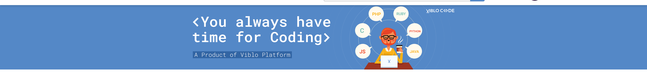

Nguyễn Quốc Tinh @tinhnq Theo dõi
854 38 21

Đã đăng vào thg 3 14, 2018 1:25 PM · 1 phút đọc

[iOS] [Swift] Tổng hợp tất cả từ khóa trong ngôn ngữ Swift (Part 3)

Bài đăng này đã được cập nhật cách đây 3 năm rồi nên nó có thể không còn được cập nhật lên nữa.

Part 1: https://viblo.asia/p/ios-swift-tong-hop-tat-ca-tu-khoa-trong-ngon-ngu-swift-part-1-E375zEAdlGW

Part 2: https://viblo.asia/p/ios-swift-tong-hop-tat-ca-tu-khoa-trong-ngon-ngu-swift-part-2-naQZRwrvlvx

Expressions and Types Keywords

Any : đại diện cho bất kỳ kiểu nào của đối tượng, bao gồm cả hàm.

```
var anything = [Any]()
anything.append("Any Swift type can be added")
anything.append(0)
anything.append({(foo: String) -> String in "Passed in \(foo)"})
```

as : dùng để ép kiểu để có thể truy xuất được thuộc tính hoặc phương thức của kiểu đó. Có thể ép đúng kiểu hoặc sai kiểu.

```
var anything = [Any]()
anything.append("Any Swift type can be added")
anything.append(0)
anything.append({(foo: String) -> String in "Passed in \(foo)" })
let intInstance = anything[1] as? Int

hoặc

var anything = [Any]()
anything.append("Any Swift type can be added")
anything.append(0)
anything.append({(foo: String) -> String in "Passed in \(foo)" })
for thing in anything {
    switch thing {
    case 0 as Int:
        print("It's zero and an Int type")
    case let someInt as Int:
        print("It's an Int that's not zero but \(someInt)")
    default:
        print("Who knows what it is")
    }
}
```

false : biến kiểu Bool, ko phải là true.

```
let alwaysFalse = false
let alwaysTrue = true
if alwaysFalse { print("Won't print, alwaysFalse is false 😺")}
```

catch : nếu trong mệnh đề clause xảy ra lỗi, thì catch sẽ xử lý lỗi đó, ta có thể catch nhiều trường hợp khác nhau như trong ví dụ dưới.

```
do {
    try haveAWeekend(4)
} catch WeekendError.Overtime(let hoursWorked) {
    print("You worked \(hoursWorked) more than you should have")
} catch WeekendError.WorkAllWeekend {
    print("You worked 48 hours :-0")
} catch {
    print("Gulping the weekend exception")
}
```

is : kiểm tra xem có phải loại subclass nào đó hay không.

```
class Person {}
class Programmer : Person {}
class Nurse : Person {}
let people = [Programmer(), Nurse()]
for aPerson in people {
    if aPerson is Programmer {
        print("This person is a dev")
    } else if aPerson is Nurse {
        print("This person is a nurse")
    }
}
```

nil : Represents a stateless value for any type in Swift. *Different from Objective-C's nil, which is a pointer to a nonexistent object. (chỗ này mình không dịch)

```
class Person{}
struct Place{}
//Literally any Swift type or instance can be nil
var statelessPerson: Person? = nil
var statelessPlace: Place? = nil
var statelessInt: Int? = nil
var statelessString: String? = nil
```

rethrows : 1 hàm ném ra 1 error nếu 1 tham số trong hàm ném ra 1 error.

```
func networkCall(onComplete:() throws -> Void) rethrows {
    do {
        try onComplete()
    } catch {
        throw SomeError.error
    }
}
```

super : là một biến tham chiếu mà được sử dụng để tham chiếu đến đối tượng lớp cha gần nhất.

```
class Person {
    func printName() {
        print("Printing a name. ")
    }
}
class Programmer : Person {
    override func printName() {
        super.printName() // Super là Person đó
        print("Hello World!")
    }
}
let aDev = Programmer()
aDev.printName() //"Printing a name, Hello World!"
```

self : là thực thể của loại mà mình đang xài đó (có thể là class, struct hoặc enum)

```
class Person {
    func printSelf() {
        print("This is me: \(self)")
    }
}
let aPerson = Person()
aPerson.printSelf() //"This is me: Person"
```

Self : Xài trong protocol, Self chính là class, struct hoặc enum nào đó conform protocol kia.

```
protocol Printable {
    func printTypeTwice(otherMe:Self)
}
struct Foo : Printable {
    func printTypeTwice(otherMe: Foo) {
        print("I am me plus \(otherMe)")
    }
}
let aFoo = Foo()
let anotherFoo = Foo()
aFoo.printTypeTwice(otherMe: anotherFoo) //I am me plus Foo()
```

throw : ném ra 1 error từ 1 hàm.

```
enum WeekendError: Error {
    case Overtime
    case WorkAllWeekend
}
func workOvertime () throws {
    throw WeekendError.Overtime
}
```

throws : Chỉ ra rằng 1 hàm có thể ném ra 1 lỗi nào đó.

```
enum WeekendError: Error {
    case Overtime
    case WorkAllWeekend
}
func workOvertime () throws {
    throw WeekendError.Overtime
}
//"throws" indicates in the function's signature that I need use try, try? or try!
try workOvertime()
```

true : giá trị true trong kiểu Bool.

```
let alwaysFalse = false
let alwaysTrue = true
if alwaysTrue { print("Always prints")}
```

try : Khi gọi hàm thì ta phải thêm try hoặc try? hoặc try! trước hàm đó.

```
let aResult = try dangerousFunction() //có lỗi là phải catch lại
let aResult = try! dangerousFunction() //có lỗi là crash luôn
if let aResult = try? dangerousFunction() //Unwrap 1 giá trị optional từ hàm này trả về
```

iOS Swift

Bài viết liên quan
Generic Protocols with Associated Type
Phan Huynh Thien An
4 phút đọc
1421 0 1 9
[iOS] [Swift] Tổng hợp tất cả từ khóa trong ngôn ngữ Swi...
Nguyễn Quốc Tinh
5 phút đọc
2851 7 2 5
3 lỗi nghiêm trọng của lập trình viên iOS
Phuong VNC
13 phút đọc
1052 5 0 6
[iOS] [Swift] Tổng hợp tất cả từ khóa trong ngôn ngữ Swi...
Nguyễn Quốc Tinh
3 phút đọc
726 5 0 6
Bài viết khác từ Nguyễn Quốc Tinh
(SwiftUI) GridStack - layout lưới trong vài dòng code
Nguyễn Quốc Tinh
4 phút đọc
252 0 0 2
Empty Strings in Swift
Nguyễn Quốc Tinh
2 phút đọc
250 1 0 1
Mở rộng ứng dụng iOS bằng module
Nguyễn Quốc Tinh
13 phút đọc
485 5 0 3
Sự hữu dụng của type alias trong Swift
Nguyễn Quốc Tinh
9 phút đọc
1286 2 0 4

Bình luận

Viết Xem trước

Viết phản hồi...

Bình luận

Chưa có bình luận nào.