

# 正規表現の 少し進んだ機能

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 / iOSテックリード
- 大阪在住、最寄のゆめみオフィスは京都（まだ物理出社していないが）
- 来月のiOSDCにパンフレット記事寄稿、トーク登壇予定
- 来月の技術書典に出展予定

# 正規表現を再勉強中

- Swiftで正規表現が言語組み込みになる
- この機会に、正規表現を再勉強



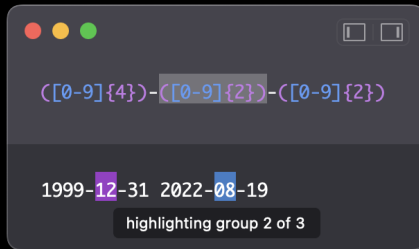
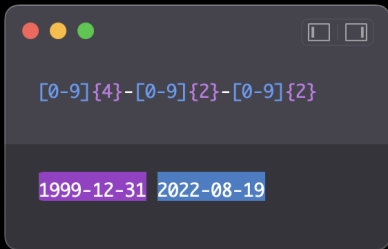
- 意外と知らない機能がいろいろあった
- 例：名前付きキャプチャ、後方参照



- 書いた：[Swift Regexでキャプチャや名前付きキャプチャを使う - Qiita](#)

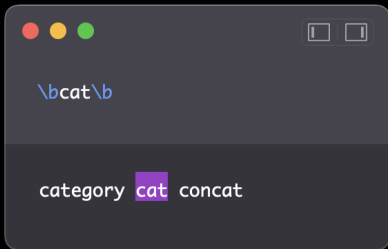
# 正規表現の基本

- 文字列のパターンマッチ
- 部分文字列の抽出（キャプチャ）



# 位置へのマッチ

- アンカー：「文字列」でなく「位置」にマッチする
- 長さ0の文字列にマッチすると考えて、ゼロ幅アサーションとも呼ばれる
- `^`（先頭） `$`（末尾） `\b`（単語の境界） など



```
\bcat\b  
  
category cat concat
```

A terminal window with a dark background. The first line shows the regex `\bcat\b` in blue. The second line shows the text `category cat concat`, where the word `cat` is highlighted in purple, demonstrating a successful match.

# 少し進んだ機能の紹介

- 先読み (lookahead)
- 後読み (lookbehind)

# 先読み

先読み (lookahead) は、位置にマッチする記法の一つで、位置の指定に正規表現が使える。 ``(?:=)`` と ``)`` で囲む。

```
a(?:..d)
```

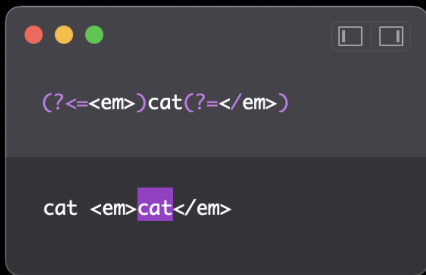
→ 「a」の次に「任意の2文字+d」が来る場合に限り、「a」にマッチする

```
a(?:..d)
```

```
abracadabra
```

# 後読み

- `(?=regex)` : 先読み。次に `regex` がくる位置にマッチ。
- `(?<=regex)` : 後読み。前に `regex` がくる位置にマッチ。



```
(?<=<em>)cat(?=</em>)
```

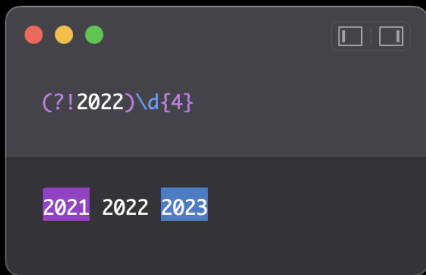
```
cat <em>cat</em>
```

The image shows a code editor window with a dark background. The top bar has three colored circles (red, yellow, green) and two window control icons. The editor contains two lines of text. The first line is a regular expression: `(?<=<em>)cat(?=</em>)`. The second line is a string: `cat <em>cat</em>`. In the string, the word `cat` inside the `<em>` tags is highlighted with a yellow background.



# 否定先読み・否定後読み

- `(?!regex)` : 否定先読み。次に `regex` がこない位置にマッチ。
- `(?<!regex)` : 否定後読み。前に `regex` がこない位置にマッチ。



# 先読みが便利な場合(1)

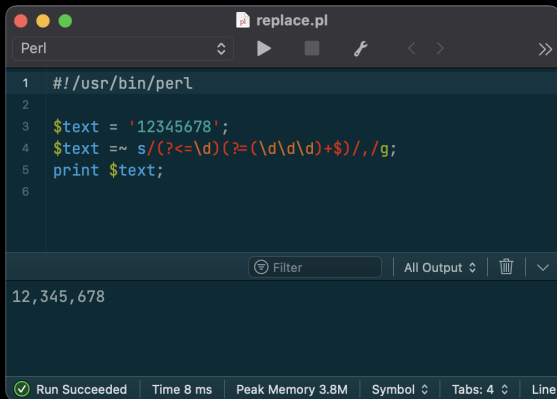
- 複数の正規表現すべてにマッチするか

```
(?=.*hoge.)*(?=.*fuga.)*(?=.*piyo.)*.
```

```
000-fuga-piyo-hoge-999
```

# 先読みが便利な場合(2)

- 数値の3桁ごとにカンマを挿入するコード



The screenshot shows a Perl script window titled 'replace.pl'. The script contains the following code:

```
1  #!/usr/bin/perl
2
3  $text = '12345678';
4  $text =~ s/(?<=\d)(?=(\d\d\d)+$)/./g;
5  print $text;
6
```

The output of the script is displayed in the bottom panel: 12,345,678. The status bar at the bottom indicates 'Run Succeeded', 'Time 8 ms', 'Peak Memory 3.8M', 'Symbol', 'Tabs: 4', and 'Line'.

→ これ、どうなってるの？

# 先読みが便利な場合(2) 解説

```
1 #!/usr/bin/perl
2
3 $text = '12345678';
4 $text =~ s/(?<=\d)(?=(\d\d\d)+$)/,/g;
5 print $text;
6
```

後読み      先読み

長さ0の文字列にマッチ      長さ0の文字列をカンマに置換

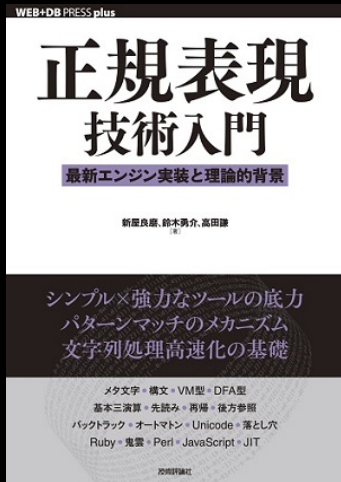
12345678

↑    ↑

12,345,678

Run Succeeded | Time 8 ms | Peak Memory 3.8M | Symbol | Tabs: 4 | Line

# 書籍



# 便利ツール

