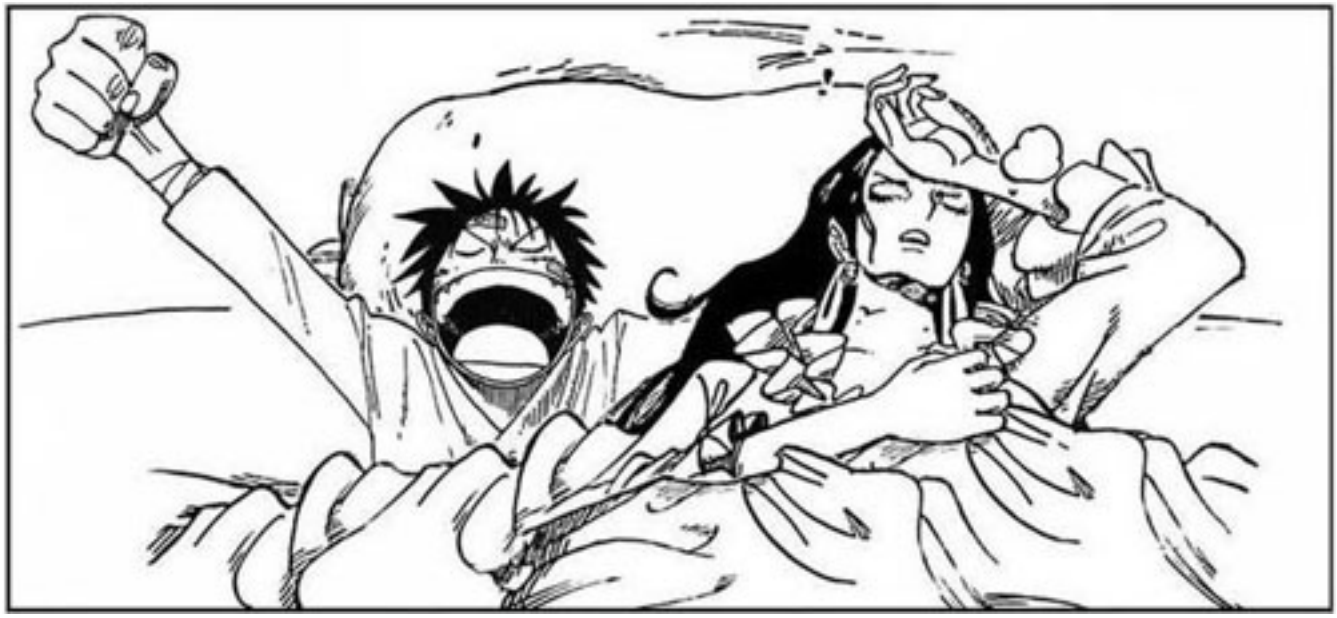
CSS3 3D transform变换

2016年6月16日 星期四 下午4:45

一、写在前面的秋裤

早在去年的去年，我就大肆介绍了2D transform相关内容。看过海贼王的都知道，带D的家伙都不是好惹的，2D我辈尚可以应付，3D的话，呵呵，估计我等早就在千里之外被其霸气震晕了~~

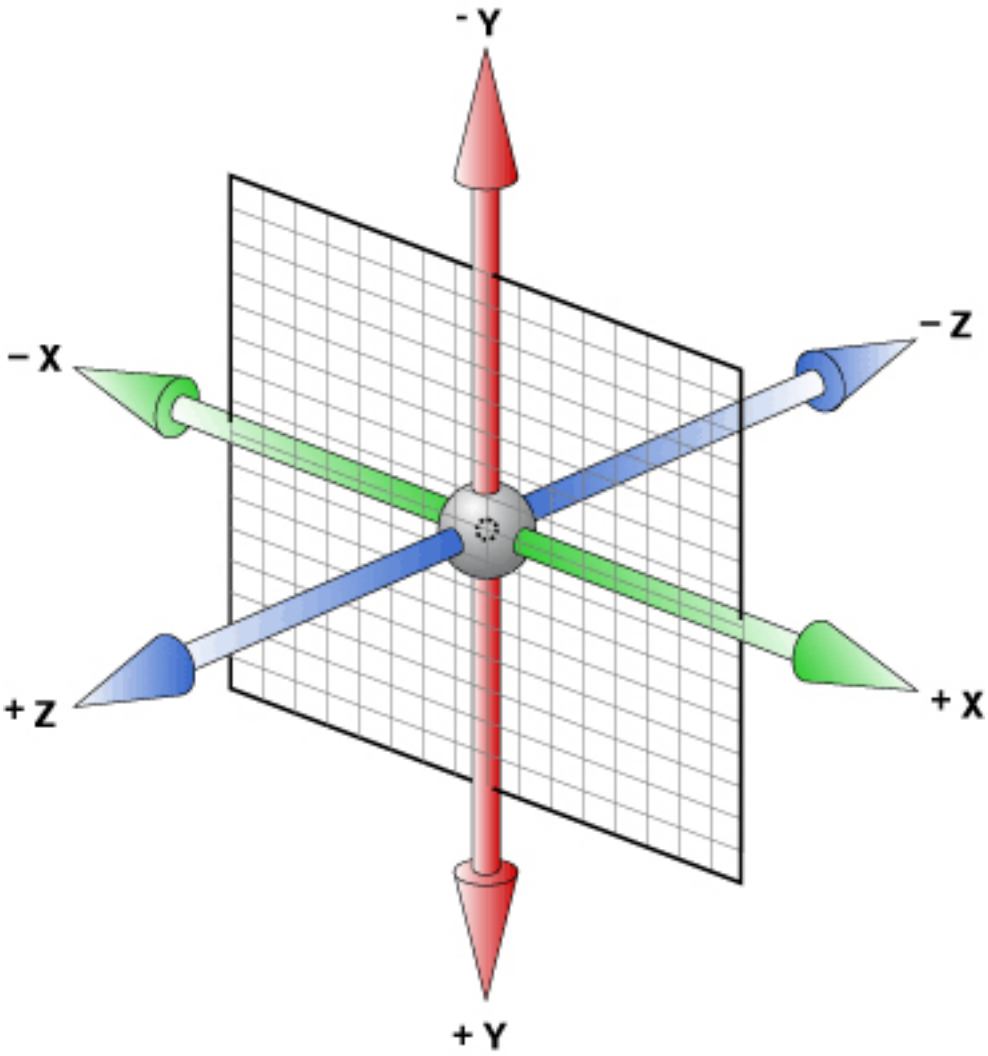
看看下图女帝的动作以及神情，就可以知道名字带D的家伙的厉害！



最近折腾iPad的一些东西，有一些3D效果的交互。有些事情，总以为是遥远的未来，谁知真正发生的时候说来就来，比如说一颗想结婚的心，又比方说在实际项目中折腾3D transform效果。

然而，虽然以前折腾过3D变换效果(webkit)，但都是依葫芦画瓢，囫囵吞枣，真正要轻松实现想要的3D效果，是需要深入理解的，于是，此时的自己苦逼了，泪奔ing……

木有办法，找资料，自己思考学习呗，当我看到下面这张基本图的时候，我的右侧的浓眉毛不由自主抖动了两下，呵，呵呵~~



这个长得像原子核一样的是什么东东？那像章鱼哥一样四处横生的箭头好吓人哦！后面怎么还有一个苍蝇拍？？

CSS好可怕，我要回去找妈妈……

想必大部分的同行应该跟我一样，没有爱因斯坦爷爷的智商，没有上镜需要把表摘掉的爸爸。因此，那些术语连篇的CSS3 3D transform介绍的资料过于耀眼，无法直视。怎么办？

好吧，佛家有云，我不入地狱谁入地狱。这里，我就从凡人们的视角说说CSS3 3D transform的一些东西，希望说的东西比较亲民，不要吓着大家。

二、首先，情感化认识

我觉得吧，要想理解一个东西，最好先有一些感性的认识。

CSS3中的3D变换效果，本质上就是我们OOXX时候各种姿势的变换，又称各种体位的变换。

虽然都是成年人，但考虑到仍有不少窝中待守的雏鸟，如果上面的解释想不过来，就想想以下这些：

1. 下图的这些人在干嘛？

跳水？NO, No, No!! 记住，他们不是在跳水，是在做3D变换！！！

2. 下图可爱baby在干嘛

？

广播体操？NO, No, No!! 记住，他不是在做操，是在做3D变换！！！

3. 来到2次元，下图这个妹子在这幅姿态称为：

卖萌？NO, No, No!! 记住，他不是在卖萌，是在做3D变换！！！

哈哈哈哈，是否意识到：在显示世界中，一切的动作（包括上面巨乳萌妹所引发的精虫上脑），都是属于3D transform变换。 因此，要学习与理解3D transform变换很简单，一句话，到现实世界找个东西映射一下即可。

三、认识的突破口：rotateX, rotateY, rotateZ

3D transform中有下面这三个方法：

- `rotateX( angle )`
- `rotateY( angle )`
- `rotateZ( angle )`

理解了这三个方法，后面更难懂的`perspective`就好下手了，可以说是突破口！

`rotate`旋转的意思，`rotateX`旋转X轴，`rotateY`旋转Y轴，`rotateZ`旋

rotate旋转的意思，rotateX旋转X轴，rotateY旋转Y轴，rotateZ旋转Z轴……

什么X轴/Y轴/Z轴，这几个词从我嘴里一出来，别说你们，我自己都晕了~~

赶快，从现实世界找对应东西理解（参照下面人的旋转）：

邹凯的体操单杠运动是`rotateX`；

蔡依林姐姐的钢管舞是`rotateY`；

旋转飞刀的特技表演是`rotateZ`。

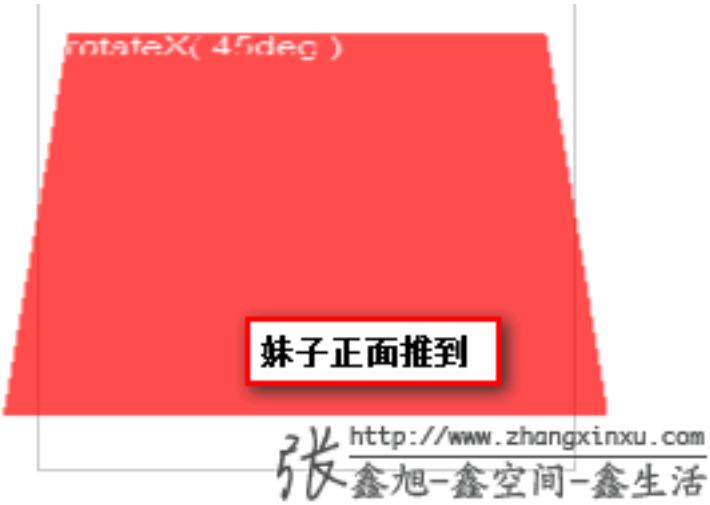
还是理解不过来？好吧，假设你是男的，以你的女朋友举例，假如原本你和她面对面站着，然后你——

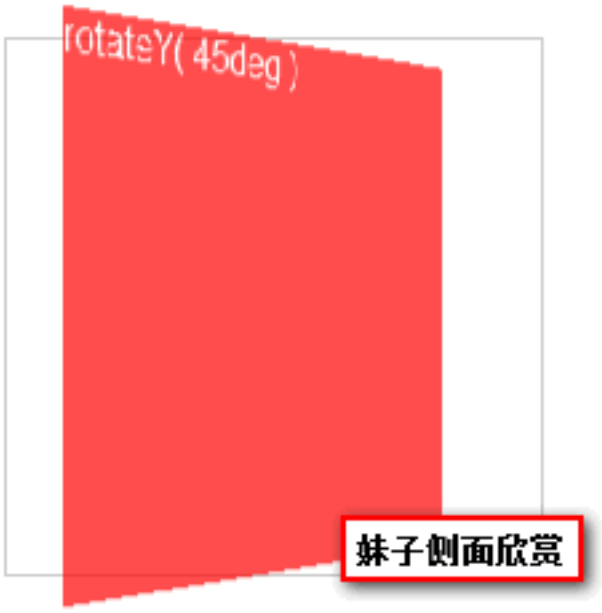
从正面将其推到就是`rotateX`；

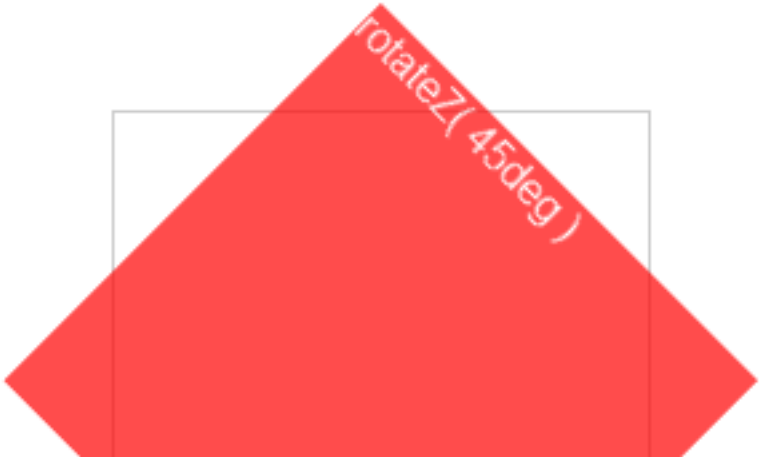
让其原地转个90度欣赏其侧面的丰满曲线就是`rotateY`；

把妹子抱到床上侧面躺着就是`rotateZ`。

于是，下面CSS世界中的简单3D效果是不是更容易理解了呢？！







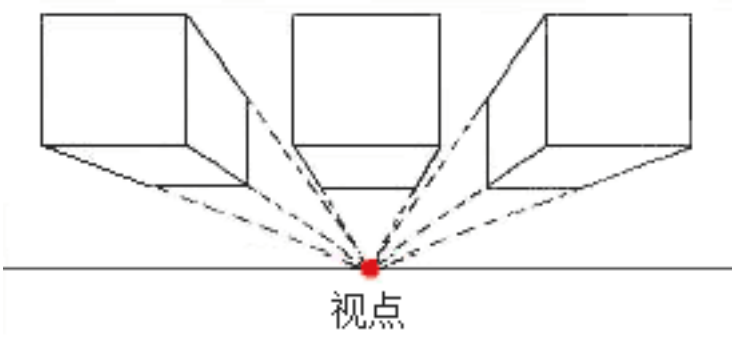
//zxx: 下面为广告~~注意不要勿点~~嘻嘻~~

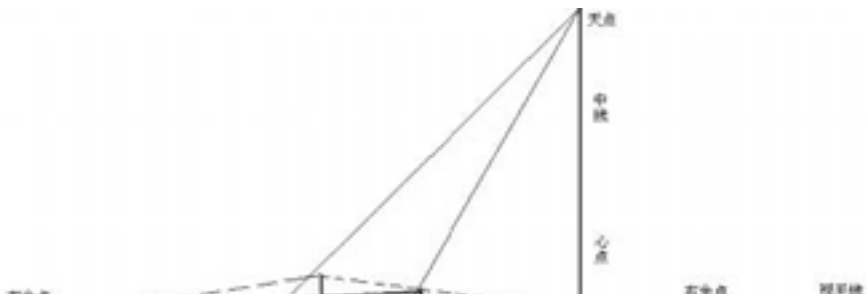
四、必不可少的perspective属性

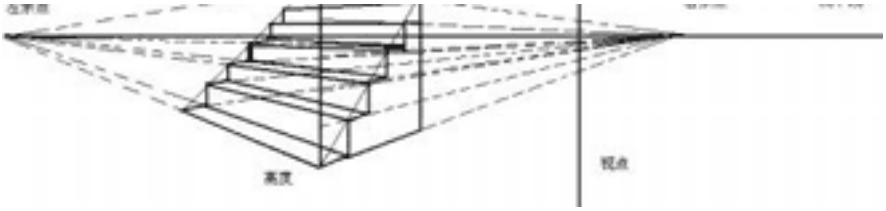
`perspective`的中文意思是：透视，视角！

`perspective`属性的存在与否决定了你所看到的是2次元的还是3次元的，也就是是`2D transform`还是`3D transform`. 这不难理解，没有透视，不成`3D`.

我们初中学美术，或者学建筑的同学肯定接触过透视的一些东西：







不过，CSS3 3D transform中的透视的透视点与上面两张示例图是不同的：CSS3 3D transform的**透视点是在浏览器的前方**！

或者这么理解吧：显示器中3D效果元素的透视点在显示器的上方（不是后面），近似就是我们眼睛所在方位！

比方说，一个1680像素宽的显示器中有张美女图片，应用了3D transform，同时，该元素或该元素父辈元素设置的`perspective`大小为2000像素。则这张美女多呈现的3D效果就跟你本人在1.2个显示器宽度的地方(1680*1.2≈2000)看到的真实效果一致！！

五、translateZ帮你寻找透视位置

如果说`rotateX`/`rotateY`/`rotateZ`可以帮助理解三维坐标，则`translateZ`则可以帮你理解透视位置。

我们都知道近大远小的道理，对于没有rotateX以及rotateY的元素，translateZ的功能就是让元素在自己的眼前或近或远。比方说，我们设置元素perspective为201像素，如下：

```
perspective: 201px;
```

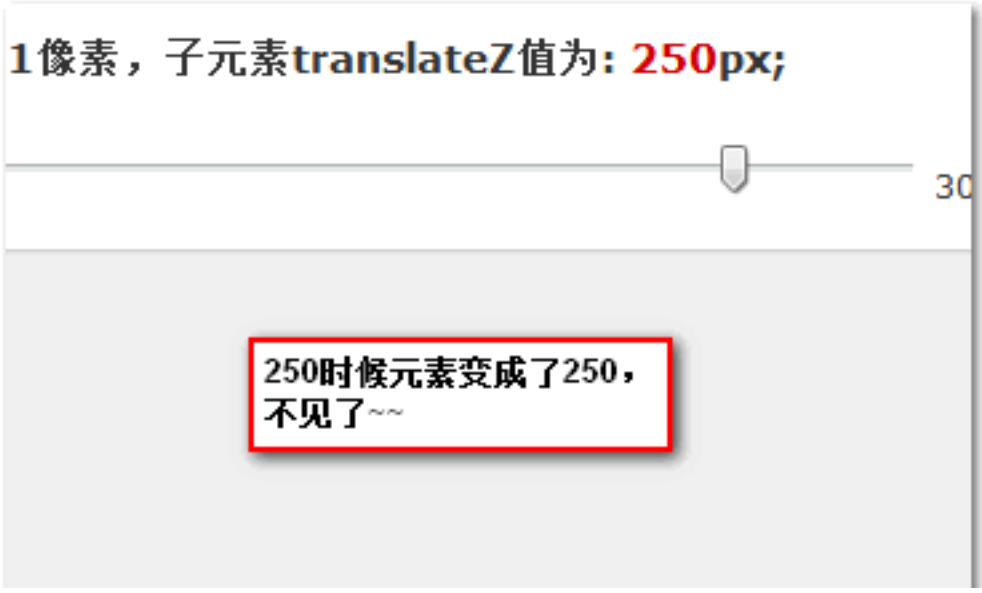
则其子元素，设置的translateZ值越小，则子元素大小越小（因为元素远去，我们眼睛看到的就会变小）；translateZ值越大，该元素也会越来越大，当translateZ值非常接近201像素，但是不超过201像素的时候（如200像素），该元素的大小就会撑满整个屏幕（如果父辈元素没有类似overflow:hidden的限制的话）。因为这个时候，子元素正好移到了你的眼睛前面，所谓“一叶蔽目，不见泰山”，就是这么回事。当translateZ值再变大，超过201像素的时候，该元素看不见了——这很好理解：我们是看不见眼睛后面的东西的！

再生动的文字描述也不如一个实例来得直观，您可以狠狠地点击这里：translateZ方法辅助理解perspective视角demo

建议Chrome浏览器下访问，可以使用range控件，演示效果更赞，如下截图：-100时候最小，200时候超级满屏（垂直方向因特殊布局限制没有显示），250的时候因为元素已经在视点之外，因此是一片空白（看不见）。


张
http://www.zhangxinxu.com
鑫旭-鑫空间-鑫生活


素，子元素translateZ值为: 200px;
300
200px时候—叶蔽目


1像素，子元素translateZ值为: 250px;
30
250时候元素变成了250，
不见了~~

六、perspective属性的两种书写

`perspective`属性有两种书写形式，一种用在舞台元素上（动画元素们的共同父辈元素）；第二种就是用在当前动画元素上，与`transform`的其他属性写在一起。如下代码示例：

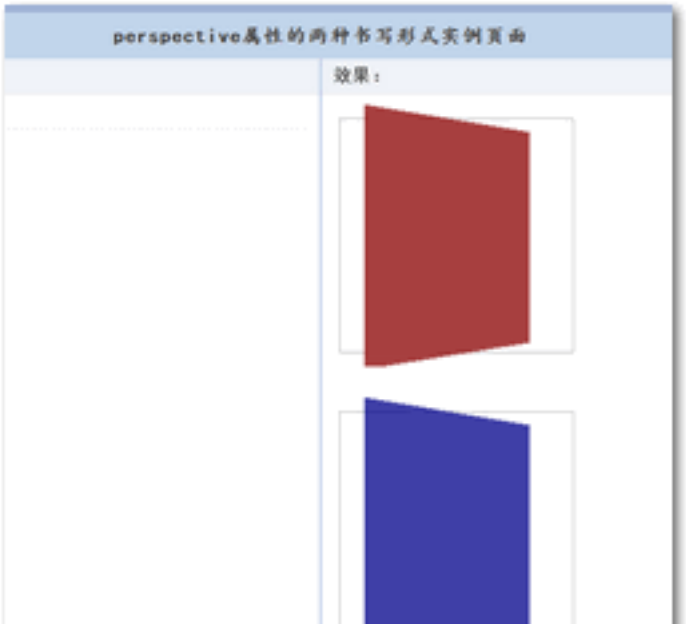
```
.stage {
    perspective: 600px;
}
```

以及：

```
#stage  .box {
    transform: perspective(600px) rotateY(45deg);
}
```

您可以狠狠地点击这里：perspective属性的两种书写demo

结果如下缩略图：

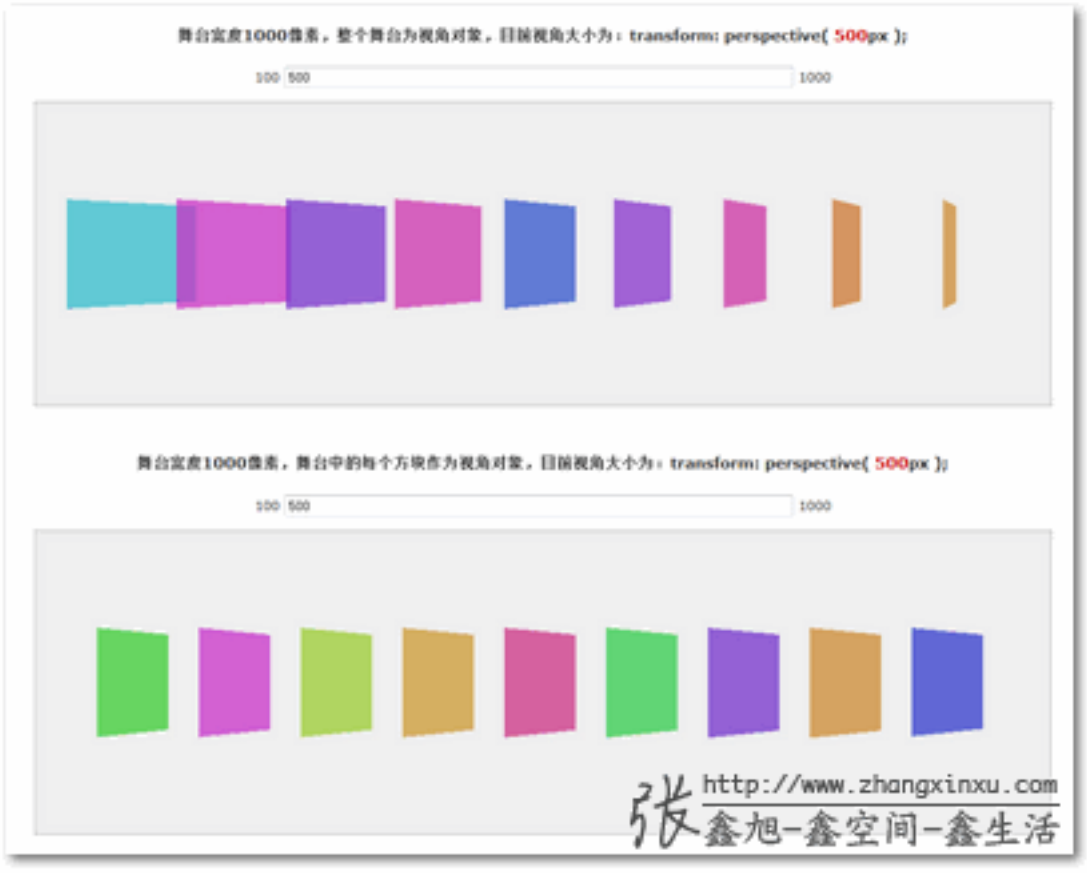


从上图我们貌似可以看到，虽然书写的形式，属性名称不一致，但是，效果貌似是一样的~~果真是这样吗？？？

实际上不然，上面的demo上下两个效果之所以会一样，是因为舞台上只有一个元素，因此，发生了巧合，其正好表现一样了。如果，如果舞台上有很多个元素，则两种书写形式的表现差异就会立马显示出来了！

您可以狠狠地点击这里：[舞台多元素下的perspective两种书写对比demo]

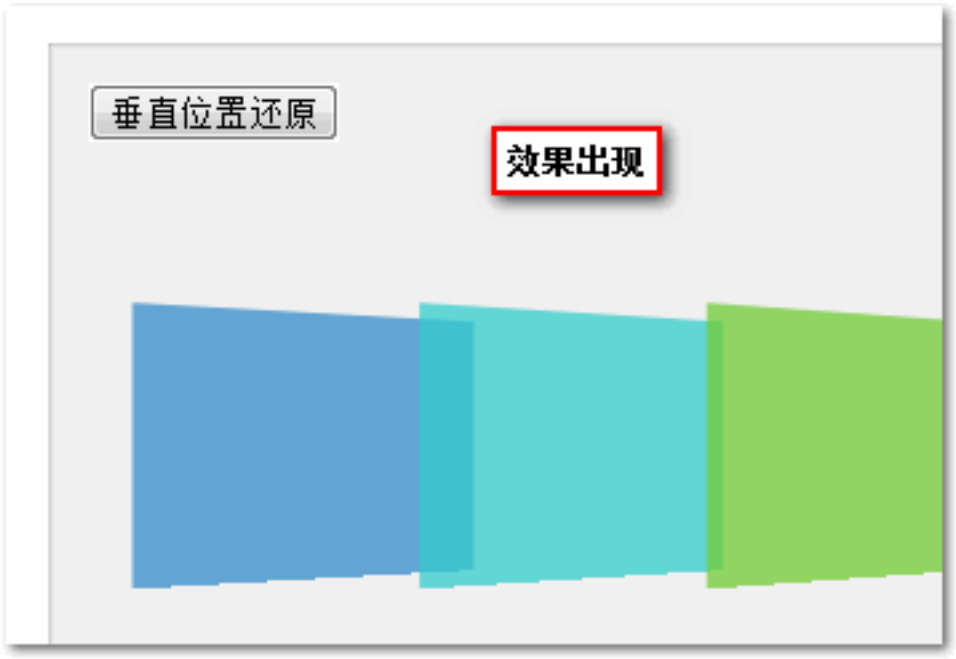
demo页面效果缩略图如下（因背景色随机，可能与下图有差异）：



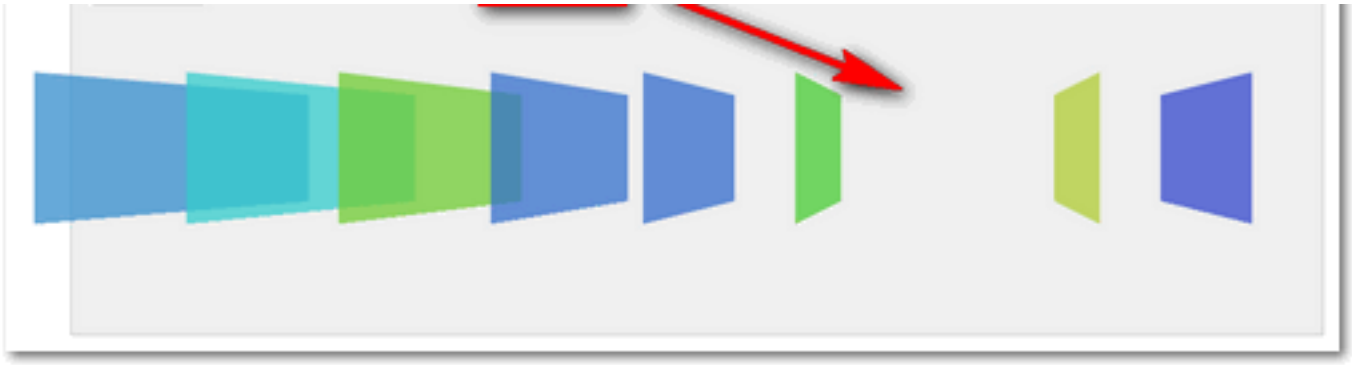
好吧，图中的效果其实不难理解。上面舞台整个作为透视元素，因此，显然，我们看到的每个子元素的形体都是不一样的；而下面，每个元素都有一个自己的视点，因此，显然，因为rotateY的角度是一样的，因此，看上去的效果也就一模一样了！

关于Chrome浏览器以及透视盲区

在Chrome浏览器下，要想看到完整的3D效果，还需要3D变换元素正好在窗体的垂直居中位置，因此，在Chrome浏览器下，生成了两个位置居中的按钮，帮助您看到想要的效果：



当我们改变第一个`range`控件值为`200`的时候，您会发现右侧第三个元素看不见了：

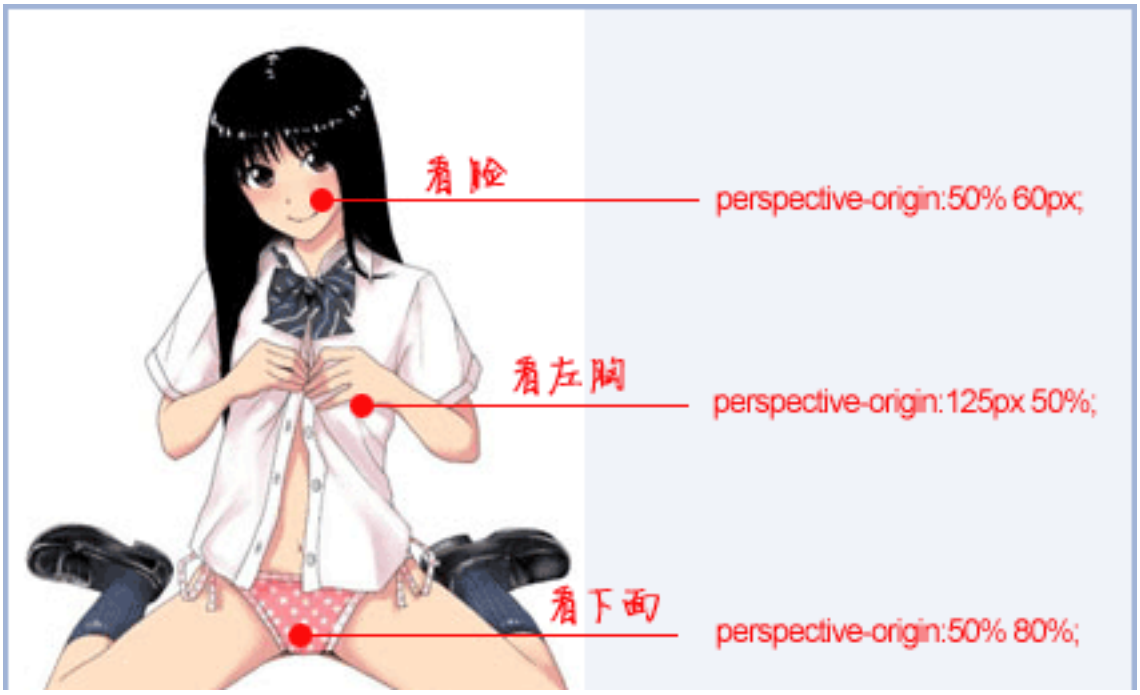


这不难理解，前面一排门，每个门都是1米，你距离门2米，显示，当所有门都开了45°角的时候，此时，距离中间门右侧的第二个门正好与你的视线平行，这个门的门面显然就什么也看不到。这就是为什么上面右侧第三个门一片空白的元素——特定的视角以及距离形成的视觉盲区。

七、理解perspective-origin

`perspective-origin`这个属性超级好理解，表示你那双色迷迷的眼睛看的位置。默认就是所看舞台或元素的中心。有时候，我们对中心的位置是不感兴趣的，希望视线放在其他一些地方。比方说

：



一图胜千言，屌丝男们这个应该都懂的。

下面为立方体的实际应用透视效果图：

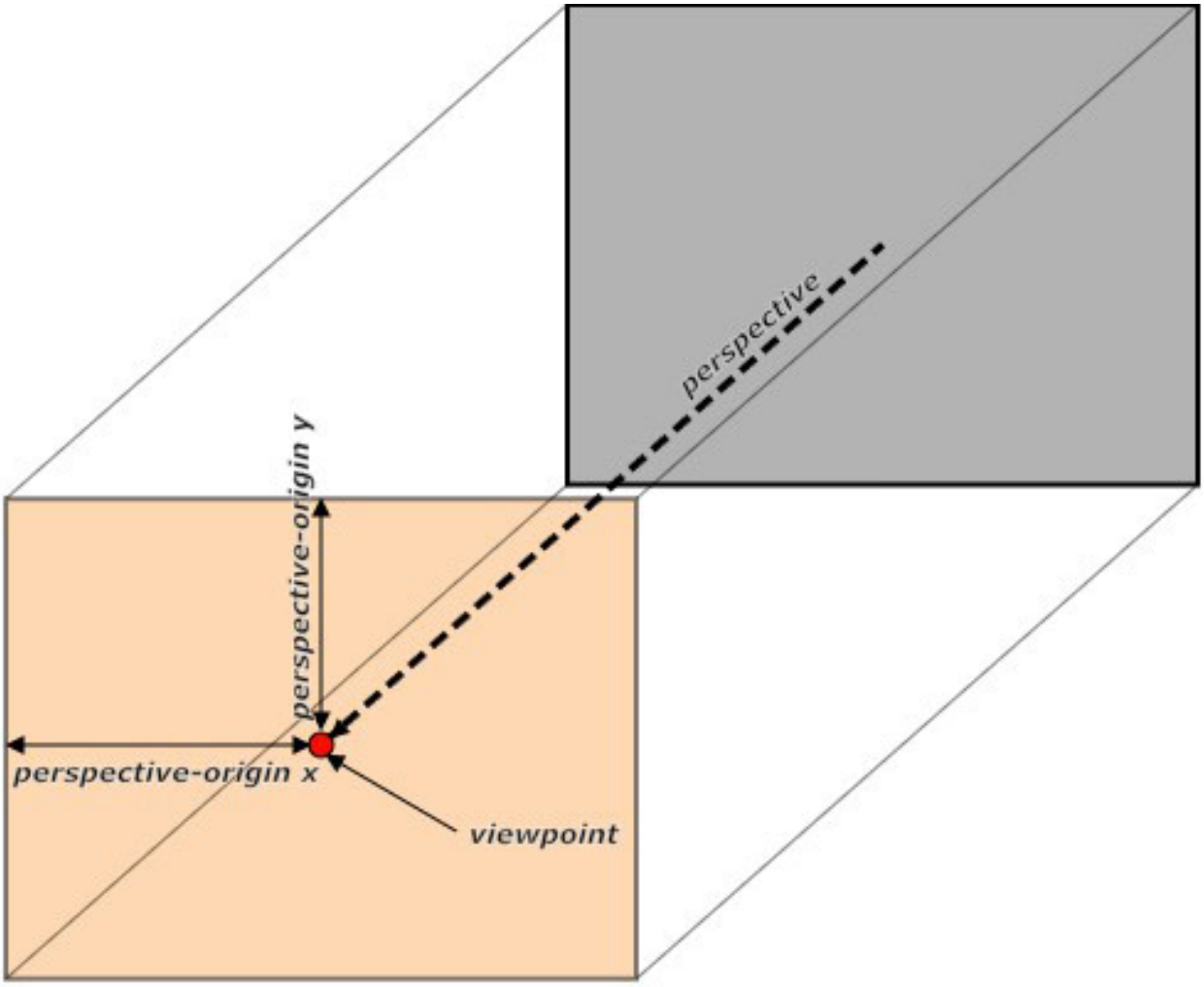
perspective-origin: 25% 75%;

语法结构

[CSS] *纯文本查看 复制代码*

```
1 perspective-origin: x-axis y-axis;
```

参数解析:
（1）.x-axis:定义视图在x轴上的位置。默认值是50%；可以是left、center、right、length和%形式。
（2）.y-axis:定义视图在y轴上的位置。默认值是50%；可以是left、center、right、length和%形式。
可能的参数值形式:left、center、right、length和%。
看了上面的介绍可能还是不够清晰，没有能在大脑中形成一个清晰的概念，那么看下面这张图片：



下面对以上图片做一下简单介绍:

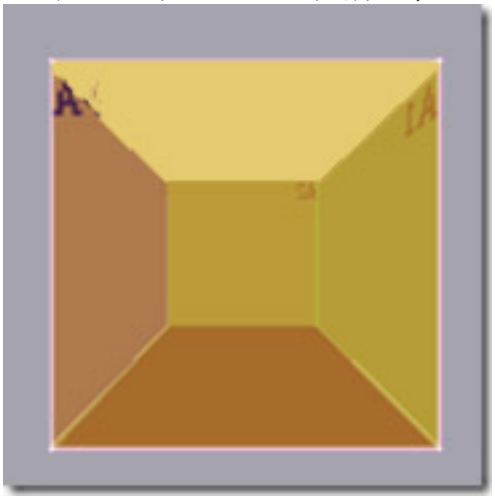
（1）.灰色部分就是我们要看的物体。

（2）.橘红色部分就是perspective属性设置的查看位置。

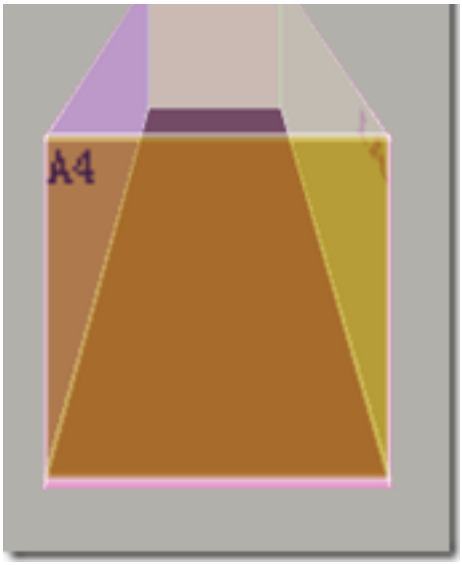
（3）.红色中心点就是默认的perspective-origin属性值所在位置。

观察物体的时候，并非都是从中心点看过去，可能要换个角度，那么就可以沿着x轴或者y轴更换一个角度。

如果上面说的还不够清楚，再来看下面的图:



上面的图片表示默认的观看角度，下面调整一下角度:



上面的图片向上提高了一下坐标，这样应该比较清晰了。

八、transform-style: preserve-3d

`transform-style`属性也是3D效果中经常使用的，其两个参数，`flat|preserve-3d`. 前者`flat`为默认值，表示平面的；后者`preserve-3d`表示3D透视。

`preserve-3d`符合我们真实世界的思维认识。比方说，你让妹子右转了45度，此时妹子脑袋左转45度想你吐舌卖萌，妹子的脸蛋应该和你是面对面平行的。

应用`transform-style: preserve-3d`声明的元素确实是这样表现的，但是，如果使用默认的`flat`值，其效果表现——恕我想象力有限——想不通：妹子的脸还是左转45度的，同时脑袋似乎移到了身体以外的地方

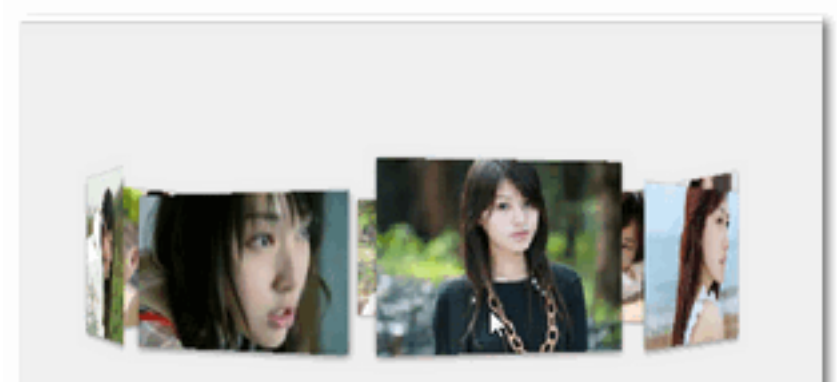
！

因此，基本上，我们想要根据现实经验实现一些3D效果的时候，`transform-style: preserve-3d`是少不了的。一般而言，该声明应用在3D变换的兄弟元素们的父元素上，也就是舞台元素。

十、实际应用-图片的旋转木马效果

您可以狠狠地点击这里：图片的旋转木马效果demo

建议在足够新版本的FireFox浏览器或Safari浏览器下观看，Chrome可能需要居中定位查看，下图为效果缩略图：



原理：

那些看上去很酷酷的CSS3 3D效果其实就颠来倒去那几个属性（本文提到的这几个），折腾来折腾去，这里这个效果显然也是如此。

首先HTML结构，如下：

```
舞台
  容器
    图片
    图片
    图片
    ...
```

对于舞台，很简单，加个视距，比方说800像素：

```
perspective: 800px;
```

对于容器，很简单，加个3D视图声明，如下：

```
transform-style: preserve-3d;
```

然后就是图片们了。为了不至于产生类似DNA的螺旋状效果，我们让所有图片`position:absolute`，公用同一个中心点。

显然，图片旋转木马是类似钢管舞旋转的运动，因此，我们关心的是`rotateY`的大小。

因为要正好绕成一个圈，因此，图片`rotateY`值正好`0~360`等分，于是，如果有9张图片，则每个图片的旋转角度累加`40(360 / 9 = 40)`度即可。因此有：

```
img:nth-child(1) { transform: rotateY(  0deg ); }
img:nth-child(2) { transform: rotateY(  40deg ); }
img:nth-child(3) { transform: rotateY(  80deg ); }
```

```
img:nth-child(4) { transform: rotateY( 120deg ); }
img:nth-child(5) { transform: rotateY( 160deg ); }
img:nth-child(6) { transform: rotateY( 200deg ); }
img:nth-child(7) { transform: rotateY( 240deg ); }
img:nth-child(8) { transform: rotateY( 280deg ); }
img:nth-child(9) { transform: rotateY( 320deg ); }
```

这样就好了吗？

No, No, No!!!

想想看那，虽然9个绝色美女每个人的方位不一样，但都站在同一个点上，早就挤作一团，A罩都挤成C了，显然是不行的（见下图只设置rotateY）！我们需要拉开空间~~

如何拉开空间，很简单。

想想看那：9个美女，分别面朝东南西北共9个不同方位，她们只要每个人向前走个4~5步，美女们之间的空间不久拉开了，呈现圆形了！想象一下夜空中，礼花绽开的场景~~

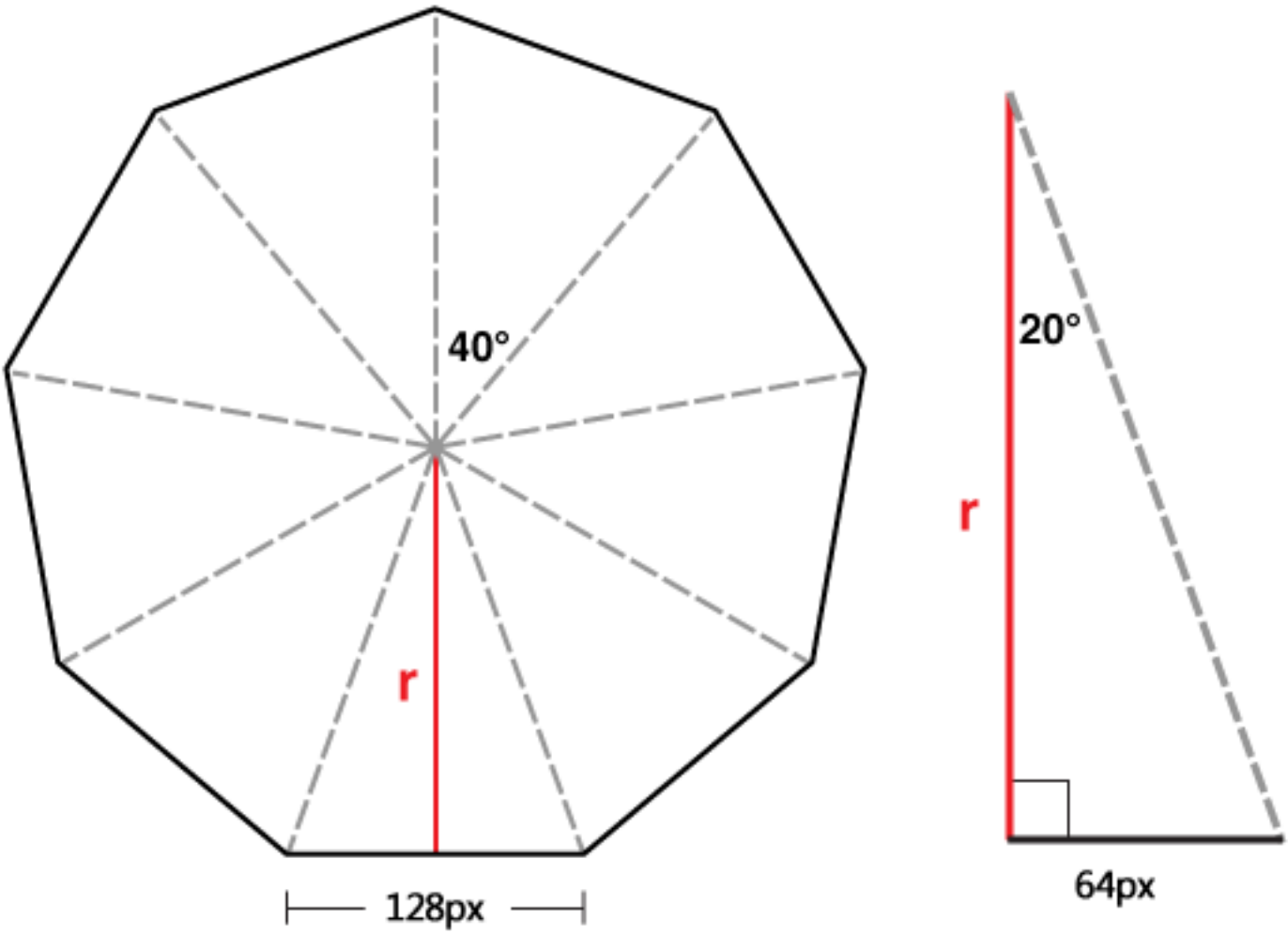
这里的向前走4~5步，聪明的人应该已经知道了，就是本文提到的`translateZ`，当`translateZ`为正值的时候，元素会向其面对的方向走去；如果元素无旋转，就会朝显示器走来！！

现在只剩下一个问题了，美女们要向前走多远呢？？

现在只剩下一个问题了，关于的要问的是多远呢？？

这个距离是有计算公式滴！

拿本demo距离，每张美女图片的宽度是128像素，因此，有如下理想方位效果图：

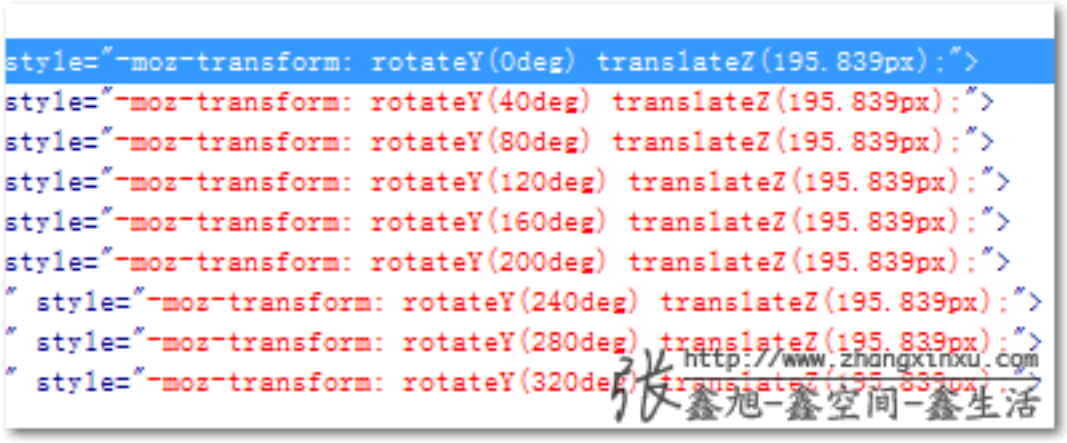


上图中红色标注的`r`就是的demo页面中图片要`translateZ`的理想值（该值可以让所有图片无缝围成一个圆）！

`r`的计算很简单，有初中数学水平的人应该都会：

r = 64 / Math.tan(20 / 180 * Math.PI) ≈ 175.8

demo页面为了好看，图片之间留了点间距，使用的`translateZ`的值为`175.8 + 20 = 195.8`.



最后的最后，要让木马旋转起来，只要让容器每次旋转40度就可以了。

节省篇幅，具体的JavaScript操作代码就不展示了，您有兴趣可以查看demo页面源代码。

理解了旋转木马3D效果实现原理，基本上，其他些3D效果可以轻松驾驭了，因此，本效果还是值得你花功夫看看滴~~

十一、好吧，结语

理论上，现实世界，及3次元世界中的各种有规律的运动效果都可以使用CSS3 transform 3D方法实现。文章最后的旋转木马效果可以说是各类千奇百怪效果中的沧海一粟~~其他各类有的没有的效果就靠你的大脑就构想了。至于实现嘛，理解了，也就都是小菜。但是，要是不理解，纯粹从网上copy些效果代码，那永远就是copy的命咯！

文章篇幅已经很长了，我的指头也敲出老茧来了，就不再啰嗦什么了。希望本文的嗑叨、卖弄、折腾能够让您学习CSS3 3D transform变换的相关东西更加轻松点！

行文仓促，文中有错误在所难免，欢迎诸位指正。最后，感谢阅读，共同进步！