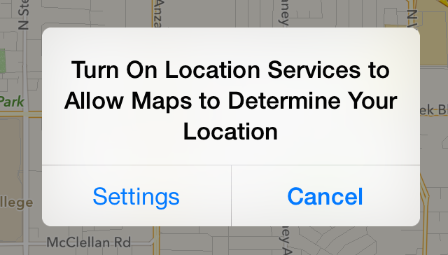
# Icon and Image Design

Temporary Views

Alert

An *alert* gives people important information that affects their use of an app or the device.



API NOTE

To use an alert in your code, you create a [UIAlertController](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIAlertController_class/index.html#//apple_ref/occ/cl/UIAlertController) and specify the[UIAlertControllerStyleAlert](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIAlertController_class/index.html#//apple_ref/c/econst/UIAlertControllerStyleAlert).

An alert:

* Displays a required title and an optional message
* Contains one or more buttons

The infrequency with which alerts appear helps users take them seriously. It’s best to minimize the number of alerts your app displays, and make sure each one offers critical information and useful choices.

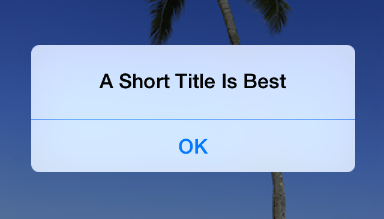
**Avoid creating unnecessary alerts.** In general, alerts are unnecessary in the following scenarios:

| If an alert does this... | Do this instead of using an alert... |
| --- | --- |
| Provides information related to the standard functioning of an app | Design an eye-catching way to display the information, one that harmonizes with the app’s style. |
| Updates users on tasks that are progressing normally | Use a progress view or activity indicator (described in [Progress View](https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/Controls.html#//apple_ref/doc/uid/TP40006556-CH15-SW7) and [Activity Indicator](https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/Controls.html#//apple_ref/doc/uid/TP40006556-CH15-SW2)) or integrate status information into the app UI. |
| Asks for confirmation of user-initiated tasks | Use an action sheet (described in [Action Sheet](https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/Alerts.html#//apple_ref/doc/uid/TP40006556-CH14-SW36)). |
| Informs users of problems they can do nothing about | If the problem isn’t critical, integrate the information into the app’s UI; otherwise, use an alert. |

As you read the guidelines for designing alert text, it’s useful to know the following definitions:

* *Title-style capitalization* means that every word is capitalized, except articles, coordinating conjunctions, and prepositions of four or fewer letters when they aren’t the first word.
* *Sentence-style capitalization* means that the first word is capitalized, and the rest of the words are lowercase unless they are proper nouns or proper adjectives.

**Succinctly describe the situation and explain what people can do about it.** Ideally, the text you write gives people enough context to understand why the alert has appeared and to decide which button to tap.



**Keep the title short enough to display on a single line, if possible.** A long alert title is difficult for people to read quickly, and it might get truncated or force the alert message to scroll.

**Avoid single-word titles.** Single-word titles, such as Error or Warning, rarely provide any useful information.

**When possible, use a sentence fragment.** A short, informative statement tends to be easier to understand than a complete sentence.

**As much as possible, write a title that makes it unnecessary to add a message.** For example, you might be able to avoid adding a message if you use a question—or, less frequently, two sentences—for the alert title.

**Avoid sounding accusatory or judgmental when you need to deliver negative news.**People understand that many alerts tell them about problems or warn them about dangerous situations. As long as you use a friendly tone, it’s better to be negative and direct than it is to be positive but oblique.

**As much as possible, avoid “you,” “your,” “me,” and “my.”** Sometimes, text that identifies people directly can be ambiguous and can even be interpreted as insulting or patronizing.

**Use capitalization and punctuation appropriately.** Specifically:

| When the alert title... | Use... |
| --- | --- |
| Is a sentence fragment or a single sentence that is not a question | Title-style capitalization and no ending punctuation |
| Is a single sentence that is a question | Sentence-style capitalization and an ending question mark |
| Consists of two or more sentences | Sentence-style capitalization and appropriate ending punctuation for each sentence |

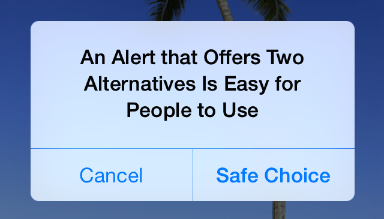
**If you must provide an optional alert message, write a short, complete sentence.** If possible, keep the message short enough to be displayed on one or two lines. If the message is too long, it will scroll, giving users a poor experience. Use sentence-style capitalization and appropriate ending punctuation in the message.



**Avoid lengthening alert text with descriptions of which button to tap.** Ideally, the combination of unambiguous alert text and logical button labels gives people enough information to understand the situation and their choices. If you must provide detailed guidance, follow these guidelines:

* Be sure to use the word “tap” (not “touch” or “click” or “choose”) to describe the selection action.
* Don’t enclose a button title in quotation marks, but do preserve its capitalization.

**Be sure to test the appearance of an alert in both orientations.** Because in landscape the height of an alert is constrained, the alert’s appearance may differ from its appearance in portrait. It’s recommended that you optimize the length of the alert text so that it can be read without scrolling no matter what the orientation.



**Generally, use a two-button alert.** A two-button alert is often the most useful, because it’s easiest for people to choose between two alternatives. A single button alert is less likely to be helpful because it informs people without giving them any control over the situation. An alert that contains three or more buttons is significantly more complex than a two-button alert and should be avoided as much as possible. If you add too many buttons to an alert, it can cause the alert to scroll, which is a bad user experience.

NOTE

If you find that you need to offer people more than two choices, consider using an action sheet instead (to learn how to use an action sheet, see [Action Sheet](https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/Alerts.html#//apple_ref/doc/uid/TP40006556-CH14-SW36)).

**Place buttons appropriately.** Ideally, the button that's most natural to tap should meet two criteria: It should perform the action that users are most likely to want and it should be the least likely to cause problems if a user taps it inadvertently. Specifically:

* When the most likely button performs a nondestructive action, it should be on the right in a two-button alert. The button that cancels this action should be on the left.
* When the most likely button performs a destructive action, it should be on the left in a two-button alert. The button that cancels this action should be on the right.

NOTE

Pressing the Home button while an alert is visible should quit the app, as expected. Doing so should also be identical to tapping the Cancel button—that is, the alert is dismissed and the action isn’t performed.

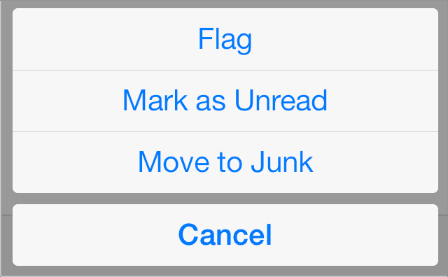
**Give alert buttons short, logical titles.** The best button titles consist of one or two words that describe the result of tapping the button. Follow these guidelines as you create titles for alert buttons:

* As with all button titles, use title-style capitalization and no ending punctuation.
* As much as possible, use verbs and verb phrases that relate directly to the alert text—for example, “Cancel,” “View All,” “Reply,” or “Ignore.”
* Use “OK” for a simple acceptance option if there is no better alternative. Avoid using “Yes” or “No.”
* Avoid “you,” “your,” “me,” and “my” as much as possible. Button titles that use these words are often ambiguous and can appear patronizing.

Action Sheet

An **action sheet** displays a set of choices related to a task the user initiates.

In a horizontally compact environment, an action sheet emerges from the bottom of the screen



In a horizontally regular environment, an action sheet is always displayed in a popover



API NOTE

To use an action sheet in your code, you create a [UIAlertController](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIAlertController_class/index.html#//apple_ref/occ/cl/UIAlertController) and specify the [UIAlertControllerStyleActionSheet](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIAlertController_class/index.html#//apple_ref/c/econst/UIAlertControllerStyleActionSheet).

An action sheet:

* Appears as the result of a user action
* Displays two or more buttons

Use an action sheet to:

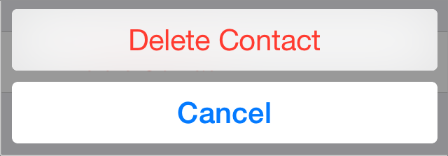
* **Provide alternative ways to complete a task.** An action sheet lets you to provide a range of choices that make sense in the context of the current task, without giving these choices a permanent place in the UI.
* **Get confirmation before completing a potentially dangerous task.** An action sheet prompts users to think about the potentially dangerous effects of the step they’re about to take and gives them some alternatives.

**In a horizontally compact environment, include a Cancel button so that users can easily and safely abandon the task.** Place the Cancel button at the bottom of the action sheet to encourage users to read through all the alternatives before making a choice.

**In a horizontally regular environment, base the way the action sheet is displayed on the way the user initiates the task.** Specifically:

| If the task is initiated from... | Display the action sheet... | Include a Cancel button? |
| --- | --- | --- |
| Outside of a popover | Without animation—that is, the action sheet and the popover appear simultaneously | No, because users can tap outside the popover to dismiss the action sheet |
| Inside a popover | With animation—that is, the action sheet slides up on top of the popover’s content | Yes, because users need to be able to dismiss the action sheet without closing the popover |

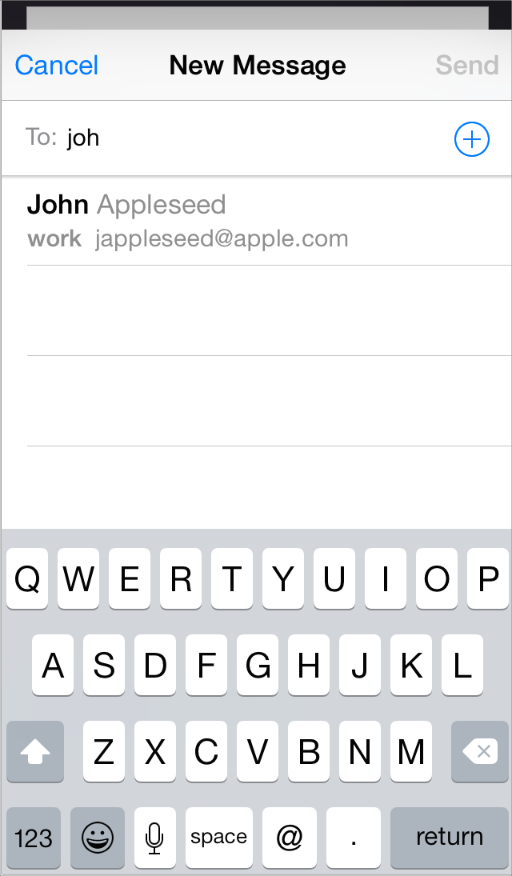
**In all environments, use red for the button that performs a potentially destructive action.**Display a red button at the top of the action sheet, because the closer to the top of the action sheet a button is, the more eye-catching it is.



**Avoid making users scroll an action sheet.** If you include too many buttons in an action sheet, users must scroll to see all their choices. This is a disconcerting experience for users, because they must spend extra time to distinguish the choices. Also, it can be very difficult for users to scroll without inadvertently tapping a button.

Modal View

A *modal view*—that is, a view presented modally—provides self-contained functionality in the context of the current task or workflow.



API NOTE

To use a modal view in your code, you create a [UIPresentationController](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIPresentationController_class/index.html#//apple_ref/occ/cl/UIPresentationController)and specify an appropriate style (for a complete list of styles, see [Modal Presentation Styles](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIViewController_Class/index.html#//apple_ref/c/tdef/UIModalPresentationStyle)).

A modal view:

* Can occupy the entire screen, the entire area of a parent view (such as a popover), or a portion of the screen
* Contains the text and controls that are necessary to complete the task
* Usually displays a button that completes the task and dismisses the view and a Cancel button that abandons the task and dismisses the view

Use a modal view when you need to offer the ability to accomplish a self-contained task related to your app’s primary function. A modal view is especially appropriate for a multistep subtask that requires UI elements that don’t belong in the main app UI all the time.

**Choose a modal view style that suits the current task, the current environment, and the visual style of your app.** You can use any of these styles, defined here:

| Modal view style | Appearance | Recommended for |
| --- | --- | --- |
| Full screen | Covers the entire screen. | Presenting a potentially complex task that people can complete within the context of the modal view. |
| Page sheet | In a horizontally regular environment, a style that partially covers the underlying content. All uncovered areas are dimmed to prevent the user from interacting with them.  (In a horizontally compact environment, this style behaves the same as the full screen style.) | Presenting a potentially complex task that people can complete within the context of the modal view. |
| Form sheet | In a horizontally regular environment, a style that displays the content centered in the screen. All uncovered areas are dimmed to prevent the user from interacting with them. In some cases, the position of the modal view is adjusted when a keyboard is present.  (In a horizontally compact environment, this style behaves the same as the full screen style.) | Gathering structured information from the user. |
| Current context | Uses the same size as its parent view. | Displaying modal content within a split view pane, popover, or other non–full–screen view. |

**Don’t display a modal view on top of a popover.** With the possible exception of an alert, nothing should display on top of a popover. In rare cases when you might need to display a modal view as a result of an action the user takes in a popover, close the popover before you open the modal view.

**Coordinate the overall look of a modal view with the appearance of your app.** For example, a modal view often includes a navigation bar that contains a title and buttons that cancel or complete the modal view’s task. When this is the case, the navigation bar should use the same appearance as the navigation bar in the app.

**Display a title that identifies the task, if appropriate.** You might also display text in other areas of the view that more fully describes the task or provides some guidance.

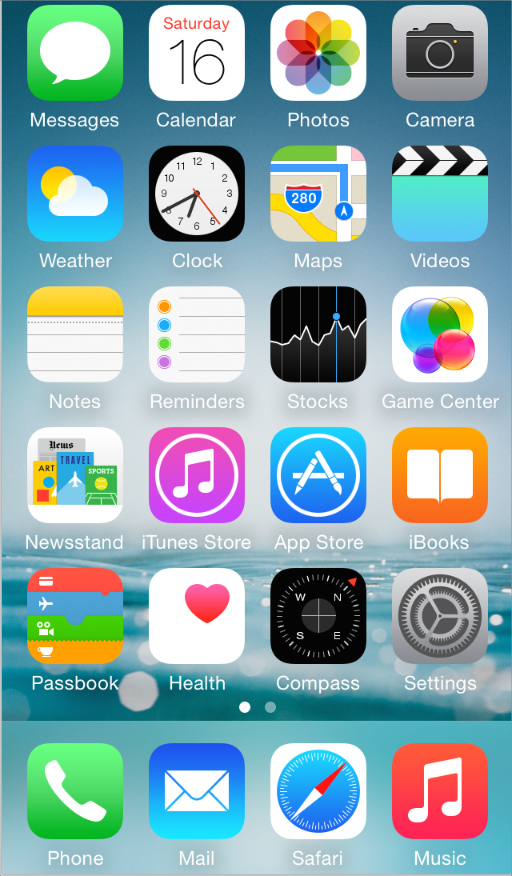
**Choose an appropriate transition style for revealing the modal view.** Use a style that coordinates with your app and enhances the user’s awareness of the temporary context shift that the modal view represents. To do this, you can specify one of the following transition styles:

* **Vertical.** In the vertical style, the modal view slides up from the bottom edge of the screen and slides back down when dismissed (this is the default transition style).
* **Flip.** In the flip style, the current view flips horizontally from right to left to reveal the modal view. Visually, the modal view looks as if it is the back of the current view. When the modal view is dismissed, it flips horizontally from left to right, revealing the previous view.

**If you vary the transition styles for modal views in an app, do so in a way that makes sense to users.** Users are quick to notice behavioral differences in an app and will assume that they mean something. It’s best to establish a logical, consistent pattern that users can easily detect and remember, and avoid changing transition styles without a good reason.

App Icon

Every app needs a beautiful, memorable app icon that attracts people in the App Store and stands out on their Home screen. iOS can use versions of the app icon in Game Center, search results, Settings, and to represent app-created documents.



**For the best results, enlist the help of a professional graphic designer.** An experienced graphic designer can help you develop an overall visual style for your app and apply that style to all the icons and images in it.

**Use universal imagery that people will easily recognize.** In general, avoid focusing on a secondary or obscure aspect of an element. For example, the Mail app icon uses an envelope, not a rural mailbox, a mail carrier’s bag, or a post office symbol.

**Embrace simplicity.** In particular, avoid cramming lots of different images into your icon. Find a single element that captures the essence of your app and express that element in a simple, unique shape. Add details cautiously. If an icon’s content or shape is overly complex, the details can become confusing and may appear muddy at smaller sizes.

TIP

To test the appearance of your app icon at small sizes, move it into a folder on the Home screen. Even better, move several app icons into a folder and see if your app icon looks good and remains distinctive.

**Create an abstract interpretation of your app’s main idea.** It rarely works well to use a photo or screenshot in an app icon because photographic details can be very hard to see at small sizes. Typically, it’s better to interpret reality in an artistic way, because doing so lets you emphasize the aspects of the subject that you want users to notice.

**If you want to portray real substances, do it accurately.** Icons that depict real objects should accurately replicate the characteristics of substances such as fabric, glass, paper, and metal, and convey the object’s weight and feel.

**Make sure the app icon looks good on a variety of backgrounds.** Don’t just test your icon on a light or dark background, because you can’t predict which wallpaper people will choose.

**Avoid transparency.** An app icon should be opaque. If the icon’s boundaries are smaller than the recommended sizes—or you use transparency to create “see-through” areas—the resulting icon can appear to float on a dark background, which tends to look especially unattractive on the beautiful wallpapers that users choose.

**Don’t use iOS interface elements in your artwork.** You don’t want users to confuse your icons or images with the iOS UI.

**Don’t use replicas of Apple hardware products in your artwork.** The symbols that represent Apple products are copyrighted and can’t be reproduced in your icons or images. In general, it’s a good idea to avoid replicas of any specific devices in your artwork, because these designs change frequently and icons that are based on them can quickly look dated.

**Don’t reuse iOS app icons in your interface.** It can be confusing to users to see the same icon used to mean slightly different things in multiple locations throughout the system.

With the exception of the App Store icon—which must be named iTunesArtwork—you can name an app icon anything you want. As long as you use the CFBundleIcons key to declare the names and you add the @2x suffix to the names of all high-resolution icons, iOS chooses an icon based on whether its size is appropriate for the intended usage. To learn more about icon naming, see [App Icons](https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/ExpectedAppBehaviors/ExpectedAppBehaviors.html#//apple_ref/doc/uid/TP40007072-CH3-SW1).

**Create different sizes of the app icon for different devices.** You want to make sure that your app icon looks great on all the devices you support. For device-specific measurements, see[Table 41-1](https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/IconMatrix.html#//apple_ref/doc/uid/TP40006556-CH27-SW2).

When iOS displays an app icon on the Home screen of a device, it automatically applies a mask that rounds the corners. Make sure your icon has 90° corners so it looks good after the mask is applied. For example:

A 120 x 120 pixel icon before the mask is applied



A 120 x 120 pixel icon after the mask is applied



**Create a large version of your app icon for display in the App Store.** Although it’s important that this version be instantly recognizable as your app icon, it can be subtly richer and more detailed. There are no visual effects added to this version of your app icon.

As shown in [Table 41-1](https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/IconMatrix.html#//apple_ref/doc/uid/TP40006556-CH27-SW2), the large version of your app icon should measure 1024 x 1024 pixels and be named iTunesArtwork@2x. (If necessary to support some @1x devices, create a version that measures 512 x 512 pixels and name it iTunesArtwork.)

NOTE

iOS might also use the large image in other ways. In an iPad app, for example, iOS uses the large image to generate the large document icon.

If you’re developing an app for ad-hoc distribution (that is, to be distributed in-house only, not through the App Store), you must also provide the large versions of your app icon. This icon identifies your app in iTunes.

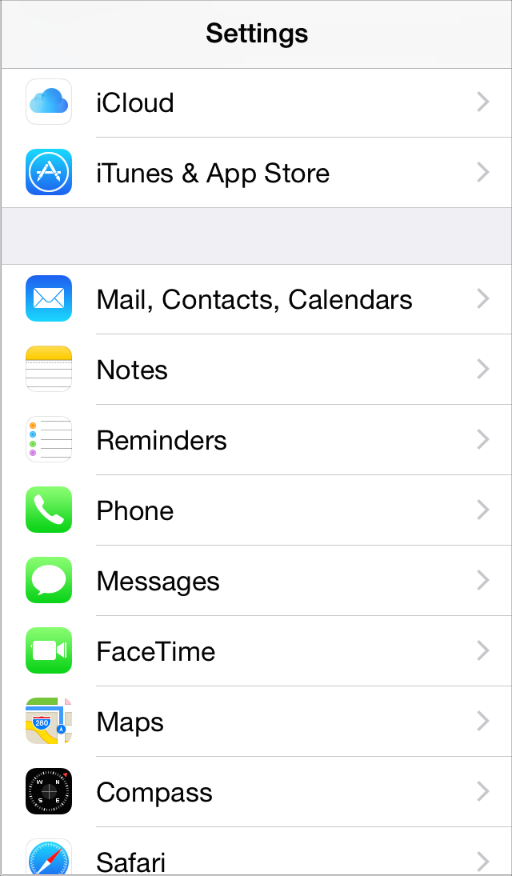
Document Icons

If your iOS app creates documents of a custom type, you want users to be able to recognize these documents at a glance. You don't need to design a custom icon for this purpose because iOS uses your app icon to create document icons for you.

Spotlight and Settings Icons

Every app should supply a small icon that iOS can display when the app name matches a term in a Spotlight search. Apps that supply settings should also supply a small icon to identify them in the built-in Settings app.

These icons should clearly identify your app so that people can recognize it in a list of search results or in Settings. For example, the icons of the built-in apps are easy to discern in Settings, even though the icons are small:



You can name these small icons anything you want as long as you use the CFBundleIconskey to declare the names and you add the @2x suffix to the names of all high-resolution icons. You can use custom names because iOS chooses an icon based on whether its size is appropriate for the intended usage. To learn more about icon naming, see [App Icons](https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/ExpectedAppBehaviors/ExpectedAppBehaviors.html#//apple_ref/doc/uid/TP40007072-CH3-SW1).

For all devices, supply separate icons for Spotlight search results and Settings. If you don’t provide these icons, iOS might shrink your app icon for display in these locations.

For Spotlight search results on iPhone, iPod touch, and iPad create an icon in the following two sizes:

* 80 x 80 pixels
* 40 x 40 pixels (standard resolution)

For Settings on iPhone, iPod touch, and iPad create an icon in the following two sizes:

* 58 x 58 pixels
* 29 x 29 pixels (standard resolution)

NOTE

If the background of your icon is white, don’t add a gray overlay in an effort to increase its visibility in Settings. iOS adds a 1-pixel border stroke so that all icons look good on the white background of Settings.

Launch Images

A launch file or image provides a simple placeholder image that iOS displays when your app starts up. The placeholder image gives users the impression that your app is fast and responsive because it appears instantly and is quickly replaced by the first screen of your app. Every app must supply a launch file or at least one static image.

In iOS 8 and later, you can create a XIB or storyboard file instead of a static launch image. When you create a launch file in Interface Builder, you use size classes to define different layouts for different display environments and you use Auto Layout to make minor adjustments. Using size classes and Auto Layout means that you can create a single launch file that looks good on all devices and display environments. (For an overview of display environments and size classes, see [Build In Adaptivity](https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/LayoutandAppearance.html#//apple_ref/doc/uid/TP40006556-CH54-SW2); to learn how to use size classes in Interface Builder, see[*Size Classes Design Help*](https://developer.apple.com/library/ios/recipes/xcode_help-IB_adaptive_sizes/_index.html#//apple_ref/doc/uid/TP40014436).)

If you also need to support earlier versions of iOS, you can continue to supply static launch images in addition to a launch file.

IMPORTANT

You use a launch XIB or storyboard file to indicate that your app runs on iPhone 6 Plus or iPhone 6.

The following design guidelines apply to both launch files and static launch images.

**Design a plain launch image that improves the user experience.** In particular, the launch image isn’t an opportunity to provide:

* An “app entry experience,” such as a splash screen
* An About window
* Branding elements, unless they are a static part of your app’s first screen

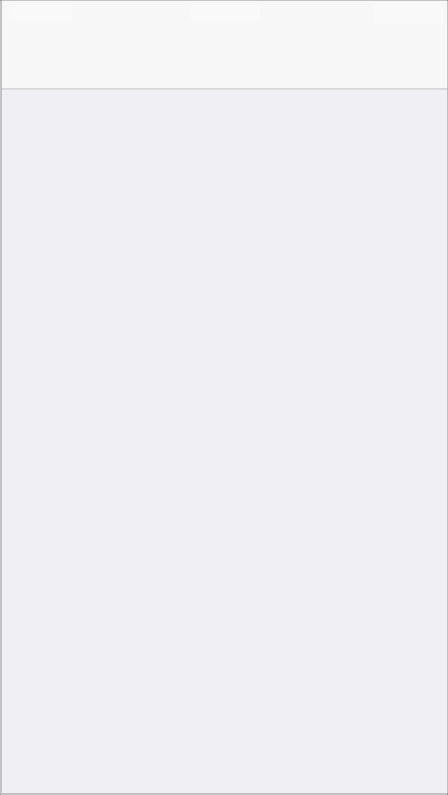
Because users are likely to switch among apps frequently, you should make every effort to cut launch time to a minimum, and you should design a launch image that downplays the experience rather than drawing attention to it.

**Design a launch image that is identical to the first screen of the app,** except for:

* **Text.** The launch image is static, so any text you display in it won’t be localized.
* **UI elements that might change.** If you include elements that might look different when the app finishes launching, users can experience an unpleasant flash between the launch image and the first app screen.

If you think that following these guidelines will result in a plain, boring launch image, you’re right. Remember, the launch image doesn’t provide you with an opportunity for artistic expression. It’s solely intended to enhance the user’s perception of your app as quick to launch and immediately ready for use. For example, Settings and Weather each supply a launch image that is little more than a background image.

The Settings launch image



The Weather launch image



If you need to use static launch images, you create images in different sizes for different devices. Static launch images for all devices must include the status bar region. For specific measurements, see [Table 41-1](https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/IconMatrix.html#//apple_ref/doc/uid/TP40006556-CH27-SW2).

Although it’s best to use a launch file for iPhone 6 and iPhone 6 Plus, you can instead supply static launch images if necessary. If you need to create static launch images for these devices, use the following sizes:

For iPhone 6:

* 750 x 1334 (@2x) for portrait
* 1334 x 750 (@2x) for landscape

For iPhone 6 Plus:

* 1242 x 2208 (@3x) for portrait
* 2208 x 1242 (@3x) for landscape

If you’re using static launch images, you can give each image a name that specifies how it should be used. The format of the launch image filename includes modifiers you use to specify the device, resolution, and orientation of the image. To learn how to name launch images appropriately, see [App Launch (Default) Images](https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/ExpectedAppBehaviors/ExpectedAppBehaviors.html#//apple_ref/doc/uid/TP40007072-CH3-SW3).

Bar Button Icons

iOS defines lots of standard bar-button icons, such as Refresh, Action, Add, and Favorites. As much as possible, you should use these buttons and icons to represent standard tasks in your app. (To learn more about the standard buttons and icons you can use, see [Toolbar and Navigation Bar Buttons](https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/Bars.html#//apple_ref/doc/uid/TP40006556-CH12-SW33) and [Tab Bar Icons](https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/Bars.html#//apple_ref/doc/uid/TP40006556-CH12-SW34).)

If your app includes tasks or modes that can’t be represented by a standard icon—or if the standard icons don’t coordinate with your app’s style—you can design your own bar button icons. At a high level, you should aim for an icon design that is:

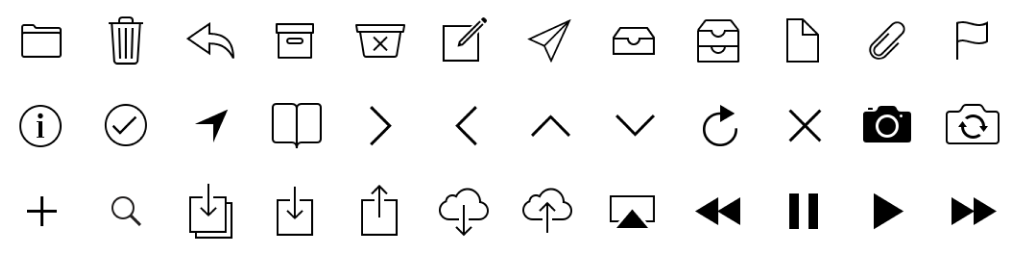
* **Simple and streamlined.** Too many details can make an icon appear sloppy or indecipherable.
* **Not easily mistaken for one of the system-provided icons.** Users should be able to distinguish your custom icon from the standard icons at a glance.
* **Readily understood and widely acceptable.** Strive to create a symbol that most users will interpret correctly and that no users will find offensive.

IMPORTANT

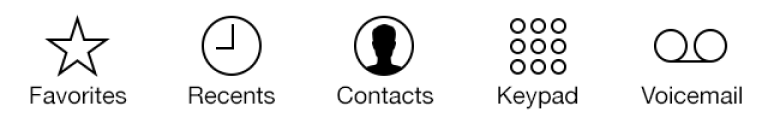
Be sure to avoid using images that replicate Apple products in your designs. These symbols are copyrighted and product designs can change frequently.

Whether you use only custom icons or a mix of custom and standard, all icons in your app should look like they belong to the same family in terms of perceived size, level of detail, and visual weight.

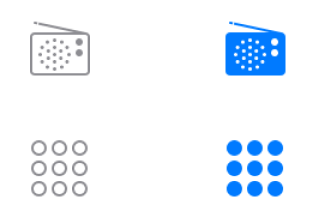
For example, take a look at the family of iOS bar icons and notice how the similarities in size, detail, and weight produce a sense of harmonious unity:



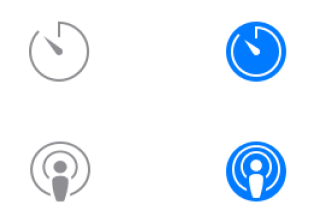
To create a coherent family of icons, consistency is key: As much as possible, each icon should use the same perspective and the same stroke weight. To ensure that all icons have a consistent perceived size, you may have to create some icons at different actual sizes. For example, the set of system provided icons shown here all have the same perceived size, even though the Favorites and Voicemail icons are actually a bit larger than the other three icons.



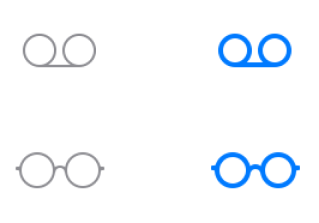
If you’re designing a custom tab bar icon, you should provide two versions—one for the unselected appearance and one for the selected appearance. The selected appearance is often a filled-in version of the unselected appearance, but some designs call for variations on this approach.



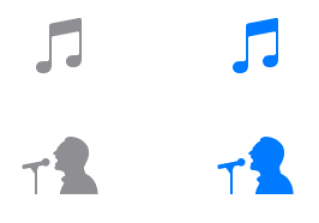
To create a filled-in version of an icon that has interior details (such as the Radio icon) invert the details so they retain their prominence in the selected version. The Keypad icon also has interior details, but the selected version would be confusing and hard to recognize if its background was filled in and the circles became white outlines.



Sometimes, a design needs a slight alteration to look good when it’s selected. For example, because the Timer and Podcasts icons include open areas, the selected versions condense the strokes a bit to fit into a circular enclosure.



If an icon becomes less recognizable when it’s filled in, a good alternative is to use a heavier stroke to draw the selected version. For example, the selected versions of the Voicemail and Reading List icons are drawn with a 4-pixel stroke, instead of the 2-pixel stroke that was used to draw the unselected versions.



Sometimes, an icon’s shape has details that don’t look good in a stroked outline. When this is the case—as it is for the Music and Artists icons—you can use the filled-in appearance for both versions of the icon. It’s easy for users to distinguish the selected and unselected appearances of such icons because the selected appearance is darker and gets the tint.

A custom icon that you create for a toolbar, navigation bar, or tab bar is also known as a*template* image, because iOS uses it as a mask to produce the icon you see when your app runs. If you create a full-color template image, iOS ignores the color.

To design a custom bar icon, follow these guidelines:

* Use pure white with appropriate alpha transparency.
* Don’t include a drop shadow.
* Use antialiasing.

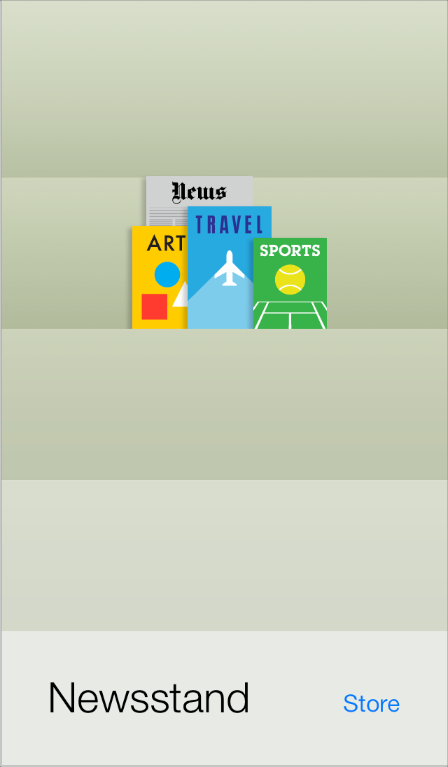
If you want to create a bar icon that looks like it's related to the iOS icon family, use a very thin stroke to draw it. Specifically, a 2-pixel stroke (high resolution) works well for detailed icons and a 3-pixel stroke works well for less detailed icons.

Regardless of the icon’s visual style, use the sizes listed in [Table 41-1](https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/IconMatrix.html#//apple_ref/doc/uid/TP40006556-CH27-SW2) to create custom toolbar, navigation bar, and tab bar icons.

Don’t include text in a custom tab bar icon. Instead, use the tab bar item APIs to set the title for each tab (for example, [initWithTitle:image:tag:](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UITabBarItem_Class/index.html#//apple_ref/occ/instm/UITabBarItem/initWithTitle:image:tag:)). If you need to adjust the automatic layout of the title, you can use the title adjustment APIs, such as[setTitlePositionAdjustment:](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UITabBarItem_Class/index.html#//apple_ref/occ/instm/UITabBarItem/setTitlePositionAdjustment:).

Newsstand Icons

If your app uses NewsstandKit to publish subscription-based periodical content, you need to provide icons for display in the App Store and on people’s devices.



All Newsstand apps need to supply a Newsstand cover icon that represents the default cover art in the App Store. The long edge of this icon should measure at least 1024 pixels (512 pixels for standard-resolution devices). Note that this icon is separate from the app icon that all iOS apps must provide.

IMPORTANT

The aspect ratio of all Newsstand icons should be between 1:2 and 2:1.

All Newsstand icons must be flat and have 90° corners.

Don’t add perspective to any of your Newsstand icons.

A default Newsstand cover icon should be a generalized facsimile of the cover of a typical issue, which focuses on the parts of the cover that are fairly consistent from issue to issue. For example:

* Avoid adding to the default cover icon elements that users would never see on an actual cover, such as a message to “tap here for the latest issue”.
* Avoid using artwork or headlines that are seasonal or topical, such as images related to holidays or headlines that refer to current events.

In particular, don’t reuse the cover of a previous issue for your default Newsstand cover icon, because users might confuse your app with a specific issue. For example, default magazine and newspaper icons could look something like this:



In addition to the default Newsstand cover icon, you also need to supply a separate icon that accurately represents each new issue so it can appear on the Newsstand shelf and in the multitasking UI on an iOS device. Unlike the default cover icon, each per-issue icon should display details about the contents of a specific issue.

It’s recommended that you create a single large icon for each issue, and allow iOS to scale it for display in both places.

Specifically, you should create a per-issue icon whose long edge measures at least 1024 pixels (512 pixels for standard-resolution devices). To display the current issue’s icon on the Newsstand shelf and in the multitasking UI, iOS scales your large icon to the following sizes:

| Device | Scaled size (Newsstand shelf) | Scaled long-edge size (multitasking UI) |
| --- | --- | --- |
| iPhone 6 Plus | 270 x 240 pixels | 180 pixels |
| All other iPhone models | 180 x 160 pixels (90 x 80 pixels for @1x) | 120 pixels (60 pixels for @1x) |
| iPad | 252 pixels for long edge (126 pixels for @1x) | 152 pixels (76 pixels for @1x) |
| **Table 45-1**Maximum scaled sizes for per-issue icons | | |

For additional information about setting up a Newsstand app, see [*iTunes Connect Developer Guide*](https://developer.apple.com/library/ios/documentation/LanguagesUtilities/Conceptual/iTunesConnect_Guide/Chapters/About.html#//apple_ref/doc/uid/TP40011225).

IMPORTANT

Beginning in iOS 7, the system doesn’t add any visual enhancements—such as the appearance of a stapled edge or multiple pages—to a Newsstand icon.

If your app needs to support earlier versions of iOS, you can add the binding type and binding edge keys to the Info.plist file to define how Newsstand icons should appear on devices running versions of iOS prior to iOS 7. For more information about these keys and their values, see [Contents of the UINewsstandIcon Dictionary](https://developer.apple.com/library/ios/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html#//apple_ref/doc/uid/TP40009249-SW15).

## Web Clip Icons

If you have a web app or a website, you can provide a custom icon that users can display on their Home screens using the web clip feature. Users tap the icon to reach your web content in one easy step. You can create an icon that represents your website as a whole or an icon that represents a single webpage.

iOS also displays web clip icons in Safari Favorites, which is the grid of icons that appears when users tap the URL field or open a new tab in Safari.

If your web content is distinguished by a familiar image or recognizable color scheme, it makes sense to incorporate it in your icon. However, to ensure that your icon looks great on the device, you should also follow the guidelines in this section. (To learn how to add code to your web content to provide a custom icon, see [Specifying a Webpage Icon for Web Clip](https://developer.apple.com/library/ios/documentation/AppleApplications/Reference/SafariWebContent/ConfiguringWebApplications/ConfiguringWebApplications.html#//apple_ref/doc/uid/TP40002051-CH3-SW4).)

For icon measurements, see [Table 41-1](https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/IconMatrix.html#//apple_ref/doc/uid/TP40006556-CH27-SW2).

NOTE

You can prevent the addition of any effects by naming your icon apple-touch-icon-precomposed.png.

## Creating Resizable Images

You can create a resizable image to customize the background of several standard UI elements, such as popovers, buttons, navigation bars, tab bars, and toolbars (including the items on these bars). Providing resizable images for these elements can result in better app performance.

For many UI elements, you can also specify end caps in addition to a background appearance. An end cap defines an area of the image that should not be resized. For example, you might create a resizable image that includes four end caps that define the four corners of a button. When the image is resized to fill the button’s background area, the portions defined by the end caps are drawn unchanged.

Depending on the dimensions of the resizable image you supply, iOS either stretches or tiles it as appropriate to fill a UI element’s background area. To stretch an image means to scale up the image, without regard for its original aspect ratio. Stretching is performant, but it isn’t usually desirable for a multipixel image that can distort. To tile an image is to repeat the original image as many times as necessary to fill the target area. Tiling is less performant than stretching, but it's the only way to achieve a textured or patterned effect.

As a general rule, you should supply the smallest image (excluding end caps) that will result in the look you want. For example:

* If you want a solid color with no gradient, create a 1 x 1 point image.
* If you want a vertical gradient, create an image that has a width of 1 point and a height that matches the height of the UI element’s background.
* If you want to provide a repeating textured appearance, you need to create an image with dimensions that match the dimensions of the repeating portion of the texture.
* If you want to provide a nonrepeating textured appearance, you need to create a static image with dimensions that match the dimensions of the UI element’s background area.