**Project Origin**

The idea for API was heavily influenced by the following 3 goals and this is how it evolved through the bootcamp.

1. Develop a system that would support my learning and development.
2. Explore the possibility of simplifying search, data exchange, integration & access to data in my API and workflow in general.
3. To generalize my code for reusability and maintenance (write once use many)

In week 3 of the bootcamp we explored classes in javascript and built a scooter app for our project. I created a class called **DSMetaData** **(data structure meta data)**. In subsequent projects, whenever I wrote a function that I thought would be useful in a wide number of scenarios, I would add it to this class.

In wk5 we explored databases, in particular sqlite and used sequelize as an interface between JS & sqlite. We would populate the database from a hard coded array of objects or from a json file.

I also explored the built in node-module fs (file system) and wrote an application to read & parse csv files into a typical JS object format that could also be used to seed the database. The set of functions used in this app were included in the class **DSM,** and that was the beginning of the project dubbed **dsm22-1030-080.**

In wk6 we learned to setup a server using express and in wk7 I had the opportunity to consolidate what I had learned and explored from wk3-6 which was rolled up into the app **dsm22-1030-080** and can be summarized as follows.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| WK | Feature | Key Files | Challenges | Value |
| 3 | Classes & Objects. Class DSM | DSMetaData.js | Learning to use classes in Javascript | makes useful code accessible from a single point, skill & knowledge acquisition by continually developing the class |
| 5 | SQL & Databases  class DSM - parsing csv and seeding database | DSMetaData.js, filesystem.js | Concurrency in JS | makes it easier to integrate data from various sources, skill & knowledge acquisition through consistent & continuous development. |
| 7 | Dynamic Sequelize Model - defined at runtime based on source data. | sql3.js, DSM.js | Steep learning curve, concurrency | sql3.js - capability to dynamically define tables and seed data based on the source file(csv) |
| 8 | A backend with a server that returns data and generates dynamic base/root endpoints, based on dynamically seeded database | server.js, sql3.js, DSM.js, deleteTable.js | Steep learning curve, concurrency | automates part of the development of the backend, useful for quickly generating app prototypes, testing, or as re-usable component/API in another app, skill & knowledge acquisition by continually developing the class |
| 10-12 | An API that generates a backend dynamically based on source data and supports CRUD | server.js, seedDB.js, DSM.js, deleteTable.js | Generalization, testing & debugging, concurrency | capability to perform CRUD, skill & knowledge acquisition |

Post Boot Camp

Here is the project spec once again, at this point in time focus was directed on extending to data sources over a network & the range of source files, generalization, refactoring, testing and debugging, developing a front end for the API and most importantly developing a system for embedding new knowledge and skills which I call feature drills.

They are simple coding challenges or exercises, that are performed/drilled repeatedly (may use spaced repetition) and incorporate any elements being learned. The output from the exercise is a feature (mini app), the feature or exercise can be varied so as to develop a more robust view and understanding.

By connecting the drills and the output, I was able to use the same approach described earlier to bootstrap my learning process. The drill supports learning and embedding the skill. The output results in an application that creates a knowledge base and an indexed point of reference which supports learning and so on.

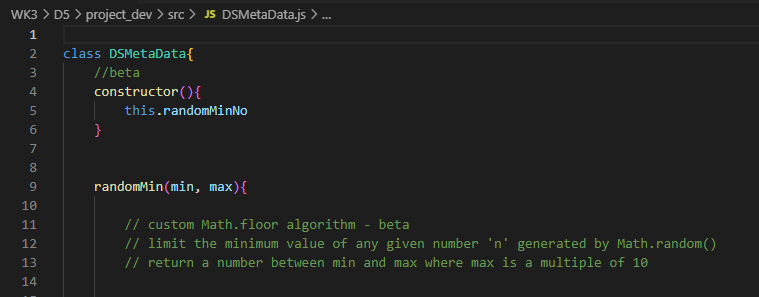
Project Spec

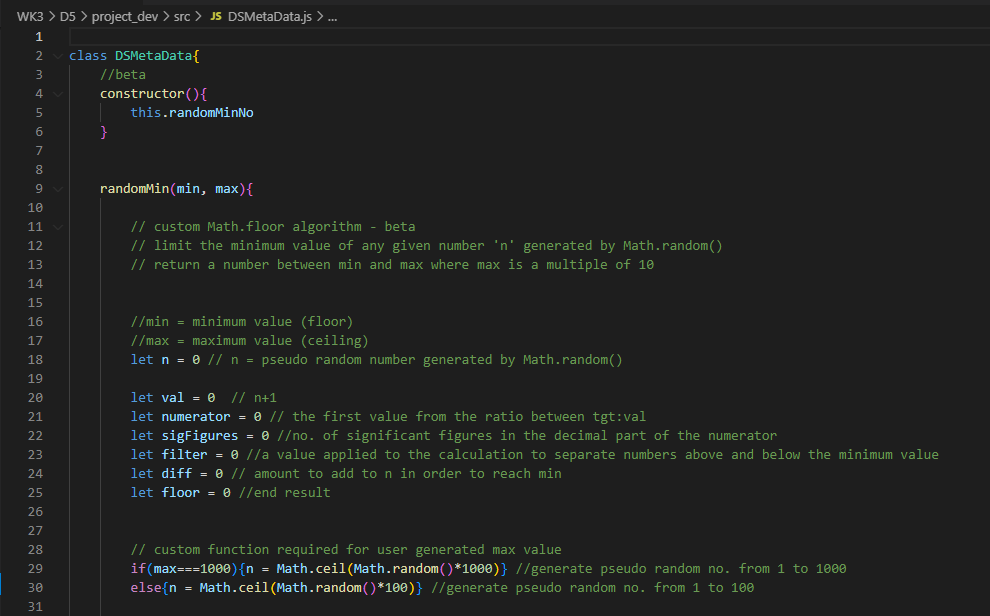
1. The API will fetch data from various sources/files (stored on the local machine or online).
2. The data will be available to interrogate and allow the user to perform CRUD operations (using apps like postman, thunderclient etc).
3. Once the data is read from the source, it can be copied or written to another file or database.
4. A simple front end for the app could render the data & allow the user to interact via a web page.
5. The API may allow for scrapping of data from various sources and transforming or parsing the data into various formats.
6. The components of the API, methods/functions, scripts and the structure at large will be generalized where possible for reusability and maintenance.
7. The app may eventually allow for integration with other tools or may be used as a component in another application.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| WK | Feature | Key Files | Challenges | Value |
| 13 - 22 | feature drills, extending range of data source and source files, front end of API, debugging & refactoring dsm22-1030-080, generalization, version control and backup. | /4-featuredrills/ | Concurrency in JS!, CORS, generalization can be time consuming, learning curve, testing & debugging, development for two environments, node and browser, browser versions of some of the dependencies. | -  Entire apprenticeship code base is backed up with version control.  - An indexed an accessible log of development journey.  - progress towards simplifying, debugging and generalizing components of the API - dsm22-1030-080  - An effective learning aid or process. Reduce some of my gaps in knowledge and understanding of (Concurrency, the FS node module & reading files including tested and generalized script with 21 read functions from fs, fse and stream, CORS, the web & web design, HTML & JS DOM)  By further exposure, improved my understanding of web file readers/api’s (fetch, xhr, indexeddb, webworker etc), conversion between response types, file formats and parsing data (csv, json, xml, xlsx, pdf etc), npm and npm libraries, cdn’s, project configuration (package.json) browser js vs node js (modules, importing and exporting components including classes in both environments), module bundlers such as browserify and webpack, speech recognition and speech synthesis, color schemes.  - a front end with mostly generalized components  - whilst I may have successfully developed various features and components that deliver against goals and project spec & I am satisfied with some of these results, not everything was delivered on time and in some cases the learning curve & challenges were severely underestimated. As my understanding grows, my view on problems/solutions will become more refined. I also plan to review my approach to testing and design will continue to generalize where possible |

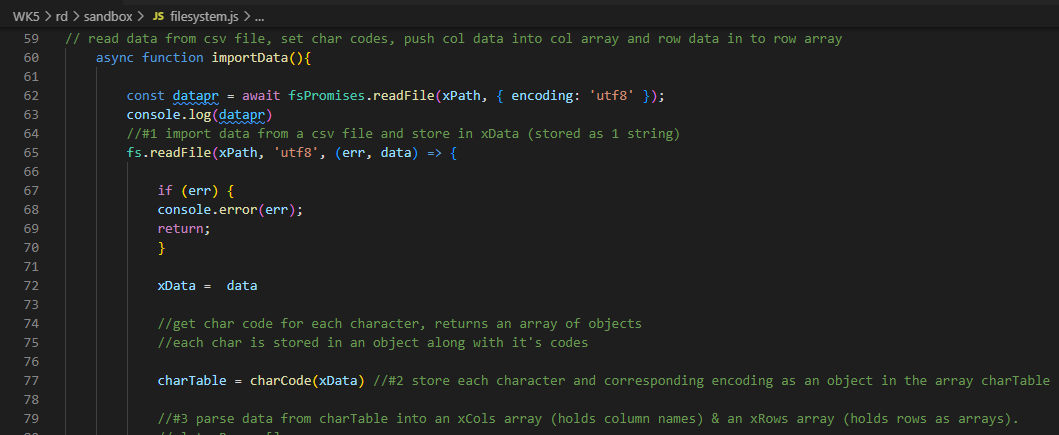
Project Origin (Images)

WK3

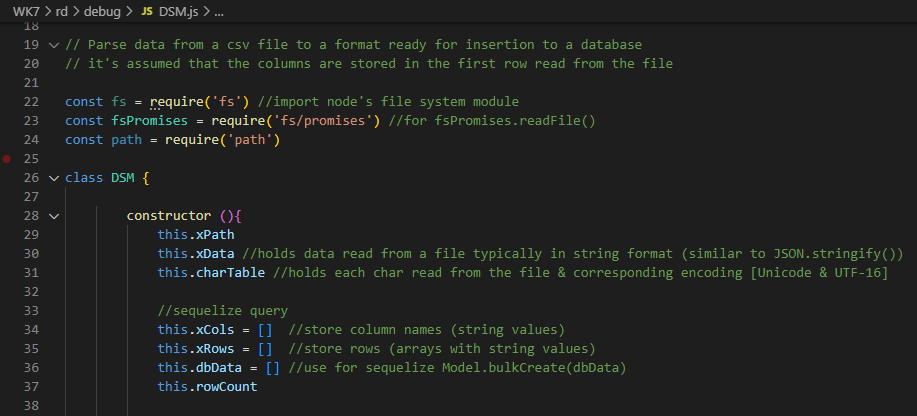


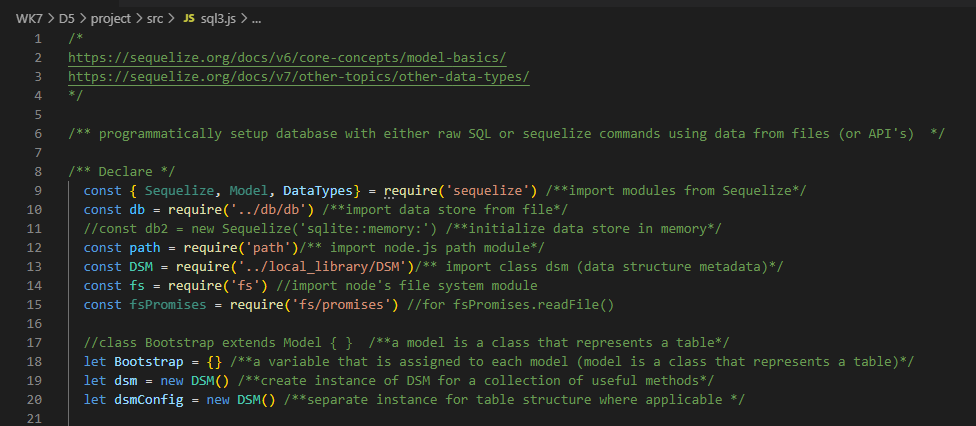


WK5



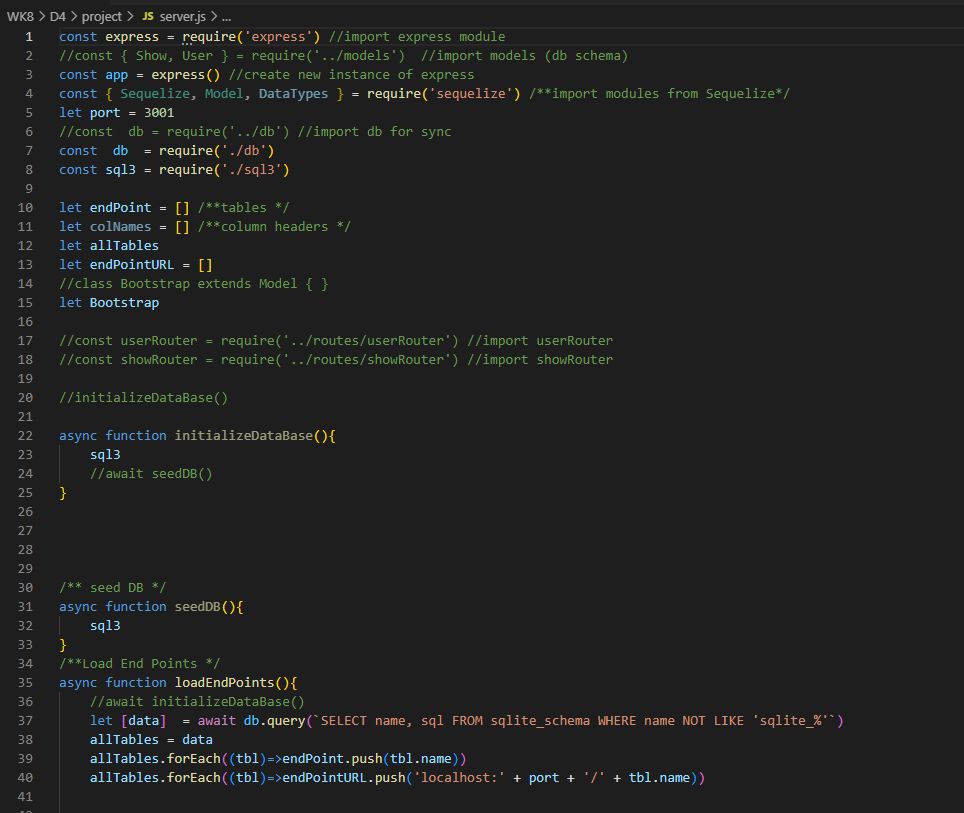
WK7



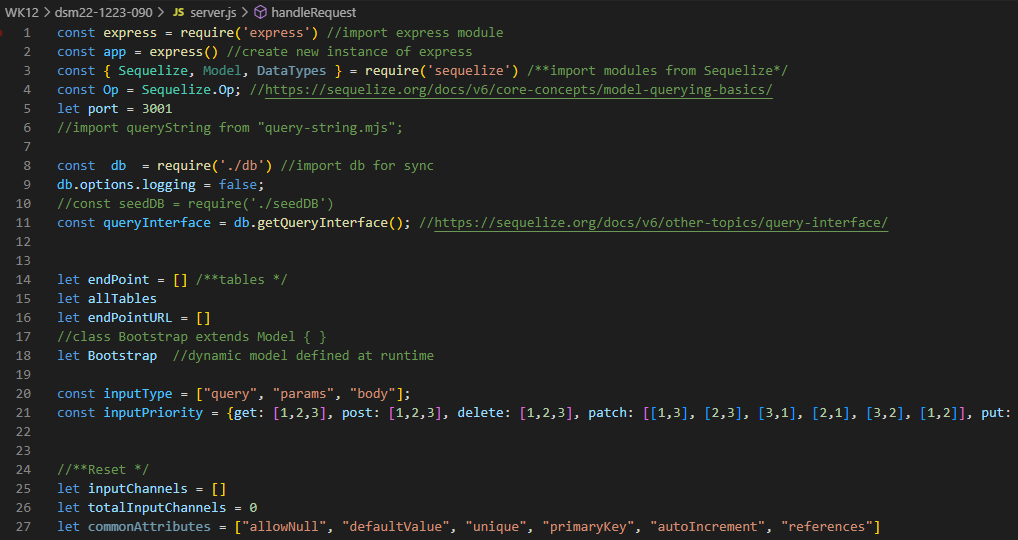


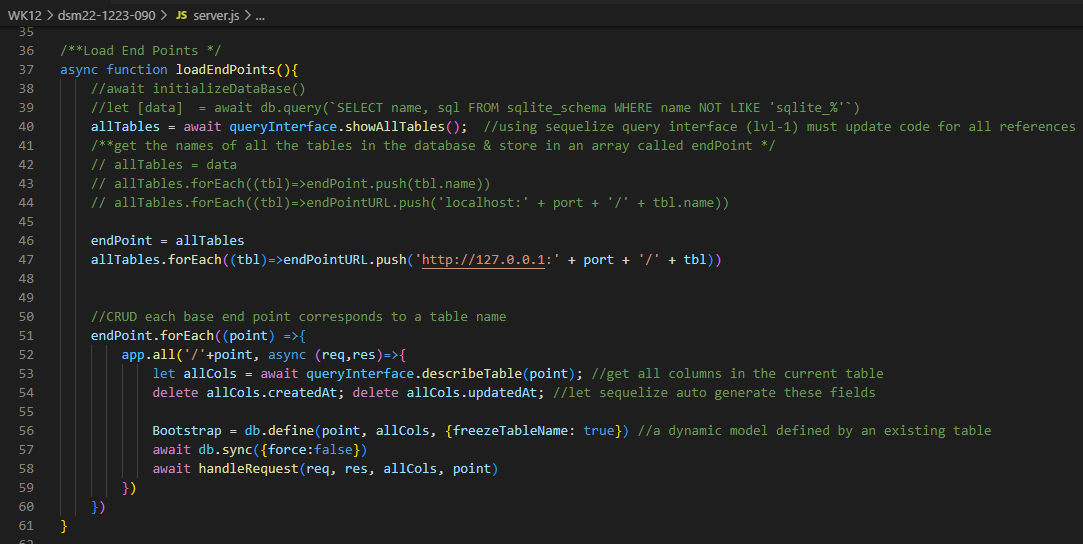
WK8



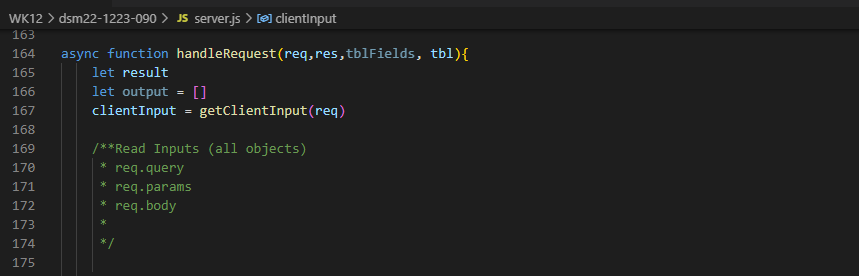


WK12



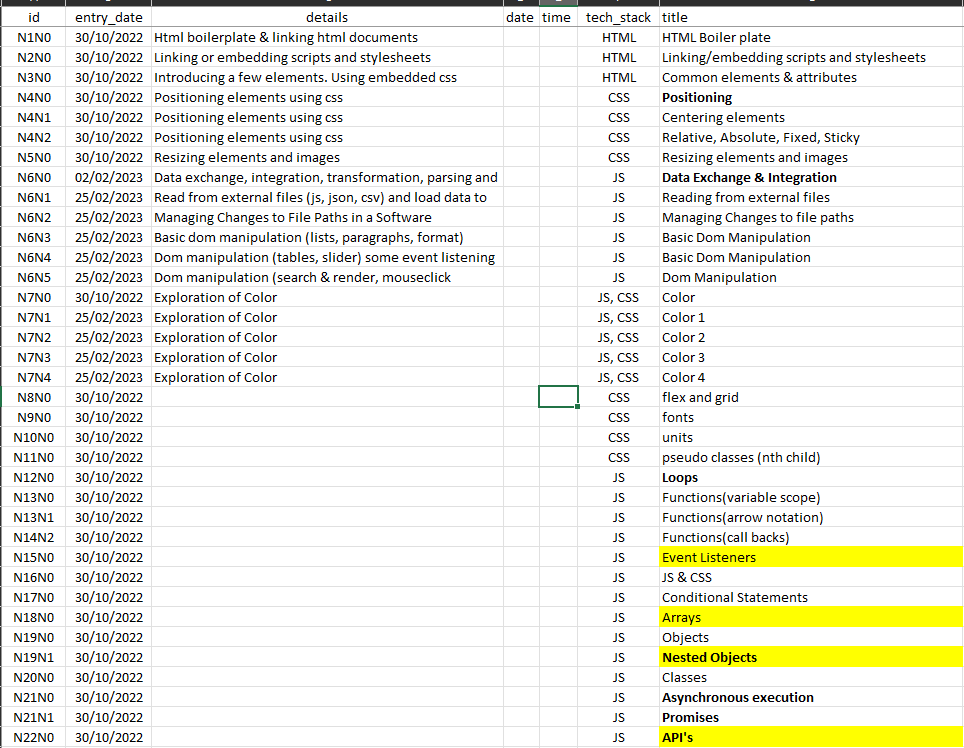


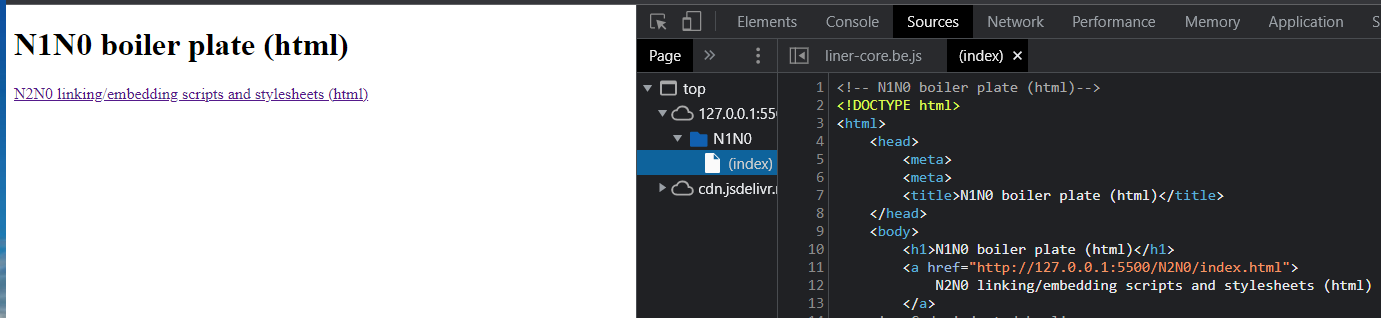
A single function (handleRequest) to handle CRUD operations via restful endpoints.

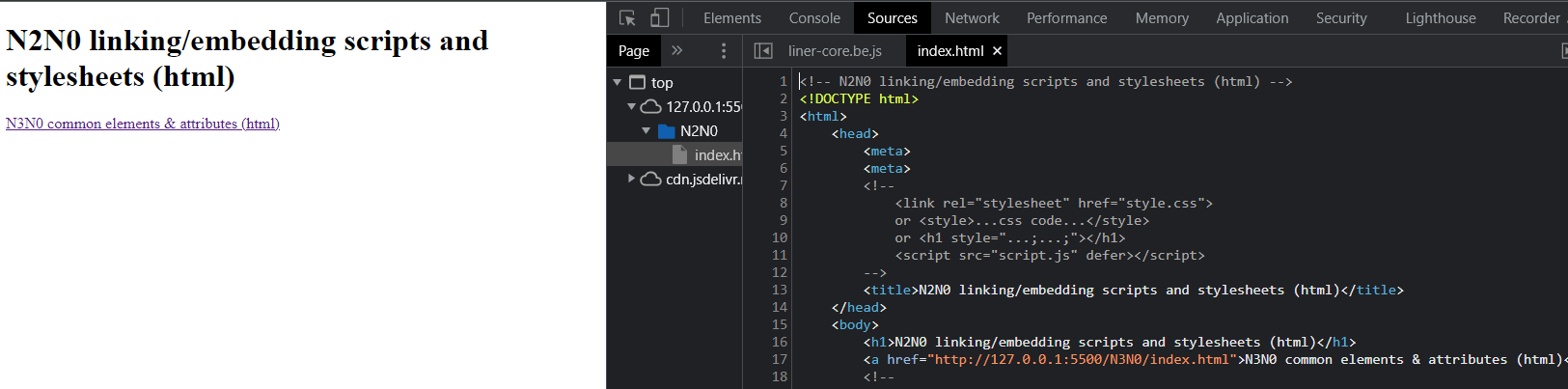


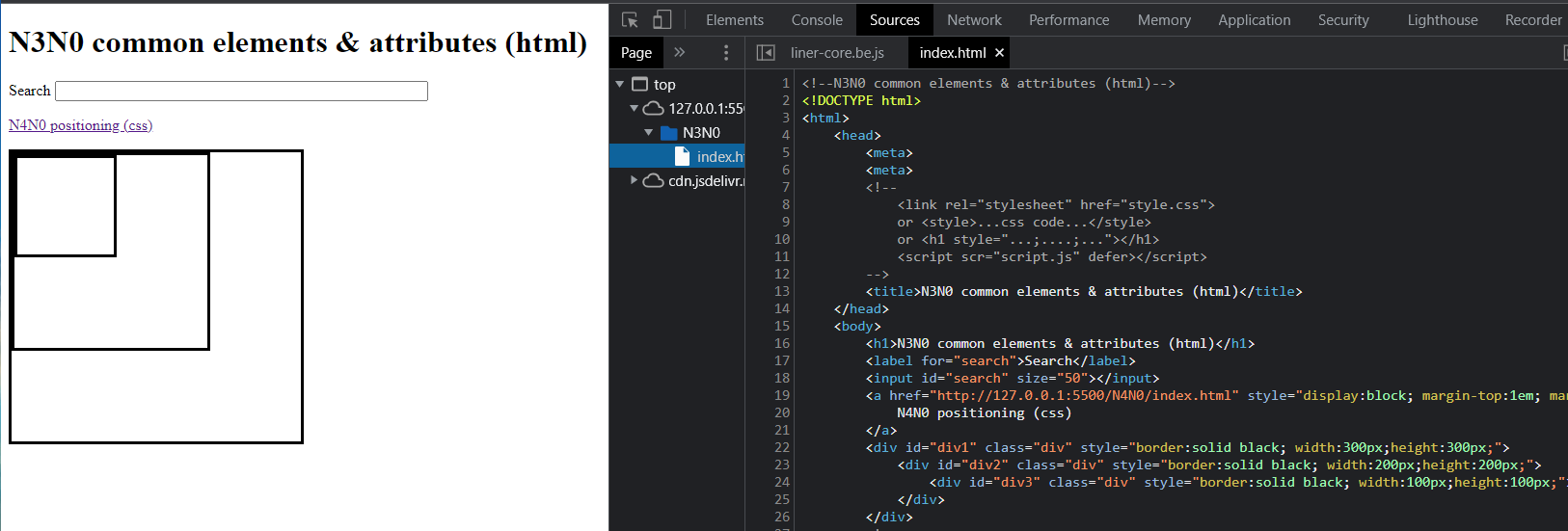
WK13

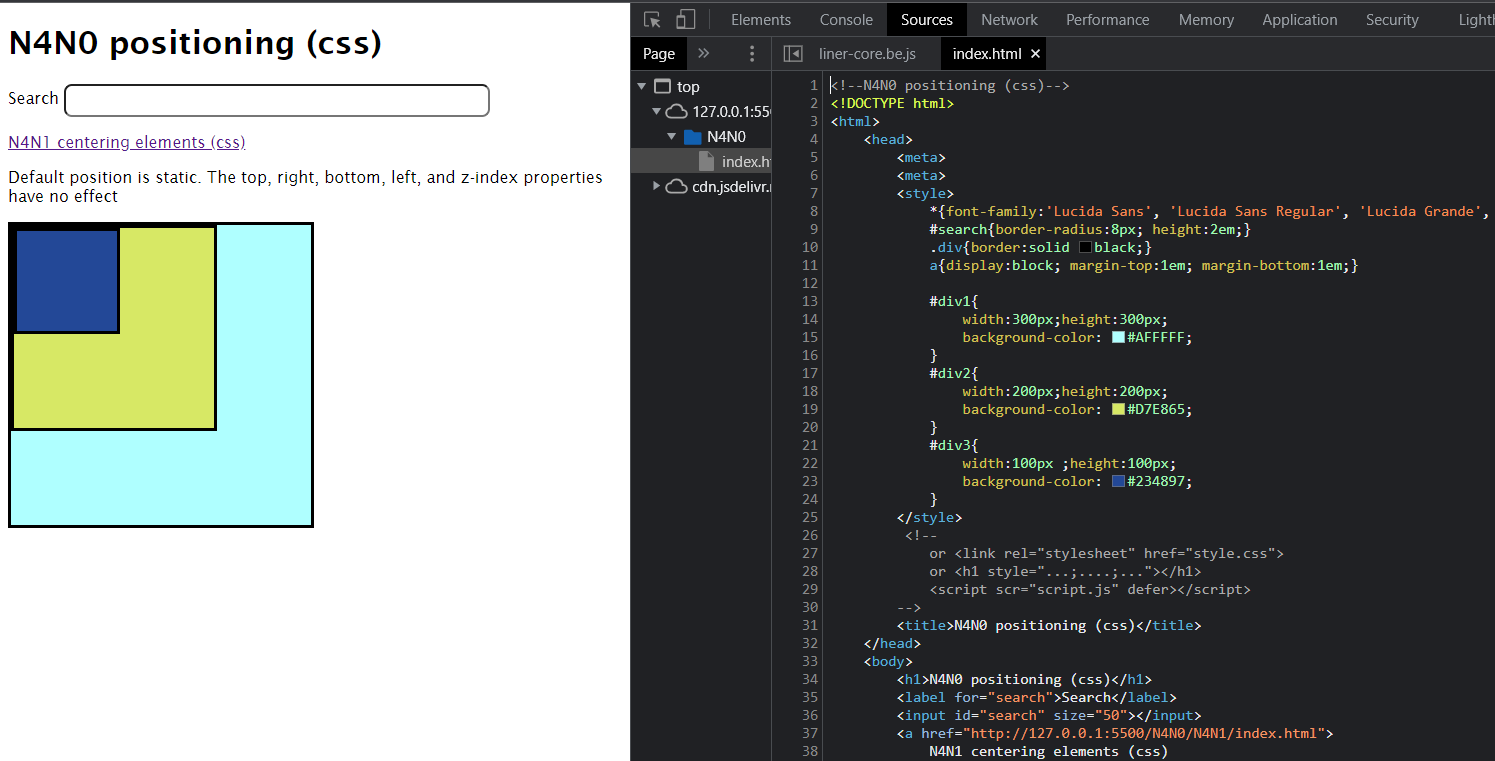
Feature Drills & The beginning of the API Front End

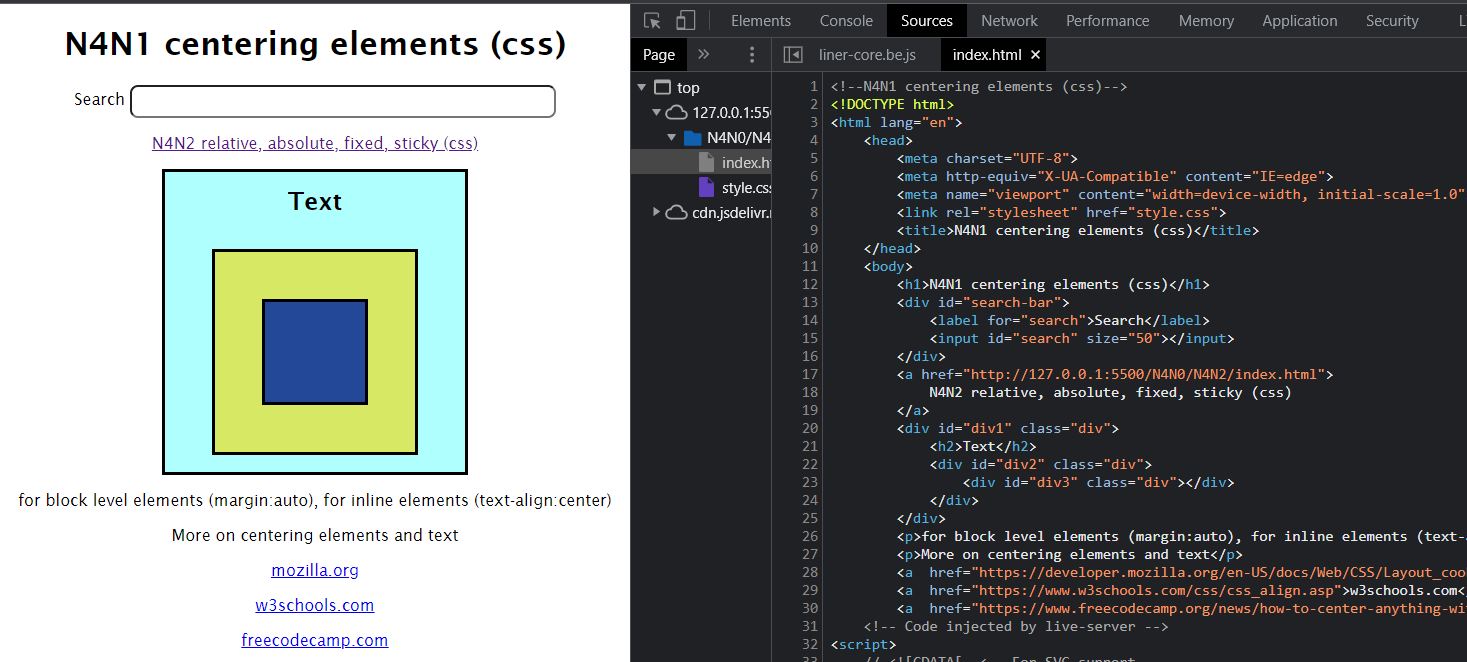


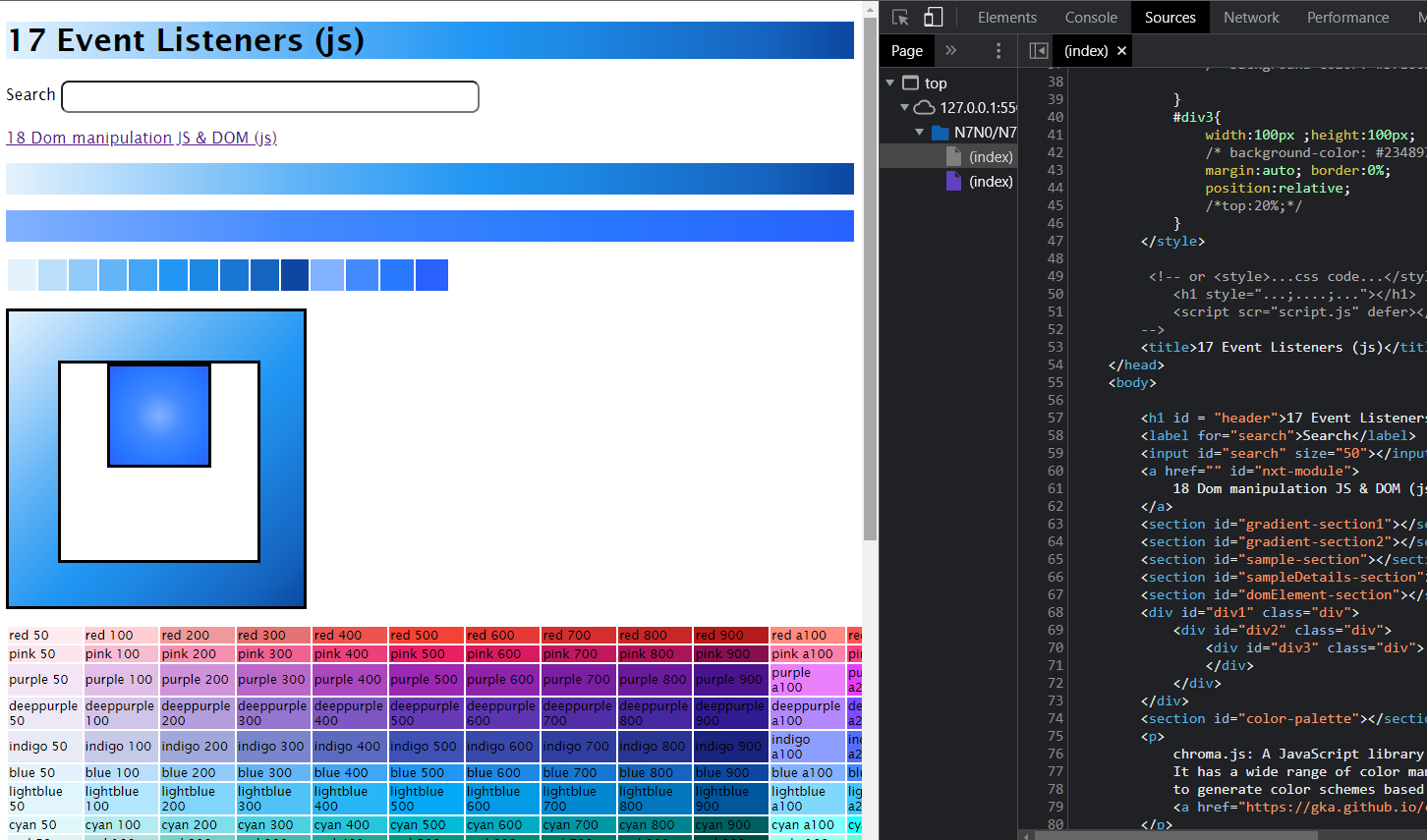












N6N0 Data exchange, integration, transformation, parsing and rendering