面向对象案例二

放大镜

效果分析

组成结构分析

1. 一个容器大盒子 - div.box
2. 一个放中等图片的盒子 - div.middleBox
    - 一张中等图片
    - 一个遮罩 - div.shade
3. 一个放小图片的盒子 - div.smallBox
    - 两张小图
4. 一个放大图的盒子 - div.bigBox

布局分析

1. 大盒子box设置相对定位
2. 中等盒子middleBox设置相对定位
3. 遮罩shade设置绝对定位在中等盒子中，隐藏
4. 放大图的盒子bigBox设置绝对定位，定位到大盒子box的右边，大图作为放大图的盒子的背景

效果分析

1. 点击小图片，大图随着小图更换
2. 鼠标移动大中等盒子上
    - 遮罩出现，并可以在中等盒子上移动
    - 遮罩移动的过程中大盒子的背景图也跟随移动
3. 鼠标离开大盒子
    - 遮罩隐藏，放大图的盒子也隐藏，并将移动事件注销

规律分析

1. 中等盒子和遮罩的比例，是大图和放大图的盒子的比例
2. 遮罩移动过的距离和中等盒子的比例，是大图移动的距离和大图的比例

代码实现

html结构

```
<div class="box">
    <div class="middleBox">
        <img src="./images/middle1.jpg">
        <div class="shade"></div>
    </div>
    <div class="smallBox">
        <img class="active" src="./images/small1.jpg" alt="">
        <img src="./images/small2.jpg" alt="">
    </div>
    <div class="bigBox"></div>
</div>
```

样式代码

```
.middleBox{
    width: 400px;
    height: 400px;
    border:1px solid #000;
    position: relative;
}
.shade{
    width: 100px;
    height: 100px;
    background:yellow;
    position: absolute;
    left:0;
    top:0;
    opacity:0.5;
    display:none;
}
.smallBox{
    margin-top:10px;

}
.smallBox img{
    border:1px solid #000;
    margin-left:5px;
}
.smallBox img.active{
    border-color:red;
}
.box{
    margin:50px;
    width: 402px;
    position: relative;
}
.bigBox{
    width: 400px;
    height: 400px;
    border:1px solid #000;
    position:absolute;
    top:0;
```

```
    left:105%;
    display:none;
    background-image:url(./images/big1.jpg);
    background-position:0 0;
    background-size:1600px 1600px;
    background-repeat:no-repeat;
}
.shade:hover{
    cursor:move;
}
```

js代码

```
// 定义一个空构造函数
function Enlarge(classname){
    this.box = document.querySelector("."+classname);
    // 把需要操作的所有的元素都获取到，并绑定到对象的属性上
    this.middleBox = this.box.querySelector(".middleBox");
    this.middleImg = this.box.querySelector(".middleBox img");
    this.shade = this.box.querySelector(".shade");
    this.smallImgs = this.box.querySelectorAll(".smallBox img");
    this.bigBox = this.box.querySelector(".bigBox");
    // 给需要操作的元素绑定事件
    var _this = this;
    for(var i=0;i<this.smallImgs.length;i++){
        this.smallImgs[i].onclick=function(){
            _this.tab(this);
        }
    }
    // 中等盒子鼠标放上去的事件
    this.middleBox.onmouseover=function(){
        _this.over();
    }
    // 中等盒子鼠标离开的事件
    this.middleBox.onmouseout=function(){
        _this.out();
    }
}
// 鼠标离开中盒子的事件
Enlarge.prototype.out=function(){
    this.shade.style.display = 'none';
    this.bigBox.style.display = 'none';
    this.middleBox.onmousemove=null;
}
// 鼠标放到中盒子上的事件
Enlarge.prototype.over=function(){
    this.shade.style.display = 'block';
    var _this = this;
    this.middleBox.onmousemove=function(){
        _this.move();
    }
}
// 鼠标在中盒子上移动的事件
```

```
Enlarge.prototype.move=function(e){
    var e = e || window.event;
    var x = e.clientX;
    var y = e.clientY;
    // 遮罩的宽高的一半
    var shadeWidthBan = this.shade.clientWidth/2;
    var shadeHeightBan = this.shade.clientHeight/2;
    // 限定x和y在左上角的最小位置
    if(x<this.box.offsetLeft+shadeWidthBan){
        x = this.box.offsetLeft+shadeWidthBan
    }
    if(y<this.box.offsetTop+shadeHeightBan){
        y = this.box.offsetTop+shadeHeightBan
    }
    // 限定x和y在右下角的最大值
    if(x>this.box.offsetLeft+this.middleBox.offsetWidth-shadeWidthBan){
        x = this.box.offsetLeft+this.middleBox.offsetWidth-shadeWidthBan;
    }
    if(y>this.box.offsetTop+this.middleBox.offsetHeight-shadeHeightBan){
        y = this.box.offsetTop+this.middleBox.offsetHeight-shadeHeightBan;
    }
    // console.log(y);
    // 给遮罩设置left和top，让遮罩移动起来
    this.shade.style.left = x - this.box.offsetLeft - shadeWidthBan + 'px';
    this.shade.style.top = y - this.box.offsetTop - shadeHeightBan + 'px';
    this.bigImgMove();
}
// 移动遮罩让大图移动
Enlarge.prototype.bigImgMove=function(){
    this.bigBox.style.display = 'block';
    /*
    遮罩移动的距离                大图移动距离
    ------------      ===         ---------
    中盒子的大小                   大图的大小
    */
    // 计算遮罩和中盒子的比例
    var xPercent = this.shade.offsetLeft/this.middleBox.clientWidth;
    var yPercent = this.shade.offsetTop/this.middleBox.clientHeight;
    // console.log(this.middleBox.clientWidth);
    // 计算大图的大小 - 大图是大盒子的背景图 - 获取背景图的大小
    var bigImgBgSize = getStyle(this.bigBox,"background-size");
    // 获取 单独的宽高的数字
    var bigImgBgWidth = parseInt(bigImgBgSize.split(" ")[0]);
    var bigImgBgHeight = parseInt(bigImgBgSize.split(" ")[1]);
    // 计算大图要移动的距离
    var xMove = bigImgBgWidth*xPercent;
    var yMove = bigImgBgHeight*yPercent;
    // 把大图要移动的距离赋值到大图的背景定位的left和top上
    this.bigBox.style.backgroundPosition = -xMove+'px '+-yMove+'px';
    // console.log(xMove,yMove);
    // console.log(this.shade.offsetLeft,this.shade.offsetTop);
}
// 点击小图换中图，并换大图
```

```
Enlarge.prototype.tab=function(smallImg){
    // 找到当前点击的小图的路径
    var smallImgSrc = smallImg.src;
    // 截取到 数字.jpg

    // 找到点的下标
    var dotIndex = smallImgSrc.lastIndexOf(".");
    // 截取到末尾
    var targetSuffix = smallImgSrc.substring(dotIndex-1);
    // 拼接成中等图片的路径
    var middleImgSrc = "./images/middle"+targetSuffix;
    // 拼接成大图的路径
    var bigImgSrc = "url(./images/middle"+targetSuffix+")";
    // 将中等图片的路径赋值到中等图的src属性上
    this.middleImg.src = middleImgSrc;
    // console.log(middleImgSrc);
    // 将大图的路径赋值到放大图的盒子的背景中
    this.bigBox.style.backgroundImage = bigImgSrc;
    // 将别人的active类名干掉，给自己添加active类名
    for(var i=0;i<this.smallImgs.length;i++){
        this.smallImgs[i].className = '';
    }
    smallImg.className = 'active';

}
// 实例化他
var enlarge = new Enlarge("box");
// console.log(enlarge);

function getStyle(ele,attr){
    if(window.getComputedStyle){
        return window.getComputedStyle(ele)[attr];
    }else{
        return ele.currentStyle[attr];
    }
}
```

烟花

效果分析

结构分析

1. 大盒子做夜空
2. 点击的时候生成一个小烟花
3. 小烟花升空后生成很多小烟花
4. 让很多小烟花移动到一个随机位置

效果分析

1. 大盒子点击事件

2. 生成小烟花，设置在大盒子最底部，横向位置是鼠标所点的位置。放到大盒子中，调用运动函数移动到鼠标所在高度
3. 将小烟花删掉，并生成很多小烟花，设置成圆形，位置就是鼠标所点击的位置，随机颜色
4. 让生成的很多小烟花移动到随机位置，移动后将很多小烟花从大盒子删除

代码实现：

```
<body>
<div class="box"></div>
</body>
<script src="./js/sport.js"></script>
<script type="text/javascript">
function Fire(classname){
    this.box = document.querySelector("."+classname);
    // 给元素设置样式
    setStyle(this.box,{
        width:"1000px",
        height:"600px",
        background:"#000",
        border:"10px solid pink",
        position:"relative"
    });
    // 给盒子注册点击事件
    this.box.onclick=()=>{
        this.click();
    }
}
// 盒子的点击事件
Fire.prototype.click=function(e){
    var e = e || window.event;
    var x = e.offsetX;
    var y = e.offsetY;
    // 生成一个小烟花
    var div = document.createElement("div");
    setStyle(div,{
        width:"10px",
        height:"10px",
        position:"absolute",
        left:x+'px',
        bottom:0,
        background:getColor()
    });
    this.box.appendChild(div);
    // 让小烟花升空
    this.toUp(div,x,y);
}
// 让小烟花升空
Fire.prototype.toUp=function(ele,x,y){
    var _this = this;
    sport(ele,{
        top:y
    },function(){
        _this.box.removeChild(ele);
```

```
        // 升空以后，在当前这个位置生成很多的小烟花
        _this.createManyFire(x,y);
    });
}
Fire.prototype.createManyFire=function(x,y){
    var num = getRondom(25,30);
    for(var i=0;i<num;i++){
        var div = document.createElement("div");
        setStyle(div,{
            width:"10px",
            height:"10px",
            position:"absolute",
            background:getColor(),
            borderRadius:"50%",
            left:x+'px',
            top:y+'px'
        });
        this.box.appendChild(div);
        // 让小烟花炸开
        this.boom(div);
    }
}
// 让很多小烟花炸开
Fire.prototype.boom=function(ele){
    var _this = this;
    sport(ele,{
        left:getRondom(0,_this.box.clientWidth-ele.clientWidth),
        top:getRondom(0,_this.box.clientHeight-ele.clientHeight),
    },function(){
        _this.box.removeChild(ele);
    });
}
var fire = new Fire("box");
console.log(fire);

// 设置样式
function setStyle(ele,styleObj){
    for(var attr in styleObj){
        ele.style[attr] = styleObj[attr];
    }
}
// 生成随机颜色
function getColor(){
    var rgb1 = getRondom(0,255);
    var rgb2 = getRondom(0,255);
    var rgb3 = getRondom(0,255);
    return `rgb(${rgb1},${rgb2},${rgb3})`;
}
// 生成随机数
function getRondom(min,max){
    if(min>max){
        var tmp = min;
        min = max;
```

```
        max = tmp;
    }
    // 20~100之间的随机数    parseInt(Math.random()*(100-20))+20
    return parseInt(Math.random()*(max-min))+min;
}
</script>
```

贪吃蛇

布局和css

```
<button id="btn">移动</button>
```

js代码

```
// 分3部分 - 1.地图 2.食物 3.蛇
// 创造地图的构造函数
function Map(){
    // 创建地图区域 - div
    this.map = document.createElement("div");
    // 设置地图的区域大小、背景、边框
    setStyle(this.map,{
        width:"600px",
        height:"400px",
        border:"3px solid #000",
        background:"pink",
        position:"relative"
    });
    // 将div放到body中
    document.body.appendChild(this.map);
}
var map = new Map();
console.log(map);
// 食物的构造函数
function Food(){
    // 创建食物
    this.food = document.createElement("div");
    // 设置食物样式
    setStyle(this.food,{
        width:"10px",
        height:"10px",
        background:getColor(),
        position:"absolute",
        // left:getRandom(0,map.map.clientWidth-parseInt(this.food.style.width))+"px"
        // left:getRandom(0,map.map.clientWidth-parseInt(this.food.style.width)) + "px",
        // top:getRandom(0,map.map.clientHeight-parseInt(this.food.style.height)) + "px",
    })
    this.food.style.left = getRandom(0,parseInt(map.map.clientWidth-
parseInt(this.food.style.width))/10)*10+"px"
    this.food.style.top = getRandom(0,parseInt(map.map.clientHeight-
parseInt(this.food.style.height))/10)*10 + "px"
```

```
    // 把食物放到地图中
    map.map.appendChild(this.food);
}
var food = new Food();
console.log(food);
// 蛇的构造函数
function Snake(){
    // 蛇的属性：移动方向、蛇的身体
    this.direction = "right";
    this.body = [
        {
            x:20,
            y:0,
        },
        {
            x:10,
            y:0,
        },
        {
            x:0,
            y:0,
        }
    ];
    // 根据body展示蛇的身体
    this.show();
    var _this = this;
    // 改变方向
    document.onkeypress=function(e){
        var e = e || window.event;
        var keyCode = e.keyCode || e.which;
        _this.change(keyCode);
    }
}
// 改变方向的方法
Snake.prototype.change=function(keyCode){
    switch(keyCode){
        case 119:
            this.direction = "up";
        break;
        case 115:
            this.direction = "down";
        break;
        case 97:
            this.direction = "left";
        break;
        case 100:
            this.direction = "right";
        break;
    }
}
// 展示蛇的身体
Snake.prototype.show=function(){
    // 遍历body，每一个元素都是蛇的一节身体
```

```
    for(var i=0;i<this.body.length;i++){
        var div = document.createElement("div");
        // 给蛇的身体做一个标志
        div.className = "snake"
        setStyle(div,{
            width:"10px",
            height:"10px",
            background:"blue",
            position:"absolute",
            left:this.body[i].x + "px",
            top:this.body[i].y + "px",
        });
        if(i==0){
            // 蛇头圆角
            div.style.borderRadius = "50%";
            div.style.background = "red";
        }
        map.map.appendChild(div);
    }
}
// 让蛇动起来
Snake.prototype.move=function(){
    // 将每一节身体的x和y交给下一节
    for(var i=this.body.length-1;i>0;i--){
        this.body[i].x = this.body[i-1].x;
        this.body[i].y = this.body[i-1].y;
    }
    // 蛇头的位置
    switch(this.direction){
        case "right":
            this.body[0].x += 10;
        break;
        case "left":
            this.body[0].x -= 10;
        break;
        case "up":
            this.body[0].y -= 10;
        break;
        case "down":
            this.body[0].y += 10;
        break;
    }
    // body改变了，根据body将身体重新展示
    // 先将原来的身体干掉，再展示新的身体
    var s = map.map.querySelectorAll(".snake");
    // console.log(s);
    for(var i=0;i<s.length;i++){
        map.map.removeChild(s[i]);
    }
    // console.log(this.body);
    this.show();
    // 吃食物
    this.eat();
```

```
    // 撞墙或者撞身体
    this.die();
}
// 撞墙和撞身体的方法
Snake.prototype.die=function(){
    // 判断撞墙
    if(this.body[0].x>map.map.clientWidth-10 || this.body[0].y>map.map.clientHeight-10 ||
this.body[0].x<0 || this.body[0].y<0){
        alert("游戏结束");
        clearInterval(this.timer);
    }
    // 撞身体
    for(var i=1;i<this.body.length;i++){
        if(this.body[i].x == this.body[0].x && this.body[i].y == this.body[0].y){
            alert("游戏结束");
            clearInterval(this.timer);
        }
    }
}
// 吃食物的方法
Snake.prototype.eat=function(){
    if(this.body[0].x == food.food.offsetLeft && this.body[0].y == food.food.offsetTop){
        // 吃到了
        // console.log(123);
        map.map.removeChild(food.food);
        food = new Food();
        this.body.push({
            x:this.body[this.body.length-1].x,
            y:this.body[this.body.length-1].y,
        })
        // console.log(this.body);
    }
}
// 设置定时器让蛇移动
Snake.prototype.dong=function(){
    var _this = this;
    this.timer = setInterval(function(){
        _this.move();
    },300);
}
var snake = new Snake();
btn.onclick=function(){
    snake.dong();
}



function setStyle(ele,styleObj){
    for(var attr in styleObj){
        ele.style[attr] = styleObj[attr];
    }
}
function getRandom(a,b){
```

```
    var diff = Math.abs(a-b);
    var min = a>b?b:a;
    return parseInt(Math.random()*diff) + min;
}
function getColor(){
    return `rgb(${getRandom(0,256)},${getRandom(0,256)},${getRandom(0,256)})`
}
```