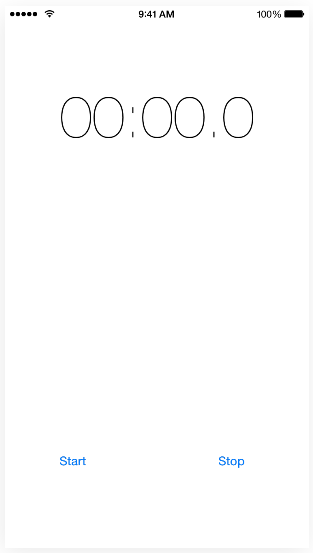
Stopwatch
Lesson 3

Description

Integrate the model, view and view controller, by declaring a `Stopwatch` property, and connecting the Start and Stop buttons to controller methods.



Learning Outcomes

- Describe the relationship between controllers and models.
- Practice declaring properties with default values.
- Explain the relationship between views and controllers.
- Apply Interface Builder to establish connections between view components and controller methods.
- Describe how controller methods interact with the model.

Vocabulary

property	default value	initialization
view	controller	action connection
`@IBAction`	Assistant Editor	connection well
`println`	console	

Materials

- **Stopwatch Lesson 3** Xcode project

Opening

How can we cause the user interface buttons to start and stop the `Stopwatch` model?

Agenda

- Discuss the relationship between models, views and controllers.
- Add a `Stopwatch` property to the `ViewController` class.

```
let stopwatch = Stopwatch()
```

- Explain why the property is declared as a constant, and how the property will be assigned its default `Stopwatch` value when the `ViewController` is initialized.
- Using Interface Builder and the Assistant Editor(⌥⌘↩), add controller actions for the Start and Stop buttons in the controller class.

```
@IBAction func startButtonTapped(sender: UIButton) {
}

@IBAction func stopButtonTapped(sender: UIButton) {
}
```

- Discuss the significance of the `@IBAction` keyword and Interface Builder connections.
- Experiment with removing `@IBAction` and observe the connection well disappear. Restore the `@IBAction` keyword and observe the connection well reappear.
- Explain the significance of the `sender` parameter.
- Implement `startButtonTapped` and `stopButtonTapped` to start and stop the `Stopwatch` property. Use `println` to examine the result of interacting with the buttons.

```
@IBAction func startButtonTapped(sender: UIButton) {
   println("Starting stopwatch")
   stopwatch.start()
}

@IBAction func stopButtonTapped(sender: UIButton) {
   println(stopwatch.elapsedTime)
   stopwatch.stop()
}
```

- Run the application (⌘R), and observe the console (⇧⌘C) while interacting with the Start and Stop buttons.

Closing

Should the view ever interact directly with the model? Why or why not?

Modifications And Extensions

- Investigate how and where the controller itself is instantiated within an iOS app.
- Replace the `println` calls with customized breakpoints that generate a console message and automatically continue.

Resources

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

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

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

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

View Controller Programming Guide for iOS https://developer.apple.com/library/ios/featuredarticles/ViewControllerPGforiPhoneOS/

Xcode Overview: Connect User Interface Objects to Code https://developer.apple.com/library/ios/documentation/ToolsLanguages/Conceptual/Xcode_Overview/edit_user_interface.html#//apple_ref/doc/uid/TP40010215-CH6-SW3

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

IBAction UIKit Constants Reference https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIKitConstantsReference/#//apple_ref/c/macro/IBAction