Swift Protocols

Cody Henrichsen

Definition

- Swift's version of an interface
- Protocols are a first class Type
- A Swift Type adopts a protocol
- A Swift Type will "conform" to a protocol if it implements all methods
- A Swift Type can conform to multiple protocols
- Conforming means the Type in question will implement all associated methods and properties

Protocol

- Methods and data members that MUST be implemented in the attached class
- Often have a variant of –able as the suffix for the protocol name as that indicates that root+able is a collection of methods and properties of things that match that root
  - Drivable
  - Flyable
  - Edible
  - Togglable

Protocol Data Members

- Must have an explicit get/set property defined
- The adopting Type must have that data member as one of its own
- Variable naming conventions are important
- Great place to use the //MARK: documentation tag in Xcode to separate regular class variables from those required for conforming to the protocol.

Protocol Methods

- No Access Control Modifier!!!
- func nameOfMethod(params) -> ReturnType
- NO IMPLEMENTATION AT ALL!!!
- Again method names REALLY matter with these since you want to make sure all methods for the protocol are easily checked
- Use of the "//MARK: - "documentation comment is highly recommended to identify all the required methods in a block

Sample Protocol

```
//
//  Squishable.swift
//  DemoRemake
//
//  Created by Cody Henrichsen on 11/14/17.
//  Copyright © 2017 CTEC. All rights reserved.
//

import Foundation
public protocol Squishable
{
    var squishState : Bool {get set}
    func squish() -> Void
    func isSquashed() -> Bool
}

```

Class Protocol Adoption

```
import UIKit
public class DebugDuck : Squishable
{
    private var duckColor :UIColor
    //MARK: Squishable data member
    public var squishState: Bool

    public init()
    {
        self.duckColor = UIColor()
        self.squishState = false
    }

    //MARK:- Squishable methods
    public func squish() -> Void
    {
        print("I am squishy")
        squishState = true
    }

    public func isSquashed() -> Bool
    {
        if(squishState)
        {
            print("squashed")
        }
        else
        {
            print("poofy")
        }
        return squishState
    }
}
```