Notifications

Mobile Application Development in iOS

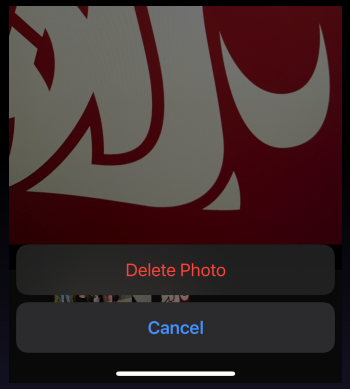
School of EECS

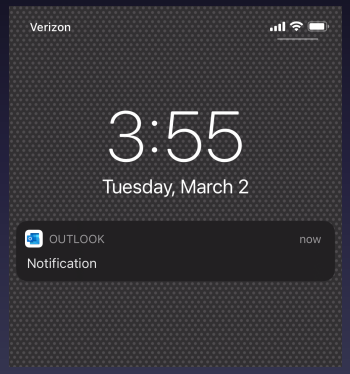
Washington State University

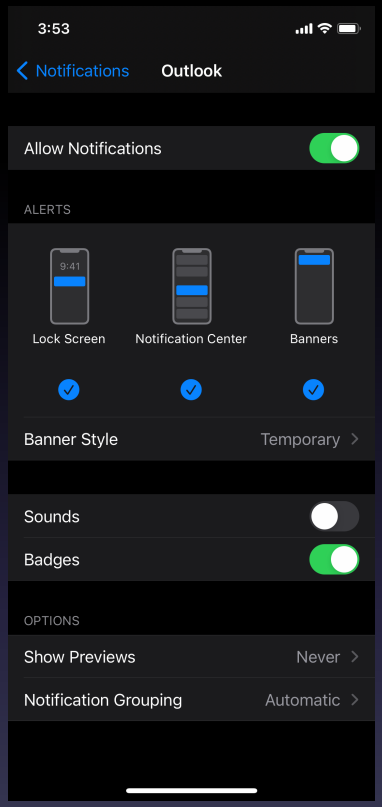
Instructor: Larry Holder

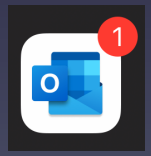
Outline

- Alerts
- Internal notifications
- Local notifications









Alerts

- UIAlertController
  - init (title, message, preferredStyle)
    - preferredStyles: .alert (popover), .actionSheet (bottom)
  - addAction (UIAlertAction)
    - init (title, style, handler)
      - Style: .default, .cancel, .destructive
  - preferredAction (bold on .alert style)
  - addTextField (configurationHandler)

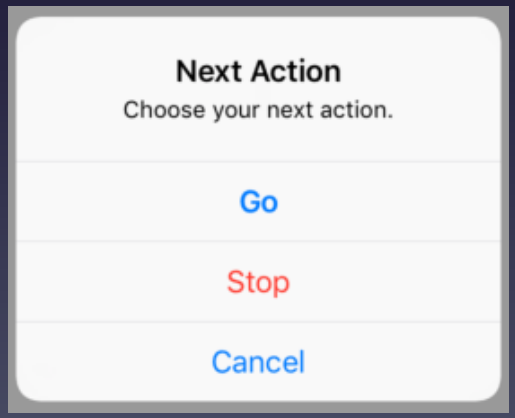
Alerts: Demo

```
let alert = UIAlertController(title: "Next Action",
    message: "Choose your next action.", preferredStyle: .alert)

let goAction = UIAlertAction(title: "Go", style: .default,
    handler: { (action) in
        // execute some code when this option is selected
        print("Go!")
    })

let stopAction = UIAlertAction(title: "Stop", style: .destructive,
    handler: { (action) in
        print("Stop!")
    })

let cancelAction = UIAlertAction(title: "Cancel", style: .cancel,
    handler: { (action) in
        print("Cancel.")
    })
```

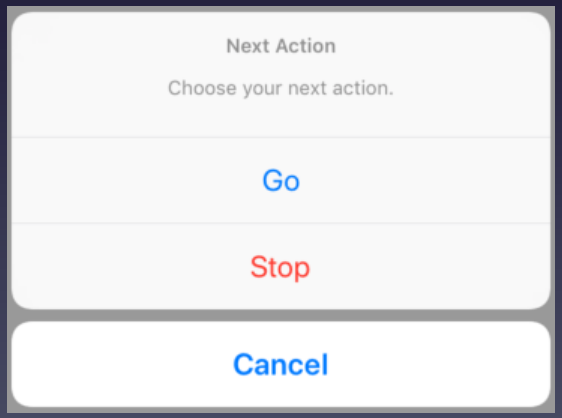
Alerts: Demo

```
...

alert.addAction(goAction)
alert.preferredAction = goAction // only affects .alert style
alert.addAction(stopAction)
alert.addAction(cancelAction)

present(alert, animated: true, completion: nil)
```

preferredStyle: .alert

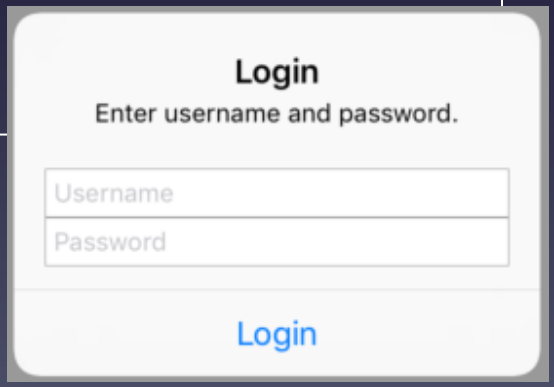


preferredStyle: .actionSheet



Alerts with Text Fields

```
func loginAlert() {
    let alert = UIAlertController(title: "Login",
        message: "Enter username and password.", preferredStyle: .alert)
    alert.addTextField(configurationHandler: { (textField) in
        textField.placeholder = "Username"
    })
    alert.addTextField(configurationHandler: { (textField) in
        textField.placeholder = "Password"
        textField.isSecureTextEntry = true
    })
    alert.addAction(UIAlertAction(title: "Login", style: .default,
        handler: { (action) in
            let username = alert.textFields?[0].text
            let password = alert.textFields?[1].text
            print("username = \(username!), password = \(password!)")
    }))
    present(alert, animated: true, completion: nil)
}
```



Internal Notifications

- Notify observers of an event
- NotificationCenter
  - addObserver
  - removeObserver
  - post(Notification)
- Notification names
  - developer.apple.com/documentation/foundation/nsnotification/name

Internal Notifications

- Example: Detect change to UserDefaults
  - Only when app in foreground

```
// In viewDidLoad
NotificationCenter.default.addObserver(self,
    selector: #selector(settingsChanged),
    name: UserDefaults.didChangeNotification, object: nil)

// In ViewController
@objc func settingsChanged (notification: Notification) {
    print("settings changed")
}
```

Internal Notifications

- Example: Detect when app becomes active

```
// In viewDidLoad
NotificationCenter.default.addObserver(self,
    selector: #selector(appActive),
    name: UIApplication.didBecomeActiveNotification, object: nil)

// In ViewController
@objc func appActive (notification: Notification) {
    print("app active")
}
```

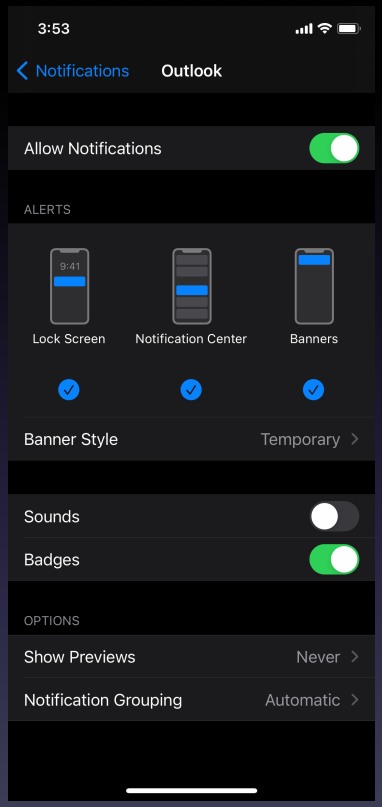
Internal Notifications

- Define you own

```
// In viewDidLoad
NotificationCenter.default.addObserver(self,
    selector: #selector(coolDetected),
    name: Notification.Name("coolNotification"), object: nil)

// In ViewController
@objc func coolDetected (notification: Notification) {
    print("Cool!")
}

// Anywhere
NotificationCenter.default.post(name:
    Notification.Name("coolNotification"), object: nil)
```

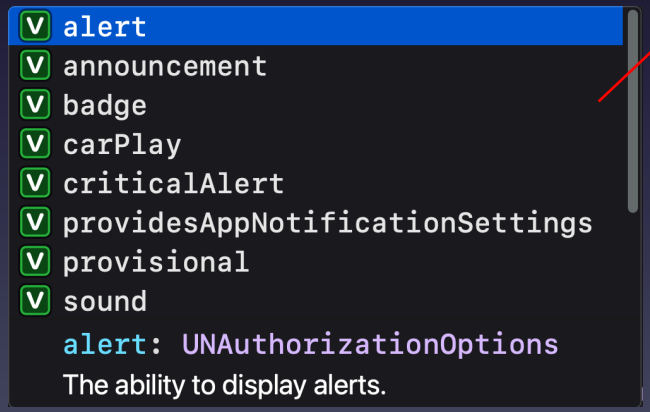
Local Notifications

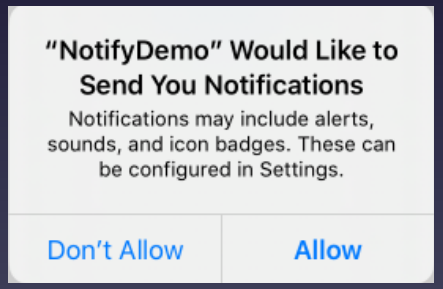
- User Notification Center
  - UNUserNotificationCenter.current()
- Request authorization to use notifications
  - requestAuthorization()
- Handle changes to authorizations
  - getNotificationSettings()

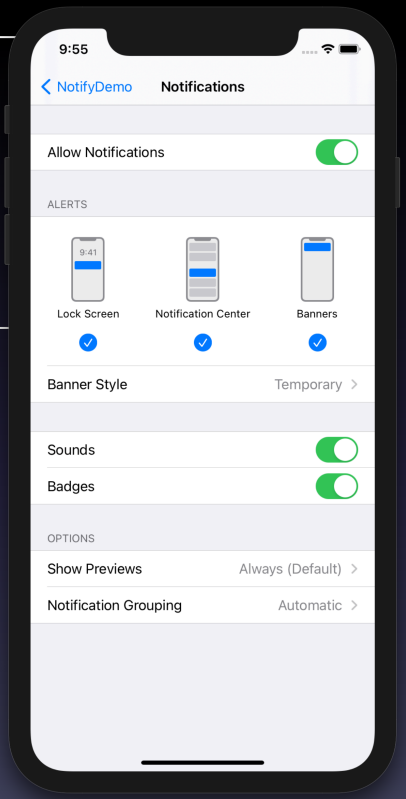


Local Notifications: Request Authorization

```
// In AppDelegate:didFinishLaunchingWithOptions
let center = UNUserNotificationCenter.current()
center.requestAuthorization(options: [.alert,.badge,.sound],
    completionHandler: { (granted, error) in
        // Enable features based on authorization
    })
```







Local Notifications: Handle Authorization Changes

```
@IBAction func scheduleNotificationTapped(_ sender: UIButton) {
    let center = UNUserNotificationCenter.current()
    center.getNotificationSettings(completionHandler: { (settings) in
        if settings.alertSetting == .enabled {
            self.scheduleNotification()
        } else {
            print("notifications disabled")
        }
    })
}
```

- Warning: getNotificationSettings returns immediately
  - completionHandler executed only after settings retrieved
  - Code inside completionHandler not executed on main thread

Scheduling Notifications

- (1) Create content
  - UNMutableNotificationContent
- (2) Create trigger
  - Based on time interval, date/time, location
  - UNTimeIntervalNotificationTrigger
  - UNCalendarNotificationTrigger
  - UNLocationNotificationTrigger

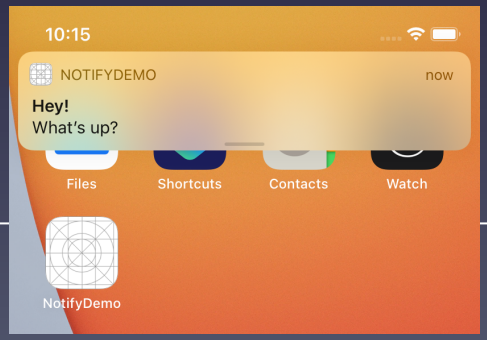
Scheduling Notifications

- (3) Create request
  - UNNotificationRequest(identifier, content, trigger)
- (4) Schedule notification
  - UNUserNotificationCenter.add(request)

Warning: Scheduling a notification request with identifier ID will remove any pending notifications with the same identifier.

Scheduling Notifications

```
func scheduleNotification() {
    let content = UNMutableNotificationContent()
    content.title = "Hey!"
    content.body = "What's up?"
    content.userInfo["message"] = "Yo!"
    // 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)
        }
    })
}
```



Receiving Notifications

- Handle in AppDelegate or SceneDelegate
  - Conform to UNUserNotificationCenterDelegate
  - UNUserNotificationCenter.current().delegate = self
  - didReceive()
    - Called if app in background or not running
  - willPresent()
    - Called if app running in foreground

Receiving Notifications: AppDelegate

```
class AppDelegate: UIResponder, UIApplicationDelegate,
    UNUserNotificationCenterDelegate {

    // In didFinishLaunchingWithOptions
    UNUserNotificationCenter.current().delegate = self

    func userNotificationCenter(_ center: UNUserNotificationCenter,
            didReceive response: UNNotificationResponse,
            withCompletionHandler completionHandler: @escaping () -> Void) {
        print("user responded to notification")
        // Do stuff with response here (non-blocking)
        let navVC = UIApplication.shared.windows.first!.rootViewController as!
                        UINavigationController
        if let mainVC = navVC.topViewController as? ViewController {
            mainVC.handleNotification(response)
        }
        completionHandler()
    }
```

Called even if app wasn't running.

```
// In ViewController
func handleNotification(_ response: UNNotificationResponse) {
    let message = response.notification.request.content.userInfo["message"]
                        as! String
    print("received notification message: \(message)")
}
```

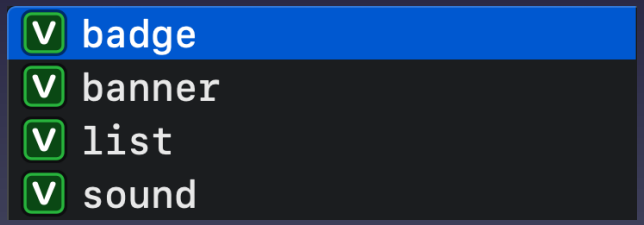
Receiving Notifications: SceneDelegate

```
class SceneDelegate: UIResponder, UIApplicationDelegate,
    UNUserNotificationCenterDelegate {

    // In scene:willConnectTo
    UNUserNotificationCenter.current().delegate = self

    func userNotificationCenter(_ center: UNUserNotificationCenter,
            didReceive response: UNNotificationResponse,
            withCompletionHandler completionHandler: @escaping () -> Void) {
        print("user responded to notification")
        // Do stuff with response here (non-blocking)
        let navVC = self.window?.rootViewController as! UINavigationController
        if let mainVC = navVC.topViewController as! ViewController {
            mainVC.handleNotification(response)
        }
        completionHandler()
    }
```

Called even if app wasn't running.

Receiving Notifications While App in Foreground

```
// In AppDelegate or SceneDelegate

func userNotificationCenter(_ center: UNUserNotificationCenter,
        willPresent notification: UNNotification, withCompletionHandler
        completionHandler: @escaping (UNNotificationPresentationOptions)
        -> Void)
{
    print("received notification while in foreground; display?")
    completionHandler([.banner]) // no options ([]) means no notification
}
```



Remove Notifications

- Pending
  - getPendingNotificationRequests(completionHandler: @escaping ([UNNotificationRequest]) -> Void)
  - removePendingNotificationRequests(withIdentifiers: [String])
  - removeAllPendingNotificationRequests()
- Delivered
  - getDeliveredNotifications(completionHandler: @escaping ([UNNotification]) -> Void)
  - removeDeliveredNotifications(withIdentifiers: [String])
  - removeAllDeliveredNotifications()

Local Notifications: Other Options

- Add sound (< 30 seconds)
  - UNNotificationSound
- Set app icon badge number
  - UNMutableNotificationContent.badge
- Configure different categories of notifications
  - Add custom actions: UNNotificationAction
  - Create category: UNNotificationCategory

Remote (Push) Notifications

- Send notifications from server to app on particular device
- Requires secure capabilities
- Delivered same as local notifications
  - didReceive, willPresent
- Only works on real device (not simulator)
- developer.apple.com/documentation/usernotifications/setting_up_a_remote_notification_server

Resources

- Alerts
  - developer.apple.com/documentation/uikit/uialertcontroller
- Internal notifications
  - developer.apple.com/documentation/foundation/notifications
- Local and remote notifications
  - developer.apple.com/documentation/usernotifications