

SHANXI **UNIQUE**  
TECHNOLOGY CO.,LTD.



安逸客

SHANXI UNIQUE TECHNOLOGY

## 正则

JavaScript是一种能让你的网页更加生动活泼的语言

什么是正则表达式？

- 一个用来描述或者匹配一系列符合某个语法规则的字符串的语言。在很多文本编辑器或其他工具里，正则表达式通常被用来检索、替换或拆分那些符合某个模式的文本内容。许多程序设计语言都支持利用正则表达式进行字符串操作
- 应用场合
  - 数据验证、文本替换、内容检索、过滤内容
  - 可以理解为：执行字符串函数无法完成的特殊的匹配、拆分、替换功能

在javascript正则表达式也是通过对象的方式来创建的，他有自己的方法。

1. 通过构造函数创建 `reg=new RegExp( "正则表达式" ," 模式修正符" )`

```
var reg = new RegExp("uek ");  
var stat = reg.test("sxuek");  
alert(stat);
```

2. 通过字面量方式创建

```
var reg = /uek/i;  
var stat = reg.test("sxuek");  
alert(stat);
```

通常将正则表达式字符串放在 `/RegExp/` 中间//称为定界符

RegExp.test(str)

- 返回一个 Boolean 值，它指出在被查找的字符串中是否存在模式

RegExp.exec() 在字符串中匹配正则，成功返回数组，失败返回null

- 返回的数组包含特殊属性：
- input //被查找字符串 index //子字符串位置
- 如果采用g修饰符
- 如果正则表达式没有设置g，那么exec方法不会对正则表达式有任何的影响，如果设置了g，那么exec执行之后会更新正则表达式的lastIndex属性，表示本次匹配后，所匹配字符串的下一个字符的索引，下一次再用这个正则表达式匹配字符串的时候就会从上次的lastIndex属性开始匹配，也就是上面两个例子结果不同的原因了。

```
var reg=/w/;  
var str="abcdefg"  
var result=reg.exec(str)  
for (var i in result) {  
    document.write(i+":"+result[i]+"<br/>")  
}  
var result=reg.exec(str)  
for (var i in result) {  
    document.write(i+":"+result[i]+"<br/>")  
}  
var result=reg.exec(str)  
for (var i in result) {  
    document.write(i+":"+result[i]+"<br/>")  
}
```



## ➤ 什么是原子

原子是正则表达式中的最小的元素，包括英文、标点符号等

- `\d` 匹配任意一个数字 [0-9]
- `\D` 与除了数字以外的任何一个字符匹配 [^0-9]
- `\w` 与任意一个英文字母,数字或下划线匹配 [a-zA-Z\_0-9]
- `\W` 除了字母,数字或下划线外与任何一个字符匹配 [^a-zA-Z\_0-9]
- `\s` 与任意一个空白字符匹配

[`\n\f\r\t\v`]

- `\f` 换页字符;
- `\n` 换行字符;
- `\r` 回车字符;
- `\t` 制表符;
- `\v` 垂直制表符;
- `\S` 与除了空白符外任意一个字符匹配 [^`\n\f\r\t\v`]

- [] 只匹配其中的一个原子
- [^] 只匹配"除了"其中字符的任意一个原子
- [0-9] 匹配0-9任何一个数字
- [a-z] 匹配小写a-z任何一个字母
- [A-Z] 匹配大写A-Z任何一个字母



在正则表达式中有一些特殊字符带表特殊意义叫元字符。

- . 除换行符以外的任何一个字符
- | 或的意思，匹配其中一项就代表匹配
- 例子:匹配身份证号，旧版是15位数字，新版是
- 18位数字
- `/^\d{15}|\d{18}$/`

## ➤ 练习

检测用户输入内容是否包含"法轮功","枪支","毒品","中石油","共产党"等非法内容，如果包含显示内容非法，不包含显示通过。

匹配多个字符时用()分组，分组代表一个原子集合或者说一个大原子，并压入堆栈(内存)用于调用，组号是从左到右计数的调用时：如果是字面量形式用\1，构造函数方式用\\1这种方式我们叫做反向引用

例：

```
var reg = new RegExp("(hdw)123\\1","i");  
alert(reg.test("hdw123hdw"));
```

- 使用形如(?:pattern)的正则就可以避免保存括号内的匹配结果，反向引用也将会失效

可以使用一些元字符，重复表示一些元子或元字符

- \* 重复零次或更多次
- + 重复一次或更多次
- ? 重复零次或一次
- {n} 重复n次
- {n,} 重复n次或更多次
- {n,m} 重复n到m次

一片两片三四片，落尽正则全找见

上面的小标题翻译成正则就是{1},{2},{3,4},{1,}。

正则匹配是贪婪的，禁止贪婪用？

- $*?$  重复任意次，但尽可能少重复
- $+?$  重复1次或更多次，但尽可能少重复
- $??$  重复0次或1次，但尽可能少重复
- $\{n,m\}?$  重复n到m次，但尽可能少重复
- $\{n,\}?$  重复n次以上，但尽可能少重复



## 字符边界

- ^ 匹配字符串的开始
- \$ 匹配字符串的结束，忽略换行符
- 单词边界限制
- \b 匹配单词的边界
- \B 匹配除单词边界以外的部分

- i 不区分大小写字母的匹配
- m 将字符串视为多行，修饰^与\$
- g 全局匹配，找到所有匹配项



模式匹配的顺序（从高到低）

顺序	元字符	说明
1	( )	模式单元
2	? * +{ }	重复匹配
3	^ \$	边界限制
4		模式选择

# 字符串中用到正则的函数

➤ `str.search(regex)`

– `regex`为正则表达式，返回索引位置，不支持全局索引（即`g`修饰符无效）找到即停止搜索

例：

```
var str = "www.sxuek.com";  
alert(str.search(/uek/));
```

# 字符串中用到正则的函数

- `replace` ( 正则或字符串,替换内容 ) //支持全局g修饰符 , 如果模式不是全局 , 当匹配到一个以后将不会继续匹配 , 反之则会继续往下匹配。

1. 写一个去除字符串两边空格的函数
2. 写一个去除字符串当中所有空格的函数
3. 将 `background-color` 替换成 `backgroundColor`





# 字符串中用到正则的函数

## ➤ split 方法

- 拆分字符串，参数可以为字符串或正则表达式

➤ 想一想

将下面的字符以 , 、 分成数组

"安逸客教学理念 , 专注、极致、创新";

- 整数或者小数 `^-?\d+\.\?\d{0,3}$`
- 只能输入数字  `/^[0-9]*$/`
- 验证用户名和密码  `/^[a-zA-Z]\w{5,18}$/`
- 验证电话号码  `/(?:\(\d{3,4}\)|\d{3,4}-?)\d{8}/`
- 验证身份证号  `/^[\d{15} |\d{18}]$/`
- 验证Email地址  
 `/^[a-zA-Z0-9]\w+@[a-zA-Z]+\. (com|cc|org|net|cn|com.cn)$/`
- 验证URL :  
 `"/^http://([\w-]+\.\.)+[\w-]+(/[\w-./?%&=]*)?$/"`  
 `"/^(http[s]?:)?(V{2})?([a-z0-]+\.\.)?[a-z0-9]+(\.(com|cn|cc|org|net|com.cn))$/i`



谢谢观看...

SHANXI **UNIQUE**  
TECHNOLOGY CO.,LTD.