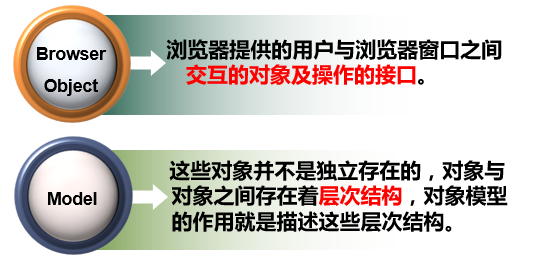
# BOM综述

BOM（Browser Object Model）



(1) BOM是Browser Object Model的缩写，简称浏览器对象模型

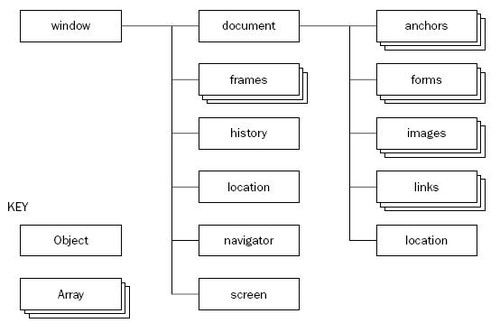
(2)BOM提供了**独立于内容而与浏览器窗口进行交互的对象**

(3)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**

(4)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

(5)这些对象并不是独立存在的，对象与对象之间存在着层次结构，对象模型的作用就是描述这些层次结构。

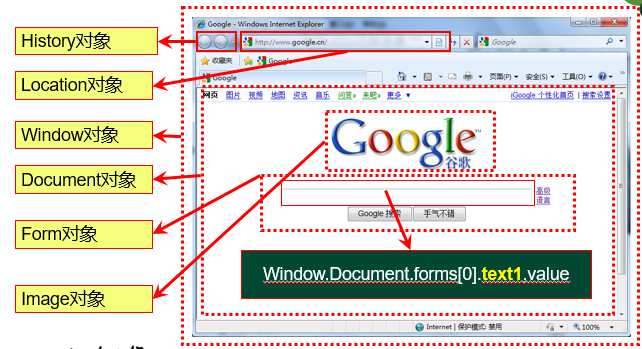
(6)BOM提供了一些访问窗口对象的一些方法，我们可以用它来移动窗口位置，改变窗口大小，打开新窗口和关闭窗口，弹出对话框，进行导航以及获取客户的一些信息如：浏览器品牌版本，屏幕分辨率。



（1）BOM体系结构，window,document,frames,histroy,location,navigator,screen

（2）Window对象是整个BOM的核心，其他BOM对象都以某种方式接回到window对象。

（3）顶层对象就是代表浏览器的Window对象，所有其他对象都是Window对象下的子对象。



## BOM中对象的引用

* BOM对象与对象之间存在层次关系，引用对象**要遵循一定的引用规则**。基本引用规则：
  + 顶层的Window对象直接使用Window来引用

Window.open() Window.close()

* + **Window对象下的子对象通过Window.** 来引用

Window.document.write()

* + 由于Window对象是全局对象，所以该对象下的子对象和方法都可以**直接使用**。
* **通过数组下标引用：**

BOM中有很多数组，如Forms数组、Images数组等，这些数组中存放的是相应的对象，可以通过数组下标的方式来引用该对象。

document.forms[0].elements[0] //数组中第一个表单里的第一个控件

* 通过名称引用：
  + 允许使用标签的**name属性值来引用对象**。
  + 由于只有代表文档的Document对象中才可能存在标签，所以在使用标签的name属性值来引用对象时，**可以省略document.** 语句。

<form name=“myForm”>

用户名：<input type=“text”name=“username”/>

</form>

myForm.username //引用form对象下的input子对象

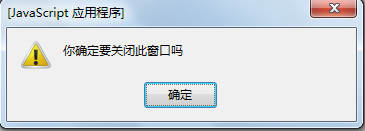
# 使用Window对象

* window对象表示整个浏览器窗口
* 系统对话框设置
* 周期性操作设置、延迟执行
* 浏览器窗口的打开和关闭
* ……

## 对话框

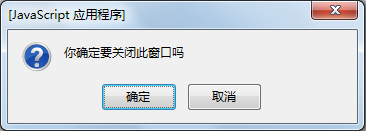
### 弹出警告框 alert()

* window.alert(message)
* Message为要在警告框中显示的内容
* **例：window.alert(‘你确定要关闭此窗口吗？’);**



### 弹出确认窗口 confirm()

* window.confirm(message);
* message为要在确认框中显示的文本内容
* **例：window.confirm(‘你确定要关闭此窗口吗？’);**



**//confirm 确定或取消,方法的返回值有两个 true false**

confirm('请确定或者取消'); **//这里按哪个都无效**

**if** (confirm('请确定或者取消')) { **//confirm 本身有返回值**

alert('您按了确定！'); **//按确定返回 true**

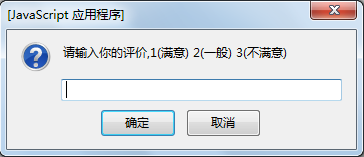
} **else** {

alert('您按了取消！'); **//按取消返回 false,按X也是返回false**

}

### 弹出输入框 prompt()

* window.prompt(message, [default]);
* message为要在提示框中显示的文本内容，default为单行文本框中的默认内容
* **例：window.prompt(‘请输入你的评价, 1(满意) 2(一般) 3(不满意)’, ‘’);**



**//prompt 输入提示框,用户向提示框中输入什么，就返回什么**

**var** num = prompt('请输入一个数字', 0);

**//两个参数，一个提示，一个值，如果不输入就返回提示的值;输入什么，就返回什么**

alert(num); ***//*返回值可以得到，取消和X返回null**

## 延迟执行

* 延迟执行 setTimeout() ----- 只执行一次
  + setTimeout(code,interval)
  + code为要延迟执行的JavaScript代码（一般来说是函数）；
  + interval为延迟执行的间隔时间，单位为毫秒
* 取消延迟执行 clearTimeout()
  + clearTimeout(id)
  + id是setTimeout()方法返回的数字

// 第一种写法

function alertmes(){

var msg = setTimeout("alert('延迟执行3秒钟')",3000);//单位是毫秒

//调用 setTimeout()之后，该方法会返回一个数值 ID，表示超时调用。这个超时调用的 ID

//是计划执行代码的唯一标识符，可以通过它来取消超时调用。

clearTimeout(msg);}

// 第二种写法

function alertmes(){

var timeoutId = setTimeout(function (){alert('延迟执行3秒钟');},3000);}

## 周期执行

* 周期执行setInterval() ----- **重复执行，直到 clearInterval() 被调用或窗口被关闭。**
  + setInterval(code,interval)
  + code为要周期执行的JavaScript代码；
  + interval为周期执行的间隔时间，单位为毫秒
  + 取消周期执行clearInterval()
  + clearInterval(id)
  + id是setInterval()方法返回的数字

var intervalId;

var time = window.document.forms[0].timer;

//判断计时开始时的初值

function judgeValue(){

var time = window.document.forms[0].timer.value;

intervalId = window.setInterval(addNumber,1000);

}

//计数器加1

function addNumber()

{

time.value=parseInt(time.value)+1;

}

## 窗口操作

* 打开浏览器窗口open ()
  + window.open(url,name,features,replace)
  + url : 在打开的窗口中加载文档的URL地址。（可选参数）
  + name : 新开窗口的名称。（可选参数）
  + features : 新开窗口的特性。（可选参数）
  + replace : 是否替换窗口中浏览的当前历史。布尔值。（可选参数）
* 关闭浏览器窗口close ()
  + window.close() //关闭当前页面的当前的窗口，可以关闭指定窗口

# document文档对象

* window对象的document子对象代表了在浏览器中加载的文档。
* 常用方法：在浏览器窗口输出文字 write( )

document.write(str)

document.write(str1,str2…)

## 子对象

forms 表单对象

<body>

<form name=“myForm1”></form>

<form name=“myForm2”></form>

<form name=“myForm3”></form>

</body>

* 同一文档中的所有表单对象都放在forms[ ]数组中
* 使用数组的方式引用表单 document.forms[0]
* 使用表单名直接引用表单document.myForm1

（1）forms 表单对象

（2）images 图片对象

（3）links 超链接对象

（4）anchors 锚对象

（5）location 当前文档的URL对象

# history历史对象

* **history对象可以访问浏览器窗口的浏览历史**

|  |  |  |
| --- | --- | --- |
| **对象** | **方法** | **说明** |
| **history** | **back()** | 后退到上一个访问过的页面 |
| **forward()** | 前进到下一个访问过的页面 |
| **go(n)** | 跳转到某个访问过的页面 |

go(n)如果n为正数，则前进到第n个访问过的网页；如果n为负数，则后退到第n个访问过的网页。

function windowForward(){

window.history.forward();

}

function windowBack(){

window.history.back();

}

# location地址对象

* location对象包含当前窗口的URL信息。
  + 可以通过修改location对象的属性来加载另一个文档。

|  |  |  |
| --- | --- | --- |
| **对象** | **属性** | **说明** |
| **location** | **host** | **主机名和端口** |
| **hostname** | **主机名** |
| **href** | **当前页面的URL** |
| **port** | **端口号** |

function jumpToBaidu(){

window.location.href = "https://www.baidu.com";

}

setTimeout("jumpToBaidu()", 3000);

# navigator浏览器对象

* navigator对象包含浏览器的信息，浏览器的类型、版本信息都可以从中获取。

|  |  |  |
| --- | --- | --- |
| **对象** | **属性** | **说明** |
| **navigator** | **appName** | **浏览器名称** |
| **appVersion** | **浏览器版本和运行平台** |
| **onLine** | **是否在线(非脱机)** |
| **platform** | **浏览器运行平台** |
| **userAgent** | **HTTP用户代理请求头的字符串** |



# srceen屏幕对象

* screen对象包含客户端显示器屏幕的相关信息。

|  |  |  |
| --- | --- | --- |
| **对象** | **属性** | **说明** |
| **screen** | **height** | **屏幕的垂直分辨率，单位：像素** |
| **width** | **屏幕的水平分辨率，单位：像素** |
| **availHeight** | **可用屏幕高度** |
| **availWidth** | **可用屏幕宽度** |