4. TDD with Mocha & Chai

Github notes

Today's menu

- TDD & Unit testing
- Using Mocha
- Using Chai

Automated Testing

- Writing test before helps clarify the purpose of the function
- Helps with refactoring
- Standardize the results of testing
- Saves time

Test-driven development (TDD) relies on tests to drive development

1. Write a failing test to indicate which functionality needs to be added and how it should behave.
2. Write the minimal amount of code to make the test pass. At this stage, the code doesn't have to be elegant or clean.
3. Refactor the code. Clean up the code to make it more readable and maintainable.

What is unit testing ?

- Testing if a specific functionality of our app is working as expected
- Usually testing a method or function

Key Caracteristics of Good Unit Tests

- Unit tests should be isolated
    - Tests should not rely on external data
    - Keeps the tests consistent
    - Tests should not fail because of external factors
- Unit tests should be fast

- Tests should not produce side effects

Installing Mocha and Chai

- Mocha is a testing framework (test runner)
- Chai is an assertion library

```
npm install --save-dev mocha chai
```

Build-in Assert in NodeJS

- `assert.equal(true, value, [message]);`
- `assert.ok(value, [message])`

```
// these three assertions are equivalent:
assert(expected == 3, "one plus two is three");
assert.ok(expected == 3, "one plus two is three");
assert.equal(expected, 3, "one plus two is three");
```

Chai

- Chai is an assertion library that allows us to more fully embrace BDD.
- You can add a script in your package.json and then run `npm run test`

```
...
"scripts": {
  "test": "mocha"
},
...
```

- Assert NodeJS Examples

View on Compass | Leave Feedback