HAAGA-HELIA
University of Applied Sciences

# JavaScript 框架:角度 vs 反应 vs Vue

埃拉·萨克斯

学士论文

商业信息技术 2019

Haaga-Helia
University of Applied Sciences

# 摘要

日期

**作者**
埃拉·萨克斯

**学位课程**
商业信息技术

**报告/论文标题**
JavaScript 框架:角度 vs 反应 vs Vue

**页数和附录页数**
42 + 1

这项研究的目的是比较三种最流行的 JavaScript 框架的受欢迎程度、学习难度和性能，以便读者在决定学习哪个框架或在项目中使用哪个框架时做出 edu 式的决定。

为了了解框架的流行程度，收集并分析了三个最常用的软件开发合作平台的数据。React 被发现是这三个框架中最受欢迎的。

框架的学习曲线通过使用其官方技术文档进行比较，以了解其语法、架构、数据管理、生命周期以及使用第三方库的便利性。对于本研究的作者来说，Vue 似乎是最容易学习的。

为了比较框架的性能，在每个框架中构建并测试了一个简单的单页应用程序。Vue 被证明是表现最快的框架。

最后得出结论:React 是找工作时学习的最佳框架，Vue 是需要快速性能的小规模应用的最佳框架。Angular 很可能最适合更大更复杂的应用。尽管如此，在构建大型应用程序时，还需要进一步的研究来比较框架。

**关键字**
JavaScript 框架，Angular，React，Vue。

# 目录

# 1 介绍

本章将讲述论文背景、目标、要研究的 JavaScript 框架以及它们所比较的领域。

## 1.1 背景

大多数现代网站以某种形式使用 JavaScript，以使页面更具交互性并处理功能。传统的网页是多页应用程序，每当用户更改页面或其内容时，都会加载一个新的 HTML 文档。与单页应用程序(SPA)开发模型的更现代版本相比，这是一个相对较慢的选项，在该版本中，只有更改的部分从服务器中获取并更新。使用 SPA 模型降低了应用程序的加载速度，提高了用户体验。

单页应用程序越来越受欢迎，内置的框架也越来越受欢迎。框架规定了应用程序开发工作流程，减少了开发时间和可能的错误。每个框架都有不同的优点和缺点，因此，为软件项目选择正确的框架对于公司和希望为自己的武器库增加新技能的开发者来说都是一个战略决策。截至 2019 年，Angular、React 和 Vue JS 是一些最流行的 JavaScript 框架。

### 1.1.1 有角的

大约在 2008 年和 2009 年，Misko Hevery 和其他两名谷歌开发人员花了 6 个月的时间为谷歌构建一个内部反馈工具，却意识到这已经变得太难继续了，因为很难测试代码。Misko 和开发人员使用谷歌内部的 Java 框架，称为谷歌网络工具包。要用 HTML 写一个标签，必须用 Java 写，编译并转换成 HTML 和 JavaScript 才能在浏览器中显示。

米斯科向他的经理布拉德·格林提出挑战，他可以在两周内独自完成整个项目，使用他和他的朋友亚当·阿布伦斯作为辅助项目开发的技术。他花了 3 个星期才得到米斯科。布拉德对此印象深刻，并要求米斯科与希亚姆·塞沙里和伊戈尔·米纳尔一起使用 Angular 完成谷歌反馈工具。在那之后，Angular 在

GitHub，并向谷歌以外的人开放。它之所以被命名为 Angular，是因为它使用了带有尖括号的 HTML。(Gudelli 2017)。

当 Angular 的第二个版本发布时，框架发生了很大的变化，可以认为是一个新的框架。Angular 版本 2 和更高版本向后兼容，直到 Angular 2，但不兼容 Angular 1。为了避免混淆，Angular 1 现在被命名为 Angu- lar JS，Angular 2 和更高版本被命名为 Angular。Angular JS 是基于 JavaScript 的，而 Angular 是基于被称为 TypeScript 的 JavaScript 超集。(Dziwoki，2017 年)。

现代 Angular 是为所有平台设计的:网络、移动网络、原生移动和原生桌面。它是为网络速度而构建的，使用户能够控制可扩展性，同时满足巨大的数据需求。它得到了谷歌的支持，在全球拥有数百万用户。

### 1.1.2 反应

React 是一个用于构建用户界面的 JavaScript 库。React 是由脸书的软件工程师 Jordan Walke 在 2011 年开发的。它于 2013 年开放源代码，并从那时起获得了更多的内容。2015 年，面向移动设备的库 React Native 开源，2017 年，面向虚拟现实开发的库 React360 发布。React 目前由脸书和个人用户社区开发。脸书在其项目中使用 React，如脸书和 Instagram。(维基百科)。

React 旨在通过在数据更改时更容易更新视图来增强交互式用户界面开发。这是通过将视图分成更小的组件来完成的，这些组件可以被组合来创建复杂的用户界面。组件是用 JavaScript 而不是模板构建的，从而实现了轻松的数据流动。(反应一)。

### 1.1.3 某视频剪辑软件

Vue.js 是一个轻量级的 JavaScript 库，由尤雨溪创建。在创建 Vue 之前，埃文一直在谷歌和 Meteor 工作。在谷歌时，埃文觉得在某些用例中使用 Angular 太重了，他决定提取他喜欢 Angular 的部分，来创建他的轻量级库。2014 年，他在 GitHub 上与他人分享了自己的作品，这就是 Vue.js 的起步之路。

虽然埃文在与 Angular 合作时开始使用 Vue，但他认为 Vue 与 React 最相似，因为它的核心思想是数据绑定和组件，就像在 React 中一样。与 Re- act 相比，Vue 更强调用户体验，如果用户知道基本知识:HTML、JavaScript 和 CSS，就很容易上手。

2015 年在帕特里翁发起众筹活动后，埃文聚集了少量私人和企业赞助商。这为他带来了足够的现金流，目前他正在 Vue JS 全职工作。(克伦威尔，2017 年)。

## 1.2 目标

每个框架都是不同的，并且会有不同的表现。这项研究旨在指出每种框架可能的优点和缺点，使读者在选择时做出明智的决定。为了实现这一点，框架将分为三个不同的类别:受欢迎程度、学习曲线和性能。

## 1.3 流行

该框架的受欢迎程度将通过收集和分析与软件开发相关的主要云平台的数据来评估，如 GitHub、NPM 趋势和 Stack overflow。

## 1.4 学习曲线

将通过比较框架语法、架构、数据管理、生命周期以及使用第三方库的难易程度来评估框架学习曲线。这方面的信息将从每个框架的技术文档中收集。

## 1.5 表演

为了对框架的性能进行基准测试，在每个框架中构建具有相同业务逻辑的小型应用程序。然后，应用程序的生产构建性能将使用谷歌 Chromes 开发工具进行测量。应用程序代码将上传到 GitHub，工作应用程序将可以通过互联网访问。

## 1.6 超出范围

在这项研究中，JavaScript 框架(Angular、React、Vue)通过构建单页应用程序进行了比较。框架不是通过构建多页应用程序来比较的。框架作品:Angular、React 和 Vue 只是相互比较，而不是与普通的 JavaScript 进行比较。本文中“Angular”的名称是指 JavaScript 框架 Angular 2 及更高版本。

## 1.7 目标

每个框架构建不同，性能也不同。本研究的目的是指出每个框架可能的优点和缺点，使读者在选择时能够做出明智的决定。为此，框架将分为三个不同的类别:受欢迎程度、学习曲线和性能。

# 2 框架流行度

本章概述了用于分析框架的总体性的数据源和数据。数据源的选择是为了让全世界大量的软件开发人员在日常操作中使用这三个数据源。从他们那里收集的数据反映了实际情况，而不是依靠意见或炒作。

## 2.1 开源代码库

GitHub 将自己宣传为一个软件开发平台，有 3100 万开发人员使用它来托管和管理开源和商业项目。(GitHub 2019)。许多不同的开发人员出于不同的目的编写代码，这使得 GitHub 成为了解软件开发市场及其趋势的独特而可靠的来源。表 1 显示了关于 GitHub 上的 Angular、React 和 Vue 存储库的信息。

表 1。截至 2019 年 3 月 22 日 GitHub 存储库的框架统计(GitHub b；GitHub c；GitHub d .)

| | 有角的 | 反应 | 某视频剪辑软件 |
|---|---|---|---|
| GitHub 之星 | 46,572 | 125,734 | 133,479 |
| GitHub 叉子 | 12,286 | 22,848 | 18,990 |
| GitHub 问题 | 2,331 | 453 | 184 |

GitHub 用户可以用星星标记他们感兴趣的存储库，这样以后更容易找到，让 GitHub 知道用户有兴趣了解哪些主题。这也将显示对存储库维护者工作的赞赏。GitHub 重新定位会进行排名，并且可以根据星星的数量在搜索中进行排序。(GitHub e 2019)。

用户可以制作其他用户存储库的个人副本，并自由地对其进行更改，而不会影响原始项目。副本充当来自原始项目的不同路径，可以通过向源发出请求来更新。可以建议分叉或路径与原始存储库合并，使其对每个人都更好。从一个存储库中出来的分支的数量是一个很好的指标，表明有多少开发人员或团队试图作为原始作者的补充来改进项目。(GitHub f 2019)。

GitHub 有一个内置的称为问题的 bug 跟踪器。它使跟踪任务和与团队共享任务变得更加容易(GitHub g 2019)。问题的数量可以提供关于项目的规模和复杂性的一些想法，以及社区对它的支持。

## 2.2 国家预防机制趋势

节点包管理器是世界上最大的软件注册中心。它允许开源开发者共享和借用 JavaScript 包。它也被许多组织用来管理私人项目(NPM。)NPM 趋势是一个提供通过 NPM 下载的 JavaScript 包数量信息的网站。图 1 显示了过去两年中下载的 Angular、React 和 Vue 包的图表。

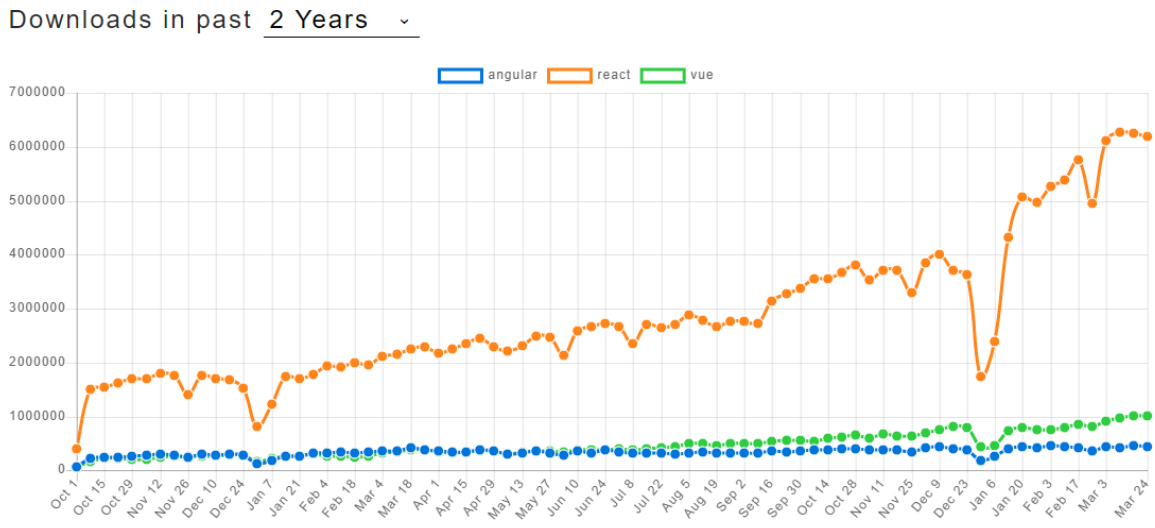


图 1。从 2017 年到 2019 年，Angular、React 和 Vue 库通过 NPM 下载(Npm 趋势)

## 2.3 堆栈溢出

Stack Overflow 是一个在线平台，开发者可以在这里提问和回答关于编程的问题。根据他们的主页，每月有超过 5000 万人访问他们的 Stack Overflow(Stack Overflow a)。

这个由开发人员和技术爱好者组成的庞大用户群，讨论不同的编程语言，使它成为了解全球开发社区正在使用和谈论的内容的绝佳场所。

图 2 是一个图表，显示了过去十年中关于堆栈溢出的角度问题、重复问题或 Vue 问题所占的百分比。

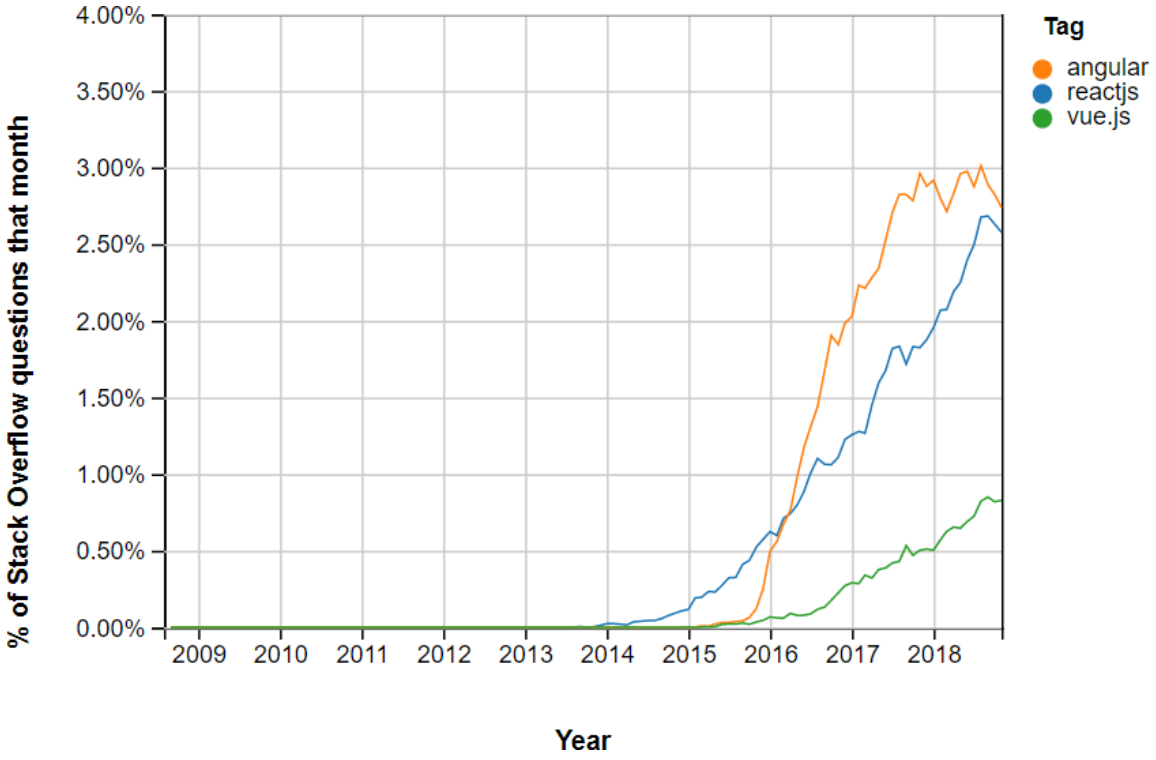


图 2。2009 年至 2019 年关于堆栈溢出(堆栈溢出 b)的 Angular、React 和 Vue 问题的百分比

# 3 框架学习曲线

本章提供了每个框架的高级技术概述。本章的信息是从每个框架的文档中收集的。因为这些都是 JavaScript 框架，所以它们的结构和工作流程都很相似。这就是为什么可以收集关于五个相同主题的每个框架的信息:语法、架构、数据管理、生命周期和第三方库。

## 3.1 有角的

Agular 的主页 Angular.io 有大量解释框架的文档。它有教程来让新的开发人员达到这个水平，它有文档来为更有经验的 Angular 开发人员提供更深入的主题。

### 3.1.1 句法

Angular 是用 HTML 和 TypeScript 编写的(Angular b)。TypeScript 是由微软开发和维护的开源编程语言，它为 JavaScript 添加了静态键入选项，并编译成 JavaScript(维基百科 b)。

Angular 模板语法是 HTML，几乎所有的 HTML 都是有效的 Angular 语法，除了< script >标记，Angular 会忽略它以避免脚本注入攻击。Angular 通过在 HTML 中提供一些 JavaScript 函数表达式来扩展常规的 HTML。双花括号用于将 JavaScript 内容呈现到视图中(Angular c)。

### 3.1.2 体系结构

角度应用程序由模块组成，模块是专用于应用程序域、工作流或一组密切相关的功能的代码块的容器。NgMod 规则为它们的内容提供编译上下文。它们可以包含组件、服务提供者和其他代码文件。他们可以从其他模块导出功能，也可以向其他模块导入功能。

每个 Angular 应用程序都有一个根模块，通常命名为 AppModule，它提供启动应用程序的引导机制。根模块可以包含无限多的子模块。将代码组织成不同的功能模块有助于管理复杂应用程序的开发。

Angular 部分利用了模型视图控制器的概念，将组件作为控制器，将模板作为视图。组件控制表示屏幕补片的视图(角度 e)。一个组件可以包含一个视图层次结构，由不同模块的视图组成。在图 3 中可以看到它的图示。，其中一个根组件属于模块 A，它的一些子组件和孙组件是从模块 b 导入的

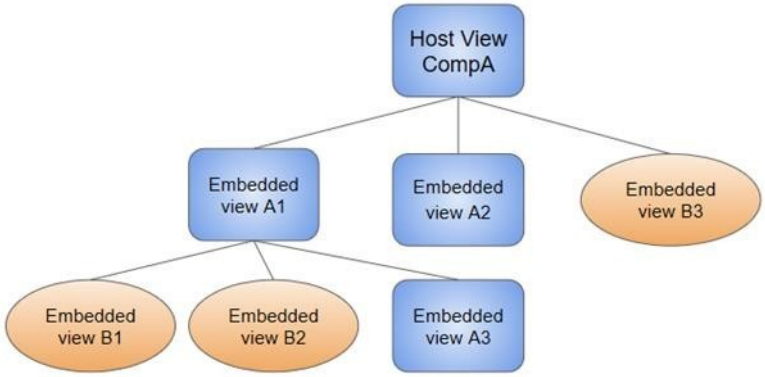


图 3。应用程序视图层次结构(角度 f)

### 3.1.3 数据管理

Angular 支持双向数据绑定，这是一种协调模板部分和组件部分的机制。数据从具有属性绑定的组件流入模板，并从模板流回具有事件绑定的组件，如图 4 所示。

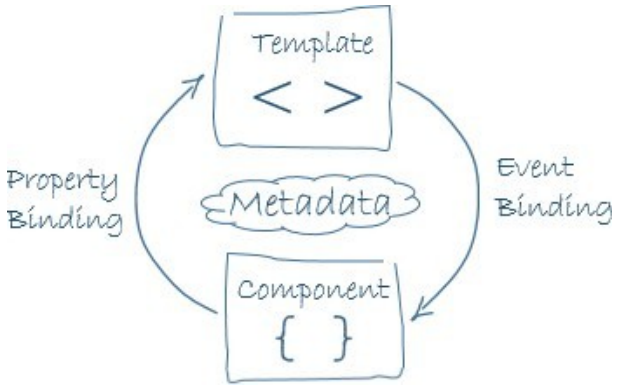


图 4。组件和模板之间的数据流(角度 g)

Angular 为每个 JavaScript 事件处理一次所有数据绑定，从应用程序的根到组件树中的所有子组件(Angular g)。

### 3.1.4 生命周期

组件的生命周期由 Angular 管理。它创建组件，呈现组件，并在数据更改时更新组件。Angular 提供生命周期挂钩，让用户有机会在组件生命周期的不同阶段与代码进行交互。

这些是 Angular 网站上解释的生命周期挂钩:

- *每当一个或多个数据属性更改时，都会调用 ngOnChanges()*
- *ngOnInit()在首次创建组件时或当 NgOnChanges()时初始化组件*
  被称为
- *ngDoCheck()在每次更改检测运行期间都会被调用，就在 ngOnChanges()和之后 ngOnInit()。用于检测 Angular 自身无法或不会检测到的变化*
- *ngOnChanges()和 ngOnInit()。用于检测 Angular 不能或不会进行的更改*
  发现自己。
- *ngaftercontentnit() - 在第一个 ngDoCheck()之后调用一次。Angular 将外部内容投影到组件视图后做出响应。*
- *ngAfterContentChecked() - 在 ngAfterContentInit()之后调用。让我们回应一下*
  ter Angular 已检查投影到组件中的内容。
- *ngAfterViewInit() - 在 ngAfterContentChecked()之后调用。让我们的用户在安之后进行响应-*
  gular 已经初始化了组件及其子视图。
- *ngAfterViewChecked() - 在 ngAfterViewInit() 之后调用，并在 ngAfterContent- Checked()之后每隔一个调用。让用户在 Angular 检查完所有渲染组件及其子视图后做出响应。*
- *ngOnDestroy() - 就在 Angular 破坏组件之前调用。通过取消订阅任何可观察到的内容，在组件销毁前进行清理，以避免内存泄漏。*

除了这些组件挂钩(Angular h)，其他 Angular 子系统或第三方库可能有自己的生命周期挂钩。

### 3.1.5 第三方软件包

第三方库可以在 Angular 中使用，方法是通过 NPM 安装它们并导入提供的功能(Angular i)。第三方库通过添加现成的样式和功能来扩展 Angular 的可用性。

最流行的角形造型库之一是角形材质。它由现成的角度材料设计组件组成。它们针对性能进行了微调，并与 Angular (Angular j)无缝集成。

## 3.2 反应

React 有适当的官方文档，在他们的家乡 reactjs.org 有许多关于如何开始的详细教程。对于新手和有经验的用户来说，主要概念解释得很清楚，说明性的，文档将深入更高级的主题。

### 3.2.1 句法

React 使用 JSX 语法，也就是 JavaScript 语法扩展。它是 JavaScript 和 HTML 的混合，可能会让一些人想起模板语言，除了拥有 JavaScript 的全部功能(React b)。

### 3.2.2 体系结构

JSX 被用来构建构成组件的元素。与浏览器 DOM 元素相比，React 元素是简单的对象，创建起来很便宜。虚拟 DOM 负责更新浏览器的 DOM。虚拟 DOM (VDOM)是一个概念，其中用户界面的“虚拟”表示保存在内存中，而“真实”DOM 通过一个库进行同步，例如 React DOM。这消除了手动属性和事件处理，并自动更新 DOM。

### 3.2.3 数据管理

React 组件是应用程序的构建模块，它允许用户界面被分成独立的、可重用的部分，这些部分可以被孤立地考虑。组件就像 JavaScript

接受输入(称为“道具”)并返回 React 元素的函数。组件可以转移到其他组件，使它们可以重用。通常，反应应用程序在视图层次结构的顶部有一个“应用”组件。

每个反应组件都可以，但不必有状态。在 React 中，状态是局部的，并封装成一个单一的组件。数据可以作为道具从父组件传递给子组件。道具是不可变的，不能在组件树层次结构中向上传递。状态可以改变，一旦状态改变，组件将被重新呈现。为了更好地控制组件，生命周期反应具有内置的生命周期功能，可以快速轻松地控制事件，如组件安装、卸载、接收数据等。

### 3.2.4 应用程序生命周期

图 3 展示了组件的生命周期。虚拟文档对象模型保存在缓存中。数据的变化是从上到下传递的。然后，React 比较虚拟 DOM 和真实 DOM 之间的差异，并在必要时进行更改。

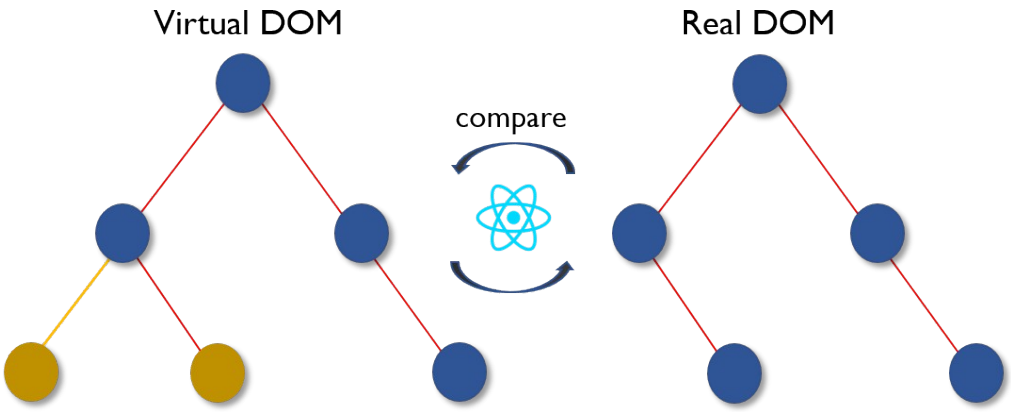


图 5。反应组件树和虚拟 DOM (Pngkey)

React 官方文档解释了组件生命周期和一些生命周期方法，但没有完整概述生命周期或生命周期挂钩。

### 3.2.5 第三方软件包

React 中的第三方包可以通过 NPM(节点包管理器)安装。在 React 之上运行第三方库增强了它的可用性，并增加了额外的功能，例如双向数据绑定和路由。React 最常用的第三方库之一可能是 Redux。

Redux 是一个 JavaScript 应用程序的状态容器，可以和 React 一起使用。在 Redux 中，整个应用程序状态存储在单个存储内的对象树中。随着应用程序的增长，Redux 存储区也在增长，并且可以划分为树状结构，与应用程序组件树相同。对于较小的应用程序来说，这似乎有些过分，但随着应用程序的扩展和变得更加复杂，这将证明是有益的(Redux a)。

图 4。显示 Redux store 如何从组件树的最低组件接收数据，并可以与树中较高的节点共享数据。

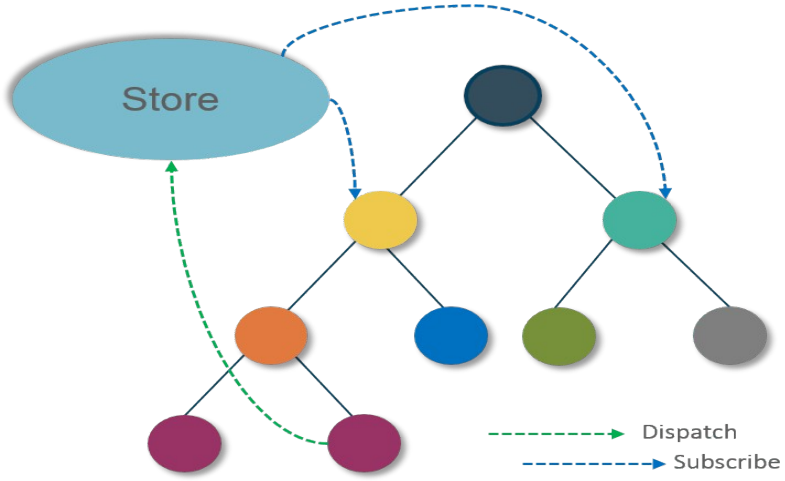


图 6。Redux 存储分发数据(Edureka)

### 3.3 某视频剪辑软件

Vue JS 在他们的主页 vuejs.org 上有关于如何开始的详细文档和例子。主要概念解释透彻，并在他们的主页和 js 小提琴中举例说明。

#### 3.3.1 句法

Vue.js 使用基于 HTML 的模板语法，它被绑定到底层的 Vue 实例 JavaS-script。所有 Vue 模板都是有效的 HTML。来自 Vue 应用程序的数据可以使用“小胡子”语法(双花括号)绑定到 HTML 中，例如:

*< p >消息:{{ msg }} </p >。在这些双花括号内编写的所有内容都将在 Vue 实例(VueJS a)的数据范围内被评估为 JavaScript。*

### 3.3.2 体系结构

Vue 应用程序由称为 Vue 实例的可重用组件组成，接受与新 Vue 实例相同的选项，例如数据、方法和生命周期挂钩。唯一的例外是一些根特定的选项。应用程序通常被组织成嵌套组件树，如图 7 所示。要在模板中使用组件，必须注册它们，以便 Vue 知道它们。组件可以在全球和本地注册。对组件进行全局注册使得它对该 Vue 实例组件树的所有子组件都可用
(VueJS b)。

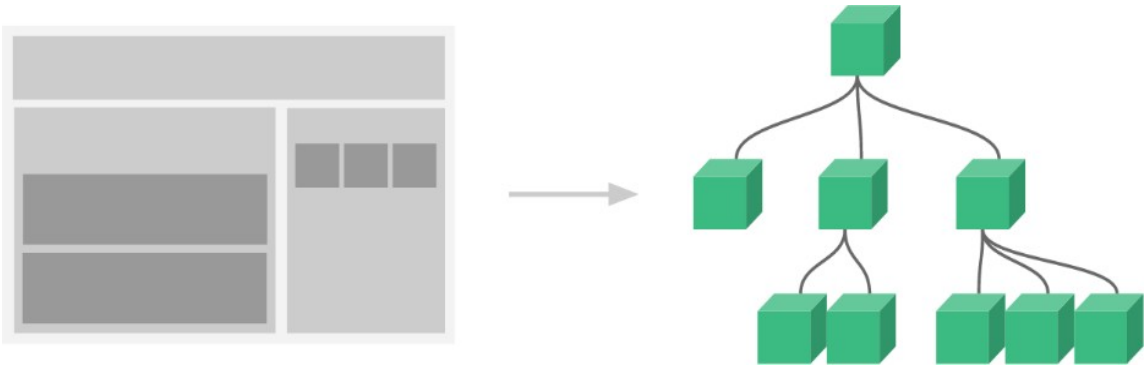

图 7。可视化用户界面和组件树

### 3.3.3 数据管理

应用程序数据存储在组件数据函数中，该函数充当组件状态。数据可以作为道具从父组件传递到子组件。当一个值作为一个适当的属性传递时，它就成为该组件实例的属性。组件中的属性值可以像数据属性一样处理。一个组件可以有无限数量的道具树(VueJS d；VueJS e)。

### 3.3.4 应用程序生命周期

类似于 React，Vue 使用虚拟 Dom 来更新页面。每个 Vue 实例在创建时都会经历一系列初始化步骤。它需要设置数据观察，编译模板，在 DOM 上挂载 Vue 实例，当数据发生变化时更新 DOM。在这个过程中，它还运行生命周期挂钩，允许用户在特定阶段添加他们的代码。

附录 1。展示了 Vue 实例的整个生命周期。当新的 Vue 实例被调用时，Vue 启动实例创建，但是在完成实例创建之前，通过调用 be- foreCreated 生命周期钩子，给用户一个机会来添加他们的代码。

然后 Vue 检查根元素是否有任何选项，比如模板。如果元素属性存在，它将被编译，如果不存在，Vue 实例父 HTML 元素将被用作模板。

在挂载编译后的元素之前，用户会获得另一个名为“挂载前”的生命周期钩子，在创建视图模型之后，用户会获得另一个机会，通过使用名为“挂载”的生命周期钩子来影响组件。之后，组件被呈现。

安装组件时，Vue 会主动观察数据的变化。当数据发生变化时，用户在更新之前会得到一个生命周期钩子，以便在虚拟 DOM 重新呈现之前添加他们的代码。

当组件被卸载并调用销毁函数时，用户有机会调用名为 beforeDestroy 的钩子，并且在所有组件方法和 chil- dren 被拆除后，有一个名为 destroy 的最终生命周期钩子。(Vue JS f .)

### 3.3.5 第三方软件包

Vue 中的第三方软件包可以通过 NPM 安装，也可以通过脚本标签作为依赖项导入。第三方库为 Vue 提供了额外的功能并增强了它的可用性。
Vue JS 最广泛使用的第三方包之一是 VueX。

VueX 是 Vue 应用程序的状态管理模式和库。它充当应用程序中所有组件的集中数据存储，确保状态只能以可预测的方式变化。VueX 的灵感来自 Flux、Redux 和 The Elm Architecture，唯一的例外是它是专门为 Vue 量身定制的。

图 8。说明了 VueX 如何与 Vue 一起运行。它从 Vue 获取新数据，启动动作，决定状态的变化方式，然后继续变化，将新状态传递回 Vue 组件。(VueX)。

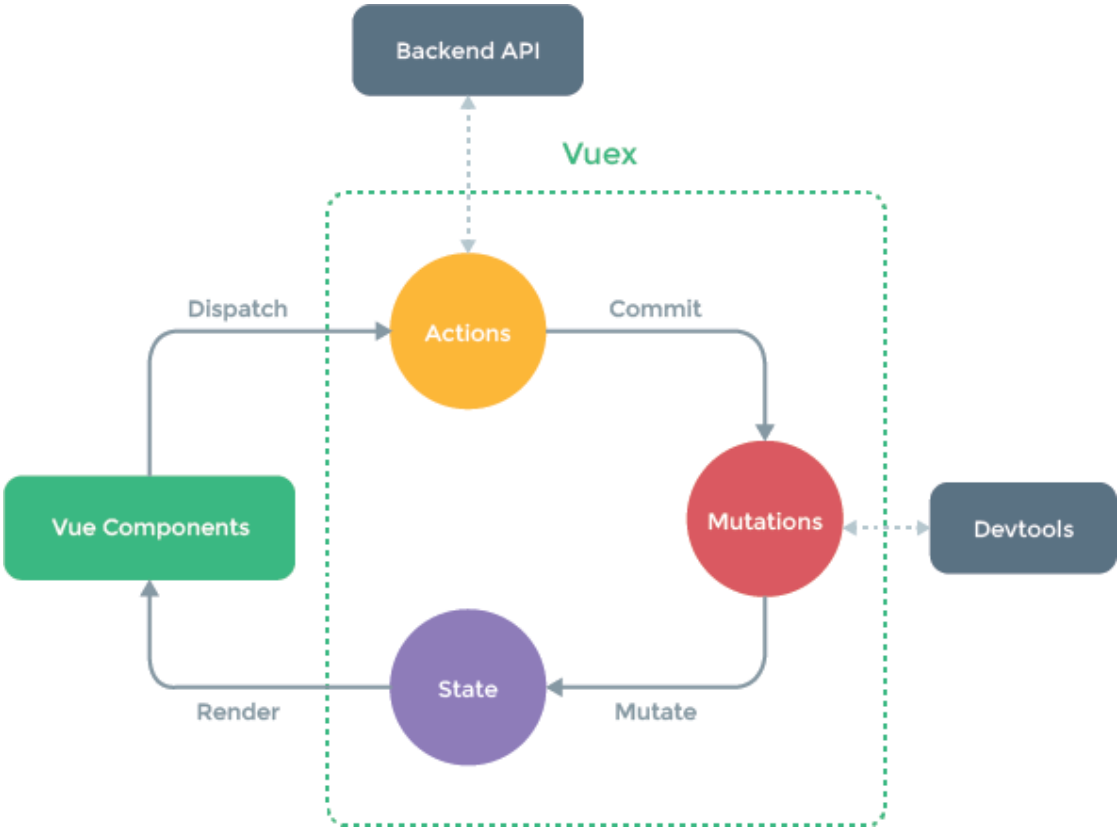


图 8。运行在 Vue 之上的 Vuex(Vuex)

# 4 框架性能

为了收集关于框架性能的数据，在每个框架中构建了一个小的测试应用程序来测试加载速度。所有测试应用程序都有相同的业务逻辑和可视化设计。

## 4.1 应用程序设计和共享部件

测试应用程序有一个数据表和一个菜单，具有五个功能:

- 创建一个有 1000 行的表
- 创建一个包含 10，0 00 行的表
- 向表中添加 1000 行
- 向表中添加 10，0 00 行
- 删除所有行

添加到表中的行由两列组成:一列由行号组成，另一列由 3 个随机单词组成的字符串组成。应用程序设计如图 9 所示。

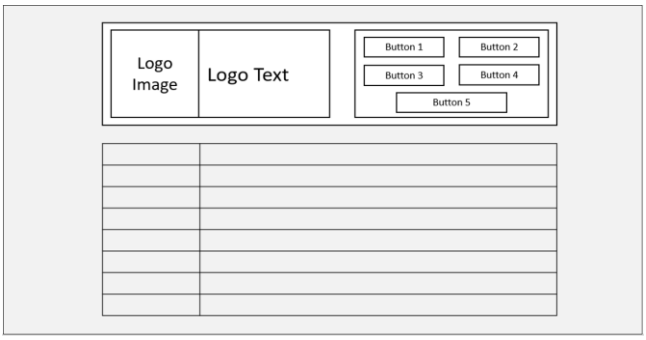


图 9。应用程序设计和结构。

应用程序分为三个独立的组件，应用程序是有两个子组件的父组件，如图 10 所示。

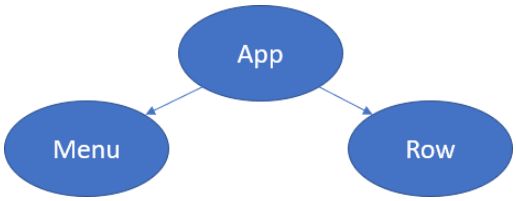


图 10。应用程序组件树

所有三个应用程序都使用相同的 CSS 文件进行样式化，并使用相同的香草 JavaScript 函数来生成数据。表行的数据由 buildData 函数在一个单独的文件中生成，并作为依赖项导入到项目中。

图 11 显示了 buildData 函数代码。函数 buildData 将行数量作为名为 count 的参数，并创建一个具有该长度的数组。然后调用 for 循环为数组中的每个位置传递一个对象。每个对象由两个键组成:id 和标签。它是一个整数，随着循环的每次迭代而增长。标签是由三个单词组成的字符串，通过调用一个随机数并将其用作索引从名为 A、B 和 c 的常量数组中选择一个单词来产生

```
function random(max) {
  return Math.round(Math.random() * 1000) % max;
}

const A = [
  "bad", "best", "better", "big", "certain", "clear", "different", "early", "e
  "great", "hard", "high", "human", "important", "international", "large", "la
  "national", "new", "old", "only", "other", "political", "possible", "public"
  "strong", "sure", "true", "white", "whole", "young", "crazy", "helpful", "mu
];
const B = [
  "red", "yellow", "blue", "green", "pink", "brown", "purple", "brown", "white
];
const C = [
  "area", "book", "business", "case", "child", "company", "country", "day", "e
  "life", "lot", "man", "money", "month", "mother", "Mr", "night", "number", "
  "right", "room", "school", "state", "story", "student", "study", "system", "
];


const buildData = function (count) {
  let nextId = 1;
  const data = new Array(count);
  for (let i = 0; i < count; i++) {
    data[i] = {
      id: nextId++,
      label: `${A[random(A.length)]} ${B[random(B.length)]} ${C[random(C.lengt
    };
  }
  return data;
}

export default buildData;
```

图 11。产生虚拟数据的普通 JavaScript 函数

## 4.2 测试环境

应用程序速度在谷歌 Chrome 浏览器中进行了测试和测量，该浏览器在安装在华硕 G73JH 笔记本电脑上的 Windows 10 操作系统上本地运行。笔记本电脑配备了 2.5 GHz i7 处理器和 16gb RAM，如图 12 所示。

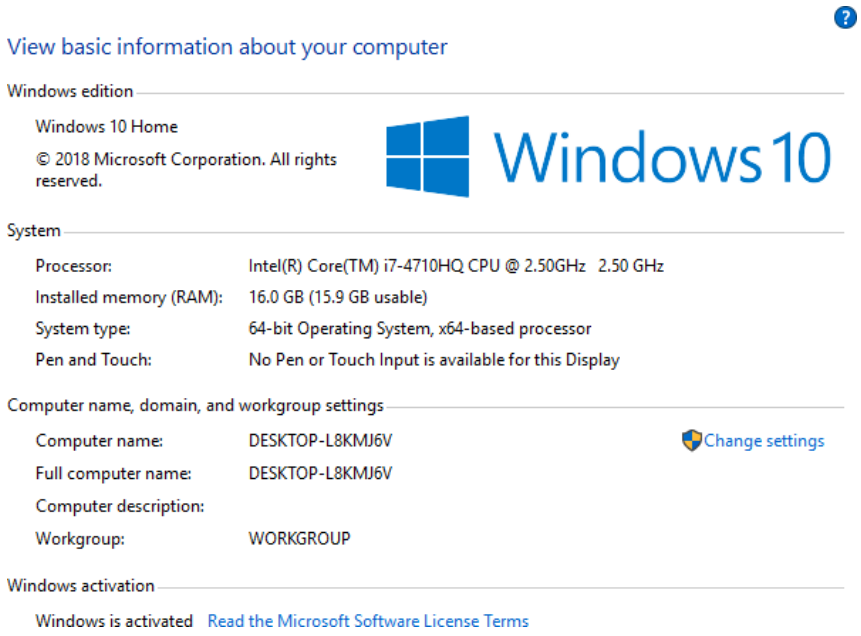


图 12。用于测试的计算机的技术属性

### 4.2.1 测试

应用程序在笔记本电脑上进行了本地测试，以避免可能由互联网速度导致的任何异常。总共进行了三次测试。每个测试都测量加载主页的时间，以及执行按钮调用的功能的时间。按钮“创建 1000”和“创建 10 000”被点击两次，以查看当重新运行相同的功能时是否有任何不同。在单击“创建 10 000”之前，通过单击清除按钮清除页面，以便在空页面上创建新的 10 000 行。速度以毫秒为单位测量。

还在亚马逊网络服务 S3 网站上提供了应用程序供进一步测试，这些应用程序是静态网站，被配置为实际位于瑞典斯德哥尔摩(AWS a；AWS b；AWS c)。此外，可以在上找到用于审查的应用程序代码
GitHub:https://github.com/elarsaks/Thesis.

## 4.3 有角的

Angular 应用程序生产构建为 15.7MB，托管在 AWS 上进行测试，托管在编写器 GitHub 上进行代码审查(AWS a；Elar Saks 2019)。应用程序组件存储在多个文件中，有一个单独的文件用于 HTML 模板、CSS 样式和 JavaS-script。

### 4.3.1 根组件:应用程序

图 13 显示了定义应用程序组件的类型脚本文件的内容。该文件从导入必要的依赖项开始，例如来自 Angular 核心的组件类和来自 dummyData 的测试数据创建函数。接下来，设置组件属性，例如组件标识符和支持文件位置。然后，应用程序组件被导出，带有一些数据属性作为组件状态，以及一些可以被调用来更新状态的方法。

```
import { Component } from '@angular/core';
import buildData from '../dummyData';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  numberOfRows: number = 0;
  data: object[];
  add: number =  0;
  create: number =  0;

  onAdd($event: number){
    this.numberOfRows = this.numberOfRows + $event;
    if (this.data === undefined) {
      this.data = buildData($event)
    } else {
      let data = this.data;
      this.data = data.concat(buildData($event))
    }
  }

  onCreate($event: number){
    this.numberOfRows = $event;
    this.data = buildData(this.numberOfRows)
  }

  onRemove(){
    this.numberOfRows = 0;
    this.data = [];
  }
}
```

图 13。App 组件声明

图 14 显示了应用组件模板或视图。该模板由一个 id 为 app 的 div 组成，用作两个子元素的容器。第一个子元素是一个名为 app-menu 的组件，它作为一个道具接收多个功能。第二个元素是一个 HTML 表，它通过使用 Angular 指令*ngFor 将应用程序行组件循环到其中。

```
<div id="app">
  <app-menu
    (addEvent)="onAdd($event)"
    (createEvent)="onCreate($event)"
    (removeEvent)="onRemove($event)"
  ></app-menu>

  <table class="data-table">
    <tbody>
      <app-row
        *ngFor="let row of data"
        [id]="row.id"
        [label]="row.label"
      ></app-row>
    </tbody>
  </table>
</div>
```

图 14。App 组件模板

### 4.3.2 菜单组件

图 15。显示菜单组件逻辑或控制器。菜单组件从必要的导入开始，如组件类、输出和事件发送器。导入后设置组件设置，如选择器和支持文件:视图模板及其样式。

接下来是 MenuComponent 类声明，首先列出从组件返回的方法。文件以定义要返回的函数结束。

```
import { Component, Output, EventEmitter} from '@angular/core';

@Component({
  selector: 'app-menu',
  templateUrl: './menu.component.html',
  styleUrls: ['./menu.component.css']
})

export class MenuComponent {
  @Output() addEvent = new EventEmitter<number>();
  @Output() createEvent = new EventEmitter<number>();
  @Output() removeEvent = new EventEmitter<number>();

  addRows(val :number){
    this.addEvent.emit(val)
  }

  createRows(val :number){
    this.createEvent.emit(val)
  }

  removeRows(){
    this.removeEvent.emit(0)
  }
}
```

图 15。菜单组件逻辑

在图 16 中。是菜单组件模板或视图。它由一些设计元素组成，例如标题和徽标以及菜单按钮容器。每个按钮都分配了用户在菜单类型脚本文件中定义的方法。

```
<div id="menu" class="menu-container" >

  <img alt="Logo" src="../assets/logo.png">
  <div class="framework" >
    <h1> Framework </h1>
  </div>

  <div class="buttons-container">
    <button class="Btn" (click)="createRows(10)"> Create 1000 rows </button>
    <button class="Btn" (click)="addRows(1000)"> Add 1000 rows </button>
    <button class="Btn" (click)="createRows(10000)"> Create 10 000 rows </button>
    <button class="Btn" (click)="addRows(10000)"> Add 10 000 rows </button>
    <button class="Btn" (click)="removeRows()"> Clear </button>
  </div>
</div>
```

图 16。菜单组件视图

### 4.3.3 行组件

*行组件逻辑部分如图 17 所示。由于它没有太多的功能，所以与其他产品相比，它非常短。它从一些依赖项导入开始，然后是组件设置，其中组件选择器是用支持文件位置声明的。它以导出一个带有两个传入数据属性的 RowComponent 类结束:id 和 label。*

```
import { Component, Input } from '@angular/core';

@Component({
  selector: 'app-row',
  templateUrl: './row.component.html',
  styleUrls: ['./row.component.css']
})

export class RowComponent {
  @Input() id: number;
  @Input() label: string;

}
```

图 17。行组件逻辑

*行组件模板仅由一个表行元素组成，由两列组成。表列有分配给它们的类，内容由 Angular 表达式传递，使用双花括号。*

```
<tr>
  <td class="small-col" > {{id}} </td>
  <td class="big-col" > {{label}} </td>
</tr>
```

图 18。行组件模板

### 4.3.4 试验结果

表 2。以毫秒为单位显示 Angular 应用程序生产构建速度测试结果。表格的最后一列显示了三次测试的平均值，并向下舍入到没有小数位的整数。最后一栏将在本文的分析部分使用。

表 2。角速度测试结果

| 有角的 | 测试 1。 | 测试 2。 | 测试 3。 | 平均的 |
|---|---|---|---|---|
| 加载页面 | 389.6 | 393.9 | 425.7 | 403 |
| 创建 1000 个 | 404.4 | 439.5 | 306.6 | 384 |
| 重新创建 1000 | 420.2 | 287.3 | 284.4 | 331 |
| 加 1000 | 288.5 | 237.9 | 234.8 | 254 |
| 创造 10 000 | 6508.6 | 6268 | 6306.6 | 6361 |
| 重新创建 10，000 | 7726.3 | 7335 | 7218.5 | 7427 |
| 增加 10，0 00 | 7823 | 7243.2 | 7804.2 | 7623 |
| 全部删除 | 2315.9 | 2205.9 | 2064.9 | 2196 |

## 4.4 反应

React 应用程序生产构建为 587KB，托管在 AWS 上进行测试，托管在 writers GitHub 上进行代码审查(Elar Saks 2019 b)。

### 4.4.1 根组件:应用程序

应用程序是视图层次结构中最高的组件，是将应用程序的其余部分结合在一起的线索。App.js 文件从导入依赖项开始，例如来自 React 库的 components 类、CSS 文件和子组件。图 19。显示应用程序组件导入。

```
import React, { Component } from 'react';
import './App.css';
import Menu from './components/Menu';
import Row from './components/Row';
import buildData from './dummyData';
```

图 19。在 App.js 中导入依赖项

导入后，应用程序组件被声明为一个 JavaScript 类，并扩展了 React library。React 组件可以分为两部分，组件逻辑和组件视图。App 是一个有状态的组件，从状态声明开始，然后是组成组件逻辑的函数，如图 20 所示。

```
class App extends Component {

  state = {
    data:[],
  }

  create = (amount) => {
    this.setState({ data: buildData(amount) });
  }

  add = (amount) => {
    this.setState({ data: this.state.data.concat(buildData(amount)) });
  }

  remove = () => {
    this.setState({ data: [] });
  }
```

图 20。应用程序组件的前半部分

App 组件的第二部分如图 21 所示。它是一个返回 App 组件视图的渲染函数。

应用程序组件视图由一个 div 组成，div 由一个菜单组件和一个 HTML 表组成。根据应用程序状态中指定的数量，HTML 表中有一个行组件被呈现到其中。

应用程序状态可以通过应用程序组件前半部分声明的函数来更改。这些功能作为道具传递给菜单组件。

虽然数据只能从上到下流动，不能从子组件传递到父组件，但仍然可以从菜单组件更新应用程序组件状态。

只有函数调用从父组件传递到子组件。函数在菜单组件中调用，但在应用组件中声明，并将应用组件用作执行环境，从而改变应用组件的状态。

```
  render() {
    return (
      <div className="App">
        <Menu
          create={this.create}
          add={this.add}
          remove={this.remove}
        />

        <table className="data-table">
          <tbody>
            {this.state.data.map((item, i) => (
              <Row key={i} item={item} ></Row>
            ))}
          </tbody>
        </table>
      </div>
    );
  }
}
export default App;
```

图 21。应用程序组件的后半部分

### 4.4.2 菜单组件

*菜单组件是一个无状态组件，但它被声明为一个类，以说明可能性。*

```
import React, { Component } from 'react';
import '../App.css';
import logo from '../logo.png'
import Button from './Button'

class Menu extends Component{
  render(){
    const { create, add,  remove } = this.props;
    return(
      <div className="menu-container" >

        <img src={logo} alt={'logo'} />
        <div className="framework" >
          <h1> Framework </h1>
        </div>

        <div className="buttons-container">
          <Button id="create1000" className="Btn" funk={() => create(1000) } title="Create 1000 rows" />
          <Button id="add1000" className="Btn" funk={() => {add(1000)}} title="Add 1000 rows" />
          <Button id="create1000" className="Btn" funk={() => {create(10000)}} title="Create 10 000 rows" />
          <Button id="add10000" className="Btn" funk={() => {add(10000)}} title="Add 10 000 rows" />
          <Button id="delete" className="Btn" funk={remove} title="Clear" />
        </div>
      </div>
    );
  }
}
export default Menu;
```

图 22。菜单组件

### 4.4.3 行组件

*行组件是一个无状态的功能组件，它将道具作为输入并返回一个视图。如果组件没有状态，则不需要类组件。*

```
import React from 'react';

function Row(props) {
  return (
    <tr className="data-row ">
      <td >{props.item.id}</td>
      <td className="col-md-4">
        {props.item.label}
      </td>
    </tr>
  );
}
export default Row;
```

图 23。行组件

### 4.4.4 试验结果

表 3。以毫秒为单位显示了 React 应用程序生产构建速度测试结果。表格的最后一列显示了三次测试的平均值，并向下舍入到没有小数位的整数。最后一栏将在本文的分析部分使用。

表 3。反应速度测试结果

| 反应 | 测试 1。 | 测试 2。 | 测试 3。 | 平均的 |
|---|---|---|---|---|
| 加载页面 | 259.4 | 268.8 | 279.4 | 269 |
| 创建 1000 个 | 400.9 | 400.6 | 459.1 | 420 |
| 重新创建 1000 | 174.2 | 206 | 191.2 | 190 |
| 加 1000 | 364.2 | 364.5 | 380.5 | 370 |
| 创造 10 000 | 2787.1 | 2978.9 | 2831.7 | 2866 |
| 重新创建 10，000 | 1181.7 | 1201.3 | 1145.7 | 1176 |
| 增加 10，0 00 | 3888.7 | 3906.7 | 3766.1 | 3854 |
| 全部删除 | 589.4 | 619.9 | 611.3 | 607 |

## 4.5 某视频剪辑软件

Vue 应用程序分为四个部分。应用程序是主要的根组件，有两个子组件:菜单和行。Vue 应用程序生产构建为 624KB，托管在 AWS 上进行测试，托管在编写器 GitHub 上进行代码重新查看(AWS c；Elar Saks 2019)。

### 4.5.1 根组件:应用程序

应用程序是 Vue 应用程序的根组件，并且是其余组件的父组件。它从视图模板部分开始，如图 24 所示。模板是由
来自 App div，包含菜单组件和一个表，根据 App 组件的数据属性将行组件呈现到其中。

```
<template>
  <div id="app">
    <Menu v-bind="{createRows, addRows, removeRows}" />
    <table class="data-table">
      <tbody>
        <Row v-for="item in data"
        :rowId="item.id"
        :rowLabel="item.label"
        class="data-row" />
      </tbody>
    </table>
  </div>
</template>
```

图 24。App.vue 模板部分

应用程序组件的后半部分由一个脚本标签组成，该标签由组成 Vue 实例及其属性的 JavaScript 组成。脚本标签中的代码可以在图 25 中看到。首先从导入子组件和构建数据函数开始。然后，应用程序组件被导出为由数据函数、组件对象和方法对象组成的对象。

```
<script>
import Menu from './components/Menu.vue'
import Row from './components/Row.vue'
import buildData from './dummyData'

export default {
  name: 'app',
  data() {
    return {
      numberOfRows: 0,
      data: [],
    }
  },
  components: {
    Menu,
    Row
  },
  methods: {
    addRows( amount) {
      this.numberOfRows = this.numberOfRows + amount;
      let data = this.data;
      this.data = data.concat(buildData(amount))
    },
    createRows( amount) {
      this.numberOfRows = amount;
      this.data = buildData(this.numberOfRows);
    },
    removeRows() {
      this.numberOfRows = 0;
      this.data = [];
    },
  },
}
</script>
```

图 25。App.vue 的后半部分

### 4.5.2 菜单组件

与 App 组件类似，Menu 组件由两部分组成:由 HTML 组成的模板部分和由 JavaScript 组成的脚本标签。

```
<template>
  <div>
    <div id="menu" class="menu-container" >

      <img alt="Logo" src="../assets/logo.png">
      <div class="framework" >
        <h1> Framework </h1>
      </div>

      <div class="buttons-container">
        <button class="Btn" @click="createRows(1000)"> Create 1000 rows </button>
        <button class="Btn" @click="addRows(1000)"> Add 1000 rows </button>
        <button class="Btn" @click="createRows(10000)"> Create 10 000 rows </button>
        <button class="Btn" @click="addRows(10000)"> Add 10 000 rows </button>
        <button class="Btn" @click="removeRows()"> Clear </button>
      </div>
    </div>
  </div>
</template>

<script>
import Button from './Button.vue'

  export default  {
    props: ['createRows', 'addRows', 'removeRows'],
    components: {
      Button,
    }
  }
</script>
```

图 26。菜单组件

### 4.5.3 行组件。

*行组件没有任何应用程序逻辑，因此被编写为只呈现视图的功能组件。这使得代码更短、更清晰，如图 27 所示。*

```
<template functional>
  <tr>
    <td class="small-col" slot="rowId" >{{props.rowId}}</td>
    <td class="big-col" slot="rowLabel" >{{props.rowLabel}}</td>
  </tr>
</template>
```

图 27。行组件

### 4.5.4 试验结果

表 4。以毫秒为单位显示 Vue 应用程序生产构建速度测试结果。表格的最后一列显示了三次测试的平均值，并向下舍入到没有小数位的整数。最后一栏将在本研究的分析部分使用。

表 4。Vue 速度测试结果

| 某视频剪辑软件 | 测试 1。 | 测试 2。 | 测试 3。 | 平均的 |
|---|---|---|---|---|
| 加载页面 | 233.3 | 264.6 | 240.44 | 246 |
| 创建 1000 个 | 248.1 | 213.5 | 234.2 | 232 |
| 重新创建 1000 | 125.6 | 111.8 | 112.6 | 117 |
| 加 1000 | 193.7 | 180.7 | 180.2 | 185 |
| 创造 10 000 | 1446.8 | 1448.1 | 1446.7 | 1447 |
| 重新创建 10，000 | 553.2 | 533.9 | 553.9 | 547 |
| 增加 10，0 00 | 1423.2 | 1452.3 | 1433.1 | 1436 |
| 全部删除 | 567.6 | 334.8 | 613.9 | 505 |

# 5 估价

比较软件工具，尤其是框架，是很困难的，因为它们有各种各样的用例和领域，有些在一个方面更好，有些在另一个方面更好。在本章中，Angular、React 和 Vue 在三个方面进行了比较:在用户中的受欢迎程度、框架学习曲线和应用程序加载速度。

## 5.1 流行

为了比较受欢迎程度，本文收集并分析了来自行业领先平台的数据，如 GitHub、NPM 和 Stack Overflow。

### 5.1.1 GitHub 数据

GitHub 为 JavaScript 市场提供了独特的见解，因为它提供了用户最“喜欢”的东西、大多数开发人员在哪个框架上工作以及哪个框架最有问题的信息。这些数据可能有些偏颇，因为旧框架有更多的时间来收集关注者，而更复杂的框架有时间来解决更多的问题。记住这一点，我们可以看看表 1 中的 GitHub 数据。请注意以下几点。

Vue 的星数比 React 略多，12.5 万颗星中有 13.3 万颗。这大约是 6%多，没有太大的区别。另一方面，棱角分明是个例外，有 4.6 万颗星。这比它的竞争对手少了两倍多。

React 是这三个中分叉最多的框架，有 2.2 万个分叉 vs. Vue，1.8 万个分叉，Angular 12 万个分叉。虽然叉子数量的差异没有星星数量的差异那么剧烈，但也足够大，值得信赖。

根据公开的问题，Angular 要么是这三个中最受欢迎的，要么是最有问题的。Angular 有 2331 期关于它，vs. React 有 453 期，vs. Vue 有 184 期。

Angular 是这些框架中最古老的，其次是 React，然后是 Vue。这可以解释很多关于它的问题，但是看看剩下的

数据，可以肯定地说，Angular 是 GitHub 上最不受欢迎的框架，在这 3 个中。Vue 比 React 更受欢迎，但是 React 有更多的人从事这项工作，所以他们两个几乎一样受欢迎。

### 5.1.2 NPM 趋势

来自 NPM 趋势的数据显示了随着时间的推移，每个框架下载了多少包。这些数据可能会有点误导，因为一些框架更大，需要第三方更少的支持包。例如，Angular 比 React 和 Vue 大得多，功能也多得多，因此在 NPM 趋势中获得的下载量更少。但是看看图 1。，很明显 React 的下载量是其他两个框架的数倍。Vue 的下载量略高于 Angular，并且正在稳步增长，但这种差异并不像 React 那样明显。

### 5.1.3 堆栈溢出趋势

Stack Overflow 是一个连接全球软件开发人员社区的云平台，它使开发人员能够共享代码并互相帮助。查看堆栈溢出数据可以很好地说明世界各地的开发人员都在使用和谈论什么。尽管如此，一些更难的框架可能比更容易的框架得到更多的问题，因此看起来比它们更受欢迎。

图 2。显示 2009 年至 2019 年间针对每个框架提出的问题的百分比。我们可以看到 Angular 的人气在 2017 年达到顶峰，在所有关于 Stack Overload 的问题中，有 3%是关于它的。从那以后，情况一直差不多。Angular 紧随其后的是 React，在 2018 年达到顶峰，所有问题中有 2.5%的问题。Vues 的增长没有那么积极，起步较晚，2019 年最高为 0.8%。

虽然看起来，由于 Angular 和 React 已经在 Stack Overflow 上实现了它们的潜力，并且 Vue 仍在增长，但差距仍然足够大，可以有把握地说 Angular 和 React 是该平台上更受欢迎的两个。正如它们之间的差别，两者是如此之小；很难说哪一个更受欢迎。

## 5.2 学习曲线

为了理解学习每个框架的难度，在语法、架构、数据管理、生命周期和第三方库中对框架进行了比较。在这一章中，收集的关于学习每个框架的数据被分析。

### 5.2.1 句法

在这三个框架中，就语法而言，Angular 是最难学的，因为它使用了 TypeScript。新用户不仅要学习一个新的框架，还要学习 Ja- vaScript 的新超集，甚至能够使用这个框架。

就学习曲线而言，React 和 Vue 彼此更接近。虽然 Vue 语法可以被认为比 React 更容易，因为它的语法是有效的 HTML，看起来更像传统的网页内置 HTML、CSS 和 JavaScript。React 使用的是 JSX 语法，对于一个对 HTML 和 JavaScript 有扎实基础的人来说，这很容易学习，但可能会觉得比 Vue 语法更难上手。

### 5.2.2 体系结构

所有三种架构框架都非常相似。应用程序可以想象成一个组件树，其中顶部是根组件，可以有无限的子组件层次结构。唯一的例外是 Angular，它的所有东西都是一样的，但是它的组件也被分成模块，这些模块包含与属于某个模块的组件相关的支持代码文件。

因此，可以肯定地说，学习 React 和 Vue 的架构同样具有挑战性，而 Angular 的架构则稍微复杂一些。

### 5.2.3 数据管理

在所有三个框架中，数据在组件树中从一个组件流向另一个组件。父组件传递的数据成为子组件的属性，不能被破坏。

Angular 与 React 和 Vue 的不同之处在于具有双向数据绑定。在 Angular 中，子组件能够将数据传递给父组件。因此，从组件树下来的数据可以从子组件接收、更改和传递回来，使得构建动态网站更加容易。

在 React 和 Vue 中，子组件不能将数据传递给父组件，除非使用第三方库。

### 5.2.4 生命周期

所有这三个框架都有相似的生命周期和相似的生命周期挂钩。其中最容易学习的是 Vue，因为 Vue 官方技术文档为用户提供了组件生命周期的完整流程图，包括所有的生命周期方法。

第二容易学习的是 Angular，因为它的官方文档从头到尾都有一个完整的生命周期挂钩列表。

就生命周期而言，他们中最难学习的是“反应”。虽然它的官方文档清楚地解释了组件生命周期的工作方式，并且学习它相对容易，但是很难找到所有生命周期挂钩的完整地图或列表。

### 5.2.5 第三方软件包

这三个框架都利用了节点包管理器，并且可以用来安装第三方库。这使得在每个框架上安装扩展同样舒适。值得注意的是，它对 React 和 Vue 比对 Angular 更有利。安装第三方库允许用户向 React 和 Vue 添加一些重要的功能，这在 Angular 中已经存在。组件外的路由器和数据存储就是很好的例子。

## 5.3 表演

并非所有框架都是相同的，对于不同类型和规模的应用程序，它们的性能可能会有所不同。然而，为这项研究构建的小应用程序应该提供一些关于所讨论的框架性能的提示。

从代码示例中可以看出，这三个框架的应用程序架构非常相似，用户编写的代码量也非常相同。如果应用程序的范围和规模会增长，这种情况可能会改变，但对于这些类型的小应用程序，没有显著的差异。

但是，从表 5 中可以看出，应用程序生产构建的大小有很大的不同。虽然 React 和 Vue 生产版本的大小非常相似，但 Angular 最终应用程序比其余版本大得多。角度分布为 15.7 兆比特，即 15 700 千字节，比 Vue 大 25 倍，比 React 生产版本大约 27 倍。

这可能是因为 Angular 是为更大、更复杂的应用程序而构建的，并带有更大的代码库。当构建更大的应用程序时，应用程序生产构建可能会在框架之间平衡。但在互联网速度和带宽非常重要的地区和领域构建小型应用程序时，这绝对是需要牢记的。

表 5。生产规模

| | |
|---|---|
| 反应 | 587 千字节 |
| 某视频剪辑软件 | 624 千字节 |
| 有角的 | 15.7 兆字节 |

对于每个框架中构建的每个应用程序，运行了三个性能测试。速度测试结果收集在表 2 中。，表 3。，以及表 4。每个框架的三个测试结果的平均值合并到表 6 中。

表 6。最左边一栏说明了测量的功能速度。结果以毫秒为单位显示在右侧的列中。列是彩色编码的，绿色表示最快的功能运行速度，黄色表示第二快，红色表示最慢。最后两行用于分析。

名为“总计”的行计算每个框架运行所有功能所用的总时间。最后一行名为“乘数”，用于说明与最快框架(Vue)相关的总运行速度。乘数取 Vue 花费的总时间，用它除以其他框架工作的总时间，说明任何给定框架比 Vue 慢多少倍。

Angular 的生产规模大了几十倍，我们可以放心地认为它是三者中最慢的，但看看表 6。可以看出，情况并非总是如此。加载页面时 Angular 是最慢的，但是加载速度的差异没有应用程序大小的差异大。Angular 加载登录页面花费了大约 400 毫秒，而 Vue 和 React 都停留在两个数百毫秒的中间。Angular 还设法在创建和添加 1000 行方面击败了 React。其余的速度测试都按照预期进行了，它比最接近的竞争对手 React 慢了 2 倍多。

尽管 Vue 的生产规模略大于 React，但它在所有测试中都击败了 Angular 和 React，总共比 Angular 快 5 倍，比 React 快 2 倍。
反应速度大约是 Angular 的两倍半。

表 6。框架加载速度

| | 有角的 | 反应 | 某视频剪辑软件 |
|---|---|---|---|
| 加载页面 | 403 | 269 | 246 |
| 创建 1000 个 | 384 | 420 | 232 |
| 重新创建 1000 | 331 | 190 | 117 |
| 加 1000 | 254 | 370 | 185 |
| 创造 10 000 | 6361 | 2866 | 1447 |
| 重新创建 10，0 00 | 7427 | 1176 | 547 |
| 增加 10，0 00 | 7623 | 3854 | 1436 |
| 全部删除 | 2196 | 607 | 505 |
| 总数 | 24978 | 9752 | 4715 |
| 乘数 | 5.30 | 2.07 | 1.00 |

# 6 结论

这项研究的目的是比较三种最流行的 JavaScript 框架的流行程度、学习难度和性能，以便读者在选择学习哪个框架或在项目中使用哪个框架时做出明智的决定。

## 6.1 流行

在这项研究中，关于框架流行度的信息是从主要的云服务提供商那里收集的，比如 GitHub、NPM 和 Stack Overflow。

在 GitHub 上，Vue 被证明是最受欢迎的，只比 React 领先一点点，多了几个明星，少了几期。Angular 是 GitHub 上的失败者，明星比其他人少一半，问题多几倍。

React 通过 NPM 下载的包数是其他两个框架总和的数倍。可能 Angular 只是不需要那么多第三方库，因为 React 和 Vue 仍然是一个新的和不断发展的框架，但可以肯定的是，React 是目前为止 NPM 最受欢迎的框架。

似乎 Angular 在堆栈溢出上的受欢迎程度已经达到顶峰，即将被 React 接管。“反应”和“角度”分享线索，两者各自占提问总数的 2.5%到 3%。Vue 虽然在稳步增长，但只占提问总数的 8%。

在 Github 和 Stack Overflow 上，React 完全控制了 NPM 并分享了领先优势，可以肯定地说，就 2019 年而言，React 是最受欢迎的 JavaScript 库。

## 6.2 学习曲线

学习一个新的框架对个人学习者来说是主观的，不能定量测量。这项研究的结果是作者的唯一意见，应该给予适当的批评。框架学习曲线在语法、架构、数据管理、生命周期以及使用第三方库的便利性方面进行了比较。

Vue 有最简单的语法，因为它是唯一一个使用纯 HTML、CSS 和普通 JavaScript 的语法。架构方面，Vue 和 React 比 Angular 更容易，它除了组件之外还使用模块。Angular 还内置了双向数据绑定，这只能通过其他框架，通过使用第三方库来实现。所有三个框架都有相似的生命周期，并且能够通过 NPM 安装第三方库。

对于新手来说，Vue 是最容易学习的一个，其次是 React，它几乎没有更复杂的语法和更糟糕的文档。Angular 是这三个中最大也是最难学的。

## 6.3 表演

为了比较框架的性能，在每个框架中构建了一个简单的单页应用程序。应用程序然后在加载速度测试。

Vue 是所有测试中表现最快的框架。总的来说，Vue 比 Angular 快五倍，比 React 快两倍多。八次测试中，有六次反应比 Angular 快。React 的生产构建最小，为 587KB，Vue 的生产构建为 624KB，Angular 的生产构建最大，为 15.7MB

值得注意的是，框架在速度方面的性能可能会随着应用程序大小和复杂性的变化而变化。因此，更复杂的应用需要更多的测试。

## 6.4 定论

对于找工作来说，反应是最受欢迎的，可能是最好的学习框架。最容易学和表演最快的是 Vue。Angular 是最难学的，也是表现最慢的框架。但是 Angular 缺乏的是速度和易学性，它弥补了可用性的不足，在构建大规模应用程序方面可能优于其他框架。在构建更大、更复杂的多页面应用程序时，需要更多的研究来比较框架。

# 参考


棱角分明的ａ．家。网址:https://angular.io/.查阅日期:2019年3月21日。

角度ｂ．指南/架构。网址:https://angular.io/指南/建筑。查阅日期:2019年7月13日。

指南/模板-语法。网址:https://angular.io/指南/模板-语法。查阅日期:2019年7月13日。

角度指南/架构模块。网址:https://angular.io/指南/架构-模块。查阅日期:2019年7月13日。

角度指南/模板-语法。网址:https://angular.io/指南/模板-语法。查阅日期:2019年7月13日。

角度指南/架构模块。网址:https://angular.io/guide/architecture-modules.查阅日期:2019年7月13日。

角度指南/架构-组件。网址:https://angular.io/指南/架构-组件#数据绑定。查阅日期:2019年7月13日。

角度ｈ．导轨/生命周期挂钩。网址:https://angular.io/指南/生命周期挂钩。查阅日期:2019年7月13日。

角度Ｉ．指南/使用库。网址:https://angular.io/指南/使用-图书馆。查阅日期:2019年7月13日。

材料。网址:http://material.angular.io。查阅日期:2019年7月13日。

AWS a. Angular。网址:http://thesis-angular.s3-website.eu-north-1.amazonaws.com。完成日期:2019年9月4日

AWS ｂ．做出反应。网址:http://thesis-react.s3-website.eu-north-1.amazonaws.com。访问日期:2019年9月4日

角形的。网址:http://thesis-vue.s3-website.eu-north-1.amazonaws.com/.访问日期:2019年9月4日

克伦威尔诉2017。在电线之间:专访Vue.js的创作者尤雨溪。网址:https://medium . freecodecamp . org/inter-the-wires-an-interview-with-vue-js-creator-Evan-you-e 383 CBF 57 cc 4。查阅日期:2019年3月21日。

Dziwoki，男，2017年。网址:https://gorrion.io/blog/angularjs-vs-angular.查阅日期:2019年3月22日。

爱德华卡。反应-还原教程。网址:https://www.edureka.co/blog/react-redux-tutorial/.完成时间:2019年3月22日。

埃拉·萨克斯。2019.论文。网址:https://github.com/elarsaks/Thesis/tree/master/react_test。查阅日期:2019年9月4日。

GitHub a. Home。网址:https://github.com/.查阅日期:2019年3月22日。

GitHub b. Angular。网址:https://github.com/angular/angular.查阅日

期:2019年3月22日。GitHub c .反应。网

址:https://github.com/facebook/react.查阅日期:2019年3月22日。

GitHub d. Vue。网址:https://github.com/vuejs/vue.查阅日期:2019年3月22日。

关于星星。网址:https://help.github.com/en/articles/about-stars.查阅日期:2019年3月22日。

关于叉子。网址:https://help.github.com/en/articles/about-forks.查阅日期:2019年3月22日。

关于问题。网址:https://help.github.com/en/articles/about-issues.查阅日期:2019年3月22日。

关于问题。网址:https://help.github.com/en/articles/about-issues.查阅日期:2019年3月22日

古德利，2017年。棱角分明的历史。网址:https://www.angularjswiki.com/angularjs/his-angularjs的历史。查阅日期:2019年3月21日。

Npm。关于npm。网址:https://docs.npmjs.com/about-npm/.查阅日期:2019年3月21日。

Npm趋势。角度vs反应vs Vue。网址:https://www.npmtrends.com/angular-vs-react-vs-vue. 查阅日期:2019年3月21日。

PngKey。虚拟世界–对虚拟世界做出反应。网址:https://www.pngkey.com/max-pic/u2e 6t 4 u2o 0 o 0 o 0/。查阅日期:2019年3月21日。

反应a .回家。网址:https://reactjs.org/.查阅日期:2019年3月21日。

反应b .介绍JSX。网址:https://reactjs.org/docs/introducing-jsx.HTML.查阅日期:2019年4月7日。

反应c .渲染元素。网址:https://reactjs.org/docs/rendering-elements.HTML.完成日期:2019年4月7日

反应d .虚拟DOM和内部。网址:https://reactjs.org/docs/faq-internals.HTML.完成时间:2019年4月7日。

反应组件和道具。网址:https://reactjs.org/docs/components-and-props.HTML。查阅日期:2019年4月7日。

对状态和生命周期做出反应。网址:https://reactjs.org/docs/state-and-lifecycle.HTML.完成时间:2019年4月7日。

Redux a .导言。网址:https://redux.js.org/introduction/getting-started.查阅日期:2019年6月29日。

堆栈溢出。网址:https://insights.stackoverflow.com/.查阅日期:2019年3月21日

堆栈溢出ｂ．堆栈溢出趋势。网址:https://insights.stackover-flow.com/trends?tags=r%2Cstatistics.查阅日期:2019年3月21日

指南/语法。网址:https://vuejs.org/v2/guide/syntax.HTML.查阅日期:2019年6月30日

指南/组件。网址:https://vuejs.org/v2/guide/components.HTML.完成日期:2019年6月30日

图像/组件。网址:https://vuejs.org/images/components.png.查阅日期:2019年6月30日

指南/组件/数据必须起作用。网址:https://vuejs.org/v2/guide/components.HTML #数据必须是函数。查阅日期:2019年6月30日

指南/组件/用道具将数据传递给子组件。网址:https://vuejs.org/v2/guide/components.将数据传递给子组件Props. 查阅日期:2019年6月30日

指南/指南/实例。网址:https://vuejs.org/v2/guide/instance.HTML#Creating-一个实例。查阅日期:2019年6月30日

VueX ａ．什么是VueX？网址:https://vuex.vuejs.org/.查阅日期:2019年6月30日

维基百科a. React (JavaScript库)。网址:https://en . Wikipedia . org/wiki/React _(Javas-cript _ library)查阅日期:2019年3月21日。

维基百科。网址:https://en.wikipedia.org/wiki/Microsoft_Type-Script 查阅日期:2019年7月13日

# 附录

附录 1。应用程序生命周期。