PostCSS深入学习: 简写和速写

作者：大漠 日期：2015-12-23 点击：4971

PostCSS
PostCSS
深入学习
CSS处理器

> 小编推荐：掘金是一个高质量的技术社区，从 ECMAScript 6 到 Vue.js，性能优化到开源类库，让你不错过前端开发的每一个技术干货。各大应用市场搜索「掘金」即可下载APP，技术干货尽在掌握中。

到目前为止，我们已经使用 PostCSS 做了很多事情，比如，优化样式，给预处理器增加功能，生成某些符合规范的样式，但是它只有帮助写原始 CSS 的人吗？

每个 web 设计师在他们的职业生涯中都会写出有很多可复用的代码，如果你能用一

点时间把每一个都收集起来到最后那将会变得很多。

在本教程中，我们要使用 PostCSS 的一系列的简写和速写来减少每天的编代码的时间。 让我们开始吧！

项目配置

到现在为止，你将会熟悉这个过程，但你需要做的第一件事是使用 Gulp 或 Grunt 设置您的项目。 如果你不能确定在项目中使用Gulp还是Grunt。那么本教程将会告诉你，Gulp是一个很好的选择，因为其简单。

无论你使用什么样的项目配置指南，你都需要一个空的Gulp或Grunt的项目。你可以阅读本系列教程中有关于Gulp或Grunt配置项目的相关教程：

- PostCSS深入学习：Gulp设置
- PostCSS深入学习：Grunt配置

如果你不想从头开始手动设置您的项目，你可以下载本教程中提供的源码附件，提取Gulp或Grunt项目到一个空的文件夹中。

然后在命令终端运行：`npm install`。

安装插件

在本教程中我们将安装几个插件，每个处理不同类型的简写或速写。

不论你用 Gulp 或 Grunt，在你的项目目录下运行以下命令来安装我们将用到的插件：

```
npm install postcss-alias postcss-crip pc
```

现在，我们已在项目中准备好要加载的插件了。

通过Gulp加载插件

如果您使用 Gulp，在 `gulpfile.js` 文件中添加下面的这些这些变量:

```
var alias = require('postcss-alias');
var crip = require('postcss-crip');
var magician = require('postcss-font-magi
var triangle = require('postcss-triangle'
var circle = require('postcss-circle');
var linkColors = require('postcss-all-lir
var center = require('postcss-center');
var clearfix = require('postcss-clearfix'
var position = require('postcss-position'
var size = require('postcss-size');
var verthorz = require('postcss-verthorz'
var colorShort = require('postcss-color-s
```

现在将每个变量名添加到 `processors` 数组中：

```
var processors = [
    alias,
    crip,
    magician,
    triangle,
    circle,
    linkColors,
    center,
    clearfix,
    position,
    size,
    verthorz,
```

```
    colorShort
];
```

在命令行中运行 `gulp css` ，做一个快速测试。然后检查新的 `style.css` 文件是否出现在你项目的 `dest/` 文件夹下。

通过Grunt加载插件

如果您使用 Grunt ，更新被嵌套在 `options` 对象下 `processors` 对象，如下所示:

```
processors: [
    require('postcss-alias')(),
    require('postcss-crip')(),
    require('postcss-font-magician')(),
    require('postcss-triangle')(),
    require('postcss-circle')(),
    require('postcss-all-link-colors')(),
    require('postcss-center')(),
    require('postcss-clearfix')(),
    require('postcss-position')(),
    require('postcss-size')(),
    require('postcss-verthorz')(),
    require('postcss-color-short')()
]
```

通过运行 `grunt postcss` 命令做一个快速测试，然后检查新的 `style.css` 文件是否出现在您的项目 `dest/` 文件夹。

好吧，现在你的插件都安装好了，接下来让我们了解如何使用它们来 实现CSS简写和速写。

属性速写

有很多的属性，我们要在 CSS 中一遍又一遍输入。 当然，每次输入字符的时间是很

小的，但是经过几年的开发加起来也会很多。 我们要看这里的两个插件，它们可以让你整理一些属性成为速写版本，为了你能够得到一个快速并且流畅的体验来编写你的 CSS。

定义你自己的速写

由 @Sean King 编写的 `postcss-alias` 插件允许你定义自己的速写或"别名"。这可以确保你使用的速写标记符合你的思想方式并且对于你的记忆也很容易。

若要定义一些别名需要在样式的顶部加上一个语法是 `@alias {...}` 。 然后在 `@` 规则内通过添加 `alias-name: property-name;` 设置你的别名

向你的 `src/style.css` 里添加下面的示例代码，为 `border-size` ， `border-style` ， `border-color` 三个属性设置别名:

```
@alias {
    bsz: border-size;
    bst: border-style;
    bcl: border-color;
}
```

然后添加以下代码来测试使用的新别名：

```
.set_border {
    bsz: 1px;
    bst: solid;
    bcl: #ccc;
}
```

编译您的文件，并在您的 `dest/style.css` 文件中，您现在可以看到下面这样的代码:

```
.set_border {
    border-size: 1px;
    border-style: solid;
    border-color: #ccc;
}
```

有关于 `postcss-alias` 更多信息，可以点击这里查阅。

定义速记方式

如果您想要使用大量的属性速记，但你不想通过自己去定义每一步骤，你可以查阅由 @Johnie Hjelm 编写的有数百个属性缩写的插件 `postcss-crip` 来插入和发挥使用。

例如，将以下代码添加到您的 `src/style.css` 文件，其中包含的 `margin-top` 、 `margin-right` 、 `margin-bottom` 和 `margin-left` 属性的简写:

```
.crip_shorthand {
    mt: 1rem;
    mr: 2rem;
    mb: 3rem;
    ml: 4rem;
}
```

编译您的代码，您应该看到你的 `dest/style.css` 代码是扩展后的:

```
.crip_shorthand {
    margin-top: 1rem;
```

```
    margin-right: 2rem;
    margin-bottom: 3rem;
    margin-left: 4rem;
}
```

有关于 `postcss-crip` 更多的信息，可以点击这里了解。另外更完整的属性缩写列表可以点击这里查阅。

一行输出@font-face

由 @Jonathan Neal 编写 `postcss-font-magician` 插件非常贴切其名称。它允许您使用简单的 `font-family` 规则来调用自定义字体，就好像你正在使用标准字体,而字体将会像施了魔术般地工作。

将以下代码添加到您的 `src/style.css` 文件:

```
body {
  font-family: "Indie Flower";
}
```

就是这样。这是使用 `postcss-font-magician` 所有的需要。 没有特殊的语法，只是使用的字体名称就好像你已经做了其它的工作。

在这个例子中，Indie Flower 是我选择的一个谷歌字体。 我还没在项目中添加任何自定义字体文件，所以该插件会查看指定的字体是否从谷歌的字体可用。 当它发现它是可用时，它将完全自动地创建适当的 `@font-face` 代码。

编译你的文件，然后再看看你的 `dest/style.css` 文件，它们已经把编译好的代码加入其中:

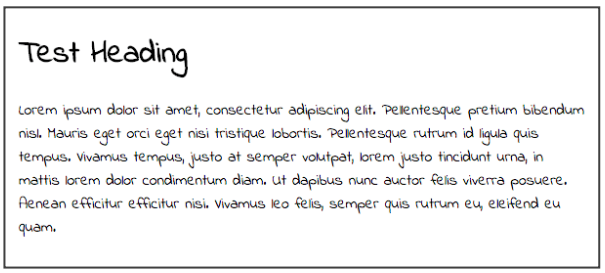
```
@font-face {
    font-family: "Indie Flower";
    font-style: normal;
    font-weight: 400;
    src: local("Indie Flower"),local(Indi
        url(//fonts.gstatic.com/s/indiefl
        url(//fonts.gstatic.com/s/indiefl
        url(//fonts.gstatic.com/s/indiefl
}
```

您可以通过在 `dest/` 文件夹中创建一个名为 `index.html` 的新文件 并将此代码添加到它里边，就可以检查该字体是否正确加载:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="style.cs
</head>
<body>

<h1>Test Heading</h1>

<p>Lorem ipsum dolor sit amet, consectetu

</body>
</html>
```

对于字体是否加载，您需要通过 `http://` 地址查看此文件，而不是一个 `file://` 地址，所以将该文件上传到 Web 主机或使用像 Prepros 的应用程序来创建实时预览。

您应该看到 Indie Flower 字体应用到你所有的文本，如下所示:



有关于 `postcss-font-magician` 更多信息，可以点击查阅。

创建 CSS 图形

如果你使用过纯CSS来创建形状，你就会知道这是一个相当棒的方法，包括像圆形和三角形的东西都可以实现，但是它是用起来也非常棘手。 尤其是三角形，要搞清楚需要哪些代码能得到一个正确方向，大小合适的形状，这些都可能有点不是很直观。

这个问题可以通过使用由 @Jed Mao 编写的 `postcss-circle` 和 `postcss-triangle` 得到缓解。 这两个插件给使用纯 CSS 创建圆形和三角形创建一个简化的语法和直观的方式。

创建圆

若要创建一个圆，使用的语法 `circle: size color;` 。 将以下代码添加到您的 `src/style.css` 文件:

```
.circle {
    circle: 8rem #c00;
}
```

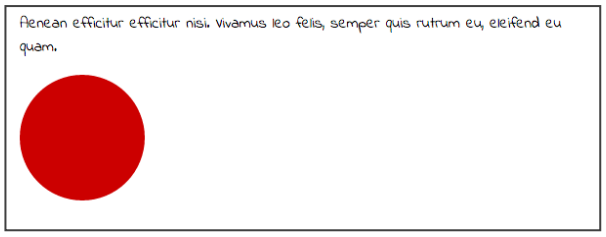
编译它，你就会看到下面的代码添加到您的 `dest/style.css` 文件中:

```
.circle {
    border-radius: 50%;
    width: 8rem;
    height: 8rem;
    background-color: #c00;
```


现在将此 HTML 添加到在上一部分创建的 `dest/index.html` 文件中:

```
<div class="circle"></div>
```

刷新一下在浏览器中打开的文件，现在您应该看到一个红色的圆圈:



有关于 `postcss-circle` 更详细的信息，可以点击这里阅读。

创建三角形

你可以使用此插件添加三种类型的三角形：等腰三角形、 直角腰三角形和等边三角形。 每个在语法的设置上都稍微有点不同，你可以在插件的 Github 仓库主页上查看完整信息。

我们会添加一个等腰三角形，它的语法是:

```
triangle: pointing-<up|down|left|right>;
width: <length>;
height: <length>;
background-color: <color>;
```

让我们将这个等腰三角形示例添加到 `src/style.css` 文件:


```
    height: 8rem;
    background-color: #c00;
}
```

编译的文件，现在您应该看到三角形的CSS代码在 `dest/style.css` 文件:

```
.isosceles-triangle {
    width: 0;
    height: 0;
    border-style: solid;
    border-color: transparent;
    border-width: 4rem 0 4rem 7rem;
    border-left-color: #c00;
}
```

在 `dest/index.html` 中添加这个三角形的HTML代码:

```
<div class="isosceles-triangle"></div>
```

浏览器刷新这个文件，现在应该可以看到一指向右边红色等腰三角形:

有关于 `postcss-triangle` 更多的信息，可以点击这里查阅。

对常见任务使用简写

设置链接样式

设置链接的颜色是在每个项目中都要做的工作，需要为链接设置默认样式和四种状态设置样式。 @Jed Mao 编写的 `postcss-all-link-colors` 插件可以让你快捷处理，一次输出链接所有状态的颜色。

将下面的代码添加到 `src/style.css` 中:

```
a {
    @link-colors all #4D9ACC;
}
```

然后编译您的文件，你会看到链接所有所需的状态都已设置:

```
a {
    color: #4D9ACC;
}

a:visited {
    color: #4D9ACC;
}

a:focus {
    color: #4D9ACC;
}

a:hover {
    color: #4D9ACC;
}

a:active {
    color: #4D9ACC;
}
```

也可以选择为特定状态生成不同的颜色。只是在规则末尾添加一些花括号，并且使用 `state: color;` 的语法插入其中。

更新的代码，你只需要添加以下代码到您的 `src/style.css` 文件:

```
a {
    @link-colors all #4D9ACC {
        hover: #5BB8F4;
    }
}
```

现在当你编译的时候你会看到悬停状态相比其余的样式有不同的颜色:

```
a {
    color: #4D9ACC
}

a:visited {
    color: #4D9ACC;
}

a:focus {
    color: #4D9ACC;
}

a:hover {
    color: #5BB8F4;
}

a:active {
    color: #4D9ACC;
}
```

有关于 `postcss-all-link-colors` 更详细的信息，可以点击这里查阅。

垂直或水平居中

垂直居中或水平居中，这些永远是CSS开发人员的恶梦之一。 @Jed Mao 的 `postcss-center` 插件使这项任务变得简单得多，使用 `top: center;` 来达到垂直居中，`left: center;` 达到水平居中。

将此代码添加到 `src/style.css` 文件:

```
.centered {
    top: center;
    left: center;
}
```

然后编译它，你会看到下面的代码:

```
.centered {
    position: absolute;
    top: 50%;
    left: 50%;
    margin-right: -50%;
    transform: translate(-50%, -50%);
}
```

注:这个居中使用了绝对定位，所以包裹它的父元素将需要使用相对定位，绝对定位或固定定位。 因为给定的绝对定位元素不会对他们父元素的高度或宽度产生影响，所以你需要自己设定父元素的高度或宽度进行适应。

例如，添加 `left: center;` 到之前写的 `.circle` 类中，这样他就会水平居中:

```
.circle {
    circle: 8rem #c00;
    left: center;
}
```

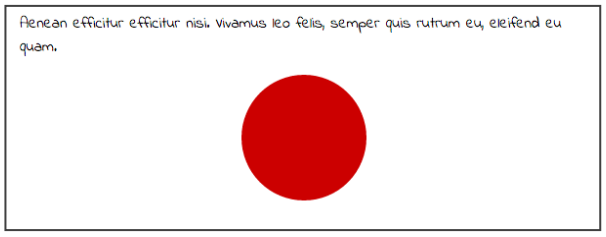
然后增加第二个类作为圆的容器，它可以为圆提供一个相对的定位和高度：

```
.circle_wrap {
    position: relative;
    height: 8rem;
}
```

现在增加一个元素作为HTML中已经存在的圆的容器：

```
<div class="circle_wrap">
    <div class="circle"></div>
</div>
```

当你刷新页面时，应该看到圆已经水平居中:



有关于 `postcss-center` 更多的信息，可以点击这里查阅。

一行输出清除浮动

在任何设计里使用浮动，清除浮动的固定类变得尤为方便且重要。 @Sean King 的 `postcss-clearfix` 插件可以通过一行创建清除浮动的样式，只需为 `clear` 属性使用 `fix` 值。

清除浮动将兼容到 IE8 及以上，添加以下代码到 `src/style.css` 文件:

```
.clearfixed {
    clear: fix;
}
```

通过编译，您将看到它生成了清除浮动的代码:

```
.clearfixed:after {
    content: '';
    display: table;
    clear: both;
}
```

如果需要兼容到 IE6 以上，要使用值不是 `fix` 而是 `fix-legacy` ，比如这样：

```
.clearfixed_legacy {
    clear: fix-legacy;
}
```

当代码被编译后，你会看到它包括一点额外的内容，这使它对旧版浏览器变的友好:

```
.clearfixed_legacy:before,
.clearfixed_legacy:after {
    content: '';
    display: table;
}

.clearfixed_legacy:after {
    clear: both;
}

.clearfixed_legacy {
  zoom: 1;
}
```

有关于 `postcss-clearfix` 更详细的信息，可以点击这里查阅。

一行设置定位

当你不想使用默认定位时，比如 `absolute` ， `fixed` 和 `relative` 。直到你安装@Sean King 编写的 `postcss-position` 插件前，你不得不手动写出元素的 `top` ， `right` ， `bottom` ， `left` 定

位。它没有像设置 `margin` 或 `padding` 那样在一行内设置的简写方法。

有了这个插件，当使用 `position` 属性时，设置值为 `absolute / fixed / relative` 后，您可以在同一行声明 `top`，`right`，`bottom` 和 `left` 的值。

将下面的示例代码添加到 `src/style.css` 文件:

```
.absolute {
    position: absolute 1rem 1rem 0 0;
}
```

编译之后, 你会看到简写的样式已经变成了需要你手写的多行代码：

```
.absolute {
    position: absolute;
    top: 1rem;
    right: 1rem;
    bottom: 0;
    left: 0;
}
```

这些值的声明方式与 `margin` 和 `padding` 具有相同的模式，比如，第一个值表示设置顶部和底部，第二个值设置左边和右边，或者你可以用一个值设置四边。

试试下面的代码:

```
.fixed_two_values {
    position: fixed 2rem 1rem;
```

```
}

.relative_one_value {
    position: relative 3rem;
}
```

编译出来的代码：

```
.fixed_two_values {
    position: fixed;
    top: 2rem;
    right: 1rem;
    bottom: 2rem;
    left: 1rem;
}

.relative_one_value {
    position: relative;
    top: 3rem;
    right: 3rem;
    bottom: 3rem;
    left: 3rem;
}
```

有关于 `postcss-position` 更详细的信息，可以点击这里查阅。

同时设置宽度和高度

@Andrey Sitnik 编写的 `postcss-size` 插件允许您缩减最常用的 `width` 和 `height` 属性，只需要写一个 `size` 属性。 你可以通过两种方式使用: 传两个值，第一个为宽度的值，第二个是长度的，或者传一个值，它将同时为宽高设置。

添加下面的 CSS 到 `src/style.css` 文件中来测试一下：

```
.size_a {
    size: 1rem 2rem;
}
```

```
.size_b {
    size: 1rem;
}
```

编译之后，在`dest/style.css`文件中可以看到下面的代码：

```
.size_a {
    width: 1rem;
    height: 2rem;
}

.size_b {
    width: 1rem;
    height: 1rem;
}
```

有关于`postcss-size`更详细的信息，可以点击这里查阅。

设置水平和垂直间距

作为崇尚编码效率爱好者，在编写外边距和内边距时，往往两侧是相等的。我希望有一个快捷方式一次可以声明水平方向和垂直方向间距。 我甚至写了几个 mixin 来做这件事情。 @David Hemphill 编写的`postcss-verthorz`插件，我再也不需要那些 mixin 了。

用这个插件可以使用`padding-vert`或者`margin-vert`设置垂直间距，使用`padding-horz`或者`margin-horz`设置水平间距。 将下面的代码示例添加到`src/style.css`文件中，看看它是如何工作的:

```
.spacing {
    padding-vert: 1rem;
```

```
    margin-horz: 2rem;
}
```

编译后，您将看到这些规则已经被扩展成内间距和外边距的声明方式了。

```
.spacing {
    padding-top: 1rem;
    padding-bottom: 1rem;
    margin-left: 2rem;
    margin-right: 2rem;
}
```

你还可以进一步简写这些属性到两个字符。 上面我们使用的示例代码可以缩写为以下形式，它们的输出将会完全一样:

```
.spacing_short {
    pv: 1rem;
    mh: 2rem;
}
```

有关于 `postcss-verthorz` 更详细的信息，可以点击这里查阅。

输出颜色代码

我最喜欢的默认文本颜色是 `#232323` ，我不知道是不是只有我这样，但我厌倦了反复推敲这些相同的两位数。 我经常希望有一个简写方式，像 `#ffffff` 可以缩减成 `#fff` 的方式一样。 @Andrey Polischuk 编写 `postcss-color-short` 插件就可以做到这一点。

当使用这个插件，你设置两位数颜色代码数字会重复进行，直到创建一个六位数的值，例如 `#23` 将成为 `#232323` 。 如

果您设置一个数字颜色代码，它将重复进行，直到有三个数字，例如 `#f` 将成为 `#fff`。 您甚至可以使用它来设置 `rgba()` 颜色，您传递的第一个数字会重复三次，第二个用来作为 `alpha` 值，如 `rgba (0, 0.5)` 将成为 `rgba (0, 0, 0, 0.5)`。

添加下面的到 `src/style.css` 文件来试一下上面所说的：

```
.short_colors {
    color: #23;
    background: #f;
    border-color: rgba(0, 0.5);
}
```

编译之后，你看到的所有颜色值都按其完整形式输出：

```
.short_colors {
    color: #232323;
    background: #fff;
    border-color: rgba(0, 0, 0, 0.5);
}
```

有关于 `postcss-color-short` 更详细的信息，可以点击这里查阅。

总结一下

让我们快速回顾一下在本教程中已经谈过的一切:

- 每天的编写的小代码看似独立，但是把它们加起来也会变的非常庞大，所以把它们进行简化编写是非常有意义的。

- `postcss-alias` 插件可以创建自定义的简写属性
- `postcss-crip` 插件有数百个预定义的属性简写
- `postcss-font-magician` 可以像使用默认字体来使用自定义字体，自动生成 `@font-face` 代码
- `postcss-circle` 和 `postcss-triangle` 插件可以直接地直观地创建圆和三角形
- `postcss-all-link-colors` 插件可以一次输出所有链接状态的颜色
- `postcss-center` 插件提供了使用 `top: center;` 和 `left: center;` 实现垂直和水平居中
- `postcss-clearfix` 插件使用 `clear: fix;` 输出代码清除浮动样式
- `postcss-position` 插件允许你同一行添加 `top`，`right`，`bottom` 和 `left` 作为 `position` 属性的值
- `postcss-size` 插件允许你一次性设置宽度和高度
- `postcss-verthorz` 插件允许你使用使用单一规则输出水平间距和垂直间距。
- `postcss-color-short` 插件使您能够使用一至两位数实现十六进制编码和其他颜色写法的简写。

在接下来的教程中

在接下来的教程中我们学习到一些很棒的插件，但是它们不属于任何特定的类别。我们很快会在 “Miscellaneous Goodies” 相见。

> 本文根据@Kezz Bracey的《PostCSS Deep Dive: Shortcuts and Shorthand》所译，整个译文带有我们自己的理解与思想，如果译得不好或有不对之处还请同行朋友指点。如需转载此译文，需注明英文出处：http://webdesign.tutsplus.com/tutorials/postcss-deep-dive-shortcuts-and-shorthand--cms-24602。

大漠

常用昵称“大漠”，W3CPlus创始人，目前就职于手淘。中国Drupal社区核心成员之一。对HTML5、CSS3和Sass等前端脚本语言有非常深入的认识和丰富的实践经验，尤其专注对CSS3的研究，是国内最早研究和使用CSS3技术的一批人。CSS3、Sass和Drupal中国布道者。2014年出版《图解CSS3：核心技术与案例实战》。

上一篇: Gulp系列教程：使用Sass（和Compass）编写CSS | 下一篇: Gulp系列教程：使用Browserify处理JavaScript

分享到：

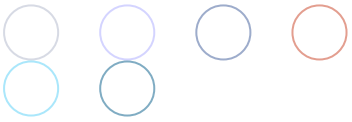
关于我们

W3cplus是一个致力于推广国内前端行业的技术博客。它以探索为己任，不断活跃在行业技术最前沿，努力提供高质量前端技术博文；其文章范围广泛，主要以CSS3、HTML5、Sass、Mobile和各类DEMO为主。

W3cplus具有一支强大的团队，提供长期的前端项目外包，Drupal建站，Drupal主题制作服务，以及提供企业广告展示与招聘发布，有需要的请联系：
QQ:81059347，E-mail:w3cplus@hotmail.com

关于站长

常用昵称“大漠”，W3CPlus创始人，目前就职于淘宝。中国Drupal社区核心成员之一。对HTML5、CSS3和Sass等前端脚本语言有非常深入的认识和丰富的实践经验，尤其专注对CSS3、Sass和Mobile的研究，是国内最早研究和使用CSS3和Sass技术的一批人。CSS3、Sass和Drupal中国布道者。2014年出版《图解CSS3：核心技术与案例实战》。



我的作品

本书是国内著名的Web前端专家历时2载的心血之作，根据最新的CSS3撰写，融入了作者在CSS领域多年的使用经验，旨在将本书打造成为CSS3领域最权威和实用的专业著作，供没有经验的读者系统学习，供有经验的读者参考备查。

本书理论知识系统全面，详细讲解了选择器、伸缩布局盒模型、渐变、过渡、动画等主题下涵盖的所有CSS3新特性。