异步图书
www.epubit.com.cn

全面透彻介绍HTML5 Canvas开发核心知识！
涵盖所有Canvas API、大部分动画开发技术以及各种开发技巧！

Web前端开发精品课

HTML5 Canvas开发详解

莫振杰 著

含金量高 前端精品内容荟萃，强化基础提升实战技能。

通俗易懂 语言风格轻松幽默，形象生动讲解枯燥知识。

系统学习 掌握前端高级技巧，清晰流畅学习进阶内容。

贴近读者 结合自身学习经历，文字极具温度不失严谨。

直击痛点 规避开发思维误区，精炼浓缩直指技术本质。

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

Web前端开发精品课

HTML5 Canvas开发详解

莫振杰 著

人民邮电出版社
北京

图书在版编目（CIP）数据

Web前端开发精品课 : HTML5 Canvas开发详解 / 莫振杰著. -- 北京 : 人民邮电出版社, 2017.5
ISBN 978-7-115-45020-3

Ⅰ. ①W… Ⅱ. ①莫… Ⅲ. ①超文本标记语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2017)第052963号

内 容 提 要

本书结合笔者在前后端大量开发工作中的实战经验，系统化知识，浓缩精华，用通俗易懂的语言直击学习者的痛点。学习本书，可以让你掌握所有 Canvas API、大部分动画技术以及各种高级开发技巧，真正获得一个稀有技能！

全书共分为两大部分，第一部分是 Canvas 基础内容，主要介绍 Canvas API 语法，其中包括图形绘制、线条操作、文本操作、图片操作、变形操作、像素操作等各种基础 API 语法；第二部分是 Canvas 进阶内容，主要介绍 Canvas 动画开发，包括事件操作、物理动画、边界检测、碰撞检测、高级动画等各种稀有技巧。

除了知识的讲解，教程还融入了大量的开发案例，并且更加注重实战编程思维的培养，为学习者提供一个流畅的学习思路。

◆ 著　　莫振杰
责任编辑　赵　轩
责任印制　焦志炜

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京画中画印刷有限公司印刷

◆ 开本：720×960　1/16
印张：21.75
字数：382 千字　　2017 年 5 月第 1 版
印数：1－3 000 册　　2017 年 5 月北京第 1 次印刷

定价：79.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广字第 8052 号

目 录

CONTENTS

第一部分 Canvas基础

第1章 Canvas概述

第2章 直线图形

第3章 曲线图形

第4章 线条操作

第5章 文本操作

第6章 图片操作

第一部分 Canvas 基础

第 1 章 Canvas概述

1.1 Canvas简介

1.1.1 Canvas是什么

在 HTML5 之前，为了达到页面绚丽多彩的效果，我们很多情况下都是借助“图片”来实现。不过使用图片这种方式，都是以“低性能”为代价的。由于图片体积大、下载速度慢等原因，因此为了应对日渐复杂的 Web 应用开发，W3C 在 HTML5 标准中引入了 Canvas 这一门技术。

我们都知道，HTML5 新增了一个 Canvas 元素。其实，Canvas 又称为“画布”，是 HTML5 的核心技术之一。我们常说的 Canvas 技术，指的就是使用 Canvas 元素结合 Javascript 来绘制各种图形的技术。

既然 Canvas 是 HTML5 核心技术，那它都有哪些厉害之处呢?

1. 绘制图形

Canvas 可以用来绘制各种基本图形如矩形、曲线、圆等，也可以绘制各种复杂绚丽的图形，如图 1-1 所示。

2. 绘制图表

很多公司业务的数据展示都离不开图表，使用 Canvas 可以用来绘制满足各种需求的

图表，如图 1-2 所示。

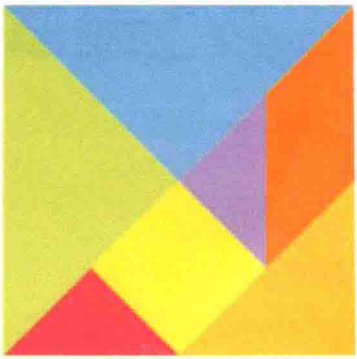

图1-1　Canvas绘制图形（七巧板）

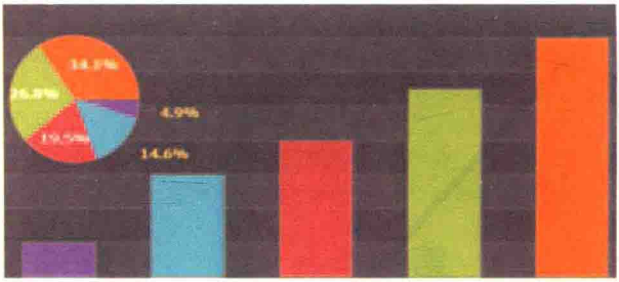

图1-2　Canvas绘制图表

3. 动画效果

使用 Canvas，我们也可以制作出各种华丽的动画效果，这也是 Canvas 为大家带来的一大乐趣，如图 1-3 所示。

4. 游戏开发

游戏开发在 HTML5 领域具有举足轻重的地位，现在我们也可以使用 Canvas 来开发各种游戏，如图 1-4 所示。这几年非常火的游戏如围住神经猫等，就是使用 HTML5 Canvas 来开发的。

图1-3　Canvas动画效果

图1-4　Canvas开发游戏

此外，Canvas 技术是一门纯 JavaScript 操作的技术，因此大家需要具备 JavaScript 入门知识。对于 JavaScript 的学习，可以关注绿叶学习网（www.lvyestudy.com）中的开源教程。

1.1.2　Canvas与SVG

HTML5 有两个主要的 2D 图形技术：Canvas 和 SVG。事实上，Canvas 和 SVG 是两门完全不同的技术。两者具有以下区别。

（1）Canvas 是使用 JavaScript 动态生成的，SVG 是使用 XML 静态描述的。

（2）Canvas 是基于“位图”的，适用于像素处理和动态渲染，图形放大会影响质量。SVG 是基于“矢量”的，不适用于像素处理和静态描述，图形放大不会影响质量。也就是说，使用 Canvas 绘制出来的是一个“位图”，而使用 SVG 绘制出来的是一个“矢量图”。如图 1-5 和图 1-6 所示。

（3）每次发生修改，Canvas 需要重绘，而 SVG 不需要重绘。

（4）Canvas 与 SVG 的关系，简单来说，就像“美术与几何”的关系一样。

图1-5 Canvas位图（放大会失真）

图1-6 SVG矢量图（放大不会失真）

此外，并非 Canvas 就比 SVG 更有前途，也并非 SVG 就比 Canvas 更有前途，因为这两个是用于不同场合的。在实际开发中，我们应该根据开发需求去选择。

当然，这里只是简单介绍了一下 Canvas 与 SVG 的区别，如果想真正了解，我们还需要深入学习这两门技术。最后给大家一个小小的建议：很多人接触新技术的时候，喜欢在第一遍学习中就把每一个细节都弄清楚，事实上这是效率最低的学习方法。在第一遍学习中，如果有些东西实在没办法理解，那就直接跳过，等到学到后面或者看第二遍的时候，自然而然就懂了。

1.2 Canvas元素知识

HTML5 Canvas，简单来说，就是一门使用 JavaScript 来操作 Canvas 元素的技术。使用 Canvas 元素来绘制图形，需要以下三步。

（1）获取 canvas 对象。

（2）获取上下文环境对象 context。

（3）开始绘制图形。

举例：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <meta charset="utf-8" />
```

```
        <script type="text/javascript">
            window.onload = function () {
                //1、获取 canvas 对象
                var cnv = document.getElementById("canvas");
                //2、获取上下文环境对象 context
                var cxt = cnv.getContext("2d");
                //3、开始绘制图形
                cxt.moveTo(50, 100);
                cxt.lineTo(150, 50);
                cxt.stroke();
            }
        </script>
    </head>
    <body>
        <canvas id="canvas" width="200" height="150" style="border:1px dashed
gray;"></canvas>
    </body>
    </html>
```

在浏览器中的预览效果如图 1-7 所示。

图1-7　Canvas画直线

分析：

在 Canvas 中，我们首先使用 document.getElementById() 方法来获取 canvas 对象（这是一个 DOM 对象），然后使用 canvas 对象的 getContext("2d") 方法获取上下文环境对象 context，最后再使用 context 对象的属性和方法来绘制各种图形。

1.2.1　Canvas元素

Canvas 是一个行内块元素（即 inline-block），我们一般需要指定其三个属性：id、width 和 height。width 和 height 分别定义 Canvas 的宽度和高度。默认情况下，Canvas 的宽度为 300px，高度为 150px。

对于 Canvas 的宽度和高度，有两种方法来定义：①在 HTML 属性中定义；②在 CSS 样

式中定义。但是在实际开发中，我们一定不要在CSS样式中定义，而是应该在HTML属性中定义。为什么呢？下面先来看一个例子。

举例：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <meta charset="utf-8" />
    <style type="text/css">
        canvas
        {
            width:200px;
            height:150px;
        }
    </style>
    <script type="text/javascript">
        window.onload = function () {
            var cnv = document.getElementById("canvas");
            var str = "canvas的宽度为:" + cnv.width + ", 高度为:" + cnv.height;
            alert(str);
        }
    </script>
</head>
<body>
    <canvas id="canvas" style="border:1px dashed gray;"></canvas>
</body>
</html>
```

在浏览器中的预览效果如图1-8所示。

分析：

从这个例子可以看出：如果在CSS样式中定义，我们使用canvas对象获取的宽度和高度是默认值，而不是实际的宽度和高度。这样就无法获取canvas对象正确的宽度和高度。获取canvas对象实际的宽度和高度是Canvas开发中最常用的操作，因此对于Canvas的宽度和高度我们就一定要在HTML属性中定义，而不是在CSS属性中定义。

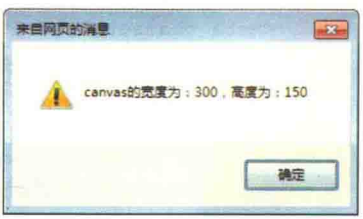

图1-8 Canvas无法获取正确的宽度和高度

1.2.2 Canvas对象

在Canvas中，我们可以使用document.getElementById()方法来获取canvas对象。

canvas 对象常用的属性和方法如下：

表 1-1　canvas 对象属性

属性	说明
width	Canvas 的宽度
height	Canvas 的高度

表 1-2　canvas 对象方法

属性	说明
getContext(“2d”)	获取 Canvas 2D 上下文环境对象
toDataURL()	获取 canvas 对象产生的位图的字符串

也就是说，我们可以使用 cnv.width 和 cnv.height 分别获取 Canvas 的宽度和高度，可以使用 cnv.getContext("2d") 来获取 canvas 2D 上下文环境对象，也可以使用 toDataURL() 来获取 canvas 对象产生的位图的字符串。在这里，cnv 是指 canvas 对象。

对于 toDataURL() 方法，我们可以暂时不去深入，在后面章节中会详细给大家介绍。这里我们只要认真学习一下 getContext("2d") 方法就可以了。在 Canvas 中，我们使用 getContext("2d") 来获取 Canvas 2D 上下文环境对象，这个对象又称为 context 对象。后面章节接触的所有图形的绘制，使用的都是 context 对象的属性和方法，这一点需要特别清楚。当然现在不理解没关系，学到后面再回过头来看看这段话就懂了。

举例：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <meta charset="utf-8" />
    <script type="text/javascript">
        window.onload = function () {
            var cnv = document.getElementById("canvas");
            var str = "Canvas 的宽度为: " + cnv.width + ", 高度为: " + cnv.height;
            alert(str);
        }
    </script>
</head>
<body>
    <canvas id="canvas" width="200" height="160" style="border:1px dashed
gray"></canvas>
</body>
</html>
```

在浏览器中的预览效果如图 1-9 所示。

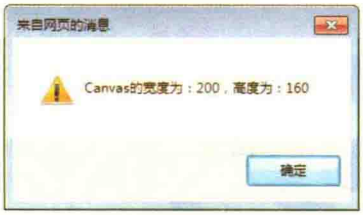

图1-9 Canvas获取正确的宽度和高度

本节要特别注意一点：**以后学习的所有图形的绘制，我们使用的都是 context 对象（上下文环境对象）的属性和方法。**

【疑问】

（1）我们可以使用 getContext("2d") 来实现 2D 绘图，那是不是意味着可以使用 getContext("3d") 来实现 3D 绘图呢？

HTML5 Canvas 暂时只提供 2D 绘图 API，3D 绘图可以使用 HTML5 中的 WebGL 进行开发。不过 3D 绘图一直以来都是 HTML5 中的“黑科技”，技术要求高并且难度大。等学完了这本书，有兴趣的小伙伴可以关注一下绿叶学习网的 WebGl 教程

（2）对于 IE 浏览器来说，暂时只有 IE9 以上版本支持 HTML5 Canvas，那么如何处理 IE7 和 IE8 的兼容性问题呢？

对于 IE7 和 IE8，可以借助 explorercanvas 这个扩展来解决。该扩展下载地址为：https://github.com/arv/explorercanvas。

我们只需要在页面像引入外部 JavaScript 文件那样引入 excanvas.js 就可以了，代码如下：

```
<!--[if IE]>
    <script src="excanvas.js"></script>
<![end if]-->
```

不过要跟大家说明一下，低版本 IE 浏览器即使引入了 excanvas.js 来使用 Canvas，在功能上也会有很多限制，例如无法使用 fillText() 方法等。

（3）对于 Canvas 的学习，除了这本书，还有什么推荐的吗？

建议大家多看看 W3C 官方网站中的 canvas 的开发文档，因为这是最权威的参考资料。W3C 官网地址：www.w3.org/TR/2dcontext/。

第2章 直线图形

2.1 直线图形简介

在 Canvas 中，基本图形有两种：①直线图形；②曲线图形。Canvas 常见的直线图形有三种，分另是直线、矩形、多边形。

这一章我们先来学习 Canvas 中的直线图形。

2.2 直线

2.2.1 Canvas坐标系

在学习 Canvas 之前，我们先来介绍一下 Canvas 中的坐标系是如何使用的。了解 Canvas 使用的坐标系是学习 Canvas 的最基本的前提。

我们经常见到的坐标系是数学坐标系，而 Canvas 使用的坐标系是 W3C 坐标系，这两种坐标系唯一的区别在于 y 轴正方向的不同。

（1）数学坐标系：y 轴正方向向上。

（2）W3C 坐标系：y 轴正方向向下。

注意：W3C 坐标系的 y 轴正方向是向下的。很多小伙伴学到后面对 Canvas 一些代码感到很困惑，那是因为他们没有清楚地意识到这一点。

数学坐标系一般用于数学形式上的应用，而在前端开发中几乎所有涉及坐标系的技术使用的都是 W3C 坐标系，这些技术包括 CSS3、Canvas、SVG 等。了解这一点，我们以后在学习 CSS3 或者 SVG 的时候，很多知识就可以串起来了。数字坐标系和 w3c 坐标系如图 2-1 所示。

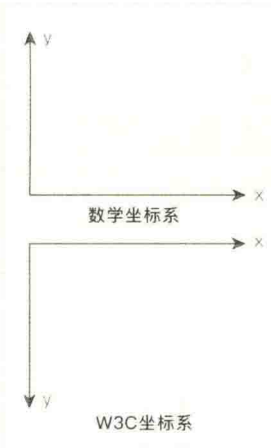

图2-1 数学坐标系和W3C坐标系

2.2.2 直线的绘制

在 Canvas 中，可以使用 moveTo() 和 lineTo() 这两个方法配合使用来画直线。利用这两个方法，我们可以画一条直线，也可以同时画多条直线。

1. 一条直线

语法：

```
cxt.moveTo(x1, y1);
cxt.lineTo(x2, y2);
cxt.stroke();
```

说明：

cxt 表示上下文环境对象 context。

(x1,y1) 表示直线“起点”的坐标。moveTo 的含义是“将画笔移到该点 (x1,y1) 位置上，然后开始绘图”。

(x2,y2) 表示直线“终点”的坐标。lineTo 的含义是“将画笔从起点 (x1,y1) 开始画直线，一直画到终点坐标 (x2,y2)”。

对于 moveTo() 和 lineTo() 这两个方法，从英文意思角度更容易帮助我们理解和记忆。

```
cxt.moveTo(x1, y1);
cxt.lineTo(x2, y2);
```

上面两句代码仅仅是确定直线的“起点坐标”和“终点坐标”这两个状态，但是实际上画笔还没开始“动”。因此我们还需要调用上下文对象的 stroke() 方法才有效。

使用 Canvas 画直线，与我们平常用笔在纸张上画直线是一样的道理，都是先确定直线起点 (x1,y1) 与终点 (x2,y2)，然后再用笔连线 (stroke())。

举例：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <meta charset="utf-8" />
    <script type="text/javascript">
        function $$(id){
            return document.getElementById(id);
        }
        window.onload = function () {
            var cnv = $$("canvas");
            var cxt = cnv.getContext("2d");

            cxt.moveTo(50, 100);
            cxt.lineTo(150, 50);
            cxt.stroke();
        }
    </script>
</head>
<body>
    <canvas id="canvas" width="200" height="150" style="border:1px dashed
gray;"></canvas>
</body>
</html>
```

在浏览器中的预览效果如图 2-2 所示。

图2-2 使用Canvas画一条直线

分析：

在这个例子中，我们定义了一个获取 DOM 对象元素的函数 $$(id)，这样减少了重复代码量，使得思路更加清晰。记住，Canvas 中使用的坐标系是“**W3C 坐标系**”。其中这个例子的分析图如图 2-3 所示。

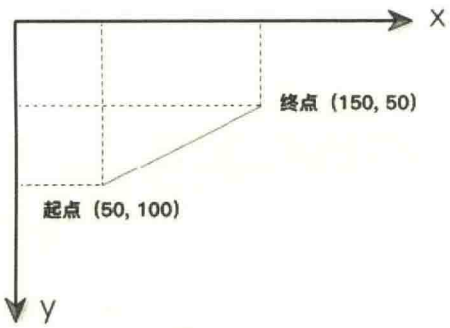

图2-3 分析图

2. 多条直线

从上面可以知道，使用 moveTo() 和 lineTo() 这两个方法可以画一条直线。其实如果我们想要同时画多条直线，也是使用这两种方法。

语法：

```
cxt.moveTo(x1, y1);
cxt.lineTo(x2, y2);
cxt.lineTo(x3, y3);
.....
cxt.stroke();
```

说明：

lineTo() 方法是可以重复使用的。第一次使用 lineTo() 后，画笔将自动移到终点坐标位置，第二次使用 lineTo() 后，Canvas 会以“上一个终点坐标”作为第二次调用的起点坐标，然后再开始画直线，以此类推。下面先来看个例子，这样更容易理解些。

举例：画两条直线

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <meta charset="utf-8" />
    <script type="text/javascript">
        function $$(id){
            return document.getElementById(id);
        }
```