vue项目单元测试宝典

# 序

## 关于本文

本文的制作初衷是让读者能从单体测试的零基础到独立完成单体测试。

就国内的开发环境而言，要求写测试的项目不多，甚至很少。这也直接导致很大一部分的程序员根本不会写单元测试，但是一个合格的程序员怎么能不会写单元试呢？鉴于此，笔者决定写一本通俗易懂的单元测试宝典。

本文中所提到的测试均是指单元试，e2e本文中不做讨论。

本文是笔者根据个人开发经验总结所著，参考了vue官方测试插件以及很多网络资源和书籍，再此强调，均是原著，所参考资源都是网上公开知识。

## 适用读者

本文的想定读者：对vue的基础知识有一定了解，使用vue作为项目开发框架，不会写单元测试或者从没写过单元测试，且项目需要写单元测试、或者对自己要求较高，想要给自己的代码写单元测试的有志骚年。

## 开发环境（笔者代码环境）

OS：windows10

Vue-Cli：v3.3.0

Node.js: v8.12.0

Npm: v6.4.1

"vue": "^2.6.6",

"vue-router": "^3.0.1",

"vuex": "^3.0.1"

"axios": "^0.18.0",

本文中示例代码为vue SPA项目

github仓库地址：https://github.com/holylovelqq/vue-project-unitTest-with-jest

## 特别说明

1. 免责声明。作者不能保证每个人都能喜欢文中的单元测试方式或者代码，如有不足欢迎指出，实际是否适用请根据项目组或个人需求斟酌使用，对于因使用本文中知识所造成的一切损失，均由使用者自己负责，笔者概不负责
2. 版权所有。本文仅供个人学习所用，未经笔者同意不得用于商业用途，违者必究。

# 为什么要写单元测试

【为什么要写单元测试】【有必要写单元测试么】【单元测试有什么用】这种问题就不在这里详细解释了，只要百度一下就会有很多很多回答，答案肯定是非常建议你写测试，但是你要是就是不写别人也没办法，不过总有一天你还是会写的，不然你不会看本文。

这里借用vue官网上的话（不知道读者朋友们在使用vue的时候有没有注意到）

组件的单元测试有很多好处：

1. 提供描述组件行为的文档
2. 节省手动测试的时间
3. 减少研发新特性时产生的 bug
4. 改进设计
5. 促进重构

自动化测试使得大团队中的开发者可以维护复杂的基础代码。

特别要提出vue有官方的单元测试组件库，如果是没必要的东西，何必花力气来维护一个组件库呢。

另外就个人而言，测试最大的意义在于不希望自己写的代码对别人来说就是一坨shit，这里说的别人包括三个月后的自己，如果自己看到三个月前自己写的代码都要抑制不住的想骂人的话，难道还想让别人夸你么？其实如果只是骂几句的话也不是什么大事，最大的问题是浪费时间、降低效率、破坏心情……花三个月写出来的代码，别人要花四个月才能看明白，这是一种什么操作？说你做的无用功吧，可代码真真就摆在这里的。

说再多其实就是要肯定单元测试的必要性，下面会详细介绍组件的分类和如何写单元测试

# 易于测试的组件与组件分类

## 易于测试的组件

易于测试的组件一般都有如下特征

* 组件内功能尽可能的少
* 组件内的依赖尽可能的少
* 组件尽可能不会根据data或者props而发生大的变化

与上面三点相反就是不易测试的组件，实际开发中应尽量避免

## 组件分类

前端组件分类一般分为两类：Presentation Component和Container Component

* Presentation Component

这类组件的主要功能是DOM展示，数据一般是通过props获取数据，然后展示出来

基本特征：

* 不依赖其他组件，与store和router无关联
* 不进行CRUD操作（如果必要可使用$emit在夫组件内操作）
* 尽量不使用生命周期函数
* 尽量具有可复用性
* Container Component

这类组件的主要功能是数据操作（API联调）

基本特征：

* 子组件的状态改变（props、$emit）
* CRUD操作、store内数据操作、router切换

以上这两类组件是项目开发中定义组件的思想，实际项目中区分这两种组件还是有一定的困难的，主要还是要看项目的规模及组件的使用，下一章文件树结构会更详细的按照文件夹内所存放的组件来讲解

# 文件树的构成

笔者一般时候的文件树构成如下

-src

-assets

-basics

-components

-containers

-pages

-App.vue

.

.

.

下面相应讲解一下每个文件夹内对应的组件类别

Bsics文件夹内存放的一般是最小的组件单元和可复用的组件，比如：按钮组件AppButton.vue、输入框组件AppInput.vue 等等，属于presentation component

Components文件夹内存放的是集合了basics或者components组件的父组件，属于presentation component

Containers文件夹内存放的是比自身组件小的组件basics或者components组件的父组件，主要是给子组件传递props等数据，属于containers component

Pages文件夹内存放的是页面组件，内涵页面内所有子组件，主要是操作数据和路由切换，属于containers component

# vue项目中哪些需要测试

Vue组件的基本模板主要有三大标签组成：template，script，Style（字符串模板不在讨论范围内），理论上来讲，这三大标签内的所有项都应该进行测试，但是本文涉及的测试是使用jest对template和script两大标签的内容进行测试，如果要测试style内的CSS样式的话，可以使用storybook，不在本文的讨论范围内，有兴趣的可以单独联系

下面就详细的介绍两大标签内可能会出现的内容

## Template标签

本标签内主要是DOM结构，可以使用jest的snapshot功能进行测试

v-on

v-bind

v-slot（vue2.6.0+）

v-if

filter

测试要配合script标签的内容进行

## Script标签

本标签内主要是逻辑实现可能出现的内容有

Props

Data

Filters

Methods

$emit

eventHub

create

mounted

watch

computed

router

axios

vuex

其他插件：momentJS等

# 从零开始写测试

## 安装开发环境

参见序中的开发环境，示例代码再此环境下完美执行，建议各软件版本不低于此开发环境，最好是相同版本，避免可能出现的版本不同导致的bug

## 创建项目

本文假定您已了解vue的基础知识，并能够使用vuecli创建项目，记得选中Unit Testing并选择Jest。

（如确实不会的话，参见：<https://cli.vuejs.org/zh/guide/creating-a-project.html#vue-create>）

且读到此处默认您已经阅读完了第二三章，并对已有的项目目录进行了调整，对组件也进行了相应的分割（如果没有建议您看一下第二三章，或者看一下笔者的github代码），虽然您不进行这些操作也不影响写测试。但是您的代码可能会看起来像一坨shit，包括测试代码。当然您也可以边写测试边分割组件，如果您大脑够清晰的话。不管如何，建议您先将项目结构清晰化，组件尽可能地优化，易于测试

# 可能的需要测试项目及测试方法（如有不足欢迎在github提issue）