**window.postMessage**

**window.postMessage()**方法可以安全地实现跨源通信。通常，对于两个不同页面的脚本，只有当执行它们的页面位于具有相同的协议（通常为https），端口号（443为https的默认值），以及主机  (两个页面的模数 [Document.domain](https://developer.mozilla.org/zh-CN/docs/Web/API/Document/domain)设置为相同的值) 时，这两个脚本才能相互通信。**window.postMessage()**方法提供了一种受控机制来规避此限制，只要正确的使用，这种方法就很安全。

从广义上讲，一个窗口可以获得对另一个窗口的引用（比如 targetWindow = window.opener），然后在窗口上调用 targetWindow.postMessage() 方法分发一个  [MessageEvent](https://developer.mozilla.org/zh-CN/docs/Web/API/MessageEvent)消息。接收消息的窗口可以根据需要自由[处理此事件 (en-US)](https://developer.mozilla.org/en-US/docs/Web/Events)。传递给 window.postMessage() 的参数（比如 message ）将[**通过消息事件对象暴露给接收消息的窗口**](https://developer.mozilla.org/zh-CN/docs/Web/API/Window/postMessage#The_dispatched_event)。

[**语法**](https://developer.mozilla.org/zh-CN/docs/Web/API/Window/postMessage#syntax)

otherWindow.postMessage(message, targetOrigin, [transfer]);

**otherWindow**

其他窗口的一个引用，比如iframe的contentWindow属性、执行[window.open](https://developer.mozilla.org/en-US/docs/Web/API/Window/open)返回的窗口对象、或者是命名过或数值索引的[window.frames](https://developer.mozilla.org/en-US/docs/Web/API/Window/frames)。

**message**

将要发送到其他 window的数据。它将会被[结构化克隆算法](https://developer.mozilla.org/en-US/docs/DOM/The_structured_clone_algorithm)序列化。这意味着你可以不受什么限制的将数据对象安全的传送给目标窗口而无需自己序列化。[[1](https://developer.mozilla.org/en-US/docs/)]

**targetOrigin**

通过窗口的origin属性来指定哪些窗口能接收到消息事件，其值可以是字符串"\*"（表示无限制）或者一个URI。在发送消息的时候，如果目标窗口的协议、主机地址或端口这三者的任意一项不匹配targetOrigin提供的值，那么消息就不会被发送；只有三者完全匹配，消息才会被发送。这个机制用来控制消息可以发送到哪些窗口；例如，当用postMessage传送密码时，这个参数就显得尤为重要，必须保证它的值与这条包含密码的信息的预期接受者的origin属性完全一致，来防止密码被恶意的第三方截获。**如果你明确的知道消息应该发送到哪个窗口，那么请始终提供一个有确切值的targetOrigin，而不是\*。不提供确切的目标将导致数据泄露到任何对数据感兴趣的恶意站点。**

***transfer* 可选**

是一串和message 同时传递的 [Transferable](https://developer.mozilla.org/zh-CN/docs/Web/API/Transferable) 对象. 这些对象的所有权将被转移给消息的接收方，而发送一方将不再保有所有权。

[**The dispatched event**](https://developer.mozilla.org/zh-CN/docs/Web/API/Window/postMessage#the_dispatched_event)

执行如下代码, 其他window可以监听分发的message:

window.addEventListener("message", receiveMessage, false);

function receiveMessage(event)

{

// For Chrome, the origin property is in the event.originalEvent

// object.

// 这里不准确，chrome没有这个属性

// var origin = event.origin || event.originalEvent.origin;

var origin = event.origin

if (origin !== "http://example.org:8080")

return;

// ...

}

Copy to Clipboard

 message 的属性有:

**data**

从其他 window 中传递过来的对象。

**origin**

调用 postMessage  时消息发送方窗口的 [origin](https://developer.mozilla.org/en-US/docs/Origin) . 这个字符串由 协议、“://“、域名、“ : 端口号”拼接而成。例如 “https://example.org (隐含端口 443)”、“http://example.net (隐含端口 80)”、“http://example.com:8080”。请注意，这个origin不能保证是该窗口的当前或未来origin，因为postMessage被调用后可能被导航到不同的位置。

**source**

对发送消息的[窗口](https://developer.mozilla.org/en-US/docs/Web/API/Window)对象的引用; 您可以使用此来在具有不同origin的两个窗口之间建立双向通信。

[**安全问题**](https://developer.mozilla.org/zh-CN/docs/Web/API/Window/postMessage#security_concerns)

**如果您不希望从其他网站接收message，请不要为message事件添加任何事件侦听器。**这是一个完全万无一失的方式来避免安全问题。

如果您确实希望从其他网站接收message，请**始终使用origin和source属性验证发件人的身份**。 任何窗口（包括例如http://evil.example.com）都可以向任何其他窗口发送消息，并且您不能保证未知发件人不会发送恶意消息。 但是，验证身份后，您仍然应该**始终验证接收到的消息的语法**。 否则，您信任只发送受信任邮件的网站中的安全漏洞可能会在您的网站中打开跨网站脚本漏洞。

**当您使用postMessage将数据发送到其他窗口时，始终指定精确的目标origin，而不是\*。**恶意网站可以在您不知情的情况下更改窗口的位置，因此它可以拦截使用postMessage发送的数据。

[**示例**](https://developer.mozilla.org/zh-CN/docs/Web/API/Window/postMessage#example)

/\*

\* A窗口的域名是<http://example.com:8080>，以下是A窗口的script标签下的代码：

\*/

var popup = window.open(...popup details...);

// 如果弹出框没有被阻止且加载完成

// 这行语句没有发送信息出去，即使假设当前页面没有改变location（因为targetOrigin设置不对）

popup.postMessage("The user is 'bob' and the password is 'secret'",

"https://secure.example.net");

// 假设当前页面没有改变location，这条语句会成功添加message到发送队列中去（targetOrigin设置对了）

popup.postMessage("hello there!", "http://example.org");

function receiveMessage(event)

{

// 我们能相信信息的发送者吗? (也许这个发送者和我们最初打开的不是同一个页面).

if (event.origin !== "http://example.org")

return;

// event.source 是我们通过window.open打开的弹出页面 popup

// event.data 是 popup发送给当前页面的消息 "hi there yourself! the secret response is: rheeeeet!"

}

window.addEventListener("message", receiveMessage, false);

Copy to Clipboard

/\*

\* 弹出页 popup 域名是<http://example.org>，以下是script标签中的代码:

\*/

//当A页面postMessage被调用后，这个function被addEventListener调用

function receiveMessage(event)

{

// 我们能信任信息来源吗？

if (event.origin !== "http://example.com:8080")

return;

// event.source 就当前弹出页的来源页面

// event.data 是 "hello there!"

// 假设你已经验证了所受到信息的origin (任何时候你都应该这样做), 一个很方便的方式就是把event.source

// 作为回信的对象，并且把event.origin作为targetOrigin

event.source.postMessage("hi there yourself! the secret response " +

"is: rheeeeet!",

event.origin);

}

window.addEventListener("message", receiveMessage, false);

Copy to Clipboard

[**注意**](https://developer.mozilla.org/zh-CN/docs/Web/API/Window/postMessage#notes)

任何窗口可以在任何其他窗口访问此方法，在任何时间，无论文档在窗口中的位置，向其发送消息。 因此，用于接收消息的任何事件监听器**必须**首先使用origin和source属性来检查消息的发送者的身份。 **这不能低估：无法检查origin和source属性会导致跨站点脚本攻击。**

与任何异步调度的脚本（超时，用户生成的事件）一样，postMessage的调用者不可能检测到侦听由postMessage发送的事件的事件处理程序何时抛出异常。

分派事件的origin属性的值不受调用窗口中document.domain的当前值的影响。

仅对于IDN主机名，origin属性的值不是始终为Unicode或punycode; 在使用此属性时，如果您期望来自IDN网站的消息，则最大程度地兼容性检查IDN和punycode值。 这个值最终将始终是IDN，但现在你应该同时处理IDN和punycode表单。

当发送窗口包含 javascript: 或 data: URL时，origin属性的值是加载URL的脚本的

[**在扩展中使用window.postMessage**](https://developer.mozilla.org/zh-CN/docs/Web/API/Window/postMessage#%E5%9C%A8%E6%89%A9%E5%B1%95non-standard_inline%E4%B8%AD%E4%BD%BF%E7%94%A8window.postmessage)

window.postMessage可用于以chrome代码运行的JavaScript（例如，在扩展和特权代码中），但是分派事件的source属性总是为空作为安全限制。 （其他属性具有其期望值。）发送到位于chrome：URL的窗口的消息的targetOrigin参数当前被错误解释，使得将导致发送消息的唯一值为“\*”。 由于此值是不安全的，当目标窗口可以导航到其他地方的恶意网站，建议postMessage不用于与chrome：页面的沟通; 使用不同的方法（如打开窗口时的查询字符串）与chrome窗口进行通信。 最后，在文件中向页面发布消息：URL当前要求targetOrigin参数为“\*”。 file://不能用作安全限制; 这个限制可能会在将来被修改。

[**规范**](https://developer.mozilla.org/zh-CN/docs/Web/API/Window/postMessage#specification)

|  |  |  |
| --- | --- | --- |
| **规范** | **状态** | **状态** |
| [HTML Living Standard postMessage()](https://html.spec.whatwg.org/multipage/web-messaging.html#dom-window-postmessage) | Living Standard |  |