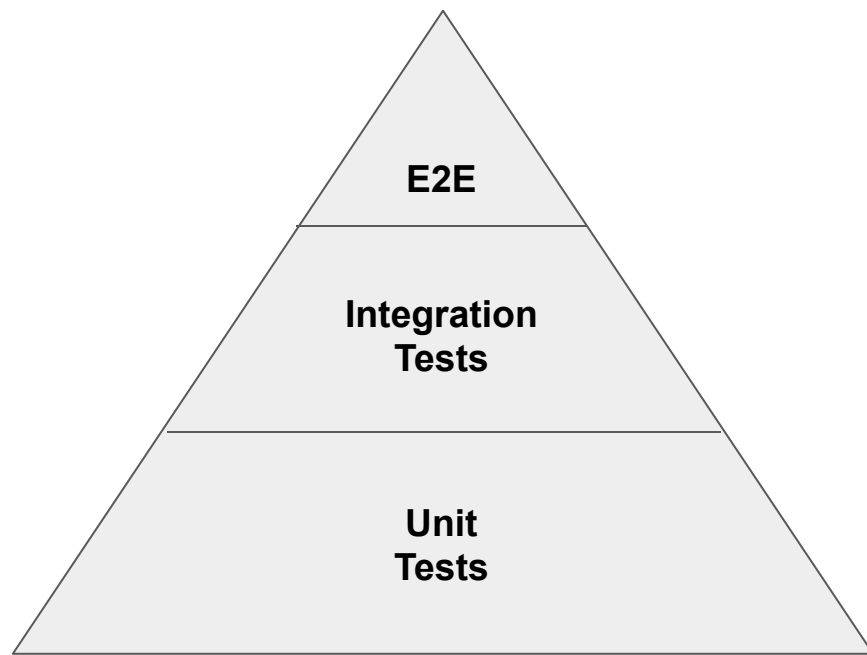


Если нужен тест,
пусть он будет интеграционным

Василий Малыхин,
QuantumSoft

Пирамида тестирования



Контекст

- React SPA (мигрировали с Angular 1), более 200 KLOC (js, css, тесты)
- Около 4.5 тыс тестов, покрытие примерно 70%
- Большая часть тестов - юнит тесты:
 - Jest snapshot tests,
 - Enzyme shallow rendering,
 - vanilla js tests

Проблемы

- Бесполезные тесты
- Хрупкие тесты
- Отсутствие единого подхода к тестированию

Бесполезные тесты

Игнорирование кусков кода из-за сложности тестирования

```
const LoginForm = (props) => {  
  return /* some jsx */  
};  
  
export default compose(  
  connect(mapStateToProps),  
  withStyles(styles),  
  fetchData(() => {  
    // some error prone logic  
  })  
) (LoginForm);  
  
export {LoginForm};
```

Бесполезные тесты

Нарушение контракта



```
const Parent = (props) => {  
  return <Child data={props.data} />  
}
```



```
const Child = (props) => {  
  return <div>{props.value}</div>  
}
```

Бесполезные тесты

Snapshot тесты

```
// Jest Snapshot v1, https://goo.gl/fbAQLP

exports[`<RatingSetup/> is matching saved snapshot 1`] = `
<div>
  <RatingProfiles
    isEditable={true}
    isInvalid={false}
    onChange={[MockFunction]}
    onCreate={[MockFunction]}
    onUpdate={[MockFunction]}
    ratingProfiles={
      Object {
        "data": Array [],
        "fetch": [MockFunction],
      }
    }
  >
  <ratingSetup={
    Object {
      "indicatorIdsWithFormulas": Array [],
      "pointsSetup": Object {
        "0": Object {
          "noAnswer": Object {
            "points": 0,
          },
          "noneOfAbove": Object {
            "points": 0,
          },
          "notApplicable": Object {
            "points": 0,
          },
        },
      },
    }
  }
  >
  </ratingSetup>
</div>
`
```

Хрупкие тесты

Использование часто меняющихся селекторов



```
it('should show No Data placeholder', () => {
  const component = shallow(<Page {...props} />);
  expect(component.find('.hint span').text()).toEqual('No Data');
});

it('should show Dialog', () => {
  const component = shallow(<Page {...props} />);
  expect(component.find('WithStyles(ForwardRef(connect(Dialog)))').length).toEqual(1);
});
```


Хрупкие тесты

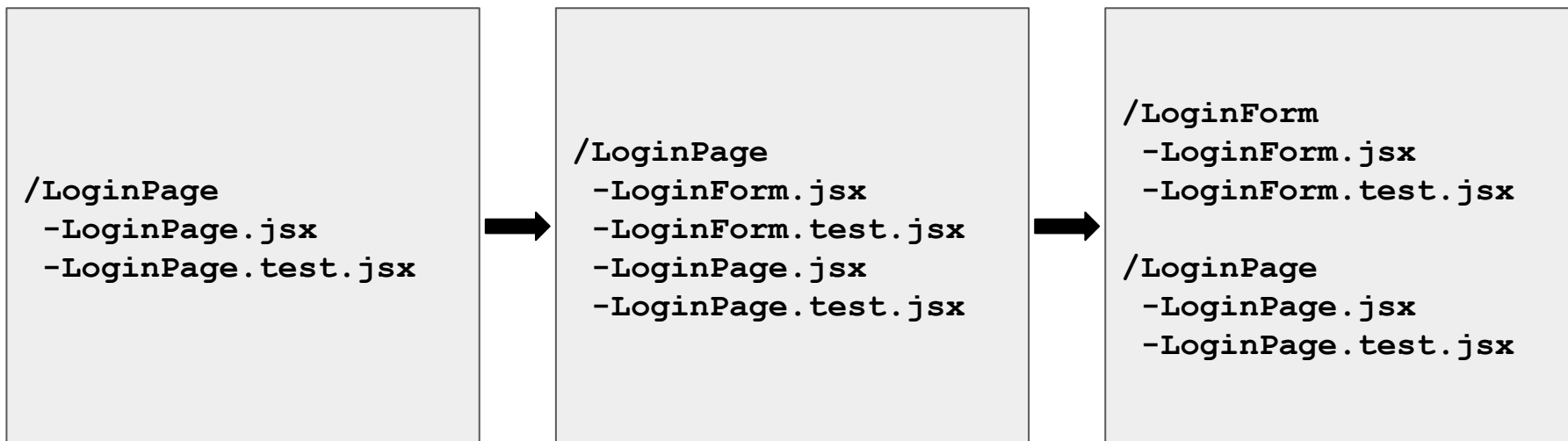
Использование методов компонента напрямую



```
it('should save data', () => {  
  const component = shallow(<Page {...props} />);  
  component.instance().saveData({email: '...', password: '...'});  
  expect(saveDataMock).toHaveBeenCalledWith(/* ... */);  
});
```

Хрупкие тесты

Частый перенос файлов и тестов при рефакторинге



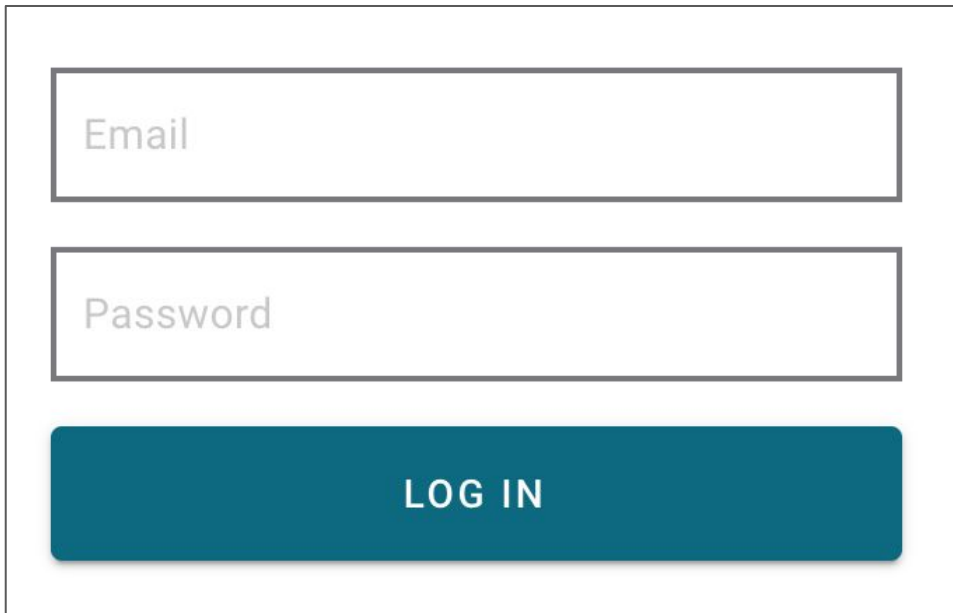
Отсутствие единого подхода к тестированию

Форма:

- react
- redux
- axios
- vanilla

Как тестить?

- Пачка юнит тестов?
- Интеграционный тест?
- Какие моки нужны?



A login form with two input fields and a button. The first input field is labeled 'Email' and the second is labeled 'Password'. Below these fields is a teal button with the text 'LOG IN' in white capital letters.

Решение



React Summit
@ReactAmsterdam



Congrats to the winning projects of the React
[#OpenSourceAwards!](#)



react-testing-library by [@kentcdodds](#)



♥ 141 5:18 PM - Apr 12, 2019



React

Docs

Tutorial

Blog

To reduce the boilerplate, we recommend using `react-testing-library` which is designed to encourage writing tests that use your components as the end users do.

@testing-library

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

"Testing library is simple and complete testing utilities that **encourage good testing practices.**"

@testing-library - API

- `findByLabelText`
- `findByPlaceholderText`
- `findByText`
- `findByAltText`
- `findByTitle`
- `findByDisplayValue`
- `findByRole`
- `findByTestId`

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

it('should disable submit button during submit', async () => {
  const onSubmit = jest.fn(() => Promise.resolve());

  const {getByText, getByPlaceholderText} = render(<Form onSubmit={onSubmit} />);

  const email = getByPlaceholderText('Email');
  const password = getByPlaceholderText('Password');
  const submit = getByText('Log in');

  fireEvent.change(email, {target: {value: 'test@test.test'}});
  fireEvent.change(password, {target: {value: 'qwerty'}});
  fireEvent.click(submit);

  expect(submit).toBeDisabled();

  await wait();

  expect(submit).toBeEnabled();
});
```

Решенные проблемы

- ~~Бесполезные тесты~~ - тесты дают больше гарантий
- ~~Хрупкие тесты~~ - тесты не привязаны к деталям реализации
- Отсутствие единого подхода к тестированию

Недостатки

- Интеграционные тесты медленные
- Скорость разработки



```
// setupFilesAfterEnv.js
beforeEach(() => {
  jest.spyOn(window, 'fetch').mockImplementation((...args) => {
    console.warn('window.fetch is not mocked for this call: ', ...args);
    throw new Error('This must be mocked')
  });
});

afterEach(() => {
  window.fetch.mockRestore();
});
```

Насколько часто тесты ловят баги?



Ссылки

- <https://kentcdodds.com/blog/write-tests>
- <https://testing-library.com/>