### 2.1 基本用法

首先，CSS Animation需要指定动画一个周期持续的时间，以及动画效果的名称。

div:hover {

animation: 1s rainbow;

}

上面代码表示，当鼠标悬停在div元素上时，会产生名为rainbow的动画效果，持续时间为1秒。为此，我们还需要用keyframes关键字，定义rainbow效果。

@keyframes rainbow {

0% { background: #c00; }

50% { background: orange; }

100% { background: yellowgreen; }

}

上面代码表示，rainbow效果一共有三个状态，分别为起始（0%）、中点（50%）和结束（100%）。如果有需要，完全可以插入更多状态。

**animation.html**

默认情况下，动画只播放一次。加入infinite关键字，可以让动画无限次播放。

div:hover { animation: 1s rainbow infinite;

}

也可以指定动画具体播放的次数，比如3次。

div:hover {

animation: 1s rainbow 3;

}

这里还有一个[心脏跳动](http://lea.verou.me/css-4d/" \l "heart-demo" \t "_blank)的例子，可供参考。

### 2.2 animation-fill-mode

动画结束以后，会立即从结束状态跳回到起始状态。如果想让动画保持在结束状态，需要使用animation-fill-mode属性。

div:hover {

animation: 1s rainbow forwards;

}

animation-fill-mode还可以使用下列值。

1. none：默认值，回到动画没开始时的状态。
2. forwards: 当动画完成后，保持最后一个属性值（在最后一个关键帧中定义）。

（3）backwards：让动画回到第一帧的状态。

（4）both？: 根据animation-direction（见后）轮流应用forwards和backwards规则。

**2.3 animation-direction**

动画循环播放时，每次都是从结束状态跳回到起始状态，再开始播放。animation-direction属性，可以改变这种行为。假定有一个动画是这样定义的。

@keyframes rainbow {

0% { background-color: yellow; }

100% { background: blue; }

}

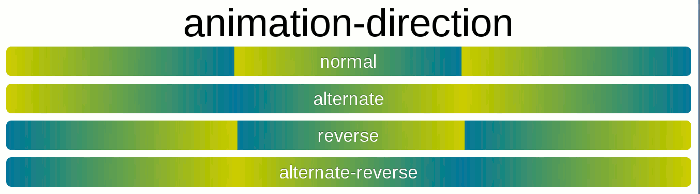
默认情况是，animation-direction等于normal。

div:hover {

animation: 1s rainbow 3 normal;

}

此外，还可以等于取alternate、reverse、alternate-reverse等值。它们的含义见下图（假定动画连续播放三次）。



简单说，animation-direction指定了动画播放的方向，最常用的值是normal和reverse。浏览器对其他值的支持情况不佳，应该慎用。

### 2.4 animation的各项属性

同transition一样，animation也是一个简写形式。

div:hover {

animation: 1s 1s rainbow linear 3 forwards normal;

}

这是一个简写形式，可以分解成各个单独的属性。

div:hover {

animation-name: rainbow;

animation-duration: 1s;

animation-timing-function: linear;

animation-delay: 1s;

animation-fill-mode:forwards;

animation-direction: normal;

animation-iteration-count: 3;

}

### 2.5 keyframes的写法

keyframes关键字用来定义动画的各个状态，它的写法相当自由。

@keyframes rainbow {

0% { background: #c00 }

50% { background: orange }

100% { background: yellowgreen }

}

0%可以用from代表，100%可以用to代表，因此上面的代码等同于下面的形式。

@keyframes rainbow {

from { background: #c00 }

50% { background: orange }

to { background: yellowgreen }

}

如果省略某个状态，浏览器会自动推算中间状态，所以下面都是合法的写法。

@keyframes rainbow {

50% { background: orange }

to { background: yellowgreen }

}

@keyframes rainbow {

to { background: yellowgreen }

}

甚至，可以把多个状态写在一行。

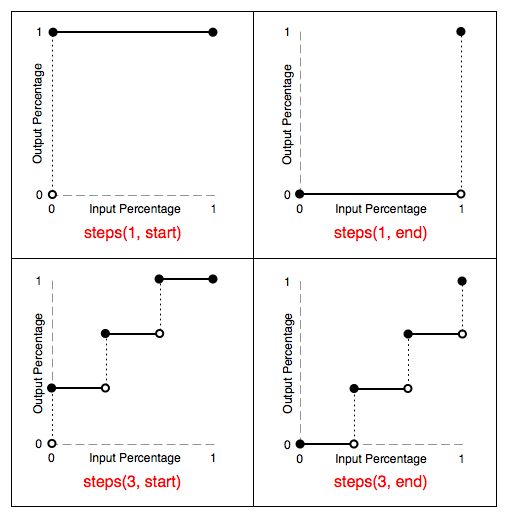
@keyframes pound {

from，to { transform: none; }

50% { transform: scale(1.2); }

}

另外一点需要注意的是，浏览器从一个状态向另一个状态过渡，是平滑过渡。steps函数可以实现分步过渡。steps 函数指定了一个阶跃函数，第一个参数指定了时间函数中的间隔数量（必须是正整数）；第二个参数可选，接受 start 和 end 两个值，指定在每个间隔的起点或是终点发生阶跃变化，默认为 end。



div:hover {

animation: 1s rainbow infinite steps(30,end);

}

这里有一个非常神奇的[例子](http://dabblet.com/gist/1745856" \t "_blank)，可以看到steps函数的用处。

**steps.html**

# [【译】css动画里的steps()用法详解](https://segmentfault.com/a/1190000007042048)

### 2.6 animation-play-state

规定动画正在运行还是暂停。有时，动画播放过程中，会突然停止。这时，默认行为是跳回到动画的开始状态。

**animation1.html**

如果想让动画保持突然终止时的状态，就要使用animation-play-state属性。

div {

animation: spin 1s linear infinite;

animation-play-state: paused;

}

div:hover {

animation-play-state: running;

}

上面的代码指定，没有鼠标没有悬停时，动画状态是暂停；一旦悬停，动画状态改为继续播放。

### 2.7 浏览器前缀

目前，IE 10和Firefox（>= 16）支持没有前缀的animation，而chrome不支持，所以必须使用webkit前缀。

也就是说，实际运用中，代码必须写成下面的样子。

div:hover {

-webkit-animation: 1s rainbow;

animation: 1s rainbow;

}

@-webkit-keyframes rainbow {

0% { background: #c00; }

50% { background: orange; }

100% { background: yellowgreen; }

}

@keyframes rainbow {

0% { background: #c00; }

50% { background: orange; }

100% { background: yellowgreen; }

}