

Apple Watch

Mobile Application Development in iOS

School of EECS

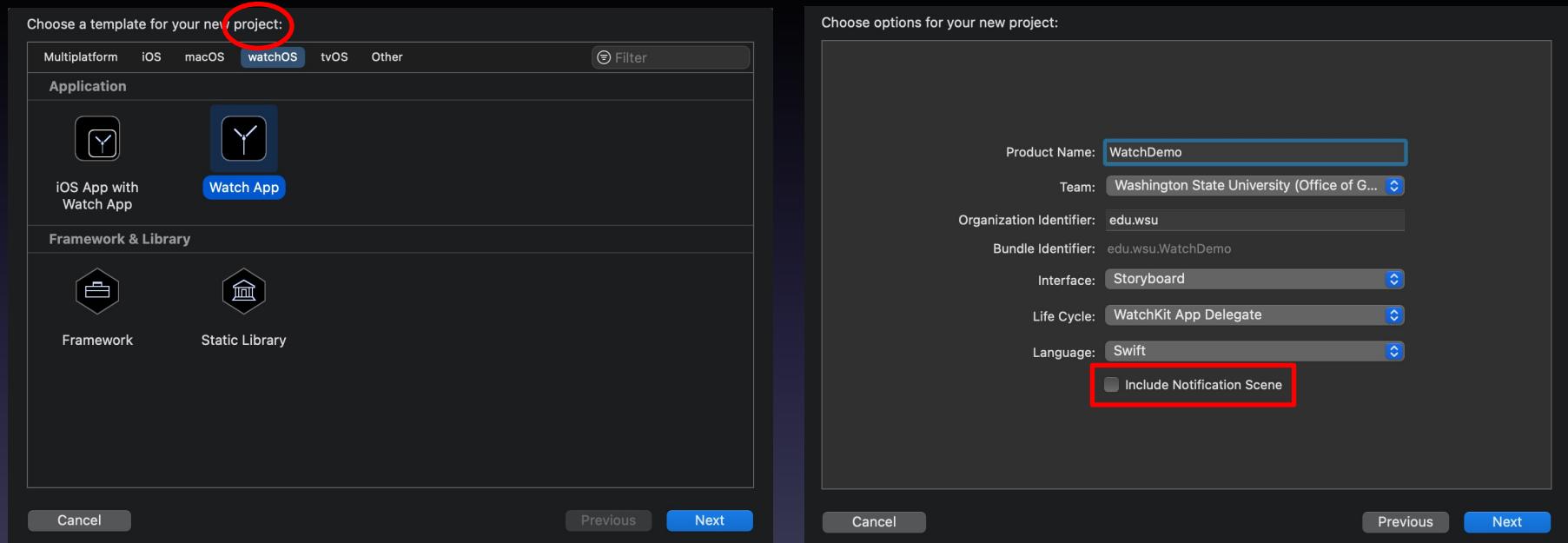
Washington State University

Outline

- Xcode configuration
- WatchKit
- Notifications
- Complications
- Sensors



Xcode Configuration: New Project

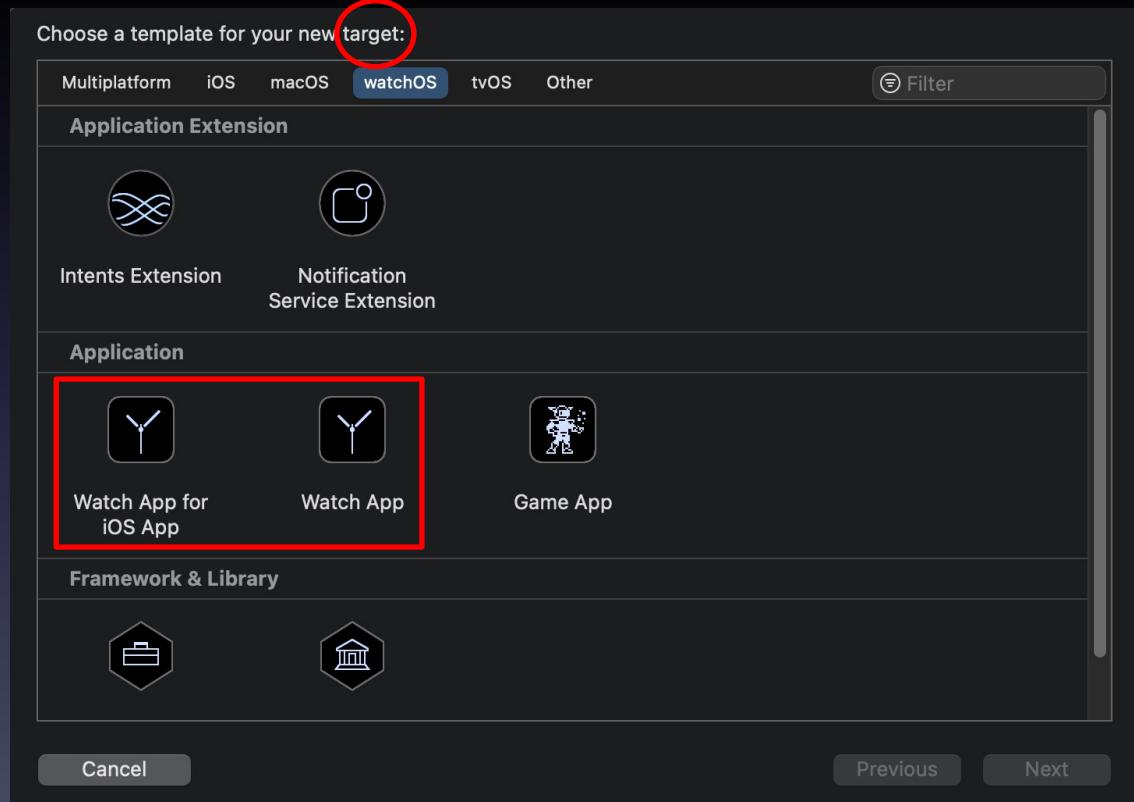


- iOS App with Watch App
- Standalone Watch App

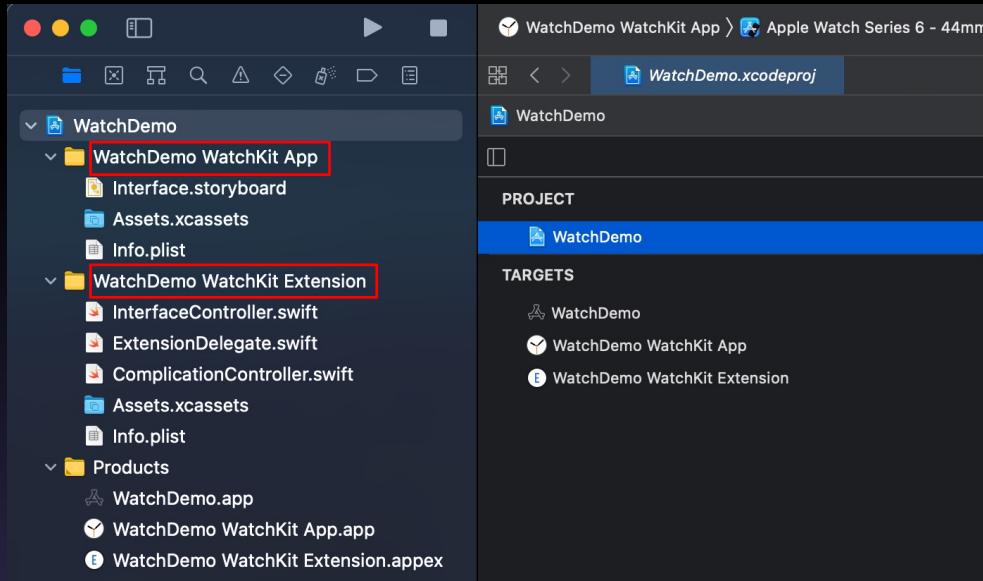
We'll do notifications later.

Xcode Configuration: Add to Existing Project

File → New → Target

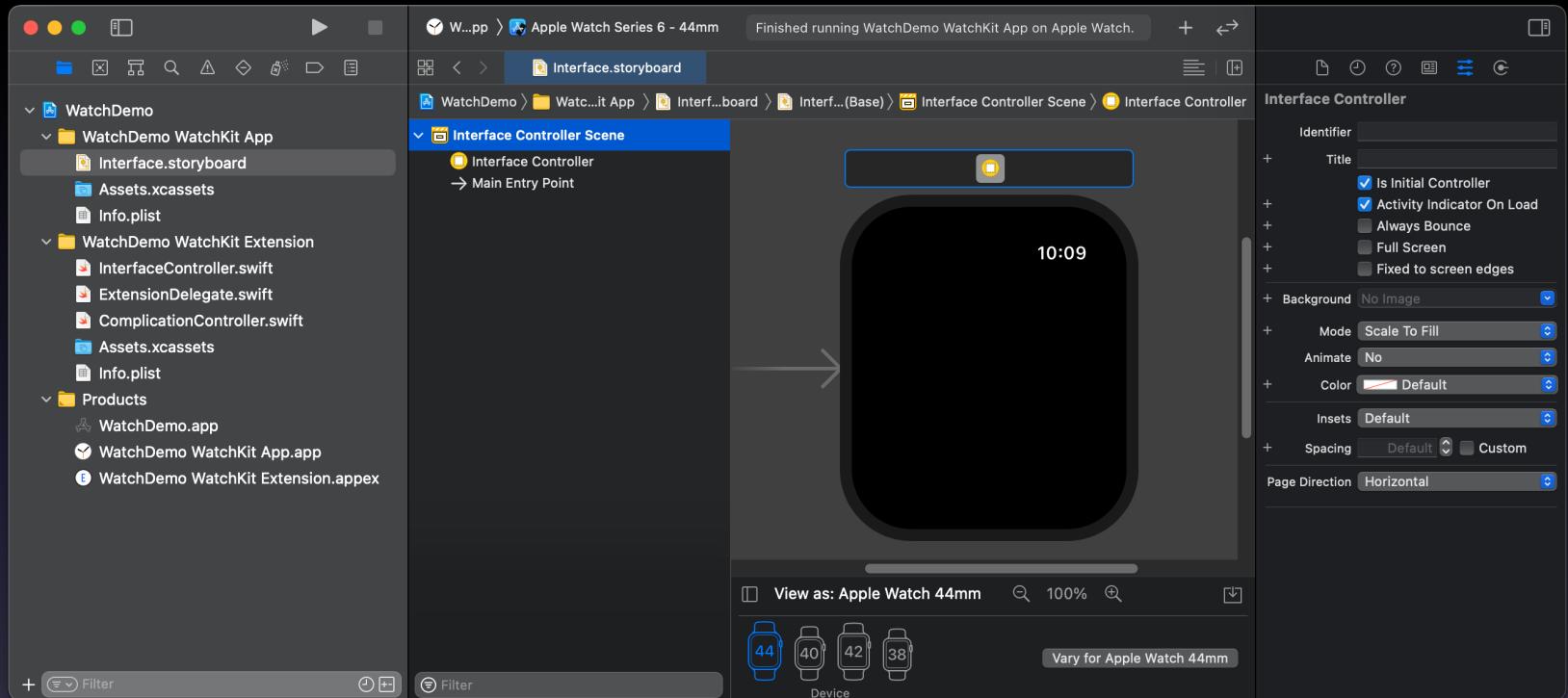


WatchKit App and Extension

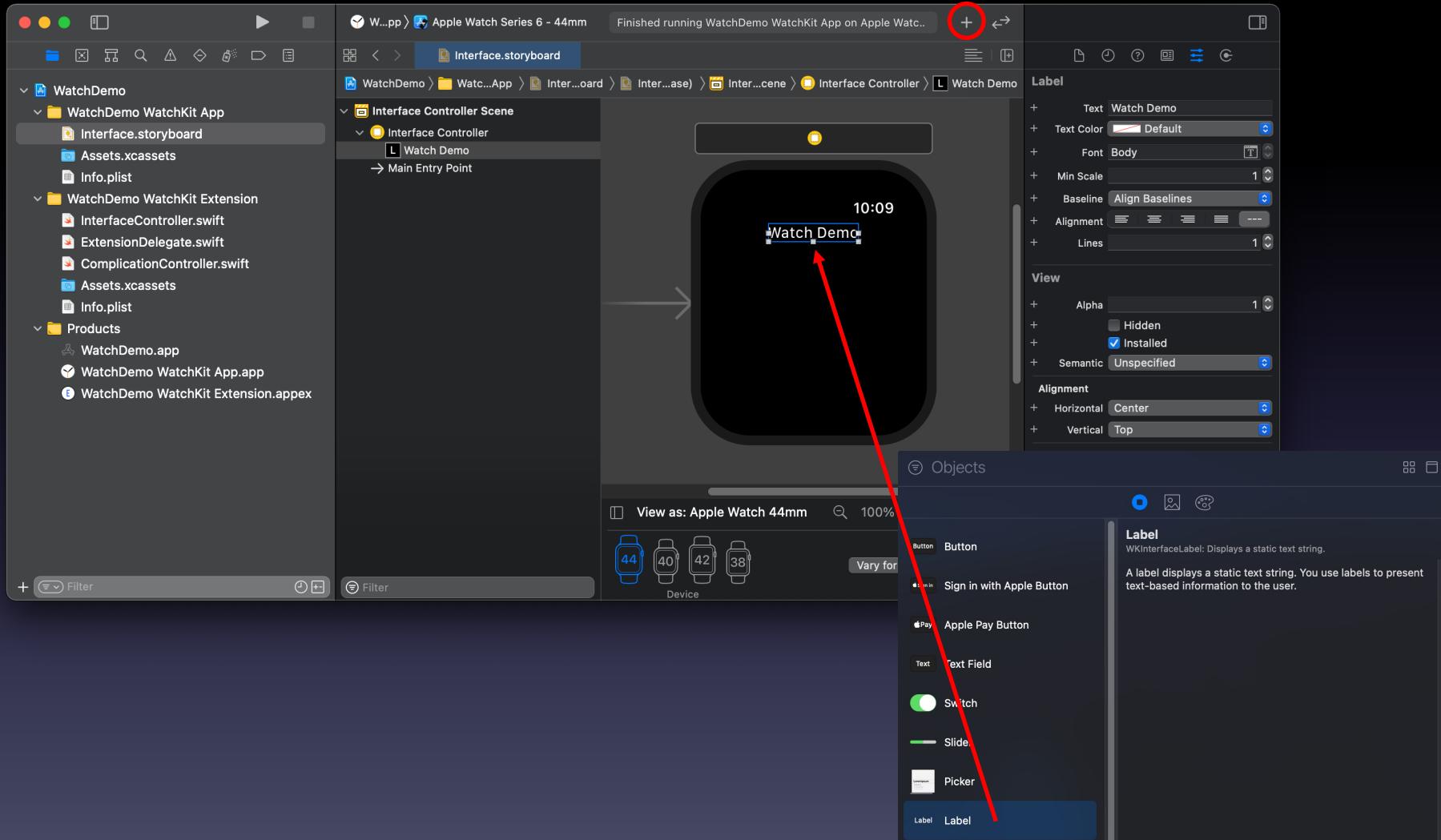


- WatchKit App
 - Storyboard
 - Storyboard assets
- WatchKit Extension
 - App code
- Simulator
 - Window → Devices and Simulators
 - "Pair" a watch

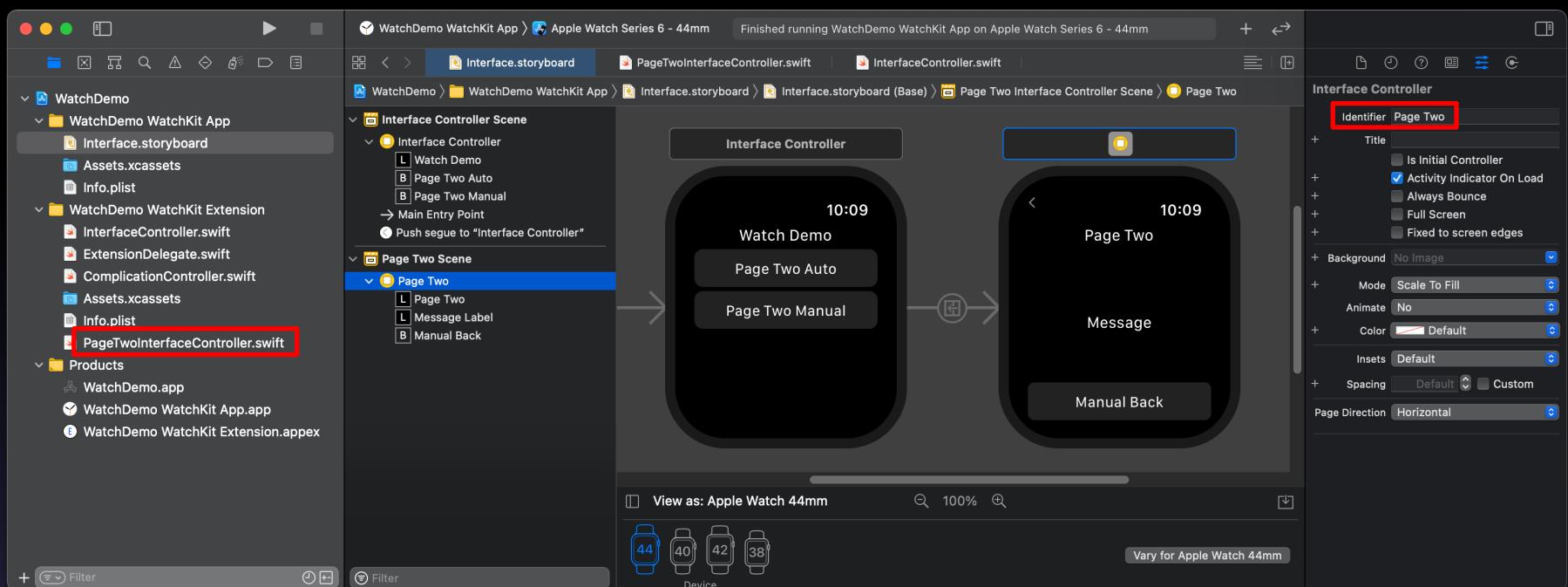
Interface Storyboard



Interface Storyboard



Interface Storyboard



Interface Controller

InterfaceController.swift

```
import WatchKit
import Foundation

class InterfaceController: WKInterfaceController {

    @IBAction func pageTwoManualTapped() {
        pushController(withName: "Page Two", context: "Hello Manually")
    }

    override func contextForSegue(withIdentifier segueIdentifier: String)
        -> Any? {
        if segueIdentifier == "toPageTwoAuto" {
            return "Hello Automatically"
        }
        return nil
    }
}
```

Return value of `contextForSegue` passed to destination's `awake` method.

Page Two Interface Controller

PageTwoInterfaceController.swift

```
import WatchKit
import Foundation

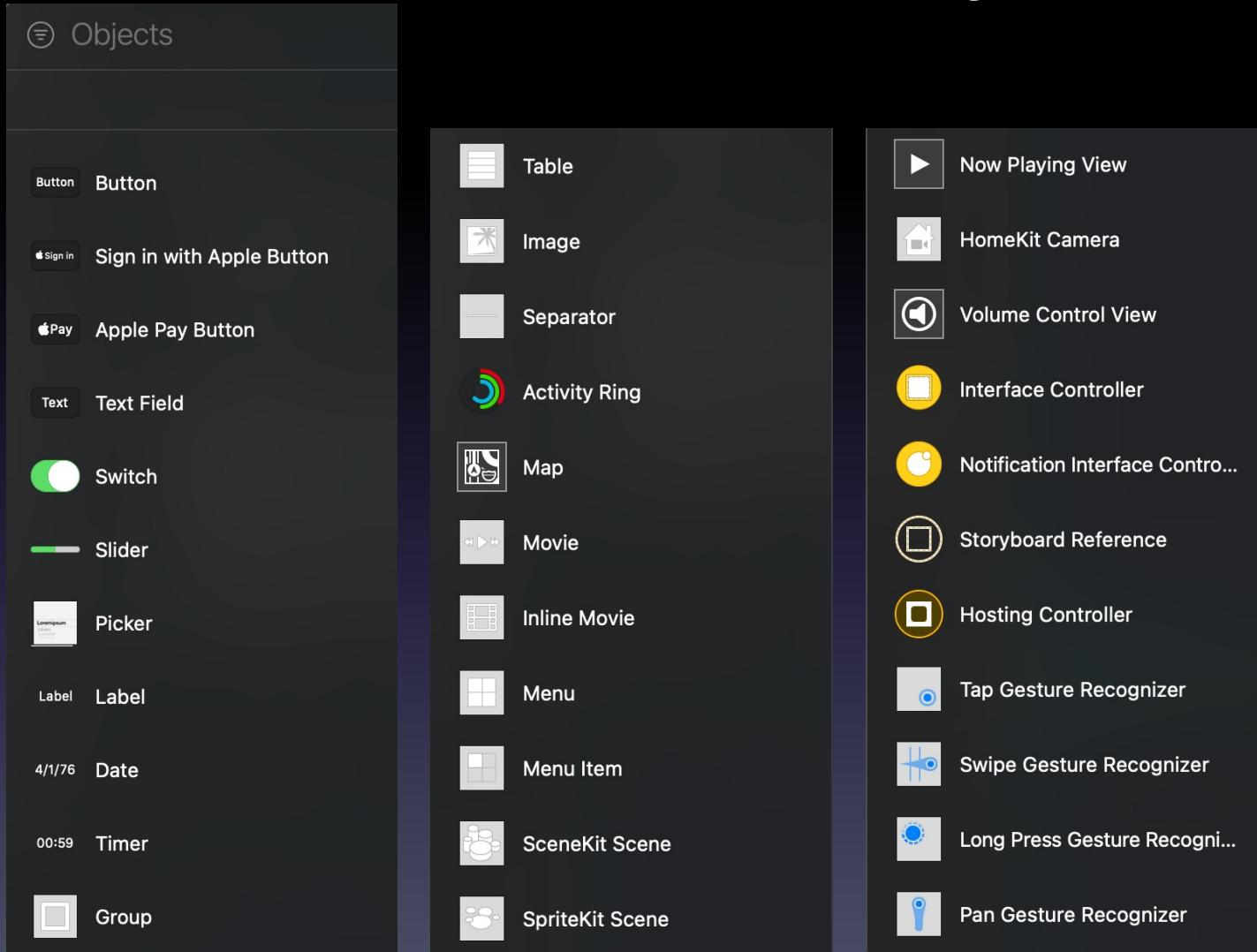
class PageTwoInterfaceController: WKInterfaceController {

    @IBOutlet weak var messageLabel: WKInterfaceLabel!

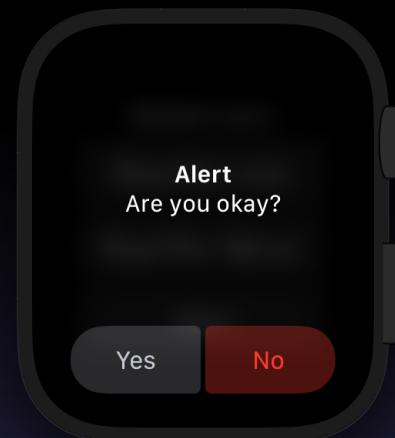
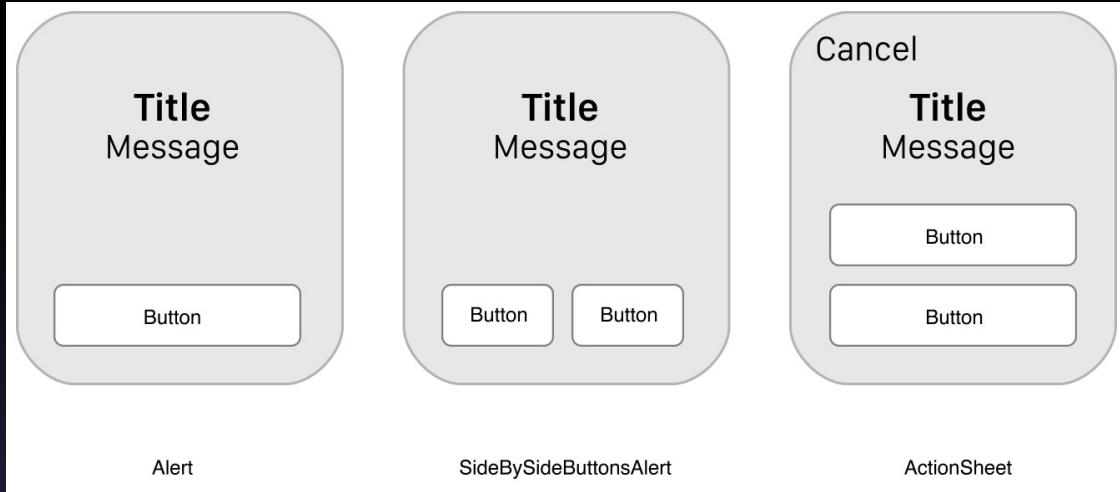
    @IBAction func manualBackTapped() {
        pop()
    }

    override func awake(withContext context: Any?) {
        super.awake(withContext: context)
        // Configure interface objects here.
        if let message = context as? String {
            messageLabel.setText(message)
        }
    }
}
```

Other Interface Objects

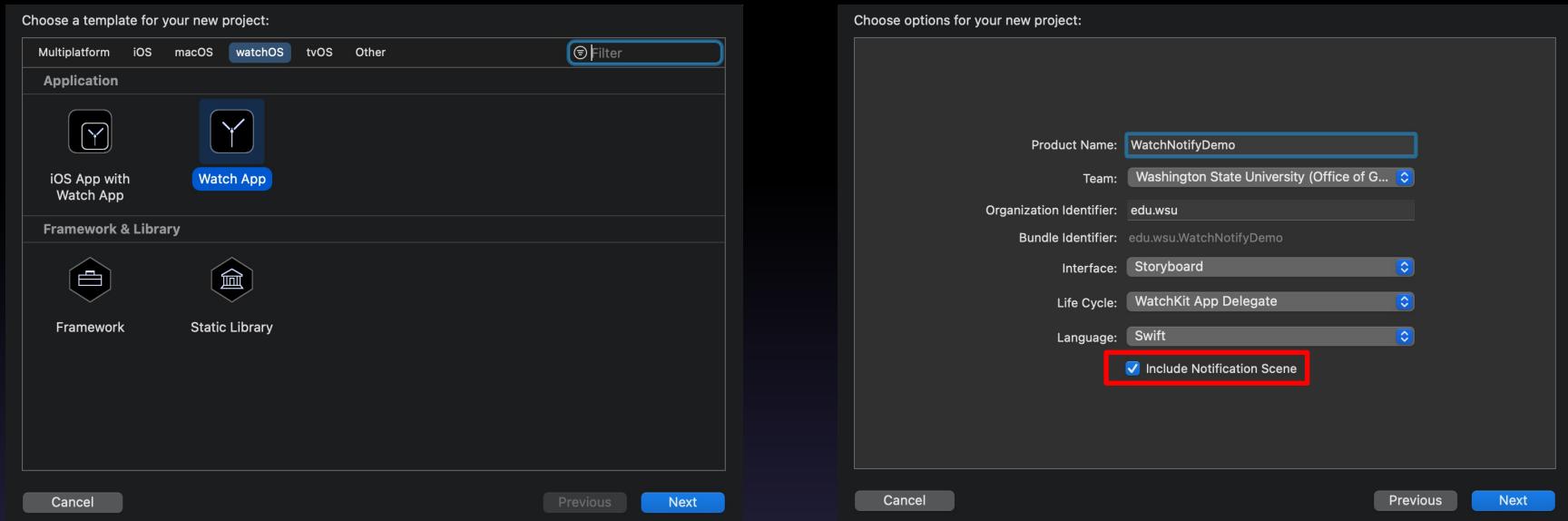


Alerts and Action Sheets



```
@IBAction func alertTapped() {
    let action1 = WKAlertAction(title: "Yes",
                                style: .default, handler: {print("Yes")})
    let action2 = WKAlertAction(title: "No",
                                style: .destructive, handler: {print("No")})
    presentAlert(withTitle: "Alert", message: "Are you okay?",
                 preferredStyle: .sideBySideButtonsAlert,
                 actions: [action1,action2])
}
```

Notifications



The image shows the Xcode IDE with the "NotificationController.swift" file open in the editor.

Project Structure: The sidebar shows the project structure: WatchNotifyDemo (WatchKit App) containing WatchNotifyDemo WatchKit App (Interface.storyboard, Assets.xcassets, Info.plist) and WatchNotifyDemo WatchKit Extension (InterfaceController.swift, ExtensionDelegate.swift, NotificationController.swift, ComplicationController.swift, Assets.xcassets, Info.plist, PushNotificationPayload.apns). The "Products" folder is also listed.

Code: The "NotificationController.swift" file contains the following Swift code:

```
7
8 import WatchKit
9 import Foundation
10 import UserNotifications
11
12 class NotificationController: WKUserNotificationInterfaceController {
13     override init() {
14         // Initialize variables here.
15         super.init()
16
17         // Configure interface objects here.
18     }
19 }
```

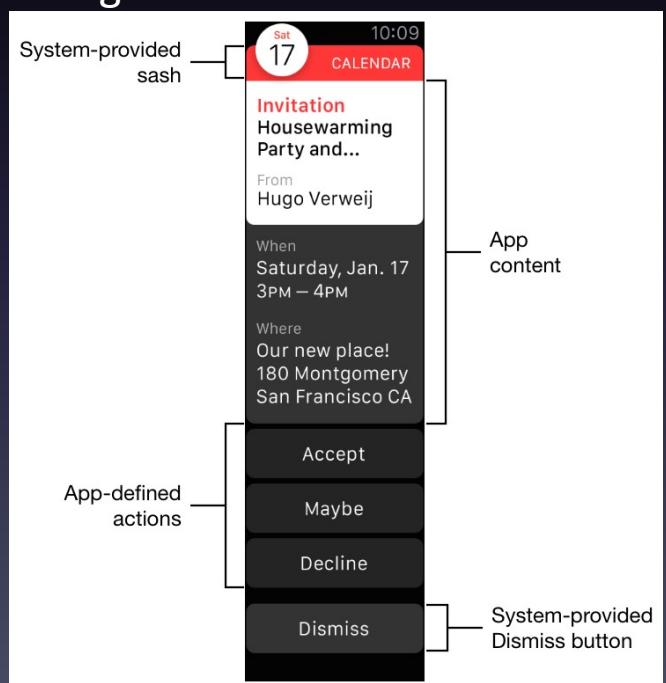
Notifications

- watchOS uses same technique as iOS
 - Call `requestAuthorization` on watch
 - Handle notifications using `didReceive` and `willPresent`
 - Schedule `UNNotificationRequest` using `UNUserNotificationCenter.add`
 - Remote and background notifications can be sent directly to watch
- Watch first displays Short Look, then Long Look

Short Look



Long Look



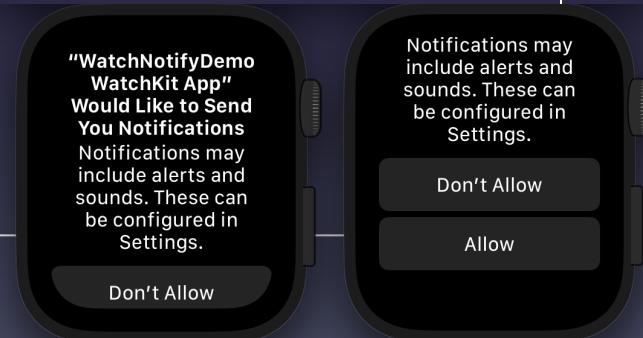
Authorize Notifications

ExtensionDelegate.swift

```
import WatchKit
import UserNotifications

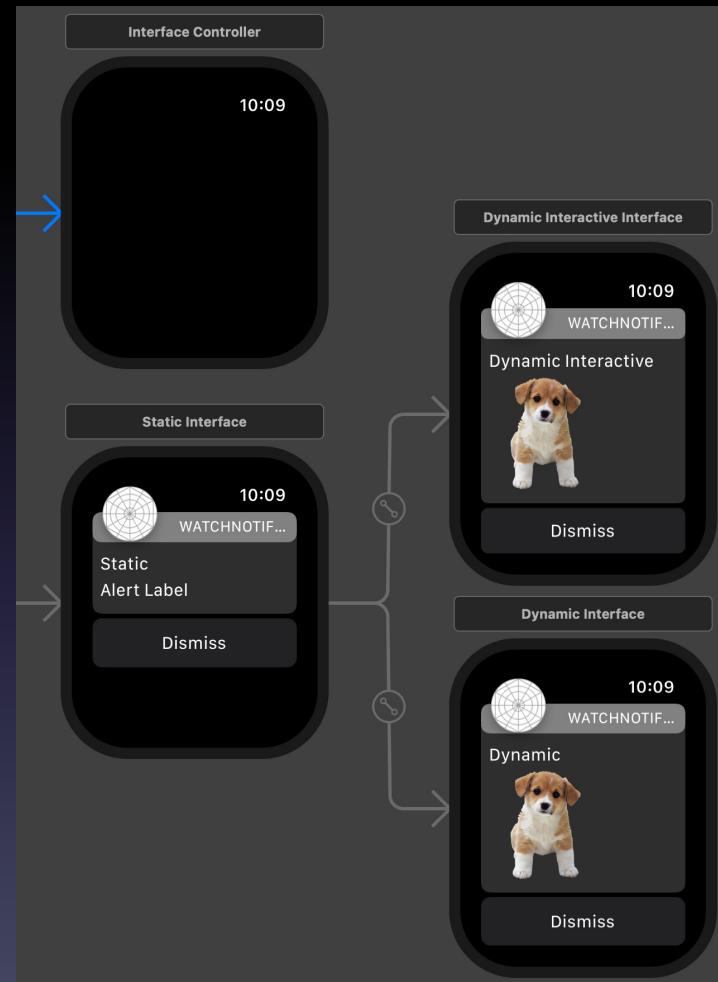
class ExtensionDelegate: NSObject, WKExtensionDelegate {

    func applicationDidFinishLaunching() {
        // Perform any final initialization of your application.
        let center = UNUserNotificationCenter.current()
        center.requestAuthorization(options: [.alert])
        { (granted, error) in
            if granted {
                print("notifications allowed")
            } else {
                print("notifications not allowed")
            }
        }
    }
}
```

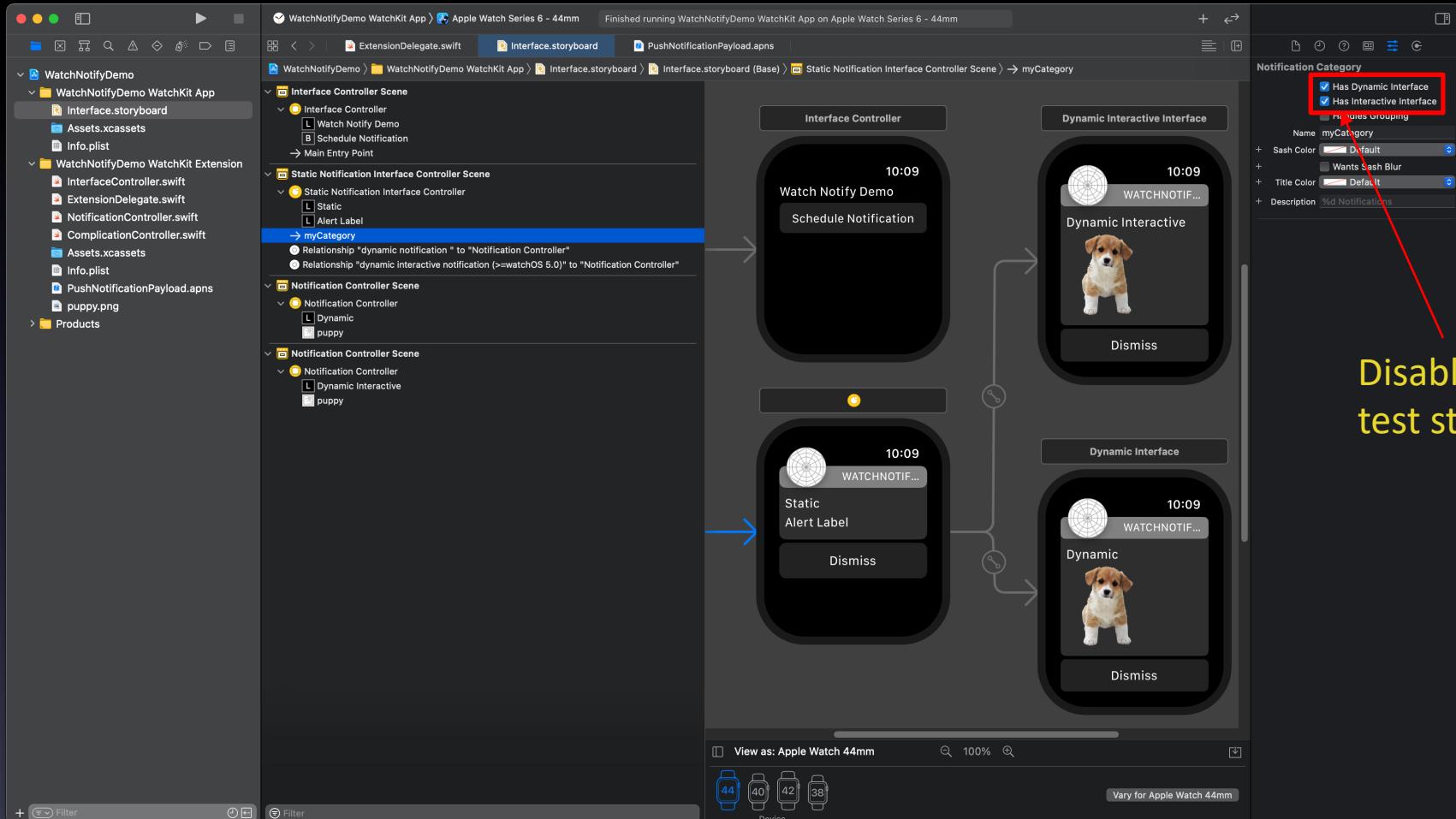


Notification Interfaces

- Static Interface
 - Simple elements (fast)
- Dynamic Interface (watchOS < 5.0)
 - Outlets in NotificationController
 - No interactive elements
 - If takes too long, uses Static
- Dynamic Interactive Interface
 - Allows interactive elements
 - Outlets/Actions in NotificationController
 - If takes too long, uses Static



Notification Interfaces



Notification Interfaces

```
import WatchKit
import Foundation
import UserNotifications

class NotificationController: WKUserNotificationInterfaceController {

    @IBOutlet weak var interactiveImage: WKInterfaceImage!
    @IBOutlet weak var dynamicImage: WKInterfaceImage!

    override init() {
        // Initialize variables here.
        super.init()

        // Configure interface objects here.
        if interactiveImage != nil {
            interactiveImage.setImage(UIImage(named: "puppy.png"))
        }
        if dynamicImage != nil {
            dynamicImage.setImage(UIImage(named: "puppy.png"))
        }
    }
}
```

NotificationController.swift

Dynamic interface elements must be initialized in `WKUserNotificationInterfaceController` class

Simulate Push Notification

The screenshot shows the Xcode interface for a WatchKit app named "WatchNotifyDemo".

Left Panel: Shows the project structure and a code editor displaying a JSON payload file (`PushNotificationPayload.apns`). The payload includes an alert message, a category, and a thread ID, along with a section for "WatchKit Simulator Actions". A note at the bottom explains how to use multiple JSON files for different interface types.

Middle Panel: Shows the Xcode scheme editor with three targets listed: "WatchNotifyDemo WatchKit App", "WatchNotifyDemo WatchKit App (Notification)", and "WatchNotifyDemo WatchKit App (Complication)". The "Edit Scheme..." button is highlighted.

Bottom Panel: Shows the "Run" button in the toolbar highlighted, and the "Info" tab of the build settings. The "Watch Interface" dropdown is set to "Dynamic Notification", which is highlighted with a red box.

Simulator Preview: Shows four Apple Watch Series 6 devices. From left to right:

- Optional title:** Shows a circular radar icon.
- Short Look:** Shows a static card with the title "WATCHNOTIFYD..." and subtitle "Static Test message".
- Long Look:** Shows a dynamic card with the title "WATCHNOTIFYD..." and subtitle "Dynamic Test message".
- Dynamic Interactive:** Shows a dynamic card with the title "WATCHNOTIFYD..." and subtitle "Dynamic Interactive".

Short Look

.....Long Look.....

Schedule Notification Programmatically

```
import UserNotifications                                         InterfaceController.swift

func scheduleNotification() { // Same as in Notifications lecture notes
    let content = UNMutableNotificationContent()
    content.title = "Hey!"
    content.body = "What's up?"
    content.userInfo["message"] = "Yo!"
    // If Notification Category set in Storyboard, then set here too
    content.categoryIdentifier = "myCategory"

    // Configure trigger for 5 seconds from now
    let trigger = UNTimeIntervalNotificationTrigger(timeInterval: 5.0, repeats: false)
    // Create request
    let request = UNNotificationRequest(identifier: "NowPlusFive",
                                         content: content, trigger: trigger)
    // Schedule request
    let center = UNUserNotificationCenter.current()
    center.add(request, completionHandler: { (error) in
        if let err = error {
            print(err.localizedDescription)
        }
    })
}
```

Handle Notifications

```
import WatchKit  
import Foundation  
import UserNotifications  
  
class NotificationController:  
    WKUserNotificationInterfaceController {  
  
    @IBOutlet var interactiveLabel: WKInterfaceLabel!  
  
    override func didReceive(_ notification: UNNotification) {  
        let title = notification.request.content.title  
        interactiveLabel.setText(title)  
    }  
}
```



Complications

- ClockKit framework
- CLKComplicationDataSource
- Provide data for a specific date/time
- See different styles at
developer.apple.com/design/human-interface-guidelines/watchos/overview/complications



Required Delegate Method

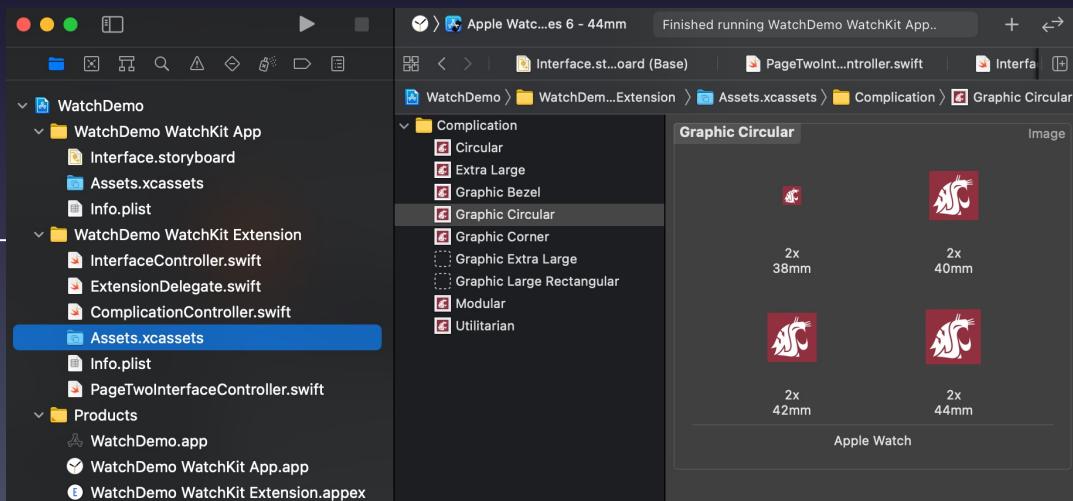
- In ComplicationController.swift
 - `getCurrentTimelineEntry(complication, handler)`
 - Create and pass time line entry to handler

getCurrentTimeLineEntry

- For desired complication families (.graphicCircular, etc.)
 - Create provider for template, e.g.,
 - CLKFullColorImageProvider(UIImage)
 - CLKSimpleTextProvider(String)
 - Create a CLKComplicationTemplate, e.g.,
 - CLKComplicationTemplateGraphicCircularImage(provider)
 - CLKComplicationTemplateCircularSmallSimpleText(provider)
 - Create time line entry for template at date
 - CLKComplicationTimelineEntry(Date, CLKComplicationTemplate)
 - Send entry to handler

Complications

```
func getCurrentTimelineEntry(for complication: CLKComplication,  
    withHandler handler: @escaping (CLKComplicationTimelineEntry?) -> Void) {  
if (complication.family == .graphicCircular) {  
    // Construct template with image, open gauge, and text  
    let image = UIImage(named: "Complication/Graphic Circular")  
    let provider = CLKFullColorImageProvider(fullColorImage: image!)  
    let template =  
        CLKComplicationTemplateGraphicCircularImage(imageProvider: provider)  
    // Create the timeline entry  
    let entry = CLKComplicationTimelineEntry(date: Date(),  
        complicationTemplate: template)  
    handler(entry)  
    return  
}  
handler(nil)  
}
```



Complications: Testing



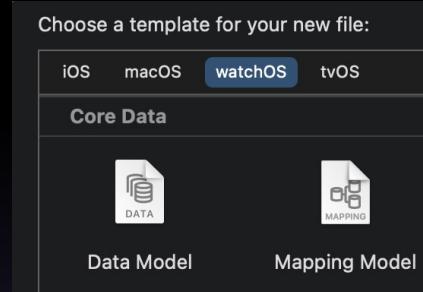
Sensors

- **CoreMotion** framework
 - Accelerometer
 - Gyroscope
- **CoreLocation** framework
 - GPS
- **HealthKit** framework
 - Heart rate

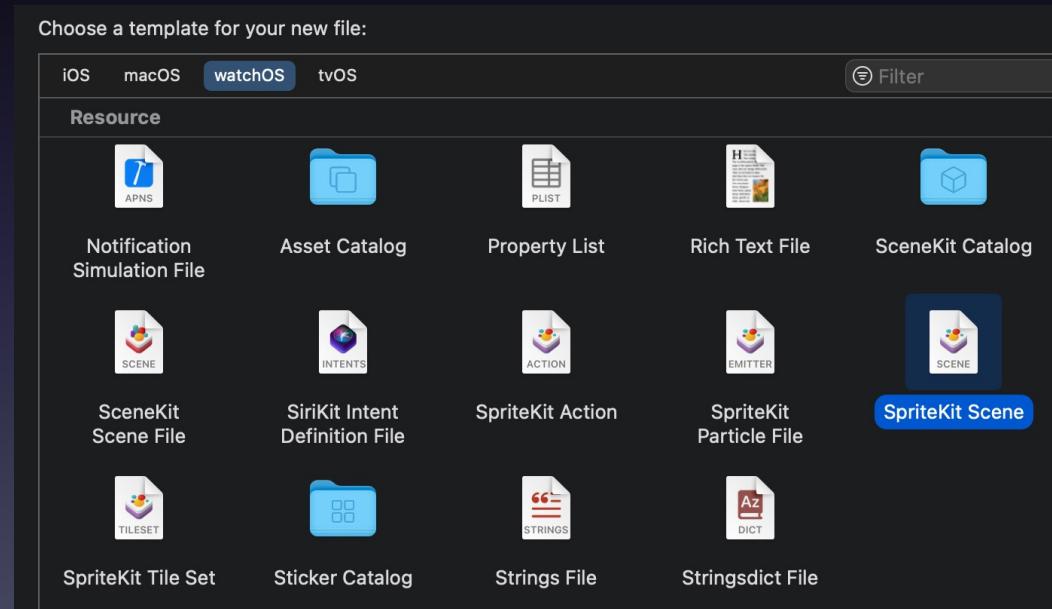


Other Elements

- Core Data



- SpriteKit



Resources

- WatchKit framework
 - developer.apple.com/documentation/watchkit
 - Notifications
 - developer.apple.com/documentation/watchkit/notifications
- ClockKit framework
 - developer.apple.com/documentation/clockkit
 - Complications
 - developer.apple.com/documentation/watchkit/notifications
- HealthKit framework
 - developer.apple.com/documentation/healthkit