SWIFT
Optional

Bill Kim(김정훈) | ibillkim@gmail.com

목차

Optional

Optional : “변수에는 값이 들어갈 수도 있고, 아닐 수도 있어(nil)”

아래와 같이 ? 및 ! 어노테이션(Annotation) 표시를 통하여 변수 및 상수 등에서 값의 존재를 나타낼 수 있습니다.

? : 값이 있을 수도 있고 없을 수도 있다(nil)

! : 값이 무조건 있다.

nil

Objective C 와 Swift에서의 nil은 아래와 같은 차이가 있습니다.

Objective C : 주소값(Pointer)가 존재하지 않는 경우
Swift : 값(Value)가 없는 경우

Wrapping

Optional 타입은 기본적으로 wrap되어 있는 상태입니다.

즉, Optional로 선언된 변수들은 값이 있는 것인지, nil인 것인지 wrap되어 있어서 모르는 상태입니다.

wrap된 상태에서는 설령 변수에 value값이 있다고 하더라도 바로 value가 출력되지 않습니다.(Swift 3.0에서 적용)

```swift
var optionalString: String? = "Hello"
print(optionalString) // Optional("Hello")
```

위의 코드에서는 optionalString이 nil일 수도 있기 때문에, 결과값 “Hello”가 출력되지 않고, Optional(“Hello”) 가 콘솔창에 출력됩니다.

Forced Unwrapping

앞선 예제에서처럼 출력 결과가 Optional(“Hello”)처럼 나오는 것은 대부분의 경우 원하는 출력값이 아닙니다.

이 때 올바른 출력을 위해 사용하는 것이 !(Forced Unwrapping(옵셔널 해제))입니다.

```swift
var optionalString: String? = "Hello"
print(optionalString!) // Hello
```

변수명 뒤의 느낌표는 Optional을 unwrap합니다. Optional은 unwrap된 상태에서만 값을 제대로 출력할 수 있습니다.

Forced Unwrapping

Unwrapping 관련 에러 예시

```swift
let value1: String! = nil
print(value1!) // 컴파일은 돼나 런타임 시 에러
```

위의 코드 예시에서 !(unwrap) 상태로 nil값을 넣을 수는 있지만 출력 시 !값으로 지정하여 출력하면 런타임 에러를 발생합니다.

Optional Binding

Optional Binding은 Optional 타입으로 선언된 변수에 값이 있는지 없는지를 확인할 수 있도록 해주는 기능입니다.

Optional Binding을 사용하면 느낌표 없이 Optional 타입의 변수 값을 출력할 수 있어서 좀 더 안전한 형태로 값을 얻을 수 있습니다.

기본 형태

```
if let 변수명 = Optional 변수 {
 // 임시 변수에 Optional 변수의 value값이 할당됩니다.
}
```

If let 대신에 if var 또한 사용 가능합니다.

Optional Binding

사용 예시

```swift
let myNumber1: Int? = 1234
let myNumber2: Int? = nil

if let actualNumber1 = myNumber1 {
    print("\(myNumber1)은 실제로 \(actualNumber1)입니다.")
} else { print("\(myNumber1)는 변환될 수 없습니다.") }
// 출력값 : Optional(1234)은 실제로 1234입니다.

if let actualNumber2 = myNumber2 {
    print("\(myNumber2)은 실제로 \(actualNumber2)입니다.")
} else { print("\(myNumber2)는 변환될 수 없습니다.") }
// 출력값 : nil는 변환될 수 없습니다.

print(actualNumber1) // actualNumber1는 로컬 변수로서 if let 문 밖에서 컴파일 에러를 낸다.
```

Optional Chaining

여러 객체를 혼합해서 사용하다보면 Optional끼리의 연산이 필요한 경우가 있습니다.

이 경우에 객체마다 옵셔널 바인딩을 사용하게 되면, if문이 꽤나 복잡해질 수 있습니다.

복잡한 옵셔널 바인딩을 Optional Chaining을 통해서 좀 더 간단하게 Optional 예외처리를 할 수 있도록 도와준다.

swift에서 . 을 통해 클래스의 프로퍼티에 접근이 가능합다는 점을 이용합니다.

Optional Chaining

Optional Chaining 미적용

```
class A {
    var b:B?
}

class B {
    var c:String?
}
```

```
// 객체와 변수 값에 대한 Optional Binding이 여러번 이루어집니다.
if let value = a.b
{
    print("\(a.b)은 실제로 \(value)입니다.")

    if let value2 = value.c
    {
       print("\(value.c)은 실제로 \(value2)입니다.")
    }
    else
    {
        print("\(value.c)는 변환될 수 없습니다.")
    }
}
else
{
    print("\(a.b)는 변환될 수 없습니다.")
}
```

Optional Chaining

Optional Chaining 적용

```
// Optional Chaining을 통하여 간편하게 Optional Binding을 처리합니다.
if((a.b?.c) != nil)
{
    // a.b?.c의 값이 nil이 아닐 경우 해당 if 문에 진입합니다.
}
```

```
var a2:A = A()

a2.b = B()
a2.b?.c = "aaa"

// 객체 및 변수 값이 제대로 할당될 경우 아래와 같이 값을 출력할 수 있습니다.
if ((a2.b?.c) != nil)
{
    print("\(a2.b?.c)은 실제로 \(a2.b?.c)입니다.")
    // Optional("aaa")은 실제로 Optional("aaa")입니다.
}
```

COALESCING

nil 병합 연산자(중위 연산자)로서 특정 변수의 값이 있으면 해당 변수값을 반환하고 없으면 원하는 값을 반환해주는 연산자입니다.

?? 으로 표현하며 기본 사용 방법은 아래와 같습니다.

```
var value1: Int?
var value2: Int? = 10

var a = value1 ?? 0
print(a) // 0, value1이 값이 없으므로(nil)으로 지정한 0을 반환

var b = value2 ?? 0
print(b) // 10, value2가 10으로 값이 있으므로 10을 반환
```

References

[1] The Swift Language Guide : https://jusung.gitbook.io/the-swift-language-guide/

[2] Swift Optional (1) : https://medium.com/@codenamehong/swift-optional-1-54ae4d37ee09

[3] swift3) Optional 개념 정리 : https://zeddios.tistory.com/16

[4] [Swift] Optional이란? 옵셔널 이해하기 : http://monibu1548.github.io/2018/05/12/swift-optional/

[5] 옵셔널 (Optional) : https://devxoul.gitbooks.io/ios-with-swift-in-40-hours/Chapter-2/optionals.html

References

[6] Generic Enumeration Optional : https://developer.apple.com/documentation/swift/optional

[7] [Swift] Optional과 Optional Binding(if let) : https://m.blog.naver.com/PostView.nhn?blogId=jdub7138&logNo=220374797671&proxyReferer=https%3A%2F%2Fwww.google.com%2F

[8] [Swift] Optional 사용 : https://slee2540.tistory.com/14

[9] [Swift]Optional Chaining 정리 : http://minsone.github.io/mac/ios/swift-optional-chaining-summary

[10] [Swift 3] 옵셔널 체이닝 (Optional Chaining) : https://beankhan.tistory.com/168

References

[11] Generic Enumeration Optional : https://jinunthing.tistory.com/47

Thank you!