

Referensi Cepat Swift

Tipe, optional, protocol, penanganan error — esensial

Dasar

Hello World

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

Konstanta & Variabel

```
let name = "Swift" // constant (immutable)
var count = 0 // variable (mutable)
count += 1
let pi: Double = 3.14 // explicit type annotation
```

Komentar

```
// single-line comment
/* multi-line
comment */
/// documentation comment (Markdown supported)
```

Tipe Data

Tipe Dasar

Int	Integer berukuran platform (64-bit di sistem modern)
Double	Floating point 64-bit (lebih diutamakan dari Float)
Float	Floating point 32-bit
Bool	true / false
String	String Unicode, value type
Character	Satu extended grapheme cluster

Inferensi Tipe & Konversi

```
let score = 95 // inferred as Int
let gpa = 3.8 // inferred as Double
let total = Double(score) + gpa // explicit conversion
let label = "Score: \(score)" // string interpolation
```

Tuple

```
let point = (x: 3, y: 5)
print(point.x) // named access
let (x, y) = point // decompose
let (first, _) = point // ignore second value
```

Type Alias

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

Alur Kontrol

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")
}
```

Loop

```
for i in 0..<5 { // half-open range
for name in names { // collection
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) // v is unwrapped and in scope
}
```

Fungsi

Fungsi Dasar

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

Label Argumen

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

Parameter Default & Variadic

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

Parameter inout

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

Closure

Sintaks Closure

```
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 }
```

Trailing Closure

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

Menangkap Nilai

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

Class & Struct

Struct (Value Type)

```
struct Point {
var x: Double
var y: Double
}
var p = Point(x: 1, y: 2) // auto memberwise init
```

Class (Reference Type)

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

Struct vs Class

struct	Value type, disalin saat penugasan, tanpa inheritance
class	Reference type, dibagikan via referensi, mendukung inheritance
mutating deinit	Keyword wajib untuk metode struct yang memodifikasi self Deinitializer khusus class (dipanggil sebelum dealokasi)

Protocol

Definisi & Konformitas

```
protocol Drawable {
var description: String { get }
func draw()
}
struct Circle: Drawable { /* implement required members */ }
```

Protocol Extension

```
extension Drawable {
func log() { print("Drawing: \(description)") }
}
// all Drawable conformers get log() for free
```

Protocol Umum

Equatable	Perbandingan == dan !=
Comparable	Pengurutan <, >, <=, >=
Hashable	Dapat digunakan sebagai kunci Dictionary atau dalam Set
Codable	Encodable + Decodable (JSON, Plist)
CustomStringConvertible	Properti description kustom
Identifiable	Membutuhkan properti id (SwiftUI)

Optional

Deklarasi Optional

```
var name: String? = "Alice" // may contain String or nil
var age: Int? = nil // currently nil
let count: Int = 5 // non-optional, never nil
```

Unwrapping

```
if let n = name { print(n) } // optional binding
guard let n = name else { return } // guard
let n = name ?? "Unknown" // nil coalescing
let n = name! // force unwrap (crashes if nil)
```

Optional Chaining

```
let count = user?.address?.zip?.count
// returns nil if any link in the chain is nil
user?.save() // called only if user is non-nil
```

Optional Map

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

Enum

Enum Dasar

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

Referensi Cepat Swift

Associated Values

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

Raw Values

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

Metode pada Enum

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

Penanganan Error

Mendefinisikan Error

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

Melempar & Menangkap

```
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)") }
```

Varian try

try	Harus dalam do-catch , meneruskan error
try?	Mengembalikan optional, nil jika error
try!	Force try, crash jika error
throws	Fungsi dapat melempar error
rethrows	Melempar hanya jika argumen closure melempar