模块化CSS指引

问题

随着移动端项目越来越大，随意的CSS命名方式就带来了很多问题。

- **迭代快**，几天一次迭代，原来随意的命名并不能考虑到后面的迭代需求，导致只能通过样式覆盖的方案解决，并且随着覆盖次数越来越多，最终越来越难维护；
- **需求不稳定**，原来的需求有可能经常被变更，导致前期书写的代码被废弃，成为垃圾，并难以轻易剥离删除；
- **大项目**，如果项目比较大，开发持续的时间必然也会比较长，随意的样式命名也就变得越发容易冲突；
- **多人协作**，每个人的样式命名习惯都有所不同，没有统一的命名方式就无法很解决项目移交或多人并行开发；

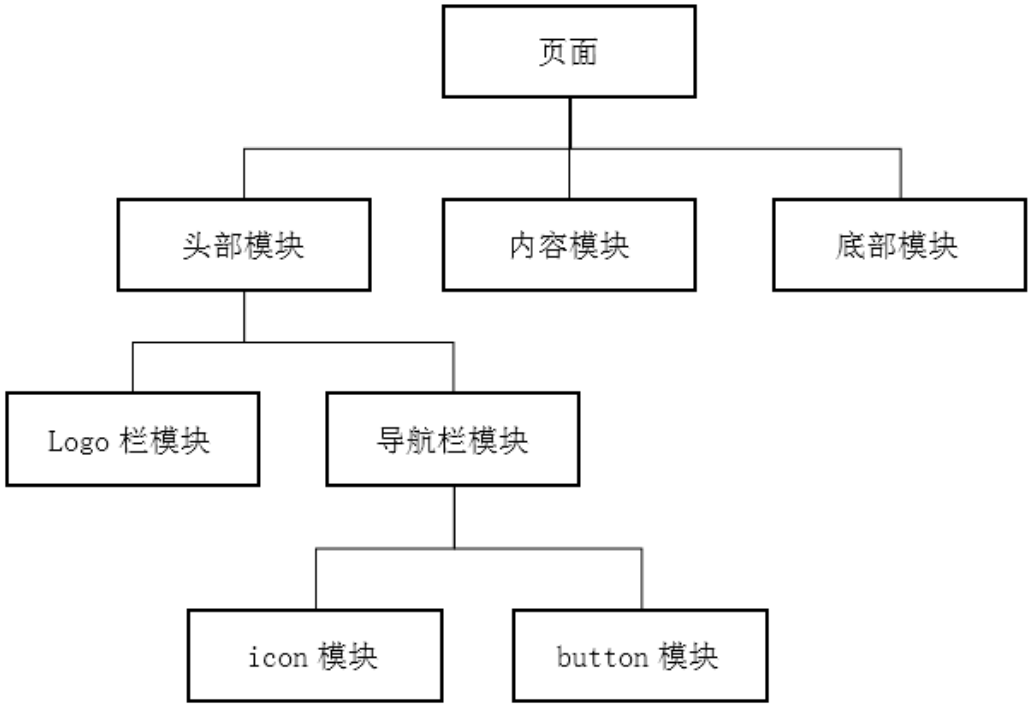
解决原则

为应对上面提出的问题，要求我们采用的方案必须具有独立性、适应性、易维护、样式低耦合、适应移动设备（PC可以在基础上调整）等特性。

核心思路

根据特性，样式命名应该围绕**组合模式**、**封闭开放**、**继承重载**三个核心来搭建

1. 组合模式



页面模块例子

理论上讲，一个完整的页面可以被划分为不同的功能模块，如上图是一个常见的基于模块化的页面结构，页面是由不同的模块组合而成，大模块又是由小模块组成。

而我们的所需要做的便是制作出组成页面所需要的模块，并把它拼合到一起

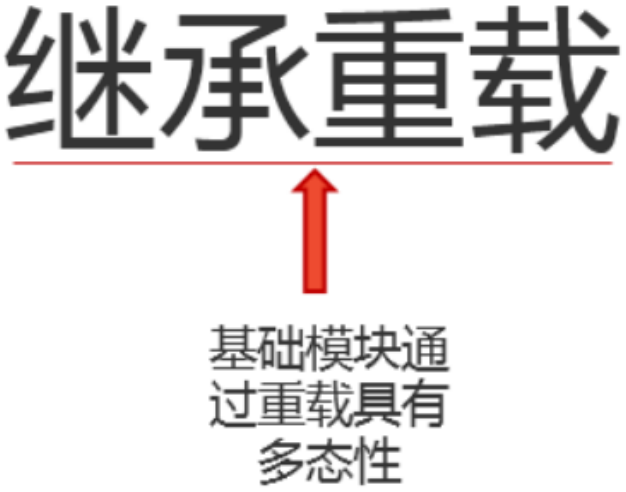
2. 封闭开放

每一个独立的模块都应该遵循“封闭开放”的原则，该原则原理与BFC类似。

封闭是指模块的基础样式处于封闭状态，其不会影响到外部元素的展示，常见的基础样式有border、color、padding、height等。

开放是指模块的布局样式应该在具体的使用场景中由外部环境定义，常见的布局样式有absolute、margin等。

3. 继承重载



每个独立模块在特定的场景可以通过部分重载成为新的模块。

具体实施

所有的定义起始都必须是基础**模块（mod-）**、**按钮（btn-）**、**图标（icon-）**、**命名空间（ns-）**4个之一，不接受除此之外的任何起始定义。

基础模块（mod-）

常见的基础模块一般可以基于功能来划分，如mod-header、mod-nav、mod-slider等等。

```
.mod-nav {
    width: 100%;
    height: 30px;
    background: red;
}
```

定义基础模块时，只定义基础样式，这样做的优点是该模块可以被复用到几乎任何环境中。

```
.mod-nav .list {
    display: -webkit-box;
    margin-top: 4px;
    height: 50px;
}
```

模块内的元素可以按需要进行自由定义，定义时选择器应当是基于“mod-xx下的xx”“这样的规则。

```
.mod-header .mod-nav {
    margin-top: 10px;
}
.mod-content .mod-nav {
    margin-bottom: 5px;
}
```

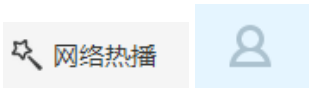
模块的布局样式由其所在的环境来定义，如上代码，nav模块在header模块环境中定义了向上margin，在content模块中则定义了向下margin，这样同一个模块在不同的环境中便有了不同的布局展示。

```
.mod-nav.sp {
    width: 300px;
    background: green;
}
```

通过选择器级联的方式很容易就能达到重载的目的，从而实现模块的多态。

`由于ie6不支持级联样式，此处的叠加样式可以写为.mod-nav-sp`

图标模块（icon-）



如上图所示，移动端页面中常常会有需要使用icon的地方，而项目中往往会时而需要带文字，时而不要，所以在定义时我们建议把icon元素定义在before：： 或after：： 中。

```
.icon-write::before {
    content: '';
    display: inline-block;
    width: 15px;
    height: 15px;
    background: url(../sp.png) no-repeat;
}
```

`同样，介于PC使用::before伪元素还不成熟，可以新建span标签，并命名为before来代替，这样此处代码应该为.icon-write .before`

这样在icon具体的使用环境中再去对其布局定义。

```
.mod-nav li .icon-write {
    position: absolute;
    top: 10px;
    left: 10px;
}
.mod-nav li .icon-write::before {
    vertical-align: -3px; /*基线处理*/
    margin-left: 5px;
}
```

经过这样处理后，图标定义具有3个优点：不依赖具体标签，可做按钮可做装饰；可加文字，可去文字；易于重复利用。

按钮模块（btn-）：

按钮的定义思路基本与图标一致，分为基础定义及布局定义。

```
.btn-detail-oper {
    display: inline-block;
    height: 28px;
    line-height: 28px;
    width: 66px;
    text-align: center;
    border: 1px solid #E6E6E6;
    -webkit-border-radius: 3px;
    border-radius: 3px;
    font-size: 13px;
    color: #fff;
    text-decoration: none;
    background: #2FB0FF;
}

.mod-detail-operate .btn-detail-oper {
    position: absolute;
    margin: 0 5px;
    top: 5px;
    left: 5px;
}
```

图标与按钮结合可以通过class叠加方式实现。

```
<span class=" btn-detail-oper icon-dl-white">下载</span>
```

命名空间（ns-）

命名空间做为一种补充存在，其作用主要有两个，一是可以定义在页面比较“根”一点的元素上（如body），做为该页面的根命名，一些模块在该页面比较特殊的变化可以通过它来完成，如夜间模式，对其做不做样式定义都可以。

```
.ns-night .mod-nav {
    background: #000;
}
```

另一种作用是对一些不需要模块化的布局元素进行定义，如wrap元素。

```
ns-wrap {
    margin: 0 atuo;
    width: 320px;
}
```

结合LESS使用