공식문서 읽기 : Subscripts

Subscripts - The Swift Programming Language (Swift 5.5)

Classes, structures, and enumerations can define subscripts, which are shortcuts for accessing the member elements of a collection, list, or sequence. You use subscripts to set and retrieve values by index without needing separate methods for setting and retrieval. For example, you access elements in an

https://docs.swift.org/swift-book/LanguageGuide/Subscripts.html

Subscripts

클래스, 구조체, 열거형에서 Subscript를 정의할 수 있다. 이러한 서브 스크립트는 해당 타입의 요소에 접근하기 위해 사용된다. 수정과 검색을 위한 별도의 메서드 없이 index 값으로 수정과 검색을 할 수 있다. 예를 들어 Array는 someArray[index]로 Dictionary는 someDictionary[key]로 접근할 수 있다. 하나의 타입에 여러 개의 서브 스크립트를 정의할 수 있고, index이 유형에 따라 적절하게 서브 스크립트를 선택하게 된다.

Subscript Syntax

서브 스크립트를 사용하면 인스턴스의 이름 뒤에 []를 사용해 데이터에 접근할 수 있다. 이러한 문법은 인스턴스 메서드나 계산 프로퍼티의 문법과 비슷하다. 서브 스크립트를 정의하는 방법은 Subscript 키워드를 사용하여 하나 이상의 매개변수와 반환 값의 타입을 지정해야 한다. 이러한 서브 스크립트는 읽기-쓰기, 혹은 읽기 전용으로 정의되며 이는 계산 프로퍼티와 같이 getter, setter에 의해 전달된다.

```
subscript(index: Int) -> Int {
    get {
        // return an appropriate subscript value here
    }
    set(newValue) {
        // Perform a suitable setting action here
    }
}
```

위 코드와 같이 서브 스크립트를 정의하면 읽기-쓰기가 가능한 서브 스크립트이다. 위의 코드에서 newValue는 setter에서 default값으로 설정된 매개변수이기 때문에 이름을 다르게 써도 된다.

```
subscript(index: Int) -> Int {
     // return an appropriate subscript value here
}
```

위의 코드는 읽기만 가능한 서브스크립트를 정의하는 방법이다. 이럴때는 get 키워드를 사용하지 않고 정의ㅎ래도 된다.

실제 예를 살펴보자.

```
struct TimesTable {
    let multiplier: Int
    subscript(index: Int) -> Int {
        return multiplier * index
    }
}

let threeTimesTable = TimesTable(multiplier: 3)
print("six times three is \(threeTimesTable[6])")
// Prints "six times three is 18)"
```

위의 코드에서 새로운 인스턴스인 threeTimesTable에 선언된 서브 스크립트는 multiplier에 입력된 정수형 매개변수를 곱해준 값을 반환하는 기능을 가지고 있다.

Subscript Usage

이러한 서브스크립트는 어떤 상황에서 사용해야할까? 일반적으로는 컬렉션과 같이 연속적인 요소에 접근할 때 사용한다.

```
var numberOfLegs = ["Spider": 8, "ant": 6, "cat": 4]
numberOfLegs["bird"] = 2
```

위의 코드에서처럼 dictionary에서 서브 스크립트를 활용해 데이터를 추가하는 것을 볼 수있다. Swift의 Dictionary 타입에서는 서브 스크립트로 접근한 데이터는 옵셔널 타입이다. 만약 데이터를 삭제하고 싶다면 value 값에 nil을 할당하면 삭제된다.

Subscript Options

서브 스크립트는 여러개의 매개변수를 받을 수 있다. 이러한 매개변수는 어떤 타입이어도 상관이 없고 모든 타입의 값을 반환할 수도 있다. 함수에서와 마찬가지로 서브스크립트는 매개변수들에 default값을 줄 수 있고 Variadic 매개변수(여러 개의 매개변수를 사용하는 것)를 사용할 수 있다. 그러나 in-out 매개변수는 서브 스크립트에서 사용할 수 없다.

클래스와 구조체에서 필요한 만큼 서브 스크립트를 만들 수 있다. 이렇게 여러개가 선언되어 있을 때는 매개변수로 주어진 값들의 타입을 보고 어떤 서브 스크립트인지 추론하게 되는데, 이렇게 여러개의 서브 스크립트를 사용하는 것을 subscript oveloading이라고 부른다.

하나의 매개변수만 사용하는 것이 서브 스크립트를 사용하는 가장 일반적인 방법이지만 여러개의 매개변수를 선언해도 된다. 다음 예에서 여러개의 매개변수를 사용하는 서브 스크립트를 살펴보자.

```
struct Matrix {
    let rows: Int, columns: Int
    var grid: [Double]
    init(rows: Int, columns: Int) {
        self.rows = rows
        self.columns = columns
        grid = Array(repeating: 0.0, count: rows * columns)
    }
    func indexIsValid(row: Int, column: Int) -> Bool {
        return row >= 0 && row < rows && column >= 0 && column < columns
    }
    subscript(row: Int, column: Int) -> Double {
        get {
            assert(indexIsValid(row: row, column: column), "Index out of range")
            return grid[(row * columns) + column]
        }
        set {
            assert(indexIsValid(row: row, column: column), "Index out of range")
            grid[(row * columns) + column] = newValue
        }
    }
}

var matrix = Matrix(rows: 2, columns: 2)
matrix[0, 1] = 1.5
matrix[1, 0] = 3.2
```

Matrix 구조체의 subscript는 두 개의 매개변수를 가진다. 이렇게 두 개의 매개변수를 가진 서브 스크립트라도 [] 안에 매개변수를 써주면 해당 서브 스크립트가 실행된다. (1차원 배열을 subscript를 사용해 2차원 배열처럼 사용하는 방법)

Type Subscripts

서브 스크립트도 타입 자체로 접근할 수 있는 타입 서브 스크립트가 존재한다. 지금까지의 방식과 동일하게 static 키워드를 사용하면 정의할 수 있고 클래스에서는 class 키워드로 타입 서브스크립트를 정의할 수 있다.

```
var matrix = Matrix(rows: 2, columns: 2)
matrix[0, 1] = 1.5
matrix[1, 0] = 3.2

enum Planet: Int {
    case mercury = 1, venus, earth, mars, jupiter, saturn, uranus, neptune
    static subscript(n: Int) -> Planet {
        return Planet(rawValue: n)!
    }
}

let mars = Planet[4]
```