App Frameworks

#WWDC17

# Mastering Drag and Drop

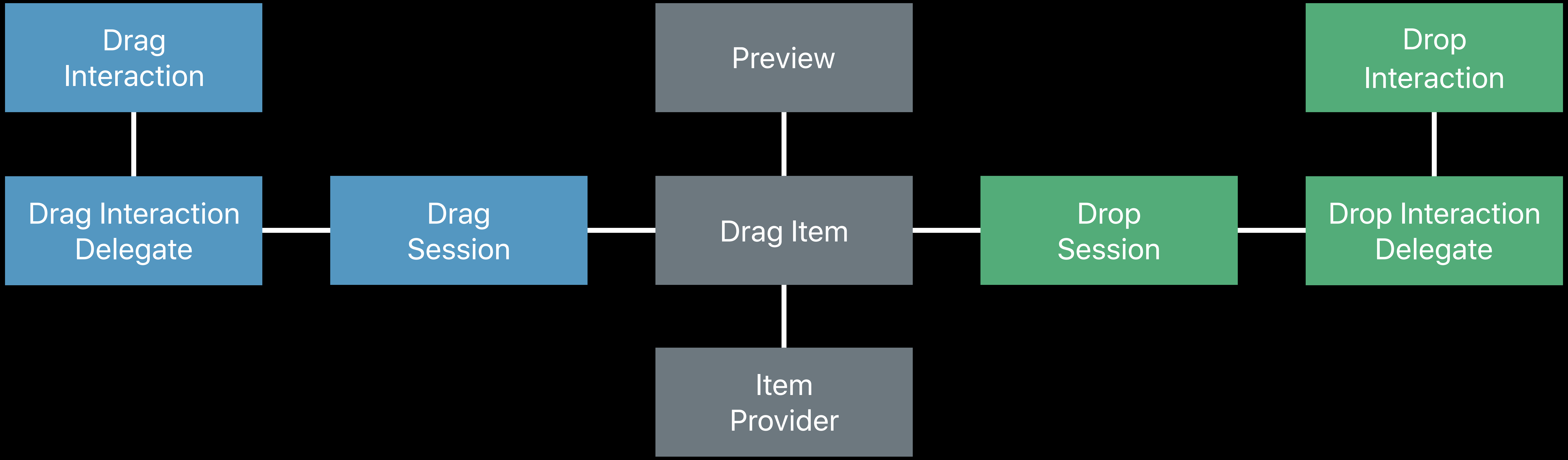
Session 213

Tom Adriaenssen, UIKit
Wenson Hsieh, WebKit
Robb Böhnke, UIKit

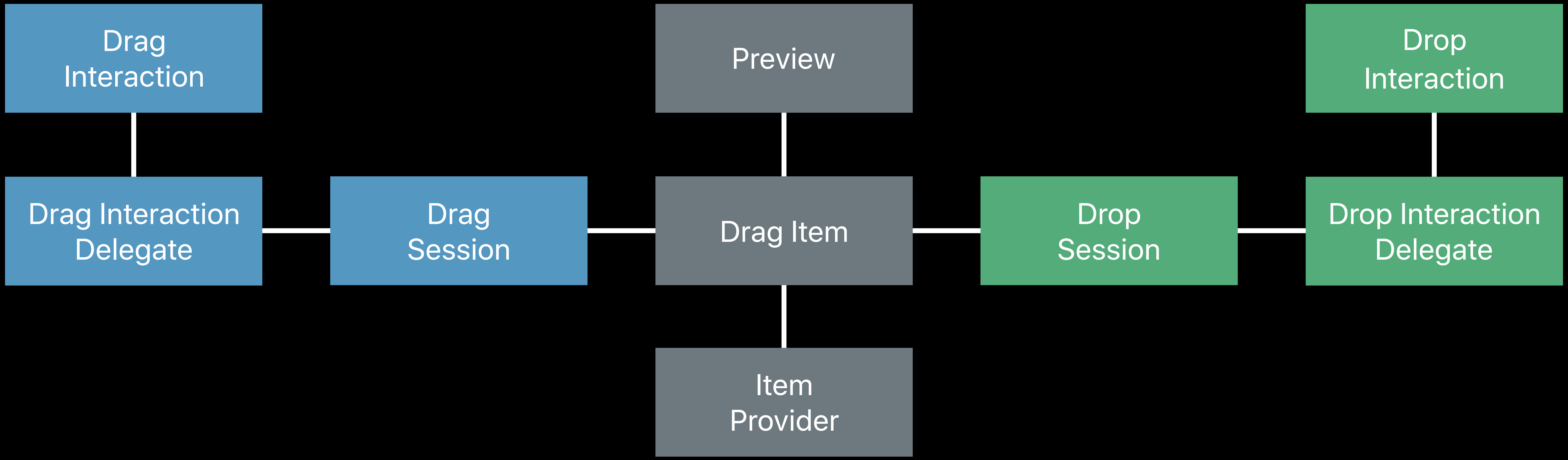
© 2017 Apple Inc. All rights reserved. Redistribution or public display not permitted without written permission from Apple.

NEW

# Drag and Drop API Roadmap

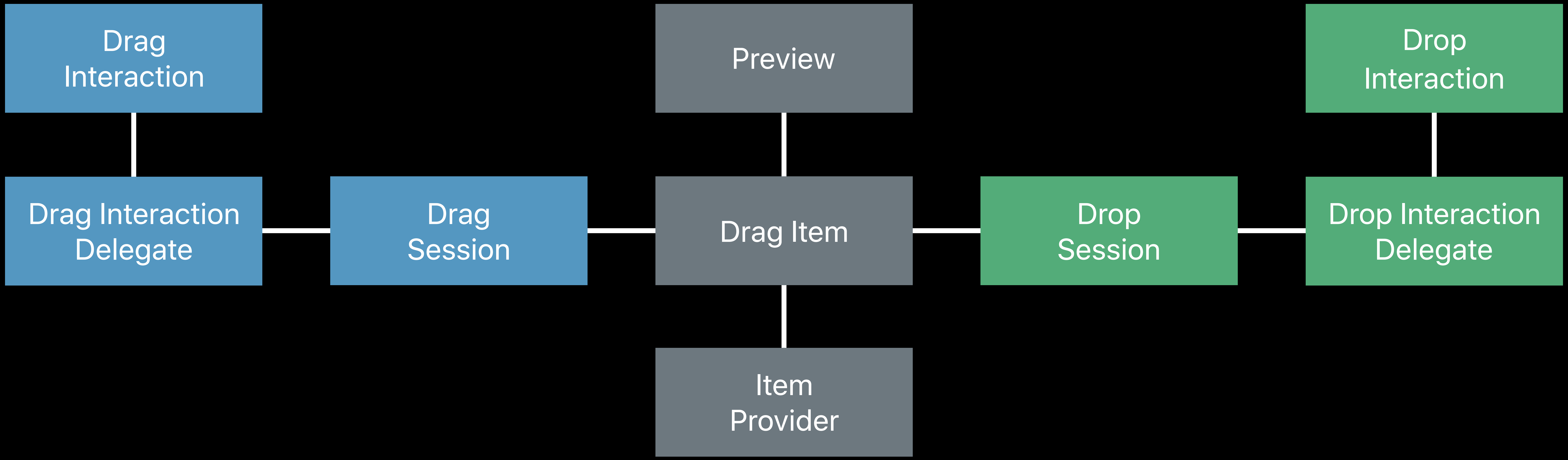


# Drag and Drop API Roadmap



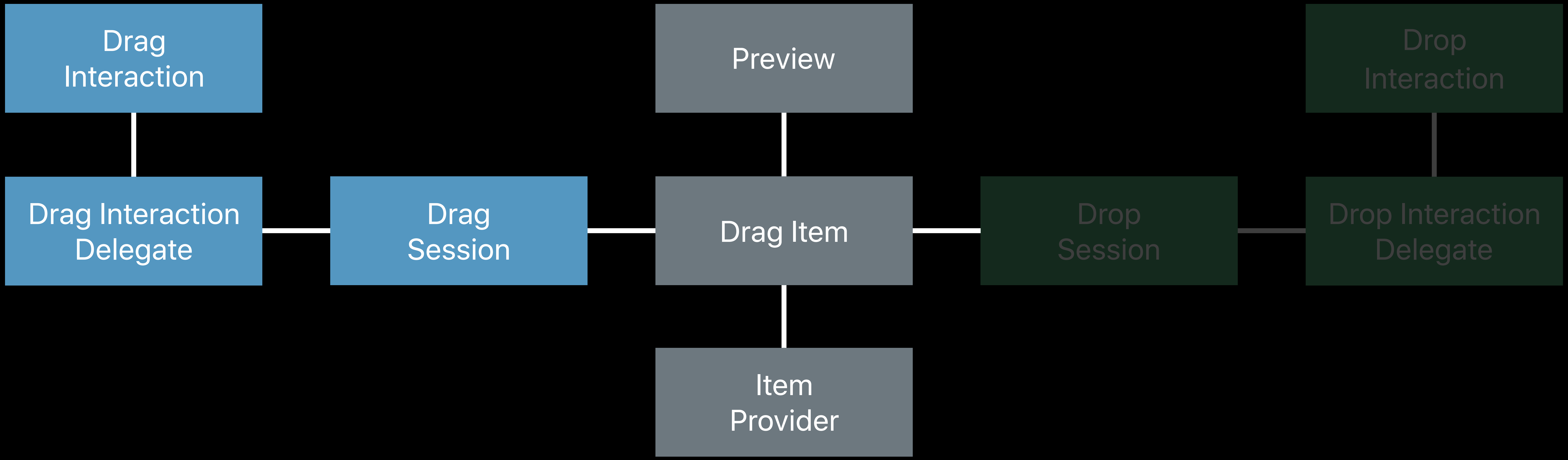
Drag and Drop with Collection and Table View | Hall 2 | Thursday 9:00AM

## Drag and Drop API Roadmap

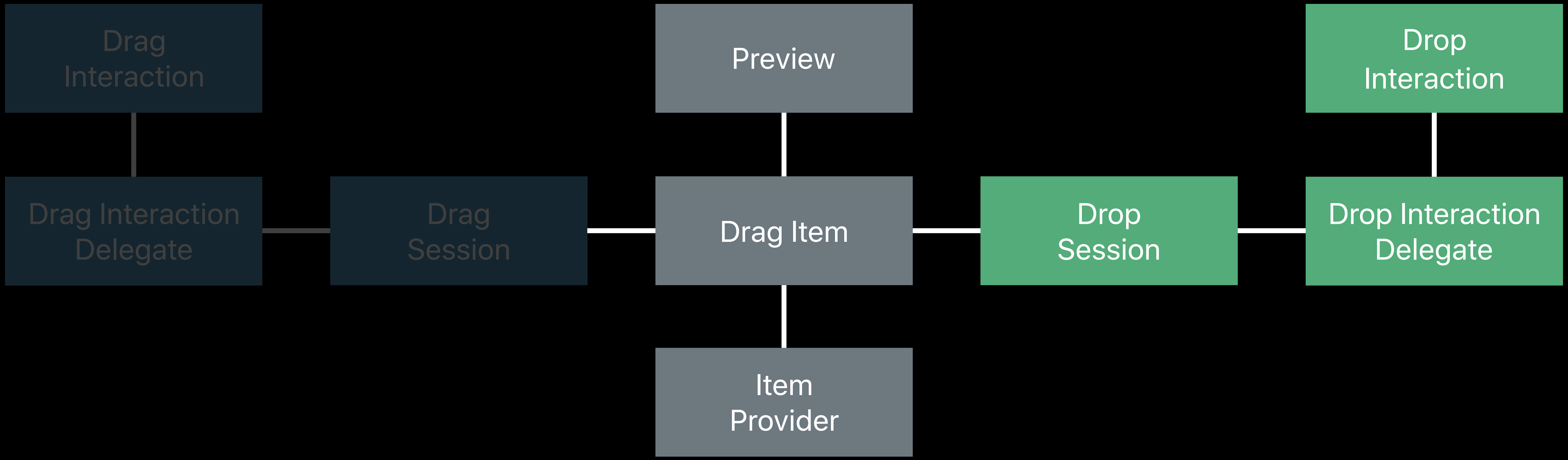


Data Delivery with Drag and Drop | Hall 2 | Thursday 10:00AM

# Drag and Drop API Roadmap



# Drag and Drop API Roadmap



## Advanced Drag Interactions

Robb Böhnke, UIKit

## UIDragInteraction

Can be added to any view

Installs gestures

Delegate does the work

## UIDragInteractionDelegate

Returns UIDragItems associated with the drag

May provide a preview during the lift or for cancelling

May can animate alongside the lift

May get notified of drag session lifecycle

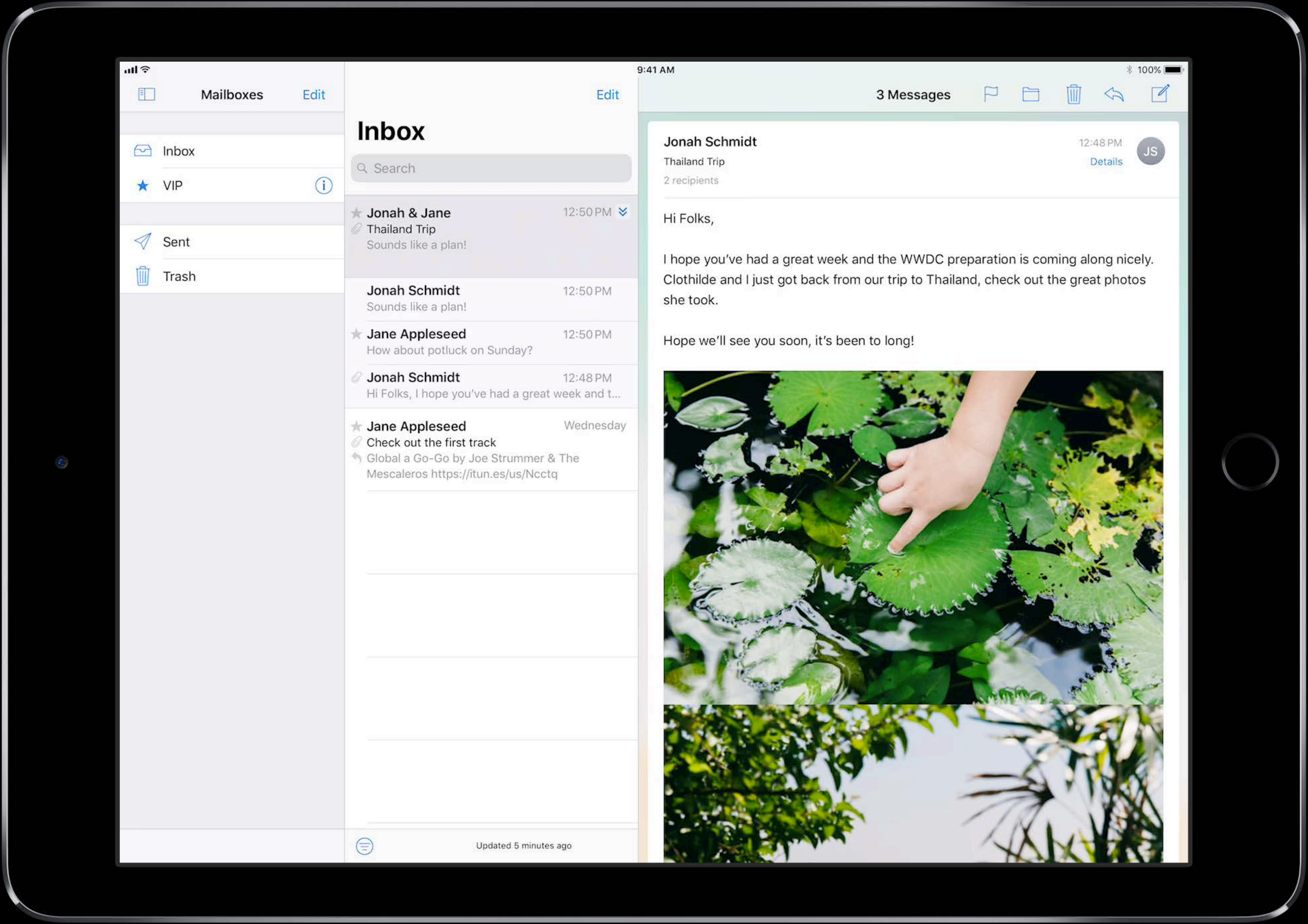
## UIDragInteractionDelegate

Returns UIDragItems associated with the drag

May provide a preview during the lift or for cancelling

May can animate alongside the lift

May get notified of drag session lifecycle



9:41 AM 100%

Mailboxes Edit

Inbox

VIP

Sent

Trash

Edit

Inbox

Search

Jonah & Jane 12:50 PM
Thailand Trip
Sounds like a plan!

Jonah Schmidt 12:50 PM
Sounds like a plan!

Jane Appleseed 12:50 PM
How about potluck on Sunday?

Jonah Schmidt 12:48 PM
Hi Folks, I hope you've had a great week and t...

Jane Appleseed Wednesday
Check out the first track
Global a Go-Go by Joe Strummer & The Mescaleros https://itun.es/us/Ncctq

Updated 5 minutes ago

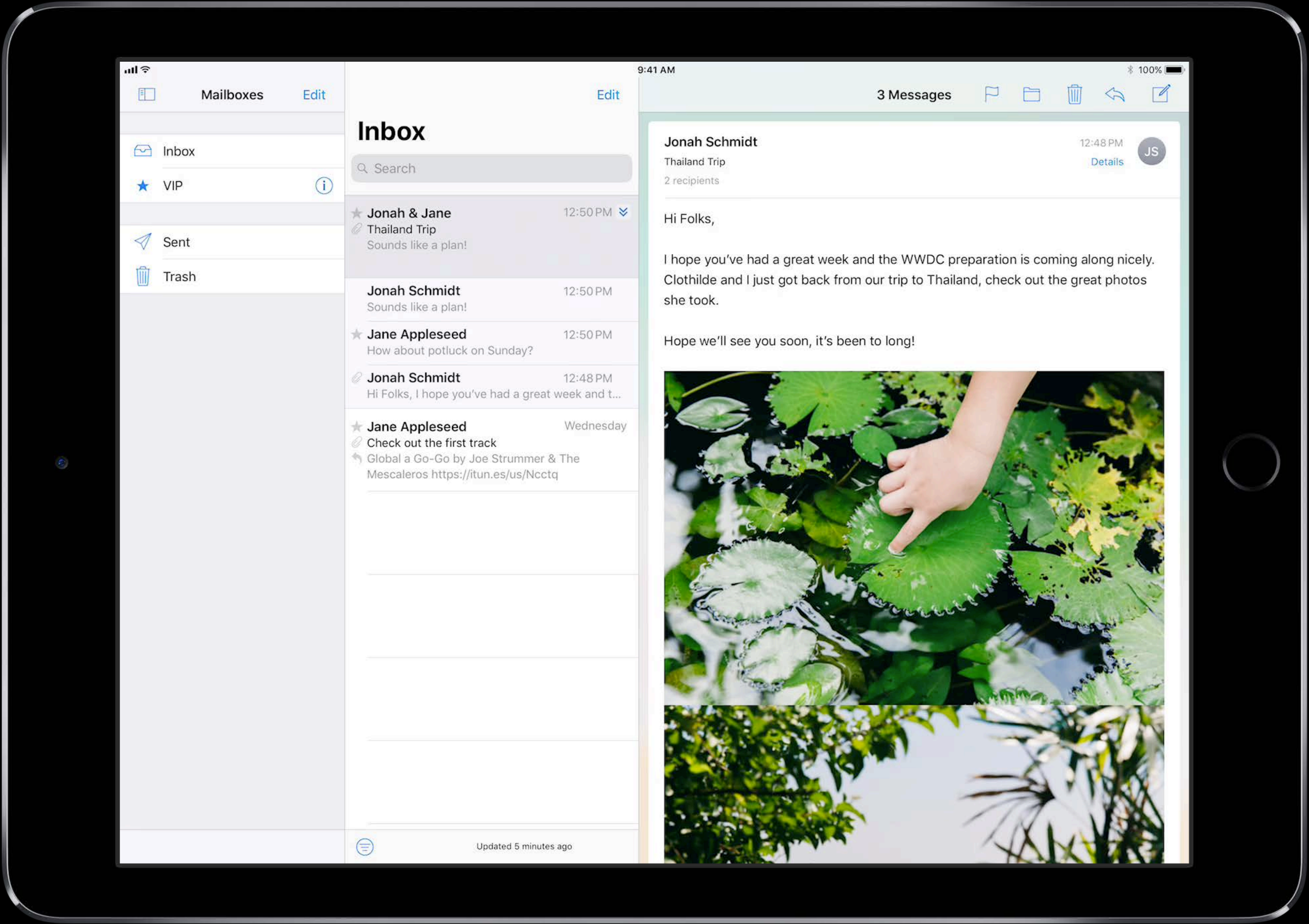
3 Messages

Jonah Schmidt 12:48 PM
Thailand Trip Details
2 recipients JS

Hi Folks,

I hope you've had a great week and the WWDC preparation is coming along nicely. Clothilde and I just got back from our trip to Thailand, check out the great photos she took.

Hope we'll see you soon, it's been to long!



9:41 AM 100%

Mailboxes Edit

Inbox

VIP

Sent

Trash

Edit

# Inbox

Search

Jonah & Jane 12:50 PM
Thailand Trip
Sounds like a plan!

Jonah Schmidt 12:50 PM
Sounds like a plan!

Jane Appleseed 12:50 PM
How about potluck on Sunday?

Jonah Schmidt 12:48 PM
Hi Folks, I hope you've had a great week and t...

Jane Appleseed Wednesday
Check out the first track
Global a Go-Go by Joe Strummer & The Mescaleros https://itun.es/us/Ncctq

Updated 5 minutes ago

3 Messages

Jonah Schmidt 12:48 PM
Thailand Trip Details
2 recipients JS

Hi Folks,

I hope you've had a great week and the WWDC preparation is coming along nicely. Clothilde and I just got back from our trip to Thailand, check out the great photos she took.

Hope we'll see you soon, it's been to long!

```
// UIDragInteractionDelegate
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    let itemProvider = NSItemProvider(message)
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    let itemProvider = NSItemProvider(message)

    let dragItem = UIDragItem(itemProvider: itemProvider)
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    let itemProvider = NSItemProvider(message)

    let dragItem = UIDragItem(itemProvider: itemProvider)
    dragItem.localObject = message
```

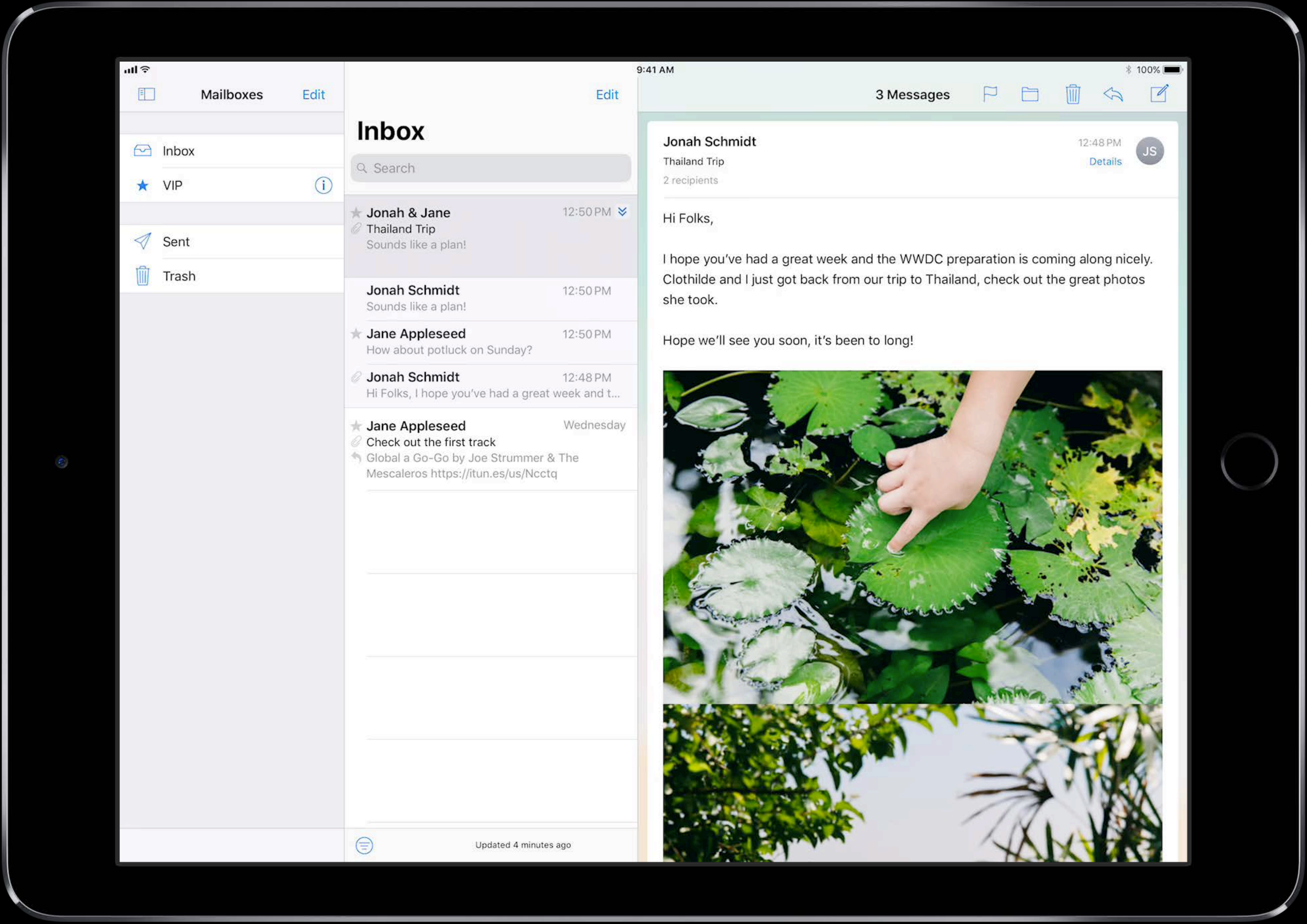
```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    let itemProvider = NSItemProvider(message)

    let dragItem = UIDragItem(itemProvider: itemProvider)
    dragItem.localObject = message

    return [ dragItem ]
}
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction, itemsForAddingTo session: UIDragSession,
                     withTouchAt point: CGPoint) -> [UIDragItem] {
    let itemProvider = NSItemProvider(message)

    let dragItem = UIDragItem(itemProvider: itemProvider)
    dragItem.localObject = message

    return [ dragItem ]
}
```

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







    let itemProvider = NSItemProvider(message)

    let dragItem = UIDragItem(itemProvider: itemProvider)
    dragItem.localObject = message

    return [ dragItem ]
}
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction, itemsForAddingTo session: UIDragSession,
                     withTouchAt point: CGPoint) -> [UIDragItem] {
    for item in session.items {
        guard item.itemProvider.hasItemConformingToTypeIdentifier("private.example.mail")
        else { return [] }



    }

    let itemProvider = NSItemProvider(message)

    let dragItem = UIDragItem(itemProvider: itemProvider)
    dragItem.localObject = message

    return [ dragItem ]
}
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction, itemsForAddingTo session: UIDragSession,
                     withTouchAt point: CGPoint) -> [UIDragItem] {
    for item in session.items {
        guard item.itemProvider.hasItemConformingToTypeIdentifier("private.example.mail")
        else { return [] }

        guard item.localObject != message { return [] }
    }

    let itemProvider = NSItemProvider(message)

    let dragItem = UIDragItem(itemProvider: itemProvider)
    dragItem.localObject = message

    return [ dragItem ]
}
```



9:41 AM 100%

Mailboxes Edit

Inbox

VIP

Sent

Trash

Edit

Inbox

Search

Jonah & Jane 12:50 PM
Thailand Trip
Sounds like a plan!

Jonah Schmidt 12:50 PM
Sounds like a plan!

Jane Appleseed 12:50 PM
How about potluck on Sunday?

Jonah Schmidt 12:48 PM
Hi Folks, I hope you've had a great week and t...

Jane Appleseed Wednesday
Check out the first track
Global a Go-Go by Joe Strummer & The Mescaleros https://itun.es/us/Ncctq

Updated 4 minutes ago

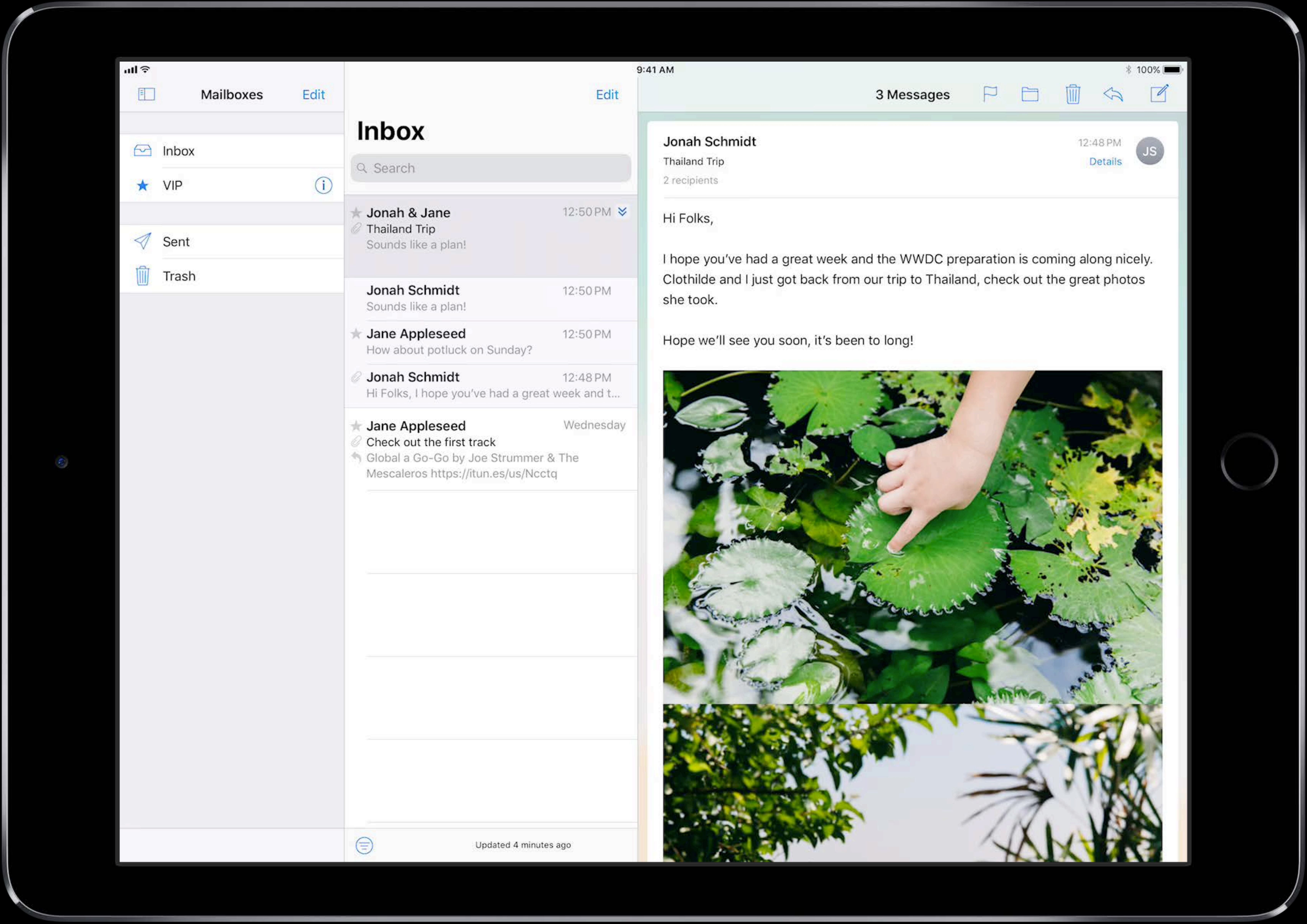
3 Messages

Jonah Schmidt 12:48 PM
Thailand Trip Details
2 recipients JS

Hi Folks,

I hope you've had a great week and the WWDC preparation is coming along nicely. Clothilde and I just got back from our trip to Thailand, check out the great photos she took.

Hope we'll see you soon, it's been to long!



9:41 AM 100%

Mailboxes Edit

Inbox

VIP

Sent

Trash

Edit

Inbox

Search

Jonah & Jane 12:50 PM
Thailand Trip
Sounds like a plan!

Jonah Schmidt 12:50 PM
Sounds like a plan!

Jane Appleseed 12:50 PM
How about potluck on Sunday?

Jonah Schmidt 12:48 PM
Hi Folks, I hope you've had a great week and t...

Jane Appleseed Wednesday
Check out the first track
Global a Go-Go by Joe Strummer & The Mescaleros https://itun.es/us/Ncctq

Updated 4 minutes ago

3 Messages

Jonah Schmidt 12:48 PM
Thailand Trip Details
2 recipients
JS

Hi Folks,

I hope you've had a great week and the WWDC preparation is coming along nicely. Clothilde and I just got back from our trip to Thailand, check out the great photos she took.

Hope we'll see you soon, it's been to long!

```
// UIDragInteractionDelegate
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    return mailThread
        .messages
```

```
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    return mailThread
        .messages
        .sorted { a, b in
            return a.sentDate.timeIntervalSince1970 > b.sentDate.timeIntervalSince1970
        }
```

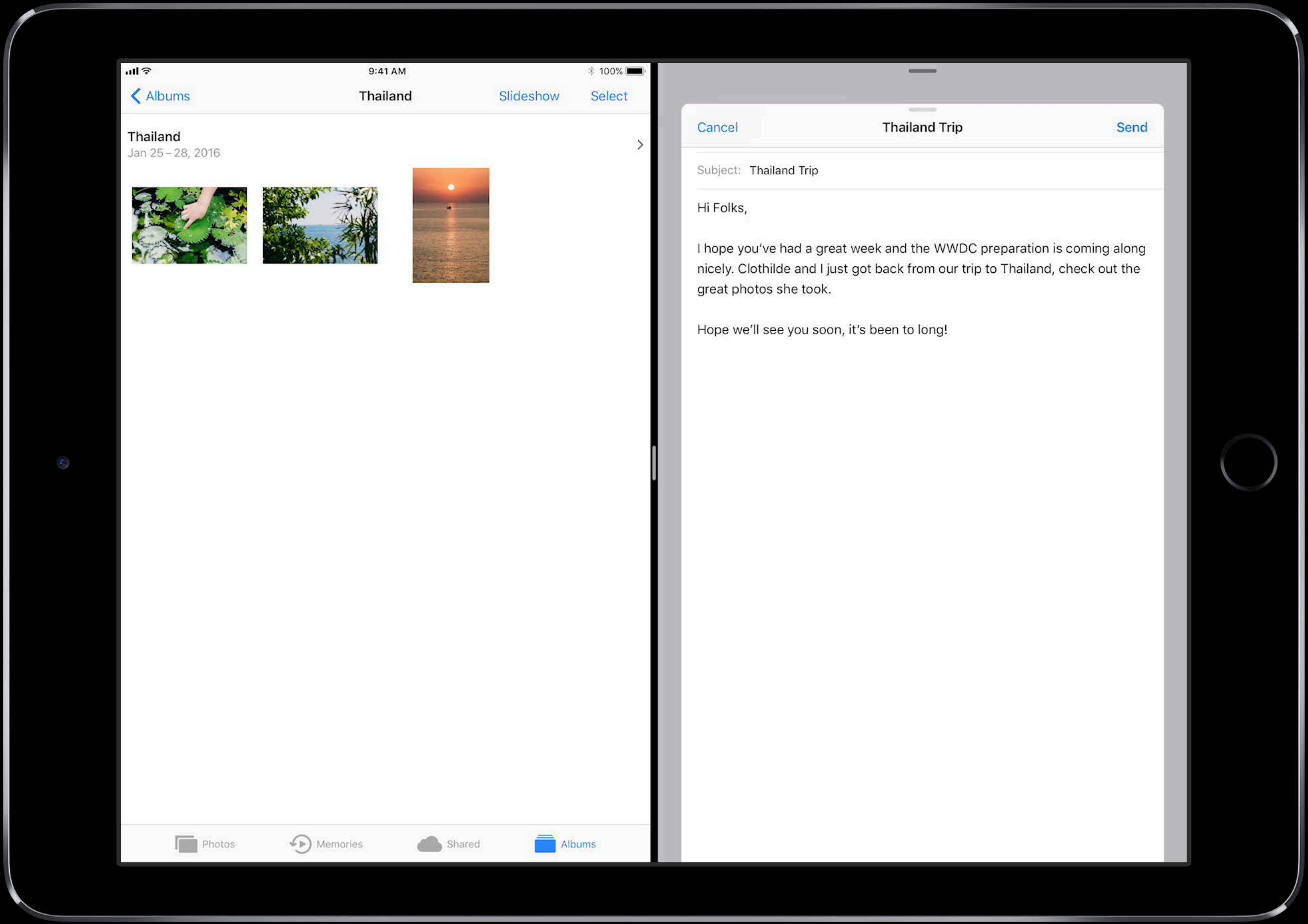
```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    return mailThread
        .messages
        .sorted { a, b in
            return a.sentDate.timeIntervalSince1970 > b.sentDate.timeIntervalSince1970
        }
        .map { message -> UIDragItem in
            let itemProvider = NSItemProvider(message)

            let dragItem = UIDragItem(itemProvider: itemProvider)
            dragItem.localObject = message

            return dragItem
        }
}
```

```
// UIDragInteractionDelegate
```

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

```
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     previewForLifting item: UIDragItem,
                     session: UIDragSession) -> UITargetedDragPreview?
    guard let message = item.localObject as? Message else {
        return nil
    }
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     previewForLifting item: UIDragItem,
                     session: UIDragSession) -> UITargetedDragPreview?
    guard let message = item.localObject as? Message else {
        return nil
    }

    return UITargetedDragPreview(view: getMessageView(for: message))
}
```



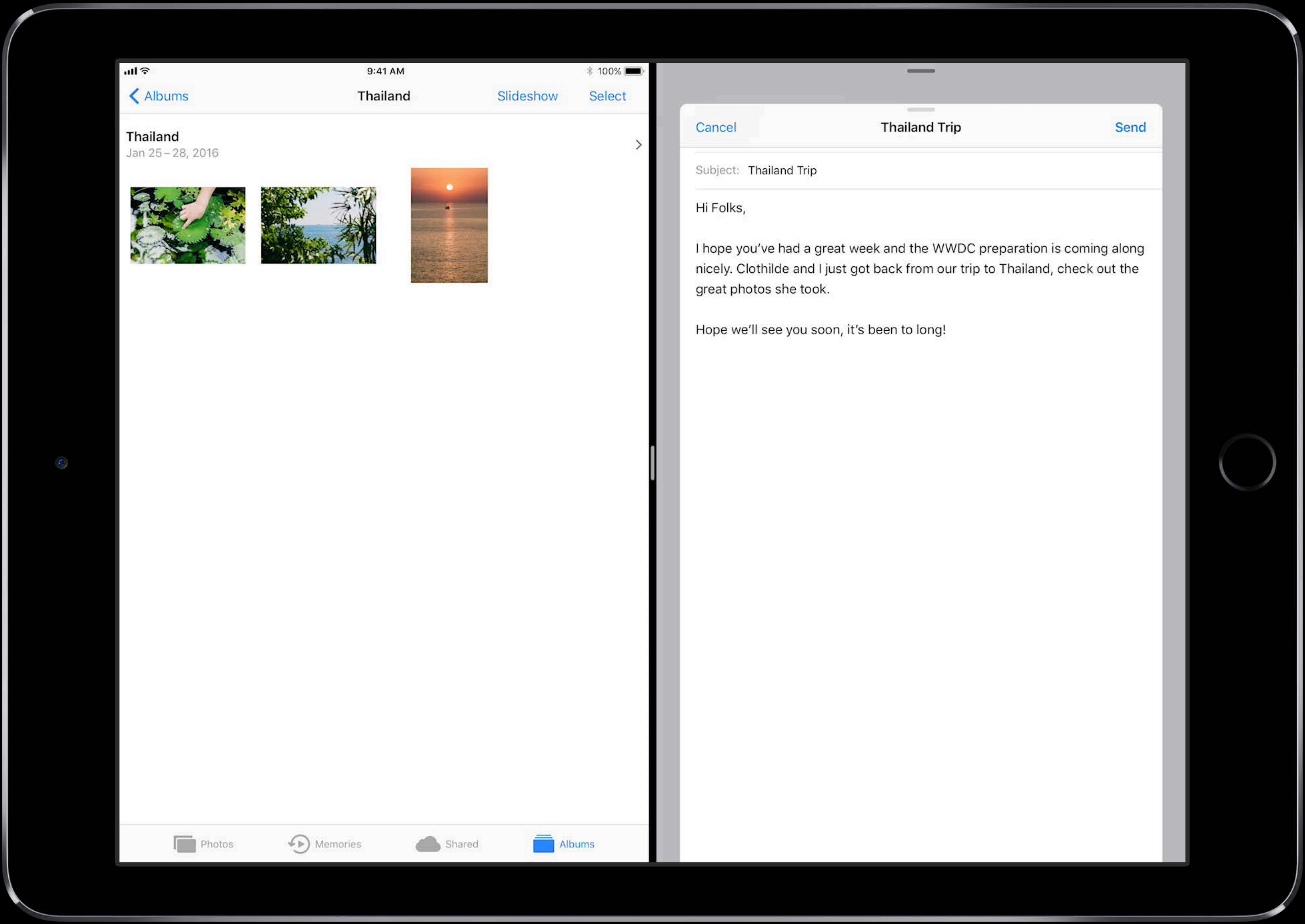
Thailand
Jan 25 – 28, 2016

Subject: Thailand Trip

Hi Folks,

I hope you've had a great week and the WWDC preparation is coming along nicely. Clothilde and I just got back from our trip to Thailand, check out the great photos she took.

Hope we'll see you soon, it's been to long!



Subject: Thailand Trip

Hi Folks,

I hope you've had a great week and the WWDC preparation is coming along nicely. Clothilde and I just got back from our trip to Thailand, check out the great photos she took.

Hope we'll see you soon, it's been to long!

```
// UIDragInteractionDelegate
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    let itemProvider = NSItemProvider(file)
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    let itemProvider = NSItemProvider(file)
    // Available in the next seed
    itemProvider.preferredPresentationSize = CGSize(width: 400, height: 300)
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    let itemProvider = NSItemProvider(file)
    // Available in the next seed
    itemProvider.preferredPresentationSize = CGSize(width: 400, height: 300)

    let dragItem = UIDragItem(itemProvider: itemProvider)

    return [ dragItem ]
}
```

## *Demo*

Wenson Hsieh, WebKit

## Customizing Drag Visuals

Robb Böhnke, UIKit

## Animating Alongside the Lift

The view being lifted is live

Lift is interactive

```
// UIDragInteractionDelegate
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     willAnimateLiftWith animator: UIDragAnimating, session: UIDragSession) {
```

```
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     willAnimateLiftWith animator: UIDragAnimating, session: UIDragSession) {
    session.items.lazy
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     willAnimateLiftWith animator: UIDragAnimating, session: UIDragSession) {
    session.items.lazy
        .flatMap { $0.localObject as? Message }
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     willAnimateLiftWith animator: UIDragAnimating, session: UIDragSession) {
    session.items.lazy
        .flatMap { $0.localObject as? Message }
        .map(getMessageView)
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     willAnimateLiftWith animator: UIDragAnimating, session: UIDragSession) {
    session.items.lazy
        .flatMap { $0.localObject as? Message }
        .map(getMessageView)
        .forEach { messageView in
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     willAnimateLiftWith animator: UIDragAnimating, session: UIDragSession) {
    session.items.lazy
        .flatMap { $0.localObject as? Message }
        .map(getMessageView)
        .forEach { messageView in
            animator.addAnimations {
                messageView.overlay.alpha = 0
            }
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     willAnimateLiftWith animator: UIDragAnimating, session: UIDragSession) {
    session.items.lazy
        .flatMap { $0.localObject as? Message }
        .map(getMessageView)
        .forEach { messageView in
            animator.addAnimations {
                messageView.overlay.alpha = 0
            }

            animator.addCompletion { position in
                messageView.overlay.alpha = 1
            }
        }
}
```

## UITargetedDragPreview

## UITargetedDragPreview

View

## UITargetedDragPreview

View

Parameters

## UITargetedDragPreview

View

Parameters

Target

## UITargetedDragPreview

View

Parameters

Target

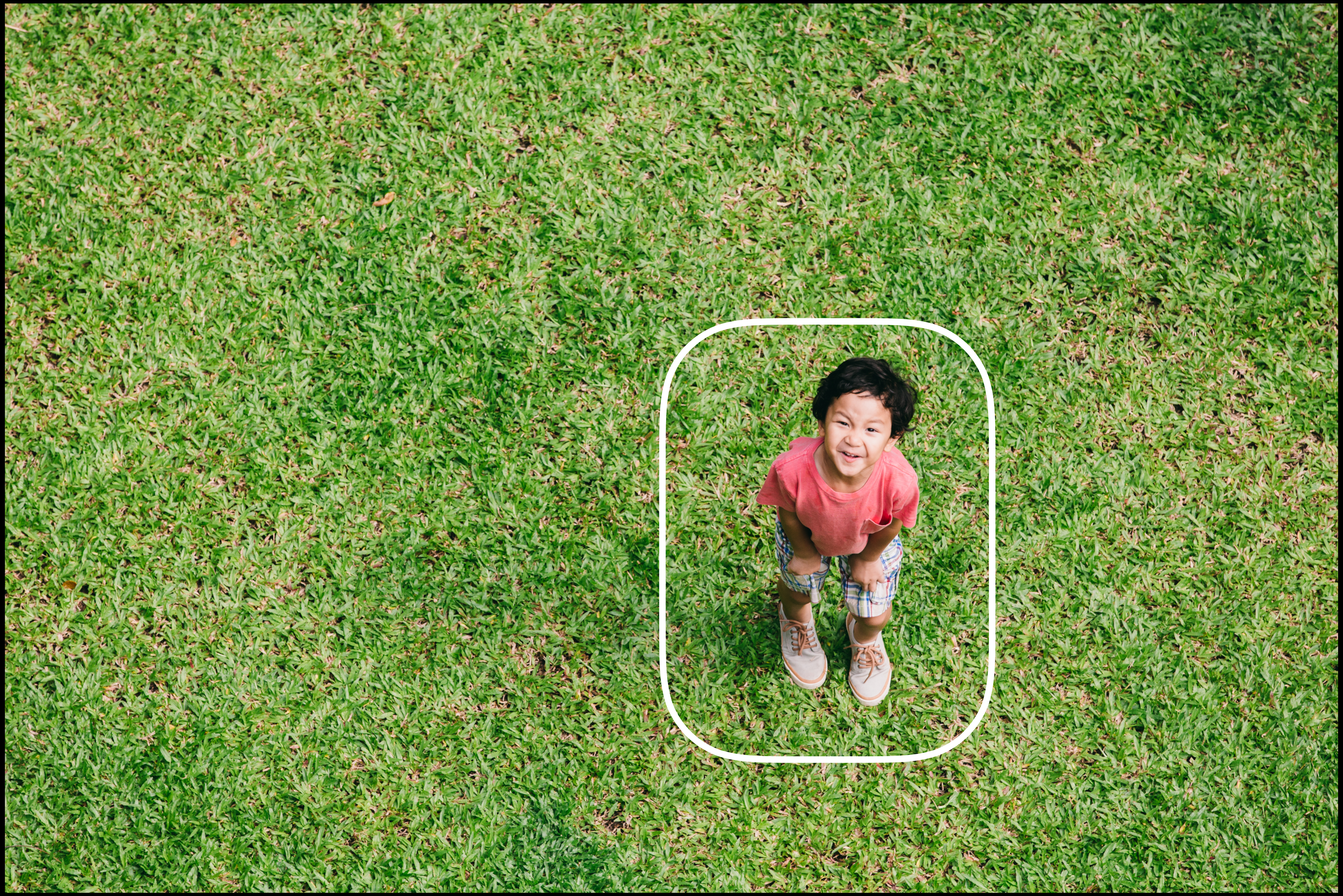
## UIDragPreviewParameters

Can color the background behind the view

Knows which portion of the preview is visible

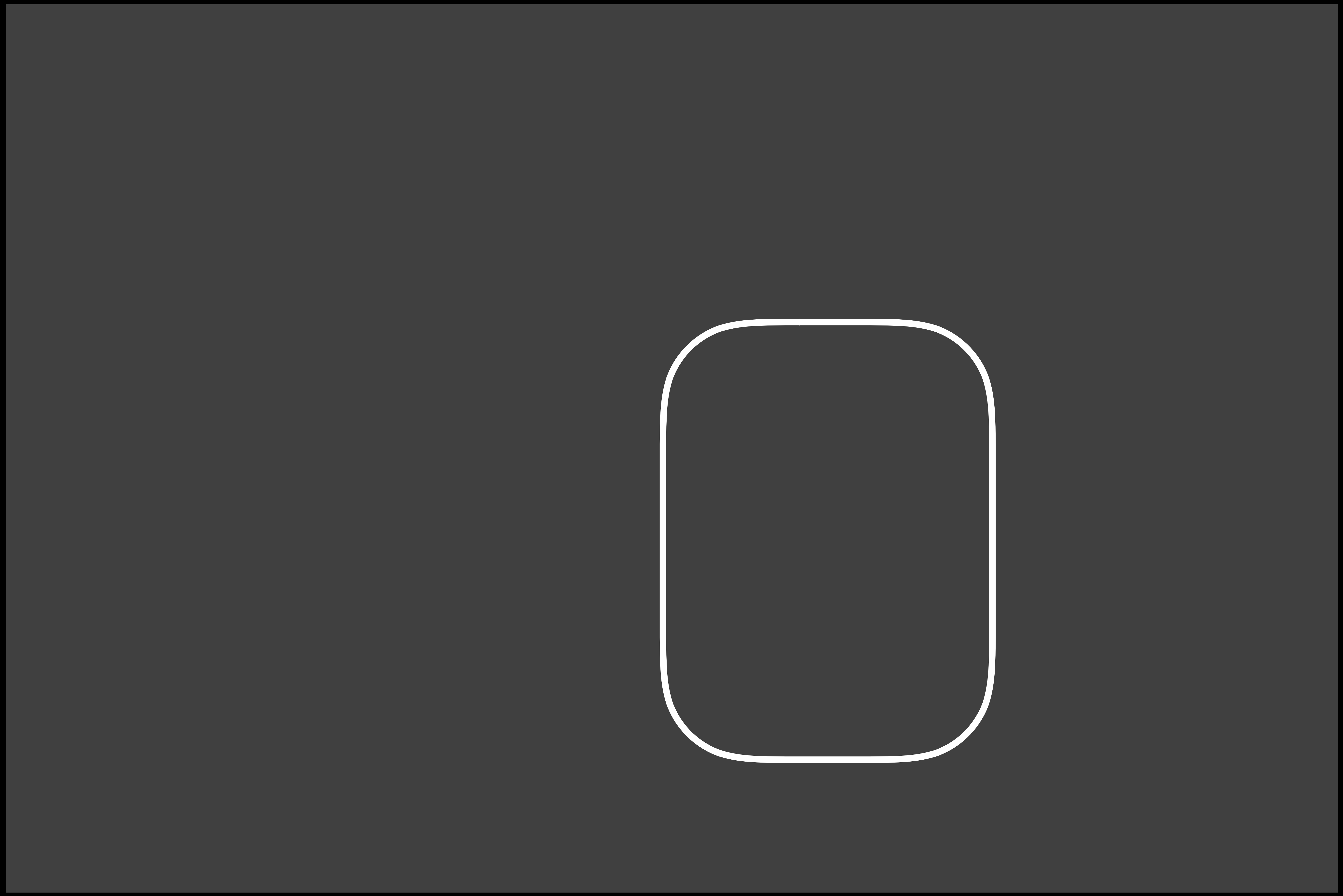
# UIDragPreviewParameters

## UIDragPreviewParameters

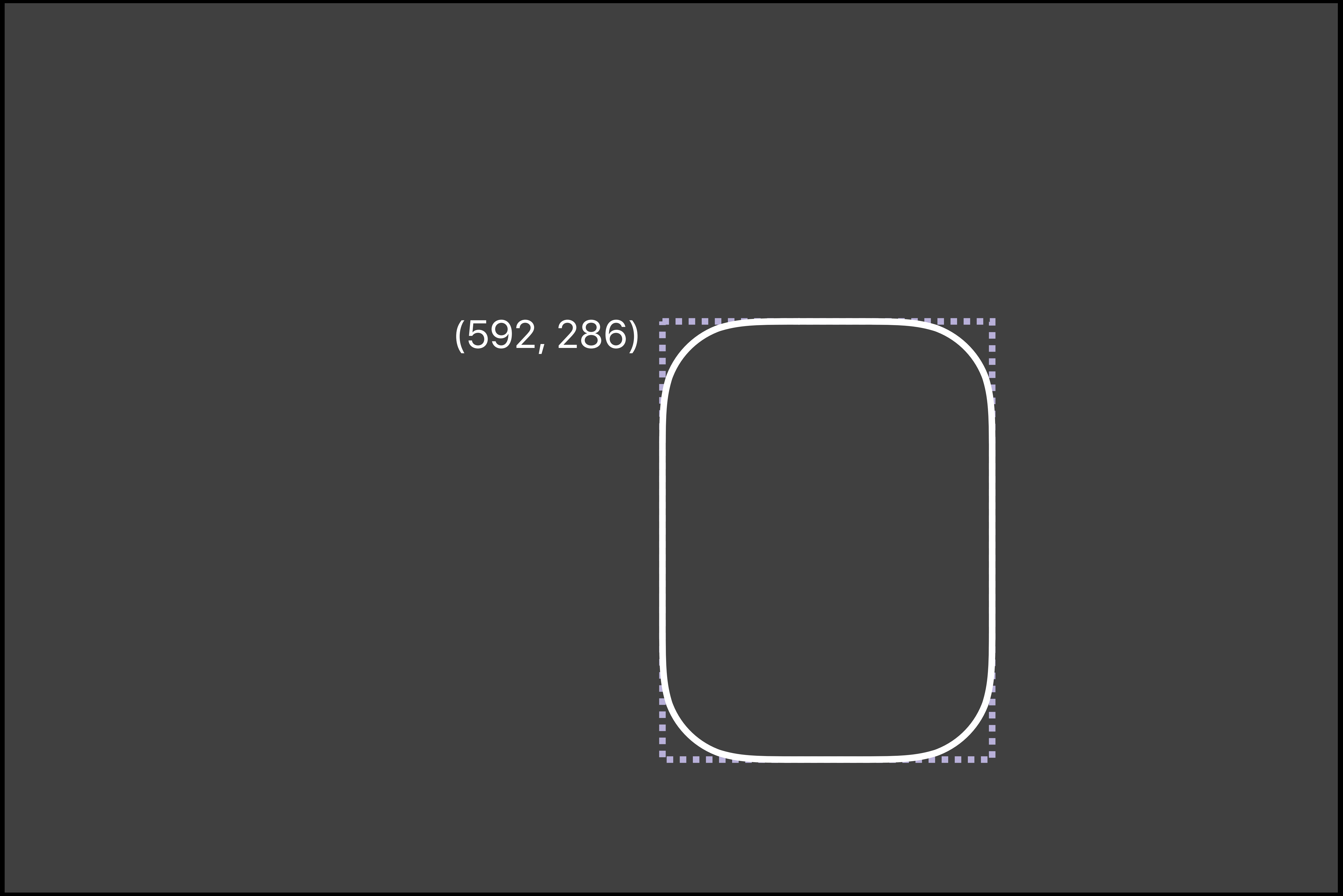


## UIDragPreviewParameters

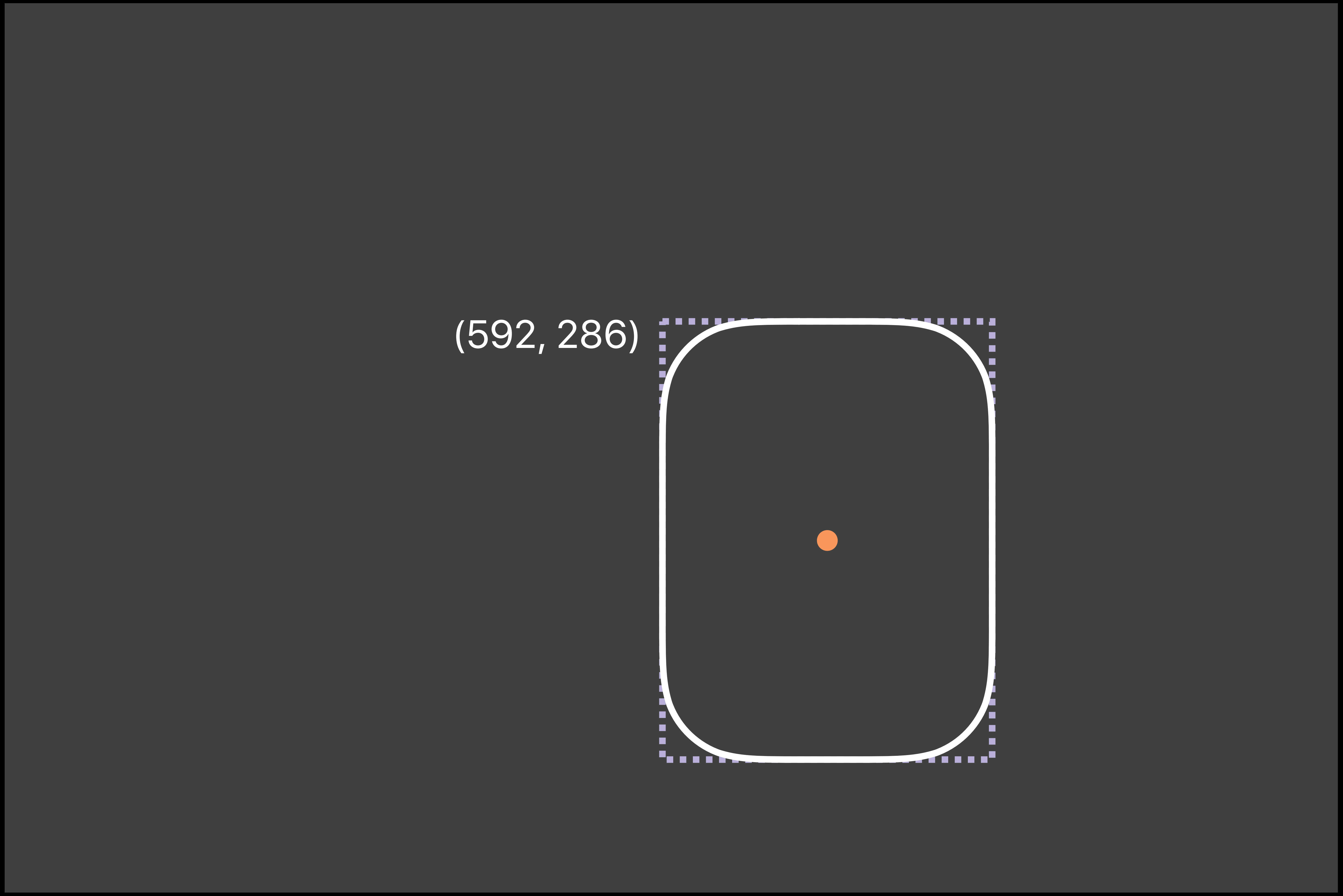
## UIDragPreviewParameters



# UIDragPreviewParameters



## UIDragPreviewParameters

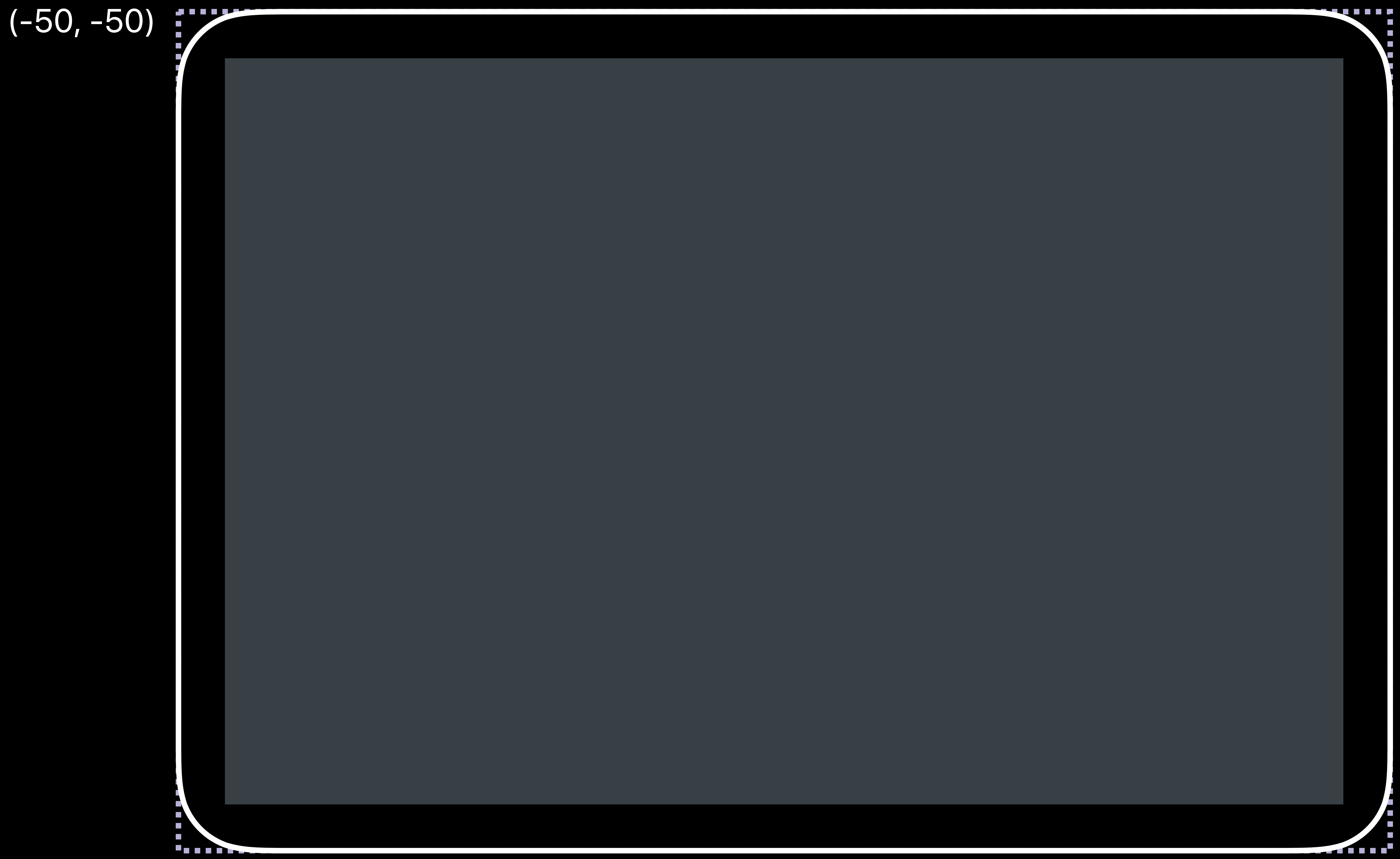


## UIDragPreviewParameters

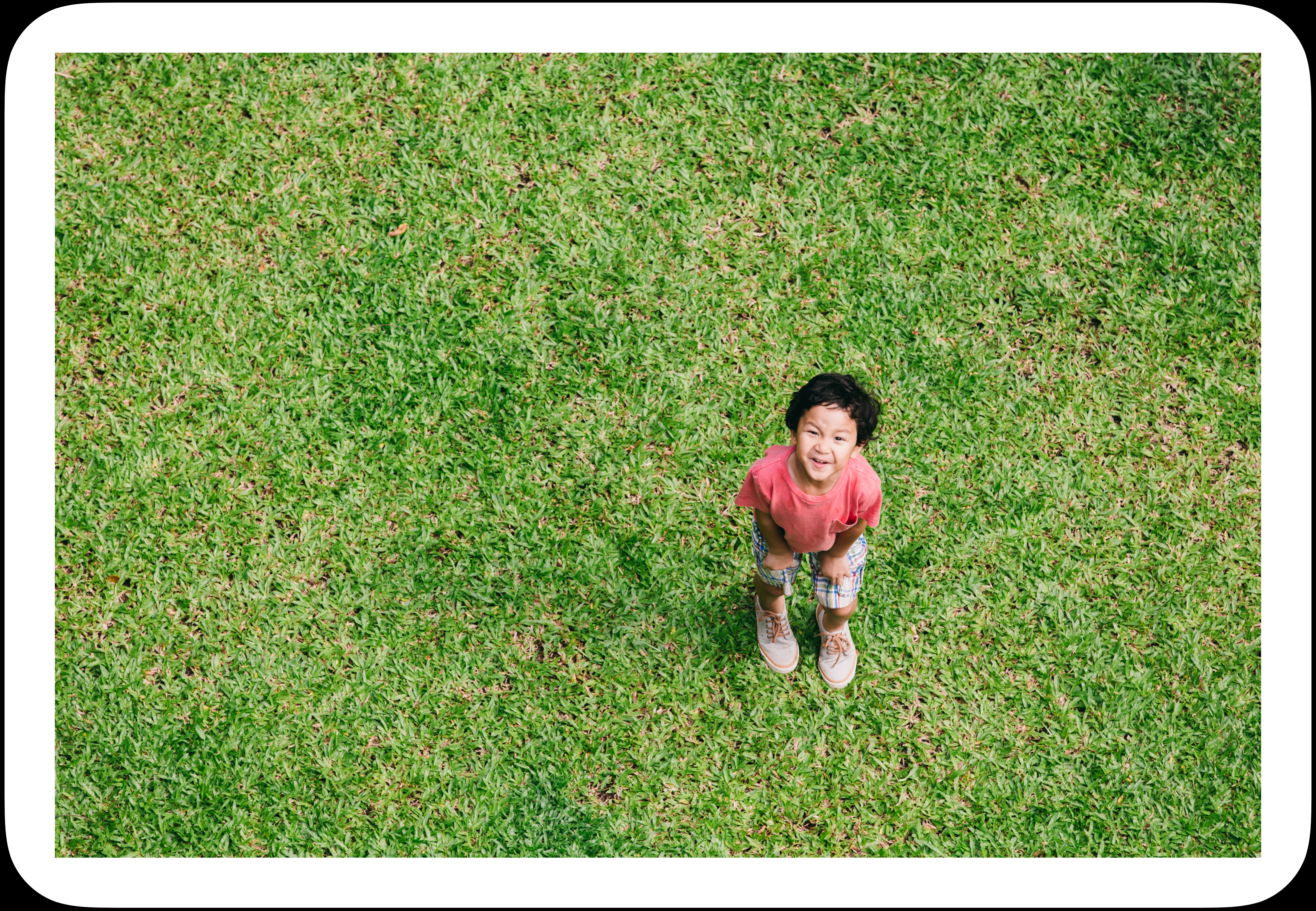
## UIDragPreviewParameters

## UIDragPreviewParameters

# UIDragPreviewParameters



## UIDragPreviewParameters



## UIDragPreviewParameters

Can color the background behind the view

Knows which portion of the preview is visible

- Special style for text

# UITargetedDragPreview

View

Parameters

Target

## UIDragPreviewTarget

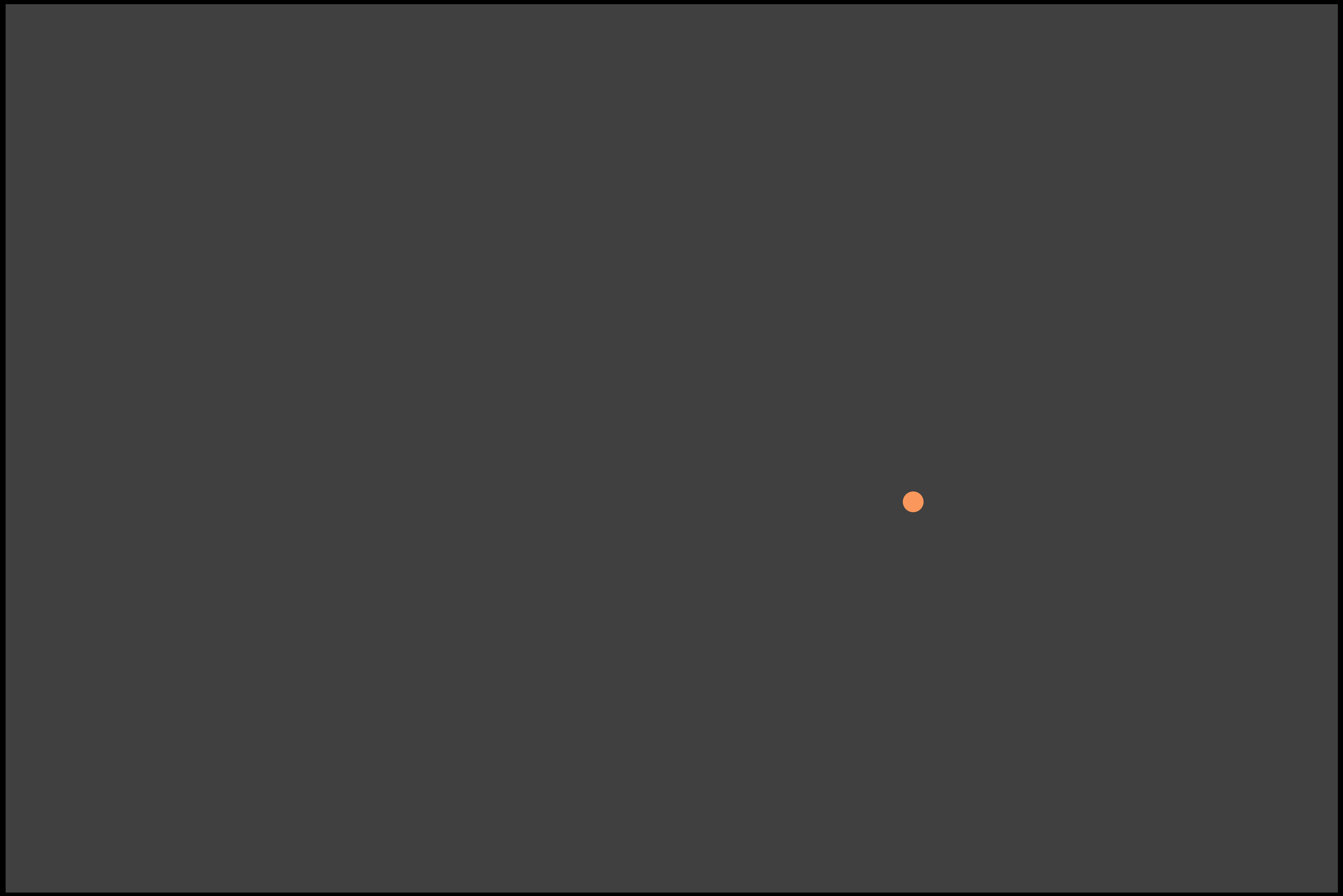
Knows where the preview is going to or coming from

Positions the preview inside its container

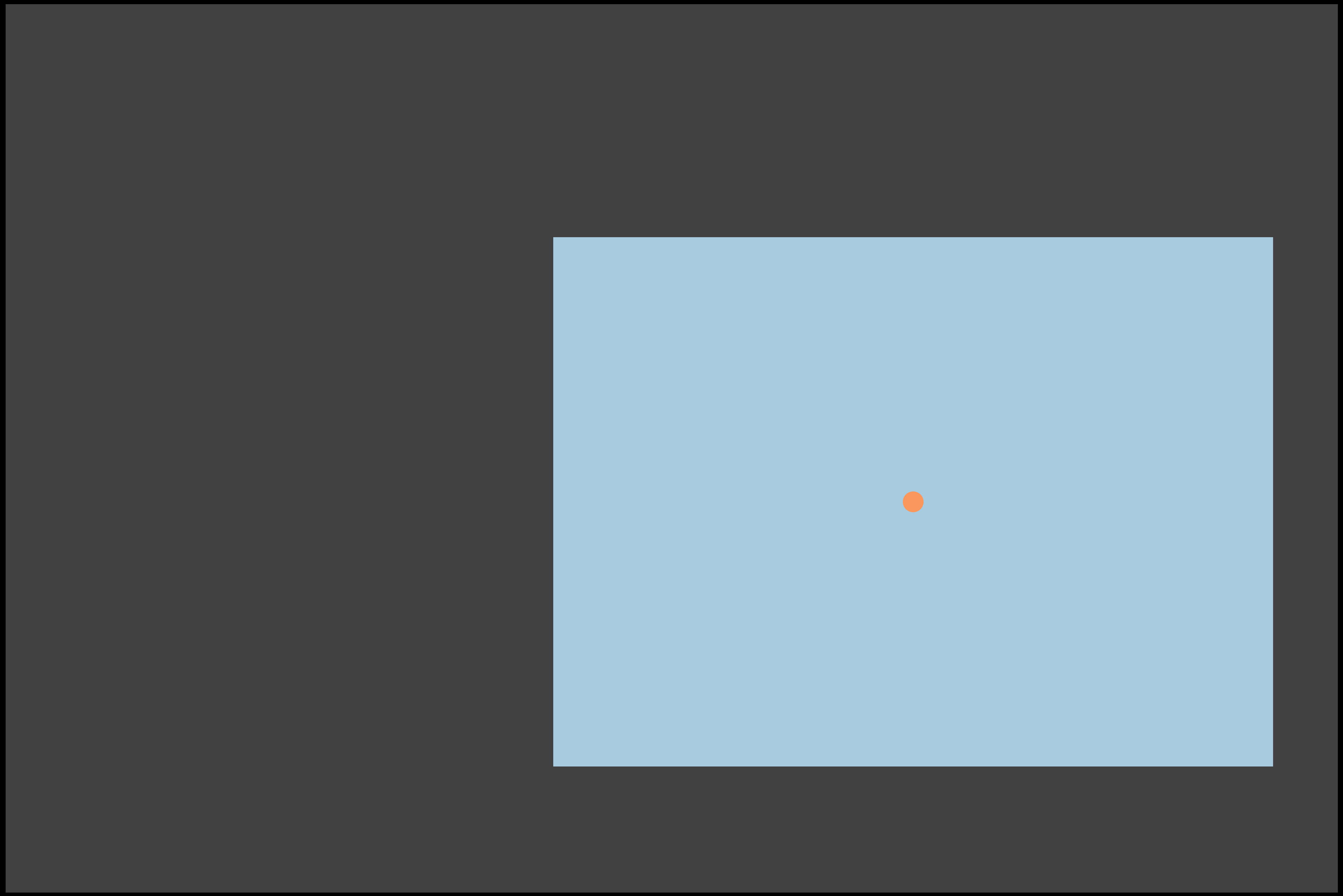
May apply a transform during set down

## Positioning with a Target

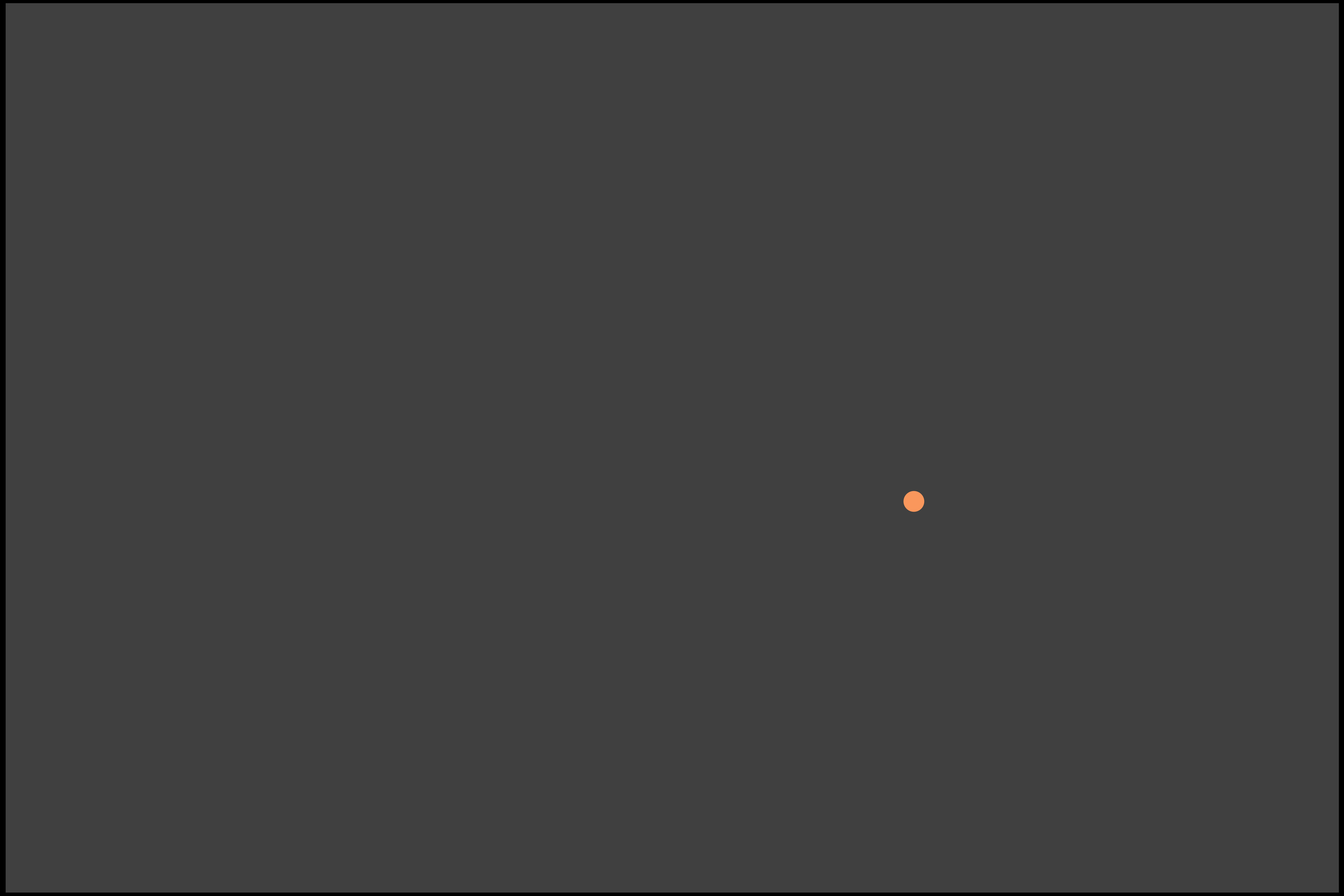
## Positioning with a Target



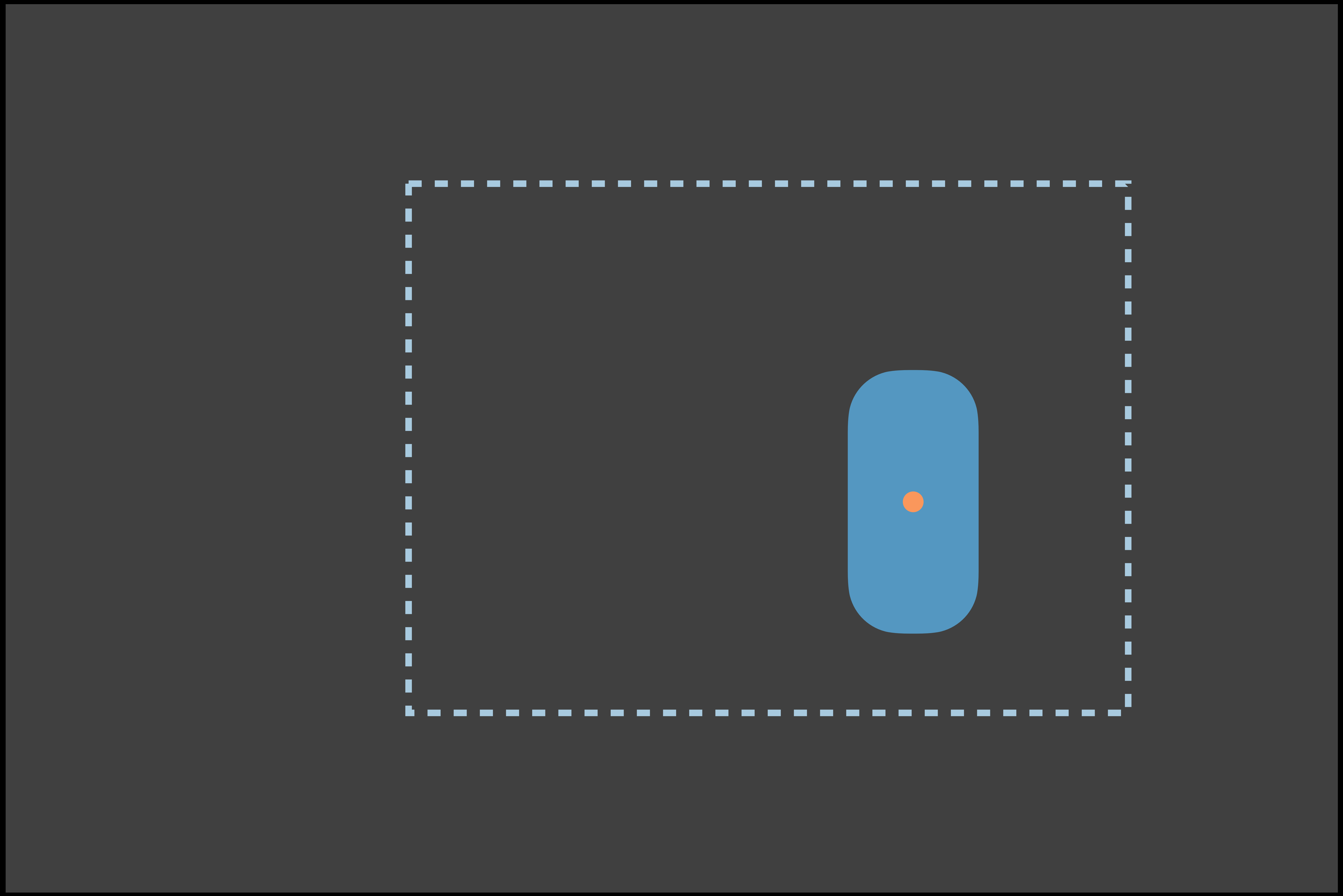
## Positioning with a Target



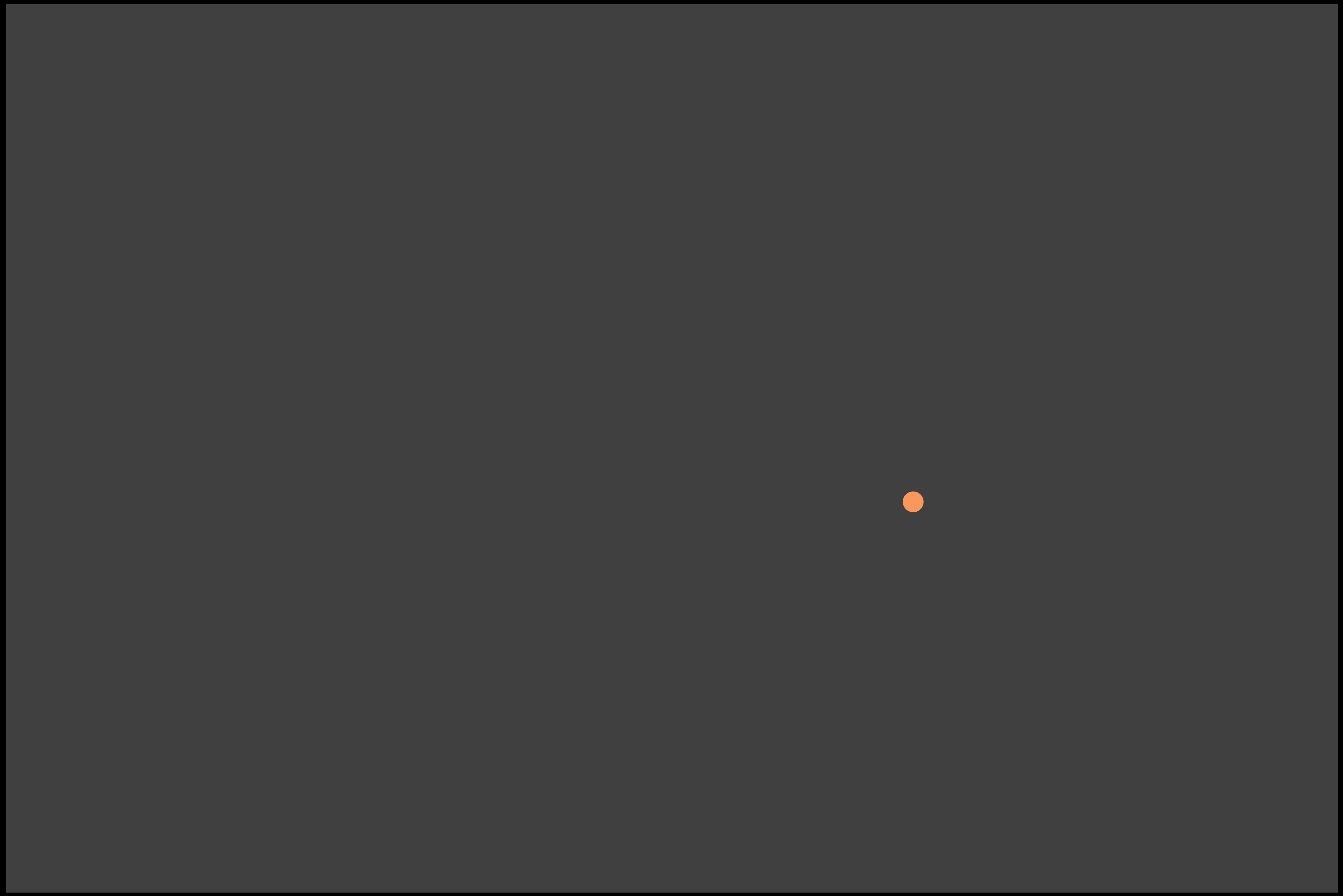
## Positioning with a Target



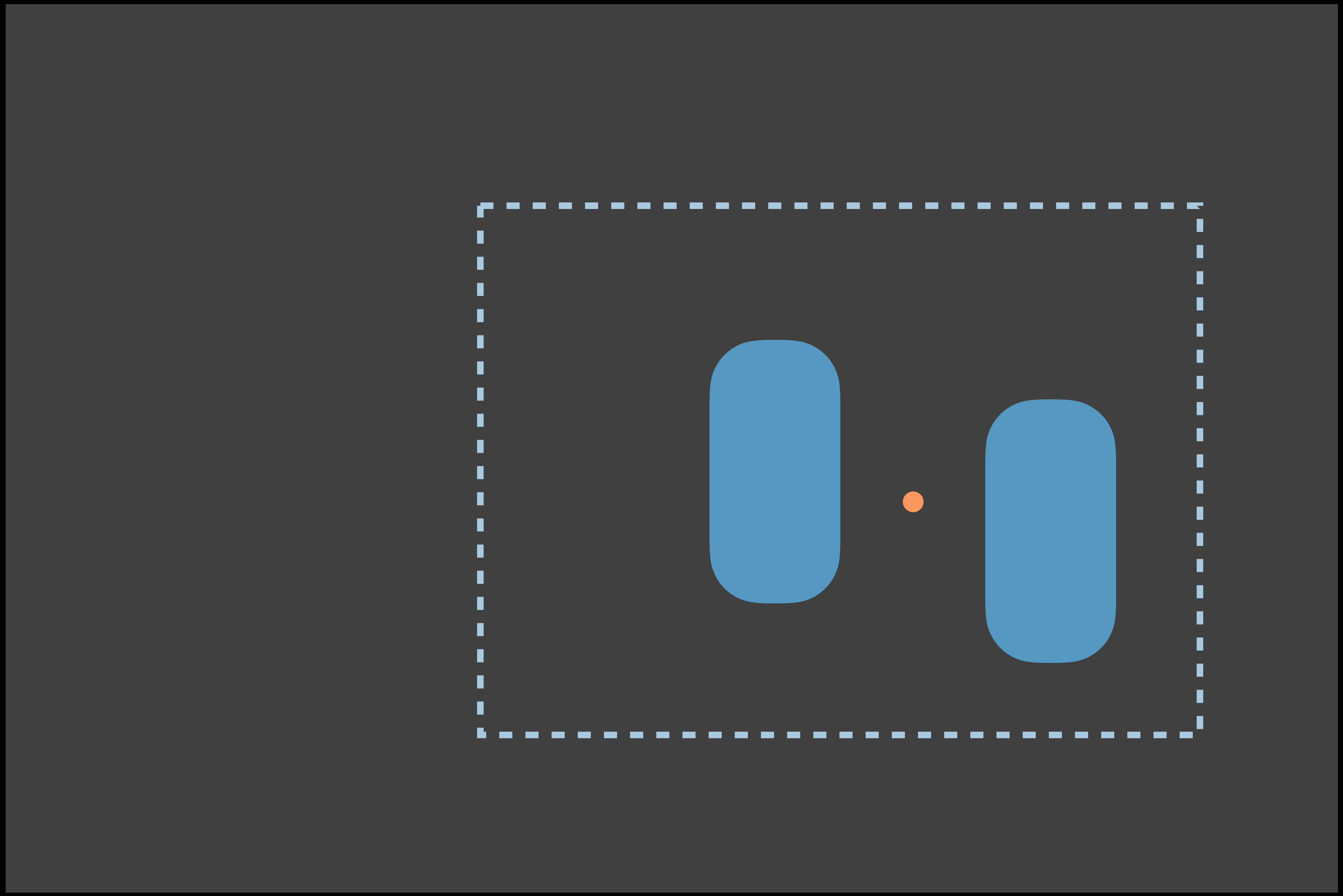
## Positioning with a Target



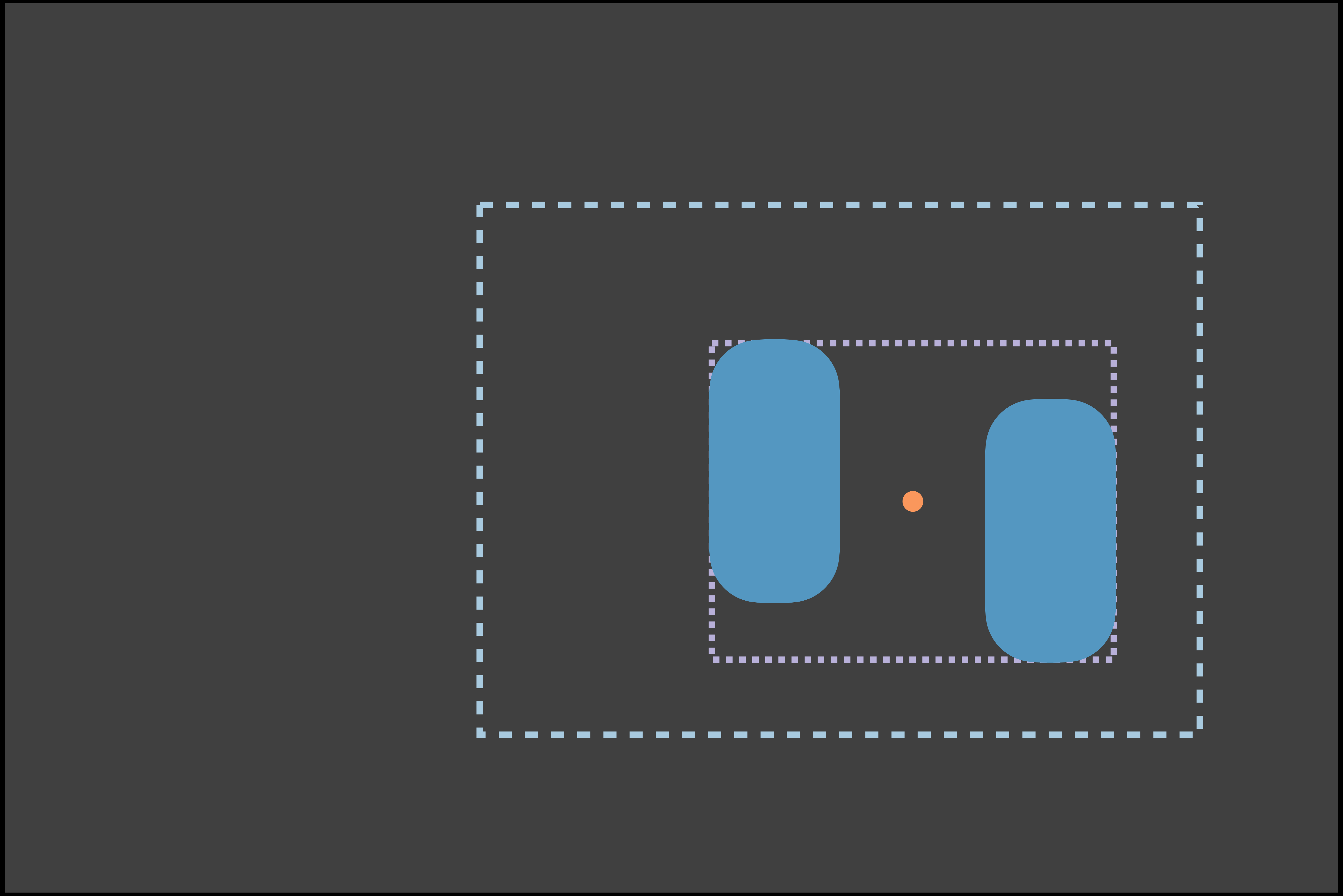
# Positioning with a Target

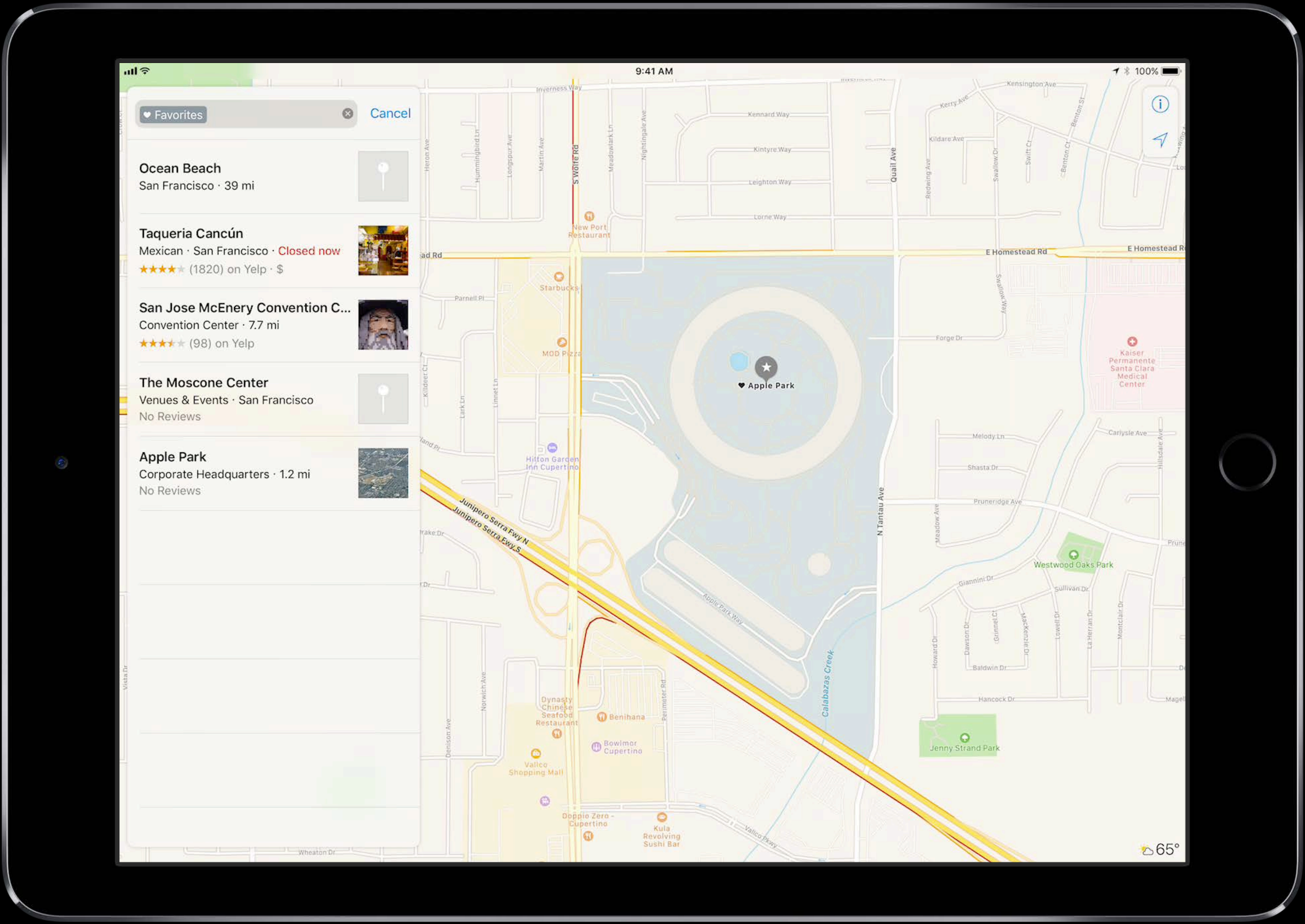


## Positioning with a Target



# Positioning with a Target





9:41 AM

100%

Favorites

Cancel

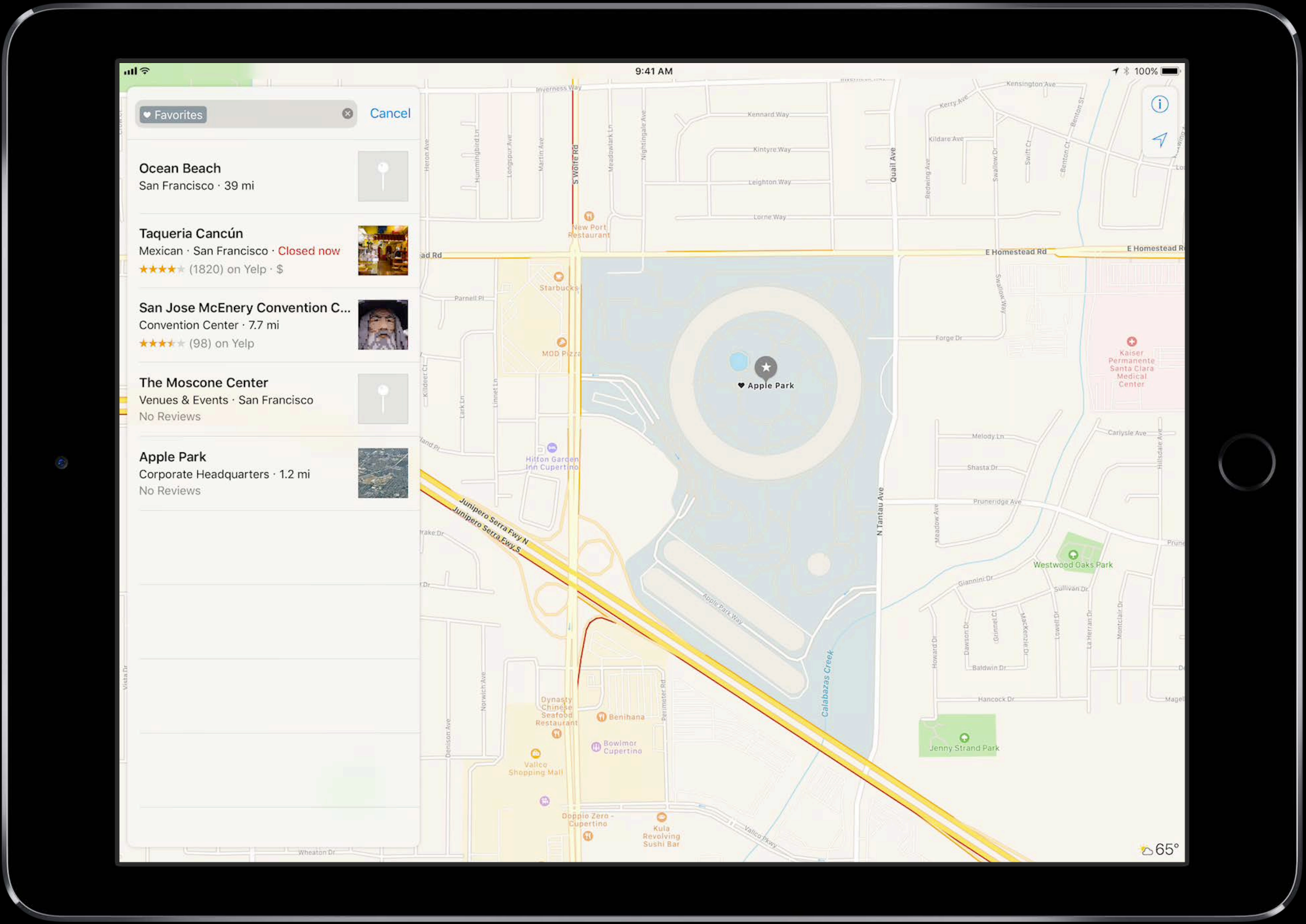
Ocean Beach
San Francisco · 39 mi

Taqueria Cancún
Mexican · San Francisco · Closed now
★★★★☆ (1820) on Yelp · $

San Jose McEnery Convention C...
Convention Center · 7.7 mi
★★★½☆ (98) on Yelp

The Moscone Center
Venues & Events · San Francisco
No Reviews

Apple Park
Corporate Headquarters · 1.2 mi
No Reviews



Ocean Beach
San Francisco · 39 mi

Taqueria Cancún
Mexican · San Francisco · Closed now
★★★★☆ (1820) on Yelp · $

San Jose McEnery Convention C...
Convention Center · 7.7 mi
★★★½☆ (98) on Yelp

The Moscone Center
Venues & Events · San Francisco
No Reviews

Apple Park
Corporate Headquarters · 1.2 mi
No Reviews

## UIDragPreview

Used to update previews while in flight

Does not have a target

```
// UIDragInteractionDelegate
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     sessionDidMove session: UIDragSession) {
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     sessionDidMove session: UIDragSession) {
    guard !listView.bounds.contains(session.location(in: listView)) else { return }
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     sessionDidMove session: UIDragSession) {
    guard !listView.bounds.contains(session.location(in: listView)) else { return }

    for item in session.items where item.localObject is Message {
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     sessionDidMove session: UIDragSession) {
    guard !listView.bounds.contains(session.location(in: listView)) else { return }

    for item in session.items where item.localObject is Message {
        guard !updatedItems.contains(item) else { continue }
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     sessionDidMove session: UIDragSession) {
    guard !listView.bounds.contains(session.location(in: listView)) else { return }

    for item in session.items where item.localObject is Message {
        guard !updatedItems.contains(item) else { continue }

        item.previewProvider = {
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     sessionDidMove session: UIDragSession) {
    guard !listView.bounds.contains(session.location(in: listView)) else { return }

    for item in session.items where item.localObject is Message {
        guard !updatedItems.contains(item) else { continue }

        item.previewProvider = {
            let imageView = UIImageView(image: UIImage(named: "envelope"))
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     sessionDidMove session: UIDragSession) {
    guard !listView.bounds.contains(session.location(in: listView)) else { return }

    for item in session.items where item.localObject is Message {
        guard !updatedItems.contains(item) else { continue }

        item.previewProvider = {
            let imageView = UIImageView(image: UIImage(named: "envelope"))

            return UIDragPreview(view: imageView)
        }
```

```swift
// UIDragInteractionDelegate
func dragInteraction(_ interaction: UIDragInteraction,
                     sessionDidMove session: UIDragSession) {
    guard !listView.bounds.contains(session.location(in: listView)) else { return }

    for item in session.items where item.localObject is Message {
        guard !updatedItems.contains(item) else { continue }

        item.previewProvider = {
            let imageView = UIImageView(image: UIImage(named: "envelope"))

            return UIDragPreview(view: imageView)
        }

        updatedItems.add(item)
    }
}
```

## *Demo*

Wenson Hsieh, WebKit

## Deep Dive into a Drop

Tom Adriaenssen, UIKit

## Deep Dive into a Drop

Drop sessions

Performing a drop

## Drop Sessions

## Drop Sessions

Your access to all things drop

- Drag location
- Items
- Configuration
- Drag session (when local)

## Drop Sessions

## Drop Sessions

One interaction + delegate = one active drop session

## Drop Sessions

One interaction + delegate = one active drop session

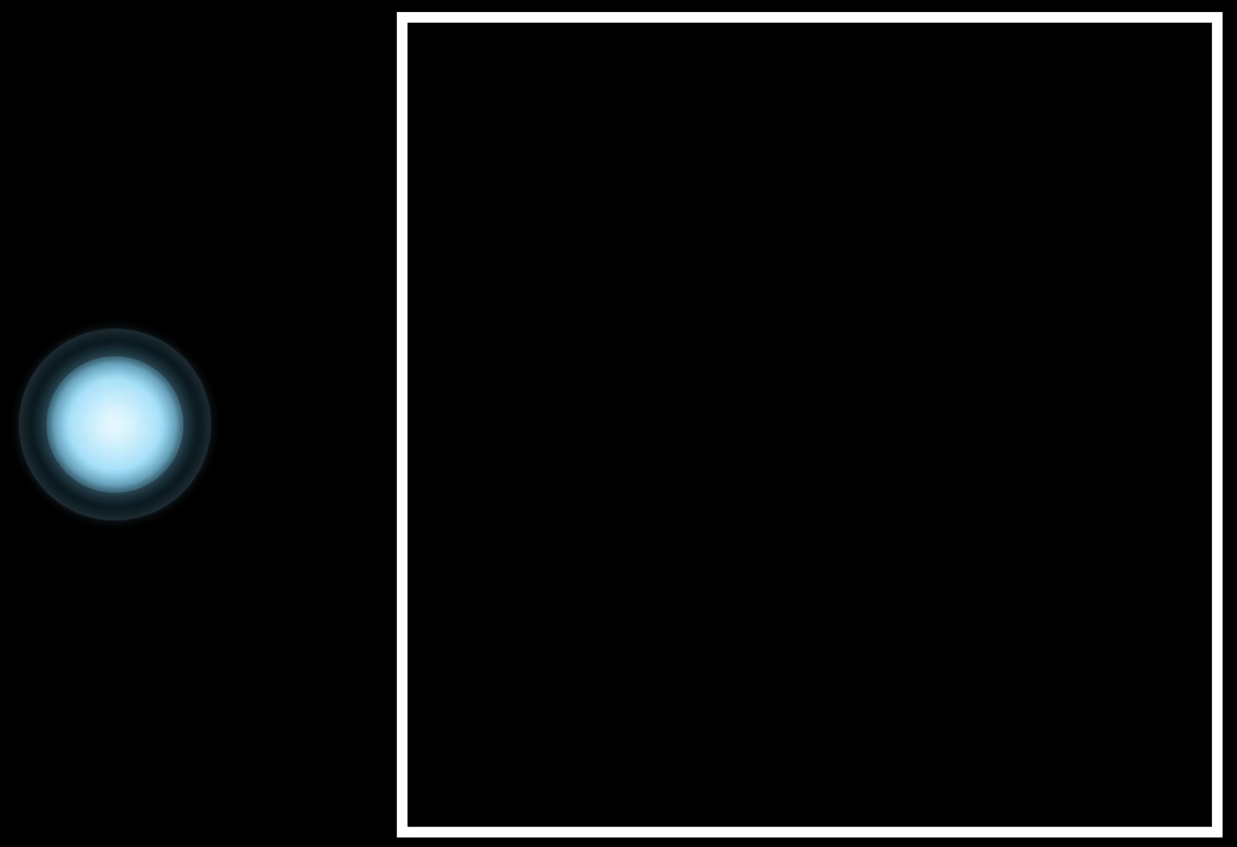
Supporting more sessions?

- Add more interactions
- Set `interaction.allowsSimultaneousDropSessions = true`

## Drop Sessions

Lifetime

## Drop Sessions
Lifetime

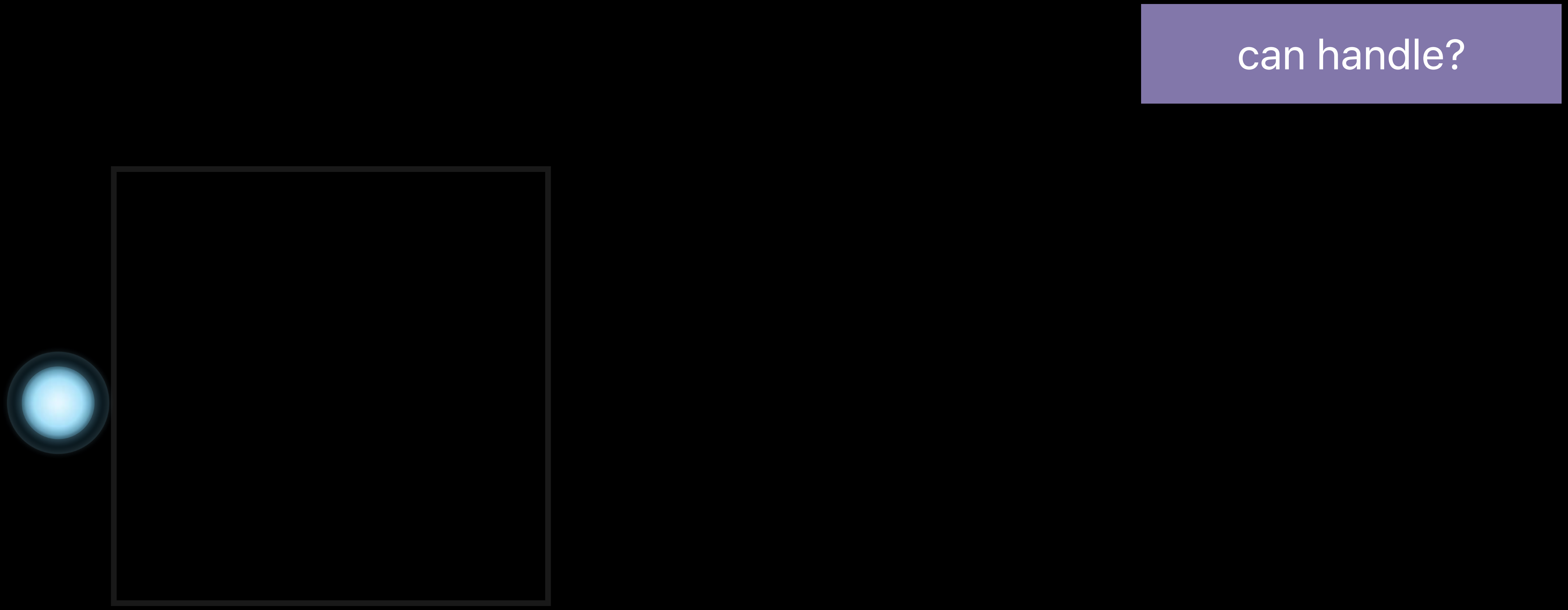


## Drop Sessions
### Lifetime

```
optional func dropInteraction(_ interaction: UIDropInteraction,
                              canHandle session: UIDropSession) -> Bool
```

## Drop Sessions

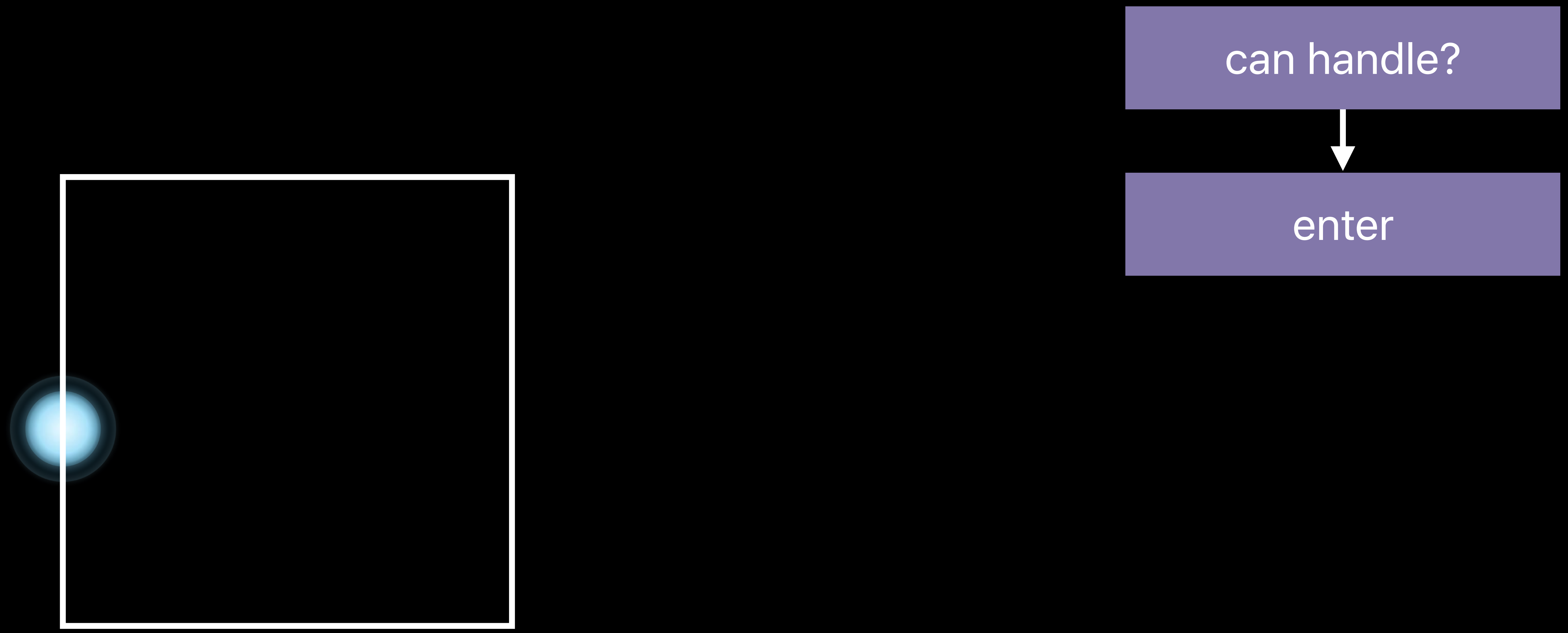
Lifetime



```
optional func dropInteraction(_ interaction: UIDropInteraction,
                              canHandle session: UIDropSession) -> Bool
```

# Drop Sessions

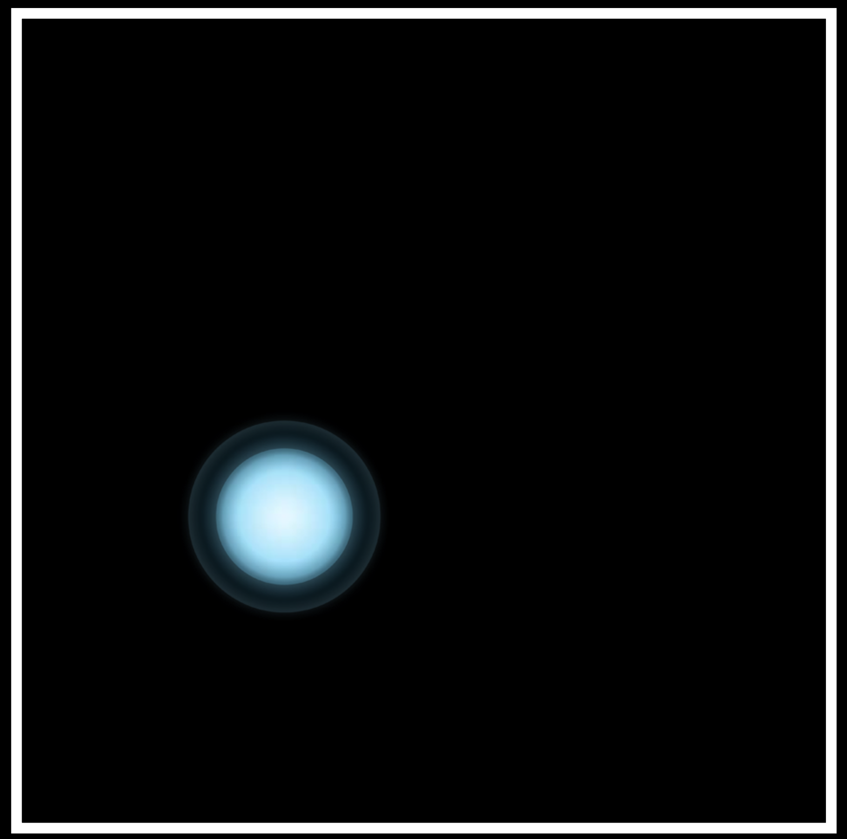
## Lifetime

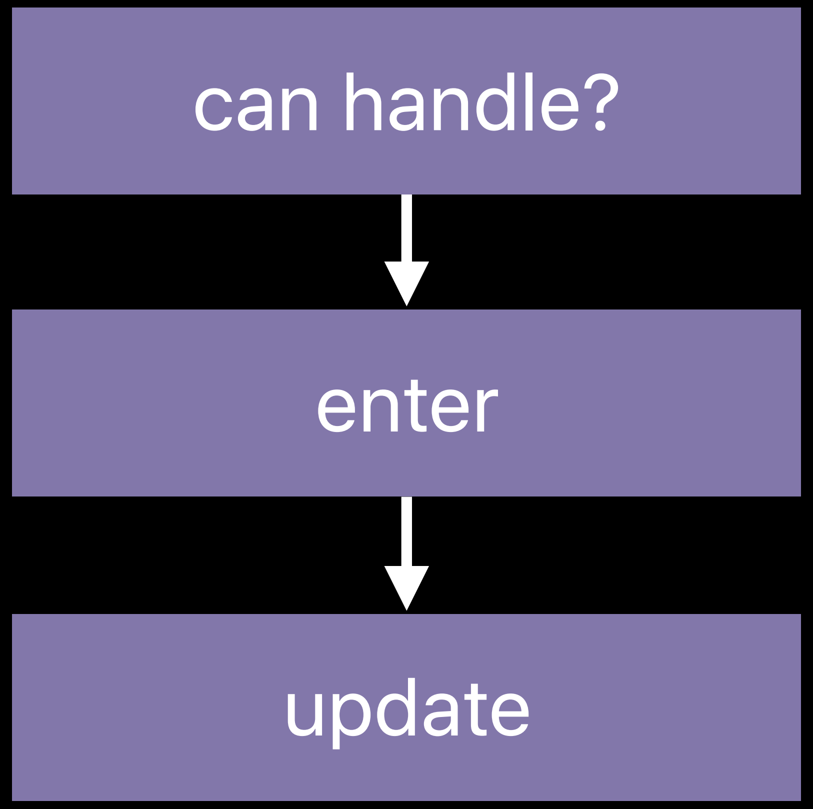


```
optional func dropInteraction(_ interaction: UIDropInteraction,
                    sessionDidEnter session: UIDropSession)
```

# Drop Sessions

Lifetime

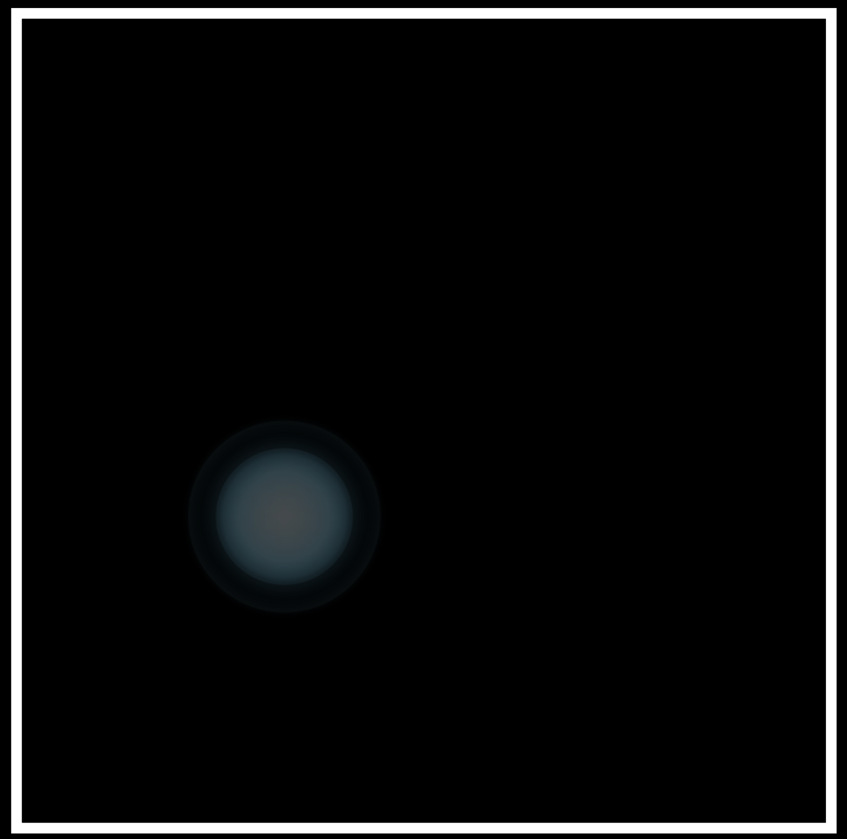


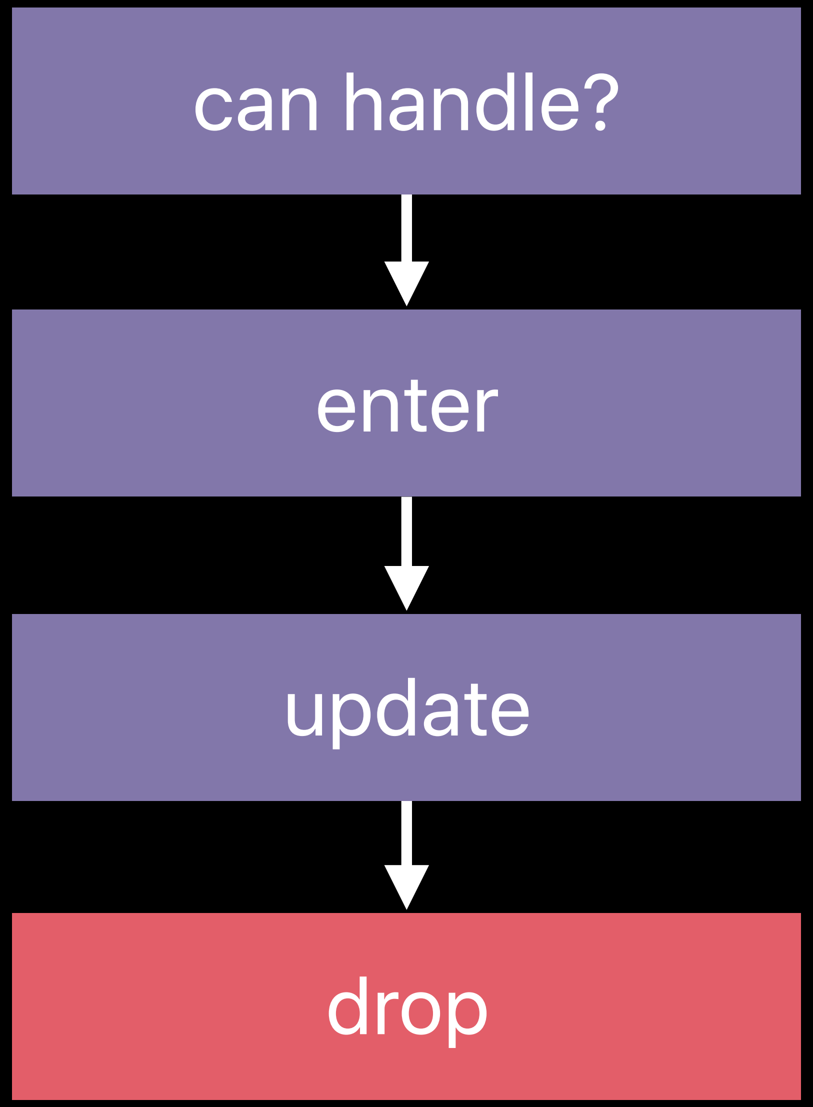


```
optional func dropInteraction(_ interaction: UIDropInteraction,
                   sessionDidUpdate session: UIDropSession) -> UIDropProposal
```

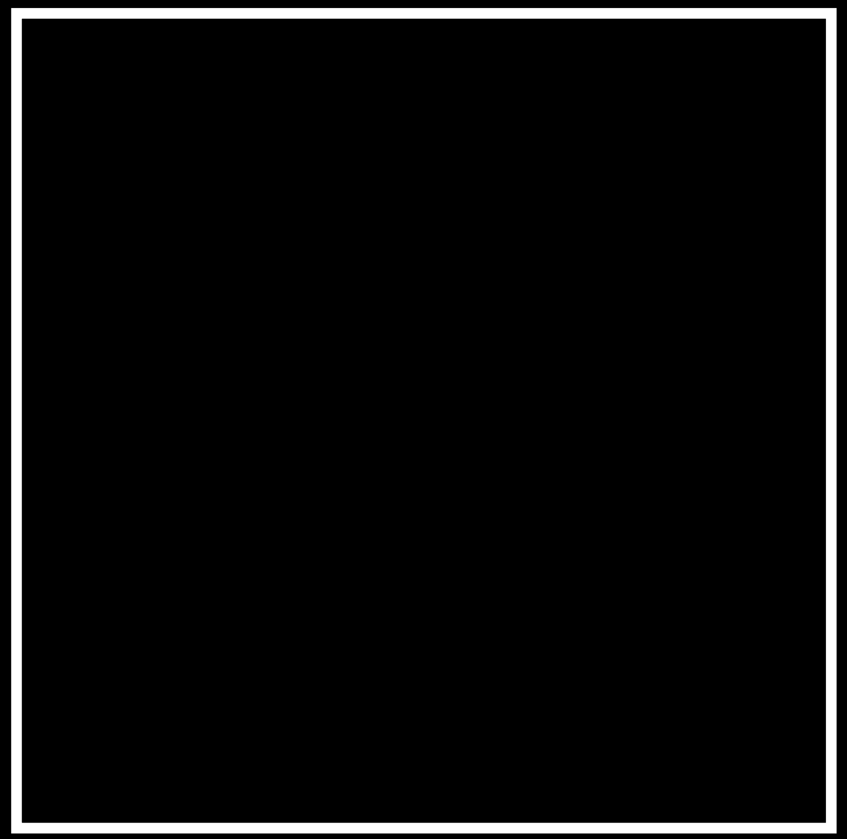
## Drop Sessions

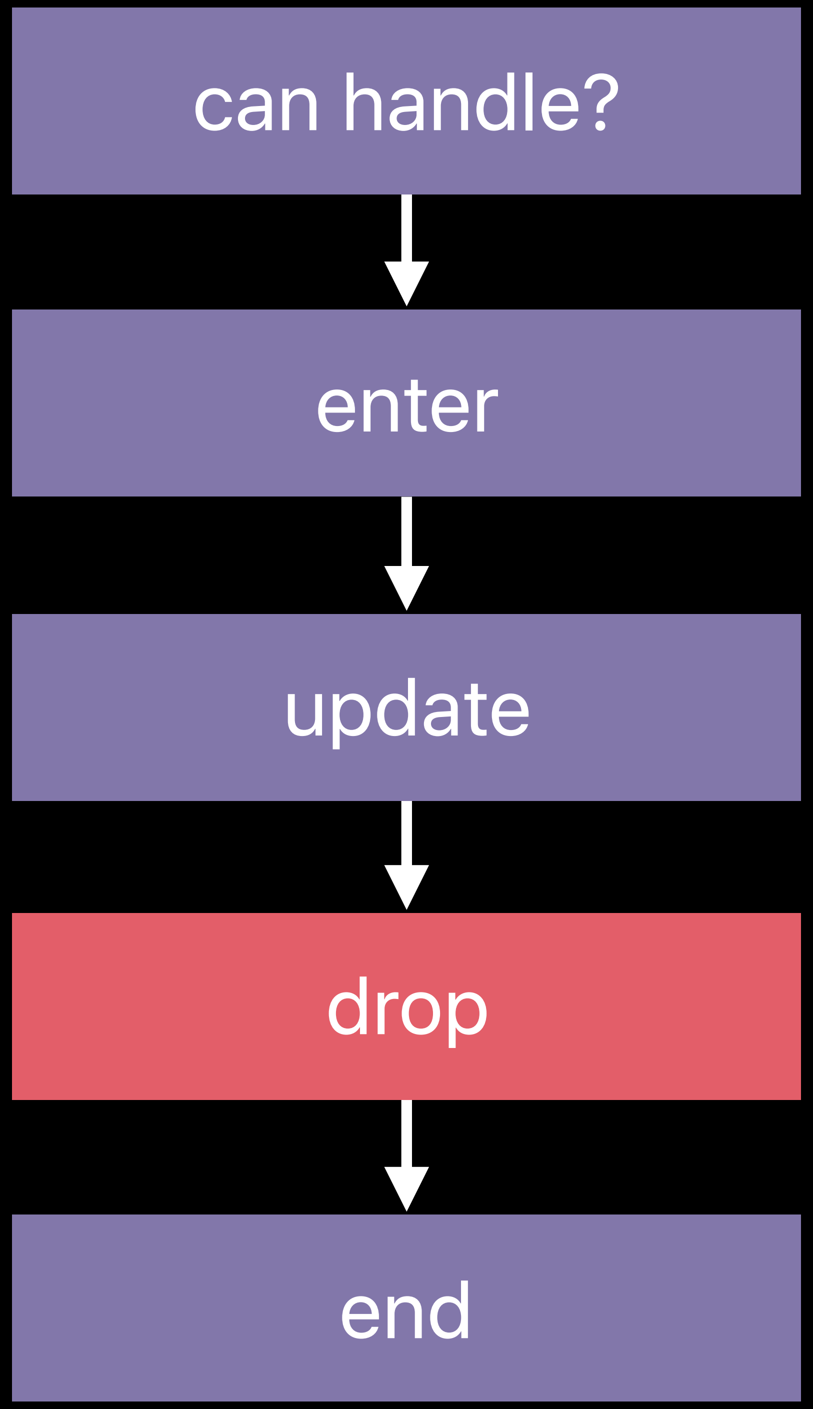
Lifetime





## Drop Sessions
### Lifetime

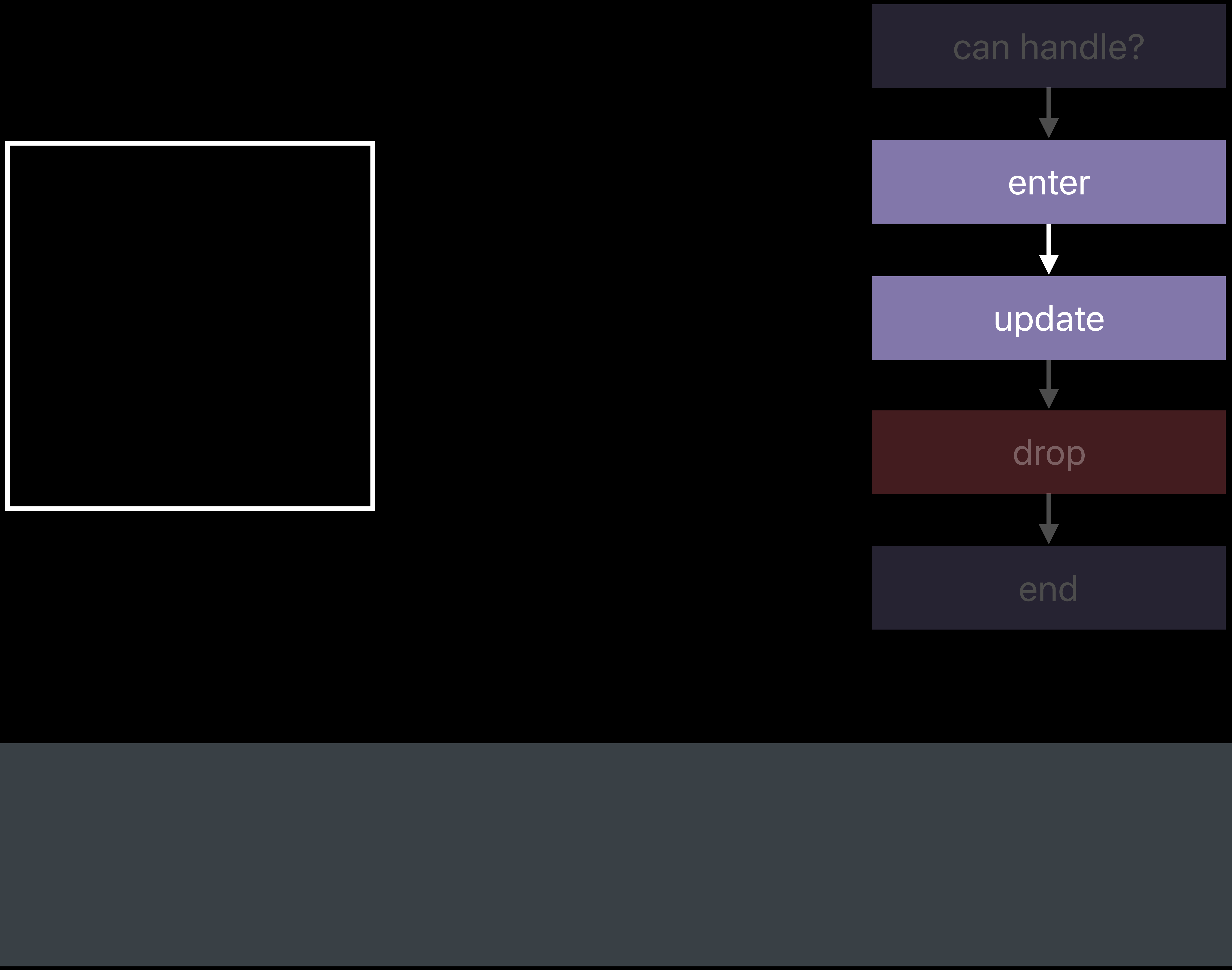




```
optional func dropInteraction(_ interaction: UIDropInteraction,
                              sessionDidEnd session: UIDropSession)
```

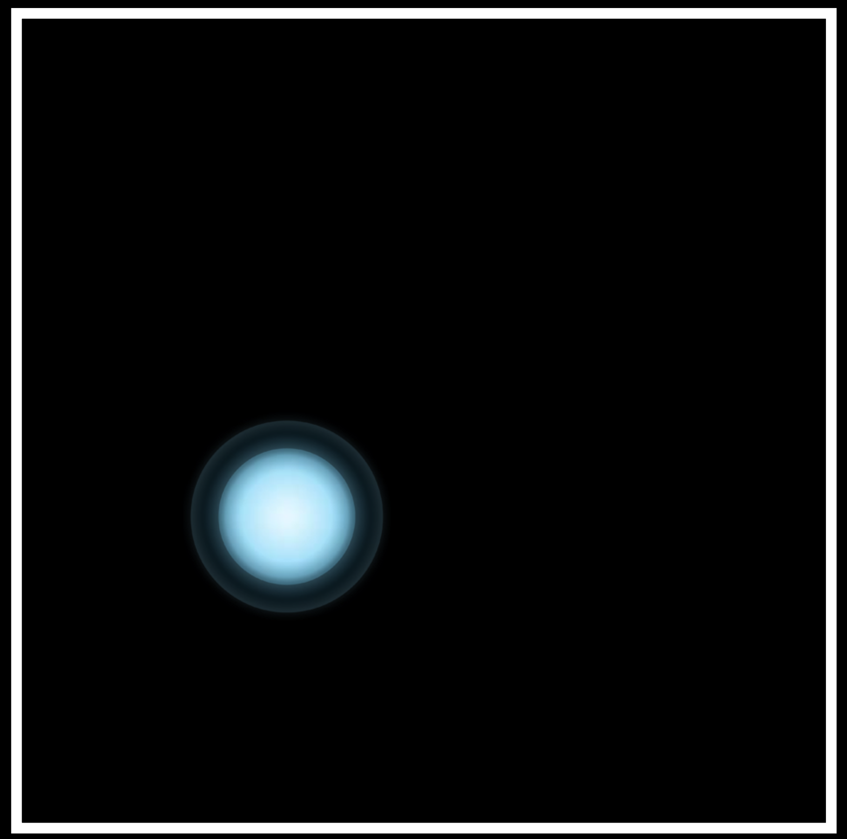
# Drop Sessions

## Lifetime

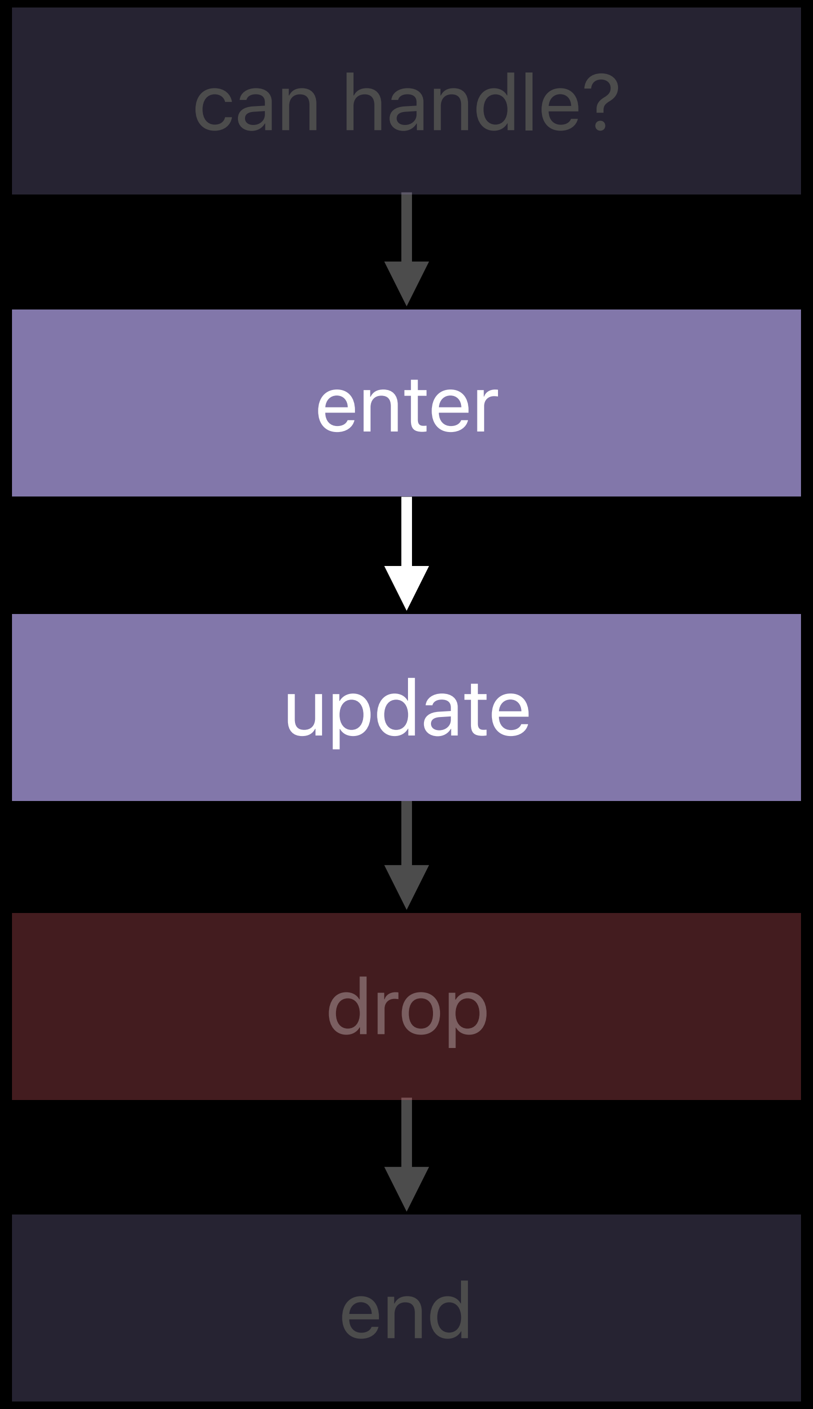


# Drop Sessions

## Lifetime

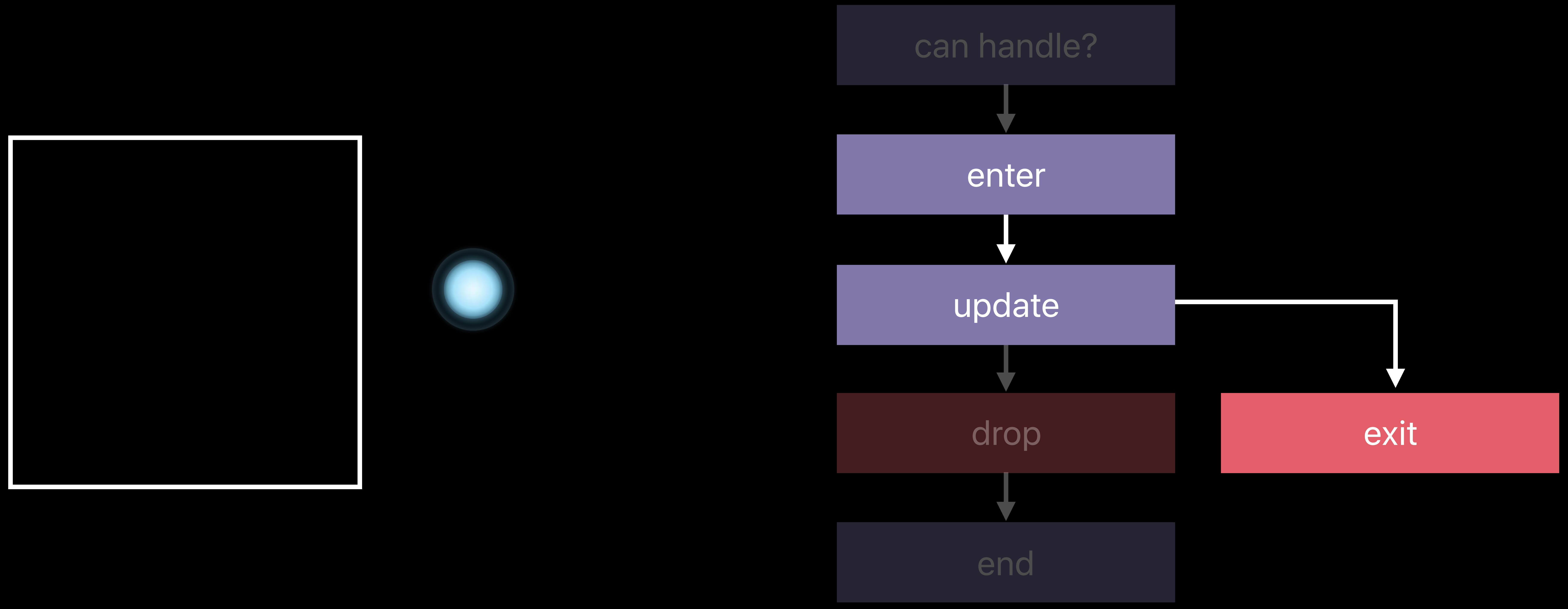


$$\begin{tikzcd}
\text{can handle?} \arrow[d] \\
\text{enter} \arrow[d] \\
\text{update} \arrow[d] \\
\text{drop} \arrow[d] \\
\text{end}
\end{tikzcd}$$



# Drop Sessions

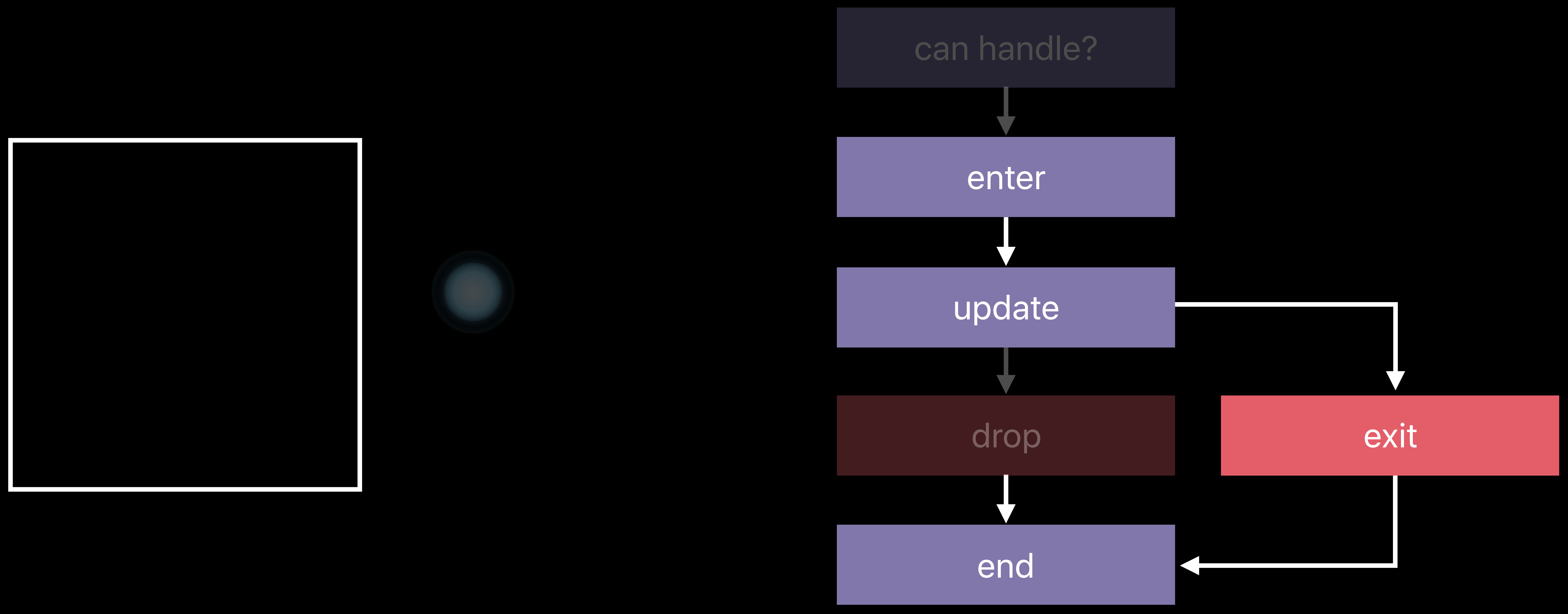
## Lifetime



```
optional func dropInteraction(_ interaction: UIDropInteraction,
                     sessionDidExit session: UIDropSession)
```

# Drop Sessions

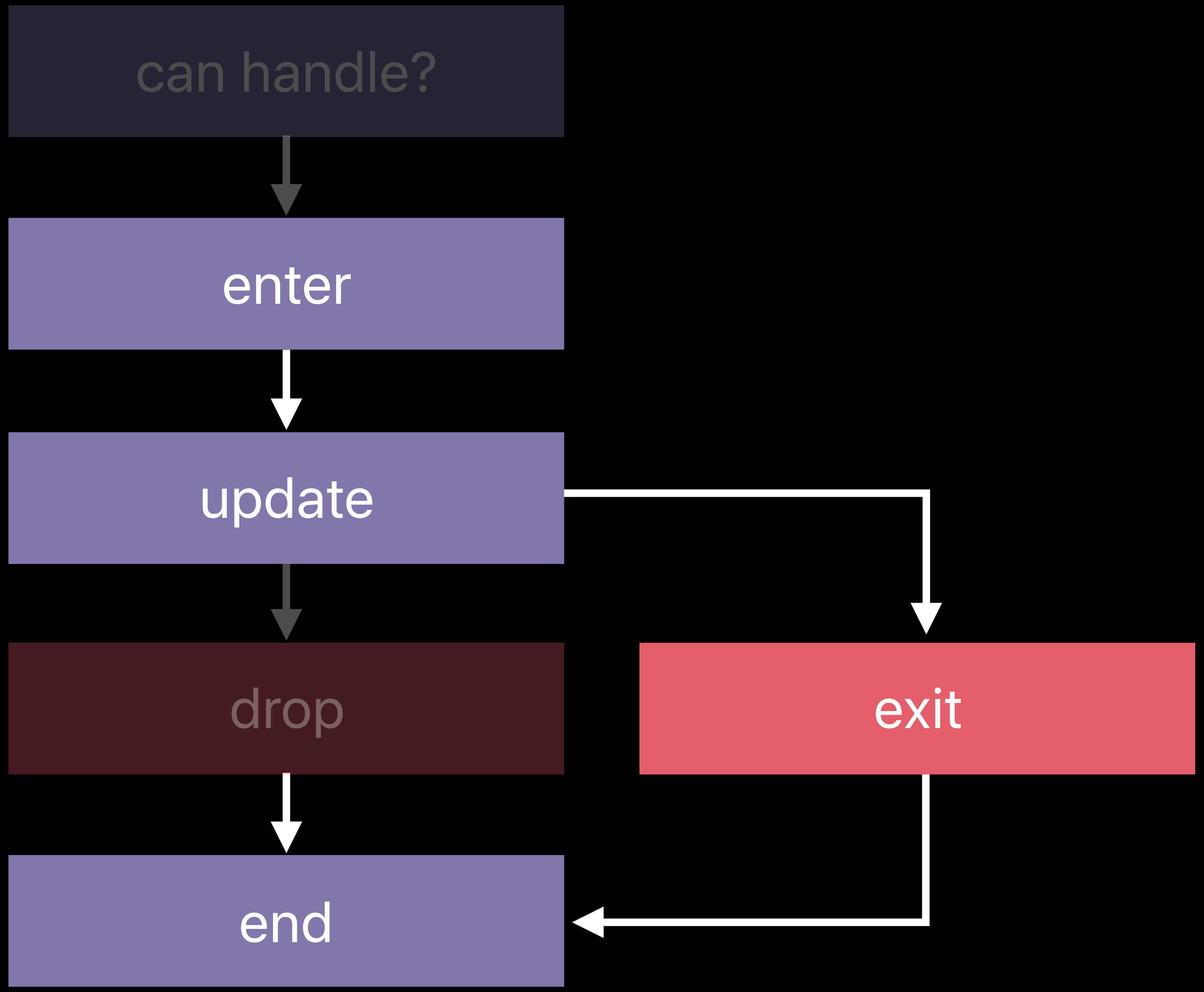
## Lifetime



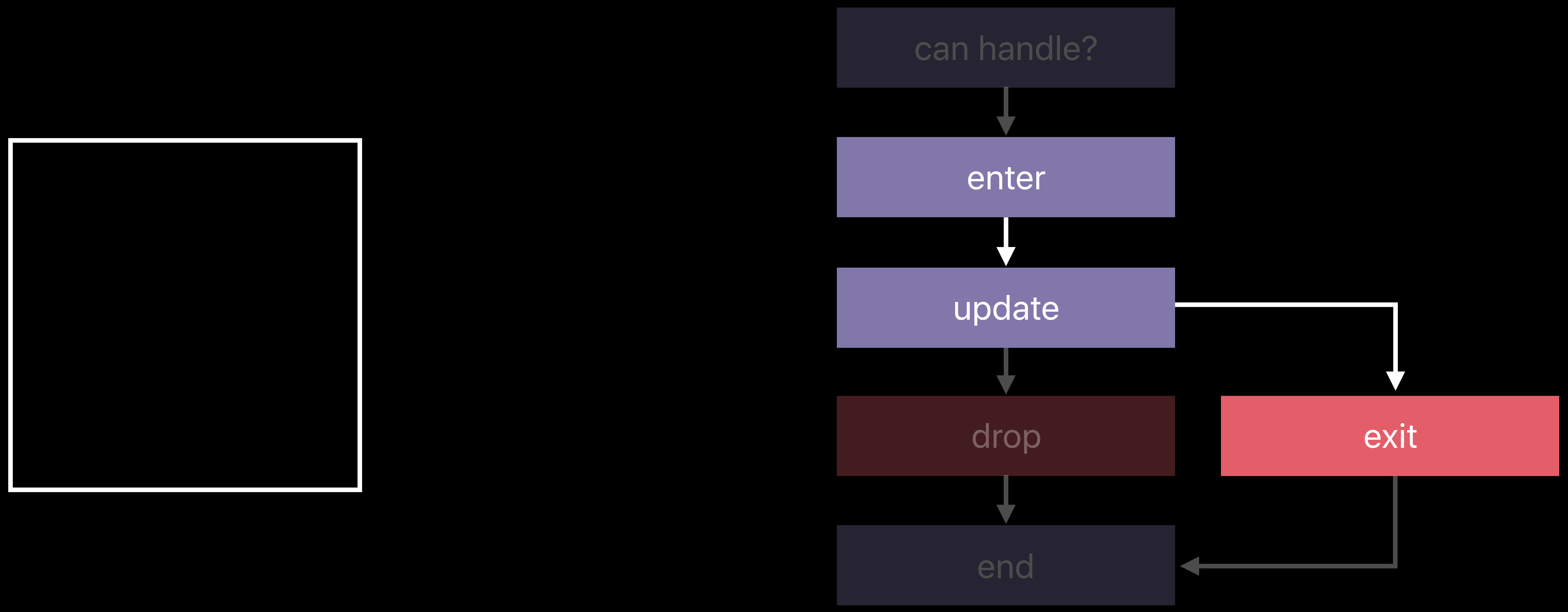
```
optional func dropInteraction(_ interaction: UIDropInteraction,
                          sessionDidEnd session: UIDropSession)
```

# Drop Sessions

Lifetime

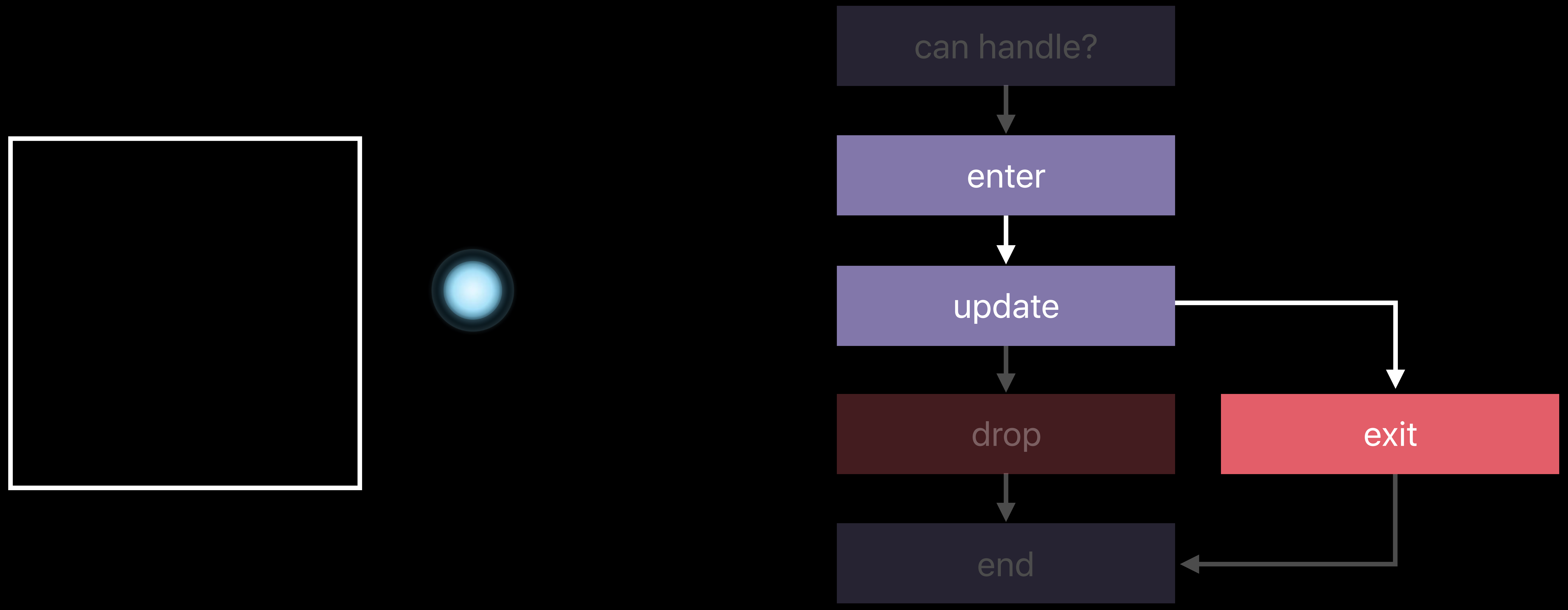


## Drop Sessions
Lifetime



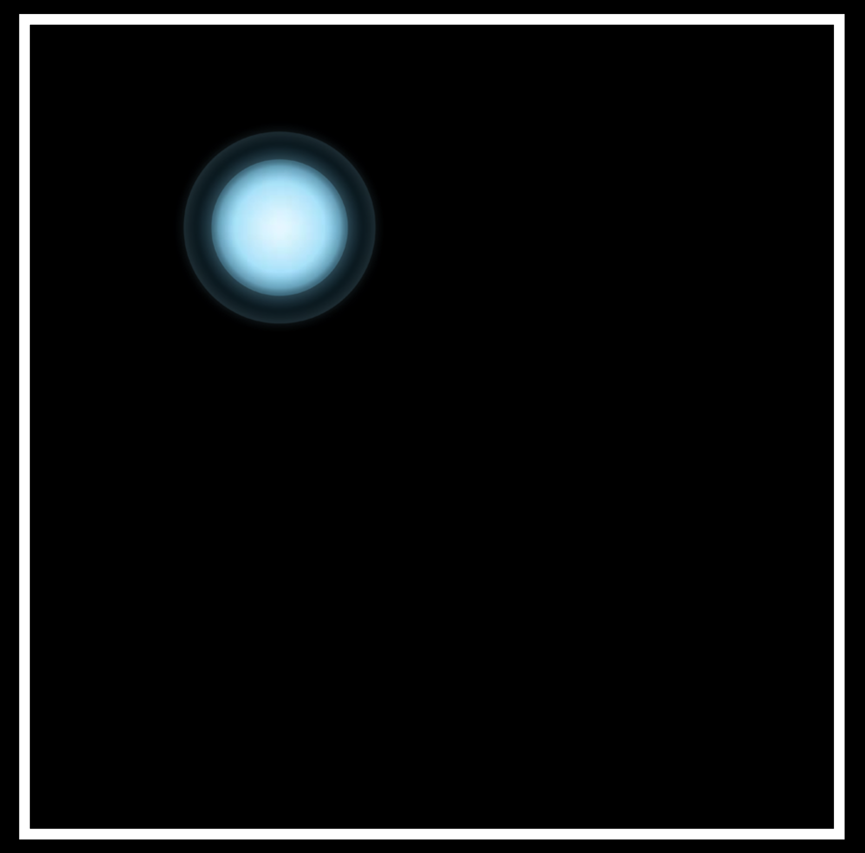
# Drop Sessions

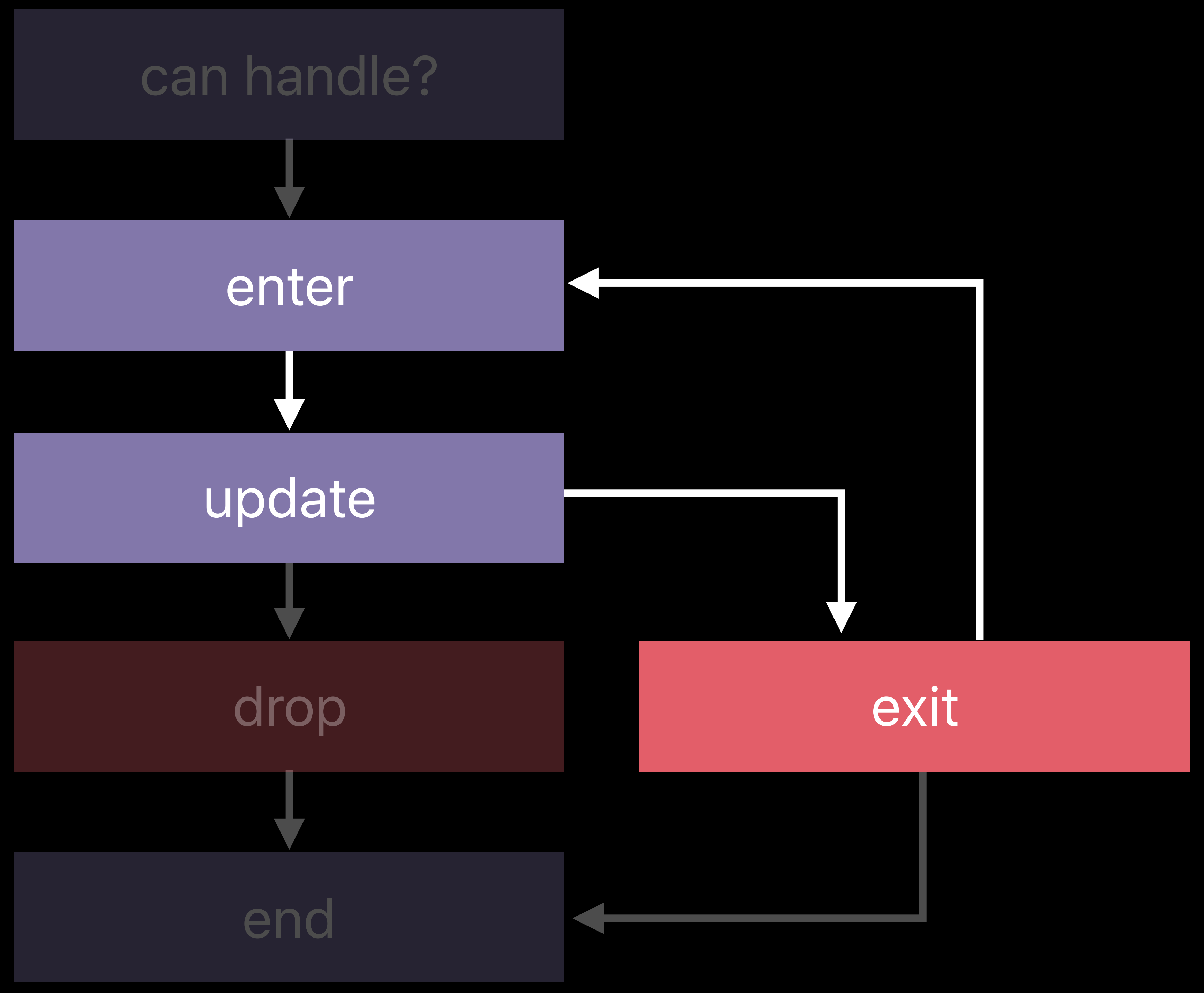
## Lifetime



# Drop Sessions

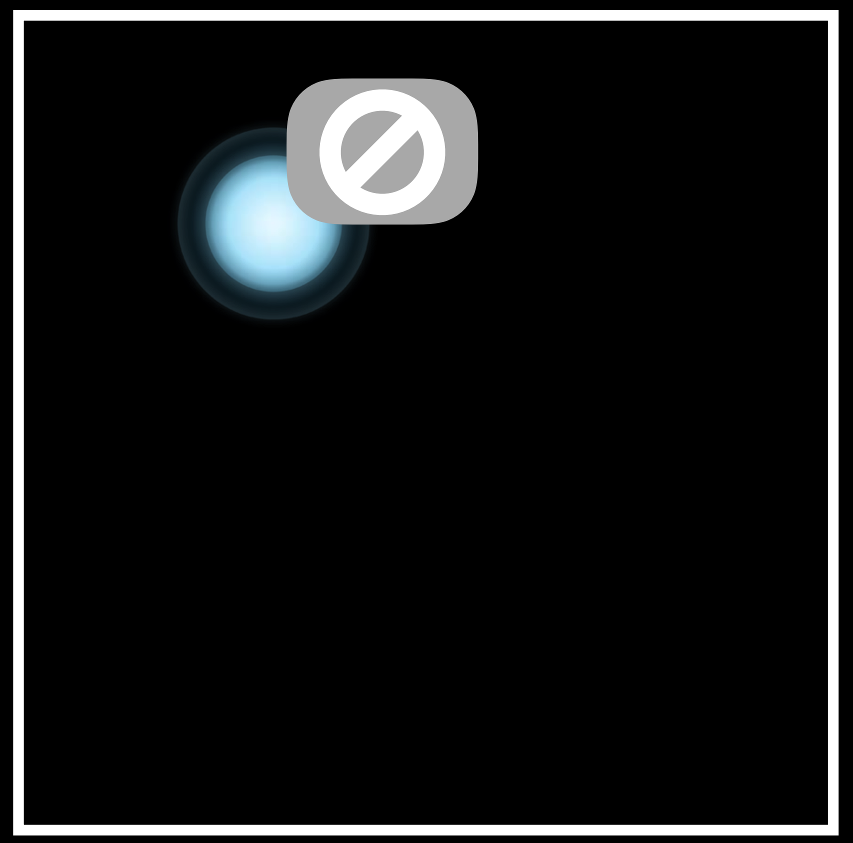
Lifetime

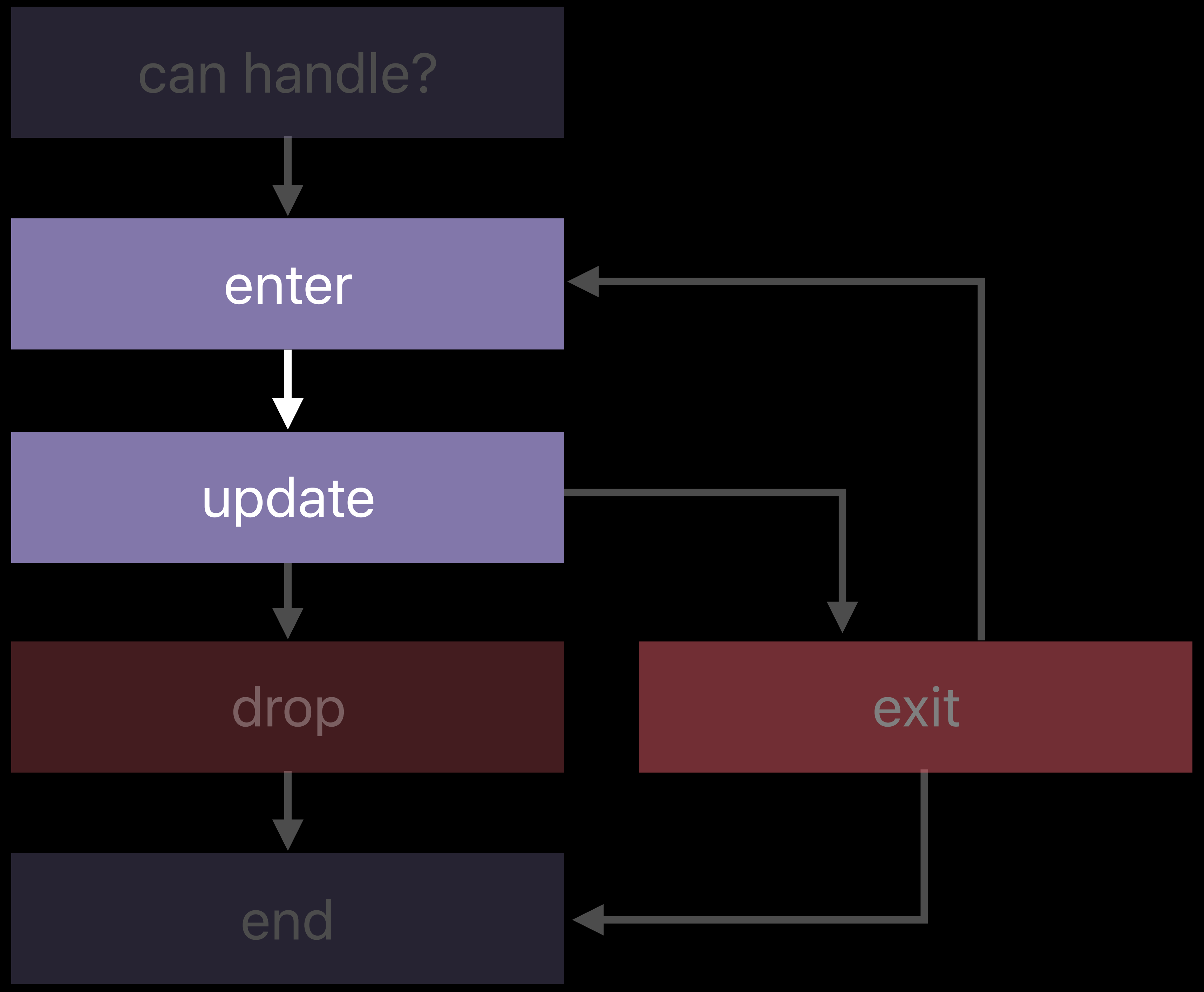




# Drop Sessions

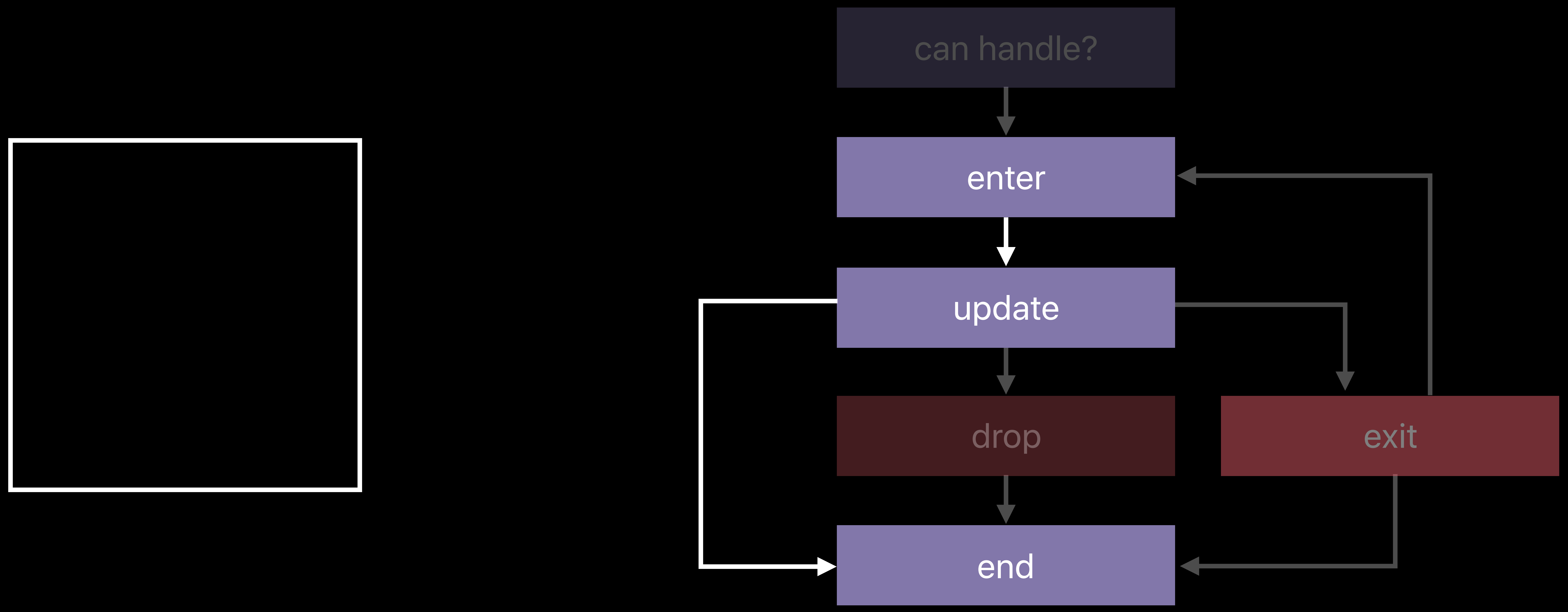
Lifetime





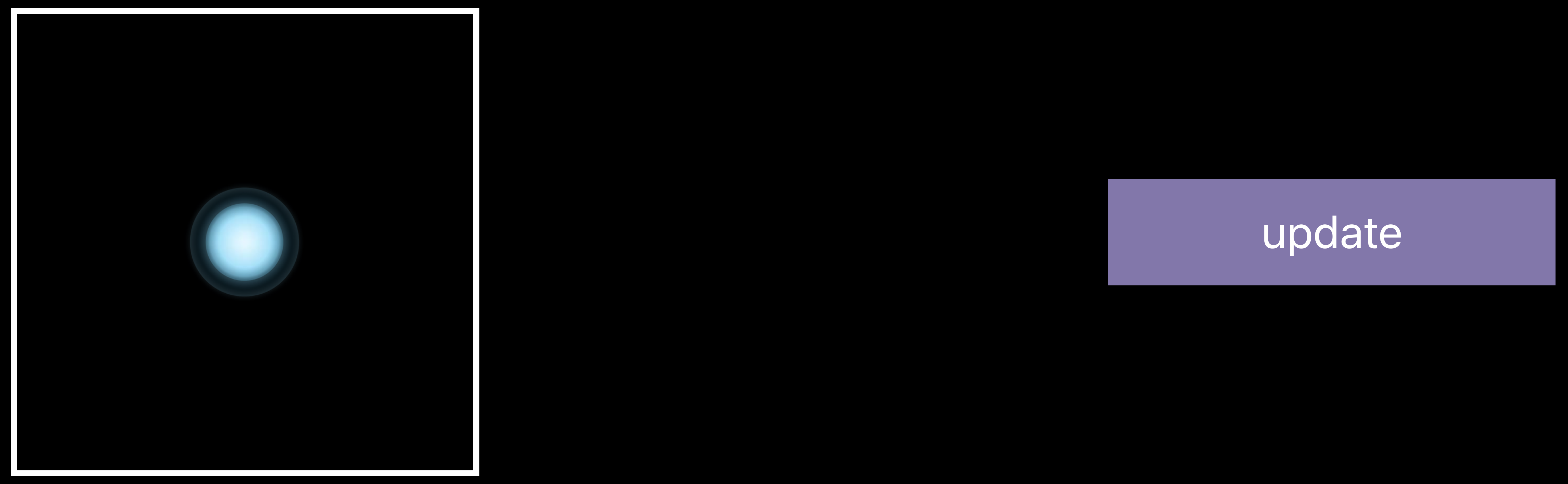
# Drop Sessions

Lifetime



## Drop Sessions

### Lifetime



# Drop Sessions

## Lifetime

update
drop proposal

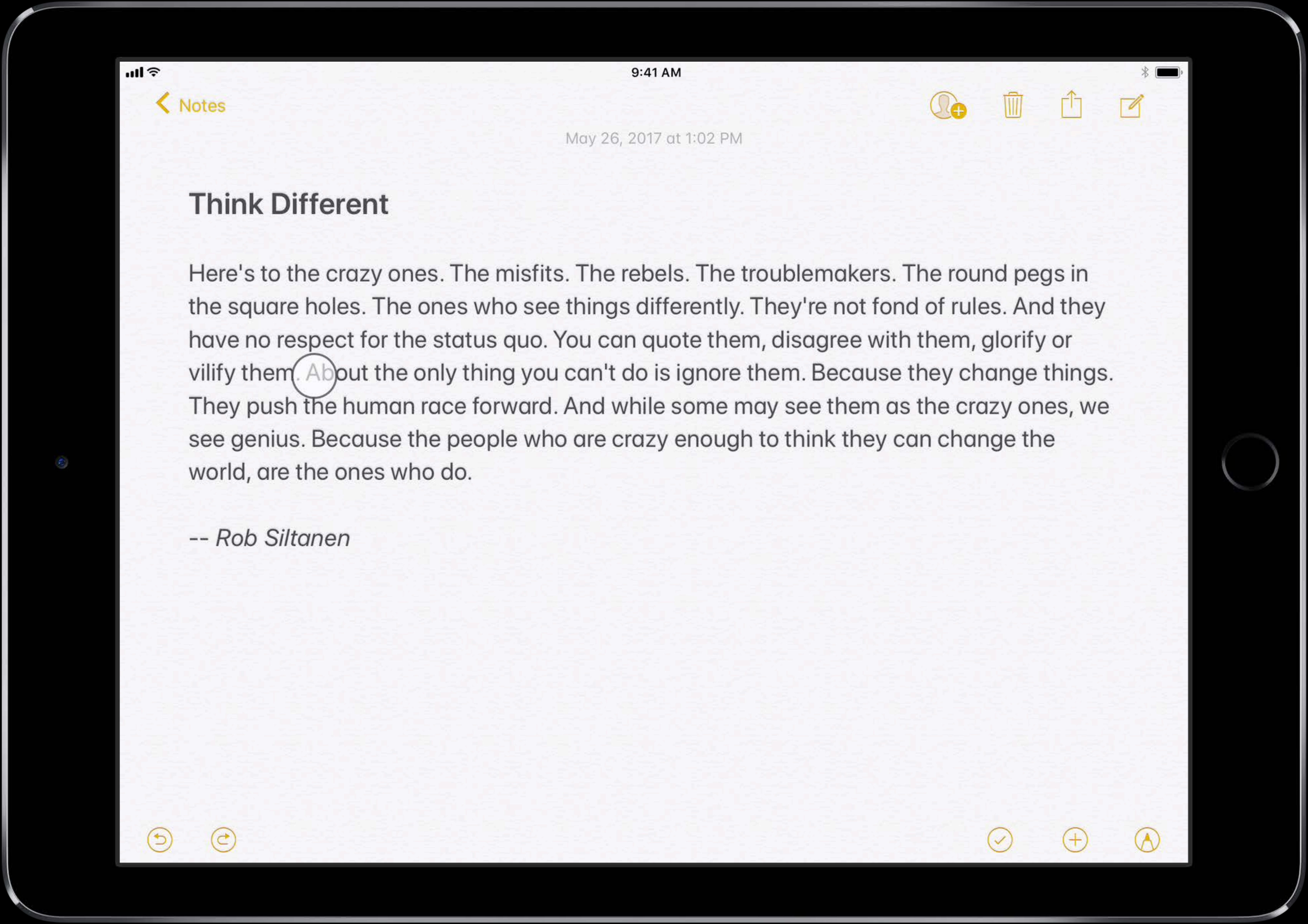
## Drop Proposals

```
class UIDropProposal : NSObject, NSCopying {

    init(operation: UIDropOperation)

    var operation: UIDropOperation { get }

    var isPrecise: Bool

    var prefersFullSizePreview: Bool

}
```

## Drop Proposals

Precision mode

```
class UIDropProposal : NSObject, NSCopying {

    init(operation: UIDropOperation)

    var operation: UIDropOperation { get }

    var isPrecise: Bool

    var prefersFullSizePreview: Bool

}
```

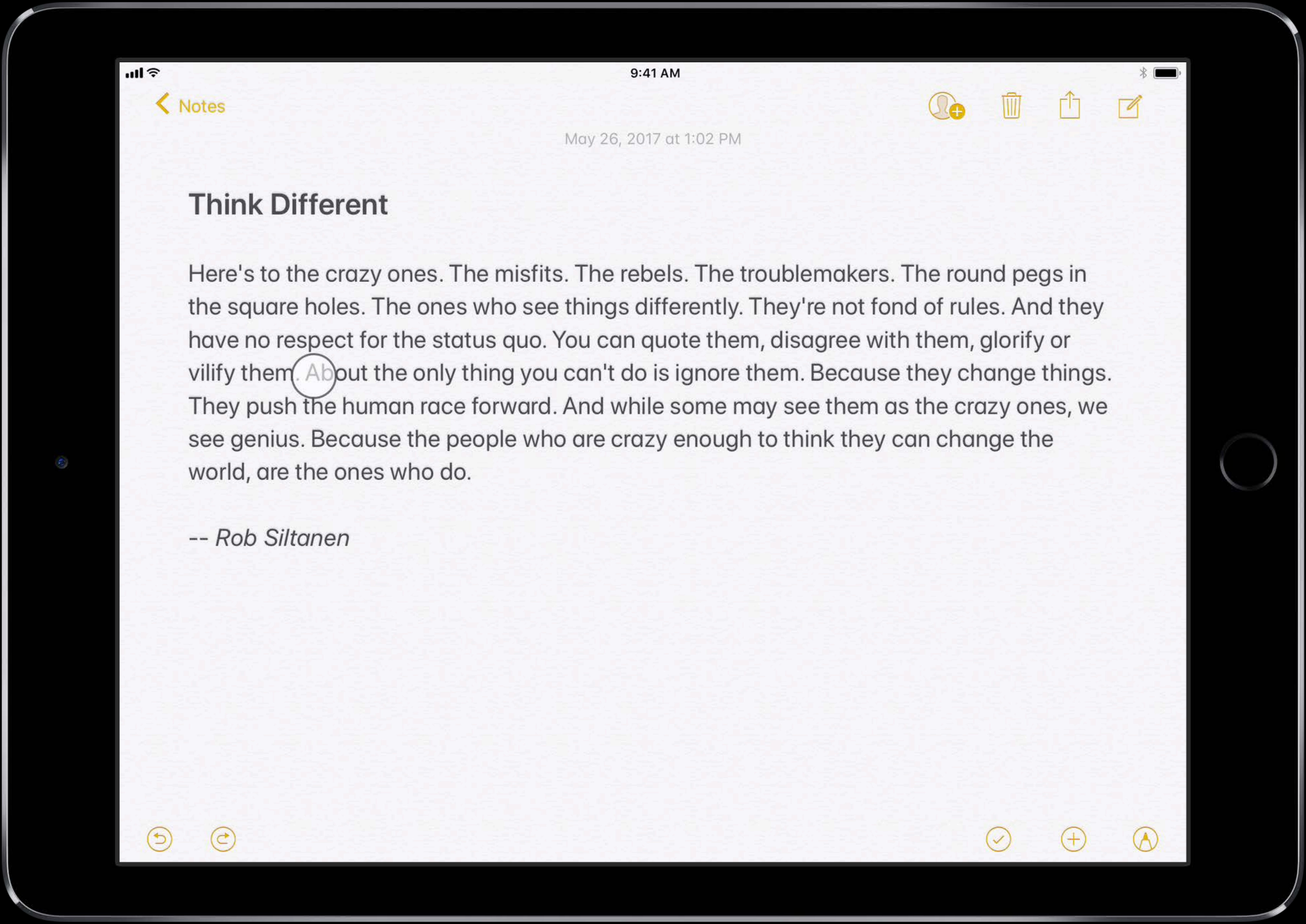


May 26, 2017 at 1:02 PM

Think Different

Here's to the crazy ones. The misfits. The rebels. The troublemakers. The round pegs in the square holes. The ones who see things differently. They're not fond of rules. And they have no respect for the status quo. You can quote them, disagree with them, glorify or vilify them. About the only thing you can't do is ignore them. Because they change things. They push the human race forward. And while some may see them as the crazy ones, we see genius. Because the people who are crazy enough to think they can change the world, are the ones who do.

*-- Rob Siltanen*



May 26, 2017 at 1:02 PM

Think Different

Here's to the crazy ones. The misfits. The rebels. The troublemakers. The round pegs in the square holes. The ones who see things differently. They're not fond of rules. And they have no respect for the status quo. You can quote them, disagree with them, glorify or vilify them. About the only thing you can't do is ignore them. Because they change things. They push the human race forward. And while some may see them as the crazy ones, we see genius. Because the people who are crazy enough to think they can change the world, are the ones who do.

*-- Rob Siltanen*

# Drop Proposals

## Preview scaling

```
class UIDropProposal : NSObject, NSCopying {

    init(operation: UIDropOperation)

    var operation: UIDropOperation { get }

    var isPrecise: Bool

    var prefersFullSizePreview: Bool

}
```

# Drop Proposals

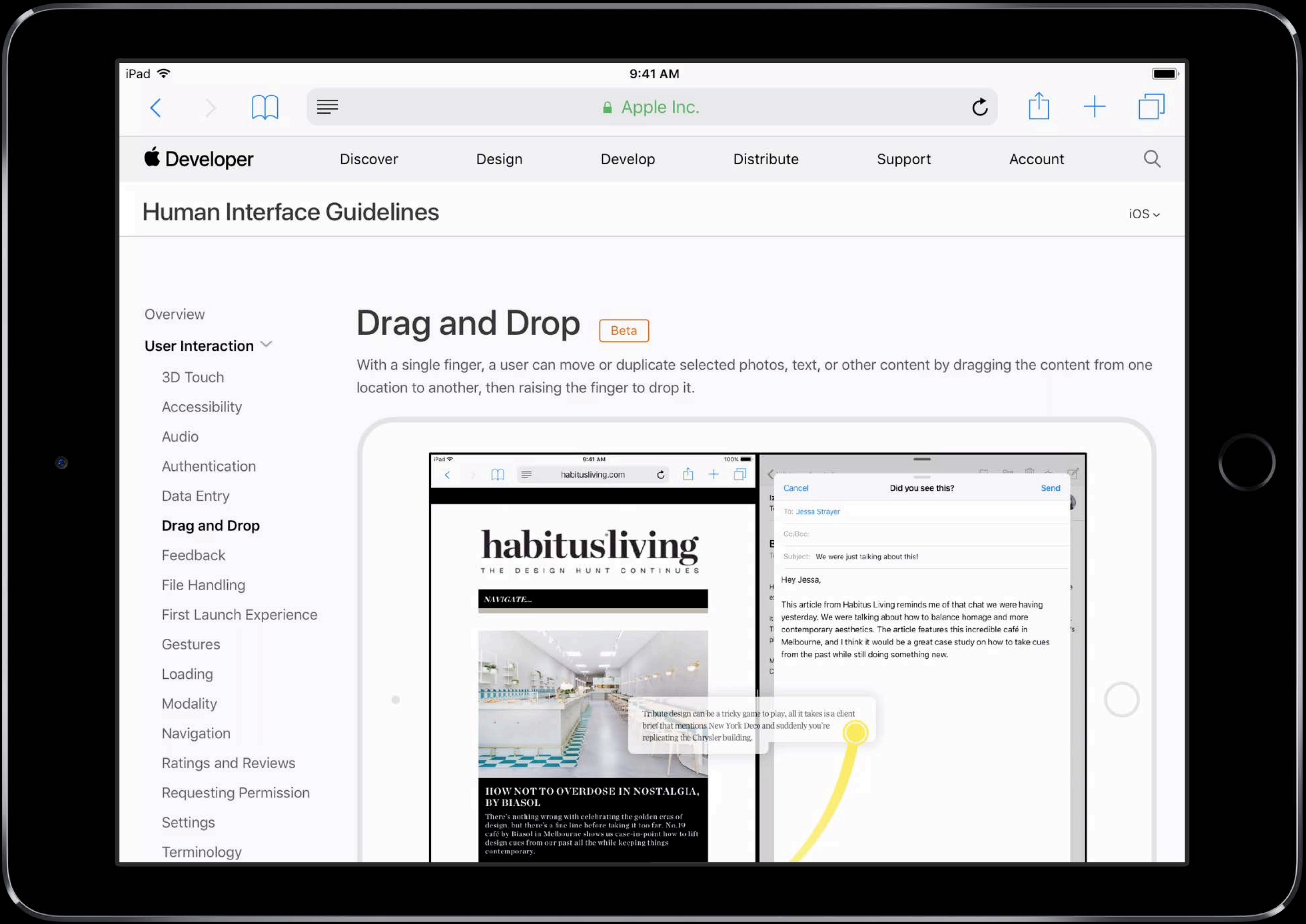
## Preview scaling

```
class UIDropProposal : NSObject, NSCopying {

    init(operation: UIDropOperation)

    var operation: UIDropOperation { get }

    var isPrecise: Bool

    var prefersFullSizePreview: Bool

}
```

## Drop Proposals

Preview scaling

```
class UIDropProposal : NSObject, NSCopying {

    init(operation: UIDropOperation)

    var operation: UIDropOperation { get }

    var isPrecise: Bool

    var prefersFullSizePreview: Bool

}
```



Human Interface Guidelines

Overview
User Interaction
3D Touch
Accessibility
Audio
Authentication
Data Entry
Drag and Drop
Feedback
File Handling
First Launch Experience
Gestures
Loading
Modality
Navigation
Ratings and Reviews
Requesting Permission
Settings
Terminology

Drag and Drop Beta

With a single finger, a user can move or duplicate selected photos, text, or other content by dragging the content from one location to another, then raising the finger to drop it.

HOW NOT TO OVERDOSE IN NOSTALGIA, BY BIASOL

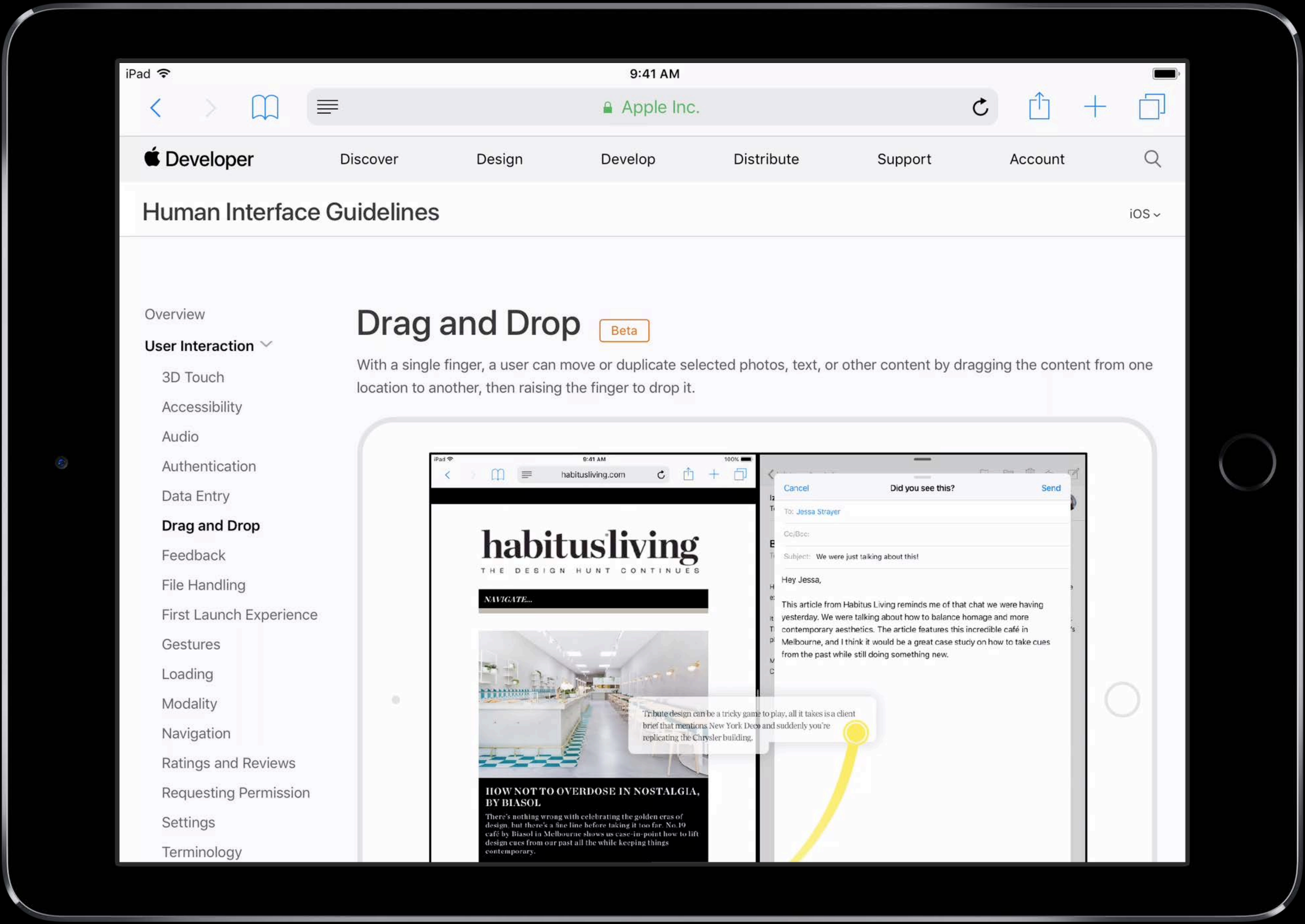
There's nothing wrong with celebrating the golden eras of design, but there's a fine line before taking it too far. No.19 café by Biasol in Melbourne shows us case-in-point how to lift design cues from our past all the while keeping things contemporary.

Tribute design can be a tricky game to play, all it takes is a client brief that mentions New York Deco and suddenly you're replicating the Chrysler building.

To: Jessa Strayer
Cc/Bcc:
Subject: We were just talking about this!

Hey Jessa,

This article from Habitus Living reminds me of that chat we were having yesterday. We were talking about how to balance homage and more contemporary aesthetics. The article features this incredible café in Melbourne, and I think it would be a great case study on how to take cues from the past while still doing something new.



## Human Interface Guidelines

Overview
User Interaction
3D Touch
Accessibility
Audio
Authentication
Data Entry
Drag and Drop
Feedback
File Handling
First Launch Experience
Gestures
Loading
Modality
Navigation
Ratings and Reviews
Requesting Permission
Settings
Terminology

## Drag and Drop Beta

With a single finger, a user can move or duplicate selected photos, text, or other content by dragging the content from one location to another, then raising the finger to drop it.

HOW NOT TO OVERDOSE IN NOSTALGIA, BY BIASOL

There's nothing wrong with celebrating the golden eras of design, but there's a fine line before taking it too far. No.19 café by Biasol in Melbourne shows us case-in-point how to lift design cues from our past all the while keeping things contemporary.

To: Jessa Strayer
Cc/Bcc:
Subject: We were just talking about this!

Hey Jessa,

This article from Habitus Living reminds me of that chat we were having yesterday. We were talking about how to balance homage and more contemporary aesthetics. The article features this incredible café in Melbourne, and I think it would be a great case study on how to take cues from the past while still doing something new.

Tribute design can be a tricky game to play, all it takes is a client brief that mentions New York Deco and suddenly you're replicating the Chrysler building.

## Drop Proposals

Preview scaling

System prefers to scale down all previews

Prefer full-size previews when it makes sense

## Drop Proposals

Preview scaling

System prefers to scale down all previews

Prefer full-size previews when it makes sense

```
optional func dragInteraction(_ interaction: UIDragInteraction,
         prefersFullSizePreviewsFor session: UIDragSession) -> Bool
```

```
class UIDropProposal : NSObject, NSCopying {

    var prefersFullSizePreview: Bool

}
```

## Drop Proposals

Preview scaling

## Drop Proposals

Preview scaling

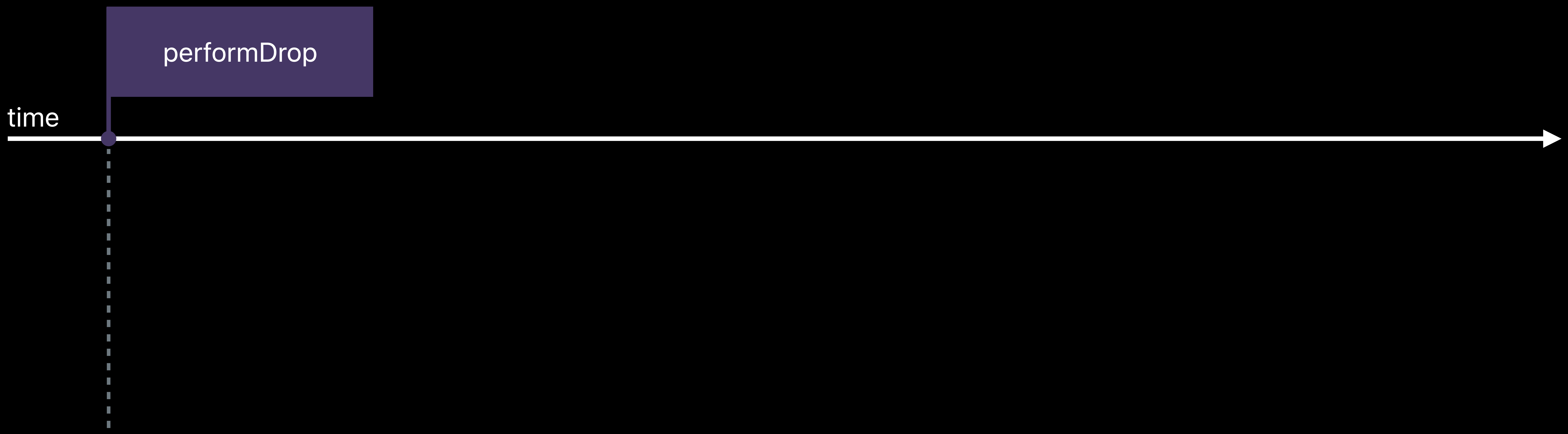
Flocks always scale

A single preview will always scale once it leaves the source app

Once scaled down, a preview never scales back up

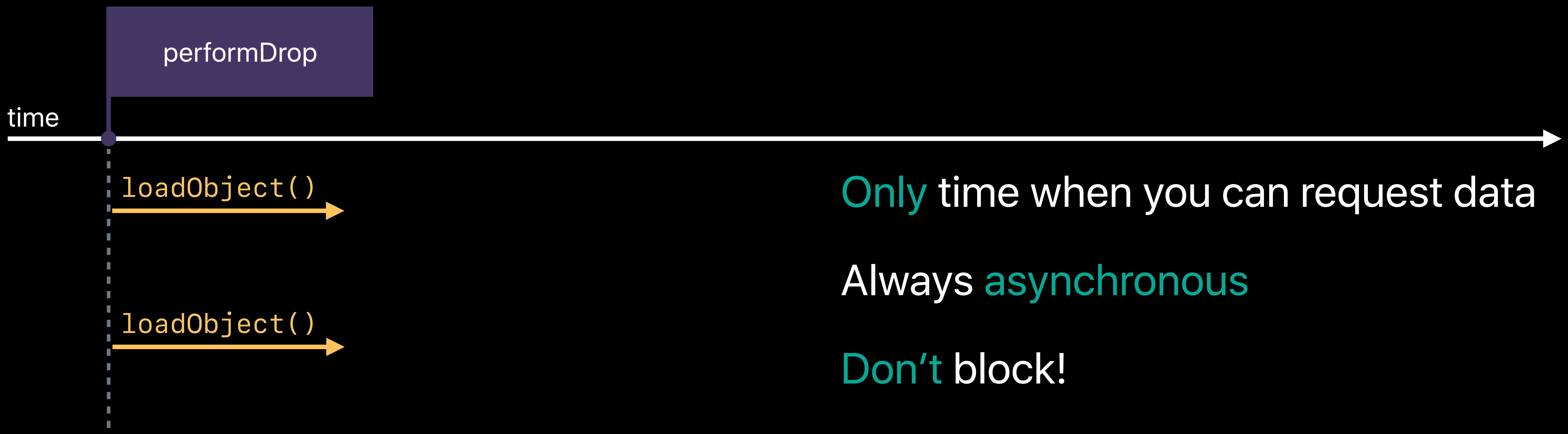
## Performing a Drop

## Performing a Drop



```
optional func dropInteraction(_ interaction: UIDropInteraction,
                              performDrop session: UIDropSession)
```

# Performing a Drop



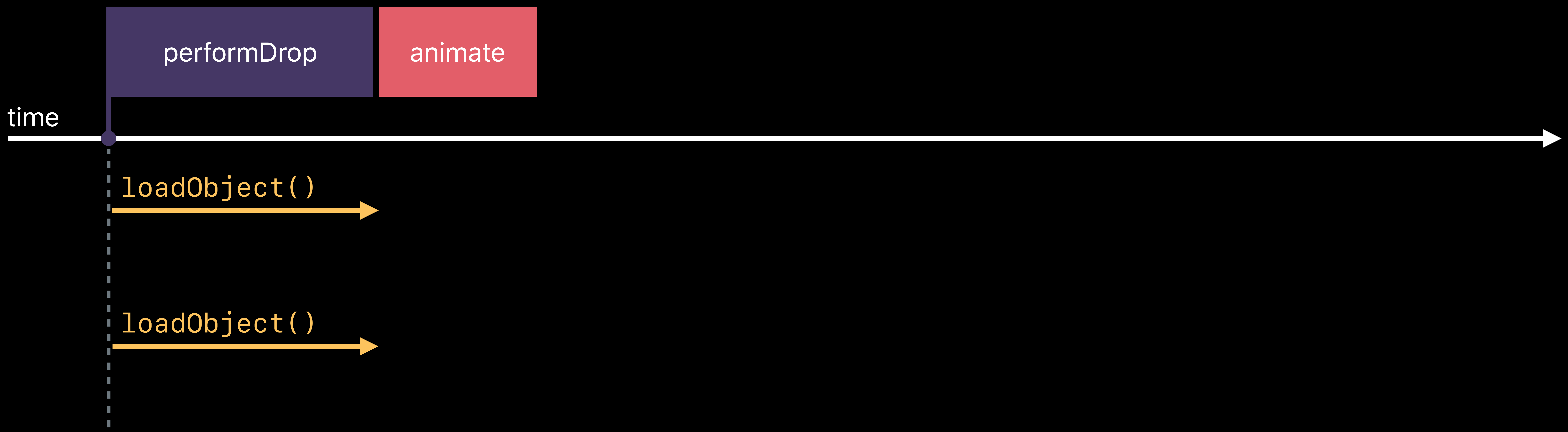
Only time when you can request data

Always asynchronous

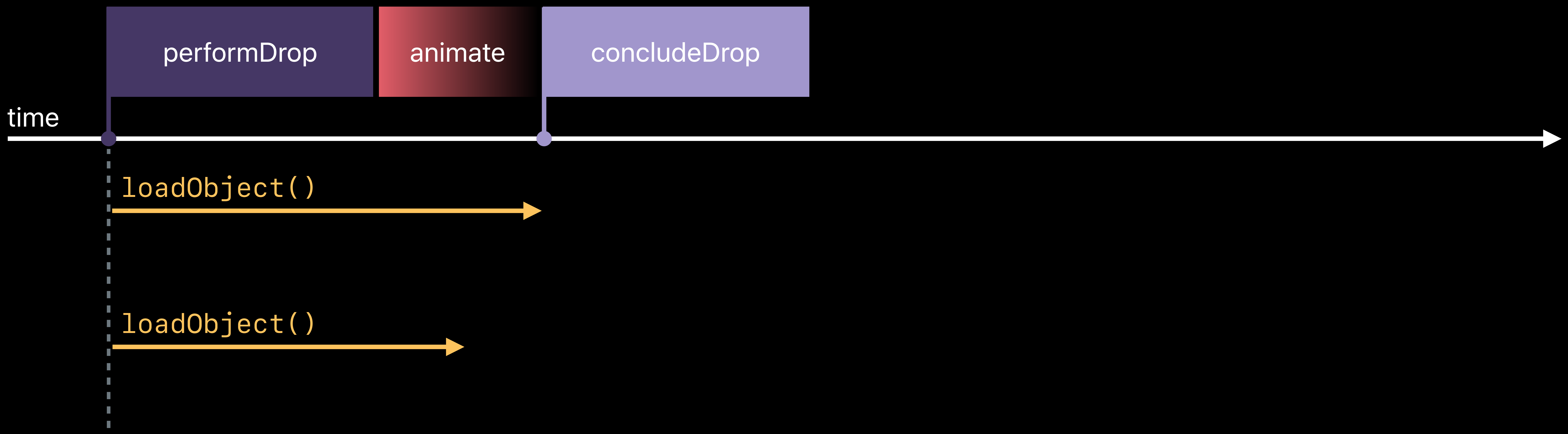
Don't block!

```
optional func dropInteraction(_ interaction: UIDropInteraction,
                              performDrop session: UIDropSession)
```

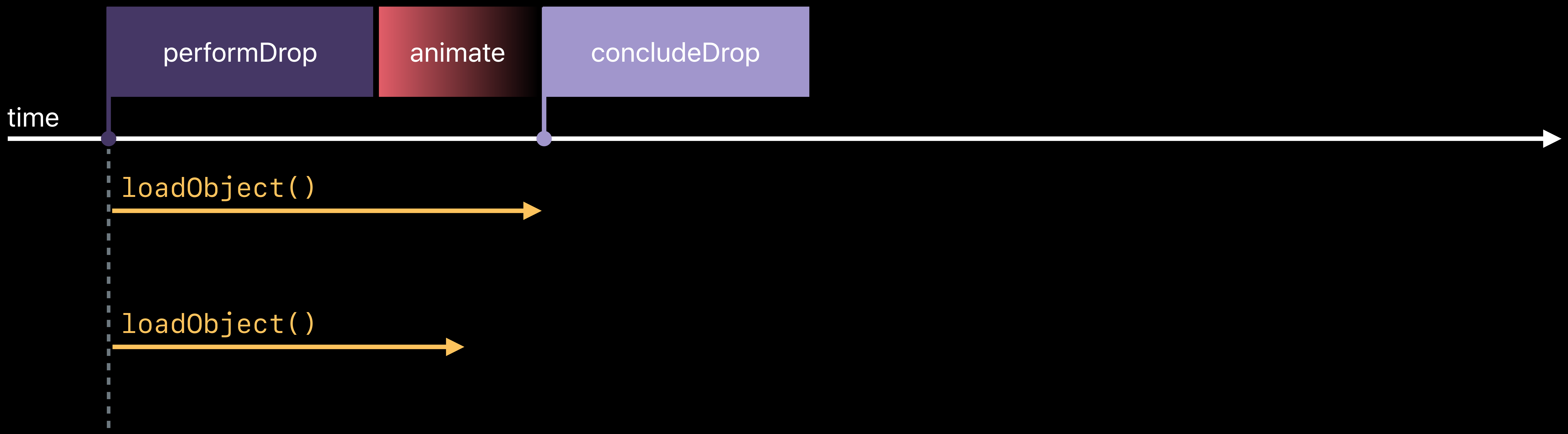
# Performing a Drop



# Performing a Drop

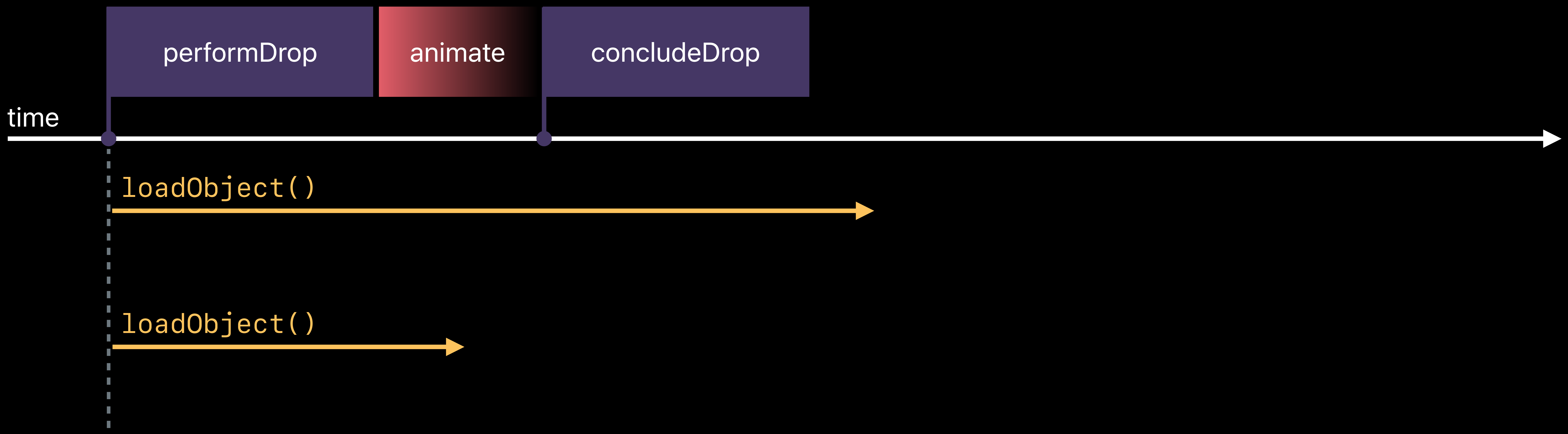


# Performing a Drop



```
optional func dropInteraction(_ interaction: UIDropInteraction,
                         concludeDrop session: UIDropSession)
```

## Performing a Drop



## Performing a Drop

Loading homogeneous data

## Performing a Drop

Loading homogeneous data

Use session’s `loadObjects(ofClass:completion:)`

Load all objects conforming to a class

Resulting array is sorted same as session’s `items` array

⚠️ Completion block will be called on main queue

## Performing a Drop

Loading heterogeneous data

## Performing a Drop

Loading heterogeneous data

Iterate over session items and load each item individually

`loadObject(), loadDataRepresentation(), loadFileRepresentation(), …`

Fine-grained control

Load multiple representations

⚠️ Completion block will be called on background queue

## Performing a Drop

Loading heterogeneous data

Iterate over session items and load each item individually

```
loadObject(), loadDataRepresentation(), loadFileRepresentation(), …
```

Fine-grained control

Load multiple representations

⚠ Completion block will be called on background queue

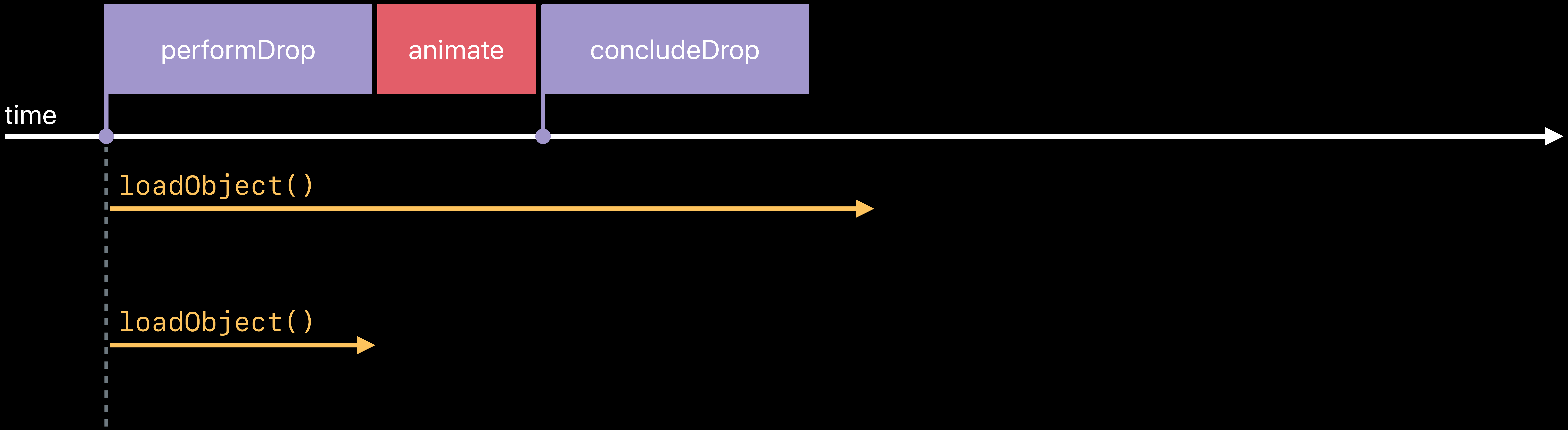
## *Demo*

Wenson Hsieh, WebKit

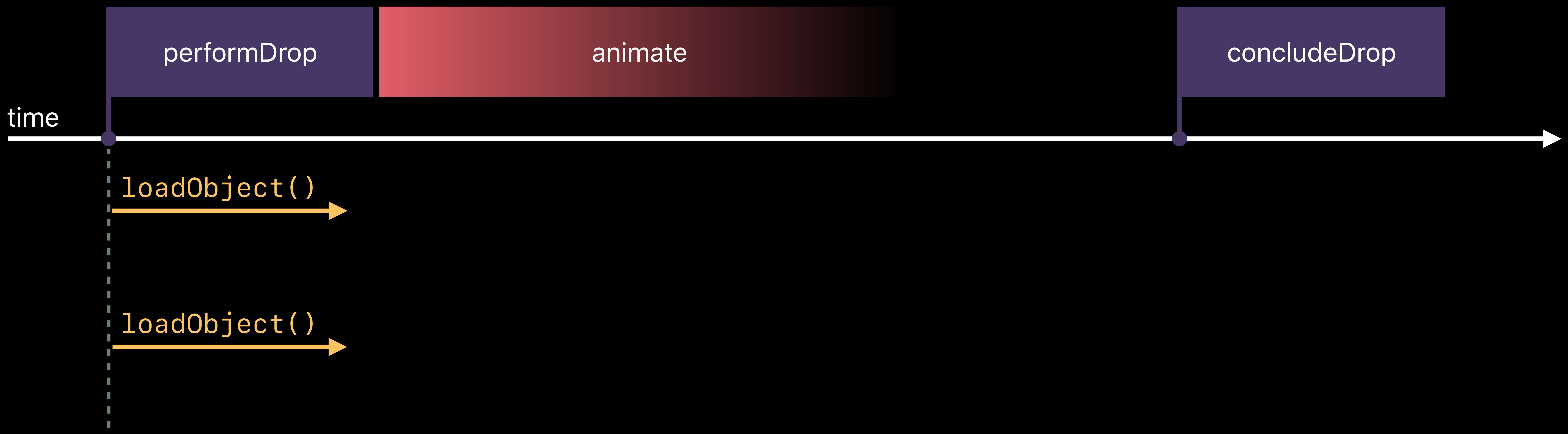
## Drop Previews and Animations

Tom Adriaenssen, UIKit

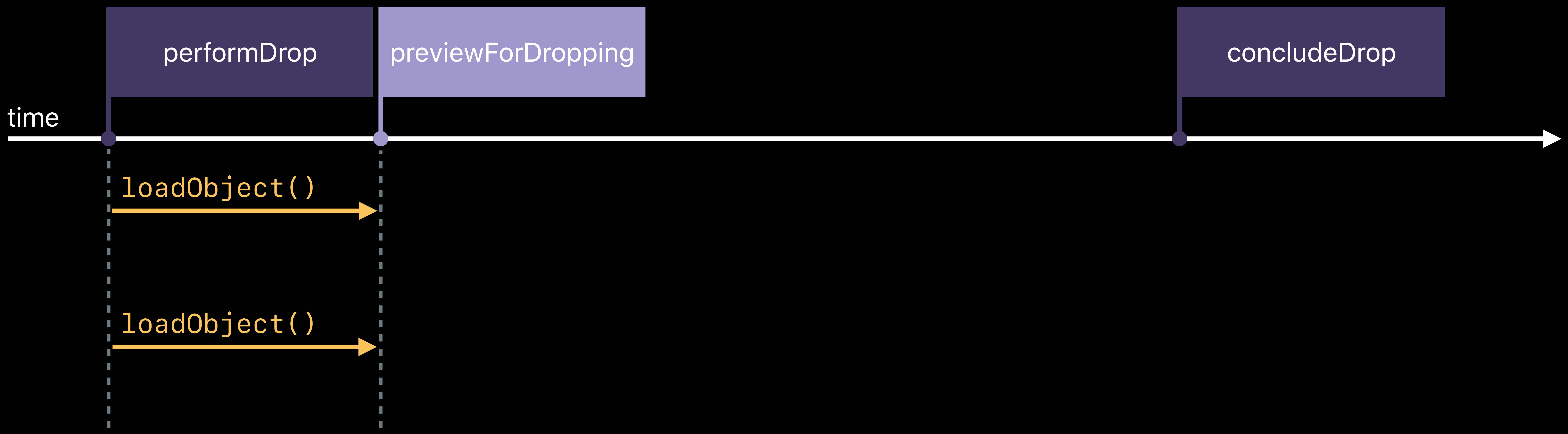
## Drop Previews



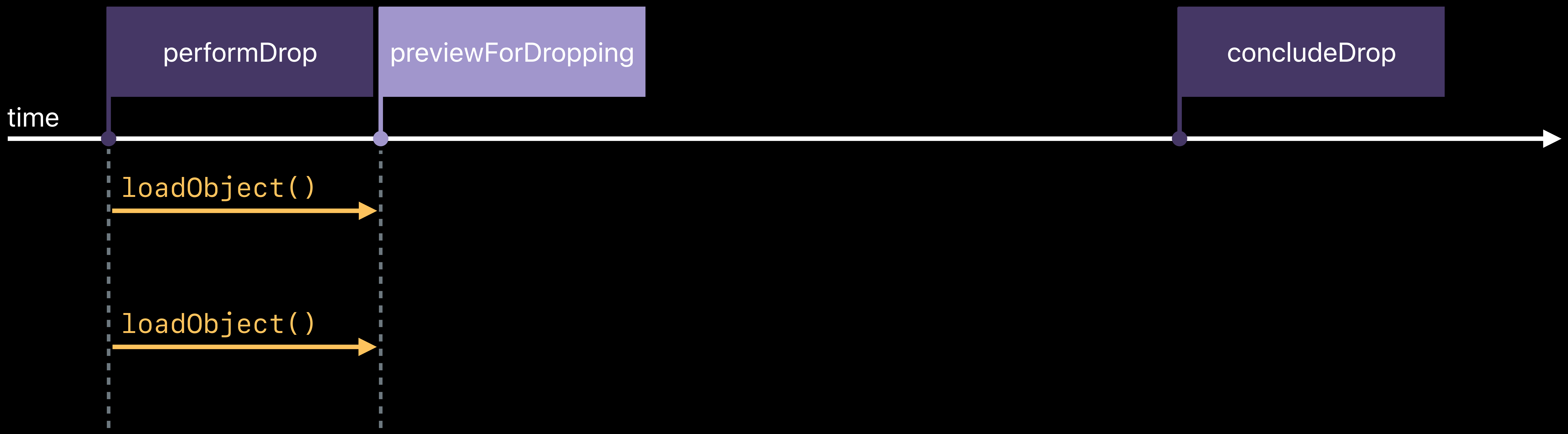
# Drop Previews



## Drop Previews

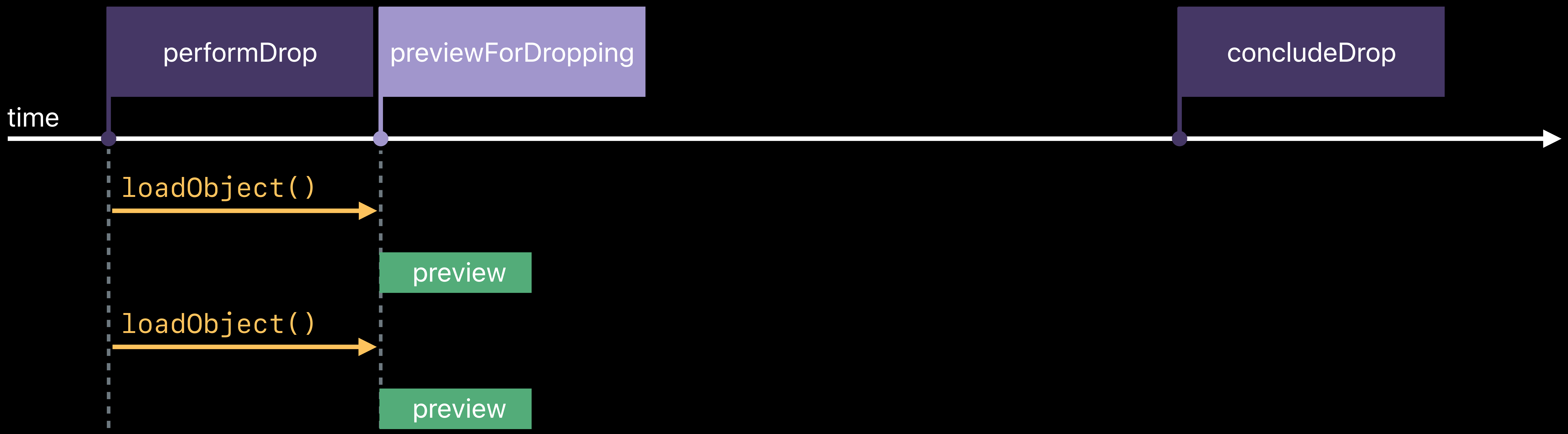


## Drop Previews



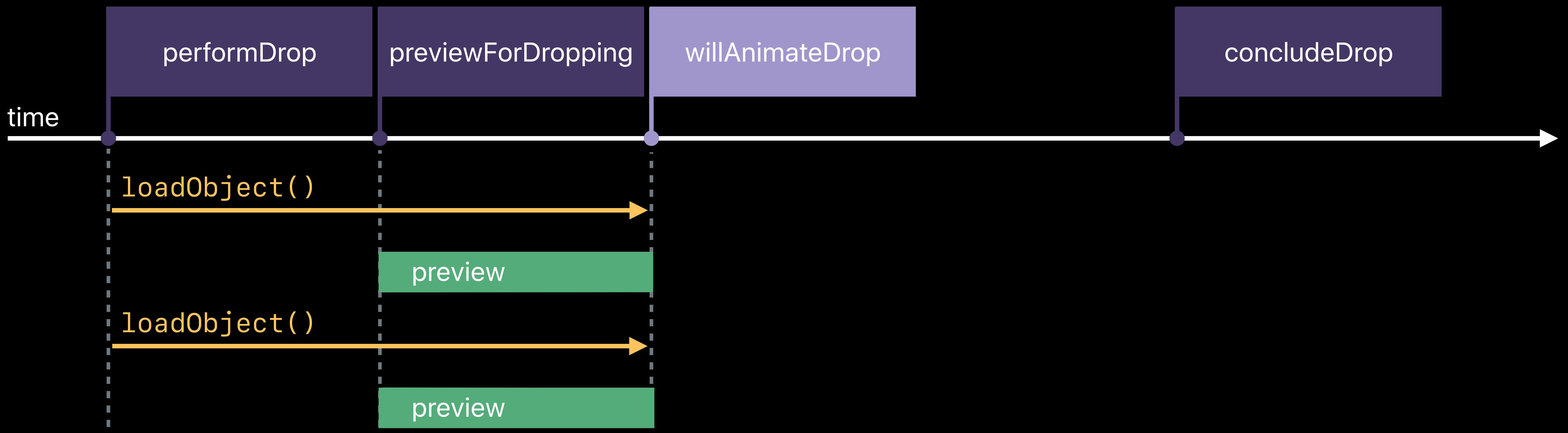
```
optional func dropInteraction(_ interaction: UIDropInteraction,
                        previewForDropping item: UIDragItem,
                     withDefault defaultPreview: UITargetedDragPreview?) -> UITargetedDragPreview?
```

# Drop Previews

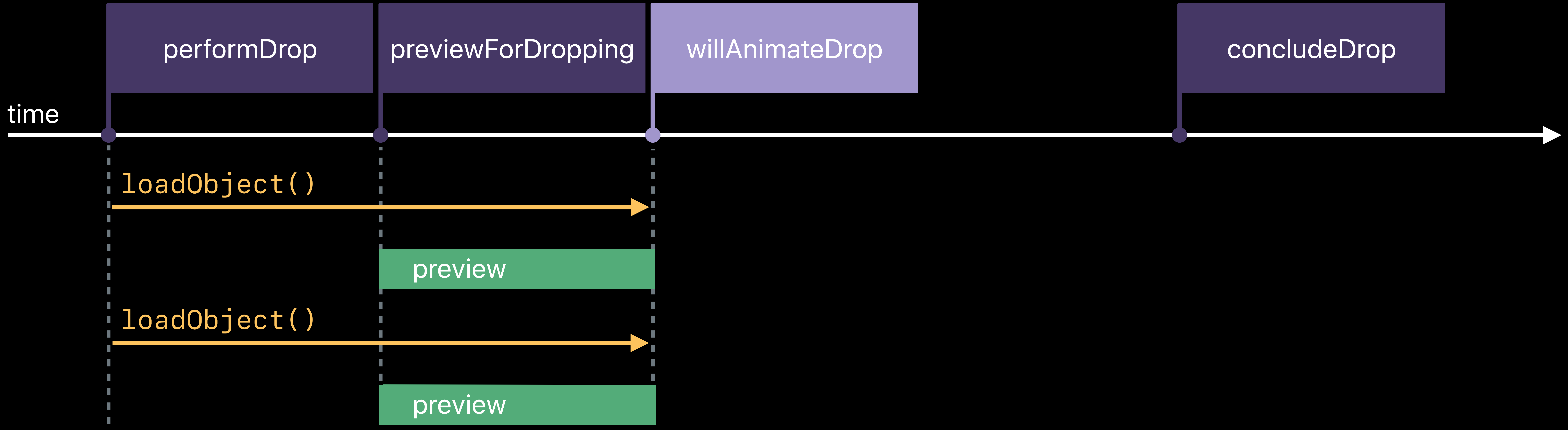


```
optional func dropInteraction(_ interaction: UIDropInteraction,
                   previewForDropping item: UIDragItem,
                withDefault defaultPreview: UITargetedDragPreview?) -> UITargetedDragPreview?
```

# Drop Previews

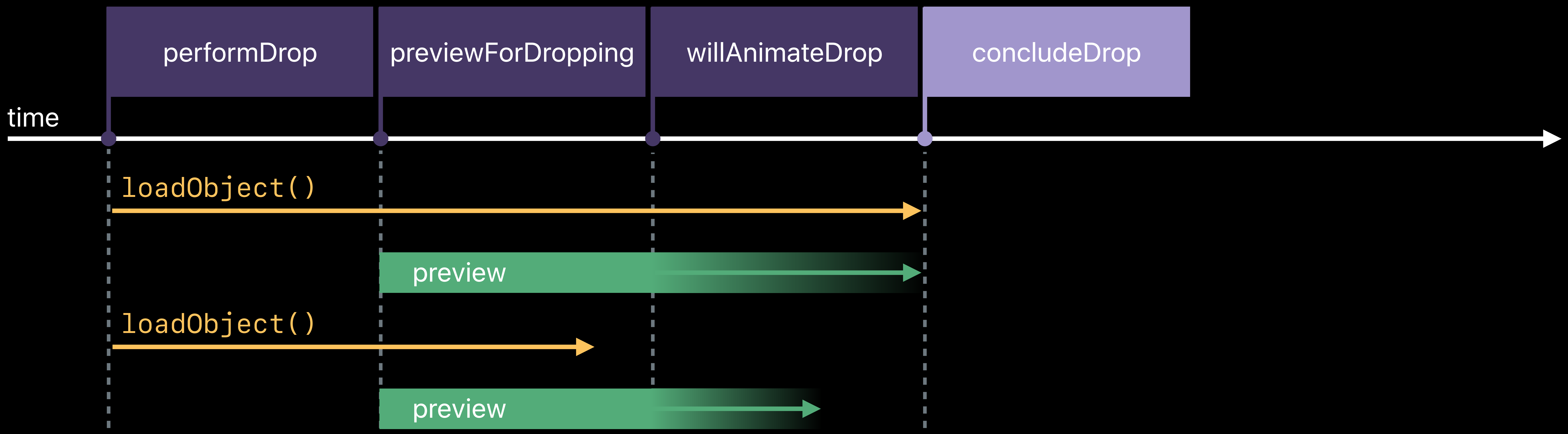


# Drop Previews

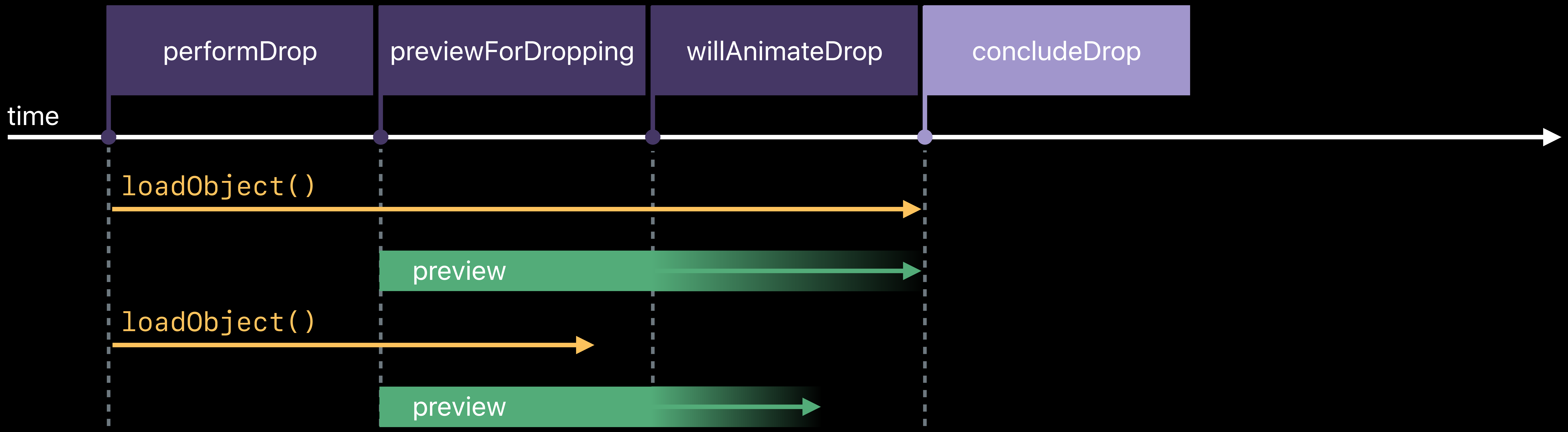


```
optional func dropInteraction(_ interaction: UIDropInteraction,
                                        item: UIDragItem,
                 willAnimateDropWith animator: UIDragAnimating)
```

# Drop Previews

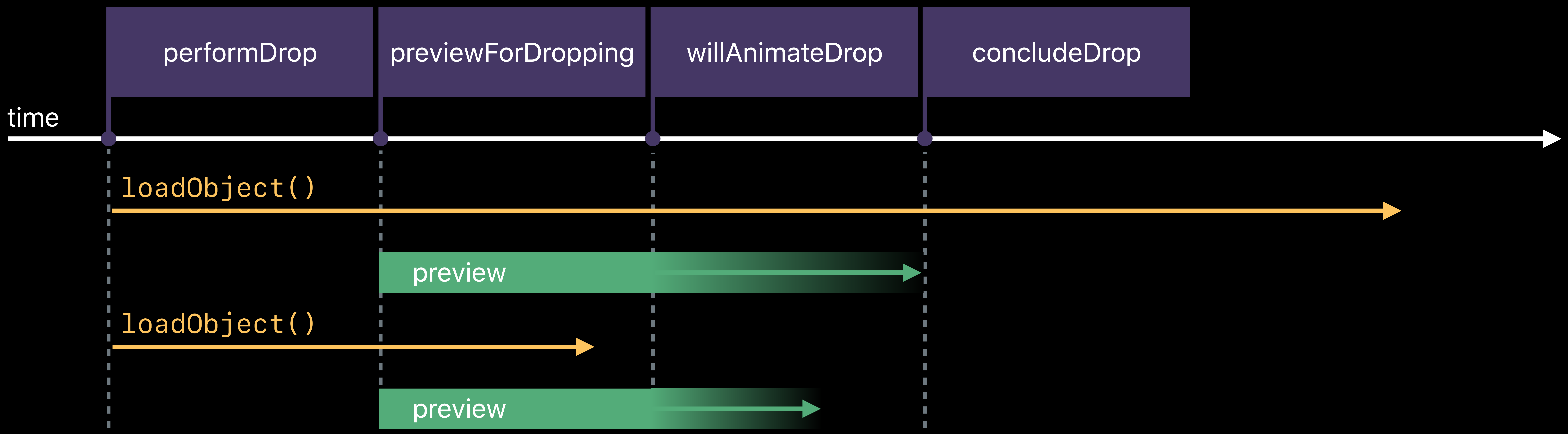


# Drop Previews



```
optional func dropInteraction(_ interaction: UIDropInteraction,
                              concludeDrop session: UIDropSession)
```

# Drop Previews



## Drop Previews... and Cancel Previews

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

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

## Drop Previews… and Cancel Previews

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

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

## Drop Previews... and Cancel Previews

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

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

Same approach

Different occasions

## Drop and Cancel Previews

Animating alongside

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

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

## Drop and Cancel Previews

Animating alongside

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

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

Called for each item in the flock

Similar in API to `UIViewPropertyAnimator`

# Drop and Cancel Previews

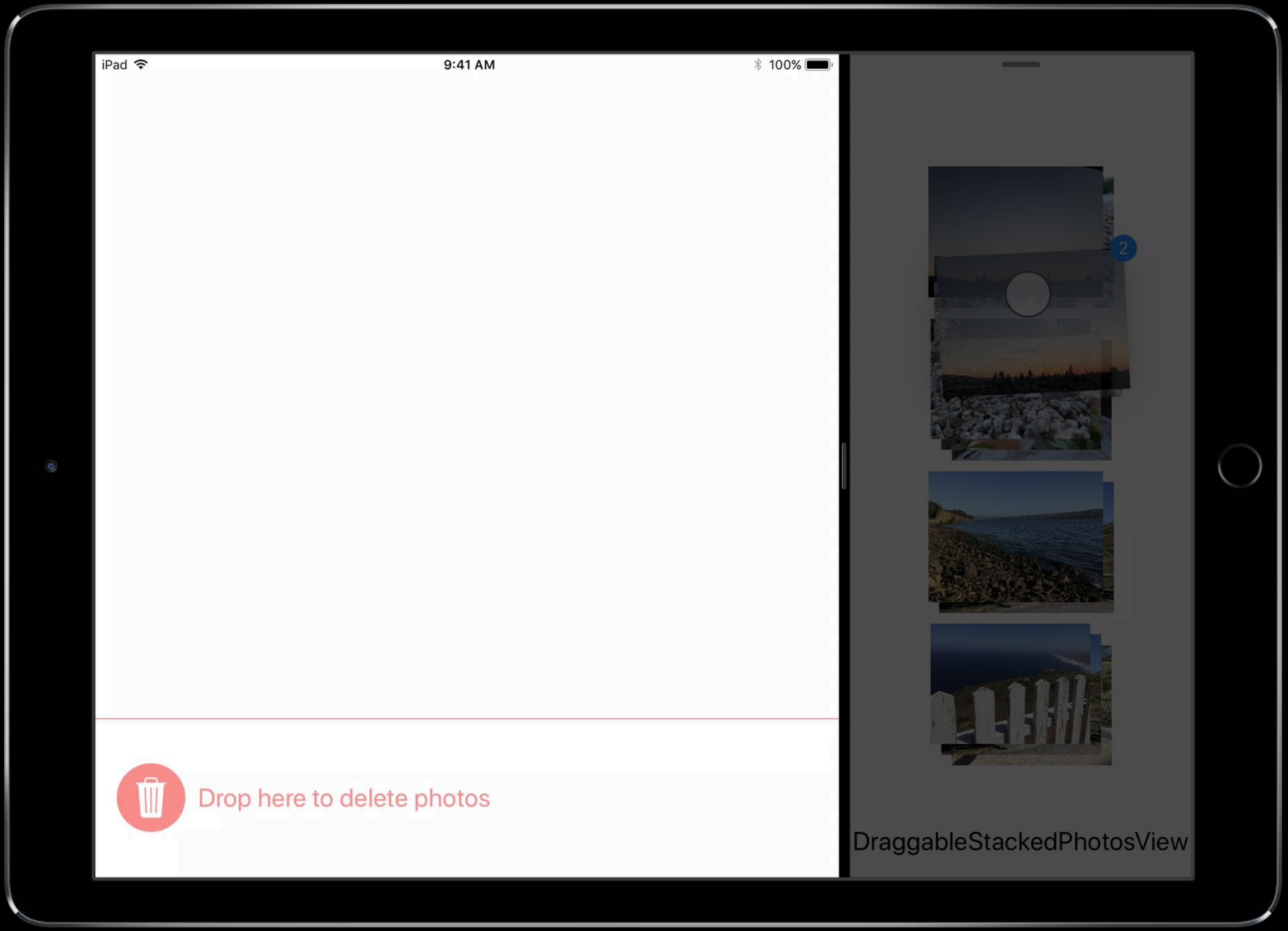
## Animating alongside

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

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

Called for each item in the flock

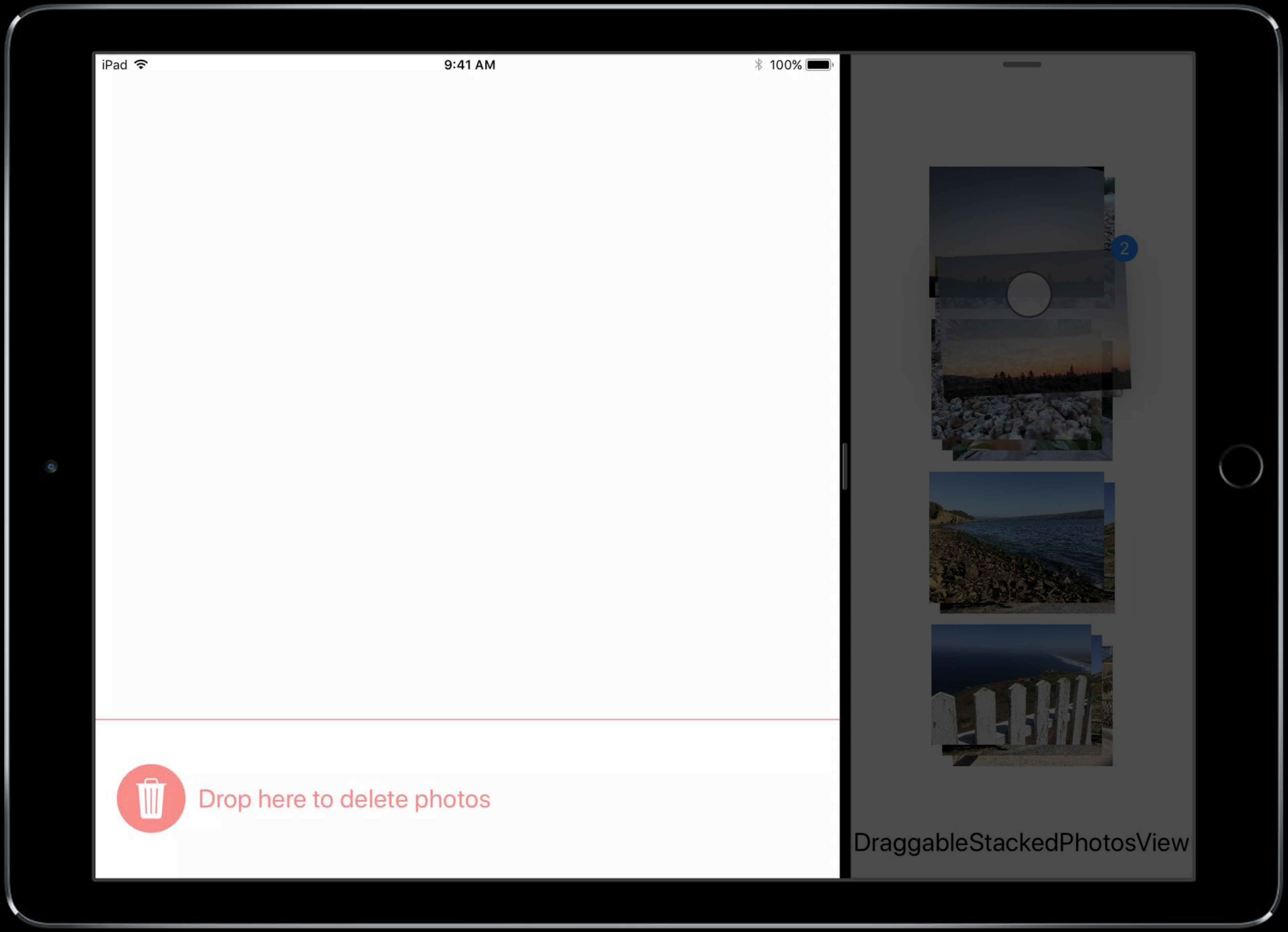
Similar in API to `UIViewPropertyAnimator`

## Drop and Cancel Previews



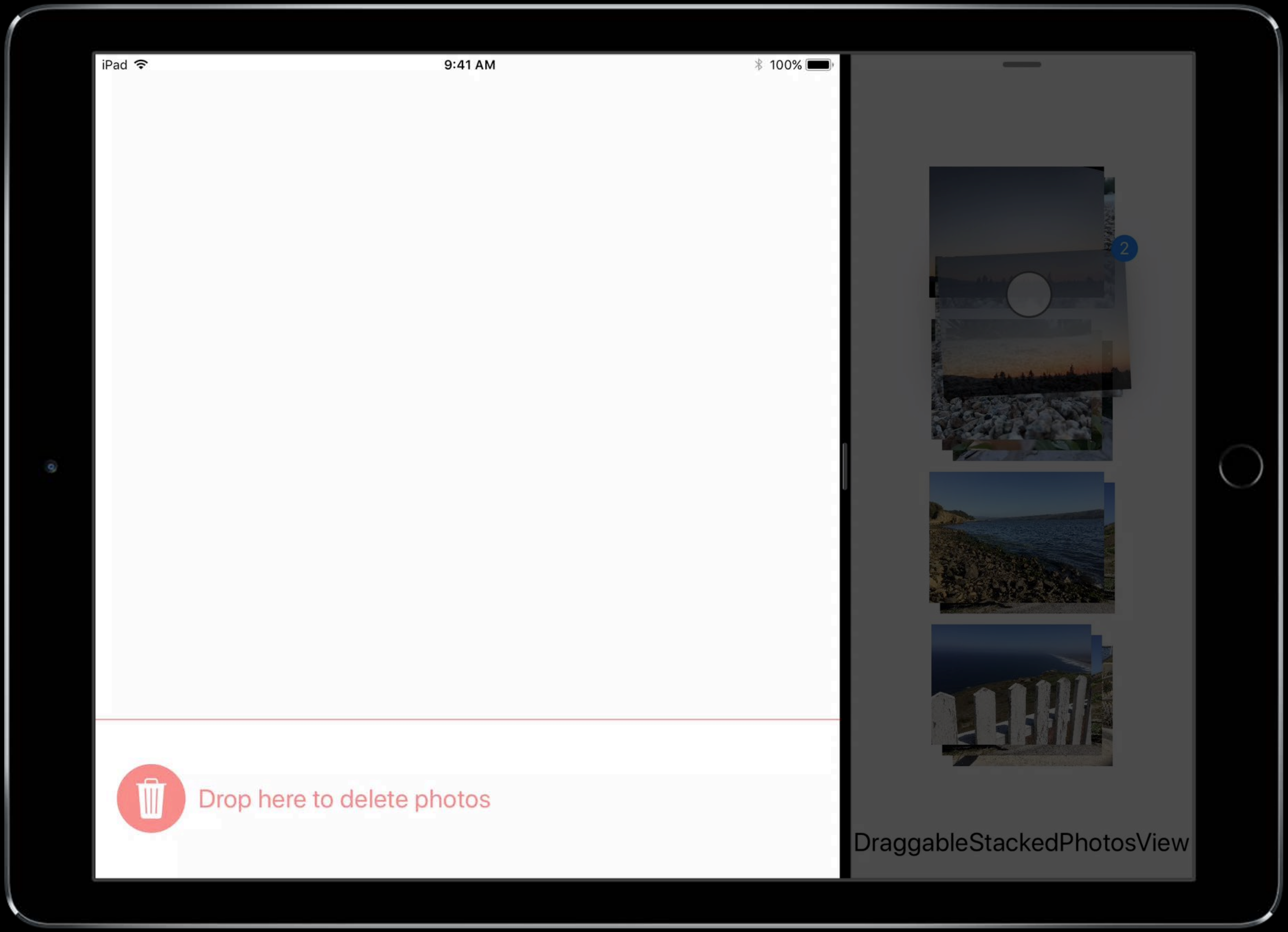
## Drop and Cancel Previews

Return `defaultPreview`



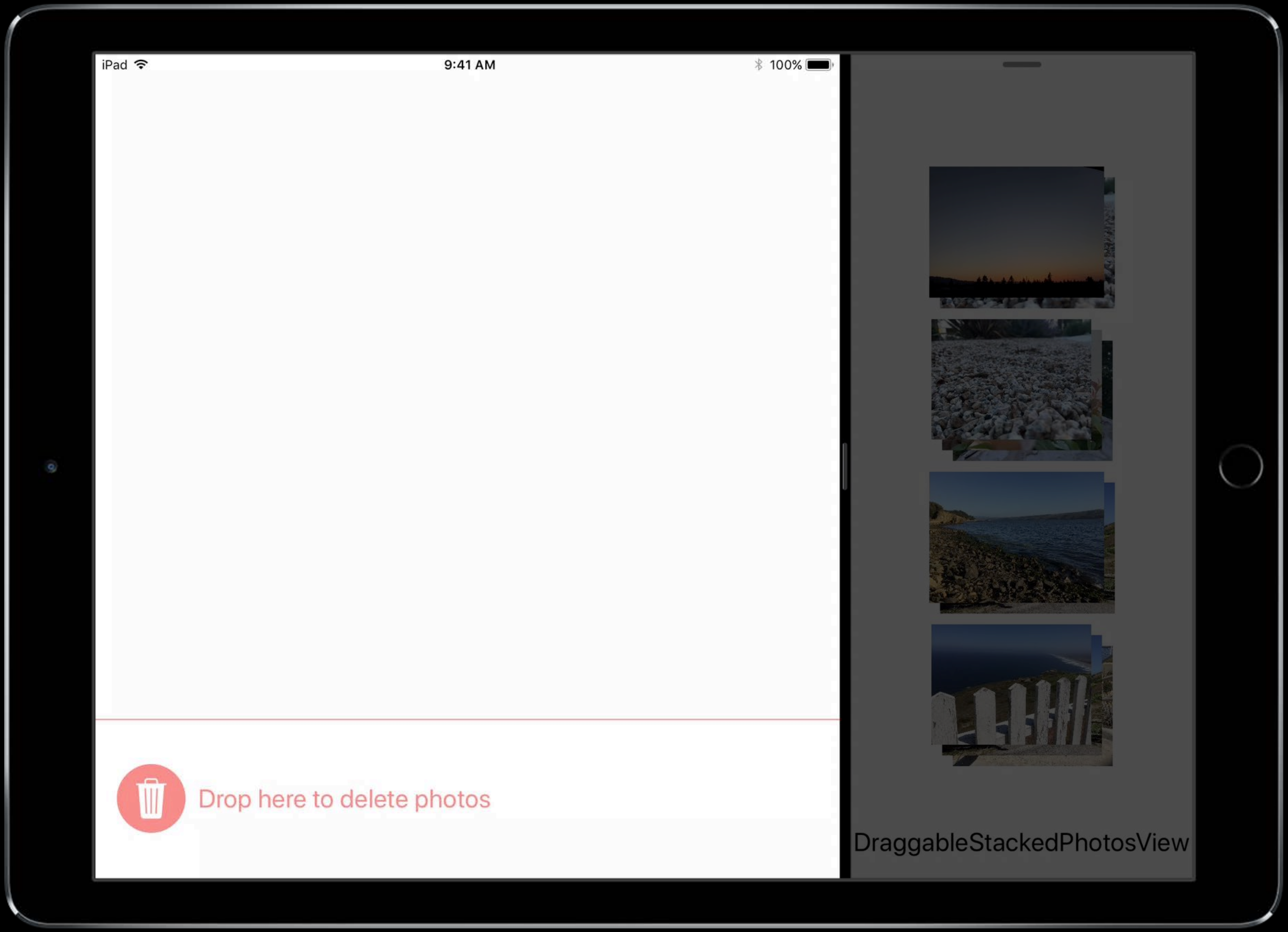
# Drop and Cancel Previews

- ✗ Return `defaultPreview`
- ✓ Return `nil`



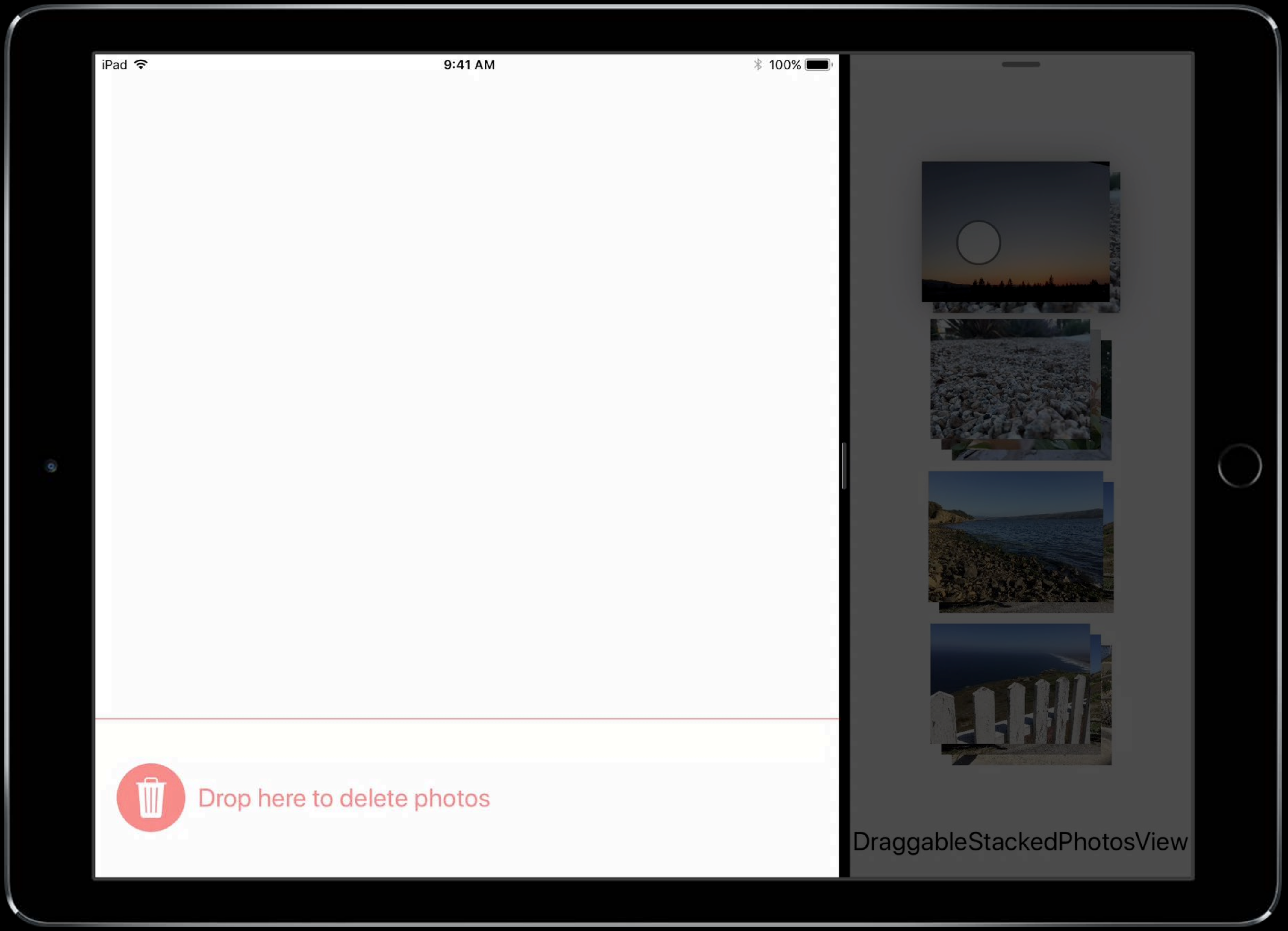
## Drop and Cancel Previews

- ✗ Return `defaultPreview`
- ✓ Return `nil`
- ✓ Retarget the `defaultPreview`



## Drop and Cancel Previews

- ✗ Return `defaultPreview`
- ✓ Return `nil`
- ✓ Retarget the `defaultPreview`
- ✓ Make your own preview



## Drop and Cancel Previews

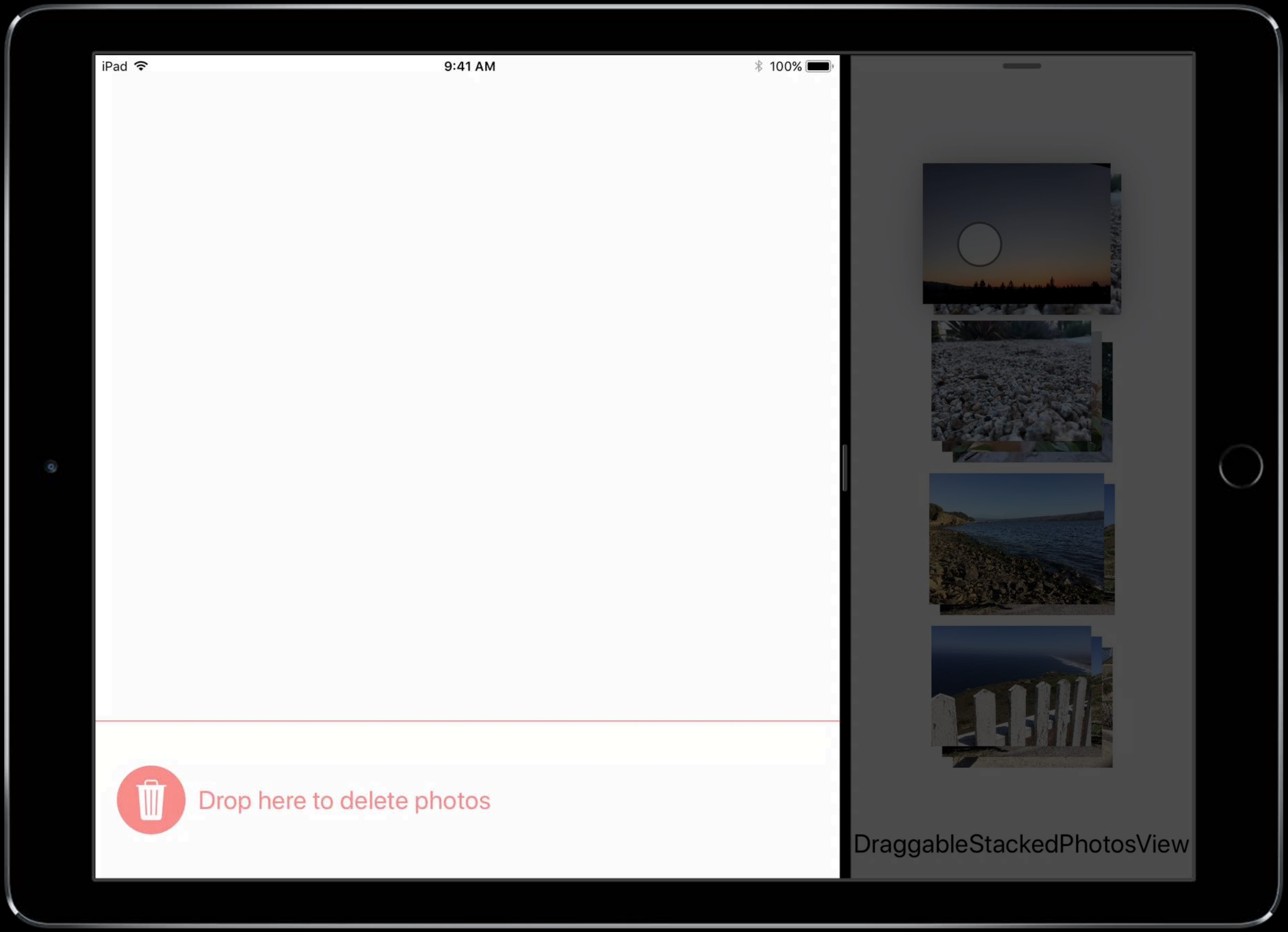
Some limits

## Drop and Cancel Previews

Some limits

Fewer items

- Preview for each item
- Alongside animations for each item



# Drop and Cancel Previews

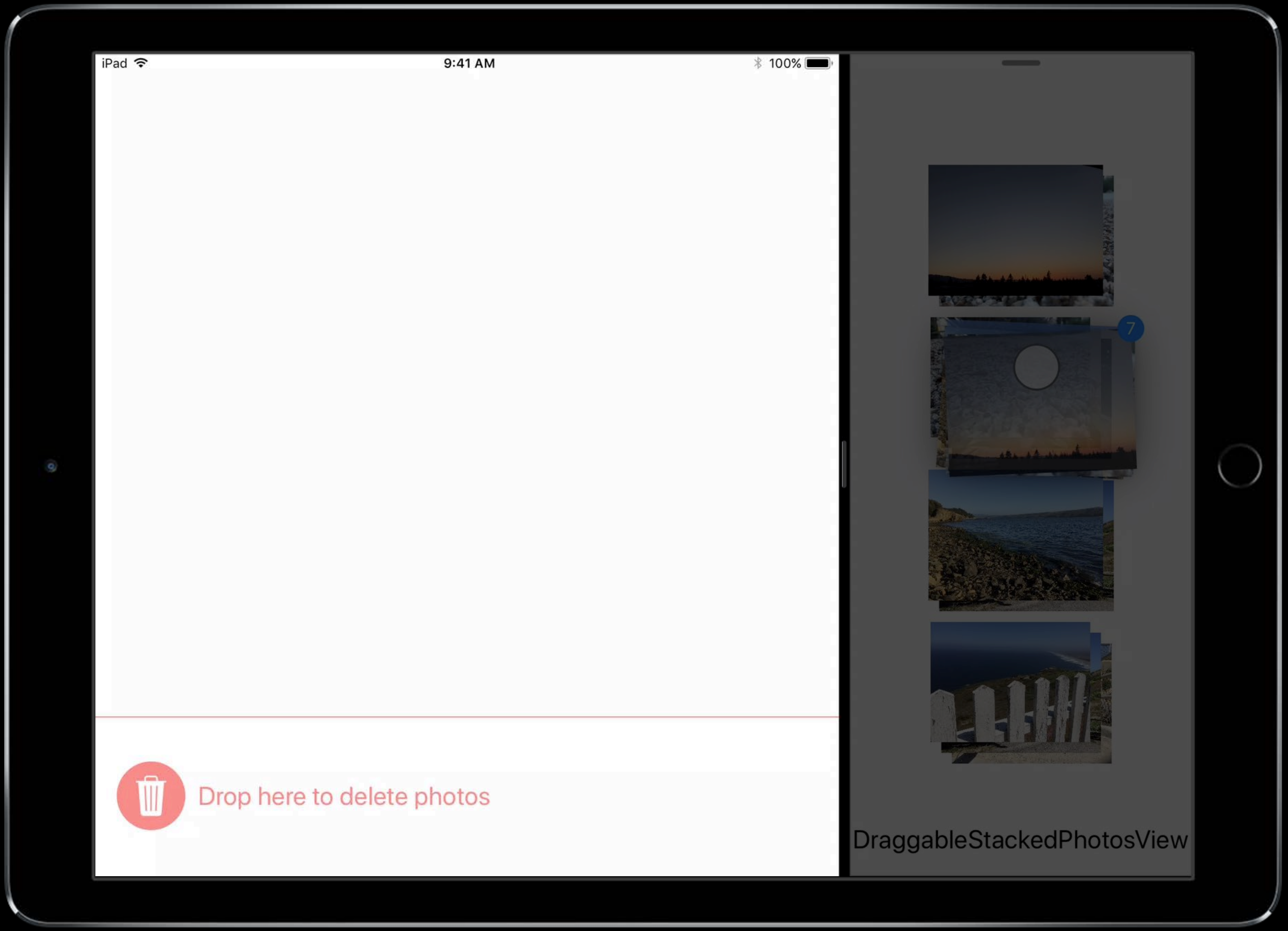
Some limits

Fewer items

- Preview for each item
- Alongside animations for each item

More items

- Default previews for all
- One alongside animation



## *Demo*

Wenson Hsieh, WebKit

## Dealing with Slow Data Delivery

Tom Adriaenssen, UIKit

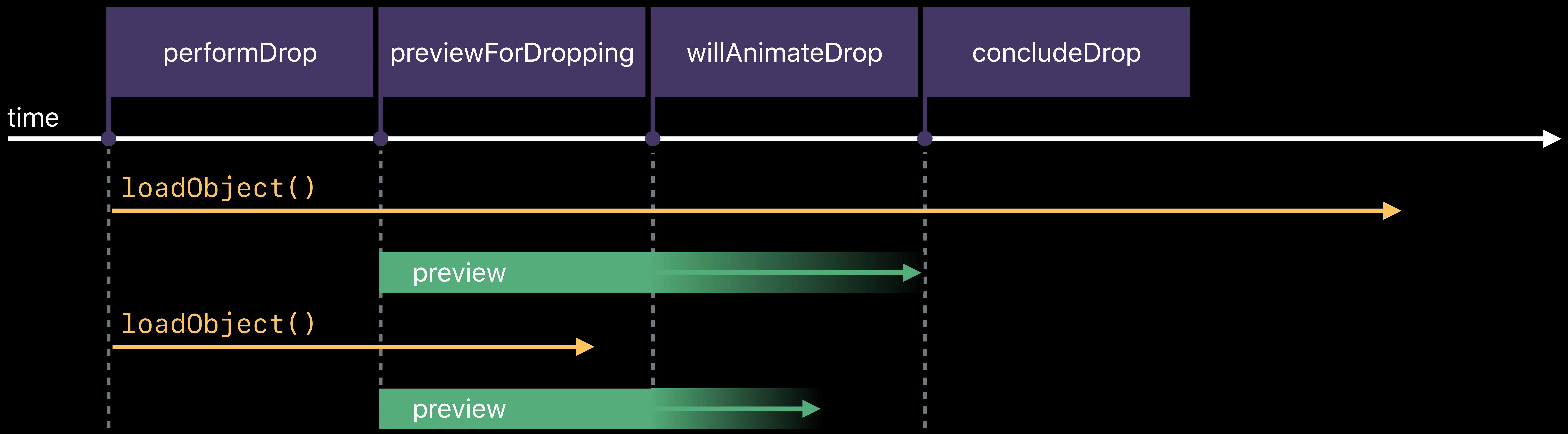
## Dealing with Slow Data Delivery

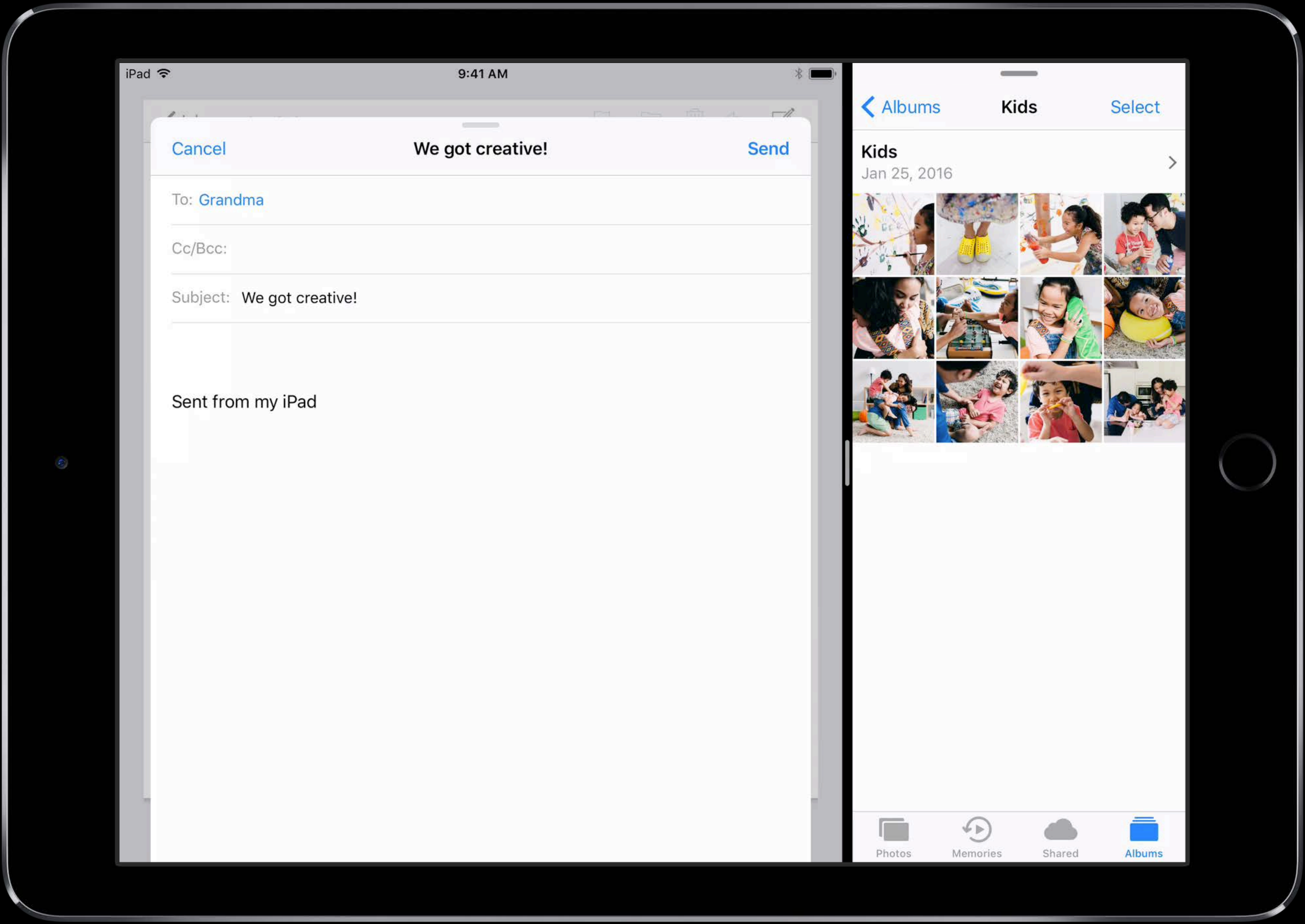
Data loads are always asynchronous

Disconnected timelines

- Data loading
- Animating drop previews

## Dealing with Slow Data Delivery




iPad 9:41 AM

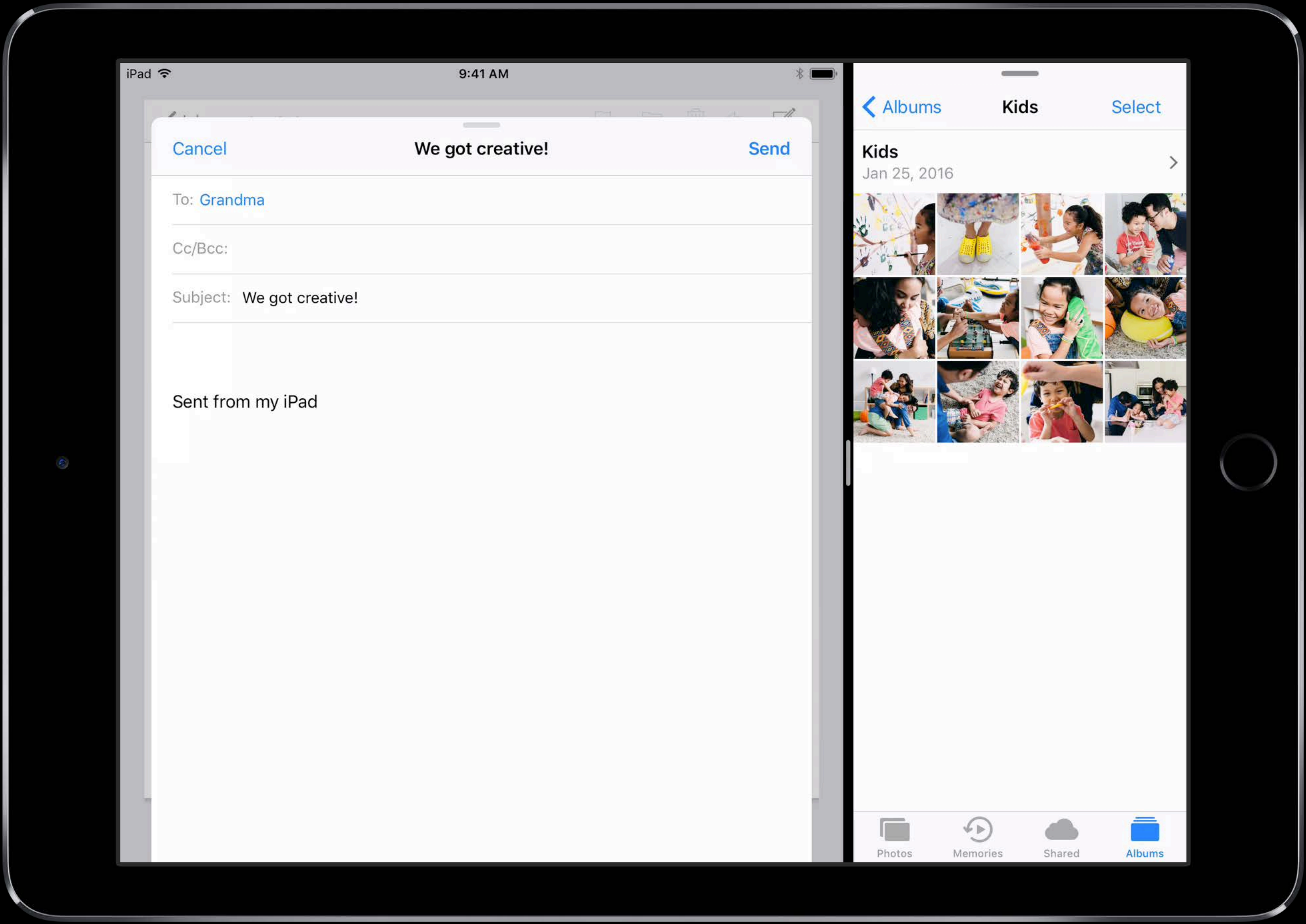
Cancel We got creative! Send

To: Grandma

Cc/Bcc:

Subject: We got creative!

Sent from my iPad


iPad 9:41 AM

Cancel We got creative! Send

To: Grandma

Cc/Bcc:

Subject: We got creative!

Sent from my iPad

## Dealing with Slow Data Delivery

Handling cancelling

Load completion blocks

- Data will be `nil`
- Error will be set

## Dealing with Slow Data Delivery

### Handling cancelling

Session and items provide progress reporting

- Session is `ProgressReporting`
- Item provider load methods return `Progress`
- Use `Progress.cancellationHandler`

# Dealing with Slow Data Delivery

## Handling cancelling

Session and items provide progress reporting

- Session is `ProgressReporting`
- Item provider load methods return `Progress`
- Use `Progress.cancellationHandler`

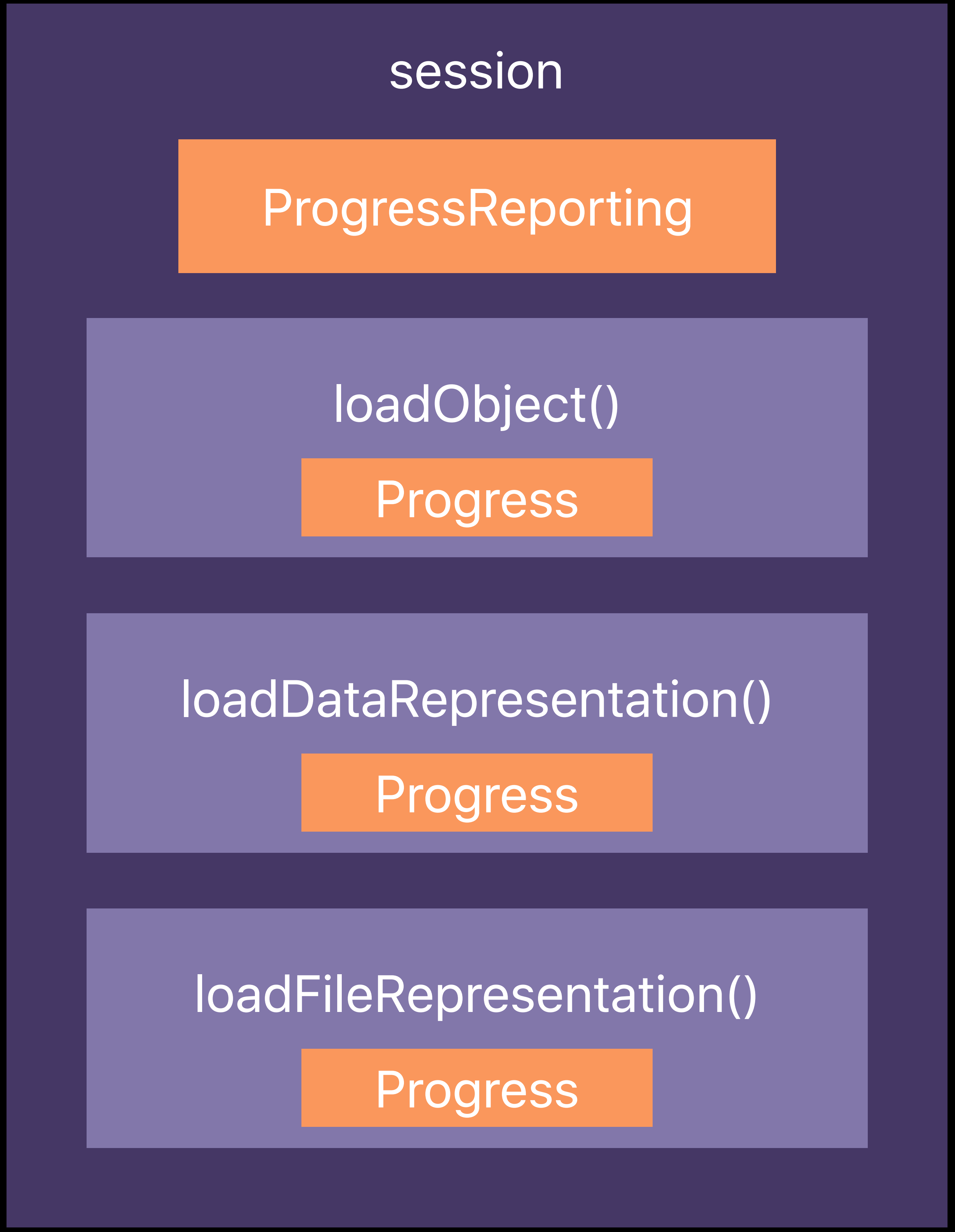
session
ProgressReporting

# Dealing with Slow Data Delivery

## Handling cancelling

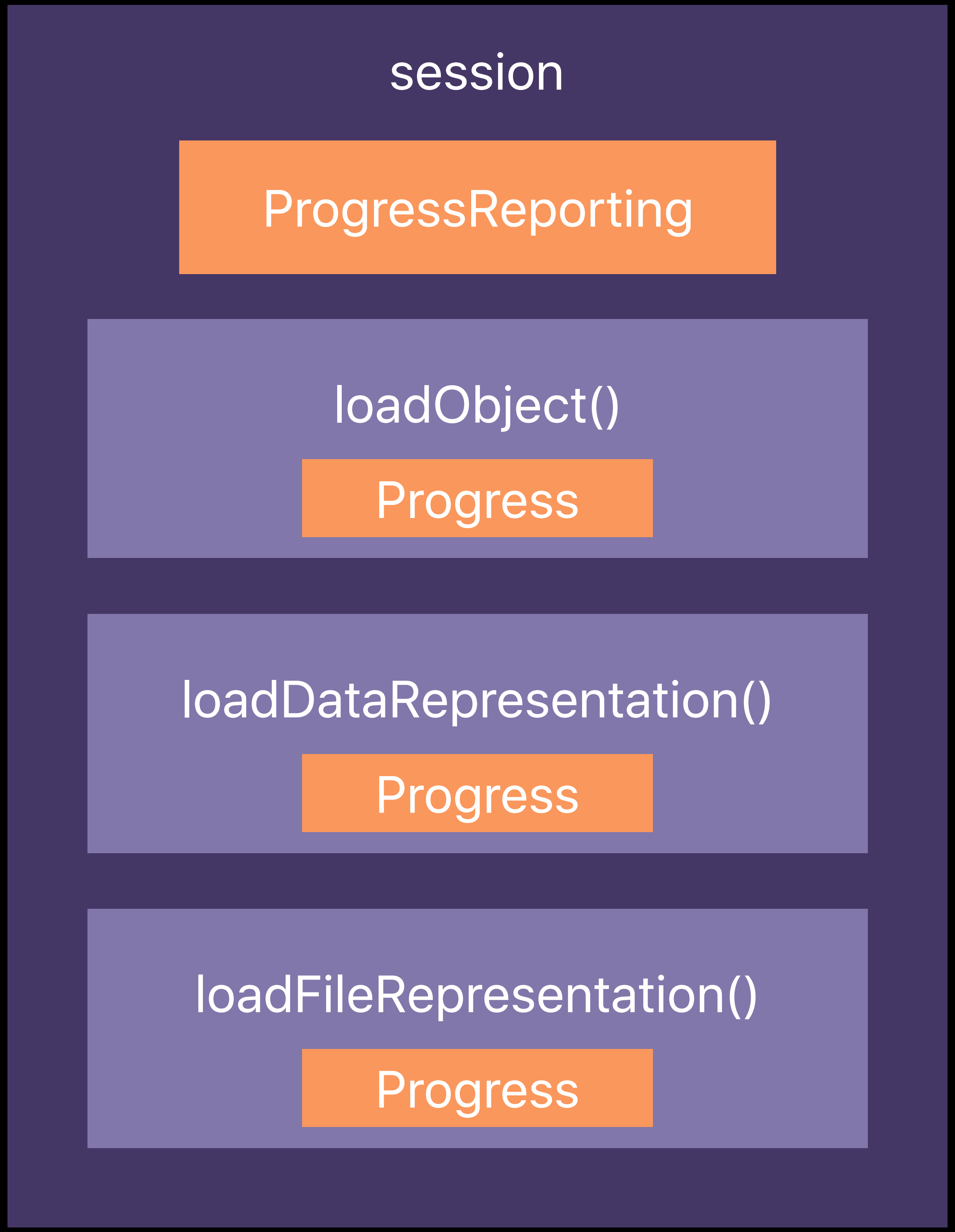
Session and items provide progress reporting

- Session is `ProgressReporting`
- Item provider load methods return `Progress`
- Use `Progress.cancellationHandler`



## Dealing with Slow Data Delivery

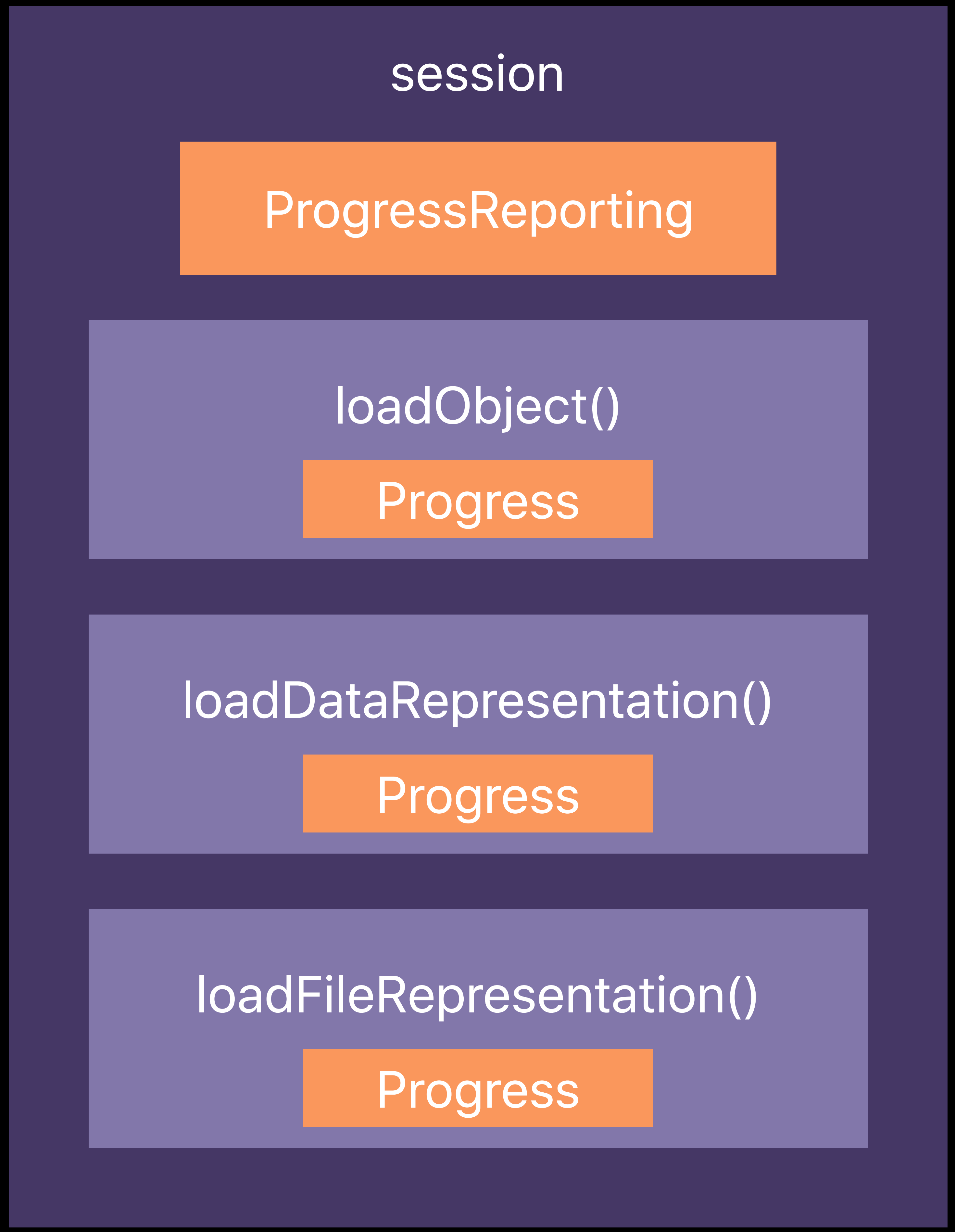
Showing custom progress



## Dealing with Slow Data Delivery

### Showing custom progress

```
session.progressIndicatorStyle = .none
```



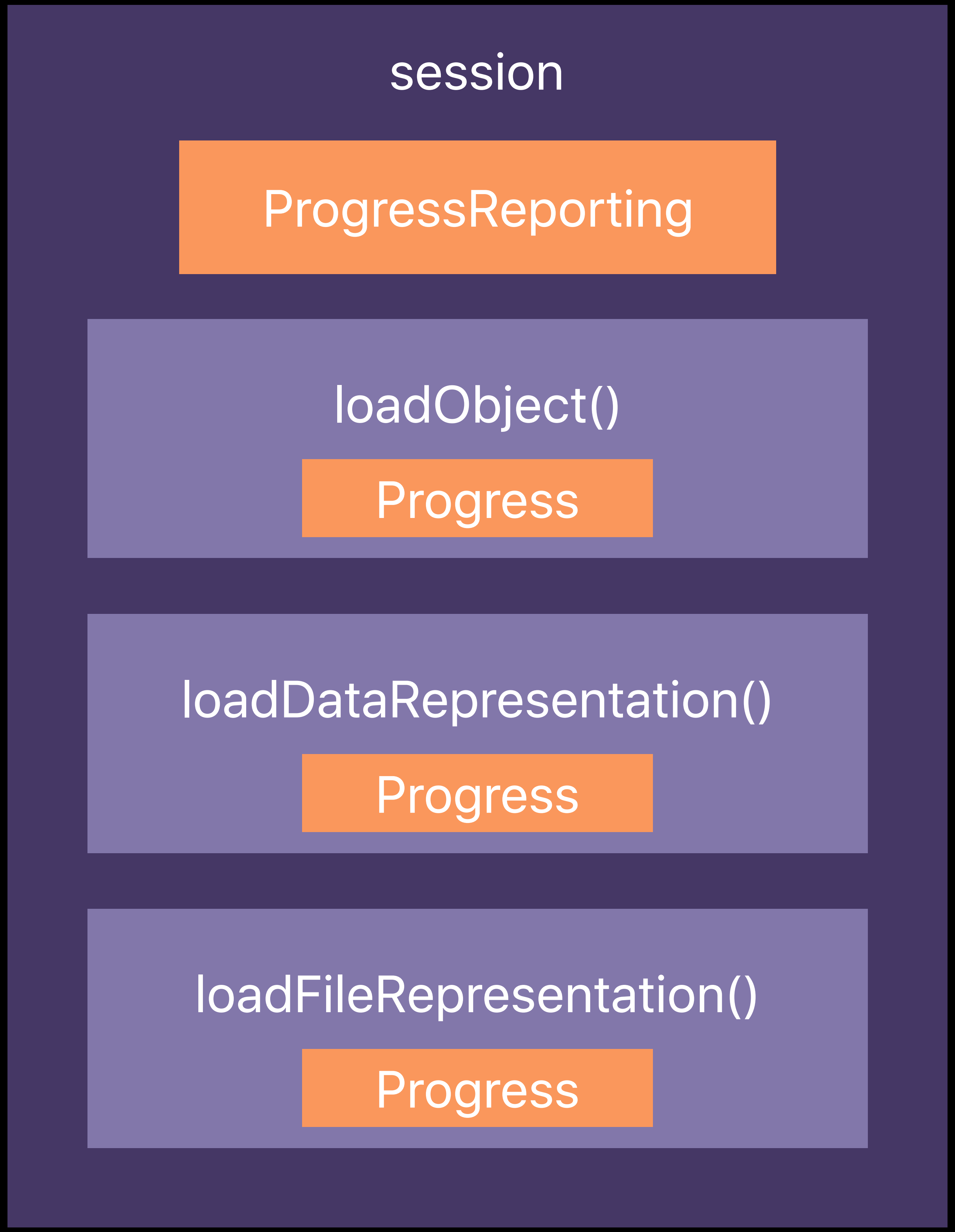
## Dealing with Slow Data Delivery
Showing custom progress

`session.progressIndicatorStyle = .none`

Observe progress

- `session.progress`
- Per item: `Progress` returned by item provider

Allow the user to cancel the transfer



“But—
how can I generate a preview
if I don’t have any data?”

## Dealing with Slow Data Delivery

Previews without data

# Dealing with Slow Data Delivery

Previews without data

Use and retarget the default previews

## Dealing with Slow Data Delivery
Previews without data

Use and retarget the default previews

Make a placeholder/progress preview

## Dealing with Slow Data Delivery

Previews without data

Use and retarget the default previews

Make a placeholder/progress preview

Never assume the data will be there

## Dealing with Slow Data Delivery

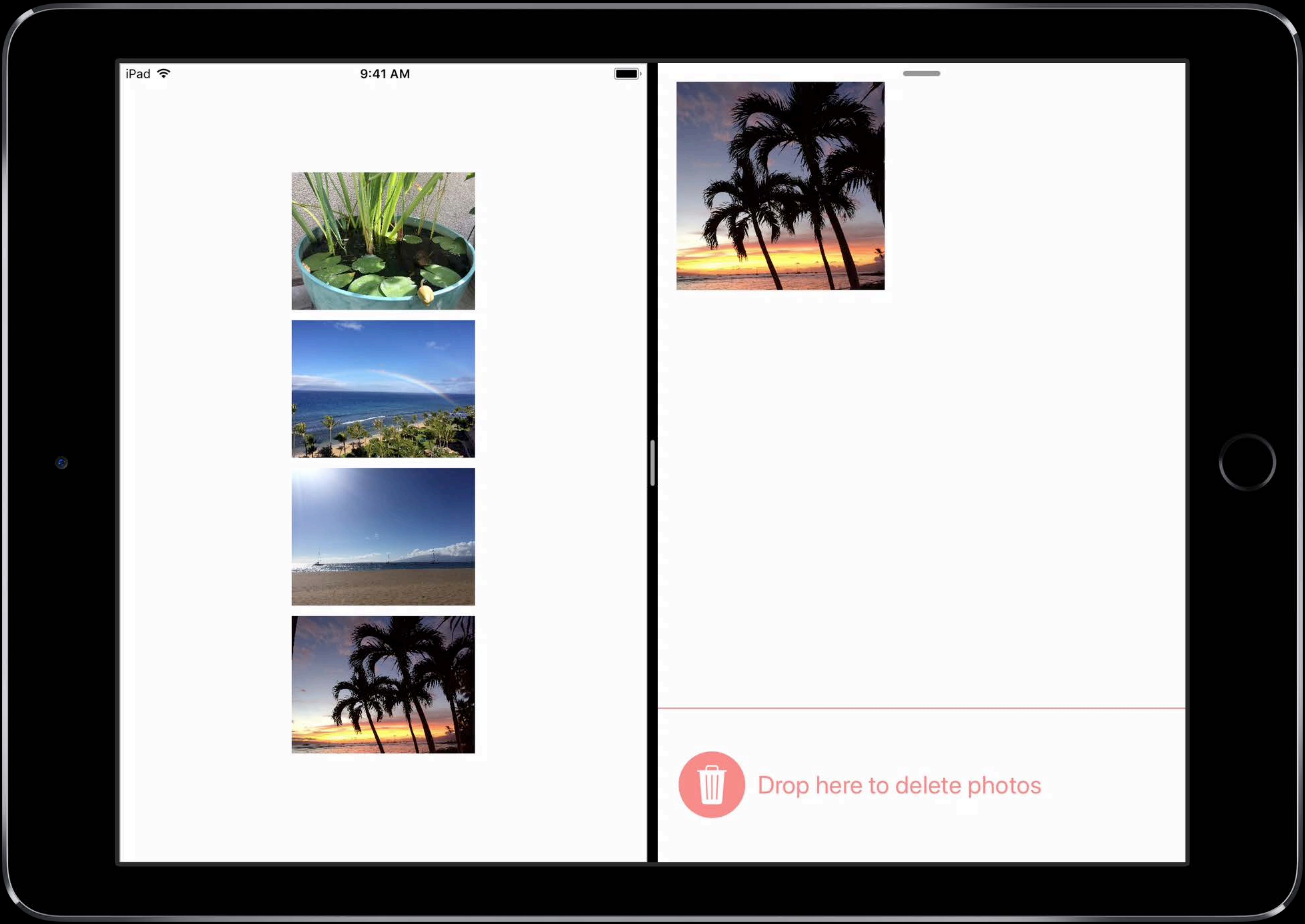
Previews without data

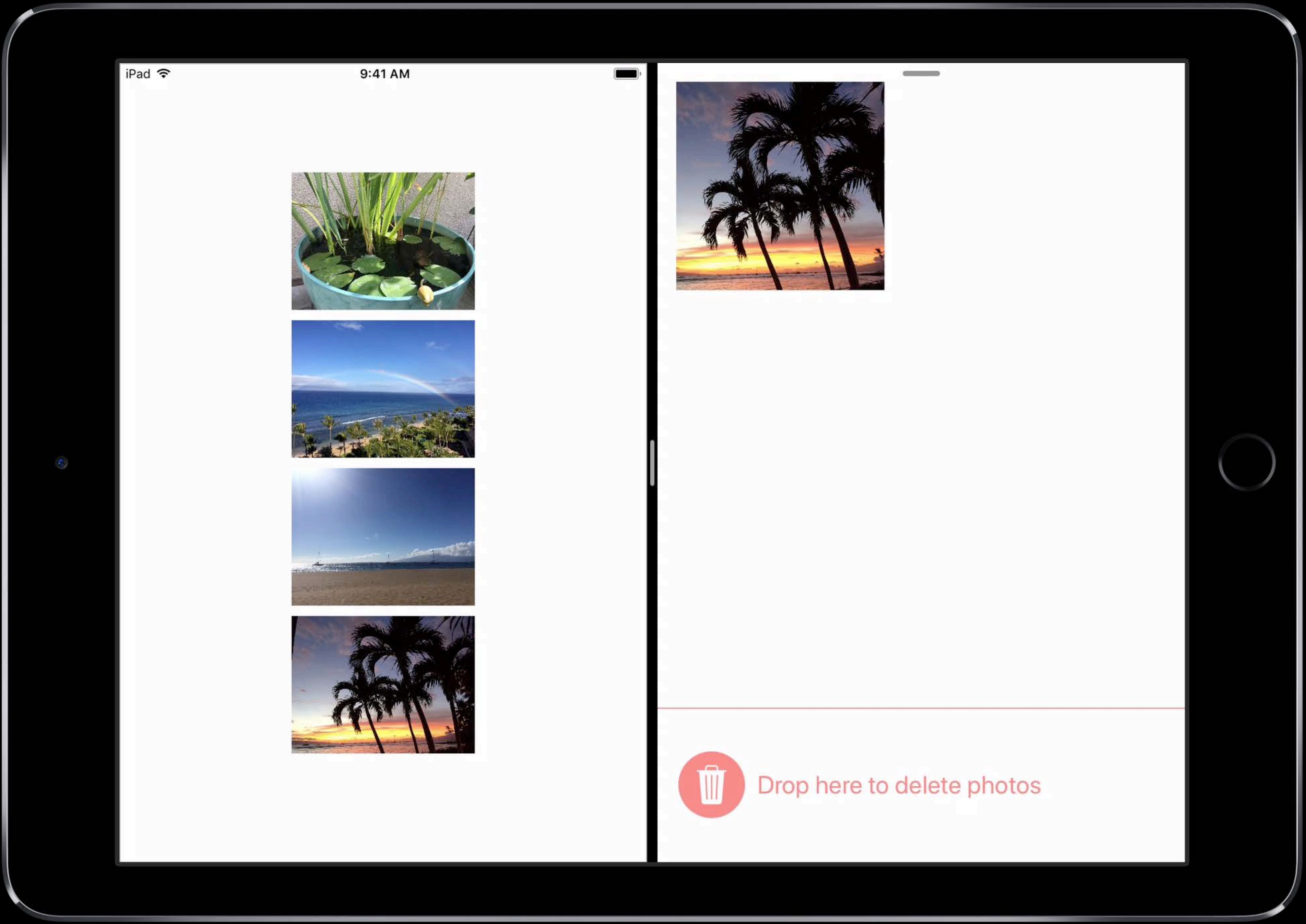
Use and retarget the default previews

Make a placeholder/progress preview

Never assume the data will be there

Always account for the worst case


iPad
9:41 AM
Drop here to delete photos


iPad
9:41 AM
Drop here to delete photos

## Improved In-App Experience by Adding Drag and Drop

## Improved In-App Experience

Local drag and drop

```
protocol UIDropSession : UIDragDropSession, ProgressReporting {

    var localDragSession: UIDragSession? { get }

}
```

## Improved In-App Experience

Local drag and drop

```
protocol UIDropSession : UIDragDropSession, ProgressReporting {

    var localDragSession: UIDragSession? { get }

}
```

Access to corresponding drag session

`nil` for cross-app drags

## Improved In-App Experience

Local drag and drop

```
class UIDragItem : NSObject {

   var localObject: Any?

   var itemProvider: NSItemProvider { get }
}
```

## Improved In-App Experience

Local drag and drop

```
class UIDragItem : NSObject {

    var localObject: Any?

    var itemProvider: NSItemProvider { get }
}
```

App-local data carrier

- State for drag
- Transfer data between views

Provider fallback item provider (if drag can go outside app)

## Improved In-App Experience

Local drag and drop

```
protocol UIDragSession : UIDragDropSession {

    var localContext: Any? { get set }

}
```

## Improved In-App Experience

Local drag and drop

```
protocol UIDragSession : UIDragDropSession {

    var localContext: Any? { get set }

}
```

Usable for state that applies to this drag session only

## Improved In-App Experience

Local drag and drop

```
optional func dragInteraction(_ interaction: UIDragInteraction,
   sessionIsRestrictedToDraggingApplication session: UIDragSession) -> Bool // default: false

var isRestrictedToDraggingApplication: Bool { get }
```

Restrict drag sessions to local application

- Drop interactions in other apps won’t see the session

## Improved In-App Experience

Local drag and drop

```
optional func dragInteraction(_ interaction: UIDragInteraction,
   sessionIsRestrictedToDraggingApplication session: UIDragSession) -> Bool // default: false

var isRestrictedToDraggingApplication: Bool { get }
```

Restrict drag sessions to local application

- Drop interactions in other apps won’t see the session

## Improved In-App Experience

### Local drag and drop

```
optional func dragInteraction(_ interaction: UIDragInteraction,
   sessionIsRestrictedToDraggingApplication session: UIDragSession) -> Bool // default: false

var isRestrictedToDraggingApplication: Bool { get }
```

Restrict drag sessions to local application

- Drop interactions in other apps won’t see the session

## Improved In-App Experience

### Local drag and drop

```
optional func dragInteraction(_ interaction: UIDragInteraction,
   sessionIsRestrictedToDraggingApplication session: UIDragSession) -> Bool // default: false

var isRestrictedToDraggingApplication: Bool { get }
```

Restrict drag sessions to local application

- Drop interactions in other apps won’t see the session

## Improved In-App Experience

Local drag and drop

## Improved In-App Experience

Local drag and drop

By default drag interactions are

- Enabled on iPad
- Disabled on iPhone

## Improved In-App Experience

Local drag and drop

By default drag interactions are

- Enabled on iPad
- Disabled on iPhone

```
var isEnabled: Bool


class var isEnabledByDefault: Bool { get }
```

## *Demo*

Wenson Hsieh, WebKit

## Summary

Powerful, user-driven I/O for your app

Custom and stunning visuals and interactions

Asynchronous nature of data

Leverage Drag and Drop to improve in-app experience

## More Information

https://developer.apple.com/wwdc17/213

## Related Sessions

| | | |
|---|---|---|
| Introducing Drag and Drop | Hall 3 | Tuesday 11:20AM |
| Drag and Drop with Collection and Table View | Hall 2 | Thursday 9:00AM |
| Data Delivery with Drag and Drop | Hall 2 | Thursday 10:00AM |

# Labs

| | | |
|---|---|---|
| UIKit and Drag and Drop Lab | Technology Lab C | Tues 1:50PM-3:10PM |
| Cocoa Touch Lab | Technology Lab I | Wed 3:10-6:00PM |
| UIKit and Collection View Lab | Technology Lab B | Thur 10:00-12:30PM |
| Cocoa Touch and Haptics Lab | Technology Lab C | Fri 12:00-1:50PM |

🎤 💧

WWDC17