Swift 5.1

2020/01/14

Kosuke Matsuda

- Default Arguments In Enum Cases
- Synthesize default values for the memberwise initializer
- Implicit returns from single-expression functions
- Warnings for ambiguous none cases
- Matching optional enums against non-optionals
- Expanding Swift Self to class members and value types
- Opaque Result Types
- Static and class subscripts
- Ordered Collection Diffing
- Key Path Member Lookup
- Property Wrappers
- ...

- Default Arguments In Enum Cases
- Synthesize default values for the memberwise initializer
- Implicit returns from single-expression functions
- Warnings for ambiguous none cases
- Matching optional enums against non-optionals
- Expanding Swift Self to class members and value types
- Opaque Result Types
- Static and class subscripts
- Ordered Collection Diffing
- Key Path Member Lookup
- Property Wrappers
- ...

Default Arguments In Enum Cases

- Swift 5.0以前ではAssociated Valuesは初期化時に値を設定する必要がある
- Swift5.1よりデフォルト値を設定することができるようになった
- ラベル引数を設定しないとコンパイルエラーになる

【Swift】Swift5.1からenumのAssociated Valuesにデフォルト値を設定できるようになった
(https://qiita.com/shiz/items/30fc7d3ff78fefbb3a40?utm_campaign=popular_items&utm_medium=feed&utm_source=popular_items)

```
enum Barcode {
    case upc(a: Int = 1, Int, Int, Int)
    case qrCode(String)
}

let productBarcode = Barcode.upc(85909, 51226, 3)
print(productBarcode) // upc(a: 1, 85909, 51226, 3)
```

Synthesize default values for the memberwise initializer

- Swiftの構造体（Struct）はプロパティにマッチするパラメータを受け入れるイニシャライザーを自動で生成してくれる（メンバワイズイニシャライザー）

- Swift 5.0以前ではデフォルト値の有無に関わらず全てのパラメータを受け入れる必要がある

- Swift 5.1ではデフォルトパラメータを使用するようになった

```swift
struct Dog {
    var age: Int = 0
    var name: String

//    init(age: Int = 0, name: String) {
//        self.age = age
//        self.name = name
//    }
}

let dogInSwift5_0 = Dog(age: 3, name: "Sparky")
print(dogInSwift5_0) // Dog(age: 3, name: "Sparky")
let dogInSwift5_1 = Dog(name: "Sparky")
print(dogInSwift5_1) // Dog(age: 0, name: "Sparky")
```

Implicit returns from single-expression functions

- 関数が単一式または単一の値から判断される構文であれば、returnを削除することができる

- なお、従来から値を返す単一行のクロージャーではreturnを省略できる

```swift
func sum(_ a: Int, _ b: Int) -> Int {
    a + b
}

let total = sum(5, 8)
print(total) // 13
```

Warnings for ambiguous none cases

- Swiftのoptionalは`some`と`none`という2つの状態がある

- `none`という状態を持つ独自のenumを作成し、それをoptionalでラップすると混乱を招くことになる

- Swift 5.1ではwarningが表示される

```swift
enum Theme {
    case none
    case light
    case dark
}

let theme1: Theme? = .none
print(theme1)
```

⚠ Assuming you mean 'Optional<Theme>.none'; did you mean '...

⚠ Expression implicitly coerced from 'Theme?' to 'Any'

Matching optional enums against non-optionals

- StringとIntegerにおけるoptionalと非optionalのswitch/caseによるパターンマッチングは適切に処理されるがenumに拡張されていない

- Swift 5.1ではenumのoptionalと非optionalでswitch/caseによるパターンマッチングを使用することができる

```swift
enum BuildStatus {
    case starting
    case inProgress
    case complete
}

let status: BuildStatus? = .inProgress
```

```swift
// Swift 5.0
switch status {
case .inProgress?:
    print("Build is starting…")
case .complete?:
    print("Build is complete!")
default:
    print("Some other build status")
}
```

```swift
// Swift 5.1
switch status {
case .inProgress:
    print("Build is starting…")
case .complete:
    print("Build is complete!")
default:
    print("Some other build status")
}
```

Expanding Swift Self to class members and value types

- `Self`キーワードは実際の具体的な型が不明なコンテキストで動的に参照できる

- Swift 5.1では`Self`のスコープがenum、struct、classなどの具象型も含むように拡張された

```swift
class NetworkManager {
    class var maximumActiveRequests: Int {
        return 4
    }

    func printDebugData() {
//        print("Maximum network requests: \(NetworkManager.maximumActiveRequests).")
        print("Maximum network requests: \(Self.maximumActiveRequests).")
    }
}

class ThrottledNetworkManager: NetworkManager {
    override class var maximumActiveRequests: Int {
        return 1
    }
}

let manager = ThrottledNetworkManager()
manager.printDebugData() // Maximum network requests: 1.
```

次回

- Opaque Result Types
- Static and class subscripts
- Ordered Collection Diffing
- Key Path Member Lookup
- Property Wrappers
- ...