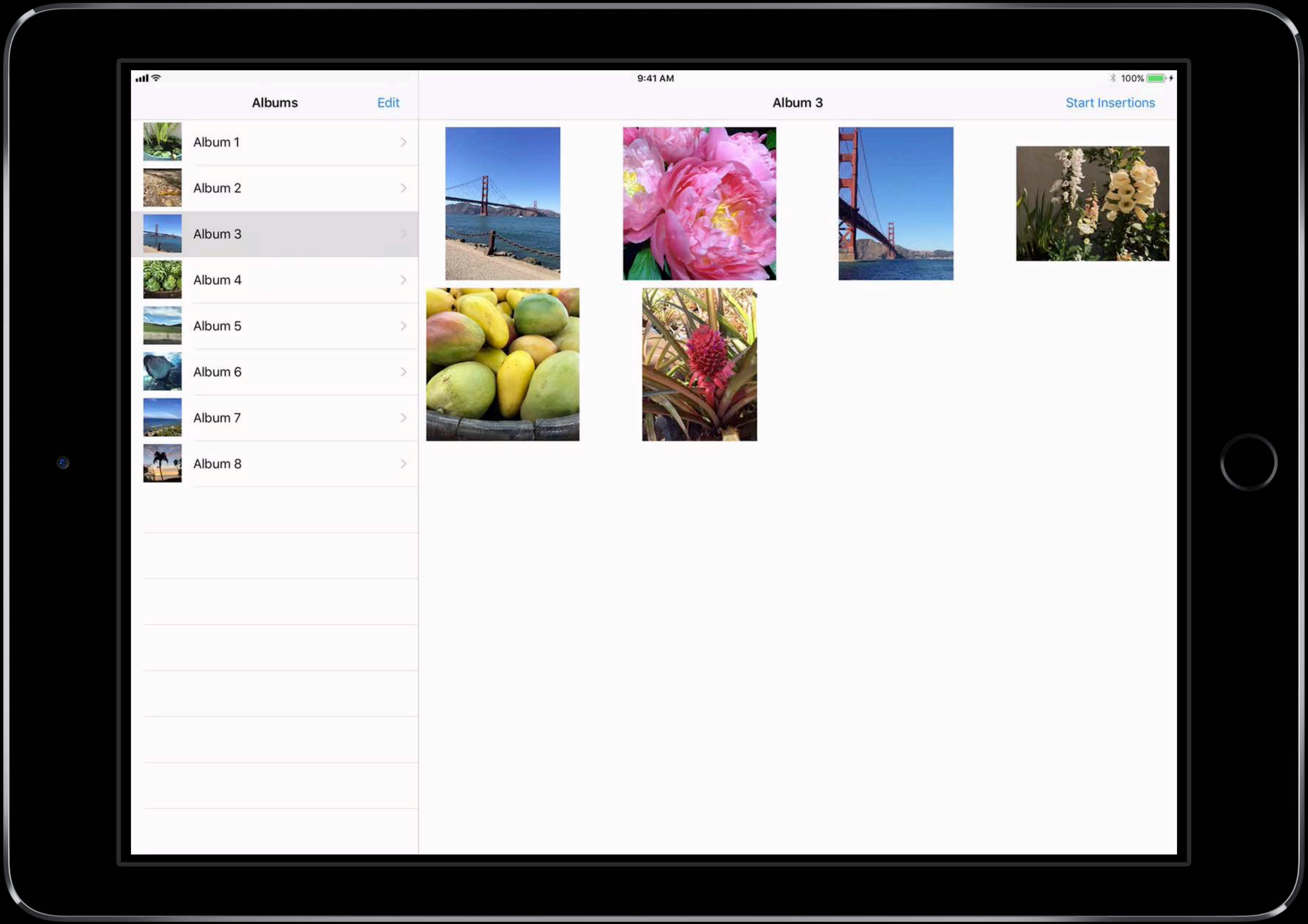
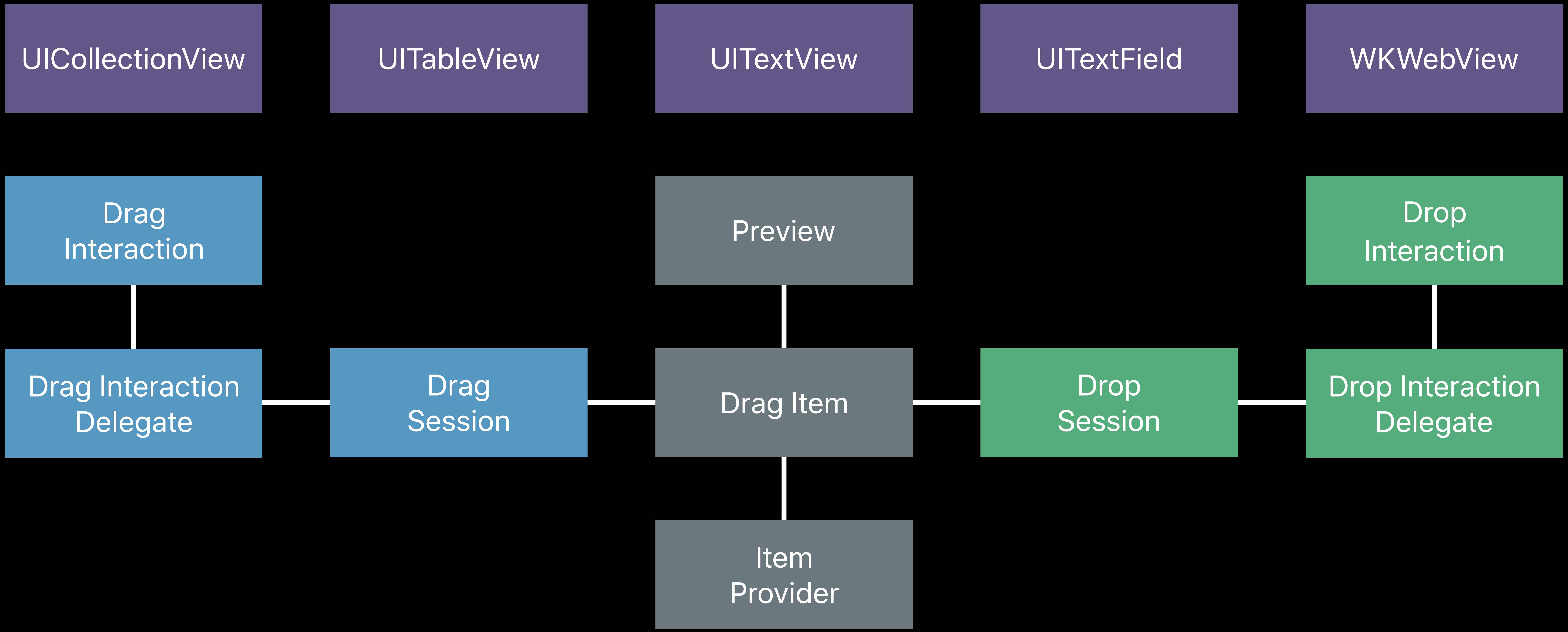
# Drag and Drop with Collection and Table View

Session 223

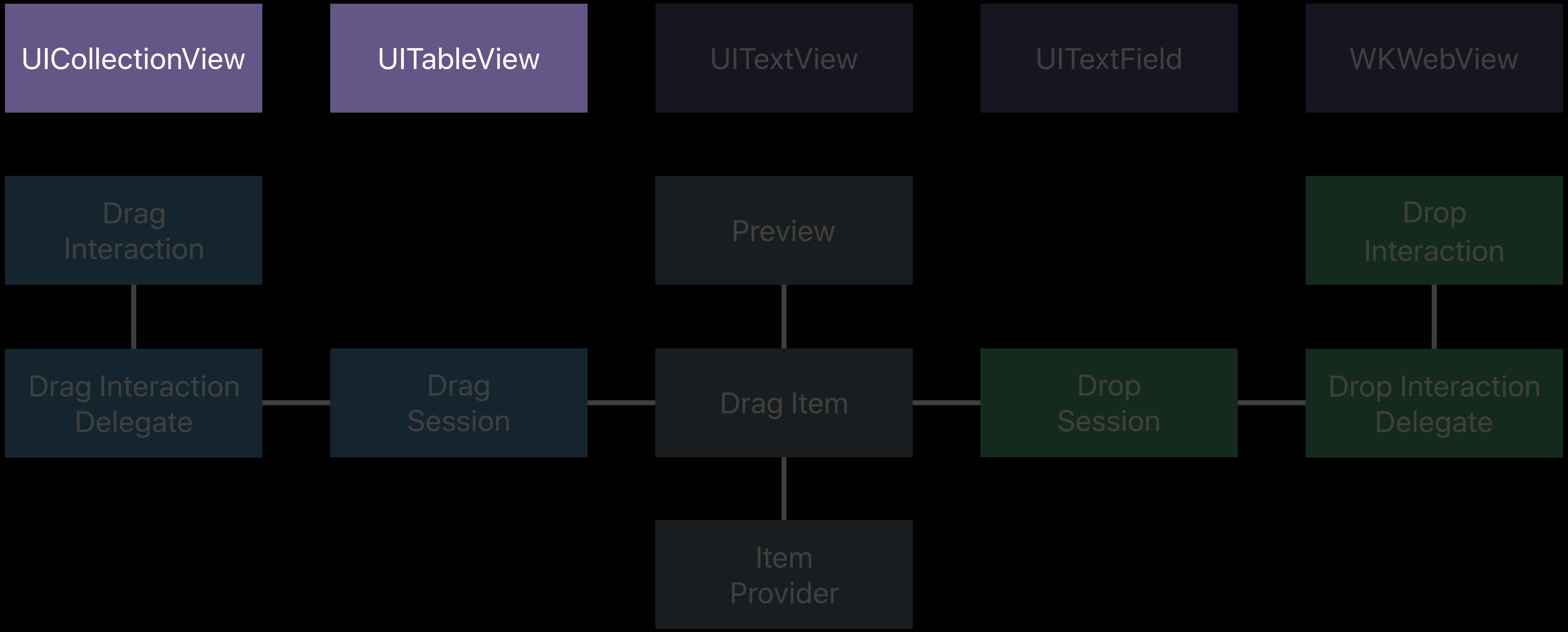
Tyler Fox, UIKit Engineer
Mohammed Jisrawi, iOS Engineer
Steve Breen, UIKit Engineer


9:41 AM
100%
Albums
Edit
Album 1
Album 2
Album 3
Album 4
Album 5
Album 6
Album 7
Album 8
Album 3
Start Insertions

# Drag and Drop API



# Drag and Drop API



## Drag and Drop API

Collection and Table View

## Drag and Drop API

Collection and Table View

Focused around cells and index paths

## Drag and Drop API
### Collection and Table View

Focused around cells and index paths

Fluid animations

# Drag and Drop API
Collection and Table View

Focused around cells and index paths

Fluid animations

Asynchronous data loading

## Drag and Drop API
Collection and Table View

Focused around cells and index paths

Fluid animations

Asynchronous data loading

Consistent API for both

## Agenda

## Agenda

Basics

## Agenda

Basics

Perfecting drops

## Agenda

Basics

Perfecting drops

Final touches

## Basics

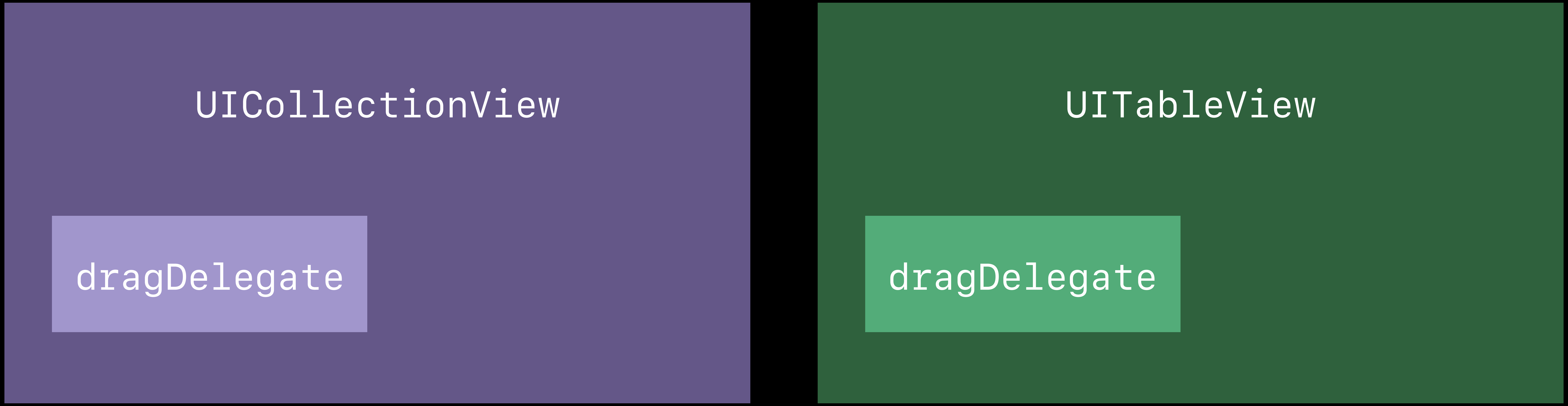
Mohammed Jisrawi, iOS Engineer

## Drag and Drop Delegates

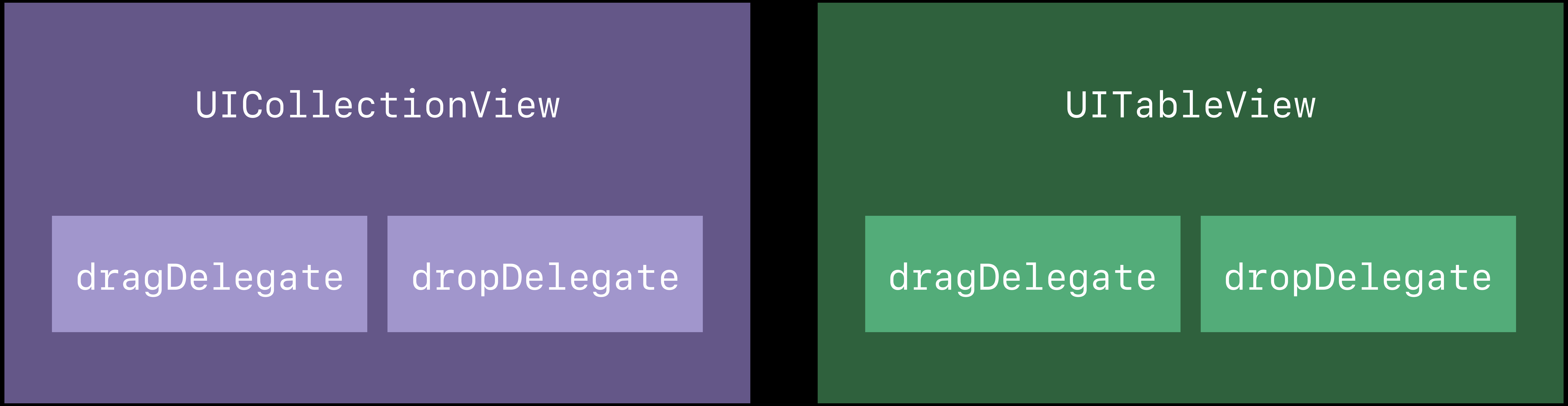
`UICollectionView`

`UITableView`

# Drag and Drop Delegates



# Drag and Drop Delegates



## Beginning a Drag

## Beginning a Drag

One required method on `dragDelegate`

## Beginning a Drag

One required method on `dragDelegate`

```
func collectionView(_: UICollectionView,
                    itemsForBeginning: UIDragSession,
                    at: IndexPath) -> [UIDragItem]
```

## Beginning a Drag

One required method on `dragDelegate`

```
func collectionView(_: UICollectionView,
                    itemsForBeginning: UIDragSession,
                    at: IndexPath) -> [UIDragItem]
```

Return an empty array to prevent the drag

## Adding Items to a Drag Session

## Adding Items to a Drag Session

Opt-in via optional method on `dragDelegate`

## Adding Items to a Drag Session

Opt-in via optional method on `dragDelegate`

```
func collectionView(_: UICollectionView,
                    itemsForAddingTo: UIDragSession,
                    at: IndexPath,
                    point: CGPoint) -> [UIDragItem]
```

## Adding Items to a Drag Session

Opt-in via optional method on `dragDelegate`

```
func collectionView(_: UICollectionView,
                    itemsForAddingTo: UIDragSession,
                    at: IndexPath,
                    point: CGPoint) -> [UIDragItem]
```

Return an empty array to handle the tap normally

## Accepting a Drop

## Accepting a Drop

One required method on `dropDelegate`

## Accepting a Drop

One required method on `dropDelegate`

```
func collectionView(_: UICollectionView,
                    performDropWith: UICollectionViewDropCoordinator)
```

## Accepting a Drop

One required method on `dropDelegate`

```
func collectionView(_: UICollectionView,
                    performDropWith: UICollectionViewDropCoordinator)
```

Drop coordinator

## Accepting a Drop

One required method on `dropDelegate`

```
func collectionView(_: UICollectionView,
                    performDropWith: UICollectionViewDropCoordinator)
```

Drop coordinator

- Access dropped items

## Accepting a Drop

One required method on `dropDelegate`

```
func collectionView(_: UICollectionView,
                    performDropWith: UICollectionViewDropCoordinator)
```

Drop coordinator

- Access dropped items
- Update collection/table view

## Accepting a Drop

One required method on `dropDelegate`

```
func collectionView(_: UICollectionView,
                    performDropWith: UICollectionViewDropCoordinator)
```

Drop coordinator

- Access dropped items
- Update collection/table view
- Specify animations

## *Demo*

Drag and Drop basics

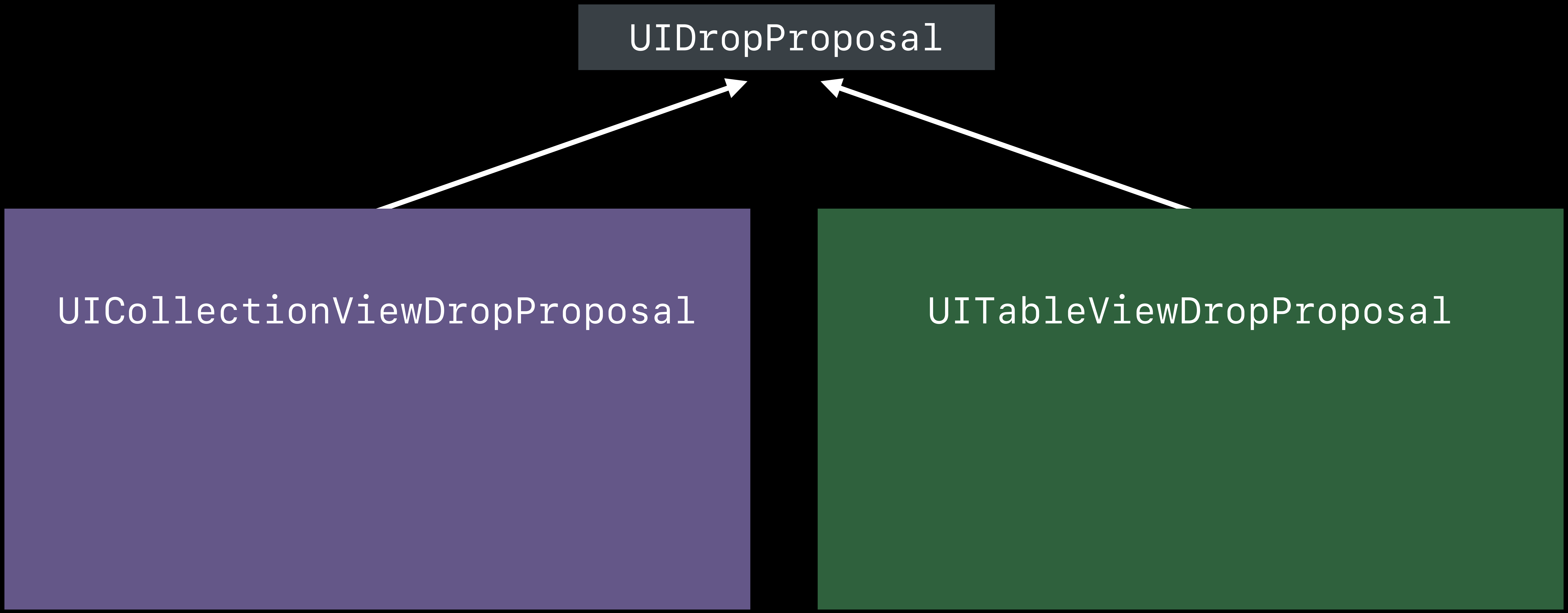
## Perfecting Drops

Tyler Fox, UIKit Engineer

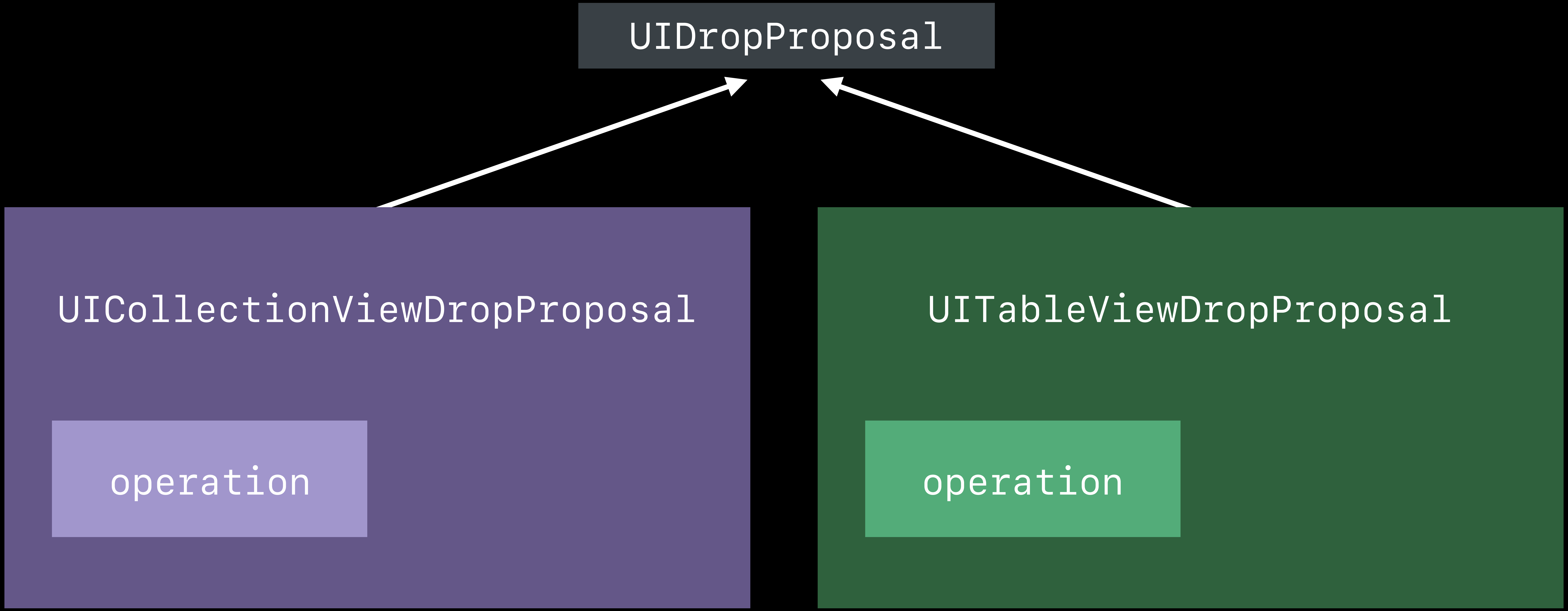
## Drop Proposal

How you want to handle the drop

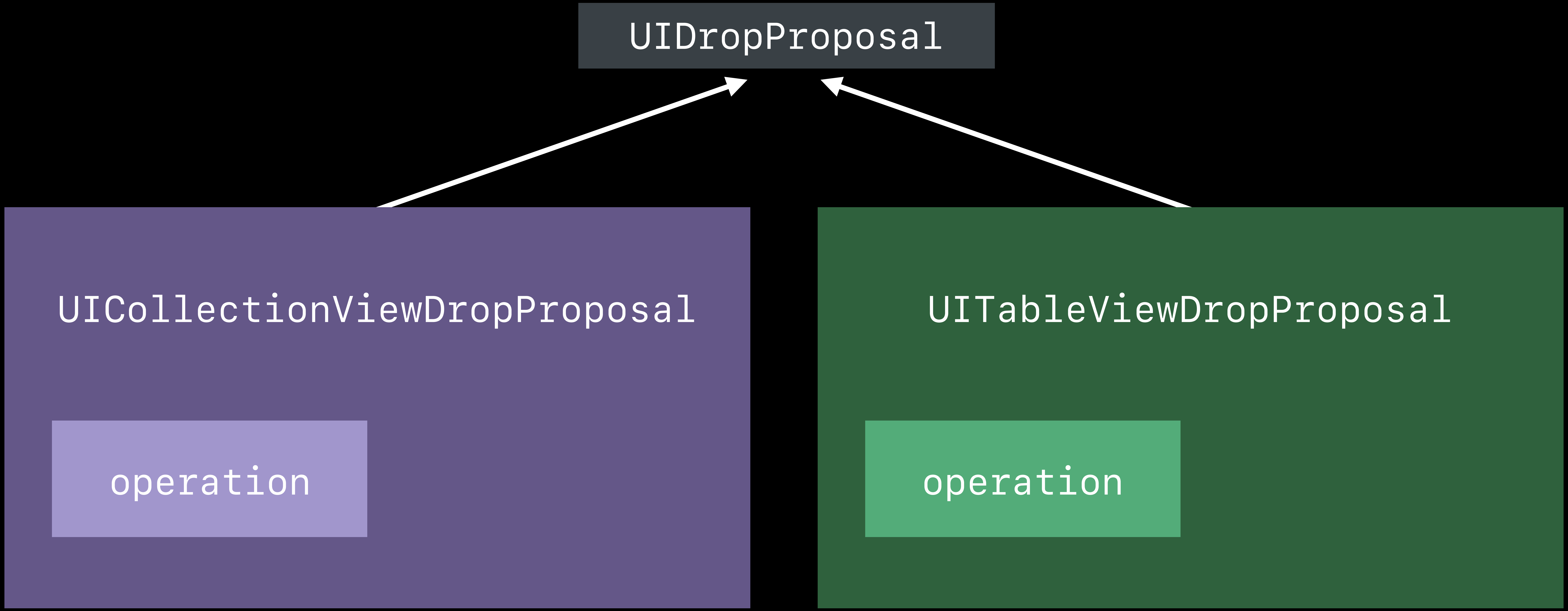
## Drop Proposal



## Drop Proposal

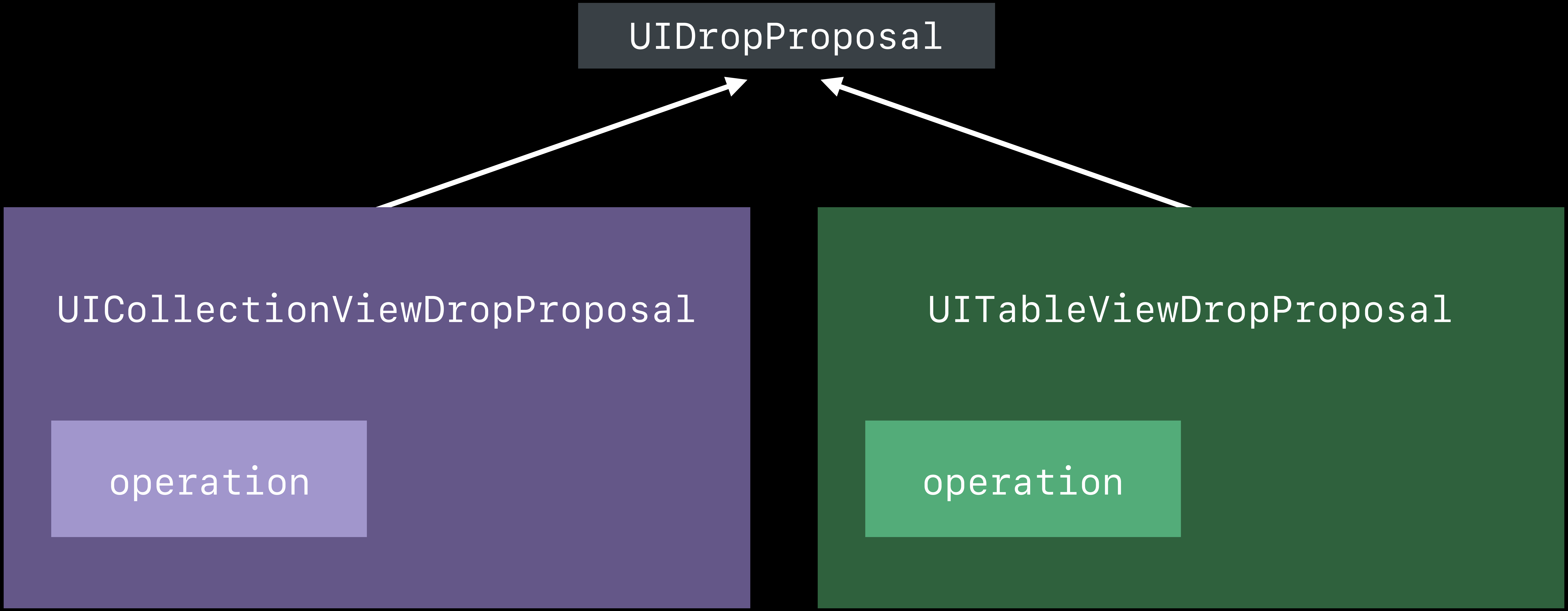


## Drop Proposal

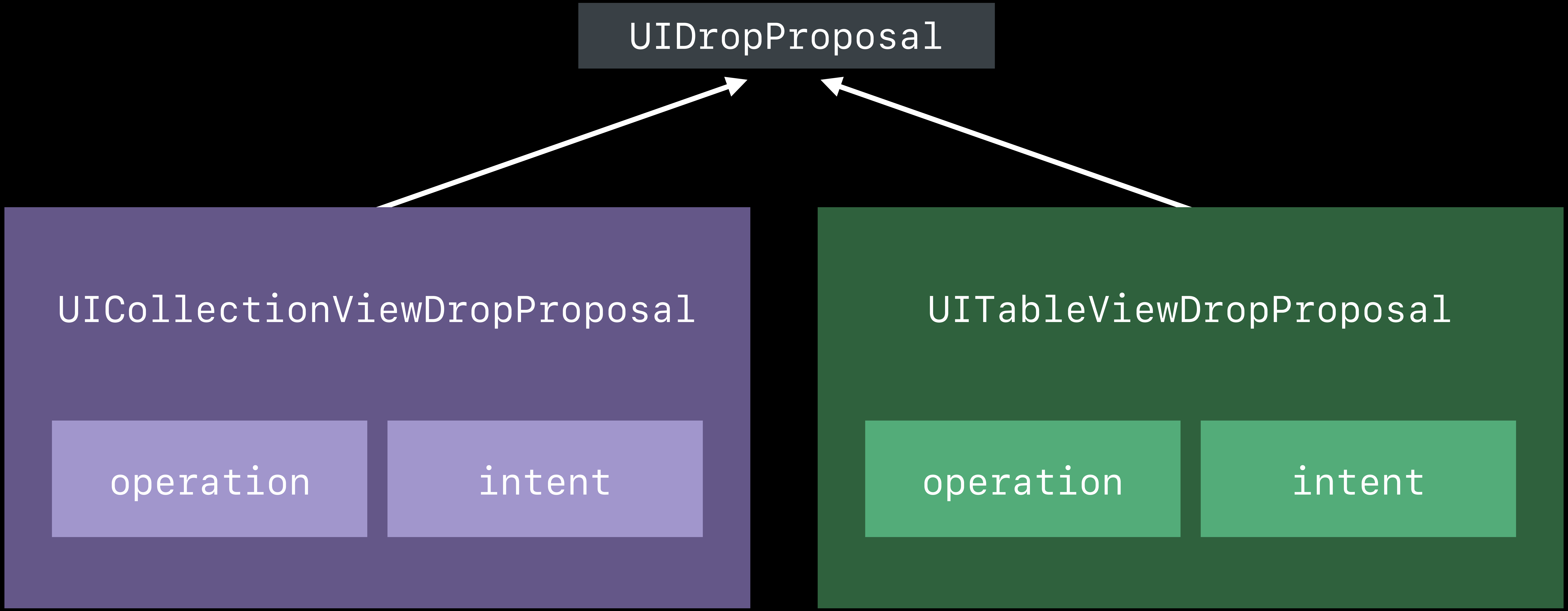


.copy .move .cancel .forbidden

## Drop Proposal



## Drop Proposal

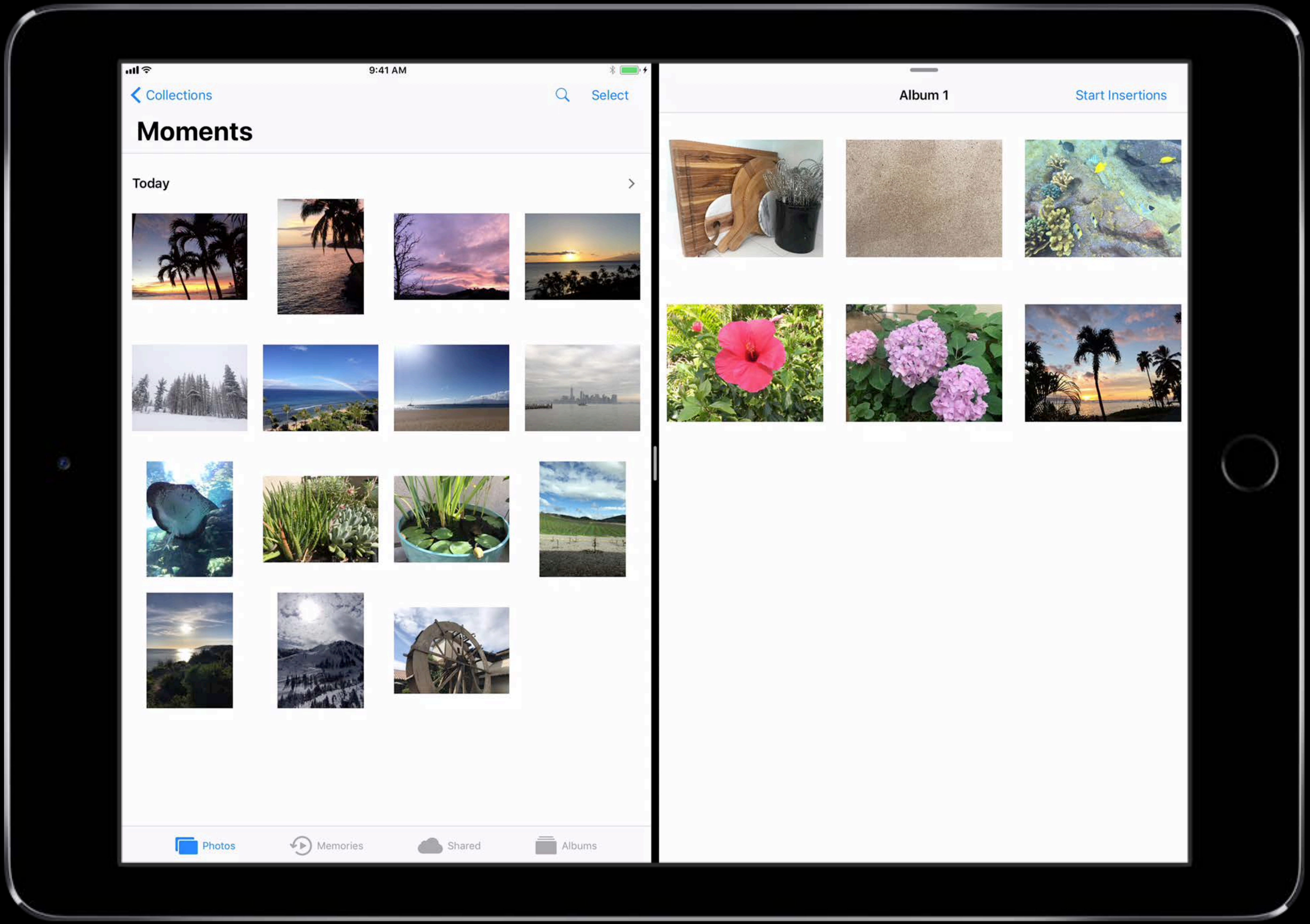


## Drop Intent

Additional information for collection and table view

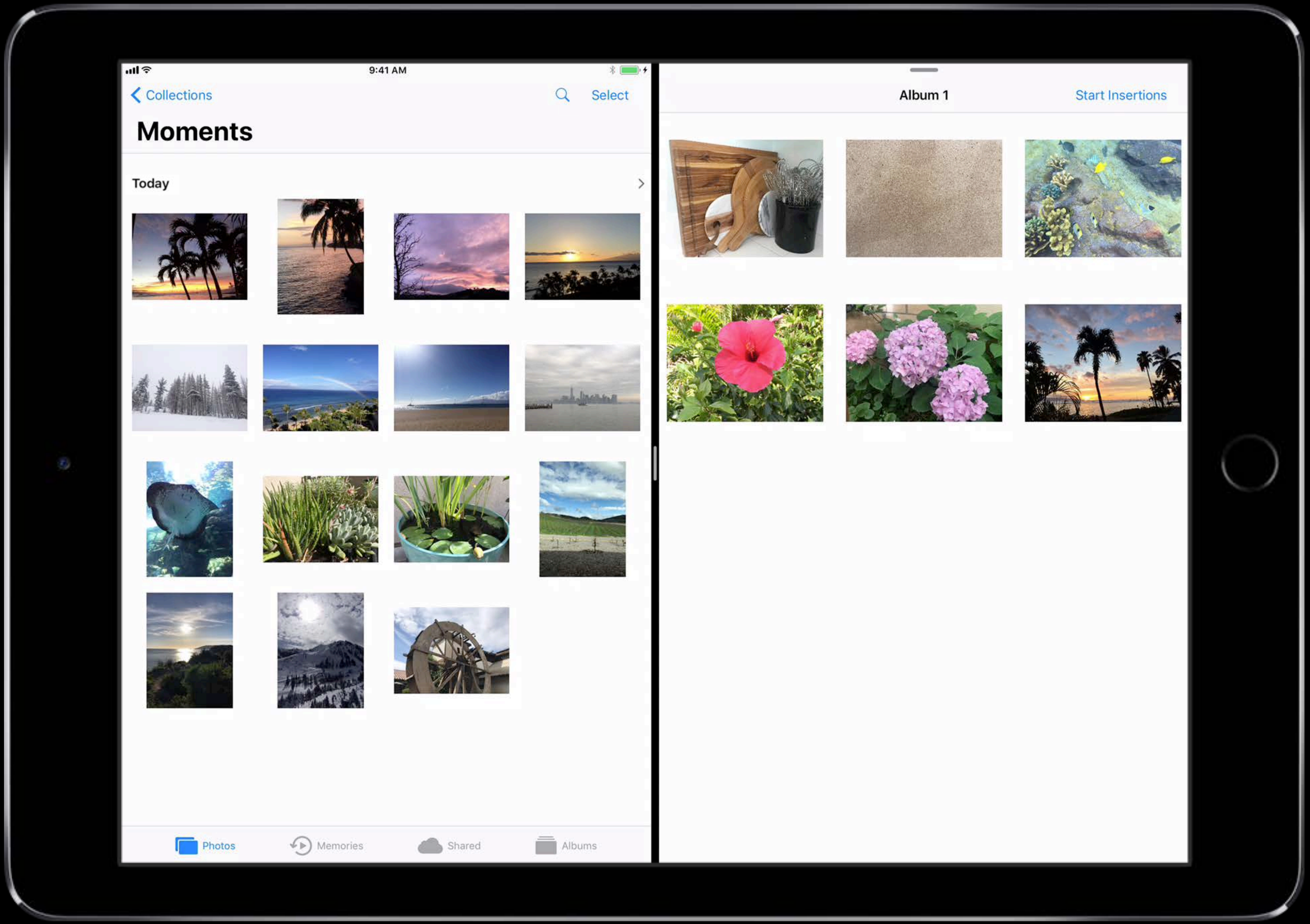
## Drop Intent

`.unspecified`



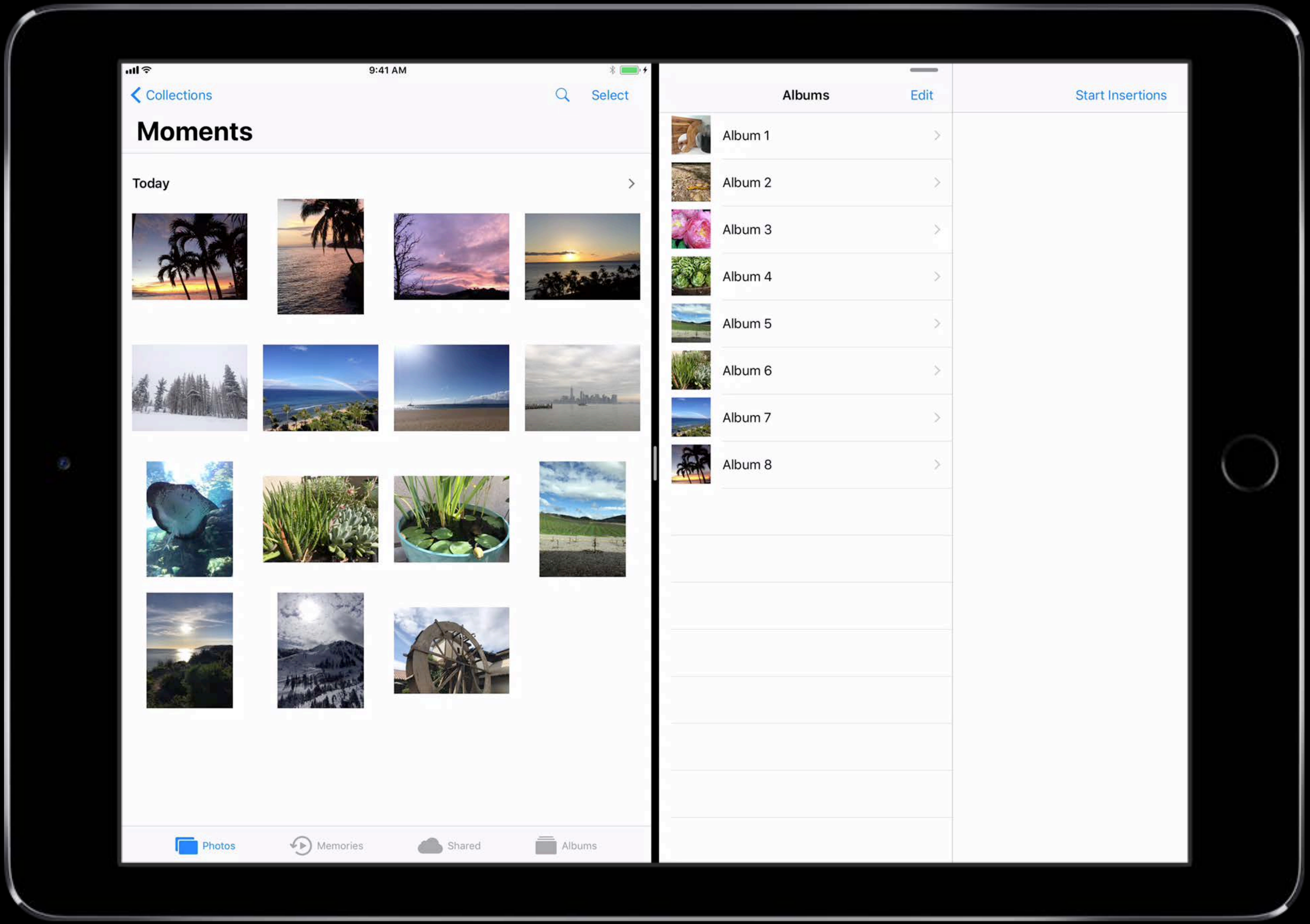
# Drop Intent

`.insertAtDestinationIndexPath`



## Drop Intent

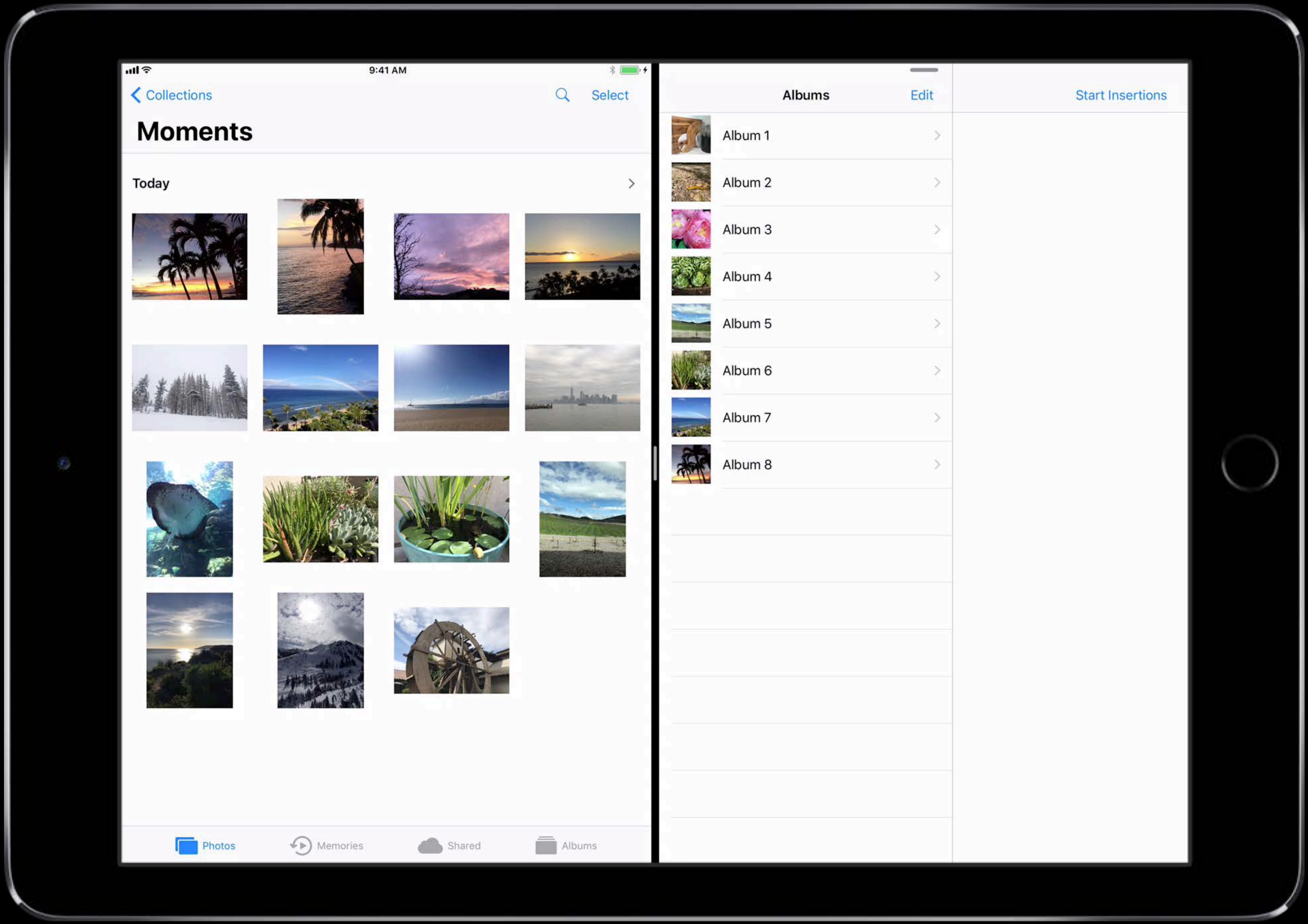
`.insertIntoDestinationIndexPath`



# Drop Intent

## Additional value for table view

`.automatic`



```swift
// Providing a Drop Proposal


func collectionView(_ collectionView: UICollectionView, dropSessionDidUpdate session:
  UIDropSession, withDestinationIndexPath destinationIndexPath: IndexPath?) ->
  UICollectionViewDropProposal {








}
```

```swift
// Providing a Drop Proposal


func collectionView(_ collectionView: UICollectionView, dropSessionDidUpdate session:
  UIDropSession, withDestinationIndexPath destinationIndexPath: IndexPath?) ->
  UICollectionViewDropProposal {







}
```

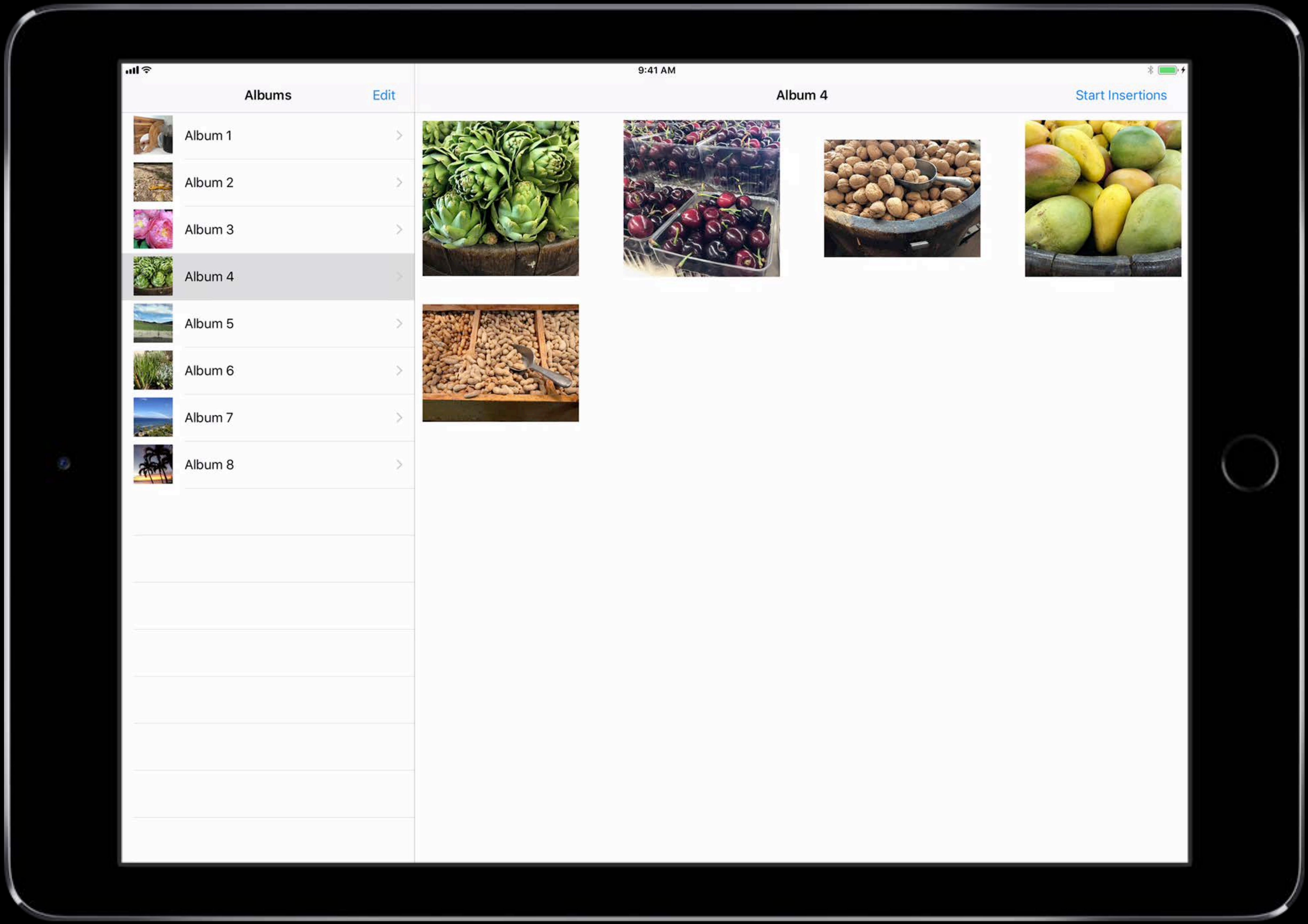
```swift
// Providing a Drop Proposal


func collectionView(_ collectionView: UICollectionView, dropSessionDidUpdate session:
  UIDropSession, withDestinationIndexPath destinationIndexPath: IndexPath?) ->
  UICollectionViewDropProposal {
    if session.localDragSession != nil {
        return UICollectionViewDropProposal(operation: .move,
                                            intent: .insertAtDestinationIndexPath)
    } else {
        return UICollectionViewDropProposal(operation: .copy,
                                            intent: .insertAtDestinationIndexPath)
    }
}
```

## Drop Animations

Set up animations using the drop coordinator

# Drop Animations

Drop to an item/row



```
// Drop to a Newly Inserted Item

func collectionView(_ collectionView: UICollectionView, performDropWith coordinator:
  UICollectionViewDropCoordinator) {
    guard let destinationIndexPath = coordinator.destinationIndexPath,
          let dragItem = coordinator.items.first?.dragItem,
          let image = dragItem.localObject as? UIImage
    else { return }







}
```

```swift
// Drop to a Newly Inserted Item

func collectionView(_ collectionView: UICollectionView, performDropWith coordinator:
  UICollectionViewDropCoordinator) {
    guard let destinationIndexPath = coordinator.destinationIndexPath,
          let dragItem = coordinator.items.first?.dragItem,
          let image = dragItem.localObject as? UIImage
    else { return }




}
```

```swift
// Drop to a Newly Inserted Item

func collectionView(_ collectionView: UICollectionView, performDropWith coordinator:
  UICollectionViewDropCoordinator) {
    guard let destinationIndexPath = coordinator.destinationIndexPath,
          let dragItem = coordinator.items.first?.dragItem,
          let image = dragItem.localObject as? UIImage
    else { return }

    collectionView.performBatchUpdates({
        self.imagesArray.insert(image, at: destinationIndexPath.item)
        collectionView.insertItems(at: [destinationIndexPath])
    })


}
```

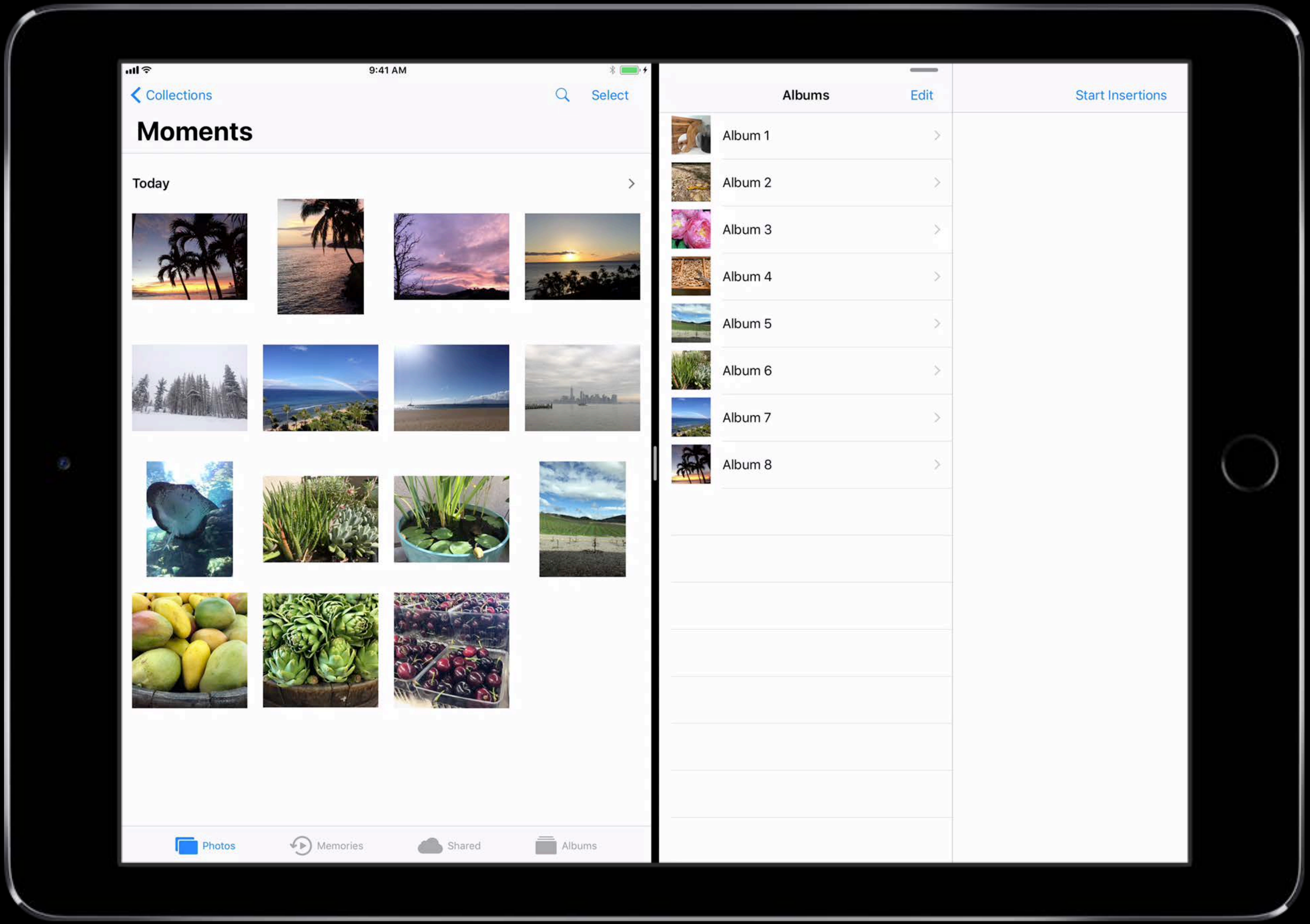
```swift
// Drop to a Newly Inserted Item

func collectionView(_ collectionView: UICollectionView, performDropWith coordinator:
  UICollectionViewDropCoordinator) {
    guard let destinationIndexPath = coordinator.destinationIndexPath,
          let dragItem = coordinator.items.first?.dragItem,
          let image = dragItem.localObject as? UIImage
    else { return }

    collectionView.performBatchUpdates({
        self.imagesArray.insert(image, at: destinationIndexPath.item)
        collectionView.insertItems(at: [destinationIndexPath])
    })


}
```

```swift
// Drop to a Newly Inserted Item

func collectionView(_ collectionView: UICollectionView, performDropWith coordinator:
  UICollectionViewDropCoordinator) {
    guard let destinationIndexPath = coordinator.destinationIndexPath,
          let dragItem = coordinator.items.first?.dragItem,
          let image = dragItem.localObject as? UIImage
    else { return }

    collectionView.performBatchUpdates({
        self.imagesArray.insert(image, at: destinationIndexPath.item)
        collectionView.insertItems(at: [destinationIndexPath])
    })

    coordinator.drop(dragItem, toItemAt: destinationIndexPath)
}
```

```swift
// Drop to a Newly Inserted Item

func collectionView(_ collectionView: UICollectionView, performDropWith coordinator:
  UICollectionViewDropCoordinator) {
    guard let destinationIndexPath = coordinator.destinationIndexPath,
          let dragItem = coordinator.items.first?.dragItem,
          let image = dragItem.localObject as? UIImage
    else { return }

    collectionView.performBatchUpdates({
        self.imagesArray.insert(image, at: destinationIndexPath.item)
        collectionView.insertItems(at: [destinationIndexPath])
    })

    coordinator.drop(dragItem, toItemAt: destinationIndexPath)
}
```

# Drop Animations

Drop into an item/row



```swift
// Drop Into a Row

func tableView(_ tableView: UITableView, performDropWith coordinator:
  UITableViewDropCoordinator) {
    guard let destinationIndexPath = coordinator.destinationIndexPath,
          let dragItem = coordinator.items.first?.dragItem,
          let image = dragItem.localObject as? UIImage
    else { return }








}
```

```swift
// Drop Into a Row

func tableView(_ tableView: UITableView, performDropWith coordinator:
  UITableViewDropCoordinator) {
    guard let destinationIndexPath = coordinator.destinationIndexPath,
          let dragItem = coordinator.items.first?.dragItem,
          let image = dragItem.localObject as? UIImage
    else { return }









}
```

```
// Drop Into a Row

func tableView(_ tableView: UITableView, performDropWith coordinator:
  UITableViewDropCoordinator) {
    guard let destinationIndexPath = coordinator.destinationIndexPath,
          let dragItem = coordinator.items.first?.dragItem,
          let image = dragItem.localObject as? UIImage
    else { return }
    let photoAlbumIndex = destinationIndexPath.row
    guard photoAlbumIndex < self.albumsArray.count else { return }
    self.add(image: image, toAlbumAt: photoAlbumIndex)






}
```

```swift
// Drop Into a Row

func tableView(_ tableView: UITableView, performDropWith coordinator:
  UITableViewDropCoordinator) {
    guard let destinationIndexPath = coordinator.destinationIndexPath,
          let dragItem = coordinator.items.first?.dragItem,
          let image = dragItem.localObject as? UIImage
    else { return }
    let photoAlbumIndex = destinationIndexPath.row
    guard photoAlbumIndex < self.albumsArray.count else { return }
    self.add(image: image, toAlbumAt: photoAlbumIndex)





}
```

```swift
// Drop Into a Row

func tableView(_ tableView: UITableView, performDropWith coordinator:
  UITableViewDropCoordinator) {
    guard let destinationIndexPath = coordinator.destinationIndexPath,
          let dragItem = coordinator.items.first?.dragItem,
          let image = dragItem.localObject as? UIImage
    else { return }
    let photoAlbumIndex = destinationIndexPath.row
    guard photoAlbumIndex < self.albumsArray.count else { return }
    self.add(image: image, toAlbumAt: photoAlbumIndex)

    if let cell = tableView.cellForRow(at: destinationIndexPath), let view = cell.imageView {
        let rect = cell.convert(view.bounds, from: view)
        coordinator.drop(dragItem, intoRowAt: destinationIndexPath, rect: rect)
    }
}
```

```swift
// Drop Into a Row

func tableView(_ tableView: UITableView, performDropWith coordinator:
  UITableViewDropCoordinator) {
    guard let destinationIndexPath = coordinator.destinationIndexPath,
          let dragItem = coordinator.items.first?.dragItem,
          let image = dragItem.localObject as? UIImage
    else { return }
    let photoAlbumIndex = destinationIndexPath.row
    guard photoAlbumIndex < self.albumsArray.count else { return }
    self.add(image: image, toAlbumAt: photoAlbumIndex)

    if let cell = tableView.cellForRow(at: destinationIndexPath), let view = cell.imageView {
        let rect = cell.convert(view.bounds, from: view)
        coordinator.drop(dragItem, intoRowAt: destinationIndexPath, rect: rect)
    }
}
```

## Drop Animations

## Drop Animations

Drop to an item/row

## Drop Animations

Drop to an item/row

Drop into an item/row

## Drop Animations

Drop to an item/row

Drop into an item/row

Drop to a target

- Animate to any location with a transform

## Drop Animations

Drop to an item/row

Drop into an item/row

Drop to a target

- Animate to any location with a transform

## Animating Items Before the Data Loads

Data loads asynchronously

## Animating Items Before the Data Loads

Data loads asynchronously

Bookkeeping is difficult

NEW

## Introducing Placeholders

## Placeholders

Temporary insertions in the collection/table view

## Placeholders

Temporary insertions in the collection/table view

You provide the cell, we do the bookkeeping for you

## Placeholders

Temporary insertions in the collection/table view

You provide the cell, we do the bookkeeping for you

Great experience while data loads asynchronously

```swift
// Drop to a Placeholder Item

func collectionView(_ collectionView: UICollectionView, performDropWith coordinator:
  UICollectionViewDropCoordinator) {

    guard let destinationIndexPath = coordinator.destinationIndexPath else { return }









}
```

```swift
// Drop to a Placeholder Item

func collectionView(_ collectionView: UICollectionView, performDropWith coordinator:
  UICollectionViewDropCoordinator) {

    guard let destinationIndexPath = coordinator.destinationIndexPath else { return }

    for item in coordinator.items {
        coordinator.drop(item.dragItem, toPlaceholderInsertedAt:
          destinationIndexPath, withReuseIdentifier: "PlaceholderCell") { cell in
            // Configure the placeholder cell
        }

    }

}
```

```swift
// Drop to a Placeholder Item

func collectionView(_ collectionView: UICollectionView, performDropWith coordinator:
  UICollectionViewDropCoordinator) {

    guard let destinationIndexPath = coordinator.destinationIndexPath else { return }

    for item in coordinator.items {
                                coordinator.drop(item.dragItem, toPlaceholderInsertedAt:
          destinationIndexPath, withReuseIdentifier: "PlaceholderCell") { cell in
            // Configure the placeholder cell
        }

    }

}
```

```swift
// Drop to a Placeholder Item

func collectionView(_ collectionView: UICollectionView, performDropWith coordinator:
  UICollectionViewDropCoordinator) {

    guard let destinationIndexPath = coordinator.destinationIndexPath else { return }

    for item in coordinator.items {
                                coordinator.drop(item.dragItem, toPlaceholderInsertedAt:
          destinationIndexPath, withReuseIdentifier: "PlaceholderCell") { cell in
            // Configure the placeholder cell
        }

    }

}
```

```swift
// Drop to a Placeholder Item

func collectionView(_ collectionView: UICollectionView, performDropWith coordinator:
  UICollectionViewDropCoordinator) {

    guard let destinationIndexPath = coordinator.destinationIndexPath else { return }

    for item in coordinator.items {
        let placeholderContext = coordinator.drop(item.dragItem, toPlaceholderInsertedAt:
          destinationIndexPath, withReuseIdentifier: "PlaceholderCell") { cell in
            // Configure the placeholder cell
        }

    }

}
```

## Using the Placeholder Context

## Using the Placeholder Context

Commit the insertion of a placeholder to exchange it for the final cell

## Using the Placeholder Context

Commit the insertion of a placeholder to exchange it for the final cell

Delete the placeholder if it’s no longer needed

```swift
// Using the Placeholder Context

for item in coordinator.items {
    let placeholderContext = /* insert the placeholder for this item */









}
```

```swift
// Using the Placeholder Context

for item in coordinator.items {
    let placeholderContext = /* insert the placeholder for this item */

    item.dragItem.itemProvider.loadObject(ofClass: UIImage.self) { (object, error) in
        DispatchQueue.main.async {







        }
    }
}
```

```
// Using the Placeholder Context

for item in coordinator.items {
    let placeholderContext = /* insert the placeholder for this item */

    item.dragItem.itemProvider.loadObject(ofClass: UIImage.self) { (object, error) in
        DispatchQueue.main.async {
            if let image = object as? UIImage {
                placeholderContext.commitInsertion { insertionIndexPath in
                    self.imagesArray.insert(image, at: insertionIndexPath.item)
                }
            }



        }
    }
}
```

```
// Using the Placeholder Context

for item in coordinator.items {
    let placeholderContext = /* insert the placeholder for this item */

    item.dragItem.itemProvider.loadObject(ofClass: UIImage.self) { (object, error) in
        DispatchQueue.main.async {
            if let image = object as? UIImage {
                placeholderContext.commitInsertion { insertionIndexPath in
                    self.imagesArray.insert(image, at: insertionIndexPath.item)
                }
            }



        }
    }
}
```

```swift
// Using the Placeholder Context

for item in coordinator.items {
    let placeholderContext = /* insert the placeholder for this item */

    item.dragItem.itemProvider.loadObject(ofClass: UIImage.self) { (object, error) in
        DispatchQueue.main.async {
            if let image = object as? UIImage {
                placeholderContext.commitInsertion { insertionIndexPath in
                    self.imagesArray.insert(image, at: insertionIndexPath.item)
                }
            } else {
                placeholderContext.deletePlaceholder()
            }
        }
    }
}
```

```swift
// Using the Placeholder Context

for item in coordinator.items {
    let placeholderContext = /* insert the placeholder for this item */

    item.dragItem.itemProvider.loadObject(ofClass: UIImage.self) { (object, error) in
        DispatchQueue.main.async {
            if let image = object as? UIImage {
                placeholderContext.commitInsertion { insertionIndexPath in
                    self.imagesArray.insert(image, at: insertionIndexPath.item)
                }
            } else {
                placeholderContext.deletePlaceholder()
            }
        }
    }
}
```

## Working with Placeholders

Avoid `reloadData`, use `performBatchUpdates(_:completion:)` instead

## Working with Placeholders

Avoid `reloadData`, use `performBatchUpdates(_:completion:)` instead

When placeholders exist, the table or collection view has uncommitted updates

```
var hasUncommittedUpdates: Bool { get }
```

## *Demo*

Providing drop animations and using placeholders

Mohammed Jisrawi, iOS Engineer

## Final Touches

Tyler Fox, UIKit Engineer

## Supporting Reordering

## Supporting Reordering

Implement `dropDelegate` method

```
func collectionView(_: UICollectionView,
                    dropSessionDidUpdate: UIDropSession,
                    withDestinationIndexPath: IndexPath?) -> UICollectionViewDropProposal
```

## Supporting Reordering

Implement `dropDelegate` method

```
func collectionView(_: UICollectionView,
                    dropSessionDidUpdate: UIDropSession,
                    withDestinationIndexPath: IndexPath?) -> UICollectionViewDropProposal
```

Return a drop proposal

```
return UICollectionViewDropProposal(operation: .move, intent: .insertAtDestinationIndexPath)
```

## Reordering

Table View

## Reordering

Table View

Continue to implement the existing data source method

```
func tableView(_: UITableView, moveRowAt: IndexPath, to: IndexPath)
```

## Reordering

Table View

Continue to implement the existing data source method

```
func tableView(_: UITableView, moveRowAt: IndexPath, to: IndexPath)
```

Called instead of `tableView(_: UITableView, performDropWith: UITableViewDropCoordinator)`

## Reordering

Collection View

## Reordering

### Collection View

Provide a `dragDelegate` and `dropDelegate`

## Reordering
Collection View

Provide a `dragDelegate` and `dropDelegate`

Inside `collectionView(_: UICollectionView, performDropWith: UICollectionViewDropCoordinator)`

## Reordering

Collection View

Provide a `dragDelegate` and `dropDelegate`

Inside `collectionView(_: UICollectionView, performDropWith: UICollectionViewDropCoordinator)`

- Delete from `UICollectionViewDropItem.sourceIndexPath`

## Reordering

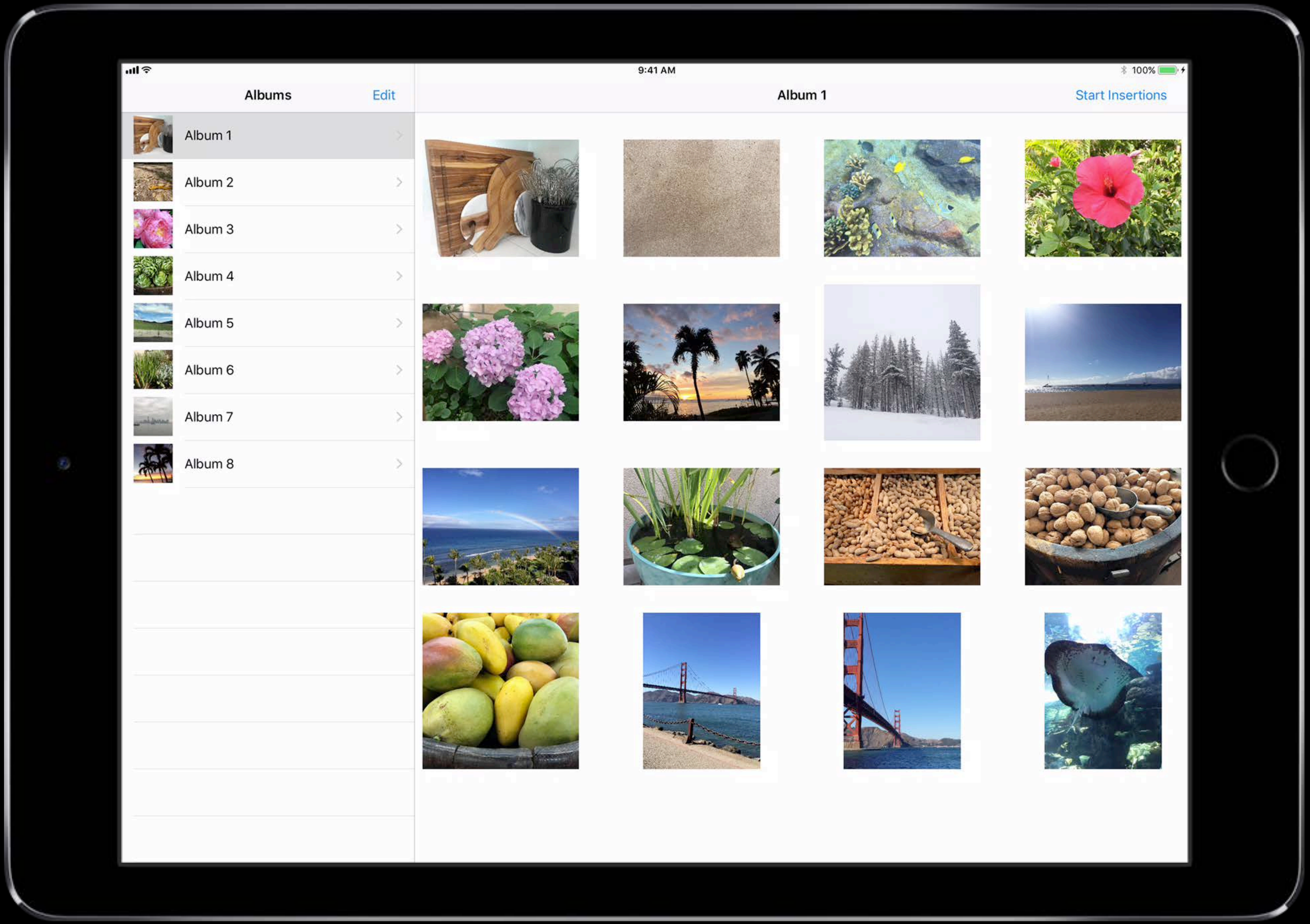
Collection View

Provide a `dragDelegate` and `dropDelegate`

Inside `collectionView(_: UICollectionView, performDropWith: UICollectionViewDropCoordinator)`

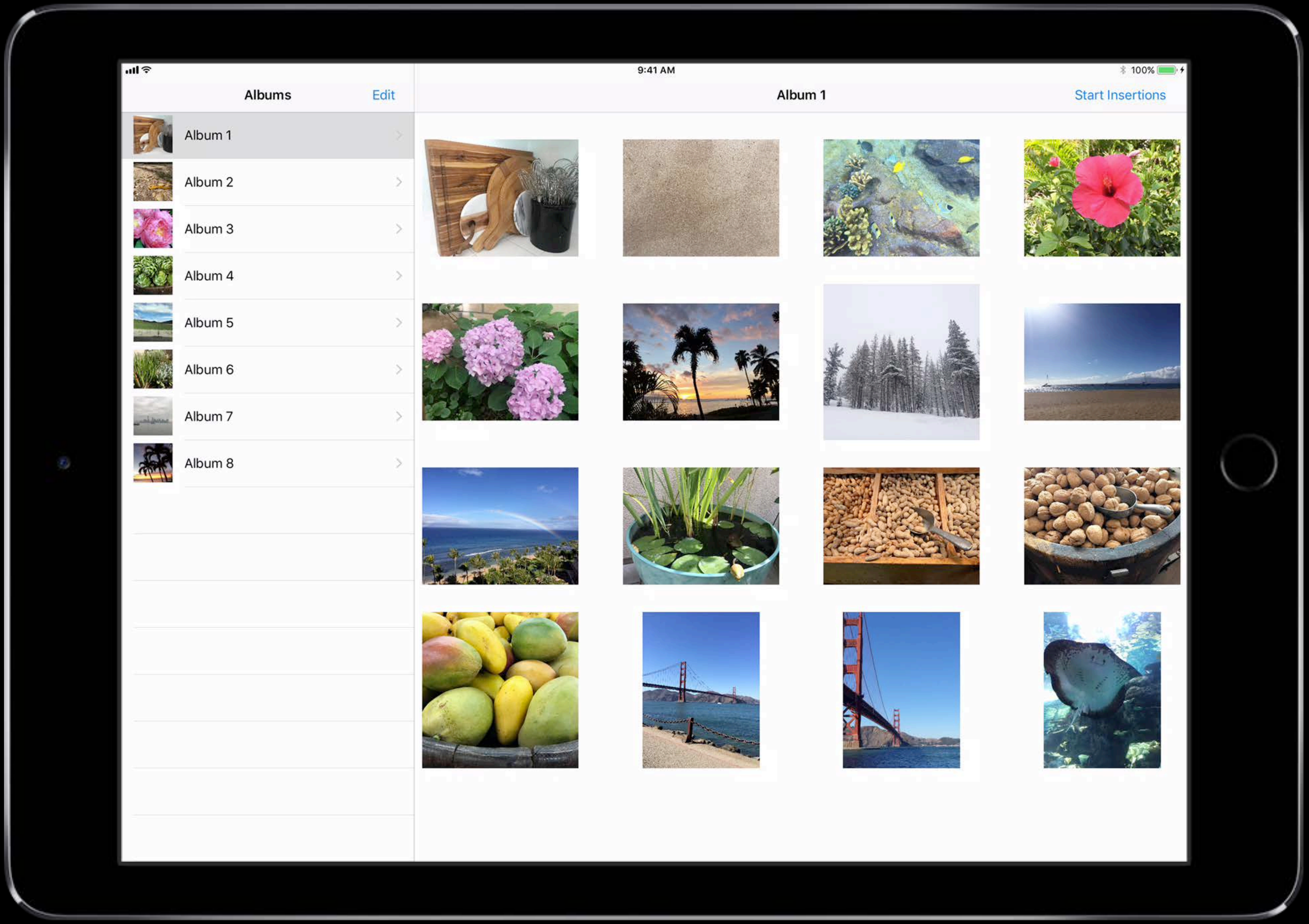
- Delete from `UICollectionViewDropItem.sourceIndexPath`
- Insert at `UICollectionViewDropCoordinator.destinationIndexPath`

# Collection View Reordering Cadence



# Collection View Reordering Cadence

```
var reorderingCadence: UICollectionViewReorderingCadence { get set }
```



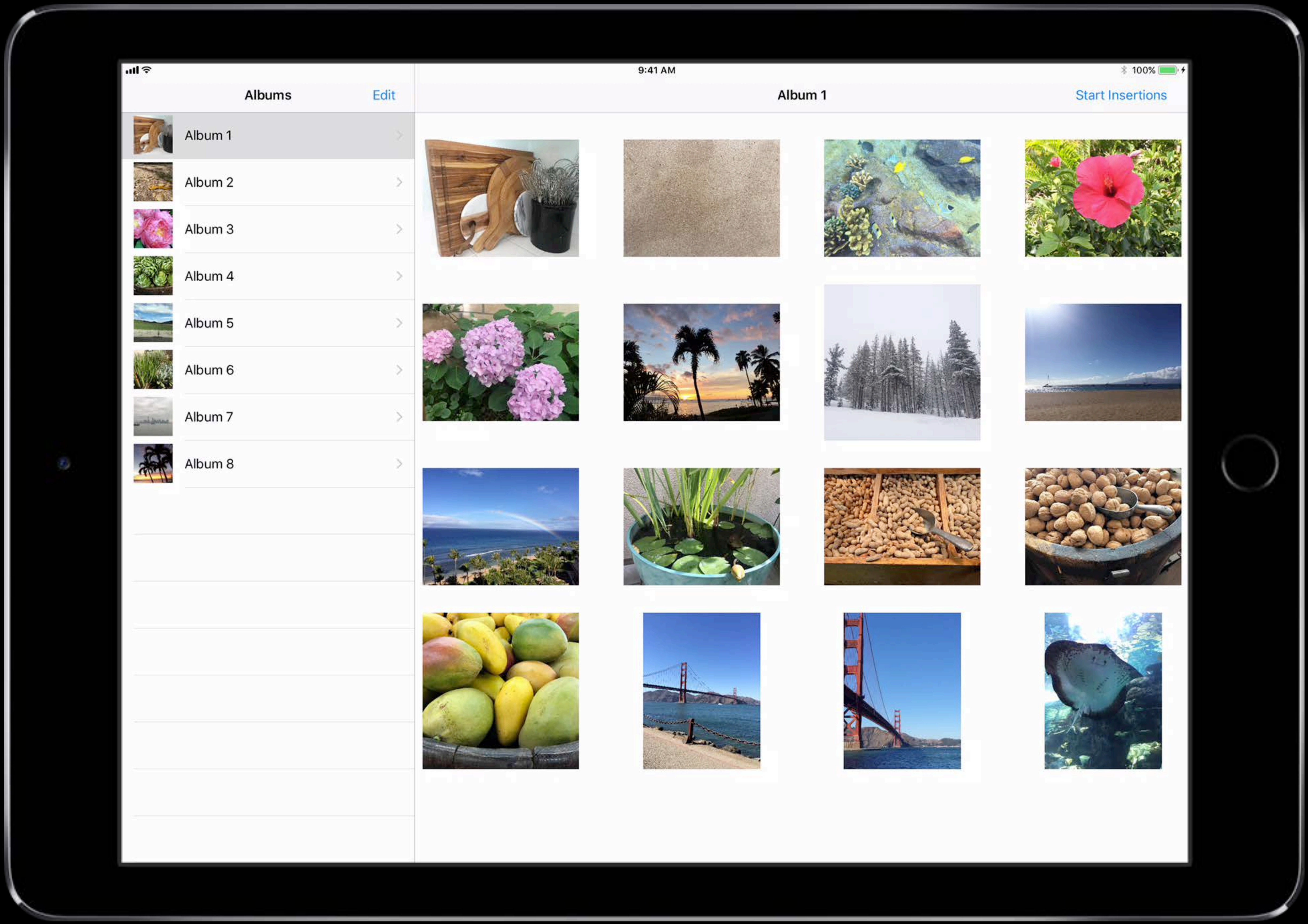
# Collection View Reordering Cadence

```
var reorderingCadence: UICollectionViewReorderingCadence { get set }
```

`.immediate`

`.fast`

`.slow`



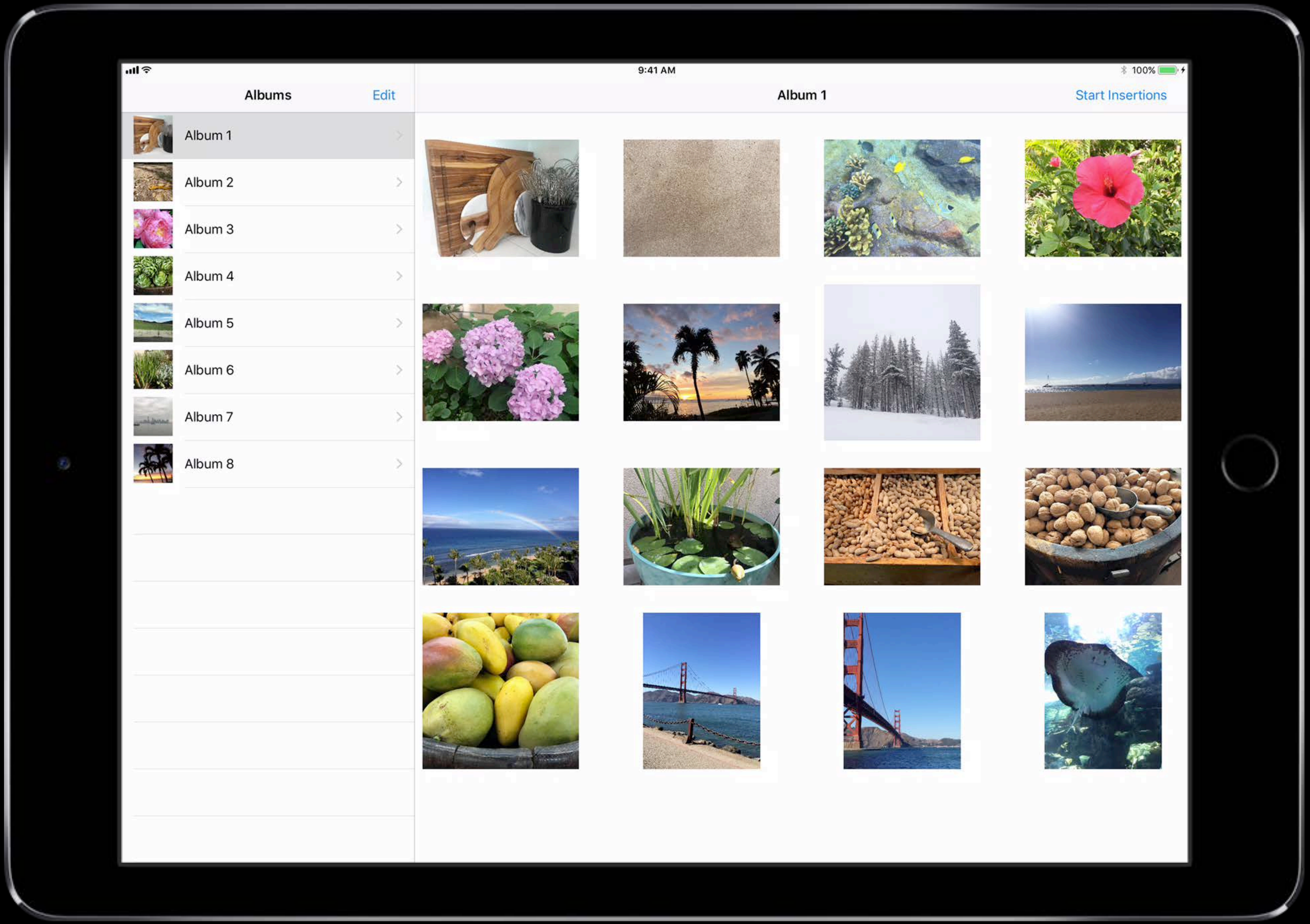
# Collection View Reordering Cadence

```
var reorderingCadence: UICollectionViewReorderingCadence { get set }
```

`.immediate`

`.fast`

`.slow`



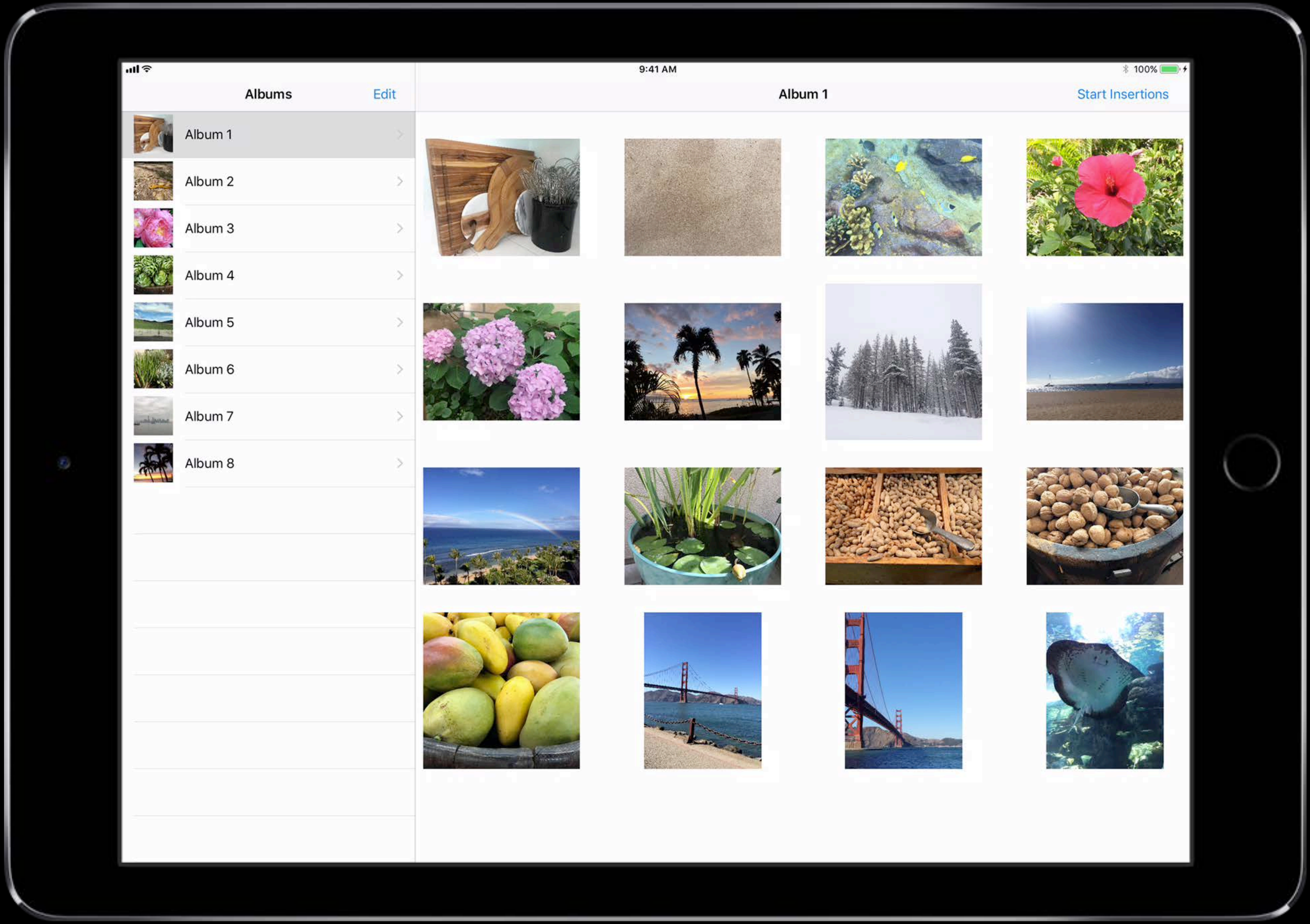
# Collection View Reordering Cadence

```
var reorderingCadence: UICollectionViewReorderingCadence { get set }
```

`.immediate`

`.fast`

`.slow`



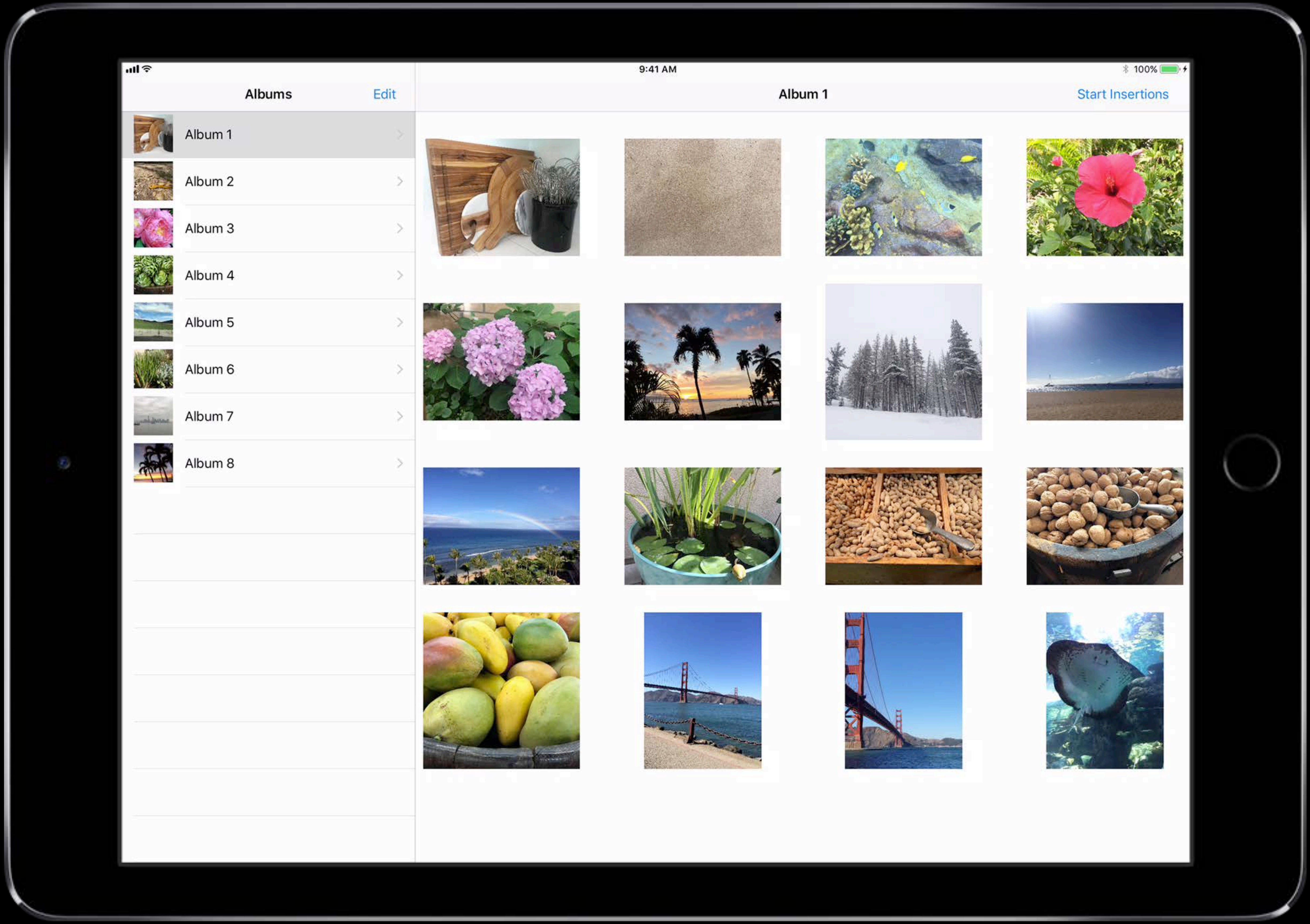
# Collection View Reordering Cadence

```
var reorderingCadence: UICollectionViewReorderingCadence { get set }
```

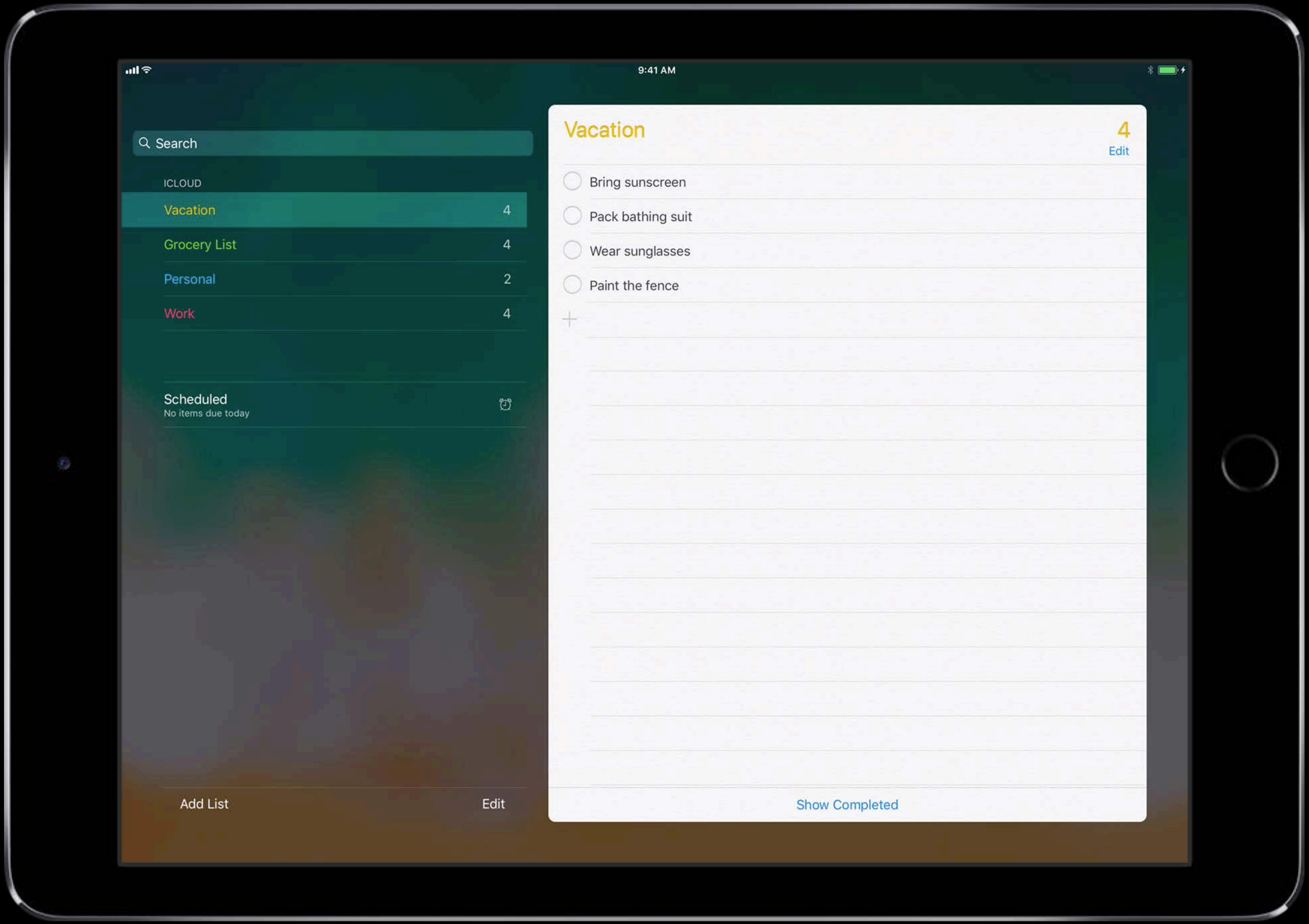
`.immediate`

`.fast`

`.slow`



# Spring Loading



Bring sunscreen
Pack bathing suit
Wear sunglasses
Paint the fence

## Spring Loading

## Spring Loading

Table and collection view conform to `UISpringLoadedInteractionSupporting`

```
var isSpringLoaded: Bool { get set }
```

## Spring Loading

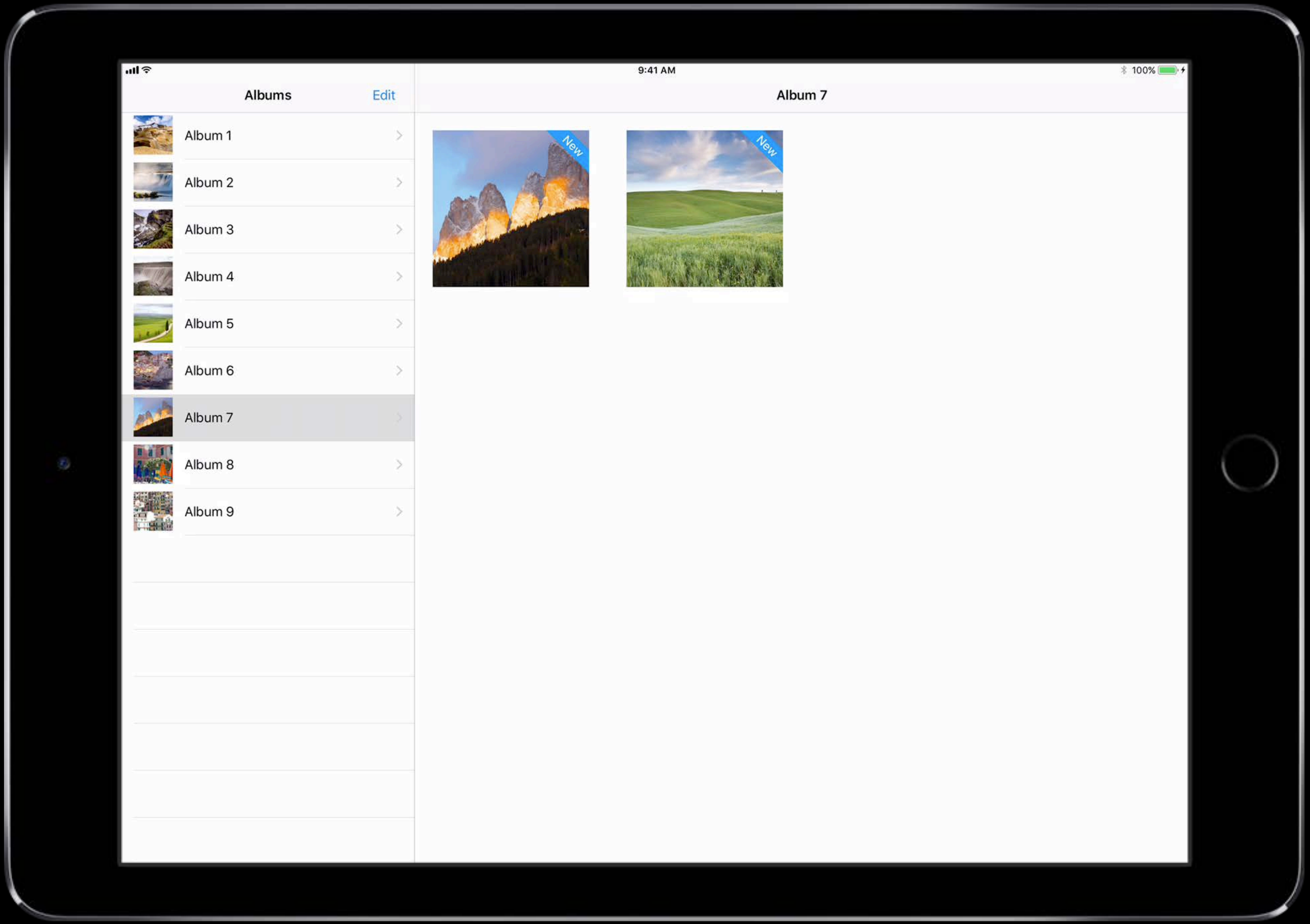
Table and collection view conform to `UISpringLoadedInteractionSupporting`

```
var isSpringLoaded: Bool { get set }
```

Customize spring loading with the optional delegate method

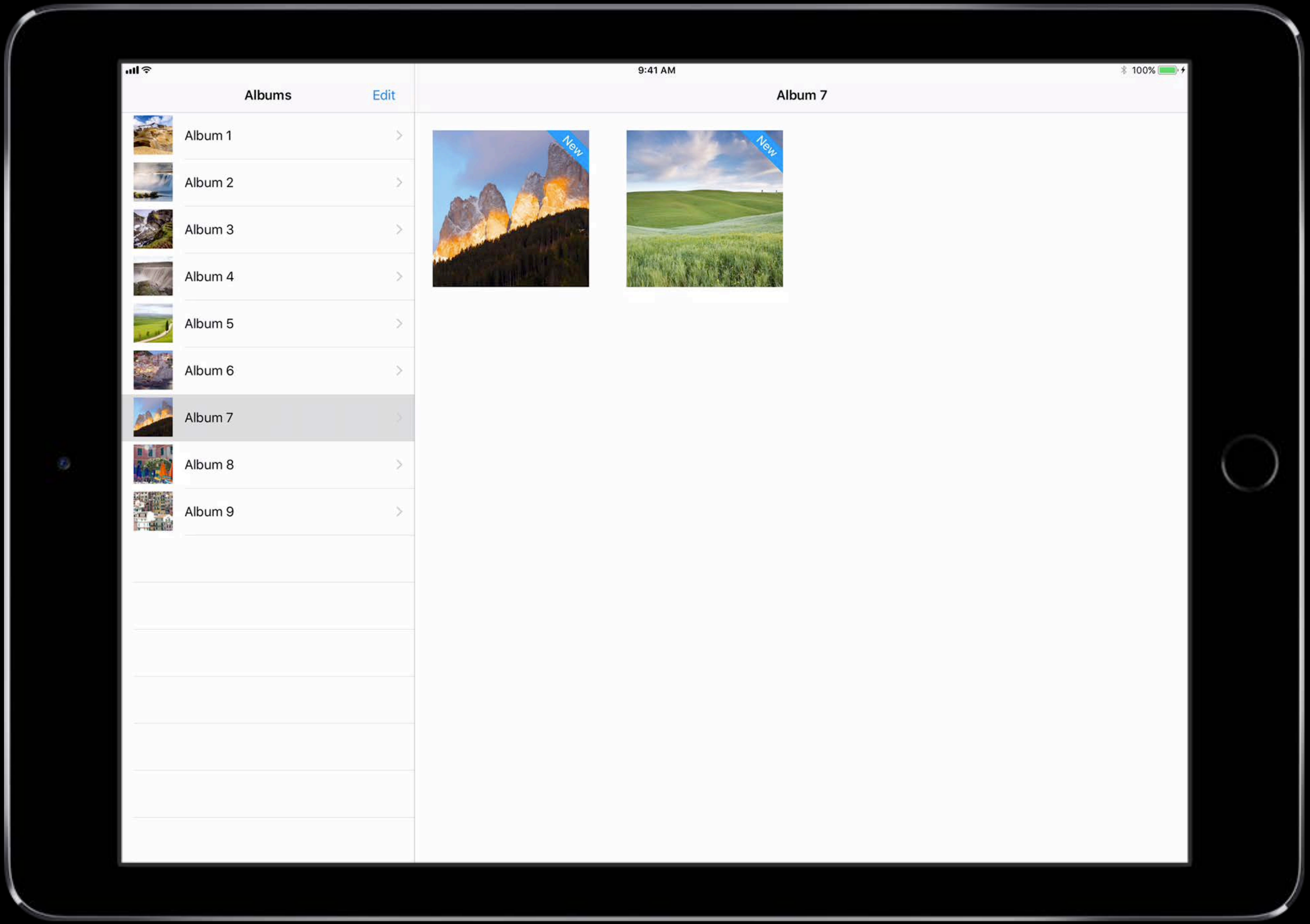
```
func collectionView(_: UICollectionView,
                    shouldSpringLoadItemAt: IndexPath,
                    with: UISpringLoadedInteractionContext) -> Bool
```

# Customizing Cell Appearance



# Customizing Cell Appearance

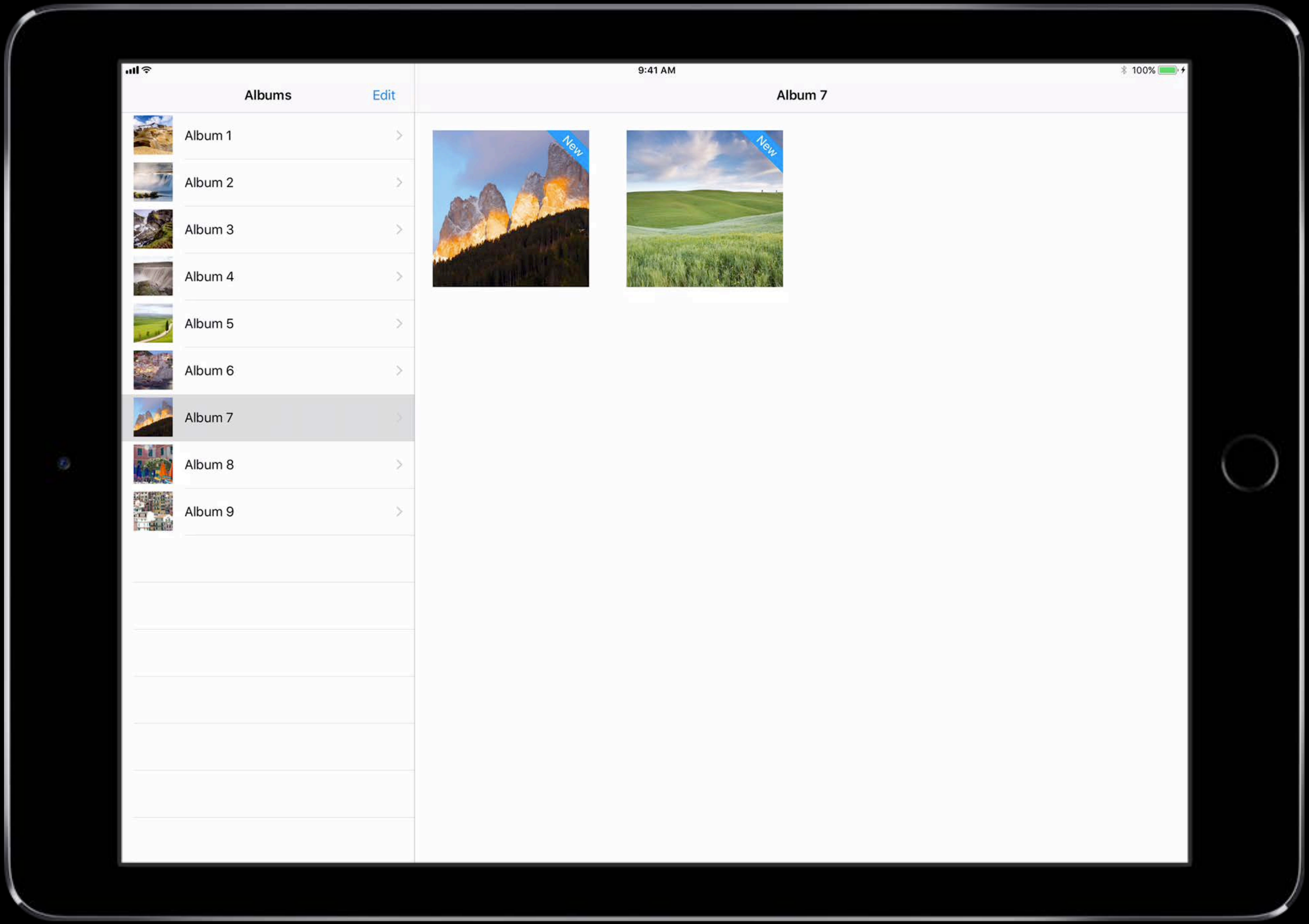
.none



# Customizing Cell Appearance

`.none`

`.lifting`

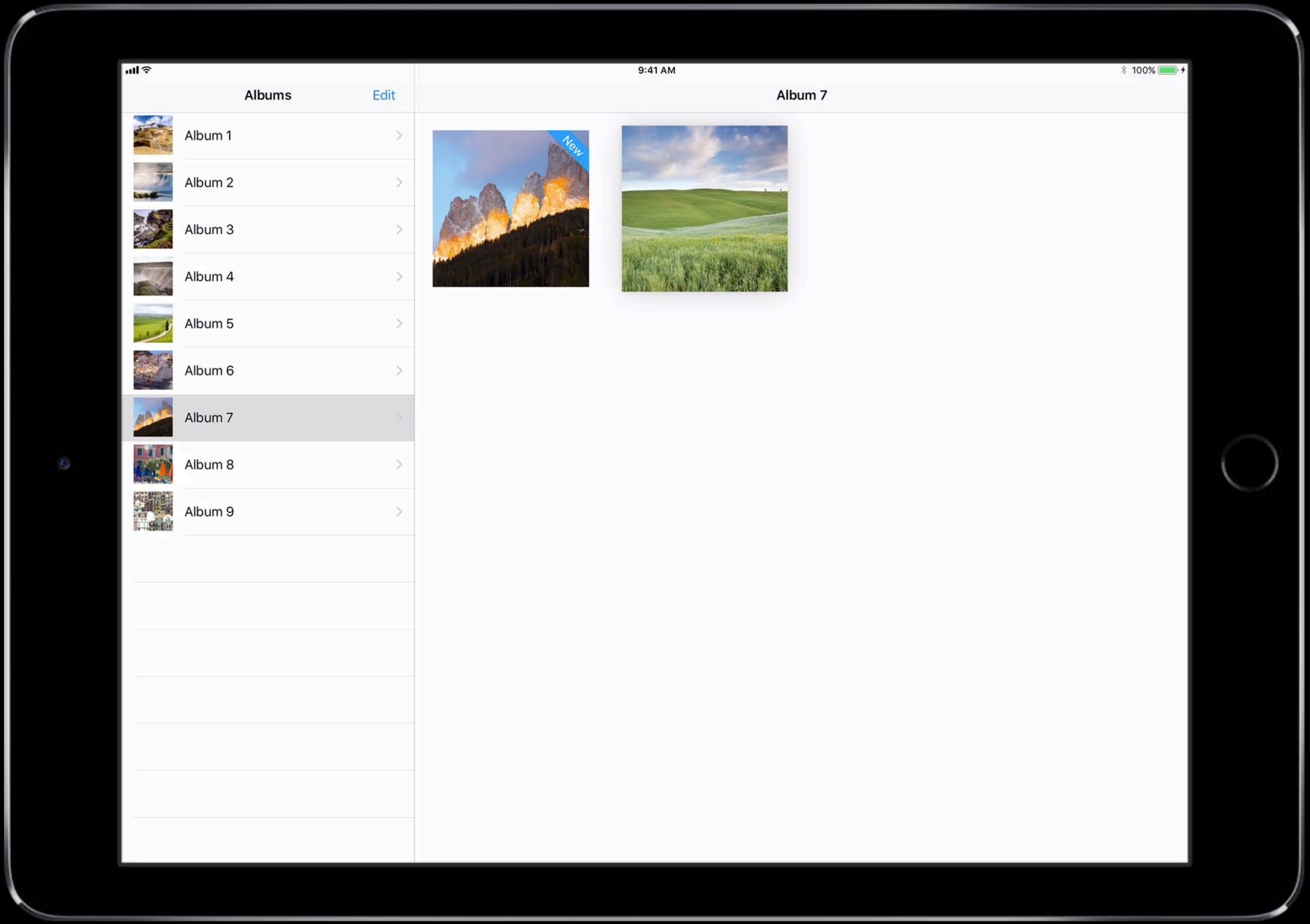


# Customizing Cell Appearance

`.none`

`.lifting`

`.dragging`

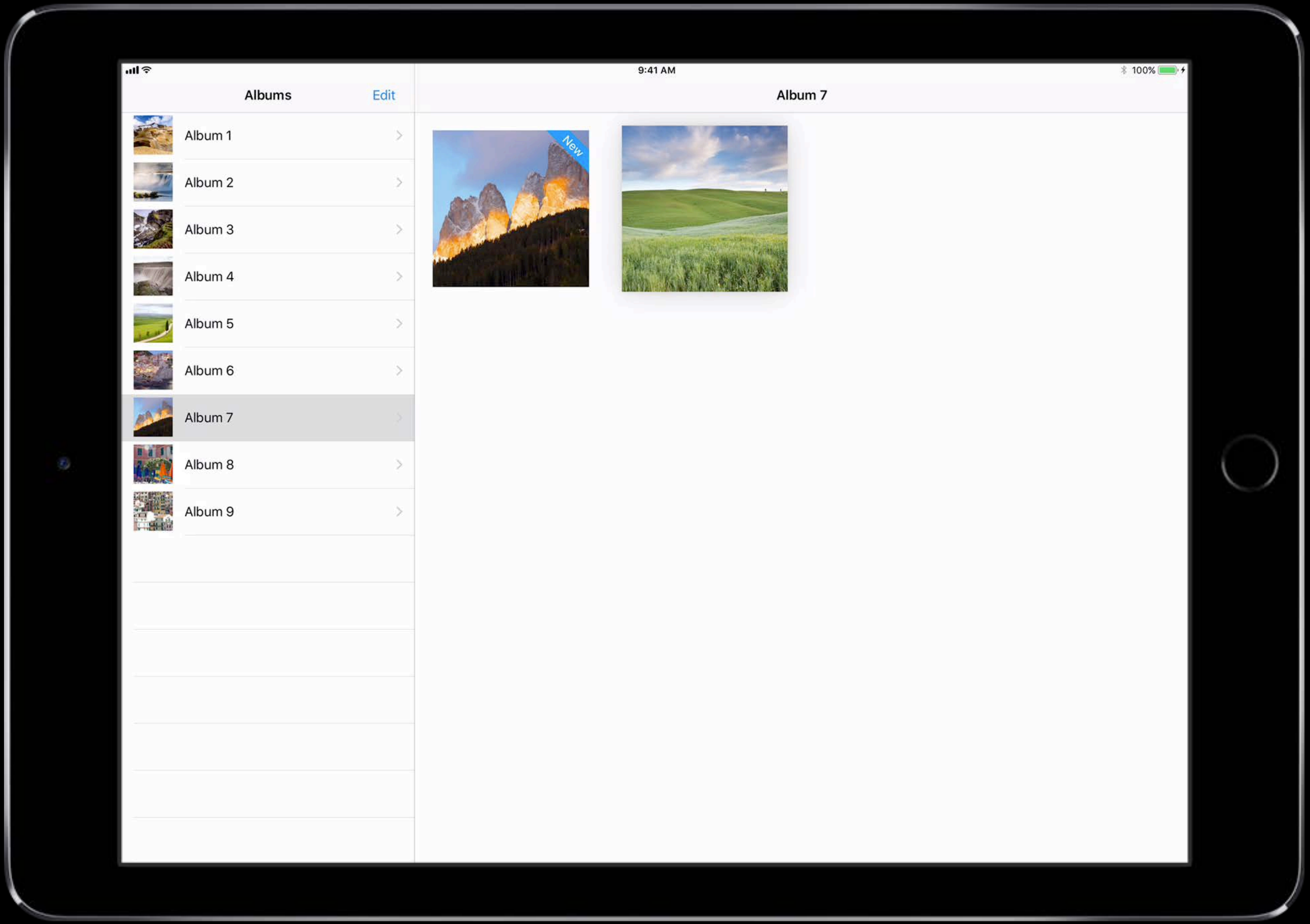


# Customizing Cell Appearance

`.none`

`.lifting`

`.dragging`

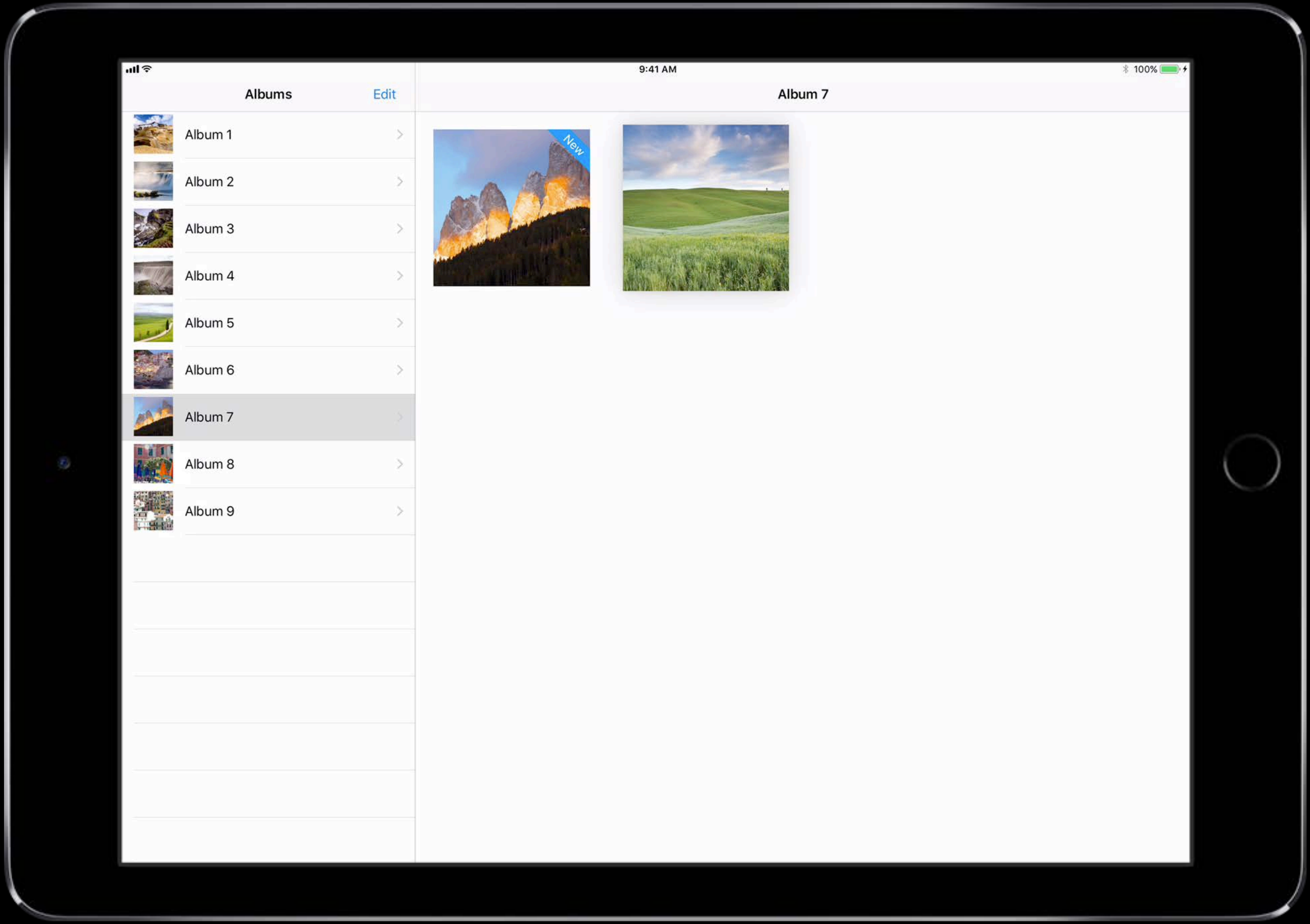


# Customizing Cell Appearance

`.none`

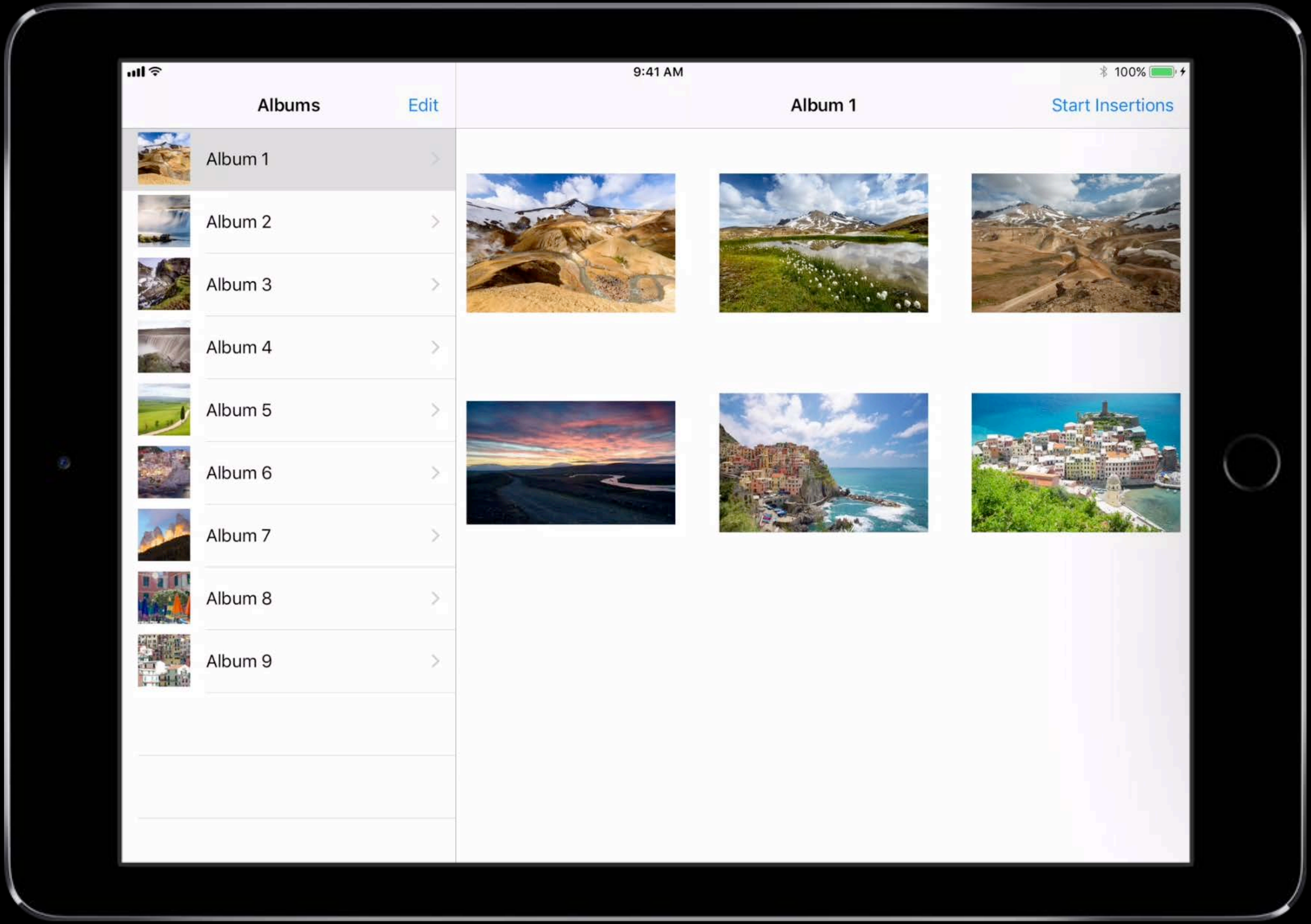
`.lifting`

`.dragging`



`dragStateDidChange(_:)`

# Customizing the Drag Preview



## Customizing the Drag Preview

## Customizing the Drag Preview

By default, the entire cell is used as the drag preview

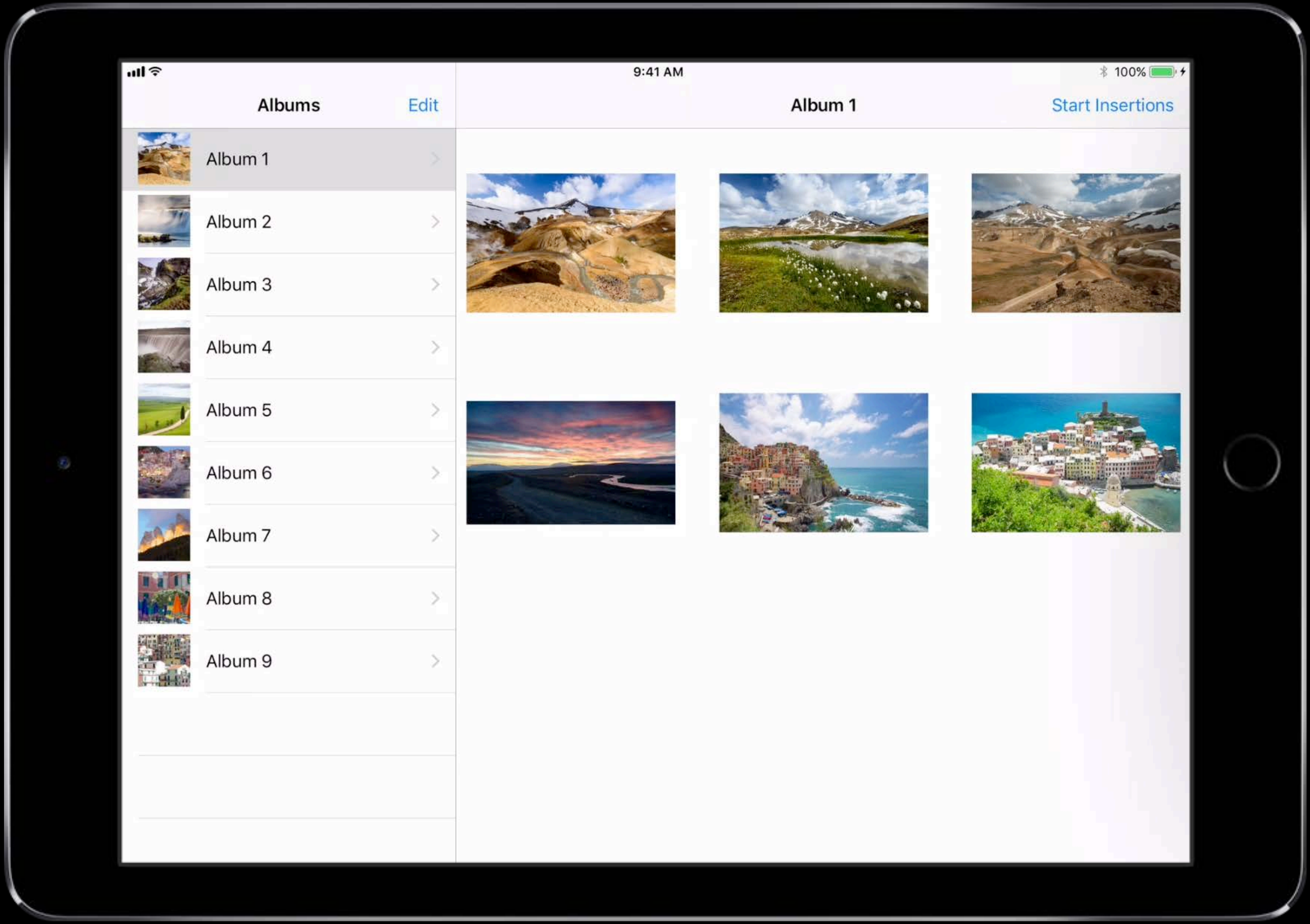
## Customizing the Drag Preview

By default, the entire cell is used as the drag preview

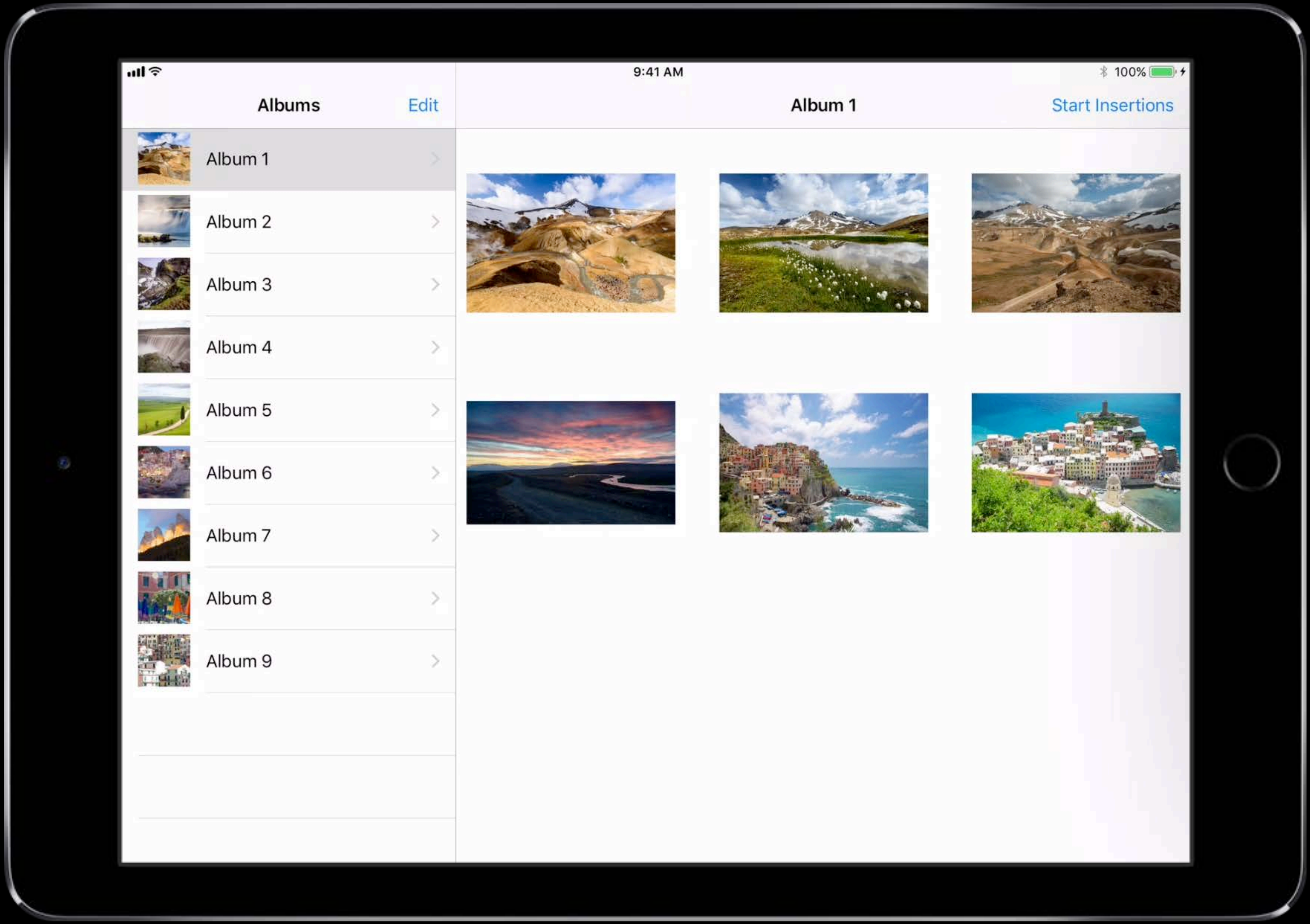
Provide drag preview parameters by implementing `dragDelegate` method

```
func collectionView(_: UICollectionView,
                    dragPreviewParametersForItemAt: IndexPath) -> UIDragPreviewParameters?
```

# Customizing the Drag Preview



# Customizing the Drag Preview



## Next Steps

## Next Steps

Add drag and drop to your collection and table views

## Next Steps

Add drag and drop to your collection and table views

Provide a drop proposal and set up drop animations

## Next Steps

Add drag and drop to your collection and table views

Provide a drop proposal and set up drop animations

Insert placeholders while data loads asynchronously

## Next Steps

Add drag and drop to your collection and table views

Provide a drop proposal and set up drop animations

Insert placeholders while data loads asynchronously

Polish the details

## More Information

https://developer.apple.com/wwdc17/223

## Related Sessions

| | | |
|---|---|---|
| Introducing Drag and Drop | Hall 3 | Tuesday 11:20AM |
| Mastering Drag and Drop | Executive Ballroom | Wednesday 11:00AM |
| Data Delivery with Drag and Drop | Hall 2 | Thursday 10:00AM |

## Labs

| | | |
|---|---|---|
| UIKit and Collection View Lab | Technology Lab B | Thur 10:00AM–12:30PM |
| Cocoa Touch and Haptics Lab | Technology Lab C | Fri 12:00PM–1:50PM |

WWDC17