

Vue.js 实践

如何使用Vue2.0开发富交互式WEB应用



| 我是谁

钟恒

360奇舞团前端工程师

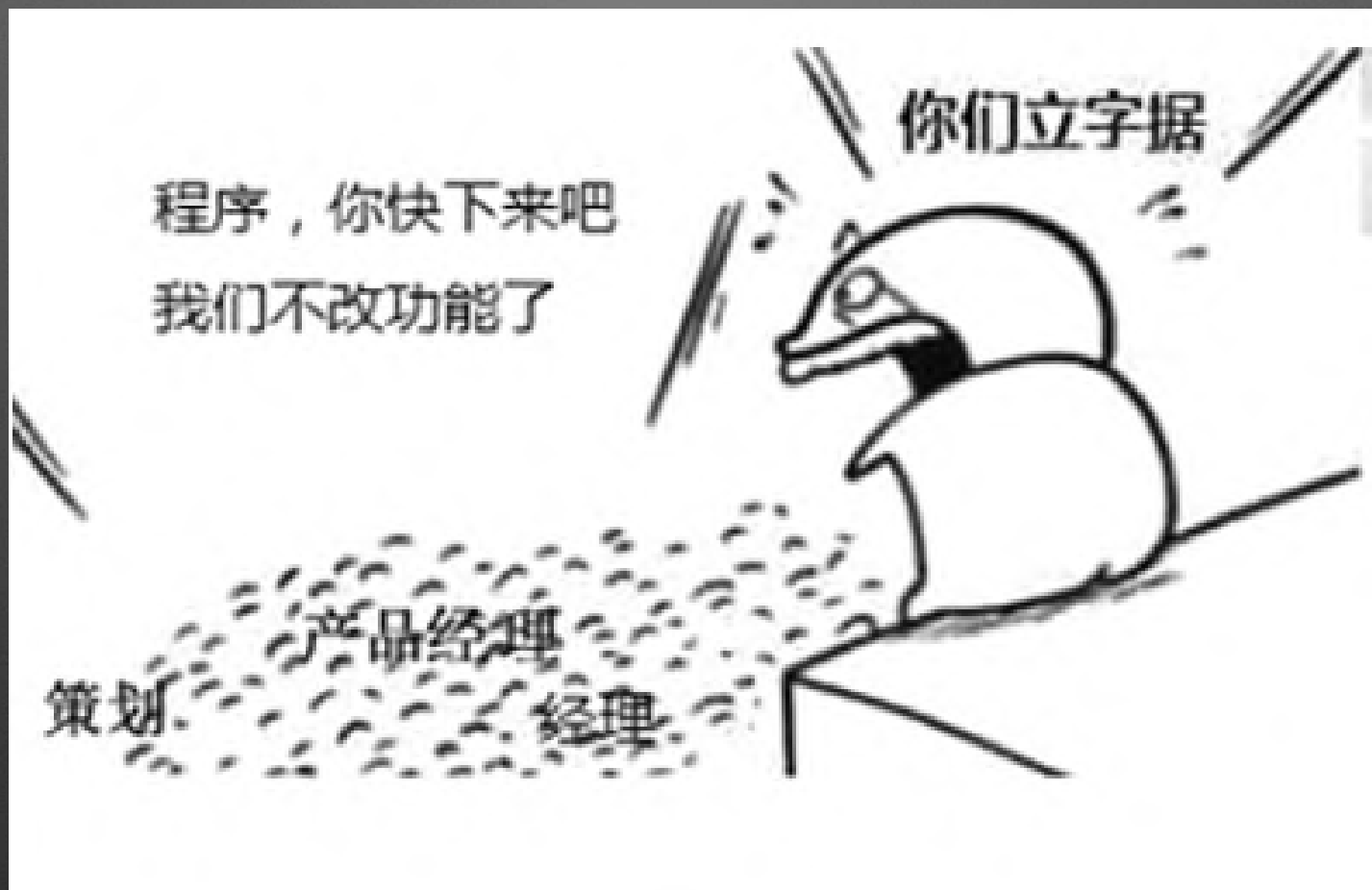


| 提纲

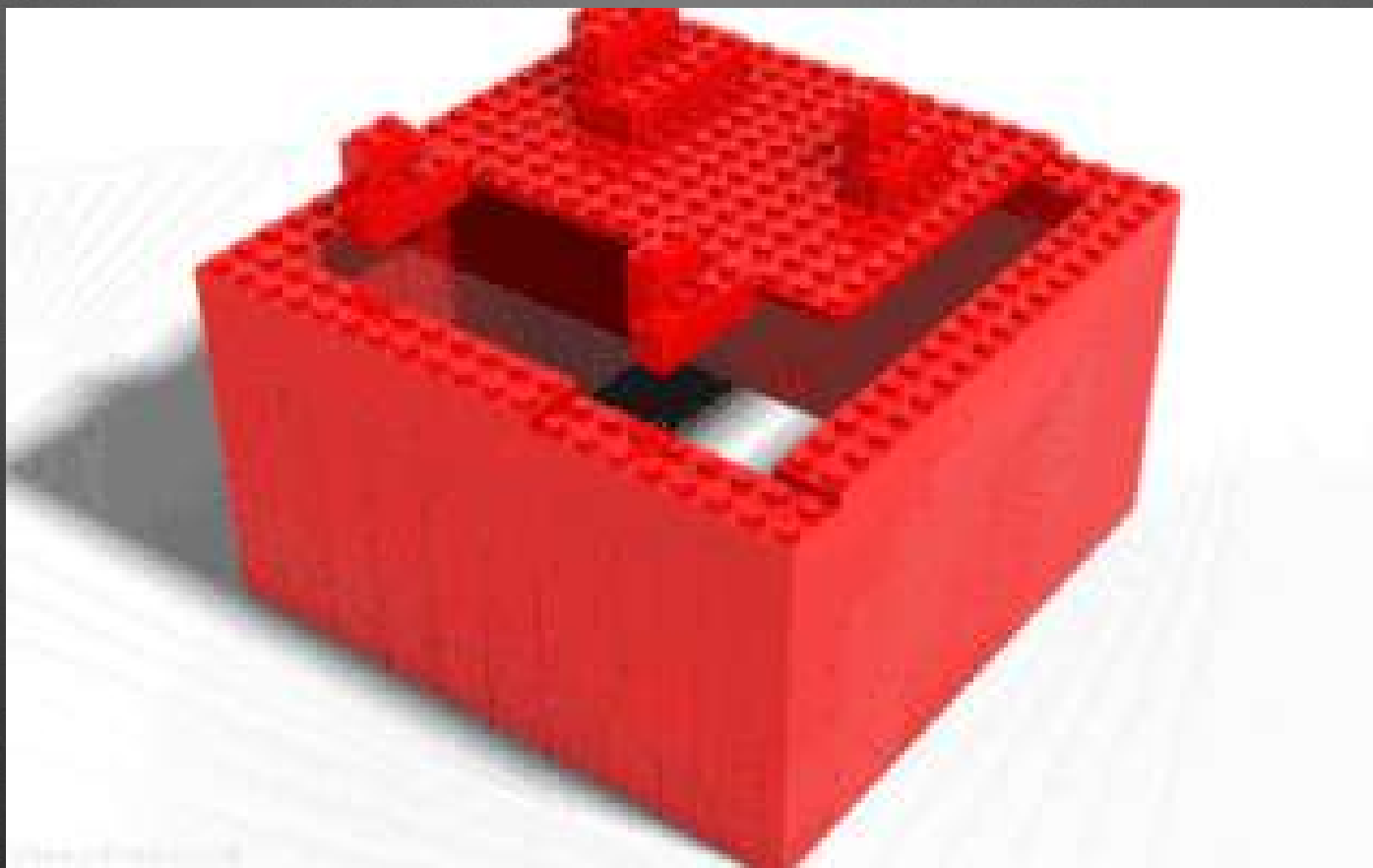
1. 架构
2. 开发
3. 填坑优化



| 架构难点



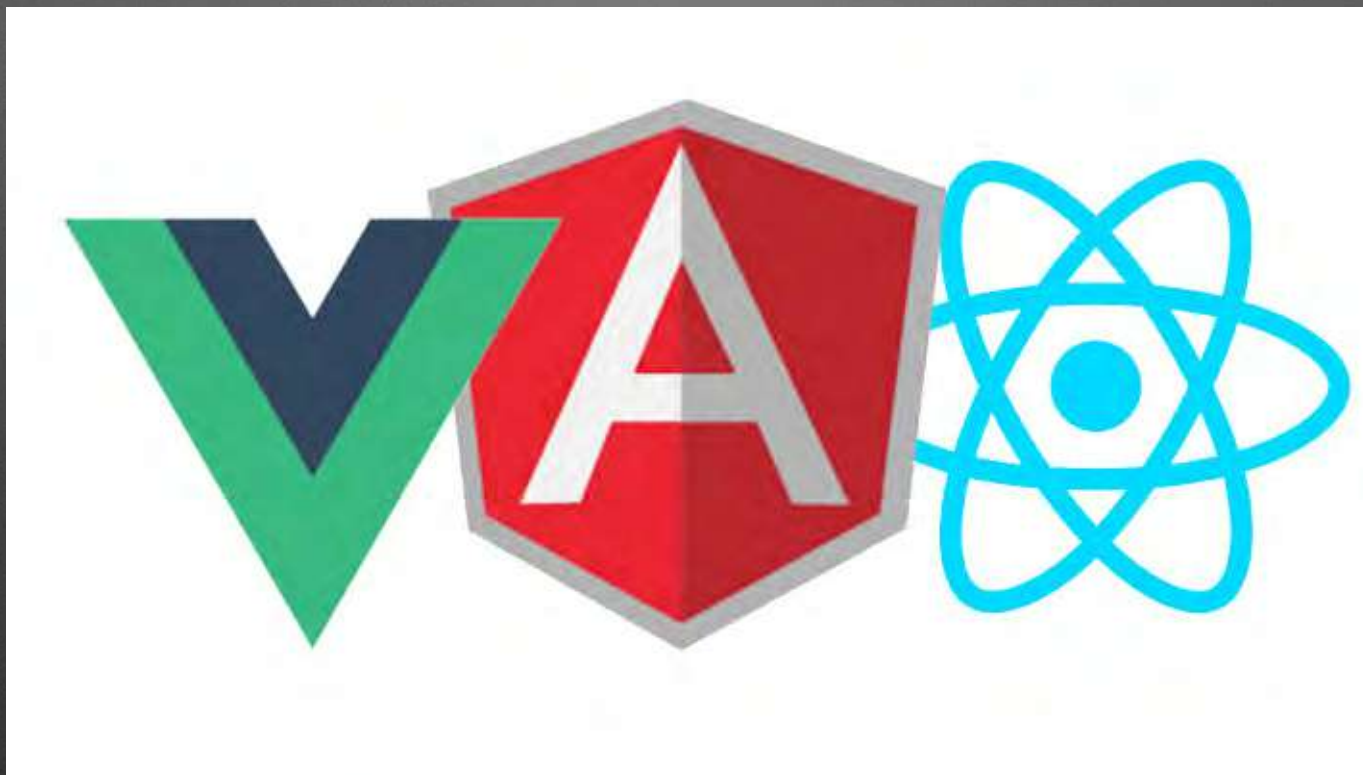
| 怎么办



前端界现在怎么组件化呢？



| 框架时代



框架对我们的开发方式有什么改变？

| 数据绑定



```
new Vue({  
  template: '<input type="number" v-model="num">',  
  data: {  
    num: 0  
  }  
})
```



双向数据流较单向数据流难以维护



| UI结构划分





组件数目过多



单组件过重

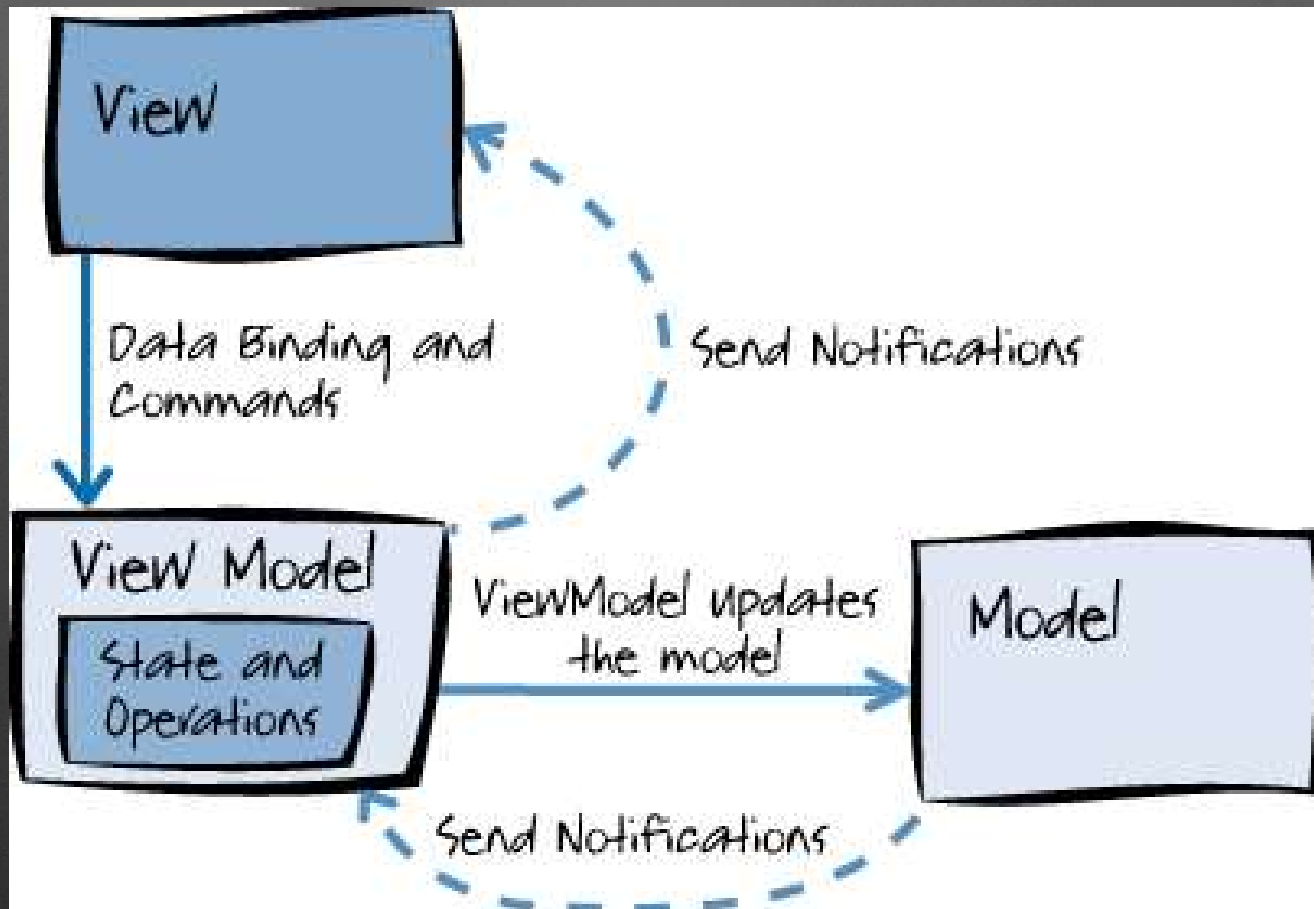


不便于更改

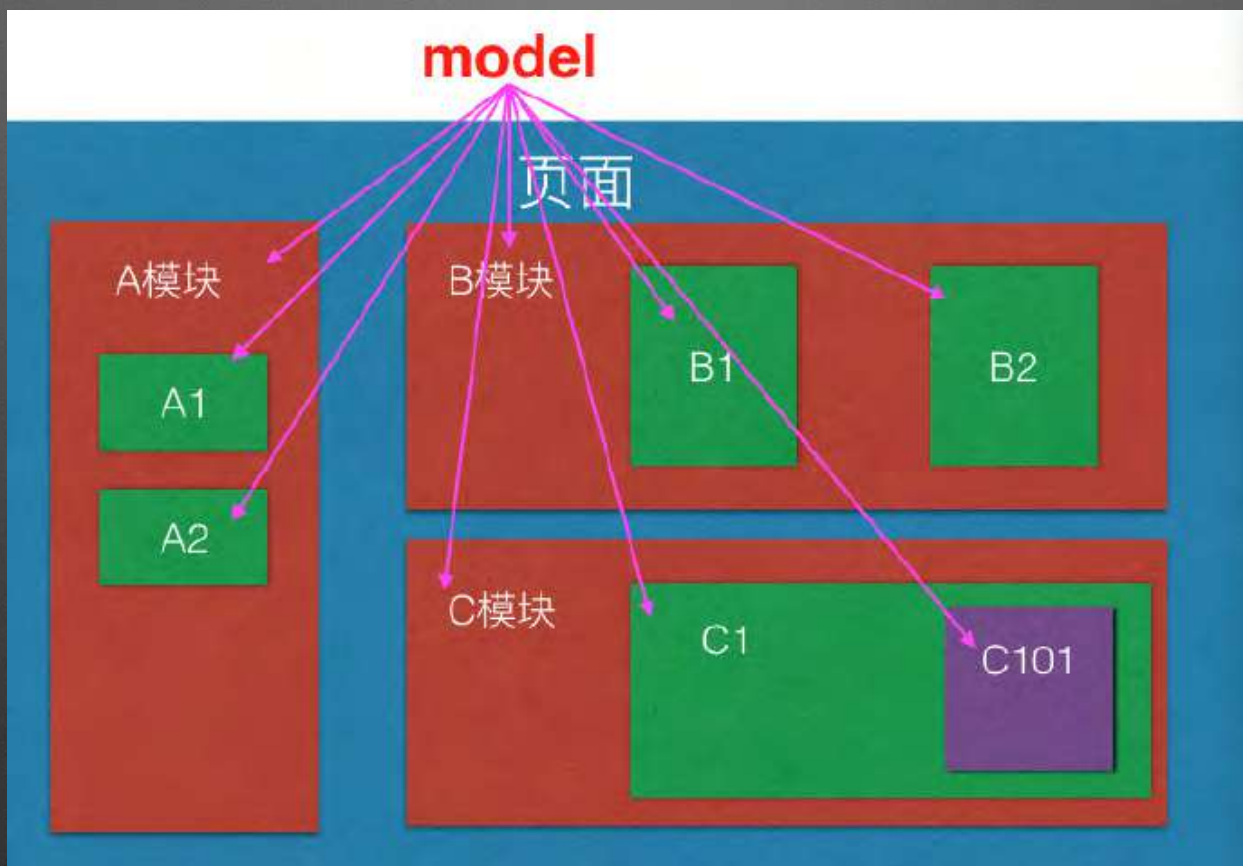
~~复用性高~~
~~易于维护~~
~~易于变更~~
团队合作

复杂的软件必须有清晰合理的架构，
否则无法开发和维护。

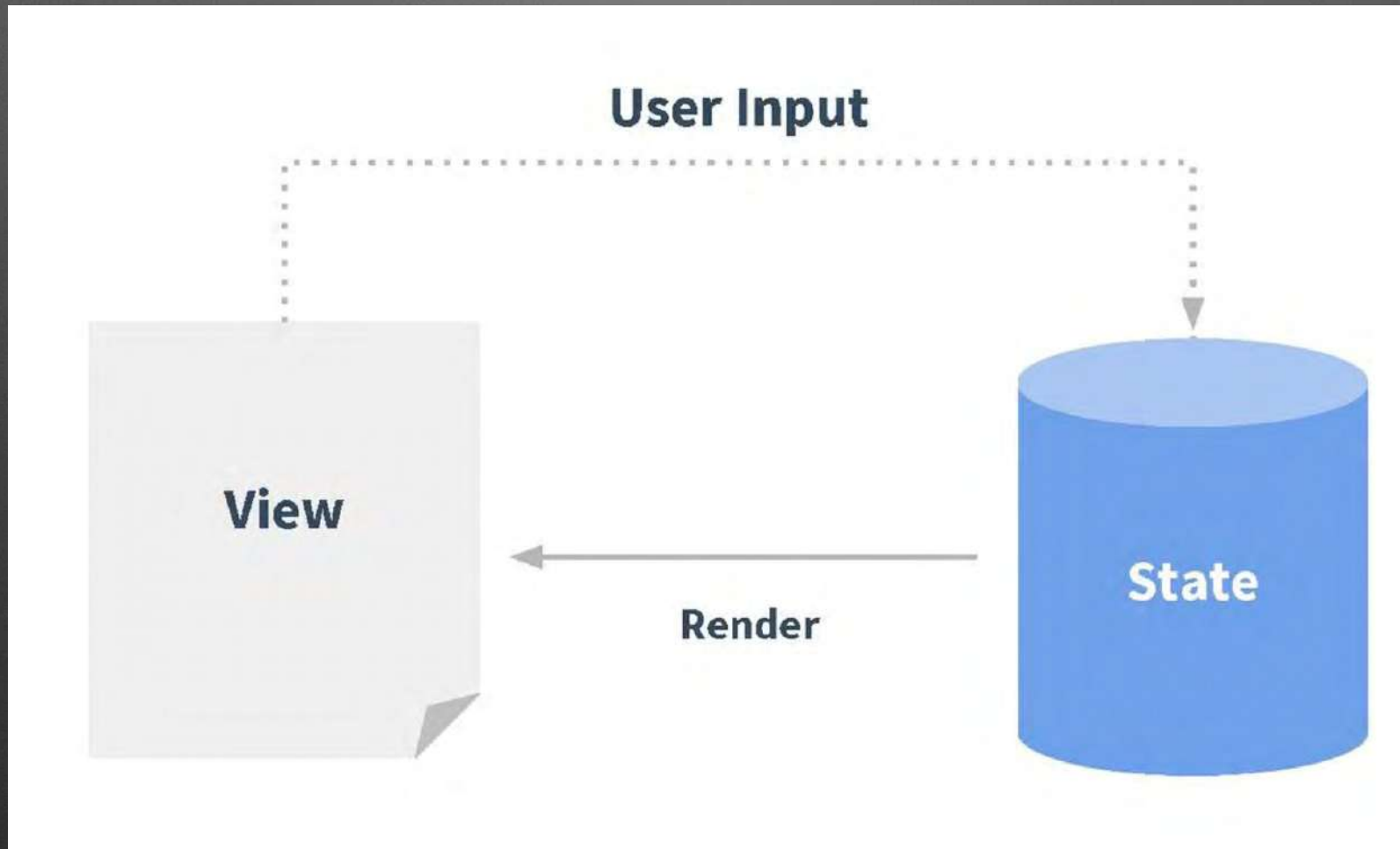
| 什么是MVVM



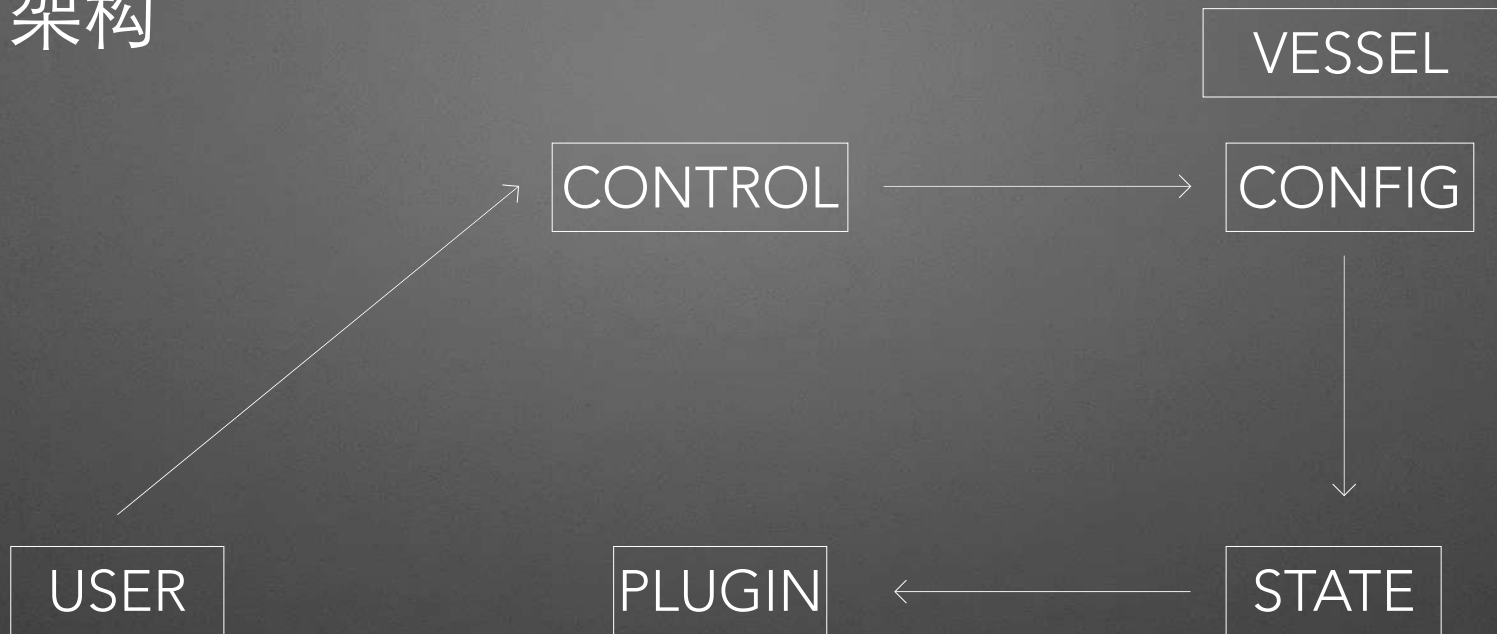
| 数据驱动



| 读写分离



| 架构



业务逻辑



交互逻辑



页面布局

展示逻辑



| 好处

保留了双向绑定

易于编写交互

数据驱动

易于开发

主业务读写分离

易于维护

业务、交互、
展示、布局分离

易于更迭

| 提纲

1. 架构 ✓

一个MVVM式的组件化架构

2. 开发

组件多

组件重

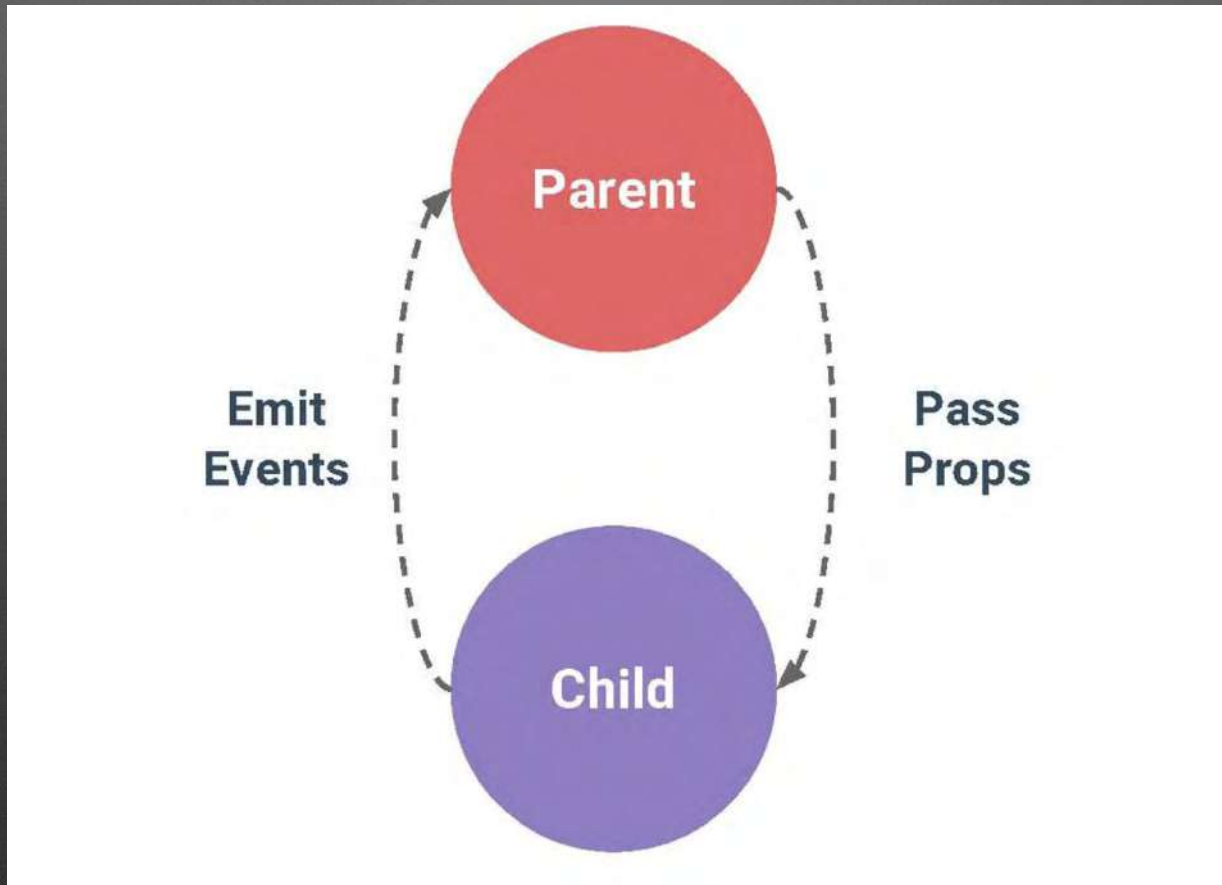
3. 填坑优化



多组件间如何通信?



| Vue.js提供的通信方式



| props

```
Vue.component('child', {  
  props: ['message'],  
  template: '<span>{{ myMessage }}</span>'  
})
```

```
<div>  
  <input v-model="parentMsg">  
  <br>  
  <child :message="parentMsg"></child>  
</div>
```



HTML属性式传数据太方便了!

props能双向传数据吗?

Vue.js 1.0

```
<!-- 双向 prop 绑定 -->  
<my-component :prop.sync="someThing"></my-component>
```

Vue.js 2.0

```
props: {  
  callback: {  
    type: Function,  
    default: () => {}  
  }  
}
```

```
<my-component :callback="setStyles"></my-component>
```

每次都要设置props太麻烦了.....

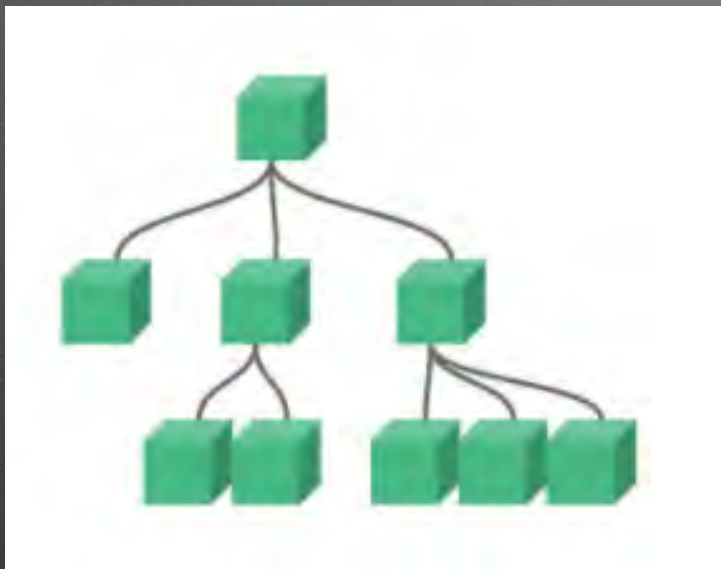


| 函数调用

```
var app = new Vue({
  el: '#app',
  template: `
<div>
  <child ref="child"></child>
  <button v-on:click="reverse">
    Reverse Message
  </button>
</div>`,
  methods: {
    reverse () {
      this.$refs.child.reverseMessage()
    }
  }
})
```



| 组件树的问题



```
this.$parent.$parent.open()
```

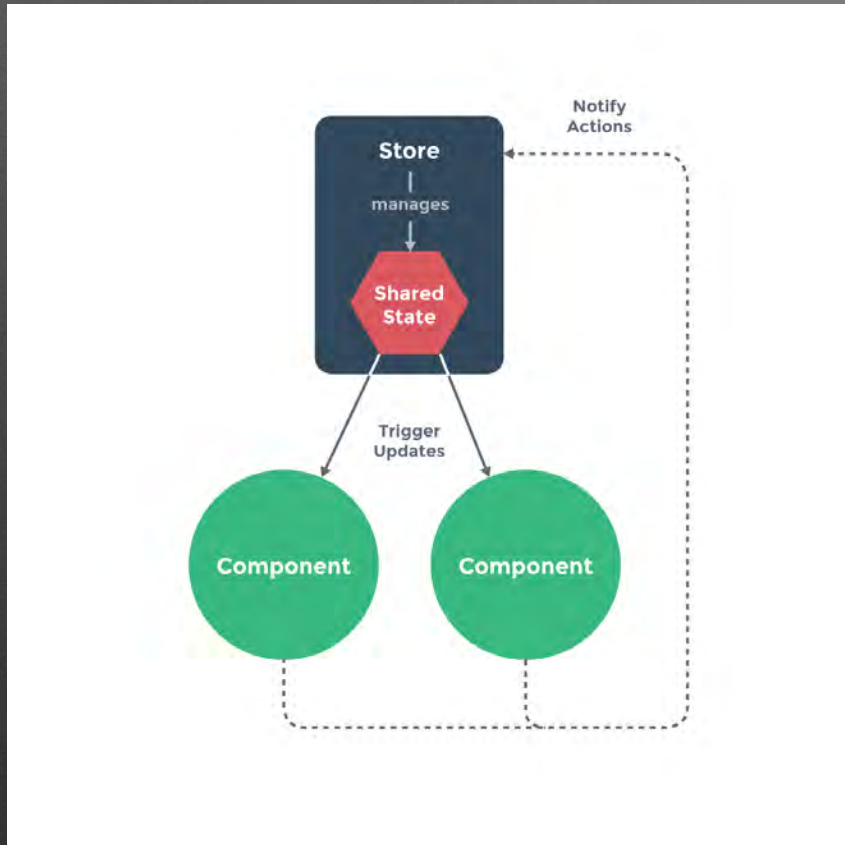
鬼知道我经历了
什么



需要更清晰扁平的通信方式



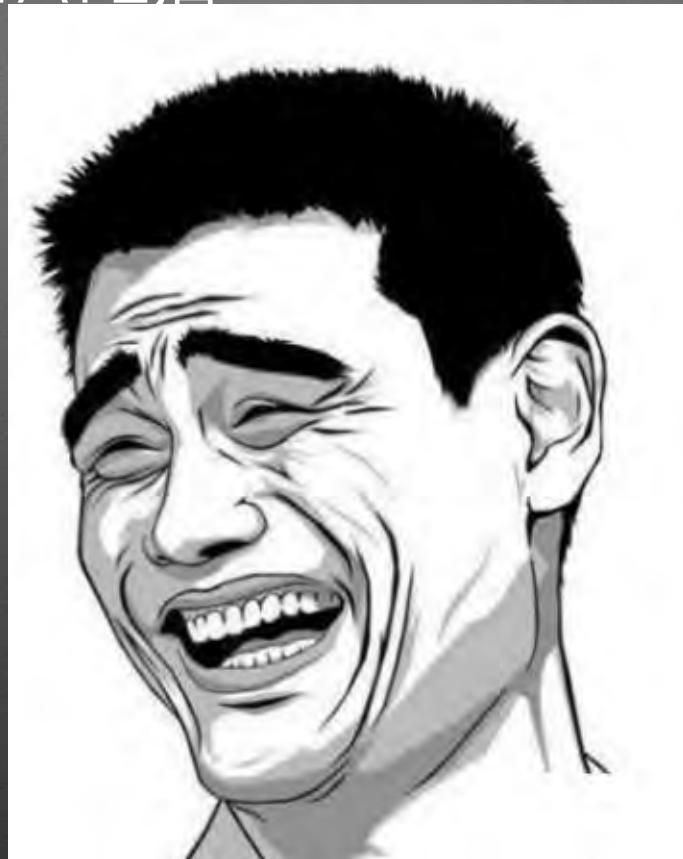
| 利用共享STATE



```
// plugin  
<span>{{num}}</span>  
  
// plus  
this.num++  
  
//minus  
this.num--
```



| 用了共享STATE后



我改的，不好意思.....你加个锁吧



| 这个功能是异步的

我先监听标识位A

我再修改标志位A，通知alpha组件去执行异步功能。

接着修改标志位B，证明标识位A是我修改的。

哦，标识位A被修改了

看看标识位B，这是alpha组件的标志

拿到数据咯

| 当数据位被多方操作



| Eventbus

```
var bus = new Vue()

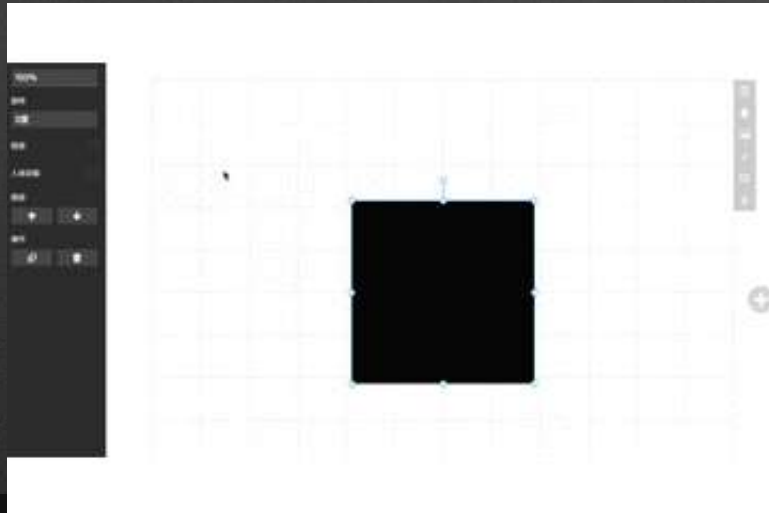
// in component A's method
bus.$emit('plus', 1)
bus.$emit('minus', 1)

// in component B's created hook
bus.$on('plus', function (n) {
  this.total += n
})

bus.$on('minus', function (n) {
  this.total -= n
})
```



| 利用eventbus解决异步问题



```
// 组件内部
bus.$on('open-resource-shape', callback => {
  this.choose().then(callback)
})

// 使用者
bus.$emit('open-resource-shape', shape => {
  this.addShape(shape)
})

// 当然也可以promise啦
getShape () {
  return new Promise((resolve, reject => {
    bus.$emit('open-resource-shape', resolve)
  })))
}

this.getShape()
  .then(this.addShape)

// 自然也可以async/await
let shape = await this.getShape()
this.addShape(shape)
```

| 通信方法选择

方法	场景
props	强耦合的组件间，单纯信息传递
function	强耦合的组件间多种通信方式
state	同步行为、数据量较小、数据位不被共用
eventbus	异步行为、数据量较大、共用的组件

| 提纲

1. 架构 ✓

一个MVVM式的组件化架构 ✓

2. 开发

组件多 ✓

组件重

3. 填坑优化

为什么组件会越来越重？



```
// wrong
<control-input type="number"></control-input>
// right
<control-number></control-number>
```



尽可能保证组件功能性单一

```
if(this.type === 'editing') {  
    // some editing code  
} else if(this.type === 'preview') {  
    // some preview code  
} else if(this.type === 'present') {  
    // some present code  
} else {  
    // some base code  
}
```



| 减法

```
// plugin-page.vue
<div>
  <slot name="page">
    i am a page
  </slot>
</div>
```

```
// present-plugin-page.vue
<div class="PresetPluginPage">
  <plugin-page ref="page">
    <h1 slot="page">
      i am a present page
    </h1>
  </plugin-page>
</div>
```

```
//output
<div class="PresetPluginPage">
  <div>
    <h1>
      i am a present page
    </h1>
  </div>
</div>
```

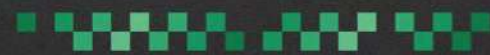
抽取公共组件

| 加法

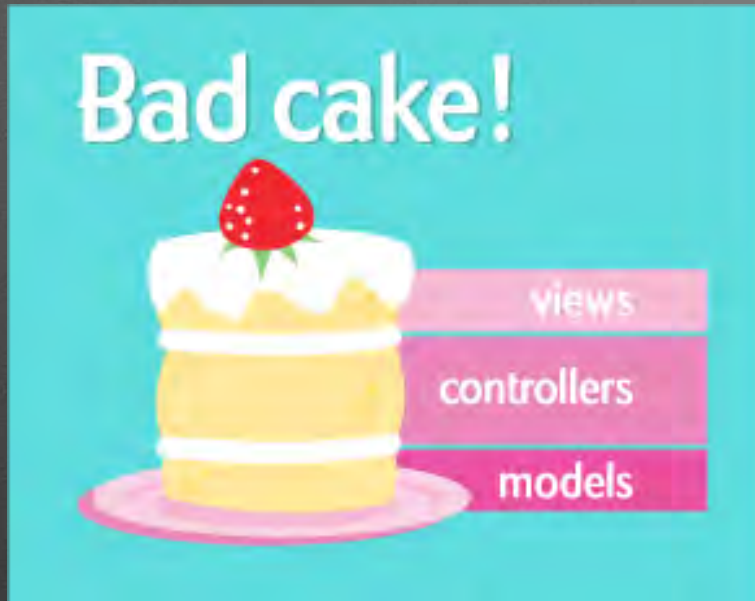
```
// define a mixin object
var myMixin = {
  created: function () {
    this.hello()
  },
  methods: {
    hello: function () {
      console.log('hello from mixin!')
    }
  }
}

// define a component that uses this mixin
var Component = Vue.extend({
  mixins: [myMixin]
})
```

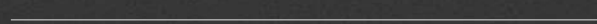
减少组件冗余性



I MVC的老问题



fat controller



fat viewModel



| 减肥

数据存储方式：数组/对象

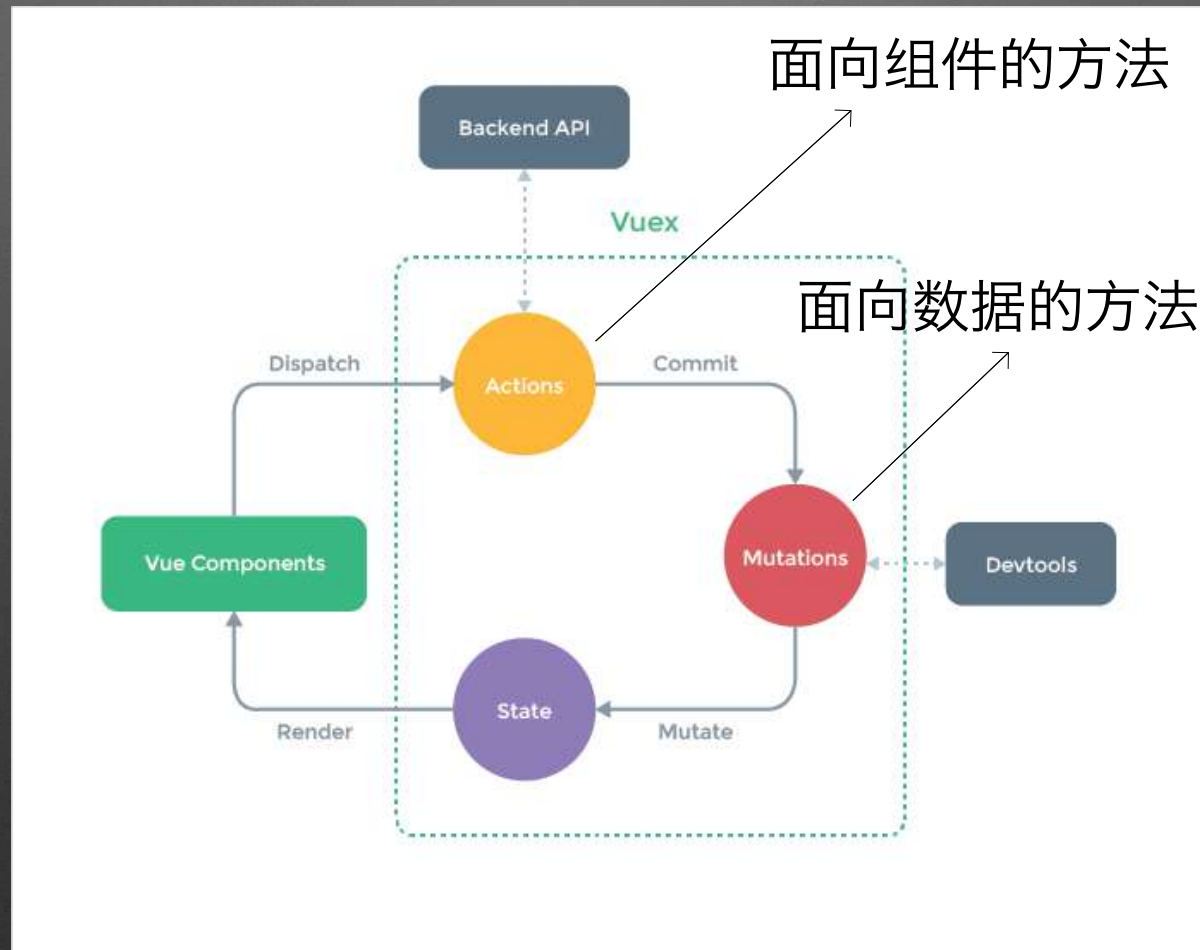
数据操作方法：增、删、改、查

抽取公用的数据处理部分

隔离变化频繁的controller



| 利用Vuex优化你的model



I bug



我们都怕bug

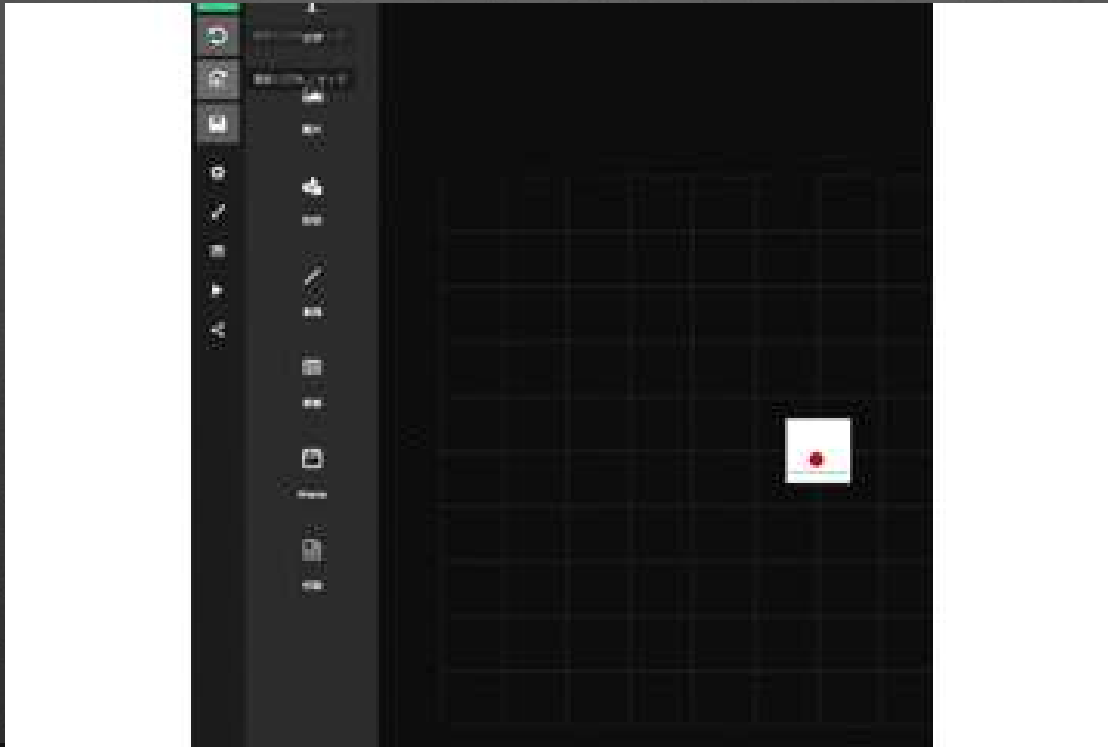


但我们更怕找不到bug



| 状态机

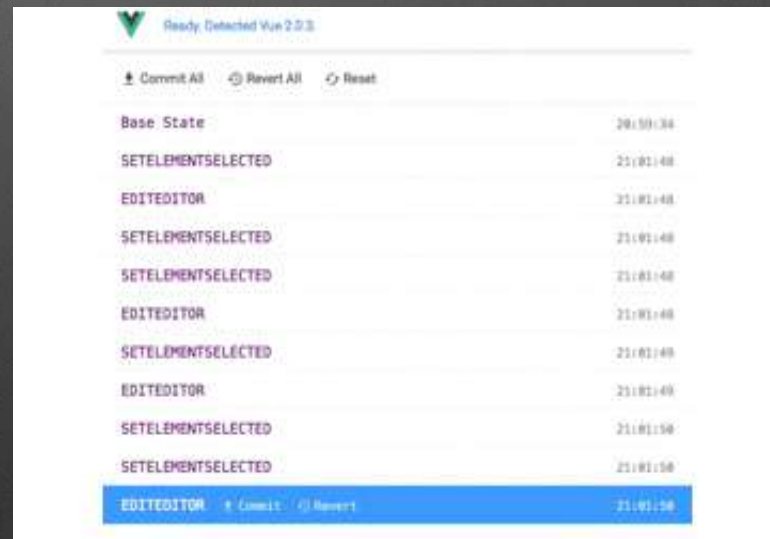
View = Vue(state)



| 打点

$change = diff(state_m, state_n)$

or mutation



Ready Detected Vue 2.2.3

Commit All Revert All Reset

Base State	20:59:34
SETELEMENTSSELECTED	21:01:40
EDITEDITOR	21:01:40
SETELEMENTSSELECTED	21:01:40
SETELEMENTSSELECTED	21:01:40
EDITEDITOR	21:01:40
SETELEMENTSSELECTED	21:01:40
EDITEDITOR	21:01:40
SETELEMENTSSELECTED	21:01:50
SETELEMENTSSELECTED	21:01:50
EDITEDITOR	21:01:50



| Vuex1.0 or Vuex2.0

	1.0	2.0
action	同步	异步
action参数	多个	单个
服务端渲染	不支持使用Vuex属性 添加actions/getters	支持
action/mutation语义	不清晰	清晰

Returning Promises from Vuex actions

| 提纲

1. 架构 ✓

一个MVVM式的组件化架构 ✓

2. 开发

组件多 ✓

组件重 ✓

3. 填坑优化

前端框架的坑

更快更好?



| 为什么会有坑



开发者



代码



浏览器

蜘蛛

读屏器

根据需求选择最适合的开发方式



| 前端框架的坑

首屏体验

lazy-load
loading

Service Worker

SSR

SEO

pre-render

SSR

老式浏览器

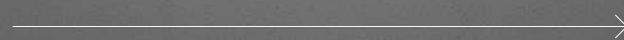
SSR



| 本质问题

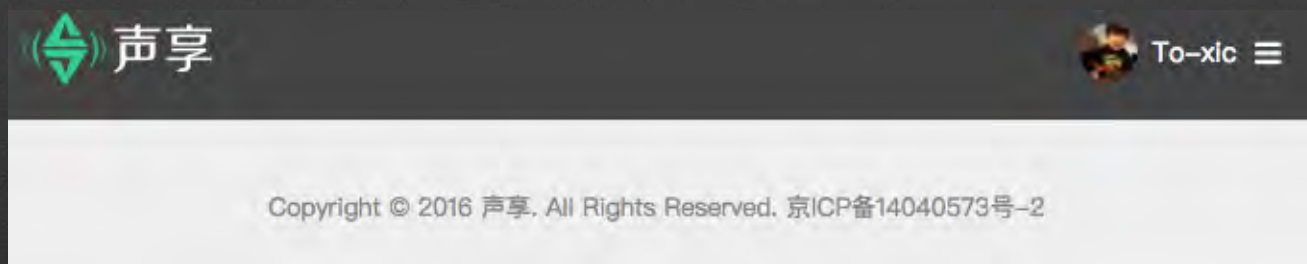
前端JavaScript渲染

模板

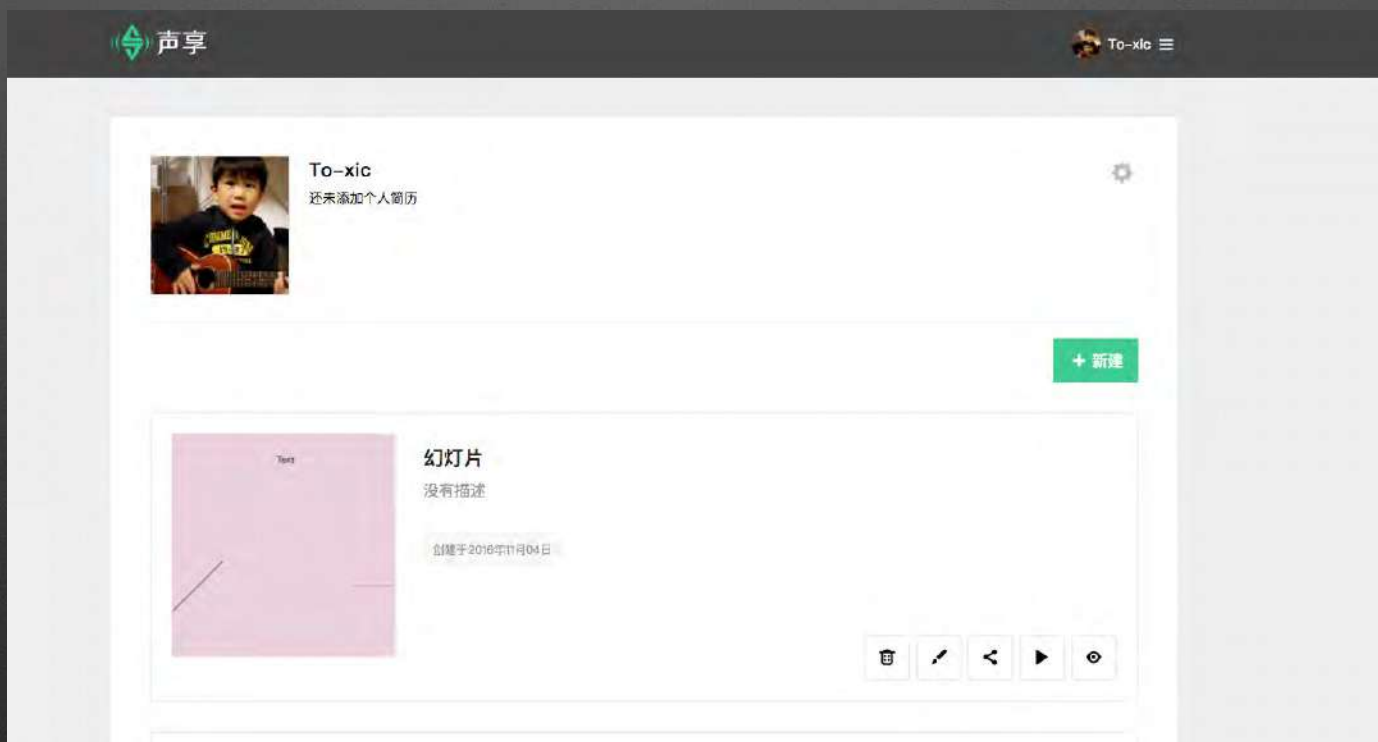


可互动的应用

```
> <header class="Header">...</header>  
  <mount-vessel></mount-vessel>  
> <footer class="Footer">...</footer>
```



| 什么是SSR



| Vue的SSR

```
// the default export should be a function
// which will receive the context of the render call
export default context => {
  return app.preFetch(context).then(() => {
    return app
  })
}
```

```
import path from 'path'
import fs from 'fs'
import * as ssr from 'vue-server-renderer'
import lru from 'lru-cache'

const filePath = path.join(__dirname, '/path/to/your/file')
const code = fs.readFileSync(filePath, 'utf8')
let renderer = ssr.createBundleRenderer(code, {
  cache: lru(1000)
})
renderer.renderToString(options, (err, html) => {
  this.assign({html})
})
```


| SSR的性能

it cost 992 to render

组件缓存

```
const LRU = require('lru-cache')

const renderer = createRenderer({
  cache: LRU({
    max: 10000
  })
})
```

```
export default {
  name: 'item', // required
  props: ['item'],
  serverCacheKey: props => props.item.id,
  render (h) {
    return h('div', this.item.id)
  }
}
```

自动化首次渲染

thinkjs的bootstrap执行



| SSR对组件的要求

- 前后端均可运行
- 区分静态/动态组件
- 数据彻底分离



| 提纲

1. 架构 ✓

一个MVVM式的组件化架构 ✓

2. 开发

组件多 ✓

组件重 ✓

3. 填坑优化

前端框架的坑 ✓

更快更好?



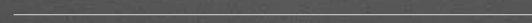
I WEB应用的痛



| 离线化

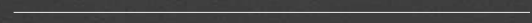
$$\text{Page} = \text{code}(\text{state})$$

state

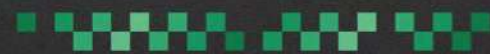


localStorage

code



Service Worker



| 离线化

```
if ('serviceWorker' in navigator) {
  navigator.serviceWorker.register('/sw.js')
  .then(registration => {
    // success
  }).catch(function (err) {
    // registration failed :(
    console.error(err)
  })
}
```

```
// 监听fetch
self.addEventListener('fetch', function (event)
  let request = event.request
  // 尝试返回线上的数据
  event.respondWith(
    fetch(request)
    .then(function (response) {
      // 若成功, 则缓存以备日后使用
      var copy = response.clone()
      caches.open(cacheKeys[1])
      .then(function (cache) {
        cache.put(request, copy)
      })
      return response
    })
    .catch(function () {
      // 若失败则反馈缓存中的部分
      return caches.match(request)
      .then(function (response) {
        return response
      })
    })
  )
})
```

离线化

The image shows a web editor interface on the left and a browser's developer console on the right. The editor has a sidebar with icons for text, images, shapes, lines, tables, iframes, and code. A red banner at the top of the editor reads "网络故障~~" (Network outage~~). The developer console shows a list of network requests and a detailed view of the console with several error messages.

Network Requests:

Name	Status	Type	Initiator	Size	Time	Timeline
ckeditor.js	200	script	global.is:1	(from Ser...	283 ms	

Console Errors:

- POST <https://ppt.baomitu.com/api/slide/update> net::ERR_INTERNET_DISCONNECTED
- Failed to load resource: https://ppt.baomitu.com/editor?slide_id=b4db119b net::ERR_INTERNET_DISCONNECTED
- Failed to load resource: <https://s.ssl.qhres.com/static/edb162a7eb9e61b8/static/css/global.css> net::ERR_INTERNET_DISCONNECTED
- Failed to load resource: https://s.ssl.qhres.com/static/66cbcb9ac7851c80/view/editor/index_index.css net::ERR_INTERNET_DISCONNECTED
- Failed to load resource: <https://s.ssl.qhres.com/static/8b945956143f545e/static/js/global.js> net::ERR_INTERNET_DISCONNECTED
- Failed to load resource: https://s.ssl.qhres.com/static/040bf9ccf41ebbcf/view/editor/index_index.js net::ERR_INTERNET_DISCONNECTED
- GET https://ppt.baomitu.com/editor?slide_id=b4db119b:101 net::ERR_INTERNET_DISCONNECTED
- Failed to load resource: <https://ppt.baomitu.com/static/js/ckeditor.js> net::ERR_INTERNET_DISCONNECTED
- TypeError: Failed to fetch <https://s.ssl.qhres.com/static/4459d3399487535b/static/font/fontello-766c6d0a979b6f2c0fef8696e43dece7.woff2>
- Uncaught ReferenceError: monitor is not defined(...) [editor?slide_id=b4db119b:102](https://ppt.baomitu.com/editor?slide_id=b4db119b:102)
- POST <https://ppt.baomitu.com/api/slide/update> net::ERR_INTERNET_DISCONNECTED
- GET <https://ppt.baomitu.com/api/template/publiclist?page=1&epage=20> net::ERR_INTERNET_DISCONNECTED
- GET <https://ppt.baomitu.com/api/template/list?page=1&epage=20> net::ERR_INTERNET_DISCONNECTED
- GET <https://widget.daovoice.io/widget/fbd35503.js> net::ERR_INTERNET_DISCONNECTED
- error

I WEB应用的痛















| 本地化

```
// 监听fetch
self.addEventListener('fetch', function (event) {
  let request = event.request
  // 尝试返回缓存数据
  event.respondWith(
    caches.match(request)
      .then(function (cacheResponse) {
        // 若无缓存则请求线上数据
        return cacheResponse || fetch(request)
      })
      .then(function (response) {
        // 若成功, 则缓存以备日后使用
        var copy = response.clone()
        caches.open(cacheKeys[1])
          .then(function (cache) {
            cache.put(request, copy)
          })
        return response
      })
  )
})
}
```



| 本地化

Name	Status	Type	Initiator	Size	Time	Timeline - Start
 editor?slide_id=567615ee	200	document	Other	{from ServiceWo...	56 ms	
 global.css	200	stylesheet	editor?slide_id=567615ee:4	{from ServiceWo...	3 ms	
 index_loader.css	200	stylesheet	editor?slide_id=567615ee:5	{from ServiceWo...	4 ms	
 index_index.css	200	stylesheet	editor?slide_id=567615ee:5	{from ServiceWo...	6 ms	
 global.js	200	script	editor?slide_id=567615ee:122	{from ServiceWo...	5 ms	
 index_index.js	200	script	editor?slide_id=567615ee:123	{from ServiceWo...	45 ms	

注意版本更新问题

| 提纲

1. 架构 ✓

一个MVVM式的组件化架构 ✓

2. 开发

组件多 ✓

组件重 ✓

3. 填坑优化

前端框架的坑 ✓

本地化/离线化 ✓



Q&A

