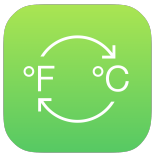
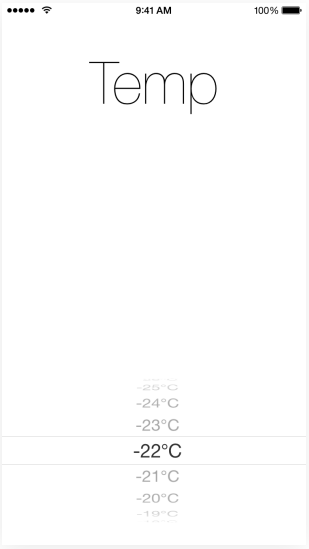
UnitConverter
Lesson 4



Description

Introduce an array of temperature values for the picker view to display.



Learning Outcomes

- Explain the array data structure and create an array using initializer syntax.
- Analyze errors due to incorrect array indexing and formulate corrections to the cause of such errors.
- Practice writing control flow statements.
- Relate `for` loops to `for-in` loops, and recognize the Swift range syntax.
- Observe how `map` transforms a range of values into an array.

Vocabulary

array	property	access level modifier
`private`	mutability	string interpolation
`for` loop	`for-in` loop	range
closure		

Materials

- **UnitConverter Lesson 4** Xcode project

Opening

How can we provide the picker view with a range of Celsius values?

Agenda

- Discuss the need for a "list" of negative and positive Celsius temperatures for the picker view to display. Consider the total number of values (how many possible temperatures?), and the range (what minimum and maximum temperatures?).
- In the controller, add a private property for an `Array` of temperature values that the controller will provide to the picker view for display.

```
private var temperatureValues = [Int]()
```

- Discuss the syntax of the property default value, using `var` to indicate mutability, and the significance of making the property private.
- Implement a naive, temporary assignment of the `temperatureValues` property during `viewDidLoad`.

```
override func viewDidLoad() {
   super.viewDidLoad()
   temperatureValues = [1, 2, 3, 4, 5]
}
```

- Update the implementation of `pickerView:titleForRow:forComponent:`.

```
func pickerView(pickerView: UIPickerView, titleForRow row: Int,
   forComponent component: Int) -> String! {
   let celsiusValue = temperatureValues[row]
   return "\(celsiusValue)°C"
}
```

- Explain how an `Int` is obtained from the array, and is interpolated within the returned `String` value.
- Run the app (⌘R), observe the values displayed in the picker, and flick the picker one row at a time until the app crashes.
- Observe the console error "fatal error: Array index out of range."
- Discuss the meaning of the error, which describes how the picker selected a row, called `pickerView:titleForRow:forComponent:`, but the row index was outside of the bounds of the `temperatureValues` array.

- Update `pickerView:titleForRow:forComponent:` to use the size of the temperatureValues array to inform the picker of how many rows to display.

```
func pickerView(pickerView: UIPickerView,
   numberOfRowsInComponent component: Int) -> Int {
   return temperatureValues.count
}
```

- Run the app (⌘R), observe the temperature values, and interact with the picker.
- Discuss strategies for creating a range of temperature values, from -100 to 100, contrasting explicit array initialization(`[-100, -99, …, 99, 100]`) with programmatic initialization using a loop.
- Modify `viewDidLoad` to naively populate the `temperatureValues` array with a loop.

```
override func viewDidLoad() {
   super.viewDidLoad()
   let lowerBound = -100
   let upperBound = 100
   for var index = lowerBound; index <= upperBound; ++index {
      temperatureValues.append(index)
   }
}
```

- Discuss the C-style `for` loop and the `Array` `append` method.
- Run the app (⌘R), observe the temperature values in the picker, and interact with the picker.
- Discuss how the loop "counts" from `lowerBound` to `upperBound`, appending each value of `index` to the `temperatureValues` array.
- Replace the C-style `for` loop in `viewDidLoad` with a `for-in` loop.

```
for index in -100...100 {
   temperatureValues.append(index)
}
```

- Explain the range syntax and how the `for-in` loop iterates over the range of numbers `-100` through `100`, assigning each value to `index` during each repetition of the loop.
- Discuss how `map` might be used to transform a range into an array of `Int` values.
- Explain the concept of the `map` function.
- Update the `temperatureValues` property declaration and remove the procedural temperature value generation from `viewDidLoad`.

```
private let temperatureValues = (-100...100).map { $0 }

override func viewDidLoad() {
    super.viewDidLoad()
}
```

- Explain how the `map` function iterates over each value in the range, and the use of the `$0` symbol to refer to the value passed to the closure.
- Run the app (⌘R), and observe the temperature values in the picker.
- Discuss how the creation of the `temperatureValues` array is now more concise, and how the property declaration no longer relies on a mutable `Array`.

Closing

Now that we have a range of temperature values to select, what specific steps should we take to display the converted temperature?

Modifications And Extensions

- Rewrite the `map` call to use an implicit constant and `in` rather than the `$0` symbol.
- Instead of calling `map` as a member function of the range, use the free function `map`, passing it the range as its first argument.

Resources

View Controller Programming Guide for iOS: Resource Management in View Controllers http://developer.apple.com/library/ios/#featuredarticles/ViewControllerPGforiPhoneOS/ViewLoadingandUnloading/ViewLoadingandUnloading.html

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

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

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

UIPickerViewDataSource Protocol Reference https://developer.apple.com/library/ios/documentation/iPhone/Reference/UIPickerViewDataSource_Protocol/

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