CSS进阶：提高你前端水平的 4 个技巧

2017-05-17 前端大全

（点击上方公众号，可快速关注）

> 编译：伯乐在线/小谢
> 如有好文章投稿，请点击 → 这里了解详情

译者注：随着 Node.js 、react-native 等技术的不断出现，和互联网行业的创业的层出不穷，了解些前端知识，成为全栈攻城师，快速的产出原型，展示你的创意，对程序员，尤其是在创业的程序员来说，越来越重要，下面我们就跟随著名国外开发者网站上的热推文章《Leveling up in CSS》，从提升你的CSS技巧开始。

CSS 在刚开始学习的时候看起来非常简单。毕竟，它仅仅就是些样式而已，事实上是这样吗？

但是，随着你的不断了解。很快，你会发现 CSS 没你想象的那么简单，它复杂且有深度。

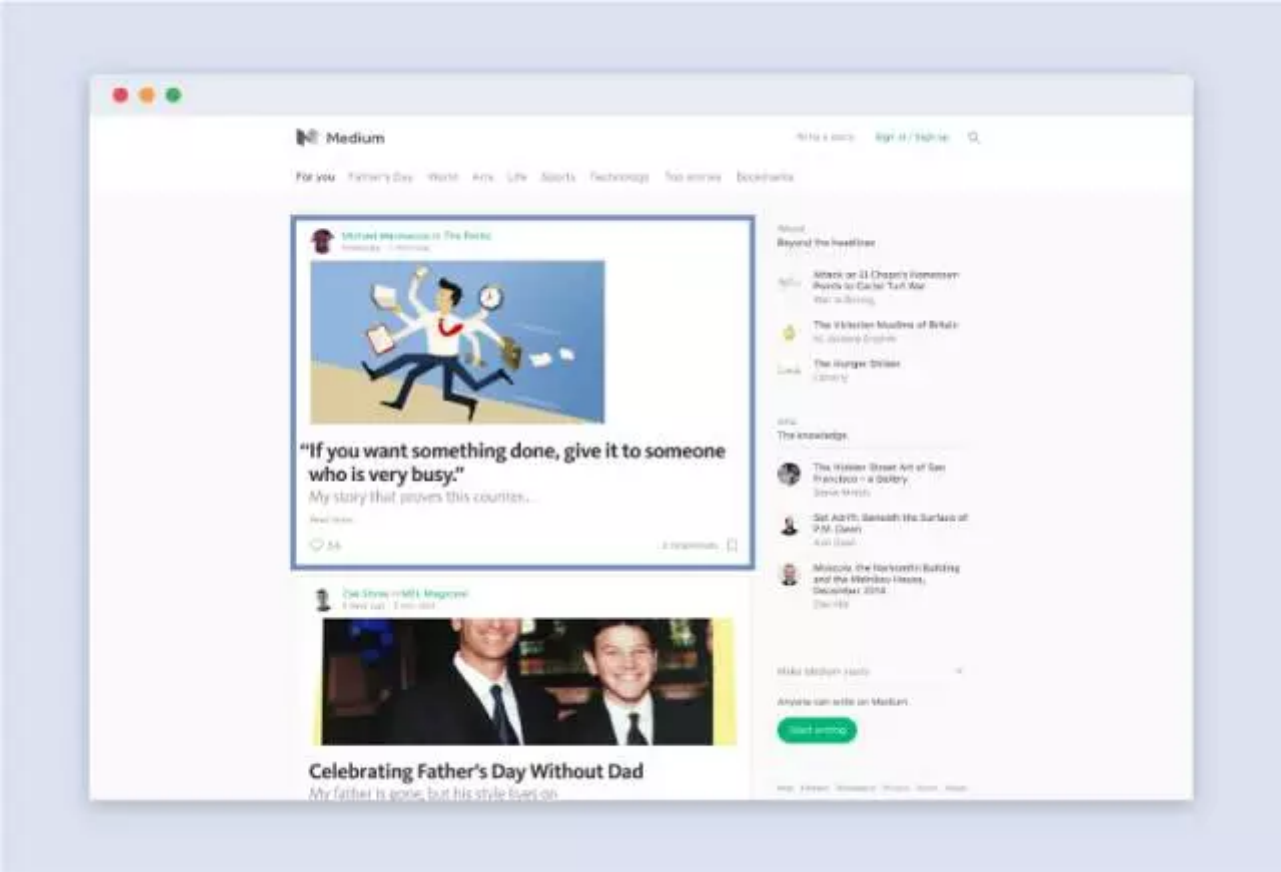
做好这四件事情，能让你在大规模使用 CSS 的时候保证代码的健壮性：使用适当的语义，模块化，采用统一的命名规范，遵循单一功能原则。

使用适当的语义

在 HTML 和 CSS 编程中有语义标注的概念。语义是指单词的含义和他们间的关系。在 HTML 编程中，意味着你需要使用一个合适的标签名字来标记。下面是一个经典的例子。

```
<!-- bad -->
<div class=" footer" ></div>
<!-- good -->
<footer></footer>
```

富有语义的 HTML 是非常简单明确的。另一方面，富有语义的 CSS 则是更抽象和主观的。编写富有语义的 CSS 意味着在选择类型的时候，类名要传达出结构和功能信息。类名要很容易被理解。确保它们不要太具体、太特殊。这样，你就可以复用它了。



为了阐述什么是一个良好的类名，请看这个简化了的 Medium 网站的 CSS 例子。

```
<div class="stream">
 <div class="streamItem">
  <article class="postArticle">
   <div class="postArticle-content">
    <!-- content -->
   </div>
  </article>
 </div>
```

```
</div>
```

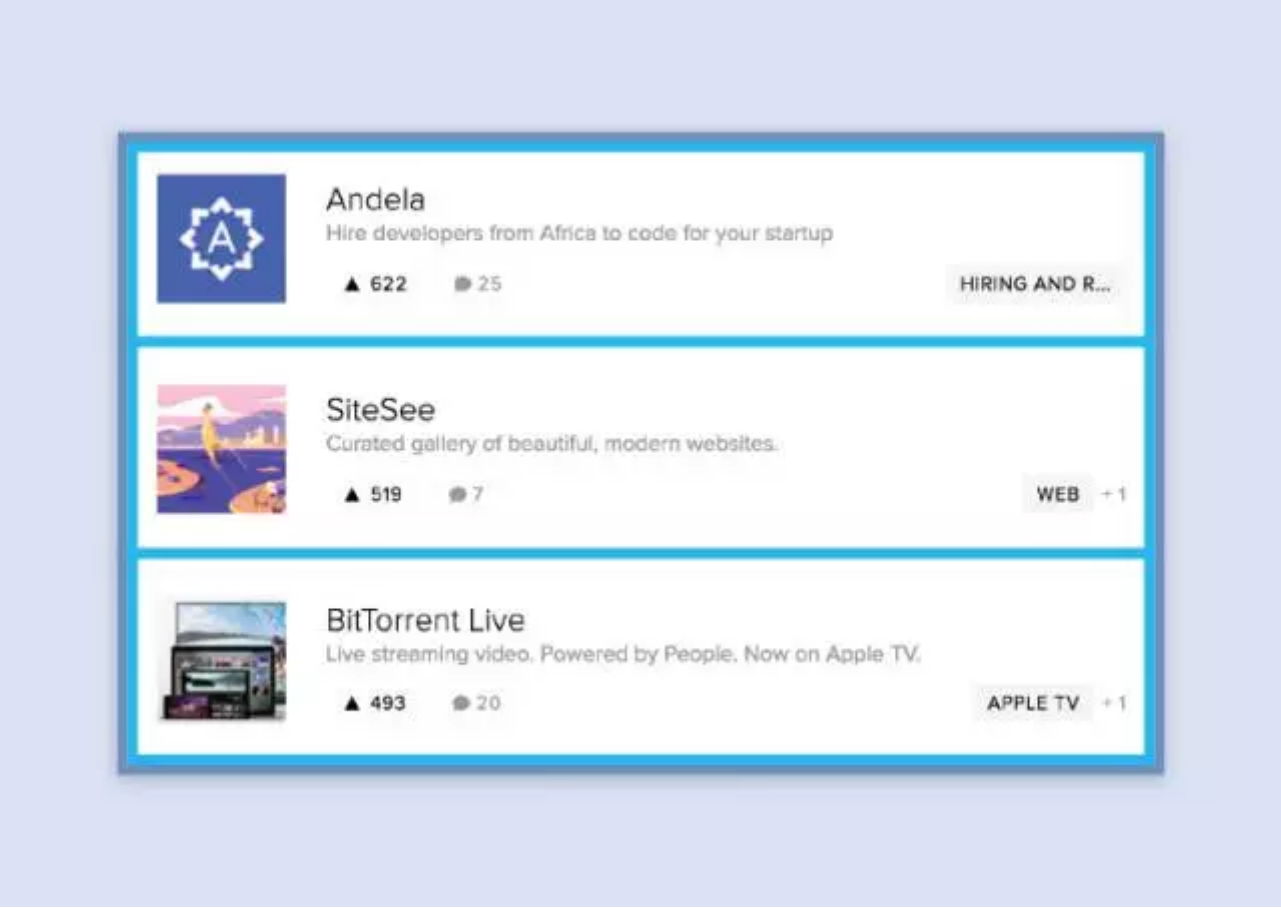
通过这些代码，你可以立即识别出它们的结构、功能和含义。父节点的类名是 stream ，内容是一个文章列表。它的子节点的类名是 streamItem ，内容是文章列表中的一篇具体的文章。这使我们很容易的了解到父节点和子节点之间的关系。并且，这些类可以在每一个有文章功能的页面中使用。

你可以像阅读一本书一样读 HTML 和 CSS。它会给你讲一个故事。通过故事你可以了解故事中的每一个角色和他们之间的关系。语义丰富的 CSS 代码容易理解，更便于维护。

若果你想进一步了解语义相关的内容，看看 《怎么富有语义的为类命名》、《CSS 命名不简单》和 《富有语义和容易识别（的代码命名）》，再看 《关于 HTML 命名和前端架构》。

模块化

在这个充满了组件库（以 React 为例）的时代，模块化就是王者。组件就是由已经解构了的接口创建的可组合的模块。下面是一个Product Hunt（一种发布好的创业项目的网站）前端页面。作为练习，让我们将这个页面分解成一系列的组件。



每种颜色框代表一个组件，stream 节点下分为好多个 stream item 子节点。

```
<div class="stream">
```

```
  <div class="streamItem">
    <!-- product info -->
  </div>
</div>
```

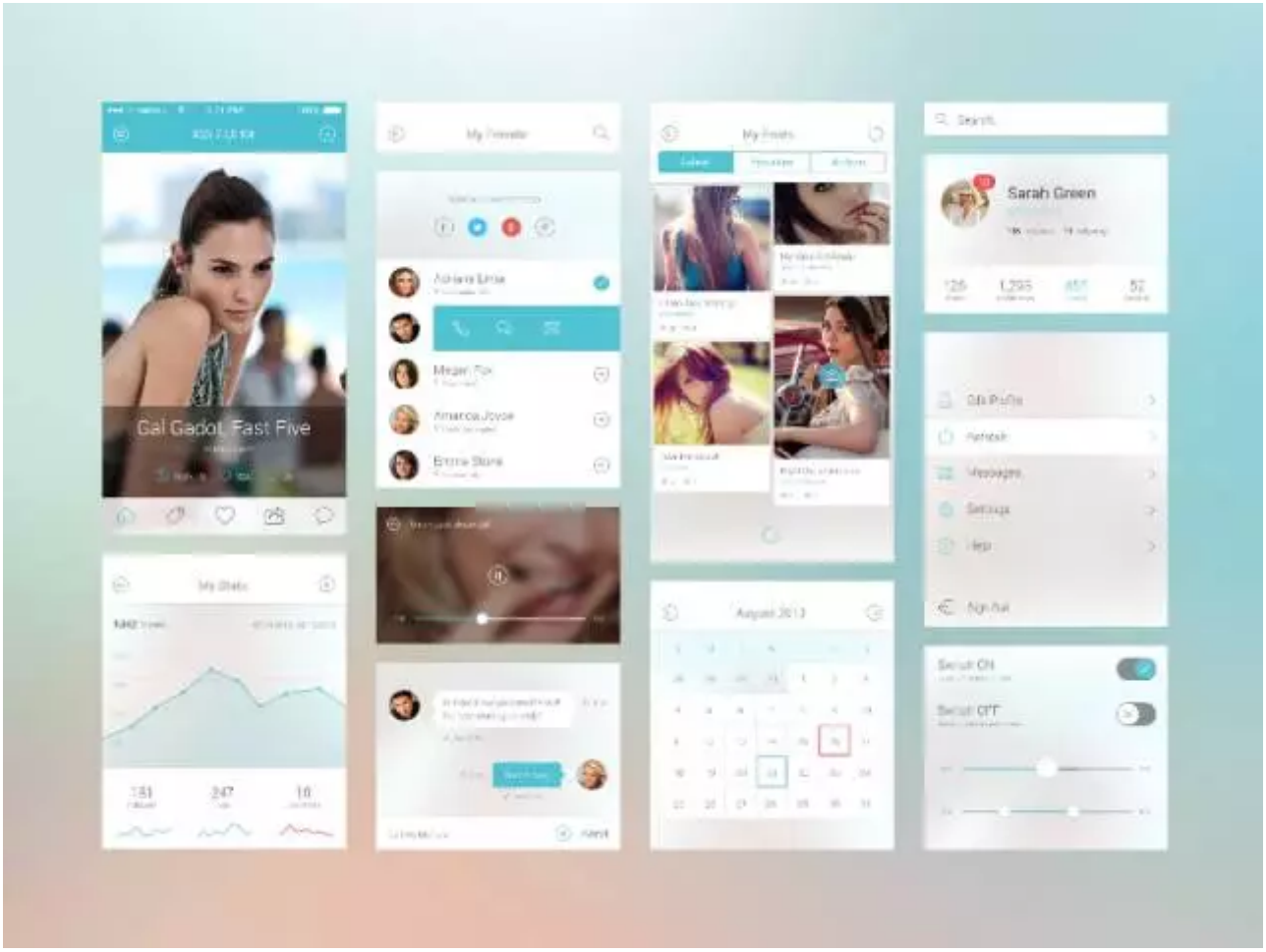
大多数组件都可以分解为更小的组件。

每一个 stream item 组件都有一个缩略图和一个特色的产品信息。

```
<!-- STREAM COMPONENT -->
<div class="stream">
  <div class="streamItem">
    <!-- POST COMPONENT -->
    <div class="post">
      <img src="thumbnail.png" class="postThumbnail"/>
      <div class="content">
        <!-- product info -->
      </div>
    </div>
  </div>
</div>
```

由于 stream 组件和它的子控件是完全独立的，你可以很容易的调整或者更换 post 组件，并且这不会对 stream 组件产生任何影响。

使用组件的思想将会使你的代码解耦。解耦的代码越多，你的类之间的依赖就越低。这会让你的代码更容易修改，并且使你的代码更长时间的工作下去而不用修改它。



组件驱动设计

模块化你的 CSS 时，首先将你的设计分解成多个组件。你可以使用纸和笔，也可以使用类似 Illustrator 或者 Sketch 这类的软件。确定你将要如何命名这些组件，同时理清各个组件之间的关系。

阅读更多关于 CSS 组件驱动的文章，详见《CSS 建构：可扩展和模块化处理》、《使用 Sass 编写模块化的 CSS》和《模块化你的前端代码——编写高可维护和条理清晰的代码》。

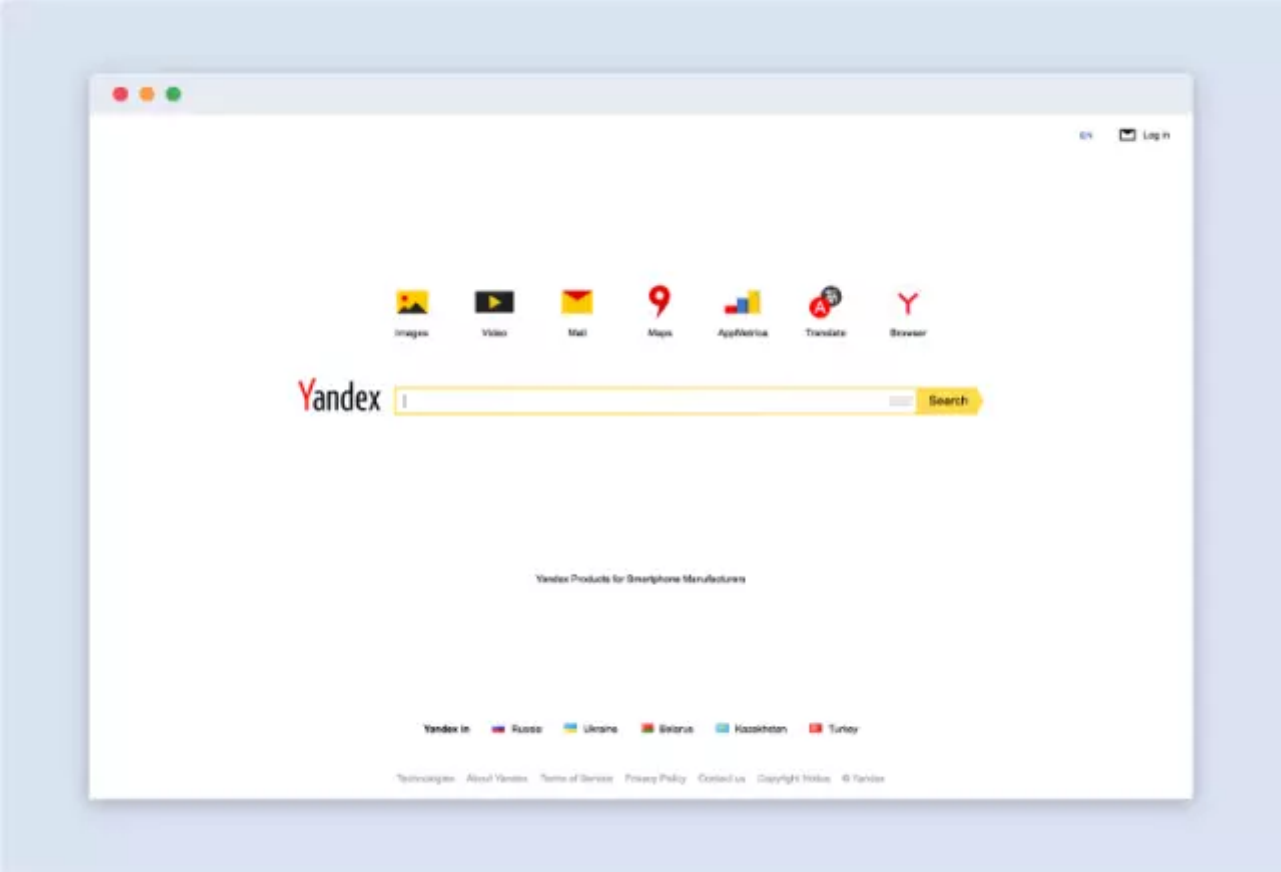
采用统一的命名规范

目前有几十个不同版本的 CSS 命名规范。有些人对他们选择的命名规范极其笃定，认为他们的比别人的更好。事实上，不同的人喜欢不同的命名规范。我听到的最好的建议是：选择你觉得最合适的命名规范。

下面简单列举一下常用的命名规范：

- Object oriented CSS OOCSS
- Block element modifier (BEM)
- Scalable and modular architecture for CSS (SMACSS)
- Atomic

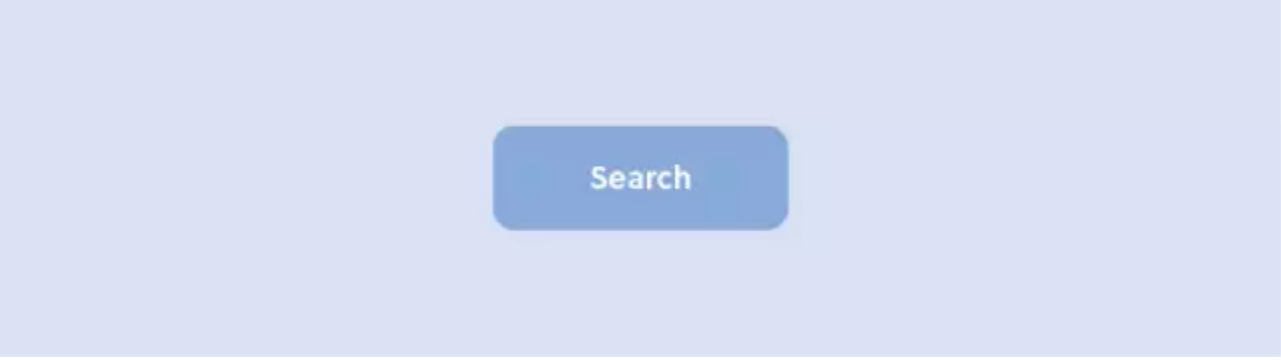
我最喜欢的命名规范是 BEM。BEM 代表块(block)、元素（element）和修饰符（modifier）。Yandex，在俄罗斯的相当于谷歌的搜索引擎，为了解决他们 CSS 代码库中的缩放问题而提出了它（它指BEM）。



BEM 是一个极其简单——又极其严格——的命名规范。

```
.block {}
.block__element {}
.block--modifier {}
```

块（Blocks）代表高级别的类。元素（Elements）是块的子模块。修饰符（modifiers）代表不同的状态。




```
<div class="search">
```

```
<input type="search__btn search__btn--active" />
</div>
```

在上面的示例中， search 是块（block），search button是它的元素（element）。如果你想要更改按钮的状态，我们可以为按钮增加一个修饰符，例如 active 。

关于命名规范要记住的一件事是，无论你喜欢哪种命名规范，你会经常继承或者工作在不同标准的代码库上。请敞开心扉去学习新的标准，用不同的思维去思考 CSS 。

你可以在《深入学习 BEM 语法》、《BEM 101》和《BEM 简介》上看到更多关于 BEM 的信息。想要了解不同的命名规范，参见《OOCSS、ACSS、BEM、SMACSS：这些是什么？我该用哪个？》。

遵循单一功能原则

单一功能原则规定每个模块和类都应该有一个单一的功能，并且该功能应该由这个类完全封装起来。

在 CSS 中，单一功能原则代表每一段代码、类和模块只做一件事。当我们提交 CSS 文件时，这意味着每个独立的组件（例如轮播效果和导航栏）都应该有自己的 CSS 文件。

```
/components
  |- carousel
  |- |- carousel.css
  |- |- carousel.partial.html
  |- |- carousel.js
  |- nav
  |- |- nav.css
  |- |- nav.partial.html
  |- |- nav.js
```

另外一个常见的组织文件的方式是按照功能将文件分组。举个栗子，如上面所示，所有和轮播效果组件有关的文件都被分类到了一起。采用这种方式可以让你更容易的找到相关文件。

除了对组件的样式进行分离之外，最好利用单一功能原则对全局样式也进行分离。

```
/base
  |- application.css
  |- typography.css
  |- colors.css
  |- grid.css
```

在这个例子中，每个相关的样式被分离到自己的样式文件中。这样，如果你想要修改样式中的颜色，那么你将会很容易的找到它。

无论你使用哪种方式组织文件结构，尽量在决定的时候参考单一功能原则。一旦有某个文件开始变的臃肿，那么考虑按照逻辑功能将它分成多个部分。

更多有关组织文件结构和 CSS 架构的文章，详见《Sass 审美 1：架构和组织样式文件》和《可扩展和可维护的 CSS 架构》。

当单一功能原则应用于你的每一个 CSS 类选择器中时，这意味着每一个类选择器都有着唯一的功能。换句话说，要根据不同关注点将样式分离到不同的类选择器中。下面是个经典的例子：

```
.splash {
  background: #f2f2f2;
  color: #fffff;
  margin: 20px;
  padding: 30px;
  border-radius: 4px;
  position: absolute;
  top: 0;
  right: 0;
  bottom: 0;
  left: 0;
}
```

在上面的例子中，我们将关注点耦合了。splash 这个类不但包含了本身的样式和逻辑，同时也包含了它的子节点的。为了解决这个问题，我们可以将这段代码分离为两个新的类。

```
.splash {
  position: absolute;
  top: 0;
  right: 0;
  bottom: 0;
  left: 0;
}
```

现在我们有 splash 和 splash content 两个类。我们可以将 splash 作为一个一般的全屏类，它可以拥有任何子节点。所有子节点关注的属性，都在 splash content 中，与父节点的属性是完全解耦的。

你可以通过阅读下列文章进一步了解单一功能原则在样式表和类中的应用。《单一功能原则在 CSS 中的应用》和《单一功能原则》。

简单胜过复杂

如果你问任何一个成功的前端开发工程师或者 CSS 架构师，他们会告诉你，他们从来没有对自己的代码完全满意。写好 CSS 是一个不断迭代的过程。从简单开始，遵循基本的 CSS 规则和样式指南，然后不断迭代。

我很想知道你的 CSS 学习之路。你喜欢的命名规范是什么？你是怎样组织你的代码文件的？你可以随时通过留言或者在 Tweet 上告诉我。

阅读原文