Internship Report – Frontend Dev

Week 7: React.js Advanced

Name: Zainab
Father Name: Assad Qayyum
Date: 5th Aug, 2025
Internship Domain: Front-end Intern
Task: Conditional Styling, React Router (pages, navigation)

Task Overview: (Day2)

Today's task focused on learning **advanced React.js concepts**, specifically **Conditional Styling** and **React Router (pages and navigation).** The goal was to understand how to style components dynamically and create multiple pages in a React application with smooth navigation.

Content Covered:

The content covered is:

- Introduction to conditional styling
- Applying dynamic styles using state and props
- Understanding React Router
- Setting up multiple pages and navigating between them

1) Conditional Styling

Conditional styling is the process of **changing the appearance of components dynamically** based on their state or props. For example, a button can appear green when active and gray when inactive.

It is changing the look of a component dynamically based on:

- **State** (like a button clicked or not)
- **Props** (data passed from parent component)
- **Other conditions** (screen size, user role, etc.)

Uses:

- Highlighting active buttons or menu items
- Showing or hiding UI elements dynamically
- Indicating error or success states in forms
- Making the interface responsive to user actions

Ways to apply Conditional Styling:

Inline Styles: Directly applying style based on conditions using ternary operators.

Conditional Classes: Switching between CSS classes based on component state.

NavLink (with isActive): Highlighting active navigation links automatically.

a) Inline Conditional Styles

Apply styles directly using the style prop: (Changes background color based on active state)

```
function MyButton() {
  const [active, setActive] = useState(false);

  return (
    <button
      style={{
        backgroundColor: active ? "green" : "gray",
        color: "white",
        padding: "10px",
      }}
      onClick={() => setActive(!active)}
    >
      {active ? "Active" : "Inactive"}
    </button>
  );
}
```

b) Conditional Class Names

Switch between CSS classes:

```
function MyButton({ isDisabled }) {
  return (
    <button className={isDisabled ? "btn-disabled" : "btn-enabled"}>
      Click Me
    </button>
  );
}
```

Css:

```
.btn-enabled { background: blue; color: white; }
.btn-disabled { background: gray; color: darkgray; }
```

2) React Router (Pages and Navigation)

React Router is a routing library that allows you to **create multiple pages in a single-page React application**. Instead of reloading the entire page when **switching between sections**, React Router **dynamically loads components**, making navigation fast and smooth.

Uses:

- Building apps with multiple pages (Home, About, Contact)
- Smooth navigation without full page reloads
- Managing private routes and nested routes
- Creating 404 pages for invalid URLs

Core Components of React Router:

BrowserRouter – Wrapper for the whole app, enables routing.

Routes – Container for all routes.

Route – Defines a path and its component.

Link / NavLink – Used to navigate between pages (avoids full reload).

useNavigate – Programmatic navigation (e.g., after login).

Outlet – For nested routes.

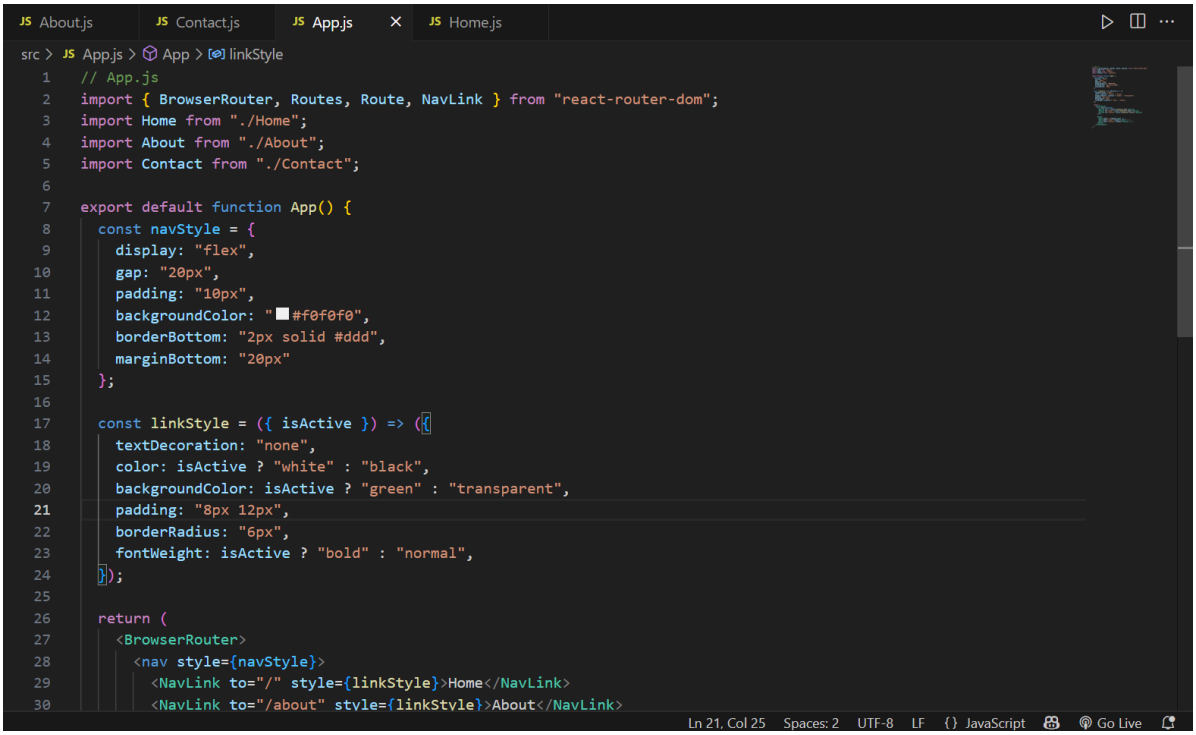
Navigate – Redirects user to a different route.

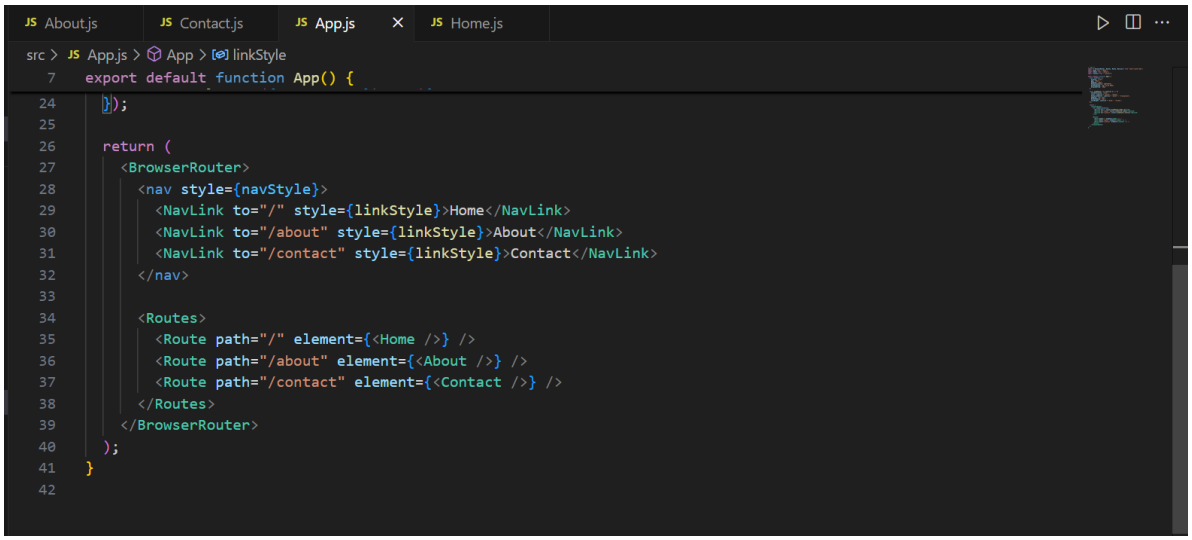
Practice Code:

In this code,

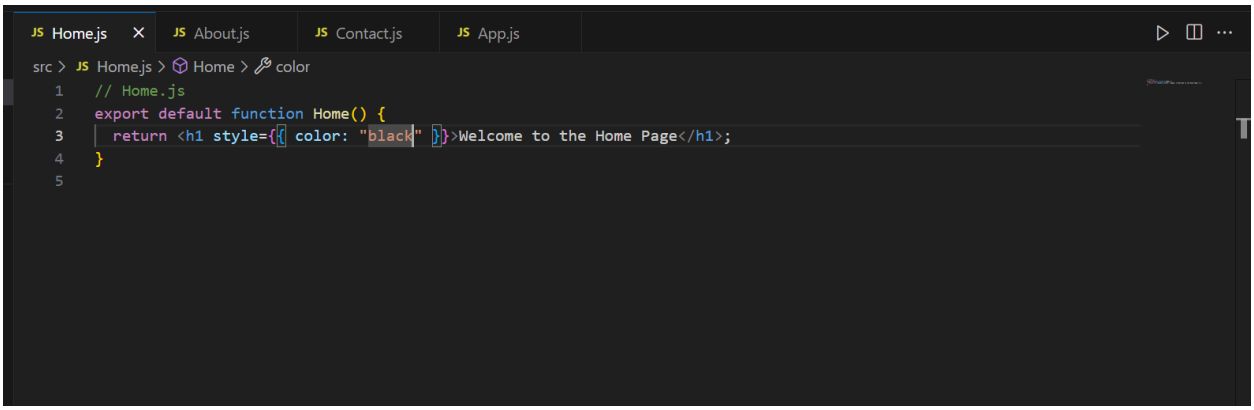
- We created a **multi-page React app** using React Router with three pages (Home, About, Contact).
- We used **NavLink for navigation** and **applied conditional inline styling** to highlight the currently active link by changing its background, text color, and font weight.
- This makes the navigation bar more interactive and user-friendly.

App.js:

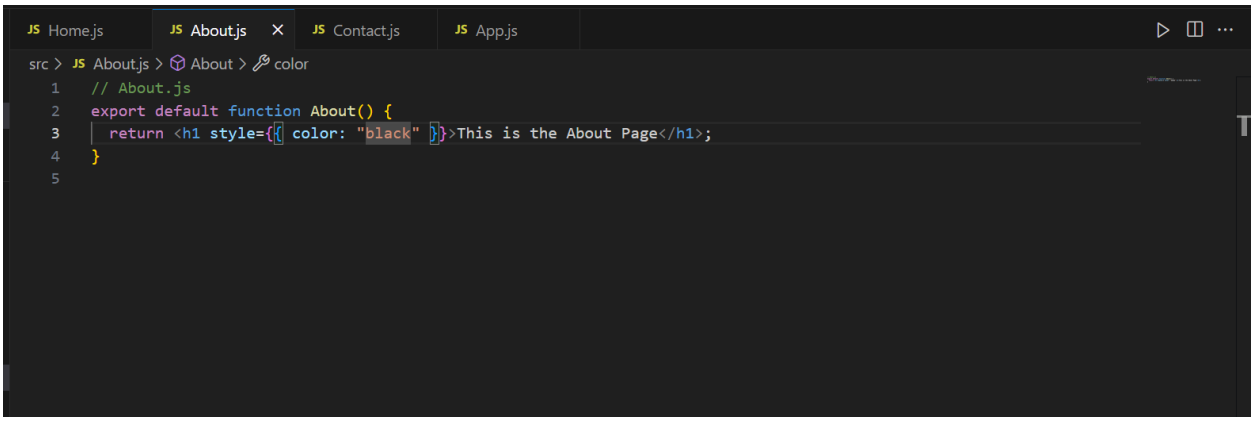




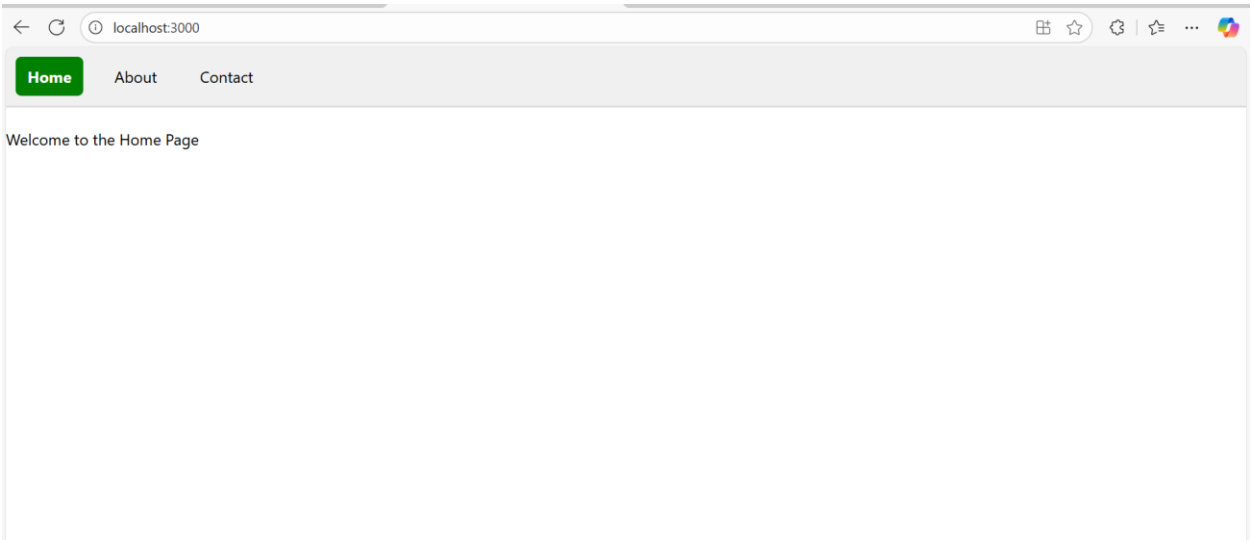
Home.js:

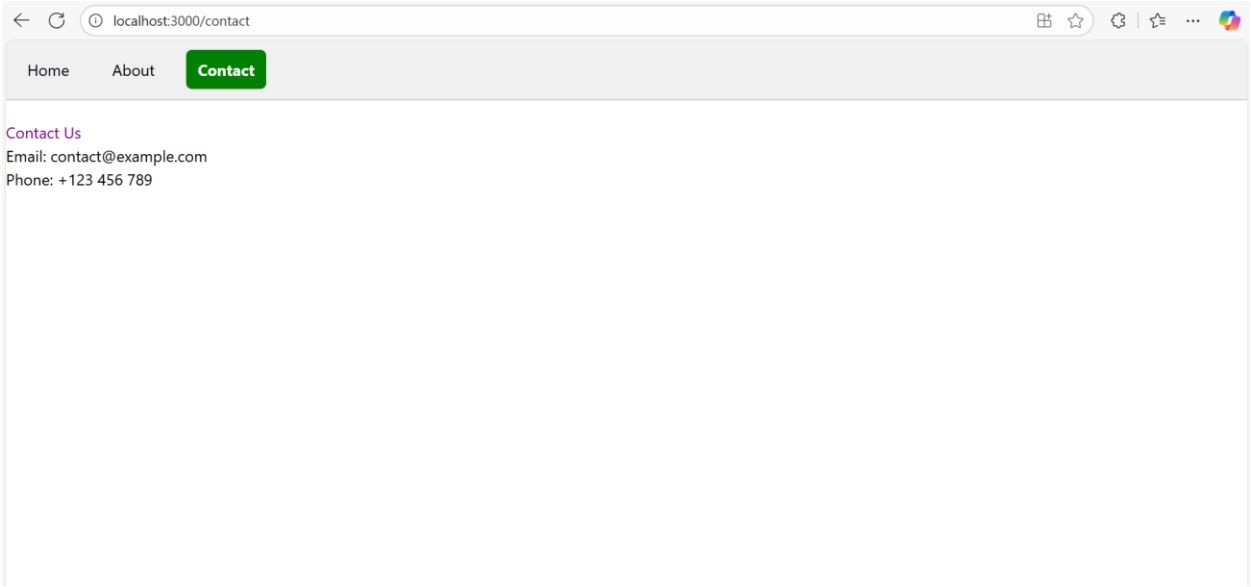


About.js:



Contact.js:





Conclusion:

In this task, I learned how to use **conditional styling** to dynamically change the appearance of components and how to use **React Router** to create and navigate between multiple pages in a React application. These concepts are essential for building interactive, multi-page, and user-friendly web applications efficiently.