← Back to Blog

Thursday, June 22nd 2023

Next.js App Router Update

Posted by

 Delba de Oliveira
@delbaoliveira

 Lee Robinson
@leeerob

The App Router represents a new foundation for the future of Next.js, but we recognize there are opportunities to make the experience better. We'd like to give an update on what our current priorities are.

For the upcoming releases of Next.js, we are focusing on the following areas:

- **Improving Performance**
- **Improving Stability**
- **Improving Developer Education**

The App Router

First, it's helpful to provide some context on how the App Router has been designed.

Growing Beyond the Pages Router by Aligning with React

As we saw increased adoption and larger scale applications being built with Next.js, we received feedback from the community and identified areas where we started to reach the limits of the Pages Router.

Most notably, the Next.js Pages Router was not designed for streaming, a cornerstone primitive in modern React, that helps us address the limitations we were facing and realize our long-term vision for Next.js.

Making streaming-friendly framework APIs for data fetching, asset loading, and page metadata, as well as taking advantage of React's newer primitives required large changes to the core architecture of Next.js.

We took the opportunity to build on top of the latest React concurrent features↗, like Server Components, Suspense, and more, which have been designed for streaming architectures↗.

Incremental Adoption is Non-Negotiable

We didn't want our community to have to rebuild their entire applications from the ground up to update to the latest version of Next.js. We believe incremental adoption is the best strategy for evolving applications over time.

- **Per-route incremental migration**: Without a major rewrite of your application, you can move a single route of your application over the App Router and start to take advantage of new features at your own pace. See our incremental adoption guide or watch a tutorial↗.
- **Easily rollback**: If you are not satisifed with the performance or developer experience of the App Router, you can easily rollback to the Pages Router for that specific route.

We are exploring further opportunities to make incremental adoption even easier.

Road to Stability

We began building the Next.js App Router over a year ago and have been steadily releasing new features and improvements since then.

- **Initial Announcement**: In May of that year, we released an RFC to outline our plans for making routing and layouts more flexible.
- **Early Beta**: In Next.js 13, we released the first version of the App Router, allowing the community to try it out and provide early feedback.
- **Stable API**: Responding to feedback, we focused our efforts on finalizing the core API. In 13.4, we marked the the core API of the App Router as stable and ready for wider adoption.

Our Current Focus

Marking stability signaled to the community that the core API was settled and would not go through major breaking changes that would require rewrites.

Since then, we've received lots of valuable feedback and increased adoption has inevitably revealed bugs and opportunities for further improvement.

We want you to know that we are **not yet satisfied** with the experience of using the App Router and it is our top priority moving forward. So, let's talk about the work we're doing to make this experience better.

Improving Performance

Over the coming months, we're focused on three aspects of performance: local iteration speed, production build times, and serverless performance.

Local development performance

As Next.js has matured, and the size of applications built with it have grown, we've slowly and incrementally been replacing pieces of its underlying architecture with faster, more scalable tools.

- **Migration Progress**: We started with replacing Babel *(compilation)* and Terser *(minification)* with SWC. This has helped improve local iteration speeds and production build times.

- **Long-term Investment**: Keeping great Fast Refresh performance regardless of an applications size means making Next.js operate as incremental as possible during local development, by only bundling and compiling code as needed.

  This is why we're currently working on replacing webpack *(bundling)* with Turbopack ↗, which is built on a low-level incremental computation engine that enables caching down to the level of individual functions.

  Next.js applications that move to Turbopack will see sustained improvements in Fast Refresh speed even as they grow in size.

  In the past few months, the Turbo team has been focused on improving Turbopack performance and support for all Next.js features and App Router APIs.

  Turbopack is currently available in beta (`next dev --turbo`).

- **Improving Today's Architecture**: In addition to investing in the future, we are continuing to make performance improvements to our existing webpack architecture.

  For certain Next.js applications, especially those refreshing thousands of modules, we have seen reports of flakiness with local development and Fast Refresh. We're working to improve performance and reliability here. For example, we recently added in pre-configured settings (`modularizeImports`) to handle large icon libraries ↗ that might accidentally force thousands of modules to reload on every request.

Build-time performance

We are also working on production builds with Turbopack (`next build --turbo`) and have started to land the first pieces of this work. Expect more updates on this in the coming releases.

Production performance

Finally, on Vercel, we're working to optimize the performance and memory usage of Vercel Functions defined through Next.js application code, ensuring minimal cold starts while retaining the benefits of a scalable serverless architecture. This work has resulted in new tracing capabilities (experimental) in Next.js and early explorations into server-side developer tools.

Improving Stability

The Pages Router has been around for six years now. The release of the App Router meant the introduction of new APIs which are still young, with just six months of usage. We've come a long way in a short amount of time, but there are still opportunities to improve as we learn more from our community and how they're using it.

We appreciate the community's willingness to eagerly adopt the App Router and provide feedback. There's been a number of bug reports we're investigating and we're thankful for the minimal reproductions you have created to help isolate issues and verify fixes.

Since 13.4, we've already patched a number of high impact bugs around stability that are available in the latest patch release (`13.4.7`). We will be continuing to focus on performance and stability with high intensity.

Improving Developer Education

While we believe the new features of the App Router and modern React are powerful, they also require additional education and documentation to help teach these new concepts.

Next.js features

We've been working over the past year to re-write the Next.js documentation from scratch. This work is now live on nextjs.org/docs. We'd like to highlight some important pieces:

- **Pages and App toggles:** You can switch between learning the Pages Router or App Router documentation using the button on the left side of the documentation. Further, you can filter search results

based on your router choice.

- **Improved content and information architecture:** Almost every single page of the App Router documentation has been refreshed, including more clear structure and cohesiveness between pages, and hundreds of new illustrations to visually explain how Next.js works.
- **More to come**: We have more work to do here. The Developer Experience team at Vercel is working hard to provide additional learning resources (including an updated course on `/learn` teaching the App Router) and real world codebase examples (including a rewrite of Next.js Commerce ↗).

We'll be releasing new content in the documentation, on Twitter ↗, YouTube ↗, and more.

New React features

We've also heard your feedback about the education around new React features that are available in the Next.js App Router.

- **Server Components**: It's important to note that features like Server Components and conventions like the `"use client"` directive ↗ are not Next.js specific, but a larger part of the React ecosystem.

  Our team, our partners at Meta, and other independent contributors are working to provide more education around these topics. It's early days for these concepts, but we're confident in the React ecosystem and continued education ↗.

- **Client Components**: With the recent conversation around Server Components, it's important to note the client components are *not* a de-optimization. The client is a valid part of the React model and is not going away.

  You can think of client components as the existing Next.js ecosystem today, where your favorite libraries and tools continue to work. For example, a common question we've seen is whether `"use client"` needs to be added to every single file to make it a client component. This is not necessary, but we understand these concepts are new and will take time to learn. You only need to mark the top level boundary where you code moves between the server to the client. This architecture allows you to interweave server and client components together ↗.

- **Growing third-party ecosystem**: In addition to education, the ecosystem around React's newer features is still growing. For example, Panda CSS ↗, a CSS-in-JS library from the makers of Chakra UI, just announced support for React Server Components.

- **Server Actions (Alpha)**: Server Actions enable server-side data mutations, reduced client-side JavaScript, and progressively enhanced forms. We do not recommend using Server Actions in production yet. We appreciate early feedback from alpha testers helping us shape the future of this feature.

Thank you

We're thankful many of you have chosen to learn and build with Next.js.

Our focus on performance, stability, and developer experience will be reflected in the upcoming releases of Next.js. We want using Next.js to be delightful—and to make you (and your team) more productive.

As always, we greatly appreciate your feedback. If you are experiencing any issues with Next.js, please open an issue ↗, or start a new discussion ↗, and we will investigate.