SWIFT

PEDAL TO THE METAL

ABOUT ME

```swift
struct Developer {
  let name = "Piotr Sochalewski"
  let photo = UIImage(named: "ps_avatar")
  let company = "Droids On Roids"
  let github = "sochalewski"
}
```

WHY?

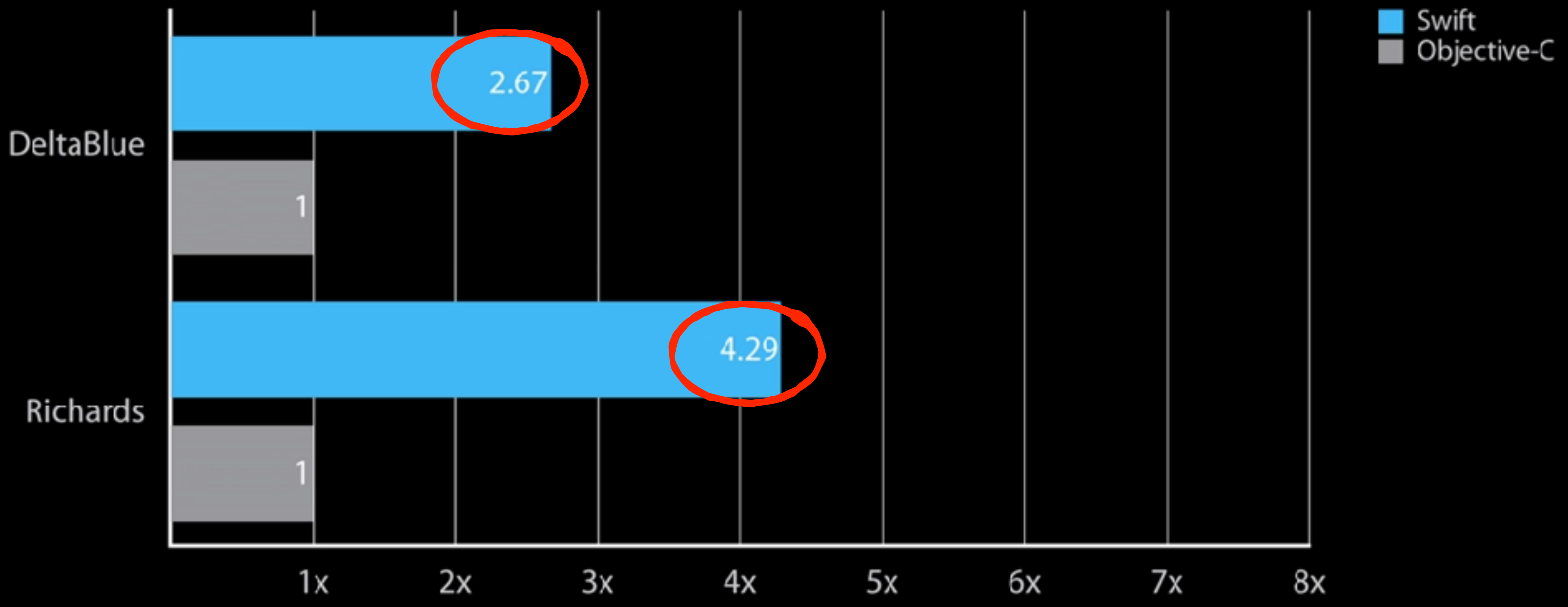
🤔

SWIFT IS A LOT FASTER THAN OBJECTIVE-C

OPTIMIZING SWIFT PERFORMANCE – WWDC 2015

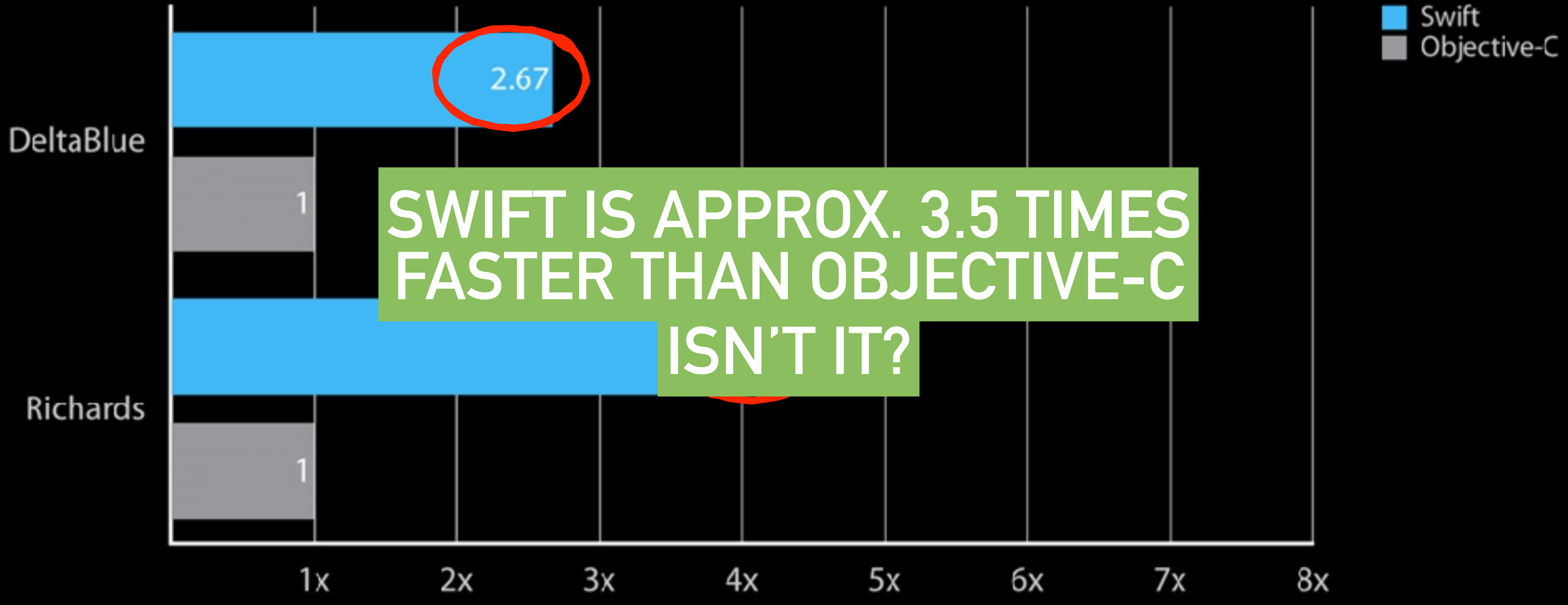
Swift vs. Objective-C

Program speed (higher is better)



Swift vs. Objective-C

Program speed (higher is better)



SWIFT IS APPROX. 3.5 TIMES FASTER THAN OBJECTIVE-C

ISN'T IT?

iOS • 10 March 2016

Swift vs Obj-C Performance Comparison

Piotr Sochalewski

0 0 0 13 0

Is Swift faster than Objective-C? This question has been asked so many times and the answer is still unclear. So I took the recent Xcode 7.3 beta and ran some tests comparing Swift 2.2 and Objective-C. The results were surprising even for me.

I found many tests proving that Swift is faster than Objective-C and some saying old Obj-C is swifter than Swift. The truth is in the middle. Straight out in some cases Swift is much faster, but Obj-C still has some advantages.

AGENDA

1. DISPATCH
2. OBJECTS
3. PROTOCOLS
4. JUST DON'T DO IT

STATIC
VS
DYNAMIC

REDUCING DYNAMIC DISPATCH

- `final`
- `private`
- Whole Module Optimization

FINAL

```
class Awesome {

  var kitties: [UIImage]?

  func showRandomKitty(in imageView: UIImageView) {
    imageView.image = kitties?.random
  }
}
```

FINAL

```swift
class Awesome {

  final var kitties: [UIImage]?

  final func showRandomKitty(in imageView: UIImageView) {
    imageView.image = kitties?.random
  }
}
```

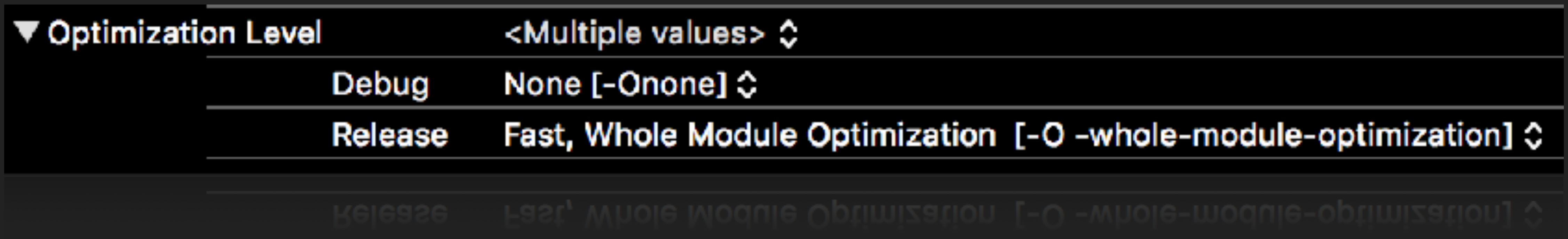
FINAL

```swift
final class Awesome {

   var kitties: [UIImage]?

   func showRandomKitty(in imageView: UIImageView) {
      imageView.image = kitties?.random
   }
}
```

PRIVATE

```
class Awesome {

  fileprivate var kitties: [UIImage]?                FINAL

  fileprivate func showRandomKitty(in imageView:  NON-FINAL
  UIImageView) {
    imageView.image = kitties?.random
  }
}


final class EvenMoreAwesome: Awesome {

  override func showRandomKitty(in imageView:        FINAL
  UIImageView) { … }
}
```

WHOLE MODULE OPTIMIZATION



▼ Optimization Level		<Multiple values>
	Debug	None [-Onone]
	Release	Fast, Whole Module Optimization [-O -whole-module-optimization]

SWIFT OBJECTS

- `class`
- `struct`

STRUCT VS CLASS

```
struct Circle {
  var center: CGPoint
  var radius: Double

  func draw() {}
}
```

STRUCT VS CLASS

```swift
struct Circle {
  var center: CGPoint
  var radius: Double

  func draw() {}
}
```

```swift
let circles = (0..<n).map { _ in
  Circle(center: .zero, radius: 1.0)
}

circles.forEach { $0.draw() }
```

⏱ **2 s**

STRUCT VS CLASS

```swift
final class Circle {
  var center: CGPoint
  var radius: Double

  init(center: CGPoint, radius: Double) {
    self.center = center
    self.radius = radius
  }
  func draw() {}
}
```

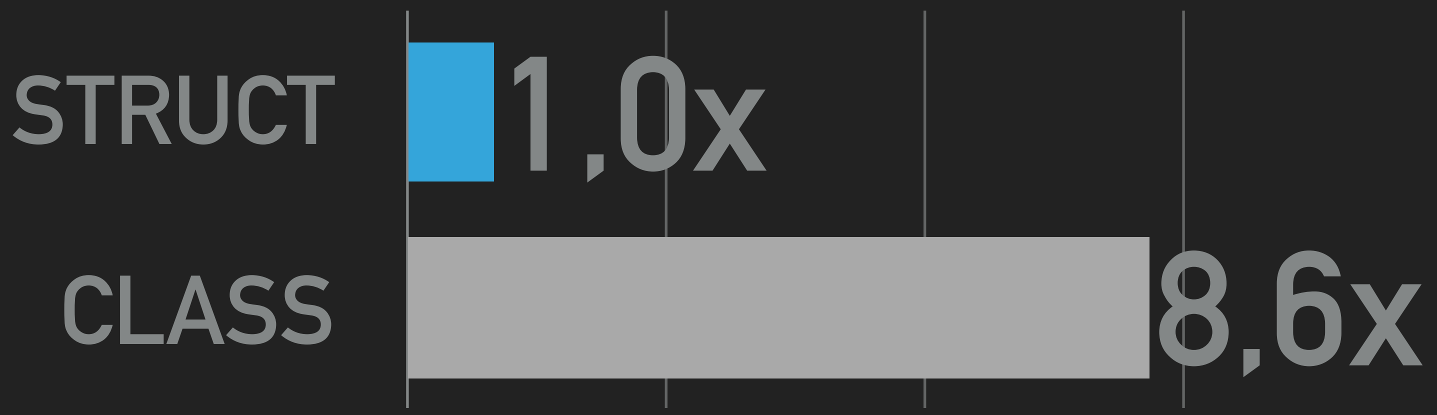
STRUCT VS CLASS

```
final class Circle {
  var center: CGPoint
  var radius: Double

  init(center: CGPoint, radius: Double) {
    self.center = center
    self.radius = radius
  }
  func draw() {}
}
```

```
let circles = (0..<n).map { _ in
  Circle(center: .zero, radius: 1.0)
}

circles.forEach { $0.draw() }
```

⏱ 17,3 s

STRUCT VS CLASS



Lower is better

STRUCT AND CLASS VS PROTOCOL CONFORMANCE

```swift
protocol Drawable {
  func draw()
}

struct Circle: Drawable {
  var center: CGPoint
  var radius: Double

  func draw() {}
}
```

STRUCT AND CLASS VS PROTOCOL CONFORMANCE

```
protocol Drawable {
  func draw()
}

struct Circle: Drawable {
  var center: CGPoint
  var radius: Double

  func draw() {}
}
```

```
let drawables: [Drawable] = (0..<n).map { _ in
  Circle(center: .zero, radius: 1.0)
}
drawables.forEach { $0.draw() }
```

⏱ **5,6 s**

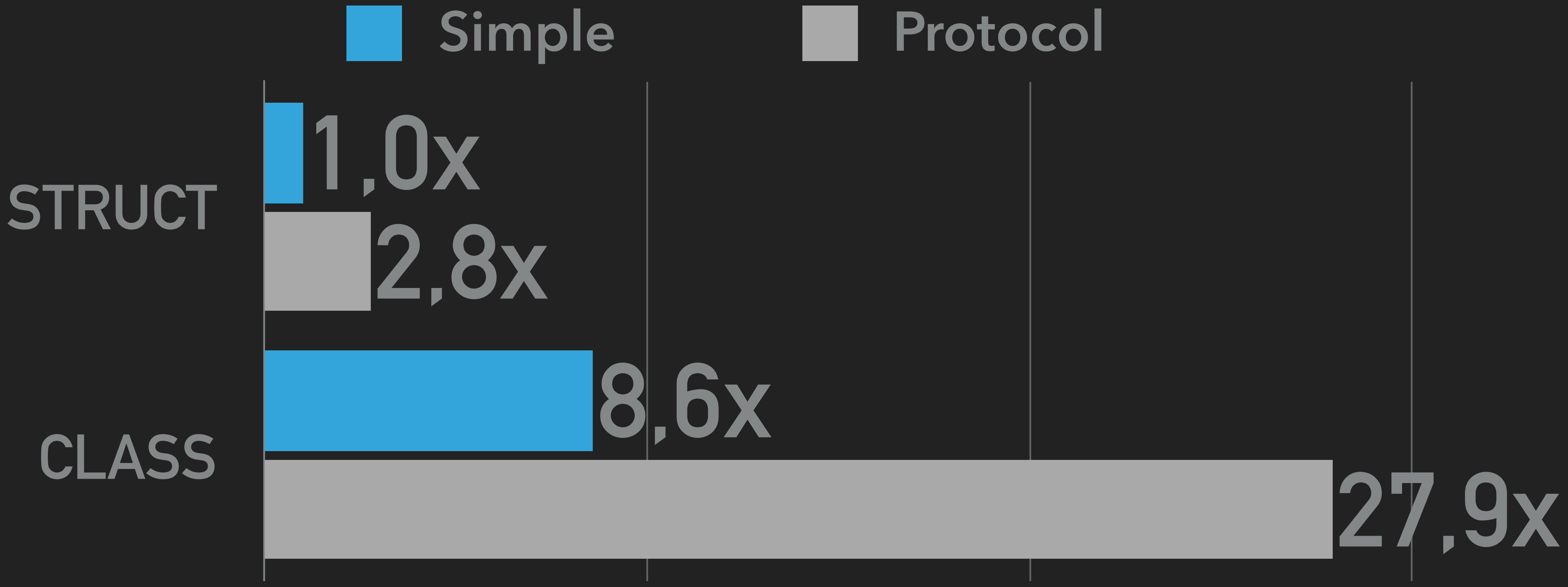
STRUCT AND CLASS VS PROTOCOL CONFORMANCE

```
protocol Drawable {
  func draw()
}

final class Circle: Drawable {
  var center: CGPoint
  var radius: Double

  init(center: CGPoint, radius: Double) {
    self.center = center
    self.radius = radius
  }
  func draw() {}
}
```

```
let drawables: [Drawable] = (0..<n).map { _ in
  Circle(center: .zero, radius: 1.0)
}
drawables.forEach { $0.draw() }
```

⏱ **56,2 s**

STRUCT AND CLASS VS PROTOCOL CONFORMANCE



Lower is better

PROTOCOLS

```swift
protocol SolarSystemable {
  func loadPartOfSolarSystem()
}

final class SunView: UIView, SolarSystemable {

  func loadPartOfSolarSystem() { … }
}
```

PROTOCOLS

```
protocol SolarSystemable: class {
  func loadPartOfSolarSystem()
}

final class SunView: UIView, SolarSystemable {

  func loadPartOfSolarSystem() { … }
}
```

⚠️ DANGEROUS OPTIMIZATION

```
&+, &-, &* // these operators don't do overflow checks

let x = UInt8.max // 255 🙋‍♀️
let y = x &+ 1 // 0 🙍‍♀️
```

ARC

```
func doSomethingFancy(withObject object: Object) {

  object.flipTheTable()

}
```

ARC

```
func doSomethingFancy(withObject object: Object) {
  __swift_retain(object)
  object.flipTheTable()
  __swift_release(object)
}
```

STOP ARC

STOP IT NOW

⚠️ DANGEROUS OPTIMIZATION

```
Unmanaged<T> // to avoid ARC calls

// Remember to have at least one strong reference
// to the object during execution
```

OPTIMIZATION WAYS

- Swift-only codebase
- Static dispatch whenever possible
- Enable Whole Module Optimization
- Use structs instead of classes
- Avoid Protocol-Oriented Programming when you need extremely high performance

THANK YOU

PIOTR SOCHALEWSKI

Droids On Roids

https://github.com/sochalewski

http://thedroidsonroids.com

Download on the App Store

Get **Smash the Code** 👍