JEST in React Native

What is Jest?

Jest is a JavaScript testing library created by the same company behind React: Facebook. But while Jest was created by the same developers who created React, it is not limited to only React—it can also be used with other frameworks like Node, Angular, Vue.js.

Advantages:

- Zero configuration: No need to set up a config file. Just install, and start writing tests!
- Snapshots: Ensure the UI (User Interface) doesn't change unexpectedly.

https://docs.expo.dev/guides/testing-with-jest/

```
$ expo init jest-app

$ cd jest-app

$ npx expo install  jest-expo  jest
```

Update package.json

```
 "scripts": {
   ...
   "test": "jest"
 },
 "jest": {
   "preset": "jest-expo"
 }
```

Configuration

A starting configuration you can use is to make sure any modules you are using within the node_modules directory are transpiled when running Jest. This can be done by including the transformIgnorePatterns property that takes a regex pattern as its value:

```
"jest": {

  "preset": "jest-expo",

  "transformIgnorePatterns": [
```

```
    "node_modules/(?!((jest-)?react-native|@react-native(-
community)?)|expo(nent)?|@expo(nent)?/.*|@expo-google-fonts/.*|react-navigation|@react-
navigation/.*|@unimodules/.*|unimodules|sentry-expo|native-base|react-native-svg)"

  ]

}
```

Unit test

A unit test is used to check the smallest unit of code, usually a function.

To write your first unit test, start by writing a simple test for App.js. Create a test file for it and call it App.test.js. Jest identifies a file with the .test.js extension as a test and includes it in the tests queue.

Add App.test.js and write code

```
import React from 'react';

import renderer from 'react-test-renderer';


import App from './App';


describe('<App />', () => {

  it('has 1 child', () => {

    const tree = renderer.create(<App />).toJSON();

    expect(tree.children.length).toBe(1);

  });

});
```

To run the test: $ npm run test

Snapshot test

A snapshot test is used to make sure that UI stays consistent, especially when a project is working with global styles that are potentially shared across components.

To add a snapshot test for <App />, add the following code snippet in the describe() in App.test.js:

```
it('renders correctly', () => {
  const tree = renderer.create(<App />).toJSON();
  expect(tree).toMatchSnapshot();
});
```

Run npm run test command, and if everything goes well, you should see a snapshot created and two tests passed.

Code coverage reports

Code coverage reports can help you understand how much of your code is tested.

If you'd like to see code coverage report in your project using the HTML format, add the following to the package.json:

package.json

```
"jest": {
  ...
  "collectCoverage": true,
  "collectCoverageFrom": [
    "**/*.{js,jsx}",
    "!**/coverage/**",
    "!**/node_modules/**",
    "!**/babel.config.js",
    "!**/jest.setup.js"
  ]
```

}

Run npm run test. You should see a coverage directory created in your project. Find the index.html file within this directory and double-click to open it up in a browser to see the coverage report.

Project structure:

Right now, you have a single test file in the project directory. Adding more test files can make it hard to organize your project directory. The easiest way to avoid this is to create a __tests__ directory and put all your tests inside it.

```
__tests__/
├── components/
│   └── button.test.js
├── navigation/
│   └── mainstack.test.js
└── screens/
    └── home.test.js
src/
├── components/
│   └── button.js
├── navigation/
│   └── mainstack.js
└── screens/
    └── home.js
```

Optional: Jest Flows:

```
"scripts": {

  ...

  // active development of tests, watch files for changes and re-runs all tests

  "test": "jest --watch --coverage=false --changedSince=origin/main",



  // debug, console.logs and only re-runs the file that was changed

  "testDebug": "jest -o --watch --coverage=false",



  // displays code coverage in cli and updates the code coverage html
```

```
  "testFinal": "jest",


  // when a screen/component is updated, the test snapshots will throw an error, this updates them
  "updateSnapshots": "jest -u --coverage=false"
}
```