# \_fold: function (point) { br为例分析

目标：根据目标p点，计算各种css参数(transformation)，显示翻折到这个c点的效果图

参数point：corner对象 corner对象 { x: c.x, y: c.y, corner: '??' }, 这里的x y 都此帧的目标函数

var data = this.data().f,

获得flip对象的数据

turnData = data.opts.turn.data(),

获得turn对象数据

o = flipMethods.\_c.call(this, point.corner),

o = point2D(this.width() - 0, this.height() - 0); 即o为折角的坐标点(width, height)。



width = this.width(),

页宽width

height = this.height();

页高height

switch (data.effect) {

case 'sheet':

var that = this,

a = 0,

alpha = 0,

beta,

px,

gradientEndPointA,

gradientEndPointB,

gradientStartVal,

gradientSize,

gradientOpacity,

shadowVal,

mv = point2D(0, 0),

df = point2D(0, 0),

tr = point2D(0, 0),

folding = flipMethods.\_foldingPage.call(this),

flipMethods.\_foldingPage() 找到正发生折角的页的pageObjs 用于做背面，就是折出来的那个

tan = Math.tan(alpha),

y轴 = x轴\*tan alpha.

ac = turnData.opts.acceleration,

硬件加速，不知道有什么用

h = data.wrapper.height(),

h为页高

top = point.corner.substr(0, 1) == 't',

top标志判断，布尔值

left = point.corner.substr(1, 1) == 'l',

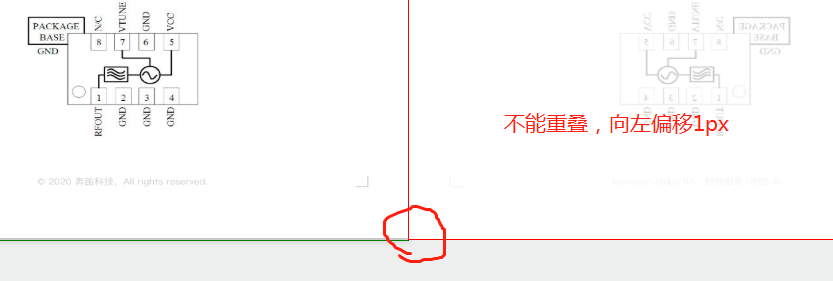
left标志判断，布尔值

switch (point.corner) {

case 'br':

point.x = Math.min(point.x, width - 1);

1是向左偏移，避免重叠用的，这里我称之为**x坐标最小值规范，必须不能超过页宽**



## compute();

var rel = point2D(0, 0);

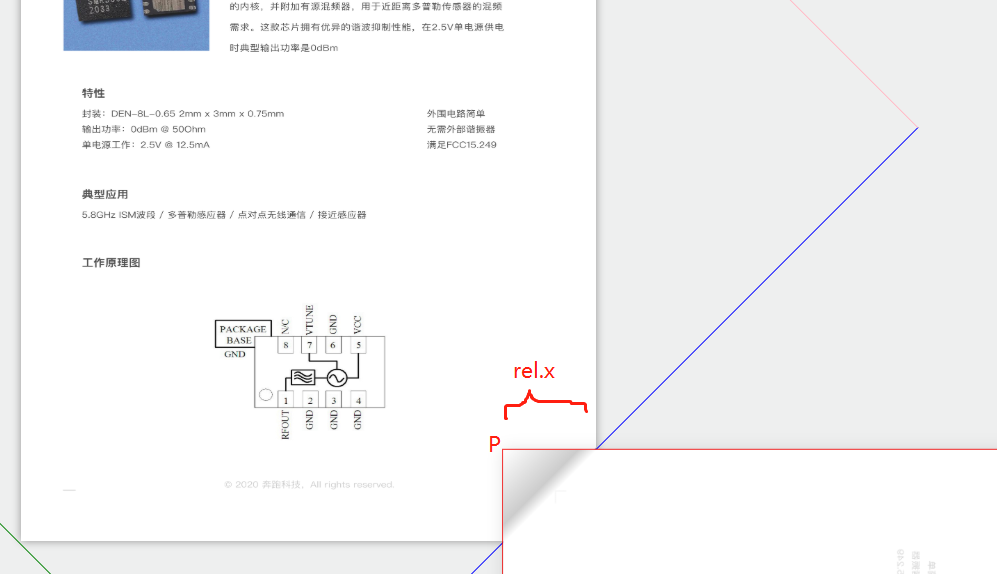
点rel=(0,0)

var middle = point2D(0, 0);

点middle=(0,0)

rel.x = (o.x) ? o.x - point.x : point.x;

在br中，不存在o.x为0，故这个rel.x= o.x - point.x,



if (!hasRot) {

rel.y = 0;

} else {

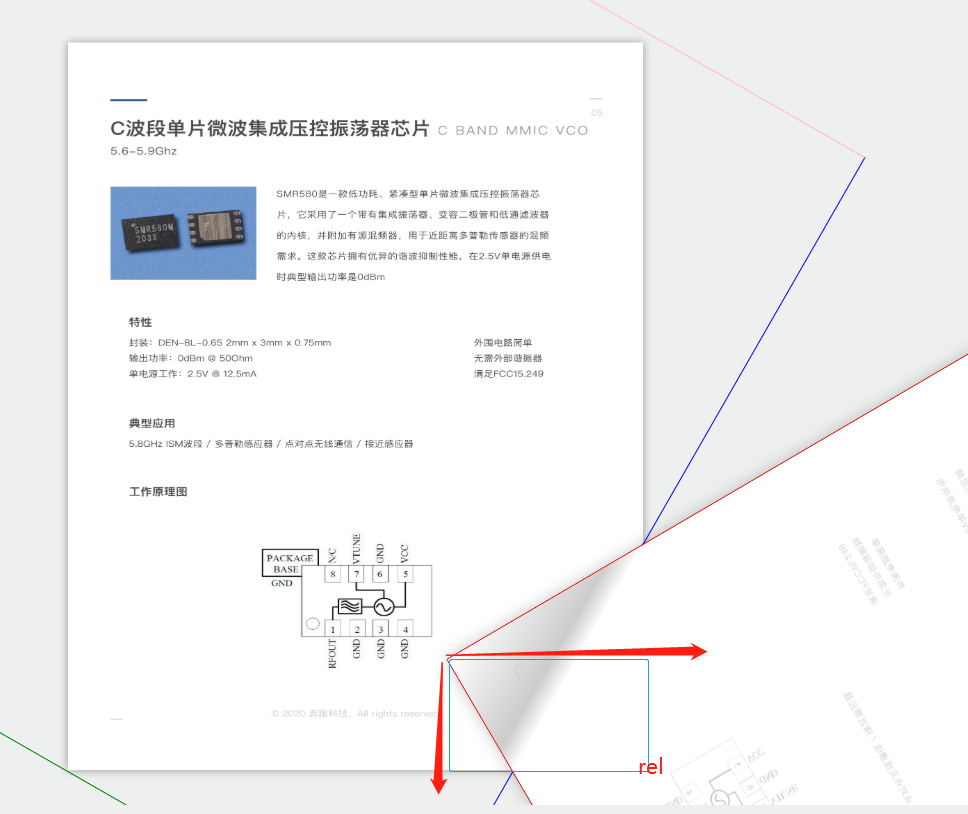
rel.y = (o.y) ? o.y - point.y : point.y;

}

// Webkit 534.3 on Android wrongly repaints elements that use overflow:hidden + rotation

考虑到webkit 小于534.3版本 某个**bug , overflow+rotation 会出现问题，**hasRot就为flase.大部分都是true.

所以, rel.y = o.y - point.y，由图可知，这个rel点：



middle.x = (left) ? width - rel.x / 2 : point.x + rel.x / 2;

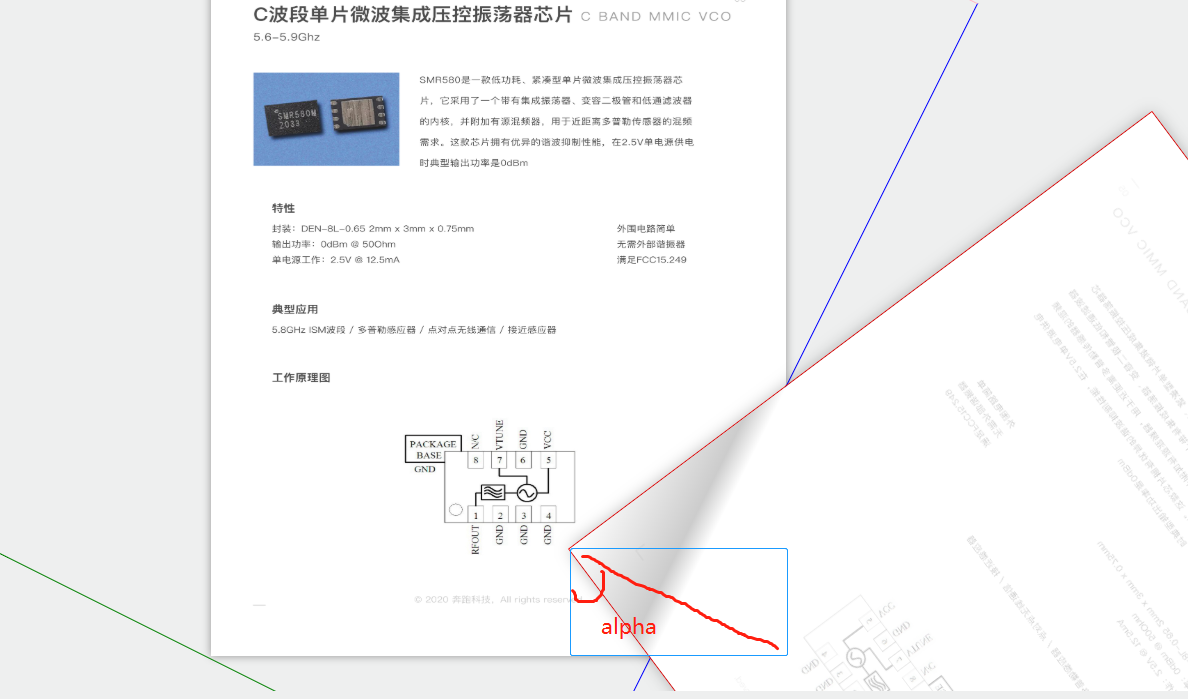
middle.y = rel.y / 2;

br的情况下：middle.x = point.x + rel.x / 2，middle.y = rel.y / 2;

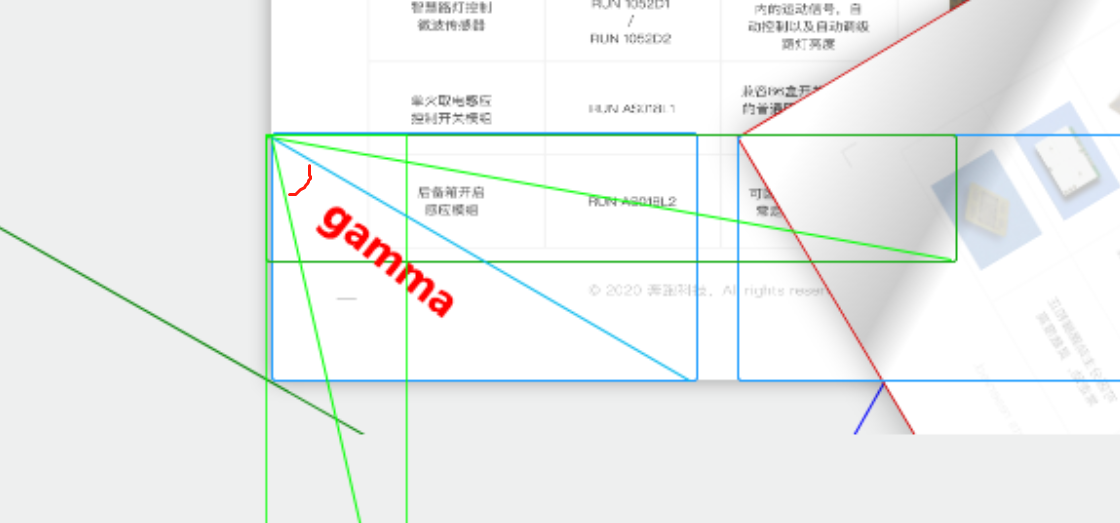


var alpha = A90 - Math.atan2(rel.y, rel.x),

A90 = PI / 2,就是90度



gamma = alpha - Math.atan2(middle.y, middle.x),



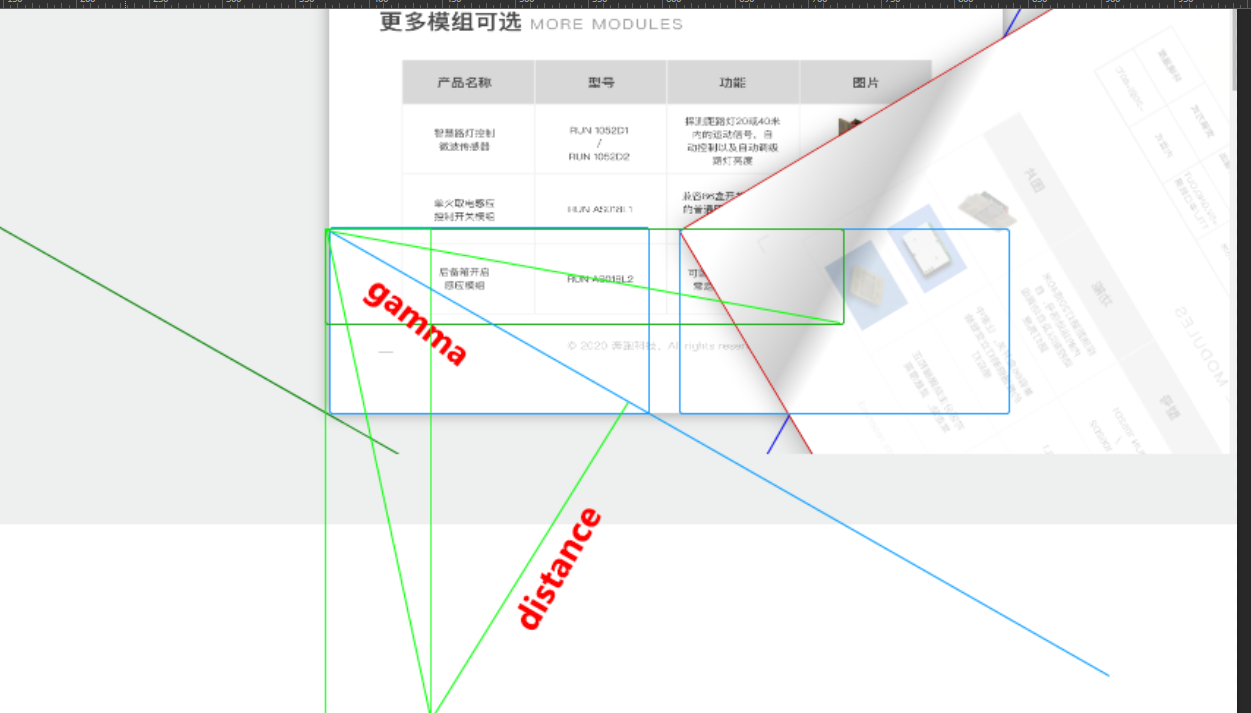
gammar角度如上，但不明确

distance = Math.max(0, Math.sin(gamma) \* Math.sqrt(Math.pow(middle.x, 2) + Math.pow(middle.y, 2)));

a = Math.sin(gamma)

b = Math.sqrt(Math.pow(middle.x, 2) + Math.pow(middle.y, 2))

distance = Math.max(0, a\*b)



开始迷糊了，这个距离又什么用？

a = deg(alpha);

a就是角度数值了。

tr = point2D(distance \* Math.sin(alpha), distance \* Math.cos(alpha));

if (alpha > A90) {

先不考虑这种情况！

tr.x = tr.x + Math.abs(tr.y \* rel.y / rel.x);

tr.y = 0;

if (Math.round(tr.x \* Math.tan(PI - alpha)) < height) {

point.y = Math.sqrt(Math.pow(height, 2) + 2 \* middle.x \* rel.x);

if (top) point.y = height - point.y;

return compute();

}

}

if (alpha > A90) {

先不考虑这种情况！

var beta = PI - alpha, dd = h - height / Math.sin(beta);

mv = point2D(Math.round(dd \* Math.cos(beta)), Math.round(dd \* Math.sin(beta)));

if (left) mv.x = - mv.x;

if (top) mv.y = - mv.y;

}

var side = width - px,

sideX = side \* Math.cos(alpha \* 2),

sideY = side \* Math.sin(alpha \* 2);

## transform(point2D(-tr.x, -tr.y), [0, 1, 1, 0], [0, 100], a);

break;