分享三个纯CSS实现26个英文字母的案例

这篇文章发布于 2019年01月11日，星期五，11:30，归类于 CSS相关。 阅读 3432 次, 今日 28 次 10 条评论

一、借助CSS border实现案例

实现效果如下（为实时渲染效果）：

原作者是joshnh，出处是这里。

原作者实现的类名过于简单，用在实际项目中很容易冲突，我对其进行了优化——基于属性选择器`[data-char="*"]`标记对应字母。

如何使用?

引用CSS文件，例如：

```
<link rel="stylesheet" href="./css-letters1.css">
```

或者直接CSS代码到你的项目中，由于篇幅较长，我这里仅显示前几个字母的CSS样式，完整CSS代码见这里：

```
/* 全局样式 */
.letter {
    color: #2486ff;
    border-style: solid;
    border-width: .5em;
    display: inline-block;
    position: relative;
}
.letter:after {
    border-style: solid;
    border-width: .5em;
    content: '';
    position: absolute;
}
/* 单个字母样式 */
.letter[data-char="A"] {
    border-bottom: none;
    border-radius: 1em 1em 0 0;
    height: 2.05em;
    margin-top: -.05em;
    width: 1em;
}
```

```
.letter[data-char="A"]:after {
    border-bottom: none;
    border-left: none;
    border-right: none;
    left: 0;
    right: 0;
    top: .75em;
}
.letter[data-char="B"] {
    border-radius: 0 1em 1em 0;
    height: .5em;
    width: 1em;
}
.letter[data-char="B"]:after {
    border-radius: 0 1em 1em 0;
    bottom: 100%;
    height: .5em;
    left: -.5em;
    width: .9em;
}
.letter[data-char="C"] {
    border-right: none;
    border-radius: 1em 0 0 1em;
    height: 1.5em;
    width: 1.5em;
}
.letter[data-char="C"]:after {
    border-bottom: none;
    border-left: none;
    border-top: none;
    height: .5em;
    right: 0;
    top: 0;
    width: .5em;
}
...
```

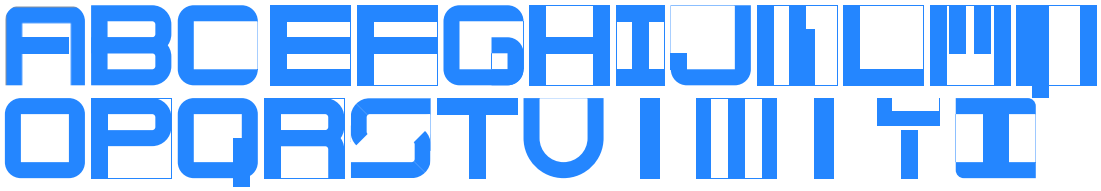
HTML部分如下：

```
<span class="letter" data-char="A"></span>
<span class="letter" data-char="B"></span>
<span class="letter" data-char="C"></span>
<span class="letter" data-char="D"></span>
<span class="letter" data-char="E"></span>
<span class="letter" data-char="F"></span>
<span class="letter" data-char="G"></span>
<span class="letter" data-char="H"></span>
<span class="letter" data-char="I"></span>
<span class="letter" data-char="J"></span>
<span class="letter" data-char="K"></span>
<span class="letter" data-char="L"></span>
<span class="letter" data-char="M"></span>
<span class="letter" data-char="N"></span>
<span class="letter" data-char="O"></span>
<span class="letter" data-char="P"></span>
<span class="letter" data-char="Q"></span>
<span class="letter" data-char="R"></span>
<span class="letter" data-char="S"></span>
<span class="letter" data-char="T"></span>
```

```
<span class="letter" data-char="U"></span>
<span class="letter" data-char="V"></span>
<span class="letter" data-char="W"></span>
<span class="letter" data-char="X"></span>
<span class="letter" data-char="Y"></span>
<span class="letter" data-char="Z"></span>
```

二、border加圆角与另一种风格字体

还是先看效果，实时渲染：



原作者是HKK，出处是这里。

原作者实现的到字母R就截止了，然后我就花了半小时仿照风格把后面STUVWXYZ这些字母都补全了，这样26个字母就一个不落了。

如何使用?

引用CSS文件，例如：

```
<link rel="stylesheet" href="./css-letters2.css">
```

或者直接复制CSS代码到你的项目中，由于篇幅限制，我这里仅显示前几个字母的CSS样式，完整CSS代码见这里：

```
.letter-a {
    position: relative;
    width: 30px;
    height: 40px;
    background: white;
    border-radius: 10px 10px 0 0;
    border-style: solid;
    border-color: currentColor currentColor transparent currentColor;
    border-width: 10px 10px 0 10px;
}
.letter-a::before {
    content: "";
    position: absolute;
    top: 10px;
    height: 10px;
    width: 30px;
    background: currentColor;
}

.letter-b {
    position: relative;
    width: 30px;
    height: 30px;
    border-width: 10px 10px 10px  10px;
    border-style: solid;
    border-color: transparent transparent transparent currentColor;
```

```
    background: transparent;
}
.letter-b::before {
    content: "";
    position: absolute;
    left: -10px;
    top: -10px;
    height: 10px;
    width: 30px;
    background: transparent;
    border-radius: 0 12.5px 12.5px 0;
    border: 10px solid currentColor;
}
.letter-b::after {
    content: "";
    position: absolute;
    left: -10px;
    bottom: -10px;
    height: 10px;
    width: 30px;
    background: transparent;
    border-radius: 0 12.5px 12.5px 0;
    border: 10px solid currentColor;
}
...
```

HTML部分代码使用示意：

```
<span class="letter-a"></span>
<span class="letter-b"></span>
<span class="letter-c"></span>
<span class="letter-d"></span>
<span class="letter-e"></span>
<span class="letter-f"></span>
<span class="letter-g"></span>
<span class="letter-h"></span>
<span class="letter-i"></span>
<span class="letter-j"></span>
<span class="letter-k"></span>
<span class="letter-l"></span>
<span class="letter-m"></span>
<span class="letter-n"></span>
<span class="letter-o"></span>
<span class="letter-p"></span>
<span class="letter-q"></span>
<span class="letter-r"></span>
<span class="letter-s"></span>
<span class="letter-t"></span>
<span class="letter-u"></span>
<span class="letter-v"></span>
<span class="letter-w"></span>
<span class="letter-x"></span>
<span class="letter-y"></span>
<span class="letter-z"></span>
```

每个字母都可以独立使用。

不过这里的实现有个不好的是，这里的字母都是使用px单位实现的，因此，想要自如控制字母的大小不

太方便。需要借助transform进行缩放控制才行。

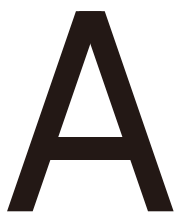
三、活用transform的css-sans字体生成

使用CSS生成的 无衬线26个英文字母。

原出处地址是这个。

实时效果如下：

//zxx: 接缝处有些间隙是因为对字体进行缩放导致，实际1:1呈现时候不会有这个现象。

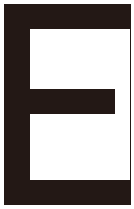


```
.A {
  position: relative;
  left: 30px;
  width: 60px;
  height: 91px;
  border-bottom: solid 14px #000000;
}
.A:before {
  transform: skew(-19deg, 0);
  position: absolute;
  content: '';
  top: 12.5px;
  left: 0;
  width: 16px;
  height: 125px;
  background-color: #000000;
}
.A:after {
  transform: skew(19deg, 0);
  position: absolute;
  content: '';
  top: 12.5px;
  left: 45px;
  width: 16px;
  height: 125px;
  background-color: #000000;
}
```

```
.B {
  position: relative;
  top: 12.5px;
  left: 10px;
  width: 60px;
  height: 125px;
  border-left: solid 16px #000000;
}
.B:before {
  position: absolute;
  content: '';
  width: 52px;
  height: 39px;
  border-width: 15px 15px 10px 0;
  border-color: #000000;
  border-style: solid;
  border-radius: 0 240% 180% 0 / 0 180% 180% 0;
}
.B:after {
  position: absolute;
  content: '';
  bottom: 0;
  width: 58px;
  height: 43px;
  border-width: 10px 15px 15px 0;
  border-color: #000000;
  border-style: solid;
  border-radius: 0 180% 220% 0 / 0 180% 180% 0;
}
```

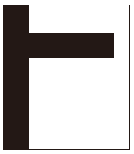
```css
.C {
  position: relative;
  top: 12.5px;
  left: 10px ;
  width: 84px;
  height: 95px;
  border-width: 15px 12px 15px 16px;
  border-color: #000000;
  border-style: solid;
  border-radius: 50%;
}
.C:before {
  transform: rotate(45deg);
  position: absolute;
  content: '';
  top: 2px;
  left: 49px;
  background-color: #ffffff;
  width: 90px;
  height: 90px;
}
```

```css
.D {
  position: relative;
  top: 12.5px;
  left: 10px;
  border-left: solid 15px #000000;
  height: 125px;
}
.D:before {
  position: absolute;
  content: '';
  top: 0;
  left: 0;
  width: 60px;
  height: 95px;
  border-width: 15px 15px 15px 0;
  border-color: #000000;
  border-style: solid;
  border-radius: 0 300% 300% 0 / 0 180% 180% 0;
}
```



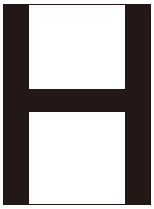
```css
.E {
  position: relative;
  top: 12.5px;
  left: 10px;
  width: 63px;
  height: 95px;
  border-width: 15px 0 15px 16px;
  border-color: #000000;
  border-style: solid;
}
.E:before {
  position: absolute;
  content: '';
  top: 38px;
  left: 0px;
  width: 53px;
  height: 15px;
  background-color: #000000;
}
```

```css
.F {
  position: relative;
  top: 12.5px;
  left: 10px;
  width: 63px;
  height: 110px;
  border-width: 15px 0 0 16px;
  border-color: #000000;
  border-style: solid;
}
```



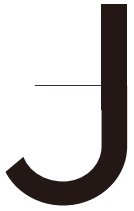
```
.F:before {
  position: absolute;
  content: '';
  top: 38px;
  left: 0px;
  width: 53px;
  height: 15px;
  background-color: #000000;
}
```

```
.G {
  position: relative;
  top: 12.5px;
  left: 10px;
  width: 84px;
  height: 95px;
  border-width: 15px 12px 15px 16px;
  border-color: #000000;
  border-style: solid;
  border-radius: 50%;
}
.G:before {
  transform: rotate(45deg);
  position: absolute;
  content: '';
  top: 2px;
  left: 48px;
  background-color: #ffffff;
  width: 90px;
  height: 90px;
}
 .G:after {
  position: absolute;
  content: '';
  bottom: 0.5px;
  right: 7px;
  width: 28px;
  height: 36px;
  border-width: 13px 14px 0 0;
  border-color: #000000;
  border-style: solid;
}
```

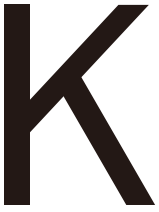


```
.H {
  position: relative;
  top: 12.5px;
  left: 10px;
  width: 60px;
  height: 125px;
  border-width: 0 16px 0 16px;
  border-color: #000000;
  border-style: solid;
}
.H:before {
  position: absolute;
  content: '';
  top: 53px;
  left: 0;
  width: 60px;
  height: 14px;
  background-color: #000000;
}
```

```css
.I {
  z-index: 1;
  position: relative;
  top: 12.5px;
  left: 20px;
  width: 16px;
  height: 125px;
  background-color: #000000;
}
```

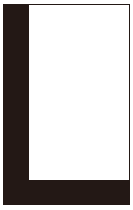


```css
.J {
  position: relative;
  top: 12.5px;
  left: -5px;
  width: 75px;
  height: 66px;
  border-right: solid 16px #000000;
}
.J:before {
  position: absolute;
  content: '';
  bottom: -60px;
  right: -16px;
  width: 50px;
  height: 60px;
  border-width: 0 16px  15px  14px;
  border-color: #000000;
  border-style: solid;
  border-radius: 0 0 75% 75%;
}
.J:after {
  transform: rotate(-40deg);
  position: absolute;
  content: '';
  top: 40px;
  left: -20px;
  width: 60px;
  height: 60px;
  background-color: #ffffff;
}
```



```css
.K {
  position: relative;
  top: 12.5px;
  left: 10px;
  width: 80px;
  height: 125px;
  border-left: solid 16px #000000;
  overflow: hidden;
}
.K:before {
  transform: skew(-43deg, 0);
  position: absolute;
  content: '';
  top: 0;
  left: 16px;
  width: 19px;
  height: 84px;
  background-color: #000000;
}
.K:after {
  transform: skew(30deg, 0);
  position: absolute;
  content: '';
  bottom: 0;
  right: 25px;
```

```
  width: 18px;
  height: 80px;
  background-color: #000000;
}
```



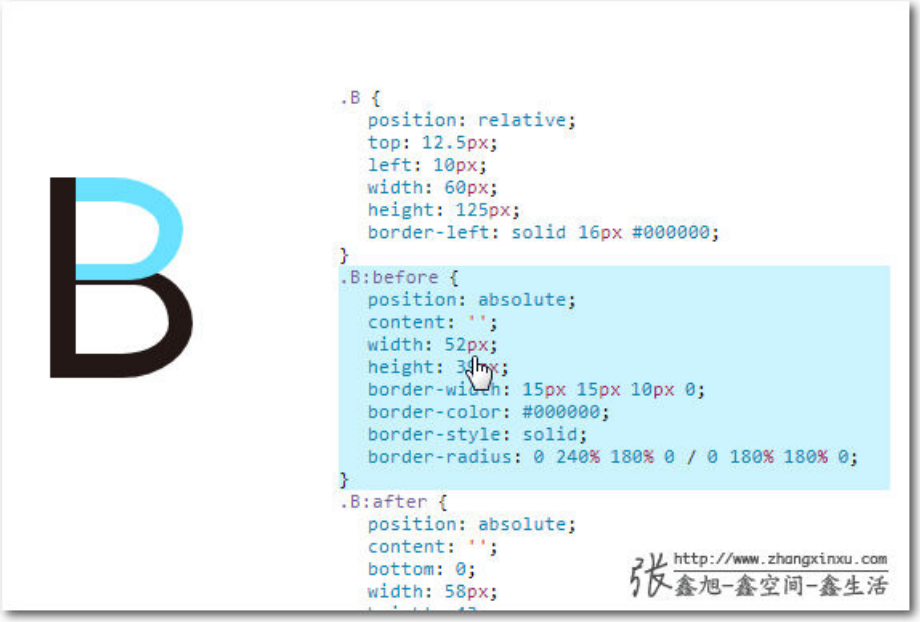
```
.L {
  position: relative;
  top: 12.5px;
  left: 10px;
  width: 63px;
  height: 110px;
  border-width: 0 0 15px 16px;
  border-color: #000000;
  border-style: solid;
}
```

```
.M {
  position: relative;
  top: 12.5px;
  left: 10px;
  width: 80px;
  height: 125px;
  border-width: 0 15px 0 15px;
  border-color: #000000;
  border-style: solid;
}
.M:before {
  transform: skew(20deg, 0);
  position: absolute;
  content: '';
  top: 0;
  left: 14px;
  width: 12px;
  height: 110px;
  background-color: #000000;
}
.M:after {
  transform: skew(-20deg, 0);
  position: absolute;
  content: '';
  top: 0;
  right: 14px;
  width: 12px;
  height: 110px;
  background-color: #000000;
}
```

```
.N {
  position: relative;
  top: 12.5px;
```

```
  left: 10px;
  width: 63px;
  height: 125px;
  border-width: 0 15px 0 15px;
  border-color: #000000;
  border-style: solid;
}
.N:before {
  transform: skew(30deg, 0);
  position: absolute;
  content: '';
  top: 0;
  left: 24px;
  width: 15px;
  height: 125px;
  background-color: #000000;
}
```

hover对应代码可以看到对应字体部件，非常使用CSS图形绘制的学习，如下截图示意：



如何使用？

复制页面上呈现的对应的CSS代码，然后HTML部分如下：

```
<div class="A"></div>
<div class="B"></div>
<div class="C"></div>
<div class="D"></div>
<div class="E"></div>
<div class="F"></div>
<div class="G"></div>
<div class="H"></div>
<div class="I"></div>
<div class="J"></div>
<div class="K"></div>
<div class="L"></div>
<div class="M"></div>
<div class="N"></div>
<div class="O"></div>
<div class="P"></div>
<div class="Q"></div>
<div class="R"></div>
<div class="S"></div>
<div class="T"></div>
<div class="U"></div>
<div class="V"></div>
<div class="W"></div>
<div class="X"></div>
```

```
<div class="Y"></div>
<div class="Z"></div>
```

要显示哪个字母，就复制对应HTML到页面上就好了。

四、点评与结束语

上面三个CSS生成26个字母的案例展示了CSS在图形绘制方面的潜力，是非常好的CSS图形绘制学习材料。

然而，要说具体的实用性，则并不见得多高，就像是顶级期刊的论文虽厉害，但并不适用于真正的商业实践，因为其中成本很好，适用场景页有限。

主要问题在于字母图形全部都是使用px进行定位的。而实际使用，我们的字号是多变的，px这种固定单位想要实时变化呈现的字号大小是很麻烦的，只能通过缩放解决，但缩放在1倍屏幕密度显示器下，容易出现接缝间隙，体验不好。

所以，上面的字体生成案例需要进一步优化，把`px`定位全部改成`em`，这样，就能通过外部`font-size`改变字形的大小，这样，实用性就很强了！

另外，上面的3个案例，全部都是大写英文字母，如果还支持小写字母，那就真正强悍了，实际项目中大肆应用是很有可能的。这个以后有时间我可以挑战下。

好了，就说这么多。

希望本文内容对您的学习有所帮助。

😉

> 《CSS世界》签名版独家发售，包邮，可指定寄语，点击显示购买码

（本篇完） // 想要打赏？点击这里。有话要说？点击这里。

« 小tips: 纯CSS实现打字动画效果

粉丝群第1期CSS小测点评与答疑 »

猜你喜欢

- 常见的CSS图形绘制合集
- CSS3图标图形生成技术个人攻略
- 秋月何时了，CSS3 border-radius知多少?
- CSS3 linear-gradient线性渐变实现虚线等简单实用图形
- 小tip: SVG和Canvas分别实现图片圆角效果
- 不借助Echarts等图形框架原生JS快速实现折线图效果
- 页面可用性之浏览器默认字体与CSS中文字体
- CSS font-feature-settings 50+关键字属性值完整介绍
- 纯CSS实现各类气球泡泡对话框效果
- 翻译：CSS中的糟粕

- 深入CSS ::first-letter伪元素及其实例等

分享到： 0

标签： border, border-radius, css相关, font, transform, 图形生成, 字符

发表评论（目前10条评论）

名称 (必须)

邮件地址(不会被公开) (必须)

网站

提交评论

1. 我们还太小说道：
2019年01月14日 14:34

是不是可以再弄个汉字库 (ω•̀ㅂ•́)و✧

回复

2. 风海流说道：
2019年01月13日 02:54

部分字缝间隙可以通过投 1px box-shadow 解决……

回复

3. 买卖货源说道：
2019年01月12日 11:25

居然不是字体 通过css写出来 强大

回复

4. **Leo**说道：
2019年01月11日 17:32

赞一个！

回复

5. **ChasLui**说道：
2019年01月11日 16:39

小说网站用着个是不是就不用怕被爬了

回复

6. **webxy说道：**
2019年01月11日 15:18

二、border加圆角与另一种风格字体 .
此处示例少字母 U

回复

> 张 鑫旭说道：
> 2019年01月12日 14:26
>
> 好的，感谢反馈，已补上。
>
> 回复

7. 多多欧巴说道：
2019年01月11日 14:53

学无止境

回复

8. **Hello说道：**
2019年01月11日 13:57

张大大加油啊！！

回复

9. **DeathGhost说道：**
2019年01月11日 13:25

你的想法很突出。哈哈👍👍👍👍👍

回复

最新文章

» 常见的CSS图形绘制合集
» 粉丝群第1期CSS小测点评与答疑
» 分享三个纯CSS实现26个英文字母的案例
» 小tips: 纯CSS实现打字动画效果
» CSS/CSS3 box-decoration-break属性简介
» CSS :placeholder-shown伪类实现Material Design占位符交互效果
» 从天猫某活动视频不必要的3次请求说起
» CSS vector-effect与SVG stroke描边缩放
» CSS ::backdrop伪元素是干嘛用的?
» 周知：CSS -webkit-伪元素选择器不再导致整行无效

今日热门

» 常见的CSS图形绘制合集 (74)
» 超级强大的SVG SMIL animation动画详解 (42)
» 小tips: 纯CSS实现打字动画效果 (40)
» 粉丝群第1期CSS小测点评与答疑 (36)
» 找到适合自己的前端发展方向 (35)
» 让所有浏览器支持HTML5 video视频标签 (32)
» 分享三个纯CSS实现26个英文字母的案例 (28)
» 好吧，CSS3 3D transform变换，不过如此！ (24)
» 未来必热：SVG Sprite技术介绍 (23)
» CSS/CSS3 box-decoration-break属性简介 (23)

今年热议

» 《CSS世界》女主角诚寻靠谱一起奋斗之人 (76)

» 不借助Echarts等图形框架原生JS快速实现折线图效果 (64)
» 看，for..in和for..of在那里吵架！ (60)
» 是时候好好安利下LuLu UI框架了！ (47)
» 原来浏览器原生支持JS Base64编码解码 (35)
» 妙法攻略：渐变虚框及边框滚动动画的纯CSS实现 (33)
» 炫酷H5中序列图片视频化播放的高性能实现 (31)
» CSS scroll-behavior和JS scrollIntoView让页面滚动平滑 (30)
» windows系统下批量删除OS X系统.DS_Store文件 (26)
» 写给自己看的display: flex布局教程 (26)

猜你喜欢

- 常见的CSS图形绘制合集
- CSS3图标图形生成技术个人攻略
- 秋月何时了，CSS3 border-radius知多少?
- CSS3 linear-gradient线性渐变实现虚线等简单实用图形
- 小tip: SVG和Canvas分别实现图片圆角效果
- 不借助Echarts等图形框架原生JS快速实现折线图效果
- 页面可用性之浏览器默认字体与CSS中文字体
- CSS font-feature-settings 50+关键字属性值完整介绍
- 纯CSS实现各类气球泡泡对话框效果
- 翻译：CSS中的槽粕
- 深入CSS ::first-letter伪元素及其实例等