Member-only story

When to Write Jest Snapshot Tests

Where to start and knowing when to use them

Chris · Follow
Published in JavaScript in Plain English · 4 min read · Mar 7, 2019

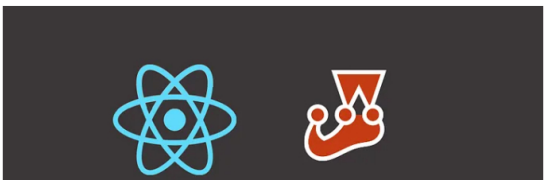
Photo by Jakob Owens on Unsplash

When I was investigating testing frameworks, specifically ones to integrate with React, Jest was an obvious choice.

One of the features of Jest was a new tool called snapshot testing. Initially, it sounded very intriguing but after doing more research and seeing many divided opinions with developers, I was cautious.

Here is what our team thinks about snapshot testing and what we have learned through experience:

- When to write them,
- Why to write them,
- How to utilize them to protect against regression.



React + Jest = a match made in heaven!

Snapshot testing is a feature of Jest that allows you to test Javascript objects.

They work well with React components because when you render a component you can view the DOM output and create a *"snapshot"* at the time of the test run. These types of tests are helpful to prevent regression because if anything causes the component to change, this test will catch it. In our team's experience, snapshot tests are best utilized when you can pass props to a functional component and test for a limited number of cases.

When to write a snapshot test

- If a component is not updated often
- If a component is not too complex
- If it is easy to see what you are actually testing

These are very vague rules but should quickly help you decide if a snapshot test is appropriate.

If you have a snapshot test which tests a component that is updated often then you need to be constantly updating the accompanying test. If you get into a bad habit of just updating these tests without carefully inspecting the output file, the tests lose their value.

Top highlight

If a component is too complex it's difficult to determine what is actually being tested. If there's a ton of business logic and conditional statements that occur before render, how can someone who didn't work on the feature know what's going on?

The most important rule to remember when writing snapshot tests is to give your tests very detailed names. If you can't easily and accurately name your test, chances are it shouldn't be a snapshot test.

Snapshot Test Example

spinner.spec.js using react-testing-library

```
// Spinner component
const Spinner = props => (
 props.loading ?
  <div className={`icon icon-spin text-center ${props.size}
${props.color}} />
  : null)

export default Spinner;

// Spinner spec
describe('Spinner snapshot test ', () => {
    it('should render a large blue spinner', () => {
        const props = {
            loading: true,
            size: "large",
            color: "blue"
        };

const container = render(<Spinner {...props} />);
        expect(container.firstChild).toMatchSnapshot();
    })
})
```

spinner.spec.js.snap

```
// Jest Snapshot v1

exports[`<Spinner /> Snapshot Tests renders as expected 1`] = `
<div
  class="icon icon-spin icon-large icon-blue"
/>
```

This is a simple example of a snapshot test that renders a spinner component. Viewing the .snap output file and seeing a clear test name, you can easily see we are expecting to render a large blue spinner component.

The benefit of a snapshot test in this case, is if someone changed the spinner from a div to a span the test would fail. If we didn't write a snapshot test we might only assert that the size and color are applied correctly. Below is an example of two snapshot assertions. If loading is false then we should not render the component and return null.

spinner.spec.js

```
it('should render a large blue spinner', () => {
    const props = {
        loading: true,
        size: "large",
```

```
        color: "blue"
    };
const container = render(<Spinner {...props} />);
    expect(container.firstChild).toMatchSnapshot();
})
it('should return null if loading is false', () => {
    const props = {
        loading: false
    };
const container = render(<Spinner {...props} />);
    expect(container.firstChild).toMatchSnapshot();
})
```

spinner.spec.js.snap

```
// Jest Snapshot v1
exports[`<Spinner /> should render a large blue spinner 1`] = `
<div
  class="icon icon-spin icon-large icon-blue"
/>;
exports[`<Spinner /> should return null if loading is false 1`] = `
null;
```

Looking at spinner.spec.js, it's not clear what is being tested. This is why it is so important to verify the spinner.spec.js.snap file and go over the output.

Updating a Snapshot Test

The first time we create a snapshot test it creates a .snap reference file and runs your test against this file every time. We must manually update the snapshot test if our component was changed.

```
jest --updateSnapshot
```

Moving Forward

Hopefully, you can take some points and advice from our experience with snapshot testing. When introducing this type of test into your own projects keep in mind:

- Create and follow your own rules of when to write snapshot tests
- Verify components are being rendered correctly in the output file
- Be strict in code reviews

Our team uses React Testing Library and we do not use shallow rendering in our tests. When writing snapshot tests, if our parent component has many child components then these will get rendered as well. On one hand, we have created a test to quickly verify the entire component was rendered, and n the other, it is up to us to check if the component was actually rendered correctly.

Through this process, our team has become more strict on code reviewing these types of tests. We look at exactly what has changed and why. We tend to write a snapshot test as a *catch-all* for shared React components.

Happy snapshotting! 📸

JavaScript React Testing Programming Web Development

514 1

Written by Chris

Follow

284 Followers · Writer for JavaScript in Plain English

Frontend Developer in Vancouver B.C — ccgirard

More from Chris and JavaScript in Plain English

Chris in HackerNoon.com

React Form Validation with Formik and Yup

Better forms with less boilerplate code

5 min read · Mar 15, 2019

2.5K 8

Somnath Singh in JavaScript in Plain English

Bill Gates: People Don't Realize What's Coming

Tech Jobs Won't Exist in 5 Years

13 min read · Apr 13

11.2K 284

The woman in JavaScript in Plain English

My Friend Quit a $160k Job Because It Was a "Developer's...

I am sure many of you are in the same trap, and this article is not clickbait...

4 min read · Apr 15

944 55

Chris in ITNEXT

Testing components with Jest and React Testing Library

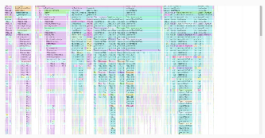
Thinking differently about test coverage

5 min read · Feb 1, 2019

590 5

See all from Chris | See all from JavaScript in Plain English

Recommended from Medium



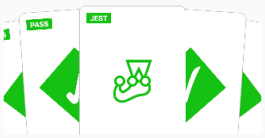
Steven Lemon in Bits and Pieces

Why Is My Jest Suite So Slow?

The simple mistake undermining Jest's performance

12 min read · Jan 11

228 3



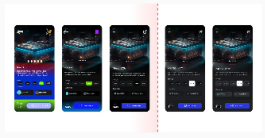
Razvan L in Dev Genius

Write Unit Tests with Jest in Node.js

Jest is a clean and concise JavaScript testing framework developed and maintained by...

4 min read · Jan 5

56



Michal Malewicz

There are FIVE levels of UI skill.

Only level 4+ gets you hired.

6 min read · Apr 25

1.7K 17

Alexander Nguyen in Level Up Coding

Why I Keep Failing Candidates During Google Interviews...

They don't meet the bar.

4 min read · Apr 13

3.2K 103

Gupta Bless in Geek Culture

Guide to NodeJS App Development with Role-Based Permissions

NodeJS enjoys a great deal of popularity among developers. Due to the fact that it is...

6 min read · Dec 8, 2022

12

Vitalii Shevchuk in ITNEXT

🔥 Mastering TypeScript: 20 Best Practices for Improved Code...

Achieve Typescript mastery with a 20-steps guide, that takes you from Padawan to Obi-...

14 min read · Jan 20

950 21

See more recommendations