1. Jest和vue-test-utils。

他们都是单元测试工具。vue-test-utils可以对组件进行测试。Jest更像是一个库，vue-test-utils更像是jest的一个插件。vue-test-utils可以在jest下写很多vue 的测试。

他们的官方文档地址。

https://jestjs.io/docs/zh-Hans/getting-started

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

2. 开始jest之前，需要一些基本配置

1.安装依赖@babel/core, @babel/preset-env

2. .babelrc基本配置,使测试代码支持ES6模块

{

"presets": [

["@babel/preset-env",{

"targets": {

"node": "current"

}

}]

]

}

3. 安装git

3. 演示一些基本的匹配器

4. 演示jest交互式命令行工具. jest记录文件的改变依赖于git. 模式并且需要使用命令git add 跟踪文件后，才会生效。 —watch 默认是o模式。

(代码2-7)

5.异步代码测试:异步代码有两种带回调函数的异步，可以直接用done确保expect 被执行。多余promise类型的异步，则需要在then或者catch中执行expect。如果需要同时对then和catch进行测试。则不能同时满足。最好的办法是用mock对每次测试用例的返回值单独进行测试。

(代码2-9, 2-10)

6.声明周期函数

钩子函数的作用域: describe外层beforeAll -> describe内层beforeAll -> describe外层beforeEach -> describe内层beforeEach -> describe内层afterAll (describe如果后面还有分组: describe外层beforeEach -> describe内层beforeEach -> describe内层afterAll) -> describe外层afterAll。

不要在describe内部直接写公用代码，而是写在钩子函数内部。

(代码2-11)

7. mock.fn() ，

使用方法: mockCallback = jest.fn(x => 42 + x)

生成一个被包装过的回调函数

ockCallback.mock 里面的属性翻记录了mockCallback被调用相关的记录

(代码3-1)

8.快照测试

快照测试适合测试，配置文件。和vue ui组件。快照会保存第一次测 试的配置

文件,每当修改配置文件, 就会测试不通过。需要通过交互式的命令 行输入,

新快照。也就是说快照测试只是对比测试，对比不一致就报错.

对于频繁变更的字段，如 date: new Date。可以增加配置参数，

expect.any(数据类型),来使其自动更新快照.

行内快照需要安装 prettier, 会把快照内容直接生成在test文件 中

9. mock另外还有一种在\_\_mocks\_\_目录下新建,与被测试文件同名的文件, 并且在里面定义同名的函数,此函数内部，

通过Promise模拟网络请求。

使用方法一: 1.jest.mock('./mock.js') 引入mock文件,

2. import {fetchData} from './mock'; 引入被测试文件

3. 写fetchData的测试代码

这种写法，有个弊端，就是对原函数(fetchData)的内部进行侵入性的模拟。如果被此时文件存在同步函数，因为同步函数不需要模拟，就不需要在./\_\_mocks\_\_/mock.js文件中也要对其进行定义。在引入同步方法时要使用, const {xxxx} = jest.requireActual('./xxx');

使用方法二: 1. 吧把方法一的第一步改成，在jest.config.js 文件中修改 automock 为true

其他相同

10. timerMock

jest.useFakeTimers();

jest.runAllTimers();

jest.runOnlyPendingTimers();

jest.advanceTimersByTime()