WEB CONF
<腾 讯 前 端 技 术 大 会/>

<kbone/>

微信小程序同构方案新思路

june

kbone 是什么？

微信小程序同构方案：

- mpvue
- wepy
- taro
- kbone
- ……

支持我们使用熟悉的方式（比如 Vue、React）来同时完成小程序端和 Web 端的开发

kbone 也是一种同构方案，只是实现思路和其他常见的方案不同，接下来就来介绍下它是如何诞生的。

内容大纲

1 背景

2 方案

3 应用

4 结语

Part 1 背景

背景：微信开放社区

现有 Web 端社区

为了让信息可以更方便有效地传播、分享和使用，需要在现有 Web 端实现上打造小程序端社区

同构到小程序需要考虑的因素

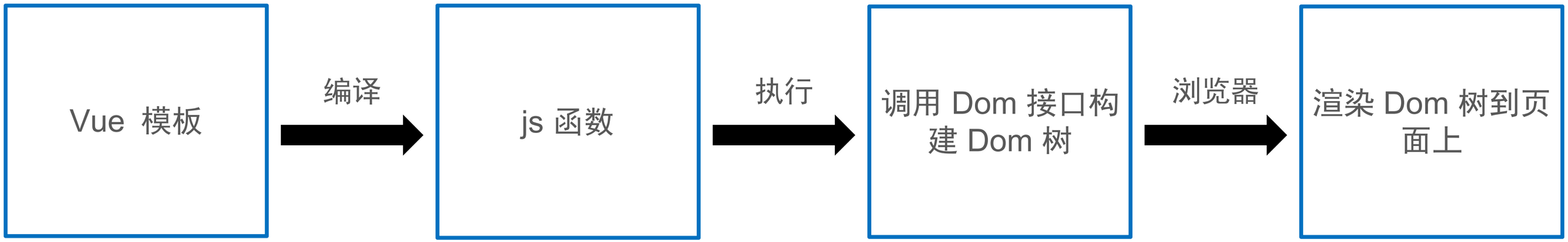
- 一套代码多端运行，便于后续维护
- 复用已有代码
- 尽可能支持已有的特性
- 保证性能

Part 2 方案

社区 Web 端实现：基于 Vue 框架

一款提供组件化的 JavaScript 框架

原理：



Vue 原理

评论组件（简化后的 Vue 模板）

```
<div>
    <label>评论: </label>
    <input />
</div>
```

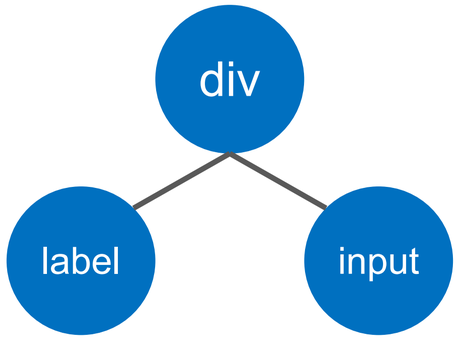
编译成 js 函数（调用 Dom 接口）

```
function render() {
    var div = document.createElement('div')
    var label = document.createElement('label')
    label.textContent = '评论: '
    div.appendChild(label)
    var input = document.createElement('input')
    div.appencChild(input)
}
```

浏览器渲染效果

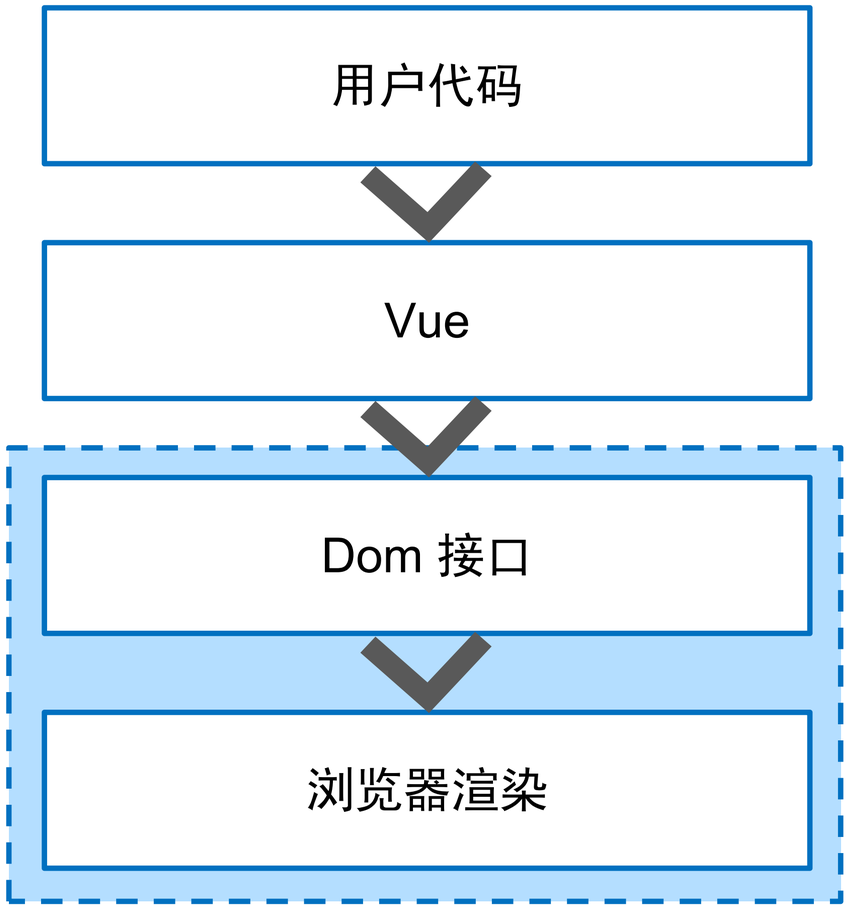
评论:

构建出 Dom 树



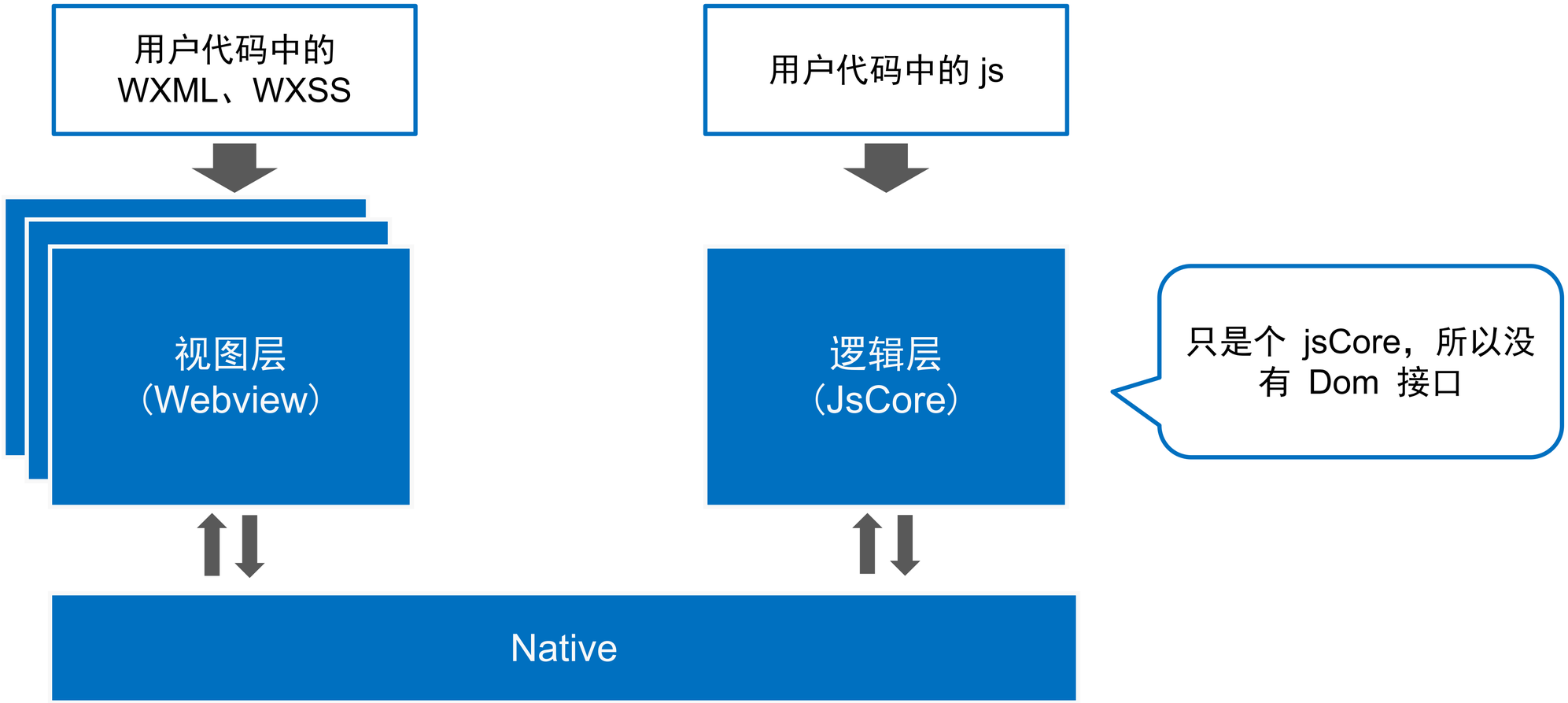
社区 Web 端代码能否直接在小程序上执行？

社区前端模型



浏览器提供的能力，在小程序中不存在，原因在于……

小程序的运行模型

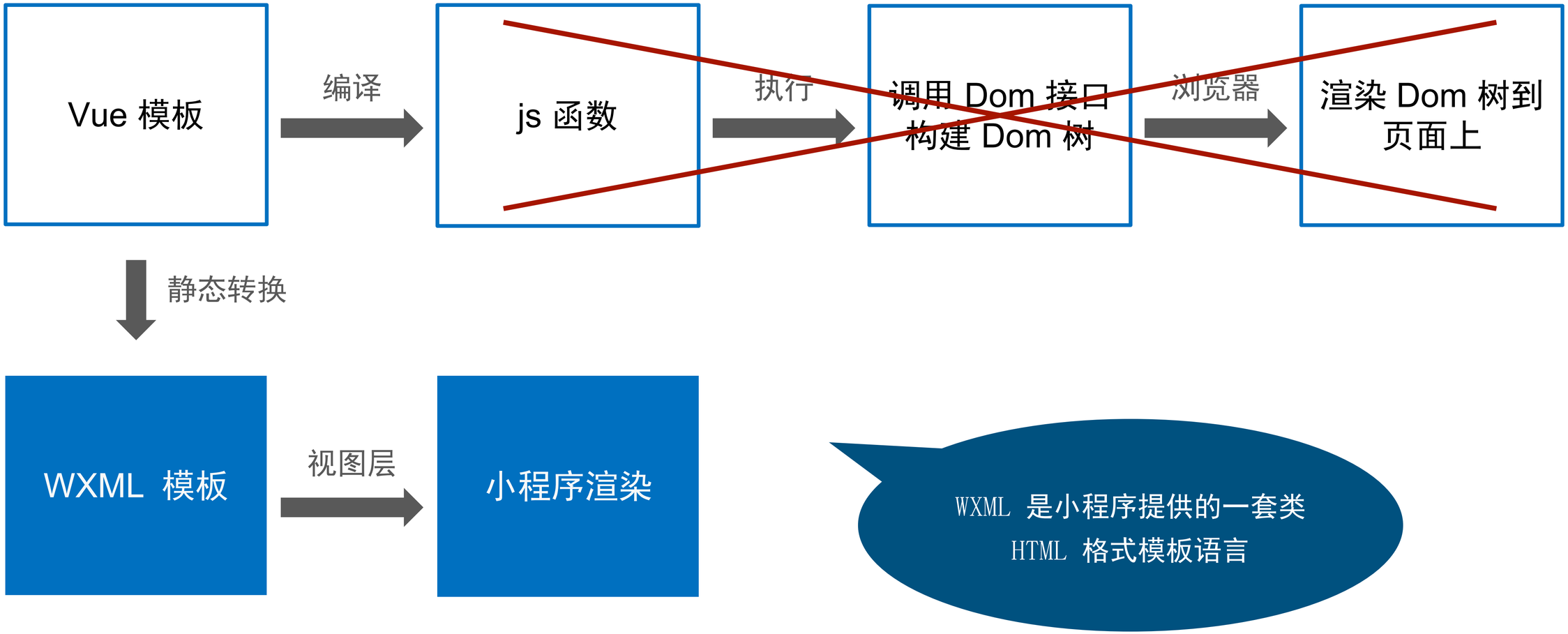


小程序双线程架构

如何将 Vue 代码转成小程序代码？

业界方案做法：将 Vue 模板静态转换成小程序 WXML 模板

业界方案原理


Vue 模板
编译
js 函数
执行
调用 Dom 接口
构建 Dom 树
浏览器
渲染 Dom 树到
页面上
静态转换
WXML 模板
视图层
小程序渲染
WXML 是小程序提供的一套类
HTML 格式模板语言

业界方案原理

评论组件（简化后的 Vue 模板）

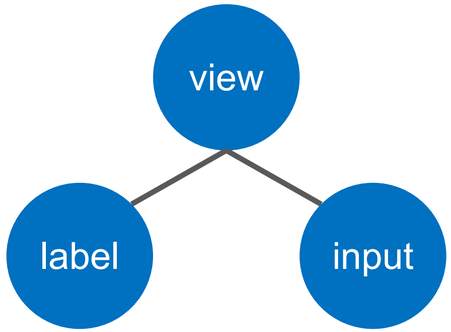
```
<div>
    <label>评论: </label>
    <input />
</div>
```

转译成 WXML 模板

```
<view>
    <label>评论: </label>
    <input />
</view>
```

小程序渲染效果

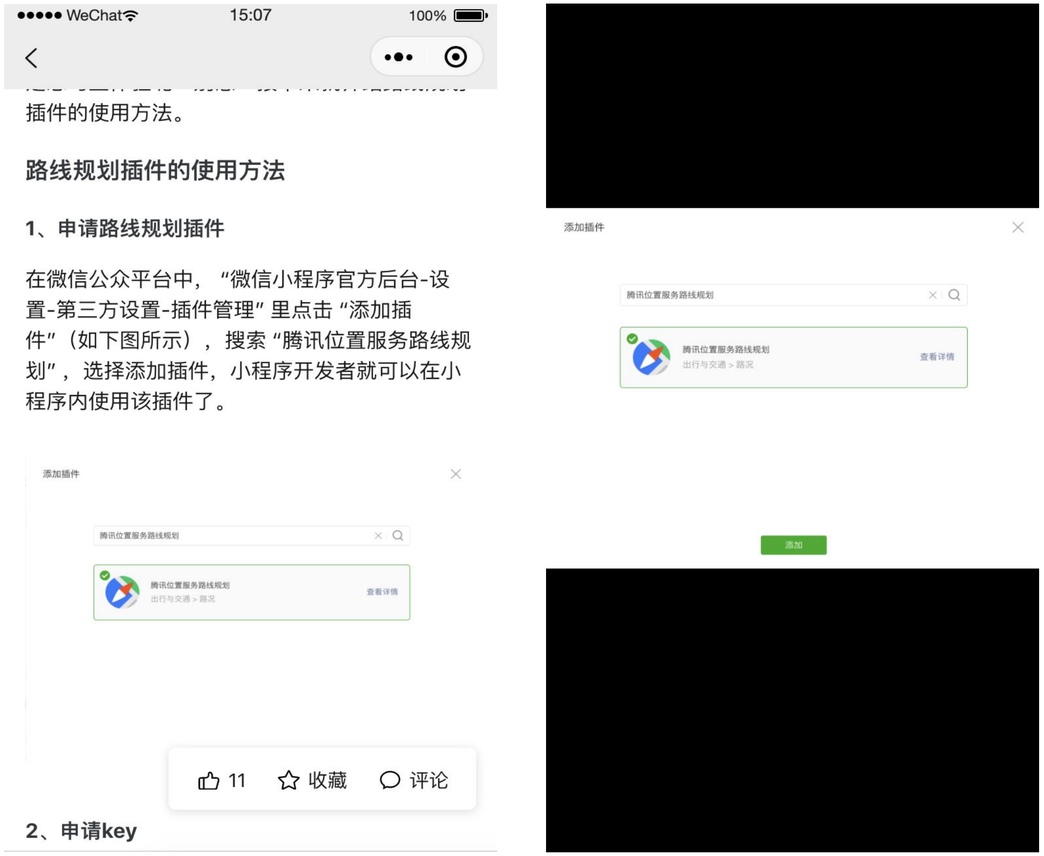
构建出小程序组件树



业界方案局限性

此方案简单场景可用，如果遇到一个复杂场景呢？

比如：展示社区帖子的富文本内容，点击内容中图片可预览



社区中的帖子详情页

```
<div>
    <div
        v-html="comment"
        @click="previewImage"
    ></div>
</div>
```

Vue 模板

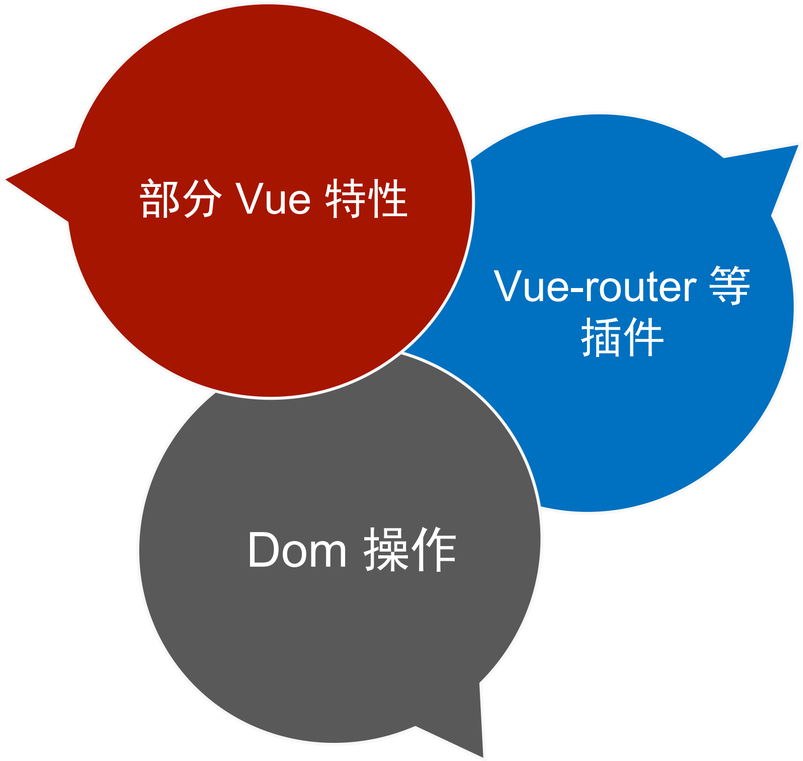
???

WXML 模板

业界方案局限性

两边语法不对等，静态转换会丢失部分 Vue 特性

业界方案局限性

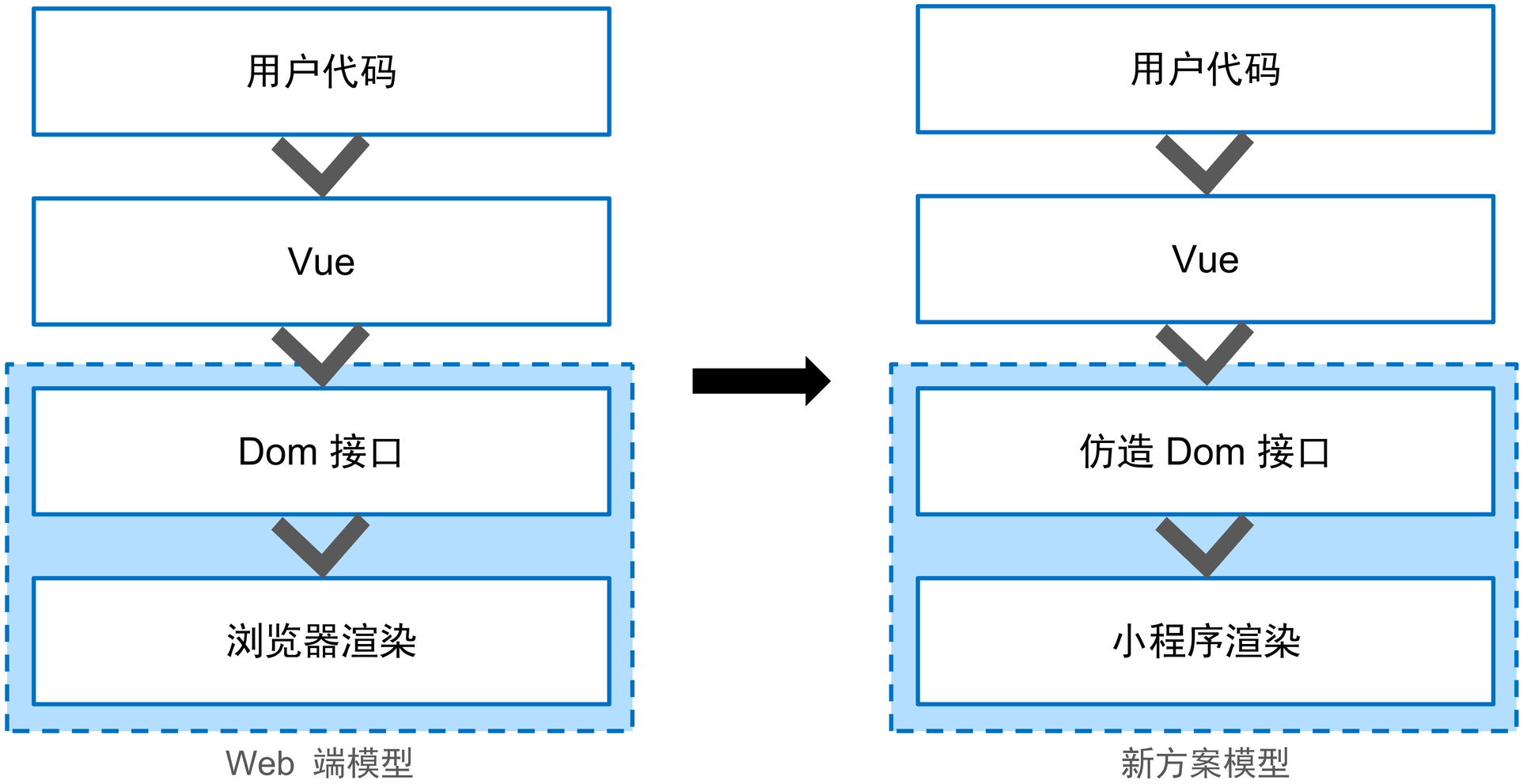


被社区高度依赖，但前述方案不支持！

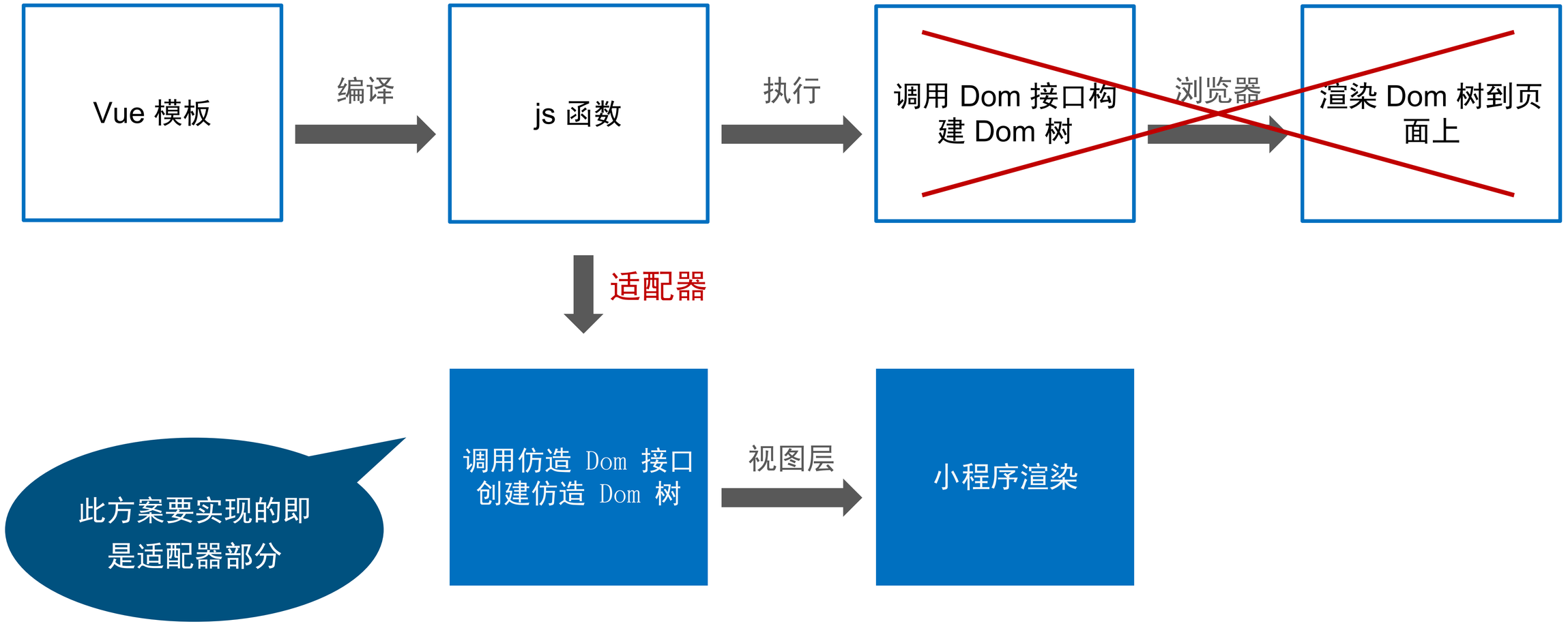
- 需要调整社区的部分交互
- 部分特性可以使用兼容方案重写，但重构成本巨大

此路不通，得换一种思路来解决 Vue 特性支持的问题

新方案设计



新方案原理


Vue 模板
编译
js 函数
执行
调用 Dom 接口构建 Dom 树
浏览器
渲染 Dom 树到页面上
适配器
此方案要实现的即是适配器部分
调用仿造 Dom 接口创建仿造 Dom 树
视图层
小程序渲染

这就是 kbone 的设计思路，接下来需要考虑如下问题：

1、如何创建仿造 Dom 树？
2、如何将仿造 Dom 树渲染到页面上？
3、如何监听用户事件？

1、如何创建仿造 Dom 树？

创建仿造 Dom 树

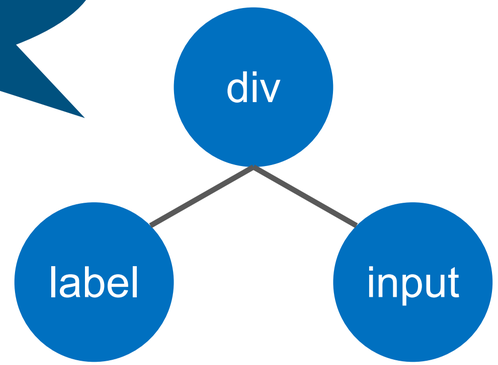
评论组件（简化后的 Vue 模板）

```
<div>
    <label>评论: </label>
    <input />
</div>
```

编译成 js 函数（调用 Dom 接口）

```
function render() {
    var div = document.createElement('div')
    var label = document.createElement('label')
    label.textContent = '评论: '
    div.appendChild(label)
    var input = document.createElement('input')
    div.appencChild(input)
}
```

在逻辑层内存中



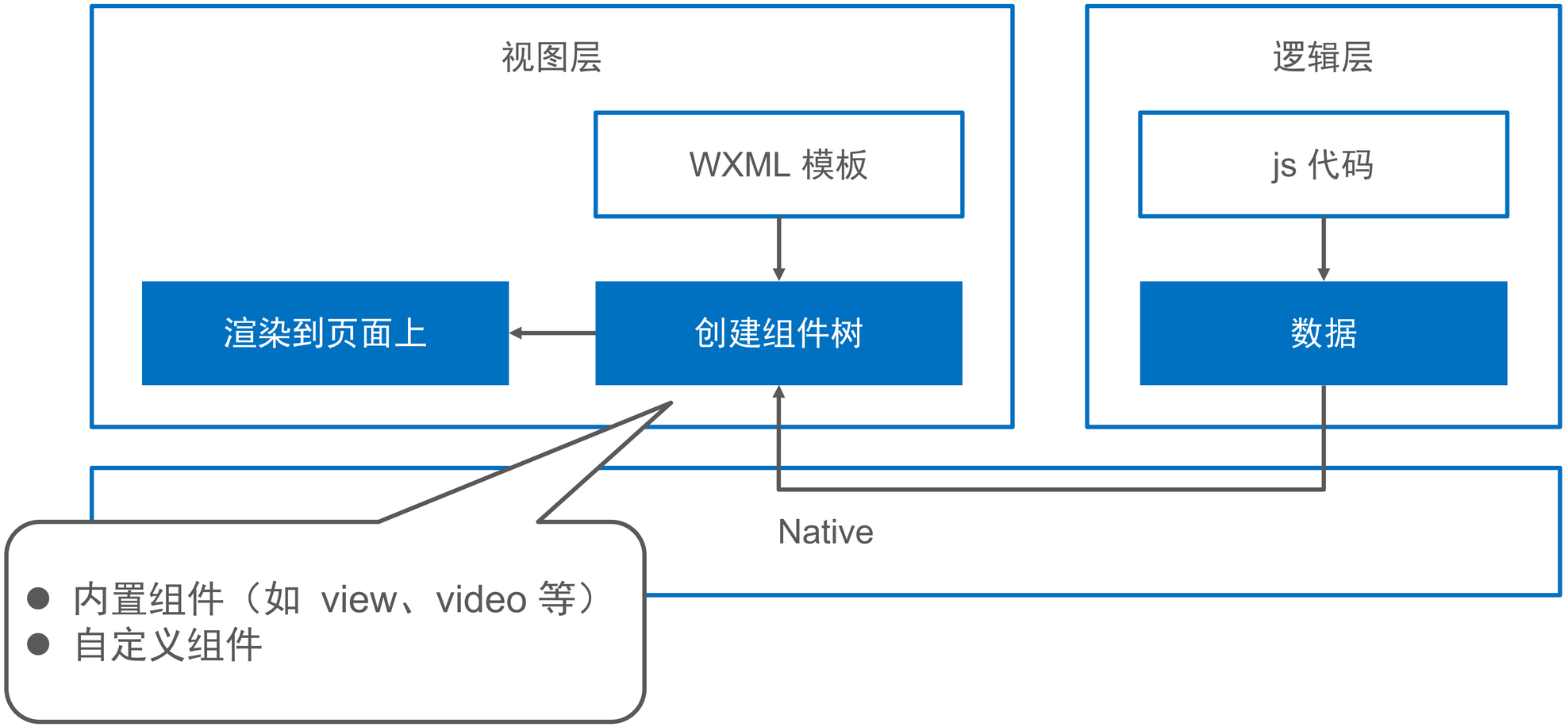
仿造 Dom 树

仿造 Dom 接口：

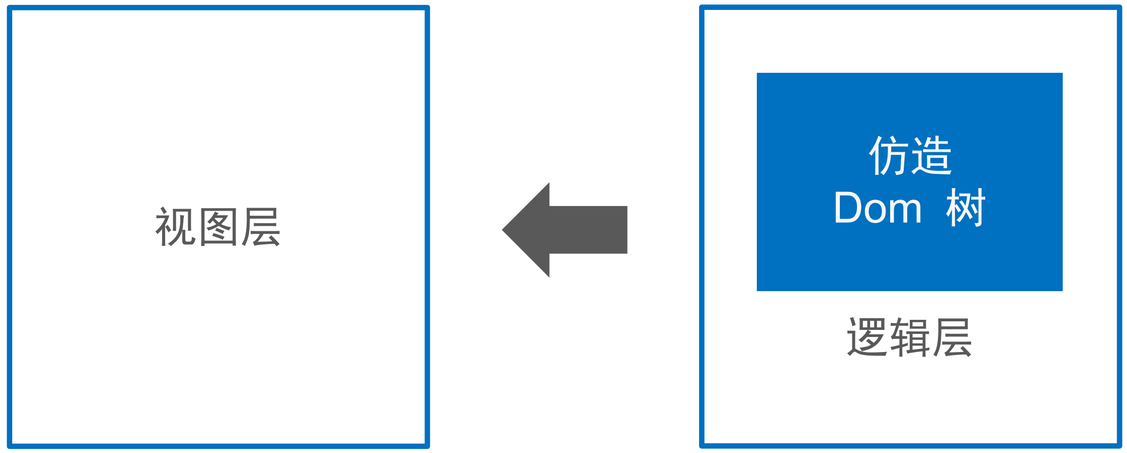
- createElement
- appendChild

2、如何将仿造 Dom 树渲染到页面上？

小程序渲染原理


视图层
逻辑层
WXML 模板
js 代码
渲染到页面上
创建组件树
数据
Native
内置组件（如 view、video 等）
自定义组件

渲染到页面上



小程序也没提供动态
创建节点的接口

在运行时动态生成的 Dom 树结构无法预知，无法提前生成对应的静态 WXML 模板

灵光一闪：能否利用前面提到的自定义组件来渲染仿造 Dom 树呢？

什么是自定义组件？

小程序渲染例子

```
<view>
    <text>{{content}}</text>

    <view>
        <button>确认</button>
    </view>

    <view>
        <button>取消</button>
    </view>
</view>
```

页面 WXML 模板

+

data: { content: ‘确认要发送？’ }

页面数据

渲染

自定义组件用法

```
<view>
    <button>{{text}}</button>
</view>
```

封装 → <custom-button> 组件

自定义组件

替换

替换

```
<view>
    <text>{{content}}</text>

    <view>
        <button>确认</button>
    </view>

    <view>
        <button>取消</button>
    </view>
</view>
```

页面 WXML 模板

简化 →

```
<view>
    <text>{{content}}</text>

    <custom-button text="确认" />

    <custom-button text="取消" />
</view>
```

页面 WXML 模板

自定义组件用法

自定义组件是一种将各种组件进行组装而成的组件

其模板可以是这样的：

```
<view>
    <button>{{text}}</button>
</view>
```

也可以是这样的：

```
<view>
    <custom-icon type="button" />
    <button>{{text}}</button>
</view>
```

还可以是这样的：

```
<view>
    <custom-button wx:if="{{hasNext}}" text="next" />
    <button>{{text}}</button>
</view>
```

任何组件均可以被组装，包括自己

可以递归创建组件

自定义组件递归用法

自己引用自己

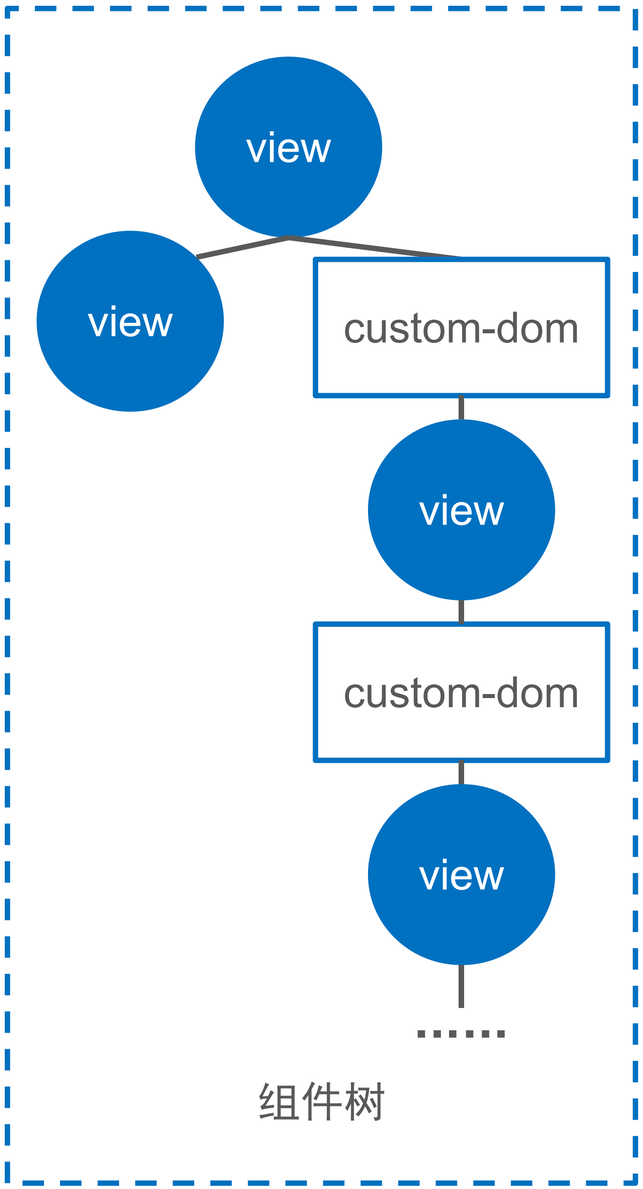
```
<view>
    <custom-dom />
</view>
```

封装

<custom-dom> 组件

```
<view>
    <view>page content: </view>
    <custom-dom />
</view>
```

页面 WXML 模板

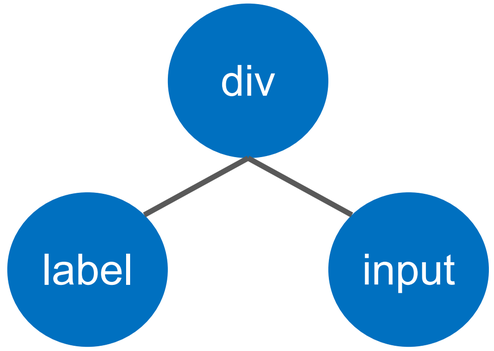


组件树

渲染到页面

评论组件（简化后的 Vue 模板）

```
<div>
    <label>评论: </label>
    <input />
</div>
```



仿造 Dom 树

回到最开始的例子，是否可以利用自定义组件的递归创建能力做些什么？

渲染到页面

将每一个 Dom 节点都映射到一个自定义组件实例上

仿造 Dom 树

组件树

如果创建的自定义组件过多会影响性能，这个后面会介绍如何优化

渲染到页面

具体实施：封装一个可递归的自定义组件

```
<!-- input 节点 -->
<input wx:if="{{attrs.name === 'input'}}" />
<!-- 其他 dom 节点 -->
<view wx:elif="{{attrs.name}}">
    <custom-dom wx:for="{{children}}" />
</view>
<!-- 文本节点 -->
<text wx:else>{{text}}</text>
```

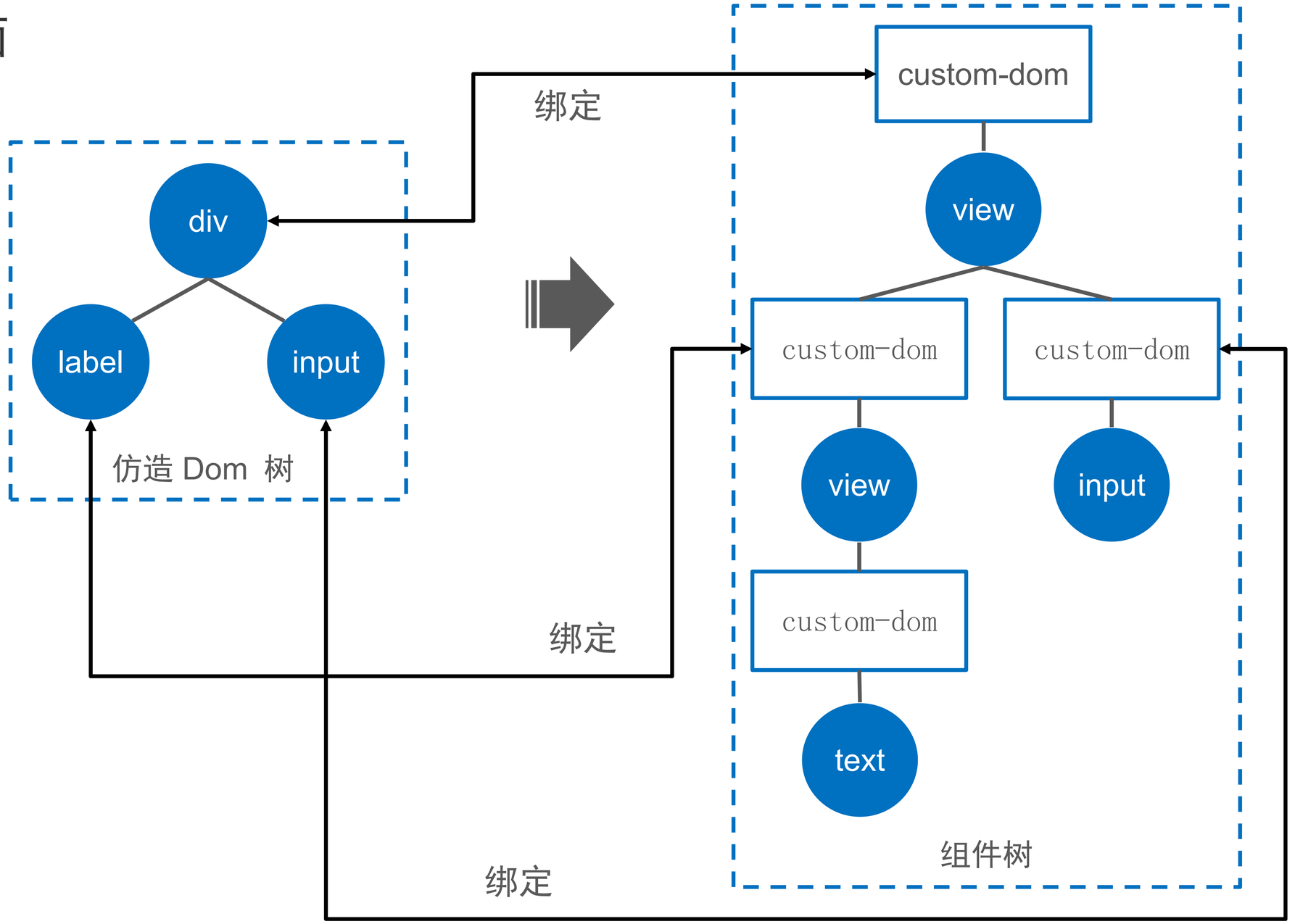
封装 → <custom-dom> 组件

+

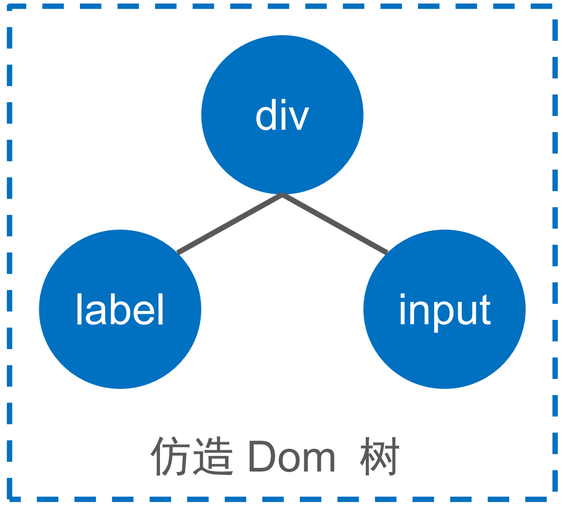
```
div: { attrs: { name: 'div' }, children: [ label, input ] }
label: { attrs: { name: 'label' }, children: [ '评论：' ] }
input: { attrs: { name: 'input' }, children: [ ] }
```

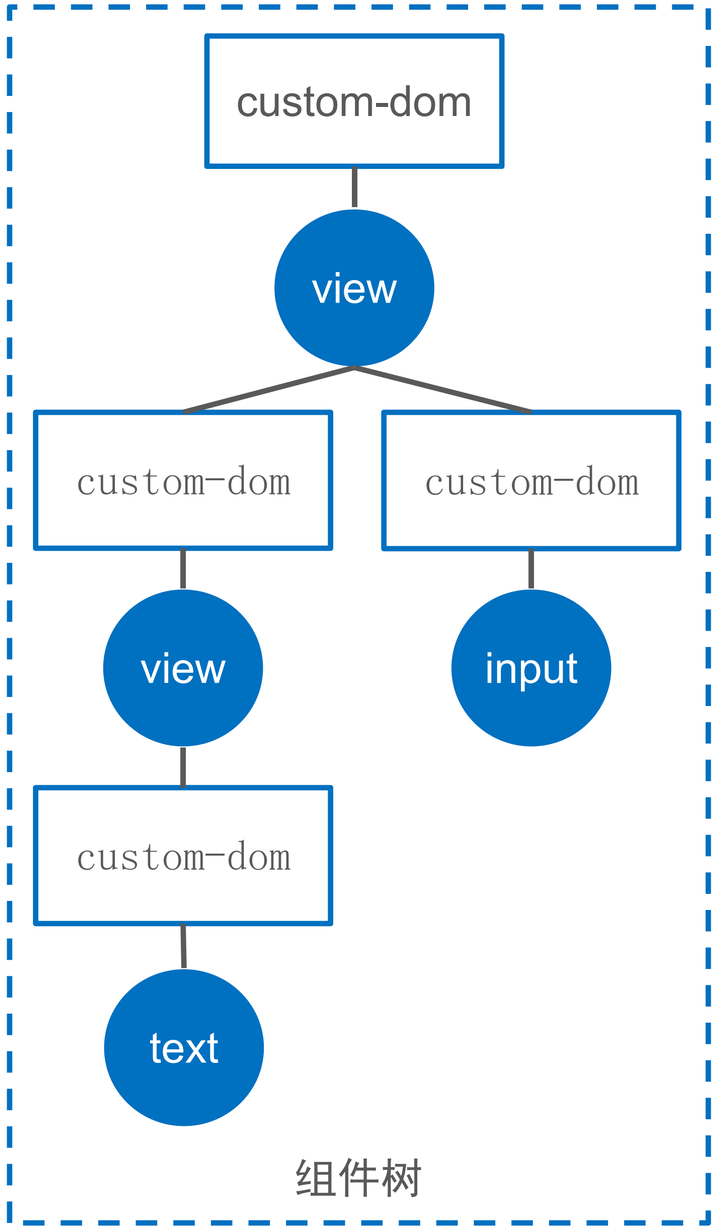
遇到非 Dom 节点或者 children 为空则终止递归

渲染到页面


绑定
div
label
input
仿造 Dom 树
custom-dom
view
custom-dom
custom-dom
view
input
custom-dom
text
组件树
绑定
绑定

渲染到页面





最终渲染的效果：

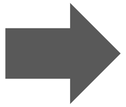
3、如何监听用户事件？

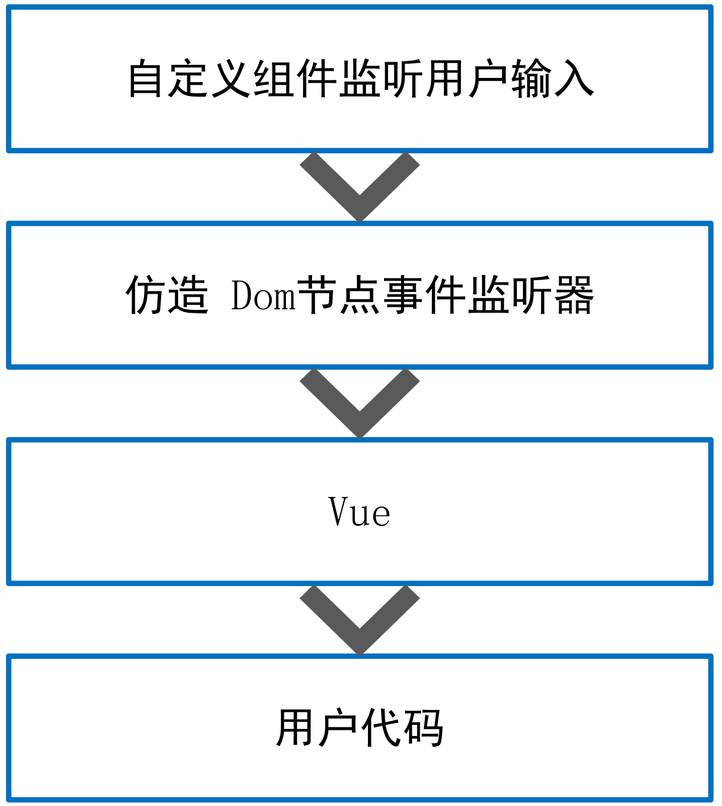
事件监听

用户在输入框输入内容：

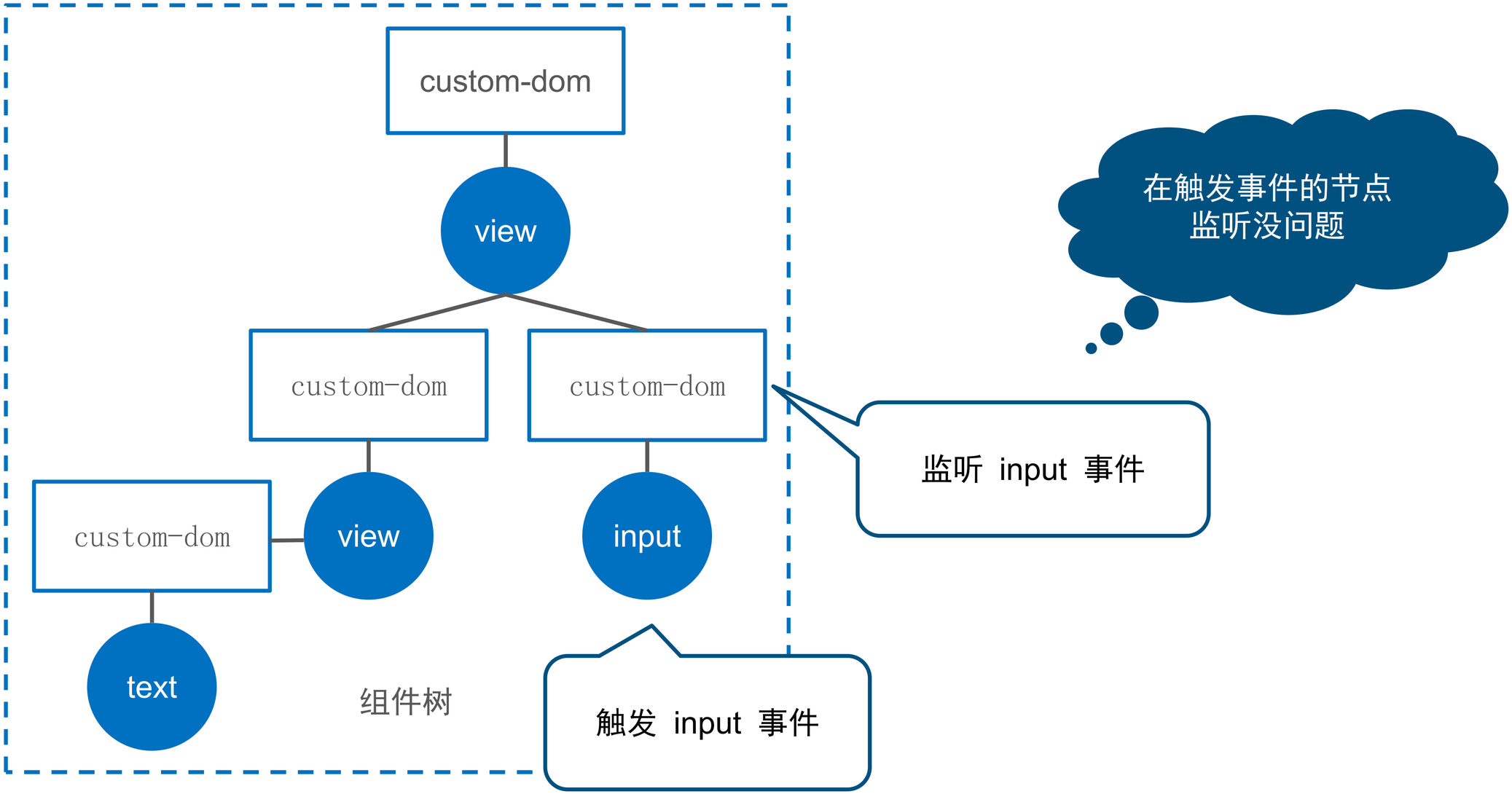
代码中监听到输入事件后，发现评论内容不符合要求，需要给个提示

```
div.addEventListener('input', function(event) {
    var input = event.target
    if (input.value.length < 20) {
        var tip = document.createElement('div')
        tip.textContent = '评论长度必须大于20'
        div.appendChild(tip)
    }
})
```

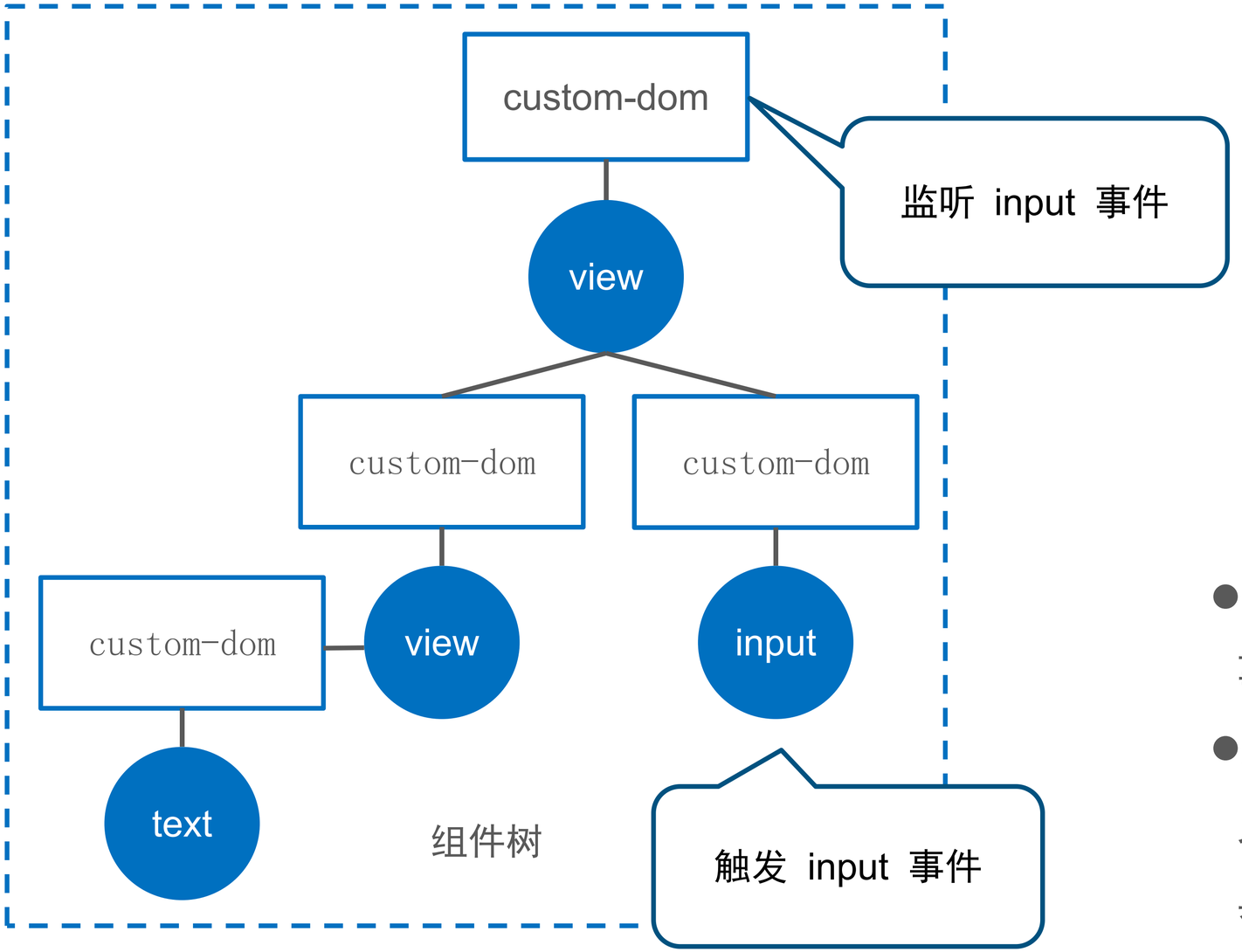




事件监听


custom-dom
view
custom-dom
custom-dom
custom-dom
view
input
text
组件树
在触发事件的节点
监听没问题
监听 input 事件
触发 input 事件

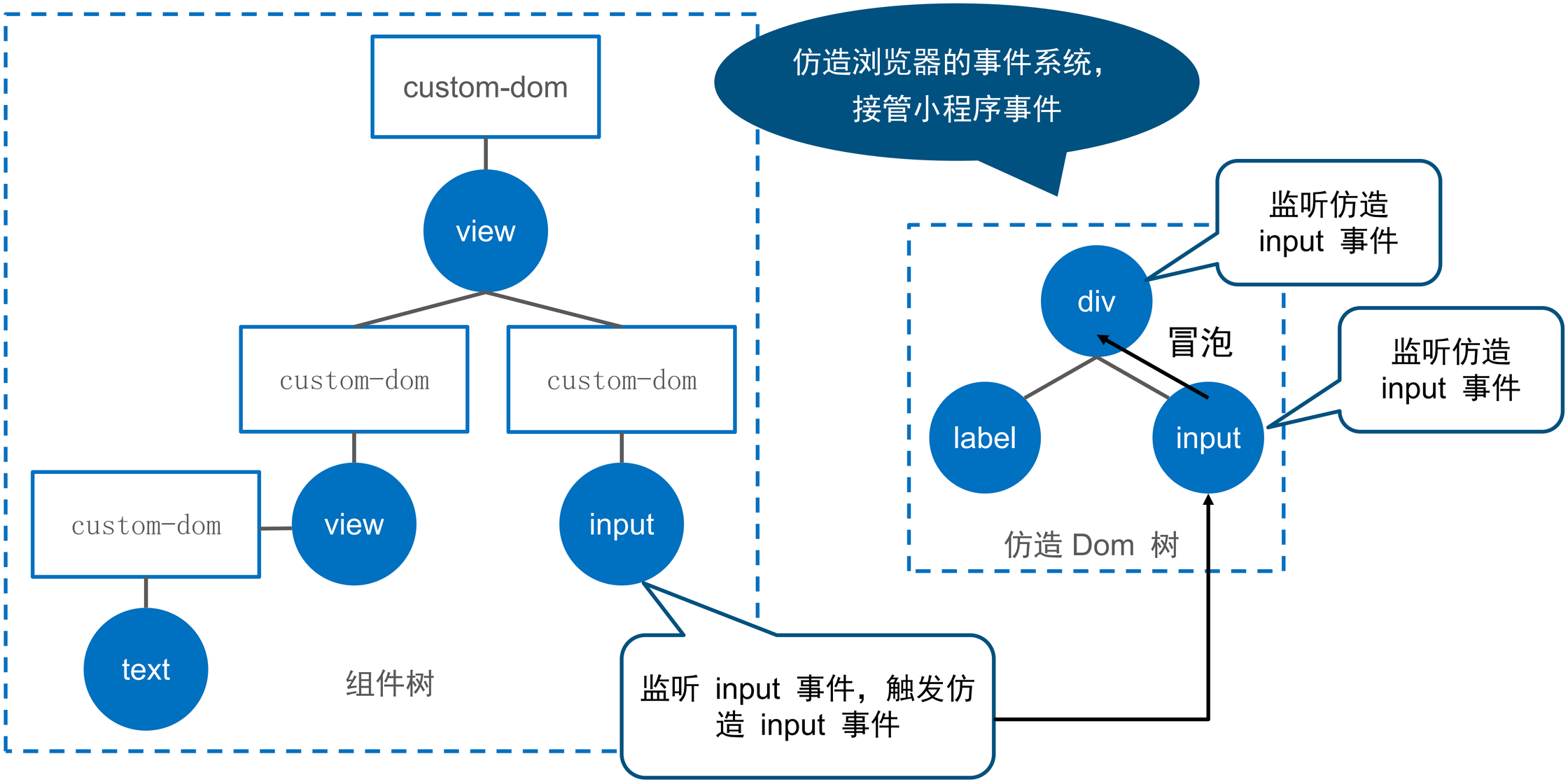
事件监听



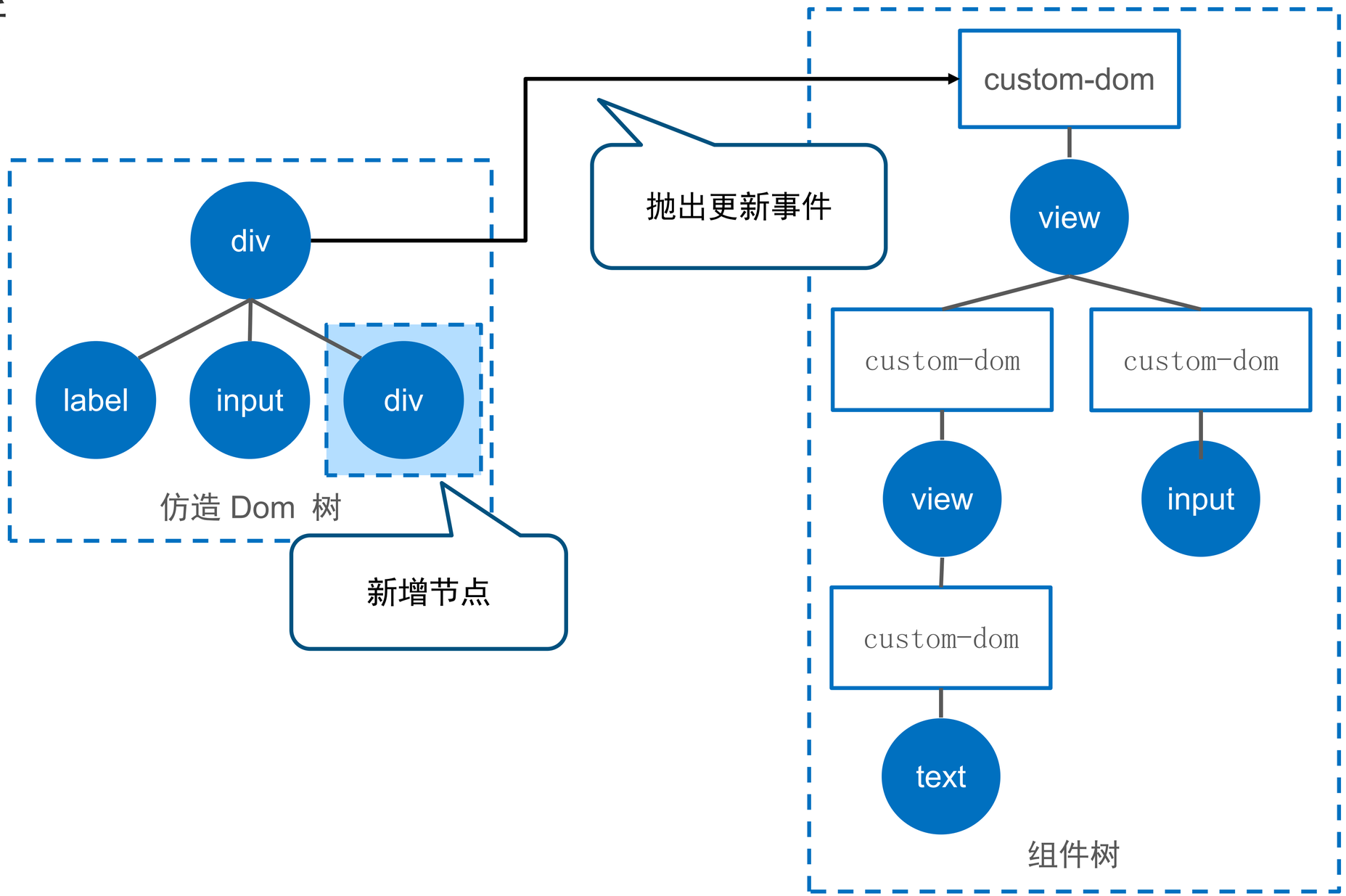
Web 端可以监听到，小程序却不行，怎么回事？

- 小程序有些事件无法冒泡，input 事件就是其中之一
- 对于可以冒泡的事件，因为自定义组件封装的特性，也无法知道具体来自哪个节点

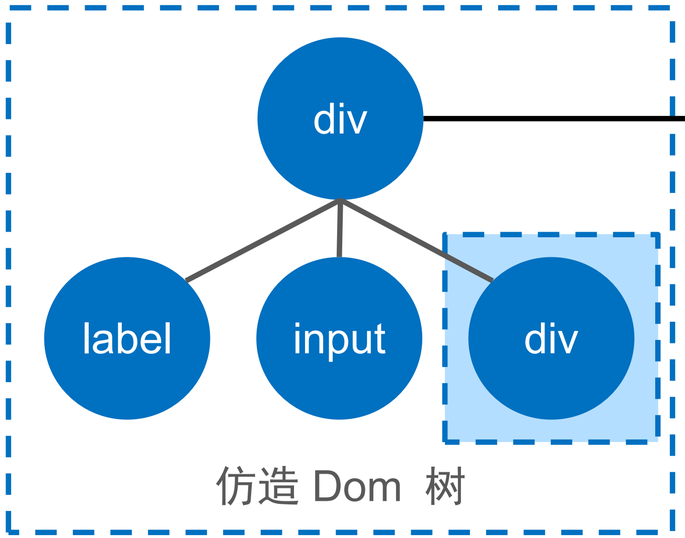
事件监听

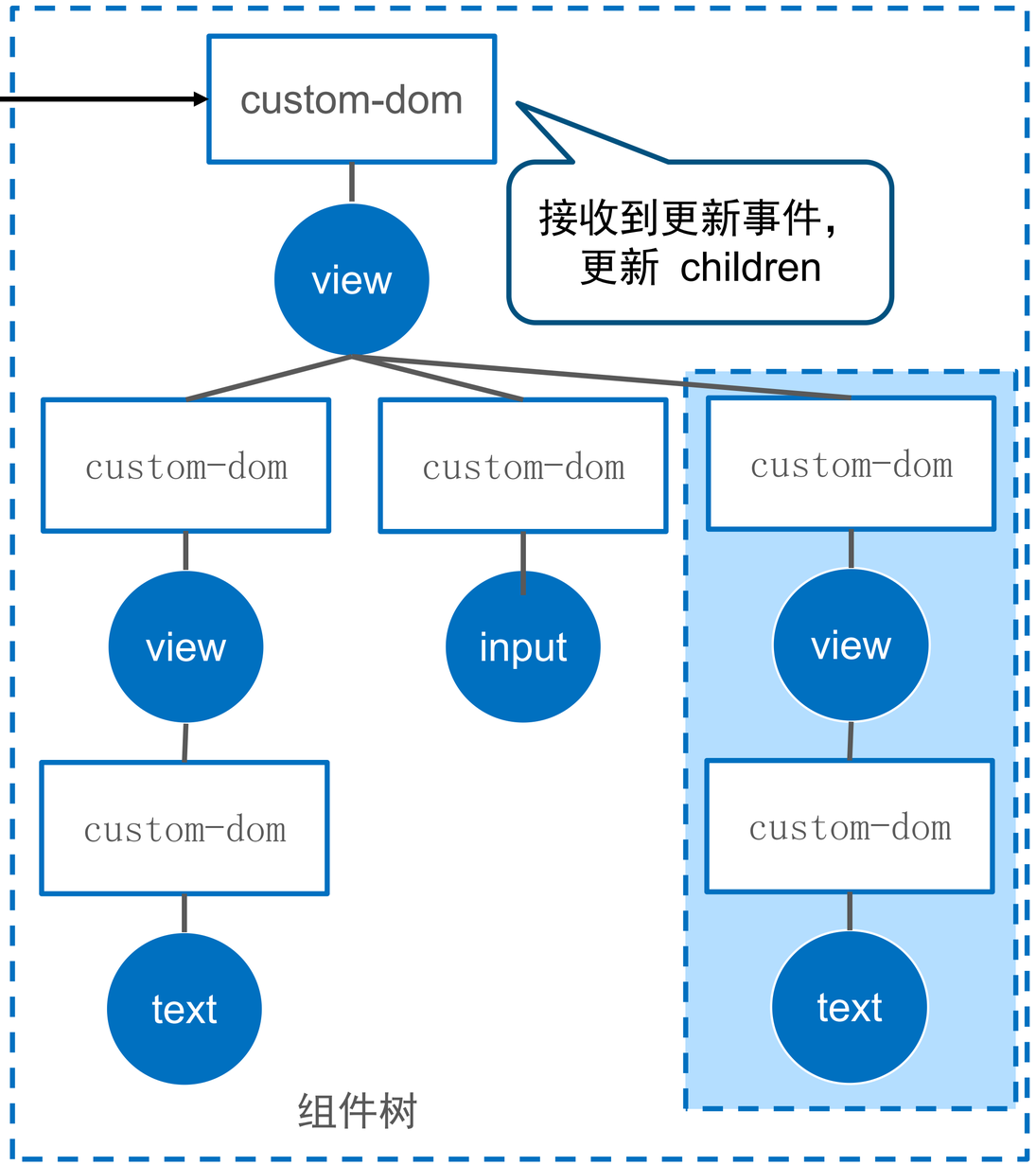

custom-dom
view
custom-dom
custom-dom
custom-dom
view
input
text
组件树
仿造浏览器的事件系统，
接管小程序事件
监听仿造
input 事件
div
冒泡
label
input
监听仿造
input 事件
仿造 Dom 树
监听 input 事件，触发仿
造 input 事件

事件监听


custom-dom
抛出更新事件
view
div
label
input
div
custom-dom
custom-dom
仿造 Dom 树
view
input
新增节点
custom-dom
text
组件树

事件监听

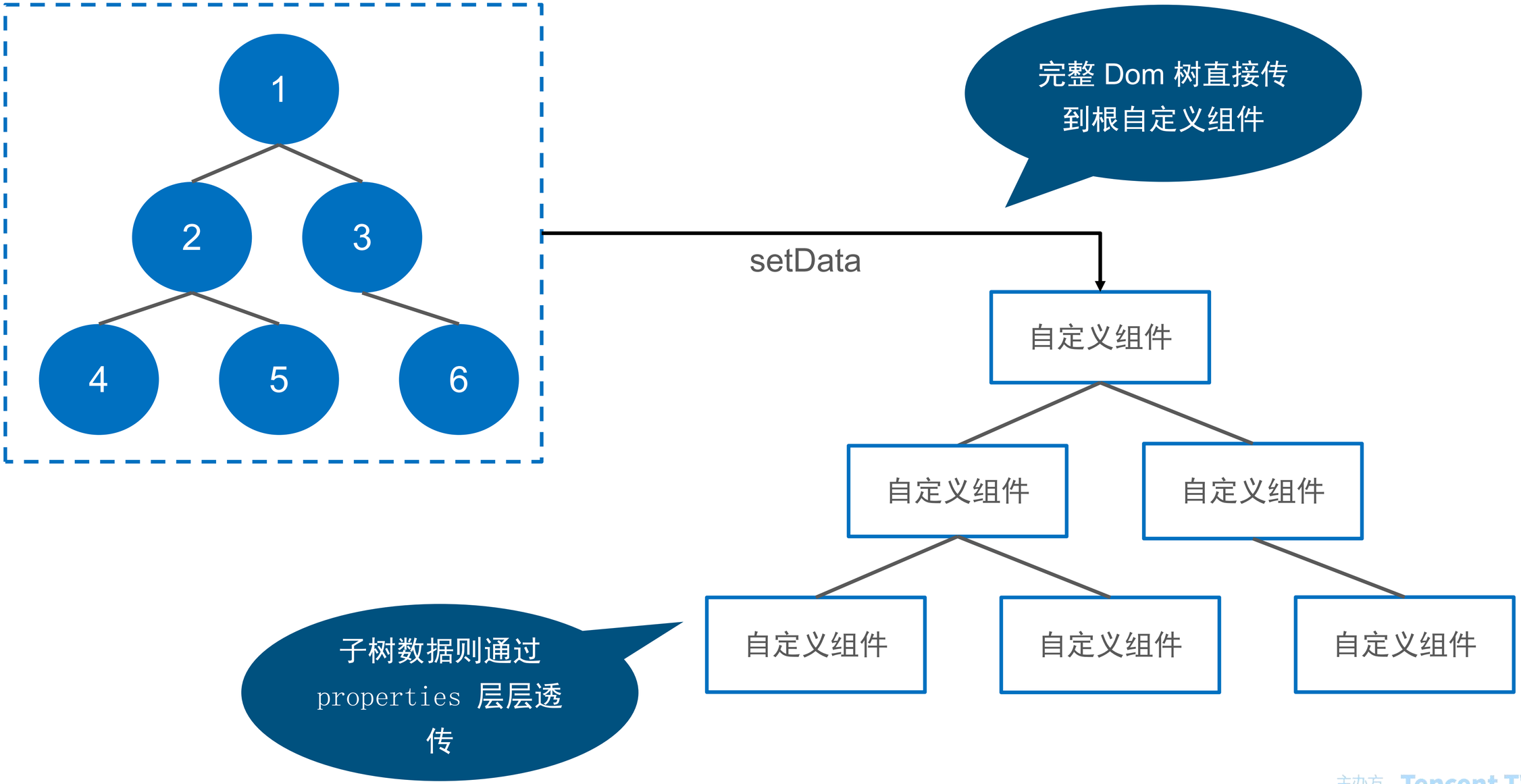




最终渲染的效果：

细节1：如何将 Dom 树传给视图层？

方案一


1
2
3
4
5
6
完整 Dom 树直接传到根自定义组件
setData
自定义组件
自定义组件
自定义组件
自定义组件
自定义组件
自定义组件
子树数据则通过 properties 层层透传

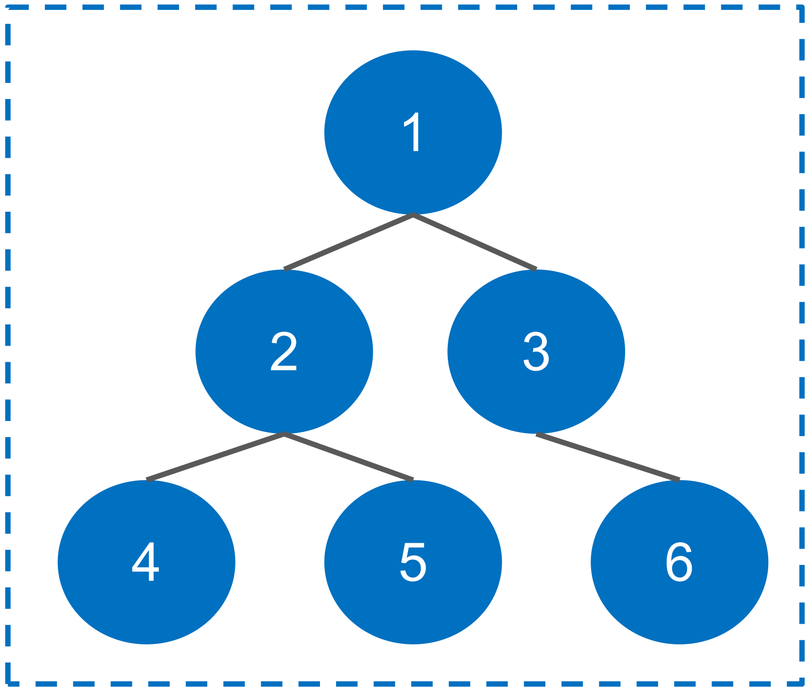
方案一

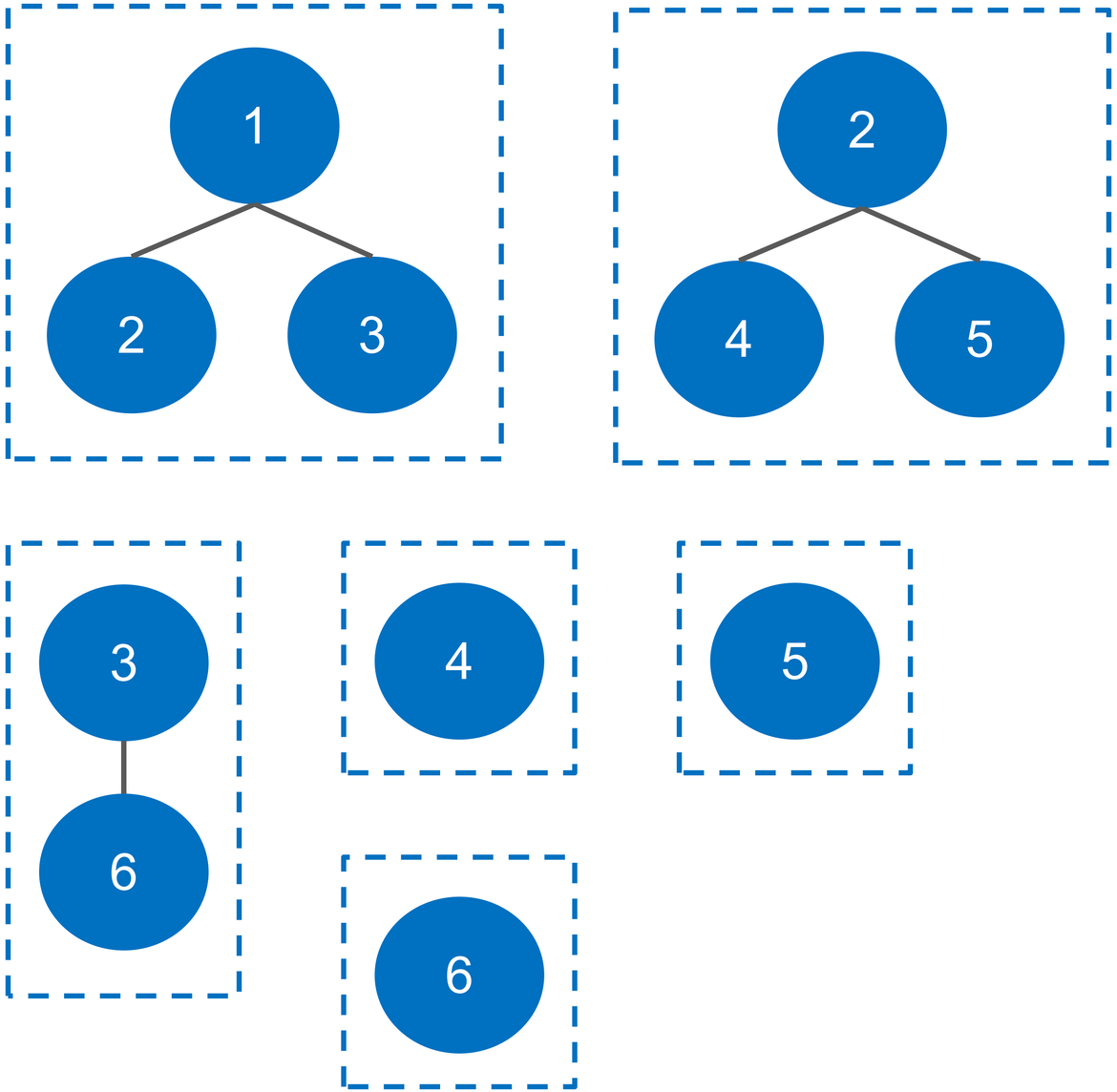
问题1、每次更新都需要做整棵树的 diff，存在性能损耗。

问题2、局部更新可能需要传输大量不变动的数据。

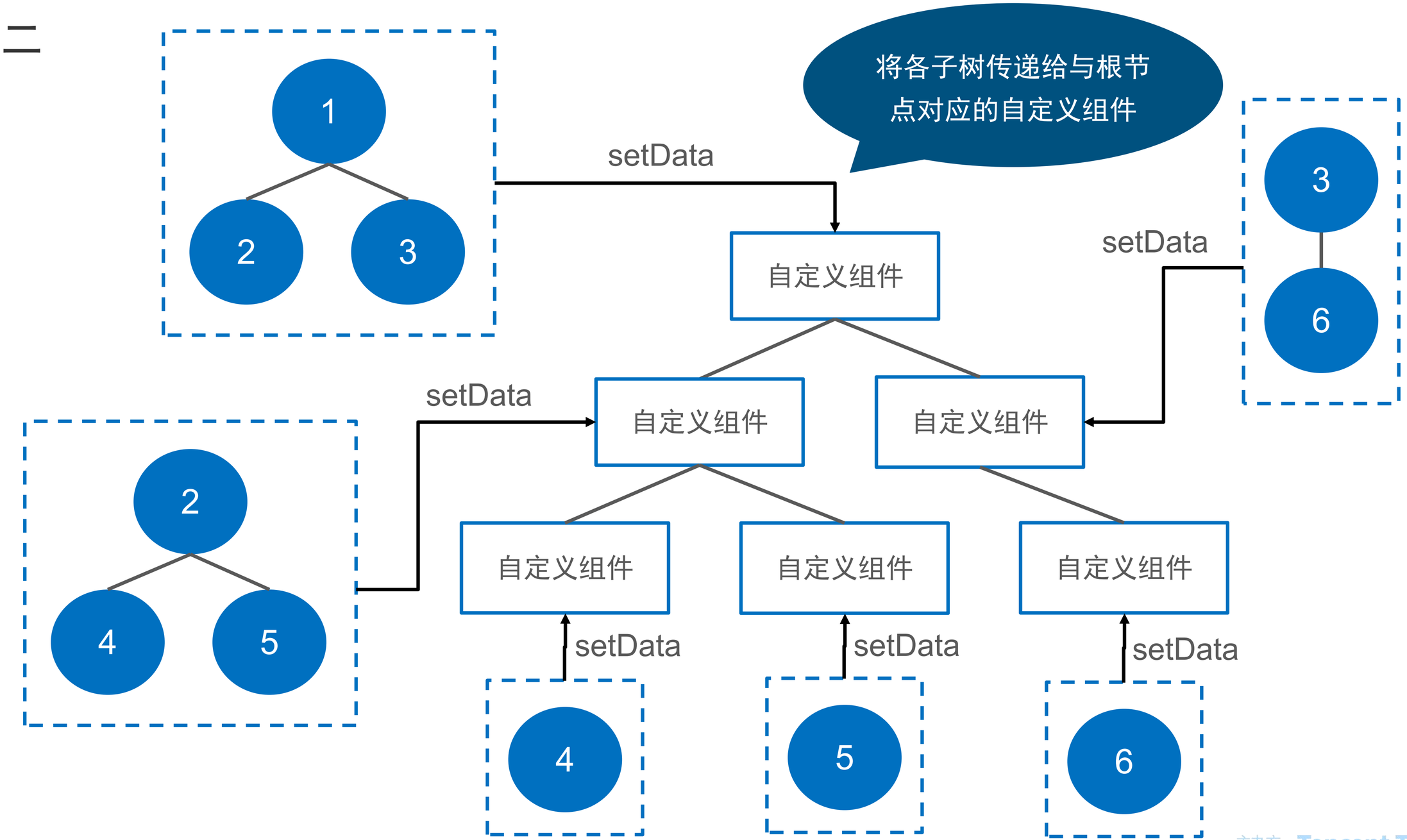
方案二

将每个节点作为根节点，
拆出至多两层的子树





方案二


1
2
3
setData
将各子树传递给与根节点对应的自定义组件
自定义组件
setData
3
6
setData
自定义组件
自定义组件
2
4
5
自定义组件
自定义组件
自定义组件
setData
setData
setData
4
5
6

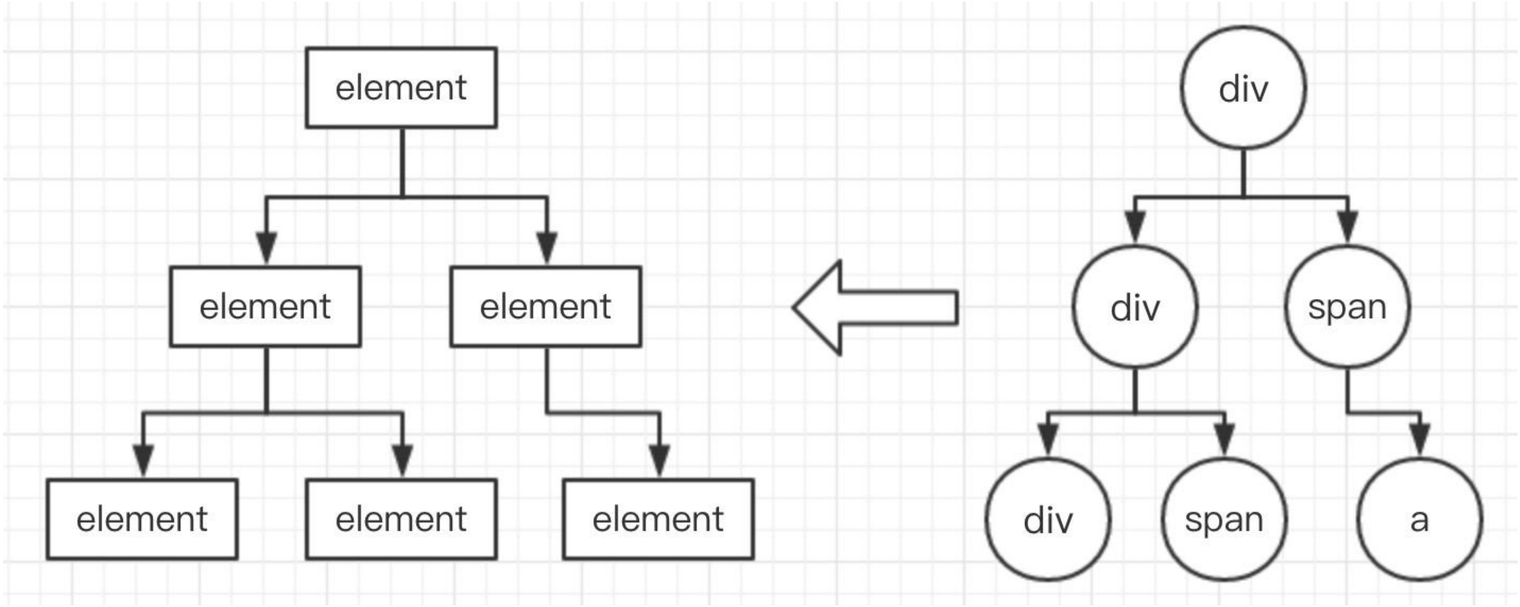
方案二

优势1：局部更新只做局部 diff。

优势2：每个自定义组件只需要自己和子节点数据，减少单次更新需要传输的数据量。

细节2:自定义组件创建过程开销较大，创建过多实例影响性能怎么办？

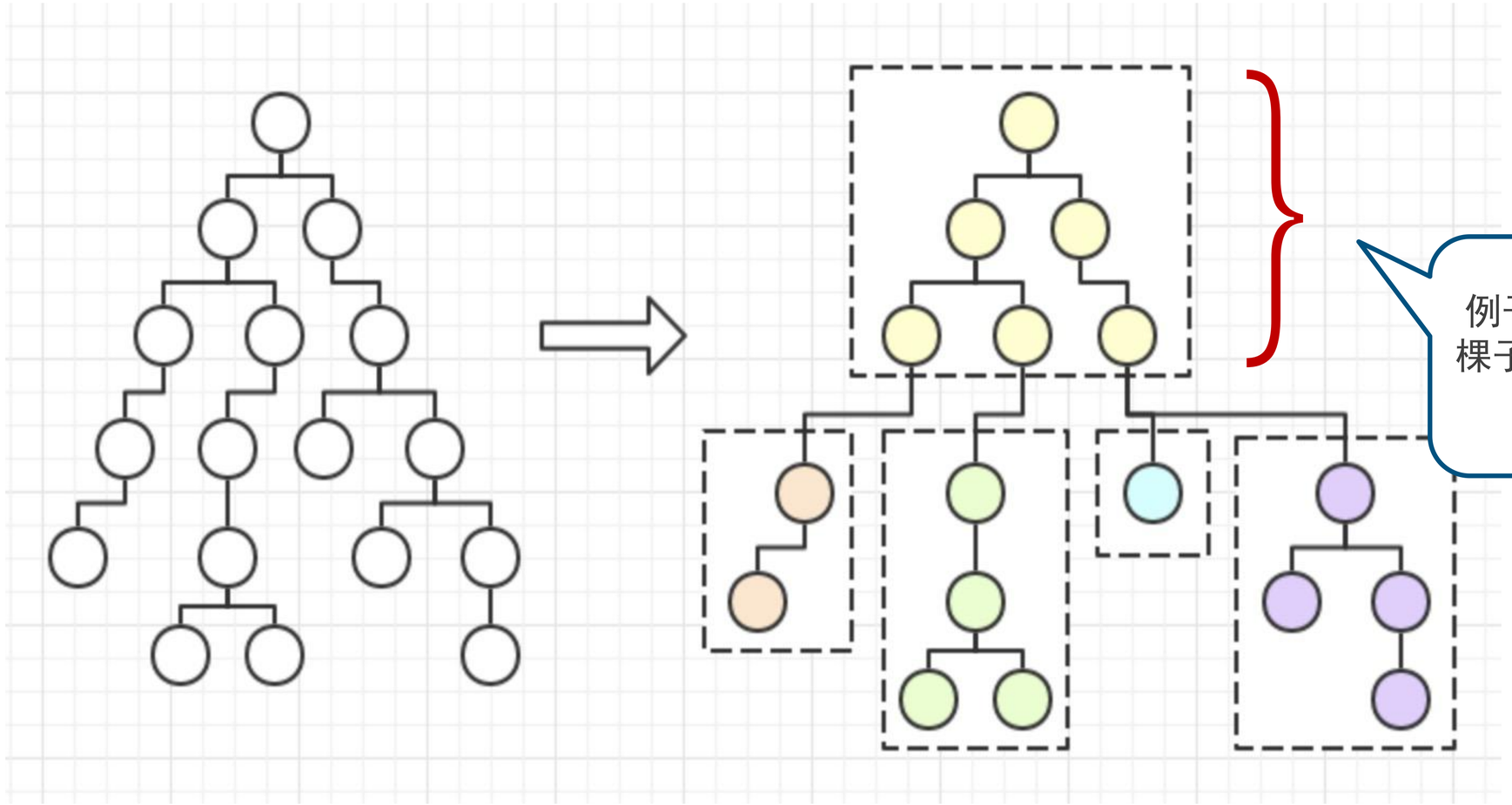
减少实例创建



依照初始方案设计，一个 Dom 节点映射一个自定义组件实例

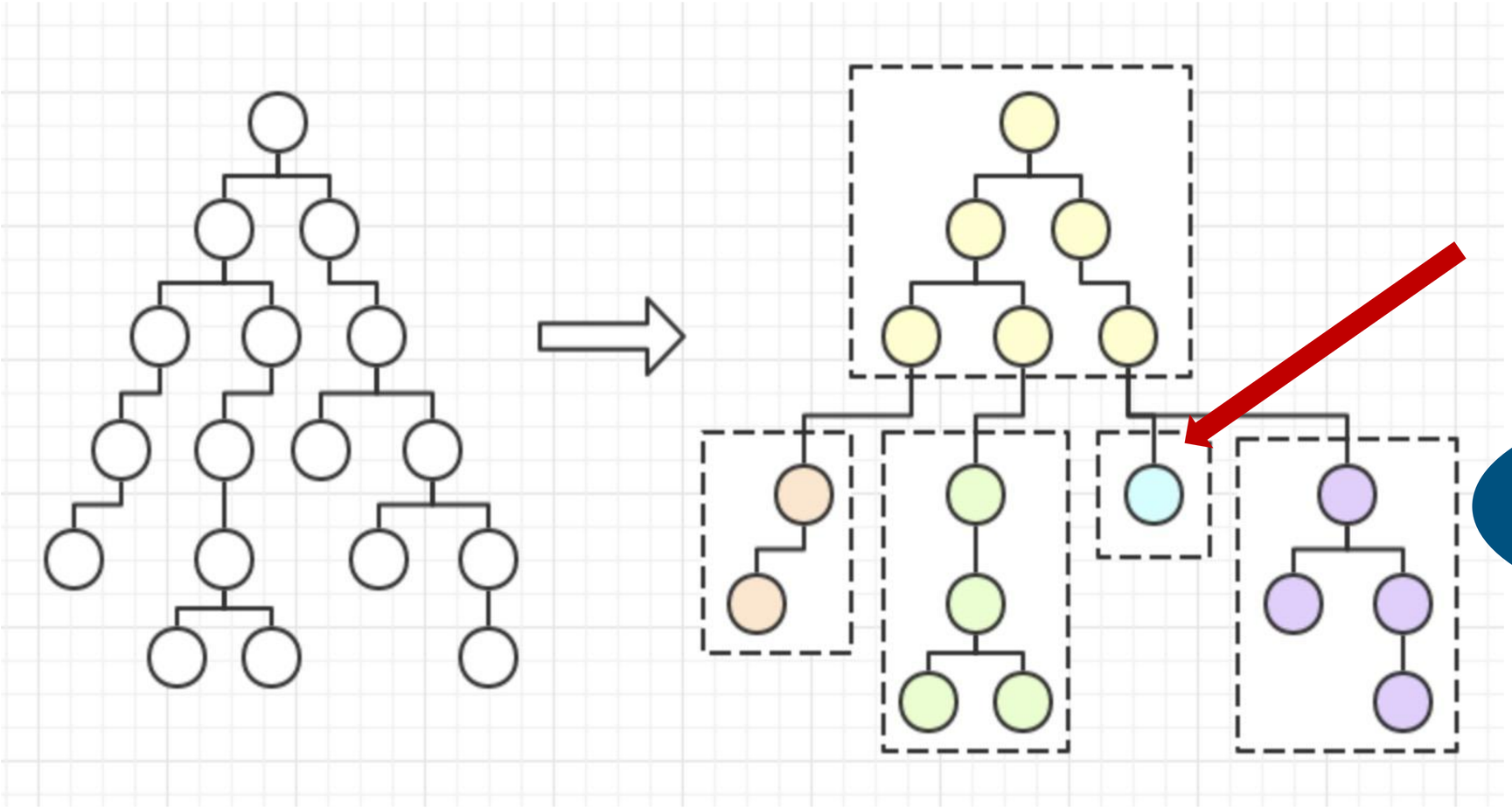
有没有办法让多个 Dom 节点映射一个自定义组件实例？

减少实例创建



17个 Dom 节点 = 5个自定义组件实例

减少实例创建



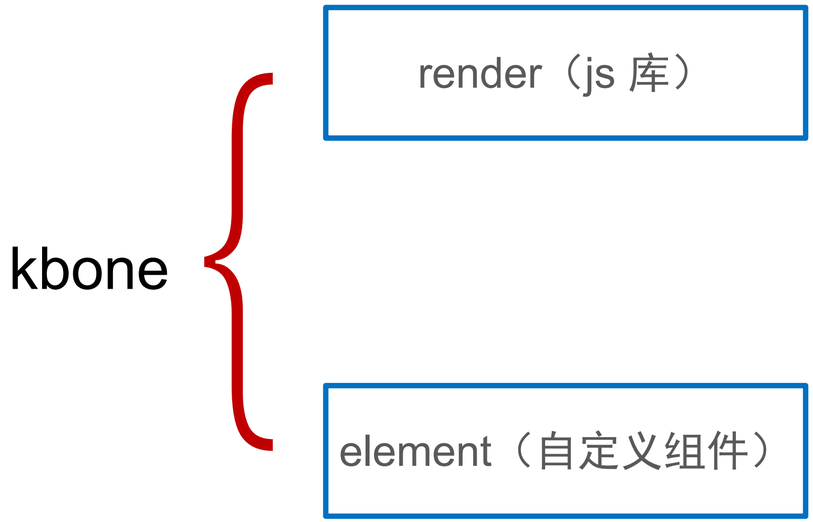
只有1个节点，却要创建1个自定义组件实例，太浪费！

单一叶子节点可以合并到上一棵子树中

其他细节

- Dom 节点、Style、ClassList 等对象复用
- Style、ClassList 等对象延迟创建
- Dom 节点属性值删减
- ……

方案总结

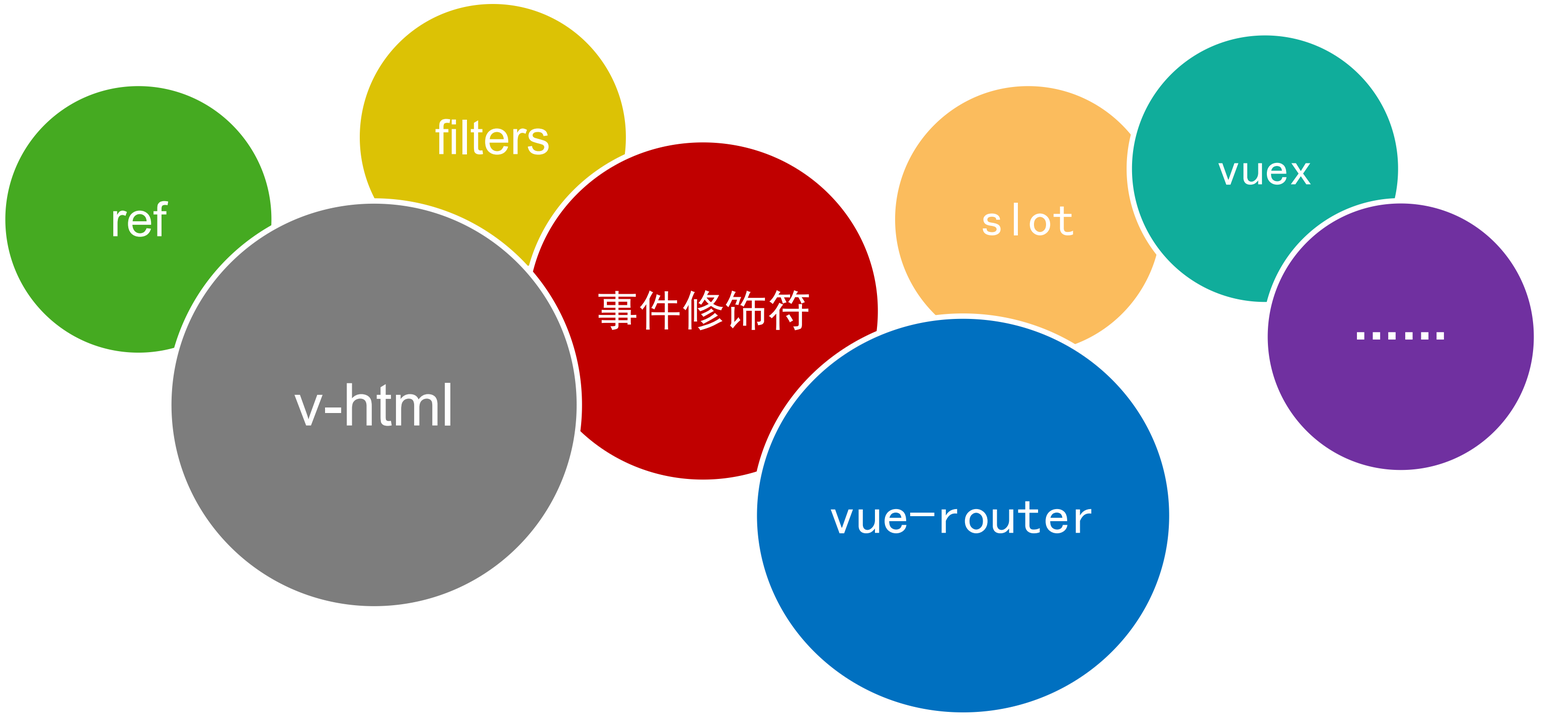


- 仿造 Dom/Bom 接口
- 构造仿造 Dom 树

- 监听仿造 Dom 树变化，渲染到页面
- 监听用户行为，触发仿造 Dom 树事件

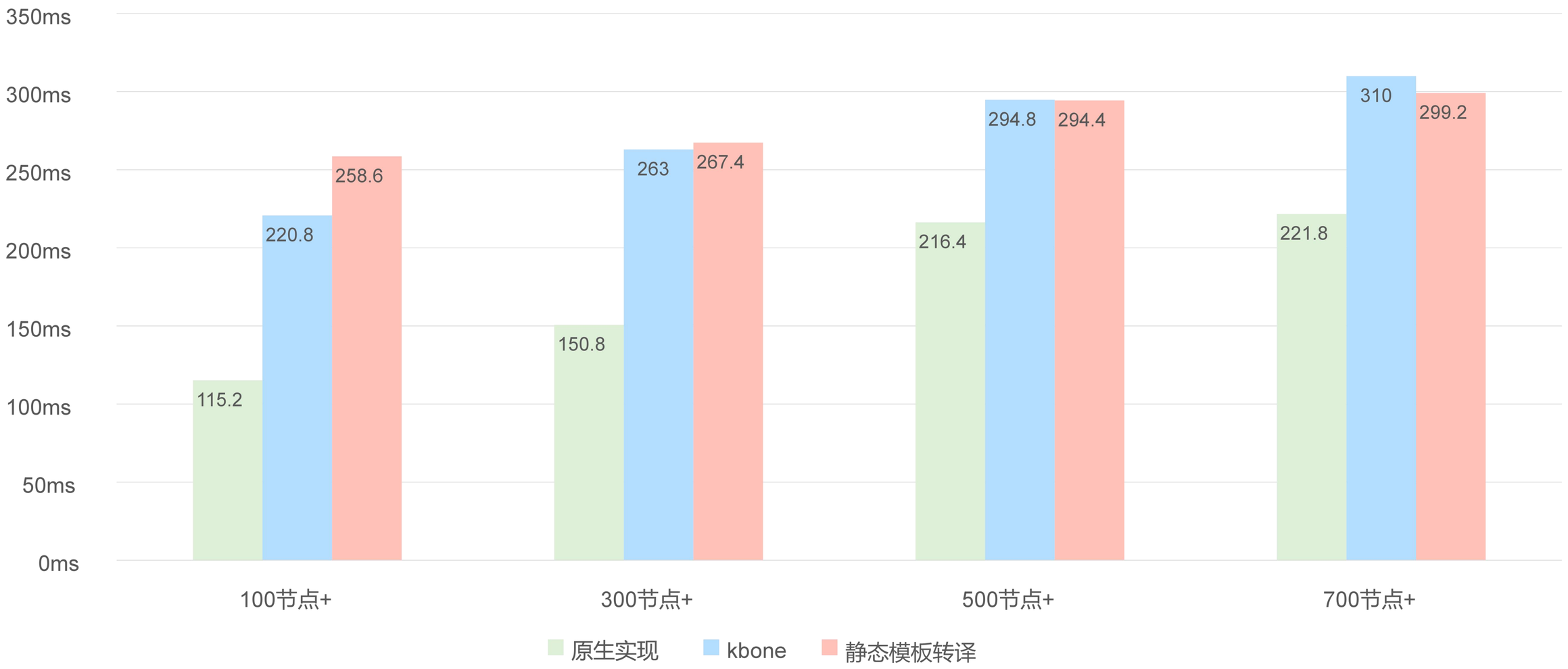
方案总结

因为是最底层的地方实现适配，所以几乎所有 vue 特性都可以直接使用：



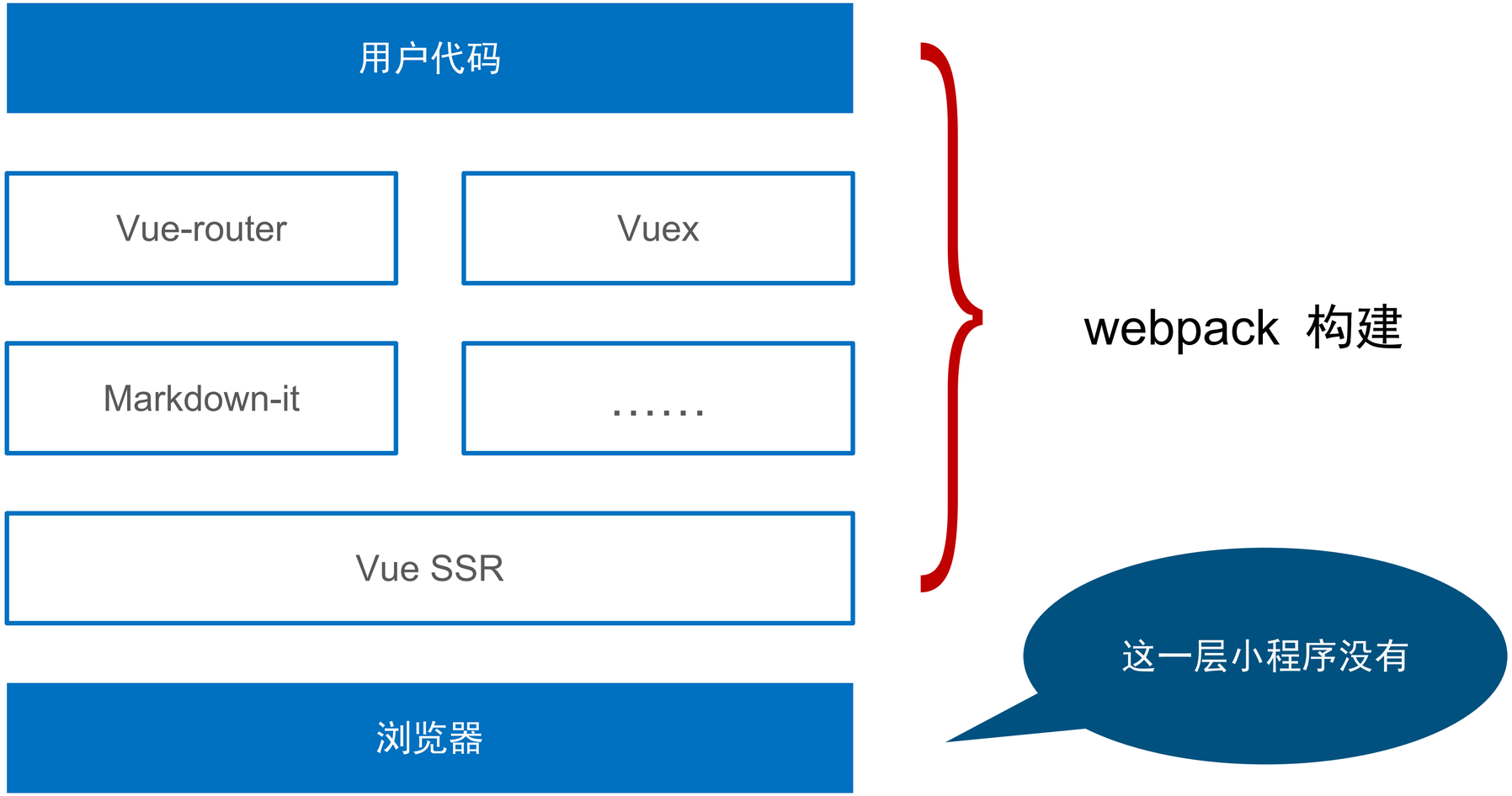
- Dom 操作
- window.location
- window.history
- document.cookie
- ……

渲染性能


350ms
300ms
250ms
200ms
150ms
100ms
50ms
0ms
115.2
220.8
258.6
150.8
263
267.4
216.4
294.8
294.4
221.8
310
299.2
100节点+
300节点+
500节点+
700节点+
原生实现
kbone
静态模板转译

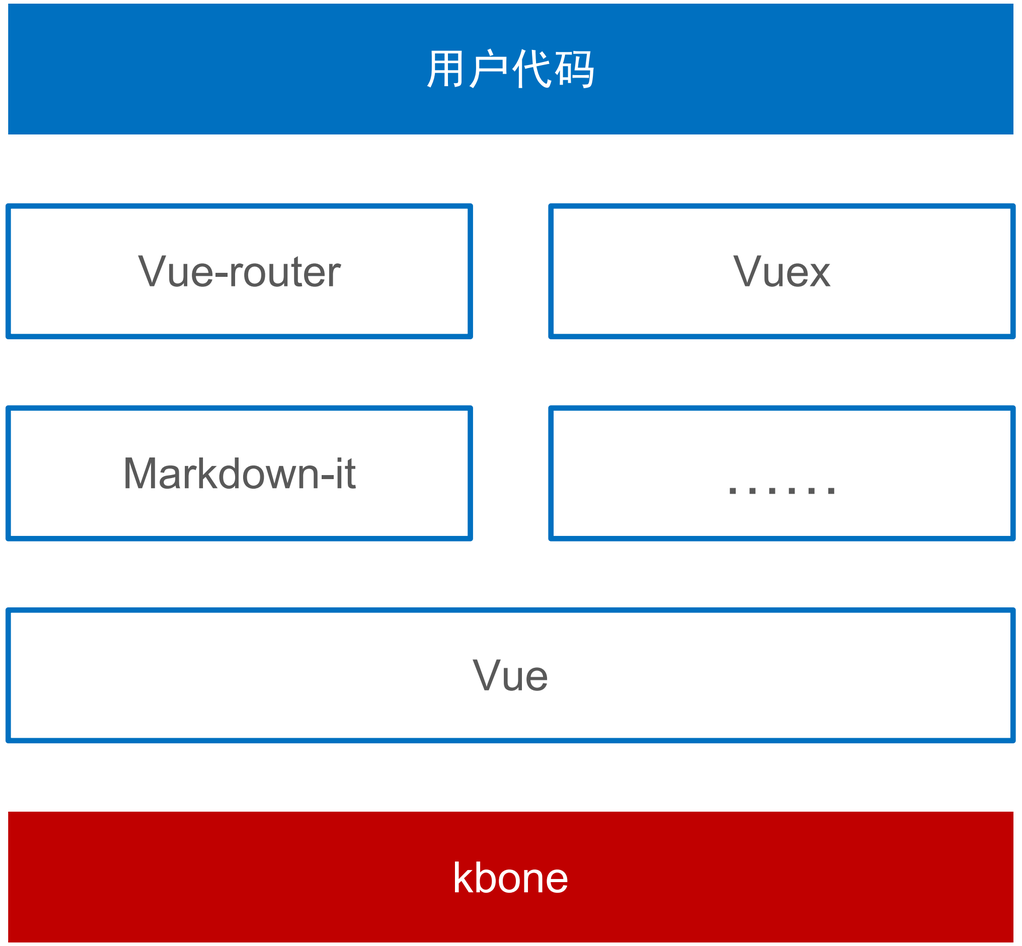
Part 3 应用

应用到微信开发社区



微信开放社区 Web 端

应用到微信开发社区



微信开放社区小程序

webpack 构建

+

mp-webpack-plugin 插件

用 kbone 仿造浏览器环境

实现得差不多时，产品来提需求了……

应用优化

希望小程序端的页面头部和 h5 端不同，不同端使用不同的交互设计

构建时注入环境变量，根据不同环境执行不同逻辑

```
plugins: [
    new webpack.DefinePlugin({
        'process.env.isMiniprogram': true,
    }),
```

应用优化

还希望可以支持小程序端的分享

支持使用小程序内置组件，实现小程序分享功能

```
<div v-if="isMiniprogram">
    <button @click="onCancel">取消</button>
    <wx-button open-type="share">分享</wx-button>
</div>
```

小程序端完成，想用手机预览一下看看……

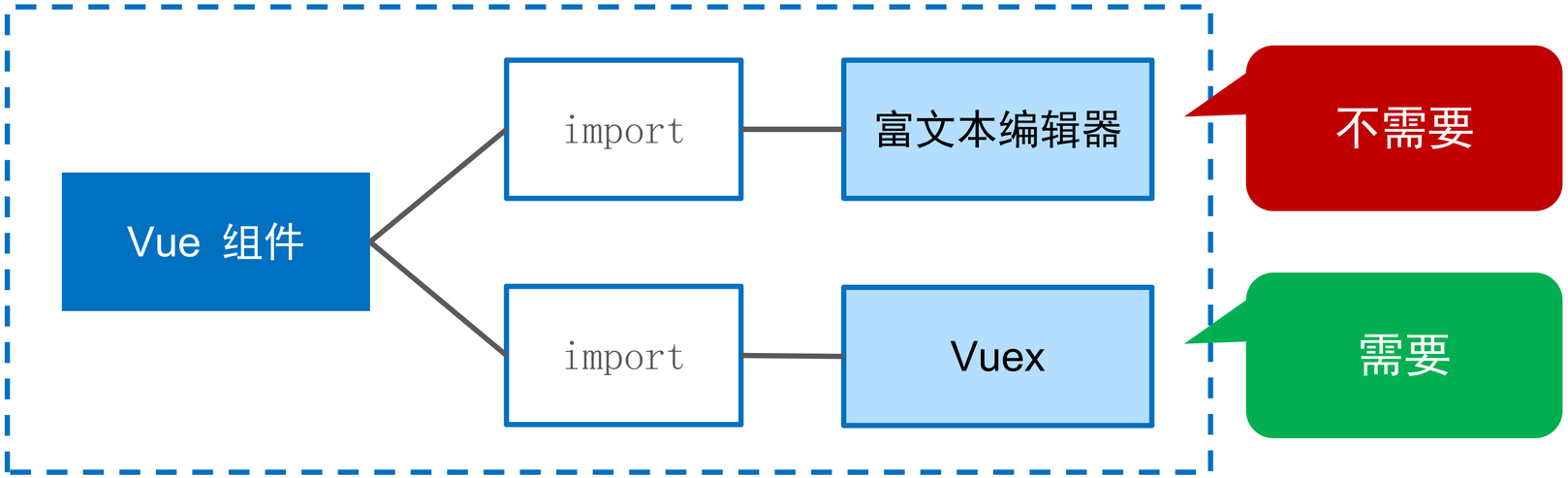
体积优化

代码体积太大，怎么办？

- 压缩混淆
- 代码分割、公共代码复用
- tree shaking
- 使用分包
- ……

体积优化

社区 Web 端是一套基于响应式设计的代码，包含很多
在移动端用不到的东西。



提供一个 webpack loader，
根据环境剔除不需要的代码

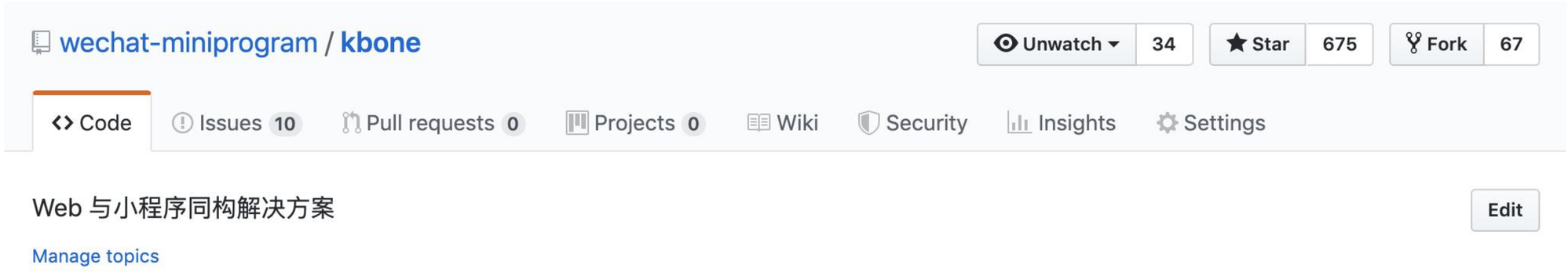
```
<script>
    import { mapState, mapGetters } from 'vuex';
    import Editor from 'reduce-loader!../Editor.vue';
```

效果展示

微信开放社区小程序渲染效果对比
（左为 h5 端，右为小程序端）：

Part 4 结语

工具链开源

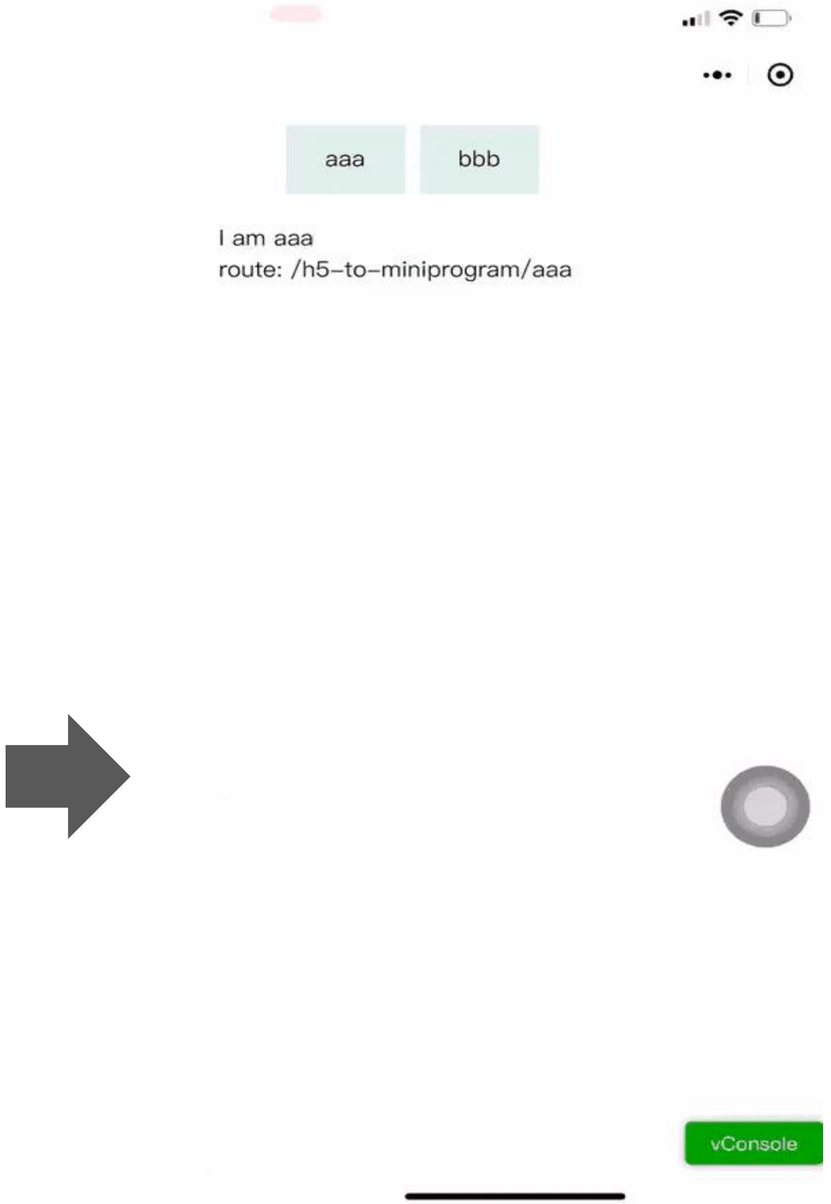


目前方案核心已稳定，周边工具和功能在逐步完善中，已有 50+ 小程序在接入中。

扩展

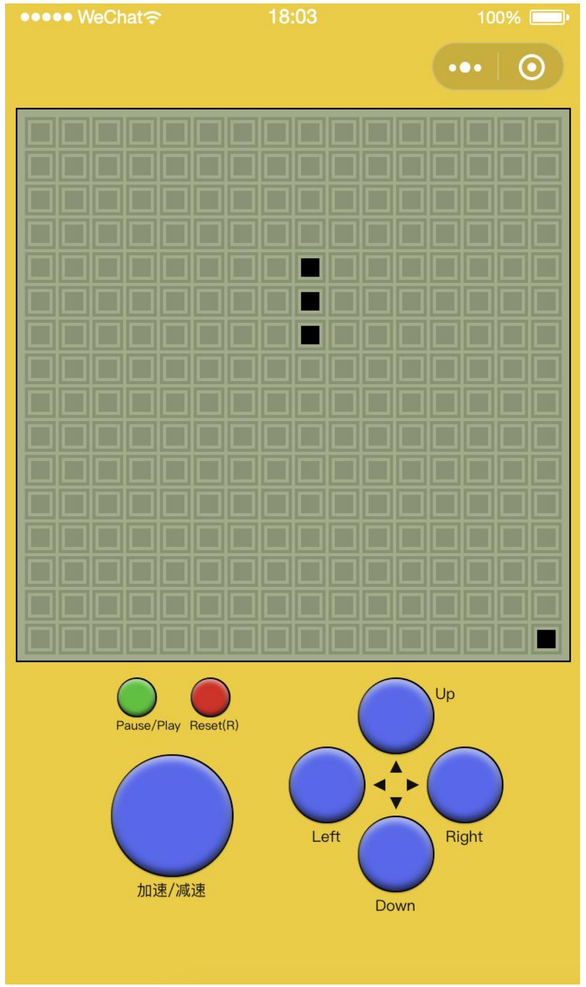
因为此方案是通过提供适配层的方式来实现的，所以可以支持扩展到其他框架，比如 React、Preact、Omi 等

```
import React, { Component } from 'react';
import { BrowserRouter as Router, Route, Link } from 'react-router-dom';

import AAA from './aaa.jsx'
import BBB from './bbb.jsx'

export default class App extends Component {
    render() {
        return (
            <Router>
                <div>
                    <ul className="tabbar">
                        <li><Link className="link" to="/h5-to-miniprogram/aaa">aaa</Link></li>
                        <li><Link className="link" to="/h5-to-miniprogram/bbb">bbb</Link></li>
                    </ul>
                    <Route path="/h5-to-miniprogram/aaa" component={AAA}></Route>
                    <Route path="/h5-to-miniprogram/bbb" component={BBB}></Route>
                </div>
            </Router>
        );
    }
}
```



扩展

一些基于 React、Preact、Omi 的例子：



总结

- 支持更全面的 Vue 特性和 Vue 工具链
- 支持常用的 Dom/Bom 接口调用
- 复用现有 Vue 相关代码，降低改造和维护成本
- 可扩展性强

WEB CONF
<腾 讯 前 端 技 术 大 会/>

<Thanks/>

主办方 Organizer Tencent TWeb

微信小程序团队招前端、后台，有意向者
可将简历发送至：junexie@tencent.com