**React Unit Testing**

**Why we need test cases?**

1. It will avoid bugs in future
2. When ever a change in code done by a colleague, we can get to know the change and ensure no functionality breaks.

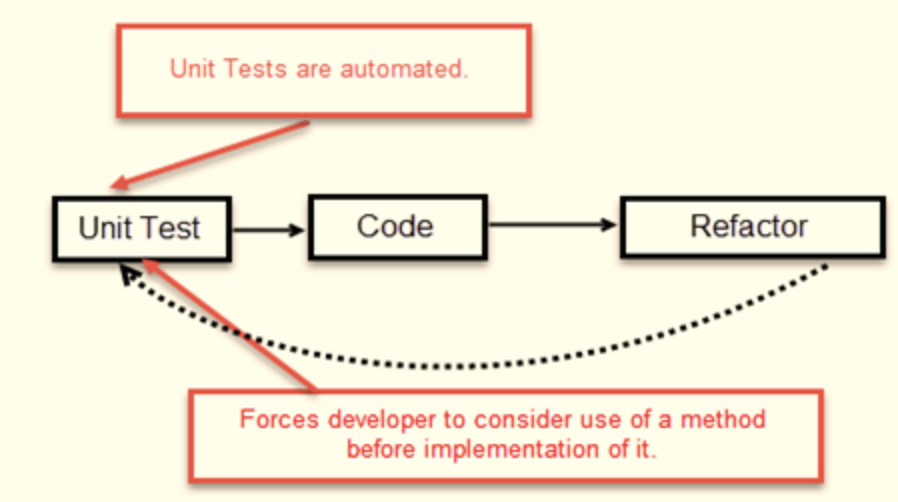
**Different Type of Testing**

1. Manual Testing — Humans manually test the application.
2. Automation Testing — Code testing a Code (Selenium tool).
3. End to End Testing — Test the whole flow (Cypress, Headless Browser — kind of actual browser used for E2E testing — has virtual Dom but no UI).
4. Unit Testing — Core job of Developers. Testing Units of the components.
5. Integration Testing — Testing combined units of the component.

**Test driven development**

Test-driven development (TDD), also called test-driven design, is **a method of implementing software programming that interlaces unit testing, programming and refactoring on source code**.

Writing test cases before developing code. But the development time increases a lot.



Example for TDD

File name - sum.js  
  
test('check sum of two numbers', () => {  
expect(sum(2,5)).toBe(5);  
expect(sum()).toBe("Please pass two arguments");  
})  
  
export const sum = (a,b) => a+b

**Jest**

Jest is a delightful JavaScript Testing Framework with a focus on simplicity.

It works with projects using: [Babel](https://babeljs.io/), [TypeScript](https://www.typescriptlang.org/), [Node](https://nodejs.org/), [React](https://reactjs.org/), [Angular](https://angular.io/), [Vue](https://vuejs.org/) and more!

[**Jest**  
*Jest is a JavaScript testing framework designed to ensure correctness of any JavaScript codebase. It allows you to…*jestjs.io](https://jestjs.io/)

**React Testing Library**

React Testing Library uses jest behind the scenes.

[**React Testing Library | Testing Library**  
*React Testing Library builds on top of DOM Testing Library by adding APIs for working with React components. Projects…*testing-library.com](https://testing-library.com/docs/react-testing-library/intro/)

**Step 1:** Install react testing lib and jest

npm install --save-dev @testing-library/react  
npm i -D jest

**Step 2:** Configure jest

npx jest --init // to install only once

Would you like to use Typescript for the configuration file? › (y/N) N  
  
? Choose the test environment that will be used for testing › - Use arrow-keys. Return to submit.  
❯ node  
 jsdom (browser-like)  
jsdom  
  
? Do you want Jest to add coverage reports? › (y/N) y  
  
? Which provider should be used to instrument code for coverage? › - Use arrow-keys. Return to submit.  
❯ v8  
 babel  
babel  
  
? Automatically clear mock calls, instances, contexts and results before every test? › (y/N) y

mac@Macs-MacBook-Pro-2 Chapter-13-Unit-testing-Jest % npx jest --init  
  
The following questions will help Jest to create a suitable configuration for your project  
  
✔ Would you like to use Typescript for the configuration file? … no  
✔ Choose the test environment that will be used for testing › jsdom (browser-like)  
✔ Do you want Jest to add coverage reports? … yes  
✔ Which provider should be used to instrument code for coverage? › babel  
✔ Automatically clear mock calls, instances, contexts and results before every test? … yes  
  
📝 Configuration file created at /Users/mac/Documents/Namaste-React/Coding-Assignments/Chapter-13-Unit-testing-Jest/jest.config.js

jest.config.js file gets created.

**Step 3:** To run

npm run test

**Step 4:** Errors

mac@Macs-MacBook-Pro-2 Chapter-13-Unit-testing-Jest % npm run test  
  
> namaste-react-live@1.0.0 test /Users/mac/Documents/Namaste-React/Coding-Assignments/Chapter-13-Unit-testing-Jest  
> jest  
  
● Validation Error:  
  
 Test environment jest-environment-jsdom cannot be found. Make sure the testEnvironment configuration option points to an existing node module.  
  
 Configuration Documentation:  
 https://jestjs.io/docs/configuration  
  
  
As of Jest 28 "jest-environment-jsdom" is no longer shipped by default, make sure to install it separately.

npm i jest-environment-jsdom

mac@Macs-MacBook-Pro-2 Chapter-13-Unit-testing-Jest % npm run test   
  
> namaste-react-live@1.0.0 test /Users/mac/Documents/Namaste-React/Coding-Assignments/Chapter-13-Unit-testing-Jest  
> jest  
  
No tests found, exiting with code 1  
Run with `--passWithNoTests` to exit with code 0  
In /Users/mac/Documents/Namaste-React/Coding-Assignments/Chapter-13-Unit-testing-Jest  
 29 files checked.  
 testMatch: \*\*/\_\_tests\_\_/\*\*/\*.[jt]s?(x), \*\*/?(\*.)+(spec|test).[tj]s?(x) - 0 matches  
 testPathIgnorePatterns: /node\_modules/ - 29 matches  
 testRegex: - 0 matches  
Pattern: - 0 matches

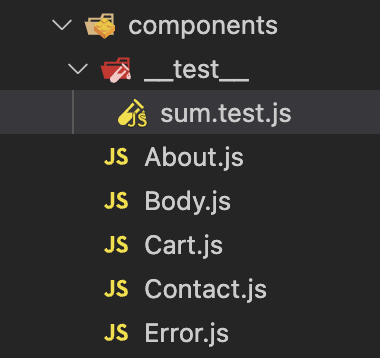
**Step 5:** Create Folder and File for Test

Create a test folder inside components -> \_\_test\_\_

\_\_test\_\_ **(\_\_ is also called dunder; \_\_test\_\_ folder naming convention)**

Whatever file is added inside \_\_test\_\_ folder is considered as test.

-> Inside folder create files -> sum.test.js. **(test.js is file naming convention)**



**Step 6:** Create Test case

components/ \_\_test\_\_ / sum.test.js  
  
  
import {Sum} from '../Sum'  
  
test ('Check sum of 3 positive numbers', () => {  
 expect (Sum(2,5)).toBe(5)  
})

Sum.js  
  
export const Sum = (a,b) => a+b

mac@Macs-MacBook-Pro-2 Chapter-13-Unit-testing-Jest % npm run test  
  
> namaste-react-live@1.0.0 test /Users/mac/Documents/Namaste-React/Coding-Assignments/Chapter-13-Unit-testing-Jest  
> jest  
  
 FAIL src/components/\_\_test\_\_/sum.test.js  
 ● Test suite failed to run  
  
 Jest encountered an unexpected token  
  
 Jest failed to parse a file. This happens e.g. when your code or its dependencies use non-standard JavaScript syntax, or when Jest is not configured to support such syntax.  
  
 Out of the box Jest supports Babel, which will be used to transform your files into valid JS based on your Babel configuration.  
  
 By default "node\_modules" folder is ignored by transformers.  
  
 Here's what you can do:  
 • If you are trying to use ECMAScript Modules, see https://jestjs.io/docs/ecmascript-modules for how to enable it.  
 • If you are trying to use TypeScript, see https://jestjs.io/docs/getting-started#using-typescript  
 • To have some of your "node\_modules" files transformed, you can specify a custom "transformIgnorePatterns" in your config.  
 • If you need a custom transformation specify a "transform" option in your config.  
 • If you simply want to mock your non-JS modules (e.g. binary assets) you can stub them out with the "moduleNameMapper" config option.  
  
 You'll find more details and examples of these config options in the docs:  
 https://jestjs.io/docs/configuration  
 For information about custom transformations, see:  
 https://jestjs.io/docs/code-transformation  
  
 Details:  
  
 /Users/mac/Documents/Namaste-React/Coding-Assignments/Chapter-13-Unit-testing-Jest/src/components/\_\_test\_\_/sum.test.js:1  
 ({"Object.<anonymous>":function(module,exports,require,\_\_dirname,\_\_filename,jest){import { Sum } from '../Sum';  
 ^^^^^^  
  
 SyntaxError: Cannot use import statement outside a module  
  
 at Runtime.createScriptFromCode (node\_modules/jest-runtime/build/index.js:1598:14)  
  
----------|---------|----------|---------|---------|-------------------  
File | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s   
----------|---------|----------|---------|---------|-------------------  
All files | 0 | 0 | 0 | 0 |   
----------|---------|----------|---------|---------|-------------------  
Test Suites: 1 failed, 1 total  
Tests: 0 total  
Snapshots: 0 total  
Time: 0.655 s, estimated 1 s

**Step 7:** Install jest babel

[**Getting Started · Jest**  
*Install Jest using your favorite package manager: Let's get started by writing a test for a hypothetical function that…*jestjs.io](https://jestjs.io/docs/getting-started#using-babel)

npm install --save-dev babel-jest @babel/core @babel/preset-env

mainfolder/ .babelrc  
  
 {  
 "presets": [["@babel/preset-env", {"targets": {"node": "current"}}]]  
 }

If the test case is wrong  
  
mac@Macs-MacBook-Pro-2 Chapter-13-Unit-testing-Jest % npm run test  
  
> namaste-react-live@1.0.0 test /Users/mac/Documents/Namaste-React/Coding-Assignments/Chapter-13-Unit-testing-Jest  
> jest  
  
 FAIL src/components/\_\_test\_\_/sum.test.js  
 ✕ Check sum of 3 positive numbers (1 ms)  
  
 ● Check sum of 3 positive numbers  
  
 TypeError: (0 , \_Sum.Sum) is not a function  
  
 2 |  
 3 | test ('Check sum of 2 positive numbers', () => {  
 > 4 | expect (Sum(2,5)).toBe(5)  
 | ^  
 5 | })  
  
 at Object.<anonymous> (src/components/\_\_test\_\_/sum.test.js:4:16)  
  
----------|---------|----------|---------|---------|-------------------  
File | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s   
----------|---------|----------|---------|---------|-------------------  
All files | 50 | 100 | 0 | 100 |   
 Sum.js | 50 | 100 | 0 | 100 |   
----------|---------|----------|---------|---------|-------------------  
Test Suites: 1 failed, 1 total  
Tests: 1 failed, 1 total  
Snapshots: 0 total  
Time: 0.964 s

If the test case is correct  
  
mac@Macs-MacBook-Pro-2 Chapter-13-Unit-testing-Jest % npm run test  
  
> namaste-react-live@1.0.0 test /Users/mac/Documents/Namaste-React/Coding-Assignments/Chapter-13-Unit-testing-Jest  
> jest  
  
 PASS src/components/\_\_test\_\_/sum.test.js  
 ✓ Check sum of 2 positive numbers (1 ms)  
  
----------|---------|----------|---------|---------|-------------------  
File | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s   
----------|---------|----------|---------|---------|-------------------  
All files | 100 | 100 | 100 | 100 |   
 Sum.js | 100 | 100 | 100 | 100 |   
----------|---------|----------|---------|---------|-------------------  
Test Suites: 1 passed, 1 total  
Tests: 1 passed, 1 total  
Snapshots: 0 total  
Time: 0.899 s, estimated 1 s  
Ran all test suites.

**Step 8:** Add @babel/preset-react Error Fix

import { useState, useContext } from "react";  
import Logo from "../assets/img/foodvilla.png";  
import { Link } from "react-router-dom";  
import useOnline from "../utils/useOnline";  
import UserContext from "../utils/UserContext";  
import { useSelector } from "react-redux";  
  
// SPA - Single Page Application???  
// Client Side Routing  
  
const Title = () => (  
 <a href="/">  
 <img className="h-28 p-2" alt="logo" src={Logo} />  
 </a>  
);  
  
const Header = () => {  
 const [isLoggedIn, setIsLoggedIn] = useState(false);  
  
 const isOnline = useOnline();  
  
 const { user } = useContext(UserContext);  
  
 const cartItems = useSelector((store) => store.cart.items);  
 console.log(cartItems);  
  
 return (  
 <div className="flex justify-between bg-pink-50 shadow-lg sm:bg-blue-50 md:bg-yellow-50">  
 <Title />  
 <div className="nav-items">  
 <ul className="flex py-10">  
 <li className="px-2">  
 <Link to="/">Home</Link>  
 </li>  
  
 <Link to="/about">  
 <li className="px-2">About</li>  
 </Link>  
 <Link to="/contact">  
 <li className="px-2">Contact</li>  
 </Link>  
 <Link to="/instamart">  
 <li className="px-2">Instamart</li>  
 </Link>  
 <Link to="/cart">  
 <li className="px-2">Cart- {cartItems.length} items</li>  
 </Link>  
 </ul>  
 </div>  
 <h1>{isOnline ? "✅" : "🔴"}</h1>  
 <span className="p-10 font-bold text-red-900">{user.name}</span>  
 {isLoggedIn ? (  
 <button onClick={() => setIsLoggedIn(false)}>Logout</button>  
 ) : (  
 <button onClick={() => setIsLoggedIn(true)}>Login</button>  
 )}  
 </div>  
 );  
};  
  
export default Header;

components/ \_\_test\_\_ / Header.test.js  
  
import { render } from "react-dom"  
import Header from "../Header"  
  
test('Logo should load on rendering header', () => {  
//load header  
const header = render(<Header/>)  
console.log("header,",header)  
//check if logo is loaded  
  
})

> namaste-react-live@1.0.0 test /Users/mac/Documents/Namaste-React/Coding-Assignments/Chapter-13-Unit-testing-Jest  
> jest  
  
 FAIL src/components/\_\_test\_\_/Header.test.js  
 ● Test suite failed to run  
  
 Jest encountered an unexpected token  
  
 Jest failed to parse a file. This happens e.g. when your code or its dependencies use non-standard JavaScript syntax, or when Jest is not configured to support such syntax.  
  
 Out of the box Jest supports Babel, which will be used to transform your files into valid JS based on your Babel configuration.  
  
 By default "node\_modules" folder is ignored by transformers.  
  
 Here's what you can do:  
 • If you are trying to use ECMAScript Modules, see https://jestjs.io/docs/ecmascript-modules for how to enable it.  
 • If you are trying to use TypeScript, see https://jestjs.io/docs/getting-started#using-typescript  
 • To have some of your "node\_modules" files transformed, you can specify a custom "transformIgnorePatterns" in your config.  
 • If you need a custom transformation specify a "transform" option in your config.  
 • If you simply want to mock your non-JS modules (e.g. binary assets) you can stub them out with the "moduleNameMapper" config option.  
  
 You'll find more details and examples of these config options in the docs:  
 https://jestjs.io/docs/configuration  
 For information about custom transformations, see:  
 https://jestjs.io/docs/code-transformation  
  
 Details:  
  
 SyntaxError: /Users/mac/Documents/Namaste-React/Coding-Assignments/Chapter-13-Unit-testing-Jest/src/components/\_\_test\_\_/Header.test.js: Support for the experimental syntax 'jsx' isn't currently enabled (6:23):  
  
 4 | test('Logo should load on rendering header', () => {  
 5 | //load header  
 > 6 | const header = render(<Header/>)  
 | ^  
 7 | console.log("header,",header)  
 8 | //check if logo is loaded  
 9 |  
  
 Add @babel/preset-react (https://github.com/babel/babel/tree/main/packages/babel-preset-react) to the 'presets' section of your Babel config to enable transformation.  
 If you want to leave it as-is, add @babel/plugin-syntax-jsx (https://github.com/babel/babel/tree/main/packages/babel-plugin-syntax-jsx) to the 'plugins' section to enable parsing.

To Fix

npm i -D @babel/preset-react

.babelrc   
  
{  
 "presets": [["@babel/preset-env", {"targets": {"node": "current"}}],  
 ["@babel/preset-react", {"runtime": "automatic"}]]  
}  
  
// runtime - place where code is run - make runtime as automatic

Run Again

mac@Macs-MacBook-Pro-2 Chapter-13-Unit-testing-Jest % npm run test  
  
> namaste-react-live@1.0.0 test /Users/mac/Documents/Namaste-React/Coding-Assignments/Chapter-13-Unit-testing-Jest  
> jest  
  
 FAIL src/components/\_\_test\_\_/Header.test.js  
 ● Test suite failed to run  
  
 Jest encountered an unexpected token  
  
 Jest failed to parse a file. This happens e.g. when your code or its dependencies use non-standard JavaScript syntax, or when Jest is not configured to support such syntax.  
  
 Out of the box Jest supports Babel, which will be used to transform your files into valid JS based on your Babel configuration.  
  
 By default "node\_modules" folder is ignored by transformers.  
  
 Here's what you can do:  
 • If you are trying to use ECMAScript Modules, see https://jestjs.io/docs/ecmascript-modules for how to enable it.  
 • If you are trying to use TypeScript, see https://jestjs.io/docs/getting-started#using-typescript  
 • To have some of your "node\_modules" files transformed, you can specify a custom "transformIgnorePatterns" in your config.  
 • If you need a custom transformation specify a "transform" option in your config.  
 • If you simply want to mock your non-JS modules (e.g. binary assets) you can stub them out with the "moduleNameMapper" config option.  
  
 You'll find more details and examples of these config options in the docs:  
 https://jestjs.io/docs/configuration  
 For information about custom transformations, see:  
 https://jestjs.io/docs/code-transformation  
  
 Details:  
  
 SyntaxError: /Users/mac/Documents/Namaste-React/Coding-Assignments/Chapter-13-Unit-testing-Jest/src/assets/img/foodvilla.png: Unexpected character '�'. (1:0)  
  
 > 1 | �PNG  
 | ^  
 2 |   
 3 |   
 4 | IHDR��7~�?�IDATx��w�T��ƹ�  
  
 1 | import { useState, useContext } from "react";  
 > 2 | import Logo from "../assets/img/foodvilla.png";  
 | ^  
 3 | import { Link } from "react-router-dom";  
 4 | import useOnline from "../utils/useOnline";  
 5 | import UserContext from "../utils/UserContext";  
  
 at instantiate (node\_modules/@babel/parser/src/parse-error/credentials.ts:62:21)  
 at instantiate (node\_modules/@babel/parser/src/parse-error.ts:60:12)  
 at Parser.toParseError [as raise] (node\_modules/@babel/parser/src/tokenizer/index.ts:1456:19)  
 at Parser.raise [as getTokenFromCode] (node\_modules/@babel/parser/src/tokenizer/index.ts:1009:16)  
 at Parser.getTokenFromCode [as nextToken] (node\_modules/@babel/parser/src/tokenizer/index.ts:246:10)  
 at Parser.nextToken [as parse] (node\_modules/@babel/parser/src/parser/index.ts:43:10)  
 at parse (node\_modules/@babel/parser/src/index.ts:67:38)  
 at parser (node\_modules/@babel/core/src/parser/index.ts:28:19)  
 at parser.next (<anonymous>)  
 at normalizeFile (node\_modules/@babel/core/src/transformation/normalize-file.ts:52:24)  
 at normalizeFile.next (<anonymous>)  
 at run (node\_modules/@babel/core/src/transformation/index.ts:38:36)  
 at run.next (<anonymous>)  
 at transform (node\_modules/@babel/core/src/transform.ts:29:20)  
 at transform.next (<anonymous>)  
 at evaluateSync (node\_modules/gensync/index.js:251:28)  
 at sync (node\_modules/gensync/index.js:89:14)  
 at fn (node\_modules/@babel/core/src/errors/rewrite-stack-trace.ts:97:14)  
 at transformSync (node\_modules/@babel/core/src/transform.ts:66:52)  
 at ScriptTransformer.\_instrumentFile (node\_modules/@jest/transform/build/ScriptTransformer.js:389:46)  
 at ScriptTransformer.\_buildTransformResult (node\_modules/@jest/transform/build/ScriptTransformer.js:491:33)  
 at ScriptTransformer.transformSource (node\_modules/@jest/transform/build/ScriptTransformer.js:551:17)  
 at ScriptTransformer.\_transformAndBuildScript (node\_modules/@jest/transform/build/ScriptTransformer.js:671:40)  
 at ScriptTransformer.transform (node\_modules/@jest/transform/build/ScriptTransformer.js:723:19)  
 at Object.<anonymous> (src/components/Header.js:2:1)  
 at Object.<anonymous> (src/components/\_\_test\_\_/Header.test.js:2:1)  
  
----------|---------|----------|---------|---------|-------------------  
File | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s   
----------|---------|----------|---------|---------|-------------------  
All files | 0 | 0 | 0 | 0 |   
----------|---------|----------|---------|---------|-------------------  
Test Suites: 1 failed, 1 total  
Tests: 0 total  
Snapshots: 0 total  
Time: 1.228 s  
Ran all test suites.

**Step 9:** png Error Fix

Create dummy image mock

mock/dummyLogo.js  
  
export default "dummy.png";

jest.config.js

Uncomment moduleNameMapper: {}, in jest.config.js

// A map from regular expressions to module names or to arrays of module names that allow to stub out resources with a single module  
 moduleNameMapper: {  
"\\.(jpg|png|svg)" : "../mocks/dummyLogo.js" // any .png take it to dummyLogo.js  
},  
  
 // An array of regexp pattern strings, matched against all module paths before considered 'visible' to the module loader  
 // modulePathIgnorePatterns: [],

Run Again

mac@Macs-MacBook-Pro-2 Chapter-13-Unit-testing-Jest % npm run test  
  
> namaste-react-live@1.0.0 test /Users/mac/Documents/Namaste-React/Coding-Assignments/Chapter-13-Unit-testing-Jest  
> jest  
  
 console.error  
 Warning: ReactDOM.render is no longer supported in React 18. Use createRoot instead. Until you switch to the new API, your app will behave as if it's running React 17. Learn more: https://reactjs.org/link/switch-to-createroot  
  
 4 | test('Logo should load on rendering header', () => {  
 5 | //load header  
 > 6 | const header = render(<Header/>)  
 | ^  
 7 | console.log("header,",header)  
 8 | //check if logo is loaded  
 9 |  
  
 at printWarning (node\_modules/react-dom/cjs/react-dom.development.js:86:30)  
 at error (node\_modules/react-dom/cjs/react-dom.development.js:60:7)  
 at render (node\_modules/react-dom/cjs/react-dom.development.js:29670:5)  
 at Object.<anonymous> (src/components/\_\_test\_\_/Header.test.js:6:22)  
  
 FAIL src/components/\_\_test\_\_/Header.test.js  
 ✕ Logo should load on rendering header (48 ms)  
  
 ● Logo should load on rendering header  
  
 Target container is not a DOM element.  
  
 4 | test('Logo should load on rendering header', () => {  
 5 | //load header  
 > 6 | const header = render(<Header/>)  
 | ^  
 7 | console.log("header,",header)  
 8 | //check if logo is loaded  
 9 |  
  
 at render (node\_modules/react-dom/cjs/react-dom.development.js:29674:11)  
 at Object.<anonymous> (src/components/\_\_test\_\_/Header.test.js:6:22)

**Step 10:** Provider Fix

import { render } from "react-dom"  
import Header from "../Header"  
import { Provider } from "react-redux"  
import store from "../../utils/store"  
  
test('Logo should load on rendering header', () => {  
//load header  
const header = render(<Provider store={store}><Header/></Provider>)  
console.log("header,",header)  
//check if logo is loaded  
  
})

Run Again

mac@Macs-MacBook-Pro-2 Chapter-13-Unit-testing-Jest % npm run test  
  
> namaste-react-live@1.0.0 test /Users/mac/Documents/Namaste-React/Coding-Assignments/Chapter-13-Unit-testing-Jest  
> jest  
  
 console.error  
 Warning: ReactDOM.render is no longer supported in React 18. Use createRoot instead. Until you switch to the new API, your app will behave as if it's running React 17. Learn more: https://reactjs.org/link/switch-to-createroot  
  
 6 | test('Logo should load on rendering header', () => {  
 7 | //load header  
 > 8 | const header = render(<Provider store={store}><Header/></Provider>)  
 | ^  
 9 | console.log("header,",header)  
 10 | //check if logo is loaded  
 11 |  
  
 at printWarning (node\_modules/react-dom/cjs/react-dom.development.js:86:30)  
 at error (node\_modules/react-dom/cjs/react-dom.development.js:60:7)  
 at render (node\_modules/react-dom/cjs/react-dom.development.js:29670:5)  
 at Object.<anonymous> (src/components/\_\_test\_\_/Header.test.js:8:22)  
  
 FAIL src/components/\_\_test\_\_/Header.test.js  
 ✕ Logo should load on rendering header (46 ms)  
  
 ● Logo should load on rendering header  
  
 Target container is not a DOM element.  
  
 6 | test('Logo should load on rendering header', () => {  
 7 | //load header  
 > 8 | const header = render(<Provider store={store}><Header/></Provider>)  
 | ^  
 9 | console.log("header,",header)  
 10 | //check if logo is loaded  
 11 |  
  
 at render (node\_modules/react-dom/cjs/react-dom.development.js:29674:11)  
 at Object.<anonymous> (src/components/\_\_test\_\_/Header.test.js:8:22)  
  
-----------------|---------|----------|---------|---------|-------------------  
File | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s   
-----------------|---------|----------|---------|---------|-------------------  
All files | 24.24 | 0 | 0 | 25 |   
 components | 16.66 | 0 | 0 | 18.18 |   
 Header.js | 16.66 | 0 | 0 | 18.18 | 12,18-55   
 mocks | 0 | 0 | 0 | 0 |   
 dummyLogo.js | 0 | 0 | 0 | 0 |   
 utils | 28.57 | 100 | 0 | 28.57 |   
 UserContext.js | 100 | 100 | 100 | 100 |   
 cartSlice.js | 40 | 100 | 0 | 40 | 10-16   
 store.js | 100 | 100 | 100 | 100 |   
 useOnline.js | 7.69 | 100 | 0 | 7.69 | 4-23   
-----------------|---------|----------|---------|---------|-------------------  
Test Suites: 1 failed, 1 total  
Tests: 1 failed, 1 total  
Snapshots: 0 total  
Time: 1.269 s

**Step 11:** Provider Fix

Add StaticRouter

import { render } from "react-dom"  
import Header from "../Header"  
import { Provider } from "react-redux"  
import store from "../../utils/store"  
import { StaticRouter } from "react-router-router-dom/server"  
  
  
test('Logo should load on rendering header', () => {  
//load header  
const header = render(  
<StaticRouter>  
<Provider store={store}>  
 <Header/>  
</Provider>  
</StaticRouter>  
)  
console.log("header,",header)  
  
const logo = header.getAllByTestId("logo")  
console.log("logo,",logo[0])  
expect(logo[0].src).toBe("dummyLogo.png")  
//check if logo is loaded  
  
})

Run Again

No errors

jest data-testid