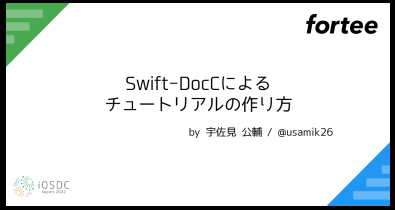
Swift Regexの話

宇佐見公輔 / 株式会社ゆめみ

自己紹介

- 宇佐見公輔 / 株式会社ゆめみ
- iOSDC Japan 2022で記事（x2）を書き、トークをします。



Swift Regex

- Swift 5.7で、正規表現を扱う `Regex` 型が追加される。
- 従来もFoundationに `NSRegularExpression` が存在した。
- 新しい `Regex` はSwiftの標準ライブラリとして組み込まれる。

Swift Regexの情報源

- Meet Swift Regex（WWDC22）
- Swift Regex: Beyond the basics（WWDC22）
- Xcode 14 beta Developer Documentation
- SE-0350 Regex type and overview
- SE-0351 Regex builder DSL
- SE-0354 Regex literals
- SE-0355 Regex syntax
- SE-0357 Regex-powered algorithms

Swift Regexの状況

- 現在のSwift 5.7はベータ版であり、未実装の機能がある。
  - セッションやドキュメントのコードが動くとは限らない。
- GitHubリポジトリ
  - apple/swift-experimental-string-processing リポジトリで開発。
  - その後 apple/swift リポジトリにマージ。

Regex型

```
let regex = try Regex(#"\d+"#)  // 補足： #"〜"# はSwiftの文字列リテラルの記法

"123".firstMatch(of: regex) // -> ["123"]
"a12b3".firstMatch(of: regex) // -> ["12"]
```

Regexの生成方法

```
let regex = try Regex(#"\d+"#)

// Regexリテラル
let regex = /\d+/

// Regexビルダー
import RegexBuilder
let regex = OneOrMore(.digit)
```

Regexリテラル

```
// 通常の区切り文字
let regex = /[a-zA-Z][a-zA-Z0-9]*/

// 拡張区切り文字
let regex = #//usr/lib/.*/#

// 複数行リテラル
let regex = #/
  \d{2} / \d{2} / \d{4}
  \P{currencySymbol}+
  \p{currencySymbol}
/#
```

- 不正な正規表現ならばビルドエラーになる。
- シンタックスハイライトもサポートされる。

リテラルによる値のキャプチャ

```swift
// キャプチャ
let regex = /user_id:\s*(\d+)/
let match = "user_id: 1234".wholeMatch(of: regex)
if let match {
    print(match.0) // -> "user_id: 1234"
    print(match.1) // -> "1234"
}

// 名前付きキャプチャ
let regex = /user_id:\s*(?<id>\d+)/
let match = "user_id: 1234".wholeMatch(of: regex)
if let match {
    print(match.id) // -> "1234"
}
```

Swift Regexの正規表現シンタックス

- 正規表現は環境によって細かいシンタックスの差異がある。
- SE-0355 Regex syntax
- ベースの文法 : PCRE 2 / Oniguruma / ICU / .NET
- Unicodeをサポートする。

Regexビルダー

```
import RegexBuilder

//let regex = /\d+/
let regex = OneOrMore(.digit)
let regex = Regex {
    OneOrMore(.digit)
}

//let regex = /user_id:\s*\d+/
let regex = Regex {
    "user_id:"
    ZeroOrMore(.whitespace)
    OneOrMore(.digit)
}
```

ビルダーの中でリテラルを使う

```
import RegexBuilder

let regex = Regex {
    "Test Suite '"
    /[a-zA-Z][a-zA-Z0-9]*/
    "' "
    ChoiceOf {
        "started"
        "passed"
        "failed"
    }
    " at "
    OneOrMore(.any, .reluctant)
    Optionally(".")
}
```

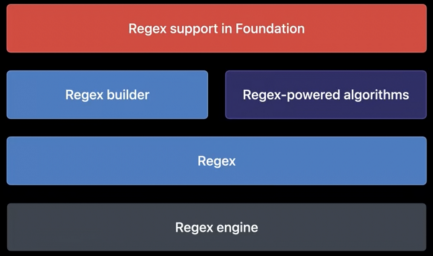
ビルダーによる値のキャプチャ

```
//let regex = /user_id:\s*(\d+)/
let regex = Regex {
    "user_id:"
    ZeroOrMore(.whitespace)
    Capture(OneOrMore(.digit))
}

let match = "user_id: 1234".wholeMatch(of: regex)
if let match {
    print(match.0) // -> "user_id: 1234"
    print(match.1) // -> "1234"
}
```

TryCapture

```
let regex = Regex {
    "user_id:"
    ZeroOrMore(.whitespace)
    TryCapture {
        OneOrMore(.digit)
    } transform: {
        Int($0)
    }
}

let match = "user_id: 1234".wholeMatch(of: regex)
if let match {
    let (wholeMatch, id) = match.output
    print(id) // -> 1234
}
```

Regexを使う


Regex support in Foundation
Regex builder
Regex-powered algorithms
Regex
Regex engine

Regex-powered algorithms

- 文字列処理にRegexサポートが追加される。

```
input.firstMatch(of: regex)
input.wholeMatch(of: regex)
input.prefixMatch(of: regex)

input.starts(with: regex)
input.replacing(regex, with: "456")
input.trimmingPrefix(regex)
input.split(by: /\s*,\s*/)
```

switch文のサポート

```
switch "abc" {
case /\w+/:
    print("It's a word!")
}
```

- ただし、現状では未実装の様子。

（あまり分かってない。動いた人がいたら教えてください）

Regex support in Foundation

- Foundationの既存のパーサーを流用できる。

```
import Foundation

let regex = Regex {
    One(.date(.numeric,
              locale: Locale(identifier: "ja_JP"),
              timeZone: TimeZone.current))
}
"2022/06/27".starts(with: regex) // -> true
```

カスタムのパーサーを使う

```
import Darwin
struct CDoubleParser: CustomConsumingRegexComponent {
    typealias RegexOutput = Double

    func consuming(
        _ input: String,
        startingAt index: String.Index,
        in bounds: Range<String.Index>
    ) throws -> (upperBound: String.Index, output: Double)? {
      // ...
    }
}

let regex = Regex {
    CDoubleParser()
}
```

Swift Regexまとめ

- `Regex` 型
- Regexリテラル / Regexビルダー
- Regex-powered algorithms / Regex support in Foundation