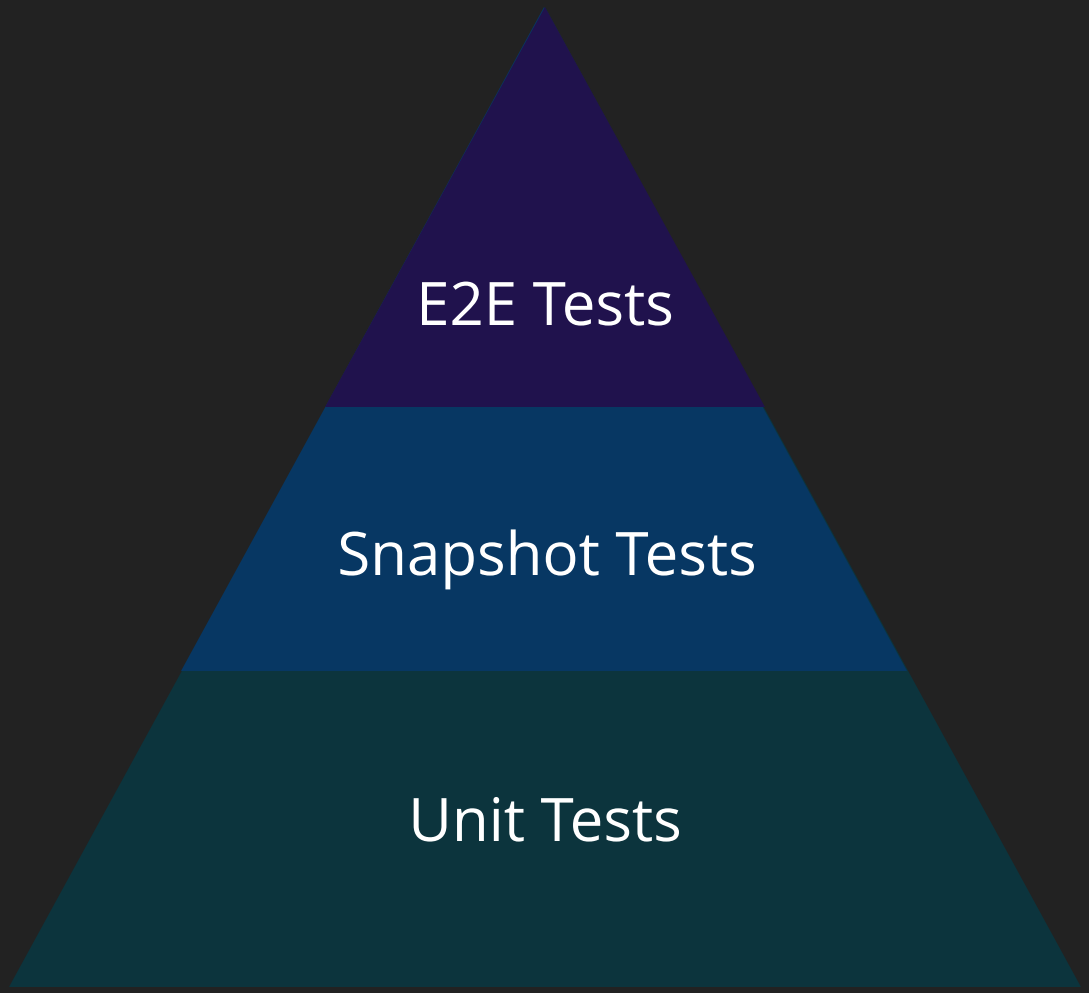
Vue.js 单元测试之旅

Jay Lu


E2E Tests
Snapshot Tests
Unit Tests

Demo

从零开始写一个单元测试

```
npm install --save-dev jest
```

sum.js

```
function sum(a, b) {
  return a + b;
}
module.exports = sum;
```

sum.test.js

```
const sum = require('./sum');

test('adds 1 + 2 to equal 3', () => {
  expect(sum(1, 2)).toBe(3);
});
```

package.json

```
{
  "scripts": {
    "test": "jest"
  }
}
```

```
npm run test

 PASS  jest-basic/sum.spec.js
  ✓ adds 1 + 2 to equal 3 (3ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.781s, estimated 1s
```

```
npm run test --coverage

 PASS  jest-basic/sum.spec.js
  ✓ adds 1 + 2 to equal 3 (3ms)

----------|----------|----------|----------|----------|-------------------|
File      |  % Stmts | % Branch |  % Funcs |  % Lines | Uncovered Line #s |
----------|----------|----------|----------|----------|-------------------|
All files |      100 |      100 |      100 |      100 |                   |
 sum.js   |      100 |      100 |      100 |      100 |                   |
----------|----------|----------|----------|----------|-------------------|

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.781s, estimated 1s
```

单元测试的问题？

我们的经历


HSBC Evolve
Winner of best e-FX platform for corporates
Winner
FX Week
e-FX
Awards
2017

2014 - 24 releases

2015 - 640 releases

2016 - 5000 releases

2017 - 9000 releases

2018 - 12000 releases

2019 - keep going ...

单元测试对我们很重要

- 有保障，自然更有信心更改代码
- 单元测试是很好的文档
- 把时间花在有意义的事情上

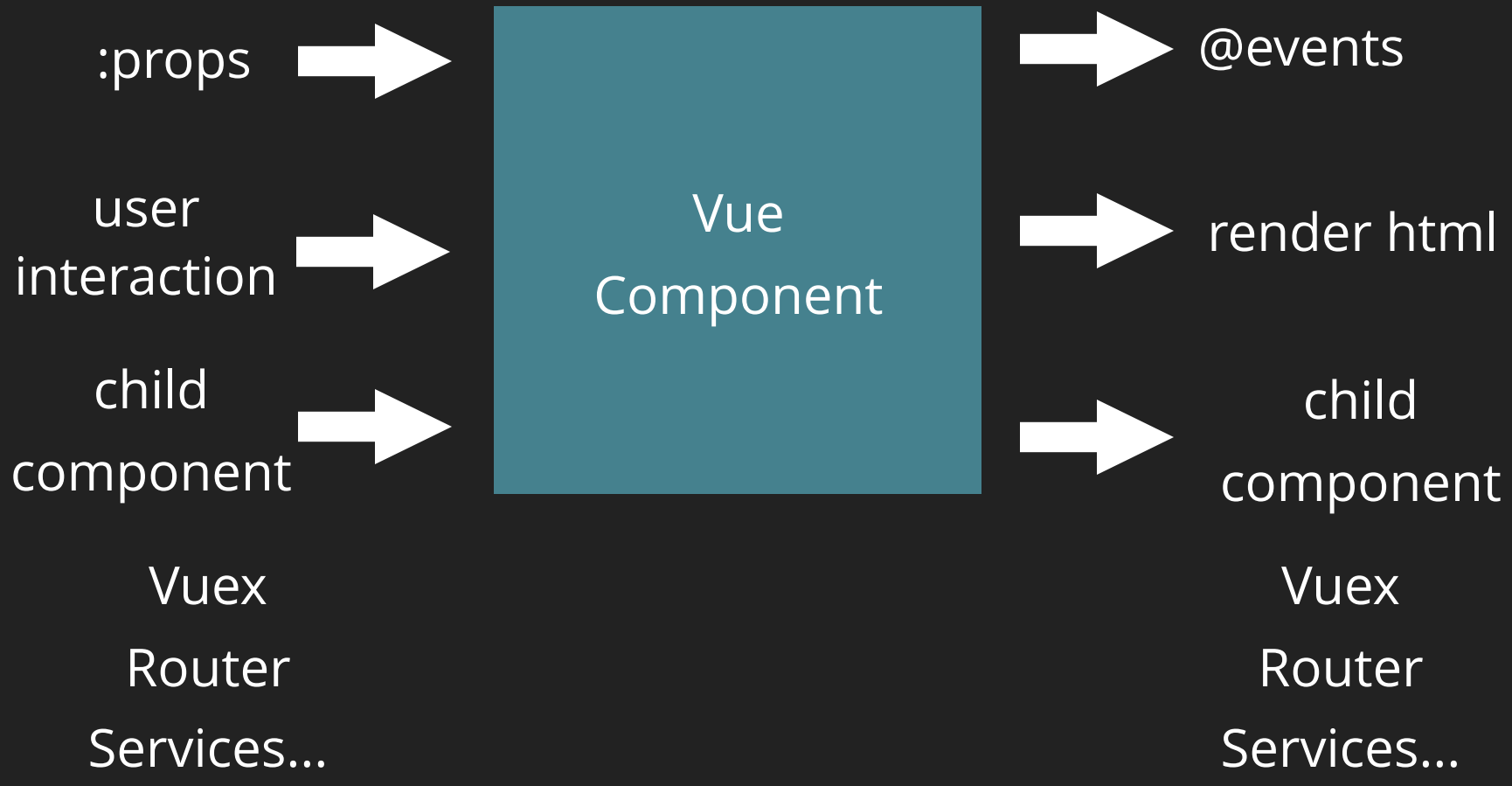
单元测试相关的库

syntax, runner, matcher, mock...

Vue.js 单元测试

+

Vue组件输入和输出


:props
user interaction
child component
Vuex
Router
Services...
Vue Component
@events
render html
child component
Vuex
Router
Services...

Home | Message | Search

New Message

Ken

Oh...

Submit

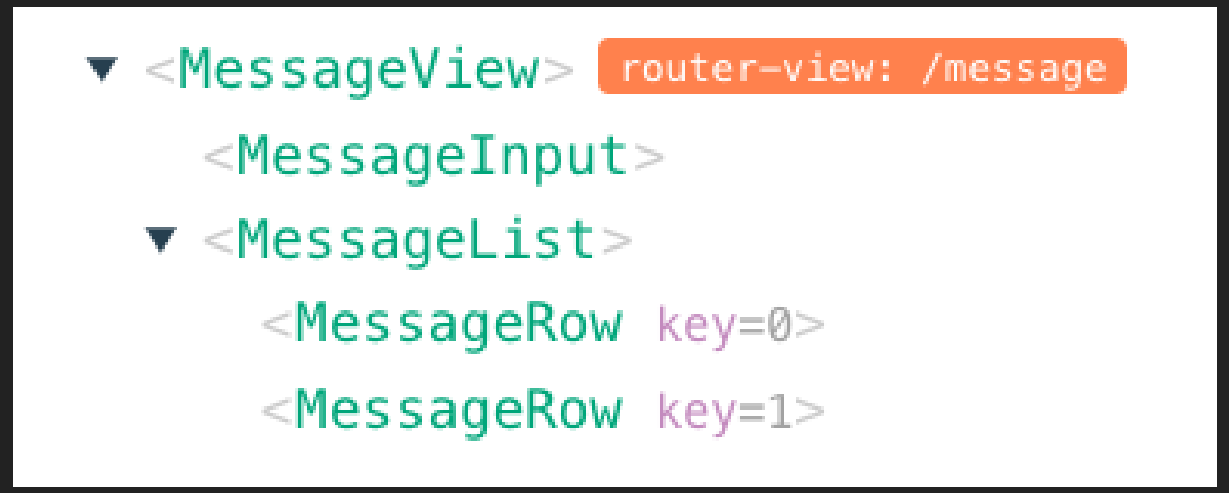
Message List

Ken 2019-05-21 09:28 said:

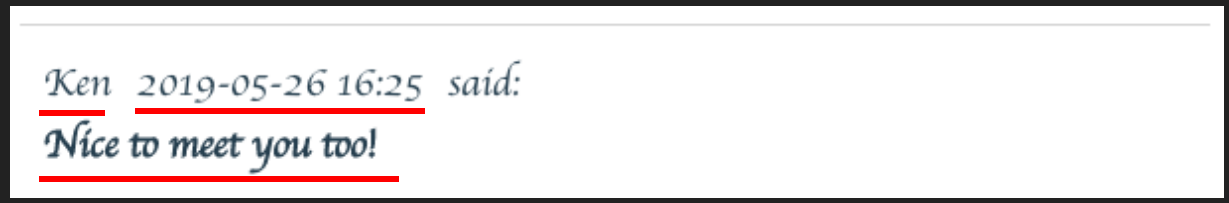
Oh...

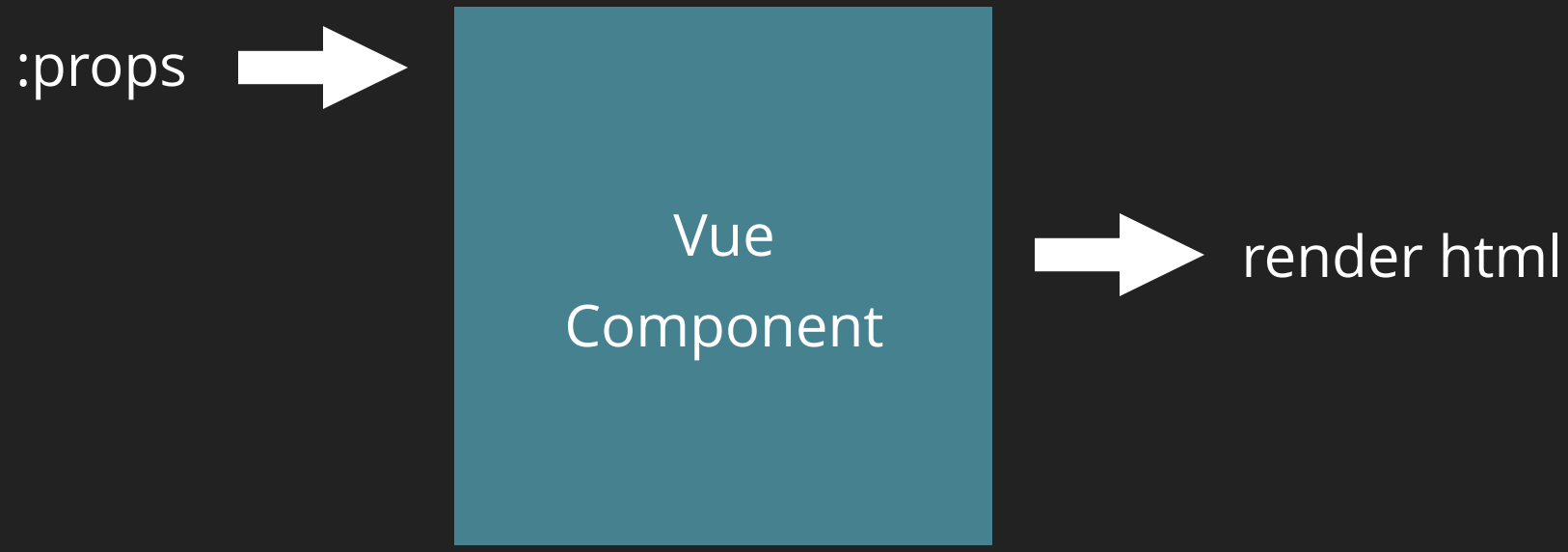
Jay 2019-05-21 09:27 said:

I'm joining vue conference today !!! ya!


▼ <MessageView> router-view: /message
<MessageInput>
▼ <MessageList>
<MessageRow key=0>
<MessageRow key=1>

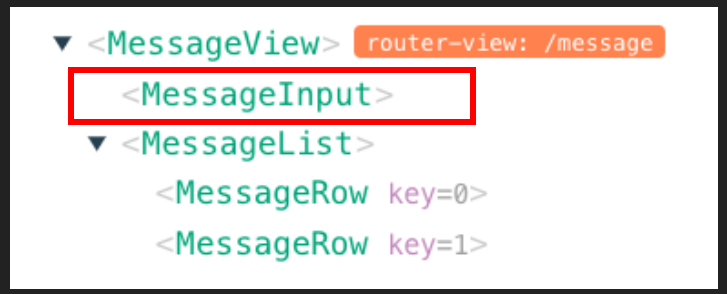
例子1：MessageRow

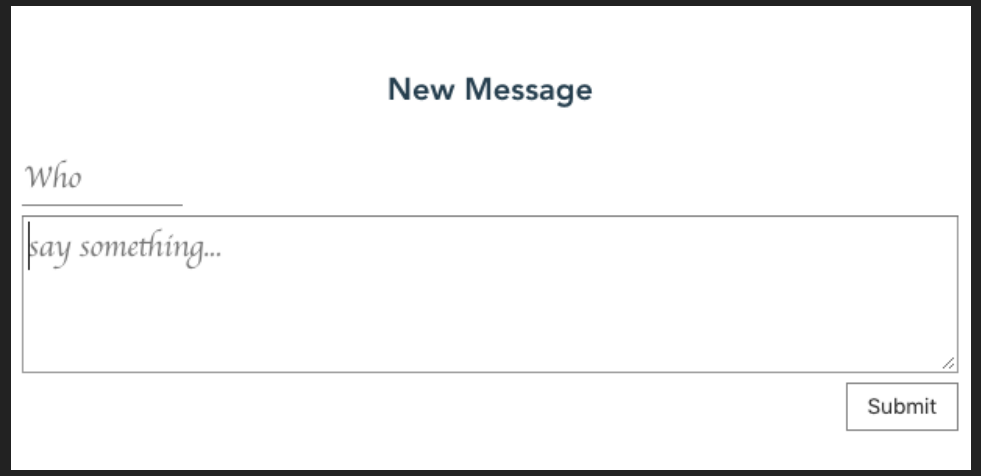



:props
Vue
Component
render html

```
1  import { shallowMount } from '@vue/test-utils'
2  import MessageRow from '@/components/message/list/row/MessageRow'
3
4  describe('MessageRow.vue', () => {
5
6    it('should render given props', () => {
7
8      const wrapper = shallowMount(MessageRow, {
9        propsData: {
10         sender: 'Jay',
11         message: 'Hello',
12         time: new Date(2019, 4, 7, 9, 0)
13       }
14     })
15
16     expect(wrapper.text()).toContain('Jay')
17     expect(wrapper.text()).toContain('Hello')
18     expect(wrapper.text()).toContain('2019-05-07 09:00')
19   })
20
```

例子2：MessageInput

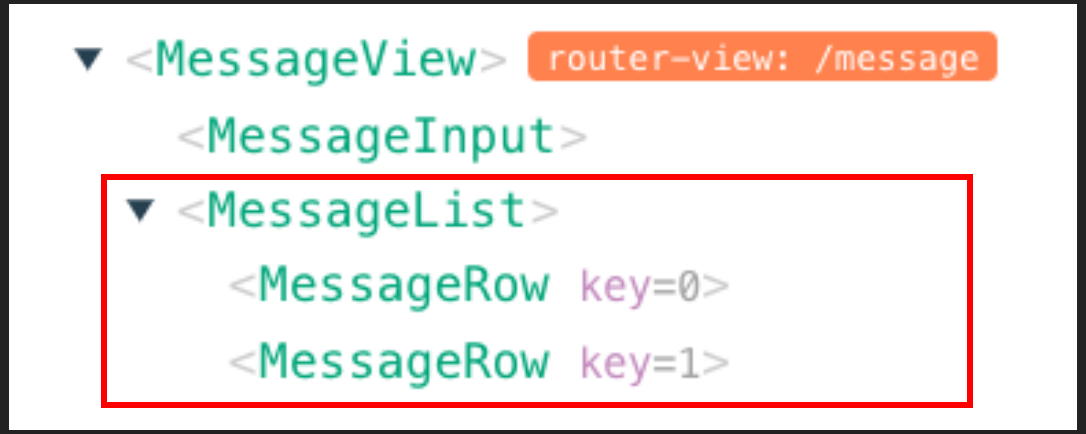


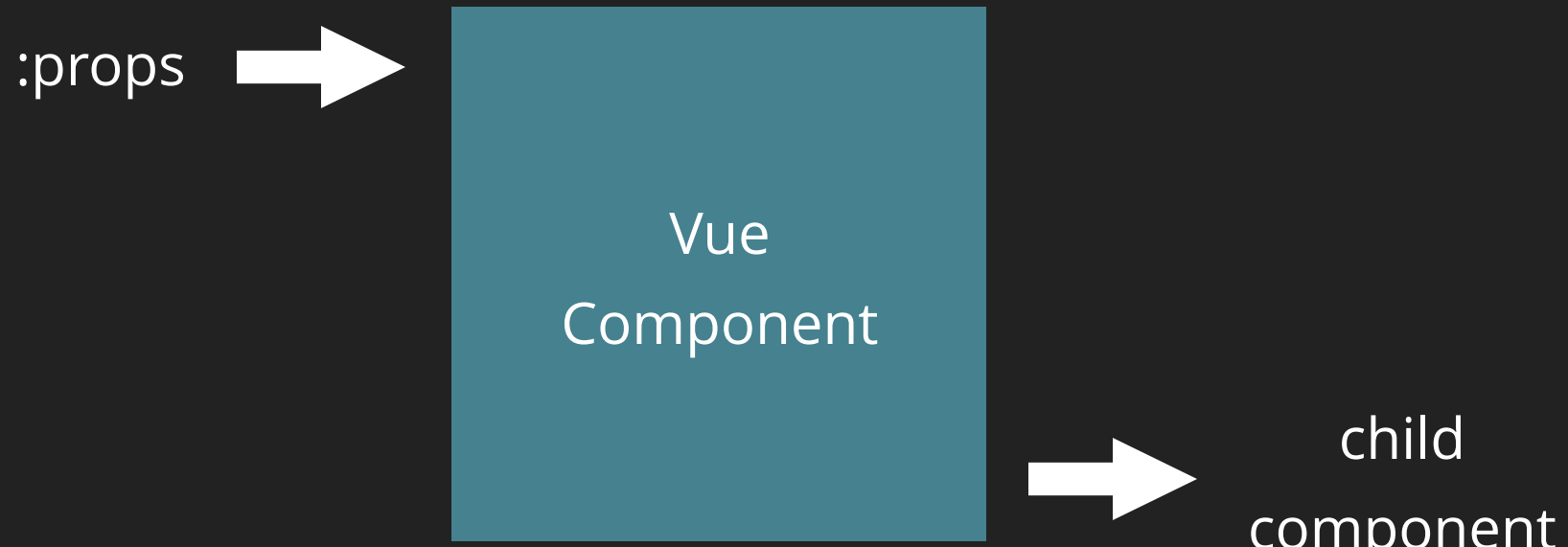



user
interaction
Vue
Component
@events

```
1   it('should emit a "submit" event when user click submit btn', () =>
2
3     let wrapper = shallowMount(MessageInputNew)
4
5     wrapper.find('input').setValue('Jay')
6     wrapper.find('textarea').setValue('Hello')
7     wrapper.find('button').trigger('click')
8
9     let emitted = wrapper.emitted('submit')
10    expect(emitted.length).toBe(1)
11    expect(emitted[0]).toEqual([
12      {
13        sender: 'Jay',
14        message: 'Hello'
15      }
16    ])
17  })
```

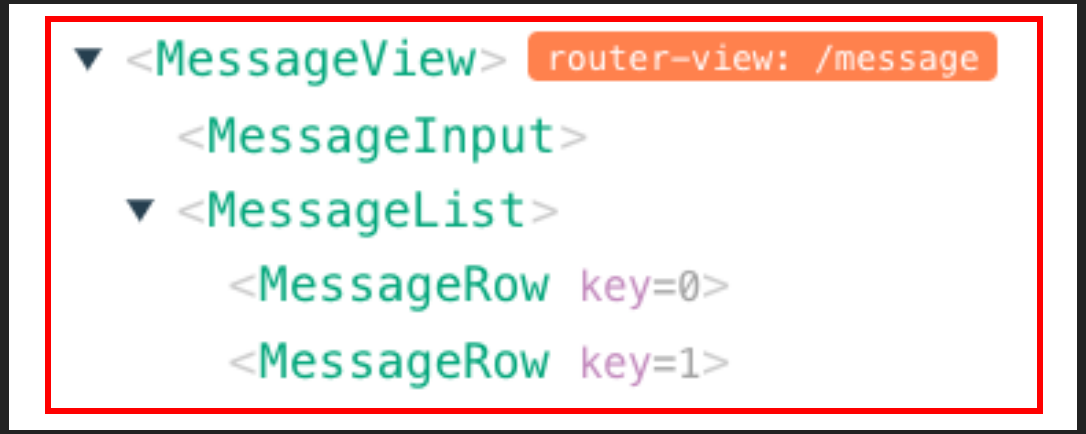
例子3：MessageList

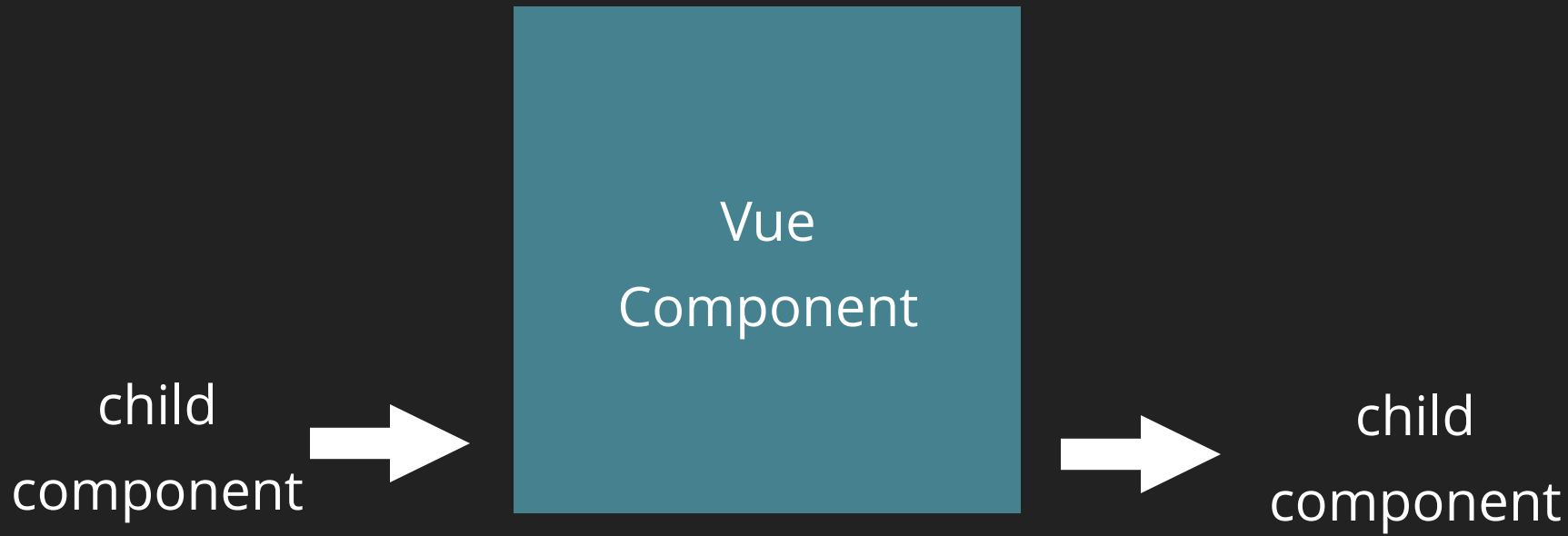



:props
Vue
Component
child
component

```
1   it('should render rows in time desc order', () => {
2
3     const wrapper = shallowMount(MessageList,{
4         propsData: {
5           data: [
6             { time: dateEarlier, sender: 'Ken', message: 'Hello' },
7             { time: dateLater, sender: 'Jay', message: 'Hi' }
8           ]
9         }})
10
11    const allMessageRows = wrapper.findAll(MessageRow)
12    expect(allMessageRows.length).toEqual(2)
13
14    expect(allMessageRows.at(0).props('time')).toEqual(dateLater)
15    expect(allMessageRows.at(0).props('sender')).toEqual('Jay')
16    expect(allMessageRows.at(0).props('message')).toEqual('Hi')
17
18    expect(allMessageRows.at(1).props('time')).toEqual(dateEarlier)
19    ...
20  })
```

例子4: MessageView




child
component
Vue
Component
child
component

```
<template>
  <div class="">
    <message-input @submit="onSubmit"></message-input>
    <message-list :data="data"></message-list>
  </div>
</template>

<script>
import postMessage, getAllMessages, ...
export default {
  name: 'MessageView',
  methods: {
    async onSubmit (payload) {
      await postMessage(payload)
      await this.reload()
    },
    async reload () {
      let { data } = await getAllMessages()
      this.data = data.map(msg => convertToMessageObject(msg))
    }
  }
}
</script>
```

```
1  import { getAllMessages, postMessage }
2      from '@/services/apis/message/message-api'
3
4  jest.mock('@/services/apis/message/message-api', () => ({
5    postMessage: jest.fn(),
6    getAllMessages: jest.fn()
7  }))
8
9  describe('MessageView.vue', () => {
10
11   afterEach(() => {
12     jest.clearAllMocks()
13   })
14
15   it('...', () => {})
16 })
```

```
1   import flushPromises from 'flush-promises'
2
3   it('@submit - api call flow', async () => {
4
5     getAllMessages.mockResolvedValue({ data: [msgA, msgB] })
6     postMessage.mockResolvedValueOnce({})
7
8     let wrapper = shallowMount(MessageView)
9     wrapper.find(MessageInput).vm.$emit('submit', requestPayload)
10
11    await flushPromises()
12
13    expect(postMessage).toHaveBeenCalledWith(requestPayload)
14    expect(getAllMessages).toHaveBeenCalledTimes(1)
15    expect(wrapper.find(MessageList).props('data'))
16        .toMatchObject([msgA, msgB])
17  })
```

不该测试什么？

- Vue 框架相关的逻辑
- 非接口层面的实现细节

测试写完就满足了?

使你的测试代码易于维护

-检视你的代码

tips 1:
使用factory方法标准化Component的创建

```
function createWrapper (overrides) {

  const propsData = {...}
  const defaultOptions = {propsData,localVue,store,mocks}

  return shallowMount(
    MessageRow,
    mergeWith(defaultOptions, overrides))
}

it(':message - should render message', () => {
  let wrapper = createWrapper({
    propsData: {
      message: 'My message'
    }
  })
  expect(wrapper.text()).toContain('My message')
})
```

tips 2:
创建自定义的Matcher
以简化verify

```
1  it('should render data list in time desc order', () => {
2    ...
3    const allMessageRows = wrapper.findAll(MessageRow)
4    expect(allMessageRows.length).toEqual(2)
5
6    expect(allMessageRows.at(0).props('sender')).toEqual('Jay')
7    expect(allMessageRows.at(0).props('time')).toEqual(dateLater)
8    expect(allMessageRows.at(0).props('message')).toEqual('Hi')
9
10   expect(allMessageRows.at(1).props('sender')).toEqual('Ken')
11   expect(allMessageRows.at(1).props('time')).toEqual(dateEarlier)
12   expect(allMessageRows.at(1).props('message')).toEqual('Hello')
13 })
```

```
let aMessageRowWith = (time, sender, message) => (row) => {
  expect(row.props('time')).toEqual(time)
  expect(row.props('sender')).toEqual(sender)
  expect(row.props('message')).toEqual(message)
}

it('should render data list in time desc order', () => {
  ...
  const allMessageRows = wrapper.findAll(MessageRow)
  expect(allMessageRows.length).toEqual(2)

  expect(allMessageRows.at(0))
      .toSatisfy(aMessageRowWith(dateLater, 'Jay', 'Hi'))
  expect(allMessageRows.at(1))
      .toSatisfy(aMessageRowWith(dateEarlier, 'Ken', 'Hello'))
})
```

tips 3:
留意每个单元测试的副作用
并及时作出清理

```
describe('MessageView.vue', () => {

  let mockFn = jest.fn()

  function createWrapper (overrides) {
    const defaultOptions = {
      ...
      localVue,
      store,
      mocks
    }
    return shallowMount(
      MessageRow,
      mergeWith(defaultOptions, overrides))
  }

  afterEach(() => {
    jest.clearAllMocks()
  })
  ...
})
```

tips 4:
以文档的方式
来组织单元测试用例

```
describe('HighlightList.vue', () => {

  describe(':props', () => {

    it(':highlight - should can highlight texts', () => {})

    it(':data - should render row in time desc order', () => {})

  })

  describe('@events', () => {

    it('@select - should submit on user click', () => {})

    it('@hover - should emit on hover', () => {})
  })
})
```

tips 5:
每个测试只负责一个特性

```
it('应该显示格式化的日期', () => {

})
```

```
it('应该显示格式化的日期，左边带一个时钟图标', () =>

})
```

```
it('应该显示格式化的日期，左边带一个时钟图标，
    如果日期大于某个特定值，则左边图标消失，
    改为显示“过期”', () => {

})
```

测试的 {FIRST} 原则

- Fast
- Isolated
- Repeatable
- Self-Validating
- Timely

测试怎样
改善生产代码?

Test love code
with clear focus

纯函数

- 减少中间状态的存储
- 把业务逻辑和Vue组件分离
- 把工具函数和Vue组件分离

设计你的组件

- 组件只关注一个重点

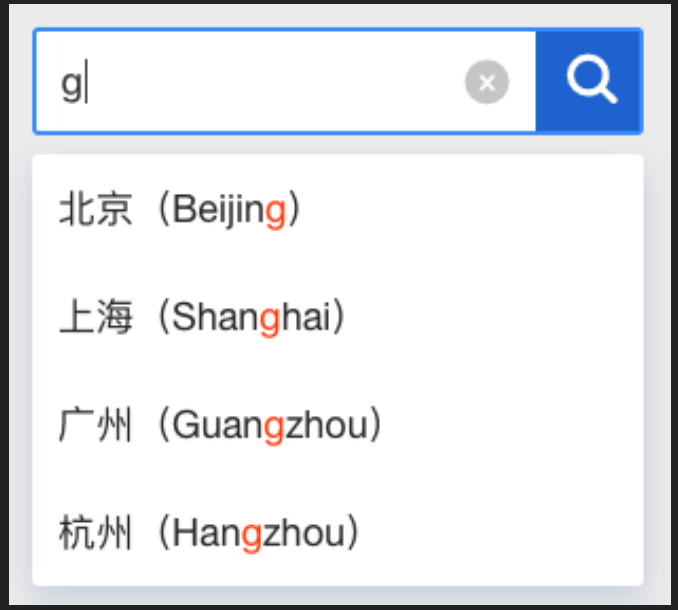
基础组件 - 简单一致的输入和输出

- 接收 :props 并渲染 html
- 获取用户输入并提供@events

粘合组件 - Controller组件

- 引入service或api依赖
- 协调基础性组件间的交互

练习：设计一个可重用的服务端交互的自动补全组件

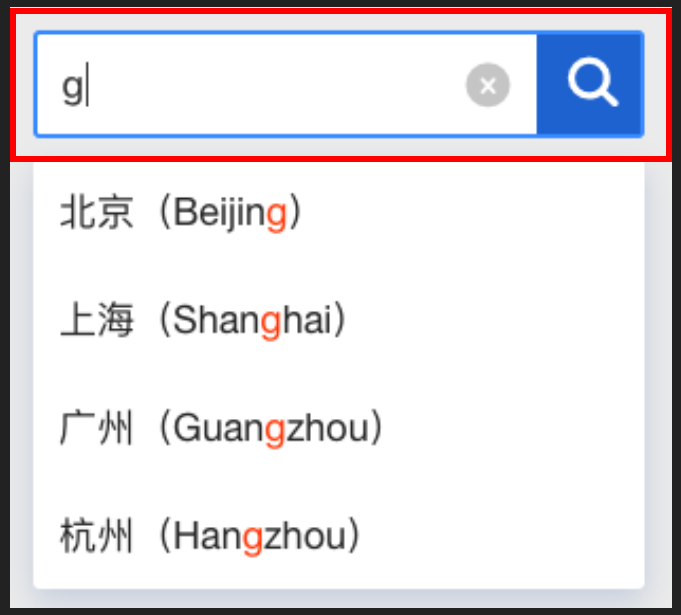


← <h-Input />

← <h-highlight-list />

<h-remote-auto-complete />

```
<h-input  :icon=""
          @input=""/>
```



```
<h-highlight-list :data="[..]"
                  :highlight-keyword=""
                  @select=""/>
```

```
<h-remote-auto-complete :api=""
                        :converter=""
                        @select=""/>
```

:api(input)

:converter(response)

<h-input>
@input

<h-highlight-list>
:data
:highlight-keyword
@select

<location-picker @select="onSelect"/>

```
<h-remote-auto-complete
    :api="xx"
    :converter="xx"
    @select="$emit('select', ...)"
```

<product-picker @select="onSelect"/>

<client-picker @select="onSelect"/>

<department-picker @select="onSelect"/>

小结

- 单元测试的问题
- 4个不同种类的Vue组件测试例子
- 5个Vue组件测试代码Tips
- 单元测试的一些基础原则
- 改善生产代码

参考资料

- <Testing Vue.js Application> by Edd Yerburgh
- Component Tests with Vue.js by Matt O'Connell on Vue NYC
  https://www.youtube.com/watch?v=OIpfWTThrK8

  https://github.com/vuejs/vue-cli/tree/dev/packages/%40vue/cli-plugin-unit-jest
  https://github.com/vuejs/vue-jest
  https://github.com/facebook/jest

我的技术，随笔公众号

示例代码地址：

https://github.com/jaylu/vueconf2019-unit-test