Version 13

Upgrading from 12 to 13

To update to Next.js version 13, run the following command using your preferred package manager:

Terminal

```
npm i next@latest react@latest react-dom@latest eslint-config-next@latest
# or
yarn add next@latest react@latest react-dom@latest eslint-config-next@latest
# or
pnpm up next react react-dom eslint-config-next --latest
```

v13 Summary

- The Supported Browsers have been changed to drop Internet Explorer and target modern browsers.
- The minimum Node.js version has been bumped from 12.22.0 to 14.18.0, since 12.x has reached end-of-life.
- The minimum React version has been bumped from 17.0.2 to 18.2.0.
- The `swcMinify` configuration property was changed from `false` to `true`. See Next.js Compiler for more info.
- The `next/image` import was renamed to `next/legacy/image`. The `next/future/image` import was renamed to `next/image`. A codemod is available to safely and automatically rename your imports.
- The `next/link` child can no longer be `<a>`. Add the `legacyBehavior` prop to use the legacy behavior or remove the `<a>` to upgrade. A codemod is available to automatically upgrade your code.
- The `target` configuration property has been removed and superseded by Output File Tracing.

Migrating shared features

Next.js 13 introduces a new `app` directory with new features and conventions. However, upgrading to Next.js 13 does **not** require using the new `app` directory.

You can continue using `pages` with new features that work in both directories, such as the updated Image component, Link component, Script component, and Font optimization.

`<Image/>` Component

Next.js 12 introduced many improvements to the Image Component with a temporary import: `next/future/image`. These improvements included less client-side JavaScript, easier ways to extend and style images, better accessibility, and native browser lazy loading.

Starting in Next.js 13, this new behavior is now the default for `next/image`.

There are two codemods to help you migrate to the new Image Component:

- next-image-to-legacy-image: This codemod will safely and automatically rename `next/image` imports to `next/legacy/image` to maintain the same behavior as Next.js 12. We recommend running this codemod to quickly update to Next.js 13 automatically.
- next-image-experimental: After running the previous codemod, you can optionally run this experimental codemod to upgrade `next/legacy/image` to the new `next/image`, which will remove unused props and add inline styles. Please note this codemod is experimental and only covers static usage (such as `<Image src={img} layout="responsive" />`) but not dynamic usage (such as `<Image {...props} />`).

Alternatively, you can manually update by following the migration guide and also see the legacy comparison.

`<Link>` Component

The `<Link>` Component no longer requires manually adding an `<a>` tag as a child. This behavior was added as an experimental option in version 12.2↗ and is now the default. In Next.js 13, `<Link>` always renders `<a>` and allows you to forward props to the underlying tag.

For example:

```
import Link from 'next/link'

// Next.js 12: `<a>` has to be nested otherwise it's excluded
```

```
<Link href="/about">
  <a>About</a>
</Link>

// Next.js 13: `<Link>` always renders `<a>` under the hood
<Link href="/about">
  About
</Link>
```

To upgrade your links to Next.js 13, you can use the `new-link` codemod.

`<Script>` Component

The behavior of `next/script` has been updated to support both `pages` and `app`. If incrementally adopting `app`, read the upgrade guide.

Font Optimization

Previously, Next.js helped you optimize fonts by inlining font CSS. Version 13 introduces the new `next/font` module which gives you the ability to customize your font loading experience while still ensuring great performance and privacy.

See Optimizing Fonts to learn how to use `next/font`.