

2025, 30_{Oct} = 24_{Dec}



I LOVED SNAPSHOTS IN 2016

A Decade Later,
Let's Revisit



ROBIN POKORNY



23 Feb 2017  [video](#)

HannoverJS Feb 2017 ➤

A hand holds a Polaroid photograph of the Roblox logo, which is a white icon on a red background. The background of the entire image is a vibrant, abstract painting with large, bold strokes in green, yellow, and blue, featuring a stylized white figure. In the bottom right corner, the text '@robloxindonesia' is visible.

17 Oct 2016

BerlinJS Nov 2016 ➤

30 May 2016 [video](#) [slides](#)

React Berlin #9 ➤

Snapshot testing

▼ History

Version	Changes
v23.4.0	Snapshot testing is no longer experimental.
v22.3.0	Added in: v22.3.0

Snapshot tests allow arbitrary values to be serialized into string values and compared against a set of known good values. The known values are stored in a snapshot file. Snapshot files are managed by the test runner, but are designed to be human readable to aid in debugging. Snapshot files should be checked into source control along with your test files.

Snapshot files are generated by starting Node.js with the `--test-update-snapshots` command-line flag. A separate snapshot file is generated for each test file. The snapshot file has the same name as the test file with a `.snapshot` file extension. This behavior can be configured using the `snapshotOptions` property. Each snapshot assertion corresponds to an export in the snapshot file.

An example snapshot test is shown below. The first time this test is executed, it will fail because the corresponding snapshot file does not exist.

```
// test.js
suite('suite of snapshot tests', () => {
```



Contents lists available at [ScienceDirect](#)

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss



New Trends and Ideas

Snapshot testing in practice: Benefits and drawbacks[☆]

Victor Pezzi Gazzinelli Cruz^{a,*}, Henrique Rocha^b, Marco Tulio Valente^a

^a Department of Computer Science, Federal University of Minas Gerais (UFMG), Belo Horizonte, Brazil

^b Department of Computer Science, Loyola University Maryland, Baltimore, United States



ARTICLE INFO

Article history:

Received 30 November 2022

Received in revised form 21 May 2023

Accepted 28 June 2023

Available online 7 July 2023

Dataset link: [Snapshot Testing: Benefits and Drawbacks \(Original data\)](#)

Keywords:

Snapshot testing

Software testing

React

Jest

ABSTRACT

In recent years, snapshot testing has gained attention as a novel testing technique. Even though it is used by well-known companies, there is a gap in the scientific literature concerning the tradeoffs of this testing technique. Therefore, in this paper, we investigate the adoption of snapshot testing in practice through a grey literature review, aiming to reveal tradeoffs, best practices, and tools commented by practitioners. Our findings show that snapshots are simple to create and can help in preventing regressions; however, they may swiftly become fragile if used improperly. We also envision an opportunity for further research on snapshot testing, for example by mining and analyzing project samples from public repositories.

© 2023 Elsevier Inc. All rights reserved.

TONIGHT



01

WHAT IS
SNAPSHOT
TESTING



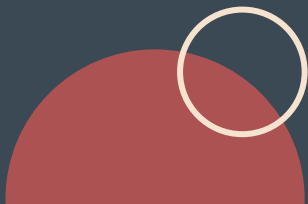
02

WHAT WENT
WRONG



03

WHAT
NOW?



01

WHAT IS SNAPSHOT TESTING



{JSON}



{JSON}



{JSON}



{JSON}



{JSON}



```
expect(json).toMatchSnapshot();
```

```
./utils.spec.js
```

```
./__snapshots__/utils.spec.js.snap
```

```
jest --watch
```

Snapshot name: `snapshots snapshot 1`

- Snapshot - 2

+ Received + 1

@@ -1,14 +1,13 @@

{

"bool": true,

- "date": 2025-10-30T13:40:02.592Z,

+ "date": 2025-10-30T13:51:06.027Z,

"func": [Function],

"name": "f",

> 1 snapshot failed.

Snapshot Summary

> 1 snapshot failed from 1 test suite. Inspect your code changes or press `u` to update them.

Test Suites: 1 failed, 1 total

Tests: 1 failed, 19 skipped, 20 total

Snapshots: 1 failed, 1 total

Time: 0.302 s, estimated 1 s

Ran all test suites.

TDD

×

SNAPSHOTS

algorithms

write before

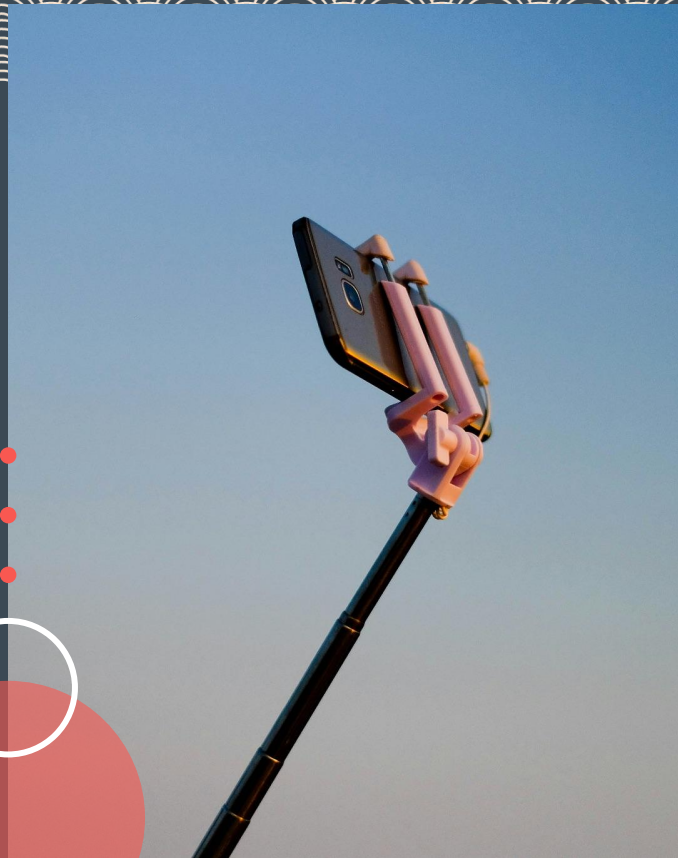
part

structures

concurrent or after

whole

inside codebase



02

WHAT
WENT
WRONG



COURAGEOUS SOFTWARE

Do Our Best; Make Our Best Better; Help Others

[Home](#)[Blog](#)[Categories](#)[Archives](#)[About](#)

SNAPSHOT TESTING: USE WITH CARE

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

Snapshot Testing has been getting a fair bit of attention recently with some new tool support that makes it easy to use. But is that a good thing?

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.

This has brought increased attention to a feature that was [introduced in version 14](#), Snapshot Testing.



Contents lists available at [ScienceDirect](#)

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss



New Trends and Ideas

Snapshot testing in practice: Benefits and drawbacks[☆]

Victor Pezzi Gazzinelli Cruz^{a,*}, Henrique Rocha^b, Marco Tulio Valente^a

^a Department of Computer Science, Federal University of Minas Gerais (UFMG), Belo Horizonte, Brazil

^b Department of Computer Science, Loyola University Maryland, Baltimore, United States



ARTICLE INFO

Article history:

Received 30 November 2022

Received in revised form 21 May 2023

Accepted 28 June 2023

Available online 7 July 2023

Dataset link: [Snapshot Testing: Benefits and Drawbacks \(Original data\)](#)

Keywords:

Snapshot testing

Software testing

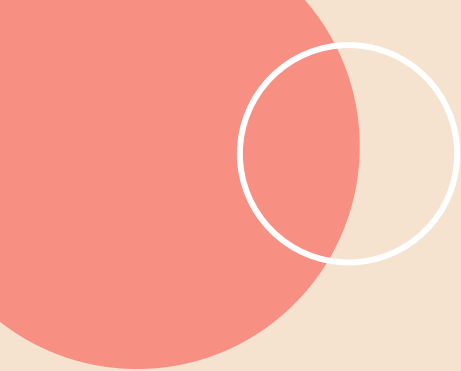
React

Jest

ABSTRACT

In recent years, snapshot testing has gained attention as a novel testing technique. Even though it is used by well-known companies, there is a gap in the scientific literature concerning the tradeoffs of this testing technique. Therefore, in this paper, we investigate the adoption of snapshot testing in practice through a grey literature review, aiming to reveal tradeoffs, best practices, and tools commented by practitioners. Our findings show that snapshots are simple to create and can help in preventing regressions; however, they may swiftly become fragile if used improperly. We also envision an opportunity for further research on snapshot testing, for example by mining and analyzing project samples from public repositories.

© 2023 Elsevier Inc. All rights reserved.



‘The most common
drawback is
the **fragility**
of snapshot tests’

—STIP PAPER



LACK OF DEVELOPER INTENTION

Tests don't encode what's important

GENERATED FILE PROBLEM

Need manual verification

LARGE SNAPSHOTS

Developers don't scrutinize them

HIGH FALSE NEGATIVES

Tests fail when code is actually fine



Justin Searls

@searls

Follow



I 📄 'd snapshot testing yesterday. Someone DM'd (not @'d!) me a question so here's my response to why. I'll blog soon [twitter.com/searls/status/...](https://twitter.com/searls/status/931111111111111111)

Thoughts on snapshot testing

Takes on snapshot testing schemes to follow. There are numerous categories of failures surrounding snapshot testing. Most of this is informed by my experience in 2008-2011 when QA teams thought Selenium RC record playback scripts were a panacea, but I've seen the same thing with tools like VCR in Ruby, HTML fixtures in JS tests, and other attempts at "easy" controls over API & DB dependencies:

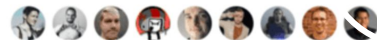
1. They are tests you don't understand, so when they fail, you don't usually understand why or how to fix it. That means you have to do true/false negative analysis & then suffer indirection as you debug how to resolve the issue
2. Good tests encode the developer's `.intention.`, they don't only look in the test's behavior without editorialization of what's important and why. Snapshot tests lack (or at least, fail to encourage) expressing the author's intent as to what the code does (much less why)
3. They are generated files, and developers tend to be undisciplined about scrutinizing generated files before committing them, if not at first then definitely over time. Most developers, upon seeing a snapshot test fail, will sooner just nuke the snapshot and record a fresh passing one instead of agonizing over what broke it.
4. Because they're more integrated and try to serialize an incomplete system (e.g. one with some kind of side effects: from browser/library/runtime versions to environment to database/API changes), they will tend to have high false-negatives (failing test for which the production code is actually fine and the test just needs to be changed). False negatives quickly erode the team's trust in a test to actually find bugs and instead come to be seen as a chore on a checklist they need to satisfy before they can move on to the next thing.

These four things lead to a near total loss in the intended utility of integrated/functional tests: as the code changes make sure nothing is broken.

Instead, when the code changes, the test will surely fail, but determining whether and what is actually "broken" by that failure is the more painful path than simply re-recording & committing a fresh snapshot. (After all, it's not like the past snapshot was well understood or carefully expressed authorial intent.) As a result, if a snapshot test fails because some intended behavior disappeared, then there's little stated intention describing it and we'd much rather regenerate the file than spend a lot of time agonizing over how to get the same test green again.

9:03 AM - 15 Oct 2017

40 Retweets 100 Likes



TDD

×

SNAPSHOTS

algorithms

write before

part

structures

concurrent or after

whole

inside codebase

03

WHAT
NOW?



Kent C. Dodds

[Blog](#)

[Courses](#)

[Discord](#)

[Chats](#)

[Calls](#)

[Workshops](#)

[About](#)

Effective Snapshot Testing

October 30th, 2017 — 10 min read





USE

- API response structures (not UI)
- Error messages and CLI output for developer-facing tools
- Code transformations
- Small, stable components in design systems
- Legacy refactoring

AVOID

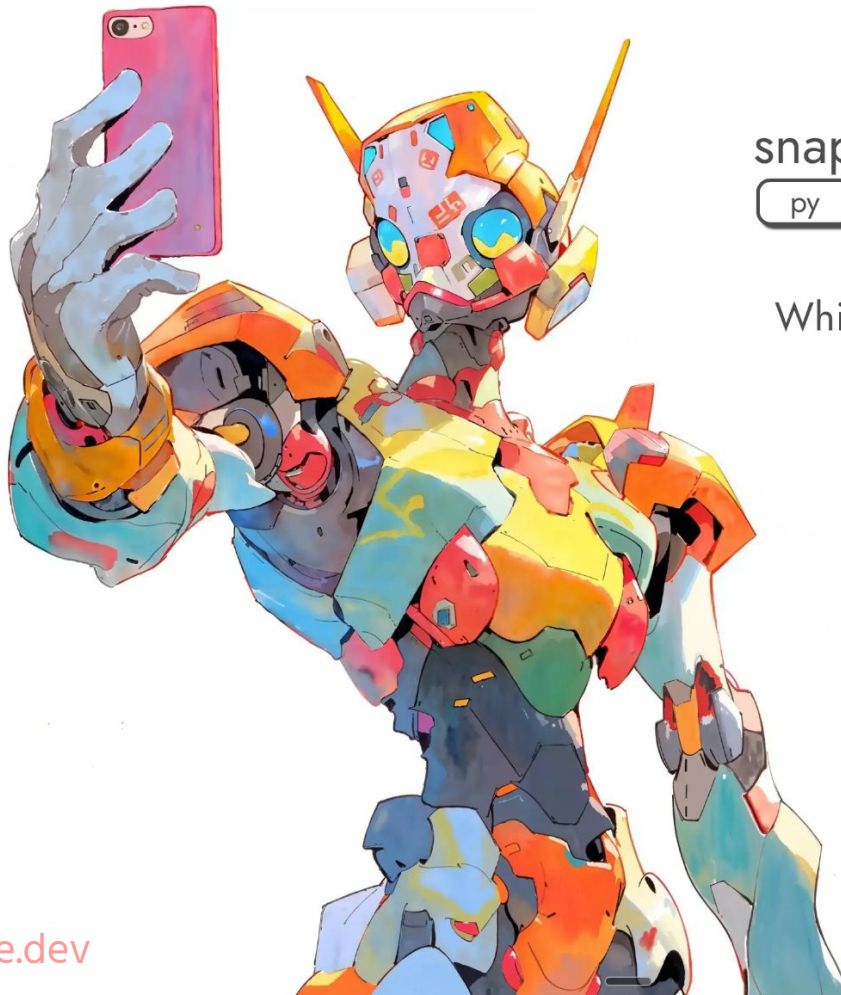
- Large component trees
- Components with frequent changes
- "It renders" smoke tests (use linting instead)
- Testing behavior
- As primary testing strategy





```
describe(`Podcast API`, () => {
  test(`responds with FrontKec episode`, () => {
    const episodeDTO = getEpisode(`frontkec-04`)

    expect(episodeDTO).toMatchInlineSnapshot(
      `
      {
        id: expect.any(String),
        publishedAt: expect.any(Date),
        downloadUrl: expect.stringContaining('.mp3'),
        listens: expect.any(Number),
      },
    `,
      {
        "downloadUrl": StringContaining ".mp3",
        "hosts": [
          "Martin Michálek",
          "Robin Pokorný",
        ],
        "id": Any<String>,
        "listens": Any<Number>,
        "podcast": "FrontKec"
        "publishedAt": Any<Date>,
        "title": "Jirka Helmich: AI vs product management a firmy odtržené od zákazníka",
      }
    ,
  )
})
})
```



selfie

snapshot testing for

py

jvm

js

...

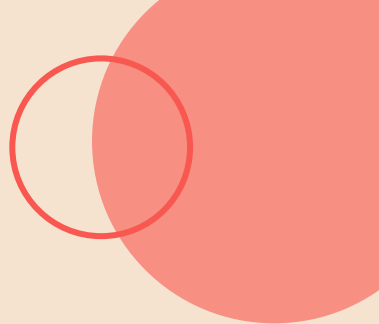
Which is literal, lensable
and like a filesystem

get started

Snapshot testing is the
fastest and most precise
mechanism to record
and specify the
behavior of your
system and its
components.

Robots are
writing their
own code. Are
you still writing
assertions by
hand?

<https://selfie.dev>



TAKEAWAY





WEB

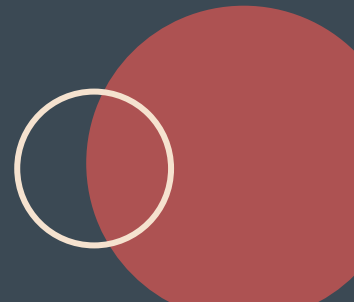
me@robinpokorny.com

NAME

IN/IG/YT



Let's connect





STATE OF JS



<https://survey.devographics.com/survey/state-of-js/2025>

