Contents

[WHAT IS TESTING? 1](#_Toc78354620)

[REQUIRED TESTING TOOLS 2](#_Toc78354621)

[WHAT TO TEST? 3](#_Toc78354622)

[WRITNG OUR FIRST TEST 3](#_Toc78354623)

[TESTING COMPONENTS CONTINUED 5](#_Toc78354624)

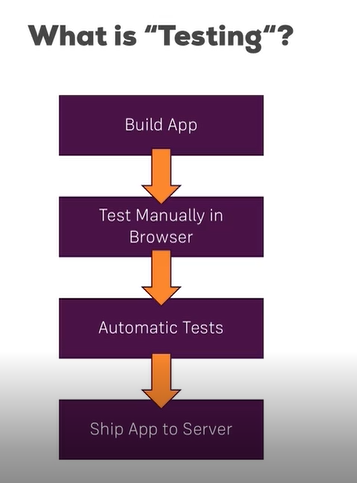
[JEST AND ENZYME DOCUMENTATIONS 8](#_Toc78354625)

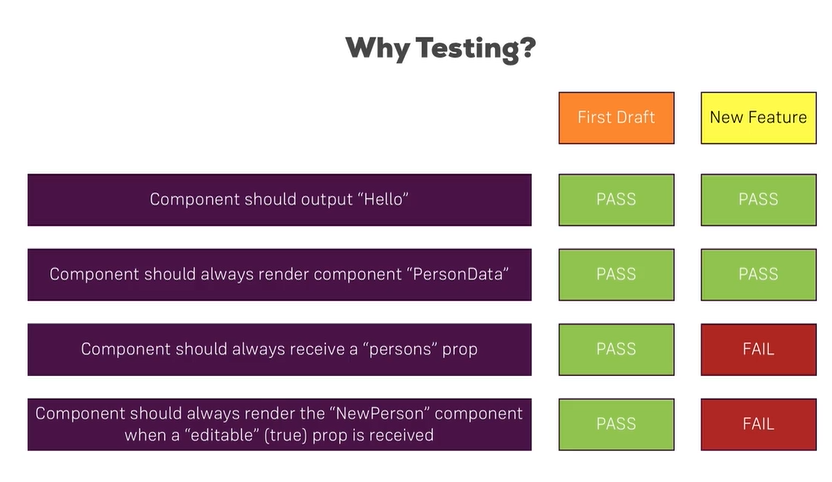
[TESTING COMPONENTS CORRECTLY 8](#_Toc78354626)

[TESTING CONTAINERS 8](#_Toc78354627)

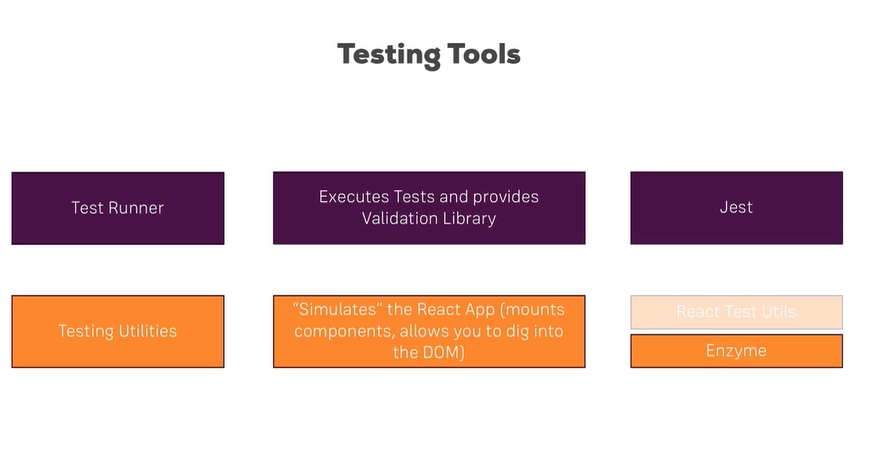
[HOW TO TEST REDUX 9](#_Toc78354628)

# WHAT IS TESTING?





# REQUIRED TESTING TOOLS



We need two tools: 1. Test runner which executes tests and provides validation library.

We need testing utilizes which will helps us in testing. We would React Test Utils but react team recommend us to use Enzyme.

# WHAT TO TEST?

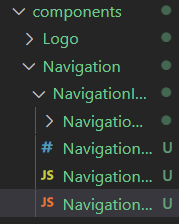


# WRITNG OUR FIRST TEST

Look package.json to see whether jest is installed or note;

Install: npm install --save enzyme react-test-renderer enzyme-adapter-react-16

Create file named: NavigationItems.test.js in NavigationItems folder.



NavigationItems.test.js:

import { configure,shallow } from 'enzyme';

import Adapter from 'enzyme-adapter-react-16';

import NavigationItems from './NavigationItems';

import React from 'react';

import NavigationItem from './NavigationItem/NavigationItem';

//shallow helps to render in many circumstances

configure({adapter: new Adapter()});

describe('<NavigationItems />',() =>{

    //actaul Test

    it('should render two <NavigationItem /> elements if note authenticated',() => { //allows to write one test at a time

        //acutal Testing logic

        const wrapper = shallow(<NavigationItems />);

        //expectations

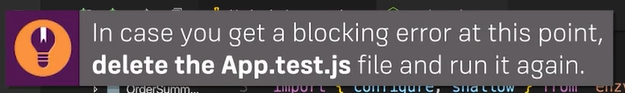
        expect(wrapper.find(NavigationItem)).toHaveLength(2);

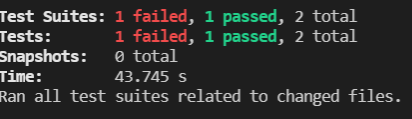
    });

});

RUN:

-npm test





# TESTING COMPONENTS CONTINUED

NavigationItems.test.js:

import { configure,shallow } from 'enzyme';

import Adapter from 'enzyme-adapter-react-16';

import NavigationItems from './NavigationItems';

import React from 'react';

import NavigationItem from './NavigationItem/NavigationItem';

//shallow helps to render in many circumstances

configure({adapter: new Adapter()});

describe('<NavigationItems />',() =>{

    //actaul Test

    it('should render two <NavigationItem /> elements if not authenticated',() => { //allows to write one test at a time

        //acutal Testing logic

        const wrapper = shallow(<NavigationItems />);

        //expectations

        expect(wrapper.find(NavigationItem)).toHaveLength(2);

    });

    //actaul Test

    it('should render three <NavigationItem /> elements if authenticated',() => { //allows to write one test at a time

        //acutal Testing logic

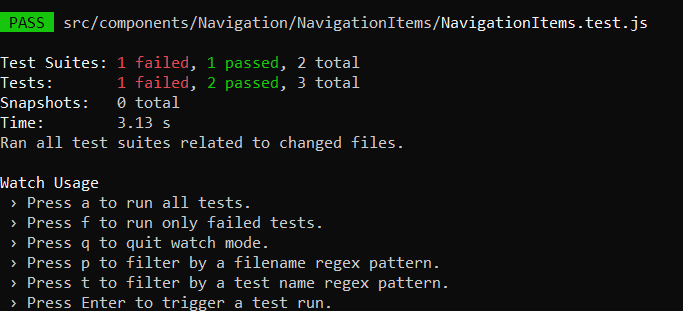
        const wrapper = shallow(<NavigationItems  isAuthenticated/>);

        //expectations

        expect(wrapper.find(NavigationItem)).toHaveLength(3);

    });

});



describe('<NavigationItems />',() =>{

    // beforeEach or afterEach

    let wrapper;

    beforeEach(() => {

        wrapper =  shallow(<NavigationItems />);

    });

    //actaul Test

    it('should render two <NavigationItem /> elements if not authenticated',() => { //allows to write one test at a time

        //acutal Testing logic

        //expectations

        expect(wrapper.find(NavigationItem)).toHaveLength(2);

    });

    //actaul Test

    it('should render three <NavigationItem /> elements if authenticated',() => { //allows to write one test at a time

        //acutal Testing logic

        //expectations

        expect(wrapper.find(NavigationItem)).toHaveLength(3);

    });

});



FAIL: it is because user is not authenticated.

        wrapper =  shallow(<NavigationItems />);

isAuthenticated is missing.

TO FIX THIS:

 //actaul Test

    it('should render three <NavigationItem /> elements if authenticated',() => { //allows to write one test at a time

        //acutal Testing logic

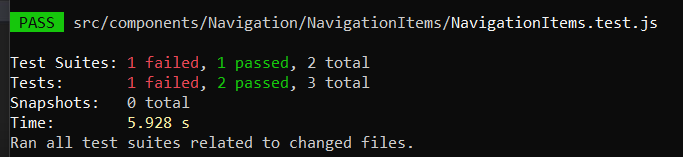
        // const wrapper = shallow(<NavigationItems  isAuthenticated/>);

        wrapper.setProps({isAuthenticated: true});

        //expectations

        expect(wrapper.find(NavigationItem)).toHaveLength(3);

    });



# JEST AND ENZYME DOCUMENTATIONS

Visit Official Docs of Jest and Enzyme to go deep.

To find logout is there or not with help of contains() :

    //actaul Test

    it('Is there logout or not',() => { //allows to write one test at a time

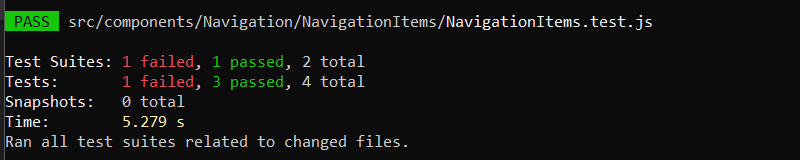
        //without wrapper it will fail, because there is no authenticated

        wrapper.setProps({isAuthenticated: true});

        expect(wrapper.contains(<NavigationItem link="/logout">Logout</NavigationItem>)).toEqual(true);

    });

Result:



# TESTING COMPONENTS CORRECTLY

Look at your components, function, and class more in depth. Know what you want to test.

# TESTING CONTAINERS

PizzaBuilder.test.js

import React from 'react';

import {PizzaBuilder} from './PizzaBuilder';

import { configure,shallow } from 'enzyme';

import Adapter from 'enzyme-adapter-react-16';

import BuildControls from  '../../components/Pizza/BuildControls/BuildControls';

//shallow helps to render in many circumstances

configure({adapter: new Adapter()});

describe('<PizzaBuilder>', () => {

    let wrapper;

    beforeEach(() =>{

        wrapper = shallow(<PizzaBuilder />);

    });

    it('should render <BuildControls /> when receiveing ingrediencts',() =>{

        wrapper.setProps({ings: {salad:0}});

        expect(wrapper.find(BuildControls)).toHaveLength(1);

    })

})

# HOW TO TEST REDUX

Auth.test.js

import reducer from './auth';

import \* as actionTypes from '../actions/actionTypes';

describe('auth reducer', () => {

    it('should return the initial state', ()=>{

        expect(reducer(undefined,{})).toEqual({

            token: null,

            userId: null,

            error: null,

            loading: false,

            authRedirectPath: '/'

        });

    });

    it('should store the token upon login',() => {

        expect(reducer({

            token: null,

            userId: null,

            error: null,

            loading: false,

            authRedirectPath: '/'

        }, {

            type: actionTypes.AUTH\_SUCCESS,

            idToken: 'some-token',

            userId: 'some user id'

        })).toEqual({

            token: 'some-token',

            userId: 'some user id',

            error: null,

            loading: false,

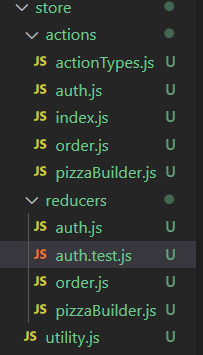
            authRedirectPath: '/'

        });

    })

});

Folder:



RESULT:

