A Swift Kickstart

Daniel H Steinberg
@dimsumthinking

Enumerations

Create an enumeration

```
enum CocoaConfTutorial {
    case GameDevelopment
    case AVFoundation
    case Swift
    case AllDayiOS
}

var tutorial = CocoaConfTutorial.AllDayiOS
```

Create an enumeration

```
enum CocoaConfTutorial {
    case GameDevelopment
    case AVFoundation
    case Swift
    case AllDayiOS
}

var tutorial = CocoaConfTutorial.AllDayiOS

tutorial = .Swift
```

Enumerations and Switch

```
switch tutorial {
case .Swift, .AllDayiOS:
    println("What a great choice")
default:
    println("Whatever")
}
```

Raw Values - Int

```
enum CocoaConfTutorial: Int {
    case GameDevelopment
    case AVFoundation
    case Swift
    case AllDayiOS
}

var tutorial = CocoaConfTutorial.AllDayiOS
tutorial.rawValue
tutorial = .Swift
tutorial.rawValue
```

Raw Values - Int

```
enum CocoaConfTutorial: Int {
    case GameDevelopment = 7
    case AVFoundation
    case Swift
    case AllDayiOS
}

var tutorial = CocoaConfTutorial.AllDayiOS
tutorial.rawValue
tutorial = .Swift
tutorial.rawvalue
```

Optional from raw

```
possibleTutorial
    = CocoaConfTutorial(rawValue: "AV Foundation Film School")

if let tutorial = possibleTutorial {
    switch tutorial {
    case .Swift, .AllDayiOS:
        println("What a great choice")
    default:
        println("Whatever")
    }
}
```

Raw Values - String

```
enum CocoaConfTutorial: String {
    case GameDevelopment = "iOS Game Development"
    case AVFoundation = "AV Foundation Film School"
    case Swift = "A Swift Kickstart"
    case AllDayiOS =  "All Day iOS Tutorial"
}

var tutorial = CocoaConfTutorial.AllDayiOS
tutorial.rawValue
tutorial = .Swift
tutorial.rawValue
```

Optional

```
var possibleTutorial: CocoaConfTutorial?

possibleTutorial = CocoaConfTutorial.fromRaw("AV Foundation Film School")

if let tutorial = possibleTutorial {
    switch tutorial {
    case .Swift, .AllDayiOS:
        println("What a great choice")
    default:
        println("Whatever")
    }
}
```

Methods

```
enum CocoaConfTutorial: String {
    case GameDevelopment = "iOS Game Development"
    case AVFoundation = "AV Foundation Film School"
    case Swift = "A Swift Kickstart"
    case AllDayiOS =  "All Day iOS Tutorial"

    func tutorialName() -> String {
        return rawValue
    }
}

var swift = CocoaConfTutorial.Swift

swift.tutorialName()
```

Properties

```
enum CocoaConfTutorials: String {
    case GameDevelopment = "iOS Game Development"
    case AVFoundation = "AV Foundation Film School"
    case Swift = "A Swift Kickstart"
    case AllDayiOS =  "All Day iOS Tutorial"

    var tutorialName: String {
        return rawValue
    }
}

var swift = CocoaConfTutorial.Swift

swift.tutorialName
```

Using Enumerations

```
class CocoaConfTutorialAttendee: CocoaConfAttendee {
    var tutorial: CocoaConfTutorial
    init(name: String, hometown: String, tutorial: CocoaConfTutorial) {
        self.tutorial = tutorial
        super.init(name: name, hometown: hometown)
    }
    override func nameBadge() -> String {
        return super.nameBadge() + ", I'm taking \(tutorial.tutorialName)"
    }
}
```

Using Enumerations

```
let tutorialAttendee = CocoaConfTutorialAttendee (name: "Kim",
                                                  hometown: "Shaker",
                                                  tutorial:.AVFoundation)
tutorialAttendee.nameBadge()
tutorialAttendee.tutorial = .Swift
tutorialAttendee.nameBadge()
```

Associated Values

```
enum CocoaConfRegistrationType {
    case Regular
    case Tutorial(CocoaConfTutorial)
}

class CocoaConfAttendee {
    let name: String
    let hometown: String?
    let registrationType = CocoaConfRegistrationType.Regular
    // ...
```

Associated Values

```
enum CocoaConfRegistrationType {
    case Regular
    case Tutorial(CocoaConfTutorial)
}

class CocoaConfAttendee {
    let name: String
    let hometown: String?
    let registrationType = CocoaConfRegistrationType.Regular
    // ...
```

Associated Values

```
class CocoaConfTutorialAttendee: CocoaConfAttendee {
    init(name: String, hometown: String, tutorial: CocoaConfTutorial) {
        super.init(name: name, hometown: hometown)
        registrationType = CocoaConfRegistrationType.Tutorial(tutorial)
    }
    override func nameBadge() -> String {
        return super.nameBadge() + ", I'm taking \(tutorial.tutorialName)"
    }
}
```

Associated Values

```
enum CocoaConfRegistrationType {
    case Regular
    case Tutorial(CocoaConfTutorial)

    var tutorialName:String? {
        switch self {
        case .Tutorial(let tutorial):
            return tutorial.tutorialName
        default:
            return nil
        }
    }
}
```

Associated Values

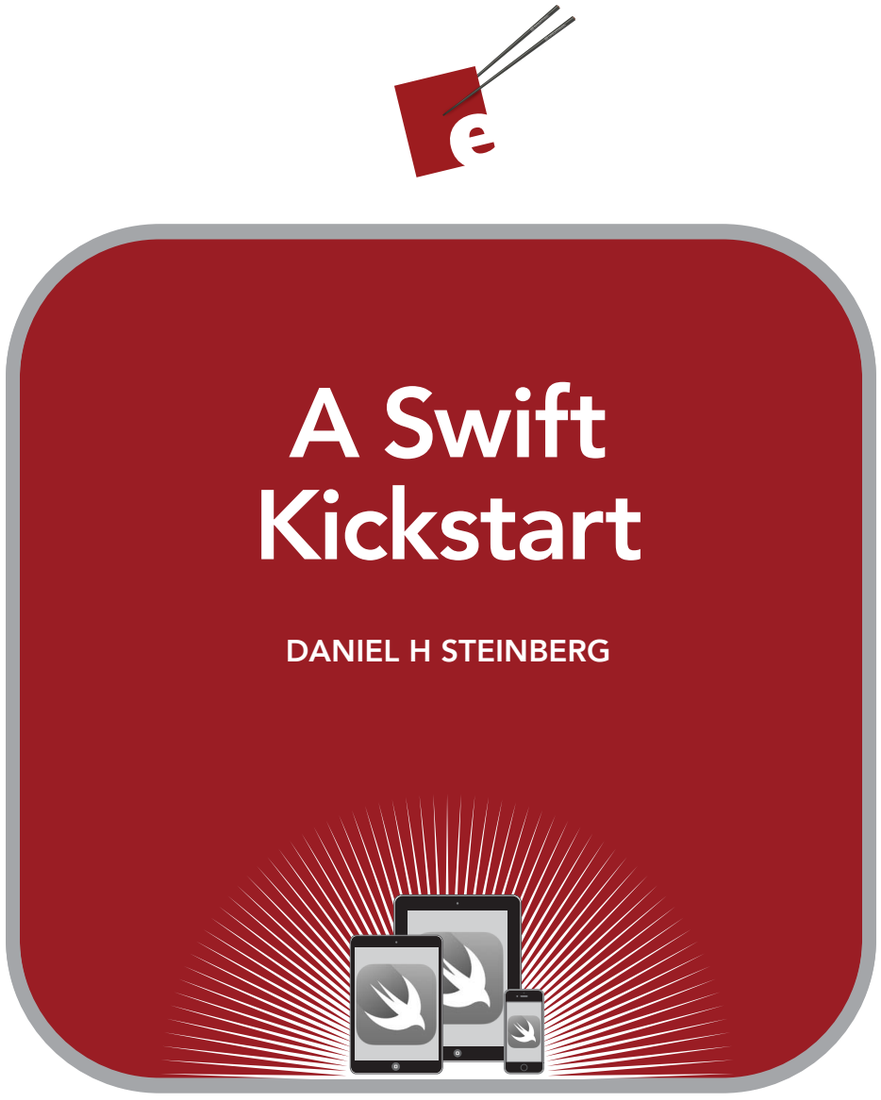
```
let tutorialAttendee = CocoaConfTutorialAttendee (name: "Kim",
                                                  hometown: "Shaker",
                                                  tutorial:.AVFoundation)
tutorialAttendee.nameBadge()
tutorialAttendee.registrationType = .Tutorial(.Swift)
tutorialAttendee.nameBadge()
```

Try this

- Continue with the example from the last section.
- Create an enumeration named SizeOfCup that has Int raw values for three different sizes.
- Create an init method for HotBeverage that takes a SizeOfCup and sets the amount to the raw value

Try this

```
enum SizeOfCup: Int {
    case Small = 8
    case Medium = 12
    case Large = 16
}
class HotBeverage: Beverage {
    func sip() {
        amount--
        if isEmpty {
            println("You're done")
        }
    }
    init(sizeOfCup: SizeOfCup) {
        super.init(amount: sizeOfCup.rawValue)
    }
}
let coffee = HotBeverage(sizeOfCup: .Medium)
for i in 1...8 {
    coffee.sip()
}
```



https://itunes.apple.com/us/book/a-swift-kickstart/id891801923?mt=11&uo=4&at=11l56E