自动化测试

# 使用 Karma + Mocha做单元测试

1. Karma（`[ˈkɑrmə]` 卡玛）是一个测试运行器，它可以呼起浏览器，加载测试脚本，然后运行测试用例

2. Mocha（`[ˈmoʊkə]` 摩卡）是一个单元测试框架/库，它可以用来写测试用例

3. Sinon（西农）是一个 spy / stub / mock 库，用以辅助测试（使用后才能理解）

## 步骤

1. 安装各种工具

`npm i -D karma karma-chrome-launcher karma-mocha karma-sinon-chai mocha sinon sinon-chai karma-chai karma-chai-spies`

2. 创建 karma 配置

// 新建 karma.conf.js，内容如下

module.exports = function (config) {

config.set({

// base path that will be used to resolve all patterns (eg. files, exclude)

basePath: '',

// frameworks to use

// available frameworks: https://npmjs.org/browse/keyword/karma-adapter

frameworks: ['mocha', 'sinon-chai'],

client: {

chai: {

includeStack: true

}

},

// list of files / patterns to load in the browser

files: [

'dist/\*\*/\*.test.js',

'dist/\*\*/\*.test.css'

],

// list of files / patterns to exclude

exclude: [],

// preprocess matching files before serving them to the browser

// available preprocessors: https://npmjs.org/browse/keyword/karma-preprocessor

preprocessors: {},

// test results reporter to use

// possible values: 'dots', 'progress'

// available reporters: https://npmjs.org/browse/keyword/karma-reporter

reporters: ['progress'],

// web server port

port: 9876,

// enable / disable colors in the output (reporters and logs)

colors: true,

// level of logging

// possible values: config.LOG\_DISABLE || config.LOG\_ERROR || config.LOG\_WARN || config.LOG\_INFO || config.LOG\_DEBUG

logLevel: config.LOG\_INFO,

// enable / disable watching file and executing tests whenever any file changes

autoWatch: true,

// start these browsers

// available browser launchers: https://npmjs.org/browse/keyword/karma-launcher

browsers: ['ChromeHeadless'],

// Continuous Integration mode

// if true, Karma captures browsers, runs the tests and exits

singleRun: false,

// Concurrency level

// how many browser should be started simultaneous

concurrency: Infinity

})

}

3. 创建 test/button.test.js 文件

const expect = chai.expect;

import Vue from 'vue'

import Button from '../src/button'

Vue.config.productionTip = false

Vue.config.devtools = false

describe('Button', () => {

it('存在.', () => {

expect(Button).to.be.ok

})

it('可以设置icon.', () => {

const Constructor = Vue.extend(Button)

const vm = new Constructor({

propsData: {

icon: 'settings'

}

}).$mount()

const useElement = vm.$el.querySelector('use')

expect(useElement.getAttribute('xlink:href')).to.equal('#i-settings')

vm.$destroy()

})

it('可以设置loading.', () => {

const Constructor = Vue.extend(Button)

const vm = new Constructor({

propsData: {

icon: 'settings',

loading: true

}

}).$mount()

const useElements = vm.$el.querySelectorAll('use')

expect(useElements.length).to.equal(1)

expect(useElements[0].getAttribute('xlink:href')).to.equal('#i-loading')

vm.$destroy()

})

it('icon 默认的 order 是 1', () => {

const div = document.createElement('div')

document.body.appendChild(div)

const Constructor = Vue.extend(Button)

const vm = new Constructor({

propsData: {

icon: 'settings',

}

}).$mount(div)

const icon = vm.$el.querySelector('svg')

expect(getComputedStyle(icon).order).to.eq('1')

vm.$el.remove()

vm.$destroy()

})

it('设置 iconPosition 可以改变 order', () => {

const div = document.createElement('div')

document.body.appendChild(div)

const Constructor = Vue.extend(Button)

const vm = new Constructor({

propsData: {

icon: 'settings',

iconPosition: 'right'

}

}).$mount(div)

const icon = vm.$el.querySelector('svg')

expect(getComputedStyle(icon).order).to.eq('2')

vm.$el.remove()

vm.$destroy()

})

it('点击 button 触发 click 事件', () => {

const Constructor = Vue.extend(Button)

const vm = new Constructor({

propsData: {

icon: 'settings',

}

}).$mount()

const callback = sinon.fake();

vm.$on('click', callback)

vm.$el.click()

expect(callback).to.have.been.called

})

})

3. 创建测试脚本

在 package.json 里面找到 scripts 并改写 scripts

"scripts": {

"dev-test": "parcel watch test/\* --no-cache & karma start",

"test": "parcel build test/\* --no-minify && karma start --single-run"

},

4. 运行测试脚本

1. 要么使用 `npm run test` 一次性运行

![截图](//video.jirengu.com/xdml/image/3ac7c224-c23d-491f-84b5-4fabfbeab9b8/2018-6-30-18-25-41.png)

2. 要么使用 `npm run dev-test` 进行 watch 运行

![截图](//video.jirengu.com/xdml/image/3ac7c224-c23d-491f-84b5-4fabfbeab9b8/2018-6-30-18-26-13.png)

## 成果

如此一来，你开发的时候新开一个命令行窗口运行 `npm run dev-test` 就可以实时查看测试结果。

如果你只想看一次结果，就只用运行 `npm run test`