Introducing Drag and Drop

Session 203

Bruce Nilo, UIKit Engineering Manager
Kurt Revis, UIKit Engineer
Emanuele Rudel, UIKit Engineer
Goals and concepts
API fundamentals
Showing the APIs in action
Next steps
What is Drag and Drop?
What is Drag and Drop?

A way to graphically move or copy data
Drag and Drop on iOS

Goals
Drag and Drop on iOS

Goals

Responsive
Drag and Drop on iOS

Goals

Responsive

• On demand and asynchronous data delivery
Drag and Drop on iOS

Goals

Responsive

Secure
Drag and Drop on iOS

Goals

Responsive

Secure

• Data is only visible to destination
Drag and Drop on iOS

Goals

Responsive

Secure

• Data is only visible to destination
• Source may restrict access
Drag and Drop on iOS

Goals

Responsive

Secure

A great Multi-Touch experience
Drag and Drop on iOS

Goals
Drag and Drop on iOS

Goals

A great Multi-Touch experience
Drag and Drop on iOS

Goals

A great Multi-Touch experience
• The interface is live
• Deep integration with all of iOS
• Great visual feedback
• Hover to navigate
• Items can be added
• Transfer drags between fingers
• Multiple drag interactions
Drag and Drop on iOS

Vs.
Concepts
Drag and Drop on iOS
Phases of a drag session
## Drag and Drop on iOS

### Phases of a drag session

<table>
<thead>
<tr>
<th>Lift</th>
<th>Drag</th>
<th>Set Down</th>
<th>Data Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>🔄</td>
<td>🔄</td>
<td>🔄</td>
<td>🔄</td>
</tr>
</tbody>
</table>
## Drag and Drop on iOS

### Phases of a drag session

<table>
<thead>
<tr>
<th>Lift</th>
<th>Drag</th>
<th>Set Down</th>
<th>Data Transfer</th>
</tr>
</thead>
</table>

- Long press

- Lift preview
## Drag and Drop on iOS

### Phases of a drag session

<table>
<thead>
<tr>
<th>Lift</th>
<th>Drag</th>
<th>Set Down</th>
<th>Data Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long press</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lift preview</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Previews**
- **Tap to add**
- **Spring-loading**
# Drag and Drop on iOS

## Phases of a drag session

<table>
<thead>
<tr>
<th>Lift</th>
<th>Drag</th>
<th>Set Down</th>
<th>Data Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long press</td>
<td>Previews</td>
<td>Cancel</td>
<td></td>
</tr>
<tr>
<td>Lift preview</td>
<td>Tap to add</td>
<td>Drop</td>
<td></td>
</tr>
<tr>
<td>Spring-loading</td>
<td></td>
<td>Targeting</td>
<td></td>
</tr>
</tbody>
</table>
## Drag and Drop on iOS

### Phases of a drag session

<table>
<thead>
<tr>
<th>Lift</th>
<th>Drag</th>
<th>Set Down</th>
<th>Data Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long press</td>
<td>Previews</td>
<td>Cancel</td>
<td>Representations</td>
</tr>
<tr>
<td>Lift preview</td>
<td>Tap to add</td>
<td>Drop</td>
<td>Lazy delivery</td>
</tr>
<tr>
<td></td>
<td>Spring-loading</td>
<td>Targeting</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By File Provider</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Progress</td>
</tr>
</tbody>
</table>
Enabling a Drag
Concepts - interactions

Diagram:
- View
- Interaction
Enabling a Drag
Concepts - interactions
Enabling a Drag
Concepts - UIDragInteraction

A drag interaction is attached to a view
Enabling a Drag
Concepts - UIDragInteraction

A **drag interaction** is attached to a view
Enabling a Drag
Concepts - UIDragInteraction

A **drag interaction** is attached to a view

```swift
import UIKit

let view: UIView = ...
let delegate: UIDragInteractionDelegate = ...

let dragInteraction = UIDragInteraction(delegate: delegate)
view.addInteraction(dragInteraction)
```
Lift Phase
Concepts - UIDragInteraction

The delegate provides **drag items** when the view **lifts**
Lift Phase
Concepts - UIDragInteraction

The delegate provides **drag items** when the view **lifts**

No drag items -> drag gesture fails
Drag Items
Concepts - UIDragItem
Drag Items
Concepts - UIDragItem

A drag item represents a model object
Drag Items
Concepts - UIDragItem

A **drag item** represents a model object

A drag item embodies

- Drag preview
- Item provider
Enabling a Drop
Concepts - UIPasteConfiguration
Enabling a Drop
Concepts - UIPasteConfiguration

UIResponders have a new paste configuration property
Enabling a Drop
Concepts - UIPasteConfiguration

UIResponders have a new `paste configuration` property

```swift
// Indicate you can accept or paste strings

let config = UIPasteConfiguration(typeIdentifiersForAcceptingClass: NSString.self)

view.pasteConfiguration = config
```
UIResponders have a new **paste configuration** property

```swift
// Will be called for both Drag and Drop, and Copy/Paste
override func paste(itemProviders: [NSItemProvider]) {
}
```
Enabling a Drop
Concepts - UIDropInteraction

A **drop interaction** is attached to a view
Drag Phase
Concepts - UIDropInteraction

The delegate responds to drag events
Set Down Phase
Concepts - UIDropInteraction

On touch up, the drag session may be cancelled

• The drag preview animates back
Set Down Phase

Concepts - UIDropInteraction

Or the drop is accepted

• The delegate is told to perform drop
Data Transfer Phase
Concepts - UIDropInteraction

Or the drop is accepted
• The delegate is told to perform drop

Diagram:
- Item Provider
- Drag Items
- Drop Interaction Delegate
  - perform drop
Data Transfer Phase
Concepts - UIDropInteraction

Or the drop is accepted
• The delegate is told to perform drop

Delegate requests data representation of items
Data Transfer Phase
Concepts - UIDropInteraction

Or the drop is accepted

• The delegate is told to perform drop

Delegate requests data representation of items
API Roadmap
Using the Drag and Drop API
Drag and drop timeline
API essentials
Introduction to the full API
Drag and Drop Timeline
Drag and Drop Timeline

Time

Touch

Begins
Drag and Drop Timeline

Time

Touch
Begins

Drag and Drop
Lift
Drag and Drop Timeline

Time

Touch

Begin
Stays Down

Drag and Drop

Lift
Drag and Drop Timeline

Time

Touch

Begin

Stays Down

Moves

Drag and Drop

Lift

Drag
Drag and Drop Timeline

Touch:
- Begins
- Stays Down
- Moves

Drag and Drop:
- Lift
- Drag
Drag and Drop Timeline

- Begin
- Stays Down
- Moves
- Ends

- Lift
- Drag
Drag and Drop Timeline

- Time:
  - Touch:
    - Begins
    - Stays Down
    - Moves
    - Ends
  - Drag and Drop:
    - Lift
    - Drag
Drag and Drop Timeline

Touch:
- Begins
- Stays Down
- Moves
- Ends

Drag and Drop:
- Lift
- Drag

Location is not on a view that accepts the drop.
Drag and Drop Timeline

Time

Touch

- Begins
- Stays Down
- Moves
- Ends

Drag and Drop

- Lift
- Drag
- Cancel Animations

Location is not on a view that accepts the drop
Drag and Drop Timeline

Time

Touch

Begins | Stays Down | Moves | Ends

Drag and Drop

Lift | Drag
Drag and Drop Timeline

Time
- Begins
- Stays Down
- Moves
- Ends

Drag and Drop
- Lift
- Drag
Drag and Drop Timeline

Time

Touch

Drag and Drop

Location is on a view that accepts the drop
Drag and Drop Timeline

Time

Touch

Begins

Stays Down

Moves

Ends

Location is on a view that accepts the drop

Drag and Drop

Lift

Drag

Perform
Drag and Drop Timeline

- **Time**
  - Begins
  - Stays Down
  - Moves
  - Ends

- **Touch**
  - Begins
  - Lift
  - Drag
  - Perform

- **Ends** Location is on a view that accepts the drop
Drag and Drop Timeline

**Time**

- **Begins**
- **Stays Down**
- **Moves**
- **Ends**

**Touch**

- **Begins**
- **Stays Down**
- **Moves**
- **Ends**

**Drag and Drop**

- **Lift**
- **Drag**
- **Perform**
- **Drop Animation**

Location is on a view that accepts the drop.
Drag and Drop Timeline

**Touch**
- Begins
- Stays Down
- Moves
- Ends

**Drag and Drop**
- Lift
- Drag
- Perform
- Drop Animation
- Data Transfer

Location is on a view that accepts the drop
Drag and Drop Timeline

Time

Perform  Drop Animation  Data Transfer
Drag and Drop Timeline

Time

Perform

Drop Animation

Data Transfer
Drag and Drop Timeline

- **Touch**
  - Begins
  - Stays Down
  - Moves
  - Ends

- **Drag and Drop**
  - Lift
  - Drag
  - Perform
  - Drop Animations
  - Data Transfer
  - Cancel Animations
func dragInteraction(_ interaction: UIDragInteraction,
    itemsForBeginning session: UIDragSession) -> [UIDragItem]
func dragInteraction(_ interaction: UIDragInteraction, 
    itemsForBeginning session: UIDragSession) -> [UIDragItem]
func dragInteraction(_ interaction: UIDragInteraction,
    itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    let itemProvider = NSItemProvider(object: "Hello World" as NSString)
    let dragItem = UIDragItem(itemProvider: itemProvider)
    return [ dragItem ]
}
func dragInteraction(_ interaction: UIDragInteraction,
    itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    let itemProvider = NSItemProvider(object: "Hello World" as NSString)
    let dragItem = UIDragItem(itemProvider: itemProvider)
    return [dragItem]
}
Get the items to drag

```swift
func dragInteraction(_ interaction: UIDragInteraction,
    itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    let itemProvider = NSItemProvider(object: "Hello World" as NSString)
    let dragItem = UIDragItem(itemProvider: itemProvider)
    return [dragItem]
}```
func dragInteraction(_ interaction: UIDragInteraction, itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    let itemProvider = NSItemProvider(object: "Hello World" as NSString)
    let dragItem = UIDragItem(itemProvider: itemProvider)
    return [dragItem]
}
func dropInteraction(_ interaction: UIDropInteraction,
        sessionDidUpdate session: UIDropSession) -> UIDropProposal {
    return UIDropProposal(operation: UIDropOperation)
}
func dropInteraction(_ interaction: UIDropInteraction,
   sessionDidChange session: UIDropSession) -> UIDropProposal {
   return UIDropProposal(operation: UIDropOperation)
}
func dropInteraction(_ interaction: UIDropInteraction,
    sessionDidUpdate session: UIDropSession) -> UIDropProposal {
    return UIDropProposal(operation: UIDropOperation)
}
UIDropOperation
API Essentials—2
UIDropOperation

- .cancel
- .copy
- .move
- .forbidden
API Essentials—2
UIDropOperation

• .cancel
• .copy
• .move
• .forbidden
API Essentials—2

UIDropOperation

- .cancel
- .copy
- .move
- .forbidden
UIDropOperation

- **.cancel**
- **.copy**
- **.move**
- **.forbidden**
Delegates must cooperate to make it look like a move.
Delegates must cooperate to make it look like a move

Only within a single app
Delegates must cooperate to make it look like a move

Only within a single app

Drag interaction delegate must allow moves
Delegates must cooperate to make it look like a move

Only within a single app

Drag interaction delegate must allow moves

Drop interaction delegate checks

```swift
UIDropSession allowsMoveOperation
```
API Essentials—2
UIDropOperation

- .cancel
- .copy
- .move
- .forbidden
API Essentials—3

Perform the drop

```swift
func dropInteraction(_ interaction: UIDropInteraction, performDrop session: UIDropSession)
```
func dropInteraction(_ interaction: UIDropInteraction, performDrop session: UIDropSession)
func dropInteraction(_ interaction: UIDropInteraction, performDrop session: UIDropSession) {
    session.loadObjects(ofClass: UIImage.self) { objects in
        for image in objects as! [UIImage] {
            self.imageView.image = image
        }
    }
}
func dropInteraction(_ interaction: UIDropInteraction, performDrop session: UIDropSession) {
    session.loadObjects(ofClass: UIImage.self) { objects in
        for image in objects as! [UIImage] {
            self.imageView.image = image
        }
    }
}
func dropInteraction(_ interaction: UIDropInteraction, performDrop session: UIDropSession) { 
    session.loadObjects(ofClass: UIImage.self) { objects in 
        for image in objects as! [UIImage] {
            self.imageView.image = image
        }
    }
}
func dropInteraction(_ interaction: UIDropInteraction, performDrop session: UIDropSession) {
    session.loadObjects(ofClass: UIImage.self) { objects in
        for image in objects as! [UIImage] {
            self.imageView.image = image
        }
    }
}
func dropInteraction(_ interaction: UIDropInteraction, performDrop session: UIDropSession) {
    for item in session.items {
        item.itemProvider.loadObject(ofClass: UIImage.self) { (object, error) in
            if object != nil {
                DispatchQueue.main.async {
                    self.imageView.image = (object as! UIImage)
                }
            } else {
                // Handle the error
            }
        }
    }
}
func dropInteraction(_ interaction: UIDropInteraction, performDrop session: UIDropSession) {
    for item in session.items {
        item.itemProvider.loadObject(ofClass: UIImage.self) { (object, error) in
            if object != nil {
                DispatchQueue.main.async {
                    self.imageView.image = (object as! UIImage)
                }
            } else {
                // Handle the error
            }
        }
    }
}
func dropInteraction(_ interaction: UIDropInteraction, performDrop session: UIDropSession) {
    for item in session.items {
        item.itemProvider.loadObject(ofClass: UIImage.self) { (object, error) in
            if object != nil {
                DispatchQueue.main.async {
                    self.imageView.image = (object as! UIImage)
                }
            }
            else {
                // Handle the error
            }
        }
    }
}
func dropInteraction(_ interaction: UIDropInteraction, performDrop session: UIDropSession) {
    for item in session.items {
        item.itemProvider.loadObject(ofClass: UIImage.self) { (object, error) in
            if object != nil {
                DispatchQueue.main.async {
                    self.imageView.image = (object as! UIImage)
                }
            } else {
                // Handle the error
            }
        }
    }
}
func dropInteraction(_ interaction: UIDropInteraction, performDrop session: UIDropSession) {
    for item in session.items {
        item.itemProvider.loadObject(ofClass: UIImage.self) { (object, error) in
            if object != nil {
                DispatchQueue.main.async {
                    self.imageView.image = (object as! UIImage)
                }
            }
        }
    } else {
        // Handle the error
    }
}
API Essentials

Time

Touch

- Begins
- Stays Down
- Moves
- Ends

Drag and Drop

- Lift
- Drag
- Perform
- Drop Animations

or

- Data Transfer
- Cancel Animations
API Essentials

Drag and Drop

Time

Touch

Begin
Stays Down
Moves
Ends

Get items
to drag

Drag and Drop

Lift
Drag
Perform
Drop Animations

Data Transfer

or

Cancel Animations
API Essentials

**Time**
- **Begins**
- **Stays Down**
- **Moves**
- **Ends**

**Touch**
- Begins
  - Get items to drag
- Stays Down
  - Get a drop proposal
- Moves
- Ends

**Drag and Drop**
- Lift
- Drag
- Perform
- Drop Animations
  - Data Transfer
- or
  - Cancel Animations
API Essentials

Drag and Drop

- **Lift**
  - Get items to drag

- **Drag**
  - Get a drop proposal

- **Perform**
  - Data Transfer

- **Ends**
  - Perform drop

- **Drop Animations**
  - Cancel Animations

Time

- **Begins**
- **Stays Down**
- **Moves**
- **Ends**
Drag Interaction Delegate
func dragInteraction(_ interaction: UIDragInteraction, previewForLifting item: UIDragItem, session: UIDragSession) -> UITargetedDragPreview?
func dragInteraction(_ interaction:UIDragInteraction, previewForLifting item:UIDragItem, session:UIDragSession) -> UITargetedDragPreview?
func dragInteraction(_ interaction: UIDragInteraction,
                 previewForLifting item: UIDragItem, session: UIDragSession)
       -> UITargetedDragPreview?
func dragInteraction(_ interaction: UIDragInteraction,
    previewForLifting item: UIDragItem, session: UIDragSession)
    -> UITargetedDragPreview? {
    let imageView = UIImageView(image: UIImage(named: "MyDragImage"))
    let dragView = interaction.view!
    let dragPoint = session.location(in: dragView)
    let target = UIDragPreviewTarget(container: dragView, center: dragPoint)
    return UITargetedDragPreview(view: imageView, parameters:UIDragPreviewParameters(),
        target:target)
}
func dragInteraction(_ interaction: UIDragInteraction, previewForLifting item: UIDragItem, session: UIDragSession) -> UITargetedDragPreview? {

    let imageView = UIImageView(image: UIImage(named: "MyDragImage"))
    let dragView = interaction.view!
    let dragPoint = session.location(in: dragView)
    let target = UIDragPreviewTarget(container: dragView, center: dragPoint)
    return UITargetedDragPreview(view: imageView, parameters: UIDragPreviewParameters(), target: target)
}
Drag Interaction Delegate

Lift

```swift
func dragInteraction(_ interaction: UIDragInteraction,
    previewForLifting item: UIDragItem, session: UIDragSession)
    -> UITargetedDragPreview? {

    let imageView = UIImageView(image: UIImage(named: "MyDragImage"))
    let dragView = interaction.view!
    let dragPoint = session.location(in: dragView)
    let target = UIDragPreviewTarget(container: dragView, center: dragPoint)
    return UITargetedDragPreview(view: imageView, parameters: UIDragPreviewParameters(),
        target: target)
}
```
func dragInteraction(_ interaction: UIDragInteraction,
    previewForLifting item: UIDragItem, session: UIDragSession)
    -> UITargetedDragPreview? {
    let imageView = UIImageView(image: UIImage(named: "MyDragImage"))
    let dragView = interaction.view!
    let dragPoint = session.location(in: dragView)
    let target = UIDragPreviewTarget(container: dragView, center: dragPoint)
    return UITargetedDragPreview(view: imageView, parameters:UIDragPreviewParameters(),
        target:target)
}
func dragInteraction(_ interaction: UIDragInteraction, previewForLifting item: UIDragItem, session: UIDragSession) -> UITargetedDragPreview? {

    let imageView = UIImageView(image: UIImage(named: "MyDragImage"))
    let dragView = interaction.view!
    let dragPoint = session.location(in: dragView)
    let target = UIDragPreviewTarget(container: dragView, center: dragPoint)

    return UITargetedDragPreview(view: imageView, parameters: UIDragPreviewParameters(), target: target)
}
func dragInteraction(_ interaction: UIDragInteraction, willAnimateLiftWith animator: UIDragAnimating, session: UIDragSession) {
    animator.addAnimations { self.view.backgroundColor = UIColor.gray }
    animator.addCompletion { position in
        if position == .end {
            // The lift ended normally, and a drag is now happening
        }
        else if position == .start {
            // The lift was cancelled and the animation returned to the start
        }
    }
}
func dragInteraction(_: interaction: UIDragInteraction, willAnimateLiftWith animator: UIDragAnimating, session: UIDragSession) {
    animator.addAnimations { self.view.backgroundColor = UIColor.gray }
    animator.addCompletion { position in
        if position == .end {
            // The lift ended normally, and a drag is now happening
        }
        else if position == .start {
            // The lift was cancelled and the animation returned to the start
        }
    }
}
func dragInteraction(_ interaction: UIDragInteraction,
            willAnimateLiftWith animator: UIDragAnimating,
            session: UIDragSession) {
    animator.addAnimations { self.view.backgroundColor = UIColor.gray }
    animator.addCompletion { position in
        if position == .end {
            // The lift ended normally, and a drag is now happening
        }
        else if position == .start {
            // The lift was cancelled and the animation returned to the start
        }
    }
}
func dragInteraction(_ interaction: UIDragInteraction, willAnimateLiftWith animator: UIDragAnimating, session: UIDragSession) {
    animator.addAnimations { self.view.backgroundColor = UIColor.gray }
    animator.addCompletion { position in
        if position == .end {
            // The lift ended normally, and a drag is now happening
        }
        else if position == .start {
            // The lift was cancelled and the animation returned to the start
        }
    }
}
Drag Interaction Delegate

Session begins and moves

```swift
func dragInteraction(_ interaction: UIDragInteraction,
                      sessionWillBegin session: UIDragSession)

func dragInteraction(_ interaction: UIDragInteraction,
                      sessionAllowsMoveOperation session: UIDragSession) -> Bool
// And more methods asking questions about the new drag session

func dragInteraction(_ interaction: UIDragInteraction,
                      sessionDidMove session: UIDragSession)
```
func dragInteraction(_ interaction: UIDragInteraction,
            sessionWillBegin session: UIDragSession)

func dragInteraction(_ interaction: UIDragInteraction,
            sessionAllowsMoveOperation session: UIDragSession) -> Bool
// And more methods asking questions about the new drag session

func dragInteraction(_ interaction: UIDragInteraction,
            sessionDidMove session: UIDragSession)
Drag Interaction Delegate
Session begins and moves

```swift
func dragInteraction(_ interaction: UIDragInteraction,
                 sessionWillBegin session: UIDragSession)

func dragInteraction(_ interaction: UIDragInteraction,
                 sessionAllowsMoveOperation session: UIDragSession) -> Bool

// And more methods asking questions about the new drag session

func dragInteraction(_ interaction: UIDragInteraction,
                 sessionDidMove session: UIDragSession)
```
Drag Interaction Delegate
Session begins and moves

```
func dragInteraction(_ interaction: UIDragInteraction,
                     sessionWillBegin session: UIDragSession)

func dragInteraction(_ interaction: UIDragInteraction,
                     sessionAllowsMoveOperation session: UIDragSession) -> Bool

// And more methods asking questions about the new drag session

func dragInteraction(_ interaction: UIDragInteraction,
                     sessionDidMove session: UIDragSession)
```
Drag Interaction Delegate
Adding items

Time

Touch
- Begins
- Stays Down
- Moves

Touch 2

Drag and Drop
- Lift
- Drag
Drag Interaction Delegate

Adding items

Time

Touch

Begins
Stays Down
Moves

Touch 2

Tap
Get items to add

Drag and Drop

Lift
Drag
Drag Interaction Delegate
Adding items during the session

```swift
func dragInteraction(_ interaction: UIDragInteraction,
   itemsForAddingTo session: UIDragSession,
   withTouchAt point: CGPoint) -> [UIDragItem]

func dragInteraction(_ interaction: UIDragInteraction,
   previewForLifting item: UIDragItem, session: UIDragSession)
   -> UITargetedDragPreview?
```
Drag Interaction Delegate
Adding items during the session

func dragInteraction(_ interaction: UIDragInteraction,
    itemsForAddingTo session: UIDragSession,
    withTouchAt point: CGPoint) -> [UIDragItem]

func dragInteraction(_ interaction:UIDragInteraction,
    previewForLifting item:UIDragItem, session:UIDragSession)
    ->UITargetedDragPreview?
Drag Interaction Delegate
Adding items during the session

```swift
func dragInteraction(_ interaction: UIDragInteraction,
                      itemsForAddingTo session: UIDragSession,
                      withTouchAt point: CGPoint) -> [UIDragItem]

func dragInteraction(_ interaction: UIDragInteraction,
                      previewForLifting item: UIDragItem, session: UIDragSession)
                      -> UITargetedDragPreview?
```
Drag Interaction Delegate

The session ends

```swift
func dragInteraction(_ interaction: UIDragInteraction,
session: UIDragSession,
willEndWith operation: UIDropOperation)
```
Drag Interaction Delegate

The session ends

```swift
func dragInteraction(_: UIDragInteraction,
    session: UIDragSession,
    willEndWith operation: UIDropOperation)
```
Drag Interaction Delegate
The session ends in a cancel

```swift
func dragInteraction(_ interaction: UIDragInteraction,
                     previewForCancelling item: UIDragItem,
                     withDefault defaultPreview:UITargetedDragPreview)
                    -> UITargetedDragPreview?

func dragInteraction(_ interaction: UIDragInteraction,
                     item: UIDragItem,
                     willAnimateCancelWith animator: UIDragAnimating)

func dragInteraction(_ interaction: UIDragInteraction,
                     session: UIDragSession
                     didEndWith operation: UIDropOperation)
```
Drag Interaction Delegate
The session ends in a cancel

```swift
func dragInteraction(_ interaction: UIDragInteraction,
    previewForCancelling item: UIDragItem,
    withDefault defaultPreview:UITargetedDragPreview)
    -> UITargetedDragPreview?

func dragInteraction(_ interaction: UIDragInteraction,
    item: UIDragItem,
    willAnimateCancelWith animator: UIDragAnimating)

func dragInteraction(_ interaction: UIDragInteraction,
    session: UIDragSession
    didEndWith operation: UIDropOperation)
```
Drag Interaction Delegate

The session ends in a cancel

```swift
func dragInteraction(_: UIDragInteraction, previewForCancelling item: UIDragItem, withDefault defaultPreview:UITargetedDragPreview) -> UITargetedDragPreview?

func dragInteraction(_: UIDragInteraction, item: UIDragItem, willAnimateCancelWith animator: UIDragAnimating)

func dragInteraction(_: UIDragInteraction, session: UIDragSession, didEndWith operation: UIDropOperation)
```
Drag Interaction Delegate

The session ends in a cancel

```swift
func dragInteraction(_ interaction: UIDragInteraction,
    previewForCancelling item: UIDragItem,
    withDefault defaultPreview:UITargetedDragPreview)
->UITargetedDragPreview?

func dragInteraction(_ interaction: UIDragInteraction,
    item:UIDragItem,
    willAnimateCancelWith animator: UIDragAnimating)

func dragInteraction(_ interaction: UIDragInteraction,
    session: UIDragSession
    didEndWith operation: UIDropOperation)
```
Drag Interaction Delegate
The session ends in a copy or move

```swift
func dragInteraction(_ interaction: UIDragInteraction,
session: UIDragSession
didEndWith operation: UIDropOperation)

func dragInteraction(_ interaction: UIDragInteraction,
sessionDidTransferItems session: UIDragSession)
```
Drag Interaction Delegate
The session ends in a copy or move

```swift
func dragInteraction(_ interaction: UIDragInteraction,
    session: UIDragSession
    didEndWith operation: UIDropOperation)

func dragInteraction(_ interaction: UIDragInteraction,
    sessionDidTransferItems session: UIDragSession)
```
Drag Interaction Delegate
The session ends in a copy or move

```swift
func dragInteraction(_ interaction: UIDragInteraction,
                    session: UIDragSession
                    didEndWith operation: UIDropOperation)

func dragInteraction(_ interaction: UIDragInteraction,
                    sessionDidTransferItems session: UIDragSession)
```
Drop Interaction Delegate
Drop Interaction Delegate
Session enters the view

```swift
func dropInteraction(_ interaction: UIDropInteraction,
                       canHandle session: UIDropSession) -> Bool {
    return session.canLoadObjects(ofClass: UIImage.self)
}
```
Drop Interaction Delegate

Session enters the view

```swift
func dropInteraction(_ interaction: UIDropInteraction,
                     canHandle session: UIDropSession) -> Bool {
    return session.canLoadObjects(ofClass: UIImage.self)
}
```
func dropInteraction(_: interaction: UIDropInteraction,
        canHandle session: UIDropSession) -> Bool {
    return session.canLoadObjects(ofClass: UIImage.self)
}
import MobileCoreServices // for kUTTypeImagePNG

func dropInteraction(_ interaction: UIDropInteraction,
                   canHandle session: UIDropSession) -> Bool {
    return session.hasItemsConforming(toTypeIdentifiers: [kUTTypeImagePNG as String])
}
import MobileCoreServices // for kUTTypeImagePNG

func dropInteraction(_ interaction: UIDropInteraction,
             canHandle session: UIDropSession) -> Bool {
    return session.hasItemsConforming(toTypeIdentifiers: [kUTTypeImagePNG as String])
}
Drop Interaction Delegate
Session enters and exits the view

```swift
func dropInteraction(_ interaction: UIDropInteraction,
                     sessionDidEnter session: UIDropSession)

func dropInteraction(_ interaction: UIDropInteraction,
                     sessionDidUpdate session: UIDropSession) -> UIDropProposal

func dropInteraction(_ interaction: UIDropInteraction,
                     sessionDidExit session: UIDropSession)
```
Drop Interaction Delegate
Session enters and exits the view

```swift
func dropInteraction(_ interaction: UIDropInteraction,
                     sessionDidEnter session: UIDropSession)

func dropInteraction(_ interaction: UIDropInteraction,
                     sessionDidUpdate session: UIDropSession) -> UIDropProposal

func dropInteraction(_ interaction: UIDropInteraction,
                     sessionDidExit session: UIDropSession)
```
Drop Interaction Delegate
Session enters and exits the view

```swift
func dropInteraction(_ interaction: UIDropInteraction,
                     sessionDidEnter session: UIDropSession)

func dropInteraction(_ interaction: UIDropInteraction,
                     sessionDidUpdate session: UIDropSession) -> UIDropProposal

func dropInteraction(_ interaction: UIDropInteraction,
                     sessionDidExit session: UIDropSession)
```
Drop Interaction Delegate
Session enters and exits the view

```swift
func dropInteraction(_ interaction: UIDropInteraction,
sessionDidEnter session: UIDropSession)

func dropInteraction(_ interaction: UIDropInteraction,
sessionDidUpdate session: UIDropSession) -> UIDropProposal

func dropInteraction(_ interaction: UIDropInteraction,
sessionDidExit session: UIDropSession)
```
Springloading
When session hovers over a view

```swift
let button = UIButton()
button.isSpringLoaded = true

let springLoadedInteraction = UISpringLoadedInteraction { (interaction, context) in
    // Activate springloading here
}
view.addInteraction(springLoadedInteraction)
```
Springloading
When session hovers over a view

```swift
let button = UIButton()
button.isSpringLoaded = true

let springLoadedInteraction = UISpringLoadedInteraction { (interaction, context) in
    // Activate springloading here
}
view.addInteraction(springLoadedInteraction)
```
Springloading
When session hovers over a view

```swift
let button = UIButton()
button.isSpringLoaded = true

let springLoadedInteraction = UISpringLoadedInteraction {
    (interaction, context) in
    // Activate springloading here
}

view.addInteraction(springLoadedInteraction)
```
Springloading
When session hovers over a view

```swift
let button = UIButton()
button.isSpringLoaded = true

let springLoadedInteraction = UISpringLoadedInteraction { (interaction, context) in
    // Activate springloading here
}

view.addInteraction(springLoadedInteraction)
```
Drop Interaction Delegate
Session ends over a different view

```swift
func dropInteraction(_ interaction: UIDropInteraction,
    sessionId: String) {
    // Implementation
}
```
func dropInteraction(_ interaction: UIDropInteraction,
sessionDidEnd session: UIDropSession)
Drop Interaction Delegate
Session ends over this view

Time

Touch
Ends

Drag and Drop
Perform
Drop Animations
Data Transfer
func dropInteraction(_ interaction: UIDropInteraction,
    previewForDropping item: UIDragItem,
    withDefault defaultPreview:UITargetedDragPreview)
    -> UITargetedDragPreview?

func dropInteraction(_ interaction: UIDropInteraction,
    item: UIDragItem,
    willAnimateDropWith animator: UIDragAnimating)

func dropInteraction(_ interaction: UIDropInteraction,
    concludeDrop session: UIDropSession)
Drop Interaction Delegate

Drop animations

```swift
func dropInteraction(_ interaction: UIDropInteraction,
    previewForDropping item: UIDragItem,
    withDefault defaultPreview: UITargetedDragPreview)
    -> UITargetedDragPreview?

func dropInteraction(_ interaction: UIDropInteraction,
    item: UIDragItem,
    willAnimateDropWith animator: UIDragAnimating)

func dropInteraction(_ interaction: UIDropInteraction,
    concludeDrop session: UIDropSession)
```
Drop Interaction Delegate

Drop animations

```swift
func dropInteraction(_ interaction: UIDropInteraction,
    previewForDropping item: UIDragItem,
    withDefault defaultPreview:UITargetedDragPreview)
    -> UITargetedDragPreview?

func dropInteraction(_ interaction: UIDropInteraction,
    item: UIDragItem,
    willAnimateDropWith animator: UIDragAnimating)

func dropInteraction(_ interaction: UIDropInteraction,
    concludeDrop session: UIDropSession)
```
func dropInteraction(_ interaction: UIDropInteraction,
    previewForDropping item: UIDragItem,
    withDefault defaultPreview: UITargetedDragPreview)
-> UITargetedDragPreview?

func dropInteraction(_ interaction: UIDropInteraction,
    item: UIDragItem,
    willAnimateDropWith animator: UIDragAnimating)

func dropInteraction(_ interaction: UIDropInteraction,
    concludeDrop session: UIDropSession)
let progress = item.itemProvider.loadObject(ofClass: UIImage.self) { (object, error) in
    // Closure is called when object or error are available
}

let fractionCompleted = progress.fractionCompleted
let isFinished = progress.isFinished
progress.cancel()

let sessionProgress = session.progress
let progress = item.itemProvider.loadObject(ofClass: UIImage.self) { (object, error) in
    // Closure is called when object or error are available
}

let fractionCompleted = progress.fractionCompleted
let isFinished = progress.isFinished
progress.cancel()

let sessionProgress = session.progress
let progress = item.itemProvider.loadObject(ofClass: UIImage.self) { (object, error) in
    // Closure is called when object or error are available
}

let fractionCompleted = progress.fractionCompleted
let isFinished = progress.isFinished
progress.cancel()

let sessionProgress = session.progress
let progress = item.itemProvider.loadObject(ofClass: UImage.self) { (object, error) in
  // Closure is called when object or error are available
}

let fractionCompleted = progress.fractionCompleted
let isFinished = progress.isFinished
progress.cancel()

let sessionProgress = session.progress
let progress = item.itemProvider.loadObject(ofClass: UImage.self) { (object, error) in
    // Closure is called when object or error are available
}

let fractionCompleted = progress.fractionCompleted
let isFinished = progress.isFinished
progress.cancel()

let sessionProgress = session.progress
let progress = item.itemProvider.loadObject(ofClass: UIImage.self) { (object, error) in
// Closure is called when object or error are available
}

let fractionCompleted = progress.fractionCompleted
let isFinished = progress.isFinished
progress.cancel()

let sessionProgress = session.progress
Drag and Drop Timeline

Time

Touch
- Begins
- Stays Down
- Moves
- Ends

Drag and Drop
- Lift
- Drag
- Perform
- Drop Animations
- Data Transfer
- Cancel Animations
Interaction Delegates

Essential functions

```swift
define dragInteraction(_ interaction: UIDragInteraction,
itemsForBeginning session: UIDragSession) -> [UIDragItem]
define dropInteraction(_ interaction: UIDropInteraction,
sessionDidChange session: UIDropSession) -> UIDropProposal
define dropInteraction(_ interaction: UIDropInteraction,
performDrop session: UIDropSession)
```
To customize drag and drop, use the interaction delegates.
Demo

Emanuele Rudel, UIKit Engineer
Next Steps
Next Steps

Explore the system
Next Steps

Explore the system

Try adding a drop target
Next Steps

Explore the system

Try adding a drop target

Enable a drag source
Next Steps

Explore the system

Try adding a drop target

Enable a drag source

Spring load some of your controls
Next Steps

Explore the system
Try adding a drop target
Enable a drag source
Spring load some of your controls
Dig deeper into the Drag and Drop APIs
Explore the APIs

Drag Interaction Delegate → Drag Session → Drag Interaction

Drag Item

Preview

Drop Session → Drop Interaction Delegate
Explore the APIs

- Drag Interaction Delegate
- Drag Session
- Drag Item
- Preview
- Drop Session
- Drop Interaction Delegate
- Item Provider
Explore the APIs

Drag Interaction

Drag Interaction Delegate

Drop Session

Drag Session

Drag Item

Preview

Drop Session

Drop Interaction Delegate

Drop Interaction

Item Provider
Explore the APIs

Data Delivery with Drag and Drop
Hall 2
Thursday 10:00AM
Explore the APIs

- Drag Interaction
  - Drag Interaction Delegate
  - Drag Session
- Preview
- Drag Item
  - Item Provider
- Drop Session
- Drop Interaction
  - Drop Interaction Delegate
Explore the APIs

UICollectionView
UITableView
UITextView
UITextField
WKWebView

Drag Interaction
Drag Interaction Delegate
Drag Session
Drop Session
Drop Interaction
Drop Interaction Delegate
Preview
Item Provider
Explore the APIs

UICollectionView
PHPUnit
UITableView
UITableView
UITextView
UITextField
UITextField
WKWebView
WKWebView

Drag Interaction
Drag Interaction Delegate
Drag Session
Drag Item
Item Provider
Preview
Drop Session
Drop Interaction
Drop Interaction Delegate
Explore the APIs

- UICollectionView
- UITableView
- UITextView
- UITextField
- WKWebView

Drag and Drop with Collection and Table View
Hall 2
Thursday 9:00AM
More Information

https://developer.apple.com/wwdc17/203
## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mastering Drag and Drop</td>
<td>Executive Ballroom</td>
<td>Wednesday 11:00AM</td>
</tr>
<tr>
<td>Data Delivery with Drag and Drop</td>
<td>Hall 2</td>
<td>Thursday 10:00AM</td>
</tr>
<tr>
<td>Drag and Drop with Collection and Table View</td>
<td>Hall 2</td>
<td>Thursday 9:00AM</td>
</tr>
<tr>
<td>File Provider Enhancements</td>
<td>Hall 3</td>
<td>Friday 11:00AM</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>
<tr>
<td>Lab</td>
<td>Location</td>
<td>Time</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>-------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>UIKit and Drag and Drop Lab</td>
<td>Technology Lab C</td>
<td>Tues 1:50–4:10PM</td>
</tr>
<tr>
<td>Cocoa Touch Lab</td>
<td>Technology Lab I</td>
<td>Wed 3:10–6:00PM</td>
</tr>
<tr>
<td>UIKit and Collection View Lab</td>
<td>Technology Lab B</td>
<td>Thur 11:00AM–12:30PM</td>
</tr>
<tr>
<td>Cocoa Touch and Haptics Lab</td>
<td>Technology Lab C</td>
<td>Fri 12:00–1:50PM</td>
</tr>
</tbody>
</table>