Sample Code

Showing help tags for views and controls using tooltip interactions

Explain the purpose of interface elements by showing a tooltip when a person positions the pointer over the element.

Download

iOS 15.0+ | iPadOS 15.0+ | Mac Catalyst 15.0+ | Xcode 13.0+

Overview

This sample shows how to display a tooltip (also known as a *help tag*) that explains the purpose of, or provide additional information about, an interface element such as a view or control without shifting a person's focus away from the primary interface. A tooltip appears when a person positions the pointer over a view or control for a few seconds. The tooltip remains visible for a few seconds or until the pointer moves away from the interface element.

For guidelines on designing the content of your tooltips, see the Help section of the macOS Human Interface Guidelines.

Configure the sample code project

Tooltips are available in the sample app when running in macOS 12 or later. To build and run the sample app:

1. Choose Product > Destination > My Mac.
2. Click Product > Run.

Add a tooltip to a view

To show a tooltip when the pointer hovers over a view, the sample creates a UIToolTip Interaction object and passes in the default tooltip text. Then the sample adds the interaction to a UIView using the view's addInteraction(_:) method.

```
lazy var viewWithDefaultTooltip: UIView = {
    let view = UIView()
    view.backgroundColor = UIColor.systemGreen
    view.addText("Hover the pointer over this view to see its default tooltip.")

    let tooltipInteraction = UIToolTipInteraction(defaultToolTip: "The default toolt
    view.addInteraction(tooltipInteraction)

    return view
}()
```

The sample uses the same approach for other types of views too, for instance, to add a tooltip to an instance of UILabel:

```
lazy var labelWithTooltip: UILabel = {
    let label = UILabel()
    label.text = "Hover the pointer over this label to see its tooltip."
    label.numberOfLines = 0

    let tooltipInteraction = UIToolTipInteraction(defaultToolTip: "The label's toolt
    label.addInteraction(tooltipInteraction)

    return label
}()
```

Add a tooltip to a control

For interface elements that derive from UIControl, such as UIButton, the sample uses the toolTip property to set the default text that appears in the tooltip instead of adding an interaction to the control.

```
lazy var buttonWithTooltip: UIButton = {
    var configuration = UIButton.Configuration.filled()
    configuration.title = "Buy"
    configuration.subtitle = "Only $9.99"
    configuration.titleAlignment = .center
    let action = UIAction { _ in print("Thank you for your purchase.") }

    let button = UIButton(configuration: configuration, primaryAction: action)
    button.toolTip = "Click to buy this item. You'll have a chance to change your mi
    button.preferredBehavioralStyle = .pad

    return button
}()
```

Change the tooltip text of a view

Setting the defaultToolTip property of a tooltip interaction and setting the toolTip property of a control are convenient ways to show tooltip text that don't change while the app runs. However, there may be times when an app needs to determine the contents of the tooltip based on the state of the app or logic specific to the app. For instance, the sample app looks up the name of a view's background color and displays the name in a tooltip when the pointer hovers over that view.

To show the background color name in a tooltip, the sample creates an instance of UIToolTipInteraction and sets its delegate property to an instance of ViewWithBackgroundColorTooltip, which is a custom view that conforms to the UIToolTipInteractionDelegate protocol. Then the sample adds the interaction to the view.

```
lazy var viewWithBackgroundColorTooltip: UIView = {
    let view = ViewWithBackgroundColorTooltip()
    view.backgroundColor = UIColor.systemYellow
    view.addText("Hover the pointer over this view to see the name of the view's bac

    let tooltipInteraction = UIToolTipInteraction()
    tooltipInteraction.delegate = view
    view.addInteraction(tooltipInteraction)

    return view
}()
```

Next, ViewWithBackgroundColorTooltip implements the toolTipInteraction(_:configurationAt:) method, and returns a UIToolTipConfiguration object that contains the name of the background color as the text of the tooltip. If the color name isn't available, the method returns nil, which prevents the tooltip from displaying.

```
class ViewWithBackgroundColorTooltip: UIView, UIToolTipInteractionDelegate {

    func toolTipInteraction(_ interaction: UIToolTipInteraction, configurationAt poi

        let configuration: UIToolTipConfiguration?
        if let accessibilityName = backgroundColor?.accessibilityName {
            configuration = UIToolTipConfiguration(toolTip: "The color is \(accessib
        } else {
            configuration = nil
        }

        return configuration
    }

}
```

Change the tooltip text of a control

To change the tooltip text of a control like the sample's shopping cart button, the sample provides a tooltip interaction delegate as it did with the custom view. But instead of creating a tooltip interaction, it sets the button's `toolTip` property to the default text. This causes the control to create a `UIToolTipInteraction` object and assign it to the `toolTipInteraction` property. The sample then uses the property to assign a delegate to the interaction.

```
lazy var shoppingCartButtonWithTooltip: UIButton = {
    var configuration = UIButton.Configuration.filled()
    configuration.title = "Add to Cart"
    configuration.image = UIImage(systemName: "cart.circle")
    configuration.imagePlacement = NSDirectionalRectEdge.leading
    configuration.imagePadding = 4

    let action = UIAction { [unowned self] _ in self.cartItemCount += 1 }

    let button = UIButton(configuration: configuration, primaryAction: action)
    button.toolTip = "Click to add the item to your cart. Your cart is empty."
    button.toolTipInteraction?.delegate = self

    return button
}()
```

The sample also implements the `toolTipInteraction(_:configurationAt:)` delegate method, which returns a `UIToolTipConfiguration` object that contains the tooltip text describing the purpose of the button and the number of items in the shopping cart.

```
func toolTipInteraction(_ interaction: UIToolTipInteraction, configurationAt point:

    let text: String
    switch cartItemCount {
    case 0:
        text = "Click to add the item to your cart. Your cart is empty."
    case 1:
        text = "Click to add the item to your cart. Your cart contains \(cartItemCou
    default:
        text = "Click to add the item to your cart. Your cart contains \(cartItemCou
    }

    return UIToolTipConfiguration(toolTip: text)
}
```

Specify the hover region

In addition to setting the tooltip text in a `UIToolTipConfiguration` object, a delegate can specify the region of an interface element where the pointer must hover to trigger the display of the tooltip. The sample app, for example, shows a view that displays a tooltip after positioning the pointer over the top or bottom regions of the view, but not when the pointer is over the middle area.

To determine whether the pointer location is in the top or bottom region of the view, the sample uses the `point` value that the method `toolTipInteraction(_:configurationAt:)` provides. When the pointer is in one of those regions, the delegate method returns a tooltip configuration that contains the tooltip text and the source rectangle, which defines the area of the view that the pointer must hover over to trigger the display of the tooltip.

```
class ViewWithTooltipRegion: UIView, UIToolTipInteractionDelegate {

    func toolTipInteraction(_ interaction: UIToolTipInteraction, configurationAt poi

        var topRect = self.bounds
        var bottomRect = self.bounds

        let partHeight = self.bounds.size.height / 3
        topRect.size.height = partHeight
        bottomRect.size.height = partHeight
        bottomRect.origin.y = partHeight * 2

        // Display the tooltip if the pointer within the top or bottom rects.
        if topRect.contains(point) {
            return UIToolTipConfiguration(toolTip: "Top area of the view.", in: topR
        } else if bottomRect.contains(point) {
            return UIToolTipConfiguration(toolTip: "Bottom area of the view.", in: b
        }

        // Pointer is in the middle of the view; don't display a tooltip.
        return nil
    }

}
```

In another example, the sample uses the source rectangle of a tooltip configuration to specify the region of selected text in a text view. When hovering the pointer over the selected text, a tooltip appears but it doesn't appear when the pointer hovers over unselected text.

```
class TextViewWithTooltip: UITextView, UIToolTipInteractionDelegate {

    func toolTipInteraction(_ interaction: UIToolTipInteraction, configurationAt poi

        guard
            let selectedTextRange = self.selectedTextRange,
            selectedTextRange.isEmpty == false
        else {
            return nil
        }

        var unionedRect = firstRect(for: selectedTextRange)
        for selectionRect in selectionRects(for: selectedTextRange) {
            unionedRect = unionedRect.union(selectionRect.rect)
        }

        if let selectedText = text(in: selectedTextRange) {
            return UIToolTipConfiguration(toolTip: "Selected text: \(selectedText)",
        }

        return nil
    }
}
```

See Also

Tooltips

`class UIToolTipInteraction`

An interaction object that makes it possible to show a tooltip when hovering a pointer over a view or control.

`protocol UIToolTipInteractionDelegate`

An interface that provides tooltip settings to an interaction.