関数を引数として渡す書き方のポイント

加藤由訓 / @yoshikuni_kato

iOSDC 2017
2017/09/17

Who am I?

- 加藤由訓（Yoshikuni Kato）
  @yoshikuni_kato
- iOSエンジニア（2.5 years）
- Yahoo! Japan -> オハコ
- 「ラジへぇ」くん →
- Interests: 設計 / FRP / Coordinator Pattern / UI実装

functionalに書く傾向

- functionalに書くことが増えてきた
  - FRP (RxSwift / ReactiveSwift)
  - map / filter / reduce
- `if`や`for`が減る
- 宣言的に書ける

このLTでシェアすること

- 関数を引数として渡す場合の書き方
  → 少しの変更だけど、より宣言的に見える方法
- arrayのmapを例に

書き方１ - クロージャを直接書く

```
let array: [Int] = [1, 2, 3]

array.map { number -> Int in
    return number * 2
}
```

arrayのmapの定義

```
func map<T>(_ transform: (Element) throws -> T) rethrows -> [T]
```

- mapの引数：Elementを受け取ってTを返すクロージャ
- 関数は、名前付きのクロージャと捉えられる
- 関数自体を渡すことができる

書き方２ - 関数を渡す

```
// 先に関数を定義
func twoTimes(of number: Int) -> Int {
  return number * 2
}


let array: [Int] = [1, 2, 3]
array.map(twoTimes) // 関数を渡す
```

パラメーターが複数ある場合

```
func someFunc(a: Int, b: Int) -> String {
    return "a = \(a), b = \(b)"
}

let array: [Int] = [1, 2, 3]
array
    .map { number  -> (a: Int, b: Int) in
        return (a: number, b: number)  // 一旦タプルにする
    }
    .map(someFunc)
```

イニシャライザの場合

```
struct Sample {

    let number: Int

    init(number: Int) {
        self.number = number
    }
}
```

書き方１ - クロージャを直接書く

```
let array: [Int] = [1, 2, 3]

array.map { number -> Sample in
    return Sample(number: number)
}
```

書き方２ - 関数を渡す

```
let array: [Int] = [1, 2, 3]

array.map(Sample.init)
```

- イニシャライザ(.init) = そのObjectを返す関数

違い 1

```
array.map { number -> Sample in
    return Sample(number: number)
}

array.map(Sample.init)
```

違い 2

```
array
  .map { number -> Int in
      return number * 2
  }
  .map { number -> Sample in
    return Sample(number: number)
  }
  .map { sample -> Foo in
    return Foo(sample: sample)
  }

array
  .map(twoTimes)
  .map(Sample.init)
  .map(Foo.init)
```

まとめ

- 少し書き方を変えるだけで、より宣言的に書ける方法を紹介
- 活用例：ModelからViewModelへの変換

```
model.map(ViewModel.init)
```

- 結果的に、処理を関数に切り出していくことになる
- 関数を渡していく感覚（手続き型の感覚から徐々に離れる）

参考

- Connecting View Controllers, Swift Talk[1]
- From Runtime Programming to Functions, Swift Talk[2]

---

[1] https://talk.objc.io/episodes/S01E05-connecting-view-controllers

[2] https://talk.objc.io/episodes/S01E19-from-runtime-programming-to-functions

おまけ

```swift
func someFunc(a: Int, b: Int) -> String {
    return "a: \(a), b: \(b)"
}

// タプルは渡せない
let parameters = (a: 0, b: 0)
someFunc(parameters) // 🙅 (swift3~)


// mapだと渡せる
let array: [(Int, Int)] = [(0, 0)]
array.map(someFunc) // 🙆 (swift3でも)
```

- なぜ書けるのか分かっている人がいたら教えてください