2. Styled Components

Styled Components are one of the new ways to use CSS in modern JavaScript. Components are used to reuse code, similarly Styled Components are used to reuse styles.

2.1 Installation

```
npm install styled-components
```

2.2 Syntax

JSX

```jsx
const StyledComponentName = styled.tagName`
  property1: value1;
  property2: value2;
  ...
`;
```

Within the template literals **()**, we define the styles.

Note

StyledComponentName should always start with a **Capital** letter.

2.3 Importing styled

In general, styled components are placed inside the **styledComponents.js** file.

File: src/styledComponents.js

JSX

```jsx
import styled from "styled-components";
```

styled is an internal utility method that transforms the styling from JavaScript into actual CSS.

2.4 Creating and Exporting Styled Component

File: src/styledComponents.js

```jsx
import styled from "styled-components";

export const Heading = styled.h1`
  color: #0070c1;
  font-family: "Roboto";
`;
```

2.5 Importing and Rendering Styled Component

File: src/App.js

```jsx
import "./App.css";
import { Heading } from "./styledComponents";

const App = () => <Heading>Hello World</Heading>;

export default App;
```

2.6 Adapting based on props

- With styled components we can re-use the styles created.

```jsx
<StyledComponent propName="propValue">...</StyledComponent>
```

Syntax:

```jsx
const StyledComponentName = styled.tagName`
```

```
  property1 : value1;
  property2: ${props => /* access prop value */ };
  ...
`
```

Example:

File: src/App.js

JSX

```jsx
import "./App.css";
import { CustomButton } from "./styledComponents";

const App = () => (
  <>
    <CustomButton type="button" color="#ffffff" bgColor="#0070c1">Click</CustomButton>
    <CustomButton type="button" color="#0070c1" bgColor="#ffffff">Click</CustomButton>
  </>
);
export default App;
```

File: src/styledComponents.js

JSX

```jsx
import styled from "styled-components";

export const CustomButton = styled.button`
  padding: 10px;
  margin-right: 20px;
  font-size: 15px;
  color: ${(props) => props.color};
  border-radius: 4px;
  border: 2px solid #0070c1;
  background-color: ${(props) => props.bgColor};
`;
```

Collapse

- Value passed as prop can be accessed as **props.propName**.

2.7 Modularity of Code

Example:

File: src/App.js

```jsx
<CustomButton type="button" outline={false}>Click</CustomButton>
```

The **color** and **bgColor** are the part of styling, instead of passing them as props, we can passthe type of button to handle styles in **styled components**.

2.8 Conditional Styling using Props

```jsx
Condition ? Expression If True : Expression If False;
```

```jsx
const StyledComponentName = styled.tagName`
  property1 : value1;
  property2: ${(props) => (props.name === someValue ? value2 : value3)};
  ...
`;
```

Example:

File: src/App.js

```jsx
import "./App.css";
import { CustomButton } from "./styledComponents";

const App = () => (
  <>
    <CustomButton type="button" outline={false}>Click</CustomButton>
    <CustomButton type="button" outline={true}>Click</CustomButton>
  </>
);

export default App;
```

Collapse

File: src/styledComponents.js

JSX

```jsx
import styled from "styled-components";

export const CustomButton = styled.button`
  padding: 10px;
  margin-right: 20px;
  font-size: 15px;
  color: ${(props) => (props.outline ? "#0070c1" : "#ffffff")};
  border-radius: 4px;
  border: 2px solid #0070c1;
  background-color: ${(props) => (props.outline ? "#ffffff" : "#0070c1")};
`;
```

Collapse