**Electron**

**(**Build cross platform desktop app using web technologies**)**

Web applications become more and more powerful every year, but there is still room for desktop apps with full access to the hardware of your computer. Today you can create desktop apps using the already familiar HTML, JS and Node.js, then package it into an executable file and distribute it accordingly across Windows, OS X and Linux.

Electron (formerly known as Atom Shell) is an [open-source](https://en.wikipedia.org/wiki/Open-source) [framework](https://en.wikipedia.org/wiki/Software_framework) developed by [GitHub](https://en.wikipedia.org/wiki/GitHub).

Electron.js helped us build the high level Windows Applications, even though we didn’t have too much experience building Windows Applications.

We didn’t have to worry about how the application worked with Windows, because Electron.js will do it for us. React.js helped us move fast on the building of the UI part

It was like we were just building a website. If you can build a website, you can build a desktop app.

**About:**

[*Electron*](https://electron.atom.io) is an open source library developed by GitHub for building cross-platform desktop applications with HTML, CSS, and JavaScript. Electron accomplishes this by combining [***Chromium***](https://www.chromium.org/Home)and [***Node.js***](https://nodejs.org) into a single runtime and apps can be packaged for Mac, Windows, and Linux.

Basically we are writing a client app.

## **Prerequisites**

You should have a basic knowledge of Javascript(ES6) and HTML. You also need to know some native Node.js APIs like file handling, processes, etc. If you are not acquainted with these, we'll suggest you to go through their tutorials first.

**Electron installation**

*# Clone the Quickstart repository*  
**$ git clone https://github.com/electron/electron-quick-start**  
  
*# Go into the repository*  
**$ cd electron-quick-start**

#Since an Electron app is just a fancy Node.js app, you will need to have *npm* installed.   
*# Install the dependencies*

**$ npm install**

#This will create a **node\_modules** folder containing all the Node.js dependencies required for the app to work. Everything should be good to go now, in the same terminal as before enter the following:

**$ npm start**

#The app should open up in it's own window. Notice it has a top menu bar and everything!

## **Structure of an Electron application**

A basic Electron app consists of three files:

1. package.json (metadata),
2. main.js (code)
3. index.html (graphical user interface)

The framework is provided by the Electron executable file (electron.exe in Windows, electron.app on macOS and electron on Linux). Developers wishing to add branding and custom icon can rename and/or edit the Electron executable file.

The most important file in the electron file is *package.json.* It keeps information about package. The most common information in *package.json* are:

1. "name", the application name
2. "version", the application version string
3. "main", the name of the main script file of the application

*package.json* is an [npm](https://en.wikipedia.org/wiki/Npm_%28software%29) file

**Benefits of using Electron**

1. Users can build Electron without a powerful machine
2. Electron applications run the same regardless of the operating system.
3. Users can view Electron applications on a variety of different browsers, including Chrome, Firefox, Internet Explorer, Edge and Safari.

**Electron has been used to create a number of apps:**

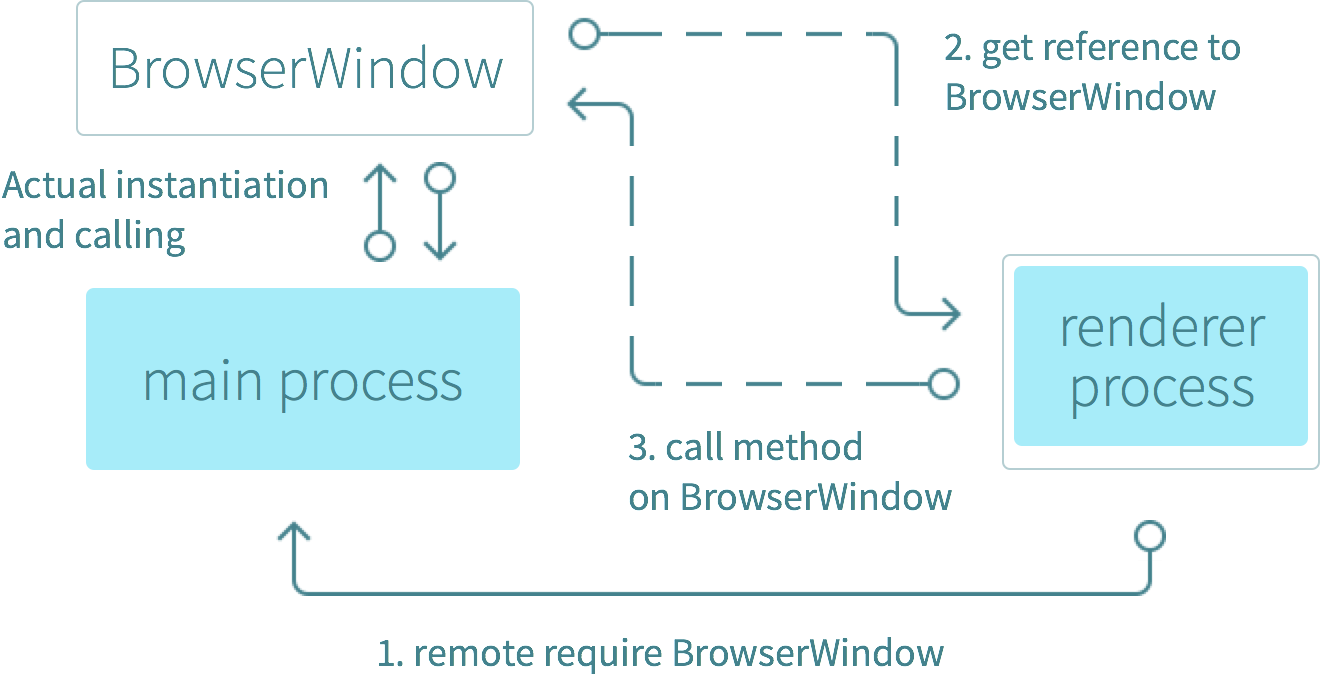
1. Slack desktop
2. Wordpress desktop app
3. Visual Studio Code

**How Electron work:**

There is a main process and Renderer Process

Main Process-Package.json main script. Create BrowserWindow instance to run web pages.

Renderer Process- Each webpages runs its own isolated process called the renderer process



# **How does it** [**Electronify-server**](https://www.npmjs.com/package/electronify-server) **work?**

Top of Form

Bottom of Form

When the Electron app loads, electronify-server will start a child process with the command you gave it in the configuration. This command is assumed to be a web server (but it doesn't have to be). If the child process was started successfully, the window will open and load the url to your server.

## **Examples**

In order to run the examples, you need to have electron installed. If you do not have it installed,

*var electronify = require('electronify-server');*

*electronify({*

*url: 'https://google.com',*

*noServer: true*

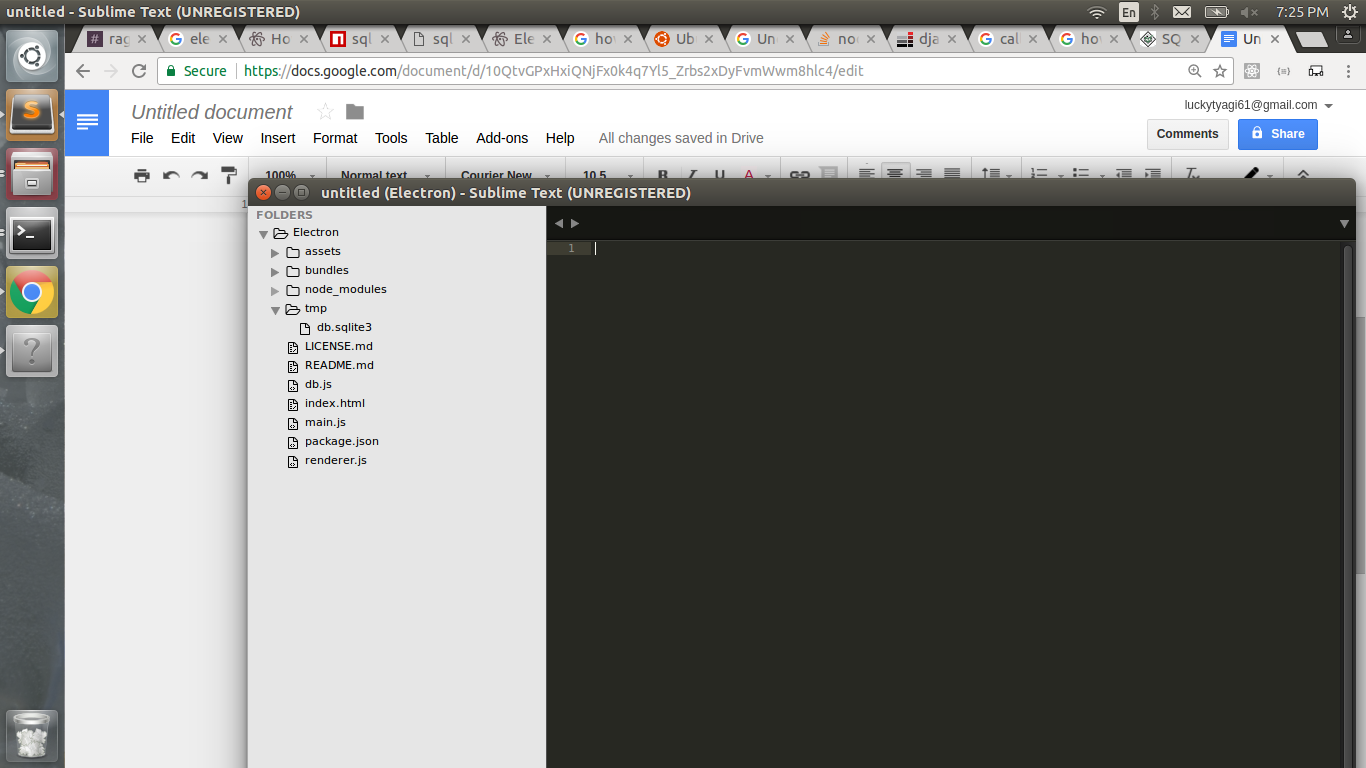
*});*

**Electron + Database**

**npm install sqlite3**

**npm install --save sql.js**

**Folder structure should be like**



To create a new SQLite database named "db" with a single  
 table named "test123", you might do this:  
  
SQLite version 3.11.0 2016-02-15 17:29:24

Enter ".help" for usage hints.

sqlite> create table test123(id int,fname varchar(20), lname varchar(20));

sqlite> insert into test123 values(1,'lalit','tyagi');

sqlite> insert into test123 values(2,'vineet','tyagi');

sqlite> insert into test123 values(3,'nihal','jumhare');

sqlite> insert into test123 values(4,'mandip','gothadiya');

sqlite> insert into test123 values(5,'jaggu','mohan');

sqlite> select \*from test123;

1|lalit|tyagi

2|vineet|tyagi

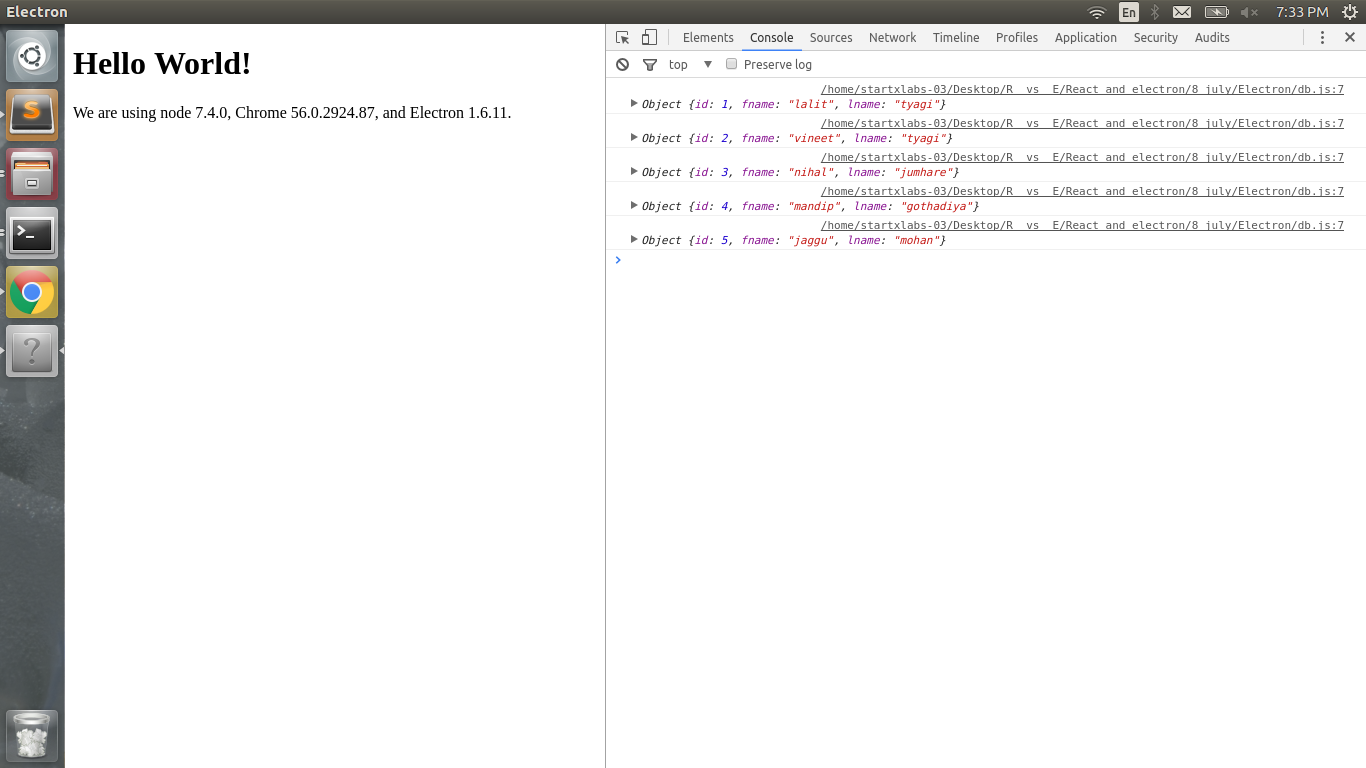
3|nihal|jumhare

4|mandip|gothadiya

5|jaggu|mohan

sqlite>

**Output**



For window we can follow

<https://www.youtube.com/watch?v=oCKDF9BU6cI>

**React + Electron**

A simple boilerplate app to demonstrate how to use ES6 and React with Electron. It uses Babel to automatically transpile ES6 and JSX code, without depending on any package manager besides npm.

**HOW ?**

The Node and Electron binaries both take a parameter -r that automatically requires a module before the rest of the code. The npm start script is modified using this, which registers Babel and loads the entry point main.js.

The renderer entry point index.html does basically the same, but loads the scripts/main.js file, which renders the views/main.jsx component into the body.

**Installation**

**git clone https://github.com/LalitTyagi/React-Electron.git**

**cd React-Electron   
npm install**

**npm install jquery**

In package.json

Add

"jquery": "^2.1.4"

And then save it. (for $ variable declaration )

In main.jsx

Address : ../view/main.jsx

Declare $ variable as:

var $ = require('jquery');

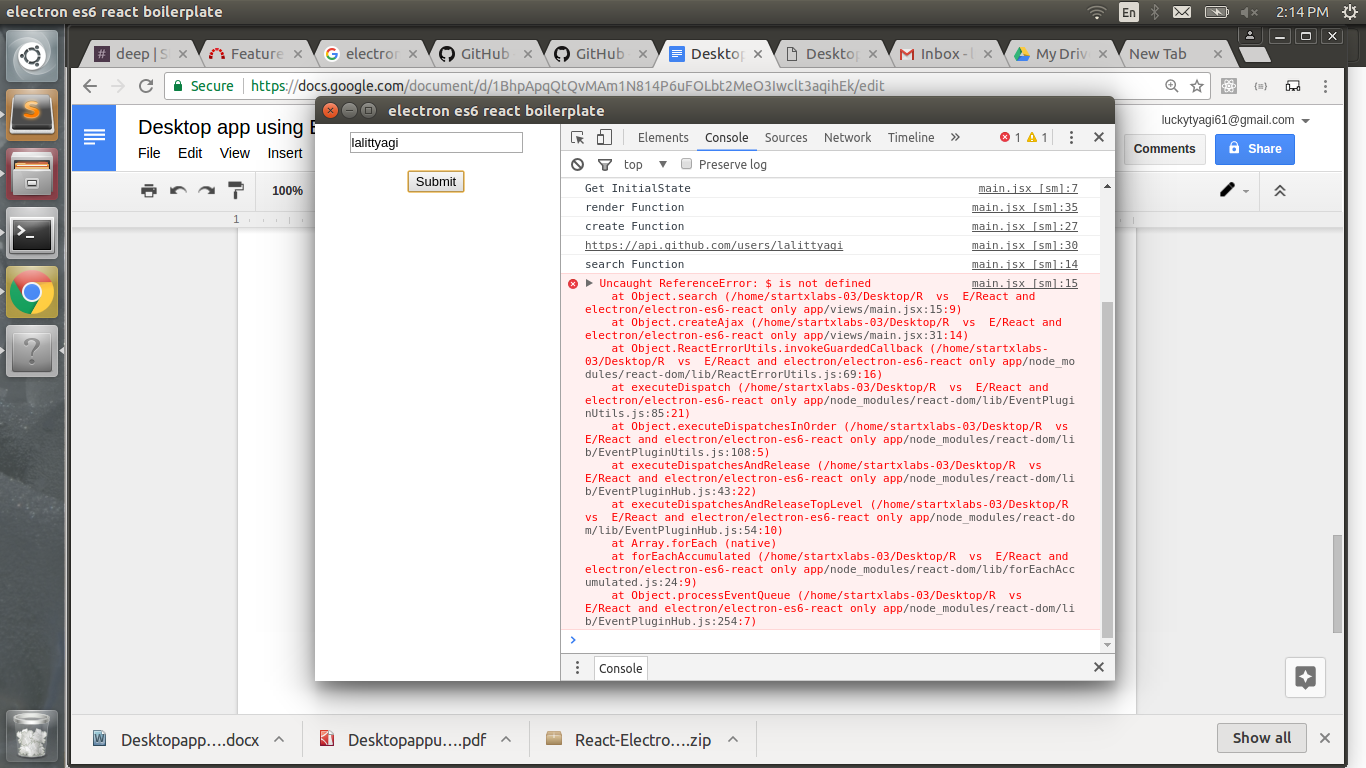
**npm start**

We can reload our app using **ctrl+R**

**Problem faced:**

The only problem we faced is due to $

As we are using it for Ajax function declaration in react component due to which it is showing error that $ in not declared.



For that first we have to install Jquery

Using cmd npm install jquery and then we have to make some modifications in our files as follows:

In package.json

Add

"jquery": "^2.1.4"

And then save it. (for $ variable declaration )

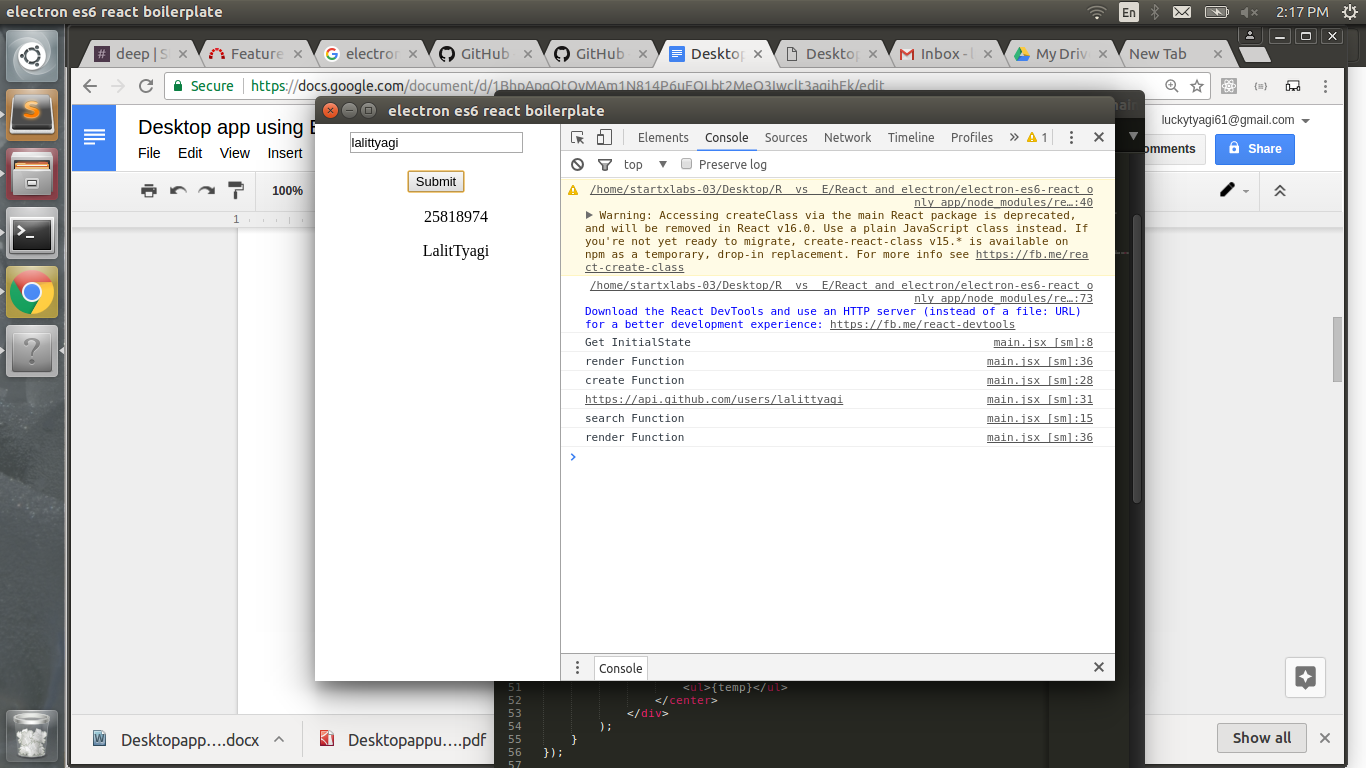
In main.jsx

Address : ../view/main.jsx

Declare $ variable as:

var $ = require('jquery');

**Final output**



**App Packing**

Add the highlighted scripts given below to your package.json file , making your package.json look like this:

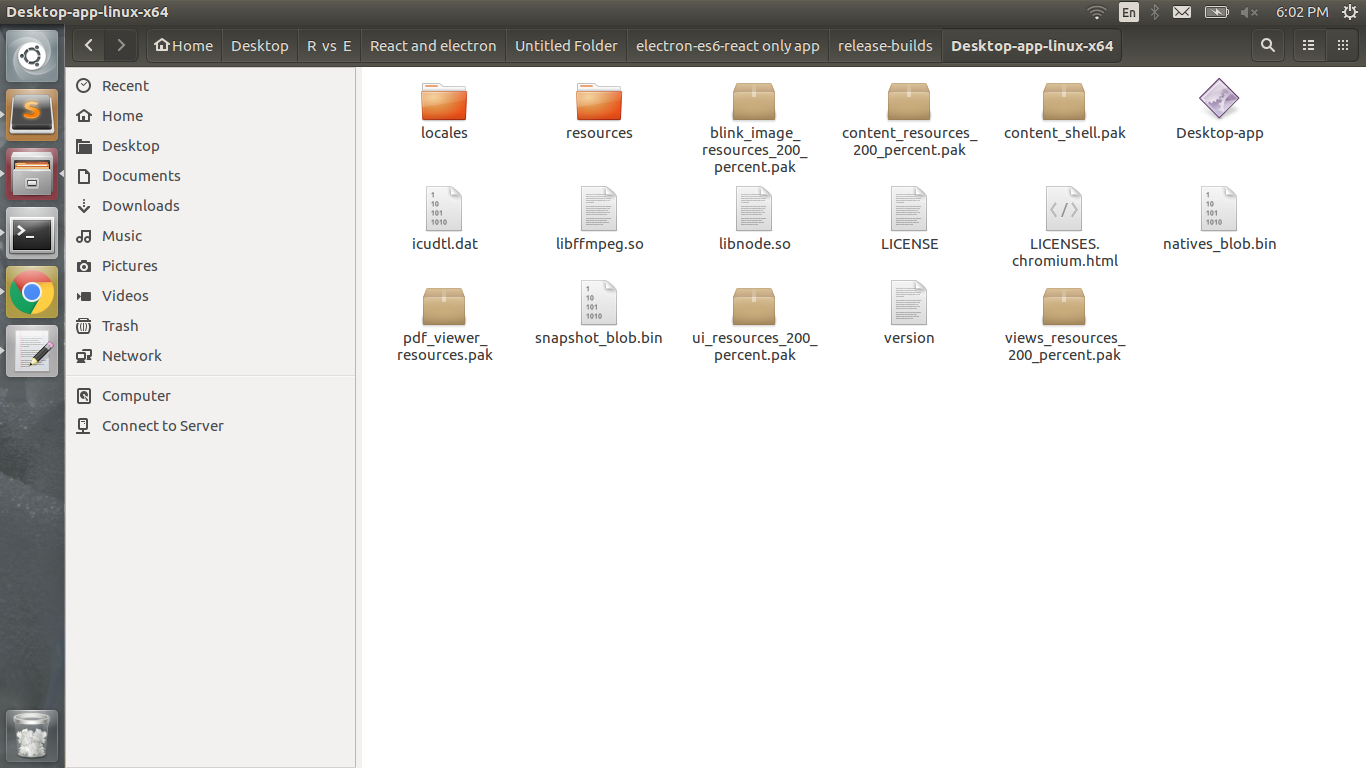
{  
 "name": "electron-tutorial-app",  
 "productName": "Desktop-app",  
 "version": "0.1.0",  
 "main": "main.js",  
 "devDependencies": {  
 "electron": "^1.4.3",  
 "electron-packager": "^8.1.0"  
 },  
 "scripts": {  
"package-mac": "electron-packager . --overwrite --platform=darwin --arch=x64 --icon=assets/icons/mac/icon.icns --prune=true --out=release-builds",  
"package-win": "electron-packager . --overwrite --asar=true --platform=win32 --arch=ia32 --icon=assets/icons/win/icon.ico --prune=true --out=release-builds --version-string.CompanyName=CE --version-string.FileDescription=CE --version-string.ProductName=\"Desktop-app\"",  
"package-linux" : "electron-packager . --overwrite --platform=linux --arch=x64 --icon=assets/icons/png/1024x1024.png --prune=true --out=release-builds"  
 }  
}

Now you can run:

npm run package-mac // for mac packager

npm run package-win // for windows packager

npm run package-linux // for linux packager



Double click on Desktop-app and you have your desktop app

**Advantages**

1. Cross platform - Your apps can be easily ported to virtually any platform with a web browser
2. Fast development cycles - JavaScript, HTML, and CSS make it very easy to build powerful apps very quickly.
3. Can be distributed offline as well.
4. No installation required
5. For a programmer who is coming from a Web development background and who wishes to make a standalone program which functions much like a Web app, then Electron may offer an ideal solution for such a development.

**Disadvantages**

1. You are stuck with JavaScript - JavaScript is the only language you can use for web development on the client. Because JavaScript is dynamic this also limits your ability to optimize your app.
2. User data is kept in the web app servers for which Internet is required and also Hosting cost

**Is it possible to build an Electron app with just Bootstrap as the front-end library?**

**Ans:** Bootstrap is only a CSS framework it doesn't give you a way to manipulate the DOM for example.

You could in principle do everything with vanilla JS, or jQuery, but I'd recommend you to take some time and learn a modern library to avoid micro managing the DOM.

*Vue* is super easy and is excellent for beginners. You just need to import *Vue* with a <script> tag, write your templates directly in your index.html and finally write your JS somewhere (either another <script> or some .js file). You can of course get it running with *Webpack*, ES6, the whole enchilada, but it's not needed.

<https://discuss.atom.io/t/is-it-possible-to-build-an-electron-app-with-just-bootstrap-as-the-front-end-library/41842/6>

**Tutorial**

* 1. Series on YouTube regarding electron framework

<https://www.youtube.com/playlist?list=PLYxzS__5yYQmocPoLUiEAfD1cJNjhdQar>

* 1. <https://www.tutorialspoint.com/electron/index.htm>

**Reference:**

1. <https://electron.atom.io/docs>
2. <https://en.wikipedia.org/wiki/Electron_(software_framework)>
3. <https://www.youtube.com/watch?v=8YP_nOCO-4Q&feature=youtu.be>
4. <https://tutorialzine.com/2015/12/creating-your-first-desktop-app-with-html-js-and-electron>
5. <https://www.npmjs.com/package/electronify-server>
6. <http://ourcodeworld.com/articles/read/259/how-to-connect-to-a-mysql-database-in-electron-framework>