第3章

# JavaScript事件处理


视频讲解

用户可以通过多种方式与浏览器中的页面进行交互，而事件是交互的桥梁。Web 应用程序开发人员通过 JavaScript 脚本内置的和自定义的事件处理器来响应用户的动作，就可以开发出更具交互性、动态性的页面。

本章主要介绍 JavaScript 脚本中的事件处理的概念、方法，列出了 JavaScript 预定义的事件处理器，并且介绍了如何编写用户自定义的事件处理函数以及如何将它们与页面中用户的动作相关联，以得到预期的交互性能。

## 3.1 JavaScript 事件的基本概念

事件是指 JavaScript 捕获到用户的操作，并做出正确的响应。例如，用户单击鼠标弹出一个窗口，把鼠标移动到某个元素上产生变化。事件处理是 JavaScript 的一个优势，可以很方便地针对某个 HTML 事件编写程序进行处理。

### 3.1.1 事件类型

JavaScript 支持丰富的事件类型，能使 Web 开发更加快速和简洁。JavaScript 所支持的事件可以分为以下 5 类。

**1. 窗口事件(Window Events)**

仅在 body 和 frameset 元素中有效，窗口事件如表 3-1 所示。

**表 3-1 窗口事件**

| 事　　件 | 说　　明 |
| --- | --- |
| onload | 当网页被载入时执行脚本 |
| onunload | 当网页被关闭时执行脚本 |

**2. 表单元素事件(Form Element Events)**

仅在表单元素中有效，表单元素事件如表 3-2 所示。

表 3-2　表单元素事件

| 事　　件 | 说　　明 |
| --- | --- |
| onchange | 当元素(文本框、复选框等)改变时执行脚本 |
| onsubmit | 当表单(form)被提交时执行脚本 |
| onreset | 当表单被重置时执行脚本 |
| onselect | 当元素被选取时执行脚本 |
| onblur | 当元素失去焦点时执行脚本 |
| onfocus | 当元素获得焦点时执行脚本 |

3. 图像事件(Image Events)

该属性可用于 img 元素,图像事件如表 3-3 所示。

表 3-3　图像事件

| 事　　件 | 说　　明 |
| --- | --- |
| onabort | 当图像加载中断时执行脚本 |

4. 键盘事件(Keyboard Events)

在下列元素中无效：base、bdo、br、frame、frameset、head、html、iframe、meta、param、script、style 以及 title 元素,键盘事件如表 3-4 所示。

表 3-4　键盘事件

| 事　　件 | 说　　明 |
| --- | --- |
| onkeydown | 当键盘被按下时执行脚本 |
| onkeypress | 当键盘被按下后又松开时执行脚本 |
| onkeyup | 当键盘被松开时执行脚本 |

5. 鼠标事件(Mouse Events)

在下列元素中无效：base、bdo、br、frame、frameset、head、html、iframe、meta、param、script、style 以及 title 元素,鼠标事件如表 3-5 所示。

表 3-5　鼠标事件

| 事　　件 | 说　　明 |
| --- | --- |
| onclick | 当鼠标被单击时执行脚本 |
| ondblclick | 当鼠标被双击时执行脚本 |
| onmousedown | 当鼠标按钮被按下时执行脚本 |
| onmousemove | 当鼠标指针移动时执行脚本 |
| onmouseout | 当鼠标指针移出某元素时执行脚本 |
| onmouseover | 当鼠标指针悬停于某元素之上时执行脚本 |
| onmouseup | 当鼠标按钮被松开时执行脚本 |

## 3.1.2　JavaScript 处理事件的基本机制

JavaScript 处理事件的基本机制如下。

(1) 对 DOM 元素绑定事件处理函数。

(2) 监听用户的操作。

(3) 当用户在相应的 DOM 元素上进行与绑定事件对应的操作时,浏览器调用事件处理函数作出响应。

(4) 将处理结果更新到 HTML 文档。

JavaScript 处理事件的过程如图 3-1 所示。

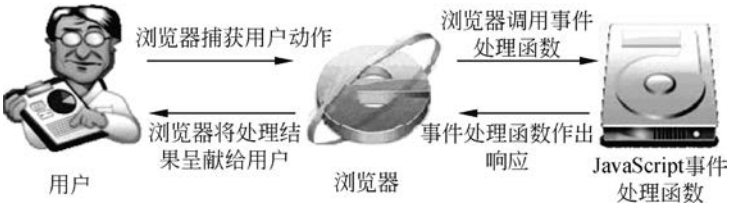


**图 3-1 JavaScript 处理事件的过程**

## 3.2 JavaScript 绑定事件的方法

JavaScript 事件一般与 DOM 元素绑定。要想让 JavaScript 对用户的操作作出响应,首先要对 DOM 元素绑定事件处理函数。所谓事件处理函数,就是处理用户操作的函数,不同的操作对应不同的名称。

在 JavaScript 中,有以下三种常用的绑定事件的方法。

### 1. 在 DOM 元素中直接绑定

这里的 DOM 元素可以理解为 HTML 标记。JavaScript 支持在标记中直接绑定事件,语法如下:

```
onXXX = "JavaScript Code"
```

其中,onXXX 为事件名称。例如,鼠标单击事件 onclick、鼠标双击事件 ondouble、鼠标移入事件 onmouseover、鼠标移出事件 onmouseout 等。

JavaScript Code 为处理事件的 JavaScript 代码,一般是函数。例如,单击一个按钮,弹出警告框的代码有如下两种写法。

(1) 原生函数。

```
< input onclick = "alert('谢谢支持')" type = "button" value = "单击我,弹出警告框" />
```

(2) 自定义函数。

```
< input onclick = "myAlert()" type = "button" value = "单击我,弹出警告框" />
< script type = "text/javascript">
function myAlert(){
    alert("谢谢支持");
}
</script >
```

### 2. 在 JavaScript 代码中绑定

在 JavaScript 代码中(即< script >标记内)绑定事件可以使 JavaScript 代码与 HTML 标记分离,文档结构清晰,便于管理和开发。在 JavaScript 代码中绑定事件的语法如下:

```
elementObject.onXXX = function(){
    //事件处理代码
}
```

其中,elementObject 为 DOM 对象(即 DOM 元素),onXXX 为事件名称。

例如,为 id="demo"的按钮绑定一个事件,显示它的 type 属性。

```
<input id="demo" type="button" value="单击我,显示 type 属性" />
<script type="text/javascript">
document.getElementById("demo").onclick = function(){
    alert(this.getAttribute("type")); //this 指当前发生事件的 HTML 元素,这里是 <input>
                                      //标记
}
</script>
```

### 3. 绑定事件监听函数

绑定事件的另一种方法是用 addEventListener()函数或 attachEvent()函数来绑定事件监听函数,addEventListener()函数的语法如下:

```
elementObject.addEventListener(eventName,handle,useCapture);
```

addEventListener()函数的参数如表 3-6 所示。

**表 3-6 addEventListener()函数的参数**

| 参 数 | 说 明 |
|---|---|
| elementObject | DOM 对象(即 DOM 元素) |
| eventName | 事件名称。注意,这里的事件名称没有"on",如鼠标单击事件 click、鼠标双击事件 doubleclick、鼠标移入事件 mouseover、鼠标移出事件 mouseout 等 |
| handle | 事件句柄函数,即用来处理事件的函数 |
| useCapture | Boolean 类型,是否使用捕获,一般用 False。这涉及 JavaScript 事件流的概念 False 表示在冒泡阶段完成事件处理;将其设置为 True 时,表示在捕获阶段完成事件处理 |

例如,使用 addEventListener()函数为 id="demo"的按钮绑定一个单击事件,显示 Hello 弹窗。

```
<input id="demo" type="button" value="单击我,显示 Hello" />
<script type="text/javascript">
    var button1 = document.getElementById("demo");
    button1.addEventListener("click",myAlert);
    function myAlert(){
        alert("Hello");
    }
</script>
```


事件句柄函数是指“函数名”,不能带小括号。另外早期版本的 IE 浏览器(IE 8.0 及其以下版本)使用 attachEvent()函数来绑定事件监听函数。

## 3.3 JavaScript 事件的 event 对象

event 对象是 JavaScript 中一个非常重要的对象,用来表示当前事件。event 对象的属性和方法包含了当前事件的状态。event 对象只在事件发生的过程中才有效。

当前事件是指正在发生的事件。状态是与事件有关的性质,如引发事件的 DOM 元素、鼠标的状态、按下的键等。

### 3.3.1 获取 event 对象

在 W3C 规范中，event 对象是随事件处理函数传入的。Edge、Chrome、FireFox、Opera、Safari、IE 9 以上都支持这种方式。在遵循 W3C 规范的浏览器中，event 对象通过事件处理函数的参数传入。

```
elementObject.OnXXX = function(e){
    var eve = e;                    //声明一个变量来接收 event 对象
}
```

上面绑定的事件处理函数中，参数 e 用来传入 event 对象。要想获取与当前事件有关的状态，如发生事件的 DOM 元素、鼠标坐标、键盘按键等，可以通过参数 e 获取。event 对象常用属性和方法如表 3-7 所示。

**表 3-7 event 对象常用属性和方法**

| 属性/方法 | 描 述 |
| --- | --- |
| event. type | 被触发事件的类型。例如，触发 button 的 click 事件，那 event. type 的值就为"click" |
| event. target | 本次事件中的目标元素。因为事件流机制的存在，当单击 button 时，会按照不同的顺序触发其他元素的事件，在这个过程中，被单击的 button 元素就是事件中的目标元素 |
| event. currentTarget | 本次事件中当前正在处理的元素。按照事件冒泡或者捕获的顺序处理到哪个元素的事件，哪个元素就是当前正在处理的元素 |
| event. cancelable | 表示是否可以取消事件的默认行为。True 表示可以取消事件的默认行为 |
| event. preventDefault() | 调用该方法可以取消事件的默认行为，但是前提是 event. cancelable 的值为 True |
| event. bubbles | 表明事件是否冒泡 |
| event. stopPropagation() | 调用该方法可以取消事件的下一步冒泡，但前提是 event. bubbles 的值为 True |

例如，要取得发生事件时鼠标的坐标，可以这样写：

```
<div id = "demo">在这里单击</div>
<script type = "text/javascript">
document.getElementById("demo").onclick = function(e){
    var eve = e;
    var x = eve.x;                          //X 坐标
    var y = eve.y;                          //Y 坐标
    alert("X 坐标:" + x + "\nY 坐标:" + y);
}
</script>
```

例如，要取得发生键盘事件时按键信息，可以这样写：

```
document.onkeydown = function(e){
    alert("按键是:" + e.keyCode);     //按任意键，得到相应的 keyCode(ASCII 中的编码)
};
```

### 3.3.2 JavaScript 获取鼠标坐标

鼠标坐标包括 X 坐标、Y 坐标、相对于浏览器客户端的坐标、相对于屏幕的坐标等。在 JavaScript 中，鼠标坐标是作为 event 对象的属性存在的。event 对象中有关鼠标坐标的属性如表 3-8 和表 3-9 所示。

**表 3-8 W3C 规范所规定的属性**

| 属 性 | 描 述 |
|---|---|
| clientX | 鼠标指针相对客户端(即浏览器文档区域)的水平坐标 |
| clientY | 鼠标指针相对客户端(即浏览器文档区域)的垂直坐标 |
| screenX | 鼠标指针相对计算机屏幕的水平坐标 |
| screenY | 鼠标指针相对计算机屏幕的垂直坐标 |

**表 3-9 IE 浏览器的特有属性**

| 属 性 | 描 述 |
|---|---|
| offsetX | 发生事件的地点在事件源元素的坐标系统中的水平坐标 |
| offsetY | 发生事件的地点在事件源元素的坐标系统中的垂直坐标 |
| x | 事件发生的位置的水平坐标，它们相对于用 CSS 动态定位的最内层包容元素 |
| y | 事件发生的位置的垂直坐标，它们相对于用 CSS 动态定位的最内层包容元素 |

**【例 3-1】** 获取鼠标的坐标信息。

```
<html>
<head>
<title>获取鼠标的坐标信息</title>
</head>
<body>
<div id="demo">单击这里</div>
<script type="text/javascript">
document.getElementById("demo").onclick=function(e){
    var eve=e;
    var x=eve.clientX,                    //相对于客户端的 X 坐标
        y=eve.clientY,                    //相对于客户端的 Y 坐标
        x1=eve.screenX,                   //相对于计算机屏幕的 X 坐标
        y1=eve.screenY;                   //相对于计算机屏幕的 Y 坐标
    alert(
        "相对客户端的坐标:\n"+
        "x="+x+"\n"+
        "y="+y+"\n\n"+
        "相对屏幕的坐标:\n"+
        "x="+x1+"\n"+
        "y="+y1
    );
}
</script>
</body>
</html>
```

### 3.3.3 JavaScript 获取事件源

事件源是指发生事件的 DOM 节点(HTML 元素)。

事件源是作为 event 对象的属性存在的。在 W3C 规范中,这个属性是 target。

【例 3-2】 获取事件源。

```
<html>
<head>
<title>获取事件源</title>
</head>
<body>
<div id="demo">单击这里</div>
<script type="text/javascript">
    document.getElementById("demo").onclick = function(e){
        var srcNode = e.target;
        alert(srcNode);
    }
</script>
</body>
</html>
```

## 3.4 JavaScript 取消浏览器默认动作

默认动作是指浏览器所执行的用户没有明确指定的操作。对于某些 HTML 标记,浏览器总会有一个默认的动作。浏览器的默认动作是可以通过 JavaScript 来取消的。

例如,<a href="http://www.baidu.com" target="_blank">单击这里进入百度</a>,单击上面的链接,浏览器会弹出窗口,进入百度首页。这个动作就是浏览器的默认动作,单击一个<a>标记会转向目标页面。

其他浏览器默认动作包括:单击提交按钮提交表单、单击重置按钮重置表单、把鼠标移动到带有 title 属性的元素上出现提示等。

对于遵循 W3C 规范的浏览器,使用 event 对象的 preventDefault()方法来取消默认动作。

【例 3-3】 取消<a>标记的默认动作。

```
<html>
<head>
<title>取消<a>标记的默认动作</title>
</head>
<body>
<a id="demo" href="http://www.baidu.com" target="_blank">单击这里试试</a>
<script type="text/javascript">
    document.getElementById("demo").onclick = function(e){
        var eve = e;
        eve.preventDefault();
    }
</script>
</body>
</html>
```

浏览页面,单击<a>标记给的链接地址本来会跳转到百度首页,但是用 JavaScript 取消了跳转,所以单击后没效果。