

**HERE'S THE**

**LOWDOWN BY**

**HAPPYCODER**

09

|

2024

Exploring NextJS

-- The ultimate roadmap to mastery! --

Jonathan Spurling

PROJECT SPACE TRAINING

[Next JS Training 3](#_Toc177294406)

[What is Next.JS 3](#_Toc177294407)

[Why Use It? 3](#_Toc177294408)

[Creating a First Next.JS App 4](#_Toc177294409)

[Required Software 4](#_Toc177294410)

[Setting Up a Next.js Project 5](#_Toc177294411)

[React vs. Next.js 6](#_Toc177294412)

[1. Content in Page Source: 6](#_Toc177294413)

[2. Rendering Approach: 6](#_Toc177294414)

[3. Client-Side vs. Full-Stack: 6](#_Toc177294415)

[Adding a New Page in Next.js 7](#_Toc177294416)

[Adding a New Page: 7](#_Toc177294417)

[Define the Page Content: 7](#_Toc177294418)

[View the New Page: 8](#_Toc177294419)

[Server-Side Rendering: 8](#_Toc177294420)

[Understanding the Different Routers in Next.js 8](#_Toc177294421)

[The Pages Router 8](#_Toc177294422)

[The App Router 9](#_Toc177294423)

[Making a Decision: Pages Router or App Router? 10](#_Toc177294424)

# Next JS Training

## What is Next.JS

Next.js is a popular open-source framework built on top of React.js. It provides a set of tools and conventions that make it easier to build modern web applications with React. Here’s a brief overview of what it is and why you might use it:

**Definition**

Next.js is a React framework that enables developers to build server rendered React applications with ease. It supports features like:

* Server-Side Rendering (SSR): Pages can be rendered on the server before being sent to the client, improving initial load performance and SEO.
* Static Site Generation (SSG): Pages can be pre-rendered at build time, allowing for fast load times and better performance.
* Incremental Static Regeneration (ISR): Allows updating static content incrementally without rebuilding the whole site.
* API Routes: Enables you to create backend API routes within the same codebase, simplifying development by keeping front-end and back-end code together.
* Built-in CSS and Sass Support: Provides support for importing CSS and Sass files directly.
* File-Based Routing: Pages are automatically routed based on the file structure in the pages directory.

## Why Use It?

1. **Performance Optimization:** Next.js optimizes performance with server-side rendering, static site generation, and automatic code splitting. This helps improve load times and overall user experience.
2. **SEO Benefits:** Server-side rendering and static generation make it easier to optimize pages for search engines since content is rendered and available to crawlers when the page loads.
3. **Developer Experience:** Next.js offers a great development experience with features like hot reloading, built-in CSS support, and a well-structured project setup.
4. **Flexibility:** You can choose between server-side rendering, static site generation, or even a hybrid approach for different pages within the same application, depending on your needs.
5. **API Routes:** It simplifies full-stack development by allowing you to build API endpoints within the same project, reducing the need for a separate backend server.
6. **Community and Ecosystem:** Next.js has a strong community and a rich ecosystem of plugins and integrations, making it easier to extend and customize your application.

In summary, Next.js is used to build performant, scalable web applications with React, offering various rendering options and developer conveniences to streamline the development process.

# Creating a First Next.JS App

To create a Next.js app, you'll need a few key pieces of software and tools.

## Required Software

**1. Node.js and npm**

**Node.js:** Next.js is built on top of Node.js, so you need to have it installed on your system. Node.js is a JavaScript runtime that allows you to run JavaScript code on the server side.

**npm (Node Package Manager)**: These are package managers that help you manage the dependencies for your Next.js application. npm comes bundled with Node.js, but you can also use Yarn if you prefer.

You can install Node.js (which includes npm) from the [official Node.js website](https://nodejs.org/en). Yarn can be installed separately from Yarn's website.

**2. Text Editor or IDE**

While you can technically use any text editor, some are particularly well-suited for JavaScript and React development. Popular choices include:

* **Visual Studio Code (VS Code):** A widely used code editor that offers great support for JavaScript, React, and Next.js through extensions. It also includes features like integrated terminal, Git integration, and debugging tools.
* **WebStorm:** A powerful IDE by JetBrains with excellent support for JavaScript and frameworks like React and Next.js.
* **Sublime Text** or **Atom**: Both are lightweight editors that can be configured with plugins for React development.

**3. Git (optional, but recommended)**

**Git:** Version control system for tracking changes in your codebase. Git is highly recommended for managing and collaborating on projects. You can download Git from the [official Git website](https://git-scm.com/).

**4. Terminal** or **Command Line Interface**

You'll use the terminal to run commands related to Node.js, npm, and Next.js. On macOS and Linux, you can use the built-in Terminal. On Windows, you can use Command Prompt, PowerShell, or the more advanced Windows Terminal.

**5. Browser**

**Web Browser:** You'll need a web browser to view and test your application. Modern browsers like **Google Chrome**, **Mozilla Firefox**, or **Microsoft Edge** are recommended.

## Setting Up a Next.js Project

Once you have these tools installed, setting up a new Next.js project is straightforward.

**Here are the typical steps:**

1. Open your terminal and navigate to the directory where you want to create your project.
2. Run the following command in the folder you want to create a new Next.js app using create-next-app:

npx create-next-app@latest

1. Then you will be asked a series of questions:
   1. What is your project named? (first-app)
   2. Would you like to use Typescript? (No)
   3. Would you like to use ESLint? (Yes)
   4. Would you like to use Tailwind CSS? (No)
   5. Would you like to use `src/` directory? (No)
   6. Would you like to use App Router? (recommended) (Yes)
   7. Would you like to customize the default import alias (@/\*)? (No)

Then your app will start to be created.

1. Navigate to your project directory:

cd first-app

1. Start your project by running:

npm run dev

**NOTE:** Your project will now be accessible from <http://localhost:3000>.

## React vs. Next.js

### Content in Page Source:

**Standard React App:**

* When you view the page source in your browser, you’ll mostly see an empty page with some script tags. The actual content visible on the screen is not included in the HTML sent from the server.
* This is because React operates entirely on the client side. It sends a barebones HTML page with a minimal setup, and then JavaScript runs in the browser to dynamically generate and manipulate the content.

**Next.js App:**

* When you view the page source in your browser, you’ll see the actual page content directly in the HTML. This is because Next.js renders the page on the server before sending it to the client.
* This means the content is included in the HTML that comes from the server, so it's visible even if you inspect the page source.

### Rendering Approach:

**Standard React App:**

* React operates primarily on the client side. It means that the page is initially loaded with minimal HTML, and then React dynamically builds and updates the page content using JavaScript in the browser.
* This approach allows for dynamic updates and interactive features without needing to reload the page from the server.

**Next.js App:**

* Next.js supports both server-side rendering (SSR) and static site generation (SSG). This means that Next.js can generate the HTML content on the server before sending it to the client.
* This approach combines client-side and server-side rendering, providing the benefits of faster initial page loads and better SEO (since search engines can see the content directly).

### Client-Side vs. Full-Stack:

**Standard React App:**

* It's mainly a client-side application. It runs entirely in the browser, and any data fetching or other backend interactions need to be handled separately (e.g., through API calls).

**Next.js App:**

* It’s a full-stack framework. This means it can handle both client-side and server-side code within the same project. You can use Next.js to render pages on the server and also build API routes for backend functionality.

In summary, Next.js provides additional features over standard React, particularly in terms of rendering and SEO, by combining server-side and client-side capabilities. This allows Next.js to serve fully rendered HTML content from the server, which can improve performance and SEO, whereas standard React relies on client-side JavaScript to build the content dynamically in the browser.

## Adding a New Page in Next.js

1. Understanding Routes and File System-Based Routing:

* **Routes** are essentially URLs that map to different parts of your application. For example, localhost:3000/awesome is a route.
* **File System-Based Routing** in Next.js means that the file structure in your project automatically determines the routes for your application. You don’t need to explicitly configure routes; just creating files and folders in a specific way sets up the routes.

### Adding a New Page:

* **Navigate to the app Folder:** This is the main folder where you set up your routes in Next.js.
* **Create a New Folder:** For instance, create a folder named awesome. The name of this folder will be part of the route’s URL path.
* **Add a page.js File:** Inside the awesome folder, create a file named page.js. In Next.js, this file will automatically be recognized as the component for the /awesome route.

### Define the Page Content:

In **page.js**, export a React component function. This function represents the content of the page.

export default function AwesomePage() {

return (

<main>

<h1>NextJS Is Awesome!</h1>

</main>

);

}

This code will render a heading with the text "NextJS Is Awesome!" when you visit the **/awesome** route.

### View the New Page:

* After saving your changes, go to your Next.js application in your browser and navigate to localhost:3000/awesome.
* You should see the content you defined in page.js.

### Server-Side Rendering:

* The content you added is rendered on the server before being sent to the client. If you inspect the page source in your browser, you'll see the HTML content directly in the source code.

**Summary**

* **Routes** are created by organizing files and folders in your project.
* **File System-Based Routing** means that the file structure determines the routes.
* By creating a folder and a page.js file, you automatically set up a new route.
* Pages are React components that define the content and structure of that route.

This approach makes it easy to add new pages and manage routes in Next.js without needing to manually configure routing settings.

## Understanding the Different Routers in Next.js

1. The Two Router Options: Pages Router vs. App Router

In Next.js, there are two ways to build your application, known as **Pages Router** and **App Router**. These approaches are used for **routing** – determining how different pages in your app are accessed via URLs.

* **Pages Router**: The older, more stable way of routing, which has been around for years. It's well-suited for projects that follow the traditional Next.js setup.
* **App Router**: A newer approach introduced in **Next.js 13**. It offers modern features but may still have some bugs since it's relatively new.

**2. Core Concept: Server-Side Rendering**

Regardless of which router you choose, both methods allow you to create **full-stack applications** in Next.js, where the pages are rendered on the server before being sent to the client.

### The Pages Router

**1. Overview:**

The **Pages Router** uses a file-based routing system. It’s an established, reliable way to build **feature-rich full-stack apps** using Next.js and React.

**Example:**

* You create a file about.js in the pages directory.
* Next.js automatically sets up the route so that when you visit yourwebsite.com/about, it renders the about.js page.

**2. Key Features:**

* **File-Based Routing**: Organize your pages into folders and files, and Next.js handles the routes for you.
* **Stability**: The Pages Router has been around for years and is widely used in many production-level projects.

**Example:**

If you have an index.js file in the pages folder, it represents the homepage (/). Similarly, creating more files like contact.js or services.js will correspond to the URLs /contact and /services.

### The App Router

**1. Overview:**

The **App Router** is the **newer approach**, introduced with Next.js 13. It brings more modern features to the table, such as **React Server Components** and **Server Actions**.

**Example:**

When you use the App Router, you can have parts of your app render on the server (for efficiency), while others render in the browser (for interactivity).

**2. Key Features:**

* **Modern Features**: Unlocks advanced capabilities like **Server Components**, which can help improve performance by running part of your app on the server.
* **Future of Next.js**: Even though it’s new and might have some issues, this approach represents the direction Next.js is heading in future versions.

**Example:**

A **React Server Component** allows parts of your app (such as fetching data from an API) to happen on the server. This makes the app faster since the browser doesn’t have to wait for data to load.

### Making a Decision: Pages Router or App Router?

1. **Which One to Use:**

* If you’re working on an **existing project** or need a stable setup, use the **Pages Router**.
* If you want to explore new features and work with modern capabilities, go with the **App Router**.

1. **Learning Both:**

In this guide (or course), you'll learn about **both the Pages Router and the App Router**, allowing you to work on any Next.js project, no matter which method is used.

**Summary**

* **Pages Router**: An older, stable way to build full-stack apps using file-based routing.
* **App Router**: The newer method with modern features like React Server Components and Server Actions.
* **Server-Side Rendering**: Both routers support server-rendered pages, ensuring faster load times and better SEO.
* **Flexibility**: Learning both routers equips you to handle any Next.js project.