UNIT TEST

(Mukesh Singh)

TDD Advantages

TDD 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.

- Clean Code
- Easy to refactor
- Easy to maintain

- ❑ We can verify that a particular function in **props was called** when certain a event is dispatched.
- ❑ We can also **get the result** of the render function given the current component's state and match it to a predefined layout.
- ❑ We can even check if the number of the component's **children matches** an expected quantity.

OPTION - 1

JEST & ENZYME

Unit Test using – Jest and Enzyme

Use **Jest** and **Enzyme** to create a React.js component with basic functionality using **TDD**

Jest

Jest is an **open-source** test framework **created by Facebook** that has a great integration with React.js. It includes a command line tool for test execution **similar to** what **Jasmine** and **Mocha** offer. It also allows us to **create mock functions** with **almost zero configuration** and provides a really nice set of matchers that makes assertions easier to read.

Enzyme

Enzyme provides a mechanism to **mount** and **traverse** React.js component trees. This will help us **get access** to its own properties and state as well as its children props in order to run our assertions.

Enzyme offers two basic functions for component **mounting**: **shallow** and **mount**. The shallow function loads in memory only the root component whereas mount loads the full DOM tree.

Project Setup

- create-react-app react-tdd-app
- yarn start
- yarn test

https://github.com/enzymejs/enzyme

https://jestjs.io/

- npm install enzyme
- npm install enzyme-adapter-react-16
- npm install react-test-render

Unit Test

Help ticket.js - react-tdd-app - Visual Studio Code

ticket.js

src > Component > Ticket > ticket.js > ...

```
1   import React, { useState } from 'react'
2
3   export const Ticket = (props) => {
4       const [count, setCount] = useState(0);
5
6       const increment = () => {
7           setCount(count => count + 1);
8       }
9
10      return (
11          <div>
12              <h1 className="title">{props.name}</h1>
13              <button onClick={increment}>Add</button
14              <span className="total">{count}</span>
15          </div>
16      )
17  }
```

Help ticket.test.js - react-tdd-app - Visual Studio Code

ticket.test.js

src > Component > Ticket > ticket.test.js > ...

```
1   import React from 'react'
2   import { shallow } from 'enzyme'
3   import { Ticket } from './ticket'
4
5   test('should increment total when button click', () => {
6       const wrapper = shallow(<Ticket />);
7       const total = wrapper.find('span.total').text();
8       expect(total).toBe('0');
9
10      const button = wrapper.find('button');
11      button.simulate('click');
12      const newTotal = wrapper.find('span.total').text();
13      expect(newTotal).toBe('1');
14  });
```

Github - Code

https://github.com/mukeshdsingh/react-test-enzyme

OPTION - 2

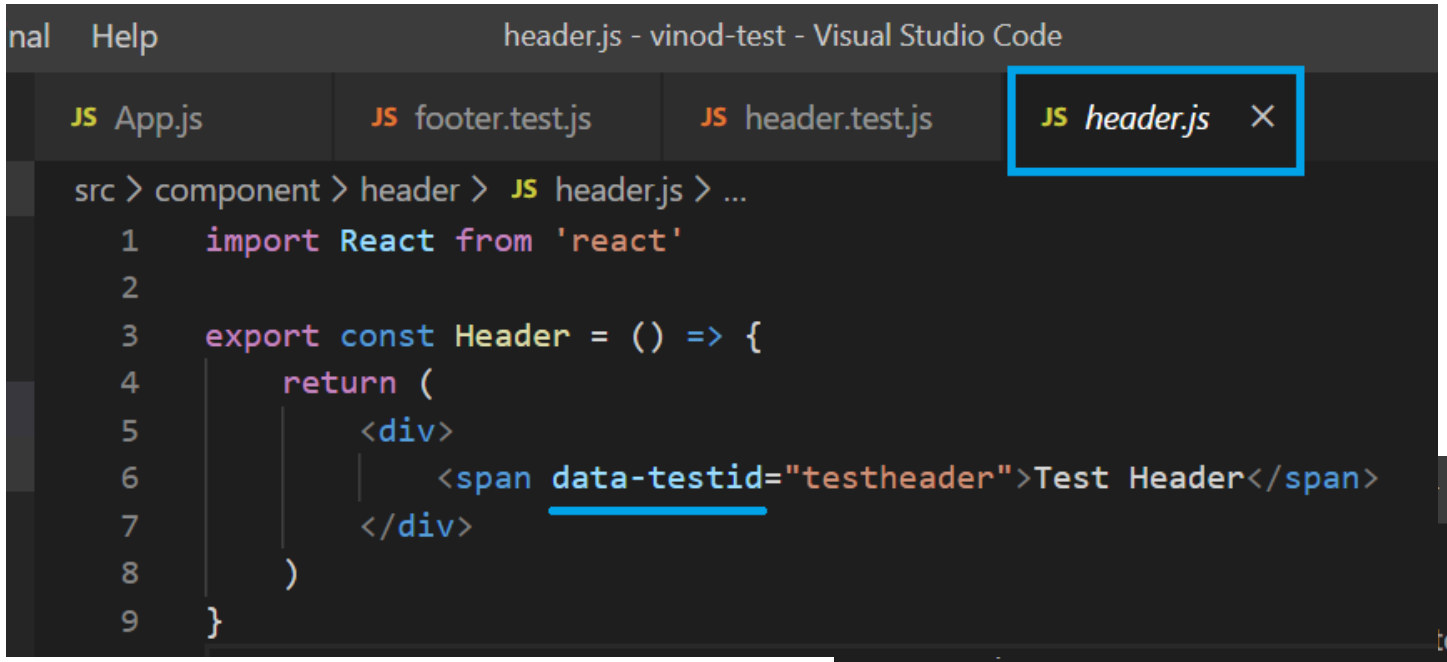
JEST & React-Testing-Library

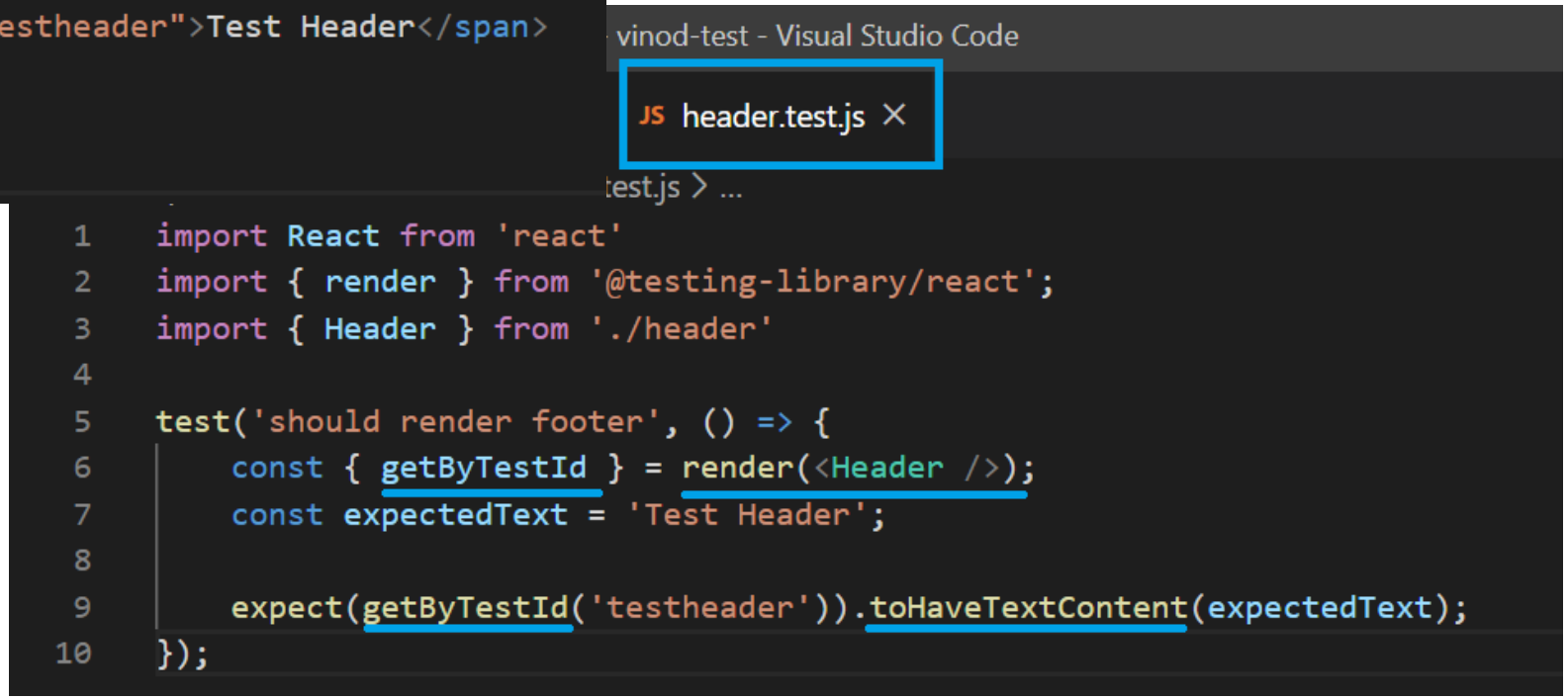
Project Setup

- create-react-app react-tdd-app
- yarn start
- yarn test

- npm install @testing-library/react
- npm install react-test-render
- npm install jest-dom

Unit Test


nal Help
header.js - vinod-test - Visual Studio Code
App.js
footer.test.js
header.test.js
header.js
src > component > header > header.js > ...
1 import React from 'react'
2
3 export const Header = () => {
4 return (
5
6 Test Header
7
8)
9 }


vinod-test - Visual Studio Code
header.test.js
test.js > ...
1 import React from 'react'
2 import { render } from '@testing-library/react';
3 import { Header } from './header'
4
5 test('should render footer', () => {
6 const { getByTestId } = render(<Header />);
7 const expectedText = 'Test Header';
8
9 expect(getByTestId('testheader')).toHaveTextContent(expectedText);
10 });