Lab 5: reactionary

Due: November 8, 2020 at 11:59:59 PM

Introduction

This will be your last lab of the quarter. After this, you should use lab time to work on your final project. In this lab, we'll be exploring how to use Node.js to create a server. Because gifs (that's pronounced gifts without the t) are the only true medium for communication now, you should have a place to store and retrieve them easily. In this lab, you'll create that little library of gifs.

This lab requires that you've completed Homework 6 (installing Node.js and NPM) on your computer. If you haven't done that, you'll need to do that before jumping into this lab. I hope you enjoy this last lab and keep it handy with you, maybe even turn it into an application.

Submission

Due Date: November 8, 2020
Demo: Yes
Format: 1 compressed file prefixed with your SCU Email with the following files:

- server.js
- package.json
- gifs.json

Goals

1. Running a server using Node.js
2. Using express.js to serve a static website
3. Using express.js to create an API

Task 0: Install Node.js

Install Node.js and NPM. If you haven't already, you need to do that before you start the lab. This is also part of your homework from Week 6.

If you're on Windows, use the official download links

1. node.js - https://nodejs.org/en/ (use the LTS version)
2. npm - https://www.npmjs.com/

If you're on Mac

1. Prefer homebrew - brew install nodejs && brew install npm
2. Use the official download links (same as windows)

When you run `node --version` after installing, the version MUST be greater than 13.

If you're unfamiliar with Node.js, you can find a small write up here.

Task 1: Read code!

This task MUST be done before you leave lab for the day. This is mainly confirmation that you know how to get started.

After downloading the starter materials, go to the directory where the starter materials are found and run the command `npm install`. Then, read through the server.js file. Comment each labelled line with your interpretation of what it's doing. You can do a Find for Task 1 to see them all.

Once you've read through and understood the code, run the server using the command node server.js. You can then use `curl` to send an HTTP request to your server. Send a request to the `/moods` endpoint by typing `curl localhost:3000/moods`

Task 2: Write the GET endpoints

Once you've read and understood the code the /moods endpoint. You should be read to write the code for the other two endpoints. I recommend starting with the /gifs endpoint.

For each possible response (can return a gif, mood not found, mood found but no content) make sure that you return an appropriate status code. You should be handling the errors as 400s (404 may be useful here) and the successes as 200s (200 and 204 may be useful here).

1. When a user sends a GET request to /gifs, the server returns the entire res.app.locals.reactions object to the user as the following JSON object

```
{ "mood": [ 'link1', 'link2', 'link3']}
```

2. When a user sends a GET request to /gif/:mood, the server returns the link to a random gif from the array stored under res.app.locals.reactions[mood]
    a. You'll have to figure out though how to express.js can give you the requested mood parameter.
    b. The JSON object should look like the following

```
{ "gif": "link1" }
```

Task 3: Write the POST endpoints

Given that we've just seeded some data using the fs module, it's time to allow users to submit new gifs and new moods. Write 2 new endpoints which allow the user to send POST requests to your server to create new moods and gifs.

1. When a user sends a POST request to /mood with a JSON body, the server looks for the "mood" property on the body. If the property is found, add a new array to the res.app.locals.reactions object with a corresponding key
2. When a user sends a POST request to /gif/:mood with a JSON body, the server looks for the "link" property on the body. If the property is found, the given link is added into the corresponding array for moods.

Remember to use proper error and success codes. Some errors to consider:

1. The body is not properly formatted JSON
2. The body is properly formatted JSON but doesn't have the correct field
3. The mood doesn't exist when trying to POST to /gif/:mood

To test out the post requests using curl, you'll have to use the following format:

```
curl -H "content-type: application/json" -d '{"link":
"https://media.giphy.com/media/13k4VSc3ngLPUY/giphy.gif" }'
localhost:3000/gif/happy
```

curl -h "content-type: application/json" -d '{"link": "https://media.giphy.com/media/13k4VSc3ngLPUY/giphy.gif" }' localhost:3000/gif/happy

Task 4: Write the DELETE endpoint

Given that you can create new entries, you should also be able to delete some possibly mislabeled gifs. Write 2 new endpoints which allow the user to send DELETE requests to your server to delete moods and gifs.

1. When a user sends a DELETE request to /mood/:mood, the server deletes the associated mood from the res.app.locals.reactions object and returns the proper status code.
2. When a user sends a DELETE request to /gif/:mood, the server looks for a JSON body with the "link" property. It then loops through specified mood's array and removes the link if found.

To test out the delete requests using curl, you'll have to use the following format:

```
curl -X DELETE localhost:3000/mood/happy
```

Rubric

This lab has a total of 100 points. Each day late is 10 points off your overall score.

Criteria	Explanation	Points
Code Style	Code is readable	10
Questions.txt	Answers correct	10

Task 1	Demo to TA you can run the server and you get back some response	10
Task 2: Gifs	The endpoint lists all gifts separated by mood with proper status	5
Task 2: Mood	The endpoint lists all gifs of a single mood with proper status	5
Task 3: Create gif	The endpoint returns proper status code and adds gif to a mood	20
Task 3: Create mood	The endpoint returns proper status code and creates a mood	15
Task 4: Delete mood	The endpoint returns proper status code and deletes a mood	10
Task 4: Delete gif	The endpoint return proper status code and deletes a gif	15

Appendix A: Resources

1. List of all status codes: https://www.restapitutorial.com/httpstatuscodes.html
2. Express.js documentation: http://expressjs.com/en/4x/api.html#express
3. fsPromises.readFile documentation: https://nodejs.org/api/fs.html#fs_fspromises_readfile_path_options
4. Using express: https://www.digitalocean.com/community/tutorials/nodejs-express-basics
    a. Everything up until the Middleware section is useful