State of Vue.js 2021

HOW IS VUE DOING
IN THE JS LANDSCAPE?

BROUGHT TO YOU BY MONTERAIL
IN COLLABORATION WITH THE VUE CORE TEAM

What's Inside?

1	Preface	3
2	What's New in Vue	5
3	Vue Today	9
4	Survey: How Developers use Vue	29
5	Vue Experts on Using Vue	53
6	Comparison with Other Technologies	73

Vue.js has been around since Evan You created it in 2014. Since then, it has grown in popularity so much that it doesn't need an extensive introduction. Among the innumerable JavaScript frameworks and libraries, it has managed to rise to the top and stay there. Currently, there are 20 members in the Vue Core team, and 130k GitHub users benefiting from the technology.

There are people who'd use Vue for virtually any job, and there are those that would rather leverage React or alternative frameworks. The bottom line is: Vue, like all other technologies, has some ideal fits, and things that can be fixed with workarounds.

The goal of this report is to give you a comprehensive overview of how Vue can be used in developement scenarios, what you can expect from it, and how it compares to other frameworks and libraries. If you're interested in how Vue can benefit your business check out the Vue for Business 2021 Report.

Let's dive in!

Błażej Cepil
Editor and Writer
Senior Content Marketing Specialist at Monterail

Peter Curac-Dahl
Writer
Digital Marketing Specialist at Monterail

Marta Klimowicz
Quantitative Analyst
Head of Growth at Monterail

Natalia Leśniak
Designer
Web Developer and Designer at Monterail

Eliza Neumann-Pluta
Writer and Publisher
Senior Content Marketing Specialist at Monterail

Contributors

Tim Benniks

Director of Web Development at Valtech Paris

Damian Dulisz

Tech Lead at Coursedog, Vue.js Core Team Member, Vue.js Consultant

Filip Rakowski

CTO & Co-founder at Vue Storefront

Artur Rosa

Frontend Developer at Monterail

Samuel Snopko

Head of Developer Relations at Storyblok

Natalia Tepluhina

Staff Frontend Engineer

Tonina Zhelyazkova

Software Developer at Wikimedia

2

What's new in Vue 3

Vue 3 was officially released in September, 2020. Version 3, titled “One Piece”, continues the anime-inspired naming convention of preceding versions. It also replaces Vue version 2 (2.0 “Ghost in the Shell” to 2.6 “Macross”) after almost four years of reliable and rewarding performance.

What does the new version mean for businesses and programmers alike who want to build using Vue? There are a lot of great new features, here are some that stand out:

- **The new version gets even smaller! Streamlined libraries result in a lightweight package (15kb), weighing in around 50% less than version 2.6. Output file sizes have also been reduced using tree-shaking, which removes unused features from your code.**
- **It’s faster. Upgrades to many aspects of the framework, including a rewrite of the Virtual DOM, have resulted in speed improvements.**
- **Composition API does wonders for code organization by keeping code for individual features together. At the end of the day this means a cleaner code, easier to read and share.**
- **The Teleport feature allows you to render code from one component to others, in different parts of the DOM tree. Previously known as portals, Teleport is now built into Vue 3.**

Continuous updates and improvements of Vue, show that it remains a solid technology choice for projects going into the future. Backward compatibility with previous versions ensures stability when it comes to updating and adding features to already completed projects. For a full overview of new features and changes in Vue 3 check out the full introduction document from Vue’s website.

And if you're interested, you can get a fresh perspective on how Vue.js came to be in Honeypot's documentary. It takes a look at popular frameworks from behind the scenes; including interviews with founders and thought leaders such as Evan You, Taylor Otwell (Laravel), Thorsten Lünborg (Vue core team), Scott Tolinski (Honeypot/Level up), and more.

From the documentary you'll learn how Evan's time at Google contributed to the creation of Vue. At first, Vue was just an idea to be more efficient at work. As the project started gaining users Evan saw it as an outlet for building something used in real life, as opposed to a concepts with no immediate use case.

The workload and user base increased in parallel. With more people using Vue, the feature requests and bug reports poured in. Evan starts working on Vue full-time and needs to find ways to monetize his new framework. The result is the Vue we know — low entry barrier with the stability and flexibility of more mature platforms. Check out the documentary yourself, we promise it's fascinating.

Vue Today

Let's take a look at the numbers. How fast is Vue growing? Should you learn it? Does it give you a competitive edge? Can you build something sustainable with it?

We've reviewed over a dozen trusted sources and compared the results to previous years as well as comparing Vue to other popular JavaScript frameworks. This data allowed us to establish an overview and draw conclusions. (Kudos to Tanguy Krotoff who prepared this data comparison that allowed us to peek into some historical records).

Although the numbers are not consistent across sources because of different survey samples, we can say with certainty: Vue is steadily growing in popularity. Some of the sources we used also don't consider the use of Vue in China, where it's quite popular. You should keep an eye on it if you didn't know. It does provide some unique (compared to React) possibilities like incremental implementation into an existing project, or ease of creating and maintaining small apps (not to mention beginner-friendliness). You can (and in some cases maybe should) build something long-lasting with it, since it's only increasing in popularity and use where in some cases the two main competitors have seen a slight downside curve.

Vue manages to do all this without the corporate backing that Angular (Google) and React (Facebook) have; they are funded by the community. Taking all this into consideration allows us to conclude that Vue is alive and kicking strong.

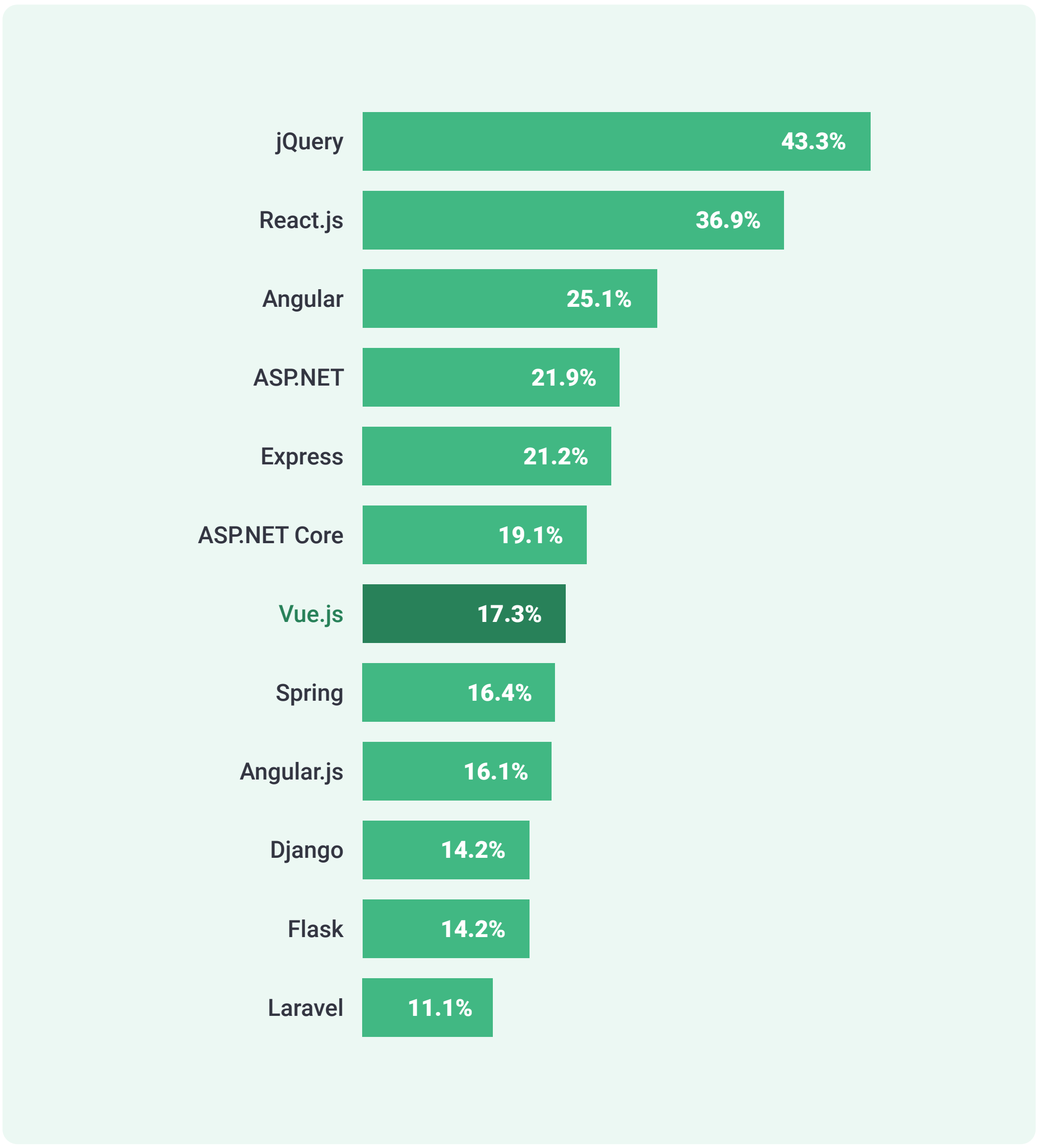
We did our best to compare Vue, React and Angular, but for various reasons in some of the sources we could compare historically only with AngularJS. Nevertheless, we hope this comparison will prove a valuable overview.

Let's dive into the details.

Statista

In February 2020, Statista asked 42,279 developers about their framework of choice and these are the results. Vue came in 7th with 17.3% of respondents opting for it. Not bad in itself, but we sure could use some historical context.

Most used web frameworks among developers worldwide, as of early 2020



Source: Statista

HackerRank

In 2020, HackerRank reached out to 116,648 developers to ask about the same thing Statista did, but they also compared the results with previous years. Vue came in 8th but, it showed a steady growth of interest among the HackerRank-related developers.

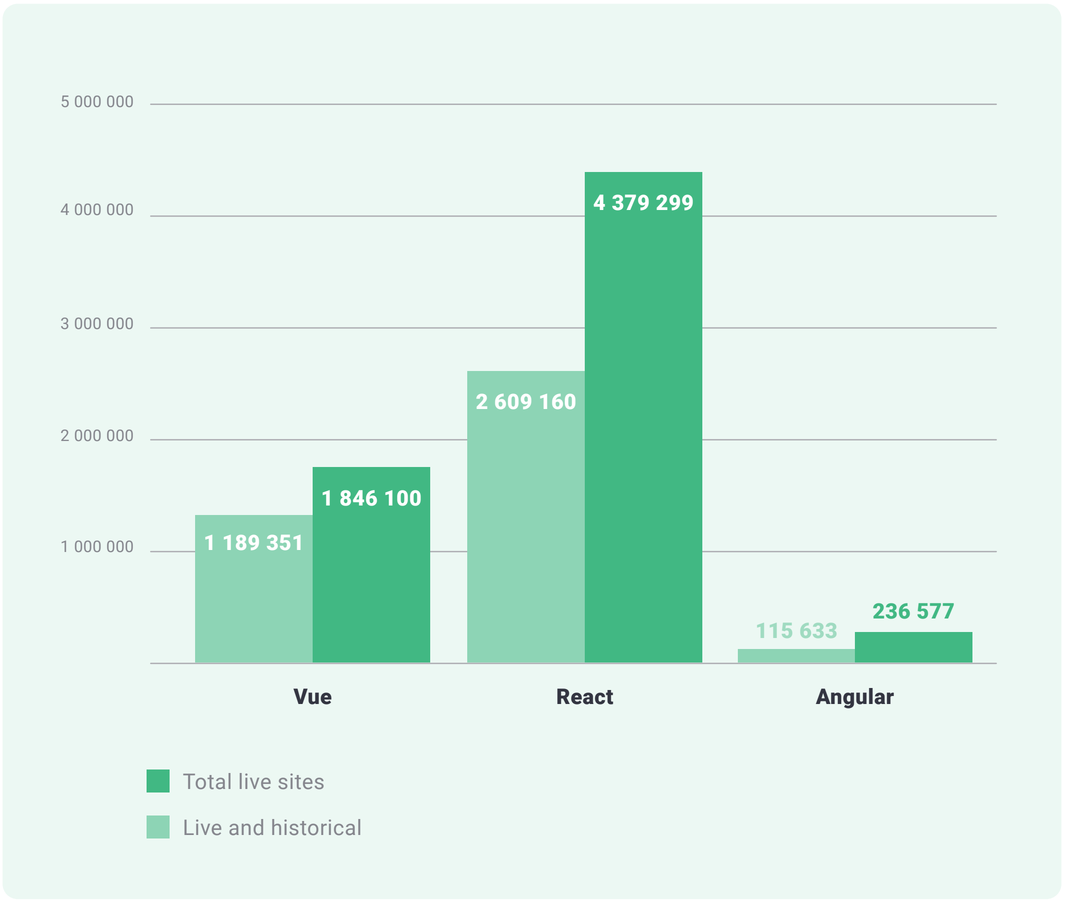
	2020	2019	2018
AngularJS	1	1	1
React	2	2	3
Spring	3	3	2
Django	▲ 4	6	6
ExpressJS	▼ 5	4	4
ASP	▼ 6	5	5
.NETCore	7	7	7
Vue.js	▲ 8	9	10
Ruby on Rails	▼ 9	8	8
JSF	10	10	9

Source: HackerRank

BuiltWith

Looking at the current (January 2021) Internet landscape, we clearly see that Vue, although not being the most widespread, positions itself remarkably well. It has more live sites than Angular and the ratio between live and live+historical websites is the best in the comparison, meaning that projects that go online with Vue, tend to stay out there.

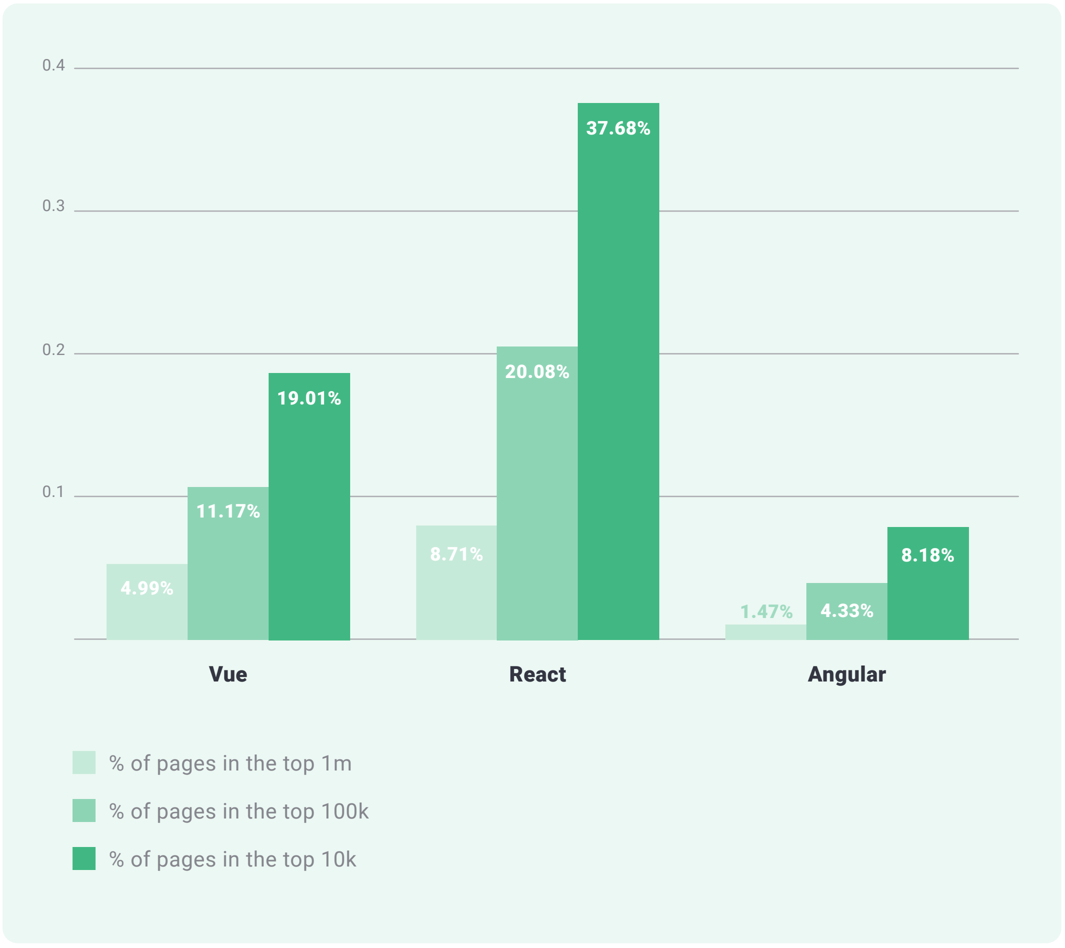
Number of websites built with



Source:
BuildWith

When it comes to a share in top websites, Vue built more of those in the top 1 million than Angular. And while it built fewer big-hitters in the top 10k when compared to React, it still has quite a significant share.

Share in top sites



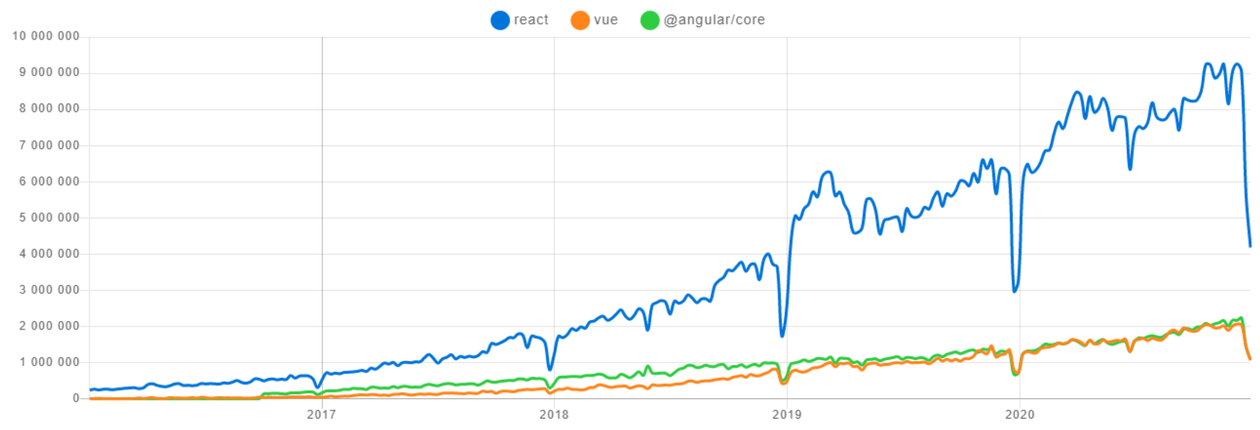
Source:
BuildWith

	Total live	Live and historical	Top 1 M	Top 100 K	Top 10 K
Vue	1 189 351	1 846 100	4.99% 49 870	11.17% 11 167	19,01% 1 901
React	2 609 160	4 379 299	8.71% 88 078	20.08% 20 807	37.68% 3 768
Angular	115 633	236 577	1.47% 14 695	4.33% 4 331	8.18% 818

NPMtrends

NPMtrends says that the use of Vue is growing steadily, maybe less so than React but Vue has more GitHub stars. Although not the best metric for judging a framework's usability, it shows the high engagement of the Vue community.

Downloads in past 5 Years



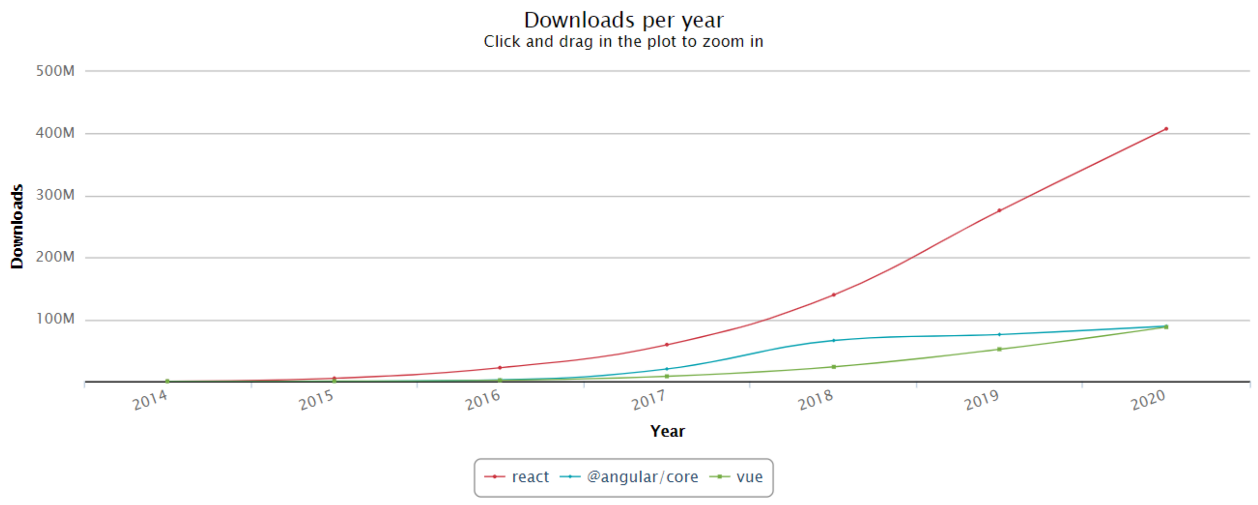
Stats

		stars	issues	updated	created	size
🟦	react	161 746	700	Jan 7, 2021	May 24, 2013	minzipped size 2.8 KB
🟧	vue	177 739	556	Jan 6, 2021	Jul 29, 2013	minzipped size 22.9 KB
🟩	@angular/core	69 391	2818	Jan 7, 2021	Sep 18, 2014	minzipped size 88.7 KB

Source:
NPMtrends

NPM-stat

NPM-stat shows different numbers and somehow different trends than NMPtrends but one thing is certain: the use of Vue packages goes up and is almost the same as Angular.



Total number of downloads between *2014-12-12* and *2020-12-31*:

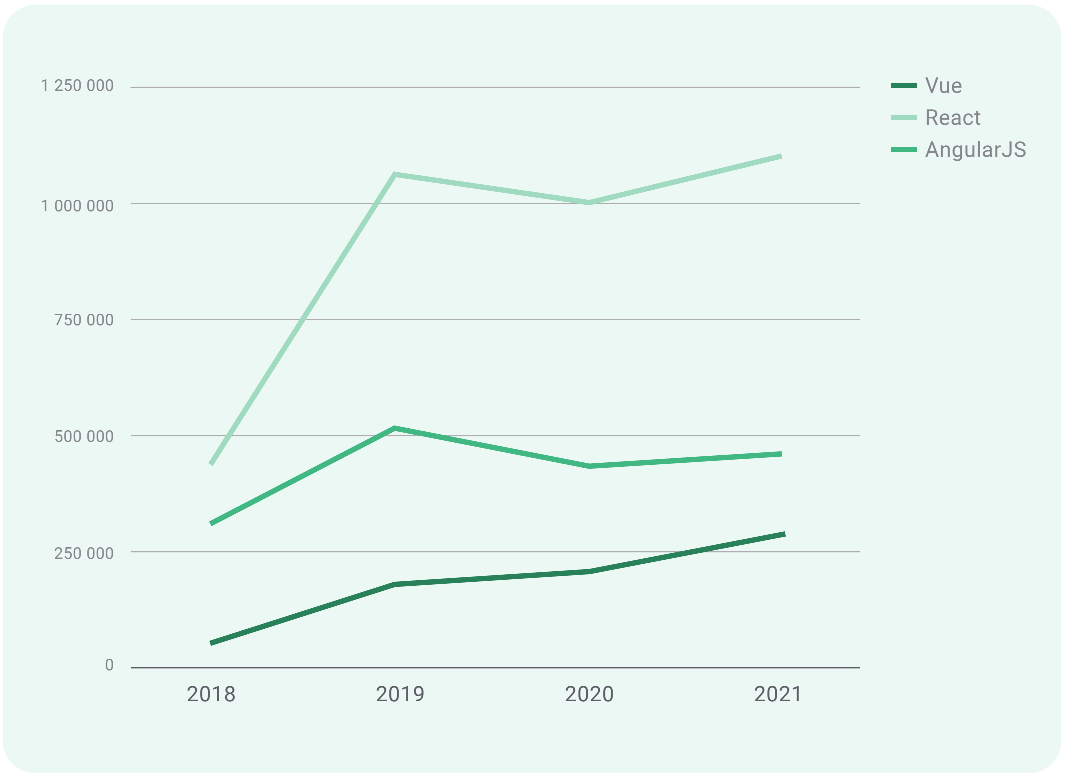
package	**downloads**
react	908,916,130
@angular/core	252,176,593
vue	172,000,056

Source:
NPM-stat

SimilarTech: market share & web usage statistics

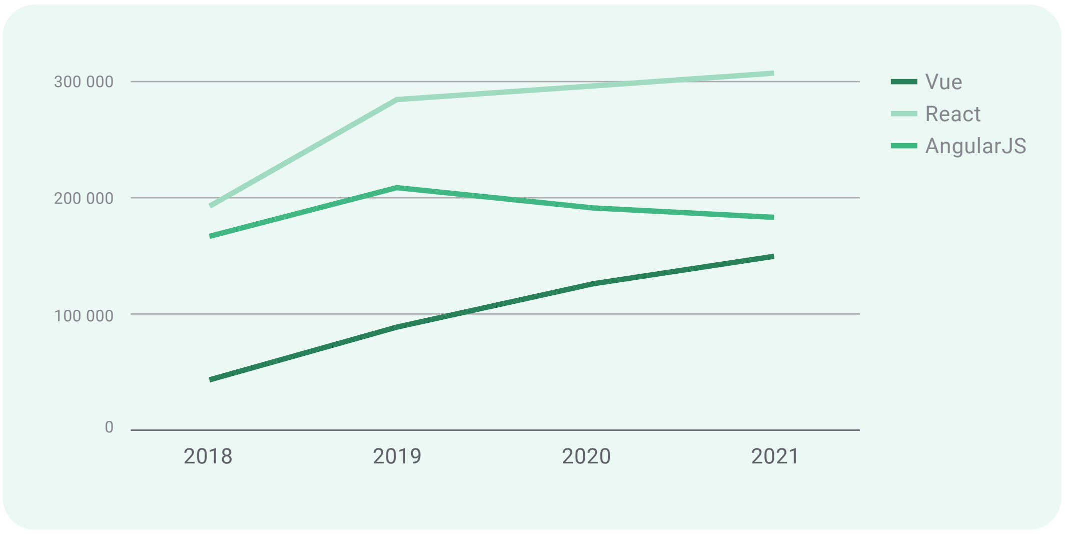
A look at SimilarTech crawls reveals some intriguing trends. The number of websites and unique domains built with Vue continues to grow while its competitors face some slowdowns. The numbers for 2020 don't match those found on BuiltWith, but the historical context here clearly shows a growth trend for Vue.

SimilarTech: number of websites



Source: **SimilarTech**

SimilarTech: unique domains



Source:
SimilarTech

Number of websites

	2021/01/07	2020/09/22	2019/12/12	2018/12/16
Vue	220 538	195 214	157 831	54 881
React	1 126 095	1 005 214	1 069 073	420 066
AngularJS	384 515	378 038	517 701	325 339

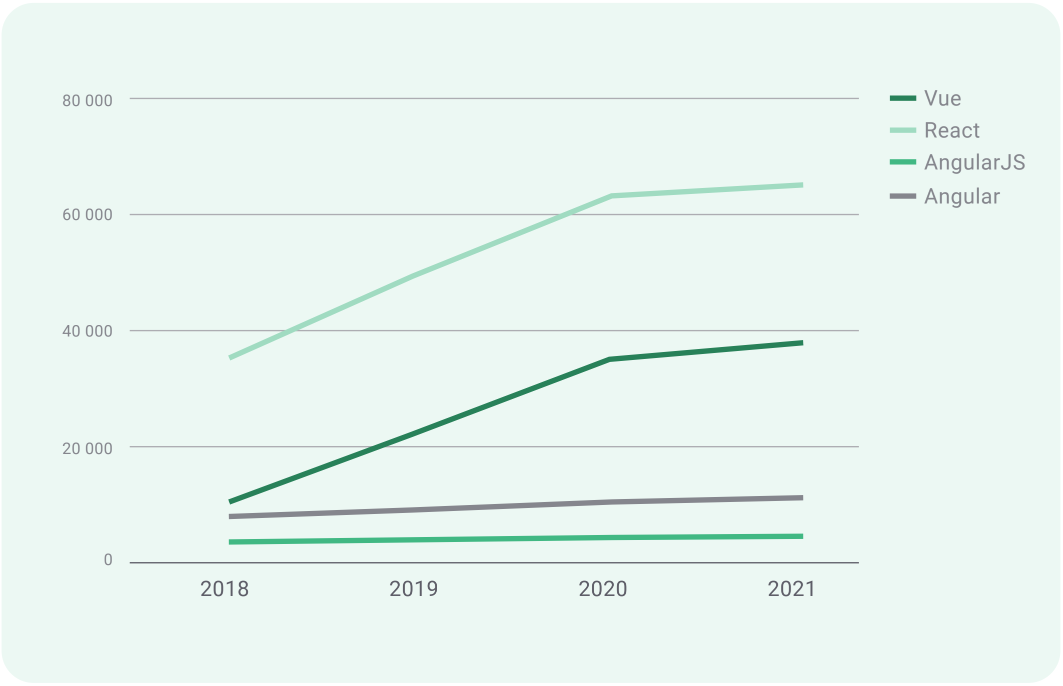
Unique domains

	2021/01/07	2020/09/22	2019/12/12	2018/12/16
Vue	140 184	121 748	91 509	40 033
React	313 355	296 347	287 997	196 048
AngularJS	188 734	189 590	203 614	171 570

GitHub dependents

Again, the number of apps that wouldn't run without Vue grows steadily over time.

Number of dependents



Source: **npmJS**

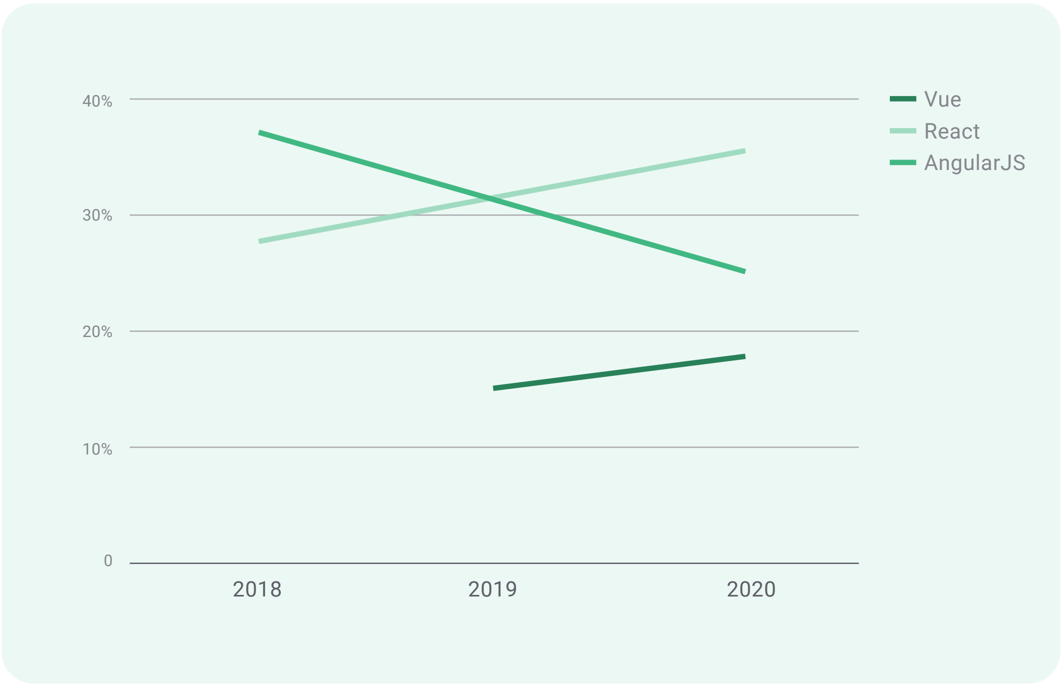
Number of dependents

	2021/01/07	2020/09/20	2019/12/12	2018/12/16
Vue	36 179	32 101	21 575	9 792
React	66 033	61 870	48 718	32 331
AngularJS	4 125	4 112	3 959	3 693
Angular 2+	11 246	10 873	9 610	7 555

Stack Overflow surveys

Stack Overflow surveys its users each year, asking about their most and least favorite frameworks. While the available survey questions are not consistent over the years, the results confirm the upward popularity trend for Vue (and downward for Angular).

Stack Overflow survey: framework popularity



Source:
Stack Overflow

2020 (65,000 developers)

	Popularity	Loved	Dreaded	Wanted
Vue	17.3%	66.0%	34.0%	16.4%
React	35.9%	68.9%	31.1%	22.4%
AngularJS	16.1%	24.1%	75.9%	7.7%
Angular	25.1%	54.0%	46.0%	10.6%

2019 (+90,000 developers)

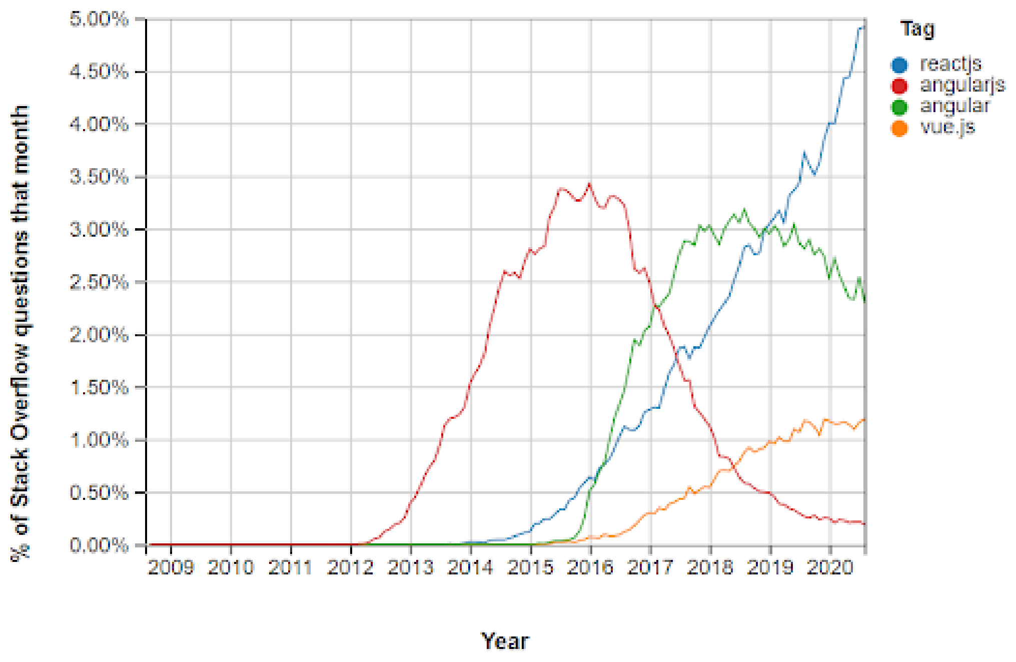
	Popularity	Loved	Dreaded	Wanted
Vue	15.2%	73.6%	26.4%	16.1%
React	31.3%	74.5%	25.5%	21.5%
AngularJS/ Angular	30.7%	57.6%	42.4%	12.2%

2018 (+100,000 developers)

	Popularity	Loved	Dreaded	Wanted
Vue	-	-	-	-
React	27.8%	69.4%	30.6%	21.3%
Angular	36.9%	54.6%	45.4%	14.3%

Stack Overflow questions

The questions asked on Stack Overflow increased in numbers for Vue and React. More so for React, but this could be explained by the fact that Vue is more beginner-friendly and is lauded for its comprehensive documentation.

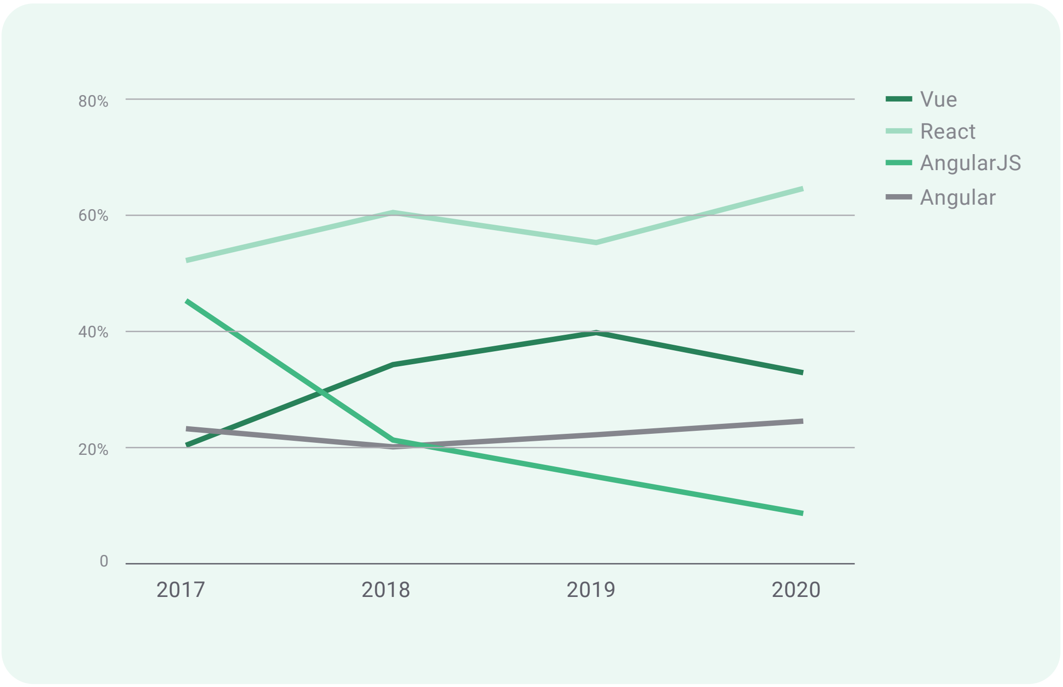


Source: StackOverflow

JetBrains survey

In 2020, JetBrains released a new report based on the answers of 19,696 developers from 18 countries. This is the only source that shows a decreasing popularity of Vue, but the reason for that might be the demographic background of the respondents. Since 2017, the majority of them are employed full-time in a company of 51-500 people, working in 2-7 people teams. It seems that React is most often used in such environments.

JetBrains survey: framework popularity (regular use)

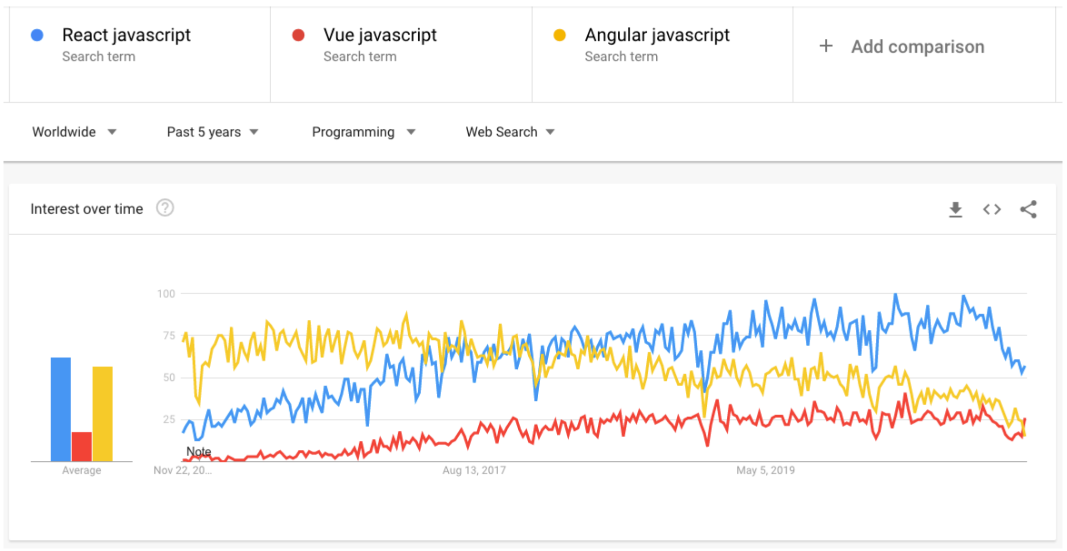


Source: **JetBrains**

	2020	2019	2018	2017
Developers	**19 696**	**7 000**	**6 000**	**5 000**
Vue	32%	39%	33%	20%
React	64%	54%	60%	49%
AngularJS	11%	14%	21%	44%
Angular	24%	23%	20%	22%

Google Trends

Google Trends gives us quite a broad spectrum of interests but for gauging popularity it's one of the most trusted sources out there. And it confirms the trend: Vue is going up (as well as React).

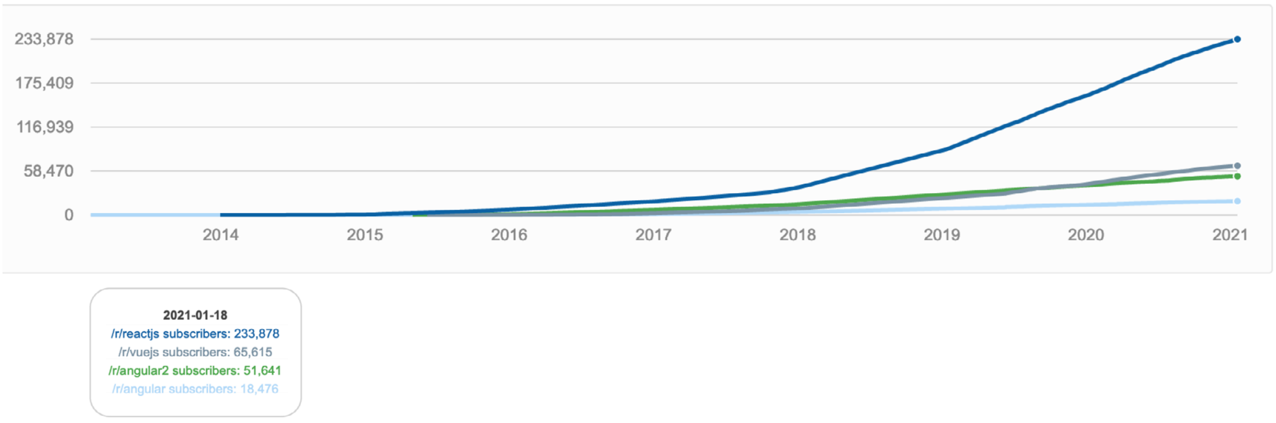


Source:
Google Trends

Social media

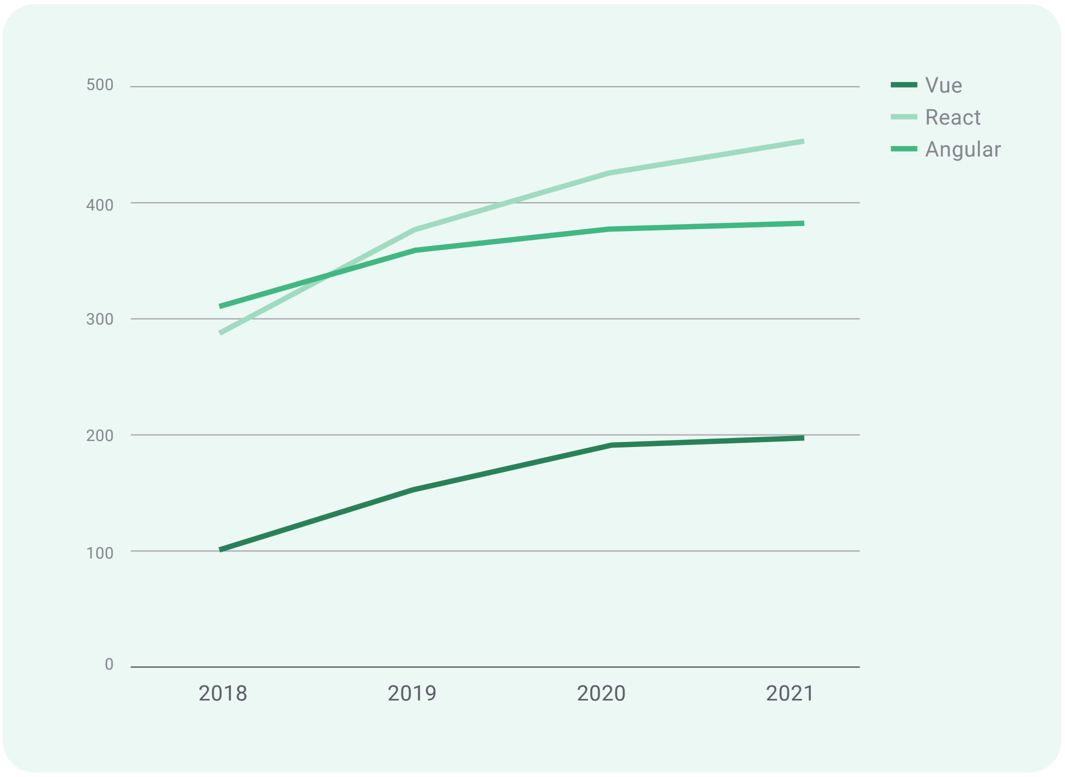
And one last place we can look at to see how Vue is doing is social media, namely Reddit and Twitter — virtually two of the most authoritative social sources out there. Looking at the number of followers over the years, it's clear that Vue continues to get traction.

Reddit followers



Source:
Frontpagemetrics

Twitter followers (k)



Source:
Twitter

	2021/01/07	2020/09/22	2019/12/12	2018/12/16
Vue	196.8 K	183.8 K	149.3 K	102 K
React	451.1 K	421.8 K	355.5 K	278 K
Angular	376.8 K	370 K	345.9 K	304 K

Summary

So there you have it — all the sources (barring JetBrains) tell us that Vue is used, liked, and will be used with success in the future. It has some unique benefits to offer and the community is exceptionally engaged. Add to this the popularity in China (that flies a bit under the radar of our tools) and the fact that Vue grows its impact without corporate backing, and you have a framework that is here to stay.

Survey: How Developers use Vue

Like the previous editions, the 2017 and 2019 State of Vue reports, this new, updated version allows us to learn more about the community of professionals using Vue.js for their projects and what they think lays ahead of Vue after the release of Vue 3.

Using an online survey, we received answers from 1635 software developers and Chief Technology Officers which we then examined to gain insights on:

- the popularity of Vue.js in their organizations,
- the reasons behind adding Vue to their tech stack and the doubts that accompanied their decision,
- the solutions they choose when developing Vue projects,
- the other libraries/frameworks and languages they use for frontend and backend development,
- their predictions and expectations for the future of Vue.js.

Report Data

All the data that was used to draft the report was collected from a survey conducted over a five-week period between December 2020 and January 2021. We received 1,635 responses, mainly from software developers and Chief Technology Officers (92.4% of the respondents held one of those roles).

Key Insights

90% of respondents claim there is a very high probability of them using Vue.js for their next project.

93% of the survey participants used the official documentation to learn about Vue and 76% pointed to great documentation as the biggest advantage of the framework.

60% decided to add Vue to their tech stack because it's a pretty easy framework to start with.

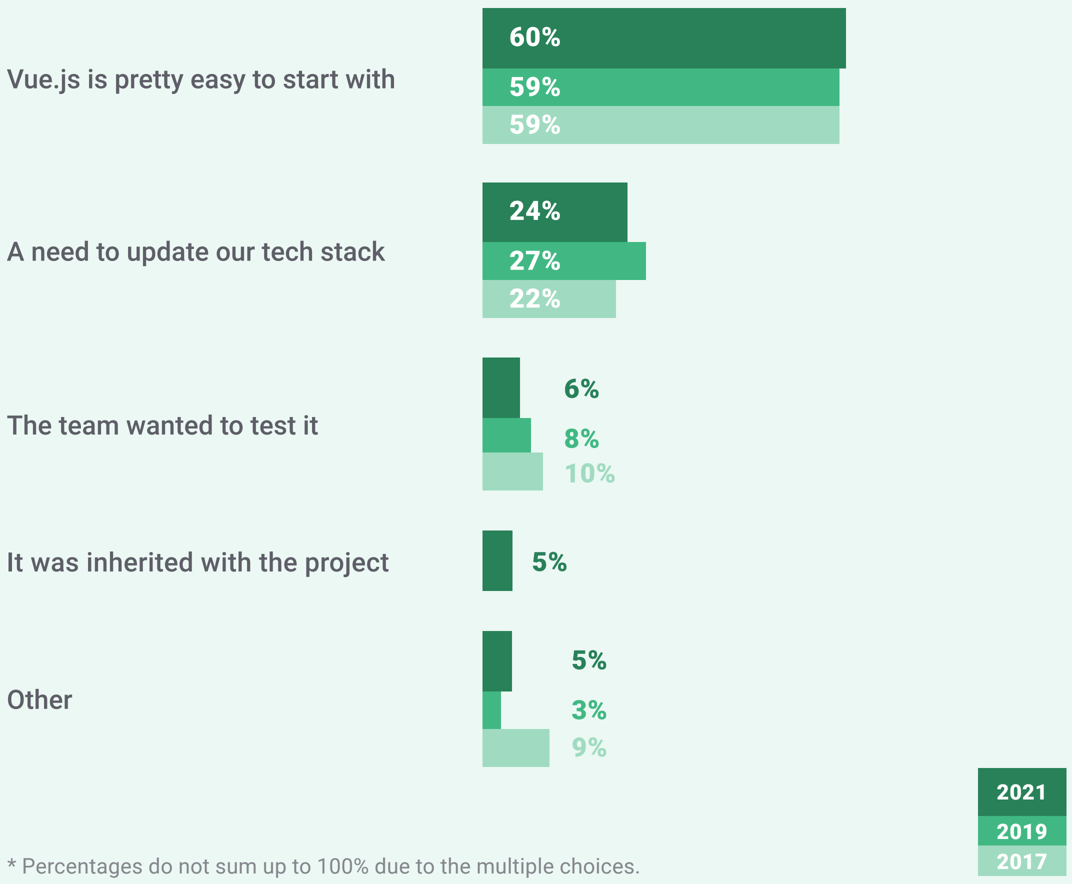
56% of the survey participants believe Vue.js will become even more popular within their organizations in the next 12 months.

Survey Questions & Answers

What was the most important reason behind adding Vue.js to the technology stack?

More than half of the respondents describe Vue.js as easy to start with. This percentage has been pretty steady: both in the 2017 and 2019 survey - 59% of the respondents chose this reason behind adding Vue.js to their technology stack.

THE MOST IMPORTANT REASON BEHIND ADDING VUE.JS TO THE TECH STACK

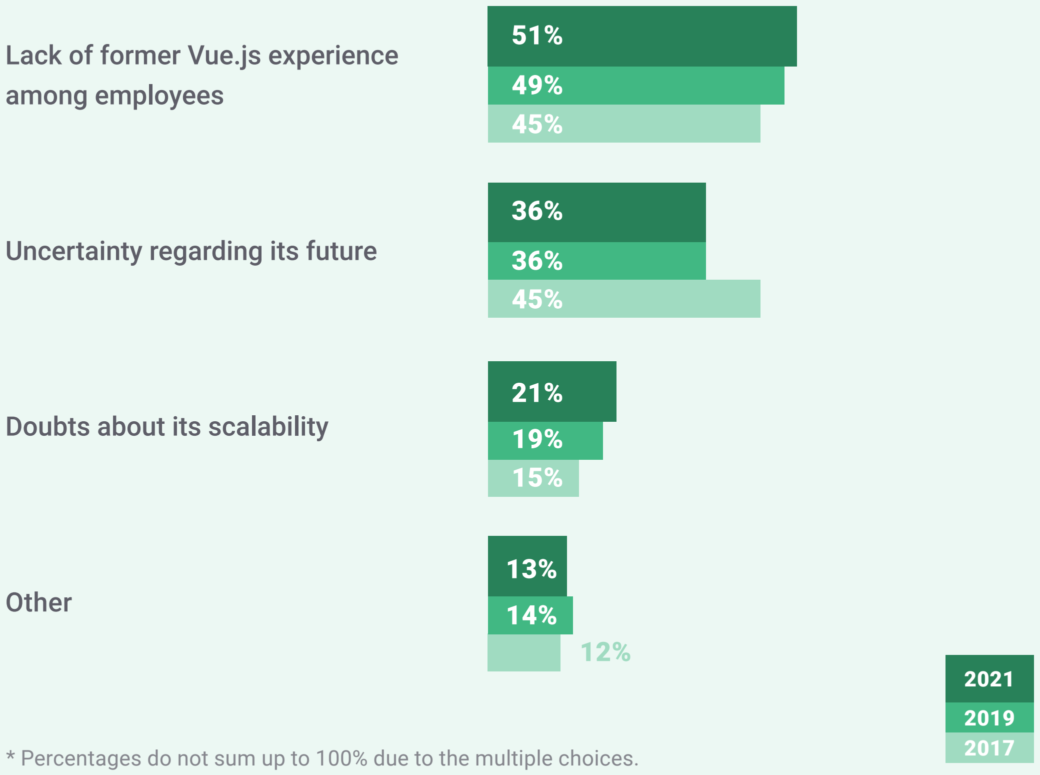


* Percentages do not sum up to 100% due to the multiple choices.

What were the doubts you and your team had when planning to add Vue.js to your tech stack?

Over 51% of the respondents say that the lack of former Vue-related experience was their main doubt when planning to add Vue.js to their tech stack. This number is pretty similar to the one we saw in 2019 and in 2017 when the answer was picked by 49% and 45% of the people we polled respectively.

DOUBTS WHEN ADDING VUE.JS TO TECH STACK

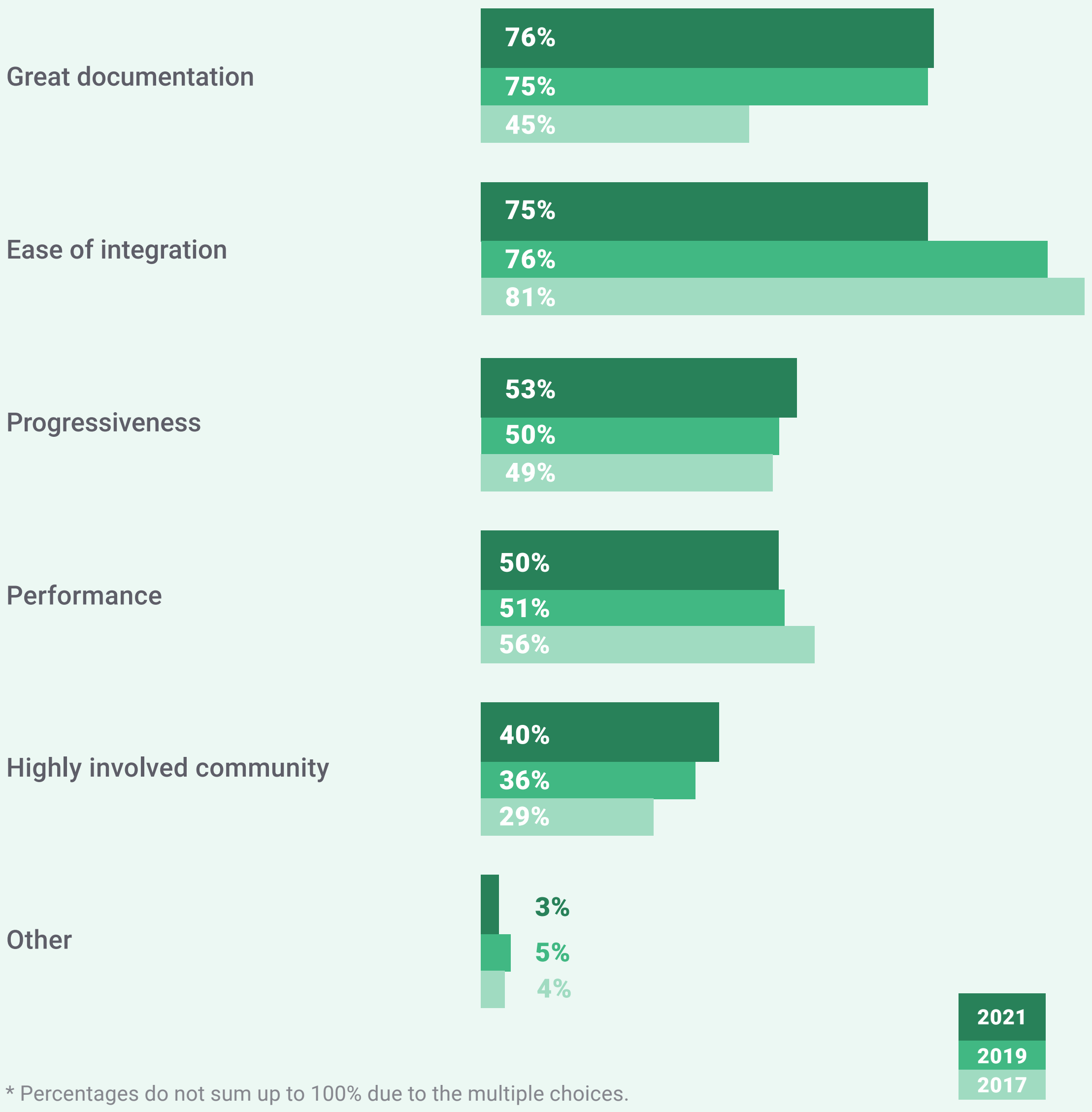


* Percentages do not sum up to 100% due to the multiple choices.

What are the biggest advantages that Vue.js brings to your organization?

For the third time in a row, we see similar results for the key benefits of bringing Vue.js to organization. There are no doubts that great documentation, ease of integration, and progressiveness support the choice of this framework.

THE BIGGEST ADVANTAGE



* Percentages do not sum up to 100% due to the multiple choices.

Is there anything you're missing when it comes to Vue.js?

Over 500 respondents shared their suggestions on what from the Vue.js framework. Most of the replies were related to individual needs of projects or developer's skills. However, we have managed to segment some of them to provide a clearer overview of the business's and developers' needs.

It's worth observing that when we asked the same question in 2019, the most frequent reply was the need for a mobile solution. That answer was given by over 130 respondents, over two times more than this year's most popular reply. This year, the lack of a native mobile solution was mentioned only by 16 people, so it seems that one of the most important needs has been addressed with a Vue Native solution.

This year, the need for TypeScript support became the most popular answer to this question, with 63 people expressing this need.

14 respondents brought up the need for better documentation, along with real-life examples of Vue.js' development, and case studies. Again, it's worth mentioning that in 2019 this need was expressed more widely – 46 respondents mentioned it in their replies. This suggests that the documentation is being improved, though there's still room for improvement in this area. Similarly, in 2019, 35 people suggested improvement of the learning resources. However, this time that need was expressed by less than 10 people, which may mean that not only the learning resources

have been improved, but also that a growing number of developers are more advanced in Vue itself, therefore they don't need that much additional help.

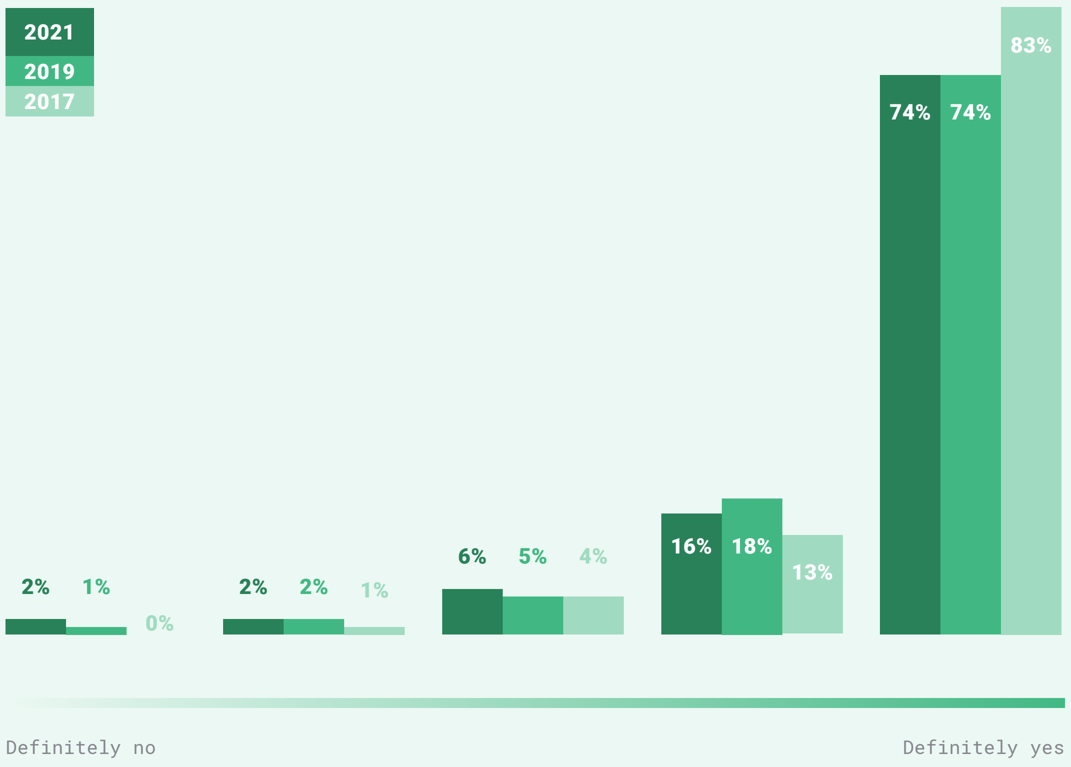
One of the new needs expressed in this year's edition of the survey is connected to Vue 3, released on the 18th of September 2020. 11 respondents mentioned the lack of easy migration to the higher version of the framework.

The same number of respondents expressed the need for IE 11 support (one of them adding "Yes, it's an enterprise").

What's the probability of you using Vue again for a new project?

Almost 90% of respondents claim there is a very high probability of them using Vue.js for their next project.

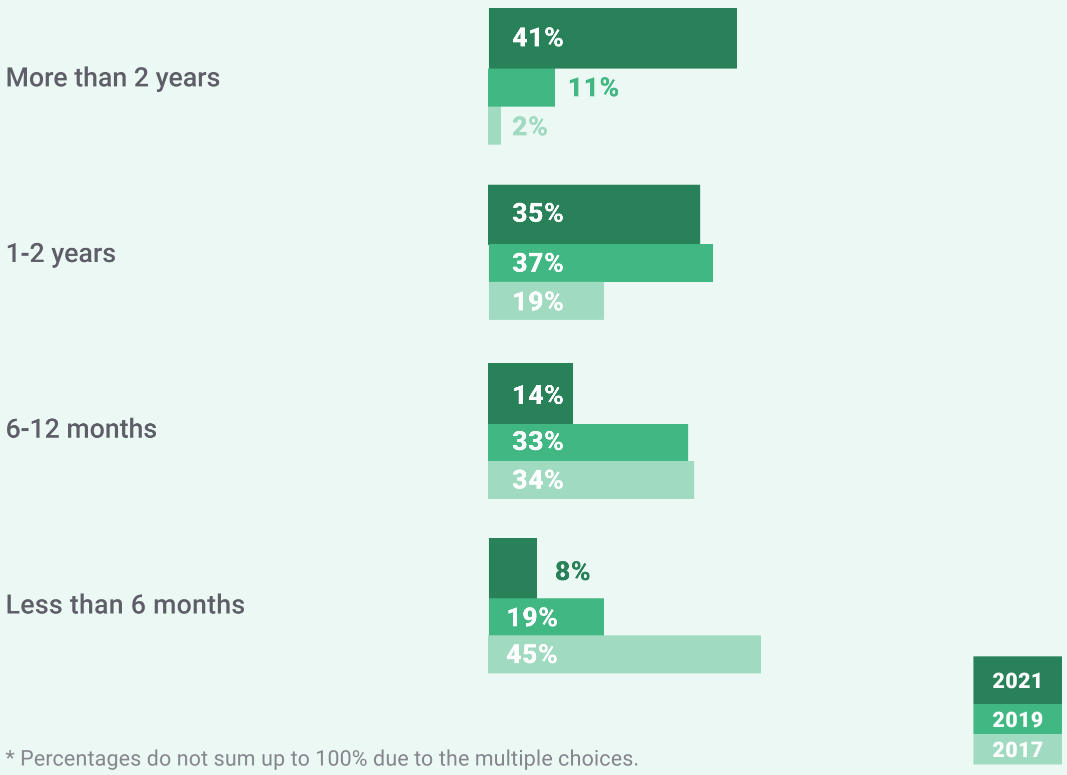
PROBABILITY OF USING VUE IN AN UPCOMING PROJECT



How long has Vue.js been in use within your organization?

In 2019, 11% of the respondents had been using Vue.js for over two years. This year that percentage has grown to 42,1%, showing that Vue is continuously being used within the companies.

HOW LONG HAS VUE.JS BEEN IN USE WITHIN YOUR ORGANIZATION?

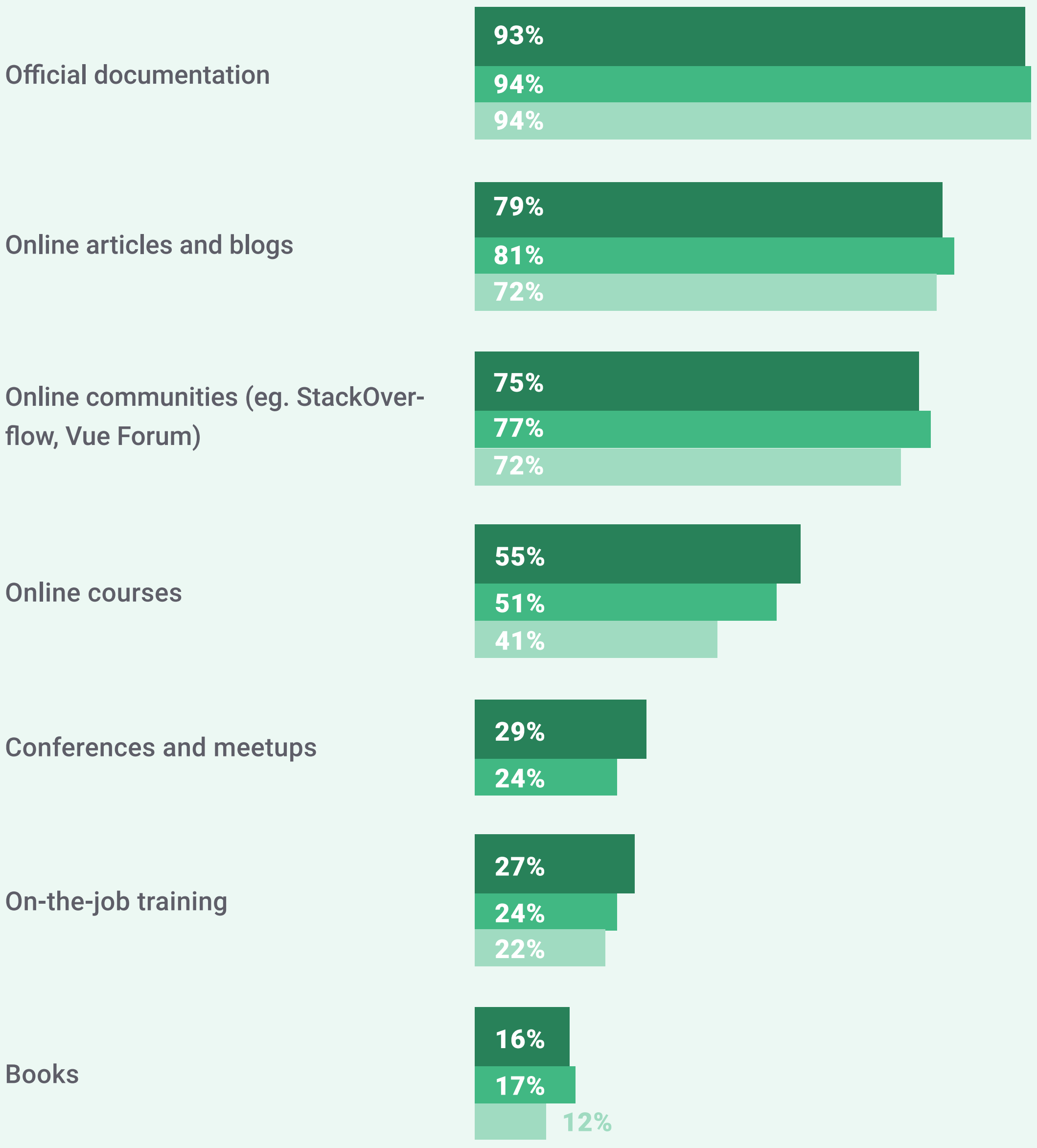


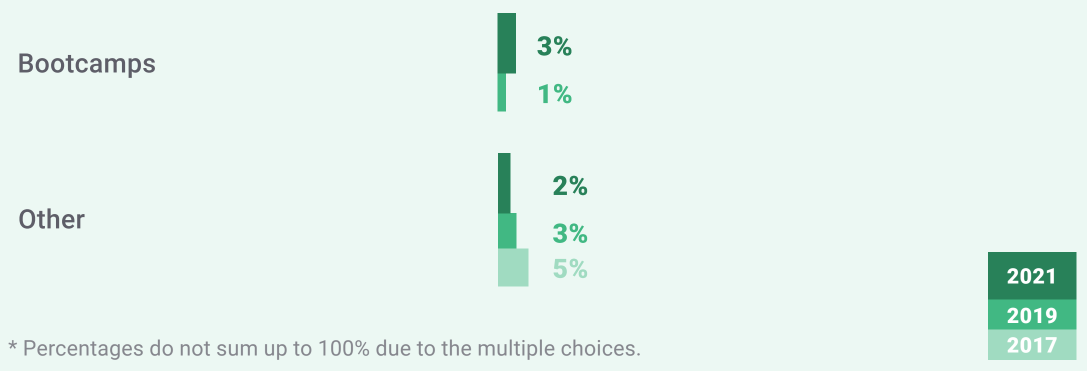
* Percentages do not sum up to 100% due to the multiple choices.

What resources do you use to learn about Vue.js?

Like twice before, the official documentation continues to be the most popular resource among those willing to improve their knowledge of the framework.

LEARNING RESOURCES



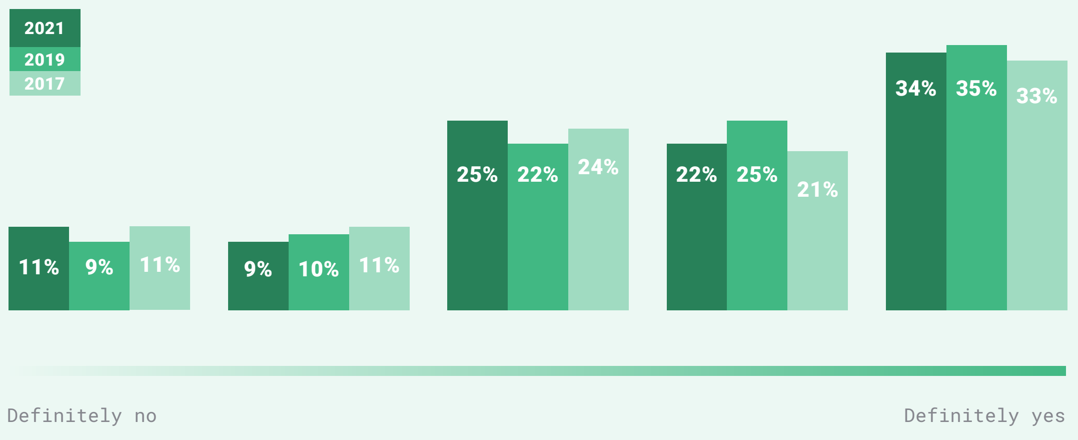


* Percentages do not sum up to 100% due to the multiple choices.

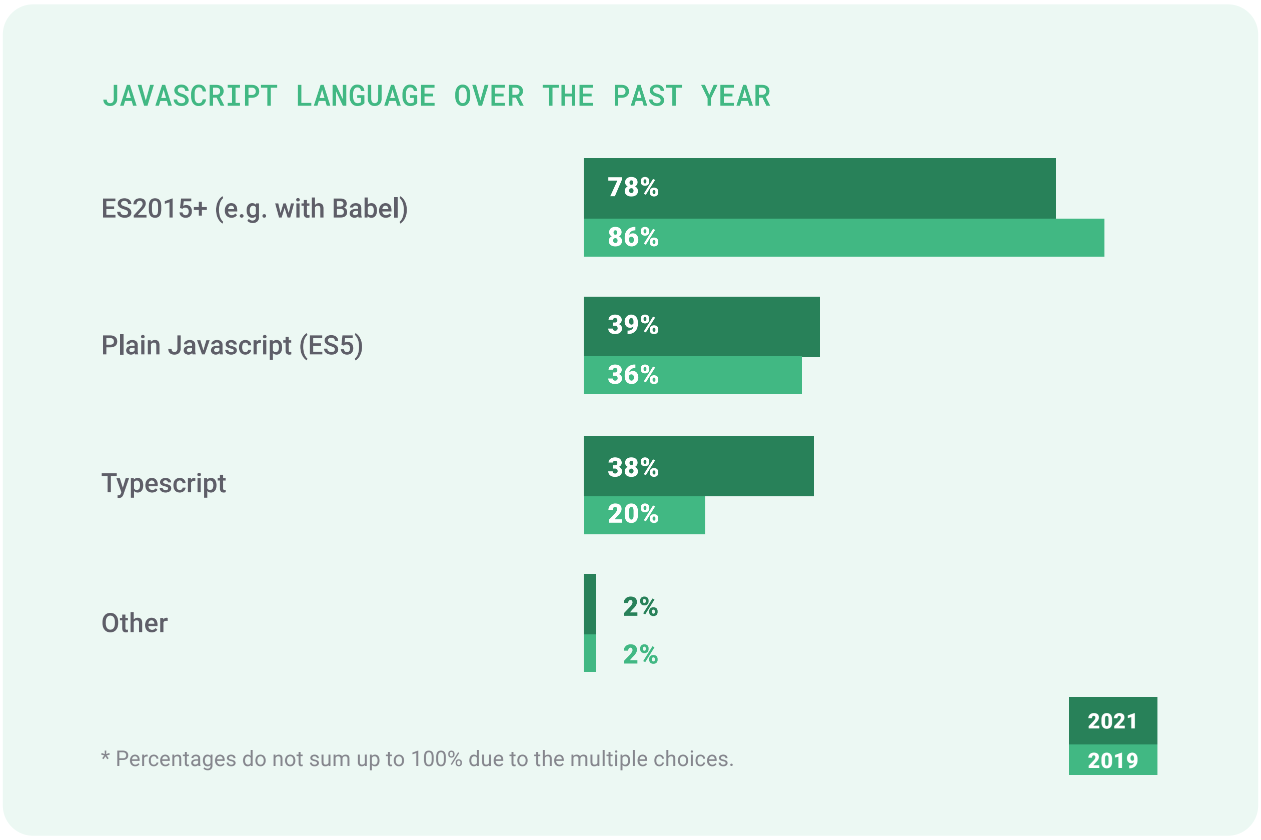
Do you think the number of employees using Vue.js in your organization will increase in the next 12 months?

Over 55% of respondents are convinced Vue.js is going to get even more popular in their organization within the next 12 months.

INCREASE IN THE NUMBER OF EMPLOYEES USING VUE.JS

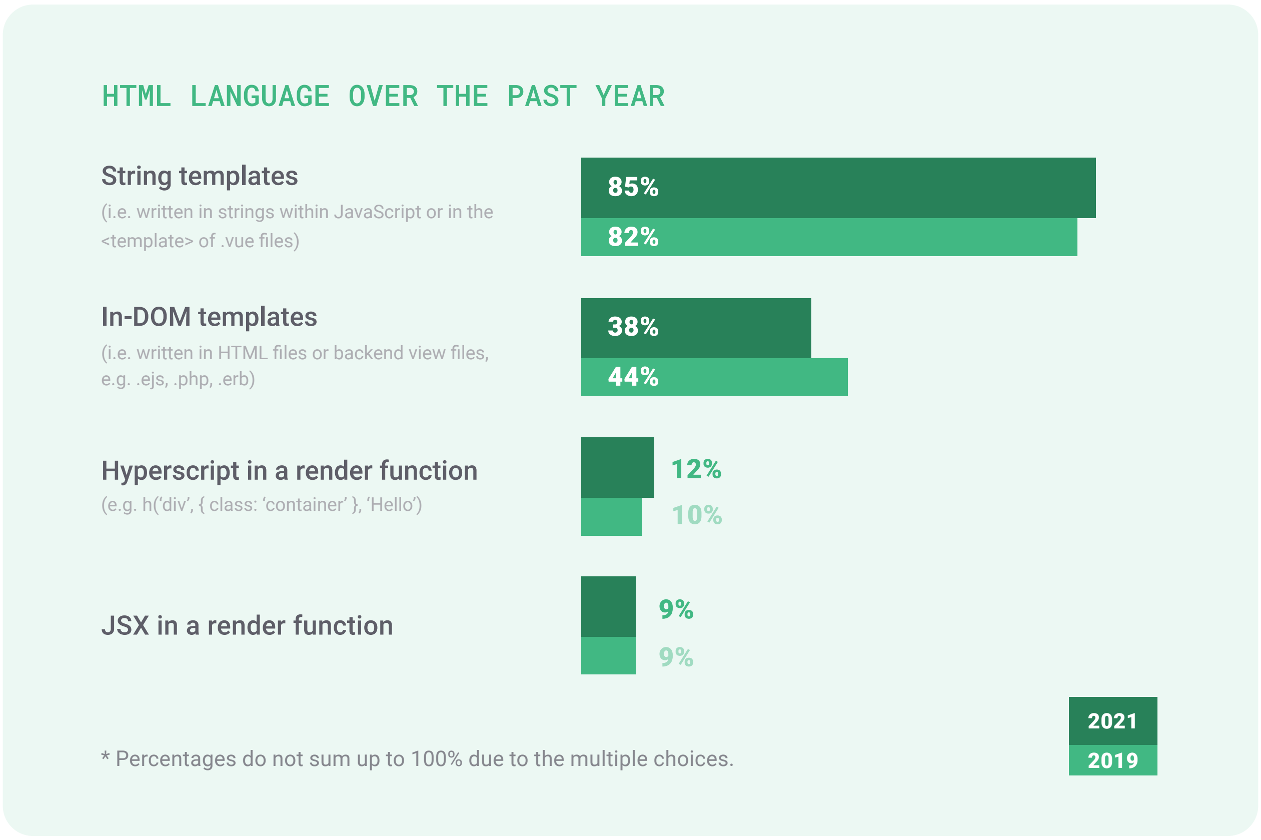


In Vue.js projects over the past year, which of the following have you used for JavaScript?



* Percentages do not sum up to 100% due to the multiple choices.

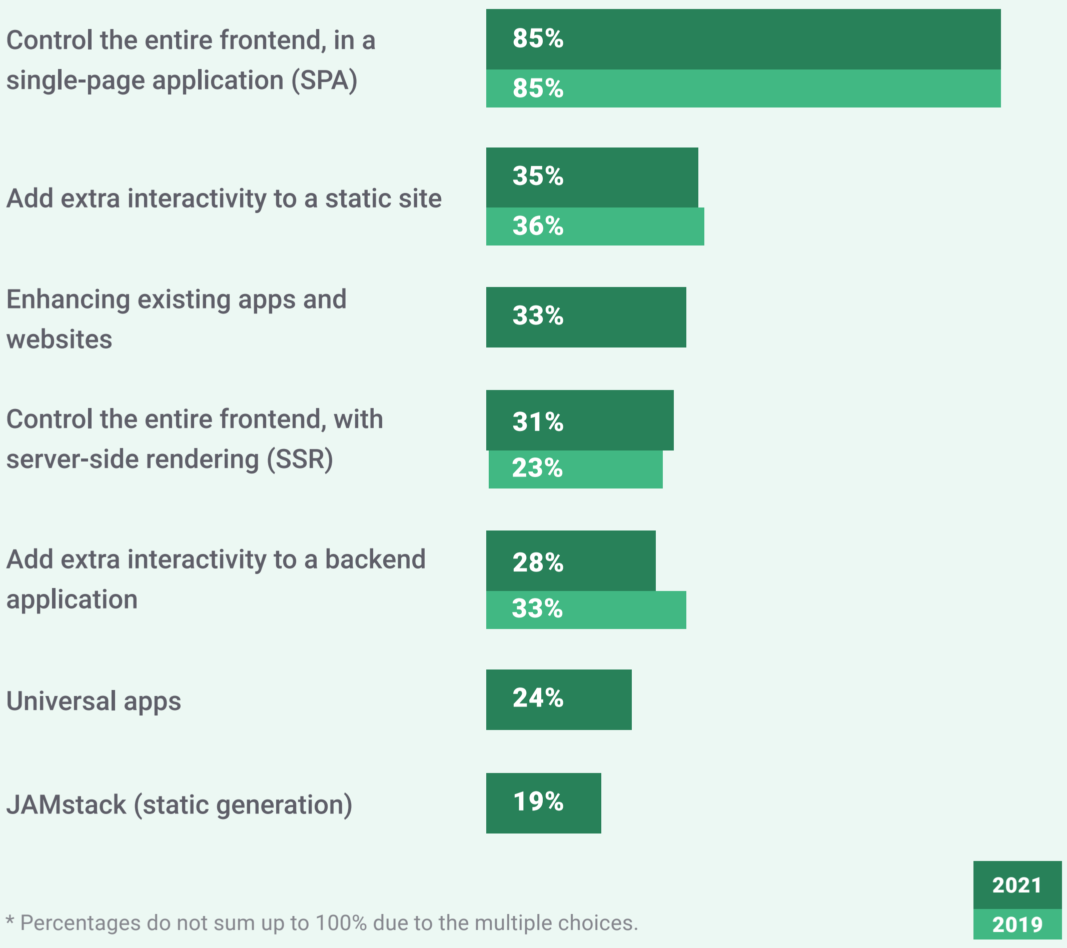
In Vue projects over the past year, which of the following have you used to write HTML?



* Percentages do not sum up to 100% due to the multiple choices.

In Vue.js projects over the past year, which of the following use cases for Vue have you had?

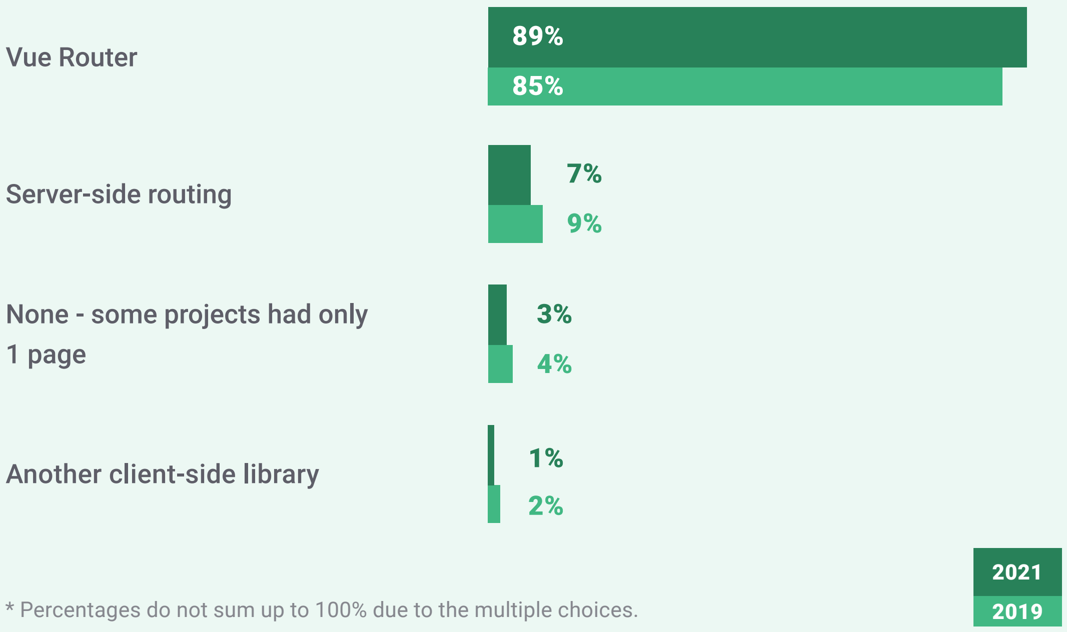
VUE USE CASES OVER THE PAST YEAR



* Percentages do not sum up to 100% due to the multiple choices.

In Vue projects over the past year, which of the following have you used for routing?

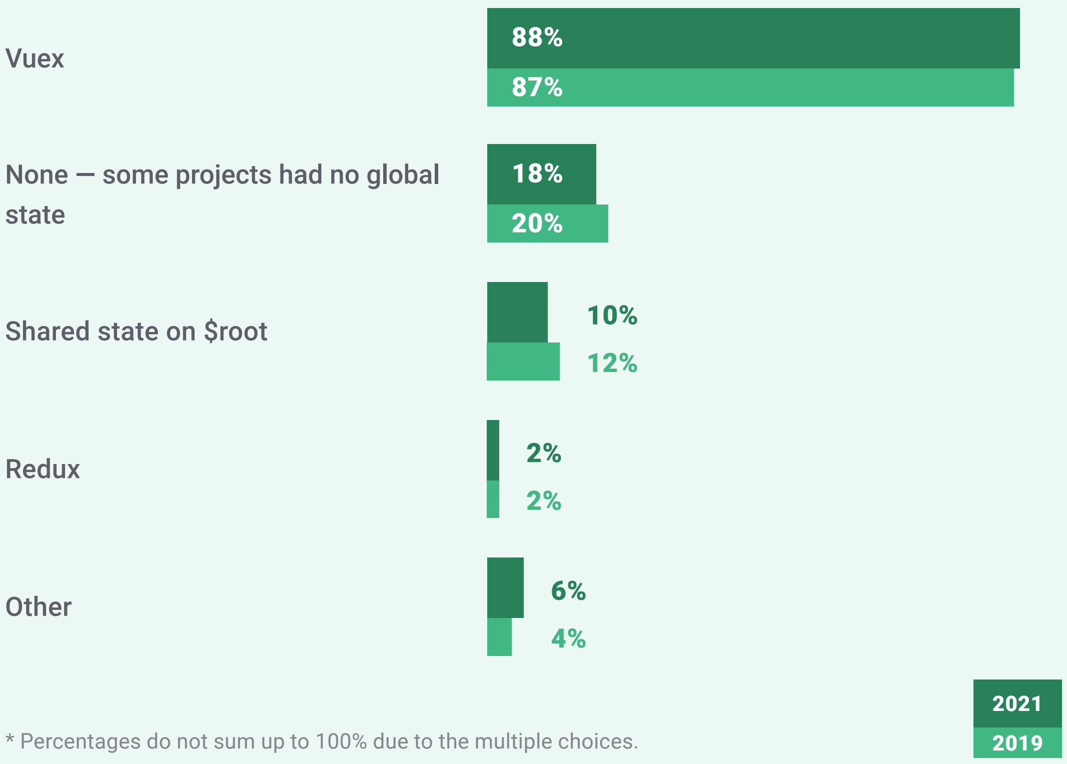
ROUTING TECHNOLOGY OVER THE PAST YEAR



* Percentages do not sum up to 100% due to the multiple choices.

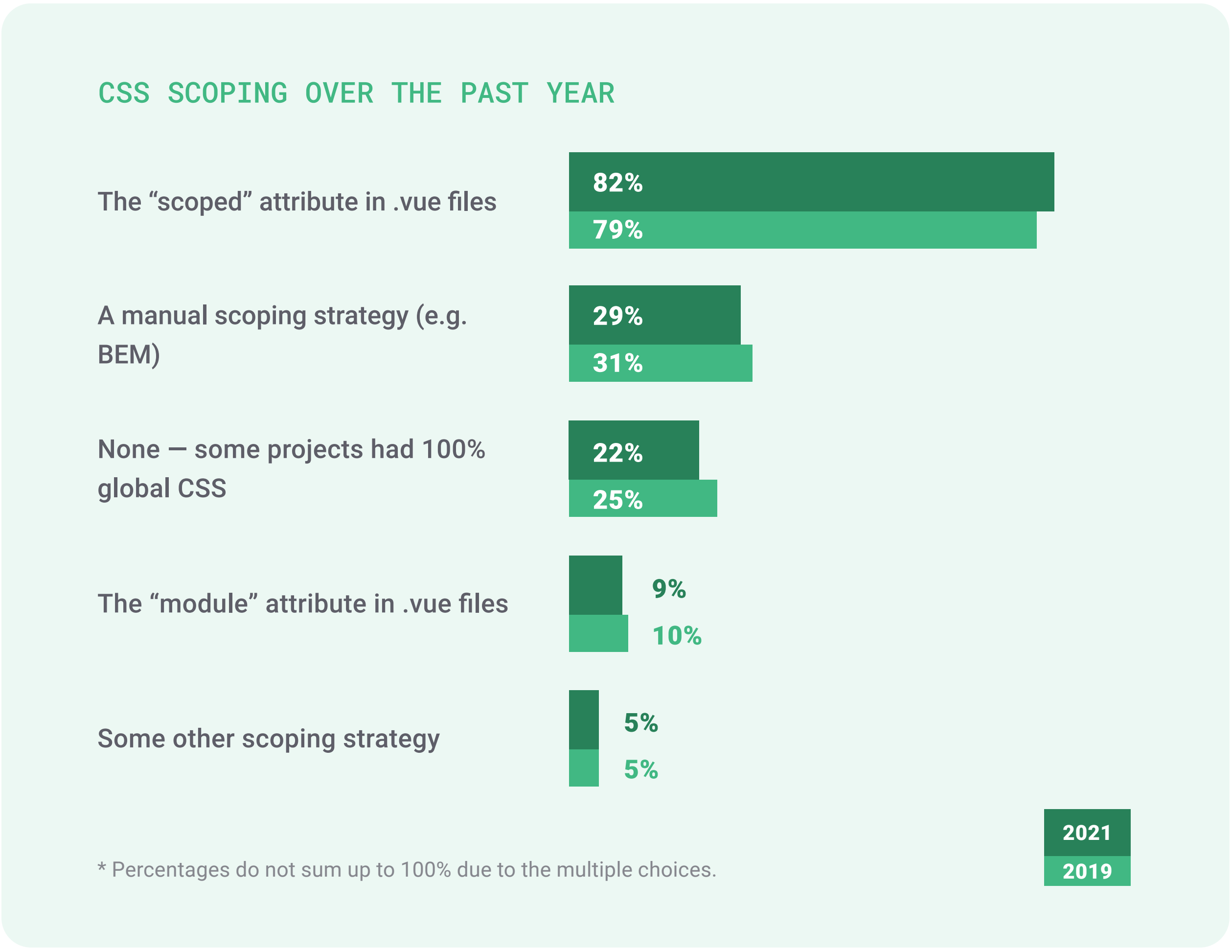
In Vue.js projects over the past year, which of the following have you used for global state management?

GLOBAL STATE MANAGEMENT OVER THE PAST YEAR



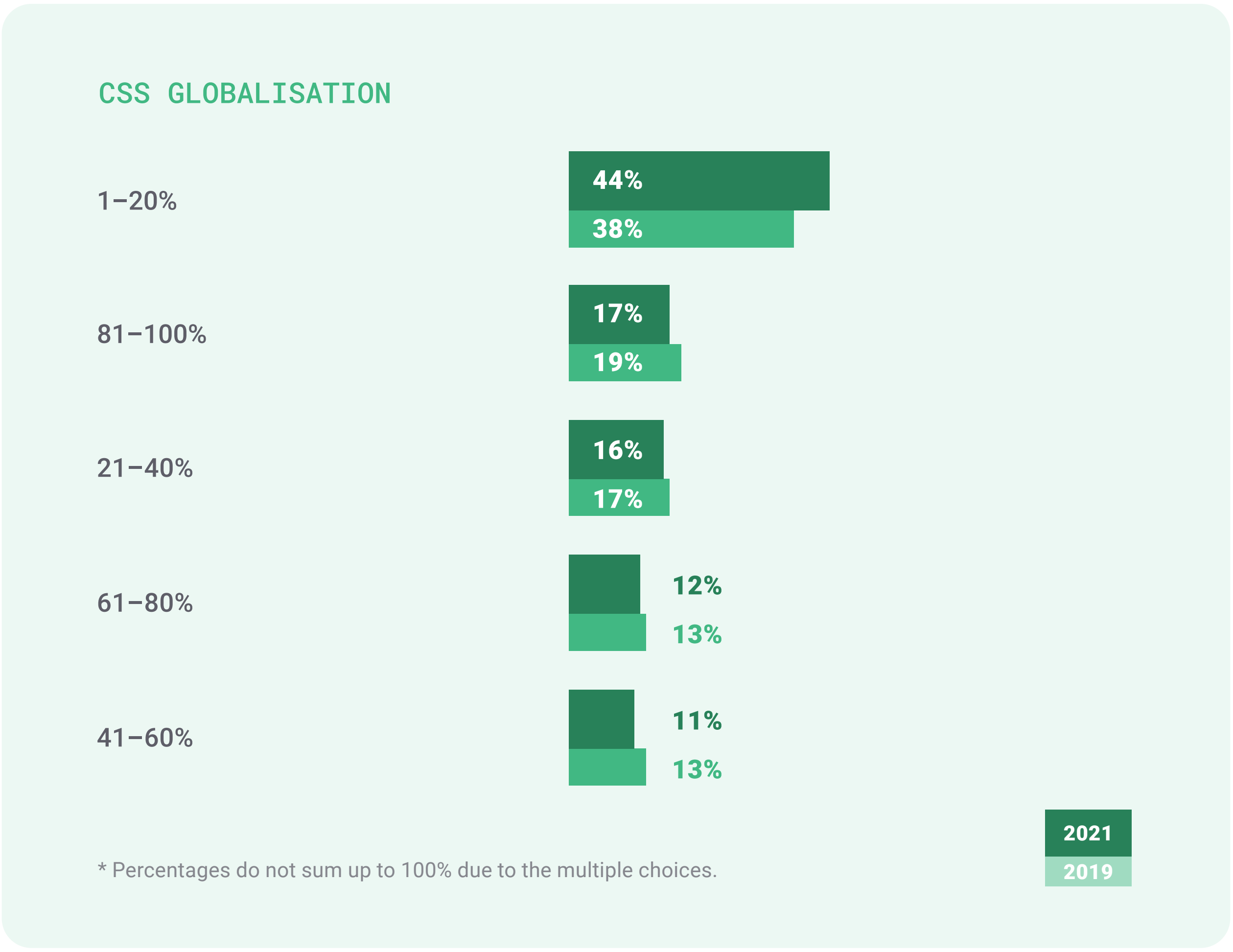
* Percentages do not sum up to 100% due to the multiple choices.

In Vue.js projects over the past year, which of the following have you used to scope CSS?



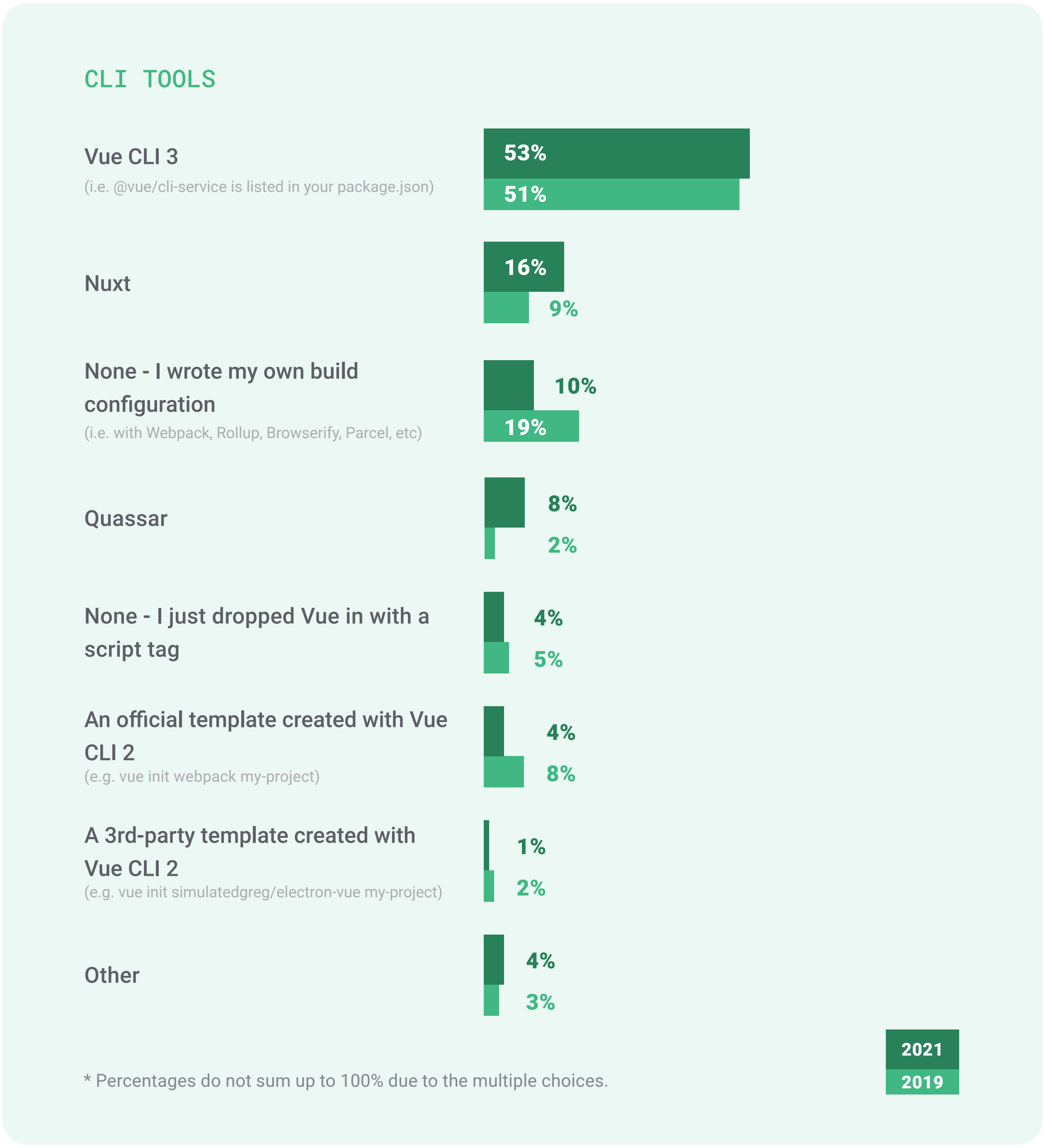
* Percentages do not sum up to 100% due to the multiple choices.

In your last Vue project, how much of the CSS was global?



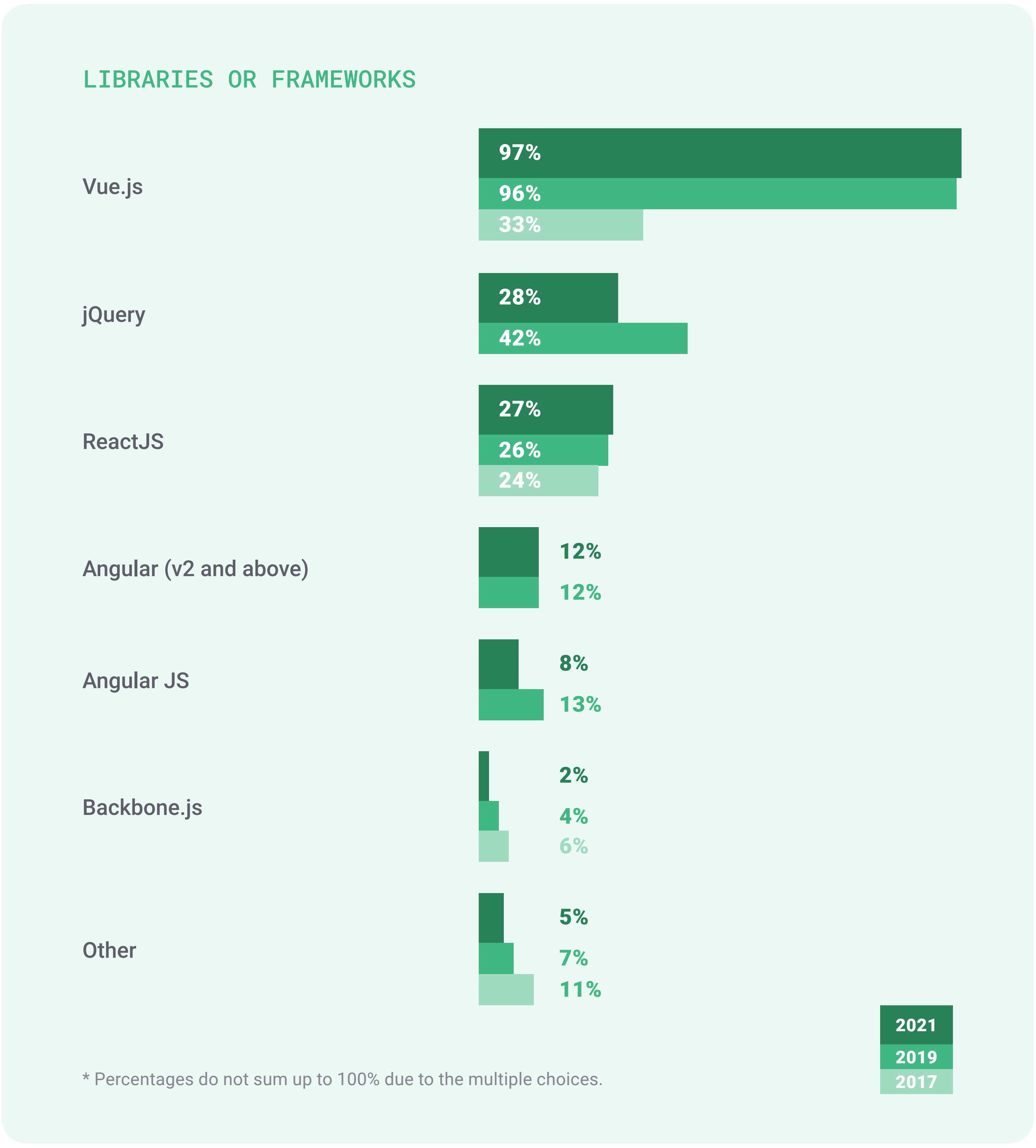
* Percentages do not sum up to 100% due to the multiple choices.

Which CLI tool did you use to create your last Vue project?



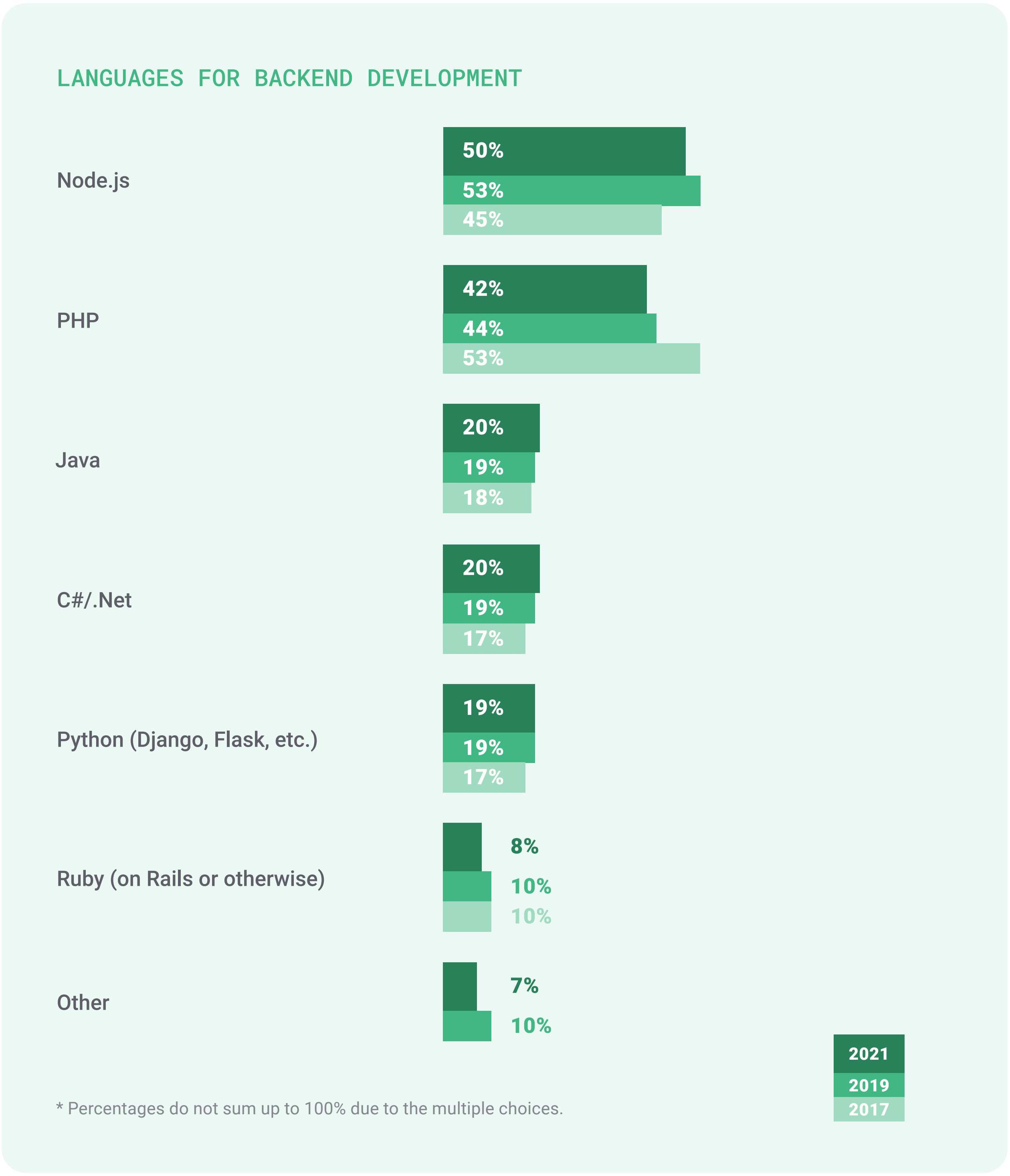
* Percentages do not sum up to 100% due to the multiple choices.

What libraries/frameworks do you use for frontend development?



* Percentages do not sum up to 100% due to the multiple choices.

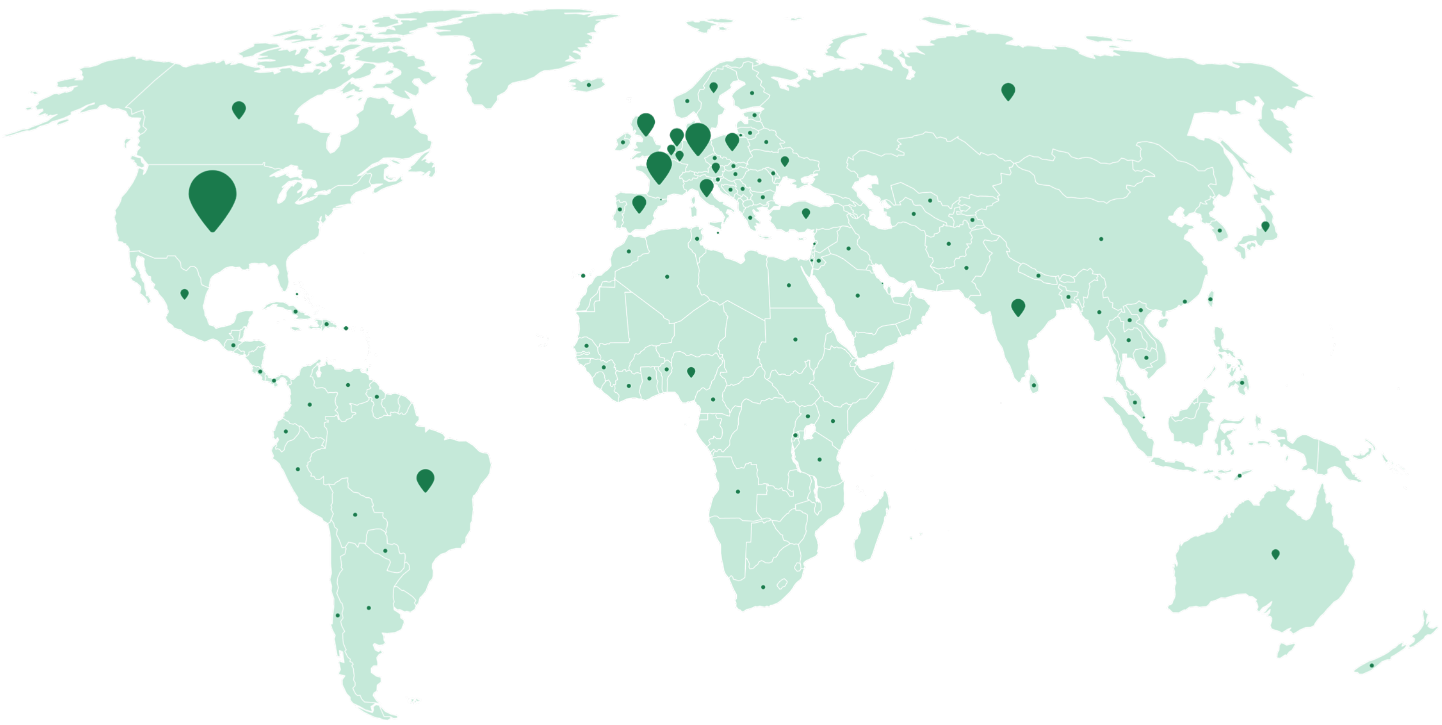
What languages does your organization use for backend development?



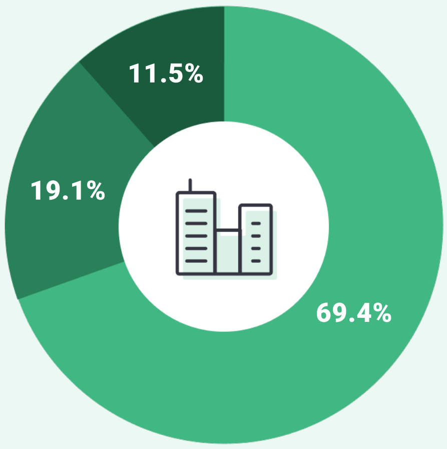
* Percentages do not sum up to 100% due to the multiple choices.

Demographic

We surveyed 1,635 software developers, CTOs, and other technical roles familiar with Vue from 114 countries.



16.2%	USA
7.6%	Germany
7.4%	France
5.7%	Brazil
5.4%	UK
Between 3.5% - 2.8%	India, Italy, Poland, Canada, Netherlands, Spain, Russian Federation, Austria, Sweden, Japan, Ukraine, Belgium, Mexico, Turkey, Nigeria, Switzerland
26%	Other



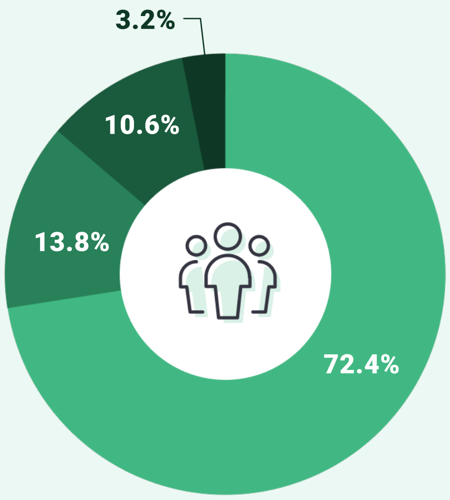
Company size

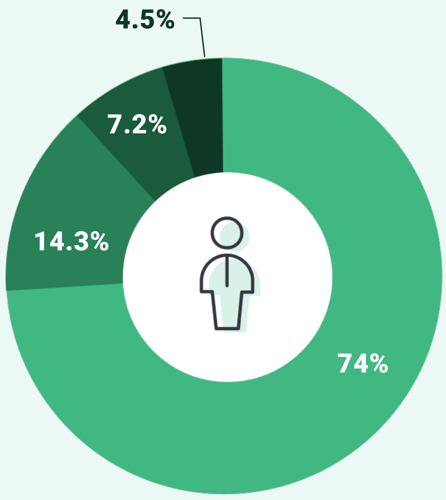
- 🟩 Small and Medium-Sized (<100)
- 🟩 Medium Enterprise (101-1.000)
- 🟩 Enterprise (1.000+)

Team size

(number of teammates)

- 🟩 Small team (2-10 people)
- 🟩 Solopreneur
- 🟩 Medium team (11-25 people)
- 🟩 Large team (25+ people)





Role in organization

- 🟩 Software developer
- 🟩 Chief of Technology Officer
- 🟩 Other
- 🟩 Project Manager

5 Vue Experts on Using Vue

We've reached out to experts working on some exciting Vue-based projects, to pick their brains a bit. They answered five questions that will help you gauge how Vue is used and can be used. The questions we asked were:

- Why do you use Vue and what do you like most about it?
- A new version of Vue was recently released — are you planning to adopt it anytime soon?
- What's the most exciting project you've done with Vue?
- For what type of projects would you recommend to try out Vue?
- What is the most burning problem you'd see Vue as a developing framework tackling next?

Opening this section is a comment from Damian Dulisz — member of the Vue Core Team — about the new Composition API and its implications.

Here you'll find a lot of insightful information, so let's jump in!

Damian Dulisz

Tech Lead at Coursedo
Vue.js Core Team Member
Vue.js Consultant

@DamianDulisz

The introduction of the Composition-API is what excites me the most. Commonly, when people talk about Composition-API, they mention improved application scalability and flexibility. That's true, of course, but the benefits of the Composition-API don't end there.

From a library author perspective, the Composition-API also enables completely new ways of architecting 3rd-party plugins. It is now easier than ever to create libraries that use Vue's reactivity in a way that is more maintainable and flexible. Gone are the hacks of using a Vue instance behind the scenes, which introduced unnecessary complexity. I'm pretty sure we will see new state management libraries as alternatives to Vuex.

Another change that I believe will impact the library landscape is that Vue itself now consists of multiple packages that have different responsibilities: compiler-core, compiler-dom, or reactivity and others. It's now easier than ever to create an alternative compile target, which means we might see some new mobile (native) solutions like React Native!

GitLab

Natalia Tepluhina

Staff Frontend Engineer
at GitLab

@N_Tepluhina

GitLab is a complete DevOps platform, delivered as a single application.

Why do you use Vue and what do you like most about it?

If we speak about the company, GitLab uses Vue because it was the best framework for the iterative migration from jQuery for the setup we had. We were able to make parts of the page interactive without rewriting the whole frontend from scratch.

If we speak about personal reasons, **I use Vue because it's one of the most flexible and progressive tools I know. You can adjust it to almost any case, be it a small app to validate a form or a full-featured enterprise-size application.**

A new version of Vue was recently released - are you planning to adopt it anytime soon?

Yes, but 'anytime soon' may be a good amount of time if we speak about a big company :) GitLab plans a migration but this will take time. I would say the most impressive changes are performance improvement and Composition API.

What's the most exciting project you've done with Vue?

Not that I have done it alone but GitLab's codebase is really exciting to work with.

For what type of projects would you recommend to try out Vue?

Honestly, I don't think there are any limiting factors. Try it for whatever you plan to build.

What is the most burning problem you'd see Vue as a developing framework tackling next?

Right now it would be aligning the ecosystem of plugins and libraries to support both 2.x and 3.x versions.

Valtech

Tim Benniks

Director of Web Development
at Valtech Paris

@timbenniks
timbenniks

Valtech is a global digital agency focused on business transformation.

Why do you use Vue and what do you like most about it?

I stumbled into using Vue for work. I didn't plan on using it as I have a Vanilla JS background. But, when I saw it in action I knew it was a game changer. **More accessible and performant than its competition, Vue saw a fast adoption at my place of work. What I like most about Vue is that it has a shallow learning curve and it allows you to use it in different forms: from agnostic native-like web components to SPA implementations. Working in enterprise for huge global clients, this is a must. I lead teams of developers with different skill sets and cultures. Vue is easy to learn, it is just strict enough to make junior developers successful and it is flexible enough to make architects happy. It's basically a slam dunk in my type of work.**

Vue 3 was recently released - are you planning to adopt it anytime soon? If so, what's the most important change you've noticed?

Personally I have used it for a while and I think the most important change is the flexibility of composition API, performance with Proxies and TypeScript support.

As I work in enterprise, software systems are large and having stable versions of tools is paramount. Therefore we will wait a little to jump on the new version train. However, **on smaller, new projects, we are already using Vue 3.**

What's the most exciting project you've done with Vue?

The L'Oréal WSF (Website Factory) program is the new vision for L'Oréal's digital landscape. L'Oréal is huge and they have an impressive number of websites up and running.

The WSF program aims to deliver high-quality modern websites. All the new sites will share the same tech stack. From continuous integration, hosting, CDN, CMS platform to front-end code and its toolsets. Projects have the same foundation while the design and tone of voice are unique for each website.

My job in the WSF was that of global front-end architect. We decided Vue.js would be used for every L'Orèal website as it enabled us to reach accessibility, SEO and performance KPIs relatively easily. With Vue.js at its center, this is one of the biggest Vue.js implementations globally. There are a bunch of other cool projects which I can't talk about. They are in the luxury space and we do not disclose our clients.

For what type of projects would you recommend to try out Vue?

Vue.js has two applications that I think work very well. First of all, when you have a headless setup where all dependencies are Cloud Native. In this case I would suggest using Nuxt.js in either Universal mode or static with jamstack.

It's other application is with enterprise systems like Sitecore or Adobe AEM. Vue is in the unique position that you do not have to use it as an SPA per se. You can also use Vue to provide a native web component like experience which could then replace the native frameworks these enterprise systems come with. This is not really possible with the competition. However Svelte is also a good contender here.

What is the most burning problem you'd see Vue as a developing framework tackling next?

I had the privilege to speak with Evan You the other day and he mentioned that the developer experience is a point of interest. Front-end technology is starting to catch up with JAVA and dotnet and compile times are starting to be slow. By creating Vite, the Vue team is making big strides in compilation performance and hot module reloading. Hence, a better developer experience.

Storyblok

Samuel Snopko

Head of Developer Relations
at Storyblok

@SamuelSnopko

Storyblok is a headless CMS that helps your team manage content for every use-case: corporate websites, e-commerce, helpdesks, mobile apps, and screen displays.

Why do you use Vue and what do you like most about it?

I like the learning curve - in my opinion, Vue is the natural step in evolution, not a revolution like React and other frameworks. It accepts the HTML and CSS standards. If somebody masters HTML and CSS, the JS & Vue is very easy to adopt.

A new version of Vue was recently released - are you planning to adopt it anytime soon?

I am not planning on adopting it soon. I will wait for Nuxt 3, as I don't have any projects with pure Vue. Meanwhile however, I will keep learning in VueMastery or somewhere else to see what has changed.

What's the most exciting project you've done with Vue?

From what I am allowed to share :)

www.sz-brandstudio.de

www.advertorial.sueddeutsche.de

For what type of projects would you recommend to try out Vue?

I'd recommend it for any project. Vue is mature enough to handle anything on the Web.

What is the most burning problem you'd see Vue as a developing framework tackling next?

Probably transitioning from Vue2 to Vue3 - this would mean also updating all plug-ins, packages and tools all around. However, this will also bring a fresh start to many solutions and projects in the future

Monterail

Artur Rosa

Frontend Developer
at Monterail

@rosickeyy

Monterail is a full-service software development company with 110+ experts on board delivering meaningful software for start-ups, SMBs and enterprises.

Why do you use Vue and what do you like most about it?

I use Vue because it is a very flexible framework that allows you to create applications in a declarative way. Flexibility often comes with a high level of difficulty, but Vue has great documentation and a style guide, which makes it hard to use it incorrectly.

What I like most about Vue is that you don’t have to actually know the framework to read the code of Vue components. Basic knowledge of HTML, CSS and JS is enough to introduce simple changes in Vue applications. Thanks to this, even less experienced developers can easily start working on a Vue project.

Are you planning to adopt Vue 3 anytime soon?

Definitely. I’m watching the Vue ecosystem carefully, especially the tools we use in our projects. As soon as Vue 3 is stably supported by these tools, we can start migrating some of the projects.

I already had the opportunity to use Vue 3 on small internal projects and one of the changes I am very happy about is the better adoption of TypeScript by the Vue community.

What’s the most exciting project you’ve done with Vue?

Some time ago I had the opportunity to work on a 3D interior designer, iDesigner. It was an application in which the user could arrange their new apartment. This is a great example of how flexible, efficient and fascinating Vue can be.

In this project, Vue acted as scene manager of the entire editor. It’s reactivity system was used to declaratively describe the scene graph. The user could work on a 3D scene made with WebGL as well as on a 2D scene that was implemented using SVG. In both cases, scene objects and their properties were two-way bounded with the corresponding Vue components that represented them. Of course, the application interface was also created in Vue.

For what type of projects would you recommend to try out Vue?

I recommend Vue especially for projects that will be created by teams of developers with different levels of experience. I'm not saying that I don't recommend Vue for senior teams, but in the first case you'll very quickly see Vue's advantage over other popular frameworks.

What is the most burning problem you'd see Vue as a developing framework tackling next?

It seems to me that there is still some room for improving the developer's experience. Especially for TypeScript users. I'm glad that there are people who are currently working on this part of the Vue ecosystem.

Wikimedia Deutschland

Tonina Zhelyazkova

Software Developer
at Wikimedia

@tonina_zh

Non-profit organization dedicated to the dissemination of free knowledge.

Why do you use Vue and what do you like most about it?

At Wikimedia Deutschland we maintain and develop open source software such as Wikibase and Wikidata. Our frontend code was originally built using a home-brewed framework based on jQuery and other JavaScript libraries. While this was the best choice at the time when the core part of the software was built, it is not the best choice of technology these days, because there are better frameworks available. Our legacy frontend code has been hard to maintain and is not well-documented. Building new features has been unnecessarily complex due to being the difficulty to onboard new developers in them, and having unexpected bugs show up in the outdated code.

To tackle these problems, we decided to adopt Vue.js gradually by creating and growing features which live alongside the legacy front end code, instead of doing a big overhaul of all our code.

We liked Vue.js because it allows us to organize our code into single file components and it has rich, easy to follow documentation. Another major factor in choosing Vue was the native SSR support it provides, allowing us to future proof ourselves from this requirement.

At Wikimedia Deutschland we believe in the power of open source software. We like to build upon the base created by the open source community, therefore we've decided to use Vue.js as the well-established open source framework. All software we develop is also open source, and we strive to contribute back to projects we use whenever possible.

A new version of Vue was recently released - are you planning to adopt it anytime soon?

We've been following the Vue.js framework and its progress closely since we started using it at WMDE and we're very excited about Vue 3 and the new Composition API.

We intend to migrate from Vue 2 to Vue 3. We are considering doing an immediate step by using the composition API Vue 2 plugin.

As our software is used by a significant number of IE11 users, we would be able to use Vue 3 itself as soon as IE11 support is provided.

What's the most exciting project you've done with Vue?

All projects we've done with Vue are exciting in their own way, but the most notable one has to be the product called Data Bridge. It's a frontend feature which runs on Wikipedia and connects it to Wikidata, allowing editors to change Wikidata items through Wikipedia.

It was particularly exciting to work on it because we used Vue with TypeScript and it is the very first project to bring TypeScript to Wikipedia.

For what type of projects would you recommend to try out Vue?

Vue is easy to integrate with existing projects using different libraries. Like in Wikimedia Deutschland's case, if you have a large codebase which uses different technologies for its frontend, and you'd like to start incrementally improving it, Vue is perfect for the job.

Vue is also perfect for building SPAs from scratch, because its community has built tooling and libraries that solve common problems in front end development, which allow you to focus on the essentials of your applications.

What is the most burning problem you'd see Vue as a developing framework tackling next?

I can't think of anything. So far, we haven't had a problem which Vue cannot solve for us.

Vue Storefront

Filip Rakowski

CTO & Co-founder
at Vue Storefront

Vue Storefront is the most popular framework for building eCommerce Storefronts that can work with any eCommerce platform and third-party service.

Why do you use Vue and what do you like most about it?

I started using Vue more than 3 years ago, just after v2 got released. Back then I was new to the modern frontend frameworks, world with previous experience in AngularJS and Knockout. I spotted Vue in some article on Medium, tried it and it just felt so much better than AngularJS. Thanks to the similarities between these two frameworks, it took me just a few days to become fluent and I was making far fewer mistakes while producing much cleaner code so it also helped me to be more productive and confident. At that time simplicity and approachability were the main reasons that gauged my interest. Right now with Vue 3 and Composition API, I'd say flexibility is a much bigger asset and this is what I value the most as an experienced developer.

A new version of Vue was recently released - are you planning to adopt it anytime soon?

Yes! We started using new Vue 3 APIs in Vue Storefront even before Vue 3 was usable - right after the release of the CAPI plugin. At that time we were figuring out new architecture so the timing was perfect. Even though we experienced a lot of struggles with the plugin itself, the API is absolutely amazing and brings a lot of improvements and new possibilities - especially for the library authors.

We're great fans of domain-driven design. Before the Composition API we had to invent our own ways of utilizing this approach. We sometimes felt it worked against the framework and its good practices. With Composition API these limitations are no longer there. I generally feel that Vue 3 shines the most when you compare the limitations of these two versions in terms of architecture or building APIs. With all the tiny bits of Vue like the reactivity system exposed as independent packages, Composition API with additions like customRef or shallowRef it's hard to imagine a frontend use case that is hard or unintuitive to achieve with Vue.

What's the most exciting project you've done with Vue?

For the last few years, we've been building Vue Storefront and a whole ecosystem of tools around it - UI library and visual UI builder to name a few. I find all of them really exciting but I think Vue Storefront Next - which is a rewrite of Vue Storefront with Composition API itself - is the one where Vue shines the most. Thanks to the declarative approach (just like Vue) and Composition API, we managed to make an extremely complex area (which is eCommerce frontend development) into something that is easily approachable, even to junior developers. I don't think it would be possible to that degree in any other framework.

For what type of projects would you recommend to try out Vue?

With Vue 3? Almost all of them! Thanks to the granularity of Vue 3 you can fit it into almost any context. It's obviously a great choice for SPAs, just like it was with Vue 2.

It's a great choice for Native apps as well with tools like Nativescript or Weex. Improvements in Vue 3 will also make it much easier to render native components in non-Web environments so we could expect these tools to be even better.

What is the most burning problem you'd see Vue as a developing framework tackling next?

If you add a new way of interacting with the framework internals while still keeping the old one it always leads to problems. Especially if you introduce the new API, yet still promote the old one in the documentation. People could get confused. I'm sure not all library authors will embrace both Composition API and Options API in their libraries. In many cases, we will see only one approach being exposed in the documentation and via public APIs. Because of that, it will be much harder to build apps following just one of the component APIs. It could lead to fragmentation or even duplication of the ecosystem which will make it much harder to navigate through it. I'm curious how the Vue community will approach this.

Vue Comparison with Other Technologies

Intro

The landscape of JavaScript frameworks and libraries is vast and rich in possibilities. It seems as if there are half as many frameworks as there are developers. OK, maybe that's a little over the top, but you know what we mean.

So — why Vue? Why not one of its competitors like React, Angular, jQuery? Or something less popular but better fitted to your use case? (Preact, Svelte, Knockout, Polymer anyone?)

We're not here to convince anyone to pick Vue against their best interest. This section will provide an overview, as objective as possible, of the most used (and up-to-date) JavaScript libraries and frameworks: Vue, React, Angular, and Svelte.

It should help you answer questions such as: should you learn and implement Vue? What projects is it best used for? Is it future-proof?

Technology overview

Vue

The first version of Vue was built by one developer, Evan You in 2014 to improve on available JS tools. A former Google employee in Google Creative Labs, Evan You wanted to create a framework that combined the best approaches towards frontend development from Angular, Ember, and React with other features that made writing Web apps faster, easier, and more pleasant.

From the get-go, Vue was a truly open source project, relying on the community, contributors, and crowdfunding to move forward.

It shares several major similarities with React, like:

- **Virtual DOM** — React and Vue work with a lightweight copy of the actual DOM, updating only those fragments of the website that need to be updated, and saving the time and resources that heavy DOM manipulations otherwise consume.
- **Component-based UI development** — both Vue and React have considerable libraries of components that facilitate code reuse,

improve developer productivity, and speed up the development process.

- **Focus on the view library** — separate libraries for routing, state management, etc.

But Vue also has some distinct features which make it stand out:

- **Gentle learning curve.**
- **Elasticity —** Vue can be both: easily integrated with existing projects and used as the main framework in huge apps.
- **Progressiveness —** Vue can be incrementally implemented into a project, you don't need to rewrite it from scratch.
- **Superb documentation.**

React

React was created in 2013 for the purpose of meeting specific needs at Facebook, and it continues to be maintained by the tech giant. In the past there were doubts regarding React’s license; currently, however, the tool operates under the MIT License — which makes it open source.

React’s corporate backing, especially from such a major player, indicates stability in the future and implies that React will continue to be developed with long-term support.

Angular

Angular came to be when Google's team decided to completely rewrite AngularJS in 2014. Their aim: solving scalability issues. With the help of the community and other corporations, Angular came to be a widely-used framework.

The framework is TypeScript-based, uses modules, components, and both property and event binding.

Some of Angular characteristics:

- Angular has many predefined modules, classes and services which help you build apps faster, but makes the entry treshold much higher than other frameworks.
- Angular introduced a new generation rendering and compiling pipeline called Ivy. It enables quick re-building and lowers payload size.
- Angular has built-in features that help you deal with lazy-loaded modules and improve app's performance.
- Angular uses Incremental DOM instead of Virtual DOM.

Svelte

Svelte was created by Rich Harris in 2016 and is now maintained by him and the Svelte core team.

This framework is the newcomer in this comparison — how its fate will turn out, only the future can tell. Its main focus is on: using less code, eliminating virtual DOMs to improve performance, and running at build time (converting components into imperative code that updates the DOM, improving the performance further).

Vue

Nintendo, zoom, Wizz, GitLab, Alibaba.com, mi, euronews., Adobe

React

NETFLIX, ATLASSIAN, Uber, PayPal

Angular

Google, Santander, Forbes, traduora, Office, Project Clarity

Svelte

teamspeak, openradio.app, WireGuard VPN, Workingoo, ChromeStats, Houses of the World

Popularity and support

Vue, React, and Angular

The popularity of these technologies has been covered in the Vue Today section, so to recap: React is the most used framework right now, and its popularity is growing, as well as Vue's.

- **161k** GitHub starts
- **13.6k** GitHub commits
- **1.5k** GitHub contributors
- **257k** questions on Stack Overflow
- **2609k** live websites

React's communities tend to be described as more fragmented than those of Vue for example, but there are a few places where you can go to ask any question: DEV's React community, Hashnode's React community, Reactiflux chat on Discord, and Reddit.

- **177k** GitHub stars
- **3.1k** GitHub commits
- **382** GitHub contributors
- **67k** questions on Stack Overflow
- **1189k** live websites

When looking for answers to Vue-related questions it's always worth taking a look at these communities: Vue forum, Discord, and Reddit. The Vue community is famed for always being ready to help.

- **69k** GitHub stars
- **19.1k** GitHub commits
- **1.2k** GitHub contributors
- **235k** questions on Stack Overflow
- **115k** live websites

Angular communities tend to gather in these places: Slack, Gitter, Stack Overflow, Reddit.

- **43k** GitHub stars
- **5.9k** GitHub commits
- **342** GitHub contributors
- **1200** questions on Stack Overflow
- **5k** live websites

As for available jobs, the number of vacancies on stackoverflow.com jobs are as follows (descending order): React, Angular, Vue. We couldn't find any openings that require specifically Svelte — it's more of a nice-to-have skill.

Documentation

Vue

If you've heard anything about Vue, you're aware of its great documentation. There, the creators describe in a clear and accessible way everything you need to know, taking you step-by-step through solutions. You'll also gain insight into Vue's ecosystem, and scaling options. Vue's clear syntax compliments the usefulness of the documentation. Additionally, you get an official styleguide which saves developers a lot of frustration when changing projects.

React

React's documentation remains far behind Vue's in terms of accessibility and structure. It goes through basics and advanced uses, but navigating all the material isn't nearly as smooth as with Vue. There's no overview of React's ecosystem.

Angular

Angular's documentation is known for its complexity and can be quite overwhelming. In 2016, the documentation was overhauled to be more user-friendly, but diving into chapters, subchapters, and sub subchapters can still confuse many developers. The documentation tries its best to guide the user through the development process but the steep learning curve of Angular compounds the issue.

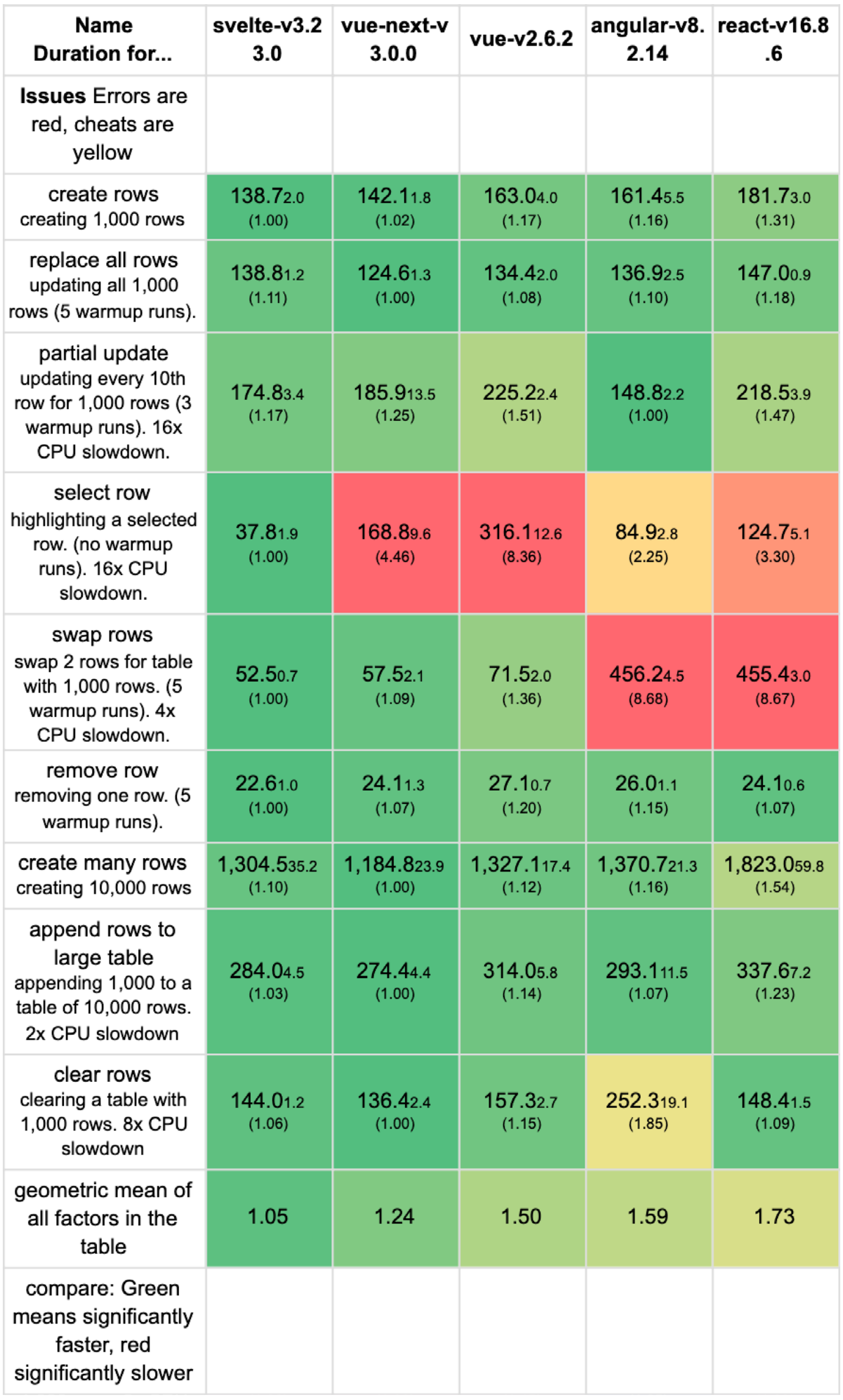
Svelte

Svelte's documentation is neatly divided into sections about operators, processes, and functions. Those entirely unfamiliar with the framework can start with an interactive tutorial or by asking around in the Discord chat room. The creators are doing their best to introduce Svelte concepts in an accessible way, and it shows — Openbase users rank Svelte documentation high.

Performance

Raw performance data can be easily found on Stefan Krause's website. The most up-to-date version, tested on Razer Blade 15 Advanced (i7-8750H, 64 GB RAM, Fedora 33 beta (Linux 5.8.13-300, mitigations=off), Chrome 86.0.4240.75 (64-bit), returns following:

Duration in milliseconds ± 95% confidence interval (Slowdown = Duration / Fastest)



Name Duration for...	svelte-v3.2 3.0	vue-next-v 3.0.0	vue-v2.6.2	angular-v8. 2.14	react-v16.8 .6
Issues Errors are red, cheats are yellow					
create rows creating 1,000 rows	138.7 2.0 (1.00)	142.1 1.8 (1.02)	163.0 4.0 (1.17)	161.4 5.5 (1.16)	181.7 3.0 (1.31)
replace all rows updating all 1,000 rows (5 warmup runs).	138.8 1.2 (1.11)	124.6 1.3 (1.00)	134.4 2.0 (1.08)	136.9 2.5 (1.10)	147.0 0.9 (1.18)
partial update updating every 10th row for 1,000 rows (3 warmup runs). 16x CPU slowdown.	174.8 3.4 (1.17)	185.9 13.5 (1.25)	225.2 2.4 (1.51)	148.8 2.2 (1.00)	218.5 3.9 (1.47)
select row highlighting a selected row. (no warmup runs). 16x CPU slowdown.	37.8 1.9 (1.00)	168.8 9.6 (4.46)	316.1 12.6 (8.36)	84.9 2.8 (2.25)	124.7 5.1 (3.30)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4x CPU slowdown.	52.5 0.7 (1.00)	57.5 2.1 (1.09)	71.5 2.0 (1.36)	456.2 4.5 (8.68)	455.4 3.0 (8.67)
remove row removing one row. (5 warmup runs).	22.6 1.0 (1.00)	24.1 1.3 (1.07)	27.1 0.7 (1.20)	26.0 1.1 (1.15)	24.1 0.6 (1.07)
create many rows creating 10,000 rows	1,304.5 35.2 (1.10)	1,184.8 23.9 (1.00)	1,327.1 17.4 (1.12)	1,370.7 21.3 (1.16)	1,823.0 59.8 (1.54)
append rows to large table appending 1,000 to a table of 10,000 rows. 2x CPU slowdown	284.0 4.5 (1.03)	274.4 4.4 (1.00)	314.0 5.8 (1.14)	293.1 11.5 (1.07)	337.6 7.2 (1.23)
clear rows clearing a table with 1,000 rows. 8x CPU slowdown	144.0 1.2 (1.06)	136.4 2.4 (1.00)	157.3 2.7 (1.15)	252.3 19.1 (1.85)	148.4 1.5 (1.09)
geometric mean of all factors in the table	1.05	1.24	1.50	1.59	1.73
compare: Green means significantly faster, red significantly slower					

Source: Stefan Krause

Startup metrics (lighthouse with mobile simulation)

Name	svelte-v3.23.0-keyed	vue-next-v3.0.0-keyed	vue-v2.6.2-keyed	angular-v8.2.14-keyed	react-v16.8.6-keyed
consistently interactive a pessimistic TTI - when the CPU and network are both definitely very idle. (no more CPU tasks over 50ms)	1,954.6 0.5 (1.00)	2,114.5 11.8 (1.08)	2,277.3 1.7 (1.17)	2,845.2 5.3 (1.46)	2,582.3 0.7 (1.32)
script bootup time the total ms required to parse/compile/evaluate all the page's scripts	16.0 0.0 (1.00)	16.0 0.0 (1.00)	16.0 0.0 (1.00)	133.1 3.0 (8.32)	16.0 0.0 (1.00)
total kilobyte weight network transfer cost (post-compression) of all the resources loaded into the page.	145.8 0.0 (1.00)	196.1 0.0 (1.34)	210.8 0.0 (1.45)	302.1 0.0 (2.07)	261.0 0.0 (1.79)
geometric mean of all factors in the table	1.00	1.13	1.19	2.93	1.33

Source:
Stefan Krause

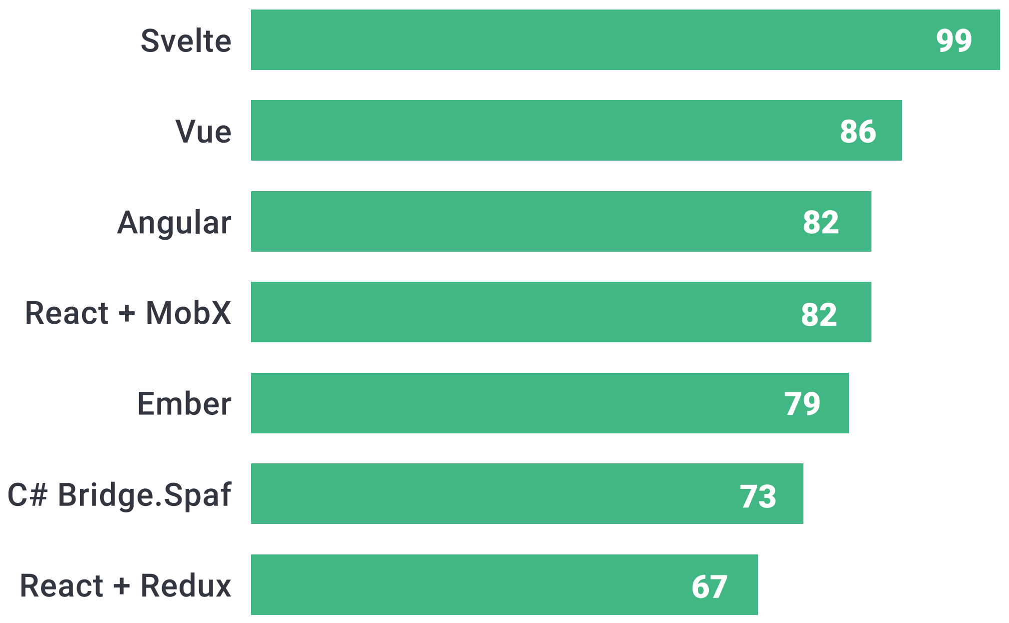
Memory allocation in MBs ± 95% confidence interval

Name	svelte-v3.23.0-keyed	vue-next-v3.0.0-keyed	vue-v2.6.2-keyed	angular-v8.2.14-keyed	react-v16.8.6-keyed
ready memory Memory usage after page load.	1.1 (1.00)	1.2 (1.14)	1.2 (1.14)	2.7 (2.50)	1.3 (1.21)
run memory Memory usage after adding 1000 rows.	2.7 (1.00)	3.5 (1.32)	4.0 (1.49)	5.1 (1.89)	4.3 (1.61)
update each 10th row for 1k rows (5 cycles) Memory usage after clicking update every 10th row 5 times	3.0 (1.00)	3.7 (1.23)	4.4 (1.45)	5.5 (1.81)	5.1 (1.68)
replace 1k rows (5 cycles) Memory usage after clicking create 1000 rows 5 times	3.2 (1.00)	4.1 (1.28)	4.6 (1.41)	6.0 (1.84)	6.6 (2.03)
creating/clearing 1k rows (5 cycles) Memory usage after creating and clearing 1000 rows 5 times	2.4 (1.00)	2.6 (1.10)	2.6 (1.10)	4.3 (1.79)	3.6 (1.50)
geometric mean of all factors in the table	1.00	1.21	1.31	1.95	1.58

Source:
Stefan Krause

Svelte is ahead of Vue in many aspects, but the differences tend to be slight. React beats Angular in 12 out of 20 metrics.

A more readable (but less in-depth) comparison was written by Jacek Schae for Daily JS (a score of 100 means the tested website was lightning-fast.)



Source: Medium

Looking at these two sources, we can draw a draft performance scoreboard that looks like this:

- Svelte
- Vue
- React
- Angular

With that being said, we have to remember that both React and Vue are considered exceptionally fast in day-to-day use. This means that Svelte has virtually no hiccups at all, while Angular occasionally gives developers a hard time (performance-wise).

Scalability

Vue

Vue is a progressive framework that can adapt to multiple environments. It can be added to an existing multi-page application to make it more interactive, but can just as well be used to build large-scale single-page applications or even universal apps, thanks to the state of the art server-side rendering capabilities.

Also, the Vue CLI comes with a ready-made scaffolding that allows you to start a new project in no time, taking care of all the necessary build tools. Thanks to plugins, the Vue CLI makes it super easy to extend the setup with additional libraries (like TailwindCSS, GraphQL, routing and state management) and tools (like Cypress for e2e testing). When you install project dependencies, you implement them as plugins, so even if some standards change, you simply update the dependency.

As for the companion libraries for state management and routing, in Vue they are a part of the core library, so they are officially supported and up-to-date (unlike React, where they are community-based).

React

React, because of its impressive community, has a lot to offer. It can build multi-page apps and single page ones.

The React's CLI comes out a bit poorer when compared to Vue's. You cannot customize your project after a generation, you can only use a single page app template, and you cannot generate projects from user-built presets.

Other than that, React is immensely useful for building scalable web apps (look at Facebook).

Angular

Just as Vue and React are purpose-built for creating interactive web apps, Angular aims primarily at being scalable. Its component-based structure can be further optimized using feature modules. This goes a long way when building really large apps. Furthermore, dependency injections in

Angular allow you to design apps that can be easily extended or have their functionalities refactored.

With that being said, the architecture of Angular components has to be carefully planned and thought-through. This can sometimes generate an overhead too large for a certain project. You can build complex Angular architectures, but for them to be effective, you need to design them before you start developing. If you've never used Angular before, learning all the pros and cons of each architecture design can be daunting.

Svelte

In theory, looking at the performance numbers, and the fact that Svelte aims at using less code and should be more readable than other frameworks, it should be the best candidate for scaling applications.

In practice, however, another core characteristic of Svelte — lack of virtual DOM — places a big caveat on code design. Two sources state that Svelte loses its upper hand in being super lightweight (compared to React) when reaching an inflection point: 137 KB or 120 KB. This means that to benefit from Svelte the developers need to wisely use code-splitting and avoid building big single pages.

Syntax

Vue and React

One of the biggest differences between Vue and React is the way the view layer is built.

By default, Vue uses HTML templates, but there's an option to write in JSX.

In React, on the other hand, there's solely JSX. Vue's traditional separation of concerns into HTML, CSS, and JS makes it easier (even for beginner frontend developers) to learn how to create Web applications. HTML templates are also familiar to most Web designers, and improve collaboration between developers and designers.

React's JavaScript Expressions (JSX) combine HTML and sometimes CSS together into JavaScript. This XML-like syntax lets developers build self-contained UI components with view-rendering instructions included. Two main advantages of this solution are the ability to use the entire strength of JS to build the view layer, and having a more advanced tooling support for JSX.

However, working with HTML when creating the view layer feels more natural to many developers, and when they need to, Vue gives them the option to switch to JSX or to a simple render function.

It is also possible to use TypeScript with Vue and with React, although this requires a bit of a set-up.

Angular

When it comes to syntax, Angular is based on TypeScript. One of the most prominent and valued features of TypeScript is its type annotations. It's easier to track bugs on compile time, although some point to more syntax noise because of that.

Because of the above, Angular is recommended for big projects where many developers work together (interfaces and access modifiers), and for those who prefer compile time type checking.

Svelte

Svelte aims at reducing the amount of code developers need to write when compared to other JS frameworks, and this is something that Svelte achieves without a doubt. The trade-off, however, is a syntax that may confuse many developers.

In Svelte syntax, the derived and computed states of a variable are kept under the mask and calling the computed state with only the variable name feels for some developers somewhat unnatural. Passing properties also tends to confuse developers, as you have to export a basically imported declaration. Conditionals and loops are another thing on a list that developers don't like — they look and feel archaic.

On the upside, Svelte finds praise for its event modifiers and attribute shorthands.

In summary — we're still waiting for Svelte to become popular enough so that we can draw some solid conclusions from thousands of developers' opinions.

Mobile development

Vue

There are a couple of ways a developer can choose from when developing a cross-platform mobile app with Vue. The most popular currently is NativeScript-Vue.

NativeScript-Vue is a NativeScript plugin that allows developers to use Vue.js syntax.

Reasons to go for NativeScript:

- Access to the entire native API — you can always access the native objects and APIs from JavaScript, and simply write overrides on plugins within your project if you want to customize anything.
- Very convenient code sharing.
- Works with Angular, core JS, TypeScript.

It is also possible to write cross-platform apps using Ionic Vue

React

React Native is the way to go when developing mobile apps with React. The framework has been praised for its “learn once, write anything” objective which means you get a native look and feel of your app for both iOS and Android, and you may also include platform-specific modules.

Unlike other cross-platform frameworks like Ionic, the Facebook team stresses that they acknowledge the differences between different platforms, and React Native reflects that. Therefore, you can still include modules specific for iOS and Android like status bars, navigation, or access to native elements like camera and contacts are built according to the platform.

Using libraries such as React Native for Web or ReactXP allows a developer to build a cross-platform app that runs on the Web too, so there’s no need to build separate apps at all.

There are some cons, though:

- Need for expertise from a native developer for some platform-specific modules.
- It’s not a fully cross-platform, single-codebase approach.

However, if you need a web-first approach, it is also possible to use Ionic React.

Angular

NativeScript is most often the way to go when developing mobile apps with Angular. On paper, this framework has everything that is needed in cross-platform projects, but in practice, developers claim that Native-Script + Angular is good when you:

- Develop a simple app with functionalities common for Android and iOS.
- Need custom native UI components, but you don't know Objective C or Java.
- Need to use the framework accompanying NativeScript (Angular in this case) to develop the web part on an app.

There are voices that point to a need for a better NativeScript documentation before the combination with Angular will be useful for ambitious projects. So if you're on a lookout for another option, you can always use Ionic Angular.

Svelte

Svelte Native is also based on NativeScript. It will have all the pros and cons of Svelte and NativeScript.

Best use cases

All of the mentioned technologies have their pros and cons, and their loyal advocates. But like with most technologies, it all comes down to what you need to achieve and how you want to do it.

To make the decision process a bit easier, here's a handy comparison table:

Choose Vue to:	Choose React to:	Choose Angular to:	Choose Svelte to:
Quickly get a working solution	Build a complex solution/SPA	Build a highly scalable app that follows an architecture design you picked from the get go	Experiment with a new and interesting framework
Build a lightning-fast app	Have the possibility of expanding your app heavily in the future	Build a big project with many separate teams working on it	Build ultra light-weight web and mobile apps
Migrate your existing project incrementally to a modern framework	Pick JavaScript over HTML	Check variable types at compile time	Target low power/low capacity devices
Learn as you go (Vue has a gentle learning curve)	Have the backing of the most-used mobile development framework	Pick a framework supported by Google	Create your own infrastructure
Have clean code and HTML templates	Make use of immense database of third-party libraries	Have component-based structure	

Monterail is a full-service software development company with 110+ experts on board delivering meaningful software for start-ups, SMBs and enterprises.

Why Monterail + Vue? Because we've been using it for a while now: we’re the first VueConf organizers, Vue evangelists, and we kinda love Vue!

www.monterail.com

hello@monterail.com

monterail