## 正则表达式

正则是什么：用于匹配字符集的表达死。

正则：匹配 修改 提取

正则的基本语法：

元字符

\d 表示数字

\ D 表示非数字

\w 表示字符 字母 数字 下划线

\W 表示非字符

\s 表示空白字符

\S 表示非空白字符

\* 表示除、n之外的任意字符

\b 表示单词边界匹配符

范围词：

1. Z] 表示取值范围在A到Z之间

[0-9] 表示取值范围在0到9之间

[a-z0-9A-Z]表示a到z,0到9，A到Z

[^A-Z] 表示取值范围不在A-Z之间 ^ 在中括号中表示非的意思

[iu4e00-u9fa5]表示匹配汉字

量词：

n\* 表示n出现0次或者多次

n? 表示n出现0次或者1次

n+ 表示n出现至少1次

n{3} 表示n恰好出现3次

n{3,} 表示n至少出现3次

n{3,6} 表示n出现3到6次

其他字符：

\ 转义字符

| 表示或者

（） 表示分组

如果我们在匹配字符串的时候想要完整的去匹配整个字符串，那么需要加上^和$ 符号，如果我们只想要字符串中做部分信息的匹配，则无需加^和$符号

表示开始和结束

^表示匹配的开始

$表示匹配的结束

正则使用的步骤：

创建正则对象

第一种 ： var reg = new RegExp ()

第二种 ： var reg = /正则表达式/ （推荐使用）

正则对象的方法

第一种： test（）方法 ：用来检测指定的字符串是否符合当前的正则返回 true 或者false

第二种： exec（）方法 ： 用来查找指定字符串中所有符合条件的内容返回一个数组

案例：

校验QQ号：/ ^[1-9]\d{4-10}$/

固定电话：/ ^0\d{2-3}-[1-9]\{7}&/

单词边界匹配符 /\b\w{3}\b/g

匹配的模式

i:忽略大小写

g：全局匹配

例如：var reg =/\d/gi

正则的提取

（）表示分组

RegExp.$1 表示提取

var reg =RegExp.$1 表示第一组

var reg =RegExp.$2 表示第二组

var reg =RegExp.$3 表示第三组

第一个出现的就是第一组其后面就是第二组第三组

字符串（string）对象中的正则使用

**字符串的方法有下面的若干方法是支持正则表达式**

Match（） 方法返回符合指定条件的数据集合 匹配

Replace（）方法替换字符串

Search（）查找字符串

Split（） 切割字符串

Var str =sa aas s55 45 48

1.方法使用 str.match (正则表达式) 匹配 返回一个数据集合

2.方法使用 str.replace(正则表达式) 替换

3.方法使用 str.search(正则表达式) 查找 返回数字下标

4.方法使用 str.split(正则表达式) 切割

**贪婪模式和懒惰模式 表示符号 ‘？’**

贪婪模式：在匹配的时候尽可能多的去匹配

懒惰模式：在匹配的时候尽可能少的去匹配

案例：

“aasdsdfdfsdf”.match(/\w+/) 贪婪模式

“aasdsdfdfsdf”.match(/\w+?/) 懒惰模式

案例：

密码强度验证

表单验证

表单form元素师块级元素

Return false 组织提交

Return turn 允许提交