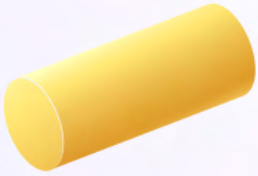
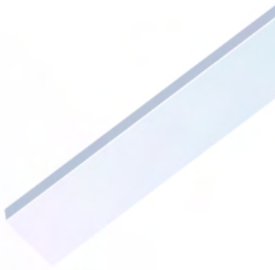
What is the future of
frontend development?

State of Frontend 2020





THE SOFTWARE HOUSE

Experts

Yan Cui
AWS Serverless Hero

Marek Gajda
CTO of The Software House

Jessica Jordan
Developer Advocate at .cult

Luca Mezzalira
VP of Architecture at DAZN

Rocky Neurock
Engineering Team Lead
at Honeypot.io

Tim Neutkens
Head of Next.js at Vercel

Guillermo Rauch
CEO of Vercel

Tomek Rudzki
Head of R&D at Onely

Dylan Schiemann
CEO of Living Spec

Bartosz Skowroński
Head of Design
at The Software House

Authors

Patryk Mamczur
Editor in Chief

Marcin Gajda
Tomasz Kajtoch
Wiktor Toporek
Andrzej Wysoczański
Technical Consulting

Joanna Swoboda
Magdalena Habarta
Kamil Głowiński
Publication Design

Table of contents

01 **Developers** 8
Everyday frontend development according to 4,500 experts

02 **Frameworks** 12
React is king. But who's the contender?

03 **Hosting** 18
Traditional DCs, cloud giants and frontend-focused hosting

04 **Jamstack** 24
Ecstatic about static

05 **Micro frontends** 28
Do we need microservice revolution in frontend development?

06 **Search engine optimization** 32
It seems that you don't care about SEO. Here's why you should

07 **Application accessibility** 36
Making the interface friendly for every user

08 **Development teams**
Frontend development? It's a team sport **40**

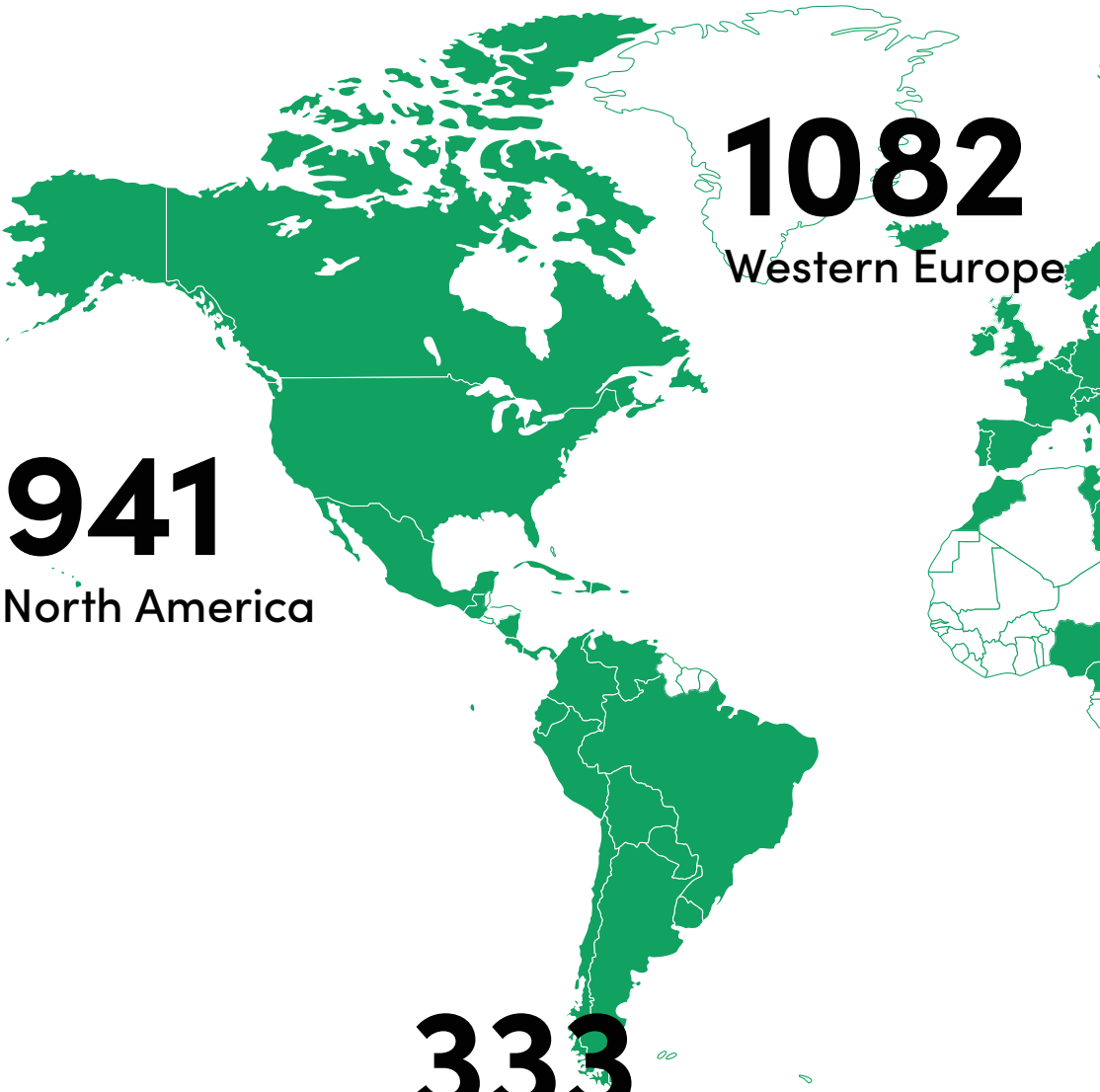
09 **Design**
Striving for close collaboration between designers and developers **44**

10 **Quality assurance**
Software testing as the cornerstone of software development **48**

11 **Future of frontend**
State of Frontend 2021? **52**

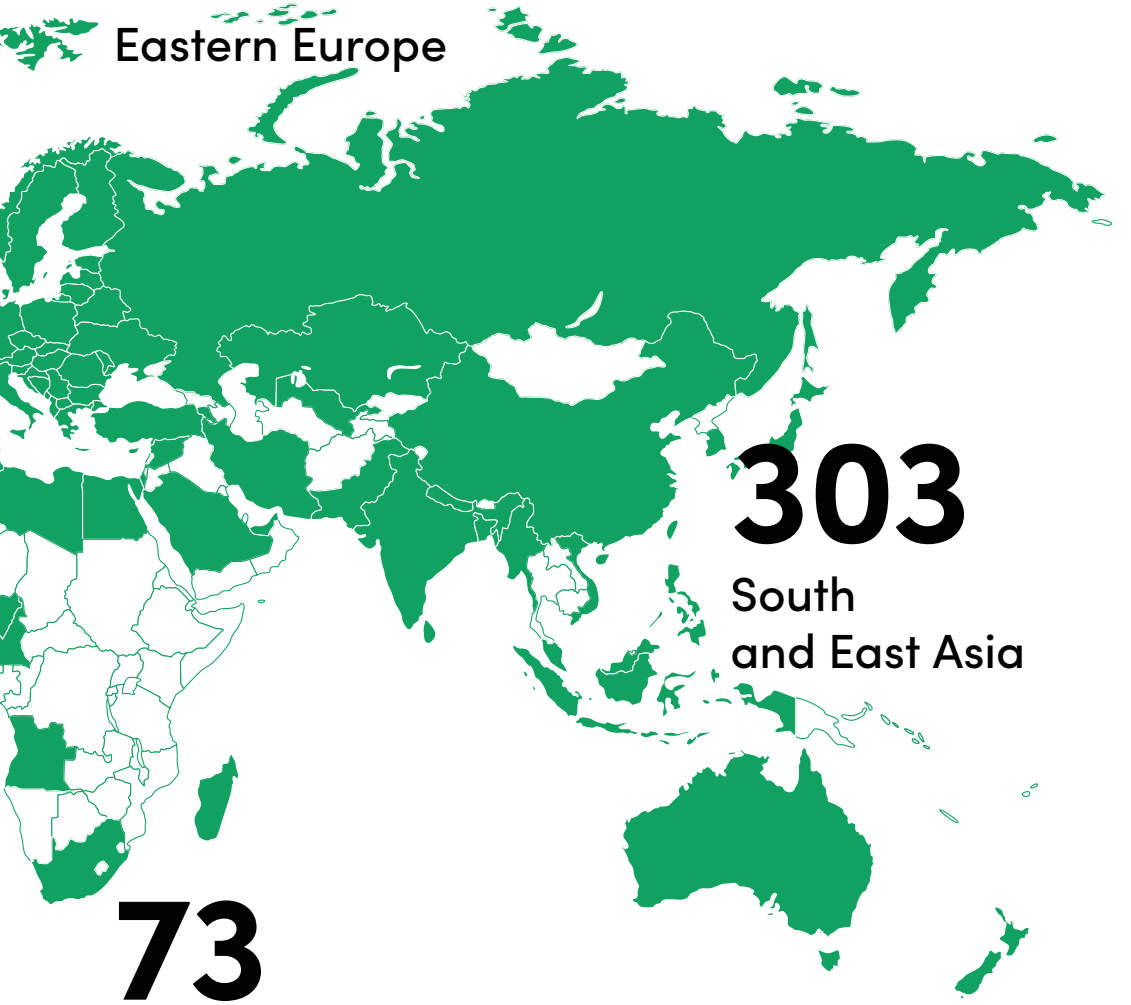
How many frontend developers took part in the survey?

Total answers: 4500



107
Other

1581
Eastern Europe



303
South
and East Asia

73
Middle East

01.

Developers

Everyday frontend development according to 4,500 experts

Patryk Mamczur
Report's Editor in Chief

> **With 4,500 folks filling in the survey, the State of Frontend 2020 is the biggest report out there focused solely on frontend development.**

When we started thinking about the State of Frontend 2020 report, I had one goal in mind: to find out what the everyday job of a frontend developer looks like. Considering that there are thousands of frontend devs around the world, that goal seemed pretty unrealistic. But hey man – once again, the frontend community surprised us all!

In just a few weeks, exactly 4,500 frontend developers took part in our State of Frontend survey! I know the number first hand because I was literally sitting with my finger on a button, observing the growing numbers and waiting for the perfect moment to close the survey. I don't know about you but for me the number is pretty perfect – as with 4,500 folks filling in the survey, the State of Frontend 2020 is the biggest report out there focused solely on frontend development.

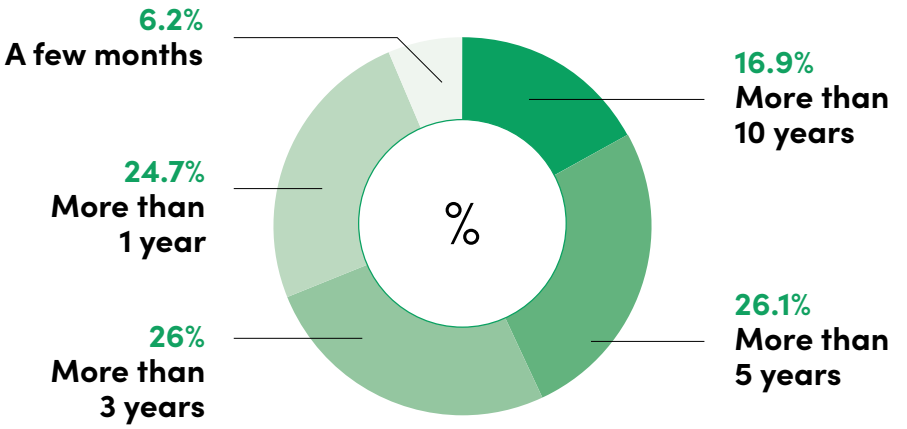
All in all, we ended up with over four thousand frontend devs telling us about their everyday job, the frameworks that they use, the frameworks that they would like to use (but, for example, the boss won't let them) and about their

thoughts on the recent frontend development trends. The results of the survey are awesome – some of them surprising, many of them inspiring, all of them showing how the everyday frontend development looks.

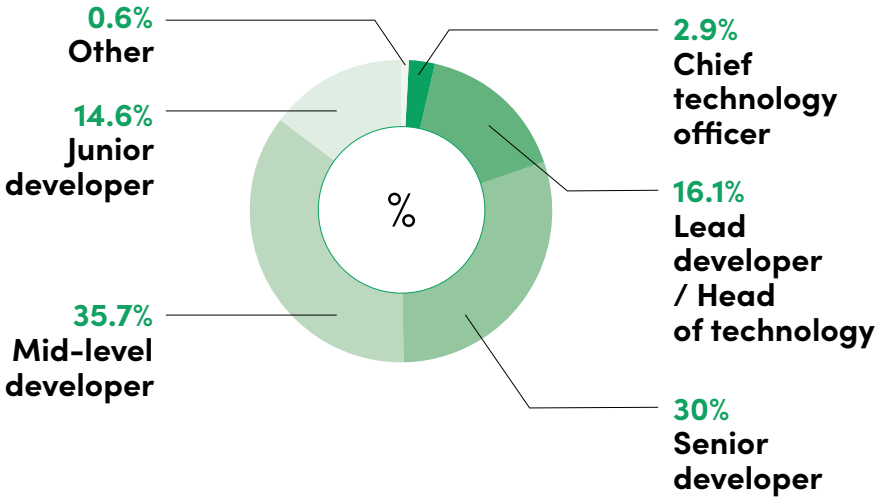
To make all this data more accessible, we invited leading software development authorities to comment on the results. With their incredible know-how and big-picture perspective, they made the State of Frontend 2020 report what it is – the most up-to-date source of knowledge on the modern frontend development.

So, I advise you to stop reading my brag-about introduction, take a look at the table of contents, choose the topics that interest you the most and find out what both the developers and the authorities have to say about it. You won't be disappointed.

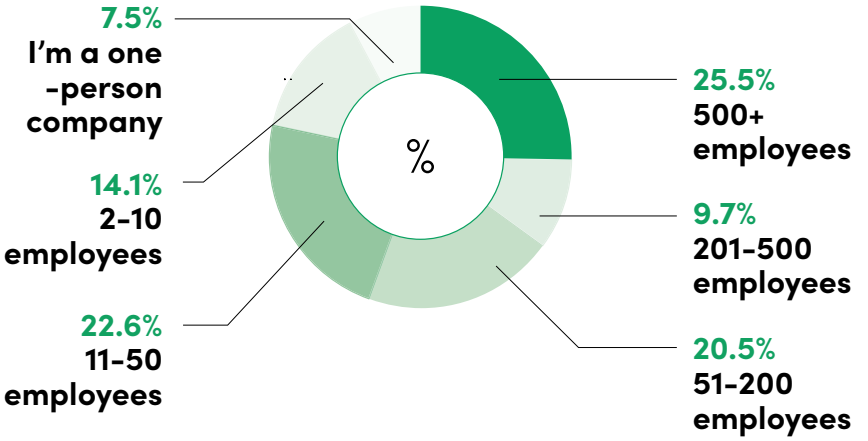
For how long have you been in the frontend development game?



How would you describe your seniority?



How big is the company you are working in?



02.

Frameworks

React is king.
But who's the contender?

Dylan Schiemann
CEO of Living Spec, Co-creator of Dojo

One thing is certain: React dominates the mind share today.

When you look at the results of the State of Frontend survey, one thing is certain: React dominates JavaScript framework mind share today. However, at the same time, it seems that next-generation reactive frameworks may soon rise as lean alternatives to the React ecosystem. And it all has much to do with the rising popularity of TypeScript.

For the past several years JavaScript developers have gravitated towards React, Vue.js and Angular as the leading frameworks. Relative interest in Angular has decreased due in part to the long delay in shipping Ivy and, similarly, interest in Vue has stalled a bit due to the long-awaited and somewhat delayed Vue 3.0 release. It all helped React dominate the JavaScript framework market with 74.2% of the survey's respondents using it – more than Angular and Vue.js users combined!

It doesn't mean, however, that the React community lives without a care in the world. A major change took place recently when developers started turning away from Redux. We can already see that, when it comes to state management, more people use React Context API and hooks (49.6%) than Redux (48.2%) – of course, some of them still use both but the trend is visible. Also, as a side note, while discussing big JavaScript frameworks is important, we must not forget about jQuery which, while rarely talked about, still remains the most widely deployed JavaScript library on the web.

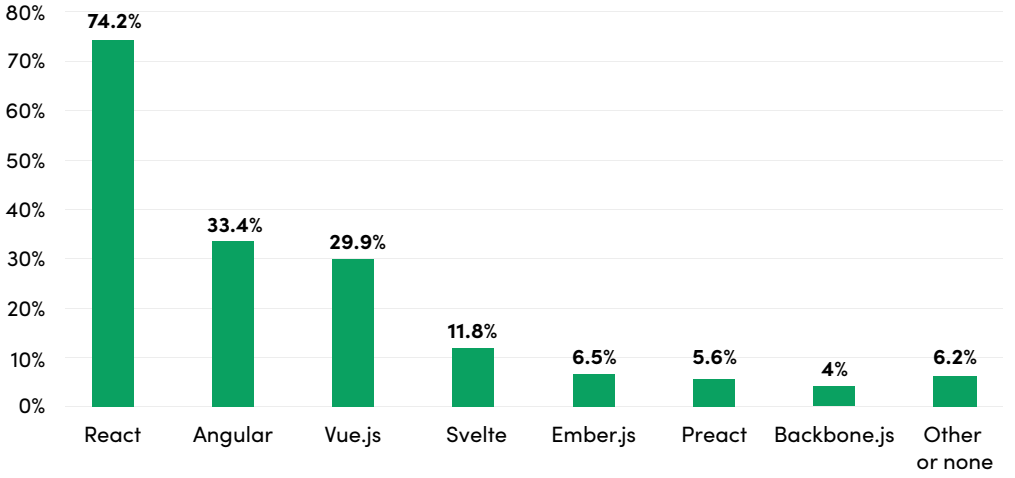
And what about the future of JavaScript? We're already seeing significant interest in next-generation reactive frameworks such as Svelte which strives to provide reactivity on top of normal DOM structures. Yet another competitor is Stencil – a framework focused on web components and, just like Svelte, on efficient compilation. Also, Dojo has re-emerged as a reactive TypeScript-first framework promising intelligent defaults for faster out-of-the-box experience.

Some argue that these next-generation frameworks may be great for smaller applications but require more work when building large apps. It's true that all of them provide much smaller default application bundle sizes as they do not carry the same legacy as frameworks which need to support features of the past few years. Also, they are very aligned with modern standards and language features.

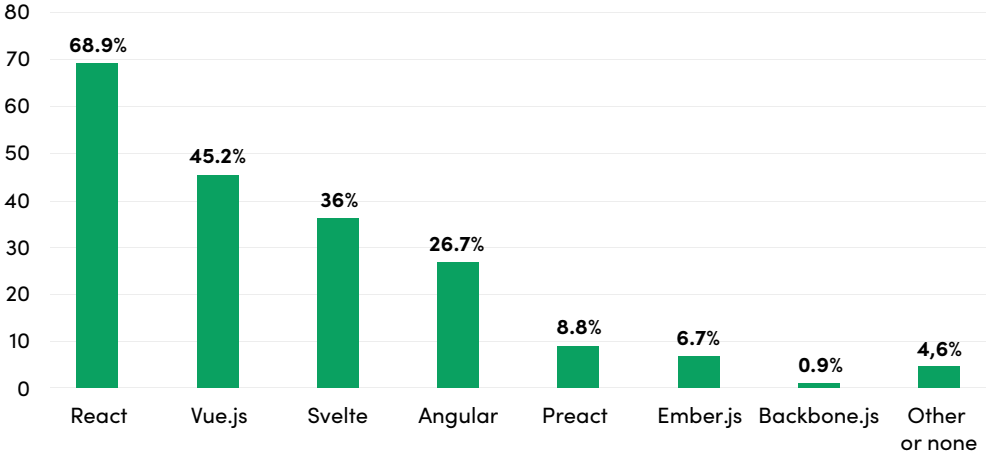
And this is where we must discuss the growing importance of TypeScript. With 77.2% of respondents already using TypeScript and most of them preferring it to JavaScript, it's not surprising that frameworks are improving their support for TypeScript and many start to leverage TypeScript internally. It's true for both the already established frameworks (like React and Angular) and the next-generation ones (Stencil and Dojo in particular).

With all these changes going on, I'm really looking forward to seeing what happens next in the realm of JavaScript frameworks. Because one thing is for sure: React is the king now but there are already a few contenders for the throne waiting.

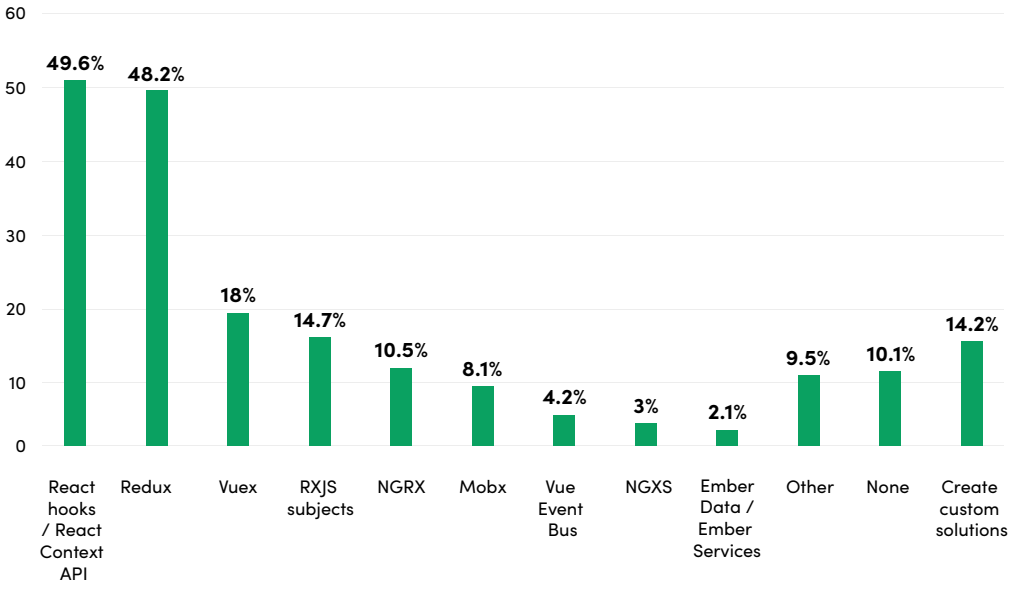
Which of these frameworks have you used during the last year?



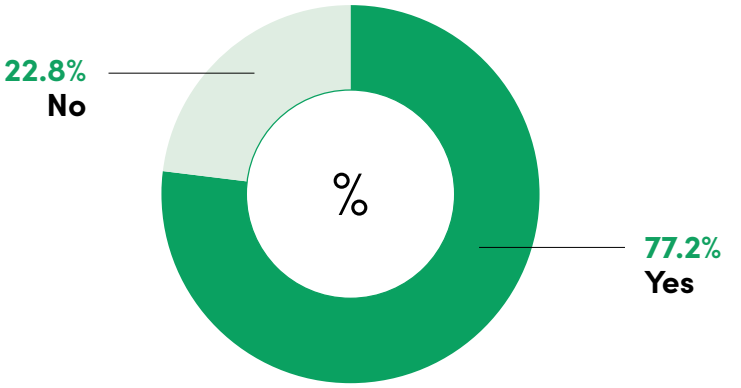
Which of these frameworks would you like to keep on using or want to learn in the future?



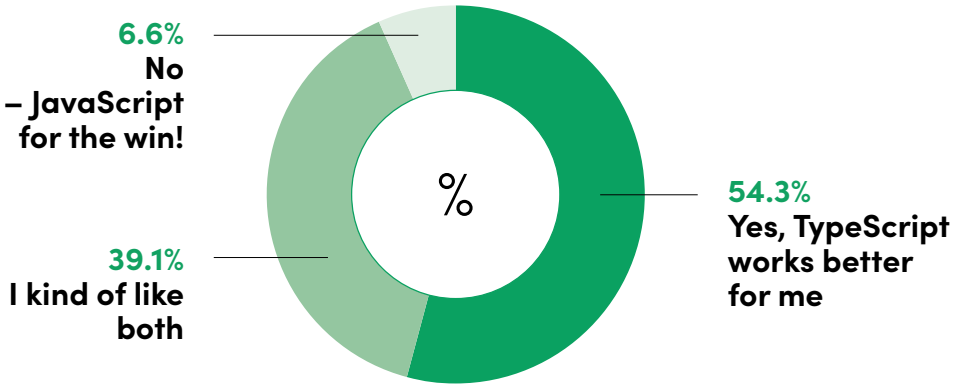
Which solutions do you use when it comes to state management?



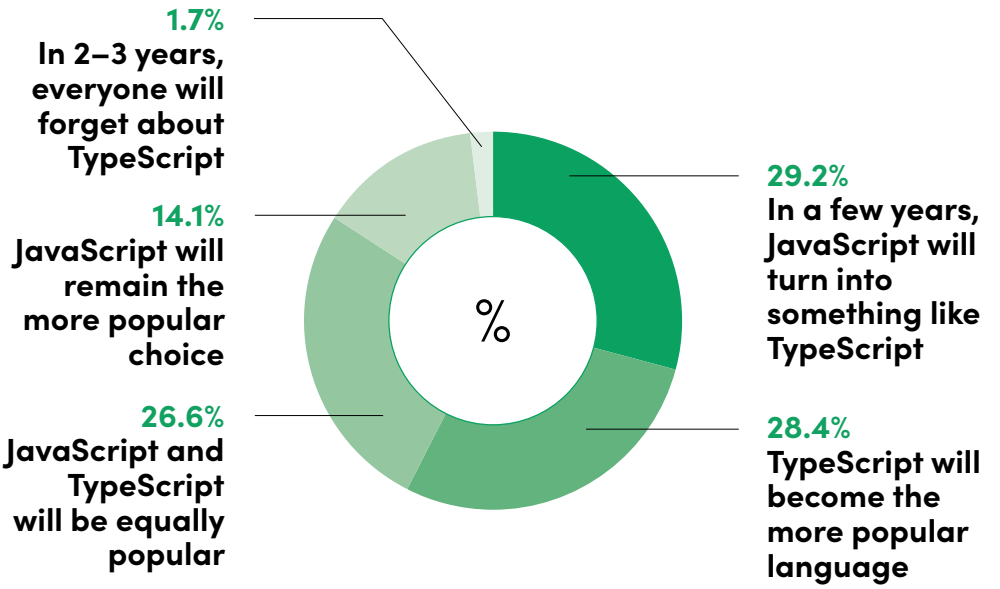
Have you used TypeScript during the last year?



Do you like TypeScript better than JavaScript?



What do you think about the future of TypeScript?



03.

Hosting

Traditional DCs, cloud giants and frontend-focused hosting

Yan Cui
AWS Serverless Hero
and Independent Consultant

> **The future of frontend development calls for platforms such as Netlify and Vercel which provide easy-to-use features for frontend-focused teams.**

The results of the State of Frontend 2020 survey appear especially interesting when it comes to the subject of hosting. When you look at the numbers, there's everything: from traditional DCs through cloud providers to the new kids on the block like Netlify and Vercel.

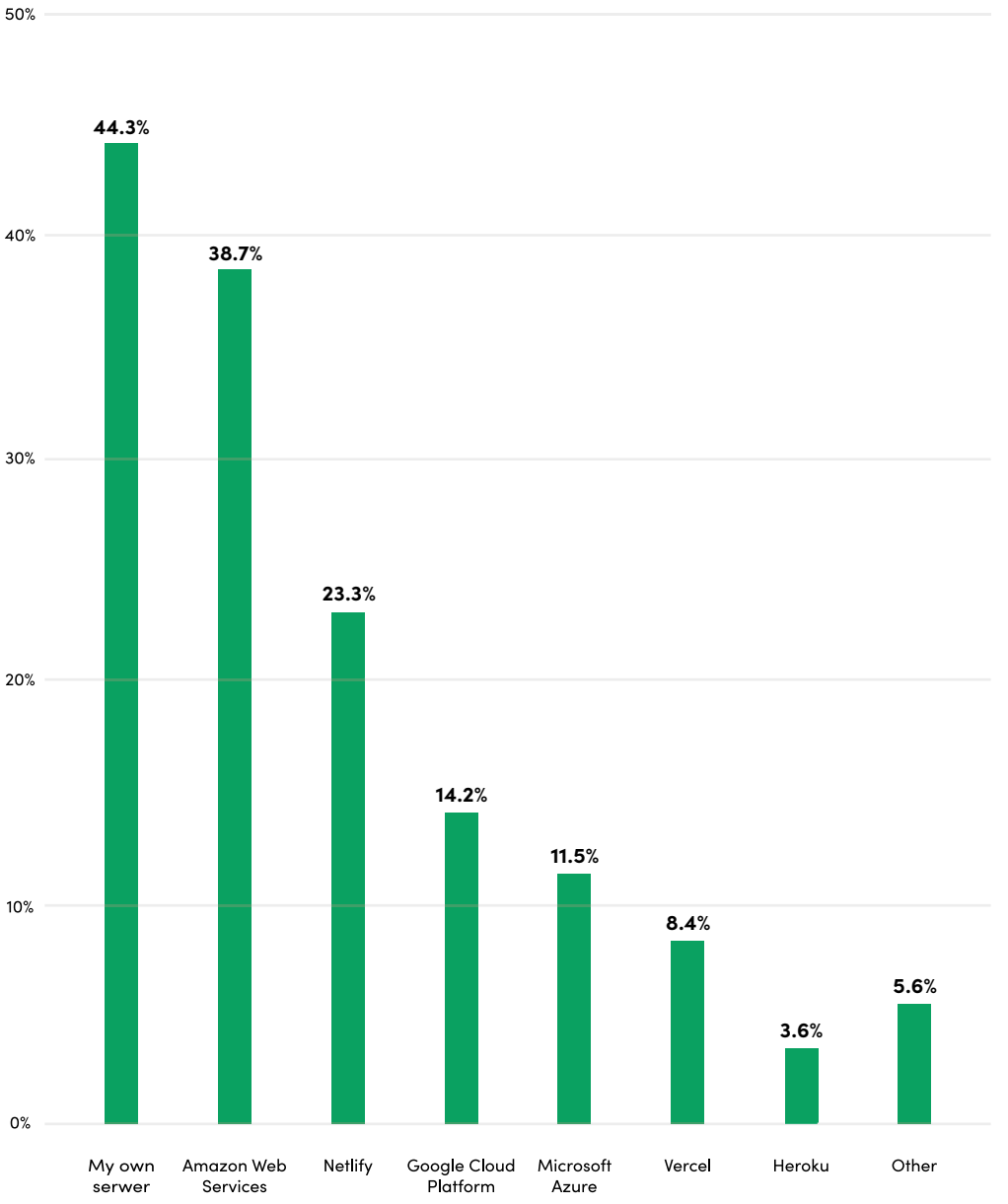
For me, by far the biggest surprise from these results is that 44.3% of respondents are still deploying their applications to their own web servers! Once again, it's a reminder that there is still a massive market for traditional DCs and that there's still much growth opportunities for the public clouds.

It's not really surprising that Amazon Web Services is the most popular deployment target among the cloud providers (38.7%). However, it may amaze (pun intended) some of you that the AWS's share is bigger than those of Google Cloud Platform (14.2%) and Microsoft Azure (11.5%) combined!

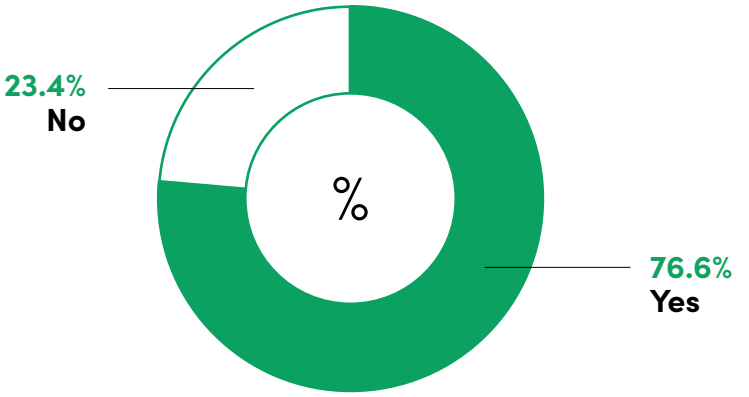
Also, the fact that Netlify has greater penetration (23.3%) than both GCP and Azure is a testament to how great a job they have done. It strengthens a growing school of thought that the future of frontend development calls for platforms such as Netlify and Vercel, which provide easy-to-use and yet powerful abstractions for backend infrastructures for frontend-focused teams.

AWS's continued push for Amplify would suggest they too see the potential here. And, on the other hand, one can't help but feel that GCP had a missed opportunity with their acquisition of Firebase all those years ago and their failure to develop it into the market leader it had the potential to become.

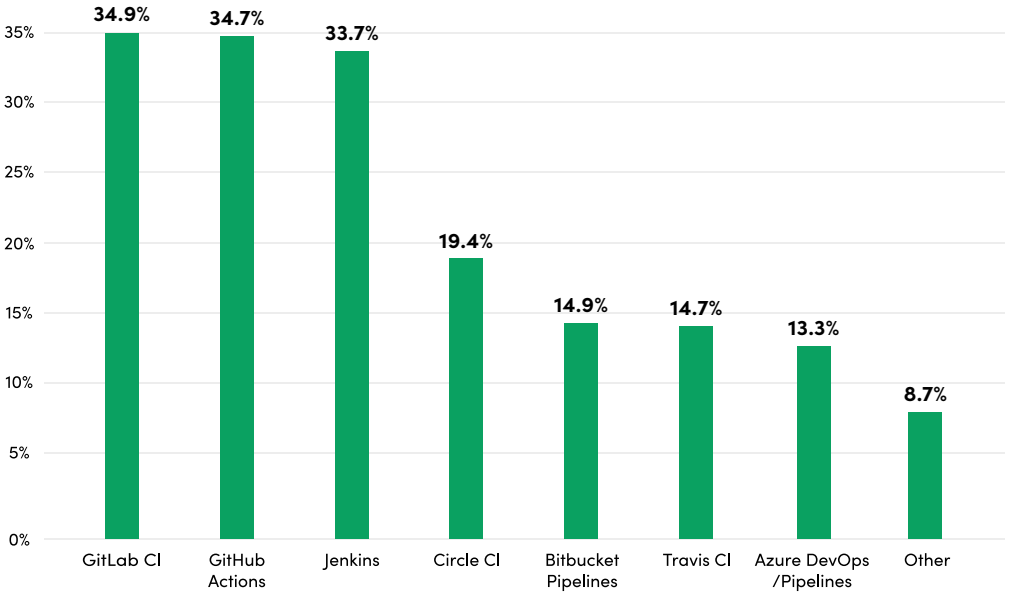
Where do you usually deploy your applications to?



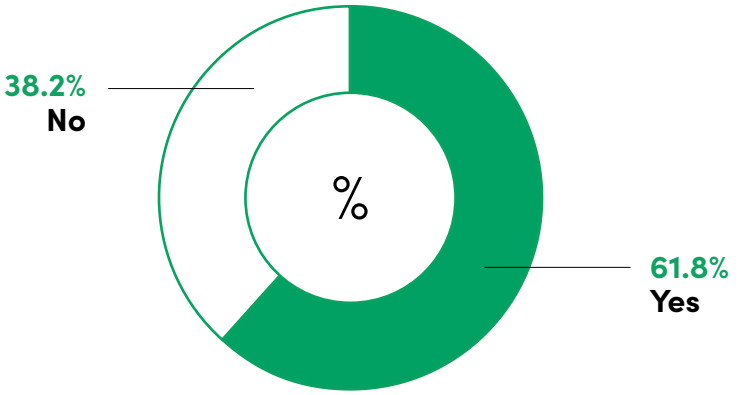
Do you use Continuous Integration?



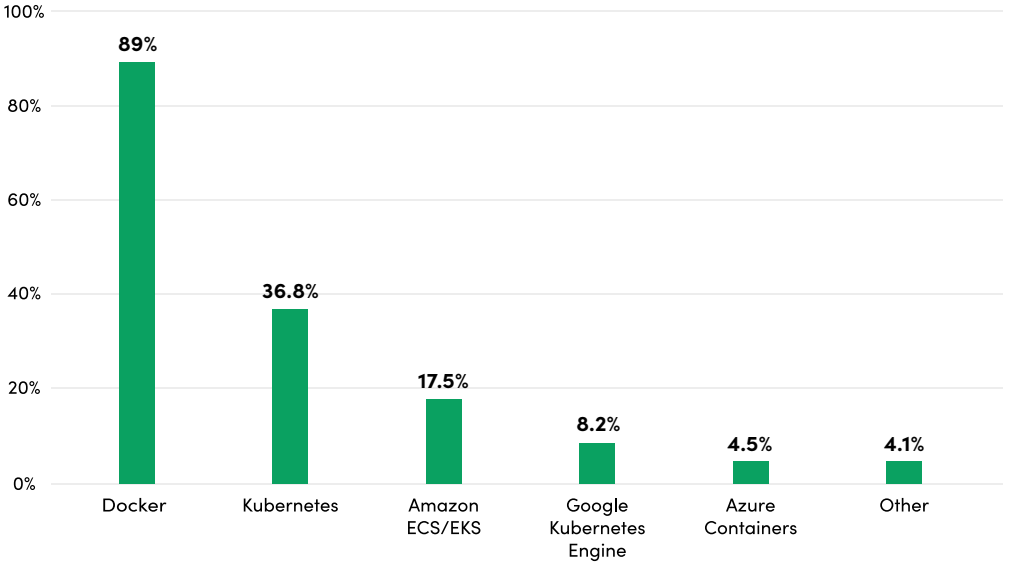
Which CI solutions have you used during the last year?



Do you make use of containerization?



Which container management solutions have you used during the last year?



04.

Jamstack

Ecstatic about static

Tim Neutkens
Head of Next.js at Vercel

> **Let me break the news: we expect an even larger share of developers building Jamstack websites in the upcoming months.**

It's great to see that almost one third of respondents have built a Jamstack (JavaScript, APIs, Markup) website lately. Also, it makes me personally happy that more than half of them have used Next.js – a React framework for Jamstack we've created at Vercel. And let me break the news: we expect an even larger share of frontend developers building Jamstack websites in the upcoming months.

To me, the appeal of Jamstack is that it lets us do less and accomplish more. With Jamstack, instead of rendering a page on every request (Server-Side Rendering), you pre-render a page before request time (Static Generation). This can be shared by all edges on a CDN (Content Delivery Network) for optimal performance, higher availability, lower costs and zero maintenance overhead.

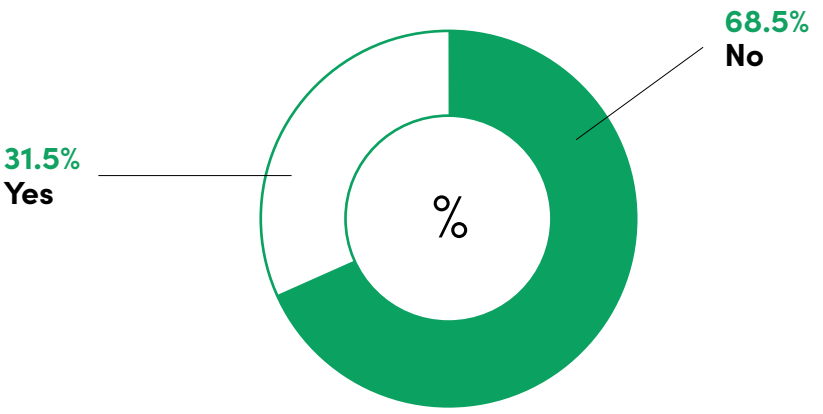
Furthermore, Jamstack frameworks are evolving beyond static and adopting the flexibility of dynamic. For example, Next.js allows you to statically generate additional pages or regenerate

existing pages after the production build (Incremental Static Generation). Even if your app has millions of pages, the initial build will complete instantly as those pages can be generated incrementally.

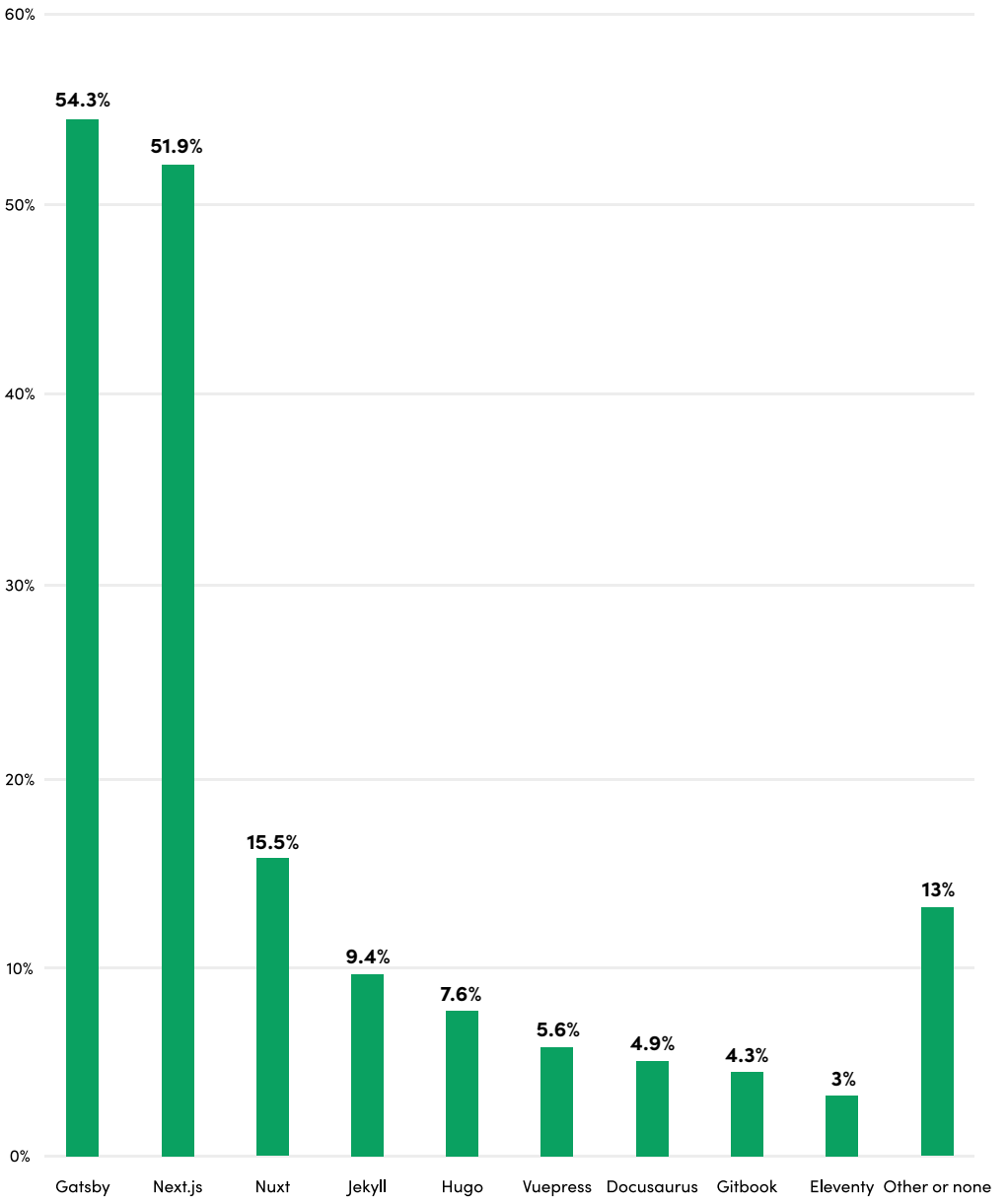
Reusable APIs (the "A" in Jamstack) are on the rise too. The market now has many providers of headless CMS, headless e-commerce, headless identity, and more. No wonder that frameworks are evolving with these trends as well. I'm speaking from experience: Next.js has the preview mode feature which lets you conditionally bypass static generation when you're previewing a page on a headless CMS.

I'm very excited about the future of Jamstack. It's great to see that the respondents are using a wide variety of Jamstack solutions. It shows that developers are experimenting with different ideas – and that's pushing the Jamstack community forward for a more simple and performant web.

Have you built JAMstack websites?



Which static site generators have you used during the last year?



05.

Micro frontends

Do we need microservice revolution in frontend development?

Luca Mezzalira
VP of Architecture at DAZN,
Author of "Building Micro-Frontends"

> **It's still early days and there are many lessons to learn but I believe that micro frontends will evolve and reach maturity – just as microservices did.**

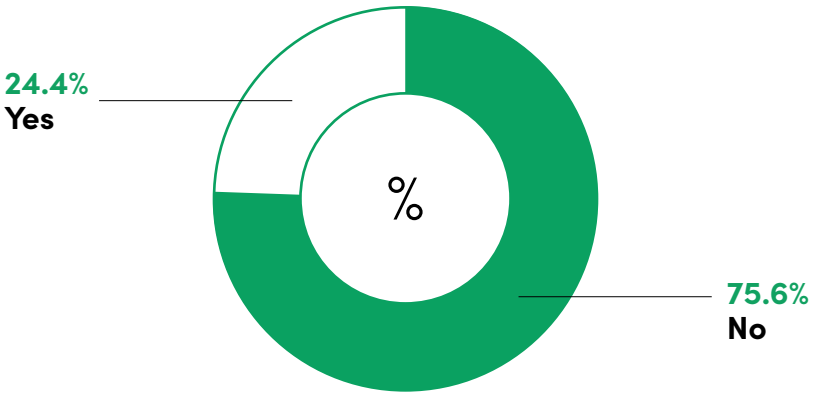
It's incredibly exciting to see how people are embracing micro-frontend architecture nowadays. We already know, that there are many companies around the world using micro frontends – just to mention American Express, DAZN, IKEA, Spotify and Starbucks. Now, with the results of the State of Frontend survey, we also know that practically ¼ of frontend devs have already developed micro frontends.

I think that web components are a great, entry-level solution for developers who are just beginning their adventure with micro frontends – and the results of the survey seem to confirm that. On the other hand, there are quite a few new frameworks available for server-side rendering (e.g., Holocron, Podium and Ara Framework), as well as for client-side composition (e.g., Module Federation or Single SPA). However, you should remember that while these frameworks are a great addition to the micro-frontend community, they should be picked carefully – always looking at the context in which you operate.

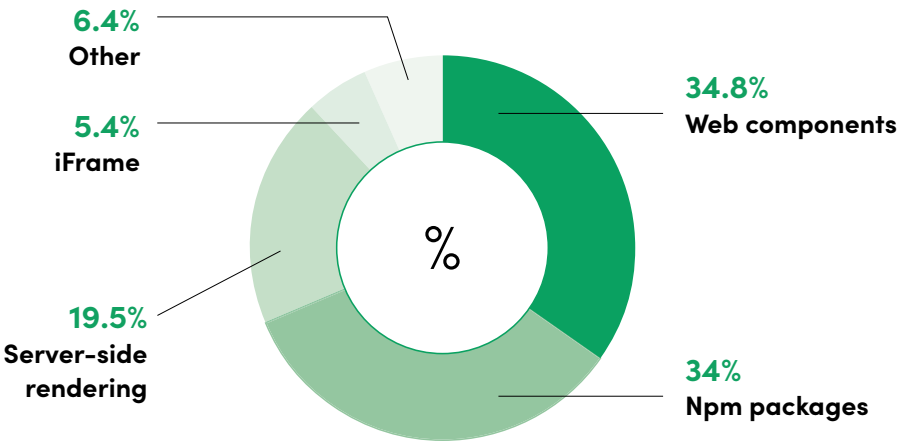
It makes me happy that only 20% of respondents agree with the statement that micro frontends will disappear in 3 years time (see: Chapter 11. Future of frontend). I also believe that the future looks promising for micro frontends – they will for sure evolve and possibly reach maturity, just as microservices did in the past few years. In fact, there are already interesting movements in the TC39 world with the Realms proposal, already in stage 2, that could open up new scenarios for micro frontends.

Micro frontends are not a silver bullet but definitely a nice addition to other architectures such as server-side rendering, Jamstack and single-page applications. It's still early days, therefore, there is definitely more work to do, tons of practices to discover and many lessons to learn. However, I feel very confident that this architecture, when used in the right context, can provide a benefit for scaling projects and teams.

Have you used micro frontends?



How do you compose your micro frontends?



06.

Search engine optimization

It seems that you don’t care about SEO. Here’s why you should

Tomek Rudzki
Head of R&D at Onely

> **Once you understand Google's perspective, it doesn't take much to build websites that are both user-centric and bot-friendly**

Traffic coming from search engines is crucial for any online business. According to the Wolfgang Digital's "KPI Report 2020", organic search is responsible for 43% of traffic. That's more than direct traffic and paid search traffic combined! Still, according to the results of the State of Frontend 2020 survey, as much as 52% of developers don't care about SEO.

I'm not here to cast blame. I guess that some of you develop password-protected, internal applications which don't have to (or even cannot) be visible in search results. However, in other cases, if you want a website that's successful on Google, you must take care of SEO. It's difficult, as SEO specialists don't always speak the developers' language. Allow me to lend a helping hand.

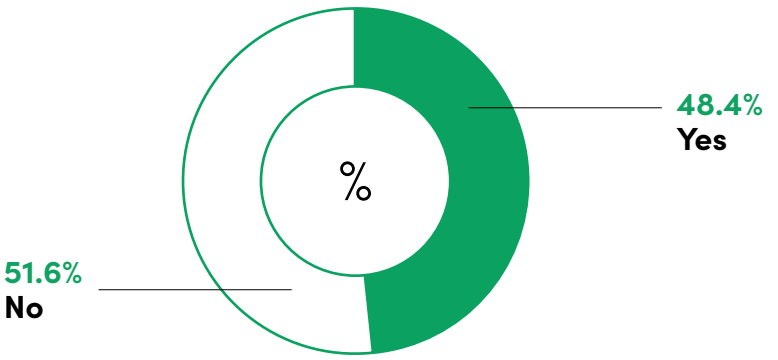
Firstly, you should always make sure that Google can properly render JavaScript on your website. For example, it's possible that you're accidentally blocking some scripts in robots.txt or using JavaScript features that are not supported by Googlebot. I recommend that you use the

Mobile-Friendly Test or the URL Inspection Tool – they are free, easy-to-use tools provided by Google. Using them, inspect the DOM to ensure all important sections of your page can be properly rendered by Google.

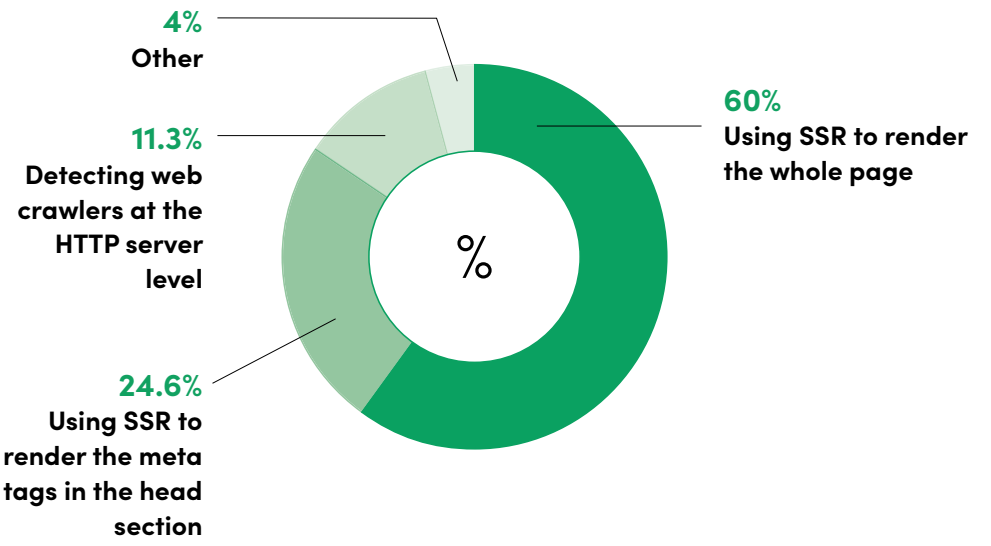
Also, according to the survey, over 11% of developers use dynamic rendering (which is essentially detecting search engine bots and serving them a static version of your page). Google calls it "a workaround for crawlers". It's risky, as dynamic rendering sometimes fails – I've seen websites presenting Googlebot with blank pages, causing their organic traffic to drop to zero. Thus, always make sure you thoroughly test if dynamic rendering works as expected.

SEO is crucial for many businesses, and rendering is just one of many aspects of SEO. You have to put as much focus on using proper HTML tags and designing a logical website structure as you do on choosing between server-side rendering, client-side rendering and dynamic rendering. Once you understand Google's perspective, it doesn't take much to build websites that are both user-centric and bot-friendly.

Do you take care of Search Engine Optimization?



How do you approach the subject of SEO?



07.

Application accessibility

Making the interface friendly
for every user

Rocky Neurock
Engineering Team Lead at Honeypot.io

> **I do think the key to increased accessibility, and better experiences overall, is for those of us "in the know" to really teach our counterparts about the benefits of accessibility.**

"Don't break the web", my friend Melanie Sumner often admonishes. As developers, we're often last in line to promote accessibility in our work. To escape this pattern, we need a shift in thinking. Accessibility won't come to us – we must become great teachers to our peers so the web can work for everyone.

This topic is near and dear to my heart. Not only because I really care about user experience but also because I suffer from impaired vision. I can say with surety that the web doesn't work for me. Small text and low contrast ratios affect me the most but I routinely encounter other frustrations.

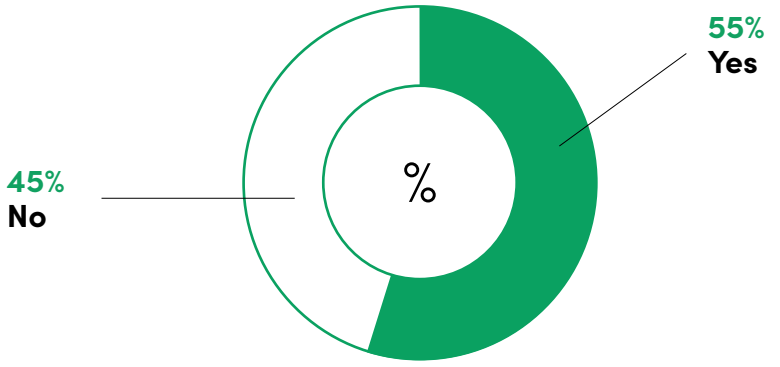
For example, non-native components that reimagine select elements or checkboxes – they can frustrate any user if not done extremely well.

Firstly, consider how nerve-racking these experiences can be for users that prefer getting around the web with their keyboards or for users on mobile devices. Then, think about users with assistive devices. Yes, their experience is even worse.

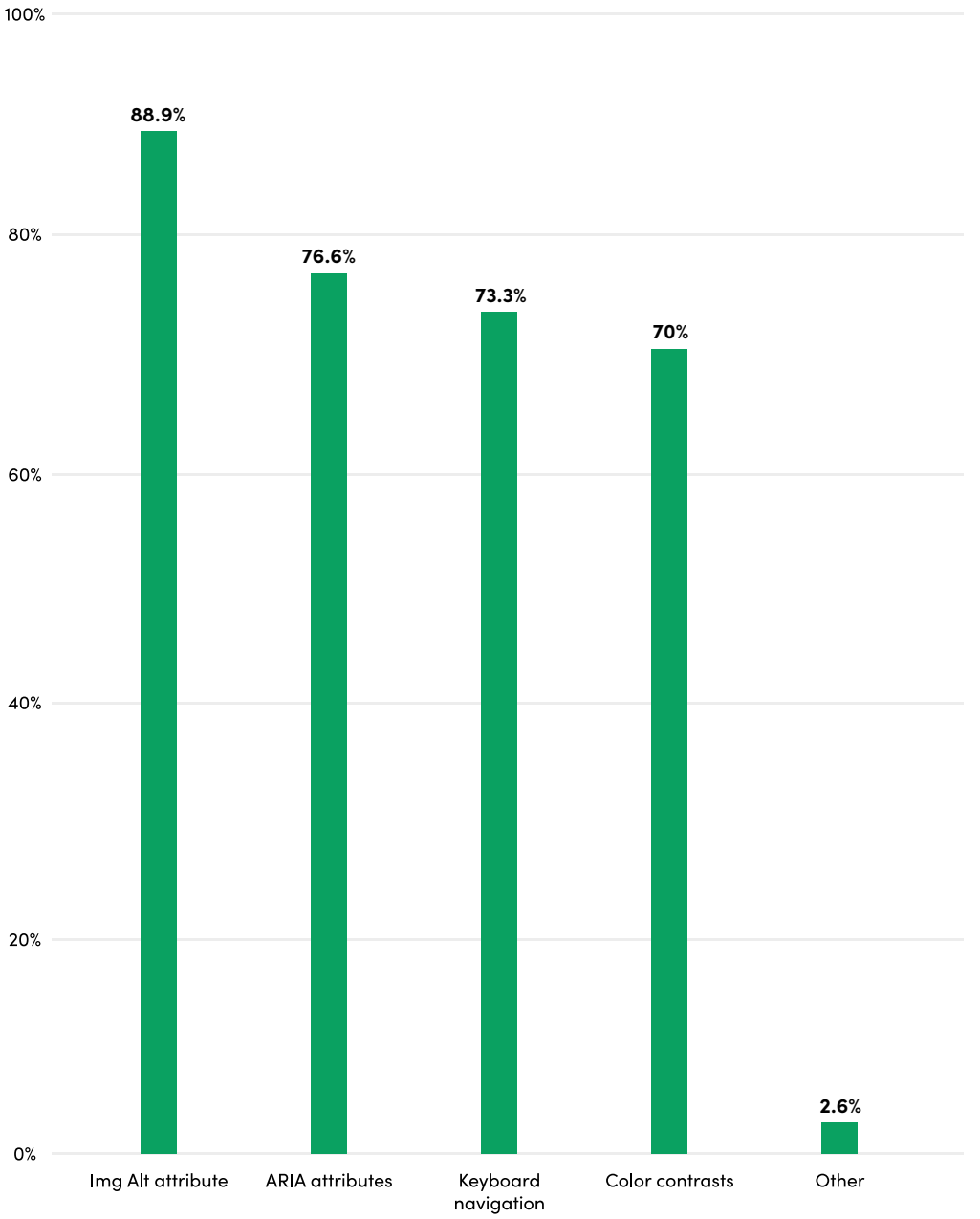
It's good to see that developers who take care of accessibility seem familiar with most of the basic Web Content Accessibility Guidelines (WCAG). In the future, we should also try to find out how many people test for accessibility. There is an ever-increasing number of tools to automatically test accessibility and I wonder if an increase in adoption would correlate to an increase in the percentage of developers who feel responsible for accessibility.

I do think the key to increased accessibility, and better experiences overall, is for those of us "in the know" to really teach our counterparts about the benefits of accessibility. If we can free up some of our own time with automated tests, even better.

Do you take care of application accessibility?



How do you take care of application accessibility?



08.

Development teams

Frontend development?
It’s a team sport

Guillermo Rauch
CEO of Vercel

Recent trends in frontend architecture and deployment infrastructure have influenced how frontend developers collaborate with their team members.

Frontend development is a team sport – shown clearly with 92% of the respondents stating they've worked as part of a development team during the last year. However, recent trends in frontend architecture and deployment infrastructure have influenced how frontend developers collaborate with their team members. At Vercel, we've seen that firsthand.

With the rise of new frontend architectures like Jamstack, frontend developers can deploy the frontend independent of the backend. They no longer have to wait for the full backend test suite to run, resulting in faster iterations. Furthermore, there's a rise in off-the-shelf backend APIs (e.g., headless CMS, identity providers, etc.) that can easily be plugged into your frontend. This enables backend developers to focus more on developing APIs that are unique to the business.

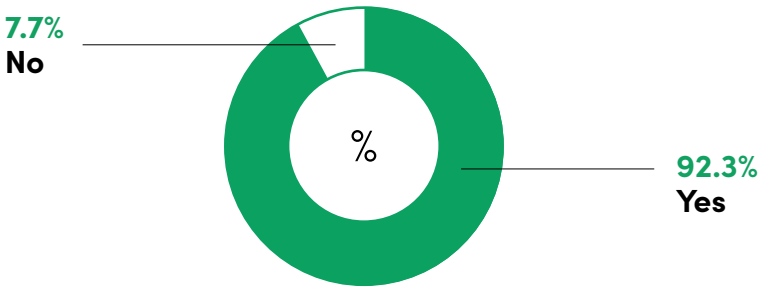
The next change is connected to working with designers and product owners. Because Jamstack apps can be deployed quickly and cheaply

to the CDN edge, it's possible to assign a unique "preview" URL to every branch and every commit. We've done that at Vercel – now, designers and product owners can simply click on the preview URL and instantly see if the changes made by the frontend developer look and work as intended. Much more effective than sharing screenshots and GIFs.

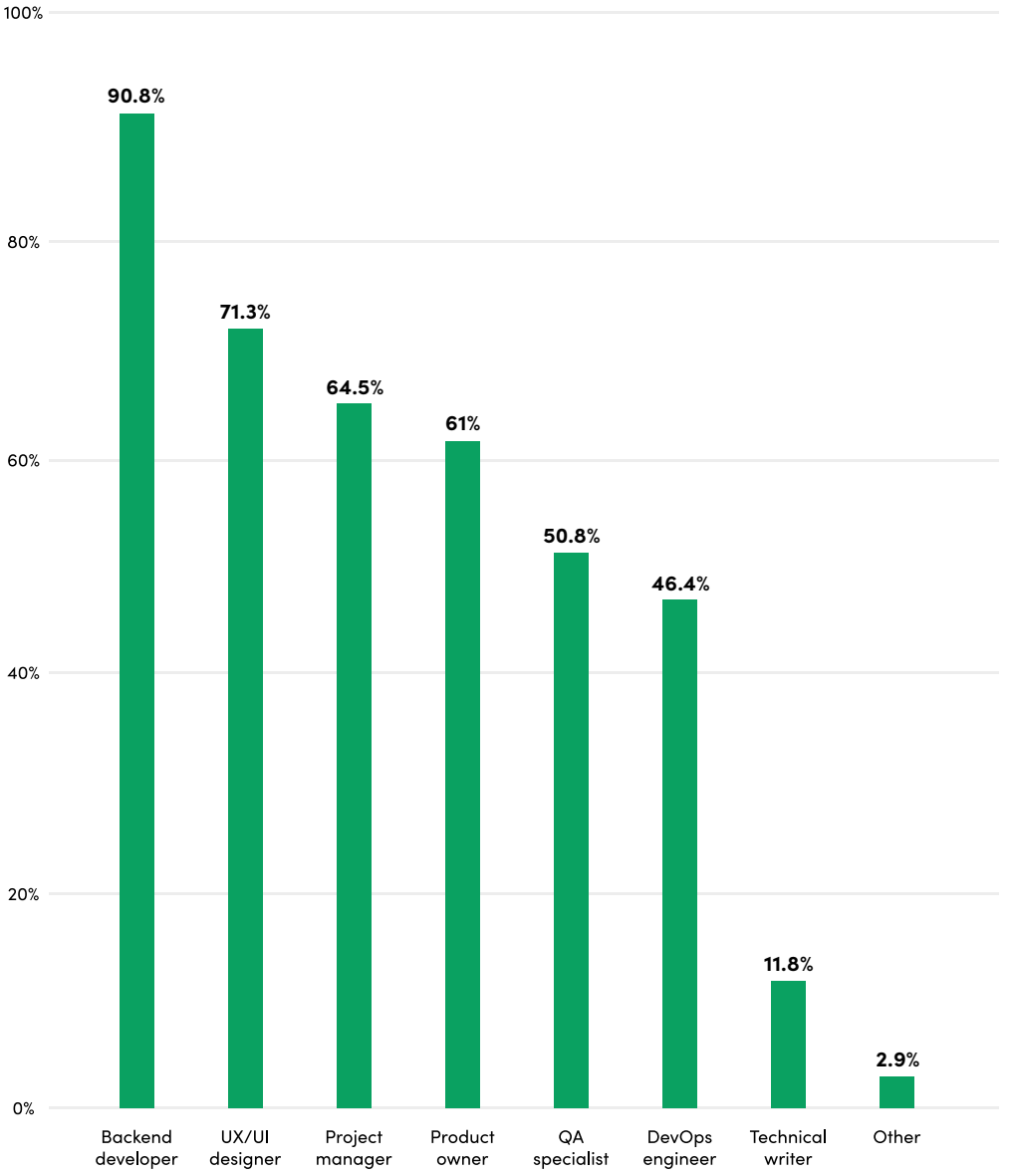
And, finally, software testing. With the introduction of puppeteer, Chrome's headless web browser, combined with serverless compute, end-to-end testing is now fast and cheap. For example, you can have services like Checkly run puppeteer tests — written by QA specialists — against the preview URL. Also, with the rise of Vercel and other frontend deployment platforms which do all the heavy lifting, DevOps engineers can spend less time supporting frontend developers.

Overall, we're very excited about how improvements in frontend architecture and deployment infrastructure are driving changes in developer collaboration. We're looking forward to see more innovations in this space.

Have you worked as part of a development team during the last year?



Which of these people were part of your project development team(s)?



09.

Design

Striving for close collaboration
between designers and developers

Bartosz Skowroński
Head of Design at The Software House

> **With product designers on board, software companies focus more and more on creating strategies and products that go together with the business goals of their clients.**

The debate over the role of graphic designers in software development teams is nothing new – I remember discussing this topic 10 years ago (maybe even before that). However, it seems that we're finally in the place where having a designer working closely with your developers is not a fad anymore but rather a standard. And we've got pretty great tools to make this collaboration even better.

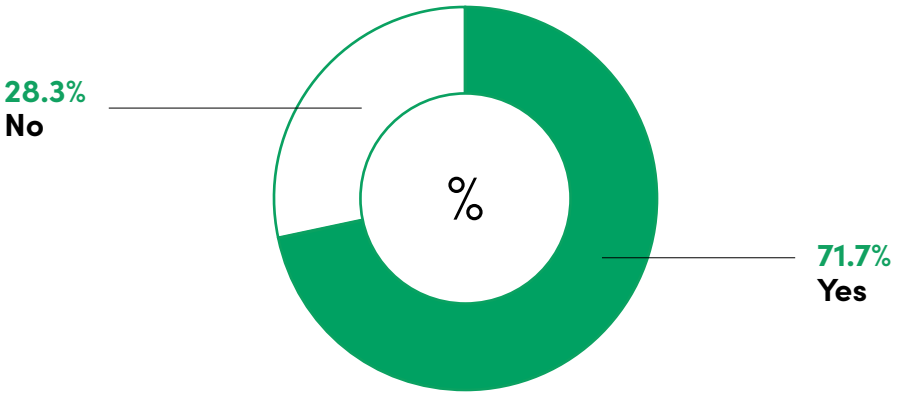
Probably the most basic categorization of design types in software development is: UX design (taking care of the best user experience possible), UI design (making sure that the interfaces have proper look and feel) and product design (thinking about the business of the client and of achieving their business goals). Nowadays, it's becoming a standard for software development companies to have two kinds of designers on board – user-focused UX/UI designers and business-focused product designers.

The emergence of product designers makes me especially happy. It means that we, as software companies, focus more and more on the real needs of our clients, on creating strategies and products that go together with their business goals. And it seems that clients start to appreciate this change – over 70% of development teams around the world already have at least one designer on board (see: Chapter 8. Development teams). Instead of hiring external, freelance designers, clients go for inclusive teams where developers and designers (as well as project managers, software testers and others) can collaborate closely.

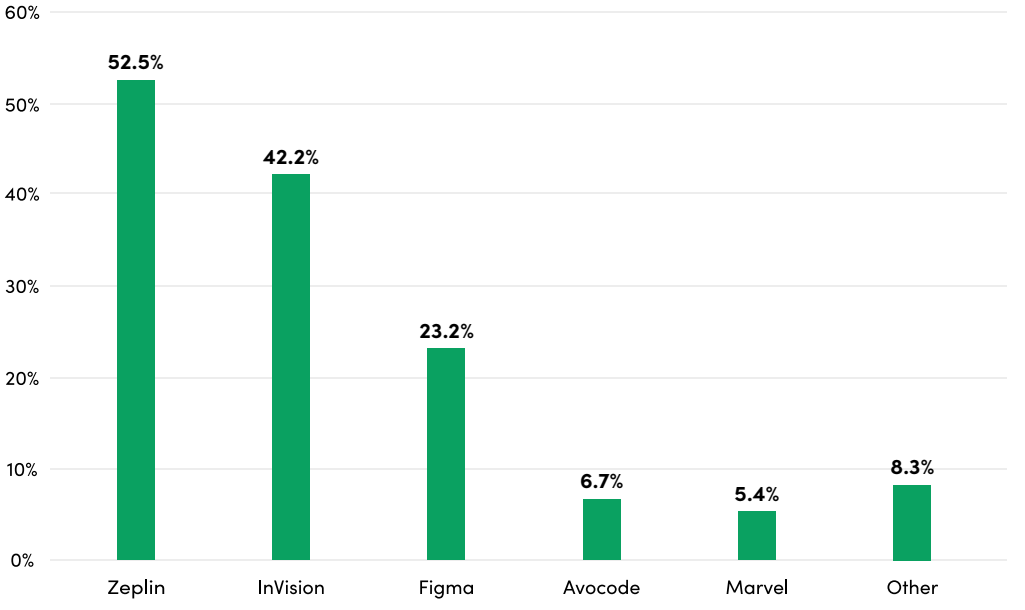
In order to make this collaboration fruitful, we need good tools. For years, designers were using software like Adobe Photoshop as it was hard to find tools tailored to the needs of designers working in the software development business. Fortunately, now we've got plenty of those – Figma, InVision, Sketch and Zeplin just to name a few. They make everything easier: creating vector graphics, collaborating with other designers, handing off designs to frontend developers. It's great that 71.7% of development teams already use such tools.

Although the love between designers and frontend developers can be tough, I think that with the popularisation of inclusive development teams and the emergence of even better design and handoff tools, we can all look into the future with confidence.

Have you used any handoff tools when working with designers during the last year?



Which handoff tools have you used?



10.

Quality assurance

Software testing as the cornerstone of software development

Jessica Jordan
Developer Advocate at .cult

Software testing and modern frontend development are inseparable subjects.

Nowadays, an increasing amount of the functionality of digital products is implemented on the client-side. This makes it obligatory for us – both software engineers and QA specialists – to make testing part of our workflow for developing, maintaining and scaling JavaScript applications. It's good to see that as much as 80% of frontend devs already perform software tests and numbers seem to be increasing over the years.

Luckily, the JavaScript ecosystem offers us a wide set of tools to build robust test suites with sufficient code coverage for the apps we build. In recent years, we see a trend in the JS testing ecosystem to make testing continuously easier to use – with a focus on improving developer ergonomics, integration with other testing solutions and many other aspects.

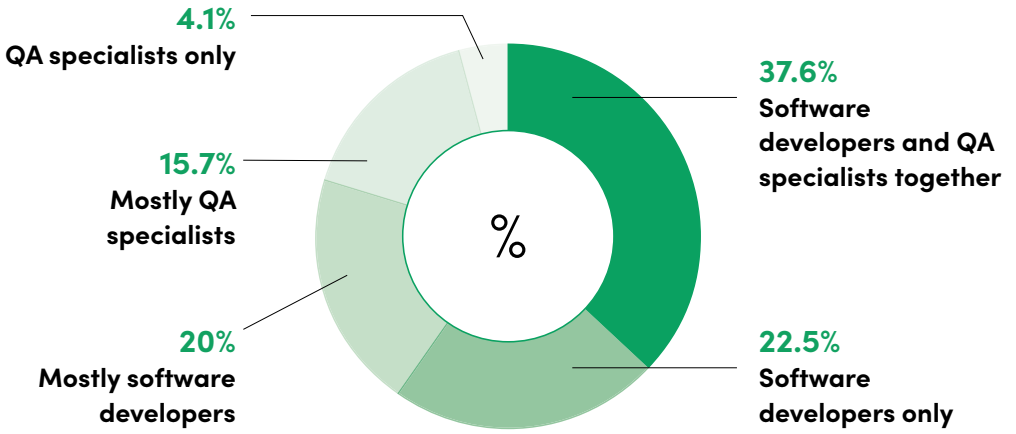
At .cult, many of our software projects – including the application powering our popular job platform Honeypot.io – are automatically tested and gain more testing coverage as code bases grow. Additionally, our QA team manually tests and verifies that the feature requirements are fully met both in functionality and in design – an essential part of our release workflow to guarantee excellent user experience.

Even though our teams allocate additional time for manual and automated testing when developing our platform, we have seen – time and time

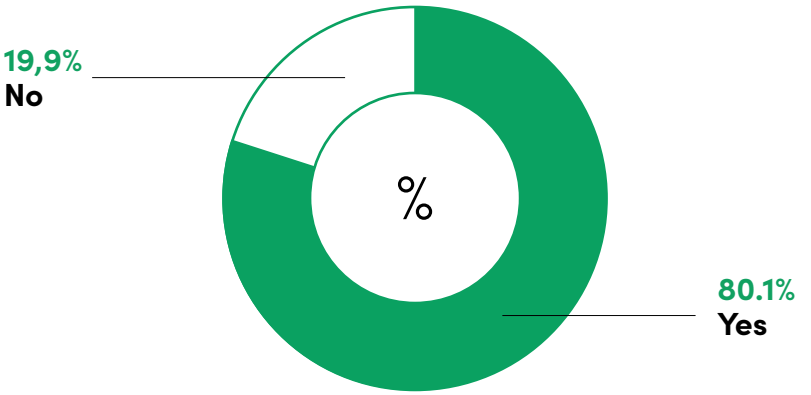
again – that software testing is a necessary investment in the stability of our digital products, ultimately leading to increased productivity in our workflow overall. We trust in tools such as Capybara, RSpec, Ember CLI and QUnit for unit, integration and end-to-end testing. And, of course, there are even more solutions out there for you to choose from.

At .cult, we believe that the continued growth of the tooling ecosystem for testing will soon allow us to cover an even larger part of the product development workflow through automation. And why do we do that? Because we know that software testing and modern frontend development are inseparable subjects.

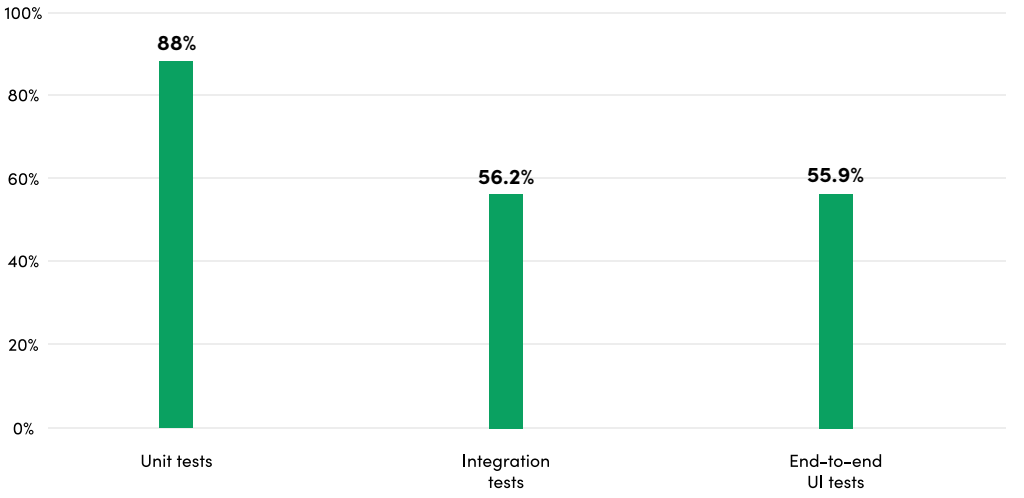
Who was responsible for testing in your software development teams?



Have you performed software tests yourself during the last year?



What kinds of tests have you performed yourself?



11.

Future of frontend

State of Frontend 2021?

Marek Gajda
CTO of The Software House

> **In the frontend development community, the line between love and hate is very thin.**

Am I surprised by the results of the survey and the recent changes in frontend development? Not really. Am I surprised by how quick these changes occur? Definitely yes. And that's why predicting the future of frontend is not an easy task.

When you look at the state of frontend development, there are some well-established technologies, tools, good practices – choices that seem obvious. Let's take JavaScript frameworks. When you see that there are more people using React than those using Angular and Vue.js combined (see: Chapter 2. Frameworks), you realise that React has gained such a solid reputation that it probably isn't going anywhere in the near future.

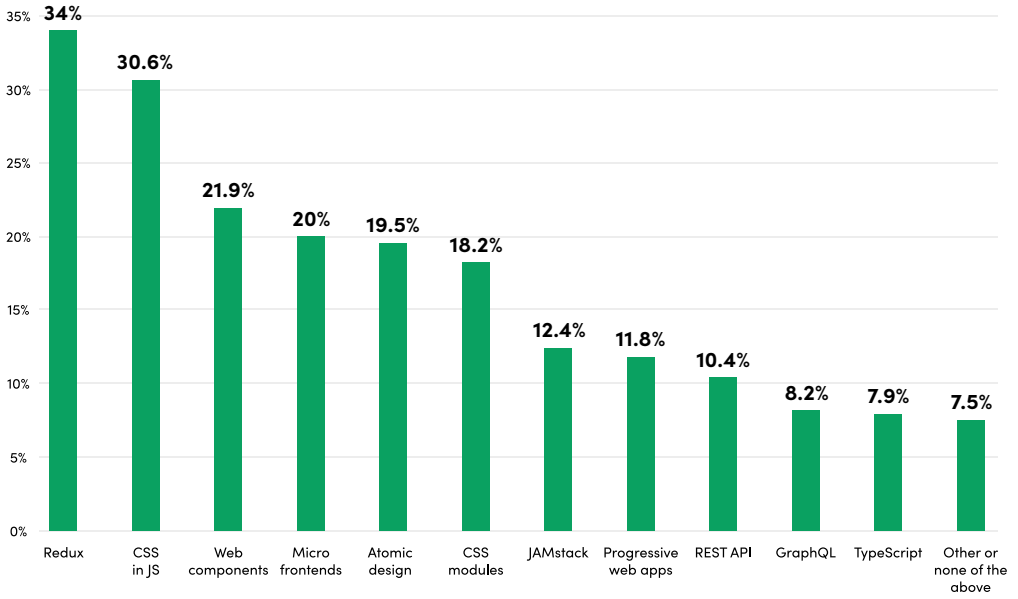
However, in the frontend development community, the line between love and hate is very thin. And probably the best proof of that is what happened to Redux. A year or two ago, when you were working with React, Redux was also "the obvious choice". But frontend developers got tired of the problems caused by using Redux and quickly jumped on the React hooks bandwagon. It's summer 2020, already more people use hooks than Redux (see: Chapter 2) and as much as 34% of frontend devs believe that Redux will be gone in 3 years from now.

Also, the world of frontend development is getting more and more complex. Again, a year or two ago, solutions like Continuous Integration and containerization were considered more of a backend thing. But, in the meantime, frontend developers realised that they too can benefit from using those solutions. Now, 77% of frontend devs use CI and 62% use containers (see: Chapter 3. Hosting) making them a new standard in frontend development.

So, how will the state of frontend development change in the next 12 months? Will Svelte become one of the 3 most popular frameworks? Will micro frontends reach maturity? Nobody can tell for sure but, in my opinion, one thing is certain: we'll be surprised by how quick some of the changes will occur.

That said, see you soon in the State of Frontend 2021 report!

Which of these trends/solutions will be pretty much dead in 3 years from now?



Software development has changed for good

We help tech managers who understand this change and want to make the most of it

Modern real-time frontend

Tech stack migration

Cloud migration, serverless

Microservice architecture

Find out how we can help you:

THE SOFTWARE HOUSE

www.tsh.io