**LOYAL-前端页面开发手册**

[1 引言 2](#_Toc496184839)

[2 前端画面设计 2](#_Toc496184840)

[2.1 Bootstrap 2](#_Toc496184841)

[3 自定义控件 3](#_Toc496184842)

[3.1 日历控件 3](#_Toc496184843)

[3.2 列表（Flexigrid)控件 4](#_Toc496184844)

[3.2.1 列表页面设计 4](#_Toc496184845)

[3.2.2 列表脚本初始化 5](#_Toc496184846)

[3.2.3 列表脚本封装 8](#_Toc496184847)

[3.2.4 列表时间处理 8](#_Toc496184848)

[3.3 数据组控件 9](#_Toc496184849)

[3.4 树控件 10](#_Toc496184850)

[3.5搜索框控件 12](#_Toc496184851)

[4 脚本封装 14](#_Toc496184852)

[4.1 isEmpty 14](#_Toc496184853)

[4.2strToJson 14](#_Toc496184854)

[4.3 nest 14](#_Toc496184855)

[4.4 openWindow 15](#_Toc496184856)

[4.5 getWidget 15](#_Toc496184857)

[4.6 value 15](#_Toc496184858)

[4.7 assign 16](#_Toc496184859)

[4.8 setSelect 16](#_Toc496184860)

[4.9 validateForm 16](#_Toc496184861)

[4.10 valid 16](#_Toc496184862)

[4.11 bAlert 18](#_Toc496184863)

[4.12 setReadOnly 18](#_Toc496184864)

[4.13 setSelectByList 18](#_Toc496184865)

[4.14 datetimepicker 19](#_Toc496184866)

[4.15 tableclone 19](#_Toc496184867)

[4.16 tableDateArray 19](#_Toc496184868)

[4.17isJson 19](#_Toc496184869)

[4.18 hide 19](#_Toc496184870)

[4.19 show 20](#_Toc496184871)

[4.20tableToExcel 20](#_Toc496184872)

[4.21 searchBox 20](#_Toc496184873)

[4.22inputSearchInit 20](#_Toc496184874)

[4.23 treeData 20](#_Toc496184875)

[4.24 getCookie 20](#_Toc496184876)

[4.25 setCookie 21](#_Toc496184877)

[4.26 formatDate 21](#_Toc496184878)

[4.27 getDateFormat 21](#_Toc496184879)

[4.28 getLocale 21](#_Toc496184880)

[4.29 uploadFile 21](#_Toc496184881)

# 1 引言

《LOYAL-前端页面开发手册》是针对LOYAL-前端平台制定的详细开发手册，它包括前端页面（布局、控件、样式等）和事件（脚本、交互方式等）的开发说明和开发规范，在此基础上应用开发人员能够定制出比较美观和实用的画面效果。

# 2 前端画面设计

## 2.1 Bootstrap

Bootstrap是一个简洁、直观的，用于快速开发Web应用程序和网站的前端开发框架。

Bootstrap为开发人员创建接口提供了一个简洁统一的解决方案，并且它包含了功能强大的内置组件，它还提供了基于Web的定制，最主要的是，Bootstrap是开源的。目前所有的主流浏览器都支持Bootstrap。

Bootstrap的安装是非常容易的，只需要在官网上下载Bootstrap的最新版本就可以使用了，一个常规的HTML文件变为使用了Bootstrap模板需要将**jquery.js**、**bootstrap.min.js**和**bootstrap.min.css**文件引用进去即可。

# 3 自定义控件

## 3.1 日历控件

需要日历控件时，需要使用<input type="text" datetype="date" name=" "format=”” local-date=" "afterToday=””beforeToday=””class=”datetime”>

（注：若是需要使用$.tableclone()方法绘制表格中的日历组件，需要将datetype设置为”none”）

方法：

$.datetimepicker()

控件属性：

name(必填):通过name获取日期控件value

datetype="date"(必填),class=”datetime”(必填)：指定此控件为日历控件；

format:日期格式；

设置了四种日历格式，分别为：

若参数为空或为1，则格式为默认，为”yyyy-MM-dd”；

若参数为2，则格式为”yyyy/MM/dd”；

若参数为3，则格式为”dd-MM-yyyy”；

若参数为4，则格式为”dd/MM/yyyy”

自定义格式：参数可以为为自定义的格式，如”mm/dd/yyHH:i:s”

具体格式如下：

| 符号 | 意义 |
| --- | --- |
| p | 12小时制且小写(‘am’ or ‘pm’) |
| P | 12小时制且大写(‘AM’ or ‘PM’) |
| s | 秒，前面不补0 |
| ss | 秒，前面补0 |
| i | 分，前面不补0 |
| ii | 分，前面补0 |
| h | 时，24小时制，前面不补0 |
| hh | 时，24小时制，前面补0 |
| H | 时，12小时制，前面不补0 |
| HH | 时，12小时制，前面补0 |
| d | 日，前面不补0 |
| dd | 日，前面补0 |
| m | 月，数字表示，前面不补0 如：4 |
| mm | 月，数字表示，前面补0 如：04 |
| M | 月，缩写表示，前面补0 如：Apr |
| MM | 月，全称表示，前面补0 如：April |
| yy | 年，后两位如：16 |
| yyyy | 年，全部如：2016 |

afterToday：今日之后的日期是否可选（'true'：不可选）

beforeToday:今日之前的日期是否可选（’true’：不可选）

local-date：指定是否默认在日期框中显示当前日期，若显示，则值为’true’

方法的具体使用：

如：<input type=*"text"*datetype=*"date"* name=*"aaa "*>

<input type="*text*" datetype=*"date"* name=*"aaa"*format=*”2”* >

<input type="*text*" datetype=*"date"*name=*"aaa"*format=*”yyyy-mm-dd”* local-date=*"true"* >

## 3.2 列表（Flexigrid)控件

### 3.2.1 列表页面设计

一个flexigrid控件就是一个表格控件，它最简单的定义方式如下：

<tableid=*"tableid"*style="display:*none*;">

<thead>

<tr>

<thname=*'id'*width=*'210'*style='display: *none*;'>id</th>

<thname=*'firstname'*width=*'210'*>firstname</th>

<thname=*'lastname'*width=*'210'*>lastname</th>

<thname=*'email'*width=*'210'*>email</th>

<thname=*'officeName'*width=*'210'*>officeName</th>

</tr>

</thead>

</table>

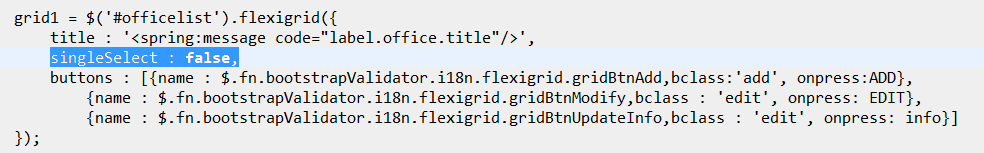
### 3.2.2 列表脚本初始化

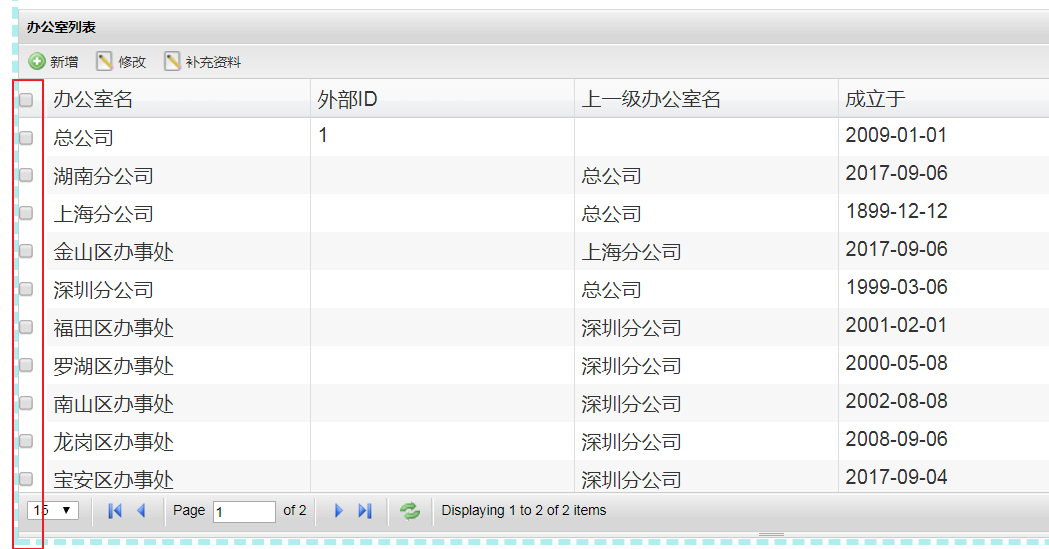
#### 3.2.2.1 普通列表初始化

说明：

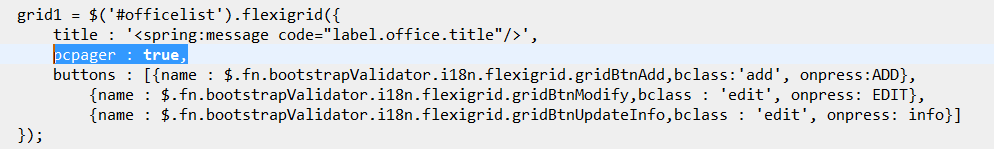
1) 默认列表的数据只有一页

2）如果列表需要设置为多选的，可以设置singleSelect : false



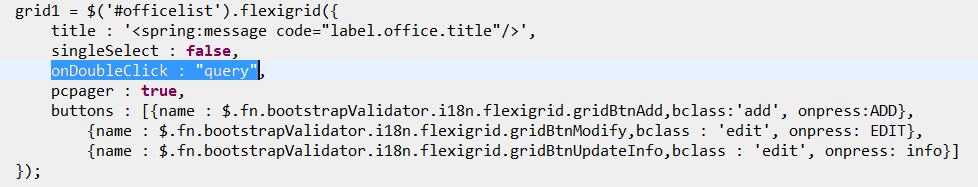


3）如果后台接口不支持分页，但列表记录又很多，可以使用PC端进行分页，可设置pcpager : true



4) 双击列表记录触发事件，可设置onDoubleClick : "query",

query为在本地页面中编写的一个funtion方法，跟列表头按钮上触发的事件意思相同。



<script type="text/javascript">

var grid1;

$(document).ready( function() {

grid1 = $('#*tableid*').flexigrid({

title : '用户列表',

buttons: [{name : $.fn.bootstrapValidator.i18n.flexigrid.gridBtnAdd,bclass:'add', onpress:ADD},{name : $.fn.bootstrapValidator.i18n.flexigrid.gridBtnModify,bclass : 'edit', onpress: EDIT},{name : $.fn.bootstrapValidator.i18n.flexigrid.gridBtnDelete,bclass : 'delete', onpress: DELETE}]

});

});

function add(){

......

}

function edit(){

......

}

function delete(){

......

}

</script>

#### 3.2.2.2 多页列表初始化

说明：列表的记录达到一定数据量时需要自动分页（根据API接口设定是否使用分页）,分页列表与普通列表写法不同的将以红色字体高亮；nestjson这个json是从初始嵌套方法$.nest中的json独立出来写的。

<script type="text/javascript">

var grid1;

varnestjson;

$(document).ready( function() {

nestjson = {

url : "<c:url value='/fineract.do'/>",

type : "post",

async : true,

data : {

"dataType" : "GET",

"mydata" : "",

"url" : "api.v1.clients",

"paging" : "Y"

},

callback : "clientlist"

};

grid1 = $('#*tableid*').flexigrid({

title : '用户列表',

nestjson :nestjson,

buttons: [{name : $.fn.bootstrapValidator.i18n.flexigrid.gridBtnAdd,bclass:'add', onpress:ADD},{name : $.fn.bootstrapValidator.i18n.flexigrid.gridBtnModify,bclass : 'edit', onpress: EDIT},{name : $.fn.bootstrapValidator.i18n.flexigrid.gridBtnDelete,bclass : 'delete', onpress: DELETE}]

});

});

function add(){

......

}

function edit(){

......

}

function delete(){

......

}

functionclientlist(param){

grid1.addRows(param.pageItems);

grid1.setPage(param.page, param.pages);

}

</script>

注意：在列表初始化时设置了buttons时，onpress为该按钮的click事件，必须在<script>内$(document).ready( function(){});外有对应的方法，否则会报脚本错误。

### 3.2.3 列表脚本封装

1) 获取列表选中行的数据

**var**td\_val = grid1.colVal('id');

**var**td\_first\_val = grid1.colVal('firstname');

**var**td\_last\_val = grid1.colVal('lastname');

**var**com\_td = td\_first\_val + td\_last\_val;

2) 动态给列表赋值

grid1.addRows(param);

//param为数组，数组的项为一个JSON。

3）动态给列表赋当前页数和总页数

grid1.setPage(page, pages);

//page为当前页，pages为总页数。

4）获取多选记录中列的数据（返回一列的数组）

var rows = grid1.getSelectRows();//获取选中的行

varcolValues = grid1.getColValues(rows,"name");//获取选中行的name列的数据，返回一个数组

### 3.2.4 列表时间处理

在<th>标签中添加type属性：

type=”datetime”返回yyyy-MM-ddhh:mm:ss格式的日期

type=”date”返回yyyy-MM-dd格式的日期

<tableclass=*"table table-hover display"*id=*"auditList"*>

<thead>

<thname=*"madeOnDate"*width=*'154'*type=*"datetime"*>datetime</th>

<thname=*"madeOnDate"*width=*'154'*type=*"date"*>datetime</th>

</thead>

<tbody></tbody>

</table>

### 3.2.5 列表列button及自定义js

在<th>标签中添加type属性：type=*"action"，actionjs*中写要添加的js代码，button=" true"会使控件为该列添加一个无样式默认按钮，按钮value等于该列的值，并且会执行actionjs中的代码。row是当前数组行，可通过row获取当前行数据。

注意：在使用row时应使用'abc('+row.id+')'形式，不能使用'abc(row.id)'，该型式是把row当成button的一个字符串变量处理了。

button是列表的button对象，可直接访问。button.addClass('btn btn-primary btn-offset');

Actionjs="", button=" true"都是type="action"才有效的。

td为当前单元格，默认内容为空，可直接使用：$(td).append('<input type="button" value="'+tdvalue+'">');

<table id=*"centerlist"*>

<thead>

<tr>

<th name=*"id"* style="display: *none*;">id</th>

<th name=*"status.value"* search=*"true"* type=*"action"* button=" *true* " actionjs=*"if(row.status.value=='Active'){button.addClass('btn btn-primary btn-offset');} button.attr('onClick','abc('+row.id+')');"*>Action</th>

</tr>

</thead>

<tbody></tbody>

</table>

## 3.3 数据组控件

1）点击添加按钮，在表格中自动生成新的行：

需要在第一行在<tr>设置属性display:none;若组件为日历组件，需要在<input>中添加*datetype=”none”，而不是datetype="date"；*

方法：

$.tableclone(tableId)

tableId：表格id

2）若需要获取其中的数据，则需要使用方法：

$. tableDateArray(tableId)

tableId：表格id

通过方法获取到的是一个数组，数据是每一行的（name：value）组成的json对象。

此方法支持的表格中的组件有input框、checkbox复选框以及select下拉菜单。

注：1）要获取表格数据，必须在<th>标签内加name属性，值与其子节点中的name相同

2）若某项数据不需要获取，需要在<th>标签内加入date-worthless=*"true"。*

3）不推荐在表头添加<thead>标签。

<button class="btnbtn-primary" id="addRow">添加</button>

<tableclass=*"table table-hover "*id=*"table"*style="width: *80%*">

<tr>

<th>日期</th>

<th>输入框</th>

<th>下拉框</th>

<th>复选框</th>

<th>操作</th>

</tr>

<trstyle="display:none">

<thname=*"date"*><inputtype=*"text"*datetype=*"none"*name=*"date"*local-date=*"true"*class=*"datetime"*></th>

<thname=*"input"*><inputname=*"input"*></th>

<thname=*"select"*><selectname=*"select"*>

<optionvalue=*"a"*>aaaaa</option>

<optionvalue=*"b"*>bbbbb</option>

<optionvalue=*"c"*>ccccc</option>

</select></th>

<thname=*"checkbox"*><inputtype=*"checkbox"*value=*"5"*name=*"checkbox"*></th>

<thdate-worthless=*"true"*><aclass=*"remove"*><spanclass=*"glyphiconglyphicon-remove"*aria-hidden=*"true"*></span></a></th>

</tr>

## 3.4 树控件

树的展示功能：

1.引用了bootstrap-treeview.js和bootstrap-treeview.css文件

2.该组件的如果使用json形式的数据，需要采用如下形式：

（需要包含text和nodes节点，如果是叶子节点可不包含）

**var**treedate = [{

id : 1,

text :'Head Office',

nodes :[{

text : '深圳分公司',

nodes : [{

text : '办事处'

}]

},{

text : '广州分公司',

nodes : [{

text : '办事处'

}]

}]

}];

3.树的初始化

<divclass=*"col-sm-8"*id=*"applicableOffices"*></div>

$('#applicableOffices').treeview({

data : treedate, //这里使用上述json格式的数据

color: "#428bca", //以下是相关初始化设置

showBorder: **false**,

showIcon :**false**,

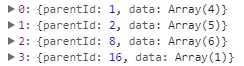
showCheckbox: **true**});

由于后台接口返回来的json数据并不具备有明显上下级关系，因此自身需要构造上述带有key为text和nodes的json格式的数据。

可提供思路如下：

1. 将所有元素进行分组

以办公室为例：按照具有相同的parentId进行分组



**function**getTreeData(param){

/\* 将param参数按照parentId对数据进行升序处理 \*/

**var**jsonarr = param.sort(**function**(x,y){

**return** (x.parentId>y.parentId)? 1 : -1;

});

//定义一个集合

**var** options = {};

//定义一个数组

**var**dest = [];

/\* 将具有同一父Id的元素进行分组 \*/

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

**if**($.type(param[i].parentId) === 'undefined'){

}**else**{

**var** keep = param[i];

**if**(!options[keep.parentId]){

dest.push({

parentId :keep.parentId,

data : [keep]

});

options[keep.parentId] = keep;

}**else**{

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

**var**tmp = dest[j];

**if**(tmp.parentId == keep.parentId){

tmp.data.push(keep);

**break**;

}

}

}

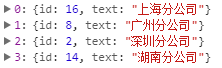
}

}

**return**dest;

}

2.获取根节点的第一层子节点，并构造节点



**var**dest = getTreeData(param);

**var**arr = [];

$.each(dest,**function**(index){ //遍历分组后的数组

**var** di = dest[index];

**if**(di.parentId == 1){

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

arr.push({

id :di.data[i].id,

text :di.data[i].name

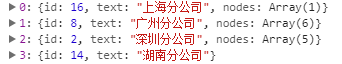
});

}

}

});

3.根据第一层节点的子节点，并构造节点下的节点，构造完成



**var**arr\_copy = arr;

$.each(arr,**function**(index){

**var**ai = arr[index];

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

**var** key = dest[i].parentId;

**if**(key != 1 && key == ai.id ){

ai['nodes'] = [];

**for**(**var** j=0;j<dest[i].data.length;j++){

**var** did = dest[i].data;

arr\_copy[index].nodes.push({

id : did[j].id,

text : did[j].name

});

}

}

}

});

4.最终的json数据

**var**treedate = [{

id : 1,

text :'Head Office',

nodes :arr\_copy

}];

## 3.5搜索框控件

搜索框功能：

搜索框格式为<input datatype="*search*" api=""name=””textname=””>

其中，datatype="*search*"确定此输入框为搜索框；name为必填属性；textname为搜索框要显示的内容的属性，即输入框对应值的属性；api为搜索框后台的api，即搜索的范围（需要自己查找api），其中输入框的值所对应的属性要放在最后！

（注：

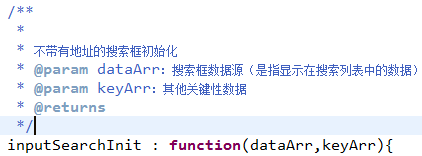
若api中有多个变量，需自己手动在js中通过attr或prop修改api的值，不可直接在标签内使用字符串拼接！）

例：<inputid=*"search"*name=*"search"*datatype=*"search"*api=*"api.v1.clients?officeId=1&orderBy=displayName&orphansOnly=true&sortOrder=ASC&displayName="*textname=”*displayName*”>

（注：此控件暂不支持Firefox浏览器！）

（2）不带有api地址的搜索框

A.调用初始化方法形式： $.inputSearchInit(dataArr,keyArr);



注意事项：

1. 该方法中的两个参数都是数组。

2. 此方法中的第一个参数（dataArr）必须已经被定义，且必须是是api接口中返回的JSON数据。

3. 此方法中的第二个参数（keyArr）表示dataArr数组中json对象的其它关键性key。

B. 搜索框控件格式为<input datatype="*match*" for=”search” source=”name” id=”” autocomplete=”off”/>其中datatype\for\source为必须属性。

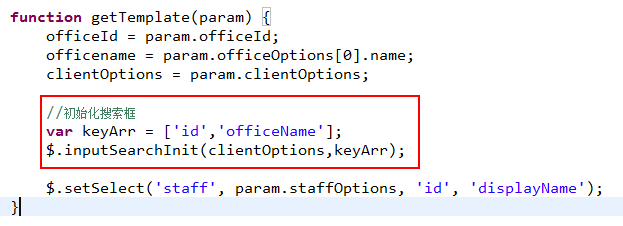
其中：

1. datatype="*match*"确定此输入框为搜索框；

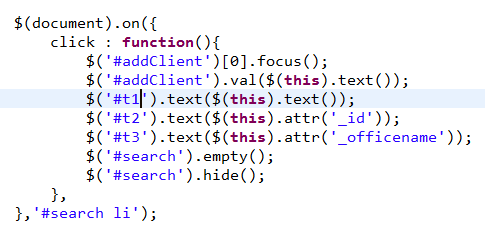
2. for=”search”表示搜索列表的id；

3. source=”displayName” --dataArr[index].displayName：确定搜索列表中需要显示的数据的key。

例子:



初始化之后，需要自定义click事件对功能做出相关处理。比如：



# 4 脚本封装

## 4.1 isEmpty

$.isEmpty(str);

说明：判断字符串是否为空；str为null或空时返回true;否则返回false;

## 4.2strToJson

$.strToJson(strjson);

说明：将JSON格式字符串转为JSON对象。

## 4.3 nest

$.nest({

url :"<c:url value='/fineract.do'/>",

type :"post",

async :**true**,

data : {

"dataType" : "GET",//接口要求的提交类别（type)

"mydata" : "",//接口要求为post提交时所要提交的表单数据

"url" : "api.v1.users"//fineract接口地址

},

callback : "callback"//嵌套成功后回调的方法

});

注意：function callback(parma){};回调方法必须在<script>内$(document).ready( function(){});外有对应的方法，否则会报脚本错误。

说明：嵌套（ajax)封装；url为请求地址； data为请求时发送的参数； callback:请求完成，服务器数据返回时的回调方法。

## 4.4 openWindow

$.openWindow(url);//默认使用横态框动态load一个JSP画面。

$.openWindow(url,nopenType);//nopenType不为空时打开一个JSP画面。

$.openWindow(url,nopenType,openWidth);//设置打开的窗口宽度

$.openWindow(url,nopenType,openWidth);//设置打开的窗口高度

说明：

1) 模态框载入一个jsp页面；url为jsp地址。

2) url为加载的JSP页面地址

3) nopenType为打开的模式，openType为空时默认为模态框方式，openType为非空时使用iframe的方式打开JSP画面

4) openWidth打开画面窗口的宽度

5) openHeigth打开画面窗口的高度

## 4.5 getWidget

$.getWidget(buttonid)

$.getWidget("office").change(function() {});

$.getWidget("office").click(function() {});

说明：根据控件名获取控件的对象，用于对控件操作时绑定事件

## 4.6 value

$.value(name);

说明：根据控件的名称获取控件的值

## 4.7 assign

$.assign(json);

说明：给控件赋值，根据json格式赋值，key为控件名称，value为值

## 4.8 setSelect

$.setSelect(fieldName,selectList,valueName,textName);

说明：

1）动态加载，根据json格式赋值

2）fieldName为所要动态加载的select控件名称。

3）selectList为动态加载的下拉列表项。

4）valueName为下拉项KEY值在JSON中的名称。

5）textName为下拉项value值在JSON中的名称。

## 4.9 validateForm

$.validateForm(formName);

说明：检查form中的校验是否通过，通过return true;不通过return false;

## 4.10 valid

$.valid(formid,validatorjson);

注意：在进行valid验证的时候，要验证的控件上一定要在<div class:"form-group">中,否则会报错。

说明：

在校验之前，准备进行验证的组件一定要在<div class:"form-group">中

校验的方法：在页面中的<script>标签中通过$.valid(formid,validatorjson)来进行校验，其中：

formid：要验证的组件所在表单的id；

validatorjson：包含了多个json的数组，格式为：

[{"name":"\*\*\*","validatorName":"\*\*\*"}]

name：要验证的组件的name属性值；

validatorName：要进行的验证类型名称。

如：

$.valid('form1',[{"name":"name1","notEmpty":"notnull"},{"name":"name2","notEmpty":"notnull"}]);

校验主要分为非空校验、邮箱校验、长度校验、日期校验、正则校验以及重复校验

非空校验：非空校验的类型名称为“notEmpty”，进行非空校验时，需要在参数中加入此类型名称作为key，value为提示信息，如：

$.valid('form1',[{"name":"name1","notEmpty":"此选项不能为空！"}]);

若不输入提示信息，默认提示信息为：请填写必填项目。

邮箱校验：邮箱校验的类型名称为“email”，进行邮箱校验时，需要在参数中加入此类型名称作为key，value为提示信息，如：

$.valid('form1',[{"name":"name1","email":"请输入正确的邮箱格式！"}]);

若不输入提示信息，默认提示信息为：请输入有效的邮件地址。

长度校验：长度校验的类型名称为“lengthMax”和“lengthMin”，进行长度校验时，需要在参数中加入其中一到两个类型名称作为key，value为具体的长度，如：

$.valid('form1',[{"name":"name1","lengthMax":"4","lengthMin":"2" }]);

日期校验：邮箱校验的类型名称为“dataFormat”，进行日期校验时，需要在参数中加入此类型名称作为key，value为日期格式，如：

$.valid('form1',[{"name":name1","dateFormat":"YYYY/MM/DD"}]);

正则校验：邮箱校验的类型名称为“regexp”和“regexpMsg”，进行正则校验时，需要在参数中加入“regexp”作为key，value为正则格式，如：

$.valid('form1',[{"name":"name1","regexp":"/^[a-zA-Z0-9\_\.]+$/","regexpMsg":"只能是数字、字母、下划线、\和."}]);

当参数中没有“regexpMsg”时，错误消息为默认的消息：“请输入符合正则表达式限制的值”；

重复校验：邮箱校验的类型名称为“identicalTo”，进行重复校验时，需要在参数中加入此类型名称作为key，value为要进行比较的组件的name属性值，如：

$.valid('form1',[{"name":name1","identicalTo":"name2"}]);

## 4.11 bAlert

$.bAlert(str);

说明：设置提示信息，相当于JS中的alert的封装；str为提示的内容

## 4.12 setReadOnly

$.setReadOnly(formName);

说明：设置form控件只读；formName：表单id值

## 4.13 setSelectByList

$.setSelectByList(fieldname,selectList);

说明：

1）动态加载，根据List格式赋值

2）fieldName为所要动态加载的select控件名称。

3）selectList为动态加载的下拉列表项。

## 4.14 datetimepicker

$.datetimepicker();

说明：

1. 日历控件，在需要日历时使用
2. 页面中必须要有<input type="text" datetype="date" name=" "format=”” local-date=" "afterToday=””beforeToday=””class=”datetime”>组件才能生效

详细的使用方法请参考本文档3.1 日期控件使用说明。

## 4.15 tableclone

$.tableclone(tableId);

说明：

1. 动态添加表格的一行
2. tableId为需要添加表格的id
3. 需要添加的表格必须有一行隐藏行

详细使用方法请参考本文档3.3 数组控件使用说明。

## 4.16 tableDateArray

$.tableDateArray(tableId);

说明：

1. 获取动态表格中的数据
2. tableId为表格的id
3. 获得的为每一行数据的(name:value)的json对象

详细使用方法请参考本文档3.3 数组控件使用说明。

## 4.17isJson

$.isJson(obj);

说明：

1. 判断一个字符串是否为JSON对象
2. 如果包含返回true,否则返回false

## 4.18 hide

$.hide(obj);

说明：

1. obj可以是单个的参数，也可以是多个参数组成如"aa","bb","cc"
2. 隐藏控件或DEV等

## 4.19 show

$.show(obj);

说明：

1. obj可以是单个的参数，也可以是多个参数组成如"aa","bb","cc"
2. 显示控件或DEV等

## 4.20tableToExcel

$.tableToExcel(tableid);

说明：

1. tableid是要导出的表格的id
2. 将表格生成Excel文件下载

## 4.21 searchBox

$.searchBox()

说明：搜索框组件，默认再header.jsp中，按照格式使用<input>标签即可。

可以参考与文档3.5 搜索框控件的介绍。

## 4.22inputSearchInit

$.inputSearchInit(dataArr,keyArr)

说明：dataArr:是指搜索列表中的数据源，一般是api接口返回的JSON数据中的数组。

keyArr:是指数组中Json对象的关键性数据的key。

## 4.23 treeData

$.treeData(param)

说明：

参数：param 回调数组,必须有id和name属性，子节点带有parentId属性

返回：返回树数据 [{"id":"1","name":"a",nodes:[{"id":"2","pid":"1","name":"b"}]}]格式。

## 4.24 getCookie

$.getCookie (name)

说明：获取客户端的Cookie值

参数：name为Cookie变量的名称

本系统国际化Cookie的变量名称为clientLanguage; 日期格式Cookie变量名称为dateFormat

## 4.25 setCookie

$.setCookie (name,value)

说明：获取客户端的Cookie值

参数：name为Cookie变量的名称，value为Cookie的值

本系统国际化Cookie的变量名称为clientLanguage; 日期格式Cookie变量名称为

## 4.26 formatDate

$.formatDate (value); //使用客户端默认的日期格转换日期

$.formatDate (value,format); //根据format的参数类型格式转换日期

$.formatDate (value,format,time) //转换日期带时间格式，fomat为空则使用默认的日期格式

说明：日期与日期时间格式化

参数：value为需要转换的日期，format为日期格式的类型，time为时间格式

返回：日期或时间

## 4.27 getDateFormat

说明：获取当前日期格式

返回：日期格式，例如：yyyy-mm-dd

## 4.28 getLocale

$.getLocale();

说明：获取当前语言的值

返回：中文返回zh\_CN ，英文返回en

## 4.29 uploadFile

$.uploadFile(formId,url,callback);

说明：上传文件方法

参数：formId：上传文件Form表单Id

（form格式如下：uploadForm）

<form id="uploadForm">  
 <input type="text" name="name" />  
 <input type="text" name="description" />  
 <input type="file" name="file">  
 <input type="button" value="上传" onclick="doUpload()" />  
 </form>

url:上传的接口 (eg: *api.v1.clients.1.documents*)

callback:成功返回的回调函数