**NPM + NPM SCRIPTS**

GENERAL

**Node.js – made using Google V8 engine to read and run javascript outside of browser**

* Node.js automatically installs npm with it
* For initial install of Node.js – go to node website and download latest version from there that is the long-term-release

**Once install run, check if node and npm installed properly**

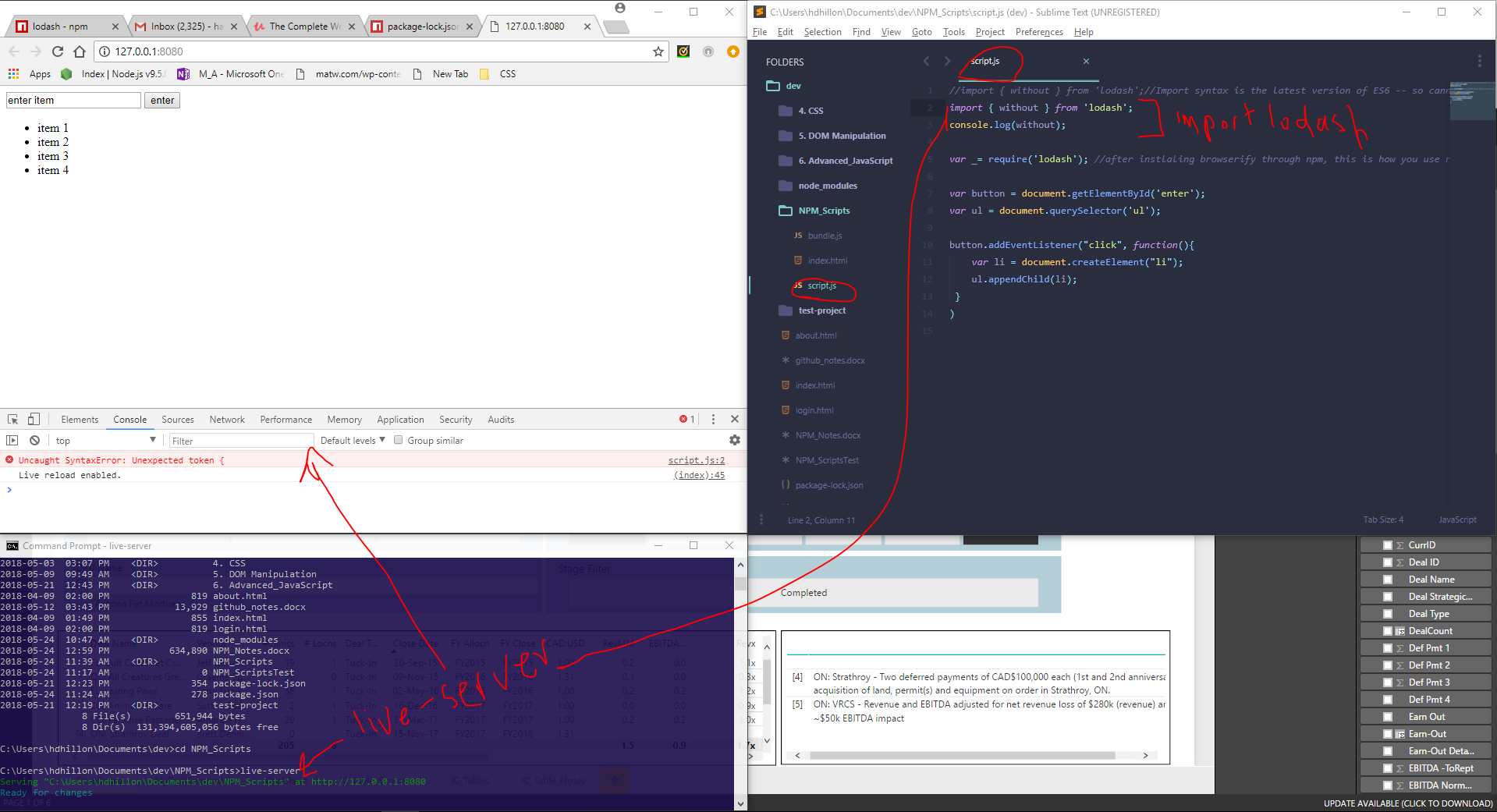
* npm – v
* node -v

**HOW TO INSTALL PACKAGES THE FIRST TIME (ONE-BY-ONE) and OVERVIEW OF PACKAGE.JSON AND PACKAGE-LOCK.JSON FILES**

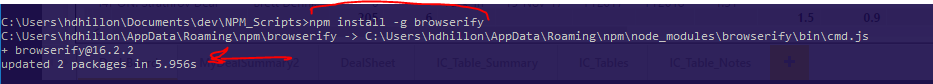
* **npm init** 🡪 creates a package.json file in the directory you are in
  + What is “.json” – file type
  + Looks like a javascript object except for fact that it has double quotes for properties
  + These are packages our website need to work
* Can install things in two ways: (1) locally (2) globally
  + **(1) Locally – package only able to run in the folder you want**
    - **E.g., lo-dash 🡪 extension of javascript (additional functions with arrays, etc. – check documentation)**
      * npm install --save lodash
      * If you go to package.json file – will see it in here if package.json file was in main folder
    - Installing locally – does the following:
      * Updates package.json file for the dependency (e.g., lo-dash and version)
      * Creates node\_modules folder and adds all lodash modules
  + **(2) Globally –** (this is what ‘-g’ mean s- you can use that package outside of the folder anywhere on computer
    - **E.g. Live server install (command line):**
      * npm install -g live-server
    - **Running live server:**
      * In command prompt:
        + Navigate to folder that index.html file is in and type “live-server”
    - Note – this does not create a node\_modules folder or package-lock.json file

BROWSERIFY PACKAGE – INSTALL AND CREATE BUNDLE.JS FILE WHEN USING ES6 WITHOUT WEB-PACK

* If we try to use lo-dash and its functions, we need to import it in our script file in some way.
  + If we use the “import” function in script.js file and we run live-server – we will get an error when the index. html file (that links to the script file) is run.

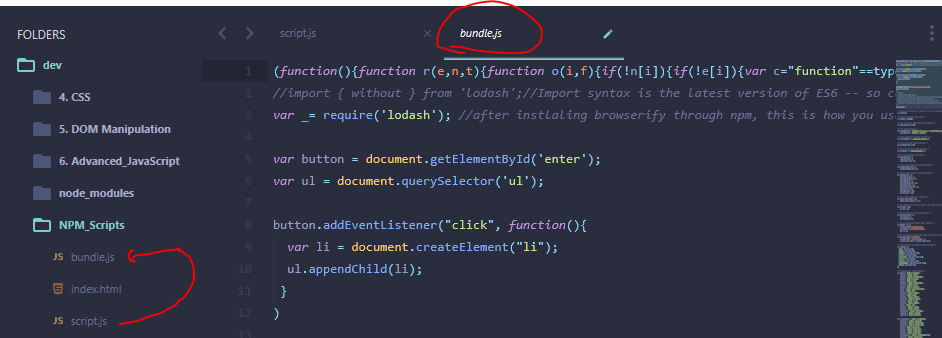


* + This is because **the “import” syntax is from ES6 which browsers have not yet come around to implement.**
    - For now, we have to use the **‘require’ syntax in order to do this. However, in order to do require syntax, we need ‘browserify’ (separate NPM package).**
    - **Browserify** creates a single ‘bundle.js’ file that brings in all code into one file and that is then deployed.
    - Note – once we install React, we’ll be able to use “web-pack” and can use the ‘import’ syntax.
  + **Installing browserify as a global package:** npm install -g browserify



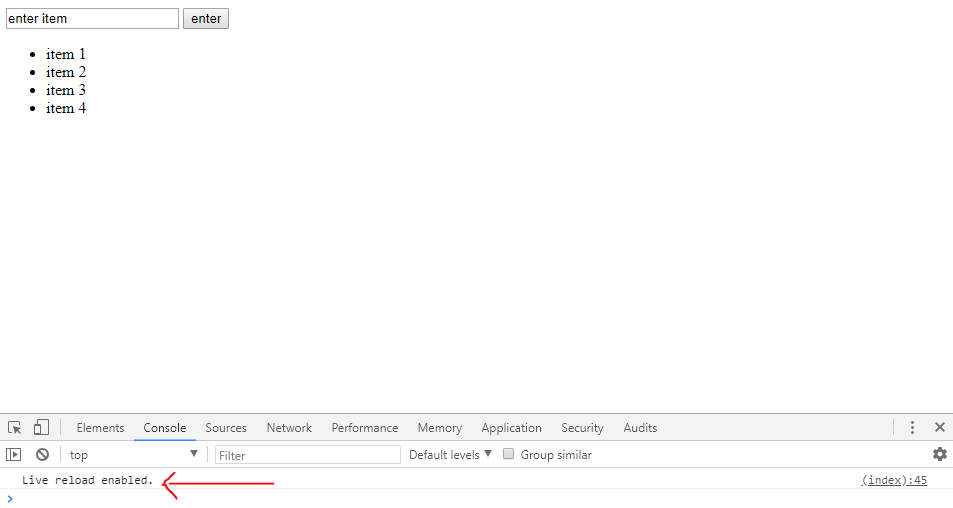
* + **Using browserify to bunde the script.js into a file called bundle.js**
    - In command line: “browserify script.js > bundle.js”





* + **Update index.html file script src to “bundle.js” instead of “script.js”, run live-server and no errors…now we can use the lodash functions**



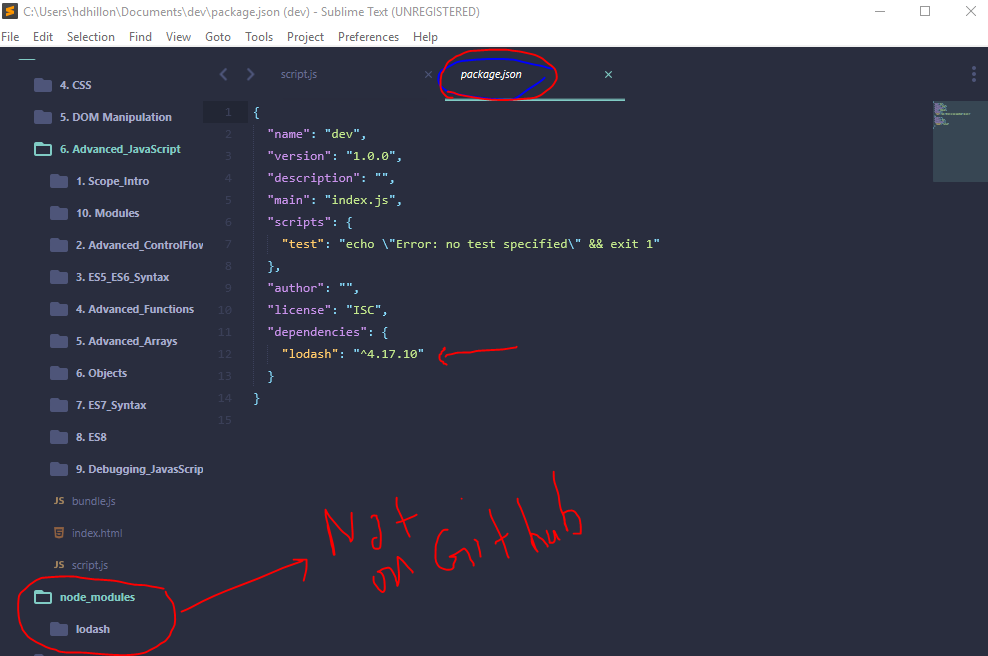


**HOW TO INSTALL PACAKGES WHEN A PRE-EXISTING PACKAGE.JSON FILE IS DOWNLOADED (e.g., from GitHub)**

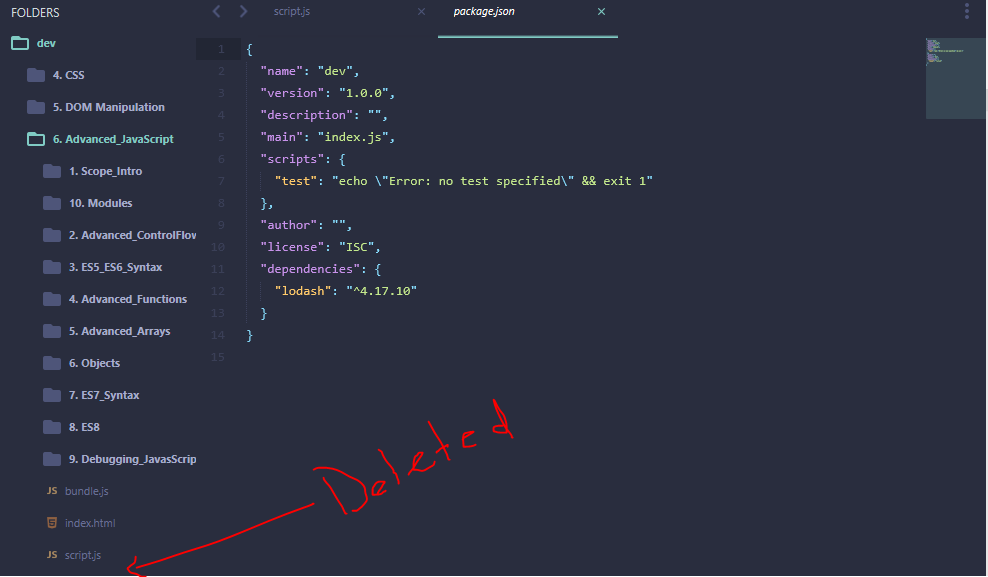
* Command line: ‘npm install’
  + This will refer to package.json file and install dependencies included in that file (e.g., lo-dash)

**BEST PRACTICES FOR PACKAGES AND GITHUB**

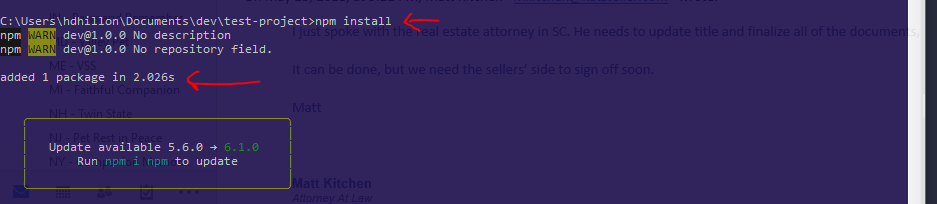
* When installing packages with npm, you end up with a folder called “node modules” and within it, you have the various packages that are installed – e.g., lo-dash.
* The node-modules folder should not be uploaded to git-hbu because the package.json file indicates what version of lodash or dependencies are needed
  + So, when someone copies your project from git-hub, as long as they have the package.json file, they will be able to run “npm install” and that will download all fo the dependencies and will create the node module.



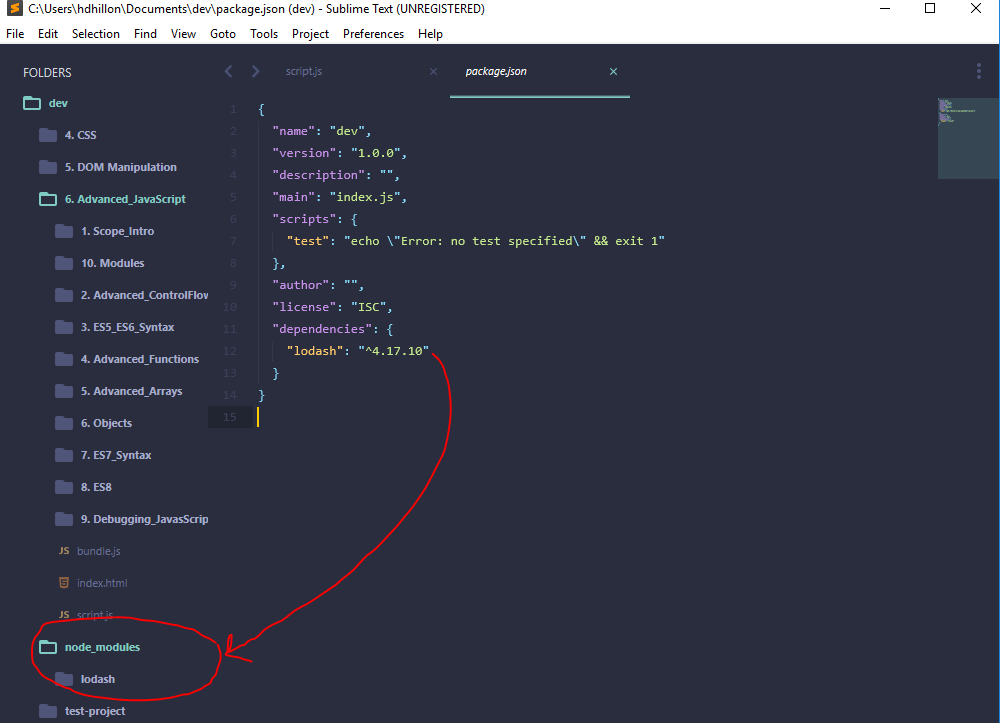
**If you delete the “node\_modules” folder, you simply enter “npm install” and will get back all required dependencies** *(doing this to replicate a git hub fork/copy of a project – bc this is how you will receive the file – i.e., no node module dependencies will be uploaded)*



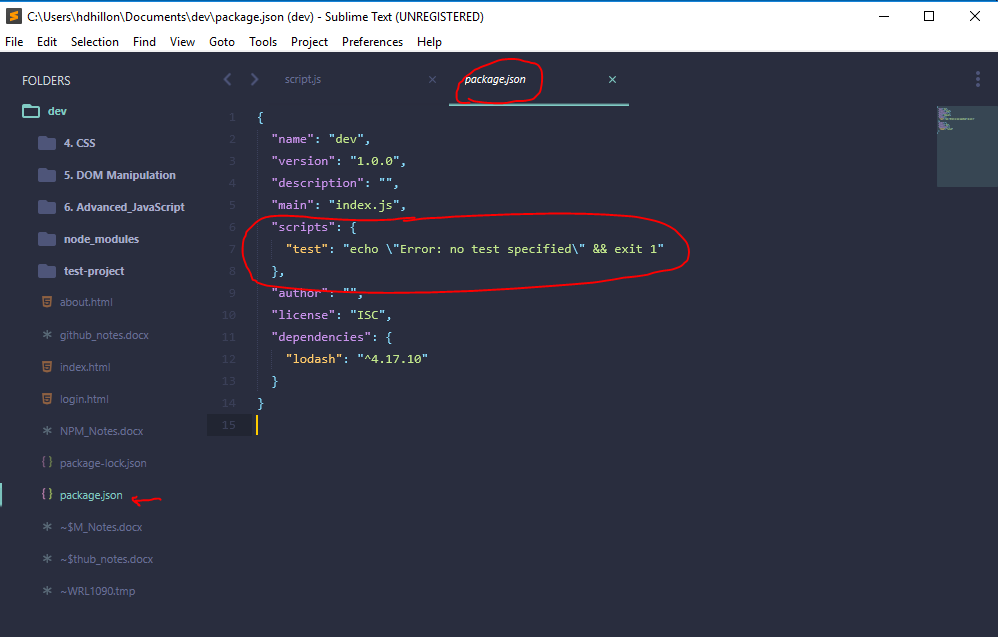
**Run npm install**

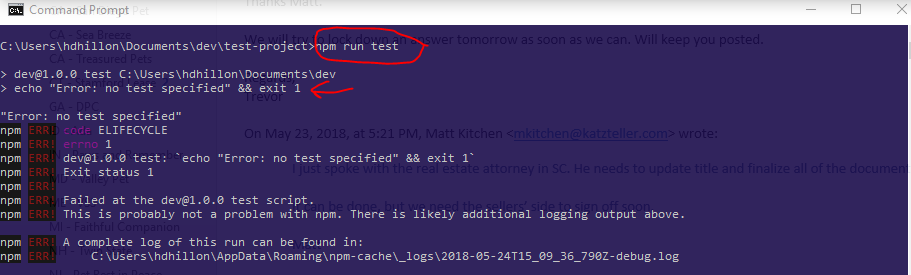


**node\_module folder added and dependencies installed**



**NPM SCRIPTS**

* Npm scripts (included in package.json) file allow you to run scripts directly from command line
* Npm scripts (included in package.json) file allow you to run commands (scripts) from the package.json file
  + E.g., script above is titled “test”
  + To run the script:
    - npm run test
  + Output shows result from script that is being run

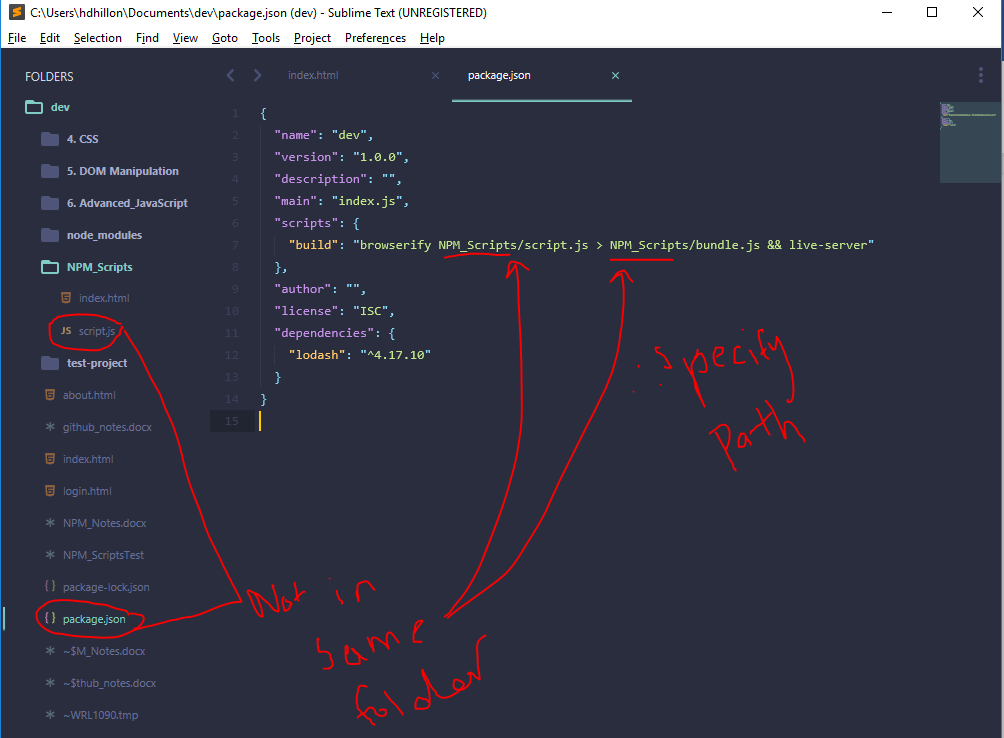


* + If we wanted to do browserify bundling directly through package.json we can write a script to do the following:
    - “scripts” {

“build”: “browserify script.js > bundle.js”

}

* + Then, when in command line, just run “npm run build”
  + Note – when package.json file and the script file are not in the same folder (below): then have to specify where to find script.js file and where to save bundle.js file



* + Note, in package.json, you can also add other scripts to run – e.g., live-server – this will execute both scripts when “npm run build” is run in the command line
    - “scripts” {

“build”: “browserify script.js > bundle.js && live-server”

}

* + After running in command line bundle.js is added using browserify and live-server is run...



