JavaScript, Node.js apps and Exponential

Moving between Exponential 6 and JavaScript applications - Next.js and other static or server-rendered front ends, headless setups, Markdown and MDX content in a repository - including using Exponential as the content...

Source file: `doc/eng-us/install/migration/javascript-apps.md` Last changed 2026-11-04
https://doc.exponential.earth/exphelp/viewer/install/migration/javascript-apps.md

A JavaScript site is a front end; its content lives somewhere else: in Markdown or MDX files in the repository, in a headless CMS, in a database behind an API, or inside the code. So "migrating a Next.js site" means two decisions: **where the content goes** (into Exponential), and **what renders the site** (Exponential's templates, or the existing front end reading Exponential).

Migration hub · Checklist · Mapping · Tools

Contents

- 1. Where the content of a JavaScript site lives
- 2. Content into Exponential
- 3. Keeping the JavaScript front end: Exponential as its content source
- 4. Addresses and redirects
- 5. From Exponential to a JavaScript front end
- 6. Checklist

1. Where the content of a JavaScript site lives

Content source	How to read it
Markdown or MDX files (blogs, documentation sites)	the files themselves: front matter (YAML) plus body
A headless CMS (Contentful, others)	that CMS's export or API; for Contentful see Contentful
JSON or YAML data files	the files
A database behind the app's own API	the API, or the database directly (other applications)
Hard-coded in components	the rendered HTML (crawl, static HTML)

Next.js stores no content of its own. According to its documentation, `output: 'export'` makes `next build` write static HTML, CSS and JavaScript to `out/` (`next export` was removed in version 14). That output is a complete crawl target for the static HTML method when the sources are hard to read.

2. Content into Exponential

1. **Collect the sources**: list the content directories, data files and APIs the app reads at build or request time (look at `getStaticProps`/`generateStaticParams`, loaders, or the data layer of your framework).

2. **Convert Markdown/MDX**: in Node.js, parse the front matter and render Markdown to HTML with the library the site already uses, so the result matches what visitors saw. MDX components (`<Callout>`, `<Video>`) have no HTML equivalent: map each to an Exponential custom tag or embed, or render it to plain HTML.
3. **Write JSON rows**: remote id from the file path or slug (`md-blog/2024/hello`), title, date, tags, author, the HTML, the route.
4. **Import** with the script skeleton on the server, or xrowextract's importer with `--map`. Media from `public/` become image/file objects.
5. **Templates**: build TPL templates, or keep the front end (3).

3. Keeping the JavaScript front end: Exponential as its content source

Exponential can be the headless content source of an existing front end; editors work in the Exponential admin, the front end fetches content at build or request time.

Interface	Use	Reference
REST v1/v2 (`ezprestapi`)	read nodes, children and fields as JSON: GET `/api/ezp/v2/content/node/<id>`, `.../list/offset/<n>/limit/<n>`, `?ResponseGroups=Fields`, `?OutputFormat=xhtml` for the rendered body	REST API v1 and v2
Personal API keys	a build server reads with `Authorization: Bearer expk_...`, limited to the key's scopes and the owner's policies	Personal API keys
OAuth2	an app that acts for signed-in users	the REST guide, section 3.1
`expservices`	the admin's functions as named HTTP services (content, users, media, tags, layouts ...)	Remote services and apps

Writes over REST are limited (create and delete at v2; update answers 501 for now), so bulk imports are done on the server, not from the front end.

4. Addresses and redirects

- Keep the routes: give each node the alias of its old route, so the front end (or Exponential's own templates) can serve the same paths.
- According to the Next.js documentation, a static export does **not** support `redirects`, `rewrites` or `headers` from `next.config.js`, nor Route Handlers that use the request, ISR, Server Actions or default image optimization: redirects of a statically exported site belong to the host or web server.
- When Exponential renders the site, the redirects are URL aliases and wildcards (checklist, 4).

5. From Exponential to a JavaScript front end

- **Keep Exponential as the content source** and build the new front end against its REST API (3).

- **Or move the content out**: export with xrowextract to JSON (leaving Exponential), render XmlText to HTML or Markdown, and write the files or the headless CMS of your choice.
- Image aliases (scaled renditions) are produced by Exponential; a static site needs its own image pipeline.

6. Checklist

☐ Every content source of the app listed (files, data, APIs, hard-coded).

☐ Markdown rendered with the site's own renderer; MDX components mapped.

☐ Imported with remote ids from paths or slugs; media imported.

☐ Decision made: Exponential templates, or the front end reading Exponential (API key scopes set).

☐ Routes preserved or redirected on the host.

Sources

- Next.js static exports: https://nextjs.org/docs/app/guides/static-exports

> **Become an Exponential Partner**
>
> Partners share their information, provide funding, contribute back and develop the products with us, and that is how Exponential grows. A partner website and dashboard are planned for mid-2027: please become a partner today to speed up our growth rate and land new features faster.
>
> Partners: start here | How to apply

This page is part of the Exponential documentation and is free documentation under the GNU Free Documentation License 1.3 or any later version (the license of this documentation).

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.