SWIFT
Functions

Bill Kim(김정훈) | ibillkim@gmail.com

목차

Defining and Calling Functions

함수를 선언할 때는 가장 앞에 func 키워드를 붙이고 (person: String) 파라미터와 형 그리고 -> String 형태로 반환형을 정의합니다.

```
func greet(person: String) -> String {
    let greeting = "Hello, " + person + "!"
    return greeting
}

print(greet(person: "Anna")) // Hello, Anna!
print(greet(person: "Brian")) // Hello, Brian!
```

Function Parameters and Return

```swift
// 파라미터가 없는 함수 (Functions Without Parameters)
func sayHelloWorld() -> String {
    return "hello, world"
}

// 복수의 파라미터를 사용하는 함수 (Functions With Multiple Parameters)
func greet(person: String, alreadyGreeted: Bool) -> String {
    if alreadyGreeted {
        return greetAgain(person: person)
    } else {
        return greet(person: person)
    }
}

// 반환 값이 없는 함수 (Functions Without Return Values)
func greet(person: String) {
    print("Hello, \(person)!")
}
```

Function Parameters and Return

```swift
// 복수의 값을 반환하는 함수 (Functions with Multiple Return Values)
func minMax(array: [Int]) -> (min: Int, max: Int) {
    var currentMin = array[0]
    var currentMax = array[0]
    for value in array[1..<array.count] {
        if value < currentMin {
            currentMin = value
        } else if value > currentMax {
            currentMax = value
        }
    }
    return (currentMin, currentMax) // 튜플을 활용하여 복수의 값을 반환
}

// 반환값에 ? 키워드를 붙여 옵셔널을 지정함
func minMax2(array: [Int]) -> (min: Int, max: Int)? {
    if array.isEmpty { return nil }

    var currentMin = array[0]
    var currentMax = array[0]
    for value in array[1..<array.count] {
        if value < currentMin {
            currentMin = value
        } else if value > currentMax {
            currentMax = value
        }
    }

    return (currentMin, currentMax)
}
```

Function Parameters and Return

함수 호출시 적절한 파라미터 이름을 지정해 함수 내부와 함수 호출시 사용할 수 있습니다.

```
func someFunction(firstParameterName: Int, secondParameterName: Int) {
    // 함수 내부에서 firstParameterName와 secondParameterName의 인자를 사용합니다.
}
someFunction(firstParameterName: 1, secondParameterName: 2)
```

Function Parameters and Return

파라미터 앞에 인자 라벨을 지정해 실제 함수 내부에서 해당 인자를 식별하기 위한 이름과 함수 호출시 사용하는 이름을 다르게 해서 사용할 수 있습니다.

```
func greet(person: String, from hometown: String) -> String {
    return "Hello \(person)!  Glad you could visit from \(hometown)."
}
print(greet(person: "Bill", from: "Cupertino"))
// Prints "Hello Bill!  Glad you could visit from Cupertino."
```

Function Parameters and Return

파라미터 앞에 _를 붙여 함수 **호출시 인자값을 생략**할 수 있습니다.

```swift
func someFunction(_ firstParameterName: Int, secondParameterName: Int) {
    // 함수 안에서 firstParameterName, secondParameterName
    // 인자로 입력받은 첫번째, 두번째 값을 참조합니다.
}

someFunction(1, secondParameterName: 2)
```

Function Parameters and Return

함수의 파라미터 값에 기본 값(: Int = 12)을 설정할 수 있습니다.
기본

값이 설정 되어 있는 파라미터는 함수 호출시 생략할 수 있습니다.
기본 값을 사용하지 않는 파라미터를 앞에 위치 시켜야 함수를 의미
있게 사용하기 쉽습니다.

```
func someFunction(parameterWithoutDefault: Int, parameterWithDefault: Int = 12) {
    // 함수 호출시 두번째 인자를 생략하면 함수안에서
    // parameterWithDefault값은 12가 기본 값으로 사용됩니다.
}

someFunction(parameterWithoutDefault: 3, parameterWithDefault: 6) //
parameterWithDefault는 6
someFunction(parameterWithoutDefault: 4) // parameterWithDefault는 12
```

Function Parameters and Return

인자 값으로 특정 형(type)의 집합 값을 사용할 수 있습니다.

```
func arithmeticMean(_ numbers: Double...) -> Double {
    var total: Double = 0
    for number in numbers {
        total += number
    }
    return total / Double(numbers.count)
}

arithmeticMean(1, 2, 3, 4, 5)
// returns 3.0, which is the arithmetic mean of these five numbers
arithmeticMean(3, 8.25, 18.75)
// returns 10.0, which is the arithmetic mean of these three numbers
```

Function Parameters and Return

인자 값을 직접 변경하는 파라미터를 선언할 수 있습니다. 선언을 위해 파라미터 앞에 inout 이라는 키워드를 사용합니다.

함수의 인자에 변수를 넣을때 & 키워드를 넣었습니다. C언어를 아시는 분은 inout 파라미터는 포인터를 넣는다고 생각하시면 이해하기 편하실 것입니다.

```
func swapTwoInts(_ a: inout Int, _ b: inout Int) {
     let temporaryA = a
     a = b
     b = temporaryA
}

var someInt = 3
var anotherInt = 107

swapTwoInts(&someInt, &anotherInt)
print("someInt is now \(someInt), and anotherInt is now \(anotherInt)")
// Prints "someInt is now 107, and anotherInt is now 3"
```

Function Types

함수 자체를 하나의 타입으로 사용할 수 있는 함수 형(타입)으로서 파라미터 형과(parameter types) 반환 형(return type)으로 두 개가 있습니다.

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

func multiplyTwoInts(_ a: Int, _ b: Int) -> Int {
    return a * b
}

// 입력 받는 파라미터와 반환 값이 없는 함수입니다.
func printHelloWorld() {
    print("hello, world")
}

// 함수를 변수처럼 정의해서 사용할 수 있습니다.
var mathFunction: (Int, Int) -> Int = addTwoInts
print("Result: \(mathFunction(2, 3))") // Result: 5
```

Function Types

```swift
// multiplyTwoInts 함수도 mathFunction과 함수 형이 같으므로 할당해 사용할 수 있습니다.
mathFunction = multiplyTwoInts
print("Result: \(mathFunction(2, 3))") // Result: 6

// 추론(Type Inferred)을 통하여 자동으로 함수를 할당할 수 있습니다.
let anotherMathFunction = addTwoInts
// anotherMathFunction is inferred to be of type (Int, Int) -> Int

// 파라미터에 함수 형을 사용할 수 있습니다.
func printMathResult(_ mathFunction: (Int, Int) -> Int, _ a: Int, _ b: Int) {
    print("Result: \(mathFunction(a, b))")
}
printMathResult(addTwoInts, 3, 5) // Result: 8
```

Function Types

함수를 반환하는 함수를 만들수도 있습니다.

```
func stepForward(_ input: Int) -> Int {
    return input + 1
}

func stepBackward(_ input: Int) -> Int {
    return input - 1
}

func chooseStepFunction(backward: Bool) -> (Int) -> Int {
    return backward ? stepBackward : stepForward
}

var currentValue = 3
let moveNearerToZero = chooseStepFunction(backward: currentValue > 0)
// moveNearerToZero는 이제 stepBackward() 함수를 가르키고 있습니다.
```

Nested Functions

함수 중에는 다른 함수 안의 body에서 동작하는 함수가 있는데 이 함수를 중첩 함수(Nested Function) 이라 합니다.

중첩함수는 함수 밖에서는 감춰져 있고 함수의 body내에서 접근 가능합니다.

```
func chooseStepFunction(backward: Bool) -> (Int) -> Int {
    func stepForward(input: Int) -> Int { return input + 1 }
    func stepBackward(input: Int) -> Int { return input - 1 }
    return backward ? stepBackward : stepForward
}

var currentValue = -4
let moveNearerToZero = chooseStepFunction(backward: currentValue > 0)
// moveNearerToZero는 이제 중첩 돼 있는 stepForward() 함수를 가르킵니다.

while currentValue != 0 {
    print("\(currentValue)... “)
    currentValue = moveNearerToZero(currentValue)
}

print(“zero!")

// -4...
// -3...
// -2...
// -1...
// zero!
```

References

[1] 함수 (Functions) : https://jusung.gitbook.io/the-swift-language-guide/language-guide/06-functions

[2] [Swift]Function 정리 : http://minsone.github.io/mac/ios/swift-function-summary

[3] [Swift 3] 함수 정의와 호출 및 함수 타입 (Function) : https://beankhan.tistory.com/157

[4] Swift - Functions : https://www.tutorialspoint.com/swift/swift_functions.htm

[5] Swift Function Parameters and Return Values : https://www.programiz.com/swift-programming/function-parameter-return-values

References

[6] Swift Functions : https://www.programiz.com/swift-programming/functions

[7] Functions In Swift Explained : https://learnappmaking.com/swift-functions-how-to/

[8] Functions : https://www.hackingwithswift.com/read/0/11/functions

[9] Swift - 함수(Function) : http://seorenn.blogspot.com/2014/06/swift-function.html

[10] Functions : https://wlaxhrl.tistory.com/39

Thank you!