#### ES2019提案预览

##### 之正则表达式

摘要

JavaScript语言特性——正则表达式

正文

命名捕获组（Named Capture Groups）

带编号的捕获组使您可以引用正则表达式匹配的字符串的某些部分。 每个捕获组都分配有一个唯一的编号，并且可以使用该编号进行引用，但这会使正则表达式难以理解和重构。

Numbered capture groups allow one to refer to certain portions of a string that a regular expression matches. Each capture group is assigned a unique number and can be referenced using that number, but this can make a regular expression hard to grasp and refactor.

正向断言（Lookahead Assertions）

反向断言（Lookbehind Assertions）

正向断言的意思是：当前位置后面的字符串应该满足断言，但是并不捕获。反向断言和正向断言的行为一样，只是方向相反。在当前的 JavaScript 正则表达式版本中，只支持正向断言。

Match Indices

示例如下：

const re1 = /a+(?<Z>z)?/;

// indices are relative to start of the input string:

const s1 = "xaaaz";

const m1 = re1.exec(s1);

m1.indices[0][0] === 1;

m1.indices[0][1] === 5;

s1.slice(...m1.indices[0]) === "aaaz";

String.prototype.matchAll

String.prototype.replaceAll

当前，不使用全局正则表达式无法替换字符串中子字符串的所有实例。 与字符串参数一起使用时，String.prototype.replace仅影响首次出现。 有很多证据表明开发人员正在尝试在JS中执行此操作-请以成千上万的票数查看StackOverflow问题。

Currently there is no way to replace all instances of a substring in a string without use of a global regexp. String.prototype.replace only affects the first occurrence when used with a string argument. There is a lot of evidence that developers are trying to do this in JS — see the StackOverflow question with thousands of votes.

□