妙味课堂——正则表达式

js 妙味课堂 前端基础 学习笔记

2014-11-20

把火星语——正则表达式的基本知识学习了一遍。但是离着熟练使用还是有一定距离的吧，算是入门了。

##复习字符串操作

- indexOf 查找
- substring 获取子字符串
- charAt 获取某个字符
- split 分割字符串，获得数组

##找出字符串中的所有数字

用传统字符串操作完成

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>复习字符串的操作</title>
        <script>
                //alert('a'<'b'); //true 字符串比较按照字符串的编码进行比较
```

```
                /* 判断一个字符串是否是数字类型的字符串
                var str = '5';
                if(str <= '9' && str >= '0'){ //如果str满足条件的话，那么它必然是一个数字类型的字符串
                        alert('是数字类型的字符串');
                } else {
                        alert('不是数字类型的字符串');
                }
                */

                //找出字符串中的所有数字
                var str = 'afg7agklha6shh5ngj76dgha4ssfhjt879';

                function findNum(){
                        var arr = [];
                        var tmp = '';
                        for(var i=0; i<str.length; i++){
                                if(str.charAt(i) <= '9' && str.charAt(i) >= '0'){
                                        //arr.push(str.charAt(i));
                                        tmp += str.charAt(i);
                                } else {
                                        if(tmp){
                                                arr.push(tmp);
                                                tmp = '';
                                        }
                                }
                        }

                        //针对最后的一个数字，添加进来
                        if(tmp){
                                arr.push(tmp);
                                tmp = '';
                        }

                        return arr;
                }

                alert(findNum(str));

        </script>
</head>
<body>

</body>
</html>
```

用正则表达式完成

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>复习字符串的操作</title>
        <script>
                var str = 'afg7agklha6shh5ngj76dgha4ssfhjt879';
                function findNum(str){
                        return str.match(/\d+/g);
                }
                alert(findNum(str));
        </script>
</head>
<body>
</body>
</html>
```

##什么是正则表达式

- 什么叫"正则"
  - 规则、模式

- 强大的字符串匹配工具

##正则的写法

- new RegExp("a")
- /a/

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title></title>
        <script>
                var arr = [];
                var arr = new Array();

                var obj = {};
                var obj = new Object();
```

```
                //正则：
                var re = /a/; //如果正则里面为空，浏览器会认为它是注释，因此正则简写的方式一定不能为
                var re = new RegExp('a');
        </script>
</head>
<body>
</body>
</html>
```

##正则表达式常用方法

###test

- 字符串判断
  - 返回真假
  - 正则.text(字符串)
  - 例子：是否有不是数字的字符

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>正则的方法.test</title>
        <script>
                // test : 正则去匹配字符串，如果匹配成功就返回真；如果匹配失败，就返回假
                // test的写法 : 正则.test(字符串)

                /*
                var str = 'abcdef';
                var re1 = /b/;
                var re2 = /w/;
                var re3 = /bc/;
                var re4 = /bd/;
                alert(re1.test(str)); //true
                alert(re2.test(str)); //false
                alert(re3.test(str)); //true
                alert(re4.test(str)); //false
                */

                /*
                //转义字符：
                alert('anb');
```

```
                alert('a\nb'); //这里的n前面加了反斜线，代表换行

                n \n  换行
                r \r  制表
                t \t  回车
                */

                /*
                        \s : 空格
                        \S : 非空格
                        \d : 数字
                        \D : 非数字
                        \w : 字符 （字母、数字、下划线 _）
                        \W : 非字符
                */

                // var str = '473t9487306';
                // var re = /t/;
                // if(re.test(str)){
                //      alert('不全是数字');
                // } else {
                //      alert('全是数字');
                // }

                var str = '473t9487306';
                var re = /\D/;
                if(re.test(str)){
                        alert('不全是数字');
                } else {
                        alert('全是数字');
                }
        </script>
</head>
<body>
</body>
</html>
```

###search

- 字符串搜索
  - 返回出现的位置
  - 字符串.search(正则)
  - 忽略大小写：i——ignore

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>正则的方法.search</title>
        <script>
                // search : 正则去匹配字符串，如果匹配成功，就返回匹配成功的位置；如果匹配失败，就返
                // search的写法 : 字符串.search(正则)
                // 正则中的默认：是区分大小写的
                // 如果不区分大小写的话，在正则的最后加标识 i （不区分大小写）

                var str = 'abcdef';
                var re1 = /bcd/;
                var re2 = /w/;
                var re3 = /B/;
                var re4 = /B/i;
                var re5 = new RegExp('B', 'i'); //这一行的写法与上面一行的写法，意义完全一样
                alert(str.search(re1)); //弹出1 弹的是匹配的首字母的位置
                alert(str.search(re2)); //弹出-1
                alert(str.search(re3)); //弹出-1 //正则默认下是区分大小写的
                alert(str.search(re4)); //弹出1
                alert(str.search(re5)); //弹出1

        </script>
</head>
<body>
</body>
</html>
```

###match

- 获取匹配的项目
  - 返回数组
  - 量词：+
  - 全局匹配：g——global
  - 例子：找出所有数字

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>正则的方法.match</title>
```

```
        <script>
                // match : 正则去匹配字符串，如果匹配成功，就返回匹配成功的数组；如果匹配失败，就返回
                // match的写法 : 字符串.match(正则)
                //正则默认：正则匹配成功就会结束，不会继续匹配
                //如果想全部查找，就要加标识 g（全局匹配）
                //量词：匹配不确定的位置
                // + : 至少出现一次

                var str = 'sdogh3woiehgxfkjb789paf34dj4pgdfhgdorg43';
                var re1 = /\d/;
                var re2 = /\d/g;
                var re3 = /\d\d/g;
                var re4 = /\d\d\d/g;
                var re5 = /\d\d\d\d/g;
                var re6 = /\d+/g; //这里说明前面的反斜杠d至少出现一次，多则不限
                alert(str.match(re1)); //[3] 返回的是找到的第一个数字
                alert(str.match(re2)); // [3, 7, 8, 9, 3, 4, 4, 4, 3]
                alert(str.match(re3)); //[78, 34, 43]
                alert(str.match(re4)); //[789]
                alert(str.match(re5)); // null
                alert(str.match(re6)); //[3, 789, 34, 4, 43]
        </script>
</head>
<body>
</body>
</html>
```

###replace

- 替换所有匹配
  - 返回替换后的字符串
  - 字符串.replace(正则,想替换的)
  - 例子：敏感词过滤
  - 匹配子项
  - 例子：日期格式化

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>正则的方法.replace</title>
        <script>
```

```
                // replace : 正则去匹配字符串，匹配成功的字符去替换成新的字符串
                // replace的写法 : 字符串.replace(正则, 新的字符串)

                var str1 = 'aaa';
                var str2 = 'aaa';
                var str3 = 'aaa';
                var re1 = /a/;
                var re2 = /a/g;
                var re3 = /a+/g;

                str1 = str1.replace(re1, 'b');
                alert(str1); //'baa'
                str2 = str2.replace(re2, 'b');
                alert(str2); //'bbb'
                str3 = str3.replace(re3, 'b');
                alert(str3); //'b'
        </script>
</head>
<body>
</body>
</html>
```

敏感词过滤实例

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>用replace方法实现敏感词过滤</title>
        <script>
                //敏感词包括“菲称”、“中国船”、“监视之下”
                // | : “或”的意思
                // replace : 第二个参数；可以是字符串，也可以是一个回调函数

                window.onload = function(){
                        var aT = document.getElementsByTagName('textarea');
                        var oInput = document.getElementById('input1');
                        var re = /菲称|中国船|监视之下/g;

                        oInput.onclick = function(){

                                //aT[1].value = aT[0].value.replace(re, '*');
                                // aT[1].value = aT[0].value.replace(re, function(){
```

```
                                //      return '*';
                                // }); //这个代码段与上面那一行代码的意义是一模一样的
                                aT[1].value = aT[0].value.replace(re, function(str){ //函数的第一个参
                                        //alert(str);
                                        var result = '';

                                        for(var i=0; i<str.length; i++){
                                                result += '*';
                                        }
                                        return result;
                                });

                        }
                }
        </script>
</head>
<body>
        替换前<br />
        <textarea>菲称仁爱礁附近17艘中国船均在菲军监视之下</textarea><br>
        替换后<br>
        <textarea></textarea>
        <input type="button" value="确定" id="input1">
</body>
</html>
```

匹配子项

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>匹配子项</title>
        <script>
                // 匹配子项 : 小括号 () （还有另外一个意思：分组操作）
                // (1+1)*2 和 1+1*2
                //把正则的整体叫做（母亲）
                // 然后把左边第一个小括号里面的正则，叫做这个第一个子项（母亲的第一个孩子）
                // 第二个小括号就是第二个孩子

                var str1 = '2013-6-7';
                var str2 = '2013-6-7';
                var str3 = '2013-6-7';
                var str4 = '2013-6-7';
```

```
                var str5 = '2013-6-7';
                var re1 = /\d+-/g;
                var re2 = /\d-+/g; //一个数字加至少一个横杠
                var re3 = /(\d-)+/g;
                var re4 = /(\d+)(-)/g;
                var re5 = /(\d+)(-)/g;

                // str1.replace(re1, function($0){
                //      alert($0); //2013- 和 6-
                // })

                // str2.replace(re2, function($0){
                //      alert($0); //3- 和 6-
                // })

                // str3.replace(re3, function($0){
                //      alert($0); //3-6-
                // })

                /*
                str4.replace(re4, function($0, $1, $2){
                        // 第一个参数 : $0 （母亲）
                        // 第二个参数 : $1 （第一个孩子）
                        // 第三个参数 : $2 （第二个孩子）
                        // 以此类推
                        alert($0); //2013- , 6-
                        alert($1); //2013 , 6
                        alert($2); // - , -
                        //运行一次弹出：2013-, 2013, -, 6-, 6, -
                })
                */

                str5 = str5.replace(re5, function($0, $1, $2){
                        // return $0.substring(0, $0.length - 1) + '.';
                        return $1 + '.'; //2013.6.7
                })

                alert(str5); //2013.6.7
        </script>
</head>
<body>
</body>
</html>
```

match方法中的匹配子项

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>匹配子项</title>
        <script>
                var str1 = 'abc';
                var re1 = /abc/;
                var str2 = 'abc';
                var re2 = /(a)(b)(c)/;
                var str3 = 'abc';
                var re3 = /(a)(b)(c)/g;

                alert(str1.match(re1)); //[abc]
                alert(str2.match(re2)); //[abc, a, b, c]（当match不加g的时候才可以获取到匹配子项的集
                alert(str3.match(re3)); //[abc] //这里因为re里面加个g，就不会获得上面那一句的结果了
        </script>
</head>
<body>
</body>
</html>
```

##正则表达式字符类

###任意字符

- [abc]
  - 例子：o[usb]t——obt、ost、out

###范围

- [a-z]、[0-9] -例子：id[0-9]——id0、id5

###排除

- [^a]
  - 例子：o[^0-9]t——oat、o?t、o t

字符类

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>正则表达式中的字符类</title>
        <script>
                //字符类 : 一组相似的元素或字符
                //字符类 : 用中括号表示，中括号的整体代表一个字符

                var str1 = 'abc';
                var re1 = /a[bde]c/;

                alert(re1.test(str1)); //true

                var str2 = 'aec';
                var re2 = /a[bde]c/;

                alert(re2.test(str2)); //true

                var str3 = 'abdc';
                var re3 = /a[bde]c/;

                alert(re3.test(str3)); //false

                //排除 : ^ 如果^写在[]里面的话，就代表排除的意思

                var str4 = 'abc';
                var re4 = /a[^bde]c/;

                alert(re4.test(str4)); //false

                var str5 = 'awc';
                var re5 = /a[^bde]c/;

                alert(re5.test(str5)); //true

                //范围 : - 如果-写在[]里面的话，就代表范围，范围一定是从小到大的

                var str6 = 'abc';
                var re6 = /a[e-j]c/;

                alert(re6.test(str6)); //false

                var str7 = 'afc';
                var re7 = /a[e-j]c/;
```

```
                alert(re7.test(str7)); //true

                var str8 = 'a3c';
                var re8 = /a[a-z0-9A-Z]c/;

                alert(re8.test(str8)); //true


                var str9 = 'a3eJ86Xc';
                var re9 = /a[a-z0-9A-Z]+c/; //在中括号外面添加了+，代表可以多位

                alert(re9.test(str9)); //true
        </script>
</head>
<body>
</body>
</html>
```

###组合

- [a-z0-9A-Z]

###实例：偷小说

- 过滤HTML标签 -自定义innerText方法

字符类实例，过滤标签

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>字符类：过滤标签</title>
        <script>
                window.onload = function(){
                        var aT = document.getElementsByTagName('textarea');
                        var oInput = document.getElementById('input1');

                        oInput.onclick = function(){

                                //var re = /<\w+>/g; 这个正则不认反斜杠和引号等特殊字符
```

```
                                var re = /<[^>]+>/g;
                                aT[1].value = aT[0].value.replace(re, '');

                            }
                }
        </script>
</head>
<body>
        替换前<br />
        <textarea><h3 class="box">标题</h3>aaaaaa</textarea><br>
        替换后<br>
        <textarea></textarea>
        <input type="button" value="确定" id="input1">
</body>
</html>
```

##转义字符

- .(点)——任意字符
- \d、\w、\s、\b
- \D、\W、\S、\B
- \1 重复子项
- 例子
  - 获取class元素
  - 找重复项最多的字符和个数

转义字符

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title></title>
        <script>

                // 转义字符
                // . : 任意字符
                // \. : 真正的点

                var str1 = 'a你c';
                var re1 = /a.c/;

                alert(re1.test(str1)); //true
```

```
                var str2 = 'a你c';
                var re2 = /a\.c/;
                alert(re2.test(str2)); //false

                // \b : 独立的部分（起始、结束、空格）
                // \B : 非独立的部分

                var str3 = 'onetwo';
                var re3 = /one/;
                alert(re3.test(str3)); //true;

                var str4 = 'onetwo';
                var re4 = /\bone/; //代表字母o前面必须是独立部分，o前面是起始位置
                alert(re4.test(str4)); //true;

                var str5 = 'onetwo';
                var re5 = /one\b/; //e后面是个t，不是个独立部分
                alert(re5.test(str5)); //false;

                var str6 = 'one two';
                var re6 = /one\b/; //e后面是个空格，是独立部分
                alert(re6.test(str6)); //true;

                var str7 = 'onetwo';
                var re7 = /\bone\b/; //前面满足后面不满足，整体也就不满足
                alert(re7.test(str7)); //false;

        </script>
</head>
<body>
</body>
</html>
```

获取class元素

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>获取class方法</title>
        <script>
```

```
                //要找到class为box1的元素，也就是第一个和最后一个li匹配

                window.onload = function(){
                        var aLi = getByClass(document, 'box1');

                        for(var i=0; i<aLi.length; i++){
                                aLi[i].style.background = 'red';
                        }

                        /* 老方法，有问题，只能匹配第一个li
                        function getByClass(oParent, sClass){
                                var arr = [];
                                var aEle = oParent.getElementsByTagName('*');
                                for(var i=0; i<aEle.length; i++){
                                        if(aEle[i].className == sClass){
                                                arr.push(aEle[i]);
                                        }
                                }
                                return arr;
                        }
                        */

                        //改进版
                        function getByClass(oParent, sClass){
                                var arr = [];
                                var aEle = oParent.getElementsByTagName('*');
                                //当正则需要传参的时候，一定要用全称的写法
                                // var re = new RegExp(sClass); //用这个正则，1、3、5都能找到

                                var re = new RegExp('\\b'+sClass+'\\b'); //注意，这里要写两个反斜杠才

                                for(var i=0; i<aEle.length; i++){
                                        if(re.test(aEle[i].className)){
                                                arr.push(aEle[i]);
                                        }
                                }
                                return arr;
                        }
                }

        </script>
</head>
<body>
        <ul>
                <li class="box1">111</li>
```

```
                <li>111</li>
                <li class="box1box2">111</li>
                <li>111</li>
                <li class="box1 box2">111</li>
        </ul>
</body>
</html>
```

重复子项

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>重复子项</title>
        <script>

                //////////////////////
                // \1 : 重复的第一个子项
                // \2 : 重复的第二个子项
                // 以此类推
                //////////////////////

                var str1 = 'abca';
                var re1 = /(a)(b)(c)\1/;

                alert(re1.test(str1)); //true

                var str2 = 'abca';
                var re2 = /(a)(b)(c)\2/;
                alert(re2.test(str2)); //false

                var re3 = /\w\w/; //ab可以匹配成功
                var re4 = /\w\1/; //c9无法匹配成功，只能cc或99这种才能匹配成功

        </script>
</head>
<body>
</body>
</html>
```

找重复项最多的字符和个数

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>找重复项最多的字符和个数</title>
        <script>

                var str = 'algkjaljgdsskglakjldssslkgjlslsjgkjsls';

                var arr = str.split('');
                str = arr.sort().join('');

                // alert(str);

                var value = '';
                var index = 0;

                var re = /(\w)\1+/g;

                str.replace(re, function($0, $1){
                        //alert($0);
                        if(index < $0.length){
                                index = $0.length;
                                value = $1;
                        }
                });
                alert('最多的字符：' + value + '，重复的次数：' + index);
        </script>
</head>
<body>
</body>
</html>
```

##量词

####什么是量词

- 出现的次数
- {n, m}，至少出现n次，最多m次
- 例子：查找QQ号

量词

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>量词</title>
        <script>
                // 量词 : {}
                // {4, 7} : 最少出现4次，最多出现7次
                // {4, } : 最少出现4次
                // {4} : 正好出现4次
                // + : {1,} 至少出现1次
                // ? : {0, 1} 出现0次或1次
                // * : {0, } 至少出现0次

                var str1 = 'ab';
                var re1 = /ab+/;

                alert(re1.test(str1)); //true


                var str2 = 'ac';
                var re2 = /ab+/;

                alert(re2.test(str2)); //false

                var str3 = 'ac';
                var re3 = /ab*/;

                alert(re3.test(str3)); //true
        </script>
</head>
<body>
</body>
</html>
```

查找QQ号

```

```

####常用量词

- {n,} 至少n次

- • 任意次 {0,}

- ? 零次或一次 {0,1}
- • 一次或任意次 {1,}

- {n} 正好n次

##正则收尾

- ^ 开始
- $ 结束
  - 例子
    - 是不是QQ号
    - 去掉前后空格

判断是不是QQ号

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>查找QQ号</title>
        <script>

                // ^ : 放在中括号里面代表排除
                // ^ : 放到正则的最开始位置，就代表起始的意思
                // $ : 放在正则的最后位置，就代表结束的意思

                window.onload = function(){
                        var aInput = document.getElementsByTagName('input');
                        var re = /^[1-9]\d{4,11}$/; //这里的规则是首字母不为零，总共有5-7位数字；起始

                        aInput[1].onclick = function(){

                                if(re.test(aInput[0].value)){
                                        alert('是QQ号');
                                } else {
                                        alert('不是QQ号');
                                }
                        }
                }
        </script>
</head>
```

```
<body>
        <input type="text"><input type="button" value="确定">
</body>
</html>
```

去掉前后空格

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>去掉前后空格</title>
        <script>
                var str = ' hello ';
                alert('(' + trim(str) + ')');

                function trim(str){
                        var re = /^\s+|\s+$/g; // \s代表空格，在开始的时候空格出现至少一次或者在结尾
                        return str.replace(re, '');
                }
        </script>
</head>
<body>
</body>
</html>
```

##常用正则例子

###高级表单校验

- 匹配中文：[\u4e00-\u9fa5]
-

行首行尾空格：^\s*	\s*$

- Email：^\w+@[a-z0-9]+(.[a-z]+){1,3}$
- 网址：[a-zA-z]+://[^\s]*
- QQ号：[1-9][0-9]{4,9}
- 邮政编码：[1-9]\d{5}
-

身份证：[1-9]\d{14}	[1-9]\d{17}	[1-9]\d{16}x