第1章 JavaScript基础事件以及Event对象

课程提要

- 事件的定义
- EventTarget 接收事件接口
- JS中常用的事件
- 事件对象
- 网页中常用坐标

1.1 事件的定义

事件指的是文档或者浏览器窗口中发生的一些特定交互瞬间。

事件是可以被 JavaScript监测到的行为。JavaScrip中的每个DOM节点都可以触发事件，如我们点击页面上的某个按钮，那么你就触发了某个按钮的点击事件。我们每个节点都有自己的事件，可以把节点比喻人，事件就是我们人的能力，例如说话，例如走路。

1.2 EventTarget 接收事件接口

EventTarget 是一个由可以接收事件的对象实现的接口，并且可以为它们创建侦听器。

DOM 的事件操作（监听和触发），都定义在EventTarget接口。所有节点对象都部署了这个接口，其他一些需要事件通信的浏览器内置对象（比如，XMLHttpRequest、AudioNode、AudioContext）也部署了这个接口。

该接口主要提供三个实例方法：

- addEventListener：绑定事件的监听函数
- removeEventListener：移除事件的监听函数
- dispatchEvent：触发事件

1.2.1 EventTarget.addEventListener()

addEventListener()用于在当前节点或对象上，定义一个特定事件的监听函数。一旦这个事件发生，就会执行监听函数。该方法没有返回值。

语法：

```
target.addEventListener = (type, listener[, useCapture])
```

解释：

- type:（必需）事件名，大小写敏感。
- listener：（必需）监听函数。事件发生时，会调用该监听函数。
- useCapture：（可选）布尔值，表示监听函数是否在捕获阶段（capture）触发，默认为false（监听函数只在冒泡阶段被触发）。

实例：

```
document.getElementById("btn1").addEventListener("click",hello);
function hello(){
    alert("hello world!");
}
```

1.2.2 EventTarget.removeEventListener()

EventTarget.removeEventListener()方法用来移除addEventListener方法添加的事件监听函数。该方法没有返回值。

语法：

```
target.removeEventListener(type, listener[, useCapture]);
```

解释：

- type：（必需）事件名，表示需要移除的事件类型，如 "click"。
- listener：（必需），需要移除的 EventListener 函数（先前使用 addEventListener 方法定义的）。
- useCapture：（可选）指定事件是否在捕获或冒泡阶段执行。true，捕获。false，冒泡。默认false。

注意：

removeEventListener方法移除的监听函数，必须是addEventListener方法添加的那个监听函数，而且必须在同一个元素节点，否则无效。

实例：

```
var btn = document.getElementById("btn");
btn.addEventListener("click",good);//执行了
btn.addEventListener("click",good2);
btn.removeEventListener("click",good2);//不执行
```

1.2.3 EventTarget.dispatchEvent()

EventTarget.dispatchEvent()方法在当前节点上触发指定事件，从而触发监听函数的执行。该方法返回一个布尔值，只要有一个监听函数调用了event.preventDefault()，则返回值为false，否则为true。

语法：

```
target.dispatchEvent(event)
```

注释：dispatchEvent方法的参数是一个event对象的实例，详情在后面介绍

实例：

```
//document上绑定自定义事件oneating
document.addEventListener('oneating', hello, false);
//创建event的对象实例
var event = new Event('oneating');
//触发自定义事件oneating
document.dispatchEvent(event);
function hello(){
    alert("hello,中国！");
}
```

判断事件是否被取消：

```
var canceled = !cb.dispatchEvent(event);
if (canceled) {
  console.log('事件取消');
} else {
  console.log('事件未取消');
}
```

1.3 JS中常用的事件

1.3.1 UI事件

UI事件中UI即(User Interface,用户界面)，当用户与页面的元素交互时触发。UI事件中主要包括load,unload,abort,error,select,resize,scroll事件。

1.3.1.1 load事件

当页面完全加载完之后（包括所有的图像、js文件、css文件等外部资源），就会触发window上面的load事件。

这个事件是JavaScript中最常用的事件，比如我们经常会使用window.onload=function(){};这种形式，即当页面完全加载完之后执行其中的函数。

实例：

```
window.onload=function(){
    alert("loaded");
}
```

1.3.1.2 unload事件

unload事件与load事件相对的。在文档被完全卸载后触发。用户从一个页面切换到另一个页面就会触发unload事件。利用这个事件最多的情况是清除引用，避免内存泄漏。

值得注意的是，一定要小心编写unload事件中的代码，因为它是在页面卸载后才触发，所以说页面加载后存在的那些对象，在unload触发之后就不一定存在了！

实例：

```
<body unload="alert('changed')">
```

1.3.1.3 resize事件

当调整浏览器的窗口到一个新的宽度或高度时，就会触发resize事件。这个事件在window（窗口）上面触发。因此同样可以通过JS或者body元素中的onresize特性来指定处理程序。

实例：

```
<body onresize="alert('changed')">// 浏览器的大小发生改变时就会弹出窗口
```

1.3.1.4 scroll事件

scroll事件会在文档被滚动期间重复被触发，所以应当尽量保持事件处理程序的代码简单。

实例：

```
window.onscroll = function() {
     console.info(window.scrollY);  //获取页面滚动出去的距离
}
```

1.3.2 焦点事件

焦点事件会在页面元素获得或失去焦点时触发。主要有：blur 、focus、focusin、focusout。

1.3.2.1 blur事件

blur事件在元素失去焦点时触发。

实例：

```
姓名：<input type="text" id="fname" onblur="toBlur()"/>
<script>
    function toBlur()
    {
        console.log("输入框失去焦点");
    }
</script>
```

1.3.2.2 focus事件

focus事件在元素获得焦点时触发。

实例：

```
<input type="text" id="fname" onfocus="toFocus()"/>
```

1.3.2.3 focusin事件

focusin事件在元素获得焦点时触发。与 focus() 方法不同的是，focusin() 方法在任意子元素获得焦点时也会触发。

该方法通常与focusout()方法一起使用。

实例：

```
<div style="width:400px;height:300px;border:1px solid red;"
onfocusin="myFunction(this)">
    输入您的名字: <input type="text">
    <p>当 input 输入框获取焦点时，JavaScript 函数将被触发，并修改div背景颜色。 </p>
</div>
<script>
    function myFunction(x) {
        x.style.background = "yellow";
    }
</script>
```

1.3.2.4 focusout事件

focusout事件在元素失去焦点时触发，与blur() 方法不同的是，focusout() 方法在任意子元素失去焦点时也会触发。

该方法通常与focusin()方法一起使用。

实例：

```
<div style="width:400px;height:300px;border:1px solid red;"
onfocusout="myFunction(this)">
    输入您的名字: <input type="text">
    <p>当 input 输入框失去焦点时，JavaScript 函数将被触发，并修改div背景颜色。 </p>
</div>
<script>
    function myFunction(x) {
        x.style.background = "red";
    }
</script>
```

1.3.3 鼠标事件

鼠标事件是Web开发中最常用的一类事件，因为鼠标是最主要的定位设备。

1.3.3.1 click事件

用户单击鼠标左键或按下回车键触发。

实例：

```
<div onclick="alert('您点击了div盒子')"></div>
```

1.3.3.2 dblclick事件

用户双击鼠标左键触发。

实例：

```
<div ondblclick="alert('您双击了div盒子')"></div>
```

1.3.3.3 mousedown事件

在用户按下了任意鼠标按钮时触发。

实例：

```
<div onmousedown="alert('您按下了鼠标按钮')"></div>
```

1.3.3.4 mouseenter事件

在鼠标光标从元素外部首次移动到元素范围内时触发。

实例：

```
<div onmouseenter="alert('mouseenter')"></div>
```

1.3.3.5 mouseleave事件

元素上方的光标移动到元素范围之外时触发。

实例：

```
<div onmouseleave="alert('mouseleave')"></div>
```

1.3.3.6 mousemove事件

光标在元素的内部不断的移动时触发。

实例：

```
<div onmousemove="console.log('光标在元素的内部移动')"></div>
```

1.3.3.7 mouseover事件

鼠标指针位于一个元素外部，然后用户将首次移动到另一个元素边界之内时触发。

实例：

```
<div onmouseover="console.log('您触发了mouseover事件')"></div>
```

1.3.3.8 mouseout事件

用户将光标从一个元素上方移动到另一个元素时触发。

实例：

```
<div onmouseout="console.log('您触发了mouseout事件')"></div>
```

1.3.3.9 mouseup事件

在用户释放鼠标按钮时触发。

实例：

```
<div onmouseup="console.log('您触发了mouseup事件')"></div>
```

1.3.4 鼠标滚轮事件

当用户通过鼠标滚轮在垂直方向上滚动页面时（向上或向下），就会触发mousewheel事件。这个事件可以在任何元素上触发，但最终都会冒泡到document或window对象上面。所有的现代浏览器都支持鼠标滚轮事件，并且在用户滚动滚轮时触发事件。

当需获取页面的滚动方向，滚动幅度等信息时，“mousewheel” 事件中的 “event.wheelDelta” 属性值可以获得。

wheelDelta：当用户向上滚动鼠标滚轮时，wheelDelta属性的值是120的倍数；当用户向下滚动鼠标滚轮时，wheelDelta属性的值是-120的倍数。

注意：FireFox不支持mousewheel，但FireFox使用DOMMouseScroll事件。

实例：

```
document.onmousewheel=function(){
    alert(event.wheelDelta);
};
```

1.3.5 键盘事件

1.3.5.1 keydown

keydown 事件会在用户按下一个键盘按键时发生；按住不放，会重复触发。

实例：

```
<input type="text" onkeydown="myFunction()">
```

1.3.5.2 keypress

kypress事件会在用户按下键盘上的字符键时触发，按住不放，会重复触发。

注意： keypress 事件在所有浏览器中不能触发所有按键(例如：ALT, CTRL, SHIFT, ESC) 。如果只对用户是否已经按下一个按键检测， 可以使用 onkeydown 取代, onkeydown被所有按键触发。

实例：

```
<input type="text" onkeypress="myFunction()">
```

1.3.5.3 keyup

keyup事件会在用户释放键盘上的按键时触发。

实例：

```
<input type="text" onkeyup="myFunction()">
```

1.4 事件对象

1.4.1 事件对象定义

事件发生以后，会产生一个事件对象，作为参数传给监听函数。浏览器原生提供一个Event对象，所有的事件都是这个对象的实例，或者说继承了Event.prototype对象。

Event对象本身就是一个构造函数，可以用来生成新的实例。

```
new Event(type, options);
```

注释：

- type：是字符串，表示事件的名称；
- options：是一个对象，表示事件对象的配置。该对象主要有下面两个属性：
- bubbles：布尔值，可选，默认为false，表示事件对象是否冒泡。
- cancelable：布尔值，可选，默认为false，表示事件是否可以被取消，即能否用event.preventDefault()取消这个事件。一旦事件被取消，就好像从来没有发生过，不会触发浏览器对该事件的默认行为。

实例：

```
var ev = new Event(
    'look',
    {
        'bubbles': true,
        'cancelable': false
    }
);
document.addEventListener('look',callback);
document.dispatchEvent(ev);
function callback(){
    alert("你好，中国！");
}
```

1.4.2 事件对象的常用属性和方法

1.4.2.1 event.currentTarget和event.target

- event.currentTarget属性：返回事件当前所在的节点，即正在执行的监听函数所绑定的那个节点。
- event.target属性：返回原始触发事件的那个节点，即事件最初发生的节点。

区别：

事件传播过程中，不同节点的监听函数内部的event.target与event.currentTarget属性的值是不一样的，前者总是不变的，后者则是指向监听函数所在的那个节点对象。

实例：

```
function hide(e) {
    console.log(this === e.currentTarget);  // 总是 true
    console.log(this === e.target);  // 有可能不是 true
    e.target.style.visibility = 'hidden';
}
var para = document.getElementById("para");
para.addEventListener('click', hide, false);
```

1.4.2.2 event.type

event.type属性返回一个字符串，表示事件类型。该属性只读。

实例：

```
var evt = new Event('toAlert');
console.log(evt.type) // "toAlert"
```

1.4.2.3 event.eventPhase

event.eventPhase属性返回一个整数常量，表示事件目前所处的阶段。该属性只读。

返回值：

- 0：事件目前没有发生。
- 1：事件目前处于捕获阶段，即处于从祖先节点向目标节点的传播过程中。
- 2：事件到达目标节点，即event.target属性指向的那个节点。
- 3：事件处于冒泡阶段，即处于从目标节点向祖先节点的反向传播过程中。

实例：

```
var btn = document.getElementById("btn");
function toAlert(e){
    alert("事件目前所处第" + e.eventPhase + "阶段");
}
btn.addEventListener("click",toAlert);
```

1.4.2.4 event. cancelable

event.cancelable属性返回一个布尔值，表示事件是否可以取消。该属性为只读属性，一般用来了解 Event 实例的特性。

大多数浏览器的原生事件是可以取消的。比如，取消click事件，点击链接将无效。但是除非显式声明，event构造函数生成的事件，默认是不可以取消的。

当event.cancelable属性为true时，调用event.preventDefault()就可以取消这个事件，阻止浏览器对该事件的默认行为。

如果事件不能取消，调用event.preventDefault()会没有任何效果。所以使用这个方法之前，最好用event.cancelable属性判断一下是否可以取消。

实例：

```
var evt = new Event('foo');
console.log(evt.cancelable);
```

1.4.2.4 event.preventDefault()

event.preventDefault方法取消浏览器对当前事件的默认行为。比如点击链接后，浏览器默认会跳转到另一个页面，使用这个方法以后，就不会跳转了；再比如，按一下空格键，页面向下滚动一段距离，使用这个方法以后也不会滚动了。该方法生效的前提是，事件对象的cancelable属性为true，如果为false，调用该方法没有任何效果。

注意：该方法只是取消事件对当前元素的默认影响，不会阻止事件的传播。如果要阻止传播，可以使用stopPropagation()方法。

实例：

```
var cb = document.getElementById('my-checkbox');
cb.addEventListener(
    'click',
    function (e){ e.preventDefault(); },
    false
);
```

1.4.2.5 event.stopPropagation()

stopPropagation方法阻止事件在 DOM 中继续传播，防止再触发定义在别的节点上的监听函数，但是不包括在当前节点上其他的事件监听函数。

实例：

```
function stopEvent(e) {
    e.stopPropagation();
    console.log(e.currentTarget);
}
```

1.4.3 事件对象的其他属性和方法

event对象属性：

属性	描述
altKey	返回当事件被触发时，”ALT” 是否被按下。
butto	返回当事件被触发时，哪个鼠标按钮被点击。
clientX	返回当事件被触发时，鼠标指针的水平坐标。
clientY	返回当事件被触发时，鼠标指针的垂直坐标。
ctrlKey	返回当事件被触发时，”CTRL” 键是否被按下。
metaKey	返回当事件被触发时，”meta” 键是否被按下。
relatedTarget	返回与事件的目标节点相关的节点。
screenX	返回当某个事件被触发时，鼠标指针的水平坐标。
screenY	返回当某个事件被触发时，鼠标指针的垂直坐标。
shiftKey	返回当事件被触发时，”SHIFT” 键是否被按下。

event对象IE属性：

属性	描述
cancelBubble	如果事件句柄想阻止事件传播到包容对象，必须把该属性设为 true。
fromElement	对于 mouseover 和 mouseout 事件，fromElement 引用移出鼠标的元素。
keyCode	对于 keypress 事件，该属性声明了被敲击的键生成的 Unicode 字符码。对于 keydow 和 keyup
offsetX,offsetY	发生事件的地点在事件源元素的坐标系统中的 x 坐标和 y 坐标。
returnValue	如果设置了该属性，它的值比事件句柄的返回值优先级高。把这个属性设置为
srcElement	对于生成事件的 Window 对象、Document 对象或 Element 对象的引用。
toElement	对于 mouseover 和 mouseout 事件，该属性引用移入鼠标的元素。
x,y	事件发生的位置的 x 坐标和 y 坐标，它们相对于用CSS动态定位的最内层包容元素。

DOM2级事件对象属性：

属性和方法	描述
bubbles	返回布尔值，指示事件是否是起泡事件类型。
timeStamp	返回事件生成的日期和时间。
initEvent()	初始化新创建的 Event 对象的属性。

1.4 事件对象的兼容性

我们需要在老版本浏览器里被迫处理的IE Model，在很多方面和DOM Model都不相同。在DOM Model中，Event对象实例是作为第一个参数传入到处理程序中的；而在IE Model中，则是通过全局上下文（window.event）的event属性来获取到的。在这两个模型中，Event实例的内容是不相同的。

Event对象的兼容性写法：

```
获得event对象兼容性写法 ：event || (event = window.event);
获得target兼容型写法 ：event.target||event.srcElement
```

阻止浏览器默认行为兼容性写法 ：

```
event.preventDefault ? event.preventDefault() : (event.returnValue = false);
```

阻止冒泡写法 ：

```
event.stopPropagation ? event.stopPropagation() : (event.cancelBubble = true);
```

1.5 网页中常用坐标

1.5.1 获取屏幕的宽高

- screen.width //屏幕的宽
- screen.height //屏幕的高
- screen.availWidth //屏幕可用宽度 屏幕的像素高度减去系统部件高度之后的值
- screen.availHeight //屏幕可用高度 屏幕的像素宽度减去系统部件宽度之后的值

1.5.2 获得窗口位置及大小

- window.screenTop //窗口顶部距屏幕顶部的距离
- window.screenLeft //窗口左侧距屏幕左侧的距离
- window.innerWidth //窗口中可视区域(viewpoint)的宽
- window.innerHeight //窗口中可视区域(viewpoint)的高度 该值与浏览器是否显示菜单栏等因素有关
- window.outerWidth //浏览器窗口本身的宽度(可视区域宽度+浏览器边框宽度)
- window.outerHeight //浏览器窗口本身的高度

注意：

chrome在最大化时浏览器窗口没有边框宽度,非最大化时有8px边框

FF和IE上下左右有8px的边框宽度

1.5.3 元素对象的信息

在 HTML DOM 中, 元素对象代表着一个 HTML 元素。元素对象 的 子节点可以是元素节点，文本节点，注释节点。

默认盒模型 box-sizing:content-box;

示例1：当不出现滚动条时。

```
body{margin:0;}
#box{
    width:100px;
    height:100px;
    padding:10px;
    border:20px solid red;
    margin:30px;
    background-color:red;
}
<div id="box">前端开发</div>
```

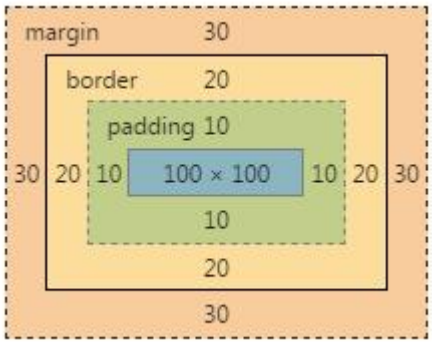

图 1-1盒子大小

盒子的大小：

```
boxWidth = 2*margin + 2*border + 2*padding + width
boxHeight = 2*margin + 2*border + 2*padding + height

clientWidth ：在页面上返回内容的可视宽度（不包括边框，边距或滚动条）

clientWidth = 2*padding + width - scrollbarWidth

clientHeight：在页面上返回内容的可视高度（不包括边框，边距或滚动条）clientHeight = 2*padding +
height - scrollbarHeight

offsetWidth ：返回元素的宽度包括边框和填充，但不包括边距

offsetWidth = 2*border + 2*padding + width

offsetHeight ：返回元素的高度包括边框和填充，但不包括边距

offsetHeight = 2*border + 2*padding + height

offsetLeft： 获取对象相对于版面或由 offsetLeft属性指定的父坐标的计算左侧位置

offsetTop ： 获取对象相对于版面或由 offsetTop属性指定的父坐标的计算顶端位置
```

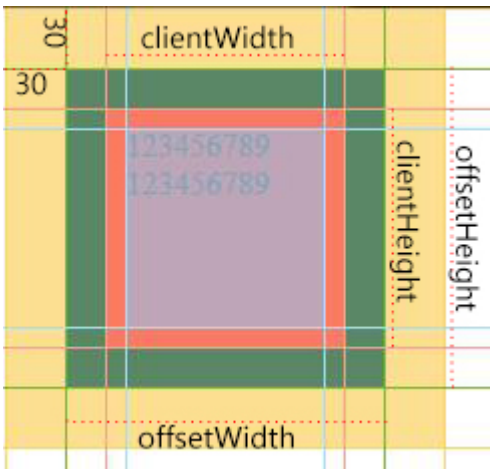

图 1-2 盒子大小

示例2：当出现滚动条时。

```
body{
margin:0;
    padding:20px;
    width:1000px;
    height:500px;
}
#box{
    width:100px;
    height:100px;
    padding:10px;
    border:20px solid red;
    margin:30px;
    background-color:red;
}
<div id="box">前端开发</div>
```

- scrollWidth ：返回元素的整个宽度（包括带滚动条的隐蔽的地方）
- scrollWidth = 2*padding + width
- scrollHeight ：返回整个元素的高度（包括带滚动条的隐蔽的地方）
- scrollHeight = 2*padding + width
- scrollTop：向下滑动滚动块时元素隐藏内容的高度。不设置时默认为0，其值随着滚动块滚动而变化
- scrollLeft： 向右滑动滚动块时元素隐藏内容的宽度。不设置时默认为0，其值随着滚动块滚动而变化

1.5.4 event对象中的坐标信息

- event.pageX:相对整个页面的坐标，以页面的左上角为坐标原点到鼠标所在点的水平距离（IE8不支持）
- event.pageY:相对整个页面的坐标，以页面的左上角为坐标原点到鼠标所在点的垂直距离（IE8不支持）
- event.clientX:相对可视区域的坐标，以浏览器可视区域左上角为坐标原点到鼠标所在点的水平距离
- event.clientY:相对可视区域的坐标，以浏览器可视区域左上角为坐标原点到鼠标所在点的垂直距离
- event.screenX:相对电脑屏幕的坐标，以屏幕左上角为坐标原点到鼠标所在点的水平距离
- event.screenY:相对电脑屏幕的坐标，以屏幕左上角为坐标原点到鼠标所在点的垂直距离
- event.offsetX:相对于自身的坐标，以自身的padding左上角为坐标原点到鼠标所在点的水平距离
- event.offsetY:相对于自身的坐标，以自身的padding左上角为坐标原点到鼠标所在点的水平距离

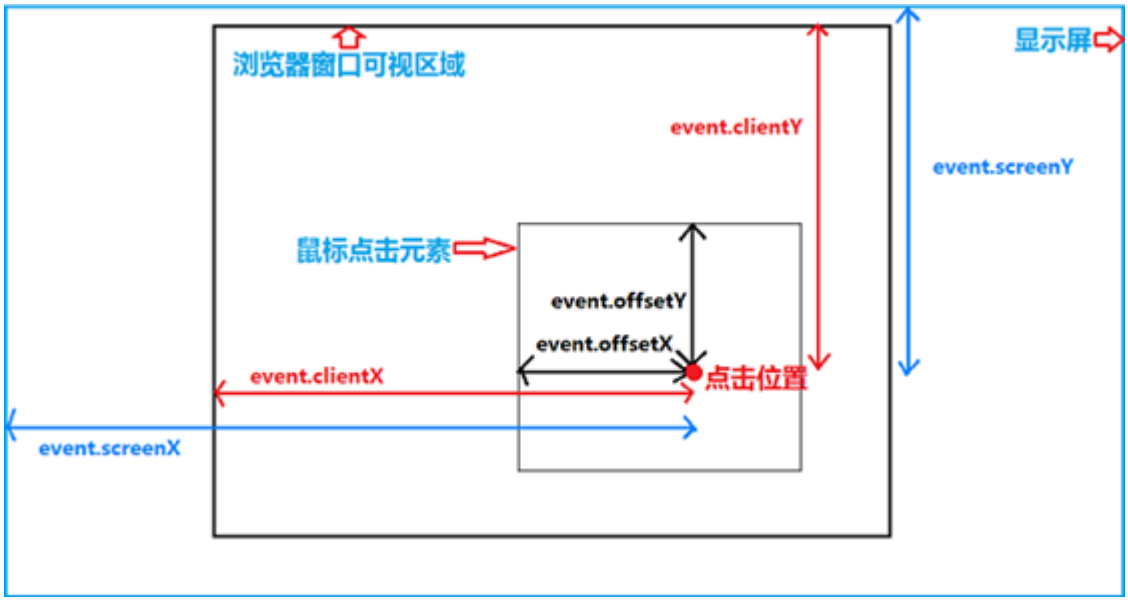

图 1-3 各种坐标信息

1.6 课程总结

- 事件的定义
- EventTarget 接收事件接口
- JS中常用的事件
- 事件对象
- 网页中常用坐标

1.7 实训

1、创建一个div块，向上滚动鼠标滚轮减少div块的高度，向下滚动鼠标滚轮增加div块的高度。

2、创建一个div块，用鼠标事件实现拖拽功能。需求：

1）当鼠标放在div块上时，按下鼠标左键并移动鼠标，div块随之运动；

2）当松开鼠标左键，div块运动停止。