ADR: Frontend Tech - Metaframework

Status	COMPLETE
Impact	HIGH
Driver	@gw
Approver	
Contributors	
Informed	Engineering
Due date	Jan 26, 2024
Resources	ADR: Frontend Tech

Relevant data

Why a metaframework? A metaframework helps us build in a more unified manner, such as how routing should be done, while having support for SSR and other patterns. It also allows us to make complex multipart mutations hidden from the client, such as editing a user.

All metaframeworks suffer from major overlap with a graphql document cache, since the graphql cache wants to be responsible for querying & mutations to ensure their internal caches are kept updated. Metaframeworks can still fill the cache during SSR with SSR Exchange. At the same time, State Management | Remix could be skipped entirely on the client, but with its own set of tradeoffs around using APIs to build a PWA.

Options

Next

The "king" of React metaframeworks in popularity State of JavaScript 2022: Rendering Frameworks. Their deep relationship with React's core has allowed them the opportunity to develop "ahead of the curve".

Pros

- No other framework has this level of "integration" ("use server' & "use client")
- Most popular metaframework, so lots of resources
  - Even part of React's list of "recommended": Start a New React Project – React
- Has already been utilized in eTicket
- Lots of third party libraries already integrate with it
  - Fuse: TypeScript API Framework

Cons

- Has fought against the rest of the community when it comes to tooling
  - Cannot deploy cleanly to non-Vercel providers without utilizing OpenNext (even SST relies on OpenNext and run it)
  - Utilizes their significantly slower Turbopack
- Next 14 has been fairly divisive among previously enthusiastic users
  - Significantly worse DX from how slow it is

Remix

Up until very recently Remix was not intended to be used solely as a CSR (client side rendered/routed) only framework, but rather a combination of CSR & SSR. Remix is backed by Shopify after “buying the team”. Remix vs Next.js

Pros

- One of the earliest metaframeworks
- Significantly more agnostic among cloud providers
- SST support, although possibly not for the new Vite based
- Has recently switched to aligning with the larger community by utilizing Vite
- Good support for utilizing newer features sooner with Future Flags | Remix
- Corporate backer
- Not relying on React canary builds
- Not relying on patching fetch

Cons

- Vite is still labeled “unstable”
- SPA mode is labeled “unstable”
- Historically react-router has been weak on type safety making tanstack router overall better

Vite based

This encompasses utilizing libraries built on top of Vite itself such as Vike. This allows more customization and less buy into a specific metaframework. It also offers choosing alternatives that can offer better DX, such as tanstack router.

Pros

- “Raw” Vite makes it easier to keep up with Vite
- Independent of any framework
- Aside from Next all other frameworks are utilizing Vite anyway

Cons

- No corporate backer like Next & Remix
- Tend to require more setup than alternatives
- Often small teams or single developer working on them
- Less resources for utilizing them
- No specific SST integration, making deployment more complex

Astro

Aside from Vite based solutions, the only framework agnostic solution out there. It also tops interest, retention, and lowest hate of it.

Pros

- One of the fastest options out there, Movies app in 7 frameworks - which is fastest and why?
- One of the few with integrated view transitions support
- Can run multiple frontend frameworks side-by-side allowing easier transitions to or from
- Early adopter of Vite, so it just works already well with it
- SST integration

Cons

- Geared more toward MPA first and foremost
    - Additional complexity of building CSR/SPA focused apps that are highly dynamic

- Multiframework isn’t free and they cannot share state trivially
- Sponsored, but not “ran” by someone who bank rolls it

Recommendation

`Remix` is the safest choice out of these for a more React centric approach. It allows to start as a fully CSR application without having to start server side. Additionally, it avoids Vercel’s Next provider specific functionality or relying on a solution like OpenNext. However, because Vite & SPA mode are both unstable, some plugins & tools have yet to be updated for it, this includes SST that still relies on the old esbuild version of building Remix.

TLDR

`Remix`

Decision

For now we’ll be punting on this decision and reevaluate the need for it later. We’re going to focus on improving the React written today first and as some of the solutions out there mature more we can retake a look.