Vue for Business 2021

IS VUE THE RIGHT TECHNOLOGY FOR YOUR BUSINESS?

BROUGHT TO YOU BY MONTERAIL

OFFICIAL VUE PARTNER

What's Inside?

What is Vue.js and why are companies using it?

In 2014 a lone developer, Evan You, built and released Vue.js, an open-sourced frontend framework in order to improve on available JavaScript tools. A former Google employee, he wanted to create a framework that combined the best approaches to frontend development from frameworks like Angular, Ember, and React with other features that made writing Web apps faster, easier, and more pleasant.

Vue might have started as a small project driven by the developer needs of its creator, Evan You, but it has matured considerably over the years — becoming a full-fledged framework with a grownup ecosystem and developer tooling.

With over 177k stars (used to show interest in a framework) on GitHub, a widely used internet hosting service for software development, Vue currently has the most stars among JavaScript front-end frameworks, beating Angular (69k) and React (161K) by a fair margin. When it comes to overall popularity, React is still the king, and Angular, thanks to its specificity, holds a solid position.

Developing with Vue has several benefits; the most important of which we will cover later in this report. For now, to provide some context on what makes Vue so great here are a few features that stand out.

- **It's open-source** — you can adapt it freely to your project's needs.
- **It's immensely popular** — there are many off-the-shelf modules/libraries which can be used to solve most existing issues, making development faster and potentially cheaper.

- **Active community** — there's a plethora of well maintained and organized documentation. This makes it easier to learn as well as find solutions to most issues one could face to any issues you could encounter.
- **Progressivity** — you can integrate Vue into an existing project without starting from scratch.
- **Easy learning curve** — not only can an existing team pick up the framework, the ease of reaching competency in Vue means there's a large talent pool of developers available. Getting a prototype running is one thing. But when you need to scale it to use enterprise architecture, Vue is scalable and ensures you have a performant product.

> It's so easy to get a quick prototype of something up and running. Yet, the moment you need to scale it to use enterprise architecture, Vue is ready for you to scale up while still ensuring you have a performant app.
>
> – Ben Hong

We collected data in a survey about Vue, conducted over a five-week period in December of 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 these roles).

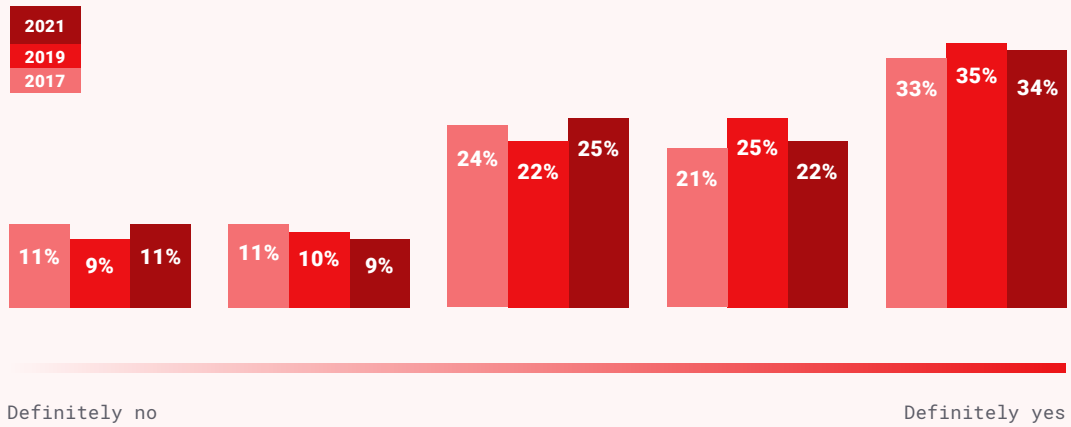
You can see in the data below, only part of our full survey, that:

- Vue continues to grow in popularity among digital product development.
- Companies that use the framework are satisfied and continue to do so.
- The primary learning resource continues to be Vue's renowned official documentation.

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.

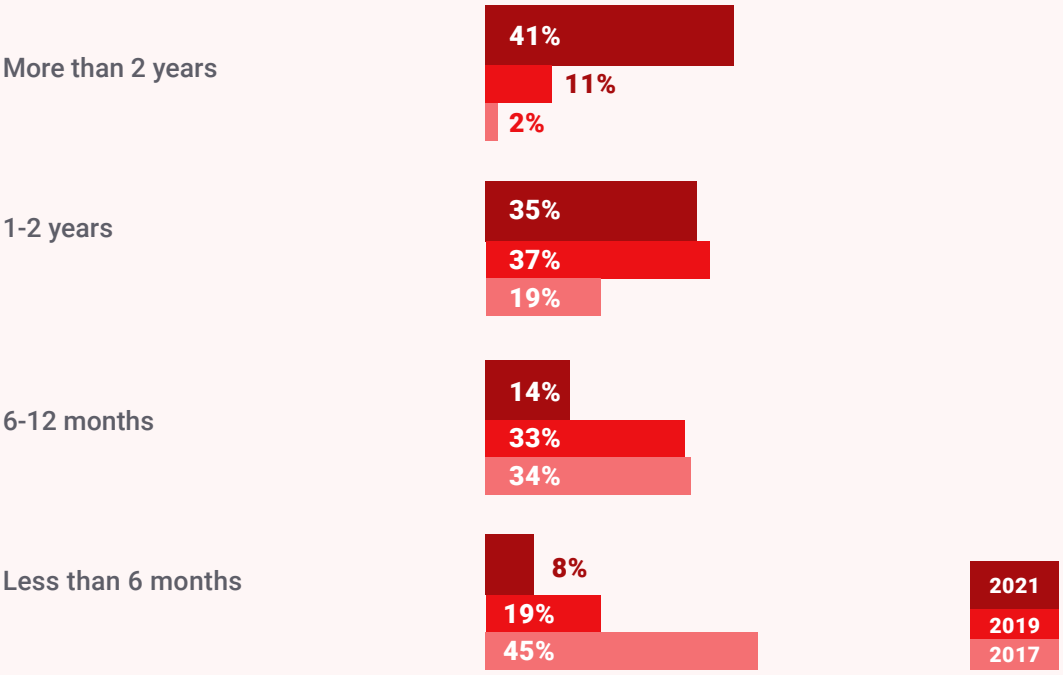
INCREASE IN THE NUMBER OF EMPLOYEES USING VUE.JS



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

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

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

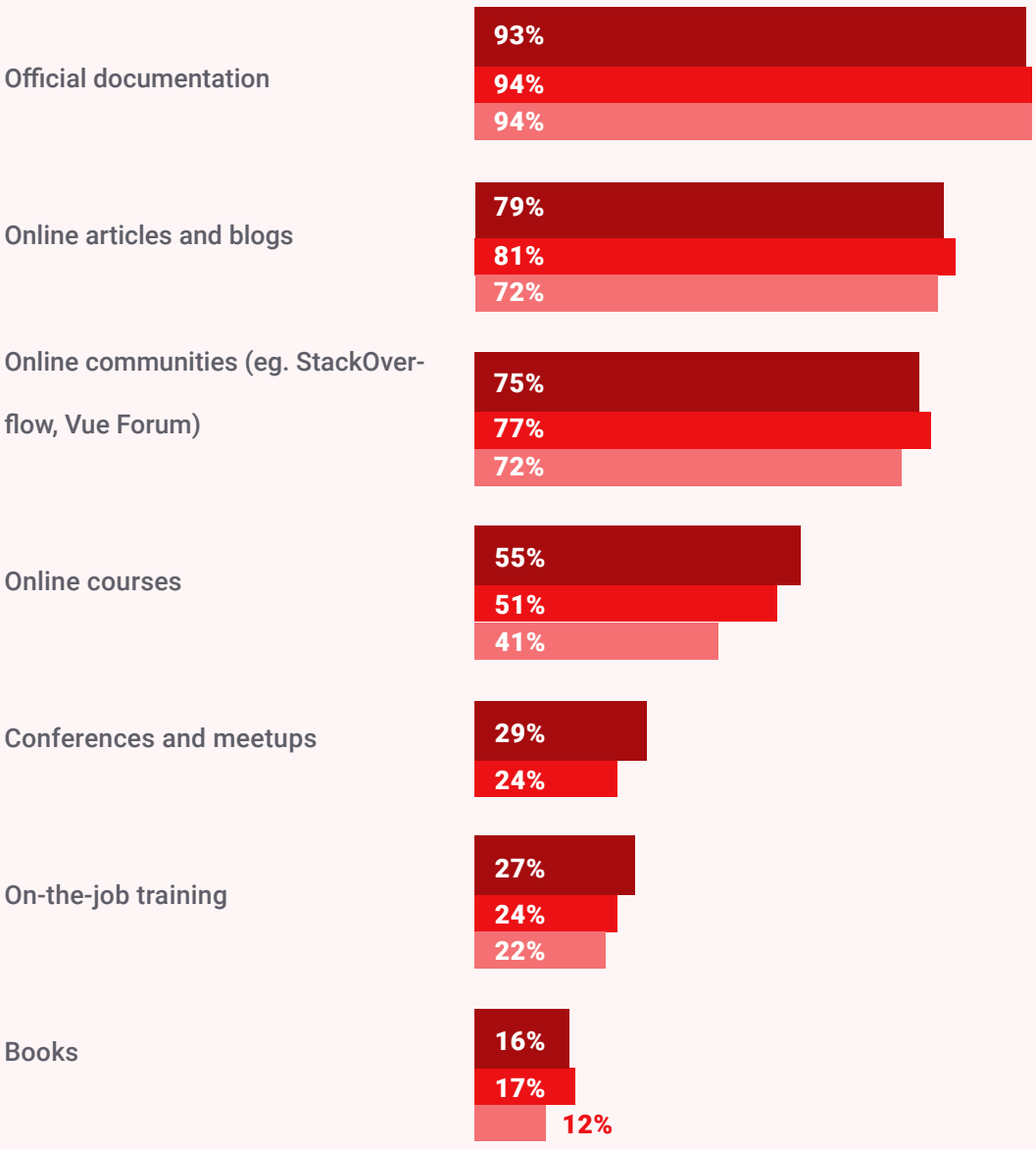


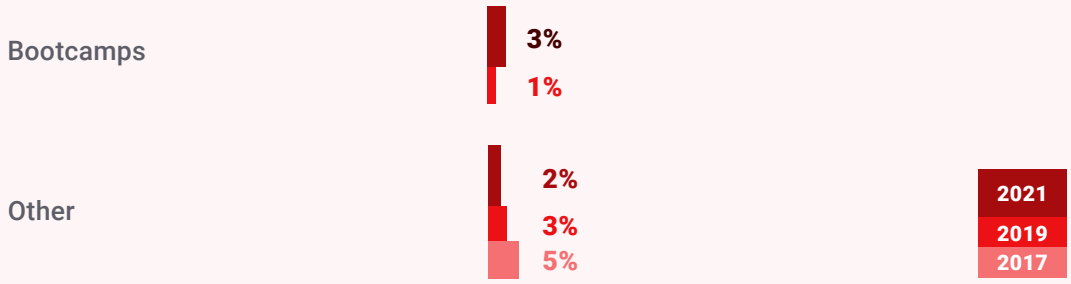
* Percentages do not sum up to 100% due to multiple choices.

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

Like twice prior, the official documentation continues to be the most popular resource for those improving their knowledge of the framework. This is, without a doubt, due to the well-known organization and thoroughness of Vue documentation.

LEARNING RESOURCES





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

What can you expect from using Vue?

Prototypes and MVPs can be built quickly to test new ideas. After this stage, when it's decided to move forward with development this framework scales well; all the way up to enterprise level applications. If you're not working on a new project, Vue can be used to add new features to ones you're already working on.

You benefit from Vue's extensive official libraries. They provide premade features or modules to speed up development. Let's not forget to mention the speed of products developed with Vue. Thanks to high performing server side rendering (see more below) you can expect quick, performant apps.

Google, Adobe, Upwork, and BMW are using Vue to great effect. To get a more detailed understanding of how the framework can be leveraged by a business, continue to the next section.

Want to know more about Vue in business?

Our experts will answer all your questions and help you decide what your next step should be.

Contact us

What can you build with Vue?

There's no definite answer to this question but a look at successful Vue implementations can give you the general idea of the framework's usefulness.

We've talked with three Vue experts and seven business owners who use Vue to distill how the framework works in business context and in which areas it can bring the most value.

We hope it will help you in answering some questions about which technology you should choose.

Let's dive in!

Vue experts on: “which projects Vue is best for?”

Vue is the kind of library that is incredibly flexible. It's so easy to get a quick prototype of something up and running. Yet, the moment you need to scale it to using enterprise architecture, Vue is ready for you to scale up while still ensuring you have a performant app.

Ben Hong

@bencodezen

Vuejs Core Team Member
Senior DX Engineer at Netlify
Vuemastery Iinstructor
Google Developer Expert

In theory, I think Vue.js suits any front end project. If you need a full-featured framework, Vue.js, with its rich ecosystem, is an apparent good fit. If you already have lots of customization in the front end layer, you can also bring in Vue.js for its reactivity utilities, or use it for simple component rendering. It’s just like jQuery.

Though, there is one area that Vue.js fits particularly well: if you are not sure how big the project will grow into and want to start with a simple but scalable architecture, Vue.js is here for you.

Vue.js calls itself the progressive framework and is designed to be incrementally adoptable. Users can start with a minimum setup and only choose to adopt complex building tools after finding the actual necessity.

Scalability is the thing that Vue.js does best. It supports all levels of users and grows together with them.

Haoqun Jiang

@haoqunjiang

Vuejs Core Team Member

When Monterail approached me with this question, my honest, very biased answer was to quickly blurt out “all of them”. Vue is a framework that has proven time and time again that it has the flexibility to be used as a full-fledged Single Page Application, Server-Side Rendering solution — without sacrificing flexibility to be plugged in as a widget or two in an existing product.

It is a framework that caters to beginners, allowing them to quickly get their hands dirty and start building powerful reactive solutions, but also a power-house in the hands of the most senior developers.

With the release of Vue 3, and in particular the composition API, this last bit becomes even better — unleashing the full flexibility of vanilla JavaScript on top of a framework that already excelled in many fronts.

So what is really my answer, in the end?

My truest, most honest answer is: Vue, as other frameworks, is as good as the developer who uses it.

If you are like me and my team, and using it every day, from Monday to Friday, month to month, component after component still makes you happy to fire up that first <template> and <script> — then you are betting on a framework that I have personally used for quickly scaffolding a proof of concept, all the way up to completely rewriting a huge backbone legacy code for the company I work for.

Marina Mosti

@MarinaMosti

Lead FE Enineer at Voicethread

Teacher at VueMastery

How modern businesses use Vue?

Case study #1

Passionate People

Luke Thomas

CEO at Passionate People

@lukevscostas

Passionate People is a software consultancy specialised in detached professional services, providing development teams additional specialised capacity to achieve their digital transformation goals. We have wide and profound roots in the Dutch front-end scene due to our extensive work with the front-end developer community (being co-organizers of both on-site and online events who have reached more than 6000 developers in the last 4 years).

In the last 3 years we have shifted from providing front-end consultancy to full-stack roles and into technical leadership and consultancy.

We operate with a very lean development team that can receive additional temporal help from other collaborators in the company, which means they need to onboard into our internal projects as quickly as possible in order to make the most of their time in our team.

We use a handful of internal tools and we wanted their front-end architecture to be congruent so that there's less cognitive load when switching from one to the other.

> **Our technical colleagues have a wide ranging experience with multiple front-end frameworks so we were also looking for an option that felt familiar to all, yet powerful enough to provide all necessary features.**

For quite some time a favourite of our team was React and a couple of experiments were started using an all-React frontend. While comprehensive enough, the desired goal to offer an easy-to-digest onboarding as well as a familiar set of features to developers not used to React was not achieved. The team had to first dive a bit deeper into the React way of working and additional concepts like Redux and Hooks before they could jump into understanding the codebase.

For a short time we also experimented with Web Components (based on Polymer 2 and then lit-html), but we found ourselves implementing way more than what we were comfortable with so we stopped the experiment.

We are a developer-run company so we pretty much had Vue.js in our minds since the creation of our company 3 years ago, it has always been one of the top skills we hear from when interviewing new colleagues and we for sure wanted to give it a try.

> **In less than a week any colleague giving Vue a try for the first time agrees that it is “quite magical” and that a lot of native features and projects in the ecosystem “just work” without having to deal with mismatched versions or excessive configuration like their experience with other frameworks and libraries.**

Our technical colleagues are really passionate about their choice of stack, so Vue.js allowed us to offer them a “neutral” alternative to the (back then) two major options available on the market back then: Angular and React.

We experienced that developers that were initially inclined for one or the other, really saw Vue as a worthy alternative that was familiar to them in many ways and offered a very friendly learning curve. A few moments later they found themselves productive in our codebase using a framework they haven’t used before and that was a priceless reaction.

For one of our most ambitious projects this year, we have a multi-theme fully dynamic administration UI for different user roles.

> **Thanks to features like inject/provide and dynamic components we were able to implement quite elegant and easy to understand solutions without writing a lot of code. The Vue ecosystem is on of the most cohesive ecosystems we’ve seen out there. Projects in the ecosystem are kept updated constantly and everyone in the community is really friendly and open to help.**

The documentation is top-notch and allows us to spend less time writing documentation for native features or for how to integrate external dependencies and focus more on our own implementation. Nuxt was a game changer for us too since every release in the last year has come up with additional features we put to use immediately.

We no longer need to have a large team dedicated to work on our internal

projects because the lean team we have is able to focus on implementing new features instead of implementing the underlying framework to work faster.

> **The team members who are able to join us for a short time are also able to quickly get up to speed and collaborate thanks to the shared architecture and possibilities offered by the combination of Vue and Nuxt.**

We started with no web platform and a super simple html + CSS website and now we have a really useful way to communicate with our audience and our clients with multiple deployments per day.

What's really exciting is improved performance of our fully dynamic apps and we are happy to see portals come now as a native feature. We are already using Vue.js 3.0 with Vite for a couple of small tools, for the rest we're waiting on Nuxt to provide official support (although we are already using the composition-API, so we're almost there!

Case study #2

FindlayWebTech

Thomas Findlay

Full-Stack Web & Mobile developer

@thomasfindlay94

FindlayWebTech is a consultancy-focused remote business that works with clients and developers across the world on developing web and mobile applications.

Back in the days, jQuery was a staple library for any project of a decent size due to the fact that it provided a lot of useful methods and ensured cross-browser compatibility. However, there were things it wasn't well suited for, especially in larger applications, where a lot of dynamic API driven content was involved. It was very easy to end up with a lot of spaghetti code due to imperativeness of the code.

> **Vue solved this problem very well, as it is component based and offers two-way data binding. The code enables you to use Single File Components which are great for encapsulating markup, business logic, and styling, without affecting other components.**

Vue offers a lot of features when used with Vue-CLI, but it also can be used in a progressive manner as a drop-in replacement for jQuery, which was very useful for smaller sites that needed only a bit of interactivity.

I did explore a few frameworks before choosing Vue. A few years ago, after finally deciding to switch from jQuery to something more declarative, 3 different frameworks were considered — Angular, React, and Vue.

Angular was dropped due to the fact that it had a lot of breaking changes and frequent major releases. No one wants to be outdated by a major version every 6 months or so.

It also offered less flexibility than React and Vue. Since then, both React and Vue are used in our projects interchangeably, however, **Vue is the more favoured framework, and usually is chosen for new projects, unless a client specifically requires a different tech stack.**

I've worked with Vue since the beginning of Vue 2. I'd read an article about it in a tech newsletter and it caught my attention. Vue has an easy learning curve and great documentation, so it wasn't hard to get the whole team using it very quickly.

Vue has solved a lot of problems and made a lot of things easier.

First of all, less code has to be written, as Vue abstracts DOM handling and automatically updates it when reactive state changes. Single File Components are great for encapsulating markup, styles, and business logic. Overall components made it much easier to create applications, as an application is basically a combination of many large and small components which can be easily dropped in and connected where needed.

The fact that Vue is very declarative makes the codebase easier to un-

derstand and maintain. It's also great that Vue supports JSX, as it's very easy for React developers to switch to Vue and work with it.

There are a lot of great features that make Vue reliable, but if I had to mention specific ones that I really find useful it would be SFCs, Reactivity, Vue-CLI, and Vue-Devtools. Composition API will also join this list now as Vue 3 will be used for new projects.

It's very easy to scaffold a new project with Vue-CLI, so that definitely saves a lot of time, as there is no need to set up Babel, Webpack, and other tools manually.

> **Instead of spending days configuring a project, running a simple script will scaffold a whole project with a lot of features working out of the box.**
>
> **Thanks to that, new JavaScript features can already be used in projects, as the code is transpiled to a format that is understood by older browsers.**
>
> **The fact that Vue is so easy to use saves a lot of time, as new developers do not need much time to get started with it. Components make it easier to abstract, encapsulate, and reuse markup and functionality.**

Other really great points about Vue include great documentation, a style guide, and official libraries for different aspects like routing and state management. These help a lot with increasing productivity. Developers

can just focus on writing the code, and don't have to make a lot of decisions about how to write Vue components, or what libraries should be used for routing, state management, etc.

> **The competitive advantage of using Vue is the fact that it allows you to develop fast and feature-rich applications in a much shorter time period.**

It has a great ecosystem with a large number of libraries, so there is no need to reinvent the wheel. Throughout the last few years, the Vue team has added quite a few new features without major breaking changes.

This is great because our and our clients' applications do not require major updates every few months, as it is the case with Angular applications for example. Therefore, we can focus on other things instead of wasting time on unnecessary updates, and shuffling between different versions and docs.

> **There are two metrics that definitely did improve — speed of developing new features, and overall performance and loading times of applications. Vue is extremely fast, and different techniques such as dynamic lazy loading and code-splitting help a lot with shipping only the code that a user needs at the time.**

A lot of my clients were very satisfied with applications that were developed with Vue, as they loaded very fast and provided great user experience.

I have been closely following Vue 3 updates and Request for Comments. The most interesting feature for me personally is Composition API which is inspired by React hooks.

Besides working with Vue, I also work a lot with React and know how big an impact hooks had when it comes to writing scalable and reusable stateful logic. After trying Composition API I can definitely say that it is one of the biggest and most important features that Vue 3 brings to the table.

Both myself and some of my clients have already started to use Vue 3, mainly on internal projects rather than user-facing ones though.

The ecosystem, as well as tooling around Vue 3 still needs some time to catch-up. Many libraries and UI frameworks are still working on providing Vue 3 compatible versions. Likewise, vue-devtools, which does already offer a Vue 3 compatible version, is not as feature-rich as the previous version, and still has some issues here and there. I think it still will be a few months before it is used for user-facing applications.

Case study #3

BridgeU

Maksim Fedotov
Web Developer
@DouTatsu

BridgeU is an edtech startup that empowers schools to provide more intelligent, data-driven university and careers guidance, through various software products.

Our application has been a monolithic Ruby on Rails application from the start, but as the product grew, we needed more and more complex functionality that required JavaScript as a result.

We briefly explored other frameworks, but Vue felt very similar to Ruby with more opinionated component structure and nicer structure as a whole. React, Vue and Angular have been the big 3 frameworks that were recommended to us, so we just tried all of them before settling on Vue.

Our most proficient front-end developer was the first one to start working with Vue. I joined shortly after, to take it up and notch and use Vue in greenfield projects, gradually introducing new developers to the teams working with Vue, until most had the basic knowledge of Vue. All in all, it took us about 2 years to start using only Vue for our JS-powered front-end.

> **Our JavaScript was used in a standard Ruby on Rails way, which made it very hard to modify and use effectively. Any JavaScript powered feature or page, regardless of complexity, would take a long time to build. But with Vue, it became a lot easier to write JavaScript pages, not to mention make use of countless libraries to make our lives easier.**

Single-file components have been one of the best features of Vue. Being able to self-contain parts of functionality and then re-use them is extremely useful and makes managing large codebases a lot easier. This also includes scoped styles, as there is no need to have large CSS files, instead you have the component context to see exactly what styles are being applied and why.

Single-file components make it easy to start working on a small part of a bigger feature, resulting in faster iteration and focus. It also makes it easier to work in teams, as everyone can be working on separate components in parallel, while before everyone would be making changes to a single, large JS file.

Finally, a lot of libraries became available for use, whatever built for Vue or just JS in general, resulting in us not having to write code from scratch.

> **Many of the companies in the educational space use quite outdated UI and technologies in general, making their pages feel very outdated, not very exciting to use as well as functionally clanky. That is not the case for us now, as Vue helped us build extremely**

> **complex and interactive pages, which are much easier to use by our clients as well as look much better in general, compared to the competition.**

We haven't particularly measure KPIs against introduction of Vue, but we definitely improved all of our KPIs when it comes to UX, due to Vue allowing for more complex and interactive pages.

We're planning to start using Vue 3.0 when the Migration Build feature is out.

The Composition API and Multiple root elements in Vue 3 are very interesting.

There are plenty of components we had to wrap in a div or split out differently, due to the original limitation. Suspense is also very useful, as we fetch a lot of data asynchronously from our Rails API, so this should allow for more elegant default component view.

Case study #4

Vue Amsterdam

Luke Thomas

Vue Amsterdam Organizer

@lukevscostas

Vuejs Amsterdam is committed to fostering and creating learning and connection opportunities for the Front-end Developer community.

We are responsible for running Vue.js Amsterdam the largest Vue & Javascript Conference in the world. Other Conferences include JS World Conference, Vue.js Road Trips in multiple European cities, as well as 4 other major conferences and multiple meetups over the years. Our work for the Front-end Developer Community has seen over 6,000 people gain knowledge in the last 4 years.

At any given time during the year, we are working on putting together an up-to-date website for our next event. We have a total of 7 brands each with its own style, website and particular set of requirements. We operate with a very lean technical team which from time to time receives help from external contributors.

We were looking for two major things:

- **A framework that allows us to move fast but with a clear and clean architecture**
- **A framework that's easy to pick-up for external collaborators not very familiar with it**

> **We explored other frameworks and while many offered a similar set of features, Vue.js immediately rose to the top as the best option because of its very friendly learning curve and its powerful capabilities.**

Features like dynamic components, scoped slots and functional components have made it a breeze to integrate and have allowed us to build a very robust and flexible multi-website platform.

We work with many developer communities and collaborate with many projects in the JavaScript ecosystem, however the Vue community quickly differentiated from the rest because of how friendly, open and cohesive it has been through the years.

When we first started Vuejs Amsterdam (back in 2018), we did it because we believed in the framework and wanted to see it succeed.

It's really remarkable how anyone with absolutely zero Vue.js experience can be quickly productive with it and understand our implementation after following some initial tutorials or reading the excellent documentation. On top of that, we use it in combination with Nuxt which greatly

reduces the cognitive load when onboarding a new teammate since all architectural decisions have not only been very carefully considered by the Vue.js and Nuxt core teams, but they are consistent across all projects where we use Vue.js + Nuxt.

When using other approaches in the past, we always faced three big pain points:

- Writing fully dynamic pages can be accomplished with pretty much any framework code, but we wanted to rely on something provided by our framework of choice without having to work out the implementation details ourselves
- Styling of multi-brand applications can be a tricky thing to implement (if the styling needs to be driven by the CMS for example), we researched different ways of achieving this and we were looking for a loosely coupled but powerful styling architecture. Solutions like styled components or CSS in JavaScript offered a good approach but they felt too tightly coupled for us.
- Libraries that don't offer an integrated development approach always pushed our development team to write some complex ad-hoc code that required maintenance and documentation that was usually out of sync with the code.

We rely on Vue because of its dynamic components. As of now, we have around 11 websites for previous and current events all with their own particular structure, branding and content all running un-

> **der the same platform: a single Vue.js project, statically generated at build time. Dynamic components help us give our content editors a great editing experience when combined with Headless CMS tools like Storyblok and Sanity. Due to the flexibility they provide, we are able to provide a blank canvas for them to craft wonderful pages with little developer input.**

The wonderful Vue.js documentation has saved us hours and hours of onboarding, it is so clear but yet so comprehensive that it allows us to point new colleagues to it and be confident they'll come back and contribute to the project in a very short time, even without any prior Vue.js experience.

Additionally, the extensive work by the open source community means there are many projects we can rely on to move faster, like Nuxt, Vuex and Vue-router.

Thanks to Vue.js, we are able to deliver completely new sites in record time thanks to how flexible and powerful Vue is. We are always exploring ways to deliver more integrated branded tools and Vue.js is allowing us to do that with ease.

Our developer events are our product, therefore we know how important having a good website is. **Pretty much with every Vue.js release our lighthouse scores go up, every time we let Vue.js do its own optimisations (by not getting in its way), we improve our loading speed and how responsive our site feels.**

As website platform, our most important metrics are how happy our end users are with the sites and how flexible the platform is four our content editors. So far, Vue.js has made this a big success.

We are really happy to see that many of the features we already use will become native to Vue 3.0: Portals, Suspense and Fragments. We have started using composition-api with Nuxt already for some time and we are currently waiting for Vue.js 3.0 support in Nuxt (Nuxt 3) to fully migrate to the new version.

Case study #5

Eve

An Phan

Head of Engineering at Eve

phanan

At eve, we are building a modern event management platform that captures, analyzes and uses data to mitigate risk and optimize live event experiences. Combined with our proactive insurance solutions, the platform allows event organizers to significantly reduce their risk of event cancellations through automation and insights gained by machine learning driven models.

As a fast-growing, multi-cultured startup, we lean toward technologies that have a lower barrier to entry, allow for quick development while providing enough flexibility to extend and can be switfly adopted by a team (or teams) of engineers with very different backgrounds. Vue is a perfect combination of each of those things (and more). Thanks to Vue and its rich, mature ecosystem, we were able to launch our core product in record time without sacrificing quality.

We opted for Vue from the get-go for all of the reasons stated above.

Personally, I tried Angular before and had an occasional hands-on React experience. Both are great frameworks but Angular felt too verbose and bloated when the complexity and extensive use of JSX with React discouraged me as a beginner — though I'm willing to consider it to be a matter of personal taste.

I found out about Vue when some time ago, I wanted to build a web-based media player. I started with React but wasn't happy with the progress. I then explored other options and decided to try Vue, which was still at version 1.x and relatively unknown.

With the Vue+Laravel combo, I managed to release my project over the course of one weekend. It went viral and remains one of the more well-known of its kind on GitHub to date. I became one of the first core members and have been a huge proponent of Vue ever since.

When we started, none of our team members (except for me) was a Vue veteran, but all it took for them to get the first Vue commit across was one day, maybe less, of reading the documentation and playing around with the code.

The more quickly teams grow, the harder it is to maintain codebase quality. Vue helps minimize this problem in many ways: having a clean and well-defined component structure, maintaining excellent official documentation, providing the best style guide, patterns and practice, etc. Having first-class support for and from other third-party technologies — testing with Vue Test Utils and Cypress, static site generation with Nuxt,

mobile app development with Vue NativeScript, to name a few — also eases the pain of maintaining and extending the product scope.

> **From a technical point of view, we believe the most powerful feature of Vue is component composition. Cross-component communication in Vue is super powerful yet straightforward, and slots are simply a godsend.**

It's much easier to write clean and declarative code in Vue. As a team, that means better collaboration — it takes less time for us to explain and understand the intention, which in turn greatly reduces review effort. Combining it with the power of TypeScript and we have a codebase of high quality yet super beginner-friendly.

> **Vue allowing quicker turnarounds means we can ship features to market faster. At the same time, its flat learning curve gives us way more options when it comes to attracting talent. In fact, we're not even looking for Vue but JavaScript developers; we know for a fact that any good JavaScript developer can master Vue in no time.**

With the release of Vue 3.0, we're pretty excited about the Composition API, which is a game-changer and promises an even better approach to authoring and sharing components. Also, the fact that Vue 3 was rewritten from scratch in TypeScript ensures the best type support one can expect from a framework.

We are already excited and ready to start using Vue 3! Right now we're only waiting for the ecosystem (plugins and libraries) to catch up with next version to kick off the transition.

Case study #6

Coursedog

Nicholas Diao

CTO at Coursedog

Nicholas Diao

Coursedog is a SaaS platform for colleges and universities that powers their class scheduling and curriculum. We initially chose Vue because of its ease of use for small, early demos when the company was still young, but also for its flexibility and power as our codebase and product grew.

We briefly explored React, but found it was far more steps to set up than Vue, and our team was looking for simplicity. We found out about Vue because one of our earliest engineers worked with it at a prior company.

It took us less than a day to start working with Vue, particularly due to the ability to drop Vue into an HTML file directly. **Vue allowed us to iterate quickly, create simple and easy to use early features, while giving us the architecture to create a large and powerful product over time.**

We rely very heavily on Vue watchers, computed properties, the Vue router, and Vuex.

Vue has a very generous learning curve, allowing junior developers (especially those with little Vue experience) to get ramped up quickly. This has allowed us to greatly cut down on our on boarding time for new hires.

We work in education technology, where the pace of progress is often very slow and arduous. Vue allowed us to iterate very quickly, while also allowing us to build highly stable, reliable, and powerful features that far exceed what competitors are able to offer in their frontend (especially the competitors who still rely on only PHP).

> **The main product metric that has been consistently high for us is feature delivery — and this is in large part due to Vue.js, and the flexibility and ease of use that it offers.**

We are actively keeping track of all the changes and try to align our codebase with the direction the framework is taking to ensure the migration will be smooth. We are big fans of the composition API, which we already use for almost a year thanks to the composition-api plugin available for Vue 2.x. It has helped us further scale our frontend by allowing our team to easily create reusable pieces of logic that can be mixed together based on the use case.

We are still waiting for the ecosystem of 3rd-party libraries to catch up with the new version of Vue, but once this happens, we will surely start our migration to 3.0. The performance gains, as well as the improved developer experience (like the new vue-devtools), will be beneficial for both our users and developers alike.

Case study #7

Extradom

monterail

Extradom is one of the biggest players in the Polish architecture industry. In 2017, they approached Monterail wanting to build a 3D interior design Web app.

The project requirements included app stability, impressive UX/UI, and integration with external assets via a REST API.

The application had to be accessible online on every modern operating system and browser without installation.

An app of this size and complexity demanded a powerful framework. Vue.js was our technology of choice for frontend development because it enabled easy data management to deal with with big and complex tools like iDesigner, which provides real-life 3D visualizations.

Vue allowed us to build an app with a unique architecture and was highly flexible during the process. Considering the app's visual nature, we knew

that top-tier UX/UI tied with the design would be key to its success. We decided to go with BabylonJS, which works well with Vue and is one of the best 3D engines out there.

By integrating iDesigner with external assets, Extradom was able to engage in close cooperation with e-commerce platforms which opened doors to many future cross-industry mergers. The MVP was released in just 8 months and the success of the project hinged on a well-thought out approach and technology solutions which resulted in a stable cross-platform software.

Would you like to see more apps made with Vue?

We can show you what we've created, and explain why it works.

We'll give you a hands-on experience of how Vue can be used to grow your business.

Contact us

Is Vue for you or should you consider a different technology?

The landscape of JavaScript frameworks and libraries is vast and rich in possibilities. There are the known and liked React, Vue, Angular, or jQuery, and there are new ones that spark interest (Preact, Svelte, Knockout, Polymer anyone?).

To choose the right technology you need a refined list of concrete goals and priorities for your project. We can help you do that, but not in this report. Here, we can tell you how the technologies compare in different aspects and in what type of projects they will perform the best.

Vue is chosen by those who seek:

- Best in the market Server-Side Rendering support. SSR translates to faster performance.
- Progressivity. Vue can be incrementally integrated into an existing backend application without the need for major refactors.
- Support coupled with flexibility. Vue has official libraries for things like routing, state management, and setup, along with guidelines on how to build apps, but developers can easily opt-out and choose the unopinionated way.
- A framework that's easy to get new people for that has a large talent pool.

It doesn't mean that the other frameworks can't do that, as we don't want to convince anyone to pick Vue against their best interest. We simply want to give you a comprehensive overview of the most common uses

and characteristics that define each framework.

With that being said, let's focus on the three most popular technologies: React, Vue, and Angular, and one that's gaining traction: Svelte.

Technology overview

Vue

The first version of Vue was built by one developer, Evan You in 2014 to improve on available JavaScript 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 update only those fragments of the website that need to be updated, 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 layer** — separate libraries for routing, state management, etc.

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

- **Gentle learning curve** — it's a great fit for junior developers; regulars and seniors can learn even faster.
- **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** — again, good news for developers, reducing the development time.

Being a progressive framework, Vue can be integrated into an existing project incrementally, per project requirements. For example, as a lightweight library to add some interactivity to a Web application.

With an easy learning curve and tools such as Vue CLI and its graphical interface, Vue is excellent for a quick delivery of MVPs and startup ideas. Because of those, Vue is also a cost-effective solution for small to medium apps.

Don't let that fool you into believing Vue isn't a good fit for large Web

apps. Quite the opposite. Vue has a vast ecosystem of tools and companion libraries, allowing the framework to respond to the complex needs of enterprise-grade applications.

And although Vue does not rely on corporate backing, it has a healthy stream of funding from sources like Patreon, GitHub sponsors, revenue share from educational content partners, and ads. The core team currently counts 20 people, and the technology is famous for having an exceptionally engaged community. Together with ever-increasing popularity, these things prove that Vue is future-proof and stable.

Companies that use Vue

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.

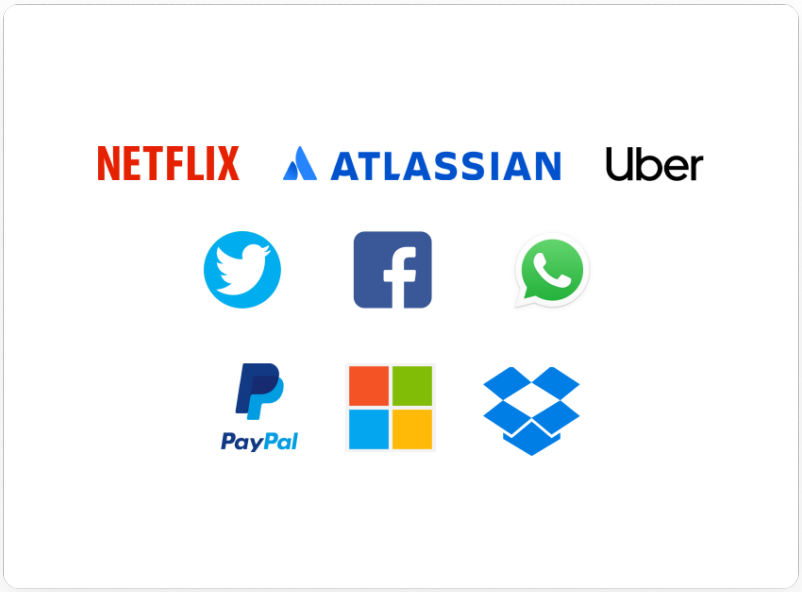
React's corporate backing, especially from such a major player, indicates stability in the future and implies that React will continue to be developed with long-term support.

Since React was created for large-scale Web projects, choosing it for small and simple apps might be overkill. Although it requires a lot of boilerplate code just to set up a working project, React's architecture ultimately pays off in the long run.

JSX leaves developers the full power of JavaScript at their disposal — flow controls and advanced IDE features such as autocompletion or listing are included.

Unlike Vue, React doesn't have official packages for routing or state management. To build complex apps, you have to use third-party solutions for pretty much everything. But the choice is vast. Experienced developers tasked with delivering advanced projects will know which of the numerous libraries will be the best choice to meet the business demands of a particular Web application.

Companies that use React



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 the Angular characteristics:

- Angular has many predefined modules, classes, and services that help you build apps faster, but make the entry threshold 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.

Companies that use Angular

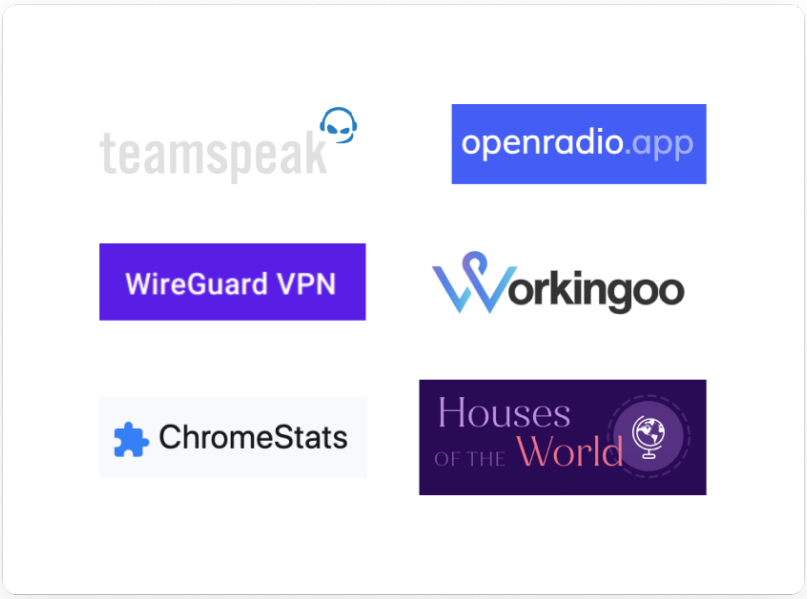
Svelte

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** — Svelte is achieving that through: simplified component implementation, no limit on the number of top-level elements, updating the state of local components with assignment operators, reactive declarations, and more.
- **Svelte is a compiler** — because of this, Svelte outputs optimized application code where and when possible, so the framework overhead is removed in favor of imperative operations; hence there's **no virtual DOM** and Svelte improves its performance.

Companies that use Svelte



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 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). And updating these plugins is also super-easy.

As for the companion libraries for state management and routing, they are managed by the Vue.js organization, 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 Create React App comes out a bit poorer when compared to Vue CLI. **You cannot customize your project after 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 sometimes can 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 be-

fore 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**.

Running at build time Svelte converts components into imperative code that updates the DOM. That's why the size of a **Svelte application is generally very small** because it does not need to contain the runtime part of the framework. Two sources state, however, 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 and avoid building big single pages.

Summary

Angular can scale to build exceptionally large applications, but the architecture has to be thought-through and planned before the development starts.

React can build both single page and multi-page applications, and handles scaling really well (look at Facebook). The only caveat is that it needs third-party libraries that are not officially supported.

Vue is great with both multi-page applications and single-page ones. It's extremely easy to implement into existing apps. Extending the Vue project is also convenient for developers because of user-friendly CLI and the large number of plugins.

Svelte is lightning-fast, but in terms of being super lightweight, there are some things that developers need to keep in mind — otherwise Svelte's unique benefits may diminish.

Performance

Raw performance data can be easily found on Stefan Krause's website (we're using Chrome 86.0.4240.75 comparison here). It's a comprehensive and detailed comparison of various JavaScript technologies. To see how Vue, React, Angular, and Svelte match in the performance area, we also took a look at a more readable but less comprehensive comparison written by Jacek Schae for Daily JS.

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

Name Duration for...	svelte-v3.23.0	vue-next-v3.0.0	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

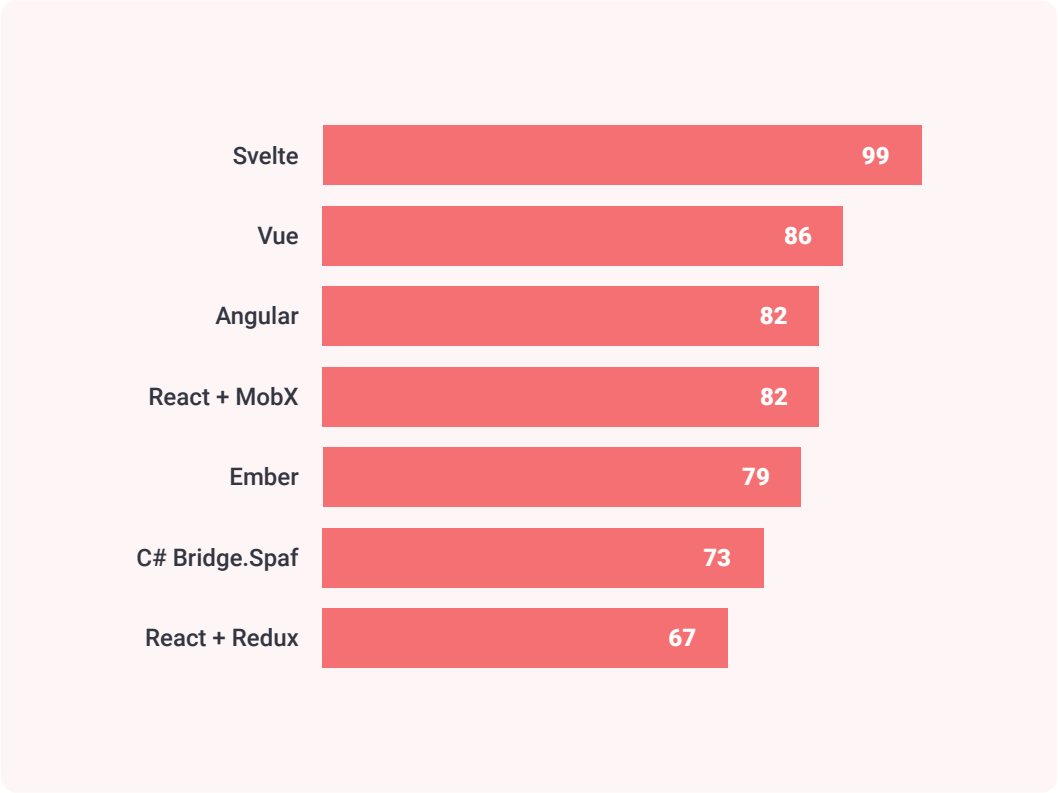
Memory allocation in MBs ± 95% confidence interval

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

Jacek Schae for Daily JS

Jacek Schae used Google's Lighthouse Audit to test how fast a RealWorld App implementation in a given technology loads. A score of 100 means that the tested website was lightning-fast. We've reduced the results to the technologies that are of interest to us.



Source: Medium

Summary

In Stefan Krause's analysis, from fastest to slowest, technologies rank: Svelte, Vue, React, Angular. Svelte is ahead of Vue in many aspects, but the differences tend to be slight. React beats Angular in 12 out of 20 metrics.

Adding Jacek Schae's comparison to that, we can rank the technologies performance-wise:

1. **Svelte**
2. **Vue**
3. **React**
4. **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).

Popularity and support

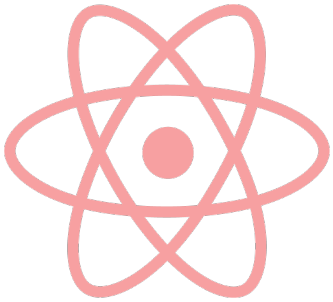
To gauge popularity and support, we looked at GitHub, where the technologies are developed, and at Stack Overflow, where the community is solving problems and where the newcomers can learn.

The metrics we chose tell us a couple of things:

- **GitHub stars** — they can be used as a general measure of popularity or likeability. Developers "star" specific repositories to stay on top of any changes.
- **GitHub commits** — a good way to know how actively the project is developed. A "commit" happens when developers upload a changed code.
- **GitHub contributors** — a number of people who know how to use and improve the technology. This metric is one of the measures that tell us how "strong" a framework is, as having more contributors means fixing issues faster.
- **Stack Overflow questions** — this represents the number of people who are learning or using the technology and looking to solve some problems.
- **Live websites** — how many real-life implementations of a given technology are currently online.

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

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



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

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

Summary

The two most actively developed technologies are Angular and React (followed by Svelte and Vue). The same goes for contributors.

As for real-life implementations, React is the most popular, then Vue, followed by Angular and Svelte.

Stack Overflow is used the most by React developers, then Angular, Vue, and Svelte. The low amount of questions asked regarding Vue can be explained by its famously comprehensive and user-friendly documentation.

Vue has the most GitHub stars, followed by React, Angular and Svelte.

We can conclude that React and Angular are the most developed frameworks with an established expert base, followed by Svelte and Vue, whereas Vue developers are the most pleased with using this technology.

When to pick each technology

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	Pick a framework supported by Google	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	Have component-based architecture that needs thorough planning but has many benefits	Create your own infrastructure
Have a code that's easy to maintain	Make use of immense database of third-party libraries		

Still can’t decide which technology to choose?

No worries! We’ll help you pick the best option for your goals and then, if you need us to, we’ll deliver a working solution that will delight the users.

Contact us

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