What’s New in Cocoa

Session 207

Ali Ozer
Daphne Larose
API Refinements
AppKit
Foundation
API Refinements

Ali Ozer, Director, Cocoa
API Refinements

Properties
Nullability
Generics
Swift naming guidelines
String enumerations
Nested types
API Refinements

Properties
Nullability
Generics
Swift naming guidelines
String enumerations
Nested types
String Enumerations and Nested Types

// NSSharingService, Swift 3

public let NSSharingServiceNameComposeEmail: String
public let NSSharingServiceNameComposeMessage: String
public let NSSharingServiceNameSendViaAirDrop: String
public let NSSharingServiceNameAddToSafariReadingList: String
...

open class NSSharingService : NSObject {
    ...
    public init?(named serviceName: String)
}

String Enumerations and Nested Types

// NSSharingService, Swift 3

public let NSSharingServiceNameComposeEmail: String
public let NSSharingServiceNameComposeMessage: String
public let NSSharingServiceNameSendViaAirDrop: String
public let NSSharingServiceNameAddToSafariReadingList: String
...

open class NSSharingService : NSObject {
    ...
    public init?(named serviceName: String)
}

String Enumerations and Nested Types

// NSSharingService, Swift 3

public let NSSharingServiceNameComposeEmail: String
public let NSSharingServiceNameComposeMessage: String
public let NSSharingServiceNameSendViaAirDrop: String
public let NSSharingServiceNameAddToSafariReadingList: String

open class NSSharingService : NSObject {
    ...
    public init?(named serviceName: String)
}
open class NSSharingService : NSObject {
...
public struct Name : RawRepresentable, Equatable, Hashable { ...
...
public init?(named serviceName: NSSharingService.Name)
}

extension NSSharingService.Name {
public static let composeEmail: NSSharingService.Name
public static let composeMessage: NSSharingService.Name
public static let sendViaAirDrop: NSSharingService.Name
public static let addToSafariReadingList: NSSharingService.Name
...
}
// NSSharingService, Swift 4

open class NSSharingService : NSObject {
  ...

  public struct Name : RawRepresentable, Equatable, Hashable {
    ...

    public init?(named serviceName: NSSharingService.Name)
  }

extension NSSharingService.Name {
  public static let composeEmail: NSSharingService.Name
  public static let composeMessage: NSSharingService.Name
  public static let sendViaAirDrop: NSSharingService.Name
  public static let addToSafariReadingList: NSSharingService.Name
  ...
}
open class NSSharingService : NSObject {
    ...
    public struct Name : RawRepresentable, Equatable, Hashable {
        ...
    }
    ...
    public init?(named serviceName: NSSharingService.Name)
}

extension NSSharingService.Name {
    public static let composeEmail: NSSharingService.Name
    public static let composeMessage: NSSharingService.Name
    public static let sendViaAirDrop: NSSharingService.Name
    public static let addToSafariReadingList: NSSharingService.Name
    ...
}
// NSSharingService, Swift 4

open class NSSharingService : NSObject {
    ...
    public struct Name : RawRepresentable, Equatable, Hashable {
        ...
    }
    public init?(named serviceName: NSSharingService.Name)
}

extension NSSharingService.Name {
    public static let composeEmail: NSSharingService.Name
    public static let composeMessage: NSSharingService.Name
    public static let sendViaAirDrop: NSSharingService.Name
    public static let addToSafariReadingList: NSSharingService.Name
    ...
}
// Swift 3
let s = NSSharingService(named: NSSharingServiceNameComposeEmail)

// Swift 4
let s = NSSharingService(named: .composeEmail)
// Swift 3
let s = NSSharingService(named: NSSharingServiceNameComposeEmail)

// Swift 4
let s = NSSharingService(named: .composeEmail)
// Swift 3
let s = NSSharingService(named: NSSharingServiceNameComposeEmail)

// Swift 4
let s = NSSharingService(named: .composeEmail)
Swift 3

```swift
let s = NSSharingService(named: NSSharingServiceNameComposeEmail)
```

Swift 4

```swift
let s = NSSharingService(named: .composeEmail)
```

```swift
import Cocoa

@NSApplicationMain
class AppDelegate: NSObject, NSApplicationDelegate {

    func applicationDidFinishLaunching(_ aNotification: Notification) {

        let s = NSSharingService(named: [])

    }

    func applicationWillTerminate(_ aNotification: Notification) {
        // Insert code here to tear down your application
    }
}
```
// Swift 3
let s = NSSharingService(named: NSSharingServiceNameComposeEmail)

// Swift 4
let s = NSSharingService(named: .composeEmail)

import Cocoa

@NSApplicationMain
class AppDelegate: NSObject, NSApplicationDelegate {

    func applicationDidFinishLaunching(_ aNotification: Notification) {
        let s = NSSharingService(named: .)
        NSSsharingService.Name addToSafariReadingList
        NSSsharingService.Name cloudSharing
        NSSsharingService.Name composeEmail
        NSSsharingService.Name composeMessage
        NSSsharingService.Name init(rawValue: String)
        NSSsharingService.Name postImageOnFlickr
        NSSsharingService.Name postOnFacebook

        Add the content to the Safari Reading List.
Source Compatibility
Objective-C
typedef NSString *NSSharingServiceName NS_EXTENSIBLE_STRING_ENUM;

APPKIT_EXTERN NSSharingServiceName const NSSharingServiceNameComposeEmail;
APPKIT_EXTERN NSSharingServiceName const NSSharingServiceNameComposeMessage;
APPKIT_EXTERN NSSharingServiceName const NSSharingServiceNameSendViaAirDrop;
APPKIT_EXTERN NSSharingServiceName const NSSharingServiceNameAddToSafariReadingList;
...
Source Compatibility
Objective-C

typedef NSString *NSSharingServiceName NS_EXTENSIBLE_STRING_ENUM;

APPKIT_EXTERN NSSharingServiceName const NSSharingServiceNameComposeEmail;
APPKIT_EXTERN NSSharingServiceName const NSSharingServiceNameComposeMessage;
APPKIT_EXTERN NSSharingServiceName const NSSharingServiceNameSendViaAirDrop;
APPKIT_EXTERN NSSharingServiceName const NSSharingServiceNameAddToSafariReadingList;
...

Source compatible
Source Compatibility
Swift
Source Compatibility
Swift

// Swift 3
public let NSSharingServiceNameComposeEmail: String
...

Source Compatibility

Swift

// Swift 3
public let NSSharingServiceNameComposeEmail: String
...

// Swift 4
public static let composeEmail: NSSharingService.Name
...
Source Compatibility
Swift

// Swift 3
public let NSSharingServiceNameComposeEmail: String
...

// Swift 4
public static let composeEmail: NSSharingService.Name
...

Source compatible in Swift 3 mode
Source Compatibility
Swift

// Swift 3
public let NSSharingServiceNameComposeEmail: String
...

// Swift 4
public static let composeEmail: NSSharingService.Name
...

Source compatible in Swift 3 mode
Migrator for moving to Swift 4
AppKit
Touch Bar
Touch Bar
Touch Bar

Escape

Application Region

Control Strip
Multi-Touch Input Device with Retina Display
Multi-Touch Input Device with Retina Display
Context Sensitive
Welcome to our WWDC present
Welcome to our WWDC present.
Welcome to our WWDC presentation
To: Paul Marcos

Subject:

From: Ali Ozer
To: Paul Marcos

Subject: Concerning my present

From: Ali Ozer
Customizable
Customizable
Customizable
Customizable
Customizable
Customizable
Customizable
Customizable
Customizable
Drag your favorite items to the bottom of the screen into the Touch Bar...
Unobstructed Access to Content
Unobstructed Access to Content

Patterns in Nature
Touch Bar Updates

NSColorPickerTouchBarItem

NSGroupTouchBarItem

Playground
Touch Bar Color Pickers
Touch Bar Color Pickers
Touch Bar Color Pickers
New Color Pickers
New Color Pickers
New Color Pickers
New Color Pickers
New Color Pickers
New Color Pickers
New Color Pickers
New Color Pickers
New Color Pickers
NSColorPickerTouchBarItem
NSColorPickerTouchBarItem

open class NSColorPickerTouchBarItem: NSTouchBarItem {
...
    open var allowedColorSpaces: [NSColorSpace]?
}

NSColorPickerTouchBarItem
NSColorPickerTouchBarItem

colorPicker.allowedColorSpaces = [.genericCMYK]
**NSColorPickerTouchBarItem**

```
colorPicker.allowedColorSpaces = [.genericCMYK]
```

```
colorPicker.allowedColorSpaces = [.genericGamma22Gray]
```
NSGroupTouchBarItem

Constructor for “alert” style
NSGroupTouchBarItem

Constructor for “alert” style
NSGroupTouchBarItem

Constructor for “alert” style

Are you sure you want to restart your computer now?
If you do nothing, the computer will restart automatically in 42 seconds.
Reopen windows when logging back in

Cancel  Restart
NSGroupTouchBarItem

Constructor for “alert” style

```swift
open class NSGroupTouchBarItem ... {

    ... 

    public init(alertStyleGroupItemWithIdentifier: NSTouchBarItem.Identifier)
}
```
NSGroupTouchBarItem

Constructor for “alert” style

Ability to specify compression options
NSGroupTouchBarItem

Constructor for “alert” style

Ability to specify compression options
NSGoupTouchBarItem

Constructor for "alert" style

Ability to specify compression options
NSGroupTouchBarItem

Constructor for "alert" style

Ability to specify compression options
NSGroupTouchBarItem

Constructor for “alert” style

Ability to specify compression options

```swift
open class NSGroupTouchBarItem ...
{
    ...
    open var prioritizedCompressionOptions: [NSUserInterfaceCompressionOptions]
}
```
NSGroupTouchBarItem

Constructor for “alert” style

Ability to specify compression options

Support for right-to-left interfaces
NSGroupTouchBarItem

Constructor for “alert” style

Ability to specify compression options

Support for right-to-left interfaces
NSGroupTouchBarItem

Constructor for “alert” style

Ability to specify compression options

Support for right-to-left interfaces
NSGroupTouchBarItem

Constructor for “alert” style

Ability to specify compression options

Support for right-to-left interfaces
NSGroupTouchBarItem

Constructor for “alert” style

Ability to specify compression options

Support for right-to-left interfaces

```swift
open class NSGroupTouchBarItem ... {
    ...
    open var groupUserInterfaceLayoutDirection: NSUserInterfaceLayoutDirection
}
```
Playground Support
import Cocoa

let button = NSButton(title: "Apply", target: nil, action: nil)
let buttonItem = NSCustomTouchBarItem(identifier: NSTouchBarItem.Identifier("applyItem")); buttonItem.view = button
buttonItem.customizationLabel = "Apply"
let touchBar = NSTouchBar()
touchBar.templateItems = [buttonItem]
touchBar.defaultItemIdentifiers = [buttonItem.identifier, .otherItemsProxy]
let colorPicker = NSColorPickerTouchBarItem(identifier: NSTouchBarItem.Identifier("myColorPicker"))

touchBar.templateItems = [colorPicker, buttonItem]
touchBar.defaultItemIdentifiers = [colorPicker.identifier, buttonItem.identifier, .otherItemsProxy]

let sliderItem = NSSliderTouchBarItem(identifier: NSTouchBarItem.Identifier("mySlider"))
sliderItem.slider.widthAnchor.constraint(equalToConstant: 100.0).isActive = true
sliderItem.minimumValueAccessory = NSSliderAccessory(image: NSImage(named: .actionTemplate))
sliderItem.maximumValueAccessory = NSSliderAccessory(image: NSImage(named: .actionTemplate))

let touchBar2 = NSTouchBar()
touchBar2.templateItems = [colorPicker, buttonItem, sliderItem]
touchBar2.defaultItemIdentifiers = [colorPicker.identifier, buttonItem.identifier, sliderItem.identifier, .otherItemsProxy]

let doneButton = NSButton(title: "Done", target: nil, action: nil)
doneButton.keyEquivalent = "\r"
doneButtonItem = NSCustomTouchBarItem(identifier: NSTouchBarItem.Identifier("done")); doneButtonItem.view = doneButton
touchBar2.templateItems = [colorPicker, buttonItem, sliderItem, doneButtonItem]
touchBar2.escapeKeyReplacementItemIdentifier = NSTouchBarItem.Identifier("done")
import Cocoa

let button = NSButton(title: "Apply", target: nil, action: nil)
let buttonItem = NSCustomTouchBarItem(identifier: NSTouchBarItem.Identifier("applyItem")); buttonItem.view = button
buttonItem.customizationLabel = "Apply"

let colorPicker = NSColorPickerTouchBarItem(identifier: NSTouchBarItem.Identifier("myColorPicker"))

touchBar.templateItems = [buttonItem]
touchBar.defaultItemIdentifiers = [buttonItem.identifier, .otherItemsProxy]

touchBar.templateItems = [colorPicker, buttonItem]
touchBar.defaultItemIdentifiers = [colorPicker.identifier, buttonItem.identifier, .otherItemsProxy]

let sliderItem = NSSliderTouchBarItem(identifier: NSTouchBarItem.Identifier("mySlider"));
sliderItem.slider.widthAnchor.constraint(equalToConstant: 100.0).isActive = true
sliderItem.minimumValueAccessory = NSSliderAccessory(image: NSImage(named: .actionTemplate))
sliderItem.maximumValueAccessory = NSSliderAccessory(image: NSImage(named: .actionTemplate))

let doneButton = NSButton(title: "Done", target: nil, action: nil)
doneButton.keyEquivalent = "\"r\"
let doneButtonItem = NSCustomTouchBarItem(identifier: NSTouchBarItem.Identifier("done")); doneButtonItem.view = doneButton
touchBar.escapeKeyReplacementItemIdentifier = NSTouchBarItem.Identifier("done")
## NSTouchBar Sessions and Lab

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Touch Bar Fundamentals</td>
<td>Grand Ballroom A</td>
<td>Wednesday 10:00AM</td>
</tr>
<tr>
<td>Advanced Touch Bar</td>
<td>Grand Ballroom A</td>
<td>Wednesday 5:10PM</td>
</tr>
<tr>
<td>Cocoa Touch Bar Lab</td>
<td>Technology Lab C</td>
<td>Thursday 9:00-11:00AM</td>
</tr>
</tbody>
</table>
Document Sharing

In Sierra, we introduced:

```swift
public static let cloudSharing: NSSharingService.Name
```
Hmm...

1. Intro
2. Middle Part
3. Conclusion
Hmm...

1. Intro
2. Middle
3. Conclusion

Add People
Choose how you'd like to send your invitation:

- Mail
- Messages
- Copy Link
- Twitter
- Facebook
- AirDrop

Share Options
Only people you invite can make changes

Demo User

Cancel
Share
Open my shared rtf: https://www.icloud.com/icloudrive/OndpwYk_28vm4vmO_HF#WWDC_Talk_Outline
Hmm...

1. Intro
2. Middle
3. Conclusion

Open my shared rtf:
https://www.icloud.com/icloudrive/0ndpwYk_26vm4vmO_hF#WWDC_Talk_Outline
Hmm...

1. Intro
2. Middle Part
3. Conclusion
Hmm...

1. Intro
2. Middle Part
3. Conclusion
Hmm...

1. Intro
2. Middle Part
3. Conclusion
Hmm...

1. Intro
2. Middle Part
3. Conclusion
Zero
Document Sharing
open class NSDocument ... {
    ...
    open var allowsDocumentSharing: Bool { get }
    open func share(with sharingService: NSSharingService,
                    completionHandler: ((Bool) -> Void)? = nil)
    open func prepare(_ sharingServicePicker: NSSharingServicePicker)
}
open class NSDocument ...
{
 ...
 open var allowsDocumentSharing: Bool { get }
 open func share(with sharingService: NSSharingService,
            completionHandler: ((Bool) -> Void)? = nil)
 open func prepare(_ sharingServicePicker: NSSharingServicePicker)
}
open class NSDocument ... {

... 

open var allowsDocumentSharing: Bool { get }

open func share(with sharingService: NSSharingService,
   completionHandler: ((Bool) -> Void)? = nil)

open func prepare(_ sharingServicePicker: NSSharingServicePicker)
}
open class NSDocument {

open var allowsDocumentSharing: Bool { get }
open func share(with sharingService: NSSharingService,
completionHandler: ((Bool) -> Void)? = nil)
open func prepare(_ sharingServicePicker: NSSharingServicePicker)
}
open class NSDocument ... {
    ...
    open var allowsDocumentSharing: Bool { get }
    open func share(with sharingService: NSSharingService,
                   completionHandler: ((Bool) -> Void)? = nil)
    open func prepare(_ sharingServicePicker: NSSharingServicePicker)
}

open class NSDocumentController ... {
    ...
    open var allowsAutomaticShareMenu: Bool { get }
    open func standardShareMenuButtonItem() -> NSMenuItem
}

Document Sharing
open class NSDocument ... {
    ...
    open var allowsDocumentSharing: Bool { get }
    open func share(with sharingService: NSSharingService,
        completionHandler: ((Bool) -> Void)? = nil)
    open func prepare(_ sharingServicePicker: NSSharingServicePicker)
}

open class NSDocumentController ... {
    ...
    open var allowsAutomaticShareMenu: Bool { get }
    open func standardShareMenuItem() -> NSMenuItem
}
open class NSDocument ... {
    ...
    open var allowsDocumentSharing: Bool { get }
    open func share(with sharingService: NSSharingService,
                   completionHandler: ((Bool) -> Void)? = nil)
    open func prepare(_ sharingServicePicker: NSSharingServicePicker)
}

open class NSDocumentController ... {
    ...
    open var allowsAutomaticShareMenu: Bool { get }
    open func standardShareMenuItem() -> NSMenuItem
}
Tabbed Windows
Tabbed Windows

A History of Lighthouses

Lighthouses are cool!

My Name
Tabbed Windows

A History of Lighthouses

Lighthouses are cool!

My Name
Tab Overview
Tab Overview

African Wildlife
Lorem ipsum dolor sit amet

[Image of a giraffe and an elephant]
Tab Overview
open class NSWindowTab : NSObject {
    open var title: String!
    @NSCopying open var attributedTitle: NSAttributedString?
    open var toolTip: String!
    open var accessoryView: NSView?
}
open class NSWindowTab : NSObject {
    open var title: String!
    @NSCopying open var attributedTitle: NSAttributedString?
    open var toolTip: String!
    open var accessoryView: NSView?
}

NSWindowTab
open class NSWindowTab : NSObject {
    open var title: String!
    @NSCopying open var attributedTitle: NSAttributedString?
    open var toolTip: String!
    open var accessoryView: NSView?
}

How to shoot on iPhone 7 -...
open class NSWindowTab : NSObject {
    open var title: String!
    @NSCopying open var attributedTitle: NSAttributedString?
    open var toolTip: String!
    open var accessoryView: NSView?
}

`NSWindowTab`
open class NSWindowTab : NSObject {
    open var title: String!
    @NSCopying open var attributedTitle: NSAttributedString?
    open var toolTip: String!
    open var accessoryView: NSView?
}

NSWindowTab
open class NSWindowTabGroup : NSObject {
    open var identifier: NSWindow.TabbingIdentifier { get }
    open var isOverviewVisible: Bool
    open var isTabBarVisible: Bool { get }

    open var windows: [NSWindow] { get }
    weak open var selectedWindow: NSWindow?

    open func addWindow(_ window: NSWindow)
    open func insertWindow(_ window: NSWindow, at index: Int)
    open func removeWindow(_ window: NSWindow)
}
open class NSWindowTabGroup : NSObject {

    open var identifier: NSWindow.TabbingIdentifier { get }
    open var isOverviewVisible: Bool
    open var isTabBarVisible: Bool { get }

    open var windows: [NSWindow] { get }
    weak open var selectedWindow: NSWindow?

    open func addWindow(_ window: NSWindow)
    open func insertWindow(_ window: NSWindow, at index: Int)
    open func removeWindow(_ window: NSWindow)
}

NSWindowTabGroup
open class NSWindowTabGroup : NSObject {
    open var identifier: NSWindow.TabbingIdentifier { get }
    open var isOverviewVisible: Bool
    open var isTabBarVisible: Bool { get }

    open var windows: [NSWindow] { get }
    weak open var selectedWindow: NSWindow?

    open func addWindow(_ window: NSWindow)
    open func insertWindow(_ window: NSWindow, at index: Int)
    open func removeWindow(_ window: NSWindow)
}

open class NSWindowTabGroup : NSObject {
    open var identifier: NSWindow.TabbingIdentifier { get }
    open var isOverviewVisible: Bool
    open var isTabBarVisible: Bool { get }

    open var windows: [NSWindow] { get }
    weak open var selectedWindow: NSWindow?

    open func addWindow(_ window: NSWindow)
    open func insertWindow(_ window: NSWindow, at index: Int)
    open func removeWindow(_ window: NSWindow)
}

NSWindowTabGroup
Opening URLs
Opening URLs

```swift
public protocol NSApplicationDelegate {
...
    optional func application(_ application: NSApplication, open urls: [URL])
}
```
NSCollectionView Prefetching
NSCollectionView
open class NSCollectionView ...
{
...
    weak open var prefetchDataSource: NSCollectionViewPrefetching?
}

NSCollectionView Prefetching
open class NSCollectionView ... {
    ...
    weak open var prefetchDataSource: NSCollectionViewPrefetching?
}

public protocol NSCollectionViewPrefetching ... {
    public func collectionView(_ collectionView: NSCollectionView,
                              prefetchItemsAt: [IndexPath])

    optional public func collectionView(_ collectionView: NSCollectionView,
                                         cancelPrefetchingForItemsAt: [IndexPath])
}

NSCollectionView Prefetching
open class NSCollectionView ... {
    ...
    weak open var prefetchDataSource: NSCollectionViewPrefetching?
}

public protocol NSCollectionViewPrefetching ... {
    public func collectionView(_ collectionView: NSCollectionView,
                              prefetchItemsAt: [IndexPath])

    optional public func collectionView(_ collectionView: NSCollectionView,
                                         cancelPrefetchingForItemsAt: [IndexPath])
}

NSCollectionView Prefetching
open class NSCollectionView ... {
...
    weak open var prefetchDataSource: NSCollectionViewPrefetching?
}

public protocol NSCollectionViewPrefetching ... {
    public func collectionView(_ collectionView: NSCollectionView,
                              prefetchItemsAt: [IndexPath])

    optional public func collectionView(_ collectionView: NSCollectionView,
                                          cancelPrefetchingForItemsAt: [IndexPath])
}

System Colors
System Colors

```swift
open class NSColor ... {
  
  open class var systemBlue: NSColor { get }
  open class var systemBrown:  NSColor { get }
  open class var systemGray:   NSColor { get }
  open class var systemGreen:  NSColor { get }
  open class var systemOrange: NSColor { get }
  open class var systemPink:   NSColor { get }
  open class var systemPurple: NSColor { get }
  open class var systemRed:    NSColor { get }
  open class var systemYellow: NSColor { get }

  }
```
open class NSColor ... {
...
    open class var systemBlue: NSColor { get }
    open class var systemBrown: NSColor { get }
    open class var systemGray: NSColor { get }
    open class var systemGreen: NSColor { get }
    open class var systemOrange: NSColor { get }
    open class var systemPink: NSColor { get }
    open class var systemPurple: NSColor { get }
    open class var systemRed: NSColor { get }
    open class var systemYellow: NSColor { get }
}
Older “Standard” Colors Now in sRGB
Older "Standard" Colors Now in sRGB

```swift
open class NSColor ... {
    open class var red: NSColor { get } /* 1.0, 0.0, 0.0 RGB */
    open class var green: NSColor { get } /* 0.0, 1.0, 0.0 RGB */
    open class var yellow: NSColor { get } /* 1.0, 1.0, 0.0 RGB */

    ...

    open class var gray: NSColor { get } /* 0.5 white */
    open class var lightGray: NSColor { get } /* 0.667 white */
    open class var white: NSColor { get } /* 1.0 white */
}
```
Older “Standard” Colors Now in sRGB

```swift
open class NSColor ... {
    open class var red: NSColor { get }  /* 1.0, 0.0, 0.0 RGB */
    open class var green: NSColor { get }  /* 0.0, 1.0, 0.0 RGB */
    open class var yellow: NSColor { get }  /* 1.0, 1.0, 0.0 RGB */
    ...
    open class var gray: NSColor { get }  /* 0.5 white */
    open class var lightGray: NSColor { get }  /* 0.667 white */
    open class var white: NSColor { get }  /* 1.0 white */
}

These now use sRGB or genericGamma22Gray
Older “Standard” Colors Now in sRGB

open class NSColor ... {
    open class var red: NSColor { get }  /* 1.0, 0.0, 0.0 RGB */
    open class var green: NSColor { get }  /* 0.0, 1.0, 0.0 RGB */
    open class var yellow: NSColor { get }  /* 1.0, 1.0, 0.0 RGB */
    ...
    open class var gray: NSColor { get }  /* 0.5 white */
    open class var lightGray: NSColor { get }  /* 0.667 white */
    open class var white: NSColor { get }  /* 1.0 white */
}

These now use sRGB or genericGamma22Gray

For apps linked against the 10.13 SDK or later
Older “Standard” Colors Now in sRGB

These now use sRGB or genericGamma22Gray

For apps linked against the 10.13 SDK or later
Older “Standard” Colors Now in sRGB

open class NSColor ... {
    open class var red: NSColor { get } /* 1.0, 0.0, 0.0 RGB */
    open class var green: NSColor { get } /* 0.0, 1.0, 0.0 RGB */
    open class var yellow: NSColor { get } /* 1.0, 1.0, 0.0 RGB */
    ...
    open class var gray: NSColor { get } /* 0.5 white */
    open class var lightGray: NSColor { get } /* 0.667 white */
    open class var white: NSColor { get } /* 1.0 white */
}

These now use sRGB or genericGamma22Gray

For apps linked against the 10.13 SDK or later
Colors in Asset Catalogs
Colors in Asset Catalogs

[Image showing a window with a color selection for the 'Sky' category]
Colors in Asset Catalogs
Colors in Asset Catalogs

open class NSColor ... {

... 

    public init?(named name: NSColor.Name)
    public init?(named name: NSColor.Name, bundle: NSBundle?)

}
NSAccessibilityCustomRotor
public protocol NSAccessibilityCustomRotor {
    ...
    func accessibilityCustomRotors() -> [NSAccessibilityCustomRotor]
}
Property Clean-Up
// 10.12
// NSWindow
unowned(unsafe) open var firstResponder: NSResponder { get }
// NSBox
unowned(unsafe) open var contentView: NSView?
...
Property Clean-Up

// 10.12
// NSWindow
unowned(unsafe) open var firstResponder: NSResponder { get }
// NSBox
unowned(unsafe) open var contentView: NSView?
...

// 10.13
// NSWindow
weak open var firstResponder: NSResponder? { get }
// NSBox
open var contentView: NSView?
...
Property Clean-Up

// 10.12
// NSWindow
unowned(unsafe) open var firstResponder: NSResponder { get }
// NSBox
unowned(unsafe) open var contentView: NSView?
...

// 10.13
// NSWindow
weak open var firstResponder: NSResponder? { get }
// NSBox
open var contentView: NSView?
...

Biggest wish for WWDC 17 “What’s new in Cocoa”? NSWindow’s firstResponder is now a weak var
4:32 PM - 17 Mar 2017
Improved handling of orphan text in NSTextField
Improved handling of orphan text in NSTextField

CGGlyph-based APIs in NSFont, NSBezierPath
Improved handling of orphan text in NSTextField

CGGlyph-based APIs in NSFont, NSBezierPath

NSFontAssetRequest for downloading fonts
Improved handling of orphan text in NSTextField

CGGlyph-based APIs in NSFont, NSBezierPath

NSFontAssetRequest for downloading fonts

Nastaliq script
Improved handling of orphan text in NSTextField
CGGlyph-based APIs in NSFont, NSBezierPath
NSFontAssetRequest for downloading fonts
Nastaliq script
Improved handling of orphan text in NSTextField
CGGlyph-based APIs in NSFont, NSBezierPath
NSFontAssetRequest for downloading fonts
Nastaliq script
Honorable Mentions
Honorable Mentions

NSSegmentedControl alignment and distribution properties
Honorable Mentions

NSSegmentedControl alignment and distribution properties

NSLevelIndicator new look and API refinements
Honorable Mentions

NSSegmentedControl alignment and distribution properties
NSLevelIndicator new look and API refinements
NSMenuItem allowsKeyEquivalentWhenHidden
Honorable Mentions

NSSegmentedControl alignment and distribution properties

NSLevelIndicator new look and API refinements

NSMenuItem allowsKeyEquivalentWhenHidden

NSTableView automatic variable row heights
Honorable Mentions

NSSegmentedControl alignment and distribution properties

NSLevelIndicator new look and API refinements

NSMenuItem allowsKeyEquivalentWhenHidden

NSTableView automatic variable row heights

Asynchronous restorable state encoding
Honorable Mentions

NSSegmentedControl alignment and distribution properties
NSLevelIndicator new look and API refinements
NSMenuItem allowsKeyEquivalentWhenHidden
NSTableView automatic variable row heights
Asynchronous restorable state encoding
Improved handling of large items during dragging
Honorable Mentions

NSSegmentedControl alignment and distribution properties
NSLevelIndicator new look and API refinements
NSMenuItem allowsKeyEquivalentWhenHidden
NSTableView automatic variable row heights
Asynchronous restorable state encoding
Improved handling of large items during dragging
Drawers - drop them
Container Views in Cocoa
Container Views in Cocoa

NSBrowser
NSTableView
NSOutlineView
NSCollectionView
NSStackView
NSGridView
Container Views in Cocoa

- NSBrowser
- NSTableView
- NSOutlineView
- NSCollectionView
- NSStackView
- NSGridView

Choosing the Right Cocoa Container View  Grand Ballroom A  Wednesday 3:10PM
I'm just trying to debug my app!
I'm just trying to debug my app!

print(obj)
I'm just trying to debug my app!
NSView and NSWindow.print() Renamed

```swift
open class NSView ... {
    ...
    open func printView(_ sender: Any?)
}

open class NSWindow ... {
    ...
    open func printWindow(_ sender: Any?)
}
```
NSView and NSWindow.print() Renamed

```swift
open class NSView ... {
    ...
    open func printView(_ sender: Any?)
}

open class NSWindow ... {
    ...
    open func printWindow(_ sender: Any?)
}
```
Tweet Your Tips!

Cocoa Development Tips

Hall 2

Friday 9:00AM
Tweet Your Tips!

Use #WWDC17 and #cocoatip

Cocoa Development Tips

Hall 2
Friday 9:00AM
What’s New in Foundation
What’s New in Foundation

Better support for key paths
What’s New in Foundation

Better support for key paths

Encoding and Decoding in Swift
Improved Key Paths
Improved Key Paths

New literal syntax
Improved Key Paths

New literal syntax

Type-safe
Improved Key Paths

New literal syntax

Type-safe

Performant
Improved Key Paths

New literal syntax

Type-safe

Performant

#keyPath(BaseType.propertyName)
Improved Key Paths

New literal syntax
Type-safe
Performant

#keyPath(BaseType.propertyName)
\BaseType.propertyName
@objc class DogOwner : NSObject {
    @objc dynamic var name: String?
    @objc dynamic var dog: Dog?
}

var daphne = DogOwner(name: "Daphne")
let observation = daphne.observe(
    
    \DogOwner.dog) {
        (observed, change) in
            print("Congrats on your new puppy \(observed.dogName)!")
    }
}
Block-based Key-Value Observing

@objc class DogOwner : NSObject {
    @objc dynamic var name: String?
    @objc dynamic var dog: Dog?
}

var daphne = DogOwner(name: "Daphne")

let observation = daphne.observe(
    \DogOwner.dog) {
    (observed, change) in
    print("Congrats on your new puppy \(observed.dogName)!")
}

Encoding and Decoding in Swift
Encoding and Decoding in Swift

Swift types ↔ JSON, Property lists, etc.
Encoding and Decoding in Swift

Swift types ↔ JSON, Property lists, etc.

Deeply customizable
Encoding and Decoding in Swift

Swift types ↔ JSON, Property lists, etc.

Deeply customizable

Easy to use
Encoding and Decoding in Swift

Swift types ↔ JSON, Property lists, etc.

Deeply customizable

Easy to use

```swift
struct Dog : Codable {
    let dogName: String
    let breed: BreedType
}
```
Encoding and Decoding in Swift

Swift types ↔ JSON, Property lists, etc.

Deeply customizable

Easy to use

```swift
struct Dog : Codable {
    let dogName: String
    let breed: BreedType
}
```
# Related Session

<table>
<thead>
<tr>
<th>What’s New in Foundation</th>
<th>Hall 2</th>
<th>Wednesday 11:00AM</th>
</tr>
</thead>
</table>
What’s New in Foundation

Hall 2
Wednesday 11:00AM
What Else is New in Foundation
What Else is New in Foundation

New and enhanced API

Available on both macOS and iOS
NSXPCConnection

Can advertise intent to publish progress
NSXPCConnection

Can advertise intent to publish progress

@protocol MyServerProtocol
-(void)performRequestWithReply:(void(^)(BOOL success))reply;
@end
NSXPCConnection

Can advertise intent to publish progress

@protocol MyServerProtocol
-(NSProgress *)performRequestWithReply:(void(^)(BOOL success))reply;
@end
NSXPCConnection

Can advertise intent to publish progress

@protocol MyServerProtocol
-(NSProgress *)performRequestWithReply:(void(^)(BOOL success))reply;
@end
NSXPCConnection

Can advertise intent to publish progress

@protocol MyServerProtocol
-(NSProgress *)performRequestWithReply:(void(^)(BOOL success))reply;
@end

Return progress before you reply
NSXPCConnection

Can advertise intent to publish progress

@protocol MyServerProtocol
-(NSProgress *)performRequestWithReply:(void(^)(BOOL success))reply;
@end

Return progress before you reply

Updates performed asynchronously
URLSession
var sessionConfig = URLSessionConfiguration.default
sessionConfig.waitsForConnectivity = true
var sessionConfig = URLSessionConfiguration.default
sessionConfig.waitsForConnectivity = true

Multipath TCP now available on iOS
Multipath TCP now available on iOS

URLSessionTask supports NSProgressReporting protocol
var sessionConfig = URLSessionConfiguration.default
sessionConfig.waitsForConnectivity = true

Multipath TCP now available on iOS

URLSessionTask supports NSProgressReporting protocol
NSFileProviderService
NSFileProviderService

Apps ↔ File Providers
NSFileProviderService

Apps ↔ File Providers

Discover file providers for any URL with NSFileManager
NSFileProviderService

Apps ↔ File Providers

Discover file providers for any URL with NSFileManager

Apps can use file provider’s specialized services
Apps ↔ File Providers

Discover file providers for any URL with NSFileManager

Apps can use file provider’s specialized services
File Provider Communication
@protocol MyServerProtocol
-(NSProgress *)performRequestWithReply:(void (^)(BOOL success))reply;
@end
File Provider Communication

App

File Provider
File Provider Communication

App ➔ performRequestWithReply: ➔ File Provider
File Provider Communication

App

File Provider
File Provider Communication

App

Returns NSProgress

File Provider
File Provider Communication

App

File Provider
File Provider Communication

App

File Provider

Downloads file
File Provider Communication

App

Updates progress

File Provider

Downloads file
File Provider Communication

- App
- File Provider
File Provider Communication

App

Download finishes

File Provider
File Provider Communication

App

Updates progress

Download finishes

File Provider
File Provider Communication

App

File Provider
File Provider Communication

App

File Provider

Replies with requested file
Improved Available Storage Space API

Check for available space more accurately

```swift
public struct URLResourceValues {
    public var volumeAvailableCapacityForImportantUsage: Int64? { get }
    public var volumeAvailableCapacityForOpportunisticUsage: Int64? { get }
}
```
Improved Available Storage Space API
Improved Available Storage Space API
Improved Available Storage Space API
Improved Available Storage Space API

- Free
- Opportunistic
- Important
Improved Available Storage Space API

Preflight check

```swift
let fileURL = URL(fileURLWithPath: "/Path/to/file")
let set: Set<URLResourceKey> = [
    URLResourceKey.volumeAvailableCapacityForOpportunisticUsageKey
]
if let fileResourceValues = try? fileURL.resourceValues(forKeys: set) {
    if (fileResourceValues.volumeAvailableCapacityForOpportunisticUsageKey > 50000000) {
        // download additional file
    }
}
```
Improved Available Storage Space API

Preflight check

```swift
let fileURL = URL(fileURLWithPath: "/Path/to/file")
let set: Set<URLResourceKey> = [URLResourceKey.volumeAvailableCapacityForOpportunisticUsageKey]
if let fileResourceValues = try? fileURL.resourceValues(forKeys: set) {
    if (fileResourceValues.volumeAvailableCapacityForOpportunisticUsageKey > 50000000) {
        // download additional file
    }
}
```
NSLinguisticTagger

Tagging a unit
NSLinguisticTagger

Tagging a unit

"Foundation is really cool. We write lots of code.\n"
Tagging a unit

"Foundation is really cool. We write lots of code."

Word
"Foundation is really cool. We write lots of code.\n"
"Foundation is really cool. We write lots of code.\n"
"Foundation is really cool. We write lots of code.\n"
NSLinguisticTagger
NSLinguisticTagger

Redone and improved
NSLinguisticTagger

Redone and improved

Common case convenience methods added
NSLinguisticTagger

Redone and improved

Common case convenience methods added

```swift
let text = "Ich weiß nicht was soll es bedeuten."
let language = NSLinguisticTagger.dominantLanguage(for: text) // "de"
```
NSLinguisticTagger

Redone and improved

Common case convenience methods added

```swift
let text = "Ich weiß nicht was soll es bedeuten."
let language = NSLinguisticTagger.dominantLanguage(for: text)  // "de"
```
JSONSerialization

Supports sorted keys
Supports sorted keys

```swift
let dict = [
    "a": 1,
    "b": 2,
    "c": 3,
    "d": 4
]
```
JSONSerialization

Supports sorted keys

```swift
let dict = \["a": 1, "b": 2, "c": 3, "d": 4\]

var data = try JSONSerialization.data(withJSONObject: dict, options: [])
// \{"b":2,"a":1,"c":3,"d":4\}
```
Supports sorted keys

```swift
let dict = 
[
    "a": 1,
    "b": 2,
    "c": 3,
    "d": 4
]

var data = try JSONSerialization.data(withJSONObject: dict, options: [])
// {"b":2,"a":1,"c":3,"d":4}

data = try JSONSerialization.data(withJSONObject: dict, options: .sortedKeys)
```
JSONSerialization

Supports sorted keys

```swift
let dict = ["a": 1, "b": 2, "c": 3, "d": 4]

var data = try JSONSerialization.data(withJSONObject: dict, options: [])
// {"b":2,"a":1,"c":3,"d":4}

data = try JSONSerialization.data(withJSONObject: dict, options: .sortedKeys)
// {"a":1,"b":3,"c":4,"d":2}
```
NSItemProvider

More explicit operations

Supports progress and cancellation

Enhance your custom classes to work with multiple representations
More explicit operations

Supports progress and cancellation

Enhance your custom classes to work with multiple representations
open class NSUserActivity : NSObject {
    open var webpageURL: URL?
}
open class NSUserActivity : NSObject {
    open var webpageURL: URL?
    open var referrerURL: URL? // Set URL of webpage that referred webpageURL
}

NSUserActivity
Failable APIs

Better error handling

Additional convenience methods that take URLs
Process Launching

// NSTask
+(nullable NSTask *)launchedTaskWithExecutableURL:(NSURL *)url
arguments:(NSArray<NSString *> *)arguments
error:(out NSError ** _Nullable)error
terminationHandler:(void(^_Nullable)(NSTask *))terminationHandler;

// Process, Swift 4
open class func run(_ url: URL,
arguments: [String],
terminationHandler: ((Process) -> Void)? = nil) throws -> Process
NSDictionary and NSArray

+(nullable NSDictionary<NSString*, ObjectType>*)dictionaryWithContentsOfURL:(NSURL *)url
    error:(NSError **)error;

+(nullable NSArray<ObjectType>*)arrayWithContentsOfURL:(NSURL *)url
    error:(NSError **)error;

-(BOOL)writeToURL:(NSURL *)url error:(NSError **)error;
NSLocalizedFailureErrorKey

Customize “what failed” while keeping “why it failed”
NSLocalizedFailureErrorKey

Customize “what failed” while keeping “why it failed”
NSLocalizedStringFailureErrorKey

Customize “what failed” while keeping “why it failed”
let greeting: String = "hello there swift"
let re = NSRegularExpression(pattern: "[sw]ift")
var found: String = ""
let greeting: String = "hello there swift"
let re = NSRegularExpression(pattern: "[sw]ift")
var found: String = ""

let nsrange = NSRange(string.indices, in: string)
for match in re.matches(in: string, range: nsrange) {
let greeting: String = "hello there swift"
let re = NSRegularExpression(pattern: "[sw]ift")
var found: String = ""

let nsrange = NSRange(string.indices, in: string)
for match in re.matches(in: string, range: nsrange) {

let greeting: String = "hello there swift"
let re = NSRegularExpression(pattern: "[sw]ift")
var found: String = ""

let nsrange = NSRange(string.indices, in: string)
for match in re.matches(in: string, range: nsrange) {
  // Do something with the match
}
let greeting: String = "hello there swift"
let re = NSRegularExpression(pattern: "[sw]ift")
var found: String = ""

let nsrange = NSRange(string.indices, in: string)
for match in re.matches(in: string, range: nsrange) {
""
let greeting: String = "hello there swift"
let re = NSRegularExpression(pattern: "[sw]ift")
var found: String = ""

let nsrange = NSRange(string.indices, in: string)
for match in re.matches(in: string, range: nsrange) {

let greeting: String = "hello there swift"
let re = NSRegularExpression(pattern: "[sw]ift")
var found: String = ""

let nsrange = NSRange(string.indices, in: string)
for match in re.matches(in: string, range: nsrange) {
    found.append(Range(match.rangeAt(1), in: string))
}

Performance Improvements
Performance Improvements

Copy-on-write NSArray, NSDictionary, NSSet
Performance Improvements

Copy-on-write NSArray, NSDictionary, NSSet

struct Data inlining
Performance Improvements

Copy-on-write NSArray, NSDictionary, NSSet

struct Data inlining

Faster calendrical calculations with lower peak memory
Performance Improvements

Copy-on-write NSArray, NSDictionary, NSSet

struct Data inlining

Faster calendrical calculations with lower peak memory

Faster bridging of NSNumber to and from Swift
Performance Improvements

Copy-on-write NSArray, NSDictionary, NSSet

struct Data inlining

Faster calendrical calculations with lower peak memory

Faster bridging of NSNumber to and from Swift
NSArchiver and NSUnarchiver

NSArchiver and NSUnarchiver replaced in 10.4

Now formally deprecated

Old formats still supported

Use NSKeyedArchiver 🙏
Additional Mentions
Additional Mentions

Core Data highlights
Additional Mentions

Core Data highlights

Core Spotlight available on macOS
Additional Mentions

Core Data highlights

Core Spotlight available on macOS

Thermal Notifications available on iOS
Additional Mentions

Core Data highlights

Core Spotlight available on macOS

Thermal Notifications available on iOS

<table>
<thead>
<tr>
<th>Event</th>
<th>Location</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s New in Cocoa Touch</td>
<td>Hall 3</td>
<td>Tuesday 10:20AM</td>
</tr>
<tr>
<td>What’s New in Core Data</td>
<td>Grand Ballroom B</td>
<td>Wednesday 10:00AM</td>
</tr>
<tr>
<td>What’s New in Core Spotlight for iOS and macOS</td>
<td>Grand Ballroom B</td>
<td>Thursday 4:10PM</td>
</tr>
</tbody>
</table>
Wrap-Up

API Refinements

AppKit

Foundation
Wrap-Up

API Refinements

AppKit

Foundation

Release Notes
More Information

Reminder
Tweet your Cocoa development tips!

Use #WWDC17 and #cocoatip

Cocoa Development Tips

Hall 2
Friday 9:00AM
## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s New in Swift</td>
<td>Hall 3</td>
<td>Tuesday 1:50PM</td>
</tr>
<tr>
<td>Natural Language Processing and your Apps</td>
<td>Hall 3</td>
<td>Wednesday 9:00AM</td>
</tr>
<tr>
<td>What’s New in Core Data</td>
<td>Grand Ballroom B</td>
<td>Wednesday 10:00AM</td>
</tr>
<tr>
<td>Touch Bar Fundamentals</td>
<td>Grand Ballroom A</td>
<td>Wednesday 10:00AM</td>
</tr>
<tr>
<td>What’s New in Foundation</td>
<td>Hall 2</td>
<td>Wednesday 11:00AM</td>
</tr>
<tr>
<td>Choosing the Right Cocoa Container View</td>
<td>Grand Ballroom A</td>
<td>Wednesday 3:10PM</td>
</tr>
<tr>
<td>Advances in Networking, Part 2</td>
<td>Executive Ballroom</td>
<td>Wednesday 4:10PM</td>
</tr>
<tr>
<td>Session</td>
<td>Location</td>
<td>Date</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>----------------</td>
<td>------------</td>
</tr>
<tr>
<td>Advanced Touch Bar</td>
<td>Grand Ballroom A</td>
<td>Wednesday</td>
</tr>
<tr>
<td>What’s New in Core Spotlight for iOS and macOS</td>
<td>Grand Ballroom B</td>
<td>Thursday</td>
</tr>
<tr>
<td>Cocoa Development Tips</td>
<td>Hall 2</td>
<td>Friday</td>
</tr>
<tr>
<td>File Provider Enhancements</td>
<td>Hall 3</td>
<td>Friday</td>
</tr>
<tr>
<td>Efficient Interactions with Frameworks</td>
<td>Hall 2</td>
<td>Friday</td>
</tr>
<tr>
<td>Lab</td>
<td>Location</td>
<td>Time</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Cocoa Lab</td>
<td>Technology Lab C</td>
<td>Wed 11:00AM-1:00PM</td>
</tr>
<tr>
<td>Foundation Lab</td>
<td>Technology Lab C</td>
<td>Wed 1:00-3:10PM</td>
</tr>
<tr>
<td>Core Data Lab</td>
<td>Technology Lab C</td>
<td>Wed 3:10-5:30PM</td>
</tr>
<tr>
<td>Cocoa Touch Bar Lab</td>
<td>Technology Lab C</td>
<td>Thu 9:00-11:00AM</td>
</tr>
<tr>
<td>Text and Fonts Lab</td>
<td>Technology Lab H</td>
<td>Thu 1:50-3:50PM</td>
</tr>
<tr>
<td>Core Data Lab</td>
<td>Technology Lab H</td>
<td>Thu 4:10-6:00PM</td>
</tr>
<tr>
<td>Cocoa Lab</td>
<td>Technology Lab B</td>
<td>Fri 1:50-3:30PM</td>
</tr>
</tbody>
</table>