Electron集成Sqlite3

> Node环境支持

安装Node环境

electron基于Node环境，下载node安装包并且安装，https://nodejs.org/en/ (Node.js网站 建议安装node-v8.11.1 以上版本，8版本以下不支持electron)。

npm切换taobao镜像源

```
# 安装nrm 管理，安装这个的前提是能确定链接到现在的镜像地址：
npm install -g nrm

# 使用nrm 切换镜像地址：
nrm use taobao

# nrm 除了淘宝站点镜像之外还有其他的，可以使用一下命令查看：
nrm ls

```

下载搭建Electron依赖环境

```
#创建electron项目文件夹 使用VSCode打开文件夹 control键+~ 呼出VSCode终端
# 要安装预编译好的的二进制文件，请使用 npm。 首选的方法是在项目中作为development dependency安装。下载完成会在左侧出现node_modules依赖包文件夹以及package-lock.json文件
npm install electron --save-dev

# 官网安装教程 http://electronjs.org/docs/tutorial/installation
```

```
E:\electron搭建教程>npm install electron --save-dev

> electron@3.0.10 postinstall E:\electron搭建教程\node_modules\electron
> node install.js

npm WARN saveError ENOENT: no such file or directory, open 'E:\electron搭建教程\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'E:\electron搭建教程\package.json'
npm WARN electron搭建教程 No description
npm WARN electron搭建教程 No repository field.
npm WARN electron搭建教程 No README data
npm WARN electron搭建教程 No license field.

+ electron@3.0.10
added 145 packages from 126 contributors in 22.108s

E:\electron搭建教程>
```

检测是否成功安装electron环境

```
# 终端运行electron命令检测是否安装成功(弹出electron窗口为electron环境搭建成功)
electron

# 或者运行electron -v 查看Electron版本号
electron -v

```

创建你的第一个Electron项目

继以上操作继续,准备好一个HTML文件和一个main.js文件作为项目我们一个demo 并且将这两个文件放入我们当前程序文件夹中

index.html

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>My First Electron Demo</title>
</head>
<body>
    <span>我的第一个项目</span>
</body>
</html>
```

main.js

```
const {app, BrowserWindow} = require('electron')
const path = require('path')
const url = require('url')

// 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 win

function createWindow () {
  // Create the browser window.
  win = new BrowserWindow({width: 1920, height: 1080,
          webPreferences: {
            nodeIntegration: false
          }
  })

  // and load the index.html of the app.
  win.loadURL(url.format({
    pathname: path.join(__dirname, 'index.html'),
    protocol: 'file:',
    slashes: true
  }))

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

  // Emitted when the window is closed.
  win.on('closed', () => {
    // Dereference the window object, usually you would store windows
    // in an array if your app supports multi windows, this is the time
    // when you should delete the corresponding element.
    win = null
  })
}

// This method will be called when Electron has finished
// initialization and is ready to create browser windows.
// Some APIs can only be used after this event occurs.
app.on('ready', createWindow)

// Quit when all windows are closed.
app.on('window-all-closed', () => {
  // On macOS it is common for applications and their menu bar
  // to stay active until the user quits explicitly with Cmd + Q
  if (process.platform !== 'darwin') {
    app.quit()
  }
})

app.on('activate', () => {
  // On macOS it's common to re-create a window in the app when the
  // dock icon is clicked and there are no other windows open.
  if (win === null) {
```

```
    createWindow()
  }
})
```

```
# 使用npm init 命令创建 根据终端提示完成创建
npm init
```

```
E:\electron搭建教程>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (electron搭建教程) my_electron
version: (1.0.0) 1.0.0
description: the first electron
entry point: (index.js) main.js
test command:
git repository:
keywords:
author: songxiaochen
license: (ISC)
About to write to E:\electron搭建教程\package.json:

{
  "name": "my_electron",
  "version": "1.0.0",
  "description": "the first electron",
  "main": "main.js",
  "dependencies": {
    "electron": "^3.0.10"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "songxiaochen",
  "license": "ISC"
}


Is this OK? (yes) y
```

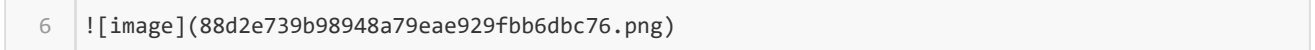

```
# 创建完成后运行命令electron . 启动我们的项目

electron .
```

> 启动项目命令替换

```
// 将package.json文件scripts 下增加启动命令start
"scripts": {
    "start": "electron ."
  }
```

```
# 替换package.json 启动命令后以后将使用以下命令进行启动项目
npm run start
````

出现以下窗口表示成功


集成Sqlite3开始

继以上操作继续

``` bash
# 安装 electron-builder 依赖
npm install --save-dev electron-builder

# 安装Sqlite3 依赖
npm install --save sqlite3
```

```
// 将package.json文件scripts 下增加启动命令 postinstall
"scripts": {
    "start": "electron .",
    "postinstall": "install-app-deps"
  }
```

```

# 运行命令
npm run postinstall
```

将js文件内容进行修改调用Sqlite3组件并且创建DB连接

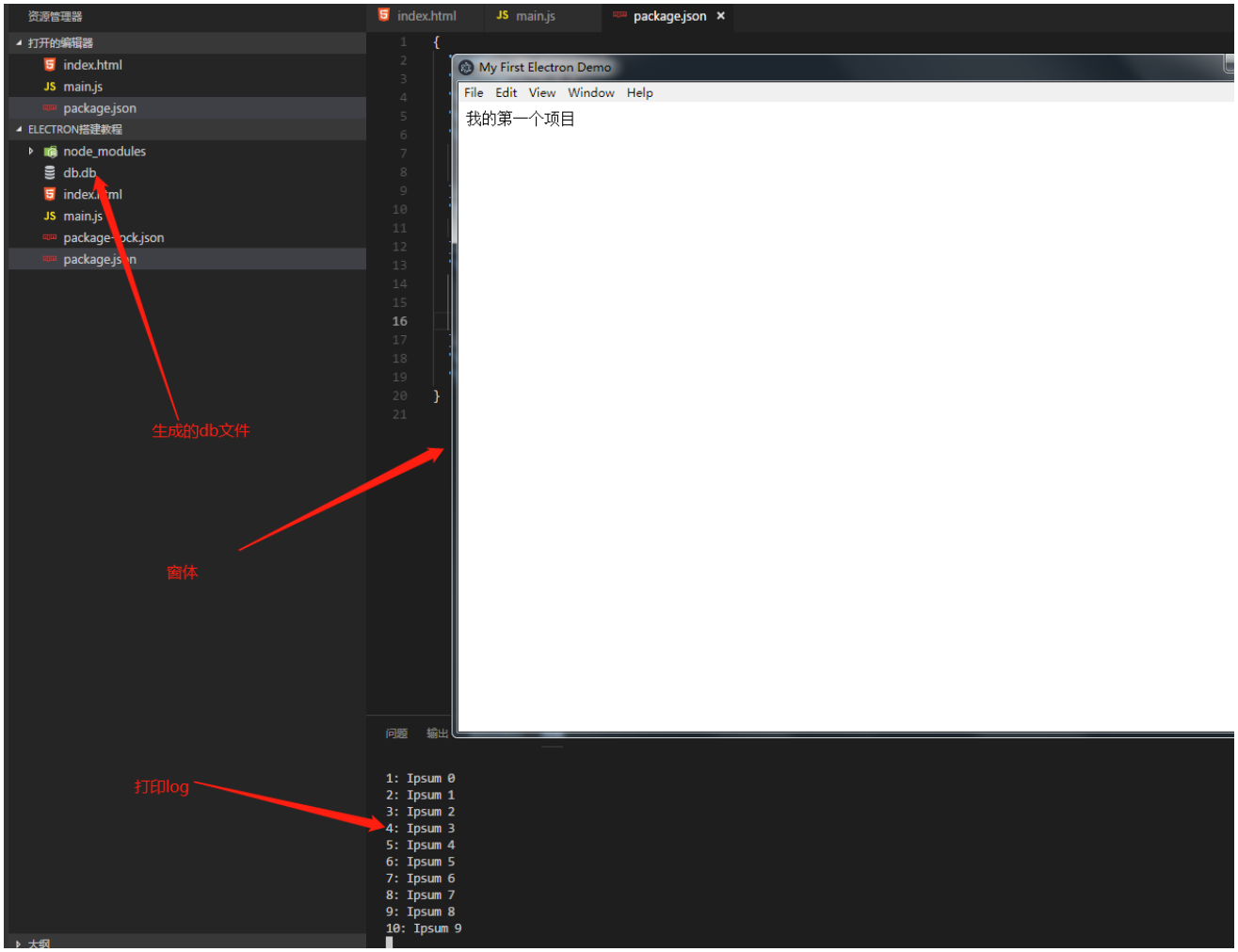
```
// This method will be called when Electron has finished
// initialization and is ready to create browser windows.
// Some APIs can only be used after this event occurs.
// app.on('ready', createWindow)
app.on('ready', function(){
    createWindow();
    var sqlite3 = require('sqlite3').verbose();
    const path = require('path');
    var db = new sqlite3.Database(path.join(__dirname, 'db.db'));
    db.serialize(function() {
        db.run("CREATE TABLE if not exists lorem (info TEXT)");

        var stmt = db.prepare("INSERT INTO lorem VALUES (?)");
        for (var i = 0; i < 10; i++) {
            stmt.run("Ipsum " + i);
        }
        stmt.finalize();

        db.each("SELECT rowid AS id, info FROM lorem", function(err, row) {
            console.log(row.id + ": " + row.info);
        });
    });
    db.close();
})
```

```
# 启动项目
npm run start
```

弹出窗体并且没有报错。项目文件中出现db.db文件后成功


资源管理器
打开的编辑器
index.html
main.js
package.json
ELECTRON搭建教程
node_modules
db.db
index.html
main.js
package-lock.json
package.json
index.html
main.js
package.json
My First Electron Demo
File Edit View Window Help
我的第一个项目
生成的db文件
窗体
打印log
问题
输出
1: Ipsum 0
2: Ipsum 1
3: Ipsum 2
4: Ipsum 3
5: Ipsum 4
6: Ipsum 5
7: Ipsum 6
8: Ipsum 7
9: Ipsum 8
10: Ipsum 9