---

link: null
title: 珠峰架构师成长计划
description: 修改js模块功能，其它模块也受影响，很难快速定位bug多人开发代码越来越难以维护,不方便迭代,代码无法重构
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=67 sentences=163, words=1362

前端常见问题

修改js模块功能，其它模块也受影响，很难快速定位bug
多人开发代码越来越难以维护,不方便迭代,代码无法重构

什么是单元测试

单元测试就是测试最小单元(一个方法，一个组件)

TDD & BDD

- Test-Driven Development, 测试驱动开发
    - 先编写测试用例代码，然后针对测试用例编写功能代码，使其能够通过
    - 很好的诠释了代码即文档
    - 清晰地了解软件的需求
- Behavior Driven Development，行为驱动开发
    - 系统业务专家、开发者、测试人员一起合作，分析软件的需求，然后将这些需求写成一个个的故事。开发者负责填充这些故事的内容
    - 保证程序实现效果与用户需求一致。

测试工具 mocha + chai / jest

- karma + Jasmine + chrome-launcher
- karma + mocha + chai / jest 使用jsdom

Karma为前端自动化测试提供了跨浏览器测试的能力,Mocha是前端自动化测试框架，测试框架需要解决兼容不同风格断言库,jest 是facebook推出的一款测试框架,集成了 Mocha,chai,jsdom,sinon等功能。

创建项目使用mocha+chai

```
vue create mocha-vue
// &#x589E;&#x52A0;setup&#x6587;&#x4EF6;
global.performance = window.performance;
// &#x914D;&#x7F6E;&#x542F;&#x52A8;&#x6587;&#x4EF6;
vue-cli-service test:unit -r tests/setup.js
```

测试第一个例子

```
export let parser = (str) =>{
    let obj = {};
    str.replace(/([^&=]*)=([^&=]*)/g,function(){
        obj[arguments[1]] = arguments[2];
    });
    return obj;
}
export let stringify = (obj) =>{
    let arr = [];
    for(let key in obj){
        arr.push(`${key}=${obj[key]}`);
    }
    return arr.join('&');
}
// &#x6D4B;&#x8BD5;parser &#x548C; stringify
console.log(parser('name=zfpx'));
console.log(stringify({name:'zfpx'}));
```

> 用例无法保存，污染代码，不直观，所有的用例全部混在一起

编写测试用例

会默认测试tests文件夹下.spec 和 .test文件

```
import { expect } from 'chai'
import {parser,stringify} from '@/code/parser';
describe('&#x6D4B;&#x8BD5;parser.js&#x662F;&#x5426;&#x9760;&#x8C31;', () => {
  it('parser&#x6D4B;&#x8BD5;',()=>{
    expect(parser('name=zfpx&age=9')).to.be.deep.eq({name:'zfpx',age:'9'});
  });
  it('stringify&#x6D4B;&#x8BD5;',()=>{
    expect(stringify({name:'zfpx'})).to.be.eq('name=zfpx');
  })
});
```

chai库的应用

```
import { expect } from 'chai'
describe('&#x6D4B;&#x8BD5;&#x5E38;&#x89C1;&#x7684;&#x6BD4;&#x8F83;&#x65B9;&#x6CD5;', () => {
  it('&#x5E38;&#x89C1;&#x5224;&#x65AD;&#x65B9;&#x5F0F;',()=>{
    expect('zfpx').to.be.equal('zfpx');
    // &#x7B80;&#x5199;
    expect(true).to.be.equal(true);
    expect(true).to.be.true;
    expect([1,2,3]).to.be.lengthOf(3);
    // &#x5224;&#x65AD;&#x5305;&#x542B;
    expect('welcome zf').to.contain('zf');
    expect({name:'zfpx'}).to.not.have.haveOwnPropertyDescriptor('address');
    expect('zfpx').match(/zf/);
    // &#x5927;&#x4E8E;&#x5C0F;&#x4E8E;
    expect(5).to.be.lessThan(6);
    expect(3).to.be.greaterThan(2);
  });
});
```

测试vue组件

```
import { expect } from 'chai';
import HelloWorld from '@/components/HelloWorld';
import Vue from 'vue';
describe('&#x6D4B;&#x8BD5;vue&#x7EC4;&#x4EF6;', () => {
  it('&#x6D4B;&#x8BD5;&#x5C5E;&#x6027;&#x662F;&#x5426;&#x6B63;&#x786E;',()=>{
    let baseExtend = Vue.extend(HelloWorld);
    let vm = new baseExtend({
      propsData:{msg:'zfjg'}
    }).$mount();
    expect(vm.$el.querySelector('h1').innerHTML).to.contain('zfjg');
  })
});
```

需要自己挂载组件,而且还需要自己查找dom元素，很麻烦

使用 Vue Test Utils简化流程

```
describe('&#x6D4B;&#x8BD5;vue&#x7EC4;&#x4EF6;', () => {
  it('&#x6D4B;&#x8BD5;&#x5C5E;&#x6027;&#x662F;&#x5426;&#x6B63;&#x786E;',()=>{
    let wrapper = shallowMount(HelloWorld,{
      propsData:{msg:'zfjg'}
    });
    expect(wrapper.find('h1').text()).to.be.contain('zfjg');
  })
});
```

单元测试事件的触发

点击事件

```
<template>
    <div>
        <span id="count">{{count}}</span>
        <button @click="increment">&#x70B9;&#x51FB;</button>
    </div>
</template>
<script>
export default {
    data(){
        return {count:10}
    },
    methods:{
        increment(){
            this.count++;
        }
    }
}
</script>

// &#x5BF9;&#x5E94;&#x5355;&#x5143;&#x6D4B;&#x8BD5;
import {shallowMount} from '@vue/test-utils';
import Counter from '@/components/Counter';
import {expect} from 'chai';
describe('Conter&#x7EC4;&#x4EF6;',()=>{
    it('&#x70B9;&#x51FB;&#x6309;&#x94AE;&#x662F;&#x5426;&#x53EF;&#x4EE5;&#x52A0;1',()=>{
        // &#x6302;&#x8F7D;counter&#x7EC4;&#x4EF6;
        let wrapper = shallowMount(Counter);
        wrapper.setData({count:10}); // &#x8BBE;&#x7F6E;&#x72B6;&#x6001;
        // mock&#x72B6;&#x6001;
        expect(wrapper.find('span').text()).to.be.equal('10');
        wrapper.find('button').trigger('click');
        expect(wrapper.find('span').text()).to.be.equal('11');
    });
});
```

自定义事件

```
<template>
    <div>
        <child @show="show"></child>
        <p v-if="flag"> {{name}} </p>
    </div>
</template>
<script>
import Child from './Child.vue';
export default {
    data(){
        return {name:'英文',flag:false}
    },
    methods:{
        show(){
            this.flag = true;
        }
    },
    components:{
        Child
    }
}
</script>

// &#x6D4B;&#x8BD5;&#x7528;&#x4F8B;
import {shallowMount} from '@vue/test-utils';
import Parent from '@/components/Parent';
import Child from '@/components/Child';
import {expect} from 'chai';

describe('&#x6D4B;&#x8BD5;&#x5B50;&#x7EC4;&#x4EF6;&#x80FD;&#x5426;&#x89E6;&#x53D1;&#x7236;&#x7EC4;&#x4EF6;&#x65B9;&#x6CD5;',()=>{
    it('&#x89E6;&#x53D1;show&#x65B9;&#x6CD5;',()=>{
        let wrapper = shallowMount(Parent);
        expect(wrapper.find('p').exists()).to.be.false;
        wrapper.find(Child).vm.$emit('show');
        expect(wrapper.find('p').exists()).to.be.true;
    })
});
```

sinon应用

模拟函数调用,统计函数调用次数和调用时的参数

```
npm install sinon
```

```
<template>
    <div>
        <button @click="handleClick">&#x70B9;&#x6211;&#x554A;</button>
    </div>
</template>
<script>
export default {
    props:{
        fn:{}
    },
    methods:{
        handleClick(){
            this.fn('hello','world');
        }
    }
}
</script>

// &#x6D4B;&#x8BD5;
import {shallowMount} from '@vue/test-utils';
import PropFn from '@/components/PropFn.vue';
import {expect} from 'chai';
import sinon from 'sinon';
describe('&#x6D4B;&#x8BD5;propFn&#x7EC4;&#x4EF6;',()=>{
    it('&#x5224;&#x65AD;&#x70B9;&#x51FB;&#x6309;&#x94AE;&#x65F6;&#x662F;&#x5426;&#x53EF;&#x4EE5;&#x89E6;&#x53D1;&#x51FD;&#x6570;&#x8C03;&#x7528;',()=>{
        let callback = sinon.spy();
        let wrapper = shallowMount(PropFn,{
            propsData:{
                fn:callback
            }
        });
        wrapper.find('button').trigger('click');
        expect(callback.callCount).to.be.equal(1);
        expect(callback.getCall(0).args).to.be.lengthOf(2);
    });
});
```

使用moxios

在mocha中测试axios

```
npm install moxios
```

```
<template>
    <div>
        {{user}}
    </div>
</template>
<script>
import axios from 'axios';
export default {
    data(){
        return {user:''}
    },
    mounted(){
        axios.get('/user').then((res)=>{
            this.user = res.data.user;
        }).catch(res=>{
            console.log(res);
        });
    }
}
</script>

// &#x5355;&#x5143;&#x6D4B;&#x8BD5;
import {shallowMount} from '@vue/test-utils';
import Axios from '@/components/Axios';
import {expect} from 'chai';
import moxios from 'moxios';

describe('&#x6D4B;&#x8BD5;Axios.vue&#x7EC4;&#x4EF6;',()=>{
    beforeEach(()=>{
        moxios.install();
    });
    afterEach(()=>{
        moxios.uninstall();
    });
    it('&#x4F7F;&#x7528;mixios&#x6A21;&#x62DF;&#x63A5;&#x53E3;',(done)=>{
        let wrapper = shallowMount(Axios);
        moxios.stubRequest('/user',  {
            status: 200,
            response: {user:'jw'}
        });
        moxios.wait(function () {
            expect(wrapper.text()).to.include('jw');
            done();
        });
    })
});
```

转向jest单元测试

```
// &#x76F8;&#x7B49;
expect(1+1).toBe(2);
expect({name:'zfpx'}).toEqual({name:'zfpx'});
// &#x5305;&#x542B;
expect('zfpx').toContain('zf');
expect({a:1}).toHaveProperty('a');
expect('zf').toMatch('z');
// &#x8DEF;&#x7531;&#x8FD0;&#x7B97;
expect(5).toBeLessThan(10);
expect(5).toBeGreaterThan(12);
```

模拟ajax请求方法

```
&#x589E;&#x52A0;__mocks__&#x6587;&#x4EF6;&#x5939;
export default {
    get:()=>Promise.resolve({data:{user:'zfpx'}})
}

import { shallowMount } from '@vue/test-utils'
import Axios from '@/components/Axios.vue'
import Vue from 'vue';
jest.mock('axios');

describe('Axios.vue', () => {
  it('&#x6A21;&#x62DF;ajax&#x8BF7;&#x6C42;', () => {
    const wrapper = shallowMount(Axios);
    return Vue.nextTick().then(()=>{
      expect(wrapper.text()).toContain('jw');
    })
  })
})
```

模拟vuex的状态和action测试组件

```
<template>
    <div>
        <span>{{username}}</span>
        <button @click="change()">&#x66F4;&#x6539;&#x7528;&#x6237;&#x540D;</button>
    </div>
</template>
<script>
import {mapMutations,mapActions} from 'vuex';
export default {
    computed:{
        ...mapState(['username'])
    },
    methods:{
        ...mapActions(['changeUsername']),
        change(){
            this['changeUsername']('newUsername');
        }
    }
}
</script>

// &#x6D4B;&#x8BD5;action
import { shallowMount , createLocalVue} from '@vue/test-utils'
import VuexComponent from '@/components/Vuex.vue';
import Vuex from 'vuex';
const localVue = createLocalVue()
localVue.use(Vuex)
describe('&#x6D4B;&#x8BD5;&#x7EC4;&#x4EF6;&#x4E2D;&#x7684;vuex', () => {
  let state;
  let actions;
  let store;
  let fn = jest.fn();
  beforeEach(() => {
    state = {username:'jw'}
    actions = {change_username:fn}
    store = new Vuex.Store({
      state,
      actions
    })
  })
  it('&#x6D4B;&#x8BD5;state&#x662F;&#x5426;&#x6B63;&#x786E;', () => {
      let wrapper = shallowMount(VuexComponent,{
        store,
        localVue
      });
      expect(wrapper.find('span').text()).toContain('jw');
  });
  it('&#x6D4B;&#x8BD5;action&#x662F;&#x5426;&#x6B63;&#x786E;&#x8C03;&#x7528;',()=>{
      let wrapper = shallowMount(VuexComponent,{
        store,
        localVue
      });
      wrapper.find('button').trigger('click');
      expect(fn.mock.calls[0][1]).toEqual('newUsername');
      expect(fn).toHaveBeenCalled()
  })
})
```

```
export default {
  increment(state) {
    state.count++
  }
}
it('&#x52A0;1', () => {
  const state = {
    count: 0
  }
  mutations.increment(state)
  expect(state.count).toBe(1)
})
```

```
import {createLocalVue} from '@vue/test-utils'
import Vuex from 'vuex';
import config from  '@/store.js'
import cloneDeep from 'lodash/cloneDeep'

describe('&#x6D4B;&#x8BD5;&#x7EC4;&#x4EF6;&#x4E2D;&#x7684;vuex', () => {
  it('&#x6D4B;&#x8BD5;store',()=>{
    const localVue = createLocalVue();
    localVue.use(Vuex);
    let store = new Vuex.Store(cloneDeep(config));
    expect(store.state.username).toEqual('zfpx');
    store.commit('set_username','jw');
    expect(store.state.username).toEqual('jw');
  });
});
```