如何使用Tween.js各类原生动画运动缓动算法

这篇文章发布于 2016年12月19日，星期一，01:51，归类于 JS实例。 阅读 62787 次, 今日 35 次 37 条评论

一、速战速决

昨天有事回了趟江苏，一来一回还堵车，1天基本上就在路上了，下午又当司机送夫人去买东西，周末工作时间严重不足，原本要早产的文章估计又要晚产了，为了争取2点前写完，我就尽量少废话了。

二、关于Tween.js

Tween.js是一个包含各种经典动画算法的JS资源，之前在多篇文章有提到过，例如之前写的“JavaScript与元素间的抛物线轨迹运动”，AS中甚至有专门的Tween类。

我在自己的Github上放了有3年了，地址为：https://github.com/zhangxinxu/Tween

我自己悄悄瞅了一看，就一个JS文件，连个描述都没有的的项目居然有147个star，看来今年用力一把，上500 star指日可待。

代码截图如下：

177 lines (177 sloc) | 6.25 KB

```
/*
 * Tween.js
 * t: current time（当前时间）；
 * b: beginning value（初始值）；
 * c: change in value（变化量）；
 * d: duration（持续时间）。
 * you can visit 'http://easings.net/zh-cn' to get effect
*/
var Tween = {
    Linear: function(t, b, c, d) { return c*t/d + b; },
    Quad: {
        easeIn: function(t, b, c, d) {
            return c * (t /= d) * t + b;
        },
        easeOut: function(t, b, c, d) {
            return -c *(t /= d)*(t-2) + b;
        },
        easeInOut: function(t, b, c, d) {
            if ((t /= d / 2) < 1) return c / 2 * t * t + b;
            return -c / 2 * ((--t) * (t-2) - 1) + b;
        }
    },
    Cubic: {
        easeIn: function(t, b, c, d) {
            return c * (t /= d) * t * t + b;
        },
```


`Quad`，`Cubic`等等都是经典的动画运动算法名称，完整列表如下：

1. **Linear**：线性匀速运动效果；
2. **Quadratic**：二次方的缓动（t^2）；
3. **Cubic**：三次方的缓动（t^3）；
4. **Quartic**：四次方的缓动（t^4）；
5. **Quintic**：五次方的缓动（t^5）；
6. **Sinusoidal**：正弦曲线的缓动（sin(t)）；
7. **Exponential**：指数曲线的缓动（2^t）；
8. **Circular**：圆形曲线的缓动（sqrt(1-t^2)）；
9. **Elastic**：指数衰减的正弦曲线缓动；
10. **Back**：超过范围的三次方缓动（(s+1)*t^3 – s*t^2）；
11. **Bounce**：指数衰减的反弹缓动。

每个效果都分三个缓动方式，分别是：

- **easeIn**：从0开始加速的缓动，也就是先慢后快；
- **easeOut**：减速到0的缓动，也就是先快后慢；
- **easeInOut**：前半段从0开始加速，后半段减速到0的缓动。

很多小伙伴`easeIn`和`easeOut`哪个先快，哪个先慢一直记不清楚，我这里再给大家传授一遍我独门的邪恶记法，想想我们第一次OOXX，是不是进去(`easeIn`)的时候都是先慢，等进去了就快了；然后出来(`easeOut`)的时候，开始很快，都要出来了恋恋不舍速度就慢了。跟我们这里的动画效果是完全匹配的。

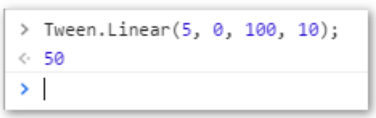
所有的这些缓动算法都离不开下面4个参数，`t`，`b`，`c`，`d`，含义如下：

```
/*
 * t: current time（当前时间）；
 * b: beginning value（初始值）；
 * c: change in value（变化量）；
 * d: duration（持续时间）。
*/
```

只看上面字面意思其实不好理解，我们套用最简单的线性匀速运动来解释下：

```
Tween.Linear = function(t, b, c, d) {
    return c*t/d + b;
}
```

比方说我们要从位置`0`的地方运动到`100`，时间是`10`秒钟，此时，`b`，`c`，`d`三个参数就已经确认了，`b`初始值就是`0`，变化值`c`就是`100-0`就是`100`，最终的时间就是`10`，此时，只要给一个小于最终时间`10`的值，`Tween.Linear`就会返回当前时间应该的坐标，例如，假设此时动画进行到第5秒，也就是`t`为5，则得到（截图自Chrome控制台）：



跟我们心中所想的值是一样的，这就是这些缓动算法的运算原理。

对了，貌似`Elastic`和`Back`有其他可选参数，但我还没时间去研究，所以，这里暂不做相关介绍。

三、如何实际使用Tween.js中的缓动算法?

上面示意的Tween.js中的线性匀速运动案例实际上只是某一个静态数值，是无法构建动画的，如果要想实现连续的具有明显轨迹的动画效果，我们需要不停地修改 `t` 的数值，一般来讲都是一直往 `d` 的数值线性靠拢即可。

这里有个动词“不停地修改”，换句话说就是不停地绘制，于是，想到了HTML5中的 `requestAnimationFrame`，关于 `requestAnimationFrame` 我之前专门有文章介绍，如果浏览器不支持 `requestAnimationFrame`，我们使用传统的 `setTimeout` 定时器兼容实现即可。

```
// requestAnimationFrame的兼容处理
if (!window.requestAnimationFrame) {
    requestAnimationFrame = function(fn) {
        setTimeout(fn, 17);
    };
}
```

因此，我们要显示一个动画效果，例如，还是拿上面的线性效果举例，则代码可以变成：

```
var t = 0, b = 0, c = 100, d = 10;
var step = function () {
    // value就是当前的位置值
    // 例如我们可以设置DOM.style.left = value + 'px'实现定位
    var value = Tween.Linear(t, b, c, d);
    t++;
    if (t <= d) {
         // 继续运动
         requestAnimationFrame(step);
    } else {
        // 动画结束
    }
};
```

基本上，所有的动画使用都是这个套路。

然后，为了让大家可以直观体验Tween.js中所有缓动算法的效果是怎样的，我特意制作了一个包含完整效果的演示页面，您可以狠狠地点击这里：Tween.js动画算法使用示意demo

点击demo页面颜色不太好看的小圆球，就会看到各自的运动速率和缓动状态了，例如，`Bounce.easeOut` 的效果就是小球像皮球落地一样弹几下：

demo页面上展示的源代码就是处理后相当精简的使用Tween.js的核心JS代码，如果大家对完整的效果实现感兴趣，可以右键页面→查看页面源代码。

四、基于Tween.js更简单调用的animation.js

Tween.js虽然原始且效果强大，但是，唯一的问题就是使用不方便，每次一个动画我都要弄个 `requestAnimationFrame`，而且4个参数有点多，不好记忆，顺序什么的一旦弄错就很麻烦，有没有什么简单的方法调用的，就像jQuery的 `animation()` 方法一样。

出于这需求，我就手不停蹄弄出了一个更容易调用的animation.js，目前已经一起放在了https://github.com/zhangxinxu/Tween这个项目上，语法如下：

```
Math.animation(form, to, duration, easing, callback);
```

其中：

- `form`和`to`是必须参数，表示动画起始数值和结束数值；
- `duration`，`easing`，`callback`理论上都是可选参数，但是实际上`callback`肯定是要使用的，因为实时变化的数值就是通过`callback`返回的。然后，`duration`，`easing`，`callback`这3个参数的顺序是任意的。具体来讲：
  - `duration`为动画持续时间，默认`300`，默认单位是毫秒，建议使用数值，例如`600`，也支持带单位，例如`600ms`或者`0.6s`；
  - `easing`为缓动的类型，字符串类型，源自Tween.js。例如：`'Linear'`，`'Quad.easeIn'`，`'Bounce.easeInOut'`等等，需要注意大小写。 其中，默认值是`'Linear'`；
  - `callback`为回调函数，支持2个参数（value, isEnding），其中`value`表示实时变化的计算值，`isEnding`是布尔值，表示动画是否完全停止。

所以，如果我们使用`Math.animation()`方法实现上面的线性运动效果则是：

```
Math.animation(0, 100, 170, function (value) {
    // value就是当前的位置值
});
```

是不是更容易理解和记忆了！

示意demo页面也有animation.js使用示意，其代码如下：

```
Math.animation(0, 800 - 42, function (value) {
    ball.style.transform = 'translateX(' + value + 'px)';
}, 'Bounce.easeInOut', 600);
```

补充于2017-01-22

基于animation.js实现了一个更复杂的动画交互效果，名为“小兔子，跳火球；腿儿短，眼泪流”，gif截屏动画如下：

亲自感受您可以狠狠地点击这里：兔子跳火球动画demo

代码细节参见上demo页面源代码。

五、结束语

Tween.js的强大之处在于，其本质上是一个算法，也就是在任何地方其实都是可以使用的，比方说canvas中或者SVG动画实现等等，可以很好弥补CSS3 `animation` 不太方便使用的场景，以及一些与动态位置打交道的交互效果，位置是不确定的，只能借助JS实现，此时配合动画算法，各种效果实现都不在话下。

有了上面的 `Math.animation()` 方法，实现不要太简单，且不依赖任何jQuery, Zepto之类的工具类JS，我们悄悄地实现，让设计师和产品经理惊讶下，喜出望外一下，岂不美哉！

比方说返回顶部效果，虽然说，直接瞬间到顶部也能满足功能，但是效果而言太干了，都如果我们主动加个动画效果，岂不是可以好好装逼一把，例如，在本文的demo演示页面，滚动到合适位置，然后打开控制台中粘贴下面JS代码然后回车：

```
Math.animation(document.documentElement.scrollTop, 0, function (value) {
    document.documentElement.scrollTop = value;
}, 'Quart.easeOut', 600);
```

就会发现页面滚动条好像自带了刹车平滑滚动到了顶部。

animation.js写得相当匆忙，时间有限，也并未详尽测试，因此如果在使用时候发现问题，欢迎及时反馈，也更加欢迎共同建设，项目地址是：https://github.com/zhangxinxu/Tween

比方说，增加loop循环控制之类的~

恩，就这些，还有13分钟2点，写个摘要差不多赶在计划前完成，速度还算不错。

最后，感谢阅读！

（本篇完） // 想要打赏？点击这里。有话要说？点击这里。

« 小tips:了解CSS/CSS3原生变量var

web移动端浮层滚动阻止window窗体滚动JS/CSS处理 »

猜你喜欢

- canvas 2D炫酷动效的实现套路和需要的技术积累
- CSS3动画那么强，requestAnimationFrame还有毛线用？
- CSS scroll-behavior和JS scrollIntoView让页面滚动平滑
- jQuery-火焰灯效果导航菜单
- 分享一个即插即用的私藏缓动动画JS小算法
- 小折腾：JavaScript与元素间的抛物线轨迹运动
- 翻译 - 解释JavaScript的“预解析(置顶解析)”
- 翻译-高质量JavaScript代码书写基本要点

- 翻编-JavaScript有关的10个怪癖和秘密
- JS中的柯里化(currying)
- 前端设计必会技能-gif动画图片制作

分享到：1

标签： animation, requestAnimationFrame, Tween类, 函数, 动画, 滚动, 缓动

发表评论（目前37条评论）

名称 (必须)

邮件地址(不会被公开) (必须)

网站

提交评论

1. **gaollard说道：**
2018年11月19日 08:59

张老师，我个人觉得文中部分代码使用不够恰当哈。以下的代码并不能保证莫动画的持续时间为 10s：

```
var t = 0, b = 0, c = 100, d = 10;
var step = function () {
// value就是当前的位置值
// 例如我们可以设置DOM.style.left = value + ‘px’实现定位
var value = Tween.Linear(t, b, c, d);
t++;
if (t <= d) {
// 继续运动
requestAnimationFrame(step);
} else {
// 动画结束
}
};
```

应该改为：

```
var from = 0, currentTime = 0, to = 300, duration = 1000;
var box = document.querySelector('.box')
var step = function () {
var value = moveJS.linear(from, to, duration, currentTime);
currentTime = currentTime + 1000/60;
box.style['marginLeft'] = value + 'px'
if (currentTime <= duration) {
requestAnimationFrame(step);
} else {
console.log('动画结束')
}
};
```

回复

2. **江湖不染风尘泪说道：**
2018年08月13日 16:42

赞一个！

回复

3. **firstBlood说道：**
2018年05月29日 09:07

最早我也是用tween.js来做H5动画, 后来无意间接触到了Mo.js 这玩意儿支持类似于css3里面自定义动画过程的效果(我一般是在ai里面先拖出过程曲线再放到js里面), 但是某次使用后发现android手机有很明显的卡顿 这个体验相当不好. 然后最近还发现在比较极限的情况下api里面的方法居然不执行, 比如短时间内连续new几次, 相当蛋疼, 张大大是否可以讨论一下呀 我现在也想找个比较完美的解决方案

回复

4. **kimpyoon说道：**
2018年04月28日 18:28

我有一个canvas，给它加了你这个animation，改变它的transform：rotateZ，来到旋转。Chrome手机调试器流畅，但是在真机上浏览就会卡帧，请问张大佬是什么问题

回复

> **张 鑫旭说道：**
> 2018年04月28日 19:12
>
> 是设置canvas元素的transform吗？试试本身绝对定位~
>
> 回复
>
> > **kimpyoon说道：**
> > 2018年05月2日 09:32
> >
> > 是的，canvas元素本身设置了绝对定位，用Quad.easeInOut会卡得明显，Quad.easeOut就好点，是不是运算量大导致耗资源大，我就做个转盘效果。
> >
> > 回复
> >
> > > **张 鑫旭说道：**
> > > 2018年05月2日 21:20
> > >
> > > 应该不至于，常规动画，canvas性能开销并不算高。
>
> > **kimpyoon说道：**
> > 2018年05月2日 15:45
> >
> > 是否可以用webworker来处理函数的计算
> >
> > 回复
>
> > **Kimpyoon说道：**
> > 2018年05月3日 22:56
> >
> > 已经发你邮箱了，请您抽空看一下，帮我分析下问题，谢谢大佬了。
> >
> > 回复

5. **jk说道：**
2018年01月27日 14:16

张大大，有个地方不知道是不是我没理解清楚你的意思。

文章此处： 所有的这些缓动算法都离不开下面4个参数，t, b, c, d，含义如下： xxxxxx

这里我觉得 t 的含义并不是当前时间，而是当前时间减去开始的时间(单位是毫秒)， 也就是从开始动画到此时此刻，过了多少时间。

回复

6. **长江长江我是黄河说道：**
2018年01月24日 11:31

段子手啊，很是黄

回复

7. **luogege说道：**
2017年06月22日 15:38

没有ooxx经验的人，没法理解啊。

回复

8. **omo说道：**
2017年06月7日 18:18

鑫个技术没的说，这个黄段子就实在……easeIn easeOut都能扯个黄段子，查下ease的意思不就得了……

回复

9. **laijbin说道：**
2017年05月12日 16:49

哈哈，发现了错别字，参数是from，不是form吧

回复

10. **jj菜说道：**
2017年04月15日 12:06

github上4000星那个tween.js，是超你这个吧？？？

回复

> **nszbf说道：**
> 2017年08月8日 11:46
>
> 嘘嘘
>
> 回复

11. **鼻子小JJ大说道：**
2017年02月22日 17:11

这是唯一一篇让我JJ看石硬的技术文章

回复

12. **bestRenekton说道：**
2017年02月13日 10:51

很小很强大，非常好用

回复

13. **loki说道：**
2017年02月7日 09:51

老司机 记忆法好赞！

回复

14. 黑猫警长说道：
2016年12月30日 10:52

短小精悍，比喻形象恰当。另：分享一些Angular2 H5方面的实践，可以吗?

回复

15. yeatszhang说道：
2016年12月25日 23:08

记忆法好顶赞

回复

16. tianxin说道：
2016年12月22日 16:45

适合我

回复

17. JSHacker说道：
2016年12月22日 11:23

被你ooxx那段的描述吸引到了

回复

18. 明成说道：
2016年12月21日 13:49

使用的时候发现一个问题：如果页面切换或者最小化的时候，运动会暂停。这是什么情况呢?

回复

> 张 鑫旭说道：
> 2016年12月21日 20:36
>
> requestAnimationFrame跟CSS3渲染使用的一个东西，所以浏览器最小化时候会关闭执行以节约性能。
>
> 回复

19. joyjoe说道：
2016年12月21日 01:13

easeIn easeOut 邪恶记忆法 完全看不懂呀 张老师果然是老司机呀

回复

20. 吴英凤说道：
2016年12月20日 23:21

张老师可不可以在博客上有上一篇和下一篇的链接，这样阅读方便

回复

21. 叔叔说道：
2016年12月20日 17:20

这个缓动算法的t，b，c，d四个参数，显然无法精确控制动画时间。完美的缓动算法只有一个参数。

回复

张 鑫旭说道：
2016年12月21日 20:37

所以，才有后面的animation.js

回复

> 叔叔说道：
> 2016年12月24日 10:44
>
> “所以”个啥？你根本就没明白我的意思。完美的算法是 当前时间/总时间=当前距离/总距离，时间是要自己算。这个tween缓动算法是给小白无脑用的。我说完美的算法是easing.js，只有1个参数的那种。
>
> 回复
>
> > 瞎扯淡说道：
> > 2017年03月7日 18:37
> >
> > 虾扯淡
> >
> > meepo说道：
> > 2017年04月6日 09:06
> >
> > 你说的easing.js是哪个？ 给个链接
> >
> > Felixzcn说道：
> > 2017年12月12日 16:38
> >
> > 难道说的是这个？https://gist.github.com/gre/1650294
> >
> > zilong说道：
> > 2018年04月18日 12:16
> >
> > t，和d的意思应该是 动画的当前次数和总次数的意思吧 ，这样理解对不对？

22. ku说道：
2016年12月19日 14:57

拜读

回复

23. 前端开发博客说道：
2016年12月19日 09:05

不错哦。原生js可以使用这个库来实现各种动画加速效果，收藏

回复

24. demo2016说道：
2016年12月19日 09:03

旧文章 不好玩

回复

最新文章
» 常见的CSS图形绘制合集
» 粉丝群第1期CSS小测点评与答疑
» 分享三个纯CSS实现26个英文字母的案例
» 小tips: 纯CSS实现打字动画效果
» CSS/CSS3 box-decoration-break属性简介
» CSS :placeholder-shown伪类实现Material Design占位符交互效果

» 从天猫某活动视频不必要的3次请求说起
» CSS vector-effect与SVG stroke描边缩放
» CSS ::backdrop伪元素是干嘛用的?
» 周知：CSS -webkit-伪元素选择器不再导致整行无效

今日热门

» 常见的CSS图形绘制合集 (190)
» 未来必热：SVG Sprite技术介绍 (119)
» 粉丝群第1期CSS小测点评与答疑 (115)
» HTML5终极备忘大全（图片版+文字版） (93)
» 让所有浏览器支持HTML5 video视频标签 (86)
» Selectivizr-让IE6~8支持CSS3伪类和属性选择器 (82)
» CSS3下的147个颜色名称及对应颜色值 (79)
» 视区相关单位vw, vh..简介以及可实际应用场景 (76)
» 写给自己看的display: flex布局教程 (76)
» 小tips: 纯CSS实现打字动画效果 (76)

今年热议

» 《CSS世界》女主角诚寻靠谱一起奋斗之人 (76)
» 不借助Echarts等图形框架原生JS快速实现折线图效果 (64)
» 看，for..in和for..of在那里吵架！ (60)
» 是时候好好安利下LuLu UI框架了！ (47)
» 原来浏览器原生支持JS Base64编码解码 (35)
» 妙法攻略：渐变虚框及边框滚动动画的纯CSS实现 (33)
» 炫酷H5中序列图片视频化播放的高性能实现 (31)
» CSS scroll-behavior和JS scrollIntoView让页面滚动平滑 (30)
» windows系统下批量删除OS X系统.DS_Store文件 (26)
» 写给自己看的display: flex布局教程 (26)

猜你喜欢

- canvas 2D炫酷动效的实现套路和需要的技术积累
- CSS3动画那么强，requestAnimationFrame还有毛线用？
- CSS scroll-behavior和JS scrollIntoView让页面滚动平滑
- jQuery-火焰灯效果导航菜单
- 分享一个即插即用的私藏缓动动画JS小算法
- 小折腾：JavaScript与元素间的抛物线轨迹运动
- 翻译 - 解释JavaScript的“预解析(置顶解析)”
- 翻译-高质量JavaScript代码书写基本要点
- 翻编-JavaScript有关的10个怪癖和秘密
- JS中的柯里化(currying)
- 前端设计必会技能-gif动画图片制作