# Day 13 — CSS in React: Inline Styles, CSS Modules, Styled-Components

## What this lesson covers (quick map)

1. Why styling matters in React
2. Global CSS / external stylesheets
3. Inline styles (React style prop)
4. CSS Modules (scoped CSS)
5. CSS-in-JS — styled-components (component-scoped styles with JS)
6. When to use which approach (guidelines)
7. Accessibility & performance tips
8. Small 15–20 minute exercise

## 1) Why styling matters in React

* Styling controls look & feel, layout, and responsive behavior.
* React separates UI logic (JSX + state) from presentation, but modern patterns let you colocate styles with components.
* Main goals: **scoping** (avoid global name collisions), **reusability**, **predictability**, and **performance**.

## 2) Global CSS / External Stylesheets

* Traditional approach: .css files (e.g., App.css) imported once (usually in index.js or App.js).
* Use for global site layout, resets, and utilities.

**How to use**

/\* src/App.css \*/  
:root { --theme: #2563eb; }  
body { margin: 0; font-family: Arial, sans-serif; }  
.header { background: var(--theme); color: #fff; padding: 12px; }  
.card { border: 1px solid #eee; padding: 12px; border-radius: 8px; }

// src/App.js  
import './App.css'; // import global CSS  
  
function App(){  
 return <div className="header">My App</div>  
}

**Notes:** - Use global CSS for site-wide patterns. Don’t put component-specific styles here if you want isolation. - Class names are strings via className in JSX.

## 3) Inline styles (React style prop)

* Useful for dynamic styles that depend on JS values (widths, colors computed at runtime).
* Uses an **object** where CSS properties are camelCased.

**Example**

function InlineBox({ size }){  
 // `size` is a number passed as prop  
 const boxStyle = {  
 width: size, // if numeric, React treats it as px only for certain properties  
 height: size,  
 backgroundColor: '#f3f4f6', // camelCase  
 display: 'flex',  
 alignItems: 'center'  
 };  
  
 return <div style={boxStyle}>Box</div>;  
}

**Important details / comments** - style={{ fontSize: 14 }} → React will output font-size: 14px (numbers become px for most properties). - Use inline only for **dynamic or one-off** values. Avoid large blocks of inline styles as they are harder to maintain and not cached by the browser.

## 4) CSS Modules (recommended for component-scoped CSS)

* CSS Modules scope CSS to a component, preventing name collisions.
* File naming: Component.module.css.
* Import like import styles from './ProductCard.module.css' and use className={styles.card}.

**Files**

/\* src/ProductCard.module.css \*/  
.card {  
 border: 1px solid #e5e7eb;  
 padding: 16px;  
 border-radius: 10px;  
 transition: transform 120ms ease;  
}  
.card:hover { transform: translateY(-4px); }  
.title { font-size: 20px; margin-bottom: 8px; }  
.price { color: #0f766e; font-weight: bold; }

// src/ProductCard.js  
import React from 'react';  
import styles from './ProductCard.module.css'; // import module (object)  
  
function ProductCard({ title, price }){  
 return (  
 <div className={styles.card}> {/\* scoped class name \*/}  
 <div className={styles.title}>{title}</div>  
 <div className={styles.price}>${price}</div>  
 </div>  
 );  
}  
  
export default ProductCard;

**Why use CSS Modules?** - Local scope by default — no global collisions. - Familiar CSS syntax (no learning new API). - Works well in most React setups (CRA, Vite) out-of-the-box.

**Dynamic classes**

<div className={`${styles.card} ${isHighlighted ? styles.active : ''}`}>...</div>

## 5) styled-components (CSS-in-JS)

* A library that lets you create styled React components using tagged template literals.
* Advantages: theming, props-driven styles, colocated styles with component logic.

**Install**

npm install styled-components

**Basic example**

// src/ProductStyled.js  
import styled from 'styled-components';  
  
const Card = styled.div`  
 border: 1px solid #eee;  
 padding: 16px;  
 border-radius: 10px;  
 background: ${props => props.primary ? '#f0f9ff' : '#fff'}; /\* style based on prop \*/  
`;  
  
const Title = styled.h3`  
 font-size: 18px;  
`;  
  
export default function ProductStyled({ title, price, primary }){  
 return (  
 <Card primary={primary}>  
 <Title>{title}</Title>  
 <div>${price}</div>  
 </Card>  
 )  
}

**Notes / comments** - styled-components injects CSS at runtime and generates unique class names, preventing collisions. - You can animate, theme, and use shared style fragments. - Slight runtime cost (negligible for many apps), consider server-side rendering needs.

## 6) When to use each approach (rules of thumb)

* **Global CSS**: reset, layout grid, typography, utility classes.
* **CSS Modules**: component-level styles with classic CSS — best for medium/large apps wanting encapsulation without extra runtime.
* **Inline styles**: one-off dynamic styles (width, transform, color from props) or styles computed at runtime.
* **styled-components**: great when you need theming, props-driven styling, or to build a component library.

## 7) Accessibility & Performance Tips

* Keep CSS separate from logic when possible — easier to debug.
* Use :focus styles for keyboard accessibility.
* Avoid too many inline styles that prevent browser optimizations.
* For large lists, prefer CSS classes over inline styles for hover/animations.

## 8) Quick Examples & Comments (copy-paste ready)

### 8.1 Inline dynamic style with state

function ColorBox(){  
 const [color, setColor] = React.useState('#f3f4f6');  
 // Clicking button changes color dynamically  
 return (  
 <div>  
 <div style={{ width: 120, height: 80, backgroundColor: color }} />  
 <button onClick={() => setColor('#fde68a')}>Yellow</button>  
 </div>  
 );  
}

### 8.2 CSS Module (files)

/\* ProductCard.module.css \*/  
.card { padding: 12px; border-radius: 6px; box-shadow: 0 3px 6px rgba(0,0,0,0.06);}

// ProductCard.js  
import styles from './ProductCard.module.css';  
export default function(){ return <div className={styles.card}>Hello</div> }

### 8.3 styled-components example with prop-based style

import styled from 'styled-components';  
  
const Badge = styled.span`  
 padding: 6px 10px;  
 border-radius: 8px;  
 background: ${p => p.success ? '#ecfccb' : '#fee2e2'};  
`;  
  
// usage: <Badge success>In Stock</Badge>

## 9) 15–20 Minute Exercise (hands-on)

**Goal:** Create a ProductCard component and style it two ways.

**Part A (required, 15 min): CSS Modules** 1. Create ProductCard.module.css with classes: .card, .title, .price, .badge. 2. Create ProductCard.js that imports the module and uses className={styles.card} etc. 3. Props: title, price, inStock. 4. Display a green badge In Stock when inStock is true, otherwise red Out of stock. 5. Add a hover effect and responsive behavior (stacked on narrow screens).

**Part B (bonus, 5–10 min): styled-components** 1. Install styled-components. 2. Create a ProductCardStyled.js showing the same UI using styled-components. 3. Use a primary prop to change background color.

If you want, I can also generate starter files you can copy-paste into your project (component + css module + styled-components version). Say the word and I’ll create them.