Migrating from Create React App

This guide will help you understand how to transition from an existing non-ejected Create React App project to Next.js. Migrating to Next.js will allow you to:

- Choose which data fetching strategy you want on a per-page basis.
- Use Incremental Static Regeneration to update *existing* pages by re-rendering them in the background as traffic comes in.
- Use API Routes.

And more! Let's walk through a series of steps to complete the migration.

Updating `package.json` and dependencies

The first step towards migrating to Next.js is to update `package.json` and dependencies. You should:

- Remove `react-scripts` (but keep `react` and `react-dom`). If you're using React Router, you can also remove `react-router-dom`.
- Install `next`.
- Add Next.js related commands to `scripts`. One is `next dev`, which runs a development server at `localhost:3000`. You should also add `next build` and `next start` for creating and starting a production build.

Here's an example `package.json`:

```
{
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start"
  },
  "dependencies": {
    "next": "latest",
    "react": "latest",
    "react-dom": "latest"
  }
}
```

Static Assets and Compiled Output

Create React App uses the `public` directory for the entry HTML file, whereas Next.js uses it for static assets. It's possible to add static assets here, but Create React App recommends importing them directly from JavaScript files.

- Move any images, fonts, or other static assets to `public`.

- Convert `index.html` (the entry point of your application) to Next.js. Any `<head>` code should be moved to a custom `_document.js`. Any shared layout between all pages should be moved to a custom `_app.js`.
- See Styling for CSS/Sass files.
- Add `.next` to `.gitignore`.

Creating Routes & Linking

With Create React App, you're likely using React Router. Instead of using a third-party library, Next.js includes its own file-system based routing.

- Convert all `Route` components to new files in the `pages` directory.
- For routes that require dynamic content (e.g. `/blog/:slug`), you can use Dynamic Routes with Next.js (e.g. `pages/blog/[slug].js`). The value of `slug` is accessible through a query parameter. For example, the route `/blog/first-post` would forward the query object `{ 'slug': 'first-post' }` to `pages/blog/[slug].js` (learn more here).

For more information, see Migrating from React Router.

Styling

Next.js has built-in support for CSS, Sass and CSS-in-JS.

With Create React App, you can import `.css` files directly inside React components. Next.js allows you to do the same, but requires these files to be CSS Modules. For global styles, you'll need a custom `_app.js` to add a global stylesheet.

Safely Accessing Web APIs

With client-side rendered applications (like Create React App), you can access `window`, `localStorage`, `navigator`, and other Web APIs out of the box.

Since Next.js uses pre-rendering, you'll need to safely access those Web APIs only when you're on the client-side. For example, the following code snippet will allow access to `window` only on the client-side.

```
if (typeof window !== 'undefined') {
  // You now have access to `window`
}
```

A recommended way of accessing Web APIs safely is by using the `useEffect` hook, which only executes client-side:

```
import { useEffect } from 'react'

useEffect(() => {
  // You now have access to `window`
}, [])
```

Image Component and Image Optimization

Since version **10.0.0**, Next.js has a built-in Image Component and Automatic Image Optimization.

The Next.js Image Component, `next/image`, is an extension of the HTML `<img>` element, evolved for the modern web.

The Automatic Image Optimization allows for resizing, optimizing, and serving images in modern formats like WebP when the browser supports it. This avoids shipping large images to devices with a smaller viewport. It also allows Next.js to automatically adopt future image formats and serve them to browsers that support those formats.

Instead of optimizing images at build time, Next.js optimizes images on-demand, as users request them. Your build times aren't increased, whether shipping 10 images or 10 million images.

```
import Image from 'next/image'

export default function Home() {
  return (
    <>
      <h1>My Homepage</h1>
      <Image
        src="/me.png"
        alt="Picture of the author"
        width={500}
        height={500}
      />
      <p>Welcome to my homepage!</p>
    </>
  )
}
```

Environment Variables

Next.js has support for `.env` Environment Variables similar to Create React App. The main difference is the prefix used to expose environment variables on the client-side.

- Change all environment variables with the `REACT_APP_` prefix to `NEXT_PUBLIC_`.
- Server-side environment variables will be available at build-time and in API Routes.

Search Engine Optimization

Most Create React App examples use `react-helmet` to assist with adding `meta` tags for proper SEO. With Next.js, we use `next/head` to add `meta` tags to your `<head />` element. For example, here's an SEO component with Create React App:

```
// src/components/seo.js

import { Helmet } from 'react-helmet'

export default function SEO({ description, title, siteTitle }) {
  return (
    <Helmet
      title={title}
      titleTemplate={siteTitle ? `%s | ${siteTitle}` : null}
      meta={[
        {
          name: `description`,
          content: description,
        },
        {
          property: `og:title`,
          content: title,
        },
        {
          property: `og:description`,
          content: description,
        },
        {
          property: `og:type`,
          content: `website`,
        },
        {
          name: `twitter:card`,
          content: `summary`,
        },
        {
          name: `twitter:creator`,
          content: twitter,
        },
        {
          name: `twitter:title`,
          content: title,
        },
        {
```

```
          name: `twitter:description`,
          content: description,
        },
      ]}
    />
  )
}
```

And here's the same example using Next.js.

```
// src/components/seo.js

import Head from 'next/head'

export default function SEO({ description, title, siteTitle }) {
  return (
    <Head>
      <title>{`${title} | ${siteTitle}`}</title>
      <meta name="description" content={description} />
      <meta property="og:type" content="website" />
      <meta property="og:title" content={title} />
      <meta property="og:description" content={description} />
      <meta property="og:site_name" content={siteTitle} />
      <meta property="twitter:card" content="summary" />
      <meta property="twitter:creator" content={config.social.twitter} />
      <meta property="twitter:title" content={title} />
      <meta property="twitter:description" content={description} />
    </Head>
  )
}
```

Single-Page App (SPA)

If you want to move your existing Create React App to Next.js and keep a Single-Page App, you can move your old application's entry point to an Optional Catch-All Route named `pages/[[...app]].js`.

```
// pages/[[...app]].js

import { useState, useEffect } from 'react'
import CreateReactAppEntryPoint from '../components/app'

function App() {
  const [isMounted, setIsMounted] = useState(false)

  useEffect(() => {
    setIsMounted(true)
```

```
  }, [])

  if (!isMounted) {
    return null
  }

  return <CreateReactAppEntryPoint />
}

export default App
```

Ejected Create React App

If you've ejected Create React App, here are some things to consider:

- If you have custom file loaders set up for CSS, Sass, or other assets, this is all built-in with Next.js.
- If you've manually added new JavaScript features (e.g. Optional Chaining) or Polyfills, check to see what's included by default with Next.js.
- If you have a custom code splitting setup, you can remove that. Next.js has automatic code splitting on a per-page basis.
- You can customize your PostCSS setup with Next.js without ejecting from the framework.
- You should reference the default Babel config and Webpack config of Next.js to see what's included by default.

Learn More

You can learn more about Next.js by completing our starter tutorial. If you have questions or if this guide didn't work for you, feel free to reach out to our community on GitHub Discussions.