**Setting up Unit Tests for React with Mocha, JSDOM and Enzyme**

Before start coding your awesome React app, it is always a good idea to set up unit test infrastructure. Once test is set up, you can write tests whenever you feel necessary before waiting to complete the entire app development.

[Mocha](https://mochajs.org/) is a JavaScript test framework that runs on Node.js. [JSDOM](https://github.com/jsdom/jsdom) can mock the browser in the node execution environment. Enzyme is a React test utility, which simulate React component behaviours. With [Enzyme](https://github.com/airbnb/enzyme), you can easily mount React component and write unit tests for the component’s expected behaviours.

This post is focused on JavaScript React. If you are interested in setting up unit tests with these tools, you can check out [this post](https://www.mydatahack.com/how-to-set-up-react-test-with-mocha-and-enzyme-for-typescript/). The set up is slightly different, but quite similar between JavaScript and TypeScript. It is always good to know how to do it in both languages.

I prefer Mocha to [Jest](https://jestjs.io/). I even prefer using [Jasmine](https://jasmine.github.io/) to Jest. This is mainly because Jest runs slow. Or more precisely, I haven’t really spent enough time to make Jest run fast. There must be a trick to run it faster by enabling caching or something. Check out this AirBnb blog about migrating from mocha to jest [here](https://medium.com/airbnb-engineering/unlocking-test-performance-migrating-from-mocha-to-jest-2796c508ec50). Jest is slow is probably not true. Jest also has nice features to write unit test for React. In fact, I wrote a small blog about it. Check it out [here](https://www.mydatahack.com/unit-testing-react-form-with-jest-and-enzyme/) if you are more of a Jest person.

**Setting up**

1. **Installation**

First of all, we need babel register used as a reporter in Mocha command. This is because tests run in Node.js execution environment and it does not support all the newer syntax used in React like import statement. You probably have @babel/core already installed for your React project. If so, you only need to install @babel/register.

If you are confused about babel, I wrote a blog about it. Check it out [here](https://www.mydatahack.com/clarifying-which-babel-to-use-for-compiling-react-today/).

[cc lang="text" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

npm i --save-dev @babel/core @babel/register

[/cc]

We can install all the testing tools. Enzyme needs adapter for the correct React version you are using.

[cc lang="text" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

npm i --save-dev mocha chai fake jsdom sinon

npm i --save-dev enzyme enzyme-adapter-react-16

[/cc]

1. **Mocha command in Package.json**

You need to add @babel/register and the jsdom set up file path (here, I called it browser.js). npm test now can run the test.

[cc lang="text" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

"test": "mocha -r @babel/register -r ./browser.js \*\*/test/\*\*/\*.spec.js --recursive"

[/cc]

1. **Adding jsdom config**

Create a browser.js file in the root folder as below. All the code is from the documentation [here](https://airbnb.io/enzyme/docs/guides/jsdom.html). What is missing is enzyme configuration. You need to add it at the bottom of the file. It will set enzyme accessible from any test file.

<pre class=" toolbar:2 lang:js font-size:12px decode:true" >

</pre>

That’s it. npm test will execute tests when you add them. Now let’s write some tests.

**Unit Test Examples**

**Simple UI test example**

<pre class=" toolbar:2 lang:js font-size:12px decode:true" >

</pre>

**Spy onSubmit handler example**

<pre class=" toolbar:2 lang:js font-size:12px decode:true" >

</pre>