第五章 JavaScript BOM

5.1、BOM概述

浏览器对象模型（BOM）使 JavaScript 有能力与浏览器"对话"。

浏览器对象模型（Browser Object Model (BOM)）尚无正式标准。

由于现代浏览器已经（几乎）实现了 JavaScript 交互性方面的相同方法和属性，因此常被认为是BOM的方法和属性。

浏览器对象模型（BOM）可以使我们通过JS来操作浏览器，在BOM中为我们提供了一组对象，用来完成对浏览器的操作，常见的BOM对象如下：

- Window：代表的是整个浏览器的窗口，同时window也是网页中的全局对象
- Navigator：代表的当前浏览器的信息，通过该对象可以来识别不同的浏览器
- Location：代表当前浏览器的地址栏信息，通过Location可以获取地址栏信息，或者操作浏览器跳转页面
- History：代表浏览器的历史记录，可以通过该对象来操作浏览器的历史记录，由于隐私原因，该对象不能获取到具体的历史记录，只能操作浏览器向前或向后翻页，而且该操作只在当次访问时有效
- Screen：代表用户的屏幕的信息，通过该对象可以获取到用户的显示器的相关的信息

这些BOM对象在浏览器中都是作为window对象的属性保存的，可以通过window对象来使用，也可以直接使用。

5.2、Window对象

5.2.1、弹出框

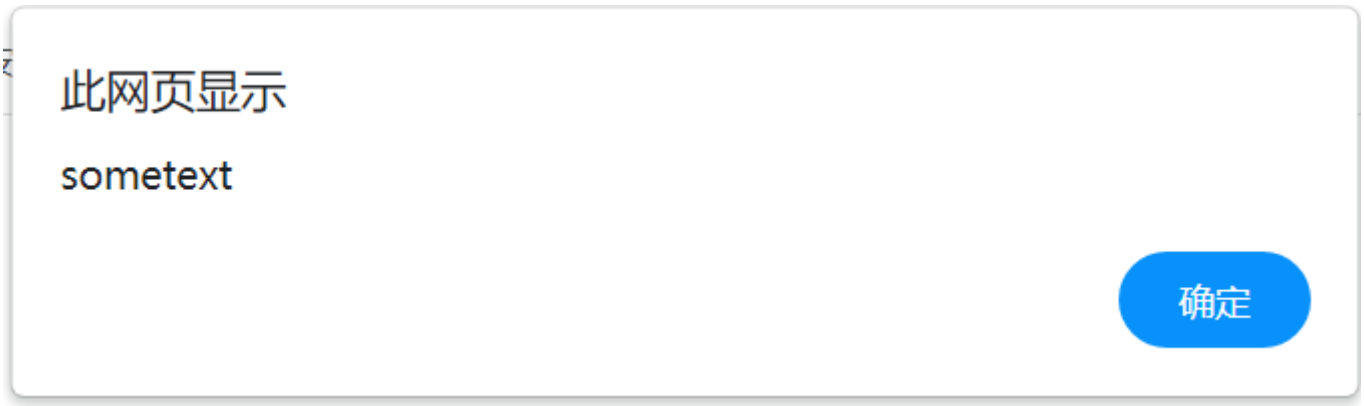
JavaScript 有三种类型的弹出框：警告框、确认框和提示框。

5.2.1.1、警告框

如果要确保信息传递给用户，通常会使用警告框。当警告框弹出时，用户将需要单击“确定”来继续。

语法

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



> 注意：window.alert() 方法可以不带 window 前缀来写。

```
<script>
    alert("我是一个警告框！");
</script>
```

5.2.1.2、确认框

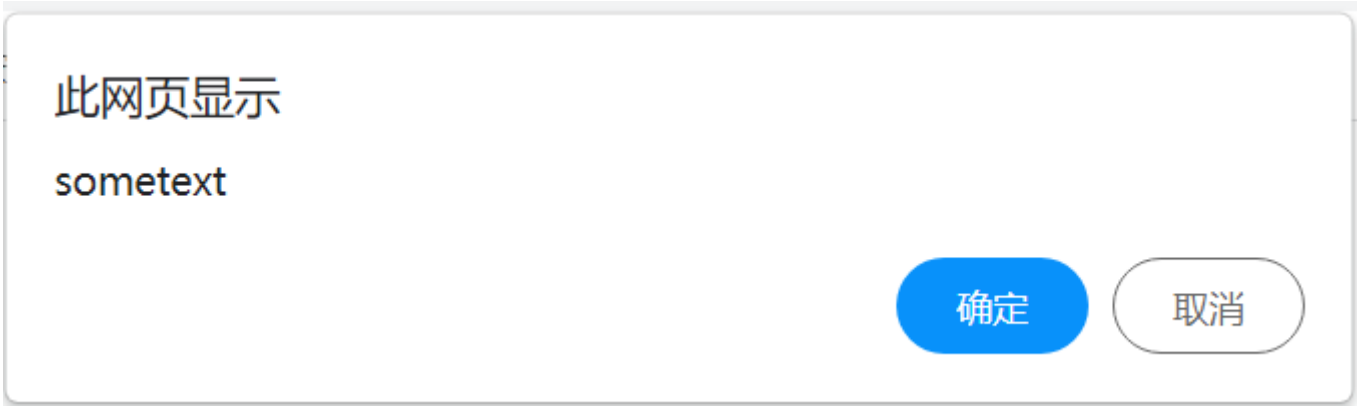
如果您希望用户验证或接受某个东西，则通常使用“确认”框。

当确认框弹出时，用户将不得不单击“确定”或“取消”来继续进行。

如果用户单击“确定”，该框返回 true。如果用户单击“取消”，该框返回 false。

语法

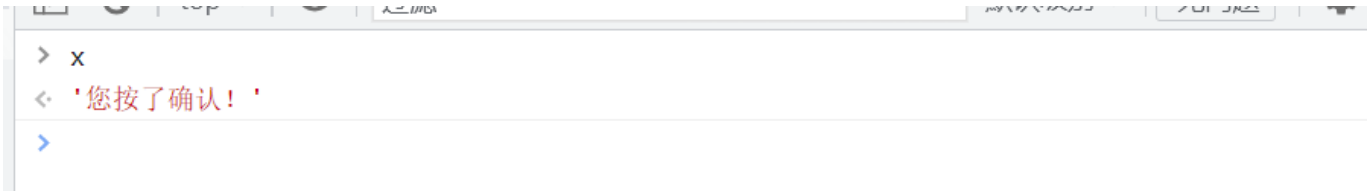
```
<script>
    window.confirm("sometext");
</script>
```



> 注意：window.confirm() 方法可以不带 window 前缀来编写。

实例：

```
<script>
    var r = confirm("请按按钮");
    if (r == true) {
        x = "您按了确认！ ";
    } else {
        x = "您按了取消！ ";
    }
</script>
```



5.2.1.3、提示框

如果您希望用户在进入页面前输入值，通常会使用提示框。

当提示框弹出时，用户将不得不输入值后单击“确定”或点击“取消”来继续进行。

如果用户单击“确定”，该框返回输入值。如果用户单击“取消”，该框返回 NULL。

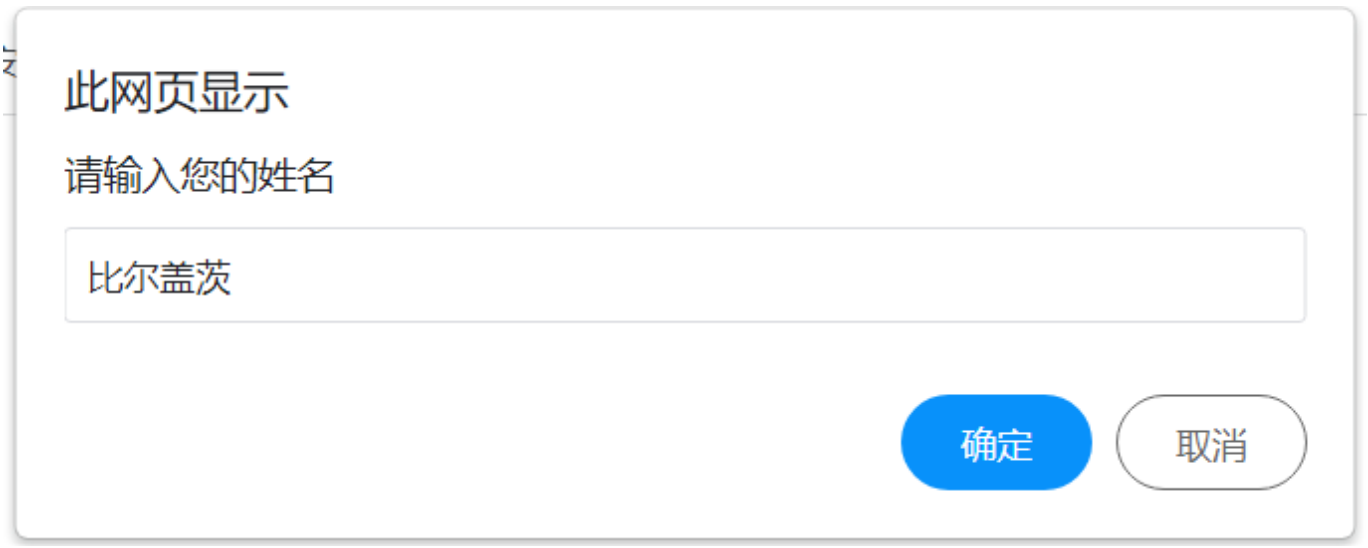
语法

```
<script>
       window.prompt("sometext","defaultText");
</script>
```

> window.prompt() 方法可以不带 window 前缀来编写。

实例

```
<script>
   var person = prompt("请输入您的姓名", "比尔盖茨");
    if (person != null) {
        console.log(person);
    }
</script>
```



5.2.2、定时事件

JavaScript 可以在时间间隔内执行，这就是所谓的定时事件（Timing Events）。

window 对象允许以指定的时间间隔执行代码，这些时间间隔称为定时事件。

通过 JavaScript 使用的有两个关键的方法：

- setTimeout(function, milliseconds)

  在等待指定的毫秒数后执行函数。

- setInterval(function, milliseconds)

  等同于 setTimeout()，但持续重复执行该函数。

> setTimeout() 和 setInterval() 都属于 window 对象的方法。

5.2.2.1、延时器

setTimeout() 方法：延时器

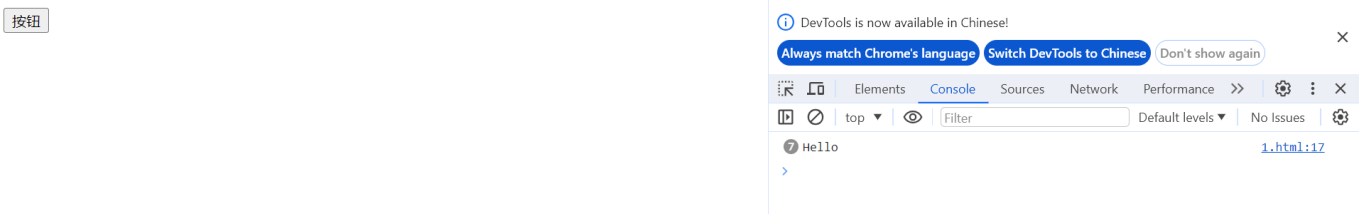
```
<script>
        window.setTimeout(function, milliseconds);
    </script>
```

> 注意：window.setTimeout() 方法可以不带 window 前缀来编写。

- 第一个参数是要执行的函数。
- 第二个参数指示执行之前的毫秒数。

案例演示：单击按钮，等待 3 秒，然后控制台会输出 "Hello"

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
<button id="btn">按钮</button>

<!-- 在这里写JavaScript代码，因为JavaScript是由上到下执行的 -->
<script>
    var btn = document.getElementById("btn");

    btn.onclick = function () {
        // 创建延时器
        var timer = setTimeout(function () {
            console.log("Hello");
        }, 3000);

        // 清除延时器
        // clearTimeout(timer);
    };
</script>
</body>
</html>
```



5.2.2.2、定时器

setInterval() 方法：定时器

setInterval() 方法在每个给定的时间间隔重复给定的函数。

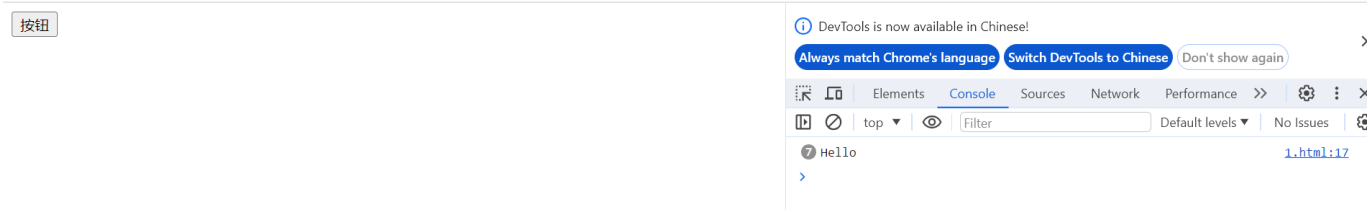
```
<script>
    window.setInterval(function, milliseconds);
</script>
```

> 注意：window.setInterval() 方法可以不带 window 前缀来写。

- 第一个参数是要执行的函数。
- 第二个参数每个执行之间的时间间隔的长度。

案例演示：单击按钮，每隔一秒向控制台输出 "Hello"

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
<button id="btn">按钮</button>

<!-- 在这里写JavaScript代码，因为JavaScript是由上到下执行的 -->
<script>
    var btn = document.getElementById("btn");

    btn.onclick = function () {
        // 创建定时器
        var timer = setInterval(function () {
            console.log("Hello");
        }, 1000);

        // 清除定时器
        // clearInterval(timer);
    };
</script>
</body>
</html>
```



拓展知识：

做一个通用移动函数来实现小汽车（黑色方块）移动的效果

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title></title>
    <style>
        /*控制器样式*/
        .controller {
            width: 600px;
            height: 50px;
            line-height: 50px;
        }

        .controller button {
            outline: none;
            border: none;
            margin: 0px;
            padding: 0px;
            width: 200px;
            height: 50px;
            font-size: 16px;
            line-height: 50px;
            text-align: center;
            background-color: #E9E9E9;
            cursor: pointer;
            float: left;
            -webkit-box-sizing: border-box;
            -moz-box-sizing: border-box;
            box-sizing: border-box;
            border: 2px solid #F0F0F0;
        }

        .controller button:hover {
            background-color: #F9F9F9;
        }


        /*公路样式*/
        .road {
            width: 100%;
            height: 100px;
            position: relative;
            margin-top: 50px;
            background: #3DB1FF;
            opacity: .90;
        }

        .road800 {
            width: 800px;
            height: 100px;
```

```
            background: pink;
            position: absolute;
            top: 0px;
            left: 0px;
            z-index: 1000;
            opacity: .75;
        }

        .road1200 {
            width: 1200px;
            height: 100px;
            background: orange;
            position: absolute;
            top: 0px;
            left: 0px;
            z-index: 500;
        }


        /*小汽车样式*/
        div#car {
            width: 135px;
            height: 100px;
            display: block;
            background: black;
            position: absolute;
            top: 0px;
            left: 0px;
            z-index: 1500;
            -webkit-box-sizing: border-box;
            -moz-box-sizing: border-box;
            box-sizing: border-box;
            /*border: 1px solid #F0F0F0;*/
        }
    </style>
</head>
<body>
<div class="controller">
    <button id="btn1">移动到800PX</button>
    <button id="btn2">移动到1200PX</button>
    <button id="btn3">回家</button>
</div>

<div class="road">
    <div class="road800"></div>
    <div class="road1200"></div>
    <div id="car"></div>
</div>

<!-- 在这里写JavaScript代码，因为JavaScript是由上到下执行的 -->
<script>
```

```
    document.getElementById("btn1").onclick = function () {
        move(document.getElementById("car"), 800);
    };

    document.getElementById("btn2").onclick = function () {
        move(document.getElementById("car"), 1200);
    };

    document.getElementById("btn3").onclick = function () {
        move(document.getElementById("car"), 0);
    };

    /*移动函数*/
    function move(element, target) {
        // 先清理定时器
        clearInterval(element.timeId);
        // 一会要清理定时器(只产生一个定时器)
        element.timeId = setInterval(function () {
            // 获取对象当前的位置
            var current = element.offsetLeft;
            // 每次移动多少像素
            var step = 10;
            // 判断是往正方向走还是往相反方向走
            step = current < target ? step : -step;
            // 每次移动后的距离
            current += step;
            // 判断当前移动后的位置是否到达目标位置
            if (Math.abs(target - current) > Math.abs(step)) {
                element.style.left = current + "px";
            } else {
                // 清理定时器
                clearInterval(element.timeId);
                element.style.left = target + "px";
            }
        }, 20);
    }
</script>
</body>
</html>
```

5.2.3、常用窗口属性

两个属性可用用于确定浏览器窗口的尺寸。

这两个属性都以像素返回尺寸：

- window.innerHeight - 浏览器窗口的内高度（以像素计）
- window.innerWidth - 浏览器窗口的内宽度（以像素计）

浏览器窗口（浏览器视口）不包括工具栏和滚动条。

对于 Internet Explorer 8, 7, 6, 5：

- document.documentElement.clientHeight
- document.documentElement.clientWidth 或
- document.body.clientHeight
- document.body.clientWidth

一个实用的 JavaScript 解决方案（包括所有浏览器）：该例显示浏览器窗口的高度和宽度（不包括工具栏和滚动条）

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<body>

<!-- 在这里写JavaScript代码，因为JavaScript是由上到下执行的 -->
<script>
    var w = window.innerWidth
        || document.documentElement.clientWidth
        || document.body.clientWidth;

    var h = window.innerHeight
        || document.documentElement.clientHeight
        || document.body.clientHeight;

    console.log(w);
    console.log(h);
</script>
</body>
</html>
```

5.2.4、其它窗口方法

- window.open()：**打开新的窗口**

语法介绍：

```
<script>
    window.open(URL,name,specs,replace);
</script>
```

参数介绍：

<table>
<tr><th>参数</th><th>说明</th></tr>
<tr><td>URL</td><td>可选。打开指定的页面的URL。如果没有指定URL，打开一个新的空白窗口</td></tr>
<tr><td>name</td><td>可选。指定target属性或窗口的名称。支持以下值：
_blank - URL加载到一个新的窗口。这是默认
_parent - URL加载到父框架
_self - URL替换当前页面
_top - URL替换任何可加载的框架集
name - 窗口名称</td></tr>
<tr><td>specs</td><td>可选。一个逗号分隔的项目列表。支持以下值：<table>
<tr><td>channelmode=yes|no|1|0</td><td>是否要在影院模式显示 window。默认是没有的。仅限IE浏览器</td></tr>
<tr><td>directories=yes|no|1|0</td><td>是否添加目录按钮。默认是肯定的。仅限IE浏览器</td></tr>
<tr><td>fullscreen=yes|no|1|0</td><td>浏览器是否显示全屏模式。默认是没有的。在全屏模式下的 window，还必须在影院模式。仅限IE浏览器</td></tr>
<tr><td>height=pixels</td><td>窗口的高度。最小.值为100</td></tr>
<tr><td>left=pixels</td><td>该窗口的左侧位置</td></tr>
<tr><td>location=yes|no|1|0</td><td>是否显示地址字段.默认值是yes</td></tr>
<tr><td>menubar=yes|no|1|0</td><td>是否显示菜单栏.默认值是yes</td></tr>
<tr><td>resizable=yes|no|1|0</td><td>是否可调整窗口大小.默认值是yes</td></tr>
<tr><td>scrollbars=yes|no|1|0</td><td>是否显示滚动条.默认值是yes</td></tr>
<tr><td>status=yes|no|1|0</td><td>是否要添加一个状态栏.默认值是yes</td></tr>
<tr><td>titlebar=yes|no|1|0</td><td>是否显示标题栏.被忽略，除非调用HTML应用程序或一个值得信赖的对话框.默认值是yes</td></tr>
<tr><td>toolbar=yes|no|1|0</td><td>是否显示浏览器工具栏.默认值是yes</td></tr>
<tr><td>top=pixels</td><td>窗口顶部的位置.仅限IE浏览器</td></tr>
<tr><td>width=pixels</td><td>窗口的宽度.最小.值为100</td></tr>
</table></td></tr>
<tr><td>replace</td><td>Optional.Specifies规定了装载到窗口的 URL 是在窗口的浏览历史中创建一个新条目，还是替换浏览历史中的当前条目。支持下面的值：
true - URL 替换浏览历史中的当前条目。
false - URL 在浏览历史中创建新的条目。</td></tr>
</table>

案例演示：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
<button id="btn">打开窗口</button>

<!-- 在这里写JavaScript代码，因为JavaScript是由上到下执行的 -->
<script>
    var btn=document.getElementById("btn");
    btn.onclick = function() {
        myWindow = window.open('https://www.baidu.com',
'_blank','width=300,height=300px');
        //myWindow.document.write(""<p>这是新建窗口</p>"")
    }
</script>
</body>
</html>
```

打开窗口

- window.close()：**关闭当前窗口**

语法介绍：

```
<script>
    window.close();
</script>
```

案例演示：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
<button onclick="openWin()">打开窗口</button>
<button onclick="closeWin()">关闭窗口</button>

<!-- 在这里写JavaScript代码，因为JavaScript是由上到下执行的 -->
<script>
    function openWin() {
        myWindow = window.open('', '', 'width=200,height=100');
        myWindow.document.write("<p>这是新建窗口</p>");
    }

    function closeWin() {
        myWindow.close();
    }
</script>
</body>
</html>
```

- **window.moveTo()：移动当前窗口**

语法介绍：

```
<script>
    window.moveTo(x,y);
</script>
```

案例演示：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
<button onclick="openWin()">打开窗口</button>
<button onclick="moveWin()">移动窗口</button>

<!-- 在这里写JavaScript代码，因为JavaScript是由上到下执行的 -->
<script>
    function openWin() {
        myWindow = window.open('', '', 'width=200,height=100');
        myWindow.document.write("<p>这是新建窗口</p>");
    }

    function moveWin() {
        myWindow.moveTo(300, 300);
        myWindow.focus();
    }
</script>
</body>
</html>
```

- **window.resizeTo()：调整当前窗口**

语法介绍：

```
<script>
    window.resizeTo(width,height);
</script>
```

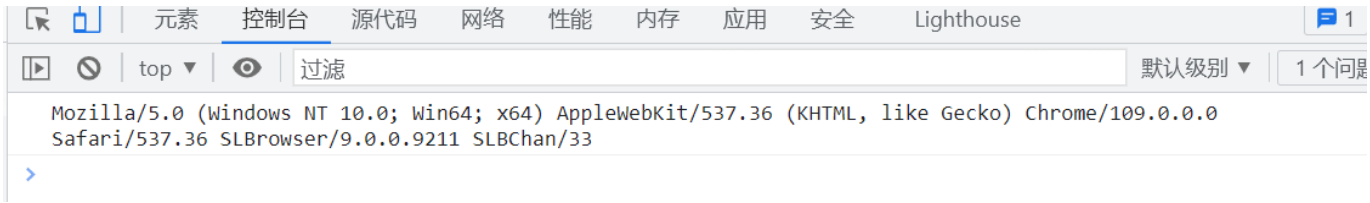
案例演示：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
<button onclick="openWin()">打开窗口</button>
<button onclick="resizeWin()">调整窗口</button>

<!-- 在这里写JavaScript代码，因为JavaScript是由上到下执行的 -->
<script>
    function openWin() {
        myWindow = window.open('', '', 'width=200,height=100');
        myWindow.document.write("<p>这是新建窗口</p>");
    }

    function resizeWin() {
        myWindow.resizeTo(300, 300);
        myWindow.focus();
    }
</script>
</body>
</html>
```

5.3、Navigator对象

Navigator代表的当前浏览器的信息，通过该对象可以来识别不同的浏览器，由于历史原因，Navigator对象中的大部分属性都已经不能帮助我们识别浏览器了，一般我们只会使用userAgent来判断浏览器的信息，userAgent是一个字符串，这个字符串中包含有用来描述浏览器信息的内容，不同的浏览器会有不同的userAgent，如下代码：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
<button onclick="openWin()">打开窗口</button>
<button onclick="resizeWin()">调整窗口</button>

<!-- 在这里写JavaScript代码，因为JavaScript是由上到下执行的 -->
<script>
    var ua = navigator.userAgent;
    console.log(ua);
</script>
</body>
</html>
```



- 谷歌浏览器：

Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36

- 火狐浏览器：

Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:81.0) Gecko/20100101 Firefox/81.0

- IE11浏览器：

Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729; rv:11.0) like Gecko

- IE10浏览器：

Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 10.0; WOW64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729)

- IE9浏览器：

Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 10.0; WOW64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729)

- IE8浏览器：

Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 10.0; WOW64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729)

- IE7浏览器：

Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; WOW64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729)

于是乎，我们就可以实现对浏览器类型的判断：

```
<script>
    var ua = navigator.userAgent;
    if (/firefox/i.test(ua)) {
        alert("你是火狐浏览器");
    } else if (/chrome/i.test(ua)) {
        alert("你是谷歌浏览器");
    } else if (/msie/i.test(ua)) {
        alert("你是IE5-IE10浏览器");
    } else if ("ActiveXObject" in window) {
        alert("你是IE11浏览器");
    }
</script>
```

> 注意：在IE11中已经将微软和IE相关的标识都已经去除了，所以我们基本已经不能通过UserAgent来识别一个浏览器是否是IE了，如果通过UserAgent不能判断，还可以通过一些浏览器中特有的对象，来判断浏览器的信息，比如：ActiveXObject

5.4、Location对象

Location对象中封装了浏览器的地址栏的信息，如果直接打印location，则可以获取到地址栏的信息（当前页面的完整路径）

5.4.1、常用属性

常用属性：

```
<script>
    console.log(location);          //输出location对象
    console.log(location.href);     //输出当前地址的全路径地址
    console.log(location.origin);   //输出当前地址的来源
    console.log(location.protocol); //输出当前地址的协议
    console.log(location.hostname); //输出当前地址的主机名
    console.log(location.host);     //输出当前地址的主机
    console.log(location.port);     //输出当前地址的端口号
    console.log(location.pathname); //输出当前地址的路径部分
    console.log(location.search);   //输出当前地址的?后边的参数部分
</script>
```

修改地址：

```
<script>
    location = "https://www.baidu.com";
    location.href = "https://www.baidu.com";
</script>
```

5.4.2、常用方法

assign()：用来跳转到其它的页面，作用和直接修改location一样

```
<script>
 location.assign("https://www.baidu.com");
</script>
```

reload()：用于重新加载当前页面，作用和刷新按钮一样，如果在方法中传递一个true，作为参数，则会强制清空缓存刷新页面

```
<script>
    location.reload(true);
</script>
```

replace()：可以使用一个新的页面替换当前页面，调用完毕也会跳转页面，它不会生成历史记录，不能使用回退按钮回退

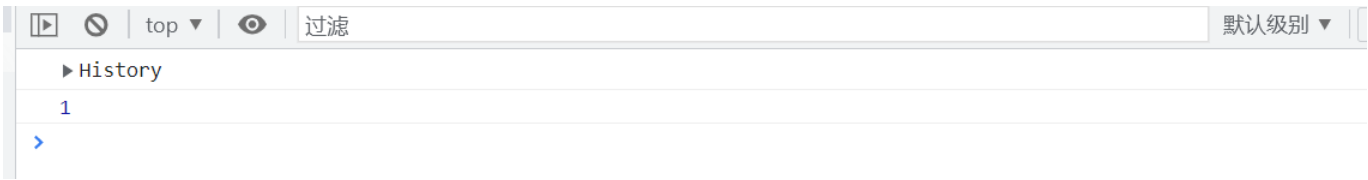
```
<script>
    location.replace("https://www.baidu.com");
</script>
```

5.5、History对象

History对象可以用来操作浏览器向前或向后翻页

5.5.1、常用属性

```
<script>
    console.log(history);           //输出history对象
    console.log(history.length);    //可以获取到当前访问的链接数量
</script>
```



5.5.2、常用方法

back()：可以回退到上一个页面，作用和浏览器的回退按钮一样

```
<script>
    history.back();
</script>
```

forward()：可以跳转到下一个页面，作用和浏览器的前进按钮一样

```
<script>
    history.forward();
</script>
```

go()：可以用来跳转到指定的页面，它需要一个整数作为参数

1：表示向前跳转一个页面，相当于forward() 2：表示向前跳转两个页面 -1：表示向后跳转一个页面，相当于back() -2：表示向后跳转两个页面

```
<script>
    history.go(-2);
</script>
```

5.6、Screen对象

Screen 对象包含有关客户端显示屏幕的信息。

注意：没有应用于 screen 对象的公开标准，不过所有浏览器都支持该对象。

5.6.1、Screen对象描述

每个 Window 对象的 screen 属性都引用一个 Screen 对象。Screen 对象中存放着有关显示浏览器屏幕的信息。JavaScript 程序将利用这些信息来优化它们的输出，以达到用户的显示要求。例如，一个程序可以根据显示器的尺寸选择使用大图像还是使用小图像，它还可以根据显示器的颜色深度选择使用 16 位色还是使用 8 位色的图形。另外，JavaScript 程序还能根据有关屏幕尺寸的信息将新的浏览器窗口定位在屏幕中间。

5.6.2、Screen对象属性

availHeight	返回显示屏幕的高度 (除 Windows 任务栏之外)。
availWidth	返回显示屏幕的宽度 (除 Windows 任务栏之外)。
bufferDepth	设置或返回调色板的比特深度。
colorDepth	返回目标设备或缓冲器上的调色板的比特深度。
deviceXDPI	返回显示屏幕的每英寸水平点数。
deviceYDPI	返回显示屏幕的每英寸垂直点数。
fontSmoothingEnabled	返回用户是否在显示控制面板中启用了字体平滑。
height	返回显示屏幕的高度。
logicalXDPI	返回显示屏幕每英寸的水平方向的常规点数。
logicalYDPI	返回显示屏幕每英寸的垂直方向的常规点数。
pixelDepth	返回显示屏幕的颜色分辨率（比特每像素）。
updateInterval	设置或返回屏幕的刷新率。
width	返回显示器屏幕的宽度。

```html
<!DOCTYPE html>
<html>
<body>
    <h1>Window Screen 对象</h1>
    <h2>Screen 属性</h2>
    <div id="demo"></div>
    <script>
        let text = "总宽度/高度: " + screen.width + "*" + screen.height
        + "<br>" +"可用宽度/高度: " + screen.availWidth + "*" +
        screen.availHeight + "<br>" +"颜色深度: " + screen.colorDepth
        + "<br>" + "颜色分辨率: " + screen.pixelDepth;
        document.getElementById("demo").innerHTML = text;
    </script>

</body>

</html>
```

Window Screen 对象

Screen 属性

总宽度/高度: 1280*720
可用宽度/高度: 1280*720
颜色深度: 24
颜色分辨率: 24