BOM

浏览器对象模型 (BOM)

浏览器对象模型（**B**rowser **O**bject **M**odel (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。

Window 对象

所有浏览器都支持 window 对象。它表示浏览器窗口。

所有 JavaScript 全局对象、函数以及变量均自动成为 window 对象的成员。

全局变量是 window 对象的属性。

全局函数是 window 对象的方法。

甚至 HTML DOM 的 document 也是 window 对象的属性之一：

```
window.document.getElementById("header");
//简写
document.getElementById("header");
```

Window 尺寸

有三种方法能够确定浏览器窗口的尺寸。

对于Internet Explorer、Chrome、Firefox、Opera 以及 Safari：

- window.innerHeight - 浏览器窗口的内部高度(包括滚动条)
- window.innerWidth - 浏览器窗口的内部宽度(包括滚动条)

对于 Internet Explorer 8、7、6、5：

- document.documentElement.clientHeight
- document.documentElement.clientWidth

或者

- document.body.clientHeight
- document.body.clientWidth

实用的 JavaScript 方案，该例显示浏览器窗口的高度和宽度。（涵盖所有浏览器，包含 IE8 及以下版本的浏览器）：

```
<p id="demo"></p>
<script>
var w=window.innerWidth
      || document.documentElement.clientWidth
      || document.body.clientWidth;
var h=window.innerHeight
      || document.documentElement.clientHeight
      || document.body.clientHeight;
x=document.getElementById("demo");
x.innerHTML="浏览器window宽度: " + w + ", 高度: " + h + "。"
</script>
```

其他 Window 方法

一些其他方法：

- window.open() - 打开新窗口
- window.close() - 关闭当前窗口
- window.moveTo() - 移动当前窗口
- window.resizeTo() - 调整当前窗口的尺寸

Window Screen

window.screen 对象包含有关用户屏幕的信息。

window.screen对象在编写时可以不使用 window 这个前缀。

一些属性：

- screen.availWidth - 可用的屏幕宽度
- screen.availHeight - 可用的屏幕高度

Window Screen 可用宽度

screen.availWidth 属性返回访问者屏幕的宽度，以像素计，减去界面特性，比如窗口任务栏。

```
<script>
document.write("可用宽度: " + screen.availWidth);
</script>
可用宽度: 1920
```

Window Screen 可用高度

screen.availHeight 属性返回访问者屏幕的高度，以像素计，减去界面特性，比如窗口任务栏。

```
<script>
document.write("可用高度: " + screen.availHeight);
</script>
可用高度: 1032
```

Window Location

window.location 对象用于获得当前页面的地址 (URL)，并把浏览器重定向到新的页面。

window.location 对象在编写时可不使用 window 这个前缀。 一些例子：

一些实例:

- location.hostname 返回 web 主机的域名
- location.pathname 返回当前页面的路径和文件名
- location.port 返回 web 主机的端口 （80 或 443）
- location.protocol 返回所使用的 web 协议（http: 或 https:）

Window Location href

location.href 属性返回当前页面的 URL。

```
<script>
document.write(location.href);
</script>

https://www.runoob.com/js/js-window-location.html
```

window Location pathname

location.pathname 属性返回 URL 的路径名。

```
<script>
document.write(location.pathname);
</script>

/js/js-window-location.html
```

Window Location assign

location.assign() 方法加载新的文档。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>菜鸟教程(runoob.com)</title>
<script>
function newDoc(){
    window.location.assign("https://www.runoob.com")
}
</script>
</head>
<body>
<input type="button" value="加载新文档" onclick="newDoc()">
</body>
</html>
```

window.location.assign(url)： 加载 URL 指定的新的 HTML 文档。 就相当于一个链接，跳转到指定的url，当前页面会转为新页面内容，可以点击后退返回上一个页面。

window.location.replace(url)： 通过加载 URL 指定的文档来替换当前文档 ，这个方法是替换当前窗口页面，前后两个页面共用一个窗口，所以是没有后退返回上一页的

Window History

window.history 对象包含浏览器的历史。

window.history对象在编写时可不使用 window 这个前缀。

为了保护用户隐私，对 JavaScript 访问该对象的方法做出了限制。

一些方法：

- history.back() - 与在浏览器点击后退按钮相同
- history.forward() - 与在浏览器中点击向前按钮相同

Window history.back()

history.back() 方法加载历史列表中的前一个 URL。

这与在浏览器中点击后退按钮是相同的：

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<head>
<script>
function goBack()
{
    window.history.back()
}
</script>
</head>
<body>

<input type="button" value="Back" onclick="goBack()">

</body>
</html>
```

Window history.forward()

history forward() 方法加载历史列表中的下一个 URL。

这与在浏览器中点击前进按钮是相同的：

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script>
function goForward()
{
    window.history.forward()
}
</script>
</head>
<body>

<input type="button" value="Forward" onclick="goForward()">

</body>
</html>
```

Window history.go()

除此之外可以用 **history.go()** 这个方法来实现向前，后退的功能。

```
function a(){
    history.go(1);  // go() 里面的参数表示跳转页面的个数 例如 history.go(1) 表示前进一个页面
}
function b(){
    history.go(-1);  // go() 里面的参数表示跳转页面的个数 例如 history.go(-1) 表示后退一个页面
}
function a(){
    history.go(0);  // go() 里面的参数为0,表示刷新页面
}
```

Window Navigator

window.navigator 对象包含有关访问者浏览器的信息。

window.navigator 对象在编写时可不使用 window 这个前缀。

```
<script>
txt = "<p>浏览器代号: " + navigator.appCodeName + "</p>";
txt+= "<p>浏览器名称: " + navigator.appName + "</p>";
txt+= "<p>浏览器版本: " + navigator.appVersion + "</p>";
```

```
txt+= "<p>启用Cookies: " + navigator.cookieEnabled + "</p>";
txt+= "<p>硬件平台: " + navigator.platform + "</p>";
txt+= "<p>用户代理: " + navigator.userAgent + "</p>";
txt+= "<p>用户代理语言: " + navigator.language + "</p>";
document.getElementById("example").innerHTML=txt;
</script>
```

警告!!!

来自 navigator 对象的信息具有误导性，不应该被用于检测浏览器版本，这是因为：

- navigator 数据可被浏览器使用者更改
- 一些浏览器对测试站点会识别错误
- 浏览器无法报告晚于浏览器发布的新操作系统

浏览器检测

由于 navigator 可误导浏览器检测，使用对象检测可用来嗅探不同的浏览器。

由于不同的浏览器支持不同的对象，您可以使用对象来检测浏览器。例如，由于只有 Opera 支持属性 "window.opera"，您可以据此识别出 Opera。

例子：if (window.opera) {...some action...}

JavaScript 弹窗

可以在 JavaScript 中创建三种消息框：警告框、确认框、提示框。

警告框

警告框经常用于确保用户可以得到某些信息。

当警告框出现后，用户需要点击确定按钮才能继续进行操作。

```
window.alert("sometext");
```

确认框

确认框通常用于验证是否接受用户操作。

当确认框弹出时，用户可以点击 "确认" 或者 "取消" 来确定用户操作。

当你点击 "确认", 确认框返回 true， 如果点击 "取消", 确认框返回 false。

```
window.confirm("sometext");
```

提示框

提示框经常用于提示用户在进入页面前输入某个值。

当提示框出现后，用户需要输入某个值，然后点击确认或取消按钮才能继续操纵。

如果用户点击确认，那么返回值为输入的值。如果用户点击取消，那么返回值为 null。

```
window.prompt("sometext","defaultvalue");
```

JavaScript 计时事件

JavaScript 一个设定的时间间隔之后来执行代码，我们称之为计时事件

通过使用 JavaScript，我们有能力做到在一个设定的时间间隔之后来执行代码，而不是在函数被调用后立即执行。我们称之为计时事件。

在 JavaScript 中使用计时事件是很容易的，两个关键方法是:

- setInterval() - 间隔指定的毫秒数不停地执行指定的代码。
- setTimeout() - 在指定的毫秒数后执行指定代码。

注意: setInterval() 和 setTimeout() 是 HTML DOM Window对象的两个方法。

setInterval() 方法

setInterval() 间隔指定的毫秒数不停地执行指定的代码

```
window.setInterval("javascript function",milliseconds);
```

window.setInterval() 方法可以不使用 window 前缀，直接使用函数 **setInterval()** 。

setInterval() 第一个参数是函数（function）；第二个参数间隔的毫秒数

如何停止执行?

clearInterval() 方法用于停止 setInterval() 方法执行的函数代码。

```
window.clearInterval(intervalVariable)
```

window.clearInterval() 方法可以不使用window前缀，直接使用函数 **clearInterval()** 。

要使用 clearInterval() 方法， 在创建计时方法时你必须使用全局变量：

```
myVar=setInterval("javascript function",milliseconds);
```

```
<p id="demo"></p>
<button onclick="myStopFunction()">停止</button>
<script>
var myVar=setInterval(function(){myTimer()},1000);
function myTimer(){
    var d=new Date();
    var t=d.toLocaleTimeString();
    document.getElementById("demo").innerHTML=t;
}
function myStopFunction(){
    clearInterval(myVar);
}
</script>
```

setTimeout() 方法

```
myVar= window.setTimeout("javascript function", milliseconds);
```

setTimeout() 方法会返回某个值。在上面的语句中，值被储存在名为 myVar 的变量中。假如你希望取消这个 setTimeout()，你可以使用这个变量名来指定它。

setTimeout() 的第一个参数是含有 JavaScript 语句的字符串。这个语句可能诸如 "alert('5 seconds!')"，或者对函数的调用，诸如 alertMsg。

第二个参数指示从当前起多少毫秒后执行第一个参数。

```
setTimeout(function(){alert("Hello")},3000);
```

如何停止执行?

clearTimeout() 方法用于停止执行setTimeout()方法的函数代码。

```
window.clearTimeout(timeoutVariable)
```

window.clearTimeout() 方法可以不使用window 前缀。

要使用clearTimeout() 方法, 你必须在创建超时方法中（setTimeout）使用全局变量:

```
myVar=setTimeout("javascript function",milliseconds);
```

如果函数还未被执行，你可以使用 clearTimeout() 方法来停止执行函数代码。

```
var myVar;

function myFunction()
{
    myVar=setTimeout(function(){alert("Hello")},3000);
}

function myStopFunction()
{
    clearTimeout(myVar);
}
```

JavaScript Cookie

Cookie 是一些数据, 存储于你电脑上的文本文件中。

当 web 服务器向浏览器发送 web 页面时，在连接关闭后，服务端不会记录用户的信息。

Cookie 的作用就是用于解决 "如何记录客户端的用户信息":

- 当用户访问 web 页面时，他的名字可以记录在 cookie 中。
- 在用户下一次访问该页面时，可以在 cookie 中读取用户访问记录。

当浏览器从服务器上请求 web 页面时， 属于该页面的 cookie 会被添加到该请求中。服务端通过这种方式来获取用户的信息。

使用 JavaScript 创建Cookie

JavaScript 可以使用 **document.cookie** 属性来创建 、读取、及删除 cookie。

JavaScript 中，创建 cookie 如下所示：

```
document.cookie="username=John Doe";
```

您还可以为 cookie 添加一个过期时间（以 UTC 或 GMT 时间）。默认情况下，cookie 在浏览器关闭时删除：

```
document.cookie="username=John Doe; expires=Thu, 18 Dec 2043 12:00:00 GMT";
```

您可以使用 path 参数告诉浏览器 cookie 的路径。默认情况下，cookie 属于当前页面。

```
document.cookie="username=John Doe; expires=Thu, 18 Dec 2043 12:00:00 GMT; path=/";
```

使用 JavaScript 读取 Cookie

在 JavaScript 中, 可以使用以下代码来读取 cookie：

```
var x = document.cookie;

document.cookie 将以字符串的方式返回所有的 cookie，类型格式： cookie1=value;
cookie2=value; cookie3=value;
```

使用 JavaScript 修改 Cookie

在 JavaScript 中，修改 cookie 类似于创建 cookie，如下所示：

```
document.cookie="username=John Smith; expires=Thu, 18 Dec 2043 12:00:00 GMT;
path=/";
```

使用 JavaScript 删除 Cookie

删除 cookie 非常简单。您只需要设置 expires 参数为以前的时间即可，如下所示，设置为 Thu, 01 Jan 1970 00:00:00 GMT:

```
document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 GMT";
```

Cookie 字符串

document.cookie 属性看起来像一个普通的文本字符串，其实它不是。

即使您在 document.cookie 中写入一个完整的 cookie 字符串, 当您重新读取该 cookie 信息时，cookie 信息是以名/值对的形式展示的。

如果您设置了新的 cookie，旧的 cookie 不会被覆盖。 新 cookie 将添加到 document.cookie 中，所以如果您重新读取document.cookie，您将获得如下所示的数据：

cookie1=value; cookie2=value;

如果您需要查找一个指定 cookie 值，您必须创建一个JavaScript 函数在 cookie 字符串中查找 cookie 值。

设置 cookie 值的函数

首先，我们创建一个函数用于存储访问者的名字：

```
function setCookie(cname,cvalue,exdays)
{
  var d = new Date();
  d.setTime(d.getTime()+(exdays*24*60*60*1000));
  var expires = "expires="+d.toGMTString();
  document.cookie = cname + "=" + cvalue + "; " + expires;
}
```

函数解析：

以上的函数参数中，cookie 的名称为 cname，cookie 的值为 cvalue，并设置了 cookie 的过期时间 expires。

该函数设置了 cookie 名、cookie 值、cookie过期时间。

获取 cookie 值的函数

然后，我们创建一个函数用于返回指定 cookie 的值：

```
function getCookie(cname)
{
  var name = cname + "=";
  var ca = document.cookie.split(';');
  for(var i=0; i<ca.length; i++)
  {
    var c = ca[i].trim();
    if (c.indexOf(name)==0) return c.substring(name.length,c.length);
  }
  return "";
}
```

函数解析：

cookie 名的参数为 cname。

创建一个文本变量用于检索指定 cookie :cname + "="。

使用分号来分割 document.cookie 字符串，并将分割后的字符串数组赋值给 ca (ca = document.cookie.split(';'))。

循环 ca 数组 (i=0;i<ca.length;i++)，然后读取数组中的每个值，并去除前后空格 (c=ca[i].trim())。

如果找到 cookie(c.indexOf(name) == 0)，返回 cookie 的值 (c.substring(name.length,c.length)。

如果没有找到 cookie, 返回 ""。

检测 cookie 值的函数

最后，我们可以创建一个检测 cookie 是否创建的函数。

如果设置了 cookie，将显示一个问候信息。

如果没有设置 cookie，将会显示一个弹窗用于询问访问者的名字，并调用 setCookie 函数将访问者的名字存储 365 天：

```
function checkCookie()
{
  var username=getCookie("username");
  if (username!="")
  {
    alert("Welcome again " + username);
  }
  else
  {
    username = prompt("Please enter your name:","");
    if (username!="" && username!=null)
    {
      setCookie("username",username,365);
    }
  }
}
```