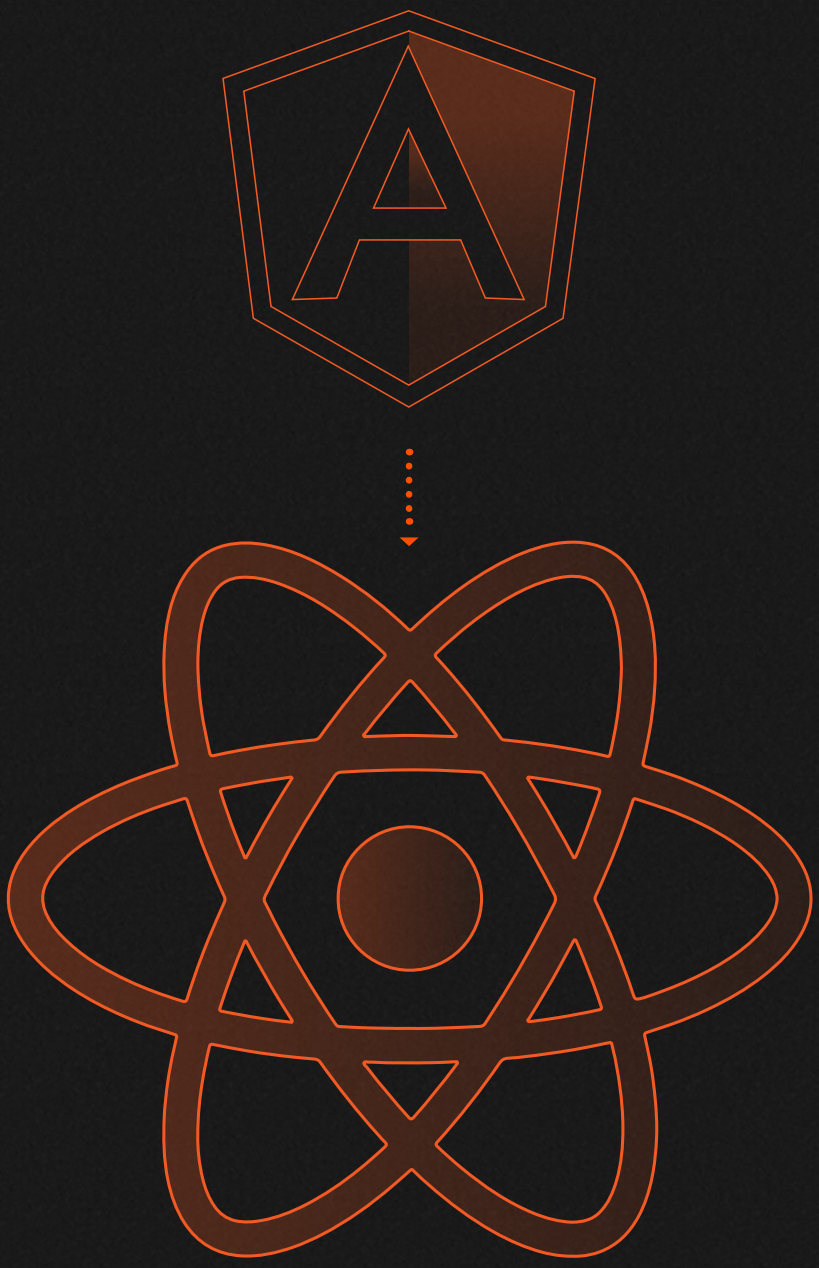
merixstudio



Why and how to migrate from AngularJS to React?

www.merixstudio.com

The End of AngularJS

In a little bit more than a year from now, AngularJS in the LTS version will lose support from Google. That's the definitive end of improving and fixing the framework. It doesn't mean that the apps using it will stop working straight away but if they are about working flawlessly or even be scaled - there's no other way but to migrate to another solution. How to do it properly and why you should choose React?

WHY MIGRATE ANYWAY?

Because sooner or later you'll have to. Although 10 years ago, when AngularJS was released, it was considered revolutionary, Google decided to end the project and the support for version 1.7.x LTS will last only until **July 1st, 2021**. There's still time to prepare software for the change and avoid technical debt and its effects - bugs, performance loss, problems with software stability, lack of code coverage and most of all - security breaches.

Oversleeping the moment to migrate the code will cause a lot of troubles with maintaining the app as there are **fewer and fewer libraries and other ready-made solutions** that simplify software development (e.g. there's no official command-line interface - CLI). Programmers are doomed to write their own, custom tools and even without that, they are not really keen on using the outmoded technology with a forthcoming expiry date.

So not only scaling the app on its own will be challenging but also the dev team that takes care of it.

There are a lot of other problems owners of the products built with AngularJS will or already are struggling with. Real Sisyphean task is fighting to keep the effective performance of the app which is undermined by one of the main functionality of the framework - **the two-way data binding**. The idea that initially appeared to be state-of-the-art with the increasing complexity of the app, could be overkill.

Thankfully, over the past decade, since AngularJS was introduced, frontend frameworks have come a long way, developing to fulfill the increasing tech requirements, ensure effective code writing and best user experience. So, what are the options within migration from AngularJS and which one of them fits your project the best?

With a bang or bit by bit - how to migrate?

The whole project of migration requires a deliberative plan and making a number of more or less vital decisions. A botched process ends up with data mess, dirty code, and even discontinuity in performance. That, in turn, leads to user's discontent and might discourage them from the product.

The first choice to face while planning any upgrade considers the way we want to pursue it. No matter if it concerns code, data, or e.g. whole ERP system, in software development, we usually distinguish two main migration strategies.

BIG-BANG MIGRATION

Big-bang migration indicates a complete re-writing of the code using a new framework/library.

It eventually means creating the app from scratch or rather reconstructing it with a more modern and reliable solution. In most cases, this type of migration is **simpler and faster** to run - a proficient dev team should be able to rebuild a medium-size app in 2-3 months. For a number of our clients, it became a great **opportunity to make changes** deferred for a long time because of the technological limitations of the existing solution. It might also become accurate timing to rethink the architecture and business logic and modernize them if needed, as well as to plan to add new features or upgrade current ones.

Despite a whole plethora of unquestionable advantages, there's one main challenge of the big bang approach - **the moment of deployment**. This is a critical point as the final transition is made in one go. It might be risky and require the engagement of a bigger team and perfect coordination of its work. It can also entail downtimes and hard to identify errors as all elements are introduced together. A flawless preparation for this crucial moment and proper risk mitigation should be the number one concern within the big bang migration process.

PROGRESSIVE MIGRATION

The second approach assumes a gradual upgrade made by "injecting" new technology to an existing AngularJS code and mixing them.

Combining two solutions and splitting the migration into pieces makes **the process less disturbing for the end users but way more complicated and time-consuming**. It also takes

longer to prepare a comprehensive plan including information about the way of dividing the app to upgrade its particular parts and establish the sequence in which they will be deployed.

The dual-code obligates devs to know both applied solutions during the whole process. It may take many months to oust AngularJS completely from the app. And, as we mentioned before, it will be more and more difficult to **find programmers eager to work with AngularJS** and also a complex, unclear code build to accommodate two different frameworks.

Nevertheless, as progressive migration proceeds in a stepwise way, it's sometimes the only option to upgrade large-scale apps with the extensive codebase or the ones that are frequently updated. It's also a good match for the microservices projects which are based on dividing the product into parts and running them separately.

Big-bang migration	Progressive migration
PROS	
Simpler and faster process for developers	Less disturbing deployment for the end users
Opportunity to re-architecture the app and rethink its business model	A good choice when the app is frequently updated
More coherent and clean code	Suitable for large-scale apps
CONS	
Risky deployment engaging lots of people	More complex and time-consuming process
The process is hard to run when the app has an extensive codebase	Possible losses in performance when mixing AngularJS with other frameworks/libraries
	Complicated code joining two different technologies

Which solution to choose to migrate to and why it should be React?

It might seem like Angular should be a natural choice for migration from AngularJS. Even though they were both created and developed by the same company, there's **no direct update path** from one to another. Google ensured solutions to simplify the transition and enhance the reusability of the code but still, **Angular 2 is considered as a completely new platform** that has very little resemblance to the forerunner. One of the biggest differences is the programming language the two technologies are based on - **the first harness JavaScript while the other uses TypeScript**. That means the migration will require more time to transcribe particular solutions, architecture, and schemes.

REACT - ESTEEMED BY DEVELOPERS...

Above-mentioned problem doesn't concern React as it completely relies on JavaScript (although it is possible to include TypeScript in it). It also provides conditions calculated to apply functional programming principles and patterns that conduce to producing highly performing code. Furthermore, Facebook puts huge attention to make **React follow the most recent programming standards and technical progress**. It shouldn't surprise anyone as the library runs most of Facebook's applications. The company invested a lot (and still does) in React which decreases the risk it will follow Google's footsteps and turn into tech George R.R. Martin, razing its projects without much warning. As the library is developed in a consistent way it's backward compatible with previous versions and arguably it will be compatible with the future ones.
Thanks to the React's API's small size, utilizing declarative programming and component-based UI development, the library offers **a low entry threshold**. And not only the onboarding is easier but also further mastering the development thanks to engineering resources delivered by Facebook and a proactive and engaged community (144K stars and 27.8K forks on Github!) that provides a plethora of ready-made libraries, components, and other solutions. That remarkably influences the developers' experience and speeds up coding. All the above advantages translate to high popularity within developers and so for companies - bigger chances to compose a dream-team responsible for the migration and further development of the app.

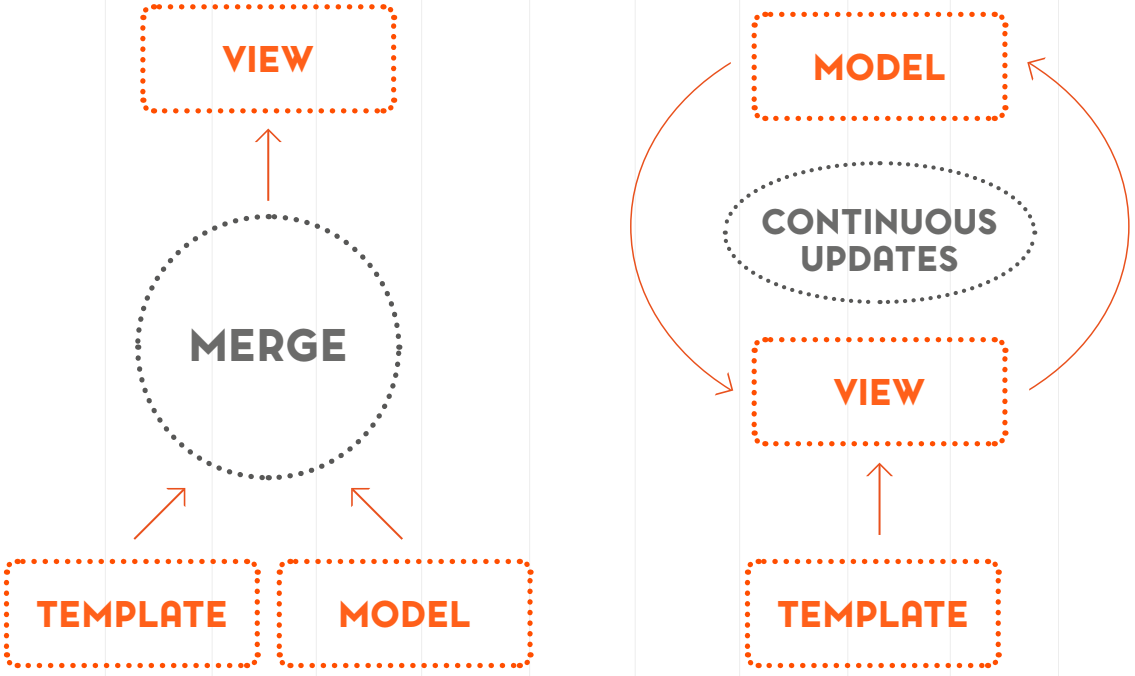
AND MADE FOR BUSINESS

But React is appreciated not only by programmers. Some even say it grew out of solving business problems rather than technical ones.

Facebook put immense attention to provide **high performance** for products made with React. You (and your app's users) will definitely feel the smoothness boost after the transition from AngularJS. Just to mention, React needs 1000 ms to execute JavaScript code which is five times less than for Angular 1. The difference arises mostly from **the approach to data binding**. As it was already said, AngularJS uses a two-way system which means that changes in the model are automatically reflected in the view and vice-versa. That creates a lot of dependency between models and views which can end up in an infinite loop of cascading updates. The performance problems are also associated with "dirty checking" necessary for AngularJS to find out if the value has changed at any point. When the app is growing to medium size and reaching about 2000 bindings on a page, it requires a lot of time to check the modifications in the model and view.

React is based on the **unidirectional flow** - data are passed from the highest component to the lower ones (although there is a possibility to build a two-way binding model). Developers are more in control with the changes in view and the app is not overloaded with heavy digest cycle.

One vs. Two Way Data Binding



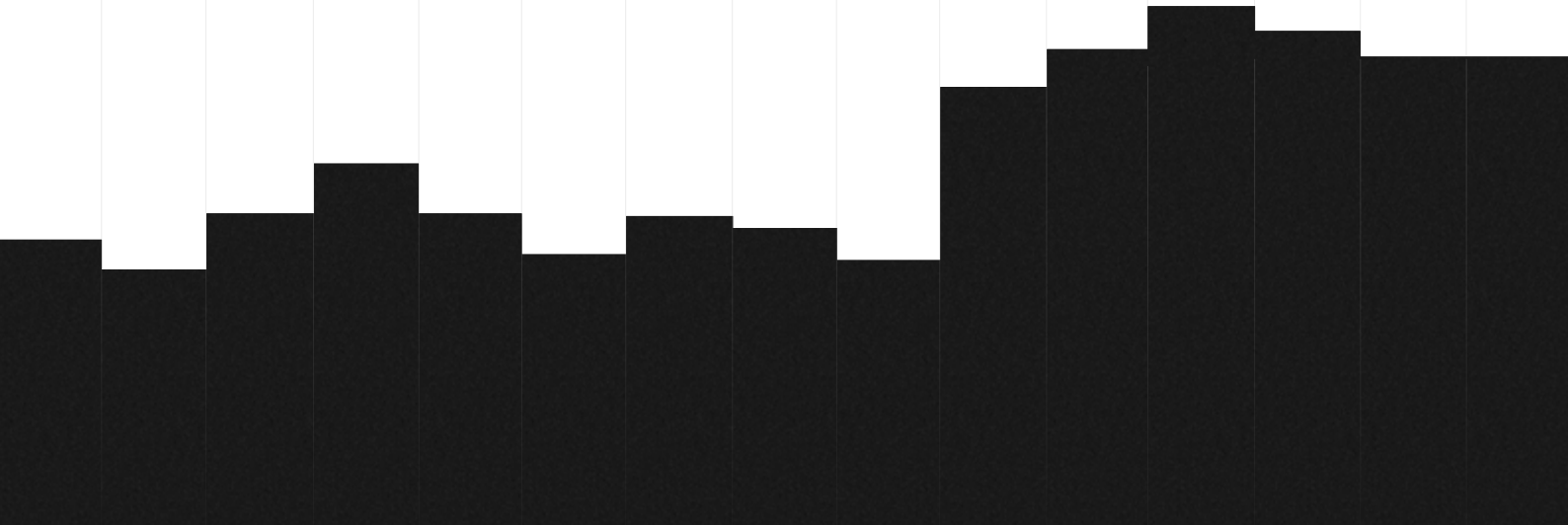
That leads to improving the user experience with more effective work of the app, quicker loading and less time for elements to render in the DOM.

Facebook couldn't have forgotten about **SEO** issues being aware of how significant is the visibility of the web apps for bots and crawlers. As Google still

has problems with seeing JavaScript content (even if some say it copes better and better) and React is fueled with JS, the goal was to improve website visibility with **Server-Side Rendering** (SSR). This method provides HTML code to the browser so the entire view shows on the screen immediately. There are a number of solutions to harness the possibilities of Server-Side Rendering in React but we recommend Next.js as the simplest and most effective one. To improve SEO as well as the performance of React web apps you can also use Lazy Loading. It reduces the time of the initial loading and limits the number of requests which have a huge impact on conversion rates and the ability to engage a visitor.

A year from now you will wish you had started today

If you haven't already made the shift, you should definitely start to seriously think it through. There's some time to consider different solutions but when the app is complex and already delivers problems with maintenance, the migration will probably be more complicated and requiring a deliberative plan.



Looking for a reliable partner in your migration process?

Let's do it together!

contact@merixstudio.com

+48 570 001 928

ul. Małachowskiego 10, 61-129 Poznań

See our work Bēhance dribbble merixstudio

Check what our clients think about us Clutch

Follow our stories

merixstudio

120+ full-stack agile software experts ready to support your business