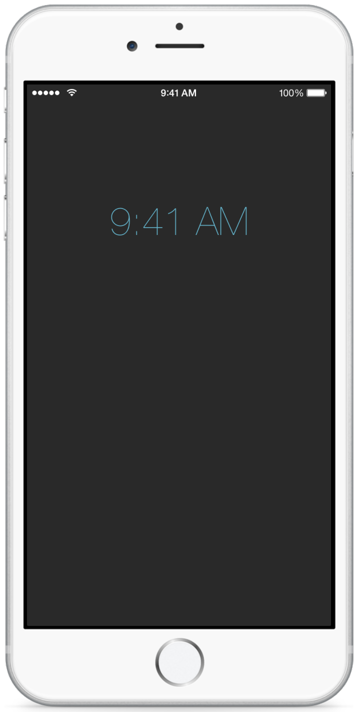
Clock
Lesson 3

Description

Observe that the time does not update when restoring the app from the background. Explore the life cycle events of starting, backgrounding, foregrounding, and quitting an app.



Learning Outcomes

- Analyze application, controller and view life cycle events.
- Describe object-oriented inheritance, and relate inheritance to iOS view controllers.
- Practice using the Xcode Documentation and API Reference to discover technical information.
- Interpret the concepts of delegates and protocols.
- Apply Xcode breakpoints as an alternative to using `print` and the Xcode console.

Vocabulary

background	foreground	iOS Multitasking Bar
force quit	inheritance	extend
`UIViewController`	override	`super`
app delegate	`UIApplicationDelegate`	`UIResponder`
life cycle events	protocol	breakpoint

Materials

- **Clock Lesson 3** Xcode project
- **Delegates and Delegation** presentation

Opening

What happens when we send an app to the background, and restore it to the foreground?

Agenda

- Using the Simulator, 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 not current.
- Using the Multitasking Bar (⇧⌘H, twice quickly), force quit the app and start it again. Notice the time is now correct.
- Discuss why the time is correct only when starting the app.
- Add a `print` call in `viewDidLoad`.

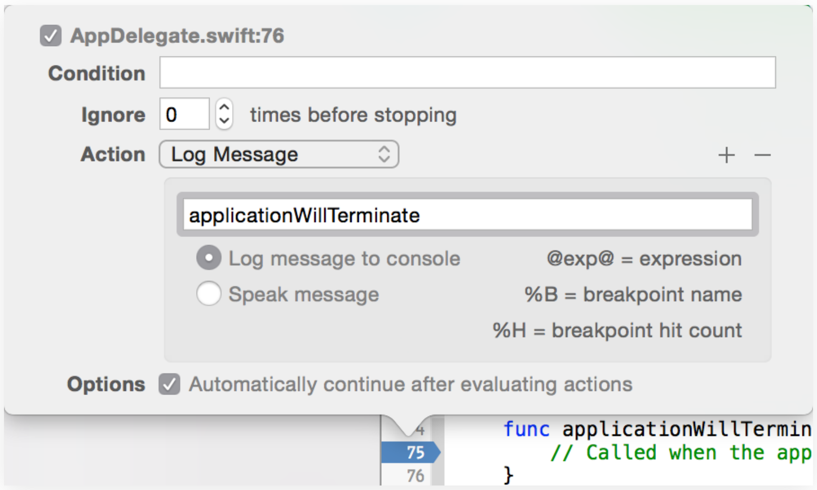
```
print("viewDidLoad")
```

- Run the app (⌘R), and observe the Xcode console (⇧⌘C) while repeating the starting, backgrounding, foregrounding and quitting of the app.
- Discuss when an iOS app seems to execute its `viewDidLoad` method.
- Examine the class declaration for `ViewController`, noting that it extends `UIViewController`.
- Discuss object-oriented inheritance.
- Using the Xcode Documentation and API Reference (⇧⌘0), explore the `UIViewController` class reference and notice its life cycle methods.
- Experiment with attempting to set the current time by overriding `viewWillAppear:`.

```
override func viewWillAppear(animated: Bool) {
    super.viewWillAppear(animated)
    print("viewWillAppear")
    let formatter = NSDateFormatter()
    formatter.timeStyle = .ShortStyle
    timeLabel.text = formatter.stringFromDate(clock.currentTime)
}
```

- Observe the Xcode console (⇧⌘C) while foregrounding and backgrounding the app. Notice how `viewWillAppear:` is also not the appropriate lifecycle method.

- Using the Project Navigator (⌘1), examine **AppDelegate.swift**.
- Present the concept of delegates.
- Briefly explain what the primary "app delegate" is, how it extends `UIResponder`, and implements the `UIApplicationDelegate` protocol.
- Using the Xcode Documentation and API Reference (⇧⌘0), explore the documentation for the `UIApplicationDelegate` protocol, and notice its life cycle methods.
- Demonstrate how, instead of adding a `print` call to all `AppDelegate` methods, to use Xcode to add breakpoints that automatically continue after writing a message to the console.



```
func applicationWillTermin
    // Called when the app
}
```

- Observe the Xcode console (⇧⌘C) while starting, backgrounding, foregrounding, quitting and restarting the app.
- Discuss which `UIApplicationDelegate` lifecycle event is likely best suited for the feature of updating the currently displayed time.
- Discuss how `applicationWillEnterForeground` is the desired method, and the challenge of how you might update the view from the app delegate when the app enters the foreground.
- Discuss how the controller should be responsible for communicating with the view, and how writing view-related code in the `AppDelegate` may violate a separation of concerns.

Closing

Might there be a convenient way for the controller to be notified when the app enters the foreground?

Modifications and Extensions

- Explicitly implement `applicationWillEnterForeground` such that it navigates the object graph to send a message to the main view controller to set the current time in the `UILabel`.

Resources

iOS Simulator User Guide: Interacting with iOS Simulator https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/iOS_Simulator_Guide/InteractingwiththeiOSSimulator.html

Searching Developer Documentation http://developer.apple.com/library/ios/recipes/xcode_help-documentation_organizer/SearchingDocumentation/SearchingDocumentation.html

Start Developing iOS Apps Today: Finding Information https://developer.apple.com/library/ios/referencelibrary/GettingStarted/RoadMapiOS/FindingInformation.html

UIViewController Class Reference https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIViewController_Class/index.html

View Controller Programming Guide for iOS: Responding to Display-Related Notifications https://developer.apple.com/library/ios/featuredarticles/ViewControllerPGforiPhoneOS/RespondingtoDisplay-Notifications/RespondingtoDisplay-Notifications.html

App Programming Guide for iOS: The App Life Cycle https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html

App Programming Guide for iOS: Execution States for Apps https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html#//apple_ref/doc/uid/TP40007072-CH2-SW3

Cocoa Application Competencies for iOS: Application object https://developer.apple.com/library/ios/documentation/General/Conceptual/Devpedia-CocoaApp/ApplicationObject.html

UIApplicationDelegate Protocol Reference https://developer.apple.com/library/ios/#documentation/UIKit/Reference/UIApplicationDelegate_Protocol/Reference/Reference.html

UIResponder Class Reference https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIResponder_Class/index.html

Source Editor Help: Adding, Disabling and Deleting Breakpoints https://developer.apple.com/library/ios/recipes/xcode_help-source_editor/chapters/Creating,Disabling,andDeletingBreakpoints.html