UIScrollView

Topics

Creating a View Object

`init(frame: CGRect)`

Initializes and returns a newly allocated view object with the specified frame rectangle.

`init?(coder: NSCoder)`

Configuring a View's Visual Appearance

`var backgroundColor: UIColor?`
The view's background color.

`var isHidden: Bool`
A Boolean value that determines whether the view is hidden.

`var alpha: CGFloat`
The view's alpha value.

`var isOpaque: Bool`
A Boolean value that determines whether the view is opaque.

`var tintColor: UIColor!`
The first nondefault tint color value in the view's hierarchy, ascending from and starting with the view itself.

`var tintAdjustmentMode: UIView.TintAdjustmentMode`
The first non-default tint adjustment mode value in the view's hierarchy, ascending from and starting with the view itself.

`var clipsToBounds: Bool`
A Boolean value that determines whether subviews are confined to the bounds of the view.

`var clearsContextBeforeDrawing: Bool`
A Boolean value that determines whether the view's bounds should be automatically cleared before drawing.

`var mask: UIView?`
An optional view whose alpha channel is used to mask a view's content.

`class var layerClass: AnyClass`
Returns the class used to create the layer for instances of this class.

`var layer: CALayer`
The view's Core Animation layer used for rendering.

Configuring the Event-Related Behavior

`var isUserInteractionEnabled: Bool`
A Boolean value that determines whether user events are ignored and removed from the event queue.

`var isMultipleTouchEnabled: Bool`
A Boolean value that indicates whether the view receives more than one touch at a time.

`var isExclusiveTouch: Bool`
A Boolean value that indicates whether the receiver handles touch events exclusively.

Configuring the Bounds and Frame Rectangles

```
var frame: CGRect
```

The frame rectangle, which describes the view’s location and size in its superview’s coordinate system.

```
var bounds: CGRect
```

The bounds rectangle, which describes the view’s location and size in its own coordinate system.

```
var center: CGPoint
```

The center point of the view's frame rectangle.

```
var transform: CGAffineTransform
```

Specifies the transform applied to the view, relative to the center of its bounds.

Managing the View Hierarchy

`var superview: UIView?`

The receiver's superview, or `nil` if it has none.

`var subviews: [UIView]`

The receiver's immediate subviews.

`var window: UIWindow?`

The receiver's window object, or `nil` if it has none.

`func addSubview(UIView)`

Adds a view to the end of the receiver's list of subviews.

`func bringSubviewToFront(UIView)`

Moves the specified subview so that it appears on top of its siblings.

`func sendSubviewToBack(UIView)`

Moves the specified subview so that it appears behind its siblings.

`func removeFromSuperview()`

Unlinks the view from its superview and its window, and removes it from the responder chain.

`func insertSubview(UIView, at: Int)`

Inserts a subview at the specified index.

`func insertSubview(UIView, aboveSubview: UIView)`

Inserts a view above another view in the view hierarchy.

`func insertSubview(UIView, belowSubview: UIView)`

Inserts a view below another view in the view hierarchy.

`func exchangeSubview(at: Int, withSubviewAt: Int)`

Exchanges the subviews at the specified indices.

`func isDescendant(of: UIView) -> Bool`

Returns a Boolean value indicating whether the receiver is a subview of a given view or identical to that view.

Observing View-Related Changes

`func didAddSubview(UIView)`
Tells the view that a subview was added.

`func willRemoveSubview(UIView)`
Tells the view that a subview is about to be removed.

`func willMove(toSuperview: UIView?)`
Tells the view that its superview is about to change to the specified superview.

`func didMoveToSuperview()`
Tells the view that its superview changed.

`func willMove(toWindow: UIWindow?)`
Tells the view that its window object is about to change.

`func didMoveToWindow()`
Tells the view that its window object changed.

Configuring Content Margins

Positioning Content Within Layout Margins
Position views so that they are not crowded by other content.

`var directionalLayoutMargins: NSDirectionalEdgeInsets`
The default spacing to use when laying out content in a view, taking into account the current language direction.

`var layoutMargins: UIEdgeInsets`
The default spacing to use when laying out content in the view.

`var preservesSuperviewLayoutMargins: Bool`
A Boolean value indicating whether the current view also respects the margins of its superview.

`func layoutMarginsDidChange()`
Notifies the view that the layout margins changed.

Getting the Safe Area

Positioning Content Relative to the Safe Area

Position views so that they are not obstructed by other content.

`var safeAreaInsets: UIEdgeInsets`

The insets that you use to determine the safe area for this view.

`var safeAreaLayoutGuide: UILayoutGuide`

The layout guide representing the portion of your view that is unobscured by bars and other content.

`func safeAreaInsetsDidChange()`

Called when the safe area of the view changes.

`var insetsLayoutMarginsFromSafeArea: Bool`

A Boolean value indicating whether the view's layout margins are updated automatically to reflect the safe area.

Managing the View’s Constraints

Adjust the size and position of the view using Auto Layout constraints.

`var constraints: [NSLayoutConstraint]`

The constraints held by the view.

`func addConstraint(NSLayoutConstraint)`

Adds a constraint on the layout of the receiving view or its subviews.

`func addConstraints([NSLayoutConstraint])`

Adds multiple constraints on the layout of the receiving view or its subviews.

`func removeConstraint(NSLayoutConstraint)`

Removes the specified constraint from the view.

`func removeConstraints([NSLayoutConstraint])`

Removes the specified constraints from the view.

Creating Constraints Using Layout Anchors

Attach Auto Layout constraints to one of the view's anchors.

`var bottomAnchor: NSLayoutYAxisAnchor`
A layout anchor representing the bottom edge of the view’s frame.

`var centerXAnchor: NSLayoutXAxisAnchor`
A layout anchor representing the horizontal center of the view’s frame.

`var centerYAnchor: NSLayoutYAxisAnchor`
A layout anchor representing the vertical center of the view’s frame.

`var firstBaselineAnchor: NSLayoutYAxisAnchor`
A layout anchor representing the baseline for the topmost line of text in the view.

`var heightAnchor: NSLayoutDimension`
A layout anchor representing the height of the view’s frame.

`var lastBaselineAnchor: NSLayoutYAxisAnchor`
A layout anchor representing the baseline for the bottommost line of text in the view.

`var leadingAnchor: NSLayoutXAxisAnchor`
A layout anchor representing the leading edge of the view’s frame.

`var leftAnchor: NSLayoutXAxisAnchor`
A layout anchor representing the left edge of the view’s frame.

`var rightAnchor: NSLayoutXAxisAnchor`
A layout anchor representing the right edge of the view’s frame.

`var topAnchor: NSLayoutYAxisAnchor`
A layout anchor representing the top edge of the view’s frame.

`var trailingAnchor: NSLayoutXAxisAnchor`
A layout anchor representing the trailing edge of the view’s frame.

`var widthAnchor: NSLayoutDimension`
A layout anchor representing the width of the view’s frame.

Working with Layout Guides

`func addLayoutGuide(UILayoutGuide)`

Adds the specified layout guide to the view.

`var layoutGuides: [UILayoutGuide]`

The array of layout guide objects owned by this view.

`var layoutMarginsGuide: UILayoutGuide`

A layout guide representing the view's margins.

`var readableContentGuide: UILayoutGuide`

A layout guide representing an area with a readable width within the view.

`func removeLayoutGuide(UILayoutGuide)`

Removes the specified layout guide from the view.

Measuring in Auto Layout

`func systemLayoutSizeFitting(CGSize) -> CGSize`

Returns the optimal size of the view based on its current constraints.

```
func systemLayoutSizeFitting(CGSize, withHorizontalFittingPriority:
UILayoutPriority, verticalFittingPriority: UILayoutPriority) ->
CGSize
```

Returns the optimal size of the view based on its constraints and the specified fitting priorities.

`var intrinsicContentSize: CGSize`

The natural size for the receiving view, considering only properties of the view itself.

`func invalidateIntrinsicContentSize()`

Invalidates the view's intrinsic content size.

```
func contentCompressionResistancePriority(for: NSLayoutConstraint
.Axis) -> UILayoutPriority
```

Returns the priority with which a view resists being made smaller than its intrinsic size.

```
func setContentCompressionResistancePriority(UILayoutPriority, for:
NSLayoutConstraint.Axis)
```

Sets the priority with which a view resists being made smaller than its intrinsic size.

```
func contentHuggingPriority(for: NSLayoutConstraint.Axis) ->
UILayoutPriority
```

Returns the priority with which a view resists being made larger than its intrinsic size.

```
func setContentHuggingPriority(UILayoutPriority, for: NSLayout
Constraint.Axis)
```

Sets the priority with which a view resists being made larger than its intrinsic size.

Aligning Views in Auto Layout

`func alignmentRect(forFrame: CGRect) -> CGRect`

Returns the view’s alignment rectangle for a given frame.

`func frame(forAlignmentRect: CGRect) -> CGRect`

Returns the view’s frame for a given alignment rectangle.

`var alignmentRectInsets: UIEdgeInsets`

The insets from the view’s frame that define its alignment rectangle.

~~`func forBaselineLayout() -> UIView`~~

Returns a view used to satisfy baseline constraints.

Deprecated

`var forFirstBaselineLayout: UIView`

Returns a view used to satisfy first baseline constraints.

`var forLastBaselineLayout: UIView`

Returns a view used to satisfy last baseline constraints.

Triggering Auto Layout

`func needsUpdateConstraints() -> Bool`

A Boolean value that determines whether the view’s constraints need updating.

`func setNeedsUpdateConstraints()`

Controls whether the view’s constraints need updating.

`func updateConstraints()`

Updates constraints for the view.

`func updateConstraintsIfNeeded()`

Updates the constraints for the receiving view and its subviews.

Debugging Auto Layout

See Auto Layout Guide for more details on debugging constraint-based layout.

`func constraintsAffectingLayout(for: NSLayoutConstraint.Axis) -> [NSLayoutConstraint]`

Returns the constraints impacting the layout of the view for a given axis.

`var hasAmbiguousLayout: Bool`

A Boolean value that determines whether the constraints impacting the layout of the view incompletely specify the location of the view.

`func exerciseAmbiguityInLayout()`

Randomly changes the frame of a view with an ambiguous layout between the different valid values.

Configuring the Resizing Behavior

Define how a view adjusts its content when its bounds change.

`var contentMode: UIView.ContentMode`

A flag used to determine how a view lays out its content when its bounds change.

`enum UIView.ContentMode`

Options to specify how a view adjusts its content when its size changes.

`func sizeThatFits(CGSize) -> CGSize`

Asks the view to calculate and return the size that best fits the specified size.

`func sizeToFit()`

Resizes and moves the receiver view so it just encloses its subviews.

`var autoresizesSubviews: Bool`

A Boolean value that determines whether the receiver automatically resizes its subviews when its bounds change.

`var autoresizingMask: UIView.AutoresizingMask`

An integer bit mask that determines how the receiver resizes itself when its superview's bounds change.

Laying out Subviews

Lay out views manually if your app does not use Auto Layout.

`func layoutSubviews()`

Lays out subviews.

`func setNeedsLayout()`

Invalidates the current layout of the receiver and triggers a layout update during the next update cycle.

`func layoutIfNeeded()`

Lays out the subviews immediately, if layout updates are pending.

`class var requiresConstraintBasedLayout: Bool`

A Boolean value that indicates whether the receiver depends on the constraint-based layout system.

`var translatesAutoresizingMaskIntoConstraints: Bool`

A Boolean value that determines whether the view’s autoresizing mask is translated into Auto Layout constraints.

Managing the User Interface Direction

`var semanticContentAttribute: UISemanticContentAttribute`

A semantic description of the view’s contents, used to determine whether the view should be flipped when switching between left-to-right and right-to-left layouts.

`var effectiveUserInterfaceLayoutDirection: UIUserInterfaceLayoutDirection`

The user interface layout direction appropriate for arranging the immediate content of the view.

`class func userInterfaceLayoutDirection(for: UISemanticContentAttribute) -> UIUserInterfaceLayoutDirection`

Returns the user interface direction for the given semantic content attribute.

`class func userInterfaceLayoutDirection(for: UISemanticContentAttribute, relativeTo: UIUserInterfaceLayoutDirection) -> UIUserInterfaceLayoutDirection`

Returns the layout direction implied by the specified semantic content attribute, relative to the specified layout direction.

Adding and Removing Interactions

`func addInteraction(UIInteraction)`
Adds an interaction to the view.

`func removeInteraction(UIInteraction)`
Removes an interaction from the view.

`var interactions: [UIInteraction]`
The array of interactions for the view.

`protocol UIInteraction`
The protocol that an interaction implements to access the view that owns it.

Drawing and Updating the View

`func draw(CGRect)`
Draws the receiver’s image within the passed-in rectangle.

`func setNeedsDisplay()`
Marks the receiver’s entire bounds rectangle as needing to be redrawn.

`func setNeedsDisplay(CGRect)`
Marks the specified rectangle of the receiver as needing to be redrawn.

`var contentScaleFactor: CGFloat`
The scale factor applied to the view.

`func tintColorDidChange()`
Called by the system when the `tintColor` property changes.

Formatting Printed View Content

`func viewPrintFormatter() -> UIViewPrintFormatter`
Returns a print formatter for the receiving view.

`func draw(CGRect, for: UIViewPrintFormatter)`
Implemented to draw the view’s content for printing.

Managing Gesture Recognizers

`func addGestureRecognizer(UIGestureRecognizer)`
Attaches a gesture recognizer to the view.

`func removeGestureRecognizer(UIGestureRecognizer)`
Detaches a gesture recognizer from the receiving view.

`var gestureRecognizers: [UIGestureRecognizer]?`
The gesture-recognizer objects currently attached to the view.

`func gestureRecognizerShouldBegin(UIGestureRecognizer) -> Bool`
Asks the view if the gesture recognizer should be allowed to continue tracking touch events.

Observing Focus

`var canBecomeFocused: Bool`
A Boolean value that indicates whether the view is currently capable of being focused.

`class var inheritedAnimationDuration: TimeInterval`
Returns the inherited duration of the current animation.

`var isFocused: Bool`
A Boolean value that indicates whether the item is currently focused.

Using Motion Effects

`func addMotionEffect(UIMotionEffect)`
Begins applying a motion effect to the view.

`var motionEffects: [UIMotionEffect]`
The array of motion effects for the view.

`func removeMotionEffect(UIMotionEffect)`
Stops applying a motion effect to the view.

Preserving and Restoring State

`var restorationIdentifier: String?`
The identifier that determines whether the view supports state restoration.

`func encodeRestorableState(with: NSCoder)`
Encodes state-related information for the view.

`func decodeRestorableState(with: NSCoder)`
Decodes and restores state-related information for the view.

Capturing a View Snapshot

```
func snapshotView(afterScreenUpdates: Bool) -> UIView?
```

Returns a snapshot view based on the contents of the current view.

```
func resizableSnapshotView(from: CGRect, afterScreenUpdates: Bool,
withCapInsets: UIEdgeInsets) -> UIView?
```

Returns a snapshot view based on the specified contents of the current view, with stretchable insets.

```
func drawHierarchy(in: CGRect, afterScreenUpdates: Bool) -> Bool
```

Renders a snapshot of the complete view hierarchy as visible onscreen into the current context.

Identifying the View at Runtime

```
var tag: Int
```

An integer that you can use to identify view objects in your application.

```
func viewWithTag(Int) -> UIView?
```

Returns the view whose tag matches the specified value.

Converting Between View Coordinate Systems

```
func convert(CGPoint, to: UIView?) -> CGPoint
```

Converts a point from the receiver’s coordinate system to that of the specified view.

```
func convert(CGPoint, from: UIView?) -> CGPoint
```

Converts a point from the coordinate system of a given view to that of the receiver.

```
func convert(CGRect, to: UIView?) -> CGRect
```

Converts a rectangle from the receiver’s coordinate system to that of another view.

```
func convert(CGRect, from: UIView?) -> CGRect
```

Converts a rectangle from the coordinate system of another view to that of the receiver.

Hit Testing in a View

```
func hitTest(CGPoint, with: UIEvent?) -> UIView?
```

Returns the farthest descendant of the receiver in the view hierarchy (including itself) that contains a specified point.

```
func point(inside: CGPoint, with: UIEvent?) -> Bool
```

Returns a Boolean value indicating whether the receiver contains the specified point.

Ending a View Editing Session

```
func endEditing(Bool) -> Bool
```

Causes the view (or one of its embedded text fields) to resign the first responder status.

Modifying the Accessibility Behavior

`var accessibilityIgnoresInvertColors: Bool`

A Boolean value indicating whether the view ignores an accessibility request to invert its colors.

Animating Views with Block Objects

Use of these methods is discouraged. Use the `UIViewPropertyAnimator` class to perform animations instead.

```
class func animate(withDuration: TimeInterval, delay: TimeInterval,
options: UIView.AnimationOptions = [], animations: () -> Void,
completion: ((Bool) -> Void)? = nil)
```

Animate changes to one or more views using the specified duration, delay, options, and completion handler.

```
class func animate(withDuration: TimeInterval, animations: () ->
Void, completion: ((Bool) -> Void)? = nil)
```

Animate changes to one or more views using the specified duration and completion handler.

```
class func animate(withDuration: TimeInterval, animations: () ->
Void)
```

Animate changes to one or more views using the specified duration.

```
class func transition(with: UIView, duration: TimeInterval,
options: UIView.AnimationOptions = [], animations: (() -> Void)?,
completion: ((Bool) -> Void)? = nil)
```

Creates a transition animation for the specified container view.

```
class func transition(from: UIView, to: UIView, duration: Time
Interval, options: UIView.AnimationOptions = [], completion:
((Bool) -> Void)? = nil)
```

Creates a transition animation between the specified views using the given parameters.

```
class func animateKeyframes(withDuration: TimeInterval, delay: Time
Interval, options: UIView.KeyframeAnimationOptions = [],
animations: () -> Void, completion: ((Bool) -> Void)? = nil)
```

Creates an animation block object that can be used to set up keyframe-based animations for the current view.

```
class func addKeyframe(withRelativeStartTime: Double, relative
Duration: Double, animations: () -> Void)
```

Specifies the timing and animation values for a single frame of a keyframe animation.

```
class func perform(UIView.SystemAnimation, on: [UIView], options:
UIView.AnimationOptions = [], animations: (() -> Void)?,
completion: ((Bool) -> Void)? = nil)
```

Performs a specified system-provided animation on one or more views, along with optional parallel animations that you define.

`class func animate(withDuration: TimeInterval, delay: TimeInterval, usingSpringWithDamping: CGFloat, initialSpringVelocity: CGFloat, options: UIView.AnimationOptions = [], animations: () -> Void, completion: ((Bool) -> Void)? = nil)`

Performs a view animation using a timing curve corresponding to the motion of a physical spring.

`class func performWithoutAnimation(() -> Void)`

Disables a view transition animation.

Animating Views

Use of these methods is discouraged. Use the `UIViewPropertyAnimator` class to perform animations instead.

`class func beginAnimations(String?, context: UnsafeMutableRawPointer?)`

Marks the beginning of a begin/commit animation block.

`class func commitAnimations()`

Marks the end of a begin/commit animation block and schedules the animations for execution.

`class func setAnimationStart(Date)`

Sets the start time for the current animation block.

`class func setAnimationsEnabled(Bool)`

Sets whether animations are enabled.

`class func setAnimationDelegate(Any?)`

Sets the delegate for any animation messages.

`class func setAnimationWillStart(Selector?)`

Sets the message to send to the animation delegate when the animation starts.

`class func setAnimationDidStop(Selector?)`

Sets the message to send to the animation delegate when animation stops.

`class func setAnimationDuration(TimeInterval)`

Sets the duration (measured in seconds) of the animations in an animation block.

`class func setAnimationDelay(TimeInterval)`

Sets the amount of time (in seconds) to wait before animating property changes within an animation block.

`class func setAnimationCurve(UIView.AnimationCurve)`

Sets the curve to use when animating property changes within an animation block.

`class func setAnimationRepeatCount(Float)`

Sets the number of times animations within an animation block repeat.

`class func setAnimationRepeatAutoreverses(Bool)`

Sets whether the animations within an animation block automatically reverse themselves.

`class func setAnimationBeginsFromCurrentState(Bool)`

Sets whether the animation should begin playing from the current state.

`class func setAnimationTransition(UIView.AnimationTransition, for: UIView, cache: Bool)`

Sets a transition to apply to a view during an animation block.

`class var areAnimationsEnabled: Bool`

Returns a Boolean value indicating whether animations are enabled.

Constants

`struct UIView.AnimationOptions`

Options for animating views using block objects.

`enum UIView.AnimationCurve`

Specifies the supported animation curves.

`enum UIView.AnimationTransition`

Animation transition options for use in an animation block object.

`enum UIView.SystemAnimation`

Option to remove the views from the hierarchy when animation is complete.

`struct UIView.KeyframeAnimationOptions`

Key frame animation options used with the `animateKeyframes(withDuration:delay:options:animations:completion:)` method.

`enum NSLayoutConstraint.Axis`

Keys that specify a horizontal or vertical layout constraint between objects.

`enum UIView.TintAdjustmentMode`

The tint adjustment mode for the view.

`class let layoutFittingCompressedSize: CGSize`

The option to use the smallest possible size.

`class let layoutFittingExpandedSize: CGSize`

The option to use the largest possible size.

`class let noIntrinsicMetric: CGFloat`

The absence of an intrinsic metric for a given numeric view property.

`struct UIView.AutoresizingMask`

Options for automatic view resizing.

`enum UISemanticContentAttribute`

A semantic description of the view’s contents, used to determine whether the view should be flipped when switching between left-to-right and right-to-left layouts.

Instance Properties

`var playgroundLiveViewRepresentation: PlaygroundLiveViewRepresentation`

Creating a View Controller Programmatically

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

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

`init?(coder: NSCoder)`

Interacting with Storyboards and Segues

`var storyboard: UIStoryboard?`

The storyboard from which the view controller originated.

`func shouldPerformSegue(withIdentifier: String, sender: Any?) -> Bool`

Determines whether the segue with the specified identifier should be performed.

`func prepare(for: UIStoryboardSegue, sender: Any?)`

Notifies the view controller that a segue is about to be performed.

`func performSegue(withIdentifier: String, sender: Any?)`

Initiates the segue with the specified identifier from the current view controller's storyboard file.

`func allowedChildrenForUnwinding(from: UIStoryboardUnwindSegueSource) -> [UIViewController]`

Returns an array of child view controllers that should be searched for an unwind segue destination.

`func childContaining(UIStoryboardUnwindSegueSource) -> UIViewController?`

Returns the child view controller that contains the source of the unwind segue.

`func canPerformUnwindSegueAction(Selector, from: UIViewController, withSender: Any) -> Bool`

Called on a view controller to determine whether it wants to respond to an unwind action.

`func unwind(for: UIStoryboardSegue, towards: UIViewController)`

Called when an unwind segue transitions to a new view controller.

Managing the View

`var view: UIView!`

The view that the controller manages.

`var isViewLoaded: Bool`

A Boolean value indicating whether the view is currently loaded into memory.

`func loadView()`

Creates the view that the controller manages.

`func viewDidLoad()`

Called after the controller's view is loaded into memory.

`func loadViewIfNeeded()`

Loads the view controller’s view if it has not yet been loaded.

`var viewIfLoaded: UIView?`

The view controller’s view, or `nil` if the view is not yet loaded.

`var title: String?`

A localized string that represents the view this controller manages.

`var preferredContentSize: CGSize`

The preferred size for the view controller’s view.

Presenting View Controllers

`var modalPresentationStyle: UIModalPresentationStyle`

The presentation style for modally presented view controllers.

`var modalTransitionStyle: UIModalTransitionStyle`

The transition style to use when presenting the view controller.

`var isModalInPopover: Bool`

A Boolean value indicating whether the view controller should be presented modally by a popover.

`func show(UIViewController, sender: Any?)`

Presents a view controller in a primary context.

`func showDetailViewController(UIViewController, sender: Any?)`

Presents a view controller in a secondary (or detail) context.

```
func present(UIViewController, animated: Bool, completion: (() ->
Void)? = nil)
```

Presents a view controller modally.

`func dismiss(animated: Bool, completion: (() -> Void)? = nil)`

Dismisses the view controller that was presented modally by the view controller.

`var definesPresentationContext: Bool`

A Boolean value that indicates whether this view controller's view is covered when the view controller or one of its descendants presents a view controller.

`var providesPresentationContextTransitionStyle: Bool`

A Boolean value that indicates whether the view controller specifies the transition style for view controllers it presents.

`var disablesAutomaticKeyboardDismissal: Bool`

Returns a Boolean indicating whether the current input view is dismissed automatically when changing controls.

Supporting Custom Transitions and Presentations

```
var transitioningDelegate: UIViewControllerTransitioningDelegate?
```

The delegate object that provides transition animator, interactive controller, and custom presentation controller objects.

```
var transitionCoordinator: UIViewControllerTransitionCoordinator?
```

Returns the active transition coordinator object.

```
func targetViewController(forAction: Selector, sender: Any?) ->
UIViewController?
```

Returns the view controller that responds to the action.

```
var presentationController: UIPresentationController?
```

The nearest presentation controller that is managing the current view controller.

```
var popoverPresentationController: UIPopoverPresentationController?
```

The nearest popover presentation controller that is managing the current view controller.

```
var restoresFocusAfterTransition: Bool
```

A Boolean value that indicates whether an item that previously was focused should again become focused when the item's view controller becomes visible and focusable.

Responding to View Events

`func viewWillAppear(Bool)`

Notifies the view controller that its view is about to be added to a view hierarchy.

`func viewDidAppear(Bool)`

Notifies the view controller that its view was added to a view hierarchy.

`func viewWillDisappear(Bool)`

Notifies the view controller that its view is about to be removed from a view hierarchy.

`func viewDidDisappear(Bool)`

Notifies the view controller that its view was removed from a view hierarchy.

`var isBeingDismissed: Bool`

A Boolean value indicating whether the view controller is being dismissed.

`var isBeingPresented: Bool`

A Boolean value indicating whether the view controller is being presented.

`var isMovingFromParent: Bool`

A Boolean value indicating whether the view controller is being removed from a parent view controller.

`var isMovingToParent: Bool`

A Boolean value indicating whether the view controller is being moved to a parent view controller.

Extending the View's Safe Area

Positioning Content Relative to the Safe Area

Position views so that they are not obstructed by other content.

`var additionalSafeAreaInsets: UIEdgeInsets`

Custom insets that you specify to modify the view controller's safe area.

`func viewSafeAreaInsetsDidChange()`

Called to notify the view controller that the safe area insets of its root view changed.

Managing the View's Margins

Positioning Content Within Layout Margins
Position views so that they are not crowded by other content.

`var viewRespectsSystemMinimumLayoutMargins: Bool`
A Boolean value indicating whether the view controller's view uses the system-defined minimum layout margins.

`var systemMinimumLayoutMargins: NSDirectionalEdgeInsets`
The minimum layout margins for the view controller's root view.

`func viewLayoutMarginsDidChange()`
Called to notify the view controller that the layout margins of its root view changed.

Configuring the View’s Layout Behavior

`var edgesForExtendedLayout: UIRectEdge`
The edges that you extend for your view controller.

`var extendedLayoutIncludesOpaqueBars: Bool`
A Boolean value indicating whether or not the extended layout includes opaque bars.

`func viewWillLayoutSubviews()`
Called to notify the view controller that its view is about to layout its subviews.

`func viewDidLayoutSubviews()`
Called to notify the view controller that its view has just laid out its subviews.

`func updateViewConstraints()`
Called when the view controller's view needs to update its constraints.

Configuring the View Rotation Settings

`var shouldAutorotate: Bool`
Returns a Boolean value indicating whether the view controller's contents should auto rotate.

`var supportedInterfaceOrientations: UIInterfaceOrientationMask`
Returns all of the interface orientations that the view controller supports.

`var preferredInterfaceOrientationForPresentation: UIInterface Orientation`
Returns the interface orientation to use when presenting the view controller.

`class func attemptRotationToDeviceOrientation()`
Attempts to rotate all windows to the orientation of the device.

Adapting to Environment Changes

```
func collapseSecondaryViewController(UIViewController, for: UISplit
ViewController)
```

Called when a split view controller transitions to a compact-width size class.

```
func separateSecondaryViewController(for: UISplitViewController) ->
UIViewController?
```

Called when a split view controller transitions to a regular-width size class.

Adjusting the Interface Style

```
var preferredUserInterfaceStyle: UIUserInterfaceStyle
```

The preferred interface style for this view controller.

```
var childViewControllerForUserInterfaceStyle: UIViewController?
```

The child view controller that supports the preferred user interface style.

```
func setNeedsUserInterfaceAppearanceUpdate()
```

Notifies the view controller that a change occurred that might affect the preferred interface style.

```
enum UIUserInterfaceStyle
```

Constants indicating the interface style for the app.

Managing Child View Controllers in a Custom Container

```
var children: [UIViewController]
```

An array of view controllers that are children of the current view controller.

```
func addChild(UIViewController)
```

Adds the specified view controller as a child of the current view controller.

```
func removeFromParent()
```

Removes the view controller from its parent.

```
func transition(from: UIViewController, to: UIViewController,
duration: TimeInterval, options: UIView.AnimationOptions = [],
animations: (() -> Void)?, completion: ((Bool) -> Void)? = nil)
```

Transitions between two of the view controller's child view controllers.

```
var shouldAutomaticallyForwardAppearanceMethods: Bool
```

Returns a Boolean value indicating whether appearance methods are forwarded to child view controllers.

```
func beginAppearanceTransition(Bool, animated: Bool)
```

Tells a child controller its appearance is about to change.

```
func endAppearanceTransition()
```

Tells a child controller its appearance has changed.

```
func setOverrideTraitCollection(UITraitCollection?, forChild:
UIViewController)
```

Changes the traits assigned to the specified child view controller.

```
func overrideTraitCollection(forChild: UIViewController) -> UITrait
Collection?
```

Retrieves the trait collection for a child view controller.

Responding to Containment Events

```
func willMove(toParent: UIViewController?)
```

Called just before the view controller is added or removed from a container view controller.

```
func didMove(toParent: UIViewController?)
```

Called after the view controller is added or removed from a container view controller.

Getting Other Related View Controllers

`var presentingViewController: UIViewController?`

The view controller that presented this view controller.

`var presentedViewController: UIViewController?`

The view controller that is presented by this view controller, or one of its ancestors in the view controller hierarchy.

`var parent: UIViewController?`

The parent view controller of the recipient.

`var navigationController: UINavigationController?`

The nearest ancestor in the view controller hierarchy that is a navigation controller.

`var splitViewController: UISplitViewController?`

The nearest ancestor in the view controller hierarchy that is a split view controller.

`var tabBarController: UITabBarController?`

The nearest ancestor in the view controller hierarchy that is a tab bar controller.

Handling Memory Warnings

`func didReceiveMemoryWarning()`

Sent to the view controller when the app receives a memory warning.

Managing State Restoration

`var restorationIdentifier: String?`

The identifier that determines whether the view controller supports state restoration.

`var restorationClass: UIViewControllerRestoration.Type?`

The class responsible for recreating this view controller when restoring the app's state.

`func encodeRestorableState(with: NSCoder)`

Encodes state-related information for the view controller.

`func decodeRestorableState(with: NSCoder)`

Decodes and restores state-related information for the view controller.

`func applicationFinishedRestoringState()`

Called on restored view controllers after other object decoding is complete.

Supporting App Extensions

`var extensionContext: NSExtensionContext?`

Returns the extension context of the view controller.

Working With 3D Touch Previews and Preview Quick Actions

The methods in this task group are available on devices that support 3D Touch. The end-user terminology for the views presented during the phases of force-based touches includes *peek*and *pop*. For clarity here, and to align with the API names, this document uses the corresponding terms *preview* and *commit view*. To learn more about 3D Touch, read Adopting 3D Touch on iPhone.

```
func registerForPreviewing(with: UIViewControllerPreviewing
Delegate, sourceView: UIView) -> UIViewControllerPreviewing
```

Registers a view controller to participate with 3D Touch preview (peek) and commit (pop).

```
func unregisterForPreviewing(withContext: UIViewController
Previewing)
```

Unregisters a previously registered view controller identified by its context object.

```
var previewActionItems: [UIPreviewActionItem]
```

The quick actions displayed when a user swipes upward on a 3D Touch preview.

Coordinating with System Gesture Recognizers

```
var preferredScreenEdgesDeferringSystemGestures: UIRectEdge
```

The screen edges for which you want your gestures to take precedence over the system gestures

```
var childForScreenEdgesDeferringSystemGestures: UIViewController?
```

Returns the child view controller that should be queried to see if its gestures should take precedence.

```
func setNeedsUpdateOfScreenEdgesDeferringSystemGestures()
```

Call this method when you change the screen edges that you use for deferring system gestures.

Managing the Status Bar

`var childForStatusBarHidden: UIViewController?`

Called when the system needs the view controller to use for determining status bar hidden/unhidden state.

`var childForStatusBarStyle: UIViewController?`

Called when the system needs the view controller to use for determining status bar style.

`var preferredStatusBarStyle: UIStatusBarStyle`

The preferred status bar style for the view controller.

`var prefersStatusBarHidden: Bool`

Specifies whether the view controller prefers the status bar to be hidden or shown.

`var modalPresentationCapturesStatusBarAppearance: Bool`

Specifies whether a view controller, presented non-fullscreen, takes over control of status bar appearance from the presenting view controller.

`var preferredStatusBarUpdateAnimation: UIStatusBarAnimation`

Specifies the animation style to use for hiding and showing the status bar for the view controller.

`func setNeedsStatusBarAppearanceUpdate()`

Indicates to the system that the view controller status bar attributes have changed.

Configuring Gestures

`var prefersHomeIndicatorAutoHidden: Bool`

Returns a Boolean indicating whether the system is allowed to hide the visual indicator for returning to the Home screen.

`var childForHomeIndicatorAutoHidden: UIViewController?`

Returns the child view controller that is consulted about its preference for displaying a visual indicator for returning to the Home screen.

`func setNeedsUpdateOfHomeIndicatorAutoHidden()`

Notifies UIKit that your view controller updated its preference regarding the visual indicator for returning to the Home screen.

Configuring a Navigation Interface

`var navigationItem: UINavigationItem`

The navigation item used to represent the view controller in a parent's navigation bar.

`var hidesBottomBarWhenPushed: Bool`

A Boolean value indicating whether the toolbar at the bottom of the screen is hidden when the view controller is pushed on to a navigation controller.

`func setToolbarItems([UIBarButtonItem]?, animated: Bool)`

Sets the toolbar items to be displayed along with the view controller.

`var toolbarItems: [UIBarButtonItem]?`

The toolbar items associated with the view controller.

Configuring Tab Bar Items

`var tabBarItem: UITabBarItem!`

The tab bar item that represents the view controller when added to a tab bar controller.

Adding Editing Behaviors to Your View Controller

`var isEditing: Bool`

A Boolean value indicating whether the view controller currently allows the user to edit the view contents.

`func setEditing(Bool, animated: Bool)`

Sets whether the view controller shows an editable view.

`var editButtonItem: UIBarButtonItem`

Returns a bar button item that toggles its title and associated state between Edit and Done.

Accessing the Available Key Commands

`func addKeyCommand(UIKeyCommand)`

Associates the specified keyboard shortcut with the view controller.

`func removeKeyCommand(UIKeyCommand)`

Removes the key command from the view controller.

Getting Nib File Information

`var nibName: String?`

The name of the view controller's nib file, if one was specified.

`var nibBundle: Bundle?`

The view controller's nib bundle if it exists.

Constants

`enum UIModalPresentationStyle`

Modal presentation styles available when presenting view controllers.

`enum UIModalTransitionStyle`

Transition styles available when presenting view controllers.

~~`enum ADInterstitialPresentationPolicy`~~

Policy options governing how and when interstitial ads may be presented from a view controller.

Deprecated

Exceptions

Exceptions raised by view controllers.

`struct UIRectEdge`

Constants that specify the edges of a rectangle.

Notifications

`class let showDetailTargetDidChangeNotification: NSNotification.Name`

Posted when a split view controller is expanded or collapsed.

Deprecated

~~`func rotatingHeaderView() -> UIView?`~~

Returns the header view to transition during an interface orientation change.

Deprecated

~~`func rotatingFooterView() -> UIView?`~~

Returns the footer view to transition during an interface orientation change.

Deprecated

~~`var interfaceOrientation: UIInterfaceOrientation`~~

Convenience property that provides the current orientation of the interface, meaningful only if the view controller is taking up the full screen.

Deprecated

~~`func willRotate(to: UIInterfaceOrientation, duration: TimeInterval)`~~

Sent to the view controller just before the user interface begins rotating.

Deprecated

~~`func willAnimateRotation(to: UIInterfaceOrientation, duration: TimeInterval)`~~

Sent to the view controller before performing a one-step user interface rotation.

Deprecated

~~`func didRotate(from: UIInterfaceOrientation)`~~

Sent to the view controller after the user interface rotates.

Deprecated

~~`var searchDisplayController: UISearchDisplayController?`~~

The search display controller associated with the view controller.

Deprecated

~~`func shouldAutomaticallyForwardRotationMethods() -> Bool`~~

Returns a Boolean value indicating whether rotation methods are forwarded to child view controllers.

Deprecated

~~`func presentMoviePlayerViewControllerAnimated(MPMoviePlayerViewController!)`~~

Presents the movie player view controller using the standard movie player transition.

Deprecated

~~`func dismissMoviePlayerViewControllerAnimated()`~~

Dismisses a movie player view controller using the standard movie player transition.

Deprecated

~~`func forUnwindSegueAction(Selector, from: UIViewController, withSender: Any?) -> UIViewController?`~~

Called when an unwind segue action wants to search a container's children for a view controller to handle the unwind action.

Deprecated

~~`func segueForUnwinding(to: UIViewController, from: UIViewController, identifier: String?) -> UIStoryboardSegue?`~~

Called when an unwind segue action needs to transition between two view controllers.

Deprecated

~~`var bottomLayoutGuide: UILayoutSupport`~~

Indicates the lowest vertical extent for your onscreen content, for use with Auto Layout constraints.

Deprecated

~~`var topLayoutGuide: UILayoutSupport`~~

Indicates the highest vertical extent for your onscreen content, for use with Auto Layout constraints.

Deprecated

~~`var automaticallyAdjustsScrollViewInsets: Bool`~~

A Boolean value that indicates whether the view controller should automatically adjust its scroll view insets.

Deprecated

~~`var canDisplayBannerAds: Bool`~~

A boolean value that indicates whether the view controller is configured to display banner ads.

Deprecated

~~`var originalContentView: UIView?`~~

The originally configured content view of the view controller before banner ads were enabled.

Deprecated

~~`var isPresentingFullScreenAd: Bool`~~

A boolean value that indicates whether the view controller is displaying a full-screen ad.

Deprecated

~~`var isDisplayingBannerAd: Bool`~~

A boolean value that indicates whether the view controller is displaying a banner ad.

Deprecated

~~`class func prepareInterstitialAds()`~~

Prepares the iAd framework to display interstitial ads, which may involve prefetching ad assets.

Deprecated

~~`var interstitialPresentationPolicy: ADInterstitialPresentationPolicy`~~

Determines whether interstitials should be presented at all and whether the framework or app should manage the presentation.

Deprecated

~~`func requestInterstitialAdPresentation() -> Bool`~~

Asks the framework to display an interstitial ad.

Deprecated

~~`var shouldPresentInterstitialAd: Bool`~~

Returns whether an interstitial ad should be displayed.

Deprecated

Instance Properties

`var playgroundLiveViewRepresentation: PlaygroundLiveViewRepresentation`