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.

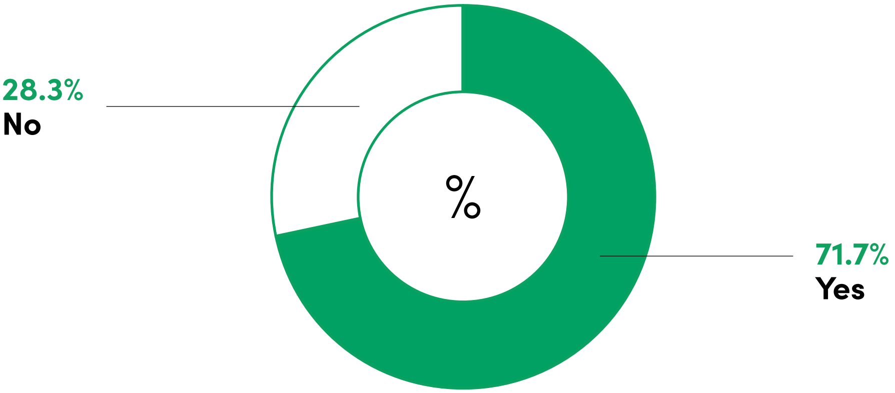
Probably the most basic categorization of design types in software development is: UX design (taking care of the best user experience possible), UI design (making sure that the interfaces have proper look and feel) and product design (thinking about the business of the client and of achieving their business 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.

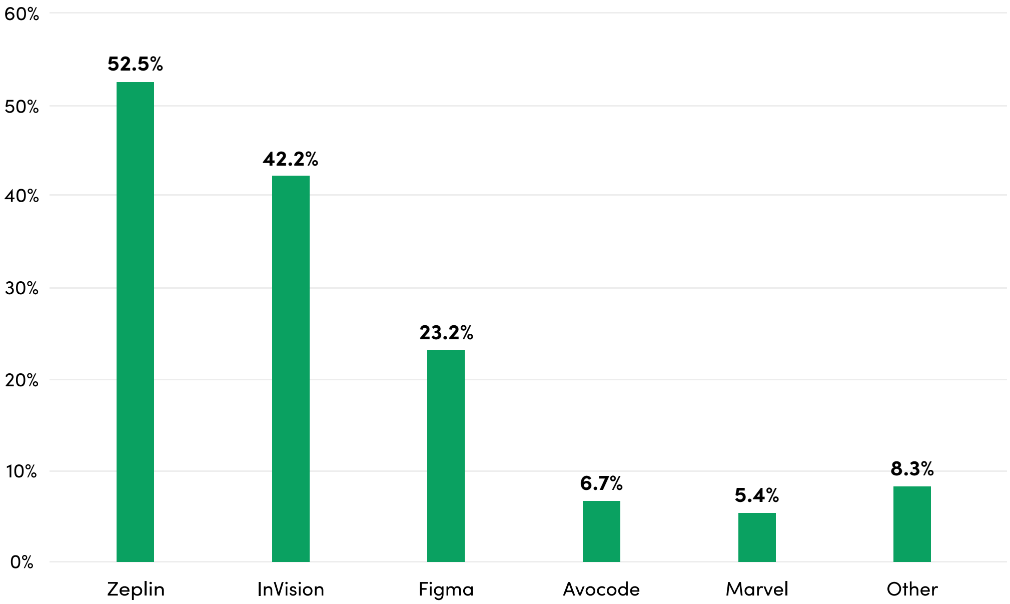
In order to make this collaboration fruitful, we need good tools. For years, designers were using software like Adobe Photoshop as it was hard to find tools tailored to the needs of designers working in the software development business. Fortunately, now we've got plenty of those – Figma, InVision, Sketch 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.

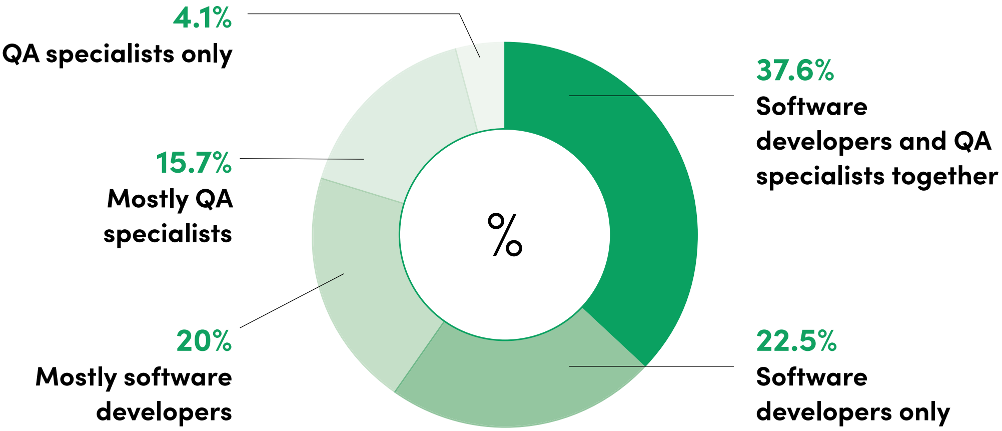
Luckily, the JavaScript ecosystem offers us a wide set of tools to build robust test suites with sufficient code coverage for the apps we build. In recent years, we see a trend in the JS testing ecosystem to make testing continuously easier to use – with a focus on improving developer ergonomics, integration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

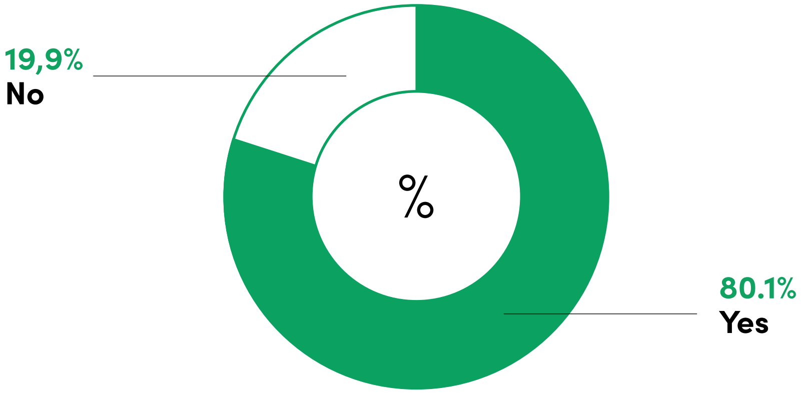
Even though our teams allocate additional time for manual and automated testing when developing our platform, we have seen – time and time again – that software testing is a necessary investment in the stability of our digital products, ultimately leading to increased productivity in our workflow overall. 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.

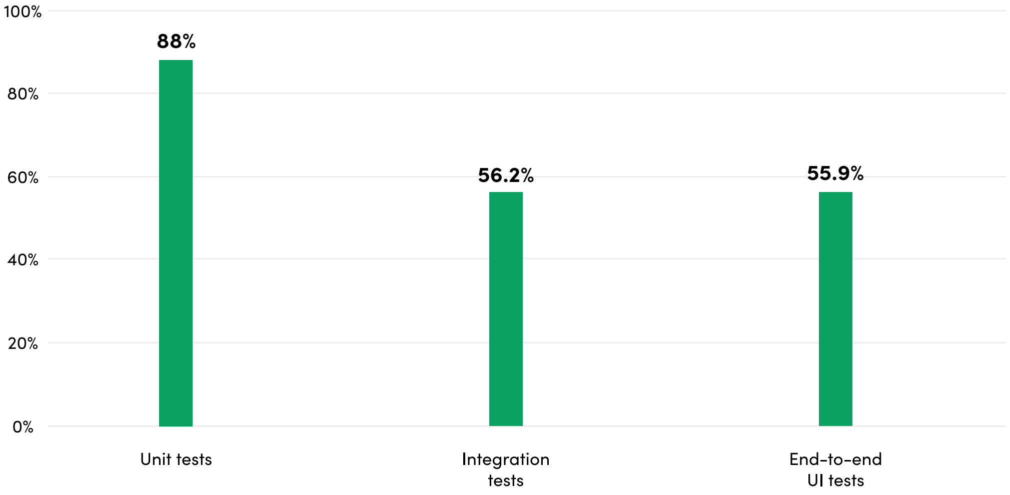
Who was responsible for testing in your software development teams?



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.

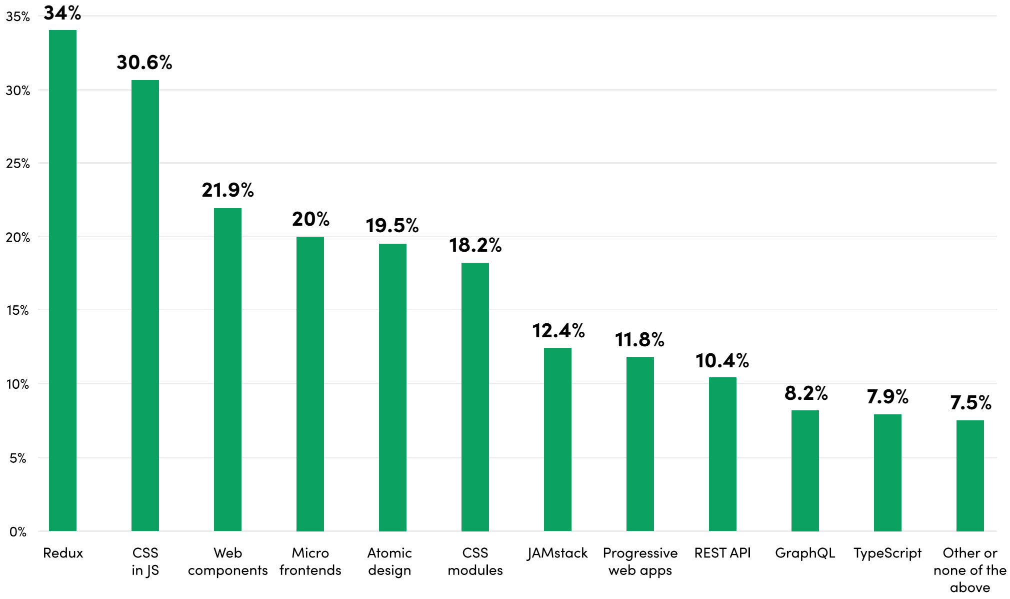
However, in the frontend development community, the line between love and hate is very thin. And probably the best proof of that is what happened to Redux. A year or two ago, when you were working with React, Redux was also "the obvious choice". But frontend developers got tired of the problems caused 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