CSS3动画帧数科学计算法

BY: bboy90 / DATE: 2014-05-21 / POSTED IN: 页面重构 / VIEWS: **1937** / COMMENTS: **6**

PLAY IN TID

CSS3动画帧数计算器

动画类型： ◉ 单向循环动画 ○ 双向循环动画 ○ 模拟双向循环动画

动作停顿： ◉ 停顿 ○ 不停顿

动作个数： 3 （最大动作数50）

计算方式： ◉ 帧数模式 ○ 时间模式

指定帧数： ◉ 动作过渡帧数 ○ 动作停顿帧数

20

生成代码

动画示例

animation: anim-name 2s linear infinite;

动作个数：3　　动作过渡帧数：20　　动作停顿帧数：20

```
@-webkit-keyframes anim-name{
    0%, 20%{  /* 动作1 */  }
    40%, 60%{  /* 动作2 */  }
    80%, 100%{  /* 动作3 */  }
}
```

```
@-webkit-keyframes anim-name{
    0%{  /* 动作1 */  }
    20%{  /* 动作1 */  }
    40%{  /* 动作2 */  }
    60%{  /* 动作2 */  }
    80%{  /* 动作3 */  }
    100%{  /* 动作3 */  }
}
```

BLOG CATEGORIES

- 交互设计
- 视觉设计
- 前端开发
- 页面重构
- 团队生活

RECENT COMMENTS

Luke：请问下深圳这百年有 重构职位需求吗？目前。。。

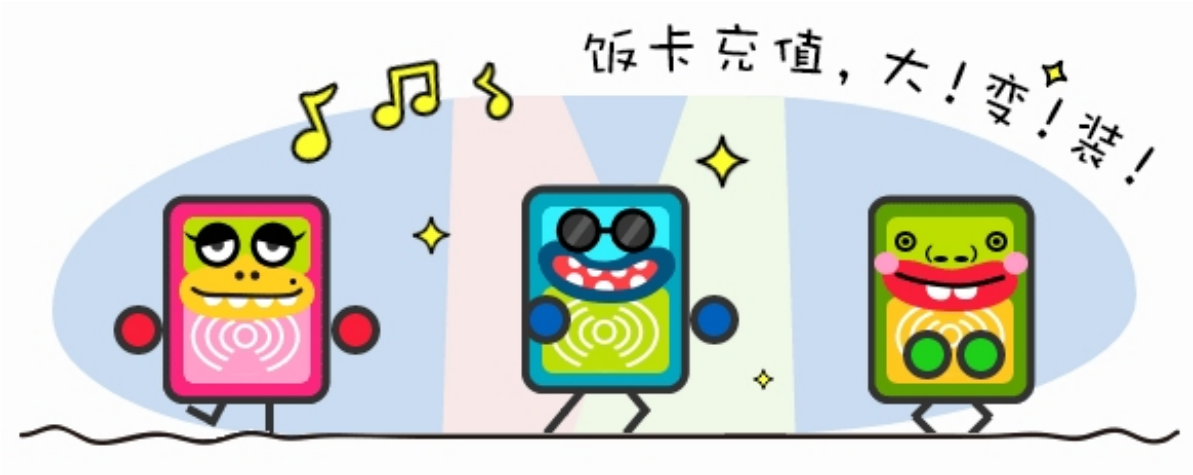
老谭：前端新来的妹纸么？ 学习了

tinno：好流弊！

总结都浓缩在这个工具里了，想知道工具的地址或想窥探工具诞生的趣事请往下看 。

————————————————————————————-　　华丽丽的开篇　　————————————————————————————-

本篇文章来自腾讯内部饭卡充值改版项目的CSS3动画经验总结。虽然你们访问不到我们的饭卡站点，不过可以小窥一下我们的动画示例哟。

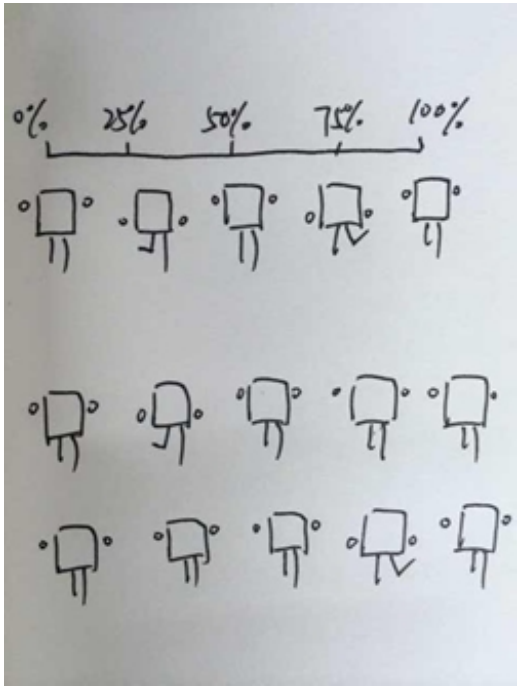


（请使用chrome、safari或firefox浏览器看效果，效果地址）

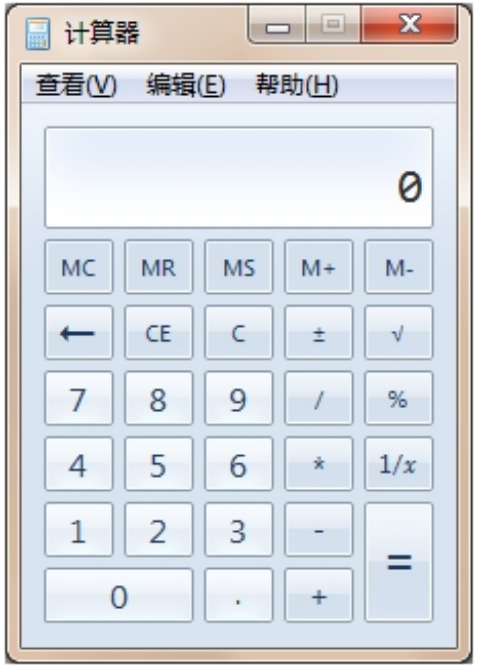
实现上面“嘀卡萌风骚乱舞”的动画，比较麻烦的是，要凭感觉自己算参数写代码，重复试个千百回，才能达到最终满意的效果。

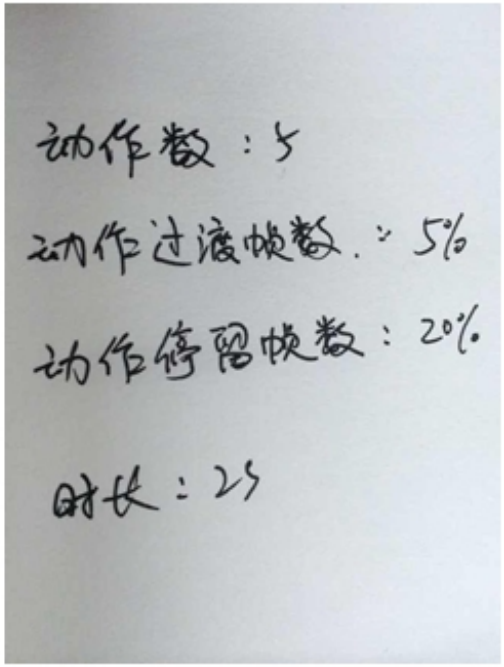
比如这个动画：

我曾经，这么干过



还这么干过





step1, 动作1在0%上，动作停留20帧

```
@keyframes anim-name{
    0%, 20%{ /* 动作1 */ }
    ...
}
```

step2，动作之间过渡5帧，动作1结束帧在20%，20+5=25, 动作2在25%帧，动作停留20帧

```
@keyframes anim-name{
    0%, 20%{ /* 动作1 */ }
```

```
    25%, 45%{ /* 动作2 */ }
    ...
}
```

……

经过一番计算后

```
@keyframes anim-name{
    0%, 20%{ /* 动作1 */ }
    25%, 45%{ /* 动作2 */ }
    50%, 70%{ /* 动作3 */ }
    75%, 95%{ /* 动作4 */ }
    100%, 120%{ /* 动作5 */ }
}
```

艾玛，帧数超出100%了>_<

重新计算了一番，动作数5，动作过渡帧数5%，动作停留帧数16%

```
@-webkit-keyframes anim-name{
    0%, 16%{  /* 动作1 */  }
    21%, 37%{  /* 动作2 */  }
    42%, 58%{  /* 动作3 */  }
```

```
    63%, 79%{  /* 动作4 */  }
    84%, 100%{  /* 动作5 */  }
}
```

感谢人民感谢党，最后一帧加起来刚好100%

刷新页面看效果之后.....（动作过渡有点快，动作停留有点长）

效果不对，重算!

效果不对，重算!

.....

就这样被折腾地体无完肤，深刻感悟我们是用生命在做动画，啊.....多么痛的领悟悟悟~~（有共鸣的，请默默的点个赞，谢谢）

所以，我们今天来探讨如何更科学的计算帧数?

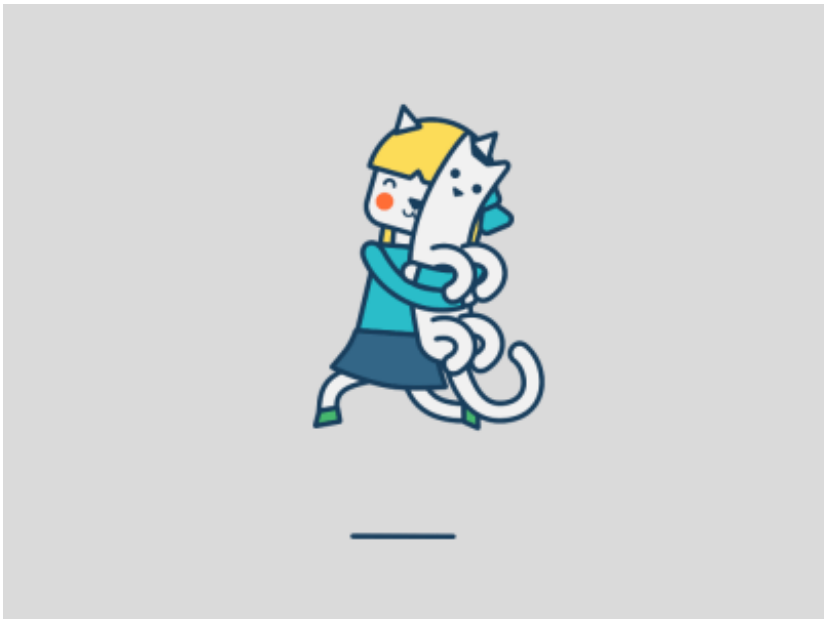
```
@keyframes anim-name{
    0% { }
    ?% { }
    100% { }
}
```

文章主要研究**循环动画**，各个动作之间的过渡**有规律性**。

比如嘀卡萌跳舞动画

走路动画

还有跑步动画



（该动画的实现，可 查看 白树同学的分享）

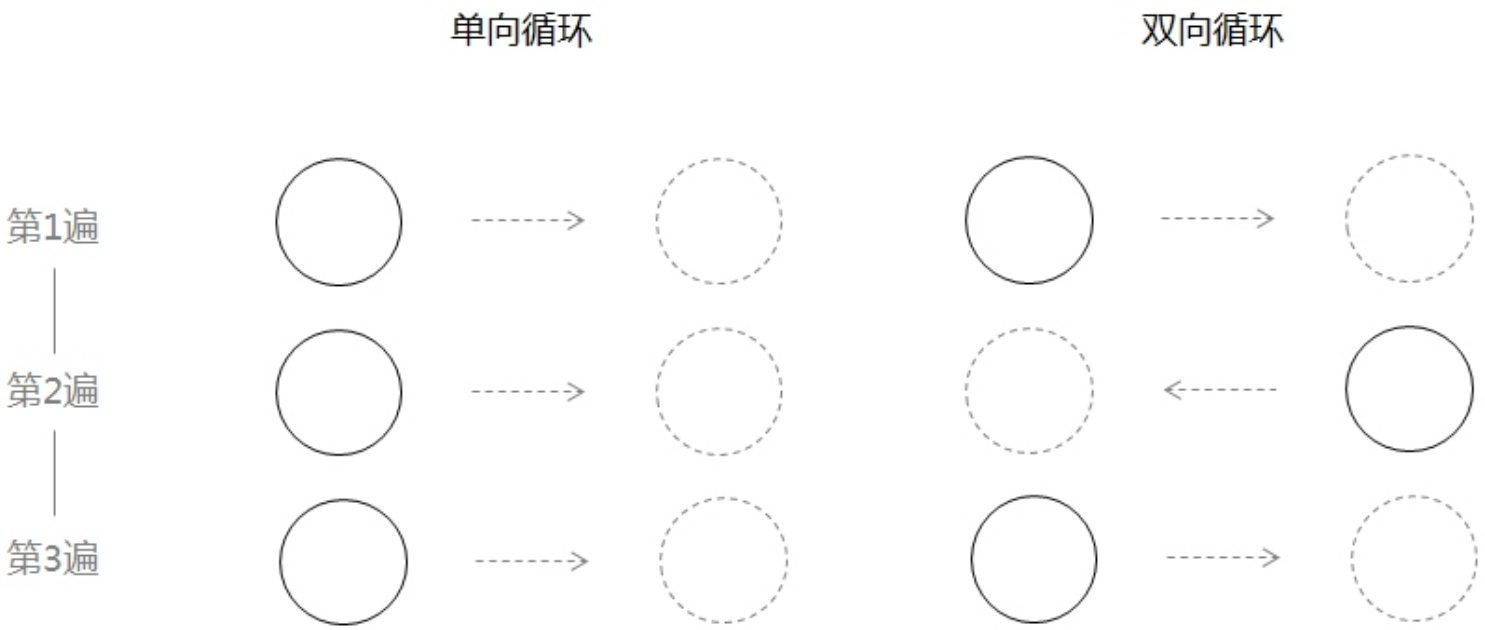
动作过渡有规律性，从代码层面也可理解为各动作之间的过渡帧数是一样的。

如上面白树同学实现的跑步动画，各动作之间的过渡帧约14.3帧，代码为

```
@keyframes anim-name{
    0% {background-position: 0 0;}
    14.3% {background-position: -180px 0;}
    28.6% {background-position: -360px 0;}
    42.9% {background-position: -540px 0;}
    57.2% {background-position: -720px 0;}
    71.5% {background-position: -900px 0;}
    85.8% {background-position: -1080px 0;}
    100% {background-position: 0 0;}
}
```

好，下面让我们愉快的进入主题吧

循环动画按循环方式可以分为：

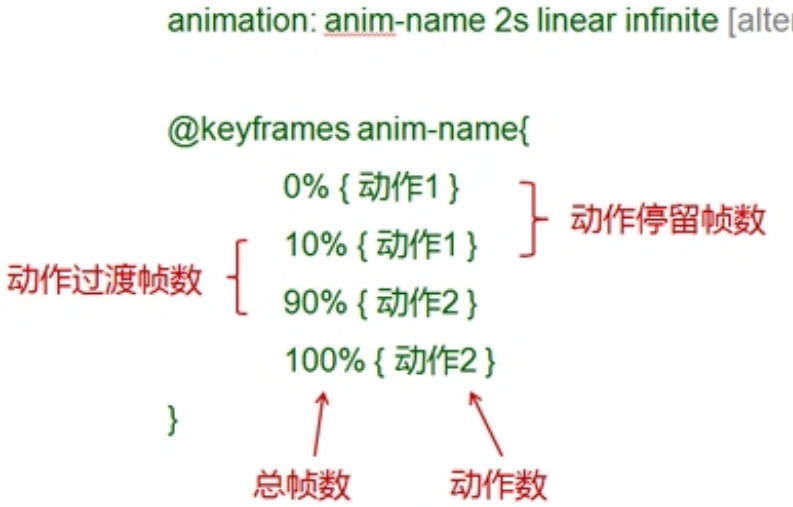


用CSS代码的方式表示，就是：

单向循环： animation-iteration-count: infinite; animation-direction: normal;

双向循环： animation-iteration-count: infinite; animation-direction: alternate;

先看看做一个动画需要哪些条件



总帧数：100 （已知参数）

CSS3帧动画的帧数设置是从0%~100%，数值可以带小数位，0%可以用from关键词替代，100%可以用to关键词替代

动作数：n （已知参数）

动画中的几个关键动作

动作停留帧数：x （未知参数）

在当前动作停留的帧数

动作过渡帧数：y （未知参数）

上一个动作过渡到下一个动作需要用的帧数

我们用示例来说明它们之间的关系。

单向循环动画

示例要求：实现一个3个动作的单向循环动画

为了方便理解，以**线段图示法**来展示

Step1，满帧100%

0% 100%

Step2，添加动作节点(总节点数 = 动作数)

0% ？% 100%

过渡y帧 过渡y帧

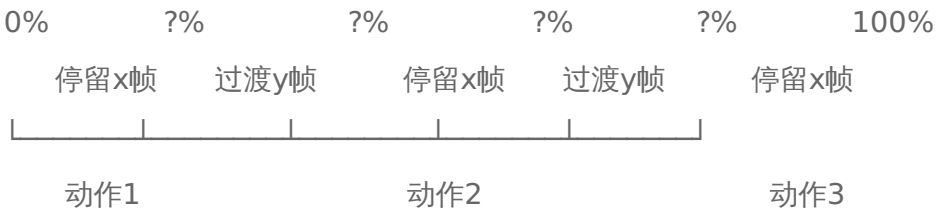
动作1 动作2 动作3

这个时候，我们很轻易的算出动作2的keyframes帧数是50%

实际上，很多时候我们需要让每个动作停顿一会，而不会闪动太快。如“嘀卡萌风骚乱舞”的动画，每个动作都需要定格

一会，这个时候我们需要给每个动作分配一些停留帧数。

*Step3，添加停留帧 (总节点数 = 动作数 * 2)*



这下就复杂了，不过我们仔细分析，会发现它们之间有一定的规律。

$$3x + 2y = 100$$

动作个数 = 3　　停留帧个数 = 3　　过渡帧个数 = 2

设动作个数为n，则

动作个数 = n　　停留帧个数 = n　　过渡帧个数 = n-1

然后，我们可以得出一个公式

$$nx + (n-1)y = 100$$

接下来我们可以有规则性的尝试动画参数了，我们尝试让每个动作停留20帧，通过公式求得动作过渡帧数y也等于20，于是得出我们的帧数代码

```
.demo{animation:anim-name 1s infinite;}  /* 单向循环 */

@keyframes anim-name{
    0%, 20%{  /* 动作1 */  }
    40%, 60%{  /* 动作2 */  }
    80%, 100%{  /* 动作3 */  }
}
```

有了公式，我们就不用瞎尝试啦，可以少死点脑细胞了

双向循环动画

示例要求：实现一个3个动作的双向循环动画

复制上面的动画代码，加个 animation-direction: alternate; 属性不就好了？

(哦，不对，按照心理学反推论，如果这么简单，作者有必要另起篇幅吗？肯定有阴谋！)

不用猜了，我就是有阴谋！

继续**线段图示**，当我们加入 animation-direction: alternate; 属性之后的效果是

问题：首尾动作从第二遍播放开始会重复停留时间！

这个并不是我们期望看到的效果，不过解决方法也很简单

通过线段图分析

$$2x + 2y = 100$$

动作个数 = 3　　停留帧个数 = 2　　过渡帧个数 = 2

设动作个数为n，则

动作个数 = n　　停留帧个数 = n-1　　过渡帧个数 = n-1

然后，我们可以得出一个公式

$$(n-1)(x+y) = 100$$

接下来我们还是尝试让每个动作停留20帧，通过公式求得动作过渡帧数y等于30，于是得出我们的帧数代码

```
.demo{animation:anim-name 1s infinite alternate;} /* 双向循环 */

@keyframes anim-name{
    0%, 10%{  /* 动作1 */  }
    40%, 60%{  /* 动作2 */  }
    90%, 100%{  /* 动作3 */  }
```

```
}
```

注意：双向循环动画，首尾动作停留帧要各减一半，示例的首尾动作停留帧为10 (20/2=10)

细心的同学会发现，其实这里还有点小瑕疵，那就是

问题：第一次播放的第一个动作只停了一半时间！

有时我们做动作衔接，一定要所有动作时间都保持一致。解决办法也不是没有，可以给动画加个延迟时间 animation-delay 属性，时长等于动作停留时间的一半，如何计算时长后面会讲到。

除了加延时解决这个问题之外，还有一个伪方法，请继续往下看

模拟双向循环动画

示例要求：实现一个3个动作的双向循环动画

模拟双向循环动画就是不使用 animation-direction: alternate; 属性实现双向循环的效果。

有点绕，上线段图

通过线段图分析

$$4x + 4y = 100$$

动作个数 = 5　　停留帧个数 = 4　　过渡帧个数 = 4

设动作个数为n，则

动作个数 = n　　停留帧个数 = n-1　　过渡帧个数 = n-1

然后，我们可以得出一个公式

$$(n-1)(x+y) = 100$$

但动作个数5包含了重复动作，不符合我们的计算习惯，不包含重复动作个数3才符合我们的计算习惯。那么设

（不含重复）动作个数为 m

（含重复）动作个数为 n，则 n = 2m-1，将 2m-1 带入上面的公式得出公式

$$(2m-1-1)(x+y) = 100$$

将m统一换成n表示，再简化公式后得到最终公式

$$(2n-2)(x+y) = 100$$

接下来我们再次尝试让每个动作停留20帧，通过公式求得动作过渡帧数y等于5，于是得出我们的帧数代码

```
.demo{animation:anim-name 1s infinite;} /* 模拟双向循环 */

@-webkit-keyframes anim-name{
    0%{  /* 动作1 */  }
    20%{  /* 动作1 */  }
    25%{  /* 动作2 */  }
    45%{  /* 动作2 */  }
```

```
    50%{  /* 动作3 */  }
    70%{  /* 动作3 */  }
    75%{  /* 动作2 */  }
    95%{  /* 动作2 */  }
    100%{  /* 动作1 */  }
}
```

缩写版代码

```
.demo{animation:anim-name 1s infinite;} /* 模拟双向循环 */

@keyframes anim-name{
    0%, 20%, 100%{  /* 动作1 */  }
    25%, 45%, 75%, 95%{  /* 动作2 */  }
    50%, 70%{  /* 动作3 */  }
}
```

模拟双向循环的方法可以让所有动作的停留时间都保持一致，缺点就是代码比较多，帧数也算得麻烦，不过也不失为一种解决方法。一般情况下，还是建议大家使用双向循环+延迟播放的方案。

提到延迟播放，跟时间有关系，这个延迟时长该怎么定？如果以上方案，每个动作我们要固定它的过渡时间，比如动作之间过渡0.4秒，那过渡帧数又该怎么定？接下来我们再挖掘一下，帧数如何跟时间结合。

时间模式计算帧数

我们在做动画的时候需要设置一个 animation-duration 动画持续时间的属性，知道持续播放时间我们就可以很轻易的计算出播放速度，还记得我们小学学的速度公式吗？

设，总帧数为s(100帧)，播放时间为t，播放速度为v，得出公式

$$v = s / t$$

继续用示例来加深理解。

示例要求：实现一个3个动作的单向循环动画，播放时间2秒，每个动作的过渡时间为0.4秒

通过播放速度公式，我们可以计算出过渡帧数。

播放速度： 100帧 / 2秒 = 50帧/秒
过渡帧数： 50帧/秒 * 0.4秒 = **20帧**

得出过渡帧数，接下来套用单向循环动画的帧数公式，计算出停留帧数，参考上面总结的公式 $nx + (n-1)y = 100$ ，推导公式得出停留帧数 $x = (100-(n-1)y) / n$

动作个数(n)： 3

过渡帧数(y)： 20

停留帧数： (100-(3-1)*20)/3 = **20帧**

于是得出我们的帧数代码

```
.demo{animation:anim-name 2s infinite;}  /* 单向循环 */

@keyframes anim-name{
    0%, 20%{  /* 动作1 */  }
    40%, 60%{  /* 动作2 */  }
    80%, 100%{  /* 动作3 */  }
}
```

这么多公式，眼都花了更别说记了。别着急，公式是给机器记的，这种破事就交给我们的机器去算。下面是一个简易的CSS3动画帧数计算器，可以帮我们省去一些计算的烦恼。

CSS3动画帧数计算器：http://tid.tenpay.com/labs/css3_keyframes_calculator.html

CSS3动画帧数计算器

动画类型： (◉) 单向循环动画 ( ) 双向循环动画 ( ) 模拟双向循环动画

动作停顿： (◉) 停顿 ( ) 不停顿

动作个数： 3 （最大动作数50）

计算方式： (◉) 帧数模式 ( ) 时间模式

指定帧数： (◉) 动作过渡帧数 ( ) 动作停顿帧数

20

生成代码

动画示例

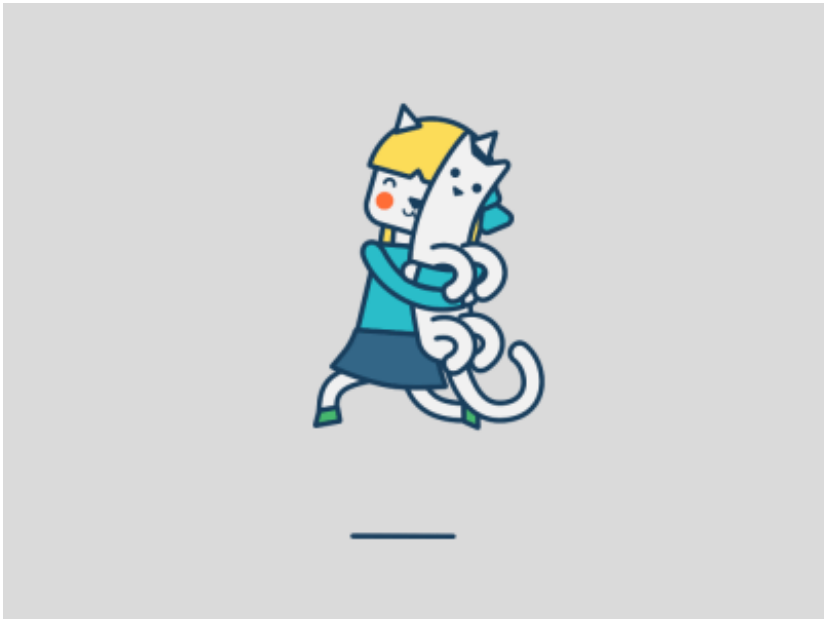
animation: anim-name 2s linear infinite;

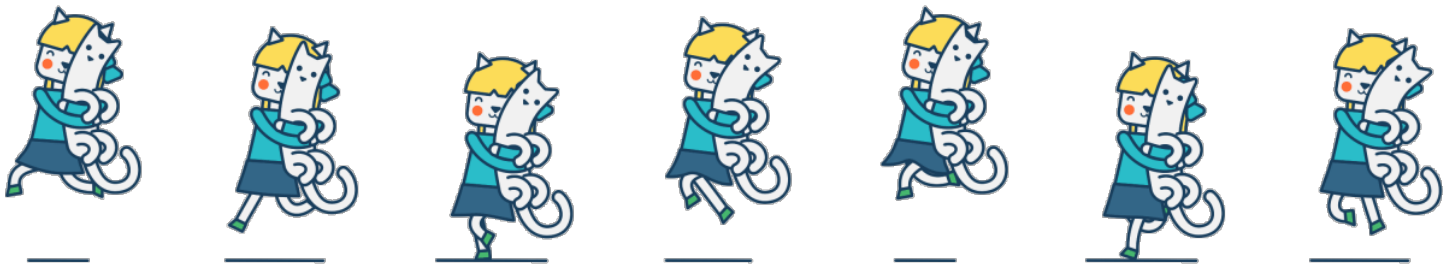
动作个数：3　　动作过渡帧数：20　　动作停顿帧数：20

```
@-webkit-keyframes anim-name{
    0%, 20%{  /* 动作1 */  }
    40%, 60%{  /* 动作2 */  }
    80%, 100%{  /* 动作3 */  }
}
```

```
@-webkit-keyframes anim-name{
    0%{  /* 动作1 */  }
    20%{  /* 动作1 */  }
    40%{  /* 动作2 */  }
    60%{  /* 动作2 */  }
    80%{  /* 动作3 */  }
    100%{  /* 动作3 */  }
}
```

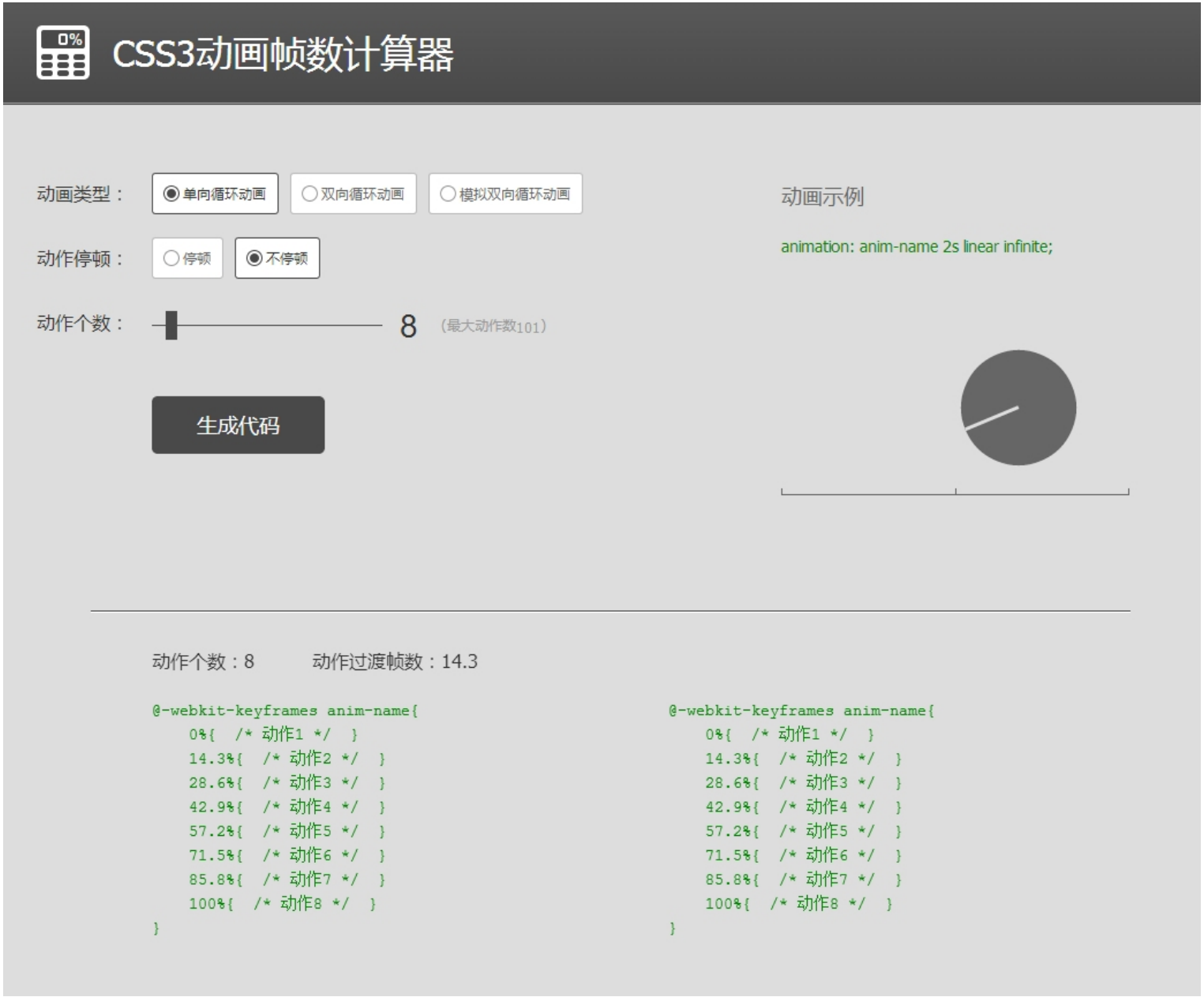
以白树同学的跑步动画为示例





动画是单向循环，有7个关键动作，动作需要使用逐帧过渡效果 animation-timing-function:step-start 实现，所以动作个数需要额外加1，即有8个动作。使用 step-start 后会自动平分动作停留时间，所以keyframes我们就不用加动作停留帧数了。

打开工具页面，选择 [单向循环动画] -> [不停顿] -> [动作个数8] -> [生成代码]



最后……就没有最后了，欢迎大家一起交流探讨。

TAGS: animation, css3, css3动画, keyframes, 网页重构

< 上一篇 创意力量从同理设计出发

从找索引浅谈性能优化 下一篇 >

RECENT COMMENTS 共有6 条评论(写评论)

css3China 2014-05-21
高大上的设计，顶顶顶

回复

白树 2014-05-21
先杀伐个，再赞~

回复

毛酱队员 2014-05-21
90大赞！

回复

andge 2014-05-21
赞

回复

灯盏细辛 2014-05-27
很折腾！赞！

回复

tinno 2014-06-18
好流弊！

回复

Leave a Reply

Name

Email (will not be published)

Website

发表