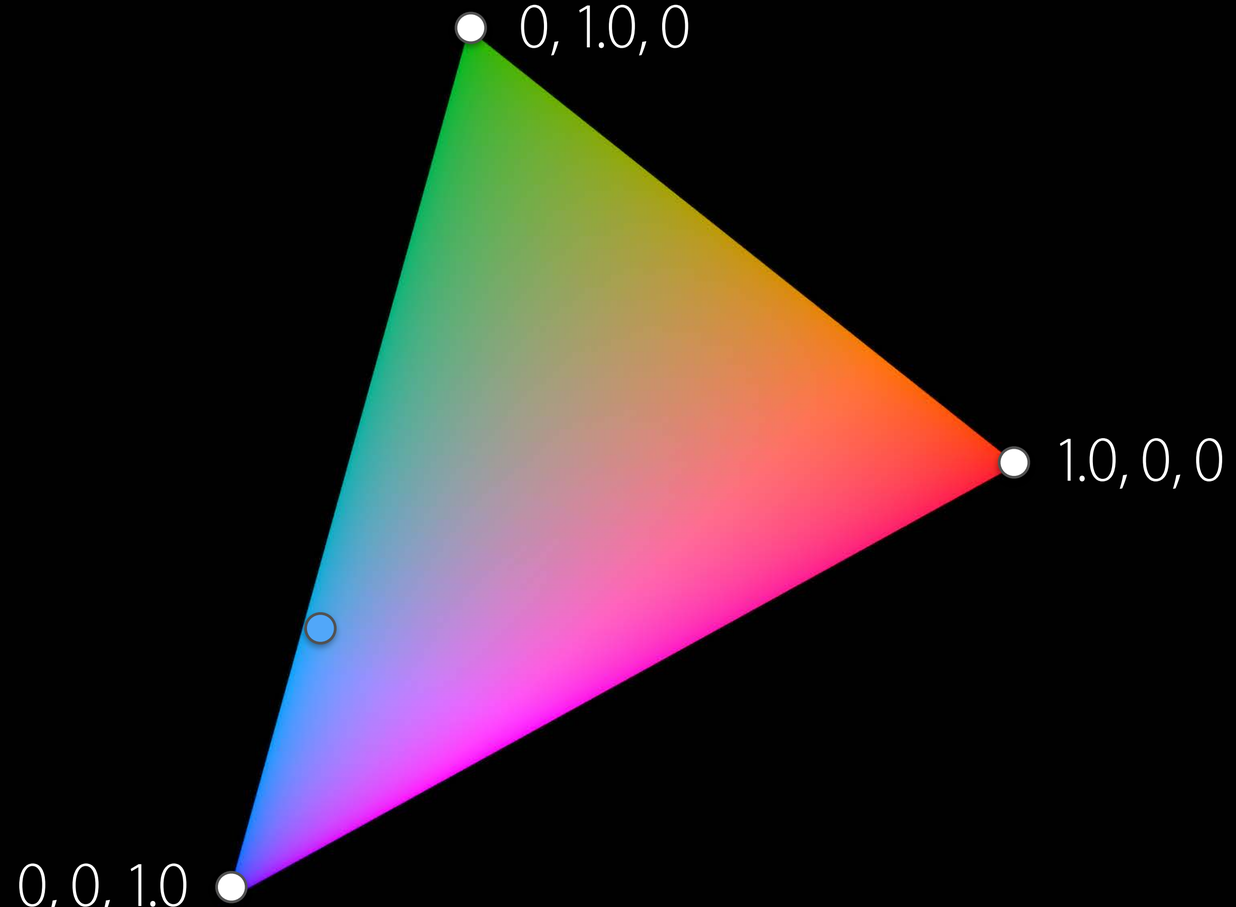
# Wide Gamut

## Extended range sRGB



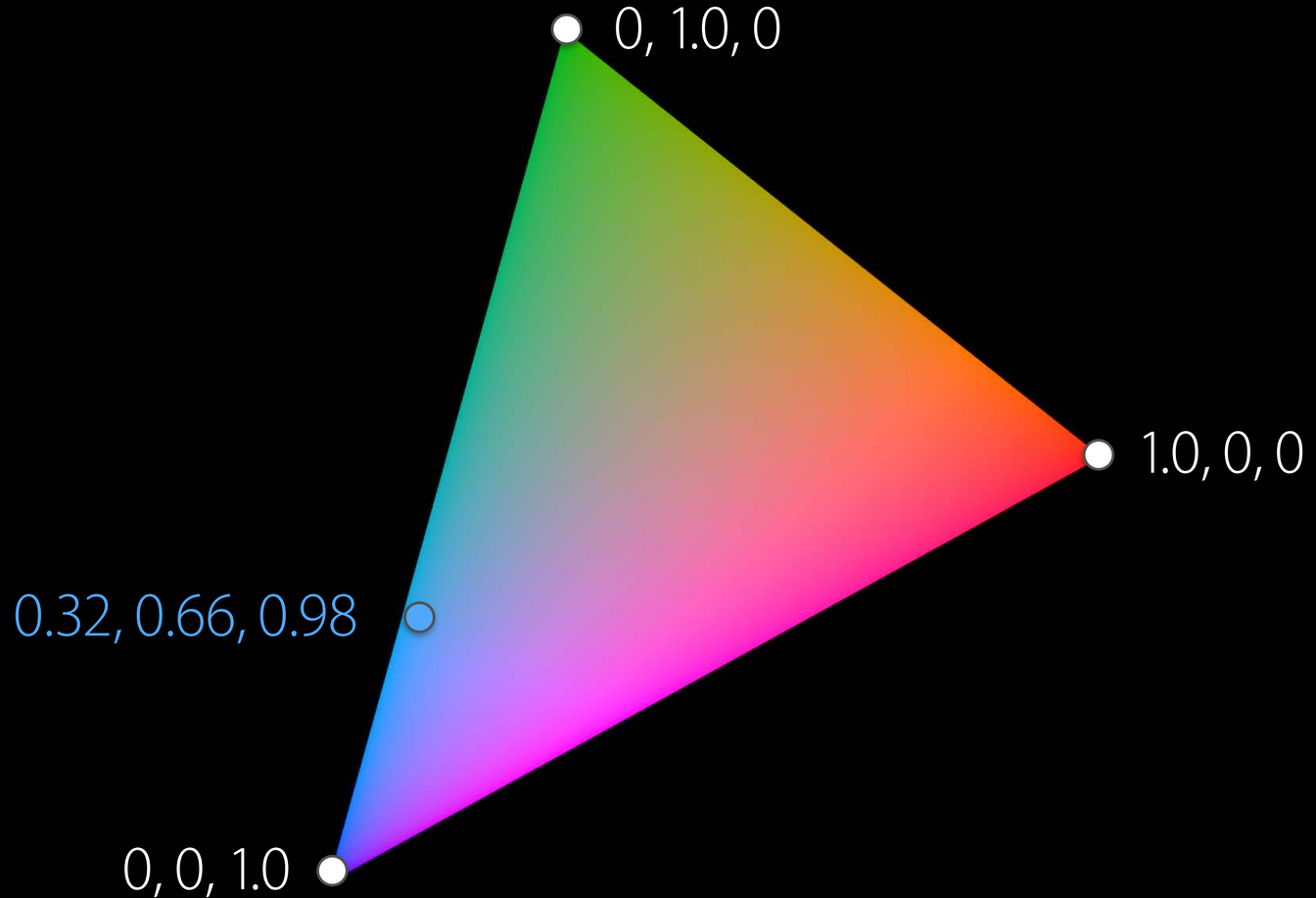
# Wide Gamut

## Extended range sRGB



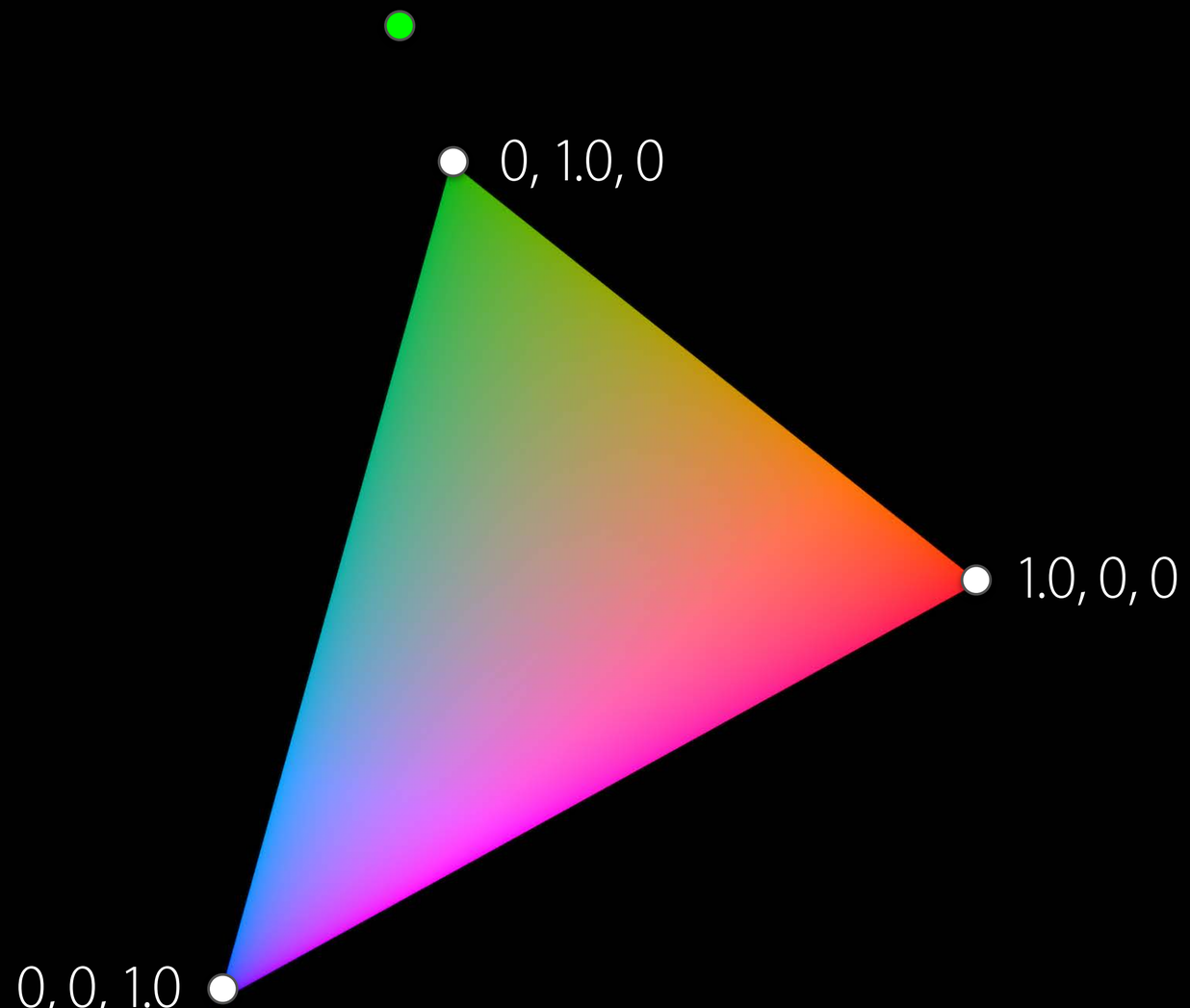
# Wide Gamut

## Extended range sRGB



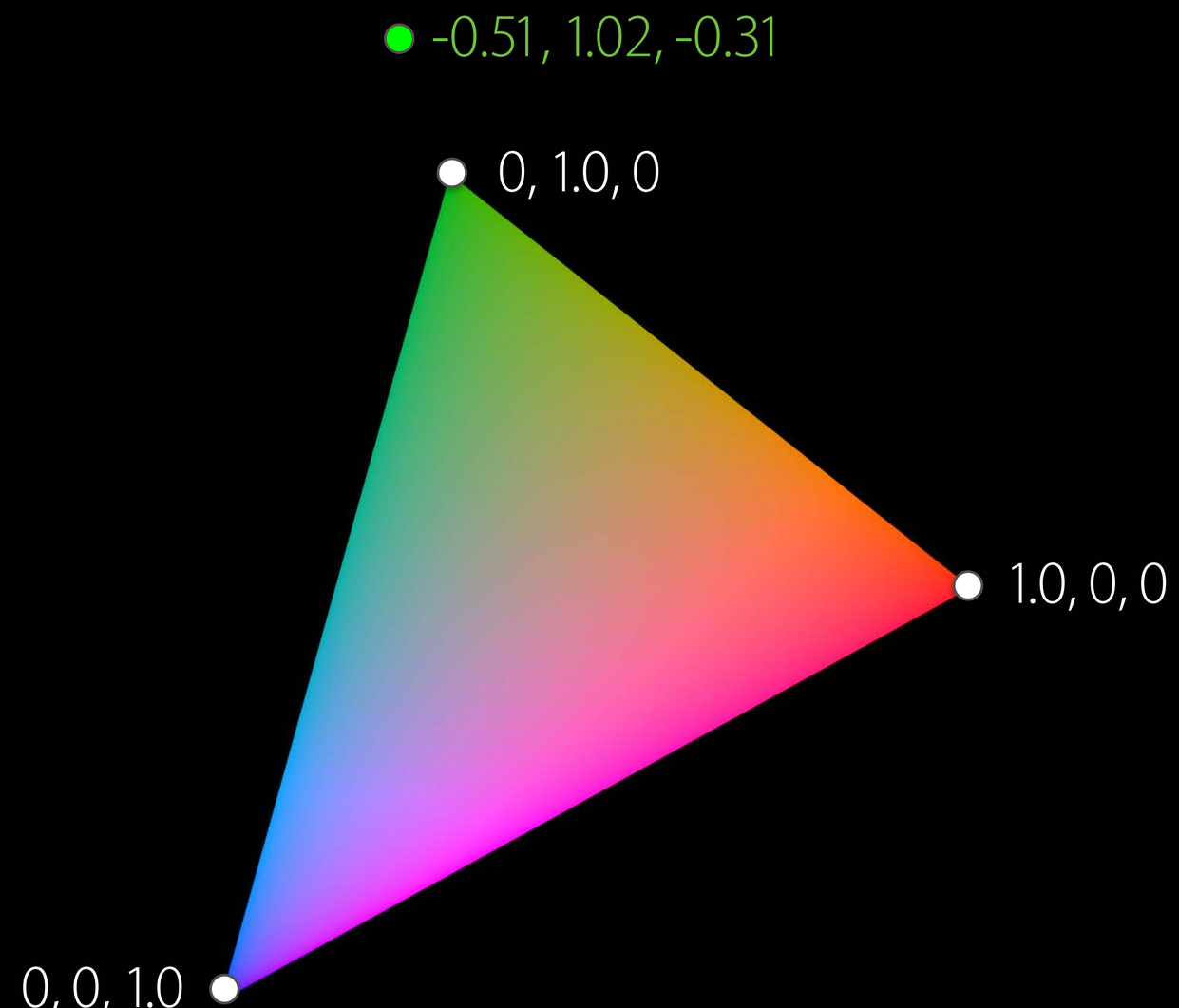
# Wide Gamut

## Extended range sRGB



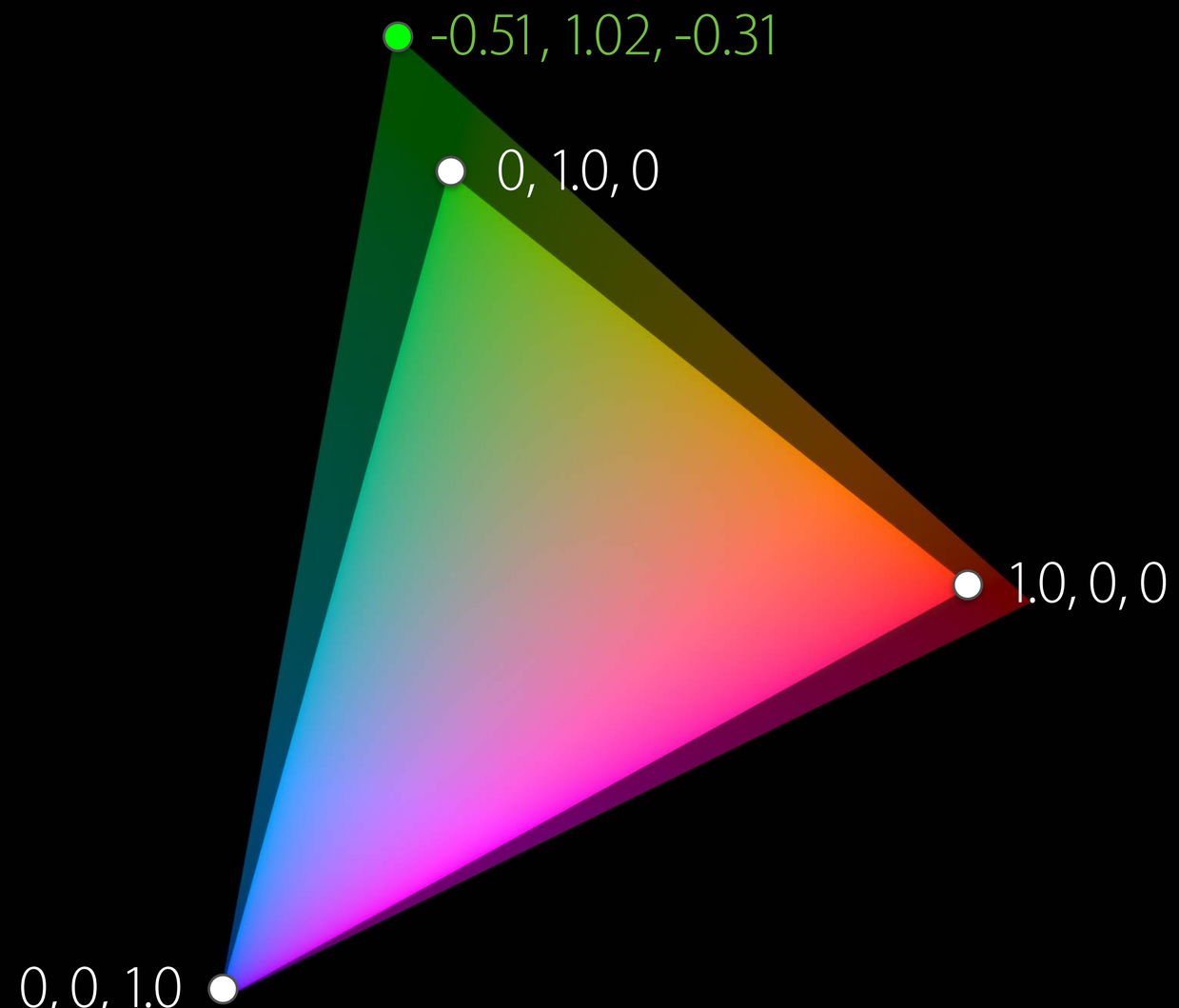
# Wide Gamut

## Extended range sRGB



# Wide Gamut

## Extended range sRGB



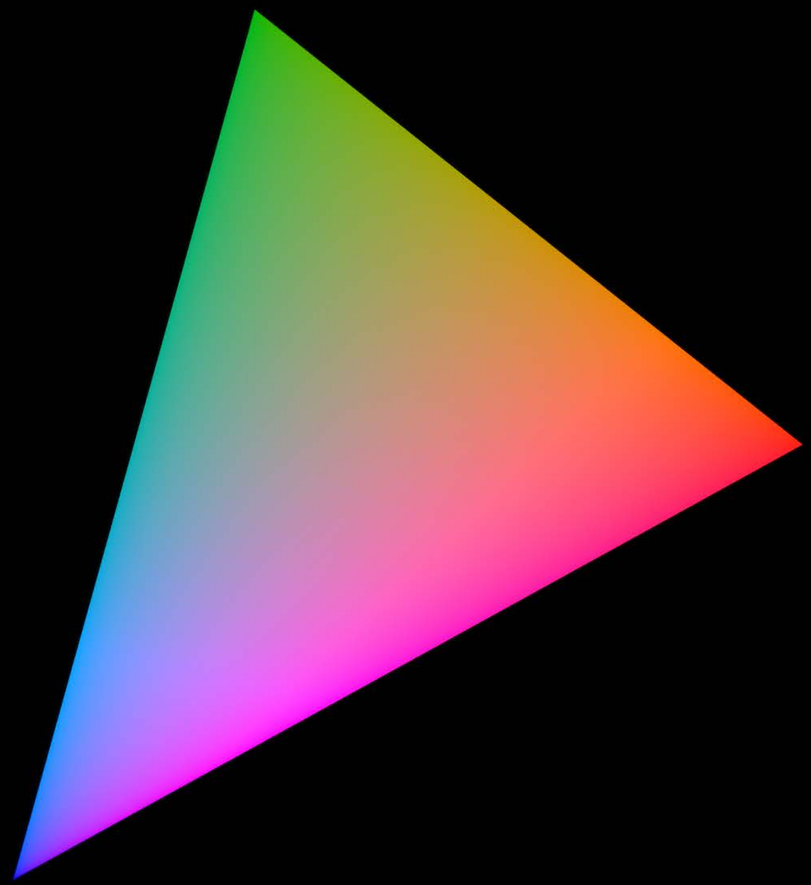
```
// Wide Color API: Extended Range sRGB

extension NSColorSpace {
    public class func extendedSRGB() -> NSColorSpace
}

extension NSColor/UIColor {
    public init(white: CGFloat, alpha: CGFloat)
    public init(red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)
    public init(hue: CGFloat, saturation: CGFloat, brightness: CGFloat, alpha: CGFloat)
}
```

```
// Wide Color API: Extended Range sRGB

extension NSColorSpace {
    public class func extendedSRGB() -> NSColorSpace
}

extension NSColor/UIColor {
    public init(white: CGFloat, alpha: CGFloat)
    public init(red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)
    public init(hue: CGFloat, saturation: CGFloat, brightness: CGFloat, alpha: CGFloat)
}
```

```
// Wide Color API: Extended Range sRGB

extension NSColorSpace {
    public class func extendedSRGB() -> NSColorSpace
}

extension NSColor/UIColor {
    public init(white: CGFloat, alpha: CGFloat)
    public init(red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)
    public init(hue: CGFloat, saturation: CGFloat, brightness: CGFloat, alpha: CGFloat)
}
```

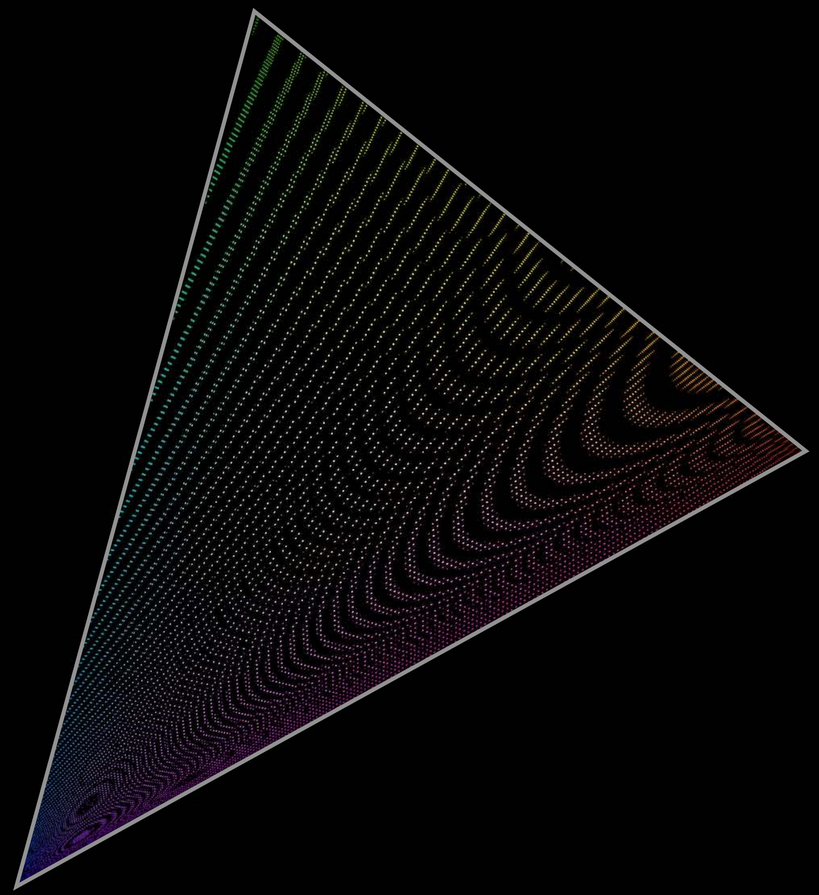
# Color Depth

## sRGB



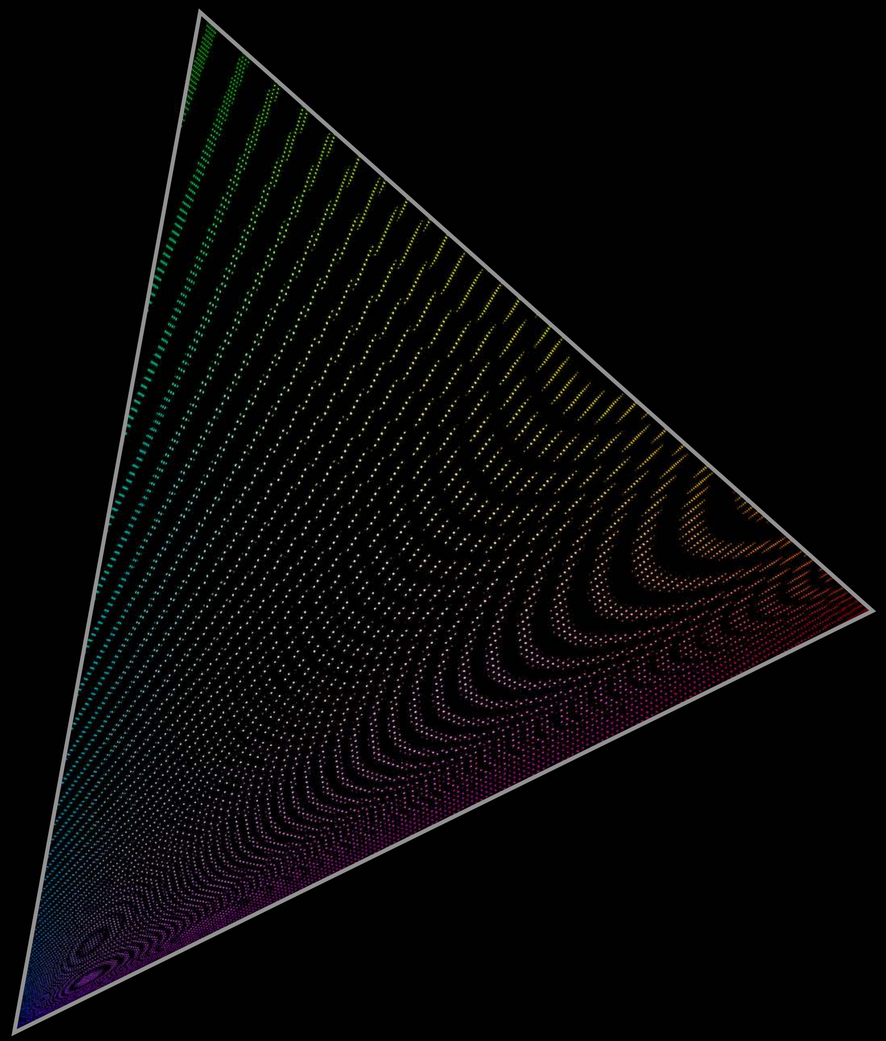
# Color Depth

## sRGB (8 bpc)



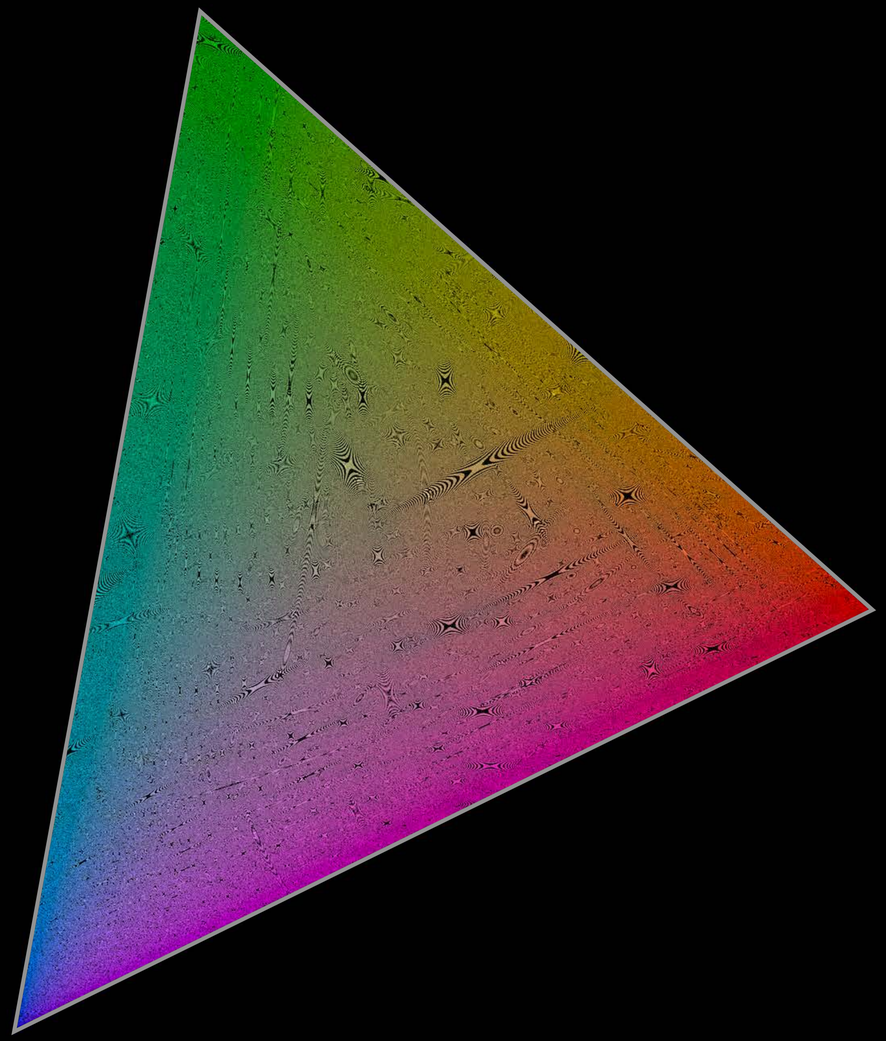
# Color Depth

## P3 (8 bpc)

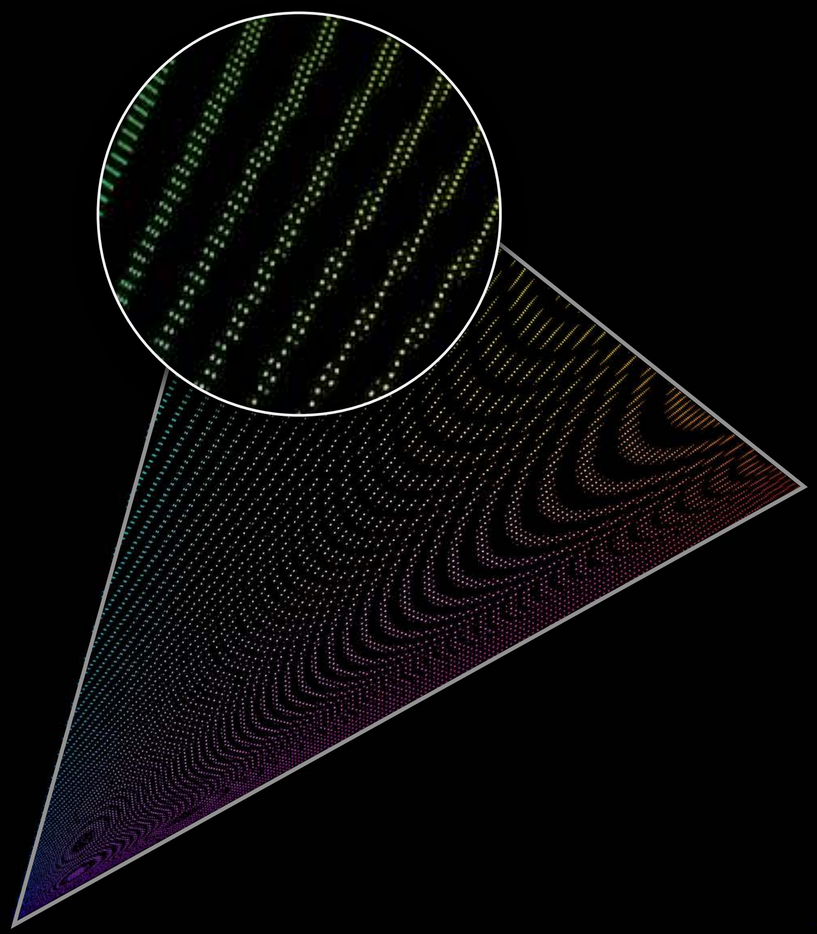


# Color Depth

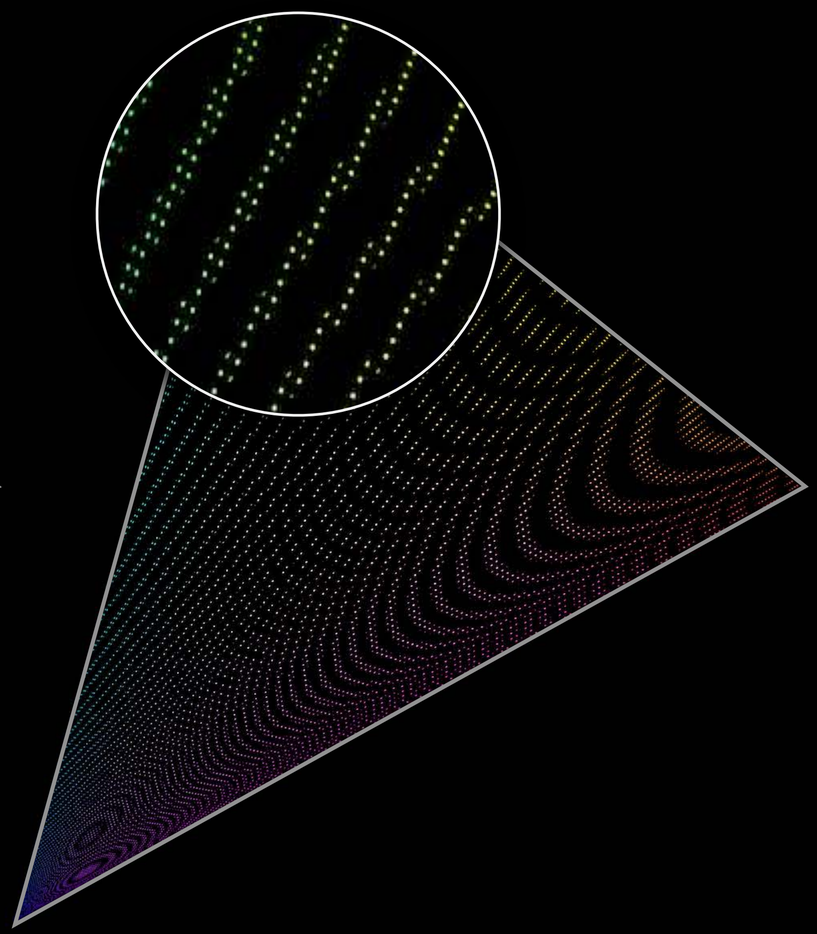
## P3 (16 bpc)



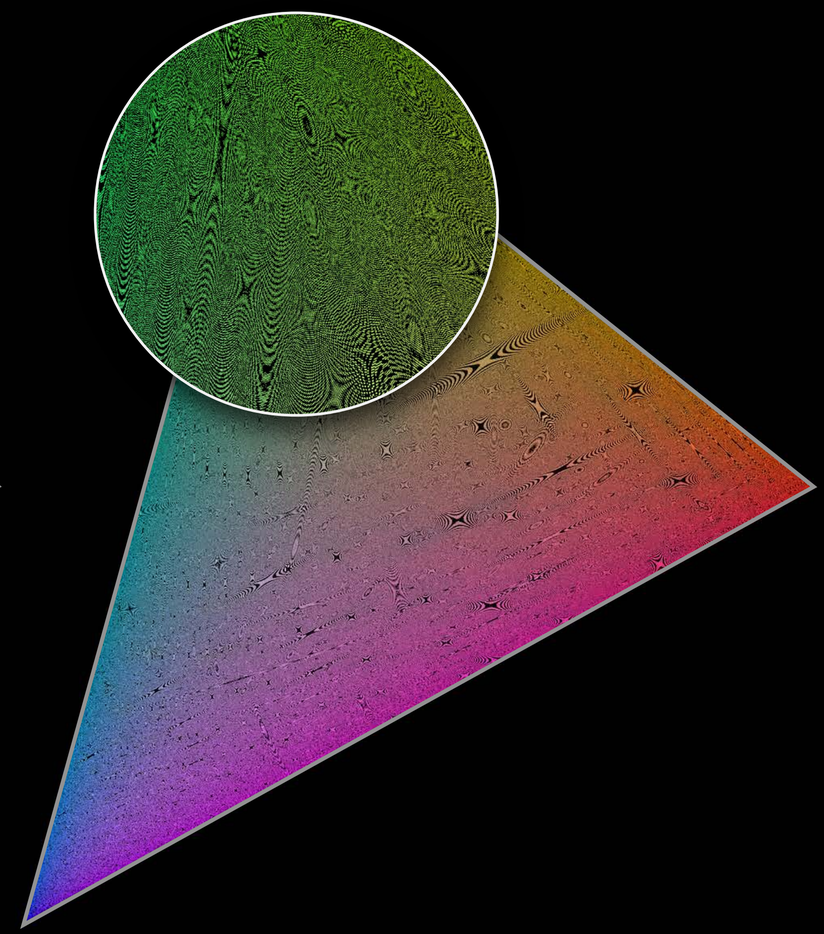
# Color Depth



8-bit sRGB



8-bit P3



16-bit P3

# Deep Color API

# Deep Color API

NSWindow

# Deep Color API

NSWindow

- Automatically deep on wide gamut displays

# Deep Color API

NSWindow

- Automatically deep on wide gamut displays
- depthLimit can be set to 8-bit or 16-bit

# Deep Color API

NSWindow

- Automatically deep on wide gamut displays
- depthLimit can be set to 8-bit or 16-bit

Views and layers inherit from their window

# Deep Color API

NSWindow

- Automatically deep on wide gamut displays
- depthLimit can be set to 8-bit or 16-bit

Views and layers inherit from their window

- Not OpenGL view—use pixel format

# Deep Color API

NSWindow

- Automatically deep on wide gamut displays
- depthLimit can be set to 8-bit or 16-bit

Views and layers inherit from their window

- Not OpenGL view—use pixel format

CALayer

# Deep Color API

NSWindow

- Automatically deep on wide gamut displays
- depthLimit can be set to 8-bit or 16-bit

Views and layers inherit from their window

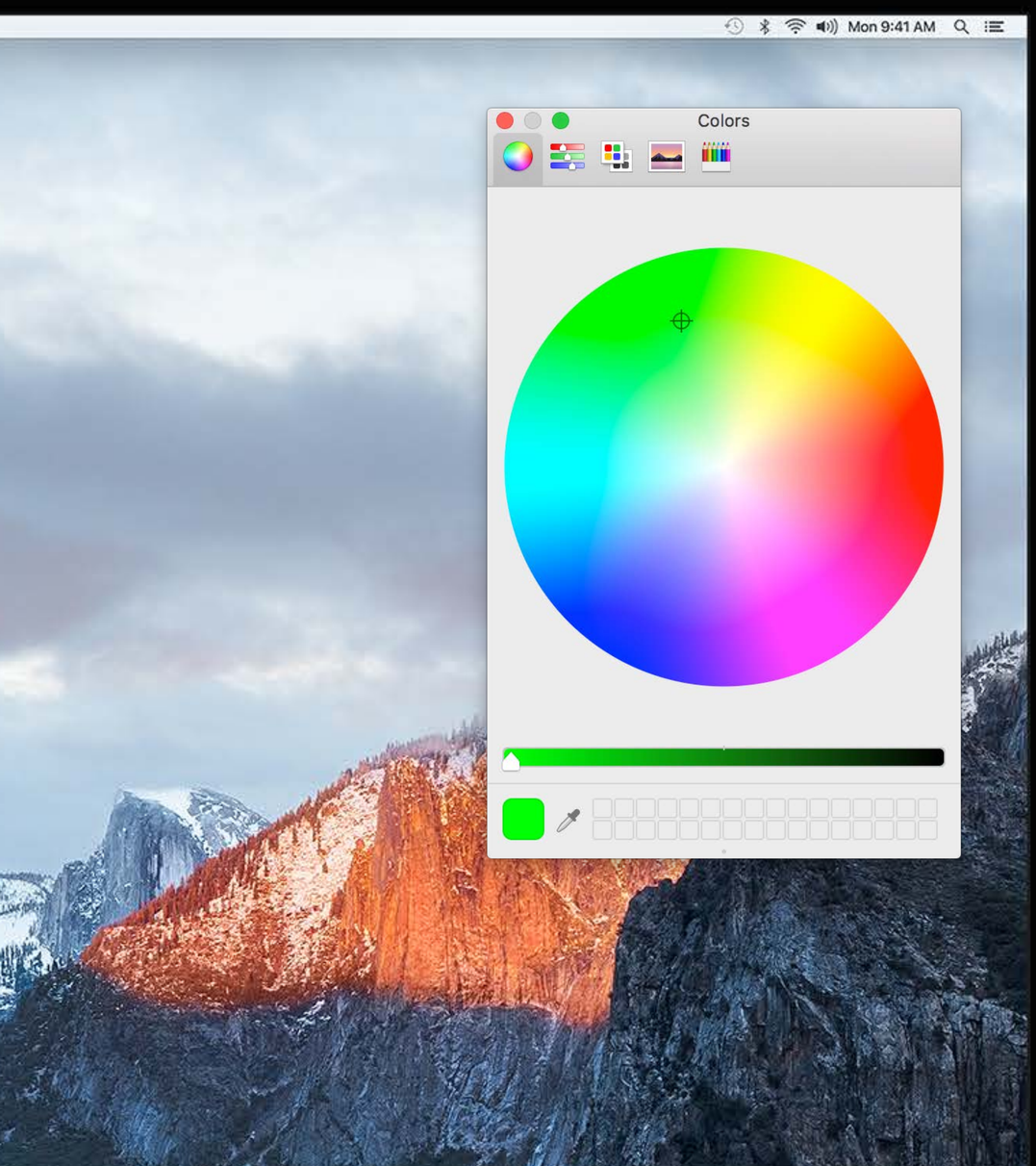
- Not OpenGL view—use pixel format

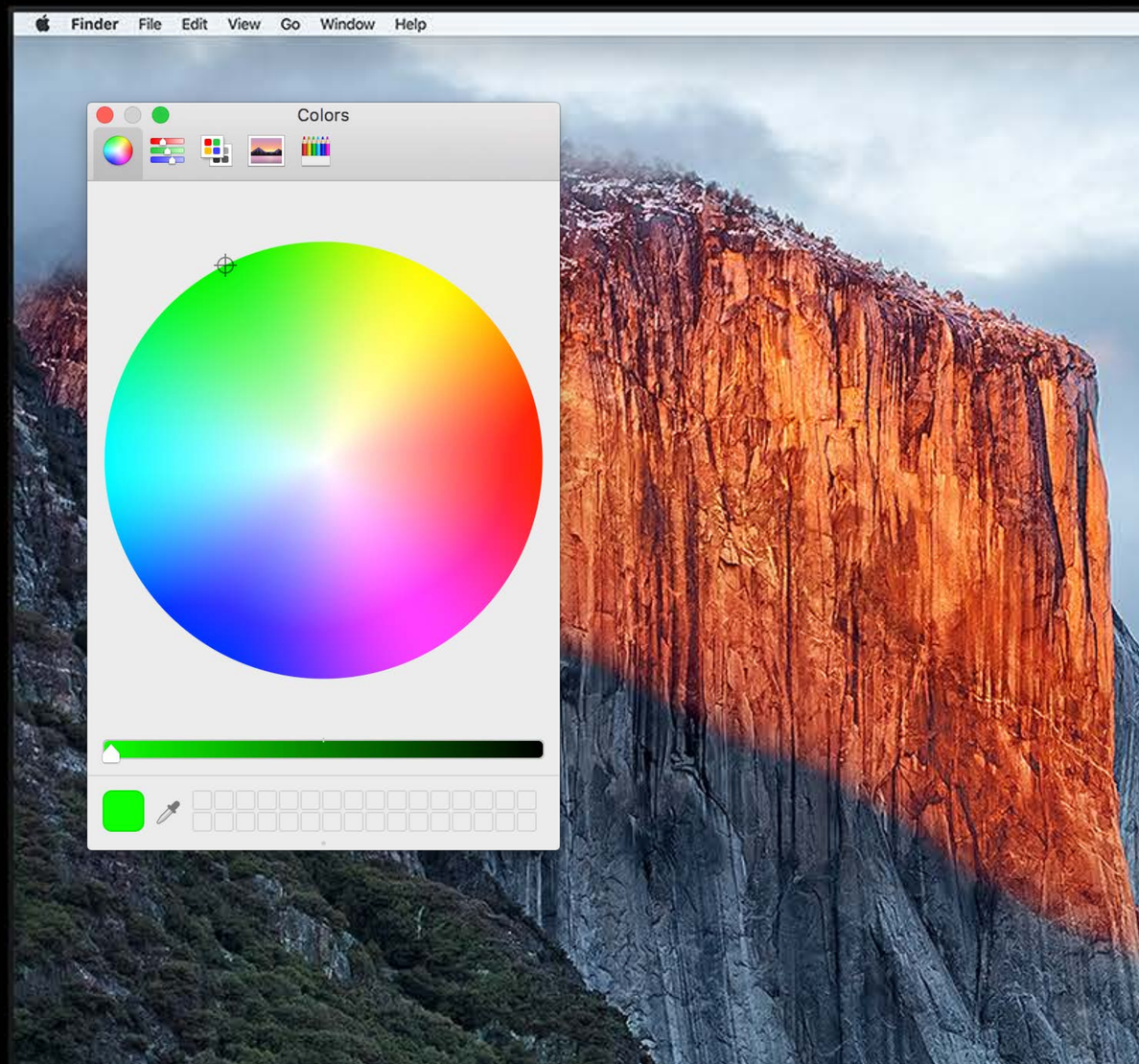
CALayer

- contentsFormat can be set to 8-bit, 16-bit (half-float)

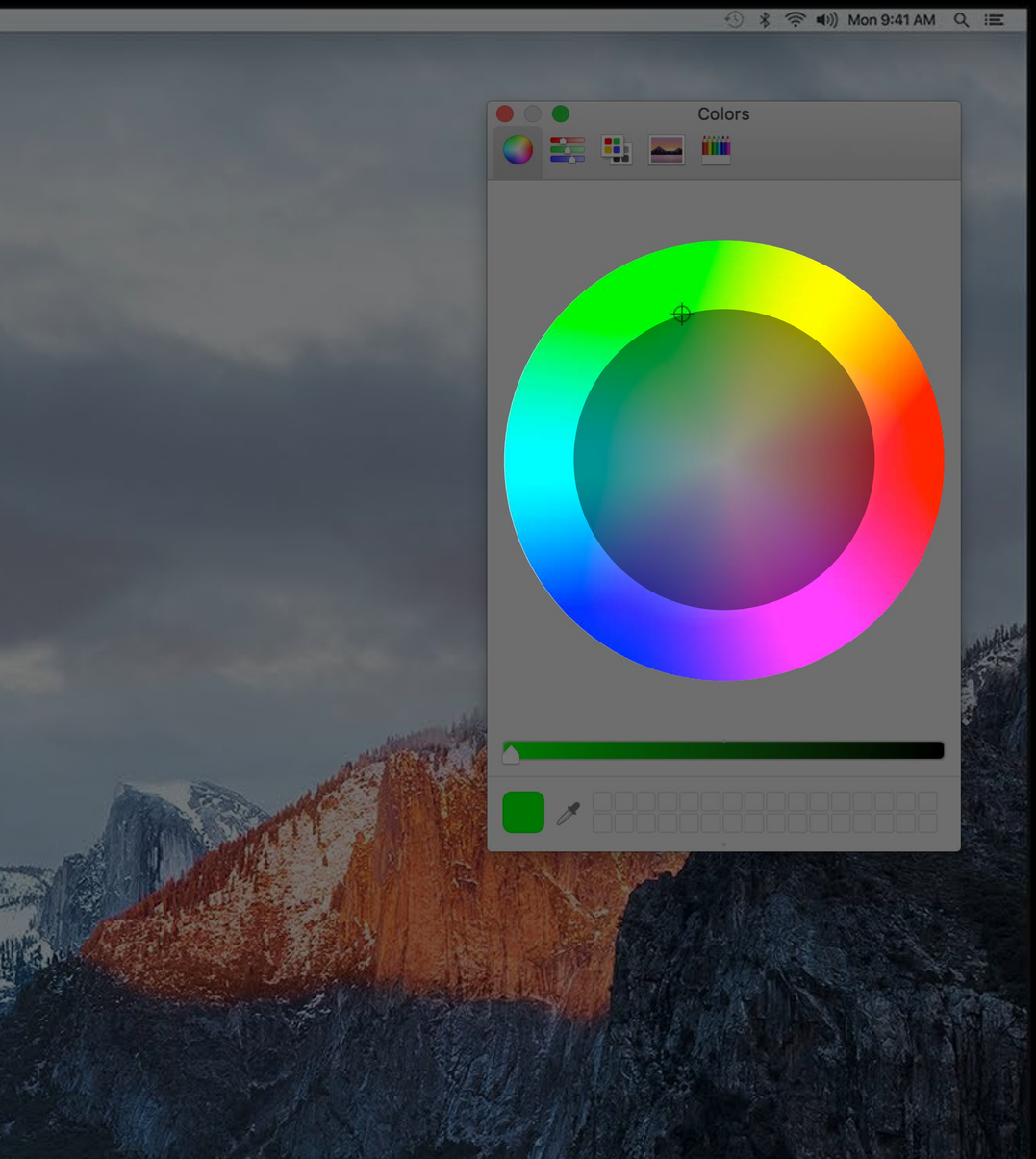
# NSColorPanel

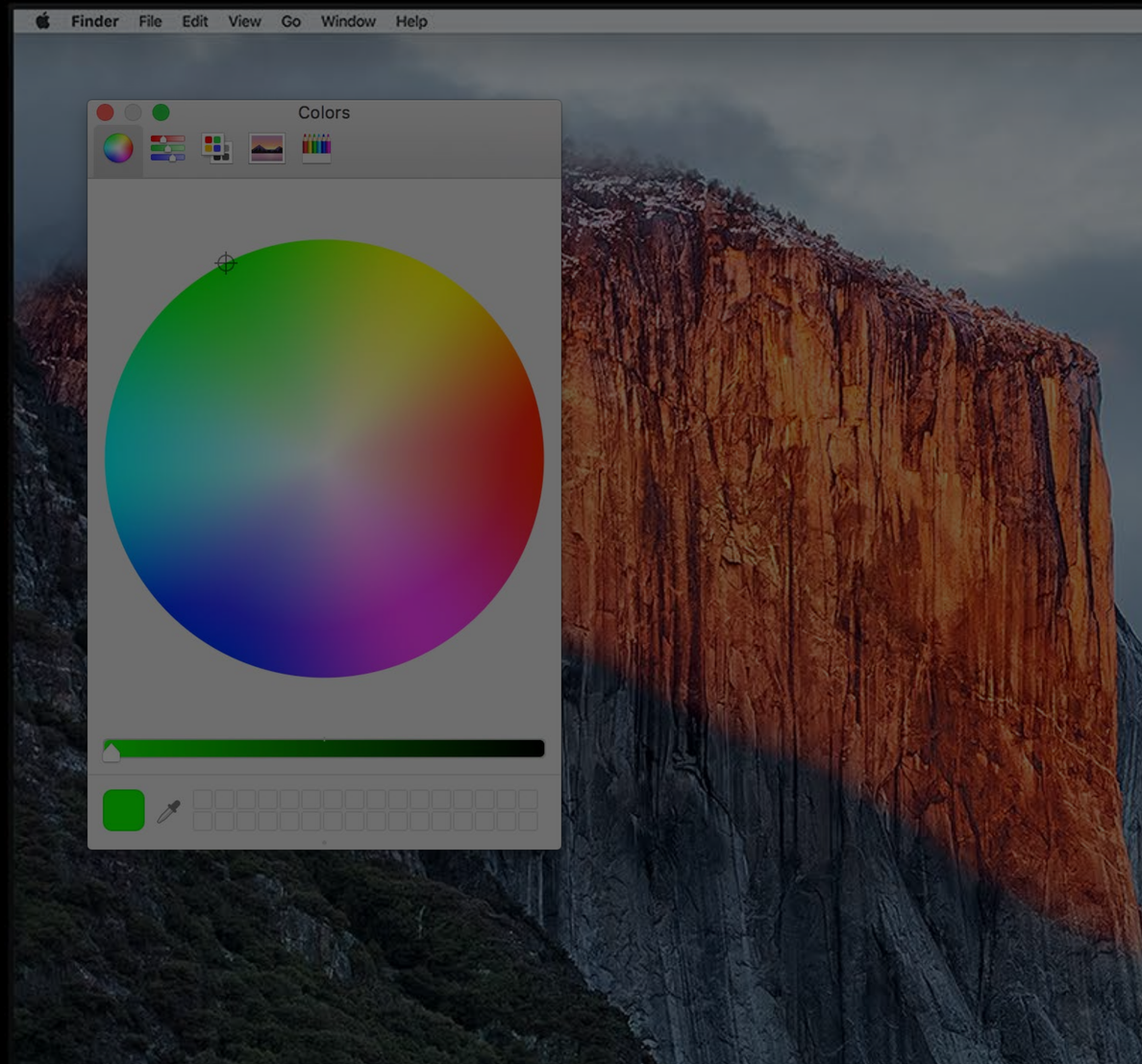

# NSColorPanel



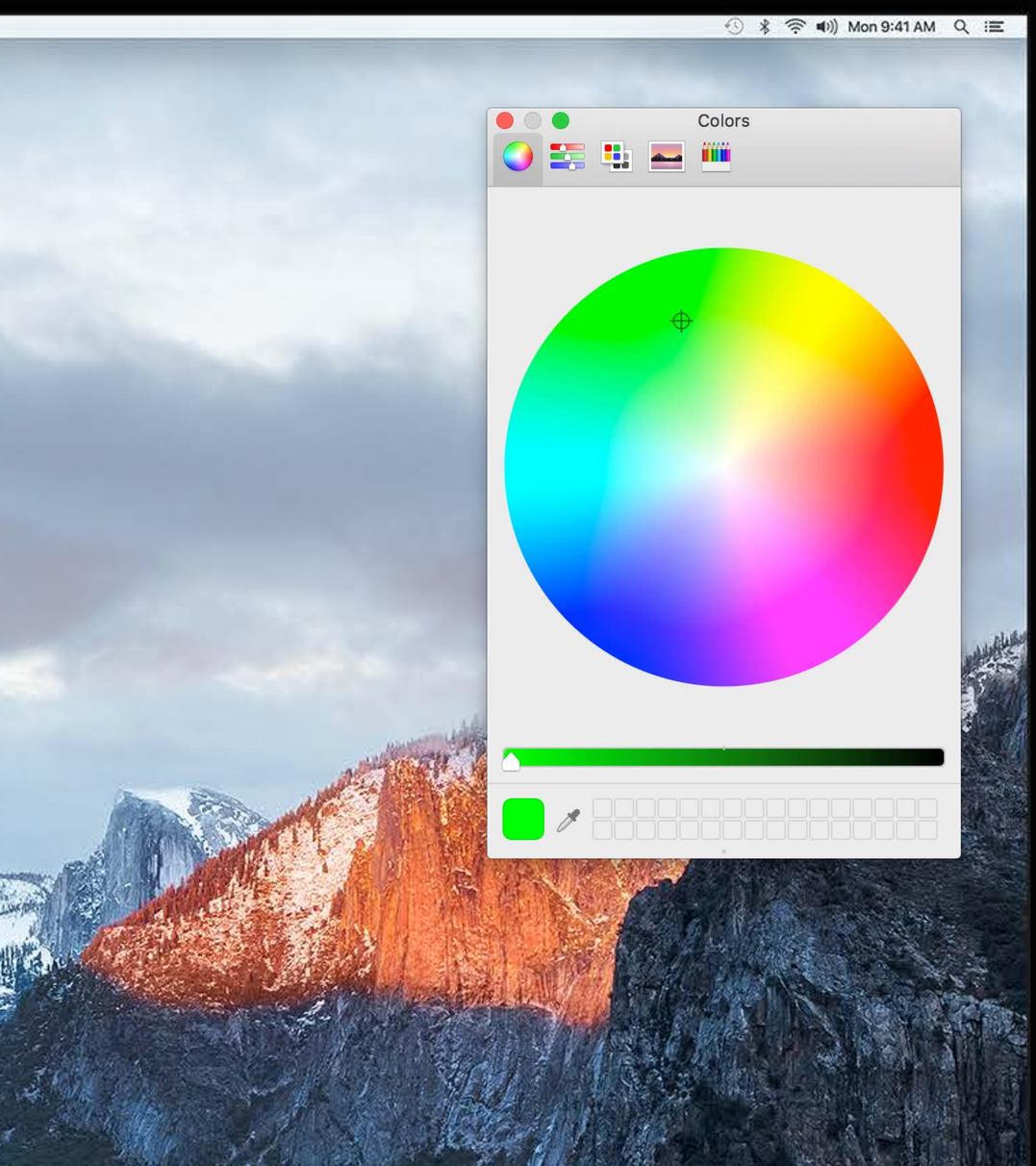



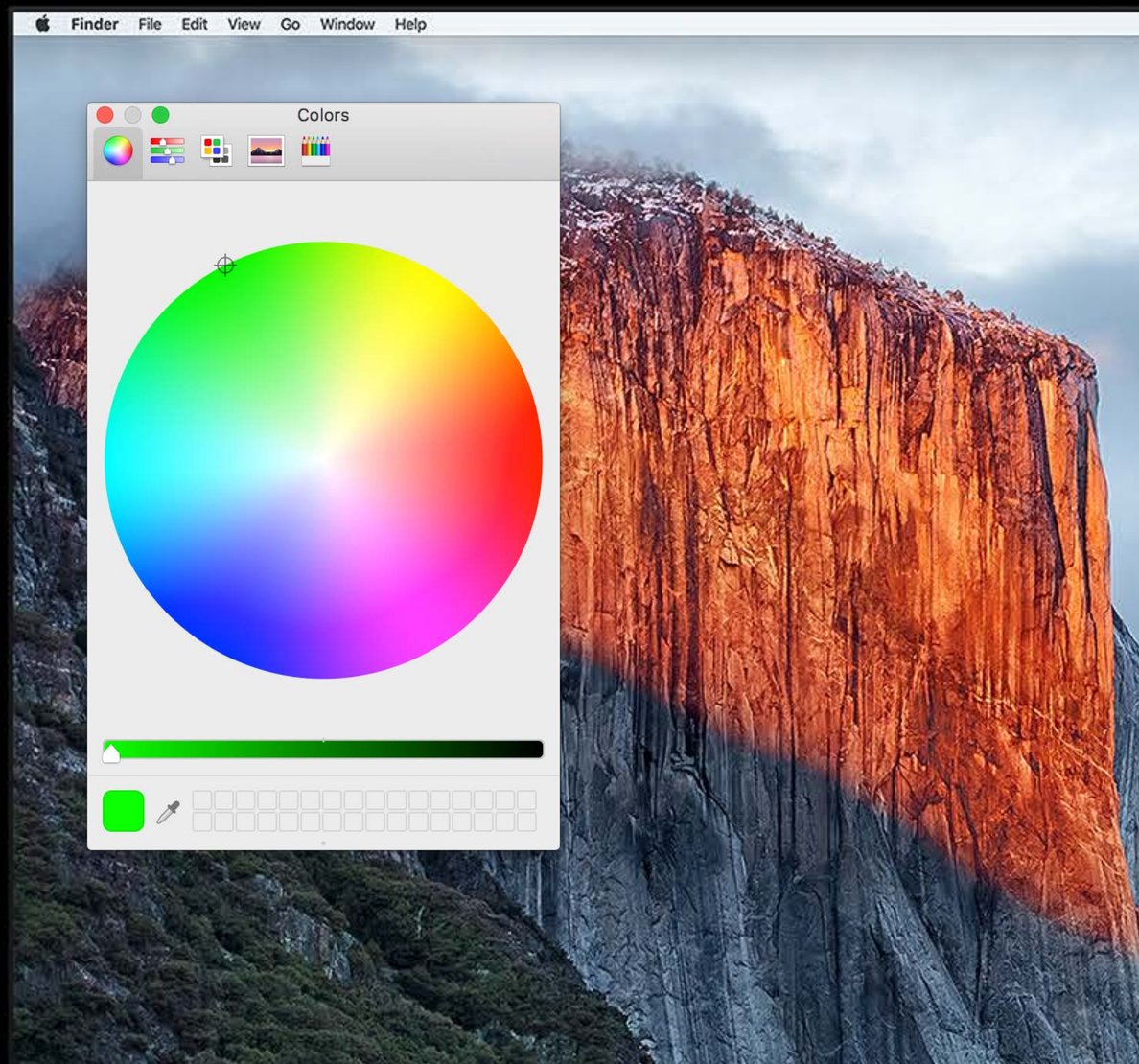
# NSColorPanel



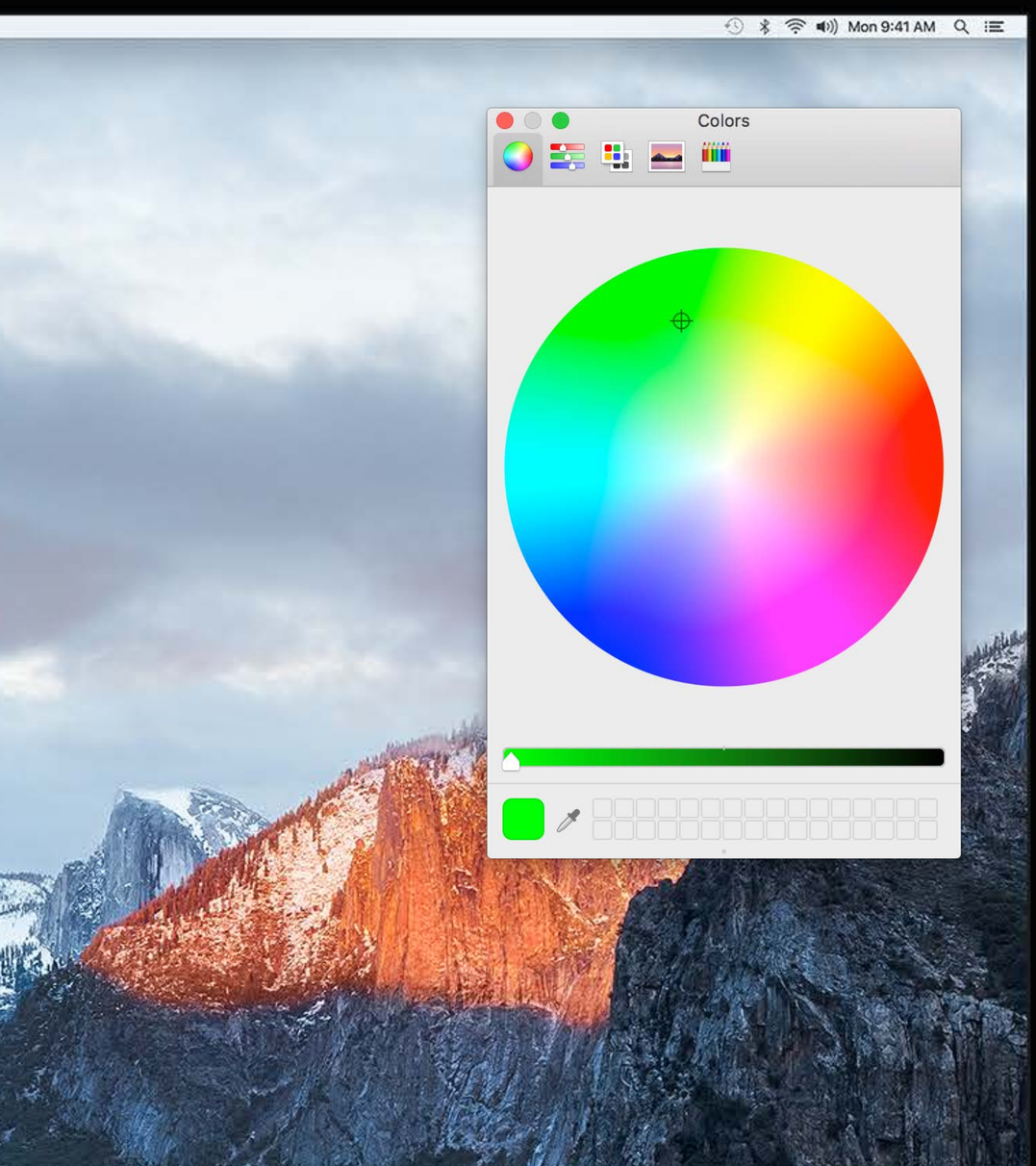



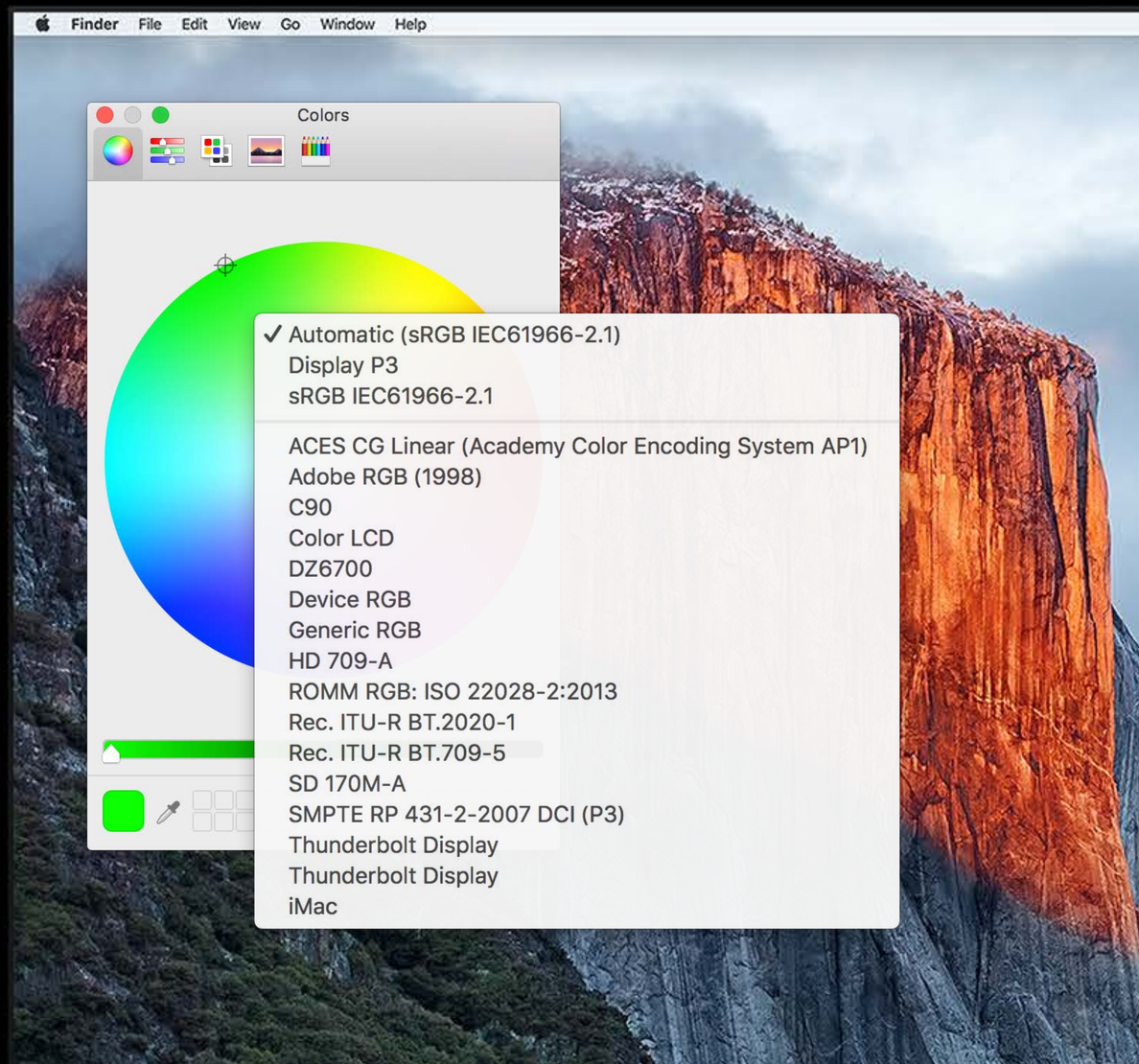
# NSColorPanel



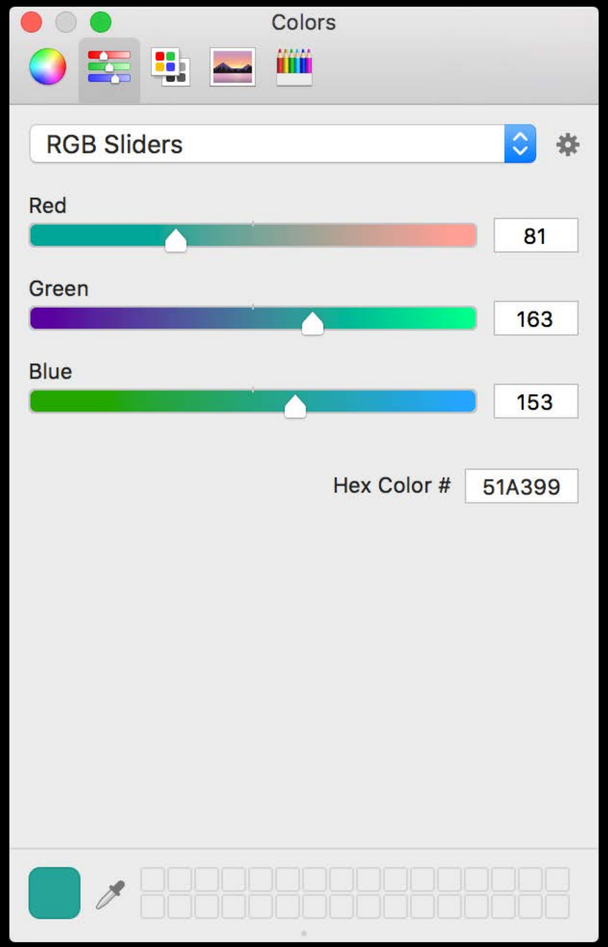



# NSColorPanel

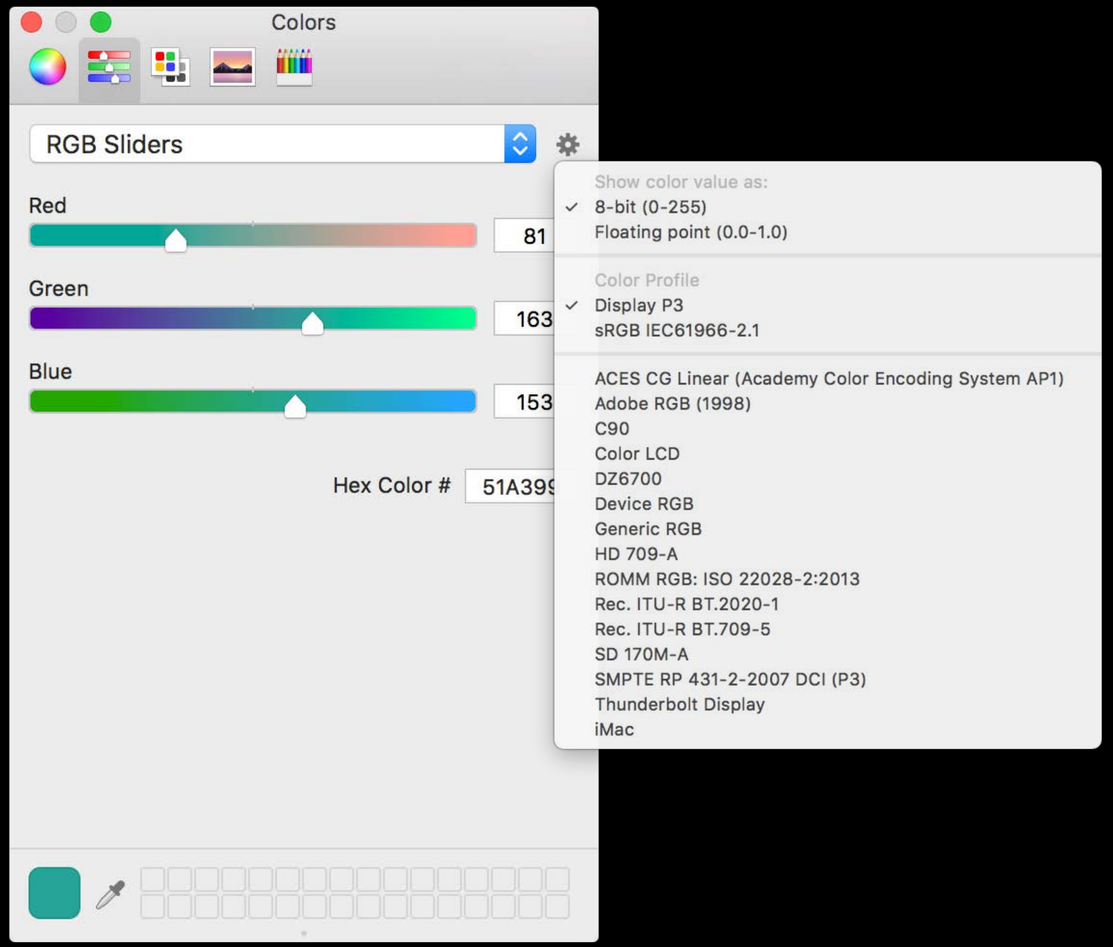


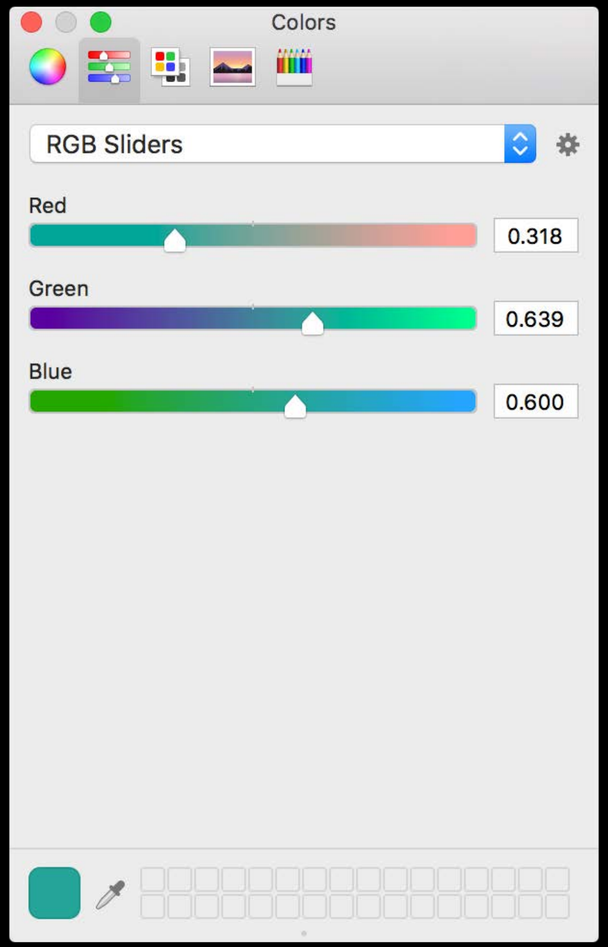
# NSColorPanel



# NSColorPanel



# NSColorPanel



# Wide Gamut Colors

Drawing wide colors

Asset catalog

RTF color space support

WebKit support

Working with Wide Color

Mission

Thursday 1:40PM

# Status Item Enhancements

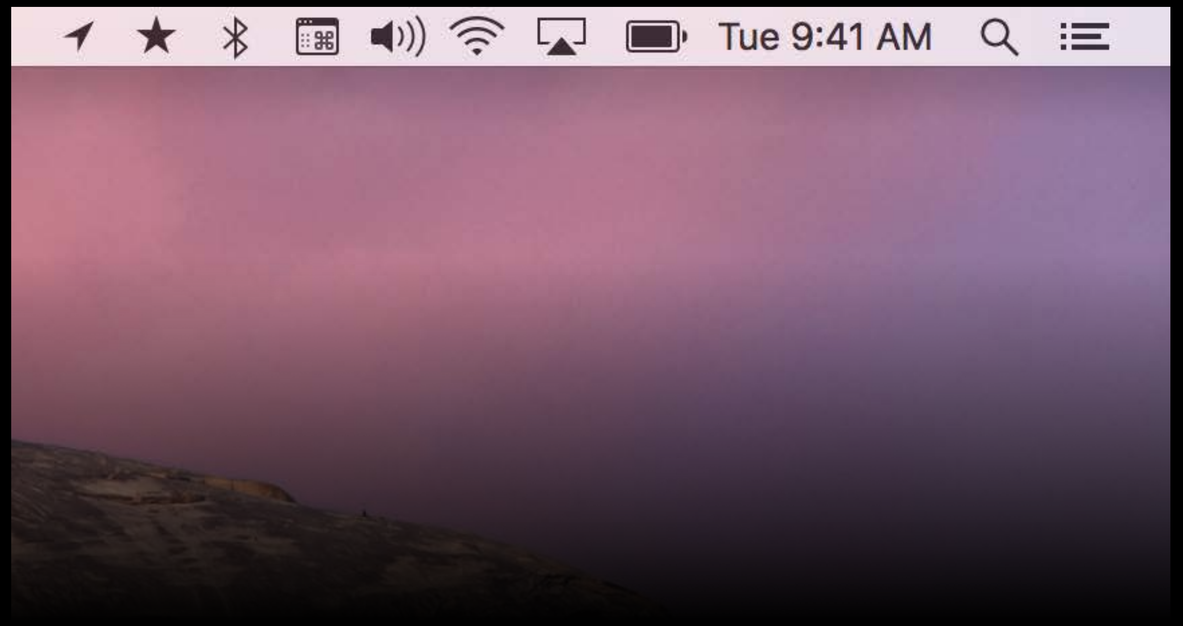
Reordering

Keyboard navigation

User hiding and removal

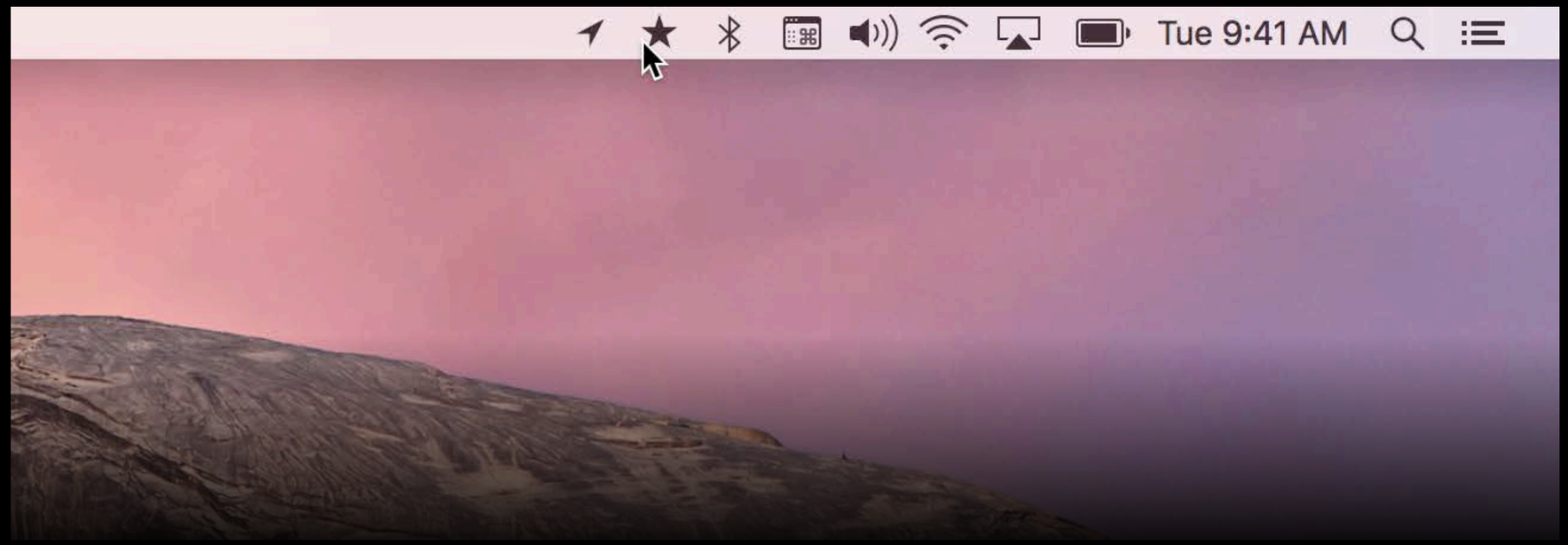
- Status bar apps

Autosave



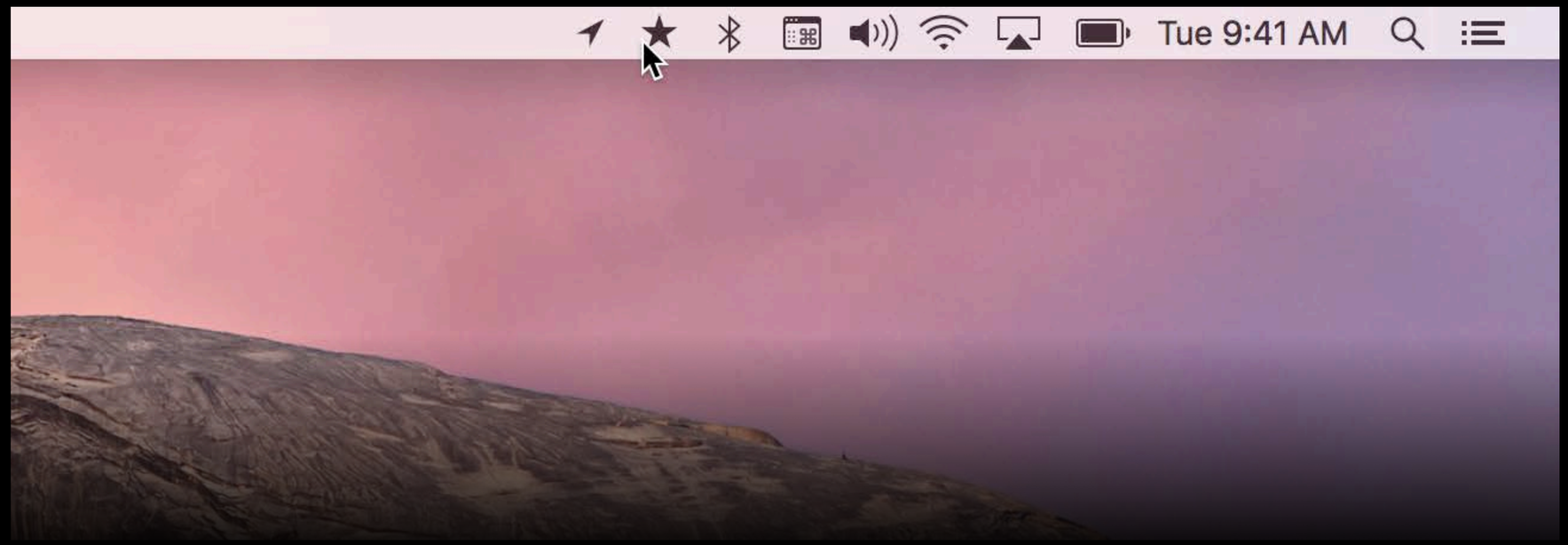
# Status Item Enhancements

## Reordering and keyboard navigation



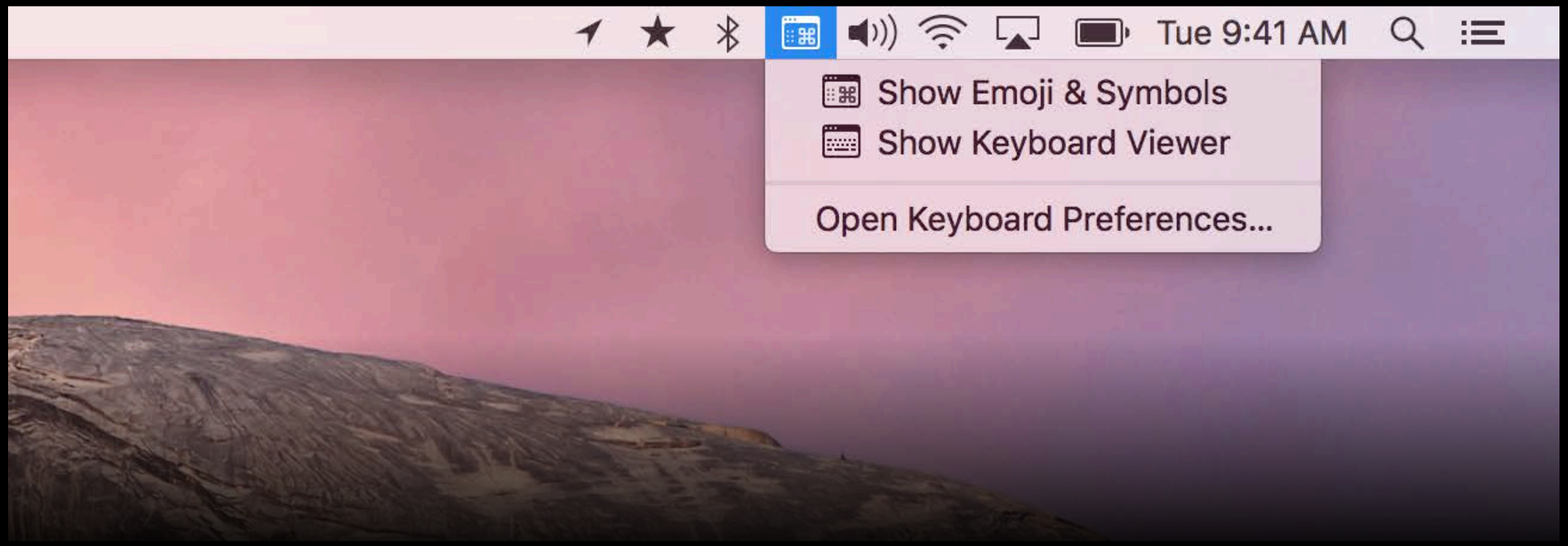
# Status Item Enhancements

## Reordering and keyboard navigation



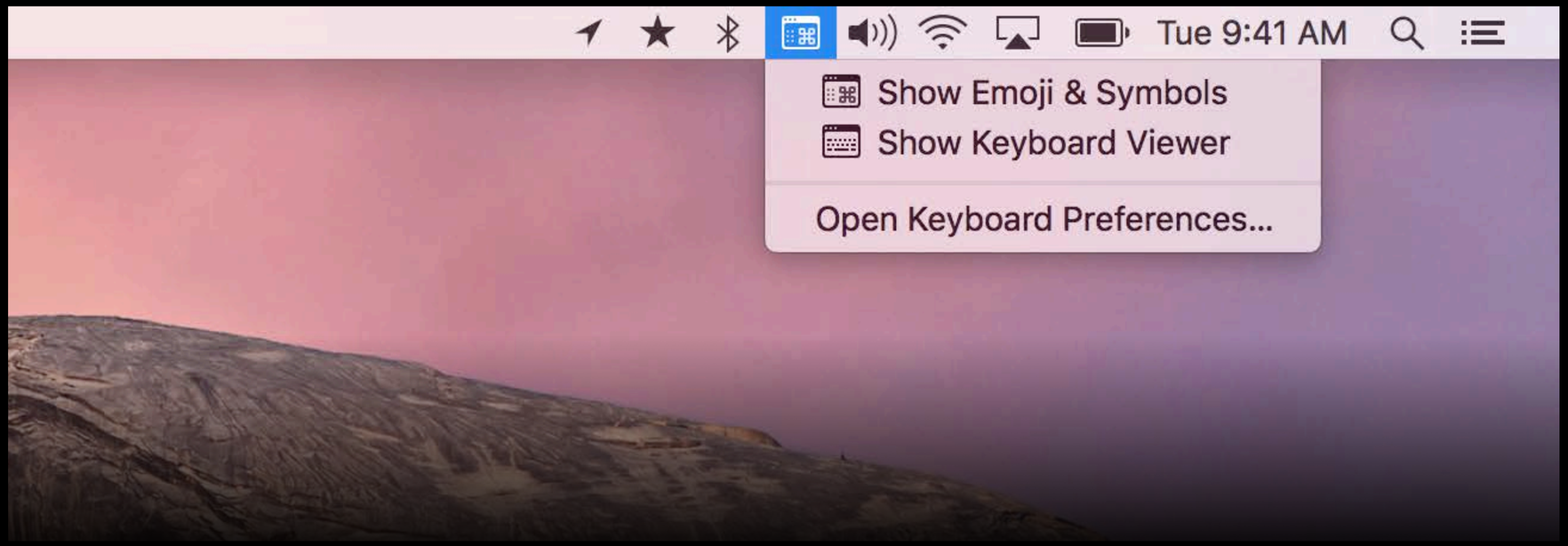
# Status Item Enhancements

## Reordering and keyboard navigation



# Status Item Enhancements

## Reordering and keyboard navigation



# Status Item Enhancements

## Reordering and keyboard navigation

# Status Item Enhancements

## Reordering and keyboard navigation

Applies to all existing status items

# Status Item Enhancements

## Reordering and keyboard navigation

Applies to all existing status items

Command-click and drag to reorder

# Status Item Enhancements

## Reordering and keyboard navigation

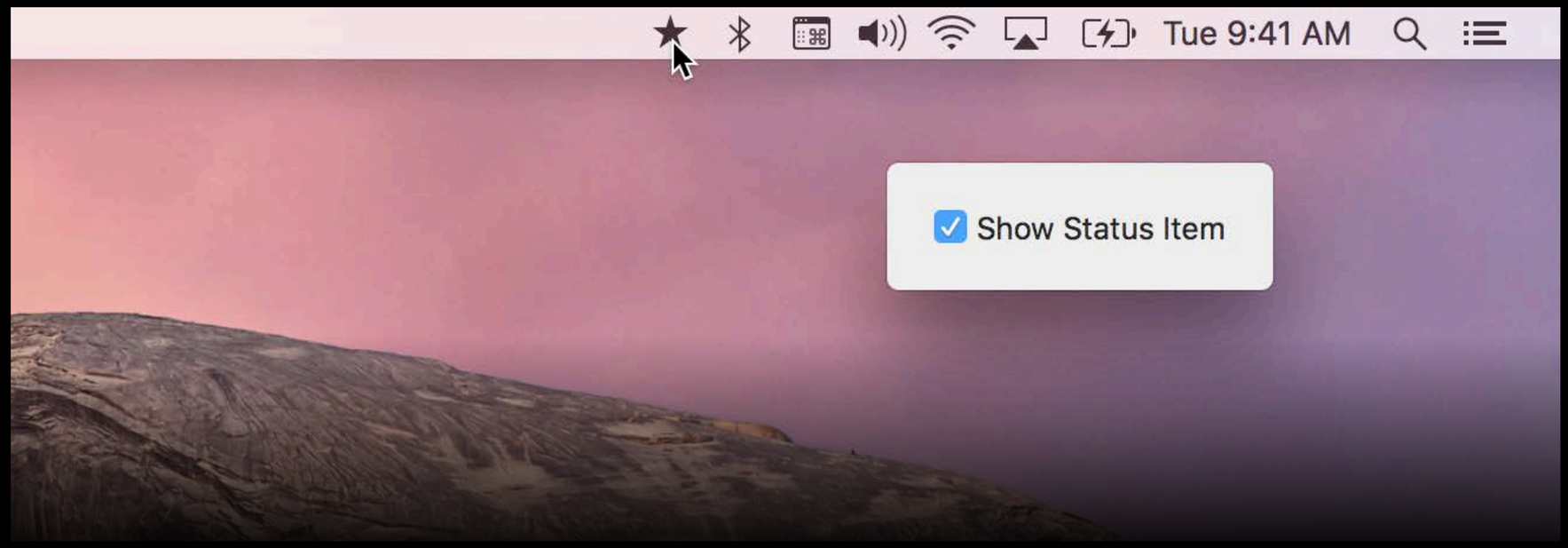
Applies to all existing status items

Command-click and drag to reorder

Cross-item keyboard navigation for items with a menu

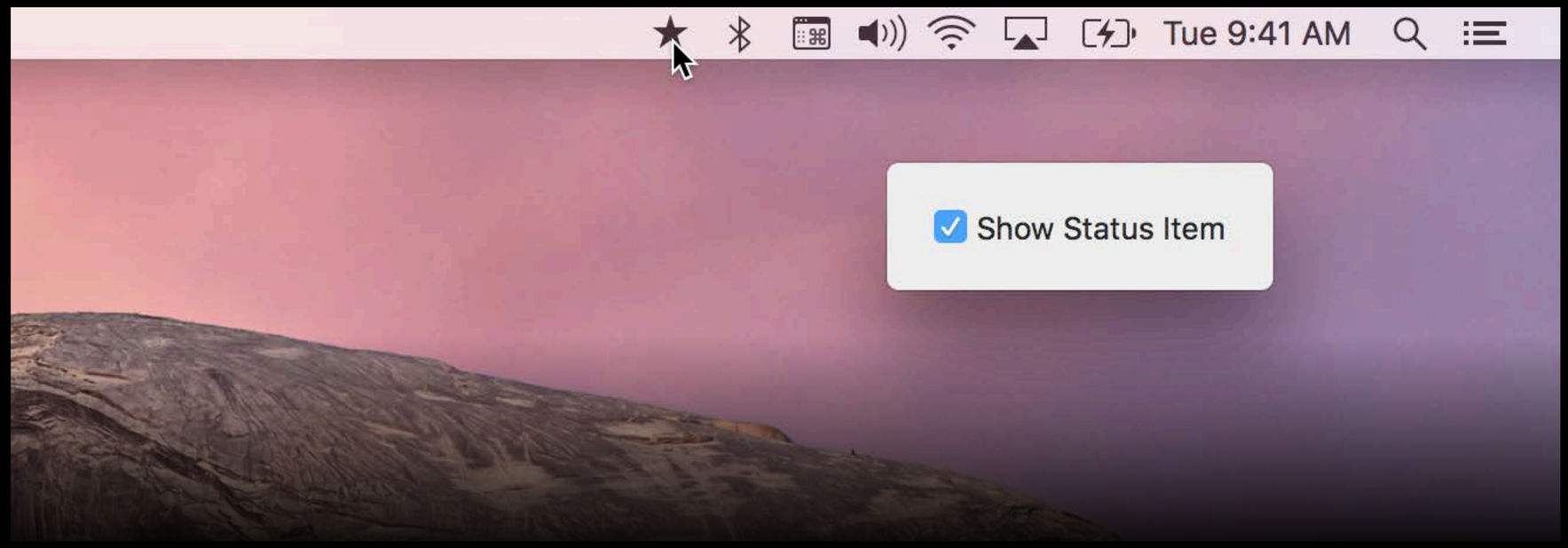
# Status Item Enhancements

## Hiding and removal



# Status Item Enhancements

## Hiding and removal



# Status Item Enhancements

## Hiding and removal

# Status Item Enhancements

## Hiding and removal

User removal is opt-in

# Status Item Enhancements

## Hiding and removal

User removal is opt-in

```
extension NSStatusItem {
    public var behavior: NSStatusItemBehavior

}
public struct NSStatusItemBehavior : OptionSet {
    public static var removalAllowed: NSStatusItemBehavior { get }

}
```

# Status Item Enhancements

## Hiding and removal

User removal is opt-in

Can be programmatically read/set

```
extension NSStatusItem {
    public var behavior: NSStatusItemBehavior
    public var isVisible: Bool
}
public struct NSStatusItemBehavior : OptionSet {
    public static var removalAllowed: NSStatusItemBehavior { get }

}
```

# Status Item Enhancements

## Hiding and removal

User removal is opt-in

Can be programmatically read/set

Status bar apps can be set up to automatically quit on removal

```
extension NSStatusItem {
    public var behavior: NSStatusItemBehavior
    public var isVisible: Bool
}
public struct NSStatusItemBehavior : OptionSet {
    public static var removalAllowed: NSStatusItemBehavior { get }
    public static var terminationOnRemoval: NSStatusItemBehavior { get }
}
```

# Status Item Enhancements

## Autosave

# Status Item Enhancements

## Autosave

Restores location of status item and visible state

# Status Item Enhancements

## Autosave

Restores location of status item and visible state

```
extension NSStatusItem {
    public var autosaveName: String!
}
```

# Status Item Enhancements

## Autosave

Restores location of status item and visible state

Defaults to generated name based on item index (nil resettable)

```
extension NSStatusItem {
    public var autosaveName: String!
}
```

# Status Item Enhancements

## Autosave

Restores location of status item and visible state

Defaults to generated name based on item index (nil resettable)

```
extension NSStatusItem {
    public var autosaveName: String!
}

statusItem.autosaveName = “com.example.StarStatusItem”
```

# Control Constructors

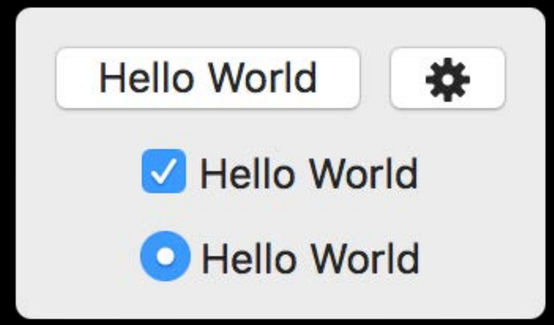
# Control Constructors

Convenience constructors on NSControls

# Control Constructors

Convenience constructors on NSControls

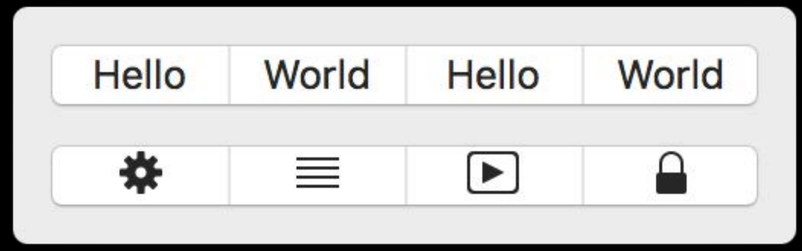
- NSButton:  push, checkbox, radio



# Control Constructors

Convenience constructors on NSControls

- NSButton:  push, checkbox, radio
- NSSegmentedControl:  titles, images



# Control Constructors

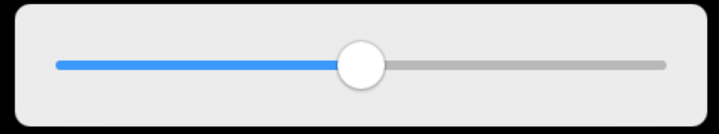
Convenience constructors on NSControls

- NSButton:  push, checkbox, radio
- NSSegmentedControl:  titles, images
- NSImageView

# Control Constructors

Convenience constructors on NSControls

- NSButton:  push, checkbox, radio
- NSSegmentedControl:  titles, images
- NSImageView
- NSSlider



# Control Constructors

Convenience constructors on NSControls

- NSButton:  push, checkbox, radio
- NSSegmentedControl:  titles, images
- NSImageView
- NSSlider
- NSTextField:  label, textfield

# Control Constructors

Convenience constructors on NSControls

- NSButton:  push, checkbox, radio
- NSSegmentedControl:  titles, images
- NSImageView
- NSSlider
- NSTextField:  label, textfield


Similar to Interface Builder object library

# Control Constructors

Convenience constructors on NSControls

- NSButton:  push, checkbox, radio
- NSSegmentedControl:  titles, images
- NSImageView
- NSSlider
- NSTextField:  label, textfield


Similar to Interface Builder object library

System standard setup—differing contexts

# Control Constructors

Convenience constructors on NSControls

- NSButton:  push, checkbox, radio
- NSSegmentedControl:  titles, images
- NSImageView
- NSSlider
- NSTextField:  label, textfield


Similar to Interface Builder object library

System standard setup—differing contexts

```swift
// Checkbox Before

let checkbox = NSButton(frame: NSZeroRect)
checkbox.title = "Hello World"
checkbox.setButtonType(.switchButton)
checkbox.bezelStyle = .roundedBezelStyle
checkbox.bordered = false
checkbox.imagePosition =
            (NSApp.userInterfaceLayoutDirection == .rightToLeft) ? .imageRight : .imageLeft
checkbox.userInterfaceLayoutDirection = NSApp.userInterfaceLayoutDirection
checkbox.target = self
checkbox.action = #selector(didToggleCheckbox(_:))
checkbox.sizeToFit() // only if not using constraint-based layout
```

```
// Checkbox Before

let checkbox = NSButton(frame: NSZeroRect)
checkbox.title = "Hello World"
checkbox.setButtonType(.switchButton)
checkbox.bezelStyle = .roundedBezelStyle
checkbox.bordered = false
checkbox.imagePosition =
            (NSApp.userInterfaceLayoutDirection == .rightToLeft) ? .imageRight : .imageLeft
checkbox.userInterfaceLayoutDirection = NSApp.userInterfaceLayoutDirection
checkbox.target = self
checkbox.action = #selector(didToggleCheckbox(_:))
checkbox.sizeToFit() // only if not using constraint-based layout
```

```swift
// Checkbox After

let checkbox = NSButton(checkboxWithTitle: "Hello World",
                        target: self, action: #selector(didToggleCheckbox(_:)))
```

```
// Label Before

let label = NSTextField(frame: NSZeroRect)
label.textColor = NSColor.labelColor()
label.font = NSFont.systemFontOfSize(0)
label.alignment = .natural
label.baseWritingDirection = .natural
label.userInterfaceLayoutDirection = NSApp.userInterfaceLayoutDirection
label.enabled = true
label.bezeled = false
label.bordered = false
label.drawsBackground = false
label.editable = false
label.selectable = false
label.stringValue = "Hello World"
checkbox.sizeToFit() // only if not using constraint-based layout
```

```
// Label Before

let label = NSTextField(frame: NSZeroRect)
label.textColor = NSColor.labelColor()
label.font = NSFont.systemFontOfSize(0)
label.alignment = .natural
label.baseWritingDirection = .natural
label.userInterfaceLayoutDirection = NSApp.userInterfaceLayoutDirection
label.enabled = true
label.bezeled = false
label.bordered = false
label.drawsBackground = false
label.editable = false
label.selectable = false
label.stringValue = "Hello World"
checkbox.sizeToFit() // only if not using constraint-based layout
```

```
// Label After

let label = NSTextField(labelWithString: "Hello World")
```

```
// Label After

let label = NSTextField(labelWithString: "Hello World”)
label.translatesAutoresizingMaskIntoConstraints = false // if positioning with constraints
```

# API Refinements

Nullability

Properties

Generics

Enumerations

@noescape

Weak delegates

Designated initializers

# API Refinements

## Weak delegates

| | |
|---|---|
| 10.9 and later | NSMenu |
| Linked on 10.11 | NSTableView and NSOutlineView |
| Linked on 10.12 | NSAlert—was effective ‘retain` |
| | NSComboBox |
| | NSSplitView |
| | NSTokenField |

# API Refinements

## Weak delegates

| | |
|---|---|
| 10.9 and later | NSMenu |
| Linked on 10.11 | NSTableView and NSOutlineView |
| Linked on 10.12 | NSAlert—was effective ‘retain` |
| | NSComboBox |
| | NSSplitView |
| | NSTokenField |

Non weak referenceable objects fallback to ‘assign` semantics

# API Refinements

## Designated initializers

# API Refinements

## Designated initializers

All AppKit classes declare designated initializer

# API Refinements

## Designated initializers

All AppKit classes declare designated initializer

- NSCursor's effective designated initializer has changed

# API Refinements

## Designated initializers

All AppKit classes declare designated initializer

- NSCursor's effective designated initializer has changed

Custom subclasses that do not call through designated initializer

# API Refinements

## Designated initializers

All AppKit classes declare designated initializer

- NSCursor's effective designated initializer has changed

Custom subclasses that do not call through designated initializer

- Objective-C: potential incorrectness and warnings

# API Refinements

## Designated initializers

All AppKit classes declare designated initializer

- NSCursor's effective designated initializer has changed

Custom subclasses that do not call through designated initializer

- Objective-C: potential incorrectness and warnings
- Swift: Build failure

# Foundation

# Foundation

"NS" prefix

Value types

Unit, Measurement

ISO8601DateFormatter

DateInterval

# “NS” Prefix

Drop “NS” prefix in key Foundation types in Swift

# “NS” Prefix

## Drop “NS” prefix in key Foundation types in Swift

Large subset of Foundation ships as part of Swift Core Libraries

# “NS” Prefix

## Drop “NS” prefix in key Foundation types in Swift

Large subset of Foundation ships as part of Swift Core Libraries

Match Foundation naming style with convention established by the Standard Library

# “NS” Prefix

## Drop “NS” prefix in key Foundation types in Swift

Large subset of Foundation ships as part of Swift Core Libraries

Match Foundation naming style with convention established by the Standard Library

- NSFormatter → Formatter
- NSJSONSerialization → JSONSerialization
- NSProgress → Progress
- NSData → Data
- NSURL → URL
- …

# “NS” Prefix

## Not dropped everywhere

Foundation only

# “NS” Prefix

## Not dropped everywhere

Foundation only

Not in APIs that are inherently tied to Objective-C

# “NS” Prefix

## Not dropped everywhere

Foundation only

Not in APIs that are inherently tied to Objective-C

- NSObject, NSProxy, NSAutoreleasePool, …

# “NS” Prefix

## Not dropped everywhere

Foundation only

Not in APIs that are inherently tied to Objective-C

- NSObject, NSProxy, NSAutoreleasePool, …

Not in platform-specific APIs

# “NS” Prefix

## Not dropped everywhere

Foundation only

Not in APIs that are inherently tied to Objective-C

- NSObject, NSProxy, NSAutoreleasePool, …

Not in platform-specific APIs

- NSUserNotification, NSXPCConnection, …

# “NS” Prefix

## Not dropped everywhere

Foundation only

Not in APIs that are inherently tied to Objective-C

- NSObject, NSProxy, NSAutoreleasePool, …

Not in platform-specific APIs

- NSUserNotification, NSXPCConnection, …

Not in classes also exposed as value types

# “NS” Prefix

## Not dropped everywhere

Foundation only

Not in APIs that are inherently tied to Objective-C

- NSObject, NSProxy, NSAutoreleasePool, …

Not in platform-specific APIs

- NSUserNotification, NSXPCConnection, …

Not in classes also exposed as value types

- NSData, NSURL, …

# Value Types

Types where value is important, not identity

# Value Types

## Types where value is important, not identity

NSString

NSData

NSURL

NSArray

...

# New Value Types

AffineTransform

CharacterSet

Data

Date

DateComponents

Decimal

IndexPath

IndexSet

DateInterval

Measurement

Notification

PersonNameComponents

URL

URLComponents

URLRequest

URLQueryItem

UUID

# Value Types

Existing class APIs still remain

```
struct Data ... {
    var count: Int
    func write(to: URL, options: WritingOptions = default) throws
    func range(of: Data, options: SearchOptions = default,
                                                in: Range<Index>? = default) -> Range<Index>?
    mutating func append(_ other: Data)
}

class NSData : NSObject, ... {
    var length: Int { get }
    func write(to: URL, options: WritingOptions = default) throws
    func range(of: Data, options: SearchOptions = default, in: NSRange) -> NSRange
}

class NSMutableData : NSData {
    func append(_ other: Data)
}
```

```
struct Data ... {
    var count: Int
    func write(to: URL, options: WritingOptions = default) throws
    func range(of: Data, options: SearchOptions = default,
                                                in: Range<Index>? = default) -> Range<Index>?
    mutating func append(_ other: Data)
}

class NSData : NSObject, ... {
    var length: Int { get }
    func write(to: URL, options: WritingOptions = default) throws
    func range(of: Data, options: SearchOptions = default, in: NSRange) -> NSRange
}

class NSMutableData : NSData {
    func append(_ other: Data)
}
```

```
struct Data ... {
    var count: Int
    func write(to: URL, options: WritingOptions = default) throws
    func range(of: Data, options: SearchOptions = default,
                                                in: Range<Index>? = default) -> Range<Index>?
    mutating func append(_ other: Data)
}

class NSData : NSObject, ... {
    var length: Int { get }
    func write(to: URL, options: WritingOptions = default) throws
    func range(of: Data, options: SearchOptions = default, in: NSRange) -> NSRange
}

class NSMutableData : NSData {
    func append(_ other: Data)
}
```

```
struct Data ... {
    var count: Int
    func write(to: URL, options: WritingOptions = default) throws
    func range(of: Data, options: SearchOptions = default,
                                                in: Range<Index>? = default) -> Range<Index>?
    mutating func append(_ other: Data)
}

class NSData : NSObject, ... {
    var length: Int { get }
    func write(to: URL, options: WritingOptions = default) throws
    func range(of: Data, options: SearchOptions = default, in: NSRange) -> NSRange
}

class NSMutableData : NSData {
    func append(_ other: Data)
}
```

```
struct Data ... {
    var count: Int
    func write(to: URL, options: WritingOptions = default) throws
    func range(of: Data, options: SearchOptions = default,
                                                in: Range<Index>? = default) -> Range<Index>?
    mutating func append(_ other: Data)
}

class NSData : NSObject, ... {
    var length: Int { get }
    func write(to: URL, options: WritingOptions = default) throws
    func range(of: Data, options: SearchOptions = default, in: NSRange) -> NSRange
}

class NSMutableData : NSData {
    func append(_ other: Data)
}
```

```
struct Data ... {
    var count: Int
    func write(to: URL, options: WritingOptions = default) throws
    func range(of: Data, options: SearchOptions = default,
                                                in: Range<Index>? = default) -> Range<Index>?
    mutating func append(_ other: Data)
}

class NSData : NSObject, ... {
    var length: Int { get }
    func write(to: URL, options: WritingOptions = default) throws
    func range(of: Data, options: SearchOptions = default, in: NSRange) -> NSRange
}

class NSMutableData : NSData {
    func append(_ other: Data)
}
```

```
struct Data ... {
    var count: Int
    func write(to: URL, options: WritingOptions = default) throws
    func range(of: Data, options: SearchOptions = default,
                                                in: Range<Index>? = default) -> Range<Index>?
    mutating func append(_ other: Data)
}

class NSData : NSObject, ... {
    var length: Int { get }
    func write(to: URL, options: WritingOptions = default) throws
    func range(of: Data, options: SearchOptions = default, in: NSRange) -> NSRange
}

class NSMutableData : NSData {
    func append(_ other: Data)
}
```

```
struct Data ... {
    var count: Int
    func write(to: URL, options: WritingOptions = default) throws
    func range(of: Data, options: SearchOptions = default,
                                                in: Range<Index>? = default) -> Range<Index>?
    mutating func append(_ other: Data)
}

class NSData : NSObject, ... {
    var length: Int { get }
    func write(to: URL, options: WritingOptions = default) throws
    func range(of: Data, options: SearchOptions = default, in: NSRange) -> NSRange
}

class NSMutableData : NSData {
    func append(_ other: Data)
}
```

# Value Types

| | | |
|---|---|---|
| What's New in Foundation for Swift | Mission | Tuesday 4:00PM |

# Unit, Measurement

Representing measured amounts

# Unit, Measurement

## Representing measured amounts

class Unit

- Miles, degrees celsius, km/h, …

# Unit, Measurement

## Representing measured amounts

class Unit

- Miles, degrees celsius, km/h, …

class Dimension

- Length, temperature, speed, …

# Unit, Measurement

## Representing measured amounts

class Unit

- Miles, degrees celsius, km/h, …

class Dimension

- Length, temperature, speed, …

# Unit, Measurement

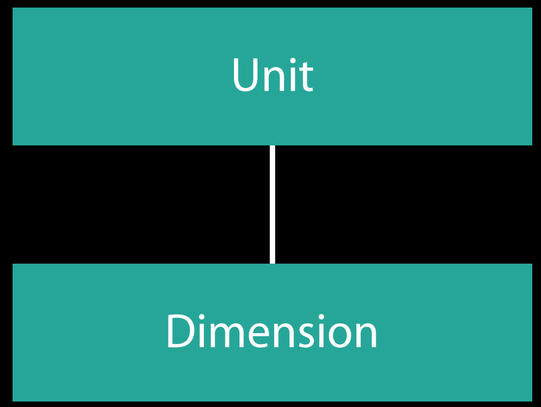
## Representing measured amounts

class Unit

- Miles, degrees celsius, km/h, …

class Dimension

- Length, temperature, speed, …



# Unit, Measurement

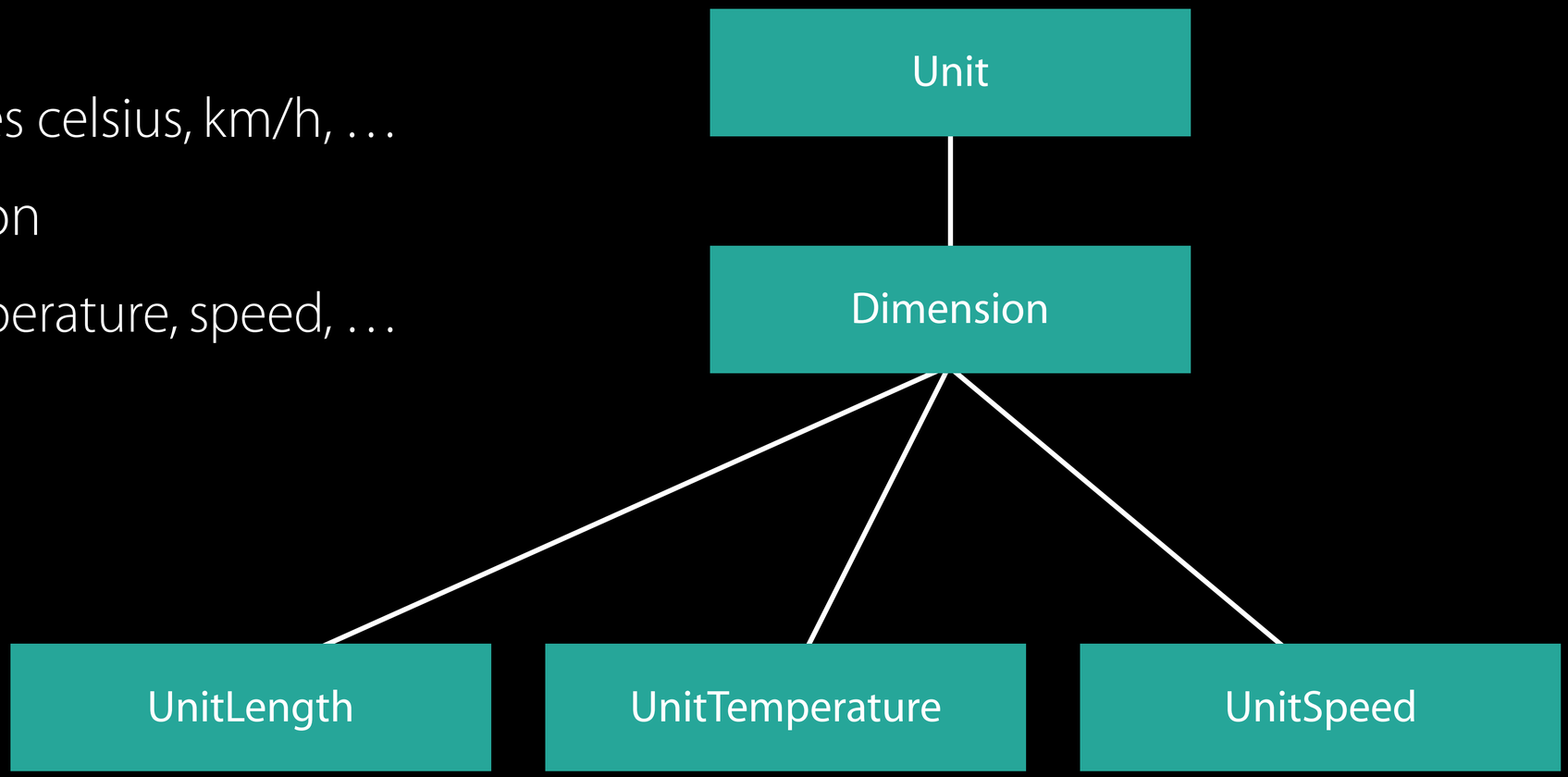
## Representing measured amounts

class Unit

- Miles, degrees celsius, km/h, …

class Dimension

- Length, temperature, speed, …



# Unit, Measurement

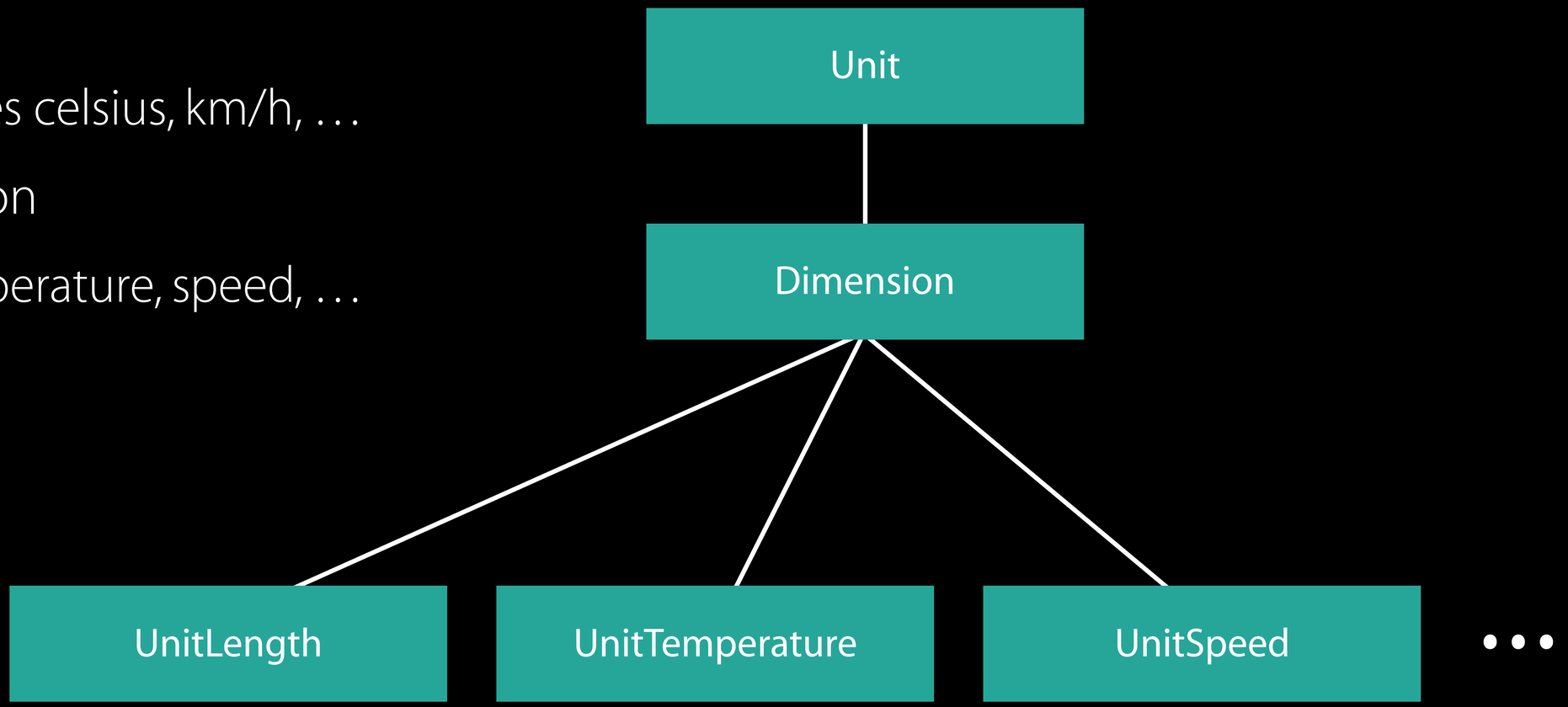
## Representing measured amounts

class Unit

- Miles, degrees celsius, km/h, …

class Dimension

- Length, temperature, speed, …



# Unit, Measurement

class Unit

- Miles, degrees celsius, km/h, …

class Dimension

- Length, temperature, speed, …

class UnitConverter

- miles ⟷ km, ℃ ⟷ K, km/h ⟷ knots, …

# Unit, Measurement

class Unit

- Miles, degrees celsius, km/h, …

class Dimension

- Length, temperature, speed, …

class UnitConverter

- miles ⟷ km, ℃ ⟷ K, km/h ⟷ knots, …

struct Measurement

- “10 miles,” “24 degrees celsius,” “42 km/h”, …

# Unit, Measurement

class Unit

- Miles, degrees celsius, km/h, …

class Dimension

- Length, temperature, speed, …

class UnitConverter

- miles ⟷ km, °C ⟷ K, km/h ⟷ knots, …

struct Measurement

- “10 miles,” “24 degrees celsius,” “42 km/h”, …

class MeasurementFormatter

# Predefined Dimensions

## Subclasses of Dimension

UnitAcceleration
UnitAngle
UnitArea
UnitConcentrationMass
UnitDispersion
UnitDuration
UnitElectricCharge
UnitElectricCurrent
UnitElectricPotentialDifference
UnitElectricResistance
UnitEnergy
UnitFrequency
UnitFuelEfficiency
UnitLength
UnitIlluminance
UnitMass
UnitPower
UnitPressure
UnitSpeed
UnitTemperature
UnitVolume

# Definition of Units

```
public class UnitTemperature : Dimension, NSSecureCoding {
    public class var kelvin: UnitTemperature { get }        // Base unit
    public class var celsius: UnitTemperature { get }
    public class var fahrenheit: UnitTemperature { get }
}
```

# Definition of Units

```
public class UnitTemperature : Dimension, NSSecureCoding {
    public class var kelvin: UnitTemperature { get }        // Base unit
    public class var celsius: UnitTemperature { get }
    public class var fahrenheit: UnitTemperature { get }
}
```

# DateInterval

# DateInterval

Represents a date interval

# DateInterval

Represents a date interval

```
struct DateInterval : Comparable, Hashable {
    var start: Date
    var end: Date
    var duration: TimeInterval
}
```

# DateInterval

Represents a date interval

```
struct DateInterval : Comparable, Hashable {
    var start: Date
    var end: Date
    var duration: TimeInterval
}
```

A new type for DateIntervalFormatter to format

# DateInterval

Represents a date interval

```
struct DateInterval : Comparable, Hashable {
    var start: Date
    var end: Date
    var duration: TimeInterval
}
```

A new type for DateIntervalFormatter to format

```
class DateIntervalFormatter : Formatter {
    func string(from: Date, to: Date) -> String
    func string(from: DateInterval) -> String?       // 10.12
}
```

# Handling Dates

## Can be tricky

# Handling Dates

## Can be tricky

To represent a ten second period

# Handling Dates

## Can be tricky

To represent a ten second period

```
let nextTenSeconds = DateInterval(start: startDate, duration: 10)
```

# Handling Dates

## Can be tricky

To represent a ten second period

```
let nextTenSeconds = DateInterval(start: startDate, duration: 10)
```

To represent a “day”

# Handling Dates

## Can be tricky

To represent a ten second period

```
let nextTenSeconds = DateInterval(start: startDate, duration: 10)
```

To represent a “day”

```
let aDay = DateInterval(start: startDate, duration: 24 * 60 * 60)
```

# Handling Dates

## Can be tricky

To represent a ten second period

```
let nextTenSeconds = DateInterval(start: startDate, duration: 10)
```

To represent a “day”

```
let aDay = DateInterval(start: startDate, duration: 24 * 60 * 60)
```

# Handling Dates

## Can be tricky

To represent a ten second period

```
let nextTenSeconds = DateInterval(start: startDate, duration: 10)
```

To represent a “day”

```
let aDay = DateInterval(start: startDate, duration: 24 * 60 * 60)
```

Days are not always 24 hours long!

# Handling Dates

## Can be tricky

To represent a ten second period

```
let nextTenSeconds = DateInterval(start: startDate, duration: 10)
```

To represent a “day”

```
let aDay = DateInterval(start: startDate, duration: 24 * 60 * 60)
```

Days are not always 24 hours long!

---

Solutions to Common Date and Time Challenges WWDC 2013

# ISO8601 DateFormatter

Formatter for dates using ISO8601 standard

# ISO8601 DateFormatter

Formatter for dates using ISO8601 standard

Distinct from DateFormatter

# ISO8601 DateFormatter

Formatter for dates using ISO8601 standard

Distinct from DateFormatter

```
// Create ISO8601 date string from current date
```

# ISO8601DateFormatter

Formatter for dates using ISO8601 standard

Distinct from DateFormatter

```
// Create ISO8601 date string from current date
let formatter = ISO8601DateFormatter()
```

# ISO8601DateFormatter

Formatter for dates using ISO8601 standard

Distinct from DateFormatter

```
// Create ISO8601 date string from current date
let formatter = ISO8601DateFormatter()
let currentDate = Date()
```

# ISO8601DateFormatter

Formatter for dates using ISO8601 standard

Distinct from DateFormatter

```
// Create ISO8601 date string from current date
let formatter = ISO8601DateFormatter()
let currentDate = Date()
let result = formatter.string(from: currentDate)
```

# ISO8601DateFormatter

Formatter for dates using ISO8601 standard

Distinct from DateFormatter

```
// Create ISO8601 date string from current date
let formatter = ISO8601DateFormatter()
let currentDate = Date()
let result = formatter.string(from: currentDate)          // Output: 2016-06-14T18:57:42Z
```

# ISO8601DateFormatter

Formatter for dates using ISO8601 standard

Distinct from DateFormatter

```
// Create ISO8601 date string from current date
let formatter = ISO8601DateFormatter()
let currentDate = Date()
let result = formatter.string(from: currentDate)          // Output: 2016-06-14T18:57:42Z
```

Also does parsing

# ISO8601DateFormatter

Formatter for dates using ISO8601 standard

Distinct from DateFormatter

```
// Create ISO8601 date string from current date
let formatter = ISO8601DateFormatter()
let currentDate = Date()
let result = formatter.string(from: currentDate)          // Output: 2016-06-14T18:57:42Z
```

Also does parsing

```
// Create a date from ISO8601 date string
let formatter = ISO8601DateFormatter()
let date = formatter.date(from: result)
```

# Other Foundation Updates

# Other Foundation Updates

URL

- New URL properties, such as canonical path
- New class URLSessionTaskMetrics

# Other Foundation Updates

URL

- New URL properties, such as canonical path
- New class URLSessionTaskMetrics

PersonNameComponentsFormatter

- Now parses names

# Other Foundation Updates

URL

- New URL properties, such as canonical path
- New class URLSessionTaskMetrics

PersonNameComponentsFormatter

- Now parses names

DateComponentsFormatter

- New brief style

# Core Data

Generics

Generational querying

Persistent store description

NSFetchedResultsController

What's New in Core Data — Pacific Heights — Friday 10:00AM

# Crafting Modern Cocoa Apps

Overview of recent APIs important to creating modern applications for the Mac

Pointers to other sessions of interest

Appropriate for everyone

| Crafting Modern Cocoa Apps | Pacific Heights | Friday 5:00PM |
| --- | --- | --- |

# More Information

https://developer.apple.com/wwdc16/203

# Related Sessions

| | | |
|---|---|---|
| Swift API Design Guidelines | Presidio | Tuesday 10:00AM |
| What's New in Foundation for Swift | Mission | Tuesday 4:00PM |
| Working with Wide Color | Mission | Thursday 1:40PM |
| What's New in International User Interfaces | Nob Hill | Friday 9:00AM |
| What's New in Core Data | Pacific Heights | Friday 10:00AM |
| What's New in Auto Layout | Presidio | Friday 3:00PM |
| Measurements and Units | Presidio | Friday 4:00PM |
| Crafting Modern Cocoa Apps | Pacific Heights | Friday 5:00PM |

# Labs

| | | |
|---|---|---|
| Cocoa Lab | Frameworks Lab B | Tuesday 12:30PM |
| Swift and Foundation Lab | Developer Tools Lab A | Wednesday 9:00AM |
| Color Lab | Frameworks Lab A | Wednesday 1:00PM |
| Cocoa Lab | Frameworks Lab D | Thursday 2:00PM |
| Cocoa Lab | Frameworks Lab A | Friday 1:00PM |
| Color Lab | Graphics, Games, and Media Lab C | Friday 4:00PM |

WWDC16