引用

<embed src="circle1.svg" type="image/svg+xml" />

<svg width="100%" height="100%" version="1.1"  
 xmlns="http://www.w3.org/2000/svg">  
 <rect width="300" height="100"  
 style="fill:rgb(0,0,255);stroke-width:1;  
stroke:rgb(0,0,0)"/>  
</svg>

# 基础绘制

**矩形 <rect>**

【 x】定义矩形的左侧位置 （例如，x="0" 定义矩形到浏览器窗口左侧的距离是 0px）

【 y】定义矩形的左侧位置 （例如，y="0" 定义矩形到浏览器窗口顶端的距离是 0px）

【 stroke-width】定义矩形边框的宽度 int

【 stroke】定义矩形边框的颜色（rgb 值、颜色名或者十六进制值）

【 fill】定义矩形的填充颜色（rgb 值、颜色名或者十六进制值）

【opacity】不透明度 (范围: 0 到 1)

【fill-opacity】内容不透明度  (范围: 0 到 1)

【stroke-opacity】边框不透明度 (范围: 0 到 1)

【rx】+【ry】属性可使矩形产生圆角。

**圆形 <circle>**

【cx】+【cy】定义圆点的x和y坐标。如果省略cx和cy，圆的中心会被设置为(0, 0)

【r】定义圆的半径

**椭圆 <ellipse>**

【cx】+【cy】和圆一样

【rx】+【ry】定义的水平半径，定义的垂直半径

**线 <line>**

【x1】在 x 轴定义线条的开始

【y1】在 y 轴定义线条的开始

【x2】在 x 轴定义线条的结束

【y2】在 y 轴定义线条的结束

**多边形 - <polygon>**

【points】定义多边形每个角的 x 和 y 坐标 逗号隔开x,y 空格下一组点

**折线 - <polyline>**

【points】定义多边形每个角的 x 和 y 坐标 逗号隔开x,y 空格下一组点

**路径 - <path>**

**注释：以下所有命令均允许小写字母。大写表示绝对定位，小写表示相对定位。**

M=移动到

L=线

只有两个值 例如：L150 150

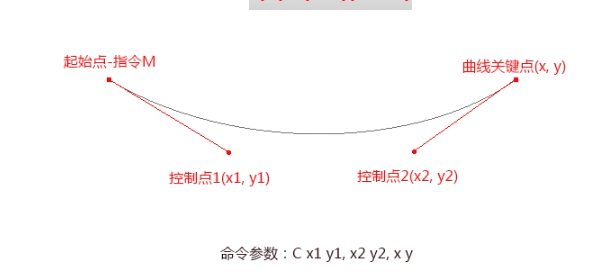
H=水平线

一个值 例如：H 180

V=垂直线

一个值 例如：V 180

C=曲线



C cp1x cp1y , cp2x cp2y , x y

参数解析如下:

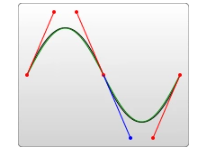
（1）.cp1x cp1y规定第一个控制点的坐标。

（2）.cp2x cp2y规定第二个控制点的坐标。

（3）. x y规定赛贝尔曲线的结束点坐标。

例如C30,30 45,45 60,30

S=平滑曲线



S x2 y2, x y

Q=平方基曲线

Q x2 y2, x y

一般用这种，其它的算不出来，不如直接生成

T=光滑的方形基础曲线

A=椭圆弧

Z=闭合路径

直接输入Z

绘制文字

<g font-size="30" font="sans-serif" fill="black" stroke="none" text-anchor="middle">

<text x="100" y="350" dx="-30">A</text>

<text x="250" y="50" dy="-10">B</text>

<text x="400" y="350" dx="30">C</text>

</g>

# SVG标签

## viewBox

比方说，我用svg画了个半径200px的圆，当把这个圆嵌入到网页里去的时候，svg的尺寸是由实际业务需要来定的，不一定是这个值，而且比例会变，这个时候就用到viewBox

viewBox="0 0 1024 1024"

# Animation

## 基本动画

【平移】

<svg width="320" height="320" xmlns="http://www.w3.org/2000/svg">  
 <g>  
 <text font-family="microsoft yahei" font-size="120" y="160" x="160">  
 平移  
 <animate attributeName="x" from="160" to="60" begin="0s" dur="3s" repeatCount="indefinite" />  
 </text>  
 </g>  
</svg>

【放大】

<svg width="320" height="320">  
 <g>  
 <text font-family="microsoft yahei" font-size="80" y="100" x="100">  
 规模从1变大到1.5  
 <animateTransform attributeName="transform" begin="0s" dur="3s" type="scale" from="1" to="1.5" repeatCount="indefinite"/>  
 </text>  
  
 </g>  
</svg>

【显隐】

<svg width="320" height="200">  
 <text font-family="microsoft yahei" font-size="120" y="160" x="160">位置透明度同时变化  
 <animate attributeName="x" from="160" to="60" begin="0s" dur="3s" repeatCount="indefinite" />  
 <animate attributeName="opacity" from="1" to="0" begin="0s" dur="3s" repeatCount="indefinite" />  
 </text>  
</svg>

## 参数详解

### attributeName = <attributeName>

     要变化的元素属性名称。

1. 可以是元素直接暴露的属性，例如，text元素上的x, y或者font-size;
2. 可以是CSS属性。例如，透明度opacity.

### attributeType = “CSS | XML | auto”

　  attributeType支持三个固定参数，CSS/XML/auto.

　　用来表明attributeName属性值的列表。x, y以及transform就属于XML, opacity就属于CSS. auto为默认值，自动判别的意思（实际上是先当成CSS处理，如果发现不认识，直接XML类别处理）。因此，如果你不确信某属性是XML类别还是CSS类别的时候，不设置attributeType值，直接让浏览器自己去判断，几乎无差错。

### from, to, by, values

　　from = “<value>“  
    　　动画的起始值。  
　　to = “<value>“  
  　　  指定动画的结束值。  
　　by = “<value>“  
    　　动画的相对变化值。  
　　values = “<list>“  
    　　用分号分隔的一个或多个值，可以看出是动画的多个关键值点。

 　　相互之间还是有制约关系的。有以下一些规则：

（不考虑values）to,by两个参数至少需要有一个出现。否则动画效果没有。to表示绝对值，by表示相对值。拿位移距离，如果from是100, to值为160则表示移动到160这个位置，但是，如果by值是160，则表示移动到100+160=260这个位置。

如果to,by同时出现，则by打酱油，只识别to.

如果to,by,values都没设置，自然没动画效果。如果任意（包括from）一个属性的值不合法，规范上说是没有动画效果。（据测试，FireFox确实如此，但是Chrome特意做了容错处理。如，本来是数值的属性，写了个诸如“a”这个不合法的值，其会当作“0”来处理，动画效果依然存在。）

values可以是一个值或多值。但在Chrome浏览器下的测试，是一个值的时候是没有动画效果。多值时候有动画效果。当values值设置并能识别时候，from, to, by的值都会被忽略。我们实现动画，不可能就是单纯的从a位置到b位置，有时候，需要去c位置过渡下。此时，实际上有3个动画关键点。而from, to/by只能驾驭两个，此时就是values大显身手的时候了！

<svg width="320" height="200" xmlns="http://www.w3.org/2000/svg">  
 <text font-family="microsoft yahei" font-size="120" y="150" x="160">来回跑  
 <animate attributeName="x" values="160;40;160" dur="3s" repeatCount="indefinite" />  
 </text>  
</svg>

总结下，也就是from-to动画、from-by动画、to动画、by动画以及values动画。

### begin, end

begin指动画开始的时间，看上去很简单。设个时间，延迟嘛~~实际上非也非也，上面出现的beigin="3s"只是最简单最基本的表示。

begin的定义是分号分隔的一组值。单值只是其中的情况之一。例如，beigin="3s;5s"表示的是3s之后动画走一下，6s时候动画再走一下（如果之前动画没走完，会立即停止从头开始）。如果一次动画时间为3s, 即dur="3s"，同时没有repeatCount属性时候，我们可以看到动画似乎连续执行了2次。

begin的单值除了普通value，还有下面这些类别的value：

offset-value | syncbase-value | event-value | repeat-value | accessKey-value | media-marker-value | wallclock-sync-value | "indefinite"

① offset-value表示偏移值，数值前面有+或-. 应该指相对于documentdocument的begin值而言。

② syncbase-value基于同步确定的值。语法为：[元素的id].begin/end +/- 时间值. 就是说借用其他元素的begin值再加加减减，这个可以准确实现两个独立元素的动画级联效果。OK，看完下面的例子一定会豁然开朗，对于上面的offset-value也会有一定的认知。后面attributeName为y的元素的begin值是x.end. x.end中的x就是上面一个animate元素的id值，而end是动画元素都有的一个属性，动画结束的时间。因此，begin="x.end"意思就是，当id为x的元素动画结束的时候，我执行动画。非常类似于PowerPoint动画的“上一个动画之后”的选项。还可以增加一些偏移值，例如begin="x.end-1s", 就表示id为x的元素动画结束前一秒开始纵向移动

<svg width="320" height="200" xmlns="http://www.w3.org/2000/svg">  
 <text font-family="microsoft yahei" font-size="120" y="160" x="160">id为x的动画结束了开始，无缝连接  
 <animate id="x" attributeName="x" to="60" begin="0s" dur="3s" fill="freeze" />  
 <animate attributeName="y" to="100" begin="x.end-1s" dur="3s" fill="freeze" />  
 </text>  
</svg>

③ event-value这个表示与事件相关联的值。类似于PowerPoint动画的“点击执行该动画”。语法是：[元素的id].[事件类型] +/- 时间值. 举个例子，点击下图的圆圈圈，马儿它就会自己跑！

<svg id="svg" width="320" height="200" xmlns="http://www.w3.org/2000/svg">  
 <circle id="circle" cx="100" cy="100" r="50"></circle>  
 <text font-family="microsoft yahei" font-size="120" y="160" x="160">点击圆圈字会跑  
 <animate attributeName="x" to="60" begin="circle.click" dur="3s" />  
 </text>  
</svg>

注意，这类与事件关联的SVG需要内联在页面中，否则click什么的都是徒劳。

④ repeat-value指重复多少次之后干嘛干嘛。语法为：[元素的id].repeat(整数) +/- 时间值. 举个例子，下面这个马儿会在水平运动2次之后，斜向运动，begin="x.repeat(2)"指id为x的元素的动画重复2次后执行~~

<svg width="320" height="200" xmlns="http://www.w3.org/2000/svg">  
 <text font-family="microsoft yahei" font-size="120" y="160" x="160">x执行2次后再执行y  
 <animate id="x" attributeName="x" to="60" begin="0s" dur="3s" repeatCount="indefinite" />  
 <animate attributeName="y" to="100" begin="x.repeat(2)" dur="3s" fill="freeze" />  
 </text>  
</svg>

⑤ accessKey-value定义快捷键。即按下某个按键动画开始。语法为：accessKey(" character "). character表示快捷键所在的字符，举个例子，按下s键动画走起。SVG代码如下：

[这个没有生效]

<svg width="320" height="200" xmlns="http://www.w3.org/2000/svg">  
 <text font-family="microsoft yahei" font-size="12" y="160" x="160">按键盘s键执行  
 <animate attributeName="x" to="60" begin="accessKey(s)" dur="3s" repeatCount="indefinite" />  
 </text>  
</svg>

⑥ wallclock-sync-value指真实世界的时钟时间定义。时间语法是基于在ISO8601中定义的语法。

⑦ "indefinite"就是这个字符串值，表示“无限等待”。据说需要beginElement()方法触发或者指向该动画元素的超链接(SVG中的a元素)。点击我们的svg, 触发animate元素的beginElement()方法，前提是begin="indefinite".

<svg id="svg" width="320" height="200" xmlns="http://www.w3.org/2000/svg">  
 <text font-family="microsoft yahei" font-size="120" y="160" x="160">触发事件后运动  
 <animate attributeName="x" to="60" begin="indefinite" dur="3s" />  
 </text>  
</svg>  
  
<script>  
 var animate = document.getElementsByTagName("animate")[0];  
 if (animate) {  
 document.getElementById("svg").onclick = function() {  
 animate.beginElement();  
 };  
 }  
</script>

a元素的xlink:href指向的我们的动画元素

<svg width="320" height="200" font-family="microsoft yahei" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">  
 <text font-size="120" y="160" x="160">点击触发<animate id="animate" attributeName="x" to="60" begin="indefinite" dur="3s" repeatCount="indefinite" />  
 </text>  
 <a xlink:href="#animate">  
 <text x="10" y="20" fill="#cd0000" font-size="30">点击我</text>  
 </a>  
</svg>

### dur

后面两种：常规时间值 | "indefinite".

“常规时间值”就是3s之类的正常值；"indefinite"指事件无限。动画时间无限，实际上就是动画压根不执行的意思。因此，设置为"indefinite"跟没有dur是一个意思，与dur解析异常一个意思。

### **calcMode, keyTimes, keySplines**

这几个参数是控制动画先快还是先慢类似这样作用的。

calcMode属性支持4个值：discrete | linear | paced | spline. 中文意思分别是：“离散”|“线性”|“踏步”|“样条”。

discrete

from值直接跳到to值。

linear

animateMotion元素以外元素的calcMode默认值。动画从头到尾的速率都是一致的。

paced

通过插值让动画的变化步调平稳均匀。仅支持线性数值区域内的值，这样点之间“距离”的概念才能被计算（如position, width, height等）。

如果”paced“指定，任何keyTimes或keySplines值都会打酱油。

spline

插值定义贝塞尔曲线。spline点的定义在keyTimes属性中，每个时间间隔控制点由keySplines定义。

keyTimes = “<list>”  
跟上面提到的<list>类似，都是分号分隔一组值。keyTimes从名字上看是关键时间点的意思，大致就是这个意思。前面提到过values也是多值，这里有一些约定的规则：

　　一、keyTimes值的数目要和values一致，如果是from/to/by动画，keyTimes就必须有两个值。

　　二、对于linear和spline动画，第一个数字要是0, 最后一个是1。

　　三、每个连续的时间值必须比它前面的值大或者相等。

 paced模式下，keyTimes会被忽略；keyTimes定义错误，也会被忽略；dur为indefinite也会被忽略。

keySplines = “<list>”  
keySplines表示的是与keyTimes相关联的一组贝塞尔控制点（默认0 0 1 1）。每个控制点使用4个浮点值表示：x1 y1 x2 y2. 只有模式是spline时候这个参数才有用，也是分号分隔，值范围0~1，总是比keyTimes少一个值。  
  
如果keySplines值不合法或个数不对，是没有动画效果的。

如下4个SVG，只展示重要部分代码：

<svg width="320" height="200" font-family="microsoft yahei" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">  
 <text font-size="12" y="20" x="0">  
 linear  
 <animate attributeName="x" dur="5s" values="0; 160" calcMode="linear" />  
 </text>  
 <text font-size="12" y="40" x="0">  
 paced  
 <animate attributeName="x" dur="5s" values="0; 160" calcMode="paced"/>  
 </text>  
 <text font-size="12" y="60" x="0">  
 linear + keyTimes  
 <animate attributeName="x" dur="5s" values="0; 80; 160" keyTimes="0; .8; 1" calcMode="linear"/>  
 </text>  
 <text font-size="12" y="80" x="0">  
 spline + keySplines  
 <animate attributeName="x" dur="5s" values="0; 80; 160" keyTimes="0; .8; 1" calcMode="spline" keySplines=".5 0 .5 1; 0 0 1 1" />  
 </text>  
</svg>

就是values, keyTimes, keySplines三个人之间事情。values确定动画的关键位置，keyTimes确定到这个关键点需要的时间，keySplines确定的是每个时间点段之间的贝塞尔曲线，也就是具体的缓动表现。平时CSS3写的transition动画效果，也是这么回事，这是values值就两个，所以，keyTimes只能是0;1, 贝塞尔曲线就只有一个，要不ease, 要不linear等

### repeatCount, repeatDur

repeatCount表示动画执行次数，可以是合法数值或者”indefinite“（动画循环到电脑死机）。  
repeatDur定义重复动画的总时间。可以是普通时间值或者”indefinite（”动画循环到电脑死机）。

动画只玩执行完整3个 + 一个1/3个动画。因为repeat总时间就10s而已。

<animate attributeName="x" to="60" dur="3s" repeatCount="indefinite" repeatDur="10s" />

### fill

fill表示动画间隙的填充方式。支持参数有：freeze | remove. 其中remove是默认值，表示动画结束直接回到开始的地方。freeze“冻结”表示动画结束后像是被冻住了，元素保持了动画结束之后的状态。

### accumulate, additive

accumulate是累积的意思。支持参数有：none | sum. 默认值是none. 如果值是sum表示动画结束时候的位置作为下次动画的起始位置。  
additive控制动画是否附加。支持参数有：replace | sum. 默认值是replace. 如果值是sum表示动画的基础知识会附加到其他低优先级的动画上，

<img ...>

<animateMotion begin="0" dur="5s" path="[some path]" additive="sum" fill="freeze" />

<animateMotion begin="5s" dur="5s" path="[some path]" additive="sum" fill="freeze" />

<animateMotion begin="10s" dur="5s" path="[some path]" additive="sum" fill="freeze" /> </img>

这里轮到第二个动画的时候，路径是从第一个动画路径结束地方开始的，于是，3个动画完美无缝连接起来了。

<animateTransform attributeName="transform" type="scale" from="1" to="3" dur="10s" repeatCount="indefinite" additive="sum"/>

<animateTransform attributeName="transform" type="rotate" from="0 30 20" to="360 30 20" dur="10s" fill="freeze" repeatCount="indefinite" additive="sum"/>;

两个动画同时都是transform，都要使用一个type属性，好在这个例子additive="sum"是累加的而不是replace替换。

于是，我们就可以是实现一边旋转一边放大的效果。

### restart

restart这个属性诞生的背景如下：很多动画呢，其触发可能与事件相关。现在，存在这种情况，希望点击一次反应一次，之后再点击就没有反应。这种需求的出现迫使restart参数的出现。支持的参数有：always | whenNotActive | never.  
  
always是默认值，表示总是，也就是点一次，触发一次事件。whenNotActive表示动画正在进行的时候，是不能重启动画的。never表示动画是一波流。

### min, max

min/max表示动画执行最短和最长时间。支持参数为时间值和"media"（媒介元素有效）, max还支持indefinite.

### 动画的暂停与播放

SVG animation中是有内置的API可以暂停和启动动画的

// svg指当前svg DOM元素

// 暂停 svg.pauseAnimations();

// 重启动 svg.unpauseAnimations()