Css动画

首先我们来看几个css动画：

<https://www.html5tricks.com/tag/css3%E5%8A%A8%E7%94%BB/>

| **属性** | **含义** |
| --- | --- |
| animation（动画） | 用于设置动画属性，他是一个简写的属性，包含6个属性 |
| transition（过渡） | 用于设置元素的样式过度，和animation有着类似的效果，但细节上有很大的不同 |
| transform（变形） | 用于元素进行旋转、缩放、移动或倾斜，和设置样式的动画并没有什么关系，就相当于color一样用来设置元素的“外表” |
| translate（移动） | translate只是transform的一个属性值，即移动。 |

先区分一下css中的几个属性：**animation（动画）、transition（过渡）、transform（变形）、translate（移动）**。

　　CSS3中的**transform(变形)**属性用于内联元素和块级元素，可以旋转、扭曲、缩放、移动元素，它的属性值有以下五个：旋转rotate、扭曲skew、缩放scale和移动translate以及矩阵变形matrix；

**transform(变形)是CSS3中的元素的属性，而translate只是transform的一个属性值；transform是transition（过渡动画）的transition-property的一个属性值。**

**Transtrom属性如下**

|  |  |
| --- | --- |
| none | 定义不进行转换。 |
| matrix(*n*,*n*,*n*,*n*,*n*,*n*) | 定义 2D 转换，使用六个值的矩阵。 |
| matrix3d(*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*) | 定义 3D 转换，使用 16 个值的 4x4 矩阵。 |
| translate(*x*,*y*) | 定义 2D 转换。 |
| translate3d(*x*,*y*,*z*) | 定义 3D 转换。 |
| translateX(*x*) | 定义转换，只是用 X 轴的值。 |
| translateY(*y*) | 定义转换，只是用 Y 轴的值。 |
| translateZ(*z*) | 定义 3D 转换，只是用 Z 轴的值。 |
| scale(*x*,*y*) | 定义 2D 缩放转换。 |
| scale3d(*x*,*y*,*z*) | 定义 3D 缩放转换。 |
| scaleX(*x*) | 通过设置 X 轴的值来定义缩放转换。 |
| scaleY(*y*) | 通过设置 Y 轴的值来定义缩放转换。 |
| scaleZ(*z*) | 通过设置 Z 轴的值来定义 3D 缩放转换。 |
| rotate(*angle*) | 定义 2D 旋转，在参数中规定角度。 |
| rotate3d(*x*,*y*,*z*,*angle*) | 定义 3D 旋转。 |
| rotateX(*angle*) | 定义沿着 X 轴的 3D 旋转。 |
| rotateY(*angle*) | 定义沿着 Y 轴的 3D 旋转。 |
| rotateZ(*angle*) | 定义沿着 Z 轴的 3D 旋转。 |
| skew(*x-angle*,*y-angle*) | 定义沿着 X 和 Y 轴的 2D 倾斜转换。 |
| skewX(*angle*) | 定义沿着 X 轴的 2D 倾斜转换。 |
| skewY(*angle*) | 定义沿着 Y 轴的 2D 倾斜转换。 |
| perspective(*n*) | 为 3D 转换元素定义透视视图。 |

animation（动画）、transition（过渡）是css3中的两种动画属性。animation强调流程与控制，对元素的一个或多个属性的变化进行控制，可以有多个关键帧（animation 和@ keyframes结合使用）；

　　transition强调过渡，是元素的一个或多个属性发生变化时产生的过渡效果，同一个元素通过两个不同的途径获取样式，而第二个途径当某种改变发生（例如hover）时才能获取样式，这样就会产生过渡动

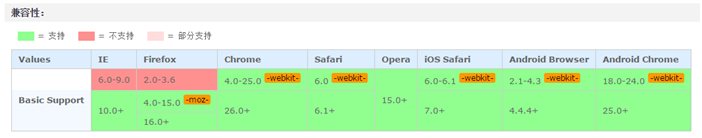
画。可以认为它有两个关键帧（Transition ＋ Transform ＝ 两个关键帧的Animation）。

弄清楚这几个问题，我们就可以头脑清醒的状态下去学习css的动画

**transition**

什么叫过渡？字面意思上来讲，就是元素从这个属性(color)的某个值(red)过渡到这个属性(color)的另外一个值(green)，这是一个状态的转变，需要一种条件来触发这种转变，比如我们平时用到的:hoever、:focus、:checked、媒体查询或者JavaScript。但是**以下情况下，属性值改变不能产生过渡效果**

1. background-image，如url(a.jpg)到url(b.jpg)（与浏览器支持相关，有的浏览器不支持）等，[浏览器支持情况](http://thewebevolved.com/support/animation/properties/" \l "background-image" \t "_blank)
2. float浮动元素
3. height或width使用auto值
4. display属性在none和其他值（block、inline-block、inline）之间变换
5. position在stativ和absolute之间变换



先从一个简单的demo来看看transition的效果

<!DOCTYPE html>

<html lang="en">

<head>

<title>transition</title>

<style>

#box {

  height: 100px;

  width: 100px;

  background: green;

  transition: transform 1s ease-in 1s;

}

#box:hover {

    transform: rotate(180deg) scale(.5, .5);

}

</style>

</head>

<body>

    <div id="box"></div>

</body>

</html>

效果：

我们来分析这一整个过程，首先transition给元素设置的过渡属性是transform，当鼠标移入元素时，元素的transform发生变化，那么这个时候就触发了transition，产生了动画，当鼠标移出时，transform又发生变化，这个时候还是会触发transition，产生动画，所以transition产生动画的条件是transition设置的property发生变化，这种动画的特点是需要“一个驱动力去触发”，有着以下几个不足：

1. 需要事件触发，所以没法在网页加载时自动发生
2. 是一次性的，不能重复发生，除非一再触发
3. 只能定义开始状态和结束状态，不能定义中间状态，也就是说只有两个状态

语法：**transition: property duration timing-function delay;**

| **值** | **描述** |
| --- | --- |
| transition-property | 规定设置过渡效果的 CSS 属性的名称 |
| transition-duration | 规定完成过渡效果需要多少秒或毫秒 |
| transition-timing-function | 规定速度效果的速度曲线（具体见下表） |
| transition-delay | 定义过渡效果何时开始 |

|  |  |
| --- | --- |
| linear | 规定以相同速度开始至结束的过渡效果（等于 cubic-bezier(0,0,1,1)）。 |
| ease | 规定慢速开始，然后变快，然后慢速结束的过渡效果（cubic-bezier(0.25,0.1,0.25,1)）。 |
| ease-in | 规定以慢速开始的过渡效果（等于 cubic-bezier(0.42,0,1,1)）。 |
| ease-out | 规定以慢速结束的过渡效果（等于 cubic-bezier(0,0,0.58,1)）。 |
| ease-in-out | 规定以慢速开始和结束的过渡效果（等于 cubic-bezier(0.42,0,0.58,1)）。 |
| cubic-bezier(n,n,n,n) | 在 cubic-bezier 函数中定义自己的值。可能的值是 0 至 1 之间的数值。 |

**animation**

在官方的介绍中这个属性是transition属性的扩展，弥补了transition的很多不足，我理解为animation是由多个transition的效果叠加，并且可操作性更强，能够做出复杂酷炫的效果(前提是你爱折腾)，我们以一个例子来介绍animation的威力：

<!DOCTYPE html>

<html lang="en">

<head>

<title>animation</title>

<style>

.box {

  height: 100px;

  width: 100px;

  border: 15px solid black;

  animation: changebox 1s ease-in-out 1s infinite alternate running forwards;

}

.box:hover {

    animation-play-state: paused;

}

@keyframes changebox {

  10% {

      background: red;

  }

  50% {

      width: 80px;

  }

  70% {

      border: 15px solid yellow;

  }

  100% {

    width: 180px;

    height: 180px;

  }

}

</style>

</head>

<body>

    <div class="box"></div>

</body>

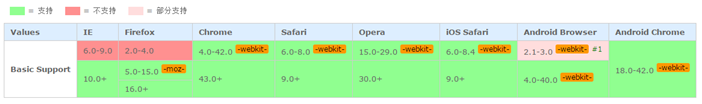
</html>

分析：

我们先来看看keyframes这个关键点，它定义了一个动画组合叫changebox，里面的10%，50%，70%，100%代表在变化中不同时间点的属性值，比如这个动画的总时间是1s，那么10%就是在0-0.1s中的动画，通过这个我们可以较精确的控制动画变化中任何一个时间点的属性效果，这大大提高了我们对动画的把控，是做复杂动画的基础，我们再回来看animation中整整八个值，是不是有点夸张，还没见过这么长的值，通过控制animation的每个值，控制动画变得非常灵活，我们来具体了解它的语法以及各个值代表着什么：

语法：**animation: name duration timing-function delay iteration-count direction play-state fill-mode;**

| **值** | **描述** |
| --- | --- |
| name | 用来调用@keyframes定义好的动画，与@keyframes定义的动画名称一致 |
| duration | 指定元素播放动画所持续的时间 |
| timing-function | 规定速度效果的速度曲线，是针对每一个小动画所在时间范围的变换速率 |
| delay | 定义在浏览器开始执行动画之前等待的时间，值整个animation执行之前等待的时间 |
| iteration-count | 定义动画的播放次数，可选具体次数或者无限(infinite) |
| direction | 设置动画播放方向：normal(按时间轴顺序),reverse(时间轴反方向运行),alternate(轮流，即来回往复进行),alternate-reverse(动画先反运行再正方向运行，并持续交替运行) |
| play-state | 控制元素动画的播放状态，通过此来控制动画的暂停和继续，两个值：running(继续)，paused(暂停) |
| fill-mode | 控制动画结束后，元素的样式，有四个值：none(回到动画没开始时的状态)，forwards(动画结束后动画停留在结束状态)，backwords(动画回到第一帧的状态)，both(根据animation-direction轮流应用forwards和backwards规则)，注意与iteration-count不要冲突(动画执行无限次) |



animation与transition 不同的是，keyframes提供更多的控制，尤其是时间轴的控制，这点让css animation更加强大，使得flash的部分动画效果可以由css直接控制完成，而这一切，仅仅只需要几行代码，也因此诞生了大量基于css的动画库，用来取代flash的动画部分