

1.webview

The screenshot shows a mobile-style interface for the Electron API Reference. At the top, there's a header with the Electron logo and the text "API 参考". Below the header, there's a navigation bar with several items: "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". A large section title "自定义 DOM 元素" is centered below the navigation. At the bottom, there's a list of four items: "File 对象", "**<webview> 标签**" (which is highlighted with a red border), "window.open 函数", and "BrowserWindowProProxy 对象".

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

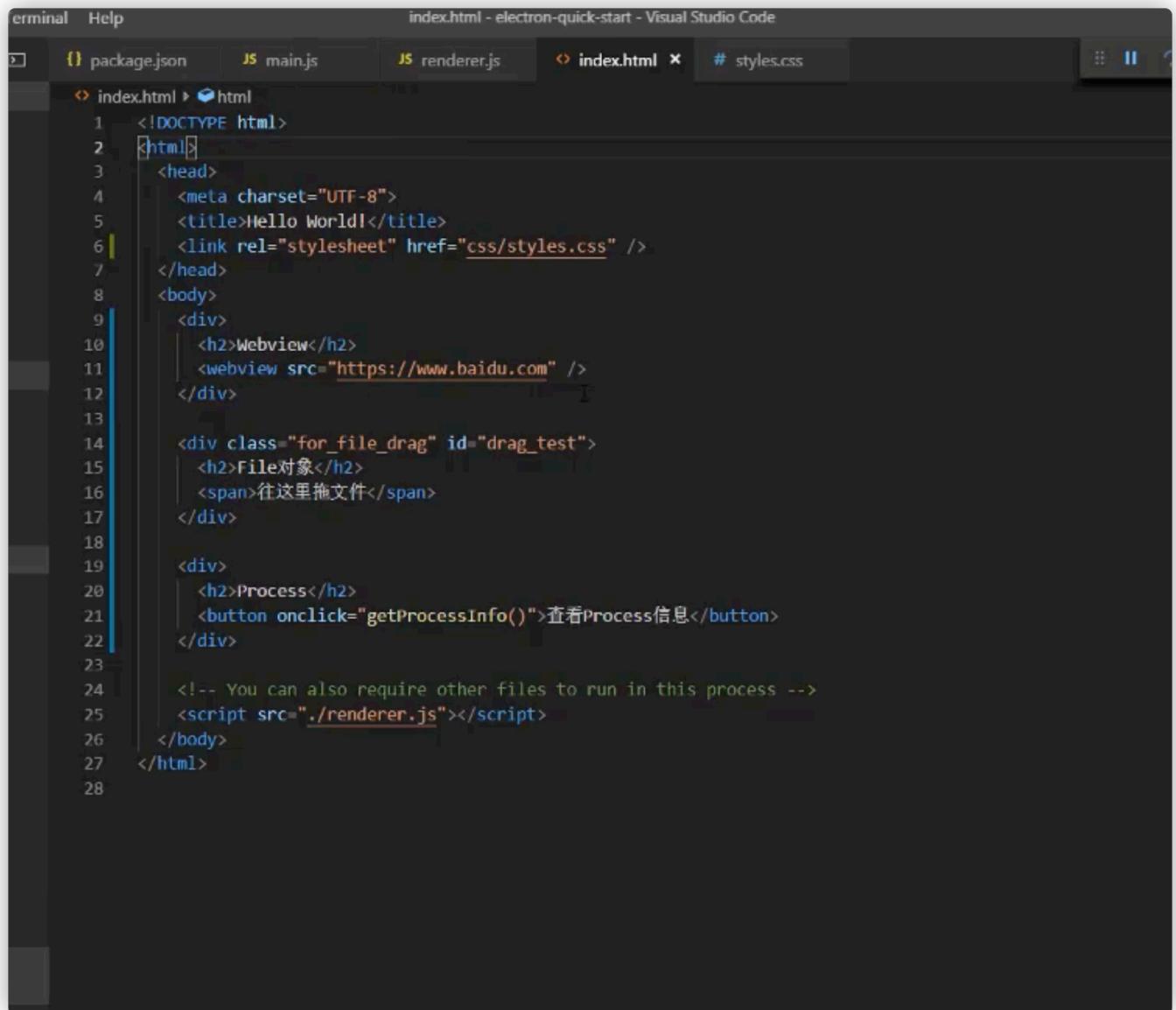
<webview> 标签

window.open 函数

BrowserWindowProProxy 对象

Main Process 模块

<https://www.electronjs.org/docs/api/webview-tag>



The screenshot shows the Visual Studio Code interface with the 'index.html' tab selected. The code editor displays the following HTML content:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Hello World!</title>
    <link rel="stylesheet" href="css/styles.css" />
  </head>
  <body>
    <div>
      <h2>Webview</h2>
      <webview src="https://www.baidu.com" />
    </div>

    <div class="for_file_drag" id="drag_test">
      <h2>File对象</h2>
      <span>往这里拖文件</span>
    </div>

    <div>
      <h2>Process</h2>
      <button onclick="getProcessInfo()">查看Process信息</button>
    </div>

    <!-- You can also require other files to run in this process -->
    <script src="./renderer.js"></script>
  </body>
</html>
```

The screenshot shows the Visual Studio Code interface with the file `styles.css` open. The code defines a class `.for_file_drag` and a selector `webview` with the following styles:

```
1 .for_file_drag {  
2     width: 100%;  
3     height: 400px;  
4     background: pink;  
5 }  
6  
7 webview {  
8     width: 100%;  
9     height: 400px;  
10    display: block;  
11 }  
12  
13
```

A red box highlights the `webview` selector and its properties.

The screenshot shows a section from the Electron documentation titled "Enabling". It explains that by default, the `webview` tag is disabled in Electron >= 5. To enable it, you need to set the `webviewTag` webPreferences option when creating a `BrowserWindow`. A red box highlights this text.

By default the `webview` tag is disabled in Electron >= 5. You need to enable the tag by setting the `webviewTag` webPreferences option when constructing your `BrowserWindow`. For more information see the [BrowserWindow constructor docs](#).

The screenshot shows the Visual Studio Code interface with the file `main.js` open. The code creates a `BrowserWindow` with the following configuration:

```
1 // Modules to control application life and create native browser window  
2 const {app, BrowserWindow} = require('electron')  
3 const path = require('path')  
4  
5 // Keep a global reference of the window object, if you don't, the window will  
6 // be closed automatically when the JavaScript object is garbage collected.  
7 let mainWindow  
8  
9 function createWindow () {  
10     // Create the browser window.  
11     mainWindow = new BrowserWindow({  
12         width: 800,  
13         height: 600,  
14         webPreferences: {  
15             preload: path.join(__dirname, 'preload.js'),  
16             webviewTag: true,  
17             nodeIntegration: true // 这个属性非常重要，用户的require和process等变量的使用需要事先设定上这个属性  
18         }  
19     })  
20  
21     mainWindow.webContents.on("did-finish-load", ()=> {  
22         console.log("****did-finish-load");  
23     })  
24 }
```

A red box highlights the `webviewTag: true` line, and a red arrow points to the `preload` line. The terminal at the bottom shows the command used to run the application.

C:\develop_quanzs\traning_video\electron\code\electron-quick-start\node_modules/.bin/electron.cmd --inspect-brk=10272 .



index.html - electron-quick-start - Visual Studio Code

```
final Help index.html - electron-quick-start - Visual Studio Code
{} package.json JS main.js JS preload.js JS renderer.js < index.html x # styles.css :: || ⌂
< index.html > html > body > div > webview#wb
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Hello World!</title>
6      <link rel="stylesheet" href="css/styles.css" />
7  </head>
8  <body>
9      <div>
10         <b>Webview</b>
11         <span id="loading"></span>
12         <webview id="wb" src="https://www.baidu.com/" style="height: 400px;" />
13     </div>
14     <div class="for_file_drag" id="drag_test">
15         <h2>File对象</h2>
16         <span>往这里拖文件</span>
17     </div>
18
19     <div>
20         <h2>Process</h2>
21         <button onclick="getProcessInfo()">查看Process信息</button>
22     </div>
23
24
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

C:\develop_quanz\traning_video\electron\code\electron-quick-start\node_modules/.bin/electron.cmd --inspect-brk-

The screenshot shows the Visual Studio Code interface with the "index.html" file open. The code editor displays the HTML structure, including a webview element that loads the Baidu homepage. A red box highlights the content area of the webview element in the DOM tree. The status bar at the bottom shows the command to start the Electron application with debugging enabled.

oview

terminal Help renderer.js - electron-quick-start - Visual Studio Code

package.json main.js preload.js render.js ✘ index.html styles.css

```
JS renderer.js > wb.addEventListener("did-stop-loading") callback
1 // This file is required by the index.html file and will
2 // be executed in the renderer process for that window.
3 // All of the Node.js APIs are available in this process.
4 const electron = require("electron");
5 const fs = require('fs');

6 // webview 实例
7 const wb = document.querySelector('#wb');
8 const loading = document.querySelector("#loading");
9 wb.addEventListener("did-start-loading", ()=> {
10   console.log("loading...");
11   loading.innerHTML = "loading...";
12 })
13 wb.addEventListener("did-stop-loading", ()=> {
14   console.log("OK.");
15   loading.innerHTML = "OK.";
16 })
17

18 // File对象 实例
19 const dragWrapper = document.getElementById("drag_test");
20 dragWrapper.addEventListener("drop", (e) => {
21   e.preventDefault();
22   const files = e.dataTransfer.files;
23   if(files && files.length > 0) {
24     PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
C:\develop_quanz\traning_video\electron\code\electron-quick-start\node_modules/.bin/electron.cmd --inspect
Debugger listening on ws://127.0.0.1:43452/e38a9e4c-6692-440e-8277-0675384c637d
For help, see: https://nodejs.org/en/docs/inspector
```

```
<script>
  onload = () => {
    const webview = document.querySelector('webview')
    const indicator = document.querySelector('.indicator')

    const loadstart = () => {
      indicator.innerText = 'loading...'
    }

    const loadstop = () => {
      indicator.innerText = ''
    }

    webview.addEventListener('did-start-loading', loadstart)
    webview.addEventListener('did-stop-loading', loadstop)
  }
</script>
```

preload

在 `<webview>` 标签中，可以使用 `preload` 属性来指定一个脚本，在访客页加载之前先执行。该脚本的 URL 必须是 `file` 或 `asar` 之一，因为在访客页中，它是通过“内部”的 `require` 去加载的。

当访客页没有 node integration，这个脚本仍然有能力去访问所有的 Node APIs，但是当这个脚本执行完成之后，通过 Node 注入的全局对象（global objects）将会被删除。

注意：在为 `will-attach-webview` 事件指定 `webPreferences` 时，这个选项将作为 `preloadURL` 出现，而不是 `preload`。

httpreferrer

使用百度网址，使用preload在加载的时候一些事件或者脚本



The screenshot shows the Chrome DevTools Elements tab for the Baidu homepage (`https://www.baidu.com`). The page content includes the Baidu logo, search bar, and various links. In the Elements panel, a red box highlights the `span.bg_s_btn_wr` element, which contains the `preload` attribute. The DevTools sidebar on the right shows the `Styles` tab with CSS rules for the `.bg_s_btn` class.

```
<span class="bg_s_btn_wr" 100 x 37>
  <span class="bg_s_btn">
    <input type="submit" id="su" value="百度一下" class="bg_s_btn">
  </span>
</span>
```

```
.bg_s_btn {
  width: 100px;
  height: 36px;
  color: #fff;
  font-size: 15px;
  letter-spacing: 1px;
  background-color: #303030;
  border-bottom: 1px solid #303030;
  outline: none;
  border-radius: 0;
  -webkit-appearance: none;
}
```

```
.bg_s_btn {
  width: 95px;
  height: 32px;
  padding-top: 2px \ 0;
  font-size: 14px;
  background-color: #303030;
  background-image: url(https://ssl);
  background-repeat: no-repeat;
  background-position: center;
  cursor: pointer;
}
```

```
.bg {
  background-image: url(https://ssl);
  background-repeat: no-repeat;
  background-position: center;
}
```

```
input {
  border: 0;
  padding: 0;
}
```

```
input[type="submit"] {
  padding: 0 6px;
}
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Hello World!</title>
<link rel="stylesheet" href="css/styles.css" />
</head>
<body>
<div>
<h2>Webview</h2>
<span id="loading"></span>
<webview id="wb" src="https://www.baidu.com/" style="height: 400px; preload="./preload.js" />
</div>
<div class="for_file_drag" id="drag_test">
<h2>File对象</h2>
<span>往这里拖文件</span>
</div>
<div>
<h2>Process</h2>
<button onclick="getProcessInfo()">查看Process信息</button>
</div>

```

File Edit Selection View Go Debug Terminal Help

EXPLORER

- OPEN EDITORS
 - package.json M
 - main.js M
 - renderer.js M
 - LICENSE.md
- JS preload.js webview_test
 - index.html M
 - # styles.css U
- ELECTRON-QUICK-START
 - .vscode
 - css
 - # styles.css
 - node_modules
 - webview_test
 - JS preload.js

PREBLEDS - electron-quick-start - Visual Studio Code

```
1 setTimeout(()=>{
2   alert(document.querySelector('.index-logo-src').src);
3 }, 5000);
```

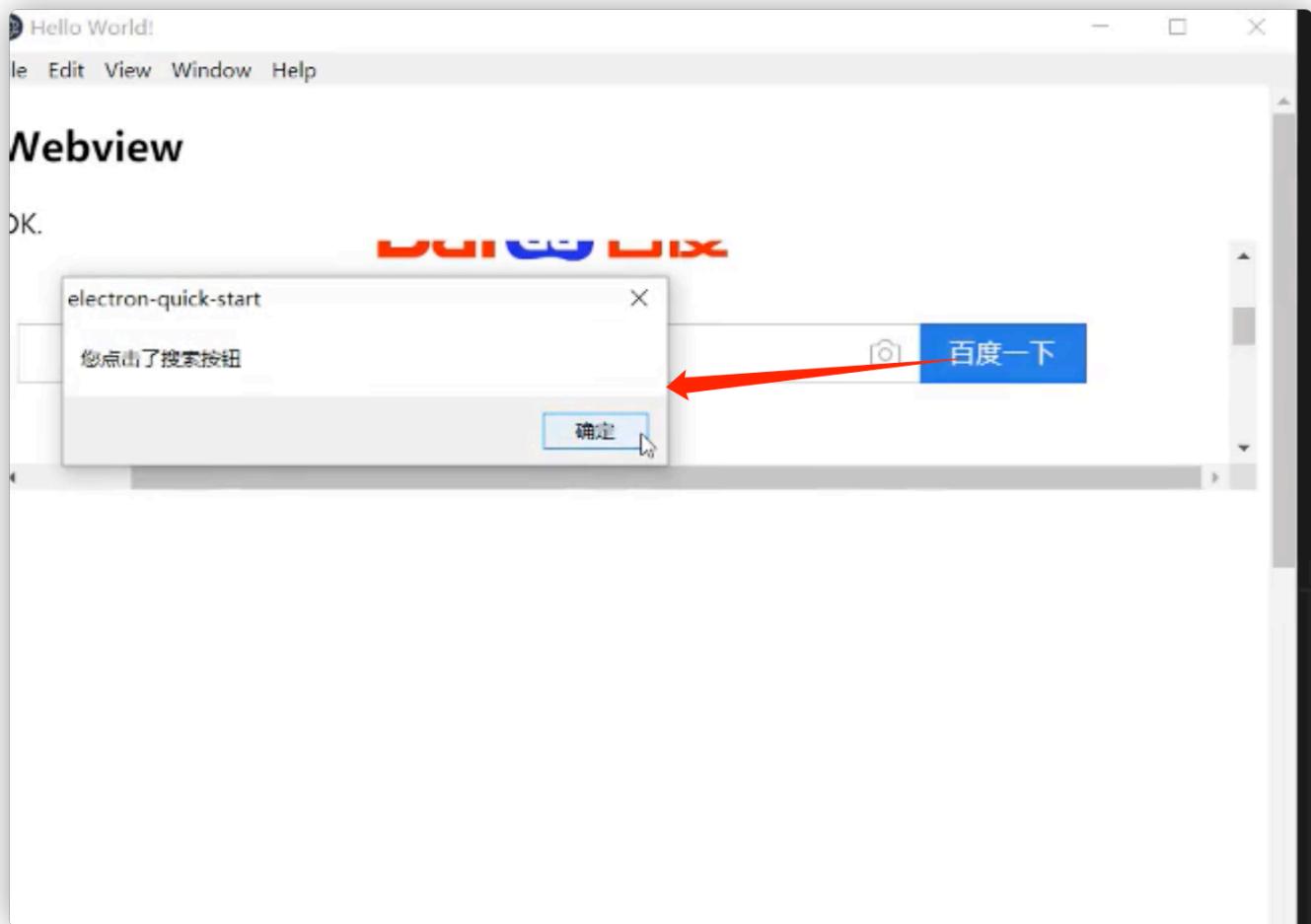
File Edit Selection View Go Debug Terminal Help

EXPLORER

- OPEN EDITORS
 - package.json M
 - main.js M
 - renderer.js M
 - LICENSE.md
- JS preload.js webview_test
 - index.html M
 - # styles.css U
- ELECTRON-QUICK-START
 - .vscode
 - css
 - # styles.css
 - node_modules
 - webview_test
 - JS preload.js

PREBLEDS - electron-quick-start - Visual Studio Code

```
1 setTimeout(()=>{
2   alert(document.querySelector('.index-logo-src').src);
3   document.querySelector('#su').onclick = () => {
4     alert("您点击了搜索按钮");
5   }
6 }, 5000);
```



insertcss

返回 `String` - 访客页的用户代理。

`<webview>.insertCSS(css)`

* `css` `String`

向访客页注入CSS。

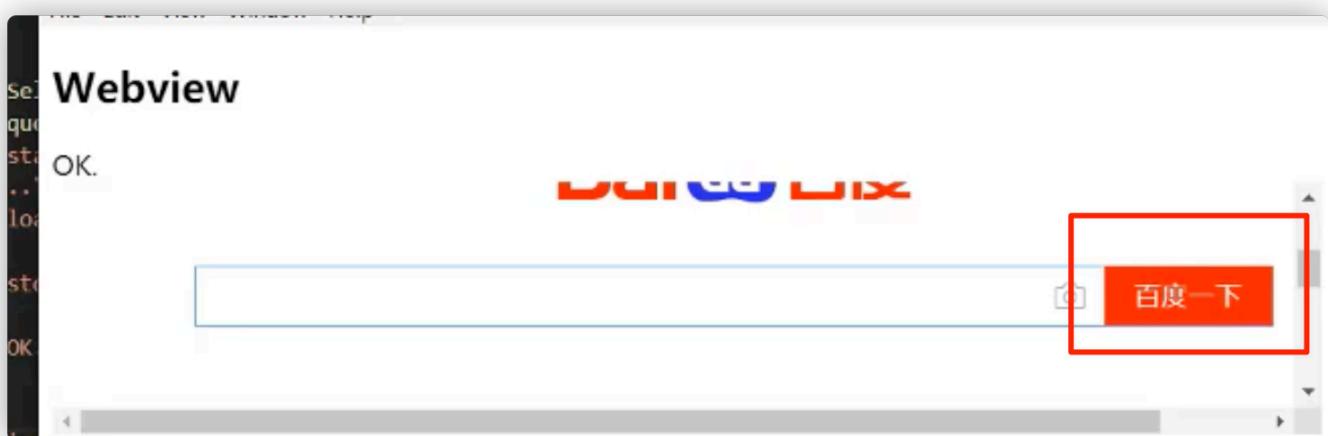
`<webview>.executeJavaScript(code[, userGesture, callback])`

- * `code` `String`
- * `userGesture` `Boolean` (可选) - 默认为 `false`。
- * `callback` `Function` (可选) - 在脚本被执行后被调用。
 - `result` `Any`

Returns `Promise<any>` - A promise that resolves with the result of the executed code or is rejected if the result of the code is a re

```
// 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");
wb.addEventListener("did-start-loading", ()=> {
    console.log("loading...");
    loading.innerHTML = "loading...";
})
wb.addEventListener("did-stop-loading", ()=> {
    console.log("OK.");
    loading.innerHTML = "OK.";
    wb.insertCSS(`#su {
        background: red !important;
    }
`)}
)
// 对外暴露一些 API
```



也可以注入js

[`<webview>.executeJavaScript\(code\[, userGesture\]\)`](#)

- `code` String
- `userGesture` Boolean (可选) - 默认为 `false`。

Returns `Promise<any>` - A promise that resolves with the result of the executed code or is rejected if the result of the code is a rejected promise.

在页面中执行 `code`。如果设置了 `userGesture`，它将在页面中创建用户手势上下文。像 `requestFullScreen` 这样的需要用户操作的HTML API可以利用这个选项来实现自动化。

Terminal Help renderer.js - electron-quick-start - Visual Studio Code

```
 package.json main.js renderer.js LICENSE.md preload.js index.html
JS renderer.js > wb.addEventListener("did-stop-loading") callback
M 18 #su {
M 19     background: red !important;
M 20 }
M 21 `)
U 22 wb.executeJavascript(`
M 23     setTimeout(()=>{
M 24         alert(document.getElementById('su').value);
U 25     }, 2000);
U 26 `)
M 27
M 28 })
U 29
U 30 // File对象 实例
M 31 const dragWrapper = document.getElementById("drag_test");
M 32 dragWrapper.addEventListener("drop", (e) => {
M 33     e.preventDefault();
M 34     const files = e.dataTransfer.files;
U 35     if(files && files.length > 0) {
```

打开devtools

<webview>.openDevTools() ⓘ

Opens a DevTools window for guest page.