Lappeenrannan–Lahden teknillinen yliopisto

School of Engineering Science

Software Development Skills

Jani Heinikoski, 0541122

LEARNING DIARY, FULL STACK MODULE

**LEARNING DIARY**

22.07.2022

I started by carefully studying the course Moodle page. I learnt that the main technology stack used in this module is the so-called MEAN-stack (MongoDB, Express JS, Angular JS and Node JS). I have previously completed the mobile and front end modules of these SDS-courses so the Moodle page was very familiar. I noticed that this module follows the same paradigm as the other two modules I completed (studying and doing exercises/coursework, writing a learning diary and finally demonstrating my new skills with a unique project).

I began by creating a new Git code repository for this module. I took the opportunity to learn something new about git as well by watching YouTube videos and reading Git documentation about branches. I might utilize branching in Git to test some features of the project in their isolated branches before merging them to the main branch. However, I already had Git installed (specifically GitHub Desktop -application on Windows) so there was not too much to learn when setting up the code repository. Furthermore, I have already setup Visual Studio Code (my go-to code editor).

I continued to the module’s task list and started at the first one called “NodeJS”. I noticed that there are two videos available: an updated video and an old one. I took the liberty of assuming that only the updated video needs to be watched. The first ten minutes of the video focused on talking about Node and npm on a general level both of which I am already familiar with. Although, I did learn that Node has been written using C++. The non-blocking nature of Node, its single threaded event loop (or rather the JavaScript event loop) and also the Node package ecosystem were familiar concepts to me so the first ten minutes was mostly a recap of the main points about Node and npm.

Next part of the video concentrated on installing Node which I had already installed. It was recommended to install Node version 12 in the course Moodle page so I just used nvm (Node version manager) to switch my current version to 12.22.6 (which I had already downloaded using nvm).

Next up I initialized the first module’s working directory with npm init to be able to install third party node packages. I installed nodemon as a “dev dependency” even though I have it globally installed on my machine. Then I added a “gitignore”-file on the coursework folder to ignore node\_modules folders and package-lock.json files in all subdirectories. Again, installing dependencies (other node packages) was familiar to me.

Next, I coded along the simple examples shown in the video. I did not learn much but I did not know about the module wrapper function which wraps the node modules before runtime giving them access to variables: exports, require, module, \_\_filename, and \_\_dirname (which I previously have used but had not thought why they are accessible).

Traversy Media (author of the Introduction video) briefly brushed on the topic that Node did not support ES6 modules (specifically the import/export syntax) at the time of making the video. I always found this topic very confusing because I have stumbled many times on problems with mixing CommonJS module system with the ES6 module system. I Googled for a good minute about the topic and from what I gathered Node has supported ES6 modules since version 12 and has become the official standard format to package JavaScript code for reuse. To tell Node to use the ES module system, the type property can be used in the package.json file or the .mjs file extension can be used.

26.07.2022

I continued working on the first module. The video continued with covering some of the important core modules of Node such as path. I learnt about few new useful functions about the path module such as path.parse -function (I already knew about path.join). Next, I learnt how to create folders, read, rename and write files with the fs and path modules. After the fs module, the video covered the os module which I was already familiar with so it was mostly a recap of the most useful functions in the os module. I learnt how to handle URLs with the, you guessed it, url module. I learnt how to create custom event emitters and listeners with the events module by creating a logger class. After following along with the reference files, I developed the simple HTTP server as demonstrated in the video however I did not learn anything new while doing it because I have been developing HTTP servers with Node and Express quite a while. Finally, I learnt how to deploy the app to Heroku using the Heroku CLI tools. The app is deployed at <https://secret-shelf-45961.herokuapp.com/>. I installed and used the Git for Windows and Heroku CLI tools to deploy the app.

I began working on the second MongoDB module. As during most of the video, no source code files are produced so I have taken screenshots from some of the queries I performed while watching the video. I have worked with MongoDB many times before primarily in combination with NodeJS and a third-party object modeling tool/package: Mongoose. The video began with introducing briefly what MongoDB is and how to install it. I have already installed the MongoDB community edition on my machine. The first thing to do was to create a new database with a new collection called posts and insert a document in it. These were very basic things, and I did not learn anything new. The next topic was querying the database which I am also already familiar with.

27.07.2022

I continued the MongoDB module and the next topic was updating documents in the database. I have not used pure MQL (Mongo query language) to update documents (as previously mentioned I have worked primarily with the Mongoose package) so this was new to me. The syntax was very familiar to me because working with the Mongoose package is somewhat like writing straight up MQL and I am familiar with the basics of MQL. I learnt how you can replace whole documents or just update/modify certain fields of them. Next topic was about creating subdocuments, but I prefer creating a separate collection and having “foreign keys” (object ids) especially when a certain document is a child to multiple documents. Next, I learnt about creating indices and performing text searches on documents’ fields which I have utilized many times with relational databases but never in MongoDB so that was something very interesting and good to learn. Rest of the video focused on the Compass and Atlas applications both of which I have used previously so it was mostly a recap (I mostly prefer working with the mongo shell and the community server).

I started the third module which concentrates on Express: a framework/package for creating HTTP servers. I have used Express many times, altogether for around 200 hours so I do not expect to learn much new during the crash course video. The first ten minutes of the video focused on what Express is and I did not learn anything new which was no surprise. I also have Postman and NodeJS already installed on my machine so I do not need to install any new software. The next 40 minutes of the video concentrated on very basic Express stuff such as creating middleware, creating routes with the router object, creating a simple CRUD REST API, serving static assets etc. I did not learn anything new but it was a nice quick recap on Express. I committed my work often to Git with descriptive commit messages so that it is clear I knew what I was doing and not just copying blindly (I also fixed a few bugs I noticed on the video). I left the rest of the video for tomorrow.

28.07.2022

I finished the Express module (there was not much left as for the video). I learnt how to use the Express Handlebars -view engine. I have not really used view engines because I have mostly created my frontends with React but Express Handlebars seemed very intuitive and simple enough to use at first glance.

I began working on the Angular module where I expect to learn the most during this course because I have never used Angular before. I only know that it is a framework currently maintained by Google for creating SPAs (single-page applications) and it is comparable to React/Vue. I decided to first read the *Introduction* and *What is Angular* from *Getting Started* chapters from Angular’s docs (<https://angular.io/docs>) before proceeding with the module’s tutorial.

First, I learnt about Angular’s component and template system which was a bit different from React. The components live in their own files as in React but they are defined as classes with a special decorator @Component. The component system at a first glance seems simple and a bit more decoupled compared to React (HTML templates dictate the UI and the logic lives in a separate TypeScript class as opposed to React’s JSX where business logic and UI are loosely coupled).

Next, I learnt about managing state by playing around with the example project in Angular’s What is Angular -chapter. It was very simple as Angular takes care of updating the UI when the bound fields change. Using them is simple: wrapping the field’s name in double curly braces tells Angular to interpolate the field’s content in the HTML. The property binding system is also very straight forward: wrapping HTML attributes in the template file within square brackets allows them to be bound into the component’s fields.

I did not yet fully understand the use of directives but if I understood correctly, they can be used to implement logic/custom behavior within the template. The last part of the What is Angular -chapter was the design pattern called dependency injection which I am semi familiar with. It allows the instantiation of dependencies (classes) to be transferred from the programmer’s/component’s responsibility to Angular’s injectors’ responsibility.

I completed the Getting Started -chapter’s example tutorial on StackBlitz and I learnt how to pass data and react to events between child and parent components. I also glanced through on how to use the Angular Router to implement navigation in a SPA.

I began working on the *Tour of Heroes* -tutorial which is the primary focus of this module. After setting up my local environment by installing the Angular CLI -tool, I proceeded to create a new workspace in the /coursework/angular folder called angular-tour-of-heroes with the skeleton project called app.

After opening up the Angular’s skeleton project I noticed that the package-lock.json file was not included in the .gitignore file which had me confused for a while because I have previously thought that it should not be included to source control. However, after doing a bit of Googling I found the following StackOverflow thread: <https://stackoverflow.com/questions/44206782/do-i-commit-the-package-lock-json-file-created-by-npm-5> which states that it should indeed be checked in and installation should then be done using npm ci command instead of npm install. This ensures that all dependencies are the exact same version on each installation.

After finishing the 1. The Hero Editor -chapter of the tutorial, I learnt about formatting text with pipes (specifically the UppercasePipe). I also learnt how to include other files/libraries using the NgModule decorators (specifically I learnt how to import the FormsModule).

After finishing the 2. Display a List -chapter, I learnt how to use Angular’s class binding. I read a bit more from the documentation and experimented with the multi-class binding. Overall, the first two chapters did not teach too much new because I already went carefully through the Getting Started -chapter’s tutorial. I decided to leave the rest of the module for tomorrow.

29.07.2022

I continued working on the third chapter of the Tour of Heroes -tutorial. I did not learn anything new on the third chapter (only thing covered was one way data binding which I already learnt during the Getting Started -chapter). During the fourth chapter: *Add Services,* I learnt about the dependency injection system in Angular. I specifically learnt how to inject services that are provided by the root injector. I also noted that there exists a hierarchy of different injectors. I read more about the dependency injection system from the documentation, but I could not really get a good grasp about the providers and multiple injectors and their hierarchy. I suppose they will become more clear when I get used to Angular a bit more.

30.07.2022

I finished the fifth chapter: *Add Navigation* of the Angular Tour of Heroes -tutorial. I learnt a lot during this chapter because something kind of clicked when I worked on the chapter. Specifically, I better understood how the NgModules work. I also learnt more about how routing in Angular works however I had already gone through the basics in the Getting Started -chapter. But now I know how to add routing using the Angular’s RouterModule to an existing Angular application, how to create custom routes, how to navigate to them using the RouterLink directive and how to extract URL parameters.

I continued with the sixth chapter: *Get Data from a Server.* I first learnt about the HttpClient module and how to set it up by yet again adding it to the AppModule’s imports array. Then I learnt how to setup the Angular’s in-memory mock data API and how to query it using the HttpClient module. I also learnt how to handle errors when using the HttpClient with the RxJS catchError operator. I learnt how to use the RxJS tap operator to log messages when the observable values are emitted. I learnt how to use the HttpClient.get, HttpClient.put, HttpClient.post and HttpClient.delete -methods to retrieve, update, add and delete the mock data in the in-memory database.

02.08.2022

I finished the sixth and final chapter: *Get Data from a Server* of the Angular Tour of Heroes -tutorial. I learnt about the AsyncPipe which can be used to subscribe to an Observable (or a Promise). It automatically subscribes/unsubscribes when necessary and makes it easy to handle Observable data. I also learnt how to chain RxJS operators to limit the amount of HTTP requests made to the mock API. The RxJS HttpClient seems to be a very powerful tool when you need to create complex logic for communicating with an HTTP server however in my opinion the basic fetch API is suitable for most cases.