1. Event对象

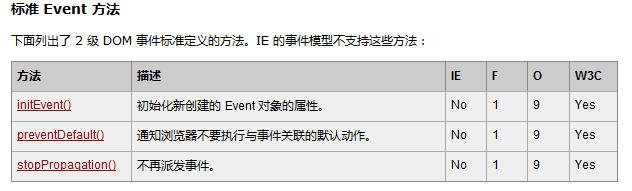
event对象：也就是“事件对象”: 代表一个网页上的事件发生时的相关信息。具体比如：鼠标的位置，按键的键值/键名，事件发生在谁身上（就是所谓的事件源）。

* 1. 事件对象的组成









注意：event对象在FF浏览器中，必须在事件调用函数的位置传如函数中

* 1. 事件的实例

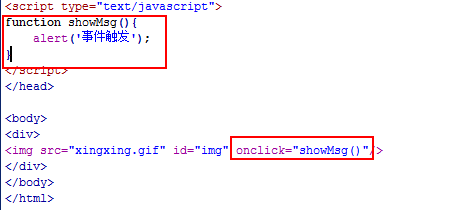
按钮改变点击数----练习事件源



练习鼠标事件: 瞄准---鼠标坐标(鼠标事件)



1. 事件注册与捕获
   1. 往元素上添加事件的三种方式
      1. 直接在元素标签上添加事件句柄



* + 1. 在元素标签的DOM对象上添加事件句柄属性



* + 1. 事件注册的方式来添加或注销

IE:

target.attachEvent(type, listener);

target： 文档节点、document、window 或 XMLHttpRequest。

type： 字符串，事件名称，含“on”，比如“onclick”、“onmouseover”、“onkeydown”等。

listener ：事件处理函数，由程序员提供；

IE注销事件

detachEvent(type,function)

W3C标准添加事件方式:

target.addEventListener(type, listener, useCapture);

参数介绍:  
 target： 文档节点、document、window 或 XMLHttpRequest。   
 type： 字符串，事件名称，不含“on”，比如“click”、“mouseover”、“keydown”等。   
 listener ：实现了 EventListener 接口或者是 JavaScript 中的函数。   
 useCapture：是否使用捕捉，一般用 false 。

例如：

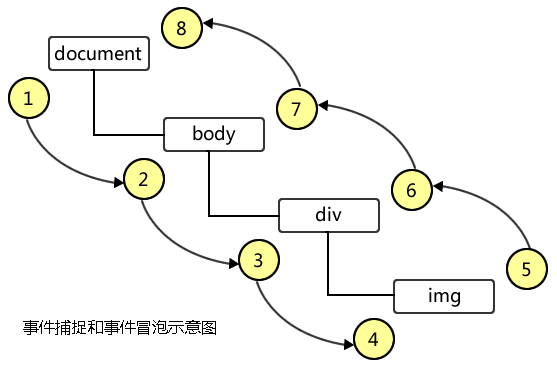
document.getElementById("testText").addEventListener("keydown", function (event) { alert(event.keyCode); }, false);

W3C标准删除事件

removeEventListener(event,function,capture/bubble);



* 1. 事件的捕捉和冒泡(了解)

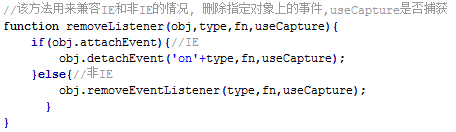


向下是捕获1-2-3-4,向上是冒泡5,6,7,8

# 2.3

事件的捕获和冒泡只能够通过注册事件的方式来完成





1. 正则表达式
   1. 什么是正则表达式

**正则表达式是描述字符模式的对象，以一个用简单方式描述或者匹配一系列符合某个句法规则的字符串。**（正则表达式:描述字符串规则的字符串，比如：人类描述：“字符串中有abc”,正则表达式描述：“/abc/”)；

**正则表达式通常被用来检索、替换那些符合某个模式的文本。**

正则表达式威力：

/\d{5,11}/ 字符串中0-9连续出现5至11次；

/\w+/字符串中必须是（a-z0-9的字符）；

* 1. 正则表达式能做什么

1. 检查字符串是否合法性
   * 1. 验证用户名 （a-z，0-9，不能全是数字，不能全是字母）
     2. 验证邮箱格式 (xxx@qq.com)
     3. 验证电话号码 (11位数字)
     4. 验证身份证 (18位 )
     5. 验证QQ号码格式（5-12纯数字，第一位不能为0）；
2. 提取字符串中信息
   * 1. 提取一条短信中数字；
     2. 提取文件名的后缀；
3. 替换字符串
   * 1. 替换字符串中的非法字符；
     2. 对电话号码进行屏蔽；（1852\*\*\*\*0102）
     3. 替换占位符 “hello {{name}} ” hello 王老二 （模板框架）
   1. 正则表达式使用

正则表达式是一种字符模式**对象（RegExp）**；

1. new构造方式

var r = new RegExp(“test”);

1. 字面量方式（推荐）

var r = /test/;

以上两种方式，完全等效，只是创建形式不同，匹配字符串中的test。

使用正则表达式中的test方法，对一个字符串进行匹配，如果模式与字符串匹配成功，返回true,否则返回false;

console.log(r.test(“你妹妹是test”));//true

console.log(r.test(“这是一个 test !!!”));//true

console.log(r.test(“这是一个 tEst !!!”));//false,不匹配，因为默认情况下正则会区分大小写

* 1. 正则表达式的修饰符

正则表达式模式区分大小写：

var r = /test/;

r.test(“这是一个 tEst !!!”); //false

使用修饰符规则匹配模式：

修饰符 描述

i 执行对大小写不敏感的匹配。

g 执行全局匹配（查找所有匹配而非在找到第一个匹配后停止,替换、提取字符串时候有用）。

m 执行多行匹配。

使用方式：

var r = /test/igm; //顺序igm顺序无所谓；

console.log(r.test(“我是大写的Test”));//true

* 1. 精确匹配

精确匹配：正则表达式中，出现了什么字符，那么匹配的字符串也必须出现什么字符，表示该字符必须在表达式中出现。

例如，在/**test**/中，有4个字符，它们表示这些字符必须在一个字符串中出现，才能匹配该模式。

一个接着一个的字符，隐式表达了“后面紧跟着”这样一个操作。所以，/test/的意思是说，“t”后面跟着“e”，“e”后面跟着“s’，“s’后面又跟着“t”。

var r = /test/igm;

console.log(r.test(“aaaaa t1est”));//false

console.log(r.test(“aaaaa test”));//true

* 1. 直接量字符串

正则表达式中的所有字母和数字都是按照字面含义进行匹配的。

/**a**/ 匹配字符串中的**a**， /**6**/匹配字符串中的**6**

JavaScript正则表达式语法也支持非字母的字符(非可见字符)匹配，这些字符需要通过反斜线(\)作为前缀进行转义。比如，转义字符\n用以匹配换行符。

|  |  |
| --- | --- |
| \0 | 查找 NUL 字符。 |
| **\n** | **查找换行符。** |
| \f | 查找换页符。 |
| \r | 查找回车符。 |
| \t | 查找制表符。 |
| \v | 查找垂直制表符。 |
| [\xxx](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-octal.html) | 查找以八进制数 xxx 规定的字符。 |
| [\xdd](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-hex.html) | 查找以十六进制数 dd 规定的字符。 |
| [\uxxxx](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-unicode-hex.html) | 查找以十六进制数 xxxx 规定的 Unicode 字符。 |

var s = “二狗：\n 妹纸加微信”;

var r = /\n/;

r.test(s);// true

* 1. 指定位置（限定位置）

|  |  |
| --- | --- |
| n$ | 匹配任何结尾为 n 的字符串。 |
| ^n | 匹配任何开头为 n 的字符串。 |
| ([?=n](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-nfollow.html)) | 匹配任何其后紧接指定字符串 n 的字符串。 |
| ([?!n](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-nfollow-not.html)) | 匹配任何其后没有紧接指定字符串 n 的字符串。 |

* 1. 特殊字符

正则表达式中：

^ $ ． \* + ? = ! ： | ＼ ／ ( ) [ ] { } 具有特殊含义，如果我们的模式中需要匹配这里的某个字符那么使用“\特殊字符”转义， 比如\**.**，这样就让**.**失去了特殊含义，作为普通字符匹配。

var r = /\.jpg/;

console.log(r.test(“xxxx.jpg”));//true

* 1. 匹配一类字符

很多时候，我们并不想匹配一个特定的字符，而是想匹配一个有限字符集中的某一个字符。我们可以通过将字符集放到中括号内，来指定该字符集操作符（也称为字符类个( character class)操作符）：[abc].

上述示例，是说我们要匹配“a”“b”和“c”中的任何一个字符。注意，即使这个表达式一共有五个字符，但它只能匹配候选字符串3个中的一个字符，[]只是一个范围。

有的时候，我们想要匹配一组有限字符集以外的字符。可以通过在中括号第一个开括号的后面加一个插入符（^）来实现。比如：

[^abc]

其意义将改变为：除了“a”、”b”、“c”以外的任意字符。

另外如果我想要匹配数字中的任意一个字符，我们可以写[0123456789],但这样写非常的麻烦，我们可以修改为[0-9];

当然对于字母而言也可以[a-zA-Z] [0-9a-zA-Z] ....

console.log(/[abc]/.test(“a123”));//true

console.log(/[abc]/.test(“a12b3”));//true

console.log(/[abc]/.test(“123”));//false

|  |  |
| --- | --- |
| [abc] | 查找方括号之间的任何字符。 |
| [^abc] | 查找任何不在方括号之间的字符。 |
| [0-9] | 查找任何从 0 至 9 的数字。 |
| [a-z] | 查找任何从小写 a 到小写 z 的字符。 |
| [A-Z] | 查找任何从大写 A 到大写 Z 的字符。 |
| [A-z] | 查找任何从大写 A 到小写 z 的字符。 |
| [adgk] | 查找给定集合内的任何字符。 |
| [^adgk] | 查找给定集合外的任何字符。 |
| (red|blue|green) | 查找任何指定的选项。 |

**元字符类**

|  |  |
| --- | --- |
| [.](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-dot.html) | 查找单个字符，除了换行和行结束符。 |
| [\w](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-wordchar.html) | 查找单词字符。 |
| [\W](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-wordchar-non.html) | 查找非单词字符。 |
| [\d](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-digit.html) | 查找数字。 |
| [\D](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-digit-non.html) | 查找非数字字符。 |
| [\s](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-whitespace.html) | 查找空白字符。 |
| [\S](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-whitespace-non.html) | 查找非空白字符。 |
| [\b](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-begin.html) | 匹配单词边界。 |
| [\B](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-begin-not.html) | 匹配非单词边界。 |

* 1. 限定字符（重复）

正则表达式的语法，可以把两位数描述成/\d\d/，四位数描述成成/ d\d\d\d/。使用这种方式非常的麻烦，如果要描述100位的数字/\d\d\d\d\d\d..../，简直要人命；

正则表达式提供了“量词”，指定字符出现的次数；

|  |  |
| --- | --- |
| [n+](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-onemore.html) | 匹配任何包含至少一个 n 的字符串。 |
| [n\*](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-zeromore.html) | 匹配任何包含零个或多个 n 的字符串。 |
| [n?](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-zeroone.html) | 匹配任何包含零个或一个 n 的字符串。 |
| [n{X}](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-nx.html) | 匹配包含 X 个 n 的序列的字符串。 |
| [n{X,Y}](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-nxy.html) | 匹配包含 X 或 Y 个 n 的序列的字符串。 |
| [n{X,}](mk:@MSITStore:E:\\Web前端与移动开发\\课件\\JavaScript-s1\\w3capi\\w3c菜鸟教程.chm::/jsref-regexp-nxcomma.html) | 匹配包含至少 X 个 n 的序列的字符串。 |

例如：

/\d{2，4}/ //匹配2-4个数字

/\w{3}\d?/ //匹配3个单词和一个可选数字

/\s+js\s+/ //匹配前后各带一个或多个空格的字符串”js”

默认重复匹配是贪婪匹配：匹配重复字符是尽可能多地匹配，而且允许后续的正则表达式继续匹配；

修改为**非贪婪匹配**只须量词后面加一个?号即可；

例子：var s = “gggg<cccc@itsource.cn>和<bbb@itsource.cn>hhhhh”;

var r = /<.?>/; //尽量多的匹配

r.exec(s);

r = /<.??>/; //尽量少的匹配

r.exec(s);

* 1. 选择

字符“|”用于分隔供选择的字符。例如，/ab|cd|ef/可以匹配字符串“ab”，也可以匹配字“cd”,或者是ef；。 “|” 可以理解为或者的意思；

比如：var re = /jpg|png|gif/; //匹配字符串中jpg、png、gif; 字符串中只需要匹配其中一个就ok了；

re.test(“xxx.jpg”);//true

* 1. 分组

正则表达式中的圆括号有多种作用，把单独的项组合成子表达式，以便可以

处理一个；像处理一个独立的单元那样用“|”、 “\*”、 “+”或者“？”等来对单元内的项进行处理。

例如：/java(script)?/可以匹配字符串“java”，其后可以有“script”也可以没有。

正则表达式中，圆括号的另一个作用是在完整的模式中定义子模式。当一个正则表达式成功地和目标字符串相匹配时，可以从目标串中抽出和圆括号中的子模式相匹配的部分。

例如，假定我们正在检索的模式是一个或多个小写字母后面跟随了一位或多位数字，则可以使用模式/[a-z]+\d+/。但假定我们真正关心的是每个匹配尾部的数字，那么如果将模式的数字部分放在括号中/[a-z]+(\d+)/，就可以从检索到的匹配中抽取数字了。

var s = “AAA8888”;//关心字符中字母后面的数字；

var re =/[a-z]+(\d+)/i;

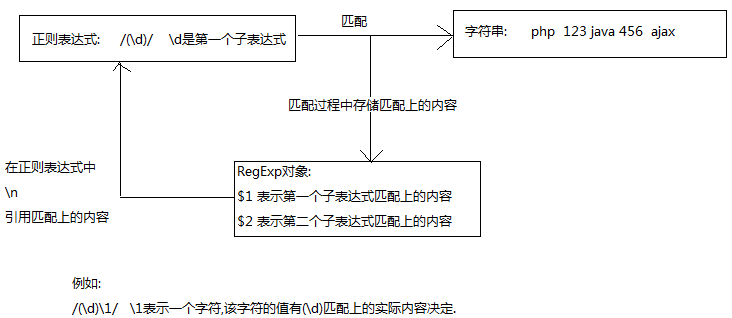
var arr = re.exec(s);

arr[1] //88888 ; 数组中的第二个元素就是正则中（**\d+**）的匹配；

* 1. 反向引用

圆括号的表达式的另一个用途是允许在同一正则表达式的后部引用前面的子表达式。是通过在字符“\”后加一位或多位数字来实现的。

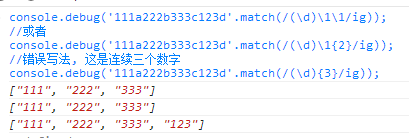
捕获之后，可以在匹配模式中，使用\n (n：数字)来引用系统的第n号缓冲区内容



**在哪里用到以上概念**

通常是用来处理相同连续的内容。

1）匹配连续相同的三个数字

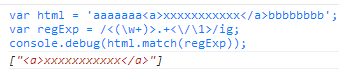


2）匹配ABAB格式的数字.例如:1212 3434  


3）匹配ABBA格式的数字.例如:1221 3443



4）在一个字符串，查找一对html标记以及中间的内容



最后一点，如果你没有不需要对分组进行引用，而只是简单的分组，那么可以使用（?:xxx）,这样就不会创建引用，同时，你也不能使用\n了哟。

1. 正则相关的方法
   1. RegExp 对象方法

|  |  |  |  |
| --- | --- | --- | --- |
| **方法** | **描述** | **FF** | **IE** |
| compile | 编译正则表达式。 | 1 | 4 |
| exec | 检索字符串中指定的值。返回找到的值，并确定其位置。 | 1 | 4 |
| test | 检索字符串中指定的值。返回 true 或 false。 | 1 | 4 |

* 1. 支持正则表达式的 String 对象的方法

|  |  |  |  |
| --- | --- | --- | --- |
| **方法** | **描述** | **FF** | **IE** |
| search | 检索与正则表达式相匹配的值。 | 1 | 4 |
| match | 找到一个或多个正则表达式的匹配。 | 1 | 4 |
| replace | 替换与正则表达式匹配的子串。 | 1 | 4 |
| split | 把字符串分割为字符串数组。 | 1 | 4 |

1. 常用正则表达式
   1. 校验数字的表达式

* 数字：^[0-9]\*$
* n位的数字：^\d{n}$
* 至少n位的数字：^\d{n,}$
* m-n位的数字：^\d{m,n}$
* 零和非零开头的数字：^(0|[1-9][0-9]\*)$
* 非零开头的最多带两位小数的数字：^([1-9][0-9]\*)+(.[0-9]{1,2})?$
* 带1-2位小数的正数或负数：^(\-)?\d+(\.\d{1,2})?$
* 正数、负数、和小数：^(\-|\+)?\d+(\.\d+)?$
* 有两位小数的正实数：^[0-9]+(.[0-9]{2})?$
* 有1~3位小数的正实数：^[0-9]+(.[0-9]{1,3})?$
* 非零的正整数：^[1-9]\d\*$ 或 ^([1-9][0-9]\*){1,3}$ 或 ^\+?[1-9][0-9]\*$
* 非零的负整数：^\-[1-9][]0-9″\*$ 或 ^-[1-9]\d\*$
* 非负整数：^\d+$ 或 ^[1-9]\d\*|0$
* 非正整数：^-[1-9]\d\*|0$ 或 ^((-\d+)|(0+))$
* 非负浮点数：^\d+(\.\d+)?$ 或 ^[1-9]\d\*\.\d\*|0\.\d\*[1-9]\d\*|0?\.0+|0$
* 非正浮点数：^((-\d+(\.\d+)?)|(0+(\.0+)?))$ 或 ^(-([1-9]\d\*\.\d\*|0\.\d\*[1-9]\d\*))|0?\.0+|0$
* 正浮点数：^[1-9]\d\*\.\d\*|0\.\d\*[1-9]\d\*$ 或 ^(([0-9]+\.[0-9]\*[1-9][0-9]\*)|([0-9]\*[1-9][0-9]\*\.[0-9]+)|([0-9]\*[1-9][0-9]\*))$
* 负浮点数：^-([1-9]\d\*\.\d\*|0\.\d\*[1-9]\d\*)$ 或 ^(-(([0-9]+\.[0-9]\*[1-9][0-9]\*)|([0-9]\*[1-9][0-9]\*\.[0-9]+)|([0-9]\*[1-9][0-9]\*)))$
* 浮点数：^(-?\d+)(\.\d+)?$ 或 ^-?([1-9]\d\*\.\d\*|0\.\d\*[1-9]\d\*|0?\.0+|0)$
  1. 校验字符的表达式
* 汉字：^[\u4e00-\u9fa5]{0,}$
* 英文和数字：^[A-Za-z0-9]+$ 或 ^[A-Za-z0-9]{4,40}$
* 长度为3-20的所有字符：^.{3,20}$
* 由26个英文字母组成的字符串：^[A-Za-z]+$
* 由26个大写英文字母组成的字符串：^[A-Z]+$
* 由26个小写英文字母组成的字符串：^[a-z]+$
* 由数字和26个英文字母组成的字符串：^[A-Za-z0-9]+$
* 由数字、26个英文字母或者下划线组成的字符串：^\w+$ 或 ^\w{3,20}$
* 中文、英文、数字包括下划线：^[\u4E00-\u9FA5A-Za-z0-9\_]+$
* 中文、英文、数字但不包括下划线等符号：^[\u4E00-\u9FA5A-Za-z0-9]+$ 或 ^[\u4E00-\u9FA5A-Za-z0-9]{2,20}$
* 可以输入含有^%&’,;=?$\”等字符：[^%&’,;=?$\x22]+
* 禁止输入含有~的字符：[^~\x22]+
  1. 特殊需求表达式
* Email地址：^\w+([-+.]\w+)[\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*$](mailto:*@\\w+([-.]\\w+)*\\.\\w+([-.]\\w+)*$)

^[a-zA-Z0-9.\_%+-]+@(?!.\*\.\..\*)[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$

\w[-\w.+]\*@([A-Za-z0-9][-A-Za-z0-9]+\.)+[A-Za-z]{2,14}

^\w+([-+.]\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*$

^\w+([-+.]\w+)\*@\w+([-.]\w+)\*\.[a-zA-Z]{2,4}$

* 域名：[a-zA-Z0-9][-a-zA-Z0-9]{0,62}(/.[a-zA-Z0-9][-a-zA-Z0-9]{0,62})+/.?
* InternetURL：

^http://([\w-]+\.)+[\w-]+(/[\w-./?%&=]\*)?$

或 ^((https|http|ftp|rtsp|mms)?:\/\/)[^\s]+

或 **[a-zA-Z0-9][-a-zA-Z0-9]{0,62}(\.[a-zA-Z0-9][-a-zA-Z0-9]{0,62})+\.?**

* 手机号码：^(13[0-9]|14[5|7]|15[0|1|2|3|5|6|7|8|9]|18[0|1|2|3|5|6|7|8|9])\d{8}$
* 电话号码(“XXX-XXXXXXX”、”XXXX-XXXXXXXX”、”XXX-XXXXXXX”、”XXX-XXXXXXXX”、”XXXXXXX”和”XXXXXXXX)：^($$\d{3,4}-)|\d{3.4}-)?\d{7,8}$
* 国内电话号码(0511-4405222、021-87888822)：\d{3}-\d{8}|\d{4}-\d{7}
* 身份证号(15位、18位数字)：^\d{15}|\d{18}$
* 短身份证号码(数字、字母x结尾)：^([0-9]){7,18}(x|X)?$ 或 ^\d{8,18}|[0-9x]{8,18}|[0-9X]{8,18}?$
* 帐号是否合法(字母开头，允许5-16字节，允许字母数字下划线)：^[a-zA-Z][a-zA-Z0-9\_]{4,15}$
* 密码(以字母开头，长度在6~18之间，只能包含字母、数字和下划线)：^[a-zA-Z]\w{5,17}$
* 强密码(必须包含大小写字母和数字的组合，不能使用特殊字符，长度在8-10之间)：^(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z]).{8,10}$
* 日期格式：^\d{4}-\d{1,2}-\d{1,2}
* 一年的12个月(01～09和1～12)：^(0?[1-9]|1[0-2])$
* 一个月的31天(01～09和1～31)：^((0?[1-9])|((1|2)[0-9])|30|31)$
* 钱的输入格式：
  + 有四种钱的表示形式我们可以接受:”10000.00″ 和 “10,000.00”, 和没有 “分” 的 “10000” 和 “10,000”：^[1-9][0-9]\*$
  + 这表示任意一个不以0开头的数字，但是，这也意味着一个字符”0″不通过，所以我们采用下面的形式：^(0|[1-9][0-9]\*)$
  + 一个0或者一个不以0开头的数字.我们还可以允许开头有一个负号：^(0|-?[1-9][0-9]\*)$
  + 这表示一个0或者一个可能为负的开头不为0的数字.让用户以0开头好了.把负号的也去掉，因为钱总不能是负的吧.下面我们要加的是说明可能的小数部分：^[0-9]+(.[0-9]+)?$
  + 必须说明的是，小数点后面至少应该有1位数，所以”10.”是不通过的，但是 “10” 和 “10.2” 是通过的：^[0-9]+(.[0-9]{2})?$
  + 这样我们规定小数点后面必须有两位，如果你认为太苛刻了，可以这样：^[0-9]+(.[0-9]{1,2})?$
  + 这样就允许用户只写一位小数。下面我们该考虑数字中的逗号了，我们可以这样：^[0-9]{1,3}(,[0-9]{3})\*(.[0-9]{1,2})?$
  + 1到3个数字，后面跟着任意个 逗号+3个数字，逗号成为可选，而不是必须：^([0-9]+|[0-9]{1,3}(,[0-9]{3})\*)(.[0-9]{1,2})?$
  + 备注：这就是最终结果了，别忘了”+”可以用”\*”替代。如果你觉得空字符串也可以接受的话(奇怪，为什么?)最后，别忘了在用函数时去掉去掉那个反斜杠，一般的错误都在这里
* xml文件：^([a-zA-Z]+-?)+[a-zA-Z0-9]+\\.[x|X][m|M][l|L]$
* 中文字符的正则表达式：[\u4e00-\u9fa5]
* 双字节字符：[^\x00-\xff] (包括汉字在内，可以用来计算字符串的长度(一个双字节字符长度计2，ASCII字符计1))
* 空白行的正则表达式：\n\s\*\r (可以用来删除空白行)
* HTML标记的正则表达式：<(\S\*?)[^>]\*>.\*?</\1>|<.\*? /> (网上流传的版本太糟糕，上面这个也仅仅能部分，对于复杂的嵌套标记依旧无能为力)
* 首尾空白字符的正则表达式：^\s\*|\s\*$或(^\s\*)|(\s\*$) (可以用来删除行首行尾的空白字符(包括空格、制表符、换页符等等)，非常有用的表达式)
* 腾讯QQ号：[1-9][0-9]{4,} (腾讯QQ号从10000开始) [1-9]([0-9]{5,11})
* 中国邮政编码：[1-9]\d{5}(?!\d) (中国邮政编码为6位数字)
* IP地址：\d+\.\d+\.\d+\.\d+ (提取IP地址时有用)
* IP地址：((?:(?:25[0-5]|2[0-4]\\d|[01]?\\d?\\d)\\.){3}(?:25[0-5]|2[0-4]\\d|[01]?\\d?\\d))或者((25[0-5])|(2[0-4]\d)|(1\d\d)|([1-9]\d)|\d)(\.((25[0-5])|(2[0-4]\d)|(1\d\d)|([1-9]\d)|\d)){3}
  1. 网上收集常用的正则表达式大全

http://blog.csdn.net/xiaanming/article/details/9137063

//判断输入内容是否为空      
function IsNull(){      
    var str = document.getElementById('str').value.trim();      
    if(str.length==0){      
        alert('对不起，文本框不能为空或者为空格!');//请将“文本框”改成你需要验证的属性名称!      
    }      
}      
     
//判断日期类型是否为YYYY-MM-DD格式的类型      
function IsDate(){       
    var str = document.getElementById('str').value.trim();      
    if(str.length!=0){      
        var reg = /^(\d{1,4})(-|\/)(\d{1,2})\2(\d{1,2})$/;       
        var r = str.match(reg);       
        if(r==null)      
            alert('对不起，您输入的日期格式不正确!'); //请将“日期”改成你需要验证的属性名称!      
        }      
}       
     
//判断日期类型是否为YYYY-MM-DD hh:mm:ss格式的类型      
function IsDateTime(){       
    var str = document.getElementById('str').value.trim();      
    if(str.length!=0){      
        var reg = /^(\d{1,4})(-|\/)(\d{1,2})\2(\d{1,2}) (\d{1,2}):(\d{1,2}):(\d{1,2})$/;       
        var r = str.match(reg);       
        if(r==null)      
        alert('对不起，您输入的日期格式不正确!'); //请将“日期”改成你需要验证的属性名称!      
    }      
}       
     
//判断日期类型是否为hh:mm:ss格式的类型      
function IsTime()       
{       
    var str = document.getElementById('str').value.trim();      
    if(str.length!=0){      
    reg=/^((20|21|22|23|[0-1]\d)\:[0-5][0-9])(\:[0-5][0-9])?$/       
        if(!reg.test(str)){      
            alert("对不起，您输入的日期格式不正确!");//请将“日期”改成你需要验证的属性名称!      
        }      
    }      
}       
     
//判断输入的字符是否为英文字母      
function IsLetter()       
{       
        var str = document.getElementById('str').value.trim();      
        if(str.length!=0){      
        reg=/^[a-zA-Z]+$/;       
        if(!reg.test(str)){      
            alert("对不起，您输入的英文字母类型格式不正确!");//请将“英文字母类型”改成你需要验证的属性名称!      
        }      
        }      
}       
     
//判断输入的字符是否为整数      
function IsInteger()       
{         
        var str = document.getElementById('str').value.trim();      
        if(str.length!=0){      
        reg=/^[-+]?\d\*$/;       
        if(!reg.test(str)){      
            alert("对不起，您输入的整数类型格式不正确!");//请将“整数类型”要换成你要验证的那个属性名称！      
        }      
        }      
}       
     
//判断输入的字符是否为双精度      
function IsDouble(val)       
{       
        var str = document.getElementById('str').value.trim();      
        if(str.length!=0){      
        reg=/^[-\+]?\d+(\.\d+)?$/;      
        if(!reg.test(str)){      
            alert("对不起，您输入的双精度类型格式不正确!");//请将“双精度类型”要换成你要验证的那个属性名称！      
        }      
        }      
}       
     
     
//判断输入的字符是否为:a-z,A-Z,0-9      
function IsString()       
{       
        var str = document.getElementById('str').value.trim();      
        if(str.length!=0){      
        reg=/^[a-zA-Z0-9\_]+$/;       
        if(!reg.test(str)){      
            alert("对不起，您输入的字符串类型格式不正确!");//请将“字符串类型”要换成你要验证的那个属性名称！      
        }      
        }      
}       
     
//判断输入的字符是否为中文      
function IsChinese()       
{       
        var str = document.getElementById('str').value.trim();      
        if(str.length!=0){      
        reg=/^[\u0391-\uFFE5]+$/;      
        if(!reg.test(str)){      
            alert("对不起，您输入的字符串类型格式不正确!");//请将“字符串类型”要换成你要验证的那个属性名称！      
        }      
        }      
}       
     
//判断输入的EMAIL格式是否正确      
function IsEmail()       
{       
        var str = document.getElementById('str').value.trim();      
        if(str.length!=0){      
        reg=/^\w+([-+.]\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*$/;      
        if(!reg.test(str)){      
            alert("对不起，您输入的字符串类型格式不正确!");//请将“字符串类型”要换成你要验证的那个属性名称！      
        }      
        }      
}       
     
//判断输入的邮编(只能为六位)是否正确      
function IsZIP()       
{       
        var str = document.getElementById('str').value.trim();      
        if(str.length!=0){      
        reg=/^\d{6}$/;      
        if(!reg.test(str)){      
            alert("对不起，您输入的字符串类型格式不正确!");//请将“字符串类型”要换成你要验证的那个属性名称！      
        }      
        }      
}       
     
//判断输入的数字不大于某个特定的数字      
function MaxValue()       
{       
    var val = document.getElementById('str').value.trim();      
        if(str.length!=0){      
        reg=/^[-+]?\d\*$/;       
        if(!reg.test(str)){//判断是否为数字类型      
            if(val>parseInt('123')) //“123”为自己设定的最大值      
            {       
                alert('对不起，您输入的数字超出范围');//请将“数字”改成你要验证的那个属性名称！      
            }       
        }      
    }      
}       
     
     
 Phone : /^((\(\d{2,3}\))|(\d{3}\-))?(\(0\d{2,3}\)|0\d{2,3}-)?[1-9]\d{6,7}(\-\d{1,4})?$/      
 Mobile : /^((\(\d{2,3}\))|(\d{3}\-))?13\d{9}$/      
 Url : /^http:\/\/[A-Za-z0-9]+\.[A-Za-z0-9]+[\/=\?%\-&\_~`@[\]\':+!]\*([^<>\"\"])\*$/     
 IdCard : /^\d{15}(\d{2}[A-Za-z0-9])?$/     
 QQ : /^[1-9]\d{4,8}$/     
 某种特殊金额：/^((\d{1,3}(,\d{3})\*)|(\d+))(\.\d{2})?$/               //说明：除“XXX    XX,XXX    XX,XXX.00”格式外

//为上面提供各个JS验证方法提供.trim()属性     
String.prototype.trim=function(){     
        return this.replace(/(^\s\*)|(\s\*$)/g, "");      
    }

调用：  
<input type="text" name="str" >  
<input type="button" value=" 确定 " onClick="">    //onClick中写自己要调用的JS验证函数

<script language="javascript" type="text/javascript">  
var patterms = new Object();  
//验证IP  
patterms.ip = /^(\d{1,2}|1\d\d|2[0-4]\d|25[0-5])(\.(\d{1,2}|1\d\d|2[0-4]\d|25[0-5])){3}$/;  
//验证EMAIL  
patterms.email = /^[a-zA-Z0-9\_-]+@[a-zA-Z0-9\_-]+(\.[a-zA-Z0-9\_-]+)+$/;  
//验证日期格式2009-07-13  
patterms.date = /^\d{4}-(0?[1-9]|1[0-2])-(0?[1-9]|[1-2]\d|3[0-1])$/;  
//验证时间格式16:55:39  
patterms.time = new RegExp("^([0-1]\\d|2[0-3]):[0-5]\\d:[0-5]\\d$");  
//验证函数  
function verify(str,pat)  
{  
    var thePat;  
    thePat = patterms[pat];  
    if(thePat.test(str))  
    {  
        return true;  
    }  
    else  
    {  
        return false;  
    }  
}  
//测试  
alert(verify("asidycom@163.com","email")+","+verify("192.168.1.1","ip")+  
    ","+verify("16:55:39","time")+","+verify("2009-07-13","date")+","+verify("192.168","ip"));

验证数字：^[0-9]\*$

验证n位的数字：^\d{n}$

验证至少n位数字：^\d{n,}$

验证m-n位的数字：^\d{m,n}$   
  
验证零和非零开头的数字：^(0|[1-9][0-9]\*)$   
  
验证有两位小数的正实数：^[0-9]+(.[0-9]{2})?$   
  
验证有1-3位小数的正实数：^[0-9]+(.[0-9]{1,3})?$   
  
验证非零的正整数：^\+?[1-9][0-9]\*$   
  
验证非零的负整数：^\-[1-9][0-9]\*$   
  
验证非负整数（正整数 + 0） ^\d+$   
  
验证非正整数（负整数 + 0） ^((-\d+)|(0+))$   
  
验证长度为3的字符：^.{3}$   
  
验证由26个英文字母组成的字符串：^[A-Za-z]+$   
  
验证由26个大写英文字母组成的字符串：^[A-Z]+$   
  
验证由26个小写英文字母组成的字符串：^[a-z]+$   
  
验证由数字和26个英文字母组成的字符串：^[A-Za-z0-9]+$   
  
验证由数字、26个英文字母或者下划线组成的字符串：^\w+$   
  
验证用户名或昵称经常用到: ^[\u4e00-\u9fa5A-Za-z0-9-\_]\*$  只能中英文，数字，下划线，减号  
  
验证用户密码:^[a-zA-Z]\w{5,17}$ 正确格式为：以字母开头，长度在6-18之间，只能包含字符、数字和下划线。   
  
验证是否含有 ^%&',;=?$\" 等字符：[^%&',;=?$\x22]+   
  
验证汉字：^[\u4e00-\u9fa5],{0,}$   
  
验证Email地址：^\w+[-+.]\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*$   
  
验证InternetURL：^http://([\w-]+\.)+[\w-]+(/[\w-./?%&=]\*)?$ ；^[a-zA-z]+://(w+(-w+)\*)(.(w+(-w+)\*))\*(?S\*)?$   
  
验证电话号码：^(\(\d{3,4}\)|\d{3,4}-)?\d{7,8}$：--正确格式为：XXXX-XXXXXXX，XXXX-XXXXXXXX，XXX-XXXXXXX，XXX-XXXXXXXX，XXXXXXX，XXXXXXXX。   
  
验证身份证号（15位或18位数字）：^\d{15}|\d{}18$   
  
验证一年的12个月：^(0?[1-9]|1[0-2])$ 正确格式为：“01”-“09”和“1”“12”   
  
验证一个月的31天：^((0?[1-9])|((1|2)[0-9])|30|31)$ 正确格式为：01、09和1、31。   
  
整数：^-?\d+$   
  
非负浮点数（正浮点数 + 0）：^\d+(\.\d+)?$   
  
正浮点数 ^(([0-9]+\.[0-9]\*[1-9][0-9]\*)|([0-9]\*[1-9][0-9]\*\.[0-9]+)|([0-9]\*[1-9][0-9]\*))$   
  
非正浮点数（负浮点数 + 0） ^((-\d+(\.\d+)?)|(0+(\.0+)?))$   
  
负浮点数 ^(-(([0-9]+\.[0-9]\*[1-9][0-9]\*)|([0-9]\*[1-9][0-9]\*\.[0-9]+)|([0-9]\*[1-9][0-9]\*)))$   
  
浮点数 ^(-?\d+)(\.\d+)?$

由于手机号段的不断更新，以前的正则表达式已经无法满足需求。重新编写这条表达式，号段资料来源依据：[http://www.von-line.com/hao.htm](http://www.von-line.com/hao.htm" \t "_blank)

[?](http://www.cnblogs.com/rubylouvre/archive/2012/09/30/2709302.html)

|  |  |
| --- | --- |
| 1  2  3 | var regex = {      mobile: /^0?(13[0-9]|15[012356789]|18[0236789]|14[57])[0-9]{8}$/  } |

**表达式分析：**  
“/”代表一个正则表达式。  
“^”代表字符串的开始位置，“$”代表字符串的结束位置。  
“?”代表匹配前面的字符一个或零个，所以这里0?的意思是手机号码可以以0开头或不以0开头。  
接下的部分验证11位的手机号码，先从13开始，因为从130-139都有所以可选区间是[0-9]，15开头的号码没有154所以[]里面没有4这个数字，当然也可以写成[0-35-9]，下面18和14开的号码同上。  
小括号括起来的代表一个子表达式，里面是4个可选分支分别用“|”来区分开来，在正则中“|”的优先级是最低的，这里每个分支匹配的都是3个字符（一个[]只能匹配一个字符，里面是可选的意思），也就是手机号码的前3位数字，那么后面还有8位数字需要匹配，可以是0-9的任意字符，所以是“[0-9]{8}”，{}中的数字代表匹配前面字符的个数。分析完毕。

。

* 1. 有多个按钮，制作点击某个按钮，就让这个按钮隐藏掉；
  2. 自定义一个浏览器右键菜单，单击右键是显示它
  3. 使用正则表达式验证百度注册表单。

