CSS届的绘图板CSS Paint API简介

这篇文章发布于 2018年11月26日，星期一，01:22，归类于 CSS相关。 阅读 3243 次, 今日 13 次 5 条评论

一、把Canvas图片作为CSS背景图片

CSS Paint API可以简单理解为（实际不能等同）把Canvas画布作为普通元素的背景图片。

也就是CSS的 `background-image` 就是一个Canvas，我们可以利用Canvas绝大多数API绘制各种复杂有趣的图形效果，以一种更高效的方式丰富web页面元素的视觉展现。例如，蓝色按钮不仅仅是个蓝色背景，上面还有白云漂漂的效果，想想就很棒！

二、一个简单的案例了解CSS Paint API

例如，我们希望创建一个透明图片背景。类似下面这样：

则完整的CSS代码和JS部分代码如下：

```
.box {
    width: 180px; height: 180px;
    /* transparent-grid自己命名 */
    background-image: paint(transparent-grid);
}
```

然后绘制图形的JS务必作为模块引入，例如，建一个名为paint-grid.js的文件，在页面上引入：

```
if (window.CSS) {
    CSS.paintWorklet.addModule('paint-grid.js');
}
```

paint-grid.js文件代码如下：

```
// transparent-grid命名和CSS中的对应
registerPaint('transparent-grid', class {
    paint(context, size) {
       // 这里就是绘制的代码了....
    }
});
```

以上就是CSS Paint API使用的固定套路：

1. CSS中paint(abc)；
2. JS添加模块CSS.paintWorklet.addModule('xxx.js')；
3. xxx.js中代码套路固定，在下面注释位置写绘制代码即可；

```
registerPaint('abc', class {
    paint(context, size, properties) {
       // 绘制代码在这里....
    }
});
```

其中`paint(context, size)`中的两个参数可以稍微介绍下：

context

为绘制上下文，全称是PaintRenderingContext2D，和Canvas的CanvasRenderingContext2D是近亲，API全部来自Canvas，一模一样，不过由于安全限制，有些Canvas中的有些API是不能使用的，可用和不可用的API见下表：

Paint可用API Paint不可用API
CanvasState
CanvasImageData
CanvasTransform
CanvasUserInterface
CanvasCompositing
CanvasText
CanvasImageSmoothing
CanvasTextDrawingStyles
CanvasFillStrokeStyles
–
CanvasShadowStyles
–
CanvasRect
–
CanvasDrawPath
–
CanvasDrawImage
–
CanvasPathDrawingStyles
–
CanvasPath
–

size

`size`是一个包含了绘制尺寸的对象，数据结构如下：

```
{
    width: 180,
    height: 180
}
```

`size`的大小受到`background-size`属性大小的影响，因此，对于重复背景，可以借助backgrou

nd-repeat进行平铺循环，不用非得在绘制的JS代码中循环。例如，下面即将要展示的demo效果，也可以这么实现，CSS部分：

```
.box {
    width: 180px; height: 180px;
    background-image: paint(transparent-grid);
    background-size: 16px 16px;
}
```

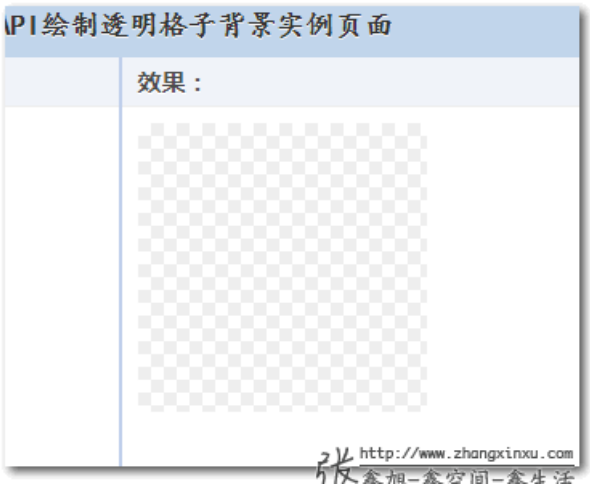
然后，paint-grid.js中只需要填充白-灰-灰-白，4个格子就好了，无需循环。

properties

可以用来获得get到的CSS属性和属性值，包括CSS变量值；以及其他一些参数。

眼见为实，您可以狠狠地点击这里：CSS Paint API绘制透明格子作为背景demo（目前仅Chrome有效果）

效果如下截图：



paint-grid.js中的完整绘制代码如下：

```
registerPaint('transparent-grid', class {
    paint(context, size) {
        // 两个格子颜色
        var color1 = '#fff', color2 = '#eee';
        // 格子尺寸
        var units = 8;
        // 横轴数轴循环遍历下
        for (var x = 0; x < size.width; x += units) {
            for (var y = 0; y < size.height; y += units) {
                context.fillStyle = (x + y) % (units * 2) === 0 ? color1 : color2;
                context.fillRect(x, y, units, units);
            }
        }
    }
});
```

补充：

类似格子这类重复背景，可以借助`background-repeat`进行平铺循环，不用非得在绘制的JS代码中循环，不过需要借助`background-size`属性帮助，改变绘制的尺寸。例如，上面demo效果，也可以这么实现，CSS部分：

```
.box {
    width: 180px; height: 180px;
    background-image: paint(transparent-grid);
    background-size: 16px 16px;
}
```

然后，paint-grid.js中只需要填充白-灰-灰-白，4个格子就好了，无需循环。

```
registerPaint('transparent-grid', class {
    paint(context, size) {
        // 两个格子颜色
        var color1 = '#fff', color2 = '#eee';
        // 两个白色格子
        context.fillStyle = color1;
        context.fillRect(0, 0, 8, 8);
        context.fillRect(8, 8, 8, 8);
        // 两个灰色格子
        context.fillStyle = color1;
        context.fillRect(0, 4, 8, 8);
        context.fillRect(4, 0, 8, 8);
    }
});
```

要更通俗易懂些。

三、CSS变量让Paint API蓬荜生辉

上面的案例展示了CSS Paint API的基本使用，但是，虽然看上去新潮，但并没有体现出CSS Paint API有什么过人之处。

你想啊，我直接用JS加Canvas API绘制一个格子图案，转换成Base64，直接作为元素的背景图片显示，不也是一样的效果，而且兼容性更好（IE9+逗支持），所有Canvas API都能用，没有限制。对比一看，完全没有使用CSS Paint API的理由嘛！

没错！如果我们只是需要一个静态背景，真不如直接Canvas绘制再转换成Base64图片（toDataURL()方法）或者Blob图片（toBlob()方法）。

CSS Paint API的优势在于：**其作为一个CSS属性值，渲染是实时的**，自动跟着浏览器重绘的，因此，只要我们的绘制是和CSS变量相关联的，所有的渲染效果都会实时刷新重绘，这可就牛逼大了！

还是上面的透明格子例子，格子的颜色以及格子的尺寸，我们可以将其作为CSS变量提取出来，如下：

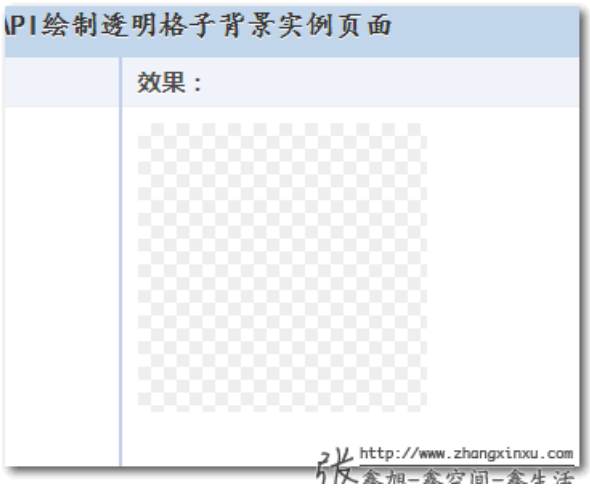
```
.box {
    width: 180px; height: 180px;
    --color1: #fff;
    --color2: #eee;
    --units: 8;
    background: paint(custom-grid);
}
```

这些定义的变量我们可以在绘制的时候获取到，示意如下：

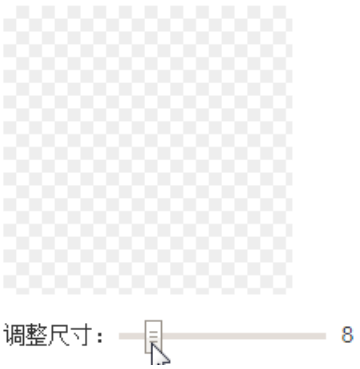
```
registerPaint('custom-grid', class {
    // 获取3个变量
    static get inputProperties() {
        return [
```

```
            '--color1',
            '--color2',
            '--units'
        ]
    }
    paint(context, size, properties) {
        // 两个格子颜色
        var color1 = properties.get('--color1').toString();
        var color2 = properties.get('--color2').toString();
        // 格子尺寸
        var units = Number(properties.get('--units'));
        // 绘制代码，和之前一样...
    }
});
```

静态效果是一样的：



但是，如果我们修改了CSS代码中定义的变量值，则，我们可以看到Paint背景图实时变化效果（见下GIF）：



眼见为实，您可以狠狠地点击这里：CSS变量外加Paint API绘制透明格子demo

无需额外的定时器，真实时渲染，控制非常方便。

配合CSS Properties & Values API，把`--units`等变量全部注册为合法的CSS属性，则，我们就能使用`transition`和`animation`属性纯CSS控制Paint背景图的运动和动画了，按钮上云朵漂漂的效果完全不在话下。

这个案例以后再介绍。

四、Houdini，兼容与其他

本文介绍的CSS Paint API是CSS Houdini的一部分，最后提到的CSS Properties & Values API也是，是目前Chrome已经支持的一部分API。CSS houdini可以自定义CSS属性，布局等，未来不可限量。

由于兼容性的问题，如果想要在实际项目中使用CSS Paint API，还需要做兼容处理，例如：

```
.box {
    width: 180px; height: 180px;
    background: url(
MElEQVQ4T2P8////fwY84P379/ikGRhHDRgWYfDu3Tu86UBQUBB/Ohg1gIFx6IcBABlfVjmYTYi7AAAAAElFTkSuQm
CC);
    background: paint(transparent-grid, whatever);
}
```

CSS Paint API更适用于动态场景，适合实现需要实时绘制渲染的需求；如果是纯静态展示，直接就用JS加Canvas实现得了，没必要为了技术而技术。

CSS Houdini及其相关的新技术可玩的东西很多，以后有机会再多多介绍，本文就到这里，感谢您的阅读！

参考文档

- https://www.w3.org/TR/css-paint-api-1/
- https://css-houdini.rocks/

最后的最后

最后，再说点其它你可能感兴趣的东西。对于本文的透明格子效果，其实最好的实现方法是直接CSS `background` 绘制，利用线性渐变和CSS3多背景。

代码如下：

```
.box {
    width: 180px; height: 180px;
    background-color: #fff;
    background-image: linear-gradient(45deg, #eee 25%, transparent 25%, transparent 75%, #
eee 75%, #eee), linear-gradient(45deg, #eee 25%, transparent 25%, transparent 75%, #eee 75
%, #eee);
    background-size: 16px 16px;
    background-position: 0 0, 8px 8px;
}
```

实时效果如下：

尺寸控制非常方便，天然支持 `animation` 动画。

（本篇完） // 想要打赏？点击这里。有话要说？点击这里。

« 纯CSS实现任意格式图标变色的研究

5分钟快速了解下CSS4 color-adjust属性 »

猜你喜欢

- 小tips:了解CSS/CSS3原生变量var
- CSS前景背景自动配色技术简介
- 小tips: 如何HTML标签和JS中设置CSS3 var变量
- 翻译 - CSS3 Backgrounds相关介绍
- 小tips: CSS3 webkit下彩条文字效果实现
- 翻译 - 解释JavaScript的“预解析(置顶解析)”
- 翻译-高质量JavaScript代码书写基本要点
- 几种纯CSS(CSS3)下的纸张效果实现展示
- 小tip:CSS3下条纹&方格斜纹背景的实现
- 视网膜New iPad与普通分辨率iPad页面的兼容处理
- 页面可用性之浏览器默认字体与CSS中文字体

标签： background, background-size, canvas, css3, css相关, houdini, Paint API, paintWorklet, Properties-Values API, registerPaint, var, 变量

发表评论（目前5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. **Ale-cc说道：**
2018年12月3日 09:44

Houdini确实令人期待~

回复

2. **coolb**说道：
2018年11月30日 15:34

气死我了 有个错别字 「IE9+逗支持」

回复

3. **XboxYan**说道：
2018年11月28日 14:08

CSS发展的太慢了，一直非常看好这个特性的潜能

回复

4. **GoStop**说道：
2018年11月27日 14:08

貌似只能在https环境下才能使用CSS.paintWorklet等相关属性

回复

5. 好多水果说道：
2018年11月26日 11:34

旭哥，高产似母猪啊，太强了！

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 (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

- 小tips:了解CSS/CSS3原生变量var
- CSS前景背景自动配色技术简介
- 小tips: 如何HTML标签和JS中设置CSS3 var变量
- 翻译 - CSS3 Backgrounds相关介绍
- 小tips: CSS3 webkit下彩条文字效果实现
- 翻译 - 解释JavaScript的“预解析(置顶解析)”
- 翻译-高质量JavaScript代码书写基本要点
- 几种纯CSS(CSS3)下的纸张效果实现展示
- 小tip:CSS3下条纹&方格斜纹背景的实现
- 视网膜New iPad与普通分辨率iPad页面的兼容处理
- 页面可用性之浏览器默认字体与CSS中文字体