

(/)



Curriculum

Short Specializations ^

Average: 97.3% v

0x05. NodeJS Basics

Back-end

JavaScript

ES6

NodeJS

ExpressJS

By: Johann Kerbrat, Engineering Manager at Uber Works

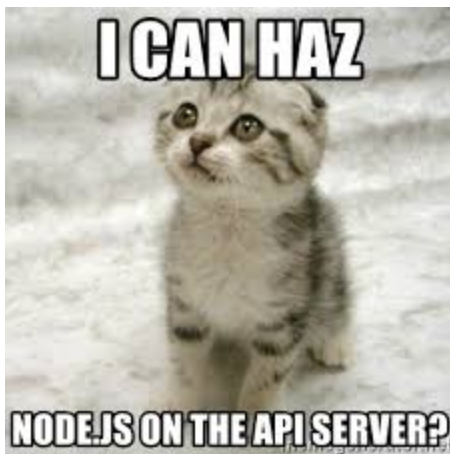
Weight: 1

Project over - took place from Jan 29, 2024 6:00 AM to Jan 31, 2024 6:00 AM

☒ An auto review will be launched at the deadline

In a nutshell...

- **Auto QA review:** 18.75/32 mandatory
- **Altogether: 58.59%**
 - Mandatory: 58.59%
 - Optional: no optional tasks



Resources

Read or watch:

- Node JS getting started (/rltoken/hROgW3QO9jqFnFP-Nzwh8A)



- Process API doc (/rltoken/Wt69QV2xygB4GEqob26AjQ)
- (/). Child process (/rltoken/IS4y9RCbIX71W_oeXpymw)
- Express getting started (/rltoken/XsfrhG9NRLuuaTpVZIZv_g)
- Mocha documentation (/rltoken/EBGDj1FwLrK_y4kgxp8hfg)
- Nodemon documentation (/rltoken/vnDSbLsicMDdxcF5YUSXlg)

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/vXmxtc5JH_CelWReMTNhDA), **without the help of Google**:

- run javascript using NodeJS
- use NodeJS modules
- use specific Node JS module to read files
- use `process` to access command line arguments and the environment
- create a small HTTP server using Node JS
- create a small HTTP server using Express JS
- create advanced routes with Express JS
- use ES6 with Node JS with Babel-node
- use Nodemon to develop faster

Requirements

- Allowed editors: `vi`, `vim`, `emacs`, Visual Studio Code
- All your files will be interpreted/compiled on Ubuntu 18.04 LTS using `node` (version 12.x.x)
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project, is mandatory
- Your code should use the `js` extension
- Your code will be tested using `Jest` and the command `npm run test`
- Your code will be verified against lint using `ESLint`
- Your code needs to pass all the tests and lint. You can verify the entire project running `npm run full-test`
- All of your functions/classes must be exported by using this format: `module.exports = myFunction;`



Provided files

database.csv

```
firstname,lastname,age,field
Johann,Kerbrou,30,CS
Guillaume,Salou,30,SWE
Arielle,Salou,20,CS
Jonathan,Benou,30,CS
Emmanuel,Turlou,40,CS
Guillaume,Plessous,35,CS
Joseph,Crisou,34,SWE
Paul,Schneider,60,SWE
Tommy,Schoul,32,SWE
Katie,Shirou,21,CS
```

package.json

Click to show/hide file contents

babel.config.js

Click to show/hide file contents

.eslintrc.js

Click to show/hide file contents

and...

Don't forget to run `$ npm install` when you have the `package.json`

Tasks

0. Executing basic javascript with Node JS

mandatory

Score: 100.0% (Checks completed: 100.0%)

In the file `0-console.js`, create a function named `displayMessage` that prints in `STDOUT` the string argument.



```
bob@dylan:~$ cat 0-main.js
const displayMessage = require('./0-console');

displayMessage("Hello NodeJS!");

bob@dylan:~$ node 0-main.js
Hello NodeJS!
bob@dylan:~$
```


Repo:

- GitHub repository: alx-backend-javascript
- Directory: 0x05-Node_JS_basic
- File: 0-console.js

☒ Done!

Help

Check your code

 Get a sandbox

QA Review

1. Using Process stdin

mandatory

Score: 100.0% (Checks completed: 100.0%)

Create a program named `1-stdin.js` that will be executed through command line:

- It should display the message `Welcome to Holberton School, what is your name?` (followed by a new line)
- The user should be able to input their name on a new line
- The program should display `Your name is: INPUT`
- When the user ends the program, it should display `This important software is now closing` (followed by a new line)

Requirements:

- Your code will be tested through a child process, make sure you have everything you need for that

```
bob@dylan:~$ node 1-stdin.js
Welcome to Holberton School, what is your name?
Bob
Your name is: Bob
bob@dylan:~$
bob@dylan:~$ echo "John" | node 1-stdin.js
Welcome to Holberton School, what is your name?
Your name is: John
This important software is now closing
bob@dylan:~$
```

**Repo:**

- GitHub repository: alx-backend-javascript
- (/). Directory: 0x05-Node_JS_basic
- File: 1-stdin.js

☒ Done!

Help

Check your code

> Get a sandbox

QA Review

2. Reading a file synchronously with Node JS

mandatory

Score: 88.33% (Checks completed: 100.0%)

Using the database `database.csv` (provided in project description), create a function `countStudents` in the file `2-read_file.js`

- Create a function named `countStudents`. It should accept a path in argument
- The script should attempt to read the database file synchronously
- If the database is not available, it should throw an error with the text `Cannot load the database`
- If the database is available, it should log the following message to the console `Number of students: NUMBER_OF_STUDENTS`
- It should log the number of students in each field, and the list with the following format: `Number of students in FIELD: 6. List: LIST_OF_FIRSTNAMES`
- CSV file can contain empty lines (at the end) - and they are not a valid student!

```
bob@dylan:~$ cat 2-main_0.js
const countStudents = require('./2-read_file');

countStudents("nope.csv");
```

```
bob@dylan:~$ node 2-main_0.js
2-read_file.js:9
    throw new Error('Cannot load the database');
    ^
```

Error: Cannot load the database

...

```
bob@dylan:~$
bob@dylan:~$ cat 2-main_1.js
const countStudents = require('./2-read_file');

countStudents("database.csv");
```

```
bob@dylan:~$ node 2-main_1.js
Number of students: 10
Number of students in CS: 6. List: Johann, Arielle, Jonathan, Emmanuel, Guillaume,
atie
Number of students in SWE: 4. List: Guillaume, Joseph, Paul, Tommy
bob@dylan:~$
```


Repo:

- GitHub repository: alx-backend-javascript
- Directory: 0x05-Node_JS_basic
- File: 2-read_file.js

☒ Done!

Help

Check your code

 Get a sandbox

QA Review

3. Reading a file asynchronously with Node JS

mandatory

Score: 88.33% (Checks completed: 100.0%)

Using the database `database.csv` (provided in project description), create a function `countStudents` in the file `3-read_file_async.js`

- Create a function named `countStudents`. It should accept a path in argument (same as in `2-read_file.js`)
- The script should attempt to read the database file asynchronously
- The function should return a Promise
- If the database is not available, it should throw an error with the text `Cannot load the database`
- If the database is available, it should log the following message to the console `Number of students: NUMBER_OF_STUDENTS`
- It should log the number of students in each field, and the list with the following format: `Number of students in FIELD: 6. List: LIST_OF_FIRSTNAMES`
- CSV file can contain empty lines (at the end) - and they are not a valid student!



```
bob@dylan:~$ cat 3-main_0.js
const countStudents = require('./3-read_file_async');

countStudents("nope.csv")
  .then(() => {
    console.log("Done!");
  })
  .catch((error) => {
    console.log(error);
  });

bob@dylan:~$ node 3-main_0.js
Error: Cannot load the database
...
bob@dylan:~$
bob@dylan:~$ cat 3-main_1.js
const countStudents = require('./3-read_file_async');

countStudents("database.csv")
  .then(() => {
    console.log("Done!");
  })
  .catch((error) => {
    console.log(error);
  });
console.log("After!");

bob@dylan:~$ node 3-main_1.js
After!
Number of students: 10
Number of students in CS: 6. List: Johann, Arielle, Jonathan, Emmanuel, Guillaume, Katie
Number of students in SWE: 4. List: Guillaume, Joseph, Paul, Tommy
Done!
bob@dylan:~$
```

Tips:

- Using asynchronous callbacks is the preferred way to write code in Node to avoid blocking threads

Repo:

- GitHub repository: alx-backend-javascript
- Directory: 0x05-NodeJS_basic
- File: 3-read_file_async.js

☒ Done![Help](#)[Check your code](#)[Get a sandbox](#)[QA Review](#)

4. Create a small HTTP server using Node's HTTP module

mandatory

Score: 65.0% (Checks completed: 100.0%)

In a file named `4-http.js`, create a small HTTP server using the `http` module:

- It should be assigned to the variable `app` and this one must be exported
- HTTP server should listen on port 1245
- Displays `Hello Holberton School!` in the page body for any endpoint as plain text

In terminal 1:

```
bob@dylan:~$ node 4-http.js
...
```

In terminal 2:

```
bob@dylan:~$ curl localhost:1245 && echo ""
Hello Holberton School!
bob@dylan:~$
bob@dylan:~$ curl localhost:1245/any_endpoint && echo ""
Hello Holberton School!
bob@dylan:~$
```

Repo:

- GitHub repository: `alx-backend-javascript`
- Directory: `0x05-Node_JS_basic`
- File: `4-http.js`

☒ Done!

Help

Check your code

 Get a sandbox

QA Review

5. Create a more complex HTTP server using Node's HTTP module

mandatory

Score: 32.5% (Checks completed: 50.0%)

In a file named `5-http.js`, create a small HTTP server using the `http` module:

- It should be assigned to the variable `app` and this one must be exported
- HTTP server should listen on port 1245
- It should return plain text
- When the URL path is `/`, it should display `Hello Holberton School!` in the page body
- When the URL path is `/students`, it should display `This is the list of our students` followed by the same content as the file `3-read_file_async.js` (with and without the database) - the name of the database must be passed as argument of the file
- CSV file can contain empty lines (at the end) - and they are not a valid student!



Terminal 1:

(/)

```
bob@dylan:~$ node 5-http.js database.csv
...
```

In terminal 2:

```
bob@dylan:~$ curl localhost:1245 && echo ""
Hello Holberton School!
bob@dylan:~$
bob@dylan:~$ curl localhost:1245/students && echo ""
This is the list of our students
Number of students: 10
Number of students in CS: 6. List: Johann, Arielle, Jonathan, Emmanuel, Guillaume, Katie
Number of students in SWE: 4. List: Guillaume, Joseph, Paul, Tommy
bob@dylan:~$
```

Repo:


- GitHub repository: alx-backend-javascript
- Directory: 0x05-NodeJS_basic
- File: 5-http.js

☐ Done?

Help

Check your code

Ask for a new correction

 Get a sandbox

QA Review

6. Create a small HTTP server using Express

mandatory

Score: 65.0% (Checks completed: 100.0%)

Install Express and in a file named `6-http_express.js`, create a small HTTP server using Express module:

- It should be assigned to the variable `app` and this one must be exported
- HTTP server should listen on port 1245
- Displays `Hello Holberton School!` in the page body for the endpoint `/`

In terminal 1:

```
bob@dylan:~$ node 6-http_express.js
...
```

In terminal 2:



```
bob@dylan:~$ curl localhost:1245 && echo ""
Hello Holberton School!
bob@dylan:~$
bob@dylan:~$ curl localhost:1245/any_endpoint && echo ""
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Cannot GET /lskdlskd</pre>
</body>
</html>
bob@dylan:~$
```

Repo:

- GitHub repository: alx-backend-javascript
- Directory: 0x05-NodeJS_basic
- File: 6-http_express.js

☒ Done!

Help

Check your code

> Get a sandbox

QA Review

7. Create a more complex HTTP server using Express

mandatory

Score: 48.75% (Checks completed: 75.0%)

In a file named `7-http_express.js`, recreate the small HTTP server using Express :

- It should be assigned to the variable `app` and this one must be exported
- HTTP server should listen on port 1245
- It should return plain text
- When the URL path is `/`, it should display `Hello Holberton School!` in the page body
- When the URL path is `/students`, it should display `This is the list of our students` followed by the same content as the file `3-read_file_async.js` (with and without the database) - the name of the database must be passed as argument of the file
- CSV file can contain empty lines (at the end) - and they are not a valid student!

Terminal 1:

```
bob@dylan:~$ node 7-http_express.js database.csv
...
```



In terminal 2:

```
bob@dylan:~$ curl localhost:1245 && echo ""
Hello Holberton School!

bob@dylan:~$
bob@dylan:~$ curl localhost:1245/students && echo ""
This is the list of our students
Number of students: 10
Number of students in CS: 6. List: Johann, Arielle, Jonathan, Emmanuel, Guillaume, Katie
Number of students in SWE: 4. List: Guillaume, Joseph, Paul, Tommy
bob@dylan:~$
```

Repo:

- GitHub repository: alx-backend-javascript
- Directory: 0x05-NodeJS_basic
- File: 7-http_express.js

☐ Done?

Help

Check your code

Ask for a new correction

>_ Get a sandbox

QA Review

8. Organize a complex HTTP server using Express

mandatory

Score: 18.57% (Checks completed: 28.57%)

Obviously writing every part of a server within a single file is not sustainable. Let's create a full server in a directory named `full_server`.

Since you have used ES6 and Babel in the past projects, let's use `babel-node` to allow to use ES6 functions like `import` or `export`.

8.1 Organize the structure of the server

- Create 2 directories within:
 - `controllers`
 - `routes`
- Create a file `full_server/utils.js`, in the file create a function named `readDatabase` that accepts a file path as argument:
 - It should read the database asynchronously
 - It should return a promise
 - When the file is not accessible, it should reject the promise with the error
 - When the file can be read, it should return an object of arrays of the firstname of students per fields

8.2 Write the App controller

Inside the file `full_server/controllers/AppController.js`:

- Create a class named `AppController`. Add a static method named `getHomepage`



- The method accepts `request` and `response` as argument. It returns a 200 status and the message (/) Hello Holberton School!

8.3 Write the Students controller

Inside the file `full_server/controllers/StudentsController.js`, create a class named `StudentsController`. Add two static methods:

The first one is `getAllStudents`:

- The method accepts `request` and `response` as argument
- It should return a status 200
- It calls the function `readDatabase` from the `utils` file, and display in the page:
 - First line: This is the list of our students
 - And for each field (order by alphabetic order case insensitive), a line that displays the number of students in the field, and the list of first names (ordered by appearance in the database file) with the following format: Number of students in FIELD: 6. List: LIST_OF_FIRSTNAMES
- If the database is not available, it should return a status 500 and the error message `Cannot load the database`

The second one is `getAllStudentsByMajor`:

- The method accepts `request` and `response` as argument
- It should return a status 200
- It uses a parameter that the user can pass to the browser `major`. The `major` can only be `CS` or `SWE`. If the user is passing another parameter, the server should return a 500 and the error `Major parameter must be CS or SWE`
- It calls the function `readDatabase` from the `utils` file, and display in the page the list of first names for the students (ordered by appearance in the database file) in the specified field `List: LIST_OF_FIRSTNAMES_IN_THE_FIELD`
- If the database is not available, it should return a status 500 and the error message `Cannot load the database`

8.4 Write the routes

Inside the file `full_server/routes/index.js`:

- Link the route `/` to the `AppController`
- Link the route `/students` and `/students/:major` to the `StudentsController`

8.5 Write the server reusing everything you created

Inside the file named `full_server/server.js`, create a small Express server:

- It should use the routes defined in `full_server/routes/index.js`
- It should use the port `1245`

8.6 Update `package.json` (if you are running it from outside the folder `full_server`)



If you are starting node from outside of the folder `full_server`, you will have to update the command dev by: `nodemon --exec babel-node --presets babel-preset-env ./full_server/server.js ./database.csv`

Warning:

- Don't forget to export your express app at the end of `server.js` (`export default app;`)
- The database filename is passed as argument of the `server.js` BUT, for testing purpose, you should retrieve this filename at the execution (when `getAllStudents` or `getAllStudentsByMajor` are called for example)

In terminal 1:

```
bob@dylan:~$ npm run dev
...
```

In terminal 2:

```
bob@dylan:~$ curl localhost:1245 && echo ""
Hello Holberton School!
bob@dylan:~$
bob@dylan:~$ curl localhost:1245/students && echo ""
This is the list of our students
Number of students in CS: 6. List: Johann, Arielle, Jonathan, Emmanuel, Guillaume, Katie
Number of students in SWE: 4. List: Guillaume, Joseph, Paul, Tommy
bob@dylan:~$
bob@dylan:~$ curl localhost:1245/students/SWE && echo ""
List: Guillaume, Joseph, Paul, Tommy
bob@dylan:~$
bob@dylan:~$ curl localhost:1245/students/French -vvv && echo ""
* Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 1245 (#0)
> GET /students/SWES HTTP/1.1
> Host: localhost:1245
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 500 Internal Server Error
< X-Powered-By: Express
< Date: Mon, 06 Jul 2020 03:29:00 GMT
< Connection: keep-alive
< Content-Length: 33
<
* Connection #0 to host localhost left intact
Major parameter must be CS or SWE
bob@dylan:~$
```



If you want to add test to validate your integration, you will need to add this file: `.babelrc`

Click to show/hide file contents

(/)
Repo:

- GitHub repository: alx-backend-javascript
- Directory: 0x05-Node_JS_basic
- File: full_server/utils.js, full_server/controllers/AppController.js, full_server/controllers/StudentsController.js, full_server/routes/index.js, full_server/server.js

☐ Done?

Help

Check your code

Ask for a new correction

>_ Get a sandbox

QA Review

Copyright © 2024 ALX, All rights reserved.

