Class-04: Styling in React

1. Introduction to Styling in React

Styling is an essential part of building React applications to enhance UI/UX. React supports multiple ways to style components, including:

- Inline Styling
- CSS Modules
- Styled Components
- (Optional) Tailwind CSS

2. Inline Styling in React

Definition

Inline styles are applied directly to elements using the `style` attribute in JSX. Unlike traditional HTML, React requires styles to be defined as objects with camelCase properties.

Syntax

```
const headingStyle = {
  color: 'blue',
  fontSize: '24px',
  textAlign: 'center'
};

function InlineStyleExample() {
  return <h1 style={headingStyle}>Hello, Inline Styling!</h1>;
}
```

Advantages

✔ Local scope prevents conflicts.

✔ Easy for dynamic styling based on state/props.

Disadvantages

✖ Not reusable across multiple components.

✖ Lacks support for pseudo-classes like `:hover`.

3. CSS Modules in React

Definition

CSS Modules provide a way to locally scope styles to components, avoiding global CSS conflicts.

Setup

1. Create a CSS file with `.module.css` extension (e.g., `Button.module.css`).
2. Import it into the React component.

Example

Button.module.css

```
.button {
  background-color: #007bff;
  color: white;
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}
```

Button.js

```
import styles from './Button.module.css';

function Button() {
  return <button className={styles.button}>Click Me</button>;
}
```

Advantages

✔ Avoids global scope conflicts.

✔ Better organization and modularity.

Disadvantages

✖ Requires build step (not plain CSS).

✖ Slightly more complex than inline styles.

4. Styled Components

Definition

Styled Components is a popular library that allows writing CSS within JavaScript using tagged template literals.

Installation

```
npm install styled-components
```

Example:

```
import styled from 'styled-components';

const StyledButton = styled.button`
  background-color: #28a745;
  color: white;
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  &:hover {
    background-color: #218838;
  }
`;
function App() {
  return <StyledButton>Click Me</StyledButton>;
}
```

Advantages

✔ Fully encapsulated styles.

✔ Supports dynamic styling via props.

✔ Easy to use with conditionals and themes.

Disadvantages

✖ Requires an additional library.

✖ May impact performance in large applications.

5. (Optional) Tailwind CSS

Tailwind CSS is a utility-first CSS framework that enables rapid UI development using predefined classes.

Installation

```
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init
```

Usage Example

```
function TailwindButton() {
  return <button className="bg-blue-500 text-white px-4 py-2
rounded">Click Me</button>;
}
```

Advantages

✔ Highly customizable.

✔ Reduces CSS file size.

✔ No need to write custom CSS for most cases.

Disadvantages

✖ Requires learning utility classes.

✖ Can lead to long class names in JSX.

6. Summary

Method	Scope	Reusability	Requires Extra Library?
Inline Styling	Local	No	No
CSS Modules	Local	Yes	No
Styled Components	Local	Yes	Yes
Tailwind CSS	Global	Yes	Yes

Each styling approach has its pros and cons, and the best choice depends on the project requirements.

7. Conclusion

React provides multiple ways to handle styling, each with unique advantages. For small projects, inline styles or CSS Modules may suffice, while larger applications benefit from Styled Components or Tailwind CSS. Understanding these approaches allows developers to create efficient and maintainable UIs.