Canvas中颜色过渡动画效果的实现

这篇文章发布于 2018年07月22日，星期日，21:14，归类于 Canvas相关。 阅读 5665 次, 今日 10 次 3 条评论

一、关于颜色动画的前言

在CSS3中，我们要想实现从A颜色到B颜色的过渡效果，是相当容易的，只要指定起止颜色，配合 `transition` 过渡或者 `animation` 动画都可以时间我们想要的效果。

但是，在 `<canvas>` 中却没有这么简单，因为 `<canvas>` 本质上是一个静态画布，要想实现颜色变化，需要JS去不断绘制，实现起来要比CSS实现麻烦很多。再加上颜色值本身就不一定是纯粹的数值，更增加了我们实现的难度。

本文就将通过多个案例，逐步深入，介绍一些在Canvas中的颜色处理技巧，有些技巧说不定会让你大开眼界。

二、Canvas颜色动画硬实现

所谓“硬实现”，就是不通过“奇巧淫技”，按照正常逻辑，一条路走到底的实现方式。

例如，我们要在 `<canvas>` 中实现一个蓝色到红色的动画效果，思路如下：

1. 获取蓝色和红色的RGB色值；
2. 使用定时器，配合运动算法，R，G，B数值同时变化产生变色。

具体如何实现，可以参见这个案例。您可以狠狠地点击这里：颜色数值变化下的颜色动画demo

实现的效果如下GIF截图（上面是Canvas实现，下面是CSS3实现，方便效果对比）：

完整JS代码如下：

```
var canvas = document.getElementById('canvas');
var context = canvas.getContext('2d');
var width = canvas.width, height = canvas.height;
// 动画执行的帧数
var start = 0, frames = 200;
// 过渡颜色 蓝色 到 红色
var from = [0, 0, 255];
var to = [255, 0, 0];
// 动画算法，这里使用Cubic.easeOut算法
var cubicEaseOut = function(t, b, c, d) {
    return c * ((t = t/d - 1) * t * t + 1) + b;
};
// 绘制方法
var draw = function () {
```

```
    context.clearRect(0, 0, width, height);
    // 计算此时r, g, b数值
    var r = cubicEaseOut(start, from[0], to[0] - from[0], frames);
    var g = cubicEaseOut(start, from[1], to[1] - from[1], frames);
    var b = cubicEaseOut(start, from[2], to[2] - from[2], frames);
    // 可以确定色值
    context.fillStyle = 'rgb('+ [r, g, b].join() +')';
    context.arc(width / 2, height / 2, height / 2, 0, 2 * Math.PI);
    context.fill();
    // 持续变化
    start++;
    if (start <= frames) {
        requestAnimationFrame(draw);
    }
};
draw();
```

“硬实现”的局限所在

“硬实现”虽然基本效果有了，但是也存在一些局限，主要就是我们前后变化的色值不一定正好就是RGB颜色，可能是`#RRGGBB`这种十六进制色值，或者`hsl`颜色，这个还好，我们可以色值转换下（转换方法参见“HEX十六进制与RGB, HSL颜色的相互转换”这篇文章）；还有可能颜色是半透明的RGBA或者HSLA颜色，这么也还行，咬咬牙，也可以实现，不就是多一个透明度变化的数值参数嘛；然而，最怕的就是色值是147个颜色关键字之一，例如请实现`blue`到`red`的效果。是不是只能望码兴叹了。

色值转换麻烦，关键字变化无能为力，有没有什么技巧可以一次性搞定呢？有！

三、任意颜色转RGB色值技巧

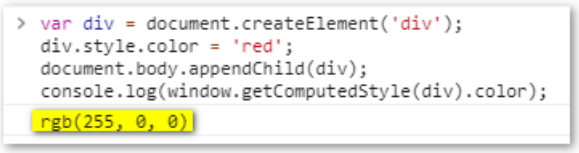
所有web浏览器有这么一个特性，当我们使用`getComputedStyle`原生接口去获取DOM元素的色值的时候，返回的全部都是RGB色值或者RGBA色值。

举个例子，如下JavaScript代码：

```
var div = document.createElement('div');
div.style.color = 'red';
document.body.appendChild(div);
console.log(window.getComputedStyle(div).color);
```

结果是：`rgb(255, 0, 0)`

并不是`'red'`。



于是，借助浏览器特性，无论什么颜色值，想要在canvas中有动画效果，都不成问题了。我们可以封装一个任意色值转RGBA颜色的方法，如下：

```
/**
* 任意颜色转RGBA方法
*/
var toRGBA = function (color) {
```

```
    // 创建div元素并设置颜色
    var div = document.createElement('div');
    div.style.color = color;
    document.body.appendChild(div);
    // 返回计算后的颜色值
    var cssColor = window.getComputedStyle(div).color;
    // div元素移除
    document.body.removeChild(div);
    // 如果是RGB颜色，则转换成RGBA表示
    var arrRGBA = cssColor.match(/\d+/g);
    if (arrRGBA.length == 3) {
        arrRGBA.push(1);
    }
    // 每一项转换成数值类型
    return arrRGBA.map(function (value) {
        return value * 1;
    });
};
```

于是乎，关键字色值的转换效果实现也不在话下了，您可以狠狠地点击这里：canvas任意色值下的过渡动画demo

实现的是深天空蓝`'deepskyblue'`到佩奇粉`'deeppink'`颜色动画效果，如下GIF示意：

JavaScript代码如下：

```
var canvas = document.getElementById('canvas');
var context = canvas.getContext('2d');
var width = canvas.width, height = canvas.height;
// 动画执行的帧数
var start = 0, frames = 200;
// 过渡颜色 天空蓝 到 佩奇粉
var from = toRGBA('deepskyblue');
var to = toRGBA('deeppink');
// 动画算法，这里使用Cubic.easeOut算法
var cubicEaseOut = function(t, b, c, d) {
    return c * ((t = t/d - 1) * t * t + 1) + b;
};
// 绘制方法
var draw = function () {
    context.clearRect(0, 0, width, height);
    // 计算此时r, g, b, a数值
    var r = cubicEaseOut(start, from[0], to[0] - from[0], frames);
    var g = cubicEaseOut(start, from[1], to[1] - from[1], frames);
    var b = cubicEaseOut(start, from[2], to[2] - from[2], frames);
    var a = cubicEaseOut(start, from[3], to[3] - from[3], frames);
    // 可以确定色值
    context.fillStyle = 'rgba('+ [r, g, b, a].join() +')';
    context.arc(width / 2, height / 2, height / 2, 0, 2 * Math.PI);
    context.fill();
    // 持续变化
    start++;
    if (start <= frames) {
        requestAnimationFrame(draw);
    }
};
draw();
```

依然存在的局限

目前Canvas中从A色到B色的过渡动画效果已经可以实现了，但是，如果我们的颜色变化是丰富的有层次的，则我们的实现又变得麻烦了。

例如，要实现红橙黄绿青蓝紫7色渐变，且每种过渡所占时间都不一样，该怎么办?

如果是CSS3实现，则很简单，定义一个`animation`动画就可以，类似这样：

```
@keyframes color {
    0% { background-color: red; }
    7% { background-color: orange; }
   17% { background-color: yellow; }
   22% { background-color: green; }
   42% { background-color: cyan; }
   82% { background-color: blue; }
   90% { background-color: purple; }
}
```

如果要在Canvas中实现，我去，不敢想象，代码估计会很啰嗦。有没有什么简单取巧的方法实现复杂颜色动画效果呢？有！

四、借助CSS3实现复杂canvas动画

无论是CSS3 `transition`过渡或者`animation`动画，我们都可以使用JS实时返回当前动画中的颜色值等CSS属性值，于是，我们根本就不需要在Canvas中“硬实现”动画效果，直接把DOM中的变化量实时赋值就好了。

您可以狠狠地点击这里：借助CSS3 animation实现canvas颜色动画demo

效果如下GIF：

我们的DOM元素（下JS代码中的`eleShape`）应用CSS3动画：

```
.shape.active {
    animation: color 3.33s both;
}
```

JS代码就简单多了，实时色值一致即可：

```
// CSS3动画元素
var eleShape = document.getElementById('shape');
// Canvas元素
var canvas = document.getElementById('canvas');
var context = canvas.getContext('2d');
var width = canvas.width, height = canvas.height;
// 绘制方法
var draw = function () {
    context.clearRect(0, 0, width, height);
    // 可以确定色值
    context.fillStyle = window.getComputedStyle(eleShape).backgroundColor;
    context.arc(width / 2, height / 2, height / 2, 0, 2 * Math.PI);
    context.fill();
```

```
    // 持续变化
    requestAnimationFrame(draw);
};
draw();
```

最关键的其实就是上面红色高亮的那一行JS代码，也就是Canvas中的圆圈的填充色等于此刻DOM元素背景色即可！

同理，本文展示的头两个案例也可以使用这种方法实现我们想要的Canvas动画效果，省去了计算、还有转换这些累心烦人的操作，真正意义上的人人都可以上手，只要你会一点CSS动画即可。

五、进一步扩展

其实不仅仅是颜色变化，其他变化，例如位移，缩放等都可以通过DOM+CSS3动画“移花接木”实现，我们可以新建一个和`<canvas>`尺寸一样大的容器，里面DOM进行CSS3过渡或动画，Canvas中的图形与之实时匹配，例如`x`，`y`匹配DOM元素的`left`，`top`定位值；至于缩放效果，Canvas中的图形的宽高尺寸实时等同于DOM元素的可视尺寸即可。

只不过从性价比上而言，颜色是是最高，因为颜色值有N多种表示方法，而不仅仅是一个单纯的数值，借助DOM特性实现相当于踩在了浏览器的肩膀上，很多工作浏览器已经帮我们完成了，我们工作会轻松很多。

再扯点远的

知道技术细节有什么用？

无论是jQuery的`$().css()`方法还是这里的`getComputedStyle`返回的色值都是RGB或RGBA，想必这个很多人都没注意到，更不知道兼容性是怎样的。实际上兼容性非常好，所有浏览器行为都一致。大多数时候，99.9%的时候，知道这个技术细节是没什么用的，因为获取颜色再赋值颜色，管他HEX，HSL还是RGB，效果都能正常显示。

但是，应用场景千千万，类似本文Canvas要实现颜色关键字的动画效果，如果不清楚浏览器有API天然转换，那可真是寸步难行了，估计要建一个偌大的color关键字rgb色值映射表，呵呵，那方向就走错了！

所以，掌握技术细节，尤其是大量的事无巨细的技术细节，可以在关键时刻给你提供更多的技术实现思路，可以保证你的技术实现几乎最佳，可以少走弯路，有不一样的技术创造力，这个可是别人无法超越的技术竞争力所在。

补充于1小时后
微博达人@王集鹄提供了另外一种Canvas关键字色值动画实现思路，挺好的，大家可以学习下。

就是利用 Canvas 的线性渐变生成渐变色的调色板，然后就能兼顾性能和颜色字面量的解析。

代码实现主要部分截图：

在线demo点击这里。

感谢阅读，重要的是实现思路，欢迎交流！

（本篇完） // 想要打赏？点击这里。有话要说？点击这里。

« 一行CSS实现滚动时藏在信息流后面的广告效果

深入理解CSS的width:auto »

猜你喜欢

- CSS前景背景自动配色技术简介
- Chrome opacity非1时border-radius圆角边框剪裁问题
- JS HEX十六进制与RGB, HSL颜色的相互转换
- CSS3下的147个颜色名称及对应颜色值
- 小tip: 了解LinearRGB和sRGB以及使用JS相互转换
- 内容loading加载后高度变化CSS3 transition体验优化
- 获取元素CSS值之getComputedStyle方法熟悉
- 伪类+js实现CSS3 media queries跨界准确判断
- CSS :visited伪类选择器隐秘往事回忆录
- 小tips: 如何HTML标签和JS中设置CSS3 var变量
- jquery之图片左右切换动画效果

标签： animation, getComputedStyle, hsla, RGB, rgba, 动画, 颜色

发表评论（目前3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. **manguo说道：**
   2018年08月17日 09:47

   又大大的学习了，编程思想很重要

   回复

2. **Eric Time说道：**
2018年08月3日 17:10

不知道有没有人发现 canvas 和 css 画出来的颜色不一样。。。。

回复

> **张 鑫旭说道：**
> 2018年08月3日 20:05
>
> 因为动画缓动函数用的不一样，执行时候有细节差异。
>
> 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 (178)
» 粉丝群第1期CSS小测点评与答疑 (112)
» 未来必热：SVG Sprite技术介绍 (111)
» HTML5终极备忘大全（图片版+文字版） (85)
» 让所有浏览器支持HTML5 video视频标签 (83)
» Selectivizr-让IE6~8支持CSS3伪类和属性选择器 (80)
» CSS3下的147个颜色名称及对应颜色值 (78)
» 小tips: 纯CSS实现打字动画效果 (72)
» 写给自己看的display: flex布局教程 (69)
» 分享三个纯CSS实现26个英文字母的案例 (69)

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

- CSS前景背景自动配色技术简介
- Chrome opacity非1时border-radius圆角边框剪裁问题
- JS HEX十六进制与RGB, HSL颜色的相互转换
- CSS3下的147个颜色名称及对应颜色值
- 小tip: 了解LinearRGB和sRGB以及使用JS相互转换
- 内容loading加载后高度变化CSS3 transition体验优化
- 获取元素CSS值之getComputedStyle方法熟悉
- 伪类+js实现CSS3 media queries跨界准确判断
- CSS :visited伪类选择器隐秘往事回忆录

- 小tips: 如何HTML标签和JS中设置CSS3 var变量
- jquery之图片左右切换动画效果