

day14-javascript 正则表达式

笔记本： 凡尘：二阶段

创建时间： 2017/8/27 10:38

更新时间： 2017/9/27 17:47

作者： 437389128@qq.com

正则表达式

001、javascript最初诞生的目的是什么？

002、什么是正则

- 0、一种高效处理字符串的规则，又称规则表达式
- 1、正则表达式(regular expression)描述了一种字符串匹配的模式
- 2、正则表达式是由普通字符（例如字符 a 到 z）以及特殊字符（称为"元字符"）组成
- 3、正则表达式是对字符串操作的一种逻辑公式，就是用事先定义好的一些特定字符、及这些特定字符的组合，组成一个“规则字符串”，这个“规则字符串”用来表达对字符串的一种过滤逻辑（匹配）

003、为什么要使用正则？

- 1、普通表单验证方法太复杂，效率低下
- 2、正则写法简单，灵活，效率高，使用非常方便

004、正则可以做什么？

- 1、正则表通常被用来**查找、替换**字符串
- 2、从字符串中**提取**出想要的信息

005、创建正则表达式

- 1、字面量形式（常用）perl风格 古老的语言

/表达式/ 第一个/表示正则开始 第二个/表示正则结束

返回结果：正则.test(字符串); 字符串复合规则返回true,否则返回false

栗子：判断字符串里面是否有a

```
var reg = /a/;
```

```
reg.test('abcd')//true
```

```
reg.test('123rrr')//false
```

2、构造函数 使用new===>一般封装的时候使用，不知道要具体匹配什么的时候

```
var reg = new RegExp(表达式,参数);
```

参数：i是否忽略大小写 g是否全局匹配

match:提取所以得想要的东西

栗子：

```
reg = new RegExp('a');
```

```
var str = 'abcdas';
```

```
alert(str.match(reg))
```

元字符

正则中使用表示特定含义的字符

比如：^表示从字符串的第一个字符开始匹配

问题

```
var str = 'abcde';
```

1、怎么判断一个字符串是不是以a开头？

2、怎么判断一个字符串是不是以e结尾？

```
a、/^a/.test('')
```

```
b、/a$/.test("")
```

^和\$(重)

```
var str = 'abc123';
```

- 1、^开始标记
/^a/ :表示判断字符串是不是从a开始
- 2、\$结束标记
/3\$/ :表示判断字符串是不是以3结尾

问题

```
var str = 'abcde12345';
```

- 1、怎么判断一个字符串是不是包含字母？
- 2、怎么判断一个字符串是不是包含数字？

```
/[a-zA-Z]/.test()
```

```
/[0-9]/.test()
```

[]匹配集合（重）

[]代表匹配一个字符字符集合。匹配包含的任意一个字符
例如：[abc]匹配str = 'afsada'中的a

```
var str = 'abcde12345';
```

- 1、怎么判断一个字符串是不是包含字母？
`var reg = /[abcdef.....zABCD....Z]/ =====> [a-zA-z]`
- 2、怎么判断一个字符串是不是包含数字？
`var reg = /[0123456789]/`

`=>var reg = /[0-9]/;`
`=>var reg = /\d/`

问题

```
var str = 'abcde12345';
```

1、怎么判断一个字符串开头五个字符是不是都是字母？

2、怎么判断一个字符串最后五个字符是不是都是数字？

a、`/^[a-zA-Z]{5}/.test()`

b、`/[0-9]{5}$/.test()`

{ }、+、*、? (重要)

1、**{n}**:n是一个非负数。匹配确定的n次

栗子：`/o{2}/`不能匹配bob中的o,但是可以匹配foot中的两个o

{n}:n是一个非负数。至少n次的匹配

{n,m}:n与m均为非负数，其中 $n \leq m$.最少匹配n次且最多匹配m次

2、**+** 匹配前面的子表达式一次或多次（大于等于1次）

栗子：`"zo+"`能匹配'zo'以及"zozoo"，但是不能匹配z

+等价于{1,}

3、***** 匹配前面的子表达式任意次

栗子：`zo*`能匹配'z',也能匹配'zo'以及'zoo'

*****等价于{0,}

4、**?** 匹配前面子表达式零次或者一次

栗子：`'dog?'`可以匹配'do'或者'dogs'中的do

?等价于{0, 1}

练习

1、判断一个数是不是6的重叠数（66、666、6666...）

2、判断一个字符串是不是包含ab或者abc

3、判断一个字符串是不是全部是字母并且是3-6位之间

```
a、 /^6{2,}$/.test();  
b、 /(ab|abc)/.test();  
c、 /^[a-zA-Z]{3,6}$/.test()
```

\w(数字字母下划线)(重)

\w: 匹配数字字母下划线的组合

如: 如何判断用户名为数字字母下划线

```
var reg = /^[a-zA-z0-9]+$/;  
var reg = /\w+$/
```

. 任意字符

点. : 匹配除'\r \n'之外的任何单个字符

注意: 如果想匹配有没有点(.)需要转义字符\

练习

1、判断一个数是不是小于0的数

```
/^-{1}\d+(\d*|\.? \d+)$/ .test('-1.')  
false  
/^-{1}\d+(\.? \d+)?$/ .test('-1.')  
false
```

\b 单词边界符

\b : 匹配一个单词边界, 也就是只单词和空格间的位置

```
var str = my name is
```

例如, 'er\b' 可以匹配never hahaha中的er, 但是不能匹配到very中的er

\s 空格

\s : 匹配一个空格

栗子:

```
var reg = /item\s/;

reg.test('item item')//true

reg.test('item1')//false
```

练习

兼容:document.getElementsByClassName()

```
function getByClass(oParent, sClass)
{
    var aEle=oParent.getElementsByTagName('*');
    var aResult=[];
    var re=new RegExp('\\b'+sClass+'\\b', 'i');
    var i=0;

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

    return aResult;
}
```

| 或模式匹配

| 将两个匹配条件进行逻辑运算

栗子: var reg = /(him)|(hre)/;

匹配"it belongs to him"和"it belongs to her",

但是不能匹配"it belongs to them."

var reg = /zhao|ya|hua/; 匹配三种其中一个字符串

var str = 'zhao ya hua';

reg.test(str);//true

建议: 使用 | 前后的内容要加 () , 以免出现不必要的bug

练习

- 1、判断一个数是不是小数
- 2、判断一个数是不是合法的微信红包金额（0-200）

a、`/^((0\.\d)|([1-9]+\.\d+))$/`

b、`/^\d{1,3}(\.\d{1,2})?$/`

g 全部匹配

```
g 匹配整个字符串（所有符合规则的字符串都匹配）
global
var str = '我很困，你们更困！';
console.log( str.replace('困','精神') );
var reg = /困/g;
console.log( str.replace(reg,'精神') ); // '我很精神，你们更精神！'
console.log( str.replace(reg,'不困') );
```

i 忽略大小写

```
i 忽略字母的大小写

var v = 'Abc6'; //abc6 ABC6 Abc6

/ab/i 能匹配 Ab ab AB aB

var verify = '0Ab5'; //0ab5 0Ab5
reg = /^0ab5$/i;
console.log( reg.test(verify) );
```

正则替换（重）

```
str.replace(reg, 替换内容)

replace 使用正则进行替换
str.replace(正则表达式, 替换效果)

var reg = /xiaoming/;
var str = /good xiaoming!xiaoming good /;

i 忽略大小写
var reg = /xiaoming/i

g 全部匹配, 替换所有

var reg = /xiaoming/g
```

练习

验证码匹配

```
var str='abFc'  
  
/^(abfc)$/i.test()
```

综合练习

检查邮政编码//共 6 位数字，第一位不能为 0

检查文件压缩包 // xxx.zip xxx.gz xxx.rar

电子邮件(xxxxx @ xxxx(.xxxx)+) xxx@xx.com.cn

a、 `/^[1-9]\d{5}$/`

b、 `/^.\+.(zip|gz|rar|7z|tar)$/.test("");`

c、 `/^\w+@\w+(\.\w+)+$/`

常用正则实战

- 1、用户名验证 (由数字、字母、下划线组成，不能以数字开头，长度为6-16位)
- 2、手机号验证 (11位数字，以13|18|15|17开头)
- 3、日期检测 (格式为 xxxx-xx-xx 或者 xxxx/xx/xx)
- 4、身份证验证 (身份证号码为15位或者18位，15位时全为数字，18位前17位为数字，最后一位是校验位，可能为数字或字符X)
- 5、中文验证 (中文汉字范围 `[\u4e00-\u9fff]`)
 - a、 `var reg = /^[a-zA-Z_]\w{5,15}$/;`
 - b、 `/^(13|18|15|17)\d{9}/.test()`
 - c、 `/^\d{4}((-|\w|\.)\d{2}){2}$/`
 - d、 `/^[1-9]((\d{14})|(\d{16}(\d{1}|X{1})))$/`
 - E、 `/[^\u4e00-\u9fa5]/g`

脚本工具网站

<http://tools.jb51.net/>

正则高级

- 1、`()` 分组模式匹配
正则中使用`()`括起来的部分称之为`一组正则`

常用于匹配一组字符或提取内容

- 2、`\1 \2 ... \n`
`\1`代表正则第一个括号匹配时的内容
`\2`代表正则中第二个括号的内容

如实现不重叠数字正则：
`var reg = /^(\d)\1+$/;`

栗子：

```
var str = 'a33333';  
  
var reg = /^(\d)\1(\d)+$/;  
  
console.log(reg.test(str));
```

- 3、`$1 $2 ... $n`
字符串替换时用于表示第几个括号的内容

栗子：
`var str = '2017/04/06';`
`var reg = /^(\d{4})\/(\d{2})\/(\d{2})$/;`
`str = str.replace(reg, "$3/$2/$1");`
`console.log(str);`

正则高级（扩展）

`[^abc]` 查找不在方括号中的字符

`[^\d]` 匹配不是数字的字符（当前字符不是数字）

正则查找

`str.search(正则表达式)`

`search` 使用正则查找

```
var pattern = /good/ig;  
var str = 'good good study! day day up!';  
//找到返回下标，否则返回-1  
alert(str.search(pattern));
```

`indexOf`不可以使用正则

正则提取

`var reg = /(需要提取的内容)/;` 将需要提取的内容使用括号括起来

`reg.exec(字符串);` 根据正则提取字符串中的内容

`exec()` 方法用于检索字符串中的正则表达式的匹配。

返回一个数组，其中存放匹配的结果。如果未找到匹配，则返回值为 `null`。

返回值解析：

```
var reg = /^(\d{6})(\d{4})(\d{2})(\d{2})\d{3}[\dX]$/;
```

数组下标：

0: 正则匹配的内容

1: 第一个括号的内容

2: 第二个括号的内容

index: 正则匹配成功时第一个字符的下标

input: 匹配的字符串

```
var str = '41242419910101123X';
var reg = /^(\d{6})(\d{4})(\d{2})(\d{2})\d{3}[\dX]$/
reg.exec(str)
str.match(reg)
```

match与exec的区别

1、`exec`是正则表达式的方法，而不是字符串的方法，它的参数才是字符串，如下所示：

2、`match`是字符串执行匹配正则表达式规则的方法，他的参数是正则表达，如

3、`exec`和`match`返回的都是数组：

实战

1、提取身份证中的年月日

2、使用正则提取出`phone`或者`username`或者`address`的值

```
var str = 'username=pine; phone=18270837879; address=henan';
```

预习

- 1、严格模式
- 2、`let/const`
- 3、ES6 变量结构赋值
- 4、ES6 字符串新增方法
- 5、`=>` 函数 （箭头函数）
- 6、Set 和 Map 结构