个人项目总结

项目完成时间：2017/09/18---2017/09/22
项目名称：“慢慢买”
合计：16个动态页面
子页面共同特征：全部使用rem布局，除了字体可以写成固定的px。

考核技术点：1）**AJAX请求** 2）动态获取到数据，并通过**模板template渲染**到页面上来 3）静态布局
4）代码的**健壮性**（当跳转的页面之间相同点多的时候，需要将共同的部分用函数封装起来，便于后期维护） 5）当页面需要点击来获取数据时，需要考虑**性能问题**，最好是点击一次获取到数据以后，当再次点击时不要重复的发送ajax请求了。以便减轻服务器的压力！

get技巧点：首先看接口文档，然后就写js,通过获取到的数据类型再去决定html里面的模板部分该怎么写！如果获取到的是dom结构，比如<img...>这样的数据，那么在模板里面就只需要写成{{#$value.productImg}},这里注意：由于数据获取到的是结构而不是图片地址，所以我们需要加上#号去解析这个结构。

1、首页index

1）点击“返回顶部”：可以加个缓动效果让页面回到顶部的状态：

```
//点击回到顶部，实现滑动的效果
var back=$('.foot .back');
back.on('click',function () {
    //滑动的速度
    $('body,html').animate({ scrollTop: 0 }, 200);
    return false;
});
```

2）页面共需要发送两次ajax请求，通过ajax获取到数据渲染到页面上。菜单栏和商品列表栏里面的每一项都是点击可跳转的。

2.1 比价搜索页面（手风琴效果）

1）点击的时候发送ajax请求并将数据展示在下拉框里面

2）这里的每个li标题其实分为两个部分【标题（模板一渲染）+下拉框（模板二渲染）】，所以这里有个需要注意的点就是：在每个标题上加上点击事件去创建下拉框数据时（需要在第一个ajax请求的成功回调函数里面给每个标题绑定点击事件，另一个ajax请求在点击事件里面去完成），需要将点击事件绑定在每个li的子元素一身上而不是li身上，因为当下拉框创建出来之后，li就是整个子元素一加上子元素的高度，所以当点击下拉框里面的元素时，相当于还是在点击li，这样容易

出问题。

3）这里考虑一个性能问题，优化后可以实现点击一次加载完数据，当再次点击时就不需要重复的发送ajax请求了，减轻服务器压力

```
//如果点击的理下面已经有了内容，就不去加载数据了
if (_this.find('.pro_intro>ul>li').length) {
    //排他展示
    _this.siblings().find('.pro_intro').slideUp(200);
    //展开隐藏当前li
    _this.find('.pro_intro').slideToggle(200);
    return;
}
```

2.2商品列表功能界面

面包屑导航：前两项写成固定值，后一项通过发送ajax请求到的数据去取到对应的值放进来

为了防止直接从跳转页面进去拿不到数据的情况出现，需要给ajax传入的参数（str）一个默认值！

```
var str = window.location.href;
//切分的方法要恰当！！
str = str.split('categoryid=')[1];
//这里要给一个容错！如果没有这个值就给一个默认值！
str = str || 0;
```

3 省钱控页面

分页：

由于需要同步左右点击按钮和中间框里的值，所以这里可以封装一个**getData**函数，并将函数里面的参数作为全局变量进行声明，然后在左右按钮的点击事件里面对这个参数进行++或者--（需要进行极值判断），然后再进行一次函数调用，就可以实现数据的同步了。

中间的**select-option**框是动态创建出来的，所以在**html**里面写模板的时候需要自定义一个属性**selected**，然后再讲选中的这个值（参数）放在中间的框里，模板里面需要从**js**里面传过来两个参数（以对象的形式传过来！）：

<!--分页模板-->

<script type="text/template" id="pagetpl">

```
    <select>
      <%for(var i=1;i<=num;i++){%>
       <%if(currentPage==i){%>
           <option value='<%= i %>' selected> <%= i %> / <%= num %>
           </option>
      <%}else{%>
           <option value="<%=i %>" ><%= i %> / <%= num %></option>
          <%}%>
     <%}%>
    </select>
</script>
```

改变中间框的值：

```
//设置onchange事件
select.change(function () {
    var option = $(this).children();
     currentPage = $("option:selected").val();
    console.log(currentPage);
    //函数内部(局部作用域)没法改变全局的变量值，所以这里不能在currentPage前面再添加var!!!!!!
    getData();
})
```

4 白菜价页面

1）头部区域的滚动可以通过引入插件(**iscroll-lite.js**)来实现左右拖动，需要加上如下代码：

```
var myscroll = new IScroll(".itemWrap", {
    scrollX: true,
    scrollY: false
});
```

ul的宽度需要动态去计算得到！//这里由于获取的Wrap（ul）是JQ方法获取的元素，所以需要用width方法去写！Wrap.width(lisWidth * lis.length);

这里需要注意：myscroll必须写在ul的长度创建出来之后！！

2）每个标题的点击事件里面再去获取相应的数据列表展示到页面，页面一上来可以默认加载第一页的数据（给参数默认传一个具体值）

3）这一页的静态布局是一个比较棘手的活，多而杂，很多小细节需要注意，而且因为这里获取到的数据全都是dom结构，所以里面的数据全都是渲染到页面之后再去调的样式。

4）进度条：

ajax请求返回的数据已经把结构写好了：

```
<i>
   <em style="width:34%;"></em>
   <span>34%</span>
</i>
```

然后拿到数据后，再去less里面添加样式:

给 **i** 添加border-radius并设置背景色，给 **em** 添加另一个背景色和border-radius，并加上transition：all 0.3s cubic-bezier(0.39, 0.58, 0.57, 1)，然后给 **span** 里面的文字水平居中就可以了！

5 优惠券页面

当点击相应页面时，跳转到相应页面，并给点击的商品上方加一个黑色带透明度的遮罩层，并且将对应的商品图片放到黑色遮罩层中央，此处可以通过插件来实现，并加上手滑效果或者轮播图的效果（待解决！）

6 凑单品页面（tab栏）

这里的逻辑和“分页”有些类似，都是通过点击事件去改变全局的某个变量，然后将它作为参数传到函数里面去，从而获取到对应的数据渲染到页面上。

1）点击上面菜单栏的时候，实现文字同步（注意：这里并不能用text或者html方法进行文字的替换，因为这样会把菜单栏里的i标签一起给干掉，所以可以先把里面的内容清空后再单独加入i标签）：

```
var txt = $(this).children('a').html();
var $i= _this.children('i');
_this.empty().append(txt).append($i);
```

2）第一个菜单栏的下拉框和第二个菜单栏的下拉框是独立渲染出来的，只不过放在了页面相同的位置进行了展示，中间商品区域的数据是依赖于前两个菜单栏选中的数据(这两个数据正是商品区域发送ajax所需要的两个参数)进行渲染的。

7 品牌大全页面

这是一个特殊的页面。请求了三个ajax，分别渲染在三个部分

1) **var arr = [ ];** 这里弄一个空数组是用来拿到第二个**ajax**里面的图片和文字信息的，从而可以渲染到第三个模板里面。

在第三个**ajax**请求是成功回调里面需要这样写：

```
success: function (data) {
    for(var i=0;i<data.result.length;i++){
            var item = data.result[i];
            for(var k in arr[i]){
                //遍历对象
        item[k] =  arr[i][k];
            }
    }
    //这里已经把原始的data数据进行了改动，添加了上一个AJAX里面的arr的
```

数据进来了！因为arr是全局的，所以在任意处都能获取到它的值.

```
    commentWrap.html(template('commentTpl', data));
}
```

2) 如果需要将模板里面的数据写成后台返回的result里面的下标的话，那么模板里面可以直接写成$index。

-----art-template-----

条件：

1）标准语法：

```
{{ if value }}
.....
{{ /if }}

{{ if v1 }}
.....
   {{ else if v2 }}
.....
{{ /if }}
```

2）原始语法

```
<% if (value) { %>
...
<% } %>

<% if (v1) { %>
.....
<% }else if (v2) { %>
.....
<% } %>
```

循环:

1) 标准语法:

```
{{each result}}
   {{$index}} {{$value}}
{{/each}}
```

2） 原始语法：

```
<% for(var i = 0; i < target.length; i++){ %>
   <%= i %> <%= target[i] %>
<% } %>
```

1. target 支持 array 与 object 的迭代，其默认值为 $data。
2. $value 与 $index 可以自定义：{{each target val key}}。