

**文献翻译**

**（ 2014 届本科）**

**学 院：信息学院**

**专 业：计算机科学与技术**

**班 级：计科四班**

**姓 名： 贾锐**

**学 号：1451435**

**指导教师：池涛**

**2018年 3 月**

英文翻译

Angular vs. React vs. Vue: A 2017 comparison

Deciding on a JavaScript framework for your web application can be overwhelming. Angular and React are very popular these days, and there is an upstart which has been getting a lot of traction lately: VueJS. What’s more, these are just a few of the new kids on the block.

So, how are we supposed to decide? A pros-and-cons list never hurts. We’ll do this in the style of my previous article, “[9 Steps: Choosing a tech stack for your web application](https://medium.com/unicorn-supplies/9-steps-how-to-choose-a-technology-stack-for-your-web-application-a6e302398e55" \t "https://medium.com/unicorn-supplies/_blank)”.

Before we start — SPA or not?

You should first make a clear decision as to whether you need a single-page-application (SPA) or if you’d rather take a multi-page approach. Read more on this in my blog post, “[Single-page-application (SPA) vs. Multi-page web applications (MPA)](https://medium.com/unicorn-supplies/angular-vs-react-vs-vue-a-2017-comparison-c5c52d620176)” (coming soon, follow me [on Twitter](http://www.twitter.com/jensneuhaus/" \t "https://medium.com/unicorn-supplies/_blank) for updates).

The starters today: Angular, React and Vue

First, I’d like to discuss lifecycle & strategic considerations. Then, we’ll move to the features & concepts of the three javascript frameworks. Finally, we’ll come to a conclusion.

Here are the questions we’ll address today:

How mature are the frameworks / libraries?

Are the frameworks likely to be around for a while?

How extensive and helpful are their corresponding communities?

How easy is it to find developers for each of the frameworks?

What are the basic programming concepts of the frameworks?

How easy is it to use the frameworks for small or large applications?

What does the learning curve look like for each framework?

What kind of performance can you expect from the frameworks?

Where can you have a closer look under the hood?

How can you start developing with the chosen framework?

Ready, set, GO!

Lifecycle & strategic considerations

**1.1 Some history**

Angular is a TypeScript-based Javascript framework. Developed and maintained by Google, it’s described as a “Superheroic JavaScript MVW Framework”. Angular (also “Angular 2+”, “Angular 2” or “ng2”) is the rewritten, mostly incompatible successor to AngularJS (also “Angular.js” or “AngularJS 1.x”). While AngularJS (the old one) was initially released in October 2010, it is still getting bug-fixes, etc. — the new Angular (sans JS) was introduced in September 2016 as version 2. The newest major release is version 4, as version 3 was skipped. Angular is used by Google, Wix, weather.com, healthcare.gov and Forbes (according to madewithangular, stackshare and libscore.com).

React is described as “a JavaScript library for building user interfaces”. Initially released in March 2013, React was developed and is maintained by Facebook, which uses React components on several pages (not as a single-page application, however). According to this article by Chris Cordle, React is used far more at Facebook than Angular is at Google. React is also used by Airbnb, Uber, Netflix, Twitter, Pinterest, Reddit, Udemy, Wix, Paypal, Imgur, Feedly, Stripe, Tumblr, Walmart and others (according to Facebook, stackshare and libscore.com).

Facebook is working on the release of React Fiber. It will change React under the hood — rendering is supposed to be much faster as a result — but things will be backward-compatible after the changes. Facebook talked about the changes at its developer conference in April 2017, and an unofficial article about the new architecture was released. React Fiber was released with React 16 in September 2017.

Vue is one of the most rapidly growing JS frameworks in 2016. Vue describes itself as a “Intuitive, Fast and Composable MVVM for building interactive interfaces.” It was first released in February 2014 by ex-Google-employee Evan You (BTW: Evan wrote an interesting blog post about the marketing activities and numbers in the first week back then). It’s been quite a success, especially given that Vue is getting so much traction as a one-man show without the backing of a big company. Evan currently has a team of dozen core developers. In 2016, version 2 was released. Vue is used by Alibaba, Baidu, Expedia, Nintendo, GitLab — a list of smaller projects can be found on madewithvuejs.com.

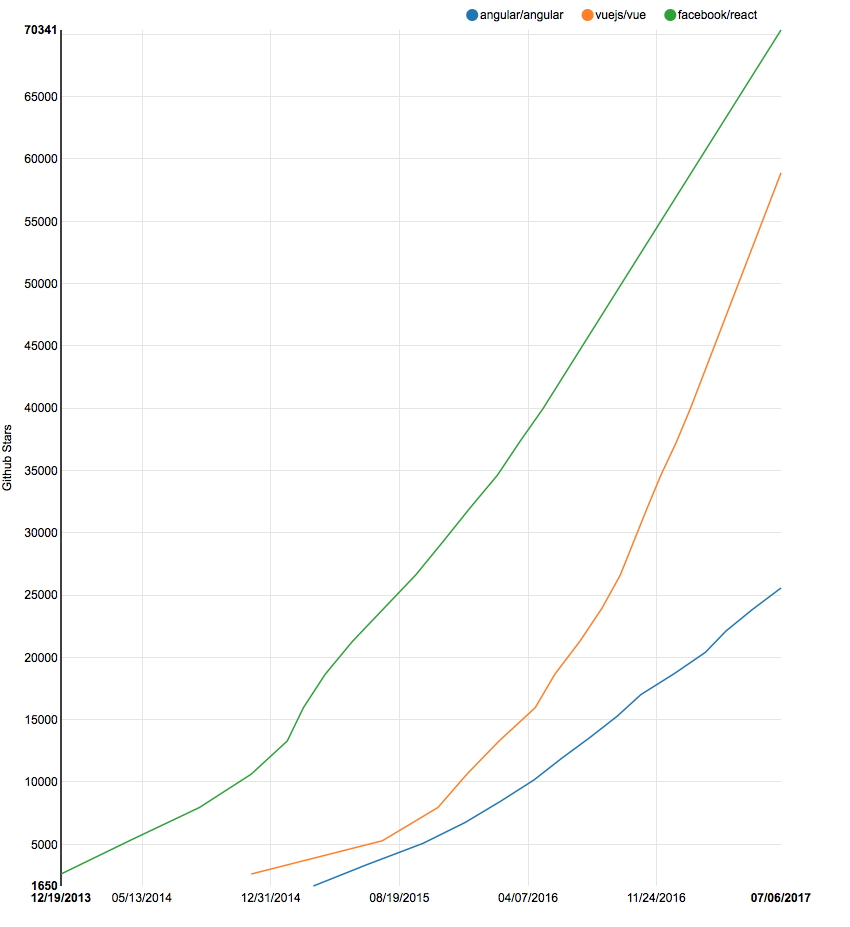
All three Frameworks are available under the MIT license.

React shipped with a special BSD3-license file until September 2017. There were a lot of discussions on the patent file. If you are interested into the history of this, you can read this Github issue discussion, the reasons and the history behind the patent file (by Ex-Facebook engineer James Ide), Why you should not be scared (by Dennis Walsh), the warning against the use for startups (by Raúl Kripalani) and a former statement from Facebook on this topic: Explaining React’s license. Anyway — all should not matter anymore as Facebook finally announced, that React will get the MIT license.

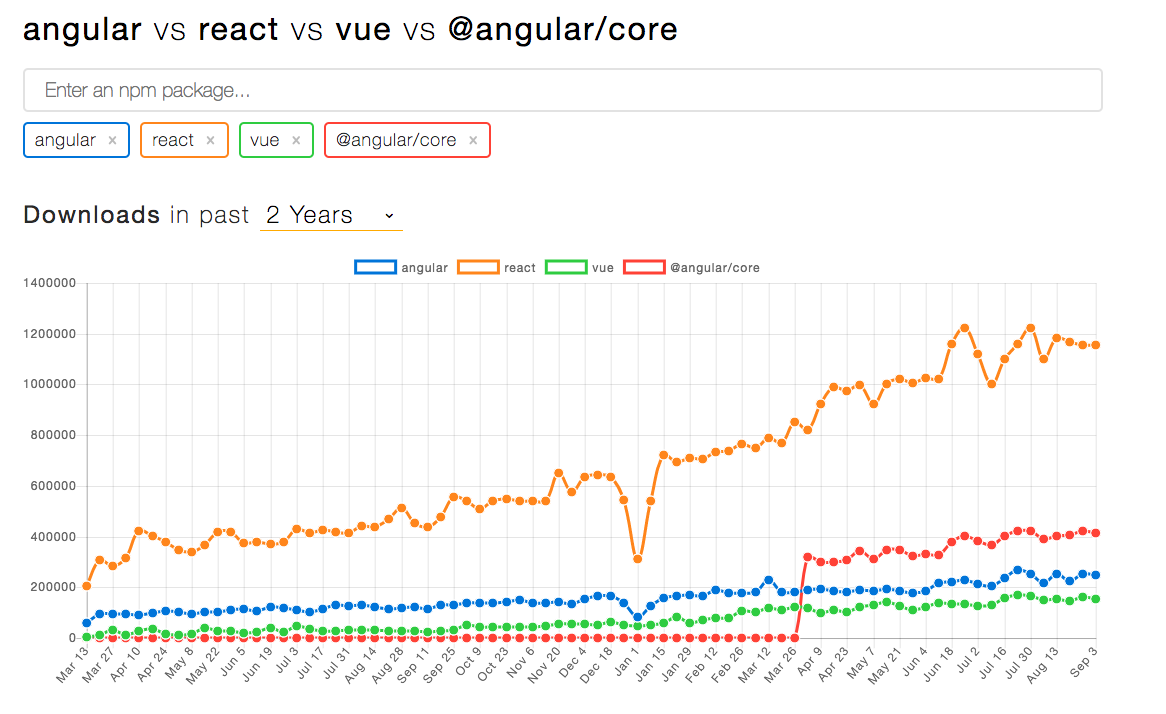
**1.2 Core development**

As already noted, Angular and React are supported and used by big companies. Facebook, Instagram and Whatsapp are using it for their pages. Google uses it in a lot of projects: for example, the new Adwords UI was implemented using Angular & Dart. Again, Vue is realized by a group of individuals whose work is supported via Patreon and other means of sponsorship. You can decide for yourself whether this is a positive or negative. Matthias Götzke thinks that Vue’s small team is a benefit because it leads to cleaner and less over-engineered code / API.

Let’s have a look at some statistics: Angular lists 36 people on their team page, Vue lists 16 people, and React doesn’t have a team page. On Github, Angular has > 25,000 stars and 463 contributors, React has > 70,000 stars and > 1,000 contributors, and Vue has almost 60,000 stars and only 120 contributors. You can also check the Github Stars History for Angular, React and Vue. Once again, Vue seems to be trending very well. According to bestof.js, over the last three months Angular 2 has been getting an average of 31 stars per day, React 74 stars, and Vue.JS 107 stars.

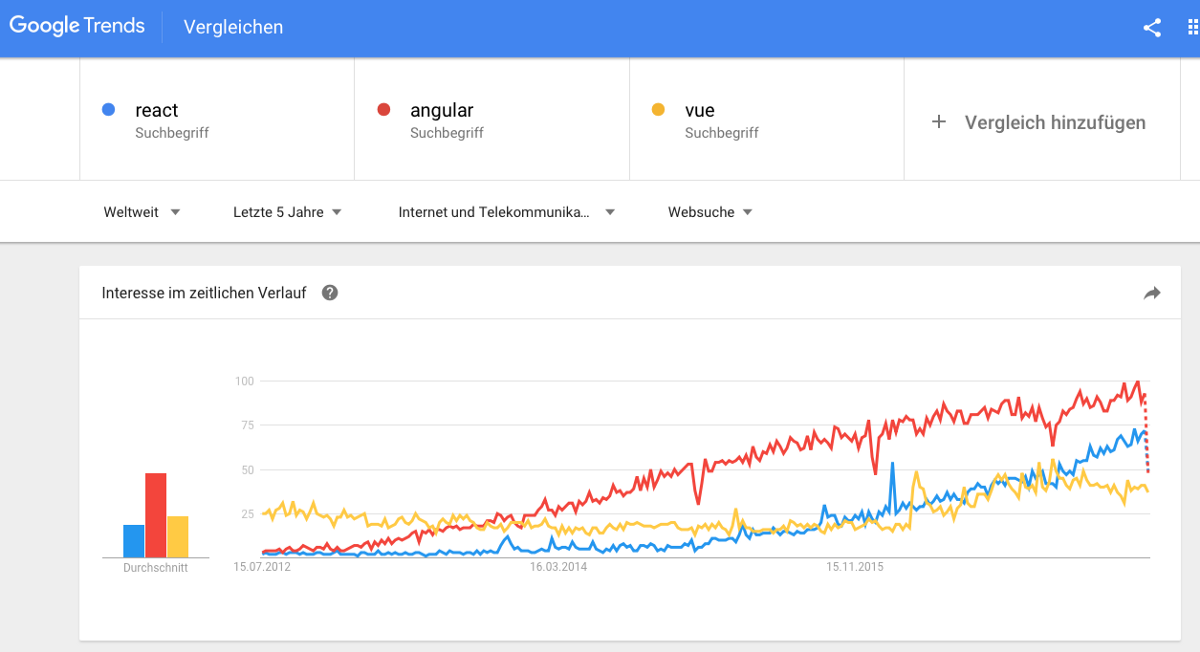


Update: Thanks to Paul Henschel for pointing out the npm trends. They show the number of downloads for the given npm packages and are even more helpful as a pure look at the Github stars:



**1.3 Market lifecycle**

It’s hard to compare Angular, React and Vue in Google Trends because of the various names and versions. One way to approximate could be to the search in the category “Internet & technologies”. Here is the result:



Oh, well. Vue was not created before 2014 — so something is amiss here. La Vue is French for“view”, “sight” or “opinion”. Maybe it’s that. A comparison of “VueJS” and “Angular” or “React” is not fair either, as VueJS has hardly any results compared to the others.

Let’s try something else, then. The Technology Radar from ThoughtWorks gives a good impression of how technologies evolve over time. Redux is in the adoption stage (to be adopted in projects!), and it has been invaluable in a number of ThoughtWorks projects. Vue.js is in the trial stage (try it out!). It is described as a lightweight and flexible alternative to Angular with a lower learning curve. Angular 2 is in the assessment stage — it is used successfully by ThoughtWork teams, but not a strong recommendation yet.

According to the last Stackoverflow 2017 survey, React is loved by 67% of surveyed developers and AngularJS by 52%. “No interest to continue developing” registers higher numbers for AngularJS (48%) vs. React (33%). Vue is not in the Top 10 in either case. Then there’s the statejs.com survey comparing “front-end frameworks”. The most interesting facts: React and Angular have 100% awareness, and Vue is unknown to 23% of the people surveyed. Regarding satisfaction, React scored 92% for “would use again”, Vue 89% and Angular 2 only 65%.

What about another customer satisfaction poll? Eric Elliott started one in October 2016 to evaluate Angular 2 and React. Only 38% of the people surveyed would use Angular 2 again, while 84% would use React again.

**1.4 Long-term support & migrations**

React APIs are quite stable, as Facebook states in their design principles. There are also some scripts to help you move from your current API to a newer one: check out react-codemod. Migrations are quite easy and there is no such thing (needed) as a long-term-support version. In this Reddit post, people note that the upgrades never really were a problem. The React team wrote a blog post about their versioning scheme. When they add a deprecation warning, they keep it for the rest of the current release version before the behavior is changed in the next major version. There is no planned change to a new major version — v14 was released in October 2015, v15 was published in April 2016, and v16 does not have a release date yet. The upgrade should not be a problem, as recently noted by a React core developer.

Regarding Angular, there is a blog post about versioning and releasing Angular starting with the v2 release. There will be one major update every six months, and there will be a deprecation period of at least six months (two major releases). There are some experimental APIs marked in the documentation with shorter deprecation periods. There is no official announcement yet, but according to this article, the Angular team has announced long-term-support versions starting with Angular 4. Those will be supported for at least one year beyond the next major version release. This means Angular 4 will be supported until at least September 2018 with bug-fixes and important patches. In most cases, updating Angular from v2 to v4 is as easy as updating the Angular dependencies. Angular also offers a guide with information as to whether further changes are needed.

The update process for Vue 1.x to 2.0 should be easy for a small app — the developer team has asserted that 90% of the APIs stayed the same. There is a nice upgrade-diagnostic migration-helper tool working on the console. One developer noted that the update from v1 to v2 was still no fun in a big app. Unfortunately, there is no clear (public) roadmap about the next major version or information on plans for LTS versions.

One more thing: Angular is a full framework and offers a lot of things bundled together. React is more flexible than Angular, and you will probably wind up using more independent, unsettled, fast-moving libraries — this means that you need to take care of the corresponding updates and migrations on your own. It could also be a detriment if certain packages are no longer maintained or some other package becomes the de facto standard at some point.

**1.5 Human resources & recruiting**

If you have in-house HTML developers who do not want to learn more Javascript, you are better off choosing Angular or Vue. React entails more Javascript (we talk about this later).

Do you have designers working close to the code? The user “pier25” notes on Reddit that React makes sense if you are working for Facebook, where everyone is a superhero developer. In the real world, you won’t always find a designer who can modify JSX — as such, working with HTML templates will be much easier.

The good thing about the Angular framework is that a new Angular 2 developer from another company will quickly familiarize themselves with all the requisite conventions. React projects are each different in terms of architectural decisions, and developers need to get familiar with the particular project setup.

Angular is also good if you have developers with an object-oriented background or who don’t like Javascript. To drive that point home, here is a quote from Mahesh Chand:

I am not a JavaScript developer. My background is building large-scale enterprise systems using “real” software platforms. I started in 1997 building applications using C, C++, Pascal, Ada, and Fortran. (…) I can clearly say that JavaScript is just gibberish to me. Being a Microsoft MVP and expert, I have a good understanding of TypeScript. I also don’t see Facebook as a software development company. However, Google and Microsoft are already the largest software innovators. I feel more comfortable working with a product that has strong backing from Google and Microsoft. Also (…) with my background, I know Microsoft has even bigger plans for TypeScript.

Well, then… I should probably mention that Mahesh is a Regional Director at Microsoft.

**2. Comparison of React, Angular & Vue**

**2.1 Components**

The frameworks in question are all component-based. A component gets an input, and after some internal behavior / computing, it returns a rendered UI template (a sign in / sign out area or a to-do list item) as output. The defined components should be easy to reuse on the webpage or within other components. For example, you could have a grid component (consisting of a header component and several row components) with various properties (columns, header information, data rows, etc.) and be able to reuse the component with different data sets on another page. Here is a comprehensive article about components, in case you’d like to learn more about this.

React and Vue both excel at handling dumb components: small, stateless functions that receive an input and return elements as output.

**2.2 Typescript vs. ES6 vs. ES5**

React focuses on the use of Javascript ES6. Vue uses Javascript ES5 or ES6.

Angular relies on TypeScript. This offers more consistency in related examples and open source projects (React examples can be found in either ES5 or ES6). This also introduces concepts like decorators and static types. Static types are useful for code intelligence tools, like automatic refactoring, jump to definitions, etc. — they are also supposed to reduce the number of bugs in an application., though there certainly isn’t consensus on this topic. Eric Elliott disagrees in his article “The shocking secret about static types”. Daniel C Wang says that the cost of using static types does no harm and that it’s good to have both test-driven development (TDD) and static typing.

You should also probably know that you can use Flow to enable type-checking within React. It’s a static type-checker developed by Facebook for JavaScript. Flow can also be integrated into VueJS.

If you are writing your code in TypeScript, you are not writing standard JavaScript anymore. Even though it’s growing, TypeScript still has a tiny user base compared to that of the whole JavaScript language. One risk could be that you’re moving in the wrong direction because TypeScript may — however unlikely it is — also disappear over time. Additionally, TypeScript adds a lot of (learning) overhead to projects — you can read more about this in the Angular 2 vs. React comparison by Eric Elliott.

Update: James Ravenscroft wrote in a comment to this article, that TypeScript has first-class support for JSX — components can be type-checked seamlessly. So if you like TypeScript and you want to use React, this should not be a problem.

**2.3 Templates — JSX or HTML**

React breaks with long-standing best practices. For decades, developers were trying to separate UI templates and inline Javascript logic, but with JSX, these are intermixed again. This might sound terrible, but you should listen to Peter Hunt’s talk “React: Rethinking best practices” (from October 2013). He points out that separating templates and logic is merely a separation of technologies, not concerns. You should build components instead of templates. Components are reusable, composable and unit-testable.

JSX is an optional preprocessor for HTML-like syntax which will be compiled in Javascript later. It has some quirks — for example, you need to use className instead of class, because the latter is a protected name in Javascript. JSX is a big advantage for development, because you have everything in one place, and code completion and compile-time checks work better. When you make a typo in JSX, React won’t compile, and it prints out the line number where the typo occurred. Angular 2 fails quietly at run time (this is argument is probably invalid if you use AOT with Angular).

JSX implies that everything in React is Javascript — it is used for both the JSX templates and the logic. Cory House points this out in his article from January 2016: “Angular 2 continues to put ‘JS’ into HTML. React puts ‘HTML’ into JS.”. This is a good thing, because Javascript is more powerful than HTML.

The Angular templates are enhanced HTML with special Angular language (Things like ngIf or ngFor). While React requires knowledge of JavaScript, Angular forces you to learn Angular-specific syntax.

Vue features “single-file components”. This seems like a trade-off with regard to the separation of concerns — templates, scripts and styles are in one file but in three different, ordered sections. This means you get syntax highlighting, CSS support and easier use of preprocessors like Jade or SCSS. I have read in other articles, that JSX is easier for debugging because Vue will not show bad HTML syntax errors. This is not true because Vue converts HTML to render functions — so errors are shown without problems (Thanks to Vinicius Reis for commenting and the correction!).

Side note: If you like the idea of JSX and want to use it in Vue, you can use babel-plugin-transform-vue-jsx.

**2.4 Framework vs. library**

Angular is a framework rather than a library because it provides strong opinions as to how your application should be structured and also has more functionality out of the box. Angular is a “complete solution” — batteries included and ready to provide you with a pleasant start. You don’t need to analyze libraries, routing solutions or the like — you can just start working.

React and Vue, on the other hand, are universally flexible. Their libraries can be paired to all kinds of packages (there are quite a lot for React on npm, but Vue has fewer packages because it’s still quite young). With React, you can even exchange the library itself for API-compatible alternatives like Inferno. However, with great flexibility comes great responsibility — there are no rules and limited guidance with React. Every project requires a decision regarding its architecture, and things can go wrong more easily.

Angular, on the other hand, comes with a confusing nest of build tools, boilerplate, linters & time-sinks to deal with. This is also true of React if starter kits or boilerplates are used. They’re naturally very helpful, but React works out of the box, and that’s probably the way you should learn it. Sometimes the variety of tools needed for a working in a Javascript environment is referred to as “Javascript fatigue”. There is an article about it by Eric Clemmons, who has this to say:

There are still a bunch of installed tools, you are not used to, when starting with the framework. These are generated but probably a lot of developers do not understand, what is happening under the hood — or it takes a lot of time for them to do.

Vue seems to be the cleanest and lightest of the three frameworks. GitLab has a blog post about its decision regarding Vue.js (October 2016):

Vue.js comes with the perfect balance of what it will do for you and what you need to do yourself.(…) Vue.js is always within reach, a sturdy, but flexible safety net ready to help you keep your programming efficient and your DOM-inflicted suffering to a minimum.

They like the simplicity and ease of use — the source code is very readable, and no documentation or external libraries are needed. Everything is very straightforward. Vue.js “does not make large assumptions about much of anything either”. There’s also a podcast about GitLab’s decision.

Another blogpost about a shift towards Vue comes from Pixeljets. React “was a great step forward for JS world in terms of state-awareness, and it showed lots of people the real functional programming in a good, practical way”. One of the big cons of React vs. Vue is the problem of splitting components into smaller components because of the JSX restrictions. Here is a quote of the article:

For me and my team the readability of code is important, but it is still very important that writing code is fun. It is not funny to create 6 components when you are implementing really simple calculator widget. In a lot of cases, it is also bad in terms of maintenance, modifications, or applying visual overhaul to some widget, because you need to jump around multiple files/functions and check each small chunk of HTML separately. Again, I am not suggesting to write monoliths — I suggest to use components instead of microcomponents for day-to-day development.

There are interesting discussions about the blog post on Hacker news and Reddit — there are arguments from dissenters and further supporters of Vue alike.

中文翻译

**Angular vs. React vs. Vue: 在2017年比较**

为你的web应用程序决定一个JavaScript框架是非常重要的。在最近Angular和React是非常流行的，而且近些时间有个新兴的框架非常引人注意：VueJS。更重要的是，这些这是这个街区里的新的年轻成员。所以我们该如何决定？一份优缺点列表是没有伤害的。

在我们开始之前—是否是SPA

您应该首先明确您是需要单页面应用程序（SPA）还是希望采用多页面方法。

现在开始：Angular，React，Vue

首先，我想讨论生命周期和战略考虑。 然后，我们将转向三个JavaScript框架的功能和概念。 最后，我们会得出结论。

以下是我们今天要解决的问题：

框架/库有多成熟？

框架可能会在一段时间内出现吗？

他们相应的社区有多广泛和有帮助的？

为每个框架找到开发人员有多容易？

框架的基本编程概念是什么？

为小型或大型应用程序使用框架有多容易？

每个框架的学习曲线是什么样的？

你可以从框架期望什么样的性能？

你在哪里可以仔细观察下引擎盖？

你如何开始用选择的框架进行开发？

预备，准备，开始！

生命周期和战略考虑

**1.1历史**

Angular是一个基于TypeScript的Javascript框架。 由Google开发和维护，它被描述为“超级英雄JavaScript MVW框架”。 Angular（也称为“Angular 2+”，“Angular 2”或“ng2”）是AngularJS（也称为“Angular.js”或“AngularJS 1.x”）的改写，是其大多不兼容的继承者。 尽管AngularJS（旧版本）最初于2010年10月发布，但它仍然在修复bug等问题 - 新的Angular（sans JS）于2016年9月推出，版本号为2。最新的主要版本为版本4， 版本3被跳过。 Angular被Google，Wix，weather.com，healthcare.gov和Forbes（根据madewithangular，stackshare和libscore.com）等使用。

React被描述为“用于构建用户界面的JavaScript库”。 React最初于2013年3月发布，由Facebook开发并维护，后者在多个页面上使用React组件（但不是作为单页面应用程序）。 根据Chris Cordle撰写的这篇文章，React在Facebook上的使用远远多于Angular在Google上的使用。 React也被Airbnb，Uber，Netflix，Twitter，Pinterest，Reddit，Udemy，Wix，Paypal，Imgur，Feedly，Stripe，Tumblr，Walmart和其他人使用（根据Facebook，stackshare和libscore.com）。

Facebook正在开发React Fiber。 它会改变子例程下的React - 渲染速度应该更快 - 但是在变化之后，框架会向后兼容。 Facebook在2017年4月的开发者大会上讨论了这些变化，并发布了一篇关于新架构的非官方文章。 React Fiber于2017年9月发布了React 16。

Vue是2016年发展最迅速的JS框架之一.Vue将自己描述为“用于构建交互式界面的直观，快速和可组合的MVVM”。该框架于2014年2月首次由前Google员工Evan You（提一下：Evan发布的时候写了一篇关于营销活动和数字的有趣博客文章）。 这是相当成功的，尤其是考虑到Vue在没有大公司支持的情况下获得如同一个人的表演一样的吸引力。 Evan目前拥有一批核心开发人员。 2016年，版本2发布。 Vue被阿里巴巴，百度，Expedia，任天堂，GitLab使用 - 小型项目列表可以在madewithvuejs.com上找到。

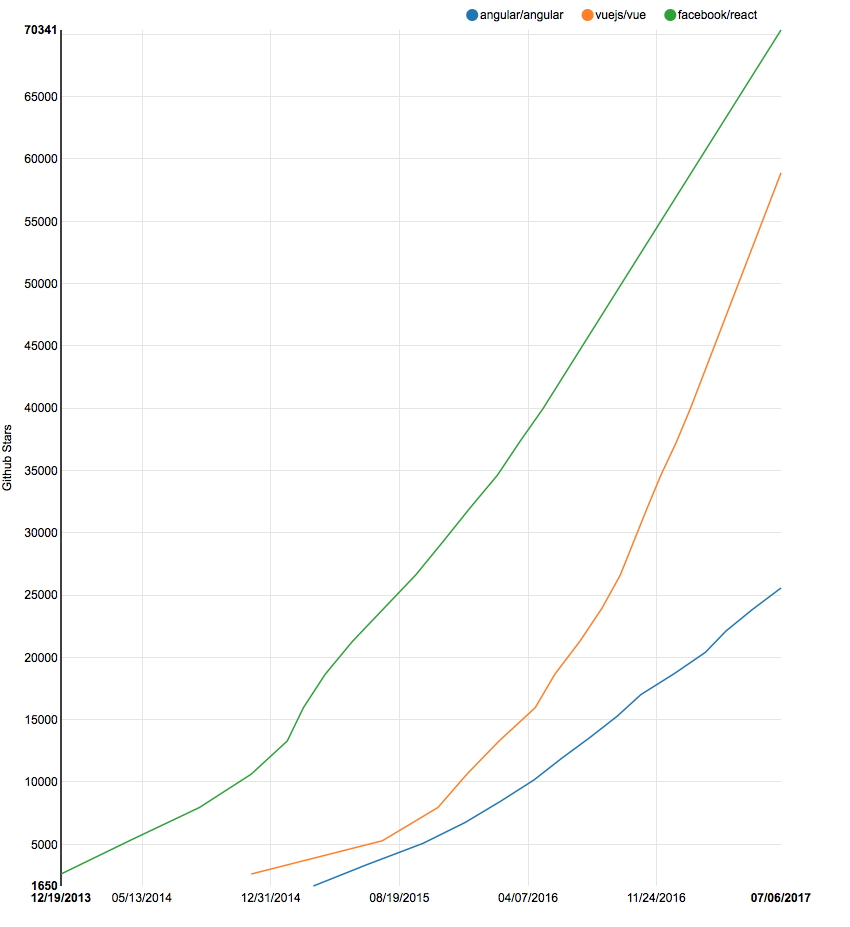
所有三种框架都可以在MIT许可下获得。

React附带特殊的BSD3许可证文件，直到2017年9月。有关专利文件的讨论很多。 如果您对这段历史感兴趣，您可以阅读Github问题讨论，专利文件背后的原因和历史（由Facebook前工程师James Ide撰写），《为什么您不应该害怕》（由Dennis Walsh提供）， 《警告不要用于初创公司》（由RaúlKripalani提供）以及Facebook上关于此主题的陈述：解释React的许可。 无论如何 - 所有这些都不应该像Facebook最终宣布的那样，React将获得MIT许可。

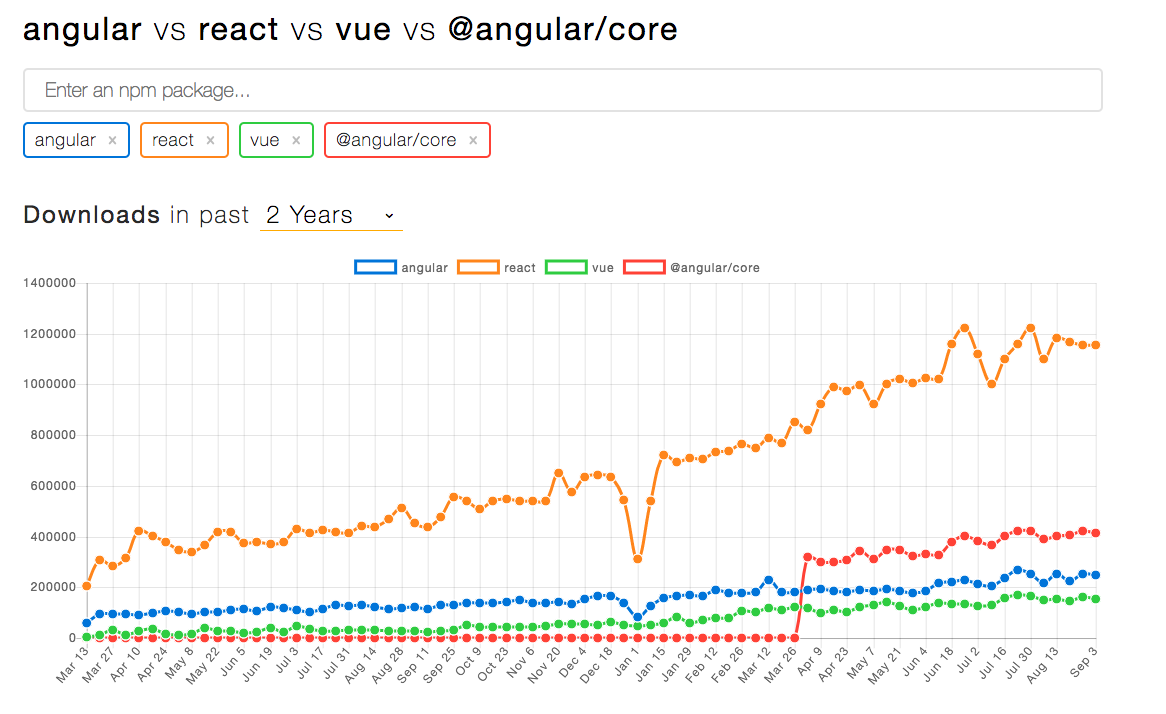
**1.2核心发展**

如前所述，Angular和React得到了大公司的支持和使用。 Facebook，Instagram和Whatsapp将其用于他们的页面。 谷歌在很多项目中使用它，例如，新的Adwords UI是使用Angular＆Dart实现的。 再次，Vue是由一群通过Patreon和其他赞助方式支持其工作的个人实现的。 您可以自行决定这是积极的还是消极的。 MatthiasGötzke认为Vue的小团队是一个好处，因为它会导致更简洁，更少的过度设计的代码/ API。

让我们来看看一些统计数据：Angular在他们的团队页面上列出了36个人，Vue列出了16个人，而React没有团队页面。 在Github上，Angular拥有> 25,000星级和463名贡献者，React拥有> 7万星级和> 1,000个贡献者，而Vue拥有近6万名明星和120名贡献者。 您还可以查看Github Stars的Angular，React和Vue历史。 再次，Vue似乎趋势非常好。 根据bestof.js的数据，在过去三个月中，Angular 2的平均每天获得31颗星星，React 是74颗星星，同时Vue.JS是107颗星星。

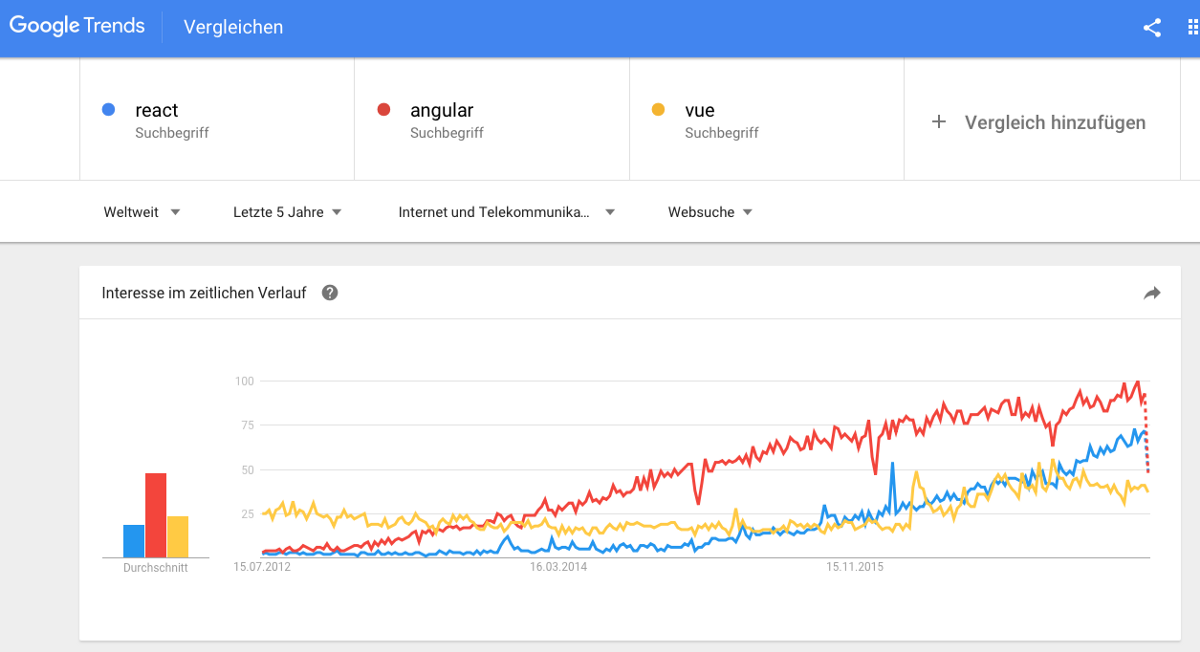


更新：感谢Paul Henschel给出的npm趋势。 它们显示给定npm软件包的下载次数，对比Github星星数量的纯粹看法更有帮助：



**1.3市场生命周期**

由于各种名称和版本，很难在Google趋势中比较Angular，React和Vue。 一种近似的方法是在“Internet＆technologies”类别中搜索。 结果如下：



好吧。 Vue并非在2014年之前创建的 - 所以这里有些不对劲。 La Vue是法语中的“观看”，“视线”或“意见”的意思。 也许就是这样。 “VueJS”和“Angular”及“React”的比较也不公平，因为VueJS几乎没有任何结果。

那么，我们试试其他的东西吧。 ThoughtWorks的技术雷达给人留下了技术如何随着时间演变的良好印象。 Redux正处于采用阶段（将在项目中采用！），并且它在许多ThoughtWorks项目中具有无可估量的价值。 Vue.js正处于试用阶段（试用！）。 它被描述为Angular的轻量级和灵活的替代品，学习曲线较低。 Angular 2处于评估阶段 - 它已被ThoughtWork团队成功使用，但尚未获得强有力的推荐。

根据上一次的Stackoverflow 2017调查，React被67％的受访开发者喜爱，AngularJS则是52％。 “没有兴趣继续开发”这一项AngularJS（48％）要高于React（33％）。 在这两项调查中，Vue都没有进入前十名。 接下来是statejs.com调查，比较了“前端框架”。 最有趣的事实是：React和Angular有100％的知名度，而23％的受访者不了解Vue。 关于满意度，React因“可重复使用”而获得92％的满意度，而Vue是 89％，Angular 2仅获得65％。

另一个客户满意度调查呢？ Eric Elliott于2016年10月开始评估Angular 2和React。 只有38％的受访者表示会再次使用Angular 2，而84％的人会选择再次使用React。

**1.4长期支持和迁移**

正如Facebook在其设计原则中所述，React API非常稳定。 还有一些脚本可以帮助您从当前的API转移到更新的API：检查react-codemod。 迁移非常简单，并且没有（需要）作为长期支持的版本。 在这篇Reddit文章中，人们注意到升级从未成为一个真正的问题。 React团队撰写了一篇关于他们的版本控制计划的博客文章。 当他们添加弃用警告时，他们会在下一个主要版本中的行为更改之前保留当前版本的其余部分。 目前没有更改为新的主要版本的计划 - v14于2015年10月发布，v15于2016年4月发布，而v16尚未发布日期。 正如React核心开发人员最近指出的，升级不应该是一个问题。

关于Angular，有一篇关于版本控制和发布Angular的博客文章，从版本2发布开始。 每六个月会有一次重大更新，并且至少有六个月的弃用期（两个主要版本）。 在文档中标记了一些实验性的API，并使用较短的弃用期。 目前还没有官方声明，但根据这篇文章，Angular团队已经宣布了从Angular 4开始的长期支持版本。这些版本将在下一个主要版本发布之后至少支持一年。 这意味着至少在2018年9月之前Angular 4将有bug修复和重要补丁的支持。 在大多数情况下，将Angular从v2更新到v4与更新Angular依赖关系一样简单。 Angular还提供了有关是否需要进一步更改的信息指南。

Vue 1.x到2.0的更新过程对于一个小应用程序来说应该很容易 - 开发人员团队声称90％的API保持不变。 在控制台上有一个很好的升级 - 诊断迁移 - 辅助工具。 一位开发人员指出，从v1到v2的更新在大型应用程序中很无趣。 不幸的是，关于下一个主要版本或关于LTS版本计划的信息没有清晰的（公共）路线图。

还有一件事：Angular是一个完整的框架，并提供了很多捆绑在一起的东西。 React比Angular更灵活，您可能会使用更独立，不稳定，快速移动的库 - 这意味着您需要自行处理相应的更新和迁移。 如果某些包不再被保留，或者某些其他包在某些时候成为事实上的标准，这对于框架来说也是不利的。

**1.5人力资源和招聘**

如果您有不想学习更多Javascript相关知识的内部HTML开发人员，最好选择Angular或Vue。 React需要更多的Javascript方面的知识（我们稍后会讨论这个）。

有没有设计人员要与代码近距离打交道？ 用户“pier25”在Reddit上指出，如果你在Facebook工作，在每个开发人员都是超级英雄的情况下，React是有意义的。 在现实世界中，你不会总是找到一个可以修改JSX的设计师 - 因此，使用HTML模板会容易得多。

关于Angular框架的好处是，来自另一家公司的新的Angular 2开发人员将很快熟悉所有必要的约定。 React项目在架构决策方面各不相同，开发人员需要熟悉特定的项目设置。

如果您的开发人员拥有面向对象的背景或者不喜欢Javascript，那么Angular也很好。 为了佐证这个观点，以下是Mahesh Chand的一句话：

我不是JavaScript开发人员。 我的背景是使用“真正”的软件平台构建大型企业系统。 我从1997年开始使用C，C ++，Pascal，Ada和Fortran构建应用程序。 （...）我可以清楚地说，JavaScript对我来说简直是胡言乱语。 作为Microsoft MVP和专家，我对TypeScript有很好的理解。 我也不认为Facebook是一家软件开发公司。 不过，谷歌和微软已经是最大的软件创新者。 我在使用由谷歌和微软提供强大支持的产品时感到更加舒适。 另外（...）在我的背景下，我知道微软有更大的TypeScript计划。

好吧，那么......我应该提到，Mahesh是微软的区域总监。

**2.React，Angular和Vue的比较**

**2.1组件**

这里所讨论的框架都是基于组件的。 组件获取输入，并在一些内部的执行/计算之后，它返回一个渲染过的UI模板（登录/注销区或待办事项列表项）作为输出。 定义的组件应该易于在网页或其他组件中重复使用。 例如，您可以创建拥有包含各种属性（列，标题信息，数据行等）的网格组件（包含标题组件和多个行组件），并且可以在另一个页面上使用具有不同数据集的该组件。 这里有一篇关于组件的综合文章，如果你想了解更多这方面的内容的话。

React和Vue在处理愚蠢的组件方面都很出色：小型无状态函数接收输入并返回元素作为输出。

**2.2 Typescript vs. ES6 vs ES5**

React专注于使用Javascript ES6。 Vue使用Javascript ES5或ES6。

Angular依赖于TypeScript。 这为相关示例和开源项目提供了更多的一致性（React示例可以在ES5或ES6中找到）。 这也引入了像装饰器和静态类型的概念。 静态类型对于代码智能工具很有用，例如自动重构，跳转到定义等 - 它们也可以减少应用程序中的错误数量，虽然在这个主题上并没有达成共识。 Eric Elliott在他的文章“关于静态类型的令人震惊的秘密”中不同意这个观点。 Daniel C Wang表示，使用静态类型的代码于成本来说没有任何影响，并且同时具备测试驱动开发（TDD）和静态类型是一个很好的选择。

你也应该知道你可以使用Flow在React中启用类型检查。这是Facebook为JavaScript开发的静态类型检查器。 Flow也可以集成到VueJS中。

如果您使用TypeScript编写代码，则不再编写标准JavaScript。 尽管它在不断发展，但与整个JavaScript语言相比，TypeScript的用户群仍然很小。 您的行进方向错误可能是一个风险，因为TypeScript可能 - 但不太可能 - 随着时间的推移消失。 此外，TypeScript为项目增加了很多（学习）开销 - 您可以在Eric Elliott的Angular 2 vs. React比较中阅读更多关于这方面的内容。

更新：James Ravenscroft在对本文的评论中写道，TypeScript对JSX有一流的支持 - 可以无缝地对组件进行类型检查。 所以如果你喜欢TypeScript并且你想使用React，这应该不成问题。

**2.3模板 - JSX或HTML**

React打破了长期的最佳实践。 几十年来，开发人员尝试将UI模板和内联Javascript逻辑分开，但使用JSX时，这些又混合在一起。 这可能听起来很糟糕，但是你应该去听一听Peter Hunt的演讲“React：反思最佳实践”（2013年10月）。 他指出，分离模板和逻辑仅仅是技术的分离，而不是注意力。 您应该构建组件而不是模板。 组件可重复使用，可组合并且支持单元测试。

JSX是一种类似HTML语法的可选预处理器，可以在JavaScript中进行编译。 它有一些怪癖 - 例如，您需要使用className而不是class，因为后者是Javascript中的受保护名称。 JSX对于开发来说是一个很大的优势，因为你在一个地方拥有了所有的东西，并且代码的完成和编译时检查可以更好地工作。 当您在JSX中拼写错误时，React将无法编译，并打印出发生错字的行号。 Angular 2则在运行时悄然失败（如果您使用包含Angular的AOT，则这个参数可能无效）。

JSX意味着React中的所有内容都是Javascript编写的--它被同时应用于JSX模板和逻辑。 Cory House在2016年1月的文章中指出：“Angular 2继续将'JS'放入HTML中。 而React将'HTML'放入JS中。“ 这是一件好事，因为Javascript比HTML更强大。

Angular模板使用特殊的Angular语言（比如ngIf或ngFor）来增强HTML。 虽然React需要JavaScript的相关知识，但Angular会迫使您学习Angular特有的语法。

Vue具有“单个文件组件”。 这似乎是关于关注点分离的一种平衡 - 模板，脚本和样式在一个文件中，但在三个不同的有序部分中。 这意味着您可以获得语法高亮，CSS支持以及更容易使用的预处理器（如Jade或SCSS）。 我在其他文章中曾经读到过，JSX更易于调试，因为Vue不会显示错误的HTML语法错误。 这不是事实，因为Vue将HTML转换为渲染函数 - 所以错误显示没有问题（感谢Vinicius Reis的评论和更正！）。

注意：如果你喜欢JSX并想在Vue中使用它，你可以使用babel-plugin-transform-vue-jsx。

**2.4框架与库**

Angular是一个框架而不是一个库，因为它提供了关于如何构建应用程序的强大选项，以及更多开箱即用的功能。 Angular是一个“完整的解决方案” - 包含一个完整的初始软件包，随时为您提供愉快的开始。 您不需要分析库，路由解决方案等，只需开始工作即可。

另一方面，React和Vue普遍更加灵活。 他们的库可以配对各种包（在npm上有很多React的包，但Vue的包较少，因为它还很年轻）。 有了React，您甚至可以将库本身交换为API兼容的替代品，如Inferno。 但是，越灵活，其责任就越重大 - 对React没有规则和有限的指导。 每个项目都需要依靠其架构的决定，并且项目可能更容易出错。

另一方面，对于Angular来说，它还有一堆令人困惑的构建工具，比如 boilerplate, linters和time-sinks。如果也使用kits或boilerplate，React也是如此。 他们对于编码自然是非常有帮助的，但React可以直接使用，这可能是您应该学习它们的方式。 有时，在Javascript环境中工作所需的各种工具被称为“Javascript疲劳”。 Eric Clemmons有一篇关于它的文章，在其中他这样说：

在框架开始工作时，仍然有许多已安装且您还没有接触过的工具。 这些都是生成的，但很可能大部分开发人员都不明白在底层发生了什么 - 或者需要花费很多时间才能明白。

Vue似乎是三个框架中最干净和最轻型的框架。 GitLab上有一篇有关于Vue.js（2016年10月）的决定的博客文章：

Vue.js完美平衡了它将为您做什么以及您需要做什么（...）Vue.js始终可以帮助你创建一个坚固而灵活的安全网络并且帮助您保持高效的编程，同时你的DOM冲突造成的痛苦将降到最低。

他们喜欢简单易用 - 源代码非常易读，不需要任何文档或外部库。 一切都非常简单。 Vue.js“对任何东西都不做大的假设”。 还有一个关于GitLab决定的播客。

关于向Vue转变的另一篇博文来自Pixeljets。 React“在国家意识方面向JS世界迈进了一大步，它以很优美，实用的方式向很多人展示了真正的函数式编程。 React 对于 Vue来说的一大缺点是由于JSX的限制，需将组件分割成更小的组件。 这里是文章的引用：

对于我和我的团队来说，代码的可读性很重要，但编写有趣的代码也是非常重要的。 在实现一个简单的计算器小部件时创建6个组件很无趣。 在很多情况下，在维护，修改或对某些小部件进行可视化检修方面体验也很坏，因为你需要绕过多个文件/函数并分别检查每个小块HTML。 再次，我不打算写很多大段的代码 - 我建议使用组件代替微组件来进行日常开发。