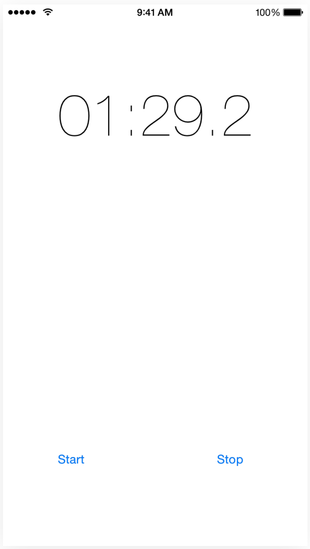
Stopwatch
Lesson 6

Description

Correctly display the elapsed time in minutes, seconds, and tenths of a second.



Learning Outcomes

- Analyze programming bugs and discover solutions to program flaws.
- Apply mathematical operations on floating point numbers to extract desired values, such as time components.
- Apply Swift type conversion to convert numeric types.
- Apply string formatting to satisfy user interface requirements.

Vocabulary

`NSTimeInterval`	type conversion	`Int`
remainder operator	modulo	`String`
string format specifier		

Materials

- **Stopwatch Lesson 6** Xcode project

Opening

Now that the label is updating, how can we display the elapsed time in minutes, seconds and tenths of a second?

Agenda

- Discuss the need to display the elapsed time at tenth of a second intervals.
- Update the `NSTimer` instantiation, using `0.1` as the time interval.

```
NSTimer.scheduledTimerWithTimeInterval(0.1 ...
```

- Discuss the need to display the elapsed time in minutes, seconds and tenths of a second.
- Using the Xcode Documentation and API Reference (⇧⌘0), explore the `NSDateComponentsFormatter` `stringFromTimeInterval:` method.
- Discuss how the `stringFromTimeInterval:` method does not provide formatting for tenths of a second.
- Using the Xcode Documentation and API Reference (⇧⌘0), explore the String Format Specifiers guide.
- Improve `updateElapsedTimeLabel:` to assign an appropriately formatted `String` to the `elapsedTimeLabel` `text` property.

```
func updateElapsedTimeLabel(timer: NSTimer) {
    println("updating...")
    if stopwatch.isRunning {
        let minutes = Int(stopwatch.elapsedTime / 60)
        let seconds = Int(stopwatch.elapsedTime % 60)
        let tenthsOfSecond = Int(stopwatch.elapsedTime * 10 % 10)
        elapsedTimeLabel.text = String(format:"%02d:%02d.%d",
            minutes, seconds, tenthsOfSecond)
    } else {
        timer.invalidate()
    }
}
```

- Discuss the use of the remainder operator (`%`), the type conversion to `Int`, and the `String` format specifiers.
- Run the app (⌘R), tap the Start button, and observe the elapsed time.

Closing

Although the basic functionality is complete, there are subtle problems with our app. How many problems can you identify?

Modifications And Extensions

- Explore the `Printable` protocol and its `description` read-only property. Have the `Stopwatch` model adopt `Printable`, such that it returns a `String` representing the elapsed time.
- Explore the `NSDateComponents` class and use one to extract the time components of the `Stopwatch` `elapsedTime`.
- Explore the `NSDateComponentsFormatter` and manipulate `elapsedTime` to enable the use of an `NSDateComponentsFormatter`.

Resources

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

NSDateComponentsFormatter Class Reference https://developer.apple.com/library/prerelease/ios/documentation/Foundation/Reference/NSDateComponentsFormatter_class/index.html

The Swift Programming Language: Remainder Operator https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/BasicOperators.html#//apple_ref/doc/uid/TP40014097-CH6-ID64

The Swift Programming Language: Numeric Type Conversion https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html#//apple_ref/doc/uid/TP40014097-CH5-ID324

Swift Standard Library Reference: String https://developer.apple.com/library/ios/documentation/General/Reference/SwiftStandardLibraryReference/

String Programming Guide: String Format Specifiers https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/Strings/Articles/formatSpecifiers.html