**Meal ordering system**

功能测试报告

测试人员：汪子豪

测试时间：2021年6月1日（第一次）

2021年6月3日（第二次）

目录

1.测试概念

1.1.测试对象

1.2.测试范围

1.3.测试目的

1.4.参考文档

2.单元测试

2.1.测试方法

2.2.测试环境

3.测试过程

4.测试结果

1.测试概念

1.1.测试对象

Meal ordering system是一款基于web开发的，线上实时订餐系统，能够简化多余的广告界面，提供一个方便、快捷的订餐服务。

1.2.测试范围

对项目下视图文件夹下每一个最主要的vue文件进行单元测试。

1.3.测试目的

测试软件系统所提供的各功能点是否达到功能目标;反馈跟踪系统功能实现的缺陷及修复情况;从而提高软件系统的质量，最终满足用户使用需求。

1.4.参考文档

Vue 项目单元测试

https://zhuanlan.zhihu.com/p/48758013

[Vue Test Utils](https://vue-test-utils.vuejs.org/zh/)（官方文档）

https://vue-test-utils.vuejs.org/zh/

2.单元测试

2.1.测试方法

* 搭建基于 jest 的 vue 单元测试环境
* 使用 vue-test-util 提高测试编码效率

jest 包含了 karma + mocha + chai + sinon 的所有常用功能，零配置开箱即用

Jest 是功能最全的测试运行器。它所需的配置是最少的，默认安装了 JSDOM，内置断言且命令行的用户体验非常好。不过你需要一个能够将单文件组件导入到测试中的预处理器。

vue-test-utils在Vue和Jest之间提供了一个桥梁，暴露出一些接口，让我们更加方便地通过Jest为Vue应用编写单元测试

2.2.测试环境

硬件：

处理器：lntel(R) Core(TM) i5-7300U CPU @ 2.60GHz2.71 GHz

机带RAM：8.00 GB

系统类型：64位操作系统,基于x64的处理器

笔和触控：为10触摸点提供笔和触控支持

软件：vscode

环境：jest 、vue-test-utils、 karma + mocha + chai + sinon

3.测试过程：

1.安装对应环境

1. 安装依赖

npm install jest vue-jest babel-jest @vue/test-utils -D

1. 编写 jest 配置文件

*// ./test/unit/jest.conf.js*

**const** path **=** require('path');

module.exports **=** {

rootDir**:** path.resolve(\_\_dirname, '../../'), *// 类似 webpack.context*

moduleFileExtensions**:** [ *// 类似 webpack.resolve.extensions*

'js',

'json',

'vue',

],

moduleNameMapper**:** {

'^@/(.\*)$'**:** '<rootDir>/src/$1', *// 类似 webpack.resolve.alias*

},

transform**:** { *// 类似 webpack.module.rules*

'^.+\\.js$'**:** '<rootDir>/node\_modules/babel-jest',

'.\*\\.(vue)$'**:** '<rootDir>/node\_modules/vue-jest',

},

setupFiles**:** ['<rootDir>/test/unit/setup'], *// 类似 webpack.entry*

coverageDirectory**:** '<rootDir>/test/unit/coverage', *// 类似 webpack.output*

collectCoverageFrom**:** [ *// 类似 webpack 的 rule.include*

'src/\*\*/\*.{js,vue}',

'!src/main.js',

'!src/router/index.js',

'!\*\*/node\_modules/\*\*',

],

};

1. 编写启动文件 setup.js

*// ./test/unit/setup.js*

**import** Vue from 'vue';

Vue.config.productionTip **=** **false**;

1. 加入启动 jest 的 npm script

"scripts": {

"unit": "jest --config test/unit/jest.conf.js --coverage",

},

2.编写对应测试文件

主要思路：通过判断关键的显示成分是否成功显示，得出文件是否正确运行。

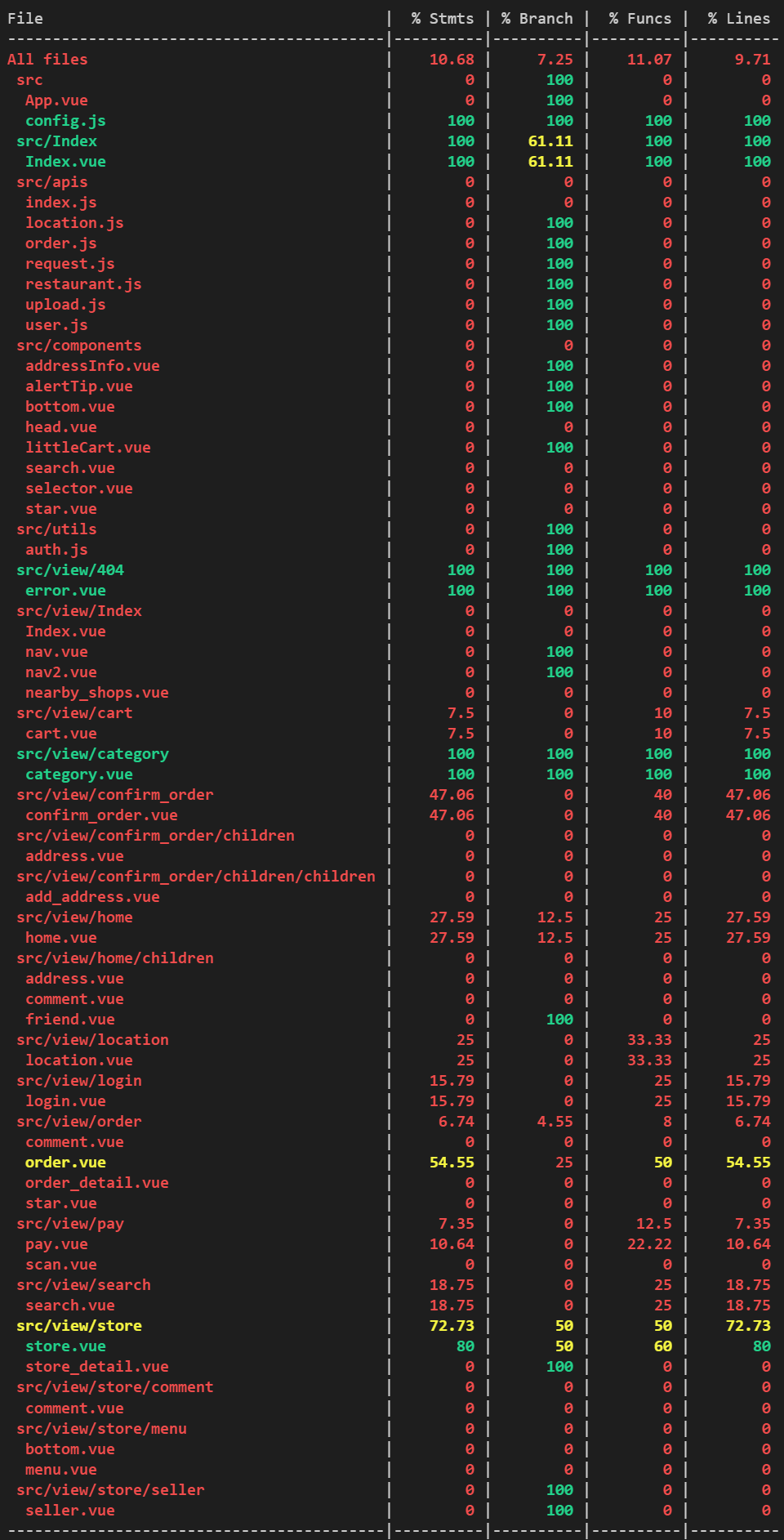
（文件 \*.spec.js）

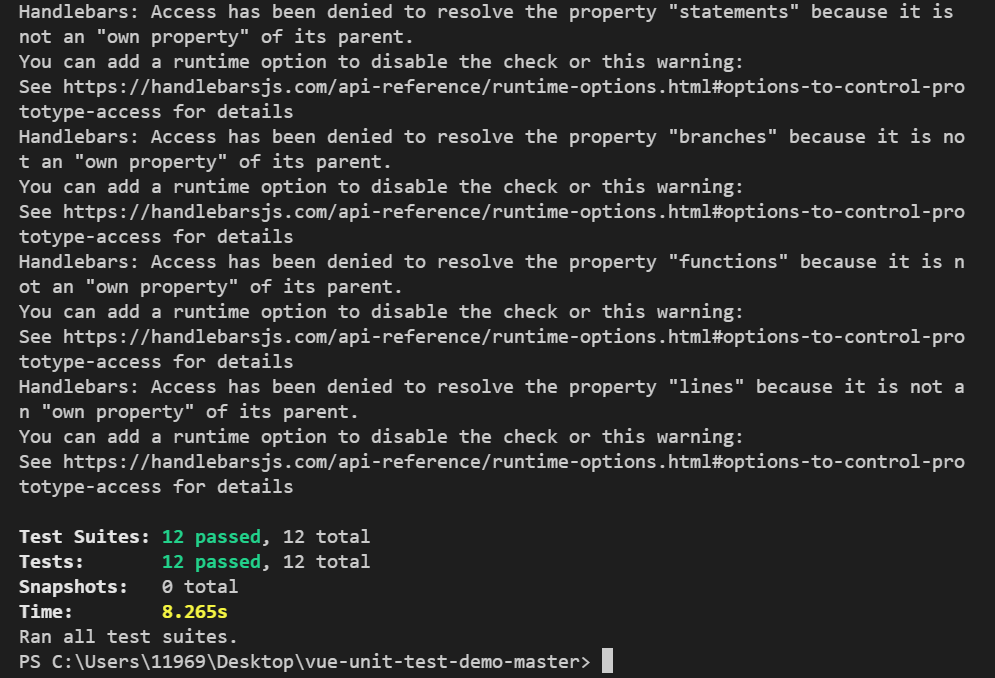
3.得出测试结果，修改有错误的文件

4.得出最终的结果

4.测试结果

a.测试报告



b.测试结论