Verifying Vue Components' Functionality with Jest

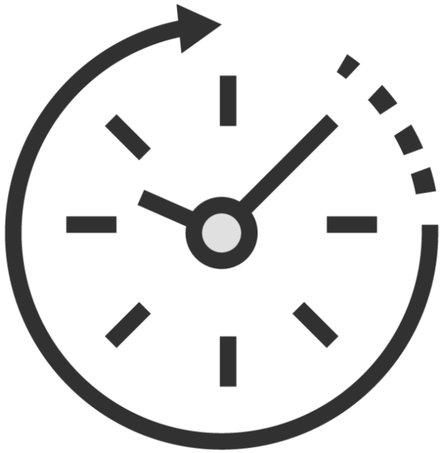
Daniel Stern
The Code Whisperer

http://danielstern.ca/social-media

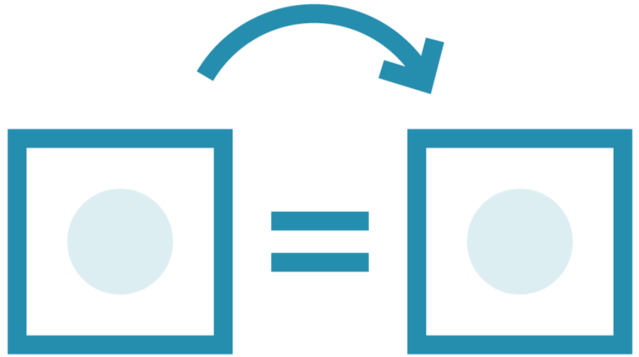
Verifying Vue Components' Functionality with Jest

Learning Objectives

Basic tests to catch standard exceptions



Asynchronous tests



Mock external API

Why Test Vue Components with Jest?

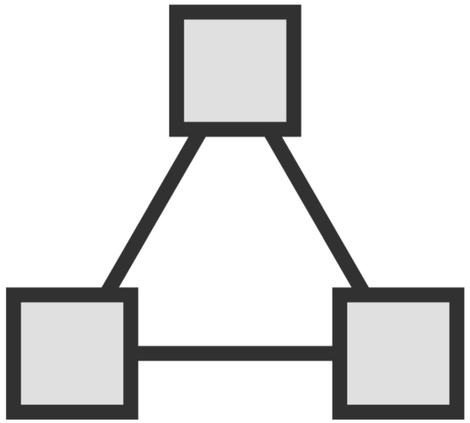
Adding Jest tests to a Vue application saves money and time.

Reduce QA time

Fosters cooperation between developers

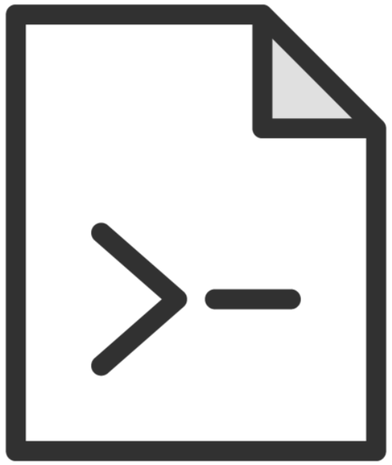
Reduced key-person risk

Testing Vue Components with Jest: Workflow

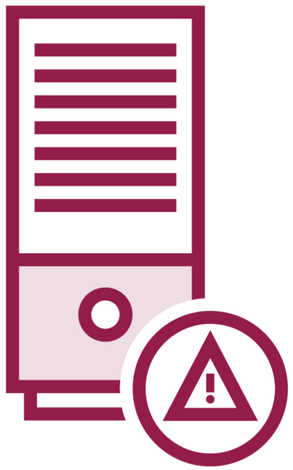


Modular element structure

Exception tests



Tests integrated with project



Tests integrated with CI

Applying Exception Tests

What Is an Exception Test?

Method with narrow focus

Contains assertions

Error if any assertions don’t match

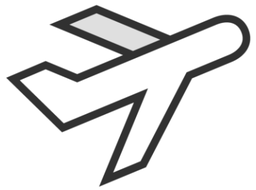
```
let a = 42;
expect(a).toBeGreaterThan(41);
expect(a).toBeLessThan(43);
expect(a).toBeCloseTo(42.0000001);

expect(a).toBeTruthy();
expect(a).not.toBeNaN();
```

What Are Assertions?

Assertion: method which throws an error if two values don’t match.

Common assertion format: expect(x).Y(z).

How Does Unit / Exception Testing Work?



One or more tests per suite and suites per application

Tests verify functioning of component or service

All tests invoked with single CLI script – script errors upon failure

```
describe(`the accounting API`) {

  it(`should return a numeric price`,
  function(){
    let price = accountAPI.getPrice();
    expect(price).toBeGreaterThan(0);
  }
}
```

- Suite explains what all child tests relate to
- Test has clear title
- Verify price is a positive number
- Test is brief
- Its job done, the test ends

Snapshot Tests

What Are Snapshot Tests?

Vue element is rendered as HTML and saved

Subsequent tests fail if HTML has changed

When change is intentional, snapshot is replaced

Snapshot tests are simple.

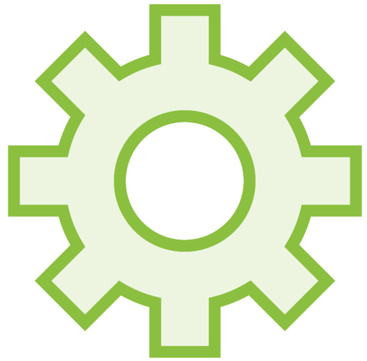
Exception tests verify that something works as expected.

Snapshot tests verify only that it has not changed.

Test Setup and Teardown with Jest

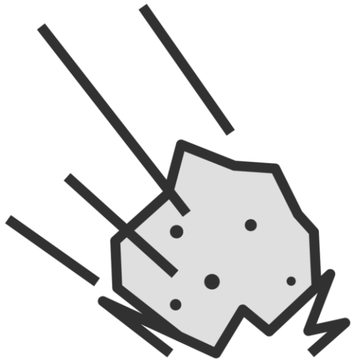
Test Setup and Teardown with Jest

Helper methods are easier to troubleshoot than conducting setup and teardown within each test.



beforeEach

Timing: Prior
Usage: Initialization



afterEach

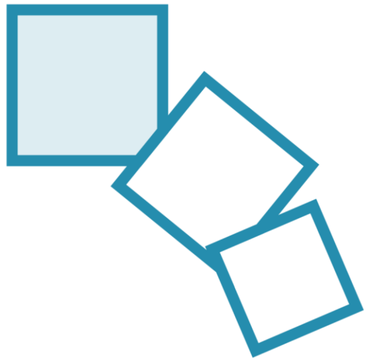
Timing: After
Usage: Cleanup

beforeAll / afterAll

Usage: Suite-wide setup and teardown

Using @vue/test-utils

These testing utilities allow components to be tested in relative isolation.

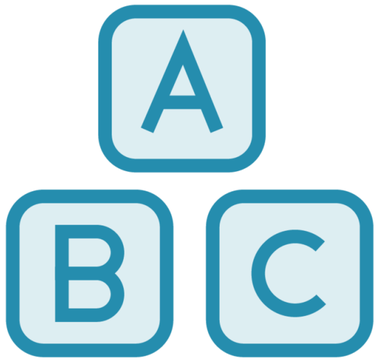


render()

Renders without app

renderToString()

Render, but as HTML



shallowMount()

Render, but omit children

Testing Vue Components for Exceptions

Demo

Use @vue/test-utils **to mount chat component**

- **Generate output without app**

Generate exceptions if component does not display correctly

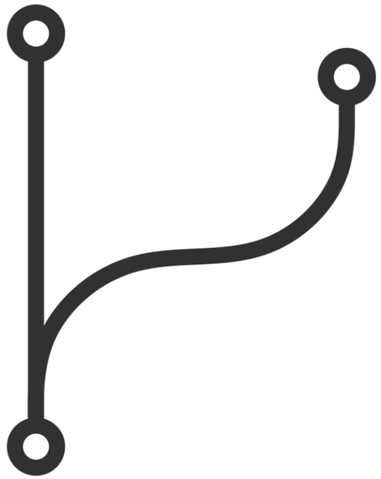
- **Given props are fixed**
- **Count mapped elements**
- **Spot check HTML contents**

Use Snapshot testing to generate additional coverage

Asynchronous Testing with Jest

When Should You Use Asynchronous Tests?

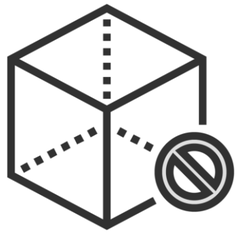
Speed is a priority – prefer synchronous



Async tests for tests which can’t be synchronous

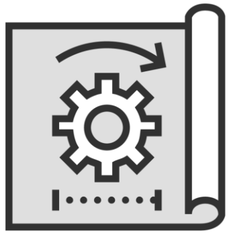
Tests that take a long time to execute are usually an anti-pattern.

Asynchronous Testing with Jest



Tests fail if an exception is thrown *before* the test ends

Non-async tests end after one computing cycle



Async tests can last many cycles, allow delayed assertions

An asynchronous test will fail even if an assertion fails late.

Testing Asynchronous Methods in Vue Applications

Demo

Create a test for Message Service

- **Note synchronous test does not work with promise**

Modify test so it is asynchronous

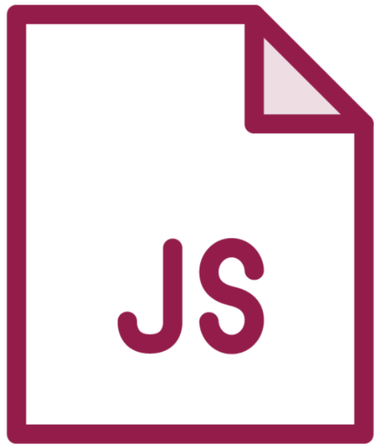
Test late values for exceptions

Mocking External Dependencies

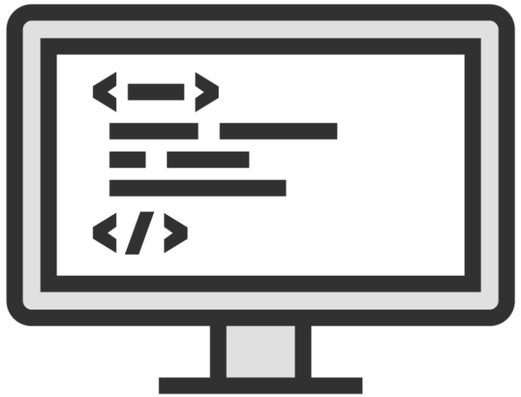
“More dependencies, more problems.”

- A wise developer who is still employed, even to this day.

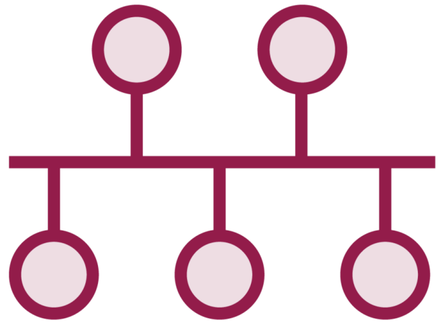
What Are Mocks?



Service substitute with same API



Minimal logic



Custom return values

Understanding Dependencies

Internal Dependencies

Code needed by app

Part of the project

Can be edited

Can add testing features

Can design synchronous interface

External dependencies

Code needed by app

Not part of the project

Cannot be edited (except by making them internal)

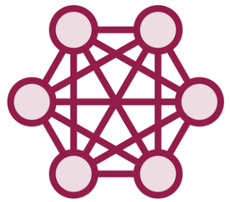
Can be wrapped with support for testing features

May require asynchronous tests

“He will fence with his own shadow.”

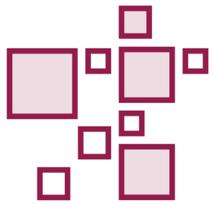
– **William Shakespeare, The Merchant of Venice**

Mocking Dependencies



Broken dependency = broken component

A flawless component with a broken dependency: still broken



Mocked dependencies: no chance of broken dependency

Failing tests + no unmocked dependencies = flawed component

Correctly Structuring Vue Applications for Testing

Correct Structuring

Pass action creators to components

Simple methods to components for event handling

Pure functions everywhere

Incorrect Structuring

Provide services as globals

import **services in component files**

Inject entire service

Use asynchronous methods

Use random methods

Use methods depending on external factors (date, time, OS)

Module Summary

Exception tests throw an error when values do not match

Async tests are required for delayed exceptions

External dependencies make testing hard

- **Mocking ensures failed assertions are not caused by dependencies**