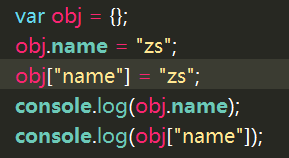
1. 动画原理

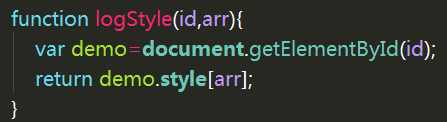
1.1 动画原理公式

1.2 动画函数的封装

1.3 访问属性的两种方式



1.4 访问样式属性的函数



1.5 获取计算后的样式

看一下知道就可以

var demo = **document**.getElementById("demo");  
**console**.log(demo.**style**.**left**); //既然没有行内样式，也没有定位，两者缺一不可，结果为“没有”  
**console**.log(demo.**offsetLeft**); //父盒子没有定位，以body左侧为准，结果为“浏览器默认的左侧8px”

**获取任意对象的任意属性**

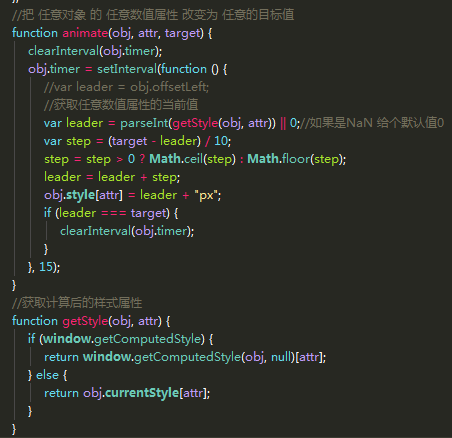
function getStyle(obj,attr){  
 if(**window**.getComputedStyle){  
 return **window**.getComputedStyle(obj,null)[attr];  
 }else{  
 return obj.**currentStyle**[attr];  
 }  
}

2 封装缓动框架

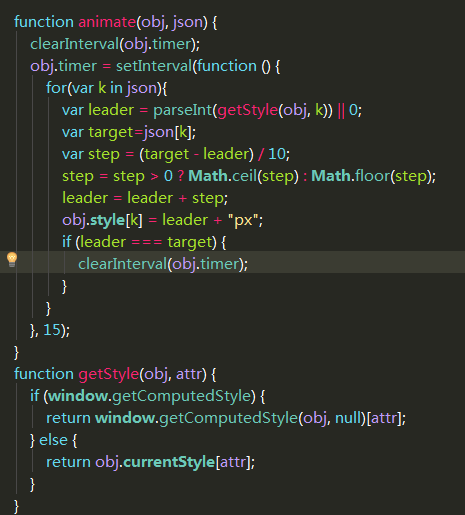
2.1最原始的缓动动画函数

//让任意对象 移动到指定位置  
function animate(obj, target) {  
 clearInterval(obj.timer);  
 obj.timer = setInterval(function () {  
 var leader = obj.**offsetLeft**;  
 var step = (target - leader) / 10;  
 step = step > 0 ? **Math**.ceil(step) : **Math**.floor(step);  
 leader = leader + step;  
 obj.**style**.**left** = leader + "px";  
 if (leader === target) {  
 clearInterval(obj.timer);  
 }  
 }, 15);  
}

2.2 封装缓动框架---任意数值属性



2.3 封装缓动框架---多个数值属性



2.4 封装缓动框架---升级版定时器

//把 任意对象 的 任意数值属性 改变为 任意的目标值  
function animate(obj, json) {  
 clearInterval(obj.timer);  
 obj.timer = setInterval(function () {  
 //先假设 这一次执行完 所有的属性都到达目标了  
 var flag = true;  
 for (var k in json) {  
 var leader = parseInt(getStyle(obj, k)) || 0;  
 var target = json[k];  
 var step = (target - leader) / 10;  
 step = step > 0 ? **Math**.ceil(step) : **Math**.floor(step);  
 leader = leader + step;  
 obj.**style**[k] = leader + "px";  
 if (leader != target) {  
 flag = false;//告诉标记 当前这个属性还没到达  
 }  
 }  
 //如果此时仍然为true 就说明真的都到达了  
 if (flag) {  
 clearInterval(obj.timer);  
 }  
 }, 15);  
}  
//全部属性都到达目标值才能清空  
function getStyle(obj, attr) {  
 if (**window**.getComputedStyle) {  
 return **window**.getComputedStyle(obj, null)[attr];  
 } else {  
 return obj.**currentStyle**[attr];  
 }  
}

2.5 封装缓动框架---回调函数

//把 任意对象 的 任意数值属性 改变为 任意的目标值  
function animate(obj, json, fn) {  
 clearInterval(obj.timer);  
 obj.timer = setInterval(function () {  
 var flag = true;  
 for (var k in json) {  
 var leader = parseInt(getStyle(obj, k)) || 0;  
 var target = json[k];  
 var step = (target - leader) / 10;  
 step = step > 0 ? **Math**.ceil(step) : **Math**.floor(step);  
 leader = leader + step;  
 obj.**style**[k] = leader + "px";  
 if (leader != target) {  
 flag = false;  
 }  
 }  
 if (flag) {  
 clearInterval(obj.timer);  
 if (fn) {  
 fn();  
 }  
 }  
 }, 15);  
}  
  
function getStyle(obj, attr) {  
 if (**window**.getComputedStyle) {  
 return **window**.getComputedStyle(obj, null)[attr];  
 } else {  
 return obj.**currentStyle**[attr];  
 }  
}

2.6 封装缓动框架---透明度和层级

function animate(obj, json, fn) {  
 clearInterval(obj.timer);  
 obj.timer = setInterval(function () {  
 var flag = true;  
 for (var k in json) {  
 if (k === "opacity") {//opacity要特殊处理  
 //opacity没有单位 参与运算自动转换成数值 所以不用parsetInt  
 //取值范围 0-1 0.1 0.33 33 为了让以前的计算公式生效 要扩大100倍  
 var leader = getStyle(obj, k) \* 100;  
 var target = json[k] \* 100;  
 var step = (target - leader) / 10;  
 step = step > 0 ? **Math**.ceil(step) : **Math**.floor(step);  
 leader = leader + step;  
 obj.**style**[k] = leader / 100;//opacity没有单位  
 } else if (k === "zIndex") {  
 obj.**style**.**zIndex** = json[k];//层级不需要渐变 直接设置即可  
 } else {  
 var leader = parseInt(getStyle(obj, k)) || 0;  
 var target = json[k];  
 var step = (target - leader) / 10;  
 step = step > 0 ? **Math**.ceil(step) : **Math**.floor(step);  
 leader = leader + step;  
 obj.**style**[k] = leader + "px";  
 }  
 if (leader != target) {  
 flag = false;  
 }  
 }  
 if (flag) {  
 clearInterval(obj.timer);  
 if (fn) {  
 fn();  
 }  
 }  
 }, 15);  
}  
//全部属性都到达目标值才能清空  
function getStyle(obj, attr) {  
 if (**window**.getComputedStyle) {  
 return **window**.getComputedStyle(obj, null)[attr];  
 } else {  
 return obj.**currentStyle**[attr];  
 }  
}

1. scroll系列

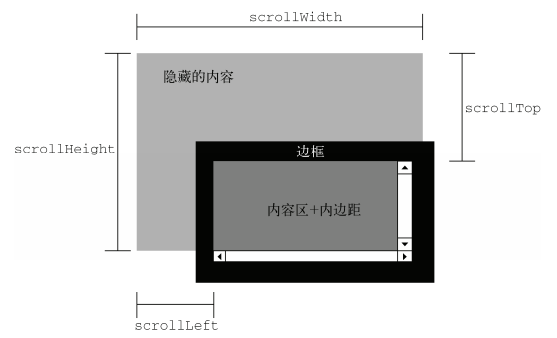
3.1 scrollHeight和scrollWidth

对象内部实际内容的高度/宽度



3.2scrollTop和scrollLeft

被卷去部分的 顶部/左侧 到可视区域 顶部/左侧 的距离



3.3 页面滚动座标，封装scroll函数

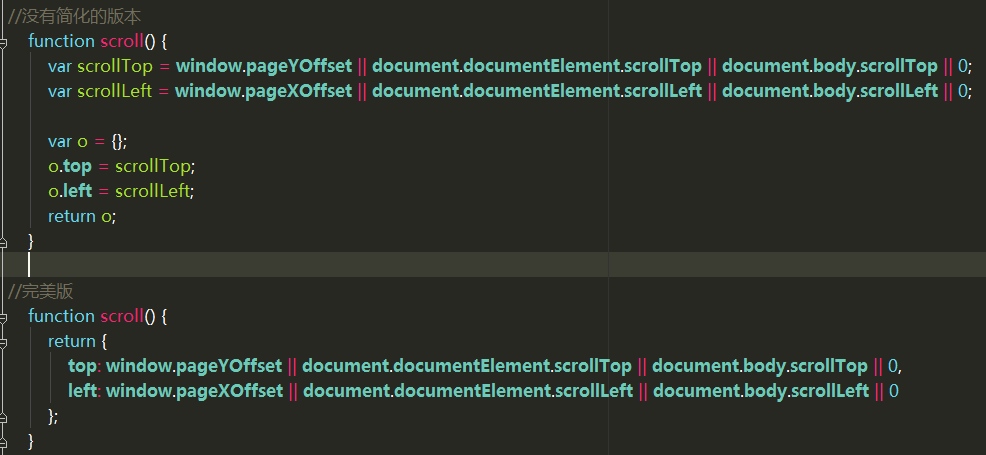
页面滚动座标非常常用，但是有很大的兼容性问题，可以合写为

var scrollTop = window.pageYOffset || document.documentElement.scrollTop || document.body.scrollTop || 0;

**注意**

**console**.log(**document**.**documentElement**);//html标签 是documentElement

**scroll函数封装**



3.4怪异模式

