Testing React Applications, v2

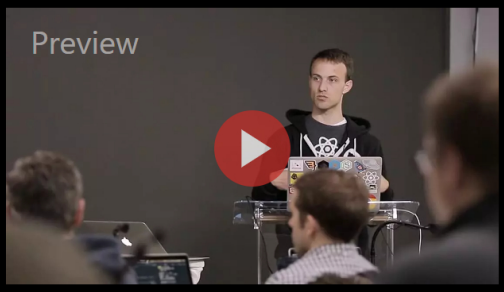
Introducing Snapshot Testing

Topics: React • Testing

Kent C. Dodds
Professional Trainer

This course is no longer being maintained. We recommend the testing section of the Intermediate React course instead. We now recommend you take the Intermediate React, v5 course.

Check out a free preview of the full Testing React Applications, v2 course:



The "Introducing Snapshot Testing" Lesson is part of the full, Testing React Applications, v2 course featured in this preview video. Here's what you'd learn in this lesson:

Kent introduces snapshot testing, which easily compares values of anything over time and keeps the tests updated. Kent takes questions from students.

Transcript from the "Introducing Snapshot Testing" Lesson

[00:00:02]
>> Kent C. Dodds: So we're actually moving on to the next section, but we're staying in the same login component for this next section. It's going to be a short section about snapshot testing. How many people are using Jest snapshots? Okay, a couple of you, cool. Looks like I broke a test.

[00:00:17] I'm not going to worry about fixing it. So I'm going to reset everything. So I'm back to square one. So snapshots are kind of nifty. So I'll come back here in just a sec, but I'm gonna give you a little demo of what snapshots really are. So let's see, it is just expect and test.

[00:00:37] [COUGH] Sorry. Okay, so here it's other/test-expect/_tests_/expect-assertions. You don't have to follow along if you don't want to, but that's where I am. If you want to reference some of these expectations that Jest provides in its assertion library, you can look at these and reference these. But down here toward the bottom we have snapshots.

[00:01:03] So here is what I like to call a manual snapshot, and actually let me run this test:expect. Okay,
>> Kent C. Dodds: So here we're getting these flying superheroes from this superheroes file I'll show you in a sec. And we're expecting that those equal this array of these superheroes who can fly.

[00:01:32] I'm pretty sure I got these from the Incredibles. So yeah, fun stuff. Okay, so here's our superheroes file. We have this data with superheroes. And then we discover, my, Jack-Jack, he can do everything. He's the coolest, shape shifting, and flying, and all kinds of cool stuff. So we're gonna save our file and now we have a failing test, right?

[00:01:55] Because now there's a new superhero, he can fly, and we see that very clearly in the output. You see the snapshot, no, sorry, one second, let me just .only this. You're seeing spoiler alert, okay. Here we go, this is the manual snapshot. So we expected to equal this stuff, but we got Jack-Jack, and it'll show us very clearly here that object.

[00:02:24] And so it's pretty easy. And this is how I write tests like this. When I wrote this test, this is how I wrote it, and this is how I update them too. Console.log(flyingHeroes), save. And go to my console.log, copy that and then I will go here and thank my lucky stars for prettier.

[00:02:42] And then I get my test passing, all right? Ignore this stuff. [SOUND] So that is a manual snapshot. And that's effectively what snapshots are doing. So it's a mechanism by which we can take some value, serialize it into a string, and then compare it in the future for our future test runs.

[00:03:10] And so yeah, I used to do this all the time. I'd have, this is the data, and it's gonna look like this, and it changed, console.log, copy paste, and that's how I do these assertions. Well, here's what a snapshot test looks like.
>> Kent C. Dodds: It's very similar, except instead of two equal and having the data right there, you call this toMatchSnapshot.

[00:03:33] And let me add this one.
>> Kent C. Dodds: And this is what it looks like for a failure. Does that look familiar? Looks pretty much exactly the same as the last one. Except to update this snapshot, you just hit the U key and, sorry, ignore that output cuz it's distracting from what we're doing.

[00:03:55] But what it does is it'll go into this _snapshots_ directory and create a file that has the same name as the test file. And inside of that, it's actually a Javascript file that has exports and then the name of the test and some other information that you can provide in the to match snapshot API.

[00:04:15] And then it shows a serialized version, but it's pretty printed, so it says, hey, this is an array, this is an object, and this is the value. And then every single call to match snapshot, it will serialize that value and then compare it with the one that it saved.

[00:04:33]
>> Kent C. Dodds: Did that make sense? Any questions about that?
>> [INAUDIBLE]
>> Kent C. Dodds: Sorry.
>> Speaker 2: I'm wondering if, it seems like the U key is a magical make your test pass button. So is there a danger to that?
>> Kent C. Dodds: Yeah, let's chat about that after I get through the rest of this.

[00:04:54] But that's a great point. Don't let me forget.
>> Speaker 2: So this automatic snapshot, so what are we achieving on an underground [INAUDIBLE] because we're only telling I'm getting a result and I'm matching with another snapshot that is already available.
>> Kent C. Dodds: Mm-hm.
>> Speaker 2: So I mean, instead of I'm telling that this is what I expect, I am getting stored data in some where or how exactly we're doing it?

[00:05:20]
>> Kent C. Dodds: Well, so you'll commit these snapshots to your source control, and then everybody who pulls this down, they're gonna have these same snapshots that they'll compare against every single time they run the test. And so any time there's a change to these snapshots you updat it, you commit that and it gets, as part of any pull request or anything.

[00:05:40] That's one of the benefits of the snapshot, it's a really, really easy way to see how your changes are impacting some output. It's a really easy way to do some really bad things, too, and we'll talk about those in a little bit. But yeah, that's kinda the value there.

[00:05:57]
>> Speaker 3: So if I [INAUDIBLE], let's say to model, I'm not going to have a name property.
>> Kent C. Dodds: Mm-hm.
>> Speaker 3: I'll just come to snapshot file, remove the names so that my test case will run.

>> Kent C. Dodds: Yeah, yeah, so you could do that. But part of the benefit of using a snapshot rather than putting things in line here is that what you do is you just change the code and then it will say, hey, your snapshot is failing.

[00:06:23] And then you can just press the U key and it will update your snapshot automatically. Yeah, which is a blessing and a curse, but it's really cool. Okay, so these objects are not the only things that you can snapshot. You can snapshot a lot more things. So here, I'm gonna just get rid of all this stuff now and we'll run them all.

[00:06:47] So here's an object. You can have any mix of anything. You can have a regex, it will serialize that, so on and so forth. Let's just look at what the snapshot looks like for that. So it will serialize the date, it's my birthday. It will serialize an error with the error message.

[00:07:06] It will serialize a map object, a bigger object, a regex set, and then what's super duper awesome is it will even serialize dom nodes. And so here we created this dom node. We expect that to match a snapshot. So you can serialize here or you can snapshot your container.

[00:07:26] I do container.firstchild because the container is always gonna have a div, so whatever I'm rendering. And so here we have a button, and and it formats it nicely so that if there's a change here where it adds another prop, it'll only be a change on a single line rather than being this really weird, like if everything was on one line and anything changed in that line is busted, so it formats it nicely for you.

[00:07:53] You can serialize something that was created with, let's see, React.createElement, and so you can serialize a React element, yeah, this one, and sterilizing those will actually show you the props of their past, which is an implementation detail in my mind. So I never do this, but that's just something to be aware of.

[00:08:17] Actually if you're using in time and you try to snapshot a mounted component, then that will also show you the props that were passed, and that's an implementation detail so I never do that either. Serialized just the HMTL, cuz the user doesn't care about the props. And then yeah, we also have R3eact test renderer.

[00:08:39] You can serialize those and then things rendered with React on, which is what React testing library is.
>> Kent C. Dodds: And you can also give a title for your snapshot like, where did I do that? Yeah, right here. And that will give you more helpful information in the area output and in the snapshot file itself.

[00:09:02] So one other place that I don't have an example here, but I'll show you just because I think it's interesting, is the, let's see, probably Babel, here, let's do, import all.macro. So I create several Babel plugins. I think they're a lot of fun, and snapshots are just the best way to test a Babel plugin.

[00:09:29] And so I have this thing called Babel testing library that manages this for me. But here's what the snapshots look like when you're using the Babel testing library. It'll show you the source code and then it'll have these little arrows and says that source code, when using this plugin, will transpile down to this result.

[00:09:54] Yeah, because anything else would just be total insanity. So Babel plugins are a really awesome place to use snapshots. Also, if you are running a command line interface tool, a CLI, the help output from that is also a good place to use a snapshot. So it's not just for Reacts.

[00:10:14] I know that's kind of where it took off. But there are lots of really great use cases for snapshots.