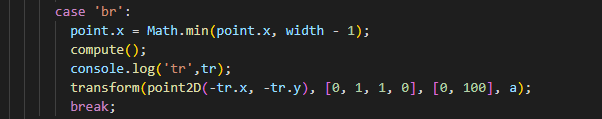


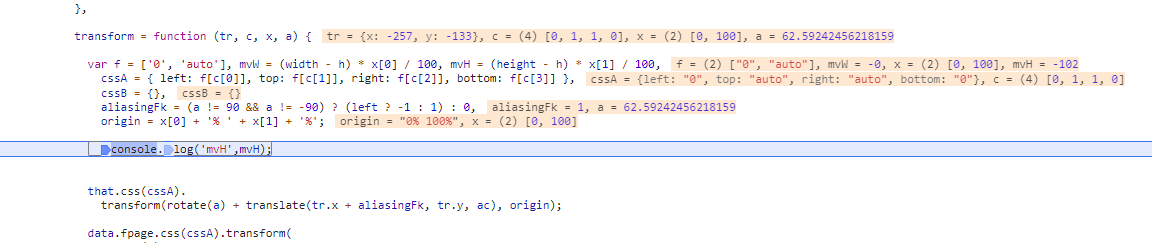
var f = ['0', 'auto'], mvW = (width - h) \* x[0] / 100, mvH = (height - h) \* x[1] / 100,

cssA = { left: f[c[0]], top: f[c[1]], right: f[c[2]], bottom: f[c[3]] },

# cssA得知[0,1,1,0]的意义



以br为例，下断点测试：

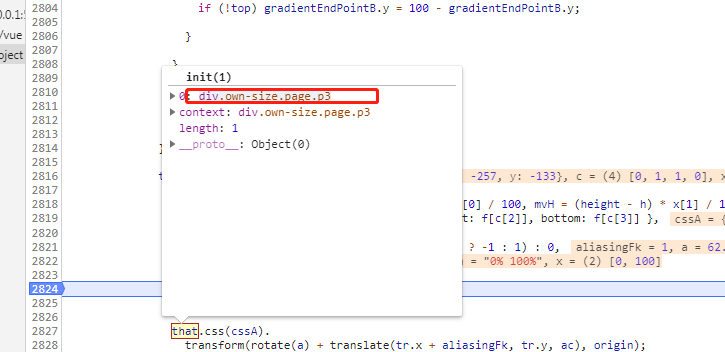


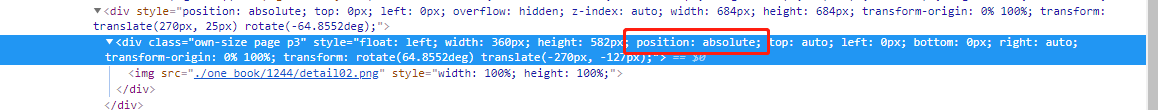
由后面

that.css(cssA).

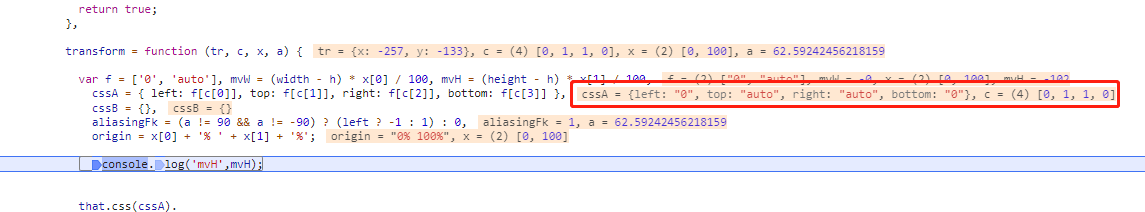
transform(rotate(a) + translate(tr.x + aliasingFk, tr.y, ac), origin);

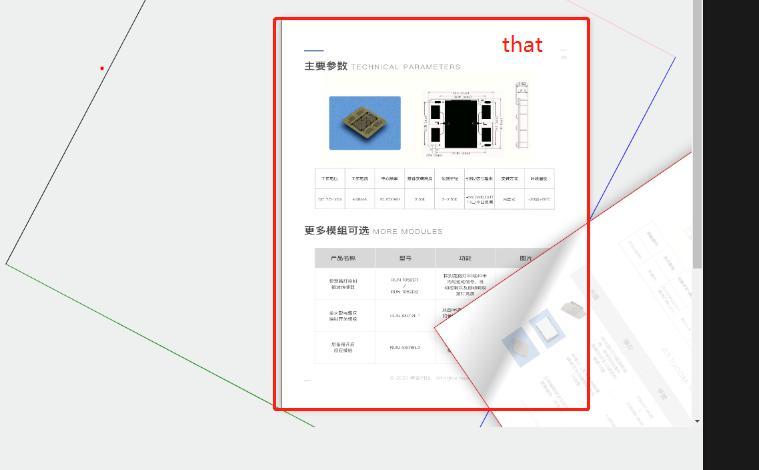
That指是下图



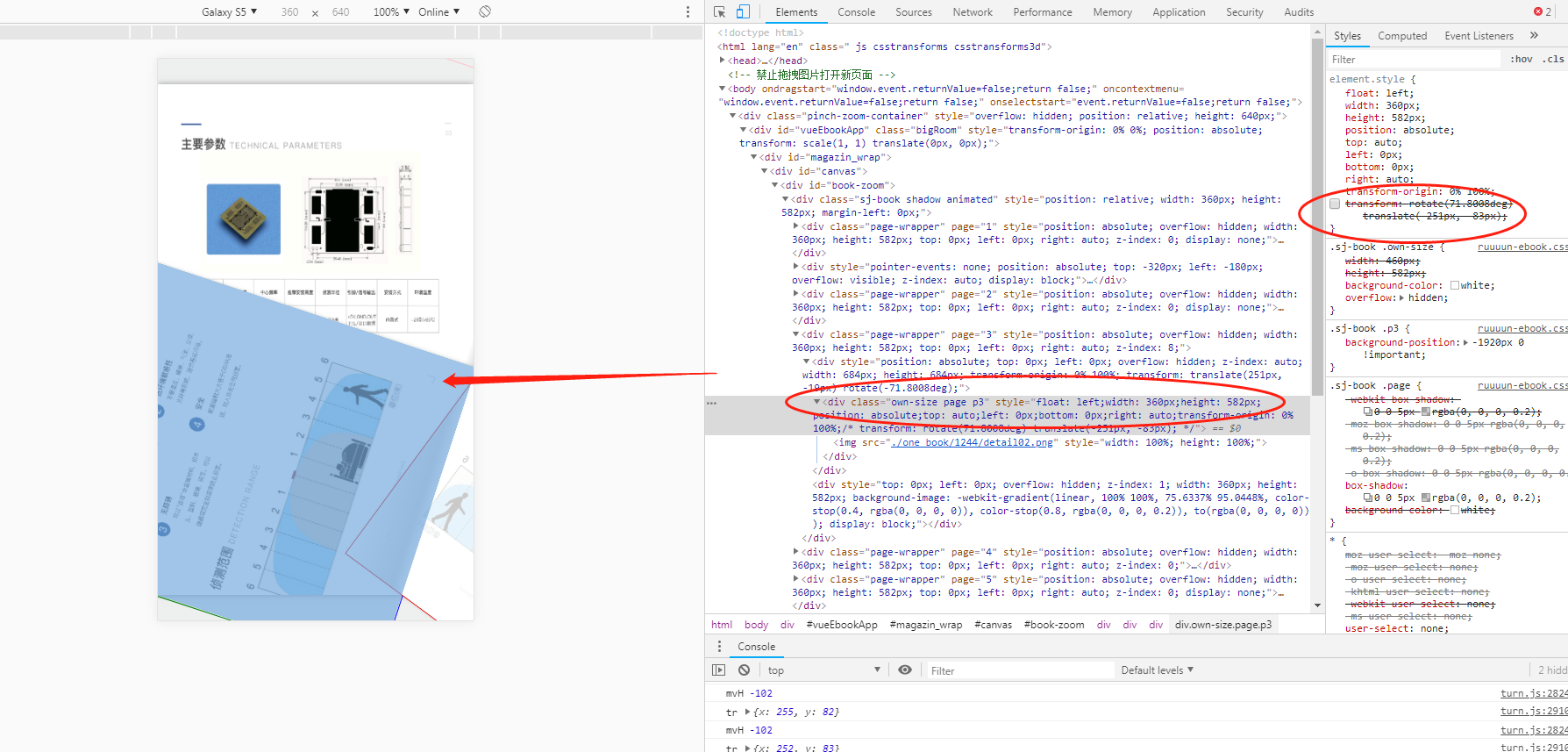


得知，cssA的作用是postion定位，

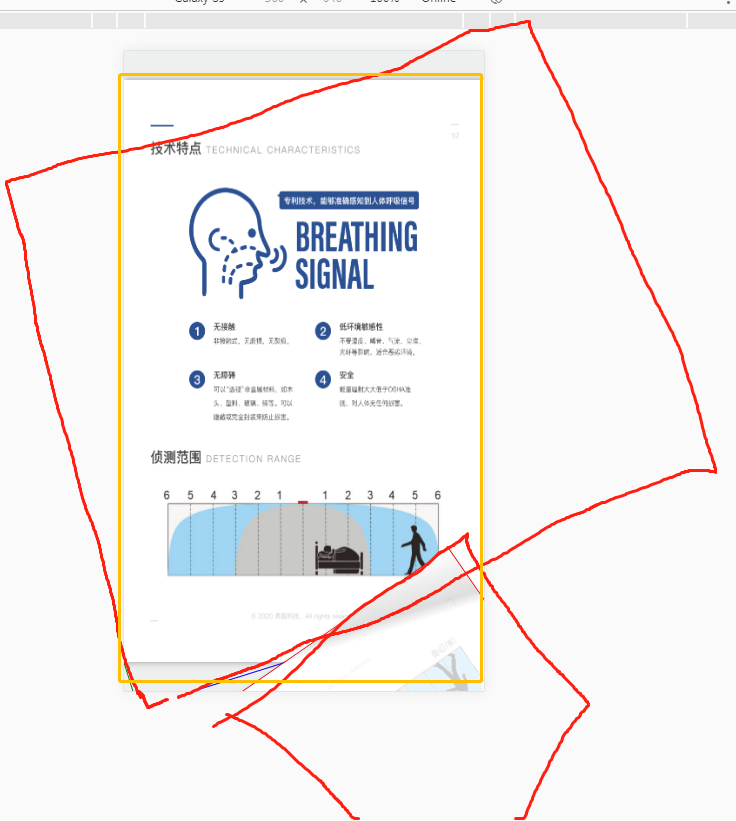




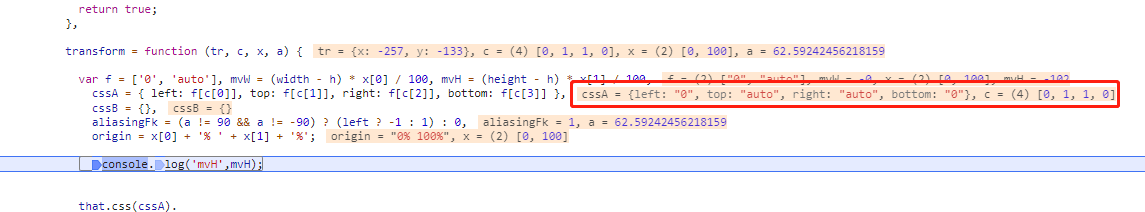
由此得知，是用left和bottom紧贴着父元素的左下角定位，如果先去掉transform,如下图



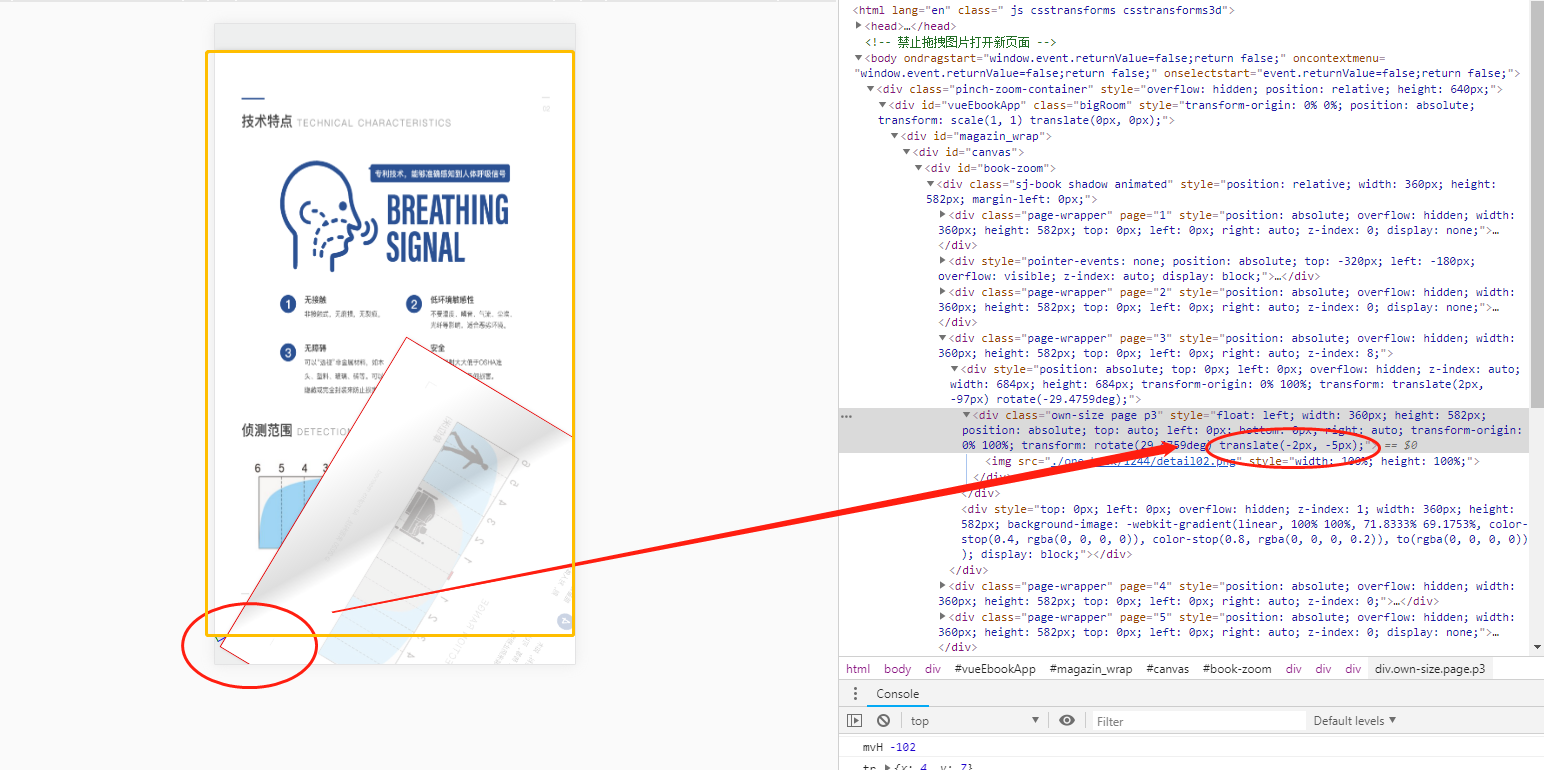
由此得知，cssA的作用是定位当前页，而cssA由第二各参数如[0, 1, 1, 0]确定，得知[0, 1, 1, 0]的作用是确定当前页的定位，至于为什么是0110,看下图：



Br这个角，就是围绕着左下角为中心进行旋转，所以当前页就是用left ,bottom左下角来定位，故就是[0,1,1,0] ，如下图：

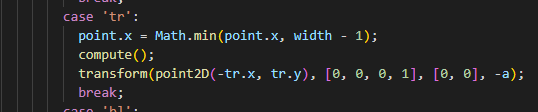


当折角碰到边的时候，怎么做到卡位？我们看下图：

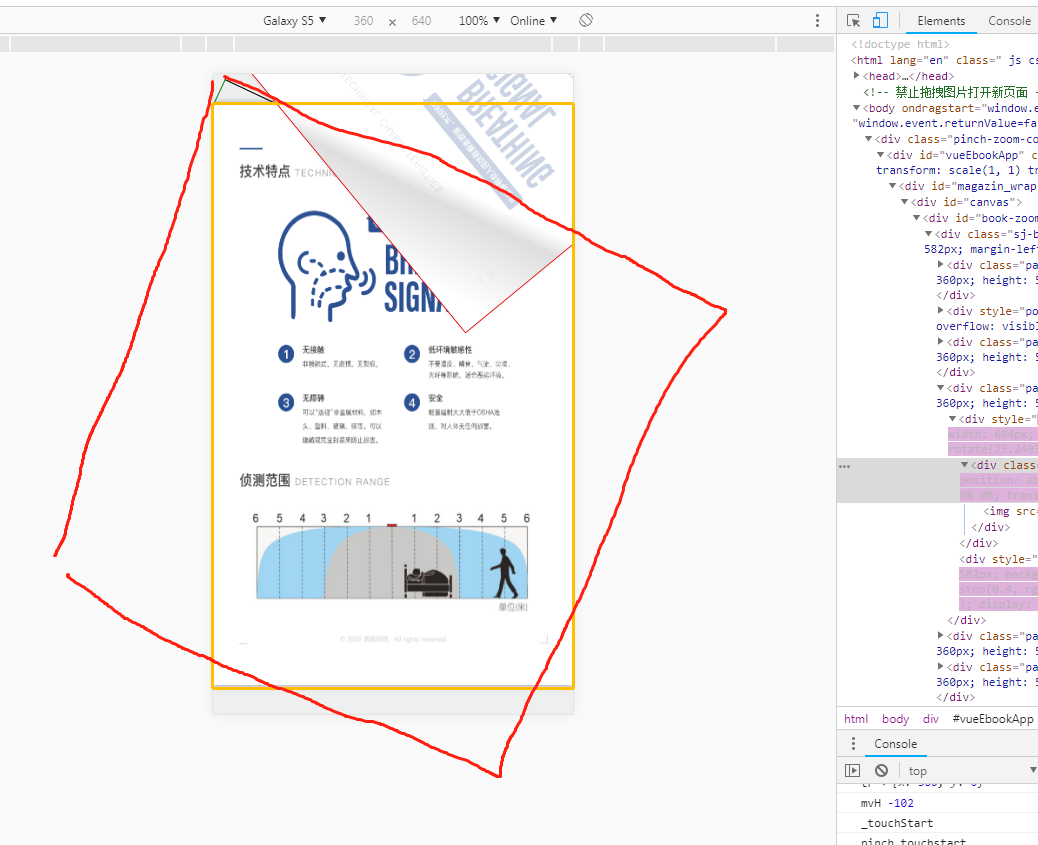


上图，that就是黄色框，cssA控制的部分，我们可以看到，tanslate的参数接近0，这个就是作为判断标准就可以实现卡位效果了。

那么，为什么tr是[0,0,0,1]呢？

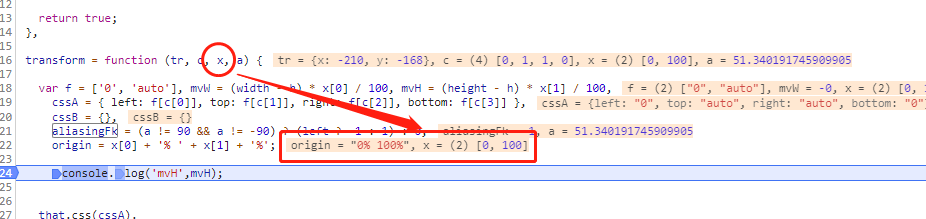


看下图：



黄色是that,父元素是红色框，可以看到通过top+left的定位+translate就可以形成这样的效果，故必须是[0,0,?,1],由于left优先级高于right，所以第三个无所谓，试过了，我把tr改成[0, 0, 1, 1]无变化。

# Origin得知第三个参数是旋转点

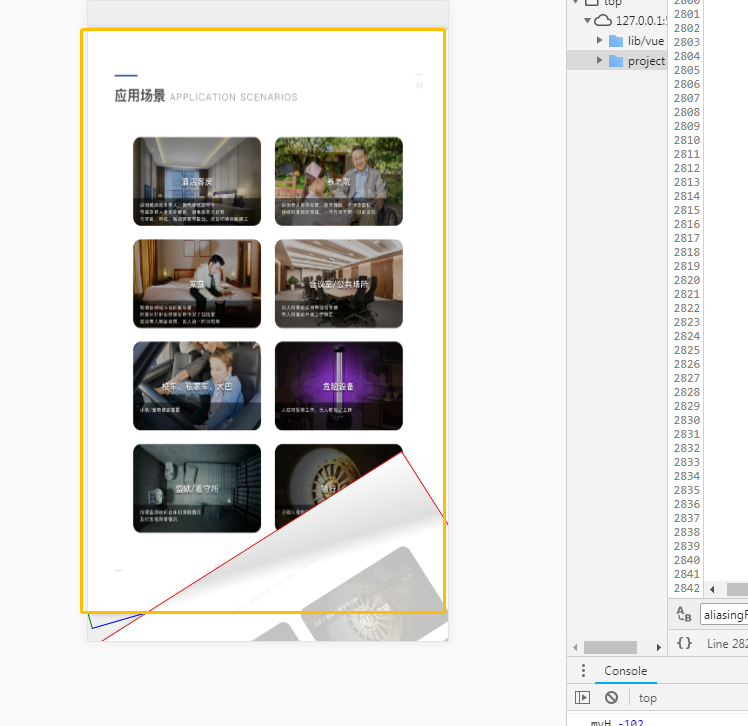


为什么 br 是 0% 100% ?

看后面代码用到了origin：

that.css(cssA).

transform(rotate(a) + translate(tr.x + aliasingFk, tr.y, ac), origin);

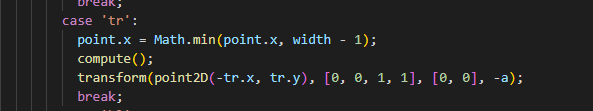


得知，that（黄色）通过left+bottom定位，需要通过调整origin 为0% 100% 实现旋转.

origin的应用 <https://c.runoob.com/codedemo/3391/>

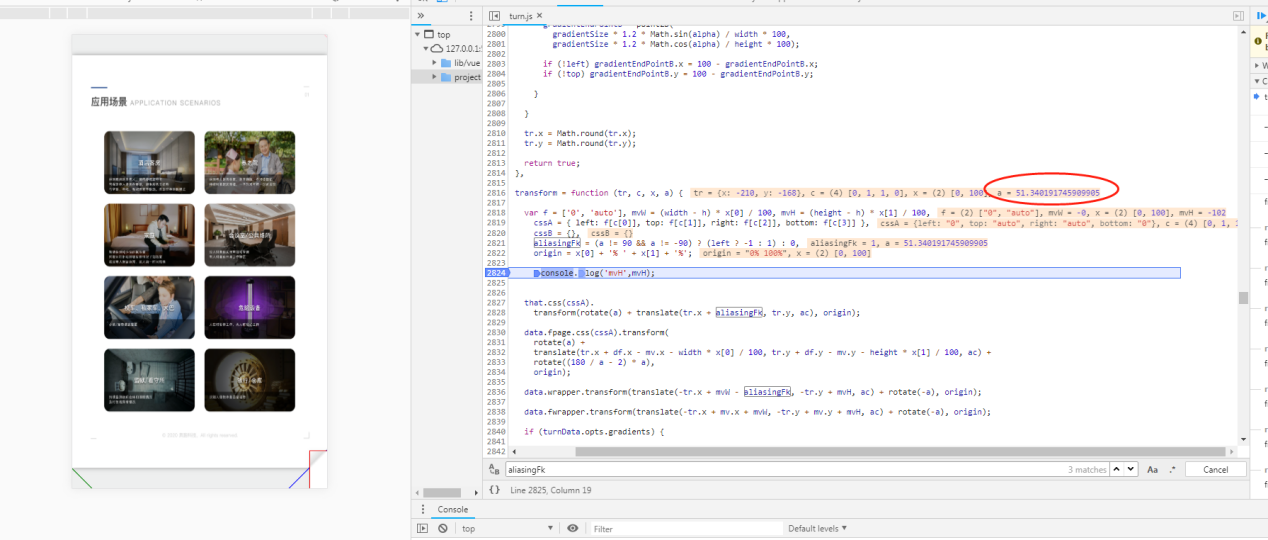
因此，第三个参数是[0,100]

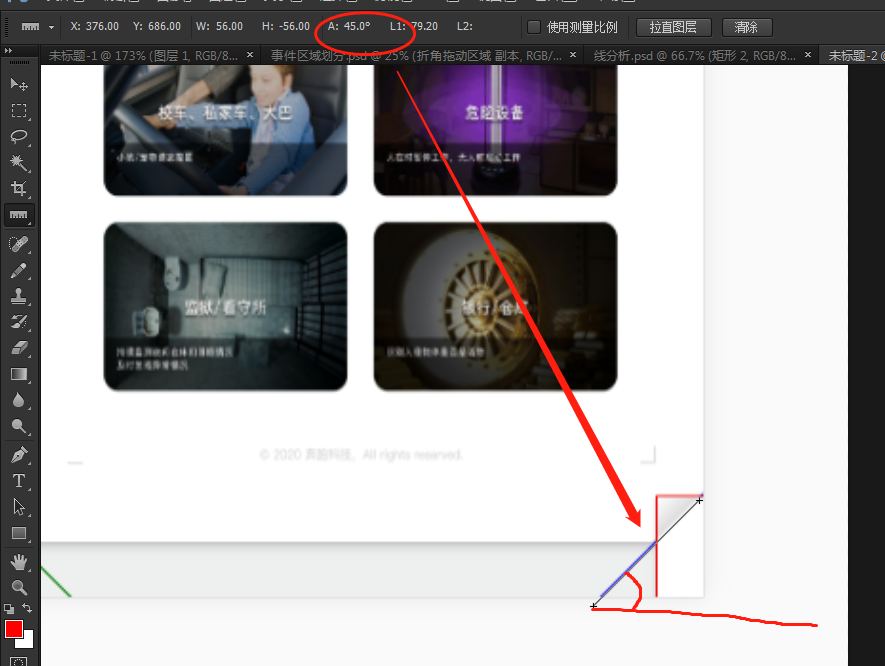
那么，为什么tr是[0,0]?



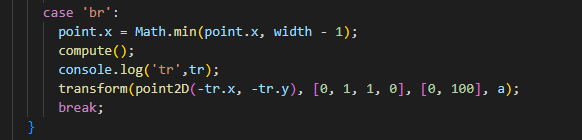
因为tr是left+top定位的，旋转点origin就是0% 0%.

# aliasingFk是折角线的角度



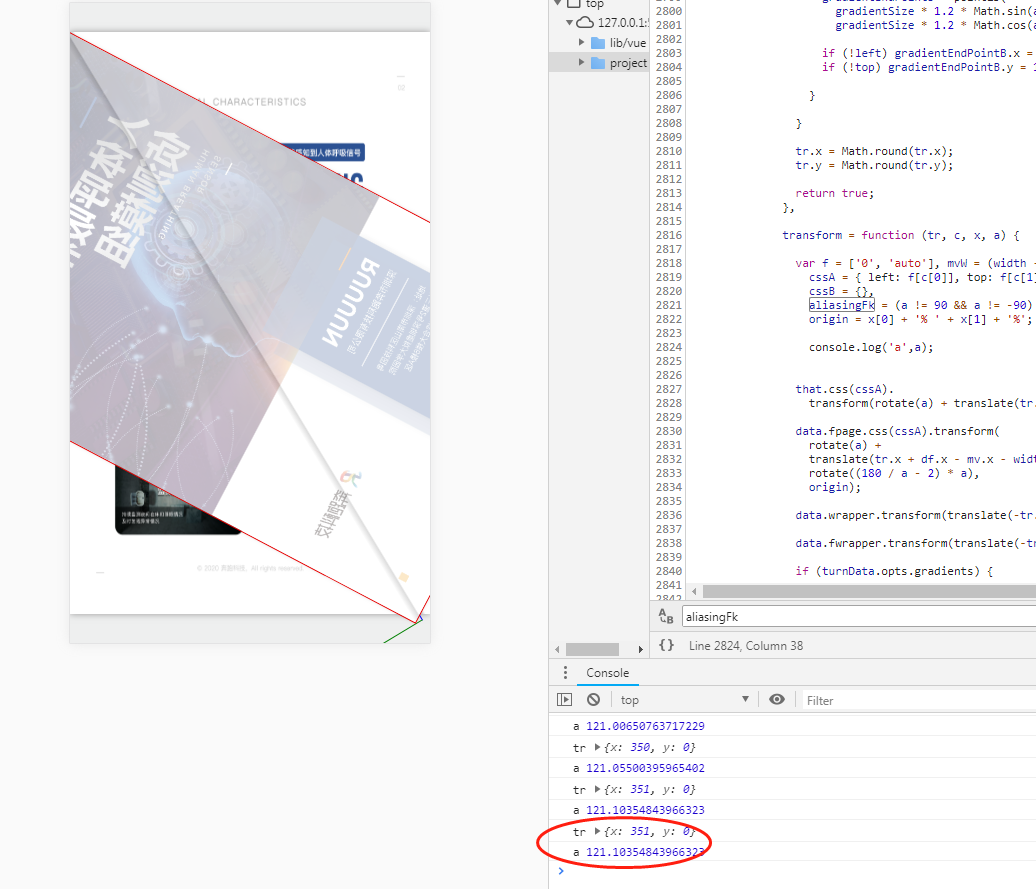


为什么br 是正a?

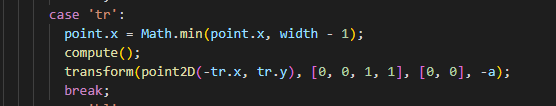


因为br折角线的角度范围就是0到180.

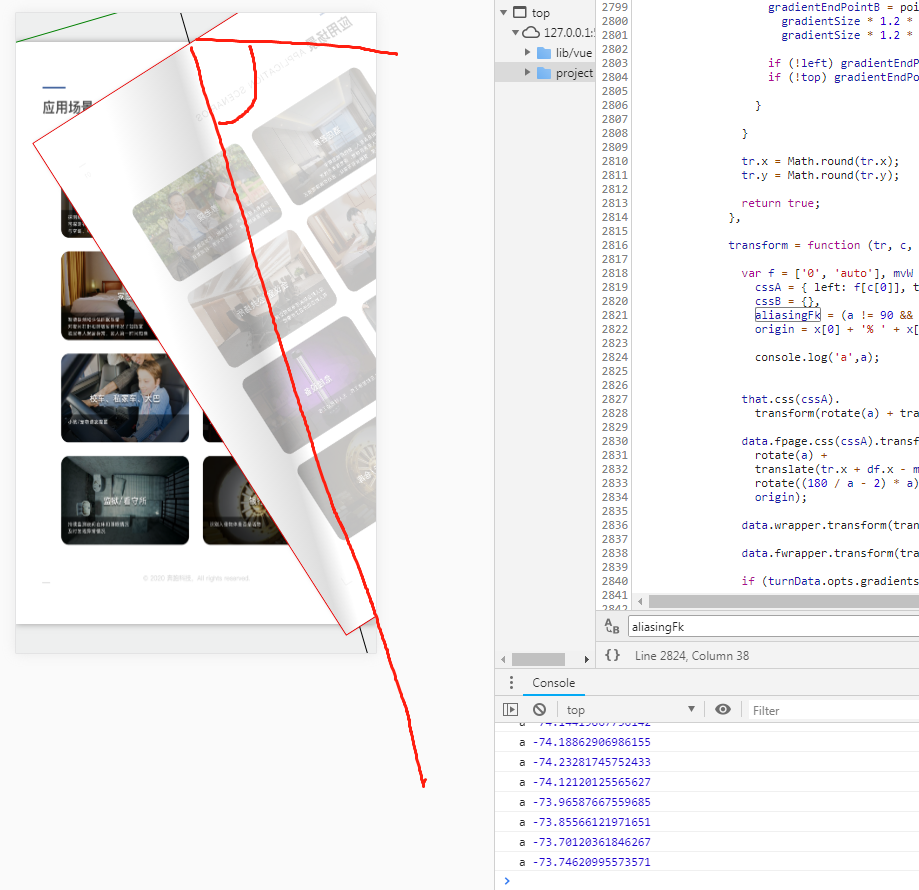
大于90度的情况如下：



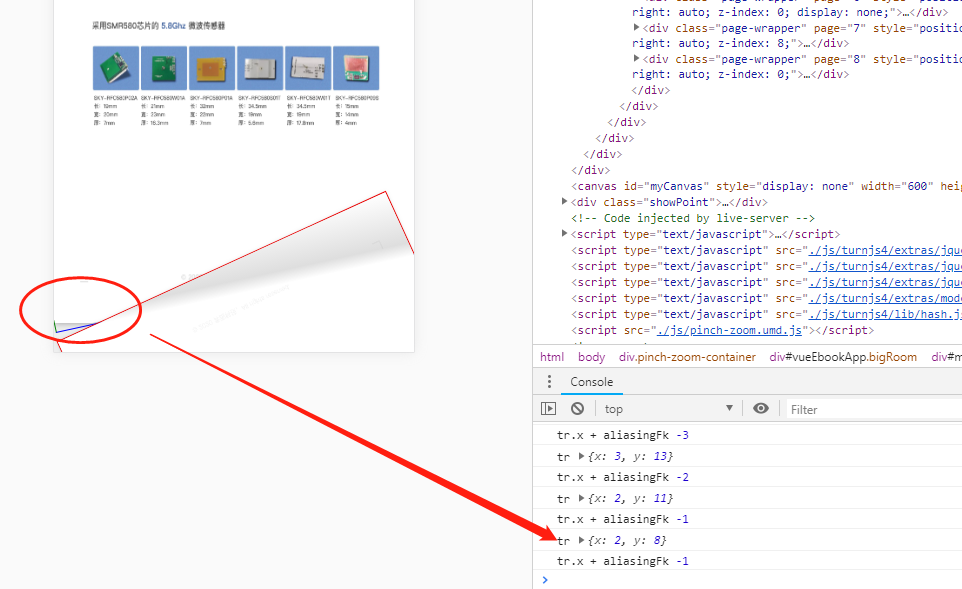
那么为什么tr 是 -a?



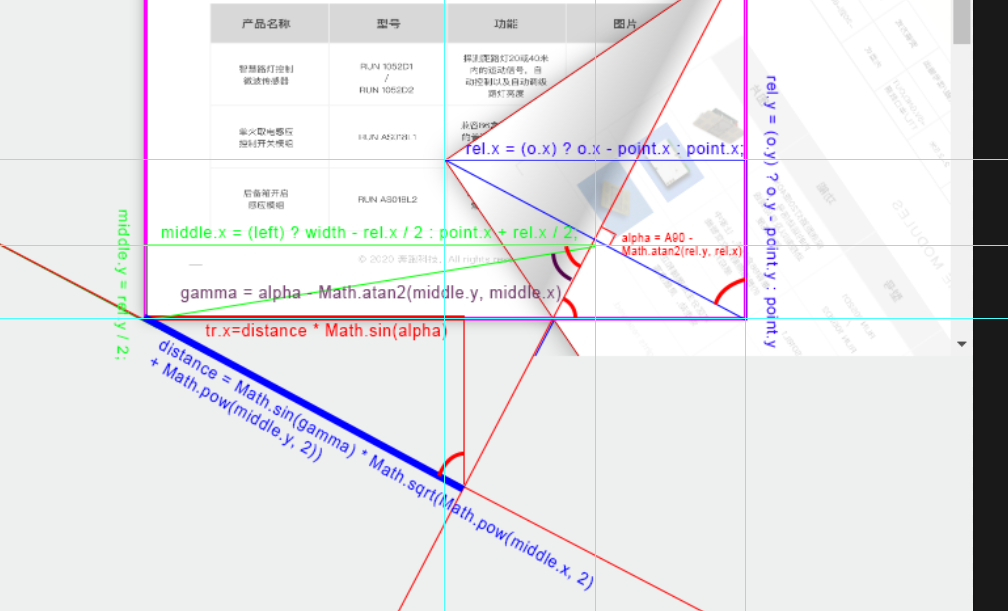
由下图得知，



# Tr的计算是关键！

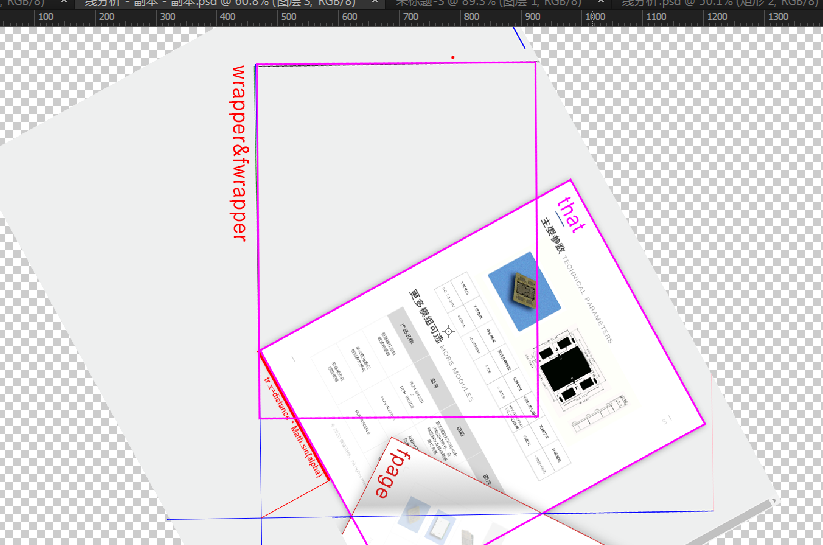


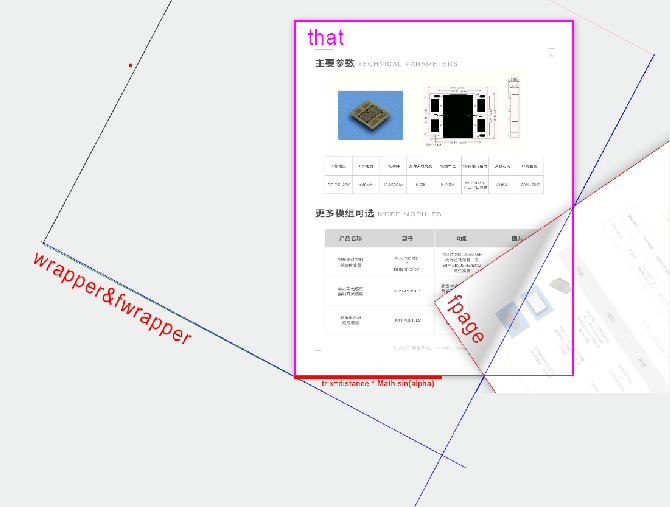
Br为例，如何计算tr?



# Transform

变形之前：

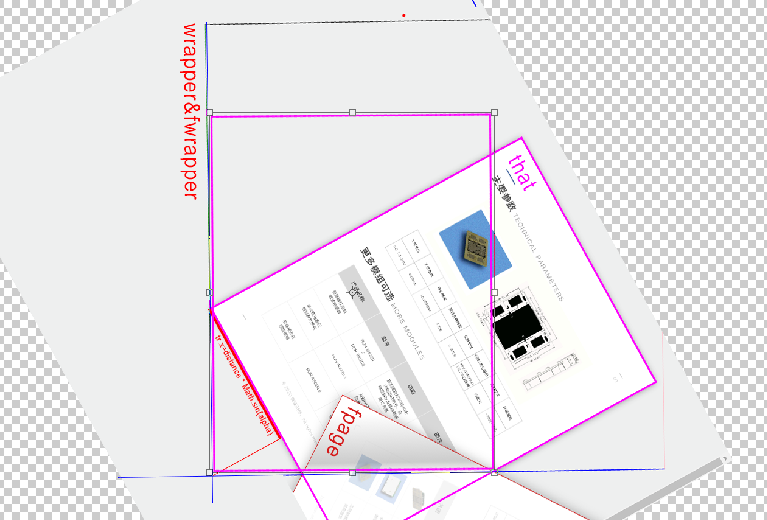




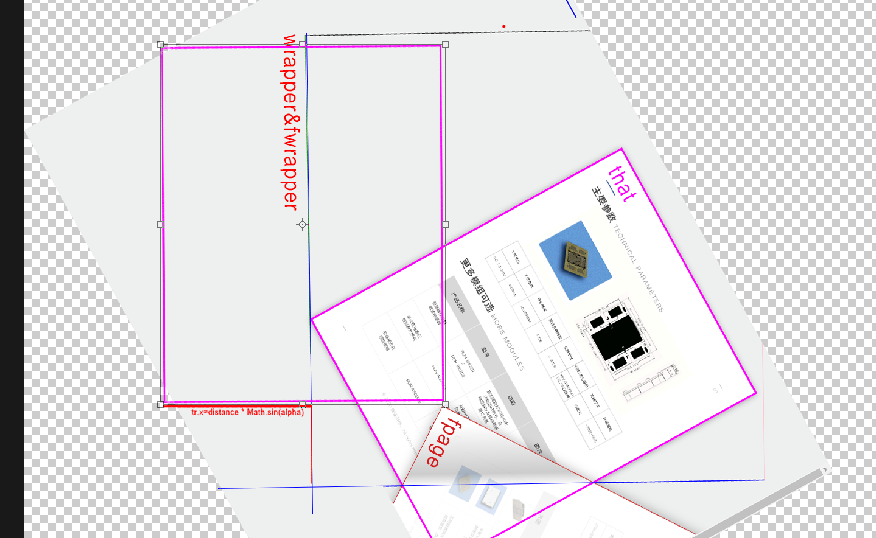
mvW = (width - h) \* x[0] / 100, mvH = (height - h) \* x[1] / 100,

## Wrapper&fwrapper

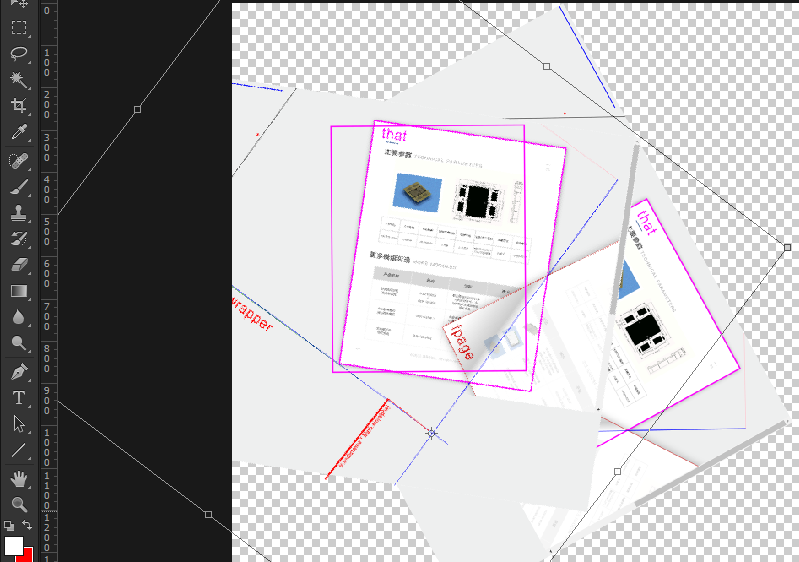
先归位：用mvH/mvW去归位：



再平移，用tr.x tr.y



再旋转：



所以代码就是：

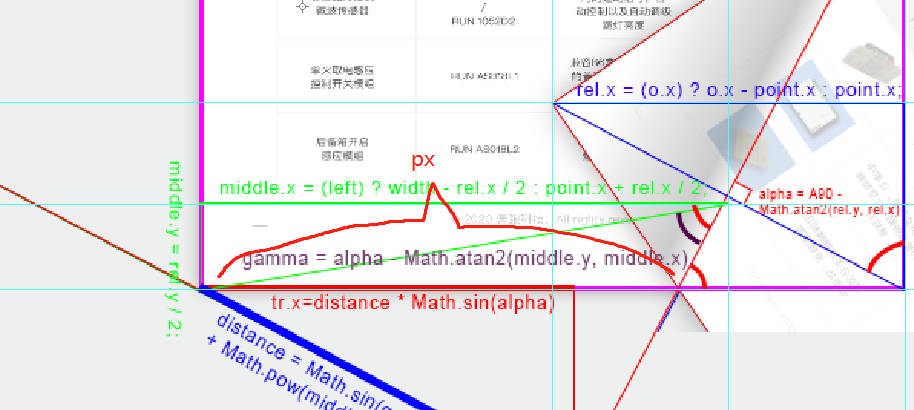
data.wrapper.transform(translate(-tr.x + mvW - aliasingFk, -tr.y + mvH, ac) + rotate(-a), origin);

aliasingFk是1px偏差。

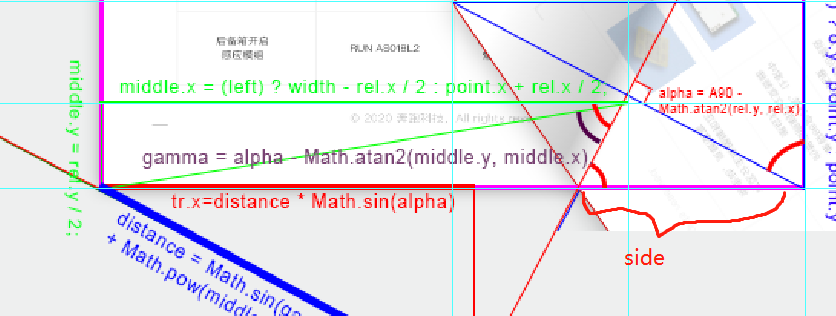
## That:就是wrapper的逆向

## 最难理解的fpage

px = Math.round(tr.y / Math.tan(alpha) + tr.x);

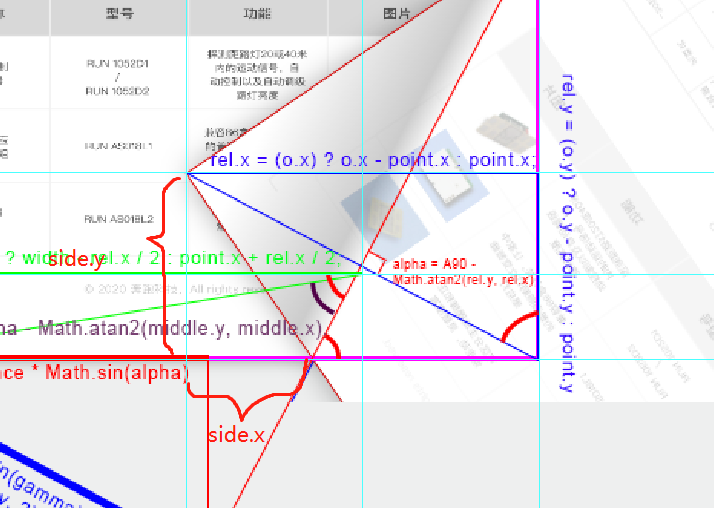


var side = width - px,



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

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



df = point2D(

Math.round((left ? side - sideX : px + sideX)),

Math.round((top) ? sideY : height - sideY));

sideX由于是负值，故df的点就是

