## **1. React测试初体验**

create-react-app zf-react-test

cd zf-react-test

npm test

sum.js

module.exports = **function**(a,b){

**return** a+b;

}

sum.spec.js

**let** sum = require('./sum');

it('a+b',**function**(){

expect(sum(1,2)).toBe(3);

});

## **2.enzyme测试**



* Shallow Rendering(浅渲染)指的是，将一个组件渲染成虚拟DOM对象，但是只渲染第一层，不渲染所有子组件，所以处理速度非常快。它不需要DOM环境，因为根本没有加载进DOM
* find()方法：只支持简单选择器 类选择器、id选择器、标签选择器、复合选择器

cnpm i enzyme enzyme-adapter-react-16 -D

TodoInput.js

**import** React, { Component } **from** 'react';**class** **TodoInput** **extends** **Component** {

render() {

**return** (

<div>

<h3>待办事项</h3>

<input defaultValue="请输入"/>

</div>

);

}

}

**export** **default** TodoInput;

TodoInput.spec.js

**import** React **from** 'react';**import** Enzyme,{shallow} **from** 'enzyme';**import** TodoInput **from** './TodoInput';**import** Adapter **from** 'enzyme-adapter-react-16';

Enzyme.configure({adapter:**new** Adapter()});//http://airbnb.io/enzyme/docs/api/ShallowWrapper/find.html//http://airbnb.io/enzyme/docs/api/selector.html

it('shoud render 请输入',**function**(){

**const** wrapper = shallow(<TodoInput/>);

**const** h3 = wrapper.find('h3');

**const** input = wrapper.find('input');

expect(h3.text()).toBe('待办事项');

expect(input.props().defaultValue).toBe('请输入');

});

src\components\TodoList.js

**static** propTypes = {

todos:PropTypes.arrayOf(PropTypes.shape({

text:PropTypes.string.isRequired,

createAt:PropTypes.object.isRequired

}))

}

## **3.点击事件**

**import** React **from** 'react';**import** Enzyme,{shallow} **from** 'enzyme';**import** TodoInput **from** './TodoInput';**import** Adapter **from** 'enzyme-adapter-react-16';

Enzyme.configure({adapter:**new** Adapter()});//http://airbnb.io/enzyme/docs/api/ShallowWrapper/find.html//http://airbnb.io/enzyme/docs/api/selector.html

describe('测试TodoInput',**function**(){

**let** todos;

beforeEach(()=>{

todos =[{text:'1'},{text:'2'}];

});

it('shoud render 请输入',**function**(){

**let** wrapper = shallow(<TodoInput/>);

**const** h3 = wrapper.find('h3');

**const** input = wrapper.find('input');

expect(h3.text()).toBe('待办事项');

expect(input.props().defaultValue).toBe('请输入');

});

it('点击按钮的时候调用addTodo方法',**function**(){

**let** addTodo = jest.fn();

**let** wrapper = shallow(<TodoInput addTodo = {addTodo}/>);

**let** button = wrapper.find('button');

button.simulate('click');

expect(addTodo).toBeCalled();

});

});

## **4. TDD**

* TDD是测试驱动开发（Test-Driven Development）是敏捷开发中的一项核心实践和技术，也是一种设计方法论
* TDD的原理是在开发功能代码之前，先编写单元测试用例代码，测试代码确定需要编写什么产品代码

**let** reducer = require('./reducer');**const** ADD\_TODO = 'ADD\_TODO';**const** DEL\_TODO = 'DEL\_TODO';

describe('reducer',()=>{

**let** INIT\_STATE = [{id:1,text:'1'},{id:2,text:'2'}];

it('初始状态',()=>{

expect(reducer(undefined,{})).toEqual(INIT\_STATE);

});

it('增加todo',()=>{

**let** todos = reducer(INIT\_STATE,{type:ADD\_TODO,todo:{id:3,text:'3'}});

expect(todos).toEqual([...INIT\_STATE,{id:3,text:'3'}]);

});

it('删除todo',()=>{

**let** todos = reducer(INIT\_STATE,{type:DEL\_TODO,id:2});

expect(todos).toEqual([{id:1,text:'1'}]);

});

});

**const** ADD\_TODO = 'ADD\_TODO';**const** DEL\_TODO = 'DEL\_TODO';**let** INIT\_STATE = [{id:1,text:'1'},{id:2,text:'2'}];**function** **reducer**(state=INIT\_STATE,action={}){

**switch**(action.type){

**case** ADD\_TODO:

**return** [...state,action.todo];

**case** DEL\_TODO:

**return** state.filter(item=>item.id != action.id);

**default**:

**return** state;

}

}module.exports = reducer;

## **5. 测试点击事件**

**import** React **from** 'react';**import** Enzyme,{shallow} **from** 'enzyme';**import** TodoItem **from** './TodoItem';**import** Adapter **from** 'enzyme-adapter-react-16'; **import** { wrap } **from** 'module';

Enzyme.configure({adapter:**new** Adapter()});//http://airbnb.io/enzyme/docs/api/ShallowWrapper/find.html//http://airbnb.io/enzyme/docs/api/selector.html

describe('TodoItem',**function**(){

it('todo',()=>{

**const** wrapper = shallow(<TodoItem todo={{id:1,text:'1'}}/>);

expect(wrapper.text()).toMatch(/1/);

expect(wrapper.hasClass('todo')).toBe(true);

expect(wrapper.hasClass('todo-selected')).toBe(false);

});

it('todo-selected',()=>{

**const** wrapper = shallow(<TodoItem todo={{id:1,text:'1'}}/>);

expect(wrapper.text()).toMatch(/1/);

wrapper.simulate('click');

expect(wrapper.hasClass('todo')).toBe(false);

expect(wrapper.hasClass('todo-selected')).toBe(true);

});

});

## **6. mount**

mount将React组件加载为真实的DOM

* .get(index)：返回指定位置的子组件的DOM节点
* .at(index)：返回指定位置的子组件
* .text()：返回当前组件的文本内容
* .html()：返回当前组件的HTML代码形式
* .props()：返回根组件的所有属性
* .prop(key)：返回根组件的指定属性
* .state([key])：返回根组件的状态

**import** React **from** 'react';**import** Enzyme,{mount} **from** 'enzyme';**import** TodoApp **from** './TodoApp';**import** Adapter **from** 'enzyme-adapter-react-16';

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

describe('TodoApp',**function**(){

it('addTodo',()=>{

**let** wrapper = mount(<TodoApp/>);

**let** len = wrapper.find('li').length;

wrapper.find('button').at(0).simulate('click');

expect(wrapper.find('li').length).toBe(len + 1);

});

it('delTodo',()=>{

**let** wrapper = mount(<TodoApp/>);

**let** len = wrapper.find('li').length;

wrapper.find('button').at(1).simulate('click');

expect(wrapper.find('li').length).toBe(len - 1);

});

});