front-loader

Small Node.js app that loads a large list of contacts from a JSON file into a HTML dropdown component.

Table of Contents

Technical presentation

This app is written in vanilla Node.js and gives an example of how doing Server Side Rendering (SSR) using streams to load a large JSON file and build a HTML document that will be rendered to the client.

The aim of this implementation is to speed up the rendering using streams in the back-end and so the front-end doesn't bother with calculation/data transformation.

How it works

The JSON file contains 10k contacts information:

```
[
  {
    "name": "Savannah Battle",
    "phone": "+1 (830) 592-2532",
    "email": "savannahbattle@architax.com"
  },
  {
    "name": "Dale Vaughn",
    "phone": "+1 (841) 554-2794",
    "email": "dalevaughn@architax.com"
  },
  ...
]
```

It has been generated on this site: https://www.json-generator.com/.

The contacts file is streamed with `fs.createReadStream` function. The *JSONStream* module, the only one dependency of this project, then parses the file with its duplex stream in object mode.

A transform stream has been implemented to build the whole HTML document. First the head of the document is pushed to its writable stream. Then it specifically handles each object received from the JSON parsing to build a HTML option component

like this: `<option value="${contactString}">${contactString}</option>` Finally the transform flush data adding the HTML document footer.

To end, the transform stream is piped to the HTTP response.

Here is the complete pipeline:

JSON file stream -> **pipe** -> *JSONStream* parsing -> **pipe** -> HTML document builder -> **pipe** -> HTTP response

Browser caching has also been implemented. A file watcher is looking for any changes in the JSON file. If any, the HTML document will be built again, otherwise a *304 Not Modified* is returned to client.

Stack - Node JS

- Language: JavaScript ES6 to ES8
- VM: NodeJS 12.13.0
- Package manager: NPM 6.12.0

Tests

`npm test`

Code quality

Code style follows Airbnb best practices using ESLint.

Unit

Mocha and Chai.

Logging and debugging

For this small app, only `util.debuglog` is used for debugging. Access all the debugging with environment variable *NODE_DEBUG* set to `front-loader*`.

Example: `NODE_DEBUG=front-loader*`

Installation and usage

Clone

`git clone https://github.com/adrienv1520/front-loader.git`

Manual run

`NODE_DEBUG=front-loader* npm start`

Docker image

`docker build .`

```
docker run -e NODE_DEBUG=front-loader* -p 2000:2000 imageId
```

Limitations

- the dropdown component can be very slow to deploy 10k contacts. I don't know any trick to the front-end that could speed up this component;
- the HTML code is hardly maintainable. One solution could be to edit a real .html file and to use `fs.read` function to get the text content to add it to the HTML builder;
- this solution must be modified if you have multiple front-end components that react during the app lifecycle. For an example, you can do all the pipeline process until the HTML builder to create only a select component that could be requested via AJAX or sent via sockets (using *socket.io*).

Licence