Building a Micro Frontend Application with Next.js 14.2 and Tailwind CSS — Part-I

Vijayasekhar Deepak · Follow
Published in Stackademic · 7 min read · Jun 29, 2024

Building a Micro Frontend Application with Next.js 14.2 and Tailwind CSS — Part-I

In modern web development, micro frontend architecture has gained popularity for its ability to scale and modularize large applications. Micro frontends allow teams to independently develop, test, and deploy features, leading to faster iteration cycles and improved maintainability. In this tutorial, we'll explore how to build a micro frontend application using Next.js 14.2 and enhance it with Tailwind CSS for consistent styling across components.

Note: This tutorial assumes a basic understanding of Next.js and Tailwind CSS. If you're new to micro frontends or need a refresher on these technologies, please refer to our detailed blog post Understanding Micro Frontends for a comprehensive introduction.

Table of Contents

1. Project Setup

We'll start by setting up the project structure. Create a root directory and three subdirectories for the main application and one micro frontend application.

```
mkdir nextjs-mf-apps
cd nextjs-mf-apps
```

Create the Main Application

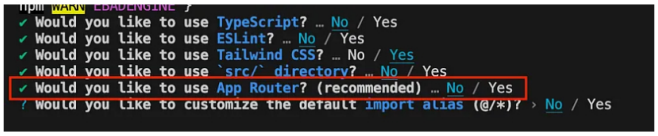
By running the below command in the terminal you can initialise the Next.js app for specific version 14.2.

```
npx create-next-app@14.2 main-app
```

If you want to get the latest versions use the below command.

```
npx create-next-app main-app
```

Once initalized the project setup, there will be multiple options provided by next.js framework, you can select the options based on your requirements except **"Would you like to use App Router?(recommended)".**



Currently in Nextjs latest versions they providing the new feature **App Router** for better folder based routing. But this version of nextjs is not supporting "**Micro frontend**" options with the **App Router** feature enabled apps. So please select **NO** for **App Router** if you want to have **Micro frontend** support.

> *Note: While I'm writing this blog **App Router** based apps don't support **Micro frontend**, but in future it may change.*

Once creating the main app, create our micro frontend app.

Create Micro Frontend App

```
npx create-next-app@14.2 shop-app
```

Follow the same steps of main-app to create this micro frontend app.

Your project structure should look like this:

```
/nextjs-mf-apps
  /main-app
  /shop-app
```

2. Installing required packages

Install the required packages in both the apps to enable **Micro frontend** features in both the apps

features in both the apps.

Run the below commands in both the apps “main-app” and “shop-app”.

```
npm i @module-federation/nextjs-mf webpack
```

In Next.js apps, We are using Webpack Modular Federation feature to enable **Micro frontend.** In my previous blog posts i have already explained all about Webpack and Modular Federation. Please click here for more details.

Once installing the packages in both the apps, go to `package.json` file and add `NEXT_PRIVATE_LOCAL_WEBPACK=true` to the `dev` command in the `script` section in both the apps as below.

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

3. Setting Up the Micro Frontend Configuration

Add the below configuration code to `next.config.js` file to use the `NextFederationPlugin` in **Main App**.

```
import NextFederationPlugin from '@module-federation/nextjs-mf';

const nextConfig = {
  webpack(config, options) {
    const { isServer } = options;
    config.plugins.push(
      new NextFederationPlugin({
        name: 'main_app',
        remotes: {},
        filename: 'static/chunks/remoteEntry.js',
        exposes: {},
        extraOptions: {
          debug: false, // `false` by default
          exposePages: false, // `false` by default
        },
        shared: {},
      })
    );

    return config;
  },
};

export default nextConfig;
```

Next, add the same configuration code in Shop app as well and update the `name` property to `shop_app` as below.

```
import NextFederationPlugin from '@module-federation/nextjs-mf';

const nextConfig = {
  webpack(config, options) {
    const { isServer } = options;
    config.plugins.push(
      new NextFederationPlugin({
        name: 'shop_app',
        remotes: {},
        filename: 'static/chunks/remoteEntry.js',
        exposes: {},
        extraOptions: {
          debug: false, // `false` by default
          exposePages: false, // `false` by default
        },
        shared: {},
      })
    );

    return config;
  },
};

export default nextConfig;
```

Explanation:

In the above configuration, you should be providing the `name` off your app (it should be a single word, should not contains any special characters **Ex: main_app, mainApp**), next the build file name for modular federation as **static/chunks/remoteEntry.js.** These are the two information used in the app configuration.

Next the `expose` object is where we list down all the components that we are trying to export from this app and the `remote` object is to configure other micro frontend app's to be configured. `shared` object will hold all the dependencies of our app.

As we are going to run two apps simultaneously, we can assign specific port to our **Shop app** as below in `package.json` file.

```
"scripts": {
    "dev": "NEXT_PRIVATE_LOCAL_WEBPACK=true next dev -p 3001",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  },
```

Now go to `index.js` file of **main-app** project and replace with the below code.

```
export default function Home() {
  return (
    <main className='h-[100vh] flex justify-center items-center'>
      Hello main app
    </main>
  );
}
```

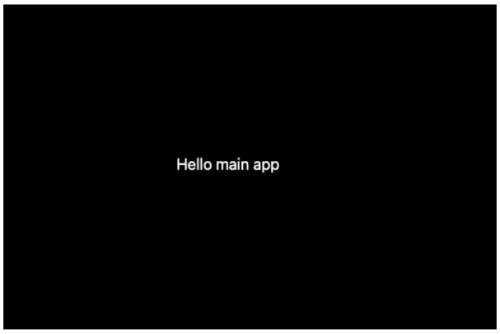
Next go to `index.js` file of **shop-app** project and replace with the below code.

```
export default function Home() {
  return (
    <main className='h-[100vh] flex justify-center items-center'>
      Hello shop app
    </main>
  );
}
```

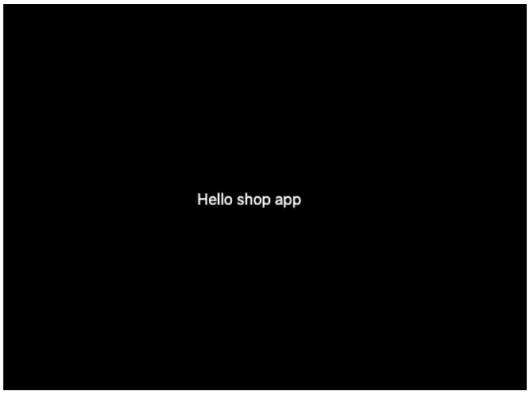
Once doing all this basic changes run both the apps with the below command.

```
npm run dev
```

The output for the both apps looks like below images.



main app



shop app

4. Creating Components into the Main Application

Now create folder `components` in the root directory of main-app and create two components `Navbar.js` and `Footer.js` and paste the code below to the respective files.

`Navbar.js`

```
import React from 'react';

export default function Navbar() {
  return (
    <nav className='bg-red-500 flex h-[15vh]'>
      <div className='text-3xl font-bold flex items-center p-5 text-white'>
        Main Application
      </div>
    </nav>
  );
}
```

`Footer.js`

```
import React from 'react';

export default function Footer() {
  return (
    <div className='flex h-[10vh] bg-blue-500'>
      <div className='text-3xl font-light flex w-[100%] items-center justify-cen
        @ Main App {new Date().getFullYear()}
      </div>
    </div>
  );
}
```

Now import these two components into the `index.js` file as below.

```
import Footer from '@/components/Footer';
import Navbar from '@/components/Navbar';

export default function Home() {
  return (
    <main className='flex flex-col bg-white'>
      <Navbar />
      <div className='h-[75vh] py-4'>
        <h1 className='text-3xl font-bold font-mono h-[100%] flex justify-center
          Welcome to Main Application
        </h1>
      </div>
      <Footer />
    </main>
  );
}
```

No the output of this changes will looks like:

main app

5. Creating Components into the Shop Application

Similar to our main application we will be creating a Products component into our Shop app. Now create folder `components` in the root directory of main-app and create a component `Products.js` and paste the code below.

```
import React, { useEffect, useState } from 'react';

export default function Products() {
  const [data, setData] = useState([]);
  async function getData() {
    fetch('https://fakestoreapi.com/products')
      .then((res) => res.json())
      .then((res) => {
        setData(res);
      });
  }
  useEffect(() => {
    getData();
  }, []);
  return (
    <div className='bg-white'>
      <div className='mx-auto max-w-2xl px-4 py-16 sm:px-6 sm:py-24 lg:max-w-7xl
        <h2 className='text-2xl font-bold tracking-tight text-gray-900'>
          Products list:
        </h2>
        <div className='mt-6 grid grid-cols-1 gap-x-4 gap-y-10 sm:grid-cols-2 lg
          {data.map((product) => (
            <div className='group relative' key={product.id}>
              <div className='aspect-h-1 aspect-w-1 w-full overflow-hidden round
                <img
                  src={product.image}
                  alt={product.title}
                  className='h-full w-full object-cover object-center lg:h-full
                />
              </div>
              <div className='mt-4 flex justify-between'>
                <div>
                  <h3 className='text-sm text-gray-700'>
                    <a>
                      <span
                        aria-hidden='true'
                        className='absolute inset-0'
                      ></span>
                      {product.title}
                    </a>
                  </h3>
                  <p className='mt-1 text-sm text-gray-500'>
                    <span className='font-medium'>⭐</span>{' '}
                    <span className='font-bold'>{product.rating.rate}</span> (
                    {product.rating.count})
                  </p>
                </div>
                <p className='text-sm font-medium text-gray-900'>
                  ${product.price}
                </p>
              </div>
            </div>
          ))}
        </div>
      </div>
    </div>
  );
}
```

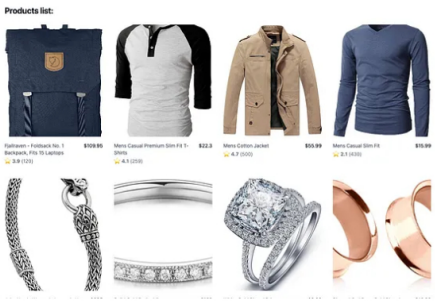
In this component we are using the **free API** to show some products in our page.

In this app, importing the Products component into the `index.js` and paste the below code.

```
import Products from '@/components/Products';

export default function Home() {
  return (
      <Products />
  );
}
```

Once done the changes, the output will looks like below.



shop app

Next Part:

https://tarzzotech.medium.com/building-a-micro-frontend-application-with-next-js-14-2-and-tailwind-css-part-ii-e1f4ae10f6e9

> *Note: This Next.js micro frontend series completely a new implementation to me, I'm also starting now. If you guys find any mistakes or any different approaches you know, please let me know i fell happy to learn from you.*

Stackademic 🎓

Thank you for reading until the end. Before you go:

- Please consider **clapping** and **following** the writer! 👏
- Follow us **X** | **LinkedIn** | **YouTube** | **Discord**
- Visit our other platforms: **In Plain English** | **CoFeed** | **Differ**
- More content at **Stackademic.com**

Nextjs | Micro Frontends | React Microfrontend | Implement Micro Frontend | Nextjs Micro Frontend

21 1

Written by Vijayasekhar Deepak

83 Followers · Writer for Stackademic

Learn form your failure

Follow

More from Vijayasekhar Deepak and Stackademic

Vijayasekhar Deepak in Stackademic

Building a Micro Frontend Application with Next.js 14.2 and...

Previously we have created the Main and Shop apps and some components for the...

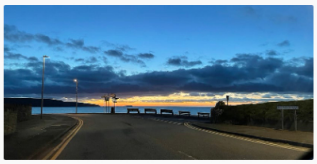
Jun 30 · 28 · 3

Dylan Cooper in Stackademic

Google Python Team Entirely Laid Off, Flutter Team Also "Facing the...

Google's good news and bad news came very suddenly.

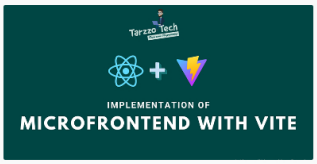
May 2 · 1.7K · 27



Peter Bunyan in Stackademic

Who needs Redis, when Postgres will do?

May 9 · 592 · 5



Vijayasekhar Deepak in Stackademic

Implementation of Microfrontend with Vite Compiler

In my previous article's, i have explained about What is Microfrontend in details and...

Mar 5 · 7 · 1

See all from Vijayasekhar Deepak

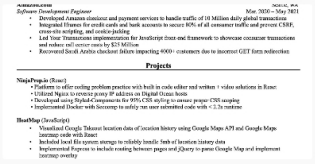
See all from Stackademic

Recommended from Medium

Mark Harris

Navigating Next.js: Crafting Maintainable Code with Clean...

Jul 3



Alexander Nguyen in Level Up Coding

The resume that got a software engineer a $300,000 job at Google.

1-page. Well-formatted.

Jun 1 · 11.8K · 155

Lists

Staff Picks
684 stories · 1124 saves

Stories to Help You Level-Up at Work
19 stories · 685 saves

Self-Improvement 101
20 stories · 2270 saves

Productivity 101
20 stories · 2002 saves

Tari Ibaba in Coding Beauty

New HTML <dialog> tag: An absolute game changer

Afan Khan in JavaScript in Plain English

Microsoft is ditching React

Here's why Microsoft considers React a

The new <dialog> tag changes everything for UI design.

Jun 29 593 8

mistake for Edge.

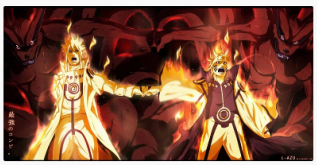
Jun 6 2.5K 60

Vijayasekhar Deepak in Stackademic

Building a Micro Frontend Application with Next.js 14.2 and...

Previously we have created the Main and Shop apps and some components for the...

Jun 30 28 3



Guhaprasaanth Nandagopal

React vs. Next.js: Microfrontend Matchup—Making Them Play Nic...

Microfrontend (MFE) architecture allows you to build complex applications by composing...

Jun 14 19

See more recommendations