Static HTML Export

▶ **Examples**

`next export` allows you to export your Next.js application to static HTML, which can be run standalone without the need of a Node.js server. It is recommended to only use `next export` if you don't need any of the unsupported features requiring a server.

If you're looking to build a hybrid site where only *some* pages are prerendered to static HTML, Next.js already does that automatically. Learn more about Automatic Static Optimization and Incremental Static Regeneration.

`next export`

Update your build script in `package.json` to use `next export`:

```
"scripts": {
  "build": "next build && next export"
}
```

Running `npm run build` will generate an `out` directory.

`next export` builds an HTML version of your app. During `next build`, `getStaticProps` and `getStaticPaths` will generate an HTML file for each page in your `pages` directory (or more for dynamic routes). Then, `next export` will copy the already exported files into the correct directory. `getInitialProps` will generate the HTML files during `next export` instead of `next build`.

For more advanced scenarios, you can define a parameter called `exportPathMap` in your `next.config.js` file to configure exactly which pages will be generated.

Supported Features

The majority of core Next.js features needed to build a static site are supported, including:

- Dynamic Routes when using `getStaticPaths`
- Prefetching with `next/link`
- Preloading JavaScript
- Dynamic Imports
- Any styling options (e.g. CSS Modules, styled-jsx)
- Client-side data fetching
- `getStaticProps`
- `getStaticPaths`
- Image Optimization using a custom loader

Unsupported Features

Features that require a Node.js server, or dynamic logic that cannot be computed during the build process, are not supported:

- Image Optimization (default loader)
- Internationalized Routing
- API Routes

- Rewrites
- Redirects
- Headers
- Middleware
- Incremental Static Regeneration
- `fallback: true`
- `getServerSideProps`

`getInitialProps`

It's possible to use the `getInitialProps` API instead of `getStaticProps`, but it comes with a few caveats:

- `getInitialProps` cannot be used alongside `getStaticProps` or `getStaticPaths` on any given page. If you have dynamic routes, instead of using `getStaticPaths` you'll need to configure the `exportPathMap` parameter in your `next.config.js` file to let the exporter know which HTML files it should output.
- When `getInitialProps` is called during export, the `req` and `res` fields of its `context` parameter will be empty objects, since during export there is no server running.
- `getInitialProps` **will be called on every client-side navigation**, if you'd like to only fetch data at build-time, switch to `getStaticProps`.
- `getInitialProps` should fetch from an API and cannot use Node.js-specific libraries or the file system like `getStaticProps` can.

We recommend migrating towards `getStaticProps` over `getInitialProps` whenever possible.