CMPS 350 Web Development Fundamentals - Spring 2021
Lab 11- AsyncJS

Objective

The objective of this lab is to practice JavaScript asynchronous programming using **Callbacks**, **Promises** and **async/await**.

Overview

This lab has three parts:

- **Part A:** Practice JavaScript Unit testing using Mocha and Chai (1h).
- **Part B:** Practice Callbacks, Promises and Async-Await (1h 50mins).

Part A – Unit Testing Using Mocha and Chai

1. Sync cmps350-lab repo to get the Lab files.
2. Copy ***Lab11-AsyncJS*** folder from cmps350-lab repo to your repository.
3. Open ***Lab11-AsyncJS\UnitConverter*** in Webstorm. You should see a JavaScript file named *UnitConverter.js*. In this exercise, you will create a spec file to unit test the function of the *UnitConverter* class.
4. First, create the package.json file using `npm init .` This file is used to define dependencies by listing the npm packages used by the app.

   Refresh your project to see the **package.json** file.
5. Install mocha and chai using *node package manager* (npm):

```
npm install mocha -D
npm install chai -D
```

   This will add 2 dev dependencies to package.json file.

```
"devDependencies": {
  "chai": "^4.1.2",
  "mocha": "^5.0.4"
}
```

6. Create a JavaScript file named **UnitConverter.spec.js**
7. Import an instance the *UnitConverter* class to be tested and the *chai expect* package.

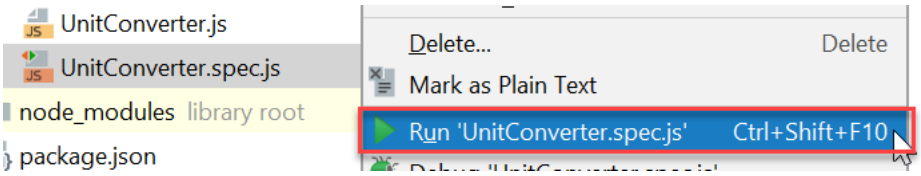
```
const unitConverter = require('./UnitConverter');
const expect = require("chai").expect;
```

8. Write 2 test cases for each method of **UnitConverter** class.

You may start with the following inputs and expected results. Then use search for "google unit converter" to compute the expected results for more input values.

Method	Input	Expected Result
kgToOunce	1	35.274
kgToPound	2	4.4092
meterToInch	1	39.3701
meterToFoot	2	6.5617

9. Run the unit tests as you develop them using WebStorm:



Also run the unit tests from the command line using: `npm test`

But first, you should have the following in package.json file.

```
"scripts": {
  "test": "mocha **/*.spec.js"
},
```

Part B – Practice Callbacks, Promises, and Async-Await

In this exercise, we will collaboratively build the solution of each step after you attempting it by yourself. Then the instructor will demonstrate and explain the model solution. At the end of this exercise, you should be able to use **callbacks**, **promises** and **async/await** to read/write files. This will enable you to complete the remaining tasks of the Lab.

1) Callbacks

Open the file **project1.js** in the attached folder. You will find the following code:

```
//Synchronous code. Change it to async using callback.
const fs=require('fs');
//Synchronous code. Change it to async using callback.
let data = fs.readFileSync('data/student.json');
console.log(JSON.parse(data));
```

Implement the following tasks in the same order.

a) Convert this code to asynchronous form using a separate callback function (callback).

b) Change the callback function to an anonymous one.

c) Take care of error handling in the callback function.

2) Nested Callbacks

Open the file **project2.js** in the attached folder. You will find the following code:

Implement the following tasks in the same order using callbacks.

a) We need to read data from two files. **course.json** and **staff.json**. Both using callbacks.

b) We need finally print all courses with their corresponding instructor names.

   i) Instructor name can be found at the staff file.
   ii) Use **staffNo** in **staff.json** property to match the instructorId from course.json

c) Create two functions **getCourses** and **setInstructorNames**.

```
function getCourses(cb)
function setInstrctorNames(courses , cb)
```

d) Instructor names are set as a new property to the course object in the **setInstrctorNames** function.

e) Your output is expected to be similar to the following:

```
{
  crn: 107,
  courseCode: 'GENG 107',
  courseName: 'Engineering Skills and Ethics',
  semester: 'Fall 2015',
  instructorId: 12,
  instrctorName: Abdulahi Hassen'
 },
```

3) Promises

Implement the following tasks in the same order.

a) **Switch to using fs-extra instead of fs.**

- **To install: npm install fs-extra**

b) **In project3.js :** Rewrite the code you created in **Part-1** using **promises**.

c) **In project4.js :** Rewrite the code you created in **Part-2** using **promises**.

4) Chaining Promisses

a) **In project5.js :** Reuse the code you created in **project4.js** adding a new promise function to find the number of students in each course.

   i. You need to use the student.json file to find the courseIds.
   ii. Match it with crn property from the course.json file.
   iii. Use map and reduce.

b) Your output should be similar to the following:

```
{
  crn: 300,
  courseCode: 'GENG 300',
  courseName: 'Numerical Methods',
  semester: 'Fall 2017',
  instructorId: 33,
  instrctorName: 'Saleh Alhazbi',
  studentCount: 50
 },
```

5) Async/Await

a) **In project6.js :** Rewrite the code you created in **project4.js** using **Async/await**.

b) In project7.js : Rewrite the code you created in **project5.js** using **Async/await**.