testtesttest

CSS3 3D Transformations

Css 3d 变换

This entry is part 3 of 5 in the series CSS3 Transformations

这个是css3变换系列的第三部分

In the third part of this series it seems appropriate to discuss three-dimensional transformations.

在这系列第三部分里面我们将讨论3d变换

As you’d expect, 3D is more complex and slightly less stable in browsers. To ensure reasonable compatibility:

如你所知，3d变换更加复杂，在浏览器之间的稳定性会稍微弱点。为此需要确保一定的兼容性：

Use the standard and -webkit- prefixed versions of all properties (these are not shown in the code samples below but are in the demonstrations).

所有属性使用标准和以-webkit前缀开头（这些在代码样例没有显示，但是在演示里面会有）

Forget IE9 and below. 3D transformations are supported in IE10 but the browser is missing a key property and you may need to adapt code for full cross-browser support (more about that late

3D on a 2D Screen?

忘掉IE9和他的前辈吧，3d变换在IE10里才部分支持，因为他不支持key属性，你也许需要让代码在多个浏览器上支持（更多是在2d的屏幕上显示3d效果）

In two dimensions, we have a horizontal x-axis and vertical y-axis. In three dimensions, we also have a z-axis. If you assume your screen is z co-ordinate zero, a positive value moves an object toward you and a negative value moves it away from you into the screen. Add some perspective and an object may disappear into the distance.

在二维空间里，我们有x轴和y轴。在三维空间里，我们有z轴。假设你的屏幕是z轴的0点，那么正值表示对象向你移动，负值表示对象远离你。添加一些透视（解释），对象有可能在远方某个地方消失

扩展：

1. 透视：我们暂时理解为我们的眼睛，后面详细介绍
2. 为什么会消失？视觉盲区？

其实这个很好理解，因为我们眼睛有个特性，就是看到远处的东西就很小，看到近处的东西很大，当物体离我们太近的时候，我们眼睛里面就只有他了，这就是所谓“一叶蔽目，不见泰山”，就是这么回事。至于为什么看不到，那是因为我们眼睛只能看到他前面的东西，而看不到他后面的东西。所以当物理超过了我们眼睛的位置，那么就看不到咯，除非你回头，哈哈。。。

There are some other points you should note which are specific to browsers:

对于特定浏览器有些其他地方你应该注意：

We’re transforming two dimensional elements in 3D space. HTML boxes remain flat and have zero depth. Even though you can scale an element in z-plane, it won’t extrude — i.e. your circle will remain a circle and won’t become a cylinder.

我们在3d空间里面变换二维元素。Html盒子保持扁平，没有深度。甚至你z轴平面扩展一个元素，他也不会被挤压——即你的圆仍然是圆，而不会变成圆柱体

A 3D transformation is applied to all inner elements.

一个元素使用3d变换，那么他的所有子元素也继承3d变换效果

It’s a little too easy to transform an element so it’s moved behind you or off the screen in some way! My recommendation: test often and undo any catastrophic transformations.

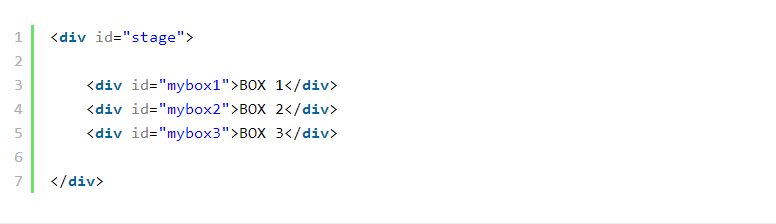
因为相对屏幕远近对一个元素变换太容易，所以我提议，需要经常测试，这样可以有效避免任何糟糕的变换效果

Applying Depth

应用深度

We’ll start with a simple example:

我们以一个简单例子开始：

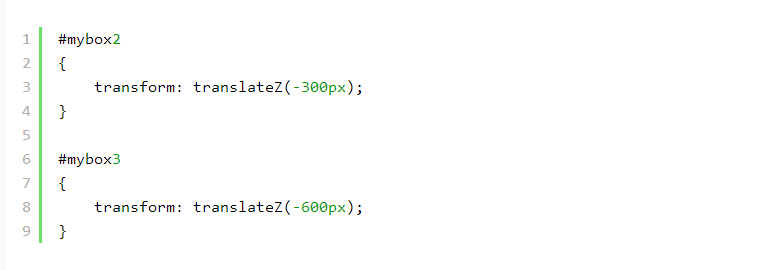


Our outer element (#stage) contains three boxes colored a transparent red, green and blue respectively. The outer element isn’t strictly required but it will allow us to rotate the whole scene which helps with visualization.

我们的外部元素(#stage)包含三个分别用红黄蓝纯色定义盒子。外部元素容许我们旋转整个屏幕，以帮助我们更形象的理解

First, we’ll attempt to move boxes two and three further back into the screen using the translateZ function which is identical to translateX and translateY except it applies to depth:

首先，我们尝试移动第二个和第三个盒子，使用translateZ沿着z轴向屏幕后面移动，x轴和y轴方向使用默认值，除了应用深度

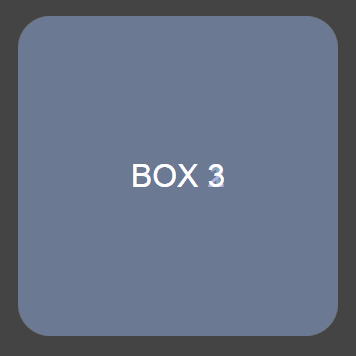


View the first 3D transformation demonstration page…

查看3d变换demo页面

<http://blogs.sitepointstatic.com/examples/tech/css3-transformations/3dtransform1.html>

The result:



That’s not what we expected; BOX 3 is certainly above the others. The reason is the transform-style property which has a default value of flat. This applies all 3D transformations in the same two dimensional plane so, while transforms occur, there is no illusion of depth. To fix this, we need to apply a value of preserve-3d to our outer #stage container:

这不是我想要的效果，盒子3在所有盒子之上。是因为变换属性有个flat的默认值。这个会应用到所有3d变换效果，同样也会对2d效果有影响。为了修正这个结果，我们需要给我们的父级元素#stage容器，应用一个preserve-3d的值



扩展：

他的完整写法是transform-style: flat|preserve-3d,有两个值，flat表示平面，preserve-3d表示3D透视，如果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

注意一点：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。

Transform默认值flat

preserve-3d：

View the second 3D transformation demonstration page…

查看demo

<http://blogs.sitepointstatic.com/examples/tech/css3-transformations/3dtransform2.html>



IE10 is Flat!

IE10居然是扁平的！

Unfortunately, IE10 does not support transform-style so BOX 3 will remain above BOX 1. All is not lost since we can apply 3D transformations to individual elements. However, it’s not possible to apply effects to a container with multiple child elements so the following demonstrations will not work in the browser. Let’s hope Microsoft address the issue in IE11.

不幸的是，IE10不支持transform-style，所以盒子3仍旧在盒子1上面。所有并不是因为我们给每个元素应用3d变换就可以了。可是，他不可能影响到拥有多个孩子元素的父容器，所以下面的例子在浏览器下面都不起作用了。让我们期待IE11可以解决这些问题

So, IE10 apart, we’re getting closer but there’s no illusion of depth. Let’s…

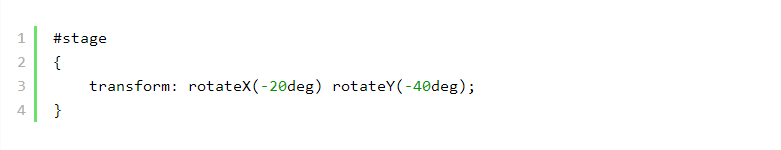
因为IE10的部分支持，虽然我们已经接近最终的效果了，但是由于没有深度，所以没有了立体感。让我们。。。

Get Some Perspective

添加一些透视

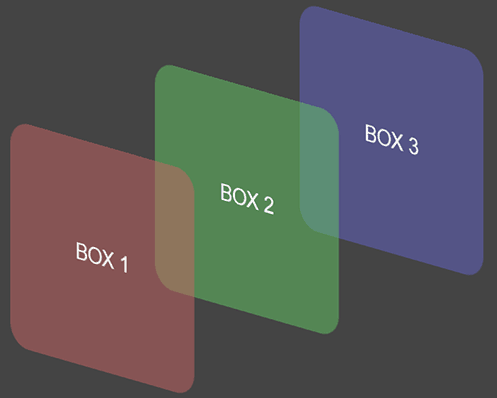
In the demonstration above we’re sitting at the front of the stage looking at our three boxes. But there’s no perspective so BOX 1 is exactly the same size as BOX 2 and BOX 3. We can illustrate this by moving the whole outer #stage:

在前面的例子中，我们是坐在舞台的前面观看三个盒子。但是因为没有透视，所以盒子1跟盒子2和盒子3完全一样的大小。我们可以通过移动父级#stage来解释这个



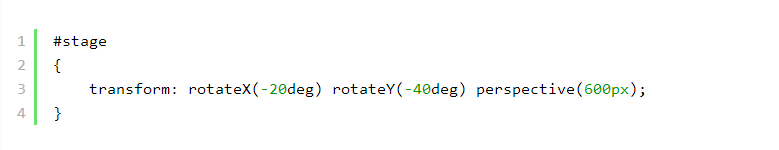
We’re now sitting on the right side of the the theater looking at our stage with the three boxes:

现在我们看到的效果就像是坐在舞台右侧一样，看到三个盒子的侧面



The perspective transform function will ensure objects which are further away become smaller:

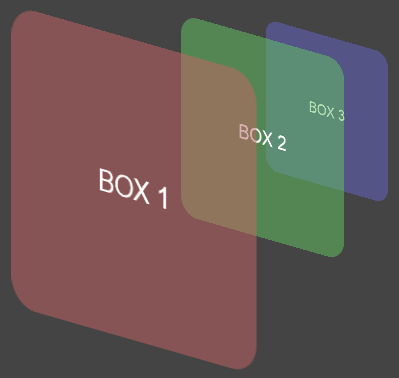
透视变换函数就是确保远处的对象看起来更小



View the third 3D transformation demonstration page…

查看demo

<http://blogs.sitepointstatic.com/examples/tech/css3-transformations/3dtransform3.html>



The perspective function is passed a single value which indicates how close the viewer is to the action. The lower the number, the more pronounced the perspective effect. Think of 0px as sitting on the front row of the #stage; it’s easier to see the relative distances between actors and objects. A value of 1000px would be the back of the upper circle; it’s more difficult to judge accurate distances so the objects appear to be a similar size.

透视函数只传入一个参数时，表示对象是如何靠近观众移动的。数值越小，那么透视效果越明显。值0px好像坐在#stage舞台的最前排，是很容易看到演员和对象之间的相对距离的。值1000px表示到了剧场的很后面，这样很难区分相似大小对象移动的准确距离

Try using your own #stage rotation and perspective values to see how it affects the elements. In the next lesson, we’ll apply other transform functions to our 3D scene.

尝试在自己的#stage元素上使用旋转和透视，看看他们是如何影响元素。下一课，我们将研究另一个3d变换函数

参考：

<http://www.zhangxinxu.com/wordpress/2012/09/css3-3d-transform-perspective-animate-transition/>

原文：<http://www.sitepoint.com/css3-transformations-3d/>