封装库--透明度渐变

学习要点：

1.问题所在

2.设置代码

本节课，我们接着运动动画再来扩展一下另一个形式的动画，透明度渐变动画。

一．问题所在

如果单独做一个方法来实现匀速渐变和缓冲渐变，问题不是很大。但如果直接在animate方法扩展就需要注意一些问题。

二．设置代码

1.创建透明度渐进动画

如果单独创建一个方法来处理透明度的渐进动画，我们可以复制animate方法，把长度匀速或缓冲改成渐进的匀速和缓冲即可。但如果还是要封装到animate进行调用，则需要做些判断。

//添加一个渐进动画的属性

var attr = obj['attr'] == 'x' ? 'left' : obj['attr'] == 'y' ? 'top' :

obj['attr'] == 'w' ? 'width' : obj['attr'] == 'h' ? 'height' :

obj['attr'] == 'o' ? 'opacity' : 'left';

PS：由于opacity:0.3属性IE不支持，需要IE专用的filter:alpha(opacity=30)，而需要进行小数处理，这样导致我们的getStyle()获取CSS内置的parseInt直接截掉了小数后的数字，所以，我们需要重新改写getStyle()，并且查询之前使用getStyle()的地方，修改一下。

2.渐进动画也分匀速和缓冲，缓冲用的多，默认。

if (attr == 'opacity') {

var temp = parseFloat(getStyle(element, attr)) \* 100;

if (step == 0) {

setOpacity();

} else if (step > 0 && Math.abs(temp - target) <= step) {

setOpacity();

} else if (step < 0 && (temp - target) <= Math.abs(step)) {

setOpacity();

} else {

element.style.filter = 'alpha(opacity='+ parseInt(temp + step) +')';

element.style.opacity = parseInt(temp + step) / 100;

}

}

function setOpacity() {

element.style.filter = 'alpha(opacity='+ target +')';

element.style.opacity = target / 100;

clearInterval(timer);

}

PS：要注意parseInt(temp + step)的用途，因为计算机对小数经常不敏感，需要取整操作，不然可能会造成渐变闪烁问题。

3.对于透明度独有或运动独有的，要分别判断，否则会混在一起。

if (attr != 'opacity') element.style[attr] = start + 'px'; //px像素是运动独有的

//在缓冲上，opacity采用parseFloat，运动采用parseInt

if (type == 'buffer') {

var parse = attr == 'opacity' ? (target - parseFloat(getStyle(element, attr) \* 100)) :

(target - parseInt(getStyle(element, attr)));

var temp = parse / speed;

step = step > 0 ? Math.ceil(temp) : Math.floor(temp);

}