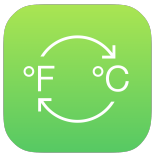
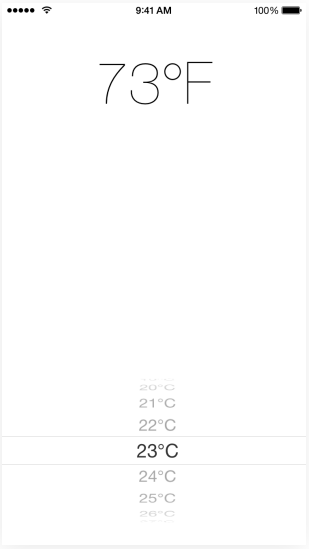
UnitConverter
Lesson 11



Description

Load and display the last selected temperature when the app is brought to the foreground.



Learning Outcomes

- Practice using Xcode breakpoints to inspect an app during runtime.
- Devise a flow of control using Boolean logic, given a problem statement.
- Practice learning about APIs using documentation.
- Observe the Swift type cast syntax.
- Apply optional binding to perform conditional logic to fulfill a feature requirement.

Vocabulary

life cycle method	breakpoint	`NSUserDefaults`
`AnyObject?`	optional	downcasting
`as?`	`nil`	optional binding
`if let`		

Materials

- **UnitConverter Lesson 11** Xcode project

Opening

How and when should the app load the last selected temperature?

Agenda

- Discuss where in the codebase the last selected temperature should be loaded.
- Experiment with placing custom breakpoints within the app and controller life cycle methods, especially `viewDidLoad`, to determine the best time to load the saved picker view `row` index.
- Refactor `viewDidLoad` to use an `initialPickerRow` method, instead of a local variable, to determine the initial selected `row` index of the picker view.

```
override func viewDidLoad() {
   super.viewDidLoad()
   let row = initialPickerRow()
   celsiusPicker.selectRow(row, inComponent: 0, animated: false)
   pickerView(celsiusPicker, didSelectRow: row, inComponent: 0)
}

func initialPickerRow() -> Int {
   // load from user defaults
   // if we obtained a last-known row index, return it
   // otherwise, return the default.
   return celsiusPicker.numberOfRowsInComponent(0) / 2
}
```

- Using the Xcode documentation and API Reference (⇧⌘0), explore the `NSUserDefaults integerForKey:` method, and observe how it returns `0` when a value for the provided key is not found.
- Implement a naive `initialPickerRow` method.

```
func initialPickerRow() -> Int {
   let savedRow = NSUserDefaults.standardUserDefaults()
      .integerForKey(userDefaultsLastRowKey)
   if savedRow != 0 {
      return savedRow
   } else {
      return celsiusPicker.numberOfRowsInComponent(0) / 2
   }
}
```

- Run the app (⌘R), select a temperature, force quit the app via the multitasking bar (⇧⌘H), restart the app, and witness the last selected temperature is correctly displayed.
- With the app still running, discuss what will happen if the user selects the first temperature in the picker view (e.g. -100) and restarts the app.
- Run the app (⌘R), select the first temperature in the picker view, force quit the app via the multitasking bar (⇧⌘H), restart the app, and witness that the default temperature is selected instead of the last selected temperature.
- Using the Xcode documentation and API Reference (⇧⌘0), explore the `NSUserDefaults objectForKey:` method, and observe how it returns an `AnyObject?` type.
- Discuss how `objectForKey:` will return an object representing the saved value, or `nil` when the provided key is not found.
- Update the implementation of `initialPickerRow`.

```
func initialPickerRow() -> Int {
   let savedRow = NSUserDefaults.standardUserDefaults()
      .objectForKey(userDefaultsLastRowKey) as? Int
   if let row = savedRow {
      return row
   } else {
      return celsiusPicker.numberOfRowsInComponent(0) / 2
   }
}
```

- Explain how `objectForKey:` returns an `AnyObject?` optional, and the use of the `as?` type cast operator to downcast to an `Int?`.
- Discuss how `savedRow` will be an `Int?` optional that wraps either an `Int` value or `nil`, and how the optional binding of `row` results in returning the `Int` value wrapped by `savedRow`, or results in returning the default row when `savedRow` does not contain a value.
- Run the app (⌘R), select the first temperature in the picker view, force quit the app via the multitasking bar (⇧⌘H), restart the app, and witness that the last selected temperature is correctly displayed.

Closing

What other features do you think we could add to the app to make it more useful or more fun to use? What about converting other units, such as monetary exchange rates? Or toggling the conversion from Fahrenheit to Celsius?

Modifications And Extensions

- Explore the `NSUserDefaults` class reference and discover what types of values an app can store and retrieve easily.
- Investigate Core Data, and determine why using Core Data is appropriate or inappropriate for persisting the user's preferences.
- Change the color of the converted temperature depending on how hot or cold the temperature is.
- Add a button to the interface to "flip" the conversion to Fahrenheit to Celsius.
- Add a Tab-Bar Controller to the app, and implement additional views for converting other units of measure.
- Investigate how to access actual weather data to set the default displayed temperature to the real current temperature outside.

Resources

Preferences and Settings Programming Guide https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/UserDefaults/

NSUserDefaults Class Reference https://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSUserDefaults_Class/

The Swift Programming Language: Type Casting for Any and AnyObject https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TypeCasting.html#//apple_ref/doc/uid/TP40014097-CH22-ID342

The Swift Programming Language: Downcasting https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TypeCasting.html#//apple_ref/doc/uid/TP40014097-CH22-ID341

The Swift Programming Language: Optional Binding https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html#//apple_ref/doc/uid/TP40014097-CH5-ID333

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