事件高级

注册事件（绑定事件）

概述

给元素添加事件，称为注册事件或者绑定事件

注册事件有两种方式：传统方式和方法监听注册方式

1、传统方式

利用 on 开头的事件 onclick

特点： 注册事件的唯一性

同一个元素同一个事件只能设置一个处理函数，最后注册的处理函数将会覆盖前面注册的处理函数

2、方法监听注册方式

w3c 标准 推荐方式

addEventListener() 它是一个方法

IE9 之前的 IE 不支持此方法，可使用 attachEvent() 代替

特点：同一个元素同一个事件可以注册多个监听器

按注册顺序依次执行

addEventListener

事件监听方式：eventTarget.addEventListener(type, listener[,useCapture])

将指定的监听器注册到eventTarget（目标对象）上，当该对象触发指定的事件时，就会执行事件处理函数

type：事件类型，例如chlick、mouseover（没有on）

listener：事件处理函数，事件发生时，会调用该监听函数

useCapture：可选参数，是一个布尔值，默认是false

attachEvent

eventTarget.attachEvent(eventNameWithOn, callback)

将指定的监听器注册到 eventTarget（目标对象） 上，当该对象触 发指定的事件时，指定的回调函数就会被执行。

eventNameWithOn：事件类型字符串，比如 onclick 、onmouseover ，这里要带 on

callback： 事件处理函数，当目标触发事件时回调函数被调用

注意：IE8 及早期版本支持

注册事件兼容性解决方案

```
function addEventListener(element, eventName, fn) {
    // 判断当前浏览器是否支持 addEventListener 方法
    if (element.addEventListener) {
        element.addEventListener(eventName, fn); // 第三个参数 默认是false
    } else if (element.attachEvent) {
        element.attachEvent('on' + eventName, fn);
    } else {
        // 相当于 element.onclick = fn;
        element['on' + eventName] = fn;
    }
}
```

删除事件（解绑事件）

传统注册方法：

eventTarget.onclick = null;

方法监听注册方式：

① eventTarget.removeEventListener(type, listener[, useCapture]);

② eventTarget.detachEvent(eventNameWithOn, callback);

兼容性解决方法：

```
function removeEventListener(element, eventName, fn) {
    if (element.removeEventListener) {
        element.removeEventListener(eventName,fn);
    } else if (element.detachEvent) {
        element.detachEvent('on'+eventName,fn);
    } else {
        element['on'+eventName] = null;
    }
}
```

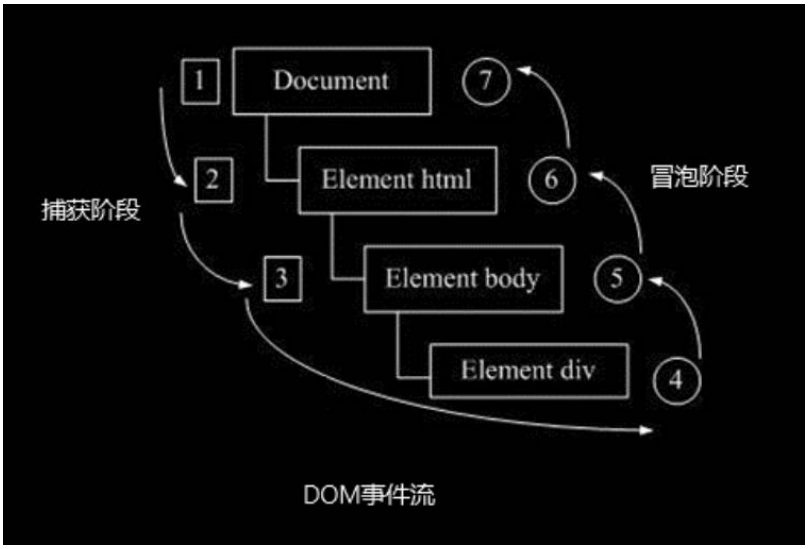
DOM事件流

事件流描述的是从页面中接收事件的顺序。

事件发生时会在元素节点之间按照特定的顺序传播，这个传播过程即 DOM 事件流。

例如：给一个div 注册了点击事件

DOM 事件流分为3个阶段： 1. 捕获阶段 2. 当前目标阶段 3. 冒泡阶段



事件冒泡： IE 最早提出，事件开始时由最具体的元素接收，然后逐级向上传播到到 DOM 最顶层节点的过程。

事件捕获： 网景最早提出，由 DOM 最顶层节点开始，然后逐级向下传播到到最具体的元素接收的过程。

注意

1、js代码中只能执行捕获或者冒泡其中一个阶段

2、onclick和attachEvent只能得到冒泡阶段

3、addEventListener(type, listener[, useCapture])第三个参数如果是 true，表示在事件捕 获阶段调用事件处理程序；如果是 false（不写默认就是false），表示在事件冒泡阶段调用事件处理 程序。

4、实际开发中我们很少使用事件捕获，我们更关注事件冒泡。

5、有些事件是没有冒泡的，比如 onblur、onfocus、onmouseenter、onmouseleave

6、事件冒泡有时候会带来麻烦，有时候又会帮助很巧妙的做某些事件，我们后面讲解。

事件对象

概述

```
eventTarget.onclick = function(event) {}
eventTarget.addEventListener('click', function(event) {})
// 这个 event 就是事件对象，我们还喜欢的写成 e 或者 evt
```

event 对象代表事件的状态，比如键盘按键的状态、鼠标的位置、鼠标按钮的状态。

简单理解：事件发生后，跟事件相关的一系列信息数据的集合都放到这个对象里面，这个对象就是事件对象 event，它有很多属性和方法。

比如： 1. 谁绑定了这个事件。 2. 鼠标触发事件的话，会得到鼠标的相关信息，如鼠标位置。 3. 键盘触发事件的话，会得到键盘的相关信息，如按了哪个键。

使用

```
eventTarget.onclick = function(event) {
// 这个 event 就是事件对象，我们还喜欢的写成 e 或者 evt
}
eventTarget.addEventListener('click', function(event) {
// 这个 event 就是事件对象，我们还喜欢的写成 e 或者 evt
})
```

这个 event 是个形参，系统帮我们设定为事件对象，不需要传递实参过去。 当我们注册事件时， event 对象就会被系统自动创建，并依次传递给事件监听器（事件处理函数）。

兼容性方案

事件对象本身的获取存在兼容问题：

1. 标准浏览器中是浏览器给方法传递的参数，只需要定义形参 e 就可以获取到。
2. 在 IE6~8 中，浏览器不会给方法传递参数，如果需要的话，需要到 window.event 中获取查找。

解决：e = e || window.event;

事件对象的常见属性与方法

e.target和this的区别：

this是**事件绑定**的元素，这个函数的调用者（绑定这个事件的元素）

e.target是**事件触发**的元素

事件对象属性方法	说明
e.target	返回触发事件的对象 标准
e.srcElement	返回触发事件的对象 非标准 ie6-8使用
e.type	返回事件的类型 比如 click mouseover 不带on
e.cancelBubble	该属性阻止冒泡 非标准 ie6-8使用
e.returnValue	该属性 阻止默认事件（默认行为） 非标准 ie6-8使用 比如不让链接跳转
e.preventDefault()	该方法 阻止默认事件（默认行为） 标准 比如不让链接跳转
e.stopPropagation()	阻止冒泡 标准

阻止事件冒泡

事件冒泡：开始时由最具体的元素接收，然后逐级向上传播到到 DOM 最顶层节点。 事件冒泡本身的特性，会带来的坏处，也会带来的好处，需要我们灵活掌握。

标准写法：利用事件对象里面的stopPropagation（）方法

e.stopPropagation()

非标准写法：IE6-8 利用事件对象cancelBubble属性

e.cancelBubble = true;

兼容性解决

```
if(e && e.stopPropagation){
e.stopPropagation();
}else{
window.event.cancelBubble = true;
}
```

事件委托（代理、委派）

事件冒泡本身的特性，会带来的坏处，也会带来的好处，需要我们灵活掌握。

生活中有如下场景： 咱们班有100个学生， 快递员有100个快递， 如果一个个的送花费时间较长。同时每个学生领取的时候，也需 要排队领取，也花费时间较长，何如？ 解决方案： 快递员把100个快递，委托给班主任，班主任把这些快递放到办公室，同学们下课自行领取即可。 优势： 快递员省事，委托给班主任就可以走了。 同学们领取也方便，因为相信班主任。

事件委托也称为事件代理，在jQuery里面称为事件委派

原理

不是每个子节点单独设置事件监听器，而是事件监听器设置在其父节点上，然后利用冒泡原理影响设置每个子节点。

作用

只操作了一次DOM，提高了程序的性能

常用的鼠标事件

鼠标事件	触发条件
onclick	鼠标点击左键触发
onmouseover	鼠标经过触发
onmouseout	鼠标离开触发
onfocus	获得鼠标焦点触发
onblur	失去鼠标焦点触发
onmousemove	鼠标移动触发
onmouseup	鼠标弹起触发
onmousedown	鼠标按下触发

禁止鼠标事件

1、禁止鼠标右键出现菜单

contextmenu主要控制应该何时显示上下文菜单，主要用于程序员取消默认的上下文菜单

```
document.addEventListener('contextmenu', function(e) {
    e.preventDefault();
})
```

2、禁止鼠标选中（selectstart 开始选中）

```
document.addEventListener('selectstart',function(e) {
    e.preventDefault();
})
```

鼠标事件对象

event对象代表事件的状态，跟事件相关的一系列信息的集合。现阶段我们主要是用鼠标事件对象 MouseEvent 和键盘事件对象 KeyboardEvent。

鼠标事件对象	说明
e.clientX	返回鼠标相对于浏览器窗口可视区的 X 坐标
e.clientY	返回鼠标相对于浏览器窗口可视区的 Y 坐标
e.pageX	返回鼠标相对于文档页面的 X 坐标 IE9+ 支持
e.pageY	返回鼠标相对于文档页面的 Y 坐标 IE9+ 支持
e.screenX	返回鼠标相对于电脑屏幕的 X 坐标
e.screenY	返回鼠标相对于电脑屏幕的 Y 坐标

案例：跟随鼠标的天使

鼠标移动事件：mousemove

只需要改变图片绝对定位的xy坐标就可以了

常用的键盘事件

键盘事件	触发条件
onkeyup	某个键盘按键被松开时触发
onkeydown	某个键盘按键被按下时触发
onkeypress	某个键盘按键被按下时 触发 但是它不识别功能键 比如 ctrl shift 箭头等

onkeypress 和前面2个的区别是，它不识别功能键，比如左右箭头，shift 等。

三个事件的执行顺序是： keydown -- keypress --- keyup

onkeydown 和 onkeyup 不区分字母大小写，onkeypress 区分字母大小写。

键盘事件对象

keyCode 返回该键的ASCII值

在我们实际开发中，我们更多的使用keydown和keyup， 它能识别所有的键（包括功能键） Keypress 不识别功能键，但是keyCode属性能区分大小写，返回不同的ASCII值

案例：模拟京东案件输入内容

按下s键，光标就定位到搜索框

keyup（不使用keydown是因为按下就检测到的话会在搜索框中留下s，而keyup在松开之后检测，就不会在搜索框留下s键）

搜索框获得焦点：使用js里面的focus（）方法

案例：模拟京东快递单号查询

难点：把搜索框中的值获取到赋值给提示框作为内容

如果搜索框里为空，则隐藏提示框