Initializing a Collection View

`init(frame: CGRect, collectionViewLayout: UICollectionViewLayout)`
Initializes and returns a newly allocated collection view object with the specified frame and layout.

`init?(coder: NSCoder)`

Providing the Collection View Data

`var dataSource: UICollectionViewDataSource?`
The object that provides the data for the collection view.

`protocol UICollectionViewDataSource`
An object that adopts the `UICollectionViewDataSource` protocol is responsible for providing the data and views required by a collection view. A data source object represents your app's data model and vends information to the collection view as needed. It also handles the creation and configuration of cells and supplementary views used by the collection view to display your data.

`protocol UICollectionViewDataSourcePrefetching`
A protocol that provides advance warning of the data requirements for a collection view, allowing the triggering of asynchronous data load operations.

Managing Collection View Interactions

`var delegate: UICollectionViewDelegate?`
The object that acts as the delegate of the collection view.

`protocol UICollectionViewDelegate`
The `UICollectionViewDelegate` protocol defines methods that allow you to manage the selection and highlighting of items in a collection view and to perform actions on those items. The methods of this protocol are all optional.

Configuring the Background View

`var backgroundView: UIView?`
The view that provides the background appearance.

Prefetching Collection View Cells and Data

`UICollectionView` provides two prefetching techniques you can use to improve responsiveness:

- **Cell prefetching** prepares cells in advance of the time they are required. When a collection view requires a large number of cells simultaneously—for example, a new row of cells in grid layout—the cells are requested earlier than the time required for display. Cell rendering is therefore spread across multiple layout passes, resulting in a smoother scrolling experience. Cell prefetching is enabled by default.
- **Data prefetching** provides a mechanism whereby you are notified of the data requirements of a collection view in advance of the requests for cells. This is useful if the content of your cells relies on an expensive data loading process, such as a network request. Assign an object that conforms to the `UICollectionViewDataSourcePrefetching` protocol to the `prefetchDataSource` property to receive notifications of when to prefetch data for cells.

`var isPrefetchingEnabled: Bool`

Denotes whether cell and data prefetching are enabled.

`var prefetchDataSource: UICollectionViewDataSourcePrefetching?`

The object that acts as the prefetching data source for the collection view, receiving notifications of upcoming cell data requirements.

Creating Collection View Cells

`func register(AnyClass?, forCellWithReuseIdentifier: String)`

Register a class for use in creating new collection view cells.

`func register(UINib?, forCellWithReuseIdentifier: String)`

Register a nib file for use in creating new collection view cells.

`func register(AnyClass?, forSupplementaryViewOfKind: String, withReuseIdentifier: String)`

Registers a class for use in creating supplementary views for the collection view.

`func register(UINib?, forSupplementaryViewOfKind: String, withReuseIdentifier: String)`

Registers a nib file for use in creating supplementary views for the collection view.

`func dequeueReusableCell(withReuseIdentifier: String, for: IndexPath) -> UICollectionViewCell`

Returns a reusable cell object located by its identifier

`func dequeueReusableSupplementaryView(ofKind: String, withReuseIdentifier: String, for: IndexPath) -> UICollectionReusableView`

Returns a reusable supplementary view located by its identifier and kind.

Changing the Layout

```
var collectionViewLayout: UICollectionViewLayout
```
The layout used to organize the collected view's items.

```
func setCollectionViewLayout(UICollectionViewLayout, animated:
Bool)
```
Changes the collection view's layout and optionally animates the change.

```
func setCollectionViewLayout(UICollectionViewLayout, animated:
Bool, completion: ((Bool) -> Void)? = nil)
```
Changes the collection view's layout and notifies you when the animations complete.

```
func startInteractiveTransition(to: UICollectionViewLayout,
completion: UICollectionView.LayoutInteractiveTransition
Completion? = nil) -> UICollectionViewTransitionLayout
```
Changes the collection view's current layout using an interactive transition effect.

```
func finishInteractiveTransition()
```
Tells the collection view to finish an interactive transition by installing the intended target layout.

```
func cancelInteractiveTransition()
```
Tells the collection view to abort an interactive transition and return to its original layout object.

{}Customizing Collection View Layouts

Customize a view layout by changing the size of cells in the flow or implementing a mosaic style.

Getting the State of the Collection View

```
var numberOfSections: Int
```
Returns the number of sections displayed by the collection view.

```
func numberOfItems(inSection: Int) -> Int
```
Returns the number of items in the specified section.

```
var visibleCells: [UICollectionViewCell]
```
Returns an array of visible cells currently displayed by the collection view.

Inserting, Moving, and Deleting Items

`func insertItems(at: [IndexPath])`
Inserts new items at the specified index paths.

`func moveItem(at: IndexPath, to: IndexPath)`
Moves an item from one location to another in the collection view.

`func deleteItems(at: [IndexPath])`
Deletes the items at the specified index paths.

Inserting, Moving, and Deleting Sections

`func insertSections(IndexSet)`
Inserts new sections at the specified indexes.

`func moveSection(Int, toSection: Int)`
Moves a section from one location to another in the collection view.

`func deleteSections(IndexSet)`
Deletes the sections at the specified indexes.

Reordering Items Interactively

`func beginInteractiveMovementForItem(at: IndexPath) -> Bool`
Initiates the interactive movement of the item at the specified index path.

`func updateInteractiveMovementTargetPosition(CGPoint)`
Updates the position of the item within the collection view's bounds.

`func endInteractiveMovement()`
Ends interactive movement tracking and moves the target item to its new location.

`func cancelInteractiveMovement()`
Ends interactive movement tracking and returns the target item to its original location.

Managing Drag Interactions

`var dragDelegate: UICollectionViewDragDelegate?`
The delegate object that manages the dragging of items from the collection view.

`protocol UICollectionViewDragDelegate`
The interface for initiating drags from a collection view.

`var hasActiveDrag: Bool`
A Boolean value indicating whether items were lifted from the collection view and have not yet been dropped.

`var dragInteractionEnabled: Bool`
A Boolean value indicating whether the collection view supports drags and drops between apps.

Managing Drop Interactions

`var dropDelegate: UICollectionViewDropDelegate?`
The delegate object that manages the dropping of items into the collection view.

`var hasActiveDrop: Bool`
A Boolean value indicating whether the collection view is currently tracking a drop session.

`var reorderingCadence: UICollectionView.ReorderingCadence`
The speed at which items in the collection view are reordered to show potential drop locations.

`enum UICollectionView.ReorderingCadence`
Constants indicating the speed at which collection view items are reorganized during a drop.

Managing the Selection

`var allowsSelection: Bool`
A Boolean value that indicates whether users can select items in the collection view.

`var allowsMultipleSelection: Bool`
A Boolean value that determines whether users can select more than one item in the collection view.

`var indexPathsForSelectedItems: [IndexPath]?`
The index paths for the selected items.

`func selectItem(at: IndexPath?, animated: Bool, scrollPosition: UICollectionView.ScrollPosition)`
Selects the item at the specified index path and optionally scrolls it into view.

`func deselectItem(at: IndexPath, animated: Bool)`
Deselects the item at the specified index.

Managing Focus

```
var rememberslastFocusedIndexPath: Bool
```
A Boolean value indicating whether the collection view automatically assigns the focus to the item at the last focused index path.

Locating Items and Views in the Collection View

```
func indexPathForItem(at: CGPoint) -> IndexPath?
```
Returns the index path of the item at the specified point in the collection view.

```
var indexPathsForVisibleItems: [IndexPath]
```
An array of the visible items in the collection view.

```
func indexPath(for: UICollectionViewCell) -> IndexPath?
```
Returns the index path of the specified cell.

```
func cellForItem(at: IndexPath) -> UICollectionViewCell?
```
Returns the visible cell object at the specified index path.

```
func indexPathsForVisibleSupplementaryElements(ofKind: String) -> [IndexPath]
```
Returns the index paths of all visible supplementary views of the specified type.

```
func supplementaryView(forElementKind: String, at: IndexPath) -> UICollectionReusableView?
```
Returns the supplementary view at the specified index path.

```
func visibleSupplementaryViews(ofKind: String) -> [UICollection ReusableView]
```
Returns an array of the visible supplementary views of the specified kind.

Getting Layout Information

```
func layoutAttributesForItem(at: IndexPath) -> UICollectionView LayoutAttributes?
```
Returns the layout information for the item at the specified index path.

```
func layoutAttributesForSupplementaryElement(ofKind: String, at: IndexPath) -> UICollectionViewLayoutAttributes?
```
Returns the layout information for the specified supplementary view.

Scrolling an Item Into View

```
func scrollToItem(at: IndexPath, at: UICollectionView.Scroll Position, animated: Bool)
```
Scrolls the collection view contents until the specified item is visible.

Animating Multiple Changes to the Collection View

```
func performBatchUpdates((() -> Void)?, completion: ((Bool) -> Void)? = nil)
```
Animates multiple insert, delete, reload, and move operations as a group.

Reloading Content

```
var hasUncommittedUpdates: Bool
```
A Boolean value indicating whether the collection view contains drop placeholders or is reordering its items as part of handling a drop.

```
func reloadData()
```
Reloads all of the data for the collection view.

```
func reloadSections(IndexSet)
```
Reloads the data in the specified sections of the collection view.

```
func reloadItems(at: [IndexPath])
```
Reloads just the items at the specified index paths.

Constants

```
struct UICollectionView.ScrollPosition
```
Constants that indicate how to scroll an item into the visible portion of the collection view.

```
typealias UICollectionView.LayoutInteractiveTransitionCompletion
```
The completion block called at the end of an interactive transition for a collection view.

Type Properties

```
class let elementKindSectionFooter: String
```

```
class let elementKindSectionHeader: String
```

Enumerations

```
enum UICollectionView.ElementCategory
```
Constants specifying the type of view.

```
enum UICollectionView.ScrollDirection
```
Constants indicating the direction of scrolling for the layout.

UICollectionViewCell

Accessing the Cell's Views

`var contentView: UIView`
The main view to which you add your cell's custom content.

`var backgroundView: UIView?`
The view that is displayed behind the cell's other content.

`var selectedBackgroundView: UIView?`
The view that is displayed just above the background view when the cell is selected.

Managing the Cell's State

`var isSelected: Bool`
The selection state of the cell.

`var isHighlighted: Bool`
The highlight state of the cell.

Managing Drag State Changes

`func dragStateDidChange(UICollectionViewCell.DragState)`
Called when the drag state of the cell changes.

`enum UICollectionViewCell.DragState`
Constants indicating the current state of the drag operation.

Protocol

UICollectionViewDataSource

Getting Item and Section Metrics

`func collectionView(UICollectionView, numberOfItemsInSection: Int) -> Int`
Asks your data source object for the number of items in the specified section.
Required.

`func numberOfSections(in: UICollectionView) -> Int`
Asks your data source object for the number of sections in the collection view.

Getting Views for Items

```
func collectionView(UICollectionView, cellForItemAt: IndexPath) -> UICollectionViewCell
```

Asks your data source object for the cell that corresponds to the specified item in the collection view.

Required.

```
func collectionView(UICollectionView, viewForSupplementaryElementOfKind: String, at: IndexPath) -> UICollectionReusableView
```

Asks your data source object to provide a supplementary view to display in the collection view.

Reordering Items

```
func collectionView(UICollectionView, canMoveItemAt: IndexPath) -> Bool
```

Asks your data source object whether the specified item can be moved to another location in the collection view.

```
func collectionView(UICollectionView, moveItemAt: IndexPath, to: IndexPath)
```

Tells your data source object to move the specified item to its new location.

Configuring an Index

```
func indexTitles(for: UICollectionView) -> [String]?
```

Asks the data source to return the titles for the index items to display for the collection view.

```
func collectionView(UICollectionView, indexPathForIndexTitle: String, at: Int) -> IndexPath
```

Asks the data source to return the index path of a collection view item that corresponds to one of your index entries.

UICollectionViewLayout

Initializing the Collection View

```
init()
```

Initializes the collection view layout object.

```
init?(coder: NSCoder)
```

Getting the Collection View Information

`var collectionView: UICollectionView?`
The collection view object currently using this layout object.

`var collectionViewContentSize: CGSize`
Returns the width and height of the collection view's contents.

Providing Layout Attributes

`class var layoutAttributesClass: AnyClass`
Returns the class to use when creating layout attributes objects.

`func prepare()`
Tells the layout object to update the current layout.

`func layoutAttributesForElements(in: CGRect) -> [UICollectionViewLayoutAttributes]?`
Returns the layout attributes for all of the cells and views in the specified rectangle.

`func layoutAttributesForItem(at: IndexPath) -> UICollectionViewLayoutAttributes?`
Returns the layout attributes for the item at the specified index path.

`func layoutAttributesForInteractivelyMovingItem(at: IndexPath, withTargetPosition: CGPoint) -> UICollectionViewLayoutAttributes`
Returns the layout attributes of an item when it is being moved interactively by the user.

`func layoutAttributesForSupplementaryView(ofKind: String, at: IndexPath) -> UICollectionViewLayoutAttributes?`
Returns the layout attributes for the specified supplementary view.

`func layoutAttributesForDecorationView(ofKind: String, at: IndexPath) -> UICollectionViewLayoutAttributes?`
Returns the layout attributes for the specified decoration view.

`func targetContentOffset(forProposedContentOffset: CGPoint) -> CGPoint`
Returns the content offset to use after an animated layout update or change.

`func targetContentOffset(forProposedContentOffset: CGPoint, withScrollingVelocity: CGPoint) -> CGPoint`
Returns the point at which to stop scrolling.

Responding to Collection View Updates

`func prepare(forCollectionViewUpdates: [UICollectionViewUpdateItem])`
Notifies the layout object that the contents of the collection view are about to change.

```
func finalizeCollectionViewUpdates()
```

Performs any additional animations or clean up needed during a collection view update.

```
func indexPathsToInsertForSupplementaryView(ofKind: String) -> [IndexPath]
```

Returns an array of index paths for the supplementary views you want to add to the layout.

```
func indexPathsToInsertForDecorationView(ofKind: String) -> [IndexPath]
```

Returns an array of index paths representing the decoration views to add.

```
func initialLayoutAttributesForAppearingItem(at: IndexPath) -> UICollectionViewLayoutAttributes?
```

Returns the starting layout information for an item being inserted into the collection view.

```
func initialLayoutAttributesForAppearingSupplementaryElement(ofKind: String, at: IndexPath) -> UICollectionViewLayoutAttributes?
```

Returns the starting layout information for a supplementary view being inserted into the collection view.

```
func initialLayoutAttributesForAppearingDecorationElement(ofKind: String, at: IndexPath) -> UICollectionViewLayoutAttributes?
```

Returns the starting layout information for a decoration view being inserted into the collection view.

```
func indexPathsToDeleteForSupplementaryView(ofKind: String) -> [IndexPath]
```

Returns an array of index paths representing the supplementary views to remove.

```
func indexPathsToDeleteForDecorationView(ofKind: String) -> [IndexPath]
```

Returns an array of index paths representing the decoration views to remove.

```
func finalLayoutAttributesForDisappearingItem(at: IndexPath) -> UICollectionViewLayoutAttributes?
```

Returns the final layout information for an item that is about to be removed from the collection view.

```
func finalLayoutAttributesForDisappearingSupplementaryElement(ofKind: String, at: IndexPath) -> UICollectionViewLayoutAttributes?
```

Returns the final layout information for a supplementary view that is about to be removed from the collection view.

```
func finalLayoutAttributesForDisappearingDecorationElement(ofKind: String, at: IndexPath) -> UICollectionViewLayoutAttributes?
```

Returns the final layout information for a decoration view that is about to be removed from the collection view.

```
func targetIndexPath(forInteractivelyMovingItem: IndexPath, withPosition: CGPoint) -> IndexPath
```

Returns the index path to for an item when it is at the specified location in the collection view's bounds.

Invalidating the Layout

```
func invalidateLayout()
```
Invalidates the current layout and triggers a layout update.

```
func invalidateLayout(with: UICollectionViewLayoutInvalidation
Context)
```
Invalidates the current layout using the information in the provided context object.

```
class var invalidationContextClass: AnyClass
```
Returns the class to use when creating an invalidation context for the layout.

```
func shouldInvalidateLayout(forBoundsChange: CGRect) -> Bool
```
Asks the layout object if the new bounds require a layout update.

```
func invalidationContext(forBoundsChange: CGRect) -> UICollection
ViewLayoutInvalidationContext
```
Returns a context object that defines the portions of the layout that should change when a bounds change occurs.

```
func shouldInvalidateLayout(forPreferredLayoutAttributes:
UICollectionViewLayoutAttributes, withOriginalAttributes:
UICollectionViewLayoutAttributes) -> Bool
```
Asks the layout object if changes to a self-sizing cell require a layout update.

```
func invalidationContext(forPreferredLayoutAttributes: UICollection
ViewLayoutAttributes, withOriginalAttributes: UICollectionView
LayoutAttributes) -> UICollectionViewLayoutInvalidationContext
```
Returns a context object that identifies the portions of the layout that should change in response to dynamic cell changes.

```
func invalidationContext(forInteractivelyMovingItems: [IndexPath],
withTargetPosition: CGPoint, previousIndexPaths: [IndexPath],
previousPosition: CGPoint) -> UICollectionViewLayoutInvalidation
Context
```
Returns a context object that identifies the items that are being interactively moved in the layout.

```
func invalidationContextForEndingInteractiveMovementOfItems(toFinal
IndexPaths: [IndexPath], previousIndexPaths: [IndexPath], movement
Cancelled: Bool) -> UICollectionViewLayoutInvalidationContext
```
Returns a context object that identifies the items that were moved

Coordinating Animated Changes

`func prepare(forAnimatedBoundsChange: CGRect)`
Prepares the layout object for animated changes to the view's bounds or the insertion or deletion of items.

`func finalizeAnimatedBoundsChange()`
Cleans up after any animated changes to the view's bounds or after the insertion or deletion of items.

Transitioning Between Layouts

`func prepareForTransition(from: UICollectionViewLayout)`
Tells the layout object to prepare to be installed as the layout for the collection view.

`func prepareForTransition(to: UICollectionViewLayout)`
Tells the layout object that it is about to be removed as the layout for the collection view.

`func finalizeLayoutTransition()`
Tells the layout object to perform any final steps before the transition animations occur.

Registering Decoration Views

`func register(AnyClass?, forDecorationViewOfKind: String)`
Registers a class for use in creating decoration views for a collection view.

`func register(UINib?, forDecorationViewOfKind: String)`
Registers a nib file for use in creating decoration views for a collection view.

Supporting Right-To-Left Layouts

`var developmentLayoutDirection: UIUserInterfaceLayoutDirection`
The direction of the language you used when designing your custom layout.

`var flipsHorizontallyInOppositeLayoutDirection: Bool`
A Boolean value indicating whether the horizontal coordinate system is automatically flipped at appropriate times.

UICollectionViewFlowLayout

Configuring the Flow Layout

`protocol UICollectionViewDelegateFlowLayout`
The `UICollectionViewDelegateFlowLayout` protocol defines methods that let you coordinate with a `UICollectionViewFlowLayout` object to implement a grid-based layout. The methods of this protocol define the size of items and the spacing between items in the grid.

Configuring the Scroll Direction

```
var scrollDirection: UICollectionView.ScrollDirection
```
The scroll direction of the grid.

Configuring the Item Spacing

```
var minimumLineSpacing: CGFloat
```
The minimum spacing to use between lines of items in the grid.

```
var minimumInteritemSpacing: CGFloat
```
The minimum spacing to use between items in the same row.

```
var itemSize: CGSize
```
The default size to use for cells.

```
var estimatedItemSize: CGSize
```
The estimated size of cells in the collection view.

```
var sectionInset: UIEdgeInsets
```
The margins used to lay out content in a section

```
var sectionInsetReference: UICollectionViewFlowLayout.SectionInset
Reference
```

```
enum UICollectionViewFlowLayout.SectionInsetReference
```

Configuring the Supplementary Views

```
var headerReferenceSize: CGSize
```
The default sizes to use for section headers.

```
var footerReferenceSize: CGSize
```
The default sizes to use for section footers.

Pinning Headers and Footers

```
var sectionHeadersPinToVisibleBounds: Bool
```
A Boolean value indicating whether headers pin to the top of the collection view bounds during scrolling.

```
var sectionFootersPinToVisibleBounds: Bool
```
A Boolean value indicating whether footers pin to the bottom of the collection view bounds during scrolling.

Constants

`enum UICollectionView.ScrollDirection`

Constants indicating the direction of scrolling for the layout.

Flow Layout Supplementary Views

Constants that specify the types of supplementary views that can be presented using a flow layout.

`class let automaticSize: CGSize`

Protocol

UICollectionViewDelegate

Managing the Selected Cells

`func collectionView(UICollectionView, shouldSelectItemAt: IndexPath) -> Bool`

Asks the delegate if the specified item should be selected.

`func collectionView(UICollectionView, didSelectItemAt: IndexPath)`

Tells the delegate that the item at the specified index path was selected.

`func collectionView(UICollectionView, shouldDeselectItemAt: IndexPath) -> Bool`

Asks the delegate if the specified item should be deselected.

`func collectionView(UICollectionView, didDeselectItemAt: IndexPath)`

Tells the delegate that the item at the specified path was deselected.

Managing Cell Highlighting

`func collectionView(UICollectionView, shouldHighlightItemAt: IndexPath) -> Bool`

Asks the delegate if the item should be highlighted during tracking.

`func collectionView(UICollectionView, didHighlightItemAt: IndexPath)`

Tells the delegate that the item at the specified index path was highlighted.

`func collectionView(UICollectionView, didUnhighlightItemAt: IndexPath)`

Tells the delegate that the highlight was removed from the item at the specified index path.

Tracking the Addition and Removal of Views

```
func collectionView(UICollectionView, willDisplay: UICollectionViewCell, forItemAt: IndexPath)
```

Tells the delegate that the specified cell is about to be displayed in the collection view.

```
func collectionView(UICollectionView, willDisplaySupplementaryView: UICollectionReusableView, forElementKind: String, at: IndexPath)
```

Tells the delegate that the specified supplementary view is about to be displayed in the collection view.

```
func collectionView(UICollectionView, didEndDisplaying: UICollectionViewCell, forItemAt: IndexPath)
```

Tells the delegate that the specified cell was removed from the collection view.

```
func collectionView(UICollectionView, didEndDisplayingSupplementaryView: UICollectionReusableView, forElementOfKind: String, at: IndexPath)
```

Tells the delegate that the specified supplementary view was removed from the collection view.

Handling Layout Changes

```
func collectionView(UICollectionView, transitionLayoutForOldLayout: UICollectionViewLayout, newLayout: UICollectionViewLayout) -> UICollectionViewTransitionLayout
```

Asks for the custom transition layout to use when moving between the specified layouts.

```
func collectionView(UICollectionView, targetContentOffsetForProposedContentOffset: CGPoint) -> CGPoint
```

Gives the delegate an opportunity to customize the content offset for layout changes and animated updates.

```
func collectionView(UICollectionView, targetIndexPathForMoveFromItemAt: IndexPath, toProposedIndexPath: IndexPath) -> IndexPath
```

Asks the delegate for the index path to use when moving an item.

Managing Actions for Cells

```
func collectionView(UICollectionView, shouldShowMenuForItemAt: IndexPath) -> Bool
```

Asks the delegate if an action menu should be displayed for the specified item.

```
func collectionView(UICollectionView, canPerformAction: Selector, forItemAt: IndexPath, withSender: Any?) -> Bool
```

Asks the delegate if it can perform the specified action on an item in the collection view.

```
func collectionView(UICollectionView, performAction: Selector, forItemAt: IndexPath, withSender: Any?)
```

Tells the delegate to perform the specified action on an item in the collection view.

Managing Focus in a Collection View

```
func collectionView(UICollectionView, canFocusItemAt: IndexPath) -> Bool
```
Asks the delegate whether the item at the specified index path can be focused.

```
func indexPathForPreferredFocusedView(in: UICollectionView) -> IndexPath?
```
Asks the delegate for the index path of the cell that should be focused.

```
func collectionView(UICollectionView, shouldUpdateFocusIn: UICollectionViewFocusUpdateContext) -> Bool
```
Asks the delegate whether a change in focus should occur.

```
func collectionView(UICollectionView, didUpdateFocusIn: UICollectionViewFocusUpdateContext, with: UIFocusAnimationCoordinator)
```
Tells the delegate that a focus update occurred.

Controlling the Spring-Loading Behavior

```
func collectionView(UICollectionView, shouldSpringLoadItemAt: IndexPath, with: UISpringLoadedInteractionContext) -> Bool
```
Returns a Boolean value indicating whether you want the spring-loading interaction effect displayed for the specified item.

Class

UICollectionReusableView

Reusing Cells

```
var reuseIdentifier: String?
```
A string that identifies the purpose of the view.

```
func prepareForReuse()
```
Performs any clean up necessary to prepare the view for use again.

Managing Layout Changes

`func preferredLayoutAttributesFitting(UICollectionViewLayoutAttributes) -> UICollectionViewLayoutAttributes`

Gives the cell a chance to modify the attributes provided by the layout object.

`func apply(UICollectionViewLayoutAttributes)`

Applies the specified layout attributes to the view.

`func willTransition(from: UICollectionViewLayout, to: UICollectionViewLayout)`

Tells your view that the layout object of the collection view is about to change.

`func didTransition(from: UICollectionViewLayout, to: UICollectionViewLayout)`

Tells your view that the layout object of the collection view changed.

Class

UICollectionViewLayoutAttributes

Creating Layout Attributes

`init(forCellWith: IndexPath)`

Creates and returns a layout attributes object that represents a cell with the specified index path.

`init(forSupplementaryViewOfKind: String, with: IndexPath)`

Creates and returns a layout attributes object that represents the specified supplementary view.

`init(forDecorationViewOfKind: String, with: IndexPath)`

Creates and returns a layout attributes object that represents the specified decoration view.

Identifying the Referenced Item

`var indexPath: IndexPath`

The index path of the item in the collection view.

`var representedElementCategory: UICollectionView.ElementCategory`

The type of the item.

`var representedElementKind: String?`

The layout-specific identifier for the target view.

Accessing the Layout Attributes

`var frame: CGRect`
The frame rectangle of the item.

`var bounds: CGRect`
The bounds of the item.

`var center: CGPoint`
The center point of the item.

`var size: CGSize`
The size of the item.

`var transform3D: CATransform3D`
The 3D transform of the item.

`var transform: CGAffineTransform`
The affine transform of the item.

`var alpha: CGFloat`
The transparency of the item.

`var zIndex: Int`
Specifies the item's position on the z axis.

`var isHidden: Bool`
Determines whether the item is currently displayed.

Constants

`enum UICollectionView.ElementCategory`
Constants specifying the type of view.

Protocol

UICollectionViewDataSource

Getting Item and Section Metrics

`func collectionView(UICollectionView, numberOfItemsInSection: Int) -> Int`
Asks your data source object for the number of items in the specified section.
Required.

`func numberOfSections(in: UICollectionView) -> Int`
Asks your data source object for the number of sections in the collection view.

Getting Views for Items

`func collectionView(UICollectionView, cellForItemAt: IndexPath) -> UICollectionViewCell`

Asks your data source object for the cell that corresponds to the specified item in the collection view.

Required.

`func collectionView(UICollectionView, viewForSupplementaryElementOfKind: String, at: IndexPath) -> UICollectionReusableView`

Asks your data source object to provide a supplementary view to display in the collection view.

Reordering Items

`func collectionView(UICollectionView, canMoveItemAt: IndexPath) -> Bool`

Asks your data source object whether the specified item can be moved to another location in the collection view.

`func collectionView(UICollectionView, moveItemAt: IndexPath, to: IndexPath)`

Tells your data source object to move the specified item to its new location.

Configuring an Index

`func indexTitles(for: UICollectionView) -> [String]?`

Asks the data source to return the titles for the index items to display for the collection view.

`func collectionView(UICollectionView, indexPathForIndexTitle: String, at: Int) -> IndexPath`

Asks the data source to return the index path of a collection view item that corresponds to one of your index entries.

Protocol

UICollectionViewDataSourcePrefetching

Managing Data Prefetching

{} Prefetching Collection View Data

Load data for collection view cells before they are displayed.

`func collectionView(UICollectionView, prefetchItemsAt: [IndexPath])`

Instructs your prefetch data source object to begin preparing data for the cells at the supplied index paths.

Required.

```
func collectionView(UICollectionView, cancelPrefetchingForItemsAt:
[IndexPath])
```

Cancels a previously triggered data prefetch request.

Class

UICollectionViewController

Initializing the UICollectionViewController Object

```
init(collectionViewLayout: UICollectionViewLayout)
```

Initializes a collection view controller and configures the collection view with the provided layout.

```
init(nibName: String?, bundle: Bundle?)
```

Returns a newly initialized view controller with the nib file in the specified bundle

```
init?(coder: NSCoder)
```

Getting the Collection View

```
var collectionView: UICollectionView!
```

The collection view object managed by this view controller.

```
var collectionViewLayout: UICollectionViewLayout
```

The layout object used to initialize the collection view controller.

Configuring the Collection View Behavior

```
var clearsSelectionOnViewWillAppear: Bool
```

A Boolean value indicating if the controller clears the selection when the collection view appears.

```
var installsStandardGestureForInteractiveMovement: Bool
```

A Boolean value indicating whether the collection view controller installs a standard gesture recognizer to drive the reordering process.

Integrating with a Navigation Controller

```
var useLayoutToLayoutNavigationTransitions: Bool
```

A Boolean that indicates whether the collection view controller coordinates with a navigation controller for transitions.