Electron Security

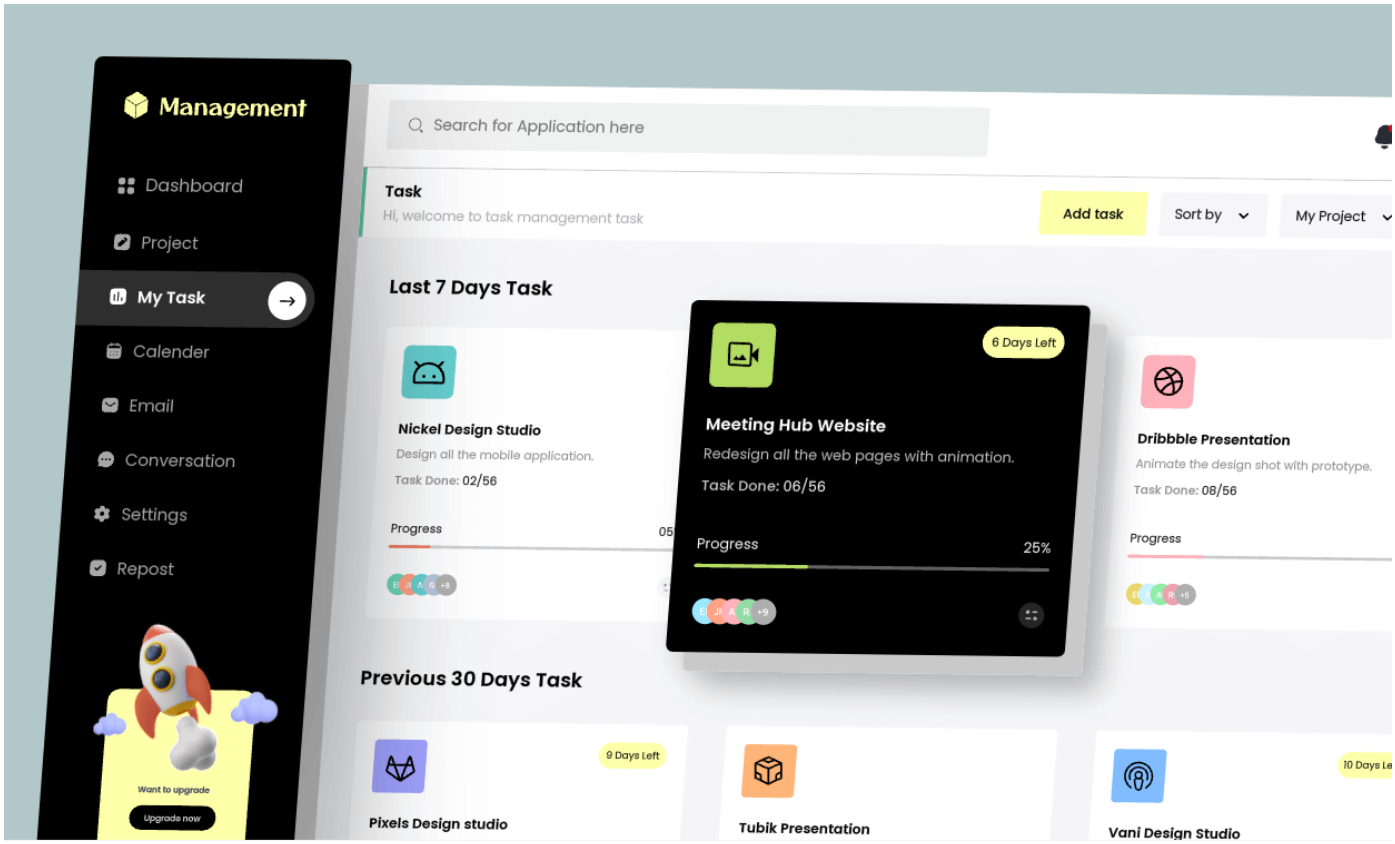
跳转与导航

NOP Team

跳转与导航 | Electron 安全

0x01 简介

导航我们都知道，高德地图对吧，我们搜索一个地点，它告诉我们如何到达。对于网站来说，导航是帮助用户到达用户想去的地方(网址)

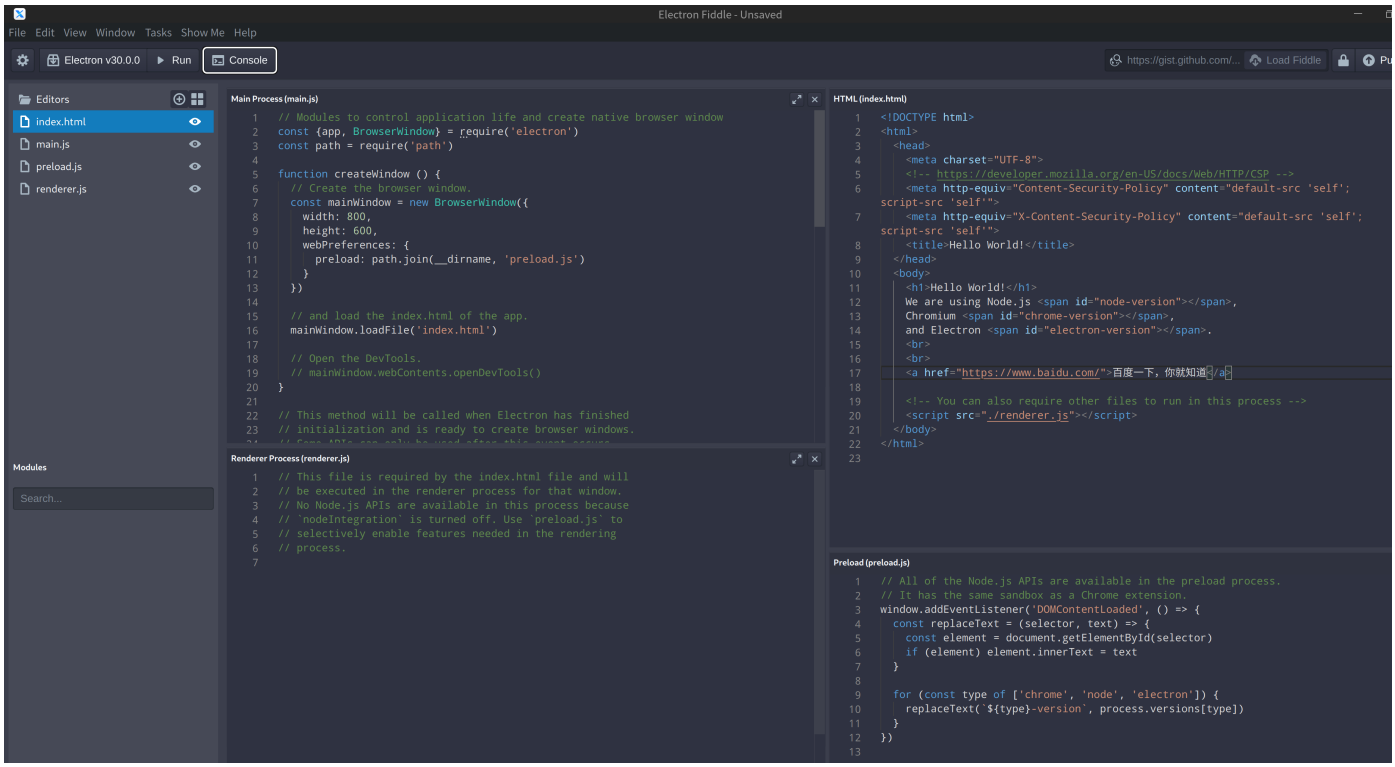


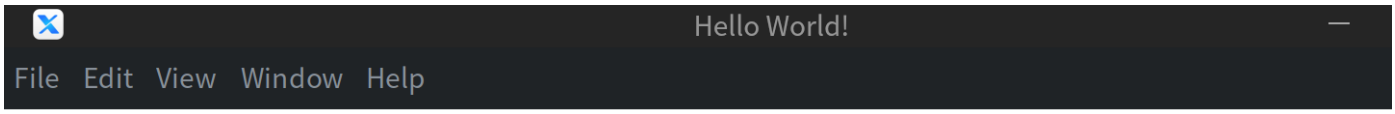
在 `Electron` 中也是一样，凡是离开当前地址的操作都可以算作是跳转和导航，最常见的是点击了某个链接，之后我们进入到链接中，点击了某个功能，进入到该功能模块中

对于应用程序来说，通常不需要在页面中渲染第三方的网页，尤其是在 `Electron` 中，加载第三方页面可能会导致用户被远程命令执行，因此官方推荐**禁用或限制网页跳转**

> 参考文章
>
> https://www.electronjs.org/zh/docs/latest/tutorial/security#13-%E7%A6%81%E7%94%A8%E6%88%96%E9%99%90%E5%88%B6%E7%BD%91%E9%A1%B5%E8%B7%B3%E8%BD%AC

0x02 效果展示





Hello World!

We are using Node.js 20.11.1, Chromium 124.0.6367.49, and Electron 30.0.0.

百度一下，你就知道

点击链接后

0x03 官方安全建议

官方建议是禁用或限制网页跳转，所谓的限制也就是说选择性地网页跳转，例如允许跳转到自己以及子域名等可控范围内的内容，官方给出代码如下

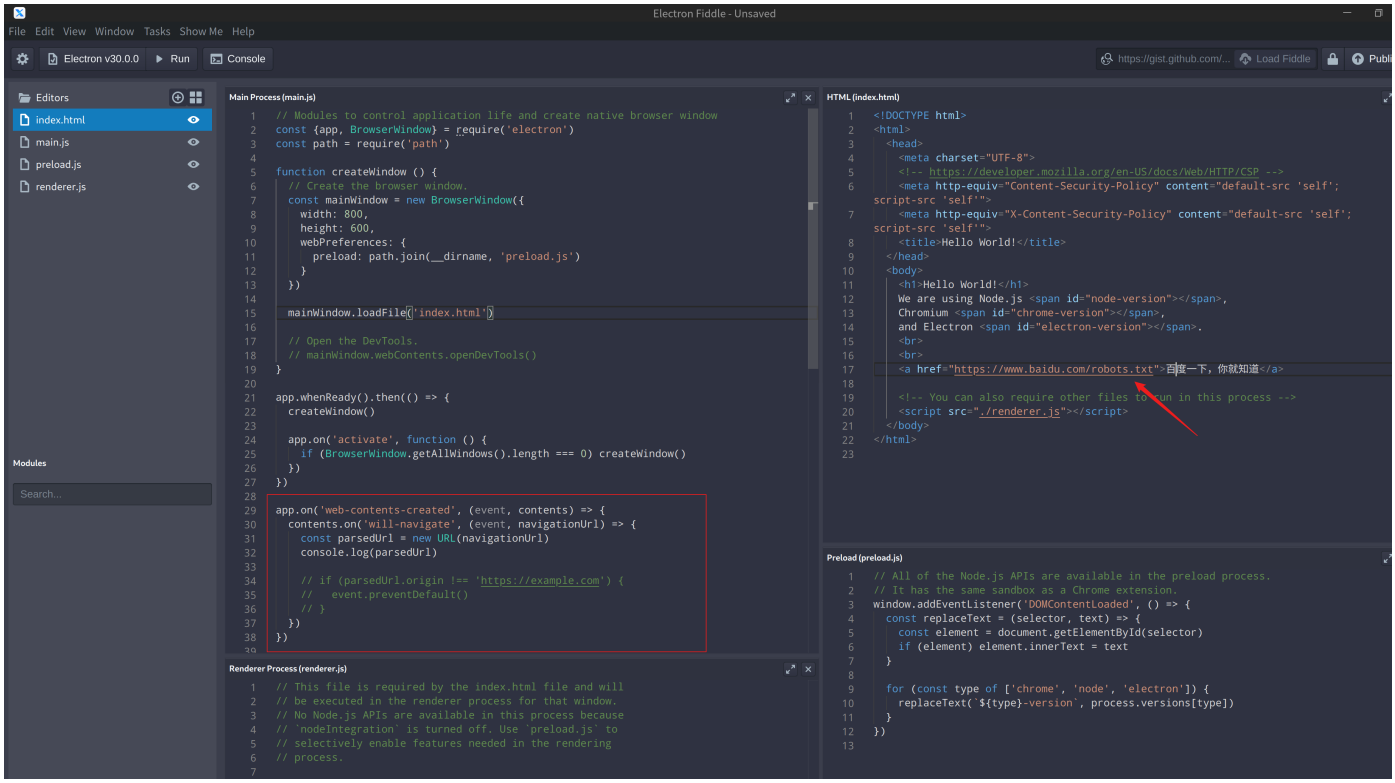
```
const { URL } = require('url')
const { app } = require('electron')

app.on('web-contents-created', (event, contents) => {
  contents.on('will-navigate', (event, navigationUrl) => {
    const parsedUrl = new URL(navigationUrl)

    if (parsedUrl.origin !== 'https://example.com') {
      event.preventDefault()
    }
  })
})
```

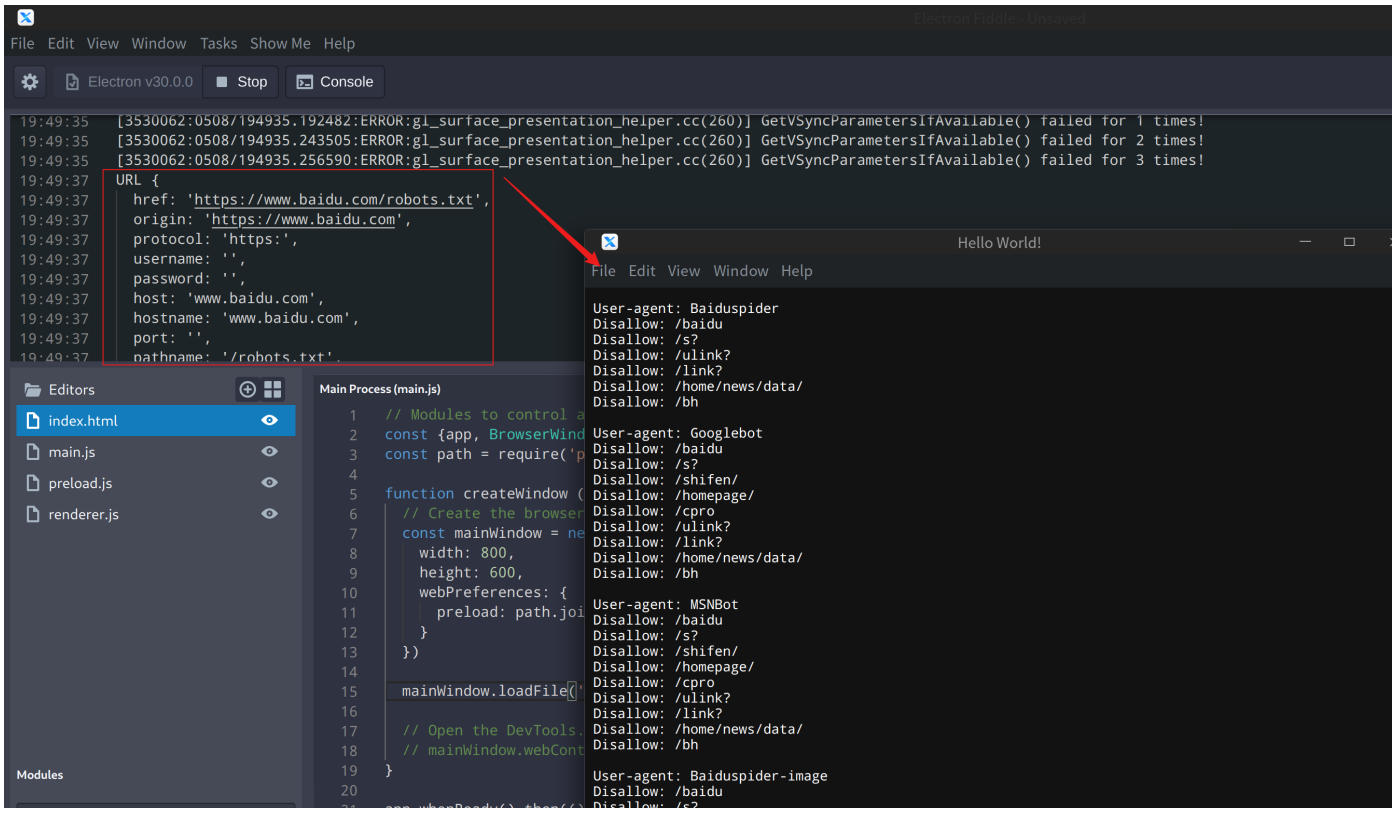
官方还专门强调，建议使用Node的解析器来处理URL, 简单的字符串比较有时会出错

0x04 哪些行为会导致网页跳转

既然有了防御代码，我们便可以测试一下，到底哪些行为会进行网页跳转

1. a 标签





点击 `a` 标签后，成功输出 `URL` 对象，其中完整内容为

```
URL {
  href: 'https://www.baidu.com/robots.txt',
  origin: 'https://www.baidu.com',
  protocol: 'https:',
  username: '',
  password: '',
  host: 'www.baidu.com',
  hostname: 'www.baidu.com',
  port: '',
  pathname: '/robots.txt',
  search: '',
  searchParams: URLSearchParams {},
  hash: ''
}
```

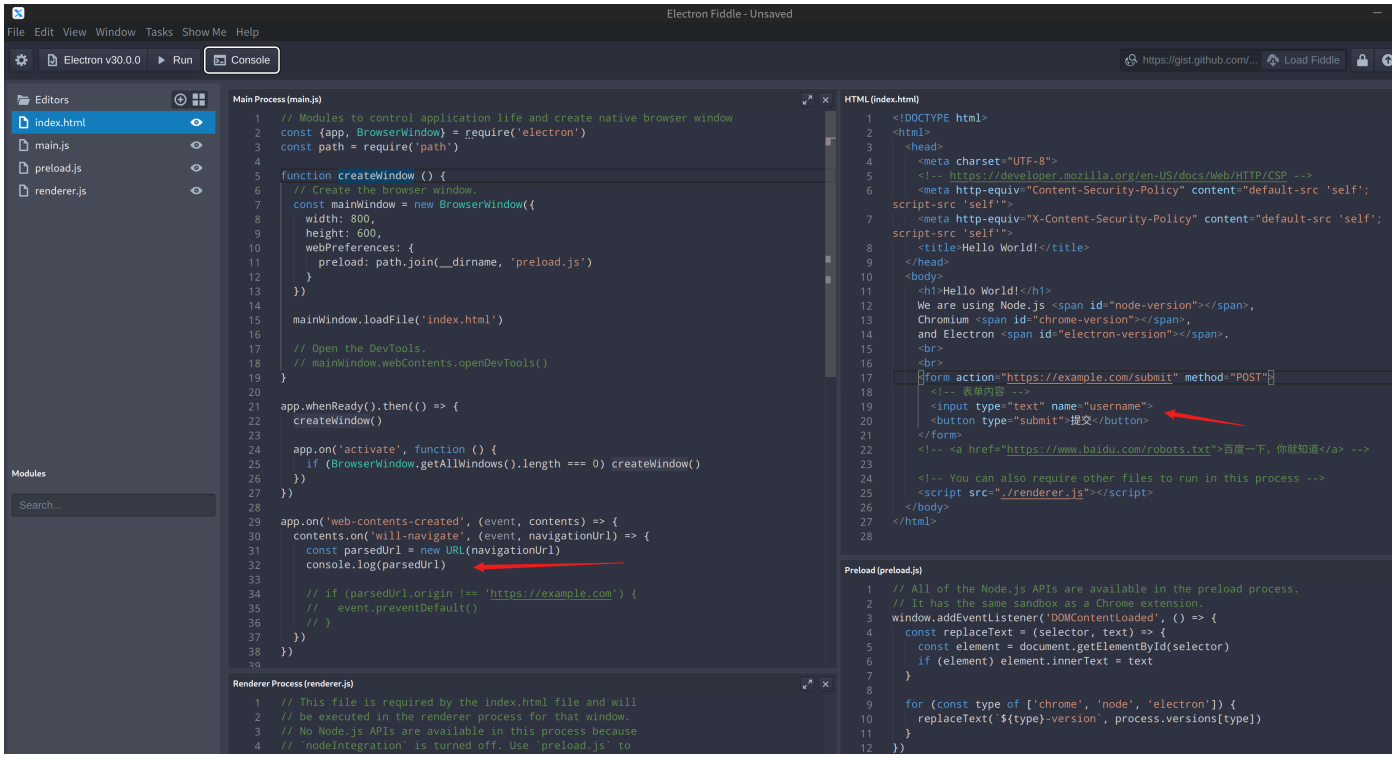
其中 `origin` 就是我们所谓的同源策略里的源，它包含协议、主机名、端口号

所以官方的防御代码就是验证是不是与 `https://example.com` 同源的，非同源则直接组织

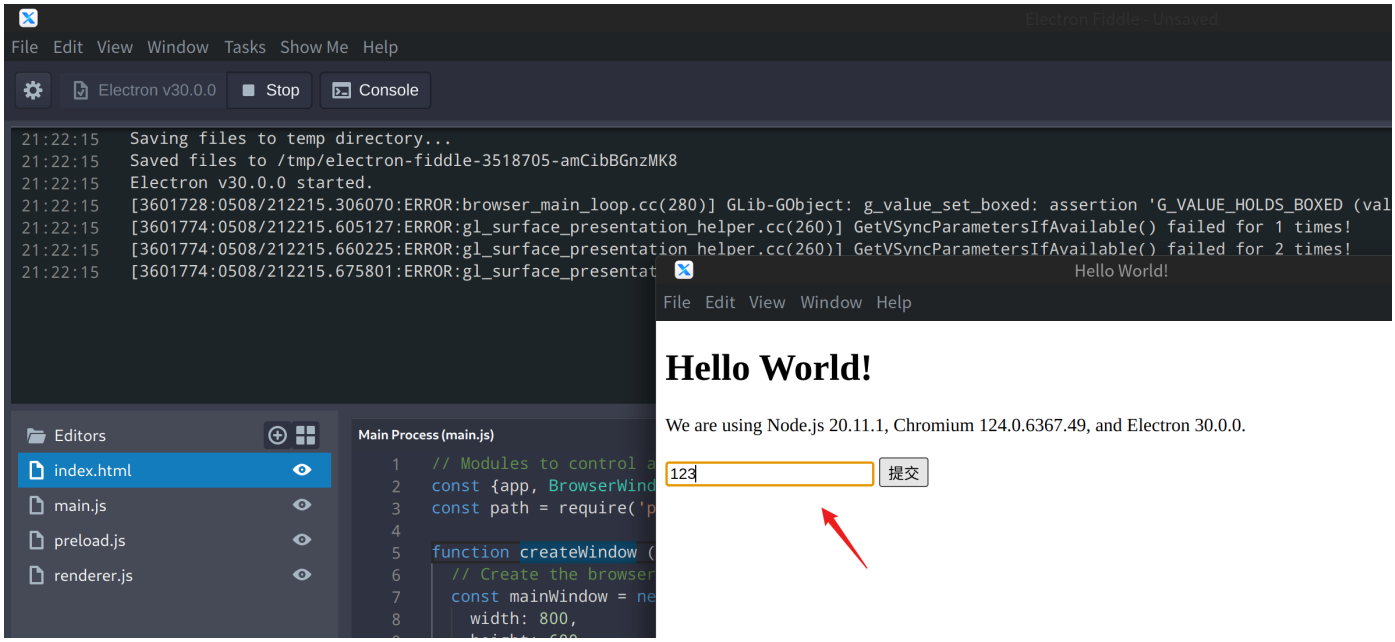
2. 表单提交

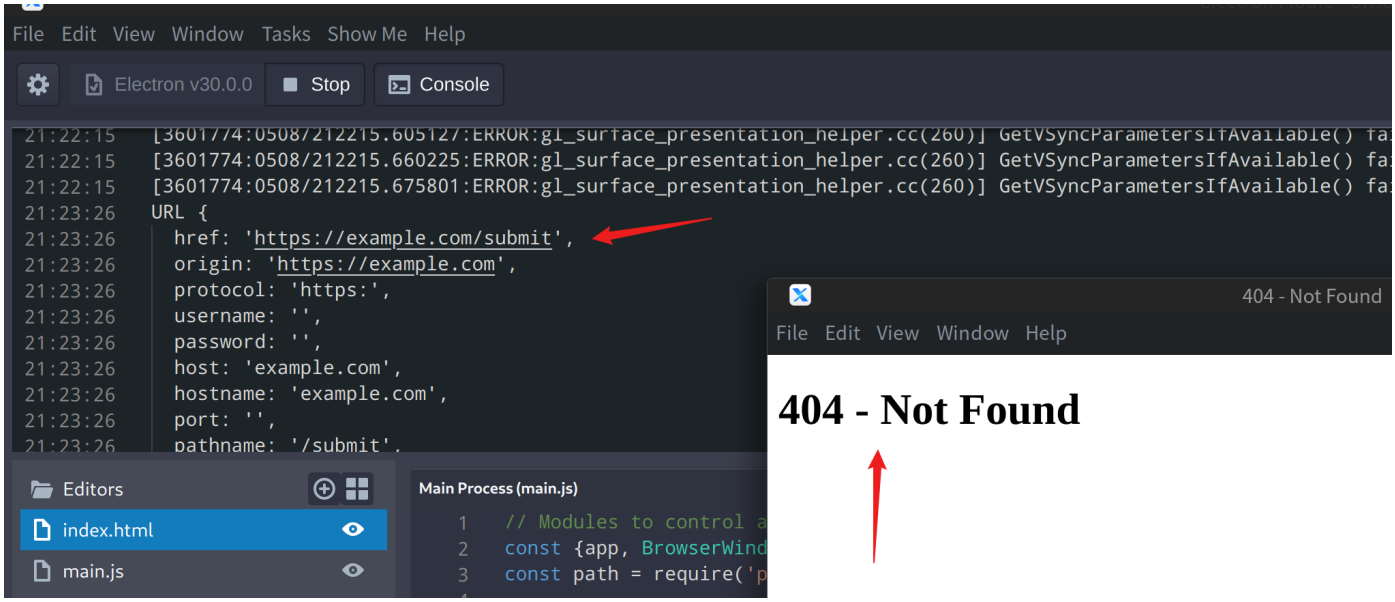
```
<form action="https://example.com/submit" method="POST">
  <!-- 表单内容 -->
  <input type="text" name="username">
  <button type="submit">提交</button>
</form>
```

上一节新窗口创建的案例，当然这里 `target` 设置什么无所谓，我们直接去掉了，关键是 `action` 属性，这个属性的值造成跳转


Electron Fiddle - Unsaved
File Edit View Window Tasks Show Me Help
Electron v30.0.0
Run
Console
Load Fiddle
Editors
index.html
main.js
preload.js
renderer.js
Modules
Search...
Main Process (main.js)
// Modules to control application life and create native browser window
const {app, BrowserWindow} = require('electron')
const path = require('path')
function createWindow () {
// Create the browser window.
const mainWindow = new BrowserWindow({
width: 800,
height: 600,
webPreferences: {
preload: path.join(__dirname, 'preload.js')
}
})
mainWindow.loadFile('index.html')
// Open the DevTools.
// mainWindow.webContents.openDevTools()
}
app.whenReady().then(() => {
createWindow()
app.on('activate', function () {
if (BrowserWindow.getAllWindows().length === 0) createWindow()
})
})
app.on('web-contents-created', (event, contents) => {
contents.on('will-navigate', (event, navigationUrl) => {
const parsedUrl = new URL(navigationUrl)
console.log(parsedUrl)
// if (parsedUrl.origin !== 'https://example.com') {
// event.preventDefault()
// }
})
})
Renderer Process (renderer.js)
// This file is required by the index.html file and will
// be executed in the renderer process for that window.
// No Node.js APIs are available in this process because
// `nodeIntegration` is turned off. Use `preload.js` to
HTML (index.html)
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<!-- https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP -->
<meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src 'self'">
<meta http-equiv="X-Content-Security-Policy" content="default-src 'self'; script-src 'self'">
<title>Hello World!</title>
</head>
<body>
<h1>Hello World!</h1>
We are using Node.js ,
Chromium ,
and Electron .

<form action="https://example.com/submit" method="POST">
<!-- 表单内容 -->
<input type="text" name="username">
<button type="submit">提交</button>
</form>
<!-- 百度一下，你就知道 -->
<!-- You can also require other files to run in this process -->
<script src="./renderer.js"></script>
</body>
</html>
Preload (preload.js)
// All of the Node.js APIs are available in the preload process.
// It has the same sandbox as a Chrome extension.
window.addEventListener('DOMContentLoaded', () => {
const replaceText = (selector, text) => {
const element = document.getElementById(selector)
if (element) element.innerText = text
}
for (const type of ['chrome', 'node', 'electron']) {
replaceText(`${type}-version`, process.versions[type])
}
})

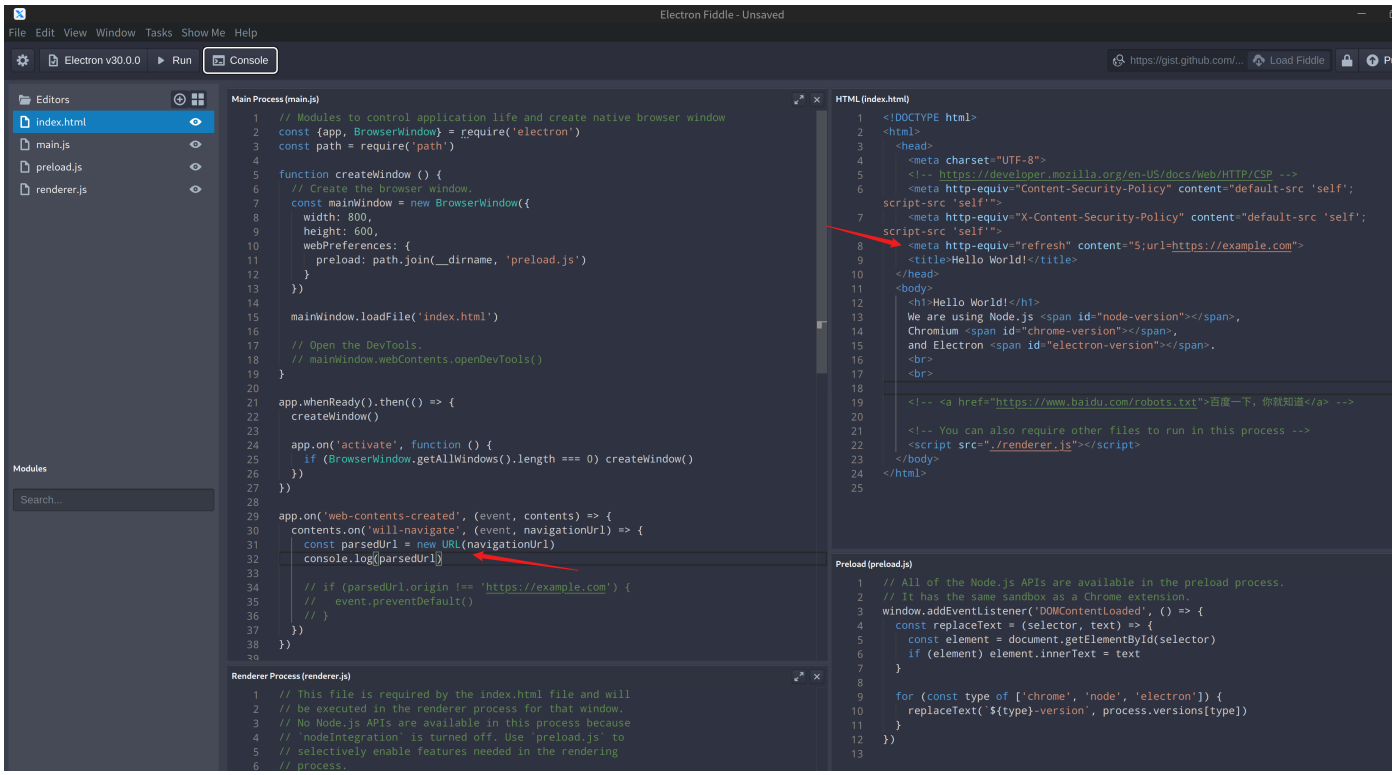

File Edit View Window Tasks Show Me Help
Electron v30.0.0
Stop
Console
21:22:15 Saving files to temp directory...
21:22:15 Saved files to /tmp/electron-fiddle-3518705-amCibBGnzMK8
21:22:15 Electron v30.0.0 started.
21:22:15 [3601728:0508/212215.306070:ERROR:browser_main_loop.cc(280)] GLib-GObject: g_value_set_boxed: assertion 'G_VALUE_HOLDS_BOXED (val
21:22:15 [3601774:0508/212215.605127:ERROR:gl_surface_presentation_helper.cc(260)] GetVSyncParametersIfAvailable() failed for 1 times!
21:22:15 [3601774:0508/212215.660225:ERROR:gl_surface_presentation_helper.cc(260)] GetVSyncParametersIfAvailable() failed for 2 times!
21:22:15 [3601774:0508/212215.675801:ERROR:gl_surface_presentat
Editors
index.html
main.js
preload.js
renderer.js
Main Process (main.js)
// Modules to control a
const {app, BrowserWind
const path = require('p
function createWindow (
// Create the browser
const mainWindow = ne
width: 800,
Hello World!
File Edit View Window Help
Hello World!
We are using Node.js 20.11.1, Chromium 124.0.6367.49, and Electron 30.0.0.
123
提交

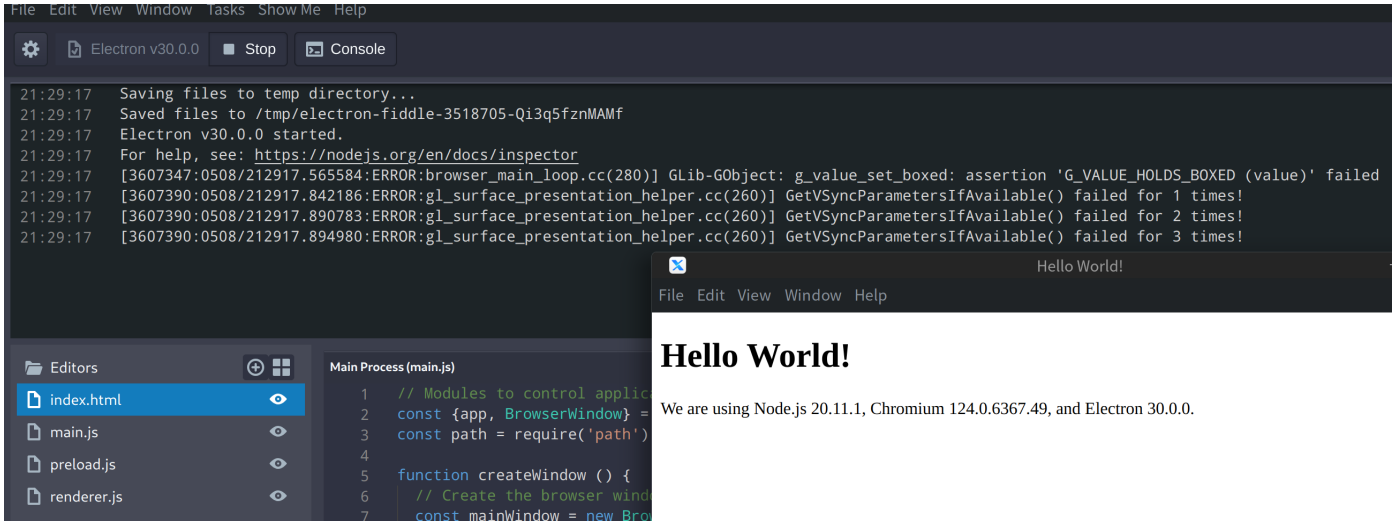


出发了跳转和导航事件

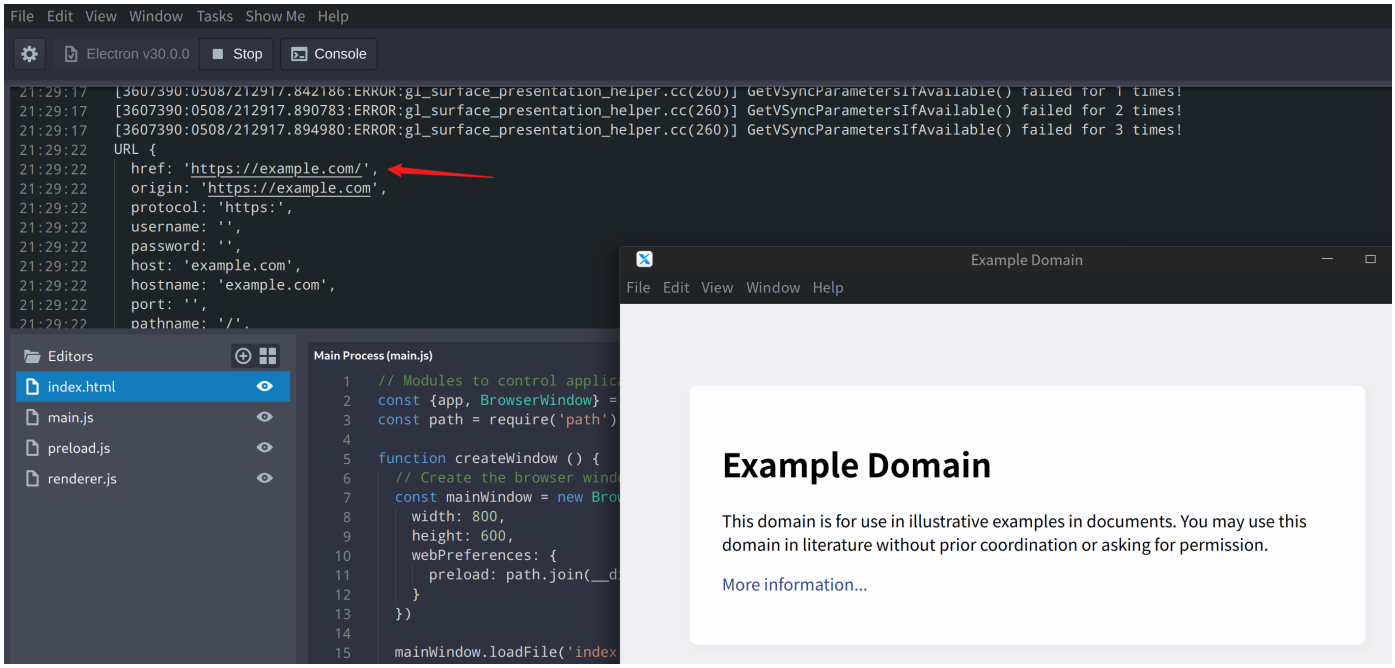
3. meta 标签自动刷新

```
<meta http-equiv="refresh" content="5;url=https://example.com">
```



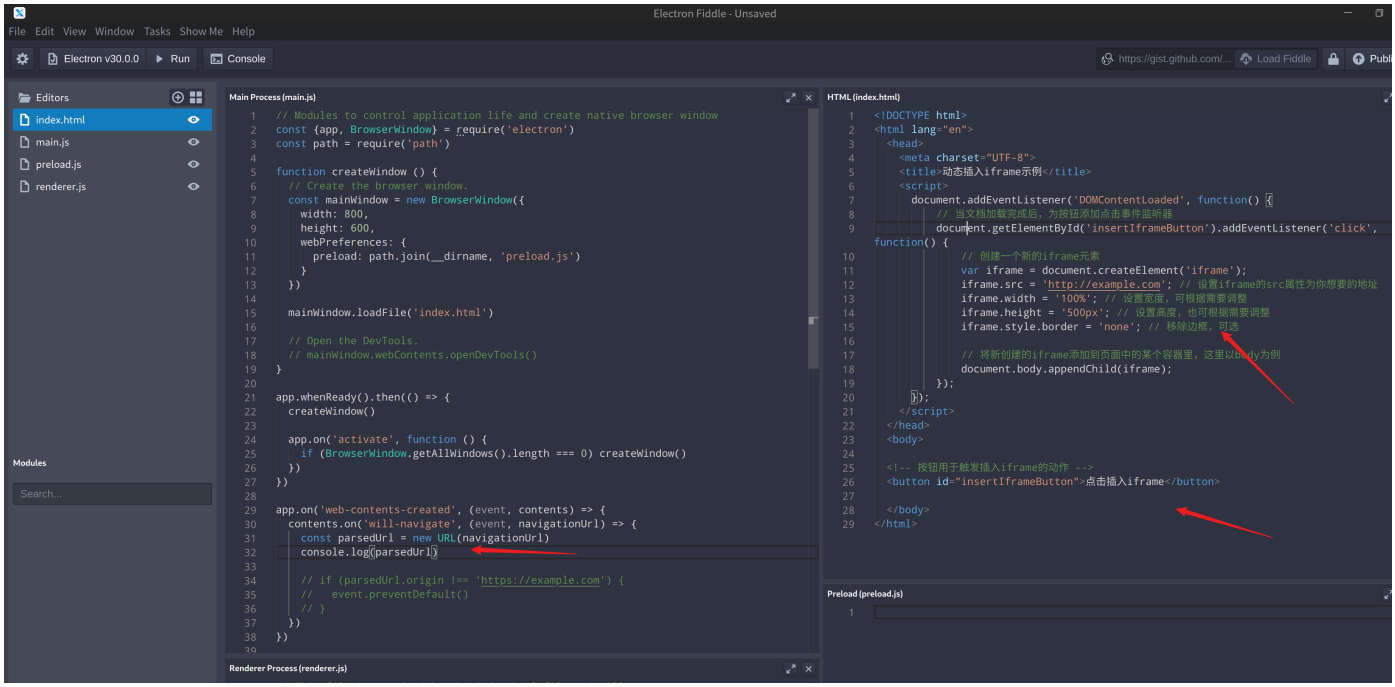


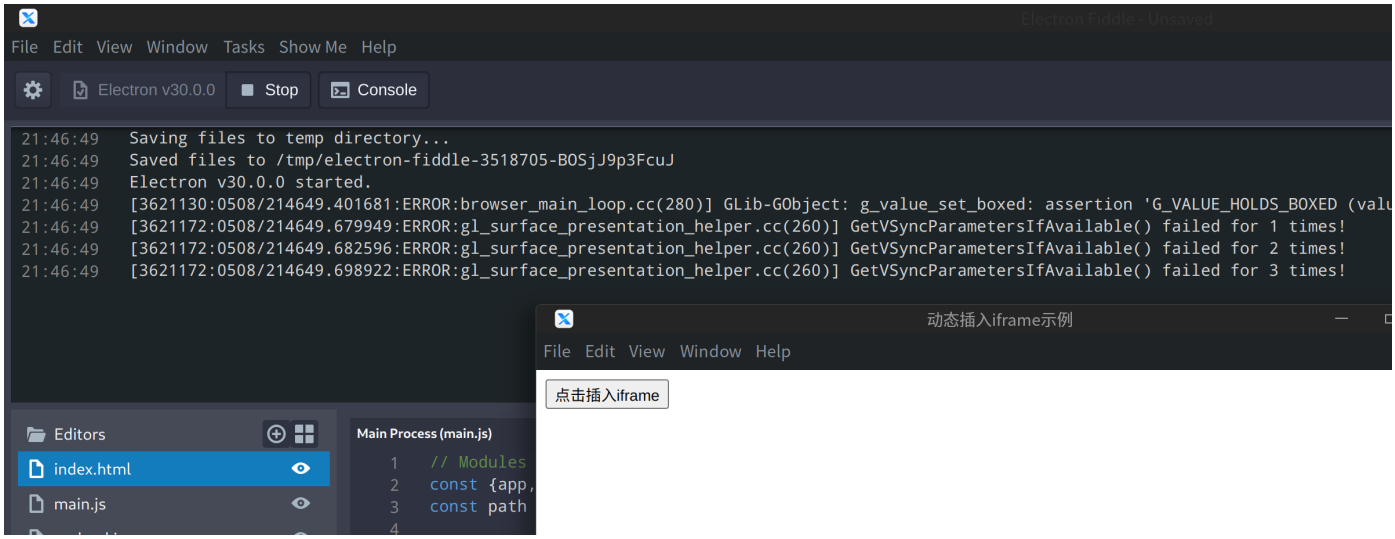
5 秒后



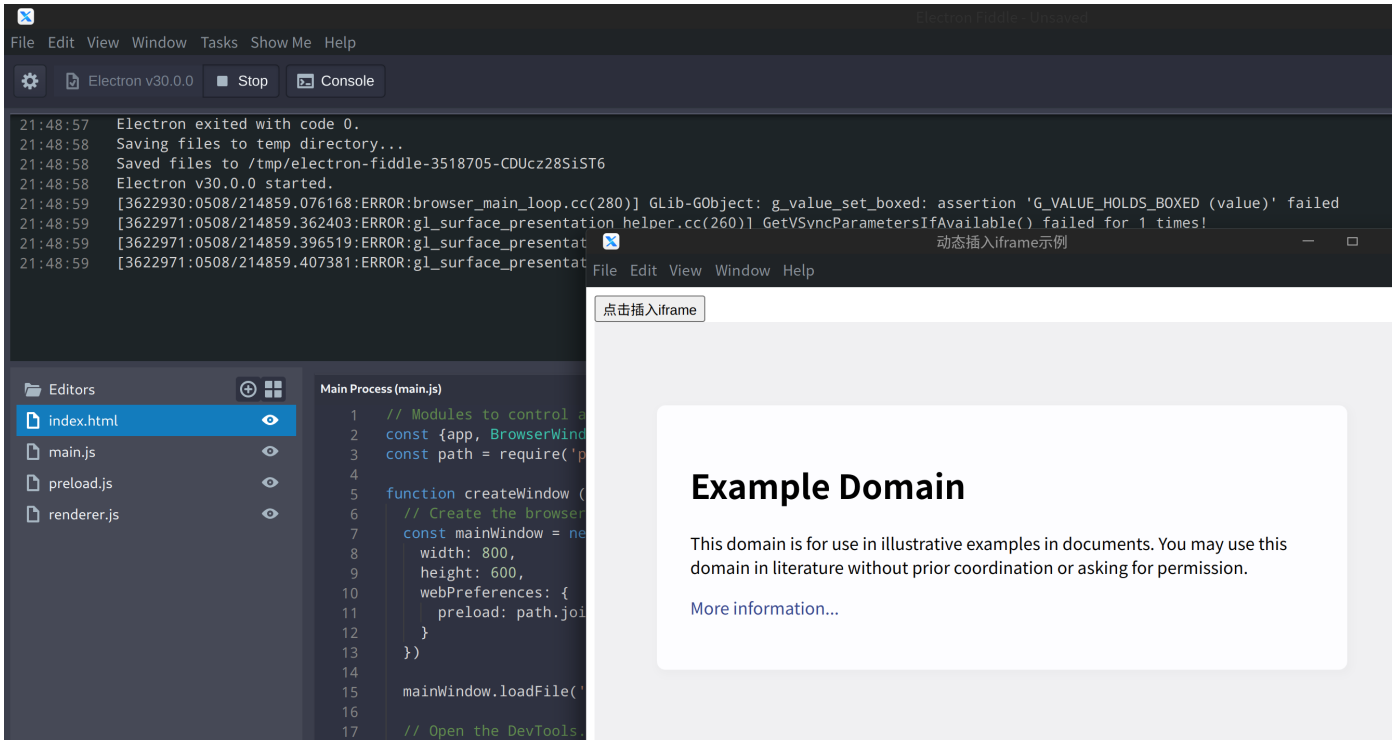
成功触发监听

4. iframe 加载

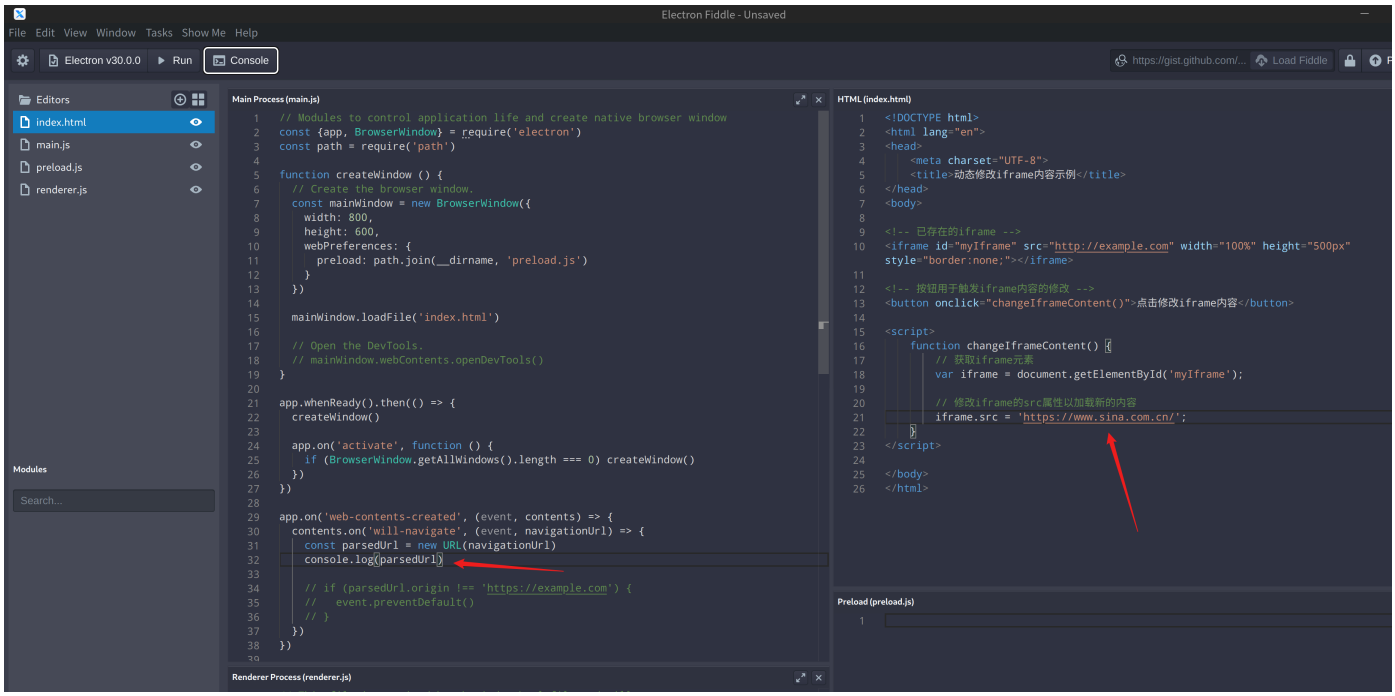

Electron Fiddle - Unsaved
File Edit View Window Tasks Show Me Help
Electron v30.0.0
Run
Console
Editors
index.html
main.js
preload.js
renderer.js
Modules
Search...
Main Process (main.js)
// Modules to control application life and create native browser window
const {app, BrowserWindow} = require('electron')
const path = require('path')
function createWindow () {
// Create the browser window.
const mainWindow = new BrowserWindow({
width: 800,
height: 600,
webPreferences: {
preload: path.join(__dirname, 'preload.js')
}
})
mainWindow.loadFile('index.html')
// Open the DevTools.
// mainWindow.webContents.openDevTools()
}
app.whenReady().then(() => {
createWindow()
app.on('activate', function () {
if (BrowserWindow.getAllWindows().length === 0) createWindow()
})
})
app.on('web-contents-created', (event, contents) => {
contents.on('will-navigate', (event, navigationUrl) => {
const parsedUrl = new URL(navigationUrl)
console.log(parsedUrl)
// if (parsedUrl.origin !== 'https://example.com') {
// event.preventDefault()
// }
})
})
Renderer Process (renderer.js)
HTML (index.html)
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>动态插入iframe示例</title>
<script>
document.addEventListener('DOMContentLoaded', function() {
// 当文档加载完成后，为按钮添加点击事件监听器
document.getElementById('insertIframeButton').addEventListener('click',
function() {
// 创建一个新的iframe元素
var iframe = document.createElement('iframe');
iframe.src = 'http://example.com'; // 设置iframe的src属性为你想要的地址
iframe.width = '100%'; // 设置宽度，可根据需要调整
iframe.height = '500px'; // 设置高度，也可根据需要调整
iframe.style.border = 'none'; // 移除边框，可选
// 将新创建的iframe添加到页面中的某个容器里，这里以body为例
document.body.appendChild(iframe);
});
});
</script>
</head>
<body>
<!-- 按钮用于触发插入iframe的动作 -->
<button id="insertIframeButton">点击插入iframe</button>
</body>
</html>
Preload (preload.js)

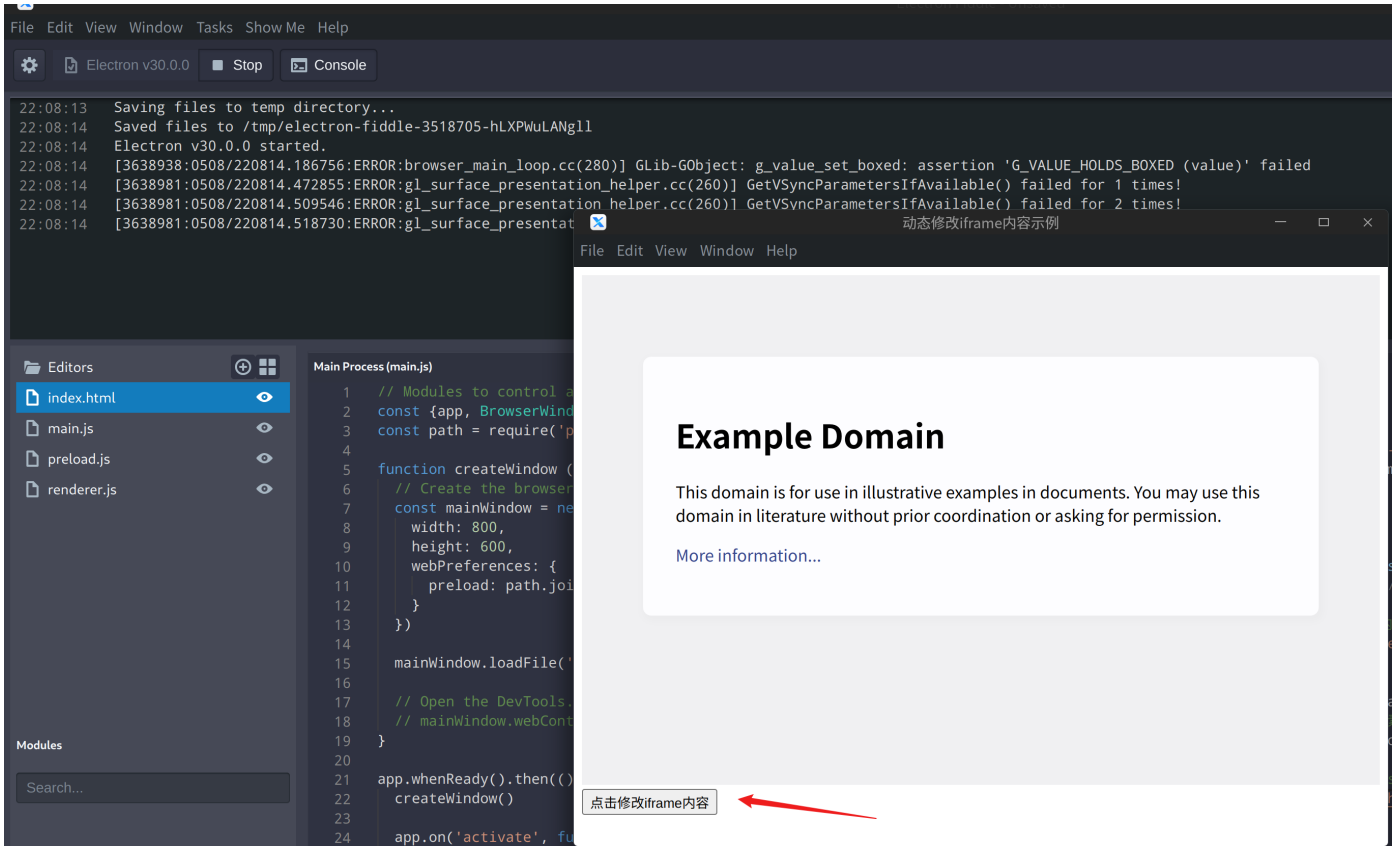

File Edit View Window Tasks Show Me Help
Electron v30.0.0
Stop
Console
21:46:49 Saving files to temp directory...
21:46:49 Saved files to /tmp/electron-fiddle-3518705-BOSjJ9p3FcuJ
21:46:49 Electron v30.0.0 started.
21:46:49 [3621130:0508/214649.401681:ERROR:browser_main_loop.cc(280)] GLib-GObject: g_value_set_boxed: assertion 'G_VALUE_HOLDS_BOXED (valu
21:46:49 [3621172:0508/214649.679949:ERROR:gl_surface_presentation_helper.cc(260)] GetVSyncParametersIfAvailable() failed for 1 times!
21:46:49 [3621172:0508/214649.682596:ERROR:gl_surface_presentation_helper.cc(260)] GetVSyncParametersIfAvailable() failed for 2 times!
21:46:49 [3621172:0508/214649.698922:ERROR:gl_surface_presentation_helper.cc(260)] GetVSyncParametersIfAvailable() failed for 3 times!
动态插入iframe示例
File Edit View Window Help
点击插入iframe
Editors
index.html
main.js
Main Process (main.js)
// Modules
const {app,
const path

点击按钮

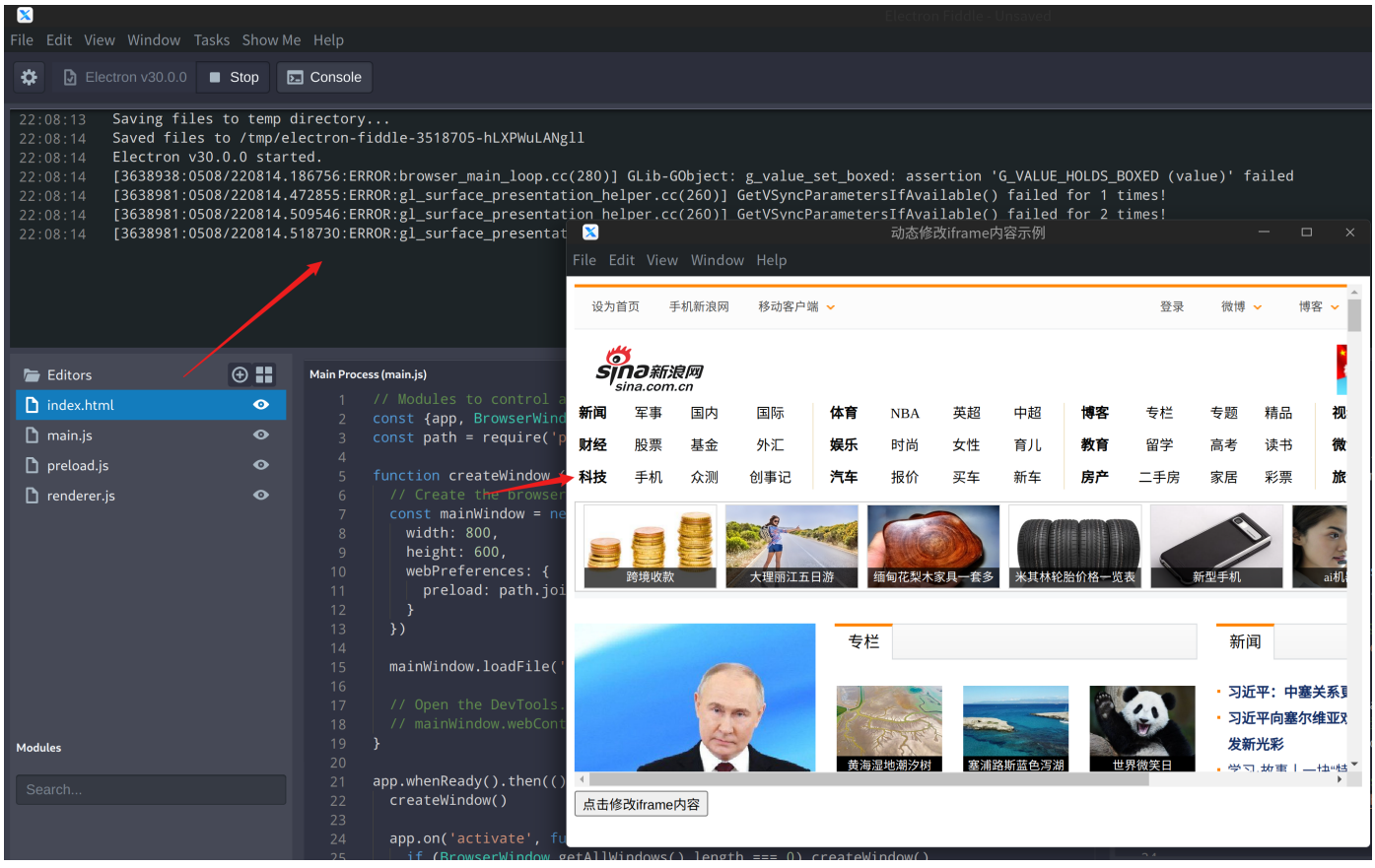


创建一个 `iframe` 并没有引起主进程的跳转和导航事件，我们修改代码，测试一下按按钮修改 `iframe` 的 `src` 属性



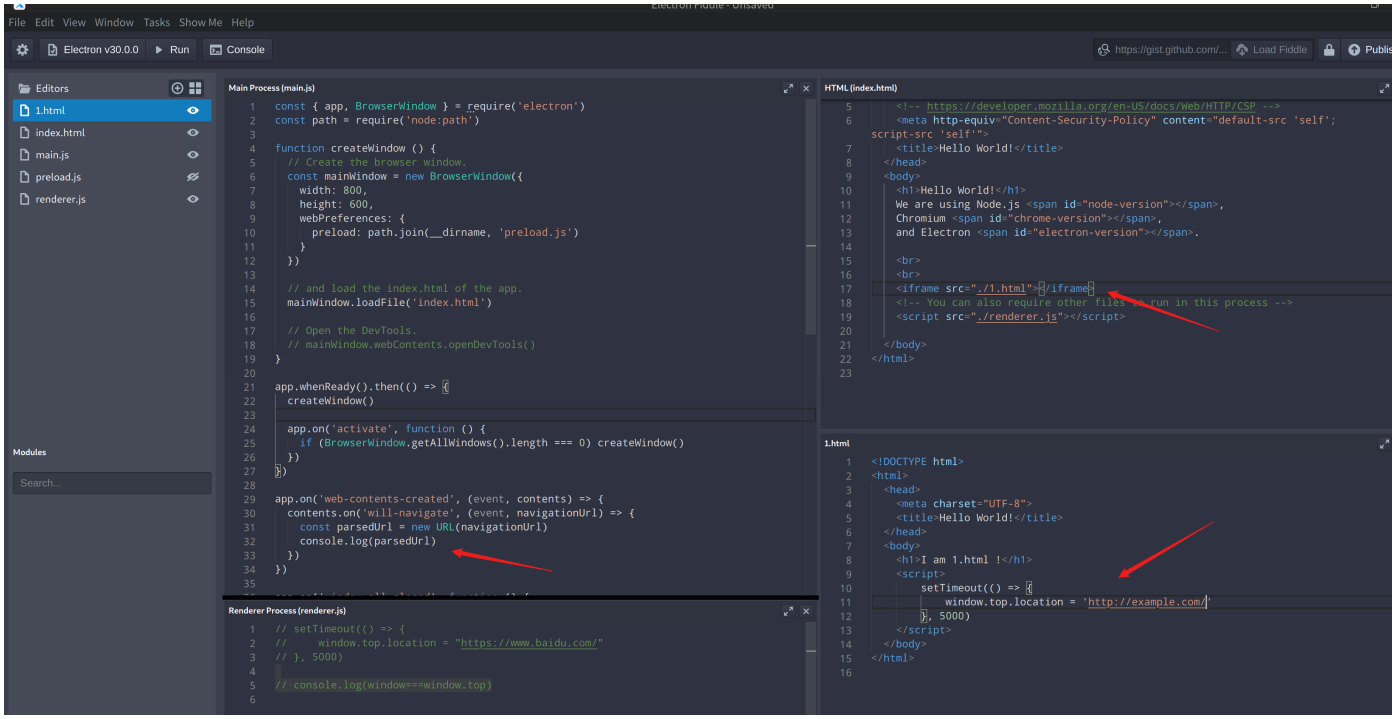


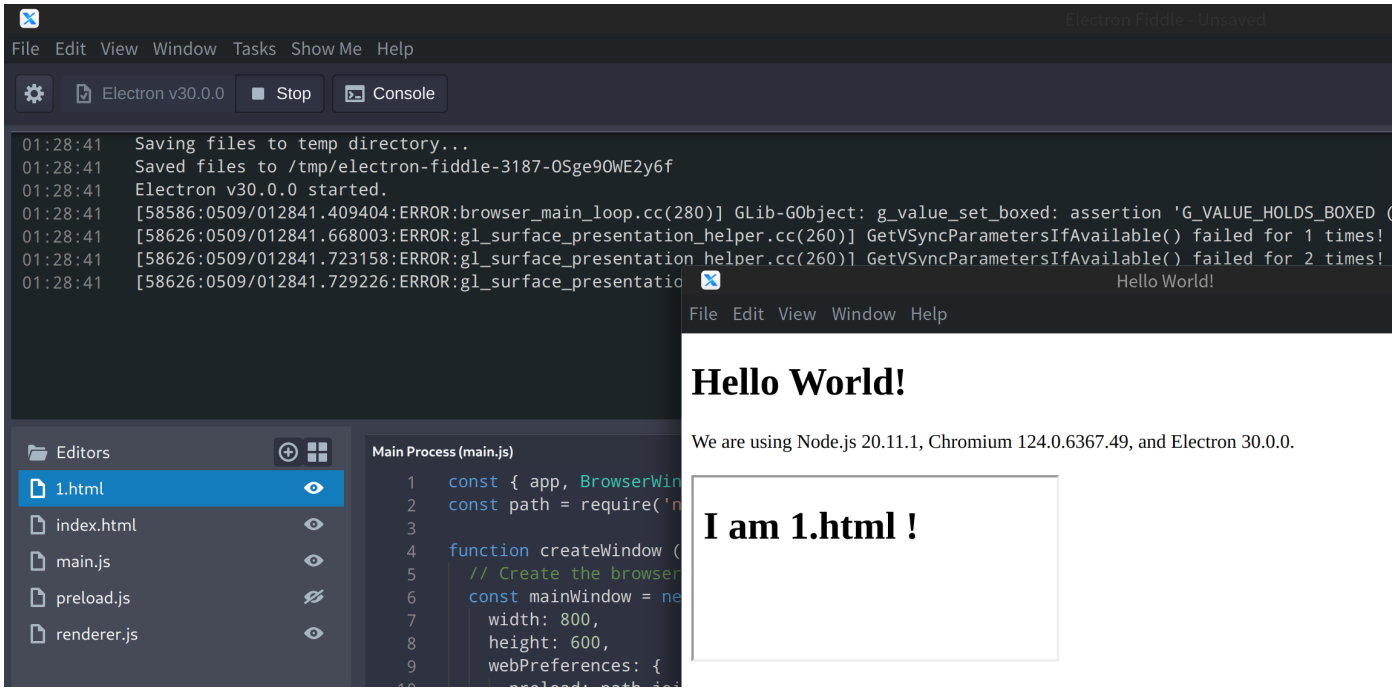
点击按钮



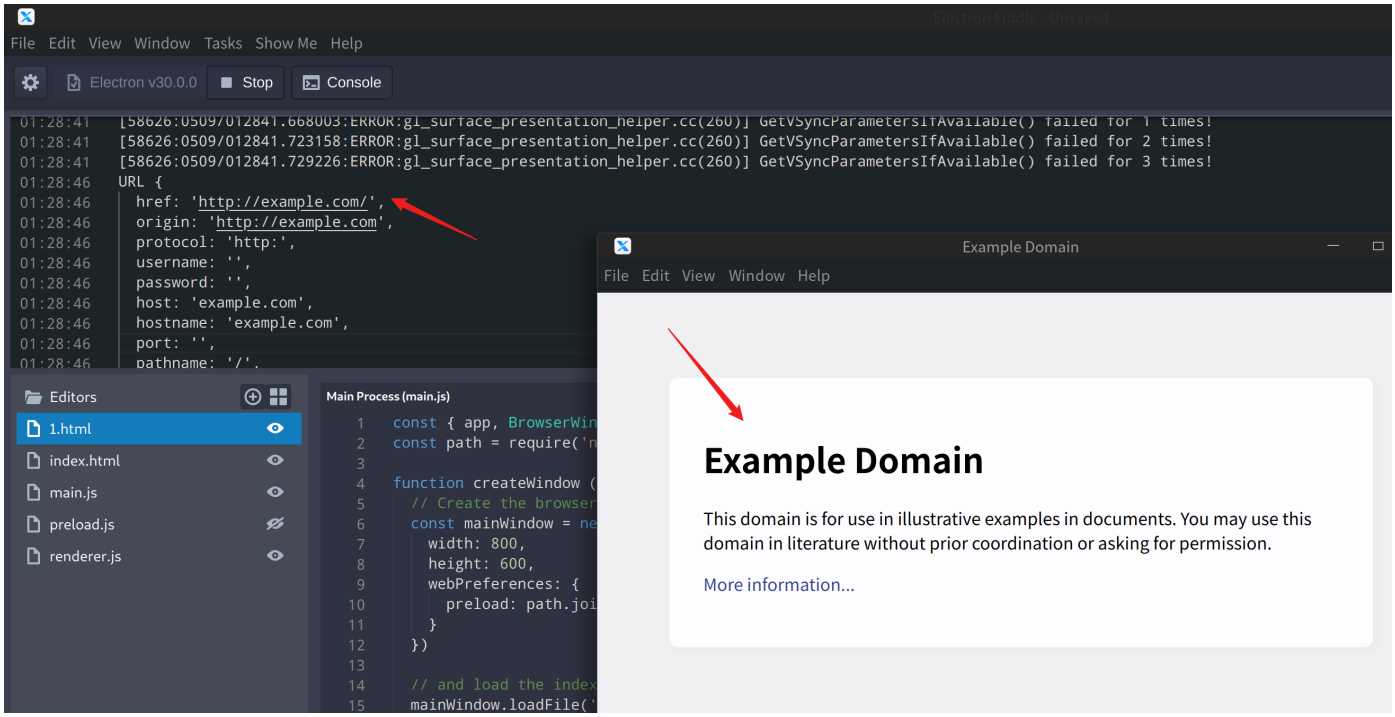
看来 `iframe` 的 `src` 修改不会触发主进程的跳转与导航事件

`iframe` 加载的内容中通过 `window.top.location` 修改顶层窗口的 URL





5 秒后



触发导航事件

5. window.location

`Window.location` 只读属性返回一个 `Location` 对象，其中包含有关文档当前位置的信息

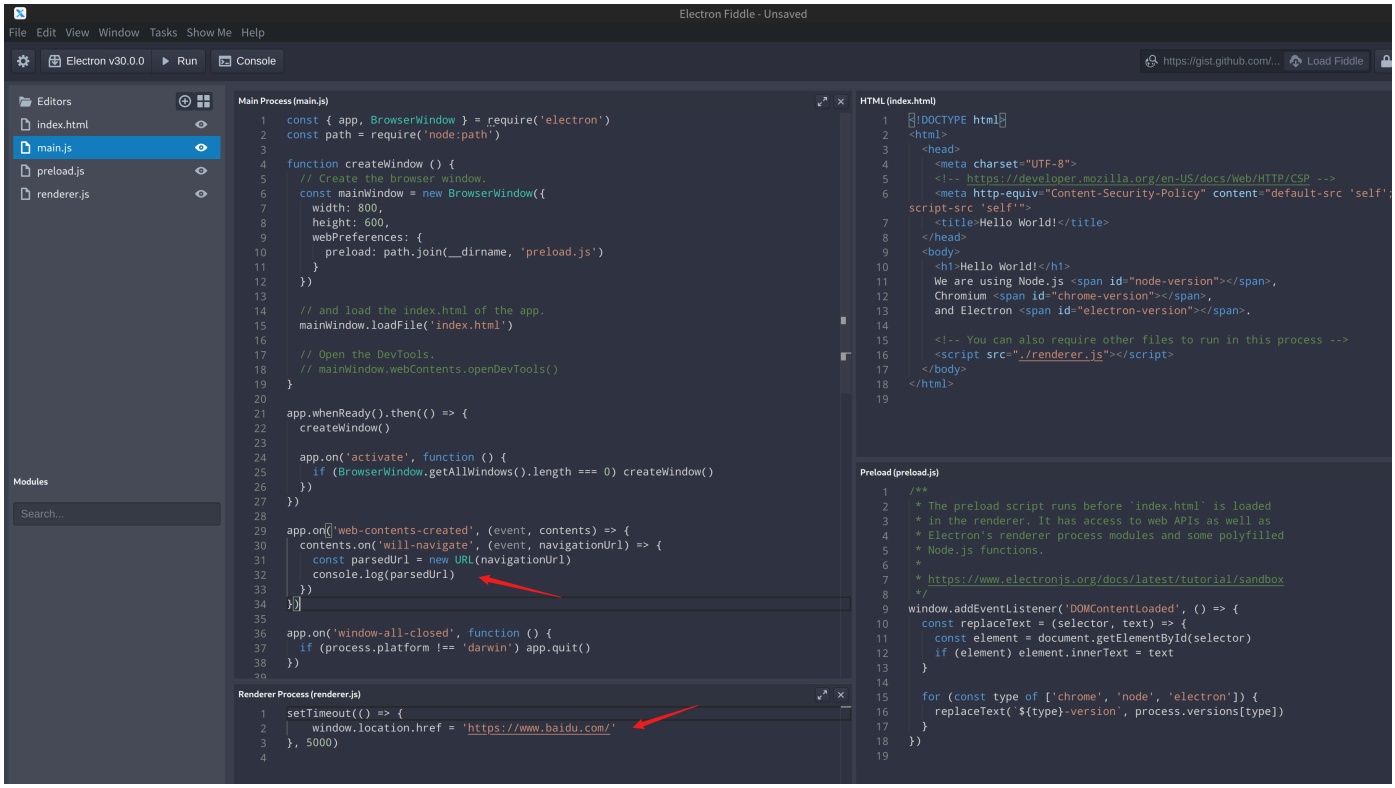
尽管 `Window.location` 是一个只读 `Location` 对象，你仍然可以将字符串赋值给它。这意味着可以在大多数情况下像字符串一样处理 `location` —— `location = 'http://www.example.com'` ——与 `location.href = 'http://www.example.com'` 等价。

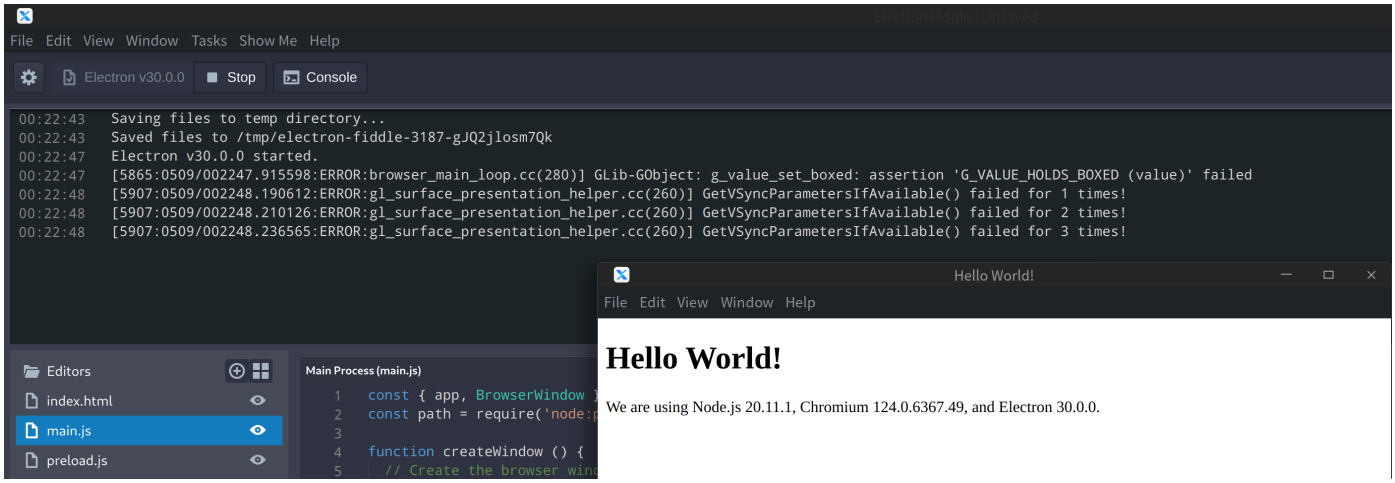
在上一篇文章中，我们介绍了通过 `window.open().location` 绕过安全限制的手法，其中 `location` 或者说 `location.href` 的值就是要导航去的位置

1) location.href

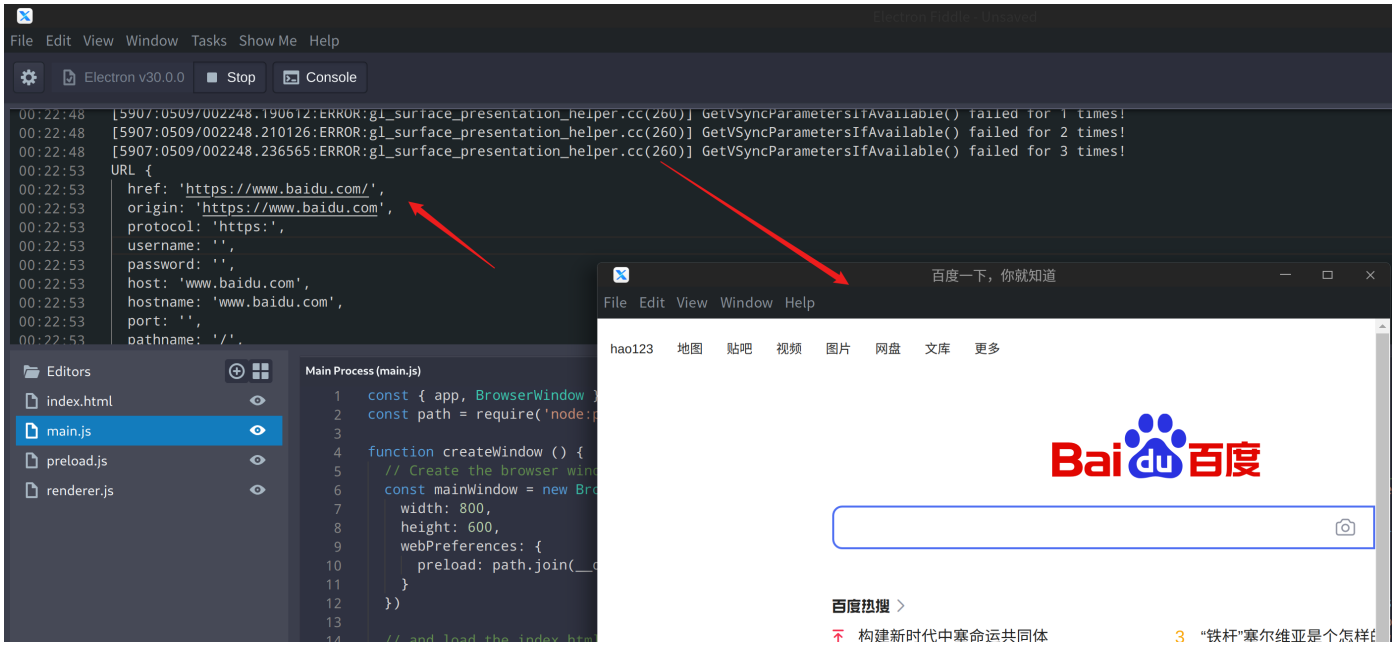
返回当前页面的完整URL字符串，也可以用来设置新的URL以导航到其他页面

```
window.location.href = "https://www.baidu.com/"
```


Electron Fiddle - Unsaved
File Edit View Window Tasks Show Me Help
Electron v30.0.0
Run
Console
Load Fiddle
Editors
index.html
main.js
preload.js
renderer.js
Modules
Search...
Main Process (main.js)
const { app, BrowserWindow } = require('electron')
const path = require('node:path')
function createWindow () {
// Create the browser window.
const mainWindow = new BrowserWindow({
width: 800,
height: 600,
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()
}
app.whenReady().then(() => {
createWindow()
app.on('activate', function () {
if (BrowserWindow.getAllWindows().length === 0) createWindow()
})
})
app.on('web-contents-created', (event, contents) => {
contents.on('will-navigate', (event, navigationUrl) => {
const parsedUrl = new URL(navigationUrl)
console.log(parsedUrl)
})
})
app.on('window-all-closed', function () {
if (process.platform !== 'darwin') app.quit()
})
Renderer Process (renderer.js)
setTimeout(() => {
window.location.href = 'https://www.baidu.com/'
}, 5000)
HTML (index.html)
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<!-- https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP -->
<meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src 'self'">
<title>Hello World!</title>
</head>
<body>
<h1>Hello World!</h1>
We are using Node.js ,
Chromium ,
and Electron .
<!-- You can also require other files to run in this process -->
<script src="./renderer.js"></script>
</body>
</html>
Preload (preload.js)
/**
* The preload script runs before `index.html` is loaded
* in the renderer. It has access to web APIs as well as
* Electron's renderer process modules and some polyfilled
* Node.js functions.
*
* https://www.electronjs.org/docs/latest/tutorial/sandbox
*/
window.addEventListener('DOMContentLoaded', () => {
const replaceText = (selector, text) => {
const element = document.getElementById(selector)
if (element) element.innerText = text
}
for (const type of ['chrome', 'node', 'electron']) {
replaceText(`${type}-version`, process.versions[type])
}
})


File Edit View Window Tasks Show Me Help
Electron v30.0.0
Stop
Console
00:22:43 Saving files to temp directory...
00:22:43 Saved files to /tmp/electron-fiddle-3187-gJQ2jlosm7Qk
00:22:47 Electron v30.0.0 started.
00:22:47 [5865:0509/002247.915598:ERROR:browser_main_loop.cc(280)] GLib-GObject: g_value_set_boxed: assertion 'G_VALUE_HOLDS_BOXED (value)' failed
00:22:48 [5907:0509/002248.190612:ERROR:gl_surface_presentation_helper.cc(260)] GetVSyncParametersIfAvailable() failed for 1 times!
00:22:48 [5907:0509/002248.210126:ERROR:gl_surface_presentation_helper.cc(260)] GetVSyncParametersIfAvailable() failed for 2 times!
00:22:48 [5907:0509/002248.236565:ERROR:gl_surface_presentation_helper.cc(260)] GetVSyncParametersIfAvailable() failed for 3 times!
Editors
index.html
main.js
preload.js
Main Process (main.js)
Hello World!
File Edit View Window Help
Hello World!
We are using Node.js 20.11.1, Chromium 124.0.6367.49, and Electron 30.0.0.

5秒后

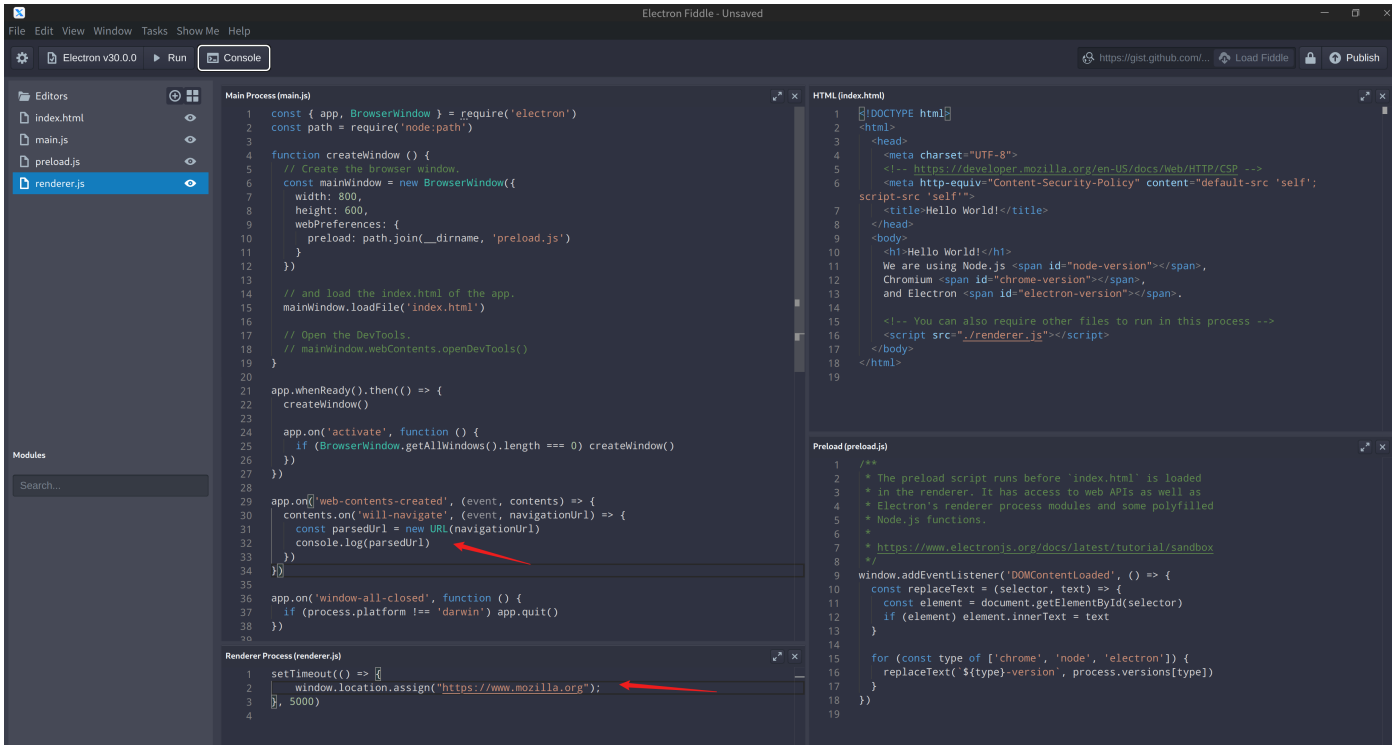


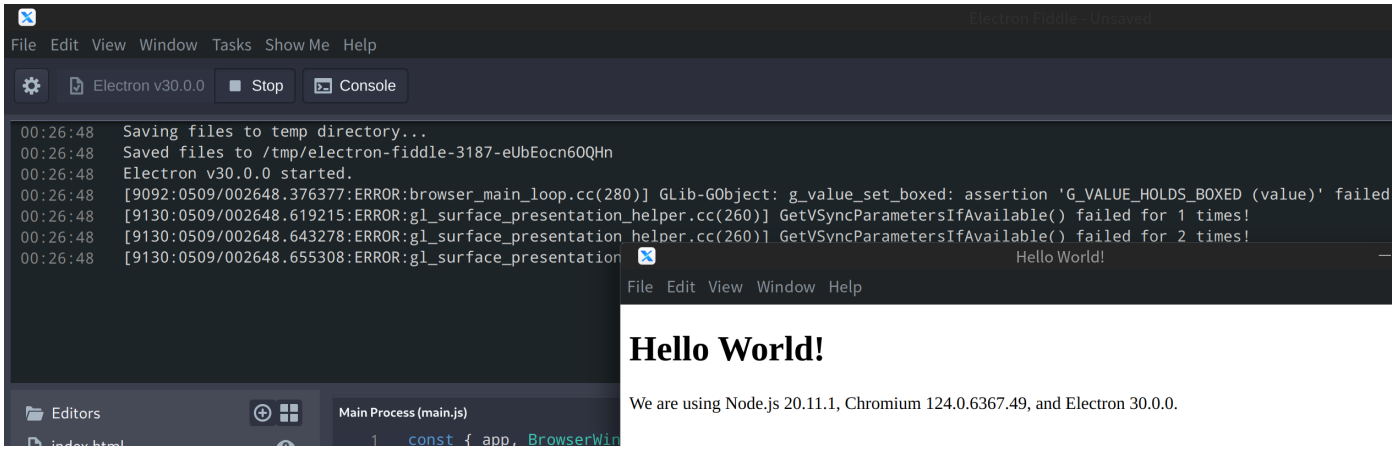
触发导航事件

2) location.assign

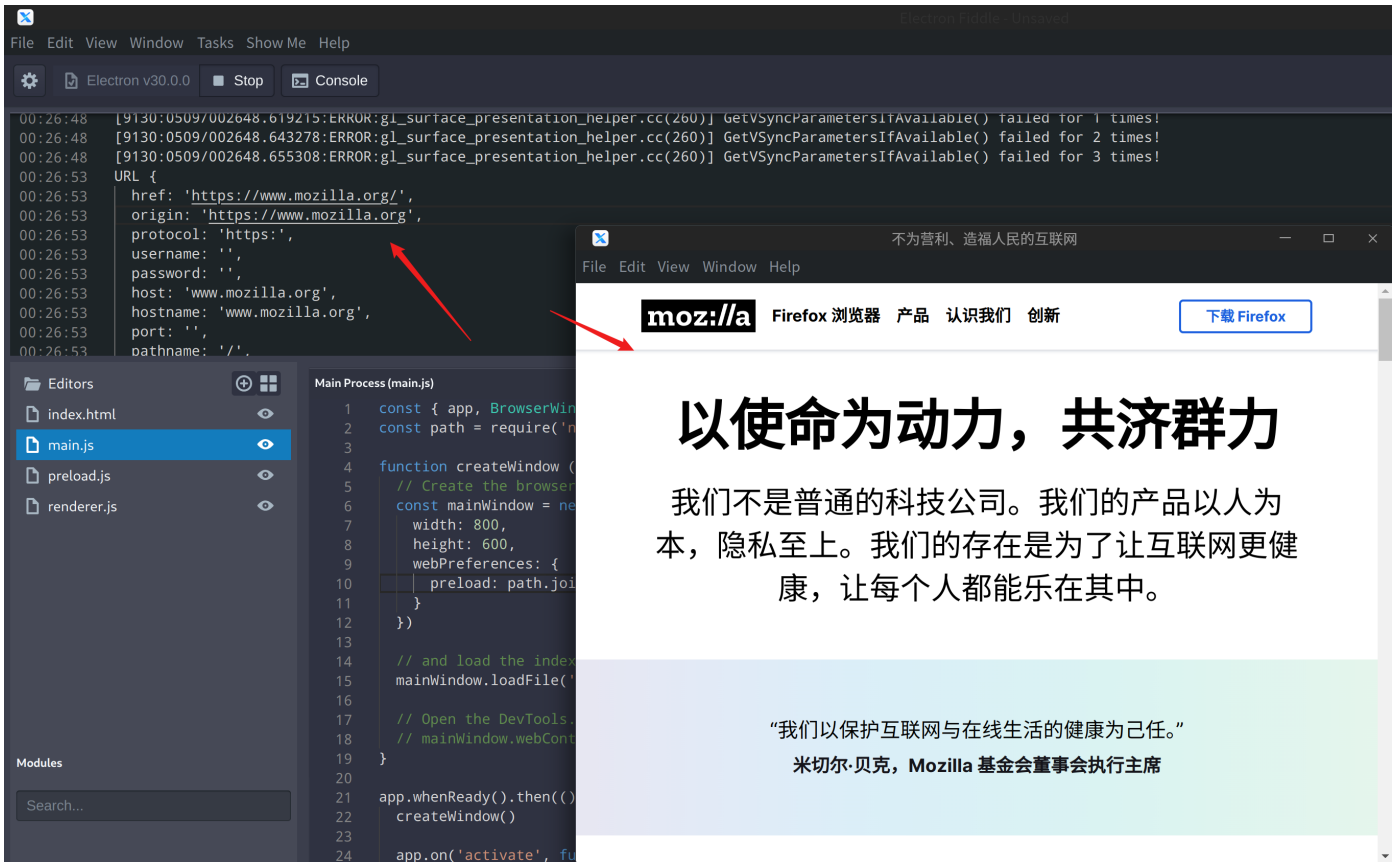
导航到一个新页面

```
window.location.assign("https://www.mozilla.org");
```





5秒后

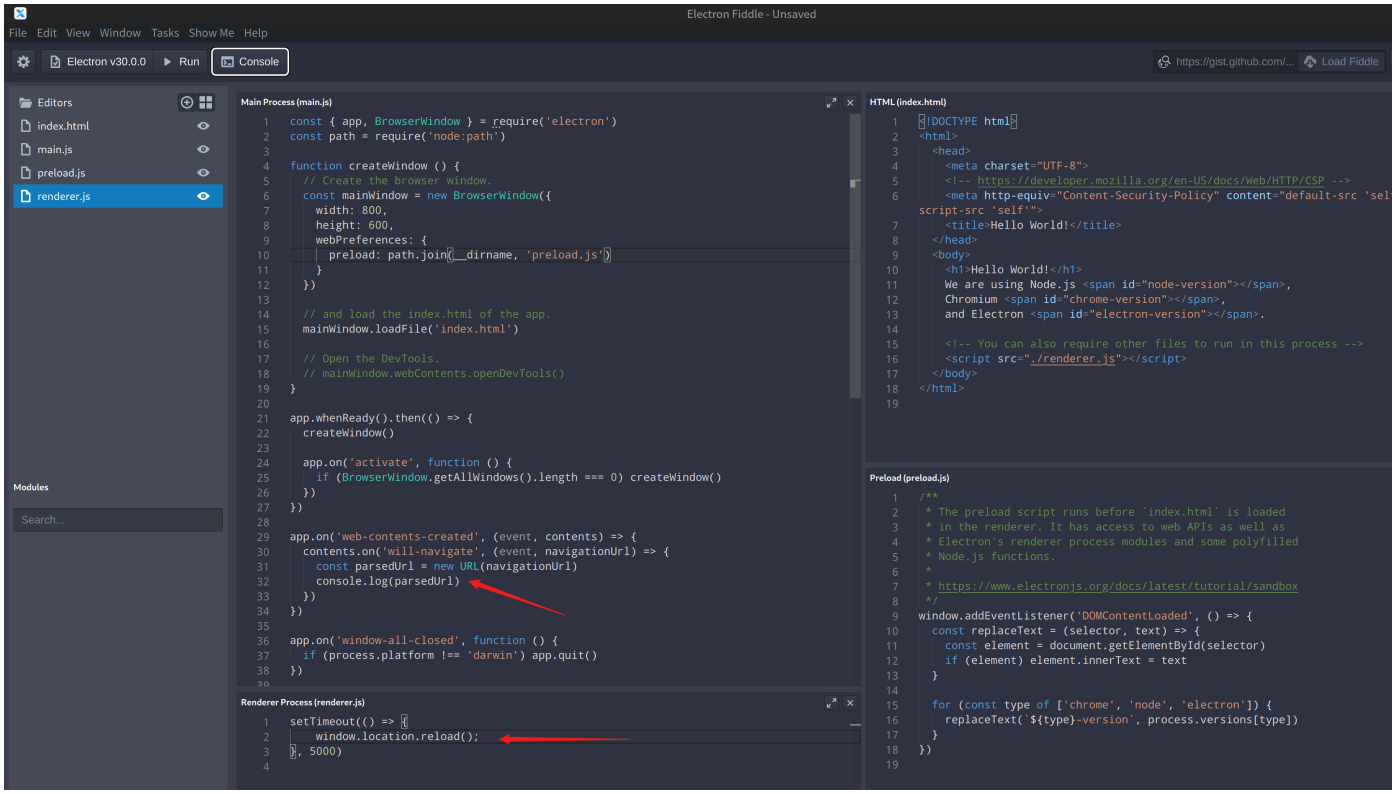


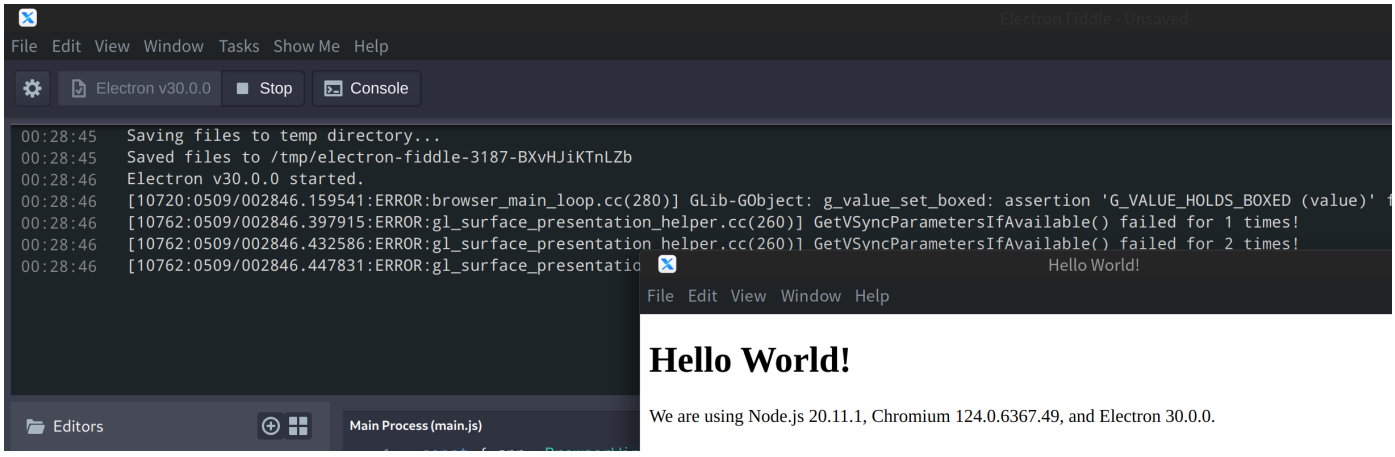
触发导航事件

3) location.reload

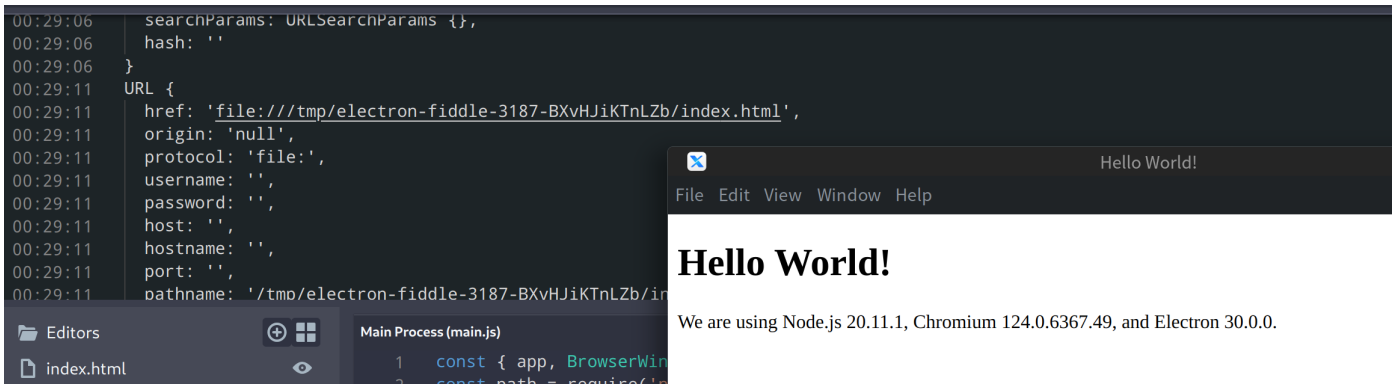
重新加载当前页面

```
window.location.reload();
```


Electron Fiddle - Unsaved
File Edit View Window Tasks Show Me Help
Electron v30.0.0
Run
Console
Editors
index.html
main.js
preload.js
renderer.js
Modules
Search...
Main Process (main.js)
const { app, BrowserWindow } = require('electron')
const path = require('node:path')
function createWindow () {
// Create the browser window.
const mainWindow = new BrowserWindow({
width: 800,
height: 600,
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()
}
app.whenReady().then(() => {
createWindow()
app.on('activate', function () {
if (BrowserWindow.getAllWindows().length === 0) createWindow()
})
})
app.on('web-contents-created', (event, contents) => {
contents.on('will-navigate', (event, navigationUrl) => {
const parsedUrl = new URL(navigationUrl)
console.log(parsedUrl)
})
})
app.on('window-all-closed', function () {
if (process.platform !== 'darwin') app.quit()
})
Renderer Process (renderer.js)
setTimeout(() => {
window.location.reload();
}, 5000)
HTML (index.html)
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<!-- https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP -->
<meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src 'self'">
<title>Hello World!</title>
</head>
<body>
<h1>Hello World!</h1>
We are using Node.js ,
Chromium ,
and Electron .
<!-- You can also require other files to run in this process -->
<script src="./renderer.js"></script>
</body>
</html>
Preload (preload.js)
/**
* The preload script runs before `index.html` is loaded
* in the renderer. It has access to web APIs as well as
* Electron's renderer process modules and some polyfilled
* Node.js functions.
*
* https://www.electronjs.org/docs/latest/tutorial/sandbox
*/
window.addEventListener('DOMContentLoaded', () => {
const replaceText = (selector, text) => {
const element = document.getElementById(selector)
if (element) element.innerText = text
}
for (const type of ['chrome', 'node', 'electron']) {
replaceText(`${type}-version`, process.versions[type])
}
})


File Edit View Window Tasks Show Me Help
Electron v30.0.0
Stop
Console
00:28:45 Saving files to temp directory...
00:28:45 Saved files to /tmp/electron-fiddle-3187-BXvHJiKTnLZb
00:28:46 Electron v30.0.0 started.
00:28:46 [10720:0509/002846.159541:ERROR:browser_main_loop.cc(280)] GLib-GObject: g_value_set_boxed: assertion 'G_VALUE_HOLDS_BOXED (value)'
00:28:46 [10762:0509/002846.397915:ERROR:gl_surface_presentation_helper.cc(260)] GetVSyncParametersIfAvailable() failed for 1 times!
00:28:46 [10762:0509/002846.432586:ERROR:gl_surface_presentation_helper.cc(260)] GetVSyncParametersIfAvailable() failed for 2 times!
00:28:46 [10762:0509/002846.447831:ERROR:gl_surface_presentatio
Editors
Main Process (main.js)
Hello World!
File Edit View Window Help
Hello World!
We are using Node.js 20.11.1, Chromium 124.0.6367.49, and Electron 30.0.0.

5秒后

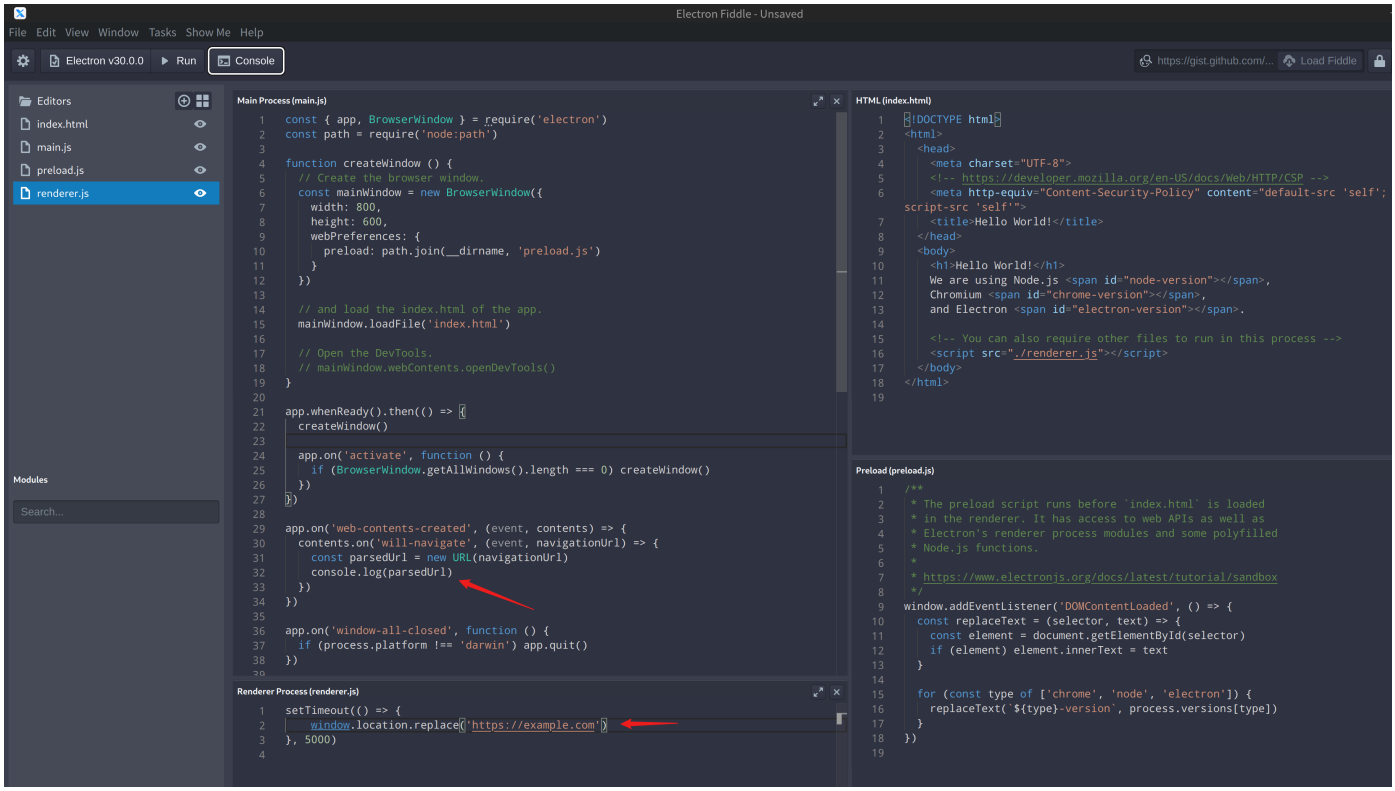

00:29:06 searchParams: URLSearchParams {},
00:29:06 hash: ''
00:29:06 }
00:29:11 URL {
00:29:11 href: 'file:///tmp/electron-fiddle-3187-BXvHJiKTnLZb/index.html',
00:29:11 origin: 'null',
00:29:11 protocol: 'file:',
00:29:11 username: '',
00:29:11 password: '',
00:29:11 host: '',
00:29:11 hostname: '',
00:29:11 port: '',
00:29:11 pathname: '/tmp/electron-fiddle-3187-BXvHJiKTnLZb/in
Editors
index.html
Main Process (main.js)
const { app, BrowserWin
Hello World!
File Edit View Window Help
Hello World!
We are using Node.js 20.11.1, Chromium 124.0.6367.49, and Electron 30.0.0.

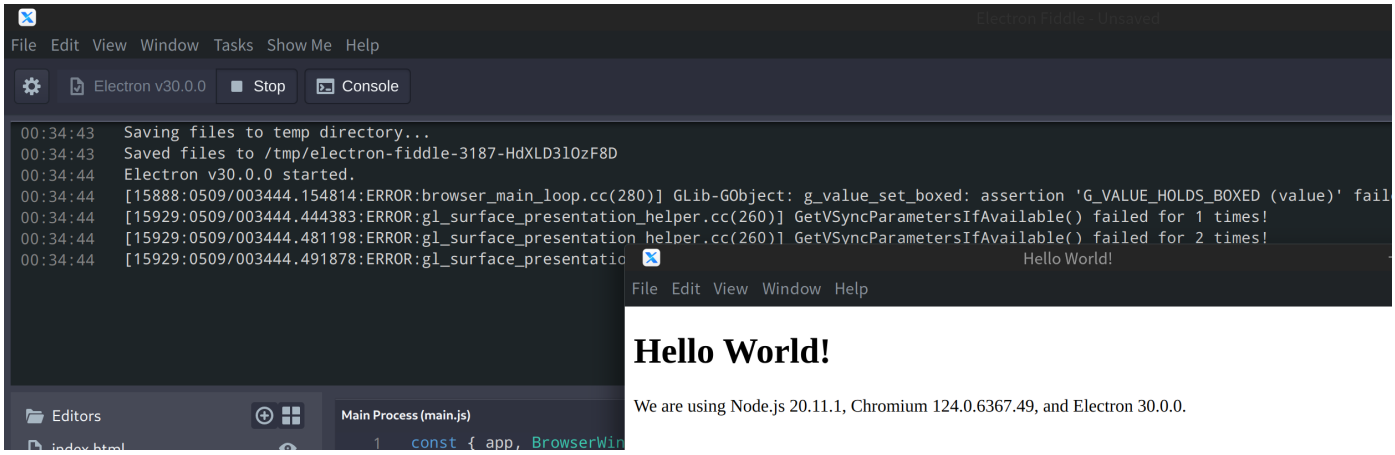
触发导航事件

4) location.replace

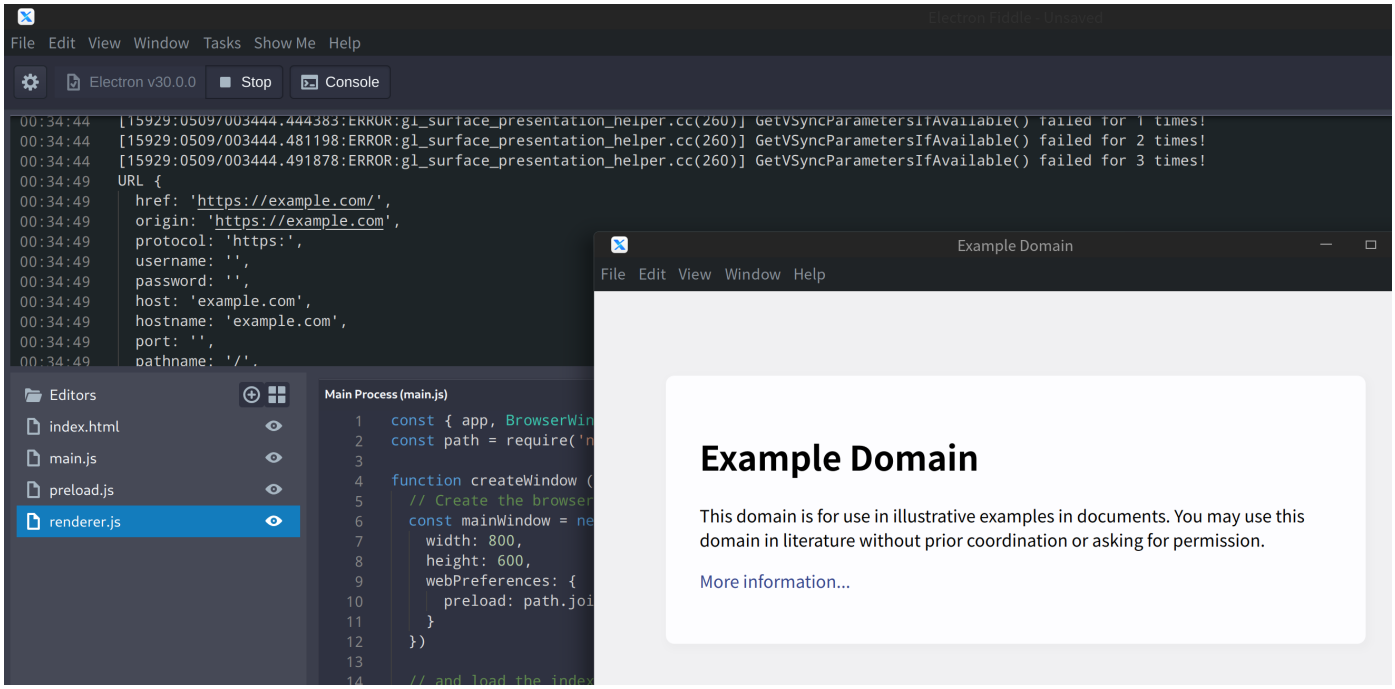
替换当前页面的 URL

```
window.location.replace('https://example.com')
```





5秒后

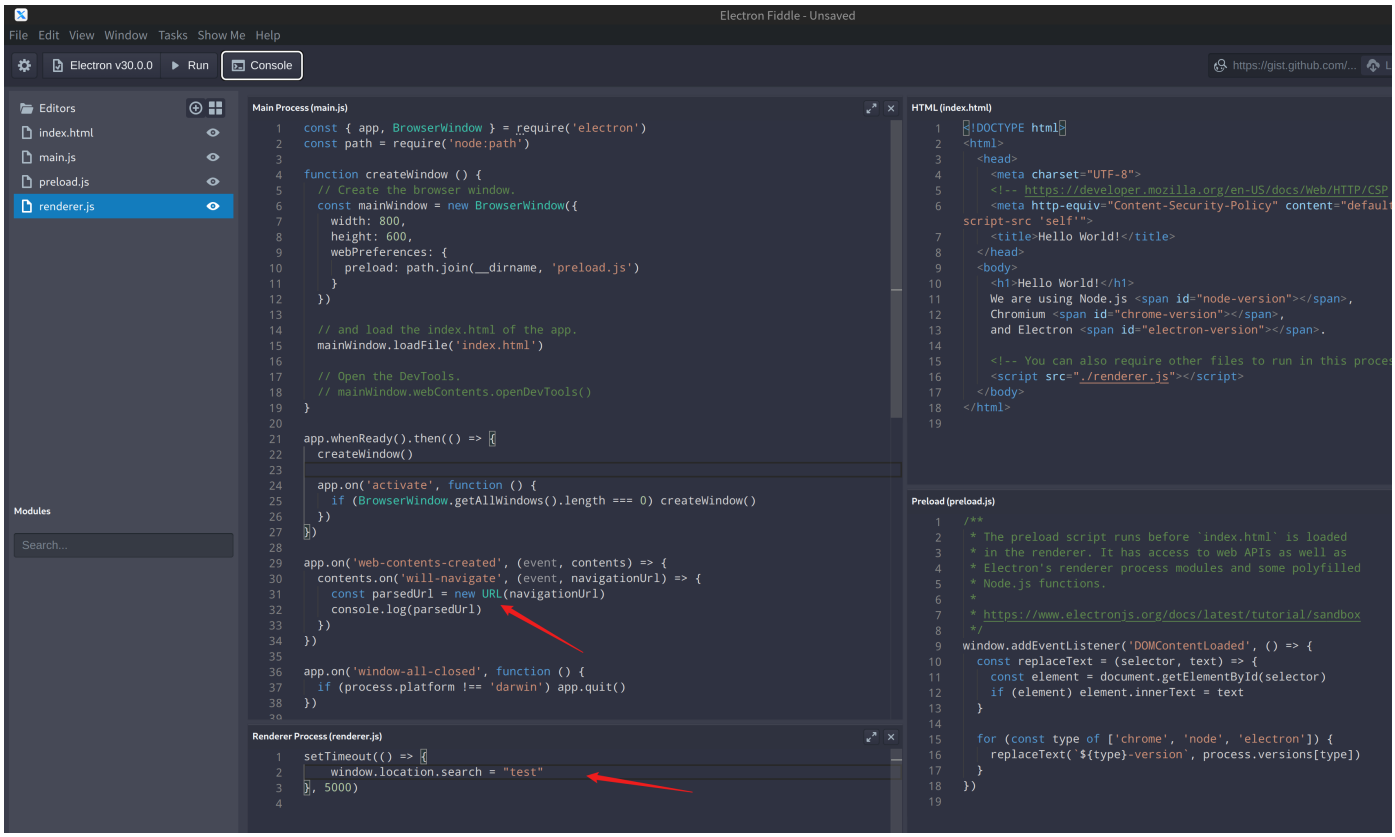


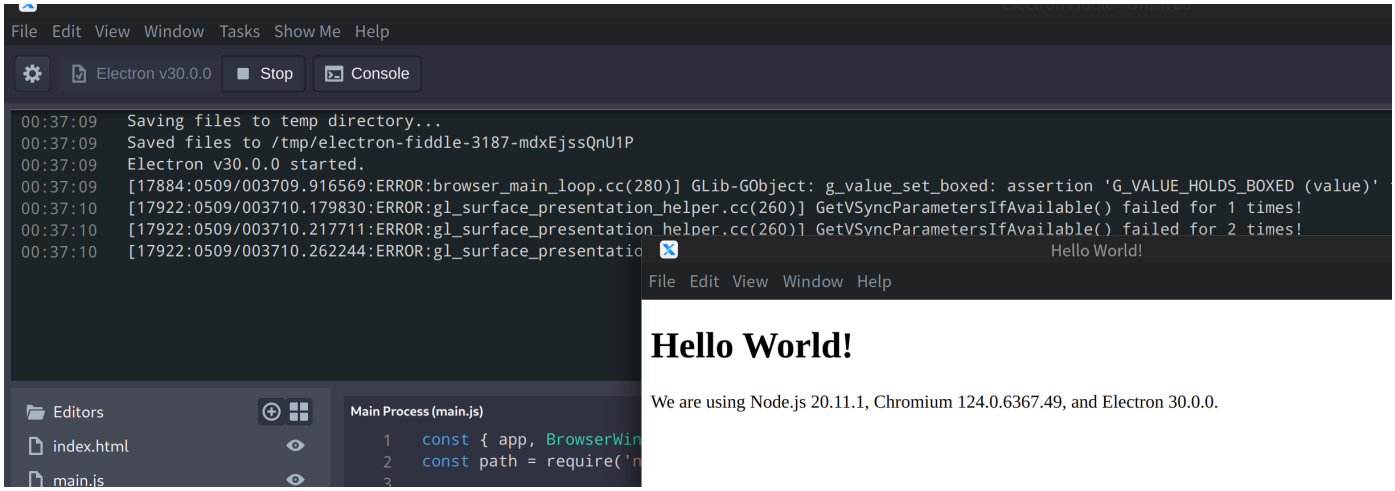
触发导航事件

5) location.search

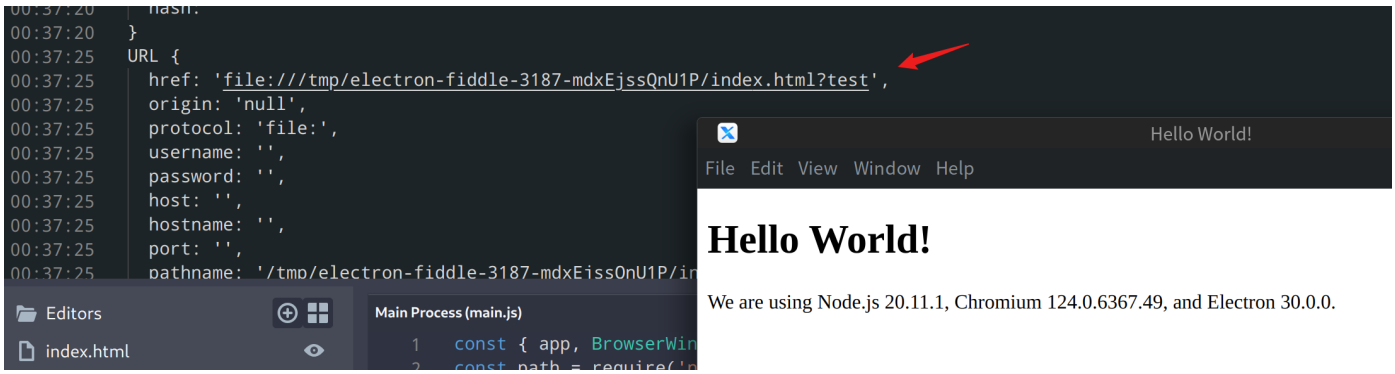
在 url 后面加上搜索字符串

```
window.location.search = "test"
```





5 秒后



6) 其他属性

属性较多，基本上都是 URL 的一部分，如果修改也会导航事件

- **href**: 返回当前页面的完整URL字符串，也可以用来设置新的URL以导航到其他页面。
- **protocol**: 返回当前URL的协议部分，例如 `http:` 或 `https:`
- **host**: 返回当前URL的主机名（域名+端口），例如 `example.com:8080`
- **hostname**: 返回当前URL的主机名（不包括端口），例如 `example.com`
- **port**: 返回当前URL的端口号，如果省略则默认端口不会显示
- **pathname**: 返回当前URL的路径部分，从根目录开始，例如 `/path/to/page.html`
- **search**: 返回URL的查询字符串部分，从问号 `?` 开始，例如 `?key=value&anotherKey=anotherValue`
- **hash**: 返回URL的哈希片段标识符（锚点），从井号 "#" 开始，例如 `#section1`
- **origin**: 返回URL的起源部分，由协议、主机名和端口组成，例如 `http://example.com:80`

6. window.history

历史记录属性可以通过其以下几个方法进行导航

- **back()**: 导航到历史记录中的上一个页面。
- **forward()**: 导航到历史记录中的下一个页面。
- **go(delta)**: 依据 `delta` 参数向前或向后导航。正值表示向前，负值表示向后，0通常不会产生导航效果但可能刷新页面。

这就相当于浏览器的前后按钮了

7. window.open

这部分上一篇文章新窗口创建的部分已经介绍了，会触发导航事件

8. window.top

`window.top` 是一个JavaScript对象属性，它引用了当前窗口或框架的最顶层窗口（即最高层级的浏览器窗口）

对于没有上层窗口的渲染进程， `window === window.top` ，所以修改 `window.top.location` 就可以修改当前页面的 url

对于 `iframe` 等子窗口，使用 `window.top.location` 可以修改顶层窗口的 URL

0x05 漏洞案例

`Masato Kinugawa` 曾经在 `Discord RCE` 的过程中利用了一个导航的漏洞 —— `CVE-2020-15174`

在 `iframe` 中，如果设置 `top.location` 的地址和 `iframe` 的地址不同源，则不会触发 `will-navigate` 事件，即导航事件，这显然是一个 bug

> https://mksben.l0.cm/2020/10/discord-desktop-rce.html

0x06 总结

网页跳转和导航的触发方法很多，但最终效果几乎都是一致的，就是在当前窗口或新窗口加载页面，在较新的版本中， `will-navigate` 能够有效地监听和阻断导航行为，开发者可以根据实际情况，考虑禁用或者限制导航行为