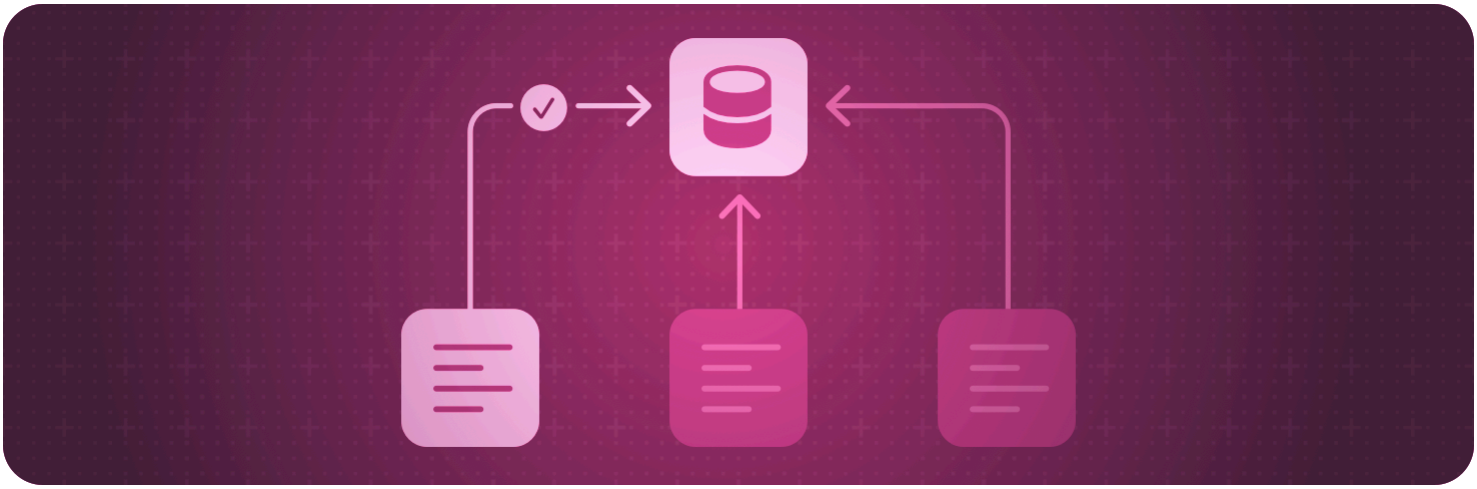
API Collection

Preferences

Indicate configuration preferences from views to their container views.

Overview

Whereas you use the environment to configure the subviews of a view, you use preferences to send configuration information from subviews toward their container. However, unlike configuration information that flows down a view hierarchy from one container to many subviews, a single container needs to reconcile potentially conflicting preferences flowing up from its many subviews.



When you use the `PreferenceKey` protocol to define a custom preference, you indicate how to merge preferences from multiple subviews. You can then set a value for the preference on a view using the `preference(key:value:)` view modifier. Many built-in modifiers, like `navigationTitle(_:)`, rely on preferences to send configuration information to their container.

Topics

Setting preferences

```
func preference<K>(key: K.Type, value: K.Value) -> some View
```

Sets a value for the given preference.

```
func transformPreference<K>(K.Type, (inout K.Value) -> Void) -> some View
```

Applies a transformation to a preference value.

Creating custom preferences

```
protocol PreferenceKey
```

A named value produced by a view.

Setting preferences based on geometry

```
func anchorPreference<A, K>(key: K.Type, value: Anchor<A>.Source, transform: (Anchor<A>) -> K.Value) -> some View
```

Sets a value for the specified preference key, the value is a function of a geometry value tied to the current coordinate space, allowing readers of the value to convert the geometry to their local coordinates.

```
func transformAnchorPreference<A, K>(key: K.Type, value: Anchor<A>.Source, transform: (inout K.Value, Anchor<A>) -> Void) -> some View
```

Sets a value for the specified preference key, the value is a function of the key's current value and a geometry value tied to the current coordinate space, allowing readers of the value to convert the geometry to their local coordinates.

Responding to changes in preferences

```
func onPreferenceChange<K>(K.Type, perform: (K.Value) -> Void) -> some View
```

Adds an action to perform when the specified preference key's value changes.

Generating backgrounds and overlays from preferences

```
func backgroundPreferenceValue<Key, T>(Key.Type, (Key.Value) -> T) -> some View
```

Reads the specified preference value from the view, using it to produce a second view that is applied as the background of the original view.

```
func backgroundPreferenceValue<K, V>(K.Type, alignment: Alignment, (K.Value) -> V) -> some View
```

Reads the specified preference value from the view, using it to produce a second view that is applied as the background of the original view.

```
func overlayPreferenceValue<Key, T>(Key.Type, (Key.Value) -> T) -> some View
```

Reads the specified preference value from the view, using it to produce a second view that is applied as an overlay to the original view.

```
func overlayPreferenceValue<K, V>(K.Type, alignment: Alignment, (K.Value) -> V) -> some View
```

Reads the specified preference value from the view, using it to produce a second view that is applied as an overlay to the original view.

See Also

Data and storage

- Model data

  Manage the data that your app uses to drive its interface.

- Environment values

  Share data throughout a view hierarchy using the environment.

- Persistent storage

  Store data for use across sessions of your app.