

BOM

内容提要

- 1 BOM
- 2 window 对象
- 3 location 对象
- 4 navigator 对象
- 5 history 对象

1 BOM

BOM (Browser Object Model, 浏览器对象模型)

- BOM 提供一系列访问浏览器功能的对象。
- BOM 提供的功能与具体的网页内容无关。
- 是 JS 组成的一部分，主要方面已纳入 HTML5 规范，如
 - 弹出新的浏览器窗口
 - 移动、关闭浏览器窗口以及调整窗口大小
 - 提供 Web 浏览器详细信息的定位对象
 - 提供用户屏幕分辨率详细信息的屏幕对象
 - 对 cookie 的支持

1 BOM

BOM 主要提供的对象和功能

- window 对象：访问浏览器接口和 Global 对象（4.1.4 节），包括
 - location 对象：提供当前窗口加载的文档有关信息，如 URL、服务器名等。
 - navigator 对象：提供浏览器和客户端的信息，如浏览器名称和操作系统等。
 - history 对象：保存当前标签页的上网记录。

window对象作用

➤ Global对象

➤所有在全局作用域中声明的变量、函数都是 window 对象的属性和方法

➤通过JavaScript访问浏览器窗口的接口

➤窗口大小

➤导航和打开窗口

➤间歇调用和超时调用

➤系统对话框

window: 作为 Global 对象

- 所有在**全局作用域**中声明的变量、函数都是 window 对象的属性和方法。

```
function setGlobalVar() {  
    window.name = "Somebody";  
    location = "Beijing";  
}  
  
alert(typeof name);           // string  
alert(typeof location);      // undefined
```

Demo 3.0

window: 访问浏览器的接口

➤ 窗口位置

- screenX 属性: 浏览器窗口相对屏幕左侧的位置。
- screenY 属性: 浏览器窗口相对屏幕顶部的位置。

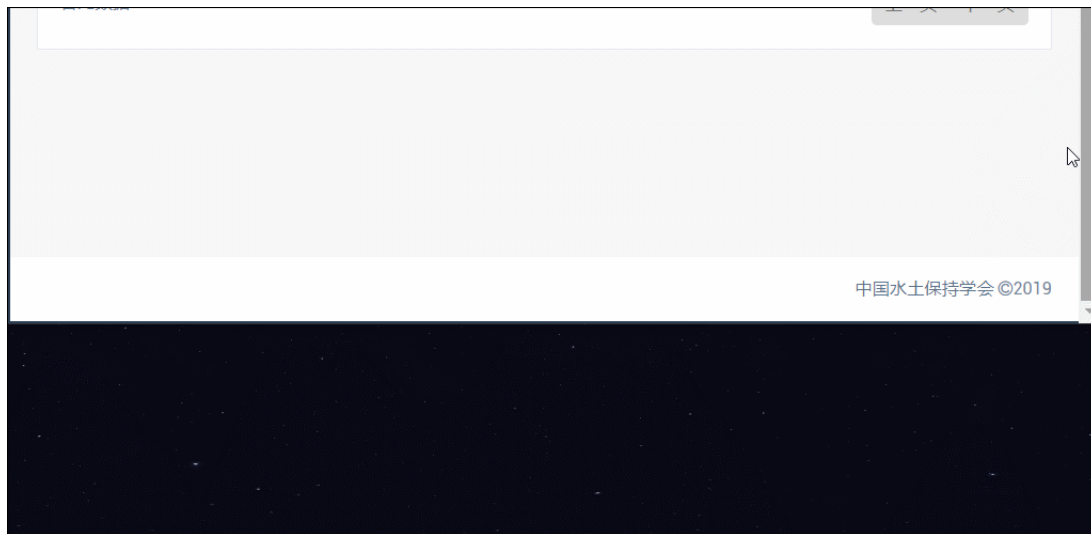
```
alert(window.screenX);  
alert(window.screenY);
```

Demo 3.1

窗口大小

- innerHeight 属性：页面视图容器（实际页面）的高度。
- innerWidth 属性：页面视图容器（实际页面）的宽度。
- outerHeight 属性：浏览器窗口本身的高度。
- outerWidth 属性：浏览器窗口本身的宽度。

Demo 3.2



可以根据页面的实际尺寸动态计算位置

跳转或弹出窗口

- `open(URL, [target, features, replace])` 方法跳转或弹出新窗口。
 - `URL(string)`: 要加载的 URL
 - `target(string)` 窗口目标: 在具有该名称的窗口或框架中加载 URL
 - `_self`: 在当前窗口载入新 URL
 - `_blank`: 弹出新窗口加载新 URL
 - `features(string)` 特性: 根据该特性创建新窗口
 - `replace(boolean)` 是否要新页面取代浏览历史记录中当前加载页面

Demo 3.3

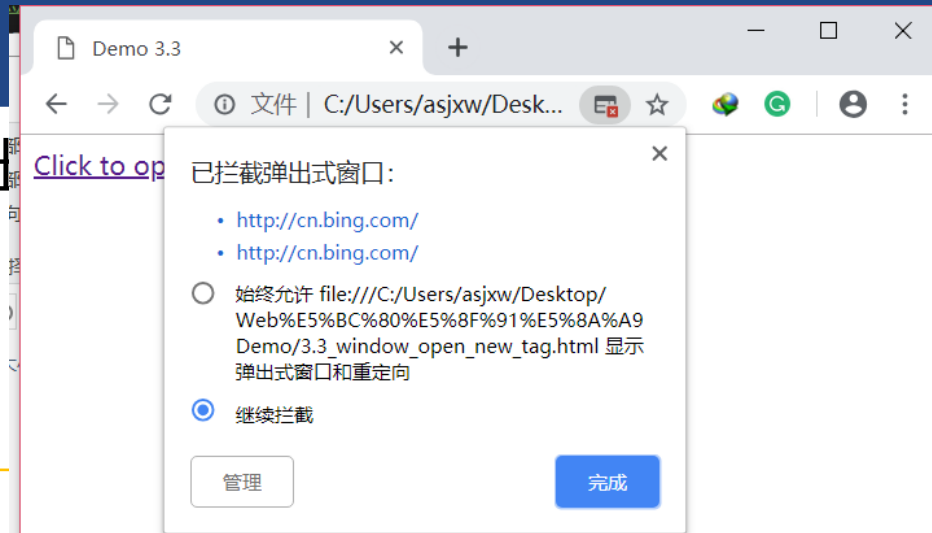
```
/* 在本窗口中打开, Chrome */  
window.open("http://cn.bing.com/", "_self");  
  
/* 在新标签页中打开尺寸为 100*100 的新标签页, Chrome */  
window.open("http://cn.bing.com", "_blank", "width=100,height=100");
```

跳转或弹出窗口

➤ 设置参数 `target='_blank'` 时部分浏览器具有拦截弹出

```
/* 判断窗口是否被浏览器扩展或其他窗口屏蔽 */
```

```
var blocked = false;
try {
  var newTag = window.open("http://cn.bing.com/", "_blank");
  if (newTag == null) {
    blocked = true;
  }
} catch (ex) {
  blocked = true;
}
if (blocked) {
  alert("The popup was blocked!");
}
```



Demo 3.3

跳转或弹出窗口

- ▶ 设置参数 `target='_blank'` 时部分浏览器具有拦截弹出式窗口功能，防止广告。

Demo 3.3

```
<!-- 直接在 HTML 的 <a> 标签中设置 target='_blank' 属性，不会被拦截 -->  
<a href="http://cn.bing.com/" target='_blank'>打开新标签</a>
```

系统对话框

Demo 3.4

- `prompt(message, default)` 弹出提示框, 包含文本输入域和确认按钮
 - `message`: 要显示给用户的文本提示
 - `default`: 文本输入域的默认值
- 若用户输入并点击"确定", 返回输入值
- 如果用户点击"取消", 返回 `null`

```
var result = prompt("请输入姓名", "Wang");  
  
if (result != null) {  
    document.write(result);  
    alert("Welcome, " + result);  
    window.print();  
}
```

此网页显示

请输入姓名

系统对话框

Demo 3.4

- `alert(message)`: 弹出警告框

此网页显示
Welcome, Wang

确定

- `confirm(message)`: 弹出确认对话框

此网页显示
打印你的名字吗?

- 返回 true/false

确定


取消

- `print()`: 显示"打印"对话框

打印

总计: 1 张纸

打印 取消

目标打印机  EPSON L380 Series

更改...

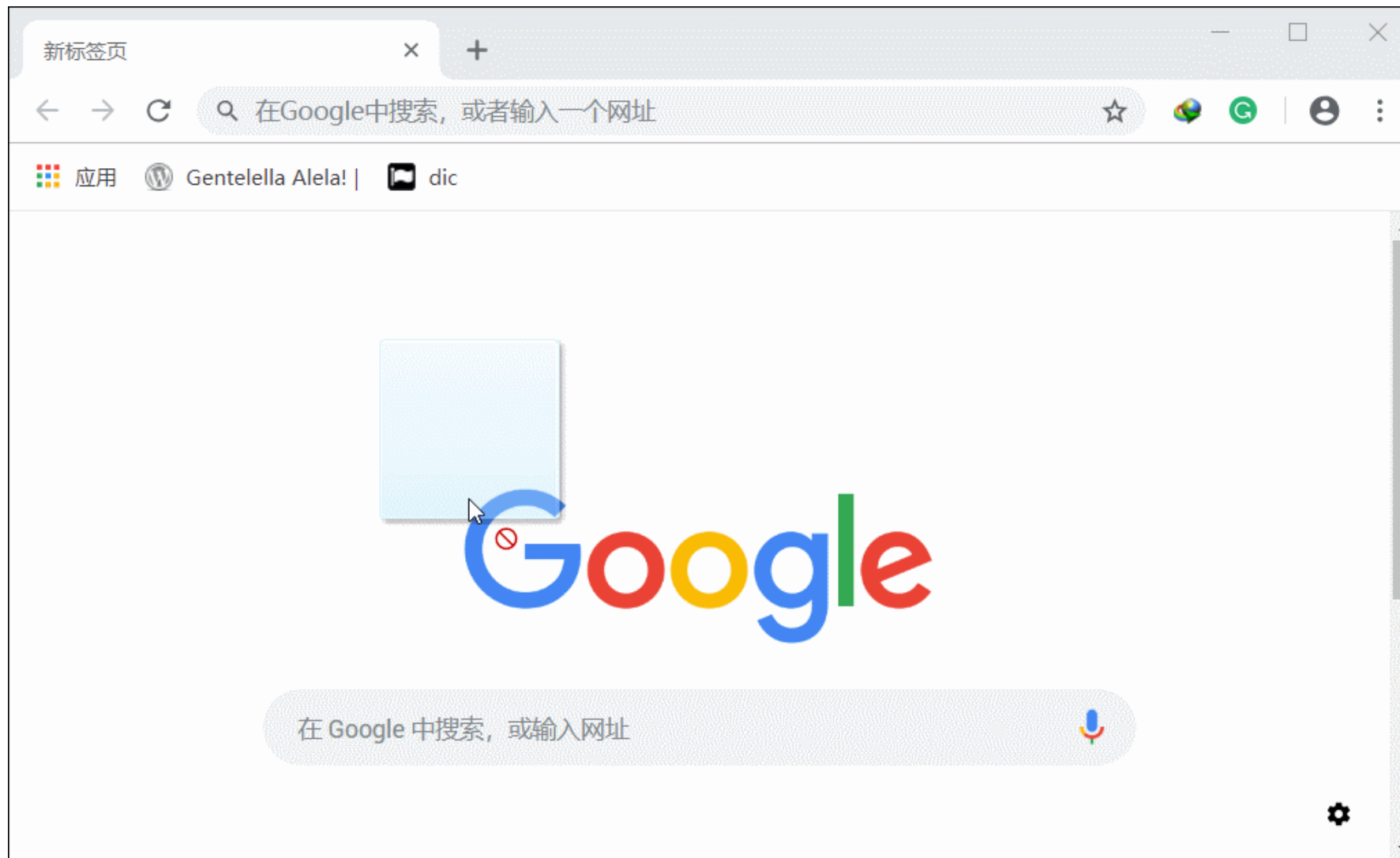
页码 全部

例如: 1-5、8、11-13

份数

倒计时

Demo 3.4



倒计时

超时调用

超时调用 `setTimeout(function/string, millisecond)`

- 将需要执行的代码加入任务队列中，并在指定时间过后执行代码。
 - *function/string*: 使包含 JS 代码的字符串 (如 `eval()` 参数, 4.1.4 节) , 也可以是一个函数。
 - *millisecond*: 第二个参数表示 `millisecond` 毫秒后执行代码。
- 返回 `string` 类型的 ID, 唯一标识计划执行代码, 可以用它取消超时调用。

取消超时调用 `clearTimeout(id)`

- 取消尚未执行的超时调用计划。

超时调用

超时调用 `setTimeout(function/string, millisecond)`

Demo 3.7

例：倒计时跳转至其他页面的 `setTimeout()` 实现

```
<span id="timer">4</span> 秒后跳转至主页...<span></span>
<script>
  var x = 3;
  setTimeout(go, 1000);           window.onload = function () {
  function go() {
    if (x >= 1) {
      document.getElementById("timer").innerText = x;
      setTimeout(go, 1000);
    } else {
      window.open("http://cn.bing.com/", "_self");
    }
    x--;
  }
</script>
```


间歇调用

间歇调用 `setInterval(function/string, millisecond)`

- *function/string*: 使包含 JS 代码的字符串 (如 `eval()` 参数, 4.1.4 节), 也可以是一个函数。
- *millisecond*: 第二个参数表示 `millisecond` 毫秒后执行代码。
- 返回 `string` 类型的 ID, 唯一标识计划执行代码, 可以用它取消超时调用。

取消超时调用 `clearTimeout(id)`

- 取消尚未执行的超时调用计划。

间歇调用

间歇调用 `setInterval(function/string, millisecond)`

Demo 3.6

例：倒计时跳转至其他页面的 `setInterval()` 实现

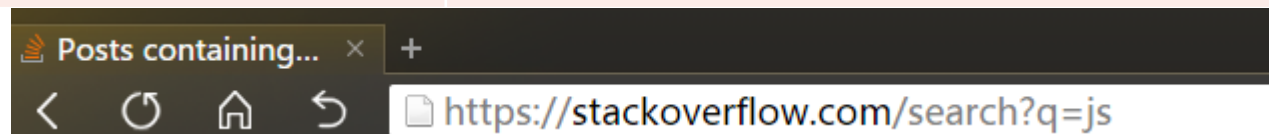
```
<span id="timer">4</span> 秒后跳转至主页...<span></span>
<script>
  var id = setInterval(go, 1000);
  var x = 3;
  function go() {
    if (x >= 1) {
      document.getElementById("timer").innerText = x;
    } else {
      clearInterval(id);
      window.open("http://cn.bing.com/", "_self");
    }
    x--;
  }
</script>
```

3 location 对象

保存当前文档信息并解析 URL

➤ location 对象属性:

属性名	说明	实例
href	返回完整 URL	"http://www.srw.com:80/#part1"
host	返回网站名和端口号 (若存在)	"www.srw.com:80"
hostname	返回网站名	"www.srw.com"
protocol	返回使用的协议	"http"
port	返回端口号	"80"
hash	返回 URL 中 # 和后面的字符(hash)	"#part1"
pathname	返回 URL 中的目录或文件夹名	"/course/web_design/"
search	返回 URL 的查询字符串	"?q="



The animation effect of this component is dependent on the **preferred-reduced-motion** media query.

```
Elements Console Sources Network Performance Memory >> 1 | X
top Filter Default levels |
> location.href
< "https://getbootstrap.com/docs/4.3/components/modal/#examples"
> location.host
< "getbootstrap.com"
> location.hostname
< "getbootstrap.com"
> location.protocol
< "https:"
> location.port
< ""
> location.hash
< "#examples"
> location.pathname
< "/docs/4.3/components/modal/"
> location.search
< ""
>
```

修改 location 属性改变当前加载的页面

```
// 初始 URL: https://getbootstrap.com/docs/4.3/components/modal/#examples  
  
//将 URL 修改为 http://cn.bing.com/  
location.url = http://cn.bing.com/  
  
// 将 hash 修改为 #live-demo  
location.hash = "#live-demo"
```

URL传递

➤ 使用 `assign(URL)` 方法传递 URL, 改变浏览器位置并生成历史记录

```
// 初始 URL: https://getbootstrap.com/docs/4.3/components/modal/#examples
```

```
//将 URL 修改为 http://cn.bing.com/
```

```
location.assign("http://cn.bing.com/");
```

➤ 使用 `replace(URL)` 方法传递 URL, 只改变浏览器位置, 不生成历史记录, 无法使用后退功能

```
// 初始 URL: https://getbootstrap.com/docs/4.3/components/modal/#examples
```

```
//将 URL 修改为 http://cn.bing.com/
```

```
location.replace("http://cn.bing.com/");
```

reload

使用 `reload([reLoad])` 方法重新加载当前页面

若页面从上次请求以来无改动，直接从浏览器缓存中重新加载。

reLoad = true: 强制重新从服务器加载页面

```
// 初始 URL: https://getbootstrap.com/docs/4.3/components/modal/#examples
```

```
// 刷新当前页面
```

```
location.reload();
```

通过 navigator 对象识别浏览器

- navigator 对象中保存浏览器和客户端的部分厂商、版本信息。
- 不同浏览器的 navigator 实现不一致，标准不统一。
- 由于历史原因，存在浏览器品牌的伪装和欺骗，使得直接检测浏览器困难，需要比对隐藏的特征字符串。


```
[object Navigator]: {activeVRDisplays: Array, appCodeName: "Mozilla", appName: "Netscape", appVersion: "5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/17.17134", cookieEnabled: true...}
```

```
activeVRDisplays: Array  
  appCodeName: "Mozilla"  
  appName: "Netscape"  
  appVersion: "5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/17.17134"  
  cookieEnabled: true  
  doNotTrack: null  
  geolocation: Object  
    hardwareConcurrency: 4  
    language: "zh-CN"  
  languages: Array  
    maxTouchPoints: 0  
  mediaDevices: Object  
  mimeTypes: Array  
    msManipulationViewsEnabled: true  
    onLine: true  
    platform: "Win32"  
  plugins: Array  
    product: "Gecko"  
    productSub: "20030107"  
  serviceWorker: Object  
    userAgent: "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/17.17134"  
    vendor: ""  
    vendorSub: ""  
    webdriver: false  
  __proto__: Object
```

```
navigator  
  Navigator  
    activeVRDisplays: Array []  
    appCodeName: "Mozilla"  
    appName: "Netscape"  
    appVersion: "5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/17.17134"  
    buildID: "20181001000000"  
    clipboard: Clipboard { }  
    cookieEnabled: true  
    credentials: CredentialsContainer { }  
    doNotTrack: "unspecified"  
    geolocation: Geolocation { }  
    hardwareConcurrency: 4  
    language: "zh-CN"  
    languages: Array(6) [ "zh-CN", "zh", "zh-TW", ... ]  
      maxTouchPoints: 0  
    mediaCapabilities: MediaCapabilities { }  
      mediaDevices: MediaDevices { ondevicechange: null }  
      mimeTypes: MimeTypeArray { length: 0 }  
      onLine: true  
      oscpu: "Windows NT 10.0; Win64; x64"  
    permissions: Permissions { }  
      platform: "Win32"  
    plugins: PluginArray { length: 0 }  
      product: "Gecko"  
      productSub: "20100101"  
    serviceWorker: ServiceWorkerContainer { controller: null, ready: Promise { "pending" }, oncontrollerchange: null, ... }  
      storage: StorageManager { }  
      userAgent: "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:65.0) Gecko/20100101 Firefox/65.0"  
      vendor: ""
```

```
navigator { vendorSub: "" }  
  appCodeName: "Mozilla"  
  appName: "Netscape"  
  appVersion: "5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/17.17134"  
  bluetooth: Bluetooth { }  
  clipboard: Clipboard { }  
  connection: NetworkInformation { }  
  cookieEnabled: true  
  credentials: CredentialsContainer { }  
  deviceMemory: 8  
  doNotTrack: null  
  geolocation: Geolocation { }  
  hardwareConcurrency: 4  
  keyboard: Keyboard { }  
  language: "zh-CN"  
  languages: (2) [ "zh-CN", "zh" ]  
  locks: LockManager { }  
  maxTouchPoints: 0  
  mediaCapabilities: MediaCapabilities { }  
  mediaDevices: MediaDevices { ondevicechange: null }  
  mimeTypes: MimeTypeArray { length: 0 }  
  onLine: true  
  permissions: Permissions { }  
  platform: "Win32"  
  plugins: PluginArray { length: 0 }  
  presentation: Presentation { }  
  product: "Gecko"  
  productSub: "20030107"  
  serviceWorker: ServiceWorkerContainer { }  
  storage: StorageManager { }  
  usb: USB { onconnect: null }  
  userActivation: UserActivation { }  
  userAgent: "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/17.17134"  
  vendor: "Google Inc."  
  vendorSub: ""  
  webkitPersistentStorage: WebKitPersistentStorage { }  
  webkitTemporaryStorage: WebKitTemporaryStorage { }  
  __proto__: Navigator
```

大量属性的值相同
可通过对比 userAgent 属性
检测浏览器

5 history 对象

history 对象保存着用户上网的历史记录，但不能直接查看历史记录的 URL

➤ 可通过调用 history 对象方法在历史记录中向前或向后跳转网页

history.go(pages)

■ *pages(int)*: 跳转页数。正数表示向前跳转，负数向后跳转。

history.back() 等同 history.go(-1)

history.forward() 等同 history.go(1)

```
// 后退一页  
history.back();
```

```
// 前进一页  
history.forward();
```

