Introducing Drag and Drop

Session 203

Bruce Nilo, UIKit Engineering Manager
Kurt Revis, UIKit Engineer
Emanuele Rudel, UIKit Engineer

Goals and concepts

API fundamentals

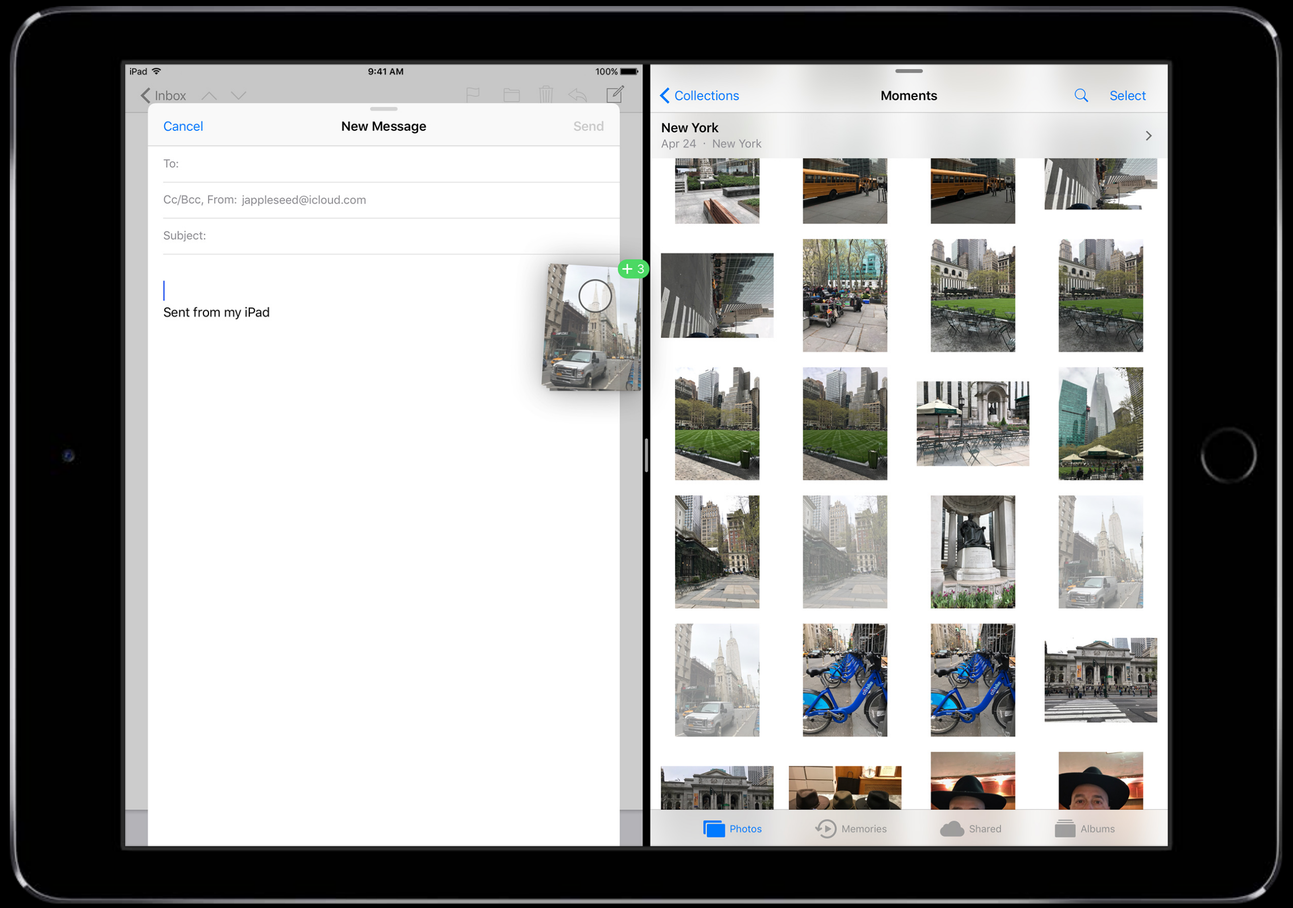
Showing the APIs in action

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


9:41 AM
100%
FaceTime
Sunday
28
Calendar
Photos
Camera
Contacts
Home
News
App Store
iTunes Store
iBooks
Settings
Tips
15,132
280
Sunday
28


9:41 AM
100%
FaceTime
Sunday
28
Calendar
Photos
Camera
Contacts
Home
News
App Store
iTunes Store
iBooks
Settings
Tips
15,132
280
Sunday
28

Drag and Drop on iOS

Goals

Drag and Drop on iOS

Goals

A great Multi-Touch experience

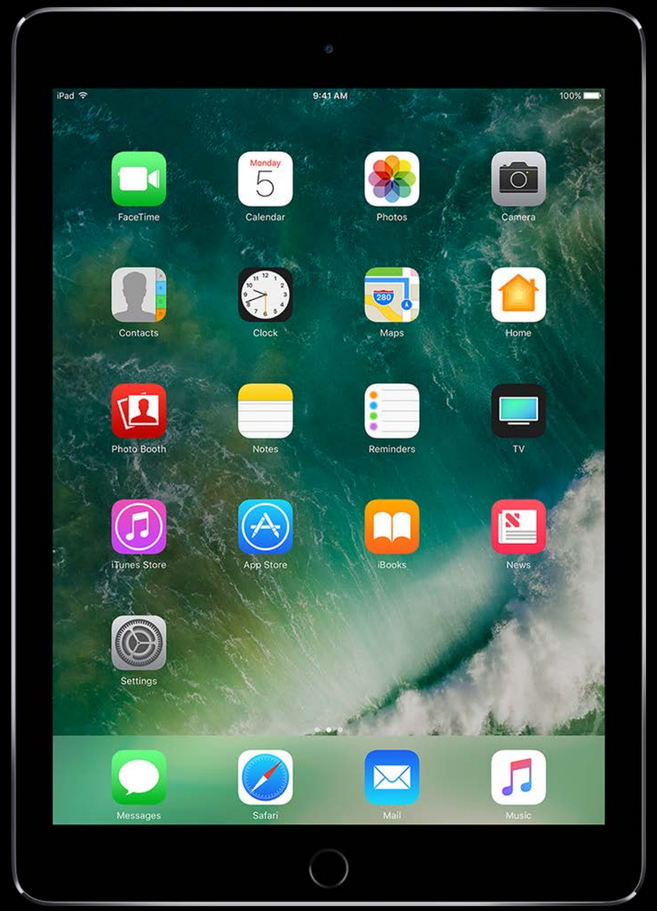
Drag and Drop on iOS

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

Lift 

Drag 

Set Down 

Data Transfer 

Drag and Drop on iOS

Phases of a drag session

Lift 

Drag 

Set Down 

Data Transfer 

Long press

Lift preview

Drag and Drop on iOS

Phases of a drag session

Lift 	Drag 	Set Down 	Data Transfer 
Long press	Previews		
Lift preview	Tap to add		
	Spring-loading		

Drag and Drop on iOS

Phases of a drag session

Lift 	Drag 	Set Down 	Data Transfer 
Long press	Previews	Cancel	
Lift preview	Tap to add	Drop	
	Spring-loading	Targeting	

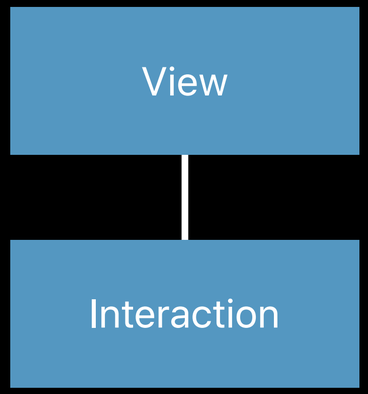
Drag and Drop on iOS

Phases of a drag session

Lift 	Drag 	Set Down 	Data Transfer 
Long press	Previews	Cancel	Representations
Lift preview	Tap to add	Drop	Lazy delivery • Background • By File Provider
	Spring-loading	Targeting	Progress

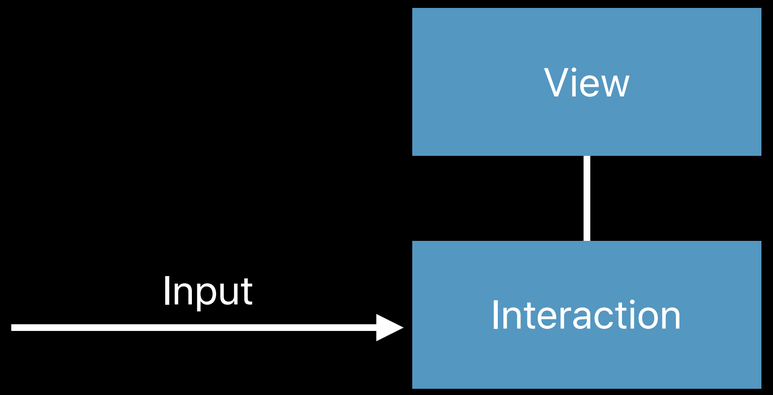
Enabling a Drag

Concepts - interactions



Enabling a Drag

Concepts - interactions



Enabling a Drag

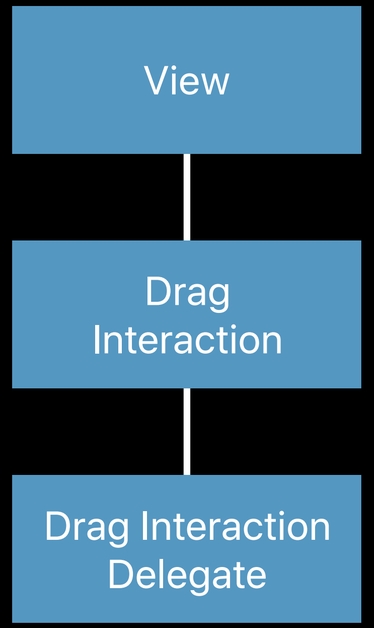
Concepts - UIDragInteraction

A drag interaction is attached to a view

Enabling a Drag

Concepts - UIDragInteraction

A drag interaction is attached to a view

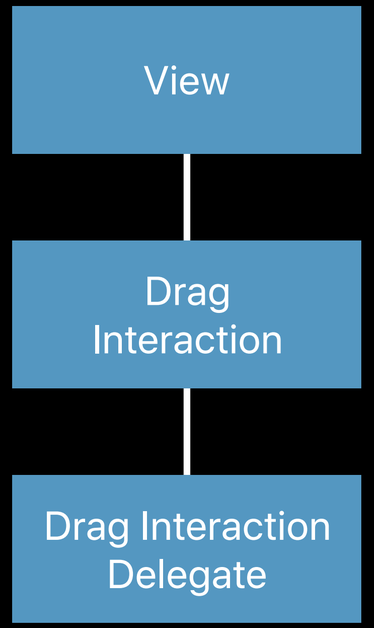


Enabling a Drag

Concepts - UIDragInteraction

A drag interaction is attached to a view

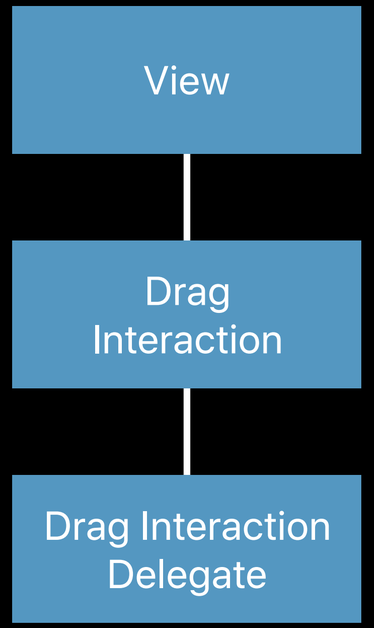
```
import UIKit

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

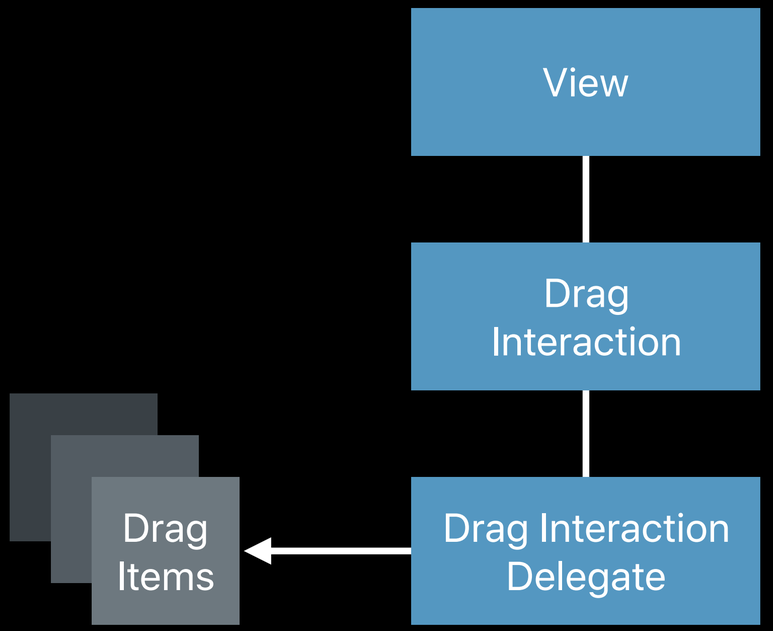


Lift Phase

Concepts - UIDragInteraction

The delegate provides drag items when the view lifts

No drag items -> drag gesture fails



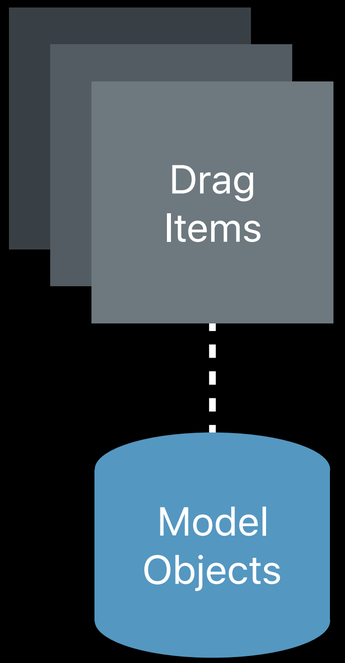
Drag Items

Concepts - UIDragItem

Drag Items

Concepts - UIDragItem

A drag item represents a model object



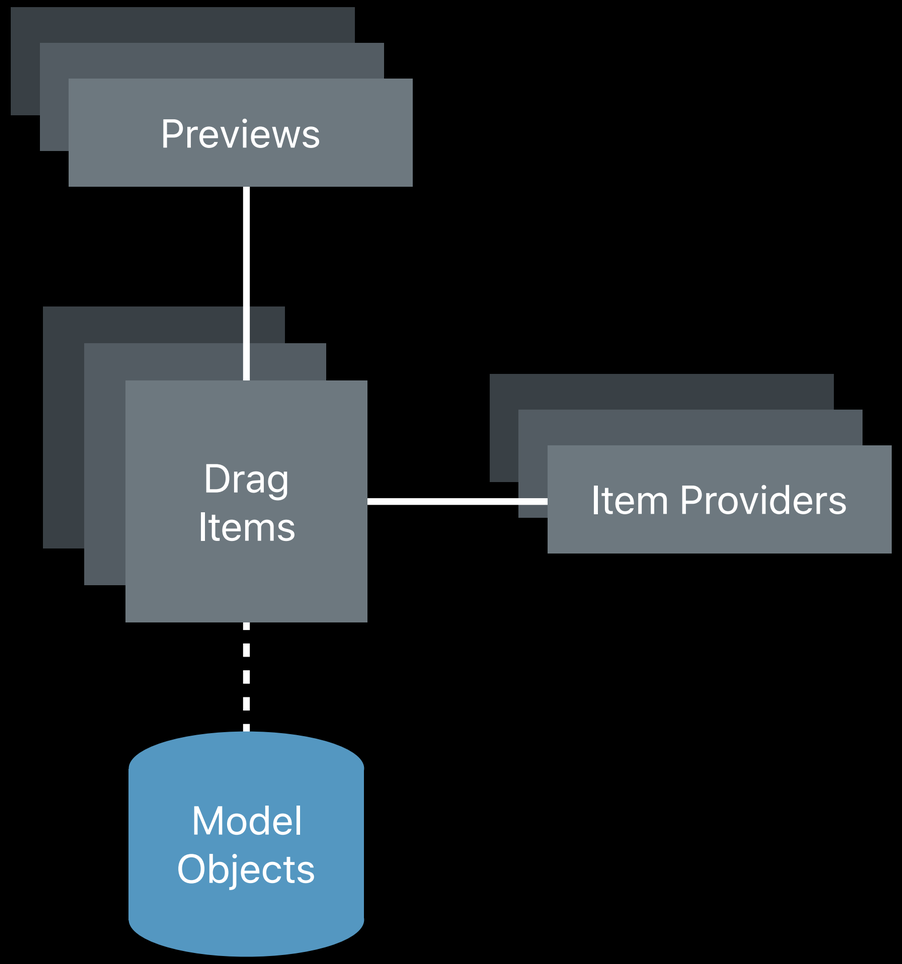
Drag Items

Concepts - UIDragItem

A drag item represents a model object

A drag item embodies

- Drag preview
- Item provider



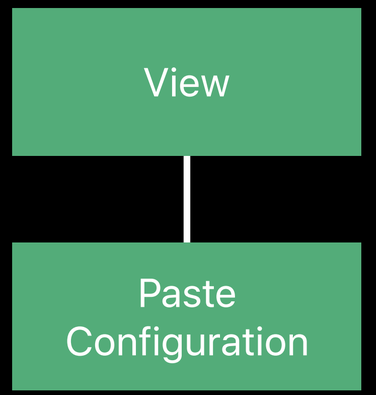
Enabling a Drop

Concepts - UIPasteConfiguration

Enabling a Drop

Concepts - UIPasteConfiguration

UIResponders have a new paste configuration property

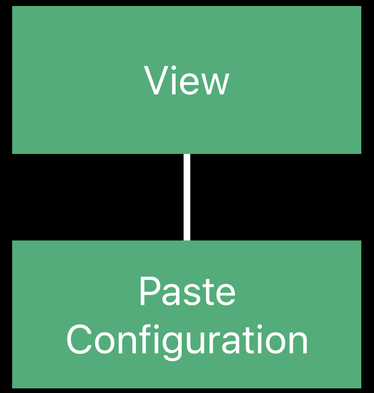


Enabling a Drop

Concepts - UIPasteConfiguration

UIResponders have a new paste configuration property

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

let config = UIPasteConfiguration(typeIdentifiersForAcceptingClass:
                                  NSString.self)
view.pasteConfiguration = config
```

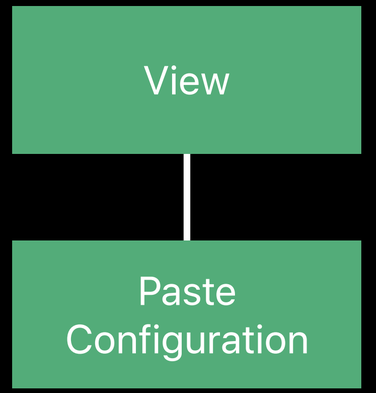


Enabling a Drop

Concepts - UIPasteConfiguration

UIResponders have a new paste configuration property

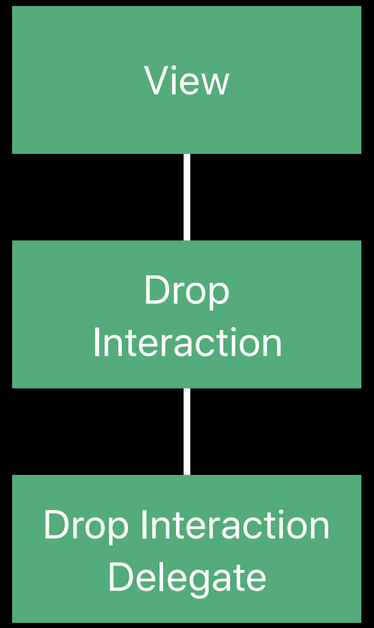
```
// Will be called for both Drag and Drop, and Copy/Paste

override func paste(itemProviders: [NSItemProvider]) {
}
```



Enabling a Drop

Concepts - UIDropInteraction

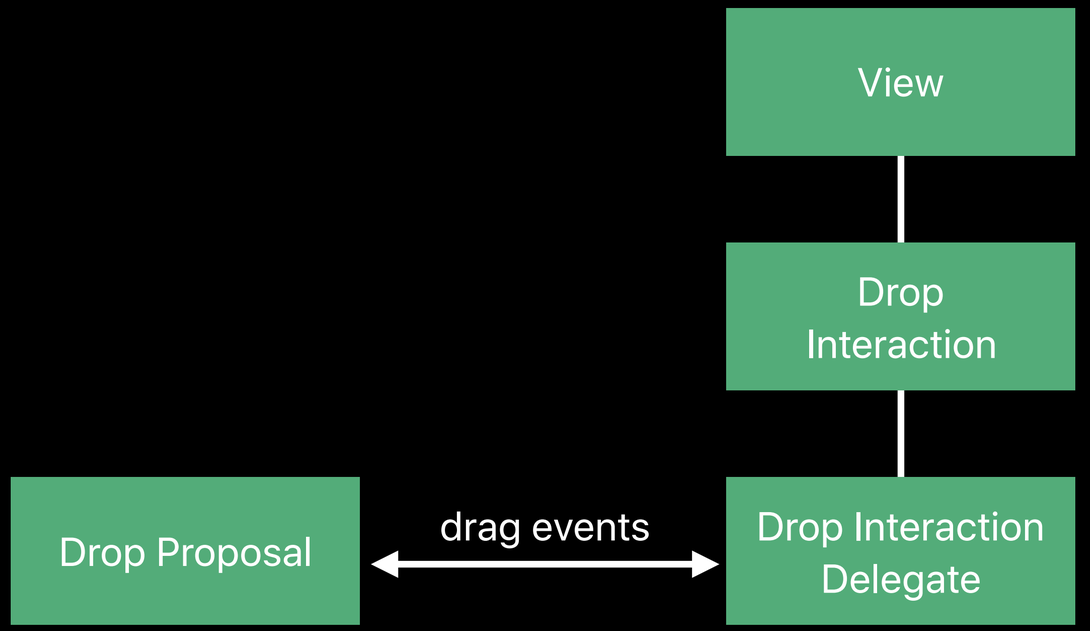
A drop interaction is attached to a view



Drag Phase

Concepts - UIDropInteraction

The delegate responds to drag events

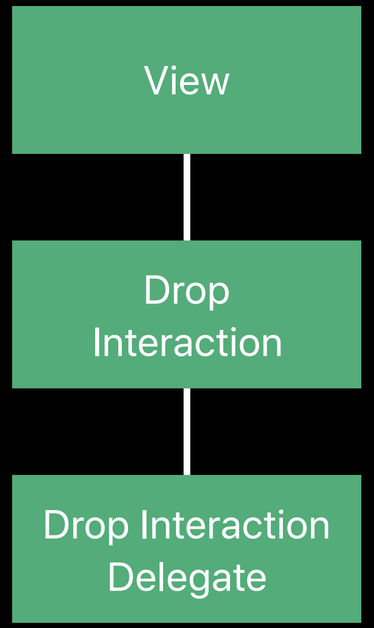


Set Down Phase

Concepts - UIDropInteraction

On touch up, the drag session may be cancelled

- The drag preview animates back

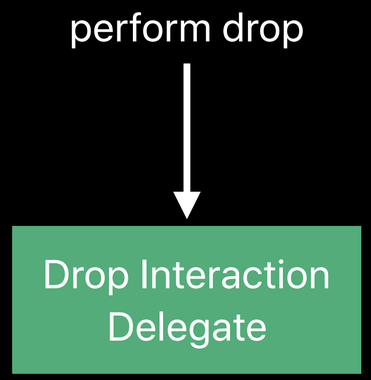


Set Down Phase

Concepts - UIDropInteraction

Or the drop is accepted

- The delegate is told to perform drop

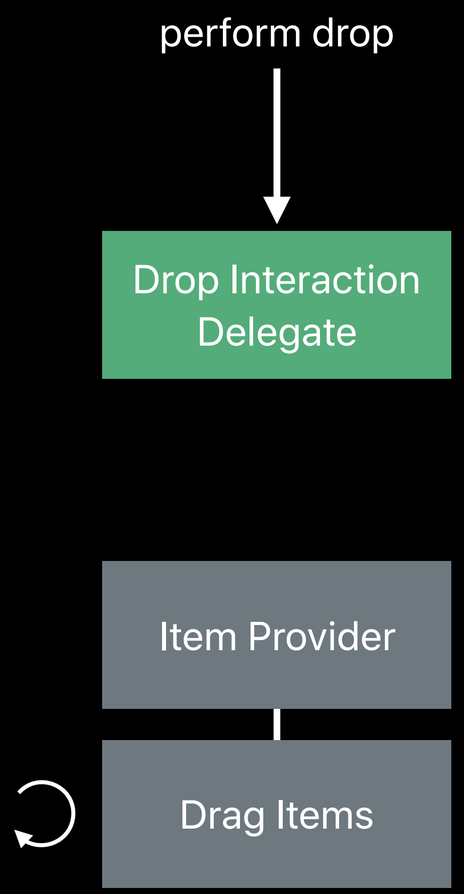


Data Transfer Phase

Concepts - UIDropInteraction

Or the drop is accepted

- The delegate is told to perform drop



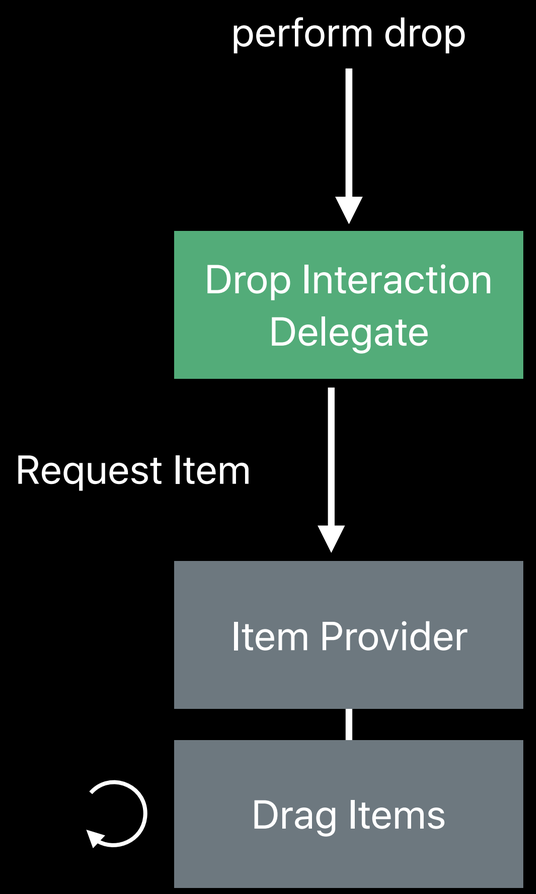
Data Transfer Phase

Concepts - UIDropInteraction

Or the drop is accepted

- The delegate is told to perform drop

Delegate requests data representation of items



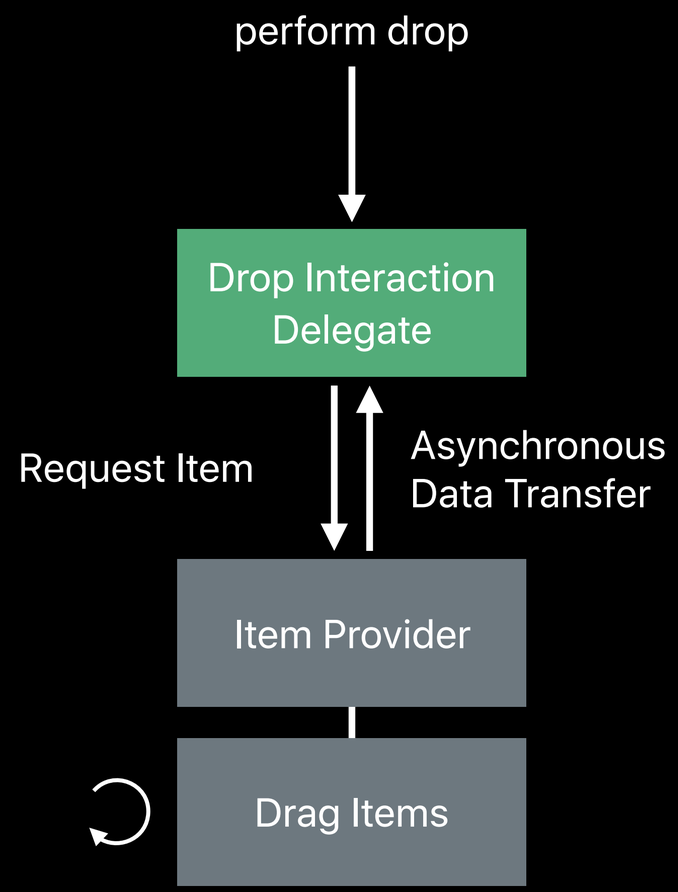
Data Transfer Phase

Concepts - UIDropInteraction

Or the drop is accepted

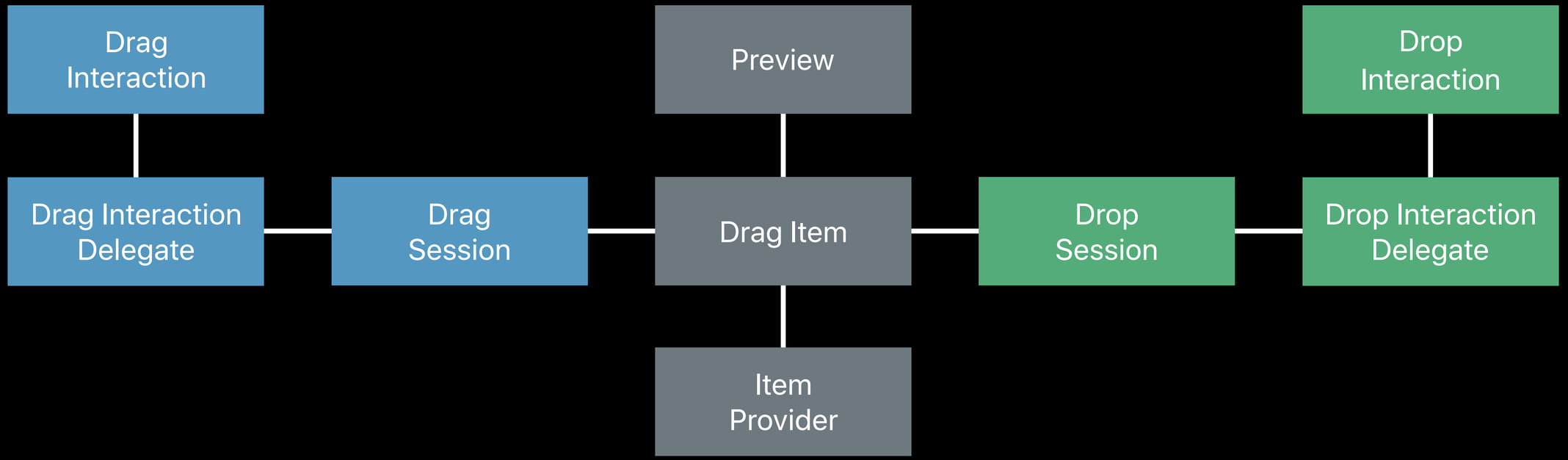
- The delegate is told to perform drop

Delegate requests data representation of items



API Roadmap

API Roadmap


Drag Interaction
Drag Interaction Delegate
Drag Session
Preview
Drag Item
Item Provider
Drop Session
Drop Interaction
Drop Interaction Delegate

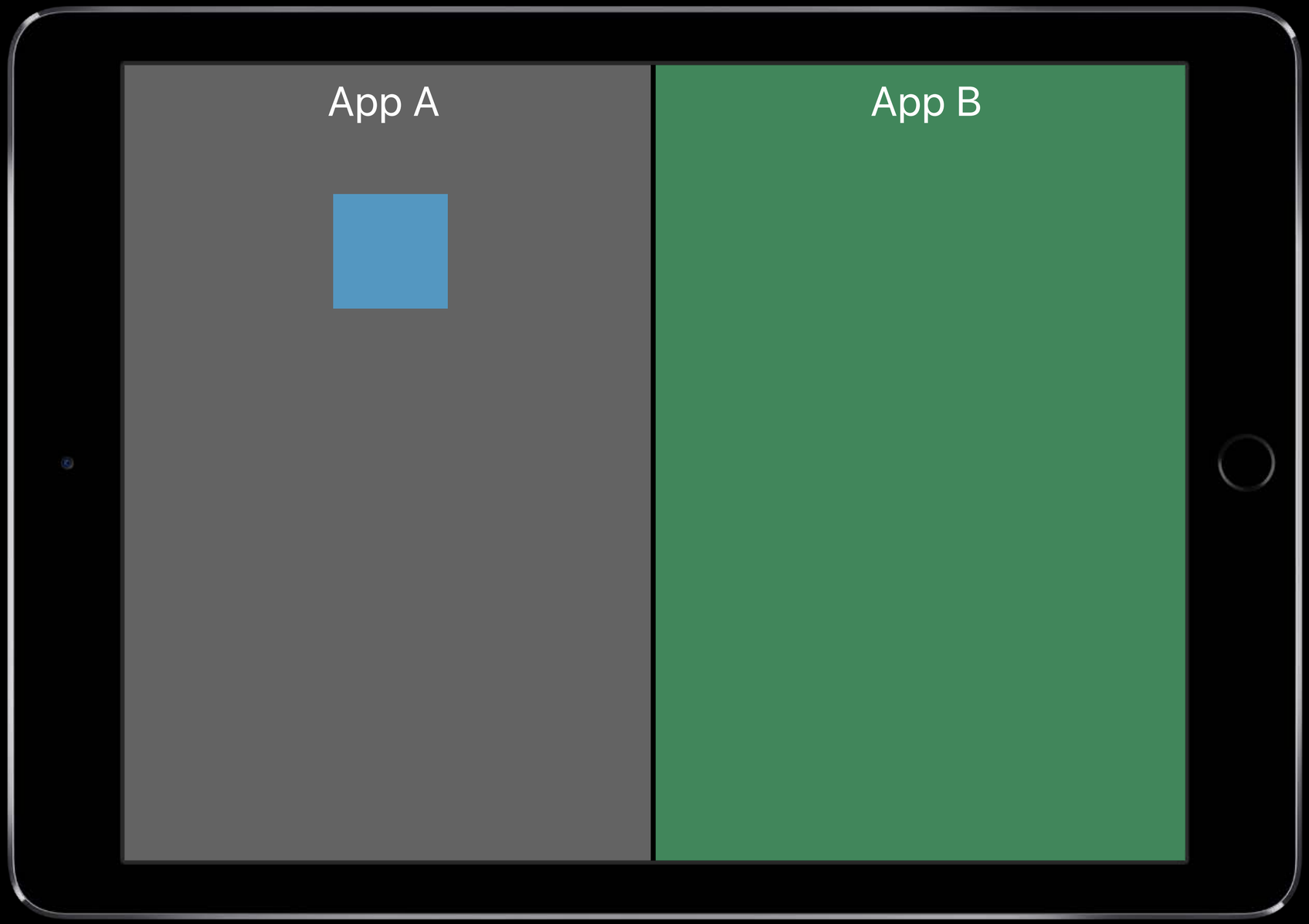
Using the Drag and Drop API

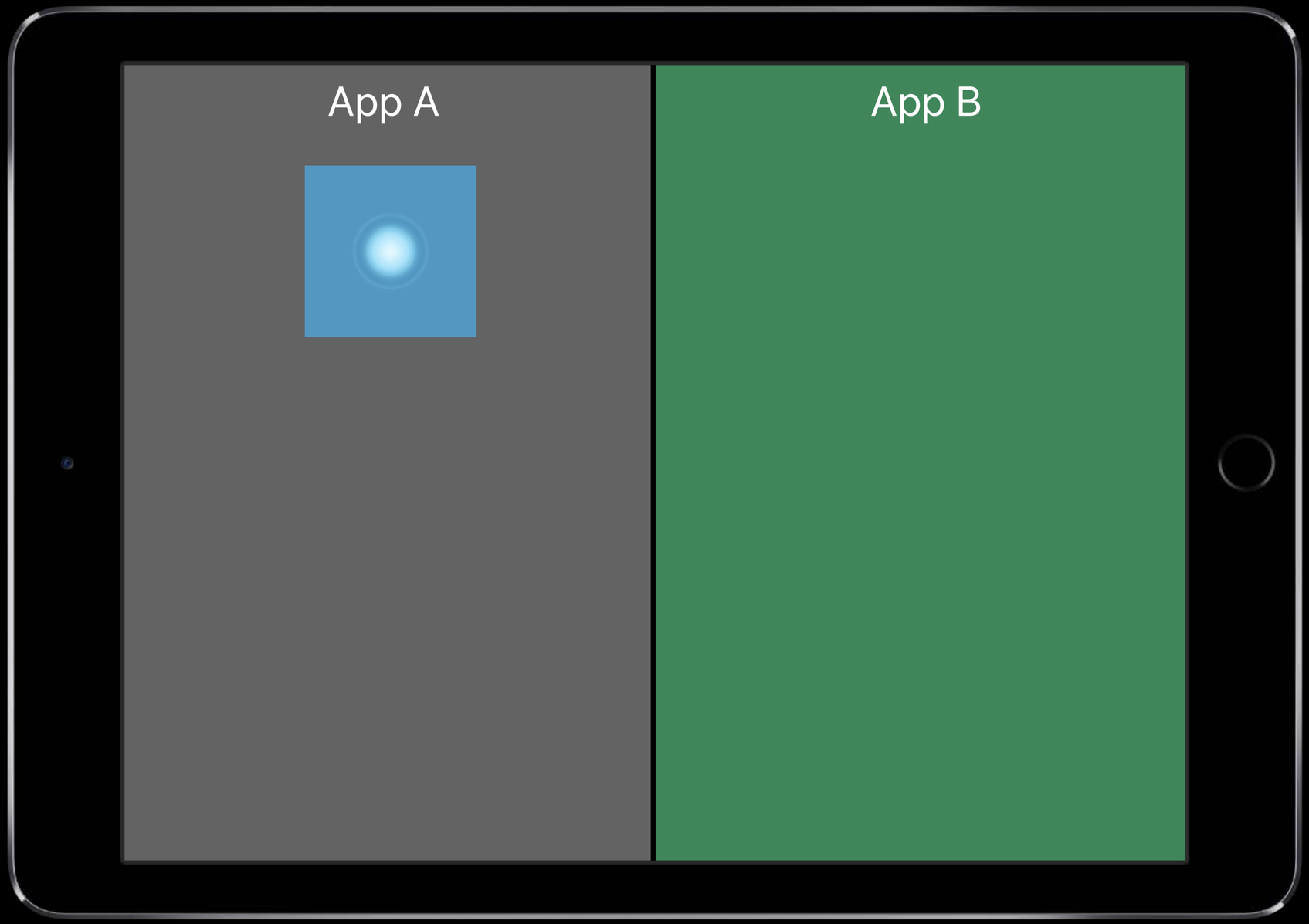
Kurt Revis, UIKit Engineer

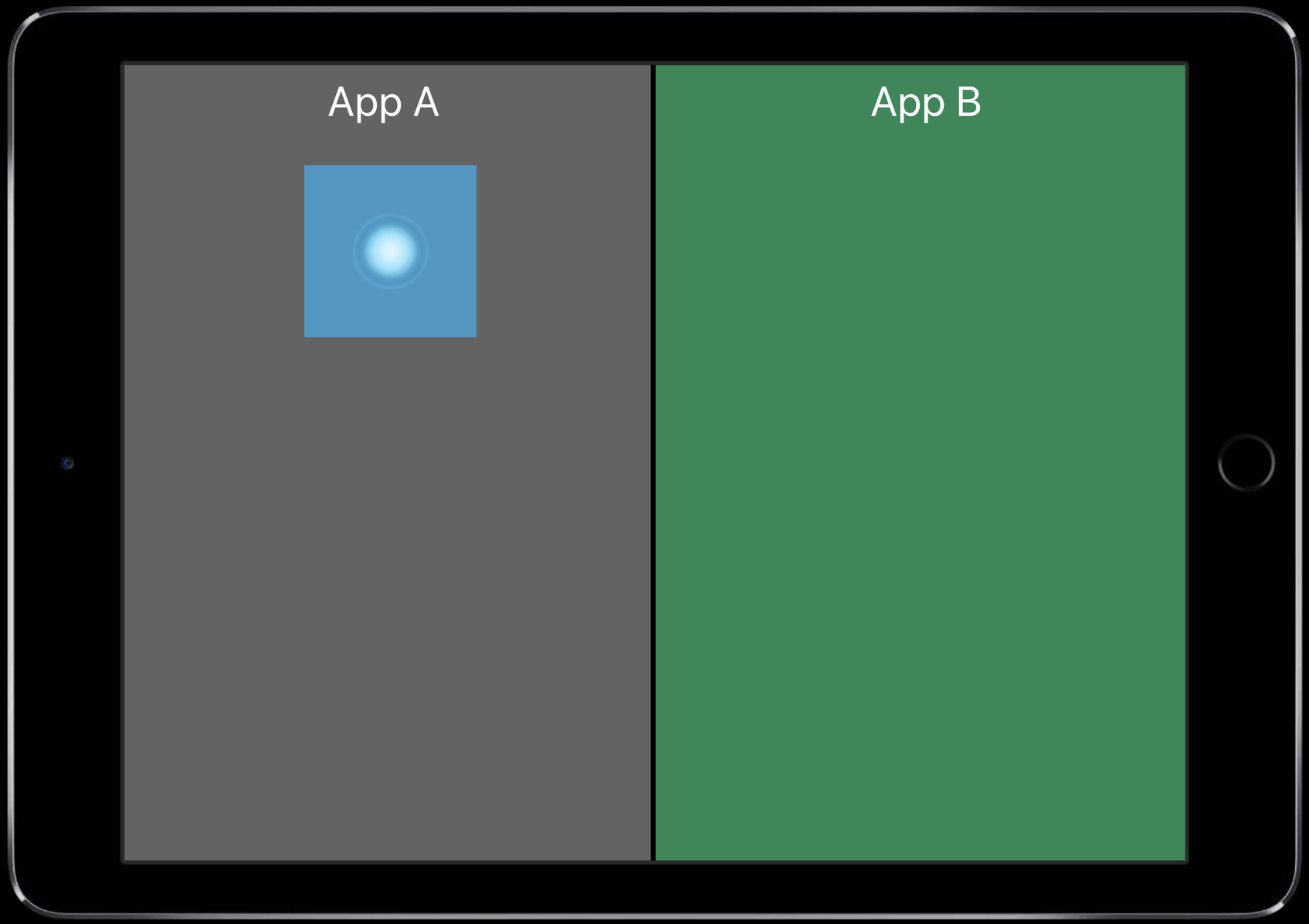
Drag and drop timeline

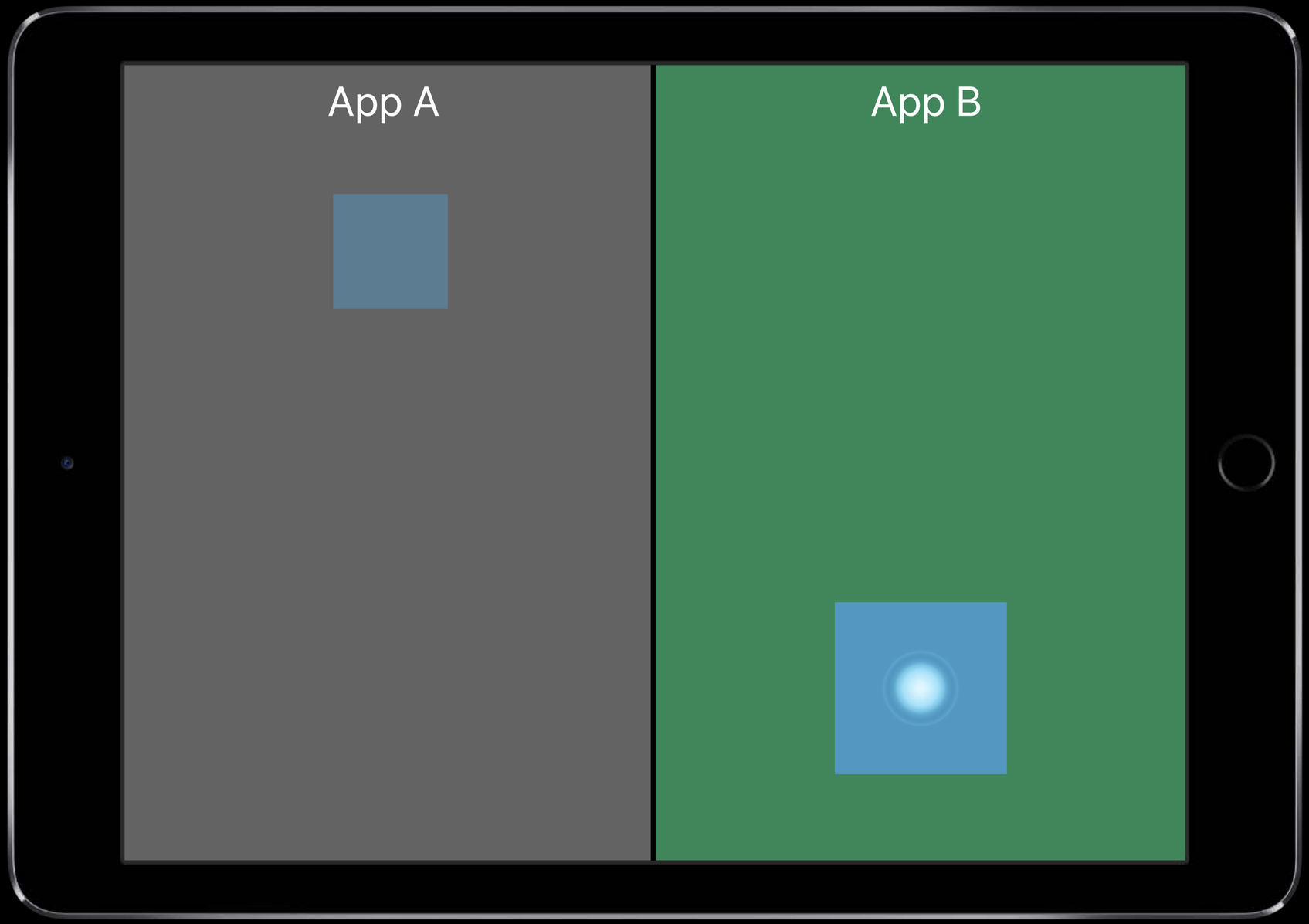
API essentials

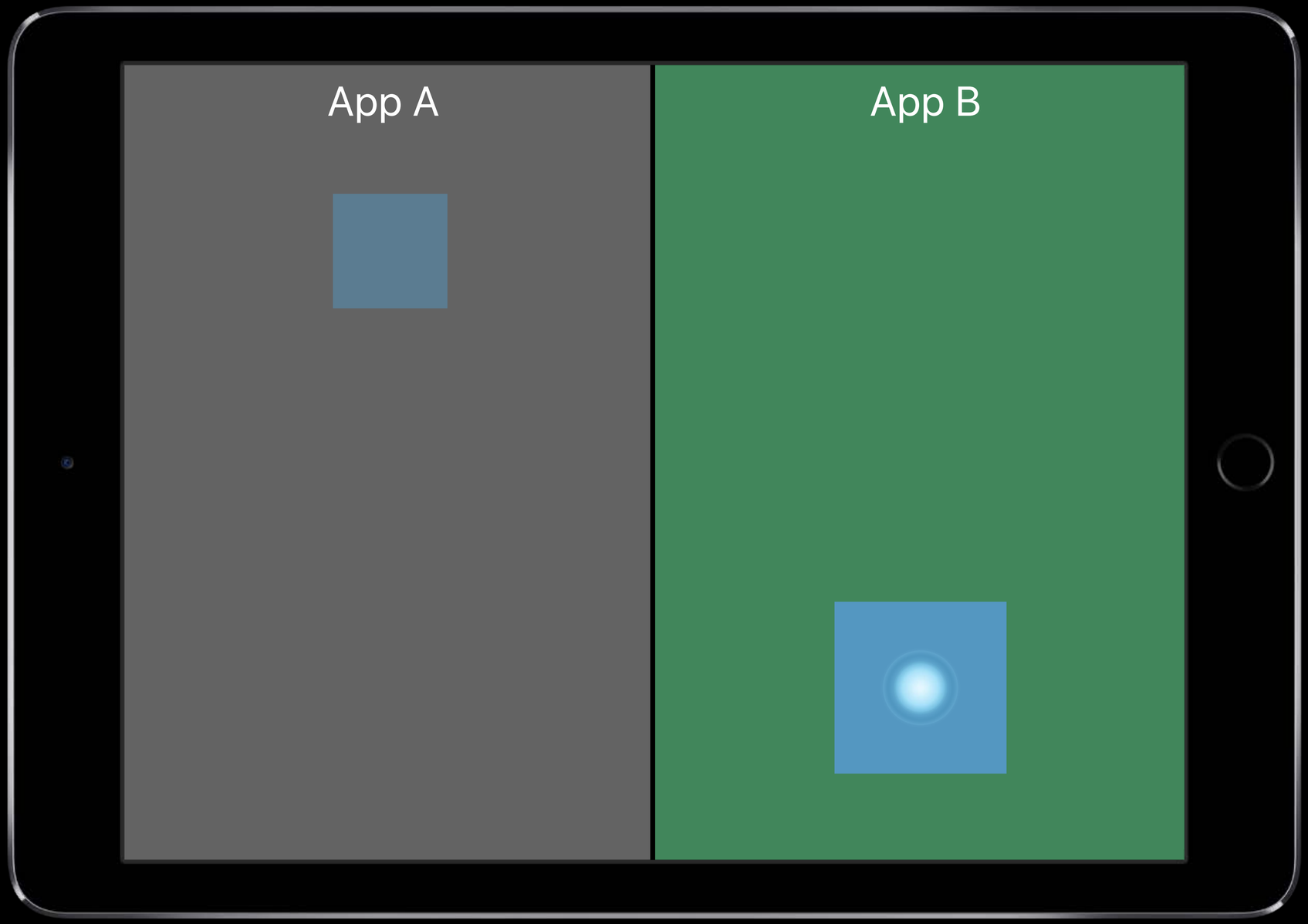
Introduction to the full API

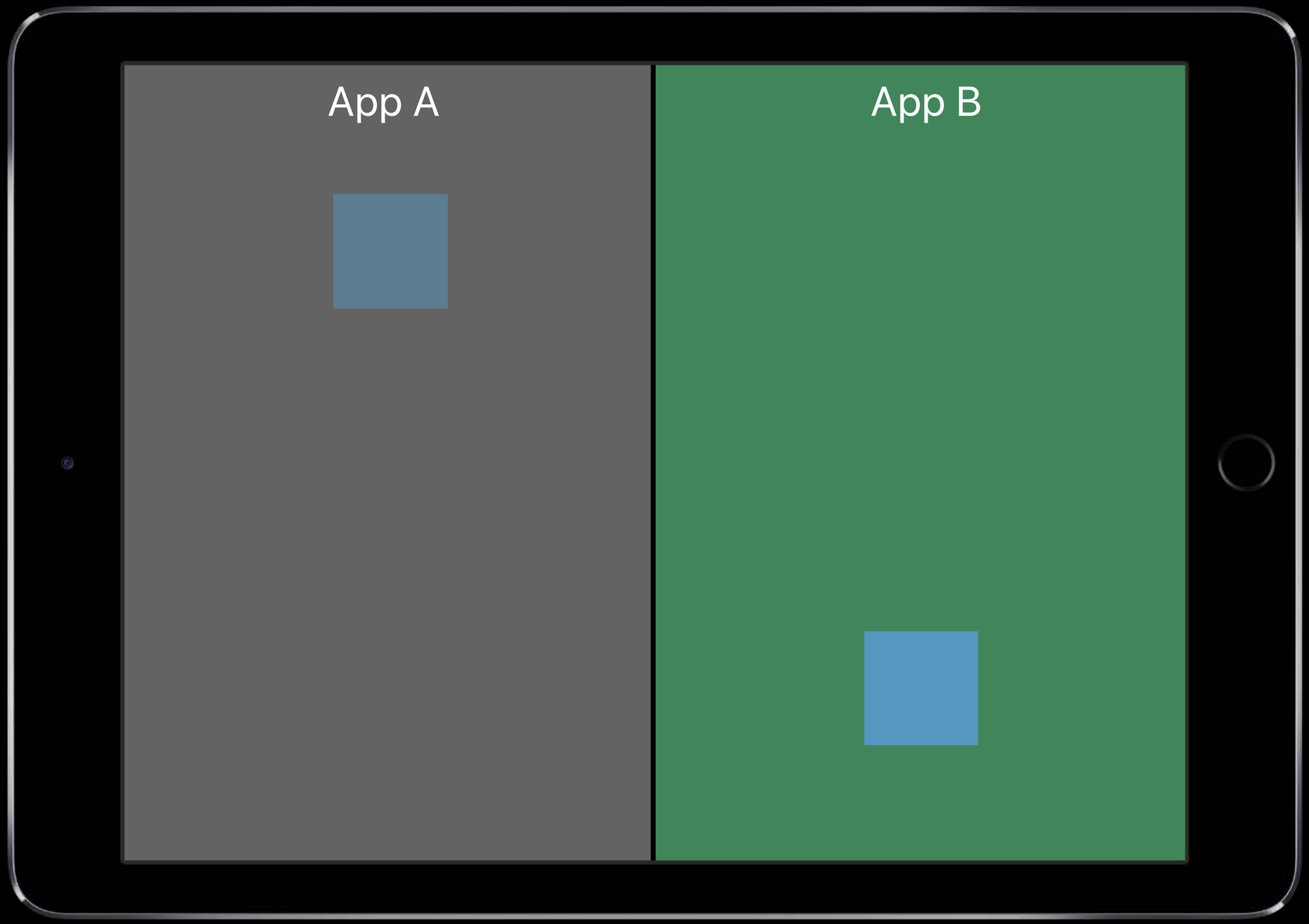

App A
App B


App A
App B


App A
App B


App A
App B


App A
App B


App A
App B

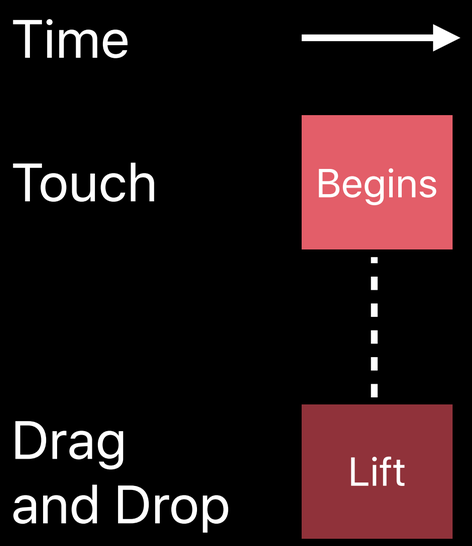
Drag and Drop Timeline

Drag and Drop Timeline

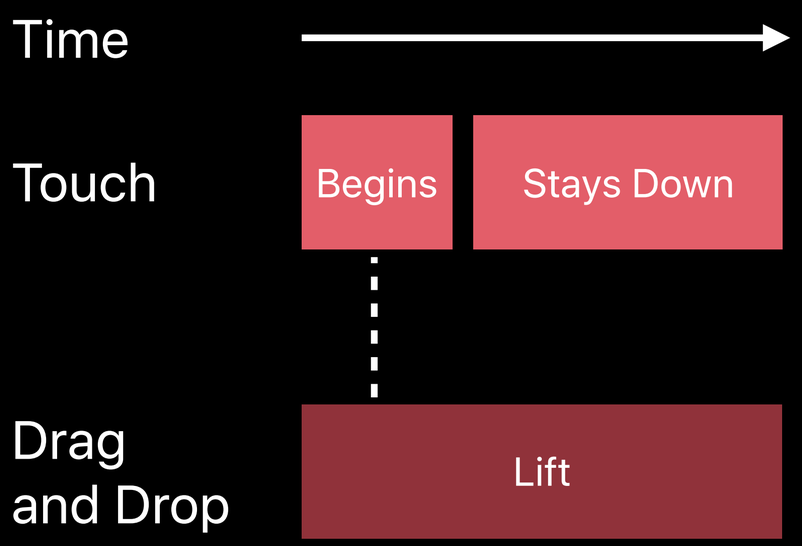
Time 

Touch Begins

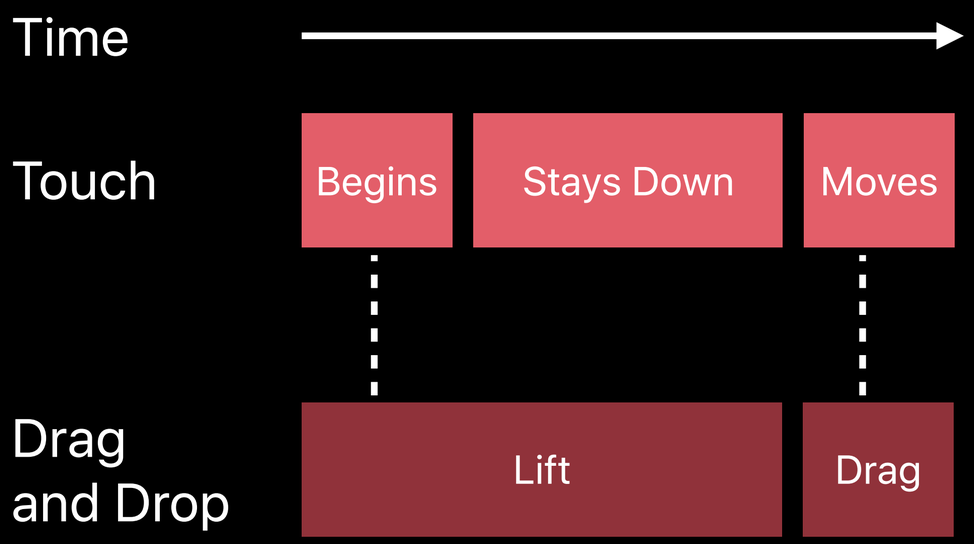
Drag and Drop Timeline



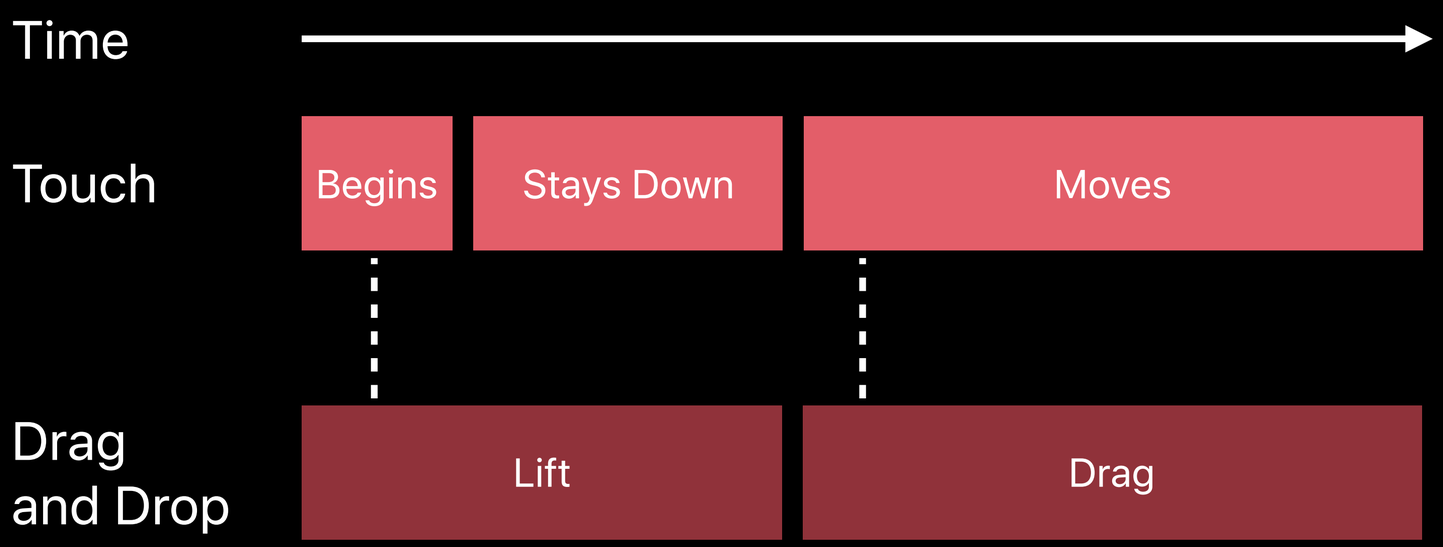
Drag and Drop Timeline



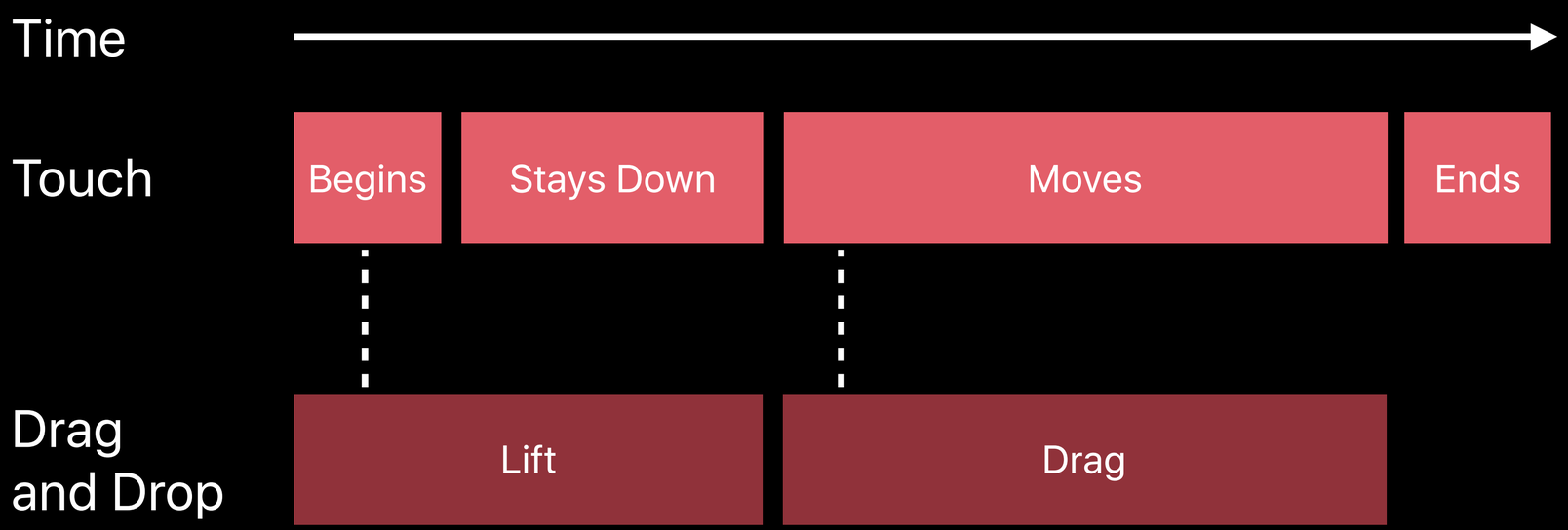
Drag and Drop Timeline



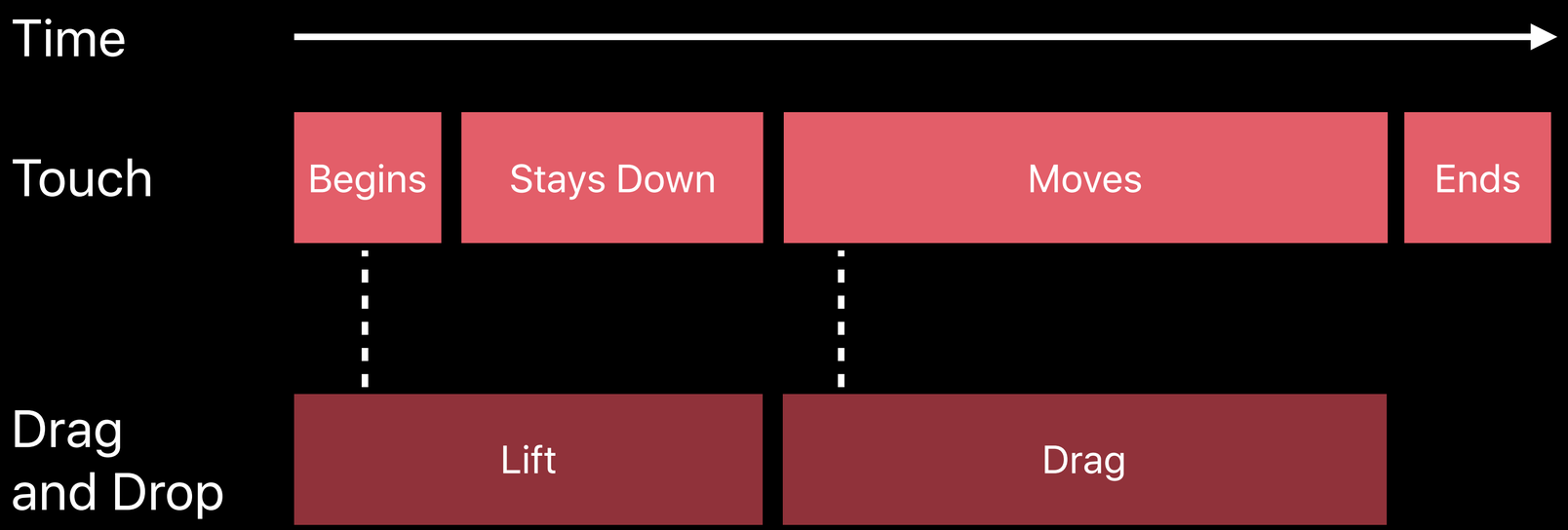
Drag and Drop Timeline



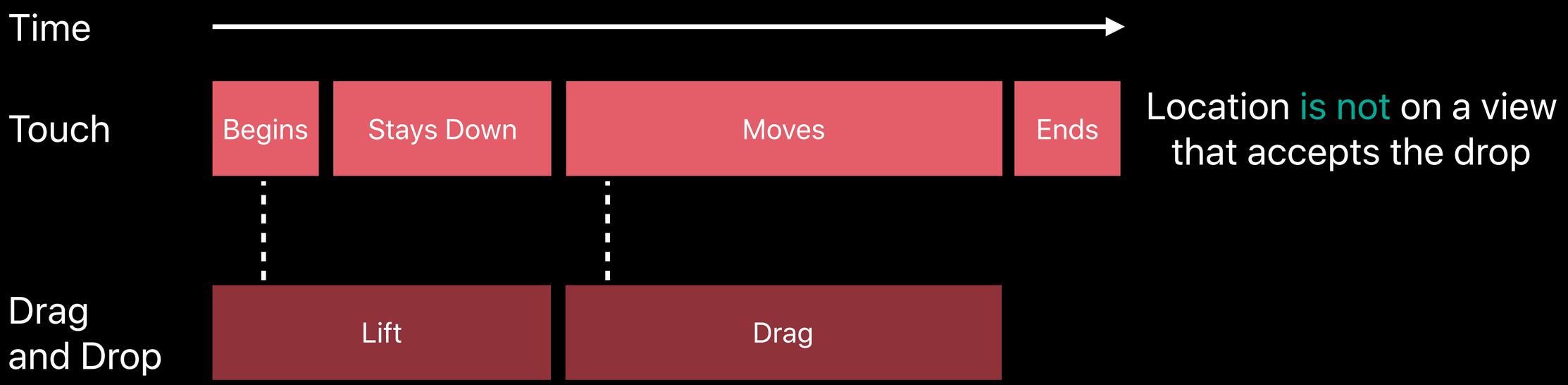
Drag and Drop Timeline



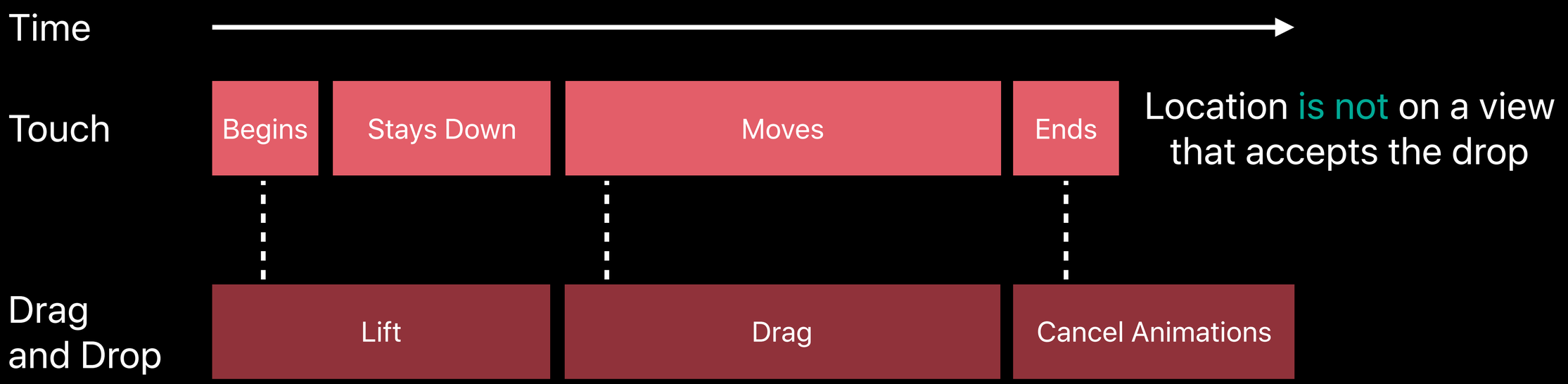
Drag and Drop Timeline



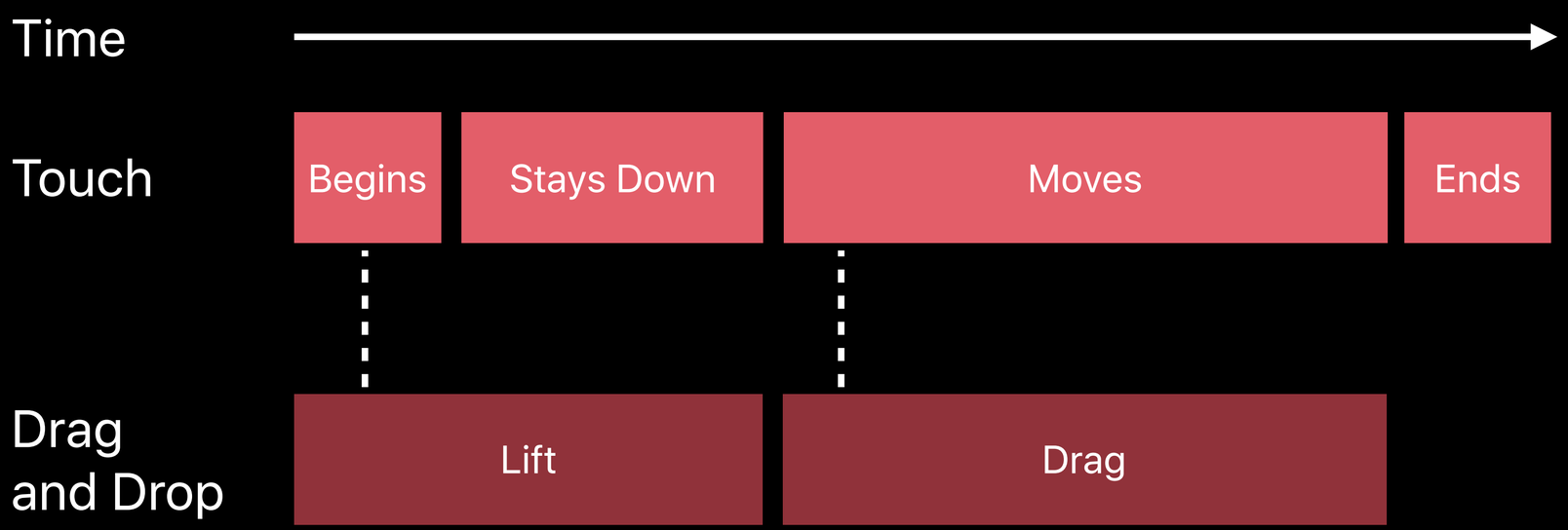
Drag and Drop Timeline



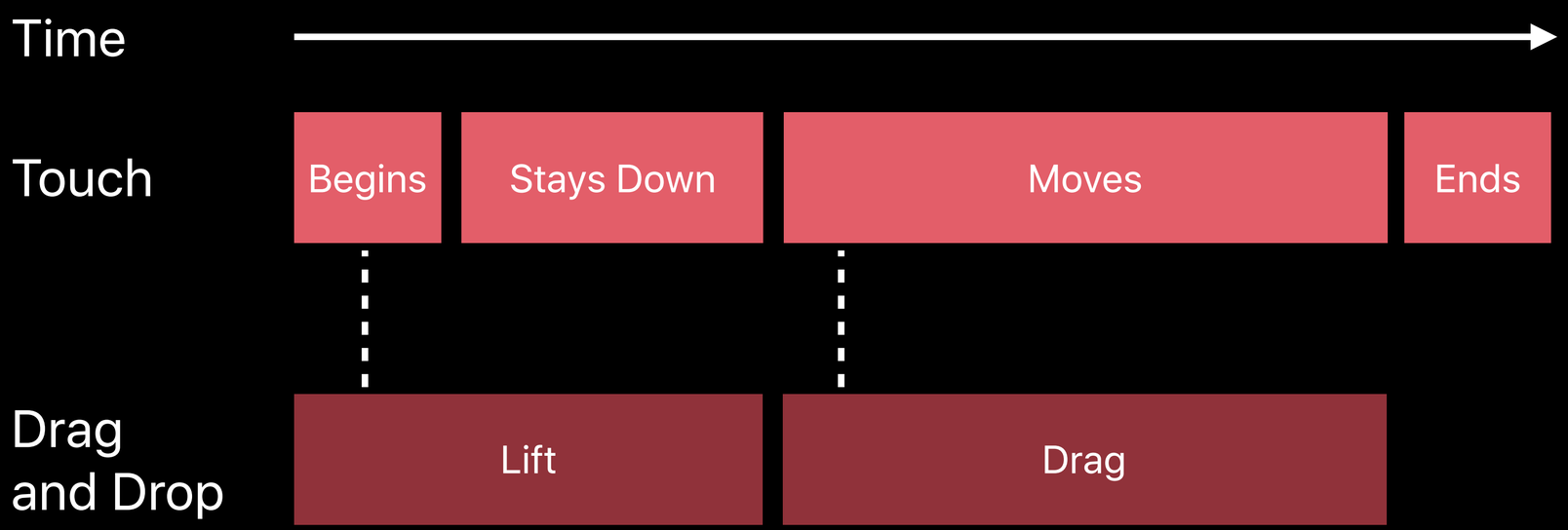
Drag and Drop Timeline



Drag and Drop Timeline



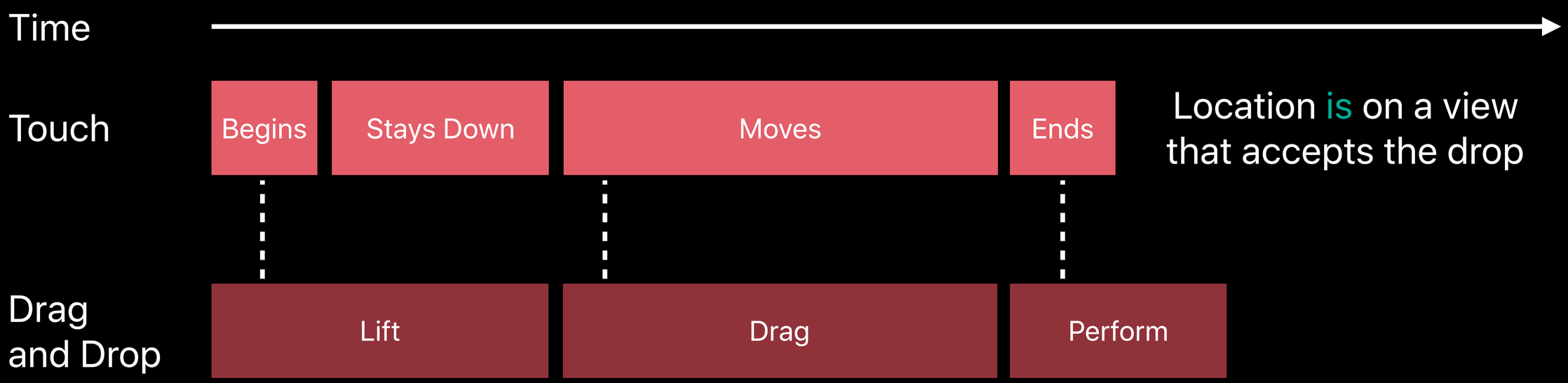
Drag and Drop Timeline



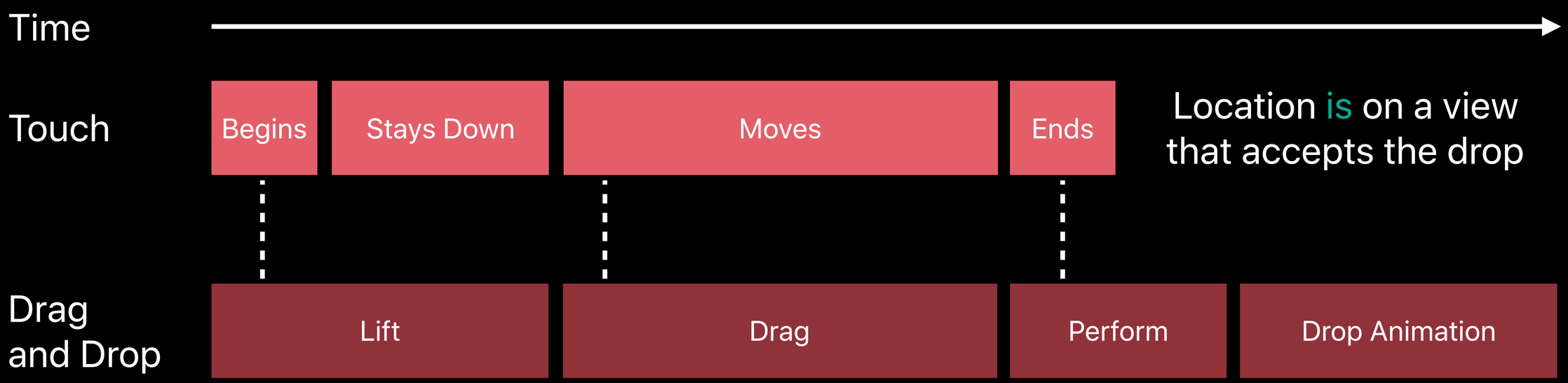
Drag and Drop Timeline

Drag and Drop Timeline

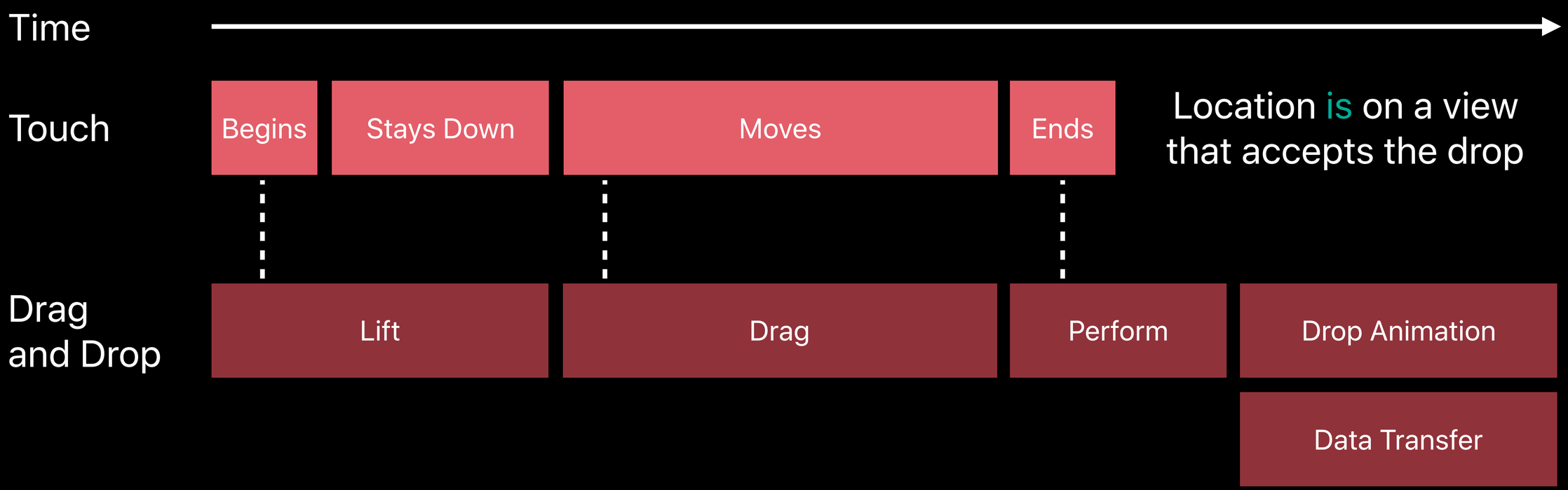
Drag and Drop Timeline



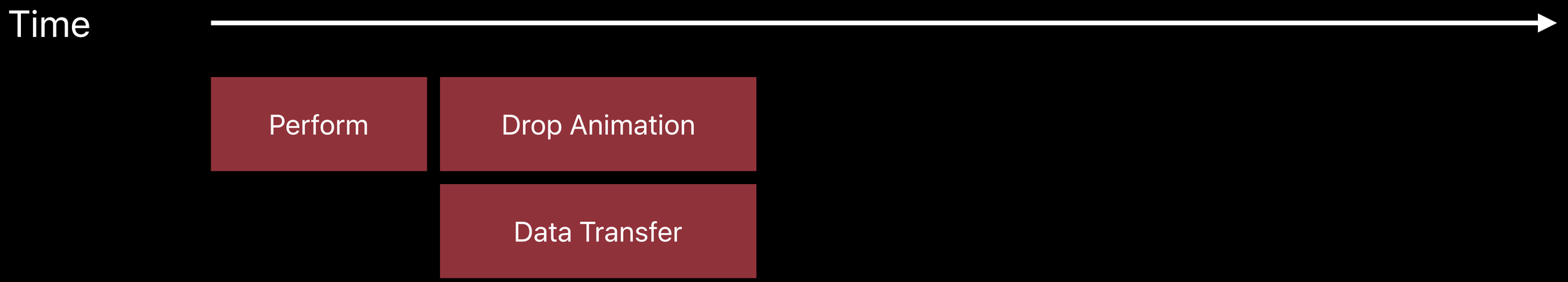
Drag and Drop Timeline



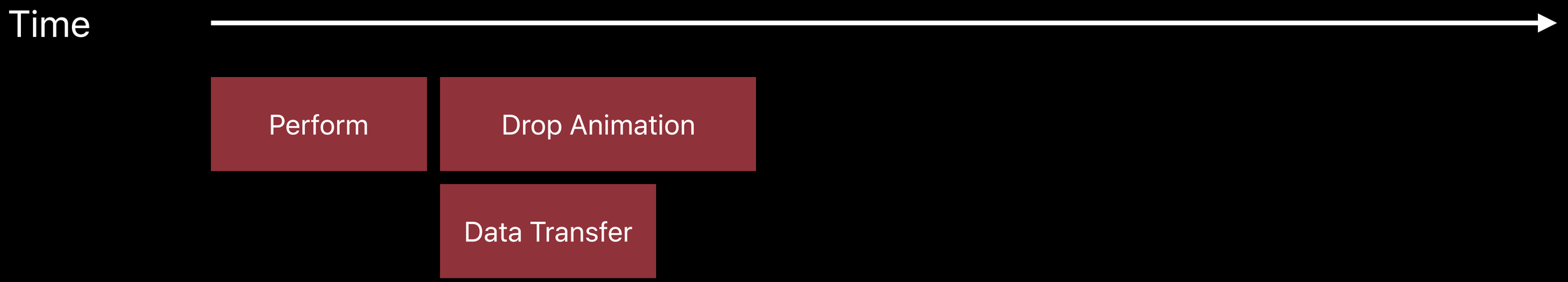
Drag and Drop Timeline



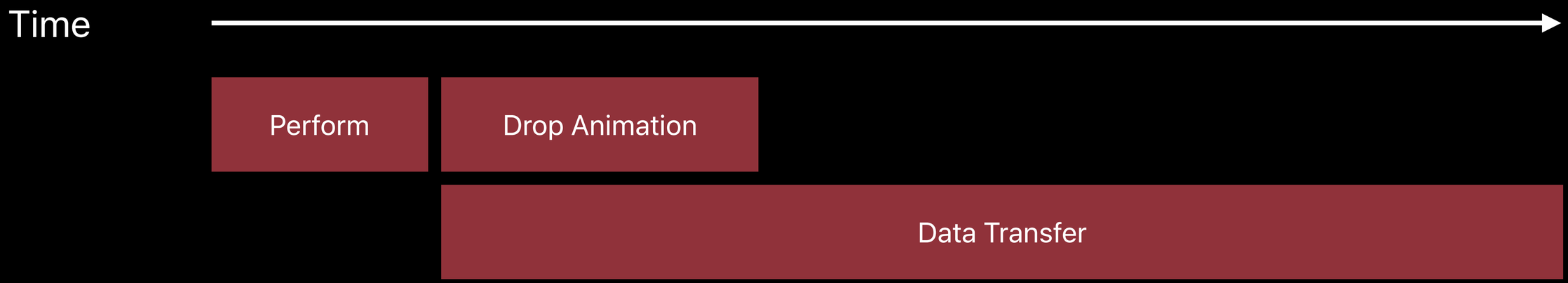
Drag and Drop Timeline



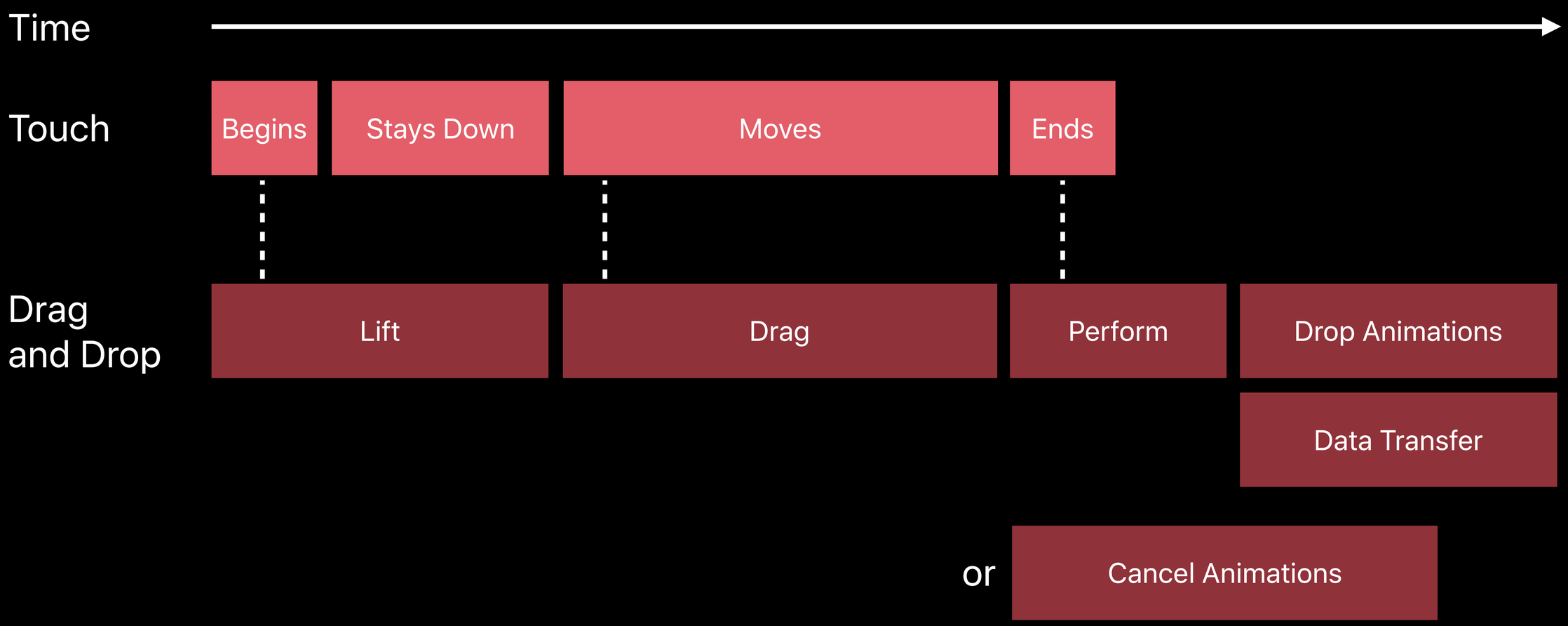
Drag and Drop Timeline



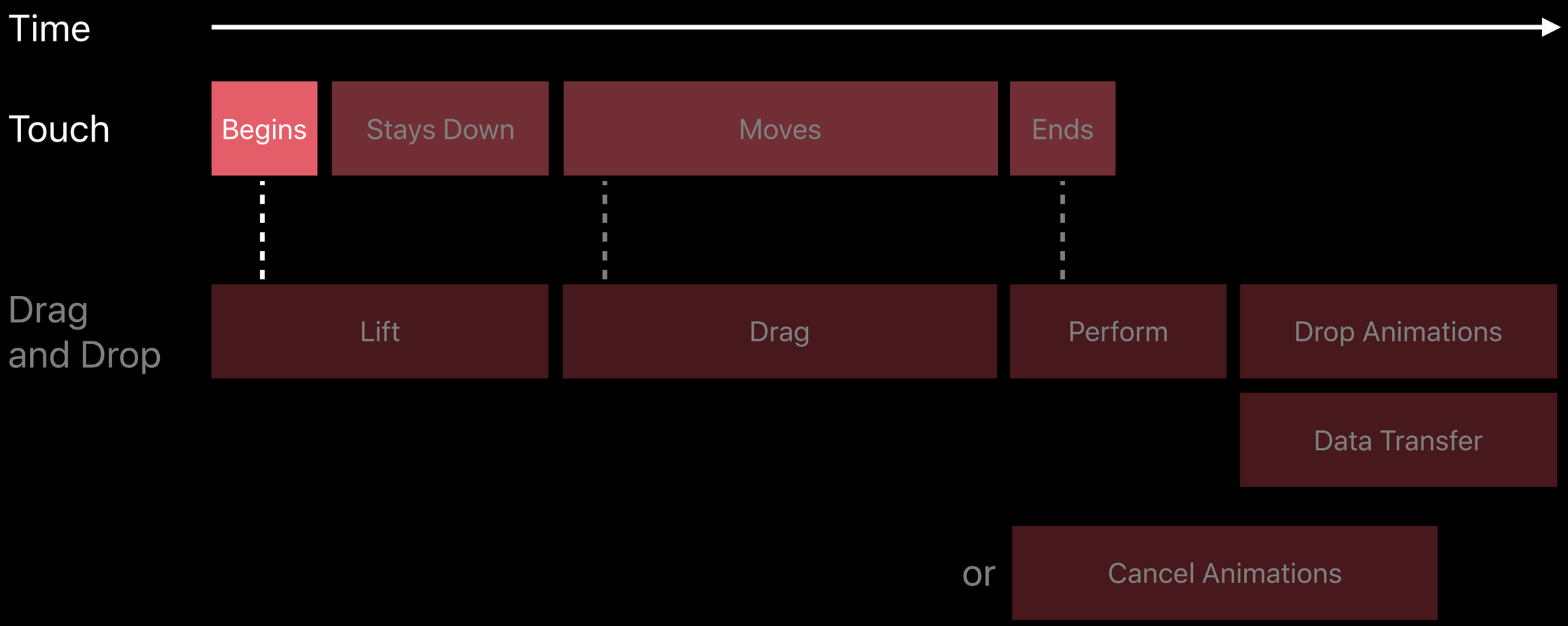
Drag and Drop Timeline



Drag and Drop Timeline



API Essentials—1



API Essentials—1

Get the items to drag

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

API Essentials—1

Get the items to drag

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

API Essentials—1

Get the items to drag

```
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    let itemProvider = NSItemProvider(object: "Hello World" as NSString)
    let dragItem = UIDragItem(itemProvider: itemProvider)
    return [ dragItem ]
}
```

API Essentials—1

Get the items to drag

```
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    let itemProvider = NSItemProvider(object: "Hello World" as NSString)
    let dragItem = UIDragItem(itemProvider: itemProvider)
    return [ dragItem ]
}
```

API Essentials—1

Get the items to drag

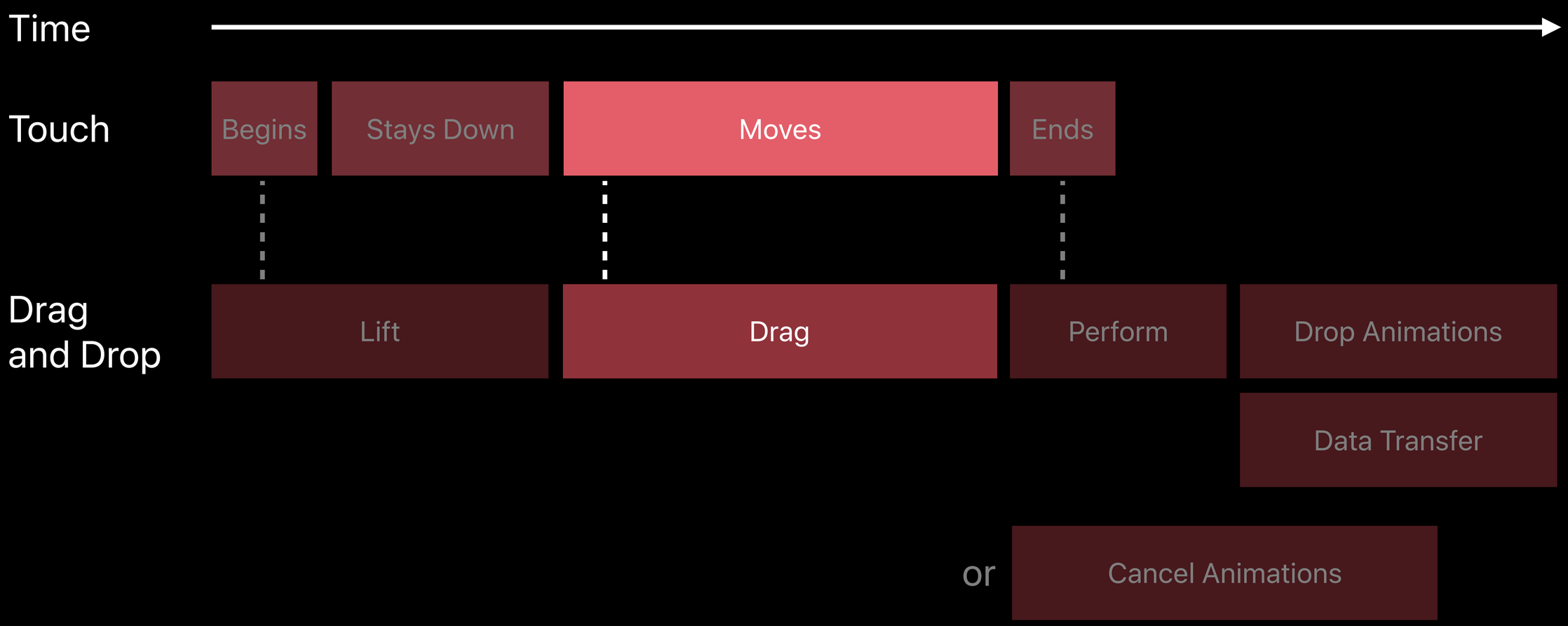
```
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    let itemProvider = NSItemProvider(object: "Hello World" as NSString)
    let dragItem = UIDragItem(itemProvider: itemProvider)
    return [ dragItem ]
}
```

API Essentials—1

Get the items to drag

```
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    let itemProvider = NSItemProvider(object: "Hello World" as NSString)
    let dragItem = UIDragItem(itemProvider: itemProvider)
    return [ dragItem ]
}
```

API Essentials—2



API Essentials—2

Get the drop proposal

```
func dropInteraction(_ interaction: UIDropInteraction,
                     sessionDidUpdate session: UIDropSession) -> UIDropProposal {
    return UIDropProposal(operation: UIDropOperation)
}
```

API Essentials—2

Get the drop proposal

```
func dropInteraction(_ interaction: UIDropInteraction,
                     sessionDidUpdate session: UIDropSession) -> UIDropProposal {
    return UIDropProposal(operation: UIDropOperation)
}
```

API Essentials—2

Get the drop proposal

```
func dropInteraction(_ interaction: UIDropInteraction,
                     sessionDidUpdate session: UIDropSession) -> UIDropProposal {
    return UIDropProposal(operation: UIDropOperation)
}
```

API Essentials—2

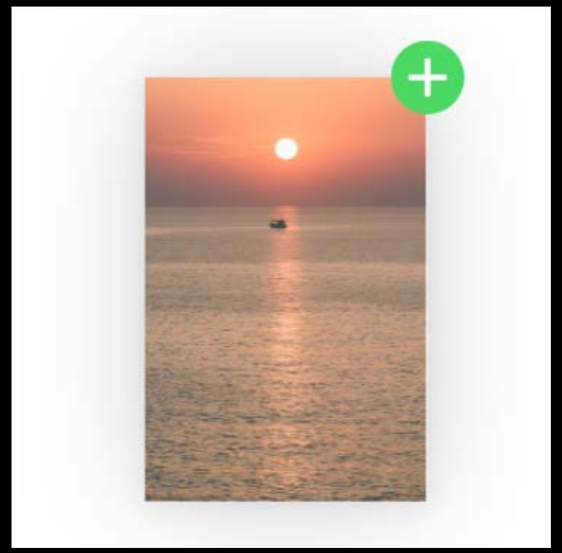
UIDropOperation

API Essentials—2

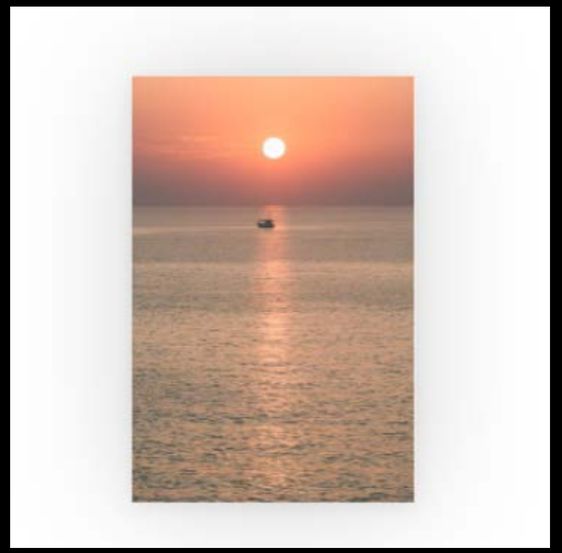
UIDropOperation

`.cancel`

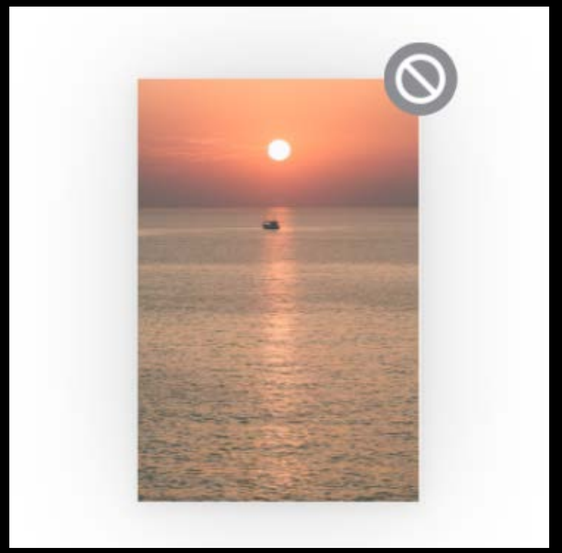
`.copy`



`.move`



`.forbidden`

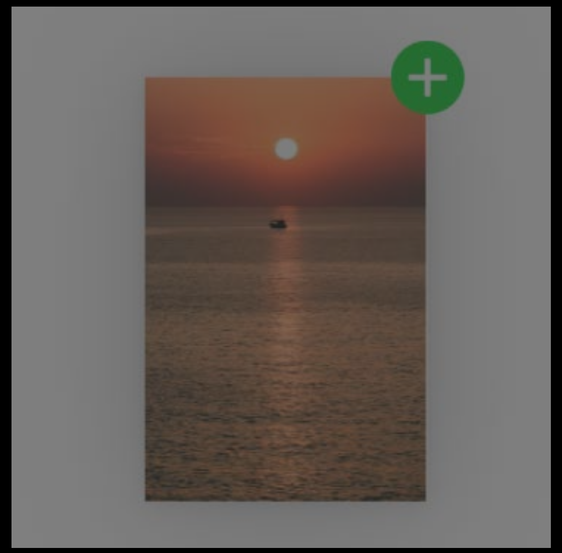


API Essentials—2

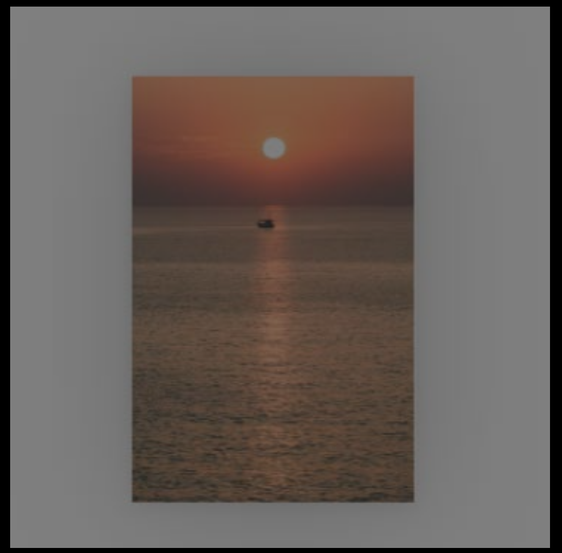
UIDropOperation

`.cancel`

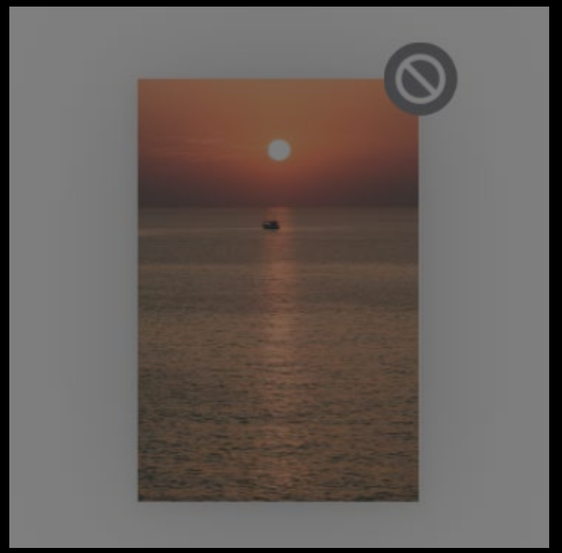
`.copy`



`.move`



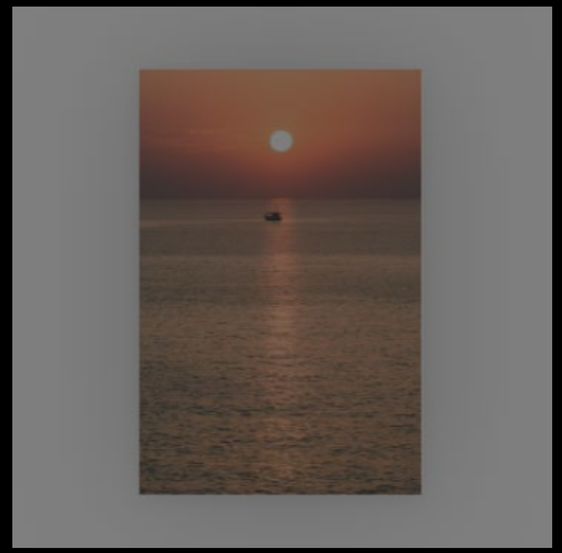
`.forbidden`



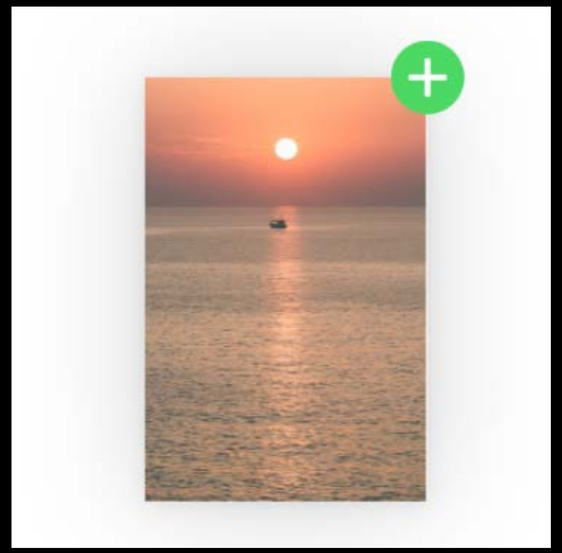
API Essentials—2

UIDropOperation

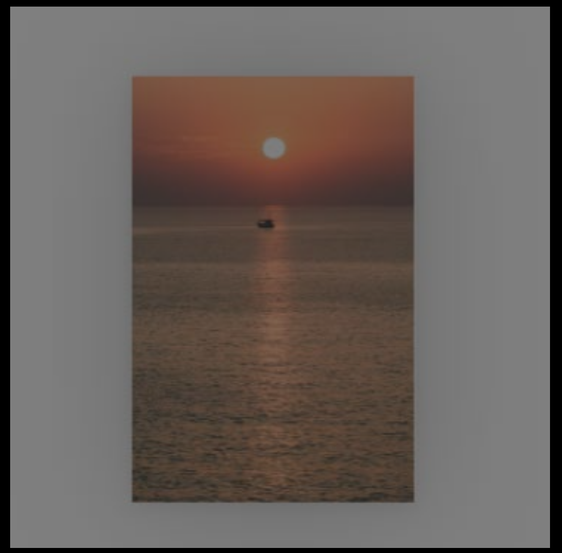
`.cancel`



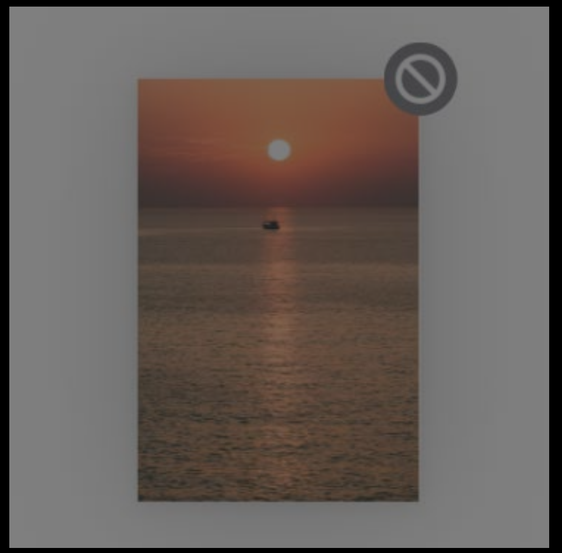
`.copy`



`.move`



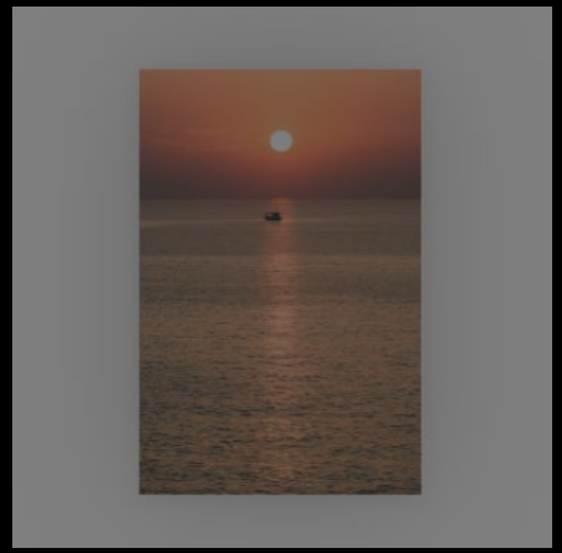
`.forbidden`



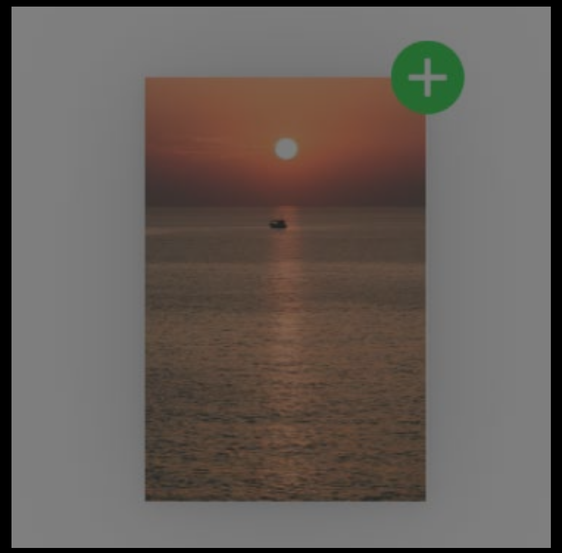
API Essentials—2

UIDropOperation

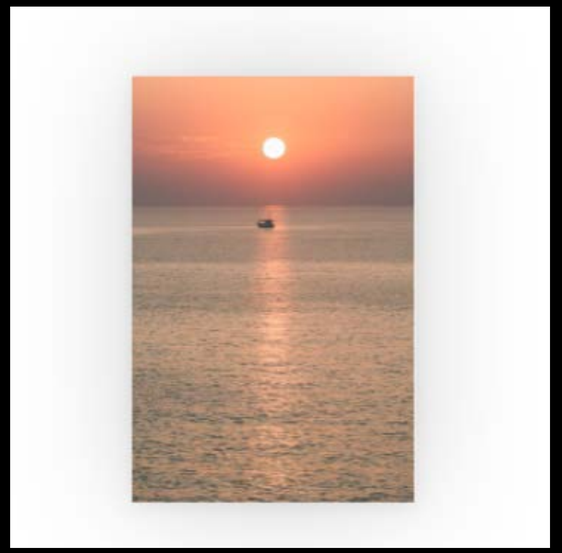
`.cancel`



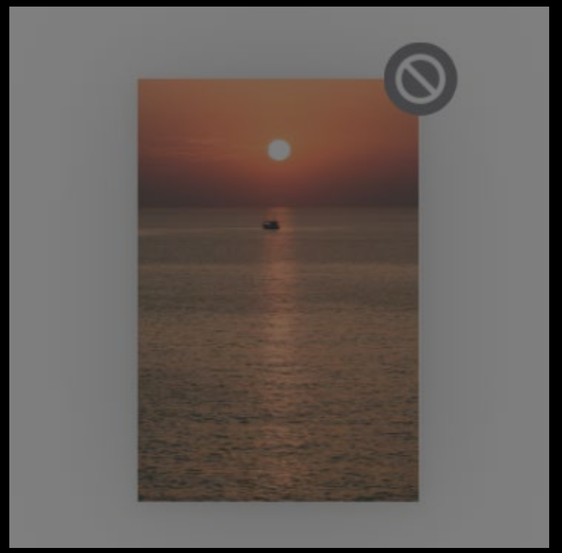
`.copy`



`.move`



`.forbidden`

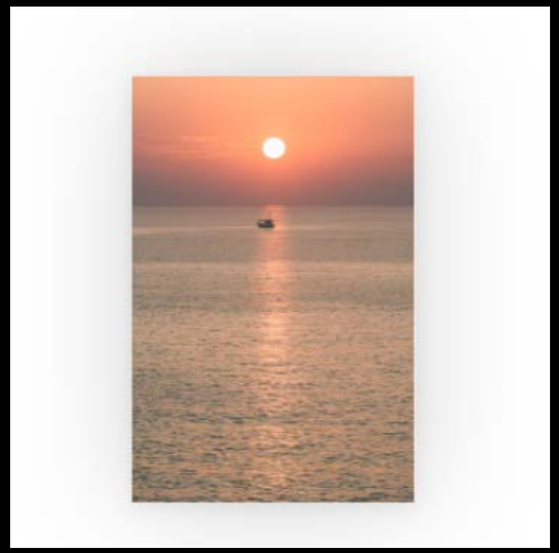


API Essentials—2

UIDropOperation

Delegates must cooperate
to make it look like a move

`.move`



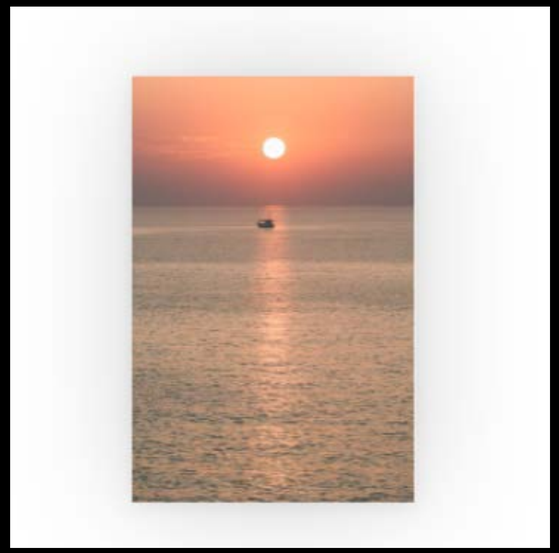
API Essentials—2

UIDropOperation

Delegates must cooperate
to make it look like a move

Only within a single app

`.move`



API Essentials—2

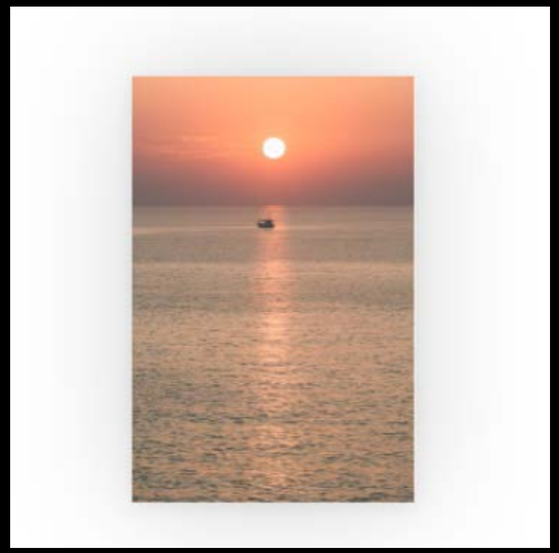
UIDropOperation

Delegates must cooperate to make it look like a move

Only within a single app

Drag interaction delegate must allow moves

`.move`



API Essentials—2

UIDropOperation

Delegates must cooperate
to make it look like a move

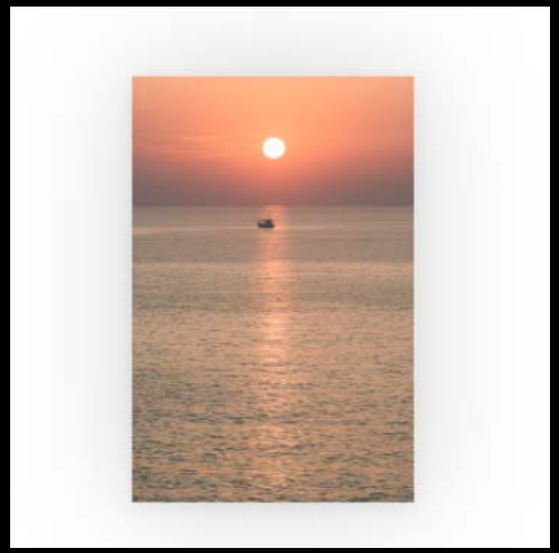
Only within a single app

Drag interaction delegate
must allow moves

Drop interaction delegate checks

`UIDropSession allowsMoveOperation`

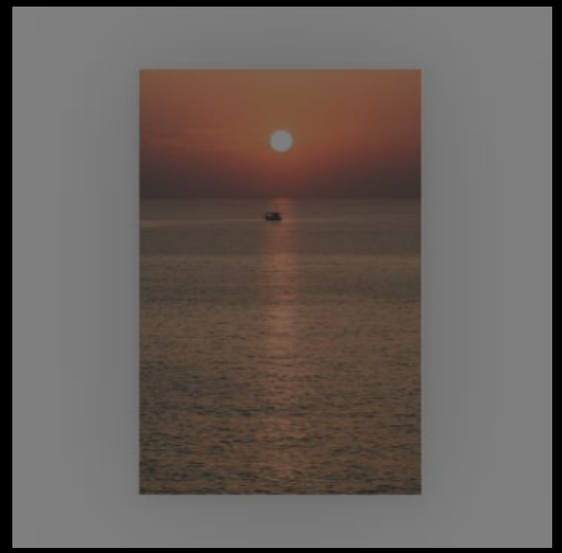
`.move`



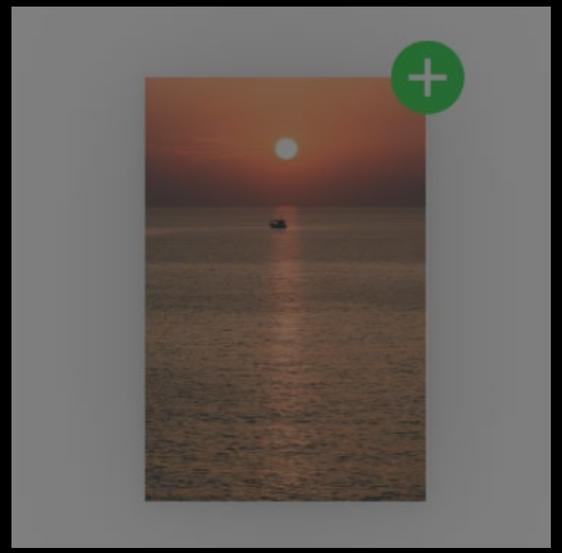
API Essentials—2

UIDropOperation

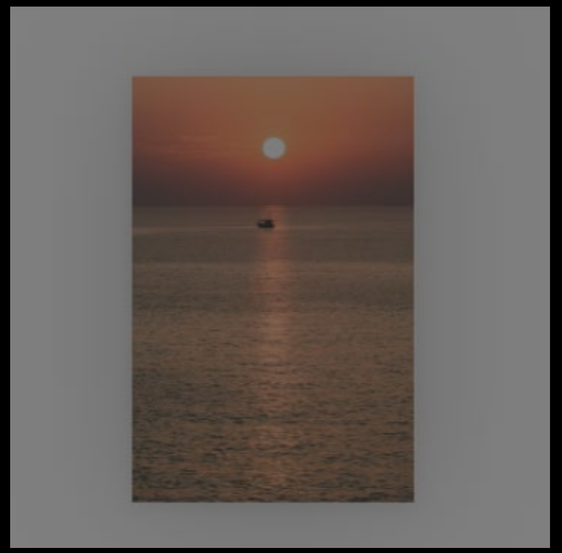
`.cancel`



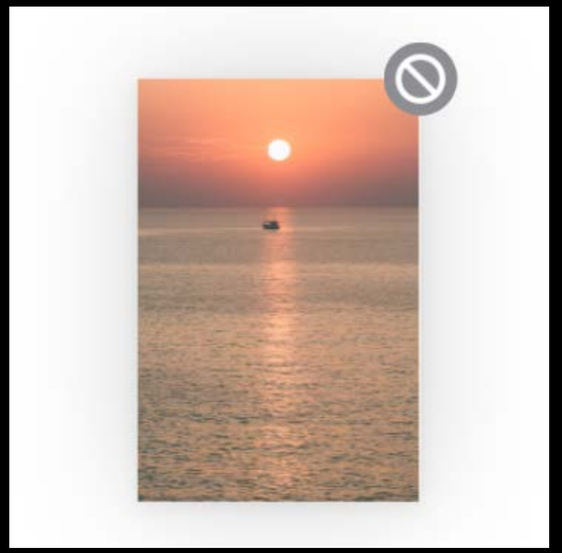
`.copy`



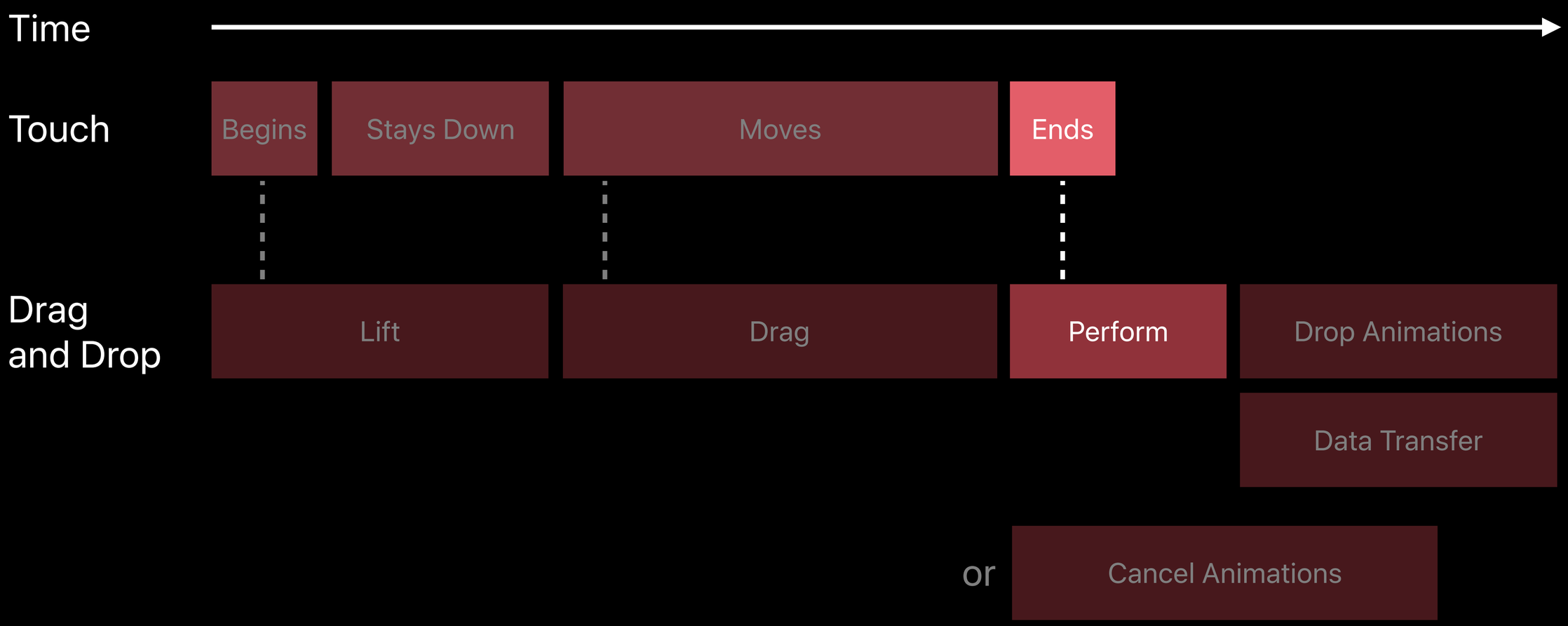
`.move`



`.forbidden`



API Essentials—3



API Essentials—3

Perform the drop

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

API Essentials—3

Perform the drop

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

API Essentials—3

Perform the drop

```
func dropInteraction(_ interaction: UIDropInteraction, performDrop session: UIDropSession) {
    session.loadObjects(ofClass: UIImage.self) { objects in
        for image in objects as! [UIImage] {
            self.imageView.image = image
        }
    }
}
```

API Essentials—3

Perform the drop

```
func dropInteraction(_ interaction: UIDropInteraction, performDrop session: UIDropSession) {
    session.loadObjects(ofClass: UIImage.self) { objects in
        for image in objects as! [UIImage] {
            self.imageView.image = image
        }
    }
}
```

API Essentials—3

Perform the drop

```
func dropInteraction(_ interaction: UIDropInteraction, performDrop session: UIDropSession) {
    session.loadObjects(ofClass: UIImage.self) { objects in
        for image in objects as! [UIImage] {
            self.imageView.image = image
        }
    }
}
```

API Essentials—3

Perform the drop

```
func dropInteraction(_ interaction: UIDropInteraction, performDrop session: UIDropSession) {
    session.loadObjects(ofClass: UIImage.self) { objects in
        for image in objects as! [UIImage] {
            self.imageView.image = image
        }
    }
}
```

API Essentials—3

Perform the drop

```
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—3

Perform the drop

```
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—3

Perform the drop

```
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—3

Perform the drop

```
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—3

Perform the drop

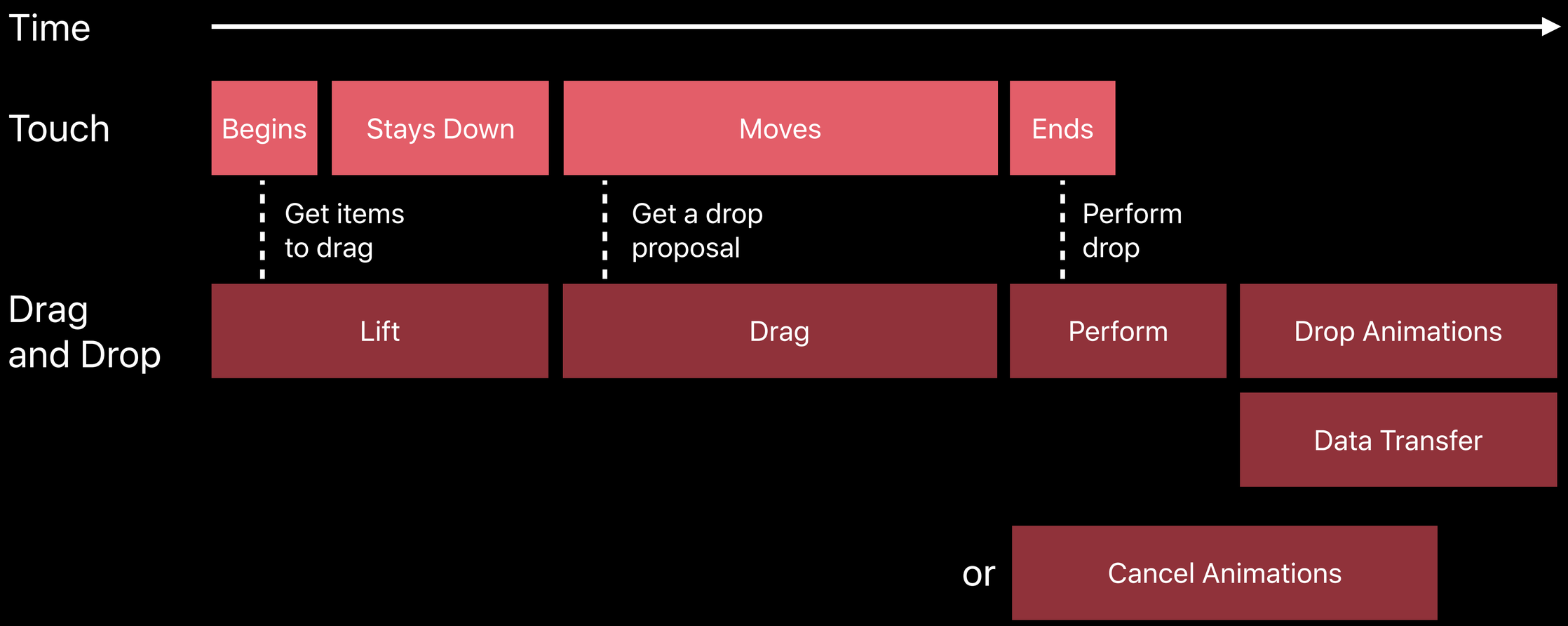
```
func dropInteraction(_ interaction: UIDropInteraction, performDrop session: UIDropSession) {
    for item in session.items {
        item.itemProvider.loadObject(ofClass: UIImage.self) { (object, error) in
            if object != nil {
                DispatchQueue.main.async {
                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

API Essentials

API Essentials

API Essentials

API Essentials



Drag Interaction Delegate

Drag Interaction Delegate

Lift

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

Drag Interaction Delegate

Lift

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

Drag Interaction Delegate

Lift

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

Drag Interaction Delegate

Lift

```
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

```
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

```
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

```
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

```
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

```
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

Lift

```
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

Lift

```
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

Lift

```
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

```
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

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)
```

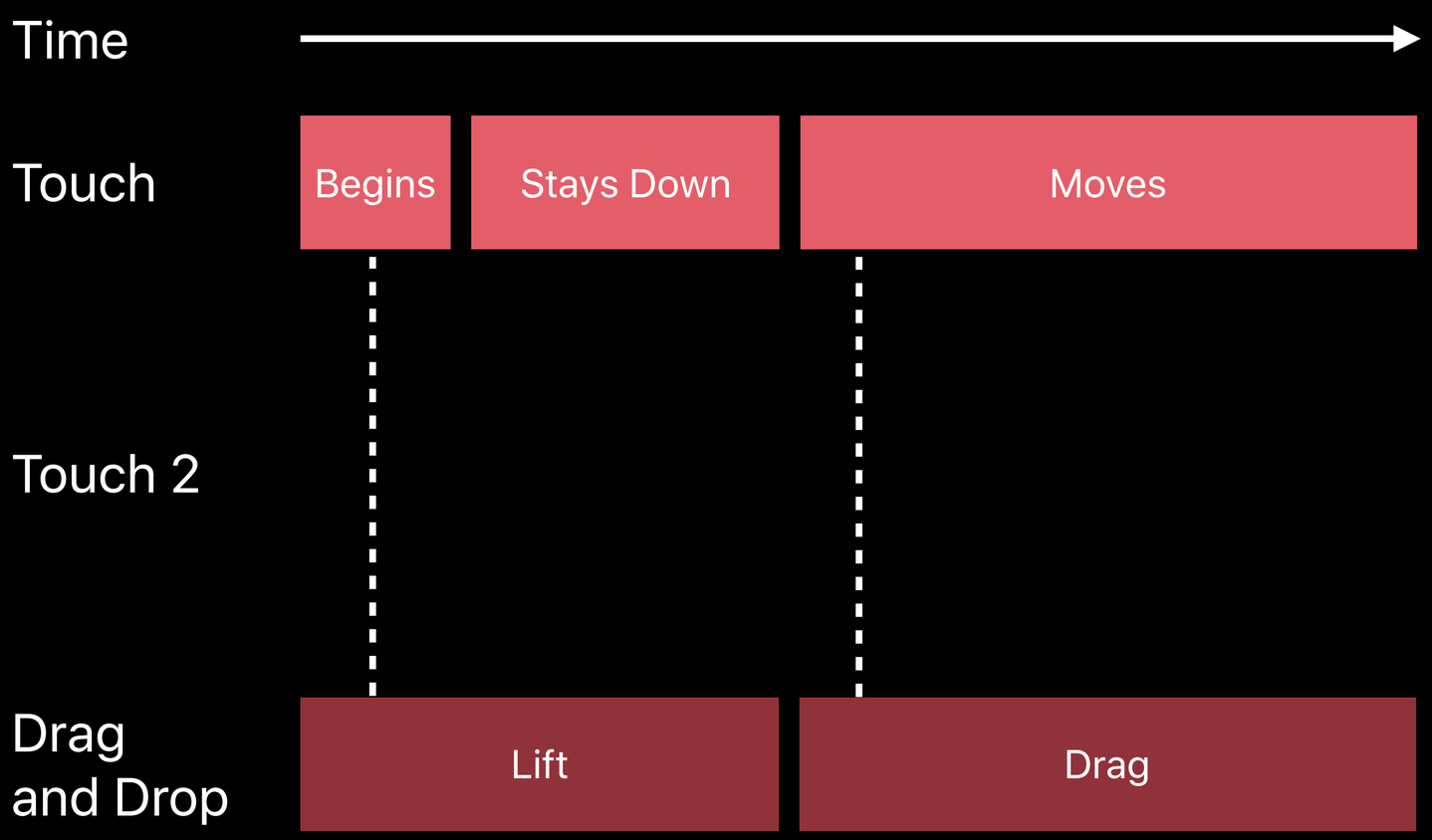
Drag Interaction Delegate

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)
```

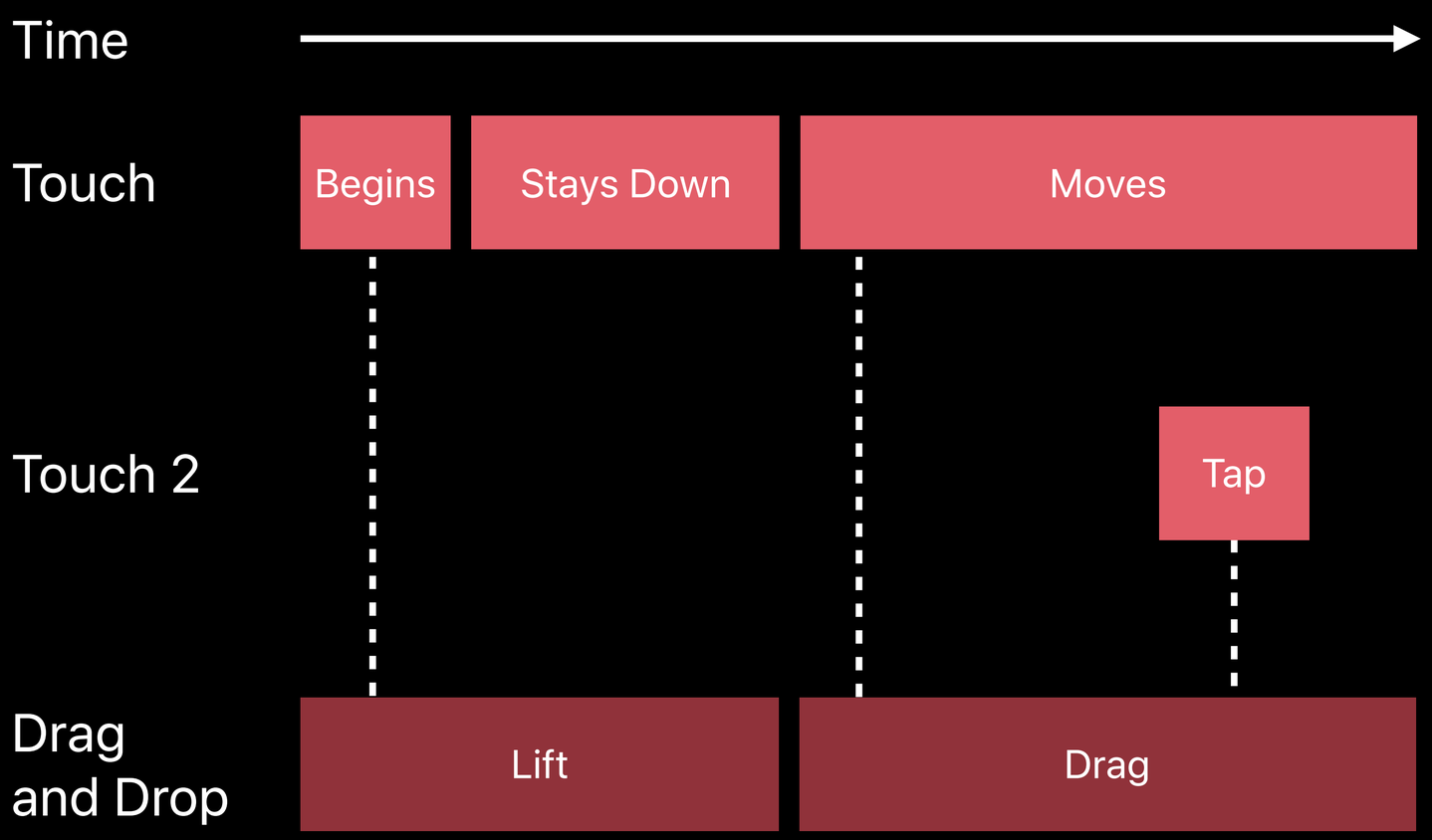
Drag Interaction Delegate

Adding items



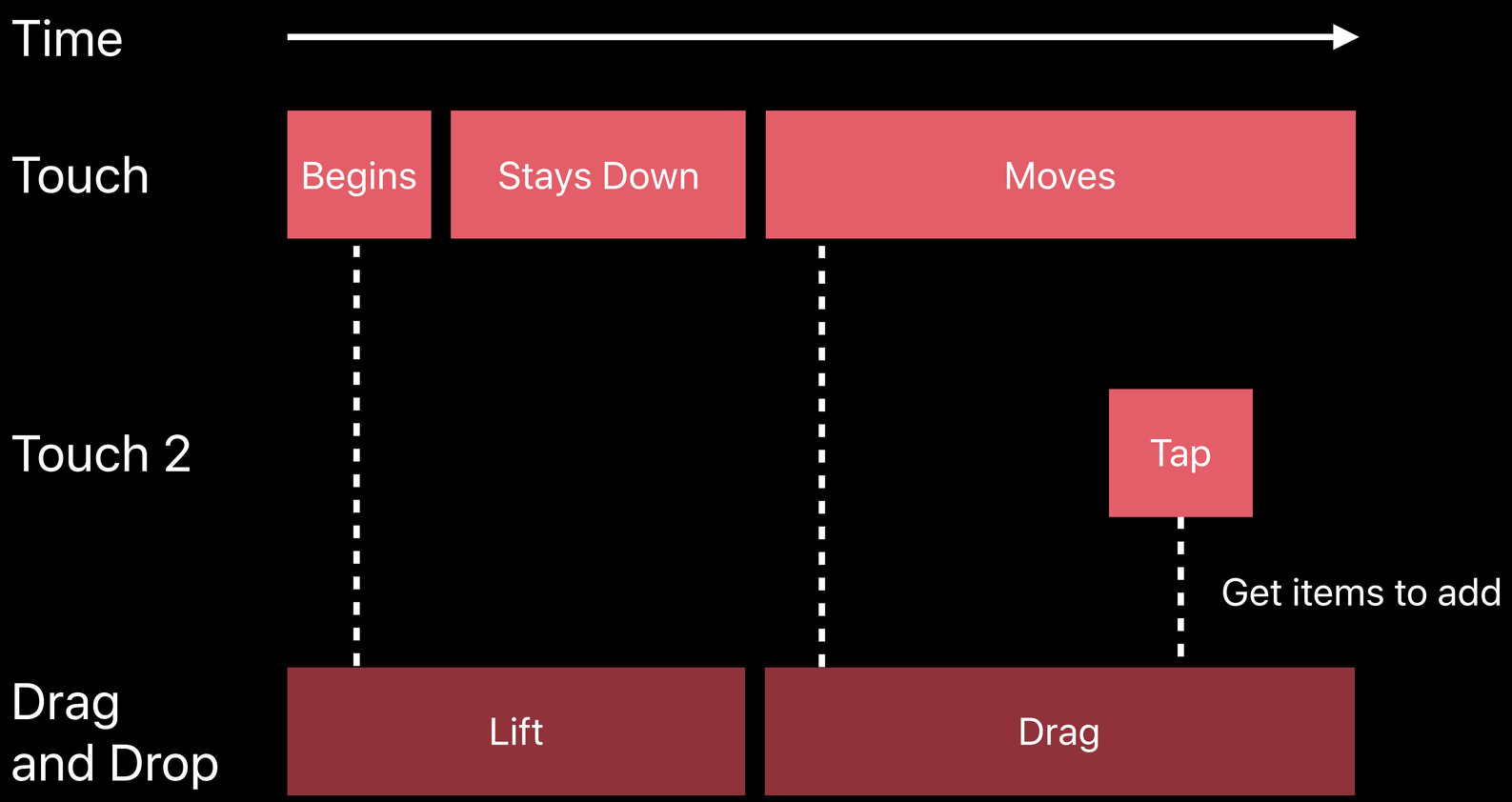
Drag Interaction Delegate

Adding items



Drag Interaction Delegate

Adding items



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

```
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

The session ends

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

Drag Interaction Delegate

The session ends

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

Drag Interaction Delegate

The session ends in a cancel

```
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

```
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

```
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

```
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

```
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

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

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

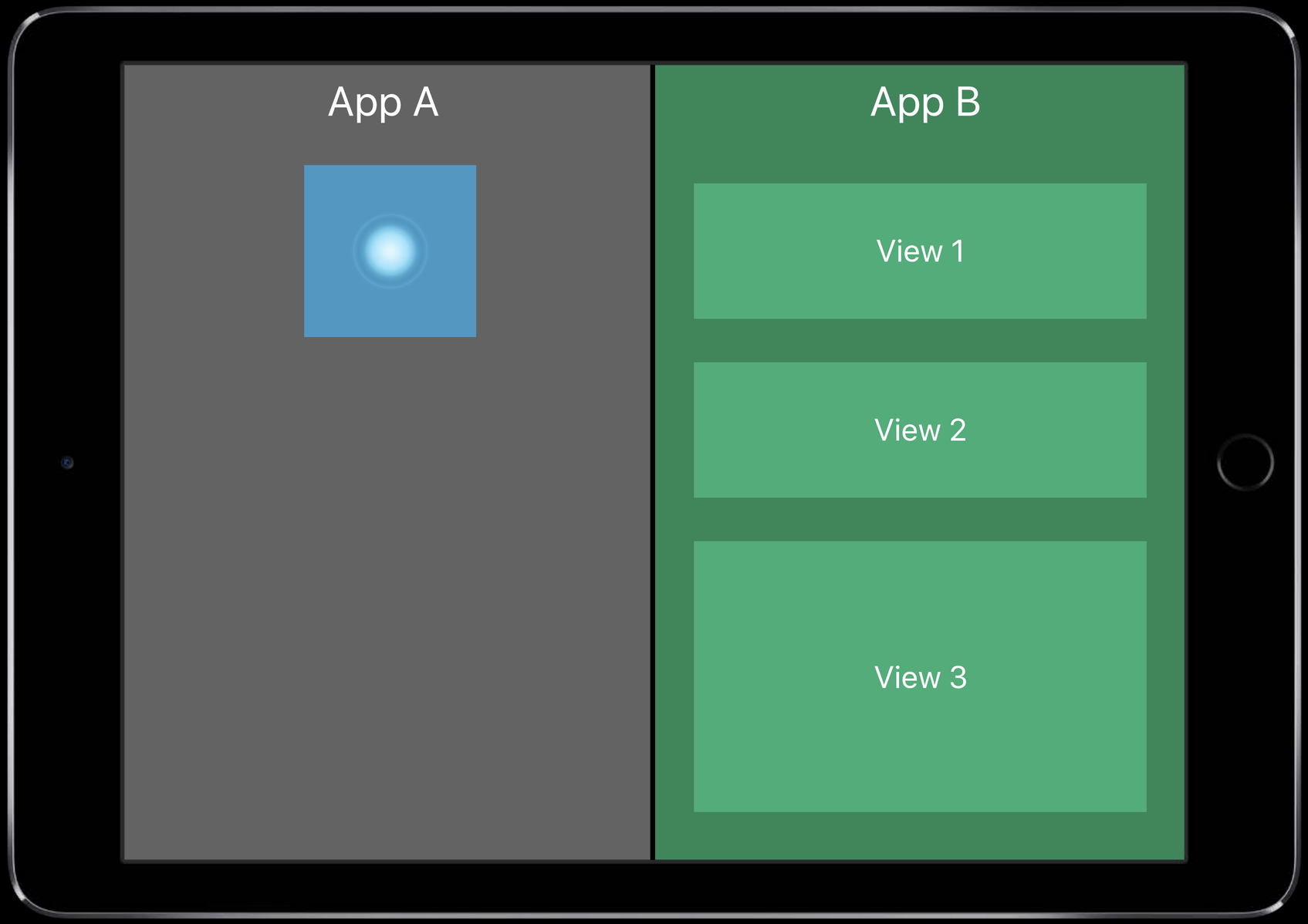
Drag Interaction Delegate

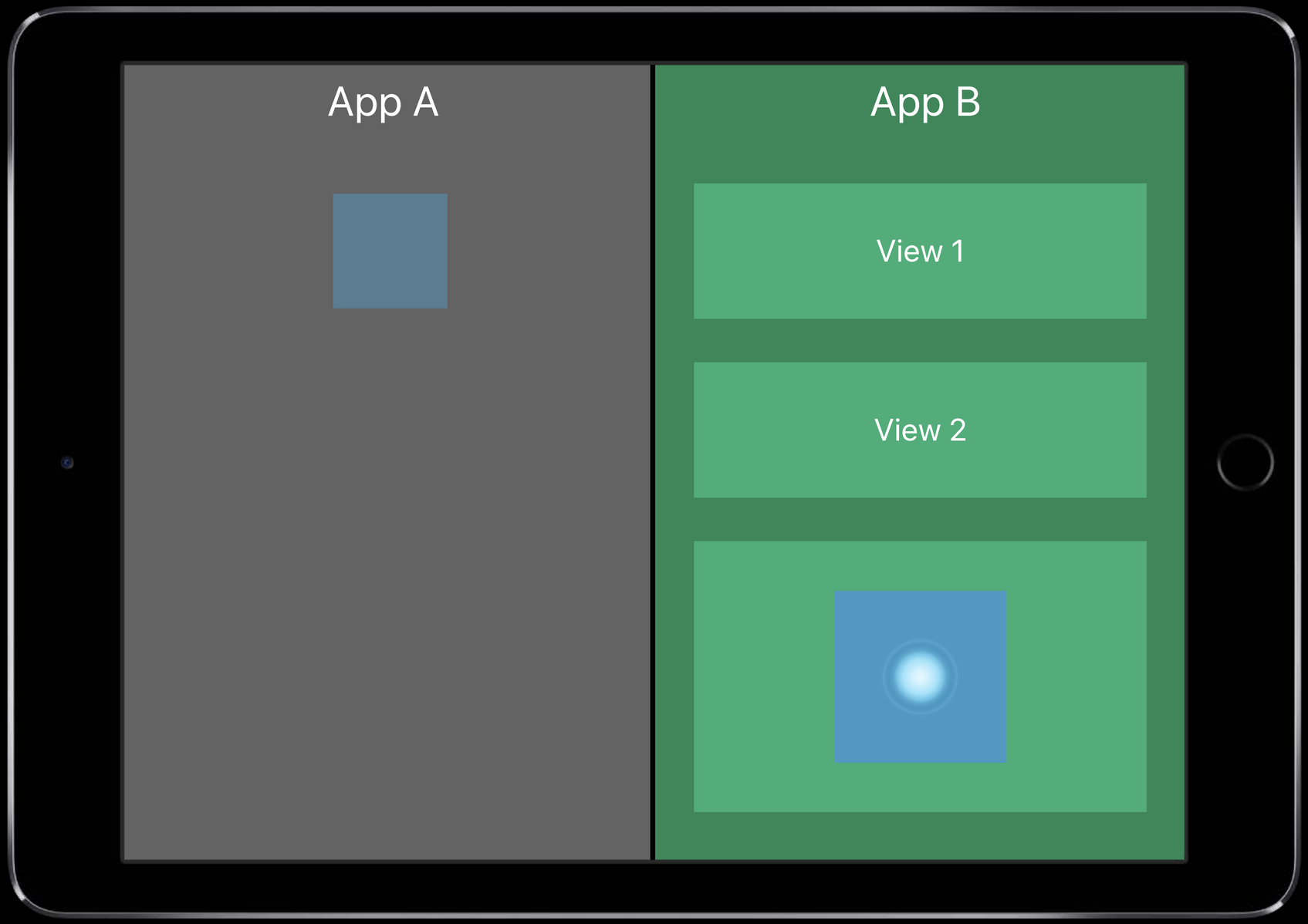
The session ends in a copy or move

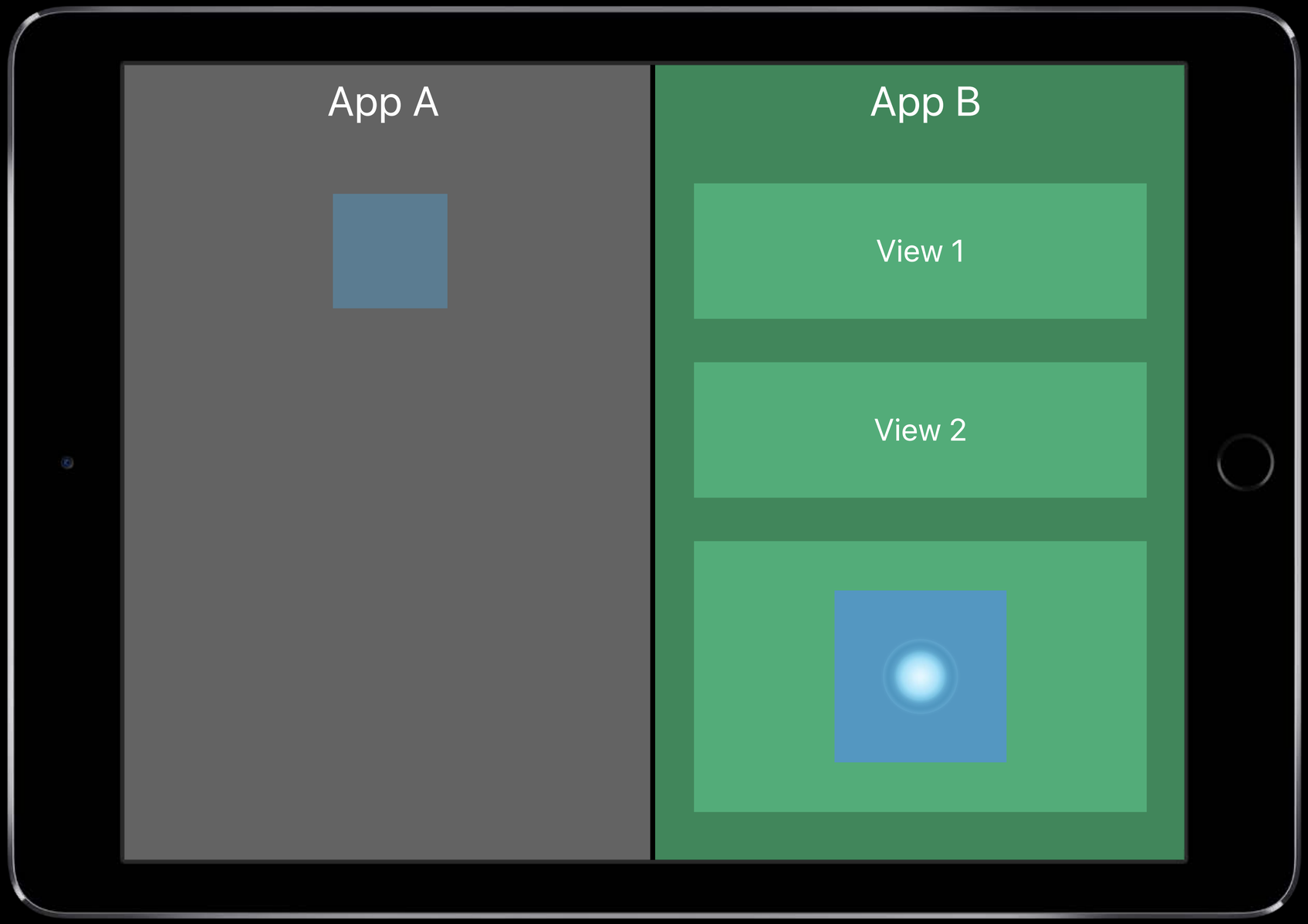
```
func dragInteraction(_ interaction: UIDragInteraction,
                     session: UIDragSession
                     didEndWith operation: UIDropOperation)
```

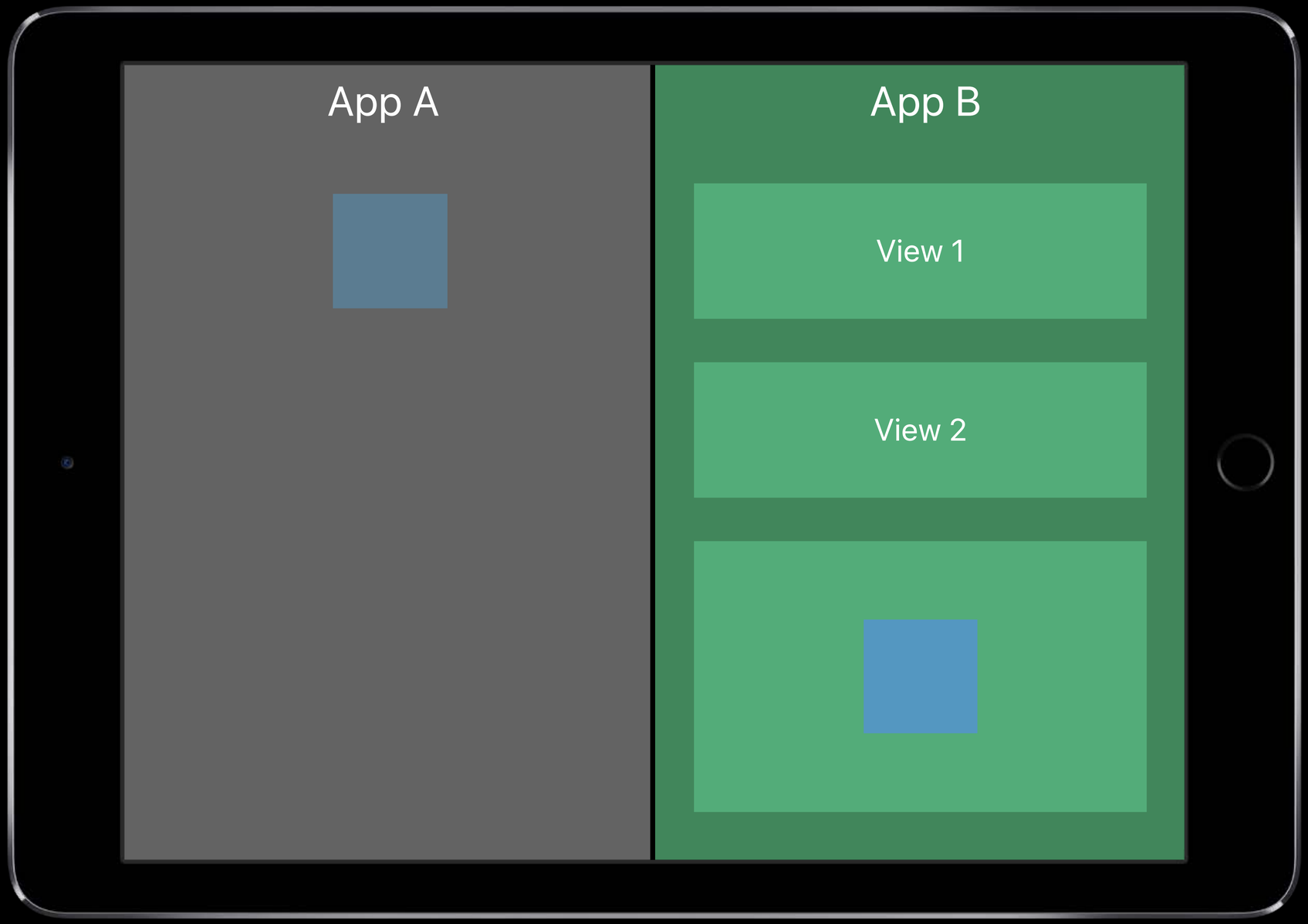
```
func dragInteraction(_ interaction: UIDragInteraction,
                     sessionDidTransferItems session: UIDragSession)
```

Drop Interaction Delegate


App A
App B
View 1
View 2
View 3


App A
App B
View 1
View 2


App A
App B
View 1
View 2


App A
App B
View 1
View 2

Drop Interaction Delegate

Session enters the view

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

Drop Interaction Delegate

Session enters the view

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

Drop Interaction Delegate

Session enters the view

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

Drop Interaction Delegate

Session enters the view

```
import MobileCoreServices // for kUTTypeImagePNG

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

Drop Interaction Delegate

Session enters the view

```
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

```
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

```
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

```
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

```
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

```
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

```
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

```
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

```
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

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

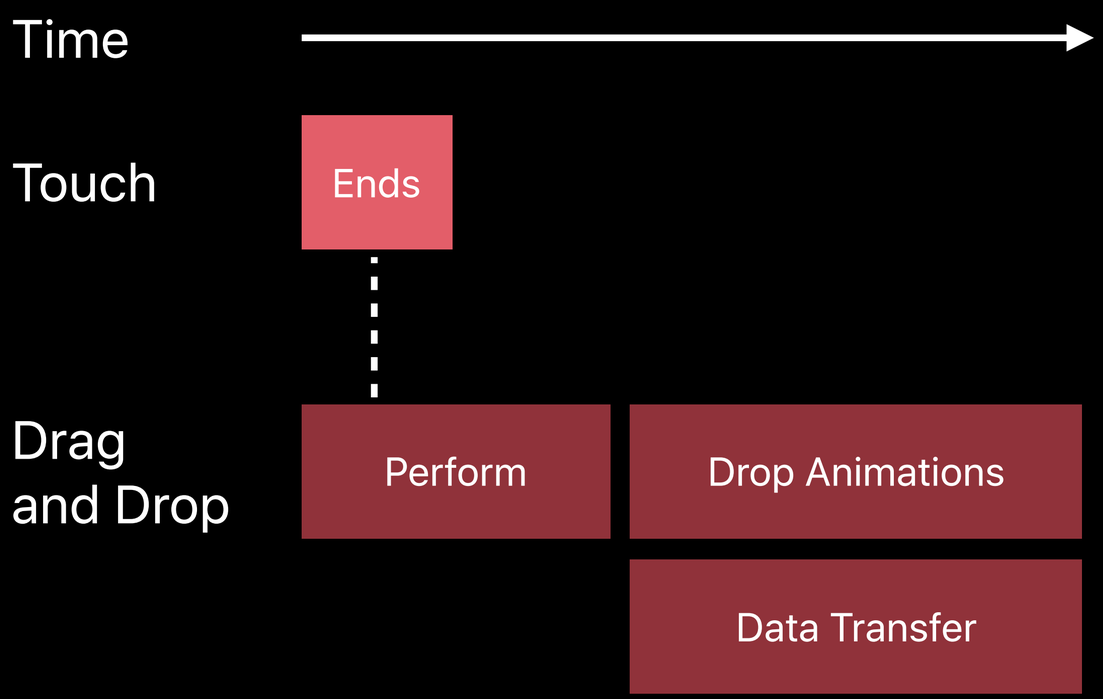
Drop Interaction Delegate

Session ends over a different view

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

Drop Interaction Delegate

Session ends over this view



Drop Interaction Delegate

Drop animations

```
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

```
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

```
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

```
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

Data transfer

```
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
```

Drop Interaction Delegate

Data transfer

```
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
```

Drop Interaction Delegate

Data transfer

```
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
```

Drop Interaction Delegate

Data transfer

```
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
```

Drop Interaction Delegate

Data transfer

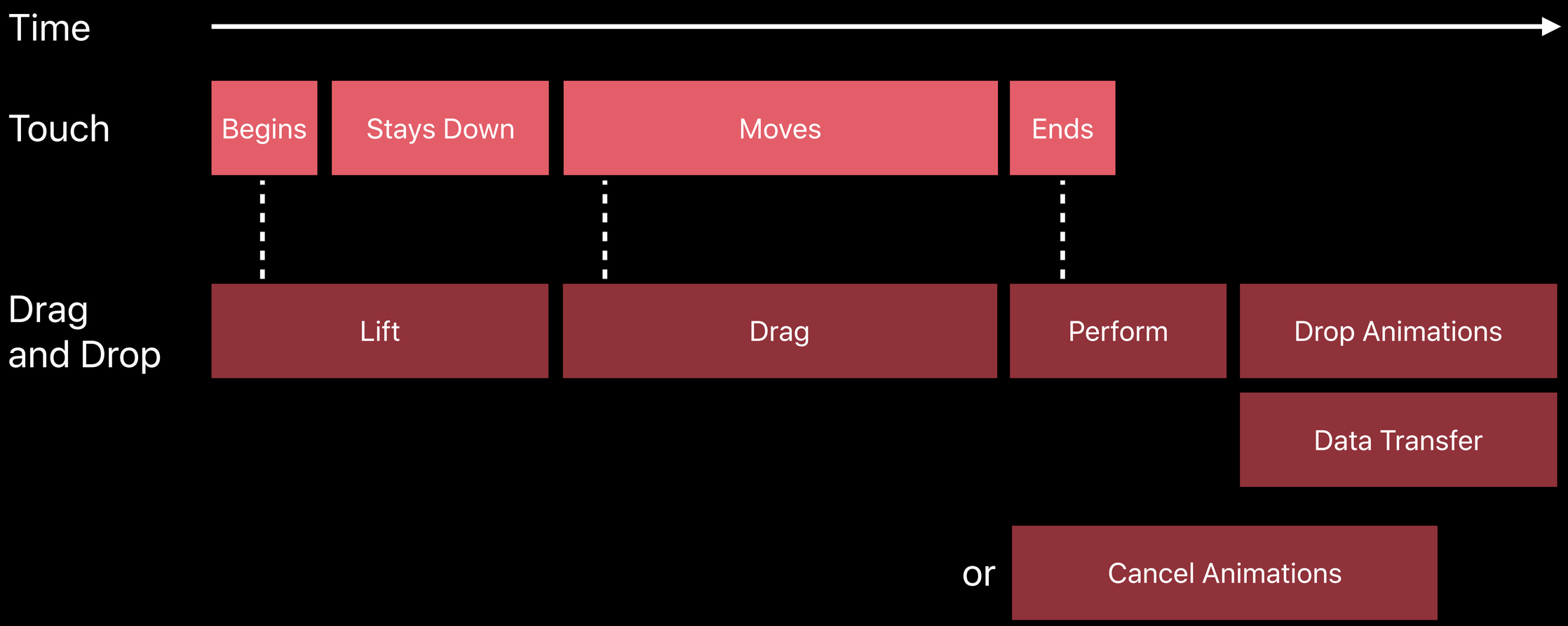
```
let progress = item.itemProvider.loadObject(ofClass: UIImage.self) { (object, error) in
    // Closure is called when object or error are available
}

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

let sessionProgress = session.progress
```

Drop Interaction Delegate

Data transfer

```
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



Interaction Delegates

Essential functions

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

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

func 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

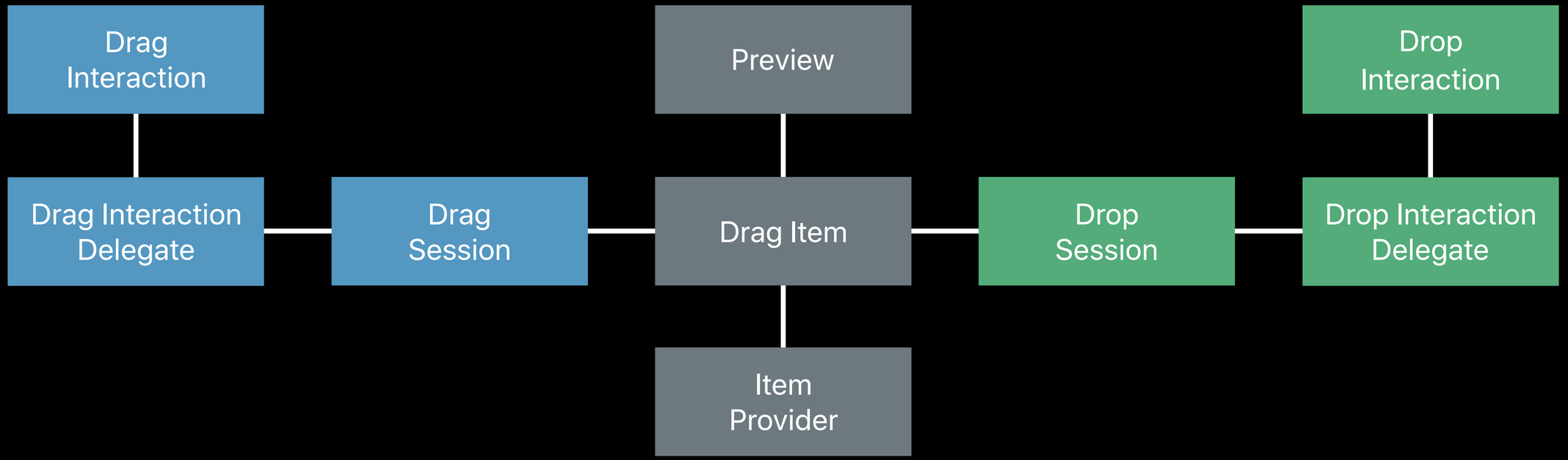
Try adding a drop target

Enable a drag source

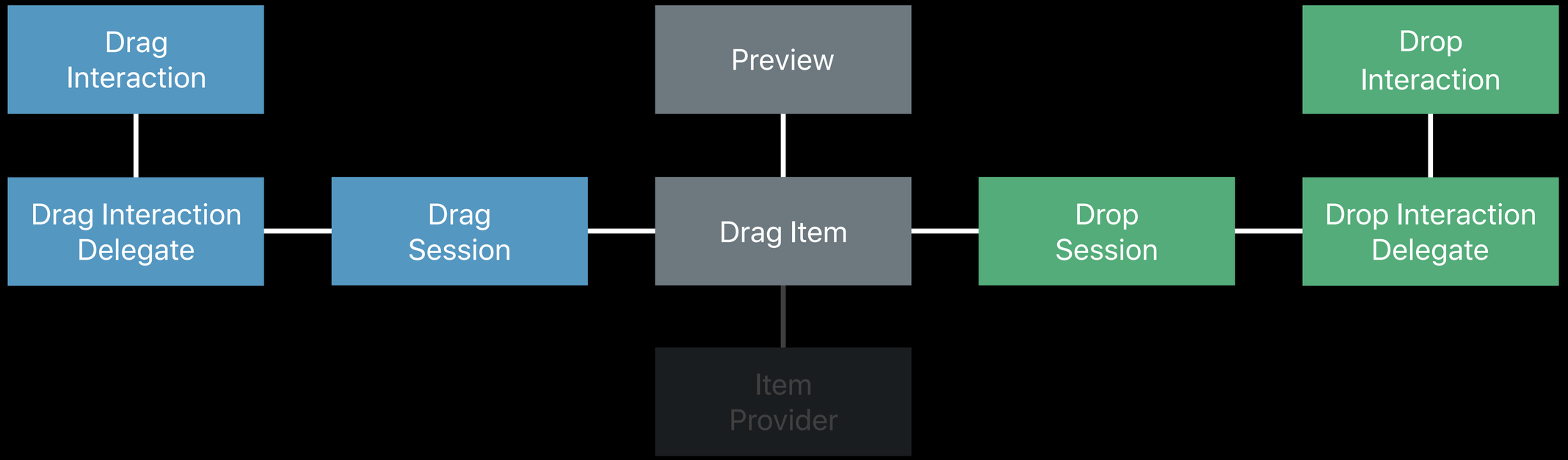
Spring load some of your controls

Dig deeper into the Drag and Drop APIs

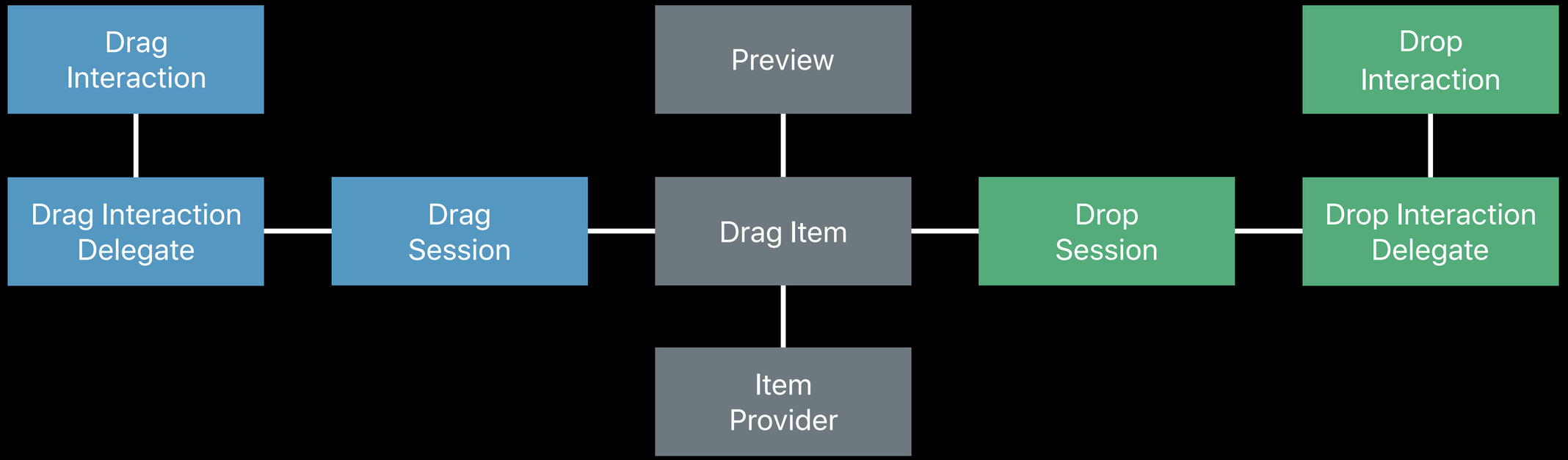
Explore the APIs



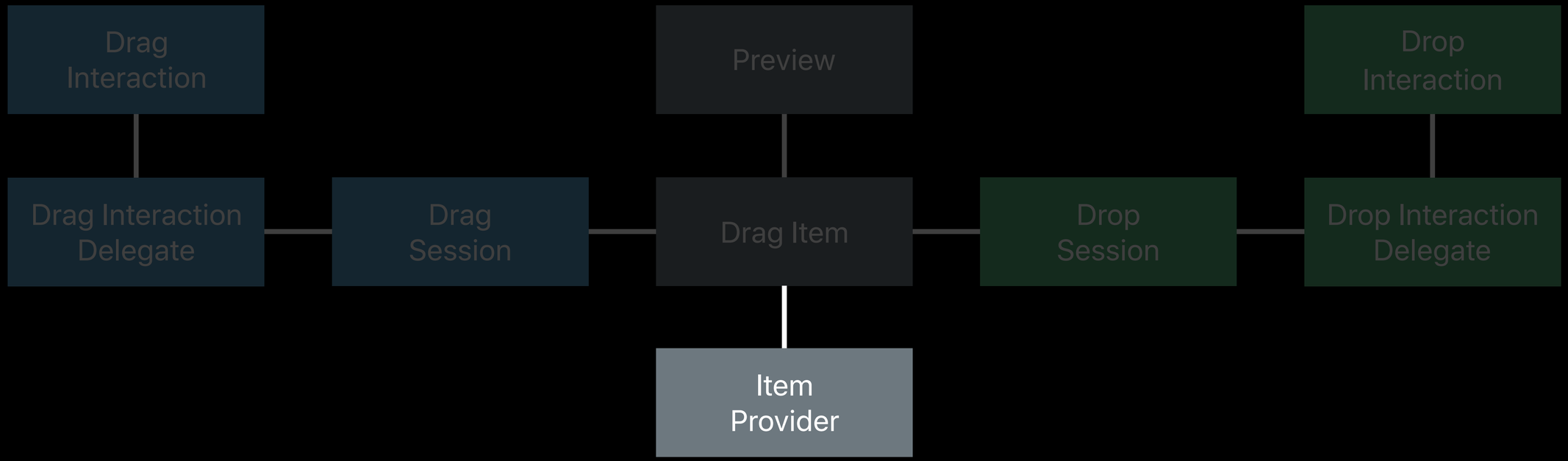
Explore the APIs



Explore the APIs



Explore the APIs

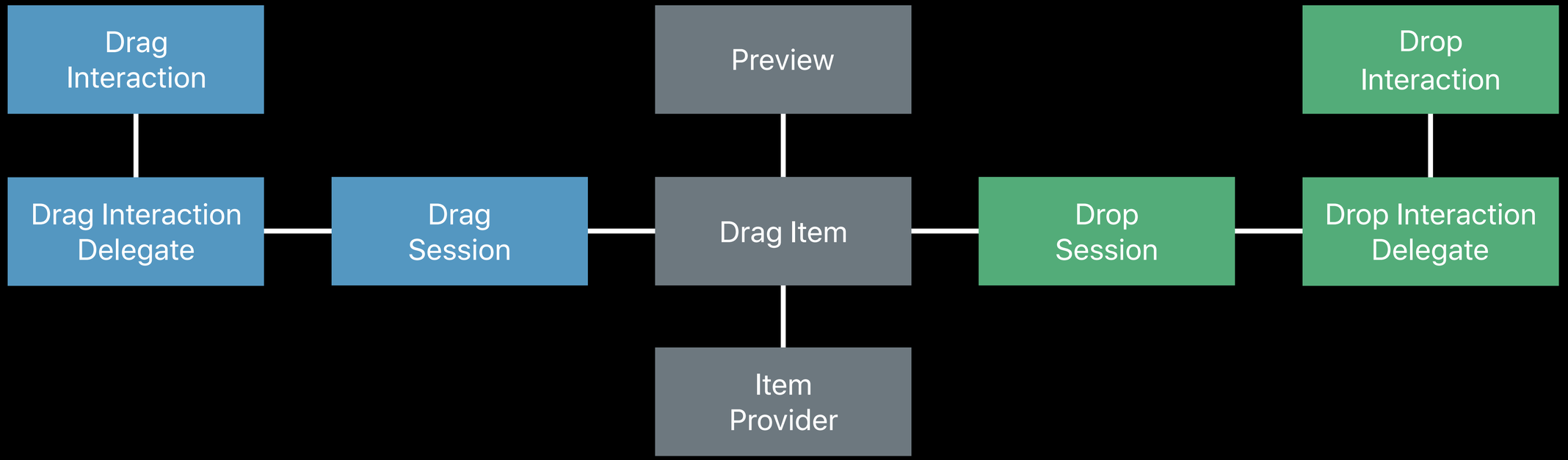


Data Delivery with Drag and Drop

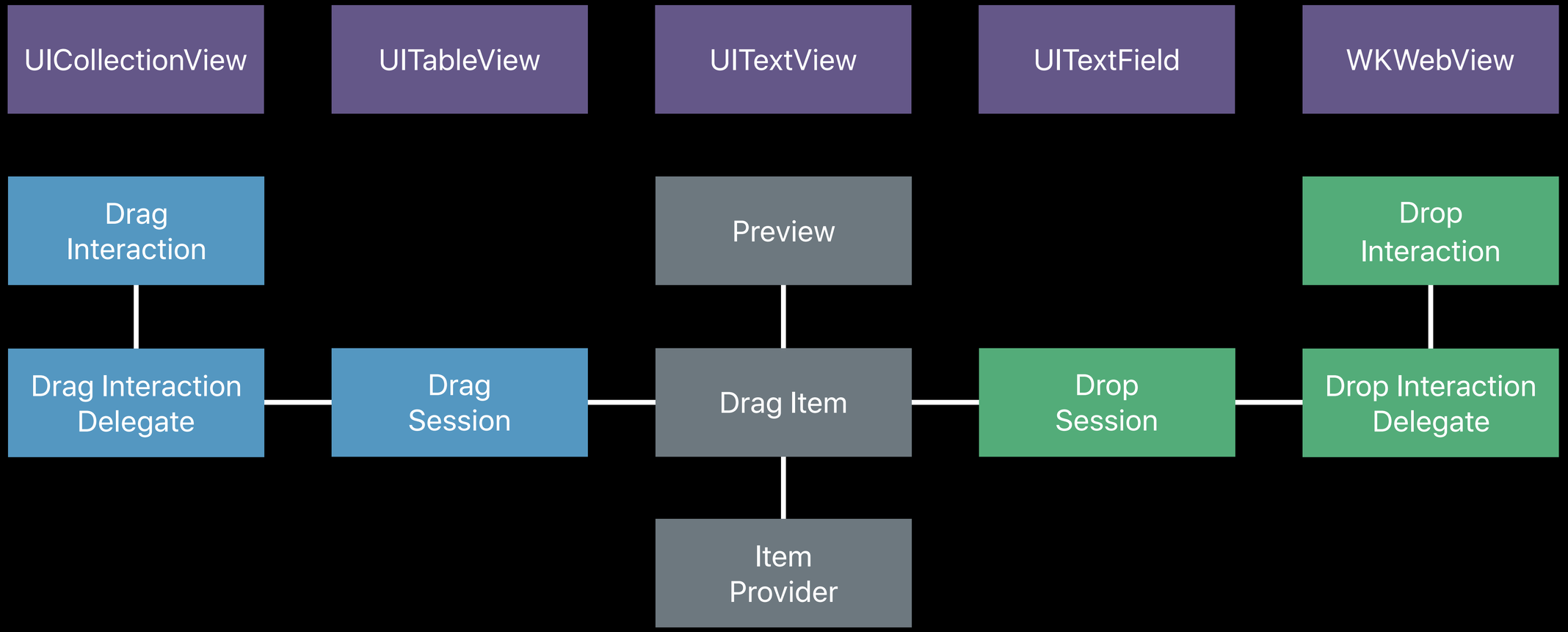
Hall 2

Thursday 10:00AM

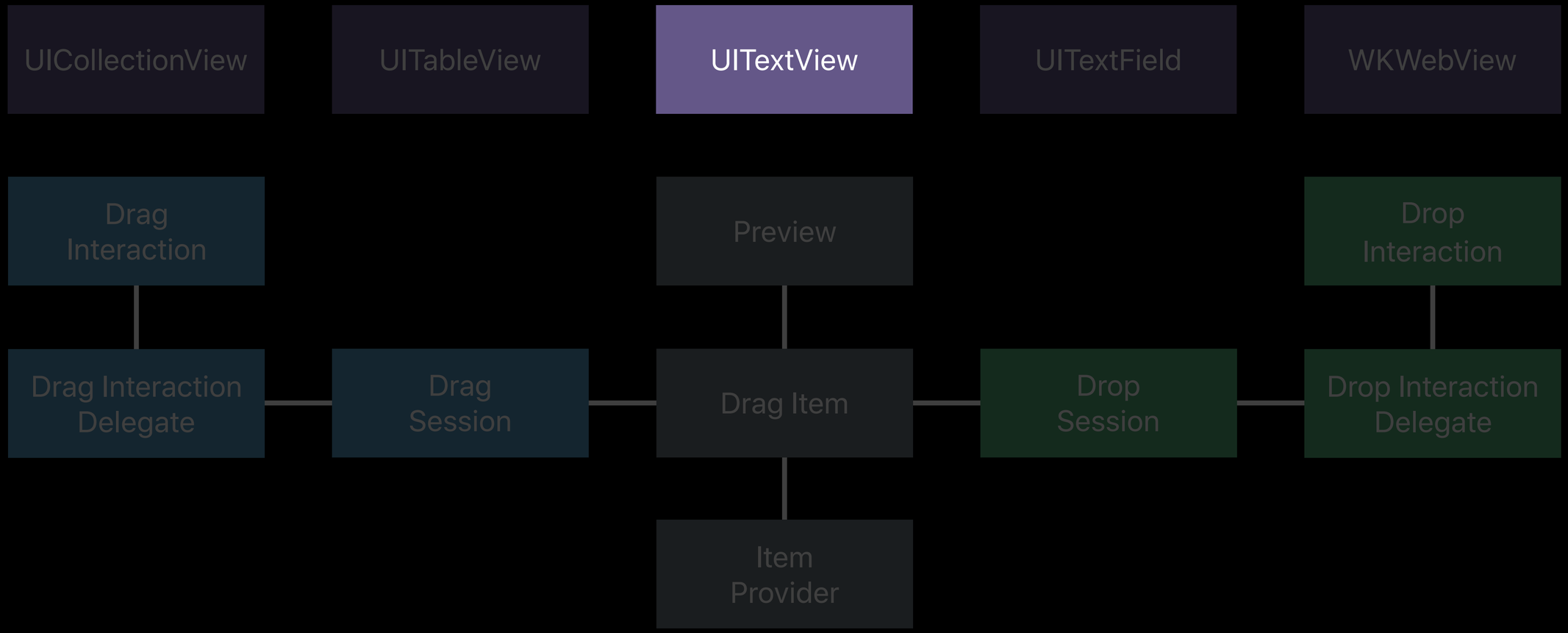
Explore the APIs



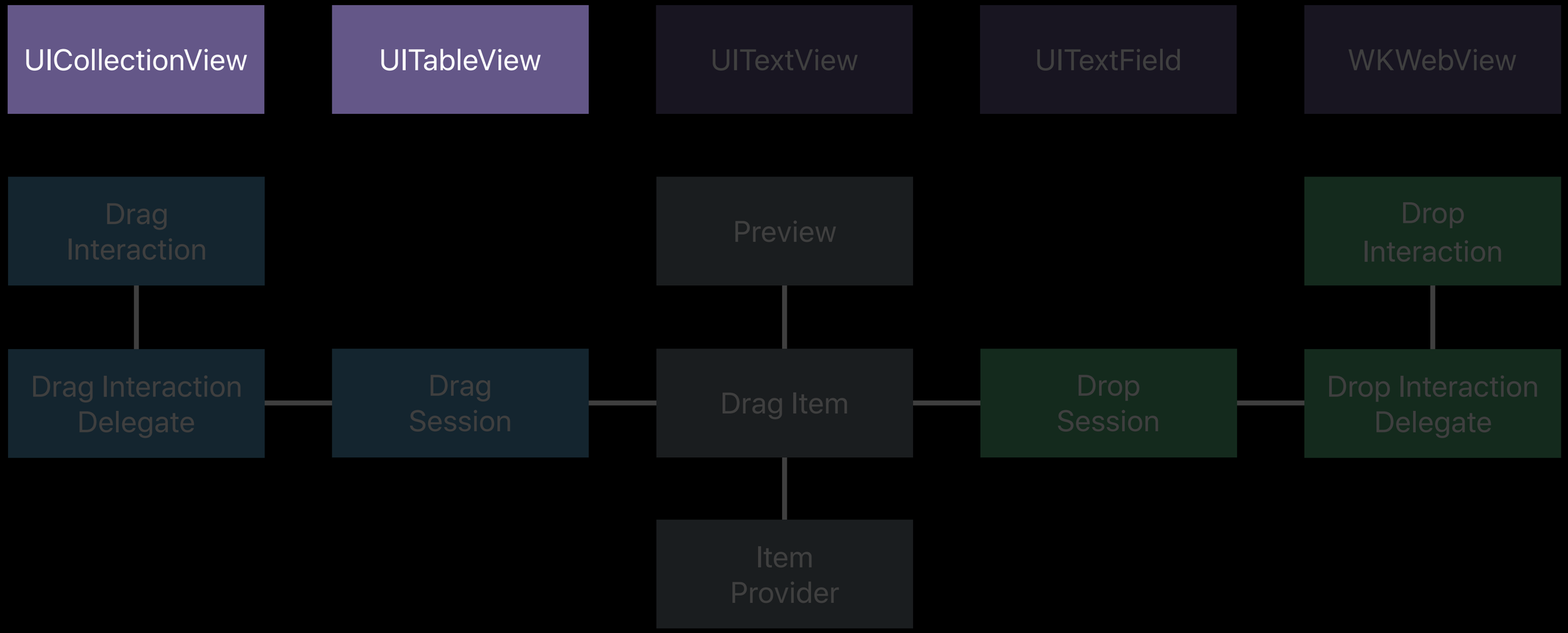
Explore the APIs



Explore the APIs



Explore the APIs



Drag and Drop with Collection and Table View

Hall 2

Thursday 9:00AM

More Information

https://developer.apple.com/wwdc17/203

Related Sessions

Mastering Drag and Drop	Executive Ballroom	Wednesday 11:00AM
Data Delivery with Drag and Drop	Hall 2	Thursday 10:00AM
Drag and Drop with Collection and Table View	Hall 2	Thursday 9:00AM
File Provider Enhancements	Hall 3	Friday 11:00AM
What's New in Core Spotlight for iOS and macOS	Grand Ballroom B	Thursday 4:10PM

Labs

UIKit and Drag and Drop Lab	Technology Lab C	Tues 1:50–4:10PM
Cocoa Touch Lab	Technology Lab I	Wed 3:10–6:00PM
UIKit and Collection View Lab	Technology Lab B	Thur 11:00AM–12:30PM
Cocoa Touch and Haptics Lab	Technology Lab C	Fri 12:00–1:50PM


WWDC17