Performance as a default with Next.js

- What will you learn?
- How is Next.js different from React?
- Setting up
- What's next?

Next.js is an opinionated React framework with a number of performance optimizations baked in. The main idea behind the framework is to ensure applications start and remain as performant as possible by having these capabilities included by default.

This introduction will briefly cover many features provided by the framework at a high level. The other guides in this collection will explore the features in more detail.

Chrome is collaborating with Next.js to improve the framework for any developer looking to build a fast, server-rendered React application. A number of newer optimizations were recently added such as module/nomodule support and an improved granular chunking strategy.

What will you learn?

Although Next.js provides a number of performance optimizations by default, these guides aim to explain them in more detail and show you how you can use them to build a fast and performant experience.

This collection assumes that you have a basic knowledge of React. If not, check out React's Getting Started page.

There are many optimizations that can be added to React sites in general that would also work for applications built with Next.js. These will not be covered since the focus is on what Next.js specifically provides. To learn more about general React optimizations, check out our React

collection.

How is Next.js different from React?

React is a library that makes it easier to build user interfaces using a component-based approach. Although powerful, React is specifically a UI library. Many developers include additional tooling such as a module bundler (webpack for example) and a transpiler (Babel for example) to have a complete build toolchain.

In the React collection, we took the approach of using Create React App (CRA) to spin up React apps quickly. CRA takes the hassle out of setting up a React application by providing a complete build toolchain with a single command.

Although there are a few default optimizations baked into CRA, the tool aims to provide a simple and straightforward setup. The choice is given to developers to decide whether to eject and modify the configurations themselves.

Next.js, which can also be used to create a new React application, takes a different approach. It immediately provides a number of common optimizations that many developers would like to have but find difficult to set up, such as:

- Server-side rendering
- Automatic code-splitting
- Route prefetching
- File-system routing
- CSS-in-JS styling (`styled-jsx`)

Setting up

To create a new Next.js application, run the following command:

```
npx create-next-app new-app
```

npx is a package runner that is installed automatically with npm 5.2.0 or later. It simplifies quite a few processes involved with managing packages including running CLI commands (like `create-next-app`) without having to install them globally on your machine.

Then navigate to the directory and start the development server:

```
cd new-app
npm run dev
```

You can also add Next.js to an existing React application. Check out Manual Setup to learn how.

The following embed shows the directory structure of a new Next.js app.

https://glitch.com/edit/#!/new-next-app

Notice that a `pages/` directory is created with a single file: `index.jsx`. Next.js follows a file-system routing approach, where every page within this directory is served as a separate route. Creating a new file in this directory, such as `about.js`, will automatically create a new route (`/about`).

Components can also be created and used like any other React application. A `components/` directory has already been created with a single component, `nav.js`, which is already imported in `index.js`. By default, every import used in Next.js is only fetched when that page is loaded, providing the benefits of **automated code-splitting**.

Moreover, every initial page load in Next.js is **server-side rendered**. If you open the Network panel in DevTools, you can see the initial request for the document returns a fully server-rendered page.

> *The Preview tab of the Network panel shows that Next.js returns visually complete HTML when a page is requested.*
>
> The Preview tab of the Network panel shows that Next.js returns visually complete HTML when a page is requested.

Check out Server Rendering to learn how server-side rendering often results in a better experience for users.

These are only a few of the many features provided by Next.js automatically. Many are customizable and can be modified for different use cases.

What's next?

Pun intended □□

Every other guide in this collection will explore a specific Next.js feature in detail:

- Route prefetching to speed up page navigations
- Serving hybrid and AMP-only pages for faster loading from search engines

- Code-splitting components with dynamic imports to reduce JavaScript footprints

---

□□□ #3
□ □□□ □□□ 2026-10-31 23:10:54 UTC
□ cocobook-sync □□□ 2026-11-04 17:39:50 UTC