

windowOpen字符窗口的通信

<https://www.electronjs.org/docs/api/window-open>

⚙️ API 参考

简介

进程对象

支持的命令行开关

环境变量

Chrome 扩展支持

重要的API变更

自定义 DOM 元素

[File 对象](#)

[<webview> 标签](#)

[window.open 函数](#)

[BrowserWindowProProxy 对象](#)

Main Process 模块

app

autoUpdater

BrowserView



Bluetooth

Certifi

Certifi

体

Cooki

CPUU

崩溃

Custo

Deskt

Disp

事件

Exter

扩展

FileF

FileP

GPU

Input

IOPC

BrowserWindow

contentTracing

对话框

IpcM
IpcRe
IpcRe

window.open 函数

打开一个新窗口并加载 URL。

当调用 `window.open` 以在网页中创建新窗口时，将为 `url` 创建一个新的 `BrowserWindow` 实例，并返回一个代理至 `window.open` 以让页面对其进行有限的控制。

该代理具有有限的标准功能，与传统网页兼容。要完全控制新窗口，你应该直接创建一个 `BrowserWindow`。

默认情况下，新创建的 `BrowserWindow` 将继承父窗口的选项。若要重写继承的选项，可以在 `features` 字符串中设置它们。

`window.open(url[, frameName][, features])`

- `url` String
- `frameName` String (可选)
- `features` String (可选)

Returns `BrowserWindowProxy` - 创建一个新窗口，并返回一个 `BrowserWindowProxy` 类的实例。

`features` 字符串遵循标准浏览器的格式，但每个 feature 必须是 `BrowserWindow` 选项中的字段。These are the features you can set via `features` string: `zoomFactor`, `nodeIntegration`, `preload`, `javascript`, `contextIsolation`, `webviewTag`.

```
index.html JS renderer.js ×
JS renderer.js ▶ ⌂ openNewWindow
1 // WEDEVIEW 大纲
2
3 const wb = document.querySelector('#wb');
4 const loading = document.querySelector("#loading");
5
6 function openNewWindow() {
7   window.open('https://www.baidu.com', "baidu")
8 }
9
10 wb.addEventListener("did-start-loading", ()=> {
11
12
13
14
15
16 })
```

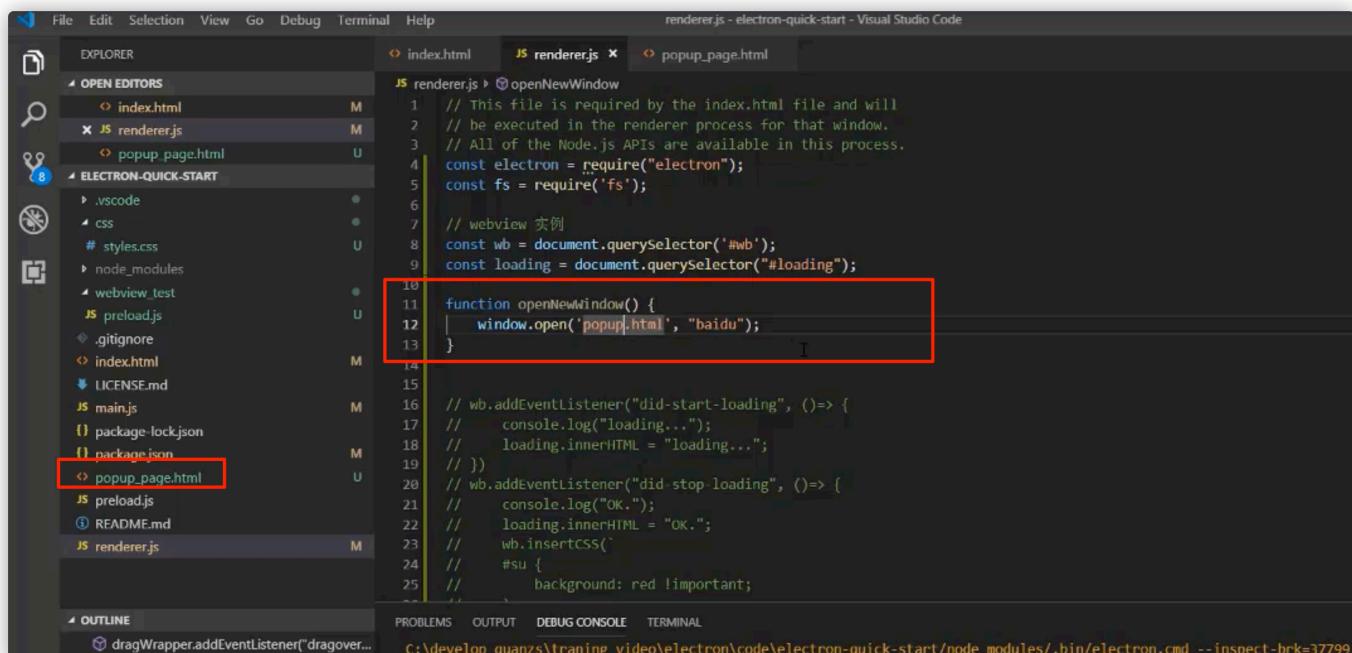
子窗口向父窗口传递消息

```
window.opener.postMessage(message, targetOrigin)
```

- message String
- targetOrigin String

将消息发送给指定来源的父窗口，如果未指定来源则发送给 *，即所有窗口。

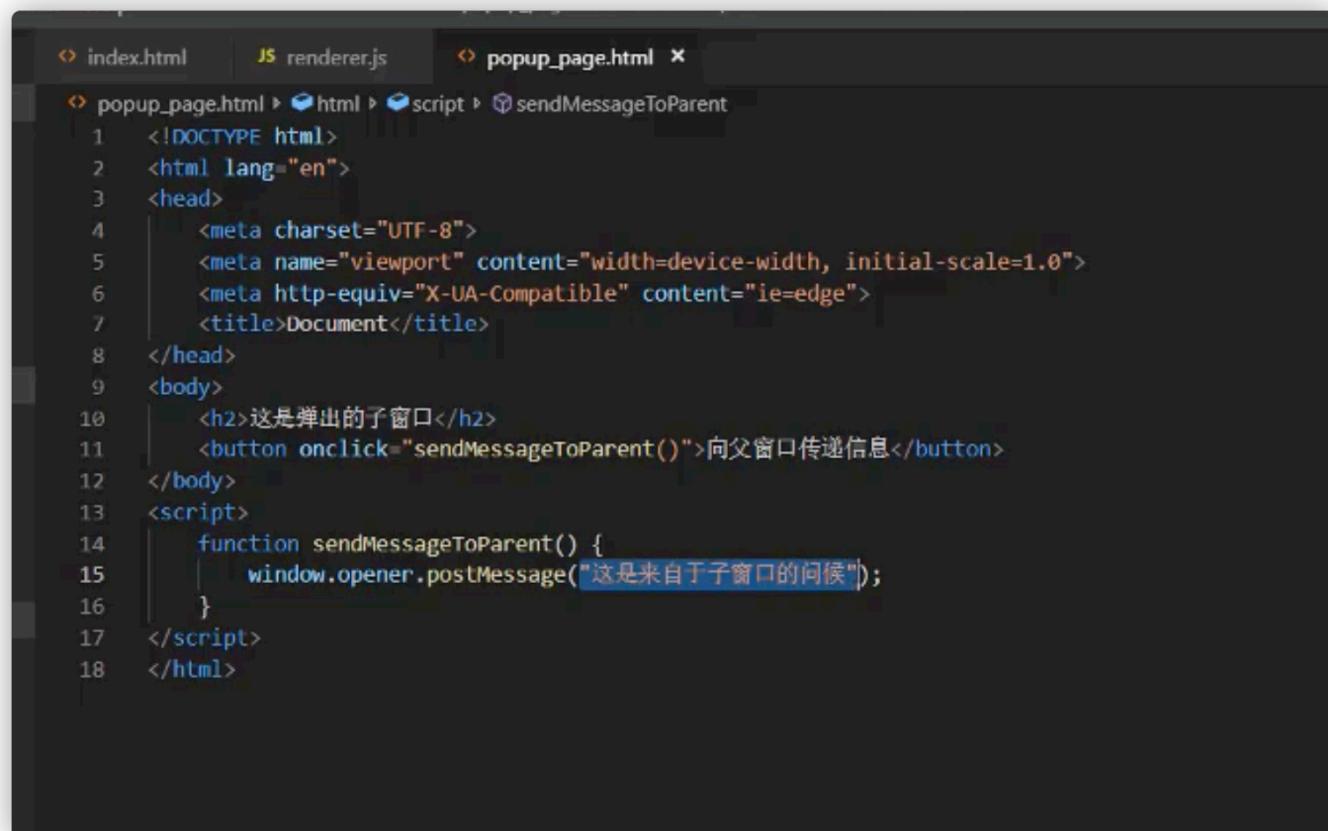
新建一个子窗口



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files like index.html, renderer.js, and popup_page.html.
- Editor:** The renderer.js file is open, containing code to open a new window. A red box highlights the following line:

```
11 function openNewWindow() {  
12     window.open('popup.html', 'baidu');  
13 }
```
- Terminal:** The command `C:\develop_quanz\traning_video\electron\code\electron-quick-start\node_modules/.bin/electron.cmd --inspect-brk=37799` is visible at the bottom.



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files like index.html, renderer.js, and popup_page.html.
- Editor:** The popup_page.html file is open, containing HTML and JavaScript code. A blue box highlights the following line of JavaScript:

```
15     window.opener.postMessage("这是来自于子窗口的问候");
```

The screenshot shows the Visual Studio Code interface with the title bar "terminal Help" and "renderer.js - electron-quick-start - Visual Studio Code". The code editor displays the "renderer.js" file, which contains the following code:

```
// This file is required by the index.html file and will
// be executed in the renderer process for that window.
// All of the Node.js APIs are available in this process.
const electron = require("electron");
const fs = require('fs');

// webview 实例
const wb = document.querySelector('#wb');
const loading = document.querySelector("#loading");

function openNewWindow() {
    window.open('popup_page.html', "popup");
}

window.addEventListener("message", (msg) => {
    console.log("接收到的消息: ", msg);
})

// wb.addEventListener("did-start-loading", ()=> {
//     console.log("loading...");
//     loading.innerHTML = "loading...";
// })
// wb.addEventListener("did-stop-loading", ()=> {
//     console.log("OK.");
// })
```

The screenshot shows a desktop application window titled "Hello World!". Inside, a modal window titled "弹出子窗口" (Pop-up Sub-Window) is displayed. The modal contains the text "File对象" (File Object) and "往这里拖文件" (Drag files here). In the background, the main window has a pink sidebar labeled "弹出" (Pop-up). The top right of the screen shows the "千锋教育" logo.

The developer tools are open, showing the "Variables" panel with the variable "msg" expanded, and the "Console" tab with the following log message:

```
这是弹出的子窗口
向父窗口传递信息
接收到的消息:
Event {isTrusted: false, data: "这是来自于子窗口的问候", origin: "file:///", source: BrowserWindowProxy}
```

也可以穿json

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <h2>这是弹出的子窗口</h2>
    <button onclick="sendMessageToParent()">向父窗口传递信息</button>
</body>
<script>
    function sendMessageToParent() {
        window.opener.postMessage({
            type: 1,
            message: "这是来自于子窗口的问候"
        });
    }
</script>
</html>
```

关闭子窗口

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hello World!</title>
    <link rel="stylesheet" href="css/styles.css" />
</head>
<body>
    <div>
        <h2>弹出子窗口</h2>
        <button onclick="openNewWindow()">弹出</button>
        <button onclick="closeWindow()">关闭子窗口</button>
    </div>
    <!-- <div>
        <h2>Webview</h2>
        <span id="loading"></span>
        <webview id="wb" src="https://www.baidu.com/" style="height: 400px;" preload=".//webview_test/pr
    </div> -->
    <div class="for_file_drag" id="drag_test">
        <h2>File对象</h2>
        <span>往这里拖文件</span>
    </div>
    ...
</body>
</html>
```

BrowserWindowProxy

<https://www.electronjs.org/docs/api/browser-window-proxy>

The screenshot shows a portion of the Electron API Reference documentation. At the top left, it says "app | Electron". Below that, there's some partially visible text starting with "ate". In the center, there's a large icon of a gear followed by the text "API 参考". To the right of the gear icon, there's a vertical sidebar with letters B through F listed vertically. Below the gear icon, there's a list of links: "简介", "进程对象", "支持的命令行开关", "环境变量", "Chrome 扩展支持", and "重要的API变更". Further down, there's a section titled "自定义 DOM 元素". Under this section, there's a list of items: "File 对象", "<webview> 标签", "window.open 函数", and "BrowserWindowProxy 对象". The "BrowserWindowProxy 对象" link is highlighted with a red rectangular box around it. At the very bottom, there's a footer section titled "Main Process 模块".

app | Electron

ate

API 参考

简介

进程对象

支持的命令行开关

环境变量

Chrome 扩展支持

重要的API变更

自定义 DOM 元素

File 对象

<webview> 标签

window.open 函数

BrowserWindowProxy 对象

Main Process 模块

The screenshot shows the Visual Studio Code interface with the file `renderer.js` open. The code is a JavaScript file that handles window operations. A red box highlights the `openNewWindow` function, which opens a new window named `popup` from the URL `popup_page.html`. The code also includes event listeners for messages and did-start-loading events.

```
// This file is required by the index.html file and will
// be executed in the renderer process for that window.
// All of the Node.js APIs are available in this process.
const electron = require("electron");
const fs = require('fs');

// webview 实例
const wb = document.querySelector('#wb');
const loading = document.querySelector("#loading");

let subwin;
function openNewWindow() {
    subwin = window.open('popup_page.html', "popup");
}

function closeWindow() {
    subwin.close();
}

window.addEventListener("message", (msg) => {
    console.log("接收到的消息: ", msg);
})

// wb.addEventListener("did-start-loading", ()=> {
```