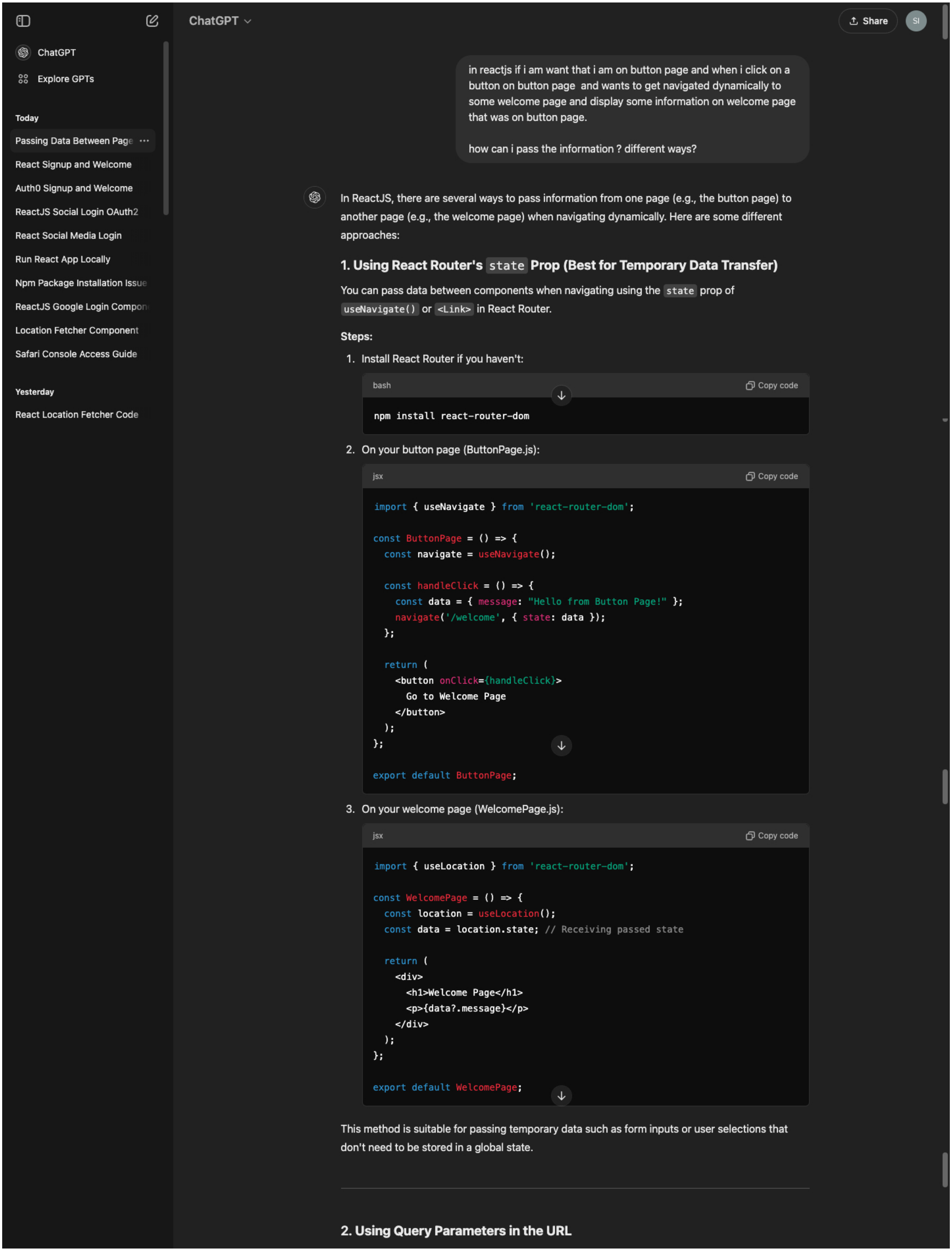

ChatGPT
Explore GPTs
Today
Passing Data Between Page
React Signup and Welcome
Auth0 Signup and Welcome
ReactJS Social Login OAuth2
React Social Media Login
Run React App Locally
Npm Package Installation Issue
ReactJS Google Login Compon
Location Fetcher Component
Safari Console Access Guide
Yesterday
React Location Fetcher Code
ChatGPT
Share
SI

in reactjs if i am want that i am on button page and when i click on a button on button page and wants to get navigated dynamically to some welcome page and display some information on welcome page that was on button page.

how can i pass the information ? different ways?

In ReactJS, there are several ways to pass information from one page (e.g., the button page) to another page (e.g., the welcome page) when navigating dynamically. Here are some different approaches:

1. Using React Router's `state` Prop (Best for Temporary Data Transfer)

You can pass data between components when navigating using the `state` prop of `useNavigate()` or `<Link>` in React Router.

Steps:

1. Install React Router if you haven't:

```bash
npm install react-router-dom
```

2. On your button page (ButtonPage.js):

```jsx
import { useNavigate } from 'react-router-dom';

const ButtonPage = () => {
  const navigate = useNavigate();

  const handleClick = () => {
    const data = { message: "Hello from Button Page!" };
    navigate('/welcome', { state: data });
  };

  return (
    <button onClick={handleClick}>
      Go to Welcome Page
    </button>
  );
};

export default ButtonPage;
```

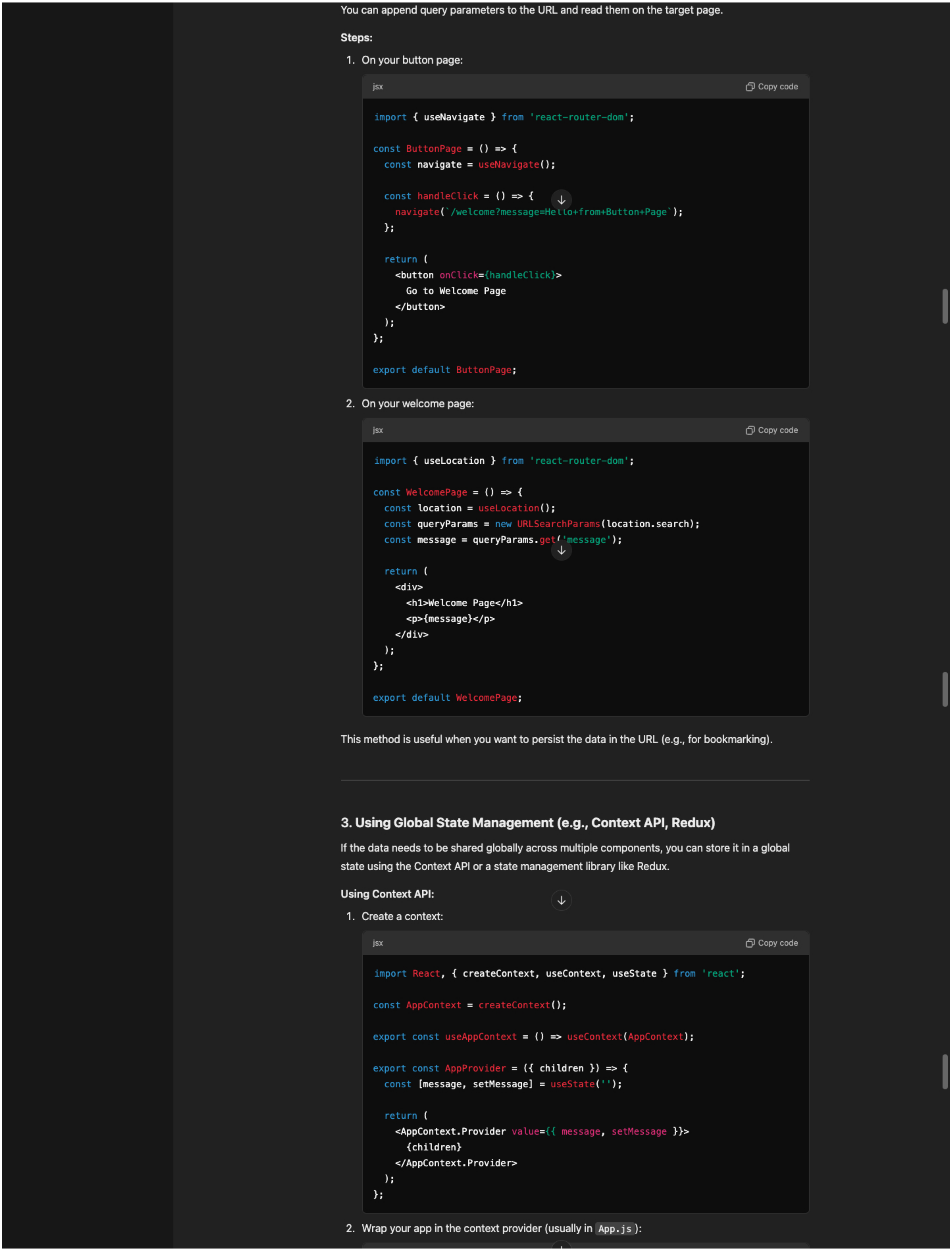
3. On your welcome page (WelcomePage.js):

```jsx
import { useLocation } from 'react-router-dom';

const WelcomePage = () => {
  const location = useLocation();
  const data = location.state; // Receiving passed state

  return (
    <div>
      <h1>Welcome Page</h1>
      <p>{data?.message}</p>
    </div>
  );
};

export default WelcomePage;
```

This method is suitable for passing temporary data such as form inputs or user selections that don't need to be stored in a global state.

---

2. Using Query Parameters in the URL



You can append query parameters to the URL and read them on the target page.

Steps:

1. On your button page:

```jsx
import { useNavigate } from 'react-router-dom';

const ButtonPage = () => {
  const navigate = useNavigate();

  const handleClick = () => {
    navigate(`/welcome?message=Hello+from+Button+Page`);
  };

  return (
    <button onClick={handleClick}>
      Go to Welcome Page
    </button>
  );
};

export default ButtonPage;
```

2. On your welcome page:

```jsx
import { useLocation } from 'react-router-dom';

const WelcomePage = () => {
  const location = useLocation();
  const queryParams = new URLSearchParams(location.search);
  const message = queryParams.get('message');

  return (
    <div>
      <h1>Welcome Page</h1>
      <p>{message}</p>
    </div>
  );
};

export default WelcomePage;
```

This method is useful when you want to persist the data in the URL (e.g., for bookmarking).

---

3. Using Global State Management (e.g., Context API, Redux)

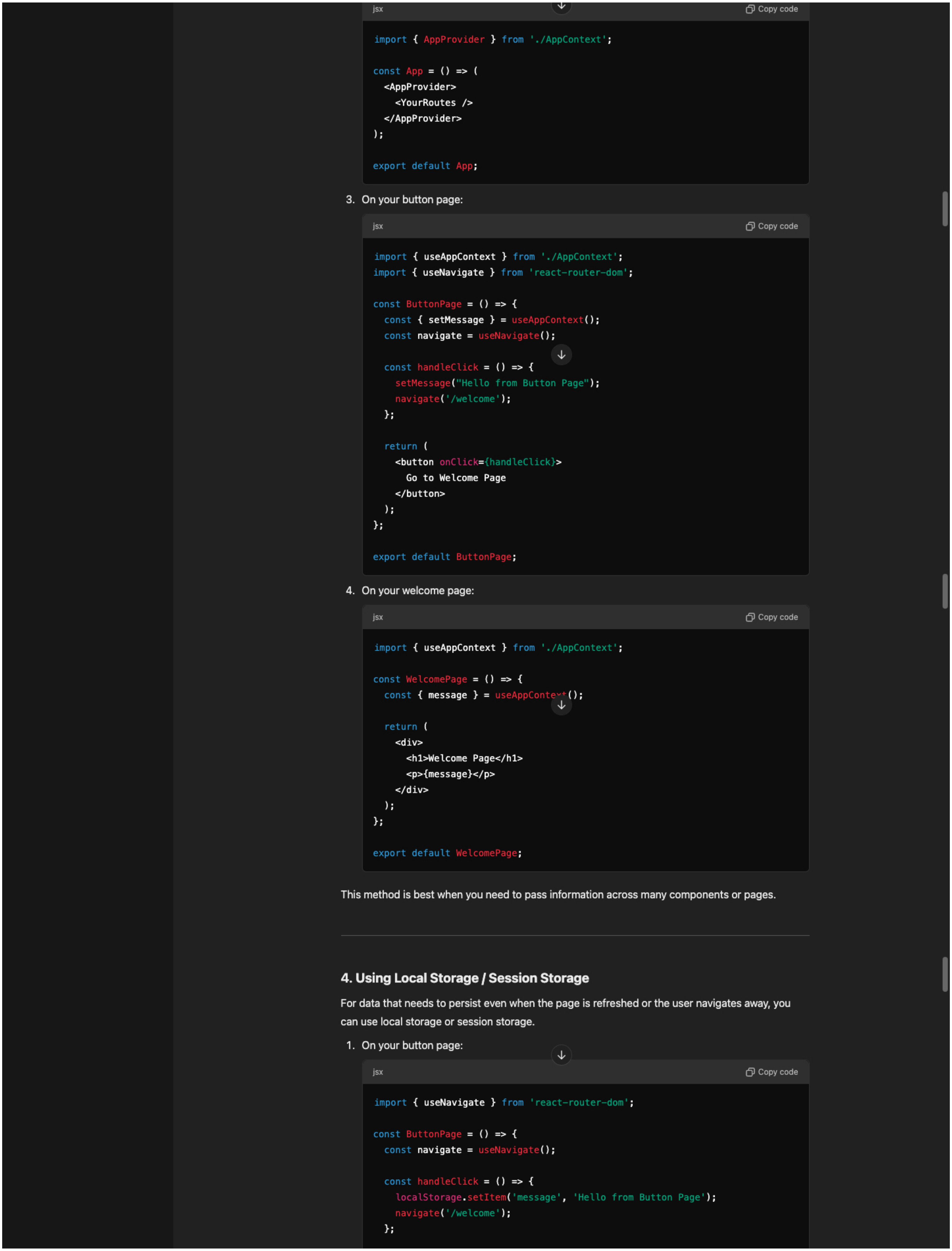
If the data needs to be shared globally across multiple components, you can store it in a global state using the Context API or a state management library like Redux.

Using Context API:

1. Create a context:

```jsx
import React, { createContext, useContext, useState } from 'react';

const AppContext = createContext();

export const useAppContext = () => useContext(AppContext);

export const AppProvider = ({ children }) => {
  const [message, setMessage] = useState('');

  return (
    <AppContext.Provider value={{ message, setMessage }}>
      {children}
    </AppContext.Provider>
  );
};
```

2. Wrap your app in the context provider (usually in `App.js`):



```jsx
import { AppProvider } from './AppContext';

const App = () => (
  <AppProvider>
    <YourRoutes />
  </AppProvider>
);

export default App;
```

3. On your button page:

```jsx
import { useAppContext } from './AppContext';
import { useNavigate } from 'react-router-dom';

const ButtonPage = () => {
  const { setMessage } = useAppContext();
  const navigate = useNavigate();

  const handleClick = () => {
    setMessage("Hello from Button Page");
    navigate('/welcome');
  };

  return (
    <button onClick={handleClick}>
      Go to Welcome Page
    </button>
  );
};

export default ButtonPage;
```

4. On your welcome page:

```jsx
import { useAppContext } from './AppContext';

const WelcomePage = () => {
  const { message } = useAppContext();

  return (
    <div>
      <h1>Welcome Page</h1>
      <p>{message}</p>
    </div>
  );
};

export default WelcomePage;
```

This method is best when you need to pass information across many components or pages.

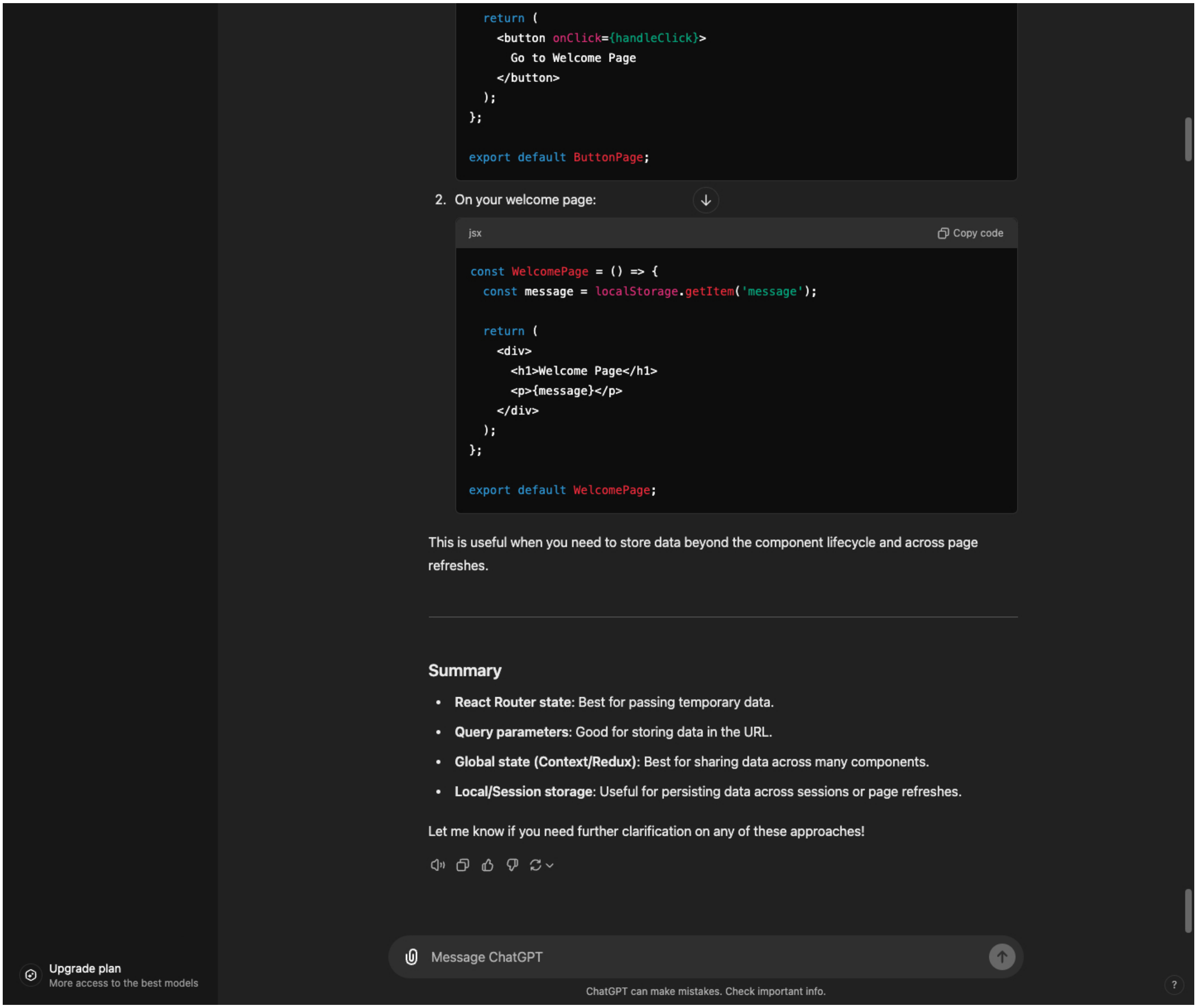
---

4. Using Local Storage / Session Storage

For data that needs to persist even when the page is refreshed or the user navigates away, you can use local storage or session storage.

1. On your button page:

```jsx
import { useNavigate } from 'react-router-dom';

const ButtonPage = () => {
  const navigate = useNavigate();

  const handleClick = () => {
    localStorage.setItem('message', 'Hello from Button Page');
    navigate('/welcome');
  };
```



```jsx
  return (
    <button onClick={handleClick}>
      Go to Welcome Page
    </button>
  );
};

export default ButtonPage;
```

2. On your welcome page:

```jsx
const WelcomePage = () => {
  const message = localStorage.getItem('message');

  return (
    <div>
      <h1>Welcome Page</h1>
      <p>{message}</p>
    </div>
  );
};

export default WelcomePage;
```

This is useful when you need to store data beyond the component lifecycle and across page refreshes.

Summary

- **React Router state**: Best for passing temporary data.
- **Query parameters**: Good for storing data in the URL.
- **Global state (Context/Redux)**: Best for sharing data across many components.
- **Local/Session storage**: Useful for persisting data across sessions or page refreshes.

Let me know if you need further clarification on any of these approaches!

Upgrade plan
More access to the best models
Message ChatGPT
ChatGPT can make mistakes. Check important info.
?