

Hot Topics

Mobile Application Development in iOS

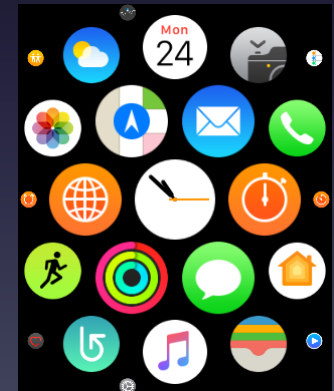
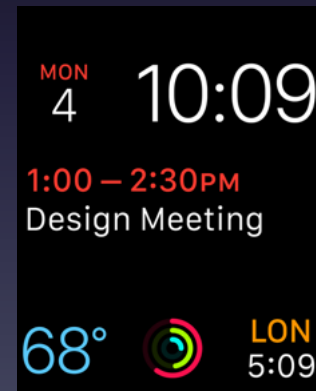
School of EECS

Washington State University

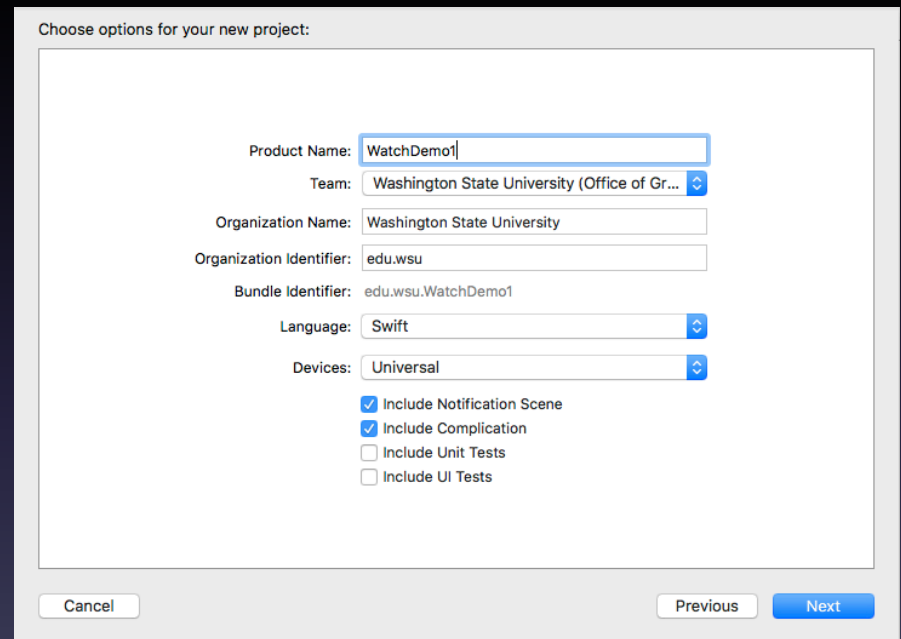
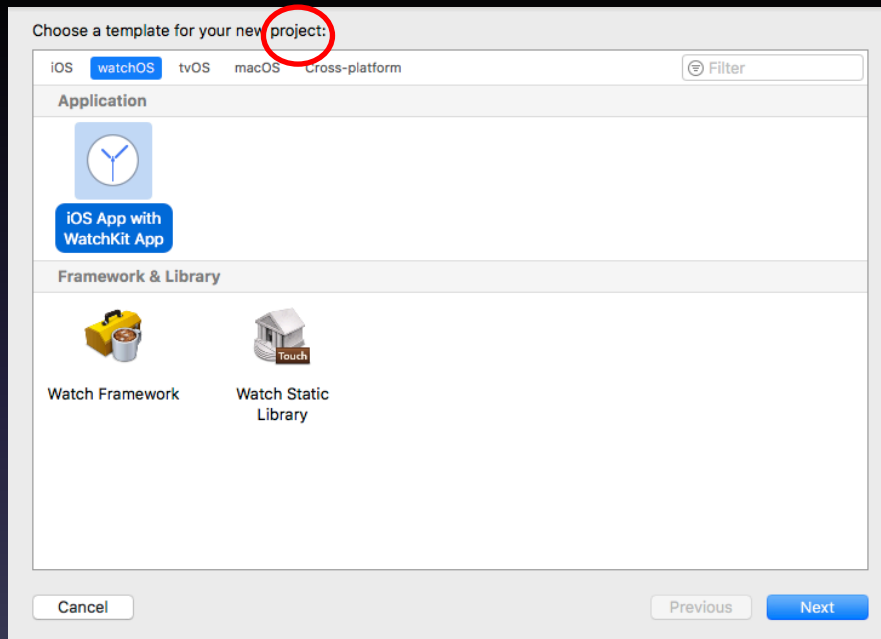
Instructor: Larry Holder

Outline

- Apple Watch
 - Xcode configuration
 - Watch App and WatchKit
 - Complications
 - Notifications
 - Communications
 - Sensors

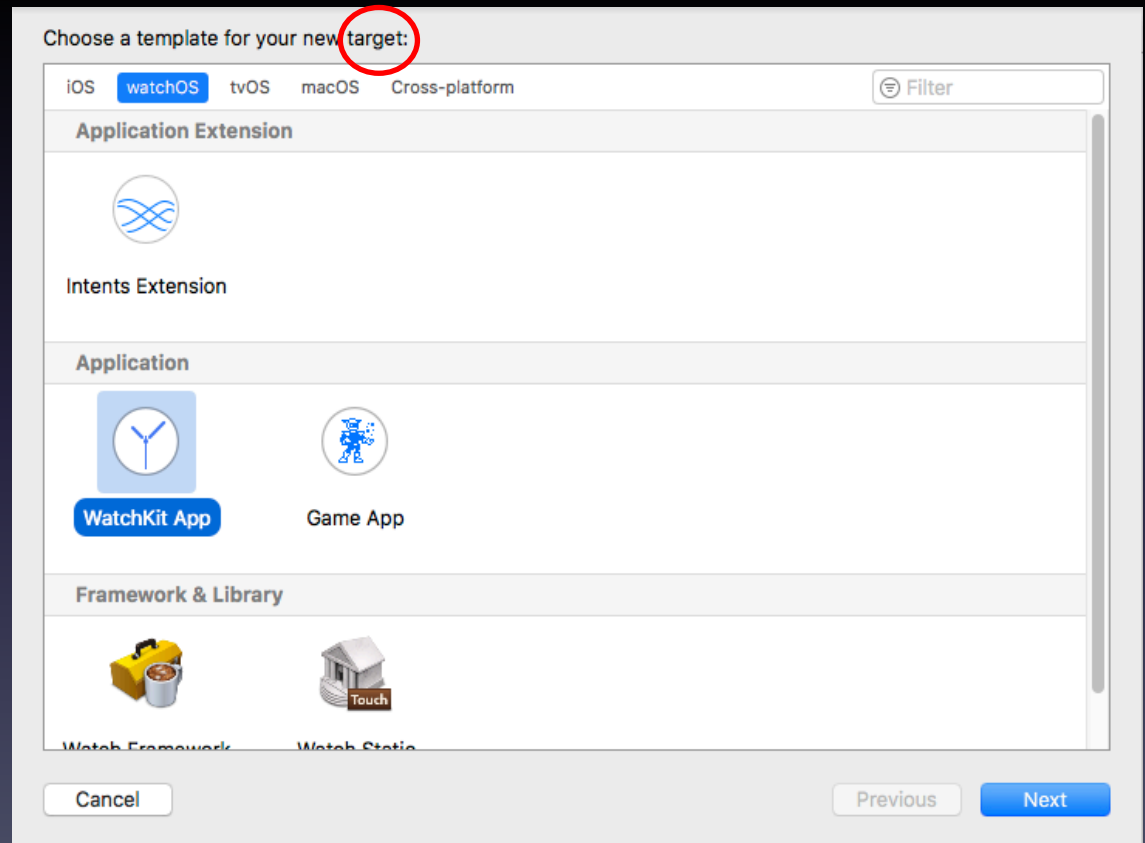


Xcode Configuration: New Project

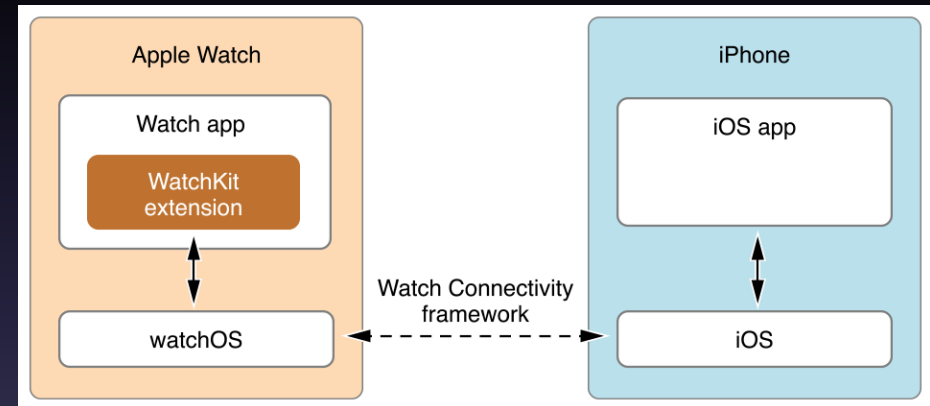
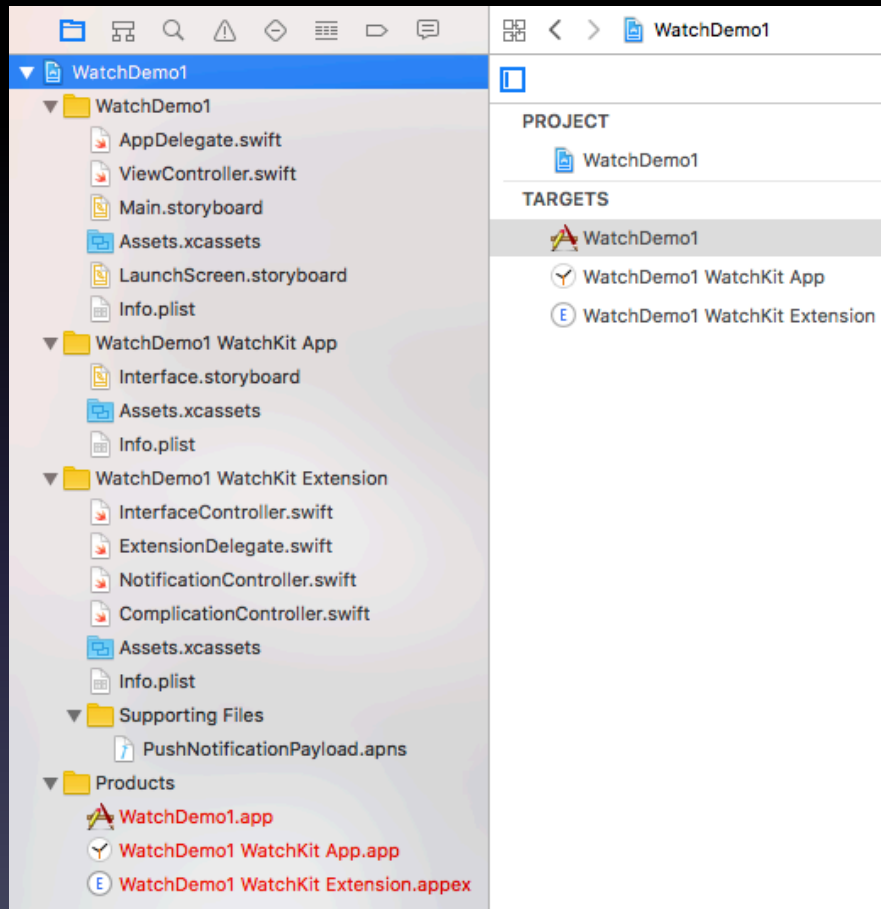


Xcode Configuration: Add to Existing Project

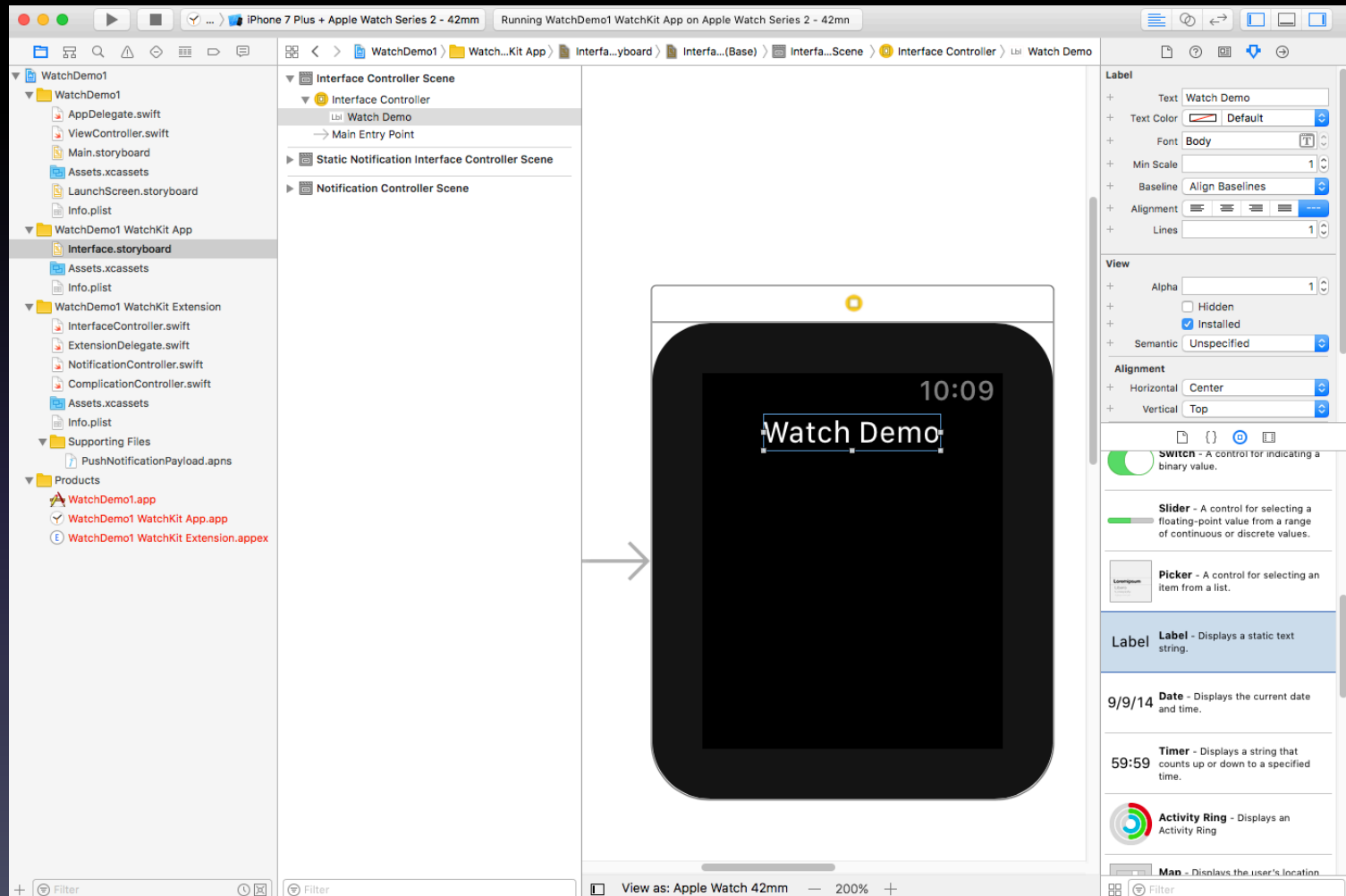
File → New → Target



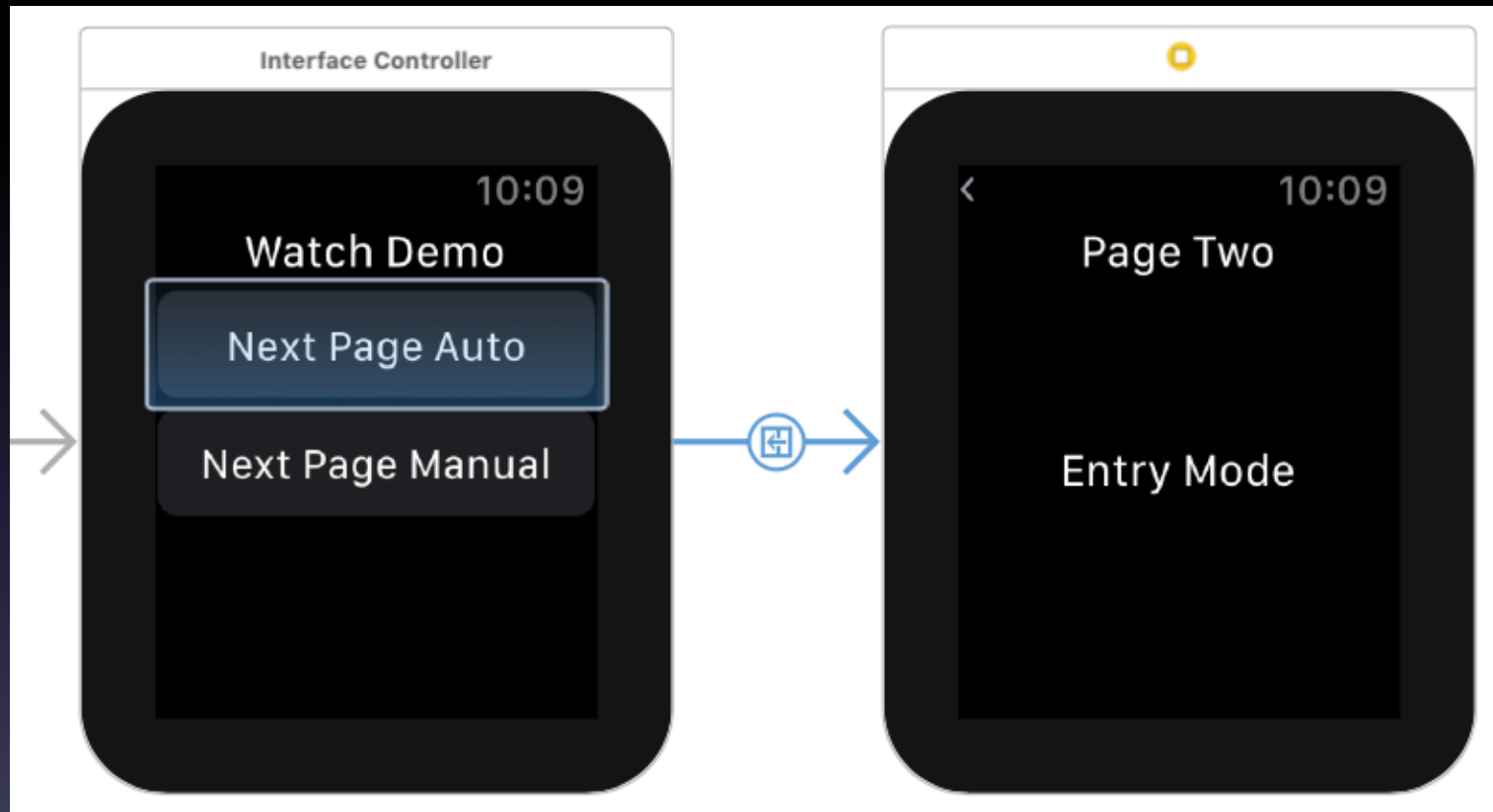
WatchKit App and Extension



Interface Storyboard



Interface Storyboard



Interface Controller

```
import WatchKit
import Foundation

enum EntryMode {
    case auto, manual
}

class InterfaceController: WKInterfaceController {

    @IBAction func nextPageManualTapped() {
        pushController(withName: "PageTwo",
                       context: EntryMode.manual)
    }

    override func contextForSegue(withIdentifier segueIdentifier:
        String) -> Any? {
        return EntryMode.auto
    }
}
```

InterfaceController.swift

Second Interface Controller

```
import WatchKit                                     SecondInterfaceController.swift
import Foundation

class SecondInterfaceController: WKInterfaceController {

    @IBOutlet var entryModeLabel: WKInterfaceLabel!

    override func awake(withContext context: Any?) {
        super.awake(withContext: context)
        // Configure interface objects here.
        let entryMode = context as! EntryMode
        if (entryMode == .auto) {
            self.entryModeLabel.setText("Auto")
        } else {
            self.entryModeLabel.setText("Manual")
        }
    }
}
```

Other Interface Objects



Table - Displays one or more rows of data.



Image - Displays a static or animated image.



Separator - A line for separating content in your interface.



Button - A tappable area with a title and/or image.



Payment Button - Standard button for initiating Apple Pay transactions.



Switch - A control for indicating a binary value.



Slider - A control for selecting a floating-point value from a range of continuous or discrete values.



Picker - A control for selecting an item from a list.

Label

Label - Displays a static text string.

9/9/14

Date - Displays the current date and time.

59:59

Timer - Displays a string that counts up or down to a specified time.



Activity Ring - Displays an Activity Ring



Map - Displays the user's location or the location of specific placemarks.



Movie - Displays a play button and poster image for audiovisual content.



Inline Movie - Displays a poster image for audiovisual content.



Menu - Displays a list of menu items.



Menu Item - Executes an action method of the parent interface controller.



SceneKit Scene - Displays SceneKit content.



SpriteKit Scene - Displays SpriteKit content.



HomeKit Camera - Displays the view of a HomeKit IP Camera



Tap Gesture Recognizer - Recognizes tap gestures, based on the number of taps.



Swipe Gesture Recognizer - Recognizes swipe gestures.

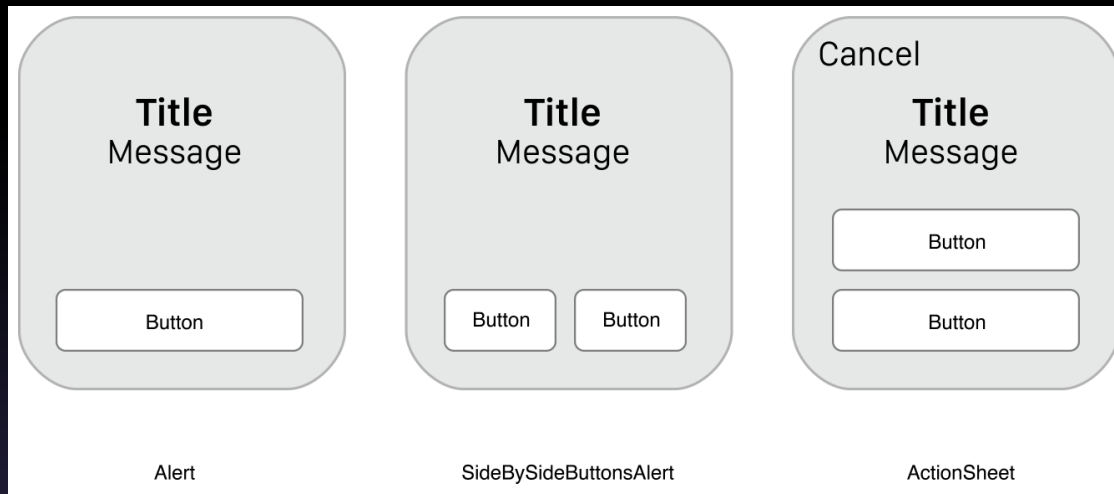


Long Press Gesture Recognizer - Recognizes long press gestures, based on the num...



Pan Gesture Recognizer - Recognizes pan (dragging) gestures.

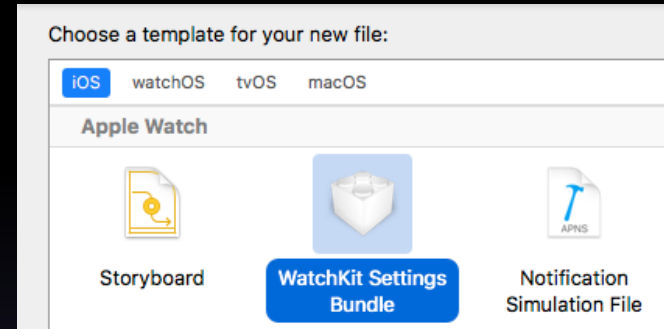
Alerts and Action Sheets



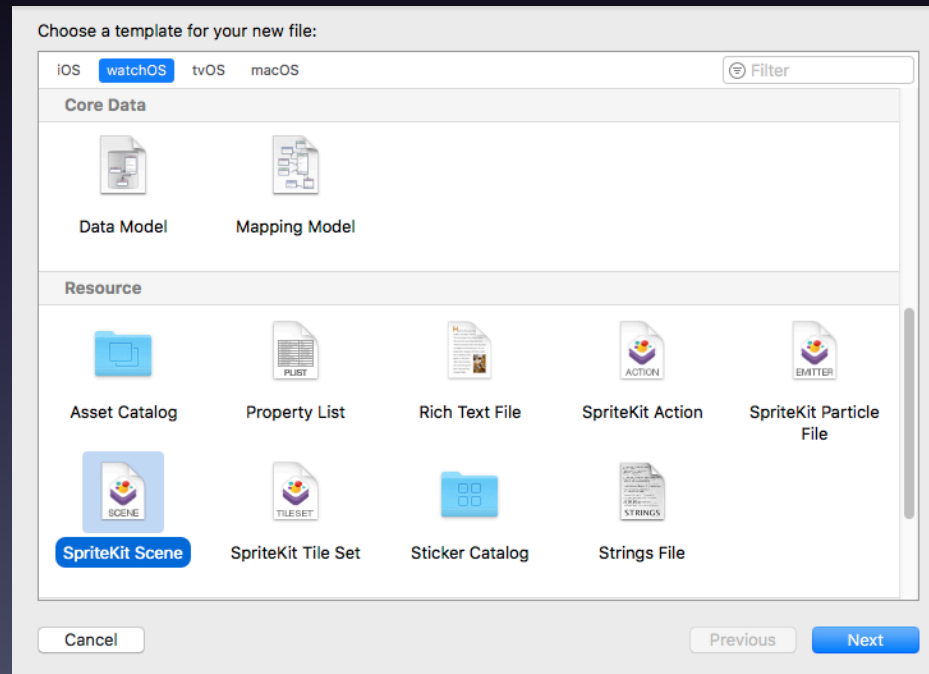
```
@IBAction func alertTapped() {  
    let action1 = WKAlertAction(title: "Yes",  
                                style: .default, handler: {})  
    let action2 = WKAlertAction(title: "No",  
                                style: .destructive, handler: {})  
    presentAlert(withTitle: "Alert",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

Other Elements

- Settings bundle

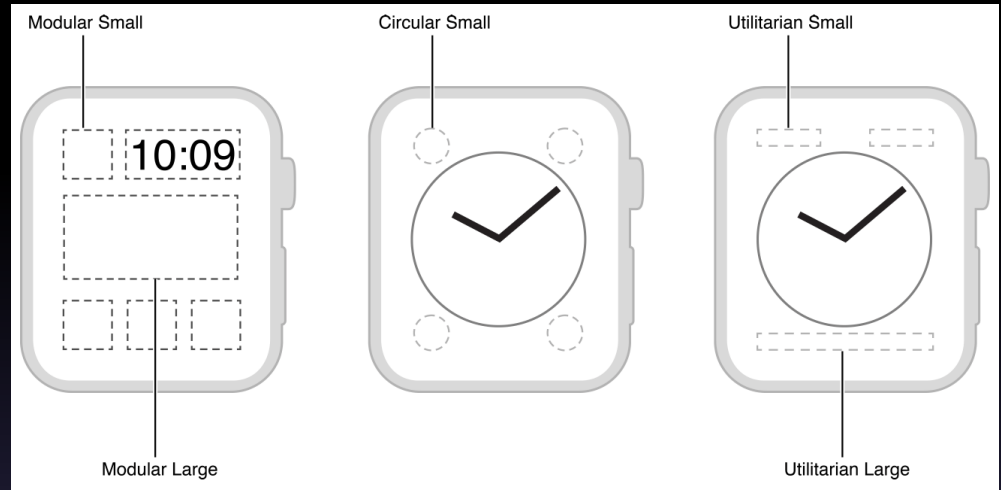


- Core data



- SpriteKit

Complications



- ClockKit framework
- CLKComplicationDataSource
- Provide data for a specific date/time
- Time Travel allows user to view past, present and future complication data (e.g., appointments)

Required Delegate Methods

- In `ComplicationController.swift`
 - `getSupportedTimeTravelDirections(complication, handler)`
 - Send `.backward`, `.forward`, neither, or both to handler
 - `getLocalizableSampleTemplate(complication, handler)`
 - Create and pass placeholder template to handler (or nil)
 - `getCurrentTimelineEntry(complication, handler)`
 - Create and pass time line entry to handler

getCurrentTimeLineEntry

- For desired complication families (.circularSmall, etc.)
 - Create a CLKComplicationTemplate, e.g.,
 - CLKComplicationTemplateCircularSmallSimpleImage
 - CLKComplicationTemplateCircularSmallSimpleText
 - Create and set providers for template, e.g.,
 - CLKImageProvider(UIImage)
 - CLKSimpleTextProvider(String)
 - Create time line entry for template at date
 - CLKComplicationTimelineEntry(Date, CLKComplicationTemplate)
 - Send entry to handler

Complications

```
import ClockKit

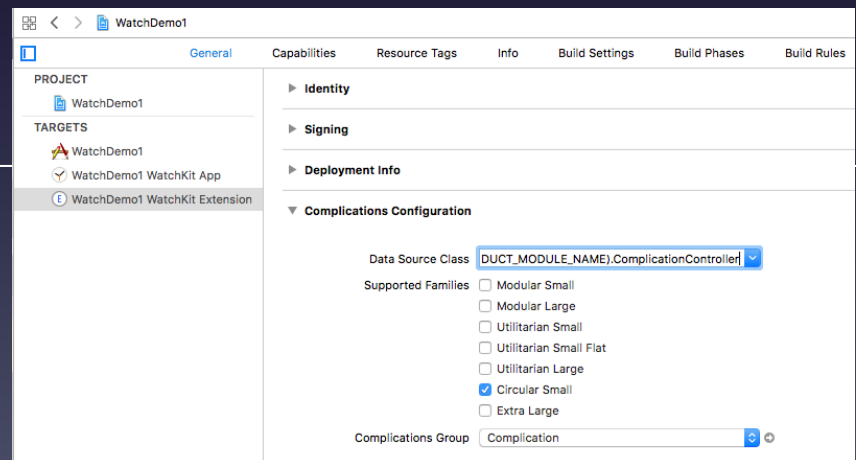
class ComplicationController: NSObject, CLKComplicationDataSource {

    func getSupportedTimeTravelDirections(for complication: CLKComplication,
        withHandler handler: @escaping(CLKComplicationTimeTravelDirections) -> Void)
    {
        handler([]) // or .forward, or .backward, or [.forward, .backward]
    }

    func getLocalizableSampleTemplate(for complication: CLKComplication,
        withHandler handler: @escaping (CLKComplicationTemplate?) -> Void)
    {
        // Called once per supported complication, results will be cached
        handler(nil) // system generates default placeholder template
    }
}
```

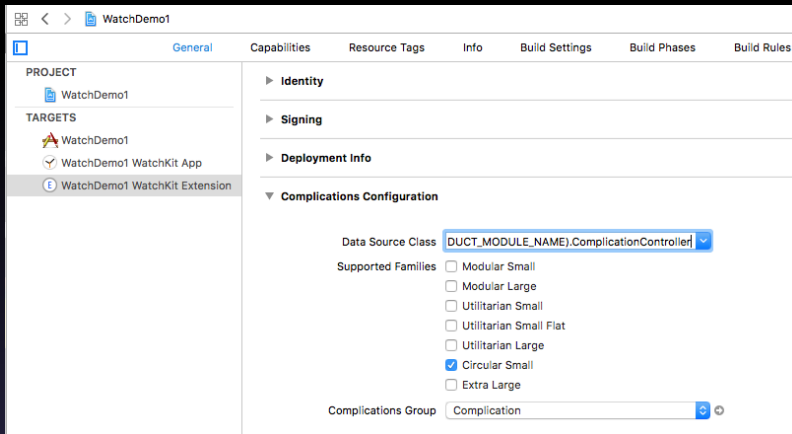

Complications

```
func getCurrentTimelineEntry(for complication: CLKComplication,  
    withHandler handler: @escaping (CLKComplicationTimelineEntry?) -> Void)  
{  
    if (complication.family == .circularSmall) {  
        let template = CLKComplicationTemplateCircularSmallSimpleImage()  
        // Only alpha channel of image used; colors ignored  
        let image = UIImage(named: "icon-D.png")  
        template.imageProvider = CLKImageProvider(onePieceImage: image!)  
        let entry = CLKComplicationTimelineEntry(date: Date(),  
            complicationTemplate: template)  
  
        handler(entry)  
    } else {  
        handler(nil)  
    }  
}
```

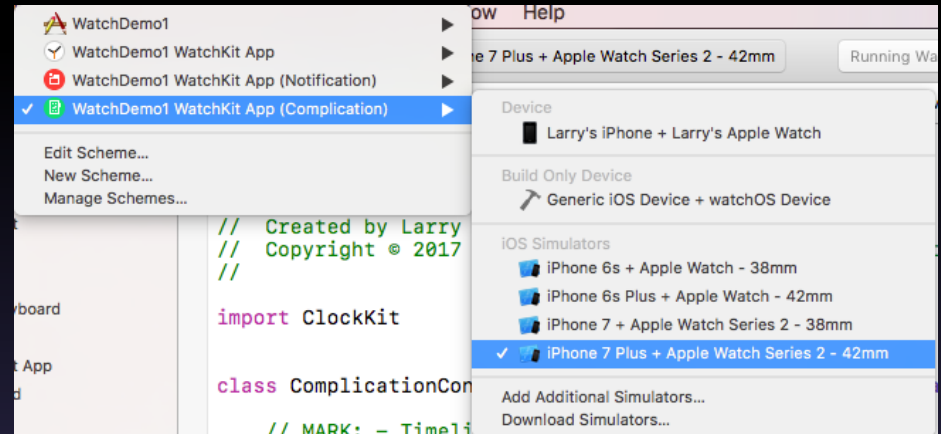


Complications: Testing

(1) Configure Complications



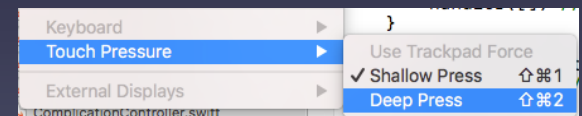
(2) Choose Complication scheme



(3) Customize clock face



Note: Shift-Command-2 for deep press on simulator.



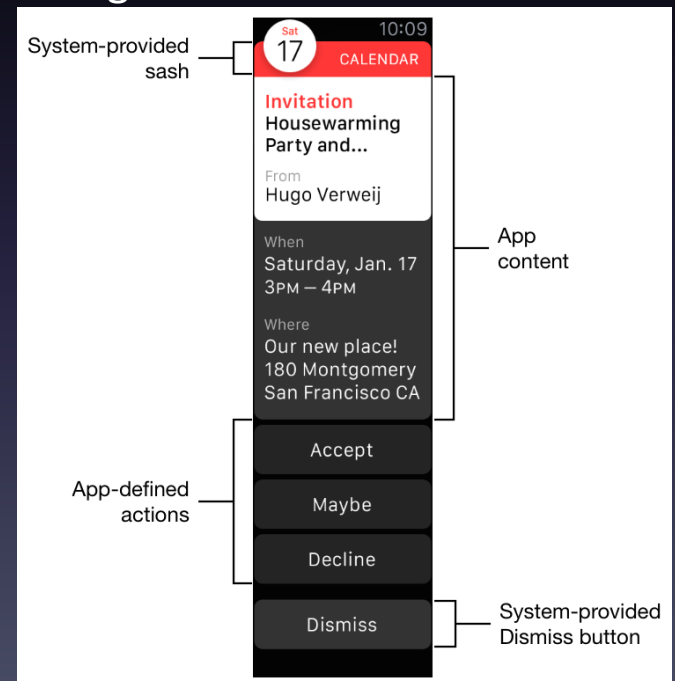
Notifications

- Uses same technique as iOS
 - Check for authorization on watch
 - Remote (push) notifications are sent first to the phone, and then from phone to watch
- Use (Notification) scheme to test notification
- First displays Short Look, then Long Look

Short Look



Long Look



Authorize Notifications

ExtensionDelegate.swift

```
import WatchKit
import UserNotifications

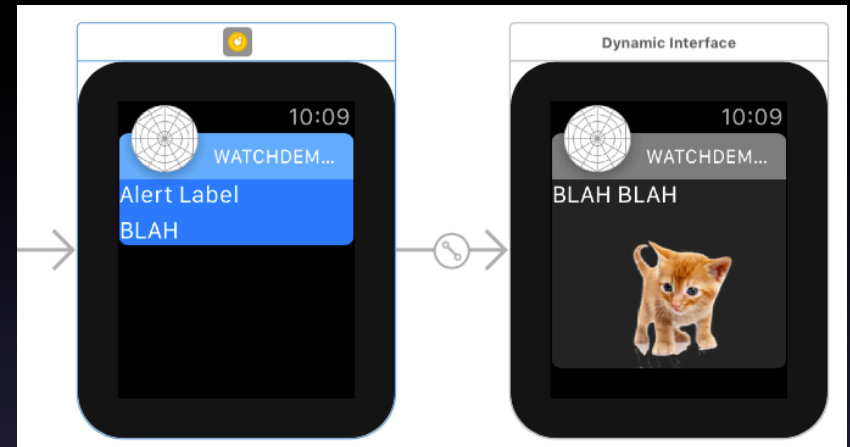
class ExtensionDelegate: NSObject, WKExtensionDelegate {

    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")
            }
        }
    }
}
```

**"WatchDemo1
WatchKit App"**
Would Like to Send
You Notifications
Notifications may
include alerts and
sounds. These can
be configured in
Settings.

Custom Notifications

- Static Notification
 - Simple interface (fast)
- Dynamic Notification
 - Allows custom content
 - Populate in code (not Storyboard)
 - If takes too long, uses Static



Handle Notifications

NotificationController.swift

```
import WatchKit
import Foundation
import UserNotifications

class NotificationController: WKUserNotificationInterfaceController {

    @IBOutlet var catImage: WKInterfaceImage!

    override func didReceive(_ notification: UNNotification,
                             withCompletion completionHandler: @escaping
                                (WKUserNotificationInterfaceType) -> Void) {
        self.catImage.setImage(UIImage(named: "cat_PNG1631.png"))
        completionHandler(.custom)
    }
}
```

Schedule Notifications

```
import UserNotifications

// Same as in Notifications lecture notes (except)
func scheduleNotification1() {
    let content = UNMutableNotificationContent()
    content.title = "Hey!"
    content.body = "What's up?"
    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) { (error : Error?) in
        if let theError = error {
            print(theError.localizedDescription)
        }
    }
}
```

Communications

- WatchConnectivity framework
- On phone and watch
 - Activate WCSSession
 - Adhere to WCSSessionDelegate
 - Send/receive messages

Watch Connectivity: Phone

```
import WatchConnectivity

class ViewController: UIViewController, WCSessionDelegate {

    override func viewDidLoad() {
        super.viewDidLoad()
        if WCSession.isSupported() {
            let session = WCSession.default()
            session.delegate = self
            session.activate()
        }
    }

    func session(_ session: WCSession, activationDidCompleteWith
        activationState: WCSessionActivationState, error: Error?) {
        print("session active")
    }

    func sessionDidBecomeInactive(_ session: WCSession) {
        print("session inactive")
    }

    func sessionDidDeactivate(_ session: WCSession) {
        print("session deactivated")
    }
}
```

Watch Connectivity: Phone

```
func session(_ session: WCSession, didReceiveMessage message: [String: Any]) {
    let msg = message["message"] as! String
    print("received message: \(msg)")
}

func sendMessage () {
    let session = WCSession.default()
    let msg = ["message": "Hello from Phone!"]
    session.sendMessage(msg, replyHandler: nil, errorHandler: nil)
}
```

Watch Connectivity: Watch

```
import WatchConnectivity

class InterfaceController: UIViewController, WCSessionDelegate {

    override func awake(withContext context: Any?) {
        super.awake(withContext: context)
        if WCSession.isSupported() {
            let session = WCSession.default()
            session.delegate = self
            session.activate()
        }
    }

    func session(_ session: WCSession, activationDidCompleteWith
        activationState: WCSessionActivationState, error: Error?) {
        print("session active")
    }

    // sessionDidBecomeActive and sessionDidDeactivate not used on Watch
}
```

Watch Connectivity: Watch

```
func session(_ session: WCSession, didReceiveMessage message: [String: Any]) {
    let msg = message["message"] as! String
    print("received message: \(msg)")
}

func sendMessage () {
    let session = WCSession.default()
    let msg = ["message": "Hello from Watch!"]
    session.sendMessage(msg, replyHandler: nil, errorHandler: nil)
}
```

Sensors

- CoreMotion framework
 - Accelerometer
 - Gyroscope
- CoreLocation framework
 - GPS
- HealthKit framework
 - Heart rate
 - developer.apple.com/reference/healthkit

Resources

- App Programming Guide for watchOS
 - developer.apple.com/library/content/documentation/General/Conceptual/WatchKitProgrammingGuide/index.html
- WatchConnectivity framework
 - developer.apple.com/reference/watchconnectivity
- HealthKit framework
 - developer.apple.com/reference/healthkit

Assets



Summary

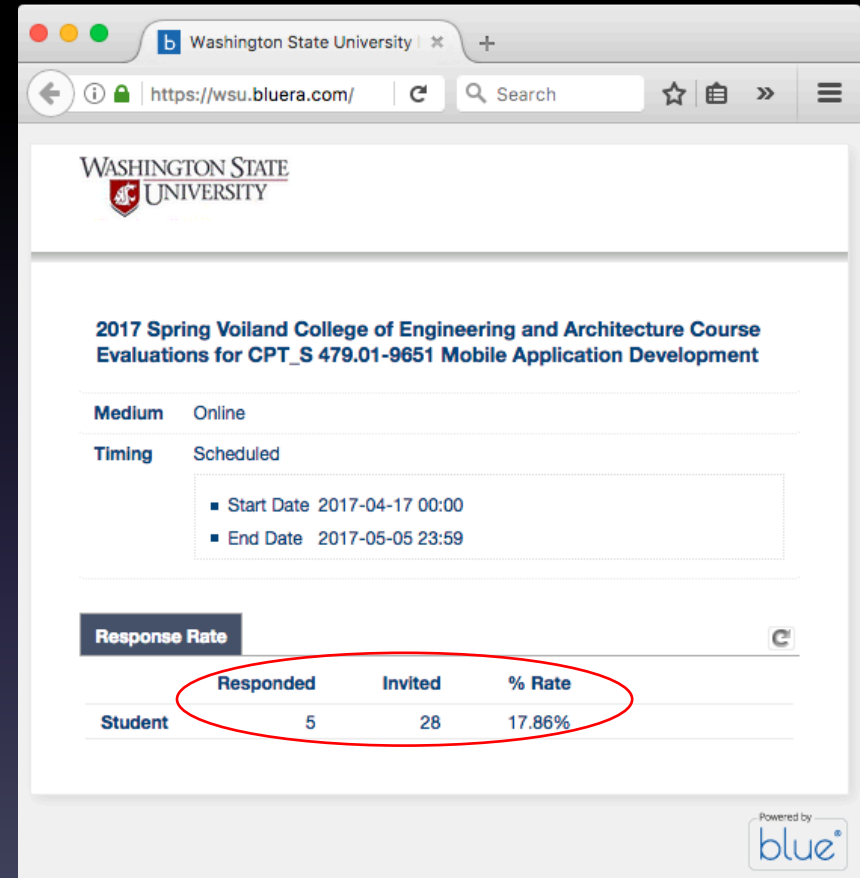
Mobile Application Development

iOS Mobile App Development

- Mobile App Development
- Swift
- UI Design and Storyboard
- Navigation and Segues
- Tables
- Settings
- Alerts and Notifications
- Gestures
- Sensors
- Communications
- Data Storage
- Graphics and Animation
- Multimedia
- Apple Watch

Thank You!

- Please fill out your Course Evaluation!
- Blue Course Evaluations on my.wsu.edu



WASHINGTON STATE UNIVERSITY

2017 Spring Voiland College of Engineering and Architecture Course Evaluations for CPT_S 479.01-9651 Mobile Application Development

Medium Online

Timing Scheduled

- Start Date 2017-04-17 00:00
- End Date 2017-05-05 23:59

Response Rate

	Responded	Invited	% Rate
Student	5	28	17.86%

Powered by blue

As of 4/27/2017