React Native Unit Testing

Isumi
Batam, Jan 2020

GOALS

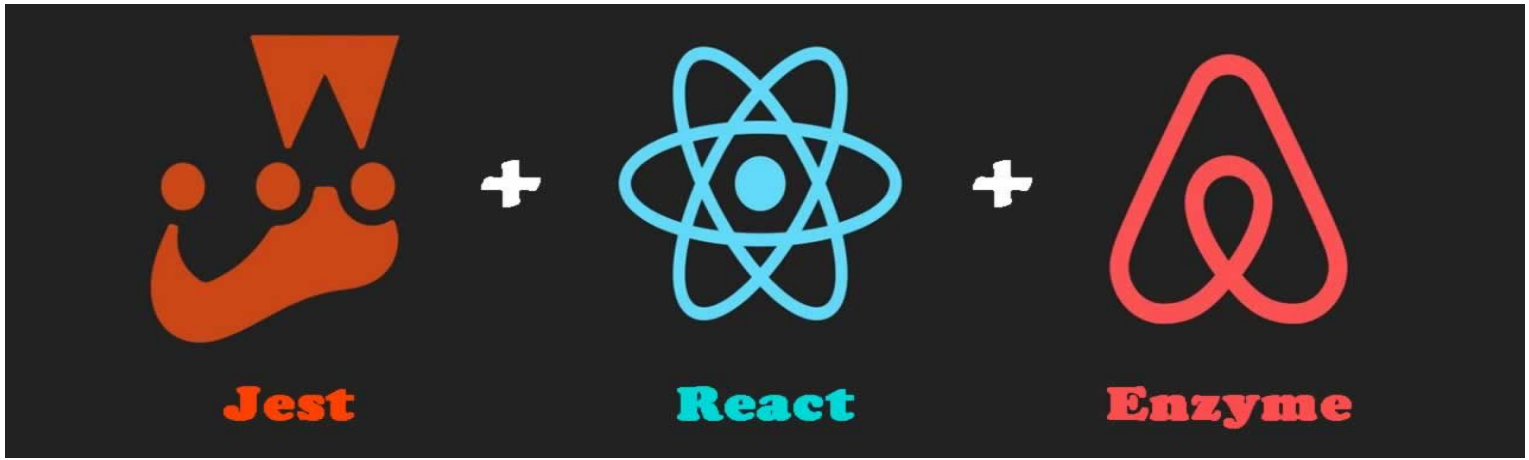
1. Intro about Unit Testing
2. Implementation Jest

1. Intro about Unit Testing

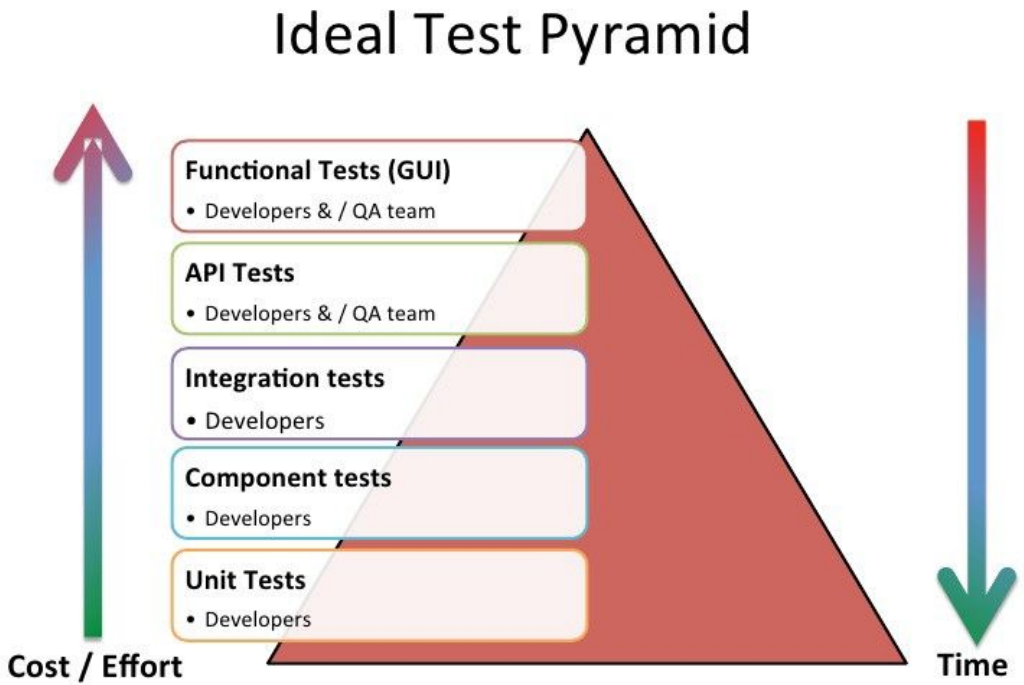
Unit testing is the first line of defence for your codebase and is key to a healthy and maintainable app.

If you don’t like testing your product, most likely your customers won’t like to test it either. — Grasshopper

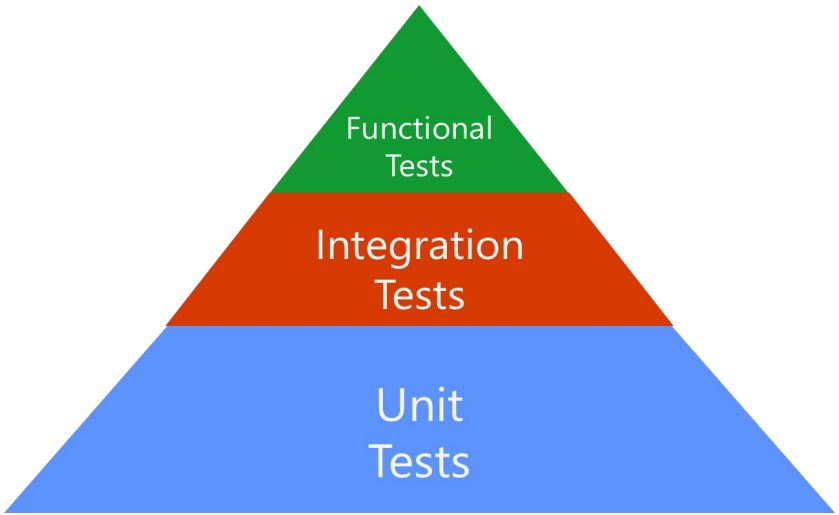
Unit Testing | Tools x Frameworks



Testing | Types



Testing | Types



Redefining a function specifically for a test to generate a result. Example: returning hard-coded data instead of relying on fetch requests or database calls. Mock functions can be defined in jest with jest.fn(() => { //function here });

Testing a multitude of functions working together, or an entire React component including children components. Example: Enzyme's mount()

Testing one isolated function, or one React component. Example: Enzyme's shallow()

Benefits?

- Quality of Code
- Provide Documentation
- Debugging Process
- Unit Testing on your resume
- etc..

Assertion

When you're writing tests, you often need to check that values meet certain conditions. Expect gives you access to a number of "matchers" that let you validate different things.

```
test('the best flavor is grapefruit', () => {
  expect(bestLaCroixFlavor()).toBe('grapefruit');
});
```

Example: Assertion

```
// File name: sum.js
export function sum(a, b) {
  return a + b;
}
```

```
// File name: __tests__/App-test.js
import {sum} from '../sum';

test('adds 1 + 2 to equal 3', () => {
  expect(sum(1, 2)).toBe(3);
});
```

2. Implementation Jest

- **Example: Repeating Setup For Many Tests**

```
beforeEach(() => {
  initializeCityDatabase();
});

afterEach(() => {
  clearCityDatabase();
});

test('city database has Vienna', () => {
  expect(isCity('Vienna')).toBeTruthy();
});

test('city database has San Juan', () => {
  expect(isCity('San Juan')).toBeTruthy();
});
```

- Example: One Time Setup

```
beforeAll(() => {
  return initializeCityDatabase();
});

afterAll(() => {
  return clearCityDatabase();
});

test('city database has Vienna', () => {
  expect(isCity('Vienna')).toBeTruthy();
});

test('city database has San Juan', () => {
  expect(isCity('San Juan')).toBeTruthy();
});
```

- Example: Another way

```
// Applies to all tests in this file
beforeEach(() => {
  return initializeCityDatabase();
});



describe('matching cities to foods', () => {
  // Applies only to tests in this describe block
  beforeEach(() => {
    return initializeFoodDatabase();
  });

  test('Vienna <3 sausage', () => {
    expect(isValidCityFoodPair('Vienna', 'Wiener Schnitzel')).toBe(true);
  });

  test('San Juan <3 plantains', () => {
    expect(isValidCityFoodPair('San Juan', 'Mofongo')).toBe(true);
  });
});
```

Sample Coding

Install!

```
npm install jest-environment-enzyme jest-enzyme
enzyme-adapter-react-16 react-dom enzyme
--save-dev
```

./package.json (cari bagian jest)

```
"jest": {
   "preset": "react-native",
   "collectCoverage": true,
   "setupFilesAfterEnv": ["jest-enzyme"],
   "testEnvironment": "enzyme",
   "testEnvironmentOptions": {
     "enzymeAdapter": "react16"
   }
 }
```

./src/components/Button.js

```
import React from 'react'
import { Text, TouchableOpacity } from 'react-native'

const Button = props => {
 const {buttonStyle, textStyle} = styles;
 const {onPress, label} = props;
 return (
   <TouchableOpacity onPress={onPress} style={buttonStyle}>
     <Text style={textStyle}>test label</Text>
   </TouchableOpacity>
 );
};
```

```
const styles = {
 textStyle: {
   alignSelf: 'center',
   color: '#fff',
   fontSize: 16,
   fontWeight: '600',
 },
 buttonStyle: {
   height: 45,
   alignSelf: 'stretch',
   justifyContent: 'center',
   backgroundColor: '#38ba7d',
   borderBottomWidth: 6,
   borderBottomColor: '#1e6343',
   borderWidth: 1,
   marginLeft: 15,
   marginRight: 15,
 },
};

export default Button;
```

./__test__/button.test.js

```
import React from 'react';
import {shallow} from 'enzyme';
import Button from '../src/component/Button';
import Enzyme from 'enzyme';
import Adapter from 'enzyme-adapter-react-16';

Enzyme.configure({adapter: new Adapter()});

describe('Button', () => {
 describe('Rendering', () => {
   it('should match to snapshot', () => {
     const component = shallow(<Button label="test label" />);
     expect(component).toMatchSnapshot();
   });
 });
});
```

BINAR ACADEMY

Creating Snapshot

With Jest we have a very easy way to test if a component is rendered correctly given certain props and state. It's called snapshot testing.

Snapshot tests are a very useful tool whenever you want to make sure your UI does not change unexpectedly.

In this case we have used Enzyme's shallow rendering to help us create a snapshot. And then ...

```
npm run test
```

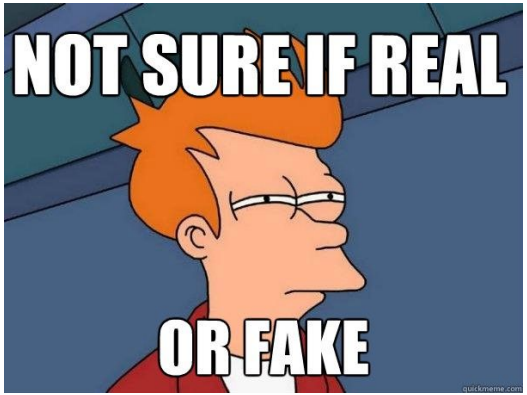
Jest will create a snapshot file for us inside the folder ***__snapshots__***

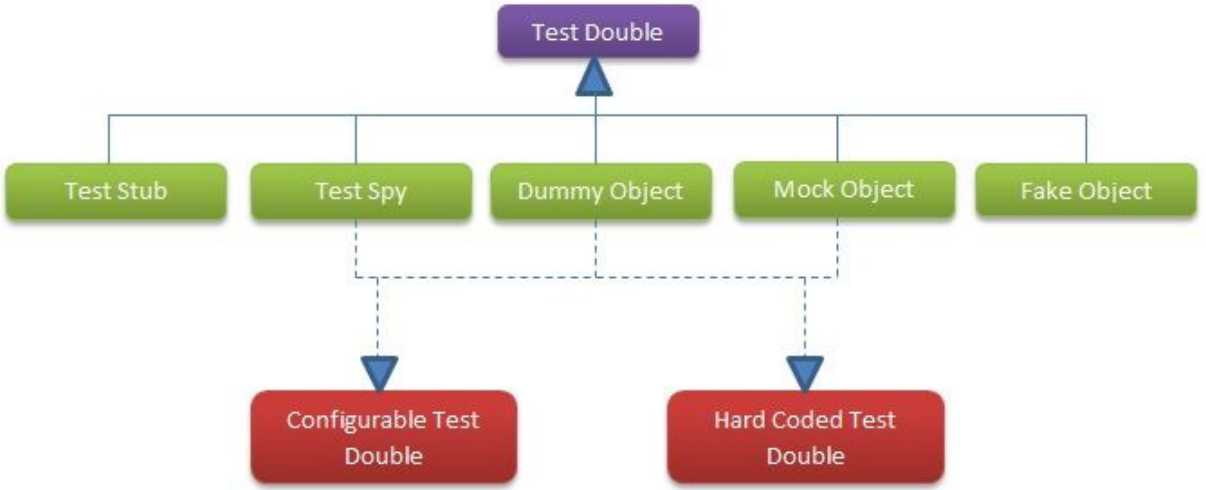
Notes

- **Kalau terjadi perubahan component..?**
  Update saja snapshot-nya:
  ```
  npm run test -- -u
  ```
- **Coba ganti ukuran font, lalu...?**
  ```
  npm run test
  ```

BINAR ACADEMY

Stubs & Mocks





BINAR ACADEMY

Implementation Stubs & Mocks

```
describe('Interaction', () => {

        describe('onPressHandler', () => {
            it('should call onPress', () => {
                // Arrange
                const mockOnPress = jest.fn();      // 1. mock function
                const component = shallow(<Button
                    label= "test label"
                    onPress={mockOnPress}           // 2. passing in mock function as props
                />);
                const instance = component.instance();  // 3. getting an instance of component
                // Act
                instance.onPressHandler();          // 4. manually triggering onPressHandler()
                // Assert
                expect(mockOnPress).toHaveBeenCalled();
                expect(mockOnPress).toHaveBeenCalledTimes(1);   // 5. checking return values
        });
});
```

References

- Unit Testing in React Native with Jest
- TDD vs BDD vs DDD
- Unit Testing (Programming with mosh)