**20 个让你效率更高的 CSS 代码技巧**

**1.注意外边距折叠**

与其他大多数属性不同，上下的垂直外边距margin在同时存在时会发生外边距折叠。

这意味着当一个元素的下边缘接触到另一个元素的上边缘时，只会保留两个margin值中较大的那个。

例如：

HTML<div class="square red"></div>

<div class="square blue"></div>

CSS.square {

width: 80px;

height: 80px;

}

.red {

background-color: #F44336;

margin-bottom: 40px;

}

.blue {

background-color: #2196F3;

margin-top: 30px;

}

clipboard.png

红色方块与蓝色方块的上下间距是40px，而不是70px。解决外边距折叠的方法有很多种，

对于初学者来说最简单的就是所有元素只使用一个方向上的margin，

比如上下的外边距我们统统使用margin-bottom。

===================================================

**2.使用flex进行布局**

flex弹性布局的出现是有原因的。浮动和inline-block虽然也能实现很多的布局效果，但它们本质上是文本和块元素布局的工具，而不是面向整个网页的。flex可以很容易的按照我们预期的方式创建布局。

flex拥有一组面向“弹性容器”的属性和一组面向“弹性项目”的属性，一旦你学会了它们，做任何响应式布局都是小菜一碟。目前各类浏览器的最新版本对flex的支持性也是没有任何问题的，所以你应该多多使用flex布局。

.container {

display: flex;

}

==================

**3.重置元素的CSS样式**

尽管这些年来有了很大的改善，但是不同浏览器对于各种元素的默认样式仍然存在很大的差异。解决这个问题的最佳办法是在CSS开头为所有的元素设置通用的CSS Reset重置代码，这样你是在没有任何默认内外边距的基础上进行布局，于是所产生的效果也就是统一的。

网络上已经有成熟的CSS代码库为我们解决浏览器不一致问题，例如normalize.css、minireset和ress，你可以在你的项目中引用它们。如果你不想使用第三方代码库，你可以使用下面的样式来进行一个非常基本的CSS reset：

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

上面的代码看起来有些霸道，将所有元素的内外边距都设置为0了，而正是没有了这些默认内外边距的影响，使得我们后面的CSS设置会更加的容易。同时box-sizing: border-box也是一个很棒的设置，我们紧接着就会介绍它。

==========================

**4.所有元素设置为Border-box**

大多数初学者都不知道box-sizing这个属性，但实际上它非常重要。box-sizing属性有两个值：

content-box（默认） - 当我们设置一个元素的宽度或高度时，就是设置它的内容的大小。所有的padding和边框值都不包含。例如，一个div的宽度设置为100，padding为10，于是这个元素将占用120像素（100+2\*10）。

border-box - padding与边框包含在元素的宽度或高度中，一个设置为width: 100px和box-sizing:

border-box的div元素，他的总宽度就是100px，无论它的内边距和边框有多少。

将所有元素都设置为border-box，可以更轻松的改变元素的大小，而不必担心padding或者border值会将元素撑开变形或者换行显示。

==================================

**5.将图片作为背景**

当给页面添加图片时，尤其需要图片是响应式的时候，最好使用background属性来引入图片，而不是<img>标签。

这看起来使用图片会更复杂，但实际上它会使设置图片的样式变得更加容易。有了background-size, background-position和其它的属性，保持或改变图片原始尺寸和宽高比会更方便。

举个例子：

HTML

<section>

<p>Img element</p>

<img src="https://tutorialzine.com/media/2016/08/bicycle.jpg" alt="bicycle">

</section>

<section>

<p>Div with background image</p>

<div></div>

</section>

CSS

img {

width: 300px;

height: 200px;

}

div {

width: 300px;

height: 200px;

background: url('https://tutorialzine.com/media/2016/08/bicycle.jpg');

background-position: center center;

background-size: cover;

}

section{

float: left;

margin: 15px;

}

clipboard.png

background引入图片的一个缺点是页面的Web可访问性会受到轻微的影响，因为屏幕阅读器和搜索引擎无法正确地获取到图像。这个问题可以通过CSS object-fit属性解决，到目前为止除了IE浏览器其他的浏览器都可以使用object-fit。

==========================

**6.更好的表格边框**

HTML中的表格总是很难看的。它们很难做成响应式的，而且总体上很难改变样式。例如，如果要向表格及其单元格添加简单的边框，则最可能的结果是：

clipboard.png

如你所见，有很多重复的边框，看起来很不好看。这里有一个快速的方法来删除所有的双倍边框：border-collapse: collapse，只需设置这个属性后，表格的边框看起来就顺眼多了：

clipboard.png

=======================

**7.更友好的注释**

CSS也许不是一种编程语言，但其代码仍然需要文档化。添加一些简单的注释可以将代码分类区分，方便自己和同事后期维护。

对于大的区域划分或者重要的组件可以使用下面的注释样式：

/\*---------------

#Header

---------------\*/

header { }

header nav { }

/\*---------------

#Slideshow

---------------\*/

.slideshow { }

对于细节和不太重要的样式可以使用单行的注释方式：

/\* Footer Buttons \*/

.footer button { }

.footer button:hover { }

另外，请记住，CSS中没有//注释，只有/\*\*/注释：

/\* 正确 \*/

p {

padding: 15px;

/\*border: 1px solid #222;\*/

}

/\* 错误 \*/

p {

padding: 15px;

// border: 1px solid #222;

}

============================================

**8.短横线命名**

当class或者ID包含多个单词时，应使用连字符（-），CSS不区分大小写，因此不能使用驼峰式命名。同样，CSS中也不建议使用下划线连接的命名方式。

/\* 正确 \*/

.footer-column-left { }

/\* 错误 \*/

.footerColumnLeft { }

.footer\_column\_left { }

当涉及到命名时，您还可以考虑BEM，它遵循一组原则，提供基于组件并增加一致性的开发方法。

=========================

**9.不要重复设置**

大多数CSS属性的值都是从DOM树中向上一级的元素继承的，因此才被命名为级联样式表。以font属性为例-它总是从父级继承的，您不必为页面上的每个元素都单独设置。

只需将要设置的字体样式添加到<html>或<body>元素中，然后让它们自动向下继承。

html {

font: normal 16px/1.4 sans-serif;

}

然后我们就可以统一的一次改变页面上所有的文字样式了。当然，CSS中并不是所有的属性都是可继承的，对于这些属性我们仍然需要在每个元素上单独设置。

========================================

**10.使用transform属性来创建动画**

最好使用transform()函数来创建元素的位移或大小动画，尽量不要直接改变元素的width，height以及left/top/bottom/right属性值。

下面的例子中，我们给.ball元素添加了一个从左向右的移动动画。推荐使用transform: translateX()函数来代替left属性。

.ball {

left: 50px;

transition: 0.4s ease-out;

}

/\* 不建议 \*/

.ball.slide-out {

left: 500px;

}

/\* 建议 \*/

.ball.slide-out {

transform: translateX(450px);

}

transform以及它的所有函数（translate, rotate, scale等）几乎没有浏览器兼容性问题，可以随意使用。

==========================

**11.不要DIY，多使用代码库**

CSS社区非常庞大，不断有新的代码库出现。它们有各种用途，从微小的片段到构建响应式应用程序的整体框架。其中大多数也是开源的。

下一次当你面对一个CSS问题时，在你试图费尽全力解决它之前，检查一下Github或Codepen上是否已经有了一个可用的解决方案。

=======================================

**12.保持选择器的低权重**

css的选择器并不都是平等的。当初学习CSS时，我总是认为选择器会覆盖它上面的所有内容。然而，情况并非如此，就如我们在下面的示例中所说明的：

HTML

<a href='#' id='blue-btn' class="active">按钮</a>

CSS

a{

color: #fff;

padding: 15px;

}

a#blue-btn {

background-color: blue;

}

a.active {

background-color: red;

}

clipboard.png

我们希望.active类中设置的样式会生效使按钮变为红色。但是它并不会起作用，因为按钮在上面有一个ID选择器，它同样设置了background-color，ID选择器具有更高的权重，所以按钮的颜色是蓝色的。选择器的权重大小规格如下：

ID (#id) > Class (.class) > Type (例如 header)

权重也会叠加，于是a#button.active的权重要比a#button的高。一开始就使用高权重的选择器会导致你在后面的维护中不断的使用更高权重的选择器，最终选择使用!important，这是非常不推荐的，具体的原因紧接着就会讲到。

==========================

**13.不要使用!important**

说真的，不要使用!important。现在看起来可以快速的解决问题，但最终可能会导致大量的重写。相反，我们应该花点时间找到CSS选择器不工作的原因并更改它。

唯一可以使用的!important的地方是当您想要覆盖HTML中的内联样式时，但是内联样式同样也是一个坏的习惯，应该尽量的避免。

============================

**14.使用text-transform转换字母为大写**

本条适用于英文环境，不适合中文

在HTML中，可以将某个单词全部写为大写字母来表达强调的含义。比如：

<h3>Employees MUST wear a helmet!</h3>

如果你需要将某段文字全部转化为大写，我们可以在HTML中正常书写，然后通过CSS来转化。这样可以保持上下文内容的一致性。

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

text-transform: uppercase;

}

==================================

**15.Em, Rem与px**

设置元素与文本的大小应该用哪种单位，em，rem，还是px？一直以来都有很多的争论。事实是，这三种选择都是可行的，都有其利弊。

在什么时候在什么项目使用哪种单位是没有一个定论的，开发人员的习惯不同，项目的要求不同，都可能会使用不同的单位。然而，虽然没有固定的规则，但是每种单位还是有一些要注意的地方的：

em - 设置元素为1em，其大小与父元素的font-size属性有关。这个单位用于媒体查询中，特别适用于响应式开发，但是由于em单位在每一级中都是相对于父元素进行计算的，所以要得出某个子元素em单位对应的px值，有时候是很麻烦的。

rem - 相对于<html>元素的font-size大小计算，rem使得统一改变页面上的所有标题和段落文本大小变得非常容易。

px - 像素单位是最精确的，但是不适用于自适应的设计。px单位是可靠的，并且易于理解，我们可以精细的控制元素的大小和移动到1px。

最重要的是，不要害怕尝试，尝试所有方法，看看最适合什么。有时候，em和rem可以节省很多工作，尤其是在构建响应式页面时。

=============================

**16.对于大型项目使用预处理器**

你一定听说过它们 - Sass, Less, PostCSS, Stylus。预处理器是CSS的未来。它们提供诸如变量、CSS函数、选择器嵌套和许多其他很酷的功能，使CSS代码更易于管理，特别是在大型项目中。

举个简单的例子，下面是一个SASS代码的片段，它使用到了一些CSS变量和函数：

$accent-color: #2196F3;

a {

padding: 10px 15px;

background-color: $accent-color;

}

a:hover {

background-color: darken($accent-color,10%);

}

预处理器的唯一的不足之处是它们任然需要编译成普通的CSS。而CSS推出的自定义属性则是真正意义上的预处理。

==========================

17.使用AutoPrefixer达到更好的兼容性

浏览器前缀是CSS中最烦人的事情之一，每个属性需要的前缀是不一致的，你永远不知道到底需要哪一个，如果真的要把它一个一个手动添加到样式表中，那无疑是一个无聊的噩梦。

值得庆幸的是，有工具可以自动为我们提供添加浏览器前缀的功能，甚至可以决定需要支持哪些浏览器：

在线工具：Autoprefixer

文本编辑器插件：Sublime Text, Atom

代码库：Autoprefixer (PostCSS)

=============================

18.压缩CSS文件

为了提高网站和应用程序的加载速度和页面负载，您应该使用压缩后的资源。压缩版本的文件将删除所有空白和重复，从而减少总文件的体积。当然，这个过程也会使样式表完全不可读，所以要在生产环境中使用.min版本，同时为开发保留常规版本。

有许多不同的方法来压缩CSS代码：

在线工具：CSS Minifier, CSS Compressor

文本编辑器插件： Sublime Text, Atom

代码库： Minfiy (PHP), CSSO, CSSNano (PostCSS, Grunt, Gulp)

根据您的工作流程，可以使用上述任何一种方式。

================================

**19.Caniuse**

对于CSS的属性Web浏览器仍然存在许多兼容性不一致的地方。使用caniuse来检查您使用的属性是否得到了广泛的支持？是否需要前缀？或者是否在某个浏览器中使用有要注意的地方？有了caniuse你在写CSS时就会更得心应手了。

==================================

**20.验证**

验证CSS可能不像验证HTML或JavaScript代码那么重要，但是通过工具运行一下你的代码仍然非常有用。它会告诉你是否犯了任何错误，警告错误的用法，并为您提供改进代码的提示。

就像压缩和Autoprefixer一样，有免费的工具可以利用：

在线工具：W3 Validator, CSS Lint

文本编辑器插件：Sublime Text, Atom

代码库：stylelint (Node.js, PostCSS), css-validator (Node.js)