Sample Code

# Customizing your app's navigation bar

Create custom titles, prompts, and buttons in your app's navigation bar.

Download

iOS 26.0+ | iPadOS 26.0+ | macOS 26.0+ | tvOS 26.0+ | visionOS 26.0+ | watchOS 26.0+ | Xcode 26.0+

## Overview

When you use a navigation controller to add navigation structure to your app, you can use a view controller's `navigationItem` to configure the `UINavigationBar` with navigational and interactive controls in a bar above your content, along the top of the iOS device's screen.

This sample code project demonstrates how to use `UINavigationController` and `UIViewController` classes as building blocks in your application's user interface.

The project walks you through a set of examples that customize the look and behavior of `UINavigationController` and `UINavigationBar`, including views, prompts, buttons and titles of your application's navigation bar.

The sample demonstrates methods to modify the navigation bar through the view controller's `UINavigationItem`. This includes:

- Customizing the buttons in the bar
- Presenting a menu from a bar button
- Adjusting how buttons are grouped in the navigation bar or toolbar
- Integrating system search in the toolbar
- Updating the bar's title and subtitle
- Adding a navigation prompt
- Customizing the back button with a title or image

## Customize the right view

The right side of the navigation bar options for customization include applying a custom `UIView` or using a `UIBarButtonItem`.

The sample demonstrates placing three kinds of `UIBarButtonItems` on the right side of the navigation bar: a button with a title, a button with an image, and a button with a `UISegmented Control`. An additional segmented control allows the user to toggle between the three buttons. The initial bar button is defined in the storyboard, by dragging a `UIBarButtonItem` out of the object library and into the navigation bar. The sample also shows how to create and add each button type using code.

For example, the sample shows how to set a segmented control as the right bar button item in `CustomRightViewController`:

```
let segmentBarItem = UIBarButtonItem(customView: segmentedControl)
navigationItem.rightBarButtonItem = segmentBarItem
```

## Use default groups for buttons

When you add bar button items to a navigation item, the system groups the buttons into a shared glass background for:

- Image buttons
- Bar button item groups with multiple items

The system provides separate glass backgrounds for:

- Text buttons
- `UIBarButtonItem.SystemItem.done` buttons
- `UIBarButtonItem.SystemItem.close` buttons
- `UIBarButtonItem.Style.prominent` buttons

The sample shows an example of default groups provided by the system in `DefaultButton GroupingViewController`:

```
let selectBarButton = UIBarButtonItem(title: NSLocalizedString("Select", comment: "'
                                      style: .plain,
                                      target: nil,
                                      action: nil)
let shareBarButton = UIBarButtonItem(image: UIImage(systemName: "square.and.arrow.up
                                     style: .plain,
```

```
                                        target: nil,
                                        action: nil)
let infoBarButton = UIBarButtonItem(image: UIImage(systemName: "info.circle"),
                                    style: .plain,
                                    target: nil,
                                    action: nil)
let doneBarButton = UIBarButtonItem(barButtonSystemItem: .done,
                                    target: nil,
                                    action: nil)

navigationItem.rightBarButtonItems = [doneBarButton,
                                      shareBarButton,
                                      infoBarButton,
                                      selectBarButton]
```

## Customize button groups

When the system groups buttons into a shared glass background and you want to split the buttons into separate glass backgrounds, add `fixedSpace()` between the buttons where you want to split the glass background.

The sample demonstrates how to do this in `CustomButtonGroupsViewController`. By default, `shareBarButton` and `infoBarButton` share a glass background because they're each buttons with an image. The sample adds a `fixedSpace()` between them so they each have their own glass background:

```
navigationItem.rightBarButtonItems = [doneBarButton,
                                      shareBarButton,
                                      .fixedSpace(0),
                                      infoBarButton,
                                      selectBarButton]
```

## Customize button labels and background color

Set a bar button item's tint color to provide a color for the button's text or image. The sample demonstrates this in `CustomButtonColorsViewController`:

```
let shareBarButton = UIBarButtonItem(image: UIImage(systemName: "square.and.arrow.up
                                     style: .plain,
                                     target: nil,
                                     action: nil)
```

```
shareBarButton.tintColor = .systemOrange
```

To draw attention to a button, you can use a prominent button style so that the sytem fills the button's background with the tint color. The sample demonstrates two ways to do this. First, it sets a bar button item's style to `UIBarButtonItem.Style.prominent`:

```
let infoBarButton = UIBarButtonItem(image: UIImage(systemName: "info.circle"),
                                    style: .plain,
                                    target: nil,
                                    action: nil)
infoBarButton.tintColor = .systemOrange
infoBarButton.style = .prominent
```

Second, it uses a system item with a system default image and color:

```
let doneBarButton = UIBarButtonItem(barButtonSystemItem: .done,
                                    target: nil,
                                    action: nil)
```

## Group toolbar items that use flexible spacing

When you use flexible space-bar button items to evenly space buttons in a toolbar, the system provides separate glass backgrounds between the flexible spaces. In `CustomToolbarLayoutViewController`, the sample shows how to have all the buttons share the same glass background, by configuring the flexible space using `hidesSharedBackground` first:

```
let flexibleSpace = UIBarButtonItem.flexibleSpace()
flexibleSpace.hidesSharedBackground = false
```

Then, the sample uses the flexible space it configured to space out the buttons:

```
toolbarItems = [
    .init(image: UIImage(systemName: "location")),
    flexibleSpace,
    .init(image: UIImage(systemName: "number")),
    flexibleSpace,
    .init(image: UIImage(systemName: "camera")),
    flexibleSpace,
    .init(image: UIImage(systemName: "trash"))
]
```

## Integrate search in your toolbar

The system provides a built-in method to integrate search in your app when you use `UISearchController` and a navigation controller's toolbar. The sample demonstrates how to perform this integration in `ToolbarSystemSearchViewController`.

First, the sample instantiates a search controller:

```
searchController = UISearchController(searchResultsController: SearchResultsViewCont
```

Then, the sample tells the navigation item to use the search controller it just instantiated:

```
navigationItem.searchController = searchController
```

Finally, the sample adds the navigation item's `searchBarPlacementBarButtonItem` to the toolbar buttons:

```
toolbarItems = [
    .init(image: UIImage(systemName: "location")),
    .init(image: UIImage(systemName: "number")),
    .init(image: UIImage(systemName: "camera")),
    flexibleSpace,
    navigationItem.searchBarPlacementBarButtonItem
]
```

With those steps complete, the system shows a search button in the toolbar. When a person taps the search bar, the system displays a search interface where the person can type in a search parameter and perform searches.

## Customize the title view

Another option is configuring the navigation bar to use a `UIView` as the title, using `UISegmentedControl` as the center custom title view.

This sample shows how to set a segmented control as the title view in `CustomTitleViewController`:

```
self.navigationItem.titleView = segmentedControl
```

## Customize the title and subtitle

The navigation bar can include a title and a subtitle. The sample demonstrates setting up a title and subtitle in `TitleSubtitleViewController`:

```
navigationItem.title = "Title"
navigationItem.subtitle = "Subtitle"
```

In addition to setting the title and subtitle with strings, you can also use attributed strings or custom views.

## Customize the large subtitle view

You can set custom views to display in place of the title, subtitle, large title, or subtitle below the large title. In `LargeTitleViewController`, the sample shows an example of a button as a custom subtitle view below the large title:

```
self.navigationController?.navigationBar.prefersLargeTitles = true

var subtitleConfiguration = UIButton.Configuration.plain()
subtitleConfiguration.title = "Subtitle Button"
subtitleConfiguration.baseForegroundColor = .systemBlue

let subtitleButton = UIButton(configuration: subtitleConfiguration)
navigationItem.largeSubtitleView = subtitleButton
```

## Modify the navigation prompt

The navigation bar can also include a prompt or single line of text at the top.

The sample demonstrates how to use the `prompt` property of a `UINavigationItem` to display a custom line of text above the navigation bar in `NavigationPromptViewController`:

```
navigationItem.prompt = NSLocalizedString("Navigation prompts appear at the top.", c
```

## Customize back button titles

People can quickly switch between different stack levels with a tap and hold on the back button. The sample pushes 10 view controllers on the current navigation stack in `MainViewController` to demonstrate customizing back button titles for each view controller level in the stack.

## Customize the back button with an image

It's also possible to use an image as the back button without any back button text and without the back arrow that normally appears next to the back button. The sample sets the back button's image in `CustomBackButtonDetailViewController` like so:

```
let backButtonBackgroundImage = UIImage(systemName: "list.bullet")
let backButton = UIBarButtonItem(image: backButtonBackgroundImage,
                                 style: .plain,
                                 target: self,
                                 action: #selector(backButtonTapped(_:)))
navigationItem.leftBarButtonItem = backButton
```

## Modify the large title in the navigation bar

Another option for customizing the navigation bar includes enabling large title display mode, so it shows a larger version of the title. When the view controller contains a scroll view, the system displays the large title at the top of the scrollable content, and then animates the title into the navigation bar when a person begins to scroll.

The code below shows how the sample enables the large title display mode for a navigation bar in `LargeTitleViewController`:

```
self.navigationController?.navigationBar.prefersLargeTitles = true
```

For more information on controlling how the navigation bar displays the navigation item's title, see `largeTitleDisplayMode`.

## Attach a menu to a bar button item

A menu attachment provides extended and easy access to application features in one place. The sample attaches a UIMenu to the right side UIBarButtonItem control in BarButtonMenu:

```
let barButtonMenu = UIMenu(title: "", children: [
    UIAction(title: NSLocalizedString("Copy", comment: ""), image: UIImage(systemNam
    UIAction(title: NSLocalizedString("Rename", comment: ""), image: UIImage(systemN
    UIAction(title: NSLocalizedString("Duplicate", comment: ""), image: UIImage(syst
    UIAction(title: NSLocalizedString("Move", comment: ""), image: UIImage(systemNam
])
optionsBarItem.menu = barButtonMenu
```

# See Also

### Configuring navigation bars

`var navigationBar: UINavigationBar`

The navigation bar managed by the navigation controller.

`func setNavigationBarHidden(Bool, animated: Bool)`

Sets whether the navigation bar is hidden.