Apple Watch

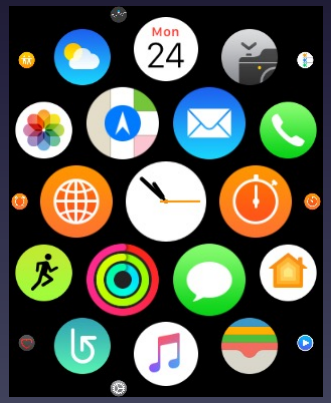
Mobile Application Development in iOS

School of EECS

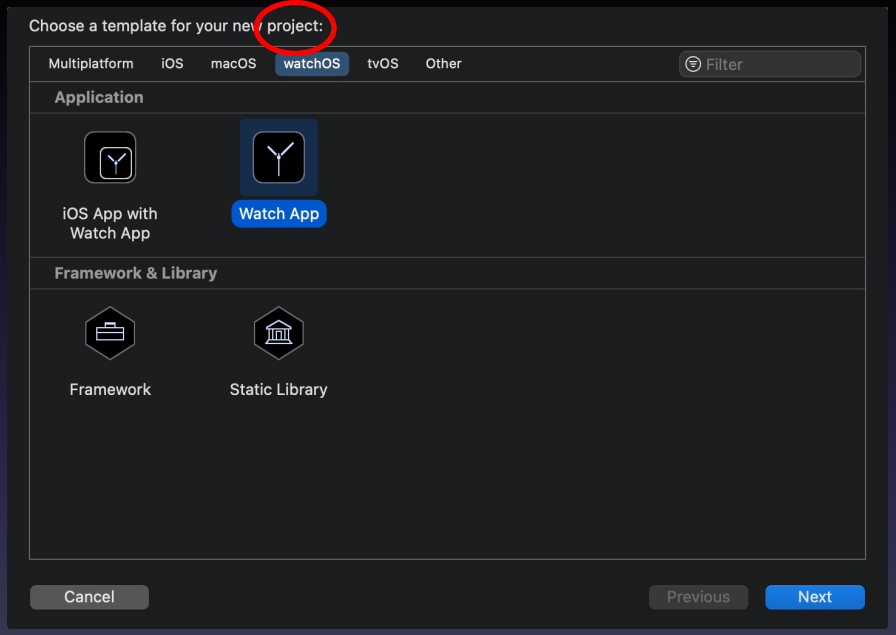
Washington State University

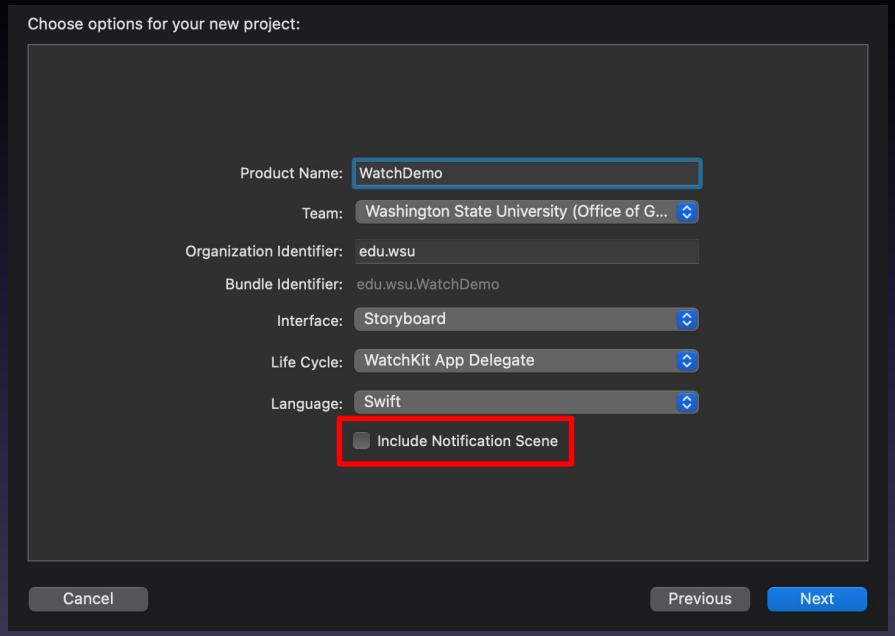
Outline

- Xcode configuration
- WatchKit
- Notifications
- Complications
- Sensors



Xcode Configuration: New Project



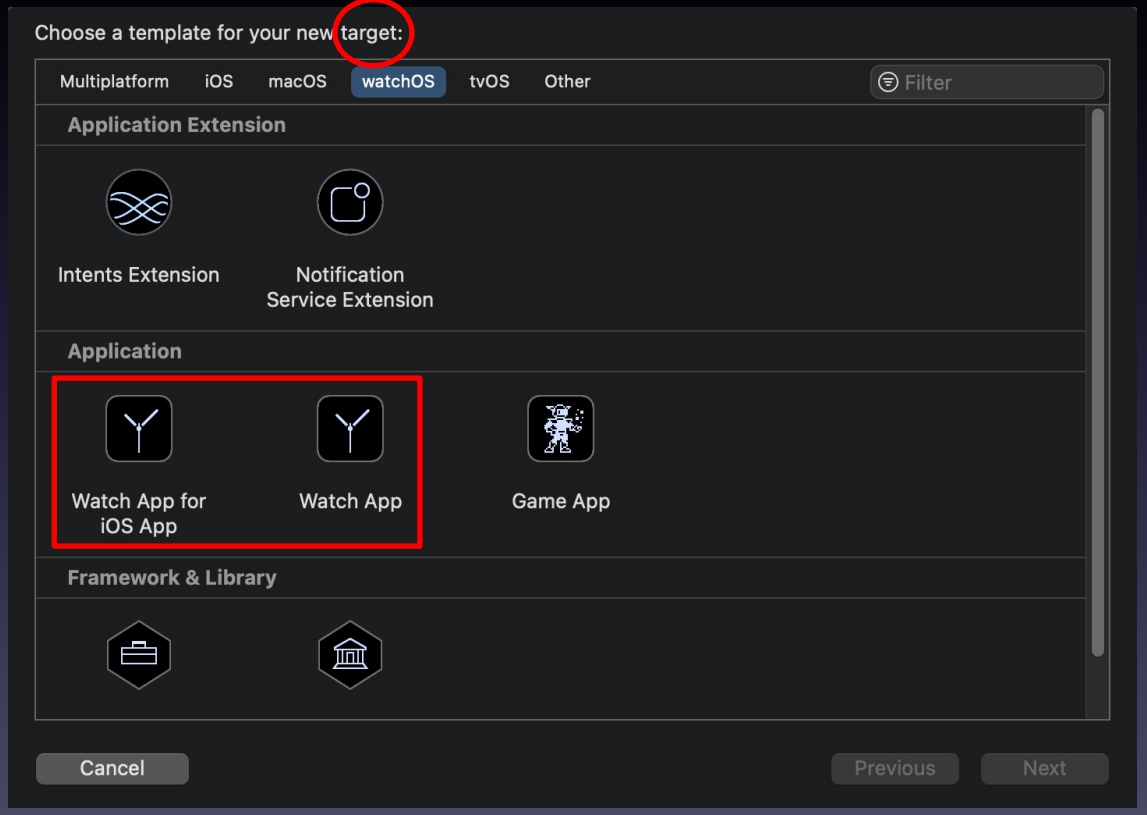


- iOS App with Watch App
- Standalone Watch App

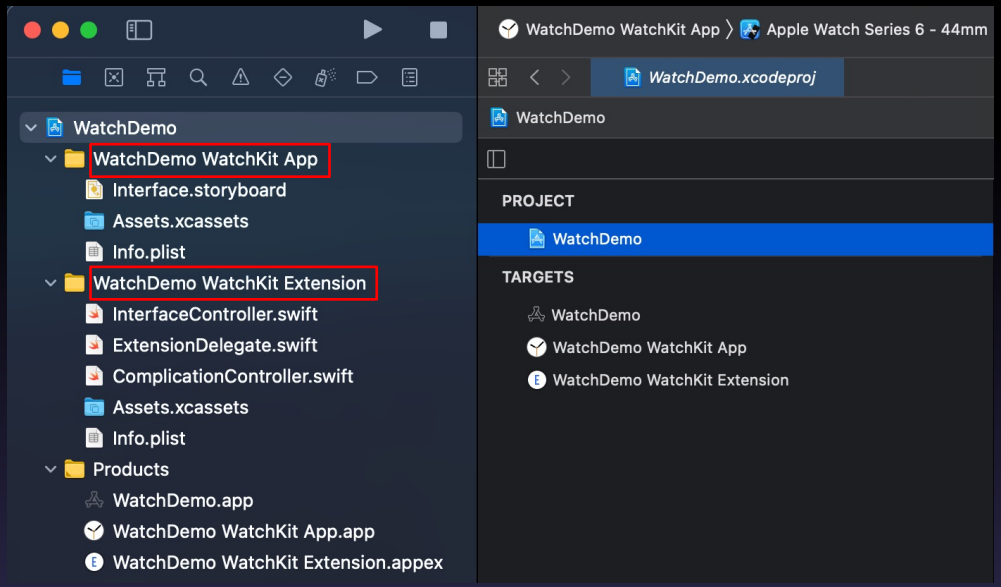
We'll do notifications later.

Xcode Configuration: Add to Existing Project

File → New→ Target

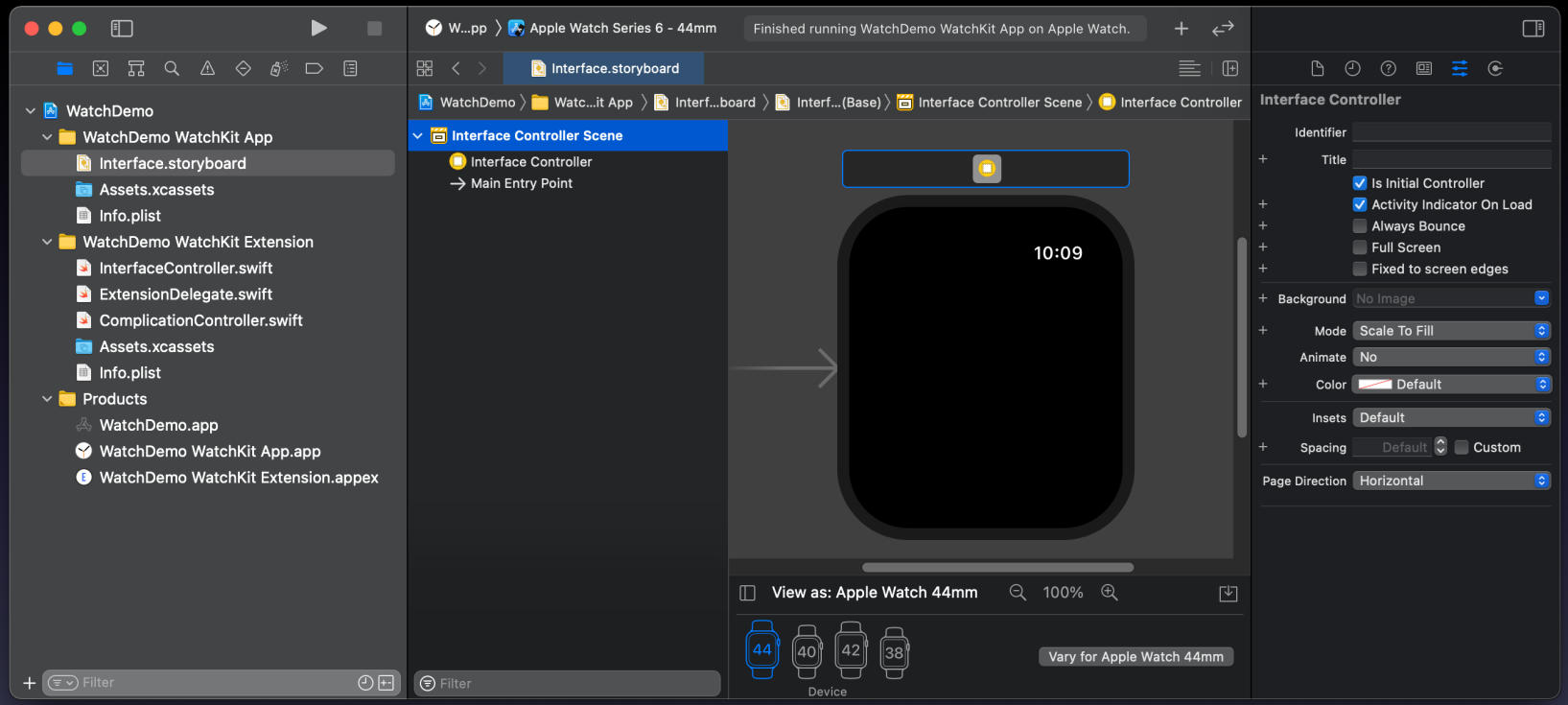


WatchKit App and Extension

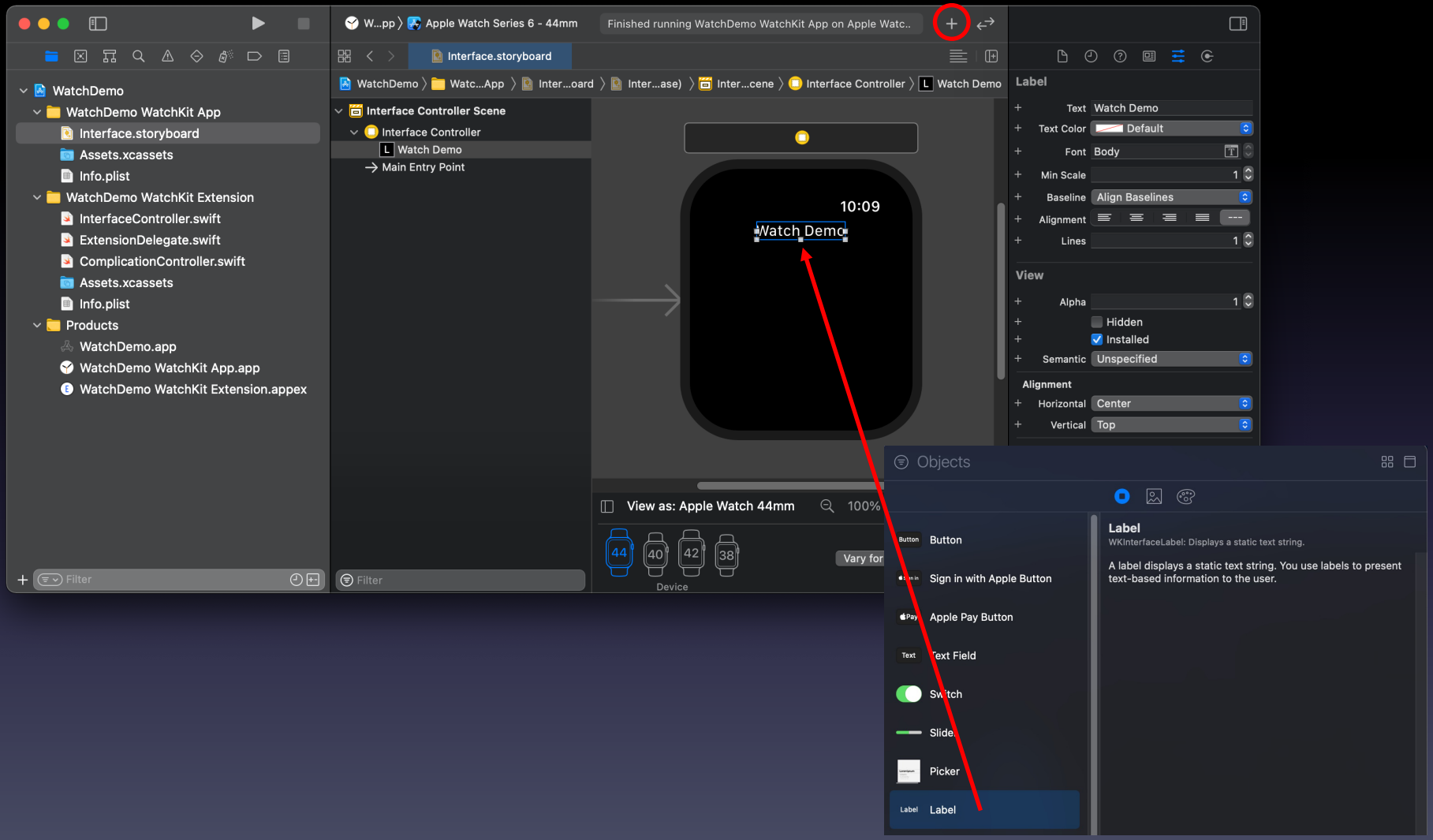


- WatchKit App
  - Storyboard
  - Storyboard assets
- WatchKit Extension
  - App code
- Simulator
  - Window → Devices and Simulators
  - "Pair" a watch

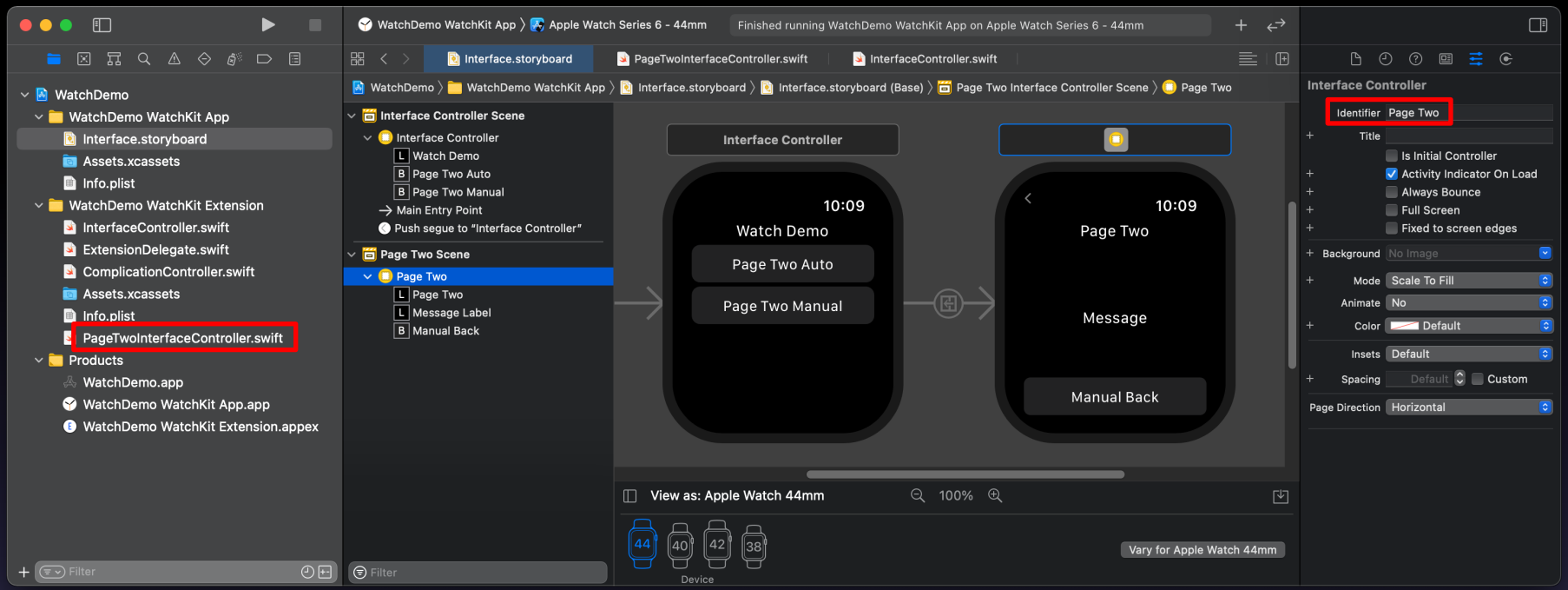
Interface Storyboard

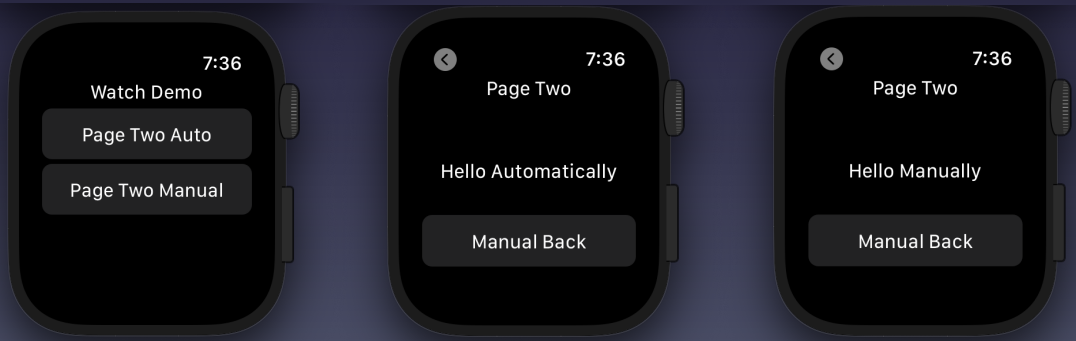


Interface Storyboard



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
    }
}
```

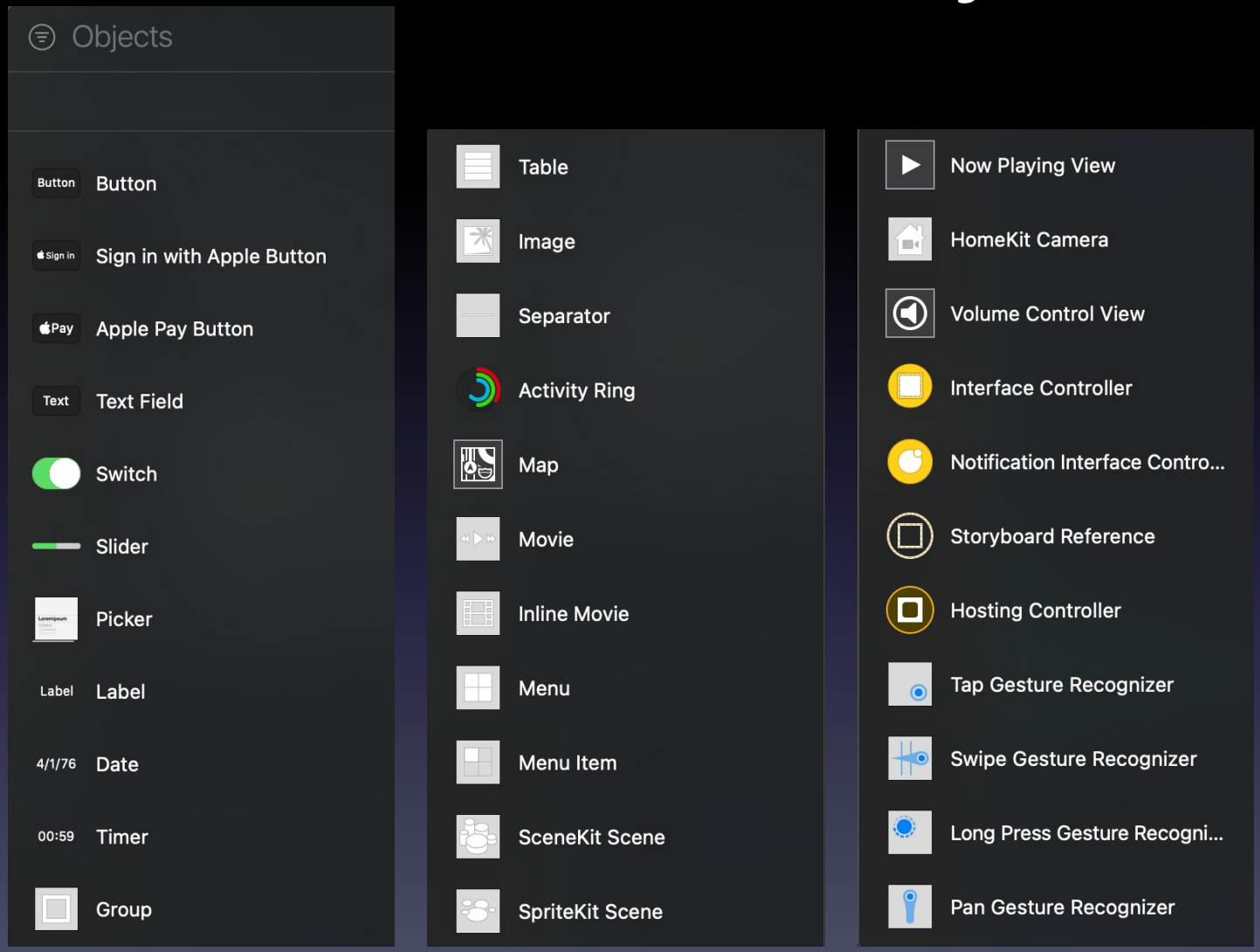
Return value of contextForSegue passed to destination's awake method.

Page Two Interface Controller

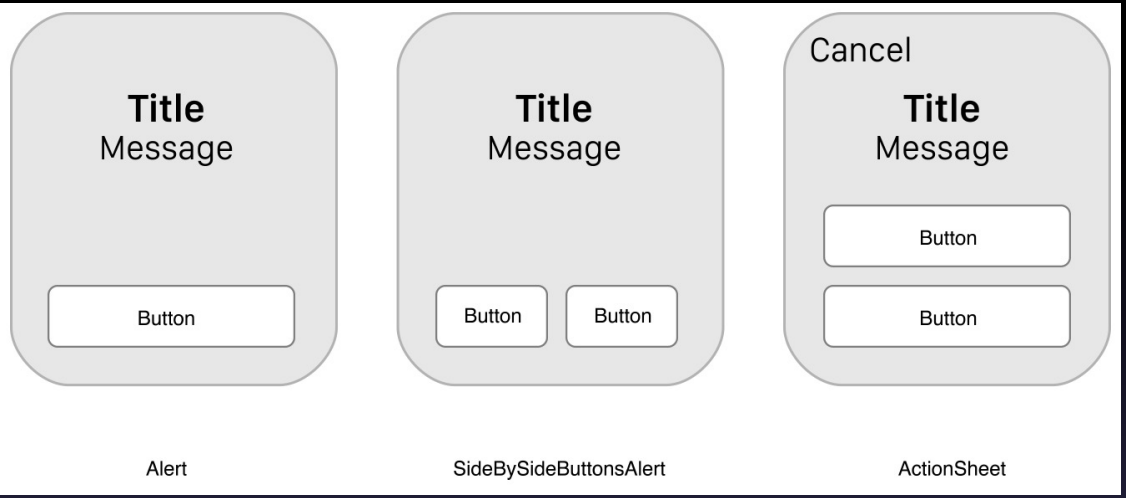
PageTwoInterfaceController.swift

```
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

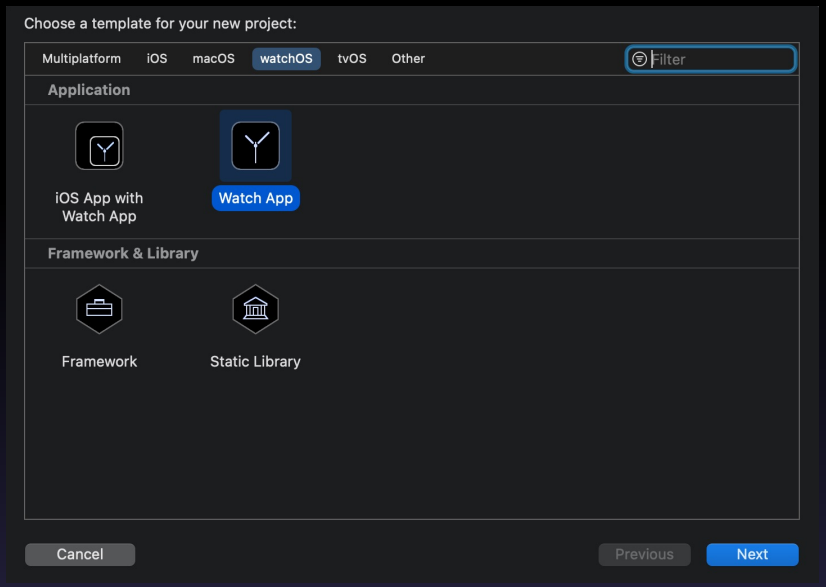


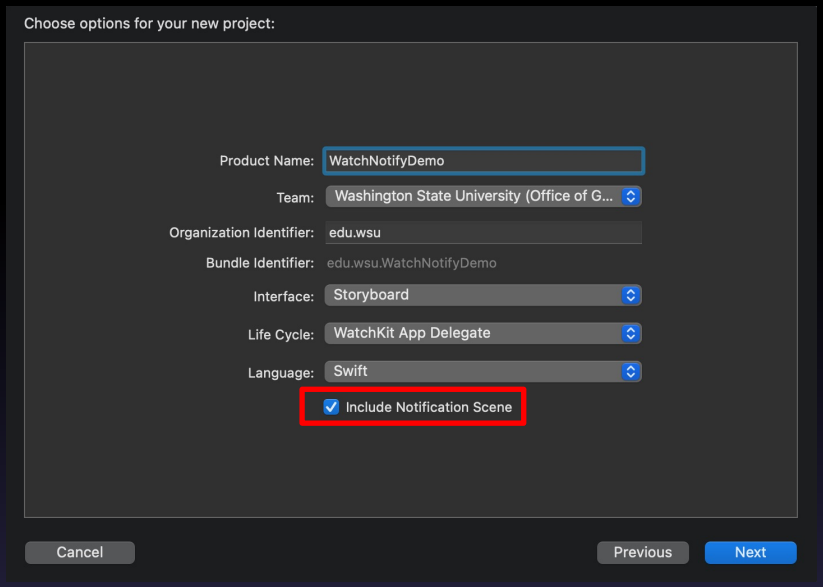
Alerts and Action Sheets

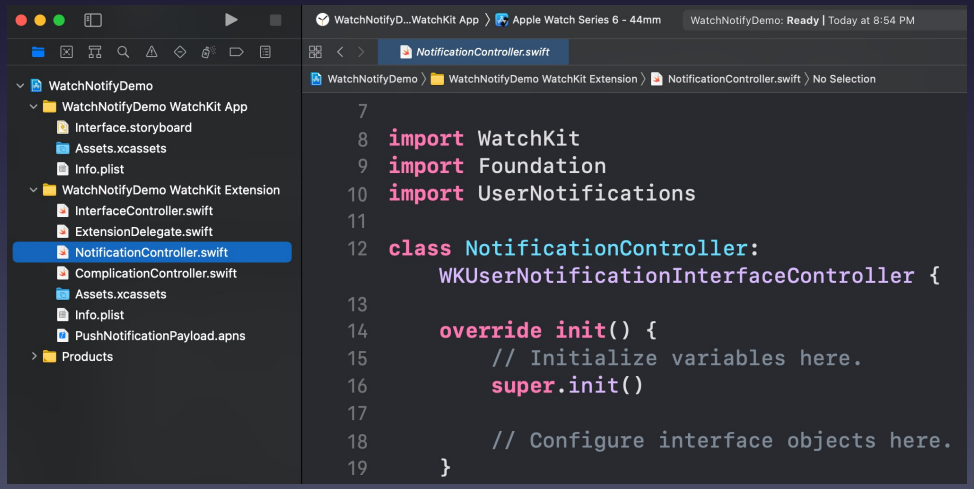



```
@IBAction func alertTapped() {
    let action1 = WKAlertAction(title: "Yes",
                        style: .default, handler: {print("Yes")})
    let action2 = WKAlertAction(title: "No",
                        style: .destructive, handler: {print("No")})
    presentAlert(withTitle: "Alert", message: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

Notifications





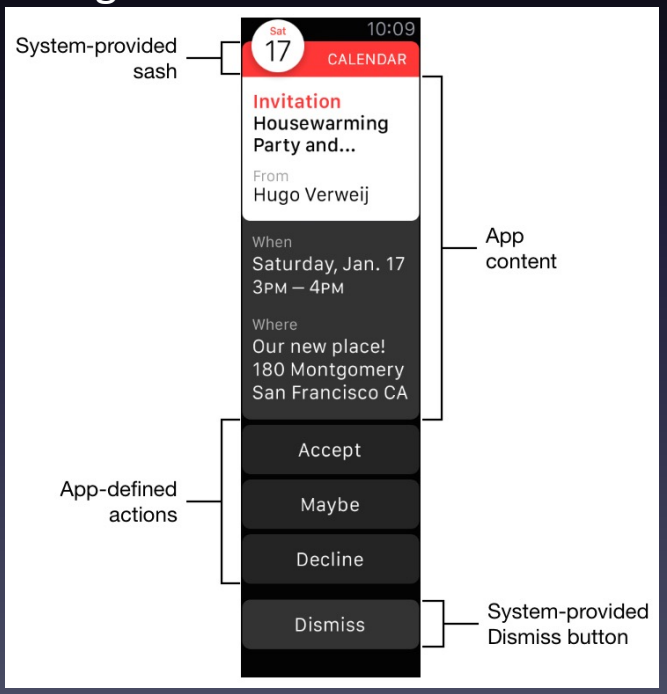


Notifications

- 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

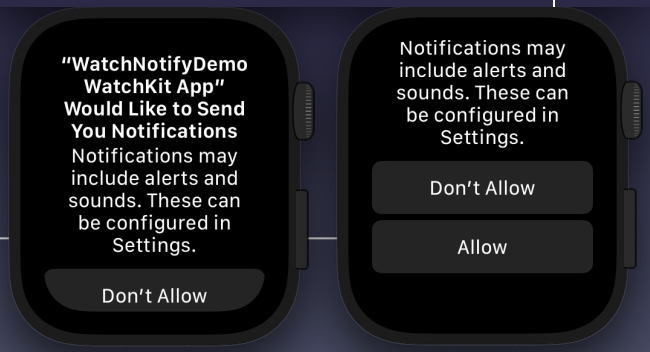
Long Look



Authorize Notifications

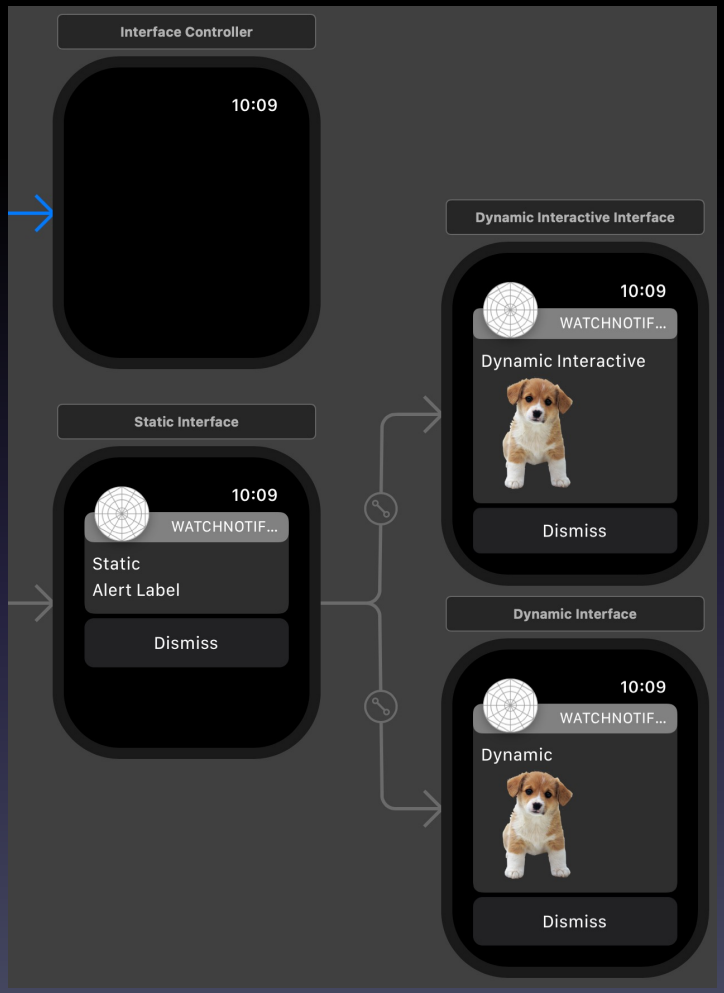
ExtensionDelegate.swift

```
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")
            }
        }
    }
}
```

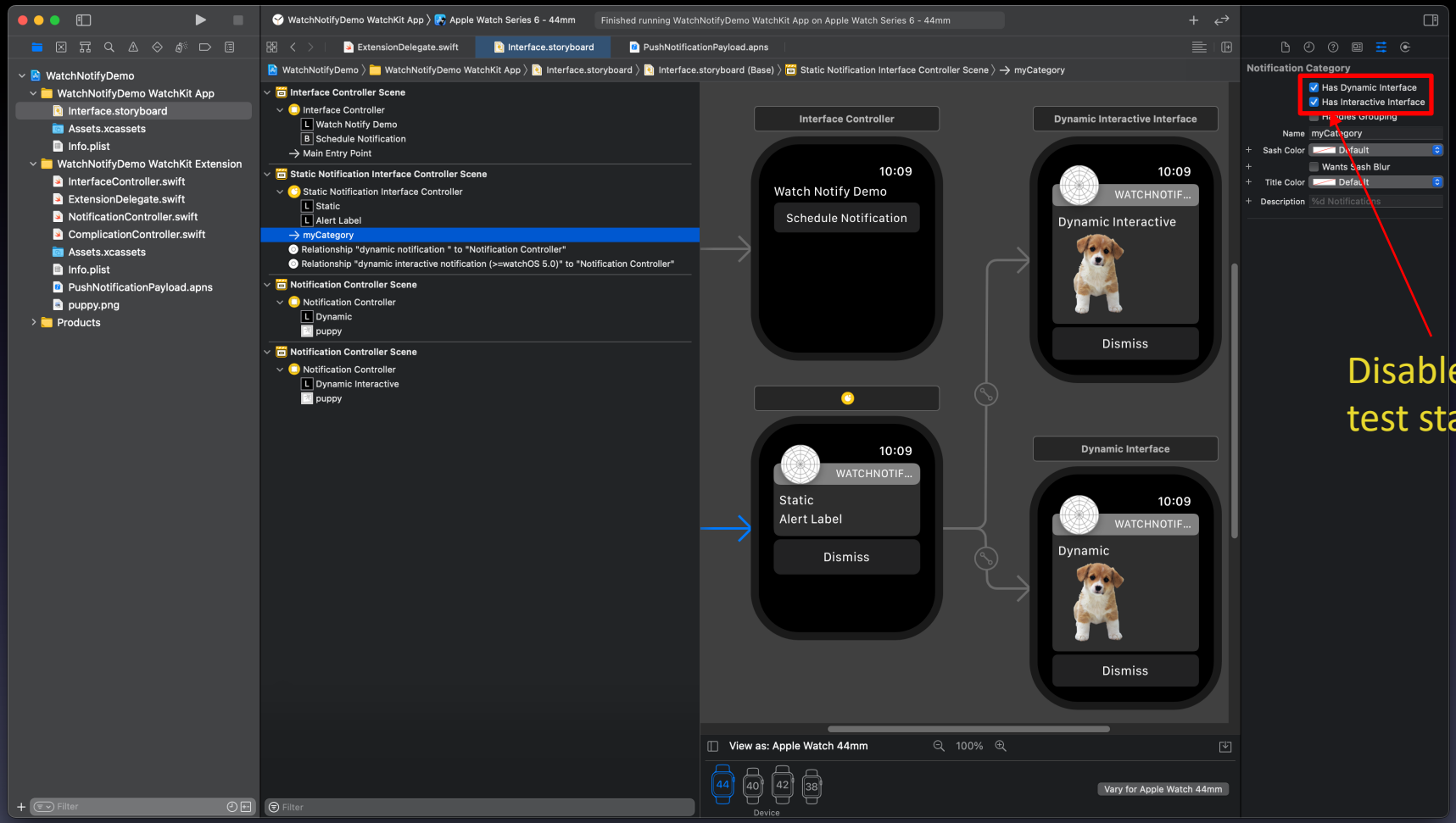


Notification Interfaces

- 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



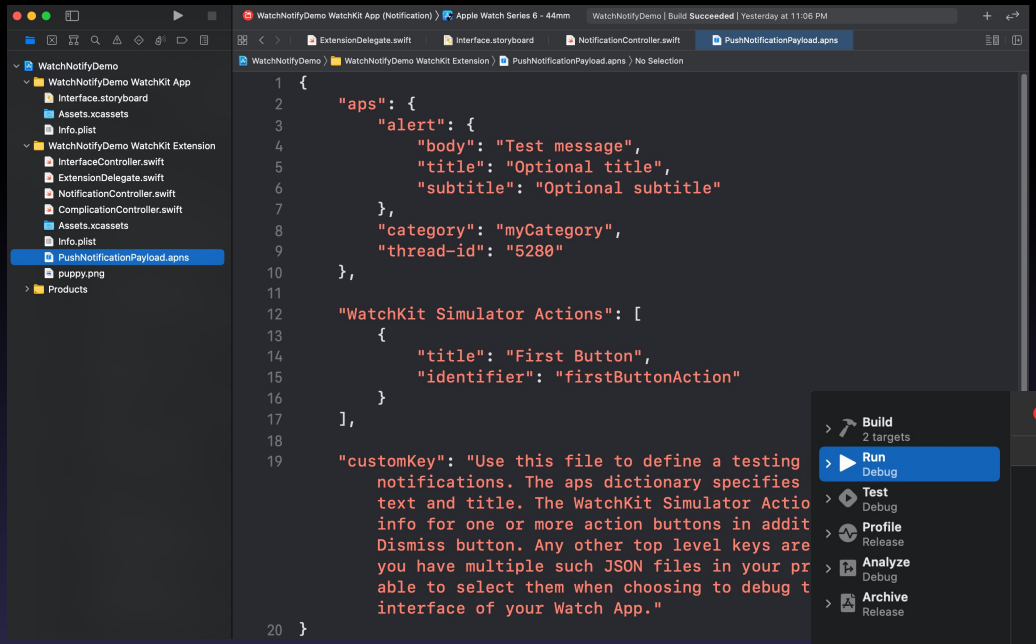
Notification Interfaces

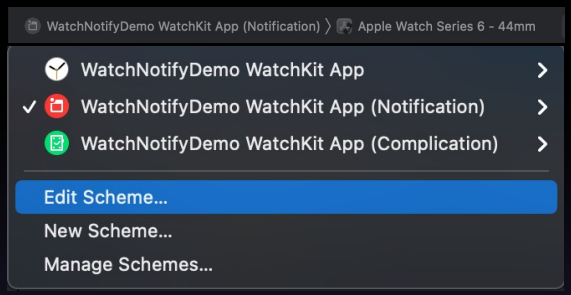
NotificationController.swift

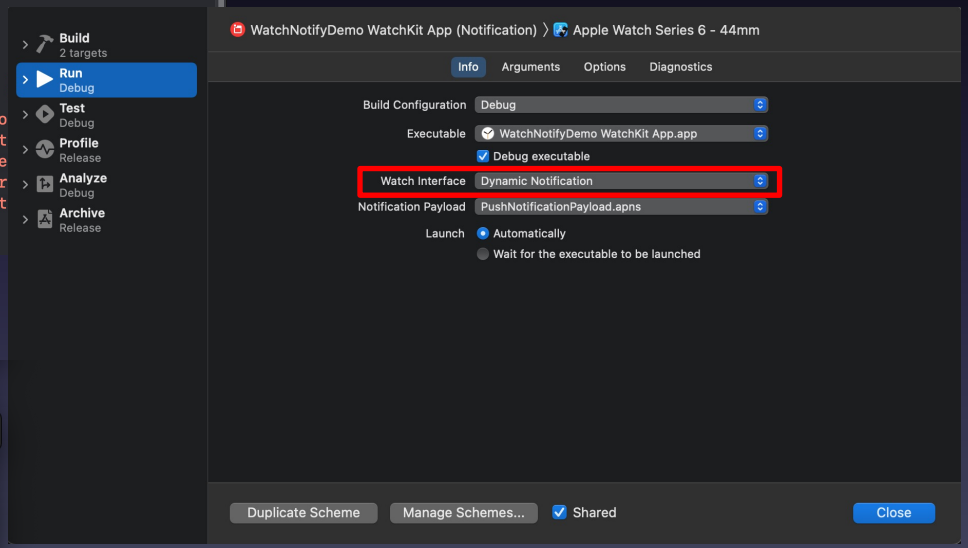
```
import WatchKit
import Foundation
import UserNotifications

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"))
        }
    }
}
```

Dynamic interface elements must be initialized in WKUserNotificationInterfaceController class

Simulate Push Notification







Short Look .....Long Look.....

Schedule Notification Programmatically

InterfaceController.swift

```
import UserNotifications

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, withCompletionHandler: { (error) in
          if let err = error {
               print(err.localizedDescription)
          }
     })
}
```

Handle Notifications

NotificationController.swift

```
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

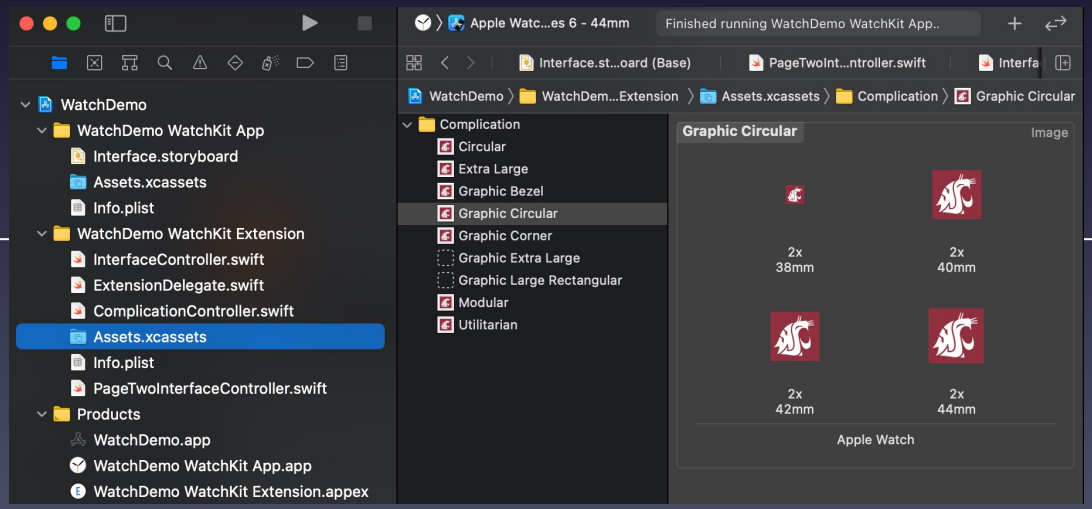
- In ComplicationController.swift
  - getCurrentTimelineEntry(complication, handler)
    - Create and pass time line entry to handler

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)
}
```



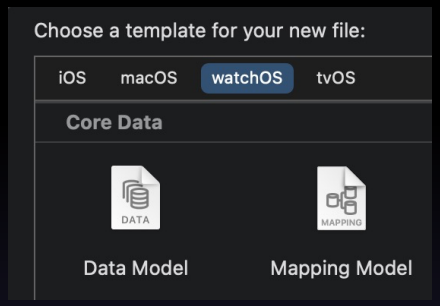
Complications: Testing

Sensors

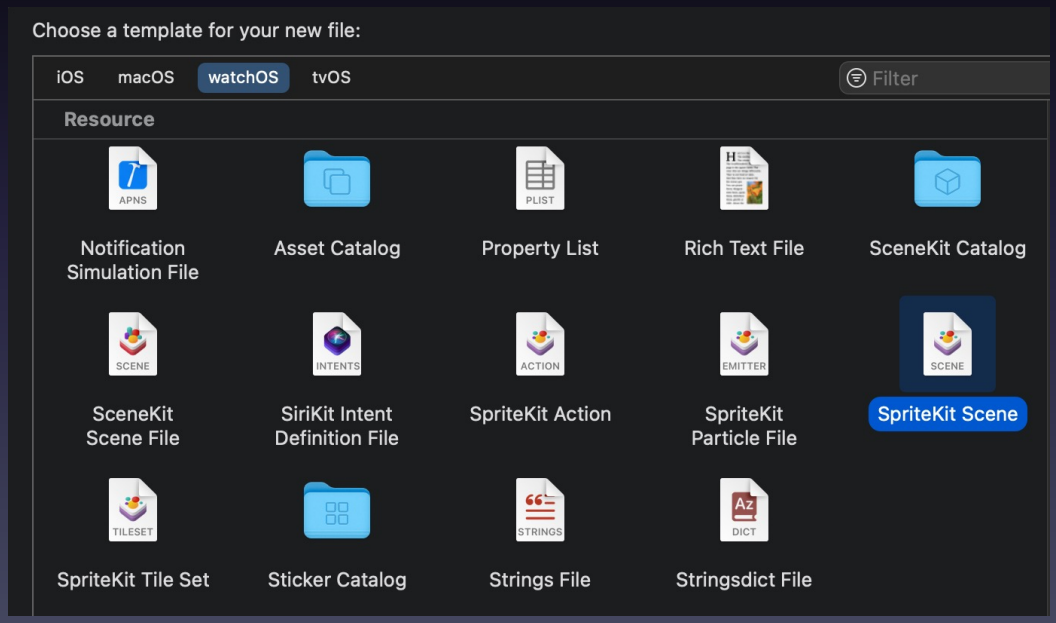
- CoreMotion framework
  - Accelerometer
  - Gyroscope
- CoreLocation framework
  - GPS
- HealthKit framework
  - Heart rate

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