DOM [window](https://developer.mozilla.org/zh-CN/docs/Web/API/Window" \o "The window object represents a window containing a DOM document; the document property points to the DOM document loaded in that window.) 对象通过 [history](https://developer.mozilla.org/zh-CN/docs/Web/API/Window/history" \o "Window.history是一个只读属性，用来获取History 对象的引用，History 对象提供了操作浏览器会话历史（浏览器地址栏中访问的页面，以及当前页面中通过框架加载的页面）的接口。) 对象提供了对浏览器历史的访问。它暴露了很多有用的方法和属性，允许你在用户浏览历史中向前和向后跳转，同时——从HTML5开始——提供了对history栈中内容的操作。

## 在history中跳转

使用 back(), forward() 和 go() 方法来完成在用户历史记录中向后和向前的跳转。

### 向前和向后跳转

在history中向后跳转：

window.history.back();

这和用户点击浏览器回退按钮的效果相同。

类似地，你可以向前跳转（如同用户点击了前进按钮）：

window.history.forward();

### 跳转到 history 中指定的一个点

你可以用 go() 方法载入到会话历史中的某一特定页面， 通过与当前页面相对位置来标志 (当前页面的相对位置标志为0).

向后移动一个页面 (等同于调用 back()):

window.history.go(-1);

向前移动一个页面, 等同于调用了 forward():

window.history.go(1);

类似地，你可以传递参数值2并向前移动2个页面，等等。

您可以通过查看长度属性的值来确定的历史堆栈中页面的数量:

var numberOfEntries = window.history.length;

**注意:** IE 支持传递URLs作为参数给 go(); 这在Gecko是不标准且不支持的。

## 添加和修改历史记录中的条目

HTML5引入了 [history.pushState()](https://developer.mozilla.org/en-US/docs/Web/API/History/pushState) 和 [history.replaceState()](https://developer.mozilla.org/en-US/docs/Web/API/History_API" \l "The_replaceState()_method) 方法，它们分别可以添加和修改历史记录条目。这些方法通常与[window.onpopstate](https://developer.mozilla.org/zh-CN/docs/Web/API/Window/onpopstate" \o "window.onpopstate是popstate事件在window对象上的事件处理程序.) 配合使用。

使用 history.pushState() 可以改变referrer，它在用户发送 [XMLHttpRequest](https://developer.mozilla.org/en/DOM/XMLHttpRequest" \o "en/XMLHttpRequest) 请求时在HTTP头部使用，改变state后创建的 [XMLHttpRequest](https://developer.mozilla.org/en/DOM/XMLHttpRequest" \o "en/XMLHttpRequest) 对象的referrer都会被改变。因为referrer是标识创建  [XMLHttpRequest](https://developer.mozilla.org/en/DOM/XMLHttpRequest" \o "en/XMLHttpRequest) 对象时 this 所代表的window对象中document的URL。

### pushState() 方法的例子

假设在 http://mozilla.org/foo.html 中执行了以下 JavaScript 代码:

var stateObj = { foo: "bar" };

history.pushState(stateObj, "page 2", "bar.html");

这将使浏览器地址栏显示为 http://mozilla.org/bar.html，但并不会导致浏览器加载 bar.html ，甚至不会检查bar.html 是否存在。

假设现在用户又访问了 http://google.com，然后点击了返回按钮。此时，地址栏将显示 http://mozilla.org/bar.html，同时页面会触发 popstate 事件，事件对象state中包含了 stateObj 的一份拷贝。页面本身与 foo.html 一样，尽管其在 popstate  事件中可能会修改自身的内容。

如果我们再次点击返回按钮，页面URL会变为http://mozilla.org/foo.html，文档对象document会触发另外一个 popstate 事件，这一次的事件对象state object为null。 这里也一样，返回并不改变文档的内容，尽管文档在接收 popstate 事件时可能会改变自己的内容，其内容仍与之前的展现一致。

### pushState() 方法

pushState() 需要三个参数: 一个状态对象, 一个标题 (目前被忽略), 和 (可选的) 一个URL. 让我们来解释下这三个参数详细内容：

**状态对象** — 状态对象state是一个JavaScript对象，通过pushState () 创建新的历史记录条目。无论什么时候用户导航到新的状态，popstate事件就会被触发，且该事件的state属性包含该历史记录条目状态对象的副本。

        状态对象可以是能被序列化的任何东西。原因在于Firefox将状态对象保存在用户的磁盘上，以便在用户重启浏览器时使用，我们规定了状态对象在序列化表示后有640k的大小限制。如果你给 pushState() 方法传了一个序列化后大于640k的状态对象，该方法会抛出异常。如果你需要更大的空间，建议使用 sessionStorage 以及 localStorage.

**标题** — Firefox 目前忽略这个参数，但未来可能会用到。传递一个空字符串在这里是安全的，而在将来这是不安全的。二选一的话，你可以为跳转的state传递一个短标题。

**URL** — 该参数定义了新的历史URL记录。注意，调用 pushState() 后浏览器并不会立即加载这个URL，但可能会在稍后某些情况下加载这个URL，比如在用户重新打开浏览器时。新URL不必须为绝对路径。如果新URL是相对路径，那么它将被作为相对于当前URL处理。新URL必须与当前URL同源，否则 pushState() 会抛出一个异常。该参数是可选的，缺省为当前URL。

**注意:** 从 Gecko 2.0 (Firefox 4 / Thunderbird 3.3 / SeaMonkey 2.1) 到 Gecko 5.0 (Firefox 5.0 / Thunderbird 5.0 / SeaMonkey 2.2)，传递的对象是使用JSON进行序列化的。 从  Gecko 6.0 (Firefox 6.0 / Thunderbird 6.0 / SeaMonkey 2.3)开始，该对象的序列化将使用[结构化克隆算法](https://developer.mozilla.org/en/DOM/The_structured_clone_algorithm" \o "en/DOM/The structured clone algorithm)。这将会使更多对象可以被安全的传递。

        在某种意义上，调用 pushState() 与 设置 window.location = "#foo" 类似，二者都会在当前页面创建并激活新的历史记录。但 pushState() 具有如下几条优点：

* 新的 URL 可以是与当前URL同源的任意URL 。而设置 window.location 仅当你只修改了哈希值时才保持同一个 [document](https://developer.mozilla.org/zh-CN/docs/Web/API/Document" \o "Document 接口提供了一些在浏览器服务中作为页面内容入口点而加载的一些页面，也就是 DOM 树。 DOM 树包括诸如 <body> 和 <table> 之类的元素，及其他元素。其也为文档（document）提供了全局性的函数，例如获取页面的 URL、在文档中创建新的 element 的函数。)。
* 如果需要，你可以不必改变URL。而设置 window.location = "#foo";在当前哈希不是 #foo 的情况下， 仅仅是新建了一个新的历史记录项。
* 你可以为新的历史记录项关联任意数据。而基于哈希值的方式，则必须将所有相关数据编码到一个短字符串里。
* 假如 标题 在之后会被浏览器用到，那么这个数据是可以被使用的（哈希则不然）。

注意 pushState() 绝对不会触发 hashchange 事件，即使新的URL与旧的URL仅哈希不同也是如此。

在 [XUL](https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XUL) 文档中，它创建指定的 XUL 元素。

在其它文档中，它创建一个命名空间URI为null的元素。

### replaceState() 方法

history.replaceState() 的使用与 history.pushState() 非常相似，区别在于  replaceState()  是修改了当前的历史记录项而不是新建一个。 注意这并不会阻止其在全局浏览器历史记录中创建一个新的历史记录项。

replaceState() 的使用场景在于为了响应用户操作，你想要更新状态对象state或者当前历史记录的URL。

**注意：** 从Gecko 2.0 (Firefox 4 / Thunderbird 3.3 / SeaMonkey 2.1) 到 Gecko 5.0 (Firefox 5.0 / Thunderbird 5.0 / SeaMonkey 2.2)，传递的对象是使用JSON进行序列化的。 从  Gecko 6.0 (Firefox 6.0 / Thunderbird 6.0 / SeaMonkey 2.3)开始，该对象的序列化将使用[结构化克隆算法](https://developer.mozilla.org/en/DOM/The_structured_clone_algorithm" \o "en/DOM/The structured clone algorithm)。这将会使更多对象可以被安全的传递。

### replaceState() 方法示例

假设 http://mozilla.org/foo.html 执行了如下JavaScript代码：

var stateObj = { foo: "bar" };

history.pushState(stateObj, "page 2", "bar.html");

        上文2行代码可以在 "replaceState() 方法示例" 部分找到。然后，假设http://mozilla.org/bar.html执行了如下 JavaScript：

history.replaceState(stateObj, "page 3", "bar2.html");

        这将会导致地址栏显示http://mozilla.org/bar2.html,，但是浏览器并不会去加载bar2.html 甚至都不需要检查 bar2.html 是否存在。

        假设现在用户重新导向到了http://www.microsoft.com，然后点击了回退按钮。这里，地址栏会显示http://mozilla.org/bar2.html。假如用户再次点击回退按钮，地址栏会显示http://mozilla.org/foo.html，完全跳过了bar.html。

### popstate 事件

        每当活动的历史记录项发生变化时， popstate 事件都会被传递给window对象。如果当前活动的历史记录项是被 pushState 创建的，或者是由 replaceState 改变的，那么 popstate事件的状态属性 state 会包含一个当前历史记录状态对象的拷贝。

使用示例请参见 [window.onpopstate](https://developer.mozilla.org/zh-CN/docs/Web/API/Window/onpopstate" \o "window.onpopstate是popstate事件在window对象上的事件处理程序.) 。

### 获取当前状态

        页面加载时，或许会有个非null的状态对象。这是有可能发生的，举个例子，假如页面（通过pushState() 或 replaceState() 方法）设置了状态对象而后用户重启了浏览器。那么当页面重新加载时，页面会接收一个onload事件，但没有 popstate 事件。然而，假如你读取了history.state属性，你将会得到如同popstate 被触发时能得到的状态对象。

你可以读取当前历史记录项的状态对象state，而不必等待popstate 事件， 只需要这样使用history.state 属性：

var currentState = history.state;

## 例子

完整的AJAX网站示例，请参阅： [Ajax navigation example](https://developer.mozilla.org/en-US/docs/Web/Guide/API/DOM/Manipulating_the_browser_history/Example" \o "/en-US/docs/Web/Guide/API/DOM/Manipulating_the_browser_history/Example).

## 规范

|  |  |  |
| --- | --- | --- |
| **Specification** | **Status** | **Comment** |
| [HTML Living Standard History](https://html.spec.whatwg.org/multipage/browsers.html" \l "history) | Living Standard | No change from [HTML5](https://www.w3.org/TR/html50/" \o "HTML5). |
| [HTML5 History](https://www.w3.org/TR/html50/browsers.html" \l "history) | Recommendation | Initial definition. |

## 浏览器兼容性

[We're converting our compatibility data into a machine-readable JSON format](https://github.com/mdn/browser-compat-data). This compatibility table still uses the old format, because we haven't yet converted the data it contains. [Find out how you can help!](https://developer.mozilla.org/zh-CN/docs/MDN/Contribute/Structures/Compatibility_tables)

* **Desktop**

* Mobile

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Feature** | **Chrome** | **Edge** | **Firefox (Gecko)** | **Internet Explorer** | **Opera** | **Safari** |
| replaceState, pushState | 5 | (Yes) | [4.0](https://developer.mozilla.org/en-US/Firefox/Releases/4" \o "Released on 2011-03-22.) (2.0) | 10 | 11.50 | 5.0 |
| history.state | 18 | (Yes) | [4.0](https://developer.mozilla.org/en-US/Firefox/Releases/4" \o "Released on 2011-03-22.) (2.0) | 10 | 11.50 | 6.0 |

****另见****

* [window.history](https://developer.mozilla.org/zh-CN/docs/Web/API/Window/history" \o "Window.history是一个只读属性，用来获取History 对象的引用，History 对象提供了操作浏览器会话历史（浏览器地址栏中访问的页面，以及当前页面中通过框架加载的页面）的接口。)
* [window.onpopstate](https://developer.mozilla.org/zh-CN/docs/Web/API/Window/onpopstate" \o "window.onpopstate是popstate事件在window对象上的事件处理程序.)