编写现代 CSS 代码的 20 个建议

2017-08-27 前端大全

（点击上方公众号，可快速关注）

> 英文：Danny Markov 译文：王下邀月熊
> https://segmentfault.com/a/1190000006834519
> 如有好文章投稿，请点击 → 这里了解详情

> 本文翻译自Danny Markov 的20-Tips-For-Writing-Modern-CSS一文。
> 本文归纳于笔者的Web Frontend Introduction And Best Practices:前端入门与最佳实践中CSS入门与最佳实践系列，其他的关于CSS样式指南的还有提升你的CSS姿势、Facebook里是怎样提升CSS代码质量的。本文更偏向于样式使用技巧，前两篇偏向于代码风格与规范

明白何谓Margin Collapse

不同于其他很多属性，盒模型中垂直方向上的Margin会在相遇时发生崩塌，也就是说当某个元素的底部Margin与另一个元素的顶部Margin相邻时，只有二者中的较大值会被保留下来，可以从下面这个简单的例子来学习:

```
.square {
    width: 80px;
    height: 80px;
}

.red {
    background-color: #F44336;
    margin-bottom: 40px;
}

.blue {
    background-color: #2196F3;
    margin-top: 30px;
}
```

在上述例子中我们会发现，红色和蓝色方块的外边距并没有相加得到70px，而是只有红色的下外边距保留了下来。我们可以使用一些方法来避免这种行为，不过建议来说还是尽量统一使用margin-bottom属性，这样就显得和谐多了。

使用Flexbox进行布局

> CSS实战之Flex详解以及其在微信中的兼容实现

在传统的布局中我们习惯使用Floats或者inline-blocks，不过它们更适合于格式化文档，而不是整个网站。而Flexbox则是专门的用于进行布局的工具。Flexbox模型允许开发者使用很多便捷可扩展的属性来进行布局，估计你一旦用上就舍不得了:

```
.container {
    display: flex;
    /* Don't forget to add prefixes for Safari */display: -webkit-flex;
}
```

我们已经在Tutorialzine上提供了很多的关于Flexbox的介绍与小技巧，譬如5 Flexbox Techniques You Need to Know About。

使用CSS Reset

虽然这些年来随着浏览器的迅速发展与规范的统一，浏览器特性碎片化的情况有所改善，但是在不同的浏览器之间仍然存在着很多的行为差异。而解决这种问题的最好的办法就是使用某个CSS Reset来为所有的元素设置统一的样式，保证你能在相对统一干净的样式表的基础上开始工作。目前流行的Reset库有 normalize.css, minireset以及 ress ，它们都可以修正很多已知的浏览器之间的差异性。而如果你不打算用某个外在的库，那么建议可以使用如下的基本规则:

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
```

上面的规则看起来没啥用，不过如果不同的浏览器在默认情况下为你设置了不同的外边距/内边距的默认值，还是会挺麻烦的。

一切应为Border-box

虽然很多初学者并不了解box-sizing这个属性，但是它确实相当的重要。而最好的理解它的方式就是看看它的两种取值:

- 默认值为content-box，即当我们设置某个元素的heght/width属性时，仅仅会作用于其内容尺寸。而所有的内边距与边都是在其之上的累加，譬如某个 标签设置为宽100，内边距为10，那么最终元素会占用120(100 + 2*10)的像素。
- border-box:内边距与边是包含在了width/height之内，譬如设置了width:100px的 无论其内边距或者边长设置为多少，其占有的大小都是100px。

将元素设置为border-box会很方便你进行样式布局，这样的话你就可以在父元素设置高宽限制而不担心子元素的内边距或者边打破了这种限制。

以背景图方式使用Images

如果需要在响应式的环境下展示图片，有个简单的小技巧就是使用该图片作为某个 的背景图而不是直接使用img标签。基于这种方式配合上background-size与background-position这两个属性，可以很方便地按比例缩放:

```
img {
    width: 300px;
    height: 200px;
}

div {
    width: 300px;
    height: 200px;
    background: url('http://cdn.tutorialzine.com/wp-content/uploads/2016/08/bicycle.jpg');
    background-position: center center;
    background-size: cover;
}

section{
    float: left;
    margin: 15px;
}
```

不过这种方式也是存在缺陷的，譬如你无法设置图片的懒加载、图片无法被搜索引擎或者其他类似的工具抓取到，有个不错的属性叫object-fit可以解决这个问题，不过该属性目前的浏览器支持并不是很完善。

Better Table Borders

HTML中使用Tables进行布局一直是个很头疼的问题，它们使用起来很简单，但是无法进行响应式操作，并且也不方便进行全局样式设置。譬如，如果你打算为Table的边与单元的边添加样式，可能得到的结果如下:

```
table {
    width: 600px;
    border: 1px solid #505050;
    margin-bottom: 15px;
    color:#505050;
}

td{
    border: 1px solid #505050;
    padding: 10px;
}
```

Title	Description
Take a Selfie With JavaScript	A quick tutorial that covers a pure-JavaScript way to take photos directly in the browser using various native APIs and interesting JS techniques.
20 Awesome PHP Libraries For Summer 2016	A handcrafted collection of 20 open-source libraries containing some of the most useful and top-quality PHP resources for 2016.

这里存在的问题是出现了很多的重复的边，会导致视觉上不协调的情况，那么我们可以通过设置border-collapse:collapse来进行处理:

Title	Description
Take a Selfie With JavaScript	A quick tutorial that covers a pure-JavaScript way to take photos directly in the browser using various native APIs and interesting JS techniques.
20 Awesome PHP Libraries For Summer 2016	A handcrafted collection of 20 open-source libraries containing some of the most useful and top-quality PHP resources for 2016.

注释格式优化

CSS虽然谈不上一门编程语言但是其仍然需要添加注释以保障整体代码的可读性，只要添加些简单的注释不仅可以方便你更好地组织整个样式表还能够让你的同事或者未来的自己更好地理解。对于CSS中整块的注释或者使用在Media-Query中的注释，建议是使用如下形式：

```
/*---------------
    #Header
---------------*/header { }header nav { }/*---------------
    #Slideshow
---------------*/.slideshow { }
```

而设计的细节说明或者一些不重要的组件可以用如下单行注释的方式:

```
/*   Footer Buttons   */   .footer button { }

.footer button:hover { }
```

同时，不要忘了CSS中是没有//这种注释方式的:

```
/*  Do  */p {
    padding: 15px;
    /*border: 1px solid #222;*/   }/*  Don't  */p {
```

```
    padding: 15px;
    // border: 1px solid #222;     }
```

使用Kebab-case命名变量

对于样式类名或者ID名的命名都需要在多个单词之间添加-符号，CSS本身是大小写不敏感的因此你是用不了camelCase的，另一方面，很久之前也不支持下划线，所以现在的默认的命名方式就是使用-:

```
/*  Do     */   .footer-column-left { }

/*  Don't  */   .footerColumnLeft { }

.footer_column_left { }
```

而涉及到具体的变量命名规范时，建议是使用BEM规范，只要遵循一些简单的原则即可以保证基于组件风格的命名一致性。你也可以参考CSS Tricks来获得更多的细节描述。

避免重复代码

大部分元素的CSS属性都是从DOM树根部继承而来，这也是其命名为级联样式表的由来。我们以font属性为例，该属性往往是继承自父属性，因此我们并不需要再单独地为元素设置该属性。我们只需要在html或者body中添加该属性然后使其层次传递下去即可:

```
html {
    font: normal 16px/1.4 sans-serif;
}
```

使用transform添加CSS Animations

不建议直接改变元素的width与height属性或者left/top/bottom/right这些属性来达到动画效果，而应该优先使用transform()属性来提供更平滑的变换效果，并且能使得代码的可读性会更好:

```
.ball {
    left: 50px;
    transition: 0.4s ease-out;
}/* Not Cool*/.ball.slide-out {
    left: 500px;
}/* Cool*/.ball.slide-out {
    transform: translateX(450px);
}
```

Transform的几个属性translate、rotate、scale都具有比较好的浏览器兼容性可以放心使用。

不要重复造轮子

现在CSS社区已经非常庞大，并且不断地有新的各式各样的库开源出来。这些库可以帮助我们解决从小的代码片到用于构建完整的响应式应用的全框架。所以如果下次你再碰到什么CSS问题的时候，在打算撸起袖子自己上之前可以尝试在GitHUB或者CodePen上搜索可行方案。

尽可能使用低优先级的选择器

并不是所有的CSS选择器的优先级都一样，很多初学者在使用CSS选择器的时候都是考虑以新的特性去复写全部的继承特性，不过这一点在某个元素多状态时就麻烦了，譬如下面这个例子:

```
a{
    color: #fff;
    padding: 15px;
}

a#blue-btn {
    background-color: blue;
}

a.active {
    background-color: red;
}
```

我们本来希望将.active类添加到按钮上然后使其显示为红色，不过在上面这个例子中很明显起不了作用，因为button已经以ID选择器设置过了背景色，也就是所谓的Higher Selector Specificity。一般来说，选择器的优先级顺序为：ID(#id) > Class(.class) > Type(header)

避免使用!important

认真的说，千万要避免使用!important，这可能会导致你在未来的开发中无尽的属性重写，你应该选择更合适的CSS选择器。而唯一的可以使用!important属性的场景就是当你想去复写某些行内样式的时候，不过行内样式本身也是需要避免的。

使用text-transform属性设置文本大写

```
<div class="movie-poster">Star Wars: The Force Awakens</div>

.movie-poster {
    text-transform: uppercase;
}
```

Em, Rem, 以及 Pixel

已经有很多关于人们应该如何使用em，rem，以及px作为元素尺寸与文本尺寸的讨论，而笔者认为，这三个尺寸单位都有其适用与不适用的地方。不同的开发与项目都有其特定的设置，因此并

没有通用的规则来决定应该使用哪个单位，这里是我总结的几个考虑:

- em – 其基本单位即为当前元素的font-size值，经常适用于media-queries中，em是特别适用于响应式开发中。
- rem – 其是相对于html属性的单位，可以保证文本段落真正的响应式尺寸特性。
- px – Pixels 并没有任何的动态扩展性，它们往往用于描述绝对单位，并且可以在设置值与最终的显示效果之间保留一定的一致性。

在大型项目中使用预处理器

估计你肯定听说过 Sass, Less, PostCSS, Stylus这些预处理器与对应的语法。Preprocessors可以允许我们将未来的CSS特性应用在当前的代码开发中，譬如变量支持、函数、嵌套式的选择器以及很多其他的特性，这里我们以Sass为例:

```
$accent-color: #2196F3;

a {
    padding: 10px 15px;
    background-color: $accent-color;
}

a:hover {
    background-color: darken($accent-color,10%);
}
```

使用Autoprefixers来提升浏览器兼容性

使用特定的浏览器前缀是CSS开发中常见的工作之一，不同的浏览器、不同的属性对于前缀的要求也不一样，这就使得我们无法在编码过程中记住所有的前缀规则。并且在写样式代码的时候还需要加上特定的浏览器前缀支持也是个麻烦活，幸亏现在也是有很多工具可以辅助我们进行这样的开发:

- Online tools: Autoprefixer
- Text editor plugins: Sublime Text, Atom
- Libraries: Autoprefixer (PostCSS)

在生产环境下使用Minified代码

为了提升页面的加载速度，在生产环境下我们应该默认使用压缩之后的资源代码。在压缩的过程中，会将所有的空白与重复剔除掉从而减少整个文件的体积大小。当然，经过压缩之后的代码毫无可读性，因此在开发阶段我们还是应该使用普通的版本。对于CSS的压缩有很多的现行工具:

- Online tools – CSS Minifier (API included), CSS Compressor
- Text editor plugins: Sublime Text, Atom

- Libraries: Minfiy (PHP), CSSO and CSSNano (PostCSS, Grunt, Gulp)

选择哪个工具肯定是依赖于你自己的工作流啦~

多参阅Caniuse

不同的浏览器在兼容性上差异很大，因此如果我们可以针对我们所需要适配的浏览器，在caniuse上我们可以查询某个特性的浏览器版本适配性，是否需要添加特定的前缀或者在某个平台上是否存在Bug等等。不过光光使用caniuse肯定是不够的，我们还需要使用些额外的服务来进行检测。

Validate:校验

对于CSS的校验可能不如HTML校验或者JavaScript校验那么重要，不过在正式发布之前用Lint工具校验一波你的CSS代码还是很有意义的。它会告诉你代码中潜在的错误，提示你一些不符合最佳实践的代码以及给你一些提升代码性能的建议。就像Minifers与Autoprefixers，也有很多可用的工具:

- Online tools: W3 Validator, CSS Lint
- Text editor plugins: Sublime Text, Atom
- Libraries: stylelint (Node.js, PostCSS), css-validator (Node.js)