# Style Guide
# 调研报告

侯鸿儒
2013.12.27

# 目录

# 1. 现存问题

目前 Web app 的设计开发流程为：在 PM 提出产品需求后，由 UE 产出设计稿，然后交由负责相应项目的 RD 来实现。这样的流程看似各种岗位各司其职，但其实在无形中带来了以下问题：首先，由于项目进行的批次不同以及开发人员开发能力、审美水平不同，带来了样式的不统一，从而导致了不一致的用户体验，例如线上导航栏中左右的两个按钮宽度分别为 54px 和 51px；其次，对于相同样式的重复性开发也降低了团队的工作效率；此外，上述冗余的代码不仅影响了页面的加载速度等性能指标，而且也不利于后期维护。

# 2. 目标

为了解决上述问题，我们需要设计、开发并维护一整套针对 Web app 的 style guide 来帮助我们提高工作效率和产品质量。从上述需求出发，针对 Web app 的 style guide 需要满足以下需求：包含所有 Web app 内所用到的元素及组件的样式，兼容性方面需要覆盖 QA 处所要求的目标机型及浏览器，并且具有可重用、易维护、专注于移动端的特点。

# 3. Style Guide 简介

事实上style guide 有很多种，作为一种 reference 形式的文档性资料，style guide 可以用在各种不同的领域中来组织信息。此处对于 style guide 做一简要分类介绍，以增进大家对于 style guide 的理解以及后续的使用。

## 3.1. 内容 style guide

内容 style guide 主要用于规范文字表述时的各种注意事项。目前最著名的内容 style guide 是由邮件分发服务提供商 MailChimp 所组织并撰写的 Voice & Tone。该文档对于如何为 MailChimp 公司撰写内容提出了详细而具体的指导方针，并让文档撰写者意识到文章中的措辞、语调对于用户体验所产生的重要影响。因此，内容 style guide 是一种为编辑人员设计并使用的 style guide。

## 3.2. 品牌 style guide

目前从数量上来看，品牌 style guide 是网络上最多的一种 style guide。每个品牌 style guide 大多都会包含logo等设计方案，并对其使用方法做出详细规定，例如 logo 周围需要有至少多少的margin或出血等等。很多知名品牌都有自己的品牌 style guide，甚至连 UCLA 这样的大学都有自己的 UCLA style guide。品牌 style guide 是一种为设计师设计并使用的 style guide。

## 3.3. 界面 style guide

界面 style guide 是一种由系统或平台提供商为了保持其生态系统或平台上产品使用体验的高质量而撰写的一种指导性 style guide。例如 Apple 的 iOS Human Interface Guidelines、Google 的 Android Design 以及 Microsoft 的 Windows Phone Dev Center 等

都属于界面 style guide 的范畴。此类 style guide 也是一种为设计师设计并使用的 style guide。

## 3.4. 开发 style guide

开发 style guide 即我们常说的开发规范文档，由于开发工作常常需要涉及到多人次开发，因而保持代码风格的一致性，便在项目的持续集成及维护等阶段显露出其价值。以 Google 的 JavaScript Style Guide 为例，其中不仅规定了正确的代码风格，而且多数规定在给出正面示例的时候也给出了反面示例。开发 style guide 的设计使用对象是各类开发人员。

## 3.5. 前端 style guide

前端 style guide 是一种作用及目标受众均介于界面 style guide 和开发 style guide 之间的文档类型。前端开发工程师所处的特殊位置决定了其需要兼顾设计与开发，因此前端 style guide 中不仅包含有设计方案（如按钮等），还会有该方案的代码实现及其调用方法。对于一些更加全面的前端 style guide 而言，其中还会包括应该在什么情况下使用所描述的元素或组件样式的文字信息。

相比其他类型的设计 style guide 而言，前端 style guide 的特点在于专门针对 Web 所设计和建立。此外，为了使得维护更加容易，前端 style guide 都是以真实的代码运行在浏览器中，而不是输出成为PDF供人们参阅。在产品不断迭代的过程中，前端 style guide 也需要随着产品的每一次更新、改版等进行修订，以保证和产品保持一直。

# 4. 业内前端SG调研

由于业内各种前端SG的侧重点各不相同，而且有的属于CSS style guide，有的属于框架中的一部分，本文中只选取了笔者认为的和我们的需求较为一致的三个 style guide 进行调研性介绍。它们分别是 Github CSS Style Guide、Bootstrap 和 Pear，本文会从分组方式、调用方式、以及实现方式三方面对以上三个调研对象进行描述。

## 4.1. Github CSS Style Guide

笔者在调研阶段看到的各种相关 style guide 中，Github 的 style guide 算是最开放也最具参考价值的一个。Github CSS SG 通过凭借与 SCSS 和 KSS 的结合，使得在文档的维护、使用等方面都带来了较大的便利。而其该文档在已开始对于 CSS 的写法提出了建议、明确了 px 和 em 的使用情况、统一了“元素名+类名”的主要命名方式，可谓描述得较为详细而全面。

按照其网站自身需求出发，Github CSS Style Guide 中的分组如下：

- Buttons
- Forms
- Source Code
- Text Styling

- Listings
- Boxed Groups
- Icons
- Navigation
- Behavior
- Discussion
- Colors
- Animations
- Select Menu
- Blank slate
- Alerts
- Pagehead
- Repo selector

从以上分组情况来看，Github CSS Style Guide 的分组逻辑性并不强，感觉较为混乱。

在调用方式方面，主要通过为 HTML 添加相应的 class 来实现相应样式的调用，而且除非能够确定元素或组件在页面内具有唯一性（例如 <header>、<footer>）时可以用 id 来调用，否则一律用 class 来调用。

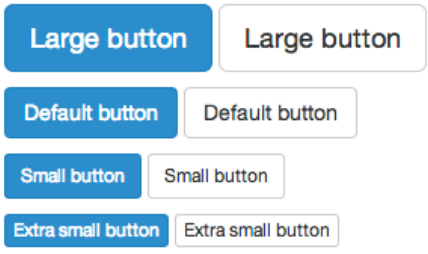
在实现方面，除了前面所提到的以“元素名+类名”为 CSS 样式命名外，值得一提的是其对于 icon 的处理：Github 采用了将常用的单色 icon 制作为 icon font 来引入页面的做法。这种做法所带来的好处在于可以帮助 Github 加速页面加载并带来统一的用户体验，因为矢量的字体能够在 Retina 屏幕和普通屏幕上都高质量地显示。下图展示了普通图片和 icon font 在页面放大后的展示效果。


## 4.2. Bootstrap

作为目前最为流行的前端框架，Bootstrap 是本次选取的三个示例中最为大家所熟知的一个，同时也是组件分类最多的一个，并且每个大类下还有若干小类，其组件分类如下：

- Glyphicons（即 icon font）
- Dropdowns
- Button groups
- Button dropdowns
- Input group
- Navs
- Navbar
- Breadcrumbs
- Pagination
- Labels
- Badges
- Jumbotron
- Page header
- Thumbnails
- Alerts
- Progress bars
- Media object
- List group
- Panels
- Wells

EXAMPLE




```
<p>
  <button type="button" class="btn btn-primary btn-lg">Large button</button>
  <button type="button" class="btn btn-default btn-lg">Large button</button>
</p>
<p>
  <button type="button" class="btn btn-primary">Default button</button>
  <button type="button" class="btn btn-default">Default button</button>
</p>
<p>
  <button type="button" class="btn btn-primary btn-sm">Small button</button>
  <button type="button" class="btn btn-default btn-sm">Small button</button>
</p>
<p>
  <button type="button" class="btn btn-primary btn-xs">Extra small button</button>
  <button type="button" class="btn btn-default btn-xs">Extra small button</button>
</p>
```

在如何调用组件方面，Bootstrap 使用相同开头、不同结尾的 class 来指定同一组件内的不同属性。例如，按钮组件分别使用 btn、btn-primary、btn-lg 来表示“按钮组件”、“蓝色按钮”和“大型按钮”，其他样式的按钮以此类推。并且所有的 CSS 挂钩都使用 class，没有使用 id 的情形。

在实现方面，Bootstrap 使用了 LESS 来作为 CSS 的预处理器，以增加代码的可维护性。另外，Bootstrap 使用了 Normalize 作为自己的的 CSS reset，并使用 Media queries 针对不同大小的屏幕来做不同效果的显示，满足了其跨设备的需求。在文件的组织管理方面，Bootstrap 虽然将不同组件的 LESS 文件进行了区分，但是并没有做分组，而是全都方正了一个文件夹下，不如 Github 的有序。

## 4.3. Pears

Pears 是一个开源的 WordPress 主题，为了方便不同的开发者开发自己的主题，Pears 事先设定好了一些通用样式（common patterns），而这样的一组通用样式恰恰成为了绝佳的前端 style guide。更加令人喜出望外的是，Pears 还在每种组件样式的页面内同时给出了该组件的实际样式、HTML 代码和 CSS 代码，从而也满足了前端 style guide 的基本要求。

分组方面，由于主要服务于 WordPress，因此其组件也主要以博客中的常见组件来分组，其分组如下所示：

- Content
  - Article
  - Data Table
  - Footer
  - Slats（即我们说的 card）
- Form
  - Multi
  - Search
  - Simple
- Lists
  - Definition
  - Stats Tabs
  - Tags
- Navigation
  - Breadcrumb
  - Horizontal Buttons
  - Horizontal Links
  - Horizontal Tabs

- Pagination
- Vertical Links

在调用和实现方面，Pears 也主要以使用 class 为主，没有之用 id，但出现了 role 属性的使用案例。Role 属性是 W3C 的一种新的规范，用于在任意的标记性语言中嵌入机器可识别的信息的方法，其用途和 data-* 属性类似。例如，在 Footer 这一组件中，Footer 的 HTML 如下（注意 footer 元素的 role 属性及其在 CSS 中的使用）：

```
<footer role="contentinfo">
  <nav role="navigation">
    <ul>
      <li><a href="#">This is a link</a></li>
      <li><a href="#">Active Link</a></li>
      <li><a href="#">Lorem link</a></li>
      <li><a href="#">Dolor link two</a></li>
      <li><a href="#">Here is a link</a></li>
    </ul>
  </nav>
  <p>Copyright © 2012 Company Co. All rights reserved.</p>
</footer>
```

其 CSS 如下：

```
footer[role="contentinfo"] {
  padding: 15px 0 0 0;
  font-size: 14px;
  border-top: 1px solid #ddd;
  }
footer[role="contentinfo"] nav ul li {
  display: inline;
  margin: 0 10px 0 0;
  padding: 0 10px 0 0;
  font-weight: bold;
  border-right: 1px solid #ccc;
  }
footer[role="contentinfo"] nav ul li:last-child {
  margin: 0;
  padding: 0;
  border: none;
  }
footer[role="contentinfo"] p {
  margin: 10px 0;
  }
```

# 5. Web app Style Guide 设计

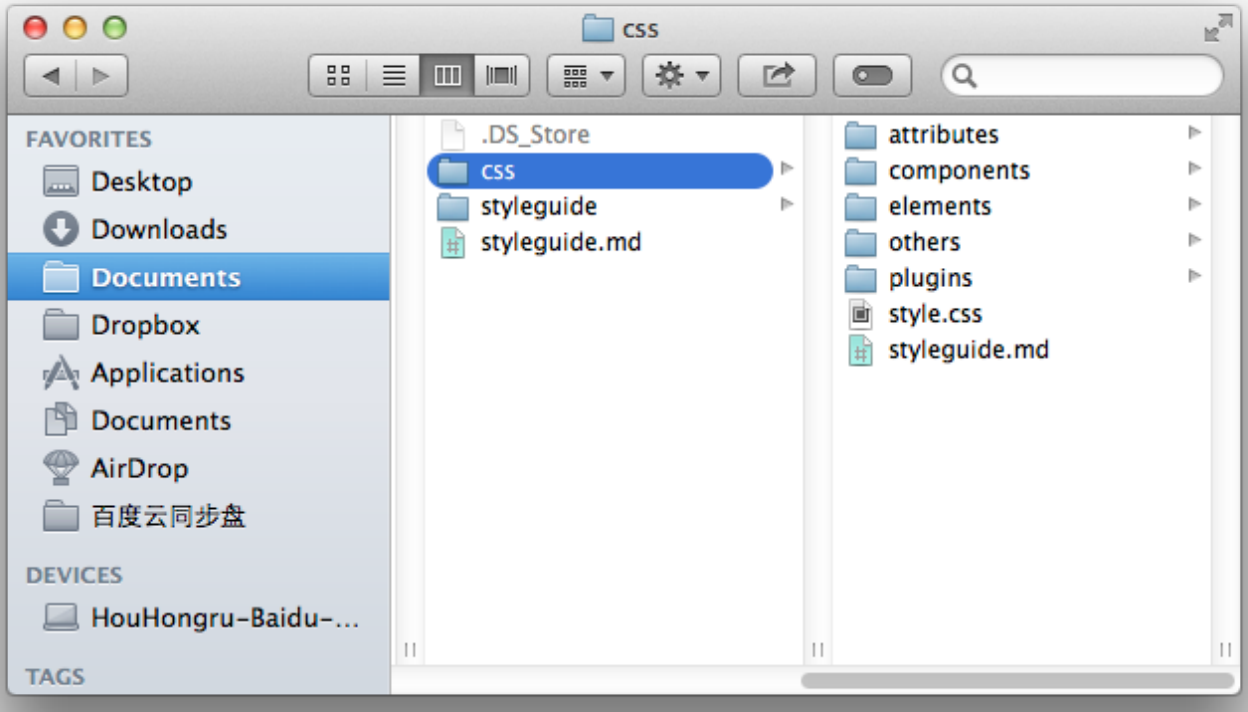
从以上所分析展示的三个不同用途、不同题量的前端 style guide 来看，在如何为元素及组件分组方面，大家差异明显，因此并没有业内通用分类方法值得借鉴，在建立我们自己的 style guide 时需要针对自家产品的实际需求来进行设计构建。笔者目前比较倾向于根据 CSS 的层级来分大组，然后再细分为小组。

首先在属性级别，定义产品内需要用到的所有颜色和字体，因为几乎所有 UI 元素或组件都要调用这两个常见属性，因此集中定义这些通用属性有利于调用及后期维护；其次定义经常以元素形态独当一面的各类元素，包括 h1 到 h6 标签及其链接形式、p 标签等等；然后定义由元素组成的常常以组件形式出现的组件类，包括搜索框、导航栏等待；最后定义其他杂项，例如 icon、图片等等。具体分类如下所示：

| Web app Style Guide 初步分类设计 | |
|---|---|
| 属性类 | color、font-family |
| 元素类 | h1-h6 (link)、p、button、form、table、ol、ul、dl |
| 组件类 | search box、navigation bar、pop window、banner、card |
| 其他类 | icon、image、grid |

对于其中的每个元素或组件，都要给出 UI 效果、HTML 结构、相应的 CSS 样式以及在什么情况下调用是合适的。这里列出的仅仅是暂时的计划项，具体实施时会涉及到哪些内容还需要经过对于 Web app 的 UI 统计后才能最终确定。

文件结构组织方面，为了方便后面扩展，相应的 CSS（或 SCSS 文件）会根据上面的分组情况放在不同的文件夹内，此外还需要针对 CSS reset 等这样的附属文件建立一个 plugins 文件夹。最终的完整的样式表位于与这些文件夹同级别的名为 css 的文件夹内，完整的文件夹结构如下所示：



调用和实现方面，采用以 class 为主、元素名 + class 为辅的 CSS 挂钩设计策略，对于含有特定元素才有的属性，其选择器中要加上特定的元素名。另外要尽可能地遵循 HTML 元素的语义，例如按钮尽可能用 button 而不是 div、有语义上的分割用 section 而不是 div 等等。在 CSS 预处理器方面倾向于使用 SCSS，其他 workflow 需要在后面实际实施过程中不断优化。

# 6. 实施

## 6.1. 过程

为了进一步根据实际的 Web app 确定需要纳入 style guide 的属性、元素、组件等，在开始着手建立 style guide 之前，需要先对 Web app 中的界面相关内容进行梳理。此处需要 UE 提供原始的交互及界面设计稿，并将其打印出来对各个部分的重要属性进行标注，设计稿里未明确写明的需要从现有代码中提取。最后与 PM 同学确定所提取的属性、元素、组件等内容是否正确，若正确则进入下一步。

在确定了要放入 style guide 的内容和值之后，就要进入实质性建立 style guide 的阶段了。首先从构建核心属性开始，例如把所有用到的颜色、字体等都一一列出，后面所有用到这些属性的地方，其值必需在此处列有的值当中。在完成属性类的 style guide 之后，就依次在前面内容的基础上，进行元素类、组件类、其他类的整理和开发实现。在完成所有内容的开发整理之后，需要对所完成的 style guide 进行优化、美化等工作，以提高其使用体验。

## 6.2. 工具平台

在前期调研过程以及自己的搜集过程中，发现了以下三种工具可以用来辅助搭建 style guide：

- Style Guide Boilerplate（Demo）
- Pattern Lab（Demo）
- Kalei Style Guide（Demo）
- KSS-node

通过逐一使用各个工具平台，最终从易用性、易维护性、美观性等方面综合考虑，较为倾向于使用 Github 所使用的 KSS，但使用的是其 node 版本的实现而不是 Github 所使用的 Ruby 版本。

# 7. 预期收益

使用 style guide 的预期收益除了本文开篇处所提到的提升用户体验、减少冗余工作和代码外，还会在其他方便带来好处，主要预期收益如下：

- 增加样式的一致性，带来用户体验的提升
- 减少重复性开发，提高工作效率
- 减少代码冗余，提高页面性能

- QA 更容易测试
- 便于设计师和开发者使用统一的语言交流
- 便于 PM 和 UE 在设计阶段查阅

由以上预期收益可以看到，建立 style guide 的受益者不仅是 FE、RD，也包括产品线内的其他角色。

# 8. 项目计划及成本

根据前面所提到的实施过程，短期计划可以以前期统计、四大类的完成以及最终的优化工作作为6个里程碑来进行计划和成本估算，而中长期计划估计较为粗略，具体情况如下表所示：

| | 工作 | 时间计划 | 人数 |
|---|---|---|---|
| 短期计划 | 统计阶段 | 3天 | 1 |
| | 属性类实现 | 2天 | 1 |
| | 元素类实现 | 5天 | 1 |
| | 组件类实现 | 2周 | 1 |
| | 其他类实现 | 3周 | 1 |
| | 整体优化 | 1周 | 1 |
| 中期计划 | 与线上整合 | 1月 | 2~3 |
| | 融入现有工作流程 | 2014 Q1 | |
| 长期计划 | 加入交互性组件 | 2014 Q2 | |
| | 推广到 Web map | 2014 Q3 | |

# 9. 总结

本文通过对 style guide 的类型介绍、现有前端 style guide 分析调研学习、提出针对 Web app style guide 的设计方案并最终给出其预计收益及成本估计，完成了对于建立前端 style guide 的一套完整计划。总而言之，建立 style guide 的受益者涵盖了产品线上的所有角色，而且为工程师和设计师架起了一道桥梁，同时为产品的设计与实现增加了一道审计步骤，以保证产品的质量。需要特别强调的一点是，建立 style guide 并不难，难的是在后面的产品迭代过程中如何持续维护 style guide，并将其融入现有的工作流程当中，使之成为所有角色的一个中央 hub 来连接彼此的工作。