Node.js

LAB GUIDE

HTTP Module

1. Create a folder, say my-website. Initialize it as a Node.js project. Create 2 pages
* index.html that includes a few lines about yourself
* contact.html that has your contact details like email id, phone number, and a contact form that takes in user's name, emailid, and message.

Have these pages served on port 3000 using an HTTP server.
* GET http://localhost:3000/ returns the index.html page
* GET http://localhost:3000/contact returns the contact.html page
* POST http://localhost:3000/message accepts the details from the contact form on submission and writes them to a messages.json file (design a schema to store the data)
* Each page should have a menu on top to switch between index page and contact page.

Hints:
* req.url has the URL from incoming request. Decide which file to serve based on it (req is the request object here).
* req is a readable stream (so you can listen for data event on it). Concatenate the chunks received to form the request body. Then parse it to get message details.
* Check the documentation to find out how to retrieve the incoming request's method (GET/POST etc.)

2. Create a simple HTTP server that runs on port 3000, does simple arithmetic operations, and responds with the result.

Some examples
* http://localhost:3000/add?x=12&y=13 returns the string 25
* http://localhost:3000/multiply?x=12&y=13 returns the string 156

Your server must support add, subtract, multiply and divide operations. For an unsupported operation/arguments it must return a sensible error message, with appropriate error code (HTTP status code 401 is for a badly constructed request)

3. Create a simple HTTP server that takes a URL as input, and generates a shorter URL (just like a [URL shortening](https://en.wikipedia.org/wiki/URL_shortening) service like bit.ly does).

Example
POST http://localhost:3000/shorten with request body https://en.wikipedia.org/wiki/Design_rule_for_Camera_File_system should return a unique short URL, says http://localhost:3000/z34DCs. The algorithm and format of exact URL generated is upto you, but generated URL must be unique, and your algorithm MUST always generate the same short URL for a particular URL (the "long" URL). i.e. if the same request is made again, the short URL returned must also be exactly the same.

Finally, when you make a GET request for the short URL, the server must send a redirect request - this is achieved with status code 304, and adding a Location header whose value is the complete URL (the "long" URL). For more information on redirects check [this page](https://en.wikipedia.org/wiki/URL_redirection).

FS Module and HTTP Module

4. Create a json-utilities module (file) that supports the following APIs

* Search for a particular key in a JSON file with a set of properties at the top-level, and get its value.
* Set the key to a particular value in a file similar to the one just above.
* Search for an item with a particular key-value pair in a JSON with an array of objects (items).
* Add/remove/edit items in a file similar to the one just above.

Use this module's APIs in another module. Also use this to make the solution to question 3 simpler.

5. Create a simple file server. The server by default lists all the files/folders in the folder within which it is run. For example, if it is run from the temp folder, then navigating to http://localhost:3000/ should list all the files in the temp folder.

Each listed file/folder is a hyperlink. When a file/folder is clicked, the URL should change to append the relative path to the file/folder. For example, if http://localhost:3000/ list xyz.html, then clicking xyz.html entry results in the request http://localhost:3000/xyz.html (i.e. URL in browser changes to it) – this obviously returns the file contents as response. In case a folder, say docs/ is clicked, the URL changes to http://localhost:3000/docs and the (clickable) list of files and folders in docs/ folder is returned as response.

Extra credits:

* You can also have a .. entry in the folder listing. Clicking it takes the user to the parent folder listing.
* Give an option to the user to create a new file/folder, under the current folder, through the UI.

Modules

6. Create a module to calculate areas of various shapes.
   - Square (public function) - makes use of Rectangle function to calculate area
   - Rectangle (private function)
   - Circle (public function)
   - PI (private variable)

   Make use of this module in another module (i.e. file)

7. Explore the chalk module on npmjs.com - install it and use it.