**REACT**

**WEEK-6 HANDSON-5 EXPLANATION**

**Comprehending Why Styling React Components is Important**

Styling in React becomes equally important for:

User Interface Design: It makes your application visually appealing and is user friendly.

User Experience: Makes reading, navigation and usability much better based on your application.

Consistency: Design elements like buttons, cards and text get to be consistent across the involve components.

Component Reusability: It makes good style that self-contained components will not have to rewrite CSS for reuse.

Branding: It makes it possible for a company or product to input specific colors, fonts and layouts.

Without styling, the React component is just plain HTML - it works, but it's not attractive visually.

**Using CSS Module and Inline Styles**

React supports different styles in components. Some of them are:

CSS Modules

CSS Modules are used for locally scoping CSS to a component without conflicting styles that may be found in large applications.

Steps to use CSS Modules:

Create a CSS file such as ComponentName.module.css

Import and reference class names using a styles object.

Example of CSS file (Button.module.css):

.button {

background-color: blue;

color: white;

padding: 10px;

}

Example usage in React(Button.js):

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

function Button() {

return <button className={styles.button}>Click Me</button>

}

Local scope ensures this style of button will not affect any other button class elsewhere

Inline Styles

Inline styles are made as a JavaScript object and attributed to style prop directly.

Example:

function Card() {

const cardStyle = {

border: '1px solid #ccc',

padding: '20px',

borderRadius: '8px',

backgroundColor: '#f9f9f9'

}

return <div style={cardStyle}>This is a card</div>

}

Advantages of inline styles:

Great for dynamic styling based on props or state

Doesn't require a separate CSS file

Limitations:

No pseudo-selectors such as :hover or :focus

No media queries

Styles are not reusable across components

Comparison

CSS Modules

Scope - Local and modular

Syntax - Similar to regular CSS

Reusability - High

Media Queries - Supported

Use Case - General component styling

Inline Styles

Scope - Local per element

Syntax - JavaScript object

Reusability - Low

Media Queries - Not supported

Use Case - Conditional or dynamic styling