**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：开源项目创新赛道命题&自由组

单 位：西南科技大学计算机科学与

技术学院

时 间：2021年11月5日

1. **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，实验室团队提出构建一款拥有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。

1. **实现方案**

**2.1框架架构方案**

框架主要包括核心模块、支持模块和辅助工具。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；辅助工具主要是应用快速构建工具以及代码自动填充工具。如图1所示：

****

**图1框架架构**

**2.1.1核心模块**

**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，来做到数据改变时候的响应式更新渲染。

**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（建议使用模板字符串）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。

**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，增量渲染时在数据发生改变时，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。

**Compiler模块**：对标签的属性进行处理，将Module的模板代码HTML串编译为虚拟DOM。

**2.1.2支撑模块**

**存储模块**：Nodom提供了缓存功能，缓存空间是一个Object，以key-value的形式存储在内存中。

**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，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。

**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**：Nodom自定义异常，继承自Error类。

**事件模块**：事件分为自有事件和代理事件。

**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

**全局缓存模块**：保存到cache以及从cache读取。

**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，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请求等。

**对象管理模快**：管理对象。

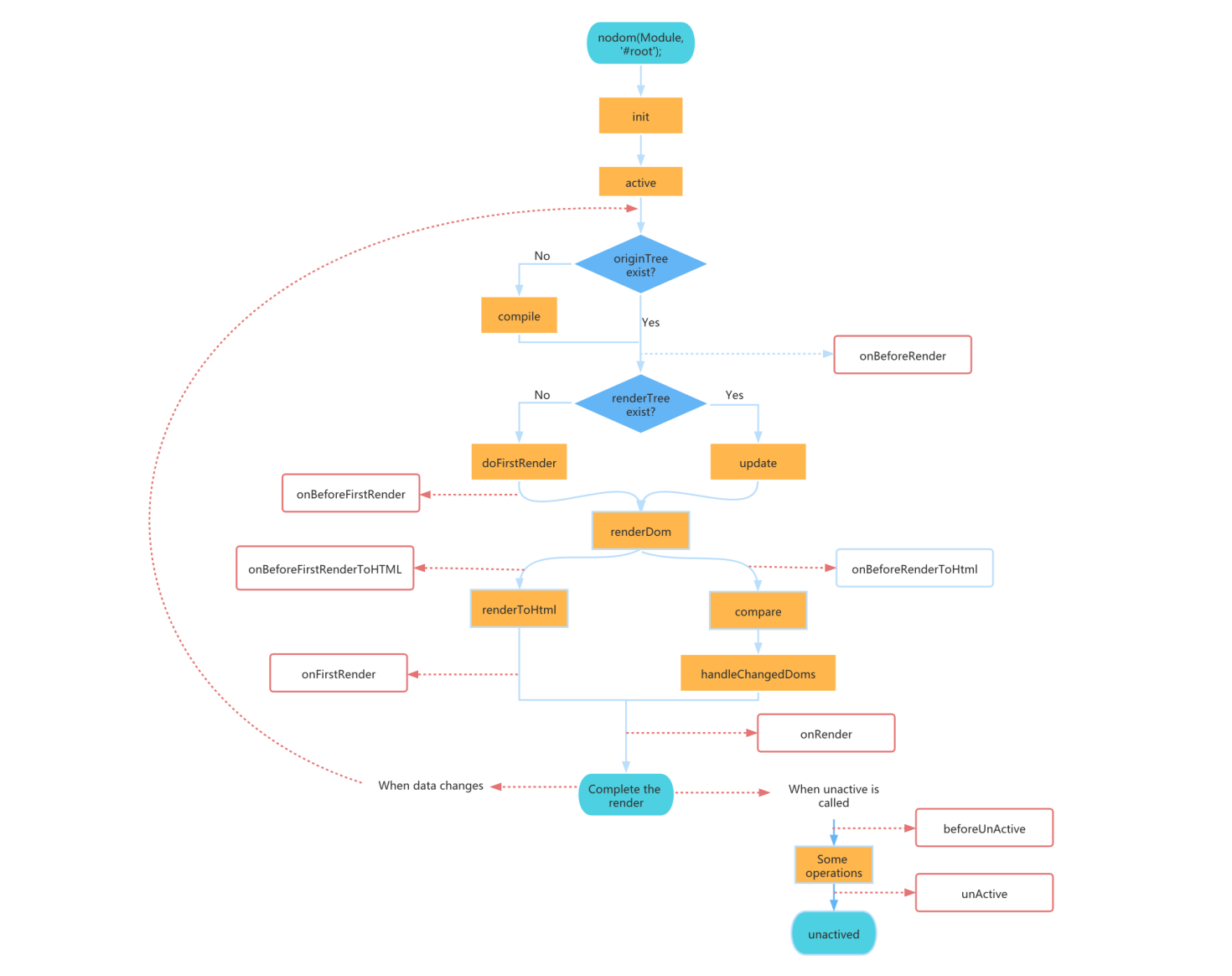
**路由模块**：Nodom内置了路由功能，开发者设置路由配置项、添加子路由、路由跳转等可以配合构建单页应用，用于模块间的切换。

**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** | **指令描述** |
| --- | --- | --- |
| 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 |
| 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。

1. **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

1. **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。

1. **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。

1. **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。

1. **运行效果/测试结果**

前端MVVM框架性能主要由编译和渲染决定，Nodom在这两方面不断进行优化，并取得了一定的成果。

目前主流的基于MVVM的前端框架主要是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运行效果与Vue的对比测试如下：

**3.1编译效果**

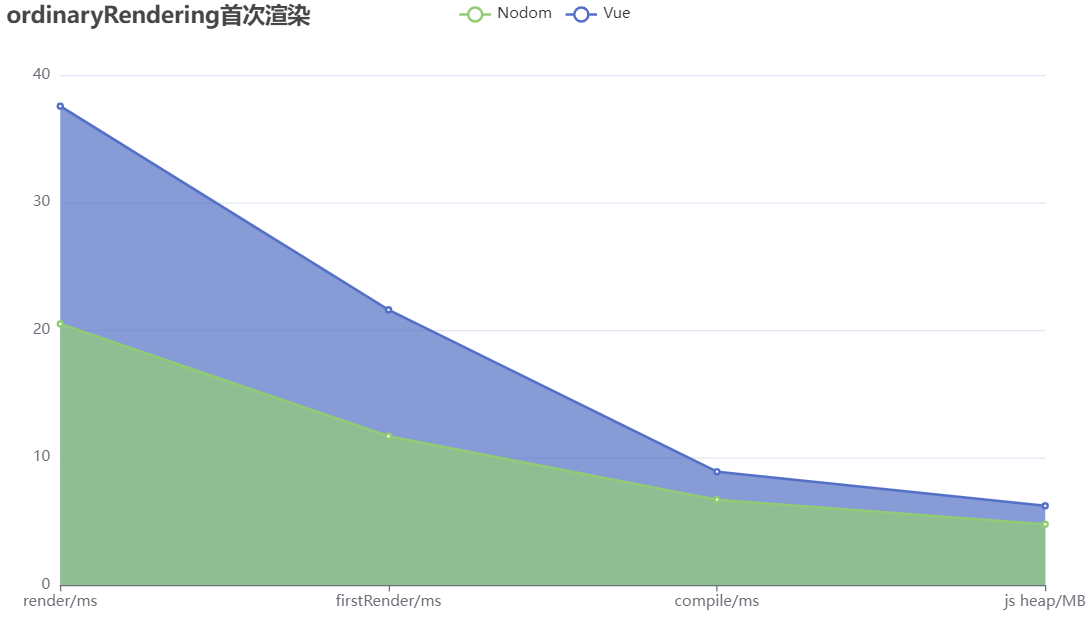
编译测试分为无属性节点编译以及带属性节点编译，具体测试数据如下图所示

**图3普通无属性节点编译对比**

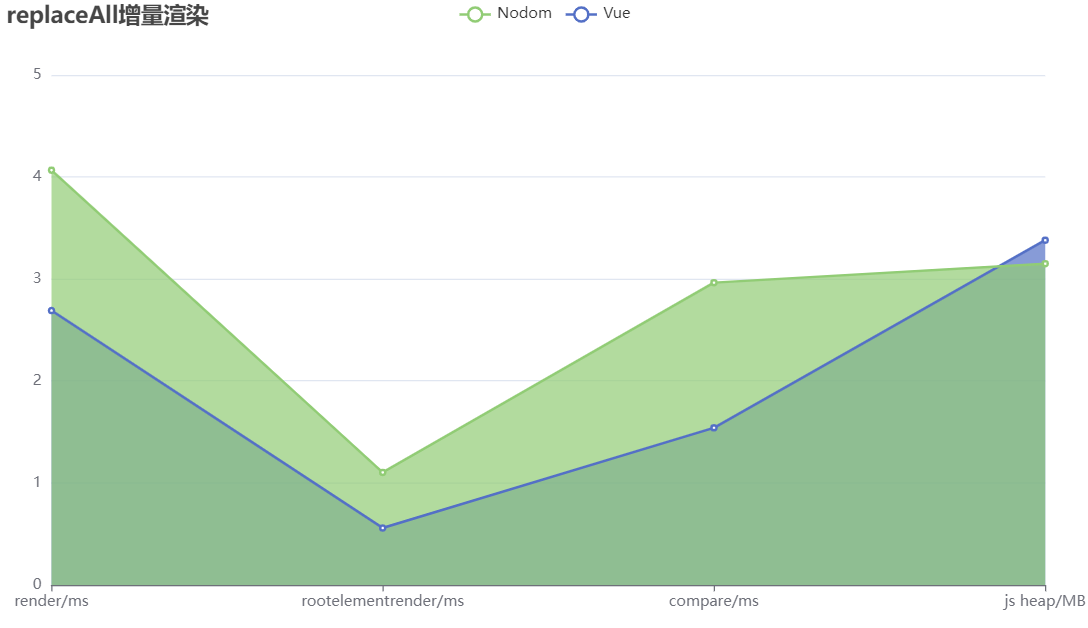
**图4带属性节点编译对比**

**3.2渲染效果**

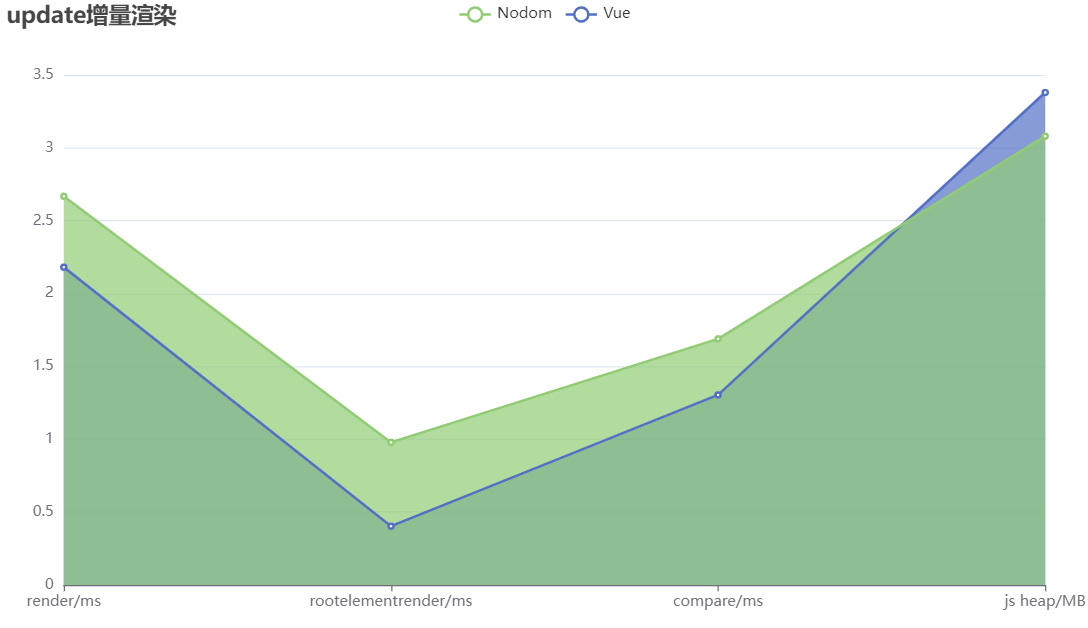
渲染主要分为首次渲染和增量渲染，增量渲染又可以细化为替换所有节点、部分更新、随机排序节点、删除—行、删除所有行，以下将会完成Nodom和Vue渲染测试对比。具体测试数据如下：

****

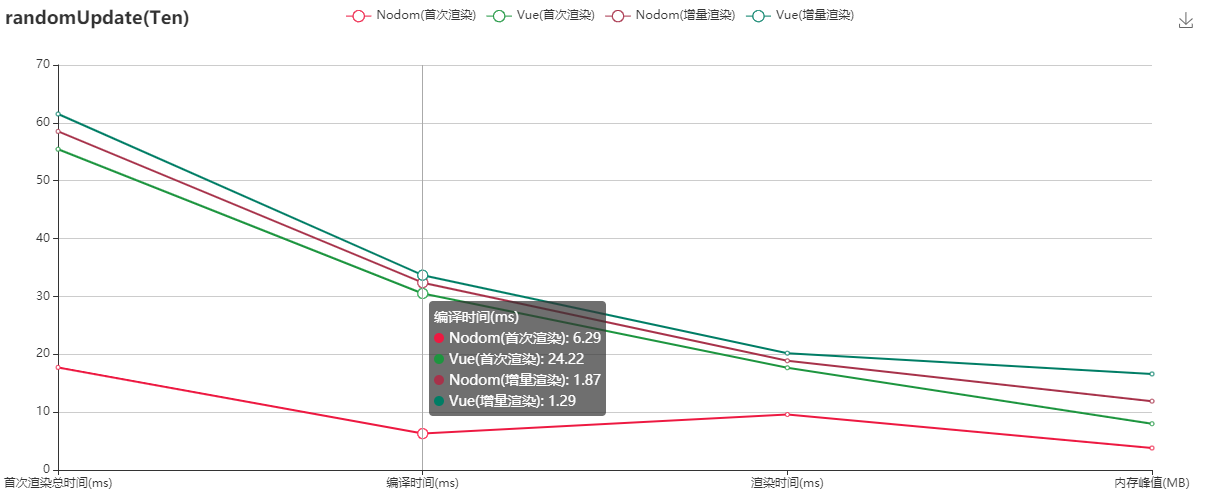
**图5首次渲染对比**

****

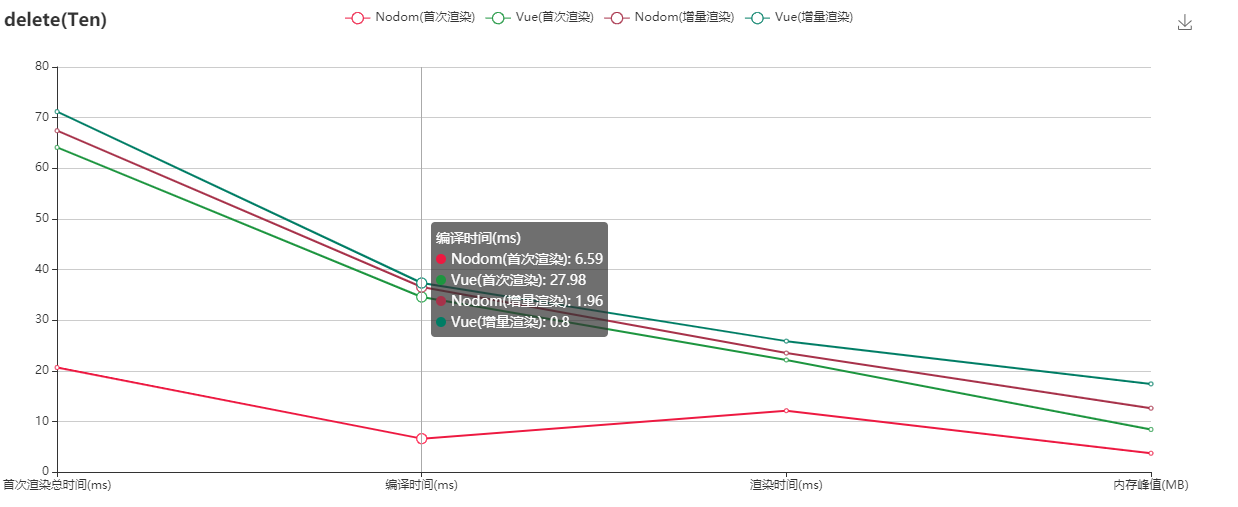
**图6替换所有节点对比**

****

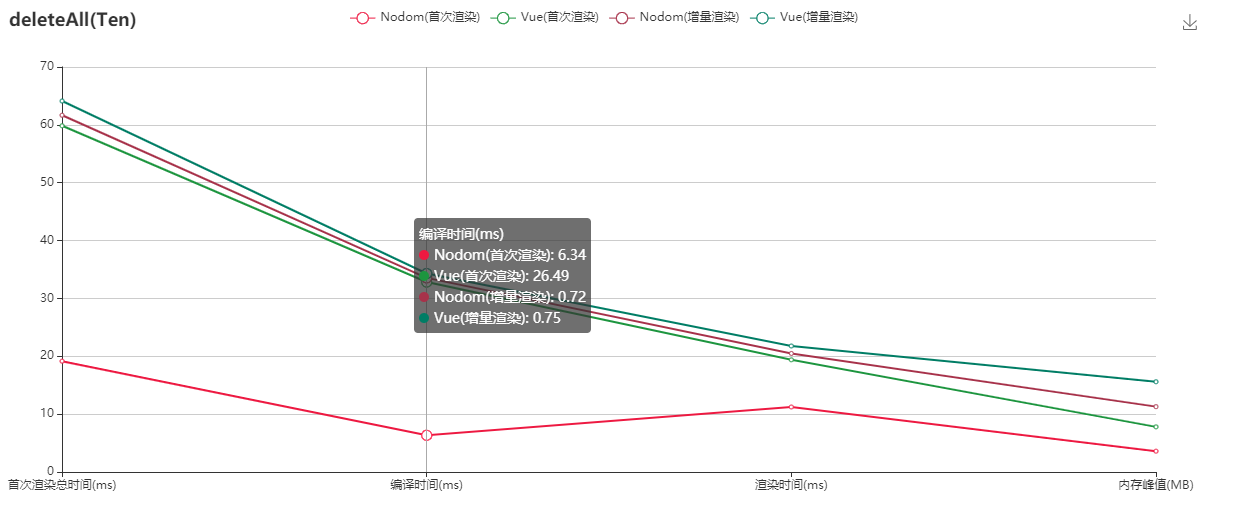
**图7节点部分更新对比**

****

**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**

Diff算法用于增量渲染时对比虚拟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Diff算法进行综合优化，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。

**4.2编译优化策略**

结合Vue的模板编译方法和JSX的模板语法特性对Nodom的编译过程进行优化。现有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，这钟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.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不低于90%），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（官网链接：[Nodom官网](http://www.nodom.cn)），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。