

SEP 6, 2016 • RANDY COULMAN • posted in TDD , javascript , ruby

Facebook recently released v15 of their Jest testing framework. Version 15 makes a number of changes that have got people taking another look at it.

A screenshot of a tweet from Kent C. Dodds (@kentcdodds). The tweet text is "Totally digging this snapshot thing from Jest! Woot! pic.twitter.com/iPqj7QhL9a". It shows 12:17 AM - Sep 4, 2016, 64 likes, and a reply button. The tweet is highlighted with a blue border.

The basic idea is that you can write a test like this (example copied from the [Jest v14 announcement blog post](#)):

In particular, I noted:

The biggest problem with manual verification is that it requires a human in the loop. This slows down the process, and is also potentially error-prone. What if the only human available is not as experienced at looking at the output and misses something critical?

What if we need to rush a hot-fix into production? That's the time when any human involved is likely to be the most stressed and rushed. That's not when we're likely to do our best at looking carefully at changed output.

The ability to quickly update a snapshot when it changes makes for a nicer, faster workflow. But it also makes it easy to accept new snapshots that have problems.

Indeed Christoph Pojer, one of the main contributors to Jest, recently warned of this danger.



Another potential issue with snapshot testing is that it can make for fragile tests. For example, a test might be concerned with only a few details of the captured output, but the test will fail when **any** detail of the captured output changes.

Finally, when writing Snapshot Tests that are intended to stick around, you need to be careful to write really solid test descriptions, because the body of the test no longer communicates anything about what's important about the test case.

SO YOU'RE SAYING I SHOULDN'T USE IT

I'm not saying that you should never use Snapshot Testing. I'm certainly not saying that it isn't a worthwhile feature to have in Jest.

If it helps you write valuable tests that you wouldn't write otherwise, I'm all for it.

There are contexts in which Golden Master Testing is the best tool for the job. The most notable example is when [testing legacy code](#).

With legacy code, especially when writing [characterization tests](#), you care more about whether the output has changed and less about whether it's correct.

In those contexts, having good tools to help with the process is essential, and Jest's Snapshot Testing feature is a great tool that is really well-executed.

What I am saying is that you should be aware of the potential dangers and downsides of Golden Master Testing and choose wisely.

CONCLUSION

It is definitely worth experimenting with Jest's Snapshot Testing feature. See what it's good at and what it's not so good at.

Pay attention to whether future you can understand the tests when you come back to them. Notice when you get test failures for unimportant details. See if you or your teammates accidentally update snapshots that shouldn't have been updated.

But also pay attention to what benefits you gain from it. Can you now test something you didn't know how to test before? Does your development cycle speed up, both short-term and long-term? Most importantly, is your software better at delighting your users?

As with any development tool or approach, keep your eyes open and think about the consequences, both good and bad, of your decisions.

[Tweet](#)

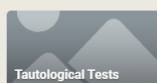
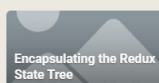
[« Relative Requires in JavaScript](#)

[Encapsulating the Redux State Tree »](#)

RECEIVE NEW POSTS BY E-MAIL

COMMENTS

ALSO ON [RANDY COULMAN](#)



6 years ago · 14 Comments

In mobile applications, it is sometimes necessary to store information securely ...

7 years ago · 10 Comments

In a Redux application, the bulk of the application's data is stored as a "state tree" ...

6 years ago · 16 Comments

When we're writing automated tests for our code, it's easy to ...

7 years ago · 10 Comments

This post is Part 1 of a series about functional programming ...

6 Comments

1 Login

G

Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS

D

f


t

G

Name

Share

BestNewestOldest



vanwilson

7 years ago


Good, balanced article, Nicholas. Funny how everything old (golden master testing) is new again.

I'm curious how this would work with a CI server. It seems like you'd have to run the tests locally to update the snapshots, then check those into version control so that the CI server could use the latest "known good" snapshots. But then, what's the point of a CI process, if you can make it pass just by force-updating the snapshots locally?

Am I being too cynical? I suppose CI would still catch lazy devs who don't bother to run tests locally at all.

00

ReplyShare



Randy Coulman

7 years ago

Thanks!


The proper workflow is, as you say, to check the snapshots into version control and have CI run against those.

The "force-update without thinking/paying attention" problem is one of my big concerns here. But I think that having a good, strong code review culture to go with snapshot tests can help.

Ben McCormick wrote a really nice post about how to use snapshot testing well at <http://benmccormick.org/201....>

00

ReplyShare



Nicolas lensen


7 years ago

Great article, Randy.

I would also add the fact that snapshots doesn't allow the developer to practice TDD, since you must write your code first to be able to run the test and generate the first snapshot for the feature you are working on.

00

ReplyShare




Randy Coulman

7 years ago

Thanks, Nicolas! And yes, I agree with you about TDD.

00

ReplyShare




John Backes

6 years ago

wouldn't the TDD approach be to manually write the snapshot, then test the output against the snapshot?

130

ReplyShare



Randy Coulman

6 years ago

Yes, if you were going to try to use snapshots as part of TDD. To me, that feels like using the wrong tool for the job. Using TDD, I normally write more targeted assertions against the (shallow) rendered component rather than testing the entire thing like a snapshot does.

00

ReplyShare

Subscribe

Privacy

Do Not Sell My Data

DISQUS

Courageous Software

Copyright © 2013 - 2023 Randy Coulman

Privacy Policy

randycoulman

randycoulman

randycoulman

Randy Coulman's blog on writing software well, and helping others to do the same.