# ES5代码规范

## 代码错误与代码理解误区：

var i = 'ok';

function changeI(){

var c = {a:1};

for(i in c){

}

}

sayI();

changeI();

sayI();

function undefinedTest(){

console.log('a', a);

console.log('b', b);

if(false){

var a = 1;

}

}

function sayI(){

console.log('i',i);

}

undefinedTest();

## 代码风格：接近C语言

* 变量先定义后使用。

buildUrl: function (pp, rf, url) {

var \_url = url;

for (i in pp) {

var baseForm = this.getForm(),

pv = pp[i],

v = '';

if (pv === "field-value") {

v = baseForm.getValues()[rf];

} else if (i === 'objCate') {

v = pp[i];

} else if (Ext.String.startsWith(pv, 'other-field:')) {

v = baseForm.getValues()[pv.replace('other-field:', '')];

}

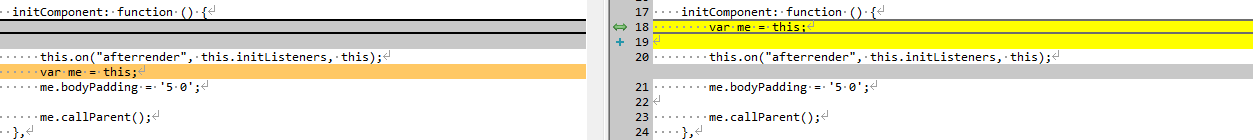
\_url = \_url.replace('{' + i + '}', v);

}

return \_url;

},

* 在一个作用域内，所有变量统一定义在起始位置。



* 最好只用一个var.

onLoad: function (me, records, success, opts) {

if (!success || !records || records.length == 0) {

return;

}

var mkeyword = this.valueField;

var selModel = this.getSelectionModel();

var store = this.getStore();

var allSelections = this.allSelections;

if (allSelections.length > 0) {

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

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

var record = records[i];

if (record.get(mkeyword) === allSelections[j].get(mkeyword)) {

selModel.select(record, true, true);

break;

}

}

}

}

}

function(olddata){

    var keyCode = "transObjCode",

     key,

        sumKey = ['qty','amount','taxAmount'],

        fn = new r2.global.BasePrinter().tplfn,

        tpl = new Ext.XTemplate('<table id="table-info" border="1" cellspacing="0" cellpadding="1" width="100%" style="border-collapse: collapse"><thead align="center"><tr><td>序号</td><td>存货编码</td><td>存货名称</td><td>规格</td><td>材质</td><td>单位</td><td>件数</td><td>重量</td><td>单价</td><td>价税合计</td></tr></thead><tbody><tpl for="transDetail\_bom"><tr align="center"><td>{#}</td><td>{transObjCode}</td><td>{objName}</td>',

'<td>{specification}</td><td><tpl if="Ext.isObject(material)">{material.text}<tpl else>{material}</tpl></td><td>{measureUnit}</td><td>{qty}</td>',

'<td>{[this.calcWeight(values,"auStandardWeight","agStandardWeight","qty","0.00")]}</td>',

        '<td>{price:number("0,000.00")}</td><td>{amount:number("0,000.00")}</td></tr></tpl></tbody><tfoot align="center"><tr><td>合计</td><td></td><td></td><td></td><td></td><td></td><td id="tdQty">{transDetail\_bom:this.sum("qty",false,"2")}</td><td>{[this.totalWeight(values,"transDetail\_bom","auStandardWeight","agStandardWeight","qty","2")]}</td><td></td><td>{transDetail\_bom:this.sum("amount",false,"2")}</td></tr><tr align="left"><td id="auWeight" colspan="7">金总重: {[this.metalWeight(values,"transDetail\_bom","auStandardWeight","qty","2")]}</td><td id="agWeight" colspan="6">银总重: {[this.metalWeight(values,"transDetail\_bom","agStandardWeight","qty","2")]}</td></tr></tfoot></table>',

     fn

        ),

     hash = {},

        newData = [],

        targetItem,

        item,

        i,

        l;

    for(i=0,l=olddata.length;i<l;i++){

        item = olddata[i];

        key = item[keyCode];

        if(key in hash){

            targetItem = newData[hash[key]];

            for(var ii = 0,ll = sumKey.length;ii<ll;ii++){

                targetItem[sumKey[ii]] +=item[sumKey[ii]];

            }

        }else{

            hash[key] = newData.length;

            newData.push(Ext.clone(item));

        }

    }

    tpl.strict = true;

    return tpl.apply({transDetail\_bom:newData});

}

* 避免重复定义变量。

copyToJsonData: function (jsonData, tempObj) {

//重复项混入单一项(目前仅限上传附件grid)

for (var k in jsonData) {

if (k.indexOf('.') > -1) {

if (jsonData[k] instanceof Array) {

var firstKey = k.split('.')[0],

lastKey = k.split('.')[1];

if (tempObj[firstKey]) {

tempObj[firstKey][lastKey] = jsonData[k];

} else {

tempObj[firstKey] = {};

tempObj[firstKey][lastKey] = jsonData[k];

}

}

break;

}

}

//复制值到特定的json节点

for (var k in tempObj) {

if (jsonData[k]) {

for (var j in tempObj[k]) {

if (jsonData[k] instanceof Array) {

var arrLength = jsonData[k].length;

for (var i = 0; i < arrLength; i++) {

jsonData[k][i][j] = tempObj[k][j];

}

} else {

jsonData[k][j] = tempObj[k][j];

}

}

}

}

},

* 变量名合理复用，增加可读性 。

refValueChangeCallback: function (it, paramItem, paramField) {

var me = this,

rValue = '',

store = me.getStore(),

proxy = store.proxy;

if (me.inGrid) {

//当参数改变时候清空grid的数据 ，但保留一行空数据

var grid = me.up('r2Grid'),

gridStore = grid.getStore(),

btn = grid.down('button');

//这里弃用 removeAll 方法，因为要能触发 store的 remove事件 必须得依次删除每一行的明细数据

while (gridStore.getData().items.length > 0) {

gridStore.removeAt(0);

}!grid.notAddOneRow && btn.handler(btn);

}

if (it.changeEventSource == 'setR2Value'){

this.changeEventSource = 'setR2Value';

setTimeout(function (){

it.changeEventSource = 'user';

},0);

} else {

me.changeEventSource = 'user';

this.valueFlag = false; //如果不是由setValue引入

}

if (it.xtype == "r2Selector" || it.xtype == "r2Combo") {

rValue = it.getRelValue(paramItem.value.valueField);

} else {

rValue = it.getValue();

}

if (me.win) {

grid = me.win.down('grid');

grid.store.proxy.setExtraParam(paramField, rValue);

grid.store.reload();

}

this.lastQuery = rValue;

proxy.setExtraParam(paramField, rValue);

for (var k in proxy.extraParams) {

if (proxy.extraParams[k] === null) return;

}

if (this.storeLoadDelay){

this.store.load({

callback: function () {

this.storeLoadDelay = false;

this.isR2Valid();

}.bind(this)

});

} else {

this.reset();

}

},

* For循环里没有局部变量

## 健壮性:

* 去掉数组对像最后一个元素的逗号。
* 使用变量的属性时先检测变量是不是undefined.

renderer:function(value,obj){

var isLeaf = obj.record.get('leaf');

if(isLeaf){

var listId = obj.record.get('url').split('/')[1];

var baseNum = r2.global.workFlowMessage[listId];

if(baseNum){

baseNum = baseNum>99?'99+':baseNum;

return '<span>'+value+'</span><span style="background-color: red;color: white;height: 25px;border-radius: 17px;padding: 0px 4px 0px 4px;position: relative;box-shadow: 0 1px 0 rgba(76,78,84,0.2);font-style:normal;font-size:12px;left: 5px;">'+baseNum+'</span>';

}

};

return value;

}

# 这样做的显示好处：

* + 避免出错
  + 好调试
  + 容易知道什么时候要拆分代码了。