canvas 2D炫酷动效的实现套路和需要的技术积累

这篇文章发布于 2017年03月18日，星期六，21:22，归类于 Canvas相关。 阅读 28007 次, 今日 13 次 18 条评论

by zhangxinxu from http://www.zhangxinxu.com/wordpress/?p=6017
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。

一、放在开头的结束语

我上学时候，06，07年的样子，总是情不自禁被炫酷的flash动效吸引。印象很深的是一个一堆线和球连在一起弹来弹去的效果，觉得还厉害，实现这个效果的人一定是大牛，然后，反编译工具去查看这些酷酷的动态是怎么实现的。我这一看里面的源代码，我勒个擦，洋洋洒洒，代码认得我，我不认识它。但是，通过改改元素，以及一些数值参数，也能实现了类似的效果，并沾沾自喜，以为自己学到了技术，还好，我没这么走下去。

其实现在很多前端小伙伴，跟我当时的做法很类似，去网上找插件，找工具，找现成代码，然后改改用用，也能满足项目需求，并认为自己学习能力很强，解决问题能力很强，实际上并没有什么竞争力，因为搬运工本质上就是一个体力劳动，一旦市场饱和，或者年龄上去搬运不动了，则职业生涯只能止步不前了。

后来为了以后更高的高度，断臂求生，置之死地，告别拿来主义，毕业后长达九个月的时间，都在一直系统学习基础知识，CSS基础，JS基础以及PHP基础，这是一段很多人都不理解的过程，但现在回过头来看看，虽然好像起步比别人晚了，但是至少从目前来看后劲要比大多数人都要足。因为万丈高楼平地起，万变不离其宗，你基础扎实了，任何场景，任何需求的实现其实都是手到擒来，即使你以前根本就没有做过这样的东西。

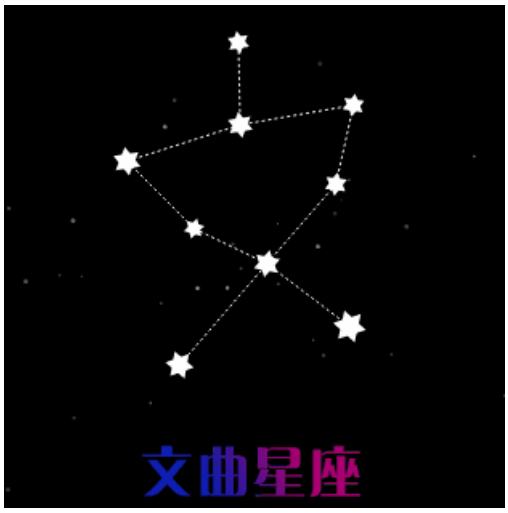
最近接了一个小需求，其中有个场景非常适合我早年见过那个弹来弹去的效果。10年过去了，我脑子里留下的只是那个动效的形，而代码一丁点的记不得。但是当我想实现类似的效果的时候，我脑子一瞬间就知道该如何实现它了，这种体验有一种说不出的感觉，就好像你这么多年一直辛辛苦苦地学那些枯燥的基础知识好像有了意义，有了价值！

我觉得自己上面这个小故事可以对一些小伙伴的学习关注点带来一些启示，不过话又说回来，自己的故事只适合自己，有些人说我就是想混口饭吃，我不想工作学习那么辛苦，我的建议可能就不适合你。

二、canvas动效眼见为实

关于上面提到的小效果，您可以狠狠地点击这里：canvas实现的星星连线联动弹动动效demo

Gif截屏效果如下：



上面这个看上去还有点小酷，看上去比较有实现难度的动画效果，实际上都是由一些非常基础的知识累积而成，而且都是有套路可循的。

三、canvas 2D炫酷动效的实现的基本套路

目前在web领域，基本上看到那些酷酷的2d动效，都是`canvas`实现的，flash已经基本上都被淘汰了，`canvas`效果的实现，无需安装任何插件，IE9及其以上浏览器支持，至少在移动端，大家可以放心大胆使用，甚至webGL 3D效果都可以尝鲜。

无论是雪花飘，星星动还是粒子飞，其`canvas`实现都是一样的套路，这里就讲讲我的理解：

1. 了解canvas画布的工作原理

实际上，`canvas`的工作原理和我们的显示器是一样的，都是不断的刷新绘制，刷新绘制，只不过显示器的刷新是实时的，而`canvas`的刷新手手动触发的，当然如果我们只想在`canvas`上实现静态的效果，是没必要不断刷新的。

举个简单例子，如果我们希望一个圆圈圈从画布左边移到画布右边的效果，我们可以在第1秒的时候让圆圈圈在最左边，然后下一秒的时候，先用黑板擦把我们的圆圈圈擦掉，然后再重新画圆圈圈再往右偏一点的样子，就这样不断擦不断绘，我们肉眼看到的就是一个动画效果了，有点类似于放电影。

一般来讲，清除画布使用下面的代码：

```
var canvas = document.querySelector('canvas');
var context = canvas.getContext('2d');
// 清除画布
context.clearRect(0, 0, canvas.width, canvas.height);
```

2. canvas画布的不断绘制

在以前我们都是使用定时器进行绘制，但是现在建议使用requestAnimationFrame来实现刷新绘制，为了向下兼容，我们一般会做类似下面的处理：

```
// requestAnimationFrame的向下兼容处理
if (!window.requestAnimationFrame) {
    window.requestAnimationFrame = function(fn) {
        setTimeout(fn, 17);
    };
}
```

于是，动效绘制大致路数会变成这样：

```
var canvas = document.querySelector('canvas');
var context = canvas.getContext('2d');

// 画布渲染
var render = function () {
    // 清除画布
    context.clearRect(0, 0, canvas.width, canvas.height);
    // 绘制
    draw();
    // 继续渲染
    requestAnimationFrame(render);
};
render();
```

而上面的`draw()`就是在canvas画布上绘制图形的代码了，但如果仅仅上面代码还不够，如果是同一个位置不断刷新，我们看到的还是静止不动的效果，还差一个运动变量。

4. 需要一个运动坐标变量

假设我们就有一个圆圈圈，同时`ball.x`就表示此圆圈圈的水平坐标位置，则，代码套路可以是这样：

```
var canvas = document.querySelector('canvas');
var context = canvas.getContext('2d');

// 坐标变量
var x = 0;
// 绘制方法
var draw = function () {
    ball.x = x;
};
// 画布渲染
var render = function () {
    // 清除画布
    context.clearRect(0, 0, canvas.width, canvas.height);
    // 位置变化
    x++;
    // 绘制
    draw();
    // 继续渲染
    requestAnimationFrame(render);
};

render();
```

然而，实际项目中，很少，就只有一个圆圈圈，而是一大波的圆圈圈，此时上面的变量设置就会有问题，哪有那么多变量让你设置呢！通常有两种处理方法，一种是变量管理，有一个大变量，变量里面放的都是小变量，类似于`[{},{},{},...]`这种数据结构；还有一种是走实例化对象，变量存储在实例对象上，很显然，后面一种对变量的管理要更加方便和易于理解。

5. 实例对象管理canvas图形颗粒变量

假设圆圈圈实例名称是Ball，则有：

```
var Ball = function () {
    this.x = 0;
```

```
    this.draw = function () {
        // 根据此时x位置重新绘制圆圈圈
        // ...
    };
};
```

于是：

```
var canvas = document.querySelector('canvas');
var context = canvas.getContext('2d');

// 存储实例
var store = {};
// 实例方法
var Ball = function () {
    this.x = 0;
    this.draw = function () {
        // 根据此时x位置重新绘制圆圈圈
        // ...
    };
};
// 假设就一个圆圈圈
store[0] = new Ball();
// 绘制画布上所有的圆圈圈的方法
var draw = function () {
    // 位置变化
    store[0].x++;
    // 根据新位置绘制圆圈圈
    store[0].draw();
};
// 画布渲染
var render = function () {
    // 清除画布
    context.clearRect(0, 0, canvas.width, canvas.height);
    // 绘制画布上所有的圆圈圈
    draw();
    // 继续渲染
    requestAnimationFrame(render);
};

render();
```

6. 随机多个实例对象坐标尺寸透明度等属性

假设现在有10个圆圈圈，如果每个圈圈的起始位置和运动速度都是一样的，是很无趣的，此时，我们就可以借助`Math.random()`构建随机属性，半径啊，位移速度啊，坐标都可以，如下代码：

```
var canvas = document.querySelector('canvas');
var context = canvas.getContext('2d');

// 存储实例
var store = {};
// 实例方法
var Ball = function () {
    // 随机x坐标也就是横坐标，以及变化量moveX，以及半径r
    this.x = Math.random() * canvas.width;
    this.moveX = Math.random();
    this.r = 5 + 5 * Math.random();
```

```
    this.draw = function () {
        // 根据此时x位置重新绘制圆圈圈
        // ...
    };
};
// 假设10个圆圈圈
for (var indexBall = 0; indexBall < 10; indexBall += 1) {
    store[indexBall] = new Ball();
}
// 绘制画布上10个圆圈圈
var draw = function () {
    for (var index in store) {
        // 位置变化
        store[index].x += store[index].moveX;
        if (store[index].x > canvas.width) {
            // 移动到画布外部时候从左侧开始继续位移
            store[index].x = -2 * store[index].r;
        }
        // 根据新位置绘制圆圈圈
        store[index].draw();
    }
};
// 画布渲染
var render = function () {
    // 清除画布
    context.clearRect(0, 0, canvas.width, canvas.height);
    // 绘制画布上所有的圆圈圈
    draw();
    // 继续渲染
    requestAnimationFrame(render);
};
render();
```

7. 补全圆圈圈的canvas绘制

`canvas`画圆直接API手册上就有，例如这里：

```
var Ball = function () {
    // 随机x坐标也就是横坐标，以及变化量moveX，以及半径r
    this.x = Math.random() * canvas.width;
    this.moveX = Math.random();
    this.r = 5 + 5 * Math.random();

    this.draw = function () {
        // 根据此时x位置重新绘制圆圈圈
        context.beginPath();
        context.fillStyle="#369";
        context.arc(this.x, canvas.height / 2, this.r, 0, Math.PI*2);
        context.closePath();
        context.fill();
    };
};
```

于是，就可以看到下面这个实时`canvas`效果，如果你发现没有效果，请移步原文（http://www.zhangxinxu.com/wordpress/?p=6017）或点击这里感受效果：

如果我们背景变成黑色，圆圈圈变成白色，尺寸再小一点，透明度再忽闪忽闪的变化，则就可以实现一个星空效果。这就是一开始星星连线联动动效demo中星空背景效果的实现套路。

四、canvas 2D炫酷动效需要的技术积累

线和星星一起弹动的动效要比星空实现要复杂点，都大体上套路上是一样的，但是，所需要的技术积累要多一些，包括：

1. 了解canvas动效原理

这个上一小节已经探讨过了，就是不断清除画布再绘制，清除再绘制。

2. 一定的JS基本功

例如，了解实例对象的属性如何获取，以及上下文`this`指代什么；如何有效地遍历以及数据存储等等。

例如，本demo 6个星星所有坐标是个三维数组，并且，每个星座数组的个数都不一样，其中就涉及到查漏补缺的数据处理，但是，如果JS基本功扎实，这些都是手到擒来，分分钟就可以搞定的，反之，这效率怕是要大打折扣了。

3. 知道如何运用一些动画算法

关于动画算法，之前有专门文章介绍过，参见“如何使用Tween.js各类原生动画运动缓动算法”一文，本文的弹动效果使用的就是其中的ElasticEaseOut算法。

准备好前后两个场景每个星星的坐标位置，然后，位置变化的值走ElasticEaseOut算法，弹动效果就出来了。

4. 了解canvas所有API，尤其基本的绘图API

如何使用canvas绘制线条，绘制圆，绘制不规则图形；如何描边，如何填充，如何控制透明度等等，都是必须要牢固掌握的。

因为，无论是是圆圈圈，还是角星星，都离不开这些基础的API绘制。

还需要了解图像绘制API，例如，本demo的星星，实际上还可以基于图片资源绘制，难度会稍微降低些，但可能要牺牲点效果。

又例如，demo中的虚线连接，语法很简单，只需要起点坐标和终点坐标就可以，渲染的时候，只要实时根据两个星星中心点（假设坐标分别是(x0,y0)和(x1,y1)）连线就可以了，永远跟着星星，星星弹动，线自然也跟着有弹动效果。

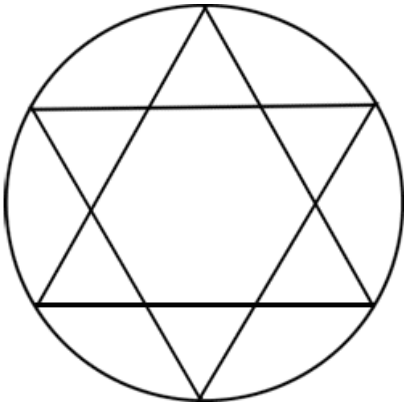
```
// 开始路径绘制
context.beginPath();
// 虚线设定
context.setLineDash([2,2]);
// 设置路径起点
context.moveTo(x0, y0);
// 绘制一条到终点点的直线
context.lineTo(x1, y1);
// 设置线宽
context.lineWidth = 1.0;
// 设置线的颜色
context.strokeStyle = '#fff';
// 进行线的着色，这时整条线才变得可见
context.stroke();
```

5. 掌握常见的曲线函数

也就是要有一定的数学功力，无论是CSS3的`transform`变幻还是我们平时所见的各种动画效果，其本质上都是数学函数运动和矩阵变换，因此，要想再图形可视化以及视觉动效领域有所建树，数学不能差。

通常而言，常见的图形绘制，或者运动轨迹之类，都是需要借助数学函数的，比方所我早些年写过的抛物线相关的文章“JavaScript与元素间的抛物线轨迹运动”，又或者本文的六角星效果。这些图形绘制所需要的数学知识仅仅高中程度就可以，算是比较简单的，拿这里的六角星举例：

六角星实际上是两个等边三角形组成，且六个顶点正好在一个圆上，如下图所示：



所以我们只要根据圆心和圆弧公式就可以求得六个点的坐标位置了，计算公式为：

```
x=a+r*cosθ
y=b+r*sinθ
```

其中，`(a,b)`是圆心坐标，`r`是半径大小，`θ`是弧度大小，在六角星中，弧度间隔都是`1/6`弧度，因此，6个坐标点分别是：

```
var arrPos = [
  [a + r * Math.cos(0), b + r * Math.sin(0)],
  [a + r * Math.cos(Math.PI * 2 / 3), b + r * Math.sin(Math.PI * 2 / 3)],
  [a + r * Math.cos(Math.PI * 2 / -3), b + r * Math.sin(Math.PI * 2 / -3)],
  [a + r * Math.cos(Math.PI / 3), b + r * Math.sin(Math.PI / 3)],
  [a + r * Math.cos(Math.PI / -3), b + r * Math.sin(Math.PI / -3)],
  [a + r * Math.cos(Math.PI), b + r * Math.sin(Math.PI)]
]
```

然后，就可以类似下面3点连线填充绘制了：

```
context.beginPath();
context.moveTo(arrPos[0][0], arrPos[0][1]);
context.lineTo(arrPos[1][0], arrPos[1][1]);
context.lineTo(arrPos[2][0], arrPos[2][1]);
context.closePath();
context.fillStyle = '#fff';
context.fill();

context.beginPath();
context.moveTo(arrPos[3][0], arrPos[3][1]);
context.lineTo(arrPos[4][0], arrPos[4][1]);
context.lineTo(arrPos[5][0], arrPos[5][1]);
```

```
context.closePath();
context.fillStyle = '#fff';
context.fill();
```

然后，只要确定6个场景，星星的中心坐标位置，就能将所有的星座绘制起来了。

然后，有了这些基本功，配合本文介绍的 `canvas` 动效实现的常见套路，就能实现最终想要的效果。

五、接在结尾的结束语

去年关于canvas效果写了篇文章“canvas图形绘制之星空、噪点与烟雾效果”，演示了一些基础的canvas效果的实现，都有未压缩的源代码，帮助学习理解。

好了，就这些，感谢阅读，欢迎交流！

> 《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。

« 【翻译】借助SVG实现背景透明JPG图片　　CSS font-family常见中文字体对应的英文名称 »

猜你喜欢

- 小折腾：JavaScript与元素间的抛物线轨迹运动
- 如何使用Tween.js各类原生动画运动缓动算法
- CSS3动画那么强，requestAnimationFrame还有毛线用？
- canvas图形绘制之星空、噪点与烟雾效果
- 小tips:使用canvas在前端实现图片水印合成
- SVG <foreignObject>简介与截图等应用
- canvas getImageData与任意字符图形点、线动效实现
- 小tips: 在canvas上实现元素图片镜像翻转动画效果
- 纯前端实现可传图可字幕台词定制的GIF表情生成器
- 翻译 - 解释JavaScript的“预解析(置顶解析)”
- canvas实现iPhoneX炫彩壁纸屏保外加pixi.js流体动效

分享到： 0

标签： canvas, drawImage, Elastic, Tween类, 函数, 弹性缓动, 数学, 绘图

发表评论（目前18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. **nicholasurey说道：**
2017年05月10日 10:48

赞~ 圆圈圈，角星星

回复

2. **miku说道：**
2017年05月9日 10:05

大大 我想问的是 复杂度比较高的canvas对网站性能影响大不大？用flash和css3做的动效只要时间长了很明显会感觉到对网站性能的拖累

回复

3. **lc说道：**
2017年05月4日 18:01

看来大神惜字如金啊

回复

4. **星座变化说道：**
2017年04月20日 09:27

张大大，请问你的每个星座中的每个星星的坐标是怎么定义的呢？

回复

> 张 鑫旭说道：
> 2017年04月20日 22:18
>
> 量的~
>
> 回复

5. **canvas动画resquestAnimationFrame()方法求教说道：**
2017年04月6日 16:31

一直关注你的博客，最近在做一个canvas动画的小项目。对于resquestAnimationFrame（）函数运用感到不解，或者说是迷茫。查找资料发现比较少恳请博主有时间、有兴趣更新一些这方面的文章。

回复

6. **喝茶的螃蟹说道：**
2017年04月5日 18:41

张大大，请问怎么判断一个动画已经结束了呢？ 如您文章中的 星座A 变换到星座B 再变成星座C。怎么样才能判定从A到B的动画已经结束了？

回复

> 张 鑫旭说道：
> 2017年04月7日 22:06
>
> 递增的变量值和你设定的最大数值一样的时候。
>
> 回复

7. **billzbc说道：**
2017年03月31日 14:19

谢谢鑫哥深入浅出的讲解！受益非凡

回复

8. 菜鸟说道：
2017年03月26日 23:18

学习了！谢谢你写的教程，很有用！

回复

9. **Tony说道：**
2017年03月23日 13:31

数学不好，荒废了公式。。。。

回复

10. 唐小狼说道：
2017年03月20日 14:11

我可以转载吗?

回复

> 张 鑫旭说道：
> 2017年03月20日 22:25
>
> 请问转到哪里?
>
> 回复

11. **dont说道：**
2017年03月20日 13:51

第三节-第六点里代码 random写错了 —> this.r = 5 + 5 * Math.randow();

第四节-第五点 “再六角星中” = =

回复

> 张 鑫旭说道：
> 2017年03月20日 22:24
>
> 好的，感谢反馈！
>
> 回复

12. 麦子说道：
2017年03月20日 09:15

大神，请问requestAnimFrame的速度如何设置，我想放慢点，可是找不到方法呀

回复

> **jzz7280**说道：
> 2017年03月22日 16:58
>
> 单位时间内，减少执行次数
>
> 回复

13. 谢谢 学习了说道：
2017年03月20日 09:04

谢谢 学习了

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 (190)
» 未来必热：SVG Sprite技术介绍 (119)
» 粉丝群第1期CSS小测点评与答疑 (115)
» HTML5终极备忘大全（图片版+文字版） (93)
» 让所有浏览器支持HTML5 video视频标签 (86)
» Selectivizr-让IE6~8支持CSS3伪类和属性选择器 (82)
» CSS3下的147个颜色名称及对应颜色值 (79)
» 视区相关单位vw, vh..简介以及可实际应用场景 (76)
» 写给自己看的display: flex布局教程 (76)
» 小tips: 纯CSS实现打字动画效果 (76)

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

- 小折腾：JavaScript与元素间的抛物线轨迹运动
- 如何使用Tween.js各类原生动画运动缓动算法
- CSS3动画那么强，requestAnimationFrame还有毛线用？
- canvas图形绘制之星空、噪点与烟雾效果
- 小tips:使用canvas在前端实现图片水印合成
- SVG <foreignObject>简介与截图等应用
- canvas getImageData与任意字符图形点、线动效实现
- 小tips: 在canvas上实现元素图片镜像翻转动画效果
- 纯前端实现可传图可字幕台词定制的GIF表情生成器
- 翻译 - 解释JavaScript的“预解析(置顶解析)”
- canvas实现iPhoneX炫彩壁纸屏保外加pixi.js流体动效

Designed & Powerd by zhangxinxu
Copyright© 2009-2019 张鑫旭-鑫空间-鑫生活
鄂ICP备09015569号