第 5 章　BOM

BOM（Browser Object Model，浏览器对象模型）定义了浏览器的操作对象，本章将讨论如何通过 BOM 操作浏览器窗口、网址、导航操作，以及屏幕信息等内容。

5.1　window 对象

window 对象表示浏览器窗口，开发者可以通过 window 对象修改浏览器的外观、控制状态栏信息等。

window 是整个 BOM 结构中的主对象，其中包含一些子对象，如：

- document 对象，表示页面的内容，将在第 6 章详细讨论。
- location 对象，用于处理页面的 URL 内容。
- navigator 对象，可以获取用户浏览器的相关信息。
- history 对象，用于控制浏览历史操作，如 back() 方法执行后退操作，forward() 方法执行前进操作，go() 方法指定前进（正数）或后退（负数）的记录数。实际开发中，建议用户使用浏览器的“前进”与“后退”功能。
- screen 对象，可以获取用户的屏幕信息。

开发中，可以通过 window 对象的属性或直接使用这些对象，接下来先了解 window 对象的一些常用操作。

5.1.1　对话框

在 JavaScript 中，可以使用三种基本的对话框，即：

- alert() 方法，显示消息对话框，参数指定显示的文本内容。
- prompt() 方法，显示输入对话框，用于获取用户输入的数据。参数一指定提示信息，参数二指定默认值，这两个参数都是可选的。
- confirm() 方法，显示确认对话框，选择“确定”按钮时返回 true 值，否则返回 false 值。

这三个方法可以使用 window 对象调用，也可以直接调用。下面的代码演示了 alert() 方法的使用。

```
<script>
    window.alert("Hello BOM");
</script>
```

打开页面，会显示如图 5-1 所示的对话框（Google Chrome 浏览器）。

prompt() 方法可以获取用户输入的文本内容，如下面的代码。

```
<script>
var input = prompt("请输入一个数值", "0");
```

```
        //
        if (Number(input) % 2 == 0)
            document.write("输入的是偶数");
        else
            document.write("输入的不是偶数")
    </script>
```

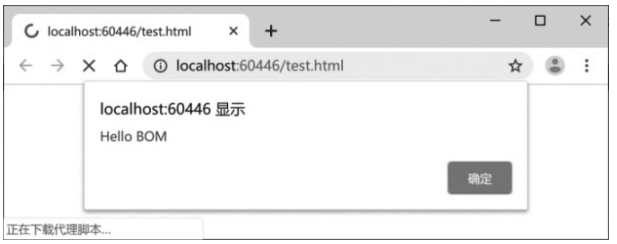

图　5-1

打开页面会显示如图 5-2 所示的对话框（Google Chrome 浏览器），然后会判断输入的内容是否为偶数。

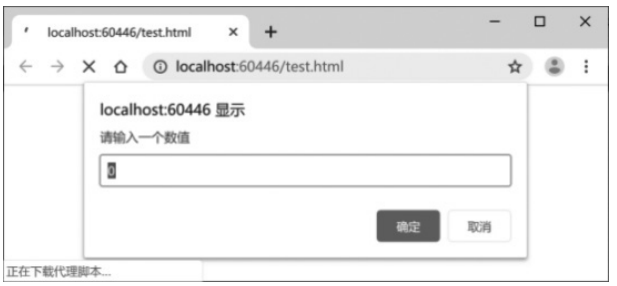

图　5-2

confirm() 方法会让用户做出选择，如下面的代码。

```
    <script>
        var result = confirm("确定要继续吗？");
        //
        if (result)
            document.write("确认操作");
        else
            document.write("取消操作")
    </script>
```

执行代码会显示如图 5-3 所示的对话框（Google Chrome 浏览器），然后显示用户的选择结果。

图　5-3

5.1.2　onload 事件

window.onload 事件会在页面元素完全载入后执行，在此添加页面元素的操作代码会比较安全，可以避免因页面没有完全载入而产生的错误。可以在此事件中添加页面的初始化代码。

指定 onload 事件代码时，一般会使用一个函数，如下面的代码，会在页面载入后显示一条信息。

```
<script>
    window.onload = function () {
        document.write("页面已载入");
    }
</script>
```

页面载入后会显示如图 5-4 所示的内容。

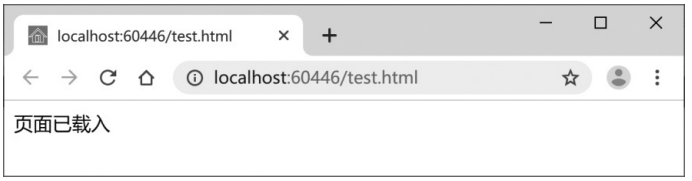

图　5-4

实际开发过程中，可能需要向 onload 事件添加多个函数，可以参考下面的代码完成。

```
<script>
    // 添加 onload() 执行函数
    function addOnLoadFunc(fn) {
        if (typeof window.onload === "function") {
            var oldFn = window.onload;
            window.onload = function () {
                oldFn();
                fn();
            };
        } else {
            window.onload = fn;
        }
    }

    // 三个初始化函数
    function init1() {
        alert("init1");
    }
    function init2() {
        alert("init2");
    }
    function init3() {
        alert("init3");
    }

    // 添加到 onload 事件
    addOnLoadFunc(init1);
```

```
        addOnLoadFunc(init2);
        addOnLoadFunc(init3);
    </script>
```

其中，addOnLoadFunc(fn) 函数用于向 window.onload 事件添加函数。其中，首先判断 window.onload 是否是函数类型，即是否已添加了函数，如果已添加过函数，使用 oldFn 对象备份 window.onload 中的函数，然后，将 window.onload 设置为新的函数，新函数中需要调用 oldFn() 和参数带入的 fn() 函数。如果 window.onload 不是函数类型，则直接将参数 fn 指定给 onload 事件。

这里共向 window.onload 事件添加了三个函数，页面加载后会显示三个消息对话框。

此外，还可以将 addOnLoadFunc(fn) 函数定义在 JavaScript 文件中，以便在项目中重复使用。本书通用代码会定义在 /js/Common.js 文件中。

5.1.3 open() 与 close() 方法

window.open() 方法用于在页面中打开另外一个页面，并会返回打开的窗口对象，方法的参数包括：

- 参数一，打开资源的地址。
- 参数二，可选，指定新窗口的名称，可以使用自定义的名称，或使用特殊值指定新窗口的打开方式，如 _blank 打一个新的窗口（或标签）显示页面，_self 在当前窗口（或标签）显示页面等。
- 参数三，可选，使用字符串指定新窗口的显示参数。

close() 方法用于关闭窗口，如 window.close() 会关闭当前窗口，在 window.open() 方法返回的对象中调用 close() 方法会关闭打开的窗口。

IE 浏览器中调用 window.close() 方法关闭当前窗口时会弹出确认对话框，如果想直接关闭当前窗口，可以使用类似下面的代码。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title></title>
</head>
<body>
<button type="button" onclick="window.close();">关闭窗口 1</button>
<button type="button" onclick="closeMe();">关闭窗口 2</button>
</body>
</html>
<script>
    function closeMe() {
        var w = window.open("","_self");
        w.close();
    }
</script>
```

在 IE11 浏览器中单击“关闭窗口 1”按钮时会弹出图 5-5 中的对话框，而单击“关闭窗口 2”按钮时会直接关闭浏览器窗口。

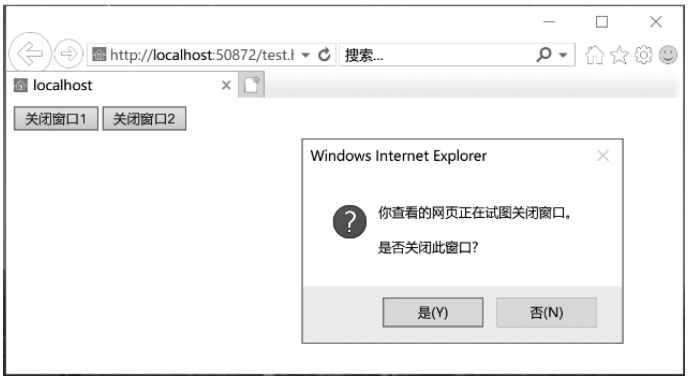

图　5-5

下面的代码在页面载入时会在一个新窗口中打开作者的个人网站。

```
<script>
    window.open("http://caohuayu.com","_blank");
</script>
```

一些情况下，浏览器或安全软件会拦截弹出式窗口，图 5-6 中就是 Google Chrome 浏览器中的提示。

图　5-6

弹出式窗口并不是一个好的网站浏览体验，因为在实际应用中，更多的是弹出广告，而不是真正有意义的内容。

项目中，如果真的需要动态创建窗口，还可以通过 window.open() 方法的第三个参数设置窗口的外观，常用的属性有：

- ❑ menubar，是否显示菜单栏，包括 yes 和 no 值。
- ❑ location，是否显示地址栏，包括 yes 和 no 值。
- ❑ resizable，是否允许修改窗口尺寸，包括 yes 和 no 值。
- ❑ scrollbars，是否允许显示滚动条，包括 yes 和 no 值。
- ❑ status，是否显示状态栏，包括 yes 和 no 值。
- ❑ width，指定窗口的宽度，设置为一个数值，单位是像素。
- ❑ height，指定窗口的高度，设置为一个数值，单位是像素。
- ❑ top 和 left，指定窗口左上角在屏幕中的坐标，单位是像素。

下面的代码会创建并显示一个提示页面。

```
<script>
    var features = "menubar=no,resizable=no,width=500,height=150";
    var win = window.open("about:blank", "new window", features);
    win.document.open();
```

```
        win.document.write("<h1>小提示 ^_^</h1>");
        win.document.write(win.name);
        win.document.close();
        //
        window.onclick = function () {
            win.close();
        }
    </script>
```

在 features 变量中，指定新的窗口不显示菜单栏、不允许改变尺寸，并设置宽度为 500 像素，高度为 150 像素。

请注意，在 window.open() 方法的第一个参数使用了 about:blank，这样，打开的窗口会是一个完全空白的页面。接下来调用 win.doucment 对象的三个方法，分别是：

- open() 方法，打开页面内容的写入流。
- write() 方法，向页面中写入内容，我们已多次使用这个方法。另一个相关的方法是 writeln()，其功能是在写入参数指定的内容后，还会添加一个换行符（\n）。
- close() 方法，关闭页面内容的写入流。

执行代码会显示如图 5-7 所示的新窗口（需要用户允许网站弹出窗口）。

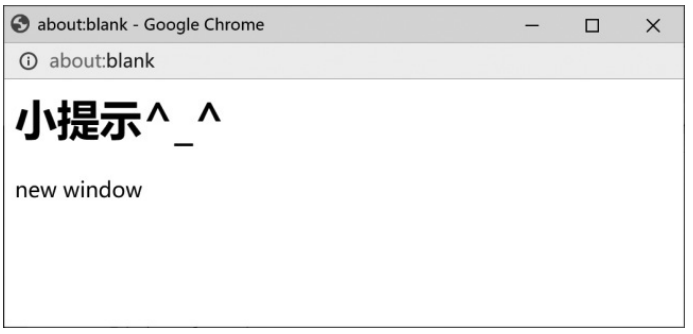

图 5-7

此外，在原页面的 window.onclick 事件中还可定义当单击原窗口时，调用 win.close() 方法关闭弹出的窗口。

5.2 location 对象

location 对象同时定义为 window 和 document 对象的属性，用于处理当前页面的网址（URL）信息。对象的常用成员包括：

- href 属性和 toString() 方法，都可以获取完整的 URL 内容。
- host 和 hostname 属性，获取 URL 中的服务器主机名称。
- pathname 属性，获取主机后的资源路径。
- port 属性，获取 URL 中主机的端口信息。
- protocal 属性，获取 URL 中使用的协议。
- search 属性，获取 URL 中 ? 符号及以后的内容，即 URL 中的查询参数。
- hash 属性，获取 URL 中 # 符号及以后的内容。

下面的代码会在页面中显示一些 URL 信息。

```
<script>
    document.write(location.href);
    document.write("<br/>");
    document.write(location.protocol);
    document.write("<br/>");
    document.write(location.host);
    document.write("<br/>");
    document.write(location.port);
    document.write("<br/>");
    document.write(location.pathname);
</script>
```

在 IIS Express 中打开页面，会显示类似图 5-8 所示的内容。

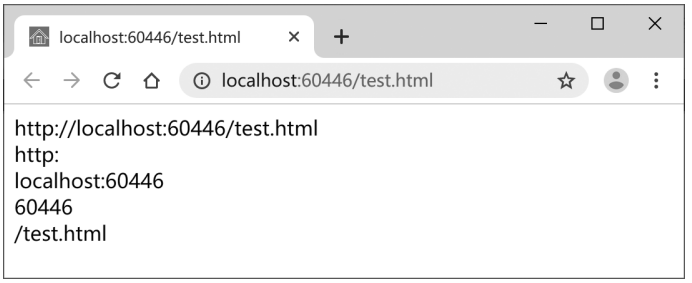

图　5-8

本例中分别显示了完整的 URL、协议、主机名、端口号，以及资源在服务器中的路径。

此外，在 location 对象中还包含 assign()、replace() 和 reload() 等方法，用于处理访问记录或重新载入页面，实际应用中，建议用户使用浏览器进行操作，如前进、后退和刷新等操作。

5.3　navigator 对象

navigator 对象可以帮助开发者获取用户的操作系统和浏览器相关信息，对象的常用成员有：

- ❑ appCodeName 属性，返回浏览器代码名称。
- ❑ appName 属性，返回浏览器名称。
- ❑ appVersion 属性，返回操作系统版本。
- ❑ language 属性，返回浏览器语言，其中 zh-CN 表示简体中文，en-US 表示美国英语。
- ❑ platform 属性，返回系统平台名称。

下面的代码演示了这几个属性的使用。

```
<script>
    document.write(navigator.appCodeName);
    document.write("<br>");
    document.write(navigator.appName);
    document.write("<br>");
    document.write(navigator.appVersion);
```

```
    document.write("<br>");
    document.write(navigator.language);
    document.write("<br>");
    document.write(navigator.platform);
</script>
```

大家可以观察在不同浏览器中显示的结果。

- cookieEnabled 属性，返回一个布尔值，表示浏览器是否开启 Cookie 功能，这是一种基于文本的客户端数据存储技术。
- plugins 属性，返回一个包含浏览器已安装插件的数组，如下面的代码，可以显示所有已安装插件的名称。

```
<script>
    var arr = navigator.plugins;
    for (var i = 0; i < arr.length; i++) {
        document.write(arr[i].name);
        document.write("<br>");
    }
</script>
```

如果需要检测浏览器中是否安装了 Flash Player 插件，可以使用类似下面的代码。

```
<script>
    function hasFlash() {
        var arr = navigator.plugins;
        for (var i = 0; i < arr.length; i++) {
            if (arr[i].name == "Shockwave Flash")
                return true;
        }
        return false;
    }
    //
    if (hasFlash())
        document.write(" 已安装 Flash 插件 ");
    else
        document.write(" 没有安装 Flash 插件 ");
</script>
```

一般来讲，网站是不应该挑选浏览器的，但如果真的需要某个类型或版本的浏览器，可以从 navigator.userAgent 属性返回的信息中获取。如图 5-9 所示，先观察不同浏览器中 userAgent 属性返回的内容，从上到下分别是 IE11、Firefox 和 Google Chrome 浏览器中显示的信息。

从图 5-9 中可以看到，浏览器的类型和版本并没有什么固定的格式，这就对获取浏览器信息的操作带来了一些困难。不过，可以从另一个角度来考虑问题，可以使用目标驱动法来判断浏览器是不是所需要的类型和版本。

比如，需要检测是否为 Firefox 浏览器时，可以使用类似如下的代码。

```
<script>
    function isFirefox() {
        return navigator.userAgent.indexOf("Firefox") >-1;
    }
    //
```

```
    if (isFirefox())
        document.write(" 正在使用 Firefox 浏览器 ");
    else
        document.write(" 没有使用 Firefox 浏览器 ");
</script>
```

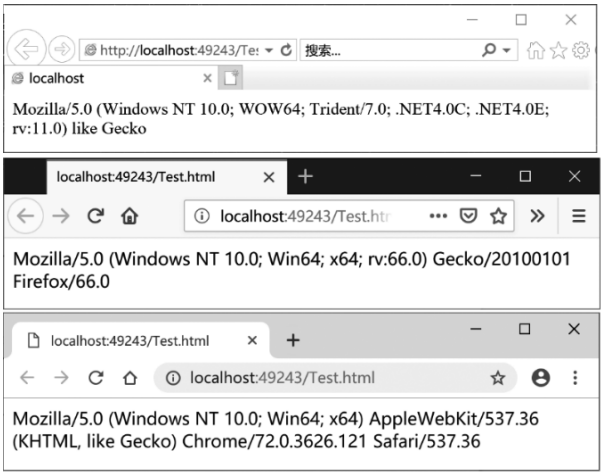

图　5-9

5.4　screen 对象

screen 对象可以返回与屏幕相关的信息，主要的成员包括：

- ❑ width 属性，返回屏幕尺寸的完整宽度（像素）。
- ❑ height 属性，返回屏幕尺寸的完整高度（像素）。
- ❑ availWidth 属性，返回屏幕可用尺寸的宽度（像素）。
- ❑ availHeight 属性，返回屏幕可用尺寸的高度（像素）。
- ❑ colorDepth 属性，返回屏幕显示的颜色位，如 24 位等。

下面的代码用于显示这五个属性的数据。

```
<script>
    document.write(screen.width);
    document.write("<br>");
    document.write(screen.height);
    document.write("<br>");
    document.write(screen.availWidth);
    document.write("<br>");
    document.write(screen.availHeight);
    document.write("<br>");
    document.write(screen.colorDepth);
</script>
```

如果需要将弹出的新窗口放在屏幕的居中位置，可以使用类似下面的代码。

```
<script>
    var winWidth = 500;
```

```
        var winHeight = 150;
        var winTop = (screen.height - winHeight) / 2;
        var winLeft = (screen.width - winWidth) / 2;
        var features = "menubar=no,resizable=no,width=500,height=150,top=" +
            String(winTop) + ",left=" + String(winLeft);
        var win = window.open("about:blank", "new window", features);
        win.document.open();
        win.document.write("<h1>窗口居中</h1>");
        win.document.close();
    </script>
```