Getting Started

Welcome to the Next.js documentation!

If you're new to Next.js, we recommend starting with the learn course.

The interactive course with quizzes will guide you through everything you need to know to use Next.js.

If you have questions about anything related to Next.js, you're always welcome to ask our community on GitHub Discussions.

System Requirements

- Node.js 12.22.0 or later
- MacOS, Windows (including WSL), and Linux are supported

Automatic Setup

We recommend creating a new Next.js app using `create-next-app`, which sets up everything automatically for you. To create a project, run:

```
npx create-next-app@latest
# or
yarn create next-app
# or
pnpm create next-app
```

If you want to start with a TypeScript project you can use the `--typescript` flag:

```
npx create-next-app@latest --typescript
# or
yarn create next-app --typescript
# or
pnpm create next-app -- --typescript
```

After the installation is complete:

- Run `npm run dev` or `yarn dev` or `pnpm dev` to start the development server on `http://localhost:3000`
- Visit `http://localhost:3000` to view your application
- Edit `pages/index.js` and see the updated result in your browser

For more information on how to use `create-next-app`, you can review the `create-next-app` documentation.

Manual Setup

Install `next`, `react` and `react-dom` in your project:

```
npm install next react react-dom
# or
yarn add next react react-dom
# or
pnpm add next react react-dom
```

Open `package.json` and add the following `scripts`:

```
"scripts": {
  "dev": "next dev",
  "build": "next build",
  "start": "next start",
  "lint": "next lint"
}
```

These scripts refer to the different stages of developing an application:

- `dev` - Runs `next dev` to start Next.js in development mode
- `build` - Runs `next build` to build the application for production usage
- `start` - Runs `next start` to start a Next.js production server
- `lint` - Runs `next lint` to set up Next.js' built-in ESLint configuration

Create two directories `pages` and `public` at the root of your application:

- `pages` - Associated with a route based on their file name. For example `pages/about.js` is mapped to `/about`
- `public` - Stores static assets such as images, fonts, etc. Files inside `public` directory can then be referenced by your code starting from the base URL (`/`).

Next.js is built around the concept of pages. A page is a React Component exported from a `.js`, `.jsx`, `.ts`, or `.tsx` file in the `pages` directory. You can even add dynamic route parameters with the filename.

Inside the `pages` directory add the `index.js` file to get started. This is the page that is rendered when the user visits the root of your application

Populate `pages/index.js` with the following contents:

```
function HomePage() {
  return <div>Welcome to Next.js!</div>
}

export default HomePage
```

After the set up is complete:

- Run `npm run dev` or `yarn dev` or `pnpm dev` to start the development server on `http://localhost:3000`
- Visit `http://localhost:3000` to view your application
- Edit `pages/index.js` and see the updated result in your browser

So far, we get:

- Automatic compilation and bundling
- React Fast Refresh
- Static generation and server-side rendering of `pages/`
- Static file serving through `public/` which is mapped to the base URL (`/`)

In addition, any Next.js application is ready for production from the start. Read more in our Deployment documentation.

Related

For more information on what to do next, we recommend the following sections:

Pages: Learn more about what pages are in Next.js.

CSS Support: Built-in CSS support to add custom styles to your app.

CLI: Learn more about the Next.js CLI.