Node Intro

Download exercise

In this exercise, you will practice working with Node, NPM, and the ***file*** API.

You'll be building a program similar to the standard UNIX utility, cat 🐈.

Step 0

- Run `npm init` to create a node project inside the project folder
- Create a git repository in your project folder
- Add ***node_modules*** to a ***.gitignore*** file

Step 1

In ***step1.js***, write a function, ***cat***.

It should take one argument, ***path***, and it should read the file with that path, and print the contents of that file.

Then, write some code that calls that function, allowing you to specify the path argument via the command line. For example:

```
$node step1.js one.txt This is file one.
```

If you give it the path of a non-existent file, it should print that error and halt the script execution:

```
$node step1.js huh.txt Error reading huh.txt: Error: ENOENT: no such file or directory, open 'huh.txt'
```

Step 2

Copy over your ***step1.js*** code to ***step2.js***

Add a new function, ***webCat***. This should take a URL and, using axios, should read the content of that URL and print it to the console.

Modify the code that invoked ***cat*** so that, based on the command-line args, it decides whether the argument is a file path or a URL and calls either ***cat*** or ***webCat***, respectively.

```
$node step2.js one.txt This is file one. $node step2.js http://google.com <!doctype html><html ...
```

If there is an error getting the page, it should print that.

```
$node step2.js http://rithmschool.com/no-such-path Error fetchinghttp://rithmschool.com/no-such-path: Error: Request failed with status code 404
```

Step 3

Copy over your ***step2.js*** code to ***step3.js***.

Add a feature where, on the command line, you can *optionally* provide an argument to output to a file instead of printing to the console. The argument should look like this: `--out output-filename.txt readfile-or-url`.

Current features should still work the same:

```
$node step3.js one.txt This is file one. $node step3.js http://google.com <!doctype html><html ...
```

However, if `--out` follows your script name, it should take the next argument and use that as the path to write to.

For example:

```
$node step3.js --out new.txt one.txt $# no output, but new.txt contains contents of one.txt $node step3.js --out new.txt http://google.com $# no output, but new.txt contains google's HTML
```

Make sure you handle errors trying to write to the file:

```
$node step3.js --out /no/dir/new.txt one.txt Couldn't write /no/dir/new.txt: Error: ENOENT: no such file or directory, open '/no/dir/new.txt'
```

It may be the case at this point that you have functions like this:

```
function cat(path) { } function catWrite(path, filename) { } function webCat(url) { } function webCatWrite(path, filename) { }
```

If so, you probably have a lot of duplicated code among these functions. Try to structure your code so that:

- your functions are small, could be tested, and do one thing
- you minimize duplication of code throughout

Further Study

- Enhance your script so you can pass any number of arguments on the command line and it would output all of those files/URLs in sequence.

Solution

View our solution