

# CSS, SVG和canvas分别实现文本文字纹理叠加效果

这篇文章发布于 2018年02月27日, 星期二, 23:22, 归类于 [Canvas相关](#), [CSS相关](#), [SVG相关](#)。阅读 12007 次, 今日 5 次 [19 条评论](#)

by zhangxinxu from <http://www.zhangxinxu.com/wordpress/?p=7395>

本文可全文转载, 但需得到原作者书面许可, 同时保留原作者和出处, 摘要引流则随意。

本文这里所说的叠加, 就是混合模式中的叠加, 也就是说, 本文要实现的效果是, 文字本身的颜色和纹理进行叠加, 而非直接填充纹理。

CSS, SVG和canvas都能实现类似的效果, 我们一个一个来看一下。

## 一、CSS/CSS3实现文本纹理叠加

HTML和CSS代码如下:

```
<h2 class="pattern-overlay">
  <span data-text="CSS纹理叠加"></span>
  CSS纹理叠加
</h2>
```

```
.pattern-overlay {
  font-size: 60px;
  font-family: 'microsoft yahei';
  background-image: url(./pattern01.jpg);
  -webkit-text-fill-color: transparent;
  -webkit-background-clip: text;
}
.pattern-overlay > span {
  position: absolute;
  background-image: linear-gradient(to bottom, #f00, #f00);
  mix-blend-mode: overlay;
  -webkit-text-fill-color: transparent;
  -webkit-background-clip: text;
}
.pattern-overlay > span::before {
  content: attr(data-text);
}
```

就可以实现类似下图的效果 (红色渐变和灰色石质纹理叠加效果):

# CSS纹理叠加

您可以狠狠的点击这里: [CSS实现文本的纹理叠加效果demo](#)

在demo页面中, 我们可以调整渐变图片的起止颜色, 或者更换我们的纹理图片, 都有实时的渲染效果:

实现原理

而在webkit浏览器下，可以通过下面CSS组合实现文本以背景显示效果：

```
.fill-bg {  
    -webkit-text-fill-color: transparent;  
    -webkit-background-clip: text;  
}
```

可以用来实现文字渐变，或者类似本站首页文字流光等效果。

购买：签名版 京东自营 当当自营 亚马逊 异步社区

于是，我们使用两层标签，分明填充渐变背景和纹理背景，然后再使用CSS3混合模式 `mix-blend-mode: overlay` 对两层标签进行叠加，效果即达成！

## 兼容性

webkit内核浏览器，Chrome，Safari等都支持。

## 关于为何不使用background-blend-mode说明

理论上，使用 `background-blend-mode` 最多背景图片进行模式混合是最简单的，因为只需要一层表现，理论支持代码如下：

```
<h2 class="pattern-overlay">CSS纹理叠加</h2>
```

```
.pattern-overlay {  
    font-size: 60px;  
    font-family: 'microsoft yahei';  
    background-image: linear-gradient(to bottom, #f00, #f00), url(./pattern01.jpg);  
    background-blend-mode: overlay;  
    -webkit-text-fill-color: transparent;  
    -webkit-background-clip: text;  
}
```

背景渐变和纹理叠加本身是没有任何问题的，效果如下截图：



但是当应用 `background-clip: text` 声明的时候，混合模式被忽略，于是最终的文本并没有叠加效果。因此才采用两层独立的标签应用 `mix-blend-mode` 来叠加的方法。

//zxx: CSS3对混合模式天然支持，可以参见这篇文章：[“CSS3混合模式mix-blend-mode/background-blend-mode简介”](#)，其中 `mix-blend-mode` 是元素间的混合，`background-blend-mode` 是背景图片之间的混合。

## 二、使用SVG实现更加兼容的文本纹理叠加效果

CSS3的方法最容易理解上手最快，但是Firefox和IE浏览器都不支持，所以只能在移动端使用，如果我们想兼容PC浏览器，可以试试使用SVG来实现，代码如下：

```
<svg width="360" height="120">  
    <defs>  
        <filter id="blend">  
            <feImage xlink:href="./pattern01.jpg" result="patternSource" x="0" y="0" width="360" height="120" />  
            <feBlend mode="overlay" in="SourceGraphic" in2="patternSource" />  
        </filter>  
    </defs>  
</svg>
```

```

</filter>
<linearGradient id="gradient" x1="0" y1="0" x2="0" y2="1">
  <stop offset="0%" stop-color="green" />
  <stop offset="100%" stop-color="red" />
</linearGradient>
<pattern id="pattern" width="360" height="120" patternUnits="userSpaceOnUse">
  <rect x="0" y="0" width="100%" height="100%" fill="url(#gradient)" filter="url(#blend)"></rect>
</pattern>
</defs>
<text x="0" y="60" font-family="Microsoft Yahei" font-size="60" font-weight="900" fill="url(#pattern)">
  SVG纹理叠加
</text>
</svg>

```

红绿渐变叠加石头质感纹理，最终实现的效果如下截图：



您可以狠狠地点击这里：[SVG实现文本的纹理叠加效果demo](#)

## 实现原理

SVG中有个和混合模式相关的滤镜元素名为 `<feBlend>`，所以我们可以定义一个 `<filter>`，让引用该 `<filter>` 的图形（`in="SourceGraphic"`）和 `<feImage>` 这个图片（`in2="patternSource"`）进行 `overlay` 混合（`mode="overlay"`）。

SVG中文本除了可以填充颜色外，还可以填充纹理，元素是 `<pattern>`，所以，我们需要一个渐变和纹理素材混合的 `<pattern>` 元素，很简单，一个和SVG尺寸一样的矩形 `<rect>` 元素，使用渐变填充，应用叠加滤镜，作为 `<pattern>` 纹理。

于是，效果达成！

## 兼容性

Chrome, Safari, Firefox浏览器都支持。

如果在IE浏览器下访问我们的demo页面，会看到纹理并没有叠加，那是因为，IE浏览器下的 `<feBlend>` 只支持规范中提到的几种混合模式，包括：`normal`，`multiply`，`screen`，`darken`，`lighten`。并没有叠加 `overlay`。

因此，如果你希望SVG纹理叠加效果IE9+全兼容，可以试试使用正片叠加混合模式-`multiply`，对于大部分用户而言，是看不出什么差异的。

## 三、使用canvas实现纹理叠加效果

canvas并没有现成的混合模式api，因此，如果要想实现叠加效果，需要通过算法重新计算方法。关于混合模式的各种算法，实际上都是公开的，搜一搜就能找到。

在本文中，canvas的混合模式效果使用了一个开源的JS方法，项目地址是：<https://github.com/Phrogz/context-blender>

JS未压缩状态也就9K不到，各种曾经的混合模式都支持。

于是，要使用canvas实现纹理叠加效果那就容易多了。

下面是实现的效果截图：



您可以狠狠地点击这里：[canvas实现文本的纹理叠加效果demo](#)

同样的，我们可以修改渐变颜色，修改纹理图片看到其他渲染效果，例如，我们选择本地一张纸张图片作为纹理：



## 实现原理

绘制JS代码如下：

```
// 先引入context_blender.js, 然后.....
// canvas绘制脚本
var canvas = document.querySelector('canvas');
var context = canvas.getContext('2d');
var width = canvas.width, height = canvas.height;
context.textBaseline = 'middle';
context.font = 'bold 60px "Microsoft Yahei"';
// 绘制方法
var draw = function () {
    context.clearRect(0, 0, width, height);
    // 渐变和纹理
    var gradient, pattern;
    // 创建材质canvas
    var canvasPattern = document.createElement('canvas');
    var contextUnder = canvasPattern.getContext('2d');
    canvasPattern.width = width;
    canvasPattern.height = height;

    // 创建渐变canvas
    var canvasGradient = document.createElement('canvas');
    var contextOver = canvasGradient.getContext('2d');
    canvasGradient.width = width;
    canvasGradient.height = height;
    // 绘制渐变对象
    gradient = contextOver.createLinearGradient(0, 0, 0, height);
    gradient.addColorStop(0, red);
    gradient.addColorStop(1, red);

    // 纹理对象, img指纹理图片对象
    pattern = contextUnder.createPattern(img, 'no-repeat');
    contextUnder.fillStyle = pattern;
    contextUnder.fillRect(0, 0, width, height);
    // 应用渐变
    contextOver.fillStyle = gradient;
    contextOver.fillRect(0, 0, width, height);
    // 叠加canvas
    contextOver.blendOnto(contextUnder, 'overlay');

    // 给当前context创建pattern
    pattern = context.createPattern(canvasPattern, 'no-repeat');

    // 绘制文本
    context.fillStyle = pattern;
    context.fillText('画布纹理叠加', 0, 60);
}
```

```
};
```

原理描述：

临时创建一个canvas绘制一个渐变，临时创建一个canvas使用纹理图片填充，两个canvas叠加混合得到新的canvas，然后页面上那个canvas上的文字就把这个叠加混合后canvas作为纹理进行填充，效果即达成。

兼容性

IE9+，Chrome, Safari, Firefox浏览器都支持。

## 四、结束语

本文所有案例，无论是CSS3，SVG还是canvas实现都是以叠加混合模式 `overlay` 示意的。其他各种混合模式也都可以轻松支持，只要修改 `overlay` 为其他关键字即可，就不一一举例了。

一点抛砖引玉，以上就是自己根据一些经验摸索出来的方法，实际上应该有更好的方法实现纹理叠加效果，欢迎不吝赐教。

感谢您花宝贵的时间阅读到这里！

《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: 使用JS检测用户是否安装某font-family字体](#)

[CSS3 :default伪类选择器简介](#) »

猜你喜欢

- CSS3混合模式mix-blend-mode/background-blend-mode简介
- -webkit-box-reflect属性简介及元素镜像倒影实现
- 深入理解SVG feDisplacementMap滤镜及实际应用
- 小tip: CSS3下的渐变文字效果实现
- 小tip: CSS3与文字渐变光影流动动画效果实现
- “蝉原则”与CSS3随机多背景随机圆角等效果
- 小tips:使用canvas在前端实现图片水印合成
- 搞懂SVG/Canvas中nonzero和evenodd填充规则
- 深入理解CSS中的层叠上下文和层叠顺序
- 理解CSS3 isolation: isolate的表现和作用
- 纯CSS实现任意格式图标变色的研究

分享到: [+](#) [QQ](#) [微信](#) [微博](#) [贴吧](#) [收藏](#) [0](#)

标签: background-blend-mode, feBlend, feImage, fill, fillText, filter, gradient, mix-blend-mode, pattern, 文字渐变, 混合模式, 滤镜, 纹理

## 发表评论（目前19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. **alice**说道：

2018年03月19日 09:14

学习了，膜拜大神

[回复](#)



2. **Web前端之家**说道：

2018年03月14日 17:26

学习

[回复](#)



3. **东元马达**说道：

2018年03月14日 16:00

多谢，正有需要

[回复](#)



4. **重庆崽儿Brand**说道：

2018年03月8日 17:16

大佬博客干货很多，学习了

[回复](#)



5. **东元**说道：

2018年03月8日 08:59

都是干货，但是一般人难以坚持

[回复](#)

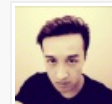


6. **Alvin**说道：

2018年03月7日 11:28

太厉害

[回复](#)



7.

东元伺服电机说道：  
2018年03月7日 11:04  
  
可以的，慢慢的来搞还是没问题的  
  
[回复](#)



8.

weiya说道：  
2018年03月6日 11:26  
  
你好，我想问您一个问题，怎么用js获得一个元素默认的风格值。比如说input这个元素我没有显式设置他的display，我怎么通过js去获得这个元素默认的display值。我看chrome的用户代理默认是inline-block，而firefox的是inline，我不论是用jq还是JavaScript得到的都是“”  
  
[回复](#)

weiya说道：  
2018年03月6日 11:35  
  
刚才一直用jq的attr(),试了下css()可以啦，哈哈  
  
[回复](#)







9.

刘东奇说道：  
2018年03月5日 08:30  
  
终于可以实现五彩斑斓的黑了  
  
[回复](#)

刘东奇说道：  
2018年03月9日 15:11  
  
调皮  
  
[回复](#)





10.

Retro说道：  
2018年03月3日 17:01  
  
膜拜大神已久，之前模仿你的网站框架做了一个个人网站http://2tro.com/index01.aspx，你会不会打我~~~~\_~~##@#@#@@  
  
[回复](#)

刘东奇说道：  
2018年03月5日 08:32  
  
厉害厉害  
  
[回复](#)



蚊子老大说道：  
2018年03月13日 10:01  
  
首页视频播放器可以更新H5了，嘻~~~~  
  
[回复](#)



长江长江我是黄河说道：  
2018年03月16日 16:17  
  
你可以的  
  
[回复](#)



11.

karen说道：

2018年03月3日 12:35

数个大拇指！

回复


12.

luoky说道：

2018年03月1日 15:40

厉害了小哥哥

回复


13.

山佳说道：

2018年03月1日 14:10

遇到这种问题 我一般都是：ui这里切一张图

回复

大曲说道：

2018年03月19日 17:54

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](#) <sup>(178)</sup>
- » [粉丝群第1期CSS小测点评与答疑](#) <sup>(112)</sup>
- » [未来必热：SVG Sprite技术介绍](#) <sup>(111)</sup>
- » [HTML5终极备忘大全（图片版+文字版）](#) <sup>(85)</sup>
- » [让所有浏览器支持HTML5 video视频标签](#) <sup>(83)</sup>
- » [Selectivizr-让IE6~8支持CSS3伪类和属性选择器](#) <sup>(80)</sup>
- » [CSS3下的147个颜色名称及对应颜色值](#) <sup>(78)</sup>
- » [小tips: 纯CSS实现打字动画效果](#) <sup>(72)</sup>
- » [写给自己看的display: flex布局教程](#) <sup>(69)</sup>
- » [分享三个纯CSS实现26个英文字母的案例](#) <sup>(69)</sup>

#### 今年热议

- » [《CSS世界》女主角诚寻靠谱一起奋斗之人](#) <sup>(76)</sup>
- » [不借助Echarts等图形框架原生JS快速实现折线图效果](#) <sup>(64)</sup>
- » [看，for..in和for..of在那里吵架！](#) <sup>(60)</sup>
- » [是时候好好安利下LuLu UI框架了！](#) <sup>(47)</sup>
- » [原来浏览器原生支持JS Base64编码解码](#) <sup>(35)</sup>
- » [妙法攻略：渐变虚框及边框滚动动画的纯CSS实现](#) <sup>(33)</sup>
- » [炫酷H5中序列图片视频化播放的高性能实现](#) <sup>(31)</sup>
- » [CSS scroll-behavior和JS scrollIntoView让页面滚动平滑](#) <sup>(30)</sup>
- » [windows系统下批量删除OS X系统.DS\\_Store文件](#) <sup>(26)</sup>
- » [写给自己看的display: flex布局教程](#) <sup>(26)</sup>



## 猜你喜欢

- CSS3混合模式mix-blend-mode/background-blend-mode简介
- -webkit-box-reflect属性简介及元素镜像倒影实现
- 深入理解SVG feDisplacementMap滤镜及实际应用
- 小tip:CSS3下的渐变文字效果实现
- 小tip: CSS3与文字渐变光影流动动画效果实现
- “蝉原则”与CSS3随机多背景随机圆角等效果
- 小tips:使用canvas在前端实现图片水印合成
- 搞懂SVG/Canvas中nonzero和evenodd填充规则
- 深入理解CSS中的层叠上下文和层叠顺序
- 理解CSS3 isolation: isolate的表现和作用
- 纯CSS实现任意格式图标变色的研究