正则表达式

1 分组符 ()

|  |  |
| --- | --- |
| 捕获性分组 () | 非捕获性分组 (?:) |
| 会保存每个分组里匹配的值 | 匹配校验，会出现在匹配结果里面，但不作为子匹配  返回 |
| **var *groupRaw*** = **'xyy test601 test34 nan'  var *rst1*** = ***groupRaw***.match(/test(\d+)/)  *//返回结果["test601","601"]* **var *rst2*** = ***groupRaw***.match(/test(\d)+/)  *//返回结果["test601","1"],贪婪模式,尽可能多的匹配分组(\d),返回最后一次匹配* **var *rst3*** = ***groupRaw***.match(/test(\d)+?/)  *//返回结果["test601","6"],非贪婪模式* | **var *rst4*** = ***groupRaw***.match(/test(?:\d+)/)  *//返回结果["test601"]* |
| 零宽断言----匹配宽度为零，满足一定的条件/断言 | |
| 前瞻（先行断言） | 后顾 （后发断言） |
| 正向前瞻(?=)，表示后面有什么  反向前瞻(?!)，表示后面不会有什么 | 正向后顾(?<=)，表示前面有什么  反向后顾(?<!)，表示前面没什么 |
| **var *foresightStr*** = **"123.jpg abc.png def.jpg" var *rst*** = ***foresightStr***.match(/\w+(?=\.jpg)/g)  *//返回结果["123","def"]*  **var *foresightStr*** = **"abc000 tes0003 ebfa0003 ac1111 bc0001 abcd1000 ABC123" var *rst*** = ***foresightStr***.match(/[a-z]{3,}(?!000)/ig)  *//返回结果["ebf", "abcd", "ABC"]* | **var *str*** = **'aaa000 aaaa111 aaaaaab222'**; ***str***.match(/(?<=aaa)\d{3}/g)  *//返回结果["000","111"]*  ***str***.match(/(?<!aaa)\d{3}/g)  *//返回结果["222"]* |