Next.js Cookbook

Learn how to build scalable and high-performance apps from scratch

Andrei Tazetdinov

www.bpbonline.com

First published: 2023

Published by BPB Online
WeWork
119 Marylebone Road
London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-55518-453

www.bpbonline.com

Dedicated to

My beloved wife:

Eugenie

&

My Daughter Alice

About the Author

Andrei Tazetdinov, is a highly experienced software engineer with over 16 years of experience in the industry. Currently working at IBM IX as a Senior Frontend Developer and Technical Architect, Andrei Tazetdinov has a passion for creating innovative and user-friendly applications using cutting-edge technology.

Throughout Andrei Tazetdinov's career, he has worked on numerous projects across various industries, ranging from healthcare to finance. Their experience has given them a deep understanding of the software development process, from ideation to deployment.

In addition to his professional work, Andrei Tazetdinov is also passionate about sharing their knowledge with others. He was a teacher at Samsung Coding School for several years and worked with teenagers to guide them and create their very first Android applications using Java.

With this book, Andrei Tazetdinov hopes to help aspiring developers and experienced professionals alike to become proficient in building full-stack applications using NextJS. His wealth of knowledge and experience in the field makes him the perfect guide for anyone looking to start or advance their career as a full-stack developer.

Thank you for choosing this book

About the Reviewer

Denis Bezrukov is a dedicated Frontend Developer with a rich professional background in the field of Forex trading platforms. His experience spans 4 years, during which he has cultivated a robust knowledge baseD and skill set in the development of complex applications utilizing React, Redux.

As a core contributor to the Rome tool project, a linter and formatter written in Rust, Denis has honed his ability to create modern web developers tools.

Acknowledgements

I would like to express my heartfelt gratitude to my family and friend, who have always been my pillars of strength and support. Their unwavering love and encouragement have helped me navigate through the ups and downs of life, and have been instrumental in shaping me into the person I am today.

In particular, I am immensely grateful to my daughter Alice, whose boundless curiosity and infectious energy have been a constant source of inspiration for me. Her insatiable thirst for knowledge and her relentless pursuit of excellence have reminded me of the importance of curiosity and determination, and have challenged me to strive for greatness in my own work.

I am also indebted to my colleagues, mentors, and friends, who have generously shared their time, expertise, and insights with me throughout my journey. Their wisdom, guidance, and constructive feedback have helped me refine my ideas, sharpen my skills, and broaden my horizons. I want to gratefully thank Denis Bezrukov from JetBrains for his support and help with this book review.

Finally, I would like to express my appreciation to all the readers of this book, whose interest and enthusiasm have motivated me to share my knowledge and insights with the world. It is my hope that this book will inspire and inform and that it will contribute to a deeper understanding and appreciation of the topics it explores.

Preface

As a full-stack developer, you need to master a variety of programming languages, frameworks, and tools to build robust, scalable, and user-friendly web applications. In recent years, NextJS has emerged as one of the most popular and powerful frameworks for building server-side-rendered React applications. With its intuitive API, powerful features, and vibrant community, NextJS has become the go-to choice for developers who want to create high-performance web applications with ease.

This book is designed to help you get started with NextJS and take your full-stack development skills to the next level. Whether you are a seasoned developer looking to expand your skillset or a newcomer to the world of web development, this book will provide you with the knowledge, tools, and techniques you need to build modern, dynamic web applications that meet the needs of today's users.

In this book, we will cover a wide range of topics, including:

- The basics of NextJS and its core features
- How to create and configure a NextJS application from scratch
- The power of server-side rendering and its benefits
- Best practices for styling, routing, and data fetching with NextJS
- Advanced topics such as testing, optimization, and deployment
- AWS Amplify as a hosting provider and many more topics

We will also provide you with plenty of hands-on examples, practical exercises that will help you improve your skills and confidence as a full-stack developer. By the end of this book, you will be able to create sophisticated web applications that leverage the power of NextJS and React, and you will be well on your way to a successful career in full-stack development.

So, let's get started and explore the exciting world of NextJS!

Chapter 1: Warming up with NextJS - In this chapter, readers will be introduced to NextJS and will learn how to set up their development environment. They will be guided through the installation of the necessary software and tools, and will

learn how to create a new NextJS application from scratch. Readers will also learn the basics of NextJS, including how to work with the NextJS file system, how to create pages and components, and how to use the built-in routing system. By the end of this chapter, readers will have created their first NextJS application and will be ready to dive deeper into the framework's features and capabilities.

Chapter 2: Using design patterns in NextJS - In this chapter, readers will learn how to use design patterns to optimize their NextJS application development. The chapter will cover several common design patterns, including the Singleton pattern, the Strategy pattern, and the Builder pattern. Readers will learn how these patterns can be applied to NextJS to improve the efficiency and scalability of their applications. The chapter will also provide practical examples of how to implement these patterns in NextJS, with step-by-step instructions and code snippets. By the end of this chapter, readers will have a solid understanding of how to apply design patterns in NextJS and will be able to create more robust and efficient web applications.

Chapter 3: Authorization in a glance with NextJS - In this chapter, readers will learn about authorization in NextJS and how to implement it in their applications. The chapter will cover the basics of authentication and authorization, and will provide an overview of different authentication methods that can be used in NextJS. Readers will also learn how to create a basic authorization system using NextJS's built-in authentication features. Additionally, the chapter will cover best practices for securing user data and preventing unauthorized access to sensitive information. By the end of this chapter, readers will have a solid understanding of how to implement authorization in NextJS and will be able to create secure and reliable web applications.

Chapter 4: Server-side power of NextJS - In this chapter, readers will learn about the server-side rendering features of NextJS and how to take advantage of them in their applications. The chapter will cover the benefits of server-side rendering, including improved performance, SEO, and user experience. Readers will also learn how to set up and configure server-side rendering in NextJS, as well as how to work with dynamic data and API calls in a server-side rendered application. Additionally, the chapter will cover best practices for optimizing server-side

rendered applications and handling errors. By the end of this chapter, readers will have a solid understanding of how to use server-side rendering in NextJS to create fast, dynamic, and highly scalable web applications.

Chapter 5: Using state management in NextJS - In this chapter, readers will learn about state management in NextJS and how to implement it using the popular Redux library. The chapter will cover the basics of state management, including why it's important and how it works. Readers will also learn how to set up and configure Redux in NextJS, as well as how to work with Redux actions, reducers, and stores. Additionally, the chapter will cover best practices for optimizing state management in NextJS, including how to handle asynchronous actions and how to use middleware. By the end of this chapter, readers will have a solid understanding of how to use state management with Redux in NextJS and will be able to create complex and dynamic web applications with ease.

Chapter 6: Implementing internal pages using NextJS - In this chapter, readers will learn how to create internal pages in NextJS and how to implement a basic CRUD system for managing user data. The chapter will cover the basics of creating dynamic pages in NextJS, including how to work with dynamic routes and how to pass data between pages. Readers will also learn how to set up and configure a database, and how to use NextJS's built-in API routes to handle requests and responses. Additionally, the chapter will cover best practices for optimizing internal pages in NextJS, including how to use caching and how to handle errors. By the end of this chapter, readers will have a solid understanding of how to create and manage internal pages in NextJS and will be able to build complex and powerful web applications.

Chapter 7: The superpower of E2E testing in NextJS - In this chapter, readers will learn about end-to-end (E2E) testing in NextJS and how to implement it using the Cypress and Playwright testing frameworks. The chapter will cover the basics of E2E testing, including why it's important and how it works. Readers will also learn how to set up and configure Cypress and Playwright in NextJS, as well as how to write and run tests for a NextJS application. Additionally, the chapter will cover best practices for optimizing E2E testing in NextJS, including how to handle asynchronous requests and how to use test-driven development principles. By the

end of this chapter, readers will have a solid understanding of how to use E2E testing with Cypress and Playwright in NextJS and will be able to create robust and reliable web applications.

Chapter 8: Deploying NextJS project to production - In this chapter, readers will learn how to deploy their NextJS application to production using the AWS Amplify platform. The chapter will cover the basics of deployment, including why it's important and how it works. Readers will also learn how to set up and configure their AWS Amplify account, and how to connect their NextJS application to the platform. Additionally, the chapter will cover best practices for optimizing deployment in NextJS, including how to use environment variables. By the end of this chapter, readers will have a solid understanding of how to deploy their NextJS application to production using AWS Amplify and will be able to launch their application with confidence.

Chapter 9: Mastering optimization tools for NextJS - In this chapter, readers will learn about optimization tools for NextJS and how to use them to improve the search engine optimization (SEO) of their application. The chapter will cover the basics of SEO, including why it's important and how it works. Readers will also learn how to set up and configure optimization tools and NextJS Image Optimization. Additionally, the chapter will cover best practices for optimizing SEO in NextJS, including how to use metadata and structured data, and how to improve page speed. By the end of this chapter, readers will have a solid understanding of how to use optimization tools for NextJS to improve the SEO of their application and will be able to maximize their online visibility.

Coloured Images

Please follow the link to download the
Coloured Images of the book:

https://rebrand.ly/g7kjstb

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

CHAPTER 1
Warming up with NextJS

'The beginning is always today.'

— Mary Shelley

Introduction

Greetings, a future chef in the NextJS kitchen. In this chapter, we will begin our journey into the world of sophisticated software development using the most advanced web application framework to date.

I will not delve into the intricacies of the initial setup of the environment. If you are here and ready to cook real masterpieces, then the environment necessary for installing and configuring the **nodejs** is already on your PC (or Mac, depending on preferences of course).

Structure

- Setup and run NextJS
  - Install using npm and the latest version of NodeJS
  - For the older npm versions
    - How to run the project for local development?

- How to customize Webpack
  - How to use Typescript in NextJS?
  - How to use SCSS in NextJS
- How to create a multipage app?
- How to change pages. Routing tools
  - How to change page params state without running data fetching methods
- Conclusion

Objectives

In this chapter, we will install and set up the local development environment for the easy start of the NextJS application. After you finish this chapter, you will set up and run your new NextJS application and make as many configurations as you possibly need for the first start. You will learn how to create a multi-page application with the framework. Also, you will learn how to navigate between pages and how you can manage the router properties.

Setup and run NextJS

For better performance and stability, I recommend a version of the NPM module equal to 5.2+ or higher. You can also use the older **npm-install-way** if you prefer it for some reason or you cannot install the latest version of NodeJS

Let us check several ways to install and set up your first project with NextJS:

In this book we will use yarn as an alternative to npm. Yarn provides a number of features that are not available in npm, such as faster and more reliable dependency installation, improved network performance, and better security features. You can collect it using this link https://yarnpkg.com/

Installing using npm and the latest version of NodeJS

Just type the commands into your console. Please note that in this book we will use Typescript to produce the application. For the correct setup please use the commands as follow:

```
yarn create next-app --typescript

// or if you prefer NPM

npx create-next-app@latest --typescript
```

***Figure 1.1**: Commands to install NextJS*

After the installation is complete, your project will contain all files and configurations to quick-start your new project and learn NextJS.

> create-next-app contains some other useful commands for project creation that will help you to understand how to use a framework. There is a possibility to use the GitHub URL as an example for your first application. The command should look like this:
>
> yarn create next-app -example **https://github.com/vercel/next.js/tree/canary/examples/auth0**

In this case, you will create a blank project with Auth0 possibility inside with configured API and pages to a successful login. Please check this link for more examples: **https://github.com/vercel/next.js/tree/canary/examples**

If you have already set up your project this way you can skip the next section.

For the older npm versions

For the older npm versions, use these commands to start the new project.

After successful setup and creation of the project do not forget to add the Typescript to your project in case when you have created the project by this way:

```
npm i -g create-next-app

// now we can use the application creation command

create-next-app nextjs-cookbook
```

***Figure 1.2**: Manual project creation*

To add the Typescript please follow these instructions

Create the file in the root of the project using your IDE or with the CLI command:

1. `touch tsconfig.json` for Linux/macOS
2. `echo > tsconfig.json` for Windows

NextJS will automatically do all required setup for the Typescript you will need only start your project using commands:

```
yarn dev or npm run dev
```

After that you will probably see answers about requirements for the project if you do not have them installed into your project yet eg. **@types/react, @types/node, @types/react-dom**. Just follow the instructions to complete the setup for your project

In the end, the whole setup will be complete. Please note several things:

> You will see a file with this name in your root directory next-env.d.ts**. Do not remove it or change the file body for any reason.** It is auto-generated by the Typescript compiler. Please do not make any changes in the file next-env.d.ts use the instruction below instead. Your configuration file tsconfig.json contains information about the types that you will use in your project. Just add a new type file into the include section to use it

Find below the example of the `tsconfig.json` file that you should have as a result:

```
{
  "compilerOptions": {
    "target": "es5",
    "lib": [
      "dom",
      "dom.iterable",
      "esnext"
    ],
    "allowJs": true,
    "skipLibCheck": true,
    "strict": true, // do not forget to make it TRUE for the production
    "forceConsistentCasingInFileNames": true,
    "noEmit": true,
    "esModuleInterop": true,
    "module": "esnext",
    "moduleResolution": "node",
    "resolveJsonModule": true,
    "isolatedModules": true,
    "jsx": "preserve",
    "incremental": true
  },
  "include": ["next-env.d.ts","any-new-types.d.ts","**/*.ts","**/*.tsx"],
  "exclude": ["node_modules"]
}
```

***Figure 1.3**: Code in tsconfig.json file*

Why do we ever need this feature? Let us use imagination a bit to understand this feature.

The general difference between using the ***.ts** files and ***.d.ts** files is that the second one is used to declare a type definition. Still, not much clearance because we can use both file types to declare a type. Ok, how about if we could declare the existing ***JavaScript*** function for TypeScript? For example, we have a function like this in the **printHello.js** file:

```
const printHello = (name) => `Hello ${name}`

export { printHello }
```

***Figure 1.4**: Code in printHello.ts file*

And as a declaration in file **printHello.d.ts** will be the following:

```
declare function printHello(name: string): string
```

***Figure 1.5:** Code in printHello.d.ts file*

Now we can use function **`printHello`** without any compilation errors as it is declared in the file with declarations.

If you made a setup in a modern and automatic way you can skip the next section and proceed with the first commands to start.

Next JS in general works with a page-oriented architecture so the basic element in the framework is a **page**. The page has its URL link from its creation. To create your first page please check that you have a **pages** folder in your root folder of the project. The main page (or the default page) will always be called the **`index.tsx`**. In the next sections we will figure out how to rewrite and redirect pages but the name of the default page is always **index.**

The next step is to create the file **`index.tsx`** in the pages folder with this code inside:

```
function MainPage() {
  return <div>Your cookbook with tasty recipes. Yammy</div>
}

export default MainPage
```

***Figure 1.6**: Code in index.tsx file*

Please check if you have these commands in your **package.json** file. If you do not have them then please create:

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

***Figure 1.7**: Code in package.json file*

How to run the project for local development

Just run the command in your project root: **yarn dev** or **npm run dev**

After that, you can proceed with the URL from your CLI. By default, it is **http://localhost:3000**

How to customize WebPack

Let us initialize the problem of why we would ever need to customize the WebPack for our project.

Imagine that we have different behavior in development and production mode. For example, we can use different environments and secret keys for each kind of build. To see it we need it to inject some logic into the build process or create logic in components.

We do not recommend the implementation of such logic inside the components themselves and highly recommend separating this behavior in the build process

To make changes in WebPack for NextJS let us check what changes we can do in the NextJS configuration. That can be done using the **`next.config.js`** file.

As in the case with declaration configuration, we highly recommend using separated files for the environment variables that will be changed with the development process. So please create files like **`.env.local`** or **`.env.development`** to change variables while your project is in support conditions

For example, we can create the process variable that will be used in the development or production flow. The body of your env file should look like this:

```
SECRET="NEXTJSCookBook2022"
```

***Figure 1.8**: Variable declaration in env file*

Your WebPack updates can be done in file **`next.config.js`**:

```
const nextConfig = {
  reactStrictMode: true,
  env: {
    SECRET: process.env.SECRET
  }
}

module.exports = nextConfig
```

***Figure 1.9**: Code in next.config.js file*

As you can see, we have added the process variable that can be used in our project. Please note that adding or changing any variable should be followed with a development server restart (or production rebuild, depending on what flow do you currently use).

After that, you will be able to use your variable in the code like this or with any flow you wish

```
function SecretPage() {
  return <div>Your secret is {process.env.SECRET}. Do not tell it to anyone</div>
}

export default ExamplePage
```

***Figure 1.10**: Code in secret page file*

Now we can insert some logic into WebPack in our configuration to act only in development start, as an example:

```
const nextConfig = {
  reactStrictMode: true,
  styledComponents: true,
  webpack: (config, { buildId, dev, isServer, defaultLoaders, webpack }) => {
    if(dev) {
      console.log('Is development flow ', process.env.SECRET) // this line will fire twice because webpack function
is runs for server and client separately
    }
    return config // do not forget to return this object
  },
  env: {
    SECRET: process.env.SECRET
  }
}

module.exports = nextConfig
```

***Figure 1.11**: Code in next.config.js file*

You can inject any logic here or update the config with new plugins that you want to use like this:

```
const nextConfig = {
  reactStrictMode: true,
  styledComponents: true,
  webpack: (config, { buildId, dev, isServer, defaultLoaders, webpack }) => {
    if(dev) {
      console.log('Is development flow ', process.env.SECRET) // this line will fire twice because webpack function
is runs for server and client separately
    }

    const newConfig = config.plugins.push(<YOUR-PLUGINS-CONFIGURATIONS>)

    return newConfig // do not forget to return this object
  },
  env: {
    SECRET: process.env.SECRET
  }
}

module.exports = nextConfig
```

***Figure 1.12**: Code in next.config.js file*

Let us look into an example of how you can use it with a real plugin. So let us check this one for example **https://github.com/vincent-herlemont/next-aws-lambda-webpack-plugin.**_Using this plugin you can use AWS Lambda functions as pages for your application. To implement this plugin you will need to install it first:

```
yarn add --dev next-aws-lambda-webpack-plugin

// or if you prefer NPM

npm install --save-dev next-aws-lambda-webpack-plugin
```

***Figure 1.13**: Command to install plugin into your project*

And then change the WebPack configuration like this to enable it:

```
const nextConfig = {
  reactStrictMode: true,
  styledComponents: true,
   webpack: (config, nextConfig) => {
        const newConfiguration = config.plugins.push(new GenerateAwsLambda(nextConfig))
        return newConfiguration
    },
  env: {
    SECRET: process.env.SECRET
  }
}

module.exports = nextConfig
```

***Figure 1.14**: Code in next.config.js file*

How to use TypeScript in NextJS

In the last NextJS version, there is no need for a special configuration of WebPack for using **TypeScript**

What if I have a NextJS project that was created with JS only?

If you have a NextJS project that was created without typescript just add a **`tsconfig.json`** file into your root and rerun your development server with **`yarn run dev`** (or **`npm run dev`**) after that NextJS will show you the next steps to proceed. You will need to enter these commands to fully configure your project