Thachnd @Thachnd Theo dõi
27 0 23

Đã đăng vào Thứ Hai, 2:12 PM - 3 phút đọc

Một số thay đổi đáng chú ý trong Swift 5.5 (phần 1)

Sendable và @Sendable closures

SE-0302 bổ sung hỗ trợ cho dữ liệu "Sendable", là dữ liệu có thể được chuyển một cách an toàn sang một chuỗi khác. Điều này được thực hiện thông qua giao thức Sendable mới và thuộc tính @Sendable cho các hàm. Nhiều thứ vốn đã an toàn để gửi qua các chuỗi:

- Tất cả các loại giá trị cốt lõi của Swift, bao gồm Bool, Int, String, v.v.
- Tùy chọn, trong đó dữ liệu được bao bọc là một loại giá trị.
- Bộ sưu tập thư viện tiêu chuẩn có chứa các kiểu giá trị, chẳng hạn như Array hoặc Dictionary.
- Tuples trong đó các phần tử là tất cả các loại giá trị.
- Metatypes, chẳng hạn như String.self.

Chúng đã được cập nhật để phù hợp với giao thức Sendable. Đối với các loại tùy chỉnh, nó phụ thuộc vào những gì bạn đang thực hiện:

- Các Actor tự động tuân theo Sendable vì chúng xử lý đồng bộ hoá một cách nội bộ.
- Các struct và enum tùy chỉnh mà bạn tạo ra cũng sẽ tự động phù hợp với Sendable nếu chúng chỉ chứa các giá trị cũng phù hợp với Sendable, tương tự như cách Codable hoạt động.
- Các class tùy chỉnh có thể tuân theo Sendable miễn là chúng kế thừa từ NSObject hoặc không gì cả, tất cả các thuộc tính đều không đổi và bản thân chúng phù hợp với Sendable, và chúng được đánh dấu là final để ngừng kế thừa thêm.

  Swift cho phép chúng ta sử dụng thuộc tính @Sendable trên các hàm hoặc closure để đánh dấu chúng là hoạt động đồng thời và sẽ thực thi các quy tắc khác nhau để ngăn chúng ta tự bắn vào chân mình. Ví dụ: hoạt động mà chúng ta chuyển vào Task initiation được đánh dấu @Sendable, có nghĩa là loại mã này được cho phép vì giá trị được Tác vụ nắm bắt là một hằng số:

```
func printScore() async {
  let score = 1

  Task { print(score) }
  Task { print(score) }
}
```

Tuy nhiên đoạn code trên sẽ không được cho phép nếu score là một variable, vì các tác vụ khác có thể can thiệp nó trong khi có 1 tác vụ đang xử lý. Bạn có thể đánh dấu các chức năng và closure của riêng mình bằng cách sử dụng @Sendable, điều này sẽ thực thi các quy tắc tương tự xung quanh các giá trị đã capture:

```
func runLater(_ function: @escaping @Sendable () -> Void) -> Void {
    DispatchQueue.global().asyncAfter(deadline: .now() + 3, execute: function
}
```

#if cho các biểu thức postfix

SE-0308 cho phép Swift sử dụng điều kiện #if với các biểu thức thành viên postfix. Điều này nghe có vẻ hơi tối nghĩa, nhưng nó giải quyết được một vấn đề thường thấy với SwiftUI: bây giờ bạn có thể tùy chọn thêm công cụ sửa đổi vào chế độ xem. Ví dụ: thay đổi này cho phép chúng ta tạo chế độ xem văn bản với hai kích thước phông chữ khác nhau tùy thuộc vào việc chúng ta đang sử dụng iOS hay nền tảng khác:

```
Text("Welcome")
#if os(iOS)
    .font(.largeTitle)
#else
    .font(.headline)
#endif
```

Bạn có thể lồng như thế này nếu muốn, mặc dù hơi khó nhìn:

```
#if os(iOS)
    .font(.largeTitle)
    #if DEBUG
        .foregroundColor(.red)
    #endif
#else
    .font(.headline)
#endif
```

Bạn có thể sử dụng các biểu thức hậu tố cực kỳ khác nhau nếu bạn muốn:

```
let result = [1, 2, 3]
#if os(iOS)
    .count
#else
    .reduce(0, +)
#endif

print(result)
```

Về mặt kỹ thuật, bạn có thể tạo ra `Result` là hai loại hoàn toàn khác nhau nếu bạn muốn, nhưng đó có vẻ là một ý tưởng tồi. Điều bạn chắc chắn không thể làm là sử dụng các loại biểu thức khác, chẳng hạn như sử dụng `+ [4]` thay vì `.count` - nếu nó không bắt đầu bằng `.` thì nó không phải là một biểu thức thành viên postfix.

Cho phép sử dụng hoán đổi các loại CGFloat và Double

SE-0307 giới thiệu một cải tiến nhỏ nhưng quan trọng : Swift có thể chuyển đổi ngầm giữa CGFloat và Double ở hầu hết những nơi cần thiết. Ở dạng đơn giản nhất, điều này có nghĩa là chúng ta có thể thêm CGFloat và Double với nhau để tạo ra một Double mới, như sau:

```
let first: CGFloat = 42
let second: Double = 19
let result = first + second
print(result)
```

Swift thực hiện điều này bằng cách chèn một bộ khởi tạo ngầm nếu cần và nó sẽ luôn ưu tiên Double nếu có thể. Quan trọng hơn, không ai trong số này đạt được bằng cách viết lại các API hiện có: về mặt kỹ thuật, những thứ như scaleEffect () trong SwiftUI vẫn hoạt động với CGFloat, nhưng Swift lặng lẽ kết nối điều này với Double.

swift 5.5

Bài viết liên quan
Async/ await trong phiên bản Swift 5.5
Thachnd
10 phút đọc
93 0 0 1
Cách sử dụng thuộc tính MainActor để tự động gửi...
Nguyen Trong Hieu
8 phút đọc
30 0 0 0
Actor trong Swift 5.5
Thachnd
6 phút đọc
21 0 0 0
Structured concurrency trong Swift 5.5
Thachnd
10 phút đọc
18 0 0 0

Bài viết khác từ Thachnd
Actor trong Swift 5.5
Thachnd
6 phút đọc
21 0 0 0
Structured concurrency trong Swift 5.5
Thachnd
10 phút đọc
18 0 0 0
Async/ await trong phiên bản Swift 5.5
Thachnd
10 phút đọc
93 0 0 1
UI Datepicker mới trong ios14
Thachnd
2 phút đọc
65 0 0 0

Bình luận