# 基于虚拟 DOM 的空间数据列表渲染方法研究与实现

张贺峰[1,2]，特日根[1,2,3]

(1.长光卫星技术有限公司,吉林 长春 130000;2.吉林省卫星遥感应用技术重点实验室,吉林 长春 130000;
3.中国科学院长春光学精密机械与物理研究所,吉林 长春 130000)

**摘　要:** 遥感数据是目前常用的空间数据,管理和查看遥感数据是各类地理信息系统的核心功能之一。其中,通过列表实现数据的分组管理以及通过时间轴实现数据的多时相展示是该类系统重要的遥感数据管理和查看方法。同时,通过浏览器对大量数据进行管理,会出现页面卡顿的现象。而通过多次执行 JavaScript 来修改 DOM,是造成浏览器渲染速度慢、页面卡顿的重要原因。本文通过 Vue.js 框架中的虚拟 DOM 技术,有效地减少了对空间数据列表的 DOM 操作。并通过对比实验,说明了当需要发生大量 DOM 操作时,Vue.js 框架具有更快的渲染速度,进而有效地解决了页面的卡顿问题。

**关键词:** 遥感数据; 地理信息系统; Vue.js 框架; JavaScript; 虚拟 DOM
**中图分类号:** P237　　**文献标识码:** A　　**文章编号:** 1672-5867(2021)06-0019-04

# Research and Implementation of Spatial Data List Rendering Method Based on Virtual DOM

ZHANG Hefeng[1,2], TE Rigen[1,2,3]

(1.Changguang Satellite Technology Co., Ltd., Changchun 130033, China;
2. Key Laboratory of Satellite Remote Sensing Application Technology of Jilin Province, Changchun 130033, China;
3. Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Science, Changchun 130033, China)

**Abstract:** Remote sensing data is currently commonly used spatial data. Managing and viewing remote sensing data is one of the core functions of various geographic information systems. Among them, group management of data through lists and multi-temporal display of data through time axis are important remote sensing data management and viewing methods of this type of system. At the same time, a large amount of data is managed through a browser, and a page freeze occurs. Modifying the DOM by executing JavaScript multiple times is an important reason for slow browser rendering and page freezes. This article effectively reduces the DOM operation on the spatial data list through the virtual DOM technology in the Vue.js framework. And through comparative experiments, it shows that when a large number of DOM operations need to occur, the Vue.js framework has faster rendering speed, and then effectively solves the problem of page freezes.

**Key words:** remote sensing data; GIS; Vue.js framework; JavaScript; virtual DOM

## 0　引　言

随着航天技术及传感器技术的发展,遥感图像在空间分辨率、时间分辨率、光谱分辨率上都有了较大的提升[1]。同时,遥感图像具有拍摄范围广、获取信息快、周期短等特点,是目前常用的对地观测手段。为了方便用

收稿日期: 2020-06-08
基金项目: 国家重点研发计划——基于空天地大数据的公共安全事件预警应用示范项目(2018YFB1004605);吉林省科技计划-科技创新中心项目——吉林省遥感信息技术应用创新基地(20180623058TC)资助
作者简介: 张贺峰(1993-),男,吉林长春人,助理工程师,硕士,2018 年毕业于吉林大学计算机软件与理论专业,主要从事遥感大数据领域的研究工作。
通讯作者: 特日根(1987-),男,蒙古族,吉林长春人,副研究员,博士,2015 年毕业于吉林大学计算机应用技术专业,主要从事遥感大数据领域的研究工作。

户查看自己的遥感数据，开发了一套以管理和展示数据为主的遥感数据展示平台。

使用平台的过程中，频繁地检索和查看数据会使浏览器产生大量的 DOM( Document Object Model，文档对象模型) 操作。DOM 是页面上数据和结构的一个树形表示，过多的 DOM 操作会占用浏览器资源，影响渲染速度，导致页面卡顿。因此，本文对虚拟 DOM 技术进行研究和分析，并通过引入 Vue.js 框架实现基于虚拟 DOM 的空间列表渲染过程。

## 1 Vue.js 框架介绍

Vue.js 是 2014 年国内出现的一款用于构建用户界面的轻量级渐进式框架，是目前三大主流前端框架之一。Vue.js 的核心部分体量小，使用者可以根据项目需要任意添加独立的功能模块或库[7]。该框架核心优势如下:

1) MVVM 模式

一直以来，MVC 模式是应用最为广泛的软件架构之一。按照功能，该模式将整个前端框架分为 Model( 模型) 、Controller( 控制器) 和 View( 视图) 。控制器负责模型和视图之间的数据处理和传输[8]。随着浏览器性能的提升，前端项目越来越大，数据结构越来越复杂，后台返回的数据可以直接在页面上显示的情况越来越少，所以在 MVC 模式的基础上，增加数据解析过程，便形成了 MVVM 模式，即 Model-View-ViewModel。

这种模式使后台数据适合页面展示的同时，也保证了通过页面修改的数据能够及时反馈给服务器。该模式可以有效地提升开发效率，同时也为数据双向绑定提供了可能。

2) 数据双向绑定

直观来说，数据双向绑定就是将页面元素的值和 JS 变量进行绑定[4]。JS 变量的改变会导致页面重新渲染，页面元素的改变也会导致 JS 变量的修改[5]。这样一来，就可以将操作 DOM 转变为修改 JS 变量，进而减少了对 DOM 的直接操作，可以有效减少代码量。面对复杂的页面交互，这种方式会让代码逻辑变得更加清晰，降低开发难度，同时更易于后期维护。

3) 组件化

组件化是 Vue.js 的核心思想之一。框架将前端项目映射为一棵组件树，每个页面可以是一个组件，页面上每个独立的可视、可交互区域也视为一个组件，并且每个组件都有用来处理数据的 ViewModel[6]。这种模式就为组件重用提供了支持。基于 Vue.js 开发的前端项目可以视为由多个组件组合而成，这些组件可以重复使用，可以单独维护，独立测试，有利于降低耦合度，提高开发质量[7]。

4) 虚拟 DOM

页面一般由多个元素组合嵌套构成，一个元素的改变会导致相关联的节点发生变化，页面越复杂，这种关联的变化就越多[8]。频繁地修改页面布局，使浏览器容易出现明显的卡顿[9]。Vue.js 采用了虚拟 DOM 技术。虚拟 DOM 是与真实 DOM 对应的一种树状数据结构，每一个节点都是一个 JS 对象，同时每个节点也都具有一系列属性，是真实 DOM 的一种抽象表示[10]。所有的 DOM 操作都会先在虚拟 DOM 上执行，并通过 DIFF 算法定位到实际需要更新的节点，最后在页面中进行局部渲染。

虚拟 DOM 有如下优势:

①虚拟 DOM 以 JS 对象为基础，不依赖正式平台环境，具有跨平台能力[11]。

②由于 JS 的执行速度远高于 DOM 操作，所以通过比较修改前后的虚拟 DOM，可以计算出真正需要更新的节点，最大限度地减少 DOM 操作，显著地提高了渲染效率。

③在大量的、频繁的数据更新下，虚拟 DOM 能够进行合理高效的更新。

## 2 DIFF 算法介绍

因为 DIFF 算法能够计算两个具有树型结构的数据差异，所以 Vue.js 引入了该算法处理具有树型结构的虚拟 DOM，并利用算法比较出虚拟 DOM 在修改前后的不同[12]。为了提升计算效率，Vue.js 去掉了 DIFF 算法中循环遍历每一级节点的操作，只保留同级比较过程。如图 1 所示，虚拟 DOM 的节点 C 发生了变化，传统 DIFF 算法会继续比较节点 C 和节点 G 的子节点，保留相同的子节点并修改子节点的指针，指向新生成的节点 G。而 Vue.js 实现的 DIFF 算法只保留同级比较过程，发现节点 C 改变后，不再比较节点 C 和节点 G 的子节点是否相同，在真实 DOM 树中，直接删除节点 C 及其子节点，然后重新生成节点 G、节点 D 和节点 E。通过这种策略，DIFF 算法的复杂度从 $O(n^3)$ 降到 $O(n)$[13]。同时，算法在判断两个节点是否相同时，并不要求属性完全相同，只要节点类型相同，就认为是相同的节点，从而最大限度地实现节点复用，减少 DOM 移动。

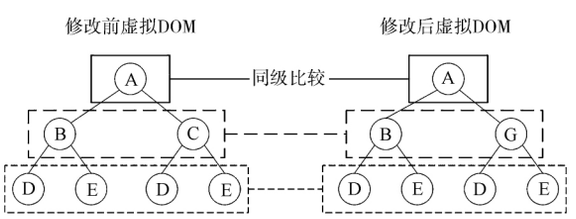


**图 1 Vue.js 中 DIFF 算法的对比方式**

**Fig.1 Comparison of DIFF algorithm in Vue.js**

算法的具体流程如下:

1) 利用 JS 对象结构表示 DOM 树，并用此树构建真正的 DOM 树，随后插到页面文档中[14]；

2) 当状态变更时，重新构造一棵新的对象树;

3) 对新的树和旧的树进行比较，记录两棵树的差异[15]；

4) 把所记录的差异应用到真正的 DOM 树上，完成视图更新。

## 3　实　验

为了实际验证虚拟 DOM 技术的有效性,本节将通过统计页面改变到渲染结束的所需时间来对比虚拟 DOM 和 JS 的渲染速度。实验环境见表 1。

表 1　实验环境

Tab.1　Experimental environment

| 环境 | 版本/类型 |
|---|---|
| 系统 | Windows10 64 位 |
| CPU | i7-8700 |
| 内存 | 16G |
| 浏览器版本 | 谷歌浏览器 80.0.3987.149(正式版本)(64 位) |
| 编译器版本 | IntelliJ IDEA2019 编译器 |
| JS 版本 | ECMAScript 5 |
| Vue.js 版本 | Vue 2.5.2 |
| 统计渲染耗时工具 | 谷歌浏览器 Performance 调试模块 |

在各类地理信息系统中,数据的分组展示以及列表切换是最常用的两个功能,也是 DOM 操作较多的两个功能。本文利用<ul>标签和一组<li>模拟类似的页面布局,当 li 节点发生增、删、改操作时,统计 Vue.js 和 JS 的渲染速度。

页面布局代码如下:

```
<div class="contain" id="contain">
<div class="item_left">
<div class="item_top">
<span id="title">start</span>
<button onclick="addDom();">添加节点</button>
<button onclick="changeDom();">修改节点</button>
<button onclick="deleteDom()">删除节点</button>
</div>
<div class="item_middle">
<ul id="uldom">
<!--添加节点的位置-->
<li>节点 1</li>
<li>节点 2</li>
<li>节点 3</li>
</ul>
</div>
<div class="item_bottom"></div>
</div>
<div class="item_right"></div>
</div>
```

设计以下 3 个实验,分别对 ul 节点的 li 子节点执行增、删、改操作,其中将 ul 节点记为"ul-target"。

### 3.1　增加节点数量

实验流程如下:

1) 在 ul-target 中初始化 50 个 li 节点,完成页面初始化;

2) 分别执行 JS 代码和 Vue.js 代码,实现向 ul-target 追加 $n$ 个 li 节点,记录页面的渲染时间,($n$=50,100,200,500,1 000);

3) 重复步骤 2),计算不同 $n$ 值下,执行 JS 代码和 Vue.js 代码的页面平均渲染时间。

实验结果见表 2。由此实验可知,随着新增节点数量的不断增加,Vue.js 的效率更高(如图 2 所示)。

表 2　增加节点耗时

Tab.2　Increase node time

| $n$(个) | JS 耗时(ms) | Vue.js 耗时(ms) |
|---|---|---|
| 50 | 7.67 | 1.44 |
| 100 | 16.68 | 2.2 |
| 200 | 36.78 | 3.76 |
| 500 | 131.22 | 8.9 |
| 1 000 | 393.45 | 17.66 |

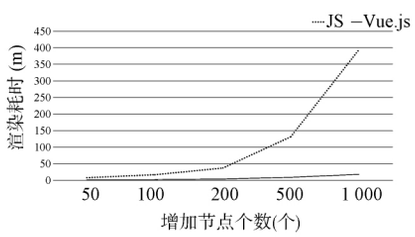


图 2　增加节点耗时折线图

Fig.2　Add node time-consuming line chart

### 3.2　删除节点

实验流程如下:

1) 在 ul-target 中初始化 1 000 个 li 节点,完成页面初始化;

2) 生成随机数组 list,表示需要删除的节点。数组长度为 $n$($n$=10,50,100,200,500),数组中任意一项 listi 满足 $0 \leqslant listi < 1\,000$ 且 $listi \in Z$;

3) 分别执行 JS 代码和 Vue.js 代码,根据 list 中保存的下标删除对应节点,并记录页面的渲染时间;

4) 重复步骤 3),计算不同 $n$ 值下,执行 JS 代码和 Vue.js 代码的页面平均渲染时间;

实验结果见表 3。由此实验可知,随着删除节点数量的增加,Vue.js 的效率更高(如图 3 所示)。

表 3 删除节点耗时

Tab.3 Deleting node time

| $n$(个) | JS 耗时(ms) | Vue.js 耗时(ms) |
|---|---|---|
| 10 | 8.04 | 10.61 |
| 50 | 30.86 | 13.8 |
| 100 | 58.63 | 13.19 |
| 200 | 106.08 | 12.04 |
| 500 | 222.45 | 7.72 |

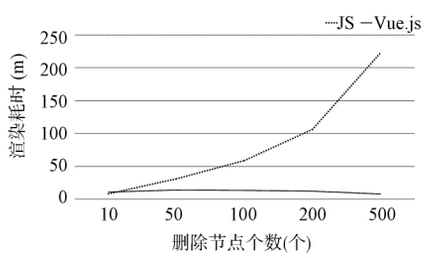


图 3 删除节点耗时折线图

Fig.3 Delete node time consuming line chart

### 3.3 修改节点

实验流程如下:

1) 在 ul-target 中初始化 1 000 个 li 节点,完成页面初始化;

2) 生成随机数组 list,表示需要修改的节点。数组长度为 $n$($n$=10,50,100,200,500),数组中任意一项 listi 满足 $0 \leqslant listi < 1000$ 且 $listi \in Z$;

3) 分别执行 JS 代码和 Vue.js 代码,根据 list 中保存的下标修改对应节点,并记录页面的渲染时间;

4) 重复步骤 3),计算不同 $n$ 值下,执行 JS 代码和 Vue.js 代码的页面平均渲染时间;

实验结果见表 4。由此实验可知,随着修改节点数量的增加,JS 的耗时远高于 Vue.js,但是在 $n$ 比较小时,JS 的效率更高,说明页面存在大量修改 DOM 操作时,Vue.js 效率更高(如图 4 所示)。

表 4 修改节点耗时

Tab.4 Time consuming to modify nodes

| $n$(个) | JS 耗时(ms) | Vue.js 耗时(ms) |
|---|---|---|
| 10 | 6.35 | 13.83 |
| 50 | 27.61 | 14.641 |
| 100 | 55.94 | 15.16 |
| 200 | 208.97 | 15.14 |
| 500 | 270.38 | 15.31 |

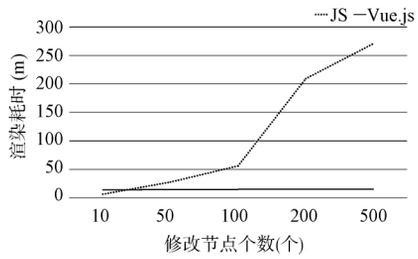


图 4 修改节点耗时折线图

Fig.4 Modify node time consuming line chart

以上 3 组对比实验说明,当页面出现较多的 DOM 操作时,Vue.js 的渲染速度更快。在修改操作较少时,JS 的渲染速度略高于 Vue.js。在地理信息平台的开发过程中,利用 Vue.js 框架对空间数据列表进行了重写,解决了列表切换时页面卡顿的问题。

## 4 结束语

空间数据列表是各类地理信息平台的核心模块[16]。该模块包含切换列表和显示隐藏数据等需要频繁调整 DOM 的操作。由于操作 DOM 会占用浏览器资源,影响渲染速度,所以随着数据增加,空间数据列表模块会变得卡顿。

本文选择虚拟 DOM 技术对空间数据列表模块进行优化。在页面发生频繁改动时,虚拟 DOM 能够高效地定位需要修改的节点并完成渲染。因此,虚拟 DOM 是一种提升渲染速度的有效手段。在实际开发过程中,Vue.js 框架解决了因 DOM 操作过多导致页面卡顿的问题。

通过 JS 和 Vue.js 的混合使用,笔者对框架有了新的理解。虚拟 DOM 是 Vue.js 框架的一种机制,在渲染速度方面的优势并不能说明 Vue.js 绝对优于 JS。在实际项目开发过程中,要根据业务需求选择适合的技术手段[17]。若只是一个简单的静态展示页,原生的 JS 更具优势;若项目涉及复杂的用户交互,则包括页面渲染速度、团队配合、代码质量、开发成本等方面 Vue.js 更具优势。

## 参考文献:

[1] 周碧莲.面向测绘应用的遥感小卫星发展趋势分析[J].中国新技术新产品,2019(24):117-118.

[2] 陈岩.轻量级响应式框架 Vue.js 应用分析[J].中国管理信息化,2018,21(3):181-183.

[3] 邓成,孙书会.MVVM 设计模式的前端应用[J].电脑知识与技术,2019,15(29):249-250.

[4] 戴志诚,程劲草.基于虚拟 DOM 的 Web 前端性能优化研究[J].计算机应用与软件,2017,34(12):21-25,31.

[5] 王鹏强.基于 Vue 的 MVVM 框架的研究与分析[J].电脑知识与技术,2019,15(11):97-98,100.

[6] 何焕春,杨怿.基于 MVVM 构架的 Web 前端框架研究[J].电脑知识与技术,2017,13(24):59-60.

(下转第 26 页)

为了从数据上更加精确地定位粗差，将高频信息 $d1$ 中的小波系数提取出来，求其标准差，并以 3 倍标准差为限值。若小波系数的绝对值超过 3 倍标准差即初步认定原始数据在该位置可能为粗差点，若小波系数小于 3 倍中误差，即认定该位置为正常值。经计算 $d1$ 中的标准差 $\sigma=0.014$ mm，$3\sigma=0.042$ mm，将 $d1$ 中每一个系数的绝对值与 $3\sigma$ 对比，探测结果见表 3。

表 3 粗差探测初步结果

Tab.3 Preliminary results of gross error detection

| 点号 | 118 | 119 | 320 | 321 | 476 | 477 | 530 | 531 | 768 | 769 |
|---|---|---|---|---|---|---|---|---|---|---|
| 突变绝对值 mm | 0.094 4 | 0.089 0 | 0.096 5 | 0.092 5 | 0.103 1 | 0.099 6 | 0.087 3 | 0.080 6 | 0.107 9 | 0.101 2 |
| 点号 | 910 | 911 | 1 098 | 1 099 | 1 158 | 1 159 | 1 560 | 1 561 | 1 788 | 1 789 |
| 突变绝对值 mm | 0.122 7 | 0.117 2 | 0.082 5 | 0.078 5 | 0.113 5 | 0.126 9 | 0.093 2 | 0.086 6 | 0.084 2 | 0.093 0 |

由表 3 可以看出，经过 db4 小波 4 尺度分解后，所有加入的粗差点均被成功探测出，但由于粗差点数据会对其后一个位置的数据产生影响，因此，每个粗差点后一个数据对应的 $d1$ 小波系数也超过了 ，在粗差判定上要考虑到这一点。通过实验可以看出，激光跟踪仪动态测量数据在经过小波变换后，我们可以明显地看到数据的高频部分与低频部分，了解不同频率内数据细节变化的特点。对于存在粗差的数据，小波系数存在明显的突变，可以迅速、直观地定位粗差。

## 4 结束语

本文针对激光跟踪仪动态测量数据粗差探测这一传统问题，结合已在其他工程领域被广泛使用的基于小波变换的粗差剔除方法，提出了适合于激光跟踪仪动态测量数据的小波变换粗差剔除方法，并结合实际实验数据对其效果进行了验证。实验结果表明：

1) 通过小波变换从频域和时域综合角度对激光跟踪仪动态测量数据进行处理，可以明显地观测到粗差的存在并进行定位，且该方法不需要获取观测数据先验信息，操作简单，提高了粗差探测工作的有效性。

2) 由于从时域和频域综合角度考察数据，该方法可以避免数据量过大时的处理效率问题，尤其适合于高采样率下的激光跟踪仪动态测量所得的海量数据。

需要说明的是，本文所采用数据是在较好的实验室条件下测量所得，其数据中并不真实存在粗差，为了验证该方法的普适性，后续还将利用该方法对真实的工程数据进行处理。

## 参考文献：


[1] 潘廷耀，范百兴，易旺民，等.大尺寸动态测量技术综述[J].测绘与空间地理信息，2015，38(8)：70-72，76.

[2] 冯小磊，华锡生，黄红女.观测值序列的粗差探测方法[J].水电自动化与大坝监测，2006，30(3)：56-59.

[3] 王慧琴.小波分析与应用[M].北京：北京邮电大学出版社，2011.

[4] 张勤，蒋廷臣，王秀萍.小波变换在变形监测中的应用研究[J].测绘工程，2005，14(1)：8-10.

[5] 范百兴.激光跟踪仪高精度坐标测量技术研究与实现[D].郑州：信息工程大学，2015.

[6] 杨凡，范百兴，李广云，等.激光跟踪仪动态测量精度测试[J].计量学报，2014，35(12)：119-122.

[7] 姚丽慧，高井祥，王坚.不同小波函数对粗差识别效果的比较[J].北京测绘，2011，25(3)：20-23.

[8] 方俊杰，鲍陈辰，方喆伟，等.小波分析在 GPS 动态观测中的应用研究[J].工程勘察，2019，47(11)：66-70.

[9] 王涛，田林亚，侯建梅，等.基于小波分析的变形监测数据去噪效果对比研究[J].勘察科学技术，2017(4)：15-18，36.

[10] 宋唐龙，肖付民，邓凯亮，等.基于小波变换的潮位粗差探测定位方法研究[J].海洋测绘，2019，39(4)：27-30.


[编辑：张 曦]

(上接第 22 页)


[7] 刘桃丽，曾志超.MVC 架构下网站的设计与实现[J].计算机技术与发展，2020，30(2)：188-191.

[8] 吕英华.渐进式 JavaScript 框架 Vue.js 的全家桶应用[J].电子技术与软件工程，2019(22)：39-40.

[9] 周伟，郑世珏.Web 前端工程化解决方案研究[J].信息技术，2018，42(8)：44-47.

[10] 李广宏.vue.js 前端应用技术分析[J].中国新通信，2019，21(20)：115.

[11] 李嘉，赵凯强，李长云.Web 前端开发技术的演化与 MVVM 设计模式研究[J].电脑知识与技术，2018，14(2)：221-222，251.

[12] 邓雯婷.基于 Vue.js 构建单页面 GIS 应用的方法研究[J].科技创新与应用，2018(14)：5-7，10.

[13] 吕品田.基于 vue.js 的共享空间平台的设计[J].数字通信世界，2018(3)：199，280.

[14] 李宇，刘彬.前后端分离框架在软件设计中的应用[J].无线互联科技，2018(17)：41-42.

[15] 朱二华.基于 Vue.js 的 Web 前端应用研究[J].科技与创新，2017(20)：119-121.

[16] 李桂华，陈畅曙，钟煜宏，等.基于 Cesium 的南中国海石油平台信息系统设计与开发[J].测绘与空间地理信息，2019，42(6)：47-50，55.

[17] 冯昌添，颜立志.数字新会地理空间框架建设与应用[J].测绘与空间地理信息，2019，42(5)：127-128，132，136


[编辑：任亚茹]