

# 如何 clone 一个正则?



老姚 Lv6

2019年02月12日 04:37 · 阅读 4260

+ 关注

克隆一个正则，Lodash 库的实现方式是：

```
const reFlags = /\w*$/

function cloneRegExp(regex) {
  const result = new regex.constructor(regex.source, reFlags.exec(regex))
  result.lastIndex = regex.lastIndex
  return result
}

cloneRegExp(/xyz/gim)
// => /xyz/gim
```

复制代码

通过这段代码，我们顺便复习一下 JS 正则对象的部分知识。



66



11



收藏

首先，`RegExp.constructor` 就是 `RegExp`。

了解 JS 原型相关知识的话，这一点应该没问题。

具体说来，`/xyz/gim` 是正则字面量，是构造函数 `RegExp` 的实例。`/xyz/gim` 取 `constructor` 属性时，根据原型链原理，对象本身没有此属性时，要再去它的原型里找。而 `/xyz/gim` 的原型是 `RegExp.prototype`。同时 `RegExp.prototype.constructor` 正是 `RegExp` 本身。

构造函数 `RegExp` 的一个典型用法是：

```
var regexp = new RegExp('xyz', 'gim');  
// 等价于  
var regexp = /xyz/gim;
```

[复制代码](#)

## 2.正则实例组成

一个正则对象可以大致分成两部分，源码(source)和修饰符(flags)。比如，`/xyz/gim` 的 `source` 是 `"xyz"`，而其 `flags` 是 `"gim"`。

```
var regexp = /xyz/gim  
regexp.source  
// => "xyz"  
regexp.flags  
// => "gim"
```

[复制代码](#)



66



11



收藏

关于修饰符，多说一句。在 JS 中，目前共有 6 个修饰符：g、i、m、s、u、y。正则对象转化为字符串时，其修饰符排序是按字母排序的。

```
var regexp = /xyz/imgyus;
regexp.flags
// => "gimsuy"
regexp.toString()
// => "/xyz/gimsuy"
```

[复制代码](#)

Lodash 的源码，获取修饰符用时没有通过 flags，而是采用正则提取：

```
/\w*$/ .exec(regexp.toString()).toString()
// => gim
```

[复制代码](#)

其中，正则 /\w\*\$/ 匹配的是字符串尾部字母。因为目标正则可能没有修饰符，因此这里量词是 \*。

估计你看出来了。是的，下面代码里有两处类型转换（转字符串）：

```
new regexp.constructor(regexp.source, reFlags.exec(regexp))
```

[复制代码](#)

### 3.lastIndex是可修改的

clone 正则时，还要 clone 其 lastIndex。这一点学到了！



66



11



收藏

`lastIndex` 表示每次匹配时的开始位置。使用正则对象的 `test` 和 `exec` 方法，而且当修饰符为 `g` 或 `y` 时，对 `lastIndex` 是有影响的。

例如：

```
var regexp = /\d/g;

regexp.lastIndex
// => 0
regexp.test("123")
// => true

regexp.lastIndex
// => 1
regexp.test("1")
// => false
```

复制代码

第 1 次 `test` 时，在输入字符串 `"123"` 中匹配到了第一个数字 `"1"`。`lastIndex` 此时也变成了 `1`，表示下次的匹配位置将会跳过第 `0` 位，直接从第 `1` 位开始。

第 2 次 `test` 时，此时输入是字符串 `"1"`，只有一位字符，其第 `1` 位是空，因此匹配失败。此时 `lastIndex` 会重置为 `0`。

最关键一点，`lastIndex` 属性不仅可读，而且可写：

```
var regexp = /\d/g;
regexp.lastIndex = 2
```

复制代码



66



11



收藏

```
regex.test("123")  
// => false
```

至此，lodash 的实现，应该都能全部看懂了：

```
const reFlags = /\w*$/  
  
function cloneRegExp(regex) {  
  const result = new regex.constructor(regex.source, reFlags.exec(regex))  
  result.lastIndex = regex.lastIndex  
  return result  
}  
  
cloneRegExp(/xyz/gim)  
// => /xyz/gim
```

复制代码

本文完。

欢迎阅读 [《JS正则迷你书》](#)。

本文参考：

[cloneRegExp.js](#)



分类：

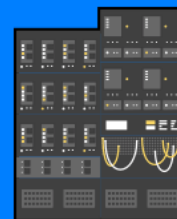
前端

标签：

JavaScript

正则表达式

找对——  
属于你的  
技术圈子



回复进群加入  
掘金  
微信交流群



评论

输入评论（Enter换行，Ctrl + Enter发送）



66



11



收藏

## 热门评论



凝香星梦 前端工程师 @ 群之脉

3年前

const result = new RegExp(regex.source, reFlags.exec(regex)) 也可以吧?  
mini 书讲的通俗易懂，对正则终于有了一个整体的概念，感谢～

 点赞  1

## 全部评论 11

 最新

 最热



JS菌  前端开发

3年前

有趣唉 之前只知道用构造函数 new 一个 一直忘记还有受 g 影响的 lastIndex

 点赞  回复



凝香星梦 前端工程师 @ 群之脉

3年前

const result = new RegExp(regex.source, reFlags.exec(regex)) 也可以吧?  
mini 书讲的通俗易懂，对正则终于有了一个整体的概念，感谢～



 点赞  1



浮浮浮浮浮萍 一个前端的公司 @ 一个...

3年前

看到老姚的正则文章先赞再看 

 点赞  回复

 66

 11

 收藏



奥巴驴无情 前端工程师

3年前

mini书真的很棒

👍 点赞 1



zachrey Lv2 前端工程师 @ 无

3年前

为什么需要采用正则提取修饰符呢？有什么好处呢？如以上，`regex.flags` 也能正常返回 修饰符不是么？

👍 点赞 1



SoCool 安全+前端+后端

3年前

有趣.

👍 点赞 1 回复



webgzh9072471... 打杂 @ 打杂

3年前

学到了~

👍 点赞 1 回复

## 相关推荐

清夜 | 23天前 | JavaScript · 正则表达式 · Vue.js

这些常用正则表达式是怎么写出来的？

👍 66

💬 11

★ 收藏



wbh爱吃西瓜 | 6月前 | JavaScript · 前端

## 手写深拷贝 DeepClone

👁 808    👍 点赞    💬 评论

泡沫的快乐 | 6月前 | 前端 · JavaScript

## 通过lodash的cloneDeep学习深拷贝

👁 801    👍 6    💬 评论

为什么不学习 | 3月前 | JavaScript

## 如何实现一个让面试官惊艳的深克隆

👁 2068    👍 31    💬 10

波比小金刚 | 3年前 | 前端

## 深度克隆的最简单实现

👁 1613    👍 13    💬 评论

Vam的金豆之路 | 1年前 | JavaScript

## 15个实用实用正则（小哥进来看看？）

👁 1207    👍 4    💬 评论

心中有个月亮 | 3年前 |

## 用正则split

👍 66

💬 11

☆ 收藏

周兴博 | 2年前 | Python · 正则表达式

## Python正则表达式 (re) 简明讲解, 以及常用正则: 邮箱、身份证、IP地址、手机号、密码等

👁 3028    👍 11    💬 3

芬达哥 | 1年前 | Git

## git clone很慢 怎么办? 是不是用了 解析github.global.ssl.fastly.net ip? ?

👁 123    👍 1    💬 评论

大码猴 | 8月前 | 前端 · 正则表达式

## 鲜为人知的正则大全, 让你不再为正则而烦恼

👁 875    👍 17    💬 3

锐玩道 | 2年前 | 正则表达式

## 常用正则

👁 5740    👍 57    💬 评论

铁皮饭盒 | 3月前 | 前端 · JavaScript · Vue.js

## 数据分析前端正则🔪TOP10, 必用★正则(77条)

👁 2161    👍 53    💬 2

Promise废物 | 2年前 | 正则表达式

## 看完不会正则, up把正则吃了

👍 66

💬 11

☆ 收藏

前端胖头鱼 | 7月前 | 面试 · JavaScript · 正则表达式

## 【建议收藏】25+正则面试题详尽解析，让你轻松通过正则面试，让你少写2000行代码

👁 8295    👍 340    💬 40

ThinkMore28092 | 4年前 | JavaScript · 前端 · 正则表达式

## 理清JS中等于（==）和全等（===）那些纠缠不清的关系

👁 4703    👍 37    💬 4

varnull | 4年前 | 正则表达式 · JavaScript · 前端

## 一次记住js的6个正则方法

👁 6448    👍 410    💬 1

锐玩道 | 2年前 | 正则表达式

## 练完这篇就会写正则

👁 2.3w    👍 676    💬 36

前端胖头鱼 | 5月前 | 前端 · 正则表达式 · JavaScript

## 就因为这三个知识点，我彻底学废了“正则表达式”

👁 3.4w    👍 1301    💬 168

一tiao咸鱼 | 1年前 | 正则表达式

## 前端js正则使用，正则你今天学会了吗

👍 66

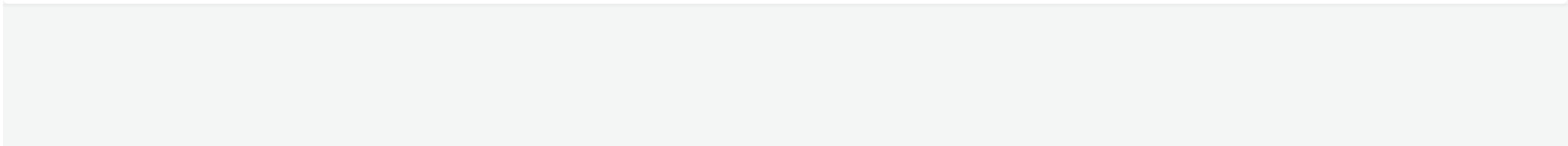
💬 11

☆ 收藏

铁皮饭盒 | 1年前 | JavaScript · 正则表达式

## 2020年这些🐛"正则"应该被收藏(更新, 64条)

👁 2.4w    👍 774    💬 57



👍 66

💬 11

☆ 收藏