**Nodom技术报告**

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。

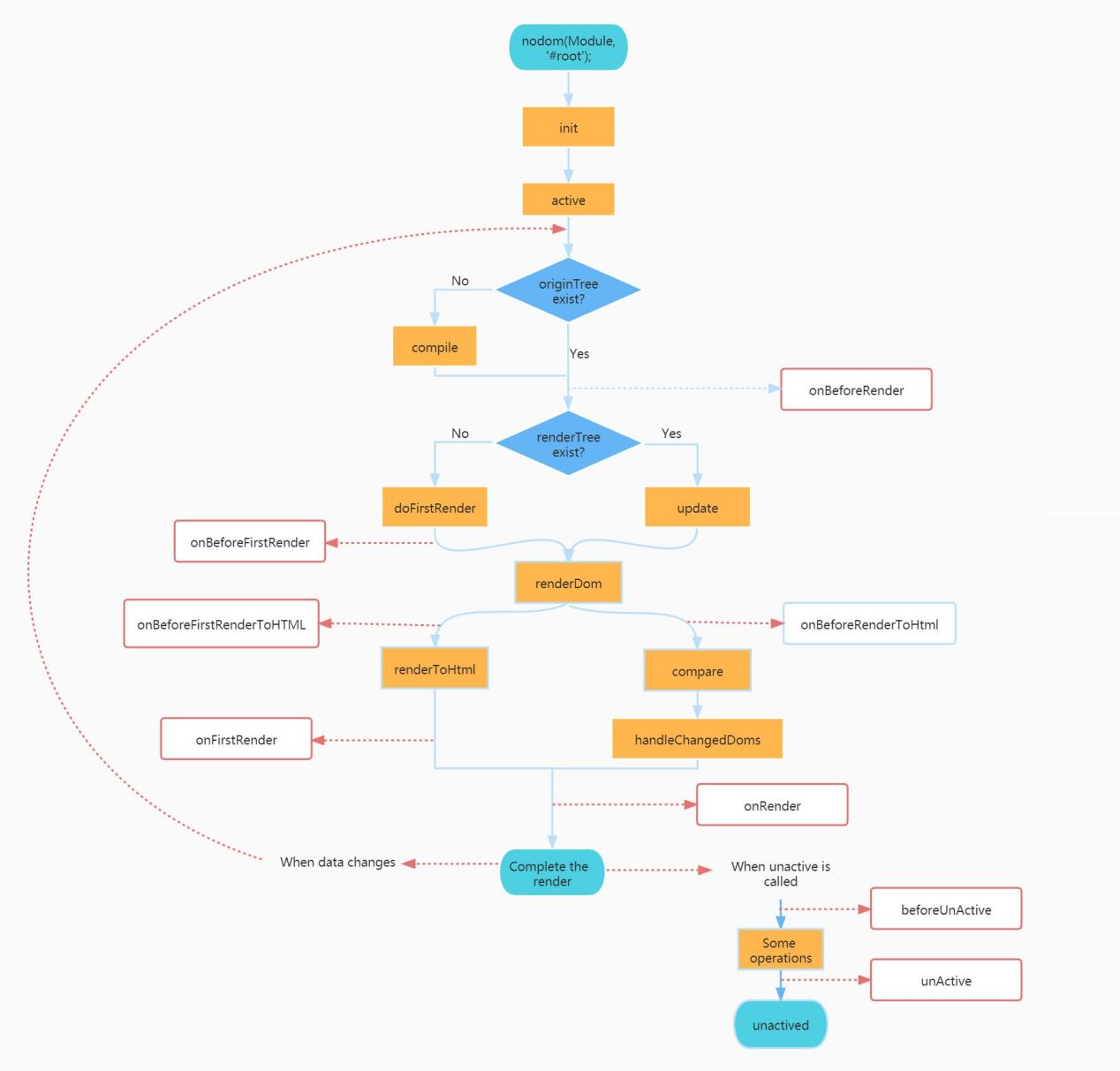
**路由模块**：Nodom内置了路由功能，开发者设置路由配置项、添加子路由、路由跳转等可以配合构建单页应用，用于模块间的切换。

**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。

**三、运行效果/测试结果**

**四、特色创新**