스위프트 프로그래밍 문법 읽기 과제

스위프트 문법에 익숙해지기 위한 **스위프트 프로그래밍 문법(Swift Programming Language)** 문서 읽기 과제[1]입니다.

모든 문서를 한번에 다 읽는 것이 아니라, 아래에 적힌 순서대로 해당 챕터를 읽으면 됩니다.

효율적인 시간 활용과 각 개념의 중요도 표시를 위해 아래와 같이 네가지로 분류했습니다. 개념의 중요도/난이도와 본인의 학습 진도를 고려하여 읽으시기 바랍니다.

빨간색 매우 중요한 개념으로 유심히 읽어야함. 이해하는데 조금 어려울 수 있음
노란색 중요한 개념. 이해하는데 크게 어렵지 않음
초록색 중요한 개념. 기초적이고 기본적인 내용
회　색 당장 필요한 개념은 아니지만 읽어두면 추후 도움이 되는 내용으로, 후반에 다시 읽어야함

그리고 하나 더, 스위프트 API 디자인 가이드 문서

스위프트 API 디자인 가이드(Swift API Design Guidelines)도 반드시 함께 읽으시기 바랍니다.

- 스위프트에 익숙하지 않다면, 가이드 내용을 완전히 이해하기 어려울 수 있습니다. 하지만 좋은 스위프트 코드 작성을 위해 가이드 문서에 익숙해지는 것은 매우 중요합니다.

- 아래 각 과제를 마친 뒤, 문서를 반복적으로 읽으며 가이드에 익숙해지십시오.

* Swift API Design Guidelines 원문 : https://swift.org/documentation/api-design-guidelines/
* Swift API Design Guidelines 번역문[2] : https://github.com/connect-boostcamp/SwiftAPIDesignGuidelines

* 문서 정리 도움 : **전미정** 님

---

[1] 본 과제는 Stanford CS193(Developing iOS 10 Apps with Swift) 강의의 Reading Assignment를 참고하여 번역 및 재구성하였습니다.

[2] 스위프트 API 디자인 가이드 원문을 번역해 놓았습니다. 원문과 비교하며 읽으면 더욱 좋습니다.

첫 번째 읽어오기 과제

The Basics

Constants and Variables
Comments
Semicolons
Integers
Floating-Point Numbers
Type Safety and Type Inference
Numeric Literals
Numeric Type Conversion
Type Aliases
Booleans
Tuples
Optionals
Error Handling
Assertions

Basic Operators

Terminology
Assignment Operator
Arithmetic Operators
Compound Assignment Operators
Comparison Operators
Ternary Conditional Operator
Nill Coalescing Operator
Range Operators
Logical Operators

Strings and Characters

String Literals
Initializing an Empty String
String Mutability
Strings Are Value Types
Working with Characters
Concatenating Strings and Characters
String Interpolation
Unicode
Counting Characters
Accessing and Modifying a String
Comparing Strings
Unicode Representations of Strings

Collection Types

Mutability of Collections
Arrays
Sets
Performing Set Operations
Dictionaries

- 만약 배열에 들어있는 원소의 개수보다 더 높은 인덱스에 접근하려고하면 프로그램이 강제종료(crash)됩니다.
- 배열(Array)에는 마지막 원소를 꺼내오는 *last*라는 메서드가 있지만 이 메서드에서 반환하는 값은 *Optional*입니다. (배열이 비어있을 경우 nil이 반환 될 수 있기 때문입니다.)
- 배열에 사용되는 += 연산자는 배열 자체를 하나의 원소로 더하지 않고, 더해지는 배열의 원소들을 오른쪽에 추가합니다. (ex. var arr1 = [1, 2, 3], var arr2 = [4, 5, 6], arr1 += arr2 // [1,2, 3, 4, 5, 6])

Control Flow

For loops
While loops
Conditional Statements(switch)(Tuples, Value Binding, Where)
Control Transfer Statements(Labeled Statements)
Early Exit
Checking API Availability

- 스위프트에서의 *switch* 구문은 C 언어나 다른 언어에서 보다 훨씬 중요합니다.

Functions

Defining and Calling Functions
Function Parameters and Return Values(Functions with Multiple Return Values, Optional Tuple Return Types)
Function Argument Labels and Parameter Names(Variadic, In-Out Parameters)
Function Types
Nested Functions

Closures

Closure Expressions
Trailing Closures
Capturing Values
Closures are Reference Types
Non-escaping Closure
Autoclosures

- 스위프트에서 함수(전달인자와 반환 값 등을 가지는 종류의 것들)가 일급객체(혹은 일급시민)으로, 타입으로 사용될 수 있다는 것을 이해하는 것은 매우 중요합니다.
- 많은 iOS의 API가 클로저를 전달인자로 사용합니다.

Enumerations

Enumerations Sytax
Matching Enumeration Values with a Swift Statement
Associated Values
Raw Values
Recursive Enumerations

Classes and Structures(클래스와 구조체)

Comparing Classes and Structures
Structures and Enumerations are Value Types
Classes are Reference Types
Choosing Between Classes and Structures
Assignment and Copy Behavior for Strings, Arrays, and Dictionaries

- 스위프트에서 클래스와 구조체는 *initializers, functions*, *properties* 부분에서 매우 유사합니다. 하지만 이 둘은 **값 타입 vs 참조 타입**이라는 중요한 차이를 지니고 있으며, 해당 챕터에서 강조하고있으니 유심히 읽어보시기 바랍니다.

Properties

Stored Properties(Lazy Stored Properties, Stored Properties and Instance Variables)
Computed Properties
Property Observers
Global and Local Variables
Type Properties