TESTY JEDNOSTKOWE

Hello

Kamil Richert

Engineering Manager at Atlassian

Rodzaje testów

- **Testy jednostkowe**

  Jest, Karma

- **Testy integracyjne**

  Cypress

- **Testy poziomu end-to-end**

  Selenium

Testowanie jednostkowe

Testowanie jednostkowe pomaga w zapewnieniu jakości kodu i zwiększa pewność, że nasza aplikacja działa poprawnie.

Korzyści testowania jednostkowego:

- Wczesne wykrywanie błędów
- Ułatwia refaktoryzację
- Zwiększa zaufanie do kodu
- Wspiera rozwój zespołowy

JEST

JEST

- Framework do testowania rozwijany przez Facebook'a
- Posiada własny test-runner → program do uruchamiania testów
- Posiada własny zbiór asercji (porównań którymi weryfikujemy wyniki testów)
- Daje nam możliwość mierzenia pokrycia testami
- Wbudowane mocki
- Obsługuje wyjątki (te celowe, które chcemy wytestować)
- Można nim testować projekty frontend'owe (używające np Angular'a, Vue czy React'a), czy projekty oparte o Node.js

Przydatne linki:

https://jestjs.io/docs/en/getting-started → cała sekcja "Introduction" to podstawy pracy z Jest
https://create-react-app.dev/docs/running-tests/ → testowanie aplikacji korzystających z create-react-app, gdzie Jest jest wykorzystany jako test runner i główną bibliotekę do testów.

Przykładowy test

```
// math.test.js
const { sum, subtract } = require('./math');

test('sum adds two numbers correctly', () => {
  expect(sum(2, 3)).toBe(5);
});

test('subtract subtracts two numbers correctly', () => {
  expect(subtract(5, 2)).toBe(3);
});
```

Asynchroniczne testowanie

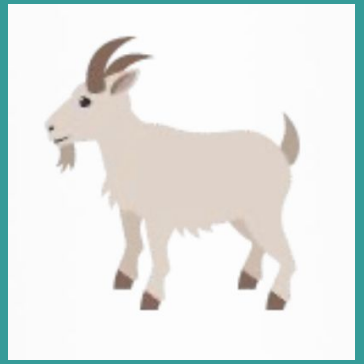
Często mamy do czynienia z asynchronicznymi operacjami w aplikacjach React, takimi jak pobieranie danych z API.

```
test('the data is peanut butter', () => {
 return fetchData().then(data => {
   expect(data).toBe('peanut butter');
 });
});

test('the data is peanut butter', () => {
 return expect(fetchData()).resolves.toBe('peanut butter');
});
```

Więcej: https://jestjs.io/docs/asynchronous

REACT TESTING LIBRARY



REACT TESTING LIBRARY

- Nie jest samodzielną bibliteką/frameworkiem do testów (jak jest)
- Daje nam możliwość, podczas testów
  - wyrenderowania komponentu React
  - interakcji z komponentem
  - odczytywania wartości jak z drzewa DOM
- Zbudowane w oparciu o DOM Testing Library

Przydatne linki:

https://testing-library.com/docs/react-testing-library/intro → cała sekcja React Testing Library opisuje podstawy pracy z biblioteką
https://testing-library.com/docs/dom-testing-library/api-queries → metody budujące wygodne zapytania do DOM
https://testing-library.com/docs/dom-testing-library/api-events → odpalanie eventów
https://create-react-app.dev/docs/running-tests/#react-testing-library → react-testing-library jest domyślnie w create-react-app

Przykładowy test

```
import { render, screen } from '@testing-library/react';
import MyComponent from './MyComponent';

test('renders component correctly', () => {
  render(<MyComponent />);
  const headingElement = screen.getByRole('heading', { name: /hello/i });
  expect(headingElement).toBeInTheDocument();
});
```

Podstawowe metody

- **render**: Renderuje komponent React do drzewa DOM.

- **screen.getBy**...: Pobiera element z drzewa DOM na podstawie różnych kryteriów, takich jak etykieta, rolę, tekst, itp.
  https://testing-library.com/docs/queries/about

Testowanie interakcji

React Testing Library oferuje wiele metod do testowania stanu i interakcji użytkownika, najważniejsze jest **fireEvent**, co symuluje zdarzenia, takie jak kliknięcie

```
import { render, screen, fireEvent } from '@testing-library/react';
import Counter from './Counter';

test('increments counter on button click', () => {
  render(<Counter />);
  const counterElement = screen.getByTestId('counter');
  const buttonElement = screen.getByRole('button', { name: /increment/i });

  fireEvent.click(buttonElement);
  expect(counterElement.textContent).toBe('1');
});
```

Snapshoty

Snapshoty są przydatnym narzędziem do automatycznego porównywania wyników testów z oczekiwanymi wynikami.
Jest automatycznie generuje snapshoty, które są zapisywane w plikach.

```
import { render } from '@testing-library/react';
import MyComponent from './MyComponent';

test('renders component correctly', () => {
  const { container } = render(<MyComponent />);
  expect(container.firstChild).toMatchSnapshot();
});
```

Dzięki

Znajdziecie mnie:
https://www.linkedin.com/in/kamil-richert/
https://github.com/krichert