Basics of Building Node.js Applications with Electron

Initiating application, using VS Code, debugging, packaging, customization, logging etc.

# Initiating, Packaging etc.

## Initiating App

### Initiating & Installing Global Packages

Open CMD from the Application Root Folder.

C:\Users\arind>cd C:\TECH\Electron Apps\Apps\HelloWorld

Initiate npm.

C:\TECH\Electron Apps\Apps\HelloWorld>npm init

Install npm.

C:\TECH\Electron Apps\Apps\HelloWorld>npm install –-save --verbose

Install following packages globally.

C:\Users\arind>npm install -g electron

C:\Users\arind>npm install -g electron-packager

C:\Users\arind>npm install -g eslint

C:\Users\arind>npm install -g jshint

C:\Users\arind>npm install -g config

### Installing Local Packages & Adding Dependencies

Install required electron & configuration modules locally.

C:\Users\arind>npm install electron -–save –verbose

C:\Users\arind>npm install electron-packager -–save --verbose

C:\Users\arind>npm install config –-save --verbose

Install the following two for code quality enhancement and error detection in IDE (e.g. VS Code, Atom etc.).

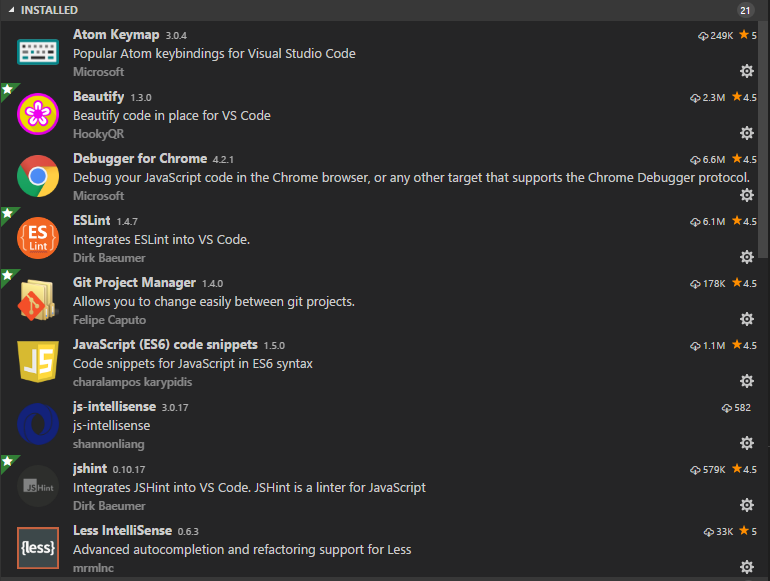
C:\TECH\Electron Apps\Apps\HelloWorld>npm install eslint –-save-dev --verbose

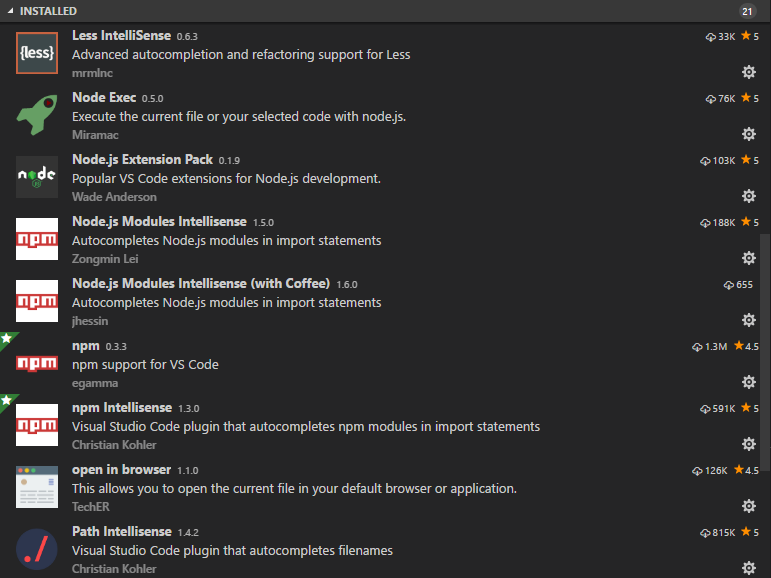
C:\TECH\Electron Apps\Apps\HelloWorld>npm install jshint –-save-dev --verbose

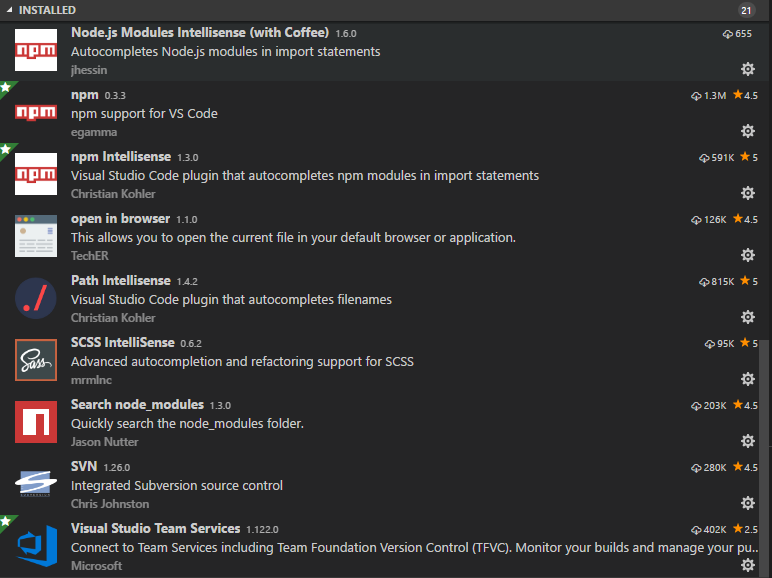
## Initiating VS Code

### Installing Extensions in VS Code

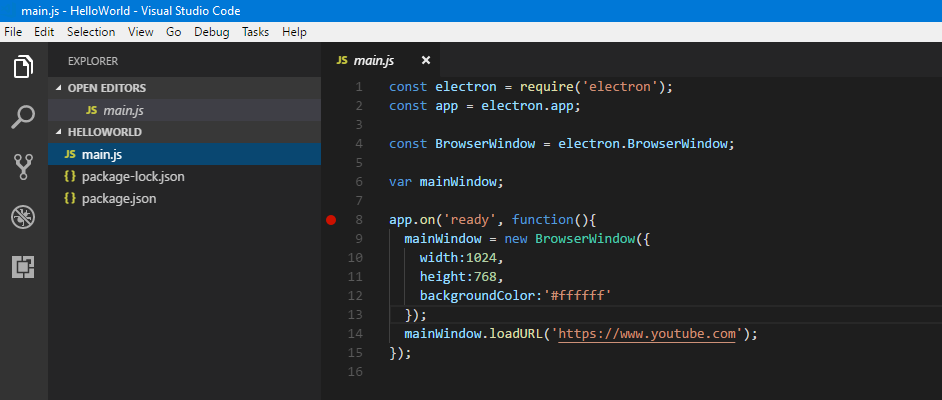
Following Extensions should be installed in VS Code to get things going.





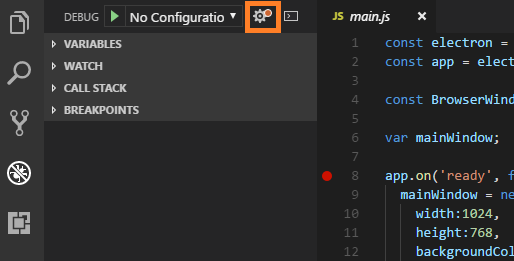


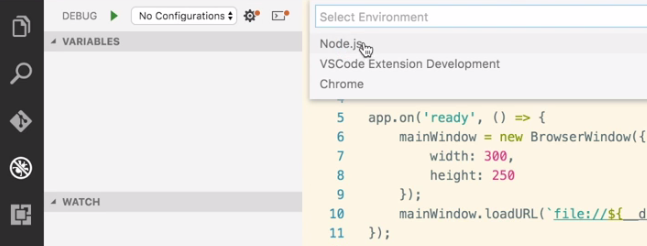
### Initiating VS Code

Open project folder in VS Code.  


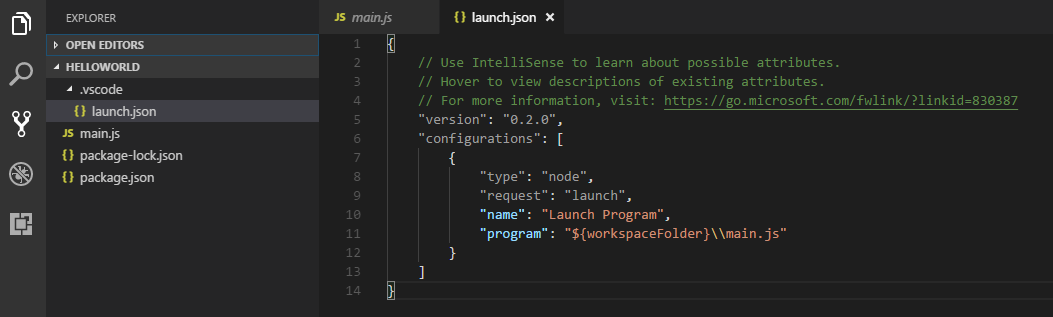
From the taskbar at the left side, click on debug. 

Click on the following tool icon to add setup files for node.js, which would add a folder named .vscode and a file launch.json inside .vscode folder.





After this, it should look like the following.



## Running & Debugging

### Running the App

Open package.json. Modify the following part –

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

To the following –

"scripts": {

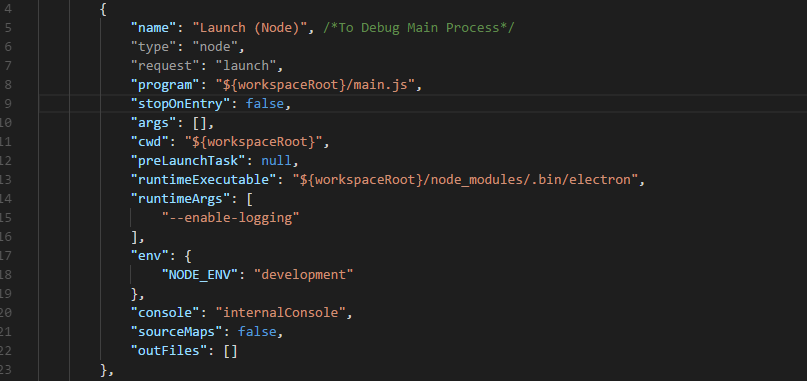
"start": "electron ."

},

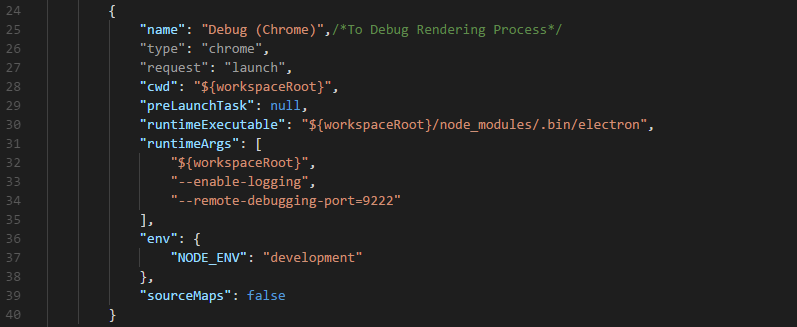
Open CMD.

C:\TECH\Electron Apps\Apps\HelloWorld>npm start

### Debugging

Use following combination to debug the main process.  


Use following combination to debug the child process.



The sample launch.json file is being scripted below.

{

"version": "0.2.0",

"configurations": [

{

"name": "Launch (Node)", /\*To Debug Main Process\*/

"type": "node",

"request": "launch",

"program": "${workspaceRoot}/main.js",

"stopOnEntry": false,

"args": [],

"cwd": "${workspaceRoot}",

"preLaunchTask": null,

"runtimeExecutable": "${workspaceRoot}/node\_modules/.bin/electron",

"runtimeArgs": [

"--enable-logging"

],

"env": {

"NODE\_ENV": "development"

},

"console": "internalConsole",

"sourceMaps": false,

"outFiles": []

},

{

"name": "Debug (Chrome)",/\*To Debug Rendering Process\*/

"type": "chrome",

"request": "launch",

"cwd": "${workspaceRoot}",

"preLaunchTask": null,

"runtimeExecutable": "${workspaceRoot}/node\_modules/.bin/electron",

"runtimeArgs": [

"${workspaceRoot}",

"--enable-logging",

"--remote-debugging-port=9222"

],

"env": {

"NODE\_ENV": "development"

},

"sourceMaps": false

}

]

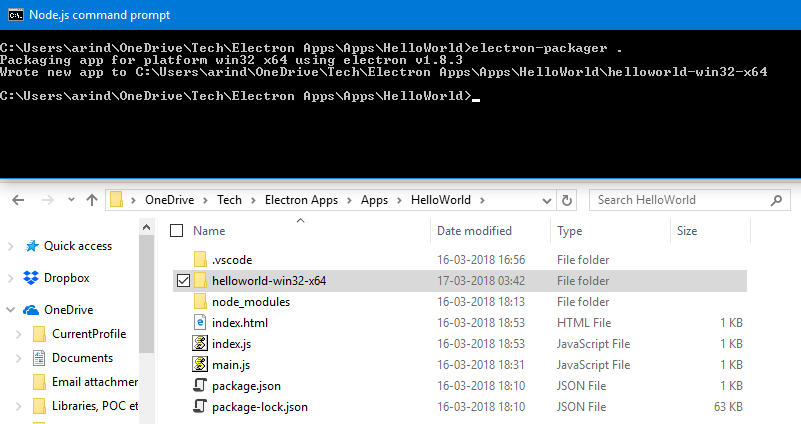
}

## Packaging Application

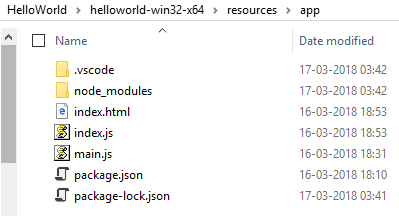
### Packaging with Source Unprotected

Following command will package the application.

C:\Users\arind\OneDrive\Tech\Electron Apps\Apps\HelloWorld>electron-packager .



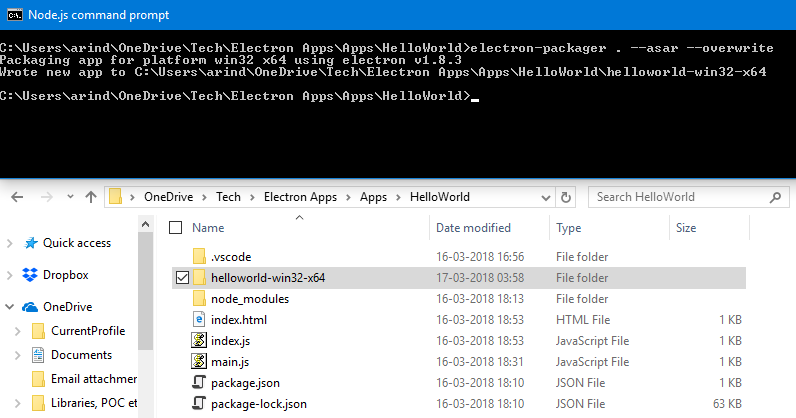
But all source codes will remain unprotected.



### Packaging with Source Protected

Following command will package the application.

C:\Users\arind\OneDrive\Tech\Electron Apps\Apps\HelloWorld>electron-packager . -–asar --overwrite



All source codes will remain protected. However, asar files could be opened in notepad.

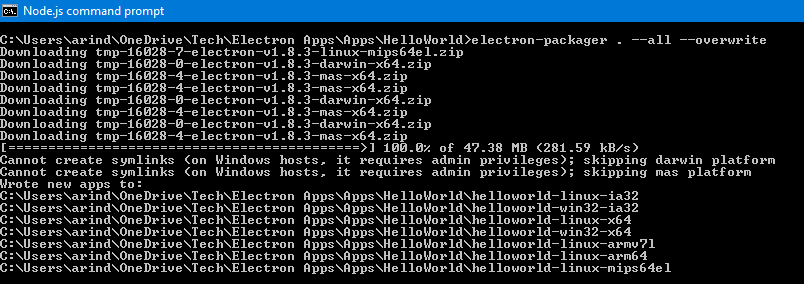


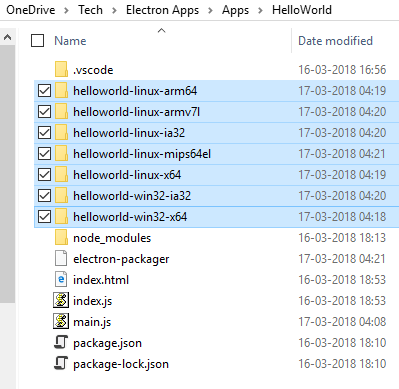


### Packaging for All Platforms

Following command will package the application for all the environment (i.e. windows, Linux etc.).

C:\Users\arind\OneDrive\Tech\Electron Apps\Apps\HelloWorld>electron-packager . -–asar -–all –overwrite





Selected folders are packages for different distributions.

## Loading JS & CSS Libraries (jQuery, Bootstrap etc.)

Electron applications throw error if we try to load js & css libraries conventionally in a HTML page.

We should keep all our libraries / static contents (e.g. application window icon) in a folder called “assets” and load the content like the following code.

<!DOCTYPE html>

<html lang="en">

<head>

<title>My First Electron App : Index</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<script>window.$ = window.jQuery = require("../../assets/js/jquery.min.js")</script>

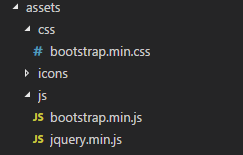
<script src="../../assets/js/bootstrap.min.js"></script>

<link rel="stylesheet" href="../../assets/css/bootstrap.min.css">

<script src="../js/test.js"></script>

</head>

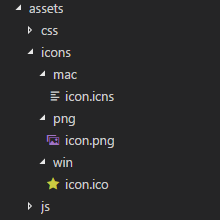
<body>

Assets folder should be structured like the following screenshot -   


## Customize Application Window Icon & Header

### Customize Application Window Icon

First create the following folder structure which will host .png, .ico & .icns files.



Now copy your three icon files as shown above. To convert .png to .ico & .icns use <https://cloudconvert.com>.

Now go to main.js and put the following attribute –

icon: path.join(\_\_dirname, 'assets/icons/png/icon.png'),

Where you are initiating a browser window from the main process, see screenshot below.

app.on('ready', function(){

mainWindow = new BrowserWindow({

icon: path.join(\_\_dirname, 'assets/icons/png/icon.png'),

title: "My First Electron App",

width: DefaultHeight,

height: DefaultWidth,

backgroundColor: BackgroundColor

});

### Customize Application Window Header

To modify the header from the browser window which is generated from the main process (main.js) open main.js and add the following line where you are initiating a browser window from the main process.

title: "My First Electron App",

After adding it should look like the screenshot below.  
app.on('ready', function(){

mainWindow = new BrowserWindow({

icon: path.join(\_\_dirname, 'assets/icons/png/icon.png'),

title: "My First Electron App",

width: DefaultHeight,

height: DefaultWidth,

backgroundColor: BackgroundColor

});

For individual HTML page the title comes from the HTML <title/> tag as shown below.

<!DOCTYPE html>

<html lang="en">

<head>

<title>My First Electron App : Index</title>

<meta charset="utf-8">

To omit / customize the title of an alert() or a confirm () window, it has to be handled from the JavaScript end.

$(document).ready(function () {

alert('ready', 'Alert Window');

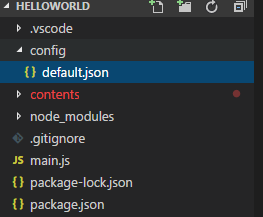
$("#p11").html(p11HTML);

});

## Reading from Configuration File

### Using “config” module

Create a folder name “config” under root and a “default.json” under it.



The default.json would hold all the configurable values which are to be passed to any JavaScript / Html file.

{

"App" : {

"Styling": {

"Main": {

"BackgroundColor": "#ffffff",

"DefaultHeight": 1024,

"DefaultWidth": 768

}

},

"TestURLLoadAddress" : "https://www.youtube.com",

"StartingPoint" : "contents/html/test.html"

},

"Keys" : {

"testKey" : "Sample String Value From config/default.json"

},

"Messages": {

"MSG001": "Hello ! ! "

}

}

Then access them like the following code snippet.

const config = require('config');

var BackgroundColor = config.get("App.Styling.Main.BackgroundColor");

var DefaultHeight = config.get("App.Styling.Main.DefaultHeight");

var DefaultWidth = config.get("App.Styling.Main.DefaultWidth");

var TestURLLoadAddress = config.get("App.TestURLLoadAddress");

var StartingPoint = config.get("App.StartingPoint");

app.on('ready', function(){

mainWindow = new BrowserWindow({

width: DefaultHeight,

height: DefaultWidth,

backgroundColor: BackgroundColor

});

//opens a browser window

//mainWindow.loadURL(TestURLLoadAddress);

//renders html page

mainWindow.loadURL(url.format({

pathname: path.join(\_\_dirname, StartingPoint),

protocol: 'file:',

slashes: true

}));

We can also keep another “production.json” where only the overriding values would exist for production.

{

"App" : {

"Styling": {

"Main": {

"BackgroundColor": "#D7BDBD"

}

},

"TestURLLoadAddress" : "https://www.facebook.com"

},

"Messages": {

"MSG001": "Hello ! ! (production)"

}

}

**INCOMPLETE**

### Using “dotenv” module

**INCOMPLETE**