## 正则表达式

1. **复习字符串方法 以及引入正则**
   1. 字符串方法
      1. indexOf 用来查找对应字符的位置
      2. Substring 用来截取子字符
      3. charAt 在字符串中根据下标去找对应的字符
      4. Split 将字符串通过分割符转成数组
   2. 练习题:
      1. var str = 'haj123sdk54hask33dknalsd789'; // 得到所有的数字 并将相连的数字进行相连存储 如 ["123","54","33","789"]
         1. 传统方法：

var arr = [];

// 建立一个临时变量 用来拼接连续的数字字符串

var temp = "";

for(var i = 0; i < str.length; i++) {

if(str.charAt(i)>="0"&&str.charAt(i)<=9) {

// 进来之后肯定是数字

temp += str[i];

} else {

if (temp) {

arr.push(temp);

// 清空这个临时变量 以便于下一次重新装载 连续的数字

temp = "";

}

}

}

// 防止 最后一些字符 是数字 不进else 就拼不进去了 所以这个在最后要判断如果temp里面还有 则在推进去一次

if (temp) {

arr.push(temp);

temp = "";

}

console.log(arr);

* + - 1. 正则方法：

**console.log(str.match(/\d+/g));**

1. **什么是正则**
   1. 定义：正则：也叫规则，是让计算机能够读懂人类的规则，正则都是**操作字符串**的，正则是一个强大的字符串匹配工具
   2. 18611223344手机号 [1651234987@qq.com](mailto:1651234987@qq.com) qq邮箱
   3. 正则：正则表达式Regular Expression var str var arr var re regexp
      1. 正则一个js中全局内置对象 （var mydate = new Date()） php (正则)
      2. 正则是一个工具
      3. 简单声明方式： var re = /abc/;
      4. 构造函数声明方式 var re2 = new RegExp(‘abc’);
2. **我们来学习一些几个正则匹配方法**
   1. Test():用正则去匹配的方法
      1. 返回值： 如果匹配成功 则返回true 否则匹配不成功 返回false
      2. 写法： re.test(str); 用re去匹配str

正则当中的转义字符（也叫元字符）：

1. .（点）——任意字符 如果就想只匹配.(点) 加转义字符 还原它本来的意思\.

2. \d、\D 匹配数字 / 非数字

3. \w、\W 匹配字符/非字符 （字符：字母数字下划线 不包含空格）

4. \s 、\S 空格/ 非空格

5. \b 、\B 匹配独立部分/ 非独立部分 （后面去讲）

// 查看该字符串中有没有非数字

var str = "374829348791";

var re = /\D/;

alert(re.test(str));

* 1. Search() 用正则进行匹配的方法
     1. 返回值： 如果匹配成功 则返回对应匹配的位置 否则匹配不成功返回-1
     2. 写法：str.search(re)

**注意：正则中严格区分大小写 如果你想忽略大小写**

总结：如果想不区分大小写，则需要在正则最后加标识i （ ignore简写）

// 查看该字符串中有没有字母b 不论大小写

var str = "abdcef";

var re = new RegExp('b');

alert(str.search(re))

* 1. Match()
     1. 返回值：如果匹配成功 则返回一个数组，会把匹配到字符保存到数组中
     2. 写法：str.match(re)

注意： 正则匹配：如果匹配成功 则不往后匹配了

g 全局匹配模式 （global） window nodeJs (global)

上面的i和g都是放置在第二斜杠后面

var str = 'abbbvdeBdbBzzsB'; // ["bbb","B","bB","B"]

var re = /B+/ig; // new RegExp(‘B+’,’ig’);

console.log(str.match(re));

* 1. Replace()
     1. 返回值 用正则去匹配字符串 ，将匹配到的字符进行替换 替换成新的字符
     2. 写法: str.replace(re,新字符)；
     3. 写法二： str.replace(re,function($0){

Console.log($0)

})

示例代码：

var str = 'aaa';

var re = /a/g;

console.log(str.replace(re,'b'));

var re = /a+/g; // + 是正则里面的一个量词 代表一个或多个

console.log(str.replace(re,'b'));

var str = 'abbbvdeBdbBzzsB';

var re = /B/ig;

console.log(str.replace(re,'o'));

案例：敏感词过滤

var str = '雾霾出现以后，空气极度的差，雾霾这种现象都是政府责任';

var re = /政府责任|雾霾|差/g;

str.replace(re,function($0){

console.log($0); // $0 每一次匹配到的整体字符

console.log($0.length);

// 建立一个空字符 用来累加\*号

var str = '';

// 开启一个循环 循环每一次匹配到的整体字符 循环长度

for(var i = 0; i < $0.length; i++) {

str += "\*"

}

// 返回str

return str;

})

练习：

var str = '菲称仁爱礁附近17艘中国船均在菲监视之下'; // 菲 中国船 监视之下

1. **正则表达式字符类**

1. 任意字符

[abc]

例子：o[usb]t——obt、ost、out

2. 范围

[a-z]、[0-9]

例子：id[0-9]——id0、id5

3. 排除

[^a]

例子：o0-9t——oat、o?t、o t

4. 组合

[a-z0-9A-Z] \w (字母数字下划线)

实例：偷小说

过滤HTML标签

1. **补充内容：**
   1. 独立部分和非独立部分

\b \B (以开头 结束位置 以及空格 和非字符作为独立分割)

var str = "onetwo";

var re = /\bone\b/;

re.test(str);

**六、什么是量词（也叫限定符）**

出现的次数

{n,m}，至少出现n次，最多m次

例子：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

两个定界符：

^ 开始

$ 结束

**七、常见正则：**

匹配中文：[\u4e00-\u9fa5]

行首行尾空格：^\s|\s$

Email：^\w+@[a-z0-9]+(.[a-z]+){1,3}$

var re=/^\w\*@[a-z0-9]{2,6}(.[a-z]{2,3}){1,3}$/;

网址：[a-zA-z]+://\s\*

QQ号：1-9{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

1. **正则的json管理，命名空间**

var rules = [

{

name: 'qq',

reg: /^[1-9]\d{4,11}$/,

tip: "请输入正确的QQ"

},

{

name: 'email',

reg: /^\w+@\w+\.\w+(\.\w+)?$/,

tip: "请输入正确的邮箱地址"

},

{

name: 'phone',

reg: /^((13[0-9])|(15[^4,\D])|(18[0-9]))\d{8}$/,

tip: "请输入正确的手机号码"

},

{

name: 'date',

reg: /^\d{4}-\d{1,2}-\d{1,2}$/,

tip: "请输入正确的出生日期"

},

{

name: 'cn',

reg: /^[\u4e00-\u9fa5]{2,4}$/,

tip: "请输入正确的姓名"

}];

function addCheck(element, reg, tip) {

element.onblur = function () {

//获取当前文本框对应的span

var span = this.nextElementSibling;

//判断验证是否成功

if(!reg.test(this.value) ){

//验证不成功

span.innerText = tip;

span.style.color = "red";

}else{

//验证成功

span.innerText = "";

span.style.color = "";

}

};

}

1. **总结** 
   1. **正则表达式的作用**
      1. 给定的字符串是否符合正则表达式的过滤逻辑(匹配)
      2. 可以通过正则表达式，从字符串中获取我们想要的特定部分(提取)
      3. 强大的字符串替换能力(替换)
   2. **正则表达式的特点**
      1. 灵活性、逻辑性和功能性非常的强
      2. 可以迅速地用极简单的方式达到字符串的复杂控制
      3. 对于刚接触的人来说，比较晦涩难懂
   3. **正则表达式的测试**
      1. [在线测试正则](https://c.runoob.com/front-end/854)