

# SWIFT 빠른 참조

타입, 옵셔널, 프로토콜, 에러 처리 핵심

## 기본

```
Hello World
import Foundation
print("Hello, World!")
```

## 상수 및 변수

```
let name = "Swift" // 상수 (불변)
var count = 0 // 변수 (가변)
count += 1
let pi: Double = 3.14 // 명시적 타입 어노테이션
```

## 주석

```
// 한 줄 주석
// 여러 줄 주석
/* 주석 */
/// 문서화 주석 (Markdown 지원)
```

## 타입

```
기본 타입
Int 플랫폼 크기 정수 (현대 시스템에서 64비트)
Double 64비트 부동소수점 (Float보다 선호)
Float 32비트 부동소수점
Bool true / false
String 유니코드 문자열 값 타입
Character 단일 확장 자소 클래스터
```

## 타입 추론 및 변환

```
let score = 95 // Int로 추론
let gpa = 3.8 // Double로 추론
let total = Double(score) + gpa // 명시적 변환
let label = "Score: \(score)" // 문자열 보간
```

## 튜플

```
let point = (x: 3, y: 5)
print(point.x)
let (x, y) = point // 분해
let (first, _) = point // 두 번째 값 무시
```

## 타입 별칭

```
 typealias Coordinate = (Double, Double)
let origin: Coordinate = (0.0, 0.0)
```

## 제어 흐름

```
If / Else
if score > 90 { print("A") }
else if score > 80 { print("B") }
else { print("C") }
```

## Switch

```
switch grade {
case "A": print("excellent")
case "B", "C": print("passing")
default: print("unknown")
}
```

## 반복문

```
for i in 0..5 { // 반개방 범위
for name in names { // 컬렉션
for (i, val) in list.enumerated() {
while condition {
repeat { } while condition // do-while
}
```

## Guard

```
func process(value: Int?) {
guard let v = value, v > 0 else { return }
print(v) // v는 연쇄됨되어 범위에 있음
}
```

## 함수

### 기본 함수

```
func greet(name: String) -> String {
return "Hello, \(name)!"
}
greet(name: "Alice")
```

### 인자 레이블

```
func move(from start: Int, to end: Int) -> Int {
return end - start
}
move(from: 0, to: 10) // 외부 레이블
func add(_ a: Int, _ b: Int) -> Int { a + b }
```

### 기본 및 가변 매개변수

```
func join(_ items: String..., separator: String = ", ") -> String {
items.joined(separator: separator)
}
join("a", "b", "c")
```

### inout 매개변수

```
func double(_ x: inout Int) { x *= 2 }
var num = 5
double(&num) // num은 이제 10
```

## 클로저

### 클로저 문법

```
let double = { (x: Int) -> Int in return x * 2 }
let nums = [3, 1, 2]
let sorted = nums.sorted { $0 < $1 }
let mapped = nums.map { $0 * 10 }
```

### 후행 클로저

```
UIView.animate(withDuration: 0.3) {
view.alpha = 0.0
}
```

### 값 캡처

```
func makeCounter() -> () -> Int {
var count = 0
return { count += 1; return count }
}
let counter = makeCounter() // counter() => 1, 2, ...
```

## 클래스 및 구조체

### 구조체 (값 타입)

```
struct Point {
var x: Double
var y: Double
}
var p = Point(x: 1, y: 2) // 자동 멤버별 초기화
```

### 클래스 (참조 타입)

```
class Vehicle {
var speed: Double = 0
init(speed: Double) { self.speed = speed }
}
class Car: Vehicle { var gear: Int = 1 }
```

### 구조체 vs 클래스

```
struct 값 타입, 할당 시 복사, 상속 없음
class 참조 타입, 참조로 공유, 상속 지원
mutating `self`를 수정하는 구조체 메서드에 필요한 키워드
deinit 클래스 전용 소멸자 (할당 해제 전 호출)
```

## 프린팅

### 정의 및 채택

```
protocol Drawable {
var description: String { get }
func draw()
}
struct Circle: Drawable { /* 필수 멤버 구현 */ }
```

### 프로토콜 확장

```
extension Drawable {
func log() { print("Drawing: \(description)") }
}
// 모든 Drawable 채택자가 log()를 무료로 사용 가능
```

### 일반 프로토콜

```
Equatable `==` 및 `!=` 비교
Comparable `<`, `>`, `<=`, `>=` 순서
Hashable Dictionary 키 또는 Set에서 사용 가능
Codable Encodable + Decodable (JSON, plist)
CustomStringConvertible 커스텀 `description` 속성
Identifiable `id` 속성 필요 (SwiftUI)
```

## 옵셔널

### 옵셔널 선언

```
var name: String? = "Alice" // String 또는 nil 포함 가능
var age: Int? = nil // 현재 nil
let count: Int = 5 // 비옵셔널, nil 불가
```

### 언래핑

```
if let n = name { print(n) } // 옵셔널 바인딩
guard let n = name else { return } // guard
let n = name ?? "Unknown" // nil 합체
let n = name! // 강제 언래핑 (nil이면 충돌)
```

### 옵셔널 체이닝

```
let count = user?.address?.zip?.count
// 제인의 어떤 링크가 nil이면 nil 반환
user?.save() // 사용자가 non-nil일 때만 호출
```

### 옵셔널 Map

```
let length = name.map { $0.count } // Int?
let upper = name.flatMap { $0.isEmpty ? nil : $0.uppercased() }
```

## 열거형

### 기본 열거형

```
enum Direction {
case north, south, east, west
}
var heading = Direction.north
heading = .east // 타입 추론
```

### 연관 값

```
enum Result {
case success(data: String)
case failure(code: Int, message: String)
}
if case .failure(let code, _) = r { print(code) }
```

### 원시 값

```
enum Planet: Int {
case mercury = 1, venus, earth, mars
}
let p = Planet(rawValue: 3) // Optional(.earth)
print(Planet.earth.rawValue) // 3
```

### 열거형의 메서드

```
enum Suit: String, CaseIterable {
case hearts, diamonds, clubs, spades
}
Suit.allCases.forEach { print($0.rawValue) }
```

## 에러 처리

### 에러 정의

```
enum NetworkError: Error {
case badURL
case timeout(seconds: Int)
case serverError(code: Int)
}
```

### 던지기 및 잡기

```
func fetch(url: String) throws -> Data {
guard url.hasPrefix("https") else { throw
NetworkError.badURL }
return Data()
}
do { let data = try fetch(url: "https://example.com") }
catch { print("Error: \(error)") }
```

### try 변형

```
try `do-catch` 내부에 있어야 하며, 에러 전파
try? 옵셔널 반환, 에러 시 `nil`
try! 강제 try, 에러 시 충돌
throws 함수가 에러를 던질 수 있음
```

**(rethrows)** 클로저 인자가 던질 때만 던짐