Part 2 & Part 3

**Introduction**

I am happy that I started early, because these two units were gigantic, and my rate of work didn't decrease for this module either. Hopefully the future units are more manageable. I think I spent about 15 hours every week for the past three weeks on Part 2 and Part 3 along with all the supplementary material.

| | |
|---|---|
| The phonebook app was deployed to: | https://fast-lowlands-99395.herokuapp.com/ |
| The notes app was deployed to: | https://murmuring-ravine-26738.herokuapp.com |

| | |
|---|---|
| Self Evaluation: | Mastery / Progressing / Needs Improvement |
| Due date: | 2021-09-17 |
| Completion: | 2021-09-14 |
| GitHub: | https://github.com/ZDGharst/FullStackOpen |

**Part 2 – Fundamentals of Web apps**

*Part2a* is about rendering collections and modules. This part re-emphasizes the console log, event handling, and VSCode tools such as snippets and refactoring. An exciting part of this material is the functional programming concepts used in JavaScript: map, find, reduce, filter, and higher-order functions. I learned how to use these functional programming methods to render entire collections (such as taking an array, mapping it to its HTML elements, and rendering the entire set of elements). I also gave each element a unique key, such that when the elements may need to be re-rendered, only that element is re-rendered according to the key for efficiency. I also returned to old practice projects and refactored to have individual modules. The exercises in this module have us create a course app that display course information according to an array, object, arrays of objects, etc. As the array changes, so should the app to update.

Forms are the main focus of *Part2b*. Using state and effects, we can render form elements that change the application. One way to do this is to modify the collections mentioned in *Part2a*, then render those (or do whatever logic must be done). I also used the functional programming method filter to filter the collection to only contain certain text. The exercises for this section has me make a phonebook app: the user types in a name, phone number, and submits the form. The name and phone number is either added to the phonebook as a person object, or an alert is sent to the user if either of those already exist in the collection. The user may also type into a filter box to view persons with that string as part of their name.

Within this part, *Part2c*, I finally start working with a backend data provider. Using a simple flat JSON file, I served it as a database using json-server as a local port on my network. I use my other app to connect to this port and receive data that is served by the app. I learned the difference here between synchronous/asynchronous actions in JavaScript, and how promises, when mixed with effects, can work to achieve good asynchronous design. The exercises in this section return to the phonebook app to make it receive data from the json-server, and I use a public API (restcountries.eu) to create an app that filters countries to observe their data.

*Part2d* is about altering data on servers using conventional RESP API and json-server. The course explains REST terminology, convention, and best practices. Within REST, resources have unique addressed assigned to them using its URL, and all resources are accessed using various HTTP methods.

Within the practice and exercises (phonebook app again) in this section, I use axios to create a service for POST, PUT, DELETE, and GET methods using the backend json-server.

The last section is a shorter section; *Part2d* is about applying CSS to components and handling errors. I apply styles to a notification type component, then handle backend service statuses using that notification component. The types of statuses handled are successful updates or creation, failed to connect to backend service, item not found, etc. The exercises has me do this for the phonebook app, as well.

*Additional notes can be found in the Part 2 README.*

**Part 3 – Introduction to React**

Within this section, *Part3a*, I started by using the http module to create a simple web server that returns "Hello World" regardless of the HTTP request. I then quickly swapped to using Express to handle HTTP routing and request types. I created many npm scripts within the entirety of part3. This took some time, but I then created a REST architecture API: fetching all resources, fetching a single, deleting, posting, and putting. I used both POSTMAN and VSCode Rest Client to test my REST API. I also installed middleware, such as json-express and morgan, to handle json parsing and console logging respectively.

*Part3b* is what I believe a lot of computer science students are missing to become engineers, and that's building and deploying. I had to enable Cross-Origin Resource Sharing in order for the backend to communicate with the frontend on a different port. Then, I deployed the entire API to Heroku, which can be queried as any API should. I also built a production build (rather than a dev mode) of the React frontend app that the backend serves statically, which was then also deployed to Heroku such that now the fullstack application is served from the same Heroku production server. A few other things were necessary, such as writing scripts to automate this process and proxying the frontend on development builds to still work in the dev environment.

I was extremely interested in this next material within *Part3c*. I'm covering NoSQL and MongoDB which isn't taught in any class at UMKC. I've learned how to initialize the database, host it online through Amazon Web Services, create a schema, create and save objects/collections, fetch objects from the database, etc. I also learned how to configure the database within its own module such that

consumers of the database won't see the details of it. The project's controllers were modified to use the database, and error handling for the database was implemented as well which was shifted off into middleware.

*Part3d* deploys the database to the backend, explains chain promising, and then teaches the student how to use linting software. In particular, I have to make the phonebook app pass all errors and warnings within the linter. I also enforce the code style guide within the linter, too. *Part3d* is quite short, but the other parts in Part3 were so lengthy that I more than had my hands full this module.

*Additional notes can be found in the Part 3 README.*

**Deliverables**

| Exercise | Commit / Link | Diff | Notes |
|---|---|---|---|
| 2.1 | 81f1ce5 | diff | screenshot |
| 2.2 | 8e413e7 | diff | screenshot |
| 2.3* | This exercise is a refactor of the sum of exercises within the app to use the reduce functional programming function; I already did this, so no refactor needed. | | |
| 2.4 | 309819a | diff | screenshot |
| 2.5 | 176488f | diff | |
| 2.6 | 7806fd4 | diff | screenshot |
| 2.7 | 1b08605 | diff | screenshot |
| 2.8 | 114c2d6 | diff | screenshot |
| 2.9* | a62347b | diff | screenshot |
| 2.10 | 04d17ed | diff | |
| 2.11 | 8a00d4a | diff | |
| 2.12* | 27db545 | diff | screenshot \| screenshot |
| 2.13* | b6c98c4 | diff | screenshot |
| 2.14* | This was an optional exercise (all exercises with an asterisk are optional) that I skipped because it requires me to sign up and use an API key for a weather service. | | |
| 2.15 | 2eb12be | diff | |
| 2.16 | | | |
| 2.17 | 25e3acc | diff | screenshot |
| 2.18* | a8da861 | diff | |
| 2.19 | c134651 | diff | screenshot |
| 2.20* | f2247e0 | diff | screenshot |
| 3.1 | da26388 | diff | |
| 3.2 | cf47e2b | diff | |
| 3.3 | c10c108 | diff | |
| 3.4 | f5a7415 | diff | |
| 3.5 | 35b3955 | diff | |
| 3.6 | dfd754d | diff | |
| 3.7 | abc9852 | diff | |
| 3.8* | e09f511 | diff | |
| 3.9 | 9c5bab0 | diff | |

| | | | |
|---|---|---|---|
| 3.10 | 993f7b6 | diff | |
| 3.11 | fd4e3a4 | diff | |
| 3.12 | 3083676 | diff | screenshot |
| 3.13 | 4003ca9 | diff | |
| 3.14 | 9a0d55a | diff | |
| 3.15 | 379bff0 | diff | |
| 3.16 | 2666c7a | diff | Included with 3.17 |
| 3.17* | 2666c7a | diff | Included with 3.16 |
| 3.18* | 092766c | diff | |
| 3.19 | 77d77c0 | diff | screenshot |
| 3.20* | 846e776 | diff | screenshot |
| 3.21 | 4b73147 | diff | Commit message mistakenly calls this exercise 3.22 |
| 3.22 | ef9b2cc | diff | |

**Supplementary Material**

What the heck is the event loop anyway? | Philip Roberts | JSConf EU

Learning Functional Programming with JavaScript - Anjana Vakil – JSUnconf

Equality comparisons and sameness – MDN Web Docs

Asynchrony: Under the Hood – Shelley Vohr – JSConf EU

Documents – MongoDB Manual

10 Things I Regret About Node.js – Ryan Dahl – JSConf EU