中国软件开源创新大赛技术报告

作品类别：其它

作品名称：前端框架 Nodom

所属赛道：开源项目创新赛道自由组

组　　名：nodom

单　　位：西南科技大学计算机科学与技术学院

目 录

作品类别：其它

一、设计方案

在前端飞速发展的大环境下，项目的可维护性和扩展性成了主要问题。为了解决这些问题，Nodom 团队研发并开源一款拥有高自主知识产权的前端 MVVM 模式框架 Nodom。用于构建用户界面，Nodom 内置路由，提供数据管理功能，支持模块化、组件化开发以及渐进式开发。在不使用第三方工具的情况下可独立开发完整的单页应用。Nodom 针对开发者的需求,提供了一系列开源库，方便开发者。包括不限于：基于 Nodom 的组件库（NodomUI）、数据管理库（Kayaks）、编辑器 Vscode 辅助插件（Nodom）。配合 Nodom，开发者可高效实现项目开发。

1.1 虚拟 Dom

Nodom 采用虚拟 dom 来描述视图。虚拟 dom 是对真实 dom 进行的一个抽象，将真实的 dom 抽象成 JavaScript 对象来描述。虚拟 dom 的的优势在于以下几点：

（1）效率高，通过虚拟 dom，Nodom 使用了 diff 算法来计算出两次 dom 变更的最小差异，并且只渲染最小差异变化，同时 Nodom 使用调度器会将多次 dom 操作进行合并并且批量更新，从而提升性能。

（2）跨端开发，虚拟 dom 的另一个重要的好处是实现了跨端开发，由于 dom 被抽象成了对象，而不是真实的节点，通过该对象我们完全可以描述我们需要渲染的内容，所以我们可以通过虚拟 dom 渲染 native 页面，并且还能将虚拟 dom 应用在 node，实现服务端渲染。

1.2 模块化

Nodom 设计时，秉持模块化思想。Nodom 以模块为单位进行应用构建，一个应用由单个或多个模块组成。开发者创建拥有各自状态的模块，再由这些模块构成更加复杂的视图。模块逻辑使用贴合原生的 HTML 语法，对于开发者而言，学习成本低。结合模块化，开发者的开发效率将会大大提升。

1.3 单向数据流

在 Jquery 时代，前端业务基于事件驱动，对于简单的交互需求而言，这确实足够了，而且开发起来较为迅速。但业务一旦复杂，项目的可维护性和扩展性成了严重的问题。

而在 Nodom 中数据会从顶层自上而下的通过 Props 进行传递，在复杂的交互操作时，你需要做的，只是更新你的数据源。Nodom 会把你的数据从顶层模块逐层地传下去，Nodom 甚至会帮你优化刷新数据时对 DOM 节点的复用，所以开发者也无需关注页面渲染的效率问题。Nodom 使模块既有复用的独立性，又保有模块间联系便捷性。

二、实现方案

2.1 框架架构方案

框架主要包括核心模块、支持模块和周边生态。

核心模块包括 Model 模块、编译模块、Module 模块、渲染模块。

支撑模块包括存储模块、CSS 管理器模块、自定义元素模块、自定义元素管理模块、比较器模块、指令类型模块、指令管理器模块、异常处理模块、事件模块、事件管理器模块、表达式模块、全局缓存模块、Model 工厂模块、Module 工厂模块、入口模块、对象管理模快、路由模块、调度器模块、工具模块。

周边生态主要是样式组件库（NodomUI）、应用快速构建工具(CLI)、数据管理库（Kayaks）、以及编码辅助插件（代码编辑器 Vscode 插件 Nodom）。

框架整体架构如图 1 所示：

图 1 框架架构

2.1.1 核心模块

Module 模块： Nodom 以模块为单位进行应用构建，一个应用由单个或多个模块组成。开发者在模块定义时需要继承 Nodom 提供的模块基类 Module。为提升模块重用性，通过 template()方法返回字符串形式的模板代码，作为模块的视图描述。通过 data()方法返回模块所需的数据对象，Nodom 再对其做响应式处理，响应式处理后的数据对象，Nodom 称为 Model 对象，并存储在模块实例中。

Model 模块： 模块所需的数据对象，由 Nodom 对其做响应式处理，并存储在模块实例中，Model 作为模块数据的提供者，绑定到模块的数据模型都由 Model 管理。Model 是一个由 Proxy 代理的对象，Model 的数据来源有两个：模块实例的 data()函数返回的对象、父模块通过$data 方式传入的值。Model 会深层代理内部的 Object 类型数据。基于 Proxy，Nodom 可以实现数据劫持和数据监听，来做到数据改变时，页面的响应式更新渲染。

Render 模块： 渲染器模块，主要分为首次渲染和增量渲染。首次渲染是将视图渲染到开发者提供的容器之中，增量渲染是在数据发生改变时，对比新旧虚拟 DOM 来实现变化侦测，然后更新有差异的 DOM 节点，最终达到以最少操作真实 DOM 更新视图的目的。

Compiler 模块：对模块的模板代码进行编译操作，生成虚拟 DOM 树，及实现虚拟 DOM 的静态检测。

2.1.2 支撑模块

CSS 管理器模块：针对不同的 rule，处理方式不同。CSSStyleRule 进行保存和替换，同时 scopeInModule(模块作用域)有效。CSSImportRule 路径不重复添加，因为必须加在 stylerule 前面，所以需要记录最后的 import 索引号。

自定义元素模块：用于扩充定义，主要对抽象语法树中的对象进行前置处理。

自定义元素管理模块：用于添加和获取自定义元素类。

比较器模块：用于增量渲染时对比虚拟 DOM 来实现变化侦测，将新旧两颗 Dom 树进行高效对比。实现对真实 DOM 的最小操作。

指令类型模块：用于构造指令、执行指令。

指令管理器模块：用于管理指令，包括添加删除等。

异常处理模块：用于处理代码异常，和异常时，语言的国际化处理。

事件模块：事件分为自有事件和代理事件，用于包装处理 DOM 原生事件。

事件管理器模块：对模块的事件进行管理，包括绑定事件、保存事件配置、事件处理等。

表达式模块：主要用于处理函数串、编译表达式串、表达式计算，实现数据的绑定视图功能。

全局缓存模块：Nodom 提供了缓存功能，用户可以自由存储公共数据，以及读取 Nodom 内提供的各种数据。

Model 工厂模块：主要用于数据模型映射、监听数据、绑定 model 到 module、绑定 model 到多个 module、model 从 module 解绑。

Module 工厂模块：主要用于管理 Module 实例，实现 IOC 功能，如添加以及移除 Module 到工厂，还可以用于获取模块实例（通过类名）。

入口模块：整个框架的入口模块，包括启动调度器、渲染器、创建

路由、创建指令、注册模块、Ajax 请求封装等。

对象管理模快：管理模块实例的一些属性集，包括:指令集、表达式集、事件集、渲染树。

路由模块：Nodom 内置了路由功能，开发者设置路由配置项、添加子路由、路由跳转等可以配合构建单页应用，用于模块间的切换。

调度器模块：调度器用于每次空闲的待操作序列调度。

工具模块：内置基础工具服务库。

2.2 模块解决方案

NoDom 以模块为单位进行应用构建，一个应用由单个或多个模块组成。模块解决方案描绘了模块的工作流程，如图 2 所示：

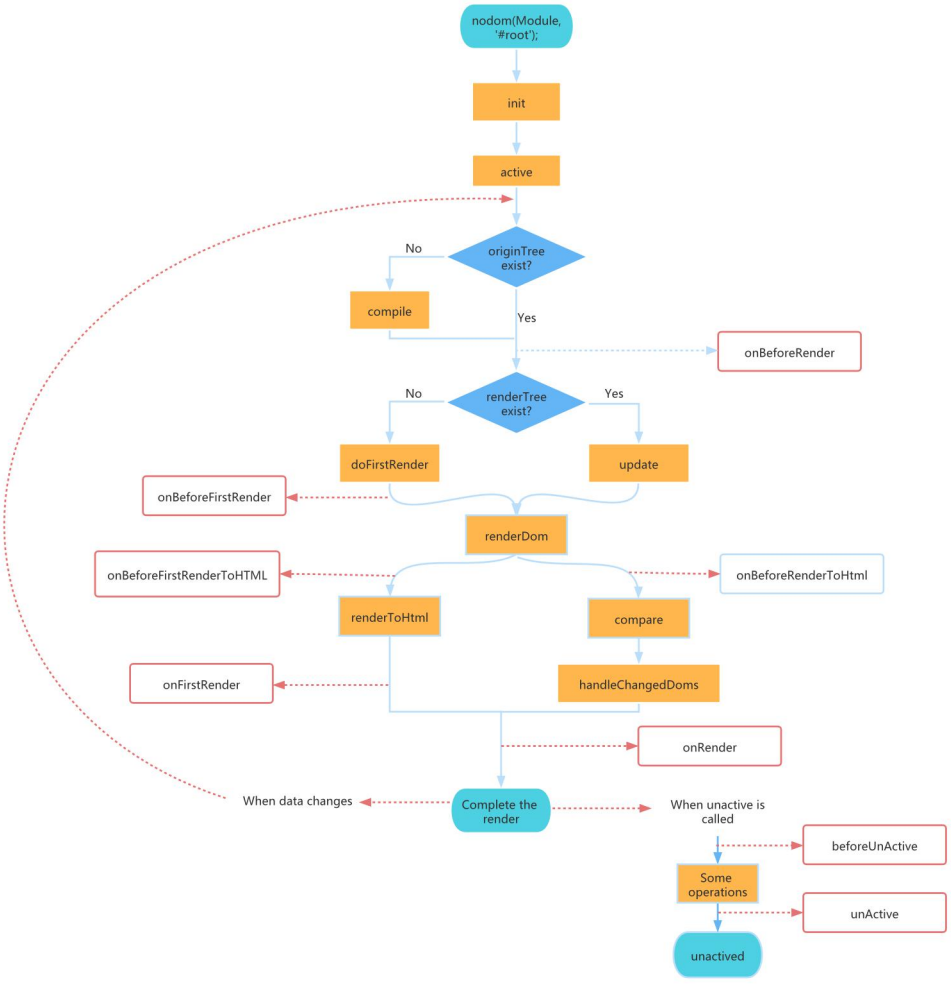

图 2 模块解决方案

（1）用户传入容器，Nodom 支持渐进式开发，框架内部会将传入的容器作为框架处理的入口以及作为渲染的容器，该容器完全交给 Nodom 托管。

（2）Nodom 对模块进行初始化以及激活。

（3）Nodom 判断模块 originTree 是否存在，不存在则由 Nodom 的编译器 Compiler 对模块的模板进行编译，存在则进入步骤（4）。

（4）根据 renderTree 存在与否进入增量渲染或首次渲染流程。

（5）渲染到浏览器。

（6）若模块搭载的 Model 数据发生改变，则由 Nodom 的调度器 Scheduler 进行调度，重复整个流程。

2.3 指令解决方案

为了增强 Dom 节点的表现能力，Nodom 根据实际业务开发的需求，实现了 13 款指令，指令以"x-"开头，以设置元素属性(attribute)的形式来使用，指令具有优先级，按照数字从小到大，数字越小，优先级越高，优先级高的指令优先执行。目前 NoDom 支持的指令如下：

指令名	指令优先级	指令功能
model	1	绑定数据
repeat	2	按照绑定的数组数据生成多个相同节点
recur	2	生成嵌套结构
if	5	条件判断
else	5	条件判断
elseif	5	条件判断

指令名	指令优先级	指令功能
endif	5	结束判断
show	5	显示视图
slot	5	插槽
module	8	加载模块
field	10	双向数据绑定
route	10	路由跳转
router	10	路由占位

Model 指令

model 指令用于给 view 绑定数据，数据采用层级关系，如：需要使用数据项 data1.data2.data3，可以直接使用 data1.data2.data3，也可以分 2 层设置分别设置 x-model='data1'，x-model='data2'，然后使用数据项 data3。model 指令改变了数据层级，NoDom 支持从根向下查找数据功能，当需要从根数据向下找数据项时，需要使用“$$”。

Repeat 指令

用于给按照绑定的数组数据生成多个 dom 节点，每个 dom 由指定的数据对象进行渲染。使用方式为 x-repeat={{item}}，其中 items 为数组对象。索引数据项为$index，为避免不必要的二次渲染,index 需要单独配置。

Recur 指令

用于生成树形节点，能够实现嵌套结构，在使用时，注意数据中的层次关系即可。recur 也可以通过使用 recur 元素来实现嵌套结构。

If/Elseif/Else/Endif 指令

if/else 指令用于条件渲染，当 if 指令条件为 true 时，则渲染该节点。当 if 指令条件为 false 时，则进行后续的 elseif 指令及 else 指令判断，如果某个节点判断条件为 true，则渲染该节点，最后通过 endif 指令结束上一个 if 条件判断。

Show 指令

用于显示或隐藏视图，如果指令对应的条件为 true，则显示该视图，否则隐藏。使用方式为 x-show='condition'。

Module 指令

用于表示该元素为一个模块容器，module 指令数据对应的模块会被渲染至该元素内。使用方式为 x-module='模块类名'，Nodom 会自动创建实例并将其渲染。

Field 指令

该指令用于实现输入类型元素，如 input、select、textarea 等输入元素与数据项之间的双向绑定。

绑定单选框 radio：多个 radio 的 x-field 值必须设置为同一个数据项，同时需要设置 value 属性，该属性与数据项可能选值保持一致。

绑定复选框 checkbox：除了设置 x-field 绑定数据项外，还需要设置 yes-value 和 no-value 两个属性，分别对应选中和未选中时所绑定数据项的值。

绑定 select：多个 option 选项可以使用 x-repeat 指令生成，同时使用 x-field 给 select 绑定初始数据即可。

绑定 textarea：直接使用 x-field 绑定数据项即可。

Router/Route 指令

router 指令用于设置模块路由渲染容器,每个模块中只能有一个 dom 设置 router 指令，如果设置多个，渲染时使用最后一个。router 指令对应的 dom 建议用块元素,通常配合 route 指令使用。

2.4 路由解决方案

Nodom 内置了路由功能，可以配合构建单页应用，用于模块间的切换。开发者需要做的是将模块映射到路由，并指定最终在哪个位置渲染它们。

（1）创建路由

Nodom 提供 createRoute 方法，用于注册路由。以 Object 配置的形式指定路由的路径、对应的模块、子路由等。

（2）嵌套路由

在实际应用中,通常由多层嵌套的模块组合而成。配置对象内 routes 属性，以数组的方式注册子路由，而同时每个配置对象内均可设置子路由，以此能够实现嵌套多层路由

（3）路由传值

开发者如果想要实现路由传值，只需在路径内以“：params”配置，Nodom 将通过路由传的值放入模块根 Model 的“$route”中。路由模块中可以通过“$route.data”获取 path 传入的值。

（4）单路由事件

每个路由可设置 onEnte 事件和 onLeave 事件，onEnter 事件在路由进入时执行，onLeave 事件在路由离开时执行，执行时传入第一个参数：当前模块的根 Model。

（5）全局路由事件

通过设置 Router.onDefaultEnter 和 Router.onDefaultLeave 事件作为全局路由事件，执行方式与单个路由事件执行方式相同，只是会作用于每个路由。

（6）默认路由

浏览器刷新时，会从服务器请求资源，nodom 路由在服务器没有匹配的资源，则会返回 404。通常的做法是：在服务器拦截资源请求，如果确认为路由，则做特殊处理。假设主应用所在页面是/web/index.html，当前路由对应路径为/webroute/member/center。刷新时会自动跳转到/member/center 路由。

三、运行效果/测试结果

前端 MVVM 模式框架的性能指标主要由模板编译和页面渲染决定，Nodom 在这两方面不断进行优化，并取得了一定的成果。

目前主流的同类型前端框架主要是 React 和 Vue，React 是国外最流行的前端框架，Vue 是国内最流行的前端框架。Nodom 初期与两者中性能指标更强的 Vue 进行了对比测试。对比详情如下：

3.1 编译效果

编译测试分为无属性节点编译以及带属性节点编译，在编译的性能上，Nodom 是 Vue 的 2 至 3 倍。具体测试数据如下图 3 所示：

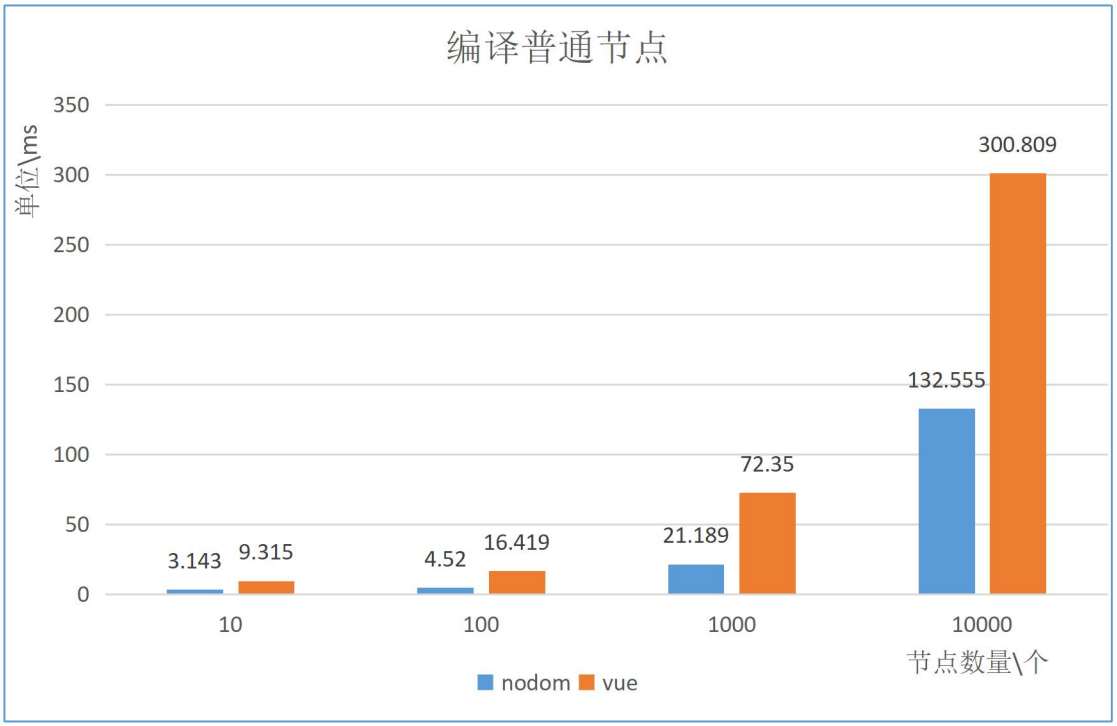

图 3 普通无属性节点编译对比

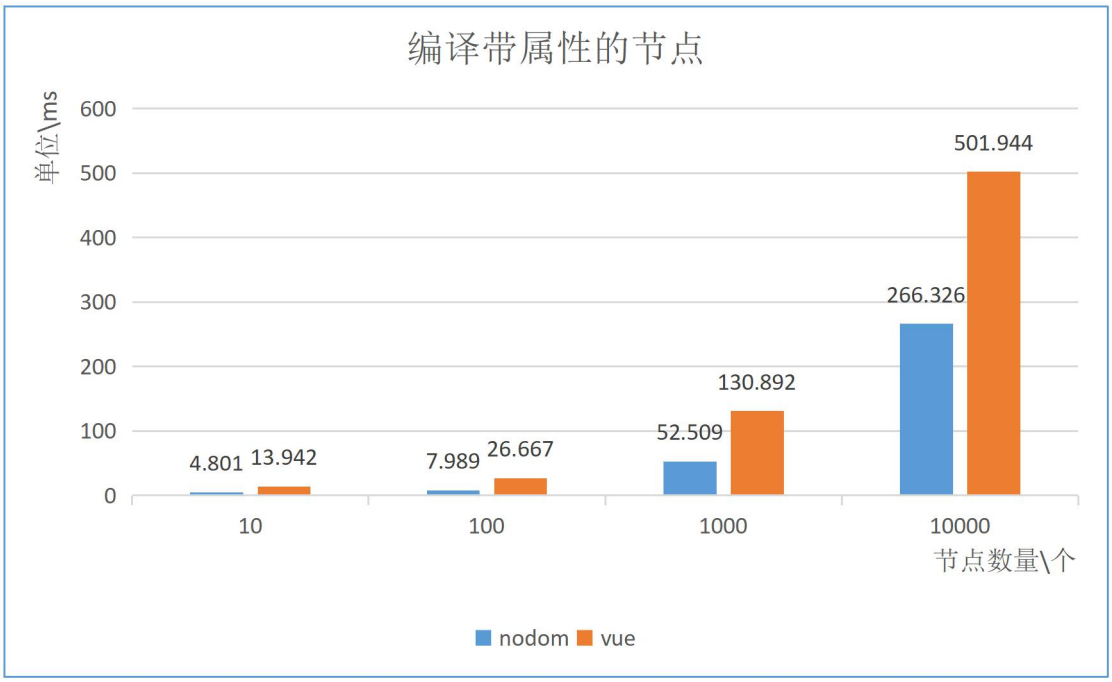

图 4 带属性节点编译对比

3.2 渲染效果

渲染主要分为首次渲染和增量渲染，增量渲染为响应式数据发生改变后，页面的部分渲染过程。针对渲染中经常出现的情况。Nodom 进行了多个测试。Nodom 与 Vue 的渲染具体测试数据如下：

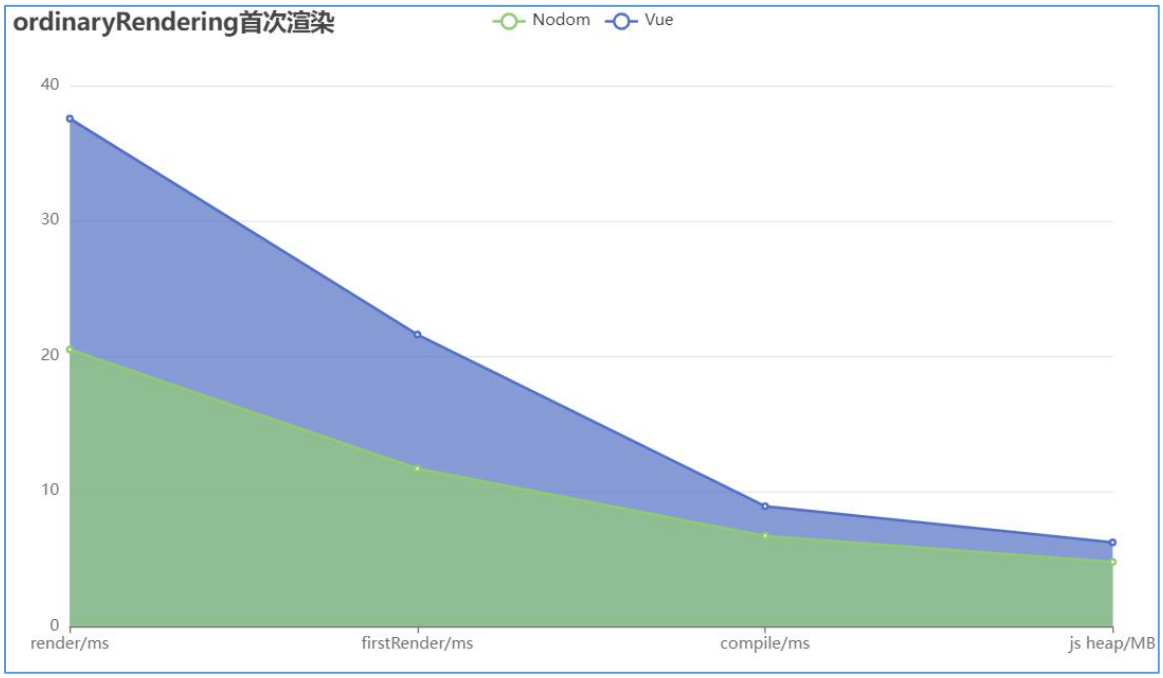

图 5 首次渲染对比

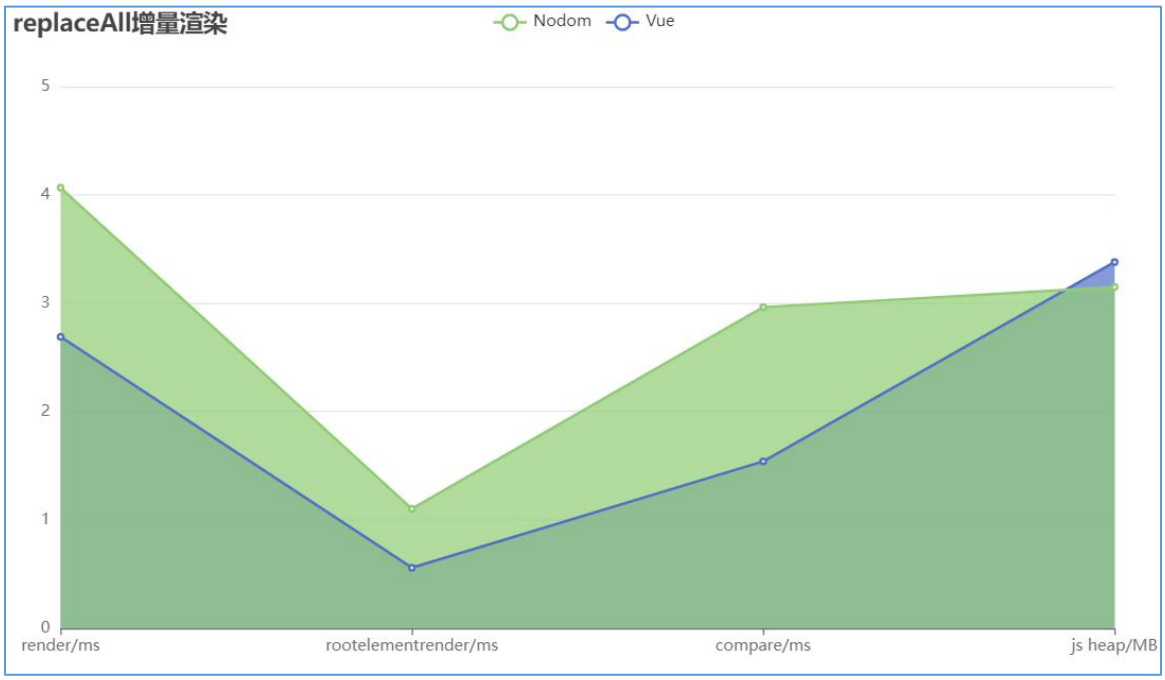

图 6 替换所有节点对比

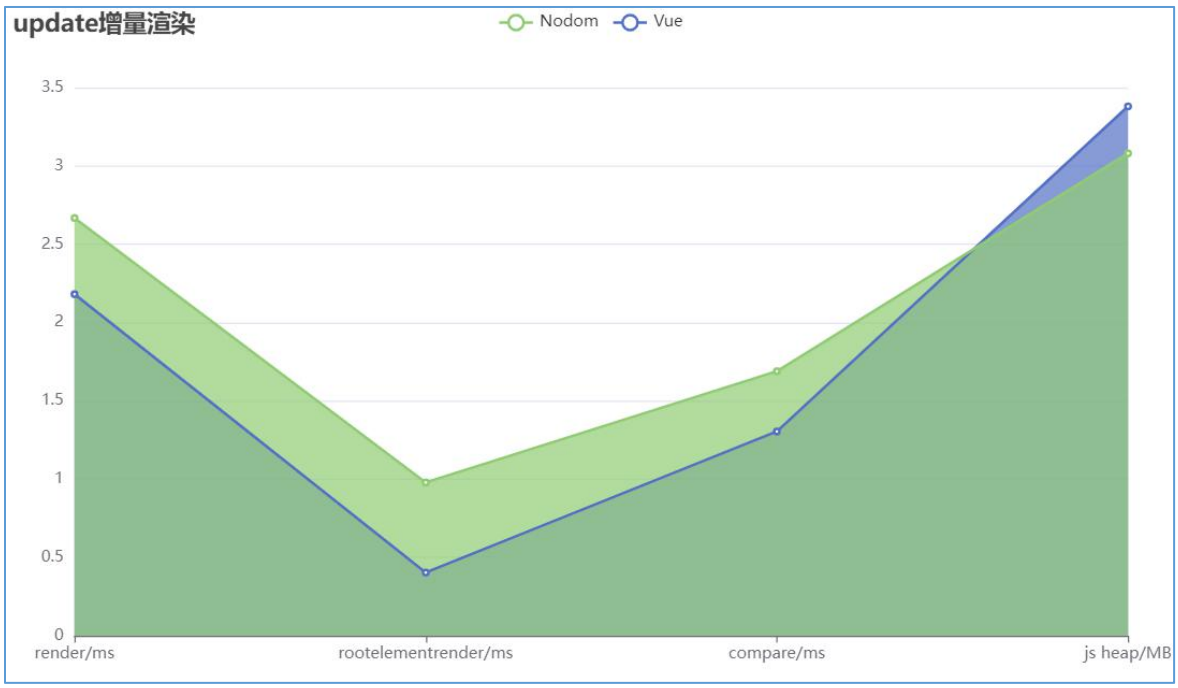

图 7 节点部分更新对比

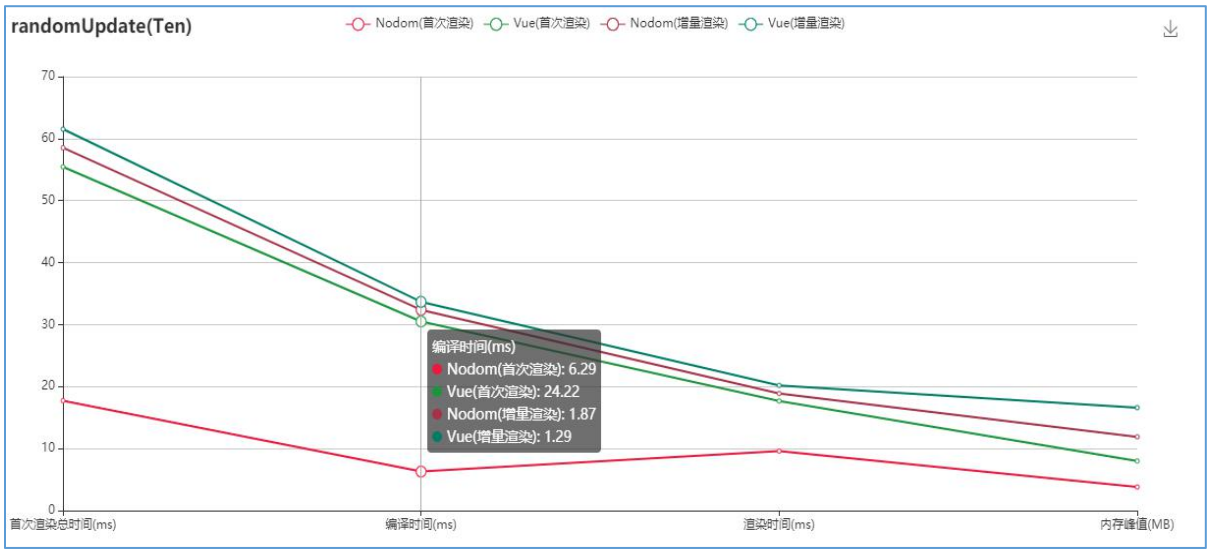

图 8 随机排序节点对比

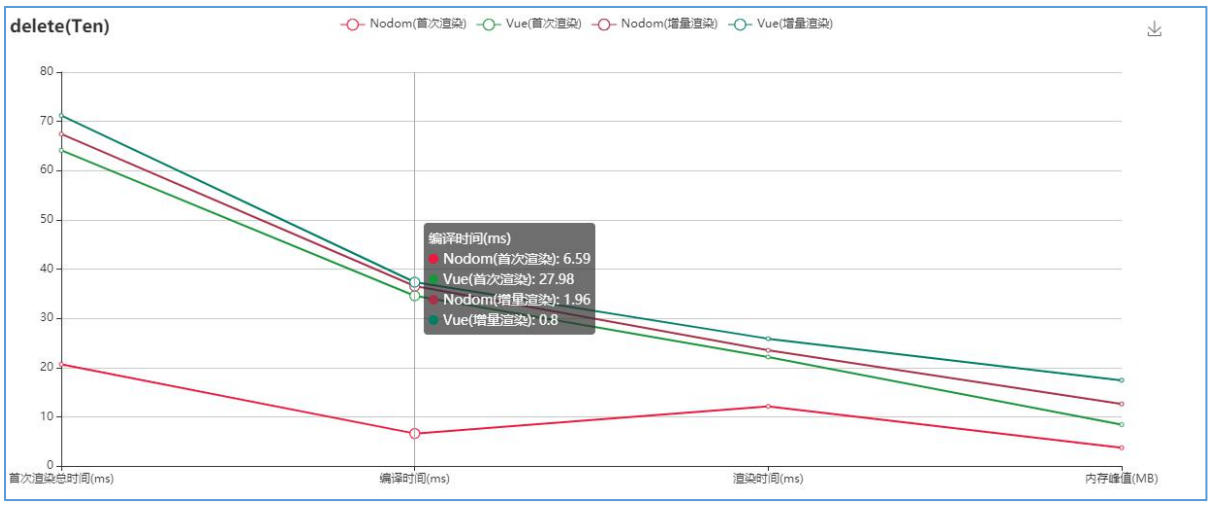

图 9 删除一个节点对比

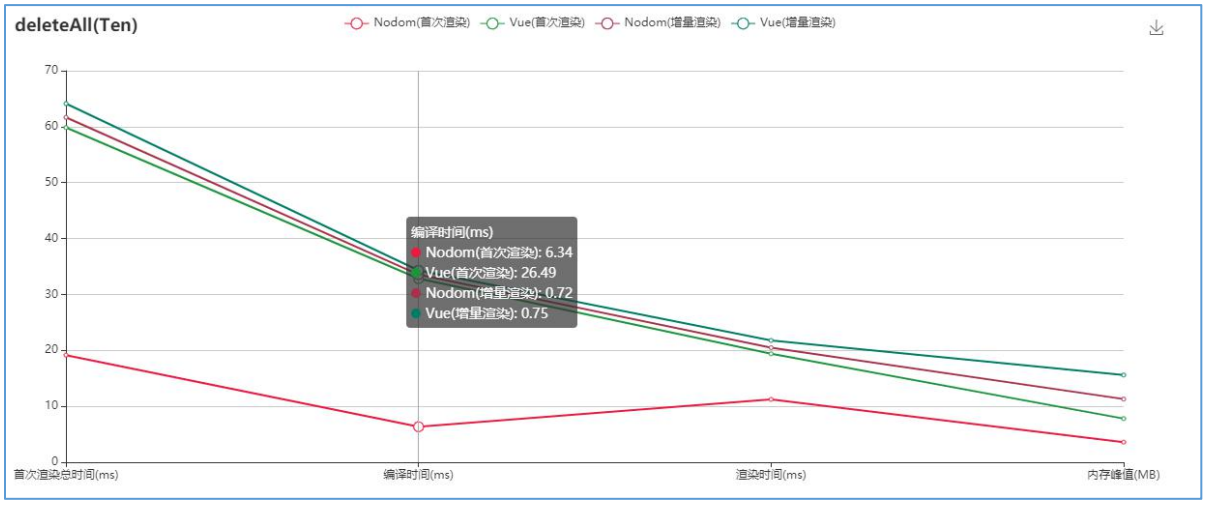

图 10 删除所有节点对比

四、特色创新

4.1 渲染优化策略

Nodom 采用 Diff 算法来实现增量渲染，Diff 算法用于对比虚拟 DOM 来实现变化侦测，然后更新有差异的 DOM 节点，最终达到以最少操作真实 DOM 更新视图的目的。Nodom 使用多种优化手段对传统 Diff 算法进行综合优化。在对比 Dom 树时，Nodom 创新的将对节点的操作进行抽象化处理，并且动态更改操作集。大幅度的提升了渲染性能。

4.2 编译优化策略

结合 Vue 的模板编译方法和 JSX 的模板语法特性。Nodom 对编译过程进行了优化。传统方案是先判断待匹配项是否是标签，然后再匹配标签名和标签值，这种标签提取策略非常低效，因此，编译优化的主要思想就是减少非必要的逻辑判断和匹配次数，只通过一次匹配抓取完整模板标签，同样的标签属性的处理也是为了避免重复的匹配过程，通过一次抓取完一个完整的属性串的方式避免了属性名和属性值的多次匹配，提高标签属性处理的效率，从而提高框架编译编译性能。

4.3 数据隔离机制

Nodom 独创数据隔离机制，将节点对应的响应式依赖对象划分至最细粒度。在框架处理中，根据节点的依赖层次，绑定最细粒度的数据。减少重复渲染以及错误渲染的可能，以多种手段增强了框架渲染性能。使节点既有自己的数据独立性，又保持了和其他数据的联系。

4.4 高自主知识产权与整体开源

Nodom 框架由团队自主开发，框架实现高自主知识产权率（核心模块自主开发率 100%，框架整体自主开发率 100%），避免在发展过程中受第三方框架的限制，实现可持续发展。框架已经整体开源并且持续更新，为前端开发者提供更多选择，繁荣国产开源社区生态，为进一步发展壮大奠定基础。