**1. 元素偏移量 offset 系列**

**1.1 offset 概述**

offset 翻译过来就是偏移量， 我们使用 offset 系列相关属性可以动态的得到该元素的位置（偏移）、大小等。

获得元素距离带有定位父元素的位置

获得元素自身的大小（宽度高度）

注意： 返回的数值都不带单位

offset 系列常用属性：



.father {

/\* position: relative; 有定位\*/

width: 200px;

height: 200px;

background-color: pink;

margin: 150px;

}

.son {

width: 100px;

height: 100px;

background-color: purple;

margin-left: 45px;

}

.w {

height: 200px;

background-color: skyblue;

margin: 0 auto 200px;

padding: 10px;

border: 15px solid red;

}

</style>

</head>

<body>

<div class="father">

<div class="son"></div>

</div>

<div class="w"></div>

<script>

// offset 系列

var father = document.querySelector('.father');

var son = document.querySelector('.son');

// 1.可以得到元素的偏移 位置 返回的不带单位的数值

console.log(father.offsetTop);

console.log(father.offsetLeft);

// 它以带有定位的父亲为准 如果么有父亲或者父亲没有定位 则以 body 为准

console.log(son.offsetLeft);

var w = document.querySelector('.w');

// 2.可以得到元素的大小 宽度和高度 是包含padding + border + width

console.log(w.offsetWidth);

console.log(w.offsetHeight);

// 3. 返回带有定位的父亲 否则返回的是body

console.log(son.offsetParent); // 返回带有定位的父亲 否则返回的是body

console.log(son.parentNode); // 返回父亲 是最近一级的父亲 亲爸爸 不管父亲有没有定位

</script>

</body>

**1.2 offset 与 style 区别**

|  |  |
| --- | --- |
| **offset** | **style** |
| offset 可以得到任意样式表中的样式值 | style 只能得到行内样式表中的样式值 |
| offset 系列获得的数值是没有单位的 | style.width 获得的是带有单位的字符串 |
| offsetWidth 包含padding+border+width | style.width 获得不包含padding和border 的值 |
| offsetWidth 等属性是只读属性，只能获取不能赋值 | style.width 是可读写属性，可以获取也可以赋值 |
| 所以，我们想要获取元素大小位置，用offset更合适 | 所以，我们想要给元素更改值，则需要用style改变 |

<style>

.box {

width: 200px;

height: 200px;

background-color: pink;

padding: 10px;

}

</style>

<body>

<div class="box" style="width: 200px;"></div>//style 只能得到行内样式表中的样式值

<script>

// offset与style的区别

var box = document.querySelector('.box');

console.log(box.offsetWidth);//220

console.log(box.style.width);//200px

//offsetWidth 等属性是只读属性，只能获取不能赋值

// box.offsetWidth = '300px';

box.style.width = '300px';

</script>

</body>

**案例：获取鼠标在盒子内的坐标**

① 我们在盒子内点击，想要得到鼠标距离盒子左右的距离。

② 首先得到鼠标在页面中的坐标（e.pageX, e.pageY）

③ 其次得到盒子在页面中的距离 ( box.offsetLeft, box.offsetTop)

④ 用鼠标距离页面的坐标减去盒子在页面中的距离，得到 鼠标在盒子内的坐标

⑤ 如果想要移动一下鼠标，就要获取最新的坐标，使用鼠标移动事件 mousemove

var box = document.querySelector('.box');

box.addEventListener('mousemove', function(e) {

//e鼠标事件对象

// console.log(e.pageX);

// console.log(e.pageY);

// console.log(box.offsetLeft);

var x = e.pageX - this.offsetLeft;

var y = e.pageY - this.offsetTop;

this.innerHTML = 'x坐标是' + x + ' y坐标是' + y;

**案例：拖动模态框**

弹出框，我们也称为模态框。

1. 点击弹出层， 会弹出模态框， 并且显示灰色半透明的遮挡层。

2. 点击关闭按钮，可以关闭模态框，并且同时关闭灰色半透明遮挡层。

3. 鼠标放到模态框最上面一行，可以按住鼠标拖拽模态框在页面中移动。

4. 鼠标松开，可以停止拖动模态框移动。

案例分析：

① 点击弹出层， 模态框和遮挡层就会显示出来 display:block;

② 点击关闭按钮，模态框和遮挡层就会隐藏起来 display:none;

③ 在页面中拖拽的原理： 鼠标按下并且移动， 之后松开鼠标

④ 触发事件是鼠标按下 mousedown， 鼠标移动mousemove 鼠标松开 mouseup

⑤ 拖拽过程: 鼠标移动过程中，获得最新的值赋值给模态框的left和top值， 这样模态框可以跟着鼠标走了

⑥ 鼠标按下触发的事件源是 最上面一行，就是 id 为 title

⑦ 鼠标的坐标 减去 鼠标在盒子内的坐标， 才是模态框真正的位置。

⑧ 鼠标按下，我们要得到鼠标在盒子的坐标。

⑨ 鼠标移动，就让模态框的坐标 设置为 ： 鼠标坐标 减去盒子坐标即可，注意移动事件写到按下事件里面。

⑩ 鼠标松开，就停止拖拽，就是可以让鼠标移动事件解除

<script>

// 1. 获取元素

var login = document.querySelector('.login');

var mask = document.querySelector('.login-bg');

var link = document.querySelector('#link');

var closeBtn = document.querySelector('#closeBtn');

var title = document.querySelector('#title');

// 2. 点击弹出层这个链接 link 让mask 和login 显示出来

link.addEventListener('click', function() {

mask.style.display = 'block';

login.style.display = 'block';

})

// 3. 点击 closeBtn 就隐藏 mask 和 login

closeBtn.addEventListener('click', function() {

mask.style.display = 'none';

login.style.display = 'none';

})

// 4. 开始拖拽

// (1) 当我们鼠标按下， 就获得鼠标在盒子内的坐标，鼠标移动和弹起事件都写在鼠标按下事件内

title.addEventListener('mousedown', function(e) {

var x = e.pageX - login.offsetLeft;

var y = e.pageY - login.offsetTop;

// (2) 鼠标移动的时候，把鼠标在页面中的坐标，减去 鼠标在盒子内的坐标就是模态框的left和top值；事件源是document因为在页面任何一个位置都可以进行鼠标移动

document.addEventListener('mousemove', move)

//把函数单独写出来命名为move，利于mousemove、mouseup事件的传参

function move(e) {

login.style.left = e.pageX - x + 'px';

login.style.top = e.pageY - y + 'px';

}

// (3) 鼠标弹起，就让鼠标移动事件移除

document.addEventListener('mouseup', function() {

document.removeEventListener('mousemove', move);

})

})

</script>

**案例：仿京东放大镜**

**step one**

① 整个案例可以分为三个功能模块

② 鼠标经过小图片盒子， 黄色的遮挡层 和 大图片盒子显示，离开隐藏2个盒子功能

③ 黄色的遮挡层跟随鼠标功能。

④ 移动黄色遮挡层，大图片跟随移动功能

**step two**

① 鼠标经过小图片盒子， 黄色的遮挡层 和 大图片盒子显示，离开隐藏2个盒子功能

② 就是显示与隐藏

**step three**

① 黄色的遮挡层跟随鼠标功能。

② 把鼠标坐标给遮挡层不合适。因为遮挡层坐标以父盒子为准。

③ 首先是获得鼠标在盒子的坐标。

④ 之后把数值给遮挡层做为left 和top值。

⑤ 此时用到鼠标移动事件，但是还是在小图片盒子内移动。

⑥ 但发现遮挡层位置不对即遮挡层的左上角跟随鼠标移动不是中间位置跟随鼠标移动，需要再减去盒子自身高度和宽度的一半。

⑦ 遮挡层不能超出小图片盒子范围。

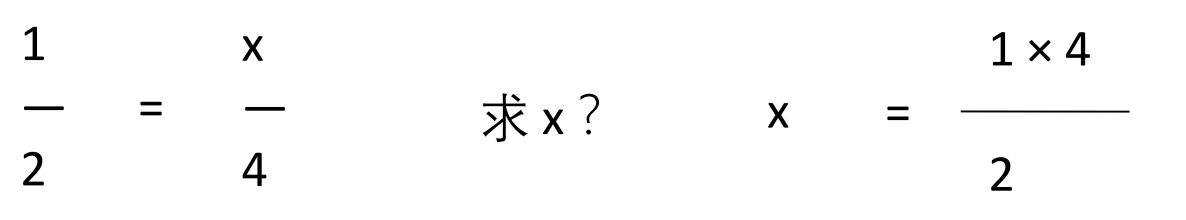
⑧ 如果小于零，就把坐标设置为0

⑨ 如果大于遮挡层最大的移动距离，就把坐标设置为最大的移动距离

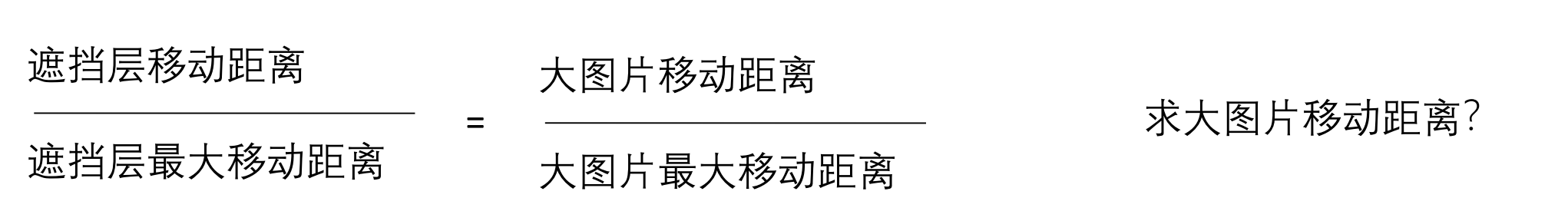
⑩ 遮挡层的最大移动距离： 小图片盒子宽度 减去 遮挡层盒子宽度

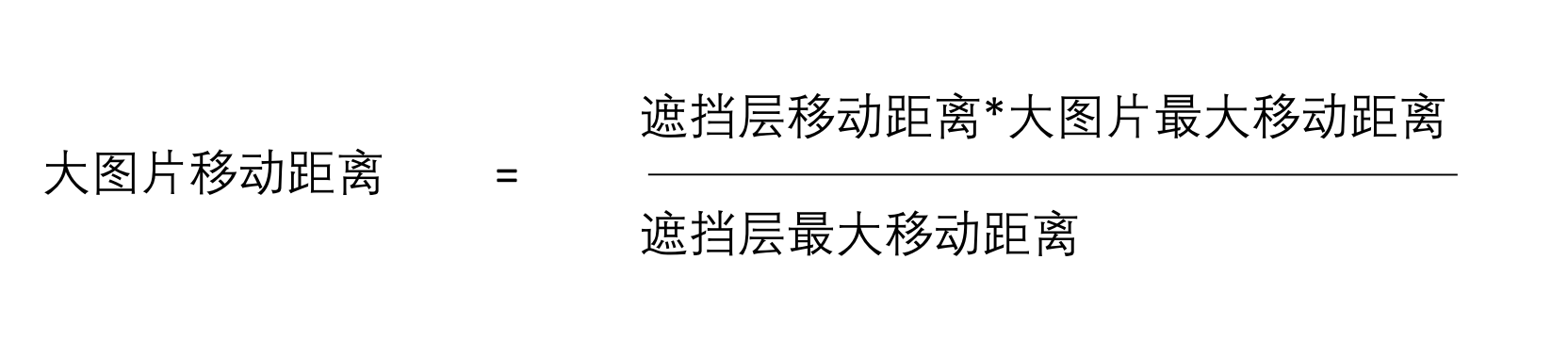
**step four**

**① 移动黄色遮挡层，大图片跟随移动功能。**



**② 求大图片的移动距离公式**





**2. 元素可视区 client 系列**

client 翻译过来就是客户端，我们使用 client 系列的相关属性来获取元素可视区的相关信息。通过 client 系列

的相关属性可以动态的得到该元素的边框大小、元素大小等。



