Moleseng Mokgosi Follow

Jul 6, 2022 · 8 min read · Listen

Migrating a Create-React-App to NextJS

And know the subtle differences between the two frameworks

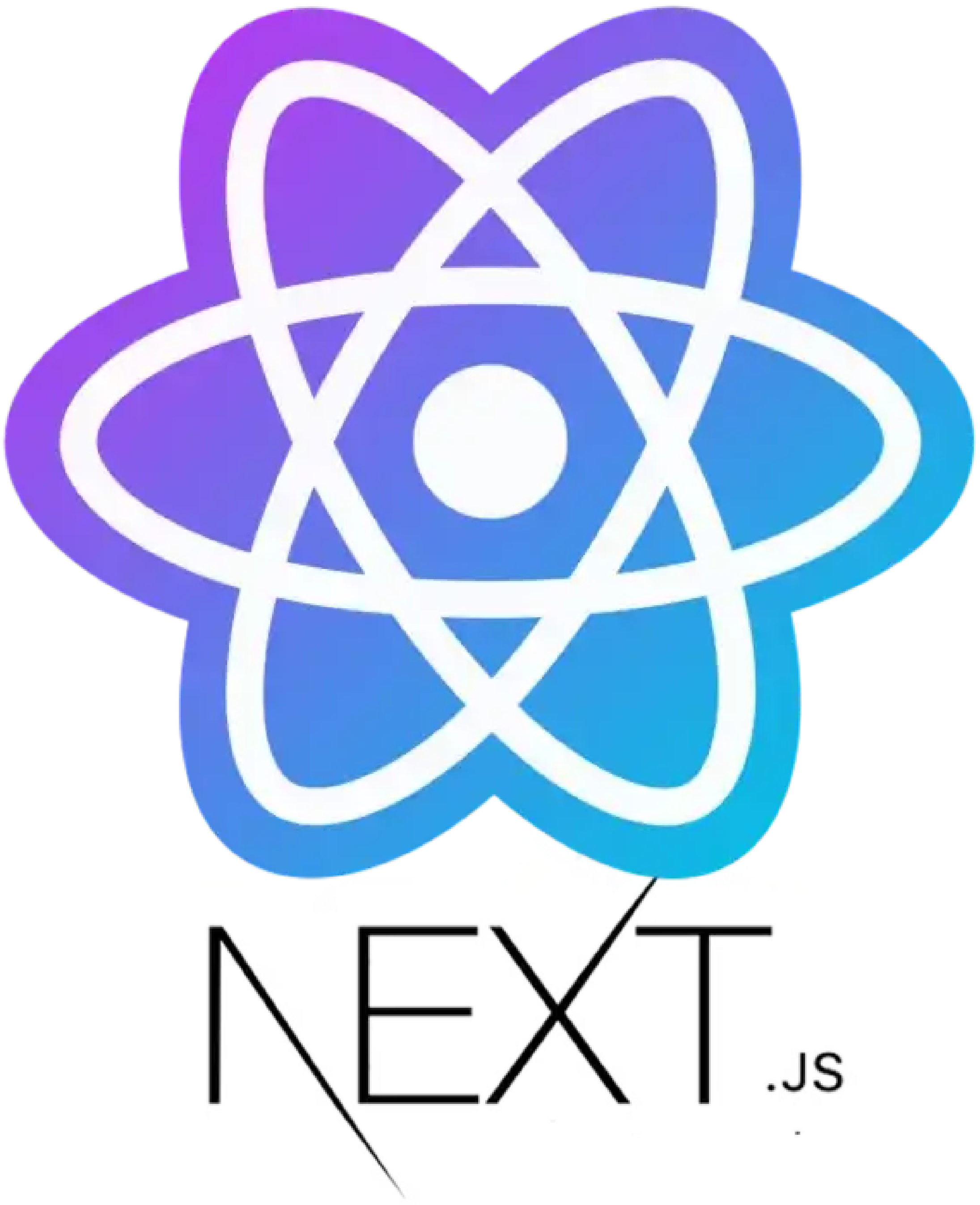


Image from ijsto

I recently started building a web app for a salon, that would include a booking system. I wanted to design this in such a way that it would be easy for the user to navigate through the app, so I want to have a home page that would be the starting point of the application that would have all the information about the salon and then have one other page that would handle the booking system.

I started and coded the complete home page with CRA, and soon realised I have to move it to Next.js. Unfortunately, I could not find help for my exact issue as I had to change most of my functionalities so they work on Next. The following article is a step-by-step guide on how to move a complete CRA to Next.js along with some of the resources that helped me along the way throughout the article.

About Create-React-App

React is a library that is used to create interactive user interfaces by building reusable components. Create-react-app is the quickest and easier way to build a single-page application in React. It sets up your development environment, so you spend less time worrying about the nitty gritty, and more time creating a beautiful interface.

About NextJS

Next.js is a React framework makes it easier for you to build efficient web applications. It makes it possible for you to move from a single-page application to a content rich application that you can navigate through by simplifying the process of creating routes. It also increases performance of your application by making use of the concept of on-demand rendering rather than build-time rendering.

Goal

To migrate a full SPA with all it’s components and styles made with Create React App to Next.js.

Prerequisites

This article is about Next.js which is a React framework. You should be knowledgeable with web development, React.js library and partial knowledge of Next.js is necessary.

This articles assumes you have already created the React app and have either completed the single-page application or have already built the necessary components and styles and all you need now is to transition your project to Next.js.

About the Project

We will be moving a single-page React App along with its components and styles to a Next.js. This will require us to change the file structure.

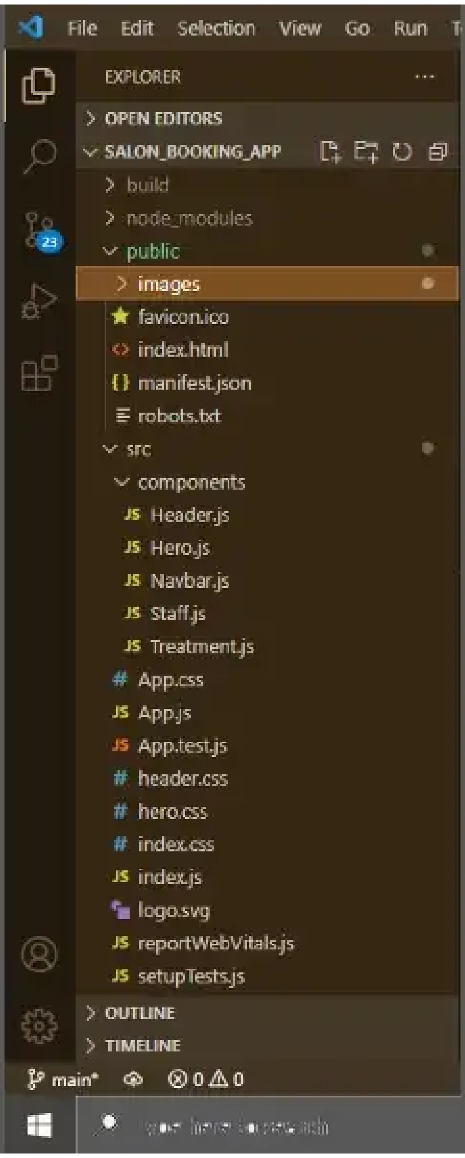
Table of Content

- Installing NextJS
- Configuring Your Application
- NextJS Project File Structure

Create React App File Structure

Create React App contains three main folders:

1. the node modules folder. This folder is created during compiling when you run the "create-react-app" command.
2. the public folder. This folder contains the index.html file, which is the entry point of the application. The images folder in the public folder is optional. Some developers advise that it is best to place your image next to the component where it will be used.
3. the last folder is the src folder. This folder will have the components you have built for your app, your css files as well as the App.js file and the index.js file. Additionally, your src file will have your package.json file, which is very important piece for your project.



```
SALON_BOOKING_APP
  build
  node_modules
  public
    images
    favicon.ico
    index.html
    manifest.json
    robots.txt
  src
    components
      Header.js
      Hero.js
      Navbar.js
      Staff.js
      Treatment.js
    App.css
    App.js
    App.test.js
    header.css
    hero.css
    index.css
    index.js
    logo.svg
    reportWebVitals.js
    setupTests.js
```

CRA File Structure

Installing NextJS

Step 1 — Uninstall React Scripts

We will start by removing the react scripts. Go to your terminal add this command, which will uninstall the react-script dependencies.

```
npm uninstall react-scripts
```

Step 2 — Install Next package

```
npm install next
```

When the Next package has installed, check your main project directory for the `.next` folder. If it isn't there, you can add it yourself (I used a `.next` folder from the create-next-app tutorial. Just copy it and paste it tp your main folder).

Step 3 — Change the scripts in the package.json file to Next scripts

Your scripts should go from this:

```
"scripts": {
   "start": "react-scripts start",
   "build": "react-scripts build",
   "test": "react-scripts test",
   "eject": "react-scripts eject"
}
```

to this:

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

and your dependencies should look like this:

```
"dependencies": {
   "@testing-library/jest-dom": "^5.16.4",
   "@testing-library/react": "^13.3.0",
   "@testing-library/user-event": "^13.5.0",
   "next": "^12.2.0"
   "react": "^18.2.0",
   "react-dom": "^18.2.0",
   "web-vitals": "^2.1.4"
}
```

Configuring your application

Now that you have all the depencies and scripts for Next to work with your project, the next step is to set up your file structure for Next. Next.js does not work in the same way that React does. Where the entry point for a React application is the index.html file, Next does not have any index.html files.

The very first thing you need to do in this section is create a "pages" folder in your main application folder.

Compiled Output

The equivalent of the index.html file in NextJS is the `_document.js` file.

> *A custom* `Document` *can update the* `<html>` *and* `<body>` *tags used to render a Page. This file is only rendered on the server, so event handlers like* `onClick` *cannot be used in* `_document` *.*

Step 1 — Create `_document.js` file

In your "pages" folder, create a new file and name it `_document.js` . Your file should look like this:

Structure of _document.js file

and then to override the default `Document` take everything you have in the head section of your `index.html` file and paste it in the head section of the `_document.js` file. This would be a good place your Google Fonts links and your Fontawesome links if you have any, the end product wil look like this:

Edited _document.js file to override default settings

For more information about how to incorporate `<Title>` and `<Head>` tags on in your project, visit this article .

Step 2 — Create a _app.js file

> *Any shared layout between all pages should be moved to a custom* `_app.js` .

The pages folder is where you will place the `_app.js` file. And if you want meta tags or components (for example a navigation bar and a footer)that will be applied to all the pages, place them in this file.

We start by importing the Head component from next/head

```
import Head from 'next/head';
```

Place your meta tags in the `<Head>` tag:

```
<Head>
  <meta charset="utf-8" />
  <meta name="author" content="Moleseng" />
  <meta name="viewport" content="width=device-width, initial-scale=1"
/>
</Head>
```

in the end, your `_app` component will look like this:

Final _app.js file structure

Step 3 — Components and pages

Now we have arrived at the most interesting part of this process; setting up our components that we made in the React app so we can render them in our Next app. NextJS does not house its components in a src folder, and rather it has two main folders for this, namely; the `components` folder and the `pages` folder.

> *In Next.js, a **page** is a React Component exported from a* `.js`*,* `.jsx`*,* `.ts`*, or* `.tsx` *file in the* `pages` *directory. Each page is associated with a route based on its file name.*

find out more about this here.

So in our pages folder, we will create an `index.js` file. The `index.js` will contain the home page, which will be the first page your user sees when they open your web app. Here, you can liThe rest of the pages folder will contain all the other pages you've made available for your user to visit. It would look something like this:

```
export default function HomePage() {
  return (
    <div>
      Hello, this is HomePage!
    </div>
  )
}
```

You can also add a `<title>` tag if you want the title to be something unique. This is something you can do for each the pages of your app and one of the cool touches I really like from NextJS:

Structure for index.js file in Next.js

The next folder you will create in your main project directory is the components folder. This folder will hold all the reusable components that make up different parts of your application, just like you would have in your React app, only this time it does not fall in the `src` folder. This components can be imported into whichever other component or page that you want to use them in.

Styling

Next.js allows you to import styles into your app just like React, but instead it will require these files to be CSS Modules instead. You can read more about it here.

You will start by creating `styles` folder in your main project directory. Next, take all the style files you have from your original React project and move them ino the `styles` folder. The next stem is to rename the files, for example if you have styling for your navigation bar, it will go from `navbar.css` to `navbar.module.css` . Importing the style into your component and using it will look like this:

```
import styles from '../styles/about.module.css';

export default function Navbar(){
   return (
      <>
       <h1 className={styles.heading}>This is the Navbar</h1>
      </>
   )
}
```

As you can see above, to use the styling you have to first import it using a variable name for it and when using it in your tag, use the dot notation starting with the name of the variable and then target the styling property from your `module.css` file.

Images

Next.js uses the public folder to store all the images from your application and a Image component for image files. This is for image optimization purposes.

To use this component, your first import it into the component you will be using it in:

```
import Image from 'next/image';
```

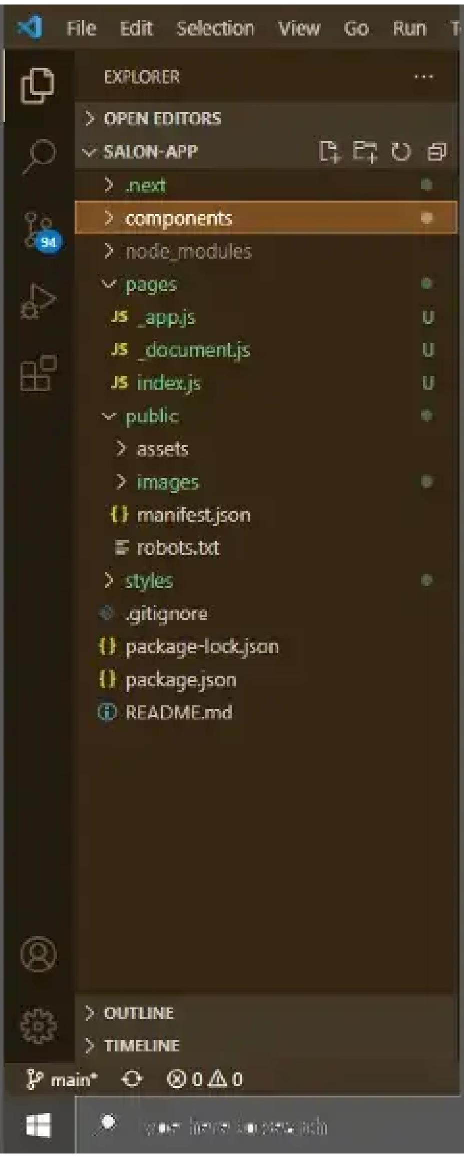
Next, you will want to use the image in your component and the final product will look like this:

Importing and using styles in Next.js component

As you can see above, the image component's `width` and `height` attributes do not receive `px` strings for sizing, instead take they take exact values. This is something you will have to change from your original React application and play around with the values to get the sizing right. Here is where you can get more information about the Image component and its attrubutes.

Next.js Project File Structure

After making all the changes we went through above, your project directory should look like this:



Next.js Project File Structure

At this point, you can now add more pages to your application that your users can be directed to by adding to your project. You can read more about it here.

You should be having a running application now and you can see if it works on your local host by running the following command on your terminal:

```
npm run dev
```

Conclusion

As you have seen above, migrating a complete CRA to Next.js is not very difficult to do. Just remember the following:

- The `index.html` equivalent for Next.js is the `_document.js` file.
- Next.js does not make use of the `src` folder. Your component folder will be inside your main project directory.
- Next.js uses modules to for style application and your styles are imported with a variable name that will be used to access style properties to your elements.
- Next.js uses the Image component to render your image files optimally and on-demand rather than on build time.
- The Next.js project's main project directory file structure is more concise than the CRA project file.

I hope you found this article helpful! Thanks for reading!

Want to Connect?

If you have any questions or suggestions, please reach out to me on LinkedIn.