Testing React Components

Daniel Stern
CODE WHISPERER

@danieljackstern

What Does Testing React Components Mean?

Verify output has not regressed

Ensure that rarely occurring corner cases produce the correct output

If component generates side effects, verify they occur but do not execute them

Verify user interactions are handled as expected

Constructing Testable React Components

A Spectrum of React Components

Components may or may not have lifecycle handlers

Components may or may not have internal state

Components may or may not generate side effects

Components may get state from arguments, or from external dependencies

Building Testable React Components

No internal state – output is an idempotent product of the props that are provided

No side-effects – any AJAX calls, UI changes or other side effects are handled by sagas, thunks, etc., but not by components

No lifecycle hooks – fetching data is handled on the application level, not the component level

React Redux and Jest: A Fine Pair

Components don't generate side effects

Component consists of logical display and container components

Components do not have internal state

Testing React Redux Components

Test Container and Display elements separately

Use unit tests to verify methods and properties passed by container are accurate

Use snapshot tests to verify the output of the display component, passing props in directly

Demo

Add a snapshot test to the Display element of a React Redux component

Add unit tests to the Container element of the same component

Note how, combined, the two tests provide excellent coverage

Enzyme vs. React Test Renderer

“Neglecting to broaden their view has kept some people doing one thing all their lives.”

– **Napoleon Hill**

React Test Renderer vs. Enzyme

React Test Renderer	Enzyme
Takes a React component and outputs the resulting HTML without a DOM	Takes a React component and outputs the resulting HTML without a DOM
From the React team	From an unrelated, but respected team (AirBnB)
Useful for getting the output HTML of a component for snapshot testing	Useful for testing a variety of interactions including click, keyboard input, and more
Recommended by the Jest Team	Has a variety of open bugs which make using it a challenge

Testing Stateful React Components

Mock dependencies, then test them

Use spies to verify side-effects

Move logic from lifecycle to services

Prevent regression with snapshots

Inject values by writing mocks for services

Make stateless components, where possible

Demo

Create a simple, stateful React component

Create a manual mock for the component's dependency

Write assertions

Set up snapshot tests

Summary

React-Test-Renderer is recommended over Enzyme

Testing stateless React components is simple and easy

- Container and display can be tested separately

Testing stateful React components is tiresome and complex

- Side effects must be verified by mocking called API
- Services must be specially mocked to coax component into desired state

Coming up in the Next Module…

What Are Matchers?

To Be, or Not to Be, That Is the Matcher

Great Expectations