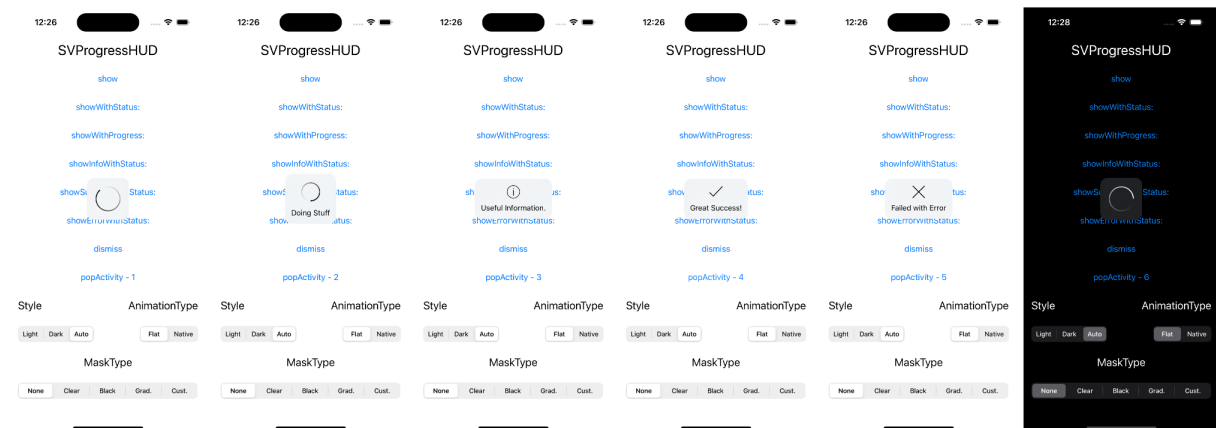


---

## SVProgressHUD



**SVProgressHUD** is a clean and easy-to-use HUD meant to display the progress of an ongoing task on iOS and tvOS.



## Installation

### Swift Package Manager

Swift Package Manager (SwiftPM) is a tool for managing the distribution of Swift code. It simplifies the process of managing Swift package dependencies.

To integrate **SVProgressHUD** into your project using SwiftPM:

1. In Xcode, select **File > Add Package Dependency**.
2. Enter the following package repository URL: <https://github.com/SVProgressHUD/SVProgressHUD.git>
3. Choose the appropriate version (e.g. a specific version, branch, or commit).
4. Add **SVProgressHUD** to your target dependencies.

**SVProgressHUD** requires at least Swift tools version 5.3.

### From CocoaPods

CocoaPods is a dependency manager for Objective-C, which automates and simplifies the process of using 3rd-party libraries like **SVProgressHUD** in your projects. First, add the following line to your Podfile:

```
1 pod 'SVProgressHUD'
```

---

If you want to use the latest features of `SVProgressHUD` use normal external source dependencies.

```
1 pod 'SVProgressHUD', :git => 'https://github.com/SVProgressHUD/SVProgressHUD.git'
```

This pulls from the `master` branch directly.

Second, install `SVProgressHUD` into your project:

```
1 pod install
```

## Carthage

Carthage is a decentralized dependency manager that builds your dependencies and provides you with binary frameworks. To integrate `SVProgressHUD` into your Xcode project using Carthage, specify it in your `Cartfile`:

```
1 github "SVProgressHUD/SVProgressHUD"
```

Run `carthage bootstrap` to build the framework in your repository's Carthage directory. You can then include it in your target's `carthage copy-frameworks` build phase. For more information on this, please see Carthage's documentation.

## Manually

- Drag the `SVProgressHUD/SVProgressHUD` folder into your project.
- Take care that `SVProgressHUD.bundle` is added to `Targets->Build Phases->Copy Bundle Resources`.
- Add the **QuartzCore** framework to your project.

## Swift

Even though `SVProgressHUD` is written in Objective-C, it can be used in Swift with no hassle.

If you use CocoaPods add the following line to your Podfile:

```
1 use_frameworks!
```

If you added `SVProgressHUD` manually, just add a bridging header file to your project with the `SVProgressHUD` header included.

---

## Usage

(see sample Xcode project in [/Demo](#))

`SVProgressHUD` is created as a singleton (i.e. it doesn't need to be explicitly allocated and instantiated; you directly call `[SVProgressHUD method]` / `SVProgressHUD.method()`).

**Use `SVProgressHUD` wisely! Only use it if you absolutely need to perform a task before taking the user forward. Bad use case examples: pull to refresh, infinite scrolling, sending message.**

Using `SVProgressHUD` in your app will usually look as simple as this.

### Objective-C:

```
1 [SVProgressHUD show];
2 dispatch_async(dispatch_get_global_queue(
3     DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
4     // time-consuming task
5     dispatch_async(dispatch_get_main_queue(), ^{
6         [SVProgressHUD dismiss];
7     });
8 });
```

### Swift:

```
1 SVProgressHUD.show()
2 DispatchQueue.global(qos: .default).async {
3     // time-consuming task
4     DispatchQueue.main.async {
5         SVProgressHUD.dismiss()
6     }
7 }
```

## Showing the HUD

You can show the status of indeterminate tasks using one of the following:

```
1 + (void)show;
2 + (void)showWithStatus:(NSString*)string;
```

If you'd like the HUD to reflect the progress of a task, use one of these:

```
1 + (void)showProgress:(CGFloat)progress;
2 + (void)showProgress:(CGFloat)progress status:(NSString*)status;
```

---

## Dismissing the HUD

The HUD can be dismissed using:

```
1 + (void)dismiss;  
2 + (void)dismissWithDelay:(NSTimeInterval)delay;
```

If you'd like to stack HUDs, you can balance out every show call using:

```
1 + (void)popActivity;
```

The HUD will get dismissed once the `popActivity` calls will match the number of show calls.

Or show an image with status before getting dismissed a little bit later. The display time depends on `minimumDismissTimeInterval` and the length of the given string.

```
1 + (void)showInfoWithStatus:(NSString*)string;  
2 + (void)showSuccessWithStatus:(NSString*)string;  
3 + (void)showErrorWithStatus:(NSString*)string;  
4 + (void)showImage:(UIImage*)image status:(NSString*)string;
```

## Customization

`SVProgressHUD` is designed with flexibility in mind, providing a myriad of customization options to fit the look and feel of your application seamlessly.

- Appearance: Make use of the `UI_APPEARANCE_SELECTOR` to adjust styles, colors, fonts, size, and images app-wide.
- Behavior: Control visibility durations, display delays, and animation speeds.
- Feedback: Enhance the user experience with options for haptic feedback and motion effects.

For a comprehensive list of properties and detailed explanations, refer to the `SVProgressHUD.h` file in the API documentation.

## Hint

As standard `SVProgressHUD` offers three preconfigured styles:

- `SVProgressHUDStyleAutomatic`: Automatically switch between the light and dark style
- `SVProgressHUDStyleLight`: White background with black spinner and text
- `SVProgressHUDStyleDark`: Black background with white spinner and text

If you want to use custom colors use `setForegroundColor:` and/or `setBackgroundColor:.` These implicitly set the HUD's style to `SVProgressHUDStyleCustom`.

---

## Haptic Feedback

Available on iPhone 7 and newer, `SVProgressHUD` can automatically trigger haptic feedback depending on which HUD is being displayed. The feedback maps as follows:

- `showSuccessWithStatus:` `<-> UINotificationFeedbackTypeSuccess`
- `showInfoWithStatus:` `<-> UINotificationFeedbackTypeWarning`
- `showErrorWithStatus:` `<-> UINotificationFeedbackTypeError`

To enable this functionality, use `setHapticsEnabled:`.

## Notifications

`SVProgressHUD` posts four notifications via `NSNotificationCenter` in response to being shown/dismissed:

- `SVProgressHUDWillAppearNotification` when the show animation starts
- `SVProgressHUDDidAppearNotification` when the show animation completes
- `SVProgressHUDWillDisappearNotification` when the dismiss animation starts
- `SVProgressHUDDidDisappearNotification` when the dismiss animation completes

Each notification passes a `userInfo` dictionary holding the HUD's status string (if any), retrievable via `SVProgressHUDStatusUserInfoKey`.

`SVProgressHUD` also posts `SVProgressHUDDidReceiveTouchEventNotification` when users touch on the overall screen or `SVProgressHUDDidTouchDownInsideNotification` when a user touches on the HUD directly. For these notifications `userInfo` is not passed but the object parameter contains the `UIEvent` that related to the touch.

## App Extensions

When using `SVProgressHUD` in an App Extension, `#define SV_APP_EXTENSIONS` to avoid using unavailable APIs. This will be done automatically when using the `AppExtension` CocoaPods subspec. Additionally, call `setViewForExtension:` from your extensions view controller with `self.view`.

## Contributing to this project

If you have feature requests or bug reports, feel free to help out by sending pull requests or by creating new issues. Please take a moment to review the guidelines written by Nicolas Gallagher:

- 
- Bug reports
  - Feature requests
  - Pull requests

## License

[SVProgressHUD](#) is distributed under the terms and conditions of the MIT license. The success, error and info icons used on iOS 12 are made by Freepik from Flaticon and are licensed under Creative Commons BY 3.0.

## Privacy

[SVProgressHUD](#) does not collect any data. A privacy manifest file is provided.

## Credits

[SVProgressHUD](#) is brought to you by Sam Vermette, Tobias Totzek and contributors to the project. If you're using [SVProgressHUD](#) in your project, attribution would be very appreciated.