Snapshot Testing

Daniel Stern

CODE WHISPERER

@danieljackstern

> “A good snapshot stops a moment from running away.”
>
> – Eudora Welty

What Is a Snapshot?

JSON-based record of a component's output

Compared to component's actual output during testing process

Committed along with other modules and tests to the application repo

```
import renderer from 'react-test-renderer';
import { MyComponent } from './MyComponent‘

const tree = renderer
    .create(
        <MyComponent title="The meaning of life"/>
    );

expect(tree.toJSON()).toMatchSnapshot();
```

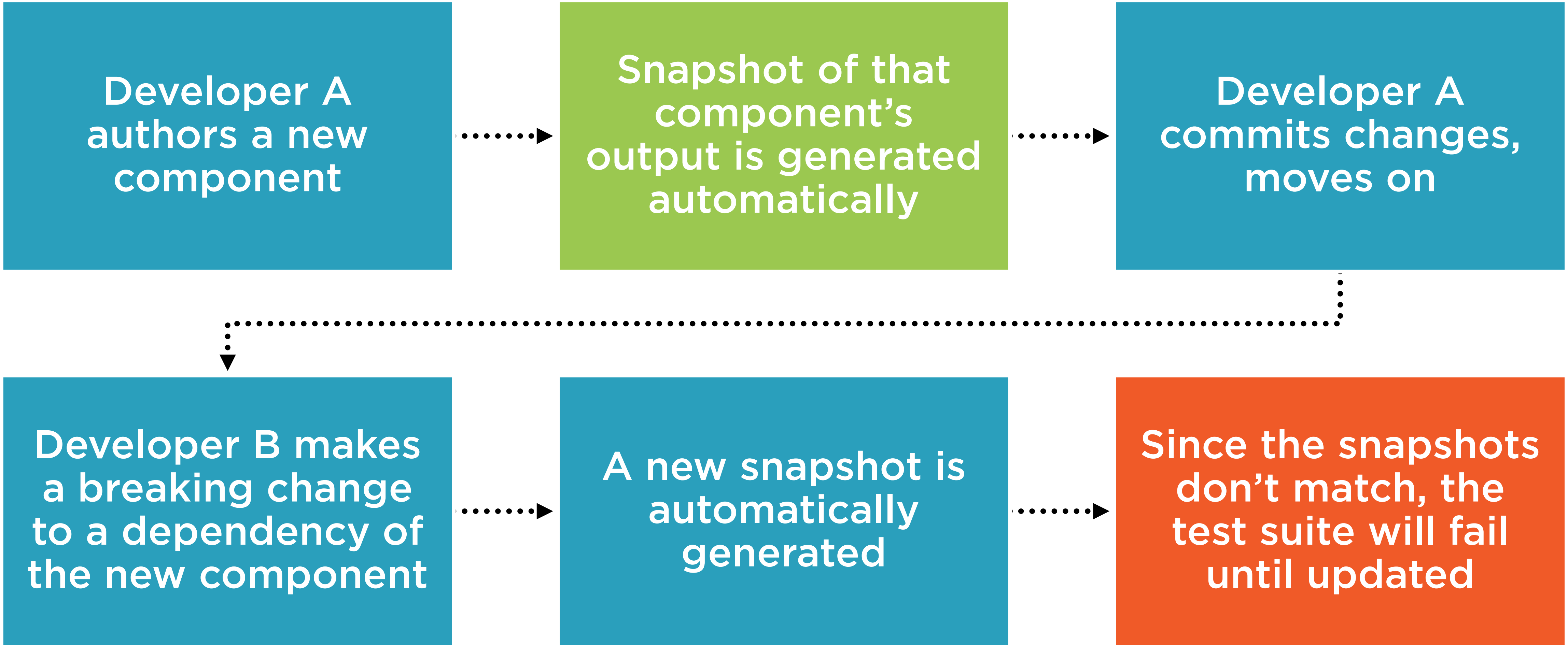
How Snapshot Testing Works

First, HTML output is generated with React.

Note: *react-test-renderer* is used instead of *react-dom, enzyme* is not used

The first time *toMatchSnapshot()* is called, a snapshot is created
Each subsequent time, the new snapshot is compared with the old one

The Snapshot Testing Process



Developer A authors a new component

Snapshot of that component’s output is generated automatically

Developer A commits changes, moves on

Developer B makes a breaking change to a dependency of the new component

A new snapshot is automatically generated

Since the snapshots don’t match, the test suite will fail until updated

Demo

Add a snapshot test to a component

Note that the test will fail on an insignificant change to the component

Advantages and Disadvantages of Snapshot Testing

Advantages

Fast and automatic

Catches regressions humans may miss

Works nicely with libraries that take in state and output HTML components (React, Angular, Vue)

Adds some protection against regression when no time is available for manually writing tests

Requires little training or knowledge of testing to use

Disadvantages

Easy to ignore and suppress

Protects only against regression

If a component is working incorrectly and then is fixed, a snapshot test will say it is now broken

Adds extra files to an already crowded repo

Sensitive to incidental changes

A waste of resources, if component is certain to be modified in near future

Updating Snapshots After Components Have Changed

"... Nor all the drowsy syrups of the world
Shall ever medicine thee to that sweet sleep,
Which thou owedst yesterday."

– William Shakespeare

Updating Snapshots After Components Have Changed

Tests must be run with --*update* flag

Old snapshots will be replaced with image of current output

Using *--update* without careful consideration diminishes the value of snapshots

Demo

Note that snapshot test fails after any change to the tested component

Use the command line to update existing snapshots

Note that the snapshot test no longer fails

Summary

Snapshots are a fast and convenient means of preventing regression

Most of the work is done automatically, only what and when needs to be specified

Failing snapshots can be resolved by updating them

Snapshots do not guarantee correct functionality, only prevent regression

Coming up in the Next Module…

Understanding Testing React Components

A Stateless State of Affairs

React Redux Revue

Digesting Enzyme