#WWDC19

What's New in AppKit for macOS?

Chris Dreessen

Colors

Screens

Text

Toolbars

Controls

Events

Geometry

Formatters

Extensions

New Frameworks

UIKit

Runs iPad Apps on the Mac

Introducing iPad Apps for Mac

WWDC 2019

SwiftUI

Describes view hierarchies

Swift native

Colors

New Colors

Tagged Pointer Colors

NSString and NSNumber already use this trick

Allocating colors becomes very cheap

fastness per second

Tagged Pointer Colors

Note that derived objects can have different lifetimes

```
CGColorRef blueCGColor = nil;

@autoreleasepool {
    blueCGColor = NSColor.blueColor.CGColor;
}

CGContextSetFillColorWithColor(..., blueCGColor);
```

NSColorSampler

NSColorSampler

Dynamic Color Provider

```
let color = NSColor(name: "userWidgetColor") { appearance in
    switch appearance.bestMatch(from: [.aqua, .darkAqua]) {
    case .darkAqua:
        return darkUserWidgetColor
    case .aqua, default:
        return lightUserWidgetColor
    }
}
```

Screens

NSScreen.localizedName

<NSScreen: 0x60000261e460>

<NSScreen: 0x60000261f900>

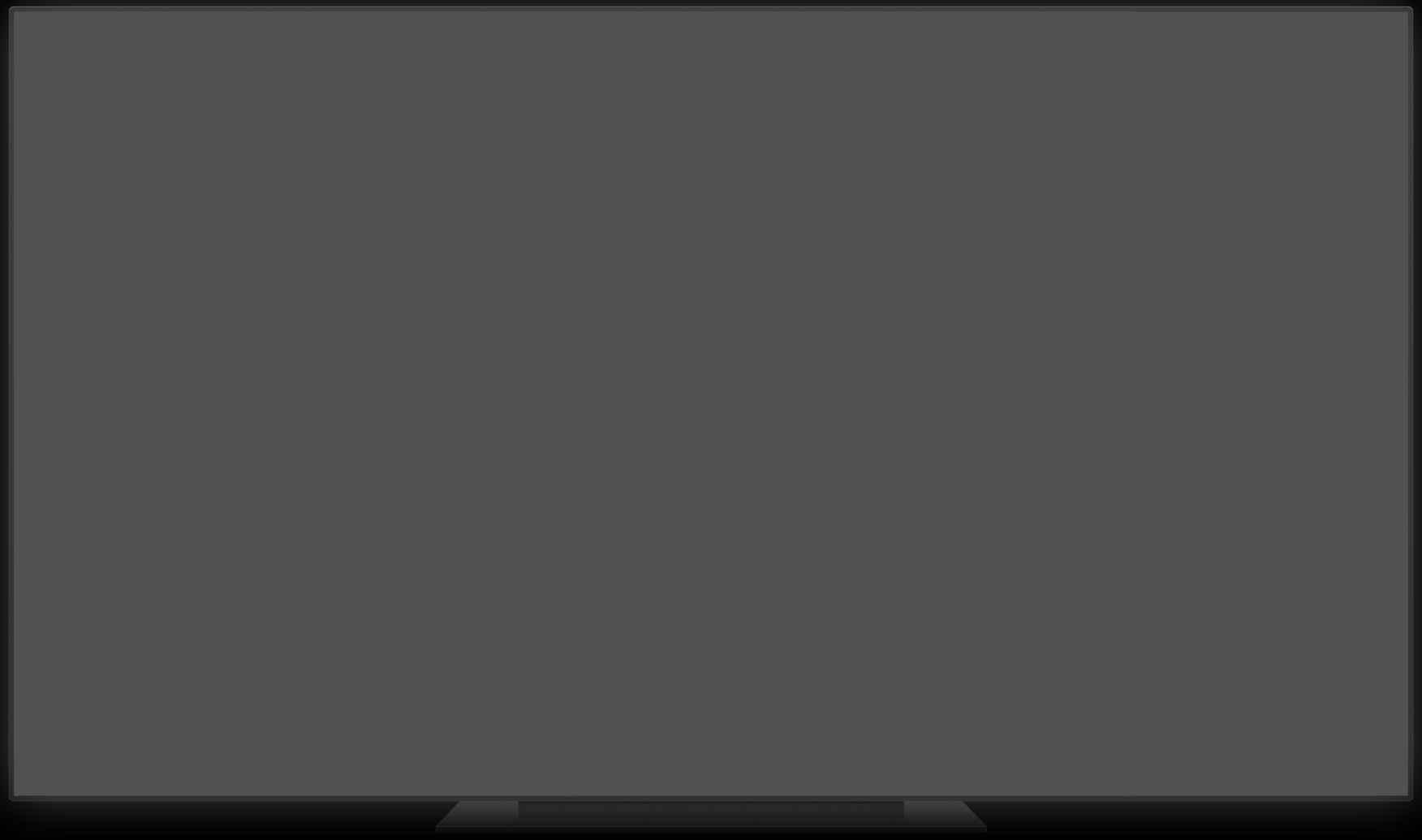
<NSScreen: 0x600002620000>

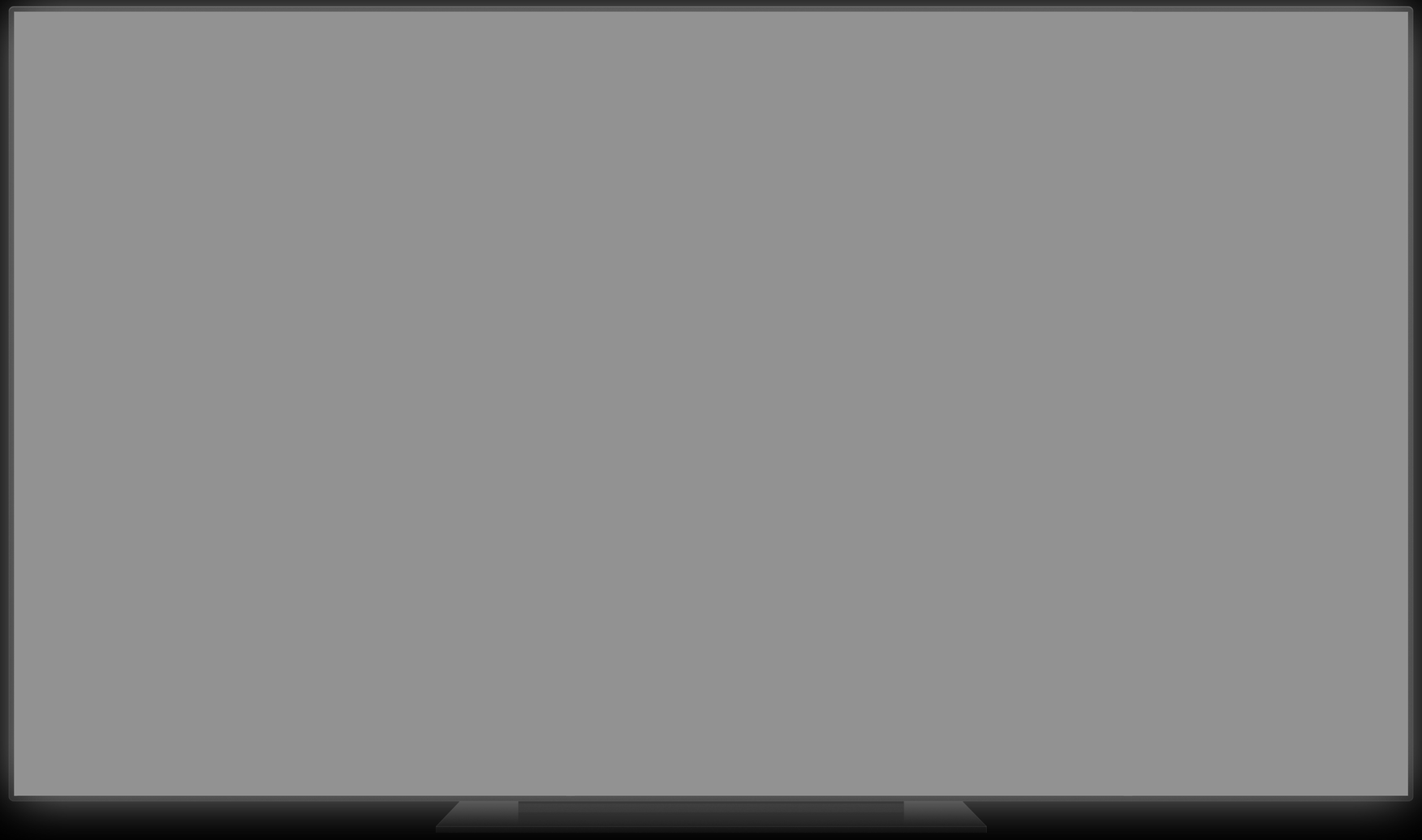
NSScreen.localizedName

<NSScreen: 0x60000261e460> -> “LG Ultrafine”

<NSScreen: 0x60000261f900> -> “Thunderbolt Display”

<NSScreen: 0x600002620000> -> “Color LCD”





Extended Dynamic Range

Extended Dynamic Range

Extended Dynamic Range

Extended Dynamic Range

NSScreen.maximumExtendedDynamicRangeColorComponentValue (10.11)

CAMetalLayer.wantsExtendedDynamicRangeContent (10.11)

Extended Dynamic Range

NSScreen.maximumPotentialExtendedDynamicRangeColorComponentValue

Extended Dynamic Range

NSScreen.maximumPotentialExtendedDynamicRangeColorComponentValue

Extended Dynamic Range

NSScreen.maximumReferenceExtendedDynamicRangeColorComponentValue

Convenient MTLDevice Identification

```
let preferredDevice =
CGDirectDisplayCopyCurrentMetalDevice(self.window?.screen?.deviceDescription[“NSScreenNumber”])
```

Convenient MTLDevice Identification

```
let preferredDevice =
CGDirectDisplayCopyCurrentMetalDevice(self.window?.screen?.deviceDescription[“NSScreenNumber”])
```

CAMetalLayer.preferredDevice

MTKView.preferredDevice

Convenient MTLDevice Identification

```
let preferredDevice = self.preferredDevice
```

CAMetalLayer.preferredDevice

MTKView.preferredDevice

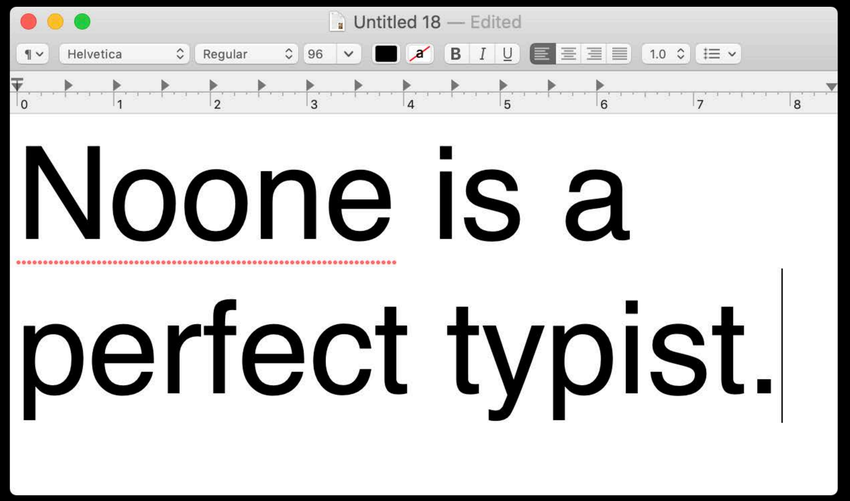
Text

NSTextView.usesAdaptiveColorMappingForDarkAppearance

NSTextCheckingController

Like NSSpellChecker

Works with the NSTextCheckingClient protocol



Secure Coding Support

NSLayoutManager

NSTextContainer

NSTextStorage

NSTextAttachment

NSTextAlternatives

NSTextTable

NSTextBlock

NSFontDescriptor System Design

Font Management and Text Scaling

Thursday, 9:00

NSAttributedString Scaling

```
NSAttributedString.DocumentAttributeKey.textScaling
NSAttributedString.DocumentAttributeKey.sourceTextScaling
NSAttributedString.DocumentReadingOptionKey.targetTextScaling
NSAttributedString.DocumentReadingOptionKey.sourceTextScaling
```

iOS

Hello 160pt

macOS

Hello 160pt

NSAttributedString Scaling

```
NSAttributedString.DocumentAttributeKey.textScaling
NSAttributedString.DocumentAttributeKey.sourceTextScaling
NSAttributedString.DocumentReadingOptionKey.targetTextScaling
NSAttributedString.DocumentReadingOptionKey.sourceTextScaling
```

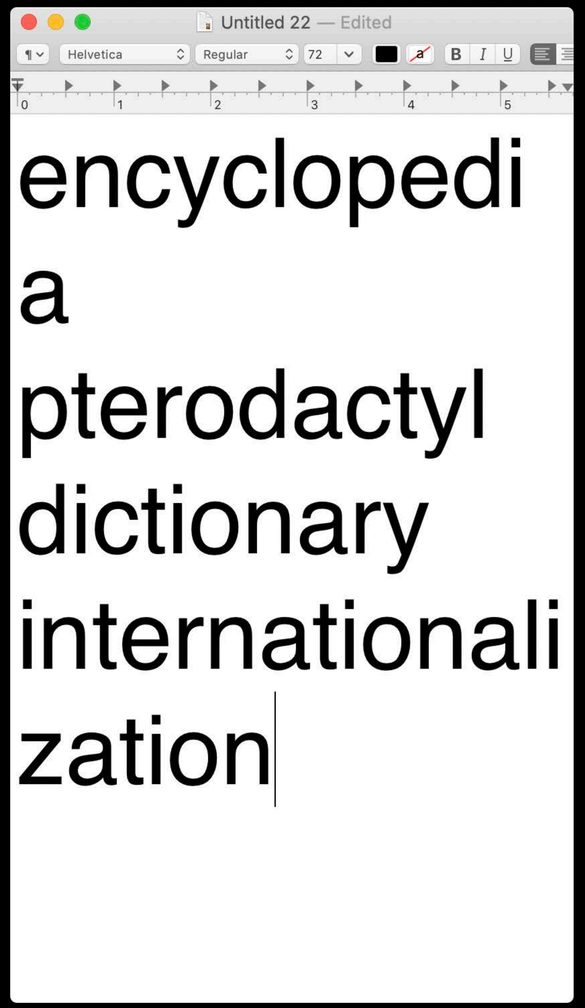
iOS

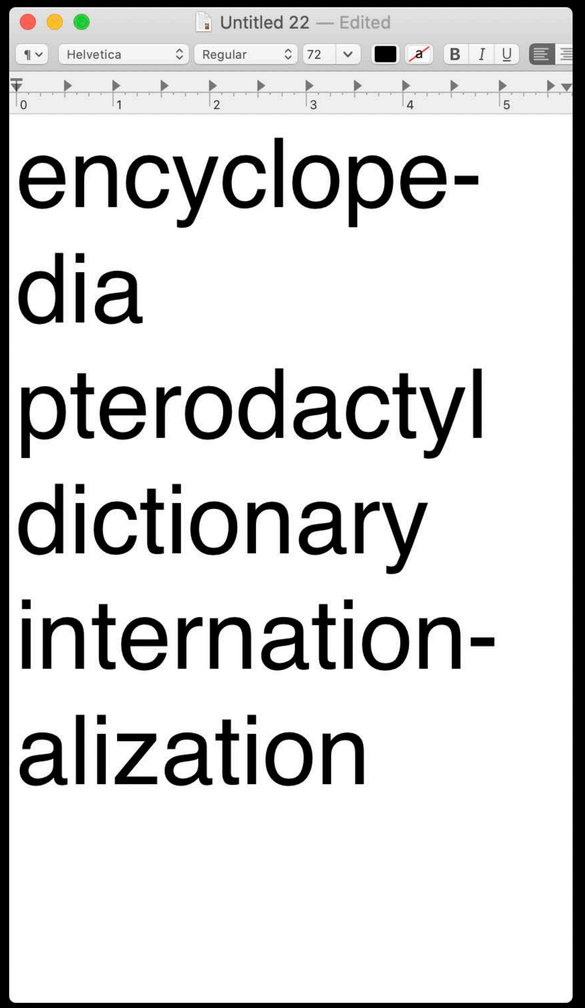
Hello 160pt

macOS

Hello 120pt

NSLayoutManager.usesDefaultHyphenation





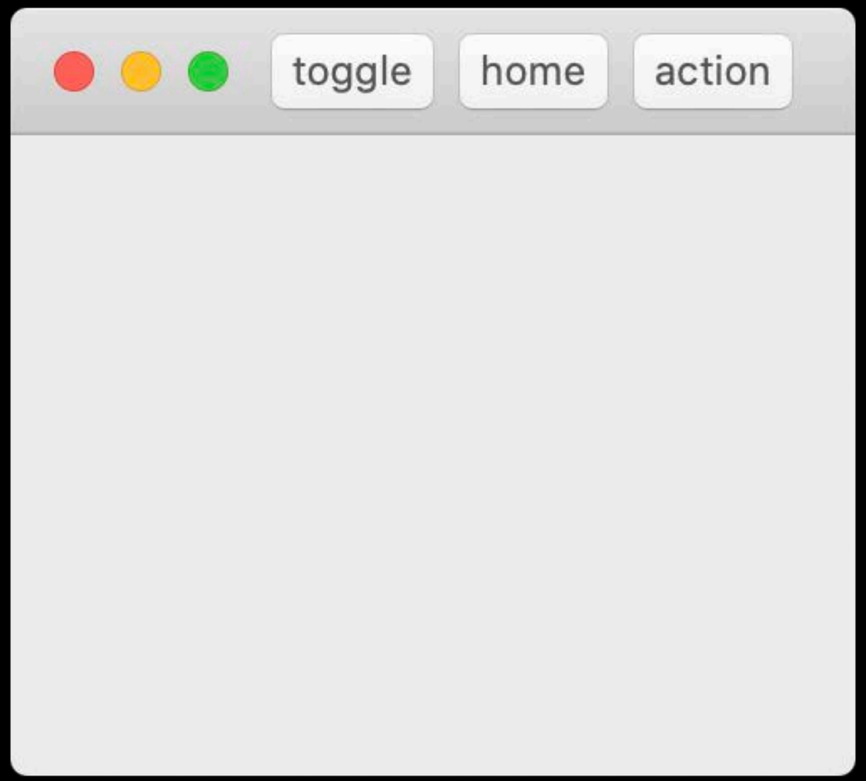
Toolbars

New NSToolbarItem Properties

NSToolbarItem.isBordered

New NSToolbarItem Properties

NSToolbarItem.title

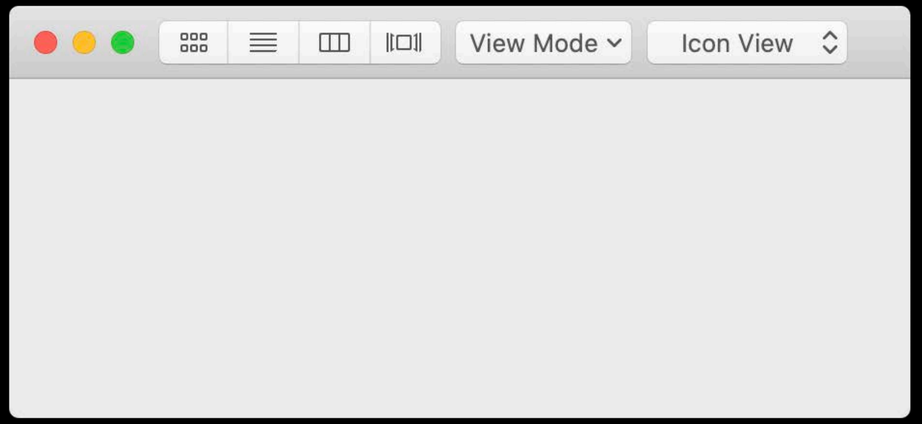


NSToolbarItemGroup

Convenience constructors

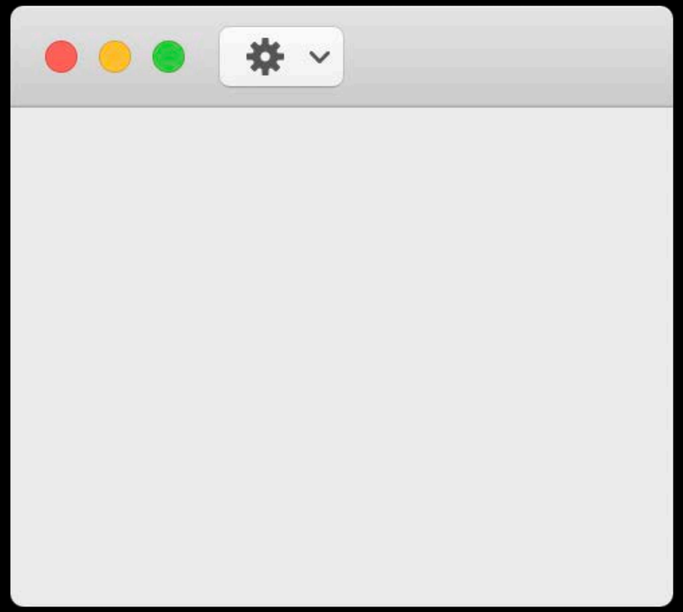
Segmented controls

Collapsed representation



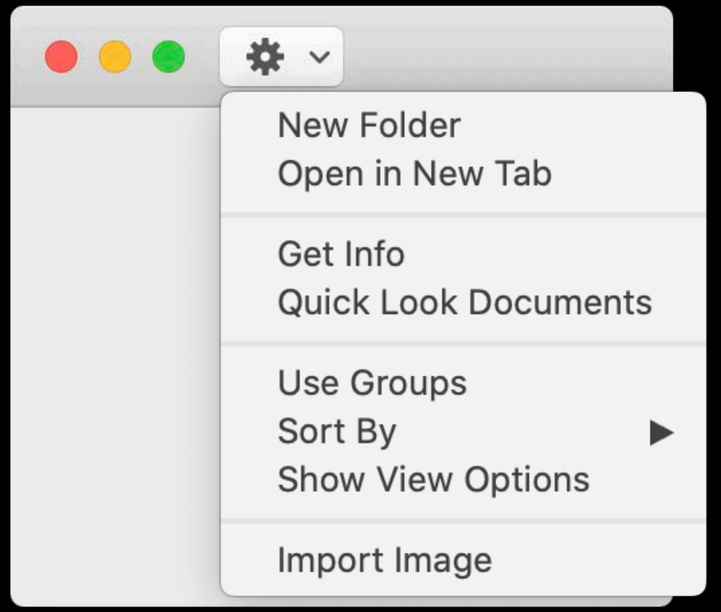
NSMenuToolbarItem

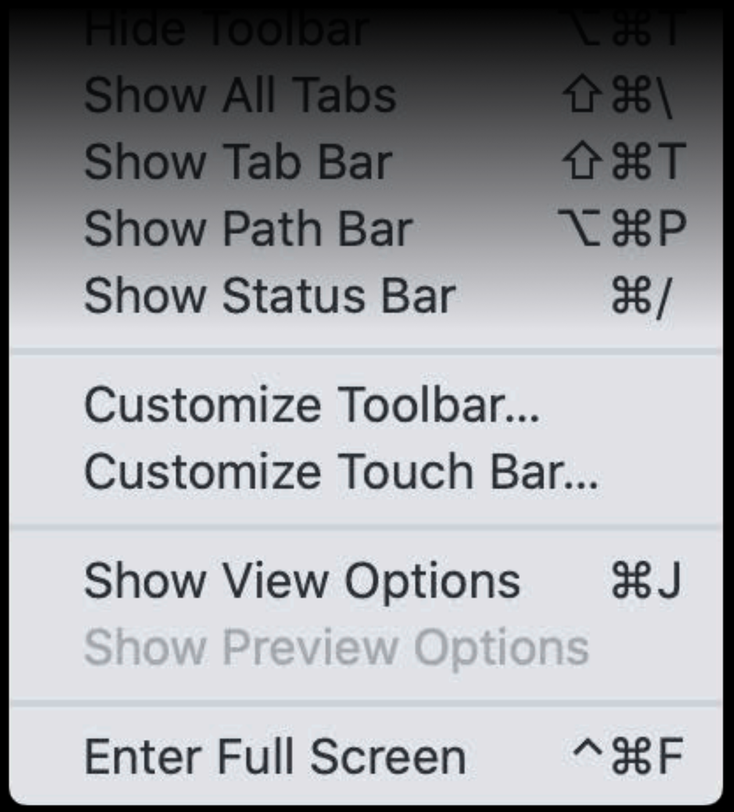
Uses NSMenu



NSMenuToolbarItem

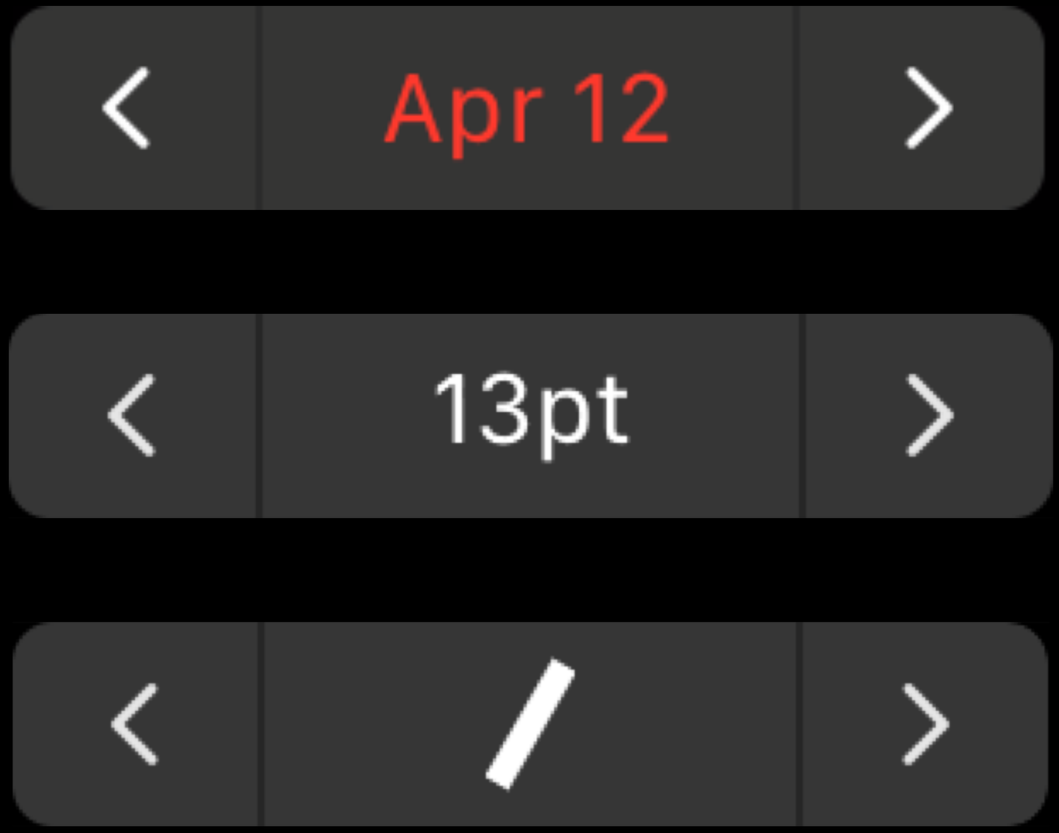
Uses NSMenu





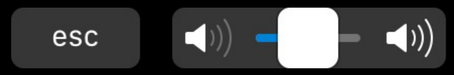
NSTouchBar.isAutomaticCustomizeTouchBarMenuItemEnabled

NSStepperTouchBarItem

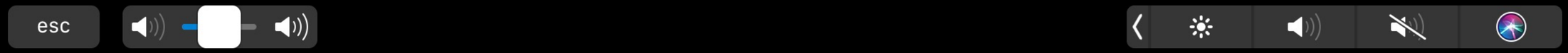


NSSliderTouchBarItem

NSSliderTouchBarItem

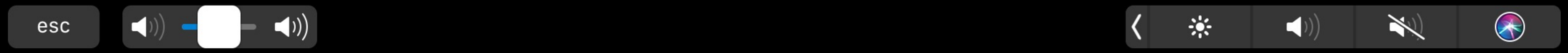


NSSliderTouchBarItem

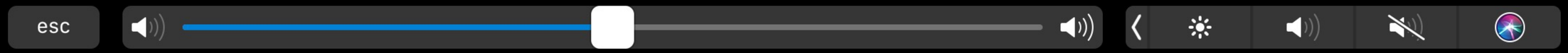


NSSliderTouchBarItem.minimumSliderWidth

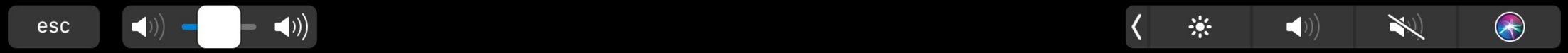
NSSliderTouchBarItem



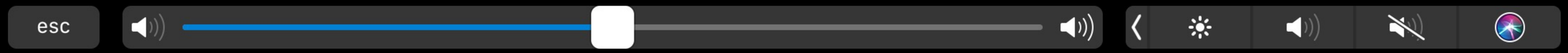
NSSliderTouchBarItem.minimumSliderWidth



NSSliderTouchBarItem

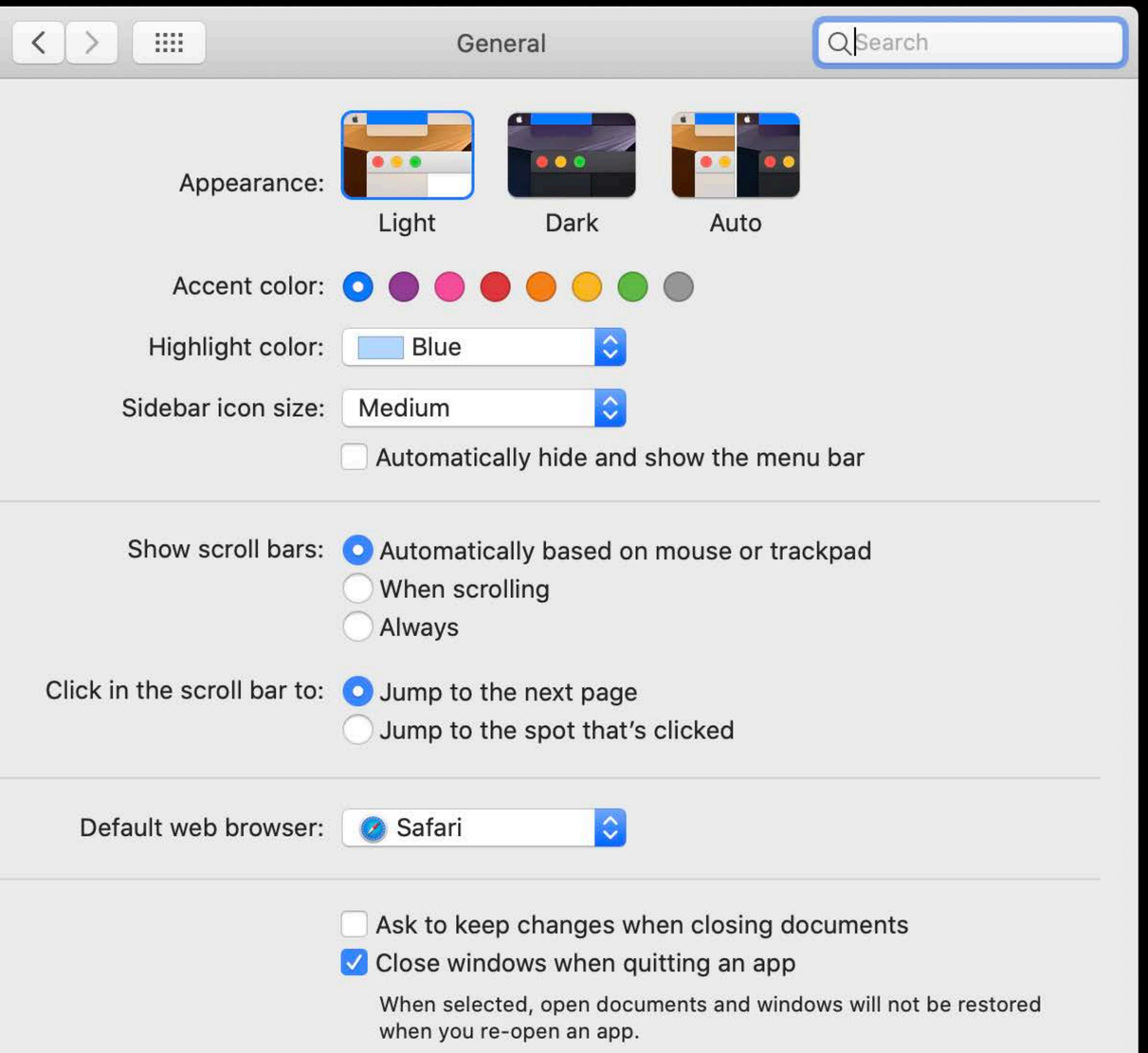


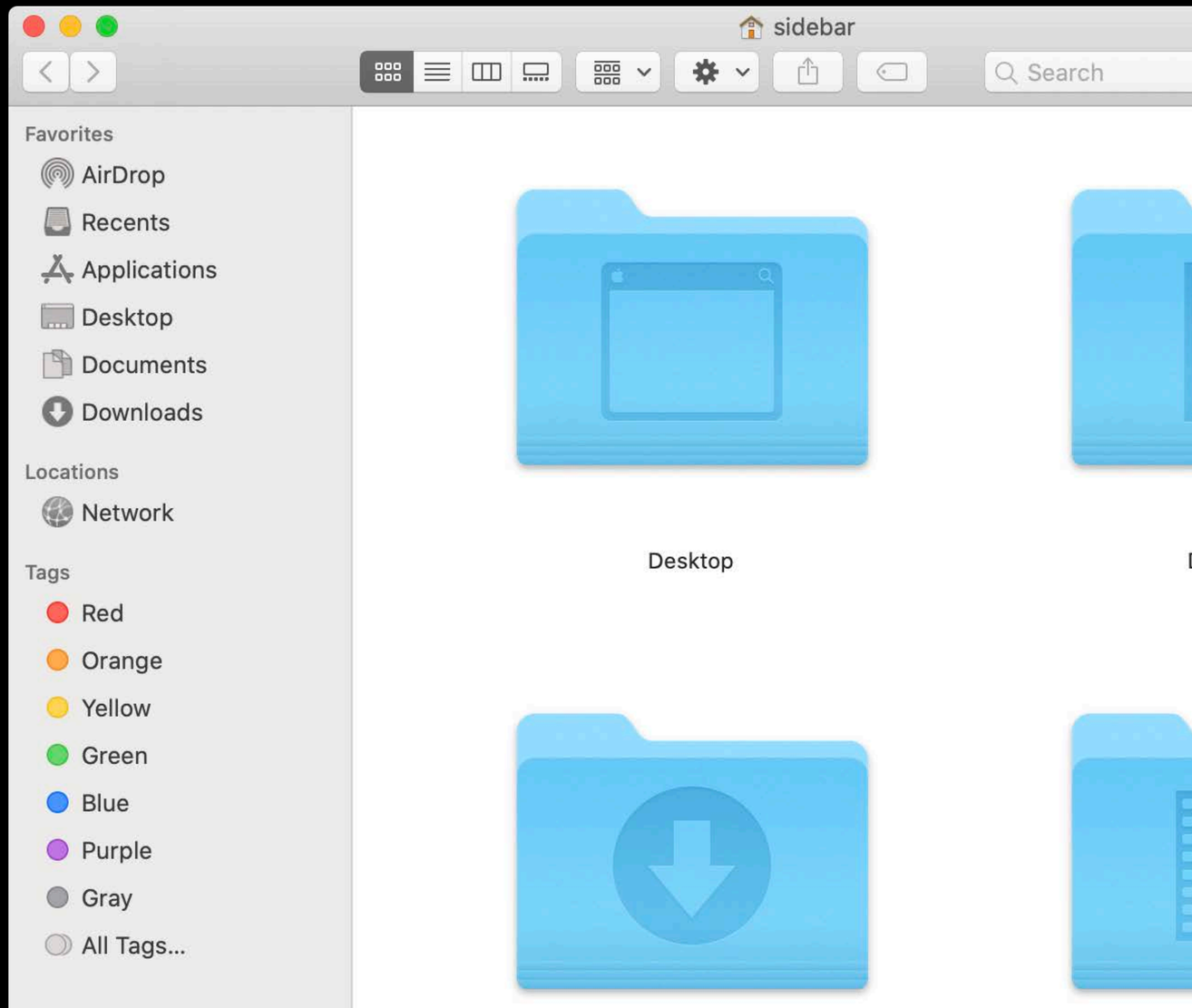
NSSliderTouchBarItem.minimumSliderWidth



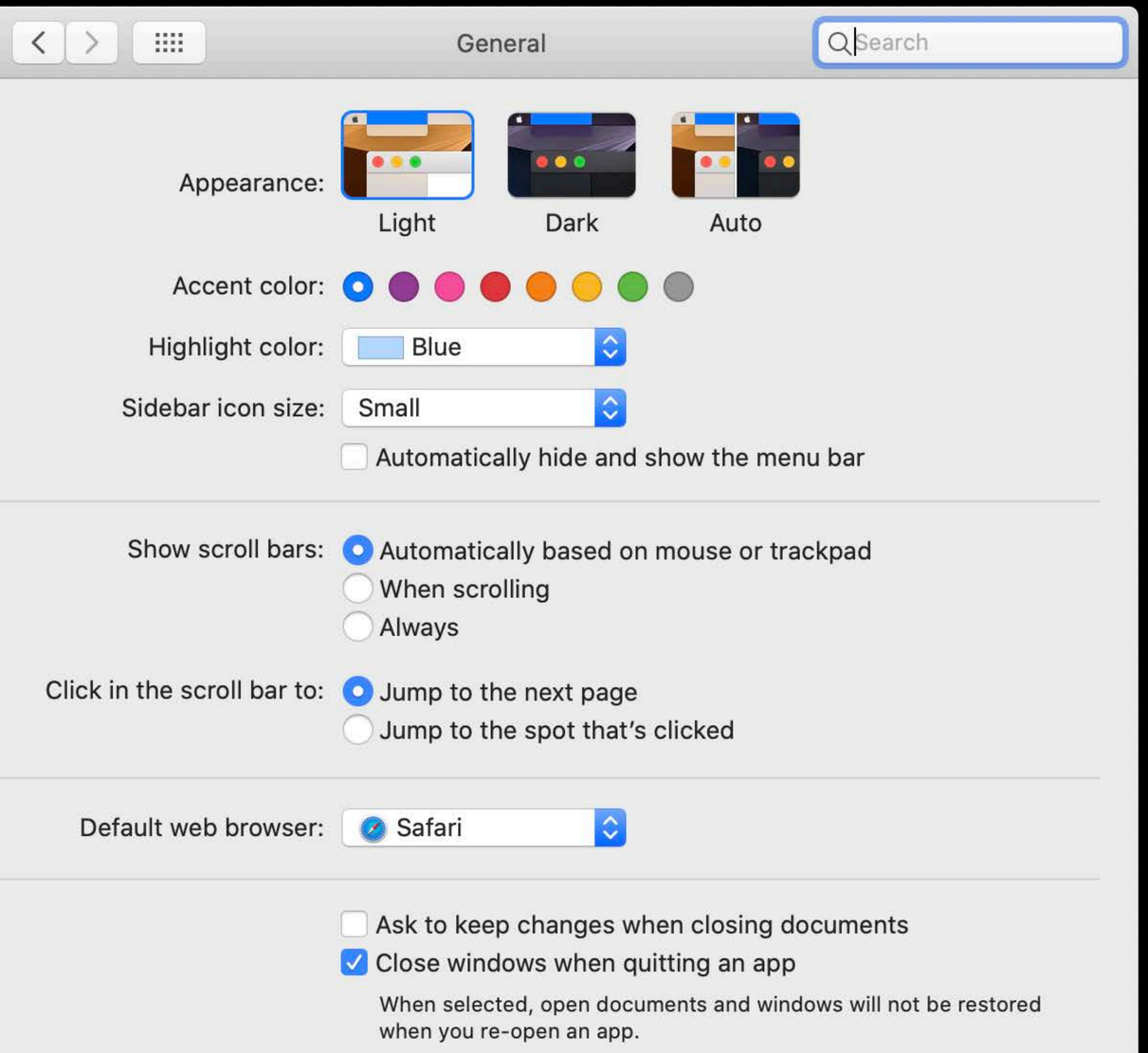
NSSliderTouchBarItem.maximumSliderWidth

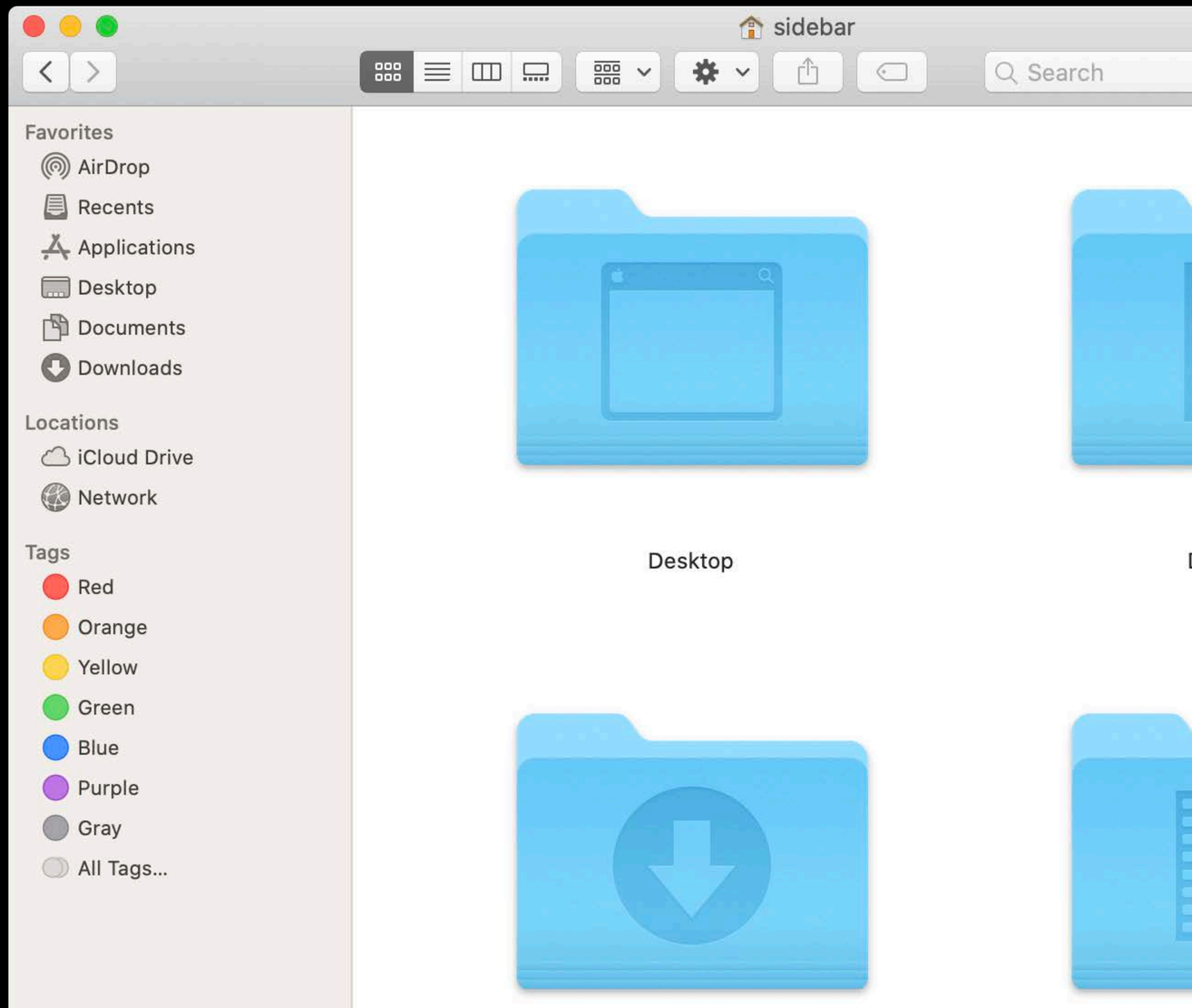
Sidebar Metrics



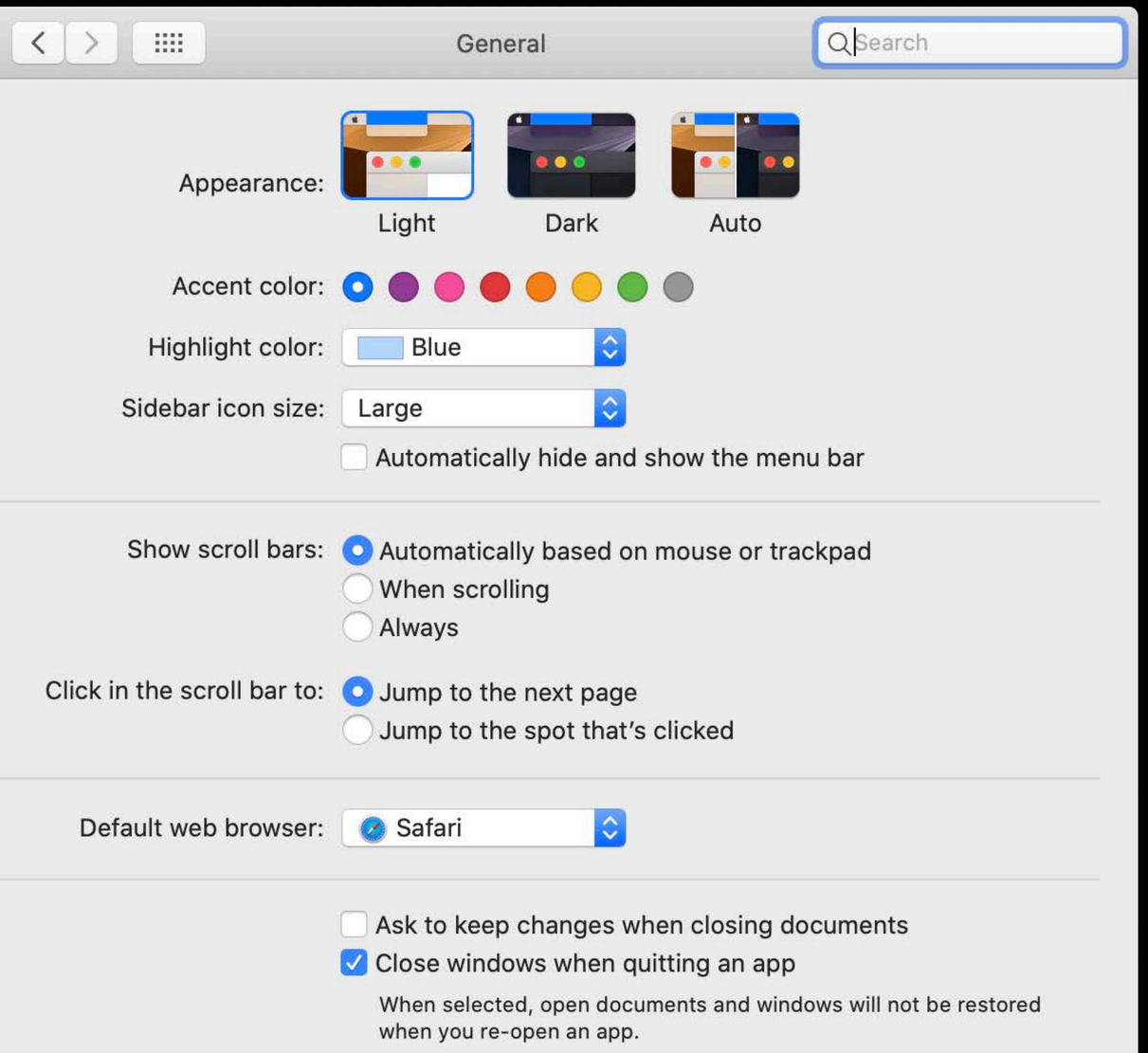


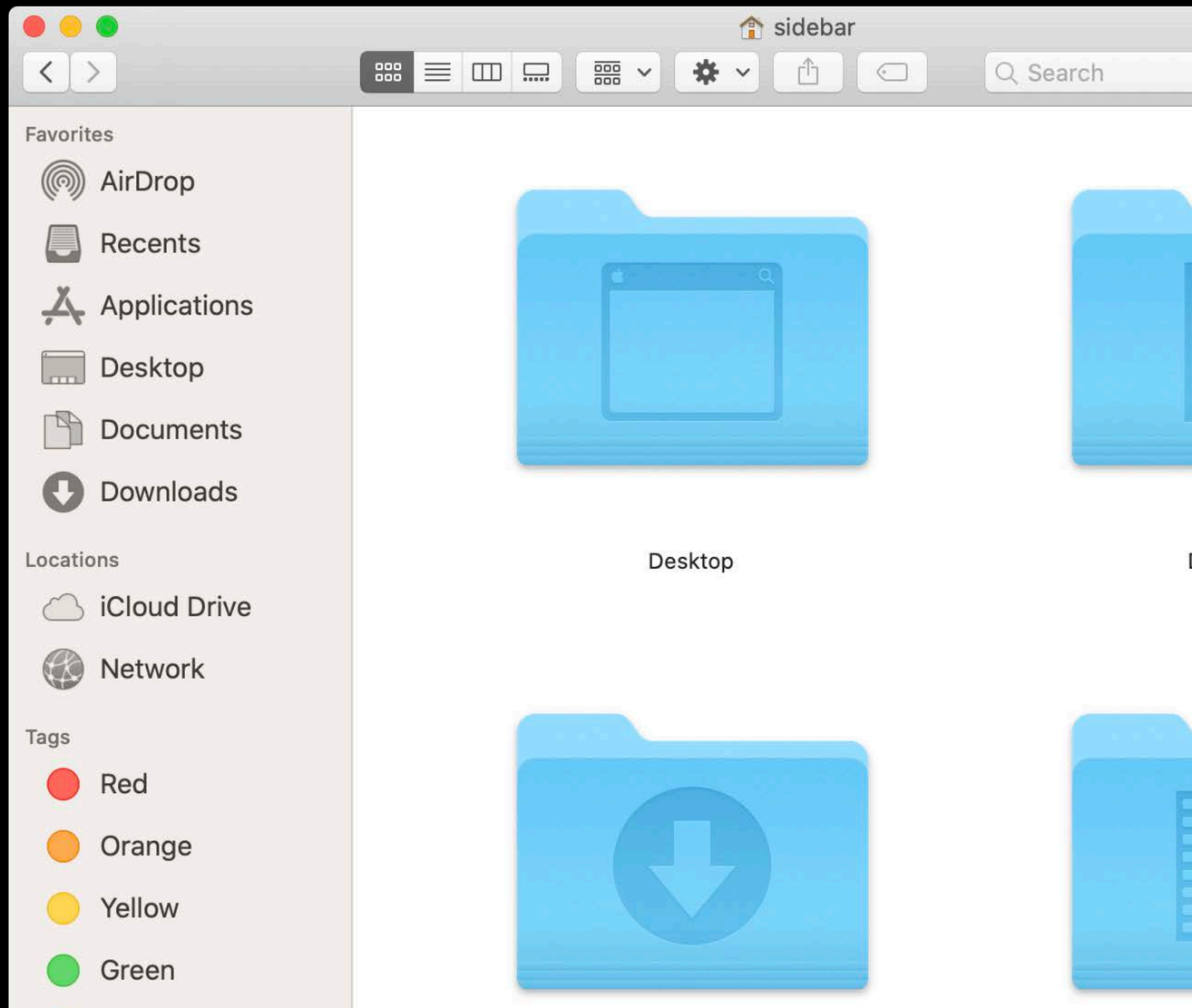
Sidebar Metrics





Sidebar Metrics



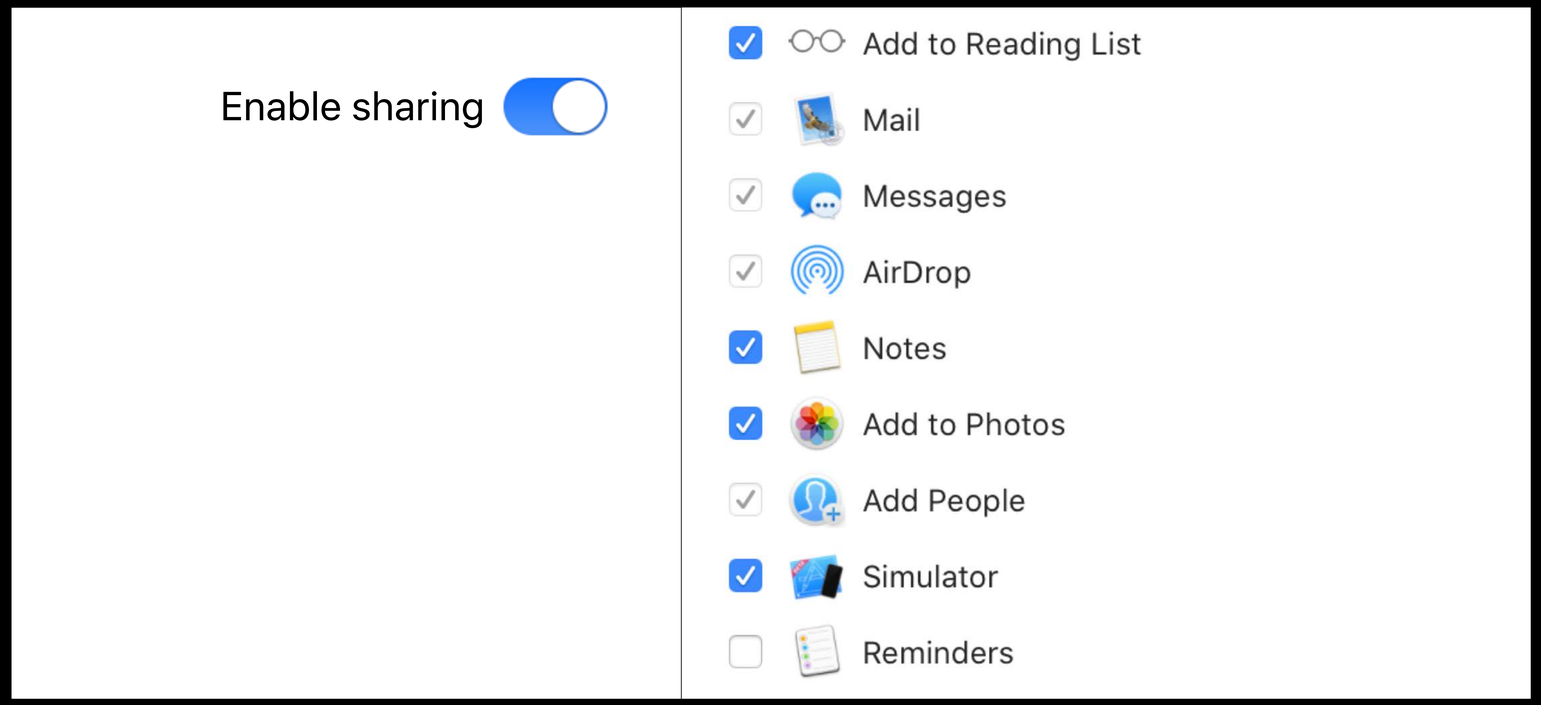


Controls

NSSwitch

Can use NSSwitch for big toggles

Avoid using NSSwitch for small things, especially lists



Collection View Compositional Layout

Layouts described by composition

Container-relative sizing

Layout breaks

Nestable groups

Scrollable sections

Collection View Diffable Data Sources

Identifier-based API for tracking item and section changes

No more performBatchUpdates() or reloadData()

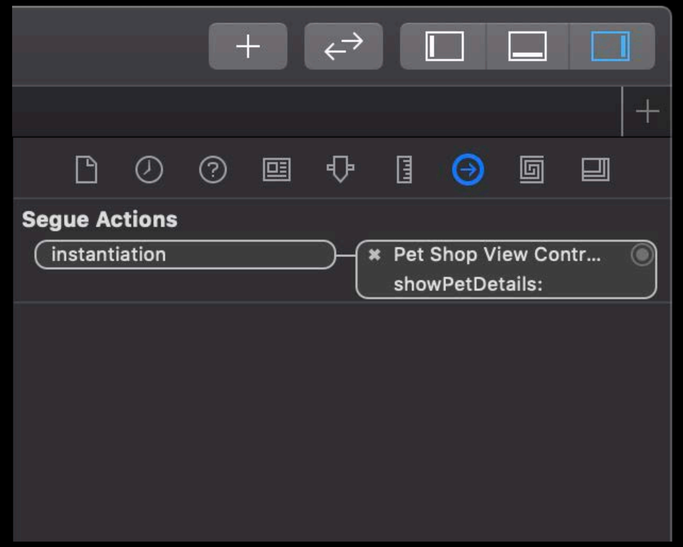
Advances in Collection View Layout

Wednesday, 11:00

IBSegueAction

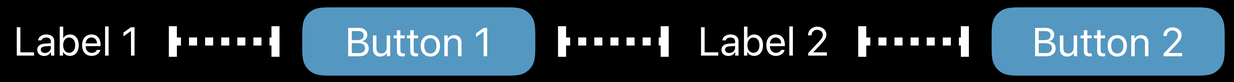
Use with IB storyboards

Can invoke your own code to instantiate view controllers




```
@IBSegueAction func showPetDetails(_ coder: NSCoder) ->
NSViewController {
    return PetDetailsViewController(coder: coder,
name:self.selectedPetName)
}
```

Disabling Intrinsic Content Sizes



Disabling Intrinsic Content Sizes

Disabling Intrinsic Content Sizes

Long Label 1	Long Label 2	Long Label 3	Long Label 4
Long Label 5	Long Label 6	Long Label 7	Long Label 8
Long Label 9	Long Label 10	Long Label 11	Long Label 12
Long Label 13	Long Label 14	Long Label 15	Long Label 16
Long Label 17	Long Label 18	Long Label 19	Long Label 20

Disabling Intrinsic Content Sizes

NSView.isHorizontalContentSizeConstraintActive

NSView.isVerticalContentSizeConstraintActive

NSResponder Block Capture Safety

```
let label = NSTextField(labelWithString:“working on it”)

dispatch_async(dispatch_get_global_queue(0, 0)) {
   // Do some work, then bounce back to the main thread to update the UI
    dispatch_async(dispatch_get_main_queue()) {
        label.value = “done”
    }
   // The label could be deallocated here on the background thread. boom!
}
```

NSResponder Block Capture Safety

```
let label = NSTextField(labelWithString:“working on it”)

dispatch_async(dispatch_get_global_queue(0, 0)) {
   // Do some work, then bounce back to the main thread to update the UI
    dispatch_async(dispatch_get_main_queue()) {
        label.value = “done”
    }
   // The label could be deallocated here on the background thread. boom!
}
```

NSResponder Block Capture Safety

```
let label = NSTextField(labelWithString:“working on it”)

dispatch_async(dispatch_get_global_queue(0, 0)) {
   // Do some work, then bounce back to the main thread to update the UI
    dispatch_async(dispatch_get_main_queue()) {
        label.value = “done”
    }
   // The label could be deallocated here on the background thread. boom!
}
```

NSResponder Block Capture Safety

```
let label = NSTextField(labelWithString:“working on it”)

dispatch_async(dispatch_get_global_queue(0, 0)) {
   // Do some work, then bounce back to the main thread to update the UI
    dispatch_async(dispatch_get_main_queue()) {
        label.value = “done”
    }
   // The label could be deallocated here on the background thread. boom!
}
```

Panels and Workspaces

Panels

Open and save panels are now always out-of-process

Even for apps that are not sandboxed

New NSWorkspace Methods

Methods to open URL, multiple URLs, and applications

All asynchronous

Highly configurable

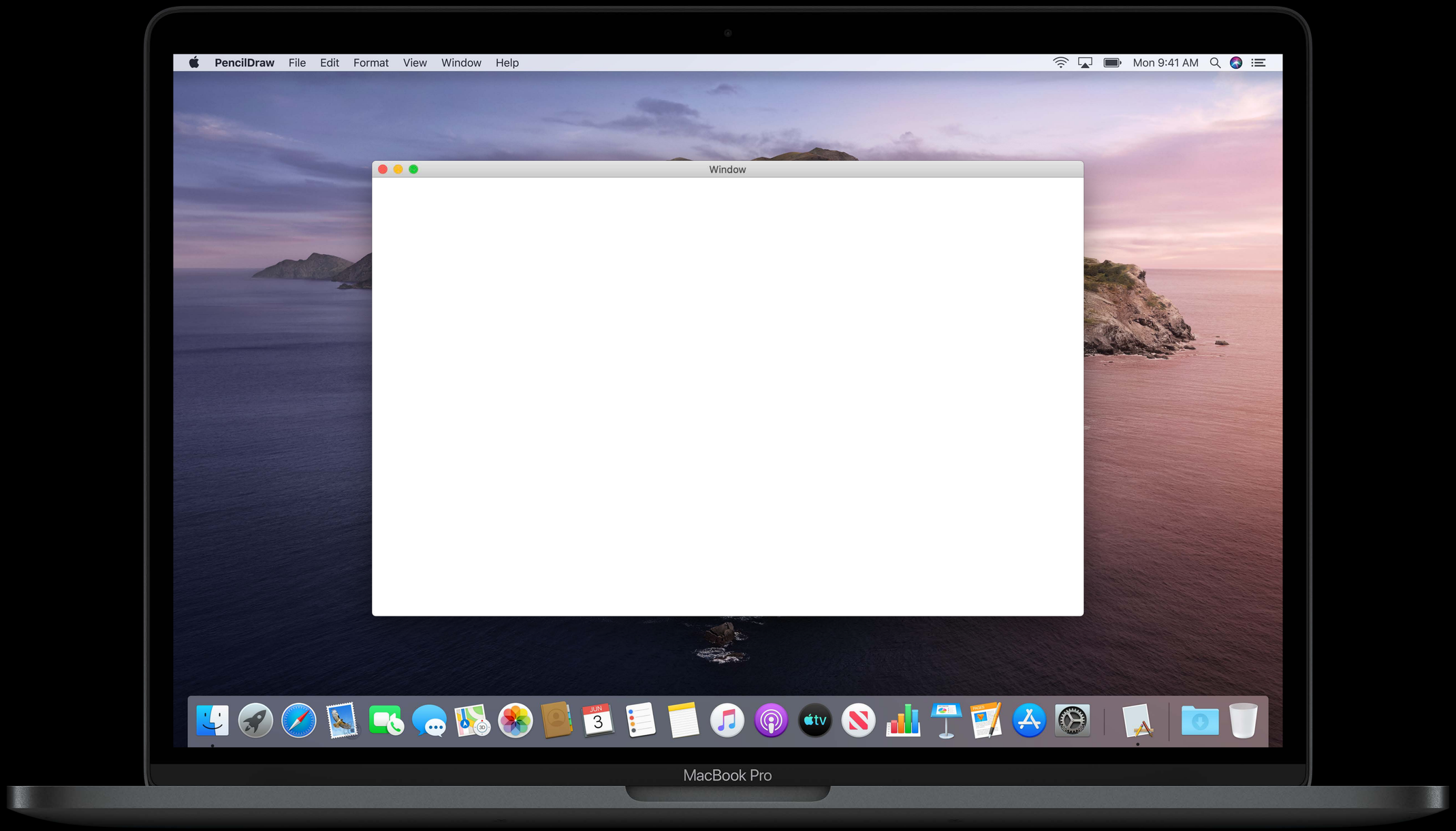
NSWorkspace.OpenConfiguration

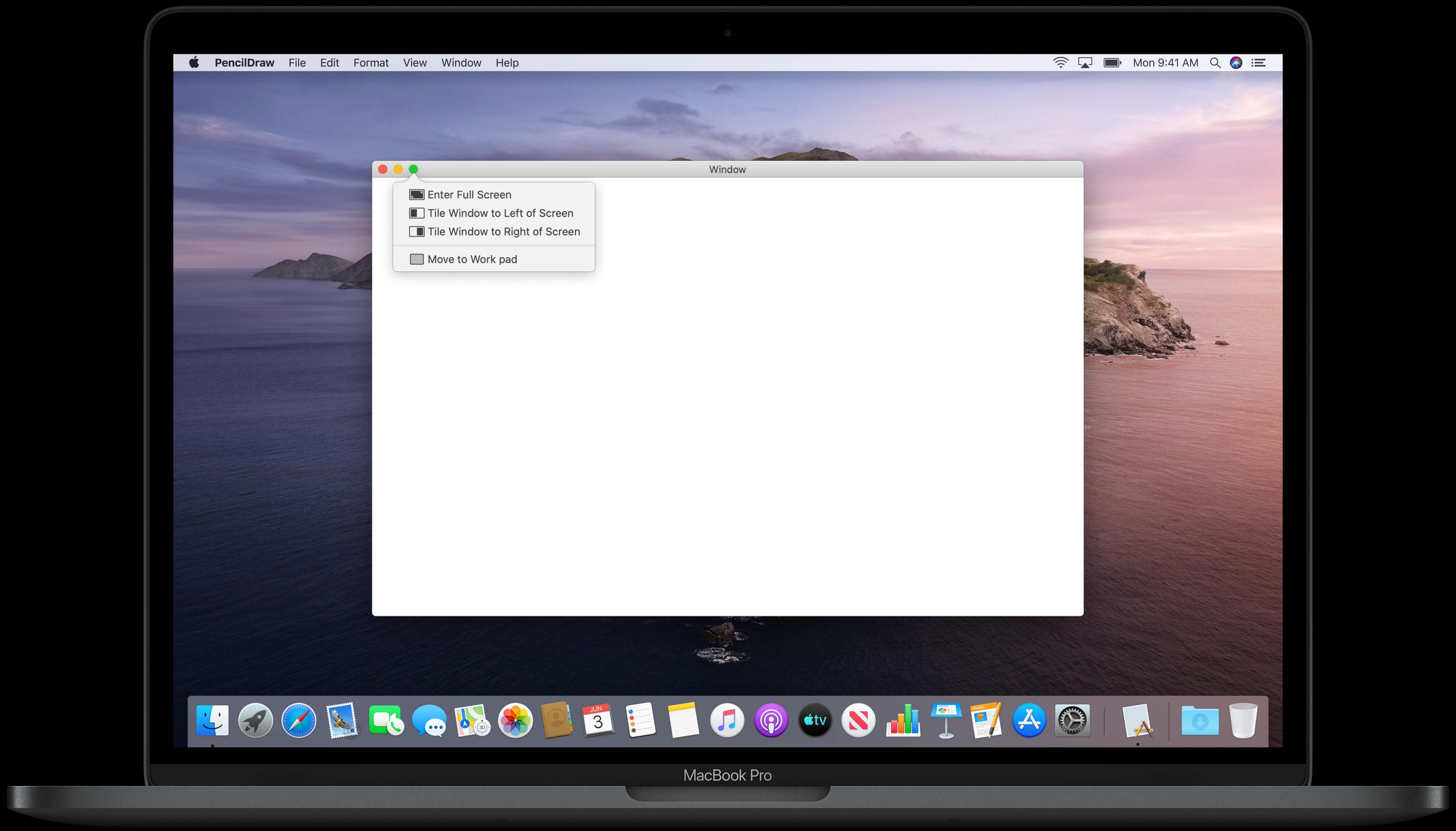
Controls user interaction

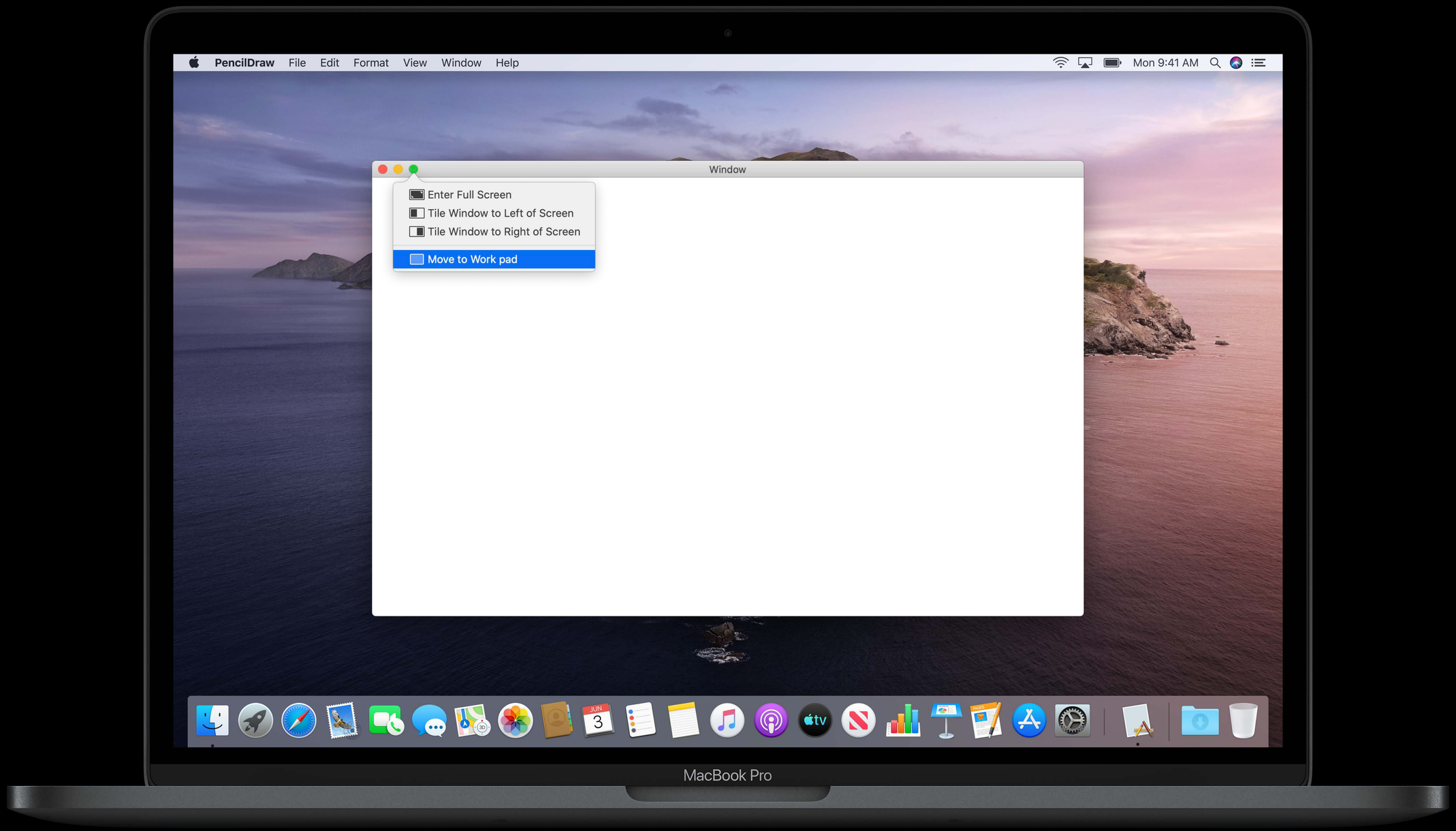
Recents menu additions

Hidden and activation states

Events


PencilDraw File Edit Format View Window Help
Mon 9:41 AM
Window
MacBook Pro


PencilDraw File Edit Format View Window Help
Mon 9:41 AM
Window
Enter Full Screen
Tile Window to Left of Screen
Tile Window to Right of Screen
Move to Work pad
MacBook Pro


PencilDraw File Edit Format View Window Help
Mon 9:41 AM
Window
Enter Full Screen
Tile Window to Left of Screen
Tile Window to Right of Screen
Move to Work pad
MacBook Pro


PencilDraw File Edit Format View Window Help
Mon 9:41 AM
Window
MacBook Pro
PencilDraw File Edit Format View Window Help
Mon 9:41 AM
Window
hello
esc

Tablet Events Review

NSEvent.SubType.tabletPoint

Pressure is not retroactively updated

New Pencil Event Types

NSEvent.EventType.changeMode

NSEvent.EventMask.changeMode

NSResponder.changeMode(withEvent:)

```
monitorID = NSEvent.addLocalMonitorForEvents(matching: .changeMode) { (event) -> NSEvent? in
    switchToNextTool()
    return event
}
```

Geometry

Geometry

NSDirectionalRectEdge

NSDirectionalEdgeInsets

NSRectAlignment

Formatters

NSRelativeDateFormatter

.dateTimeStyle (1 week ago, last week)

.unitsStyle (1 month ago, one month ago, 1 mo. ago)

NSListFormatter

.itemFormatter for formatting individual elements

localization savvy delimiter placement

Introducing Combine and Advances in Foundation Thursday, 10:00

Combine

Swift API for connecting and transforming data

Useful to connect UIs, too

```
@IBOutlet label : NSTextField
func awakeFromNib {
    model.$name.assign(to:\.stringValue, on:label)
}
```

Combine in Practice Thursday, 2:00

Extensions

New Extension Types

Non-UI file provider action extensions

New Extension Types

Replacing Kernel Extensions

- Network Extensions
- DriverKit
- Endpoint Security

System Extensions and DriverKit	WWDC 2019
Network Extensions for the Modern Mac	Friday, 9:00

Summary

New Colors and NSColorSampler

New APIs for extended dynamic range

Richer Text APIs

Collection View

Tablet Events

Geometry

Formatters

More Information

developer.apple.com/wwdc19/210


WWDC19