RegExp Updates

RegExp

Sticky

除了u修饰符，ES6还为正则表达式添加了y修饰符，叫做“粘连”（sticky）修饰符。

y修饰符的作用与g修饰符类似，也是全局匹配，后一次匹配都从上一次匹配成功的下一个位置开始。不同之处在于，g修饰符只要剩余位置中存在就可，而y修饰符确保匹配必须从剩余的第一个位置开始，这也就是“粘连”的涵义。

```
const s = 'aaa_aa_a'
const r1 = /a+/g
const r2 = /a+/y

r1.exec(s) // ["aaa"]
r2.exec(s) // ["aaa"]

r1.exec(s) // ["aa"]
r2.exec(s) // null
```

上面代码有两个正则表达式，一个使用g修饰符，另一个使用y修饰符。这两个正则表达式各执行了两次，第一次执行的时候，两者行为相同，剩余字符串都是_aa_a。由于g修饰没有位置要求，所以第二次执行会返回结果，而y修饰符要求匹配必须从头部开始，所以返回null。

如果改一下正则表达式，保证每次都能头部匹配，y修饰符就会返回结果了。

```
const s = 'aaa_aa_a'
const r = /a+_/y

r.exec(s) // ["aaa_"]
r.exec(s) // ["aa_"]
```

上面代码每次匹配，都是从剩余字符串的头部开始。

使用lastIndex属性，可以更好地说明y修饰符。

```
const REGEX = /a/g

// 指定从2号位置（y）开始匹配
REGEX.lastIndex = 2

// 匹配成功
const match = REGEX.exec('xaya')

// 在3号位置匹配成功
console.log(match.index) // 3

// 下一次匹配从4号位开始
console.log(REGEX.lastIndex) // 4

// 4号位开始匹配失败
REGEX.exec('xaxa') // null
```

上面代码中，lastIndex属性指定每次搜索的开始位置，g修饰符从这个位置开始向后搜索，直到发现匹配为止。

y修饰符同样遵守lastIndex属性，但是要求必须在lastIndex指定的位置发现匹配。

```
const REGEX = /a/y

// 指定从2号位置开始匹配
REGEX.lastIndex = 2

// 不是粘连，匹配失败
REGEX.exec('xaya') // null
```