codereview 问题总结 与 反思

出现问题的简单归纳

1. TPL命名不够语义化，应做到能通过名字了解模板的内容或作用；
2. 模板内容相近的，可以合并成一套；
3. 模板中对value的赋值需要双引号，否则低版本浏览器会报错；
4. 页面中多次出现ajax请求，可将ajax请求封装一个通用方法，然后在需要时调用；
5. ajax中，用fail 方法替换error方法，done替换success方法；
6. 在多交互的页面，alert提示信息较为重要时，要考虑用户浏览器alert功能的禁用情况，应将alert方法替换为自己写的alert组件；
7. 变量的命名不够语义化，同时命名相近，容易让人混淆，注意不同的变量或者类起名字的时，最好不要内容格式太相近，很容易误导别人；
8. 命名代码规范：常量用大写，变量用首字母小写的驼峰；
9. 定义方法时，使用function xx(){}方式好一些（后面有解释）；
10. jquery工具使用中，对data()，attr()，prop()的错误使用；
11. less中，对于多次使用的属性，比如颜色，长度值，应保存为变量，便于使用及后续更改；
12. 正则表达式使用错误；

问题详细描述

1. TPL命名不够语义化 及 模板内容重复：

```
var TPL = [
    '<select class="field-control region region-level-{{level}}" level="{{level}}">',
        '<option value="">请选择</option>',
        '{{if list && listLen>0}}',
            '{{each list as item}}',
                '<option value={{item.rid}}{{if item.rid == selVal}} selected{{/if}}>{{item.name}}</option>',
            '{{/each}}',
        '{{/if}}',
    '</select>'
].join( '' );
var OPTION = [
    '<option value="">请选择</option>',
    '{{if list && listLen>0}}',
        '{{each list as item}}',
            '<option value={{item.rid}}{{if item.rid == selVal}} selected{{/if}}>{{item.name}}</option>',
        '{{/each}}',
    '{{/if}}'
].join('');
```

分析：

如图中所示，模板的命名为TPL与OPTION，其中TPL的命名显然无法清晰标明该模板的作用，同时，上下两个模板中存在完全重复部分；

修改：

```
var tpls = {
    'select': [
        '{{if isSelect}}',
        '<select class="field-control region region-level-{{level}}" level="{{level}}">',
        '{{/if}}',
            '<option value="">请选择</option>',
            '{{if list && list.length>0 }}',
                '{{each list as item}}',
                    '<option value="{{item.rid}}" {{if item.rid == selVal}} selected{{/if}}>{{item.name}}</option>',
                '{{/each}}',
            '{{/if}}',
        '{{if isSelect}}',
        '</select>',
        '{{/if}}'
    ].join('')
};

module.exports = tpls;
```

将模板合并为一个，由传入的参数决定使用哪种形式，value赋值使用双引号；

结论：

对于需要使用模板的页面，最好将模板单独存放于一个js文件中，这样方面管理，且业务逻辑的js中更方便的去格式化代码；模板的命名一定要语义化，能通过名字了解到模板的作用。

---

2. 模板中对value的赋值没有使用双引号：

```
'<option value={{item.rid}}{{if item.rid == selVal}} selected{{/if}}>{{item.name}}</option>',
```

分析：

如图中所示，模板中对value的赋值，没有使用双引号，这在低版本浏览器中，就会导致模板编译报错；

修改：

```
'<option value="{{item.rid}}" {{if item.rid == selVal}} selected{{/if}}>{{item.name}}</option>',
```

结论：

在使用artTemplate编译模板时，注意书写规范，属性的赋值要用双引号或者单引号（具体使用哪个看模板的数组元素使用的是哪个，不同即可）避免因为书写不规范导致出现错误。

---

3. 页面中多次出现ajax请求却没有抽离出公共部分，同时使用了过时的方法error和success：

```
$.ajax( {
    url: EXIST,
    data: {trans_bill_id:aimTransbill },
    dataType: "json",
    type: "POST",
    success: function(response){
        if(response.content.exist){
            COULDMOVE = true;
        }else{
            COULDMOVE = false;
            $(".undefined-error" ).show();
        }
    },
    error: function () {
        Alert.show("请求失败!");
    }
} );
```

分析：

如图中所示，类似的ajax请求在页面中共出现了四次，每次请求只有url和传参不同，同时，对于请求结果，使用了过时的方法success和error；

修改：

```
function ask(opts){
    return $.ajax( {
        url: opts.url,
        data: opts.data,
        dataType: "json",
        type: "POST"
    } );
};
```

```
ask({
    url: UPDATEMONEY,
    data: moneyParams
}).done(function ( response ) {
    Loading.hide();
    if(response && !response.msg){
        Alert.show( '修改成功', function () {
            window.location.reload();
        } );
    }else{
        Alert.show(response.msg);
    }
} ).fail(function ( ) {
    Loading.hide();
    Alert.show("请求失败!");
});
```

将$.ajax方法抽离出来，省略掉了每次都需要传的其他参数（类型等），同时将error和success方法替换为fail和done方法；

结论：

当在一个功能页面中，一个类似的方法出现多次时，应该考虑将其拆分出来，变成一个可以通用的方法，方便整体的使用和维护；在之前，一直觉得像这种ajax请求并没有封装的必要，因为封装也只是减少了寥寥几行代码而已，反而还因为封装而浪费时间，但是听了导师的说法，觉得还是很有道理的，能力的锻炼和培养是要在平时不断积累的，好的思想和习惯要在平常主动去养成，不能觉得效果不大就不去做，在小的细节上不注意，不去培养自己这种思维，习惯使然，在遇到其他问题时也很难自己去想到要那么做，平常的不注意，无所谓，实则浪费了很多可以锻炼自己的机会。

jquery1.5版本对$.ajax()返回的对象添加了回调函数error(),success(),complete()方法，但是由于版本的升级，从 1.8 版本开始，以上三个方法开始被弃用过时，将从 3.0 版本开始删除这三个函数，取而代之的是.done(),.fail(),.always()三个方法，具体内容可查看网址：http://www.css88.com/jqapi-1.9/jQuery.ajax/#jqXHR。

---

4. 多交互页面，没有考虑用户禁用alert弹窗的可能：

```
if(response && !response.msg){
    alert( shipList.length+'个订单已成功转移至运单'+aimTransBillNo);
    window.location.reload();
}else{
    alert(response.msg);
}
```

分析：

如图中所示，由于页面中设计交互较多，所以很多地方都需要alert弹窗来提示用户当前的状态及下一步的操作，但是当用户浏览器禁用了alert功能时，页面的交互将不再提示这个必要信息，严重影响用户的操作；

修改：

```
if(response && response.ret === 0){
    Alert.show( shipList.length+'个订单已成功转移至运单'+aimTransBillNo, function () {
        window.location.reload();
    } );
}else{
    Alert.show(response.msg);
}
```

用Alert组件替换掉了浏览器自带的alert方法。

结论：

使用自己写的Alert组件来实现浏览器的alert提示功能，避免了因浏览器禁用alert而影响用户的使用；同时，对于类似的问题，如果有因外界因素可能导致失效的必要功能，应该考虑用其他方式代替。

---

5. 变量名或类名起名过于相似：

```
var $regionSelect = opts.$mod.find( '.region-select' );

//目前页面上已经存在的地址选择框
var domList = $( 'select.region' );
```

分析：

如图中所示，选择器region-select与select.region命名相似，让人容易混淆；domList命名不够语义化，不易理解；

修改：

```
var $regionSelect = opts.$mod.find( '.region-select' );

//目前页面上已经存在的地址选择框
var domRegionList = $( 'select.region-item' );
```

将两个class重新命名，区分出父子级，region-select为整体组件外层，region-item为组件内部每个选择单元；domList 改为 domRegionList，增强语义；

结论：

变量的命名注意语义化，相似的东西避免出现类似的名称，减少让人误会几率；

---

6. 变量与常量的命名规范：

```
var UPDATEMONEY = '/order/transbill/updatemoney';
var UPDATEORDER = '/order/transbill/move';
var EXIST = '/order/transbill/exist';
var EXPORT = '/order/transbill/export';
var COULDMOVE = false;
```

分析：

如图中所示，对于变量COULDMOVE不应该使用大写；同时对于常量，使用的是大写，不易阅读；

修改：

```
var SHIPLIST = '/order/transbill/getshiplist';          //获取当前运单下的订单列表
var UPDATEMONEY = '/order/transbill/updatemoney';       //调整运费
var UPDATEORDER = '/order/transbill/move';              //调整运单(移动订单)
var EXIST = '/order/transbill/exist';                   //校验目标运单是否存在
var EXPORT = '/order/transbill/export';                 //导出接口
var couldMove = false;
var transMoneyConf = conf.tplData.content.trans_money_conf;
```

将变量修正为驼峰式命名，对于不易阅读的大写常量，增加了注释；

结论：

命名规范：常量命名用大写；变量命名用驼峰；对于不易阅读的变量，要添加注释；

---

7. 方法的声明方式问题：

（1）表达式（字面量）方式：

```
var transbill = function (){}
```

（2）函数式方式：

```
function transbill (){}
```

分析：
原代码中使用第一种方式，函数表达式的方式对函数进行了声明；

资料整理：
（1）两者的区别：
后者是指函数声明，前者是指函数表达式，他们之间的区别是后者会在代码执行之前被JS解释器加载到作用域中，这样一来就可以在编程时在定义函数之前调用这个函数，此法是有效的；而前者则是在代码执行到那一行时候才会有定义，此外函数表达式是创建了一个匿名函数，然后将匿名函数赋值给一个变量。

（2）使用示例：如果，像下面这样调用函数：

```
transbill()    //会报错
var transbill = function (){}
transbill()    //不会报错
```

```
transbill()     //不会报错
function transbill (){}
transbill()     //不会报错
```

表达式（var 方式）定义的函数，不能先调用函数，后声明，只能先声明函数，然后调用。函数式（function方式）定义函数可以先调用，后声明。

结论：
由于在codereview的代码中，transbill变量只是用于函数声明，后面并没有用到transbill变量，所以使用表达式的方式声明函数是没有实际意义的，同时还会增加调用失败的风险，所以这里使用函数式声明的方式更合理一些。

---

8. jquery方法，attr(),prop(),data()的使用问题，各方法的区别及使用场景：

资料整理：
attr函数和prop函数都用于设置或获取指定的属性，它们的参数和用法也几乎完全相同。

attr：attributes，prop：property，均表示属性的意思；
区别：

> *（1）版本上：attr()是jQuery 1.0版本就有的函数，prop()是jQuery 1.6版本新增的函数。*
>
> ---
>
> *（2）概念上：attribute表示HTML文档节点的属性，property表示JS对象的属性。*
>
> *所以，attr()函数针对的是该文档节点的attribute，也就是该DOM非自身带的属性；prop()函数针对的是该DOM元素(msg)自身的property；*
>
> ---
>
> *（3）使用：由于attr()函数操作的是文档节点的属性，因此设置的属性值只能是字符串类型，如果不是字符串类型，也会调用其toString()方法，将其转为字符串类型；prop()函数操作的是JS对象的属性，因此设置的属性值可以为包括数组和对象在内的任意类型。*
>
> *（4）其他：对于表单元素的checked、selected、disabled等属性，在jQuery 1.6之前，attr()获取这些属性的返回值为Boolean类型：如果被选中(或禁用)就返回true，否则返回false。*
> *但是从1.6开始，使用attr()获取这些属性的返回值为String类型，如果被选中(或禁用)就返回checked、selected或disabled，否则(即元素节点没有该属性)返回undefined。并且，在某些版本中，这些属性值表示文档加载时的初始状态值，即使之后更改了这些元素的选中(或禁用)状态，对应的属性值也不会发生改变。*
> *因为jQuery认为：attribute的checked、selected、disabled就是表示该属性 初始状态 的值，property的checked、selected、disabled才表示该属性 实时状态 的值(值为true或false)。*

data()方法：

> *data()方法同样可以对html元素进行数据的存取，尽管"data-" 是HTML5才出现的属性，但jquery是通用的，所以，在非HTML5的页面或浏览器里，你仍然可以使用.data(obj)方法来操作"data-" 数据。在使用data(value)修改数据时，如果value是undefined类型，那么数据不会保存或更新；和attr()值返回string类型的值不一样，data()会对静态绑定*

> *的数字、布尔、对象、数组和null进行转换，data对大小写严格；在IS里第一次调用data()时，会将HTML里静态绑定的数据，复制到jQuery.cache变量里，复制时key中的字符都转换为对应的小写字母。再次使用data()修改数据或添加新数据时，对key不会转换为小写字母，也不会对存入的数据做类型转换！*

结论：
对于HTML元素本身就带有的固有属性，在处理时，使用prop方法。
对于HTML元素我们自己自定义的DOM属性，在处理时，使用attr方法。
在页面用于存储和传递页面数据，data比attr更合适。

---

9. less中，多处使用的属性值统一存储问题：

```
      border-radius: 3px;
      border: 1px solid #cccccc;
      padding-left: 8px;
    }
    .search-error{
      color: red;
      margin-left: 5px;
      display: none;
    }
  }
}
.modal-body{
  .waybill-body{
    .order-list-head{
      background-color: #cccccc;
      width: 100%;
      span{
        display: inline-block;
        width: 47%;
        text-align: center;
        height: 30px;
        line-height: 30px;
      }
    }
    ul.order-list{
      max-height: 300px;
      overflow-y: auto;
      border: 1px solid #cccccc;
      li{
        height: 30px;
        line-height: 30px;
        border-bottom: 1px solid #cccccc;
        padding-left: 5px;
```

分析：

如图中所示，在less文件中，多次用到了#cccccc这个颜色，多个地方使用，修改时需要改多个地方，不易维护；

修改：

```
@background-color: #cccccc;  //背景颜色

.order-list-head{  background-color: @background-color;}
```

结论：
对于less中多次使用到的属性值，应存储为变量，方便后面的多次使用和修改维护；

---

10. 正则表达式的使用错误：

```
/^\d+.?\d{0,2}$/        //验证输入为金额
```

分析：
该正则表达式会把“11 22”等当做正确金额；

修改：

```
/^\d+\.?\d{0,2}$/          //验证输入为金额
```

结论：

“.”匹配除换行符 \n之外的任何单字符。要匹配 “.”， 需使用"\点 "进行转意。

总结

基于本次的codereview，暴露出自己很多的写代码上的问题，一些是源于不认真，一些是真的不会，让自己认识到，自己急需提高自己的代码质量了；所以短期内，有以下主要计划：
1. 针对codereview中提出的问题，对自己之前所写的代码，进行整理修改；
2. 对于自己欠缺的知识，比如jquery中很多的API，需要抓紧时间去学习；
3. 抓住这次codereview的机会，不仅从自己身上找不足，也要从别人身上多学习优点，从每次别人的codereview中发现的问题，去找自己可以优化的地方；
4. 合理安排时间，每天留给自己代码学习和优化的时间，这个刚开始的时候因为开发进度比较紧，可能会有点困难，但是可以从每天写的代码中去约束自己，多注意细节，多思考之前犯得错；