

16.03.2023

Mews

DARIA DORONINA

Frontend Platform Engineer

# Praktické testování Reactu

pomocí Jest a React testing library

- I Úvod
  - Proč psát testy?
- II Základy testování
  - Druhy testů
  - Testovací pyramida & AAA pattern
  - Testovací knihovny
- III Testování s Jest a React testing library
  - Jest a RTL
  - Testování základní komponenty
  - Mockování
  - Testování pokročilé komponenty
- IV Závěr

# Proč psát testy?



## Jistota

- minimalizuje chyby
- zajišťuje správnou funkčnost

## Údržba

- usnadňuje úpravy kódu
- zlepšuje udržitelnost projektu

# Druhy testů

## Unit testy

- testují jednotlivé části kódu izolovaně od zbytku aplikace.

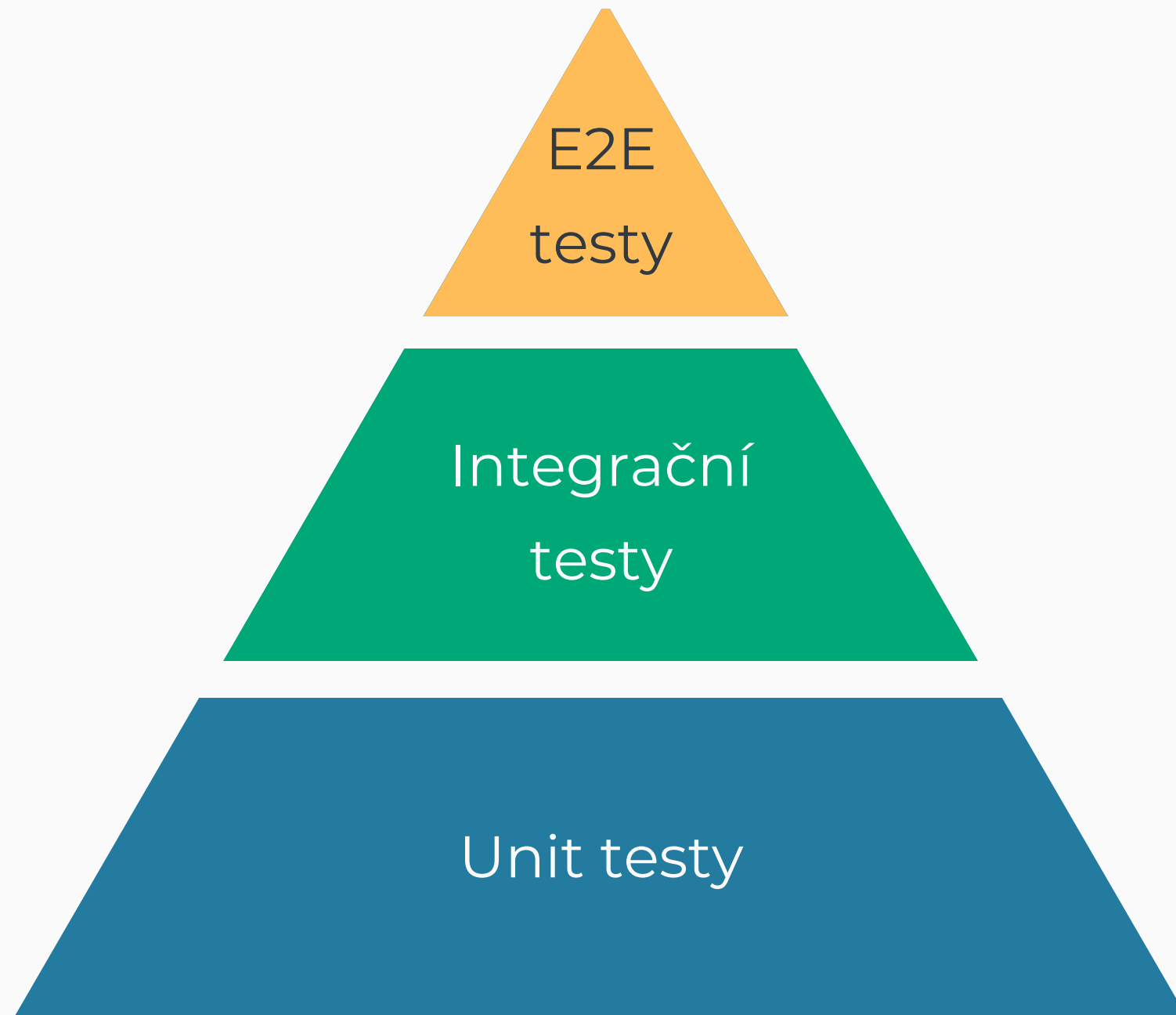
## Integrační testy

- testují, zda jednotlivé části kódu spolupracují správně.

## E2E testy

- simulují reálnou interakci uživatele s aplikací od začátku do konce

# Testovací pyramida & AAA pattern



## Arrange

nastavuje test

## Act

provede akci, kterou chceme otestovat

## Assert

ověří, zda výsledek akce odpovídá našim očekáváním

# Testovací knihovny



Jest



React testing  
library



Enzyme



Cypress

# Jest a React testing library



Jest

- Testování jakéhokoliv kódu
- Vestavěné testovací funkce
- Paralelní spouštění testů
- Detailní výstupy a reporty
- Snadná integrace s dalšími nástroji



React testing  
library

- Testuje pohled uživatele
- Omezený přístup k detailům
- Využívá utility pro interakce
- Zvyšuje spolehlivost a kvalitu
- Zaměřuje se na výstup komponenty

# Testování základní komponenty

```
describe('Greeting', () => {  
  // Arrange - nastavíme test a potřebná data  
  const props = { name: 'Jane' };  
  
  // Act - zobrazíme komponentu a provedeme akci  
  render(<Greeting {...props} />);  
  const greetingElement = screen.getByText('Hello Jane!');  
  
  // Assert - ověříme očekávaný výsledek testu  
  test('renders a greeting with the correct name', () => {  
    expect(greetingElement).toBeInTheDocument();  
  });  
});
```



# Mockování

```
// příklad mocku  
jest.fn();
```

## Co znamená mock?

- je falešný objekt nebo funkce
- nahrazuje skutečný objekt nebo funkci
- simuluje chování s omezeným rozsahem
- dává kontrolu nad výstupem

# Testování pokročilé komponenty

**Příklad 2**

**Name**

**Email**


**Password**

**Save**

# Závěr

- Proč psát testy - minimalizace chyb a zlepšení kvality kódu
- Jak určit, které části aplikace testovat - zaměřit se na funkce, které mají největší dopad na uživatelskou zkušenost
- Jak psát jednoduché testy - testovat pouze výstup UI, rozdělit testy na jednotkové a integrační, používat mocky pro izolaci částí kódu



Kent C. Dodds   
@kentcdodds



The more your tests resemble the way your software is used, the more confidence they can give you.

4:05 AM · 23 Mar, 2018

14 replies 350 shares 1.1K likes

16.03.2023

Mews

DARIA DORONINA

Frontend Platform Engineer

**Děkuji za  
pozornost!**

