Create Electron App with React+Express js

STEP1: Create folder of your App Like

mkdir my-electron-app && cd my-electron-app  
 npm init -y

STEP:2.

Your package.json file should look something like this:

{  
   "name": "my-electron-app",  
   "version": "1.0.0",  
   "description": "Hello World!",  
   "main": "main.js",  
   "author": "Jane Doe",  
   "license": "MIT"  
 }

Description must be require

STEP:3

Then, install the electron package into your app's devDependencies.

npm install --save-dev electron

STEP:3

CREATE 3 PAGES IN ROOT FOLDER  
 main.js : Entry point of your electron app

Index.html, preload.js index.html

Main.js file require two most important topic

1. Require server like :- require(./server/app)
2. Give correct path of index.html

Main.js

*// Modules to control application life and create native browser window*

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

const path = require('path');

const url = require('url');

const server = require('./server/app');

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(

    path.join(\_\_dirname, './client', 'build', 'index.html')

  )

*// Open the DevTools.*

*// mainWindow.webContents.openDevTools()*

}

*// 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.whenReady().then(() => {

  createWindow()

  app.on('activate', function () {

*// 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 (BrowserWindow.getAllWindows().length === 0) createWindow()

  })

})

*// Quit when all windows are closed, except on macOS. There, it's common*

*// for applications and their menu bar to stay active until the user quits*

*// explicitly with Cmd + Q.*

app.on('window-all-closed', function () {

  if (process.platform !== 'darwin') app.quit()

})

*// In this file you can include the rest of your app's specific main process*

*// code. You can also put them in separate files and require them here.*

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'; style-src 'self' 'unsafe-inline'">

    <link href="./styles.css" rel="stylesheet">

    <title>Hello World!</title>

  </head>

  <body>

    <h1>Hello World!</h1>

    We are using Node.js <span id="node-version"></span>,

    Chromium <span id="chrome-version"></span>,

    and Electron <span id="electron-version"></span>.

*<!-- You can also require other files to run in this process -->*

*<!-- <script src="./renderer.js"></script> -->*

  </body>

</html>

Preload.js is not required

STEP: 4

Now create two folder client and server

Server folder will inside my-electron-app folder and cliend folder will be now outside of my-elecron-app

After create build file of client folder we will paste build folder inside of my-electron-app

After that install npm module only client folder.

if server has node module then delete it

STEP: 5

Update package.json file like this type

{

  "name": "desktop",

  "version": "1.0.0",

  "description": "this is hello",

  "main": "main.js",

  "scripts": {

    "start": "electron .",

    "build": "electron-builder build",

    "test": "react-scripts test",

    "pack": "electron-packager . --overwrite",

    "postpack": "rm -rf myapp-win32-x64/resources/app/node\_modules"

  },

  "author": "",

  "license": "ISC",

  "devDependencies": {

    "electron": "^24.1.3",

    "electron-packager": "^17.1.1",

    "electron-builder": "^23.6.0"

  },

  "dependencies": {

    "axios": "^1.4.0",

    "body-parser": "^1.20.2",

    "cors": "^2.8.5",

    "dotenv": "^16.0.3",

    "ejs": "^3.1.9",

    "express": "^4.18.2",

    "mongoose": "^7.1.0",

    "morgan": "^1.10.0",

    "multer": "^1.4.5-lts.1",

    "mysql": "^2.18.1",

    "nodemon": "^2.0.22"

  }

}

Add all server dependency like we need express js , mysql, multer etc.

In package.json we must check "main": "main.js", same as main file in this mainfile is main.js

STEP: 5

After all changes we should build client folder

For that we fire below command on client folder

**E:\My project\tutorial of all js\Electron js\sam\client> npm run build**

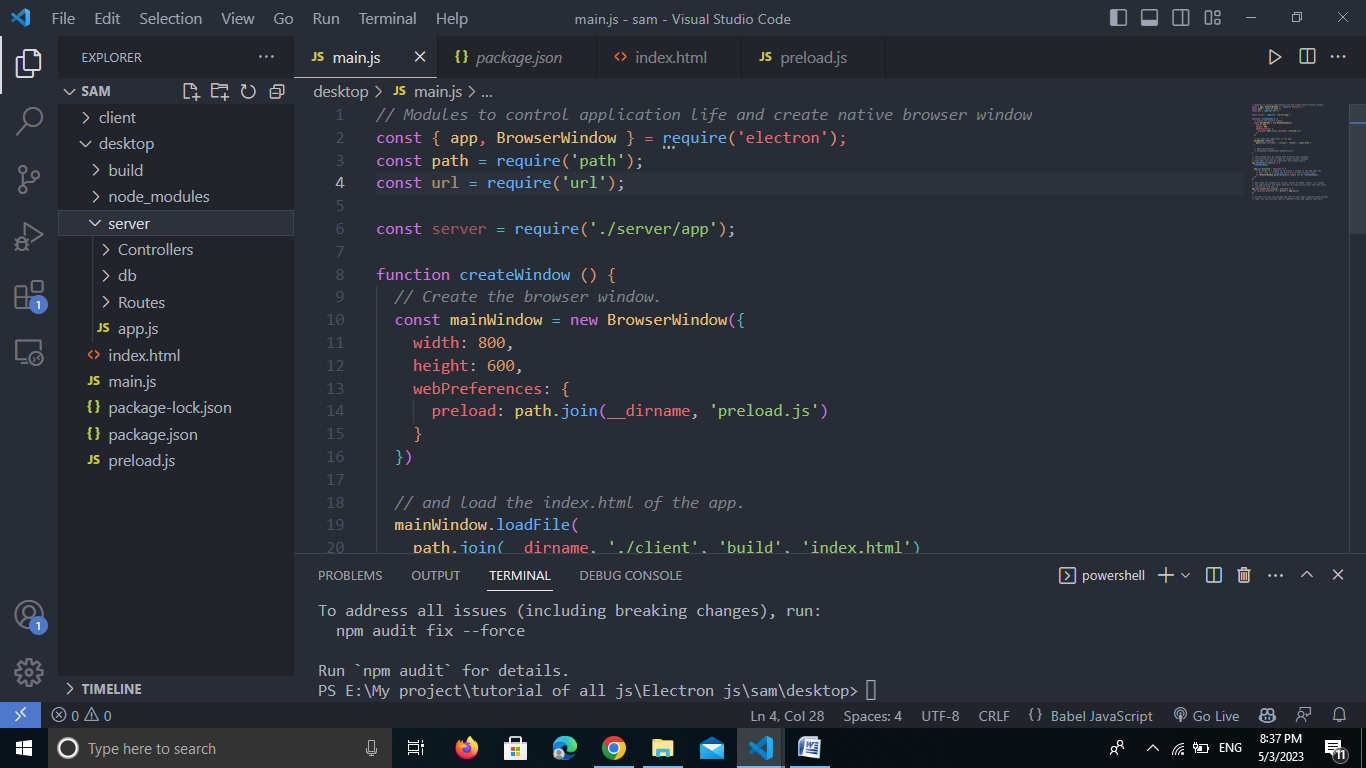
**A**fter build we should copy build folder from client folder to paste inside desktop folder

Now we call index.html inside of build folder

All client folder work is over. We can delete client folder

STEP: 6

We come is my-electron-app folder



Here desktop folder is our my-electron-app folder cosiderd

Server has not its own package.json

We added all dependency of server in main package.json which is available in my-electron-app

Now install all dependency of package.json

E:\My project\tutorial of all js\Electron js\sam\desktop> npm i

Check path in main.js it should be like this

 mainWindow.loadFile(

    path.join(\_\_dirname,  './build', 'index.html')

  )

STEP: 7

Now add three development dependency for create exe file

npm install electron-builder --save-dev

npm install electron-packager --save-dev

npm install -g electron-packager

STEP: 8

Now our code is done

We will start to create exe file

First way: this way is not recommended because when we install application then all source code show in file

Run command : npm run pack

Second way: this is a best way to generate exe file because it don’t show source code so it is more safe

Run command : electron-packager . --platform=win32 --arch=x64 --out=dist --overwrite