var str = "<p>Para 1.</p>" +

"<img src='smiley.jpg'>" +

"<p>Para 2.</p>" +

"<div>Div.</div>";

/<p>.\*<\/p>/i.test(str);

正则表达式一上来就匹配了字符串开始的三个字母<p>。然后是.\*。点号匹配除换行符以外的任意字符，星号这个贪婪量词表示重复零次或多次——匹配尽量多的次数。因为目标字符串中没有换行符，它将吞噬

剩下的全部字符串！不过正则表达式模板中还有更多内容需要匹配，所以正则表达式尝试匹配<。它在字符串末尾匹配不成功，所以它每次回溯一个字符，继续尝试匹配<，直到它回到</div>标签的<位置。然后它尝试匹配\/（转义反斜杠），匹配成功，然后是p，匹配不成功。正则表达式继续回溯，重复此过程，直到第二段末尾时它终于匹配了</p>。匹配返回成功，它从第一段头部一直扫描到最后一个的末尾，这可能不是你想要的结果。

你可以将正则表达式中的贪婪量词\*改为懒惰（又名非贪婪）量词\*?，以匹配单个段落。懒惰量词的回溯工作以相反方式进行。当正则表达式/<p>.\*?<\/p>/推进到.\*?时，它首先尝试全部跳过然后继续匹配<\/p>。它这么做是因为\*?匹配零次或多次，但尽可能少重复，尽可能少的话那么它就可以重复零次。但是，当随后的<在字符串的这一点上匹配失败时，正则表达式回溯并尝试下一个最小的字符数：一个。它继续像这样向前回溯到第一段的末尾，在那里量词后面的<\/p>得到完全匹配。

