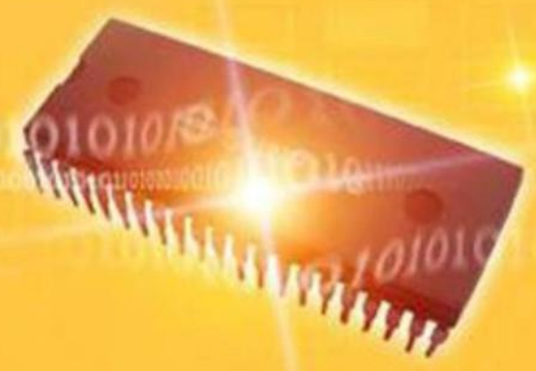
事件机制



教学目标

- 掌握事件的绑定以及移除方法
- 总结IE的事件机制方面与其他浏览器的不同点

✦ 一般事件

onclick	鼠标点击时触发此事件
ondblclick	鼠标双击时触发此事件
onmousedown	按下鼠标时触发此事件
onmouseup	鼠标按下后松开鼠标时触发此事件
onmouseover	当鼠标移动到某对象范围的上方时触发此事件
onmousemove	鼠标移动时触发此事件
onmouseout	当鼠标离开某对象范围时触发此事件
onkeypress	当键盘上的某个键被按下并且释放时触发此事件
onkeydown	当键盘上某个按键被按下时触发此事件
onkeyup	当键盘上某个按键被按放开时触发此事件

✦ 表单相事件

onblur	当前元素失去焦点时触发此事件
onchange	当前元素失去焦点并且元素的内容发生改变而触发此事件
onfocus	当某个元素获得焦点时触发此事件
onreset	当表单中RESET的属性被激发时触发此事件
onsubmit	一个表单被递交时触发此事件
oninput	实时监听输入框值变化（IE9以下不支持）
onpropertychange	同oninput类似，是IE专属的事件

例14-1

✦ 页面相关事件

onload	页面内容完成时触发此事件
onresize	当浏览器的窗口大小被改变时触发此事件
onscroll	浏览器的滚动条位置发生变化时触发此事件
oncontextmenu	当右键弹出上下文菜单时发生

曾经的事件绑定

```
<div>点击</div>
<script>
    var div=document.getElementsByTagName("div");
    div[0].onclick=function () {
        console.log("我是点击后的执行结果");
    }
</script>
```

例14-3

曾经的事件绑定

- 为同一对象绑定多个事件处理程序，所有程序都发生作用吗?

```
<div>点击</div>
<script>
    //为同一对象绑定多个事件处理程序，所有程序都发生作用吗?
    var div=document.getElementsByTagName("div");
    div[0].onclick=function () {
        console.log("第一个事件发生作用?");
    };
    div[0].onclick=function () {
        console.log("第二个事件发生作用?")
    }
</script>
```

例14-4

曾经的事件绑定

- 如果希望两个事件都能发挥作用怎么办？

  将两个事件分别封装到两个函数当中
  在单击事件发生时，调用这两个函数

```
<div>点击</div>
<script>
    //如果希望两个事件都能发挥作用怎么办?
    var div=document.getElementsByTagName("div");
    div[0].onclick=function () {
        eve1();
        eve2();
    };
    function eve1() {
        console.log("第一个事件发生作用?")
    };
    function eve2() {
        console.log("第二个事件发生作用?")
    }
</script>
```

例14-5

事件绑定

- 如果希望多个事件都能发挥作用还可以怎么办？

  运用新的事件绑定方法——事件监听器

  语法

  通过id或标签名获取标签

  var 变量名 = document.getElementsByTagName（‘标签名’）

  var 变量名 = document.getElementById（‘标签id名’）

  变量名.addEventListener（“事件方法”，函数名，true/false）;

例14-6

事件绑定

- 事件绑定语法详解

变量名.addEventListener(“事件方法”，函数名，true/false);

divs[0].addEventListener(“click” , evt1 , false);

onclick→click，注意要使用“”将事件名包起来

evt1：函数名称或者也可以内联地声明一个函数

第三个参数：确定以何种事件流触发事件

例14-6

事件绑定

- 事件绑定示例

```
<div id="clk">点击</div>
<script>
    var clk=document.getElementById("clk");
    clk.addEventListener("click",eve1,false);
    clk.addEventListener("click",eve2,false);
    function eve1() {
        console.log("第一个事件发生作用");
    }
    function eve2() {
        console.log("第二个事件发生作用");
    }
</script>
```

例14-7

事件绑定

- 事件绑定示例—内联式

```
<div id="clk">点击</div>
<script>
    var clk=document.getElementById("clk");
    clk.addEventListener("click",function () {
        console.log("第一个事件发生作用");
    },false);
    clk.addEventListener("click",function () {
        console.log("第二个事件发生作用");
    },false);
</script>
```

例14-8

事件绑定

IE?

事件绑定

- addEventListener适用于所有支持DOM Level 2的浏览器
- IE自己的事件模型

变量名.attachEvent（“事件方法”，函数名);

divs[0].attachEvent（“onclick”，evt1);

onclick，注意要使用“”将事件名包起来

evt1：函数名称或者也可以内联地声明一个函数

没有第三个参数

```
<div id="clk">点击</div>
<script>
    var clk=document.getElementById("clk");
    clk.attachEvent("onclick",eve1);
    function eve1() {
        console.log("我是IE自己的事件模型1");
    }
    clk.attachEvent("onclick",function () {
        console.log("我是IE自己的事件模型2");
    })
</script>
```

例14-9

事件绑定

- DOM2中提供绑定事件和移除事件的方法

  变量名.removeEventListener(“事件方法”，函数名，true/false);

  通过addEventListener添加的事件处理程序只能够使用removeEventListener来删除；移除时传入的参数与添加处理程序时使用的参数相同。

```
var clk=document.getElementById("clk");
clk.addEventListener("click",eve1,false);
clk.addEventListener("click",eve2,false);
function eve1() {
    console.log("第一个事件发生作用");
}
function eve2() {
    console.log("第二个事件发生作用");
}
clk.removeEventListener("click",eve2,false);
```

例14-10

事件移除

- DOM2中提供移除事件的方法

  变量名.removeEventListener(“事件方法”，函数名，true/false);

  注意：对于addEventListener()添加的匿名函数无法移除

```
//添加的匿名函数(内联式)无法移除
var clk=document.getElementById("clk");
clk.addEventListener("click",function () {
    console.log("第一个事件发生作用");
},false);
clk.addEventListener("click",function () {
    console.log("第二个事件发生作用");
},false);
clk.removeEventListener("click",function () {
    console.log("第二个事件发生作用");
},false);
```

例14-11

事件移除

- IE提供的移除事件的方法

  obj.detachEvent(事件方法”，函数名)

  IE提供的删除事件处理函数的方法，移除时传入的参数与添加处理程序时使用的参数相同。

```
<div id="clk">点击</div>
<script>
    var clk=document.getElementById("clk");
    clk.attachEvent("onclick",eve1);
    function eve1() {
        console.log("我是IE自己的事件模型");
    }
    clk.detachEvent("onclick",eve1);
</script>
```

例14-12

- 说说addEventListener的第三个参数

addEventListener中的第三个参数是true/false

attachEvent中没有第三个参数

true ——————————→ 捕获

false ——————————→ 冒泡

事件流

- 事件流：页面中接收事件的顺序

捕获	从document的事件开始执行，逐步向内层标签执行，最后执行最内层标签的事件
冒泡	从最内层的标签的事件往外层逐步执行，直到执行document的事件

事件流

- 事件流：页面中接收事件的顺序

冒泡

```
function getTag(obj){
    return document.getElementsByTagName(obj);
}
var body=getTag("body")[0];
var div=getTag("div")[0];
var p=getTag("p")[0];
document.addEventListener("click",function () {
    alert("我是document绑定的事件");
},false);
body.addEventListener("click",function () {
    alert("我是body绑定的事件");
},false);
div.addEventListener("click",function () {
    alert("我是div绑定的事件");
},false);
p.addEventListener("click",function () {
    alert("我是p绑定的事件");
},false);
```

例14-13

事件流

- 事件流：页面中接收事件的顺序

捕获

```
function getTag(obj){
    return document.getElementsByTagName(obj);
}
var body=getTag("body")[0];
var div=getTag("div")[0];
var p=getTag("p")[0];
document.addEventListener("click",function () {
    alert("我是document绑定的事件");
},true);
body.addEventListener("click",function () {
    alert("我是body绑定的事件");
},true);
div.addEventListener("click",function () {
    alert("我是div绑定的事件");
},true);
p.addEventListener("click",function () {
    alert("我是p绑定的事件");
},true);
```

例14-14

事件监听兼容

- 跨浏览器的事件处理程序

怎样创建适用于多个浏览器的事件处理代码?

```
function addEventHandler(obj,eventName,handler) {
    if(document.attachEvent){
        return obj.attachEvent("on"+eventName,handler);
    }else{
        return obj.addEventListener(eventName,handler,false);
    }
}
```

如何调用?

addEventHandler(要绑定事件的对象，‘click’,函数名);

```
var clk=document.getElementById("clk");
addEventHandler(clk,"click",eve1);
function eve1() {
    console.log("我是事件监听");
}
```

例14-15

事件对象

✦ event对象

- 在这里，evt参数表示的是事件对象

```
<div id="clk">点击</div>
<input type="text" id="txt"/>
<script>
    var clk=document.getElementById("clk");
    var txt=document.getElementById("txt");
    clk.onclick=function (evt) {
        console.log(evt);
    };
    document.onkeydown=function (evt) {
        console.log(evt);
    };
    txt.onfocus=function (evt) {
        console.log(evt);
    }
</script>
```

添加参数evt，保证事件对象能在对应的事件处理函数内部可以访问到

✦ event对象

- window对象的一个属性：window.event

```
<div id="clk">点击</div>
<input type="text" id="txt"/>
<script>
    var clk=document.getElementById("clk");
    var txt=document.getElementById("txt");
    clk.onclick=function () {
        console.log(window.event);
    };
    txt.onclick=function (evt) {
        console.log(evt);
    };
</script>
```

✦ event对象

- 获取事件对象兼容写法

```
document.onclick=function (evt) {
    var event=evt||window.event;
    console.log(event);
};
```

例14-18

✦ event对象

- 取得事件目标对象：给哪个元素节点绑定事件，目标对象就是谁。

```
<div id="clk">点击</div>
<input type="text" id="txt"/>
<script>
    var clk=document.getElementById("clk");
    var txt=document.getElementById("txt");
clk.onclick=function (evt) {
    var event=evt||window.event;
    console.log(event);
    //获取事件目标对象
    //event.target是W3C标准(非IE)
    //event.srcElement是IE支持的
    console.log(event.target||event.srcElement);
};
txt.onclick=function (evt) {
    var event=evt||window.event;
    console.log(event);
    console.log(event.target||event.srcElement);
};
```

例14-19

✦ event对象

- 取得事件目标对象 有何用途?

```
<div id="clk">
    我是div
    <h3>我是h3
        <span>我是span</span>
    </h3>
</div>
<script>
    var clk=document.getElementById("clk");
    var txt=document.getElementById("txt");
    clk.onclick=function (evt) {
        //获取事件对象
        var event=evt||window.event;
        //获取事件目标对象
        //event.target是W3C标准(非IE)
        //event.srcElement是IE支持的
        var target=event.target||event.srcElement;
        target.style.background="red";
        console.log(target);
    };
</script>
```

经常需要获取触发某个事件的目标对象。
然后根据目标对象进行不同的业务处理。

- 阻止事件发生时浏览器的默认行为

非IE浏览器

```
event.preventDefault();
```

IE浏览器

```
window.event.returnValue = false;
```

- 阻止浏览器的默认行为兼容写法

```
var a=document.getElementsByTagName("a")[0];
var form=document.getElementsByTagName("form")[0];

//封装一个阻止浏览器默认行为的兼容性方法
function prevent(evt){
    var event=evt||window.event;
    if(event&&event.preventDefault){
        event.preventDefault();
    }else{
        event.returnValue=false;
    }
}

a.onclick=function (evt) {
    prevent(evt);
};
form.onclick=function (evt) {
    prevent(evt);
};
```

例14-22

阻止浏览器默认行为

使用addEventListener进行事件绑定

假设要进行的操作是阻止默认事件 应该如何书写?

```
var a=document.getElementsByTagName("a")[0];
var form=document.getElementsByTagName("form")[0];
//封装一个阻止浏览器默认行为的兼容性方法
function prevent(evt){
    var event=evt||window.event;
    if(event&&event.preventDefault){
        event.preventDefault();
    }else{
        event.returnValue=false;
    }
}
a.addEventListener("click",eve1,false);
function eve1(evt) {
    prevent(evt);
}
```

例14-22

- 阻止冒泡

阻止冒泡的发生

非IE浏览器

```
event.stopPropagation();
```

IE浏览器

```
window.event.cancelBubble = true;
```

- 阻止冒泡语句兼容写法

```
div.addEventListener("click",function (evt) {
    stopBubble(evt);
    alert("我是div绑定的事件");
},false);
p.addEventListener("click",function (evt) {
    stopBubble(evt);
    alert("我是p绑定的事件");
},false);
//封装一个阻止冒泡的兼容性方法
function stopBubble(evt){
    var event=evt||window.event;
    if(evt&&event.stopPropagation){
        event.stopPropagation();
    }else{
        event.cancelBubble=true;
    }
}
```

1、把ppt里的例子全部理解并熟练掌握

2、阻止浏览器右键弹出菜单

3、城市切换效果

Event对象参考网页

http://www.w3school.com.cn/jsref/dom_obj_event.asp

总结

- 总结IE的事件机制方面与其他浏览器有何不同?
- 如何为一个对象绑定多个事件?
- 获取事件对象目标的方法
- 阻止冒泡的语句是什么?
- 阻止浏览器默认行为的语句是什么?

- 总结事件处理程序的知识点
- 书写能够兼容各大浏览器的事件的类

谢

凌阳教育
www.sunplusedu.com

值得信赖的教育品牌

Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com