Lappeenrannan teknillinen yliopisto
School of Business and Management

Software Development Skills

Alexey Kokorev, 0598010

LEARNING DIARY, <CHOSEN MODULE NAME> MODULE

LEARNING DIARY

23.05.2020

The environment is ready.
The code editor is, of course, VS Code.
Refreshed my knowledges about git, git in the command line, git in the vs code plugin, branches, mergins, pull, push, commits, etc.
I have just took some concepts about the back end too (regarding node js, express, body-parcer, etc). So, ready!:)

26.05.2020

Did not understood, what exactly materials should be provided as a mandatory part «Material from the exercise projects that you perform, following the tutorial series», but might be get it later.

From the «introduction into nodeJS» I've found out - how to create a simple server without EXPRESS library.
And also tested - how to create and send the headers, in order to let the browser know - that the server answered 200(ok), and to let browser know - which type of data will outcome (text/plain or text/html).
I've installed a mongoDB as a local service, it was a bit tricky to perform that on Mac OS.

27.05.2020

I've watched the part of the course regarding the mongoDB, got basic knowledges about it. And now I have an idea for a full-stack project, that will be definitely useful for me, and might be for others too.
And, of course, I'd like to deploy that project then. The heroku would be the best free solution as now. But I will find out - is the mongoDB supported on the heroku for free, or I will have to use a cloud version of the mongoDB. But as for me, if there will not be a free solution using mongoDB, I'd prefer rather using just the files via nodeJS (files as DB).

28.05.2020

I've decided to use a mongoose driver in order to work with a mongoDB in the nodeJS.
The mongoose is better then the native «mongodb» driver, since the mongoose simplifies the code required to connect and manipulate the db within nodeJS.
And i decided to add an app.js file into the «mongoDB» section of my module materials.

29.05.2020

The mongoose has been successfully installed, and tested by a working code.
The CRUD operations are have been learned now too, and they are all represented in my app.js fully working code with a mongoose library. The materials regarding mongoDB are located in the folder «materialsFromExerciseProjects». That folder contains also another parts of the module (such as nodeJS materials, express, e.t.c.).

And now I feel my self needing to learn about EJS first, before I'll start working together front- and the back-end. Of course, I'll try to connect the front- and the back-end by the Ajax (fetch function without these garbage old requests types…). But the template system as EJS would be the nice to know.

01.06.2020

I learned today about the Express.js. Installing, starting server, working with a POST and GET requests.
Body-parser plugin with Express is a great solution passing data from Front-Eend to Back-End.

I used a EJS templating system. The EJS allows to pass data from back-End to Front-End, and vice versa.
EJS allows to use a javascript with a node's variables (from the back-End) right inside an html template. I used a forEach loop through an array to create a list of items. These items were passed from back-End side. The input field and «add» button allows to add one more item into an array, then redirect it back into front-End.
In the back-End we are using «render» function in a app.get section. But in a app.post section we are just working with an already existing variables, and with a data from body.
Then, in that app.post section we passing data back to front-End by «redirect» command, which not only redirects, creating a get request, but also pass a new data from back to forward.
And got knowledges - how to include another ejs files (such as footer, header) into a main template by a command <- include(«templateName»); ->

02.06.2020
I used «app.use(express.static(«public")); » to make the static files, (such as styles, scripts) work, and to have an access to them from the client.

Refreshed knowledges about how to use module.exports (from another js file).
In my project, I've used the typical express's routing and the Ajax requests from the client side, and responses from a server side.

The best practice, as I feel, would be not using the routing at all, but load the whole pages with an JQuery's.load() function. But decided not to overwrite almost whole project for that way.

03.06.2020

The async + await with a try… catch works also in the node js too perfectly. And with an examples from another courses the async hasn't been used with databases. But in my case, the async await was an amazing decision in order to work with data from db only after full process of getting that data's finished.
After creating a new page, user will be provided with a key for His/Her page, and that key should not be published, otherwise, others can run the page too. The passwords weren't used because it would not be great idea to enter the password any time, when the page needs to be executed (opened for running).

04.06.2020

The routing with parameters allow me to pass a user's key (id for a user's page) in order to run or edit the page.
Each task, user has added with it's options will be saved as an object after «save» button is pressed. After list of tasks is parsed, each object is stored inside the array (taskList). And eventually, that array with all tasks will be saved inside the db. I did not use any safe rules to avoid any injections or so, because, the safety would take much more time, and You should be a safety coding professional in order to properly protect Your site from breaking by hackers.

05.06.2020

Each task's object consists of a «taskName», that's a name of the task, and options, that user has selected during adding that task to a taskList.
I got an error, saying something like unexpected end of the response, or so. Found out, that such error occurred because of next thing:
The data after https.get can be given to server by many chunks (didn't know that), and there should be used an array to hold that, concatenate it, and only on.end(… can be given to a client (res.json)…

06.06.2020

Working on front-end side, I've found, that even if I use the script on the end of the page(call it in the end of the page), I anyway should always use any of jQuery commands after window is loaded jquery(window).on('load', function…). Otherwise, the safari in a iPhone will not work properly with a jquery.

07.06.2020
Calling javascript functions from the templates, I learned, that if I pass an objects as a parameter for the function, I have to JSON.stringify(…) them. Otherwise, browser won't be understanding that data.

08.06.2020

In order to prevent user entering His/Her key each time, when the main page's loaded, I used a jscookie library. And this library will not work in iPhone's safari, if we will not be waiting till window loaded.
Bootstrap has been used to make the app more responsive.
Also, I added a jQuery, which helped me much reducing the amount of code written.

09.06.2020

As a db, I used mongoDB on a local, but then, I used a heroku's one. Which is provided free by mongoLab for a 500mb. Heroku has it's own addon, that allows users to save time avoiding the db's configuration. So, using heroku for mongoDB setup is much faster, then mongoLab's setup requires. Downside only needing to add a creditcard information, that slows down the process).
And heroku's configurated db connection works also if I run the project locally.

10.06.2020
Await + async should be used both in functions, that I've wrote for working with db, and when these functions are called.
I've created a delay function, that allows to make a delay before proceed to a next task.
This beautiful example: «*const* delay = *ms* => new *Promise* resolve => setTimeout () => resolve(), *ms*));
»
And that Promise is convenient to use like this:
await delay(1000);

11.06.2020

All user's tasks are performing on a client's given function, which accepts as a parameter an array with objects.
In cases with an api keys, I saved them in a server's side, of cause. And was getting the data by an Ajax requests.
I used the fetch function, but jquery's get or post functions are faster to write, and these functions are promises too.

12.06.2020

Interesting foundation for me was, that async await doesn't work properly inside the forEach loop. So, the idea was to not perform anything, on-till the current forEach's iteration Is running. But, actually, all iterations of forEach performed without waiting my delay function returned.
And if I'd took try… catch block out of forEach's scope, I'd get an error. So, as salvation, forEach has been replaced with a classic for loop. And in that case, the next iteration is performing only after first's await functions are returned.

13.06.2020

Trying to find a really interesting api, I discovered next few:
Weather conditions api from openweather;
News api from newsapi, but their data is not quite structured sometimes (null easily can be returned, wrong parsed data exists. ;
NASA's «APOD» (Astronomy Picture of the Day» is really beauty and neat stuff;
And the currency rates are really useful sometimes.

14.06.2020

The «wait for click» task fits well for my project. The click listener adding dynamically on a document. (by jQuery). And as a callback, it changes a variable when user click.
After that, the do… while loop checks that variable's state each 1/4 second (delayed by my «delay» await function .
Jquery's effects, such as slideDown/up, fadeIn/Out don't wait till the end of the effect, but code continue. In order to wait till the end of the each effect, I delayed the code.

15.06.2020

Working with a json's data, there is a good way to use a json path pickers. They simplify the json's structure understanding process.
There are plugins for a chrome browser exist.

16.06.2020

One problem appears, after I added a «wait for click» functionality. On the iPhone's safari it won't work. Only on the elements with a «cursor:pointer;». But I need it to be working all across a document. That's why, the next decision camy useful (from stack overflow forum):

```
jQuery(document).on('click touchstart tap', function(e) {
```

17.06.2020

I've discovered a jQuery UI. And that's a really great functionality for a future feather projects. I'll be keeping in mind about that one.
Dragdrop methods, sortable lists and grids are really useful things from that. And especially I've liked an autocomplete example from there. That example could be neat to use while user's adding a City option for a «weather in certain city» task.

18.06.2020

Spending few hours discovering how to bundle (compile) an executable apps from a javascript (css, html optionally), I've red about the «electron» and the «nwJs». But also possible to bundle a functional app using a «pkg» (npm Install pkg). And the final apps work without even node installed on a user's computer.

19.06.2020

For node js there is an interesting thing, called «robotJS». This thing can «listen» a mouse and keyboard events, produce the mouse events, move, keyup/down, etc. Also it allows to get an images from screen.

20.06.2020

Thanks for this course! I've liked it very well, especially, creating my absolutely own project by my own idea! 👍