Sample Code

Responding to User Interaction

Update onscreen information by adding event listeners to your Apple TV app.

Download

tvOS 12.1+ | Xcode 11.2+

Overview

As the user navigates to a menu item along the top of the screen, the screen updates with information related to that menu item. This app shows you how to use events to update the screen when the user moves to a menu item, but doesn't select the item.

Configure the Sample Code Project

Before running the app, set up a local server on your machine:

1. In Finder, navigate to the Incorporating Event Listeners directory inside of the Incorporating Event Listeners directory.
2. In Terminal, enter `cd`, followed by a space.
3. Drag the Incorporating Event Listeners folder from the Finder window into the Terminal window, and press Return. The current directory changes to that folder.
4. In Terminal, enter `ruby -run -ehttpd . -p9001` to run the server.
5. Build and run the app in Apple TV Simulator.

After testing the sample app in Apple TV Simulator, you can quit the local server by pressing Control-C in Terminal or closing the Terminal window.

Add URL Information to the menuBar Template Code

Add the `selectTargetURL` attribute to each `menuItem` element inside of the `menuBar Template`. As the user moves between menu items, an event fires in order to update the onscreen information. The `selectTargetURL` attribute identifies the location of the TVML page associated with the highlighted menu item. An example of a formatted menu item follows:

```
<menuItem selectTargetURL="Server/Templates/firstmenutabpage.xml">
    <title>Tab 1</title>
</menuItem>
```

Set Up the Event Listeners

Use the `XMLHttpRequest` function to load the `menuBar` template and then add an event listener to the document using the JavaScript `addEventListener` function. The event listener notifies the app when the user moves to a new menu item. The app calls the user-defined `handleSelect Event` function and automatically sends all of the the events to the function.

```
if (request.status == 200) {
    var document = request.responseXML;
    document.addEventListener("select", handleSelectEvent);
    navigationDocument.replaceDocument(document, loadingDocument)
```

Respond to an Event

Use the `handleSelectEvent` function to verify that the event fired for the correct TVML element. The event's `target` property contains the highlighted element. The function then verifies that the highlighted element contains the `selectTargetURL` attribute. If the highlighted element is also a `menuItem` element and not a different element with the `selectTargetURL` attribute, the function calls the `updateMenuItem` function, which will update the displayed document, that is, the information associated with the highlighted menu item.

```
function handleSelectEvent(event) {
    var selectedElement = event.target;

    var targetURL = selectedElement.getAttribute("selectTargetURL");
    if (!targetURL) {
        return;
    }
    targetURL = baseURL + targetURL;
```

```
    if (selectedElement.tagName == "menuItem") {
        updateMenuItem(selectedElement, targetURL);
    }
    else {
        loadAndPushDocument(targetURL);
    }
}
```

Update the Document

Use the `updateMenuItem` function to load the new document from the server. After loading the new document, retrieve the menu item's parent node and the `MenuBarDocument` object. Finally, associate the new document to the `menuItem` element using the `setDocument`. This displays the document associated with the menu item.

```
function updateMenuItem(menuItem, url) {
    var request = new XMLHttpRequest();
    request.open("GET", url, true);

    request.onreadystatechange = function() {
        if (request.status == 200) {
            var document = request.responseXML;
            var menuItemDocument = menuItem.parentNode.getFeature("MenuBarDocument")
            menuItemDocument.setDocument(document, menuItem)
        }
    };

    request.send();
}
```

See Also

App Initialization

`App`

An object that provides access to—and a means to respond to—app life-cycle events.

`UserDefaults`

An object that contains the app's default preferences.

`NavigationDocument`

A document stack that holds the individual TVML documents for a client-server app.

`EventListenerObject`

An object that communicates events and allows other objects to add themselves as listeners.