**COMP 3123 – Full Stack Development – Lab 4**

* NPM – Node Package Manager
* Testing with Mocha framework
* Web server routing with Node

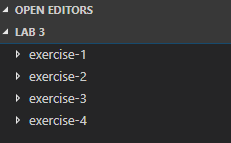
**Node Reference**

An optional reference for Node JS

**https://www.w3schools.com/nodejs**

**Developer Note:**When working on your exercises, please create separate folder for your work. This way you won’t putting all your code in the same file, which can pollute the global name space. In short, it will prevent you from overwriting your own work and causing your code to compile incorrectly.

Organize your folder structure in this way.



**GBLearn Account Setup**

Please setup your GLearn account and use FTP Filezilla to submit this lab and for the following Node Lab Test. There should be an email in your GeorgeBrown email account to access <https://my.gblearn.com/>

Note: Please use Microsoft Edge for setup, there is an error in Chrome browser when trying to add the instructor to your profile for COMP3123.

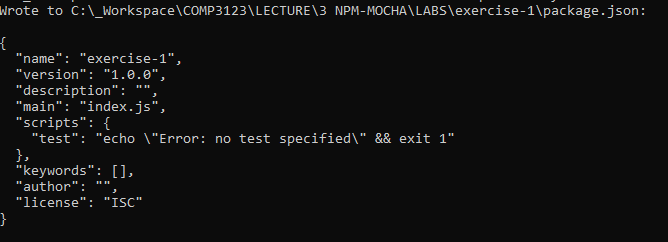
There will be a lab exercise to setup your GITHUB in lab 6.

1. Open a command prompt create a directory for **exercise-1**
2. Open Visual Studio Code and open the folder **exercise-1**
3. In command prompt go to the directory and run the following **npm init** command to create a **package.json** file

<https://docs.npmjs.com/cli/init>



The console will return the following text contained in the newly created **package.json** file

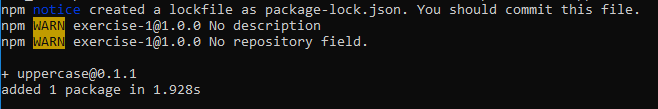


1. Next execute the following command to install 3rd party module from npm named **uppercase**.

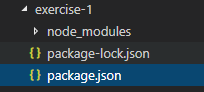
<https://www.npmjs.com/package/upper-case>



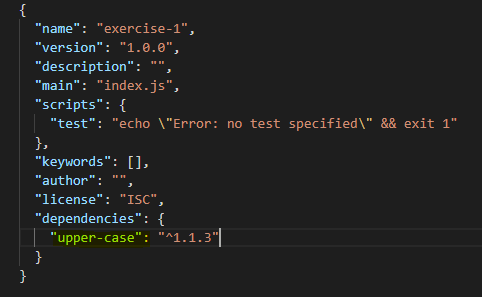
The console will return the following output



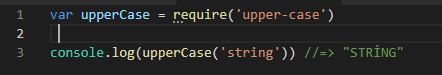
The folder structure will look as follows. The **package-lock.json** can be deleted.



The module reference to uppercase will now be listed as a **dependency** in the **package.json** file



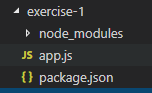
1. Create an app.js file and write the following code.



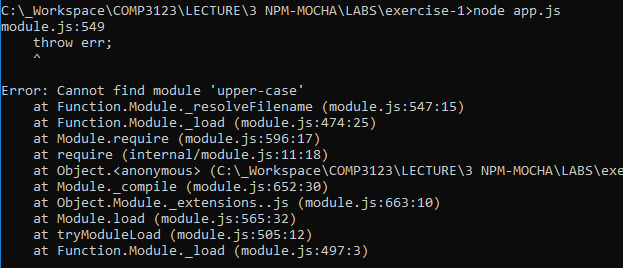
1. Run on the command line with **node app.js** and you will get the following output.



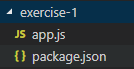
1. Browser your folder structure and delete the **node\_modules** folder



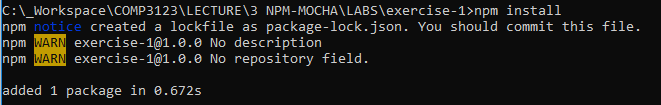
1. Repeat step 6 to run the **app.js** again and you will get the following error.



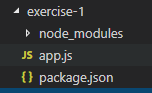
1. \* This is what the folder structure will look like when we deploy or submit our code. The **node\_module** will not be part of deployment. The **package.json** contain all the dependencies needed for your application.



Use the **npm -install** command and the **upper-case** module will be restored.



1. The **node\_module** folder will now be restored and when you run execute the **app.js** it will compile again.



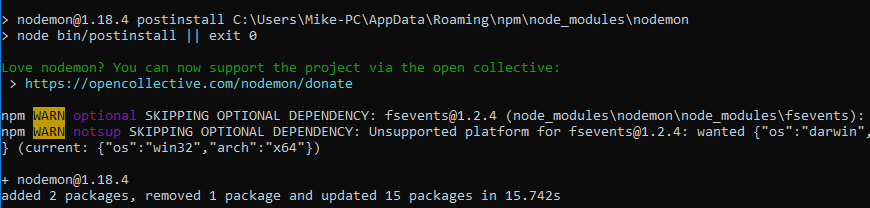


**Exercise 2: Nodemon + NPM Registry modules**

* 1. Open a command prompt create a directory for **exercise-2**
  2. Open Visual Studio Code and open the folder **exercise-2**
  3. Install **nodemon** (node monitor) from NPM using **install** using **–global** flag   
     <https://www.npmjs.com/package/nodemon>



This will **install nodemon** as a **global package** of Node on your computer to be used everywhere in all projects you create.

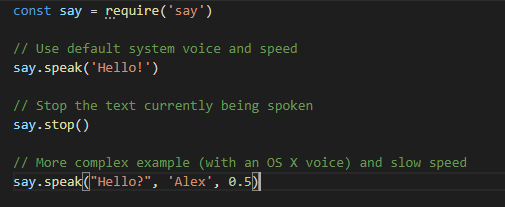


* 1. Install module named **say.js** from NPM Registry using install and **–save** flag

<https://www.npmjs.com/package/say>



* 1. Create an **app.js** file and write the following code



* 1. Use the newly installed **nodemon** to run the **app.js**. The text should be played through on your pc system voice.



* 1. Create a new callback function that will use the **say.speak** command   
     ie. A function named sorryDave speaks the text “I’m sorry, Dave”
  2. Use the **setTimeOut** function to trigger a callback after a delayed amount of time ie. 5 seconds, so that the text is spoken after the first “Hello, Alex”

<https://javascript.info/settimeout-setinterval>

**Exercise 3: Mocha + Testing**

1. Open a command prompt create a directory for **exercise-3**
2. Open Visual Studio Code and open the folder **exercise-3**
3. Run **npm init -yes** to setup the **package.json** file



1. Run **npm install** **should** **–save—dev** to save **should.js** as a development dependency



1. Run **npm install mocha –save—dev** to should mocha.js as a development dependency



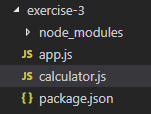
1. Run **npm install mocha –global** to install mocha globally to be used elsewhere.



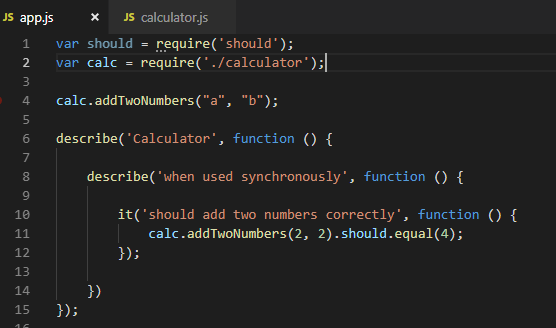
1. Run **mocha –version** to check if it is installed correctly



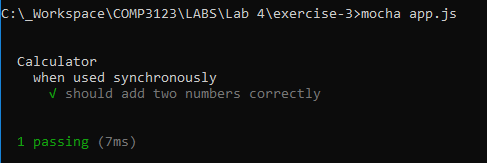
1. Create a file named **calculator.js** and create a method named **AddTwoNumbers**. Using module.exports export this method to be used in app.js



1. In **app.js** write the following mocha test case.



1. To run the mocha test on the command line use **mocha app.js**



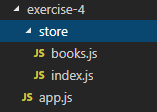
1. Using the should.js documentation create a couple more test cases using notEqual and throws

<https://shouldjs.github.io/#should-notequal>

1. Create a second method named **SubstractTwoNumber** in **calculator.js** module. In **app.js** require it and write some of the same test cases.
2. Bonus: Try to create a async method **AddTwoNumberAsync** using **SetTimeout** that takes callback and returns a result. Write an async Mocha test for this.

**Exercise 4: Web Server + Http Routing**

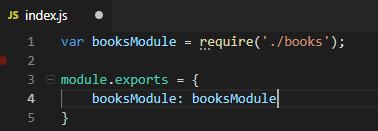
1. Open a command prompt create a directory for **exercise-4**
2. Open Visual Studio Code and open the folder **exercise-4**
3. Create a folder named store and create the following files to organize the file structure in the following way.



1. Add the following code to the **books.js** file from the following location

[**https://drive.google.com/open?id=1uS0WCI34\_fVCdeBl2OFlkXAc1\_gOsZ90**](https://drive.google.com/open?id=1uS0WCI34_fVCdeBl2OFlkXAc1_gOsZ90)

1. In the **index.js** file write the following code to include the **books.js** module



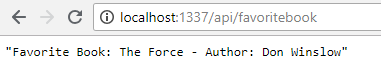
1. Copy and paste the starter Web server code from npm.org in the **app.js**

<https://nodejs.org/en/about/>

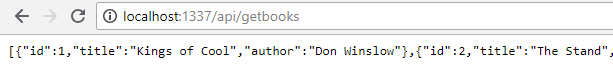


1. By inspecting the URL in the request handle the different routes and take the following actions:

* **The default route ‘/’**
  + Header response will be **200 OK**, with **Content-type application/json**
  + Return ‘No data found’ to the browser
* **The route ‘/api/favoritebook’**
  + Header response will be **200 OK**, with **Content-type application/json**
  + Use the book store module to return the following

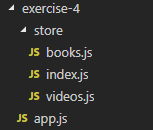


* **The route** **‘/api/getbooks’**
  + Header response will be **200 OK**, with **Content-type application/json**
  + User the book store module to return the following.



* **Unknown routes** that do not match any named routes
  + Header response will be **404 Not Found**, with **Content-type application/json**
  + Return ‘Error’ to the browser

1. Create a new **video.js** module and place it in the store folder.



1. Require it the same way as **book.js** in **index.js** file and include it in the **app.js** file. Use to resolve the following route and return data.

* **The route** **‘/api/getvideos’**
  + Header response will be **200 OK**, with **Content-type application/json**
  + User the video store module to return the following.

