**Testing in React Native**

1. **Why Testing**

Testing is important because it helps you uncover these mistakes and verifies that your code is working. Perhaps even more importantly, testing ensures that your code continues to work in the future as you add new features, refactor the existing ones, or upgrade major dependencies of your project.

More about Testing: <https://reactnative.dev/docs/testing-overview>

1. **Main Type of Tests**

* Unit Tests
* Components Tests
* End-2-End Tests

More on each on this: <https://reactnativetesting.io/>

1. **Using Jest for Test**
   * There more test framework out there (chai JS, Jasmin, Mocha, etc.) like jest but jest is mostly used and well maintained
   * Jest Doc: <https://jestjs.io/>
2. **Testing Library for React Native**

* 'React-test-renderer’ npm: <https://www.npmjs.com/package/react-test-renderer>
* Enzyme JS, doc: <https://enzymejs.github.io/enzyme/>
* React-Native-Testing-Library, doc: <https://callstack.github.io/react-native-testing-library/>

**Follow guide to Set-up Jest and React-Native-Testing-Library**

Use Jest guide to set-up Jest environment, (Most react project have jest environment set-up by default)

Install @testin-library/react-native

Add require config in babel or V8 config file

For more details on jest command and its configuration consider jest.io doc.

Check by set-up by running smoke test.



Add npm command In package.json



For coverage of tests add –coverage after jest

Use \_\_tests\_\_ folder for including tests.

Name test file nameOfFile.test.tsx/ts

ex. Home.test.tsx , Validate.test.ts

User \_\_mocks\_\_ folder for including universal mocks

Name mock file mock-(package/filename).ts

Ex. mock-async-storagemock.ts, mock-i18n.ts

**For Mock and test on library use their documentation for mocks and tests.**

* <https://reactnavigation.org/docs/testing>
* <https://react-native-async-storage.github.io/async-storage/docs/advanced/jest>
* <https://redux.js.org/introduction/ecosystem#testing>
* <https://redux-saga.js.org/docs/advanced/Testing/>

**Some Important things**

Use jest.fn() and jest.SpyOn() to mock and observe some function call and there behavior read jest documentation for all possible return type and mock implementation.

If application use navigation or some other library like redux, redux-saga, react-native-paper

User their Provider to wrap component in.

Ex,

<NavigationContainer>  
 <Provider store={mockStore}>  
 <Home />  
 </Provider>  
</NavigationContainer>

Or create helper faction for warping and rendering it

Ex:

export function renderWithNavigatorAndRedux(  
 renderComponent: any,  
 reduxEvent?: (store: Store) => void,  
) {  
 const sagaMiddleware = createSagaMiddleware();  
 const store = createStore(rootReducer, applyMiddleware(sagaMiddleware));  
 sagaMiddleware.run(root);  
  
 if (reduxEvent) {  
 reduxEvent(store);  
 }  
  
 return render(  
 <NavigationContainer>  
 <Provider store={store}>{renderComponent}</Provider>  
 </NavigationContainer>,  
 );  
}

**Use Coverage to get details of all files.**

Coverage generates html report for viewing how many files and line of code are being tested and which are not being tested.

