# CSS3新特性+less实验——animation

原创

hjb2722404 于 2015-06-04 11:51:20 发布 3873 收藏

分类专栏： CSS理论与应用



本文链接：https://blog.csdn.net/hjb2722404/article/details/46358711





自从CSS3推出animation以来，一大批H5应用纷纷利用animation来制作以往需要JS或FLASH才能制作出的特效。今天就来看看animation的庐山真面目吧。

## 实验对象：animation

> animation可以被用来定义一组动画效果，此效果可以被应用在任何元素之上，并且可以通过它提供的各项参数精确控制动画的细节。

## 语法

> animation：[[ animation-name ] || [ animation-duration ] || [ animation-timing-function ] || [ animation-delay ] || [ animation-iteration-count ] || [ animation-direction ]]

## 说明

> animation-name：动画名称，必须与@keyframes配合使用，所有动画由@keyframes单独定义，在元素样式中通过animation-name来调用
> animation-duration:指定动画的持续时间，单位s或ms
> animation-timing-function :指定过渡类型，即速度虽时间的变化幅度，取值范围与transform的timing-function一样。
> animation-delay:定义动画开始前的延迟时间，单位s或ms
> animation-iteration-count ：指定动画循环次数。值可以为数字，也可以为infinite代表无限循环
> animation-direction:动画进行的方向，取值为normal[正常方向]和alternate[正常与反向交替]
> @keyframes：此属性单独使用，不在元素样式中，以下是一个示例：

```
@-webkit-keyframes animations{
    0%{-webkit-transform:translate(0,0);}
    50%{-webkit-transform:translate(100px,100px);}
    100%{-webkit-transform:translate(100px,0);}
}
```

我们看到，它的基本格式为：
@-浏览器前缀-keyframes 动画名称{
0% {这里定义一些transform效果}
50% {这里定义一些transform效果}
100% {这里定义一些transform效果}
……
}
这里的百分比表示的是时间的长度，总长度为animation-duration里定义的时间。

## 实例

由于@keyframes定义起来太过繁琐，还包括要兼容不同的浏览器，所以这里我们直接推荐一些成熟的animation组件——animate.css，免去了大家编写@keyframes效果的麻烦，当然，如果要做出更精致的动画来，在大家熟悉该属性的用法后就可以自己定义吊炸天的动画效果了。
animate.css官网下载

### HTML

```
……
 <title>CSS3新特性实验—animation</title>
    <link rel="stylesheet" href="animate.css" type="text/css"/>
    <link rel="stylesheet" href="style.css" type="text/css"/>
</head>
<body>
    <div class="content">

        <div class="box1"></div>
    </div>
</body>
</html>
```

这里要注意，animate.css要在style.css之前引入，这也是所有前端制作中应该遵循的规则：插件在自定义样式与脚本之前引入，防止插件样式和脚本覆盖自定义样式和脚本，并且自定义样式要调用插件中的样式，所以要确保插件在自定义样式之前加载，才能保证调用时可以找到相关对象。

### less

```
.animation(@n:fly,@t:2s,@fn:ease-in-out,@i:infinite,@dur:alternate){
  animation: @n @t @fn @i @dur;
  -webkit-animation: @n @t @fn @i @dur;
  -o-animation: @n @t @fn @i @dur;
  -moz-animation: @n @t @fn @i @dur;
}

.content {
  width: 1400px;
  height: 600px;
  border: 2px solid #ccc;
  margin-top: 100px;
  margin-left: 100px;
}

.box {
  width: 100px;
  height: 100px;
  background: blue;
  float: left;
  margin-right: 100px;
}

.box1{
  .box;
  .animation(slideOutUp);
}
```

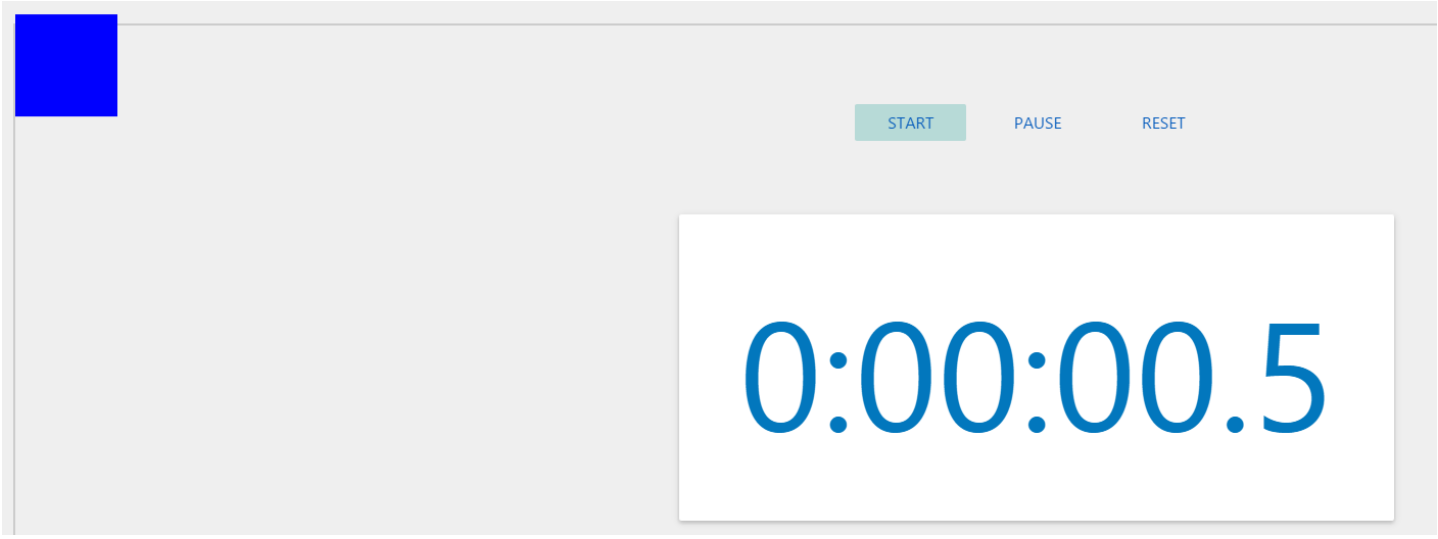
## CSS

```
.content {
  width: 1400px;
  height: 600px;
  border: 2px solid #ccc;
  margin-top: 100px;
  margin-left: 100px;
}
.box {
  width: 100px;
  height: 100px;
  background: blue;
  float: left;
  margin-right: 100px;
}
.box1 {
  width: 100px;
  height: 100px;
  background: blue;
  float: left;
  margin-right: 100px;
  animation: slideOutUp 2s ease-in-out infinite alternate;
  -webkit-animation: slideOutUp 2s ease-in-out infinite alternate;
  -o-animation: slideOutUp 2s ease-in-out infinite alternate;
  -moz-animation: slideOutUp 2s ease-in-out infinite alternate;
}
```

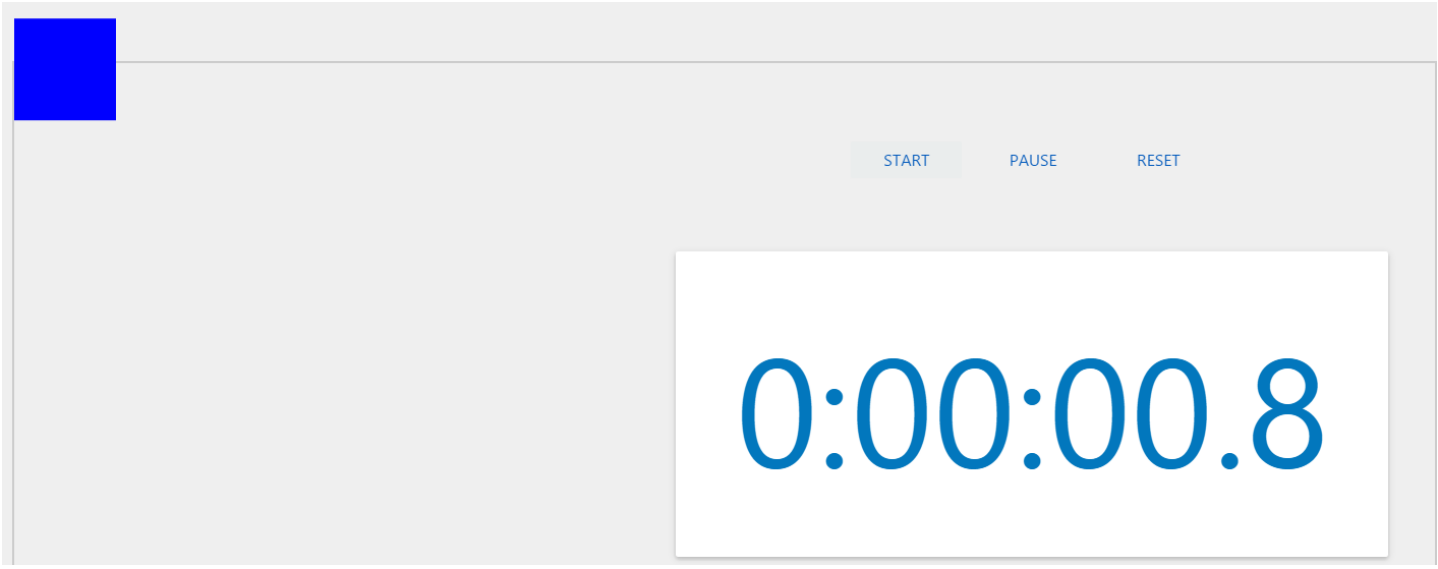
animate.css中的slideOutUp动画定义部分：

```
@-webkit-keyframes slideOutUp {
  0% {
    -webkit-transform: translate3d(0, 0, 0);
    transform: translate3d(0, 0, 0);
  }

  100% {
    visibility: hidden;
    -webkit-transform: translate3d(0, -100%, 0);
    transform: translate3d(0, -100%, 0);
  }
}

@keyframes slideOutUp {
  0% {
    -webkit-transform: translate3d(0, 0, 0);
    transform: translate3d(0, 0, 0);
  }

  100% {
    visibility: hidden;
    -webkit-transform: translate3d(0, -100%, 0);
    transform: translate3d(0, -100%, 0);
  }
}
```

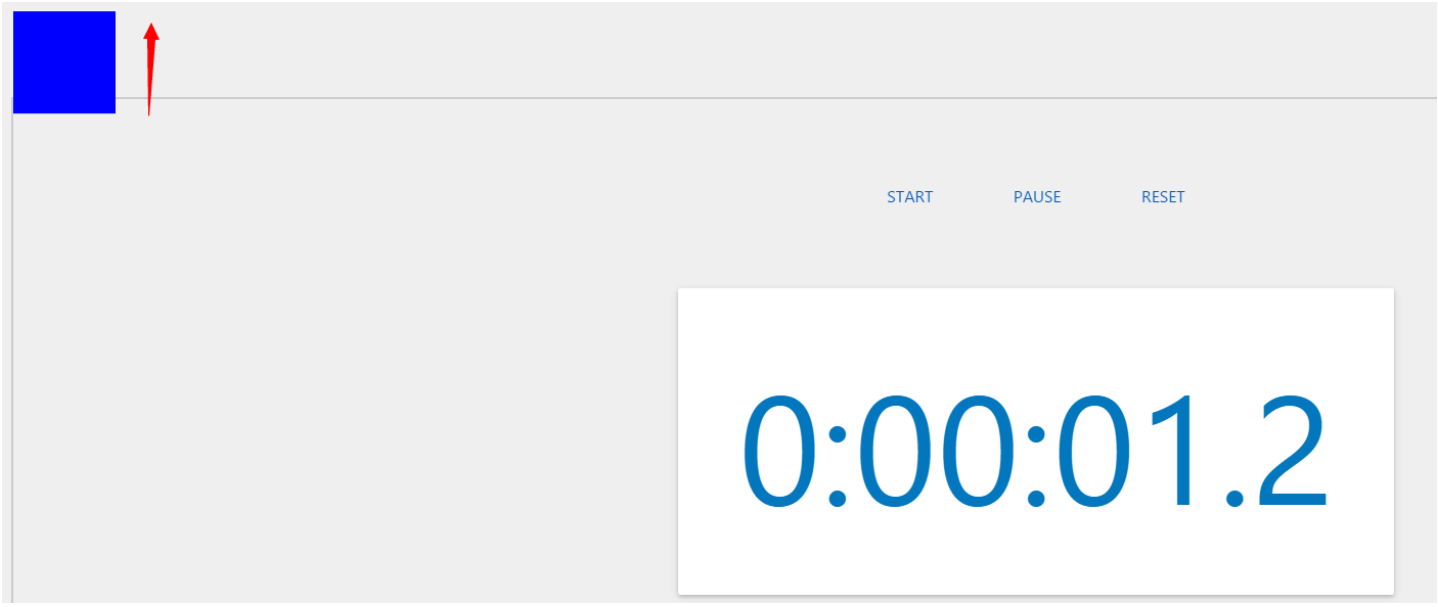
**效果**



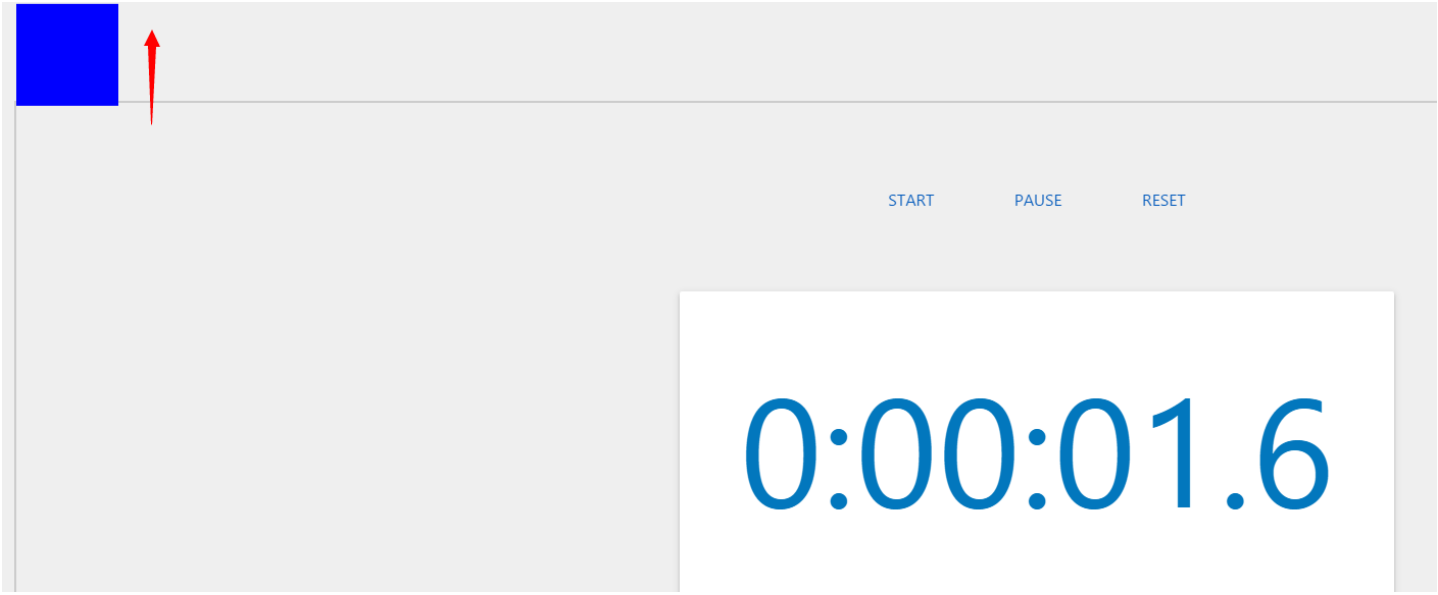

动画开始0.5s时的状态



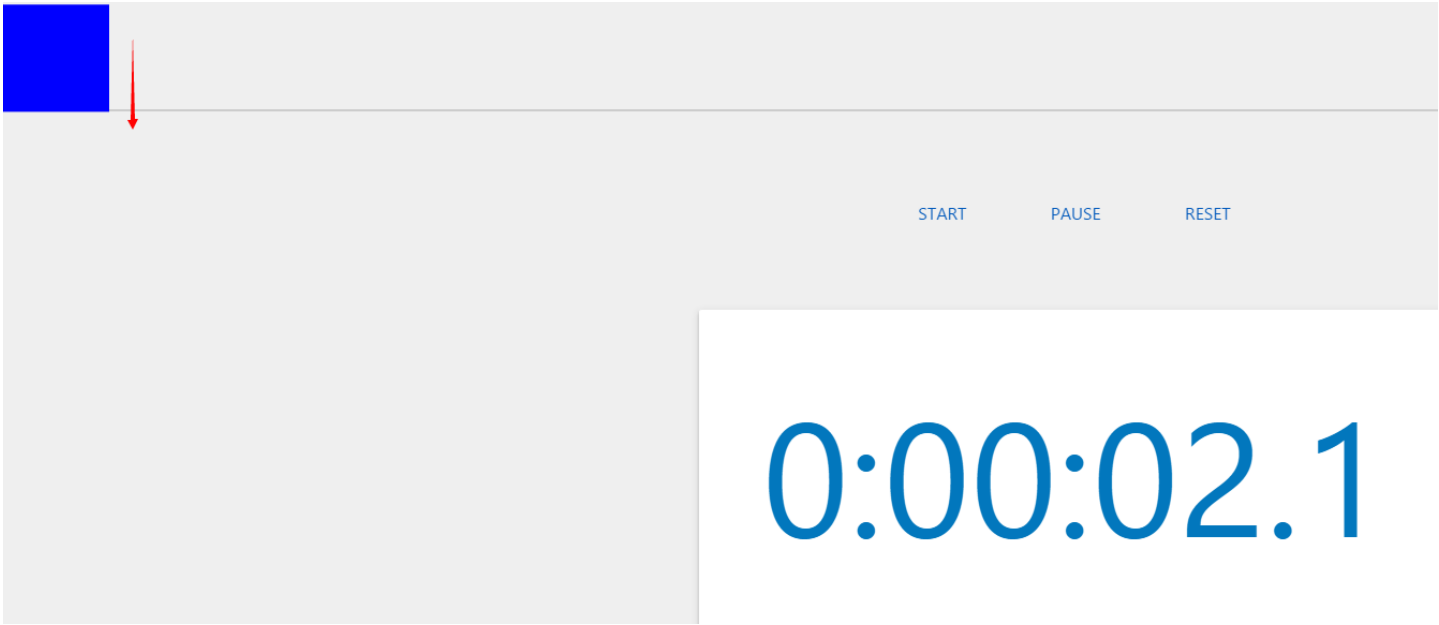

动画开始后0.8s时的状态




动画开始后1.2s时的状态




动画开始后1.6s时的状态




动画开始后2.1s时的状态

为了方便大家看清楚动画的变化过程与时间的关系，我特意在页面加入了秒表计时器。从最终效果我们可以看出：
1、我们设置的动画时间为2s，而动画slideOutUp里定义时间100%时元素运动到自身高度的100%的反方向处（ transform: translate3d(0, -100%, 0);），这里元素高度为100px,所以它在2秒时运动到相对自己-100px的地方，然后开始恢复到原始状态。
2、从图中效果我们可以看出，从0-0.8s,元素运动的幅度占到整个运动幅度的40%，从0.8s-1.2s的运动占了整个运动幅度的40%，而从1.2s到2.0s，只运动了整个运动幅度的20%，这个就是因为我们设置了timing-function为ease-in-out。(由慢到快再到慢)，大家可以自己改为其他过渡方式看看又会是怎样的效果。
3、我们这里动画方向设置为交替进行，所以元素恢复到初始状态时也是平滑过渡的，但是如果将动画方向设置为normal，在元素运动到-100px后，就会很生硬的瞬间跳回到初始位置，这个大家可以自己做实验验证啦，大家还可以把动画进行的次数改为1试试看有什么效果。
好啦，animation就讲到这里啦，其他各种效果其实就是不同参数组合的结果，千变万化，不离其宗，通过本文掌握了基本原理和写法，其他的就 so easy啦。