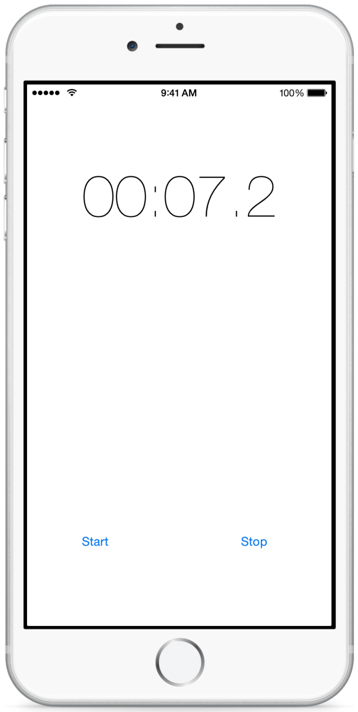
Stopwatch
Lesson 7



Description

Refactor the elapsed time formatting into a model property, and make the frequently changing elapsed time label accommodate assistive devices.

Learning Outcomes

- Recognize user diversity and the concept of accessibility, and describe accessibility features of iOS.
- Recognize the principle of "separation of concerns" between controllers and models, and apply refactoring to properly abstract model concerns.
- Assess a working application and formulate additional features related to usability and accessibility.

Vocabulary

user experience	accessibility	assistive device
Identity Inspector	separation of concerns	refactor
computed property		

Materials

- **Stopwatch Lesson 7** Xcode project
- **Apple Accessibility** web resources

Opening

Without enhancing its functionality, what subtle improvements might we make to our app?

Agenda

- Present the concepts of user diversity, user experience, and accessibility with the Apple Accessibility web resources.
- Explain how assistive devices notify the user of changes in the view. Consider how frequently the view is changing while the `Stopwatch` is running, and how this might cause an excessive number of notifications to assistive devices.
- Using Interface Builder, select the elapsed time label and use the Identity Inspector (**⌥⌘3**) to check the *Accessibility* > *Updates Frequently* trait, and uncheck the *User Interaction Enabled* trait.
- Discuss the numerous `print` calls in the code, and how using customized breakpoints is a better approach.
- Delete the remaining `print` calls in the `ViewController` implementation.
- Discuss how the creation of individual time components in the `ViewController` `updateElapsedTimeLabel:` method sounds more like a concern of the `Stopwatch` model.
- Extract the formatted `String` generation and time component code from the `ViewController` `updateElapsedTimeLabel:` method into a new computed property in the `Stopwatch` class.

```
var elapsedTimeAsString: String {
   return String(format:"%02d:%02d.%d",Int(elapsedTime / 60),
      Int(elapsedTime % 60), Int(elapsedTime * 10 % 10))
}
```

- Discuss the computed property syntax.
- Update the `ViewController` `updateElapsedTimeLabel:` method to use the new `elapsedTimeAsString` property.

```
func updateElapsedTimeLabel(timer: NSTimer) {
   if stopwatch.isRunning {
      elapsedTimeLabel.text = stopwatch.elapsedTimeAsString
   } else {
      timer.invalidate()
   }
}
```

- Discuss how the `updateELapsedTimeLabel:` method now relies on the `Stopwatch` `elapsedTimeAsString` property, rather than formatting the elapsed time itself.
- Run the app (⌘**R**), and observe that the functionality has not changed.
- Discuss the read-only nature of the `Stopwatch` model's computed `isRunning` property, and how it is impossible to explicitly set `isRunning` to `false` after starting a `Stopwatch`.
- Discuss how the `private` access modifier of the `startTime` property is important, preventing the ability to directly access or modify the `Stopwatch` `startTime` property.
- Discuss the lack of a Reset button, but how the Start button restarts the elapsed time.
- Discuss how the Start, Stop and reset functionalities exist, and discuss if the user experience is good enough or not.
- Discuss how usability, user experience and accessibility might further improve.

Closing

What other features could we add to our Stopwatch app?

Modifications and Extensions

- Enhance the user interface by changing the colors of the buttons depending on the model state.
- Investigate key-value-observing and how the model might notify the controller to update the elapsed time label instead of using an `NSTimer`.
- Design additional features to enhance the usability, accessibility and value of the app.

Resources

Accessibility for iOS Developers https://developer.apple.com/accessibility/ios/

Accessibility Programming Guide for iOS: Making Your App Accessible https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/iPhoneAccessibility/Making_Application_Accessible/Making_Application_Accessible.html

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

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

The Swift Programming Language: Computed Properties https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Properties.html#//apple_ref/doc/uid/TP40014097-CH14-ID259