TEST DRIVEN DEVELOPMENT NOTES

Use git status, ideally between each and every git command (especially when starting out).
Have the documentation up. Link to: MOCHA DOCS & CHAI DOCS API
Ask for help after 10 mins.
Set up your project to use Git and NPM
1. Create or have a project ready then move into that project directory.
2. Use Git
issue the command: $ git init to *initialize* git
1. Use NPM
issue the command: $ npm init to *initialize* npm, this creates a file called package.json in your project directory. Commit to git.
When we install packages through NPM they are installed in a directory named node_modules. We **walways** ignore this file from being tracked by git.
1. At the root of project: create a file named .gitignore
2. in that file, add node_modules/ and save the file.
3. Commit .gitignore to git.
Installing Mocha and Chai with NPM
1. Now that NPM is installed, we can install our **Mocha** and **Chai** packages. Since testing is used during development, we will use the -D flag to let NPM know to put these packages in the "devDependencies" section of our package.json file.

issue the command: $ npm install -D mocha chai
commit your package.json
Create your test/ directory and first test file
1. Although you can tell Mocha where to look for your tests, by default mocha will looks for tests in a folder called test/. We will also need to create a file in that directory. The name of this file will be [yourFileNameHere].spec.js, where [yourFileNameHere] is the name of the file you want to test (this file should not be in your test/ directory). So, for example if you want to write tests for a file named "evenFibs" then the test file should be named evenFibs.spec.js.

create a directory called test/
create your spec file, [yourFileNameHere].spec.js.
2. In your test file, here is some boilerplate starter code, **please type this out**:

```
var chai = require('chai'); // pulls in the 'chai' package installed through NPM
var expect = chai.expect;     // this is how we use chai's `expect` assertion
```

```
chai.should();                    // this is how we use chai's `should` assertion

// please be sure to update the argument for `require` to point to the right file
// the example here the `myTestFunction.js` is one directory "up" from test/
var myTestFunction = require('../myTestFunction.js');

/* all your tests will be written below */

// this describe and it block is our 'hello world', we will replace this later with
describe('Test Suite', function(){
  it('should pass', function(){
    expect(true).to.equal(true);

    var result = myTestFunction();

    expect(result).to.be.equal('hello test');
  });
});
```

more info on require from nodejitsu

NodeJS docs on Module

Next we will create the file that the line: var fileToTest = require('../path/to/file'); refers to.

Create the file which will eventually contain your working code.

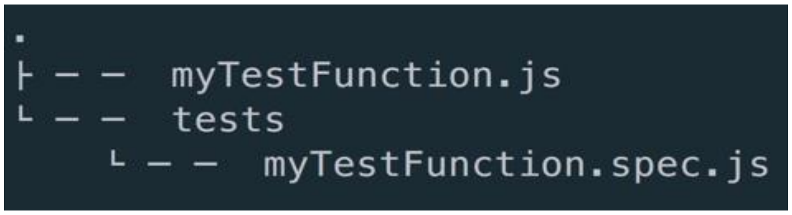
This new file **should not** be created in your test/ directory. It must be created elsewhere, tests/ is only for tests!

This file should export something. Whether it be a function, a class, or an object, you must use either exports or modules.export.

Example code to get started:

```
function myTestFunction() {
  console.log('this is coming from inside of the function "myTestFunction"');
  return 'hello test';
}

module.exports = myTestFunction;
```

Example file structure, notice how `myTestFunction.js` is **NOT in the tests folder.



```
.
├ — —  myTestFunction.js
└ — —  tests
    └ — —  myTestFunction.spec.js
```

Running your tests

There are two ways to run your tests with mocha.
1. Install mocha on a system globally via NPM * issue the command $ npm install -g mocha * after the installation completes you can run mocha **anywhere on your computer** with the command: $ mocha

2. Modify package.json to run the mocha binary file found within node_modules/ * open package.json in your text editor and change this line:

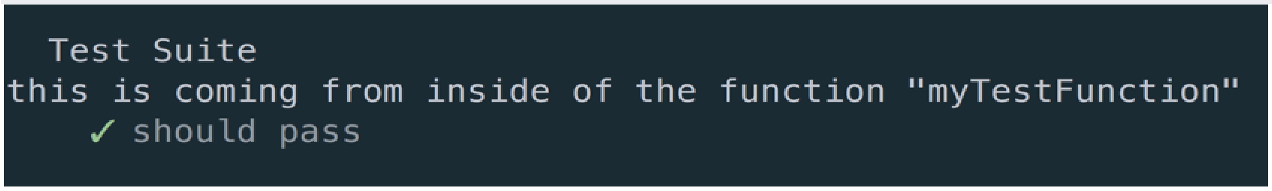
~~~
**from:**

```json
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1"
},
```

**to:**

```json
"scripts": {
  "test": "node_modules/.bin/mocha"
},
```
~~~

* now you can run your tests in your project with the command: `$ npm test`

Why use #2?

If another person clones your repository in the future, they do not need to have mocha installed globally as long as they install packages through the command: $ npm install granted you set up your package.json file correctly. Also, this ensures a certain mocha version (as also defined in package.json).

Run your test with one of the methods mentioned previously. You should see something like this:



```
  Test Suite
this is coming from inside of the function "myTestFunction"
    ✓ should pass
```

If you see it, great. **commit your code**. Otherwise, read the errors and try to fix it. Reach out to an instructor after 10 mins if you are stuck.

Now go write some tests!