Testing library  

From scared 😱 to confident 💪 - Benefits of user centric testing

By Daniel Wolmerud

2020-04-22

Developer & Consultant
at **DEVTRACK**

10 years □□□
Backend & Frontend

Current client is
Discovery Networks
working with
streaming services
for
Dplay, Eurosport
and more

github.com/dwolmerud
linkedin.com/in/daniel-wolmerud

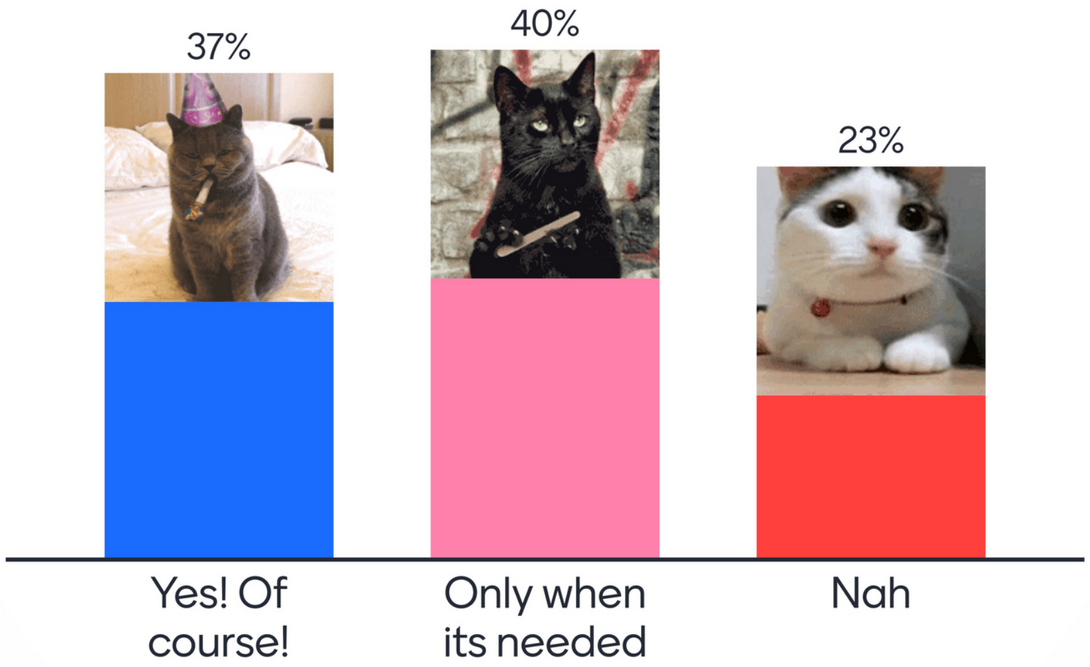
Quiz time!

Go to menti.com and use the code 82 01 51

or -->

Do you write tests?

Mentimeter



30

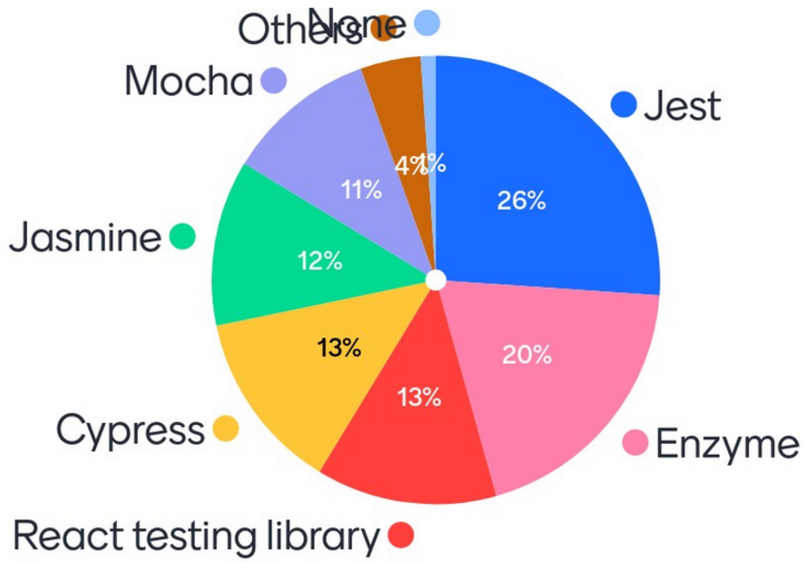
Why do we write test?

Mentimeter

27

What test related libraries/frameworks have you used?

Mentimeter



28

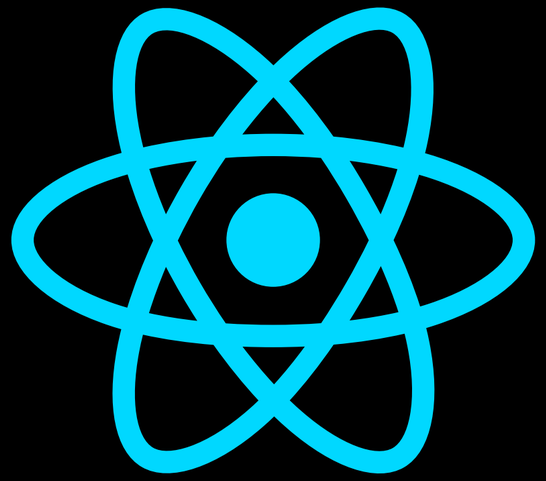
Why do we test?

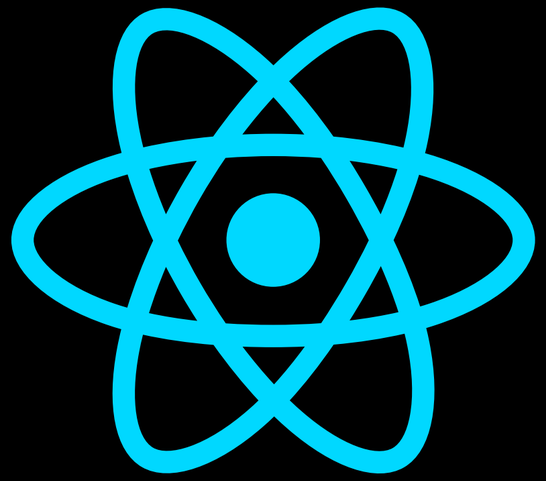
Confidence

Code Design

Documentation


Power tool!

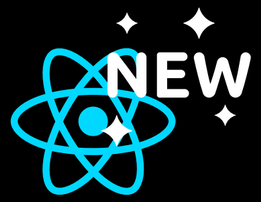




A few Jest and Enzyme tests
No static type checking

😟

Rewrite from
Angular to React



New
modules


```
npm install --save-dev @testing-library/react
```

testing-library.com

THE FOUR TYPES OF TESTS

End to End

A helper robot that behaves like a user to click around the app and verify that it functions correctly.

Sometimes called “functional testing” or e2e.

Integration

Verify that several units work together in harmony.

Unit

Verify that individual, isolated parts work as expected.

Static

Catch typos and type errors as you write the code.

testingjavascript.com

Testing Library

Simple testing utilities that encourage good testing practices

Query the DOM for nodes in a way that's similar
to how the user finds elements on the page

Tests only break when your app breaks,
not implementation details

WHY?

Recommended by Facebook

Testing user behavior rather than implementation details

More stable tests that do not break when refactoring or gives false positives due to shallow-rendering

Best practices

Make your tests resilient to change (with good selectors)

Test a few reality scenarios

Test higher up the tree

youtube.com/watch?v=Fha2bVoC8SE

```
render(<User name="Daniel" />)
```

findBy*

getBy*

queryBy*

ByText
ByLabelText
ByPlaceholderText
ByAltText
ByTitle
ByDisplayValue
ByRole
ByTestId

fireEvent

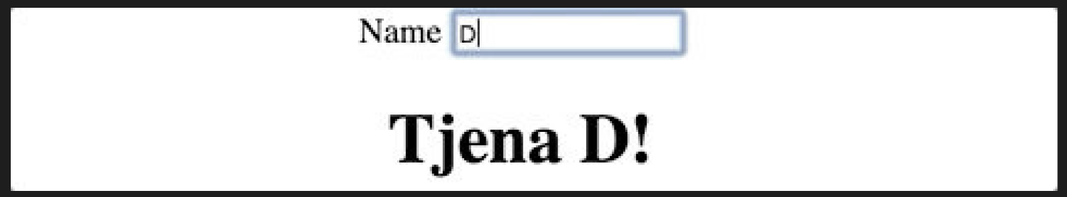
expect

```jsx
import React, { useState } from "react";
import "./user.css";

export default function User({ name: initName = "" }) {
  const [name, setName] = useState(initName);

  return (
    <div className="container">
      <div>
        <label htmlFor="name">Name</label>

        <input
          type="text"
          label="Name"
          aria-label="Name"
          onChange={(e) => setName(e.target.value)}
          value={name}
          placeholder="Fill in your name"
        />
      </div>

      {name && name !== "" && <h1>Greetings {name}!</h1>}
    </div>
  );
}
```

render()
getByText()


```
import React from "react";
import { render } from "@testing-library/react";
import User from "./User";

test("shows name in greeting header", () => {
  const { getByText } = render(<User name="Daniel" />);
  getByText("Greetings Daniel!");
});
```

getByTestId()




```
import React from "react";
import { render } from "@testing-library/react";
import User from "./User";

test("shows name in greeting header", () => {
  const { getByText } = render(<User name="Daniel" />);
  getByText("Greetings Daniel!");
});
```

```
 {name && name !== "" && (
   <h1 data-testid="greeting-message">Greetings {name}!</h1>
 )}

it("shows name in greeting header in Swedish", () => {
  const { getByTestId } = render(<User name="Daniel" />);
  const header = getByTestId("greeting-message");
  expect(header).toBeInTheDocument();
});
```

queryByText()
debug()

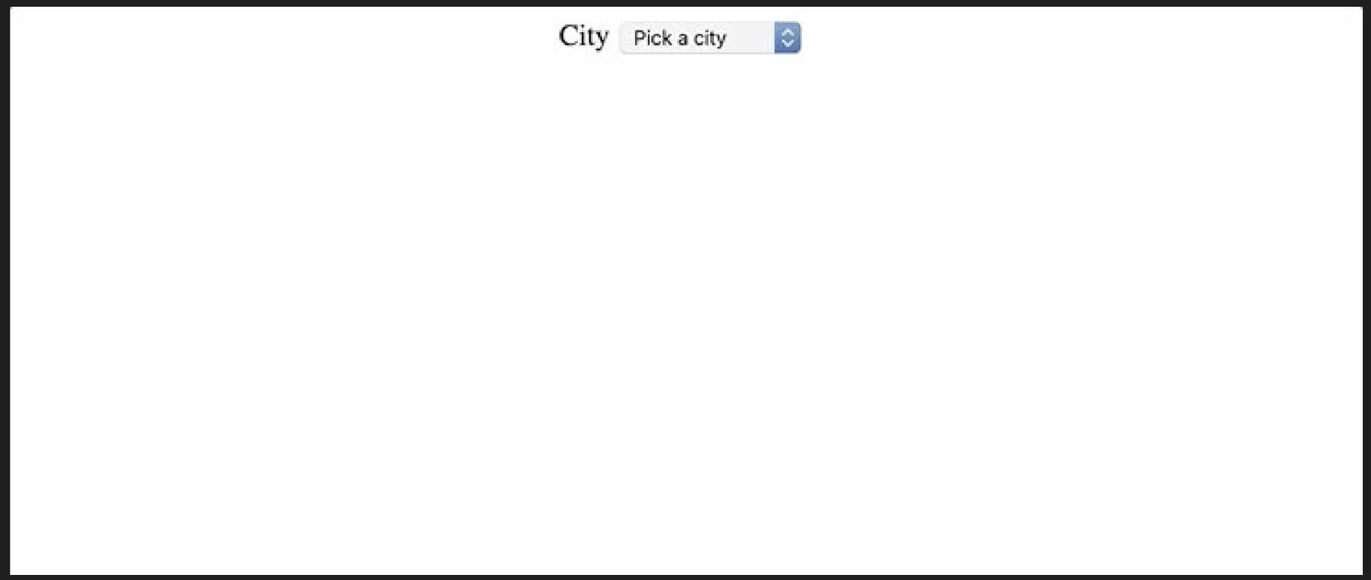
```
it("shows no greeting if no name is passed", () => {
  const { queryByText, debug } = render(<User />);
  debug();
  expect(queryByText("Greetings")).not.toBeInTheDocument();
});
```

```
PASS src/User/User.test.js
  User component
    ✓ shows no greeting if no name is passed (5ms)

  console.log node_modules/@testing-library/react/dist/pure.js:94
    <body>
      <div>
        <div
          class="container"
        >
          <div>
            <label
              for="name"
            >
              Name
            </label>
            <input
              aria-label="Name"
              label="Name"
              placeholder="Fill in your name"
              type="text"
              value=""
            />
          </div>

        </div>
      </div>
    </body>
```

getByLabelText()
getByText()
fireEvent()


```
it("shows a greeting message when typing", () => {
  const { getByLabelText, getByText } = render(<User />);

  const input = getByLabelText("Name");

  fireEvent.change(input, { target: { value: "Daniel" } });

  const header = getByText("Greetings Daniel!");

  expect(header).toBeInTheDocument();
});
```

async findByTestId()



DEMO

github.com/dwolmerud/react-meetup-testing-lib

render with providers

```
import React from "react";
import { render } from "@testing-library/react";
import { createStore } from "redux";
import { Provider as ReduxProvider } from "react-redux";
import { createHistory } from "@reach/router";
import { reducer } from "../state";

export function renderWithProviders(
  ui,
  {
    initialState = {},
    store = createStore(reducer, initialState),
    history = createHistory(),
    pattern,
  } = {}
) {
  const ProviderWrappedUI = () => (
    <ReduxProvider store={store}>
      <ThemeProvider theme={theme}>
        <MuiPickersUtilsProvider>
          <Router history={history}>
            <Route path={pattern}>
              {ui}
            </Route>
          </Router>
        </MuiPickersUtilsProvider>
      </ThemeProvider>
    </ReduxProvider>
  );
  return { ...render(<ProviderWrappedUI />), store, history };
}
```

Questions?

Thank you!

linkedin.com/in/daniel-wolmerud
github.com/dwolmerud