设计系统的 CSS 结构

海淘科技大家提供一个高大上的技术：**设计系统的 CSS 结构。**内容如下：

我们刚刚给一个大型机构搭建了一个设计系统并且创建了一个让我们十分满意的 CSS 架构。这是第一次，我们在最后期限前完成任务，并且没有希望我做一些不同的东西。因此，我认为这是一个非常好的分享机会，告诉大家我们如何搭建系统 CSS 架构的。

给项目一点背景，我们开始搭建一个设计系统和样式指导，为了满足组织中成千上百的开发者，采用大量技术来创建它们超过 500 个内部 web 应用。

组织中的大量的开发者都专注与应用编程，数据和逻辑，不专注与前端开发。因为他们的开发时间很紧张，开发者需要完成他们的 app 并且快速上线，他们经常简单地复制粘贴来自其他应用前端的代码或者使用框架，像用 Bootstrap 来完成 UI 的工作。就像你想的那样，这样积累的结果就是这些行为让，本来就不一致的前端体验变成了大杂烩。当然这就是我们想要拯救的，通过搭建组织结构，他们自己的周到的、稳健的 UI 设计系统。

搭建 CSS 法则

在项目开始的时候，我们谈论了开发者关于他们的流程和痛点，并问他们的接口设计系统如何让他们的工作量变简单。

完成我的前端指导问卷，这些导致一系列前端规则和系统封装。这里有些我们创建的 CSS 具体规则。

- 模块化 —— 设计系统在每一个方面都是模块，这是非常实用写 CSS 的方法。这在组件之间需要清晰分隔。
- 可读性是关键 ——开发者必须在第一眼理解 CSS 代码，并且理解每一个选择器的目的。
- 清晰胜过简洁 —— 设计系统有时候看上去很冗长，但是作为交换，它提供清晰和韧性。保持 CSS 可读性和可扩展性意味着牺牲简洁的语法。
- 保持平面化 —— 长的选择器要回避，无论什么地方，尽可能保持 CSS 独立 DOM 和模块化。
- 避免冲突 —— 因为组件会部署许多不同的应用，保证设计系统之间的 CSS 不会和其他的库和系统有冲突，这很重要。通过系统空间命名 Class 名可以完成这个，更多的会在之后描述。

从这些规则中，我们搭建了制约和语法，包含了这些规则，以满足开发者的需求。这里有一个我们总结出的 class 语法：

全局命名空间

所有的 Class 都和设计系统关联的都以全局命名空间为前缀，这就是公司名称后面加一个连体符

```
`.cn-`
```


如果你工作的 CSS 框架是用于单个网站或者如果你对你的开发环境有绝对控制，那么引入全局命名空间是不需要的。但是如果你的设计系统是混合的技术，那么为系统特定代码创建一个标识是很重要的。作为第

三方开发者，在多个环境中利用他们的系统，营销团队可能会失控，因此 Lightning Design System 引用了相似的方法到他们的系统之中(通过前缀.slds-)，在我们的例子中，许多我们客户的开发者使用 Angular，因此他们已经很熟悉命名空间的概念，因为 Angular 使用 ng-作为命名空间，为 Angular 特殊的代码。

Class 前缀

除了命名空间，我们添加前缀到每个 Class，为了使之更加明显，这个这个 Class 是做什么的。下面是我们使用的类前缀：

- c- 用于 UI 组件,比如.cn-c-card 或.cn-c-header
- l- 用于布局相关样式，比尔.cn-l-grid__item 或.cn-l--two-column
- u- 用于公共部分，比如.cn-u-margin-bottom-double 或.cn-u-margin-bottom-double
- is- 和 has- 用于特定状态，比如.cn-is-active 或 .cn-is-disabled. 适用于这些状态为基础的样式。
- js- 用于目标特定功能，比如.js-modal-trigger. 这些 class 没有绑定样式；他们只是为了行为而保留的. 对于大多数案例，这些 js- 类将在元素上会切换基于状态的类。

我被灌输来自 Harry Roberts 的一个概念，并且一开始在我认为这有道理的同事，我还是持有质疑的态度的，仅仅因为这是额外的字符并且我认为前缀会降低代码可读性。然而我的想法是不对的。在实施类前缀之后，我发现他们对于分清每个类的角色十分有帮助并且对于破译一个应用的代码库十分容易一目了然。对于设计系统用户，这种清晰的代码能够整理清楚头绪，特别有用。

BEM 语法

BEM 代表了"块元素修饰"，这意味着：

- Block 主要组件块，比如.cn-c-card 或者.cn-c-btn
- Element 是主要块的一个子类,比如.cn-c-card__title
- Modifier 是一个组件样式的各种变化，比如.cn-c-alert--error

这种方法论已经很受欢迎了，将这些概念和全局命名空间和类前缀结合在一起，允许我们创造更明显封装的类名。

把它们都放到一起：解剖一个类

全局命名空间的结合，类别前缀，和 BEM 语法引出了一个明确的(是的，冗长的)类字符创，允许开发者们在构造 UI 的时候演绎他在之间扮演的角色。

让我们检查下以下的例子：

```
`.cn-c-btn--secondary`
```

- cn- 是来自设计系统的用于所有样式的全局命名空间。
- c- 是 class 的类别，在案例中，c- 一位置“组件”
- btn 是块名(“Block(块)” 就是 BEM 中的“B”)
- --secondary 是一个修饰成分，指向一个块的变化多端的样式 (“Modifier(修饰)” 就是 BEM 中的“M”)

这里有另一个例子：

```
`.cn-l-grid__item`
```


- cn- 再一次出现就是系统的全局命名空间。
- l- 是类的类别，在这种情况下 l- 意味着“布局”
- grid 是块名
- __item 是一个元素，表明那是块中的一个分支(“Element”在 BEM 中指“E”)

还有一个：

```
`.cn-c-primary-nav__submenu`
```


- cn- 是系统的全局命名空间。
- c- 是类的类别，在这个例子中 c- 意味着“component”
- primary-nav 是块名
- __submenu 是一个元素，指出他是块的子元素 (“Element” 在 BEM 中是“E”)

此外，毫无疑问，这些类比大多数其他方法的类更加冗长，但是对于这种特殊的系统，这些约定很有意义。

其他技巧

明确细节

为了防止代码瓦解，我们详细说明如何处理这么多细小的细节，就像注释、代码块之间的空间距，tab 还是 space 等等。感谢上天，Harry Roberts 已经将一个极佳的综合的资源整合在了一起，称之为 CSS Guidelines，对于这些类型的约定，这个作为我们的底线。我们梳理所有的代码并且标记出我们偏离 Harry 指出地方的计划。

Sass 父选择器

我一直有个关于 CSS 的一个问题，是找出究竟在哪里放一个规定的规则。如果我有一个主要的导航组件，我要把这些样式放在头部还是在部分的主要导航 Sass?谢天谢地，Sass 父元素原则器出现了，这允许我们把所有的组件特定的样式放在一个根元素下：

```
.cn-c-primary-nav {
    /**
     * Nav appearing in header
     * 1) Right-align navigation when it appears in the header
     */
    .cn-c-header & {
        margin-left: auto; /* 1 */
    }
}
```

这意味着，所有的主要导航样式都可以在一个主导航 Sass 部分中找到，而不是将他们分成好几个文件。

Sass 嵌套的明确规则

在 Sass 中嵌套可能十分方便，但是增加了糟糕输出的危险，会有过长的选择器字符创。我们遵循《盗梦空间》规则，嵌套永远不超过 3 层。

牢记设计系统的 CSS 平坦规则，我们希望在下列情况中限制嵌套：

1. 一个样式块修饰
2. 媒体查询
3. 父元素选择器
4. 状态
5. 样式块装饰 对于装饰来说，如果规则只有几行长度，装饰块可以被嵌套在父元素中，就像下面这样：

```
.cn-c-alert {
    border: 1px solid gray;
    color: gray;

    /**
 * 错误弹出
 */
    &--error {
        border-color: red;
        color: red;
    }
}
```

由于&符号，这会编译成：

```
.cn-c-alert {
    border: 1px solid gray;
    color: gray;
}

.cn-c-alert--error {
    border-color: red;
    color: red;
}
```

对于长样式块，我们不会嵌套装饰代码，因为这减少了代码的可读性。

媒体查询器

组件特定媒体查询器能够在组件块中嵌套。

```
.cn-c-primary-nav {
    /* Base styles */

    /**
 * 1) On larger displays, convert to a horizontal list
 */
    @media all and (min-width: 40em) {
        display: flex;
    }
}
```

这个会被编译成：

```
.cn-c-primary-nav {
    /* Base styles */
}

@media all and (min-width: 40em) {
    .cn-c-primary-nav {
        display: flex;
    }
}
```

父元素选择器

设计系统会充分使用 Sass 的父元素选择器原理。这里允许所有的给定组件的规则在一个地方维护。

这会被编译成：

```
.cn-c-primary-nav {
    /**
 * Nav appearing in header
 * 1) Right-align navigation when it appears in the header
 */
    .cn-c-header & {
        margin-left: auto; /* 1 */
    }
}
```


cn-c-primary-nav 所有样式都会在一个地方找到，而不是分散在许多部分文件之中。

1.状态

组件的状态必须包括在一个嵌套的元素之中。这包括了 hover, focus,和 active 状态：

```
.cn-c-btn {
    background: blue;

    &:hover, &:focus {
        background: red;
    }
}
```


这需要编译为：

```
.cn-c-btn {
    background: blue;
}

.cn-c-btn:hover, .cn-c-btn:focus {
    background: red;
}
```


状态同样可以选用通用类的形式，比如 is-和 has-:

```
.cn-c-accordion__panel {
    overflow: hidden;
    max-height: 0;

    &.cn-is-active {
        max-height: 40em;
    }
}
```


再者会被编译成：

```
.cn-c-accordion__panel {
    overflow: hidden;
  max-height: 0;
}

.cn-c-accordion__panel.cn-is-active {
    max-height: 40em;
}
```


为了创建一个坚固的系统，将这些规则都放入一个地方中，给我们需要坚持的一些制约和规定。当我们遇到一些规定不是很明显或者有多重解决方案的情况下，我们需要一次谈话，讨论如何处理这些问题，如果需要的话可以更新方针。

这对每个人都用嘛?

在你开始沉迷于，困惑以及开始不同意一些我们在创建系统的时候的决定，记住这个结构对我们正在运行的系统很重要。这是否意味着，这是对任何项目都坚不可摧的解决方案?不，我不是提议一定用那个。特定需求和组织设置，对设计 CSS 架构的系统有足够的影响。

我为很多项目工作过，在哪里我可以用类似于.table-of-contents li a 这样的字符串对付过去，但是这些项目大多数都是由我管理的。对于设计一个团队环境的户项目，我倾向于冗长，明确的语法，就像我上面描述的那样，因为他们不给人们搞砸的空间。看看其他团队像 Sparkbox 得出的类似的结论，这是非常赞的。

距离项目已经过去几个礼拜了，我们重新在 1.1 版本上的设计系统继续工作。我希望回到这个代码库，并且能够看到我是如何速度地重新适应它的!