Assessment: Node/Express 1

Download code <../node-express-1.zip>

> **Warning: Assessments**
>
> Remember, assessments are meant to be completed **by you**, not as a shared exercise with friends or other members of your cohort.
>
> All code submitted should be **written by you**. If you incorporate code from elsewhere, it must be clearly specified.
>
> **Please do not** put your assessment on GitHub.

Part 1: Conceptual

Answer the questions inside the ***conceptual.md*** file.

Part 2: Async and Scripting Practice

Write a Command-Line Script

Write a script, ***urls.js***, that does the following:

- It is called on the command line like `node urls.js FILENAME`, and it reads the contents of ***FILENAME*** (each line of that file will be a URL).
- For each URL, it will get that page (a GET request to the URL) and save the HTML in a new file.
- For each URL, the output filename should be the hostname of the URL. For example, for the input URL *http://yahoo.com/blah/blah*, your script should write the contents to a plain text file called ***yahoo.com***

Handle Errors

- If you cannot read the original file (***FILENAME***), immediately end your script with an error printed to the console.
- If you cannot download a particular URL or cannot write to a particular output file, print an error to the console saying so, but **continue on with the rest of the script.**

Examples

We provide a file, ***urls.txt***, which contains:

urls.txt

```
http://rithmschool.com
http://postgresql.com
http://foozlemcblargh.com
https://nodejs.org/api/console.html
```

Invoking the script should look something like this:

```
$ node urls.js urls.txt
Couldn't download http://foozlemcblargh.com <http://foozlemcblargh.com>
Wrote to rithmschool.com
Wrote to nodejs.org
Wrote to postgresql.com

$ ls
nodejs.org
postgresql.com
rithmschool.com

$ cat nodejs.org
<!doctype html>...
```

Bonus!

A straightforward implementation of that would download the URLs one-by-one.

This works, but it would be faster to do all of the downloading at once.

As a bonus, see if you can find a nice way to start all of the downloading at once.

Part 3: Broken App!

Yay! You got a job as an Express developer!

Boo! The last developer left in a hurry, and left a mess.

You've been given a non-working small app, ***broken-app***. It should have one route:

POST /

> Given a list of GitHub users names, this should return information about those developers.
>
> Should get JSON body like `{developers: [username, ...]}`
>
> Should return `[ {name, bio}, ... ]`

For example, if we POST:

```
{ "developers": ["joelburton", "elie"] }
```

to `/`, it should return:

```
[
  {
    "bio": "Open source developer. Former dev at Apple...",
    "name": "Joel Burton"
  },
  {
    "bio": "Co-founder + Lead Instructor @rithmschool ",
    "name": "Elie Schoppik"
  }
]
```

The order of the output array does not matter.

Fix It!

Most of the script is written and working. There's a least one bug in it, though, so it doesn't work now.

Find and fix any bugs!

> **Warning: GitHub Rate Limit**
>
> GitHub has some pretty strict rate limit rules:
>
> **"For unauthenticated requests, the rate limit allows for up to 60 requests per hour. Unauthenticated requests are associated with the originating IP address and not the user making requests."**
>
> If you exceed 60 requests per hour, your code may appear like it has a bug, but it could just be the case that you have to wait a bit before proceeding.

Refactor It!

In addition, the code is *terrible*. Use some of the common best practices you've learned for Express apps, and make this better. Feel free to change variable names, add comments, write helper functions/middleware, etc.

Document Issues!

Lastly, you want to get credit with your boss for all the hard work you put in fixing this code.

In the starter code, document bad things into the ***issues.md*** file.

Solution

Take a look at the our solution here. <solution/index.html> (password required).