Unit Testing With Jest

Introduction to unit testing in JavaScript

Ezz Abuzaid - Senior Software Engineer @Kortext 24/12/2022

Agenda

- Testing Introduction.
- Types of testing.
- What is Unit Testing.
- Test Case & Test Suite.
- Pros & Cons
- Attributes of unit testing.
- Anti Patterns.
- Why?
- Two Different Kind Of Testing.
- TDD?
- Should I always write test?

Types Of Testing

- Unit tests.
- Integration tests.
- End-to-end tests.
- Performance testing.
- Smoke testing.
- **Regression testing**

What is Testing?

- Testing is the process of making sure that a system meets certain predefined requirements, by executing it either manualy or by using an automation tool.
- The process of identifying defects and errors before shipping a system to the users.

Requirements?

A requirement can be as high level as "A user should be able to order a meal and pay for it from within the app itself" or low as "When a user clicks on "Add to cart" button the user should receive a notification.

Requirements can range to different levels based on the department (Marketing, Sales, Compliance) or based on a speciality (Architect, Product Owner, Developer)

But why testing is essential?

- To deliver working software to the users.
- Cumlativlty build trust in the users.
- More resilient (recoverable) and robust software.

Hint: We refer to the thing that is being executed as the "{Thing} Under Test", so in case of testing a System as whole we say, "System Under Test" or if we're taking about unit test then we say, "Unit Under Test".

What is Unit testing?

A unit is the smallest part of a software. it can as small as function or a class.

Unit testing is the process of making sure the unit under test does what expected.

Unit testing is the process of detecting defects in the unit under test.

Example

```
function Counter() {
    const [counter, setCounter] = useState(0);
    return (
        <>
            <div>{counter}</div>
            <button onClick={() => setCounter(counter + 1)}>Increase</button>
        </>
    );
}
```

Test Suite & Test Case

Test Suite is group of related test cases that speak about a functionallity or behaviour.

Test Case is the setup, invokation, and assertion of the unit under test, it should do one thing: assert single output or verify one behaviour

- Test Suite can be defined using the `describe` function
- Test Case can be defined using the `it` or `test` function

Test Report

Test Report should clearly tell what is being tested and under which cirumstance. The test report should be simple and strightforward. Anyone with fair bit of knoweldge in the team should be able to understand the test suite and test case from its report.

Attributes of unit testing

- It should be fast.
- It should be small.
- It should be isolated.

Pros & Cons

Pros

- Reduce techincal dept.
- Boost developer confidance in doing changes.
- Reduce regressions.

Cons

- It requires more time **initially**.
- Maintanace.
- Hard to agree on standards.
- Often, no direct result.

Techincal dept

Techincal dept is the cost of choosing the fast way to develop a feature. where if you spend a bit more time to refine your work you might reduce that techincal dept

Quick Example

```
function Counter(props: readonly { startAt: number, increaseBy: number } = {startAt:0, increaseBy:1}) {
    const [counter, setCounter] = useState(props.startAt);
    return (
        <>
            <div data-testid="counter-label">{counter}</div>
            <button data-testid="increase-button" onClick={() => setCounter(counter + props.increaseBy)}>Increase</butt
        </>
    );
}

import userEvent from "@testing-library/user-event";
import { render, screen } from "@testing-library/react";
import "@testing-library/jest-dom";
it("increases the counter on click by 1", () => {
    // ARRANGE
    render(<Counter startAt={0} />);
    const expected = 1;

    // ACT
    await userEvent.click(screen.getByTestId("increase-button"));

    // Assert
    expect(screen.getByTestId("counter-label")).toEqual(expected);
});
```

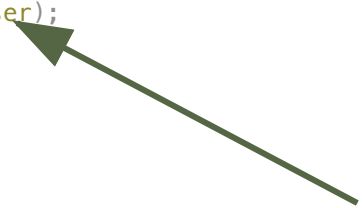
it should sum up a set of numbers

Code

Use code snippets and get the highlighting directly![1]

```
interface User {
    id: number;
    firstName: string;
    lastName: string;
    role: string;
}

function updateUser(id: number, update: User) {
    const user = getUser(id);
    const newUser = { ...user, ...update };
    saveUser(id, newUser);
}
```



Learn More

Components

You can use Vue components directly inside your slides.

We have provided a few built-in components like `<Tweet/>` and `<Youtube/>` that you can use directly. And adding your custom components is also super easy.

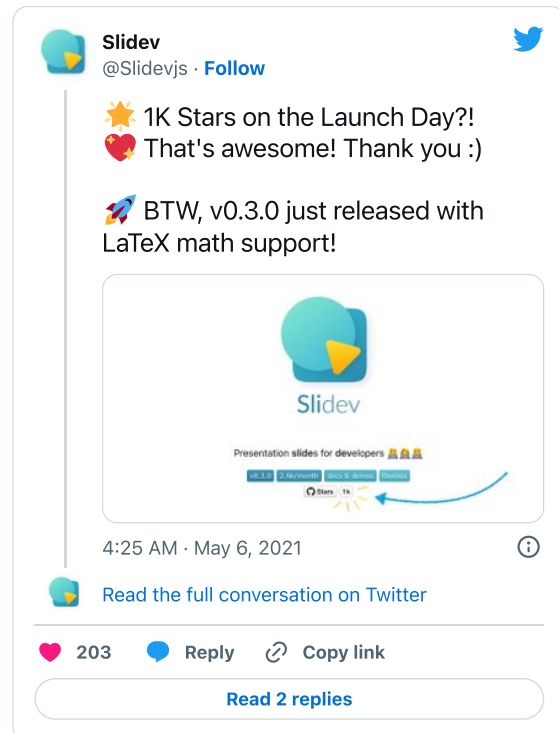
```
<Counter :count="10" />
```

<Counter :count="10" />

Check out the guides for more.

```
<Tweet id="1390115482657726468" />
```

<Tweet id="1390115482657726468" />



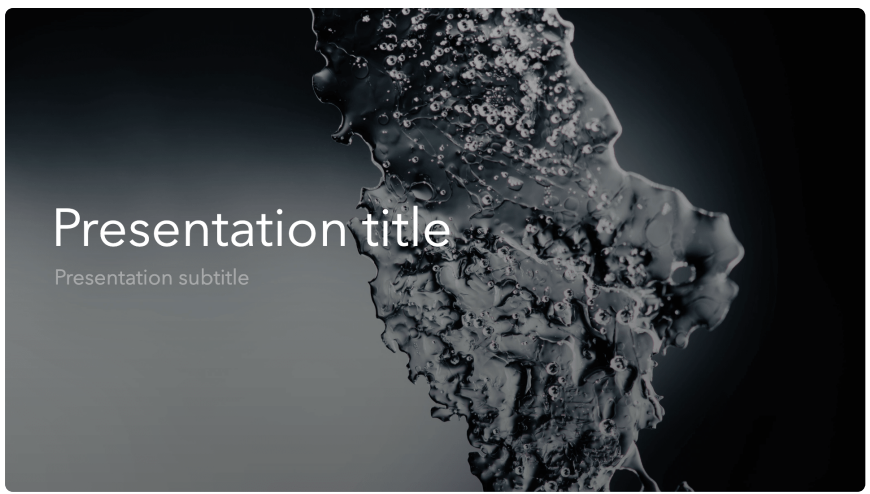
class: px-20

Themes

Slidev comes with powerful theming support. Themes can provide styles, layouts, components, or even configurations for tools. Switching between themes by just **one edit** in your frontmatter:

```
---
theme: default
---
```

```
---
theme: seriph
---
```



Read more about How to use a theme and check out the Awesome Themes Gallery.

preload: false

Animations

Animations are powered by @vueuse/motion.

```
<div v-motion :initial="{ x: -80 }" :enter="{ x: 0 }">Slidev</div>
```

Post Talk

- Testing categories

    Funtional testing.

    Non functional testing.