Lahti teknillinen yliopisto

School of Software Engineer

Sofware Development Skills

<Hongyue Zhang>, <001219638>

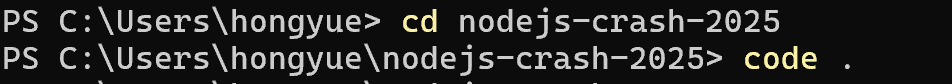
LEARNING DIARY, <FULL-BACK> MODULE

**LEARNING DIARY**

20.03.2025

I learned what is Node.js and how it works. Node is built on top of the V8 JS Engine and extened the engine to work on the server side. Node is also called non-blocking which means that it doesn’y wait around for to operations.Node also uses event loop which is a mechanism that allows node to perform non-blocking IO operations. During watching the video, I unsderstood what is Node.js used for like APIs, Microservices, and etc.And Node is not good a t CPU-intensive tasks.

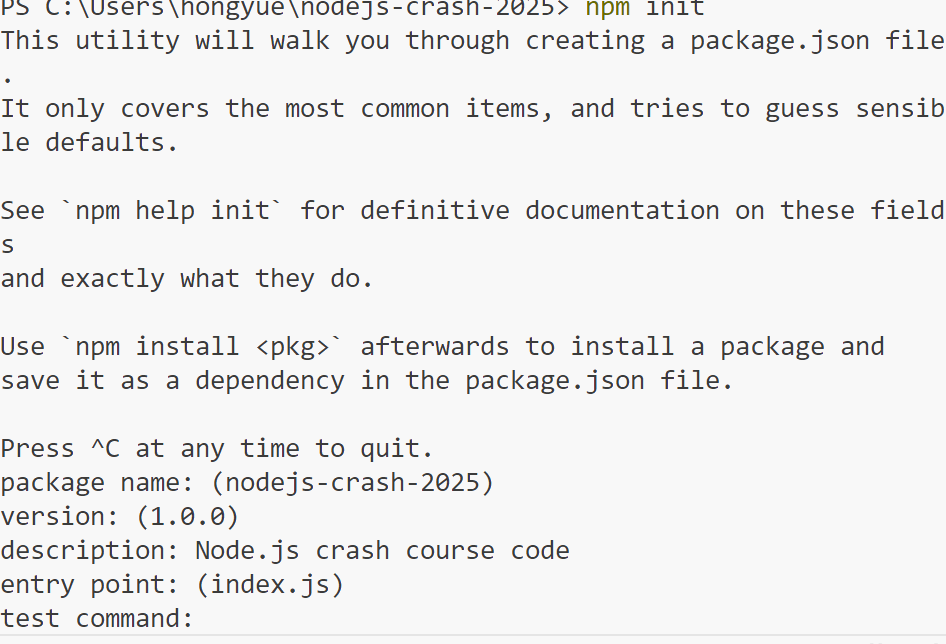
I learned also how to create a node project under command line.



Run PowerShell as an administrator and run the following command in an administrator PowerShell window

Set-ExecutionPolicy RemoteSigned -Scope CurrentUser

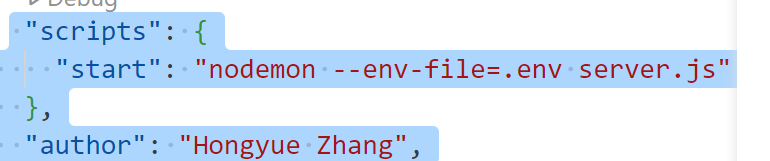
Then I can run the terminal



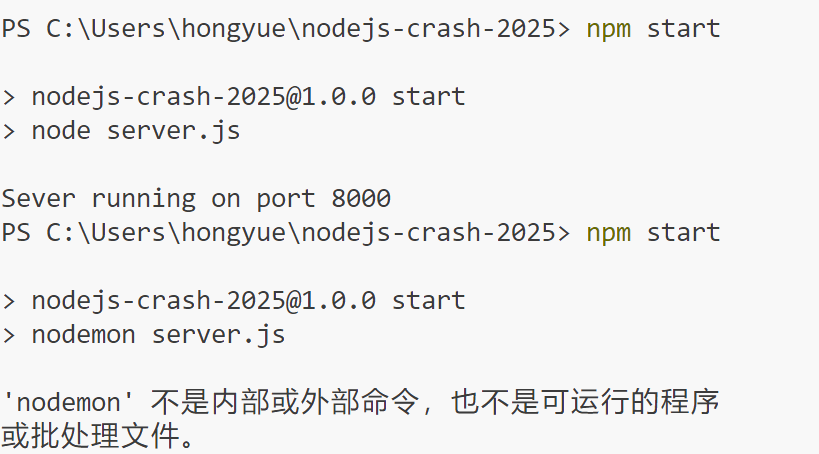
22.3.2025

I learned some basic code about the Node.js. Like, how to export functions.And learned how to create a basic HTTP server, read files asynchronously, and how to handle and return a 404 status when it is not a GET request.And during the development process.

When I chage the json file ,like this



Then it shows:



So I installed the nodemon tool to automatically restart the server.

Then run the command:

npm start

The server can run automayically.

Then I installed the postman. Used to test HTTP requests. I use postman to send a get request to get data, and submit the data by using post method.

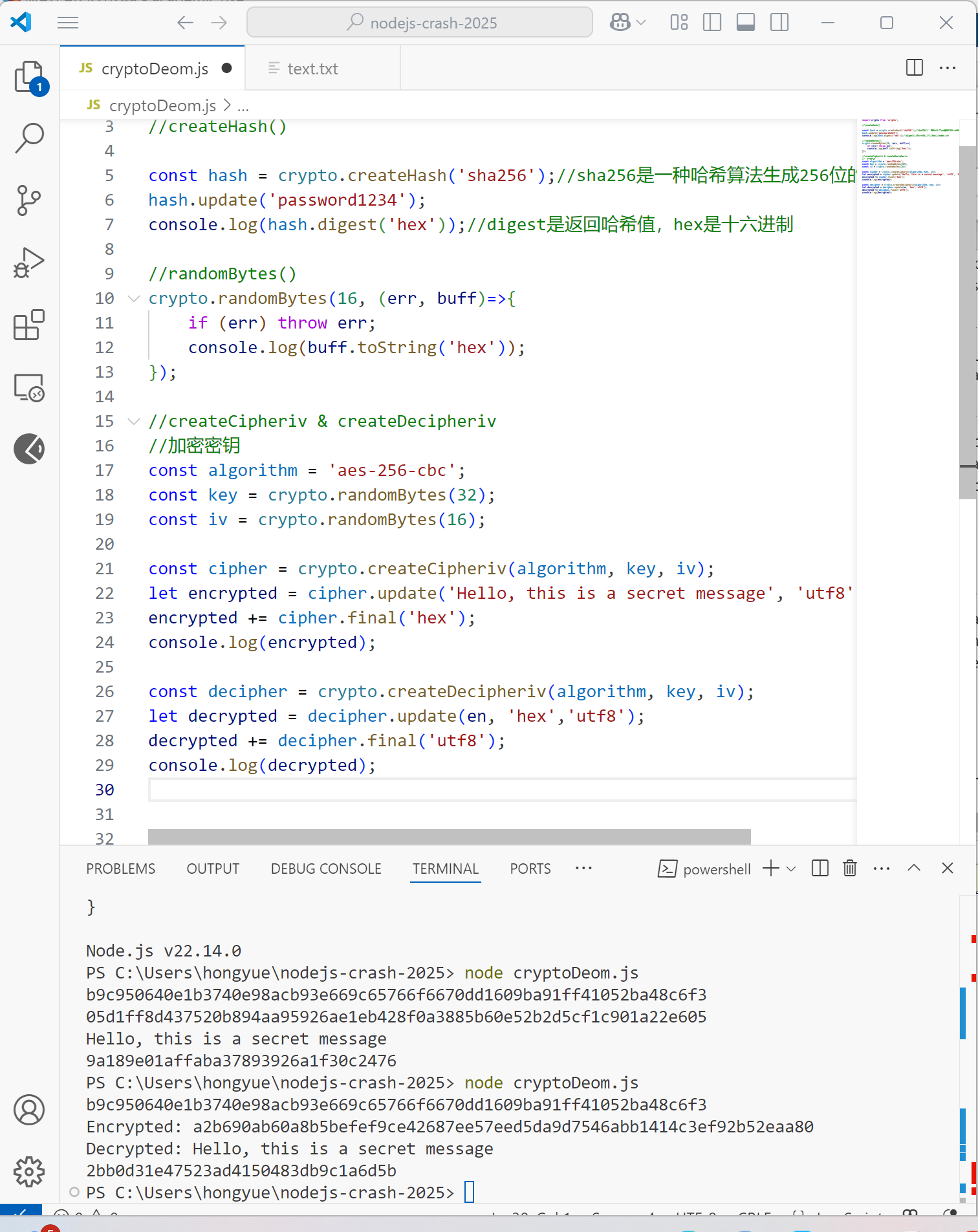
24.03.2025

Today I learned how to handle /api /users and find the user id in the user array that matches the extracted ID.

I also learned about HTTP servers and their middleware. For example, I created a middleware to record the method and url of each request to control the server. I also learned how to handle POST requests. I can use req.on('data') to receive data, and then use req.on('end') to process the data after receiving it, and add the new user data to the user array.

In addition, I also learned the URL, fs, and os modules. In the url module, I understood how to process URL strings, parse query strings in urls, and convert file URLs into paths, etc.

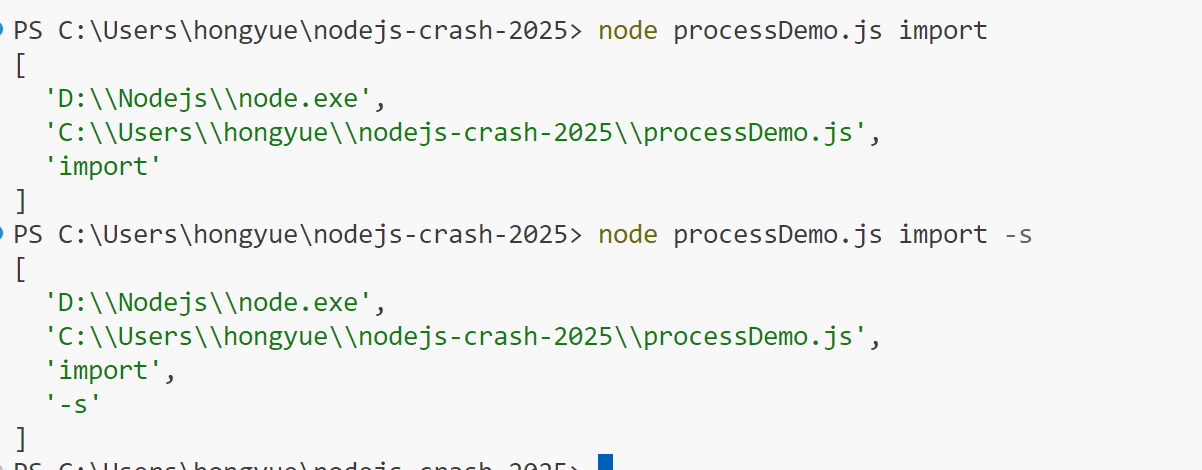
But when I learned crypto module, I have some troubles,like:



The last line of the output should be the decrypted message, but it showed an encrypted value.

I fixed the issue by rewatching the video again. This is because randomByte will randomly generate a 16-byte string, which has nothing to do with the encryption and decryption method, just delete it.

Use process.argv to get command line parameters such as import, -s, etc.



25.03.2025

Today I learned something about MongDB. MongoDB can use sql which stands for structured query. MongoDB is actually a document database that data is stored in bsan under the hood. Today, I first learned how to download mongodb shell, mongodv compass and mongobd in vs code. I also learned how to use mongodb to aggregate my database. For example, using mongodb compass, I can aggregate the Airbnb homestays with a monthly rent of less than or equal to (lts) 500 in the json data set.

Then I also learned how to use vscode to view my aggregated data set.

27.03.2025

I learned the concept of express. It is a simple and flexible node.js web framework that provides many features to help me create various web applications. Today I learned how to use express to process request and response data using callback function parameters: request and response objects. I learned a lot of different requests such as get, post, put, etc. I also wrote about how to send JSON responses and how to use postman to view relevant request data.



01.04.2025

Today I continue to learn express. Today I learned how to create an error handler to catch errors in the application and return a normal response. I can customize the format of the error middleware and use next to pass the error to the middleware.

In addition, I learned how to use HTML forms or fetch methods to send requests to the backend API. Then, I installed EJS and used res.render to pass data to the view.

05.04.2025

Today I learned React and read the React documentation. Through this part of learning, I have a basic understanding of React, such as understanding the basic concepts of React, which is a JavaScript library for building user interfaces. I learned how to define different react components and add styles to components (usually using the classname attribute). Then I also rendered according to different drop-in statements (if ?:, &&). Then, I learned some response events, such as onClick, and I also tried to use state to update the page. Whenever the state is updated, the component will automatically re-render and show the new content.

07.04.2025

Today I watched a video about React. I first learned how to create a React project.

npx create-react-app task-tracker

Using this command line, you can create a new React project, which contains all the configurations needed for development. Then, I learned to create components in a functional way, and different components are type-constrained through the prop-type package. This is also the first time I learned about React's non-mandatory type constraints. Then I learned three ways to add styles: such as external CSS files, inline styles, etc. Event handling functions are usually not called directly but passed into functions for reference.

08.04.2025

Today, according to the requirements of the video, I used the map method to render the task array into multiple subcomponents. Each task requires a unique key value. Then I learned in depth how to use different operators for conditional rendering to display different UI states. Form controls (such as input boxes) need to be bound to the React state. By clicking a button, the user's input content is added to the task list and the form is cleared. I also learned how to switch the section status (show or hide) by clicking a button. Finally, by running npm run build, a project can be generated and deployed to the server.

Then I used json-server to quickly create a local API and used useEffect and fetch to get the task list.

12.04.2025

Today I learned the remaining two videos of the mern project, such as how to use React and Redux to build a complete user authentication system. I started to create page components such as registration page, login page and home page, and each page was placed in the pages folder. I used App.js as the main routing container. I set up a header.js to display the navigation menu including links to the home page, login page and registration page. Then I started to implement the rendering of the registration page, such as using useSate to manage user input (name, email, password), etc. I also learned to install and configure Redux.

15.04.2025

Today I started to try to build my own project based on the original project. I want to keep the login and registration pages unchanged, create a shopping list in the dashboard, and users can add the items they need to buy in the shopping list. This is a very simple project.

18.04.2025

Today I re-watched the YouTube video based on the example code. In fact, the steps to create a shopping list are the same as the steps to create a goal in the example code, except that you need to add post or put requests to delete and add items in shoppingListController. I followed the goalController and completed the following request.

|  |  |  |
| --- | --- | --- |
| Function | HTTP request | URL |
| get all shopping lisrs for logged-in user | GET | http://localhost:5000/api/lists |
| Set a new shopping List | POST | http://localhost:5000/api/lists |
| Delete list | DELETE | http://localhost:5000/api/lists/:id |
| Add an item to a shopping list | POST | http://localhost:5000/api/lists/:id/items |
| Remove an item from a shopping list | DELETE | http://localhost:5000/api/lists/:listId/items/:itemId |

23.04.2025

Today I wrote some frontend code, such as creating listService, which is basically the same as goalService. This module uses axios to send requests to the backend to create, obtain, add and delete shopping lists or items. For example, the createList function sends POST /api/lists/ to the backend, names the list, and then returns the data of the created shopping list.

Then create a Redux slice and name it lists to manage the state of the entire shopping list server, including the loading state and all shopping list data. When the dispatch(addNewItem({ listId, name })) statement is triggered, the system will automatically execute the axios request in createAsyncThunk. When the backend returns the request, it will update the current state with the new data in extraReducers. At this time, the page will automatically respond to the state change and update the page.

01.05.2025

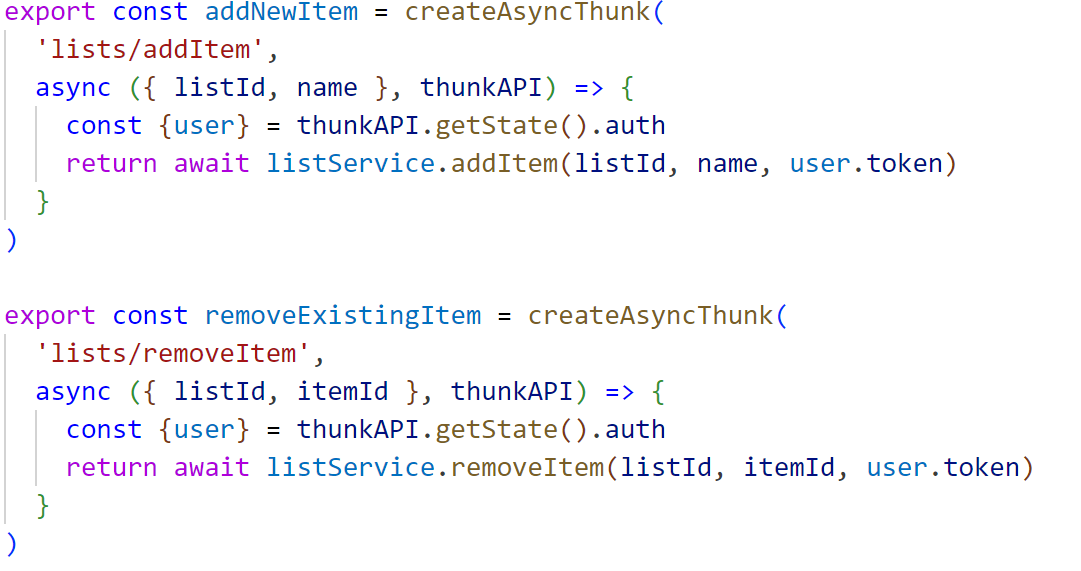
Today I mainly wrote a React component, which is mainly about creating a new shopping list. The user can fill in the name of the shopping list in the input box, and then click the "Add List" button to complete the creation of the list. Usually useState is used to manage the state of the input box. When the button is clicked, it will automatically respond to the function onSubmit, which will flip out the Redux asynchronous action to add the request.

09.05.2025

A few days ago, I didn't finish the project because I was preparing for an exam. Today, I plan to write item.jsx and ItemForm.jsx. Item is mainly used to display each item. ItemForm is mainly used to add an input box for items. This component is basically the same as listForm. When you click the + button or press Enter, the onSubmit function is triggered and the item is added to the shopping list responded by the backend. After submission, the input box will be automatically cleared.

11.05.2025

Today I completed the frontend but encountered some minor problems, such as I could not send requests to the backend. Later I learned that each token has a corresponding user, and I should find the token of the corresponding user first before sending a request to the user. As shown in the figure below:



15.05.2025

Today, when I was using Postman to send a POST request to add items to a shopping list on the backend, I found that users could not post the key value "checked" to the created shopping list.

So, I kept checking the code and found that I didn't define the return value of checked as boolean, so I added the relevant function. At this time, I could send the request correctly.

17.05.2025

Today I tried to run a file in the root directory, but it didn't work. I searched online for tutorials and found that I had to install npm install concurrently --save-dev in the root directory. And I had to add it to the package.json of the root directory.

{

"name": "mern-tutorial",

"version": "1.0.0",

"scripts": {

"server": "nodemon backend/server.js",

"client": "npm start --prefix frontend",

"dev": "concurrently \"npm run server\" \"npm run client\""

},

"devDependencies": {

"concurrently": "^7.0.0"

}

}

Then I could run npm run dev in the root directory.