

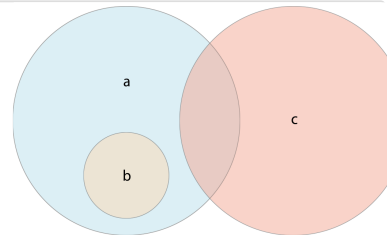


공식문서 읽기 : Collection Type

Collection Types - The Swift Programming Language (Swift 5.5)


Swift provides three primary collection types, known as arrays, sets, and dictionaries, for storing collections of values. Arrays are ordered collections of values. Sets are unordered collections of unique values.

<https://docs.swift.org/swift-book/LanguageGuide/CollectionTypes.html>



아래는 번역된 페이지입니다.

컬렉션 타입 (Collection Types)

 <https://jusung.gitbook.io/the-swift-language-guide/language-guide/04-collection-types>



Swift에서는 컬렉션 타입으로 Array, Set, Dictionary 세 가지를 지원한다.

컬렉션의 변경

변수(var)에 할당하면 컬렉션은 변경이 가능하고 상수(let)에 할당하면 변경이 불가능하다

배열

배열의 축약형 문법

배열의 타입은 Array로 적을 수 있는데 축약형으로 [Element] 형태로 사용할 수 있다.

빈 배열의 생성

```
var someInts = [Int]()
someInts = []
```

기본 값으로 빈 배열 생성

```
var threeDoubles = Array(repeating: 0.0, count: 3)
```

다른 배열을 추가한 배열의 생성

'+' 연산자를 이용해 배열을 합칠 수 있다.

```
var anotherThreeDoubles = Array(repeating: 2.5, count: 3)

var sixDoubles = threeDoubles + anotherThreeDoubles
// [0.0, 0.0, 0.0, 2.5, 2.5, 2.5]
```

리터럴을 이용한 배열의 생성

[value1, value2, value3] 형태를 사용해 배열을 생성할 수 있다.

```
var shoppingList: [String] = ["Eggs", "Milk"]
```

배열의 접근 및 변환

배열의 원소 개수: array.count

배열이 비었는지: array.isEmpty

배열에 원소 추가: array.append(element)

배열의 특정 위치의 원소에 접근: array[0], array[1..3]

특정 위치에 원소 추가/삭제/접근

```
array.insert(element, at: 0)
array.remove(at:0)
array.removeLast()
```

배열의 순회

for-in loop을 이용해 배열을 순회할 수 있다.

```
for item in shoppingList {
    print(item)
}
```

배열의 값과 index가 필요할 때는 enumerated() 메소드를 사용한다.

```
for (index, item) in shoppingList.enumerated() {
    print("Item \(index+1): \(value)"
}
```

Set

Set 형태로 저장되기 위해서는 반드시 타입이 hashable이어야만 한다.

String, Int, Double, Bool같은 기본 타입은 기본적으로 hashable이다.

빈 Set 생성

```
var letters = Set<Character>()
```

배열 리터럴을 이용한 Set 생성

```
var favoriteGenres: Set<String> = ["Rock", "Classical", "Hip Hop"]
// 타입을 추론할 수 있으므로 Set<String>을 Set으로 작성해도 된다
```

Set 접근과 변경

Set의 원소 개수: set.count

Set이 비었는지: set.isEmpty

Set에 원소 추가: set.insert(element)

Set의 원소 삭제: set.remove(element)

Set가 원소를 가지고 있는지: `set.contains(element)`

Set의 순회

for-in loop을 이용해 set을 순회할 수 있다.

```
for genre in favoriteGenres {  
    print(genre)  
}
```

Set의 명령

```
// 교집합  
a.intersection(b)  
// 합집합-교집합  
a.symmetricDifference(b)  
// 합집합  
a.union(b)  
// 차집합 (a-b)  
a.subtracting(b)
```

Set의 멤버십과 동등비교

Set의 동등비교와 멤버 여부를 확인하기 위해 각각 `==` 연산자와 `isSuperset(of:)`, `isStrictSubset(of)`, `isStrictSuperset(of:)`, `isDisjoint(with:)` 메소드를 사용한다.

```
// 공통값이 없는 경우 return true  
a.isDisjoint(with:b)
```

Dictionary

축약형 Dictionary

[Key: Value] 형태로 Dictionary를 선언해 사용할 수 있다.

빈 Dictionary 생성

```
var namesOfIntegers = [Int: String]()  
namesOfIntegers = [:]()
```

리터럴을 이용한 Dictionary의 생성

[key1: value1., key2: value2, key3: value3] 형태로 Dictionary를 선언할 수 있다.

```
var airports: [String: String] = ["YYZ": "Toronto Pearson", "DUB": "Dublin"]
```

Dictionary의 접근과 변경

Dictionary의 원소 개수: dict.count

Dictionary이 비었는지: dict.isEmpty

Dictionary에 값 할당

```
airports["LHR"] = "Londen"
```