正则对象

1. 概述
2. 新建正则表达式对象
3. 使用字面量 var reg = /xyz/; 编译时新建表达式
4. RegExp构造函数 var reg = new RegExp(‘xyz’); 运行时新建表达式
5. RegExp构造函数可接受第二个参数，表示修饰符

var reg = /xyz/ 等价于 var reg = new RegExp(‘xyz’， ‘i’)

1. 正则对象的两种使用方式

正则对象的方法： 将字符串作为参数 reg.test(string);

字符串对象的方法：将正则对象作为参数 string.match(reg);

1. 正则对象的属性和方法

属性：

1. 与修饰符相关，返回一个布尔值，表示对应的修饰符是否设置（只读属性）

》ignoreCase : 是否设置了i修饰符

》global ：是否设置了g修饰符

》multiline : 是否设置了m修饰符

1. 与修饰符无关

》lastIndex : 返回下一次开始搜索的位置，可读写，只在设置了g修饰符时有意义

》source ：返回正则表达式的字符串形式（不包括反斜杠），只读

方法：

1. test() 返回一个布尔值，表示当前模式是否能匹配参数字符串

若正则表达式带有g修饰符，则test方法每次都从上一次结束为止开始向后匹配。

可设置lastIndex的值。

1. exec() 返回匹配结果。

若匹配成功，则返回一个数组，成员是每一个匹配成功的子字符串，否则返回null。

1. 字符串对象的方法中与正则对象有关的四种方法

》match（） : 返回数组，成员是所有匹配的子字符串， 失败返回null

（一次性返回所有匹配成功结果， 所以lastIndex属性无效）

》search（） ： 返回一个整数，表示匹配开始的位置， 失败返回-1

（该方法忽略lastIndex属性）

》replace（） ： 按照给定的正则表达式进行替换，返回替换后的字符串

》split（） ： 按照规则分割字符串，返回一个数组，包含分割后的成员

Var str= ‘abcadfsa’, reg = /a/g; match()

Var arr = str.match( reg ); // arr = [“a”, “a”, “a”] 可用来计算某字符串中某字符串的个数

‘\_x\_x’.search(/x/); // 1 search()

Str.replace(search, replacement); 参数 1：搜索模式 2：替换的内容 replace

搜索模式若不加g修饰符，就替换第一个匹配成功的值，否则替换所有匹配成功的值。

‘abca’.replace(‘/a/g’, ‘g’); // gbcg

Var str = ‘ #div .red ’;

Str.replace(/^\s+|\s+$/g, ‘’); // 消除字符串首尾两端的空格

Replace方法的第二个参数：可以用$,呆滞所替换的内容

》$& :匹配的子字符串

》$` :匹配结果前面的文本

》$n :匹配成功的d第n组内容，n从1开始，自然数

》$’ :匹配结果后面的文本

》$$ :美元符号

1. 匹配规则
2. 字面量字符和元字符

字面量字符：在正则表达式中，代表字面含义的字符（如：a, b, c）

元字符： 在正则表达式中，有特殊含义的字符。

元字符介绍

行分隔符

回车

1. 点字符（.） 匹配除\r 、\n 、\u2028、\u2029以外所有字符。

换行隔符

段分隔符

1. 位置字符：提示字符所处的位置

》^ : 字符串开始位置

》$ : 字符串结束位置

/^test/.test(‘test123’); // true test必须在开始位置

/test$/.test(‘new test’); // true test必须在结束位置

/^test$/.test(‘test’); // true 从开始到结束只有test

/^test$/.test(‘test test’); // false 从开始到结束只有test

1. 选择符（|）：表示或关系

/11|22/.test(‘test 11’); // true

多个选择符可以联合使用。

/apple|banana|watermelon/.test(‘test apple’); // true

/a( |\t)b/.test(‘a\tb’); // true a和b之间有一个空格或一个制表符

1. 其他元字符

\\、\\*、+、?、（） 、[ ]、{ }

1. 转义符

正则表达式中那些有特使含义的字符，如果要匹配他们本身，就要在他们前面加反斜杠转义。

12个需要转义的字符：^ . [ $ ( ) | \* + ? { \\

/1+1/.test(‘1+1’); // false

/1\+1/.test(‘1+1’); // true

注：若使用RegExp方法生成正则对象，转移使用两个斜杠，因为字符串内部会先转义一次。

/new RegExp(‘1\+1’)/.test(‘1+1’); // false

new RegExp(‘1\\+1’).test(‘1+1’); // true

1. 特殊字符 针对不能打印的特殊字符的表达方法

》 \cX : Ctrl-[x] x:A-Z中任一英文字母，用来匹配控制字符

》 [\b] : 匹配退格键（U+0008），不是\b

》 \n : 换行键

》 \r : 回车

》 \t : 制表符tab（U+0009）

》 \v : 垂直制表符（U+000B）

》 \f : 换页符（U+000C）

》 \0 : null字符（U+0000）

》 \xhh : 匹配一个以两位十六进制数（\x00 - \xFF）表示的字符

》 \uhhhh : 匹配一个以四位十六进制数（\u0000 - \uFFFF）表示的unicode字符

1. 字符类

字符类：有一系列字符可供选择，只要匹配其中一个就可以了。所有可供选择的字符都放在[]里。

如：[xyz] : 表示x, y, z之中任选一个匹配。

有两个字符在字符类中有特殊含义

1. 脱字符（^） 若[ ]内第一个字符为^, 则表示除了字符类中的字符，其他都可以匹配。

[^xyz] : 除了x、y、z

[^abc].test(‘hello world’); // true 包含a、b、c之外的字符

[^abc].test(‘bbc’); // false 不包含a、b、c之外的字符

[^] 匹配一切字符，包括换行符 （.）不包括换行符

[^abc].test(‘bbc’); // false 不包含a、b、c之外的字符

1. ad