Modern User Interaction on iOS

Mastering the UIKit UIGestureRecognizer System

Session 219

Dominik Wagner, UIKit Engineer
Michael Turner, UIKit Engineer
Glen Low, UIKit Engineer
Multi-Touch
The UIGestureRecognizer system

System gesture interaction

Playing nice with Drag and Drop
The UIGestureRecognizer system
System gesture interaction
Playing nice with Drag and Drop
The UIGestureRecognizer system
System gesture interaction
Playing nice with Drag and Drop
Basics

UITouch

UIGestureRecognizer
AssistiveTouch

AssistiveTouch allows you to use your iPhone if you have difficulty touching the screen or if you require an adaptive accessory.

Customize Top Level Menu...

CUSTOM GESTURES

Create New Gesture...

Custom gestures allow you to record gestures that can be activated from Custom in the Menu.
AssistiveTouch

AssistiveTouch allows you to use your iPhone if you have difficulty touching the screen or if you require an adaptive accessory.

Customize Top Level Menu...

CUSTOM GESTURES

Create New Gesture...

Custom gestures allow you to record gestures that can be activated from Custom in the Menu.
UITouch began

Tap Gesture Recognizer

View

Touch Handling
UITouch began

possible
Tap Gesture Recognizer

began

View

Touch Handling
UITouch

Tap Gesture Recognizer

View

Touch Handling

began

began

possible

began
UITouch -> possible
Tap Gesture Recognizer -> View
View -> Touch Handling
UITouch

possible
Tap Gesture Recognizer

View

Touch Handling

ended

began

ended

began
UITouch

Ended

Tap Gesture Recognizer

Began

Ended

View

Touch Handling

Began
// Influencing responder based touch handling

class UIGestureRecognizer : NSObject {

    open var delaysTouchesEnded: Bool // default is true.
    open var cancelsTouchesInView: Bool // default is true.
    open var delaysTouchesBegan: Bool // default is false.
}

// Influencing responder based touch handling

class UIGestureRecognizer : NSObject {

  open var delaysTouchesEnded: Bool // default is true.
  open var cancelsTouchesInView: Bool // default is true.
  open var delaysTouchesBegan: Bool // default is false.
}
Gesture Recognizers first
Pan Gesture Recognizer → Tap Gesture Recognizer → View → Touch Handling
began

UIApplication

Pan Gesture
Recognizer

Tap Gesture
Recognizer

View

Touch

Handling
UITouch

possible
Pan Gesture Recognizer
began
began

possible
Tap Gesture Recognizer
began

View

Touch Handling
UITouch

possible
Pan Gesture Recognizer

possible
Tap Gesture Recognizer

began
began
began

View

began

Touch Handling
UITouch

possible

Pan Gesture Recognizer
began

possible

Tap Gesture Recognizer
began

View

Touch Handling

began
UITouch

Pan Gesture Recognizer
- began

Tap Gesture Recognizer
- began

View

Touch Handling
- began
- cancelled
UITouch

- Pan Gesture Recognizer
  - began
  - moved
  - ended
- Tap Gesture Recognizer
  - began
  - moved
  - moved

View

- Touch Handling
  - began
  - cancelled
Exclusion
// Influencing exclusion

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {
    
    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
    shouldRecognizeSimultaneouslyWith otherGestureRecognizer: UIGestureRecognizer) -> Bool

}

open class UIGestureRecognizer : NSObject {

    open func canPrevent(_ preventedGestureRecognizer: UIGestureRecognizer) -> Bool
    open func canBePrevented(by preventingGestureRecognizer: UIGestureRecognizer) -> Bool

}
// Influencing exclusion

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                                            shouldRecognizeSimultaneouslyWith otherGestureRecognizer: UIGestureRecognizer) -> Bool

}

open class UIGestureRecognizer : NSObject {

    open func canPrevent(_ preventedGestureRecognizer: UIGestureRecognizer) -> Bool
    open func canBePrevented(by preventingGestureRecognizer: UIGestureRecognizer) -> Bool

}
// Influencing exclusion

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
        shouldRecognizeSimultaneouslyWith otherGestureRecognizer: UIGestureRecognizer) -> Bool
}

open class UIGestureRecognizer : NSObject {

    open func canPrevent(_ preventedGestureRecognizer: UIGestureRecognizer) -> Bool
    open func canBePrevented(by preventingGestureRecognizer: UIGestureRecognizer) -> Bool
}

UITouch

began

Pan Gesture Recognizer

Tap Gesture Recognizer

View
UITouch -> Pan Gesture Recognizer -> Tap Gesture Recognizer -> View
UITouch

possible Pan Gesture Recognizer
began
moved

possible Tap Gesture Recognizer
began

View
gestureRecognizer( _,
  shouldRecognizeSimultaneously: ) -> true

UIKit

Pan Gesture Recognizer

begin

Tap Gesture Recognizer

begin

View

moved

moved

moved

moved
gestureRecognizer(_:shouldRecognizeSimultaneously:) -> true
gestureRecognizer(_,
    shouldRecognizeSimultaneously:) -> true
gestureRecognizer(_:shouldRecognizeSimultaneously:) -> true
gestureRecognizer(_:shouldRecognizeSimultaneously:) -> true
UITouch

Pan Gesture Recognizer
  - began
  - moved
  - ended

Tap Gesture Recognizer
  - began
  - moved
  - ended

View

gestureRecognizer(_, shouldRecognizeSimultaneously:) -> true
gestureRecognizer(_:shouldRecognizeSimultaneously:) -> true
gestureRecognizer(_,
shouldRecognizeSimultaneously:) -> true
gestureRecognizer(_:shouldRecognizeSimultaneously:) -> true

UI Touch

Pan Gesture Recognizer
- began
- moved
- ended

Tap Gesture Recognizer
- began
- moved
- ended

View
Failure Requirements
// Failure Requirements

class UIGestureRecognizer : NSObject {

    open func require(toFail otherGestureRecognizer: UIGestureRecognizer)
}

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                                             shouldRequireFailureOf otherGestureRecognizer: UIGestureRecognizer) -> Bool

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                                             shouldBeRequiredToFailBy otherGestureRecognizer: UIGestureRecognizer) -> Bool
}

// Subclasses
open class UIGestureRecognizer : NSObject {

    open func shouldRequireFailure(of otherGestureRecognizer: UIGestureRecognizer) -> Bool
    open func shouldBeRequiredToFail(by otherGestureRecognizer: UIGestureRecognizer) -> Bool
}
// Failure Requirements

class UIGestureRecognizer : NSObject {

    open func require(toFail otherGestureRecognizer: UIGestureRecognizer)
}

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                                             shouldRequireFailureOf otherGestureRecognizer: UIGestureRecognizer) -> Bool

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                                             shouldBeRequiredToFailBy otherGestureRecognizer: UIGestureRecognizer) -> Bool
}

// Subclasses
open class UIGestureRecognizer : NSObject {

    open func shouldRequireFailure(of otherGestureRecognizer: UIGestureRecognizer) -> Bool
    open func shouldBeRequiredToFail(by otherGestureRecognizer: UIGestureRecognizer) -> Bool
}
// Failure Requirements
class UIGestureRecognizer : NSObject {

    open func require(toFail otherGestureRecognizer: UIGestureRecognizer)
}

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                                          shouldRequireFailureOf otherGestureRecognizer: UIGestureRecognizer) -> Bool

optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                                          shouldBeRequiredToFailBy otherGestureRecognizer: UIGestureRecognizer) -> Bool

}

// Subclasses
open class UIGestureRecognizer : NSObject {

    open func shouldRequireFailure(of otherGestureRecognizer: UIGestureRecognizer) -> Bool
    open func shouldBeRequiredToFail(by otherGestureRecognizer: UIGestureRecognizer) -> Bool

}
class UIGestureRecognizer : NSObject {

    open func require(toFail otherGestureRecognizer: UIGestureRecognizer)

}

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                                            shouldRequireFailureOf otherGestureRecognizer: UIGestureRecognizer) -> Bool

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                                            shouldBeRequiredToFailBy otherGestureRecognizer: UIGestureRecognizer) -> Bool

}

// Subclasses
open class UIGestureRecognizer : NSObject {

    open func shouldRequireFailure(of otherGestureRecognizer: UIGestureRecognizer) -> Bool
    open func shouldBeRequiredToFail(by otherGestureRecognizer: UIGestureRecognizer) -> Bool

}
Pan Gesture Recognizer \hspace{1cm} \text{require(toFail:)} \hspace{1cm} \text{Tap Gesture Recognizer}
UITouch

began

Pan Gesture Recognizer

require(toFail:)

Tap Gesture Recognizer

View
UITouch
began

Pan Gesture Recognizer
possible

require(toFail:)

Tap Gesture Recognizer
possible

View
UITouch

began

moved

Pan Gesture Recognizer

possible

began

require(toFail:)

Tap Gesture Recognizer

possible

possible

View
UITouch

- began
- moved

Pan Gesture Recognizer

- possible

Begin phase

no action

View

Pan Gesture Recognizer

- began

possible

possible

possible
UIDTouch

Pan Gesture Recognizer

require(toFail:)

no action

UITouch

began

moved

moved

View

Tap Gesture Recognizer
UITouch

Pan Gesture Recognizer

require(toFail:)

Tap Gesture Recognizer

View

began

possible

began

no action

moved

changed

possible

possible
UITouch
began
moved
moved
ended

Pan Gesture Recognizer
possible
began
changed
no action

require(toFail:)

Tap Gesture Recognizer
possible
no action
possible

View
UITouch

Pan Gesture Recognizer

require(toFail:)

Tap Gesture Recognizer

View

began

possible

began

no action

possible

changed

no action

possible

ended

possible

ended

began

moved

moved

ended
UITouch
began
moved
moved
ended

Pan Gesture Recognizer
possible
began
changed
ended

require(toFail:)
no action
no action
no action

Tap Gesture Recognizer
possible
possible
possible
ended

View
163 UIGestureRecognizers
Hit Testing
Hit Testing
Hit Testing
Hit Testing
// UIView subclasses
open class UIView : NSObject {

    open func hitTest(_ point: CGPoint, with event: UIEvent?) -> UIView?
    open func point(inside point: CGPoint, with event: UIEvent?) -> Bool

}

class UIView : NSObject {

    var isUserInteractionEnabled: Bool
    var alpha: CGFloat
    var isHidden: Bool
    var isMultipleTouchEnabled: Bool

}

public struct UIViewAnimationOptions : OptionSet {

    public static var allowUserInteraction: UIViewAnimationOptions { get }

}

public static var allowUserInteraction: UIViewAnimationOptions { get }
// UIView subclasses
open class UIView : NSObject {

    open func hitTest(_ point: CGPoint, with event: UIEvent?) -> UIView?
    open func point(inside point: CGPoint, with event: UIEvent?) -> Bool
}

class UIView : NSObject {

    var isUserInteractionEnabled: Bool
    var alpha: CGFloat
    var isHidden: Bool
    var isMultipleTouchEnabled: Bool
}

public struct UIViewAnimationOptions : OptionSet {

    public static var allowUserInteraction: UIViewAnimationOptions { get }

}
// UIView subclasses
open class UIView : NSObject {

    open func hitTest(_ point: CGPoint, with event: UIEvent?) -> UIView?
    open func point(inside point: CGPoint, with event: UIEvent?) -> Bool
}

class UIView : NSObject {

    var isUserInteractionEnabled: Bool
    var alpha: CGFloat
    var isHidden: Bool

    var isMultipleTouchEnabled: Bool
}

public struct UIViewAnimationOptions : OptionSet {

    public static var allowUserInteraction: UIViewAnimationOptions { get }
}

}
// UIView subclasses
open class UIView : NSObject {

    open func hitTest(_ point: CGPoint, with event: UIEvent?) -> UIView?
    open func point(inside point: CGPoint, with event: UIEvent?) -> Bool
}

class UIView : NSObject {

    var isUserInteractionEnabled: Bool
    var alpha: CGFloat
    var isHidden: Bool

    var isMultipleTouchEnabled: Bool
}

public struct UIViewAnimationOptions : OptionSet {

    public static var allowUserInteraction: UIViewAnimationOptions { get }

    public static var allowUserInteraction: UIViewAnimationOptions { get }
}
// UIView subclasses
open class UIView : NSObject {

    open func hitTest(_ point: CGPoint, with event: UIEvent?) -> UIView?
    open func point(inside point: CGPoint, with event: UIEvent?) -> Bool

}

class UIView : NSObject {

    var isUserInteractionEnabled: Bool
    var alpha: CGFloat
    var isHidden: Bool
    var isMultipleTouchEnabled: Bool

}

public struct UIViewAnimationOptions : OptionSet {

    public static var allowUserInteraction: UIViewAnimationOptions { get }

}
// UIView subclasses
open class UIView : NSObject {

    open func hitTest(_ point: CGPoint, with event: UIEvent?) -> UIView?
    open func point(inside point: CGPoint, with event: UIEvent?) -> Bool
}

class UIView : NSObject {

    var isUserInteractionEnabled: Bool

    var alpha: CGFloat
    var isHidden: Bool

    var isMultipleTouchEnabled: Bool
}

public struct UIViewAnimationOptions : OptionSet {

    public static var allowUserInteraction: UIViewAnimationOptions { get }
}


Hit Testing and Animations

Presentation layer vs model layer

UIViewPropertyAnimator and `isManualHitTestingEnabled`

Building Interruptible and Responsive Interactions

Advances in UIKit Animations and Transitions
Relevant Gesture Recognizers
Relevant Gesture Recognizers
Relevant Gesture Recognizers
// Influencing participation in the interaction

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer, shouldReceive touch: UITouch) -> Bool

    optional public func gestureRecognizerShouldBegin(_ gestureRecognizer: UIGestureRecognizer) -> Bool

}

class UIGestureRecognizer : NSObject {

    var isEnabled: Bool

    var allowedTouchTypes: [NSNumber]
    var requiresExclusiveTouchType: Bool

}
// Influencing participation in the interaction

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                                           shouldReceive touch: UITouch) -> Bool

    optional public func gestureRecognizerShouldBegin(_ gestureRecognizer:
                                                 UIGestureRecognizer) -> Bool

}

class UIGestureRecognizer : NSObject {

    var isEnabled: Bool

    var allowedTouchTypes: [NSNumber]
    var requiresExclusiveTouchType: Bool

}
public protocol UIGestureRecognizerDelegate : NSObjectProtocol {
    
    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer, shouldReceive touch: UITouch) -> Bool
    
    optional public func gestureRecognizerShouldBegin(_ gestureRecognizer: UIGestureRecognizer) -> Bool
}

class UIGestureRecognizer : NSObject {
    
    var isEnabled: Bool
    
    var allowedTouchTypes: [NSNumber]
    var requiresExclusiveTouchType: Bool
}
// Influencing participation in the interaction

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                                           shouldReceive touch: UITouch) -> Bool

    optional public func gestureRecognizerShouldBegin(_ gestureRecognizer: UIGestureRecognizer) -> Bool

}

class UIGestureRecognizer : NSObject {

    var isEnabled: Bool

    var allowedTouchTypes: [NSNumber]
    var requiresExclusiveTouchType: Bool

}
public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer, shouldReceive touch: UITouch) -> Bool

    optional public func gestureRecognizerShouldBegin(_ gestureRecognizer: UIGestureRecognizer) -> Bool
}

class UIGestureRecognizer : NSObject {

    var isEnabled: Bool

    var allowedTouchTypes: [NSNumber]
    var requiresExclusiveTouchType: Bool
}
// Influencing participation in the interaction

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
        shouldReceive touch:UITouch) -> Bool

    optional public func gestureRecognizerShouldBegin(_ gestureRecognizer:
        UIGestureRecognizer) -> Bool
}

class UIGestureRecognizer : NSObject {

    var isEnabled: Bool

    var allowedTouchTypes: [NSNumber]

    var requiresExclusiveTouchType: Bool
}
New on UIGestureRecognizer

Debugging only

@available(iOS 11.0, *)

open var name: String? // name for debugging to appear in logging
Debugging

Breakpoint opportunities

gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
    shouldReceive touch: UITouch) -> Bool

touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?)

Things to inspect

po touches.first?.gestureRecognizers

po event?.touches(for:someGestureRecognizer)

po touches.first?.view?.superview?.gestureRecognizers
Custom UILegureRecognizers

Begin late and fail fast!

Ignore touches `ignore(_ touch:, for event:)`

Don’t forget `.touchesCancelled(_:with:)`
Gesture Recognizer System
Gesture Recognizer System

Revisit your setups
Gesture Recognizer System

Revisit your setups

Exclusion and failure requirements
Gesture Recognizer System

Revisit your setups

Exclusion and failure requirements

Are your gesture recognizers on the right views?
System Gesture Interaction

Glen Low, UIKit Engineer
Cover Sheet or Killjoy?
Life on the Edges
Life on the Edges

Multitasking / Dock
Life on the Edges
Life on the Edges

Gesture Recognizers and Responders

Multitasking / Dock
Who Gets the Touches?

System ↔ App

Tap
Pinch
Rotate
Long Press
<table>
<thead>
<tr>
<th>Who Gets the Touches?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System ↔ App</strong></td>
</tr>
<tr>
<td>Tap</td>
</tr>
<tr>
<td>Pinch</td>
</tr>
<tr>
<td>Rotate</td>
</tr>
<tr>
<td>Long Press</td>
</tr>
<tr>
<td>• System</td>
</tr>
<tr>
<td>• App</td>
</tr>
<tr>
<td>Pan</td>
</tr>
<tr>
<td>Swipe</td>
</tr>
<tr>
<td>Responders</td>
</tr>
</tbody>
</table>
System gesture or app gesture?

We made an educated guess.
System gesture or app gesture?

You tell us.
class MyViewController: UIViewController {

    // override to return which screen edges to defer system gestures
    override func preferredScreenEdgesDeferringSystemGestures() -> UIRectEdge {
        return deferControlCenter ? .bottom : UIRectEdge()
    }

    // call whenever your method would return a different screen edge
    var deferControlCenter : Bool {
        didSet { setNeedsUpdateOfScreenEdgesDeferringSystemGestures() }
    }
}
Let’s Talk Terms

class MyViewController: UIViewController {

    // override to return which screen edges to defer system gestures
    override func preferredScreenEdgesDeferringSystemGestures() -> UIRectEdge {
        return deferControlCenter ? .bottom : UIRectEdge()
    }

    // call whenever your method would return a different screen edge
    var deferControlCenter : Bool {
        didSet { setNeedsUpdateOfScreenEdgesDeferringSystemGestures() }
    }

}
class MyViewController: UIViewController {

    // override to return which screen edges to defer system gestures
    override func preferredScreenEdgesDeferringSystemGestures() -> UIRectEdge {
        return deferControlCenter ? .bottom : UIRectEdge()
    }

    // call whenever your method would return a different screen edge
    var deferControlCenter : Bool {
        didSet { setNeedsUpdateOfScreenEdgesDeferringSystemGestures() }
    }

}
override func preferredScreenEdgesDeferringSystemGestures() -> UIRectEdge {
    return deferControlCenter ? .bottom : UIRectEdge()
}
override func preferredScreenEdgesDeferringSystemGestures() -> UIRectEdge {
    return deferControlCenter ? .bottom : UIRectEdge()
}
override func preferredScreenEdgesDeferringSystemGestures() -> UIRectEdge {
    return deferControlCenter ? .bottom : UIRectEdge()
}
class MyContainerViewController: UIViewController {

    // override to return which child view controller to consult
    override func childViewControllerForScreenEdgesDeferringSystemGestures() -> UIViewController? {
        return mySelectedChildViewController
    }
}

Don’t Do It, Because...

Don’t mess with the familiar

Your recognizers may already get the touches

Telling us ≠ us doing it
Don’t mess with the familiar
Your recognizers may already get the touches
Telling us ≠ us doing it
Your app is only used casually
Playing Nice With Drag and Drop

Michael Turner, UIKit Engineer
Adding UIDragInteraction to a UIView within your app is easy

```swift
let dragInteraction = UIDragInteraction(delegate: myDelegate)
myView.addInteraction(dragInteraction)
```
Initiating a Drag
Initiating a Drag
UIDragInteraction Example

UIView

var interactions

UIDragInteraction

var gestureRecognizers

UIGestureRecognizer

UIGestureRecognizer

UIGestureRecognizer
UIDragInteraction Example

UIView

```swift
var interactions

UIDragInteraction
```

UIViewController

```swift
var gestureRecognizers

UIGestureRecognizer
UIGestureRecognizer
UIGestureRecognizer
UILongPressGestureRecognizer
```

```swift
func longPress(gesture: UIGestureRecognizer) {
    switch gesture.state {
        case .began:
            presentActivityViewController()
        case .cancelled:
            dismissActivityViewController()
        default: break
    }
}
```
Long Press and Move
UILongPressGestureRecognizers are delayed
Long Press, Hold, and Move
A beginning drag will cancel the touch
Compact Trait Environment
Long Press, Hold, and Move
Long presses are delayed until the touch ends
Long Press and Lift
Adding to a Drag
Adding to a Drag

Control this behavior using UIDragInteractionDelegate

```swift
func dragInteraction(_ interaction: UIDragInteraction, itemsForAddingTo session: UIDragSession, withTouchAt point: CGPoint) -> [UIDragItem] {
    // Returning 0 items allows normal touch processing to continue
    return []
}
```
How to Adapt for UIDragInteraction

Examine your existing actions

Present modal UI carefully

Handle the `.cancelled` state

Your app is interactive during a drag!
Summary

Leveraging the gesture system

Use the new deferred system gesture APIs

Working alongside UIDragInteraction
Summary

Leveraging the gesture system

Use the new deferred system gesture APIs

Working alongside UIDragInteraction
Summary

Leveraging the gesture system
Use the new deferred system gesture APIs
Working alongside UIDragInteraction
More Information

<table>
<thead>
<tr>
<th>Session Title</th>
<th>Conference Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building Advanced Gesture Recognizers</td>
<td>WWDC 2012</td>
</tr>
<tr>
<td>Making the Most of Multi-Touch on iOS</td>
<td>WWDC 2011</td>
</tr>
<tr>
<td>Leveraging Touch Input on iOS</td>
<td>WWDC 2016</td>
</tr>
<tr>
<td>Building Interruptible and Responsive Interactions</td>
<td>WWDC 2014</td>
</tr>
<tr>
<td>Advanced Scrollviews and Touch Handling Techniques</td>
<td>WWDC 2014</td>
</tr>
<tr>
<td>Session</td>
<td>Location</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Introducing Drag and Drop</td>
<td>Hall 3</td>
</tr>
<tr>
<td>What's New in Cocoa Touch</td>
<td>Hall 3</td>
</tr>
<tr>
<td>Updating Your App for iOS 11</td>
<td>Hall 3</td>
</tr>
<tr>
<td>Mastering Drag and Drop</td>
<td>Grand Ballroom B</td>
</tr>
<tr>
<td>Advanced Animations with UIKit</td>
<td>Hall 3</td>
</tr>
<tr>
<td>Lab</td>
<td>Location</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>Cocoa Touch and Haptics Lab</td>
<td>Technology Lab C</td>
</tr>
<tr>
<td>UIKit and Collection View Lab</td>
<td>Technology Lab B</td>
</tr>
</tbody>
</table>