**第13章 事件操作**

在Canvas中，常见的事件共有3种：鼠标事件、键盘事件和循环事件。

**13.5.1 鼠标事件**

在Canvas中，鼠标事件分为3种：

（1）鼠标按下：mousedown

（2）鼠标松开：mouseup

（3）鼠标移动：mousemove

这里我们封装了一个获取鼠标位置的函数getMouse()，然后存放到“tools.js”文件中去，以便后续开发使用。兼容代码如下所示。

**语法：**

//将tools定义为window对象的属性，该属性的值是一个对象

window.lvye = {};

//获取鼠标位置

window.tools.getMouse = function (element) {

//定义一个mouse的对象

var mouse = { x: 0, y: 0 };

//为传入的元素添加mousemove事件

addEvent(element, "mousemove", function (e) {

var x, y;

//在IE中，event对象是作为window对象的一个属性存在

var e = e || window.event;

//获取鼠标当前位置，并作兼容处理

//兼容Firefox、chrome、IE9及以上

if (e.pageX || e.pageY) {

x = e.pageX;

y = e.pageY;

}

//兼容IE8及以下，以及混杂模式下的Chrome和Safari

else {

x = e.clientX + document.body.scrollLeft + document.documentElement.scrollLeft;

y = e.clientY + document.body.scrollTop + document.documentElement.scrollTop;

}

//将当前的坐标值减去canvas元素的偏移位置，则x、y为鼠标在canvas中的相对坐标

x -= element.offsetLeft;

y -= element.offsetTop;

mouse.x = x;

mouse.y = y;

})

//返回值为mouse对象

return mouse;

}

**说明：**

getMouse()方法接受一个DOM元素作为参数，并且使用addEventListener()为这个元素添加一个mousemove事件，最终返回一个包含鼠标坐标的对象。

**13.5.2 键盘事件**

在Canvas中，常用的键盘事件共有2种：

**（1）键盘按下：keydown**

**（2）键盘松开：keyup**

由于canvas元素本身不支持键盘事件，因此一般情况下我们都是使用window对象来实现Canvas中键盘事件的监听。

**语法：**

window.addEventListener(type , fn , false)

**说明：**

type是一个字符串，指的是事件类型，如果只针对Canvas键盘事件，取值只有2个： "keydown"和"keyup"。fn是一个事件处理函数，我们在键盘事件中操作都是在里面处理。

在Canvas键盘事件中，我们一般都是根据按键的keyCode来判断用户按下了键盘中的哪个键，然后再进行相应操作的。常用的按键的keyCode如下表所示。这些按键往往都是用来控制物体移动的方向。

**表13-2 keyCode值对应按键**

|  |  |
| --- | --- |
| **keyCode** | **按键** |
| W（上） | 87 |
| S（下） | 83 |
| A（左） | 65 |
| D（右） | 68 |
| ↑ | 39 |
| ↓ | 41 |
| ← | 37 |
| → | 40 |

对于控制方向按键的keyCode，我们将其封装成一个函数getKey()，以便在实际开发中直接调用。

**语法：**

//获取键盘控制方向

window.tools.getKey = function () {

var key = {};

window.addEventListener("keydown", function (e) {

if (e.keyCode == 38 || e.keyCode == 87) {

key.direction = "up";

} else if (e.keyCode == 39 || e.keyCode == 68) {

key.direction = "right";

} else if (e.keyCode == 40 || e.keyCode == 83) {

key.direction = "down";

} else if (e.keyCode == 37 || e.keyCode == 65) {

key.direction = "left";

} else {

key.direction = "";

}

}, false);

return key;

}

**说明：**

getKey()方法返回一个对象key，这个对象有一个direction属性，表示用户控制物体移动的方向。每次我们只需要判断direction属性值是什么，就可以知道物体移动的方向了。这种使用名称而不是使用数字的方式，使得代码更加简单易懂。

**13.5.3 循环事件**

在Canvas中，我们都是使用requestAnimationFrame()方法来实现循环，从而达到动画效果。

**语法：**

(function frame(){

window.requestAnimationFrame(frame);

cxt.clearRect(0, 0, cnv.width, cnv.height);

……

})();

**分析：**

在这个语法中，我们定义了一个自执行函数frame()，然后在函数内部使用window.requestAnimationFrame()来不断调用frame，从而实现循环效果。

此外，requestAnimationFrame()方法存在严重的兼容性问题，因此对于各种浏览器我们需要做兼容处理，其中兼容代码如下：

window.requestAnimationFrame = (

window.webkitRequestAnimationFrame ||

window.mozRequestAnimationFrame ||

window.msRequestAnimationFrame ||

window.oRequestAnimationFrame ||

function (callback) {

return window.setTimeout(callback, 1000/60);

}

);

对于这个兼容代码，我们同样也是放到“tools.js”文件中去，以便开发直接调用。