



A presentation slide titled "Swift Protocols" by Cody Henriksen. It features a large purple square on the left with a small pink plus sign in the top-left corner. To its right are four smaller squares arranged in a 2x2 grid: yellow (top-left), light blue (top-right), olive green (bottom-left), and dark purple (bottom-right).

## Swift Protocols

Cody Henriksen

---

---

---


---

---

---

---

---



A slide titled "Definition" with a pink plus sign in the top-left corner. It contains a bulleted list of six points. A vertical purple bar is on the right side.

## 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

---

---

---


---

---

---

---

---



A slide titled "Protocol" with a pink plus sign in the top-left corner. It contains a bulleted list of two points, with the second point having a sub-list of four items. A vertical purple bar is on the right side.

## 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
  - Toggleable

---

---

---

---

---

---

---

---

## + 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

```
1 //
2 // Squishable.swift
3 // DemoRemake
4 //
5 // Created by Cody Henrichsen on 11/14/17.
6 // Copyright © 2017 CTEC. All rights reserved.
7 //
8
9 import Foundation
10 public protocol Squishable
11 {
12     var squishState : Bool {get set}
13     func squish() -> Void
14     func isSquashed() -> Bool
15 }
16
```

---

---

---

---

---

---

---

## + Class Protocol Adoption

```
1 import UIKit
2 public class DebugDuck : Squishable
3 {
4     private var duckColor : UIColor
5     //MARK: Squishable data member
6     public var squishState: Bool
7
8     public init()
9     {
10         self.duckColor = UIColor()
11         self.squishState = false
12     }
13
14     //MARK:- Squishable methods
15     public func squish() -> Void
16     {
17         print("I am squishy")
18         squishState = true
19     }
20
21     public func isSquashed() -> Bool
22     {
23         if(squishState)
24         {
25             print("squashed")
26         }
27         else
28         {
29             print("poofy")
30         }
31         return squishState
32     }
33 }
```