链滴

CoffeeScript单行代码

作者：R

原文链接：https://ld246.com/article/1469606193826

来源网站：链滴

1. 让列表中的每个元素都乘以2

```
[1..10].map (i) -> i*2
```

或

```
i * 2 for i in [1..10]
```

2. 求列表中的所有元素之和

```
[1..1000].reduce (t, s) -> t + s
```

(reduce == reduceLeft, reduceRight 也可以)

3. 判断一个字符串中是否存在某些词

```
wordList = ["coffeescript", "eko", "play framework", "and stuff", "falsy"]
tweet = "This is an example tweet talking about javascript and stuff."

wordList.some (word) -> ~tweet.indexOf word
```

下面的例子会返回匹配的单词：

```
wordList.filter (word) -> ~tweet.indexOf word
```

~ is not a special operator in CoffeeScript, just a dirty trick. It is the bitwise NOT operator, whi h inverts the bits of it’ s operand. In practice it equates to -x-1. Here it works on the basis tha we want to check for an index greater than -1, and -(-1)-1 == 0 evaluates to false.

4. 读取文件

```
fs.readFile 'data.txt', (err, data) -> fileText = data
```

同步版本：

```
fileText = fs.readFileSync('data.txt').toString()
```

In node.js land this is only acceptable for application start-up routines. You should use the as nc version in your code.

5. 祝你生日快乐！

```
[1..4].map (i) -> console.log "Happy Birthday " + (if i is 3 then "dear Robert" else "to You")
```

下面这一版读起来更像是伪代码：

```
console.log "Happy Birthday #{if i is 3 then "dear Robert" else "to You"}" for i in [1..4]
```

6. 过滤列表中的数值

```
(if score > 60 then (passed or passed = []) else (failed or failed = [])).push score for score in [4
, 58, 76, 82, 88, 90]
```

更函数式的方法：

```
[passed, failed] = [49, 58, 76, 82, 88, 90].reduce ((p,c,i) -> p[+(c < 60)].push c; p), [[],[]]
```

7. 获取XML web service数据并分析

这里用json代替XML：

```
request.get { uri:'path/to/api.json', json: true }, (err, r, body) -> results = body
```

8. 找到列表中最小或最大的一个数字

```
Math.max.apply @, [14, 35, -7, 46, 98] # 98
Math.min.apply @, [14, 35, -7, 46, 98] # -7
```

9. 并行处理

Not there yet. You can create child processes on your own and communicate with them, or u
e the WebWorkers API implementation. Skipping over.

10. “Sieve of Eratosthenes”算法

下面的代码可以写成一行吗?

```
sieve = (num) ->
  numbers = [2..num]
  while ((pos = numbers[0]) * pos) <= num
    delete numbers[i] for n, i in numbers by pos
    numbers.shift()
  numbers.indexOf(num) > -1
```

跟紧凑的版本：

```
primes = []
primes.push i for i in [2..100] when not (j for j in primes when i % j == 0).length
```

真正的一行实现：

```
(n) -> (p.push i for i in [2..n] when not (j for j in (p or p=[]) when i%j == 0)[0]) and n in p
```

```
(n) -> (p.push i for i in [2..n] when !(p or p=[]).some((j) -> i%j is 0)) and n in p
```

原文链接：CoffeeScript单行代码