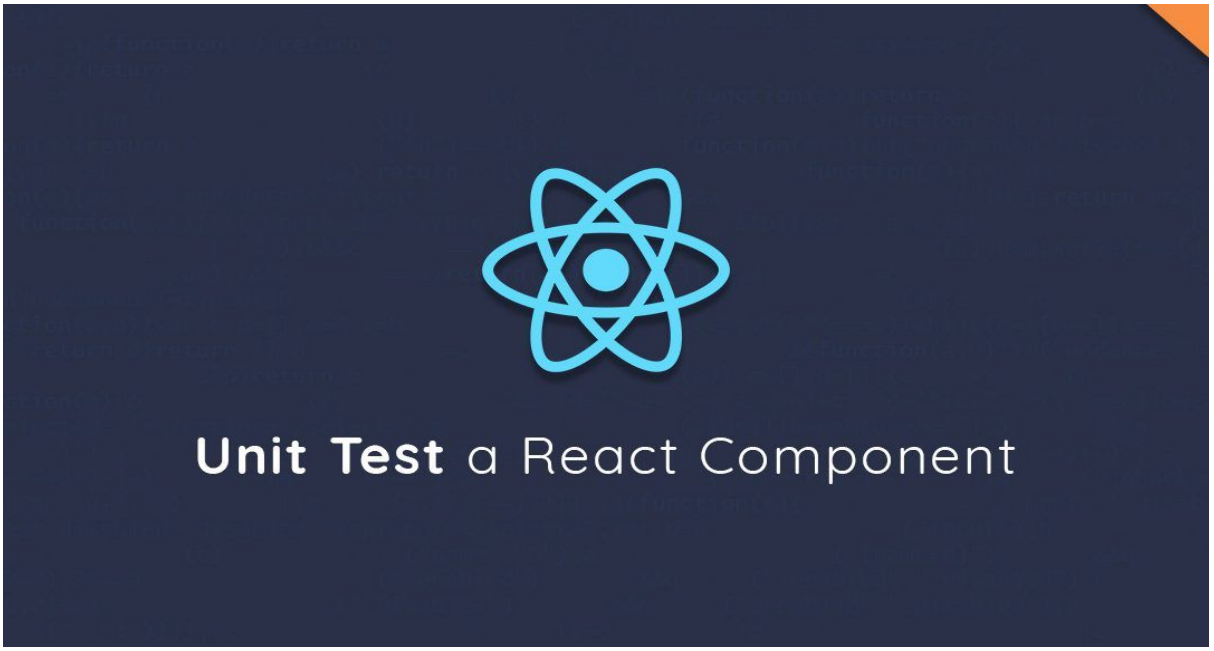


Writing rendering tests for React components

What are Jest and Enzyme?

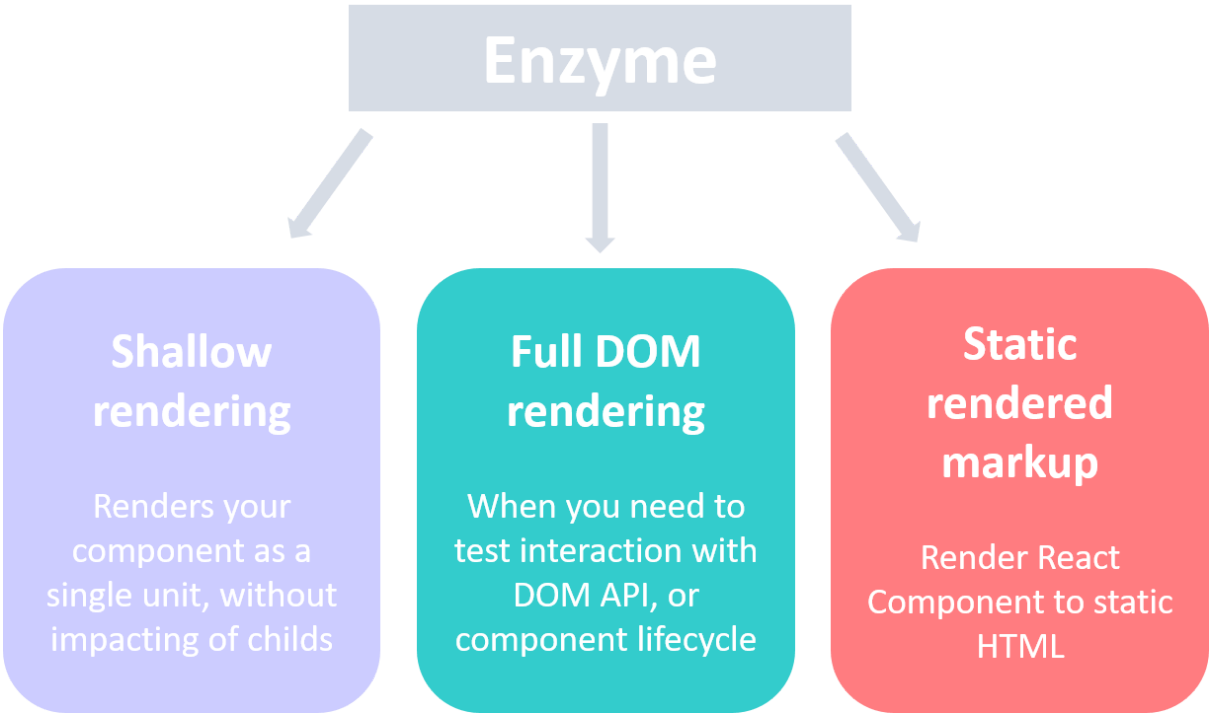
Jest is a testing library that can be used to test simple Javascript code or React components. Using the simple APIs provided by Jest, it is possible to make assertions on how functions should behave and then test our expected outcome against the test outcome. It is very easy to set up and is usable right out of the box. That's why it's the preferred testing library for Facebook.

Enzyme is another library commonly used with Jest. With Enzyme we can create a mock DOM to test whether components are rendered correctly, and whether they behave correctly when acted upon. Enzyme's mock rendering can either be done through shallow rendering or full DOM rendering.

What are Shallow and Full DOM Rendering?

Shallow rendering is used when doing unit testing of a component. This allows for tests to focus only on that component and how it functions, without caring about other components it might interact with. Shallow rendering renders components only one level deep (that is, it does not render any of the child components of the component it renders). This helps you test components in isolation and should be your first point of call for testing React components. Since shallow is testing components as a unit, so it should be used for 'parent' components. (ex. Tables, Wrappers, etc).

Full DOM rendering involves rendering your component and all children components. The allows for more in-depth testing to see how your components on the DOM interact with each other. Full DOM rendering is ideal for cases where you have components that may interact with DOM APIs, or may require the full lifecycle in order to fully test the component (i.e., componentDidMount, etc).



Why do we need unit tests for React components?

TDD (Test-Driven Development) brings many benefits to your code. One of the advantages of high test coverage is that it enables easy code refactoring while keeping your code clean and functional.

Every React component lacking unit tests has legacy code that becomes difficult to maintain. We could add unit tests after we create the production code. However, we may run the risk of overlooking some scenarios that should have been tested. By creating tests first, we have a higher chance of covering every logic scenario in our component, as the component's code will then be checked at every stage which would make it easy to refactor and maintain.

Writing rendering tests for React components of Skill CMS

As the current development of Skill CMS is involving creation of a lot of new components, hence it is a good idea to write rendering tests for them while they are at their initial stages of development. Let us take a look at how simple shallow rendering tests are written for SUSI.AI Skill CMS components.
Following is the code for shallow rendering test of BotBuilder.js component:

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

it('render BotBuilder without crashing',()=>{
  shallow(<BotBuilder />);
});
```

The above code will shallow render the BotBuilder component, and will fail if there is some issues with its shallow rendering. As this is a recently made component, now every time some changes are made in this component, this test will ensure that the code is automatically maintained.

The `it()` function defines a Jasmine test. It is so named because its name makes reading tests almost like reading English. The second argument to the `it()` function is itself a function, that when executed will probably run some number of `expect()` functions. We don’t have the `expect()` function yet in any of the tests. However, `expect()` functions are used to actually test the things you "expect" to be true, and are generally a part of the test.

This is how the passed test looks like when we run the `npm test` command.

This is how rendering tests are written for any React component. We should always create rendering tests after setting up any new React component.

Resources

- GitHub Repo for Enzyme
- Shallow Rendering API
- Testing React components with Enzyme

Tags:

FOSSASIA, SUSI.AI, Testing, React, Enzyme, Jest, Tutorial, GSoC