MY TRANSITION FROM SWIFT TO KOTLIN

TWITTER: @ALLONSYKRAKEN BLOG: KRAKENDEV.IO

A LITTLE BIT ABOUT ME

TWITTER: @ALLONSYKRAKEN BLOG: KRAKENDEV.IO

- WRITE AND MAINTAIN A BLOG AT KRAKENDEV.IO THAT GETS CLOSE TO 10K VIEWS A WEEK
- SPOKEN ABOUT IOS/SWIFT IN 3 DIFFERENT CONTINENTS
- SR. IOS DEVELOPER FOR ALMOST 6 YEARS NOW
- I WORK AT TWITTER (TWEET. TWEET)


I WANTED MORE
But who cares?
No big deal...
TWITTER: @ALLONSYKRAKEN BLOG: KRAKENDEV.IO

SO I LOOKED AND I SEARCHED

AND ABOUT A YEAR AGO

TWITTER: @ALLONSYKRAKEN BLOG: KRAKENDEV.IO

BUT WE NEED YOU FOR THIS PROJECT

TWITTER: @ALLONSYKRAKEN BLOG: KRAKENDEV.IO

TWITTER: @ALLONSYKRAKEN BLOG: KRAKENDEV.IO

SWIFT: 0

KOTLIN: 0

DATA CLASSES

TWITTER: @ALLONSYKRAKEN BLOG: KRAKENDEV.IO

```
data class Conference(val name: String, val date: Date, val speakers: List<Speaker>)
val kotlinConf = Conference(name = "KotlinConf", date = Date(), speakers = listOf())
```

```
struct Conference {
    let name: String
    let date: Date
    let speakers: [Speaker]
}
let kotlinConf = Conference(name: "KotlinConf", date: Date(), speakers: [])
```

```
struct Conference {
    let name: String = "KotlinConf"
    let date: Date
    let speakers: [Speaker]
}
// This line throws a compiler error
🛑let kotlinConf = Conference(name: "BootlegKotlinConf", date: Date(), speakers: [])

// We have to initialize without the default value.
✅let kotlinConf = Conference(date: Date(), speakers: [])
```

SWIFT: 0
KOTLIN: 1

ENUMS

KOTLIN

```
enum class ColorSpace {
    CYMK,
    RGBA,
    HSBA
}
// LOVE this feature ❤️
val allColorSpaces = ColorSpace.allValues()
for (colorSpace in allColorSpaces) {
    print(colorSpace.name)
    print(colorSpace.ordinal)
}
```

SWIFT

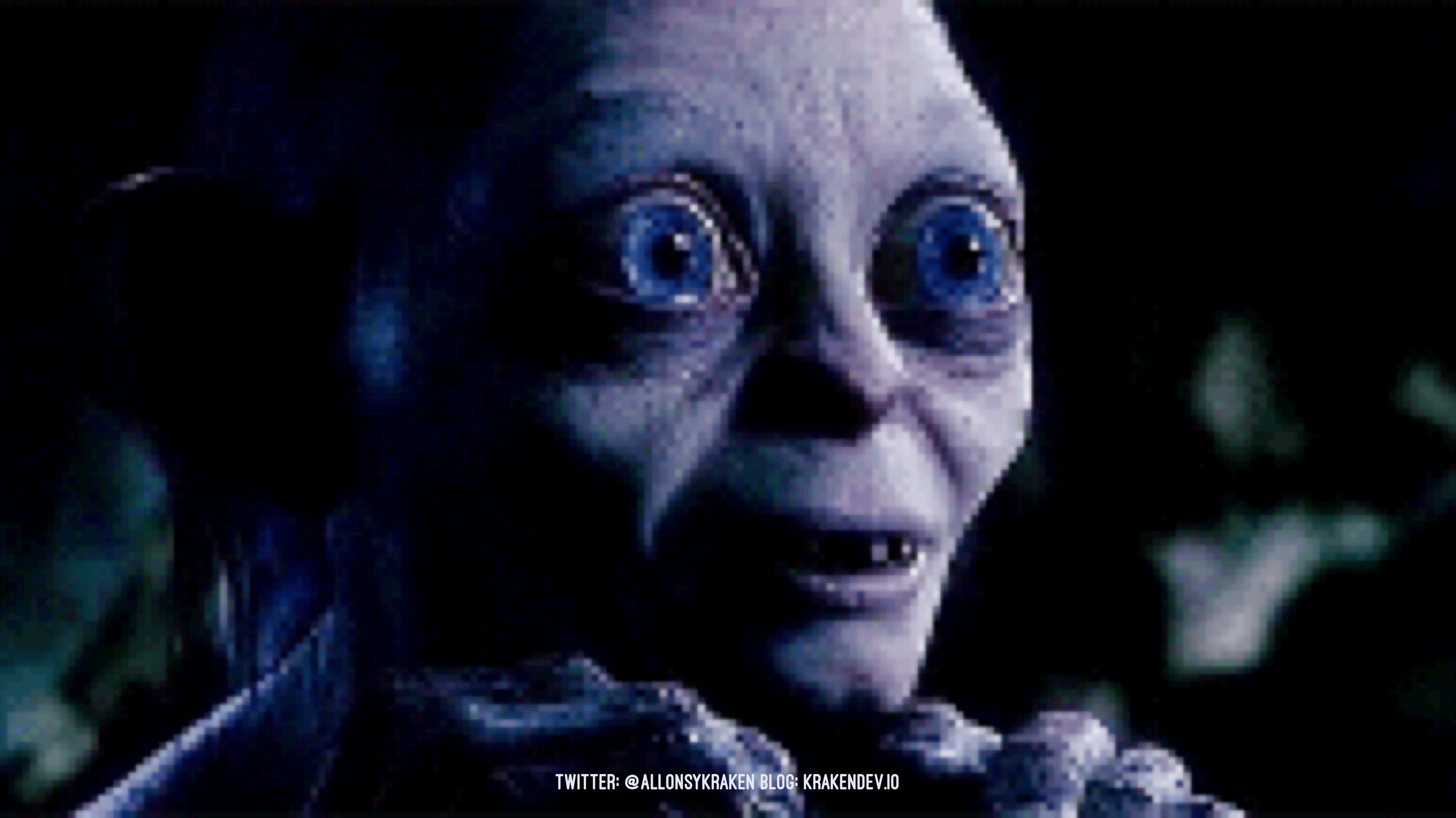
```
enum ColorSpace {
    // Love this feature more though ❤️
    case cymk(Float, Float, Float, Float),
    case rgb(Float, Float, Float),
    case hsb(Float, Float, Float)
}

switch colorSpace {
    case .cymk(let cyan, let yellow, let magenta, let black):
    case .rgb(let red, let green, let blue):
    case .hsb(let hue, let saturation, let brightness):
}
```

EXAMPLE: EXPRESSIONS

```
indirect enum Expr {
    case value(Double)
    case sum(Expr, Expr),

    var eval: Double {
        switch self {
        case .number(let value):
            return value
        case .sum(let lhs, let rhs):
            return lhs.eval + rhs.eval
    }
}
```



TWITTER: @ALLONSYKRAKEN BLOG: KRAKENDEV.IO

KOTLIN SEALED CLASSES

```
sealed class Expr
data class Const(val number: Double) : Expr()
data class Sum(val e1: Expr, val e2: Expr) : Expr()

fun eval(expr: Expr): Double = when(expr) {
    is Const -> expr.number
    is Sum -> eval(expr.e1) + eval(expr.e2)
}
```

SWIFT: 1
KOTLIN: 1

DELEGATED PROPERTIES & CLASSES

TWITTER: @ALLONSYKRAKEN BLOG: KRAKENDEV.IO

EXAMPLE: VIEW ASSIGNMENT

```
public class PersonView(context: Context) : LinearLayout(context) {
  val firstName: TextView by bindView(R.id.first_name)
}
```

SWIFT: 1
KOTLIN: 2

TUPLES

```swift
let person = (name: "Paul", age: 35)

func printPerson(person: (String, Int)) {
    print("\(person.0) is \(person.1) years old")
}
```

```
// Have to create an entire class
// to destructure into two variables, not one.
val (name, age) = Person()
```

SWIFT: 2

KOTLIN: 2

EASE OF LEARNING

SWIFT: 3

KOTLIN: 2

MEMORY MANAGEMENT

TWITTER: @ALLONSYKRAKEN BLOG: KRAKENDEV.IO

SWIFT: 4

KOTLIN: 2

COROUTINES

TWITTER: @ALLONSYKRAKEN BLOG: KRAKENDEV.IO

SWIFT: 4

KOTLIN: 3

TRUE TYPE ERASURE

TWITTER: @ALLONSYKRAKEN BLOG: KRAKENDEV.IO

SWIFT

```swift
protocol Pokemon {
    associatedType Type
    func attack() -> Type
}
```

KOTLIN

```kotlin
interface Pokemon<out Type> {
  fun attack(): Type
}
```

```
class PokemonTrainer {
    var favoritePokemon: Pokemon // Won't compile
}
```

```
struct AnyPokemon<P: Pokemon>: Pokemon {
    private let pokemon: P
    init(_ pokemon: P) {
        self.pokemon = pokemon
    }
    func attack() -> P.Type {
        return pokemon.attack()
    }
}

class PokemonTrainer {
    var favoritePokemon: AnyPokemon<Fire>
}

// Usage
trainer.favoritePokemon = AnyPokemon(Charmander())
```

GOOD OL' KOTLIN

```
data class Trainer(val favoritePokemon: Pokemon<Any>)
```

SWIFT: 4

KOTLIN: 4

INTEROPERABILITY

SWIFT: 4
KOTLIN: 5

VALUE/REFERENCE SEMANTICS

```
var developers = ["Hector", "Aaron", "Tre"]
for developer in developers {
    if !developer.isCool() {
        developers.remove(developer)
    }
}
print(developers) // Prints out "Hector"
```

```
// reference semantics
class Person {
    let name: String
}
// value semantics
struct Person {
    let name: String
}
```

FLEXIBILITY/CONTROL

SWIFT: 5

KOTLIN: 5

...And one more thing


SWIFT: 5
KOTLIN: 6

TWITTER: @ALLONSYKRAKEN BLOG: KRAKENDEV.IO