Static Exports

Next.js enables starting as a static site or Single-Page Application (SPA), then later optionally upgrading to use features that require a server.

When running `next build`, Next.js generates an HTML file per route. By breaking a strict SPA into individual HTML files, Next.js can avoid loading unnecessary JavaScript code on the client-side, reducing the bundle size and enabling faster page loads.

Since Next.js supports this static export, it can be deployed and hosted on any web server that can serve HTML/CSS/JS static assets.

Note: We recommend using the App Router for enhanced static export support.

Configuration

To enable a static export, change the output mode inside `next.config.js`:

next.config.js

```
/**
 * @type {import('next').NextConfig}
 */
const nextConfig = {
  output: 'export',
  // Optional: Add a trailing slash to all paths `/about` -> `/about/`
  // trailingSlash: true,
  // Optional: Change the output directory `out` -> `dist`
  // distDir: 'dist',
};

module.exports = nextConfig;
```

After running `next build`, Next.js will produce an `out` folder which contains the HTML/CSS/JS assets for your application.

You can utilize `getStaticProps` and `getStaticPaths` to generate an HTML file for each page in your `pages` directory (or more for dynamic routes).

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

Image Optimization through `next/image` can be used with a static export by defining a custom image loader in `next.config.js`. For example, you can optimize images with a service like Cloudinary:

next.config.js

```
/** @type {import('next').NextConfig} */
const nextConfig = {
  output: 'export',
  images: {
    loader: 'custom',
    loaderFile: './app/image.ts',
  },
};

module.exports = nextConfig;
```

This custom loader will define how to fetch images from a remote source. For example, the following loader will construct the URL for Cloudinary:

app/image.ts

```ts
export default function cloudinaryLoader({
  src,
  width,
  quality,
}: {
  src: string;
  width: number;
  quality?: number;
}) {
  const params = ['f_auto', 'c_limit', `w_${width}`, `q_${quality || 'auto'}`];
  return `https://res.cloudinary.com/demo/image/upload/${params.join(
    ',',
  )}${src}`;
}
```

You can then use `next/image` in your application, defining relative paths to the image in Cloudinary:

app/page.tsx

```tsx
import Image from 'next/image';

export default function Page() {
  return <Image alt="turtles" src="/turtles.jpg" width={300} height={300} />;
}
```

Unsupported Features

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

- Internationalized Routing
- API Routes
- Rewrites
- Redirects
- Headers
- Middleware
- Incremental Static Regeneration
- `getStaticPaths` with `fallback: true`

- `getStaticPaths` with `fallback: 'blocking'`
- `getServerSideProps`
- Image Optimization (default loader)

Deploying

With a static export, Next.js can be deployed and hosted on any web server that can serve HTML/CSS/JS static assets.

When running `next build`, Next.js generates the static export into the `out` folder. Using `next export` is no longer needed. For example, let's say you have the following routes:

- `/`
- `/blog/[id]`

After running `next build`, Next.js will generate the following files:

- `/out/index.html`
- `/out/404.html`
- `/out/blog/post-1.html`
- `/out/blog/post-2.html`

If you are using a static host like Nginx, you can configure rewrites from incoming requests to the correct files:

nginx.conf

```
server {
  listen 80;
  server_name acme.com;

  root /var/www;

  location / {
      try_files /out/index.html =404;
  }

  location /blog/ {
      rewrite ^/blog/(.*)$ /out/blog/$1.html break;
  }

```

```
    error_page 404 /out/404.html;
    location = /404.html {
        internal;
    }
}
```

Version History

Version	Changes
`v13.4.0`	App Router (Stable) adds enhanced static export support, including using React Server Components and Route Handlers
`v13.3.0`	`next export` is deprecated and replaced with `"output": "export"`