swipe

NEXT.JS

Setup Redux in NextJS

Wasif Alam
@wasif alam

Step #1: Installing Next App

Let’s first install Next.js using the following command:

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

Ensure that you use the following settings to achieve the same result as mine

```
→  Desktop npx create-next-app@latest
✔ What is your project named? … redux-next
✔ Would you like to use TypeScript? … No / Yes
✔ Would you like to use ESLint? … No / Yes
✔ Would you like to use Tailwind CSS? … No / Yes
✔ Would you like to use `src/` directory? … No / Yes
✔ Would you like to use App Router? (recommended) … No / Yes
✔ Would you like to customize the default import alias (@/*)? … No / Yes
? What import alias would you like configured? › @/*
```

Wasif Alam
@wasif alam

Step #2: Installing Redux

Now, navigate to the root directory of the Next.js app we've just created and proceed to install the following npm packages:

```
npm i react-redux @reduxjs/toolkit
```

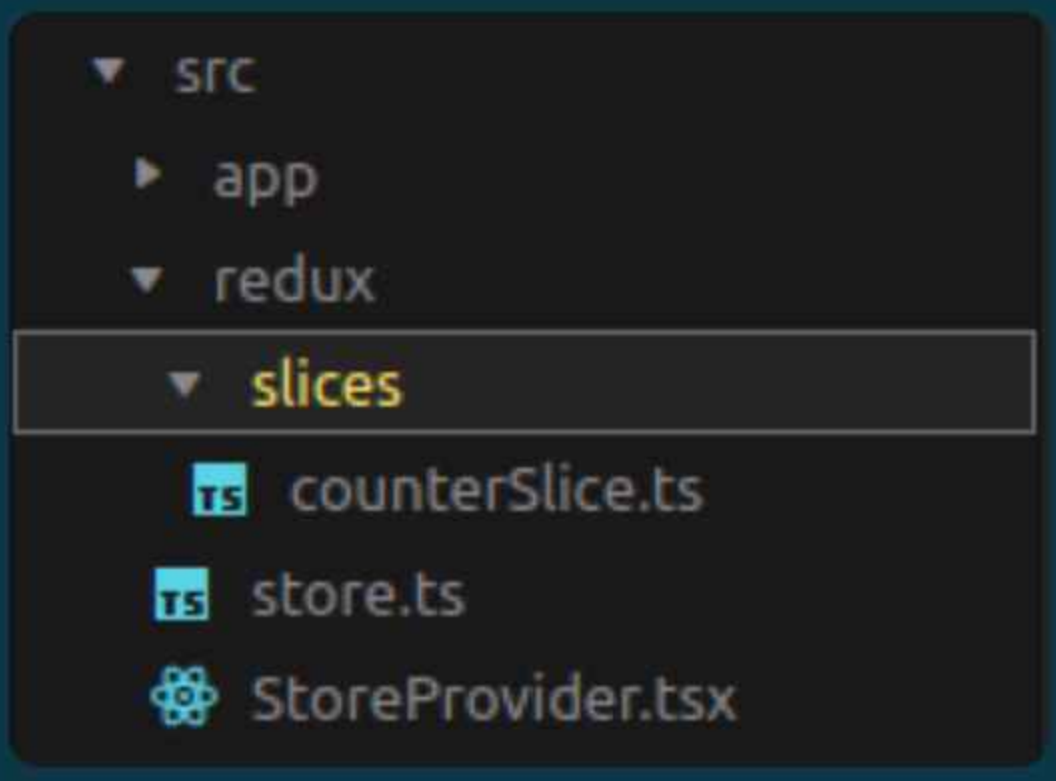
Make sure your 'package.json' dependencies resemble this:

```
"dependencies": {
    "@reduxjs/toolkit": "^1.9.7",
    "next": "13.5.4",
    "react": "^18",
    "react-dom": "^18",
    "react-redux": "^8.1.3"
  },
```

Wasif Alam
@wasif alam

Step #3: Setting Up Redux

Create a 'redux' folder within the 'src' folder, and set up the folders and files as shown below:



We will be implementing a very basic counter app for an example.

Wasif Alam
@wasif alam

Step #4: Creating a Redux Slice

Place the following code within the 'counterSlice.ts' file:

```
import { createSlice } from '@reduxjs/toolkit';

const counterSlice = createSlice({
  name: 'counter',
  initialState: {
    count: 0,
  },
  reducers: {
    increment(state) {
      state.count++;
    },

    decrement(state) {
      state.count--;
    },
  },
});

export const counterActions = counterSlice.actions;
export default counterSlice;
```

Wasif Alam
@wasif alam

Step #5: Setting up the Redux Store

Place the following code within the 'store.ts' file:

```
import { configureStore } from '@reduxjs/toolkit';
import counterSlice from './slices/counterSlice';

export const store = configureStore({
  reducer: {
    counter: counterSlice.reducer,
  },
});
```

Wasif Alam
@wasif alam

Step #6: Setting up the StoreProvider

Place the following code within the 'StoreProvider.tsx' file:

```
'use client';

import { store } from './store';
import { Provider } from 'react-redux';

export const StoreProvider = (
{ children }: { children: React.ReactNode }
) => {
  return <Provider store={store}>{children}</Provider>;
};
```

Wasif Alam
@wasif alam

Step #7: Connecting Redux to Next.js

Next, go to the 'src/app' folder and open 'layout.tsx'.

Import the 'StoreProvider' compoent.

```
import { StoreProvider } from '@/redux/StoreProvider';
```

Wrap the 'children' prop with 'StoreProvider'.

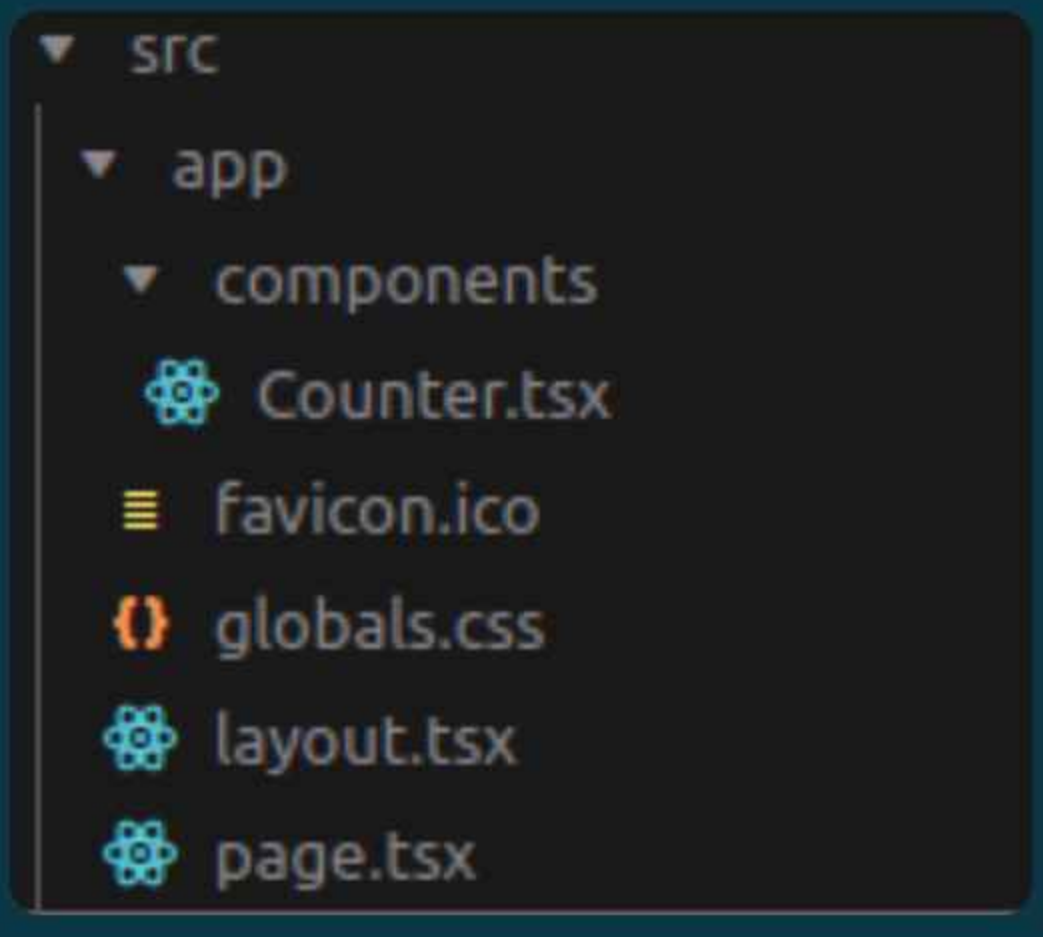
```
export default function RootLayout({
  children,
}: {
  children: React.ReactNode;
}) {
  return (
    <html lang="en">
      <body className={inter.className}>
        <StoreProvider>{children}</StoreProvider>
      </body>
    </html>
  );
}
```

Wasif Alam
@wasif alam

Step #8: Dispatching Actions & Accessing state

Within the same 'src/app' folder create a folder named 'components' and create a file named ‘Counter.tsx’ inside of it.

Should look something like this:



Wasif Alam
@wasif alam

Inside of the 'Counter.tsx' file we need to import a couple of hooks and our 'counterActions' action creators for dispatching actions.

Imports should look something like this:

```
'use client';
import { useSelector, useDispatch } from 'react-redux';
import { counterActions } from '@/redux/slices/counterSlice';
```

Note - We have to use ‘use client’ directive at the top in order to access state and disptach actions in Next.js

Wasif Alam
@wasif alam

Now we need to add the following code in order to access state and dispatch actions inside of the 'Counter' component:

```
const Counter = () => {
  const counter = useSelector(
  (state: any) => state.counter.count);

  const dispatch = useDispatch();

  const incrementHandler = () => {
    dispatch(counterActions.increment());
  };

  const decrementHandler = () => {
    dispatch(counterActions.decrement());
  };

  return // Return goes here

export default Counter;
```

Wasif Alam
@wasif alam

Next, simply return the following ‘div’ from the 'Counter' component to display a basic UI in the browser:

```
const Counter = () => {
// Redux state, dispatch and handler
// function goes here

  return (
    <div>
      <div>
        <button onClick={decrementHandler}>
          Decrement
        </button>
        <button onClick={incrementHandler}>
          Increment
        </button>
      </div>
    </div>
  );
export default Counter;
```

Wasif Alam
@wasif alam

Now, one last step remains – import the 'Counter' component into the 'page.tsx' file, located at the root of the 'src/app' folder.

```
import Counter from './components/Counter';

const Home = () => {
  return (
    <main>
      <Counter />
    </main>
  );
};

export default Home;
```

Note: The 'page.tsx' file is a server component, and we are importing 'Counter.tsx' which is a client component, into it.

Wasif Alam
@wasif alam

That's it, everyone! Now, when we run our development server for the Next.js app, we should expect to see a very basic UI like this, and it should work just fine:

Note: I've added basic styles for a better look, which are not part of our initial 'Counter' component.

Wasif Alam
@wasif alam