**快速入门**

**简介**

Electron 可以让你使用纯 JavaScript 调用丰富的原生 APIs 来创造桌面应用。你可以把它看作是专注于桌面应用而不是 web 服务器的，io.js 的一个变体。

这不意味着 Electron 是绑定了 GUI 库的 JavaScript。相反，Electron 使用 web 页面作为它的 GUI，所以你能把它看作成一个被 JavaScript 控制的，精简版的 Chromium 浏览器。

**主进程**

在 Electron 里，运行 package.json 里 main 脚本的进程被称为**主进程**。在主进程运行的脚本可以以创建 web 页面的形式展示 GUI（me：这个页面GUI由js来控制，就像控制web页面一样）。

**渲染进程**

由于 Electron 使用 Chromium 来展示页面，所以 Chromium 的多进程结构也被充分利用。每个 Electron 的页面都在运行着自己的进程，这样的进程我们称之为**渲染进程**。

在一般浏览器中，网页通常会在沙盒环境下运行，并且不允许访问原生资源。然而，Electron 用户拥有在网页中调用 io.js（node.js的一个分支） 的 APIs 的能力，可以与底层操作系统直接交互。

**主进程与渲染进程的区别**

主进程使用 BrowserWindow 实例(var { app, BrowserWindow } = require('electron')）创建网页(

var mainWindow = new BrowserWindow({

width: 1000, height: 800,

});

)。每个 BrowserWindow 实例都在自己的渲染进程里运行着一个网页(me:一个渲染进程对应一个网页，而一个王爷为一个BrowserWindow实例)。当一个 BrowserWindow 实例被销毁后，相应的渲染进程也会被终止。

主进程管理所有页面和与之对应的渲染进程。每个渲染进程都是相互独立的，并且只关心他们自己的网页。

由于在网页里管理原生 GUI 资源是非常危险而且容易造成资源泄露，所以在网页面调用 GUI 相关的 APIs 是不被允许的。如果你想在网页里使用 GUI 操作，其对应的渲染进程必须与主进程进行通讯（me:即通过ipc模块），请求主进程进行相关的 GUI 操作。

在 Electron，我们提供用于在主进程与渲染进程之间通讯的 [ipc](https://github.com/atom/electron/blob/master/docs-translations/zh-CN/api/ipc-main-process.md) 模块。并且也有一个远程进程调用风格的通讯模块 [remote](https://github.com/atom/electron/blob/master/docs-translations/zh-CN/api/remote.md)。

**打造你第一个 Electron 应用**

大体上，一个 Electron 应用的目录结构如下：

your-app/

├── package.json

├── main.js

└── index.html

package.json的格式和 Node 的完全一致，并且那个被 main 字段声明的脚本文件是你的应用的启动脚本，它运行在主进程上。你应用里的 package.json 看起来应该像： （json）

{

"name" : "your-app",

"version" : "0.1.0",

"main" : "main.js"

}

**注意**：如果 main 字段没有在 package.json 声明，Electron会优先加载 index.js。

main.js 应该用于创建窗口和处理系统时间，一个典型的例子如下： ```javascript

~~var app = require('app'); // 控制应用生命周期的模块。~~

~~var BrowserWindow = require('browser-window'); // 创建原生浏览器窗口的模块~~

~~var mainWindow = null;// 保持一个对于 window 对象的全局引用，不然，当 JavaScript 被 GC， // window 会被自动地关闭~~

~~// 当所有窗口被关闭了，退出。~~

~~app.on('window-all-closed', function() { // 在 OS X 上，通常用户在明确地按下 Cmd + Q 之前，应用会保持活动状态~~

~~if (process.platform != 'darwin')~~

~~{~~

~~app.quit();~~

~~}~~

~~});~~

~~// 当 Electron 完成了初始化并且准备创建浏览器窗口的时候~~

~~// 这个方法就被调用~~

~~app.on('ready', function() { // 创建浏览器窗口。~~

~~mainWindow = new BrowserWindow({~~

~~width: 800,~~

~~height: 600~~

~~});~~

~~// 加载应用的 index.html~~

~~mainWindow.loadURL('file://' + \_\_dirname + '/index.html');~~

~~// 打开开发工具~~

~~mainWindow.openDevTools();~~

~~// 当 window 被关闭，这个事件会被发出~~

~~mainWindow.on('closed', function() { // 取消引用 window 对象，如果你的应用支持多窗口的话， 通常会把多个 window 对象存放在一个数组里面， 但这次不是。~~

~~mainWindow = null;~~

~~});~~

~~});~~

**当前版本下使用这个：**

const {app,BrowserWindow}=require('electron');

app.on('ready', function() { // 创建浏览器窗口。

mainWindow = new BrowserWindow({

width: 800,

height: 600

});

// 加载应用的 index.html

mainWindow.loadURL('file://' + \_\_dirname + '/index.html');

// 打开开发工具

mainWindow.openDevTools();

// 当 window 被关闭，这个事件会被发出

mainWindow.on('closed', function() { // 取消引用 window 对象，如果你的应用支持多窗口的话， 通常会把多个 window 对象存放在一个数组里面， 但这次不是。

mainWindow = null;

});

});

 最后，你想展示的 `index.html` ：

```html

<!DOCTYPE html>

<html>

<head>

</head>

<body>

<h1>Hello World!</h1>

<p>We are using io.js and Electron .</p>

</body >

</html>

```

**运行你的应用**

一旦你创建了最初的 main.js， index.html 和 package.json 这几个文件，你可能会想尝试在本地运行并测试，看看是不是和期望的那样正常运行。

**electron-prebuild**

如果你已经用 npm 全局安装了 electron-prebuilt，你只需要按照如下方式直接运行你的应用： bash electron . 如果你是局部安装，那运行： bash ./node\_modules/.bin/electron .

**手工下载 Electron 二进制文件**

如果你手工下载了 Electron 的二进制文件，你也可以直接使用其中的二进制文件直接运行你的应用。

**Windows**

bash $ .\electron\electron.exe your-app\

**Linux**

bash $ ./electron/electron your-app/

**OS X**

bash $ ./Electron.app/Contents/MacOS/Electron your-app/ Electron.app 里面是 Electron 发布包，你可以在[这里](https://github.com/atom/electron/releases)下载到。

**以发行版本运行**

在你完成了你的应用后，你可以按照[应用部署](https://github.com/atom/electron/blob/master/docs-translations/zh-CN/tutorial/application-distribution.md)指导发布一个版本，并且以已经打包好的形式运行应用。