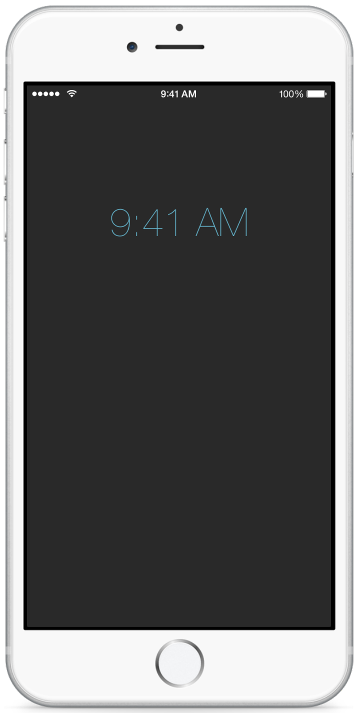
Clock
Lesson 4

Description

Leverage iOS notifications and observers to display the correct time when the app is started and brought to the foreground.



Learning Outcomes

- Recognize the concept of notifications in the iOS platform.
- Apply `NSNotificationCenter` and an observer to perform behavior during application life cycle events.
- Compare the approach of using notifications with other explicit means of event handling.
- Combine Swift parameters into multi-parameter method calls.
- Define *selector*, and compose a method call that expects a `selector` argument.

Vocabulary

notification	observer	`NSNotificationCenter`
parameter	function arity	refactor
observer registration	selector	object life cycle
memory management	deinitializer	`deinit`

Materials

- **Clock Lesson 4** Xcode project
- **Notifications** presentation

Opening

How can we tell the app to update the displayed time when the app enters the foreground?

Agenda

- Present the concepts of `NSNotificationCenter`, notifications, and observers.
- Discuss the difference between internal application notifications and app notifications for end users.
- Explore the `NSNotificationCenter` class reference, its `defaultCenter` class method and the `addObserver:selector:name:object:` method.
- Register the controller as an observer in `viewDidLoad`.

```
NSNotificationCenter.defaultCenter().addObserver(self,
   selector: "updateTimeLabel",
   name: UIApplicationWillEnterForegroundNotification,
   object: nil)
```

- Discuss parameterized Swift method syntax, and how to keep long method calls readable with formatting.
- Discuss the meaning of calling the `addObserver:selector:name:object:` method and the significance of the passed arguments.
- Explain the concept of selectors.
- Implement the controller `updateTimeLabel` method.

```
func updateTimeLabel() {
   let formatter = NSDateFormatter()
   formatter.timeStyle = .ShortStyle
   timeLabel.text = formatter.stringFromDate(clock!.currentTime)
}
```

- Refactor `viewWillAppear:` to update the initial displayed time.

```
override func viewWillAppear(animated: Bool) {
   super.viewWillAppear(animated)
   updateTimeLabel()
}
```

- Run the app (⌘**R**) and use the Simulator to send the app to the background (⇧⌘**H**). Wait until the OS X menu bar time indicator has changed, and bring the app to the foreground. Observe that the time is current.
- Experiment with using an invalid selector name when registering an observer in `viewDidLoad`. Run the app (⌘**R**), send the app to the background (⇧⌘**H**), bring the app to the foreground, and observe the app crashing. Restore the correct selector name.
- Explain the best practice of unregistering observers when an application quits or is "destroyed" from memory.
- Unregister the observer in a deinitializer.

```
deinit {
   NSNotificationCenter.defaultCenter().removeObserver(self)
}
```

- Explain the deinitializer's role in object life cycles and iOS memory management.
- Discuss how the app delegate has no controller-related responsibilities, and how the view controller encapsulates the coordination of updating the view.

Closing

What happens when you run the app for longer than a minute? Does the time update itself? How do we continuously update the display with the current time?

Modifications and Extensions

- Explore the ability to observe additional app life cycle notifications using `NSNotificationCenter`.

Resources

Cocoa Core Competencies: Notification https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/Notification.html

Notification Programming Topics: Registering for a Notification https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/Notifications/Articles/Registering.html

NSNotificationCenter Class Reference https://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSNotificationCenter_Class/index.html

Using Swift with Cocoa and Objective-C: Objective-C Selectors https://developer.apple.com/library/ios/documentation/Swift/Conceptual/BuildingCocoaApps/InteractingWithObjective-CAPIs.html#//apple_ref/doc/uid/TP40014216-CH4-XID_40

The Swift Programming Language: Deinitialization https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Deinitialization.html

The Swift Programming Language: Automatic Reference Counting https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/AutomaticReferenceCounting.html