动效开发 2：聊一聊 3D

我们在前一小节的案例中制作了一个立方体，其实就已经接触到了 3D。

所有东西一跟 3D 扯上关系，复杂指数都是噌噌噌往上走。不过也正常，毕竟多了一个维度，要有三维应有的尊严。

3D Transforms 要怎么写？能写翻牌效果吗？能写翻书效果吗？能写出立体书的效果吗？往下看，答案都在这里面。

很多时候，仅仅将元素进行二维层面的变换显然不是人类的终点，毕竟十二维空间都可能不是极限（视频: 从一维空间到十二维空间 (http://v.youku.com/v_show/id_XNjA0MjU5NzA4.html?from=s1.8-1-1.2)）。

Intro to 3D Transforms (https://desandro.github.io/3dtransforms/) 的作者 David DeSandro 说，现在可是 21 世纪，可我们竟然还在跟三十年前的二维空间界面扯皮。所幸 2011 年，我们有了 CSS3，我们还有了 3D Transforms，真是一个值得奔走相告的大事件。

3D 变换相较 2D 变换，坐标系中多了 Z 轴，也就意味着物体除了上下左右，还可以前后移动。而 rotate 在 2D 中的旋转方式，在 3D 中与 rotateZ 相当。

那么，单纯地将 transform 中的参数扩展出 Z 维度，就能实现 3D 效果了吗？我看见 CSS3 笑了。

perspective 概念理解

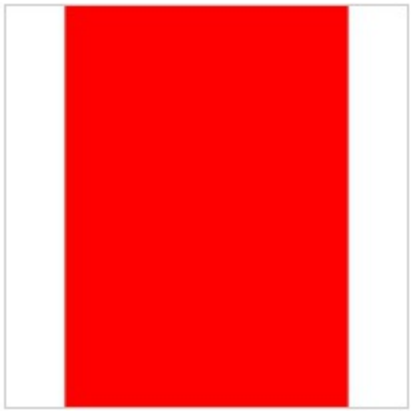
什么是 perspective？词典中翻译为观点、远景、透视图。这是一个非常抽象的概念，需要一点空间想象力。

我们先抛开这个概念，尝试使用刚才说到的知识点进行翻牌（咦）效果的尝试，聪明的你一定分分钟码出来：

```
<div class="card">
  <!-- 卡牌正面 -->
  <figure class="card-front">1</figure>
  <!-- 卡牌反面 -->
  <figure class="card-back">2</figure>
</div>
```

```
.card-front {
  background: red;
}
.card-back {
  background: blue;
  transform: rotateY( 180deg );
}
/* 翻牌动作 */
.card.flipped {
  transform: rotateY( 180deg );
}
```

但是放浏览器里一看，这不对呀，为什么用 3D 的代码写出了 2D 的效果?



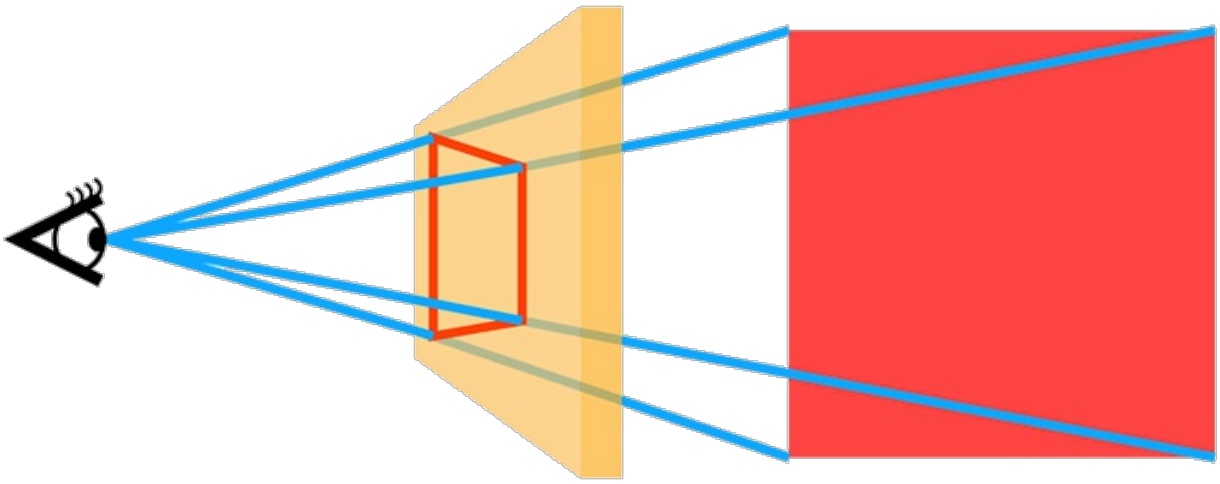
这个时候有请我们的 perspective 透视君。

学过素描的人一定对透视的概念不陌生，透视是保证素描写生真实合理的基础。

> 视频：[透视学之一点透视法 (https://www.bilibili.com/video/av14392523/)](https://www.bilibili.com/video/av14392523/)

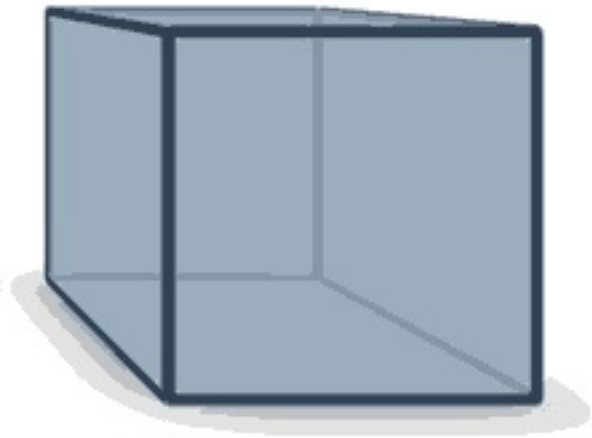
CSS3 中的 perspective 在这样一个体系里就代表着元素与观者之间的距离，形象点说，就是元素 3D 效果的强度。CSS3 中的 3D 效果消失点固定，变化的是观者与元素之间的距离。不过 perspective 数值与 3D 效果强度是成反比的，数值越大，元素的 3D 效果越不明显 —— 2000px 的视点意味着你看的是远方的物体，而 100px 则意味着这个物体就在你眼前。

这里有幅图或许能帮助我们想象 3D 效果强度这个概念。

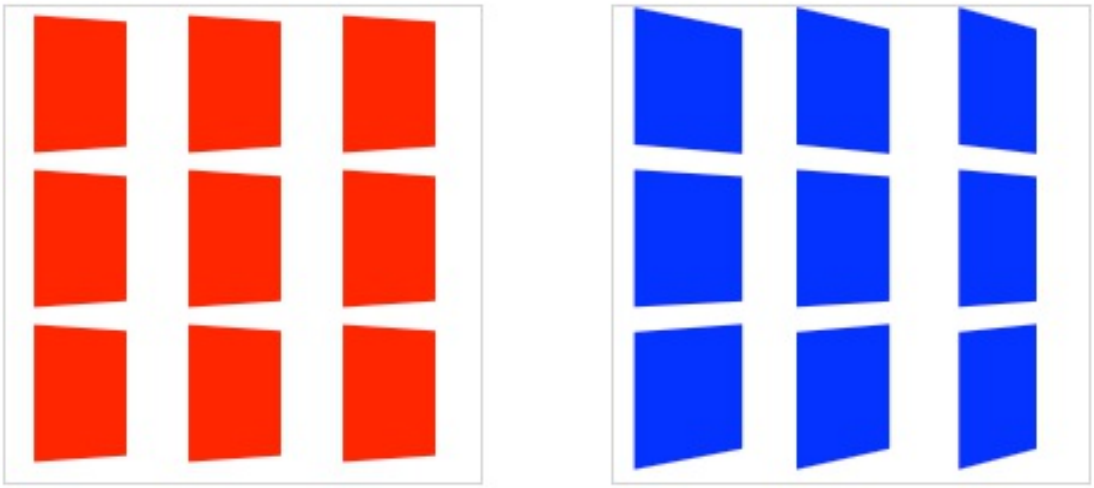


（图片来源：[维基百科](https://en.wikipedia.org/wiki/Perspective_%28graphical%29)）

如果还是不懂，还有一个办法，就是在浏览器中边调整 `perspective` 数值边观察 3D 效果。



消失点

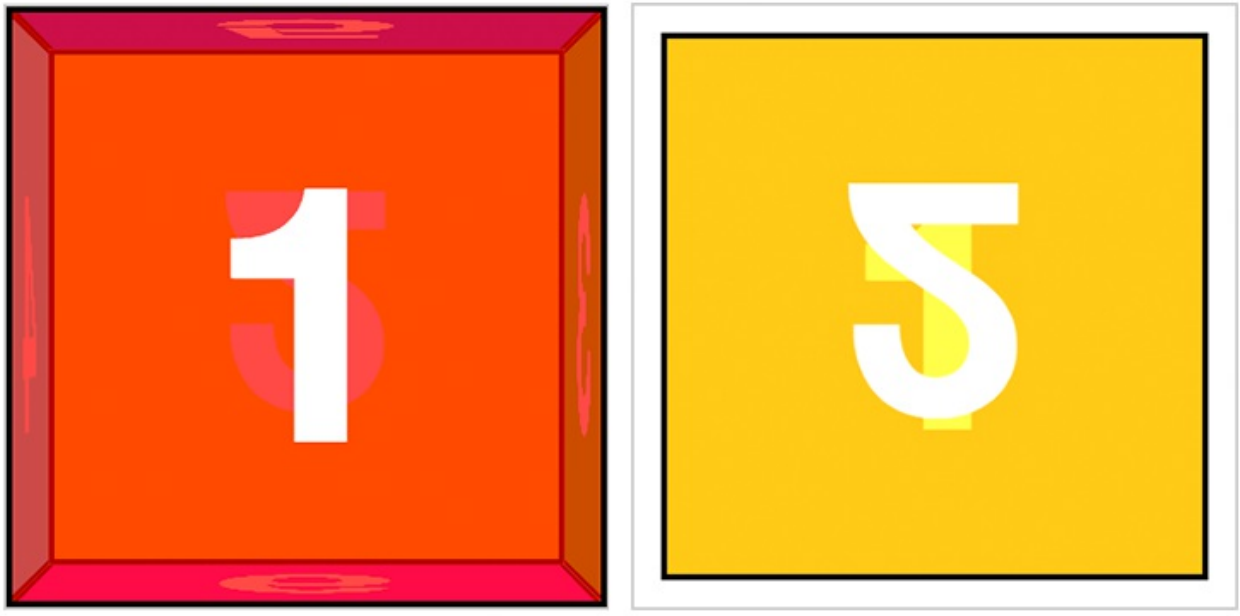


（图片来源：Intro to CSS 3D transforms - Perspective (https://desandro.github.io/3dtransforms/docs/perspective.h

左图与右图的元素均绕 Y 轴旋转了 45°，但差别很明显，右图更容易让人想到一个画面中集体开启的窗户。左图的问题就在于，每个元素的消失点各自为政，都在元素的中心点位置，而右图的消失点则统一在实线方框的中心位置。实现方法就是将元素的 perspective 设置转移至元素父容器上。

明眼人会说，这样子可以画个正方体出来了耶。我看见 CSS3 又笑了。

建立三维空间体系



现实总是乳齿残酷~

有了 perspective 属性，我们顶多是一群会在纸上画素描的家伙，要想徒手造模型，还是太嫩。就拿刚才的翻牌效果来说，如果你翻滚 card 父容器，无论怎么翻，能看到的只有正面的卡片，因为现在的体系就是一张素描绘画，你拿着再逼真的素描画翻到背面，也是看不到真实物体的背面的对吧。超越平面 3D 的关隘就在于 transform-style: preserve-3d 的属性设置，默认值为 flat，即“素描作品”。这个属性的设置旨在告诉子元素需要遵循怎样的空间体系规则。这个属性不能继承，因此只要有子元素需要设置空间体系规则，就得在父元素声明这个属性。

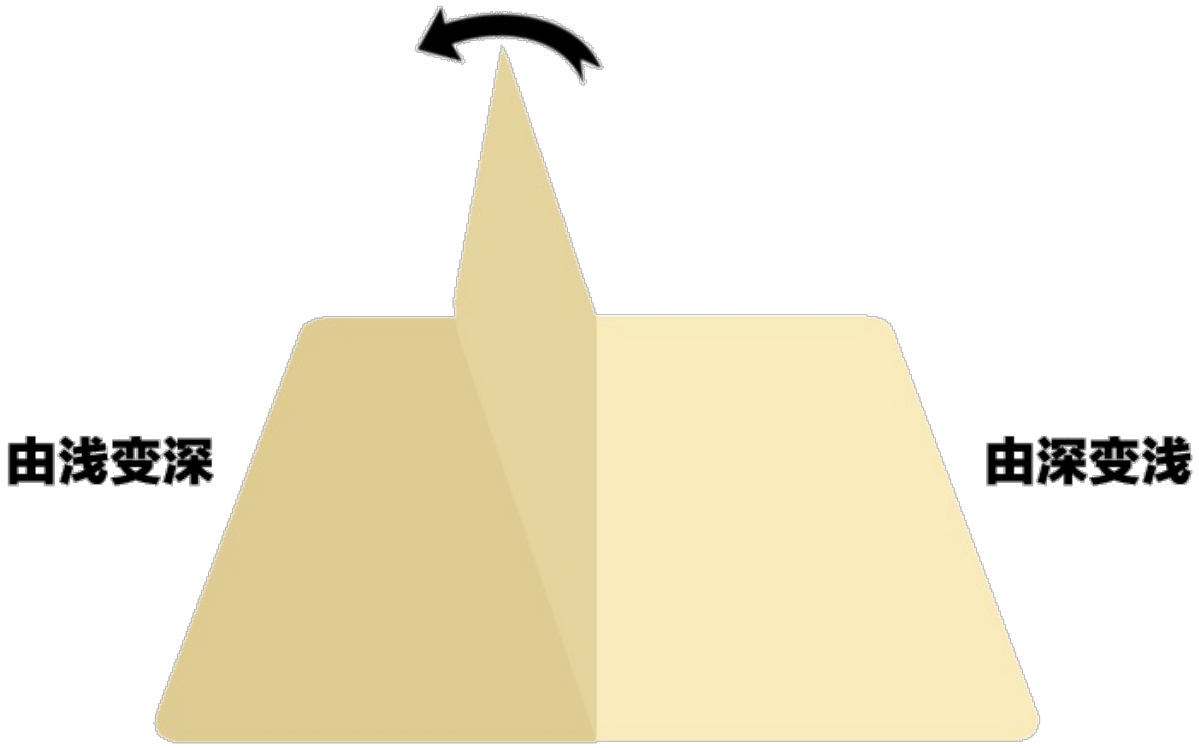
有了浏览器为我们处理空间体系规则，可以省不少事，不需要你担心层级问题，不需要你操心哪个元素转到哪里要消失，哪个元素转到哪里要出现。嗯，笔者从没自己这么干过，从没。

从翻牌到翻书

翻牌那是皇帝干的事儿，我们文化人得翻书。刚才的翻牌都是在方块的中部为轴进行的变换，我们把变换原点 transform-origin 一换，就变成书页在翻了。

一本合上的书正常来说是在 Y 轴右侧，每一页都包含两面，也就是说一本书是由若干个翻页效果组合而成，每一页的变换原点在元素左侧。由此可以在翻牌的基础上迅速整出一个翻书 DEMO（猛戳 查看翻书 DEMO (http://lyxuncle.github.io/pageturning/demo/demo.html)）。

阴影的使用能让翻书效果变得更真实。



（猛戳 查看 DEMO（带阴影） (http://lyxuncle.github.io/pageturning/demo/demo2.html)）

3D 动画之 Hard Level：立体书

立体书在外国叫 Pop-Up Book，满满的 “Surprise!” 感。这种超越传统平面书籍的阅读模式常被用于儿童书籍。

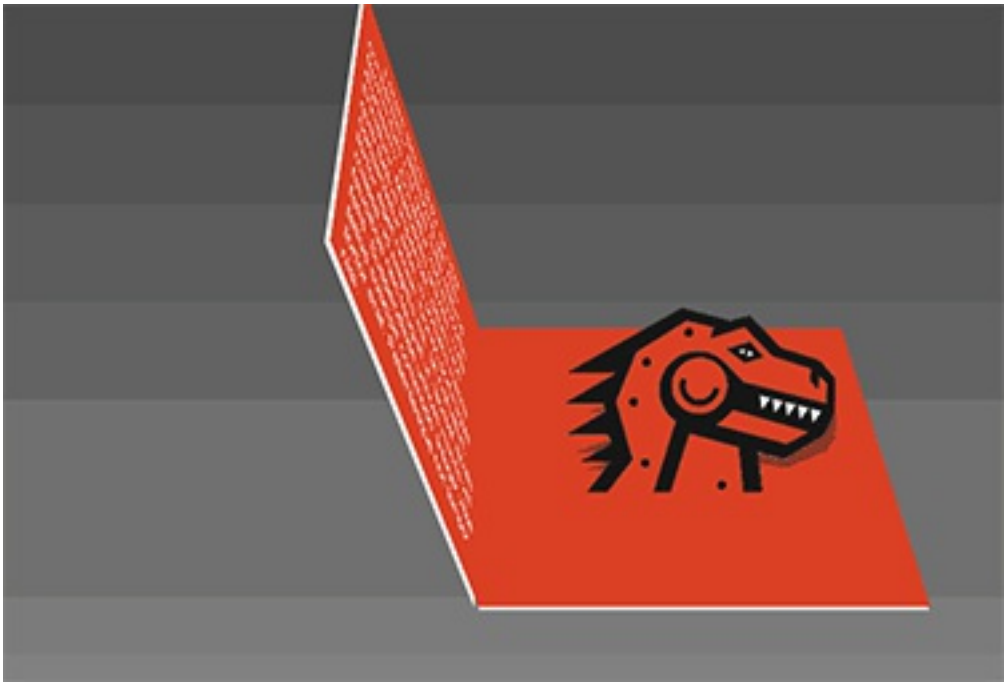
（图片来源：[A Guided Tour of THE MEL BIRNKRANT COLLECTION (http://melbirnkrant.com/collection/page48.html)](http://melbirnkrant.com/collection/page48.html)）

要用 CSS3 实现这种效果，想想还有点小激动。

首先建立一个立体书规则：

- 书开，元素起
- 元素竖起速度小于等于书页开启速度
- 元素折叠后不可露出书边
- 元素层叠关系不可反自然

剩下的事也就水到渠成，无非是在每一页建立 3D 体系、立体元素从 `rotateY(90deg)` 转换到 `rotateY(0deg)` 的事儿。



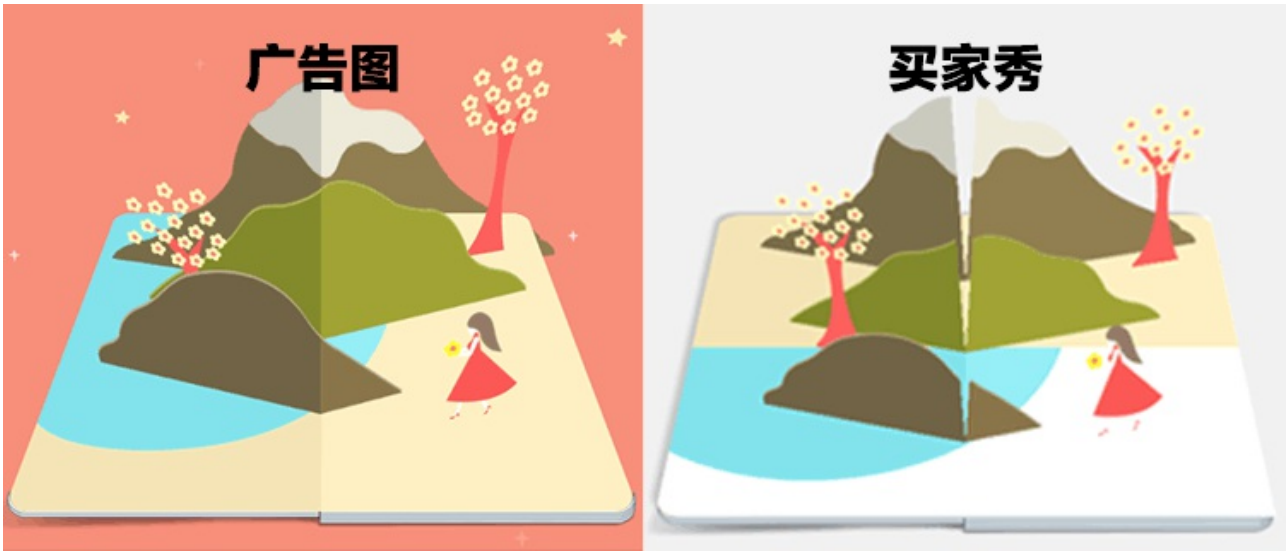
（Mozzilla 的小 DEMO (http://www.html5tricks.com/demo/css3-3d-book/index.html)）

笔者曾做过一个丧心病狂的立体书触屏页，由于立体书左右两页互相关联的特性，翻牌的方式不太适合用在这里，这里使用的是另一种较为麻烦的方式 —— 不像翻牌方式中的前后两页捆绑，这里的书页左右两页属于一个 3D 体系，通过 translateZ 值的变换控制层级关系，因为在 3D 体系里，z-index 已被抛弃。

猛戳进入 麦芒推广页 (http://jdc.jd.com/fd/pp/maimang/index.html) 体验 3D 立体书效果。

终端支持

由于截至目前为止，CSS3 的 3D 功能还止于炫技的阶段，安卓机与 iOS 的支持效果存在差异且难以调和，从上面那个案例中肉眼可见的坑就能看出，因此除了简单的 3D 转换，不建议在生产项目中大面积使用 3D 深层功能。



3D 与硬件加速

坊间流传这这样一个传说：一旦使用 3D 属性，就能触发设备的硬件加速，从而使得浏览器的表现更佳。但这句话也得看情境——

> 想象使用 GPU 加速的动画就像是 Vin Diesel（速度与激情的主角）开着 Dominic 标志性的汽车 —— Dodge Charger。它的定制 900 hp 引擎可以让它在一瞬间从 0 加速到 60 码。但是如果你开着它在拥挤的高速公路上又有什么用呢？这种情况下错的不是你的车辆，而是你还在一条拥堵的高速公路上。—— 《CSS 硬件加速的好与坏》(http://efe.baidu.com/blog/hardware-accelerated-css-the-nice-vs-the-naughty/)

因此千万别贪心，将 3D 效果数量控制在一定范围内，页面性能才是重中之重。——来自得到惨痛教训的笔者的忠告。

扩展阅读

- Intro to CSS3 3D transforms (https://desandro.github.io/3dtransforms/) by David

DeSandro —— 详尽又新鲜的 3D Transformers 手册，包含许多一看就懂的小 Demo，妈妈再也不用担心我的 3D 了。
- Perspective (graphical) (https://en.wikipedia.org/wiki/Perspective_%28graphical —— 对透视学还一知半解的可以看看维基的详细说明。
- Unfolding the Box Model: Exploring CSS 3D Transforms (http://rupl.github.io/unfold/) by Chris Ruppel —— 非常赞的 3D Transforms 介绍，从 2D 到 3D 过渡，启动联想学习法，一看就明白，就怕你不看。
- CSS 硬件加速的好与坏 (http://efe.baidu.com/blog/hardware-accelerated-css-the-nice-vs-the-naughty/) —— 很多事情都不是一两句能讲清楚的，但是只要深入了解原理，一两句都不用讲就清楚了。

小结

本小节结合案例为大家介绍了实现 3D 效果的几个关键点：透视的概念理解—— `perspective`、空间变换体系 —— `transform-style`、Z 轴位移 —— `translateZ`。读者可以通过我们提供的丰富案例进一步体会 3D 效果的具体实现。