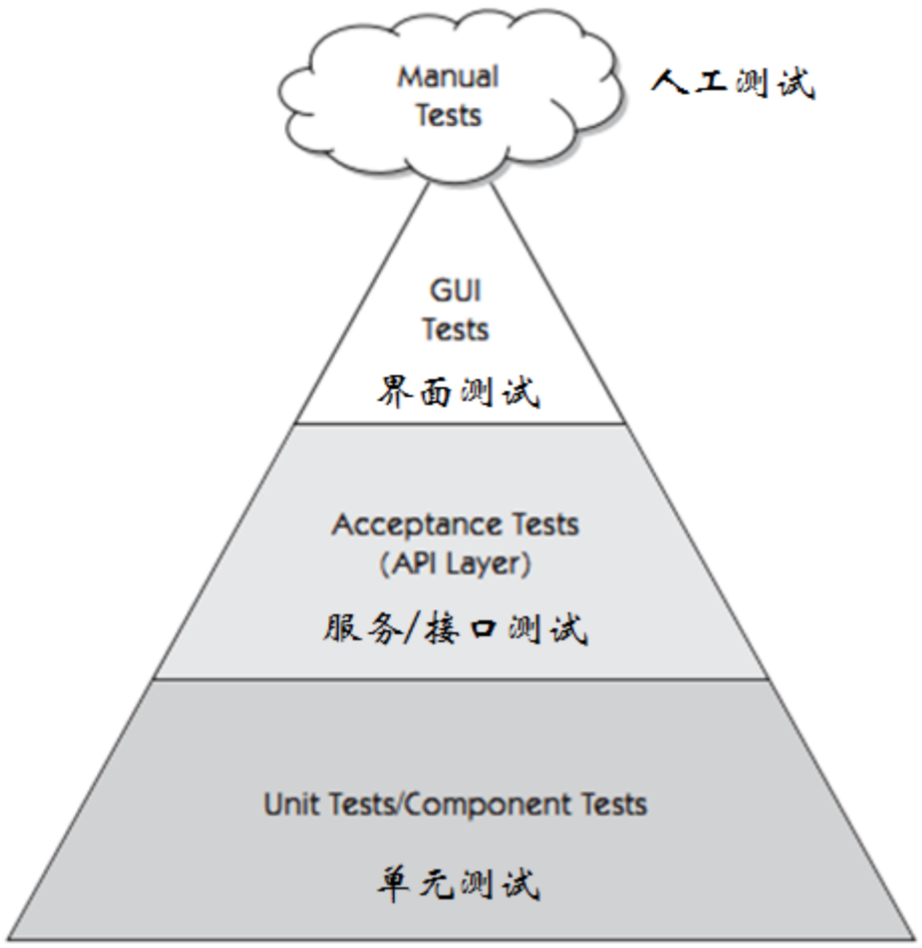
前端单测

主要内容

- 单元测试的相关内容
- 代码要求
- jest + vue test utils
- 例子讲解

现状

很多人都认为前端都是ui dom，没啥好单测的，事实上确实也没很少有团队严格执行要求前端单测覆盖率的，多数是针对一些开源的库、基础库这种复用率极高的代码，不过很多团队已经开始在补充这块的建设
其实单元测试都是很有必要的，前端也不例外



为什么需要单元测试

- 增加程序的健壮性
- 保证重构正确性
- 代替注释阅读测试用例了解代码的功能
- 测试驱动开发（TDD,Test-Driven Development）

单元测试基本概念

断言（assert）

断言是编写测试用例的核心实现方式，根据期望值判断本条case是否通过

测试用例

设置一组输入条件、预期结果，判断该代码是否符合要求

覆盖率

- 语句覆盖率（statement coverage）
- 分支覆盖率（branch coverage）
- 函数覆盖率（function coverage）
- 行覆盖率（line coverage）

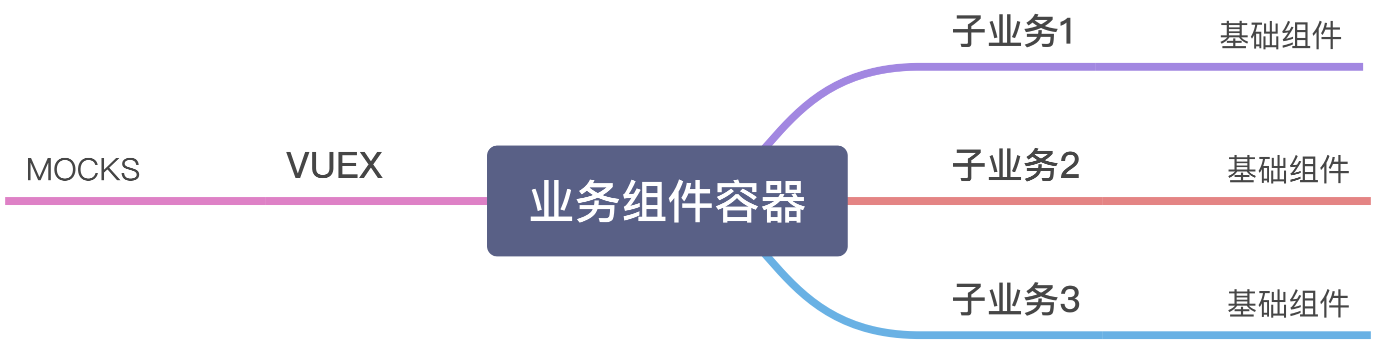
代码要求

战绩部沟通前期战绩部的cli脚本只统计java代码，后续会增加前端代码的统计，等确定要统计前端的时候需要再沟通覆盖方案方案
很早之前就想完善单元测试这块，所以加到了今年的目标中

覆盖范围

js工具类 + 基础组件 + vuex + 部分简单业务

代码要求



业务组件模块化，拆分为：业务组件（多层） + 基础组件

　现在我们现在的order-vue，业务组件可以适当优化、抽离，将业务和基础展示拆分出来

　订购容器组件（组装数据请求 + 业务拼接） + 多级子业务组件 + 基础组件（当前配置 + 地域组件等）

完善mock数据，增加npm run mock

　使用第三方库，指定格式造数据

　本地创建文件存储

本地启动服务，将数据存储在服务中

数据存储在某个稳定的服务器中

前端单元测试框架

框架	描述	备注
jest	Delightful JavaScript Testing. Works out of the box for any React project.Capture snapshots of React trees	出自facebook，集成完善，默认配置jsdom,内置断言库、自带命令行，支持puppeteer react、antd等在使用
mocha	Simple, flexible, fun JavaScript test framework for Node.js & The Browser	需要手动集合chai、js-dom使用
karma	A simple tool that allows you to execute JavaScript code in multiple real browsers. Karma is not a testing framework, nor an assertion library. Karma just launches an HTTP server, and generates the test runner HTML file you probably already know from your favourite testing framework.	element-ui, iview使用
chai	BDD / TDD assertion framework for node.js and the browser that can be paired with any testing framework.	断言库
sinon	Standalone and test framework agnostic JavaScript test spies, stubs and mocks	

框架	描述	备注
jsmine	Jasmine is a Behavior Driven Development testing framework for JavaScript. It does not rely on browsers, DOM, or any JavaScript framework. Thus it's suited for websites, Node.js projects, or anywhere that JavaScript can run.	
vue/test-utils	Utilities for testing Vue	vue官方推出的方案，方案仍然不完善，现在仍然是1.0beta版本，vuetify和一些项目在使用
vue/avoriaz	a Vue.js testing utility library	2年多没人维护，语法和test-utils类似

技术选择

vue/test-utils + jest

为什么选择Jest

jest有facebook背书，并且集成完善，基本只依赖一个库就行，vue/test-utils友好支持jest
mocha需要其他比如chai、jsdom等配合使用

为什么选择vue/test-utils

官方推荐，持续有人更新维护，良好的支持了组件的setProps、setData、事件模拟、vuex、vue-router等，github上很多开源项目在使用：vuetify等

jest

基础语法

except：（期望）
toBe：值相等
toEqual：递归检查对象或数组
toBeUndefined
toBeNull
toBeTruthy
toMatch：正则
使用 expect(n).toBe(x)

异步处理

callbck done/promise/async await

```
// done callback
test('the data is peanut butter', done => {
  function callback(data) {
    try {
      expect(data).toBe('peanut butter');
      done();
    } catch (error) {
      done(error);
    }
  }
  fetchData(callback);
});
// promise
test('the data is peanut butter', () => {
  return fetchData().then(data => {
    expect(data).toBe('peanut butter');
  });
});
// async await
test('the data is peanut butter', async () => {
  const data = await fetchData();
  expect(data).toBe('peanut butter');
});
```

前置执行

```
// 重复执行， 每次调用都执行
beforeEach(() => {
  initializeCityDatabase();
});
afterEach(() => {
  clearCityDatabase();
});
// 单次执行
beforeAll(() => {
  return initializeCityDatabase();
});

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

作用域

```
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);
  });
});
// 查看执行顺序
beforeAll(() => console.log('1 - beforeAll'));
afterAll(() => console.log('1 - afterAll'));
beforeEach(() => console.log('1 - beforeEach'));
afterEach(() => console.log('1 - afterEach'));
test('', () => console.log('1 - test'));
describe('Scoped / Nested block', () => {
  beforeAll(() => console.log('2 - beforeAll'));
  afterAll(() => console.log('2 - afterAll'));
  beforeEach(() => console.log('2 - beforeEach'));
  afterEach(() => console.log('2 - afterEach'));
  test('', () => console.log('2 - test'));
});
```

mock

```
// mock ajax
jest.mock('axios', () => ({
  get: jest.fn()
}));
```

```
beforeEach(() => {
  axios.get.mockClear()
  axios.get.mockReturnValue(Promise.resolve({}))
});

jest.mock('axios', () => {
    post: jest.fn.xxx(() => Promise.resolve({
        status: 200
    }))
})
// mock fn
jest.fn(() => {
    // function body
})
```

Vue Test Utils API

挂载组件

- mout(component)
  创建一个包含被挂载和渲染的 Vue 组件的 Wrapper
- createLocalVue
  返回一个 Vue 的类供你添加组件、混入和安装插件而不会污染全局的 Vue 类
  比如要测试vuex、router时使用
- shallowMount
  只挂载一个组件不渲染子组件

参数：

  propsData、slot等

属性

vm(Component)、element(HTMLElement)、Options

方法

设置属性

setProps、setData、setMethods

返回的组件属性

attribus、classes

查找

find、findAll、contains、exits

判断

is、isEmpty、isVisible、isVueInstance

事件触发

trigger、emit、emitOrder

Debugger

运行在node环境下，debugger需要特殊配置

```
# macOS or linux
node --inspect-brk ./node_modules/.bin/vue-cli-service test:unit

# Windows
node --inspect-brk ./node_modules/@vue/cli-service/bin/vue-cli-service.js test:unit
```

快速开始

前置要求

编辑器
node环境
vue工程

安装依赖：

babel-jest(23.1), jest(23.1), vue-jest, @vue/test-utils, identity-obj-proxy

package.json配置，也可以单独提取到jest.config.js

scripts增加："test": "jest"
coverage相关：收集覆盖率信息
moduleFileExtensions：通知jest处理的后缀文件
transform：vue-jest处理vue文件， babel-jest处理js文件
moduleNameMapper：
    处理webpack中的resolve alias,
    处理css、image等静态资源，需要依赖identity-obj-proxy， 创建fileMock.js

```
// fileMock.js
module.exports = 'test-file-stub';
```

```
"jest": {
    "collectCoverage": true,
    "coveragePathIgnorePatterns": [
      "/node_modules/",
      "package.json",
      "package-lock.json"
    ],
    // "collectCoverageFrom": [
    //   "**/*.{js,jsx}",
    //   "!**/node_modules/**",
    //  "!**/vendor/**"
    // ],
    "coverageReporters": [
      "html",
      "text-summary"
    ],
    "moduleFileExtensions": [
      "js",
      "vue"
    ],
    "transform": {
      ".*\\.(vue)$": "<rootDir>/node_modules/vue-jest",
      "^.+\\.js$": "<rootDir>/node_modules/babel-jest"
    },
    "moduleNameMapper": {
      "^@/(.*)$": "<rootDir>/src/$1",
      "\\.(jpg|jpeg|png|gif|eot|otf|webp|svg|ttf|woff|woff2|mp4|webm|wav|mp3|m4a|aac|oga)$": "<rootDir>/tests/mocks/fileMock.js",
      "\\.(css|scss|sass)$": "identity-obj-proxy"
    }
  }
```

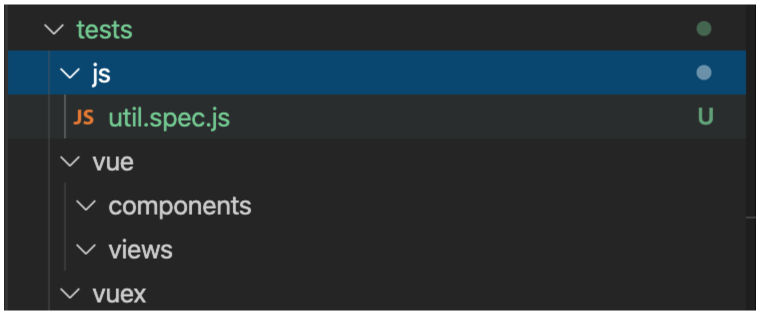
运行

npm install

npm run test

单元测试用例代码

以order-vue为例，新建tests目录， 创建**.spec.js, spec是 Unit Testing Specification的缩写，jest会执行所有.spec.js后缀的文件



js文件

创建对应的**.spec.js， 编码单测用例，例子：

```
import Utils from '@/assets/js/util'
describe('js utils', () => {
  test('isEmpty', () => {
    expect(Utils.isEmpty(null)).toBeTruthy()
    expect(Utils.isEmpty({})).toBeTruthy()
    expect(Utils.isEmpty({a: 11})).toBeFalsy()
  })
})
```

vue组件

步骤：

1. 加载组件
2. 初始化实例
3. 模拟事件
4. 设置data、props
5. 判断props、data是否正确， dom是否值是否符合预期

```
import { shallowMount } from '@vue/test-utils'
import Demo from './demo'

describe('demo vue component', () => {
  test('set props', () => {
    const vm = shallowMount(Demo, {
      // propsData: {
      //   data: 'test_data'
      // }
    })
    // set props
    vm.setProps({
      data: 'test_data'
    })
    // expect(wrapper.isVueInstance()).toBeTruthy()
    expect(vm.props().data).toBe('test_data')
  })
  test('set data', () => {
    const wrapper = shallowMount(Demo)
    wrapper.setData({
      count: 2
    })
    expect(wrapper.vm.count).toBe(2)
  })
  test('trigger increase click', () => {
    const wrapper = shallowMount(Demo)
    const increase = wrapper.find('.increase')
    increase.trigger('click')
    expect(wrapper.vm.count).toBe(1)
  })
  test('trigger reduce click', () => {
    const wrapper = shallowMount(Demo)
    expect(wrapper.vm.count).toBe(0)
    const increase = wrapper.find('.reduce')
    increase.trigger('click')
    expect(wrapper.vm.count).toBe(-1)
  })
})
```

vuex

主要步骤：

- 加载组件
- 初始化实例
- 引用插件vuex
- 伪造store
- 编写测试逻辑

vuex主要元素包括actions、mutation、getter等，可以一个个单独测试，也可以跟随业务组件将 组件内事件调用，执行action，触发mutation，改变state到组件compute变化、dom变化一块处理。

mutation测试

mutation就是一个单独的函数执行，这里举一个order-vue的例子，因为我们项目中的mutation都是后端数据，所以第一步先mock一组数据（直接从网站返回值复制），执行mutation，检查state变化，下面看代码

```
/*
    decorator.spec.js
*/
import decorator from '@/store/decorator'
import userInfo4Web from './userInfo4Web'

describe('vuex decorator', () => {
    test('mutation getUserInfo', () => {
        const state = {
            userName: ''
        }
        decorator.mutations['USER_INFO_REQUEST'](state, userInfo4Web)
        expect(state.userName).toBe('test')
    })
})

/*
    store/decorator.js  列举部分代码
*/
import * as mutation from './mutations'
const state = {
  userName: '',
}
const mutations = {
  [mutation.USER_INFO_REQUEST] (state, payload) {
    state.userName = ''
    let dataResult = payload.body
    if (!Util.isEmpty(dataResult)) {
      state.userName = dataResult.userName
    }
  }
}
const actions = {}
export default {
  state,
  mutations,
  actions
}
```

action测试

用于action只是用来处理状态提交或者处理异步请求，单独测试action只需要测试接口是否调用，是否返回promise等异步，这里不做单独的例子。

getter测试

项目中只是使用全量的store.state， 没有getter的代码，这里先不单独处理getter的例子，后面有需要在处理。

完整的vuex测试

- 引用vuex插件
- 构造action
  因为项目中的action调用的第三方utils调用axios发起请求，成功返回后再执行mutation，这块模拟太复杂，暂时手动构造模拟action，直接在action执行mutation。 通过jest.fn构造一个action方法，再布局替换自己项目工程的action
- 调用action
- 查看state变化

```
import { createLocalVue } from '@vue/test-utils'
import Vuex from 'vuex'
import { cloneDeep } from 'lodash'
import decorator from '@/store/decorator'
import userInfo4Web from './userInfo4Web'

describe('vuex decorator', () => {
    let store
    beforeEach(() => {
        const localVue = createLocalVue()
        localVue.use(Vuex)
        const actions = {
            getUserInfo: jest.fn(({ commit }) => {
                commit('USER_INFO_REQUEST', userInfo4Web)
            })
        }
        const decoratorClone = cloneDeep(decorator)
        // 覆盖actions
        decoratorClone.actions = actions
        store = new Vuex.Store(decoratorClone)
    })
    test('all vuex test', () => {
        expect(store.state.userName).toBe('')
        store.dispatch('getUserInfo')
        expect(store.state.userName).toBe('test')
    })
})
```

vue复杂组件

尝试对order-vue中的header.vue 写一些简单的单元测试，但是header组件其实是包含了一些复杂逻辑在里面的，混合了vuex，初始化获取请求等，这里首先要关注下面几个问题：

- 使用mapState，自定义state的时候，需要在包裹一层

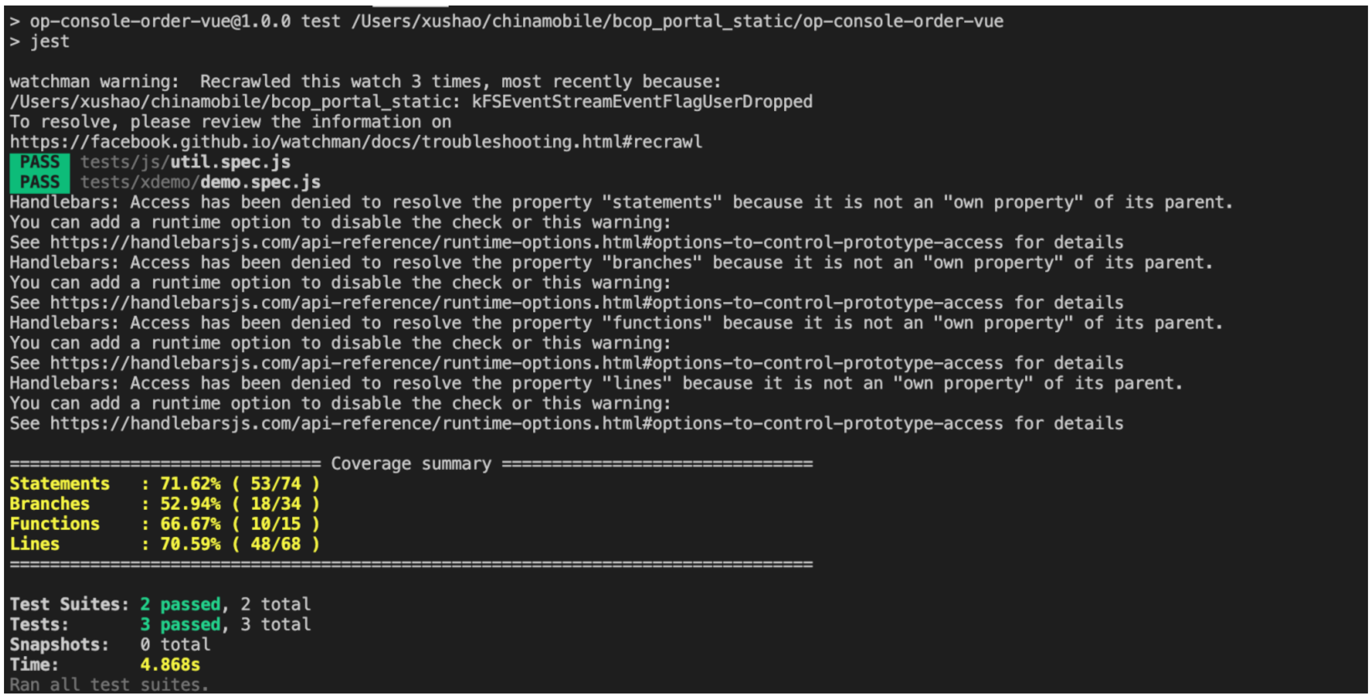
  解决mapState问题，decoratorClone.state.decorator = decoratorClone.state，但是值变化可能有问题
- 组件内依赖element-ui等第三方组件需要处理

  处理element-ui等插件，全量应用组件
- 组件异步请求

  初始化请求问题，首先代码中单独依赖Vue.axios，和localVue不能使用一个实例，使用Vue.use(VueAxios, axios)初始化，然后mock ajax.get/post请求。

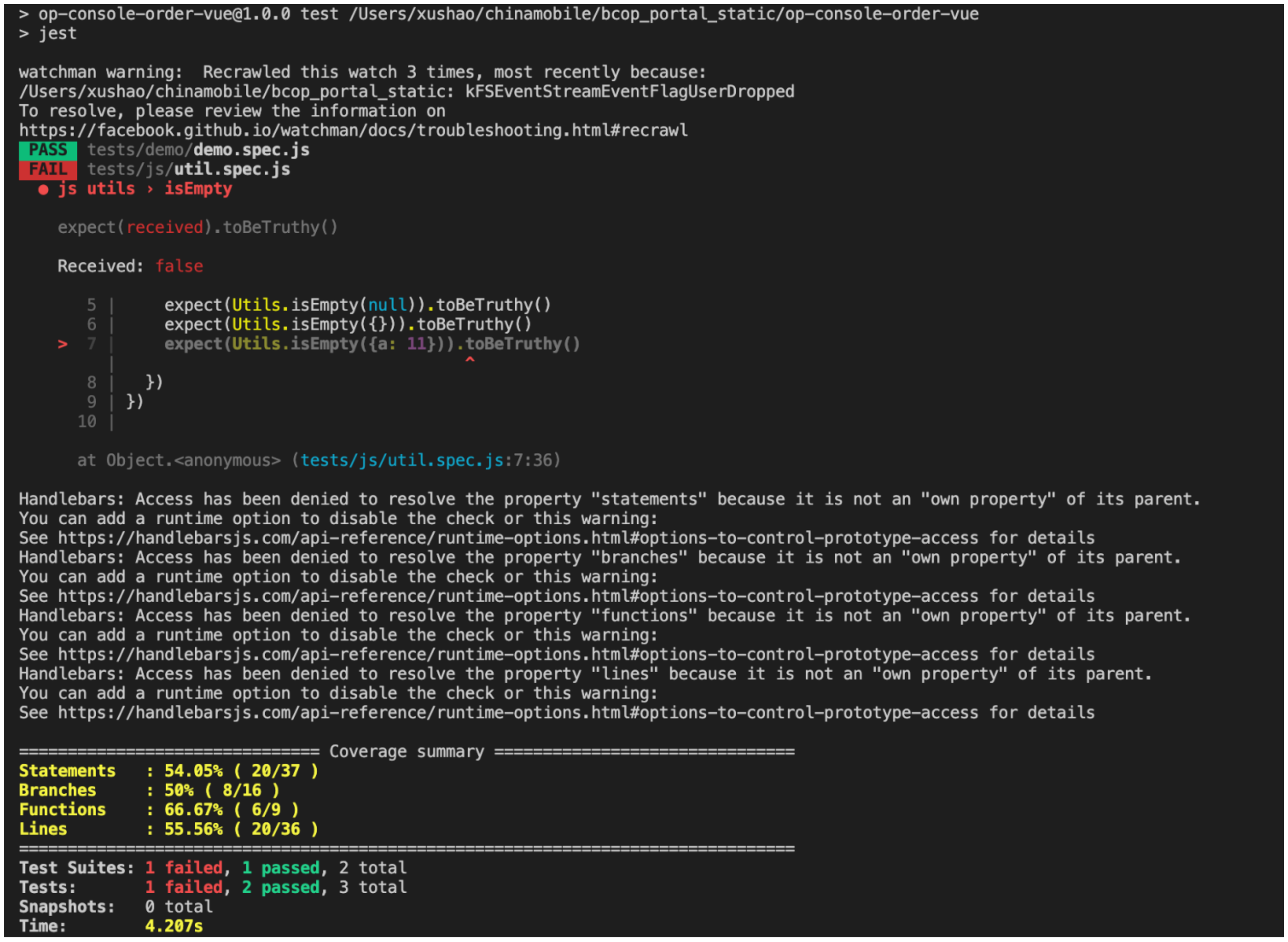
```
import { shallowMount, createLocalVue } from '@vue/test-utils'
import Vuex from 'vuex'
import axios from 'axios'
import VueAxios from 'vue-axios'
import Vue from 'vue'
import { cloneDeep } from 'lodash'
import { Pagination } from 'element-ui'
import SvgIcon from '@/components/common/svgIcon'// svg组件
import decorator from '@/store/decorator'
import userInfo4Web from '../../mocks/userInfo4Web'
import HeaderComponent from '@/components/decorator/header.vue'

jest.mock('axios', () => ({
    get: jest.fn(),
}));
describe('header component test', () => {
    let store
    let localVue
    let wrapper
    beforeAll(async() => {
        localVue = createLocalVue()
        // 引入插件
        Vue.use(VueAxios, axios)
        localVue.use(Vuex)
        localVue.prototype.$ELEMENT = {
            size: 'small'
        }
        localVue.use(Pagination)
        // 省去其他element代码
        localVue.component('svg-icon', SvgIcon)

        const decoratorClone = cloneDeep(decorator)
        // 覆盖actions
        const actions = {
            getUserInfo: jest.fn(({ commit }) => {
                commit('USER_INFO_REQUEST', userInfo4Web)
            })
        }
        Object.assign(decoratorClone.actions, actions)
        decoratorClone.state.decorator = decoratorClone.state
        decoratorClone.state.bigdataAI = {
            IoTMenu: [],
            bigdataAI: []
        }
        store = new Vuex.Store(decoratorClone)
        Vue.axios.get.mockClear()
        Vue.axios.get.mockReturnValue(Promise.resolve({}))

        // wrapper前置，减少重复初始化
        wrapper = await shallowMount(HeaderComponent, { store, local
Vue })
    })

    test("vue data instance", async () => {
        await wrapper.vm.$nextTick()
        expect(wrapper.vm.menuDisplayFlag).toBe('baisc')
    })
    test("user name render", async () => {
        expect(store.state.userName).toBe('test')
        await wrapper.vm.$nextTick()
        expect(wrapper.find('.userName').text()).toBe('test')
    })
})
```

查看结果



```
> op-console-order-vue@1.0.0 test /Users/xushao/chinamobile/bcop_portal_static/op-console-order-vue
> jest

watchman warning:  Recrawled this watch 3 times, most recently because:
/Users/xushao/chinamobile/bcop_portal_static: kFSEventStreamEventFlagUserDropped
To resolve, please review the information on
https://facebook.github.io/watchman/docs/troubleshooting.html#recrawl
 PASS  tests/js/util.spec.js
 PASS  tests/xdemo/demo.spec.js
Handlebars: Access has been denied to resolve the property "statements" because it is not an "own property" of its parent.
You can add a runtime option to disable the check or this warning:
See https://handlebarsjs.com/api-reference/runtime-options.html#options-to-control-prototype-access for details
Handlebars: Access has been denied to resolve the property "branches" because it is not an "own property" of its parent.
You can add a runtime option to disable the check or this warning:
See https://handlebarsjs.com/api-reference/runtime-options.html#options-to-control-prototype-access for details
Handlebars: Access has been denied to resolve the property "functions" because it is not an "own property" of its parent.
You can add a runtime option to disable the check or this warning:
See https://handlebarsjs.com/api-reference/runtime-options.html#options-to-control-prototype-access for details
Handlebars: Access has been denied to resolve the property "lines" because it is not an "own property" of its parent.
You can add a runtime option to disable the check or this warning:
See https://handlebarsjs.com/api-reference/runtime-options.html#options-to-control-prototype-access for details

=============================== Coverage summary ===============================
Statements   : 71.62% ( 53/74 )
Branches     : 52.94% ( 18/34 )
Functions    : 66.67% ( 10/15 )
Lines        : 70.59% ( 48/68 )
================================================================================

Test Suites: 2 passed, 2 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        4.868s
Ran all test suites.
```

错误结果：



```
> op-console-order-vue@1.0.0 test /Users/xushao/chinamobile/bcop_portal_static/op-console-order-vue
> jest

watchman warning:  Recrawled this watch 3 times, most recently because:
/Users/xushao/chinamobile/bcop_portal_static: kFSEventStreamEventFlagUserDropped
To resolve, please review the information on
https://facebook.github.io/watchman/docs/troubleshooting.html#recrawl
 PASS  tests/demo/demo.spec.js
 FAIL  tests/js/util.spec.js
  ● js utils › isEmpty

    expect(received).toBeTruthy()

    Received: false

       5 |     expect(Utils.isEmpty(null)).toBeTruthy()
       6 |     expect(Utils.isEmpty({})).toBeTruthy()
    >  7 |     expect(Utils.isEmpty({a: 11})).toBeTruthy()
         |                                    ^
       8 |   })
       9 | })
      10 |

      at Object.<anonymous> (tests/js/util.spec.js:7:36)

Handlebars: Access has been denied to resolve the property "statements" because it is not an "own property" of its parent.
You can add a runtime option to disable the check or this warning:
See https://handlebarsjs.com/api-reference/runtime-options.html#options-to-control-prototype-access for details
Handlebars: Access has been denied to resolve the property "branches" because it is not an "own property" of its parent.
You can add a runtime option to disable the check or this warning:
See https://handlebarsjs.com/api-reference/runtime-options.html#options-to-control-prototype-access for details
Handlebars: Access has been denied to resolve the property "functions" because it is not an "own property" of its parent.
You can add a runtime option to disable the check or this warning:
See https://handlebarsjs.com/api-reference/runtime-options.html#options-to-control-prototype-access for details
Handlebars: Access has been denied to resolve the property "lines" because it is not an "own property" of its parent.
You can add a runtime option to disable the check or this warning:
See https://handlebarsjs.com/api-reference/runtime-options.html#options-to-control-prototype-access for details

=============================== Coverage summary ===============================
Statements   : 54.05% ( 20/37 )
Branches     : 50% ( 8/16 )
Functions    : 66.67% ( 6/9 )
Lines        : 55.56% ( 20/36 )
================================================================================
Test Suites: 1 failed, 1 passed, 2 total
Tests:       1 failed, 2 passed, 3 total
Snapshots:   0 total
Time:        4.207s
```

遇到的问题

1. collectCoverageFrom设置!node_modules不生效，使用coveragePathIgnorePatterns替换
2. 项目中使用mapstate，vue.axios 需要单独设置
3. Coverage生成的页面结果有问题，待排查
4. jest最新版本24* 默认依赖babel7的版本， vue的文档中并没有说明

op-console-order-vue 是用的是babel6，为了减少配置时间，将jest，babel-jest 版本设置为23

> `babel-jest`。
>
> 我们假设 webpack 使用了 `babel-preset-env`，这时默认的 Babel 配置会关闭 ES modules 的转译，因为 webpack 已经可以处理 ES modules 了。然而，我们还是需要为我们的测试而开启它，因为 Jest 的测试用例会直接运行在 Node 上。
>
> 同样的，我们可以告诉 `babel-preset-env` 面向我们使用的 Node 版本。这样做会跳过转译不必要的特性使得测试启动更快。
>
> 为了仅在测试时应用这些选项，可以把它们放到一个独立的 `env.test` 配置项中 (这会被 `babel-jest` 自动获取)。
>
> `.babelrc` 文件示例：
>
> ```json
> {
>   "presets": [["env", { "modules": false }]],
>   "env": {
>     "test": {
>       "presets": [["env", { "targets": { "node": "current" } }]]
>     }
>   }
> }
> ```

目前存在的问题

vue-test-utils 版本问题
虽然是官方指定的test方案，但是目前仍然停留在1.0beta，令人慌慌的
https://github.com/vuejs/vue-test-utils/issues/1329

相关链接

https://jestjs.io/
https://vue-test-utils.vuejs.org/
https://github.com/puppeteer/puppeteer
https://github.com/vuetifyjs/vuetify
https://github.com/jsdom/jsdom

Q&A