第6章 BOM操作

课程介绍

- BOM介绍
- window对象
- location对象
- navigator对象
- screen对象
- history对象

6.1 BOM介绍

BOM是浏览器对象模型BOM是browser object model的缩写，简称浏览器对象模型。BOM提供了独立于内容而与浏览器窗口进行交互的对象

由于BOM主要用于管理窗口与窗口之间的通讯，因此其核心对象是window。BOM由一系列相关的对象构成，并且每个对象都提供了很多方法与属性。BOM缺乏标准，JavaScript语法的标准化组织是ECMA，DOM的标准化组织是W3C。BOM最初是Netscape浏览器标准的一部分。

6.1.1 BOM结构示意图

BOM学习中我们将学到与浏览器窗口交互的一些对象，例如可以移动，调整浏览器大小的window对象，可以用于导航的location 对象与history对象，可以获取浏览器，操作系统与用户屏幕信息的navigator与screen对象，可以使用document作为访问HTML文档的入口，管理框架的frames对象等。BOM结构示意图如图1所示：

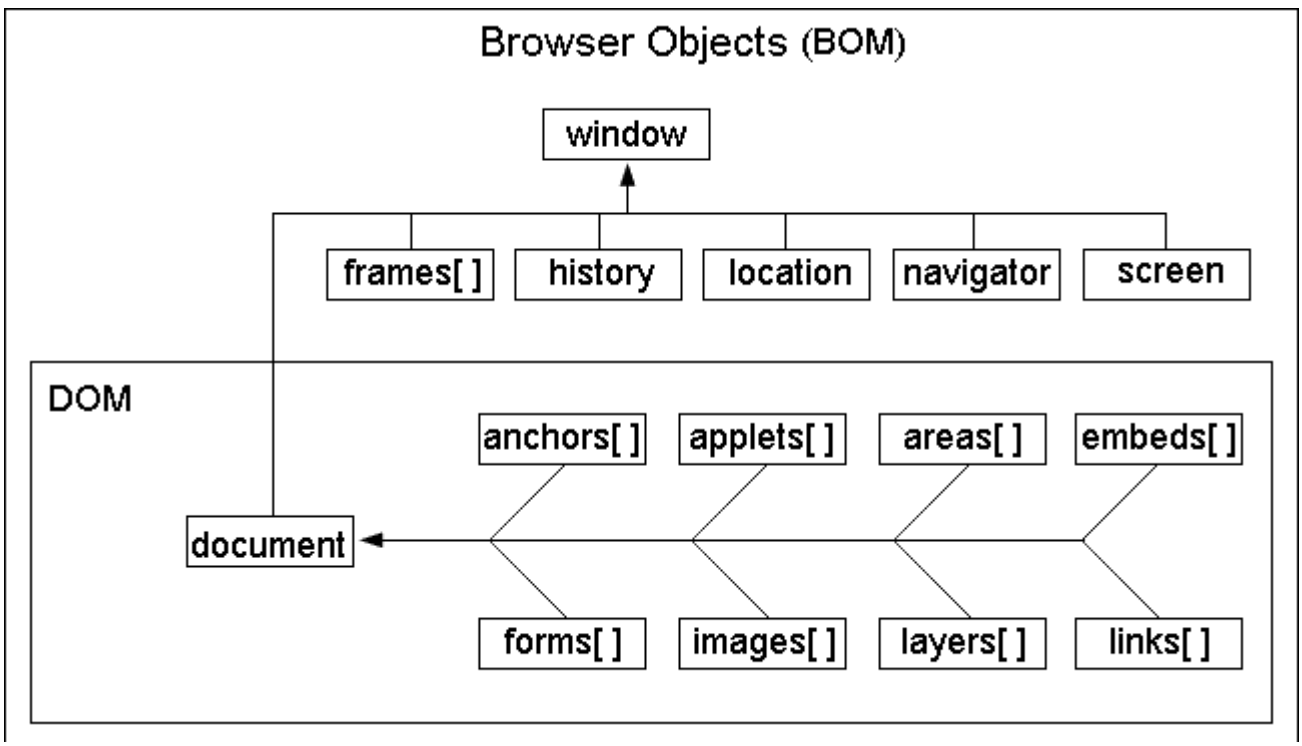

图1

6.1.2 DOM和BOM的区别

DOM通过脚本动态地访问和更新文档的内容、结构以及样式的接口。BOM通过脚本操作浏览器的各个功能组件的接口。

区别：DOM描述了处理网页内容的接口，BOM描述了与浏览器进行交互的接口。document是DOM的核心对象，window则是BOM的核心对象。

6.2 window对象

6.2.1 window对象是什么

浏览器打开一个文档，就创建了一个 window 对象，即 window 对象表示浏览器中打开的窗口。

window 对象是全局对象，可以把窗口的属性作为全局变量来使用。例如，可以只写 document，而不必写 window.document。同样，可以把当前窗口对象的方法当作函数来使用，如只写 alert()，而不必写 Window.alert()。

如果文档包含框架（frame），浏览器会为文档创建一个 window 对象，并为每个框架创建一个额外的 window 对象。

6.2.2 window对象属性

属性	描述	说明
closed	检测窗口是否已被关闭	无
document	对 document 对象的只读引用	无
history	对 history 对象的只读引用	无
length	设置或得到窗口中的框架数量	无
location	用于窗口或框架的 Location 对象	无
name	设置或得到窗口的名称	无
navigator	对 Navigator 对象的只读引用	无
opener	返回对创建此窗口的窗口的引用	无
parent	返回父窗口，常用于 frame 页面调用父页面对象	无
screen	对 Screen 对象的只读引用	无
self	对当前窗口（自己）的引用，一般省略，同 window 属性	无
top	返回最顶层的先辈窗口	无
window	对当前窗口（自己）的引用，一般省略，同 self 属性	无
screenLeft	只读整数，表示窗口的左上角在屏幕上的的 x 坐标	IE、Safari和Opera 支持
screenTop	只读整数，表示窗口的左上角在屏幕上的的 y 坐标	IE、Safari和Opera 支持
screenX	只读整数，表示窗口的左上角在屏幕上的的 x 坐标	Firefox和Safari 支持
screenY	只读整数，表示窗口的左上角在屏幕上的的 y 坐标	Firefox和Safari 支持
defaultStatus	设置或得到窗口状态栏中的默认文本	Firefox 不支持
status	设置窗口状态栏中的文本	Firefox 不支持

6.2.3 window 对象方法

方法	描述	说明
alert()	显示一个警告框	无
blur()	把键盘焦点从顶层窗口移开	部分浏览器无效
clearInterval()	取消由 setInterval() 方法设置的 timeout	无
clearTimeout()	取消由 setTimeout() 方法设置的 timeout	无
close()	关闭浏览器窗口	无
confirm()	显示带有确认按钮和取消按钮的对话框	无
focus()	赋予一个窗口键盘焦点	无
moveBy()	相对窗口的当前坐标把窗口移动指定的像素	无
moveTo()	把窗口的左上角移动到一个指定的坐标	无
open()	打开一个新的浏览器窗口或查找一个已命名的窗口	无
print()	打印当前窗口的内容	无
prompt()	显示可接受用户输入的对话框	无
resizeBy()	按照指定的像素调整窗口的大小	无
resizeTo()	按照指定的宽度和高度调整窗口的大小	无
scrollBy()	按照指定的像素值来滚动内容	无
scrollTo()	把内容滚动到指定的坐标	无
setInterval()	按照指定的毫秒周期来调用函数或计算表达式	无
setTimeout()	在指定的毫秒数后调用函数或计算表达式	无
createPopup()	创建一个 pop-up 窗口	仅 IE 支持

在window对象属性和方法中，某些属性和方法存在浏览器兼容性的问题，而有些属性则已经被浏览器弃用。所以我们这里主要着重讲解：location、navigator、screen和history。

6.3 location对象

Location是什么？

JavaScript Location 对象用于获取或设置当前的 URL 信息。Location 对象是 window 对象的一部分，可通过 window.location 属性对其进行访问。

Location 对象常用于得到 URL 地址中的信息，或者刷新当前页面，页面重定向等，具体可见下面列出的各属性和方法。

6.3.1 Location 对象属性

属性	描述
location.hash	设置或取得 URL 中的锚
location.host	设置或取得 URL 中主机（包括端口号）
location.hostname	设置或取得 URL 中的主机名
location.href	设置或取得完整 URL（页面重定向应用）
location.pathname	设置或取得 URL 中的路径
location.port	设置或取得 URL 中的端口号
location.protocol	设置或取得 URL 使用的协议
location.search	设置或取得 URL 中的查询字串（一般是？符号后面的内容）

我们以打开“http://127.0.0.1:8001/2.08/01.location.html?__hbt=1532484540442#1“这个网址为例：

实例：

```
console.log(window.location.hash);//  #1
console.log(window.location.host);//  127.0.0.1:8001
console.log(window.location.hostname);//  127.0.0.1
console.log(window.location.href);//  http://127.0.0.1:8001/2.08/01.location.html?
__hbt=1532484540442#1
console.log(window.location.pathname);//  /2.08/01.location.html
console.log(window.location.port);//  8001
console.log(window.location.protocol);//  http:
console.log(window.location.search);//   ?__hbt=1532484395449
```

6.3.2 Location 对象方法

- Location对象有如下 3 个方法：
- location.assign()：加载新页面文档
- location.reload()：重新加载（刷新）当前页面
- location.replace()：用新的文档替代当前文档

6.3.2.1 JavaScript location.assign() 方法

Location 对象的 assign() 方法用于加载一个新的文档，语法如下：

```
location.assign(URL)
```

实例：

```
<body>
    <button onclick="setAssign()">加载新文档</button>
</body>

<script type="text/javascript">
    function setAssign(){
        window.location.assign("http://www.hqjy.com");
    }
</script>
```

运行该例子，点击 加载新文档 按钮，触发 assign() 函数，浏览器将访问恒企首页。

说明：实际上 location.assign() 方法的效果与 location.href 是一样的。

6.3.2.2 JavaScript location.reload() 方法

Location 对象的 reload() 方法用于重新加载当前文档（页面），语法如下：

```
location.reload( false|true )
```

如果该方法参数为 false 或者省略参数，它就会用 HTTP 头 If-Modified-Since 来检测服务器上的文档是否已改变。如果文档已改变，location.reload() 会再次下载该文档。如果文档未改变，则该方法将从缓存中装载文档。

如果要强制刷新当前页面，请将参数设置为 true。

实例：

```
<body>
    <button onclick="setReload()">刷新页面</button>
</body>

<script type="text/javascript">
    function setReload(){
        window.location.reload();
    }
</script>
```

6.3.2.3 JavaScript location.replace() 方法

Location 对象的 replace() 方法用于重新加载当前文档（页面），语法如下：

```
location.replace( new_URL )
```

实例：

```
<body>
    <button onclick="setReplace()">加载新页面</button>
</body>

<scrip type="text/javascript">
    function setReplace(){
        window.location.replace( "http://www.hqjy.com" );
    }
</script>
```

运行该例子，点击 加载新页面 的按钮，触发 setReplace() 函数，浏览器将加载恒企首页以替换当前页面。

6.3.2.4 location.replace() 与 location.reload() 的区别

location.reload() 方法用于刷新当前页面，如果有 POST 数据提交，则会重新提交数据；location. replace() 则将新的页面以替换当前页面，它是从服务器端重新获取新的页面，不会读取客户端缓存且新的 URL 将覆盖 History 对象中的当前纪录（不可通过后退按钮返回原先的页面）。

如果想要刷新当前的页面，又避免 POST 数据提交，可以使用：

```
window.location.replace( location.href );
```

6.4 navigator对象

Navigator 对象是什么？

JavaScript Navigator 对象包含了有关浏览器的相关信息。

提示：Navigator 对象虽然没有明确的相关标准，但所有浏览器都支持该对象。

6.4.1 Navigator 对象属性

Navigator 对象属性为只读属性，下面表中列出了各属性及以 IE8 为例得到的值。

属性	描述	IE8 的结果
navigator.appCodeName	取得浏览器的代码名，注1	Mozilla
navigator.appName	取得浏览器的名称，注2	Microsoft Internet Explorer
navigator.appVersion	取得浏览器的平台和版本信息	4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30)
navigator.browserLanguage	取得浏览器使用的语言（仅 IE 支持，注3）	zh-cn
navigator.cookieEnabled	检测浏览器是否启用 cookie 支持，返回布尔值，true 表示支持	TRUE
navigator.cpuClass	取得浏览器所在系统的 CPU 等级（仅 IE 支持，注4）	x86
navigator.onLine	检测系统是否处于脱机模式，返回布尔值，false 表示脱机（仅 IE 支持，注5）	TRUE
navigator.platform	得到浏览器所在的操作系统平台	Win32
navigator.systemLanguage	得到浏览器所在操作系统使用的语言（仅 IE 支持，注6）	zh-cn
navigator.userAgent	得到浏览器用于 HTTP 请求的用户代理头的值	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30)
navigator.userLanguage	得到浏览器所在操作系统的自然语言设置（仅 IE 支持，注7）	zh-cn

注意说明

1. navigator.appCodeName：IE/Firefox/Chrome 系列浏览器中，它的值都是 "Mozilla"。
2. navigator.appName：Firefox/Chrome 均为 Netscape。
3. navigator.browserLanguage：Firefox/Chrome 返回 "undefined"。
4. navigator.cpuClass：Firefox/Chrome 返回 "undefined"。
5. navigator.onLine：Firefox/Chrome 返回 "undefined"。
6. navigator.systemLanguage：Firefox/Chrome 返回 "undefined"。
7. navigator.userLanguagee：Firefox/Chrome 返回 "undefined"。

利用 Navigator 对象提供的浏览器信息，可以方便的得到访问用户的浏览器名称及版本。

实例：

```
console.log(navigator.appCodeName);//Mozilla
console.log(navigator.appName);    //Netscape
console.log(navigator.appVersion); //5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/68.0.3440.75 Safari/537.36
console.log(navigator.browserLanguage);       //undefined
console.log(navigator.cookieEnabled);     //true
console.log(navigator.cpuClass);       //undefined
console.log(navigator.onLine);     //true
console.log(navigator.platform);       //Win32
console.log(navigator.systemLanguage);    //undefined
console.log(navigator.userAgent);      //Mozilla/5.0 (Windows NT 6.1; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.75 Safari/537.36
console.log(navigator.userLanguage);      //undefined
```

这是以谷歌浏览器为例运行的得出的结果。

6.4.2 Navigator 对象方法

Navigator 对象有如下 2 个方法：

1. navigator.javaEnabled()：检测浏览器是否启用了 java 支持，返回布尔值，true 表示支持。
2. navigator.taintEnabled()：检测浏览器是否启用数据污点(data tainting)，返回布尔值，true 表示启用。

注意：目前只有 Internet Explorer 和 Opera 浏览器支持 taintEnabled() 方法。

注释：JavaScript 数据污点(data tainting)：Navigator 3.0 中的 JavaScript 有一个叫做数据污点的特性，其中保留了安全限制，但是提供了对于页面中指定部件的安全访问手段。当允许数据污点的时候，一个窗口中的 JavaScript 可以观看另一个窗口的属性，而不管另外的窗口是从哪个服务器上装入的；而当禁止数据污点的时候，脚本不能访问其它服务器窗口的任何属性。

实例：

```
console.log(navigator.javaEnabled()); //true **说明**已开启java支持
console.log(navigator.taintEnabled()); //false  **说明**未启用数据污点
```

这是以IE浏览器为例运行的得出的结果。

6.5 Screen 对象

6.5.1 Screen 对象是什么？

JavaScript Screen 对象包含了有关客户端显示屏幕的相关信息。JavaScript 程序可以利用这些信息来优化显示输出，以达到更精准的显示要求。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等。

提示：Screen 对象虽然没有明确的相关标准，但所有浏览器都支持该对象。

6.5.2 Screen 对象属性

Screen 对象的属性并不是所有浏览器都支持，具体见下表说明。

属性	描述	说明
Screen.availHeight	取得显示屏幕的高度(Windows 任务栏除外)，像素单位	都支持
Screen.availWidth	取得显示屏幕的宽度(Windows 任务栏除外)，像素单位	都支持
Screen.colorDepth	目标设备或缓冲器上的调色板的比特深度	都支持
Screen.height	取得显示屏幕的高度	都支持
Screen.width	取得显示器屏幕的宽度	都支持
Screen.bufferDepth	设置或得到调色板的比特深度	仅 IE 支持
Screen.deviceXDPI	得到显示屏幕的每英寸水平点数	仅 IE 支持
Screen.deviceYDPI	得到显示屏幕的每英寸垂直点数	仅 IE 支持
Screen.fontSmoothingEnabled	得到用户是否在显示控制面板中启用了字体平滑	仅 IE 支持
Screen.logicalXDPI	得到显示屏幕每英寸的水平方向的常规点数	仅 IE 支持
Screen.logicalYDPI	得到显示屏幕每英寸的垂直方向的常规点数	仅 IE 支持
Screen.updateInterval	设置或得到屏幕的刷新率	仅 IE 支持
Screen.pixelDepth	得到显示屏幕的颜色分辨率（比特每像素）	仅 IE 不支持

说明：

Screen.availHeight 和 Screen.availWidth 属性值不包括 Windows 操作系统下任务栏占用的长度。

在生产环境中，尽量不要使用有浏览器不支持的属性，否则可能会导致程序运行错误。

实例：

检测用户客户端屏幕信息的实例

```
document.write("<p>屏幕分辨率： ")
document.write(screen.width + "*" + screen.height + "</p>")
document.write("<p>屏幕可显示面积：")
document.write(screen.availWidth + "*" + screen.availHeight + "</p>")
document.write("<p>颜色深度： ")
document.write(screen.colorDepth + "</p>")
document.write("<p>缓冲深度：")
document.write(screen.bufferDepth + "</p>")
document.write("<p>每英寸水平点数：")
document.write(screen.deviceXDPI + "</p>")
document.write("<p>每英寸垂直点数：")
document.write(screen.deviceYDPI + "</p>")
document.write("<p>水平方向的常规点数：")
document.write(screen.logicalXDPI + "</p>")
document.write("<p>垂直方向的常规点数：")
document.write(screen.logicalYDPI + "</p>")
document.write("<p>是否启用了字体平滑：")
document.write(screen.fontSmoothingEnabled + "</p>")
```

```
document.write("<p>屏幕的颜色分辨率（比特/像素）：")
document.write(screen.pixelDepth + "</p>")
document.write("<p>屏幕刷新率：")
document.write(screen.updateInterval + "</p>")
```

在谷歌浏览器下运行：

屏幕分辨率： 1920*1080

屏幕可显示面积：1920*1050

颜色深度： 24

缓冲深度：undefined

每英寸水平点数：undefined

每英寸垂直点数：undefined

水平方向的常规点数：undefined

垂直方向的常规点数：undefined

是否启用了字体平滑：undefined

屏幕的颜色分辨率（比特/像素）：24

屏幕刷新率：undefined

在IE浏览器下运行：

屏幕分辨率： 1920*1080

屏幕可显示面积：1920*1050

颜色深度： 24

缓冲深度：0

每英寸水平点数：96

每英寸垂直点数：96

水平方向的常规点数：96

垂直方向的常规点数：96

是否启用了字体平滑：true

屏幕的颜色分辨率（比特/像素）：24

屏幕刷新率：undefined

可见，不同的浏览器，显示出来的数据可能不尽相同，比较通用的是屏幕尺寸，这也是实际上最常用的。

6.6 History 对象

6.6.1 什么是History 对象？

JavaScript History 对象用于记录操作浏览器的访问历史。History 对象是 window 对象的一部分，可通过 window.history 属性对其进行访问。

提示：History 对象的有效作用范围都是指当前窗口。

6.6.2 History 对象 length 属性

History 对象有唯一的一个 length 属性，用于得到浏览器访问历史记录中的 URL 数量。

实例：

```
document.write(history.length);
```

运行该例子，输出：0。

说明：

该例子输出的结果取决于当前页面的浏览记录，如果是新窗口打开该例子，IE 浏览器会输出 0（即从 0 开始计算），而 Firefox、Chrome 等浏览器则会输出 1。

6.6.3 History 对象方法

History 对象有如下 3 个方法：

1. history.back()：返回前一个浏览页面（如果存在）
2. history.forward()：前往下一个浏览页面（如果存在）
3. history.go()：前往 history 列表中的某个指定页面（如果存在）

6.6.3.1 back() 方法

back() 方法用于返回前一个浏览页面（如果存在），其效果相当于点击浏览器的返回按钮。以下是常用的返回上一页。

实例：

```
<body>
    <a href="javascript:window.history.back()" />返回上一页</a>
</body>
```

6.6.3.2 forward() 方法

forward () 方法用于前往下一个浏览页面（如果存在），其效果相当于点击浏览器的前进按钮。例子：

实例：

```
<body>
    <a href="javascript:window.history.forward()" />前往下一页</a>
</body>
```

6.6.3.3 go() 方法

go() 方法用于前往 history 列表中的某个指定页面（如果存在），语法如下：

```
history.go( number|URL )
```

参数说明:

参数	说明
number	要访问的页面相对当前页面的位置，负整数表示往后返回，正整数表示往前进。
URL	要访问的 URL，或 URL 的子串。

下面的例子效果与 history.back() 相同。

实例：

```
<body>
    <a href="javascript:window.history.go(-1)" />返回上一页</a>
</body>
```

6.7 课程总结

1. BOM介绍
   1 ）BOM结构示意图
   2 ）DOM和BOM的区别
2. window对象
   1 ）window对象是什么
   2 ）window对象属性
   3 ）window 对象方法
3. location对象
   1 ）Location 对象属性
   2 ）Location 对象方法
   3 ）JavaScript location.assign() 方法
   4 ）JavaScript location.reload() 方法
   5 ）JavaScript location.replace() 方法
   6 ）location.replace() 与 location.reload() 的区别
4. navigator对象
   1 ）Navigator 对象属性

2 ）Navigator 对象方法

5. Screen 对象

1 ）Screen 对象是什么

2 ）Screen 对象属性

6. History 对象

1 ）History 对象

2 ）History 对象 length 属性

3 ）History 对象方法

4 ）back() 方法

5 ）forward() 方法

6 ）go() 方法

6.8 实训

1.写一个小球在浏览器中无规律的弹跳，弹跳不会超出边框，如图所示：

2.写出一个输入框和按钮，填写网址时，点击按钮可实现跳转网页。