

WidgetKit

- Displaying the right widget back...
- Optimizing your widget for accent...
- Adding StandBy and CarPlay supp...
- WidgetRenderingMode
- WidgetAccentedRenderingMode
- AccessoryWidgetBackground
- WidgetLocation
- Timeline updates
- Keeping a widget up to date
- TimelineProvider
- AppIntentTimelineProvider
- IntentTimelineProvider
- TimelineProviderContext
- TimelineEntry
- Timeline
- WidgetCenter
- Push notification updates
- Updating widgets with WidgetKit p...

Filter

/

WidgetKit / Widgets and watch complications / Displaying the right widget background

Article

Displaying the right widget background

Group your widget’s background views and mark them as removable to ensure your widget appears correctly for each context and platform.

Overview

Widgets appear differently based on their context, including their background view, for example:

- In the [vibrant](#) appearance, the system removes the background of your widget or renders it with semi-translucent appearance.
- In StandBy on iPhone, the system removes the background of your widget.

To make sure your widget appears correctly, mark your background views as removable for every widget size.

Important

If a widget doesn’t support removable background views or doesn’t explicitly mark a background as nonremovable, the system displays a warning message that overlays the widget during development with Xcode.

To mark your background views as removable:

- Add the [containerBackground\(for:\)](#) modifier to your background views to define the background appearance of your widget and tell WidgetKit that it can remove the view where applicable.
- Move code that declares any background color or views inside the `containerBackground(for:)` view modifier and pass [widget](#) to it. This makes sure WidgetKit automatically removes the background as needed.

The following code snippet from the [Emoji Rangers: Supporting Live Activities, interactivity, and animations](#) sample code project defines a `gameBackground` color for the widget’s container background to make sure WidgetKit renders it with or without the background color as applicable.

```
var body: some View {
    switch family {
        // Logic for additional widget sizes.
        case .accessoryRectangular:
            HStack(alignment: .center, spacing: 0) {
                VStack(alignment: .leading) {
                    Text(entry.hero.name)
                        .font(.headline)
                        .widgetAccentable()
                    Text("Level \(entry.hero.level)")
                    Text(entry.hero.fullHealthDate, style: .timer)
                }.frame(maxWidth: .infinity, alignment: .leading)
                Avatar(hero: entry.hero, includeBackground: false)
            }
            .containerBackground(for: .widget) {
                Color.gameBackground
            }
        // Logic for additional widget sizes.
    }
}
```

Marking background views as removable by defining a background container so you can use the same layout and views for your widget across contexts. For example, if you support the rectangular accessory widget size as shown in the code snippet above, WidgetKit renders it with a rich, full-color background in the Smart Stack on Apple Watch and without a background as a watch complication or on the Lock Screen of iPhone and iPad.

On Apple Watch, rectangular widgets in the Smart Stack use a default dark material background. By adding a container background, the widget renders with a background that visually ties it to your app and makes it more recognizable to people.

To detect whether a widget appears with or without a background, use the [showsWidgetContainerBackground](#) environment variable.

Explicitly opt out of background removal

Some widgets don’t have distinct foreground content, and background removal can negatively impact their functionality. For example, a widget might display a photo or map that takes up the entirety of the widget’s bounds. Removing the photo or map removes the functionality of the widget. If this case applies to your widget, set the [containerBackgroundRemovable\(_:\)](#) modifier to `false` for your widget configuration.

Important

Marking a background as nonremovable excludes your widget from the widget gallery in contexts that require a removable background; for example, on the iPad Lock Screen and in StandBy.

Set the background color of accessory widgets

Depending on your accessory widget or complication, you may need to set a consistent background for your accessory widget. Use [AccessoryWidgetBackground](#) to draw a consistent background for your widget, as shown in the following example. It creates a view that’s similar to the circular Lock Screen widget that the Calendar app offers:

```
ZStack {
    AccessoryWidgetBackground()
    VStack {
        Text("MON")
        Text("6")
        .font(.title)
    }
}
```

See Also

Layout and presentation

- [Supporting additional widget sizes](#)
Offer widgets in additional contexts by adding support for various widget sizes.
- [Optimizing your widget for accented rendering mode and Liquid Glass](#)
Make your widget feel at home on Apple platforms and Liquid Glass by using accented rendering mode.
- [Adding StandBy and CarPlay support to your widget](#)
Ensure that your small system family widget works well in StandBy and CarPlay.

```
struct WidgetRenderingMode
    Constants that indicate the rendering mode for a widget.

struct WidgetAccentedRenderingMode
    Constants that indicate the rendering mode for an Image in when displayed in a widget in accented mode.

struct AccessoryWidgetBackground
    An adaptive background view that provides a standard appearance based on the the widget’s environment.

struct WidgetLocation
    Values that indicate different widget locations.
```

Platforms

- iOS
- iPadOS
- macOS
- tvOS
- visionOS
- watchOS

Tools

- Swift
- SwiftUI
- Swift Playground
- TestFlight
- Xcode
- Xcode Cloud
- SF Symbols

Topics & Technologies

- Accessibility
- Accessories
- App Extension
- App Store
- Audio & Video
- Augmented Reality
- Design
- Distribution
- Education
- Fonts
- Games
- Health & Fitness
- In-App Purchase
- Localization
- Maps & Location
- Machine Learning & AI
- Open Source
- Security
- Safari & Web

Resources

- Documentation
- Tutorials
- Downloads
- Forums
- Videos
- Support
- Support Articles
- Contact Us
- Bug Reporting
- System Status
- Account
- Apple Developer
- App Store Connect
- Certificates, IDs, & Profiles
- Feedback Assistant

Programs

- Apple Developer Program
- Apple Developer Enterprise Program
- App Store Small Business Program
- MFI Program
- News Partner Program
- Video Partner Program
- Security Bounty Program
- Security Research Device Program
- Events
- Meet with Apple
- Apple Developer Centers
- App Store Awards
- Apple Design Awards
- Apple Developer Academies
- WWDC