What’s New in Cocoa Touch

Session 205

Olivier Gutknecht  Senior Engineering Manager, UIKit
What’s New in Cocoa Touch
What’s New in Cocoa Touch
What’s Not New in Cocoa Touch
Since Last Year
Since Last Year

3D Touch
Since Last Year

3D Touch
iPad Pro
Since Last Year

3D Touch
iPad Pro
Apple Pencil
Since Last Year

3D Touch
iPad Pro
Apple Pencil
Smart Keyboard
Adaptivity
Adaptivity

UIUserInterfaceSizeClass.compact
Adaptivity

UIUserInterfaceSizeClass.regular
Adaptivity

UIUserInterfaceSizeClass.gigantic
Adaptivity

UIUserInterfaceSizeClass.gigantic
Adaptivity

UIUserInterfaceSizeClass.gigantic
Adaptivity

UIUserInterfaceSizeClass.gigantic
Making Apps Adaptive
The fundamentals

Traits
Size Classes
Size-based layouts
Making Apps Adaptive

Interface Builder
Making Apps Adaptive

Tools and Techniques

Auto Layout
Dynamic Type
Layout Guides
UIAppearance
Asset Catalogs
Making Apps Adaptive

Making Apps Adaptive, Part I
Presidio
Thursday 11:00AM

Making Apps Adaptive, Part II
Presidio
Friday 9:00AM

Inclusive App Design
Pacific Heights
Tuesday 10:00AM
Advanced Touch Input
Advanced Touch Input

120 Hz touch scanning on iPad Air 2 and iPad Pro
Advanced Touch Input

120 Hz touch scanning on iPad Air 2 and iPad Pro
Orientation, Precise Location, Force, and 240 Hz scanning with Apple Pencil
Advanced Touch Input

120 Hz touch scanning on iPad Air 2 and iPad Pro

Orientation, Precise Location, Force, and 240 Hz scanning with Apple Pencil

Force on iPhone with 3D Touch
Advanced Touch Input

120 Hz touch scanning on iPad Air 2 and iPad Pro
Orientation, Precise Location, Force, and 240 Hz scanning with Apple Pencil
Force on iPhone with 3D Touch
New APIs in iOS 9 & iOS 9.1
Advanced Touch Input

120 Hz touch scanning on iPad Air 2 and iPad Pro
Orientation, Precise Location, Force, and 240 Hz scanning with Apple Pencil
Force on iPhone with 3D Touch
New APIs in iOS 9 & iOS 9.1

Leveraging Touch Input on iOS
Keyboard Support
Keyboard Support

Keyboard shortcuts
Keyboard Support

Keyboard shortcuts
Dynamic, context-sensitive
override var keyCommands: [UIKeyCommand]? {
    return [
        UIKeyCommand(input: "F",
                      modifierFlags: .Command,
                      action: #selector(ViewController.find(_:)),
                      discoverabilityTitle: "Find...")],

        UIKeyCommand(input: "N",
                      modifierFlags: [.Command, .Alternate],
                      action: #selector(ViewController.newDocument(_:)),
                      discoverabilityTitle: "New document"),
    ]
}

func find(sender: UIKeyCommand) {
    // ...
}
// Keyboard Support: UIKeyCommand

override var keyCommands: [UIKeyCommand]? {
    return [
        UIKeyCommand(input: "F",
            modifierFlags: .Command,
            action: #selector(ViewController.find(_:)),
            discoverabilityTitle: "Find…"),

        UIKeyCommand(input: "N",
            modifierFlags: [.Command, .Alternate],
            action: #selector(ViewController.newDocument(_:)),
            discoverabilityTitle: "New document"),
    ]
}

func find(sender: UIKeyCommand) {
    // ...
}
Easy to use. Even easier to take with you.
What’s Not New in Cocoa Touch
What’s New in Cocoa Touch
What's New in Cocoa Touch
Agenda
Core technologies
Agenda

Core technologies
Building better user interfaces
Agenda

Core technologies
Building better user interfaces
Adopting system features
Agenda

Core technologies
Building better user interfaces
Adopting system features
Integrating with iOS
Core Technologies
Swift 3
let font = UIFont.preferredFontForTextStyle(UIFontTextStyleBody)
let color = UIColor.blackColor()
let title = content.stringByTrimmingCharactersInSet(.whitespaceAndNewlineCharacterSet())

let transform = CGAffineTransformRotate(baseTransform, angle)
CGContextConcatCTM(context, transform)

title.drawAtPoint(position, withAttributes: attributes)

let queue = dispatch_queue_create("com.example.queue", nil)
dispatch_async(queue) {
    // ...
}

// ...
let font = UIFont.preferredFontForTextStyle(UIFontTextStyleBody)
let color = UIColor.blackColor()
let title = content.stringByTrimmingCharactersInSet(.whitespaceAndNewlineCharacterSet())

let transform = CGAffineTransformRotate(baseTransform, angle)
CGContextConcatCTM(context, transform)
title.drawAtPoint(position, withAttributes: attributes)

let queue = dispatch_queue_create("com.example.queue", nil)
dispatch_async(queue) {
    // ...
}
let font = UIFont.preferredFont(forTextStyle: UIFontTextStyleBody)
let color = UIColor.blackColor()
let title = content.stringByTrimmingCharactersInSet(.whitespaceAndNewlineCharacterSet())

let transform = CGAffineTransformRotate(baseTransform, angle)
CGContextConcatCTM(context, transform)
title.draw(at: position, withAttributes: attributes)

let queue = dispatch_queue_create("com.example.queue", nil)
dispatch_async(queue) {
    // ...
}

let font = UIFont.preferredFont(forTextStyle: UIFontTextStyleBody)
let color = UIColor.blackColor()
let title = content.stringByTrimmingCharactersInSet(.whitespaceAndNewlineCharacterSet())

let transform = CGAffineTransformRotate(baseTransform, angle)
CGContextConcatCTM(context, transform)
title.draw(at: position, withAttributes: attributes)

let queue = dispatch_queue_create("com.example.queue", nil)
dispatch_async(queue) {
    // ...
}

let font = UIFont.preferredFont(forTextStyle: UIFontTextStyleBody)
let color = UIColor.black()
let title = content.trimmingCharacters(in: .whitespaceAndNewline())

let transform = CGAffineTransformRotate(baseTransform, angle)
CGContextConcatCTM(context, transform)
title.draw(at: position, withAttributes: attributes)

let queue = dispatch_queue_create("com.example.queue", nil)
dispatch_async(queue) {
    // ...
}
let font = UIFont.preferredFont(forTextStyle: UIFontTextStyleBody)
let color = UIColor.black()
let title = content.trimmingCharacters(in: .whitespaceAndNewline())

let transform = CGAffineTransformRotate(baseTransform, angle)
CGContextConcatCTM(context, transform)

title.draw(at: position, withAttributes: attributes)

let queue = dispatch_queue_create("com.example.queue", nil)
dispatch_async(queue) {
    // ...
}

// ...
let font = UIFont.preferredFont(forTextStyle: UIFontTextStyleBody)
let color = UIColor.black()
let title = content.trimmingCharacters(in: .whitespaceAndNewline())

let transform = baseTransform.rotate(angle)
context.concatCTM(transform)

title.draw(at: position, withAttributes: attributes)

let queue = dispatch_queue_create("com.example.queue", nil)
dispatch_async(queue) {
    // ...
}

let font = UIFont.preferredFont(forTextStyle: UIFontTextStyleBody)
let color = UIColor.black()
let title = content.trimmingCharacters(in: .whitespaceAndNewline())

let transform = baseTransform.rotate(angle)
context.concatCTM(transform)
title.draw(at: position, withAttributes: attributes)

let queue = dispatch_queue_create("com.example.queue")
dispatch_async(queue) {
    // ...
}

// ...
let font = UIFont.preferredFont(forTextStyle: UIFontTextStyleBody)
let color = UIColor.black()
let title = content.trimmingCharacters(in: .whitespaceAndNewline())

let transform = baseTransform.rotate(angle)
context.concatCTM(transform)

title.draw(at: position, withAttributes: attributes)

let queue = DispatchQueue(label: "com.example.queue")
queue.async {
    // ...
}

// ...
Grand Central Dispatch
Grand Central Dispatch

Create a private queue
Grand Central Dispatch

Create a private queue
Schedule asynchronous work items
Grand Central Dispatch

Create a private queue
Schedule asynchronous work items
GCD can automatically wrap each work item in an autorelease pool
Grand Central Dispatch

Create a private queue

Schedule asynchronous work items

GCD can automatically wrap each work item in an autorelease pool

```swift
let q = DispatchQueue(label: "com.example.queue", attributes: [.autoreleaseWorkItem])
```
Grand Central Dispatch

Create a private queue
Schedule asynchronous work items
GCD can automatically wrap each work item in an autorelease pool

```swift
let q = DispatchQueue(label: "com.example.queue", attributes: [.autoreleaseWorkItem])
```
Foundation
Foundation

Swift improvements
Foundation

Swift improvements

Units and measurements
Foundation

Swift improvements
Units and measurements
NSISO8601DateFormatter
Foundation

Swift improvements
Units and measurements
NSISO8601 DateFormatter
NSDateInterval
Foundation

Swift improvements
Units and measurements
NSISO8601DateFormatter
NSDateInterval

What’s New in Foundation for Swift

<table>
<thead>
<tr>
<th>Measurements and Units</th>
<th>Mission</th>
<th>Tuesday 4:00PM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Presidio</td>
<td>Friday 4:00PM</td>
</tr>
</tbody>
</table>
UIPasteboard
Universal Clipboard
UIPasteboard
Universal Clipboard
UIPasteboard
Universal Clipboard

A paste operation might have to retrieve remote data
UIPasteboard
Universal Clipboard

A paste operation might have to retrieve remote data
A paste operation might have to retrieve remote data
Check for Pasteboard content without fetching
UIPasteboard
Universal Clipboard

A paste operation might have to retrieve remote data
Check for Pasteboard content without fetching

```swift
public class UIPasteboard : NSObject {

    public var hasStrings: Bool { get }
    public var hasURLs: Bool { get }
    public var hasImages: Bool { get }
    public var hasColors: Bool { get }
```
UIPasteboard

Universal Clipboard

Control what you publish

extension UIPasteboardOption {

    public static let expirationDate: UIPasteboardOption
    public static let localOnly: UIPasteboardOption

}
Wide Color

Technology shift
Wide Color

Technology shift
Wide Color

Technology shift
From sRGB to extended sRGB
Wide Color

Technology shift
From sRGB to extended sRGB
iOS 9.3 is color managed!
Wide Color

Technology shift
From sRGB to extended sRGB
iOS 9.3 is color managed!
Exposed as API in iOS 10.0
Wide Color

UIImageView, color-managed since iOS 9.3
Wide Color

UIColor Support
public class UIColor : NSObject {

    public init(red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)

    public init(displayP3Red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)

}
Wide Color

UIColor Support

Go beyond [0-1] for extended sRGB with the existing initializer

```swift
public class UIColor : NSObject {
    public init(red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)
    public init(displayP3Red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)
```
Wide Color

UIColor Support

Go beyond [0-1] for extended sRGB with the existing initializer
Use displayP3 for content creation and interchange

```swift
public class UIColor : NSObject {
    public init(red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)
    public init(displayP3Red: CGFloat, green: CGFloat, blue: CGFloat, alpha: CGFloat)
```
Wide Color

UIColor Support

Go beyond [0-1] for extended sRGB with the existing initializer
Use displayP3 for content creation and interchange

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

Working with Wide Color

Mission

Thursday 1:25PM
Image Renderer

UIKitBeginImageContext and UIGraphicsEndImageContext
Image Renderer

- `UIGraphicsBeginImageContext` and `UIGraphicsEndImageContext`
- 32 bits and sRGB only
Image Renderer

- `UIGraphicsBeginImageContext` and `UIGraphicsEndImageContext`
  - 32 bits and sRGB only
  - Error prone
Image Renderer

- `UIGraphicsBeginImageContext` and `UIGraphicsEndImageContext`
- 32 bits and sRGB only
- Error prone
- Not extensible
Image Renderer

- UIGraphicsBeginImageContext and UIGraphicsEndImageContext
  - 32 bits and sRGB only
  - Error prone
  - Not extensible

```swift
func createDrawing(size: CGSize) -> UIImage {
    let renderer = UIGraphicsBeginImageContext(size)
    // Do your drawing here
    let image = UIGraphicsGetImageFromCurrentImageContext()
    UIGraphicsEndImageContext()
    return image
}
```
Image Renderer
Image Renderer

New `UIKitRenderer` class
Image Renderer

New `UIGraphicsRenderer` class

- Fully color managed
Image Renderer

New **UIGraphicsRenderer** class

- Fully color managed
- Block-based
New **UIGraphicsRenderer** class

- Fully color managed
- Block-based
- Subclasses for images and PDF
New **UIGraphicsRenderer** class

- Fully color managed
- Block-based
- Subclasses for images and PDF
- Manages context lifetime
New **UIGraphicsRenderer** class

- Fully color managed
- Block-based
- Subclasses for images and PDF
- Manages context lifetime

```swift
func createDrawing(size: CGSize) -> UIImage {
    let renderer = UIGraphicsImageRenderer(size: size)
    return renderer.image { rendererContext in
        // Do your drawing here
    }
}
```
Asset Management

Wide color assets
Directional image assets
Compression

Integrated with the UIKit trait system
Asset Management

Wide color assets
• Automatic variants generation
• Compatible with App Thinning
Asset Management

Compression
• Now supports automatic lossy compression
• Great compromise between footprint and quality
• Automatic variant will find the right tradeoff
• Compatible with App Thinning
Asset Management

Directional image assets for right-to-left and left-to-right UIs

• Specify if an image should be flipped or not
• Provide specific images

What's New in International User Interfaces

Nob Hil
Friday 9:00AM
Building Better User Interfaces
Accessibility Inspector
Accessibility Inspector

- **Dynamic Text unsupported**: This element does not support dynamic text. Consider using a dynamic text font.
- **Label duplicates traits**: The accessibilityLabel of this element duplicates information that is available in the traits.
- **Element has no description**: This element is missing useful accessibility information.

Accessibility Inspector window with warnings shown on right.
Accessibility Inspector

- Dynamic Text unsupported
  This element does not support dynamic text. Consider using a dynamic text font

- Label duplicates traits
  The accessibilityLabel of this element duplicates information that is available in the traits.

- Element has no description
  This element is missing useful accessibility information.
We added a new speech recognizer
We added a new speech recognizer
We added a new speech recognizer

Il fonctionne même avec d'autres langues que l'anglais (*)
We added a new speech recognizer

Il fonctionne même avec d'autres langues que l'anglais (*)
Speech Recognition
SFSpeechRecognizer
Speech Recognition

SFSpeechRecognizer

Continuous speech recognition
Speech Recognition
SFSpeechRecognizer

Continuous speech recognition
From audio files or audio buffers
Speech Recognition
SFSpeechRecognizer

Continuous speech recognition
From audio files or audio buffers
Optimized for free-form dictation or search-style strings
Speech Recognition
SFSpeechRecognizer

Continuous speech recognition
From audio files or audio buffers
Optimized for free-form dictation or search-style strings

```swift
let recognizer = SFSpeechRecognizer()
let request = SFSpeechURLRecognitionRequest(url: audioFileURL)

recognizer?.recognitionTask(with: request, resultHandler: { (result, error) in
    print(result?.bestTranscription.formattedString)
})
```
Speech Recognition
SFSpeechRecognizer

Continuous speech recognition
From audio files or audio buffers
Optimized for free-form dictation or search-style strings

```swift
let recognizer = SFSpeechRecognizer()
let request = SFSpeechURLRecognitionRequest(url: audioFileURL)

recognizer?.recognitionTask(with: request, resultHandler: { (result, error) in
    print(result?.bestTranscription.formattedString)
})
```
Smarter Text Input
Smarter Text Input

Semantic tagging of text fields, text views and web content
Smarter Text Input

Semantic tagging of text fields, text views and web content
Provides intelligent suggestions
Smarter Text Input

Semantic tagging of text fields, text views and web content
Provides intelligent suggestions
Many predefined content type
Smarter Text Input

Semantic tagging of text fields, text views and web content
Provides intelligent suggestions
Many predefined content type
  People
Smarter Text Input

Semantic tagging of text fields, text views and web content
Provides intelligent suggestions
Many predefined content type
  People
  Locations

```objective-c
textField.textContentType = UITextContentTypeFullStreetAddress
```
Smarter Text Input

Semantic tagging of text fields, text views and web content
Provides intelligent suggestions
Many predefined content type
  People
  Locations

textField.textContentType = UITextContentTypeFullStreetAddress

Email, telephone, credit card number, …
Dynamic Type
Content size category trait
Dynamic Type
Content size category trait

No longer a property on UIApplication
Dynamic Type
Content size category trait

No longer a property on UIApplication
No need to listen to notifications
Dynamic Type
Content size category trait

No longer a property on `UIApplication`
No need to listen to notifications
Now available in extensions
Dynamic Type

Apps that support Dynamic Type will adjust to your preferred reading size below.

Drag the slider below

What's New in Cocoa Touch

The UIKit framework provides the crucial infrastructure needed to construct and manage iOS apps, and UIKit continues to advance the state of the art for app design with many new features. Dive straight into new features across the iOS frameworks that allow your apps to take advantage of many new system services, such as to offer services to other apps. This is your first stop to discover the many sessions to see about harnessing the power of iOS 10.
Dynamic Type

Apps that support Dynamic Type will adjust to your preferred reading size below.

Drag the slider below

Larger sizes are available in Accessibility Settings.

What’s New in Cocoa Touch

The UIKit framework provides the crucial infrastructure needed to construct and manage iOS apps, and UIKit continues to advance the state of the art for app design with many new features. Dive straight into new features across the iOS frameworks that allow your apps to take advantage of many new system services, as well as to offer services to other apps. This is your first stop to discover the many sessions to see about harnessing the power of iOS 10.
Dynamic Type

Apps that support Dynamic Type will adjust to your preferred reading size below.

Drag the slider below

What's New in Cocoa Touch
The UIKit framework provides the crucial infrastructure needed to construct and manage iOS apps, and UIKit continues to advance the state of the art for app design with many new features. Dive straight into new features across the iOS frameworks that allow your apps to take advantage of many new system services, as well as to offer services to other apps. This is your first stop to discover the many sessions to see about harnessing the power of iOS 10.

```swift
label.font = UIFont.preferredFont(forTextStyle: UIFontTextStyleBody)
label.adjustsFontForContentSizeCategory = true
```
Dynamic Type

Automatic support in label, text views and controls

label.font = UIFont.preferredFont(forTextStyle: UIFontTextStyleBody)
label.adjustsFontForContentSizeCategory = true
Improved Customization

Tab bar items
Improved Customization

Tab bar items

Custom badge colors and text attributes
Customizable unselected Tint Color
Improved Customization

Tab bar items

Custom badge colors and text attributes

Customizable unselected Tint Color
Improved Customization

Tab bar items

Custom badge colors and text attributes

Customizable unselected Tint Color

```swift
tabBarItem.badgeColor = UIColor.white()

badgeTextAttributes = [ 
    NSKForegroundColorAttributeName : UIColor.blue(),
    NSKFontAttributeName : UIFont.italicSystemFont(ofSize: 12)
]

tabBarItem.setBadgeTextAttributes(textAttributes: badgeTextAttributes, 
forState: UIControlStateNormal)

tabBar.unselectedTintColor = UIColor.brown()
```
Improved Customization

Tab bar items

Custom badge colors and text attributes

Customizable unselected Tint Color

```swift
tabBarItem.badgeColor = UIColor.white()

badgeTextAttributes = [NSForegroundColorAttributeName : UIColor.blue(),
                       NSFontAttributeName : UIFont.italicSystemFont(ofSize: 12)]

tabBarItem.setBadgeTextAttributes(textAttributes: badgeTextAttributes,
                                  forState: UIControlStateNormal)

tabBar.unselectedTintColor = UIColor.brown()
```
Peek & Pop

Improved WKWebView Support
Peek & Pop

Improved WKWebView Support

Fine control of Peek & Pop behaviors
Peek & Pop

Improved WKWebView Support

Fine control of Peek & Pop behaviors
Custom view controllers
Peek & Pop

Improved WKWebView Support

Fine control of Peek & Pop behaviors
Custom view controllers
Preview actions
Peek & Pop

Improved WKWebView Support

Fine control of Peek & Pop behaviors
Custom view controllers
Preview actions
Pop inside your app
Peek & Pop

Improved WKWebView Support

Fine control of Peek & Pop behaviors
Custom view controllers
Preview actions
Pop inside your app

```swift
public func webView(_ webView: WKWebView,
shouldPreviewElement elementInfo: WKPreviewElementInfo) -> Bool

public func webView(_ webView: WKWebView,
previewingViewControllerForElement elementInfo: WKPreviewElementInfo,
defaultActions previewActions: [WKPreviewActionItem]) -> UIViewController?

public func webView(_ webView: WKWebView,
commitPreviewingViewController previewingViewController: UIViewController)
```
Peek & Pop

Bring your own UI!
Peek & Pop

Bring your own UI!

UIPreviewInteraction
Peek & Pop
Bring your own UI!

UIKit provides the “feel” of Peek & Pop
Bring your own UI!

UIKit provides the “feel” of Peek & Pop

```swift
func previewInteraction(_ previewInteraction: UIPreviewInteraction,
didUpdatePreviewTransition transitionProgress: CGFloat, ended: Bool) {

    self.updateUIToPeek(transitionProgress)
    if ended {
        self.showPeekUI()
    }
}
```
Bring your own UI!

UIKit provides the “feel” of Peek & Pop

```swift
func previewInteraction(_ previewInteraction: UIPreviewInteraction,
   didUpdatePreviewTransition transitionProgress: CGFloat, ended: Bool) {

    self.updateUIToPeek(transitionProgress)
    if ended {
        self.showPeekUI()
    }
}
```
What's New in Scroll Views?
What's New in Scroll Views?
Refresh Control
Support for UIScrollView and subclasses
Collection View
Collection View

Automatic self-sizing cells in flow layout
Collection View

Automatic self-sizing cells in flow layout
Paging support in collection view reordering
Collection View
Smooth scrolling
Collection View

Smooth scrolling

Cell prefetching
Collection View

Smooth scrolling

Cell prefetching

Data prefetching
Collection View
Smooth scrolling

Cell prefetching
Data prefetching
(also available in UITableView)
Collection View

Smooth scrolling

Cell prefetching

Data prefetching

(also available in UITableView)
Advances in UIKit Animations

UIViewPropertyAnimator
Advances in UIKit Animations

UIViewPropertyAnimator

Interruptible
Advances in UIKit Animations

UIViewPropertyAnimator

Interruptible
Scrubbable
Advances in UIKit Animations

UIViewPropertyAnimator

- Interruptible
- Scrubbable
- Reversible
Advances in UIKit Animations

UIViewPropertyAnimator

Interruptible
Scrubbable
Reversible
Rich timing features
Advances in UIKit Animations

UIViewPropertyAnimator

Interruptible
Scrubbable
Reversible
Rich timing features
Dynamic
let timing = UICubicTimingParameters(animationCurve: .easeInOut)
let animator = UIViewPropertyAnimator(duration: duration, timingParameters: timing)

animator.addAnimations {
    self.squareView.center = CGPoint(x: point.x, y: point.y)
}

animator.startAnimation()
let timing = UICubicTimingParameters(animationCurve: .easeInOut)
let animator = UIViewPropertyAnimator(duration: duration, timingParameters: timing)

animator.addAnimations {
    self.squareView.center = CGPoint(x: point.x, y: point.y)
}

animator.startAnimation()
Adopting System Features
Opening applications
Opening applications

Improving openURL
Opening applications

Improving openURL
• Asynchronous, with a completion handler
Opening applications

Improving openURL

- Asynchronous, with a completion handler
- Let you check if a handler app is installed for universal links
Opening applications

Improving openURL

• Asynchronous, with a completion handler
• Let you check if a handler app is installed for universal links

```swift
UIApplication.shared().
open(url, options: [UIApplicationOpenURLOptionUniversalLinksOnly: true]) {
(didOpen: Bool) in
    if !didOpen {
        // No application available
    }
}
```
Core Data
Core Data

Query generations
Core Data

Query generations
Concurrency improvements
Core Data

Query generations
Concurrency improvements
Tooling improvements
Core Data

Query generations
Concurrent improvements
Tooling improvements
CloudKit

Public databases
Private, per user databases
CloudKit

Public databases
Private, per user databases
Record sharing
CloudKit

UICloudSharingController
CloudKit

UICloudSharingController

Managing the invitation flow
CloudKit

UICloudSharingController

Managing the invitation flow

Private and secure
CloudKit
UICloudSharingController

Managing the invitation flow
Private and secure

let sharingController = UICloudSharingController(share: share, container: self.container)
Family Grocery list

Dad is going to Trader Joe's on Tue.

Don't forget the bbq this Saturday.

- Milk - gallon
- Butter
- Eggs - two dozen
- 5 racks baby back ribs
- Avocados (6)
- Corn chips
- Tomato sauce
Family Grocery list

Dad is going to Trader Joe's on Thursday.

Don't forget the BBQ this Saturday. And:

- Milk - gallon
- Butter
- Eggs - two dozen
- 5 racks baby back ribs
- Avocados (6)
- Corn chips
- Tomato sauce

People

People you invite can make changes

Derek Parker (Owner)
Emily Parker

Add People

Copy Link

Stop Sharing
What’s New with CloudKit

Presidio  Thursday 3:00PM
NSUserActivity
NSUserActivity

Capture the state of your application
NSUserActivity

Capture the state of your application
NSUserActivity

Capture the state of your application
Infrastructure for Handoff, Spotlight, …
NSUserActivity

Capture the state of your application
Infrastructure for Handoff, Spotlight, …
Now supports locations
Capture the state of your application
Infrastructure for Handoff, Spotlight, …
Now supports locations

```
activity.mapItem = myLocation
```
Capture the state of your application
Infrastructure for Handoff, Spotlight, …
Now supports locations

```swift
activity.mapItem = myLocation
```
App Search

In iOS 9, we added support for indexed activities and indexed content
In iOS 9, we added support for indexed activities and indexed content.

```swift
let userActivity = NSUserActivity(activityType: "myActivityType")
userActivity.eligibleForSearch = true
userActivity.eligibleForPublicIndexing = true
userActivity.title = "Presenting What’s New in Cocoa Touch"

let attributes = CSSearchableItemAttributeSet(itemContentType: "public.item")
attributes.displayName = ...

userActivity.contentAttributeSet = attributes
```
Users can continue their search in the app.
App Search

Users can continue their search in the app.
App Search
App Search

Add a **CoreSpotlightContinuation** key in your Info plist
App Search

Add a `CoreSpotlightContinuation` key in your Info plist
Implement a new `UIApplicationDelegate` method
App Search

Add a `CoreSpotlightContinuation` key in your Info.plist

Implement a new UIApplicationDelegate method

```swift
func application(application: UIApplication,
    continueUserActivity userActivity: NSUserActivity,
    restorationHandler: ([AnyObject]?) -> Void) -> Bool {
    if userActivity.activityType == CSQueryContinuationActionType {
        if let searchQuery = userActivity.userInfo?[CSSearchQueryString] as? String {
            // Search
        }
        return true
    }
    return false
}
```
CoreSpotlight Search API

CSSearchQuery
CoreSpotlight Search API

CSSearchQuery

Search the data you’ve already indexed with Spotlight
CoreSpotlight Search API

CSSearchQuery

Search the data you’ve already indexed with Spotlight

Great power and performance, full content search
CoreSpotlight Search API

CSSearchQuery

Search the data you’ve already indexed with Spotlight
Great power and performance, full content search
Powerful query syntax
CoreSpotlight Search API

CSSearchQuery

Search the data you’ve already indexed with Spotlight
Great power and performance, full content search
Powerful query syntax

```swift
let query = CSSearchQuery(queryString: queryString, attributes: ["displayName"])

query.foundItemsHandler = {
    (items: [CSSearchableItem]) in
        /* process received items */
}

query.start()
```
CoreSpotlight Search API

CSSearchQuery

Search the data you’ve already indexed with Spotlight
Great power and performance, full content search
Powerful query syntax

```swift
let query = CSSearchQuery(queryString: queryString, attributes: ["displayName"])

query.foundItemsHandler = {
    (items: [CSSearchableItem]) in
    /* process received items */
}

query.start()
```
ReplayKit
RPBroadcastActivityViewController
ReplayKit
RPBroadcastActivityViewController

Now supports live broadcasting
ReplayKit
RPBroadcastActivityViewController

Now supports live broadcasting
Third-party services support
ReplayKit
RPBroadcastActivityViewController

Now supports live broadcasting
Third-party services support

Go Live with ReplayKit
Mission
Tuesday 10:00AM
In this puzzle, Byte must collect four gems that are located in the same relative locations around a square. You’ll create a loop that repeats the code below for each of the sides to solve the entire puzzle.

1. Drag out a for loop from the code library, then drop it above the existing code.
2. Tap the bottom curly brace to select the loop.
3. Tap and hold on that curly brace, then drag it downwards to pull the existing code into the loop.

```swift
for i in 1...4 {
    moveForward()
    collectGem()
    moveForward()
}[
    moveForward()
    turnRight()
```
SceneKit

New realistic rendering
  • Physically-based rendering
  • High dynamic range
  • Linear color space
Advances in SceneKit Rendering

Presidio
Tuesday 2:00PM
Apple Pay
# Apple Pay

Currently

<table>
<thead>
<tr>
<th></th>
<th>In Apps</th>
<th>In Stores</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPhone</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Apple Watch</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>iPad</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In Apps</td>
<td>In Stores</td>
</tr>
<tr>
<td>----------------</td>
<td>---------</td>
<td>-----------</td>
</tr>
<tr>
<td>iPhone</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Apple Watch</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>iPad</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Mac</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Apple Pay
Apple Pay

Apple Pay in UI code
Apple Pay

Apple Pay in UI code
Apple Pay in Safari
Apple Pay

Apple Pay in UI code
Apple Pay in Safari
• Also available in SFSafariViewController
Apple Pay

Apple Pay in UI code
Apple Pay in Safari
  • Also available in SFSafariViewController
Apple Pay in non-UI extensions
Apple Pay

Apple Pay in UI code

Apple Pay in Safari
  • Also available in SFSafariViewController

Apple Pay in non-UI extensions

Great feature for your iMessage apps
Apple Pay

Apple Pay in UI code
Apple Pay in Safari
• Also available in *SFSafariViewController*
Apple Pay in non-UI extensions
Great feature for your iMessage apps
Apple Pay

Apple Pay in UI code

Apple Pay in Safari
• Also available in `SFSafariViewController`

Apple Pay in non-UI extensions

Great feature for your iMessage apps
Apple Pay

Apple Pay in UI code

Apple Pay in Safari
• Also available in `SFSafariViewController`

Apple Pay in non-UI extensions

Great feature for your iMessage apps
Apple Pay in UI code
Apple Pay in Safari
• Also available in `SFSafariViewController`
Apple Pay in non-UI extensions
Great feature for your iMessage apps
Apple Pay

Apple Pay in UI code

Apple Pay in Safari

- Also available in `SFSafariViewController`

Apple Pay in non-UI extensions

Great feature for your iMessage apps
Apple Pay

Apple Pay in UI code
Apple Pay in Safari
• Also available in SFSafariViewController
Apple Pay in non-UI extensions
Great feature for your iMessage apps
Apple Pay

Apple Pay in UI code
Apple Pay in Safari
• Also available in `SFSafariViewController`
Apple Pay in non-UI extensions
Great feature for your iMessage apps

<table>
<thead>
<tr>
<th>Apple Pay on the Web</th>
<th>Mission</th>
<th>Tuesday 1:40PM</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s New with Wallet and Apple Pay</td>
<td>Mission</td>
<td>Tuesday 3:00PM</td>
</tr>
</tbody>
</table>
Apple Pay

Apple Pay in UI code

Apple Pay in Safari

• Also available in SFSafariViewController

Apple Pay in non-UI extensions

Great feature for your iMessage apps

<table>
<thead>
<tr>
<th>Apple Pay on the Web</th>
<th>Mission</th>
<th>Tuesday 1:40PM</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s New with Wallet and Apple Pay</td>
<td>Mission</td>
<td>Tuesday 3:00PM</td>
</tr>
</tbody>
</table>
Integrating with iOS
Keyboards Extensions

Automatically switch your multi-language keyboard extension based on text content
Add system Globe Key functionality in your own keyboard extension
Widgets

Display modes

Widgets now have the concept of “display modes”
Widgets

Display modes

Widgets now have the concept of “display modes”

• User-controlled
• Compact is fixed height
Widgets

Display modes

Widgets now have the concept of “display modes”

- User-controlled
- Compact is fixed height
- Expanded is variable
public enum NCWidgetDisplayMode : Int {
    case compact
    case expanded
}

public var widgetLargestAvailableDisplayMode: NCWidgetDisplayMode
public var widgetActiveDisplayMode: NCWidgetDisplayMode { get }
public func widgetMaximumSize(for displayMode: NCWidgetDisplayMode) -> CGSize

optional public func widgetActiveDisplayModeDidChange(_ activeDisplayMode: NCWidgetDisplayMode, withMaximumSize maxSize: CGSize)
Widgets
Privacy best practices
Widgets

Privacy best practices

Your widget will be on the lock screen
Widgets
Privacy best practices

Your widget will be on the lock screen
Don’t surprise your users
User Notifications
User Notifications

New Framework in iOS 10
User Notifications

New Framework in iOS 10
User Notifications

New Framework in iOS 10

Feature Parity
User Notifications

New Framework in iOS 10

Feature Parity
Unifies local and remote notification
User Notifications

New Framework in iOS 10

Feature Parity
Unifies local and remote notification
Better delivery management
User Notifications

New Framework in iOS 10

Feature Parity
Unifies local and remote notification
Better delivery management
In-app presentation option
User Notifications

New Framework in iOS 10

Feature Parity
Unifies local and remote notification
Better delivery management
In-app presentation option
Multi-Platform
User Notifications

Service extension

Server-side application → APNS → Extension → Smartphone

Introduction to Notifications
Session 707
Welcome! Let’s talk about Notifications!
User Notifications

Service extension

Non-UI extension point
User Notifications

Service extension

Non-UI extension point

Use cases
User Notifications

Service extension

Non-UI extension point

Use cases

Media attachments
User Notifications

Service extension

Non-UI extension point

Use cases
  Media attachments
  End-to-end encryption
User Notifications

Content extension
User Notifications

Content extension
User Notifications

Content extension

UI extension point
User Notifications

Content extension

UI extension point

Custom views
User Notifications

Content extension

UI extension point
Custom views
No direct interaction
User Notifications

<table>
<thead>
<tr>
<th>Introduction to Notifications</th>
<th>Pacific Heights</th>
<th>Wednesday 9:00AM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Notifications</td>
<td>Pacific Heights</td>
<td>Wednesday 10:00AM</td>
</tr>
</tbody>
</table>
CallKit
Directory Extension

Blocking
CallKit

Directory Extension

Blocking
Identification
public class CXCallDirectoryExtensionContext : NSExtensionContext {

    public func addBlockingEntry(withNextSequentialPhoneNumber phoneNumber: String)

    public func addIdentificationEntry(withNextSequentialPhoneNumber phoneNumber: String, label: String)

}
CallKit
Call Provider API
CallKit
Call Provider API

A 1st party experience for your VoIP application
CallKit

Call Provider API

A 1st party experience for your VoIP application

Full screen incoming call UI
CallKit
Call Provider API

A 1st party experience for your VoIP application
Full screen incoming call UI
Integrated with other types of calls
CallKit

Call Provider API

A 1st party experience for your VoIP application
Full screen incoming call UI
Integrated with other types of calls
VoIP calls appears in Favorites and Recents
CallKit

Call Provider API

A 1st party experience for your VoIP application
Full screen incoming call UI
Integrated with other types of calls
VoIP calls appears in Favorites and Recents
Supports Siri, CarPlay, Do Not Disturb, Bluetooth
CallKit
Call Provider API

A 1st party experience for your VoIP application
Full screen incoming call UI
Integrated with other types of calls
VoIP calls appears in Favorites and Recents
Supports Siri, CarPlay, Do Not Disturb, Bluetooth
SiriKit
SiriKit
SiriKit
SiriKit
SiriKit Recognition
SiriKit
SiriKit

Recognition   Domain   Intents
SiriKit
SiriKit

Recognition → Domain → Intents → Your Extension

Vocabulary
Intents Extension

Handle the interaction between Siri and your application

• Intents and responses
Intents Extension

Handle the interaction between Siri and your application

• Intents and responses

Intents are domain specific
Intents Extension

Handle the interaction between Siri and your application

• Intents and responses

Intents are domain specific

• Make sure Siri and your app agree on the request before performing it
Intents Extension

Handle the interaction between Siri and your application

- Intents and responses

Intents are domain specific

- Make sure Siri and your app agree on the request before performing it

“Tell Miko on WWDCChat we need to meet after this session”
Intents Extension

Handle the interaction between Siri and your application

- Intents and responses
- Intents are domain specific
- Make sure Siri and your app agree on the request before performing it

“Tell Miko on WWDCChat we need to meet after this session”
IntentsUI Extension

Embed your own UI in the Siri Transcript

• Optional
Intents are Shared

Intents describe requests

• For Siri to communicate with your app
• To integrate with CallKit
• For Ride Sharing in Maps
• To donate information to the system about a contact
Intents are Shared

Intents describe requests

• For Siri to communicate with your app
• To integrate with CallKit
• For Ride Sharing in Maps
• To donate information to the system about a contact
Intents are Shared

Intents describe requests

• For Siri to communicate with your app
• To integrate with CallKit
• For Ride Sharing in Maps
• To donate information to the system about a contact
Intents are Shared

Intents describe requests
- For Siri to communicate with your app
- To integrate with CallKit
- For Ride Sharing in Maps
- To donate information to the system about a contact
iMessage Apps

Write apps for Messages
iMessage Apps

Write apps for Messages
Sticker packs
iMessage Apps

Write apps for Messages
Sticker packs
Messages extension
iMessage Apps
Sticker Packs
iMessage Apps
Sticker Packs

No code required
iMessage Apps
Sticker Packs

No code required
Package and distribute your images
iMessage Apps
Message Extensions
iMessage Apps
Message Extensions

Dynamic stickers content
iMessage Apps
Message Extensions

Dynamic stickers content
Customize your UI
Let's build an ice cream
iMessage Apps
Message Extensions

Interactive Messages
iMessage Apps
Message Extensions

Interactive Messages
Connect and integrate with a Messages session
iMessage Apps

Message Extensions

Interactive Messages
Connect and integrate with a Messages session
Custom content
iMessage Apps
iMessage Apps

<table>
<thead>
<tr>
<th>Event</th>
<th>Location</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>iMessage Apps and Stickers, Part 1</td>
<td>Presido</td>
<td>Tuesday 11:00AM</td>
</tr>
<tr>
<td>iMessage Apps and Stickers, Part 2</td>
<td>Presidio</td>
<td>Thursday 1:40PM</td>
</tr>
</tbody>
</table>
More Information

<table>
<thead>
<tr>
<th>Lab Name</th>
<th>Framework Lab</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cocoa Touch Lab</td>
<td>Frameworks Lab D</td>
<td>Tuesday 2:30PM</td>
</tr>
<tr>
<td>Cocoa Touch Lab</td>
<td>Frameworks Lab A</td>
<td>Wednesday 3:00PM</td>
</tr>
<tr>
<td>UIKit and UIKit Animations Lab</td>
<td>Frameworks Lab C</td>
<td>Thursday 1:00PM</td>
</tr>
<tr>
<td>Cocoa Touch 3D Touch Lab</td>
<td>Frameworks Lab C</td>
<td>Friday 10:30AM</td>
</tr>
</tbody>
</table>