#### jQuery格式化 $.formatString

举例（用于表示函数的实参）

var a3 = $.formatString('<li class="fa fa-trash" onclick="deletePwd(\'{0}\',\'{1}\');" title="删除密码"></li>',id,devicePwdList[i].id);

a3变成<li class="fa fa-trash" onclick="deletePwd('1','70');" title="删除密码"></li>，其中\'{0}\'和\'{1}\'分别代表了id和devicePwdList[i].id这两个值

#### jQuery eval函数

eval() 函数可计算某个字符串，并执行其中的的 JavaScript 代码，举例：

eval("x=10;y=20;document.write(x\*y)")，输出200

var data={“id”:”89”,”name”:”mary”} data是字符创

var obj=eval(“+data+”) obj是json对象

#### jQuery UI 对话框

$("#dialog\_pwd\_confirm").dialog({

autoOpen: false, //只初始化，不显示

resizable: false, //不能调节对话框的大小

height:"auto", //高度自动

width:"auto", //宽度自动

title:"密码二次确认", //对话框的标题

modal: true, //创建模式对话框,挡住底层元素

buttons: {

"关闭": function() { //响应“关闭”按钮点击事件，关闭对话框

$( this ).dialog( "close" );

},

"确认": function() { //响应“确认”按钮点击事件，执行某操作

}

})

#### jQuery遍历和选择

var id=$(“li.item-2”).parents("thead").next().attr("data\_id");

parents可遍历当前元素的祖先元素，parent可遍历到当前元素的父元素

next可获取到紧邻当前元素的下一个同辈元素，prev可获取到紧邻当前元素的上一个同辈元素，nextAll可获取到紧邻当前元素之后的所有同辈元素，prevAll可获取到紧邻当前元素之前的所有同辈元素

attr可以获取/修改某个属性值

$(env).parents("thead").next().children().not(":first").text('');

$('input[name="deviceId"]','#pwd\_form').val(id);

text() - 设置或返回所选元素的文本内容

html() - 设置或返回所选元素的内容（包括 HTML 标记）

val() - 设置或返回表单字段的值，如input

$(":input")选择所有input元素，button也会被选择，而$("input")仅选择所有input元素

$('input[name="deviceId"]','#pwd\_form')选择id为pwd\_form容器内name为deviceId的input元素

$("input[type='checkbox']").attr("checked",false); //使checkbox不被选中

var check=$("input[type='checkbox']").is(":checked"); //获取checkbox的选中状态

var tr = $(this).closest('tr');//获得匹配选择器的第一个祖先元素，从当前元素开始沿 DOM 树向上搜索

$("body").find(".copy-tips").remove().end().append($copysuc);

find() 方法获得当前元素集合中每个元素的后代，通过选择器、jQuery 对象或元素来筛选。

remove() 方法移除被选元素，包括所有文本和子节点。empty() 方法移除该元素所有文本和子节点，本身不会从dom中除去。

append() 方法在被选元素的结尾（仍然在内部）插入指定内容。

end() 方法结束当前链条中的最近的筛选操作，并将匹配元素集还原为之前的状态。主要是在利用 jQuery 的链条属性（命令链）时，jQuery 会比较有用。如果不使用命令链，我们一般是通过变量名来调用之前的对象，这样我们就不需要操作堆栈了。不过通过 end()，我们可以把所有方法调用串联在一起。在例子中，首先是查找删除，然后end，再添加

#### jQuery ajax序列化表单值

serialize() 方法通过序列化表单值，创建 URL 编码文本字符串。可以选择一个或多个表单元素（比如 input 及/或 文本框），或者 form 元素本身。 序列化的值可在生成 AJAX 请求时用于 URL 查询字符串中。

var formSer=$('#pwd\_form').serialize();

formSer的结果类似于FirstName=Bill&LastName=Gates

只会将”成功的控件“序列化为字符串。如果不使用按钮来提交表单，则不对提交按钮的值序列化。如果要表单元素的值包含到序列字符串中，元素必须使用 name 属性。

#### Clipboard对象

clipboard.js 实现了纯 JavaScript （无 Flash）的浏览器内容复制到系统剪贴板的功能。

clipboard.on('success', function(e) {})

clipboard.on(error, function(e) {})

复制功能

var copy = new clipBoard(document.getElementById('data'), {

beforeCopy: function() {

},

copy: function() {

return document.getElementById('data').value;

},

afterCopy: function() {

}

});

剪切功能

var cut = new clipBoard(document.getElementById('data'), {

beforeCut: function() {

},

Cut: function() {

return document.getElementById('data').value;

},

afterCut: function() {

}

});

粘贴功能

var paste = new clipBoard(document.getElementById('data'), {

beforePaste: function() {

},

paste: function() {

return document.getElementById('data').value;

},

afterPaste: function() {

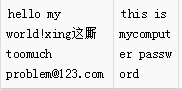
}

});

#### 行内文字换行

word-break:break-all;把所有超过容器的部分直接截断（下图中右边）

word-wrap:break-word;保留单词的完整性，一行放不下一个单词就把该单词放到下一行（下图中左边）



#### 悬浮对话框popover

悬浮对话框显示后，

var content="<p style='color:gray; font-size:12px;'>"

+"1.当输入内容为完整IP时，是精确匹配<br/>"

+"2.支持按段查询，例如:10.10.1.1-9<br/>"

+"3.支持竖线分隔符查询，例如:<br/>10.10.0.2|10.10.0.3|10.10.1.123<br/>"

+"4.不符合以上格式均为模糊查询</p>";

$("#ip\_tip").popover({

trigger:'manual',//manual 触发方式

placement : 'bottom',

title:'<div style="text-align:left; color:gray; font-size:12px;">IP查询规则</div>',

html: 'true',

content : content, //这里可以直接写字符串，也可以 是一个函数，该函数返回一个字符串；

animation: false

}).on("mouseenter", function () {

var \_this = this;

$(this).popover("show");

$(this).siblings(".popover").on("mouseleave", function () {

$(\_this).popover('hide');

}); //鼠标从对悬浮中滑出，隐藏悬浮框

}).on("mouseleave", function () {

var \_this = this;

setTimeout(function () {

if (!$(".popover:hover").length) {

$(\_this).popover("hide");

}

}, 100);

});