

前端规范：

HTML部分

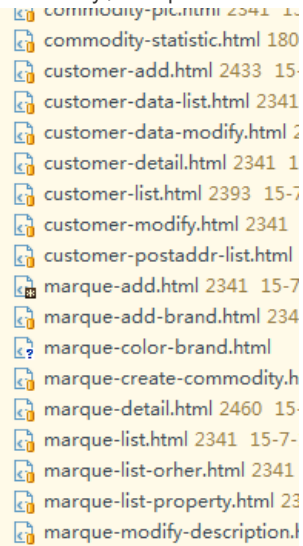
- 下面的规范都是为HTML以整洁易维护为前提，所以首先尽量避免在HTML写样式，如实在无法避免，也请统一将样式写在文件的CSS文件引用之后的style标签体内，不要写标签内联样式。
- 对于CSS和JS文件的引用位置，CSS在头，JS在文档尾部就不用多说了；引用优先级遵循如下规则：

基础框架库的CSS和JS，如JQ和Bootstrap及对应模板。

各种小工具插件，如表单验证插件，弹框插件。

项目公用CSS，公用的command方法和项目命名空间的API。

最后引入当前页的样式和对应JS文件。

- 设置body标签 的ID 值为当前页的文件名称。
 - 为body或是最外层的div 追加class类名，名称根据所在功能模块的英文名，比同一功能下的几个文件名的相似的前缀单词，这图里面的commodity和marque
- 
- 对于常见复用较高的表单，应该设置相同的类名，现在的大家直接用模板类来设置格式，而没有别的名字，维护很麻烦。
 - 类的命名规范，如果是form和button这种功能性比较突出的标签，类名建议可以为form-*,btn-，其它类型标签，就请按其内容所属功能或是数据特征进行命名。
 - 对于JS脚本生成的标签，建议在用\$(*).html()方法写入dom时，对该元素或其子元素追加标识此元素为js生成的类名 jscr-el，方便自己调试维护和添加代码可读性。

CSS部分

- 首先是批注原则：

CSS文件头标注对应CSS环境和版本

```
/*
```

```
@name: 易道ERP后台系统
@description: 1.0
@require: edaw.css
@branch:base
*/
```

branch 注明所属分支，比如现在大家按文件夹分类的各功能类别的名。

针对某特定页面的则标该页面名。

```
/*
@branch:人事管理
@pages:员工信息查看
*/
body#employee-detail #gender,
body#employee-detail #stature,
body#employee-detail #cardId,
body#employee-detail #folk{
    max-width: 250px;
}
```

- 嵌套原则：

对于全局标签样式，嵌套上一级的公用类名，如无有明显含义的类名，则可不嵌套。

对于针对特定几个页面的或是特定页的样式，则必须嵌套主节点ID或类名（如form、table标签或主DIV）。

实在没有则现在就可以用到之前HTML规范里写好的主body的类或ID名。

- 对于复用属性，可利用多重选择器，一起设置，如上图中就有部分样例。

JavaScript

目录结构：

```
|---/js/
|---- /bootstrap/           框架和模板所用到的JS
|---- /libs/jquery-1.9.x.min.js  jq等基础库
|---- /libs/pluginA/         使用到的js插件A
|---- /libs/pluginB/         使用到的js插件B
|---- /libs/pluginC/         使用到的js插件C
|---- /pageA~Z/              对应各页面所写的业务js目录
|---- commond.js             单独书写的js(如公用方法)
|---- edow.js                单独书写的js(如一些业务API)
|---- plugins.js             调用的plugins汇总（如弹窗和表单验证调整后的API接口）
```

- 引用的js库保留版本号，压缩版本的文件名保留.min，另外插件要把未压缩版本也放在该插件的.min压缩版本文件夹下，方便调试和功能修改时查看源码。
- PC端的JS虽说刚开始不用太注重性能优化，但至少要满足易读性和通用JS规范；
- 能避免就尽量避免直接在HTML页面中写代码块。
- **JS注释：**

如果包含了其子页的中的代码，应该把孩子页面所用到的方法放一起,头部和尾部包括形进行标注属性哪个子页面。

```
/**
@pages 人事管理
@comprise 业务处理函数返回值和所使用到的业务数据对象
**/
业务代码区
/**
人事管理END
**/
```

```
/**
@pages 新增员工信息
@comprise 业务处理函数返回值和所使用到的业务数据对象
**/
业务代码区
/**
新增员工信息END
**/
```

对于大量生成页面元素的方法，尽量作一下标注，并至少对主标签或全部加上一个标识的类名，方便后期样式的修改索引查找。

- 尽量减少声名全局变量，将所需要用到的业务功能方法，数据转存或是处理对象，都尽量合并到一个全局API对象或是common命名空间中，建议借鉴移动APP端的一些规范，将所有公用方法API放在一个对象waker中，将业务方法放在对应该页的一个以页面或是js命名的对象中。
- 对于插件类的吊用，对于复用很高的函数，虽然各页面需要使用的参数不一样，也可以在公用API中将该方法写好，然后将可能各页面需要使用到该插件的不同API通过参数传过来，接着对复用最多的参数值设置成不带参运行时的默认环境值；如我对toPage函数的改造，以及对ajax的小改造；
- 尽量对公用方法中一些方法的传入的参数进行简化和兼容，比如能传对象或是数组的，就不要设置成只能接受拼接字符串，当然这属于个人习惯，我个人建议优化接收参数的多样性，让公用方法使用起来简单些。

好了，暂时就这些，如有新的建议随时提出讨论。
