Unit Testing Lab

Our environment could be considered complete. But many people would say that you can't call it complete until unit testing is incorporated.

In this lab, we're going to set up unit tests using jest.

Setting up Jest

1. Edit your package.json file. Look for the "test" entry. If it isn't there, go ahead and add it.

```
"test": "jest",
```

2. Next, install Jest with npm

```
npm install --save-dev jest
```

3. Create a folder called "`__test__`" inside the src folder. (Hint: there are two underscores around the word test).
4. Create a test file in the __test__ folder called romanNumeralConverter.test.js. Make it say this:

```
describe('Roman numeral converter', () => {
});
```

describe is a grouping or *suite* of tests. You'll put all tests that are related to one another in a describe group.

5. Rerun Jest:

```
npm run test
```

6. Look in the terminal window. You should see that Jest detected the group of roman numeral tests but there were no tests/specs to run.

Let's create one!

Writing a simple test

7. Inside the *describe* grouping, create a test:

```
it('will return 1 given i', () => {
  let actual = convert('i');
  expect(actual).toEqual(1);
});
```

8. Again, run Jest. This time it will fail because the convert() function doesn't exist. Let's create it.
9. Create a new JavaScript file under src called romanNumeralConverter.js. Put these lines in it:

```
export function convert(romanNumeral) {
  throw "Not yet implemented " + romanNumeral;
}
```

10. Add this line to the top of romanNumeralConverter.test.js:

```
import { convert } from '../romanNumeralConverter';
```

11. Run Jest again. It should still fail because convert is throwing an exception. We'll fix that soon enough. But for now let's automate it.

Making tests part of the build process

It might make sense for us to re-run all unit tests as part of every build and before deploying our site. This will ensure that we have regression testing to protect us from new code breaking existing working code.

12. Open package.json and find your "build" script entry. Add "npm run test" to it.

```
"build": "eslint src --ext .js && npm run test && webpack",
```

13. Run this script and notice that before the webpack transpile your tests are also run. Since your tests are currently failing, the build aborts. Nice.