Drag and drop

Using drag and drop, people can move or duplicate selected photos, text, and other content by dragging the selection from one location to another.

Supported platforms

To perform drag and drop, people select content in one location, called the *source*, and drop it in another, called the *destination*. These locations can be in the same container — like a text view — or in different containers, like text views on opposite sides of a split view, or even in different apps.

Depending on various factors, the drag and drop action might *move* the selected content to the destination or *copy* it. After a successful drop, moved content exists only in the destination; copied content exists in both locations. As a general rule, dropping selected content within the same container moves it, whereas dropping content in a different container copies it. Dragging and dropping content between apps always results in a copy.

People use different interactions to perform drag and drop depending on platform. For example:

- In visionOS, people pinch and hold a virtual object while dragging it to a new location in any direction, including along the z-axis.
- iOS and iPadOS support drag and drop through gestures on the touchscreen, interactions with a pointing device, and through full keyboard-access mode.
- Universal Control lets people drag content between their Mac and iPad.
- On a Mac, people can interact with a pointing device, use full keyboard access mode, or use VoiceOver to perform drag and drop.

Best practices

As much as possible, support drag and drop throughout your app. Most people are familiar with drag and drop and they often try it everywhere. When you use system-provided components — such as text fields and text views — you get built-in support for drag and drop.

Offer alternative ways to accomplish drag-and-drop actions. Sometimes, drag-and-drop operations are inconvenient or impossible for people to perform, so it's important to provide other ways to do the same things. For example, you can include menu commands that people can use to copy an item and move it to another location. In iOS and iPadOS, you can use accessibility APIs to identify sources and destinations so that people can use assistive technologies to drag and drop in your app (for developer guidance, see `accessibilityDragSourceDescriptors` and `accessibilityDropPointDescriptors`).

Determine when dragging and dropping content within your app results in a move or a copy. In general, a move makes sense when the source and destination containers are the same — such as dragging text from one location to another within a document — and a copy makes sense when they're different, like dragging an image from one document to another. Before you change these defaults, consider the behavior that most people expect and prefer the one that is least likely to result in frustration or data loss.

Support multi-item drag and drop when it makes sense. People appreciate the convenience of dragging a group of items to a destination, instead of dragging each item separately. In iOS, iPadOS, macOS, and visionOS, people can select multiple items and drag them as a group; macOS also lets people select multiple items from several apps and drag them as a group. In iPadOS, people can select an item, start dragging it, and add other items to the group without stopping the drag operation.

Prefer letting people undo a drag-and-drop operation. Sometimes, people inadvertently drop content in the wrong destination, so they appreciate being able to undo the action and return to their previous state. You might also be able to help people avoid mistakes by asking for confirmation before completing a drag-and-drop operation that can't be undone. In macOS, for example, the Finder asks for confirmation when people drag a file into a write-only folder because they won't be able to open the folder and remove the dropped item. In some situations, it might make sense to provide a way to reverse the results of drag and drop when people can't undo it. For example, Photos lets people cancel photo sharing after dropping a photo into a shared photo stream.

Consider offering multiple versions of dragged content, ordered from highest to lowest fidelity. By providing multiple alternatives, the destination can choose the highest quality version it can accept. For example, if people can drag a line drawing they created in your app, you could offer a PDF vector representation, a lossless PNG image with transparency, and a lossy JPEG image without transparency, in that order. Another example is an app that uses rich, complicated objects, like charts. This app might offer the native chart object followed by a simpler version — like an image of the chart — for destinations that don't support chart objects.

Consider supporting spring loading. Spring loading lets people activate certain controls, like buttons and segmented controls, by dragging selected content over them. For example, Calendar lets people drag a selected event over the day, week, month, or year segments in the toolbar, giving them a convenient way to move the event to a different date. On a Mac equipped with a Magic Trackpad, a button or segmented control can activate when people force-click it while continuing to hold the content; on iPad, these components can activate when people hover over them while holding the content.

Providing feedback

Drag and drop is a dynamic process that can result in multiple outcomes. To help people feel in control the process, it's crucial to provide clear and continuous feedback throughout.

Display a drag image as soon as people drag a selection about three points. It works well to create a translucent representation of the content people are dragging. Translucency helps distinguish the representation from the original content and lets people see destinations as they pass over them. Display the drag image until people drop the content.

If it adds clarity, modify the drag image to help people predict the result of a drag-and-drop operation. For example, when dragging a photo into a document, the drag image could expand to show the default size of the photo in the document. You can also use drag *flocking* to visually group multiple drag items — letting people confirm that they haven't missed an item they want to drag — and then ungroup the items when people drop them. Although changing the drag image can provide valuable feedback, avoid creating a distracting experience in which the drag image is constantly and radically changing.

Show people whether a destination can accept dragged content. For example, you might display an insertion point or highlight a containing view only when the destination can accept a dragged item, and show no visual feedback — or an explicit "not allowed" image, like the `circle.slash` from SF Symbols — when it can't. Display highlighting or other visual cues only while the content is positioned above the destination, removing the visual feedback when people drag the content away. When there are multiple possible destinations, provide visual cues that help people identify one at a time.

When people drop an item on an invalid destination, or when dropping fails, provide visual feedback. For example, the item can move back from its current location to its source (if the source is still visible) or it can scale up and fade out to give the impression of the item evaporating instead of landing successfully.

Accepting drops

Scroll the contents of a destination when necessary. When people drag an item within a scrolling container that has a lot of content, the content can automatically scroll as people move the item over it. This behavior makes it easy for people to find the right place to drop the item, but if they continue the drag operation outside of the container, automatic scrolling is no longer necessary. System-provided text views and text fields behave this way by default.

When there's a choice, pick the richest version of dropped content your app can accept. For example, if people drag a chart object from another app, the drag operation might offer both the rich, native chart object and a simple image of it. If your app supports charts, extract and display the native chart object; it it doesn't, use the image instead.

Extract only the relevant portion of dropped content if necessary. For example, when people drag a contact to a recipient field in an email, Mail displays only the name and email address, not the contact's address information.

When a physical keyboard is attached, check for the Option key at drop time. When people hold the Option key while dragging, they can force a drag-and-drop operation within the same container to behave like a copy. If people stop holding Option before dropping content in the same container, the drag operation results in a move.

Provide feedback when dropped content needs time to transfer. For example, you might display a progress indicator to help people estimate how long the transfer will take. In collections, lists, and tables, you might also display a placeholder at the drop location so people know where to find the content after it finishes transferring. The system can display an alert when a time-consuming transfer occurs between apps.

Provide feedback when dropped content initiates a task or action. If people drop content onto a control that initiates a task — such as printing — show people that the task has begun and keep them informed of its progress.

Apply appropriate styling to dropped text. When the source and destination both support the same text styles, make sure dropped text maintains its original font, typeface, size, and other attributes. Otherwise, apply the destination's style to dropped text.

After a drop, maintain the content's selection state in the destination, updating it in the source as needed. People expect the content they drop to remain selected so they can immediately act on it. When the source and destination are the same container, the content disappears from its original location when the drag operation performs a move. When a drag operation within the same container performs a copy, remove the selection state from the content that remains in the original location. When people drag selected content to a different container, deselect the content in the source.

Platform considerations

Not supported in tvOS or watchOS.

iOS, iPadOS

Let people perform multiple simultaneous drag activities. In iPadOS, people can sequentially add items to an in-progress drag session, gathering as many items as their fingers can handle. For example, people can select an app icon on the Home Screen, start dragging it, and select additional app icons before dropping all of them in a different Home Screen or in a folder. To support this interaction, you need to let people add items during a drag — providing visual feedback through flocking — and accept multiple, simultaneous drops.

macOS

Consider letting people drag content from your app into the Finder. When you support this, be sure to present the content in a format your app can open later. For example, Calendar lets people drag an event to the Finder as a `.ics` file. People can share this file with others or drag it back to Calendar to open it. When necessary, you can output dragged content in a *clipping*, which is a temporary container for storing dragged content. For example, most system apps let people drag text to the Finder, where it appears as a clipping. Later, people can drag the clipping into a text field or other location that accepts text. Note that a drag-and-drop clipping isn't related to the Clipboard.

Let people drag selected content from an inactive window without first making the window active. Selected content in an inactive window is known as a *background selection* and has a different appearance from selected content in the active window. In general, people expect to drag a background selection to the active window without bringing the inactive window forward.

When possible, let people drag individual items from an inactive window without affecting an existing background selection. For example, people can drag an unselected file from an inactive Finder window without deselecting any of the window's selected files.

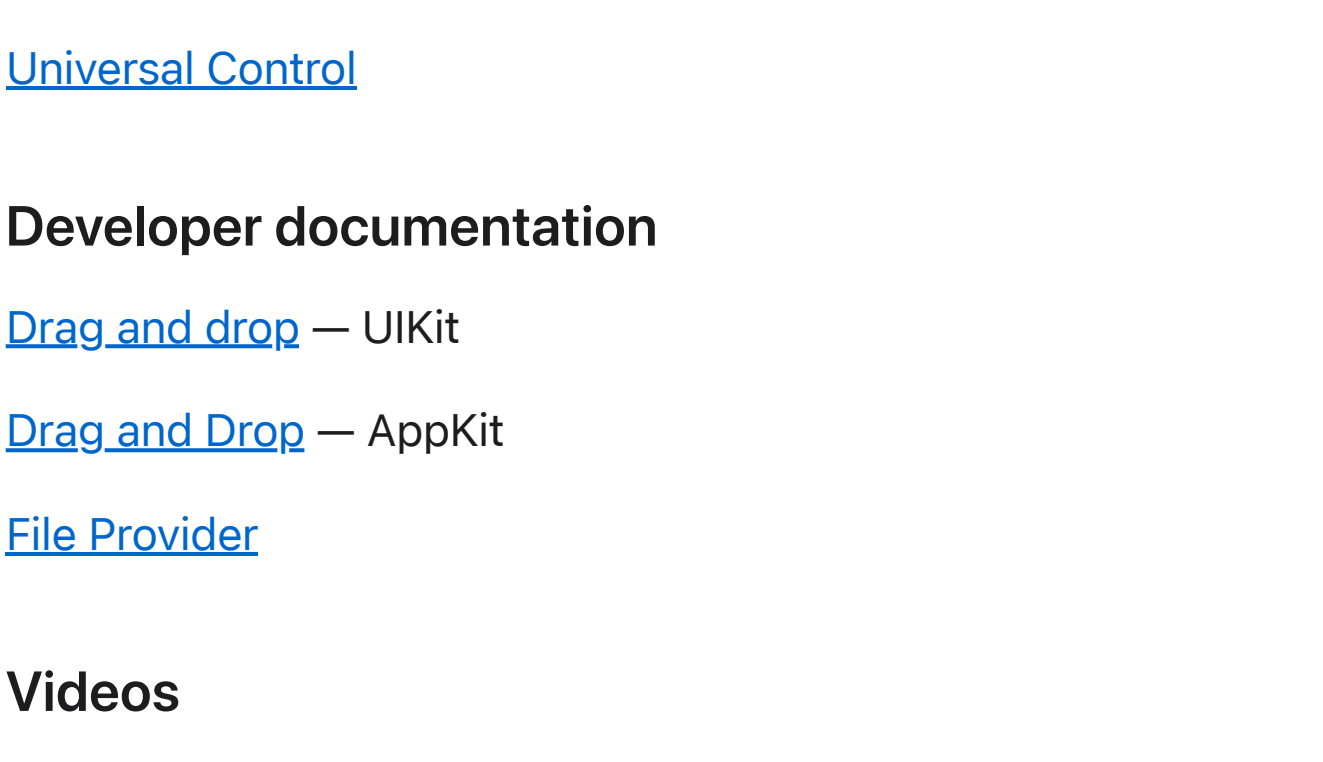
Consider displaying a badge during multi-item drag operations. A badge is a small filled oval containing a number you can use to indicate the number of items people are dragging. If a destination can accept only a subset of dragged items, update the badge to show the new number.

Consider changing the pointer appearance to indicate what will happen when people drop content. In addition to using the *copy* pointer, you might want to use the *drag link*, *disappearing item*, and *operation not allowed* pointers, depending on the situation. For guidance, see Pointers.

As much as possible, let people select and drag content with a single motion. Unless people are selecting multiple items, they appreciate it when they don't have to pause between making a selection and starting the drag operation.

visionOS

When possible, launch your app to handle content that people drop into empty space. When you associate a user activity with draggable app content, your app can open a window or scene that handles the content when people drop it. For example, when people drop a URL into empty space, it launches Safari; when people drop Quick Look–supported content, Quick Look launches to display it. For developer guidance, see `NSUserActivity`.



Play

Resources

Related

Universal Control

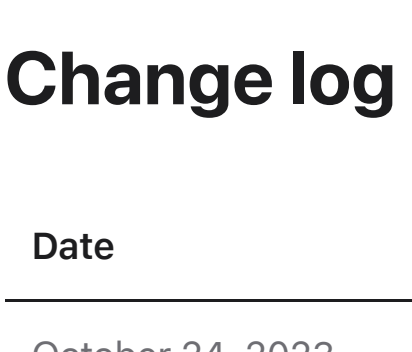
Developer documentation

Drag and drop — UIKit

Drag and Drop — AppKit

File Provider

Videos



What's new in UIKit

SwiftUI on the Mac: The finishing touches

Designed for iPad

Change log

Date	Changes
October 24, 2023	Added artwork.
June 21, 2023	Updated to include guidance for visionOS.