

快速掌握Vue.js技术，提高Web前端开发效率

- 按入门的难易程度讲解Vue.js知识从简单到复杂，逐层推进，深入讲解
- 结合案例作者对实际项目的开发经验



渐进式JavaScript框架

# Vue.js快速入门

申思维 著



本书  
示例源代码



清华大学出版社

Web  
前端技术  
丛书

# V Vue.js 快速入门

申思维 著

清华大学出版社  
北京

## 内 容 简 介

目前单页应用框架层出不穷，其中 Vue.js 是十分耀眼的项目之一，受到国内外开发人员的极度推崇。

全书分为 8 章，内容包括 Vue.js 概述、Vue.js 的安装、定义页面、渲染视图、路由、发送 HTTP 请求、表单的绑定和提交、打包、部署、解决 js 的跨域问题、Debug、Component、Mixin、Vuex、页面的生命周期等，最后还给出一个实战案例供读者了解 Vue.js 项目开发过程。

本书适合 Vue.js 初学者、Web 前端开发人员，也适合高等院校和培训学校的师生教学参考。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售

版权所有，侵权必究。侵权举报电话：010-62782989 13701121933

### 图书在版编目 (CIP) 数据

Vue.js 快速入门 / 申思维著. —北京：清华大学出版社，2019

(Web 前端技术丛书)

ISBN 978-7-302-51754-2

I. ①V… II. ①申… III. ①网页制作工具—程序设计 IV. ①TP392.092.2

中国版本图书馆 CIP 数据核字 (2018) 第 271412 号

责任编辑：夏毓彦

封面设计：王 翔

责任校对：闫秀华

责任印制：沈 露

出版发行：清华大学出版社

网 址：<http://www.tup.com.cn>，<http://www.wqbook.com>

地 址：北京清华大学学研大厦 A 座 邮 编：100084

社总机：010-62770175 邮 购：010-62786544

投稿与读者服务：010-62776969，[c-service@tup.tsinghua.edu.cn](mailto:c-service@tup.tsinghua.edu.cn)

质量反馈：010-62772015，[zhiliang@tup.tsinghua.edu.cn](mailto:zhiliang@tup.tsinghua.edu.cn)

印 装 者：清华大学印刷厂

经 销：全国新华书店

开 本：190mm×260mm 印 张：16

字 数：410 千字

版 次：2019 年 1 月第 1 版

印 次：2019 年 1 月第 1 次印刷

定 价：59.00 元



产品编号：078579-01

# 作者简介

申思维(<http://siwei.me>), 软件行业老兵。stackoverflow.com 分数 13k, 排名前 3%。2001~2005 年本科就读于华南理工大学计算机学院软件技术专业, 毕业后在北京工作, 经历了人力外派、私企、中等规模外包公司、顶级外企和国内互联网知名企业。

- 2006 年服务于人力外派公司。
- 2007~2008 年服务于必联北京。
- 2009 年服务于同方鼎欣。
- 2010~2012 年服务于摩托罗拉 (移动)。
- 2013~2014 年服务于优酷。
- 2014~2015 年担任乐单软件联合创始人兼 CTO。
- 2015 年担任悦装网联合创始人兼 CTO、匠优家合伙人兼 CTO。
- 2016 年至今担任明创软件创始人兼 CTO。

从事软件行业十多年, 具有深厚的全栈开发功力。

- 后端技术背景: Java、Ruby on Rails、Python、全栈运维 (DevOps)。
- H5 端技术背景: Vuejs、jQuery、CSS。
- 移动端技术背景: Android、React Native。

熟悉互联网运维, 擅长技术团队的搭建、管理和对人员的培养。录制过 Ruby、Rails、Git、自动化部署、Vim 和程序员职业规划等教程和视频。对于国内的软件现状理解深刻、对于行业前景和职业规划有着非常独到的见解。目前重点专注区块链技术的应用和软件行业的相关培训。作者微信二维码如下:



# 前言

本书是根据笔者公司过去两年多的实际项目经验编写的。

笔者从 2016 年经营自己的软件公司，到 2018 年 6 月，做了近三十个项目。这些项目中，对于手机端的 Webpack 呼声最高，大部分项目都要求在手机端使用 Webpack 打包。

在使用 Vue.js 之前，笔者考察过 Angular（包括 1.x、2.x 版本）、React、Meteor，这几个框架都适合做快速开发。要么是学习曲线陡峭，概念复杂，把简单的事情复杂化（如 Angular），要么就是编码风格不好，前后端代码混写在一起（如 React、Meteor）。而 Vue.js 是当时在 stackoverflow 等国外技术站点上被一致看好的技术。

第一次使用 Vue.js 1.x 是在 2016 年 4 月，我们发现 Vue.js 入门特别快，稍微有一定 Webpack 开发经验的程序员，在一周内就可以上手做项目，认真学习的话，一个月就可以达到熟练水平（快速的开发项目），两三个月就可以达到高级水平（熟练使用 Vuex，自己写 Component 等）。这么快的上手速度，在其他语言中是不可想象的。根据笔者的实际体会，使用 Angular 入门仅需要一个月，使用 React 入门速度也没有这么快。总之，越是简洁的框架，就越好学。

后来，笔者在项目中使用它，一发不可收拾，只要是 H5 项目，就可以很好地用起来。快速开发，快速迭代，性能“杠杠”的。而且，额外的好处就是可以非常好地与 Native App 的开发结合。在 iOS 上可以做到完美呈现，让人无法分辨哪个页面是原生，哪个页面是 H5 制作的。

## 学习目标

本书是笔者公司的员工培训教程，可以在极短的时间内（如一周）上手 Vue.js 项目。让读者：

- 看得懂代码。
- 可以写一些基本的功能。
- 可以调试和部署。

这就算入门 H5 开发了。

## 使用说明

如果把文档分成两类：

- guide，教程型文档。
- API，接口型文档。

本文档就是入门的教程式文档，在线查看地址为 [http://vue\\_book.siwei.me/](http://vue_book.siwei.me/)。

本书使用 gitbook 写成，可以自行编译（安装环境和编译命令可参考 <http://siwei.me/blog/posts/gitbook-gitbook>）。

本书中出现的命令行统一以\$作为开始。例如：

```
$ npm install vue-cli
```

对命令行不熟悉的读者，在写命令时跳过前面的\$即可。

## 源代码下载

本书中所有的示例源代码，都可以在 [https://github.com/sg552/happy\\_book\\_vuejs](https://github.com/sg552/happy_book_vuejs) 上找到。

## 版本说明

截至 2018 年 6 月底，Vue.js 的版本是 2.5.16。本书中的大部分示例都是在该版本下演示的。

如果您是一位没有任何工作经验的新人，在 Windows 环境下，建议使用 sublime（免费）+git bash（免费），就可以运行本书中的所有例子了。如果您是一名有工作经验的老鸟，那么 Linux、Mac 则是非常好的选择。

那么，我们就开始一段令人兴奋的学习历程吧。

编者

2018 年 9 月

# 目 录

第 1 章	Vue.js 概述 .....	1
1.1	单页应用的出现 .....	1
1.2	为什么要使用 Vue.js .....	2
1.2.1	单页应用 .....	2
1.2.2	知名的单页应用 (SPA) 框架对比 .....	5
1.2.3	被腾讯和阿里巴巴所青睐 .....	9
1.2.4	用到 Vue.js 的项目 .....	9
第 2 章	原生的 Vue.js .....	10
2.1	极速入门 .....	10
2.2	实际项目 .....	11
2.2.1	运行整个项目 .....	12
2.2.2	HTML 代码的<head>部分 .....	18
2.2.3	HTML 代码的<body>部分 .....	19
2.2.4	js 代码部分 .....	21
2.2.5	小结 .....	25
第 3 章	Webpack+Vue.js 开发准备 .....	26
3.1	学习过程 .....	26
3.1.1	可以跳过的章节 .....	26
3.1.2	简写说明 .....	26
3.1.3	本书例子文件下载 .....	27
3.2	NVM、NPM 与 Node .....	27
3.2.1	Windows 下的安装 .....	28
3.2.2	Linux、Mac 下的安装 .....	31
3.2.3	运行 .....	31
3.2.4	使用 NVM 安装或管理 node 版本 .....	32
3.2.5	删除 NVM .....	33
3.2.6	加快 NVM 和 NPM 的下载速度 .....	33
3.3	Git 在 Windows 下的使用 .....	34
3.3.1	为什么要使用 Git Bash .....	34

3.3.2	安装 git 客户端	35
3.3.3	使用 Git Bash	40
3.4	Webpack	41
3.4.1	Webpack 功能	42
3.4.2	Webpack 安装与使用	43
3.5	开发环境的搭建	44
3.5.1	安装 Vue.js	44
3.5.2	运行 vue	44
3.6	Webpack 下的 Vue.js 项目文件结构	45
3.6.1	build 文件夹	46
3.6.2	config 文件夹	46
3.6.3	dist 文件夹	47
3.6.4	node_modules 文件夹	47
3.6.5	src 文件夹	49
<b>第 4 章</b>	<b>Webpack+Vue.js 实战</b>	<b>50</b>
4.1	创建一个页面	50
4.1.1	新建路由	50
4.1.2	创建一个新的 Component	51
4.1.3	为页面添加样式	52
4.1.4	定义并显示变量	53
4.2	Vue.js 中的 ECMAScript	55
4.2.1	let、var、常量与全局变量	55
4.2.2	导入代码: import	56
4.2.3	方便其他代码使用自己: export default {...}	56
4.2.4	ES 中的简写	57
4.2.5	箭头函数=>	57
4.2.6	hash 中同名的 key、value 的简写	58
4.2.7	分号可以省略	58
4.2.8	解构赋值	58
4.3	Vue.js 渲染页面的过程和原理	59
4.3.1	渲染过程 1: js 入口文件	59
4.3.2	渲染过程 2: 静态的 HTML 页面 (index.html)	61
4.3.3	渲染过程 3: main.js 中的 Vue 定义	62
4.3.4	渲染原理与实例	63
4.4	视图中的渲染	64
4.4.1	渲染某个变量	64
4.4.2	方法的声明和调用	65



4.4.3	事件处理: v-on.....	66
4.5	视图中的 Directive (指令) .....	67
4.5.1	前提: 在 directive 中使用表达式 (Expression) .....	67
4.5.2	循环: v-for .....	67
4.5.3	判断: v-if .....	69
4.5.4	v-if 与 v-for 的优先级 .....	70
4.5.5	v-bind.....	72
4.5.6	v-on.....	73
4.5.7	v-model 与双向绑定.....	75
4.6	发送 http 请求.....	77
4.6.1	调用 http 请求.....	78
4.6.2	远程接口的格式.....	80
4.6.3	设置 Vue.js 开发服务器的代理.....	81
4.6.4	打开页面, 查看 http 请求.....	83
4.6.5	把结果渲染到页面中.....	84
4.6.6	如何发起 post 请求.....	85
4.7	不同页面间的参数传递.....	86
4.7.1	回顾: 现有的接口.....	86
4.7.2	显示博客详情页.....	87
4.7.3	新增路由.....	88
4.7.4	修改博客列表页的跳转方式 1: 使用事件.....	89
4.7.5	修改博客列表页的跳转方式 2: 使用 v-link.....	91
4.8	路由.....	92
4.8.1	基本用法.....	92
4.8.2	跳转到某个路由时带上参数.....	93
4.8.3	根据路由获取参数.....	94
4.9	使用样式.....	94
4.10	双向绑定.....	96
4.11	表单项目的绑定.....	99
4.12	表单的提交.....	102
4.13	Component 组件.....	105
4.13.1	如何查看文档.....	105
4.13.2	Component 的重要作用: 重用代码.....	106
4.13.3	组件的创建.....	106
4.13.4	向组件中传递参数.....	108
4.13.5	脱离 Webpack, 在原生 Vue.js 中创建 component.....	110

<b>第 5 章 运维和发布 Vue.js 项目</b> .....	112
5.1 打包和部署.....	112
5.1.1 打包.....	112
5.1.2 部署.....	114
5.2 解决域名问题与跨域问题.....	117
5.2.1 域名 404 问题.....	118
5.2.2 跨域问题.....	119
5.2.3 解决域名问题和跨域问题.....	120
5.3 如何 Debug.....	122
5.3.1 时刻留意本地开发服务器.....	122
5.3.2 看 developer tools 提出的日志.....	122
5.3.3 查看页面给出的错误提示.....	123
5.4 基本命令.....	125
5.4.1 建立新项目.....	125
5.4.2 安装所有的第三方包.....	125
5.4.3 在本地运行.....	126
5.4.4 打包编译.....	127
<b>第 6 章 进阶知识</b> .....	128
6.1 js 的作用域与 this.....	128
6.1.1 作用域.....	128
6.1.2 this.....	130
6.1.3 实战经验.....	131
6.2 Mixin.....	133
6.3 使用 Computed Properties（计算得到的属性）和 watchers（监听器）.....	135
6.3.1 典型例子.....	135
6.3.2 Computed Properties 与 普通方法的区别.....	136
6.3.3 watched property.....	137
6.3.4 Computed Property 的 setter（赋值函数）.....	140
6.4 Component（组件）进阶.....	141
6.4.1 实际项目中的 Component.....	142
6.4.2 Prop.....	144
6.4.3 Attribute.....	146
6.5 Slot.....	146
6.5.1 普通的 Slot.....	147
6.5.2 named slot.....	148
6.5.3 slot 的默认值.....	149
6.6 Vuex.....	150

6.6.1	正常使用的顺序 .....	150
6.6.2	Computed 属性 .....	154
6.6.3	Vuex 原理图 .....	155
6.7	Vue.js 的生命周期 .....	156
6.8	最佳实践 .....	157
6.9	Event Handler 事件处理 .....	158
6.9.1	支持的 Event .....	158
6.9.2	使用 v-on 进行事件绑定 .....	159
6.10	与 CSS 预处理器结合使用 .....	168
6.10.1	SCSS .....	168
6.10.2	LESS .....	169
6.10.3	SASS .....	170
6.10.4	在 Vue.js 中使用 CSS 预编译器 .....	171
6.11	自定义 Directive .....	172
6.11.1	例子 .....	172
6.11.2	自定义 Directive 的命名方法 .....	173
6.11.3	钩子方法 (Hook Functions) .....	174
6.11.4	自定义 Directive 可以接收到的参数 .....	174
6.11.5	实战经验 .....	175
<b>第 7 章</b>	<b>实战周边及相关工具 .....</b>	<b>176</b>
7.1	微信支付 .....	176
7.2	Hybrid App: 混合式 App .....	177
7.3	安装 Vue.js 的开发工具: Vue.js devtool .....	178
7.4	如何阅读官方文档 .....	181
<b>第 8 章</b>	<b>实战项目 .....</b>	<b>183</b>
8.1	准备 1: 文字需求 .....	183
8.2	准备 2: 需求原型图 .....	186
8.2.1	明确前端页面 .....	186
8.2.2	如何画原型图 .....	186
8.2.3	首页 .....	186
8.2.4	商品列表页 .....	187
8.2.5	商品详情页 .....	187
8.2.6	购物车页面 .....	188
8.2.7	支付页面 .....	188
8.2.8	我的页面 .....	189
8.2.9	我的订单列表页面 .....	189

8.2.10	总结 .....	190
8.3	准备 3: 微信的相关账号和开发者工具 .....	190
8.3.1	微信相关账号的申请 .....	190
8.3.2	微信开发者工具 .....	190
8.4	项目的搭建 .....	192
8.5	用户的注册和微信授权 .....	193
8.6	登录状态的保持 .....	202
8.7	首页轮播图 .....	203
8.8	底部 Tab .....	213
8.9	商品列表页 .....	217
8.10	商品详情页 .....	219
8.11	购物车 .....	225
8.13	微信支付 .....	233
8.14	回顾 .....	244

# 第 1 章

## ◀ Vue.js 概述 ▶

### 1.1 单页应用的出现

随着移动电话的普及和微信的流行，很多的 Wap (H5) 应用也随之出现了，如微店、网站各个 App 中包含的 H5 页面。

手机的硬件特点有：

- 硬件设备差。同主频的手机 CPU 性能往往是台式机的十分之一（手机的供电与台式机设备相差很远）。
- 网络速度慢。4G 网络在很多时候下载速度只有几百 KB，打开一个微信中的网页可能也要很久。

因此，使用传统的 Webpack 技术开发的网页，在手机端的表现往往特别差。传统技术的特点是：

- 单击某个链接/按钮，或者提交表单后，Webpack 页面整体刷新。
- js/css 的请求往往很多，过百是很常见的事情。

每次页面整体刷新，都要导致浏览器重新加载对应的内容，特别“卡顿”。另外，加载的内容也很多。很多传统页面的 css/js 多达上百个，每次打开页面都需要发送上百次请求。如果页面中包含 websocket 等内容，打开速度就会更慢。

苹果的机器表现还好，iOS 设备打开 Web 页面速度很快，Android 设备则大部分都很慢。这个是由手机设备操作系统、软件及智能硬件决定的。

单页应用 (Single Page App, SPA) 体现出了其强大的优势。

- 页面是局部刷新的，响应速度快，不需要每次加载所有的 js/css。
- 前后端分离，前端（手机端）不受（服务器端的）开发语言的限制。

越来越多的 App 采用 SPA 的架构。如果你的项目要用在 H5 上，那么一定要使用单页应用框架，如 Angular、React、Vue.js 都是很好的框架。

我们在公司实际项目中，都使用 Vue.js，效果非常好。开发速度快，维护效率高。

因为本文与官方文档不同，是根据实际项目经验，以培养新人的角度来写的，所以会有以下特点。

- 很少使用的技术略过。
- 只讲解常见的知识。
- 在章节上按照入门的难易度程度从简单到复杂。

## 1.2 为什么要使用 Vue.js

在本章中，我们会从多个角度思考这个问题。

### 1.2.1 单页应用

Web 的应用分两类：传统 Web 页面应用和单页应用（Single Page App）。

#### 1. 传统 Web 页面

传统 Web 页面就是打开浏览器，整个页面都会打开的应用。例如，笔者的个人网站 <http://siwei.me> 就是一个典型的“传统 Web 应用”，每次单击其中任意一个链接，都会引起页面的整个刷新，如图 1-1 所示。

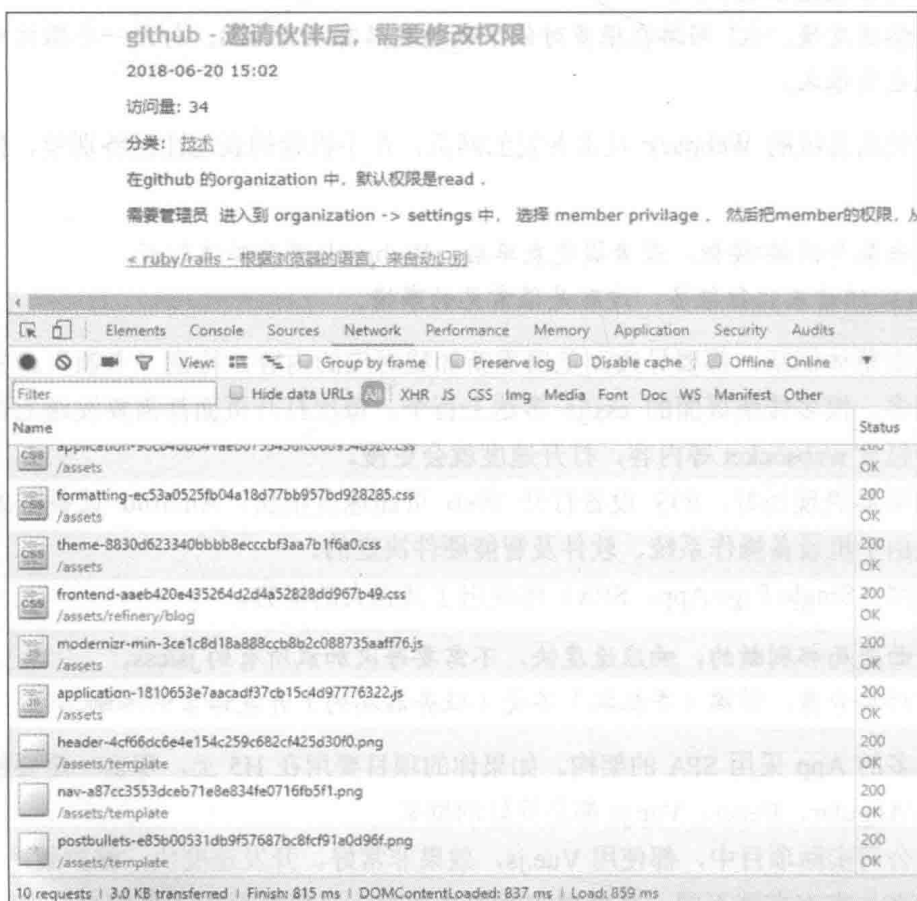


图 1-1 个人网站

从图 1-1 中可以看出，传统的页面每次打开，都要把页面中的.js、.css、图片文件、html 文件等资源加载一遍。在图 1-1 的左下角可以看到，本次共加载了 10 个请求（4 个 css，2 个 js，3 个 png 图片及 1 个 html 文件），耗时 0.837s。这个在 PC 端可以，但是在手机端就会特别慢，特别是在安卓手机上。

传统页面的特点就是下面任何一个操作，都会引起浏览器对于整个页面的刷新：

- 单击链接。
- 提交表单。
- 触发 `location.href='...'` 这样的 js 代码。

我们来看一个传统 Web 页面的例子。

```
<html>
<head>
  <script src="my.js"></script>
  <style src="my.css"></style>
</head>
<body>
  <img src='my.jpg' />
  <p> 你好! 传统 Web 页面! </p>
</body>
</html>
```

每个浏览器都会从第一行解析到最后一行，然后继续加载 `my.js`、`my.css`、`my.jpg` 这三个外部资源。

其实很好理解，这个就是大家最初想象的 Web 页面的打开方式。

## 2. 单页应用 ( Single Page App )

单页应用，确切的诞生时间不详，可以肯定的是这个概念 2003 年就在论坛上被人讨论了。在 2002 年 4 月，诞生了一个网站：<http://slashdotslash.com>，就使用了这种思想。

单页应用的精髓是点/单击任何链接，都不会引起页面的整体刷新，只会通过 JavaScript 替换页面的局部内容。

## 3. Ajax 和 XML

说到这里，就不得不提到另一个概念：Ajax (Asynchronous JavaScript)，中文可以称之为“js 的异步请求”，国内统一称为 Ajax。

Ajax 的概念是每次打开新的网页时，不要让页面整体刷新，而是由 js 发起一个“HTTP 异步请求”，这个“异步请求”的特点就是不让当前的网页“卡”死。

用户可以一边上下滚动页面，播放视频一边等待这个请求返回数据。结果被正常返回

后，由 js 控制刷新页面的局部内容。

这样做的好处是：

- (1) 大大节省了页面的整体加载时间。各种.js、.css 等资源文件加载一次就够了。
- (2) 节省了带宽。
- (3) 同时减轻了客户端和服务端的负担。

在智能手机和 App 应用（特别是微信）流行起来之后，大量的网页都需要在手机端打开，Ajax 的优势就体现的淋漓尽致。

虽然 Ajax 的名称本意是“异步 js 与 XML”，但是现在在服务器端返回的数据中几乎都使用 JSON，而抛弃了 XML。

在 2005 年，国内的程序员论坛开始提及 Web 2.0，其中 Ajax 技术被人重视。到了 2006 年初，可以说 Ajax 是前端程序员的加薪利器。市面上的所有招聘“前端 Web 程序员”的职位描述中都认为 Ajax 是重要的加分项。

可惜当时 jQuery 在国内不是很普及，Prototype 也没有流行起来。笔者与北京软件圈子里的各大公司的同行们交流时，发现大家用的都是“原生的 JavaScript Ajax”，这种不借助任何第三方框架的代码写起来非常臃肿、累人，而且考虑到浏览器的兼容问题，开发起来也很让人头疼。

例如，当时的代码往往是这样的：

```
var xmlhttp = false;
/*@cc_on @*/
/*@if (@_jscript_version >= 5)
try {
    xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
} catch (e) {
    try {
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    } catch (e2) {
        xmlhttp = false;
    }
}
@end @*/
if (!xmlhttp && typeof XMLHttpRequest != 'undefined') {
    xmlhttp = new XMLHttpRequest();
}
```

上面的代码仅仅是为了兼容各种浏览器。实际上，后面还有几十行的冗余代码，之后才是正常的业务逻辑代码。

到 2008 年，国内开始流行 Prototype、jQuery 之后，发起一个 Ajax 请求的代码精简成几行：



```
jQuery.get('http://some_url?para=1', function(data) {  
    // 正常代码  
})
```

那时候开始, Ajax 在国内变得越来越普及。

#### 4. Angular

第一个单页应用 (SPA) 的知名框架应该是 Angular, 由 Google 在 2010 年 10 月推出。当时的 Gmail、Google Map 等应用对于 Ajax 技术运用到了极致。而 Angular 框架一经推出, 立刻引燃了单页应用这个概念。尽管后来各种 SPA 框架层出不穷, 在 2015 年之前, Angular 稳坐 SPA 的头把交椅。

#### 5. 当下的 SPA 技术趋势

SPA 框架已经成为了项目开发必不可少的内容, 只要有移动端开发, 就会面临以下两个选择。

- 做成原生 App。
- 做成 SPA H5。

无论是 iOS 端还是 Android 端, 都对 SPA 青睐有加。

(1) 打开页面速度特别快。打开传统页面, 手机端往往需要几秒, 而 SPA 则在 0.x 秒内。

(2) 耗费的资源更少。因为每次移动端只请求接口数据和必要的图片资源。

(3) 对于点/单击等操作响应更快。对于传统页面, 手机端的浏览器在操作时, 点击按钮会有 0.1s 的卡顿, 而使用 SPA 则不会有卡顿的感觉。

(4) 可以保存浏览的历史和状态。不是每一个 Ajax 框架都有这个功能。例如, QQ 邮箱, 虽然也是页面的局部刷新, 但是每次打开不同的邮件时, 浏览器的网址不会变化。而在所有的 SPA 框架中, 都会有专门处理这个问题的模块, 叫做 router (路由)。

例如:

`http://mail.my.com/#/mail_from_boss_on_0620` 对应老板在 6 月 20 日发来的邮件。

`http://mail.my.com/#/mail_from_boss_on_0622` 对应老板在 6 月 22 日发来的邮件。

在 2011 和 2012 年, 各种 SPA 框架出现了井喷的趋势, 包括 Backbone、Ember.js 等上百个不同的框架。近几年, 比较流行的框架是 Angular、React 和 Vue.js。

### 1.2.2 知名的单页应用 (SPA) 框架对比

在学习 Vue.js 之前, 我们要知道为什么学习它。

目前市面上比较知名的单页应用 SPA 框架是 Vue.js、React、Angular, 我们依次来了解一下。