第3章

JavaScript事件处理

本章学习目标：

通过本章内容的学习，读者将能够了解 HTML 事件类型，理解 JavaScript 事件在页面中的作用，理解事件、事件句柄和事件处理函数等概念；通过编写事件处理函数来响应相关事件，完成特定的功能需求。

Web 前端开发工程师应知应会以下内容。

- 了解 JavaScript 事件类型。
- 理解事件、事件句柄与事件处理函数的概念与关联方式。
- 掌握常用的 HTML 事件。
- 掌握常用的鼠标单击、双击及移动等事件。
- 掌握常用的键盘事件及窗口事件。

3.1 JavaScript 事件

采用事件驱动是 JavaScript 语言的一个最基本的特征。事件是指一些可以通过脚本响应的页面动作。当用户单击鼠标或者提交一个表单，甚至在页面上移动鼠标时，就会产生相关的事件。绝大多数事件的命名是描述性的，很容易理解，如 Click、Submit、MouseOver 等，通过名称就可以猜测其含义。

3.1.1 事件类型

JavaScript 事件大多数与 HTML 标记相关，都是由用户操作页面元素时触发的。根据事件触发的来源及作用对象的不同，可以把事件分为鼠标事件、键盘事件、HTML 事件、文档事件及手势事件(移动端)等类型。

1. 鼠标事件

鼠标事件主要指使用鼠标操作 HTML 元素时触发的事件。常用的鼠标事件有鼠标单击、鼠标双击、选择文本框、选中单选按钮、选中复选框等。例如，当鼠标移动、悬停、移出网页上相关区域内的特定元素时将触发 MouseMove、MouseOver 和 MoveOut 事件。

2. 键盘事件

键盘事件主要指用户在键盘上按键时触发的事件。例如，用户在键盘上按下某一个键时会触发 KeyDown 事件，用户释放按下的键时会触发 KeyUp 事件。

3. HTML 事件

HTML 事件主要指当窗口发生变动或者发生特定的客户端/服务器端交互时触发的事件。例如，页面完全载入时在 Window 对象上会触发 Load 事件；任何元素或者窗口本身失去焦点时会触发 Blur 事件。

4. 文档事件

文档事件主要指文档对象底层元素发生改变时触发的事件。例如，在元素中添加子元素时触发、元素中任意子元素被删除时触发、元素结构中发生任何变化时触发、元素从文档中移除之前触发、元素从文档中被添加之前触发等事件。

5. 手势事件

在移动端设备上发生手势触发事件。例如，发生手指触摸时触发、手指在元素上滑动时触发、手指从屏幕上移开时触发、当系统停止跟踪触摸时触发等事件。

3.1.2 事件句柄

事件句柄（又称事件处理函数）是指事件发生时要进行的操作。每一个事件均对应一个事件句柄，在程序执行时，将相应的函数或语句指定给事件句柄，则在该事件发生时，浏览器便执行指定的函数或语句，从而实现网页内容与用户操作的交互。当浏览器检测到某事件发生时，便查找该事件对应的事件句柄有没有被赋值，如果有，则执行该事件句柄。通常，事件句柄的命名原则是在事件名称前加上前缀 on。例如，鼠标移动 MouseOver 事件，其事件句柄为 onMouseOver。语法和示例如下。

```
<标记  事件句柄 = "JavaScript 代码">… </标记>
< input type = "button" name = "" value = "显示" onclick = "show();">
```

事件句柄名称与事件属性名称相同，都作为 HTML 标记的属性，与事件名称不同，只是在事件名称前面加上了“on”。例如，Click 事件的事件句柄为 onClick，该项标记对应的事件属性也为 onClick；Blur 事件的事件句柄为 onBlur，该项标记对应的事件属性也为 onBlur；其他事件的事件句柄以此类推。常用的事件类型、事件句柄及说明如表 3-1 所示。

表 3-1 事件类型、事件句柄及说明

事件类型	事件句柄	值	说　明
键盘事件	onKeyDown	script	当键被按下时运行脚本
	onKeyPress	script	当键被按下后又松开时运行脚本
	onKeyUp	script	当键被松开时运行脚本
鼠标事件	onClick	script	当鼠标单击时运行脚本
	onDblclick	script	当鼠标双击时运行脚本
	onMouseDown	script	当鼠标按键按下时运行脚本
	onMouseMove	script	当鼠标指针移动时运行脚本
	onMouseOut	script	当鼠标指针移出某元素时运行脚本
	onMouseOver	script	当鼠标指针悬停于某元素之上时运行脚本
	onMouseUp	script	当鼠标按键松开时运行脚本
HTML 事件	onChange	script	当元素改变时运行脚本
	onSubmit	script	当表单被提交时运行脚本
	onReset	script	当表单被重置时运行脚本
	onSelect	script	当元素被选取时运行脚本
	onBlur	script	当元素失去焦点时运行脚本
	onFocus	script	当元素获得焦点时运行脚本
	onLoad	script	当文档载入时运行脚本
	onUnload	script	当文档卸载时运行脚本

3.1.3 事件处理

只要给指定的事件句柄（事件属性）绑定事件处理代码就可以响应事件。事件处理指定方

式一般有两种：在 HTML 标记中静态指定、在 JavaScript 中动态指定。

1. 静态指定

```
<标记 事件句柄 1 = "事件处理程序 1" [事件句柄 2 = "事件处理程序 2"  …  事件句柄 n = "事件处理程序 n"]>…</标记>
```

静态指定是在开始标记中设置相关事件句柄，并绑定事件处理程序即可。一个标记可以设置一个或多个事件句柄，并绑定事件处理程序。事件处理程序可以是 JavaScript 代码串或函数，通常将事件处理程序定义成函数。

例如，给< a >标记和< body >标记添加事件句柄，并绑定事件处理函数：

```
< a onClick = "show();">事件处理</a >
< body onLoad = "alert('页面装载成功!');" onbeforeunload = "pageLoad();"></body >
```

【例 3-1】 事件处理方式实战——静态指定。

JS 代码如下，页面效果如图 3-1 所示。

（1）JS 代码 js-3-1.html 如下。

```
<!-- js-3-1.html -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>静态指定事件处理函数</title>
    <script src="js-3-1.js"></script>
  </head>
  <body>
    <h3>静态指定方式应用</h3>
    <input type="button" value="JS 语句"  onclick=
      "document.getElementById('content').innerHTML='使用 document.write()输出信息'"  />
    <input type="button" value="绑定事件处理函数"
      onclick="displayInfo('调用 displayInfo()函数输出信息')"  />
    <p id="content"></p>
  </body>
</html>
```

图 3-1 静态指定方式应用

（2）外部 JS 文件 js-3-1.js 代码如下。

```
// js-3-1.js
 function displayInfo(message) {
   document.getElementById("content").innerHTML = message;  //innerHTML 属性赋值
 }
```

2. 动态指定

通常情况下使用静态指定方式来处理事件，但有时也需要在程序运行过程中动态指定事件，也称为分配某一事件，这种方式允许程序像操作 JavaScript 属性一样来处理事件。语法和示例如下。

```
<事件源对象>.<事件句柄> = function(){<事件处理程序>;}
Object.onclick = function(){handler();}      //动态给对象指派事件，绑定事件处理函数
Object.onclick();                            //调用方法
```

```
Object.onclick = null              //事件销毁
//事件注册(添加事件监听器)
element.addEventListener(event, function,useCapture)
//例如,给 window 对象添加 beforeunload 事件监听器
window.addEventListener('beforeunload',beforeUnloadHandler,true);
function beforeUnloadHandler(event){
   event.returnValue = "要离开吗?"
}
```

动态指定用法中,“事件处理程序”必须使用无名函数 function(){}来定义,函数体内可以是字符串形式的代码,也可以是函数。

addEventListener()方法用于向指定元素添加事件。其参数说明如下。

- event:必需。字符串,指定事件名。注意:不要使用“on”前缀。例如,使用“click”,而不是使用“onclick”。

 **注意** 关于所有 HTML DOM 事件,可以通过查看完整的 HTML DOM Event 对象参考手册进行了解。

- function:必需。指定事件触发时执行的函数。当事件触发时,事件对象会作为第一个参数传入函数。事件对象的类型取决于特定的事件,例如,“click”事件属于 MouseEvent(鼠标事件)对象。
- useCapture:可选。布尔值,指定事件是否在捕获或冒泡阶段执行。其值为 true,表示事件句柄在捕获阶段执行;其值为 false(默认值),表示事件句柄在冒泡阶段执行。

【例 3-2】 指定事件方式实战——动态指定。

JS 代码如下,页面效果如图 3-2 所示。

(1) JS 代码 js-3-2.html 如下。

```
<!-- js-3-2.html -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>动态指定实战</title>
    <style type="text/css">
      input {width: 100px; height: 40px;border-radius: 20px;border: 1px dashed black; }
    </style>
  </head>
  <body>
    <h3>动态指定实战</h3>
    <input type="button" onclick="staticFun()" value="静态指定" />
    <!-- 初始时未静态指定 -->
    <input id="myInput" name="myInput" type="button" value="显示信息" />
    <p id="content"></p>
    <script src="js-3-2.js"></script>
    <script type="text/JavaScript">
      //向 button 元素动态分配 onclick 事件句柄
      document.getElementById('myInput').onclick = function(){dynmicFun()};
      myInput.onclick(); //程序触发
      //事件注册,使用此功能,需要屏蔽第 20~21 行代码
      //document.getElementById('myInput').addEventListener('click', dynmicFun)
    </script>
  </body>
</html>
```

(2) 外部 JS 文件 js-3-2.js 代码如下。

```
//js-3-2.js
var count = 0;      //定义全局变量
```

```
function staticFun() {
  document.getElementById("content").innerHTML =
    "<h3 style='color:red;'>这是静态指定方式</h3>";
}
function dynmicFun() {
  count++;          //累加
  if (count >= 2) {
    //从第 2 次开始,以后是单击事件触发
    document.getElementById("content").innerHTML =
      "<h3 style='color:green;'>这是单击事件触发!</h3>";
  } else {
    //第 1 次是程序触发
    document.getElementById("content").innerHTML =
      "<h3>这是动态指定,程序触发!</h3>";
  }
}
```

图 3-2　动态指定与静态指定混合应用

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和根节点之间按照特定的顺序传播，路径所经过的所有节点都会收到该事件，这个传播过程即 DOM 事件流。

DOM 事件流包含三个阶段，分别是事件捕获阶段、处理目标阶段、事件冒泡阶段。首先发生的事件捕获为截获事件提供机会，然后是实际的目标接收事件，最后一个阶段是事件冒泡阶段，可以在这个阶段对事件做出响应。

在 DOM 事件流中，事件的目标在捕获阶段不会接收到事件，这意味着在捕获阶段事件从 document 到 就停止了，下个阶段是处理目标阶段，于是事件在 上发生，并在事件处理中被看成冒泡阶段的一部分。然后，冒泡阶段发生，事件又传播回 document，如图 3-3 所示。

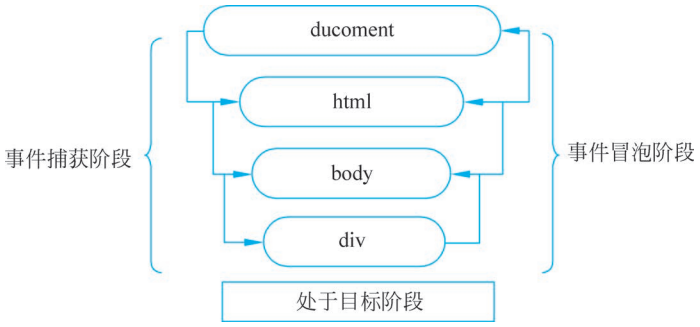

图 3-3　事件流执行流程图

【例 3-3】　DOM 事件流(事件捕获与事件冒泡)实战。

JS 代码如下，页面如图 3-4 所示。

```
<!-- js-3-3.html -->
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <style>
```

```
 7.        #container {margin: 0 auto;   text-align: center;}
 8.        #parentDiv {margin: 0 auto;   width: 500px; height: 200px;
 9.          background-color: rgb(210, 218, 225);   text-align: center;
10.        }
11.        #subDiv { margin: 50px auto; width: 300px;
12.          height: 100px;   background-color: #99ddaa;
13.        }
14.        #information {   height: 100px; margin: 0 auto; }
15.      </style>
16.    </head>
17.    <body>
18.      <div id="container">
19.        <h3>DOM 事件流</h3>
20.        <div id="parentDiv">
21.          <div id="subDiv">子 Div</div>
22.        </div>
23.        <div id="information"></div>
24.      </div>
25.      <script>
26.        function $(id) { return document.getElementById(id);}   //通过 id 获取页面元素
27.        var parentDiv = $("parentDiv");                          //获取父 div
28.        var subDiv = $("subDiv");                                //获取子 div
29.        //以下是事件冒泡,从内层元素向外层元素依次执行单击事件
30.        parentDiv.addEventListener( "click", function () {
31.          $("information").innerHTML += "<br>parentDiv-冒泡单击事件!";
32.        }, false);
33.        subDiv.addEventListener("click", function () {
34.          $("information").innerHTML += "<br>subDiv-冒泡单击事件!";
35.        },  false );
36.      </script>
37.    </body>
38.  </html>
```

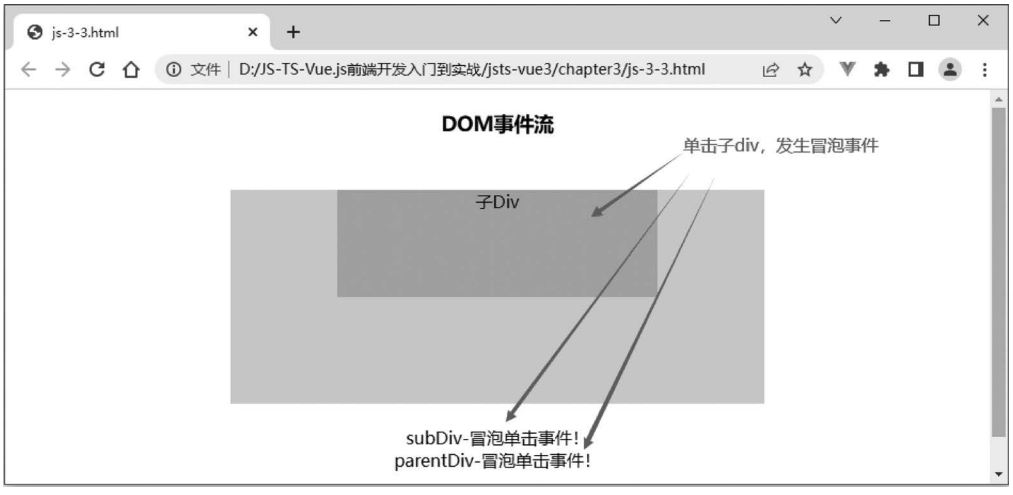

图 3-4 DOM 事件流——冒泡事件

只需要将代码第 32、35 行中的 false 改为 true，就可以改为捕获事件，同时将第 31、34 行中的“冒泡”改为“捕获”即可。

3.1.4 事件处理程序的返回值

在 JavaScript 中如果事件处理程序不需要有返回值，此时浏览器会按默认方式进行处理。但有些情况下需要使用返回值来判断事件处理程序是否正确进行处理，或者通过这个返回值来判断是否进行了下一步操作。事件处理程序的返回值都为布尔型值，如果为 false，则阻止浏览器的下一步操作；如果为 true，则进行默认的操作。语法如下。

```
<标记 事件句柄 = "return 函数名(参数);" ></标记>
```

事件处理代码中的函数必须具有布尔型的返回值，即函数体中最后一句必须是带返回值的 return 语句。

【例 3-4】 事件处理程序返回值的应用。

JS 代码如下，页面如图 3-5 和图 3-6 所示。

(1) JS 代码 js-3-4.html 如下。

```
<!-- js-3-4.html -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>事件处理程序返回值的应用</title>
    <script type="text/JavaScript">
     function $(id){return document.getElementById(id);}
     function showName() {
      if ($("myName").value == "") {
        $("content").innerHTML = "没有输入内容!";
        return false;
      } else {
        $("content").innerHTML = "欢迎你!" + $("myName").value;
        return true;
      }
     }
    </script>
  </head>
  <body>
    <h3>事件处理程序返回值的应用</h3>
    <!-- onsubmit 事件处理函数返回真值就执行 action 指定的网页 -->
    <form name="myForm" action="js-3-4-1.html" onsubmit="return showName();">
     姓名:<input type="text" name="myName" id="myName" />
      <input type="submit" value="提交" />
    </form>
    <p id="content"></p>
  </body>
</html>
```

(2) JS 代码 js-3-4-1.html 如下。

```
<!-- js-3-4-1.html -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>这是跳转的网页</title>
  </head>
  <body>
    <h3>这是跳转网页。</h3>
    <script>
     var ss = document.location.search.split("="); //获取 URL 中搜索内容,并用"="分隔
     //对 URI 进行解码 decodeURI()
     document.write(ss[0].substring(1) + "=" + decodeURI(ss[1]));
    </script>
  </body>
</html>
```

在"姓名"文本框中输入姓名，单击"提交"按钮，触发 Submit 事件，调用执行代码"return showName();"，返回值为 true 则浏览器进行下一操作，访问网页 js-3-4-1.html，如图 3-6 所示，同时在页面中显示传递过来的姓名数据。如果在文本输入框中不输入任何内容就单击"提

图 3-5 事件处理程序的返回值应用

图 3-6 返回值为真时的跳转页面

交”按钮则返回 false 值，浏览器阻止进行下一操作，提示“没有输入内容!”。

3.2 HTML 事件

HTML 事件是发生在 HTML 元素上的“事情”。当在 HTML 页面中使用 JavaScript 时，JavaScript 能够“应对”这些事件。

HTML 事件可以是浏览器或用户做的某些事情。例如，HTML 网页完成加载、HTML 输入字段被修改、HTML 按钮被单击等。通常，当事件发生时，用户会希望做某件事。

JavaScript 允许在事件被侦测到时执行代码。通过 JavaScript 代码，HTML 允许用户向 HTML 元素添加事件处理程序。常见的 HTML 事件属性有 onChange、onClick、onMouseOver、onMouseOut、onKeyDown、onLoad 等。

3.2.1 onChange 与 onSelect 事件属性

在 HTML 表单中，当选择了单行文本输入框或多行文本输入框内的文字时会触发 Select 选择事件。部分示例代码如下。

```
<input type="text" name="" value="文本被选择后触发事件" onSelect="JavaScript:alert('内容已被选中!')">
<textarea rows="5" cols="40" onSelect="alert('内容已被选中!') "></textarea>
<select id="mySel" onChange="changeOptions()">
    <option disabled>-- 请选择 --</option>
    <option value="1">Web 前端开发技术</option>
</select>
```

代码中第 1 行定义了一个单行文本输入框，并设置 onSelect 属性值为 JavaScript 代码；当文本框的内容被选中后，将触发 Select 事件，调用代码，通过告警消息框弹出一个显示“内容已被选中!”的对话框；当一个单行文本输入框或多行文本输入框失去焦点并更改值时或当 select 下拉选项中的一个选项状态改变后会触发 Change 事件。

【例 3-5】 onChange 与 onSelect 事件属性实战。

JS 代码如下，页面如图 3-7 和图 3-8 所示。该例的难点在于如何获取在单行文本框和多行文本域中选择的文字内容。可以借助于对象的 selectionStart、selectionEnd 属性来获取所选区域的文字内容的起始和终止位置。

```
1.  <!-- js-3-5.html -->
2.  <!DOCTYPE html>
3.  <html lang="en">
4.    <head>
```

```
    <meta charset = "UTF - 8" />
    <title>下拉菜单</title>
    <style>
     fieldset { margin: 0 auto; width: 500px; height: 200px; padding: 20px; }
    </style>
    <script language = "JavaScript">
     function $(id) { return document.getElementById(id); } //获取元素
     function changeOptions() {
      var index = $("mySel").selectedIndex;                    //获取下拉框中选项
      $("content").innerHTML =  "选择的课程: "
        + $("mySel").options[index].text;                      //获取选项文本
     }
     function selectText(id) {
      $("content").innerHTML =  "选择的文本: " +
        $(id).value.substring($(id).selectionStart, $(id).selectionEnd);
     }
    </script>
  </head>
  <body>
    <div align = "center">
      <form>
        <fieldset>
          <legend>HTML 事件应用</legend>
          姓名: <input type = "text"  name = ""  id = "myName"
            onselect = "selectText('myName')"  placeholder = "请输入姓名"  /><br />
          意见: <textarea  id = "myTxt"  cols = "30" rows = "4"
            placeholder = "请输入意见" onselect = "selectText('myTxt')" ></textarea>
          <br />选择课程: <select id = "mySel" onChange = "changeOptions()">
            <option disabled>-- 请选择 --</option>
            <option value = "1">Web 前端开发技术</option>
            <option value = "2">Vue.js 前端框架技术</option>
            <option value = "3">JSP 程序设计</option>
            <option value = "4">JavaEE 编程</option>
          </select>
          <p align = "center" id = "content"></p>
        </fieldset>
      </form>
    </div>
  </body>
</html>
```

图 3-7　HTML 选择与改变事件初始页面

图 3-8　选择部分文本与选择列表框中选项执行页面

代码第 19 行中的“selectionStart”表示选择所选区域的起始位置，“selectionEnd”表示选择所选区域的终止位置。然后通过 substring()函数来获取所选中的文本。

3.2.2 onSubmit 与 onReset 事件属性

在 HTML 表单中单击“提交”按钮后，会触发 Submit 事件，将表单中的数据提交到服务器端；当单击“重置”按钮后，会触发 Reset 事件，将表单中的数据重置为初始值。在表单中，插入一个 type 属性值为 submit 的< input >标记来添加一个“提交”按钮，当单击该按钮时会触发表单的 Submit 事件；同样可以插入一个 type 属性值为 reset 的< input >标记来添加一个“重置”按钮，当单击该按钮时会触发表单的 Reset 事件。语法如下。

```
<form onSubmit = "return submitTest();"  onReset = "resetTest()">
    <input type = "submit" name = "" value = "提交事件" >
    <input type = "reset" name = "" value = "重置事件" >
</form >
```

如果需要在表单 Submit 事件及 Reset 事件触发时完成特定的功能，如需要对表单数据进行合法性验证，则需要为表单设置 onReset 和 onSubmit 事件句柄，并自定义相关函数。

【例 3-6】 表单提交、重置事件实战。

JS 代码如下，页面如图 3-9 和图 3-10 所示。

```
<!-- js-3-6.html -->
<!DOCTYPE html>
<html lang = "en">
  <head>
    <meta charset = "UTF-8" />
    <title>表单提交、重置事件属性应用</title>
    <style type = "text/css">
     fieldset {width: 350px;height: 150px;margin: 0 auto;padding: 10px 20px;}
    </style>
    <script type = "text/JavaScript">
     function $(id){return document.getElementById(id);}     //通过 ID 获取页面元素
     function checkInfo(){
      //用户名长度大于或等于 8,密码长度大于或等于 8
      var name1 = $("myName").value,psw1 = $("myPsw").value;
      if(name1.length >= 8 && psw1.length >= 8 ){
        var msg  = "用户名:" + name1 + "\n 密码是:" + psw1;
        $("info").innerHTML = msg;                          //给 p 添加提示信息
       return true;
      }else{
        $("info").innerHTML = "用户名或密码长度不合法!";    //给 p 添加提示信息
       return false
      }
     }
     function clearInfo(){ $("info").innerHTML = "将数据清空";}      //给 p 添加提示信息
    </script>
  </head>
  <body>
    <form onSubmit = "return checkInfo();" onReset = "clearInfo()">
      <fieldset>
        <legend align = "center">表单数据验证</legend>
        <label>用户名: </label>
        <input type = "text" id = "myName" placeholder = "请输入用户名(长度>= 8)" /><br/>
        <label>密  码: </label>
         <input type = "password" id = "myPsw" placeholder = "请输入密码(长度>= 8)"/>
         <br/>
        <input type = "submit" value = "提交" />
        <input type = "reset" value = "重置" />
        <p id = "info"></p>
      </fieldset>
```

```
39.        </form>
40.     </body>
41.  </html>
```

代码中第 11～24 行定义了三个函数，分别是 $(id)、checkInfo()和 clearInfo()；第 28 行为表单设置了 onSubmit 和 onReset 事件属性，并分别绑定事件处理函数。

当单击“提交”按钮，触发 Submit 事件，调用代码“return checkInfo();”，这段代码将调用 checkInfo()，获取输入框中的用户名和密码，并判断用户名和密码的长度是否大于或等于 8，分别在<p>标记内提示不同的信息。当单击“重置”按钮时，触发 reset 事件，调用代码“clearInfo()”，执行后在<p>标记内显示“将数据清空”信息。

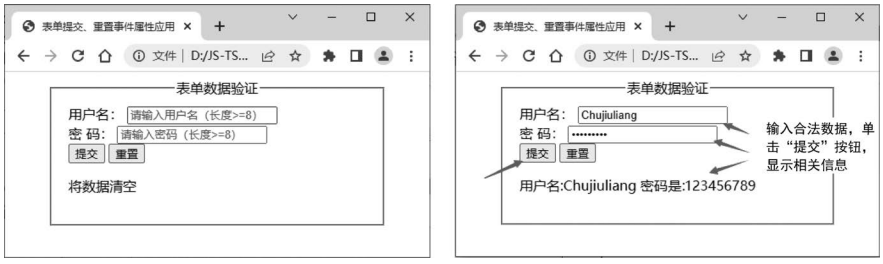

图 3-9　初始页面和输入合法数据单击“提交”按钮时的提示信息

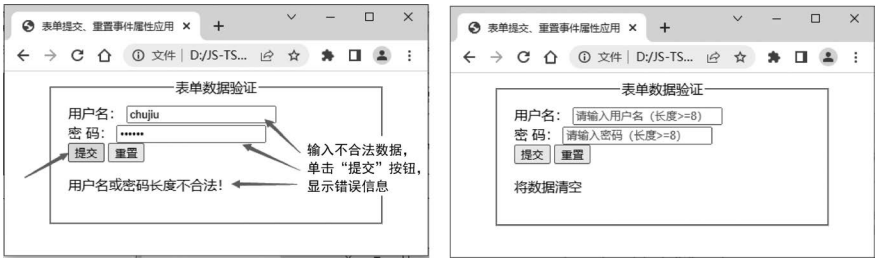

图 3-10　输入不合法数据单击“提交”按钮的提示信息和重置时的提示信息

3.2.3　onFocus 与 onBlur 事件属性

当 HTML 表单控件获得焦点时会触发 focus 事件，当表单控件失去焦点时会触发 blur 事件。当单击表单中的按钮时，该按钮就获得了焦点；当单击表单中的其他区域时，该按钮就失去了焦点。

【例 3-7】　onFocus 与 onBlur 事件属性实战。

JS 代码如下，页面如图 3-11 和图 3-12 所示。该例的难点是如何实现对对象的样式进行控制。可以通过 HTML DOM style 对象的属性进行相关属性的设置，从而达到设置或获取相关属性的值。初始时<input>标记的外在表现边框为有颜色的虚线，获得焦点时边框变成有颜色的实线边框。button 按钮失去焦点时，将改变背景颜色。

```
1.  <!-- js-3-7.html -->
2.  <!DOCTYPE html>
3.  <html lang="en">
4.    <head>
5.      <meta charset="UTF-8" />
6.      <title>获得/失去焦点测试</title>
7.      <script type="text/JavaScript">
8.       function $(id){return document.getElementById(id)}//通过 ID 获取页面元素
9.       function loseFocus(){
10.         $("btn").style.background="#F0F0FF"          //通过 style 对象来设置背景色
11.       }
12.      function changeStyle(id){
13.        $(id).style.border="1px solid #345678";       //通过 style 对象来设置边框
```

```
14.          $(id).style.background = "white";                    //通过 style 对象来恢复背景色
15.        }
16.      </script>
17.      <style>
18.        input {border-radius: 10px;   border: 1px dashed #123456;
19.          margin: 5px;   height: 25px;   background-color: white;   }
20.          fieldset {margin: 0 auto; width: 300px; height: 160px; padding: 30px; text-align:
           center;}
21.        [type = "button"] { width: 200px;   height: 30px;   } /* 属性和值选择器 */
22.      </style>
23.    </head>
24.    <body>
25.      <form autocomplete = "off">
26.        <fieldset>
27.          <legend>获得/失去焦点事件属性应用</legend>
28.          姓名:<input   type = "text"   id = "myName"   size = "30"
29.            onFocus = "changeStyle('myName')"   /><br/>
30.          住址:<input   type = "text"   id = "myAddr"   value = ""
31.            size = "30"   onFocus = "changeStyle('myAddr')"   /><br />
32.          <input   type = "button"   id = "btn"   value = "获得/失去焦点触发事件"
33.            onfocus = "changeStyle('btn')"   onblur = "loseFocus()"   />
34.        </fieldset>
35.      </form>
36.    </body>
37.  </html>
```

图 3-11 onFocus 与 onBlur 事件属性应用初始页面

图 3-12 onFocus 与 onBlur 事件属性生效时页面

3.3 鼠标事件

在 HTML 页面中，通过鼠标对页面中的对象(元素)进行操作时会触发鼠标事件。当单击时会触发 Click 事件，双击时会触发 DblClick 事件，按下鼠标按键后再松开时会触发 MouseUp 事件等。下面对一些常用的鼠标事件进行介绍。

3.3.1 onClick 与 onDblClick 事件属性

鼠标事件指用鼠标对页面中的对象(元素)进行单击、双击操作时触发的事件。当单击页面中的按钮时可以触发鼠标单击事件，当双击页面中的按钮时可以同时触发鼠标单击、鼠标双击事件。语法如下。

```
< input type = "button" name = "click" value = "鼠标单击" onClick = "alert('单击了!')">
< input type = "button" name = "click"  value = "鼠标双击" onDblClick = "alert('双击了!')">
```

【例 3-8】 onClick 与 onDblClick 事件属性实战。

JS 代码如下，页面如图 3-13 所示。

```
<!-- js-3-8.html -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>onClick 与 onDblClick 事件属性</title>
    <script type="text/JavaScript">
     function $(id){return document.getElementById(id);}
     function copyText(){$("target").value=$("source").value;}
     function copySplitText(){$("target").value=$("source").value.split("");} //分隔
     //split()
    </script>
    <style>
     fieldset{margin: 0 auto;  width: 450px;height: 160px;
       padding: 10px;   text-align: center;  }
     input { border-radius: 10px; border: 1px dashed #334455; height: 26px;  }
    </style>
  </head>
  <body>
    <form method="post" action="">
      <fieldset>
        <legend>onClick 与 onDblClick 事件属性</legend>
        原始内容:<input type="text" id="source"  value="ABCDEF 文本"
          placeholder="请输入文本内容"  /><br />
        目标内容:<input type="text" id="target" readonly /><br /><br />
        <input type="button" value="请单击,复制文本" onclick="copyText();" />
          <input type="button" value="请双击,分隔文本后复制" ondblclick=
          "copySplitText();" />
      </fieldset>
    </form>
  </body>
</html>
```

代码中第 8～10 行定义了三个函数，分别为 $(id)、copyText()、copySplitText()；第 22～24 行定义了两个单行文本输入框，且第二个文本框为只读；第 25、26 行定义了两个普通按钮，分别设置了 onCilck 和 onDblClick 事件属性。在第一个单行文本输入框中输完内容后，单击“请单击，复制文本”按钮时会触发 Click 事件，调用 copyText()函数完成文本框内容的复制。单击“请双击，分隔文本后复制”按钮时会触发 DblClick 事件，调用 copySplitText()函数完成先将文本分隔再复制文本内容。

图 3-13 onClick 与 onDblClick 事件属性应用

3.3.2 onMouseOver、onMouseOut、onMouseDown、onMouseUp 事件属性

鼠标事件除了常用的 Click 事件之外，还有鼠标进入页面元素 MouseOver 事件、鼠标退出页面元素 MouseOut 事件和鼠标按键检测 MouseDown 及 MouseUp 事件等。利用这些事

件属性绑定相关事件处理函数，就可以实现相关功能。

【例 3-9】 鼠标移动事件属性实战。

JS 代码如下，页面如图 3-14 和图 3-15 所示。该例要求使用 HTML DOM style 对象的相关属性和 HTML 对象的 innerHTML 属性来进行编程。

```
<!-- js-3-9.html -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>鼠标移动事件</title>
    <script type="text/JavaScript">
     function $(id){return document.getElementById(id);}
     function mouseOver(){
       $('div1').style.border ="10px dashed #AABBCC";
       $('div1').style.background ="#AFFAFC";
       $("content").innerHTML="鼠标移动事件响应-mouseOver"
     }
     function mouseOut(){
       $('div1').style.border ="10px dotted #BDBDBD";
       $('div1').style.background ="#F1F2F3";
       $("content").innerHTML="鼠标移动事件响应-mouseOut"
     }
     function mouseDown(){
       $('div1').style.border ="10px solid #FDFBBD";
       $('div1').style.background ="#F2F1F3";
       $("content").innerHTML="鼠标移动事件响应-mouseDown"
     }
    </script>
    <style type="text/css">
     div {width: 300px;  height: 180px; margin: 0 auto;
      border: 10px double #99aadd; text-align: center;
     }
    </style>
  </head>
  <body>
    <div  id="div1" onmouseover="mouseOver()"
     onmouseout="mouseOut()"  onmousedown="mouseDown()"  >
     <h3>鼠标移动事件响应</h3>
     <p id="content">鼠标移动事件响应。</p>
    </div>
  </body>
</html>
```

代码中第 7～24 行定义了 4 个 JavaScript 函数，分别为 $(id)、mouseOver()、mouseOut()和 mouseDown()，后三个函数均是实现边框样式和< p >标记内容的改变。第 32 行定义了一个 div，并分别设置 onMouseOver、onMouseOut 和 onMouseDown 等事件属性。当鼠标悬停、移出和按下按键时触发相关事件，执行改变 div 边框样式和 p 的内容。

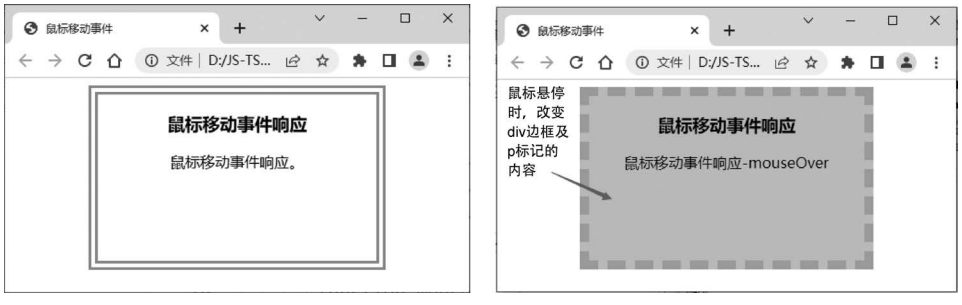

图 3-14 页面初始效果和鼠标悬停时的效果

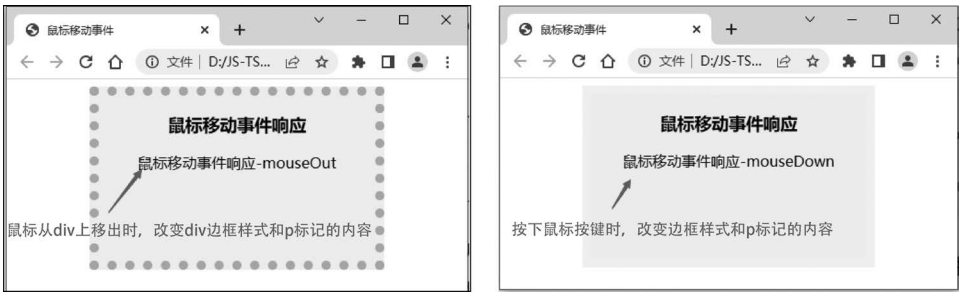

图 3-15　鼠标移出和鼠标按键按下时的效果

3.4　键盘事件

键盘事件主要分为 KeyDown、KeyPress 及 KeyUp，分别用来检测键按下、键按下后松开及键完全松开等动作。通过 Window 对象的 event.keyCode 可以获得按键对应的键码值。常用的字母和数字键的键码值(keyCode)对应表如表 3-2 所示。

表 3-2　字母和数字键的键码值对应表

键　　位	码值	键　　位	码值
0～9(数字键)	48～57	A～Z(字母键)	65～90
BackSpace(退格键)	8	Tab(制表键)	9
Enter(回车键)	13	Space(空格键)	32
Shift	16	Control	17
Alt	18	Caps Lock	20
Home	36	end	35
←(左箭头键)	37	↑(上箭头键)	38
→(右箭头键)	39	↓(下箭头键)	40

【例 3-10】　键盘操作事件属性实战——通过移动 4 个方向键在父 div 中移动子 div。JS 代码如下，页面如图 3-16 和图 3-17 所示。

该案例中需要使用到元素的 HTML DOM Element offsetLeft 属性和 HTML DOM Element offsetTop 属性。其中，offsetLeft 属性返回相对于父元素的左侧位置(以像素计)，此属性为只读。offsetTop 属性返回相对于父元素的顶部位置(以像素计)，此属性为只读。通过动态指定事件处理函数来实现操作 4 个方向键移动子 div(移动速度为 10 像素/次)。代码如下。

```
document.onkeydown = function(event){}
```

```
<!-- js-3-10.html -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>键盘事件的应用</title>
    <style>
     div {margin: 0;padding: 0; border: 0;  }
     #myDiv { top: 0; left: 0;  width: 100px; height: 100px;
       background-color: #D9DADB;position: absolute; font-size: 14px;
     }
     #parentDiv { margin: 0 auto; width: 400px; height: 400px;
       background-color: #F9FAFB; position: relative;overflow: hidden;
     }
```

```
    </style>
    <script type="text/JavaScript">
     function $(id){return document.getElementById(id) }
      document.onkeydown = function(event){
       var myDiv1 = $("myDiv");        //获取 div 元素
       event = event||window.event;    //解决事件对象的兼容性问题
       var speed = 10;                 //定义移动速度
       switch(event.keyCode){
        case 37:                       //左箭头
         if(myDiv1.offsetLeft >= 10){
           myDiv1.style.left = myDiv1.offsetLeft - speed + 'px';
         }
         break;
         case 39:                      //右箭头
          if(myDiv1.offsetLeft <= 290){
            myDiv1.style.left =  myDiv1.offsetLeft + speed + 'px';
          }
           break;
        case 38:                       //上箭头
        if(myDiv1.offsetTop >= 10){
            myDiv1.style.top =   myDiv1.offsetTop - speed + 'px';
         }
         break;
         case 40:                      //下箭头
         if(myDiv1.offsetTop <= 290){
             myDiv1.style.top =   myDiv1.offsetTop + speed + 'px';
          }
           break;
        }
     //在子 div 中显示 offsetLeft 和 offsetTop 值
        myDiv1. innerHTML = " offsetLeft:" + myDiv1. offsetLeft + " \ noffsetTop:" +
        myDiv1.offsetTop
     }
    </script>
  </head>
  <body>
    <h3 align="center">键盘操作事件属性实战</h3>
    <div id="parentDiv">
     <div id="myDiv"></div>
    </div>
  </body>
</html>
```

图 3-16 未操作 4 个方向键时初始页面

图 3-17　操作键盘上 4 个方向键移动子 div 的页面

3.5　窗口事件

窗口事件是指浏览器窗口在调整大小 Resize、DOM 内容加载完成 DOMContentLoaded、窗口滚动 Scroll 以及加载页面 Load 或卸载页面 beforeUnload 时触发的事件。

3.5.1　onResize 与 onScroll 事件属性

onResize 事件会在窗口或框架被调整大小时发生，onScroll 事件会在文档被滚动时发生。

【例 3-11】　窗口事件属性实战 1——onResize 与 onScroll 事件属性。

JS 代码如下，页面如图 3-18 和图 3-19 所示。

```
<!-- js-3-11.html -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>JS 窗口事件</title>
    <style>
      #containter {width: 200px; height: 250px;margin: 0 auto;text-align: center;}
      #parentDiv {width: 200px; height: 200px;border: 1px dashed black; overflow: scroll;}
      #subDiv {width: 400px; height: 400px;background-color: #f1f1f2;text-align: left;}
    </style>
    <script>
      function changeSize() {
        var w = window.innerWidth;
        var h = window.innerHeight;
        document.getElementById("content").innerHTML ="窗口宽: " + w + "窗口高: " + h;
      }
      var count = 0;
      function countScroll() {
        count++;
        document.getElementById("scrollNum").innerHTML = "滚动次数: " + count;
      }
    </script>
  </head>
  <body onresize="changeSize()">
    <div id="containter">
      <h3>窗口事件属性实战</h3>
      <div id="parentDiv" onscroll="countScroll()">
        <div id="subDiv">这是子 div</div>
      </div>
      <p id="content"></p>
      <p id="scrollNum"></p>
```

```
32.       </div>
33.     </body>
34.   </html>
```

图 3-18 操作窗口事件之前初始页面

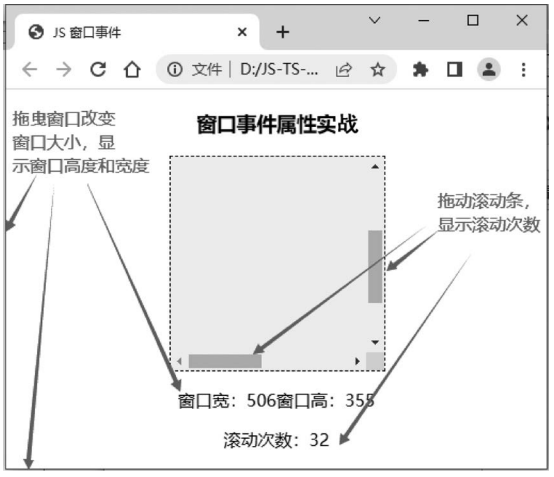

图 3-19 改变窗口大小和拖曳滚动条时页面

3.5.2 onDOMContentLoaded、onLoad 与 onBeforeUnload 事件属性

1. DOMContentLoaded 事件的触发

DOMContentLoaded 事件在 HTML 文档(不包括样式表、图像和 Flash 等)加载完毕，并且 HTML 所引用的内联 JS 和链接外部 JS 的同步代码都执行完毕后触发。

2. Load 事件的触发

当页面 DOM 结构中的 JS、CSS、图像以及 JS 异步加载的 JS、CSS、图像都加载完成之后，才会触发 Load 事件。

3. BeforeUnload 事件触发

当窗口文档及其资源即将卸载时，会触发该事件。该文档仍然可见时事件仍可取消。该事件可以弹出对话框，提示用户是继续浏览页面还是离开当前页面。对话框默认的提示信息根据不同的浏览器有所不同，标准的信息类似“确定要离开此页吗?”。该信息不能删除。

采用添加监听器的方法来解决这一问题。在< body >标记内增加如下代码，实现当关闭窗口时可以触发 BeforeUnload 事件，而不需要在< body >标记上设置 onBeforeUnload 事件属性。代码如下。

```
1.  < script type = "text/JavaScript">
2.      window.onbeforeunload = function() {return "onbeforeunload is work";}
3.  </script >
```

【例 3-12】 窗口事件属性实战 2——onDomContentLoaded、onLoad 与 onBeforeUnload 事件属性。

JS 代码如下，页面如图 3-20 和图 3-21 所示。

(1) HTML 文档 js-3-12.html 代码如下。

```
1.  <!-- js - 3 - 12.html -->
2.  <! DOCTYPE html >
3.  < html lang = "en">
4.    < head >
5.      < meta charset = "UTF - 8" />
6.      < title >窗口事件的应用 - load、DOMContentLoaded 等</title >
7.      < link rel = "stylesheet" href = "js - 3 - 12.css" />
8.    </head >
```

```
    <body>
      <div>
        <h4>load、DOMContentLoaded、beforeunload 应用</h4>
        <p onclick="$('info').value+='单击了段落!\n'">单击我!</p>
        <img src="image-3-12.png" alt="" />
        <textarea id="info" rows="5" cols="40"></textarea>
      </div>
      <script type="text/JavaScript">
       document.addEventListener("DOMContentLoaded", function(e) {
        $("info").value+="DOM 内容装载完成...\n";
       });
       window.onload=function(){$("info").value+="页面装载完成...\n";}
       window.onbeforeunload = function() {
        $("info").value+="onbeforeunload is work!\n"
        return "onbeforeunload is work!";
       }
       function $(id){return document.getElementById(id);}
      </script>
    </body>
  </html>
```

(2) 链接的外部 CSS 文件 js-3-12.css 代码如下。

```
/* js-3-12.css */
div {margin: 0 auto;height: 200px; width: 400px; text-align: center; }
p {margin: 0 auto;width: 200px;height: 40px; border: 1px solid black;}
```

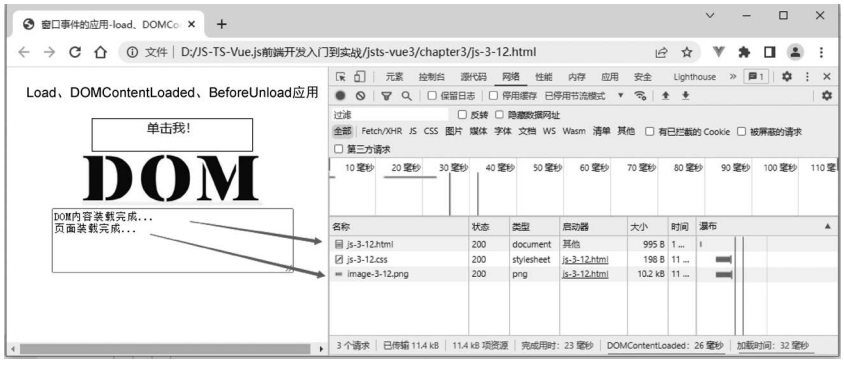

图 3-20 页面加载完成时页面

从图 3-20 可以看出，先使用 26ms 完成 DOMContentLoaded 事件，接着才完成 Load 事件，总用时 32ms。当用户单击段落时，会向多行文本域中添加一行信息为“单击了段落!”，如图 3-21 所示。当用户单击标题栏右侧的“关闭”按钮时，会出现如图 3-22 所示的对话框，先单击“取消”按钮，然后同样会向多行文本域中添加一条信息为“onbeforeunload is work!”。然后再次单击浏览器标题栏右侧的“关闭”按钮，此时触发 BeforeUnload 事件，关闭窗口。

图 3-21 加载页面时效果图

图 3-22 单击时效果图

项目实战 3

1. JavaScript 事件处理实战——学生信息录入

扫一扫

文本

扫一扫

视频

2. JavaScript 事件处理实战——图书选择

扫一扫

文本

扫一扫

视频

小结

本章主要介绍 JavaScript 脚本中的事件、事件句柄和事件处理函数概念以及三者之间的关联关系，重点介绍了两种事件处理方式，分别是静态指定和动态指定，也可以采用事件注册，给指定的元素添加事件监听器。同时介绍了 DOM 事件流，包含事件捕获阶段、处理目标阶段和事件冒泡阶段三个阶段。

最后重点介绍了常用的 HTML 事件、鼠标事件、键盘事件以及窗口事件等。在 HTML 事件中，详细介绍了 Change、Select、Reset、Submit、Focus 与 Blur 事件。在鼠标事件中，详细介绍了鼠标 Click、DblClick 及鼠标移动事件。在窗口事件中，主要介绍了窗口调整大小、窗口滚动事件、DOM 内容装载完成事件、装载事件和卸载事件。通过给这些事件绑定相关事件处理函数来解决实际工程的业务需求。

练习 3

扫一扫

习题

扫一扫

自测题