**正则**

1. 转义字符：\”,会将“\ “后字符强制转换为正常文本,\n字符串换行符。
2. 字符串没有多行，如：

var innerHTML=”<div></div> //系统报错

<span></span>”

var innerHTML=”\

<div></div>\ \\将文本形式回车转义掉

<span></span>\

”

1. 正则表达式匹配模式：i(ignoranceCase)忽略大小写，g(global)全局匹配，m多行匹配（可以识别换行符\n）
2. 正则表达式上的两个方法：reg. test(str) str.match(reg)
3. []一位，里面内容为区间（范围），eg：/[0-9]/g;^在[]里面表示除了，在外面表示以什么为开头/^a/g以a为开头
4. 元字符：\w等价于[0-9A-z\_],\W等价于[^\w]；

\d等价于[0-9]，\D等价于[^\d]；

\s表示空白字符（各种缩进转义字符）

\S等价于[^\s]；

\b单词边界，\B非单词边界，单词前后的空格；’.’

等价于[^\r\n],查找单个字符

eg：var reg=/\tc/g;var str1=’ c’,str2=’\tcad’;

str1.match(reg);str2.match(reg)

//null 回车符+c

1. unicode编码：\u000000🡪16进制4位数，前两位为层级，后四位为编码，unicode编码包含一切字符

\u010000:\u010000🡪\u01ffff 第一层，涵盖中文（第一层可忽略01）

\u020000:\u020000🡪\u02ffff 第二层

·····

\u0f0000:\u0f0000🡪\u0fffff 第十五层

\u100000:\u100000🡪\u10ffff 十六层

1. 量词：贪婪匹配原则

‘+’出现1次以上🡪{1,infinity}

‘\*’0次到多次

‘？’0次或1次

‘｛x,y｝’x次到y次

1. 检验一一个字符串首尾是否包含数字：

var reg=/^\d|\d$/g

1. 正则表达式方法：
2. reg.exec(str)返回第一个，继续执行返回下次匹配成功字符串的第一个字符索引位置（全局匹配下），根据lastIdex游标位置进行记录，可以手动改变游标；
3. reg.test(str)返回true或false
4. str.search(reg)返回匹配位置，否则返回-1
5. str.match(reg)找到匹配
6. str.split(reg)
7. str.replace(reg)
8. String. replace(‘a’,’b’)只能匹配一个

配合正则reg=/(\w)\1(\w)\2/g,可以全局匹配了

eg：将XXYY形式的字符串倒转为YYXX

code1：var reg=/(\w)\1(\w)\2/g;

var str=’aabbccdd’;

console.log(str.replace(reg,’$2$2$1$1’));

//bbaaddcc

code2:str.replace(reg,function($,$1,$2){

return $2+$2+$1+$1;

}) //$全局(all matching case),$1第一个匹配选项…,$2第二个匹配选项

1. 反向引用：eg：找出形式为XXYY的字符串，reg=/(\w)\1/🡪\1反向引用第一个，所以reg=/(\w)\1(\w)\2/
2. 正向预查（正向断言）：/a(?=b)/g b只是修饰，选出a
3. 打破贪婪匹配，量词后加‘？’，如/a+?/g
4. 字符串去重：var str=’aaaaabbbbbbccc’;

var reg=/(\w)\1\*/g;console.log(reg,’$1’)

1. 百度面试题：

var str=’10000000000’; //变为100.000.000

后面往前面查

answer:

var reg=/ (?=(\B))(\d{3}+$)/g;

16、去重：

var arr=[1,2,4,2,1,3,4,1,3,5,6,8];

Array.prototype.uniqueStr=function(arr){

var newArr=[];

for(var i=0,len=arr.length;i<len;i++){

if(newArr.indexOf(arr[i])===-1){

newArr.push(arr[i]);

}}

return newArr;}

arr.uniqueStr(arr)