面向对象的拖拽

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>无标题文档</title>

<style type="text/css">

#div1{ width:100px; height:100px; position:absolute; top:0px; left:0px; cursor:move; background:red; }

</style>

</head>

<body>

<div id="div1"></div>

</body>

</html>

<script src="event.js"></script>

<script>

//面向对象版的拖拽，难点之一就是为什么要用面向对象的方式来编程。

//它是管理方式，管理思想，它在扩展性和可维护性上，还功能的独立性上要好。

//this的原则：this要指向当前类的实例

function Drag(ele){//构造函数负责初始化

this.ele=ele;//把需要拖动的元素保存在自己的实例上

this.x=null;

this.y=null;

this.mx=null;

this.my=null;

this.DOWN=processThis(this,this.down);

this.DOWN=this.down.bind(this);

this.MOVE=this.move.bind(this);

this.UP=this.up.bind(this);

on(this.ele,"mousedown",this.DOWN);

}

Drag.prototype.down=function(e){

this.x=this.ele.offsetLeft;

this.y=this.ele.offsetTop;

this.mx=e.pageX;

this.my=e.pageY;

if(this.ele.setCapture){

this.ele.setCapture();

on(this.ele,"mousemove",this.MOVE);

on(this.ele,"mouseup",this.UP);

}else{

on(document,"mousemove",this.MOVE);

on(document,"mouseup",this.UP);

}

}

Drag.prototype.move=function(e){

this.ele.style.left=e.pageX-this.mx+this.x+"px";

this.ele.style.top=e.pageY-this.my+this.y+"px";

}

Drag.prototype.up=function(e){

if(this.ele.releaseCapture){

this.ele.releaseCapture();

off(this.ele,"mousemove",this.MOVE);

off(this.ele,"mouseup",this.UP);

}else{

off(document,"mousemove",this.MOVE);

off(document,"mouseup",this.UP);

}

}

var obj1=new Drag(div1);

/\*var obj2=new Drag(div2);

var obj3=new Drag(div3);\*/

/\*obj2.addBorder();//

obj2.range({left:100,top:100,right:800,bottom:500})

Drag 面向对象的类，是更高级的分类

单例模式：是一个简单的功能的分类\*/

</script>

事件库

//使用设计模式时，首先要做一些约定：制定一些标准 ASCII

//约定：凡是自定义事件的事件类型，都以字符串self开头，以避免和系统事件冲突

//"selfdragstart","selfdragging","selfdragend"

function on(ele,type,fn){

if(/^self/.test(type)){

if(!ele["selfEvent"+type]){

ele["selfEvent"+type]=[];

}

var a=ele["selfEvent"+type];

for(var i = 0;i<a.length;i++){

if(a[i]==fn)return;

}

a.push(fn);

return ;

}

if(ele.addEventListener){

ele.addEventListener(type,fn);

return;

}

if(!ele["myEvent"+type]){

ele["myEvent"+type]=[];

ele.attachEvent("on"+type,function(){run.call(ele)})

}

var a=ele["myEvent"+type];

for(var i=0;i<a.length;i++){

if(a[i]==fn)return;

}

a.push(fn);

}

function run(){

var e= window.event;

if(!e.target){

e.target=e.srcElement;

e.pageX=e.clientX+(document.documentElement.scrollLeft||document.body.scrollLeft);

e.pageY=e.clientY+(document.documentElement.scrollTop||document.body.scrollTop);

e.preventDefault=function(){e.returnValue=false}

e.stopPropagation=function(){e.cancelBubble=true}

}

var ary=this["myEvent"+e.type];

for(var i=0;i<ary.length;i++){

var curFn=ary[i];

if(typeof curFn==="function"){

curFn.call(this,e)

}else{

ary.splice(i,1);

i--;

}

}

}

function off(ele,type,fn){

if(/^self/.test(type)){

var a =ele["selfEvent"+type];

if(a){

for(var i=0;i<a.length;i++){

if(a[i]==fn){

a[i]=null;

return;

}

}

}

return;

}

if(ele.removeEventListener){

ele.removeEventListener(type,fn,false);

return;

}

var a=ele["myEvent"+type];

if(a){

for(var i=0;i<a.length;i++){

if(a[i]==fn){

a[i]=null;

return;

}

}

}

}

function processThis(obj,fn){

return function(e){fn.call(obj,e);}

}

function selfRun(selfType,e){//“通知”,用来发布自定义事件的.selfType是指在这儿发布的自定义事件的事件类型。e是系统的事件对象

var a=this["selfEvent"+selfType];

for(var i=0;i<a.length;i++){

if(typeof a[i] == "function"){

a[i].call(this,e);

}else{

a.splice(i,1);

i--;

}

}

}

拖拽动画的效果

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>无标题文档</title>

<style type="text/css">

#div1{ width:100px; height:100px; position:absolute; top:0px; left:0px; cursor:move; background:red; }

</style>

</head>

<body>

<div id="div1">

sssss中国人发

</div>

</body>

</html>

<script src="event.js"></script>

<script>

var ele=document.getElementById("div1");

on(ele,"mousedown",down);

function down(e){//准备拖拽：把原始的坐标保存，并且绑定mousemove和up事件

this.x=e.clientX;//鼠标的原始坐标

this.y=e.clientY;

this.l=ele.offsetLeft;//盒子的原始坐标

this.t=ele.offsetTop;

if(this.setCapture){

this.setCapture();//IE专门用来处理mousemove事件的，使元素不失鼠标焦点

on(this,"mousemove",move);

on(this,"mouseup",up);

}else{

this.MOVE=move.bind(this);//相当于processThis,IE不支持

this.UP=up.bind(this);

on(document,"mousemove",move);

on(document,"mousemove",this.MOVE);

on(document,"mouseup",this.UP);

}

e.preventDefault();

selfRun.call(this,"selfdragstart",e);//通知，或发布事件

}

function move(e){

this.style.left=e.pageX-this.x+this.l+"px";

this.style.top=e.pageY-this.y+this.t+"px";

selfRun.call(this,"selfdragging",e);//通知，或发布事件

}

function up(e){

if(this.releaseCapture){

this.releaseCapture();

off(this,"mousemove",move);

off(this,"mouseup",up);

}else{

off(document,"mousemove",this.MOVE);

off(document,"mouseup",this.UP);

}

selfRun.call(this,"selfdragend",e);//通知，或发布事件

}

/\* 以上是drag模块 以下是动画效果的模块 \*/

function clearEffect(){

clearTimeout(flyTimer);

clearTimeout(dropTimer);

}

function getSpeed(){

if(this.prevTime){//用来保存上一次触发mousemove事件的时间点

var currentTime=+new Date;

this.prevTime=currentTime;

this.speed=this.offsetLeft-this.prevPosi;

this.prevPosi=this.offsetLeft;//更新上一次的位置

}else{

this.prevTime=+new Date;

this.prevPosi=this.offsetLeft;//初始化上一次的位置

}

}

var flyTimer=null

function fly(){

w=document.documentElement.clientWidth-this.offsetWidth;

if(this.offsetLeft+this.speed>=w){

this.style.left=w+"px";

this.speed\*=-1;//调头，让速度的方向变反相

}else if(this.offsetLeft+this.speed<=0){

this.style.left=0;

this.speed\*=-1;

}else{

this.style.left=this.offsetLeft+this.speed+"px";

}

this.speed\*=.97;//让动能衰减，每次减少3%

if(Math.abs(this.speed)>=0.5){

flyTimer=window.setTimeout(processThis(this,fly),20)

}else{

this.speed=null;

this.prevPosi=null;

}

}

var dropFlag=0;//用来判断是否到达边界的标识：如果大于1则说明到站了

var dropTimer=null;//是定时器变量

function drop(){

if(this.dropSpeed){

this.dropSpeed+=9;

}else{

this.dropSpeed=9;

}

var bottom=document.documentElement.clientHeight-this.offsetHeight;

if(this.offsetTop+this.dropSpeed>=bottom){

this.style.top=bottom+"px";

this.dropSpeed\*=-1;

dropFlag++;

}else{

this.style.top=this.offsetTop+this.dropSpeed+"px";

dropFlag=0;

}

if(dropFlag>=2){

//把动画停止之后的一些操作写在这儿

}else{//dropFlag小于2说明动画还需要正常的运动

dropTimer=window.setTimeout(processThis(this,drop),20);

}

//一百行代须

}

/\* 以下是自定义事件的绑定 \*/

on(ele,"selfdragstart",clearEffect)

on(ele,"selfdragging",getSpeed);

on(ele,"selfdragend",drop);

on(ele,"selfdragend",fly);

//this，call 我

//事件绑定的时候，这个方法绑定给那个元素的事件，则这个方法里的this就是那个元素

//面向对象编程里，this是指的当前实例

//冲突出现了：把一个实例方法：比如说a.fn绑定给ele.onclick=a.fn;

</script>