Full Stack Development III– Lab 3

- NPM – Node Package Manager

Node Reference

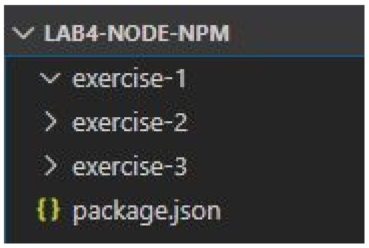
An optional reference for Node JS

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.

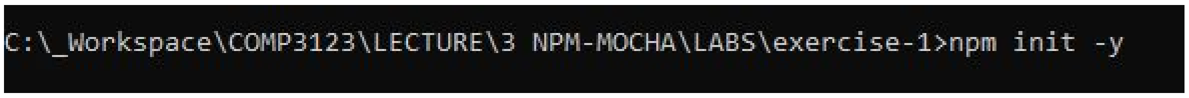


```
LAB4-NODE-NPM
  exercise-1
  exercise-2
  exercise-3
  package.json
```

Exercise 1

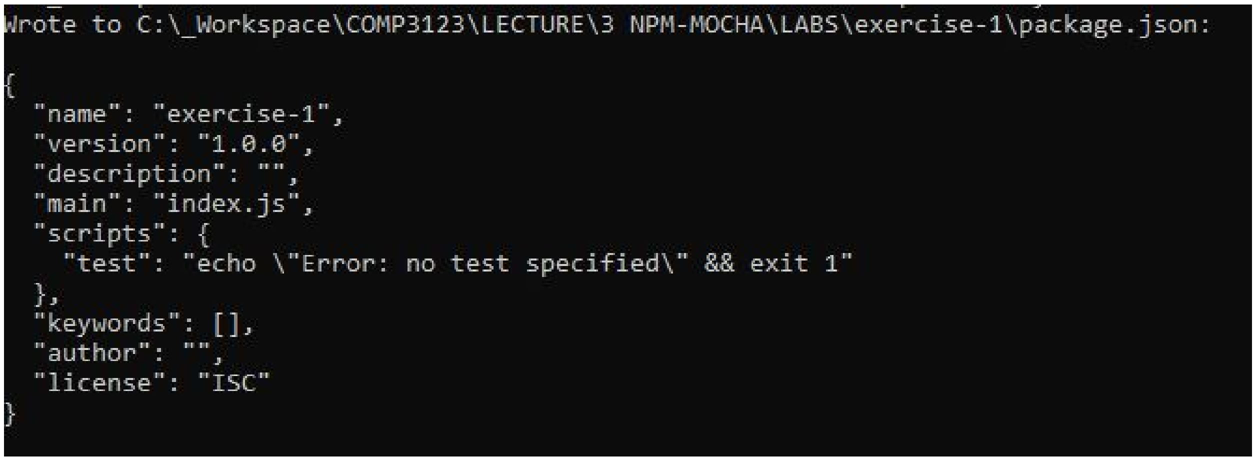
1. Open a command prompt create a directory for **exercise-1**
2. Open Visual Studio Code and open the folder **exercise-1**
3. In command prompt go to the directory and run the following **npm init** command to create a **package.json** file

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



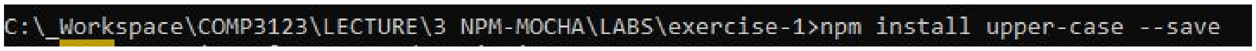
```
C:\_Workspace\COMP3123\LECTURE\3 NPM-MOCHA\LABS\exercise-1>npm init -y
```

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



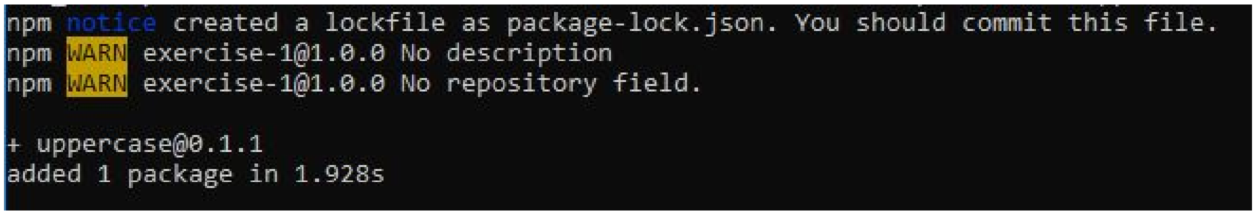
```
Wrote to C:\_Workspace\COMP3123\LECTURE\3 NPM-MOCHA\LABS\exercise-1\package.json:

{
  "name": "exercise-1",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

4. Next execute the following command to install 3rd party module from npm named **uppercase**.
https://www.npmjs.com/package/upper-case



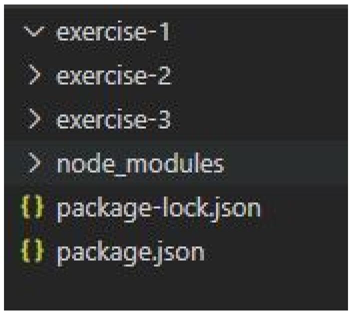
```
C:\_Workspace\COMP3123\LECTURE\3 NPM-MOCHA\LABS\exercise-1>npm install upper-case --save
```

The console will return the following output



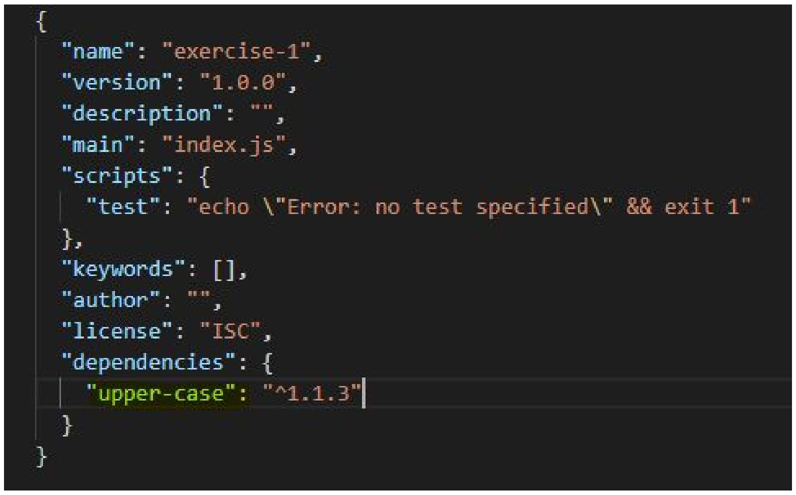
```
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN exercise-1@1.0.0 No description
npm WARN exercise-1@1.0.0 No repository field.

+ uppercase@0.1.1
added 1 package in 1.928s
```

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



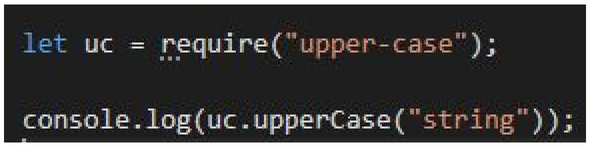
```
∨ exercise-1
> exercise-2
> exercise-3
> node_modules
{} package-lock.json
{} package.json
```

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



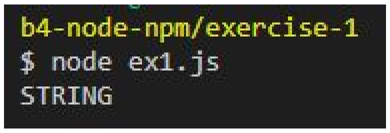
```json
{
  "name": "exercise-1",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "upper-case": "^1.1.3"
  }
}
```

5. Create an **ex1.js** file and write the following code.



```javascript
let uc = require("upper-case");

console.log(uc.upperCase("string"));
```

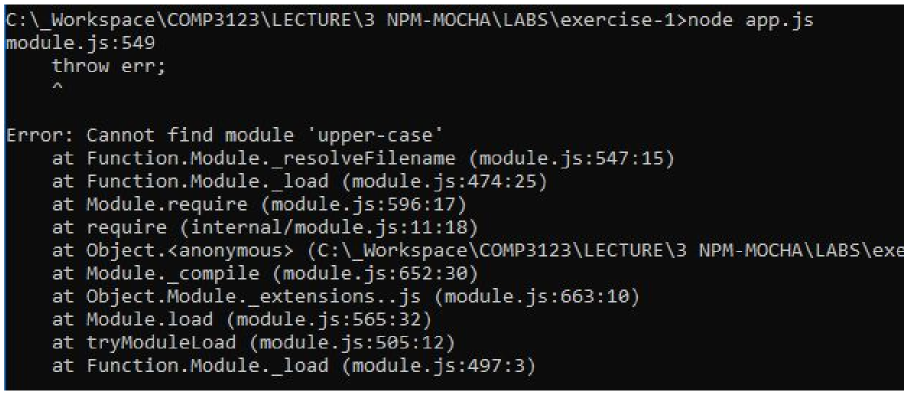
6. Run on the command line with **node ex1.js** and you will get the following output.



```
b4-node-npm/exercise-1
$ node ex1.js
STRING
```

7. Browser your froot older structure and delete the **node_modules** folder

8. Repeat step 6 to run the **ex1.js** again and you will get the following error.

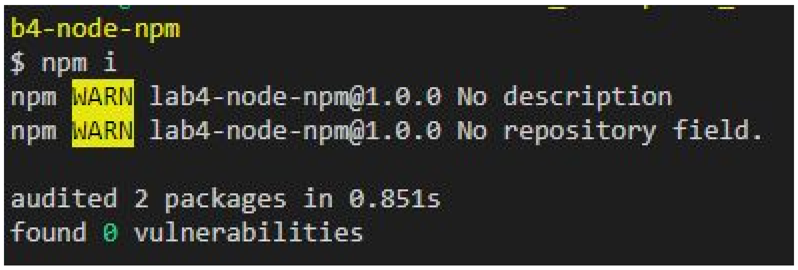


```
C:\_Workspace\COMP3123\LECTURE\3 NPM-MOCHA\LABS\exercise-1>node app.js
module.js:549
    throw err;
    ^

Error: Cannot find module 'upper-case'
    at Function.Module._resolveFilename (module.js:547:15)
    at Function.Module._load (module.js:474:25)
    at Module.require (module.js:596:17)
    at require (internal/module.js:11:18)
    at Object.<anonymous> (C:\_Workspace\COMP3123\LECTURE\3 NPM-MOCHA\LABS\exe
    at Module._compile (module.js:652:30)
    at Object.Module._extensions..js (module.js:663:10)
    at Module.load (module.js:565:32)
    at tryModuleLoad (module.js:505:12)
    at Function.Module._load (module.js:497:3)
```

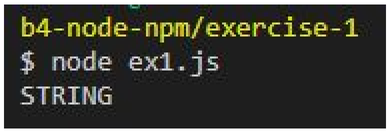
9. * This is what the folder structure will look like when we deploy or submit our code. The **node_module** will not be part of deployment. The **package.json** contain all the dependencies

needed for your application.

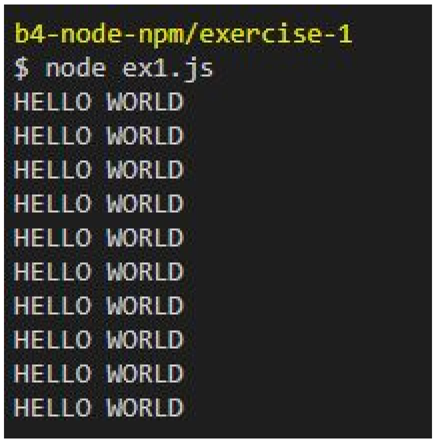
Use the **npm -install** in the root folder command and the **upper-case** module will be restored.


```
b4-node-npm
$ npm i
npm WARN lab4-node-npm@1.0.0 No description
npm WARN lab4-node-npm@1.0.0 No repository field.

audited 2 packages in 0.851s
found 0 vulnerabilities
```

10. The **node_module** folder will now be restored and when you run execute the **app.js** it will compile again.

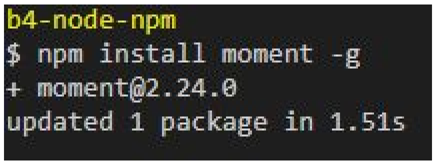

```
b4-node-npm/exercise-1
$ node ex1.js
STRING
```

Task: Create a function named greeter in app.js that will use upper case and output hello world 10 times and log it to the console.

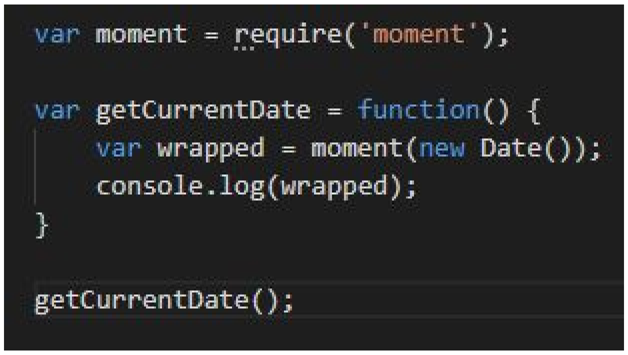

```
b4-node-npm/exercise-1
$ node ex1.js
HELLO WORLD
HELLO WORLD
HELLO WORLD
HELLO WORLD
HELLO WORLD
HELLO WORLD
HELLO WORLD
HELLO WORLD
HELLO WORLD
HELLO WORLD
```

Exercise 2: Using require and moment.js

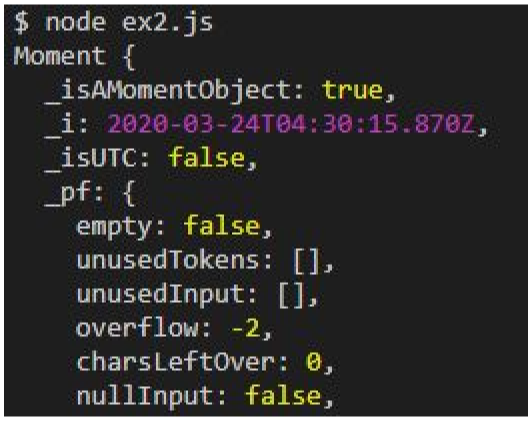
i. Download moment.js using npm


```
b4-node-npm
$ npm install moment -g
+ moment@2.24.0
updated 1 package in 1.51s
```

ii. In the **exercise 2** folder, create a new file named **ex2.js** using require to make **moment.js** available in your code and run using node. The output will be a large moment date object



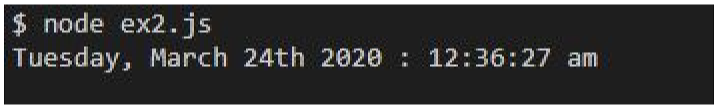
```
var moment = require('moment');

var getCurrentDate = function() {
    var wrapped = moment(new Date());
    console.log(wrapped);
}

getCurrentDate();
```



```
$ node ex2.js
Moment {
  _isAMomentObject: true,
  _i: 2020-03-24T04:30:15.870Z,
  _isUTC: false,
  _pf: {
    empty: false,
    unusedTokens: [],
    unusedInput: [],
    overflow: -2,
    charsLeftOver: 0,
    nullInput: false,
```

iii. Task 1: Convert the javascript to **ES6** syntax

iv. Task 2: Using the **moment.js** documentation find a way to parse the date and time into separate strings and output to console. https://momentjs.com/



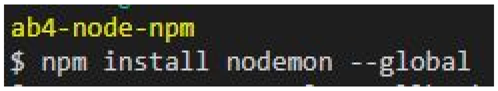
```
$ node ex2.js
Tuesday, March 24th 2020 : 12:36:27 am
```

Exercise 3: Nodemon + NPM Registry modules

1. Open a command prompt create a directory for **exercise-3**

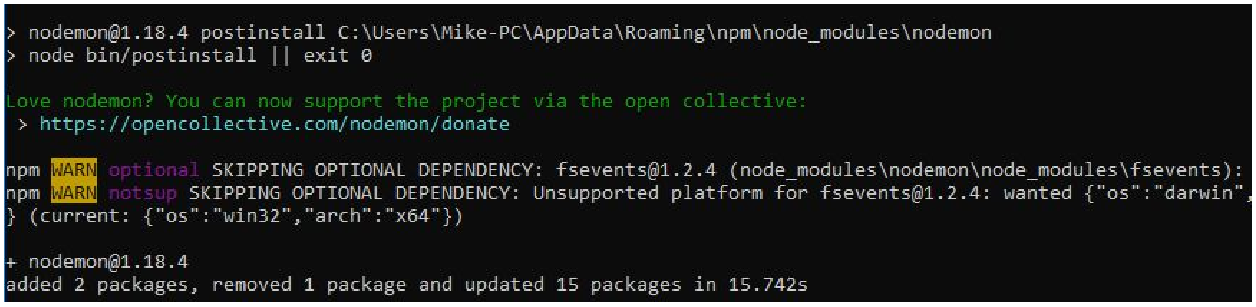
2. Open Visual Studio Code and open the folder **exercise-3**

3. Install **nodemon** (node monitor) from NPM using **install** using **–global** flag
   https://www.npmjs.com/package/nodemon



```
ab4-node-npm
$ npm install nodemon --global
```

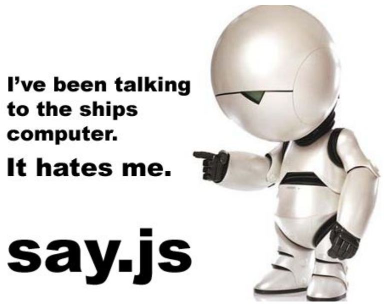
This will **install nodemon** as a **global package** of Node on your computer to be used everywhere in all projects you create.

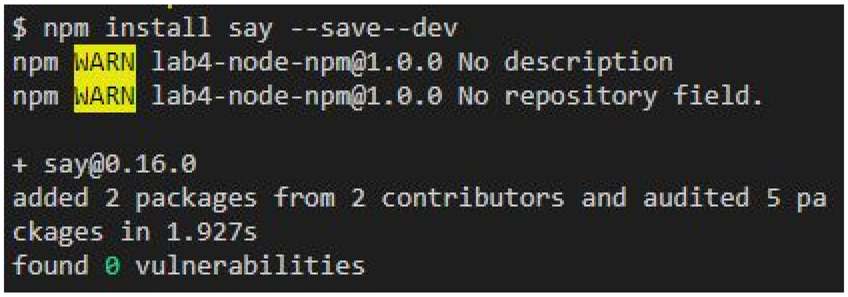


```
> nodemon@1.18.4 postinstall C:\Users\Mike-PC\AppData\Roaming\npm\node_modules\nodemon
> node bin/postinstall || exit 0

Love nodemon? You can now support the project via the open collective:
 > https://opencollective.com/nodemon/donate

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.4 (node_modules\nodemon\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin",
} (current: {"os":"win32","arch":"x64"})

+ nodemon@1.18.4
added 2 packages, removed 1 package and updated 15 packages in 15.742s
```

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

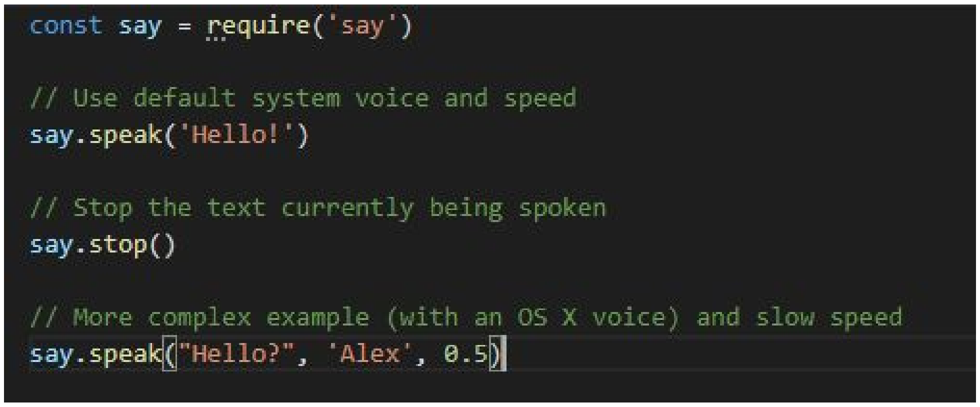
https://www.npmjs.com/package/say





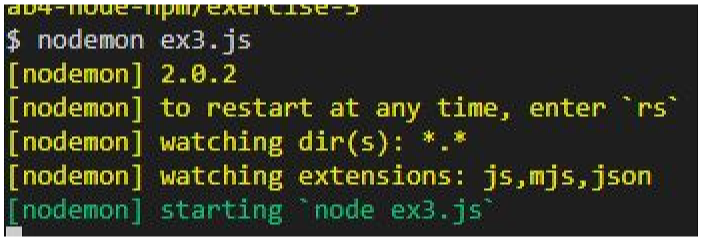
```
$ npm install say --save--dev
npm WARN lab4-node-npm@1.0.0 No description
npm WARN lab4-node-npm@1.0.0 No repository field.

+ say@0.16.0
added 2 packages from 2 contributors and audited 5 pa
ckages in 1.927s
found 0 vulnerabilities
```

5. Create an **ex3.js** file (file name does not matter as long as it is in the exercise 3 folder) and write the following code



```
const say = require('say')

// Use default system voice and speed
say.speak('Hello!')

// Stop the text currently being spoken
say.stop()

// More complex example (with an OS X voice) and slow speed
say.speak("Hello?", 'Alex', 0.5)
```

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



```
ab4-node-npm/exercise-3
$ nodemon ex3.js
[nodemon] 2.0.2
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ex3.js`
```

7. Create a new callback function that will use the **say.speak** command
   ie. A function named sorryDave speaks the text “I’m sorry, Dave”

Challenge: 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