作为开始，我们先看下面的正则：

var str = 'a "witch" and her "broom" is one';

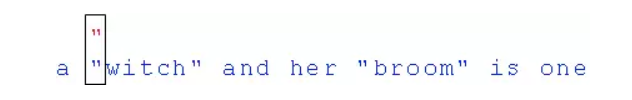
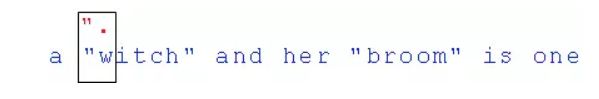
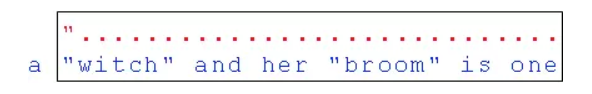
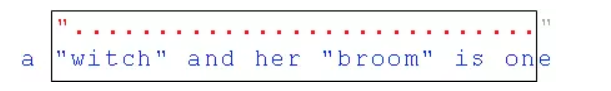
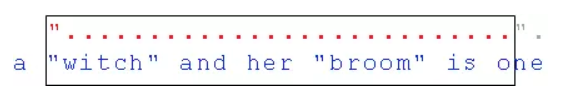
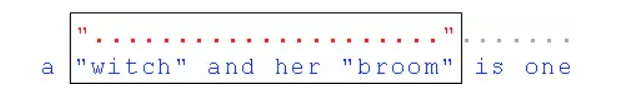
str.match( /".\*"/g);

我们本来预想上面会匹配得到"witch"和"broom"两个字符串，运行上面的例子，却发现结果只匹配到"witch" and her "broom"一个字符串。

之所以出现这个结局，是因为正则的贪婪模式在起作用。

**一、贪婪模式（默认）**

首先我们假设自己是正则引擎，来模拟搜索实现的过程。  
正则引擎先从字符串的第0位开始搜索。

1. 第一个查找字符是"，正则引擎在第三个位置匹配到了它：
2. 
3. 之后，引擎尝试匹配正则的剩余部分，第二个字符是.,它代表任意字符。引擎匹配到了w:
4. 
5. .代表任意字符重复一次到多次，因此正则引擎匹配到所有字符
6. 
7. 当文本结束后，点的匹配停止了，但仍然有剩余的正则"需要匹配，因此正则引擎开始倒过来回溯，换句话说，就是一个字符一个字符缩减匹配。
8. 
9. 当匹配缩减后，它开始尝试匹配剩余的正则，但"没有匹配上字符e。
10. 因此正则继续缩减.所重复的字符，继续尝试。
11. 
12. 正则引擎回溯，一次一次缩减.重复的字符个数，直到剩余的正则都匹配上：
13. 
14. 现在"终于匹配上了。 如果正则是global的，正则引擎会从上次匹配结果之后继续查找更多结果。

**总结：在贪婪（默认）模式下，正则引擎尽可能多的重复匹配字符。**

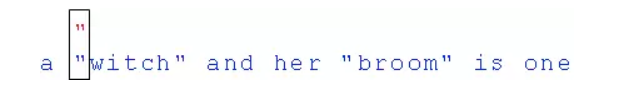
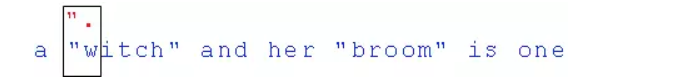
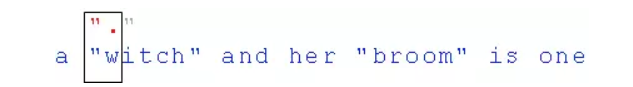
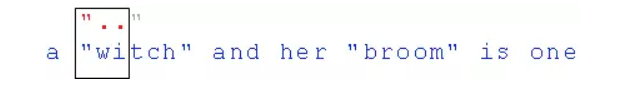
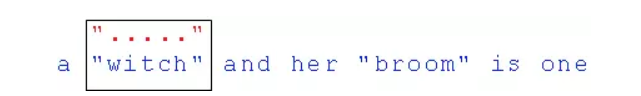
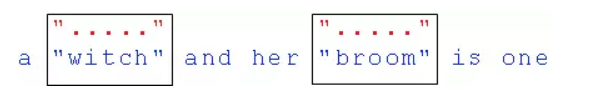
# 二、非贪婪模式

非贪婪模式和贪婪模式相反，可通过在代表数量的标识符后放置?来开启非贪婪模式，如?、+?甚至是??。

var str = 'a "witch" and her "broom" is one';

str.match(/".\*?"/g ) // "witch", "broom"

我们来看看非贪婪模式.?是怎么运转的：

1. 第一步和上面类似，引号"被匹配上
2. 
3. 第二步也一样， '.'被匹配上
4. 
5. 下面是二者的重要区别。 正则引擎尝试用**最小可能**的重复次数来进行匹配，因此在.匹配了w后，它立即尝试"的匹配
6. 
7. 可惜没有匹配上，因为t!="。
8. .重复更多的字符，再进行尝试
9. 
10. 又没匹配上，继续~~
11. 下面终于匹配上了
12. 
13. 因为正则是global的，所以正则引擎继续后面的匹配，从引号后面的a字符开始，后面又匹配到第二个字符串
14. 
15. **总结：在非贪婪模式下，正则引擎尽可能少的重复匹配字符。**