HTML

正则表达式

- 正则表达式概述
- 如何创建正则表达式对象
- 正则表达式基本语法
- 正则表达式基本使用
- 正则表达式对象方法使用
- 字符串正则表达式方法使用

正则表达式基本概述

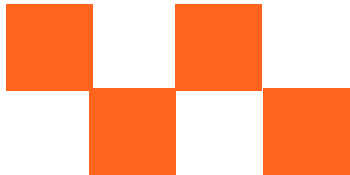
正则表达式(regular expression)描述了一种字符串匹配的模式，可以用来检查一个串是否含有某种子串、将匹配的子串做替换或者从某个串中取出符合某个条件的子串等。

创建正则表达式对象方法一

```
var reg = new RegExp(参数1, 参数2);
```

参数1:所规定的规则

参数2:设置的属性(attribute),可选的,可省略.



创建正则表达式对象方法二

```
var reg = /参数1/参数2
```

参数1:所规定的规则

参数2:设置的属性(attribute),可选的,可省略.

创建正则表达式对象的参数二

这些都是模式匹配符，放在正则表达式的最后，当参数使用。
i：ignorCase忽略大小写
m：mutiple允许多行匹配
g：globle进行全局匹配，指匹配到目标串的结尾

正则表达式基本语法

正则表达式是由以下两种类型的字符组成

1.普通字符（a-z, A-Z, 0-9, 汉字, _）

2.元字符 (非打印字符, 特殊字符,限定符)

非打印字符

字符	描述
\cx	匹配由x指明的控制字符。例如，\cM 匹配一个 Control-M 或回车符。x 的值必须为 A-Z 或 a-z 之一。否则，将 c 视为一个原义的 'c' 字符。
\f	匹配一个换页符。等价于 \x0c 和 \cL。
\n	匹配一个换行符。等价于 \x0a 和 \cJ。
\r	匹配一个回车符。等价于 \x0d 和 \cM。
\s	匹配任何空白字符，包括空格、制表符、换页符等等，等价于 [\f\n\r\t\v]。
\S	匹配任何非空白字符。等价于 [^ \f\n\r\t\v]。
\t	匹配一个制表符。等价于 \x09 和 \cI。
\v	匹配一个垂直制表符。等价于 \x0b 和 \cK。

特殊字符

特别字符	描述
$	匹配输入字符串的结尾位置。如果设置了 RegExp 对象的 Multiline 属性，则 $ 也匹配 '\n' 或 '\r'。要匹配 $ 字符本身，请使用 \$。
()	标记一个子表达式的开始和结束位置。子表达式可以获取供以后使用。要匹配这些字符，请使用 \(和 \)。
*	匹配前面的子表达式零次或多次。要匹配 * 字符，请使用 *。
+	匹配前面的子表达式一次或多次。要匹配 + 字符，请使用 \+。
.	匹配除换行符 \n 之外的任何单字符。要匹配 .，请使用 \.。
[	标记一个中括号表达式的开始。要匹配 [，请使用 \[。
?	匹配前面的子表达式零次或一次，或指明一个非贪婪限定符。要匹配 ? 字符，请使用 \?。
\	将下一个字符标记为或特殊字符、或原义字符、或向后引用、或八进制转义符。例如，'n' 匹配字符 'n'。'\n' 匹配换行符。序列 '\\' 匹配 "\"，而 '\(' 则匹配 "("。
^	匹配输入字符串的开始位置，除非在方括号表达式中使用，此时它表示不接受该字符集合。要匹配 ^ 字符本身，请使用 \^。
{	标记限定符表达式的开始。要匹配 {，请使用 \{。
|	指明两项之间的一个选择。要匹配 |，请使用 |。

限定符

字符	描述
*	匹配前面的子表达式零次或多次。例如，zo* 能匹配 "z" 以及 "zoo"。* 等价于{0,}。
+	匹配前面的子表达式一次或多次。例如，'zo+' 能匹配 "zo" 以及 "zoo"，但不能匹配 "z"。+ 等价于 {1,}。
?	匹配前面的子表达式零次或一次。例如，"do(es)?" 可以匹配 "do" 或 "does" 中的"do" 。? 等价于 {0,1}。
{n}	n 是一个非负整数。匹配确定的 n 次。例如，'o{2}' 不能匹配 "Bob" 中的 'o'，但是能匹配 "food" 中的两个 o。
{n,}	n 是一个非负整数。至少匹配n 次。例如，'o{2,}' 不能匹配 "Bob" 中的 'o'，但能匹配 "foooood" 中的所有 o。'o{1,}' 等价于 'o+'。'o{0,}' 则等价于 'o*'。
{n,m}	m 和 n 均为非负整数，其中n <= m。最少匹配 n 次且最多匹配 m 次。例如，"o{1,3}" 将匹配 "fooooood" 中的前三个 o。'o{0,1}' 等价于 'o?'。请注意在逗号和两个数之间不能有空格。

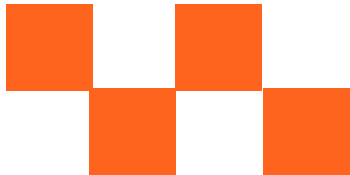
正则表达式方法

test()方法

该方法的返回值是布尔值，通过该值可以匹配字符串中是否存在于正则表达式相匹配的结果，如果有匹配内容，返回ture，如果没有匹配内容返回false，该方法常用于判断用户输入数据的合法性，比如检验Email的合法性

```
var str1 = "today is SUNNY day!";
var regExp1 = new RegExp("sunny");
var res = regExp1.test(str1);
console.log(res);
```

```
true
```



字符串正则表达式方法使用

match(参数)方法

字符串的match方法,跟正则表达式对象有关.
参数:正则表达式对象
返回值:如果一个字符串按照正则表达式对象的规则可以匹配到内容,就把内容写进数组,进行返回;如果匹配不到,返回null.

```
var str = "hello, world, hello, world";
var re = /hello/g;
console.log(str.match(re));
```

```
▼ Array[2]
    0: "hello"
    1: "hello"
    length: 2
  ▶ __proto__: Array[0]
```

字符串正则表达式方法使用

replace(参数1, 参数2)方法

参数1:正则表达式对象
参数2:要替换的内容
返回值:返回经过替换后的字符串
* 注意:只是函数的返回值是被替换后的字符串,原字符串不被修改

```
var str = "今天是周六";
var re = /周六/g;
console.log(str.replace(re, "周一"));
console.log(str);
```

```
今天是周一
今天是周六
```

字符串正则表达式方法使用

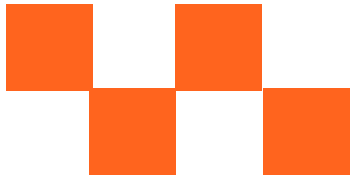
search(参数)方法

参数:正则表达式

返回值:如果匹配到正则中的字符串,就返回对应的下标,如果匹配不到,返回-1.

```
var str = "ilovestudy";
var re = /love/;
console.log(str.search(re));
```

```
1
```



练习

```
var book = ["百年孤独.txt",  "狼图腾.txt", "chinaDaily.txt", "水浒传.txt","apple.doc", "time.txt",
"HAHA.txt"];
```

有一个数组,分别输出所有的中文书籍和英文书籍

谢　　谢