先讨论几个 css 问题

1，css 清除浮动的方法

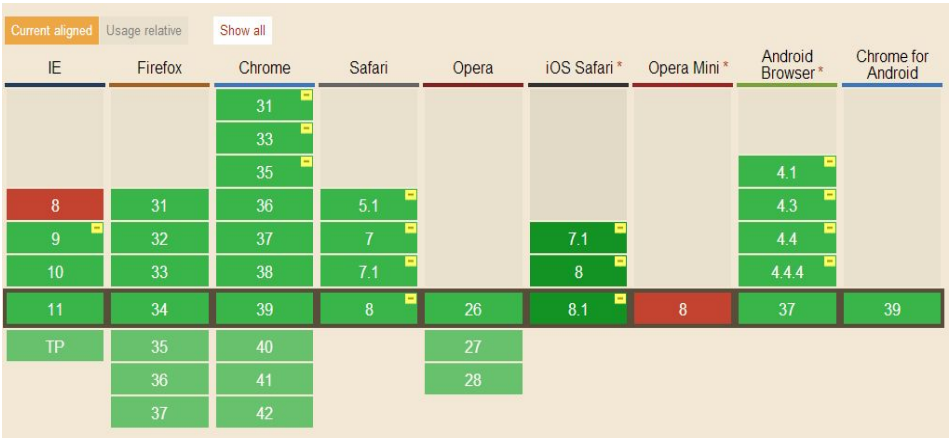
2，css 居中

3，多行省略号

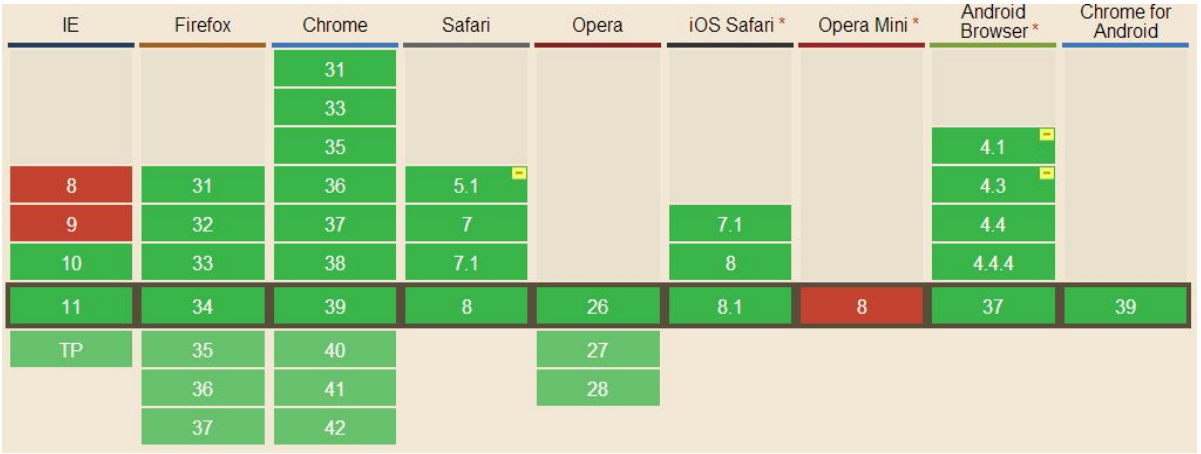
4，小布局技巧

2D 动画功能属性兼容性：transform、transition、animation

transform 可以实现旋转、缩放、倾斜、移动四种类型的变形处理。



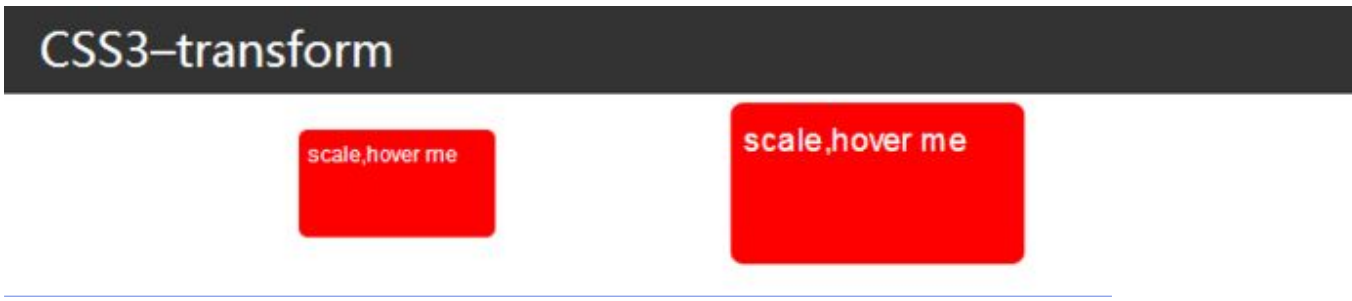
transition、animation



一、transform

1.scale：缩放，1 为原始大小。scale(x)。正数放大，负数缩小。属性值为一个时，x/y 轴同时缩放；属性值为两个值时，分别控制 x、y 轴的缩放。

效果图如下：



2.rotate：水平旋转，属性值格式为 Xdeg。(deg 是“度”的意思)rotate(Xdeg)。X 为正数时，顺时针旋转；为负数时，逆时针旋转

效果图如下：

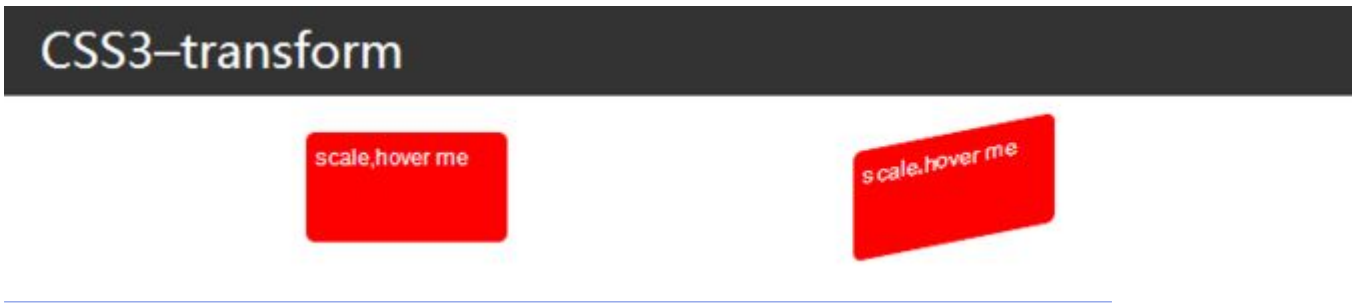
3.translate：定位元素，基于 XY 轴重新定位元素。translate(Xpx,Ypx)。效果图如下：

4.skew：将元素沿水平方向倾斜变形。skew(Xdeg,Ydeg)。这个比较难表述，想象一下平行四边形吧。属性值为一个时，x、y 轴参数相同；为两个时，x、y 轴各自倾斜

效果图如下：

5.matrix：矩阵，六个值。

效果图如下：



关于 **Matrix:**

2d matrix 提供 6 个参数啊 a,b,c,d,d,e,f 其基本写法如下：

$$\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

回顾一下高中数学，或者线性代数，即可知道 matrix 计算方法。x 和 y 是元素初始的坐标，x′ 和 y′则是通过矩阵变换后得到新的坐标。通过中间的那个 3×3 的变换矩阵，对原先的坐标施加变换，就能得到新的坐标了。依据矩阵变换规则即可得到：**x′=ax+cy+e, y′=bx+dy+f。**

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax+cy+e \\ bx+dy+f \\ 1 \end{bmatrix}$$

transform 中 translate，scale，rotate，skew 背后实现原理也对应着 matrix 变化：

translate(x,y)

一、移动 translate

移动 matrix 参数为：matrix(1,0,0,1,Δx,Δy)（Δx,Δy 分别对应 x 和 y 轴的增量）。由此公式可知：

-webkit-transform: translate(100px,100px);即对应着-webkit-transform: matrix(1, 0, 0, 1, 100, 100);

推算出： **x’ = 1*x+0 *** y+100 = x+100 ， **y’ = 0 *** x+1 * y+100 = y+100。

二、缩放 scale

缩放 matrix 参数为：matrix(kx*x,0,0,ky*y,0,0)（kx，和 ky 分别对应 x 和 y 轴缩放比率）。由此公式可知：

-webkit-transform: scale(1.5,1.5);及对应着 -webkit-transform: matrix(1.5, 0, 0, 1.5, 0, 0);

推算出： **x’ = 1.5*x+0** * y+0 = 1.5 * x， **y’ = 0** * x+1.5 * y+0 =1.5 * y。

三、旋转 rotate

旋转 matrix 参数为：matrix(cosθ,sinθ,-sinθ,cosθ,0,0)，由此可知

-webkit-transform: rotate(45deg);即对应着 -webkit-transform: matrix(0.53, 0.85, -0.85, 0.53, 0, 0);

（sin(45′)=0.85，cos(45′)=0.53）

推算： **x’ = x*cos(45′)-y*sin(45′)+0 = x*cos(45′)-y*sin(45′)，y’ = x*sin(45′)+y*cos(45′)+0 = x*sin(45′)+y*cos(45′)**

四、斜切 skew

斜切 matrix 参数为 matrix(1,tan(θy),tan(θx),1,0,0)，由此可知

-webkit-transform: skew(45deg);即对应着 -webkit-transform: matrix(1,0,1,1,0,0);

（tan(45′)=1）

推算出 **x’ = x+y*tan(45′)+0** = x+y*tan(**45′**)， **y’ = x*tan(45′)**+y+0 = x*tan(**45′**)+y

四个值可以组合使用,并可以指定基准点使用，transform-origin

3D 动画功能属性

处理思路和 2D 类似，只是有 2D 的矩阵处理扩展到 3D 的矩阵处理

3d 矩阵即为透视投影，推算方法与 2d 矩阵相类似

$$\begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3d 变换矩阵代码示例，matrix 变为 matrix3d

```
-webkit-transform: matrix3d(1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
```


```
1, 0, 0, 0, 0, 1)
```

---

rotate3d(x, y, z, angle)

【参数说明】

x,y,z　　组成了方向向量(Direction Vector),旋转的方向是从方向向量这个点指向

transition-origin 的顺时针方向

angle　　沿着 transition-origin ==> V(x,y,z)旋转轴的顺时针方向的旋转角度

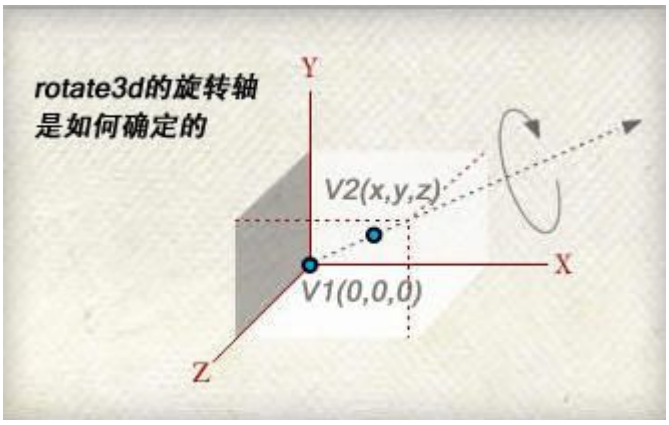


图　rotate3d 如何确定旋转轴

由上图来看，V1 为 transition-origin 的点的位置,默认为 V1(0,0,0),V2 则为 rotate3d(x,y,z,angle)的前三个值组成的点，这样就确定了旋转轴，然后在沿着旋转轴顺时针转动 angle 的角度便可。

rotateX(angle)　==> rotate3d(1,0,0,angle)

rotateY(angle)　==> rotate3d(0,1,0,angle)

rotateZ(angle)　==> rotate3d(0,0,1,angle)　即 rotate(angle)

类似的原理结合 2D 的可以推出其他几个属性的 matrix 的变换矩阵参数

二、transition

transiton: porperty duration timing-function delay

property 表示对哪个属性进行过度，duration 表示动画时间，timing-function 表示通过什么方式进行过渡,delay 表示延时后执行。

```
p {
-webkit-transition: all .5s ease-in-out 1s;
-o-transition: all .5s ease-in-out 1s;
-moz-transition: all .5s ease-in-out 1s;
transition: all .5s ease-in-out 1s;
}
```

transition 涉及到的则是另一个数学概念：贝塞尔插值。

transition 的变换函数有 linear ease ease-in ease-out ease-in-out 几种，通常我们尝试使用时能感觉到它们之间稍有不同。

实际上，它们是使用了不同的参数进行三次贝塞尔插值计算的结果。复习一下贝塞尔插值：

一个量（可以是任何矢量或者标量）从一个值到变化到另一个值，如果我们希望它按照一定时间平滑地过渡，就必须要对它进行插值。

最基本的情况，我们认为这个变化是按照时间均匀进行的，这个时候，我们称其为线性插值。而实际上，线性插值不大能满足我们的需要，因此数学上出现了很多其它的插值算法，其中贝塞尔插值法是非常典型的一种。它根据一些变换中的控制点来决定值与时间的关系。

K 次贝塞尔插值算法需要 k+1 个控制点，最简单的一次贝塞尔插值就是线性插值，将时间表示为 0 到 1 的区间，一次贝塞尔插值公式是：

$$\mathbf{B}(t) = \mathbf{P}_0 + (\mathbf{P}_1 - \mathbf{P}_0)t = (1-t)\mathbf{P}_0 + t\mathbf{P}_1 \ , t \in [0,1]$$

二次贝塞尔插值有 3 个控制点，相当于对 P0 和 P1，P1 和 P2 分别做贝塞尔插值，再对结果做一次贝塞尔插值计算

$$\mathbf{B}(t) = (1-t)^2\mathbf{P}_0 + 2t(1-t)\mathbf{P}_1 + t^2\mathbf{P}_2 \ , t \in [0,1]$$

三次贝塞尔插值则是两次二次贝塞尔插值的结果在做一次贝塞尔插值：

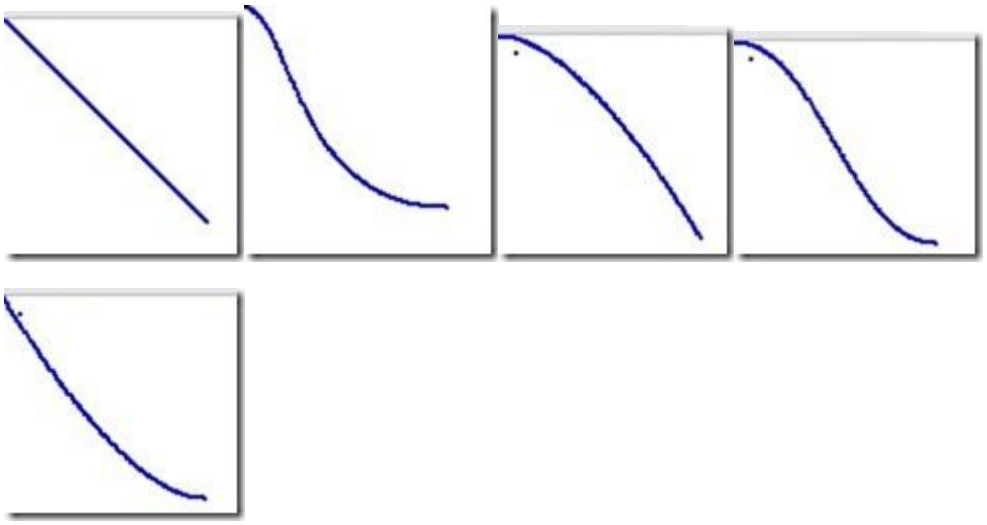
$$\mathbf{B}(t) = \mathbf{P}_0(1-t)^3 + 3\mathbf{P}_1 t(1-t)^2 + 3\mathbf{P}_2 t^2(1-t) + \mathbf{P}_3 t^3 \ , t \in [0,1]$$

回到我们的 transition，我们要用到的就是三次贝塞尔插值算法了。

把时间认为是 0,1 区间，待变换属性也认为是 0,1 区间，那么所有的控制函数都是三次贝塞尔函数：

ease：
ease 函数等同于贝塞尔曲线(0.25, 0.1, 0.25, 1.0).
linear：
linear 函数等同于贝塞尔曲线(0.0, 0.0, 1.0, 1.0).
ease-in：
ease-in 函数等同于贝塞尔曲线(0.42, 0, 1.0, 1.0).
ease-out：
ease-out 函数等同于贝塞尔曲线(0, 0, 0.58, 1.0).
ease-in-out：
ease-in-out 函数等同于贝塞尔曲线(0.42, 0, 0.58, 1.0)
cubic-bezier：
特定的 cubic-bezier 曲线。(x1, y1, x2, y2)四个值特定于曲线上点 P1 和点 P2。所有值需在[0, 1]区域内，否则无效。

最后附上各函数图，请自行辨认：



三、animation

功能与 transition，但是通过 transition 功能只能通过指定的属性的开始值和结束值，然后在这两个属性之间进行平滑过渡，不能实现复杂的动画效果，而 animation 则是通过关键帧以及定义关键帧中的元素属性值来实现更为负载的复杂效果。

animation: name duration timing-function delay iteration_count direction;

（1） -webkit-animation-name　这个属性的使用必须结合@-webkit-keyframes 一起使用

```
@-webkit-keyframes fontchange{
0%{font-size:10px;}
30%{font-size:15px;}
100%{font-siez:12px;}
}
```

百分比的意思就是 duration 的百分比，如果没有设置 duration 的话，则表示为无穷大

div{ -webkit-animation-name:fontchange;}

（2）-webkit-animation-duration　表示动画持续的时间

（3）-webkit-animation-timing-function　表示动画使用的时间曲线

【语法】：　-webkit-animation-timing-function: ease | linear | ease-in | ease-out | ease-in-out

（4）-webkit-animation-delay　　表示开始动画之前的延时

【语法】 -webkit-animation-delay: delay_time;

（5）-webkit-animation-iteration-count　表示动画要重复几次

【语法】-webkit-animation-iteration-count: times_number;

（6） -webkit-animation-direction　表示动画的方向

【语法】-webkit-animation-direction: normal(默认) | alternate

normal　方向始终向前

alternate 当重复次数为偶数时方向向前，奇数时方向相反