Assessment: Node/Express 2

Download code <../node-express-2.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.
>
> **Use a private GitHub repo to submit this assignment. Add your mentor as a collaborator. Please do not** put your assessment on public GitHub repo.

Part 1: Conceptual

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

Part 2: Timeword.js

Solve the following problem in JavaScript:

Turn a string of 24h time into words.

You can trust that you'll be given a valid *string* (it will always have a two-digit hour 00-23, and a two-digit minute 00-59). Hours 0-11 are am, and hours 12-23 are pm.

Examples of the output we'd like:

Input	Expected Output
00:00	midnight
00:12	twelve twelve am
01:00	one o'clock am
06:01	six oh one am
06:10	six ten am
06:18	six eighteen am
06:30	six thirty am
10:34	ten thirty four am
12:00	noon

Input	Expected Output
12:09	twelve oh nine pm
23:23	eleven twenty three pm

Write tests for these cases and make sure your code passes these.

Also, do this **without the aid of any external packages**. The goal here is to have you think about how you'd solve the problem, not have you show us how good you are at finding third-party libraries.

Part 3: Buggy App

You've just joined a team that is building a web app for a financial system, *bankly*.

You have a generally-nicely-written application and it even has tests, and all tests pass!

However, **there are several bugs in the application** — they're just not things that are being tested for (so, in a sense, there are two bugs for each: the broken code, and the fact that there's no test that caught it).

Some bugs relate to methods returning slightly-wrong information, or being too flexible in what they accept, or failing to check reasonable security requirements.

Get the App Running

First step is to get everything running. We've made the commands easy for you:

```
cd bankly
npm install
npm run seed # sets up the regular and test databases
npm test  # runs jest
npm start  # runs server w/ 'node'; feel free to change to nodemon
```

Strategies

- read the docstrings for the routes carefully. Treat these as your "source of truth" for what the function is supposed to do.
- look at what tests are provided, and think about whether these tests are strict enough to sufficiently test what the function needs to do.
- think about what tests are missing that could catch these bugs.

Goals

We've put in **six bugs** (it's possible that there are other, unintentional bugs, too).

Be mindful of the definition of a **bug**. If this application is missing a feature or something that would be "nice to have" - that does not mean it's a bug. Please **only include things that certainly lead to errors or security issues** in our application.

You don't need to find all six — but do the best you can!

Warning: one of the bugs is quite tricky and subtle, and would require some careful study to find.

Want a clue on the tricky, hard-to-notice, elusive bug #6? Hover below!

Hint on Last Bug

It's in the ***middleware/auth.js*** file. Take a close look at the function ***authUser***.

What You Should Submit For This

You should turn in:

- a short description of the bug
- A test that catches that case (put in a comment, eg `// TESTS BUG #1`, so we can find it)
- A fix for the code (put in a comment, eg `// FIXES BUG #1`, so we can find it)

Full Example Bug Fix

If you were given an ***add*** function:

add.js

```
/** Given two numbers, add positive versions of them together. */

function addAbsoluteVals(x, y) {
  return Math.abs(x) + y;
}
```

And it had some tests already:

__tests__/add.test.js

```
describe('addAbsoluteVals', function () {
  it('should add two positive numbers', function () {
    expect(addAbsoluteVals(2, 3)).toBe(5);
  });
});
```

You could write a test that fails, proving a bug exists:

__tests__/add.test.js

```
describe('addAbsoluteVals', function () {
  it('should add two positive numbers', function () {
    expect(addAbsoluteVals(2, 3)).toBe(5);
  });

  // TESTS BUG #1
  it('should add two negative numbers', function () {
    expect(add(-2, -3)).toBe(5);
```

```
  });
});
```

Then document the bug in your markdown file:

bugs.md

```
- BUG #1: add() doesn't add positive version of y
```

Then fix the code and document the fix in a comment:

add.js

```
/** Given two numbers, add positive versions of them together. */

function addAbsoluteVals(x, y) {
  // FIXES BUG #1
  return Math.abs(x) + Math.abs(y);
}
```

Solution

View our Solution <solution/index.html>