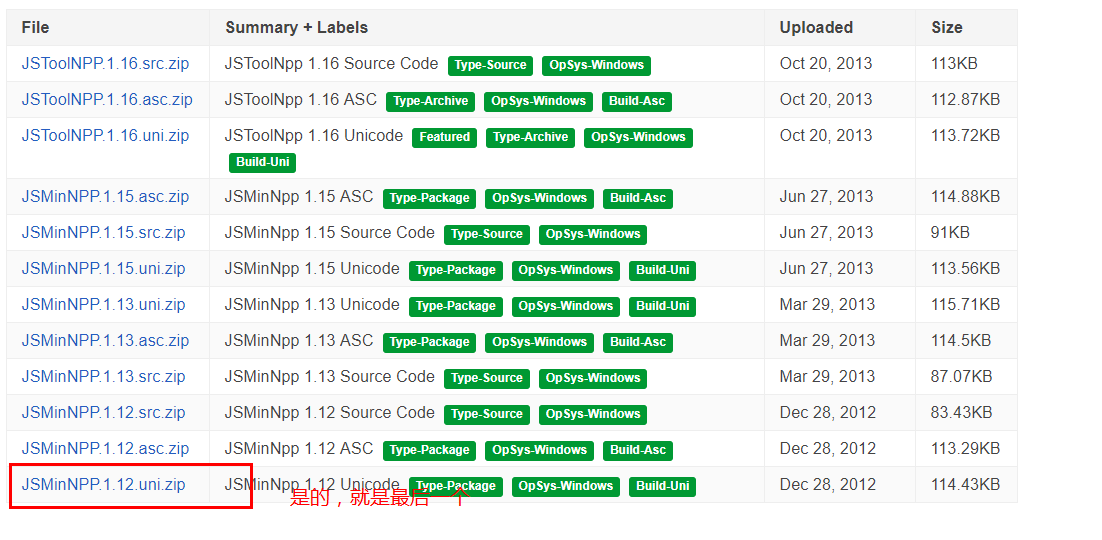
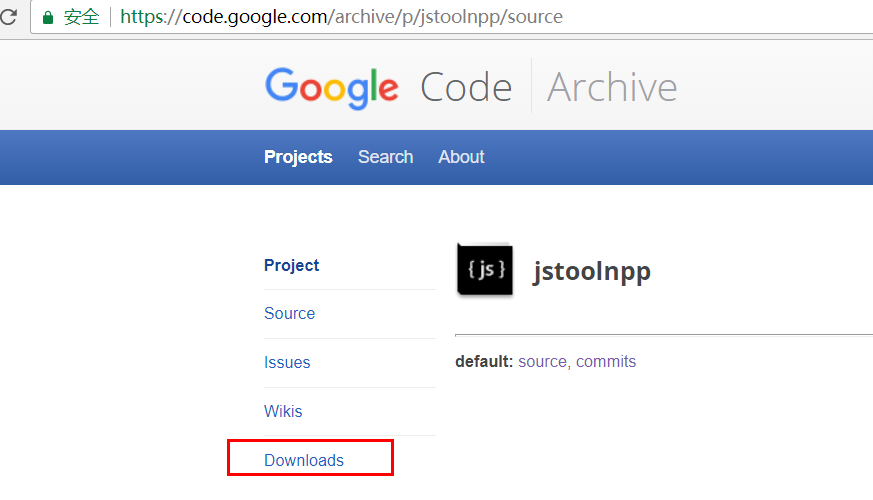
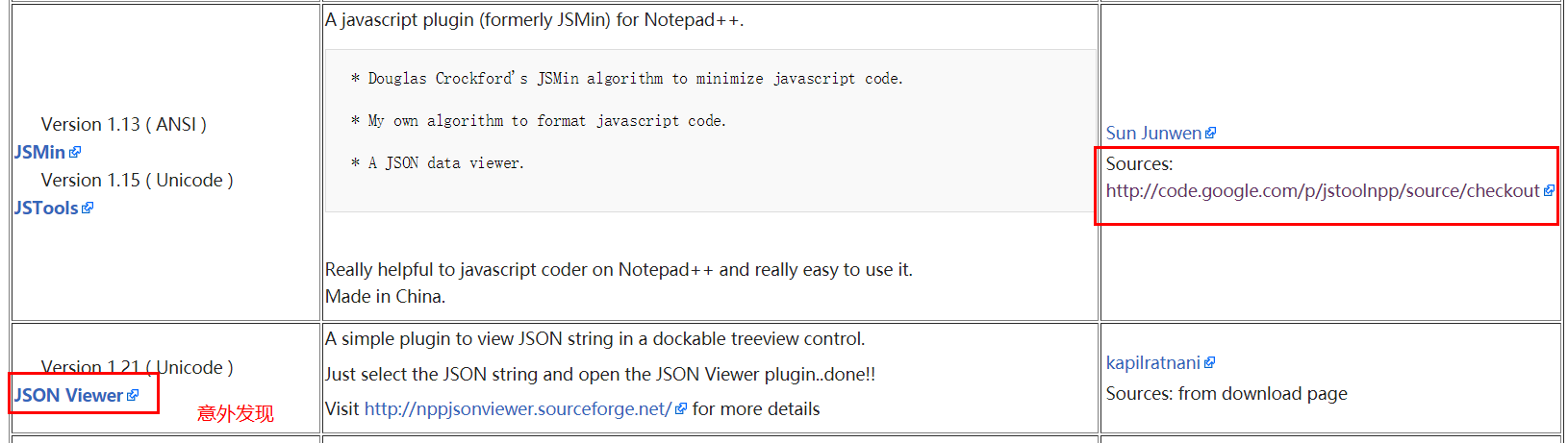
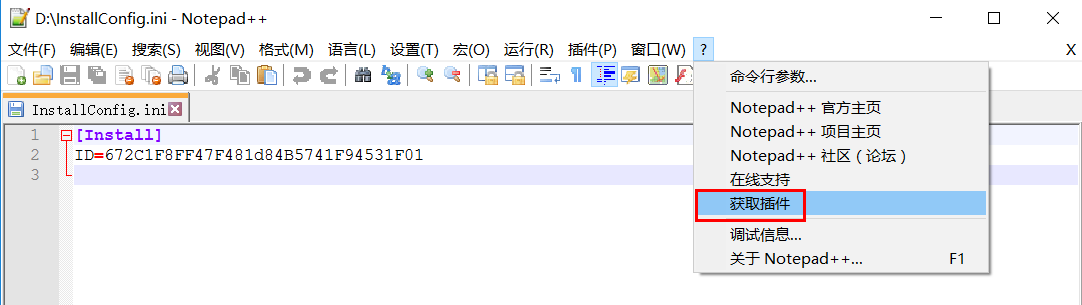
# notpad++的js格式化

<http://www.makaidong.com/JavaScript/331978.shtml>

<http://blog.csdn.net/u011037869/article/details/47170653>

## （1）下载

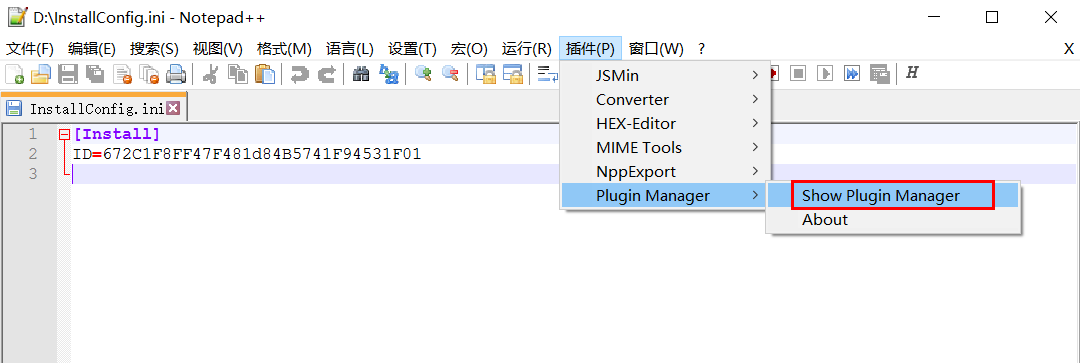
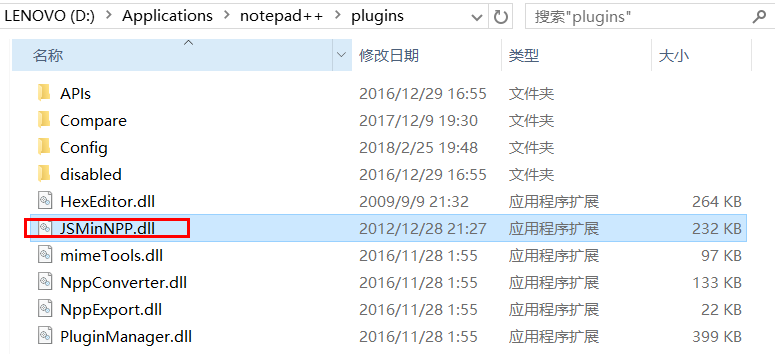
这一步需要科学上网



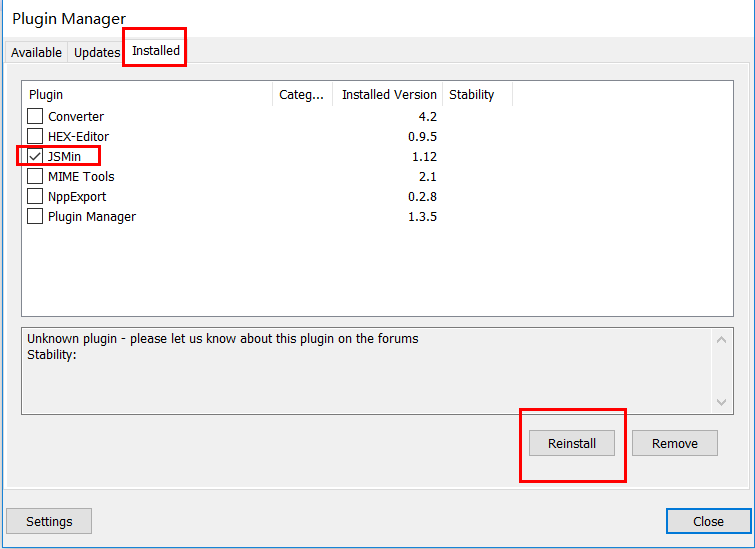
## （2）安装

解压之后，复制到notpad++的plugins中：

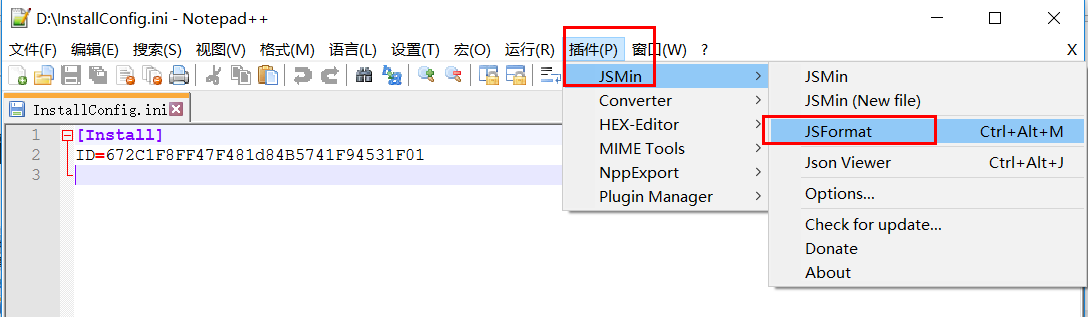
以D:\Applications\notepad++\plugins为例



我这里已经安装了，所以显示的是reinstall



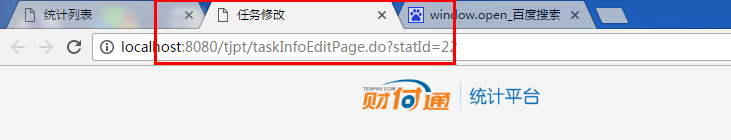
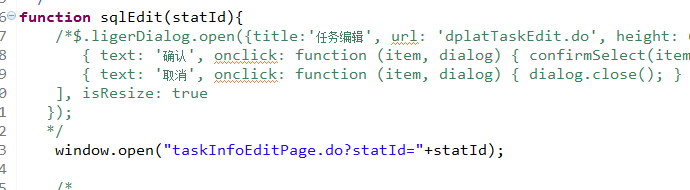
## （3）使用



# 页面跳转

## （1）[window.open("")](https://www.baidu.com/link?url=X_ZseeRbKNBJm9N1BA4br6a-ZWV07HybuzvCwqsSIsN5Af_dcuy7bkBXVE2V-jyVqIlPi26ArAM_LdHKrnAvwq&wd=&eqid=9d968910001060670000000658afa5bc" \t "https://www.baidu.com/_blank);新增加

open() 方法用于打开一个新的浏览器窗口或查找一个已命名的窗口。



## （2）window.location.href=””;本页跳转

**var** args = getUpdateArgs();

$.ligerDialog.waitting('正在更新中，请稍后...');

$.ajax({

type : 'POST',

url : 'dplatTaskUpdate.do',

dataType : 'json',

data : {

args : JSON.stringify(args)

},

success : **function**(data) {

**var** code = data.code;

**if** (code == '200'){

$.ligerDialog.closeWaitting();

window.location.href="dplatlistPage.do";

}

**else**{

$.ligerDialog.closeWaitting();

$.ligerDialog.error('更新失败:'+ data.msg);

}

}

});

# Jsp页面引入js

## （1）使用path：

1，在jsp页面定义path，将path存到session中

<%

String path = "http://" + request.getServerName() + ":" + request.getServerPort() + request.getContextPath();

request.getSession().setAttribute("path",path);

%>

2，使用path：

<script src=*"*${path}*/pages/js/tjpt/tjptCommon.js"*></script>

<script src=*"*${path}*/pages/js/dplatShellPage.js"*></script>

<link href=*"*${path}*/pages/icheck1.x/skins/square/blue.css"* rel=*"stylesheet"*>

<script src=*"*${path}*/pages/icheck1.x/icheck.js"*></script>

<script src=*"*${path}*/pages/js/myConfigPage.js"*></script>

## （2）使用相对路径：

<!-- jQuery -->

<script src=*"pages/jquery/jquery-1.8.3.min.js"*></script>

<!-- 拖拽js，必须放到jQuery的后面 -->

<script src=*"pages/jquery.gridly/javascripts/jquery.gridly.js"* type=*"text/javascript"*></script>

# JS正则\_RegExp对象

**注意，regExp正则没有单引号**

<http://blog.csdn.net/qin9r3y/article/details/24982195>

<https://technet.microsoft.com/zh-cn/library/7ddfax0d.aspx/>

1. .     IE下[^\n]，其它[^\n\r]  匹配除换行符之外的任何一个字符
2. \d    [0-9]                   匹配数字
3. \D    [^0-9]                  匹配非数字字符
4. \s    [ \n\r\t\f\x0B]         匹配一个空白字符
5. \S    [^ \n\r\t\f\x0B]        匹配一个非空白字符
6. \w    [a-zA-Z0-9\_]            匹配字母数字和下划线
7. \W    [^a-zA-Z0-9\_]           匹配除字母数字下划线之外的字符

|表示或者。

0至100间的整数 ：^([0-9]|[1-9][0-9]|100)$

## （1）i不分大小、g全局、m多行

**Js没有replaceAll，只有replace**

1，g关于replace：替换字符串中所有正则。

var str ="some some    \tsome\t\f";

var re = /\s+/g;

var s = str.replace(re,"@")); //some@some@some@

2，m多行模式 ：有换行符，几行就是几个字符串。

1. var re = /[a-z]$/; // 加上$，re后不满足匹配
2. var str = "ab\ncdef";
3. console.log(str.replace(re,"#"));//ab\ncde#
4. re =/[a-z]$/m;
5. console.log(str.replace(re,"#"));//a#\ncde#

## （2）exec分组

### （1）exec返回的数组包括

完整匹配arr[0]、子匹配arr[1]、arr[2]...。

### （2）lastParen最近一次子匹配

osVersion = "Ubuntu 8.10";

re = /^[a-z]+\s+(\d+)\.(\d+)$/i; // \.为转义后的.

arr = re.exec(osVersion);

console.log(arr[0]);//Ubuntu 8.10

console.log(arr[1]);//8

console.log(arr[2]);//10

console.log(RegExp.lastParen);//10

### （3）(?:)取消该子匹配的缓存、索引号

var string = 'April 15, 2003';  
**var** pattern = /(\w+) (\d+), (\d+)/i;

var arr = pattern.exec(string);

//arr = ["April 15, 2003", "April", "15", "2003"]，arr[1] = "April"

**var** pattern = /(?:\w+) (?:\d+), (\d+)/i;

var arr = pattern.exec(string);

//arr = ["April 15, 2003", "2003"]，arr[1] = "2003"

### （4）...(?=匹配)，返回满足匹配的...

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

var str = "abc every1 abc";

var arr = re.exec(str); //arr = ["every"]

### （5）...(?!匹配)，返回不满足匹配的...

与([^...]\*)的区别：([^...]\*)是分组

var str = "123abc456";

var re = /\d+(?!\d)/i;

var arr = re.exec(str); //arr = ["123"]

## （3）RegExp对象的静态属性

RegExp使用静态属性，是多个对象的整体属性。

（1）input 最近一次被正确匹配的字符串（传递给test,exec方法的字符串）

1. var re = /[A-Z]/;
2. var str = "Hello,World!!!";
3. var arr = re.exec(str);
4. console.log(RegExp.input);//Hello,World!!!
5. re.exec("tempstr");
6. console.log(RegExp.input);//仍然是Hello,World!!!,因为tempstr不匹配

（2）lastIndex 下一次开始匹配(检验)的字符位置，(正则必须使用g)。正确匹配后会被的exec 和 test 修改。

匹配失败（不符合），或RegExp对象的lastIndex值大于字符串长度时，再执行exec等方法会将lastIndex设为0(开始位置)

1. var str = "Hello,World!!!";
2. re = /[A-Z]/g;
3. arr = re.exec(str);
4. console.log(re.lastIndex);//1
5. arr = re.exec(str);
6. console.log(re.lastIndex);//7

（3）leftContext  返回被查找的字符串中从开始位置到最近一次被正确匹配位置间的字符

（4）rigthContext 返回被搜索的字符串中从最近一次被正确匹配位置开始到结尾间的字符

1. re = /[A-Z]/g;
2. str = "123ABC456";
3. re.test(str);
4. console.log(RegExp.leftContext);//123
5. console.log(RegExp.rightContext);//BC456  如果re=/[A-Z]+/g;则为456

## （4）^、$、+、\*、？

^为字符开始位置

$为字符结束位置

+为字符至少要出现1次，\*出现0次或以上，?出现0次或者1次

1. str = "Certainly!He loves her!";  str^为C、$为!
2. re = /^he/i;
3. console.log(re.test(str));//false
4. str = "He is a good boy!";
5. console.log(re.test(str));//true
6. re = /^he$/i;//$表示字符结束位置
7. console.log(re.test(str));//false
8. str = "He";
9. console.log(re.test(str));//true

^在[]中表示负字符集

/[^a-z]/i 表示不为字母

([^\'\"]\*) 表示不能为'也不能为"

## （5）[]作用

（1）特殊字符在([^...]\*)中

<https://www.zybang.com/question/510cf9bbbb4b2049f0c142d24ccb63cb.html>

Js中：[]里的特殊符号失去特殊意义,但\[]:^-除外。中括号里(无特意，[(]与[\(]都表示(。

可用([^...]\*)表示需要的变量。

1. **或者作用**

[+-]：只能匹配+或者-。

[a-f1-5]：匹配a~f和1-5之间的字符。

（3）并且作用。java中[[...]&&[...]]，js不支持。

[[a-z]&&[^aeiou]]：a-z 且排除掉5个元音字符aeiou

## 例子一：js通过([^...]\*)获取表名

1，[^f\(]\*、[^f(]\*都表示除f、(的任意字符多个。

2，arr[0]为完整匹配，arr[1]为子匹配1，arr[2]为子匹配2。

3，子匹配为([^\s^(]\*)，arr[2]可以取到表名。或者使用(?:)取消缓存。

var myRegexp = /(from|join|truncate)[\s]{1,}([^f]\*)/ig;

var a = "from test ( fdsfsa";

var match = myRegexp.exec(a);

match = ["from test ( ", "from", "test ( ",]

var myRegexp = /(from|join|left join|right join|truncate)[\s]{1,}([^f\(]\*)/ig;

match = ["from test ( ", "from", "test ",]

var myRegexp = /(from|join|left join|right join|truncate)[\s]{1,}([^f(]\*)/ig;

match = ["from test ( ", "from", "test ",]

## 例子二：替换所有空格、多个;

（1）将多个空格替换为""

**var** old2 = $("[name=interface]").val().replace(/\s/g, "").replace(/;{1,}/g, ";");

（2）将多个空格替换为单个空格

.replace(/\s+/g, " ")

## 例子三：获取建表语句的表名

**var** tmpsql = "CREATE TABLE if not exists tmp\_tmp\_llll20180207\_1 ( fdate BIGINT COMMENT '统计日期', fdata\_int INT COMMENT '数据' )";

**var** pattern2 = /create\s+table\s+(if\s+not\s+exists\s+)\*[^ (]+/i;

**var** arr2 = pattern2.exec(tmpsql);

**if** (arr2 != **null** && arr2.length > 0) {

**var** arr2\_length = arr2[0].split(/\s+/).length;

**var** tablename = arr2[0].split(/\s+/)[arr2\_length-1];

}

对于有没有if not exists 都可以获取到表名

# JS、JQ

## （1）js、jq取值、event、find()

### （1）取值

1，js原生属性使用、设置

.id、.innerText、.style、.children、.files[0]

2，jq需要函数获取、设置、判断

$(event.target).children()[0]

$(".notupdate").attr("属性")、$("#id").css("width","190px")

$(**this**).hasClass("具体class")

**var** sql\_num = $("#tjSQLTab > li input[value="+namenum+"]").parent().children()[0].innerText.replace(/\s+/g,"").substring(3);

3，html().replace()可能会报错，要使用(html()+"").replace()

### （2）event使用

1，js的event通过函数传递

xxx(event)

2，jq的event直接在声明jq函数使用

$("\*").click(**function** (event) {

});

### （3）find()的使用

（1）可以找id、class、name找

$(".verifyAll").each(**function**(){

**var** verifyAllType = $(**this**).find(".userDefineVarWay").find("option:selected").val();

});

（2）可以直接找标签

**var** pre = $('#isVerifySqlSuccess\_' + sqlId).parent().find("pre").children();

**var** textarea = $("#tjSQLListTab").find("textarea");

## （2）冒泡事件

冒泡事件就是点击子节点，会向上触发父节点，祖先节点的点击事件。

event.stopPropagation()、return false取消冒泡：

1，取消外部函数选择器$("#某class")的其他标签继续执行该函数（click）。

2，这里点击页面的某个标签，会触发点击该标签的所有父标签。

$("\*").click(**function** (event) {

**if** (!$(**event.target**).hasClass("hidedivclass")){

$(".hostdiv").hide();

$(".interfacediv").hide();

}

event.stopPropagation();

// return false; // 也可以阻止冒泡

});

应用：在点击搜索框之外的区域时，让搜索框消失

## （3）disabled不可编辑、display 隐藏

（1）不可用：

$().attr("disabled",**true**);

$().attr("disabled","disabled");

（2）可用：

$().attr("disabled",**false**);

（3）隐藏

$(".tijiaoBtn")[0].style.display = "none";

$("#id").css('display','none');

（4）显示

$(".gengxin")[0].style.display = "";  
$("#id").css('display','block');

## （4）children()、children

1，Jq：children()。Js：children。

2，使用了[0]后，jq变成js。

3，如果有name重复，可能先指定父标签的id

$("#tjSQLListTab textarea[id=tdwStatSql\_"+liNameNum+"]").attr("name","tdwStatSql\_" + num);

通过a获取70%的th里面的input的值

**var** tips =

"<tr>" +

"<th style='width:70%'><input style='width: 300px;'></th>" +

"<th style='width:30%'>" +

"<a href='javascript:void(0)' onclick='operateBusiness(event)'>编辑</a>" + "<a href='javascript:void(0)' onclick='keepBusiness(event)'>保存</a>"

+ "<ipnut style='dispaly:none'>" +

"</th>" +

"</tr>";

**var** target = e.target;

**var** args = **new** Object();

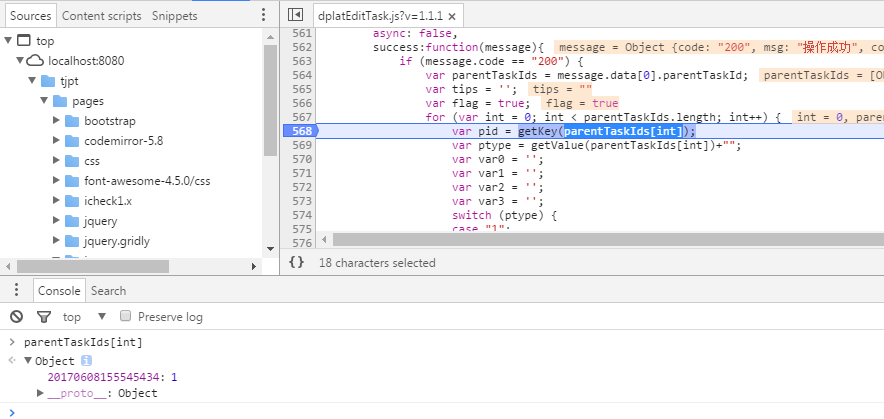
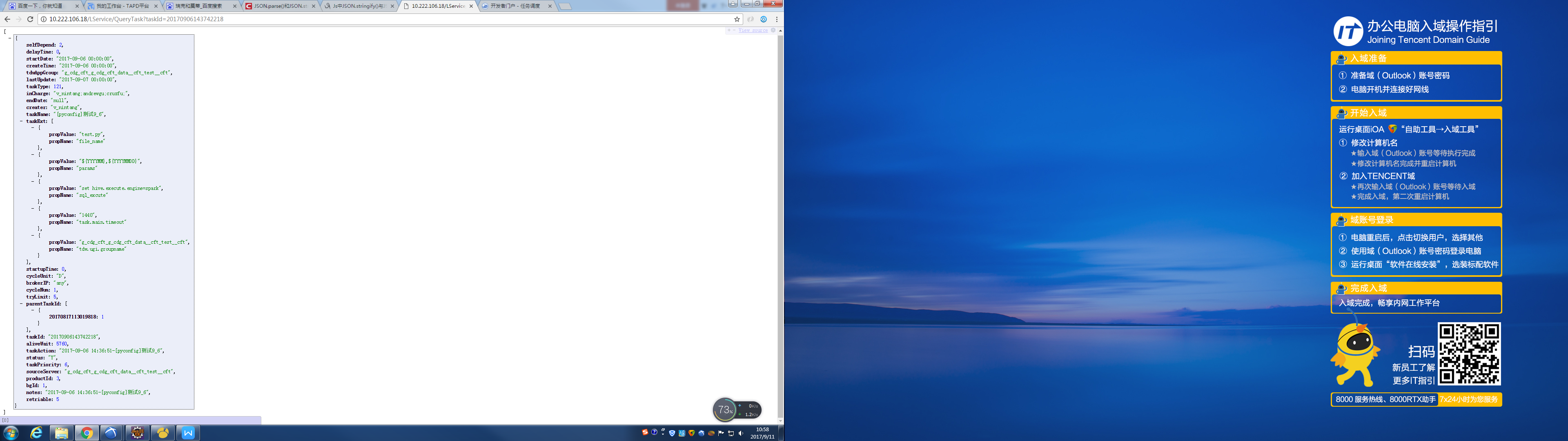
args["businessName"] = $($(target).parent().parent().**children()[0].children**).val();

## （5）js处理json数据中的json数据（json对象转数组）

<http://blog.csdn.net/chenlong220192/article/details/46622031>

（1）本身请求返回的是JSONArray中的json对象：message.data[0]

（2）同时，parentTaskId对应的值还是一个JSONArray包裹的json对象



这里每一个parentTaskIds[int]储存**一对lzid:type**键值对：（我认为多对可以存到数组中）

1，获取第一个键就退出：

**function** getKey(json){

**for**(**var** k **in** json) {

**return** k;

}

}

2，获取第一个值就退出

**function** getValue(json){

**for**(**var** k **in** json) {

**return** json[k];

}

}

（3）json对象如果含有多对，可以存到数组中

1，获取json对象的长度，不能直接使用length

function getJsonLength(jsonData){

var jsonLength = 0;

for(var item in jsonData){

jsonLength++;

}

return jsonLength;

}

2，json对象转数组：

**function** getKeyArray(json){

var jsonLength = 0;

**for**(**var** k **in** json) {

array[jsonLength] = k;

jsonLength++;

}

return array;

}

## （6）未定义

var parentTaskIds =

if(parentTaskIds != null && undefined != typeof(parentTaskIds) && "undefined" != typeof (parentTaskIds)){

}

## （7）搜索框回车、上下箭头事件

以回车为例，回车键盘抬起事件的keyCode为13

$("[name=param]").keyup(**function**(event){

**if**(event.keyCode ==13){

searchListData(); // 执行搜索

}

});

## （8）jq获取data("键")

注意：这里的data-xxx是不识别大写的，xxx只能用小写

/\*\*

\* 加载目标服务器

\*/

**function** getTargetServer(){

$.ajax({

type:'get',

url:'data/getTargetServer.do',

dataType:"json",

async: **false**,

success:**function**(message){

**if** (message.code == "200") {

**var** list = message.data;

**var** tips = "<option>请选择</option>";

**for** (**var** int = 0; int < list.length; int++) {

tips += '<option data-dbname="'+list[int].db\_name+'" value="'+list[int].lz\_server\_name+'">'+list[int].server\_name+'</option>';

}

$("[name=targetServer]").append(tips);

}

}

});

}

$("[name=targetServer]").change(**function**(){

$("[name=dbTableName]").val($("[name=targetServer] option:selected").data("dbname"));

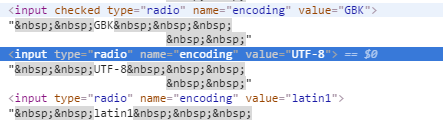
});

## （9）单选框如何在更新时展示

1，页面



2，jsp



3，js

$("[name=encoding][value="+shell\_conf["{FIT\_PARA:CHARSET}"]+"]").attr("checked",**true**);

## （10）each(function(){})不能跳出循环

jq的循环中，不能使用continue、break，否则那个方法都不会被执行

js的循环for中可以使用

# CSS

## （1）td转div这种块标签

为了限制td的宽度width:300px;将td转为div这样的区域块标签，才有效果

display: block;

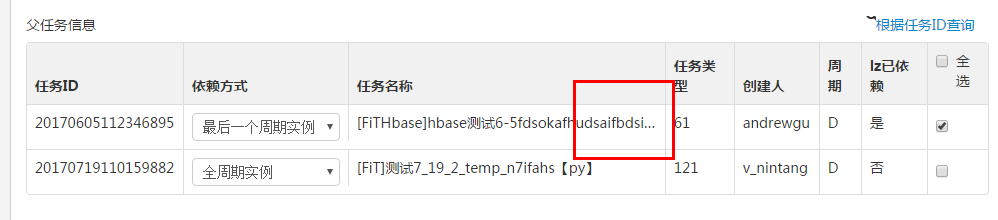
由于滚动条本身有宽度，会遮盖内容，故选择隐藏部分内容

## （2）强制不换行、超过宽度省略、显示省略号

white-space:nowrap; 不换行

overflow:hidden; 超出宽度时隐藏超出部分

text-overflow:ellipsis; 与overflow:hidden;一起使用，如果超出，显示省略号



## （3）textarea

### （1）样式

<div class=*"span10"*>

<!-- 下面一排的textarea标签不换行，否则造成框中有空格 -->

<textarea id=*"filterSQL"* name=*"filterSQL"* onmouseover="over(event,this)"></textarea>

<!-- 设置textarea自动适应高度 -->

<script>

autoTextarea($('[name=filterSQL]')[0]);//自适应高度

</script>

</div>



### （2）换行符丢失问题

在做更新等将值设置到textarea中的时候：

（1）直接将值拼到中间：会丢失换行符

<textarea id=*"filterSQL"*></textarea>

（2）使用val()设置值不会丢失换行符

## （4）表头固定

<https://www.cnblogs.com/ariter/p/5970298.html>

# 文件上传样式

<http://www.haorooms.com/post/css_input_uploadmh>



## （1）css

*.file* {

position: *relative*;

display: *inline-block*;

background: *#D0EEFF*;

border: *1px* *solid* *#99D3F5*;

border-radius: *4px*;

padding: *4px* *12px*;

overflow: *hidden*;

color: *#1E88C7*;

text-decoration: *none*;

text-indent: *0*;

line-height: *30px*;

}

*.file* **input** {

position: *absolute*;

font-size: *100px*;

right: *0*;

top: *0*;

opacity: *0*;

}

*.file:hover* {

background: *#AADFFD*;

border-color: *#78C3F3*;

color: *#004974*;

text-decoration: *none*;

}

## （2）jsp

<link rel=*"stylesheet"* href=*"pages/css/uploadfile.css"*>

<a href=*"javascript:;"* class=*"file"*>

<span id=*"span"*>选择文件</span>

<input type=*"file"* name=*"pyfile"* id=*"pyfile"*>

</a>

## （3）js

/\*\*

\* 文件选择

\*/

$(".file").on("change","input[type='file']",**function**(){

**var** filePath=$(**this**).val();

**var** arr=filePath.split('\\');

**var** fileName=arr[arr.length-1];

$("#span").html(fileName);

// 必不可少的一步：改变span的 value

$("[name=file\_name]").val(fileName);

});

## （4）注意

1，必选要a标签内部的span或者input的value发生改变，a标签才能触发change事件多次选择文件。

2，否则只能选择一次文件，但是input的value一旦发生改变，就会将file文件清空

3，因此，只能改变span的value

## 把一个div隐藏起来

.tlinks {  
 text-indent: -9999px;  
 height: 0;  
 line-height: 0;  
 font-size: 0;  
 overflow: hidden;  
}

# Html页面中的函数

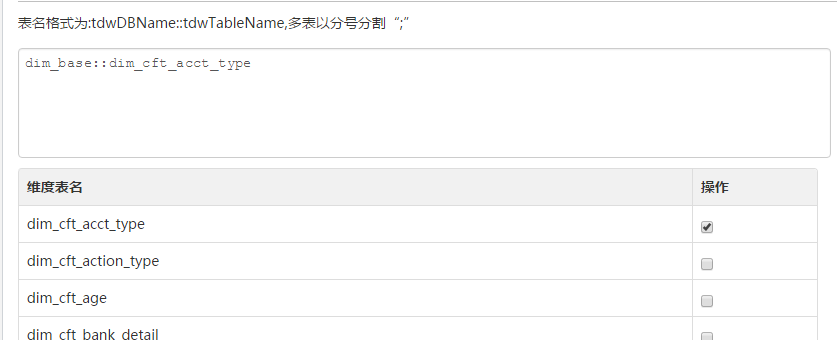
## （1）特殊符号

Html中的onlick=”函数名()”函数名中不能出现\_等特殊符号，否则不会执行

## （2）不能与其他标签的name相同

否则会提示is not a function

## （3）event、this：方法名(event,this)



/\*\*

\* 加载维度表

\*/

**function** getDimensionTables(){

**var** args = **new** Object();

args["username"] = $("[name=userName]").val();

args["password"] = $("[name=password]").val();

$.ajax({

type:'get',

url:'getDimensionTables.do',

dataType:"json",

data:{

args: JSON.stringify(args)

},

async:**false**,

success:**function**(message){

**if** (message.code == '200') {

$("#dimensionbody tr").remove();

**var** list = message.data;

**var** tips = "";

**for** (**var** int = 0; int < list.length; int++) {

tips += "<tr>" +

"<td>"+list[int]+"</td>" +

"<td><input class='dimensionTables' type='checkbox' onclick='dimensionTablesClick(**event,this**)'></td>"+

"</tr>";

}

$("#dimensionbody").append(tips);

} **else** {

$.ligerDialog.warn("维度表读取失败,desc:"+message.msg);

}

}

});

}

/\*\*

\* 维度表选择

\*/

**function** dimensionTablesClick(e,t){

**var** target = e.target;

**var** dimensionTable = $(target).parent().parent().children()[0].innerText;

**if**(t.checked==**true**){

**var** lz = $("textarea[name=dimensionTable]").val();

**if** (lz == '') {

$("textarea[name=dimensionTable]").val('dim\_base::' + dimensionTable);

} **else** {

$("textarea[name=dimensionTable]").val(lz+';'+ 'dim\_base::' + dimensionTable);

}

}**else**{

**var** lz = $("textarea[name=dimensionTable]").val();

**var** newlz = '';

**var** lzs = lz.split(";");

**for** (**var** i = 0; i < lzs.length; i++) {

**if** (lzs[i].split("::")[1] != dimensionTable) {

newlz += lzs[i]+';';

}

}

**if** (newlz != '') {

$("textarea[name=dimensionTable]").val(newlz.replace(/^[^A-Z a-z 0-9\_:]+|[^A-Z a-z 0-9\_:]+$/g, ""));

} **else** {

$("textarea[name=dimensionTable]").val('');

}

}

}

## （4）data传递值规避函数传值

（1）函数中，普通值不能与this一起传递，可以与event一起传递，event可以与this一起

1，通过data-xx传递值

operStr = '<td><a href="#" **data-lzId**="'+ rowData.lzId +'" onclick=sqlEdit('+ rowData.statId +',**event**) readonly="readonly">修改</a>|<a href="#" onclick="deleteTJTaskByStatId('+ rowData.statId +')" </a>删除</td>';

**错误**处理方式：通过函数传递值，**数值过长，进行四舍五入取0，导致报错**

operStr = '<td><a href="#" onclick=sqlEdit('+ rowData.statId +','+ rowData.lzId +') readonly="readonly">修改</a>|<a href="#" onclick="deleteTJTaskByStatId('+ rowData.statId +')" </a>删除</td>';

2，event.target.dataset.xx取值

**function** sqlEdit(statId,e){

window.open("taskInfoEditPage.do?statId="+statId+"&lzId="+e.target.dataset.lzid);

}

（2）this可以使用title，event不能使用

<input name=*"sourceFilePath"* type=*"text"* onmouseover="over(event,this)" style="width: *450px*;"/>

<input name=*"sourceColumnNames"* type=*"text"* onmouseover="over(event,this)" style="width: *450px*;"/>

/\*\*

\* 鼠标移动

\*/

**function** over(e,t){

**var** name = $(e.target).attr("name");

**if** (name == "sourceFilePath") {

t.title="规则： /stage/interface/BU名/应用名/表名/${YYYYMMDD}/in/";

}**else** **if** (name == "sourceColumnNames") {

t.title="源文件的栏位名称，以逗号分割（结尾不能是逗号），必须保证列数和文件内容一致。例如column1,column2,column3 注：不允许输入空格，源文件栏位名称只能由大小写字符、数字和下划线组成";

}**else** **if** (name == "tableName") {

t.title="入库数据表名";

}**else** **if** (name == "targetColumnNames") {

t.title="表的列名，英文逗号分隔，表示的列的内容顺序需和DB列字段保持一致。决定从临时表往目的表里写的字段顺序。日期和常量需要用中括号包起来，例如：[${YYYYMMDD}],['test'],column\_1,column\_2,column\_3";

}

}

## （5）规避html()

textarea使用html()会将>符号变为gt

推荐使用val()、text()

# Js动态加载下拉列表

<http://bbs.csdn.net/topics/391056834?page=1>

## （1）要记录后再查询新select

$("[name=hostName]").click(**function** () {

// 记录之前的值。

**var** slv = $("[name=hostName]").val();

$("[name=hostName] option").remove();

hostNameInfo();

// 加载之后要将之前的值给他。否则造成丢失。

$("[name=hostName]").val(slv);

});

**function** hostNameInfo(){

$.ajax({

type : 'get',

url : 'hosts/selectall.do',

dataType : 'json',

async: **false**,

success : **function**(message) {

**var** code = message.code;

**if** (code == '200') {

**var** list = message.data;

**var** tips = "<option value=''>请选择</option>";

**for** (**var** int = 0; int < list.length; int++) {

tips += "<option value="+list[int]+">"+list[int]+"</option>";

}

$("[name=hostName]").append(tips);

}**else**{

$.ligerDialog.warn("初始化hostname列表为空");

}

}

});

}

例子：因为click会同时触发select、option，因此在option触发click的时候要记录已选值

<select style='width: 300px;' onclick='showTheme(event)' name='pid'><option value=''>请选择主题</option></select>

**function** showTheme(event){

$.ajax({

type:'GET',

url:"data/getAllTheme.do",

dataType:'json',

async:**false**,

success:**function**(message){

**if** (message.code == '200') {

**var** slv = $(event.target).find("option:selected").val();

$(event.target).empty();

**var** list = message.data;

**var** tips = '<option value="">请选择主题</option>';

**for** (**var** int = 0; int < list.length; int++) {

tips += '<option value="'+list[int].id+'">'+list[int].cname+'</option>';

}

$(event.target).append(tips);

$(event.target).val(slv);

}

}

});

}

## （2）ajax查询list里面是map取值

$.ajax({

type : 'POST',

url : 'hbase/selectAll.do',

dataType : 'json',

success : **function**(message) {

**var** code = message.code;

**if** (code == '200') {

**var** list = message.data;

**var** tips = "<option value=''>请选择</option>";

**for** (**var** int = 0; int < list.length; int++) {

tips += "<option value="+list[int].e\_name+">"+list[int].e\_name+"</option>";

}

$("[name=maintable]").append(tips);

} **else** {

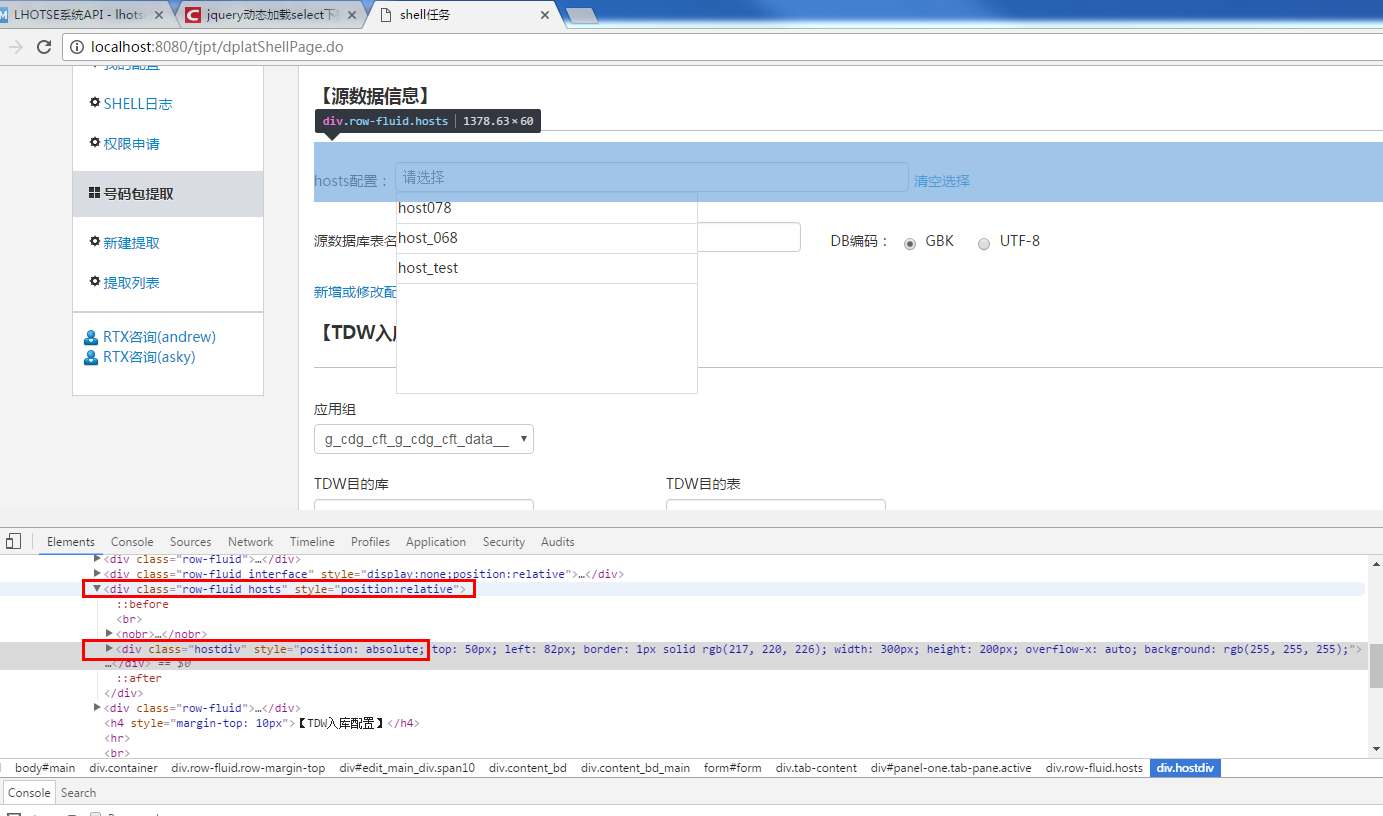
$.ligerDialog.warn("主表为空");

}

}

});

# Js动态加载div+ipnut=select



## （1）Div设置

<http://blog.sina.com.cn/s/blog_a595138e01013ogk.html>

1. 外部div设置为relative，内部div设置为absolute，漂浮状态，宽高不会影响其他区域。
2. 设置边框、宽高、超过高度出现滚动条overflow-x:auto。
3. 背景色background，否则会透明。
4. Table设置为border-collapse:*collapse*;使得tr有底线。

<table style="border-collapse:*collapse*;width:*300px*;" class=*"hosttable"*>

</table>

## （2）动态加载

（1）点击时加载查询到div的table中。增加tr的底线。

$("[name=hostName]").click(**function**(){

hostNameInfo();

$(".hostdiv").show();

});

**function** hostNameInfo(){

$.ajax({

type : 'get',

url : 'hosts/selectall.do',

dataType : 'json',

async: **false**,

success : **function**(message) {

**var** code = message.code;

**if** (code == '200') {

**var** list = message.data;

// 移除旧tr

$(".hosttable tr").remove();

**var** tips = '';

**for** (**var** int = 0; int < list.length; int++) {

tips += '<tr style="width:100%;height: 30px;border-bottom:1px solid #d9dce2"><td><span onclick="hosttable(event)">'+list[int]+'</span></td></tr>';

}

$(".hosttable").append(tips);

}**else**{

$.ligerDialog.warn("初始化hostname列表为空");

}

}

});

}

（2）键盘抬起时索引局部到div的table中。

$("[name=hostName]").keyup(**function**(){

**var** host = $("[name=hostName]").val();

**if** (host.substring(host.length-1,host.length) == ";" || host == '') {

hostNameInfo();

**return**;

}

**var** is = $("[name=hostName]").val().split(";");

**var** i = is[is.length-1];

$.ajax({

type : 'get',

url : 'hosts/selectall.do?i=' + i,

dataType : 'json',

success:**function**(message){

**var** code = message.code;

**if** (code == '200') {

**var** list = message.data;

$(".hosttable tr").remove();

**var** tips = '';

**for** (**var** int = 0; int < list.length; int++) {

tips += '<tr class="hidedivclass" style="width:100%;height: 30px;border-bottom:1px solid #d9dce2"><td class="hidedivclass"><span class="hidedivclass" onclick="hosttable(event)">'+list[int]+'</span></td></tr>';

}

$(".hosttable").append(tips);

}**else**{

$(".hosttable tr").remove();

}

}

});

});

## 判断input中的值是否在列表中

先new Array()数组存储span中的文本，再判断是否在数组中

$.inArray(某个串, 数组) == -1可以判断某个串是否在数组中

**var** hostNames = $("[name=hostName]").val().split(";");

**var** span = **new** Array();

**if** ($(".hosttable span").length == 1) {

span[0] = $(".hosttable span").html();

} **else** {

**for** (**var** int = 0; int < $(".hosttable span").length; int++) {

span[int] = $(".hosttable span")[int].innerText;

}

}

**for** (**var** int = 0; int < hostNames.length; int++) {

**if** ($.inArray(hostNames[int], span) == -1) {

$.ligerDialog.warn("hosts与已有配置不符");

**return**;

}

}

**if**($("[name=hostName]").val().replace(/\s/g, "") == "" || $("[name=hostName]").val().replace(/\s/g, "") == "请选择"){

$.ligerDialog.warn("hosts配置为空");

**return**;

}

## （4）点击div、input以外的区域$("\*").click()，隐藏div

<http://bbs.csdn.net/topics/390704559>

$("\*").click(**function** (event) {

event.stopPropagation();

});

（1）所有标签被点击都将被触发判断。

（2）stopPropagation()函数阻止冒泡，当**某元素触发click事件，剩余$("\*")不再继续触发**click事件（冒泡触发）。

（3）ipnut、div、div所有子标签（table、tr、td、span）都需要添加class="hidedivclass"。

$("\*").click(**function** (event) {

**if** (!$(**this**).hasClass("hidedivclass")){

$(".hostdiv").hide();

$(".interfacediv").hide();

}

event.stopPropagation();

});

# JS时间设置

## **（1）JS设置时间输入**



<input type=*"date"* name=*"starttime"*>

<input type=*"date"* name=*"endtime"*>

/\*\*

\* 默认导入时间段

\*/

**function** defaulttime(){

**var** ddd = **new** Date();

ddd.setTime(ddd.getTime()-24\*60\*60\*1000);

**var** day =ddd.getDate();

**if**(ddd.getMonth()<10){

**var** month = "0"+(ddd.getMonth()+1);

}

**if**(ddd.getDate()<10){

day = "0"+ddd.getDate();

}

**var** starttime = ddd.getFullYear()+"-"+month+"-"+day;

$("[name=starttime]").val(starttime.toString());

**var** ddd = **new** Date();

**var** day =ddd.getDate();

**if**(ddd.getMonth()<10){

**var** month = "0"+(ddd.getMonth()+1);

}

**if**(ddd.getDate()<10){

day = "0"+ddd.getDate();

}

**var** endtime = ddd.getFullYear()+"-"+month+"-"+day;

$("[name=endtime]").val(endtime);

}

## （2）JS获取当前时间戳（1970至今秒）

（1）第一种方法

var timestamp = Date.parse(new Date()); //结果：1280977330000  
（2）第二种方法

var timestamp = (new Date()).valueOf(); //结果：1280977330748

（3）第三种方法

var timestamp=new Date().getTime()； //结果：1280977330748

## js时间戳转时间格式

（1）英文格式时间

$("[name=createtime]").val(**new** Date(parseInt(map.create\_time)));



（2）中文格式时间

$("[name=createtime]").val(**new** Date(parseInt(map.create\_time)).toLocaleString());



## （4）js获取日历时间

<http://www.cnblogs.com/hanwenhua/articles/3853352.html>

/\*\*

\* 格式化时间

\*/

**function** getNowFormatDate(time) {

**var** date = **new** Date(time); // 如果是当前，**new** Date()

**var** seperator1 = "-";

**var** seperator2 = ":";

**var** month = date.getMonth() + 1;

**var** strDate = date.getDate();

**var** hour = date.getHours();

**if** (month >= 1 && month <= 9) {

month = "0" + month;

}

**if** (strDate >= 0 && strDate <= 9) {

strDate = "0" + strDate;

}

**if** (hour >= 0 && hour <= 9) {

hour = "0" + hour;

}

**var** Minutes = date.getMinutes()+"";

**if**(Minutes.length == 1){

Minutes = "0" + Minutes;

}

**var** Seconds = date.getSeconds()+"";

**if**(Seconds.length == 1){

Seconds = "0" + Seconds;

}

**var** currentdate = date.getFullYear() + seperator1 + month + seperator1 + strDate + hour + seperator2 + Minutes + seperator2 + Seconds;

**return** currentdate.replace(/-|:/g,"");

}

得到：201710121436

## （5）java设置日历

**public** **static** Date addDate(Date date, **long** day) **throws** ParseException {

**long** time = date.getTime(); // 得到指定日期的毫秒数

day = day \* 24 \* 60 \* 60 \* 1000; // 要加上的天数转换成毫秒数

time += day; // 相加得到新的毫秒数

**return** **new** Date(time); // 将毫秒数转换成日期

}

**long** currentTimeMillis = System.*currentTimeMillis*();

Calendar fews\_days\_calendar = Calendar.*getInstance*();

fews\_days\_calendar.setTime(**new** Date(currentTimeMillis));

DateFormat format = DateUtil.*yyyyMMdd\_threadLocal*.get();

String past\_1\_day = format.format(*addDate*(fews\_days\_calendar.getTime(), -1));

String past\_30\_day = format.format(*addDate*(fews\_days\_calendar.getTime(), -30));

# Js版本号

<http://www.jb51.net/web/200602.html>

客户端会缓存css或js文件，改变版本号，客户端浏览器就会重新下载新的js或css文件，在js或css后加?v= 版本号的用法如下：

形如.css?v=与.js?v= 或 .css?version=与.js?version=

<script src=*"*${path}*/pages/js/dplateditPage.js?v=1.1"*></script>

等价于：

<script src=*"*${path}*/pages/js/dplateditPage.js"*></script>

但是浏览器会重新加载js

<title>新增统计</title>

<jsp:include page=*"common\_head.jsp"*></jsp:include>

<!-- checkbox样式 -->

<script src=*"*${path}*/pages/js/tjpt/tjptCommon.js"*></script>

<!-- 在引用的js后面加上参数 -->

<script src=*"*${path}*/pages/js/dplateditPage.js?v=1.1"*></script>

<link href=*"*${path}*/pages/icheck1.x/skins/square/blue.css"* rel=*"stylesheet"*>

<script src=*"*${path}*/pages/icheck1.x/icheck.js"*></script>

<script src=*"*${path}*/pages/js/myConfigPage.js"*></script>

# ajax

## （1）同步ajax

$.ajax({

async: false

## （2）then()异步依次执行ajax

<https://www.2cto.com/kf/201507/424202.html>

（1）后台

@RequestMapping("test11\_27\_1")

@ResponseBody

**public** Message test11\_27\_1(HttpServletRequest request) **throws** Exception{

Message message = **new** Message();

message.setCode("200");

message.setData("test11\_27\_1");

**return** message;

}

@RequestMapping("test11\_27\_2")

@ResponseBody

**public** Message test11\_27\_2(HttpServletRequest request) **throws** Exception{

Message message = **new** Message();

Thread.*sleep*(4000);

message.setCode("200");

message.setData("test11\_27\_2");

**return** message;

}

@RequestMapping("test11\_27\_3")

@ResponseBody

**public** Message test11\_27\_3(HttpServletRequest request) **throws** Exception{

Message message = **new** Message();

message.setCode("200");

message.setData("test11\_27\_3");

**return** message;

}

（2）前端：jquery必须使用**jquery1.8**之后的版本才有then函数

<button id=*"button11\_27"*>测试</button><br>

<input type=*"button"* id=*"testa"*/><br>

<input type=*"button"* id=*"testb"*/><br>

<input type=*"button"* id=*"testc"*/>

由于**使用了then**，**虽然是异步**，但是必须要等到每一个ajax返回参数之后，才会继续执行其他的ajax。

这样做的好处：前面的ajax都是异步，**特别是第一个ajax是异步的**，这样页面刚加载的时候就不用查询很多的参数（比如第一个ajax也是异步）。

$(**function** (){

test();

});

**function** test(){

**var** promise1 = $.ajax({

url: "test11\_27\_1.do",

type: "get",

dataType:"json",

success:**function** (message) {

$("#testa").val("显示a");

}

});

**var** promise2 = promise1.then(**function**(){

**return** $.ajax({

url: "test11\_27\_2.do",

type: "get",

dataType:"json",

success:**function** (message) {

$("#testb").val("显示b");

}

});

});

**var** promise3 = promise2.then(**function**(){

**return** $.ajax({

url: "test11\_27\_3.do",

type: "get",

dataType:"json",

success:**function** (message) {

$("#testc").val("显示c");

}

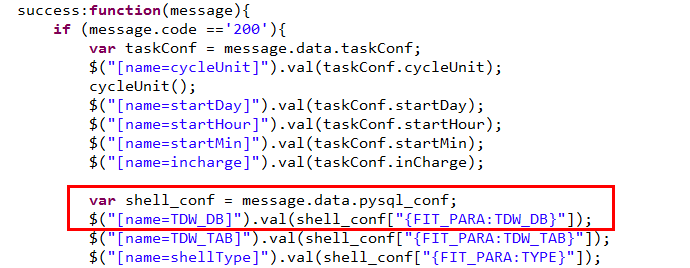
});

});

}

## **（3）返回参数**键有特殊符号用[xxx]

特殊符号数据展示data[“特殊符号的参数”]



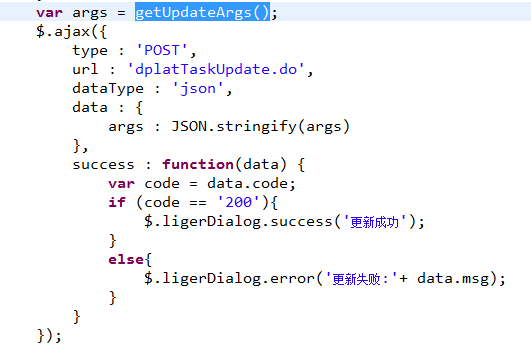
## （4）传递参数

<http://blog.csdn.net/pamki/article/details/60572436>

### （1）JSON.stringify()

JSON.stringify()将**json结构的对象**转为json结构的字符串

**避免springmvc繁琐的数据解析！**



### （2）JSON.parse()

JSON.parse()将json格式的字符串转为json格式的对象，就是JSON.stringify的反向方法

用途：比如

[ {name: "1", sex:1, age: 1}, {name: "2", sex:0, age: 2}, {name: "3", sex:1, age: 3} ]

这种格式的数据，如果没有base64编码导致格式丢失，可以在前段将数据恢复为json格式的对象，不需要再后端使用JSONObject.fromobject(String str)方法变为对象然后再给前端。

### （3）$("#xxx").serialize()（了解）

Jq的方法，将表单对象序列化为键值对字符串，个人感觉很不爽。

var data = $("#formID").serialize();

必须使用

String 键\_name request.getparam("键\_name")

来获取值\_value

### （4）serializeArray()

Jq的方法，将表单对象转为**json格式的对象，个人感觉很强。**

var jsonData = $("#formID").serializeArray(); // 转为JSONArray



相当于一个var args = new Object();键相当于表单中的name，值相当于表单中的value。再配合JSON.stringify(jsonData)，直接将得到json结构的字符串，可以直接传递到Controller。

var jsonData = $("#formID").serializeArray();

$.ajax({

type: "POST",

url: url,

data: {args : JSON.stringify(jsonData)},

success: **function**(msg){

$.messager.alert('提示','新增内容成功!');

$("#contentList").datagrid("reload");

TT.closeCurrentWindow();

},

});

## （5）formdata上传文件、普通参数

<http://www.cnblogs.com/jiangxiaobo/p/5892146.html>

（1）我估计formdata内部封装了文件上传三要素。

**var** formData = **new** FormData();

formData.append("文件参数名", $("[name=file]")[0].files[0]);

formData.append("普通参数名", $("[name=username]").val());

（2）ajax设置

<https://stackoverflow.com/questions/20795449/jquery-ajax-form-submission-enctype-multipart-form-data-why-does-contentt>

contentType: **false**,是为了防止jq将multipart/form-data的请求头。

processData: **false**,是为了防止jq将formdata参数转换为string类型。

<head>

<title>Insert title here</title>

<jsp:include page=*"common\_head.jsp"*></jsp:include>

<script type=*"text/javascript"*>

$(**function**(){

$("#upload").click(**function** () {

**var** formData = **new** FormData();

formData.append("myfile", $("[name=file]")[0].files[0]);

formData.append("username", $("[name=username]").val());

$.ajax({

url: "myfiletest.do",

type: "POST",

data: formData,

contentType: **false**,

processData: **false**,

success:**function** (message) {

**if** (message.code == "200") {

alert("上传成功");

}

}

});

});

});

</script>

</head>

<body>

用户名：<input type=*"text"* name=*"username"*/><br>

请选择文件<input type=*"file"* name=*"file"*/><br>

<input type=*"button"* id=*"upload"* value=*"上传"* />

</body>

## （6）Freemarker传递参数

在hmtl中不能进行if判断，将参数传递到Controller进行判断。

（1）Fdb、ftable通过freemarker传到html静态页面：

<span id=*"fdb"*>${fdb}</span>::<span id=*"ftable"*>${ftable}</span>

1. 不能直接将freemarker页面的参数如${fdb}当成EL表达式${fdb}传递过去，先存到span，从span中取出来。

**function** GetJsonData() {

**var** json = {

"fdbName": $('#fdb').[0].innerText,

"ftableName": $('#ftable').[0].innerText,

"fuserRtx": "用户9527"

};

**return** json;

}

$(**function**(){

$.ajax({

type:"get",

url:"/metadata/frequent.do",

data:GetJsonData(),

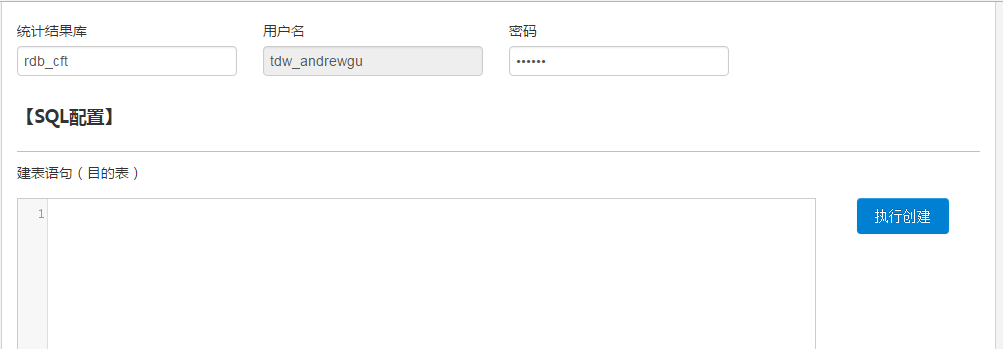
contentType:"application/json",

dataType: "json"

});

});

# 校验密码，定义校验旗帜



## （1）页面加载时就查询密码。



## （2）获取密码



## （3）密码与输入框不一致，有更新的可能。

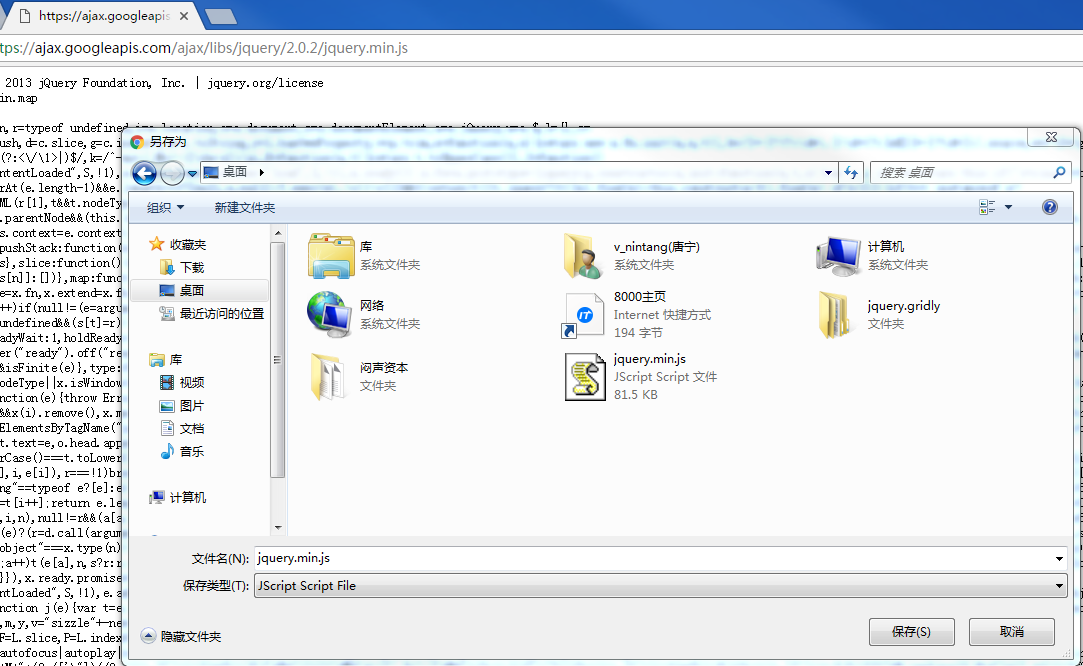
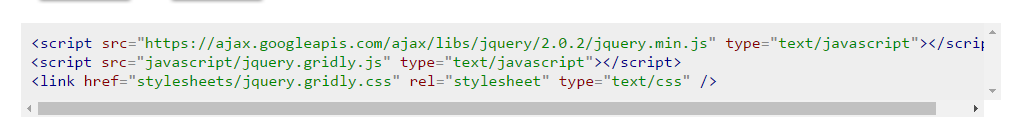
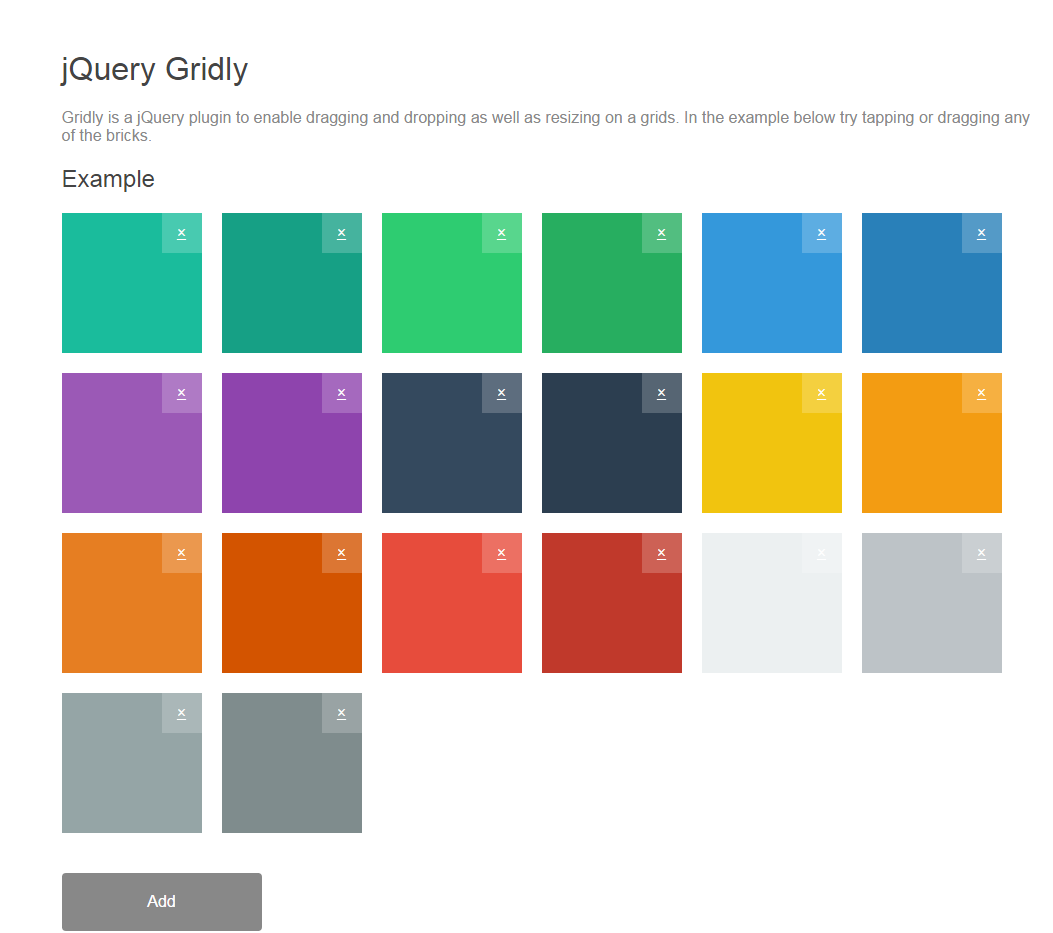


## （4）有更新的必要并且校验成功再更新



# 拖拽功能

## （1）访问依赖js，Ctrl+S保存到本地js



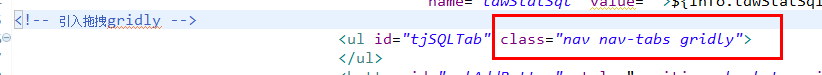
## （2）顺序：jquery.gridly必需在jquery后面



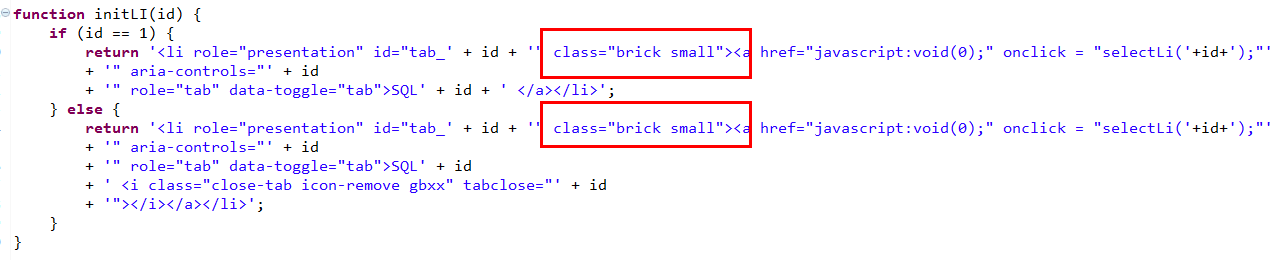
## （3）用法



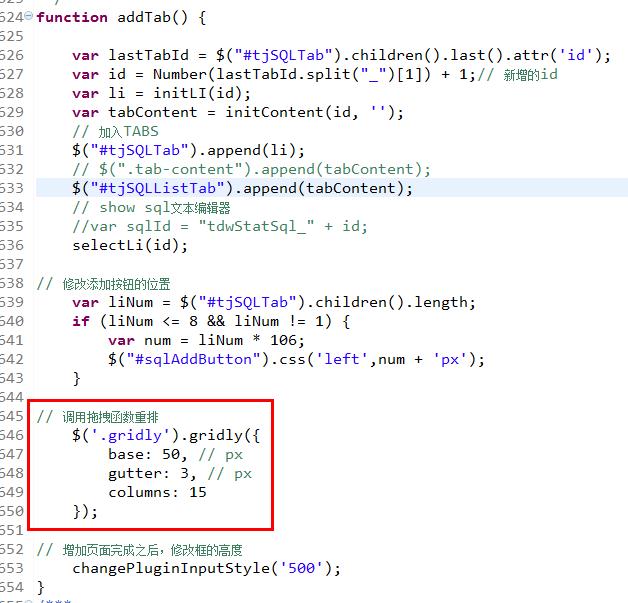
Jsp中：



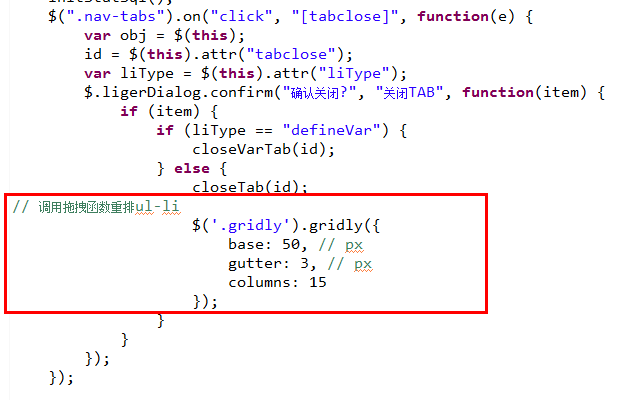
Js中：这里li也可以不写gridly的class，就不是small的大小，是其他样式的大小



添加按钮事件：



删除按钮：



# 拖拽后获取div或li对应textarea的内容

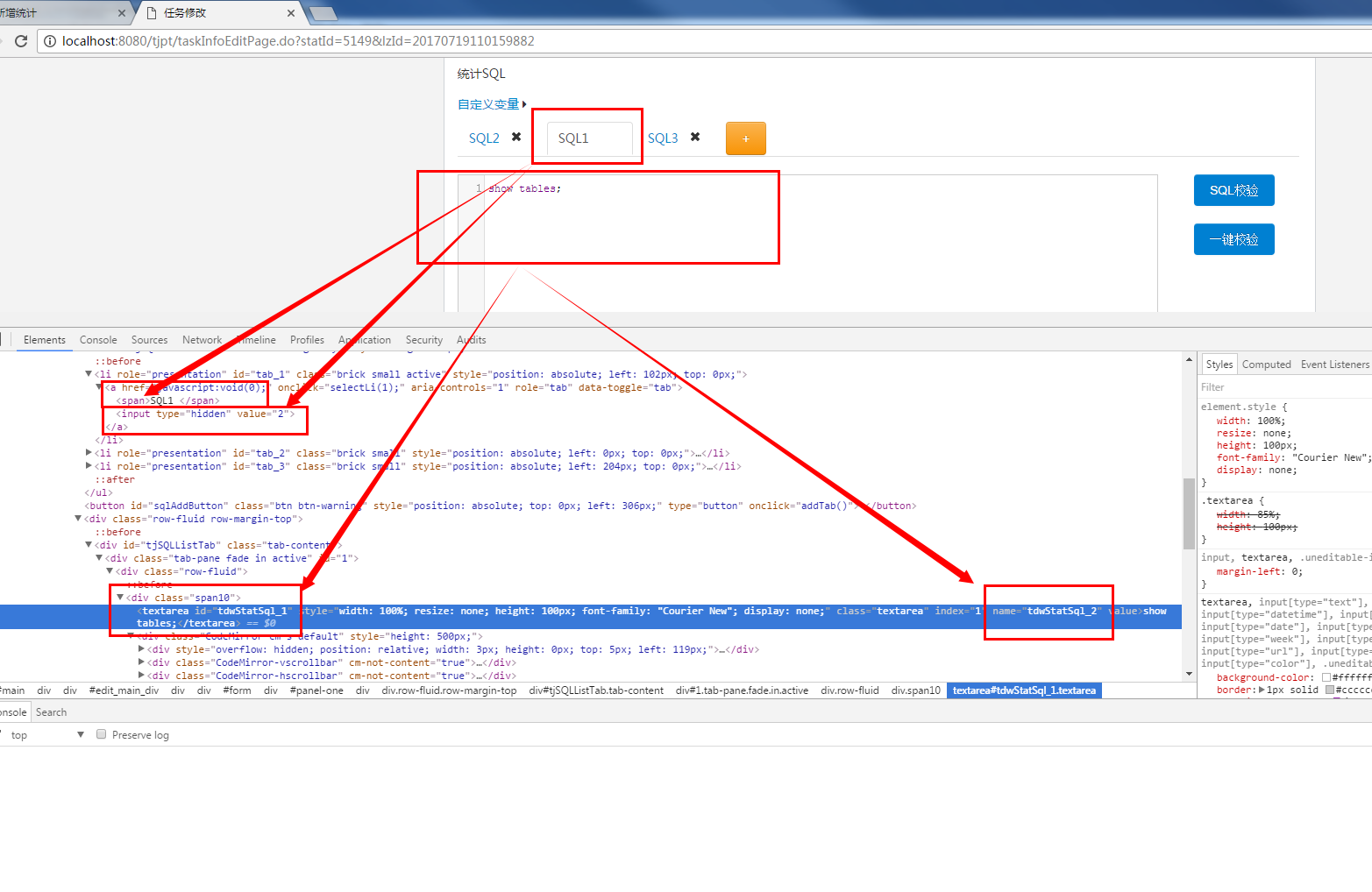
## （1）不变对应，改变是顺序

（1）span名（SQL1）、textarea的id（tdwStatSql\_2）不会改变

li中的span永远对应自己的textarea，**单独校验按钮对应自己的SQL**。

（2）根据相对位置，改变li中input的id、textarea的name。

通过name的顺讯依次获取内容，**一键校验、最后提交按照name的顺序**。



## （2）拖拽有延迟，改变name可能会失败，就多次调用

**function** sleep(numberMillis) {

**var** now = **new** Date();

**var** exitTime = now.getTime() + numberMillis;

**while** (**true**) {

now = **new** Date();

**if** (now.getTime() > exitTime)

**return**;

}

}

$verifyAllSql = **function** (obj) {

**var** linum = $("#tjSQLTab > li").length;

**for**(**var** int = 1; int <= linum ; int ++){

**var** id = int;

**var** sqlName = $($("input[value="+id+"]").parent().children()[0]).html();

**var** sqlId = (sqlName+"").replace(/\s/g,"").substring(3);

**var** sqlEditContent = $("#tjSQLListTab textarea[name=tdwStatSql\_"+id+"]").html();

**var** udpatenum = 0;

**while** ("undefined" == **typeof** (sqlEditContent) || **undefined** == **typeof** (sqlEditContent)){

// 重新根据位置计算name，尝试获取sql

updatenameagain();

sqlEditContent = $("#tjSQLListTab textarea[name=tdwStatSql\_"+id+"]").html();

sleep(100);

**if**(udpatenum >= 5){

$.ligerDialog.error("拖拽已崩溃，请刷新页面！");

**return** **false**;

}

udpatenum ++;

}

sqlEditContent = sqlEditContent + "";

// ajax验证。。。

//......

}

//......

}

## （3）($().html()+"").replace()，直接replace会报错

//重新计算li对应textarea的name，该方法失败可能较大，需要多次调用

**function** updatenameagain(){

**for** (**var** int = 0; int < $("#tjSQLTab li").length; int++) {

**var** left = $("#tjSQLTab li")[int].style.left;

**var** top = $("#tjSQLTab li")[int].style.top;

**var** x = parseInt(left.split('px')[0] / 102);

**var** y = parseInt(top.split('px')[0] / 38);

**var** num = (x % 8 + 1) + (y \* 8);

// 隐藏input之前value

**var** temp = $("#tjSQLTab li")[int].children[0].children[1].value;

// 隐藏input对应li的num

**var** liNameNum = ($($("input[value="+temp+"]").parent().children()[0]).html()+"").replace(/\s/g,"").substring(3);

// 隐藏input之后value

$("#tjSQLTab li")[int].children[0].children[1].value = num;

// 存sql的textarea

$("#tjSQLListTab textarea[id=tdwStatSql\_"+liNameNum+"]").attr("name","tdwStatSql\_" + num);

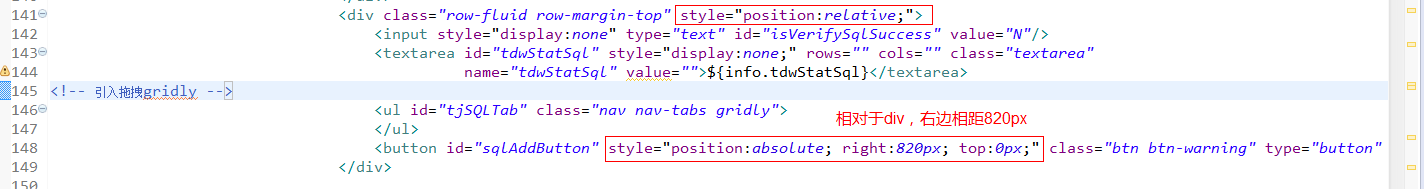
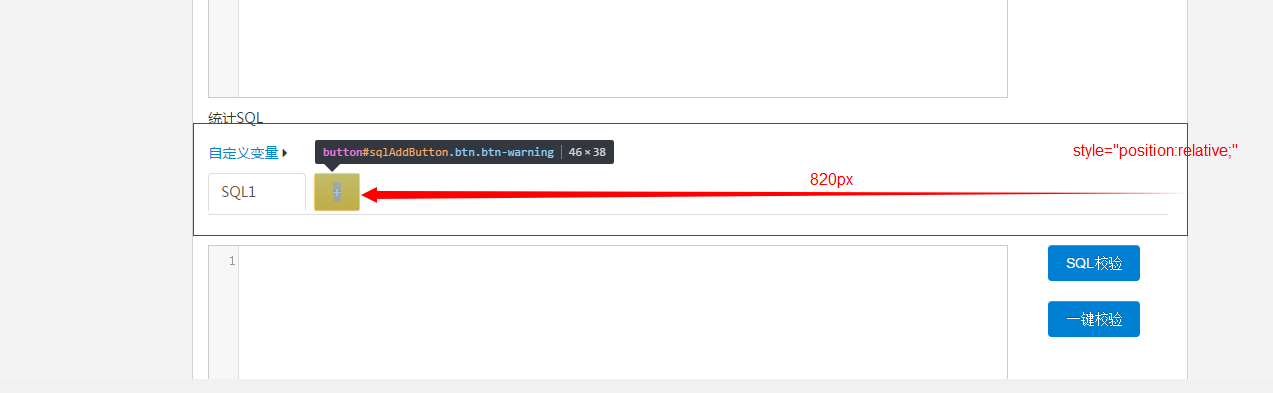
// 单个校验的按钮

$("#tjSQLListTab button[id=verifySqlId\_"+liNameNum+"]").attr("name","verifySqlId\_" + num);

}

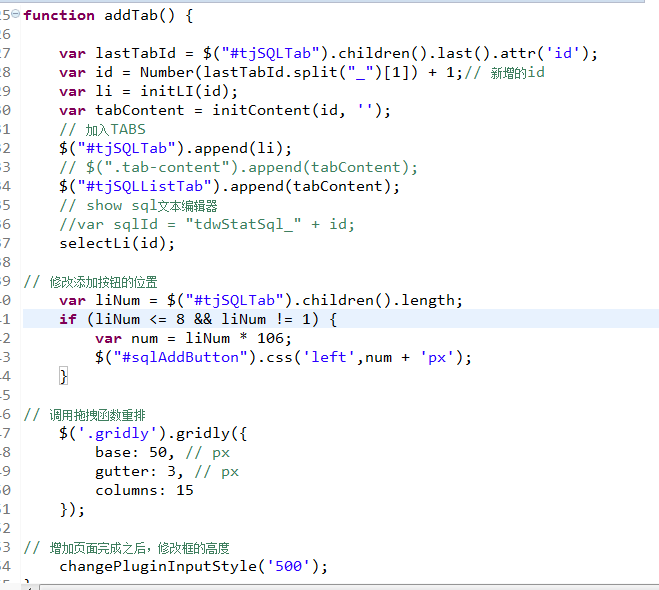
}

# 相对、绝对位置



外部relative、内部absolute，内部会悬浮在外部的上面，然后调right、top

## （1）增加：



## （2）删除：



# 存取值

## Select在td中，循环获得不同select的option

select.options[select.selectedIndex].value表示被选中option的value。

<http://www.cnblogs.com/itdream/archive/2012/05/31/2528345.html>

**var** tdw\_field = '';

**for** (**var** int = 0; int < $("[name=tdwtype]").length; int++) {

// $("[name=tdwtype]")为select数组。

**var** select = $("[name=tdwtype]")[int];

tdw\_field += select.options[select.selectedIndex].value + ",";

}

tdw\_field = tdw\_field.substring(0,tdw\_field.length-1);

shell\_task['tdw\_field'] = tdw\_field;

## 先new Array()数组储存在判断

**var** hostNames = $("[name=hostName]").val().split(";");

**var** span = **new** Array();

**if** ($(".hosttable span").length == 1) {

span[0] = $(".hosttable span").html();

} **else** {

**for** (**var** int = 0; int < $(".hosttable span").length; int++) {

span[int] = $(".hosttable span")[int].innerText;

}

}

**for** (**var** int = 0; int < hostNames.length; int++) {

**if** ($.inArray(hostNames[int], span) == -1) {

$.ligerDialog.warn("hosts与已有配置不符");

**return**;

}

}

**if**($("[name=hostName]").val().replace(/\s/g, "") == "" || $("[name=hostName]").val().replace(/\s/g, "") == "请选择"){

$.ligerDialog.warn("hosts配置为空");

**return**;

}

## （3）指定的option设为选中

**if** (**undefined** != **typeof** (mysql\_task['dbTableName']) && "undefined" != **typeof** (mysql\_task['dbTableName']) && "" != mysql\_task['dbTableName'] && **null** != mysql\_task['dbTableName']) {

**var** targetServers = $("[name=targetServer] option");

**for**(**var** i = 0; i< targetServers.length; i++){

**if**( $(targetServers[i]).data("dbname") == mysql\_task['dbTableName'].split(".")[0]){

targetServers[i].selected = **true**;

}

}

}

# Append标签添加字符串

## （1）传统table

**for** (**var** int = 0; int < fields.length; int++) {

**var** field = fields[int].substring(fields[int].indexOf("(")+1,fields[int].indexOf(","));

tips += '<tr class="filedTr'+(int+1)+'"><td style="width:14%"><input style="width:90%" name="tdwField" value="'+field+'"></td>'

+'<td style="width:4%;"><a style="margin-left: 15px;" class="mark" href="javascript:void(0)" onclick="mark(event)">标记</a></td>'

+'<td style="width:10%;"><span style="vertical-align: middle"><select style="width:80%;margin-top:5px;margin-bottom:5px;"><option>string</option><option>bigint</option></select></span></td>'

+'<td style="width:14%"></td>'

+'<td style="width:10%"></td>'

+'<td style="width:22%"><input name="logic" style="width:95%" onMouseOver="this.title=this.value" value='+fields[int]+'></td>'

+'<td style="width:10%"><input style="width:98%" value=""></td>'

+'<td style="width:17%">&nbsp;<a onclick="deleteTr(event)" href="javascript:void(0)">删除</a>&nbsp;&nbsp;<a onclick="upTr(event)" href="javascript:void(0)">上移</a>&nbsp;&nbsp;<a onclick="downTr(event)" href="javascript:void(0)">下移</a>&nbsp;&nbsp;<a onclick="addTr(event)" href="javascript:void(0)">插入</a></td></tr>';

}

$(".fileds").append(tips);

## table中tbody的append

<table class=*"table table-bordered"*>

<thead>

<tr>

<th style="width:*35%*">源字段</th>

<th style="width:*35%*">入库字段</th>

<th style="width:*30%*">操作</th>

</tr>

</thead>

<tbody id=*"hostConfig"*>

</tbody>

</table>

## （3）页面显示特殊字符如${}

解决方法：写到js中

// 初始化分区格式

$("[name=partitionType] option").remove();

**var** tips = '<option value="">NULL</option>'

+'<option value="P\_${YYYYMM}">P\_${YYYYMM}</option>'

+'<option value="P\_${YYYYMMDD}">P\_${YYYYMMDD}</option>'

+'<option value="P\_${YYYYMMDDHH}">P\_${YYYYMMDDHH}</option>';

$("[name=partitionType]").append(tips);

# 富文本编辑器codemirror

## （1）下载插件

<https://github.com/codemirror/codemirror>

## （2）引用方式

（1）引入方式：

<http://www.cnblogs.com/onlyonely/p/4450029.html>

1，引入核心文件（具体位置要根据项目中codemirror的位置决定）

**<link rel="stylesheet" href="lib/codemirror.css">**

**<script src="lib/codemirror.js"></script>**

codemirror.css和codemirror.js文件是CodeMirror插件的核心文件，无论你要高亮的代码是何种语言，都要引入这两个文件。

2，引入代码高亮js

接下来要引入的文件可以根据您的项目需求引入，不必要引入所有文件，比如您要显示html代码，则引入：

**<script src="xml-hint.js"></script>**

代码调用

<script>

myTextarea = document.getElementById("code");

  var editor = CodeMirror.fromTextArea(myTextarea, {

    mode: "text/html"

  });

</script>

## （3）例子

### （1）引入核心

1，核心

<link type=*"text/css"* rel=*"stylesheet"* href=*"pages/codemirror-5.8/lib/codemirror.css"*/>

<script type=*"text/javascript"* src=*"pages/codemirror-5.8/lib/codemirror.js"*></script>

### （2）引入某些特定参数需要的css、js

明显需要自己手动导入的：与lib核心同级的那些路径

1，mode模式：sql、html、java、c++，以sql为例

*<http://codemirror.net/mode/>*

*<http://codemirror.net/mode/sql/index.html>*

目前支持以下sql

[text/x-sql](http://codemirror.net/mode/sql/index.html?mime=text/x-sql), [text/x-mysql](http://codemirror.net/mode/sql/index.html?mime=text/x-mysql), [text/x-mariadb](http://codemirror.net/mode/sql/index.html?mime=text/x-mariadb), [text/x-cassandra](http://codemirror.net/mode/sql/index.html?mime=text/x-cassandra), [text/x-plsql](http://codemirror.net/mode/sql/index.html?mime=text/x-plsql), [text/x-mssql](http://codemirror.net/mode/sql/index.html?mime=text/x-mssql), [text/x-hive](http://codemirror.net/mode/sql/index.html?mime=text/x-hive), [text/x-pgsql](http://codemirror.net/mode/sql/index.html?mime=text/x-pgsql), [text/x-gql](http://codemirror.net/mode/sql/index.html?mime=text/x-gql), [text/x-gpsql](http://codemirror.net/mode/sql/index.html?mime=text/x-gpsql). [text/x-esper](http://codemirror.net/mode/sql/index.html?mime=text/x-esper).

<!-- 基本sql，**必须加** -->

<script type=*"text/javascript"* src=*"pages/codemirror-5.8/mode/sql/sql.js"*></script>

<!-- 详细sql，属于那种特定 -->

<script type=*"text/javascript"* src=*"pages/codemirror-5.8/addon/hint/sql-hint.js"*></script>

2，theme主题： 黑夜、白天

<http://codemirror.net/demo/theme.html>

<link rel=*"stylesheet"* href=*"pages/codemirror-5.8/theme/monokai.css"*>

### （3）使用富文本编辑器

<https://zhidao.baidu.com/question/530030662.html>

（1）设置参数

**var** editor = CodeMirror.fromTextArea(document.getElementById(sqlId), {

**// 这两项需要引入特定的js、css**

mode: {name: "text/x-hive"}, // 上面引入的特定sql

theme: "monokai", // 上面引入的主题，默认可以设置default

styleActiveLine: **true**, // 是否高亮

lineNumbers: **true**, // 是否加行号

//lineWrapping:true, // 当一行文字太长，是换行还是横滚动条

extraKeys: {"Ctrl": "autocomplete"}, // 输入提示键

//content:show tables, // 如果是textarea这种标签，默认的value

smartIndent: **true**, // 缩进：换行后，自动与上一行保持

indentWithTabs: **true**, // 缩进时，将空格换位tab

fixedGutter: **true**, // 滚动条上下true、水平false

autofocus: **true** // 初始化时自动获取焦点

});

#### （2）存在多个editor

（1）储存editor

只需要定义全局的数组，新定义了一个editor就添加进去，这样只需要初始化一次，就可以在所有function中使用。否则每次初始化，都会自动聚焦该editor。

**var** codeMirrorInstances={};

**var** editor1 = CodeMirror.fromTextArea(document.getElementById(sqlId1), {});

**var** editor2 = CodeMirror.fromTextArea(document.getElementById(sqlId2), {});

**var** editor3 = CodeMirror.fromTextArea(document.getElementById(sqlId3), {});

codeMirrorInstances[sqlId1] = editor1;

codeMirrorInstances[sqlId2] = editor2;

codeMirrorInstances[sqlId3] = editor3;

（2）给多个editor赋值

可能会出现不能编辑的情况，多个editor会冲突，这时候需要异步刷新

for(){

**var** editor1 = CodeMirror.fromTextArea(document.getElementById(sqlId1), {});

setTimeout(**function**(){

editor1 .refresh();

}, 50);

}

（3）取值、设置值

**var** filterSQL = editor.getValue();

editor.setValue(tdw\_conf['filterSQL']);

（4）跳转、获取某一行的值

<https://my.oschina.net/u/2391658/blog/713129>

http://blog.csdn.net/qq\_17365131/article/details/75090289

（1）获取总行数

lineCount()

（2）获取某一行的内容

getLine(i)

（3）跳转

scrollIntoView({line: 行数, ch:列数});

假设返回了红、白名单，包括红名单的行要被标记为红色，包括白名单的行要取消红色

**var** redList = message.data.redList;

**var** whiteList = message.data.whiteList;

**if**(redList != **null** && **undefined** != **typeof**(redList) && "undefined" != **typeof** (redList)){

**var** createTblSqlCodeMirror = CodeMirror.fromTextArea(document.getElementById("createTblSql"), {

mode: {name: "text/x-hive"},

theme: "monokai",

styleActiveLine: **true**,

lineNumbers: **true**,

extraKeys: {"Ctrl": "autocomplete"},

smartIndent: **true**,

indentWithTabs: **true**,

fixedGutter: **true**,

autofocus: **true**

});

// 获取总行数

**var** lineCount = createTblSqlCodeMirror.lineCount();

**var** theLine = **null**;

**if**(whiteList != **null** && **undefined** != **typeof**(whiteList) && "undefined" != **typeof** (whiteList)){

**for** (**var** i = 0; i < lineCount; i++) {

// 根据行号获取某一行的内容（codemirror是从第0行才是的，第0行就是第1行）

**var** line = (createTblSqlCodeMirror.getLine(i)+"").toLowerCase();

**for** (**var** j = 0; j < redList.length; j++) {

**if** (line.indexOf(redList[j]) != -1) {

**var** whiteFlag = **true**;

**for** (**var** k = 0; k < whiteList.length; k++) {

**if** (line.indexOf(whiteList[k]) != -1) {

whiteFlag = **false**;

}

}

**if**(whiteFlag == **true**){

theLine = i;

**break**;

}

}

}

**if**(theLine != **null**){

**break**;

}

}

}**else**{

**for** (**var** i = 0; i < lineCount; i++) {

**var** line = (createTblSqlCodeMirror.getLine(i)+"").toLowerCase();

**for** (**var** j = 0; j < redList.length; j++) {

**if** (line.indexOf(redList[j]) != -1) {

theLine = i;

**break**;

}

}

**if**(theLine != **null**){

**break**;

}

}

}

**if**(theLine != **null**){

// 跳转到指定行数（也是从第0行开始的）

createTblSqlCodeMirror.scrollIntoView({line: theLine, ch:0});

}

// 跳转后才能获取富文本编辑器的pre中的span，因为pre会随着滚动条变化而变化

**var** pre = $("#createTblSql").parent().find(".CodeMirror-code").find("pre").children();

**for** (**var** i = 0; i < pre.length; i++) {

**for** (**var** j = 0; j < redList.length; j++) {

**var** line = ($(pre[i]).html()+"").toLowerCase().replace(/<span[^>]+>/g,'').replace(/<\/span>/g,'');

**if** (line.indexOf(redList[j]) != -1) {

$(pre[i]).css("background-color","red");

**break**;

}

}

}

**if**(whiteList != **null** && **undefined** != **typeof**(whiteList) && "undefined" != **typeof** (whiteList)){

**for** (**var** i = 0; i < pre.length; i++) {

**for** (**var** j = 0; j < whiteList.length; j++) {

**var** line = ($(pre[i]).html()+"").toLowerCase().replace(/<span[^>]+>/g,'').replace(/<\/span>/g,'');

**if** (line.indexOf(whiteList[j]) != -1) {

$(pre[i]).css("background-color","");

**break**;

}

}

}

}

}

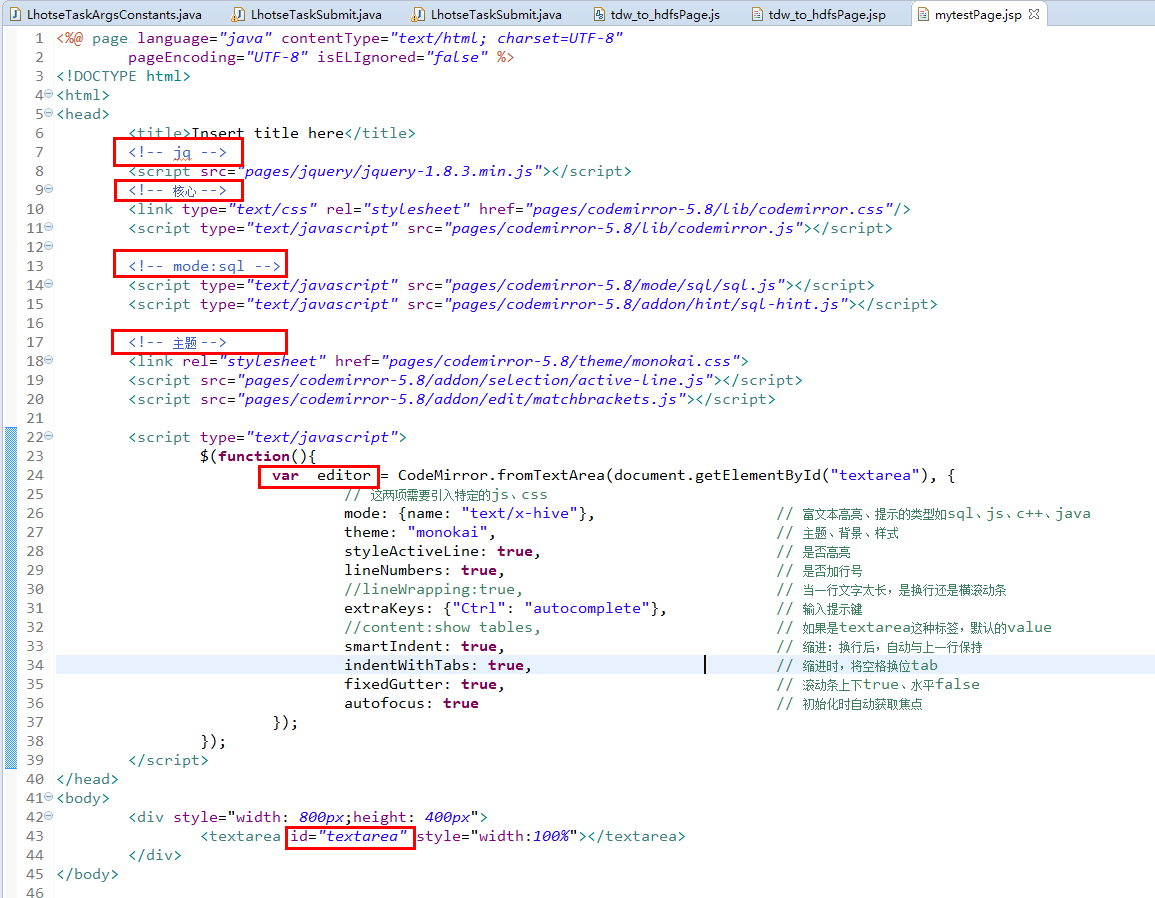
效果：



（4）其余设置参数：

<http://www.hyjiacan.com/codemirror-config/>

### （4）效果



## 多个编辑器赋值问题

# Es6

### Includes用法

1. ES6提供了Array.includes()函数判断是否包含某一元素，除了不能定位外，解决了indexOf的上述的两个问题。它直接返回true或者false表示是否包含元素，对NaN一样能有有效。
2. [?](https://www.jb51.net/article/124164.htm#)

|  |  |
| --- | --- |
| 1. 1 2. 2 3. 3 4. 4 | 1. const arr1 = ['a','b','c','d','e','f','g','h','i','j','k', NaN] 2. console.log('%s', arr1.includes('c')) 3. console.log('%s', arr1.includes('z')) 4. console.log('%s', arr1.includes(NaN)) |