CHAPTER 14

UNIT TESTING

Chapter Breakdown

14.1 Why Test Your Code

Why Test Your Code?

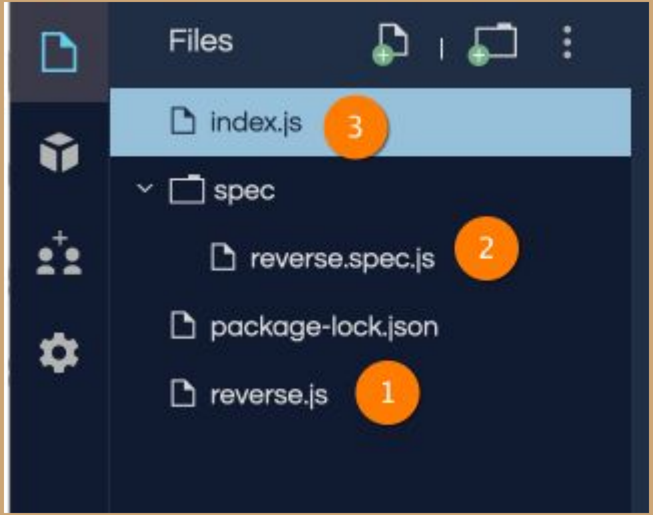
Know if your code really works
Find Regression
Tests as Documentation

- Test Automation helps remove the burden of manual testing.
- Manual testing can take FOREVER
- Unit Testing allows testing the smallest components(units) of code
- Regression testing allows for you to make sure your new code doesn't break your old code
- Allows for self documenting code.

14.2 Hello, Jasmine

Hello Jasmine!

Files structure is important!
All tests should be *test*.spec.js


Files

index.js 3

spec

reverse.spec.js 2

package-lock.json

reverse.js 1

index.js runs the program and reverse.js is the module we are testing.

14.3 Unit Testing in Action

Unit Testing in Action

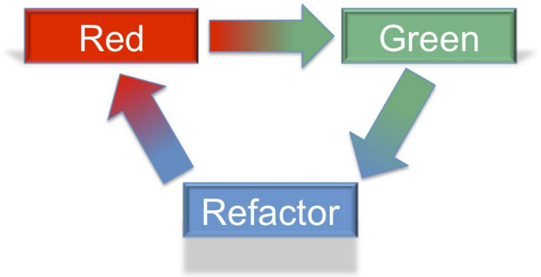
Self Documenting
Positive Cases
Negative Cases
Edge Cases

- You can’t possibly test every situation or input!
- Having both positive and negative results shows what the program should do in most cases
- Edge cases test the extreme limit that the test should be able to handle

14.4 Test-Driven Development

Test Driven Development

Test/Code Cycle
Red/Green/Refactor



- In TDD we **<u>START</u>** with the tests.
- We must start with how the feature will be implemented.
- Then we write the unit test as if the parameter/function we imagined already exists.
- Now write the code! If it code fails, adjust until the test is passed!
- This builds confidence because you know that the code has already passed!

The refactor is also done in a TDD process:

- Decide how to improve the implementation of the feature,
- Change the unit test to use this new idea,
- Run the code to see the test fail,
- Refactor the code to implement the new idea,
- Finally, see the test pass with the refactored design.

14.5 TDD in Action

TDD in action!

LET’S DO SOME CODING SHALL WE?