



黑马程序员™  
[www.itheima.com](http://www.itheima.com)

传智播客旗下  
高端IT教育品牌

# 事件高级



# 目录 Contents

- ◆ 注册事件（绑定事件）
- ◆ 删除事件（解绑事件）
- ◆ DOM事件流
- ◆ 事件对象
- ◆ 阻止事件冒泡
- ◆ 事件委托（代理、委派）
- ◆ 常用的鼠标事件
- ◆ 常用的键盘事件

# 1. 注册事件（绑定事件）

## 1.1 注册事件概述

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**

### 传统注册方式

- 利用 on 开头的事件 onclick
- `<button onclick= "alert('hi~') " ></button>`
- `btn.onclick = function() {}`
- 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

### 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

# 1. 注册事件（绑定事件）

## 1.2 addEventListener 事件监听方式

```
eventTarget.addEventListener(type, listener[, useCapture])
```

`eventTarget.addEventListener()` 方法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。

该方法接收三个参数：

- **type**：事件类型字符串，比如 `click`、`mouseover`，注意这里不要带 `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`。学完 DOM 事件流后，我们进一步学习

# 1. 注册事件（绑定事件）

## 1.3 attachEvent 事件监听方式

```
eventTarget.attachEvent(eventNameWithOn, callback)
```

`eventTarget.attachEvent()` 方法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。

该方法接收两个参数：

- **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

# 1. 注册事件（绑定事件）

## 1.4 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;  
    }  
}
```

兼容性处理的原则：首先照顾大多数浏览器，再处理特殊浏览器

# 目录 Contents

- ◆ 注册事件（绑定事件）
- ◆ 删除事件（解绑事件）
- ◆ DOM事件流
- ◆ 事件对象
- ◆ 阻止事件冒泡
- ◆ 事件委托（代理、委派）
- ◆ 常用的鼠标事件
- ◆ 常用的键盘事件

## ■ 2. 删除事件（解绑事件）

### 2.1 删除事件的方式

#### 1. 传统注册方式

```
eventTarget.onclick = null;
```

#### 2. 方法监听注册方式

- ① `eventTarget.removeEventListener(type, listener[, useCapture]);`
- ② `eventTarget.detachEvent(eventNameWithOn, callback);`



## ■ 2. 删除事件（解绑事件）

### 2.2 删除事件兼容性解决方案

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

# 目录 Contents

- ◆ 注册事件（绑定事件）
- ◆ 删除事件（解绑事件）
- ◆ DOM事件流
- ◆ 事件对象
- ◆ 阻止事件冒泡
- ◆ 事件委托（代理、委派）
- ◆ 常用的鼠标事件
- ◆ 常用的键盘事件

## 3. DOM 事件流

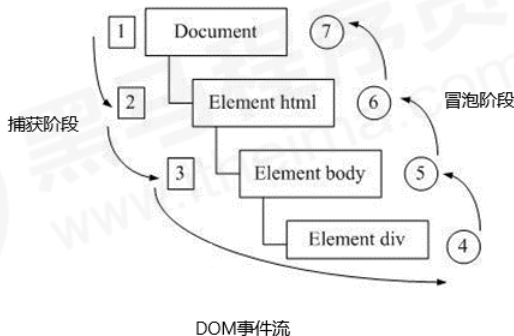
事件流描述的是从页面中接收事件的顺序。

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。

比如我们给一个div 注册了点击事件：

DOM 事件流分为3个阶段：

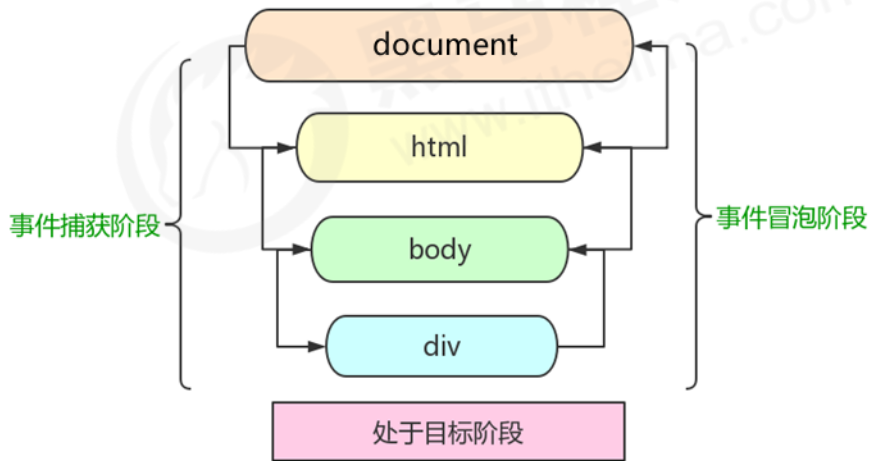
1. 捕获阶段
2. 当前目标阶段
3. 冒泡阶段



- 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 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最具体的元素接收的过程。

### 3. DOM 事件流

我们向水里面扔一块石头，首先它会有一个下降的过程，这个过程就可以理解为从最顶层向事件发生的最具体元素（目标点）的捕获过程；之后会产生泡泡，会在最低点（最具体元素）之后漂浮到水面上，这个过程相当于事件冒泡。



## 3. DOM 事件流

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。

### 注意

1. JS 代码中只能执行捕获或者冒泡其中的一个阶段。
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。

# 目录 Contents

- ◆ 注册事件（绑定事件）
- ◆ 删除事件（解绑事件）
- ◆ DOM事件流
- ◆ 事件对象
- ◆ 阻止事件冒泡
- ◆ 事件委托（代理、委派）
- ◆ 常用的鼠标事件
- ◆ 常用的键盘事件

## 4. 事件对象

### 4.1 什么是事件对象

```
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
```

官方解释：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。

## 4. 事件对象

### 4.2 事件对象的使用语法

```
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）。



## 4. 事件对象

### 4.3 事件对象的兼容性方案

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;
```

## 4. 事件对象

### 4.4 事件对象的常见属性和方法

`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。



## 4. 事件对象

### 4.4 事件对象的常见属性和方法

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

# 目录 Contents

- ◆ 注册事件（绑定事件）
- ◆ 删除事件（解绑事件）
- ◆ DOM事件流
- ◆ 事件对象
- ◆ 阻止事件冒泡
- ◆ 事件委托（代理、委派）
- ◆ 常用的鼠标事件
- ◆ 常用的键盘事件

## 5. 阻止事件冒泡

### 5.1 阻止事件冒泡的两种方式

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 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。

#### 阻止事件冒泡

- 标准写法：利用事件对象里面的 `stopPropagation()` 方法

```
e.stopPropagation()
```

- 非标准写法：IE 6-8 利用事件对象 `cancelBubble` 属性

```
e.cancelBubble = true;
```

## ■ 5. 阻止事件冒泡

### 5.2 阻止事件冒泡的兼容性解决方案

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

# 目录 Contents

- ◆ 注册事件（绑定事件）
- ◆ 删除事件（解绑事件）
- ◆ DOM事件流
- ◆ 事件对象
- ◆ 阻止事件冒泡
- ◆ 事件委托（代理、委派）
- ◆ 常用的鼠标事件
- ◆ 常用的键盘事件

## 6. 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。



## 6. 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。程序中也有如此场景：

```
<ul>
  <li>知否知否，应该有弹框在手</li>
  <li>知否知否，应该有弹框在手</li>
  <li>知否知否，应该有弹框在手</li>
  <li>知否知否，应该有弹框在手</li>
  <li>知否知否，应该有弹框在手</li>
</ul>
```

点击每个 li 都会弹出对话框，以前需要给每个 li 注册事件，是非常辛苦的，而且访问 DOM 的次数越多，这就会延长整个页面的交互就绪时间。

## 6. 事件委托（代理、委派）

### 事件委托

事件委托也称为事件代理，在 jQuery 里面称为事件委派。

### 事件委托的原理

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。

以上案例：给 ul 注册点击事件，然后利用事件对象的 target 来找到当前点击的 li，因为点击 li，事件会冒泡到 ul 上，ul 有注册事件，就会触发事件监听器。

### 事件委托的作用

我们只操作了一次 DOM，提高了程序的性能。

# 目录 Contents

- ◆ 注册事件（绑定事件）
- ◆ 删除事件（解绑事件）
- ◆ DOM事件流
- ◆ 事件对象
- ◆ 阻止事件冒泡
- ◆ 事件委托（代理、委派）
- ◆ 常用的鼠标事件
- ◆ 常用的键盘事件

## 7. 常用的鼠标事件

### 7.1 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

# 7. 常用的鼠标事件

## 7.1 常用的鼠标事件

### 1. 禁止鼠标右键菜单

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();  
})
```

## 7. 常用的鼠标事件

### 7.2 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
<code>e.clientX</code>	返回鼠标相对于浏览器窗口可视区的 X 坐标
<code>e.clientY</code>	返回鼠标相对于浏览器窗口可视区的 Y 坐标
<code>e.pageX</code>	返回鼠标相对于文档页面的 X 坐标 IE9+ 支持
<code>e.pageY</code>	返回鼠标相对于文档页面的 Y 坐标 IE9+ 支持
<code>e.screenX</code>	返回鼠标相对于电脑屏幕的 X 坐标
<code>e.screenY</code>	返回鼠标相对于电脑屏幕的 Y 坐标

## 7. 常用的鼠标事件



### 案例：跟随鼠标的天使

这个天使图片一直跟随鼠标移动



## 7. 常用的鼠标事件



### 案例分析

- ① 鼠标不断的移动，使用鼠标移动事件：mousemove
- ② 在页面中移动，给document注册事件
- ③ 图片要移动距离，而且不占位置，我们使用绝对定位即可
- ④ 核心原理：每次鼠标移动，我们都会获得最新的鼠标坐标，把这个x和y坐标做为图片的top和left 值就可以移动图片



## 7. 常用的鼠标事件



### 实现代码

```
var pic = document.querySelector('img');  
document.addEventListener('mousemove', function(e) {  
    var x = e.pageX;  
    var y = e.pageY;  
    pic.style.top = y - 40 + 'px';  
    pic.style.left = x - 50 + 'px';  
})
```

# 目录 Contents

- ◆ 注册事件（绑定事件）
- ◆ 删除事件（解绑事件）
- ◆ DOM事件流
- ◆ 事件对象
- ◆ 阻止事件冒泡
- ◆ 事件委托（代理、委派）
- ◆ 常用的鼠标事件
- ◆ 常用的键盘事件

## 8. 常用的键盘事件

### 8.1 常用键盘事件

事件除了使用鼠标触发，还可以使用键盘触发，注意给文档 `document` 添加键盘事件

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并弹起时触发

**注意：** `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。

2013/08/08

## 8. 常用的键盘事件

### 8.3 键盘事件对象

键盘事件对象	说明
keyCode	返回改键的ASCII 值

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

## 8. 常用的键盘事件



### 案例：模拟京东按键输入内容

当我们按下 s 键，光标就定位到搜索框



## 8. 常用的键盘事件



### 案例分析

- ① XXX
- ② XXX
- ③ XXX



## 8. 常用的键盘事件



实现代码

```
Var XXX;
```



## 8. 常用的键盘事件



### 案例：模拟京东快递单号查询

要求：当我们在文本框中输入内容时，文本框上面自动显示大字号的内容。

京东物流 - 查快递, 寄快递, 上快递100

公司名称

JD 京东物流

快递单号

请输入您的快递单号

查询

京东物流

京东

官网地址: [www.jdwl.com](http://www.jdwl.com)

客服电话: 950616

京东供应链物流

京东快递

快递100 数据来自快递100

## 8. 常用的键盘事件



### 案例分析

- ① XXX
- ② XXX
- ③ XXX



## 8. 常用的键盘事件



实现代码

```
Var XXX;
```



黑马程序员

[www.itheima.com](http://www.itheima.com)

传智播客旗下高端IT教育品牌