構文糖衣なしで
Swiftのオプショナルを
使うとどうなるか？

2016年1月9日

藤本尚邦 / Cocoa勉強会(関東) #76

自己紹介

- 藤本尚邦 (@fhisa)
- https://github.com/fhisa
- フリーランスプログラマー
- RubyCocoaフレームワーク原作者
- Mac開発歴、薄く長く約25年
- iOS開発歴、約1年

Agenda

- オプショナルおさらい
- 変数の宣言
- 構文糖衣なしでオプショナル
- オプショナル使いこなしの注意点
- まとめ

オプショナルおさらい

普通の型に"?"または"!"の１文字を付けて型宣言するとオプショナルになります。見た目はそれだけの違いですが、この２つは全く別の型です。

```
var absolutelyInt: Int // Int型

var maybeInt: Int?  // Optional<Int>型
var probablyInt: Int!  // 暗黙のOptional<Int>型
```

オプショナルおさらい

オプショナルはおおよそ以下のように定義された普通の enum です[1]:

```
enum Optional<Wrapped> {
    case None
    case Some(Wrapped)
}
```

[1] この発表で不要な情報は省いています

オプショナルおさらい

豊富な構文糖衣(Syntax Sugar)を持っている。

オプショナルが他のenumで定義された型と違うのはこの一点です。

```
var maybeArray: [Int]? // Optional Type
var probablyArray: [Int]! // Implicitly Unwrapped Optional Type

if let array = maybeArray { ... } else { ... } // Optional Binding
let x = maybeArray?.count  // Optional Chaining
let x = maybeArray!  // Forced Unwrapping
let x = probablyArray.count   // Implicitly Forced Unwrapping
let x = maybeArray ?? [1,2,3]  // Nil Coalescing Operator
```

構文糖衣なしでオプショナル

- 変数の宣言
- Optional Binding
- Optional Chaining
- Forced Unwrapping
- Nil Coalescing Operator

変数の宣言

```
var maybeInt: Int?
var maybeArray: [Int]?
```

Without Syntax-Sugar:

```
var maybeInt: Optional<Int>
var maybeArray: Optional<Array<Int>>
```

Optional Binding

```
if let array = maybeArray {
    IF-CLAUSE
} else {
    ELSE-CLAUSE
}
```

Without Syntax-Sugar:

```
switch maybeArray {
case .Some(let array):
    IF-CLAUSE
case .None:
    ELSE-CLAUSE
}
```

Optional Chaining

```
let x = maybeArray?.count  // Optional Chaining
```

Without Syntax-Sugar:

```
let x = ({ Void -> Optional<Int> in
    switch maybeArray {
    case .None: return .None
    case .Some(let array): return array.count
    }
})()
```

Forced Unwrapping

```
let x = maybeArray!   // Forced Unwrapping
```

Without Syntax-Sugar:

```
let x = ({ Void -> Int in
    switch maybeArray {
    case .None: fatalError("unexpectedly found nil ...")
    case .Some(let array): return array.count
    }
})()
```

Nil Coalescing Operator

```
let x = maybeArray ?? [1,2,3]  // Nil Coalescing Operator
```

Without Syntax-Sugar:

```
let x = ({ (arg:[Int]) -> [Int] in
    switch maybeArray {
    case .None: return arg
    case .Some(let array): return array
    }
})([1, 2, 3])
```

オプショナル使いこなしの注意点

- "?"付きで型宣言するのが基本
- "!"付き型宣言は明確な理由がなければ使わない
- 強制アンラップ・キャストは値がnilならバグのときのみ使う
- "?"や"??"を積極的に使うとコードの可読性が増す

```
if let delegate = delegate { delegate.smoeMethod() } // 冗長
delegate?.someMethod() // 簡潔・可読性良し
```

まとめ

- 普通の型とオプショナル型は見た目以上に異なる
- オプショナルはenumで定義された単なる型
- Swiftプログラミングではオプショナルが頻出
- 構文糖衣なしでのオプショナルのプログラムは苦痛
- だからたくさん構文糖衣があるんだよ
- オプショナルをきちんと理解して良いSwiftプログラムを書こう

参考文献

- The Swift Programming Language (Swift 2.1) https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift*Programming*Language/

Thank you!

2016年1月9日

藤本尚邦 / Cocoa勉強会(関東) #76