

# 使用electron构建跨平台Node.js桌面应用经验分享

这篇文章发布于 2017年05月16日, 星期二, 01:05, 归类于 JS相关。 阅读 45073 次, 今日 33 次 25 条评论

by zhangxinxu from <http://www.zhangxinxu.com/wordpress/?p=6154>

本文可全文转载, 但需得到原作者书面许可, 同时保留原作者和出处, 摘要引则随意。

最近, 把团队内经常使用的一个基于Node.js制作的小工具给做成了可视化操作的桌面软件, 使用的是 **electron**, 这里简单分享一下使用 **electron** 的一些经验和心得。

## 一、如何使用electron把基本的开发环境给跑起来?

我是这么处理的, **electron** 官方提供了一个名为“electron-quick-start”的示例项目, 地址为: <https://github.com/electron/electron-quick-start>

然后把相关资源给弄下来, 如果你是下载Zip包解压的, 则资源默认都会放在一个名为“electron-quick-start-master”的文件夹中, 把“electron-quick-start-master”改成你项目的名字, 当然你不改也没关系, 就怕过段时间忘记, 然后小手一抖, 当做普通资源给删掉了, 到时候就男默女泪了。

然后安装:

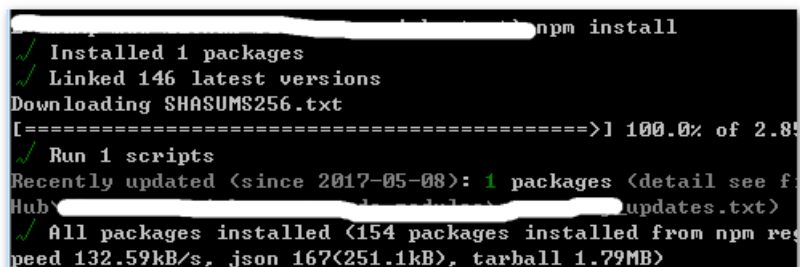
```
npm install
```

由于安装包比较大, 所以 **-\** 要转好几分钟才能装好。如果安装不顺利, 试试换成使用淘宝NPM镜像:

```
npm install -g cnpm --registry=https://registry.npm.taobao.org
```

然后再这么安装:

```
cnpm install
```

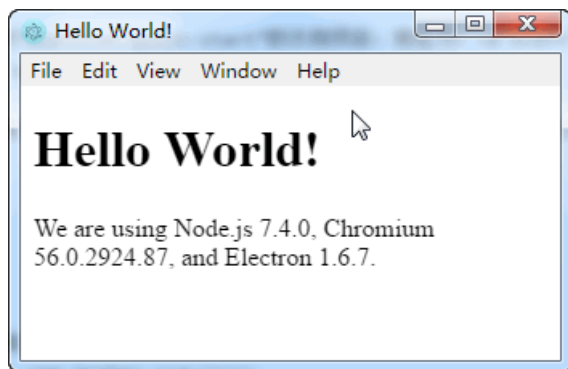


```
npm install
✓ Installed 1 packages
✓ Linked 146 latest versions
Downloading SHASUMS256.txt
[=====>] 100.0% of 2.85
✓ Run 1 scripts
Recently updated (since 2017-05-08): 1 packages (detail see f
Hub updates.txt)
✓ All packages installed (154 packages installed from npm reg
speed 132.59kB/s, json 167<251.1kB>, tarball 1.79MB)
```

然后启动:

```
npm start
```

然后就会出现这样的框框:



环境就这么跑起来了。

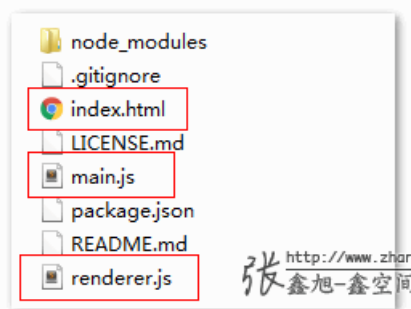
**Ctrl** + **R** 就可以刷新。

## 二、electron开发该怎么入手?

一旦环境跑起来，接下来的工作就跟做一个网页几乎就没什么区别，加载点CSS，图片啊，JS什么的，就可以了。因为本质上，`electron` 就是给你搞了一个Chrome浏览器的壳子，只是比传统网页多了一个访问桌面文件的功能。

当然，具体操作并不可能像嘴皮子动的那么简单，前期还是需要了解一些基础知识。

我们可以重点关注一下上一节安装好的开发环境的一些资源文件，主要是 `index.html`，`main.js` 和 `renderer.js`，如下图：



在我看来，如果我们要开发的桌面应用只要不像QQ软件那样复杂，其实可以完全不用管`main.js`，`main.js`的作用就是用来显示 `npm start` 后出现的那个窗口的，而我们的工作只是窗口里面内容，因此，`main.js` 无需关心。

`index.html` 是打开的窗口加载的页面，可以看成是入口页面，就是一个普通的静态页面啊，没什么特殊的。

`renderer.js` 默认里面就一堆注释，用来放业务相关JS的，和网页JS的区别在于，这里的JS不仅可以访问DOM，还能使用Node.js所有的API。能前能后，想怎么玩都行。

所以，我们的工作思路就很清晰了：

1. 先把我们桌面应用的可视窗口界面给弄出来，这个就需要使用CSS和HTML代码了。相比网页开发而言，开发桌面应用要更轻松，因为根本不要考虑兼容性的问题，而且很多最新的chrome特性，都可以也很愉快的玩起来。我们的CSS代码可以外链线上的资源，也可以放到本地，也可以直接内联在页面中，非常自由，非常随意啊，都可以。我个人建议是放在本地的，因为就算断网了我们的桌面应用也能正常使用。

假设一番折腾，我们的界面弄好了，类似这样：

接下来就是折腾交互了。

- 交互效果开发和传统web网站一样，很自由，你喜欢jQuery，就用jQuery，你喜欢Vue，也可以使用Vue等等，没有什么顾虑，就是干！

例如我给团队做的这个桌面应用就是用的jQuery，最后应用跑得很畅快。

- 借助Node.js API或者其他第三方的npm工具或者 `electron` API开发我们的应用。

例如，引入Node.js API：

```
const fs = require('fs');
stat = fs.stat;

const path = require('path');
const url = require('url');
```

引入第三方库：

```
const minify = require('html-minifier').minify;
```

等等。

例如我做这个桌面应用有需要选择本地文件夹的功能，这个时候就需要借助 `electron` API，由于我们的业务JS都是写在 `renderer.js` 中的，并非主线程，因此，调用的使用要使用 `remote`，例如：

```
const electron = require('electron');

const dialog = electron.remote.dialog;
```

此时，我们想要点击按钮打开系统的选择文件夹弹框就可以这么处理：

```
dialog.showOpenDialog({
  properties: ['openDirectory', 'createDirectory']
}, callback);
```

具体可参考 `electron` API文档，有中文版。

于是，简简单单的三步曲，我们的桌面应用功能就开发好了，逻辑还是以前Node.js工具的逻辑，多的仅是可视化的界面，以及参数是从输入框等表单控件获取。

开发的过程要比之前预估的要轻松得多，那种随便怎么玩都支持的感觉真的很美妙。

### 三、electron开发好的应用该如何发布？

`electron` 桌面在自己的开发环境下跑起来了，跑通了，如让其他小伙伴也能方便快捷地使用呢？我们的目标是windows系统下直接点击个 `.exe` 文件，Mac OS X直接点击 `.app` 文件就可以跑起来，我们的小伙伴无需再麻烦安装一堆node modules。

我们需要使用专门的打包工具，我是使用的 `electron-packager`，首先全局安装一下：

```
npm install electron-packager -g
```

然后就可以执行打包了，例如：

```
electron-packager . bobo --out ../electron
```

这段语句表示的意思是把当前文件目录下的资源（`.`）命名为 `bobo` 打包到父级的 `electron` 文件夹。

此时 `electron-packager` 就会自动判别当前的操作系统打包对应的文件，例如windows系统下就会打包成 `.exe` 格式。

如果你想一次性把所有的操作系统都打包一遍，可以在上面打包语句后面加上 `-all`。

由于打包的时候会把浏览器内核完整打包进去，所以就算你的项目开发就几百k的资源，但最终的打包文件估计有40到50M。

然后有一点需要特别注意一下，如果你开发的桌面要有第三方的 `npm` 模块依赖，则你打包好的运行文件无论是跑不起来的，有打包的时候并不会把第三方的 `npm` 模块依赖也打不进去，需要自己手动复制进去。我的做法是把第三方依赖的 `npm` 模块打包成一个名为 `require-node_modules.zip` 文件夹，此时这个文件会一起被打包带走，一同被放在 `app` 文件夹下，具体路径为：

windows: \resources\app

OS X: 显示包内容 → \Contents\Resources\app

此时，只要直接解压就可以使用了。

补充于翌日

感谢@小小瘦杜的纠正，实际上，是可以直接打包第三方的 `npm` 模块的，就是安装依赖的时候不要 `--save-dev`。

## 四、electron发布好的桌面应用如何有效升级？

我们平常的桌面软件要升级的话，一般都需要下载完整的安装包。`electron` 作为桌面应用，似乎也逃脱不了这种宿命，但实际上，在绝大部分场景下，我们根本就无需要下载完整的安装包，因为 `electron-packager` 打包的其实是浏览器内核和主线程控制脚本，具体的业务代码全部都是独立在 `app` 文件夹下的，也就是说，只要我们的桌面应用主线程逻辑不变，什么UI样式调整，什么交互效果改变，什么业务逻辑变更，我们都只要更新 `app` 文件夹下的这资源就可以了：

windows: \resources\app

OS X: 显示包内容 → \Contents\Resources\app

例如，我们的 `renderer.js` 做了一些升级改动，此时我们的小伙伴想要更新怎么办，无需再重新发布一个安装包，直接把 `app` 文件夹下 `renderer.js` 切换一下就好了，非常简单和快捷。

甚至如果有精力的话，我们桌面应用可以做成自动检测是否有版本更新以及在线升级，升级的内容就是CSS，HTML，image或者JS这些静态资源。

更新于2017-06-27

实际上，上面的更新升级还是很麻烦，经过我的实践和数周的体验，是可以实现稳定的热更新功能的，

具体看参见这篇文章：“[我是如何实现electron的热更新功能的？](#)”。

## 五、结束语

有了 `electron`，理论上所有基于的Node.js的工具都可以桌面化，例如，小图标合并啊，图片压缩呀等等。

而且从本文的描述来看，基本上没什么难度，其实大家都可以搞一搞，因为对于设计师和小白开发人员而言，他们还是更喜欢使用可视化的东西，这其实对于提高团队效率还是很有帮助的，比如那个设计师做了一个图片压缩或者合成的可视化工具，人家自然是很是欢喜的。

好了，就这些，以后再想到什么遗漏的再补充上去。

《CSS世界》签名版独家发售，包邮，可指定寄语，点击显示购买码

(本篇完) // 想要打赏? 点击[这里](#)。有话要说? 点击[这里](#)。



« [HTML accesskey属性与web自定义键盘快捷访问](#)

[小tips:使用canvas在前端实现图片水印合成](#) »

猜你喜欢

- 我是如何实现electron的在线升级热更新功能的?
- JS一般般的网页重构可以使用Node.js做些什么
- 小tip: 我是如何初体验uglifyjs压缩JS的
- Stylus-NodeJS下构建更富表现力/动态/健壮的CSS
- 高富帅seajs使用示例及spm合并压缩工具露脸
- Service Worker实现浏览器端面渲染或CSS,JS编译
- windows系统下批量删除OS X系统.DS\_Store文件

分享到: 1

标签: electron, electron-packager, node, nodejs

### 发表评论（目前25条评论）

名称 (必须)

邮件地址(不会被公开) (必须)

网站

提交评论

1. 哈压库说道:

2018年08月31日 16:58

saodisilai

[回复](#)



2. boss说道:

2017年12月20日 18:30

我想问的是，如何在前端浏览器调用启动本地安装的electron程序，并传值过去

[回复](#)



3. now说道:

2017年12月11日 11:29

打包后一些引入的模块找不到不能使用是什么情况；

[回复](#)



4. qiaoqiao说道:

2017年10月6日 16:36

我把我们做好的网站直接通过window.location.href='mywebsite'; 放在index.html 中请问为什么不能操作呢？

[回复](#)



5. fexp说道:

2017年09月19日 15:35

不知道为什么 发现打包的windows版本可以使用 mac版本打包出来不能使用

[回复](#)



zsai说道:

2017年10月22日 18:43

想要打包对应平台的应用只能在相应平台上打包

[回复](#)



6. 倾诉,...说道:

2017年09月8日 17:28

在mac对electron进行打包成.exe, 如何将包变成安装程序呢,electron-winstaller没看明白怎么用呢？

[回复](#)



7.

llfylwg说道:

2017年09月4日 21:21

升级这块感觉有点问题，升级的话，开发者发布的肯定是一个已经打包成exe的安装包了，怎么提取app里的东西替换本地的？

回复

张鑫旭说道:

2017年09月6日 10:51

实际上electron打包出来的.exe更像是一个快捷方式，资源都是解压可访问的。

回复


8.

清凌渡说道:

2017年07月2日 18:28

打包推荐electron-builder, 包含electron-packager全部功能，并且还增加了自动更新、多平台打包等功能。

回复


9.

awen说道:

2017年06月2日 22:46

挺好

回复


10.

DH说道:

2017年05月23日 18:01

看了您这篇文章，我就做了个小工具练手，对前端来说electron确实是福音。

回复

gdp说道:

2017年06月8日 16:09

你做的什么小工具，可以问下吗

回复




11.

神秘博士说道:

2017年05月23日 17:33

<https://segmentfault.com/q/1010000007594059>  
这是安装 electron 卡在 node install.js 的解决办法

回复

qiaoqiao说道:

2017年10月6日 16:37

直接换成cnpm 就行了

回复




12.

exoticknight说道:

2017年05月21日 14:42

想问一下，第三方依赖打包的方法，运行的时候需要自己再手动解压吗？另外 electron 能直接读取 asar 打包的格式，是否也能从这里考虑呢？

回复



13. **qqa**说道:

2017年05月16日 17:14

老师您好, 如果可以的话能否在以后讲一讲关于electron的API的使用?

[回复](#)



14. **zhoukekestar**说道:

2017年05月16日 09:55

我之前也写了一下, 最头疼的就是升级, 为了能像web一样, 发布即升级。所以我就只加载web资源, 所有的资源都发布在web服务器上。

图片:

![demo](https://cloud.githubusercontent.com/assets/7157346/26086322/9430d54e-3a1c-11e7-8e1f-dfb35e797331.png)

于是, 我就有一个拥有`native`能力的网页了。js中就可以这么写:

```
“`js
const remote = require('electron').remote
const fs = remote.require('fs');
“`
```

不仅没有了浏览器兼容性, 也没有升级的烦恼(可以实时发布网页到服务器, 本地就更新了), 而且网页拥有了“原生”的能力(能调用fs读取文件)。

当然这种方案安全是一个大问题, 所以, 做好安全加载会显得比较重要。毕竟, 要是网页被黑了, 权限接近无限大。

[回复](#)



**mfk**说道:

2017年06月6日 09:35

好办法。不建议调用原生功能。  
感觉就是加了个外壳并内置Chrome浏览器, 同时还需要node.js运行环境。  
还不如直接让别人把网页发送到桌面快捷方式。

[回复](#)



15. **nextnext**说道:

2017年05月16日 09:19

张老师你好! 一直看着博客成长起来的, 有个问题想问您一下。  
前端的意义在哪?

<https://www.zhihu.com/question/44812950>

这个知乎上讨论的人很多但是越是这样就越纠结, 感觉那些大神说的都好好有道理, 不知道该听谁的。  
谢谢您了!

[回复](#)



16. **meepo**说道:

2017年05月16日 08:40

很久以前有一个node-webkit, 不知道是不是同一个东西.

[回复](#)

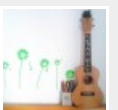


**dont**说道:

2017年05月16日 10:03

node-webkit好像早更名nw.js了...  
<https://www.zhihu.com/question/36292298/answer/102418523>

[回复](#)



**jScript**说道:

2017年05月16日 10:24

nw.js

跟electron.js都差不多





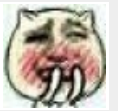
[回复](#)

charten说道:

2017年05月16日 21:44

这两个都是同一个作者鼓捣出来的

[回复](#)



exoticknight说道:

2017年05月21日 14:41

不是，并且指导思想并不一样

[回复](#)



### 最新文章

- » [常见的CSS图形绘制合集](#)
- » [粉丝群第1期CSS小测点评与答疑](#)
- » [分享三个纯CSS实现26个英文字母的案例](#)
- » [小tips: 纯CSS实现打字动画效果](#)
- » [CSS/CSS3 box-decoration-break属性简介](#)
- » [CSS :placeholder-shown伪类实现Material Design占位符交互效果](#)
- » [从天猫某活动视频不必要的3次请求说起](#)
- » [CSS vector-effect与SVG stroke描边缩放](#)
- » [CSS ::backdrop伪元素是干嘛用的?](#)
- » [周知: CSS -webkit-伪元素选择器不再导致整行无效](#)

### 今日热门

- » [常见的CSS图形绘制合集](#) <sup>(190)</sup>
- » [未来必热: SVG Sprite技术介绍](#) <sup>(119)</sup>
- » [粉丝群第1期CSS小测点评与答疑](#) <sup>(115)</sup>
- » [HTML5终极备忘大全 \(图片版+文字版\)](#) <sup>(93)</sup>
- » [让所有浏览器支持HTML5 video视频标签](#) <sup>(86)</sup>
- » [Selectivizr-让IE6~8支持CSS3伪类和属性选择器](#) <sup>(82)</sup>
- » [CSS3下的147个颜色名称及对应颜色值](#) <sup>(79)</sup>
- » [视区相关单位vw, vh..简介以及可实际应用场景](#) <sup>(76)</sup>
- » [写给自己看的display: flex布局教程](#) <sup>(76)</sup>
- » [小tips: 纯CSS实现打字动画效果](#) <sup>(76)</sup>

### 今年热议

- » [《CSS世界》女主角诚寻靠谱一起奋斗之人](#) <sup>(76)</sup>
- » [不借助Echarts等图形框架原生JS快速实现折线图效果](#) <sup>(64)</sup>
- » [看, for..in和for..of在那里吵架!](#) <sup>(60)</sup>
- » [是时候好好安利下LuLu UI框架了!](#) <sup>(47)</sup>
- » [原来浏览器原生支持JS Base64编码解码](#) <sup>(35)</sup>
- » [妙法攻略: 渐变虚框及边框滚动动画的纯CSS实现](#) <sup>(33)</sup>
- » [炫酷H5中序列图片视频化播放的高性能实现](#) <sup>(31)</sup>

- » [CSS scroll-behavior和JS scrollIntoView让页面滚动平滑](#) <sup>(30)</sup>
- » [windows系统下批量删除OS X系统.DS\\_Store文件](#) <sup>(26)</sup>
- » [写给自己看的display: flex布局教程](#) <sup>(26)</sup>

## 猜你喜欢

- [我是如何实现electron的在线升级热更新功能的?](#)
- [JS一般般的网页重构可以使用Node.js做些什么](#)
- [小tip: 我是如何初体验uglifyjs压缩JS的](#)
- [Stylus-NodeJS下构建更富表现力/动态/健壮的CSS](#)
- [高富帅seajs使用示例及spm合并压缩工具露脸](#)
- [Service Worker实现浏览器端页面渲染或CSS,JS编译](#)
- [windows系统下批量删除OS X系统.DS\\_Store文件](#)