

## 6周学习计划，攻克JavaScript难关 (React/Redux/ES6 etc.)



余博伦 · 1 个月前

原文链接：[A Study Plan To Cure JavaScript Fatigue](#)

作者：[Sacha Greif](#)

和大家一样，最近我也看了Jose Aguinaga写的[How it feels to learn JavaScript in 2016](#)。

显然这篇文章击中了人们的痛处。它在[Hacker News](#)上排了不止一次第一。同样也是[/r/javascript](#)上最火的一篇，在Medium上也有超过10k的推荐。

这并不能算是哗众取宠：我很早以前就了解到，JS的生态圈确实很令人困惑。事实上，我开展[State Of JavaScript](#)调查正是为了了解真正受欢迎的JS库，去其糟粕，取其精华。

今天，我不只是简单地陈述JS的发展现状，我将会向你展现一个十分具体，一步一步掌握JS知识体系的学习计划。

### 目标人群

这份学习计划写给：

- 你熟悉例如变量方法一类的基本的编程概念；
- 你之前有了解过PHP/Python一类的后端语言，也使用过jQuery一类的库；
- 你想要更深入的了解前端开发，却迷失在无数的框架和类库当中不知道何去何从。

# 涵盖内容

- 当前JS应用的构成
- 为什么仅使用jQuery是不够的
- 为什么React是最合适的选择
- 为什么你没必要一开始就“全面掌握JavaScript”
- 如何学习ES6语法
- 为什么以及如何学习Redux
- GraphQL是个什么玩意儿
- 在这之后该学些什么

## 学习资料说明

免责声明：文章会提到一些来自[Wes Bos](#)的教程链接，仅作参考，并不是推广广告。

如果你想要找到别的相关资料，Mark Erikson在Github上有一个收集列表[React, ES6, and Redux](#)

## 此JavaScript非彼JavaScript

首先明确一下这份学习计划主要目的。随便你一搜“学习JavaScript”或者“JavaScript学习计划”，就能够找到无数有关学习**JavaScript语言**的教程资源。

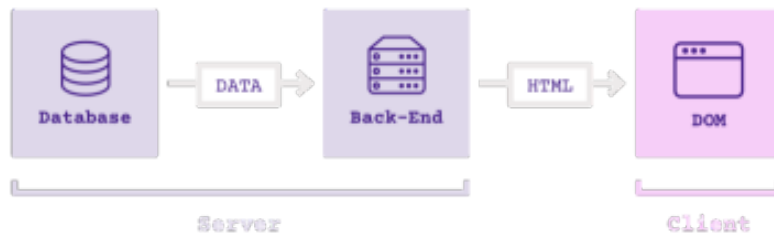
但这其实属于比较简单的部分。对一门语言的学习理解可以挖的很深，可是事实上大部分Web应用的代码是很简单的。换句话讲，编写Web应用所需的80%的知识在你看的JavaScript教程书籍的前几章就都涵盖了。

真正的困难在于掌握包含了数不胜数的框架和库的JavaScript的生态体系。这篇学习计划主要关注的就是这部分内容。

## JavaScript应用的框架结构

为了理解当前JavaScript应用的复杂性，我们首先需要了解他们是如何运作的。

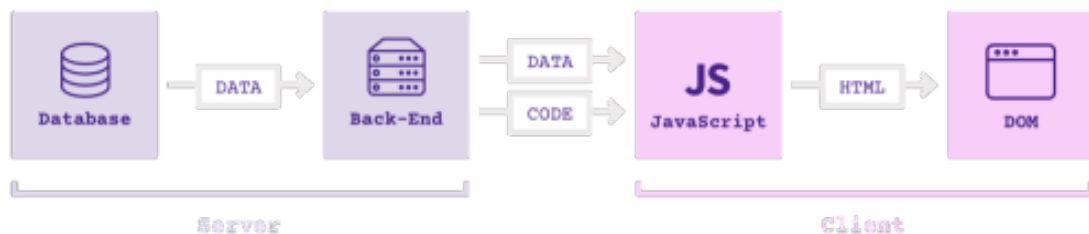
首先我们看一下2008年标准的Web应用的构成：



1. 数据库为后端提供数据（后端可以是PHP或Rails等）；
2. 后端解析数据并输出HTML；
3. HTML被发送到浏览器，以网页的形式展示。

现在这类应用也有一些客户端的JS代码，用来增加交互（例如标签、模态窗口等），但本质上，浏览器仍是从服务器获取HTML的内容。

现在拿当前2016年的Web应用（也称作：单页面应用）与之比较：



发现其中的区别了么？并不是从服务器发送HTML内容，而是只返回数据，数据到HTML的渲染步骤都发生在客户端（同样也返回客户端如何根据会话进行反馈的代码）。

这么做有好有坏，首先好处是：

- 对于相同的某块内容，只发送数据要比发送HTML页面快。
- 客户端不需要刷新页面就可以进行交互改变内容（这也是为什么称其为单页面应用的原因）

缺陷是：

- 初次加载需要更久，因为“数据转换成HTML”的代码量会很大。
- 为了方便缓存和查询，你在客户端也需要有一块地方管理和存储数据。

比较脏的是：

- 恭喜你，现在客户端的技术栈和服务器端一样复杂啦。

## 客户端与服务器的范畴

既然有这么多缺陷我们为什么还要遭这份儿罪呢？就像以前那样使用PHP后端不好么？

那么，试想一下你开发一个计算器应用。如果用户想知道 $2+2$ 的结果，在服务器端计算再返回是非常傻的，因为浏览器完全有能力实现这种需求。

但另外一方面，如果你只是搭建博客一类的静态站点的话，在服务器端直接生成HTML内容就挺合适的。

而事实上，大部分的Web应用都介于这两者之间，问题就是我们到底采用何种架构。

关键是两种架构是无法过渡的：你不能开始写一个纯服务器端应用然后慢慢迁移到纯客户端。有些情况下，你将不得不停下来把所有的部分都重构，或者遗留下一堆无法维护的杂乱代码。



这也解释了你为什么不能在所有项目中“只用jQuery”。你可以把jQuery想象成万金油。房间里需要修修补补的时候它有奇效，而你如果滥用的话只会让一切看起来很糟糕。另外一方面，当前的JavaScript框架就好像3D打印技术革新一样：耗时更久，但结果更简洁可靠。

换句话说，掌握当前流行的JavaScript技术栈就是赌绝大多数的Web应用可能最终都将把服务器端客户端划分开来。你确实需要付出更多的学习成本。不过少壮不努力，老大只得徒伤悲。

## 第一周：JavaScript基础

除非你之前是个纯后端开发者，你多少都应该知道点JavaScript.即使你不懂，JavaScript的语法看起来也很像C，对于PHP或Java开发者来说应该不会太陌生。

要是你之前一点都不了解JavaScript也没关系。网上可以找到很多相关的免费教程让你快速入门。例如[Codecademy's JavaScript lessons](#) 就很不错。

- [JavaScript 教程](#)
- [JavaScript全栈教程](#)
- [JavaScript MDN](#)
- [JavaScript 标准参考教程](#)

## 第二周：开始学习React

掌握了JavaScript的基本语法之后，你多少明白些为什么JavaScript应用会如此复杂了，我们直接说点具体的，从哪里上手呢？

我想答案就应该是[React](#) .

# React

A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES

[Get Started](#)[Take the Tutorial](#)

## Declarative

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Declarative views make your code more predictable and easier to debug.

## Component-Based

Build encapsulated components that manage their own state, then compose them to make complex UIs.

Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.

## Learn Once, Write Anywhere

We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

React can also render on the server using Node and power mobile apps using [React Native](#).

React是Facebook开源的UI工具库。换句话讲，它主要用来处理数据到页面的渲染步骤（视图层）。

不要误会我：我推荐你选用React不是因为它是全世界最好的框架（这太过主观了），而是因为它确实非常受欢迎。

- React也许不是全世界最受欢迎的框架，但真的有很多人都喜欢它。
- React也许不是最轻量的框架，但也并不臃肿。
- React也许并不是最易上手的框架，但也挺友好的。
- React也许并不是最优雅的框架，但也够优雅了。

换句话讲，React也许不是所有情况下的最优选择，但它是最靠谱的。而其相信我，在你刚刚开始的时候，还是不要太特立独行的好。

学习React同样有助于你了解一些实用的概念。例如组件、应用状态、无状态方法等。这对你学习任何别的框架和库都是很有价值的。

最后，React有着当前最大的生态体系，许多包和库都能和它非常好地协同，同时它庞大的社区也可以让你比较轻松地在网上获取到帮助。

我个人推荐[React for Beginners](#) 这个教程。我自己当时就学了 this 教程，而且它最近刚刚更新了React的最佳实践。

- [React 入门实例教程](#)

- [React 教程](#)
- [一看就懂的ReactJs入门教程-精华版](#)
- [十分详细的React入门实例](#)
- [React 入门与最佳实践](#)

## 你需要先“全面掌握JavaScript”么？

假如你是个按部就班的同学，你可能希望先踏踏实实地牢固打好JavaScript的基础。

但对于大多数人来讲，这就好像为了学游泳去学人体解剖和流体力学一样。确实，这两门专业知识对游泳来说都至关重要，不过还是直接跳到池子里开始游爽啊！

这个问题没有对错，全在于你的学习方法。事实上，大部分的React基础教程都只需要很少一部分JavaScript知识，你只需要先掌握这部分内容就足够了。

这个道理同样适用于广义上的JavaScript知识体系。你并不需要担心不理解Webpack或Babel的输入输出。事实上，React推出的[命令行工具](#)可以让你完全不必担心配置就初始化好应用。

## 第三周：你的第一个React应用

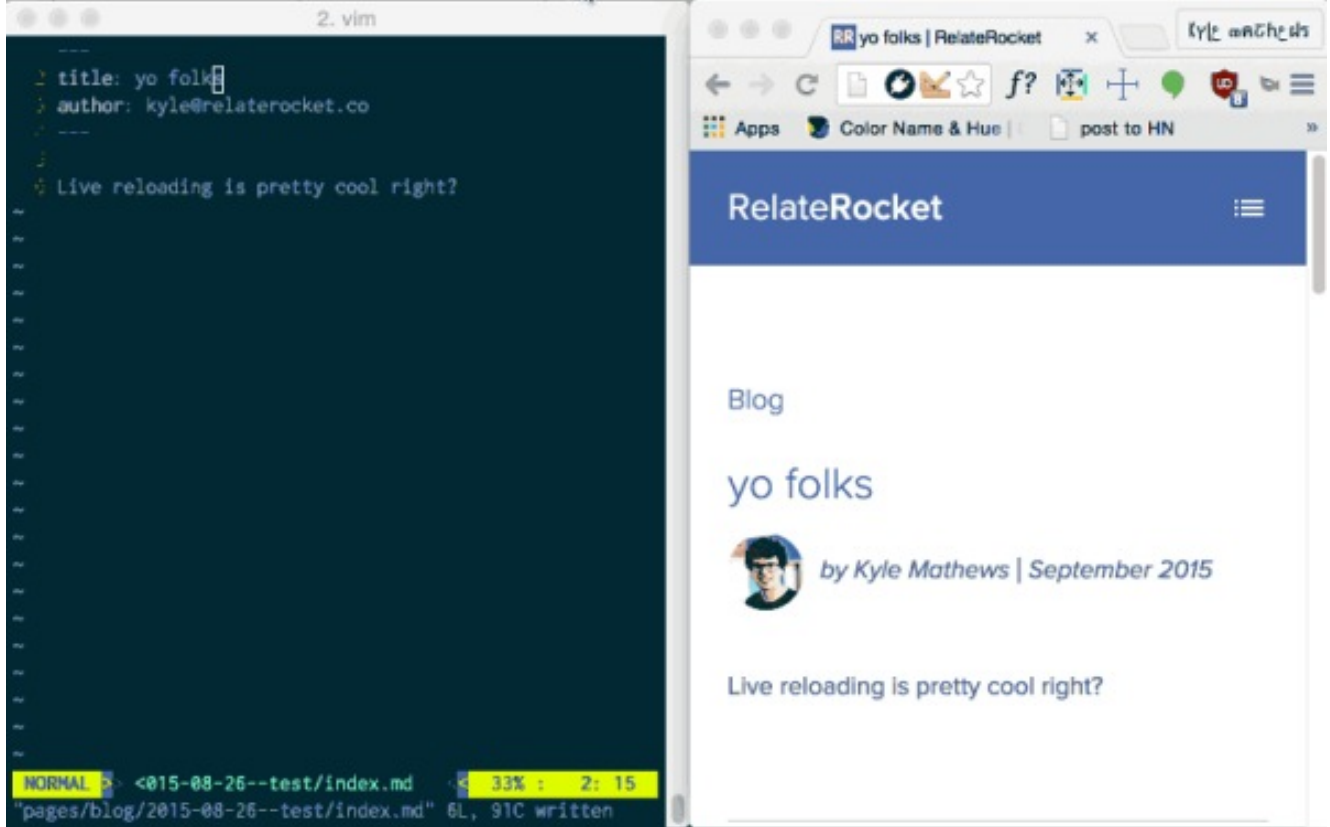
在你学完了React的教程之后，你可能面临和我当时相同的状况：

- 你把你刚学的内容可能已经忘了一半了。
- 你等不及要带着剩下的一半知识上手实践一下。

我坚信学习一个框架或一门语言的最好方法就是上手实践。而写一些个人项目则是实验新技术的最好选择。

一个个人开发项目可以简单到只是一个页面，也可以复杂到一个完整的Web应用。我感觉用新技术重新设计你的个人网站难度刚刚好，另外，你估计也很久没有更新过你个人站的架构了。

我之前提到过，用单页应用展示静态内容确实有些大材小用了，不过React有一款很棒的工具[Gatsby](#)，一个React架构的静态站点生成器，可以让你体验React的所有优点。



使用Gatsby入门React的好处有以下几条：

- 一个预先配置好的Webpack，这意味着你要省下很多麻烦（Webpack配置简直太反人类）。
- 根据你的目录结构自动生成路由。
- 所有的HTML都会在服务器端渲染，弥补了客户端渲染的不足。
- 静态站点也可以不必担心服务器端，并且可以轻松托管在[Github Pages](#)上。

[State Of JavaScript](#) 就是我用Gatsby开发的，不需要操心路由、构建工具配置、服务器端渲染等烦人的问题，为我节约了大量的时间。

## 第四周：熟悉ES6

在我学习React的过程中，我很快就发现我可以很轻松地复制粘贴代码示例，但剩下的很多仍然不懂。

尤其是不熟悉[ES6](#) 的语法：

- 箭头函数Arrow functions
- 对象解构Object destructuring
- 类Classes
- 展开操作符The spread operator



如果你也有相同的感受，那就是时候花些时间好好学学ES6了。[ES6 for Everybody](#) 就是一个很不错的教程。

你想要免费的教程也有，Nicolas Bevacqua的[Practical ES6](#) 就不错。

- [ECMAScript 6入门](#)

掌握学习ES6的一个比较好的实践方法是重构你在第三周写的代码，尽量都转换为更简洁的ES6写法。

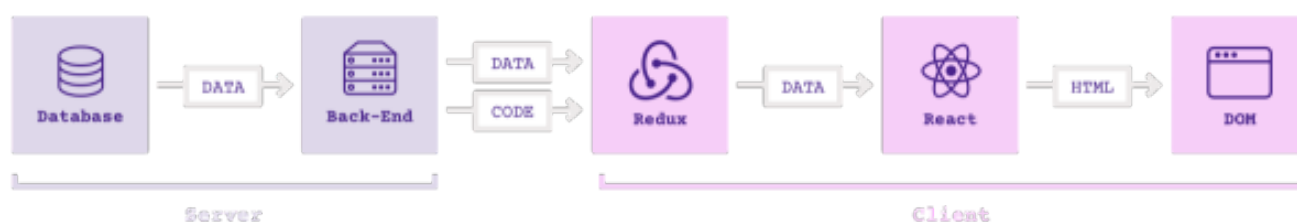
## 第五周：掌握状态管理

到现在这个阶段你应该已经能写一些静态内容的React前端了。

但真正的Web应用肯定不是静态的：他们需要从数据库一类的后端获取数据。

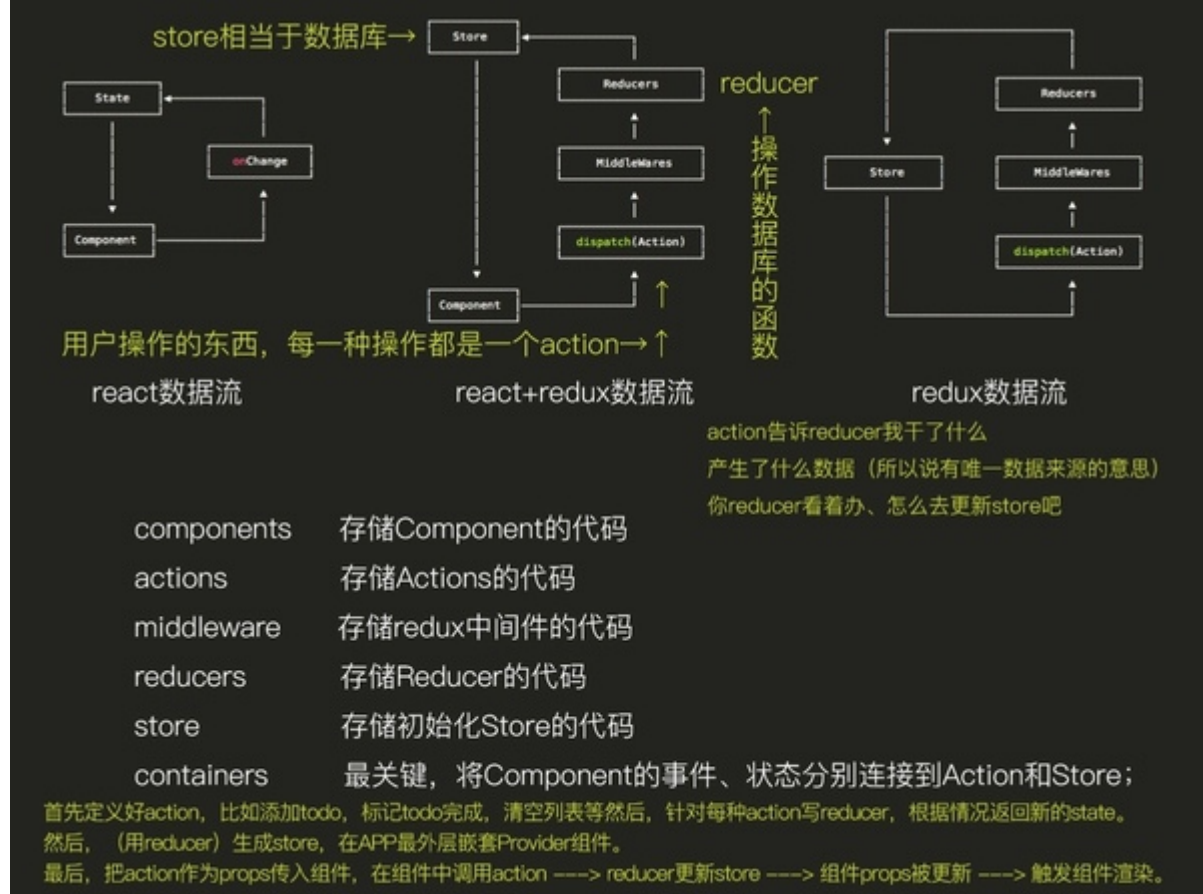
现在你可以用React给每个独立的组件传入数据，可应用一旦复杂了使用这种方式就会很凌乱。例如当两个组件需要展示同一组数据，或者需要相互通信的时候。

这就是引入**状态管理**概念的时候了。与在你的各个组件中一小块一小块存储状态(state)不同的是，你可以将所有的数据存储在一個全局store中，然后再分发给每个React组件：



在React阵营里，Redux是最受欢迎的状态管理库。Redux不仅能帮助你集中管理数据，同样可以将对数据的操作限制在一定规范内。

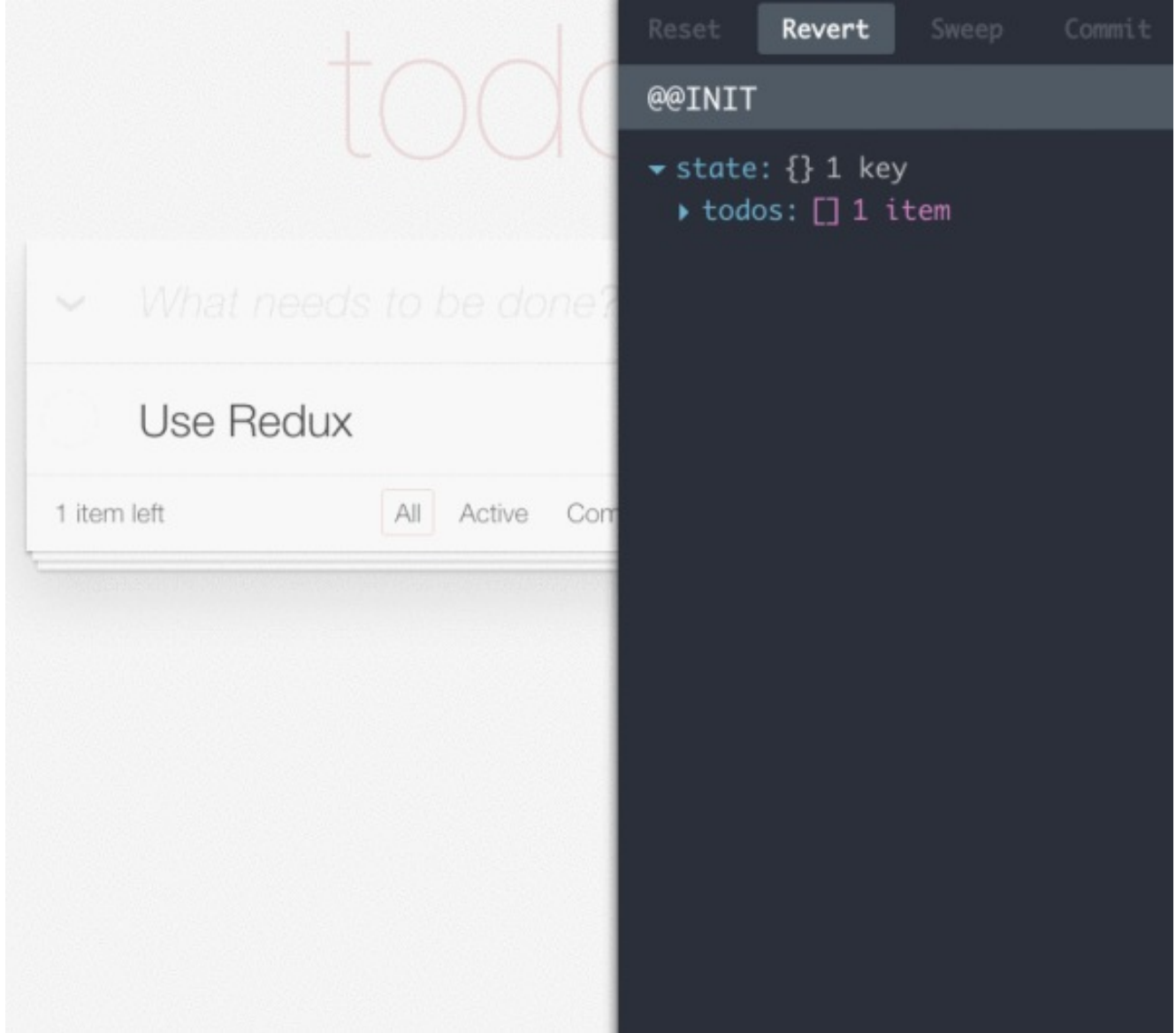




你可以将Redux想象成一个银行：你不能直接修改你账户的存款数字（来来来，让我在后面多加几颗零吧！）。而是需要填写存款表单，让银行出纳认证后来完成这个操作。

相似的，Redux也不允许你直接修改全局state的数据。而是通过向reducers传递actions来进行，reducer其实就是一个接收旧状态和操作返回新状态的方法。

这些看似多余的工作可以让你很好地维护管理你应用中的数据流。[Redux Devtools](#) 这样的工具可以很好地显示数据流的变化。



同样你也可以在Wes的网站上学习[Redux 教程](#)，这个是免费的。

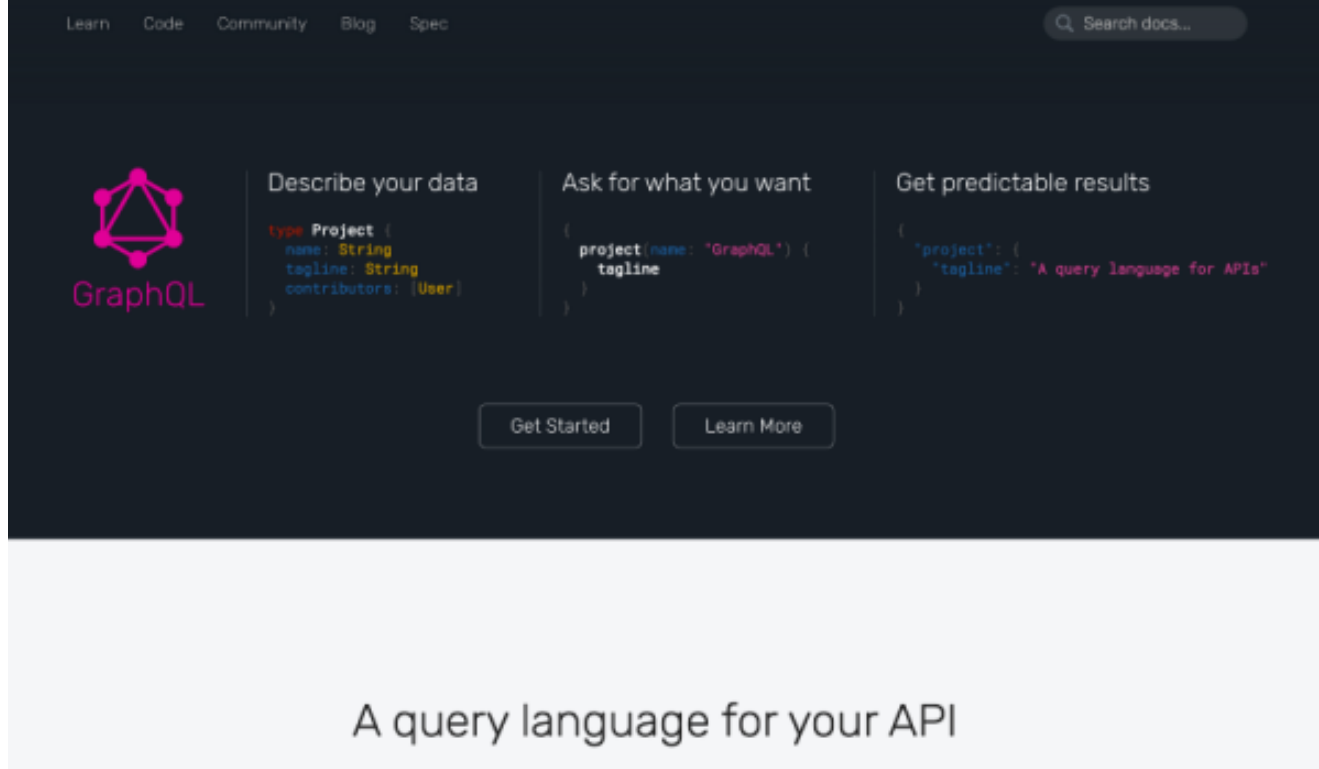
或者你喜欢视频教程，也可以参考Dan Abramov在[egghead.io](#)上的视频教程

- [Redux 入门教程](#)
- [Redux 入门与最佳实践](#)

## 最后一周：使用GraphQL构造APIs

到目前为止，我们谈论的大部分内容都在客户端，这只是整个应用的一半而已。即便你现在不需要完全了解Node的生态体系，你也需要了解对任何Web应用都至关重要的一点：数据是如何从服务器获取并传到客户端的。

同样也是由Facebook开源的项目[GraphQL](#)，它可以作为传统REST APIs的替代解决方案。



不同于REST API根据你预先定义的数据集（例如 `/api/posts`, `/api/comments`, etc.）分发出不同的REST路径。GraphQL可以让你只通过一个数据端像操作数据库一样按需查询数据。

想象成一个人分别多次去生食店、面包店、果树店与他带着一个购物清单去的区别。

这种新的策略在你需要请求多组数据源的时候非常有意义。就像我们上面举的购物清单例子，你只需要一次请求就可以获取所有的数据。

GraphQL在过去的一年里火了起来，很多例如Gatsby的项目都开始打算使用它。

GraphQL本身只是一项协议，它最好的实现是能和Redux非常好地协同的[Apollo](#) 这个库。现在网上的相关教程确实比较少，不过[Apollo 官方文档](#) 已经能让你很好地了解它了。

## 除了React之外呢

我推荐你从学习React生态体系开始因为选它确实很靠谱。但并不意味着只有这一种可靠的前端技术栈。我这里还有两个别的推荐：

### Vue

[Vue](#) 是新近火起来的一个框架，很多百度阿里一类的大公司也都开始使用了。它也是PHP框架Laravel的官方前端实现。



# The Progressive JavaScript Framework

[GET STARTED](#)[GITHUB](#)

## Approachable

Already know HTML, CSS and JavaScript? Read the guide and start building things in no time!

## Versatile

Simple, minimal core with an incrementally adoptable stack that can handle apps of any scale.

## Performant

15kb min+gzip Runtime  
Blazing Fast Virtual DOM  
Minimal Optimization Efforts

PROUDLY SPONSORED BY

THE  
DIFFERENCE  
ENGINE

相比React，它的主要卖点是：

- 官方提供维护的路由和状态管理类库
- 专注于性能表现
- 更低的学习成本，可以直接使用HTML模板而不是JSX
- 更少的模板代码

React更强大的地方在于它庞大的生态体系，例如[React Native](#) 一类的实现。不过Vue也在越来越壮大。

- [新手向：Vue 2.0 的建议学习顺序](#)

## Elm

如果说Vue还和React比较类似的话，[Elm](#) 是更加前卫的一种实现。Elm不仅是一个框架，更是一种建立在JavaScript之上的语言，类似CoffeeScript/TypeScript等。

它有很多优点，例如性能提升，语义化的版本，没有运行异常等。

在[State Of JavaScript](#) 调查中，有84%使用过它的开发者都很满意。

## 接下来学什么？

相信在学完上述的内容之后你已经熟练掌握了React的前端技术栈了。希望你在实际开发中也能很好地应用它。

但这并不能证明你已经精通了JavaScript！这仅仅是掌握整个JavaScript技术体系的开始而已。这里还有一些你可能感兴趣的内容（内容本来也是英文所以就不翻译啦）：

- JavaScript on the server (Node, [Express ...](#))
- JavaScript testing ([Jest](#) , [Enzyme ...](#))
- Build tools ([Webpack ...](#))
- Type systems ([TypeScript](#) , [Flow ...](#))
- Dealing with CSS in your JavaScript apps ([CSS Modules](#) , [Styled Components ...](#))
- JavaScript for mobile apps ([React Native ...](#))
- JavaScript for desktop apps ([Electron ...](#))

一篇文章远远不足以介绍所有这些内容。万事开头难，不过你要对自己有信心。等到你了解了JS的各部分实现是如何协同之后，接下来要做的也不过就是每个月都学习点火起来的新技术而已。

有任何问题以及好的意见或建议欢迎在评论区参与讨论。

「好好学习，天天向上」

赞赏

7 人赞赏





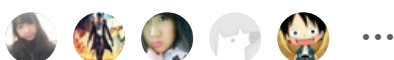
JavaScript

前端开发

React

 734

 分享  举报



文章被以下专栏收录



从零学习前端开发

良心干货、暖心教程、真心分享

[进入专栏](#)



写下你的评论



张传峰

最近正好到了这个迷茫的阶段，这篇简直雪中送碳！男神请翻我牌！

1 个月前

3 赞



明日天涯

迷糊糊的自学居然对上本篇计划。。

1 个月前

1 赞

以上为精选评论



没有翅膀的猪

先马克下

1 个月前



动机在未来

get

1 个月前



书香墨剑

写的很中肯

1 个月前



Celery Liu

学起来

1 个月前



话梅

我学了三个月的东西被你一周学完了

1 个月前

3 赞

刘瑜



可以的 先mark下

1 个月前



atom 回复 话梅

[查看对话](#)

6周，不是6天...

1 个月前



周小得

为啥angularjs连提都不提。。。。

1 个月前

4 赞

1

2

3

4

...

8

[下一页](#)

## 推荐阅读

### 你的第一门编程语言应该学什么？[javascript : answer](#)

原文链接：What programming language should you learn first? [javascript : answer](#)作者：Quincy Larson (FreeCodeCamp 创始人)... [查看全文](#) >

余博伦 · 1 个月前

发表于 [从零学习前端开发](#)

### 在2016年如何学习JavaScript？

原文链接 Want to learn JavaScript in 2016?作者：Vincent O译者：相信有不少人已经读过在 2016 年学 JavaScript 是一种什么样的体验？这篇... [查看全文](#) >

余博伦 · 2 个月前

发表于 [从零学习前端开发](#)

### 为什么日本街头净是白色的车

在日本生活，如果你多看两眼路上的车子，就会发现一个与其他国家不太一样的特征：白色的车子要比别的颜色的车多。虽然不是每个人都会注意到，... [查看全文](#) >

赤坂 · 14 天前 · [编辑精选](#)

发表于 [键盘车神与女司机](#)



## 地下区域自然光照的秘密

想象一下当你下班回家时，在一个漆黑沉闷的地铁站等候地铁。这时如果不经意地瞥见一缕难得的阳光，心情一定会马上舒畅起来，回家的路也会变得... [查看全文](#) >

COMSOL 中国 · 1 个月前 · 编辑精选