时尚且健壮：实现更优秀的 CSS

作者 Rouan Wilsenach ，译者 邵思华 发布于 2015 年 12 月 24 日 | 讨论

让我们正视一个现实的问题吧：所有的网站都需要用到 CSS，而你的 CSS 代码却很可能写得很糟糕。代码过于臃肿，并且出现了大量的重复。它就像是一张精密的蜘蛛网，紧紧地交织在一起，并且非常脆弱。代码中的补丁比自行车轮胎上的还要多。其实 CSS 代码原本不必变得这么糟糕，我将在本文中为你介绍基于组件的样式设计的概念，并为你展示如何为你的网站创建一份样式指南。

为何写好 CSS 这么难？

> 我恨 CSS，我只想写代码。
>
> - 一个孤独的开发者在绞尽脑汁才把 `margin-right` 的问题搞定之后的心声。

这种感受是不是似曾相识？以下是一些常见的 CSS 代码坏味道，或许你也曾经遇到过

哇，怎么这么多 CSS。你知道我所描述的文件是什么样的，文件中的 CSS 代码足有上千行，在其中进行浏览十分困难。
在不必要的地方编写 CSS。你所参与的网站并非全新的项目，在其他各个页面中已经存在了各种按钮、表格与列表。尽管如此，一旦要开始开发某个新特性，你仍然不得不编写大量的 CSS 代码，才能让这个页面中的元素与网站中其他页面中的元素看上去相同
无法重用。你的代码中是否也有过类似于 `user-table` 和 `transaction-table` 的类，他们的样式各不相同？你是否希望能够有一个单一的表格类，让以上两者都能够加以利用？
每个特性都是独特的。让两个表格都使用同一个类并不难，但无法直接进行复制/粘贴。每个表格都需要一些独特的调整。整个网站缺乏一致性，每个元素都有些调整过度了。
副作用。感谢上帝，你的页面看起来终于能够正常显示了，这种感觉非常好。但很快你就发现网站中的其他部分出现了一些奇怪的问题。一旦你修改了某个部分，总是会产生某种连锁反应，破坏其他某部分的功能。

大可不必如此编写 CSS

有两条原则能够帮助你在设计网站的样式时保持健壮。第一条原则是：你应当**将 CSS 当做代码一样看待。**CSS 是一门最易于被忽略的编程语言。只因为我们认为保持 CSS 代码的可维护性过于困难，因此我们甚至放弃了尝试。如果你在 CSS 代码中应用了与其他语言相同的代码实践，你将发现编写 CSS 也不是那么糟糕的事。

第二条原则是：你应当**以使用者的角度来思考你的样式。**因为我们在设计网站的样式时从没有考虑过重用，因此我们的样式自然无法成为可重用的。一旦为新的功能编写代码时，我们会发现很难了解什么是正确的实现方式，甚至不知道最终的产品看起来应当是什么样的。如果你在编写 API 代码同时去实际地调用它，要理解并使用这段代码就容易得多。同样，如果以一种适于重用的方式设计 CSS 代码的结构并展现它，那么你将能够更容易地从之前已经完成的工作中受益。
我将带你进行一次我称之为**基于组件的样式设计**的实践，并为你展示这种实践将如何帮助你将 CSS 视为代码、以使用者的角度来思考你的样式、并使你的网站中的样式更易于理解。

样式指南介绍

基于组件的样式设计方法的一个核心实践就是为你的网站创建一份样式指南。那么什么是样式指南呢？让我们来看一个类似的示例吧。

如果你曾经使用过某种样式设计框架，例如 Bootstrap 或 Foundation，你就会意识到这一概念。假设你打算使用由 Twitter 或 Zurb 的杰出工程师们所创建的以上两种框架，并且利用其中某个漂亮的组件。举个例子，你访问了 Bootstrap 的网站并到处浏览，最终你找到了你所需要的组件，即某个错误警告框。

你所做的第一件事是将他们的 CSS 文件包含在你的网站中。随后你打开那个需要显示警告框的 HTML 文件，并将你在框架的示例中所找到的代码段中的标签结构应用到文件中。

```
<div>This is my scary alert!</div>
```

这段示例非常简单，你所要做的只是将消息文本放到一个 标签里。接下来，你还要添加一些类，正如框架中自带的示例一样。

```
<div class="alert alert-danger">This is my scary alert!</div>
```

搞定！现在你可以在浏览器中打开这个页面了，你将发现你的警告框组件的样式与 Bootstrap 网站中的示例完全一样。

你刚刚所完成的任务有何特别之处呢？你在没有编写一行 CSS 的情况下就为你的网站中的某个组件添加了样式。你所做的只是添加了一些标签，复制了一些类而已，这就足以让你的组件看上去像一个警告框一样。现在请想象一下，你是否能够让你自己的网站中的样式也实现同样的功能呢？

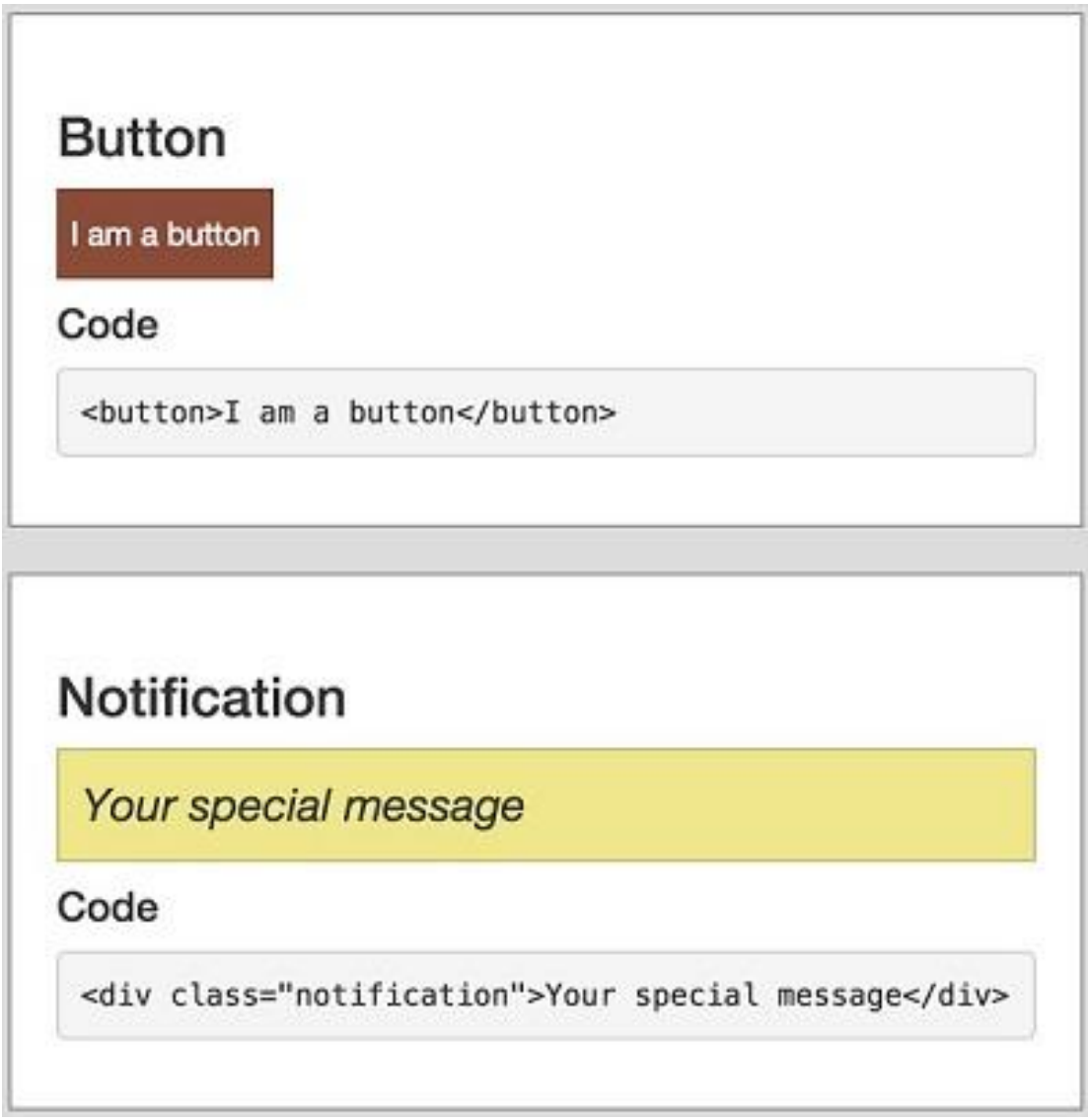
创建你自己的样式指南

示例代码
演示
（演示页面可能不够漂亮，但它确实能够验证这一概念）

如果在网站中设计新页面的样式也能这么方便，是不是很美妙？这一点是完全可以实现的，你需要做的就是创建属于自己的样式指南。

启动这一工作的最简单方式就是在你的项目根目录下新建一个 `style-guide.html` 文件，在其中包含整个网站中的 CSS。如果你已经对 CSS 实现了合并与最小化操作，这一点应当很容易。

下一步是开始为你自己的组件添加示例。组件应当保持简单，就像一个警告框或一个按钮一样：



组件也可以以特定于你的网站或业务领域的形式存在，例如这个用于显示产品的显示框：

Product Box

A title for your item

A description of what this is and why someone would want to buy it.

Add to cart

Code

```
<div class="product-box">
<div class="item-title">A title for your ite
m</div>
<p>
<img src="images/duffel_bag.jpg">
A description of what this is and why someone
would want to buy it.
</p>
<button>Add to cart</button>
</div>
```

请注意：每个组件的介绍都要包含其外观的一个示例，以及创建该组件所需的 HTML 代码。

让我们来看一看样式指南本身的代码。每个组件单独成一节，以一个示例突出这一节。由于你已经在指南中引入了网站中的所有 CSS，因此这些 CSS 也将为你的示例添加样式。

```
<div class="component">
<h3>Button</h3>
        <div class="demo">
            <button>I am a button</button>
        </div>
</div>
```

我们在这里可以使用一些简短的 JavaScript 代码，让它自动找到所有.demo 元素，随后将渲染这一组件所需的代码加入一个显示框中。通过这种方式，你就能够避免重复编写 HTML。另一个好处在于，我们所看到的某个组件的代码与实际用于渲染该组件的代码总是同步的。在我看来，这就是活文档的一种非常实用的形式。

样式指南真正的魔力在于它使用了与你的网站相同的 CSS，它为你的网站中可用的各种样式提供了一个准确的摘要。这一点在设计与开发过程中非常实用，它为你展示了能够选择的各种选项，并且专门为你体验网站的外观提供了一个地方。

拥有一份活文档是非常实用的，如果你修改了 CSS，那么样式指南与实际的网站会同时得到更新。因为他们共享了同样的 CSS，因此组件本身与组件的文档也是相同的。如果你所更新的组件在样式指南中看上去是正常的，但在实际的网站中却出现了问题，那就意味着你的 HTML 代码并不一致。而如果你的网站看上去正常，而样式指南中却出了错，就意味着你的组件的可重用性有所欠缺，你必需对你的 CSS 进行修正。

如果你的网站是全新设计的，那么你在创建新组件的同时就可以将他们加入样式指南中。但即使你是在维护一个现有的网站，也无需因此感到不安。如果你要为某个现有的网站创建一份样式指南，你可以在需要使用某些组件时相应地更新样式指南中的内容。在实际工作中，你还可以对 CSS 进行一些重构工作，让他们更符合基于组件的模式。随着时间的推移，你将不断改善你的 CSS，使你网站中的组件更加一致。

提示：创建你自己的样式指南并不意味着你就不能够使用 Foundation 或 Bootstrap 这样现成的框架了。你可以照原样包含他们的样式表，然后通过结合他们的样式与你自己设计的样式，创建属于你自己的组件。

让你的 CSS 保持 DRY

在上文中，我们已经涵盖了基于组件的样式设计的概念，以及一些如何着手为你的网站创建一份样式指南的提示。但你可能还有些疑惑，具体要怎样编写这些代码呢？让我们看一看一些 CSS 编码实践，他们将帮助你创建你的样式指南，并让你的 CSS 保持 DRY。

CSS 的第一原则就是：不要编写任何 CSS。

CSS 本身是一门已经过时的语言，其特性对于保持你的代码整洁没有提供多少帮助。因此，不要手写 CSS 代码，而应当使用类似于 Sass 或 Less 这样的语言，他们会为你提供一些额外的语言特性，并将代码编译为 CSS，以使浏览器能够理解。这些语言提供了例如变量或函数等特性，使你编写样式的方式更类似于真正的代码编写方式，从而使你专注于可维护性。

只对每个组件进行一次样式设计

你的每个组件的 CSS 都应当是易于查找及修改的，这一点十分重要。为了实现这一点，我们要确保某个特定组件的所有代码都放置在一个单一的地方。对于每个组件，你应当创建一个专门的文件（例如_button.scss、_table.scss）。在这一文件内，你应当为你的组件设计一个类或元素，在其中嵌入所有相关联的样式。我喜欢将所有这些文件都旋转在一个 components 目录中。在下文的示例中，你将看到产品框这一组件所对应的 Sass 代码，以及它对按钮的样式所做的小改动。

```
.product-box {
  background-color: white;
  border: 0.2rem solid $grunge_grey;
  padding: 1rem;
  min-height: 24rem;
  margin: 2rem 0;

  button {
    margin-top: 2rem;
    width: 100%;
  }
}
```

一致的标签

还记得那个 Bootstrap 的示例吧。我们仅仅复制了一些 HTML，而没有编写任何 CSS，就让组件的外观与我们所期望的一致了。要实现这一点，HTML 必须是一致的。如果你的产品框在某个地方使用了 标签，而在另外某个地方又使用了标签，那你永远也

达不到你所期望的一致性。你将不得不编写额外的 CSS，以补偿 HTML 的区别。这一点怎样强调都不过分：**编写一致性更好的 HTML 就意味着能够减少 CSS 的数量**，从而可以减少你的烦恼。

限定特定样式的范围

尽管基于组件的样式设计能够帮助我们尽量减少编写特定的 CSS 代码，但你仍然不可避免地要编写一些特定的 CSS。总是存在一些情况，某个特别页面中的某个组件需要一些额外的 padding，或是要求使用不同的对方方式。这里的要点在于：如果你的 CSS 必须要迎合这一场景的需求，**那么就让这些代码只在这种场景中生效，而在其他场景中将自动忽略**。我们将利用 SCSS 的功能以实现这一点，将这些特定的样式内嵌在某个类中，而这个类仅与需要这个特定组件的页面相关联。举例来说，我们希望网站的付款页面中的按钮变为正常大小的两倍，这样能够让用户更有购物欲望，你懂的。我们想要通过编写 CSS 实现这一点，不过，我们首先要为该页面中的 HTML 添加一个新的类。

```
<div class="checkout-page">
<!-- … -->
    <button>Pay now</button>
<!-- … -->
</div>
```

随后我们就能够使用这个 checkout-page 类为我们的 SCSS 限定范围了，让这段样式仅适用于这一段 HTML。

```
.checkout-page {
    button {
        height: 6rem;
        font-size: 2rem;
    }
}
```

这种方式能够帮助我们确保在这个范围内的任何代码都不会影响网站的其余部分，也不会影响按钮在其他部分的行为。

以上所列举的代码实际上提出了一个有趣的问题，即我们是否应当创建一个全新的组件？你可以提出异议，认为应当创建一个 big-button 组件，让我们日后能够重用这个组件，而不是让这段代码只应用于付款页面。其实你当然可以这么做，至于你是否应当这么做，这要取决于你是否打算在今后添加更多的大按钮了。

批判性地看待每一行 CSS

CSS 语言存在一个有趣的现象：虽然这门语言非常容易上手，却很少有开发者能够完全掌握它。对于我们中的大多数人而言，编写 CSS 就是一个不断尝试与出错的过程，这一状况今后也不大会发生什么改变。为了尝试让某些组件的外观正确地显示，在此过程中你会尝试各种各样的代码，以试图实现你所期望的效果。要点在于，当完成代码的编写后，你应当重新审视你的代码。你可以试着删除你所编写的每一行 CSS 代码，并检查这一行代码有没有什么实际的作用。你是否真的需要 `overflow: hidden`，将 `display` 属性改为 `inline` 有没有实际意义？如果答案是否定的，那就删除它。你所留下的代码越少，等到你下一次回顾这些代码时，它们为你所造成的困扰就越少。

保持简单

对于网站的样式设计来说，越简单就越好。HTML 是一种非常简单的技术，它只需要一些嵌套的元素，从上至下，从左至右进行铺排。你应当尽量不要过多地违背 Internet 的本义，尽量不要过于追求酷炫的元素组织方式。你的标签与页面弄得越复杂，样式的设计就越困难，尤其是对于移动设备来说更为明显。

我们的客户会在移动设备上提出一些视觉上的修改，其中的差异往往一眼看上去并不复杂。例如：“是否能够在移动设备上让这张图片显示在标题的下方，而不是上方？”许多开发者与设计师在这种场合下都会错误地回答：”没问题“，而接下来的整个星期都在与 CSS 做斗争，以实现客户的需求。在这种情况下，我通常会发现进行一次对话会非常有帮助。在对话中解释这种改动的复杂度，以及你认为需要额外投入多少时间实现这一需求。我经常听到客户会这样回应：“噢，我没有意识到这一点会这么复杂，那就照最简单的方式来做就好了，然后直接去做第 X 号特性吧。”

这里的要点在于让那些非技术背景的项目干系人了解这些信息。让他们理解 web 是怎样工作的、以及阻碍最少的实现方式是什么。这将对他们的日常决策起到极大的作用。

移动优先

试图让一个现有的桌面网站在移动设备上具有同样良好的显示效果，这是一项艰巨的任务，你应当尽可能避免这种情形的出现。移动设备无处不在，用户则期望网站能够在他们的手机上同样可用。我最讨厌的一种情形是在我的手机上打开某个餐厅的菜单时，不得不忍受翻阅杂志式的糟糕的动画效果，或是提示我缺少了某种插件的情形。而事实上，只需一个简单的文本列表项，在每一项旁边标明价格就足以满足我的需求了。

当你着手设计一个全新的网站（或是某个现存网站上的新页面）时，应当以一种恰当的方式组合你的标签与 CSS，让页面能够在移动设备上也能够正确地渲染。这是你首先要做的事。接下来，如果页面能够正确显示，并且看上去效果还不错，那么你就可以添加一些媒

体查询，在较大的屏幕尺寸上修饰你的页面。让各个元素之间具有更多的空间，使他们横向地展开，再添加一些对于移动设备来说过大的图片。通过这种方式，你的用户在多个设备之间能够保持相同的核心体验，你的 CSS 也会变得简单许多。以我的经验来看，让一个适应移动设备的页面在桌面上显示良好，比起相反的要求，所需的代码要少很多。

CSS 越少，困难也越少

我们以上所涵盖的技术有一个基本的前提，即较少的代码与较多的代码相比更易于开展工作。只要你能够以对待其他代码同样的方式处理你的 CSS 与 HTML（并通过使用那些支持这一点的工具实现），你就能够减少重复，你所创建的代码库也更容易理解与使用。一旦你采取了基于组件的样式设计方法，你实际上就在促使你自己，让你的样式更易于重用。我建议你尝试一下这些技术，仔细思考你的 CSS 是如何组织的，并且建立一份样式指南。请告诉我这一切进行得是否顺利，希望它能够为你节省一些时间，使你打开某个 CSS 文件时不会再感到毛骨悚然。

关于作者

Rouan Wilsenach 是一位任职于 ThoughtWorks 的软件开发者。他在为客户进行顾问工作方面具有丰富的经验，其范围包括商业服务、医疗、媒体以及教育部门等等。他很乐于帮助软件团队找到创建更优秀软件的方法。

查看英文原文：Stylish and Sane: A Guide to Better CSS

URL：http://www.infoq.com/cn/articles/guide-to-better-css