canvas文本绘制自动换行、字间距、竖排等实现

这篇文章发布于 2018年02月5日，星期一，02:57，归类于 Canvas相关。 阅读 20495 次, 今日 25 次 16 条评论

一、canvas对文字排版的支持很弱

和CSS相比，SVG以及canvas对文字排版的支持很弱。

在CSS中天然支持的文本自动换行，其他 `letter-sapcing` 字间距， `writing-mode` 竖排等都是一个CSS属性就可以实现。但是在canvas中，全部都不支持。

canvas绘制文本API为：

```
CanvasRenderingContext2D.fillText(text, x, y [, maxWidth]);
```

text

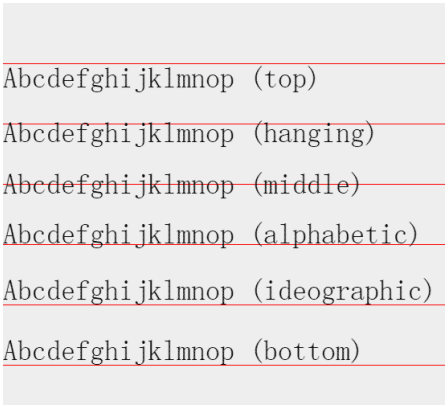
`text` 是需要绘制的文本。

x

`x` 是文本绘制的水平参考点坐标。随着 `CanvasRenderingContext2D.textAlign` 的设置不同， `x` 的坐标位置也不同。可以表示这段文字内容左侧坐标，或水平中心坐标，或右侧坐标。

y

`y` 是文本绘制的垂直参考点坐标。随着 `CanvasRenderingContext2D.textBaseline` 的设置不同， `y` 的坐标位置也不同。支持多种基线类型（CSS中也有对应概念），MDN上有一张图可以很好地表示文本基线和文本垂直位置的关系。



maxWidth

`maxWidth` 表示文本内容占据的最大宽度。这里的 `maxWidth` 概念和CSS中的 `max-width` 差别很大，其最终的文本表现是：当文本占据宽度超过 `maxWidth` 的后，所有的文本自动变窄以适应这个最大宽度限制。表现类似这样：

您可以狠狠地点击这里：maxWidth参数让文字变窄demo

相关测试代码如下：

```
var canvas = document.querySelector('canvas');
var context = canvas.getContext('2d');
context.font = '32px sans-serif';
context.fillText('我是一段被maxWidth限制的文本', 0, 50, 200
```

如果没有`maxWidth`限制，则文本会一行走到底，直到超出画布尺寸，有点类似CSS中设置容器`white-space:nowrap` + `overflow:hidden`的表现。

二、如何让canvas支持自动换行?

如何让canvas支持自动换行?

首先有一点可以肯定，就是到目前为止，canvas中并没有任何可以让文本自动换行的现成的API。

因此注定这个看上去简单的事情实践起来并没有那么容易。

通常比较好的实现方法有下面两种：

1. canvas计算与逐行绘制

实现原理的核心是`CanvasRenderingContext2D.measureText(text)`这个API，可以返回一个TextMetrics对象，其中包含了当前上下文环境下`text` `double`精度的占据宽度，于是我们就可以通过每个字符宽度的不断累加，精确计算哪个位置应该可以换行。

下面就是我扩展的文本自动换行方法JS代码：

```
CanvasRenderingContext2D.prototype.wrapText = function (text, x, y, maxWidth, lineHeight)
{
    if (typeof text != 'string' || typeof x != 'number' || typeof y != 'number') {
        return;
    }

    var context = this;
    var canvas = context.canvas;

    if (typeof maxWidth == 'undefined') {
        maxWidth = (canvas && canvas.width) || 300;
    }
    if (typeof lineHeight == 'undefined') {
        lineHeight = (canvas && parseInt(window.getComputedStyle(canvas).lineHeight)) || p
arseInt(window.getComputedStyle(document.body).lineHeight);
    }

    // 字符分隔为数组
    var arrText = text.split('');
    var line = '';

    for (var n = 0; n < arrText.length; n++) {
        var testLine = line + arrText[n];
        var metrics = context.measureText(testLine);
        var testWidth = metrics.width;
        if (testWidth > maxWidth && n > 0) {
            context.fillText(line, x, y);
            line = arrText[n];
```

```
            y += lineHeight;
        } else {
            line = testLine;
        }
    }
    context.fillText(line, x, y);
};
```

API如下：

```
CanvasRenderingContext2D.wrapText(text, x, y, maxWidth, lineHeight)
```

其中 `text`，`x`，`y` 3个参数和 `fillText()` 方法中的这3个参数含义是一样的，不赘述。
而 `maxWidth` 表示的含义可就不一样了，表示最大需要换行的宽度，此参数可缺省，默认会使用canvas画布的 `width` 宽度作为 `maxWidth`； `lineHeight` 表示行高，同样可缺省，默认会使用 `<canvas>` 元素在DOM中继承的 `line-height` 作为行高。

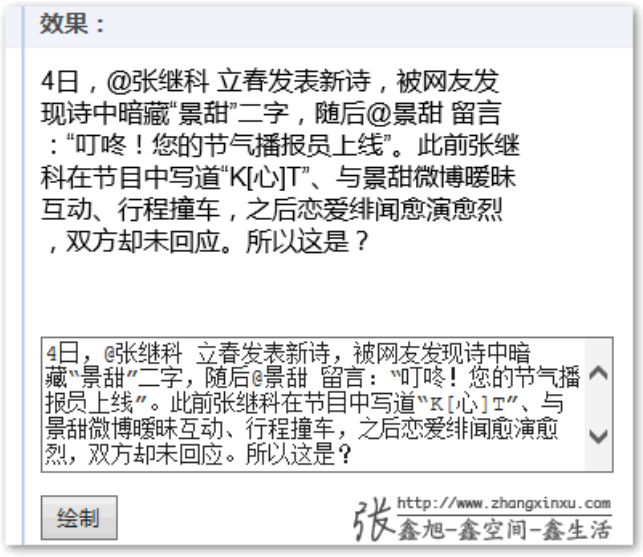
使用示例如下：

```
var canvas = document.querySelector('canvas');
var context = canvas.getContext('2d');
context.font = '16px sans-serif';
context.textBaseline = 'top';
context.wrapText('我是一段会换行的文字啦啦啦', 0, 0);
```

用法很简单，使用 `wrapText` 代替原生的 `fillText` 即可！

您可以狠狠的点击这里：自动换行扩展API wrapText演示demo

下面截图就是demo页面绘制效果（截自IE9浏览器）：



可以看到上方绘制的文字在核实位置自动换行了，您可以修改 `<textarea>` 中的文字内容，点击“绘制”按钮体验下其他文本内容的自动换行绘制效果。

2. 借助SVG <foreignObject>直接把CSS效果绘制上去

关于SVG `<foreignObject>` 让HTML转换成canvas图片的原理和细节可以参见我之前写的“SVG <foreignObject<简介与截图等应用”这篇文章。

基本上，本文后面会介绍到的字符间距，文字竖排等实现都可以使用这个方法实现，因此，为了避免不必要的啰嗦，仅本效果会具体演示代码细节，后面效果大家自行拷贝改改就好了。

我们先看实例，您可以狠狠地点击这里：canvas借助SVG foreignObject实现文本自动换行demo

结果Chrome浏览器下：

JS实现如下：

```
var canvas = document.querySelector('canvas');
var context = canvas.getContext('2d');
context.font = '16px sans-serif';
var width = canvas.width;
var height = canvas.height;

var tempImg = new Image();
tempImg.width = width;
tempImg.height = height;
tempImg.onload = function () {
    // 把img绘制在canvas画布上
    context.drawImage(this, 0, 0, width, height);
};
tempImg.src = 'data:image/svg+xml;charset=utf-8,<svg xmlns="http://www.w3.org/2000/svg"><f
oreignObject width="'+ width +'" height="'+ height +'"><body xmlns="http://www.w3.org/1999
/xhtml" style="margin:0;font:'+ context.font +';">我是一段需要换行的文字啦啦啦</body></foreignO
bject></svg>';
```

此方法优点在于足够简单，只要一段带 `style` 样式的HTML代码即可！

唯一不足在于兼容性，IE浏览器不支持 `<foreignObject>`，最新的Firefox浏览器虽然支持 `<foreignO bject>`，但是只能以 `<img<` 形式呈现，无法绘制到canvas画布上（若谁知道原因欢迎不吝赐教）。

不过好的是移动端Safari浏览器以及微信浏览器都是支持的，因此，此方法理论上是可以在移动端使用的。例如我手机Safari的效果截图：

二、如何让canvas支持字符间距?

1. 如果只需要兼容Chrome，直接letter-spacing控制

对于Chrome浏览器，无论是字符间距还是单词间距，都可以自动继承于 `<canvas>` 元素，这个特性让人非常感动。

也就是：

```
canvas { letter-pacing: 5px; }
```

绘制的文字字符间距自动就是 `5px`。

如此欣喜的特性有必要亲眼见证一下，您可以狠狠地点击这里：canavs文本间距使用CSS letter-spacing实现demo

效果如下GIF示意：

完整测试JS代码如下：

```
var canvas = document.querySelector('canvas');
var context = canvas.getContext('2d');
var range = document.querySelector('input[type=range]');
// 绘制方法
var draw = function () {
    // 清除之前的绘制
    context.clearRect(0, 0, canvas.width, canvas.height);
    // 字符间距设置
    canvas.style.letterSpacing = range.value + 'px';
    // 并绘制文本，font属性值设置一定要在这里
    context.font = '32px sans-serif';
    context.fillText('我是一段文本', 0, 50);
};
// 改变字符间距后重绘
range.addEventListener('change', draw);
// 一进来根据默认值绘制
draw();
```

根据我的观察，貌似Chrome浏览器在设置`font`属性值的时候，把`letter-spacing`等信息一起算作上下文中了。所以，虽然看上去`context.font = '32px sans-serif'`一直都没变，但却不能放在`draw()`方法之外，否则，还是按照老的`letter-spacing`渲染而看不到字符间距变化。

此方法最简单最容易理解，只可惜，根据我的测试，目前仅Chrome浏览器支持。Firefox以及Safari全都不行。

2. canvas计算与逐字绘制

原理为，每一个字符单独作为一个绘制单元，然后根据字符宽度+letterSpacing间距动态绘制，同样，离不开使用`CanvasRenderingContext2D.measureText(text)`这个API。

以下就是自己直接在原型上扩展的字符间距绘制方法letterSpacingText，大家可以直接拷贝过去使用，MIT协议，保留原出处即可。

```
/**
* @author zhangxinxu(.com)
* @licence MIT
* @description http://www.zhangxinxu.com/wordpress/?p=7362
*/
CanvasRenderingContext2D.prototype.letterSpacingText = function (text, x, y, letterSpacing
) {
    var context = this;
    var canvas = context.canvas;

    if (!letterSpacing && canvas) {
        letterSpacing = parseFloat(window.getComputedStyle(canvas).letterSpacing);
    }
    if (!letterSpacing) {
        return this.fillText(text, x, y);
    }

    var arrText = text.split('');
    var align = context.textAlign || 'left';
```

```
    // 这里仅考虑水平排列
    var originWidth = context.measureText(text).width;
    // 应用letterSpacing占据宽度
    var actualWidth = originWidth + letterSpacing * (arrText.length - 1);
    // 根据水平对齐方式确定第一个字符的坐标
    if (align == 'center') {
        x = x - actualWidth / 2;
    } else if (align == 'right') {
        x = x - actualWidth;
    }

    // 临时修改为文本左对齐
    context.textAlign = 'left';
    // 开始逐字绘制
    arrText.forEach(function (letter) {
        var letterWidth = context.measureText(letter).width;
        context.fillText(letter, x, y);
        // 确定下一个字符的横坐标
        x = x + letterWidth + letterSpacing;
    });
    // 对齐方式还原
    context.textAlign = align;
};
```

API详细：

```
CanvasRenderingContext2D.letterSpacingText(text, x, y, letterSpacing);
```

其中 `text`，`x`，`y` 3个参数和 `fillText()` 方法中的这3个参数含义是一样的，不赘述。 `letterSpacing` 表示字符间距大小，数值。可缺省，默认会拿 `<canvas>` 元素在DOM环境下的 `letter-spacing` 大小作为计算值。

使用示意：

```
var canvas = document.querySelector('canvas');
var context = canvas.getContext('2d');
context.font = '32px sans-serif';
context.textAlign = 'center';
// 字符间隙5px
context.letterSpacingText('我是一段文本', canvas.width / 2, 50, 5);
```

您可以狠狠地点击这里：canavs字符间距JS逐字计算demo

效果如下GIF示意（居中对齐，截自IE Edge）：

此方法兼容性非常好，IE9+浏览器都支持，PC和Mobile通吃。

3. 借助SVG <foreignObject>直接把CSS效果绘制上去

和自动换行实现类似，详细略。

四、如何让canvas支持竖直排列？

文字竖直排列，对于玩英文的老外，可以使用 `context.rotate()` 旋转 `90deg` 实现，但是对于中文等中亚文字，却是完全不适合的。因为两种语言的竖直排版规则是不一样的。

中文等东亚文字上，例如一些古诗词文字还是正的，仅仅是阅读方向是从上往下，但是，英文（以及阿拉伯数字）由于本身的字符特性，直接就是旋转排列的。

所以单纯旋转策略对于大中国并不实用。

在CSS中，我们可以使用 `writing-mode` 改变文档流的方向，从而实现文字竖排，相关文章可以参见我之前的文章：“改变CSS世界纵横规则的writing-mode属性”，或者购买我的《CSS世界》，其中有详细介绍。

那在canvas中又该如何实现呢？

1. JS混合计算逐字排列

混合计算规则如下：全角字符竖排，英文数字等半角字符旋转排列。

下面是我扩展的竖排方法，同样MIT协议，可随意使用，保留上面一段作者和出处说明即可。

```
/**
* @author zhangxinxu(.com)
* @licence MIT
* @description http://www.zhangxinxu.com/wordpress/?p=7362
*/
CanvasRenderingContext2D.prototype.fillTextVertical = function (text, x, y) {
    var context = this;
    var canvas = context.canvas;

    var arrText = text.split('');
    var arrWidth = arrText.map(function (letter) {
        return context.measureText(letter).width;
    });

    var align = context.textAlign;
    var baseline = context.textBaseline;

    if (align == 'left') {
        x = x + Math.max.apply(null, arrWidth) / 2;
    } else if (align == 'right') {
        x = x - Math.max.apply(null, arrWidth) / 2;
    }
    if (baseline == 'bottom' || baseline == 'alphabetic' || baseline == 'ideographic') {
        y = y - arrWidth[0] / 2;
    } else if (baseline == 'top' || baseline == 'hanging') {
        y = y + arrWidth[0] / 2;
    }

    context.textAlign = 'center';
    context.textBaseline = 'middle';

    // 开始逐字绘制
    arrText.forEach(function (letter, index) {
        // 确定下一个字符的纵坐标位置
        var letterWidth = arrWidth[index];
        // 是否需要旋转判断
        var code = letter.charCodeAt(0);
        if (code <= 256) {
            context.translate(x, y);
```

```
            // 英文字符，旋转90°
            context.rotate(90 * Math.PI / 180);
            context.translate(-x, -y);
        } else if (index > 0 && text.charCodeAt(index - 1) < 256) {
            // y修正
            y = y + arrWidth[index - 1] / 2;
        }
        context.fillText(letter, x, y);
        // 旋转坐标系还原成初始态
        context.setTransform(1, 0, 0, 1, 0, 0);
        // 确定下一个字符的纵坐标位置
        var letterWidth = arrWidth[index];
        y = y + letterWidth;
    });
    // 水平垂直对齐方式还原
    context.textAlign = align;
    context.textBaseline = baseline;
};
```

API名称是`fillTextVertical`，语法如下：

```
CanvasRenderingContext2D.fillTextVertical(text, x, y)
```

其中`text`，`x`，`y` 3个参数和`fillText()`方法中的这3个参数含义是一样的，不赘述。

实现的效果是：英文数字等旋转，中文垂直排列。支持`textAlign`和`textBaseline`等基本设置。

您可以狠狠地点击这里：canavs文本竖排JS逐字计算实现demo

使用JS如下：

```
var canvas = document.querySelector('canvas');
var context = canvas.getContext('2d');
context.font = '24px STKaiti, sans-serif';
context.textAlign = 'center';
context.textBaseline = 'top';
context.fillTextVertical('anglebaby和黄晓明', canvas.width / 2, 0);
```

结果如下图所示：

2. 借助SVG <foreignObject>直接把CSS效果绘制上去

和自动换行实现类似，详细略。

五、结束语

当年CSS之所以一统天下就是在文本展现文字排版这一块非常方便。

看看SVG的文本展现，在看看canvas的文本呈现，难用的很。全靠友军衬托啊！

问题来了，CSS文本呈现这里厉害，那还需要canvas干什么？

因为canvas可以方便把文字转换成图片，例如一些广告工具等等，需要前端合成的，就需要canvas大放异彩了。

本文扩展的这些方法并未实际项目大规模验证，有疏漏之处在所难免，欢迎指正！

感谢阅读！

> 《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: 滚动容器尺寸变化子元素视觉上位置不变JS实现

CORS ajax跨域请求php简单完整案例一则 »

猜你喜欢

- SVG <foreignObject>简介与截图等应用
- 图片动态局部毛玻璃模糊效果的实现
- CSS direction属性简介与实际应用
- 改变CSS世界纵横规则的writing-mode属性
- CSS margin-inline和margin-block区别是什么？
- canvas getImageData与任意字符图形点、线动效实现
- 小tips: 使用JS检测用户是否安装某font-family字体
- 纯前端实现可传图可字幕台词定制的GIF表情生成器
- CSS, SVG和canvas分别实现文本文字纹理叠加效果
- 图片旋转效果的一些研究、jQuery插件及实例
- 小tip: SVG和Canvas分别实现图片圆角效果

标签： canvas, fillText, foreignObject, letter-spacing, line-height, measureText, rotate, textAlign, textBaseline, writing-mode

发表评论（目前16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. 厉害说道：
2018年09月3日 13:21

屌爆了,贴主,爱死你了,分享东西真棒,我要以身相许

回复

2. **jeffwcx**说道：
2018年07月31日 08:21

// 水平垂直对齐方式还原
context.textAlign = align;
context.textBaseline = baseline;

类似这样的还原操作，可以采用
context.save();
context.restore();

回复

> 张 鑫旭说道：
> 2018年07月31日 21:21
>
> 赞！
>
> 回复

3. **zlw**说道：
2018年07月26日 22:50

foreignObject

如果使用了这个，canvas很难再转化成图片了。被污染的canvas无法输出数据。被浏览器限制了。

回复

4. **lemotw**说道：
2018年07月25日 21:06

你好:
context.setTransform(1, 0, 0, 1, 0, 0)
放在 fillText() 之前先行初始會表較好些，在某些情況下會使得第一個字跑版。

然後感謝大大提供這解法

回复

5. 城管大队长说道：
2018年06月10日 21:49

“最新的Firefox浏览器虽然支持，但是只能以形式呈现，无法绘制到canvas画布上”

关于这个可以尝试一下，svg转为dataURI方式，作为imag中src的属性值。

回复

6. **perfect**说道：
2018年06月1日 09:48

请问，在IE9及以上浏览器，canvas画图，宽度超过一定值，部分内容不显示，怎么处理？谷歌浏览器正常。使用excanvas在IE8浏览器也正常。

回复

张 鑫旭说道：
2018年06月1日 13:46

没有遇到过，CSS控制下canvas的宽度？

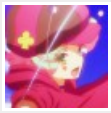
回复

7. 野变说道：
2018年03月13日 15:35

绘制结果看起来不清楚~

回复

8. **Handle**说道：
2018年03月5日 16:50



最近看到一个中文文字排版引擎，不知道靠谱不……感觉的话……还算……简单？？？
https://github.com/Icemic/huozi.js

回复

9. **mfk**说道：
2018年02月9日 08:32

那个wrapText方法中，计算换行宽度的，你的算法是一次加一个字符，然后反复尝试。
用二分法（折半法）尝试，可能效率更好。

回复

戈壁说道：
2018年03月6日 17:05

+1

回复

10. **hovis**说道：
2018年02月6日 14:42

谢谢老师分享
http://hovis.cn

回复

11. 夏目贵志说道：
2018年02月6日 14:07

大大我兴趣爱好 想学习 html+css

你写的新作 是否适合零基础的人 去阅读学习呢？

抱歉，打扰了！

回复

12. 天下第一说道：
2018年02月6日 11:15

看到了沙发想坐一下

回复

13. **Twinkleee说道：**

2018年02月5日 10:33

这个等了好久了，谢谢分享

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

- SVG <foreignObject>简介与截图等应用
- 图片动态局部毛玻璃模糊效果的实现
- CSS direction属性简介与实际应用

- 改变CSS世界纵横规则的writing-mode属性
- CSS margin-inline和margin-block区别是什么?
- canvas getImageData与任意字符图形点、线动效实现
- 小tips: 使用JS检测用户是否安装某font-family字体
- 纯前端实现可传图可字幕台词定制的GIF表情生成器
- CSS, SVG和canvas分别实现文本文字纹理叠加效果
- 图片旋转效果的一些研究、jQuery插件及实例
- 小tip: SVG和Canvas分别实现图片圆角效果