

browseWindow

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

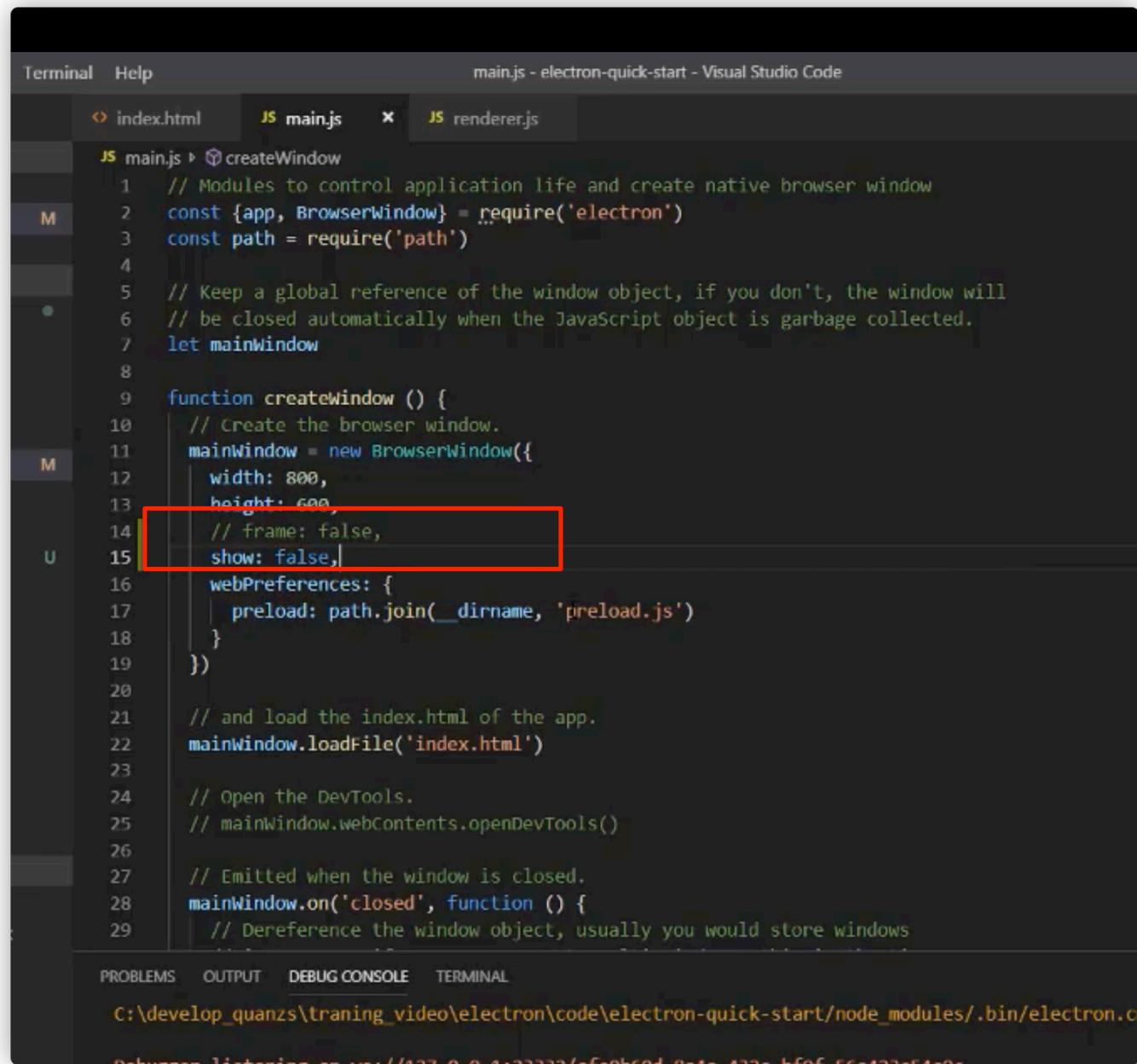
使用 ready-to-show 事件

EN

在加载页面时，渲染进程第一次完成绘制时，会发出 ready-to-show 事件。在此事件后显示窗口将没有视觉闪烁：

```
const { BrowserWindow } = require('electron')
let win = new BrowserWindow({ show: false })
win.once('ready-to-show', () => {
  win.show()
})
```

Copy



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

- Terminal:** Help
- File:** main.js - electron-quick-start - Visual Studio Code
- Content:** The code defines a main window with initial settings like width, height, and web preferences, and sets its visibility to false before showing it.
- Line 15:** `show: false,` is highlighted with a red rectangle.
- Status Bar:** C:\develop_quanzs\traning_video\electron\code\electron-quick-start\node_modules/.bin/electron.a
- Bottom Status:** Debugging listening on ws://127.0.0.1:22222/afec0b0d-8a4e-422c-bf0f-56a423ef4e0c

```
// Modules to control application life and create native browser window
const {app, BrowserWindow} = require('electron')
const path = require('path')

// Keep a global reference of the window object, if you don't, the window will
// be closed automatically when the JavaScript object is garbage collected.
let mainWindow

function createWindow () {
  // Create the browser window.
  mainWindow = new BrowserWindow({
    width: 800,
    height: 600,
    // frame: false,
    show: false,
    webPreferences: {
      preload: path.join(__dirname, 'preload.js')
    }
  })

  // and load the index.html of the app.
  mainWindow.loadFile('index.html')

  // Open the DevTools.
  // mainWindow.webContents.openDevTools()

  // Emitted when the window is closed.
  mainWindow.on('closed', function () {
    // Dereference the window object, usually you would store windows
  })
}

// and load the index.html of the app.
createWindow()
```

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows files: index.html, JS main.js (selected), JS renderer.js.
- Code Editor:** Displays the content of main.js. A red box highlights the following code block:

```
mainWindow.once("ready-to-show", () => {
  mainWindow.show();
})
```
- Bottom Status Bar:** Shows the path: C:\develop_quanzs\traning_video\electron\code\electron-quick-start\node_modules/.bin/electron

设置 backgroundColor

EN

对于一个复杂的应用，`ready-to-show` 可能发出的太晚，会让应用感觉缓慢。在这种情况下，建议立刻显示窗口，并使用接近应用程序背景的 `backgroundColor`

```
const { BrowserWindow } = require('electron')

let win = new BrowserWindow({ backgroundColor: '#2e2c29' })
win.loadURL('https://github.com')
```

Copy

请注意，即使是使用 `ready-to-show` 事件的应用程序，仍建议使用设置 `backgroundColor` 使应用程序感觉更原生。

ninal Help

main.js - electron-quick-start - Visual Studio Code

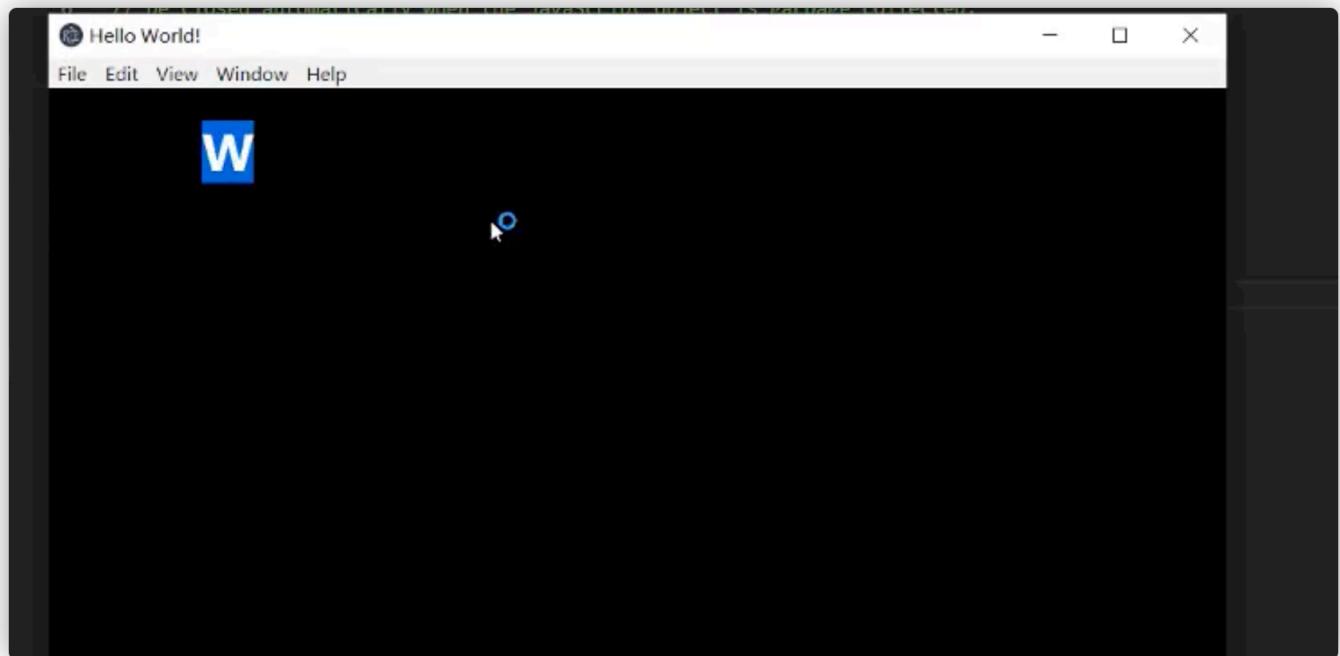
index.html

JS main.js

JS renderer.js

JS main.js > ⚡ createWindow

```
4
5  // Keep a global reference of the window object, if you don't
6  // be closed automatically when the JavaScript object is
7  let mainWindow
8
9  function createWindow () {
10    // Create the browser window.
11    mainWindow = new BrowserWindow({
12      width: 800,
13      height: 600,
14      // frame: false,
15      // show: false,
16      backgroundColor: "#000000",
17      webPreferences: {
18        preload: path.join(__dirname, 'preload.js')
19      }
20    })
21
22    // and load the index.html of the app.
23    mainWindow.loadFile('index.html')
24
25    // Open the DevTools.
```



父子窗口

EN

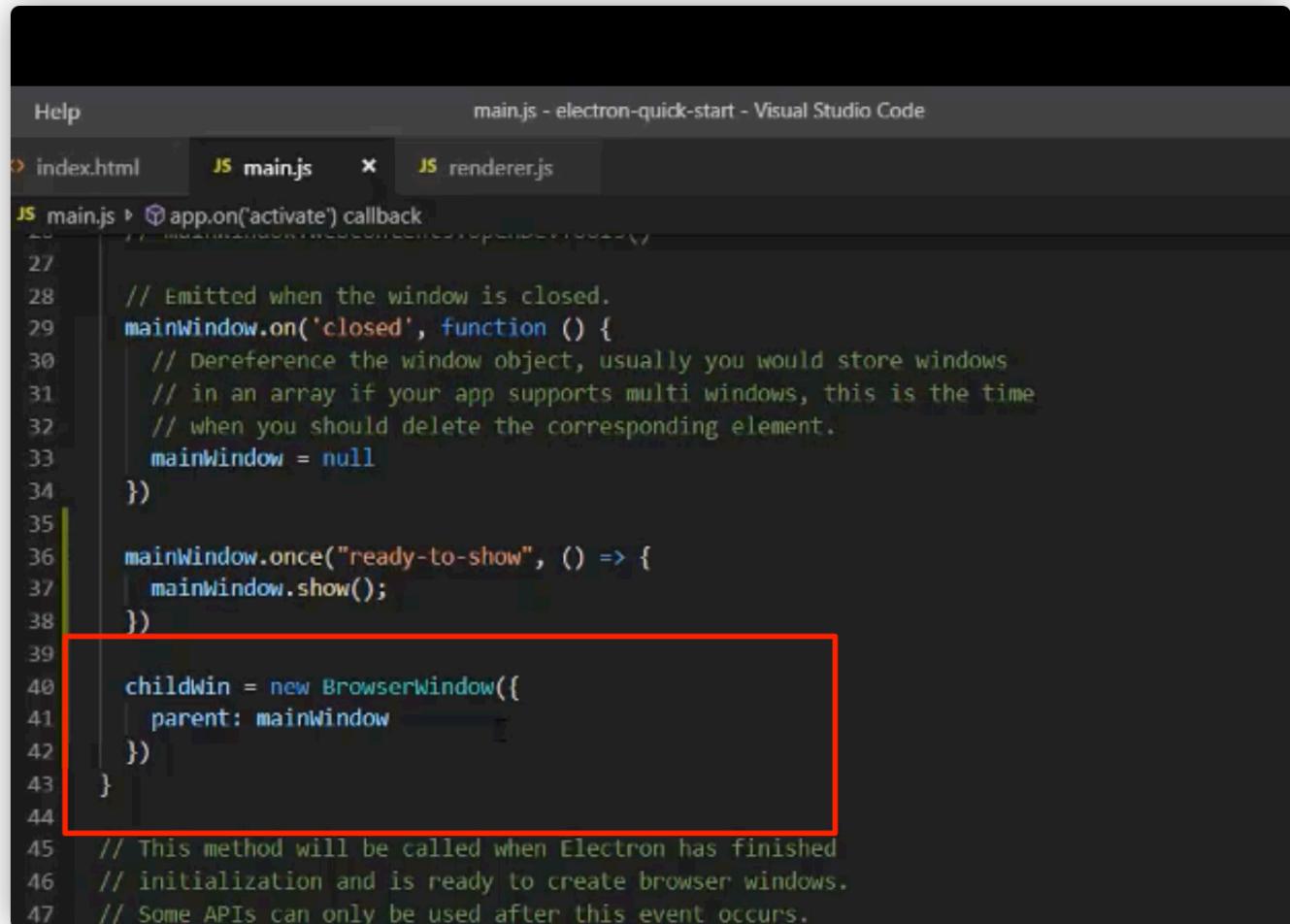
通过使用 `parent` 选项，你可以创建子窗口：

```
const { BrowserWindow } = require('electron')

let top = new BrowserWindow()
let child = new BrowserWindow({ parent: top })
child.show()
top.show()
```

Copy

`child` 窗口将总是显示在 `top` 窗口的顶部。



Help main.js - electron-quick-start - Visual Studio Code

index.html JS main.js X JS renderer.js

JS main.js > ⓘ app.on('activate') callback

```
27
28 // Emitted when the window is closed.
29 mainWindow.on('closed', function () {
30     // Dereference the window object, usually you would store windows
31     // in an array if your app supports multi windows, this is the time
32     // when you should delete the corresponding element.
33     mainWindow = null
34 })
35
36 mainWindow.once("ready-to-show", () => {
37     mainWindow.show();
38 })
39
40 childwin = new BrowserWindow({
41     parent: mainWindow
42 })
43 }
44 // This method will be called when Electron has finished
45 // initialization and is ready to create browser windows.
46 // Some APIs can only be used after this event occurs.
```

模态窗口

EN

模态窗口是禁用父窗口的子窗口，创建模态窗口必须设置 `parent` 和 `modal` 选项：

```
const { BrowserWindow } = require('electron')

let child = new BrowserWindow({ parent: top, modal: true, show: false })
child.loadURL('https://github.com')
child.once('ready-to-show', () => {
    child.show()
})
```

Copy

File Help

main.js - electron-quick-start - Visual Studio Code

index.html main.js renderer.js

JS main.js ↗ createWindow

```
27
28 // Emitted when the window is closed.
29 mainWindow.on('closed', function () {
30     // Dereference the window object, usually you would store it
31     // in an array if your app supports multiple windows, this is the
32     // time when you should delete the corresponding element.
33     mainWindow = null
34 })
35
36 mainWindow.once("ready-to-show", () => {
37     mainWindow.show();
38 })
39
40 childWin = new BrowserWindow({
41     parent: mainWindow,
42     modal: true
43 })
44
45
46 // This method will be called when Electron has finished
47 // initialization and is ready to create browser windows.
48 // Some APIs can only be used after this event occurs.
```

设置弹出位置样式

- 选项 Object (可选)

- `width` Integer (可选) - 窗口的宽度, 单位为像素。默认为 `800` .
- `height` Integer(可选) - 窗口的高度, 单位为像素。默认为 `600` .
- `x` Integer (可选) (如果 `y` 存在时**必填**) - 窗口相对于屏幕左侧的偏移位置. 默认居中.
- `y` Integer (可选) (如果 `x` 存在时**必填**) - 窗口相对于屏幕顶部的偏移位置. 默认居中.
- `useContentSize` Boolean (可选) - `width` 和 `height` 将设置为 web 页面的尺寸(译注: 不包含边框), 这意味着窗口的尺寸将包括窗口边框的大小, 稍微会大一点。默认值为 `false` .
- `center` Boolean (可选) - 窗口在屏幕居中.
- `minWidth` Integer (可选) - 窗口的最小宽度, 默认值为 `0` .
- `minHeight` Integer (可选) - 窗口的最小高度, 默认值为 `0` .
- `maxWidth` Integer (可选) - 窗口的最大宽度, 默认无限制.
- `maxHeight` Integer (可选) - 窗口的最大高度, 默认无限制.
- `resizable` Boolean (可选) - 窗口是否可以改变尺寸. 默认值为 `true` .
- `movable` Boolean (可选) - 窗口是否可以移动. 在 Linux 中无效. 默认值为 `true` .
- `minimizable` Boolean (可选) - 窗口是否可以最小化. 在 Linux 中无效. 默认值为 `true` .
- `maximizable` Boolean (可选) - 窗口是否可以最大化. 在 Linux 中无效. 默认值为 `true` .
- `closable` Boolean (可选) - 窗口是否可以关闭. 在 Linux 中无效. 默认值为 `true` .
- `focusable` Boolean (可选) - 窗口是否可以聚焦. 默认值为 `true` 。在 Windows 中设置 `focusable: false` 也意味着 `skipTaskbar: true` . 在 Linux 中设置 `focusable: false` 时窗口停止与 wm 交互, 并且窗口将始终置顶。
- `alwaysOnTop` Boolean (可选) - 窗口是否永远在别的窗口的上面. 默认值为 `false` .
- `fullscreen` Boolean (可选) - 窗口是否全屏. 当明确设置为 `false` 时, 在 macOS 上全屏的按钮将被隐藏或禁用. 默认值为 `false` .
- `fullscreenable` Boolean (可选) - 窗口是否可以进入全屏状态. 在 macOS 上, 最大化/缩放按钮是否可用. 默认值为 `true` .
- `simpleFullscreen` Boolean (可选) - 在 macOS 上使用 pre-Lion 全屏. 默认为 `false` .
- `skipTaskbar` Boolean (可选) - 是否在任务栏中显示窗口. 默认值为 `false` .
- `kiosk` Boolean (可选) - kiosk 模式. 默认值为 `false` .
- `title` String(可选) - 默认窗口标题默认为 "Electron" 。如果由 `loadURL()` 加载的HTML文件中含有标签 `<title>` 属性将被忽略。

terminal Help

main.js - electron-quick-start - Visual Studio Code



index.html

JS main.js



JS renderer.js

```
JS main.js ▶ ⚡createWindow
27
28     // Emitted when the window is closed.
29     mainWindow.on('closed', function () {
30         // Dereference the window object, usually you would store windows
31         // in an array if your app supports multi windows, this is the time
32         // when you should delete the corresponding element.
33         mainWindow = null
34     })
35
36     mainWindow.once("ready-to-show", () => {
37         mainWindow.show();
38     })
39
40     childWin = new BrowserWindow({
41         parent: mainWindow,
42         modal: true,
43         x: 0,
44         y: 0
45     })
46 }
47
48 // This method will be called when Electron has finished
49 // initialization and is ready to create browser windows.
```