Lappeenrannan teknillinen yliopisto

School of Business and Management

Sofware Development Skills

Matti Bragge

LEARNING DIARY, FULL STACK MODULE

**LEARNING DIARY**

29.05.2022

I started by reading through the course page, which was familiar as I’ve completed two other modules before. I have version control set up, so I went ahead and started watching the newer NodeJS video.

The slides at the beginning of the video improved my understanding of what NodeJS is and when it’s typically used. Following the note below the video links, I downgraded my NodeJS installation from version 17 to 12.22.12 by using nvm-windows. Once that was done, I opened Visual Studio Code and ran npm init in the course folder I created. Next I installed uuid and nodemon, and learned about dev dependencies.

While following along with the video, I learned that to include variables in the middle of a string in JavaScript you use backtick characters and not quotes. During the path demo part of the video, I learned about the join function, which seems very useful to avoid delimiter errors. While doing the fs demo I ran into an issue which was caused by not having installed the modules globally. After that, I had no issues with the rest of the demos.

Doing the server creation helped remind me of some things I learned when doing the front-end module. It also connected the previous module demos well. Once I was finished with the server code and other parts, I signed up for a Heroku account and installed the Heroku CLI. To upload the app to Heroku I copied all of the files to a different directory so that the course GitHub repository is not affected. The deployment went without a hitch.

04.06.2022

I began watching the MongoDB video and installed MongoDB Community Server as a service. I learned about MongoDB’s syntax and functions by doing the same commands as shown in the video. During the text search part I had to look up why the escapes and quotes were needed. I found out from Mongo’s documentation (<https://www.mongodb.com/docs/manual/reference/operator/query/text/#std-label-text-operator-phrases>), that to match specific phrases you need to enclose the phrase in escaped double quotes. The rest of the video was clear.

06.06.2022

I started the Express JS crash course. I have no prior knowledge of Express, so the beginning of the video had a lot of new things for me. Continuing with the tutorial, much of it was similar to the NodeJS course so it helped hammer in the earlier lessons. Initially I found it somewhat hard to wrap my mind around routes and how Express works. There were also so many new functions introduced and little things to keep in mind that I will most likely need to revisit this video later.

During the later part of the video I ran into an issue with express-handlebars where I could not use exphbs as a function like in the video and instead had to use exphbs.engine. I found the solution on Stack Overflow: <https://stackoverflow.com/questions/69959820/typeerror-exphbs-is-not-a-function>

10.06.2022

I installed Angular CLI, which didn’t work at first since my Node version was incompatible. I looked up a compatible version of Angular (<https://gist.github.com/LayZeeDK/c822cc812f75bb07b7c55d07ba2719b3>) and installed version 13.3.7. Next I created the Tour of Heroes workspace. Continuing with the tutorial, I learned about:

* TypeScript syntax
* braces of interpolation
* creating components
* displaying components
* using pipes for formatting
* two-way data binding
* AppModule
* importing modules

12.06.2022

I continued the Angular course from part 2. I learned how to display a list, conditional html blocks and how to toggle a CSS class with a class binding. In the third part I learned how to separate a component into smaller components.

While working on the fourth part, I ran into an issue where the error message was: “TS2339: Property 'subscribe' does not exist on type 'Hero[]'.” After a few minutes I realized that I hadn’t saved my previous changes to the HeroService file, which changed the return type of getHeroes to Observable. The rest of the part went smoothly and I learned how to use services.

14.06.2022

I completed part 5 of the Angular course. It taught me about how routing works in Angular. The use of a router-outlet element to display different views makes a lot of sense in my mind, I had never thought about a solution like that before.

15.06.2022

I finished the final part of the Angular course. I had some trouble understanding how the InMemoryDataService works at first, I had to re-read that part multiple times. I also had to downgrade the InMemoryWebApi module to version 0.13.0 as 0.14.0 wasn’t compatible with my Angular version. I learned a lot about using HTTP with Angular and new things about Observables.

21.06.2022

I started watching the MEAN stack tutorial series. I watched the introduction video and part 2. At this point I’m getting used to making new apps with Node as I’ve done it a few times. There wasn’t anything new in the video except for mongoose.

23.06.2022

After watching part 3, I have a better idea of how mongoose works. I was originally confused because I didn’t manually create a database in Mongo, but it actually creates it automatically.

During part 4 I ran into an issue, I needed to use express-session in addition to passport in order to have session support. I looked up how it works from here: <https://www.npmjs.com/package/express-session>. After fixing that, I got another error: Expected “payload” to be a plain object. From the comments in the video, I found out that this can be fixed by changing the user parameter in jwt.sign(user, …) to an object: jwt.sign({user}, …). I had no issues while following along with the rest of the video.

27.06.2022

I started with part 5. I uninstalled the version of Angular I had and installed version 1.0.0-beta.28.3. After trying to run ‘ng serve’, I got an error saying “Invalid ‘reference’ directive”. I looked it up and I found a solution from here: <https://github.com/DefinitelyTyped/DefinitelyTyped/issues/10097#issuecomment-499783519>. Updating typescript to version 3.5.1 fixed the issue. Another thing that differed from the tutorial was that while generating a component with “ng g component” it couldn’t find the app module and I had to import the generated components myself.

Upon testing the navbar at 18:10 in the video, mine looked a lot different. It had a button that says ‘Toggle navigation’ and the login and register buttons were not visible. I dug through the comments and found that using a different bootswatch link (<https://bootswatch.com/3/sandstone/bootstrap.min.css>) fixes it.

28.06.2022

Continuing with part 6 of the MEAN stack tutorial. Not too much new information, but I learned how to use regex in typescript. I installed the same version of flash messages to avoid any issues. That wasn’t quite enough, as I also had to add “.forRoot()” when importing FlashMessagesService in app.module.ts. I was getting an error saying there was no provider and found the fix from here: <https://stackoverflow.com/questions/51838324/error-nullinjectorerror-no-provider-for-flashmessagesservice>.

After restarting the server, I got a metadata version mismatch error and had to downgrade flash-messages to 1.0.8 and remove the .forRoot() part. I noticed that installing this version was mentioned on the course page afterwards.

30.06.2022

I went through part 8 and learned how to use local storage. This part also served as a good reminder of some of the things that we did in the back-end portion of the tutorial.

From part 9 I learned how to use authentication guards to disable certain routes if a user is not logged in and how to hide the navigation buttons accordingly. I also learned about the ng build command.

02.07.2022

I finished the MEAN stack tutorial and deployed the app to Heroku. In the final tutorial video mLab was used, but that has been migrated to Mongo as Atlas so I watched TraversyMedia’s video on that to continue: <https://youtu.be/KKyag6t98g8>. It was easy to use Atlas instead since the process was very similar. The app is available here: <https://lit-gorge-43048.herokuapp.com/>.

06.07.2022

I started working on the project. After a bit of thinking I decided to build a barebones version of Twitter called Twoiter. It will be based on the MEAN stack example app, but I will use the newest versions of modules and everything so that solving possible problems will be easier. The service will have a home page where you can see all posts or “twoits” made by users with the newest ones being at the top of the page. There will also be a button to send a twoit. Another page will show a user’s profile with all their twoits. I might also add a like system and maybe some fun CSS.

Creating the initial back-end was basically as easy as copying the files from the example project. I just modified the database.js file and defined some options when using express-session to get rid of some warnings. User registering and authenticating was working well.

I also updated Node to the LTS version (16.15.1) and installed Angular-cli 14.0.5. After copying over some of the front-end files from the example project and trying to build the app, there was a mountain of errors to fix as I expected. These required a whole lot of annoying googling and changing things everywhere. I had to use auth0/angular-jwt instead of angular2-jwt, which works slightly differently. I also couldn’t call private properties (authService) in template files so I created a wrapper function that calls the property instead and used that.

Once all the errors in the terminal were gone and the app compiled, it was time to mend some errors that showed up in the browser console. These were mostly caused by auth0/angular-jwt and I got them fixed by adding providers in app.module.ts. I also had to follow instructions I found from here: <https://github.com/auth0/angular2-jwt/issues/476> and modified them to fit my app.

Now I had no more errors, however the CSS was not working. This was because my index.html file didn’t have the stylesheet specified. After adding that, the site looked like the example project. Except it wasn’t connecting to the back-end, because I forgot that when deploying the example app to Heroku I had to change the routes in AuthService to not have http://localhost in the beginning. I changed those back and everything was finally okay.

Almost regretted updating everything for a second there…

09.07.2022

I began adding posts into the project. Soon after I realized that the profile page was not working. After a few hours of poking around and debugging to find the root cause, I figured out that the request headers were empty. This was happening due to the fact that the new HttpHeaders objects work differently, and using append() to add values didn’t work. Instead, setting the values in the constructor did the trick.

Another bug I came across was that the app didn’t see the user as logged in if the page is refreshed, even though the local storage still has the info. This was happening because the token was not being loaded, so I made it do that in the app component OnInit function. Now everything (actually) worked.

11.07.2022

I added the routes and functionality of creating and getting posts from the database. This was pretty simple to do, I referenced the code for registering users and getting profiles, as well as the Tour of Heroes files for displaying all of the posts in a list. To display timestamps for posts in a readable format I looked up how to format them: <https://stackoverflow.com/questions/9532664/how-to-display-a-date-object-in-a-specific-format-using-javascript>.

12.07.2022

I added a component for creating posts. Once again the Tour of Heroes tutorial came in handy. Getting the logged in user’s id felt a little tricky, in the end I settled on using the AuthService’s getProfile function.

Once I got the component working, I noticed two issues: the list of posts in the home component doesn’t get updated instantly and the posts are not sorted by date. I fixed the latter issue first by using JavaScript’s sort function after getting posts from the database. The updating problem was simple enough to fix by using a BehaviorSubject as explained here: <https://stackoverflow.com/questions/46047854/how-to-update-a-component-without-refreshing-full-page-angular>.

13.07.2022

The next feature I wanted to add was showing the posters’ usernames next to the posts, and since I want username changes to be possible, the posts only contain the poster IDs. This meant that I had to add a way to get usernames by ID, so I created a route for that which doesn’t need authentication. I learned that trying to call a function that subscribes to an observable inside an ngFor loop is a bad idea. It created an endless loop of requests that froze and crashed my browser. I decided to try adding both the formatted timestamp and the poster’s username to all post objects in a forEach loop after getting them in the getPosts method. This worked well.

I started working on the site’s theme. First I switched out the Sandstone theme that was used in the tutorial and replaced it with one called ‘Journal’. The examples of elements on the Bootswatch site were very helpful in getting everything to look like how I wanted, and I learned about things like input addons. I used panels for displaying the posts. I also made the navbar’s position fixed when scrolling. Another thing I learned was that you can put autocomplete=”off” into an input element to turn off suggestions, which was useful for the post content input.

15.07.2022

Today’s feature to implement was pagination for posts. I didn’t want the app to load every single post immediately as that would be slow and take a lot of memory if there were a lot of posts. Instead, it should only load around 10 posts and have buttons for seeing older or newer posts.

I started by adding a new function to the Post module that loads a specific page of posts by using the Mongoose model’s sort, skip and limit functions. Next, I added optional query parameters to the ‘get posts’ route that specify a page number and page size and added those parameters to the PostService’s getPosts function. To know which page should be loaded, I added some variables and functions to the home component that keep track.

For the ‘older’ and ‘newer’ buttons I referenced the Bootswatch page. To disable them when the user is on the newest or oldest page, I used the NgClass directive. At first, their OnClick event would still fire even if they had the ‘disabled’ class. To prevent that, I used the CSS attribute ‘pointer-events: none’. I found this solution from here: <https://stackoverflow.com/questions/13955667/disabled-href-tag>. The buttons can still be triggered by using Tab+Enter, but the function logic has checks to not go over the page limits so it’s fine.