jQuery 效果 – 滑动 jQuery 效果 – 停止动画

jQuery 效果- 动画

jQuery animate() 方法允许您创建自定义的动画。

jQuery 动画实例

jQuery

jQuery 动画 - animate() 方法

jQuery animate() 方法用于创建自定义动画。

语法：

```
$(selector).animate({params},speed,callback);
```

必需的 params 参数定义形成动画的 CSS 属性。

可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是动画完成后所执行的函数名称。

下面的例子演示 animate() 方法的简单应用。它把 元素往右边移动了 250 像素：

实例

```
$("button").click(function(){
$("div").animate({left:'250px'});
});
```

尝试一下 »

默认情况下，所有 HTML 元素都有一个静态位置，且无法移动。
如需对位置进行操作，要记得首先把元素的 CSS position 属性设置为 relative、fixed 或 absolute！

jQuery animate() - 操作多个属性

请注意，生成动画的过程中可同时使用多个属性：

实例

```
$("button").click(function(){
$("div").animate({
left:'250px',
opacity:'0.5',
height:'150px',
width:'150px'
```

```
});
});
```

尝试一下 »

可以用 animate() 方法来操作所有 CSS 属性吗？

是的，几乎可以！不过，需要记住一件重要的事情：当使用 animate() 时，必须使用 Camel 标记法书写所有的属性名，比如，必须使用 paddingLeft 而不是 padding-left，使用 marginRight 而不是 margin-right，等等。

同时，色彩动画并不包含在核心 jQuery 库中。

如果需要生成颜色动画，您需要从 jquery.com 下载 颜色动画 插件。

jQuery animate() - 使用相对值

也可以定义相对值（该值相对于元素的当前值）。需要在值的前面加上 += 或 -=：

实例

```
$("button").click(function(){
$("div").animate({
left:'250px',
height:'+=150px',
width:'+=150px'
});
});
```

尝试一下 »

jQuery animate() - 使用预定义的值

您甚至可以把属性的动画值设置为 "show"、"hide" 或 "toggle"：

实例

```
$("button").click(function(){
$("div").animate({
height:'toggle'
});
});
```

尝试一下 »

jQuery animate() - 使用队列功能

默认地，jQuery 提供针对动画的队列功能。

这意味着如果您在彼此之后编写多个 animate() 调用，jQuery 会创建包含这些方法调用的"内部"队列。然后逐一运行这些 animate 调用。

实例 1

```
$("button").click(function(){
var div=$("div");
div.animate({height:'300px',opacity:'0.4'},"slow");
div.animate({width:'300px',opacity:'0.8'},"slow");
div.animate({height:'100px',opacity:'0.4'},"slow");
div.animate({width:'100px',opacity:'0.8'},"slow");
});
```

尝试一下 »

下面的例子把 元素往右边移动了 100 像素，然后增加文本的字号：

实例 2

```
$("button").click(function(){
var div=$("div");
div.animate({left:'100px'},"slow");
div.animate({fontSize:'3em'},"slow");
});
```

尝试一下 »

← jQuery 效果 – 滑动 jQuery 效果 – 停止动画 →

2 篇笔记

写笔记

队列操作

jquery中有一个Queue队列的接口，这个模块没有单独拿出来作为一个章节是因为这个是内部专门为动画服务的，Queue队列如同data数据缓存与Deferred异步模型一样，都是jQuery库的内部实现的基础设施

Queue队列

队列是一种特殊的线性表，只允许在表的前端（队头）进行删除操作（出队），在表的后端（队尾）进行出入操作（入队），队列的特点是先进先出，最先插入的元素最先被删除。

为什么要引入队列

```
var a = 1;
setTimeout(function(){
    a=2;
},0)
alert(a);
```

我们一直习惯于线性的编写代码逻辑，但是在JavaScript编程几乎总是伴随着异步操作：

setTImeout，css3Transition/Animation,ajax,dom的绘制，postmessage，web Database 等等，大量异步操作所带来的回调函数会把我们的算法分解，**对于“异步+回调”的模式，怎么“拉平”异步操作使之跟同步一样，因为异步操作进行流程控制的时候无非避免的要嵌套大量的回调逻辑，所以就会出现 promises 约定了。**

那么 jQuery 引入队列其实从一个角度上可以认为：**允许一系列函数被异步地调用而不会阻塞程序。**

看一个代码段：

```
$("#Aaron").slideUp().fadeIn()
```

这是 jQuery 的一组动画链式序列，它的内部其实就是一组队列 Queue，所以队列和 Deferred 地位类似，是一个内部使用的基础设施。

- 当 slideUp 运行时，fadeIn 被放到 fx 队列中
- 当 slideUp 完成后，从队列中被取出运行

Queue 函数允许直接操作这个链式调用的行为，同时 Queue 可以指定队列名称获得其他能力而不局限于 fx 队列。

jQuery 提供了 2 组队列操作的 API：

```
jQuery.queue/dequeue
jQuery.fn.queue/dequeue
```

但是不同与普通队列定义的是：

- jQuery.queue 和 jQuery.fn.queue 不仅执行出队操作返回队头元素，还会自动执行返回的队头元素
- fn 是扩展在原型上的高级API是提供给实例使用的
- .queue/.dequeue 其内部是调用的 .queue，.dequeue 静态的底层方法实现入列与出列

Yunhero 12个月前 (03-23)

动画调度

对于 jQuery 的动画的设计我们要分 2 个层面理解：

1. 每一个动画效果可以看作一个独立的动画对象，每个对象都实现了针对自己这个动画的生命周期的控制
2. 动画对象与动画对象之间其实是没有直接关系，但是为了做到连续调用就需要引入一套队列机制也就是 Queue 来控制对象之间的转换的控制

动画的源码：

```
animate: function(prop, speed, easing, callback) {
   doAnimation = function() {
      var anim = Animation(this, args, optall);
   };
   this.queue(optall.queue, doAnimation);
}
```

这个代码缩减了，但是我们上面提到的最重要的 2 点这里都涉及到了：通过 queue 调度动画的之间的衔接，Animation 方法执行单个动画的封装。

jQuery 在 queue 的调度上涉及了一个关键的处理：同步与异步代码同时执行，同步收集动画序列，异步调用序列，看看整个调用的流程是这样的：

1. 通过多个 animate 方法形成动画链，那么这个动画链其实都是会加入到 queue 队列里面
2. 在每一次 queue 方法中会把动画数据写到队列中，然后取出队列中的第一个序列通过 dequeue 方法执行
3. 开始执行之前写一个进程锁“inprogress”到 queue 里面，代表这个动画还在执行中，防止同个序列的多个动画重复执行，这个就是异步执行同步收集的处理方案
4. 此时动画开始了，这里注意动画是在异步执行的同步的代码，继续调用下一个 animate
5. 执行同样的 animate 方法逻辑但是此时问题来了，动画可能还在执行可是后续的 animate 还在继续调用，所以这个时候后面的动画代码就需要等待了（进程锁）
6. 队列头是有一把“inprogress”进程锁的，那么这时候动画只需要加入队列，但是可以通过 inprogress 是否存在来判断是否执行
7. 所有的 animate 方法在加入队列都是按照以上的逻辑依次执行，动画执行完毕了就会有一个结束通知，然后从 queue 取出第一个队列继续执行了，如此循环

   以上是整个动画的调度一个流程，其实都是利用队列异步的空闲然后执行同步的代码，这样在处理上是没有浪费资源的，而且精确度也是最高的。

Yunhero 12个月前 (03-23)