React + Vite Course (Kids 13–14)

Module 6 — Styling React Components (80 minutes)

Audience: Kids 13–14
This module builds directly on Modules 1–5.
All code stays in the same project.
Today's work goes into `src/day6/`.

Learning Objectives

By the end of this module, students will:

1. Understand how styling works in React
2. Know the difference between `class` and `className`
3. Style components using normal CSS files
4. Use inline styles correctly
5. Understand and use CSS Modules
6. Get a first idea of styled-components
7. Build clean and readable UI components
8. Review full homework solutions

80-Minute Agenda

- **0–10 min** — Recap (Modules 1–5)
- **10–20 min** — Styling in React (concepts)
- **20–35 min** — CSS classes & inline styles
- **35–55 min** — CSS Modules (main focus)
- **55–65 min** — Intro to styled-components
- **65–75 min** — Hands-on practice
- **75–80 min** — Homework + recap

Recap from Module 5 (0–10 min)

Ask students:

- What is a component lifecycle?
- What does `useEffect` do?
- What is a side-effect?
- Why do we need cleanup?

Reminder: State changes cause React to re-render the UI.

How Styling Works in React (10–20 min)

Important Difference from HTML

HTML:

```
<div class="box"></div>
```

React (JSX):

```
<div className="box"></div>
```

Why `className`?

- JSX is JavaScript
- `class` is a JavaScript keyword
- React avoids conflicts

Teaching line: *"JSX looks like HTML, but it follows JavaScript rules."*

Styling Method 1: CSS Files (20–30 min)

Create CSS File

`src/day6/box.css`

```
.box {
  background-color: #4f46e5;
  color: white;
  padding: 20px;
  border-radius: 10px;
  margin-bottom: 15px;
}
```

Component

`src/day6/StyledBox.jsx`

```jsx
import './box.css';

function StyledBox() {
  return (
    <div className="box">
      <h2>Styled with CSS</h2>
      <p>This box uses a normal CSS file.</p>
    </div>
  );
}

export default StyledBox;
```

Teaching Notes

- CSS is global
- Easy to use
- Can cause conflicts in large apps

Styling Method 2: Inline Styles (30–35 min)

Example

`src/day6/InlineBox.jsx`

```jsx
function InlineBox() {
  const style = {
    backgroundColor: 'orange',
    padding: '20px',
    borderRadius: '10px',
    color: 'white'
  };

  return (
    <div style={style}>
      <h2>Inline Styles</h2>
      <p>Styles written inside JavaScript.</p>
    </div>
  );
}

export default InlineBox;
```

Rules

- Styles are objects
- Use camelCase
- Values are strings

Inline styles are good for small, quick styling.

Styling Method 3: CSS Modules (35–55 min)

Why CSS Modules?

- Styles are scoped
- No class name conflicts
- Best default choice in React

Create Module

`src/day6/Card.module.css`

```
.card {
  background-color: #16a34a;
  color: white;
  padding: 20px;
  border-radius: 12px;
}

.title {
  font-size: 22px;
}
```

Component

`src/day6/Card.jsx`

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

function Card() {
  return (
    <div className={styles.card}>
      <h2 className={styles.title}>CSS Module Card</h2>
      <p>This style is safe.</p>
    </div>
  );
```

```
}

export default Card;
```

Teaching line: *"CSS Modules protect your styles."*

Intro to Styled-Components (55–65 min)

- Styles written in JavaScript
- Used in advanced React apps
- Not required now

Example (concept only):

```
const Box = styled.div`
  background: purple;
  padding: 20px;
`;
```

Hands-On Practice (65–75 min)

Profile Component

`src/day6/Profile.jsx`

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

function Profile() {
  return (
    <div className={styles.profile}>
      <h2 className={styles.name}>John Doe</h2>
      <p className={styles.age}>Age: 14</p>
    </div>
  );
}

export default Profile;
```

`Profile.module.css`

```css
.profile {
  background-color: #0ea5e9;
  color: white;
  padding: 20px;
  border-radius: 10px;
}

.name { font-size: 24px; }
.age { font-size: 18px; }
```

Homework (Module 6)

1. Create a **Button** component using CSS Modules
2. Create a **Product Card** with name, price, and Buy button

---

Homework — FULL SOLUTIONS

Button

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

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

export default Button;
```

```css
.btn {
  background-color: #2563eb;
  color: white;
  padding: 10px 18px;
  border: none;
  border-radius: 8px;
}

.btn:hover {
  background-color: #1e40af;
}
```

Product

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

function Product() {
  return (
    <div className={styles.product}>
      <h2>Headphones</h2>
      <p>$29.99</p>
      <button>Buy</button>
    </div>
  );
}

export default Product;
```

```
.product {
  border: 2px solid #ddd;
  padding: 20px;
  border-radius: 10px;
}
```

Final Recap

- `className` replaces `class`
- Inline styles are limited
- CSS Modules are the best default
- Clean UI builds trust

Ready for Module 7 — Context API & State Sharing