STYLING OPTIONS IN REACT

In React, there are multiple ways to apply styles, each with its advantages. Whether you're applying styles directly within your JSX or using sophisticated libraries, React makes it flexible to suit your project's needs.

INTRODUCTION TO STYLING IN REACT

INTRODUCTION TO STYLING IN REACT

- Inline Styling

INTRODUCTION TO STYLING IN REACT

- Inline Styling
- CSS Modules

INTRODUCTION TO STYLING IN REACT

- Inline Styling
- CSS Modules
- Styling Libraries

INTRODUCTION TO STYLING IN REACT

- Inline Styling
- CSS Modules
- Styling Libraries
    - Styled Components

INTRODUCTION TO STYLING IN REACT

- Inline Styling
- CSS Modules
- Styling Libraries
  - Styled Components
  - Material UI

INTRODUCTION TO STYLING IN REACT

- Inline Styling
- CSS Modules
- Styling Libraries
  - Styled Components
  - Material UI
  - Tailwind CSS

INLINE STYLING

- Definition: Applying styles directly within JSX using the style attribute.
- Example:

```
const Counter = () => {
  return <div style={{
    margin: 10,
    padding: 20,
    border: '1px solid red'
  }}>
    Hello, counter
  </div>
}
```

INLINE STYLING

Pros and Cons:

- Quick and easy for simple styles
- Limited scalability and reusability

CSS MODULES

- Definition: A CSS file in which all class names and animation names are scoped locally by default.
- Code example:

```css
/* Button.module.css */
.myButton {
  padding: 10px 15px;
  border: none;
  background-color: blue;
  color: white;
  border-radius: 5px;
}
```

```jsx
// Button.jsx
import { FC } from 'react';
import styles from './Button.module.css';
const ButtonProps = {
  text: string;
}
const Button: FC<ButtonProps> = ({text}) => {
  return <button className="styles.myButton">{text}</button>;
}
```

CSS MODULES

Pros and Cons:

- Scoped styles prevent conflicts
- More maintainable than inline styles

STYLING LIBRARIES

STYLED COMPONENTS

- Utilize tagged template literals to style your components
- Code example:

```
1  import styled from 'styled-components';
2  import { FC } from 'react';
3
4  const ButtonProps = {
5    text: string;
6  }
7  const StyledButton = styled.button`
8    background-color: purple;
9    color: white;
10   padding: 8px 12px;
11   border-radius: 5px;
12   border: none;
13 `;
14
15 export const Button: FC<ButtonProps> = ({text}) => {
```

MATERIAL UI

- A popular React UI framework that follows Google's Material Design.
- Code example:

```
1 import Button from '@mui/material/Button';
2
3 const App = () => {
4   return <Button variant="contained" color="primary">Hello World</
5 }
```

TAILWIND CSS

- One of my favorite libaries. A utility-first CSS framework packed with classes that can be composed to build any design, directly in your markup.
- Code example:

```
import { FC } from 'react';

const ButtonProps = {
  text: string;
}
export const Button: FC<ButtonProps> = ({text}) => {
  return <button class="h-10 px-6 font-semibold rounded-md bg-bla
    {text}
  </button>
}
```