Designing for Apple Watch

IMPORTANT

This is a preliminary document for an API or technology in development. Apple is supplying this information to help you plan for the adoption of the technologies and programming interfaces described herein for use on Apple-branded products. This information is subject to change, and software implemented according to this document should be tested with final operating system software and final documentation. Newer versions of this document may be provided with future betas of the API or technology.

Apple Watch embodies the following themes:

Personal. Because Apple Watch is designed to be worn, its UI is attuned to the wearer's presence. A raise of the wrist shows the time and new alerts. Digital Touch—particularly with its Heartbeat and Sketch features—enables new types of communication that are incredibly personal. An accelerometer and heart rate sensor provide personalized information about the wearer's activity from day to day. No other Apple device has ever been so connected to the wearer. It's important to be mindful of this connection as you design apps for Apple Watch.

Holistic. Apple Watch was designed to blur the boundaries between physical object and software. The Digital Crown is a finely tuned hardware control that enables nuanced software navigation. The Taptic Engine produces subtle, physical feedback associated with alerts and onscreen interaction. And Force Touch—a physical gesture interpreted by hardware—reveals a new dimension of contextual software controls. Even the physical border of the Retina display has been considered, resulting in edge-to-edge UI design that effectively renders that border invisible. Thoughtful app design should contribute to this experience of hardware and software feeling indistinguishable.

Lightweight. Apps on Apple Watch are designed for quick, lightweight interactions that make the most of the display size and its position on the wrist. Information is accessible and dismissible quickly and easily, for both privacy and usability. The notification Short Look, for example, is designed to provide a minimal alert, only revealing more information if the wearer remains engaged. And Glances provide information from apps in an easy-to-access, swipe-able interface. Apps designed for Apple Watch should respect the context in which the wearer experiences them: briefly, frequently, and on a small display.

A Watch app complements your iOS app; it does not replace it. If you measure interactions with your iOS app in minutes, you can expect interactions with your Watch app to be measured in seconds. So interactions need to be brief and interfaces need to be simple.

App Anatomy

Apple Watch must be paired with the user's iPhone for your app to run.

Interface Styles

Watch apps support two navigation methods:

Hierarchical. This style matches the navigation style in iOS and is best suited for apps with hierarchical information. In a hierarchical app, users navigate by making one choice per screen until they reach their destination. To navigate to another destination, users must retrace some or all of their steps and make different choices.

A hierarchical model is typically better than a flat, paginated navigation model for more complex app interactions.

Page-based. A paginated interface lets the user navigate between pages of content by swiping horizontally. A page-based interface is best suited for apps with simple data models where the data on each page is not directly tied to the data on the other pages.

A dot indicator at the bottom of each page shows the user's place in the set. Keep the total number of pages as small as possible to simplify navigation.

You cannot combine hierarchical and page-based interface styles. At design time, you must choose the style that best suits your app's content and design for that style.

Apps using either interface style can present content modally. Modal interfaces give the user a way to complete a task or get information without distractions, but in a way that temporarily prevents them from interacting with the rest of the app. For more information, see Modal Sheets.

User Interactions

Action-based events. The single tap gesture is the primary way that users interact with your app. Table rows, buttons, switches, and other controls are all operated by tapping on them. These taps are then reported to the code in your WatchKit extension.

Gestures. You cannot add gesture recognizers to your app. Instead, the system handles all gestures on your behalf, using them to implement standard behaviors:

- Vertical swipes scroll the current screen.
- Horizontal swipes display the previous or next page in a page-based interface.
- Left edge swipes navigate back to the parent interface controller.
- Taps indicate selection or interaction. Taps are handled by the system and reported to your WatchKit extension's action methods.

Apple Watch does not support multi-finger gestures such as pinches.

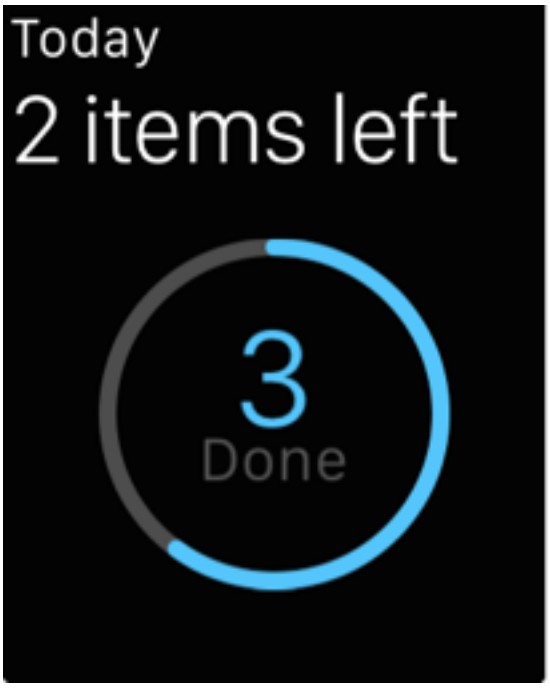
Force Touch. A small screen can only accommodate so many controls. That's why Apple Watch introduces an entirely new interaction model: Force Touch. As well as sensing touch, the Retina display also senses force. Force Touch interactions display the context menu (if any)

associated with the current screen. Apps use this menu to display actions relevant to the current content. For more information, see Menus.

The Digital Crown. Designed for finely tuned, accelerated scrolling—without obstructing the Apple Watch display—the Digital Crown makes it easy to scroll through longer pages.

Glances

On Apple Watch, a Glance is a quick view of a focused set of content from an app. Ideally, it is timely and contextually relevant. Viewed together, Glances are a browsable collection of meaningful moments across the wearer's favorite apps. And Glances are pull, rather than push. So unlike an alert pushed to a device, Glances are accessed at the wearer's discretion.



Glances are:

Template-based. There are separate templates for the upper and lower portions of the Glance screen. Use Xcode to pick the templates you want and then design your content to those specifications.

Not scrollable. All content must fit on a single screen.

Read-only. Tapping anywhere on the glance opens the app.

Optional. Not all apps need a Glance, and users can select which Glances to display.

The area at the bottom of the Glance is reserved for the page indicator dots.

Configure the Glance based on the user's current context. Stale or irrelevant information makes a glance less useful. Use time and location to reflect what is relevant to the user right now.

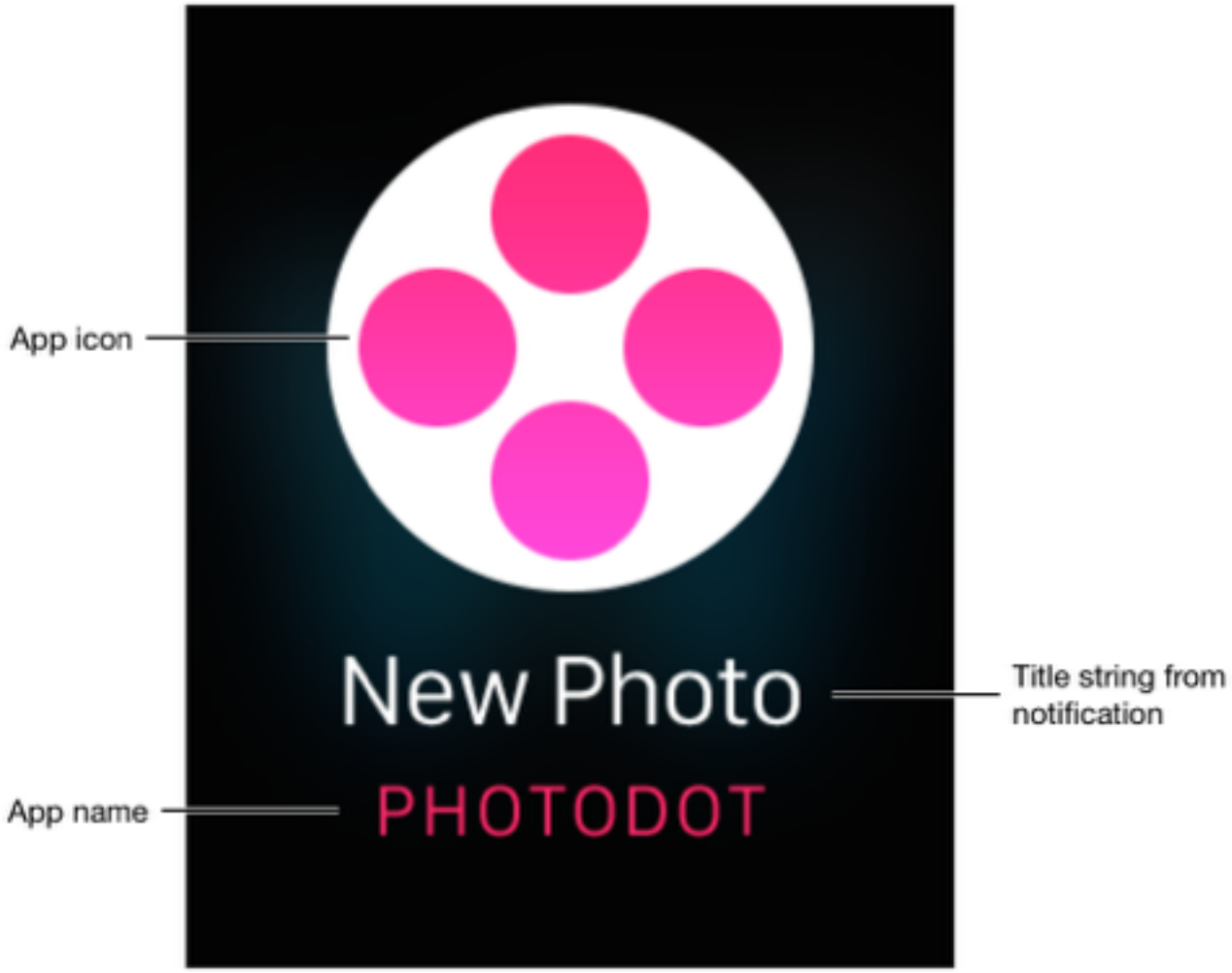
Glances can deep link into their corresponding app. Glances can make use of Handoff to inform the app what the Glance was displaying when the user tapped it. The app can then use this information to present a different interface or configure the existing interface accordingly.
Glances must provide useful content to the user. Do not provide a Glance simply to facilitate the launching of your app.

Notifications

Notifications on Apple Watch facilitate quick, lightweight interaction in two parts: the Short Look and the Long Look. A Short Look appears when a local or remote notification needs to be presented to the user. A Short Look provides a discreet, minimal amount of information—preserving a degree of privacy. If the wearer lowers his or her wrist, the Short Look disappears. A Long Look appears when the wearer's wrist remains raised or the user taps the short look interface. It provides more detailed information and more functionality—and it must be actively dismissed by the wearer.
Be sensitive to the frequency with which you send notifications to users. Users might perceive a frequent notifications as annoying and disable notifications for your app on Apple Watch. Always make sure notifications are relevant to what the user wants.

Short Look Notifications

Short Looks let the user know which app received a notification and are visible only briefly. The Short Look interface is template-based and contains the app name, app icon, and the title string from the accompanying notification. The system displays the app name using the app's key color.



Keep title strings short and focused. The space available for displaying title strings is minimal, so keep them brief and to the point. Title strings do not provide detail about the notification. They only provide a brief hint about what the notification is about.

Custom Long Look Notifications

Long Looks provide more detail about an incoming notification. The system provides a default long look appearance but apps can customize the Long Look to incorporate custom graphics and branding. The structure of the Long Look is the same for all apps. The system provides a sash and overlays app icon and app name on top of it. It also adds a Dismiss button and any app-defined action buttons at the end of the notification. In between is your custom content.

App content can underlap the sash or start just below it. Use the underlapping option for photos and other graphical content. For notification interfaces that contain mostly text, enable the Offset Notification Content option in Xcode to start that content below the sash.

For custom Long Look interfaces, apps must provide a static interface and may optionally provide a dynamic interface. The dynamic interface is more configurable than the static interface but both display the same notification type using your graphics and branding. The static interface provides a fallback position in cases where the dynamic interface is unavailable.

Long look notifications can display up to four custom action buttons. Apple Watch leverages the interactive notifications registered by your iOS app to display action buttons in the Long Look interface. These action buttons are displayed automatically based on the notification's category.

The Dismiss button is always present. This button is in addition to the four action buttons you provide.

For information about static and dynamic interfaces, and about how to configure action buttons, see *WatchKit Programming Guide*.

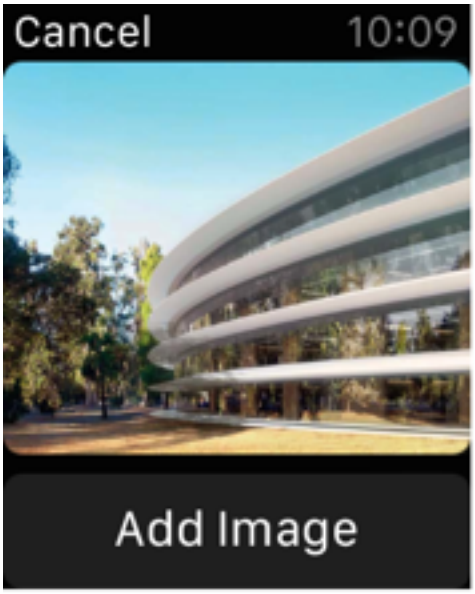
Modal Sheets

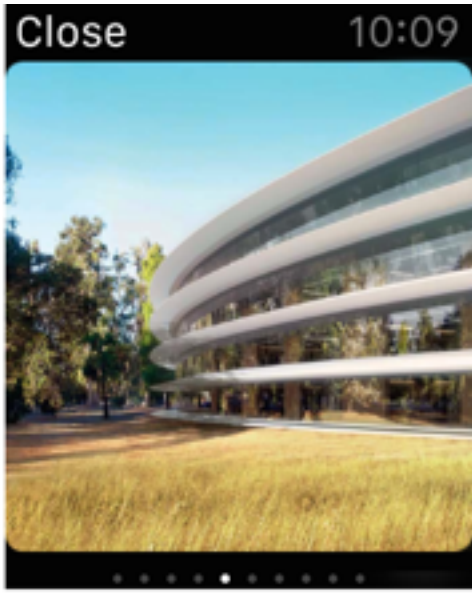
Modal sheets have advantages and disadvantages. It can give users a way to complete a task, to get information without distractions, or to continue a choice that was made initially in the Force Touch menu. A modal sheet achieves this by temporarily preventing the user from interacting with the rest of the app.

It's best when you can minimize the number of modal experiences in your app. In general, consider creating a modal context only when:

- It is critical to get the user's attention.
- A self-contained task must be completed—or explicitly abandoned—to avoid leaving the user's data in an ambiguous state.

Modal interfaces contain a single screen or they contain multiple screens in a page-based layout. The only difference between the two interfaces is the addition of dot indicators at the bottom of the page-based interface.





Single **Paged**

The top-left corner of a modal interface is reserved for the close button. When the user taps this button (or performs a left-edge swipe), the system dismisses the modal interface without any further actions. The presence of the close button is mandatory but you can change its text to convey meaning. You do not need to add a separate close button in the body of your content. Typical titles for the close button are "Close" and "Cancel." The color of the close button is always white.

If the task requires acceptance, provide the accept button in the body of the modal interface. Use a standard button for acceptance actions. Tapping

the button should dismiss the modal interface in addition to performing the appropriate actions.

Keep modal tasks simple. Avoid displaying a second modal interface from your initial modal interface.

Layout

Layout Guidelines

Limit the number of side-by-side controls in your interface. When placing buttons side-by-side, use icons instead of text to identify the buttons. Never place more than three items side-by-side. Including more items makes the tap targets too small for the user.

Use the space all the way up to the edges of the screen. Because the bezel on Apple Watch adds visual padding around your content, you do not need to include margins between the screen edge and your content. Note that this padding is not present in iOS Simulator.

Apps use relative positioning for items. Because apps display the same interface on different sizes of Apple Watch, relative positioning lets items expand as needed to fill the available space.

Prefer the use of left alignment during layout. Elements of your interface are laid out top-to-bottom and left-to-right. Aligning elements from the left edge ensures that they have room to expand and display their content fully.

Make text buttons full width. Buttons that display a label should always be created full width to ensure that the entire button label is always visible.

Use the context menu to present secondary actions. Rather than adding buttons to your interface, use the context menu for actions that are used less frequently.

Screen Sizes

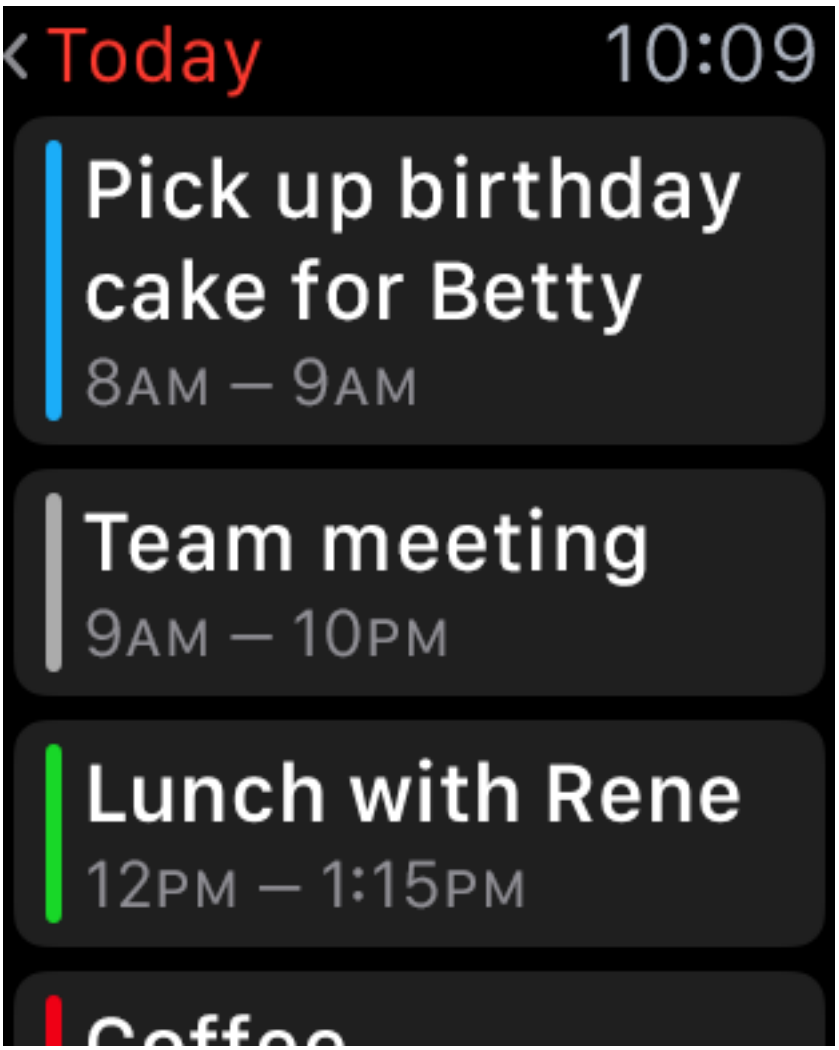
The content you display on Apple Watch should be the same regardless of the device's screen size. When designing your layout, let items expand or contract naturally to fill the available space.

Provide image assets for different screen sizes as needed. Use the same image resource if it looks good on both screen sizes; otherwise, provide separate image resources for each screen size.

Color and Typography

Color

Color helps provide visual continuity and branding for your app.



Use black for your app's background color. A black background blends seamlessly with the device bezel and maintains the illusion of there being no screen edges. Avoid bright background colors in your interface.

Use your app's key color for branding or status. Every app defines a key color. The system uses this key color for the title string in the upper-left corner of the screen and in notification interfaces to highlight your app name or other key information. You should similarly use the key color as part of the branding of your app.

Use high contrast colors for text. High contrast colors make text more legible.

Avoid using color to show interactivity. Apply color as appropriate for your branding but do not use color solely to indicate interactivity for buttons and other controls.

Be aware of color blindness. Most color blind people have difficulty distinguishing red from green. Test your app to make sure that there are no places where you use red and green as the only way to distinguish between two states or values (some image-editing software includes tools that can help you proof for color-blindness).

Color communicates, but not always in the way you intend. Everyone sees color differently, and many cultures differ in how they assign meanings to colors. Spend time to research how your use of color might be perceived in other countries and cultures. As much as possible, you want to be sure that the colors in your app send the appropriate message.

Typography

Above all, text must be legible. If users can't read the words in your app, it doesn't matter how beautiful the typography is.

The system font was designed specifically for legibility on Apple Watch. At large sizes, the font's slightly condensed letters are set tight to take up less horizontal space. But at small sizes, they are spaced more loosely and have bigger apertures in glyphs like 'a' and 'e' to make these easier to read at a glance. Punctuation is also proportionally larger when the font gets smaller. And as text size changes, Apple Watch dynamically switches between fonts to maintain clarity and legibility at all times.

Apps should always adopt Dynamic Type. When you adopt Dynamic Type, you get:

- Automatic adjustments to letter spacing and line height for every font size
- The ability to specify different text styles for semantically distinct blocks of text, such as `Body`, `Footnote`, or `Headline`
- Text that responds appropriately to changes the user makes to text-size settings (including accessibility text sizes)

NOTE

If you use a custom font, you can still scale type according to the system setting for text size. Your app is responsible for responding appropriately when the user changes the setting.

If you use the built-in text styles, you get Dynamic Type support for free. If you use custom fonts, you must do some work to adopt the feature. To learn how to use text styles and ensure that your app gets notified when the user changes the text size setting, see Text Styles in *Text Programming Guide for iOS*.

Use the built-in text styles whenever possible. The built-in styles automatically support Dynamic Type and were designed to look good on Apple Watch.

Prefer the use of a single font throughout your app. Apps can use an additional font for branding purposes, but use that extra font sparingly. Mixing many different fonts can make your app seem fragmented and sloppy. Use the `UIFont` text styles API to define different areas of text according to semantic usage, such as body or headline.

When specifying system font sizes manually, the point size determines the correct size to use. Choose the San Francisco Text font for text that is 19 points or smaller. Choose the San Francisco Display font for text that is 20 points or larger.

Animations

Beautiful, subtle animation pervades Apple Watch and makes the experience more engaging and dynamic for the user. Appropriate animation can:

- Communicate status and provide feedback
- Help people visualize the results of their actions

Create prerendered animations using a sequence of static images. Store canned animations in your Watch app bundle so that they can be presented quickly to the user. Canned animations also let you deliver high frame rates

and smoother animations. Creating animations dynamically from your WatchKit extension and transferring them to Apple Watch adds a delay before playback can begin.

Playback controls are available only for image and group objects. Most interface objects display animated image sequences in an endless loop. To stop an animation or to playback a specific set of frames from the animation, you must use an image or group object.

Branding

Successful branding involves more than adding brand assets to an app. The best apps integrate existing assets with a unique look and feel to give users a delightful, memorable experience.

There are many ways to brand your app, including icons, colors, custom buttons, custom fonts, and the text you use for your copy. As you design your app's graphical elements, remember that each custom element should look good and function well by itself, but it should also look like it belongs with the other elements in the app, whether they're custom or standard.

Incorporate a brand's assets in a refined, unobtrusive way. People use your app to get things done or be entertained; they don't want to feel as if they're being forced to watch an advertisement. For the best user experience, you want to quietly remind users of the brand identity through your choice of font, color, and imagery.

Resist the temptation to display your logo in your app and Glance. Space on Apple Watch is at a premium, and every occurrence of a logo takes space away from the content that users want to see. What's more, displaying a logo in an app doesn't serve the same purpose as displaying it in a webpage: It's common for users to arrive on a webpage without knowing its owner, but users will usually see your app icon before opening the app.