Lappeenrannan teknillinen yliopisto

School of Business and Management

Sofware Development Skills

<Insert name here>, <Insert student number here>

LEARNING DIARY, <CHOSEN MODULE NAME> MODULE

**LEARNING DIARY**

21.01.2024

* Learned that Heroku is an app posting platform.
* Learned that Node.JS is not good for CPU intensive server logic, as event loop will be blocked.
* I was already familiar with basic Node.js things including path, fs, etc. so breezed through the video there.
* Learned that Node.js has built-in EventEmitter class and logic. Doesn’t seem too interesting but could save some reinventing of the wheel in simple applications.
* Refreshed memory on using http module directly in Node.JS, and writing local HTML file as response.
* Learned that path module has handy extname function.
* Followed the simple http server example, and it was a little bit interesting, different viewpoint. But I felt the example felt short, since he was trying to show how the file server now can serve CSS files for example, but actually that part of the demonstration worked even without the code added for serving files.
* Followed the Heroku onboarding until met with a pay-wall – in order to create Heroku apps you need to add a credit card payment method to Heroku. I don’t know if this was different in the past, but I stopped here because I don’t want to add my billing information just for a small step of a school course. In any case, I watched the video tutorial to the end and learned enough of Heroku in my opinion.

04.02.2024

* Compared to traditional table databases, mongodb works with collections of documents, similar to JSON objects. “BSON”. Thus, popular with JS development.
* Scales horizontally, add cheap machines to build cluster, performant and fast.
* Flexible, data structure is very much like JavaScript objects basically. Don’t have to map entire data structure (columns, data types) beforehand like relational databases. None of that with Mongodb, freedom to structure data however you want.
* Certain use cases are better for Mongodb, others for relational databases.
* Local installation setup. Mongodb server, compass, …
* Mongodb atlas (cloud instead of local setup).
* Created account, test database with M0 deployment, installed Node.js driver and connected to cloud deployment using local IP address and username + password.
* Continuing to Express crash course.
* Middleware = server-side logic that can edit request and response objects.
* Followed addition of hosting public folder as static assets (html,css,etc.).
* Followed addition of simple GET endpoint, tested with Postman.
* Followed addition of simple custom logger middleware.
* Followed addition of end point with ID query parameter.
* Followed example of using express Router to manage end point routes.
* Followed addition of POST end point and express built-in JSON parsing middleware.
* Most apps usually have either React/other framework based frontend which is served as JSON assets, OR full server side rendered UI.
* Followed through handlebars guide, showing how to render HTML content in server side with express and handlebars. “Template engine”
* In proper apps with Vue, React, etc. would need JSON web token authentication back and forth. Passport can be used to help not reinvent wheel here “passport-jwt”.
* Continuing to Angular.
* For single-page apps development.
* Angular-js, or angular-1 (legacy), completely different. Angular 2 is a separate framework.
* Dynamic frontend apps, full featured more like React/Vue, more stuff built-in like Router, HTTP, dependency injection. Steeper learning curve than React,Vue. TypeScript by default. Uses RxJS observables by default for async programming. Friendly for test and end-to-end testing. Used a lot in large enterprise business. More strict and standardized than React.
* Component driven. Component = HTML output + styling. Can be embedded as XML-like tags. Class based with life cycle methods.
* TaskTracker: header + close button + add task form + tasks + task + footer. Using angular router at end.
* Concept of service to increase modularity and reusability. Rather than all functionality in components. eg. input validation, data fetching.
* Angular CLI to run local dev server, creating apps, generate components, services, etc.
* Angular Material library available.
* Created new angular application using CLI.
* At 27:25 using some changed [ngStyle] syntax. Didn’t find any solution for this.
* Font awesome install command did not work.
* Forms syntax was also very difficult to follow as syntax has HEAVILY changed since the video.

18.02.2024

* Continuing to MEAN stack video.
* Authentication using JWT access tokens, implemented using Passport library for convenience.
* CORS middleware for security.
* Mongodb database with Mongoose mapping.
* Angular 2 frontend. Angular CLI is used to generate components etc.
  + Will be compiled directly to express public folder, so frontend build is hosted using express.
* Angular2 JWT is used to handle authentication tokens, check that they are valid. Stored in local storage.
* Video will cover deployment to digital ocean or Heroku.
* Alternatively, the option to use mean stack generators etc. but the from scratch video is better for learning.
* Guards prevent access to specific pages without necessary app state, such as auth.
* Services are used to connect frontend to backend.
* 7 years old tutorial video – yikes… What could go wrong?
* Following video but using Promises and async functions instead of callbacks, feels like more modern practice.
* Using Visual Studio Code HTTP client to test endpoints.
* Got issues with connection to mongodb with mongoose. Had to change “localhost” to 127.0.0.1 to get it working.
* Had to add express-session middleware to work.
* Had to add toJSON() before passing User object to jsonwebtoken sign method.
* Newer passport-jwt strategy was a bit different, had to use “Bearer <token>” instead of “JWT <token>”.
* Contrary to video, before route links work, had to add imports: RouterModule to navbar component. Very time consuming to find these differences.
* Angular forms also have heavy changes, very time consuming to find basic form connection working.
* Angular 2 flash messages seem to be some totally outdated library not maintained or usable anymore with latest angular versions. Skipped this part.

30.03.2024

* Continuing MEAN stack video part 7.
* Got a lot of issues from the following videos being 7 years old and got the feeling I’m wasting my time to learn techniques that are not recommended or not functional anymore. As such, decided to learn the last few parts of the video from scratch using the up-to-date Angular documentation.