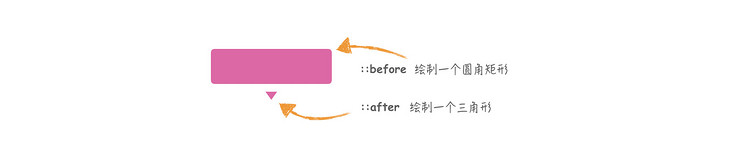
今天来看一种十分常见的交互：**提示框（tooltips）**。通常提示框都是纯色的，比如下面这个



这类布局实现还不算复杂，可以用一个圆角矩形和一个小三角拼接形成，设置相同的颜色就可以了

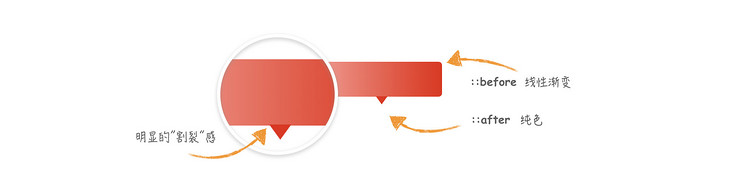


这个并不是本文的重点，有兴趣的可以访问 [css-tips (codepen.io)](https://codepen.io/xboxyan/pen/MLJjWQ)点击预览

有时候，为了突出强调产品的特点或者为了跟随设计的潮流，设计会用上渐变背景，比如 [lulu UI Edge](https://www.zhangxinxu.com/sp/lulu/mockup/edge/about.design.html) 版本中的 Tips 组件



如果仍然采用**“拼接”**的方式，不可避免会出现衔接不上的问题，有明显的**“割裂”**感，视觉还原会大打折扣



那么，如何实现这类效果呢？一起来看看吧

一、clip-path 裁剪

[**clip-path**](https://developer.mozilla.org/zh-CN/docs/Web/CSS/clip-path) 可能是很多人马上就能想到的方式。但是实际操作下来，还是会遇到很多麻烦

1. **clip-path: path** 可以支持任意形状，但是却没法实现自适应宽高
2. **clip-path: polygon** 可以实现小尖角，但是无法实现圆角
3. **clip-path: inset** 可以实现自适应圆角矩形，但是无法实现下方的小尖角

如何解决这个问题呢？其实把 2 和 3 结合起来就可以了

这里需要两个相同大小的容器，可以用 **::before** 和 **::after** 来代替，然后设置相同的背景色，可以通过自定义属性定义

.tips{

position: relative;

--bg: linear-gradient(45deg, #ff3c41, #ff9800);

}

.tips::before,.tips::after{

content:'';

position: absolute;

width: 100%;

height: 100%;

left: 0;

top: 0;

background: var(--bg);*/\*完全相同的背景\*/*

z-index: -1;

}

**为什么要用两个相同大小的容器呢？** 这是为了保证接下来渐变背景在裁剪时完全吻合



接着其中一个裁剪成圆角矩形，另外一个裁剪成小三角，然后**重叠**起来就可以了

.tips::before{

clip-path: inset(0 0 5px 0 round 5px);

*/\*round 可以设置圆角\*/*

}

.tips::after{

clip-path: polygon(calc(50% - 5px) calc(100% - 5px), calc(50% + 5px) calc(100% - 5px), 50% 100%);

*/\* 实现小三角，只需要3个点的坐标就可以了 \*/*

}

可以看到提示框完全是自适应的，实时效果如下



完整代码可访问 [tooltips-clip-path (codepen.io)](https://codepen.io/xboxyan/pen/ExWLyKR)点击预览

二、mask 遮罩

除了 **clip-path** ，**mask** 也是一种思路。如果还不熟悉 **mask**，可以参考这一篇 [奇妙的 CSS MASK (juejin.im)](https://juejin.cn/post/6846687594693001223) 。这里的原理如下

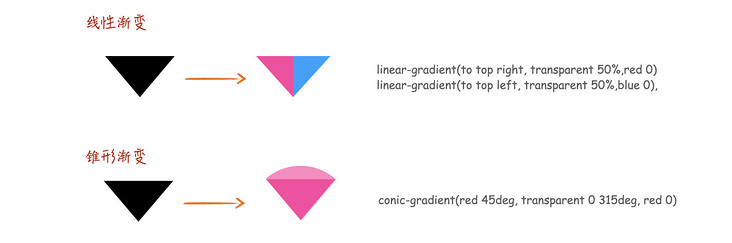


利用 **mask** ，现在的问题就转变成了：**如何通过 CSS 绘制这样一个图形？**



1. 万能的 gradient

没有什么图形是 **CSS 渐变** 绘制不出来的，这个也不例外。首先我们把这个图形进行分解，这里可以分成一个**圆角矩形**和一个**三角形**，三角形比较容易，可以通过 **conic-gradient** 或者 **linear-gradient** 绘制



圆角矩形就稍微有点麻烦了，不过还是可以分解的，如下



可以由4个径向渐变和2个线性渐变合成，用代码实现就是

tips{

-webkit-mask-image:

/\*4个径向渐变和2个线性渐变\*/

radial-gradient(circle at 5px 5px, green 5px,transparent 0),

radial-gradient(circle at 5px 5px, green 5px,transparent 0),

radial-gradient(circle at 5px 5px, green 5px,transparent 0),

radial-gradient(circle at 5px 5px, green 5px,transparent 0),

linear-gradient(red,red),

linear-gradient(blue,blue);

-webkit-mask-size:

10px 10px,

10px 10px,

10px 10px,

10px 10px,

100% calc(100% - 15px),

calc(100% - 10px) calc(100% - 5px)

-webkit-mask-position:

left top,

right top，

left 0 bottom 5px,

right 0 bottom 5px,

left 5px,

5px top;

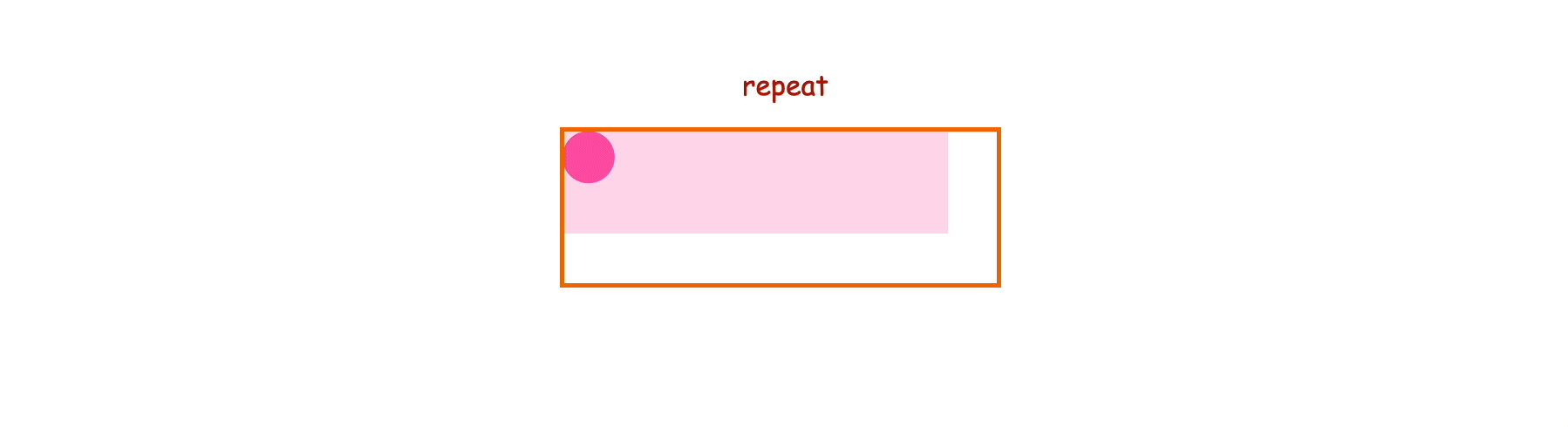
-webkit-mask-repeat: no-repeat

}

只要有点耐心，都可以很顺利的写出来

但是...

太长了，有很多重复的（4个radial-gradient），非常啰嗦，有没有什么办法优化呢？这里有一个技巧，碰到重复有规律的东西，可以多想想 **repeat**，利用背景的平铺特性，合理设置背景尺寸就可以了，如下



可以看到，背景尺寸设置成 **calc(100% - 10px)** 就可以达到平铺效果，代码实现就是

tips{

-webkit-mask-image:

/\*只需要一个径向渐变即可\*/

radial-gradient(circle at 5px 5px, green 5px,transparent 0),

linear-gradient(red,red),

linear-gradient(blue,blue);

-webkit-mask-size:

calc(100% - 10px) calc(100% - 15px),/\*圆角的尺寸，高度由于还需要减去三角形尺寸，所以多了5px\*/

100% calc(100% - 15px),

calc(100% - 10px) calc(100% - 5px);

-webkit-mask-position:

left top,

left 5px,

5px top;

-webkit-mask-repeat: repeat,no-repeat,no-repeat;

}

是不是精简了许多？然后再把三角形的合过来就行了，可以得到如下效果



完整代码可访问 [tooltips-mask-gradient (codepen.io)](https://codepen.io/xboxyan/pen/KKWRWOj)点击预览

2. 自适应的svg

尽管做了一些优化，上面的代码量仍然非常可观，有没有更加简便的方式呢？

想到了 svg...

一般情况下，svg 路径是固定尺寸的，只能 **等比缩放** ，无法做到自适应。不过基本图形是支持自适应的，可以设置百分比尺寸，比如 <rect>

<svg xmlns='http://www.w3.org/2000/svg' width='100%' height='100%'>

<rect rx="5" width='100%' height='100%'/>

</svg>

rx 可以设置矩形的圆角，当不设置 ry 时，默认与 rx 相同

这样一个 svg 是可以自适应的，在改变尺寸的情况下不会变形（注意观察圆角），如下



三角形就很容易了，可以用 <polygon> 实现

<svg xmlns='http://www.w3.org/2000/svg'>

<polygon points='0 0,10 0,5 5' />

</svg>

然后，把两段 svg 直接用作遮罩背景就行了，可以用 mask-size 和 mask-position 分别设置 **尺寸** 和 **位置**

tips{

-webkit-mask-image: url("data:image/svg+xml,<svg xmlns='http://www.w3.org/2000/svg'><polygon points='0 0,10 0,5 5' /></svg>"),url("data:image/svg+xml,<svg xmlns='http://www.w3.org/2000/svg'><rect rx='6' width='100%' height='100%'/></svg>");

-webkit-mask-size: 10px 5px, 100% calc(100% - 5px);

-webkit-mask-repeat: no-repeat;

-webkit-mask-position: center bottom, 0 0;

}

**svg** 用作背景需要在前面添加 data:image/svg+xml ，并且内容需要转义，详细可参考这篇文章： [学习了，CSS中内联SVG图片有比Base64更好的形式](https://www.zhangxinxu.com/wordpress/2018/08/css-svg-background-image-base64-encode/)

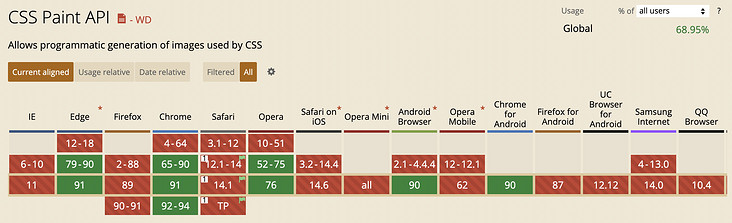
还是挺不错的，代码量也不多，也比较容易理解，实时效果如下



完整代码可访问 [tooltips-mask-svg (codepen.io)](https://codepen.io/xboxyan/pen/poeVrqB)点击预览

三、paint 绘制

再来介绍一种未来的解决方式， [CSS paint](https://developer.mozilla.org/en-US/docs/Web/API/CSS_Painting_API) 。关于 **CSS paint**，又称 “CSS 界的绘图板”，简单来说，就是用 canvas 绘图的方式来绘制背景，canvas 几乎什么都能绘制吧，所以这是一种更为通用的解决方案。想快速了解 **CSS paint** 的可以参考这一篇入门文章：[CSS届的绘图板CSS Paint API简介](https://www.zhangxinxu.com/wordpress/2018/11/css-paint-api-canvas/)，不过目前仅支持 Chrome，兼容性如下



不过并不影响我们的学习，毕竟是未来的解决方案，先看看大致的语法，如下

1. 首先，JS 注册模块 **registerPaint**

*// paint-tips.js*

registerPaint('tips-bg', **class** {

**paint**(ctx, size, properties) {

*// 在这里绘制背景，语法和canvas类似*

}

});

1. 接着，JS 添加模块 **CSS.paintWorklet.addModule**

**if** (window.CSS) {

CSS.paintWorklet.addModule('paint-tips.js');

}

1. 最后，CSS 中使用 **paint(tips-bg)**

tips{

-webkit-mask-image: paint(tips-bg); */\*这里作为遮罩背景使用\*/*

}

下面就来绘制提示框了，如果仍然借助 mask ，那么问题就变成了：**如何通过 canvas 绘制这样一个图形？**



在 canvas 中，相对于 CSS 来说， 这类图形简直就是小儿科，只需要使用 **lineTo** 和 **arc** 两个指令就可以绘制了。最关键的一点是，这里的尺寸是实时渲染的，可以通过 size 来获取

关于 canvas 学习，这里推荐一下[张鑫旭老师](https://www.zhangxinxu.com/)的 [Canvas API中文文档](https://www.canvasapi.cn/)，api 和 案例比 mdn 文档清晰太多

绘制代码如下（下面就是很普通的 canvas 代码了，其实就是几段线段连接起来，然后填充纯色）

registerPaint('tips-bg', **class** {

**paint**(ctx, size) { *// ctx为绘制上下文，size为容器尺寸*

**const** { width,height } = size; *// 容器尺寸*

**const** radius = 5; *// 圆角大小*

**const** deg = Math.PI / 2;

**const** edge = 5; *// 三角形大小*

**const** pos = width / 2; *// 三角形位置*

ctx.beginPath();

ctx.moveTo(radius,0);

ctx.lineTo(width-2\*radius,0);

ctx.arc(width-radius,radius,radius,-deg,0);

ctx.lineTo(width,height-2\*radius-edge);

ctx.arc(width-radius,height-radius-edge,radius,0,deg);

ctx.lineTo(pos+edge,height-edge);

ctx.lineTo(pos,height);

ctx.lineTo(pos-edge,height-edge);

ctx.lineTo(radius,height-edge);

ctx.arc(radius,height-radius-edge,radius,deg,2\*deg);

ctx.lineTo(0,radius-edge);

ctx.arc(radius,radius,radius,-2\*deg,-deg);

ctx.closePath();

ctx.fillStyle = '#000';

ctx.fill();

}

});

实时效果如下



完整代码可访问 [tooltips-mask-paint (codepen.io)](https://codepen.io/xboxyan/pen/JjWvxEN)点击预览

另外，也可以通过 CSS 变量进行自定义，比如定义一个--r为圆角大小，--t为三角形大小

<tips style="--r:5;--t:5"></tips>

registerPaint('tips-bg', **class** {

**static** **get** **inputProperties**() { *// 定义允许的自定义属性*

**return** [

'--r',

'--t'

]

}

**paint**(ctx, size, properties) { *// properties为自定义属性*

**const** radius = Number(properties.get('--r'));

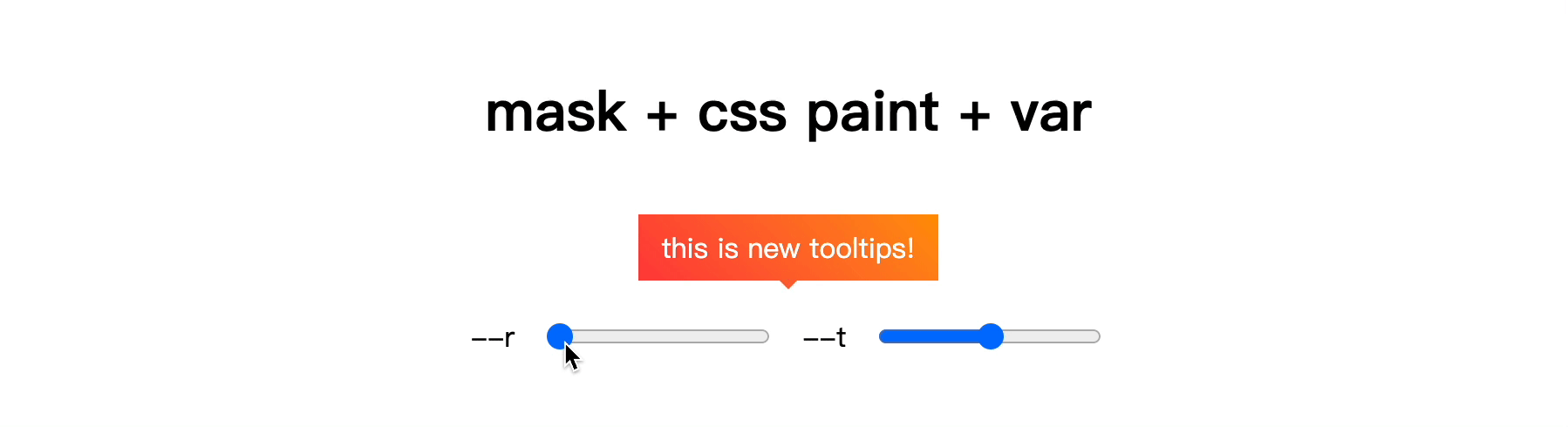
**const** edge = Number(properties.get('--t'));

*// ...*

}

})

可以看到绘制是实时更新的（改变圆角），无需 JS 额外处理，实时效果如下



完整代码可访问 [tooltips-mask-paint-var (codepen.io)](https://codepen.io/xboxyan/pen/JjWvVMd)点击预览

四、描边效果

有时候提示框可能还会有描边的效果，比如这样的



这类带描边的其实以上方式都不太适用，**clip-path** 和 **mask** 都无法实现描边，不过有一个边框生成方案可以参考：[有意思！不规则边框的生成方案 (juejin.cn)](https://juejin.cn/post/6944892753402822686)，可惜效果不是特别完美（略微模糊）

如果尺寸固定，那么可以直接使用 **svg** 方式，参考这篇文章：[用SVG实现一个优雅的提示框 (juejin.cn)](https://juejin.cn/post/6926353919333531661)

就目前而言，确实没有比较好的实现方案（有更好的实现方式欢迎补充😂，我暂时想不出来了），不过如果借助 **CSS paint** ，那一切就都有可能了！只需要在 **paint** 函数中绘制边框和背景就行了

绘制代码如下

registerPaint('tips-bg', **class** {

**paint**(ctx, size) {

**const** { width,height } = size; *// 容器尺寸*

**const** radius = 5; *// 圆角大小*

**const** deg = Math.PI / 2;

**const** edge = 5; *// 三角形大小*

**const** pos = width / 2; *// 三角形位置*

**const** lineWidth = 2; *// 描边宽度*

ctx.beginPath();

ctx.moveTo(radius+lineWidth,lineWidth);

ctx.lineTo(width-2\*radius-lineWidth,lineWidth);

ctx.arc(width-radius-lineWidth,radius+lineWidth,radius,-deg,0);

ctx.lineTo(width-lineWidth,height-2\*radius-edge-lineWidth);

ctx.arc(width-radius-lineWidth,height-radius-edge-lineWidth,radius,0,deg);

ctx.lineTo(pos+edge,height-edge-lineWidth);

ctx.lineTo(pos,height-lineWidth);

ctx.lineTo(pos-edge,height-edge-lineWidth);

ctx.lineTo(radius+lineWidth,height-edge-lineWidth);

ctx.arc(radius+lineWidth,height-radius-edge-lineWidth,radius,deg,2\*deg);

ctx.lineTo(lineWidth,radius+lineWidth);

ctx.arc(radius+lineWidth,radius+lineWidth,radius,-2\*deg,-deg);

ctx.closePath();

**const** gradient = ctx.createLinearGradient(0, 0, width, 0); *// 渐变背景*

gradient.addColorStop(0, '#F57853');

gradient.addColorStop(1, '#F8B578');

ctx.fillStyle = gradient;

ctx.fill();

ctx.strokeStyle = '#FBF8F8'; *// 绘制边框*

ctx.lineWidth = lineWidth;

ctx.lineCap = 'round';

ctx.stroke();

}

});

tips{

*/\* -webkit-mask-image: paint(tips-bg); \*/*

background: paint(tips-bg); */\*不再借助mask，纯js绘制背景，包括渐变\*/*

}

实时效果如下



完整代码可访问 [tooltips-paint-stroke (codepen.io)](https://codepen.io/xboxyan/pen/QWpxdVp)点击预览

五、总结和说明

以上针对 **tooltips 布局**共介绍了3种不同类型的实现方式，分别是 **clip-path**、**mask**、**CSS paint**。其中 mask 的实现重点其实是CSS图形的绘制，主要有 **渐变** 和 **svg** 两种，虽然 **渐变** 的写法稍微复杂一点，但是最为通用，其他方式可能换一种布局就不适用了。现在总结一下要点：

1. 可以用多个容器重叠配合 clip-path 实现复杂的自适应效果
2. 在使用 CSS 渐变绘制图形时，相同的形状充分利用平铺特性
3. svg 基本形状支持百分比尺寸，用作背景同样有效，可以使用多张背景来组合
4. CSS paint 是未来的最佳解决方式，也能轻易的实现描边效果
5. CSS paint 唯一的缺陷是兼容性不够好（现仅支持 Chrome 65+ ），但是值得学习

当然，这些方式不仅仅是实现本文的布局而已，更多的是提供一种思路，下次碰到其他的**“异形布局”**也能马上联想出相应的解决方案，而不