大规模 CSS 组织最佳实践

@糖饼

关于我

- 2009 ~ 2011　深圳-众创力合（合伙人）
- 2012 ~ 2014　腾讯，CDC、Qzone
- 2014 ~ 现在　厦门-欢乐逛

- UR 问卷系统
- 腾讯移动分析
- QQ 空间 8.0 版
- 微群组 Android 版

大规模 CSS 工程

- 业务规模与复杂度
- 多人协作

CSS 工程中坑无处不在

——项目下一个加入的同事会证明给你看

现状

```
<body>
    <div class="w40 h20">...</div>
    <div style="width: 40px; height:20px">...</div>
    <div class="clearfix">...</div>
    <div class="box card image">...</div>
    <div id="news">...</div>
    <div class="list">...</div>
    <div class="mod-list mod-list-hot">...</div>
    <div class="block__element">...</div>
</body>
```

三大现象

- 原子类
- 表现类
- 多重类

原子类

- 将 CSS 属性与值简写成一个类，然后根据需要在html元素上添加一个或者多个类，快速的实现想要的效果

- 示例：class="w20 h40 fl"

表现类

- 根据使用频率将某个局部样式封装成一个类，例如清除浮动
- 示例：class="clearfix"

多重类

- 将多个无关联的类组合成新的类
- 示例：class="box card image"

```
<body>
    <!-- 低开发效率（无法复用） -->
    <div id="news">...</div>
     <div class="w40 h20">...</div>
    <div style="width: 40px; height:20px">...</div>

    <!-- 难以持续维护（表现命名、依赖混乱） -->
    <div class="w40 h20”>...</div>
    <div class="clearfix">...</div>
    <div style="width: 40px; height:20px”>...</div>
    <div class="box card image">...</div>
</body>
```

```
<body>
    <!-- 容易产生样式冲突 -->
    <div class="list">...</div>
</body>
```

违背表现与结构分离原则

- 不利于理解与 SEO 优化
- 大量规则会增加团队学习成本
- 冗余代码会不断产生

关注点

- 如何提高 CSS 在多人协作开发效率?
- 如何避免样式随着项目规模导致复杂度攀升?

CSS VS 编程语言

- 优点：简单灵活，极其容易复用
- 弊端：缺乏保护机制、无法管理依赖关系

制定规范

- 设计公用样式
- 保护内部样式
- 设计继承规则

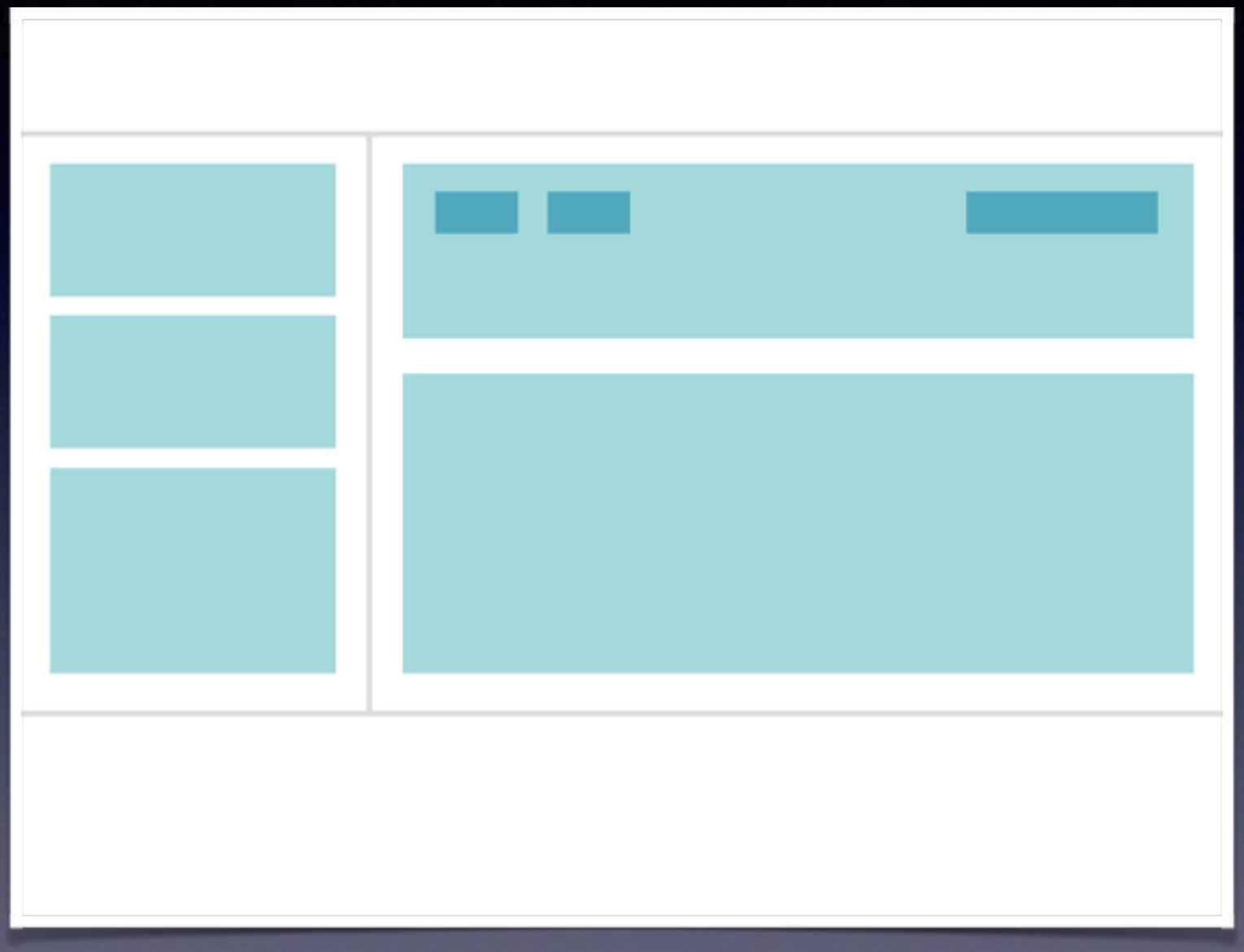
CSS MOD+ 规范

- 严格要求语义化：禁止表现命名
- 控制全局样式：以命名空间与作用域约束样式

怎么做?

模块化思想

- 将页面根据内容的关联性分解成不同的且相互独立的模块进行开发，每个模块之间没有必然的联系，互不影响



模块化与命名空间

- 布局
- 模块
- 控件

布局

- 负责站点整体结构，也是“模块”的容器
- 例如：header、main、sidebar、footer

选择器

- 使用 HTML5 标签或 ID 作为选择器开头

要求

- 布局的直接子元素只能是“模块”，不能是其他元素
- 布局元素必须是唯一的

模块

- 承载业务的容器，将页面根据内容的关联性分解成不同的且相互独立的区域

选择器

- 使用 .mod- 作为选择器开头

选择器

```
.mod-cards {...}
.mod-cards .title {...}
.mod-cards .content {...}
```

要求

- 所有模块应该直接堆砌在“布局”元素下
- 禁止模块之间嵌套，复用代码通过模块继承
- 模块子元素不能使用 .mod- 开头命名

模块类型

- 公用模块：放置在公用样式文件中
- 私有模块：跟随页面样式文件

模块类型

- 公用与私有模块不在命名上进行区别
- 公用与私有模块类型可以灵活转换

多态

状态与子元素使用短命名

```
.mod-cards-img.error .title {...}
```

继承

- 经典方式：使用类组合实现
- 例如：class="mod-cards mod-cards-img"

属性选择器实现继承

HTML

```
<div class="mod-cards-img">...</div>
<div class="mod-cards-txt">...</div>
```

属性选择器

- [attr]
- [attr=value]
- [attr~=value]
- [attr|=value]
- [attr^=value]
- [attr$=value]
- [attr*=value]

基类

```
/* 基类：卡片列表 */
[class^="mod-cards"],
[class*=" mod-cards"] {
    ...
}
```

继承基类

```
/* 不同类型的卡片列表，继承自 .mod-cards */
.mod-cards-img {...}
.mod-cards-txt {...}
```

More..

间距控制

```
/*设置模块之间默认间距*/
[class^="mod"],
[class*=" mod-"] {
    margin: 20px 20px 0 20px;
}

/*对页底最后一个模块特殊处理*/
#main [class^="mod"]:last-child,
#main [class*=" mod-"]:last-child {
    margin-bottom: 20px;
}
```

响应式

```
/* 对基类进行响应式布局 */
@media all and (min-width:640px){
    [class^="mod-"],
    [class*=" mod-"] {
        ...
    }
    [class^="mod-cards"],
    [class*=" mod-cards"] {
        ...
    }
}
```

语法提示

```
.debug [class^="mod"] [class^="mod"]:before,
.debug [class*=" mod"] [class*=" mod"]:before {
    display: block;
    content: '模块禁止相互嵌套，请遵循 CSS Mod+ 规范';
    color: red;
}
```

控件层

- 站点基础交互控件，与具体业务无关
- 例如：button、tabs、menu 等

选择器

- 使用 .ui- 开头，可根据需要继续拆分 .icon-、.button 等命名控件。子元素、状态使用长命名避免样式冲突

选择器

```
/*对话框*/
.ui-dialog {...}
.ui-dialog-title {...}
.ui-dialog-content {...}

/*图标*/
[class^="icon-"],
[class*=" icon-"] {
    ...
}
.icon-edit {...}
```

多态

状态与子元素使用长命名

```
.ui-dialog-focus .ui-dialog-title {...}
```

小结

类别	命名空间	状态与子元素	允许嵌套
布局	HTML5 Tags 或 ID		X
模块	.mod-	短命名	X
控件	.ui- .icon- ...	长命名	√

保证规范有效实施

- 公用样式的添加需要由专人负责（团队骨干）
- 公用样式一定需要有规范文档或Demo
- 规范一定是与设计师、产品经理共同制定

感谢