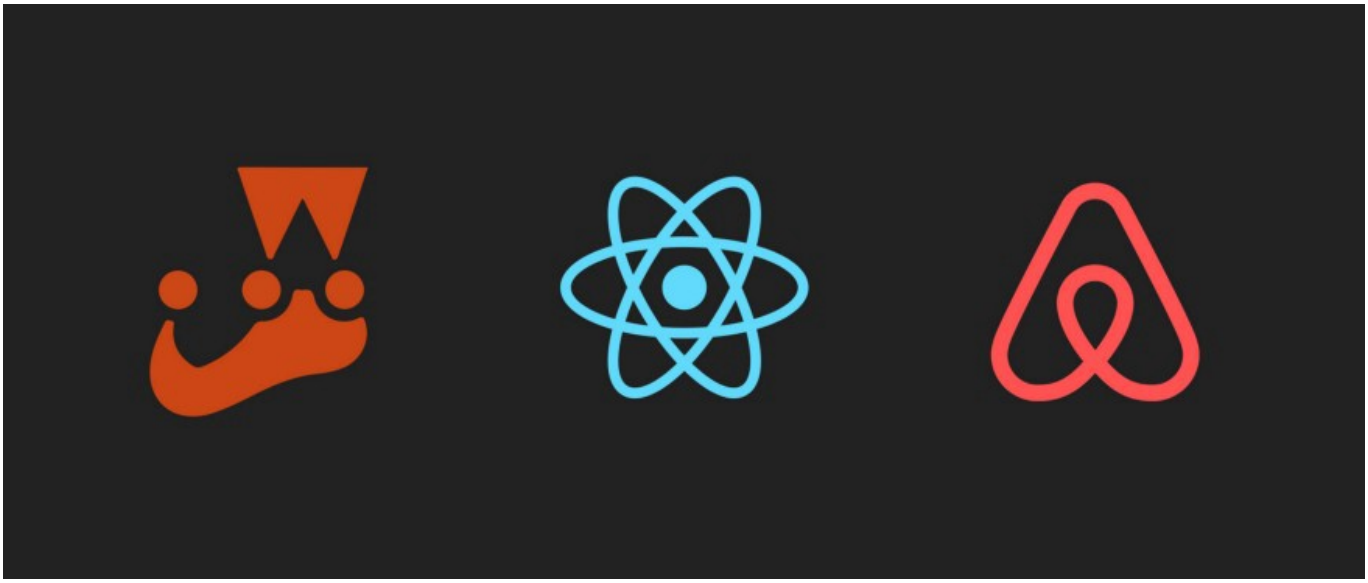
How to Setup Jest & Enzyme in your Existing React App in 5 mins

Kaiz Hudda Follow
Jul 28, 2019 · 3 min read

This post assumes you are using Facebook's Create React App Boilerplate.



Use Jest/Enzyme to build robust test suites for your React application

Jest

Jest is a JavaScript based test runner, which allows tests to be run blazing fast and in parallel. Jest also allows the ability for test to run in watch mode, hence changes to files will re-run the tests instantly as you are developing. It also generates code coverage reports which helps in understanding test coverage percentage.

Snapshot testing is another great addition to Jest. Snapshot testing as its name suggests is a feature that allows components to be tested by taking a snapshot of what the component should look like internally once it has been rendered.

Enzyme

Enzyme is a JavaScript Testing utility for React that makes it easier to test your React Components' output. You can also manipulate, traverse, and in some ways simulate run time given the output.

Setup and install

In the terminal or command prompt, install the following dependencies to your existing create-react-app project.

```
npm install enzyme enzyme-adapter-react-16 enzyme-to-json --save-dev
```

or using `yarn`

```
yarn add enzyme enzyme-adapter-react-16 react-test-renderer enzyme-
to-json --dev
```

Note: *we do not need to install Jest, as CRA is bundled with Jest out of the box.*

Configuration

Next, lets configure Jest & Enzyme to work together with React.

In the `src/` directory, create a new file called `setupTests.js`

```
import { configure } from 'enzyme';
import Adapter from 'enzyme-adapter-react-16';

configure({ adapter: new Adapter() });
```

In the `package.json` file, add the following Jest configuration:

```
"jest": {
  "snapshotSerializers": ["enzyme-to-json/serializer"],
  "collectCoverageFrom": ["src/**/*.js", "!src/index.js"],
```

```
  "coverageReporters": ["text"]
}
```

`enzyme-to-json` helps make snapshot tests more readable & by adding it to the `snapshotSerializers` it is automatically applied to each Snapshot test via this above configuration.

`collectCoverageFrom` indicates for which set of files should coverage information be collected. In our case, we have restricted Jest to run on all files with `.js` extension except `src/index.js` as it is usually just boilerplate code, which doesn't need to be tested.

`coverageReporters` tells jest to output the coverage report via terminal instead of creating an html page to display these results. As a developer it is much easier to check the terminal to understand where you stand in terms of test coverage.

Our first test

Lets say we have a component called `Title.js`

```
import React from 'react';

const Title = ({ children }) => (
  <h1>{children}</h1>
);

export default Title;
```

To create our first test, we will create a new file called, `Title.test.js` , this will be collocated next to `Title.js` component file.

```
import React from 'react';
import { shallow } from 'enzyme';
import Title from './Title';

const title = 'Test Title';
let wrapped = shallow(<Title>{title}</Title>);
```

```
describe('Title', () => {
  it('should render the Title Component correctly', () => {
    expect(wrapped).toMatchSnapshot();
  });

  it('renders the Titles children', () => {
    expect(wrapped.find('h1').text()).toEqual(title);
  });
});
```

The first `it` block will create a new snapshot of the `Title` component.

The second `it` block will do an assertion to check if the value in `h1` is **equal** to the variable named `title`, which stores **'Test Title'**. If true the test will pass.

Running our first test

Type `npm test` in the root directory of your project via cmd/terminal. This will run all the test files in the CRA project and output the results to the terminal/cmd.

Test results

To generate a code coverage report, run `npm test --watchAll --coverageReport`. This report will detail, in depth information about each file including, statement, branch, functions, lines coverage percentage.

Conclusion

This was a quick tutorial on how to setup Jest and Enzyme with an existing `create-react-app` application. For more examples on how to use Enzyme to its fullest potential and to start learning how to use Jest and Enzyme with React, please refer to the official documentation. Also, check out this testing crash course on Udemy, if you would like gain more practical experience with using these tools.

Let me know what you think of Jest/Enzyme and React? I would love to hear from you in the comments below.

JavaScript React Testing Jest Enzyme