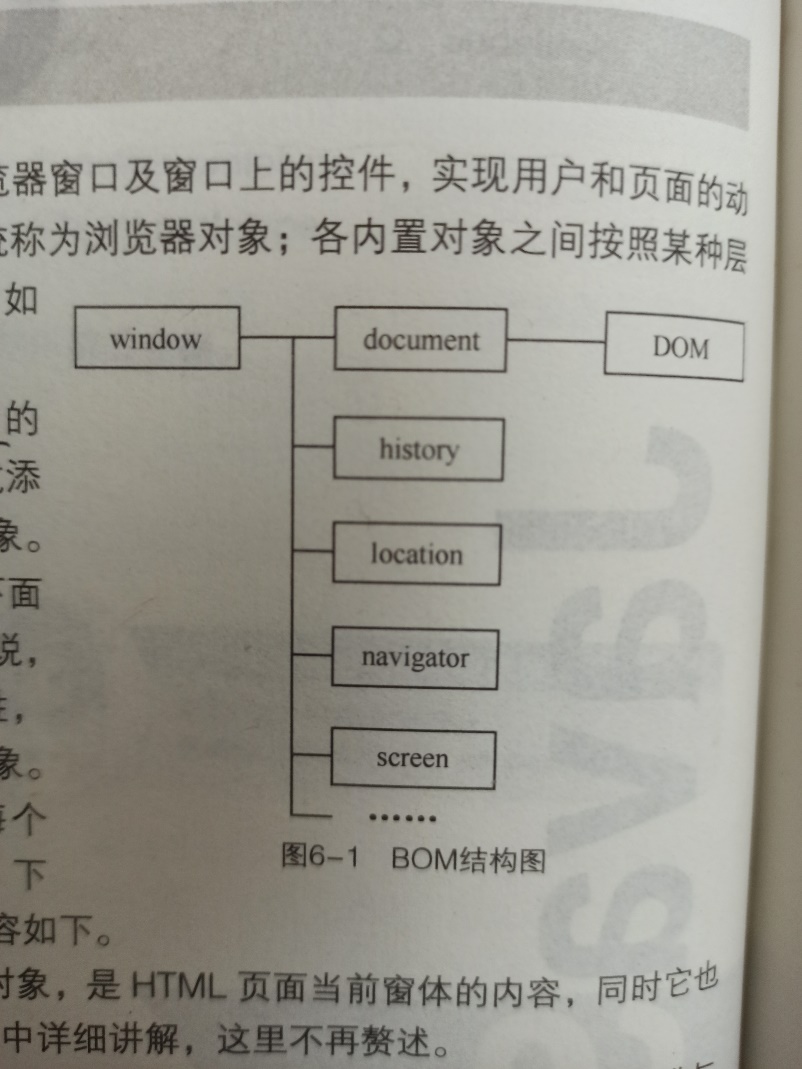
# 什么是BOM对象

## JavaScript组成

JavaScript是由ECMAScript、BOM、和DOM组成的。其中ECMAScript就是前面学习的JavaScript基本语法、数组、函数和对象。BOM（Brower Object Model）指的是浏览器对象模型，DOM（Document Object Model）文档对象模型。

## 认识BOM

在实际开发中，JavaScript经常需要操作浏览器窗口及窗口上的控件（如返回上一页、返回下一页等）实现用户和页面的动态交互。为此，浏览器提供了一系列内置对象，统称为BOM浏览器对象模型。BOM的结构下图所示：



从图中可以看出，window对象时BOM的顶层（核心）对象，其他对象都是以属性的方式添加到window对象下，也可以称为window的子对象。例如document对象（DOM）是window对象下面的一个属性，同时它也是一个对象。

BOM为了访问和和操作浏览器各组件，每个window子对象中都提供了一系列的属性和方法。下面将对window子对象的功能进行介绍：

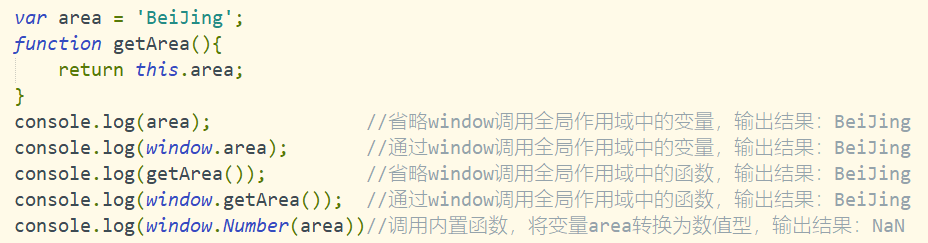
1. document（文档对象）：也称为DOM对象，是HTML页面当前窗体的内容，同时它也是JavaScript重要的组成部分之一，将在下一章节详细讲解。
2. history（历史对象）：主要用于记录浏览器的访问历史记录，也就是浏览器网页的前进和后退功能。
3. location（地址栏对象）：用于获取当前浏览器中URL地址栏内的相关数据。
4. navigator（浏览器对象）：用于获取浏览器的相关数据，如浏览器的名称、版本等，也称为浏览器的嗅探器。
5. screen（屏幕对象）：可获取与屏幕相关的数据，如屏幕的分辨率、坐标信息等。

值得一提的是，BOM没有一个明确的规范，所以浏览器提供商会按照各自的想法随意去扩展BOM。而各浏览器间共有的对象就成为了事实上的标准。不过在利用BOM实现具体功能时，要根据实际的开发情况考虑浏览器之间的兼容问题，否则会出现不可预料的情况。

# window对象

## 全局作用域

1. window对象时BOM中所有对象的核心，同时也是BOM中所有对象的父对象，其他对象都是作为window对象属性存在的。
2. 定义在全局变量作用域中的变量、函数以及JavaScript中的内置函数都可以被window对象调用。如下：



定义在全局作用域中的getArea( )函数，函数体内的this关键字指向window对象；同时，对于window对象的属性和方法在调用时可以省略window，直接访问其属性和方法即可。

1. 值得一提的是，在JavaScript中直接使用一个未声明的变量会报语法错误，但是使用“window.变量名”的方式则不会报错，而是获得一个undefined结果。除此之外，delete关键字仅能删除window对象自身的属性，对于定义在全局作用域下的变量不起作用。

## 弹出对话框和窗口

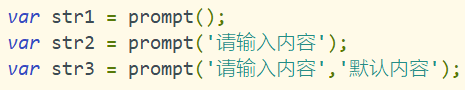
window对象中除了前面提过的alert( )和prompt( )方法外，还提供了很多弹出对话框和窗口的方法，以及相关的操作属性。如下表所示：

|  |  |  |
| --- | --- | --- |
| 分类 | 名称 | 说明 |
| 属性 | closed | 返回一个布尔值，该值声明了该窗口是否以及关闭 |
| name | 设置或返回存放窗口名称的一个字符串 |
| opener | 返回对该创建该窗口的window对象的引用 |
| parent | 返回当前窗口的父窗口 |
| self | 对当前窗口的引用，等价于window对象 |
| top | 返回最顶层的父窗口 |
| 方法 | alert( ) | 显示带有一段消息和一个确认按钮的警告框 |
| confirm( ) | 显示带有一段信息以及确认按钮和取消按钮的对话框 |
| prompt( ) | 显示可提示用户输入信息的对话框 |
| open( ) | 打开一个新的浏览器窗口或者查找一个已命名的窗口 |
| close( ) | 关闭浏览器窗口 |
| focus( ) | 把键盘焦点给予一个窗口 |
| print( ) | 打印当前窗口的内容 |
| scrollBy( ) | 按照指定的像素值来滚动内容 |
| scrollTo( ) | 把内容滚动到指定的坐标 |

以上方法和属性在常见的浏览器中全部支持。通过几个示例更好的理解和使用这些属性和方法。

1. **输入对话框**

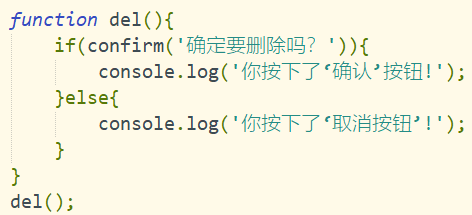
window对象提供的prompt( )方法用于生成用户输入的对话框，具体示例如下：



prompt( )方法的参数都是可选的，第一个参数用于设置提示文字，提示用户要输入什么；第二个参数用于设置输入框中的默认值。当用户点击确定后，如果输入框中有内容，则返回，如果没有，则返回一个空白字符串。

1. **确认对话框**

在Web开发中，用户在进行某些操作（如删除数据）时，通常会弹出一个确认对话框，提示用户是否确认或取消该操作。具体示例如下：



当用户点击“确定”按钮返回true，点击“取消”按钮则返回false。

1. **打开与关闭窗口**

open( )方法用于打开一个新的浏览器窗口，或查找一个已命名的窗口。语法如下：

**open( URL , name , space , replace )**

·参数URL表示要打开指定页面的URL地址，如果没有指定，则打开一个新的空白

窗口；

·参数name表示指定target属性或窗口的名称。可选值如下：

|  |  |
| --- | --- |
| 可选值 | 含义 |
| \_blank | URL加载到一个新的窗口，也是默认值 |
| \_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 | 指定窗口名称 |

·参数specs用于设置浏览器窗口的特征（如大小、位置、滚动条等），多个特征

使用逗号分隔。可选参数和可选值如下：

|  |  |  |
| --- | --- | --- |
| 可选参数 | 值 | 说明 |
| height | Number | 窗口的高度，最小值为100 |
| width | Number | 窗口的宽度，最小值为100 |
| left | Number | 该窗口的左侧位置 |
| location | yes|no|1|0 | 是否显示地址字段，默认值是yes |
| menubar | yes|no|1|0 | 是否显示菜单栏，默认值为yes |
| resizable | yes|no|1|0 | 是否可调整窗口大小，默认值是yes |
| scrollbars | yes|no|1|0 | 是否显示滚动条，默认值是yes |
| status | yes|no|1|0 | 是否要添加一个状态栏，默认值是yes |
| titlebar | yes|no|1|0 | 是否显示标题栏被忽略，除非调用HTML应用程序或一个值得依赖的对话框，默认值是yes |
| toolbar | yes|no|1|0 | 是否显示浏览器工具栏，默认值yes |

调用open( )方法后，会返回一个window对象，表示新打开的窗口，与open( )相反的方法是close( )方法，用于关闭浏览器窗口，调用该方法的对象就是需要关闭的窗口对象。具体示例如下：



## 窗口的位置和大小

BOM中用来获取或更爱window窗口位置、窗口高度与宽度、文档区域高度与宽度的先关属性和方法有很多。如下表所示：

|  |  |  |
| --- | --- | --- |
| 分类 | 名称 | 说明 |
| 属性 | screenLeft | 返回相对于屏幕窗口的x坐标（FireFox不支持） |
| screenTop | 返回相对于屏幕窗口的y坐标（FireFox不支持） |
| screenX | 返回相对于屏幕窗口的x坐标（IE8不支持） |
| screenY | 返回相对于屏幕窗口的y坐标（IE8不支持） |
| innerHeight | 返回窗口的文档显示区的高度（IE8不支持） |
| innerWidth | 返回窗口的文档显示区的宽度（IE8不支持） |
| outerHeight | 返回窗口的外部高度，包含工具条和滚动条（IE8不支持） |
| outerWidth | 返回窗口的外部宽度，包含工具条和滚动条（IE8不支持） |
| 方法 | moveBy( ) | 将窗口移动到相对的位置 |
| moveTO( ) | 将窗口移动到指定的位置 |
| resizeBy( ) | 将窗口大小调整到相对的宽度和高度 |
| resizeTo( ) | 将窗口大小调整到指定的宽度和高度 |

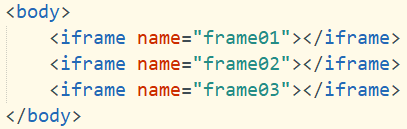
目前只有window.open( )方法打开的窗口和选项卡（Tab），FireFox和Chrome才支持窗口位置和大小的调整。具体示例如下：



## 框架操作

window对象提供的frames属性可通过集合的方式，获取HTML页面中所有的框架，length属性可以获取当前窗口中frames的数量。

假设HTML中有如下框架：



在JavaScript中可使用3种方式获取窗口对应给定对象<frame>或<iframe>给定的内容，例如重新设置框架中显示的内容。具体代码如下：



上述代码中，在获取具体的窗口时，既可以通过name值，采用数组或访问对象属性的方式获取；又可以利用下标的方式访问，默认从0开始。除此之外，还可以利用parent获取当前window对象所在的父窗口。如下：



## 定时器

JavaScript中可通过window对象提供的方法实现在指定时间后执行特定操作，也可以让程序代码每隔一段时间执行一次，实现间歇操作。具体方法如下表所示：

|  |  |
| --- | --- |
| 方法 | 说明 |
| setTimeout( ) | 在指定的毫秒数后调用函数或执行一段代码（执行一次） |
| clearTimeout( ) | 取消由setTimeout( )方法设置的定时器 |
| setInterval( ) | 按照指定的周期（以毫秒计）来调用函数或执行一段代码（重复执行） |
| clearInterval( ) | 取消由setInterval( )设置的定时器 |

setTimout( )和setInterval( )方法虽然都可以在一个固定时间段内执行JavaScript程序，不同的是，setTimeout( )方法只执行一次代码，而setInterval( )方法会在指定的时间后，自动重复执行代码。

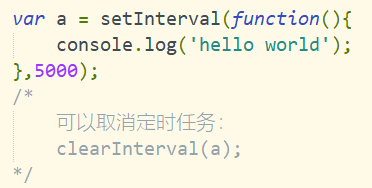
1. **setTimeout( )**



表示隔5秒后打印一次hello world。

1. setTimeout( )方法的第一个参数表示要执行的代码，第二个参数表示定时间隔，即多少毫秒后开始执行。
2. setTimeout( )方法设置的定时任务执行一次后即停止操作，不会再执行第二次。所以上述代码在5秒后会输出一次“hello world”，之后不再执行。
3. 可以将setTimeout( )方法的返回值（定时器ID）作为clearTimeout( )方法的参数，取消该定时器的执行。

**2、setInverval( )**



表示每隔5秒后打印一次hello world

1. setInterval( )方法的第一个参数表示要执行的代码，第二个参数表示执行时间间隔。
2. setInterval( )方法设置的定时任务会每隔指定的时间执行一次。所以上述代码每隔5秒执行一次，一直重复执行下去。除非窗口关闭或取消定时器。
3. 可以将setInterval ( )方法的返回值（定时器ID）作为clearInterval ( )方法的参数，取消该定时器的执行。

# location对象

## 更改URL

BOM中location对象提供的方法，可以更改当前用户在浏览器中访问的URL，实现新文档的载入、重载以及替换等功能。

1. **认识URL**
2. 在Internal上访问每一个网页文件，都有一个访问标识符，用于唯一标识它的访问位置，以便浏览器可以访问到，这个访问标识符称为URL（Uniform Resource Locator，统一资源定位符）。
3. 在URL中，包含了网络协议、服务器的主机号、端口号、URI（Uniform Resource Identifier，统一资源标识符）、参数以及锚点，具体示例如下：



1. **更改URL**

location对象提供的用于改变URL地址的方法，所有主流的浏览器都支持。

|  |  |
| --- | --- |
| 方法 | 说明 |
| assign( ) | 载入一个新文档 |
| reload( ) | 重新载入当前文档 |
| replace( ) | 用新的文档替换当前文档 |

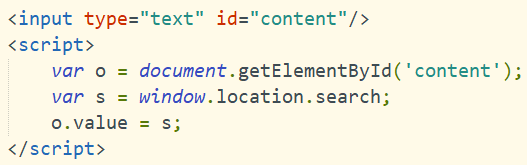
这些方法应该都有参数，其中reload( )方法的唯一参数是一个布尔值，将其设置为true时，它会绕过缓存，从服务器上重新下载该翁当，类似于浏览器中的刷新页面按钮。

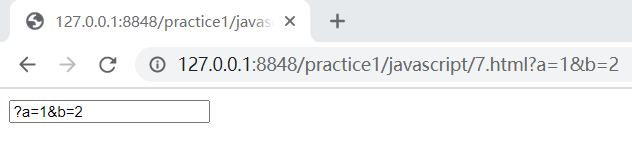
使用示例如下：



## 获取URL参数

Web开发中，经常通过URL地址传递的参数执行指定的操作，如商品的搜索、排序等。此时，可以利用location对象提供的search属性获取URL地址中的参数。示例如下：





可以看出，文本框中获取的参数与地址栏中的参数一致。

此外，location对象还提供了其他属性，用于获取和设置对应的URL地址的组成部分，如服务器主机号、端口号、URL协议以及完整的URL地址等。如下表：

|  |  |
| --- | --- |
| 属性 | 说明 |
| hash | 返回一个URL的锚部分 |
| host | 返回一个URL的主机名和端口 |
| hostname | 返回URL的主机号 |
| href | 返回完整的URL |
| pathname | 返回URL的路径名 |
| port | 返回一个URL服务器使用的端口号 |
| protocol | 返回一个URL协议 |

通过修改这些属性会使浏览器根据修改后的URL重新加载页面。

# history对象

## 历史记录跳转

BOM中提供的history对象，可以对用户在浏览中访问的URL历史记录进行操作。出于安全方面的考虑，history对象不能直接获取用户浏览过的URL，但可以控制浏览器实现“后退”和“前进”的功能。

|  |  |  |
| --- | --- | --- |
| 分类 | 名称 | 说明 |
| 属性 | length | 返回历史列表中的网站数 |
| 方法 | back( ) | 加载历史列表中的前一个URL |
| forward( ) | 加载历史列表中的下一个URL |
| go( ) | 加载历史列表中的某个具体页面 |

上述方法中，go( )方法可根据参数的不同设置，完成历史记录的任意跳转。当参数值是一个负整数时，表示“后退”指定的页面；当参数值是一个正整数时，表示“前进”指定的页数。

## 无刷新更改URL地址

HTML5为history对象引入了history.pushState( )和history.replaceState( )方法，用来在浏览历史中添加和修改记录，实现无刷新更改URL地址。语法如下：

**pushState ( state , title [ , url] )**

**replaceState( state , title [ , url ] )**

·参数state表示一个与指定网址相关的状态对象，popstate事件触发时，该对象会传

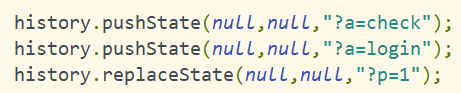
入回调函数。如果不需要这个对象，此处可以填null或空字符串。

·参数title表示新页面的标题，但是所有浏览器目前都忽略这个值，因此这里可以填

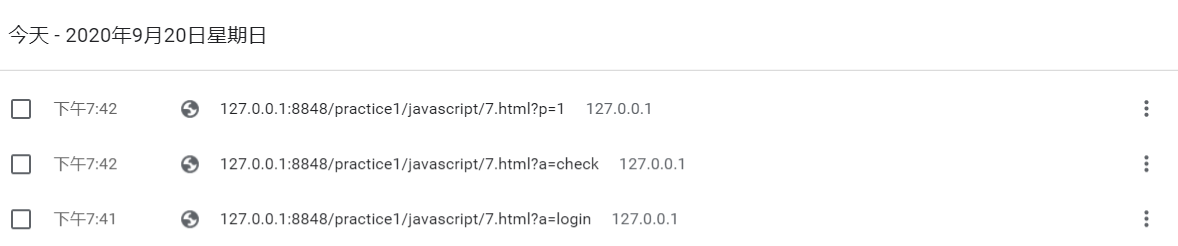
null或空字符串。

·参数url表示新的网址，并且必须与当前页面处在同一域中。

方法执行后，浏览器的地址栏将显示最后添加或修改的网址。具体示例如下：



执行以上JavaScript程序后，历史记录中将添加相对应的记录：



pushState( )方法会改变浏览器的历史列表中记录的数量，而replaceState( )方法仅用于修改历史记录，历史记录列表的数量不变。

# navigator对象

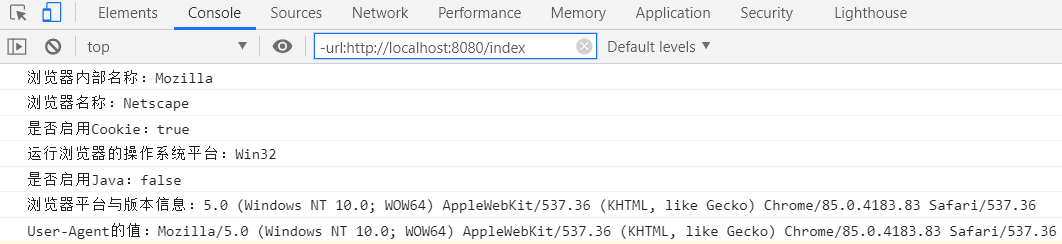
navigator对象提供了有关浏览器的信息，但是每个浏览器中的navigator对象中都有一套自己的属性。下面了列举主流浏览器中都存在的属性和方法：

|  |  |  |
| --- | --- | --- |
| 分类 | 名称 | 说明 |
| 属性 | appCodeName | 返回浏览器的内部名称 |
| appName | 返回浏览器的名称 |
| appVersion | 返回浏览器的平台和版本信息 |
| cookieEnabled | 返回指明浏览器中是否启用cookie的布尔值 |
| platform | 返回运行浏览器的操作系统平台 |
| userAgent | 返回由客户端发送服务器的User-Agent头部的值 |
| 方法 | javaEnabled( ) | 指定是否在浏览器中启用Java |

具体示例如下：



查看控制台输出：



# screen对象

screen对象用于返回当前渲染窗口中与屏幕相关的属性信息，如屏幕的高度和宽度等。需要注意的是，每个浏览器中的screen对象都包含不同的属性。下面展示了主流浏览器中都支持的screen属性：

|  |  |
| --- | --- |
| 属性 | 说明 |
| height | 返回整个屏幕的高 |
| width | 返回整个屏幕的宽 |
| availHeight | 返回浏览器窗口在屏幕可占用的垂直空间 |
| availWidth | 返回浏览器窗口在屏幕上可占用的水平空间 |
| colorDepth | 返回屏幕的颜色深度 |
| pixelDepth | 返回屏幕的位深色/色彩深度 |

具体示例如下：

