Tài liệu: Các loại CSS và công nghệ liên quan

Mục tiêu

- Hiểu rõ các loại CSS khác nhau, bao gồm CSS gốc, các preprocessor (SCSS, Sass, Less, Stylus), và các phương pháp hiện đại (CSS-in-JS, CSS Modules).
- Nắm được cách hoạt động, cú pháp, ưu/nhược điểm, và ứng dụng thực tế của từng loại.
- Cung cấp ví dụ minh họa để người học dễ dàng áp dụng vào dự án thực tế.

Đối tượng

- Sinh viên hoặc lập trình viên mới học về CSS hoặc muốn tìm hiểu các công cụ mở rộng.
- Người muốn so sánh các công nghệ CSS để chọn giải pháp phù hợp cho dự án.

---

1. CSS (Cascading Style Sheets)

Định nghĩa

CSS là ngôn ngữ chính thức để định dạng và tạo kiểu cho các tài liệu HTML, được sử dụng để kiểm soát giao diện, bố cục, màu sắc, và hiệu ứng của trang web.

Cách hoạt động

- **Cơ chế**: CSS sử dụng các quy tắc (rules) gồm selector và declaration block để áp dụng kiểu dáng cho các phần tử HTML. Trình duyệt phân tích CSS và render giao diện tương ứng.
- **Nhúng CSS**: Có 3 cách nhúng:
    1. **Inline**: Sử dụng thuộc tính `style` trong thẻ HTML.
    2. **Internal**: Sử dụng thẻ `<style>` trong `<head>`.
    3. **External**: Liên kết file `.css` qua thẻ `<link>`.
- **Cơ chế kế thừa và ưu tiên**: CSS áp dụng quy tắc kế thừa (inheritance) và tính ưu tiên (specificity) để xác định kiểu dáng nào được áp dụng khi có xung đột.

Cú pháp

```
selector {
  property: value;
}
```

Ví dụ minh họa

Tạo kiểu cho một đoạn văn bản và tiêu đề.

```
<!DOCTYPE html>
<html lang="vi">
<head>
```

```
  <meta charset="UTF-8">
  <title>CSS Example</title>
  <style>
    h1 {
      color: navy;
      font-size: 2rem;
    }
    p {
      background-color: #f0f0f0;
      padding: 10px;
    }
  </style>
</head>
<body>
  <h1>Tiêu đề</h1>
  <p>Đây là một đoạn văn bản được định dạng bằng CSS.</p>
</body>
</html>
```

Ưu điểm

- **Chuẩn web**: Được hỗ trợ bởi tất cả trình duyệt, không cần công cụ bổ sung.
- **Đơn giản**: Dễ học cho người mới bắt đầu.
- **Hiệu suất cao**: Không cần biên dịch, render trực tiếp.

Nhược điểm

- **Hạn chế về tính năng**: Thiếu các tính năng nâng cao như biến, lồng (nesting), hoặc hàm.
- **Khó bảo trì**: Trong dự án lớn, mã CSS có thể trở nên dài và khó quản lý nếu không có cấu trúc tốt.
- **Không hỗ trợ logic**: Không có vòng lặp, điều kiện, hoặc module hóa.

Ứng dụng thực tế

- **Website tĩnh**: Các trang web nhỏ, đơn giản không cần tính năng phức tạp.
- **Học tập**: Phù hợp cho người mới học lập trình web.
- **Dự án không yêu cầu preprocessor**: Các dự án ưu tiên hiệu suất và không muốn thêm công cụ.

---

2. SCSS (Sassy CSS)

Định nghĩa

SCSS là một phiên bản của Sass (Syntactically Awesome Style Sheets), một CSS preprocessor, sử dụng cú pháp tương tự CSS nhưng bổ sung nhiều tính năng mạnh mẽ như biến, lồng, mixin, và kế thừa.

Cách hoạt động

- **Preprocessor**: SCSS là một siêu tập (superset) của CSS, nghĩa là mọi mã CSS hợp lệ đều hoạt động trong SCSS. File SCSS được biên dịch thành CSS thông qua công cụ như `sass` hoặc các build tool (Webpack, Vite).
- **Tính năng**:
  - **Biến**: Lưu trữ giá trị để tái sử dụng.
  - **Lồng (Nesting)**: Viết CSS lồng nhau để tăng tính đọc.
  - **Mixin**: Tạo khối mã tái sử dụng.
  - **Kế thừa (@extend)**: Chia sẻ kiểu dáng giữa các selector.
- **Cơ chế**: File `.scss` được biên dịch thành file `.css` để trình duyệt hiểu.

Cú pháp

```
$primary-color: #007bff;

.container {
  padding: 20px;
  .title {
    color: $primary-color;
    font-size: 1.5rem;
  }
}

@mixin button-style {
  padding: 10px 20px;
  border-radius: 5px;
}
.button {
  @include button-style;
  background-color: $primary-color;
}
```

Ví dụ minh họa

Tạo một card sử dụng SCSS với biến và lồng.

```
<!DOCTYPE html>
<html lang="vi">
<head>
  <meta charset="UTF-8">
  <title>SCSS Example</title>
  <link rel="stylesheet" href="styles.css"> <!-- File CSS đã biên dịch từ
SCSS -->
</head>
<body>
  <div class="container">
    <h2 class="title">Tiêu đề</h2>
    <button class="button">Click me</button>
```

```
  </div>
</body>
</html>
```

styles.scss:

```
$primary-color: #007bff;

.container {
  padding: 20px;
  background-color: #f0f0f0;
  .title {
    color: $primary-color;
    font-size: 1.5rem;
  }
}

@mixin button-style {
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
}
.button {
  @include button-style;
  background-color: $primary-color;
  color: white;
  &:hover {
    background-color: darken($primary-color, 10%);
  }
}
```

Lưu ý: File `styles.scss` cần được biên dịch thành `styles.css` bằng công cụ như `sass` (`sass styles.scss styles.css`).

Ưu điểm

- **Tính năng mạnh mẽ**: Biến, lồng, mixin, và hàm giúp viết mã ngắn gọn và dễ bảo trì.
- **Tương thích với CSS**: Mọi mã CSS đều hợp lệ trong SCSS.
- **Cộng đồng lớn**: Sass/SCSS được sử dụng rộng rãi, có nhiều tài liệu và thư viện hỗ trợ.

Nhược điểm

- **Cần biên dịch**: Yêu cầu công cụ biên dịch, làm phức tạp quy trình phát triển.
- **Khúc học tập**: Người mới cần thời gian làm quen với các tính năng như mixin, extend.
- **Hiệu suất**: Quá lạm dụng lồng có thể tạo ra CSS dài, ảnh hưởng hiệu suất.

Ứng dụng thực tế

- **Dự án lớn**: Sử dụng trong các dự án cần quản lý mã CSS phức tạp, như trang thương mại điện tử hoặc dashboard.
- **Design systems**: Xây dựng thư viện giao diện với các biến màu sắc, kích thước.
- **Framework integration**: Tích hợp với Bootstrap, Foundation để tùy chỉnh giao diện.

---

3. Sass

Định nghĩa

Sass là một CSS preprocessor, cung cấp hai cú pháp: **SCSS** (đã đề cập) và **Sass** (cú pháp gốc, sử dụng thụt lề). Cú pháp Sass gọn hơn, không dùng dấu `{}` và `;`.

Cách hoạt động

- **Cú pháp thụt lề**: Sass sử dụng thụt lề (indentation) để xác định khối, thay vì dấu `{}` như CSS/SCSS.
- **Tính năng**: Tương tự SCSS (biến, lồng, mixin, extend), nhưng cú pháp ngắn gọn hơn.
- **Biên dịch**: File `.sass` được biên dịch thành CSS bằng công cụ `sass`.
- **Cơ chế**: Trình duyệt không hiểu cú pháp Sass, nên cần biên dịch thành CSS trước khi sử dụng.

Cú pháp

```
$primary-color: #007bff

.container
  padding: 20px
  .title
    color: $primary-color
    font-size: 1.5rem

@mixin button-style
  padding: 10px 20px
  border-radius: 5px

.button
  @include button-style
  background-color: $primary-color
```

Ví dụ minh họa

Tạo một card sử dụng cú pháp Sass.

```
<!DOCTYPE html>
<html lang="vi">
<head>
  <meta charset="UTF-8">
```

```
  <title>Sass Example</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <h2 class="title">Tiêu đề</h2>
    <button class="button">Click me</button>
  </div>
</body>
</html>
```

styles.sass:

```
$primary-color: #007bff

.container
  padding: 20px
  background-color: #f0f0f0
  .title
    color: $primary-color
    font-size: 1.5rem

@mixin button-style
  padding: 10px 20px
  border: none
  border-radius: 5px

.button
  @include button-style
  background-color: $primary-color
  color: white
  &:hover
    background-color: darken($primary-color, 10%)
```

Lưu ý: Biên dịch `styles.sass` thành `styles.css` bằng lệnh `sass styles.sass styles.css`.

Ưu điểm

- **Cú pháp ngắn gọn**: Loại bỏ `{}` và `;`, giúp mã sạch hơn.
- **Tính năng mạnh mẽ**: Tương tự SCSS, hỗ trợ biến, lồng, mixin, v.v.
- **Tương thích với SCSS**: Có thể chuyển đổi dễ dàng giữa Sass và SCSS.

Nhược điểm

- **Khác biệt cú pháp**: Không giống CSS, yêu cầu học cách viết thụt lề.
- **Ít phổ biến hơn SCSS**: Cộng đồng sử dụng SCSS nhiều hơn, tài liệu cho cú pháp Sass ít hơn.
- **Cần biên dịch**: Tương tự SCSS, yêu cầu công cụ biên dịch.

Ứng dụng thực tế

- **Dự án cá nhân**: Phù hợp với lập trình viên thích cú pháp ngắn gọn, không rườm rà.
- **Dự án nhỏ**: Sử dụng khi không cần tương thích trực tiếp với CSS.
- **Tùy chỉnh framework**: Tùy chỉnh các framework như Bootstrap bằng Sass.

---

4. Less

Định nghĩa

Less là một CSS preprocessor, cung cấp các tính năng như biến, lồng, mixin, và hàm, tương tự Sass, nhưng có cú pháp gần với CSS hơn và dễ học hơn.

Cách hoạt động

- **Preprocessor**: File `.less` được biên dịch thành CSS bằng công cụ như `lessc` hoặc build tool.
- **Tính năng**:
  - **Biến**: Lưu trữ giá trị (`@variable`).
  - **Lồng**: Viết CSS lồng nhau.
  - **Mixin**: Tạo khối mã tái sử dụng.
  - **Hàm**: Hỗ trợ các hàm như `darken`, `lighten`.
- **Cơ chế**: Less được biên dịch thành CSS, nhưng có thể chạy trực tiếp trong trình duyệt bằng `less.js` (không khuyến khích trong sản xuất).

Cú pháp

```
@primary-color: #007bff;

.container {
  padding: 20px;
  .title {
    color: @primary-color;
    font-size: 1.5rem;
  }
}

.button-style() {
  padding: 10px 20px;
  border-radius: 5px;
}
.button {
  .button-style();
  background-color: @primary-color;
}
```

Ví dụ minh họa

Tạo một card sử dụng Less.

```
<!DOCTYPE html>
<html lang="vi">
<head>
  <meta charset="UTF-8">
  <title>Less Example</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <h2 class="title">Tiêu đề</h2>
    <button class="button">Click me</button>
  </div>
</body>
</html>
```

styles.less:

```
@primary-color: #007bff;

.container {
  padding: 20px;
  background-color: #f0f0f0;
  .title {
    color: @primary-color;
    font-size: 1.5rem;
  }
}

.button-style() {
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
}
.button {
  .button-style();
  background-color: @primary-color;
  color: white;
  &:hover {
    background-color: darken(@primary-color, 10%);
  }
}
```

Lưu ý: Biên dịch `styles.less` thành `styles.css` bằng lệnh `lessc styles.less styles.css`.

Ưu điểm

- **Dễ học**: Cú pháp gần giống CSS, phù hợp cho người mới.
- **Nhẹ**: Less có kích thước nhỏ hơn Sass, dễ tích hợp.
- **Hỗ trợ trình duyệt**: Có thể chạy trực tiếp bằng `less.js` (dù không khuyến khích).

Nhược điểm

- **Ít tính năng hơn Sass**: Thiếu một số tính năng nâng cao như vòng lặp phức tạp.
- **Cộng đồng nhỏ hơn**: Ít được sử dụng hơn Sass/SCSS.
- **Hiệu suất**: Chạy `less.js` trong trình duyệt làm chậm hiệu suất.

Ứng dụng thực tế

- **Dự án nhỏ**: Phù hợp cho các dự án không cần tính năng phức tạp của Sass.
- **Tùy chỉnh Bootstrap**: Bootstrap sử dụng Less (trong các phiên bản cũ), dễ tùy chỉnh.
- **Học tập**: Dùng để làm quen với preprocessor trước khi học Sass.

---

5. Stylus

Định nghĩa

Stylus là một CSS preprocessor mạnh mẽ, linh hoạt, hỗ trợ cả cú pháp giống CSS và cú pháp thụt lề (tương tự Sass), với các tính năng như biến, lồng, mixin, và hàm.

Cách hoạt động

- **Cú pháp linh hoạt**: Cho phép viết mã với `{}` và `;` (giống CSS) hoặc thụt lề (giống Sass).
- **Tính năng**: Hỗ trợ biến, lồng, mixin, hàm, và cả các biểu thức logic như điều kiện, vòng lặp.
- **Cơ chế**: File `.styl` được biên dịch thành CSS bằng công cụ `stylus`. Stylus rất linh hoạt, cho phép bỏ qua dấu `:` hoặc `;` trong một số trường hợp.
- **Lưu ý**: Stylus ít phổ biến hơn Sass, nhưng mạnh mẽ trong các dự án cần tính linh hoạt cao.

Cú pháp

```
$primary-color = #007bff

.container
  padding 20px
  .title
    color $primary-color
    font-size 1.5rem

button-style()
  padding 10px 20px
```

```
  border-radius 5px

.button
  button-style()
  background-color $primary-color
```

Ví dụ minh họa

Tạo một card sử dụng Stylus.

```
<!DOCTYPE html>
<html lang="vi">
<head>
  <meta charset="UTF-8">
  <title>Stylus Example</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <h2 class="title">Tiêu đề</h2>
    <button class="button">Click me</button>
  </div>
</body>
</html>
```

styles.styl:

```
$primary-color = #007bff

.container
  padding 20px
  background-color #f0f0f0
  .title
    color $primary-color
    font-size 1.5rem

button-style()
  padding 10px 20px
  border none
  border-radius 5px

.button
  button-style()
  background-color $primary-color
  color white
  &:hover
    background-color darken($primary-color, 10%)
```

Lưu ý: Biên dịch `styles.styl` thành `styles.css` bằng lệnh `stylus styles.styl -o styles.css`.

Ưu điểm

- **Linh hoạt**: Hỗ trợ nhiều cú pháp, phù hợp với nhiều phong cách viết mã.
- **Mạnh mẽ**: Có các tính năng logic (if, for) mạnh hơn Less.
- **Nhẹ**: Kích thước nhỏ, dễ tích hợp vào dự án.

Nhược điểm

- **Cộng đồng nhỏ**: Ít được sử dụng hơn Sass/SCSS, tài liệu hạn chế.
- **Cần biên dịch**: Yêu cầu công cụ biên dịch, tương tự Sass.
- **Khúc học tập**: Cú pháp linh hoạt có thể gây nhầm lẫn cho người mới.

Ứng dụng thực tế

- **Dự án sáng tạo**: Phù hợp cho các dự án cần tùy chỉnh cú pháp linh hoạt.
- **Dự án cá nhân**: Lập trình viên muốn thử nghiệm preprocessor mới.
- **Tích hợp với Node.js**: Dễ dàng tích hợp trong các dự án sử dụng Node.js.

---

6. CSS-in-JS

Định nghĩa

CSS-in-JS là cách viết CSS trong JavaScript, sử dụng các thư viện như **Styled-Components**, **Emotion**, hoặc **JSS** để quản lý kiểu dáng theo component, thường dùng trong các framework như React.

Cách hoạt động

- **Kiểu dáng trong JS**: CSS được viết trong file JavaScript, gắn với component cụ thể.
- **Scope cục bộ**: Mỗi component có kiểu dáng riêng, tránh xung đột tên class.
- **Cơ chế**: Thư viện CSS-in-JS tạo ra các class duy nhất khi render, thường sử dụng template literals (`` ` ``) để viết CSS. Kiểu dáng có thể thay đổi động dựa trên props hoặc state.
- **Lưu ý**:
    - Phù hợp với các ứng dụng component-based.
    - Có thể làm tăng thời gian tải nếu không tối ưu (server-side rendering giúp cải thiện).

Cú pháp

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

const StyledDiv = styled.div`
  background-color: #e0f7fa;
  padding: 20px;
```

```
  border-radius: 8px;
`;
```

Ví dụ minh họa

Tạo một card với Styled-Components.

```
<!DOCTYPE html>
<html lang="vi">
<head>
  <meta charset="UTF-8">
  <title>CSS-in-JS Example</title>
  <script
src="https://cdn.jsdelivr.net/npm/react@17/umd/react.development.js"></scr
ipt>
  <script src="https://cdn.jsdelivr.net/npm/react-dom@17/umd/react-
dom.development.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/styled-
components@5/dist/styled-components.min.js"></script>
</head>
<body>
  <div id="root"></div>
  <script>
    const { styled } = window.styled;
    const Card = styled.div`
      background-color: #e0f7fa;
      padding: 20px;
      border-radius: 8px;
      text-align: center;
      &:hover {
        background-color: #b2ebf2;
      }
    `;
    ReactDOM.render(<Card>Card CSS-in-JS</Card>,
document.getElementById('root'));
  </script>
</body>
</html>
```

Ưu điểm

- **Scope cục bộ**: Tránh xung đột tên class, lý tưởng cho ứng dụng lớn.
- **Dynamic styling**: Thay đổi kiểu dáng dựa trên props/state.
- **Tích hợp với JS**: Dễ dàng tích hợp với React, Vue, hoặc các framework khác.

Nhược điểm

- **Phụ thuộc vào JS**: Không hoạt động nếu JavaScript bị tắt.
- **Hiệu suất**: Có thể làm tăng thời gian tải nếu không tối ưu.
- **Khúc học tập**: Yêu cầu quen thuộc với JavaScript và framework.

Ứng dụng thực tế

- **React/Vue apps**: Quản lý kiểu dáng cho các component độc lập.
- **Single-page applications**: Tăng tốc phát triển với kiểu dáng gắn liền component.
- **Dynamic UI**: Tạo giao diện thay đổi theo dữ liệu người dùng (ví dụ: theme switching).

---

7. CSS Modules

Định nghĩa

CSS Modules là một phương pháp quản lý CSS, đảm bảo scope cục bộ cho các class bằng cách tạo tên class duy nhất trong quá trình build, thường được sử dụng với JavaScript frameworks.

Cách hoạt động

- **Scope cục bộ**: Mỗi file CSS được xem như một module, các class được tự động đổi tên thành duy nhất (ví dụ: `className` thành `className__[hash]`).
- **Tích hợp với build tool**: CSS Modules hoạt động với Webpack, Vite, hoặc các bundler khác để biên dịch CSS.
- **Cơ chế**: File `.module.css` được import vào JavaScript, và các class được truy cập dưới dạng object. Trình duyệt chỉ thấy CSS đã biên dịch với tên class duy nhất.
- **Lưu ý**: CSS Modules vẫn là CSS, không yêu cầu cú pháp mới, nhưng cần cấu hình build tool.

Cú pháp

```
/* styles.module.css */
.container {
  padding: 20px;
  background-color: #f0f0f0;
}
.title {
  color: #007bff;
  font-size: 1.5rem;
}
import styles from './styles.module.css';

<div className={styles.container}>
  <h2 className={styles.title}>Tiêu đề</h2>
</div>
```

Ví dụ minh họa

Tạo một card sử dụng CSS Modules với React.

```
<!DOCTYPE html>
<html lang="vi">
<head>
  <meta charset="UTF-8">
  <title>CSS Modules Example</title>
  <script
src="https://cdn.jsdelivr.net/npm/react@17/umd/react.development.js"></scr
ipt>
  <script src="https://cdn.jsdelivr.net/npm/react-dom@17/umd/react-
dom.development.js"></script>
</head>
<body>
  <div id="root"></div>
  <script type="module">
    const styles = {
      container: 'container__xyz123',
      title: 'title__abc456'
    };
    // Giả lập CSS Modules (thực tế cần Webpack/Vite)
    const styleSheet = document.createElement('style');
    styleSheet.innerText = `
      .${styles.container} {
        padding: 20px;
        background-color: #f0f0f0;
      }
      .${styles.title} {
        color: #007bff;
        font-size: 1.5rem;
      }
    `;
    document.head.appendChild(styleSheet);
    ReactDOM.render(
      <div className={styles.container}>
        <h2 className={styles.title}>Tiêu đề</h2>
        <p>Nội dung card</p>
      </div>,
      document.getElementById('root')
    );
  </script>
</body>
</html>
```

Lưu ý: Ví dụ trên giả lập CSS Modules mà không cần build tool. Trong thực tế, cần cấu hình Webpack/Vite để import file `.module.css`.

Ưu điểm

- **Scope cục bộ**: Tránh xung đột tên class, lý tưởng cho dự án lớn.
- **Tương thích với CSS**: Không yêu cầu học cú pháp mới.
- **Tích hợp với JS**: Dễ sử dụng với React, Vue, hoặc các framework khác.

Nhược điểm

- **Cần build tool**: Yêu cầu cấu hình Webpack, Vite, hoặc tương tự.
- **Phức tạp hơn CSS thông thường**: Thêm bước xử lý trong quy trình phát triển.
- **Hạn chế về dynamic styling**: Không linh hoạt bằng CSS-in-JS khi cần kiểu dáng động.

Ứng dụng thực tế

- **React/Vue apps**: Quản lý kiểu dáng cho các component độc lập.
- **Dự án lớn**: Đảm bảo không xung đột class trong các đội phát triển lớn.
- **Modular development**: Tạo các module CSS tái sử dụng.

So sánh các loại CSS

Công nghệ	Cú pháp	Cần biên dịch	Scope	Tính năng nổi bật	Ứng dụng chính
CSS	Chuẩn CSS	Không	Toàn cục	Đơn giản, chuẩn web	Website tĩnh, dự án nhỏ
SCSS	Giống CSS	Có	Toàn cục	Biến, lồng, mixin	Dự án lớn, design systems
Sass	Thụt lề	Có	Toàn cục	Biến, lồng, cú pháp gọn	Dự án cá nhân, tùy chỉnh
Less	Giống CSS	Có	Toàn cục	Biến, lồng, đơn giản	Dự án nhỏ, tùy chỉnh Bootstrap
Stylus	Linh hoạt	Có	Toàn cục	Biến, lồng, logic	Dự án sáng tạo, Node.js
CSS-in-JS	Trong JS	Có	Cục bộ	Dynamic styling	React/Vue apps
CSS Modules	Chuẩn CSS	Có	Cục bộ	Scope cục bộ	Dự án lớn, modular

---

Tài nguyên bổ sung

- **MDN Web Docs**: Tài liệu về CSS (developer.mozilla.org).
- **Sass Official**: Hướng dẫn SCSS/Sass (sass-lang.com).
- **Less Official**: Tài liệu Less (lesscss.org).
- **Stylus Official**: Tài liệu Stylus (stylus-lang.com).
- **Styled-Components**: Tài liệu CSS-in-JS (styled-components.com).
- **CSS Modules**: Tài liệu CSS Modules (github.com/css-modules).
- **CodePen**: Thực hành và chia sẻ mã (codepen.io).

Bài tập thực hành

1. **CSS**: Tạo một trang HTML với 1 tiêu đề và 1 đoạn văn, định dạng bằng CSS thuần (màu sắc, padding, border).
2. **SCSS**: Viết một file SCSS để tạo một card với biến màu sắc, lồng, và mixin cho nút.
3. **Sass**: Chuyển file SCSS trên thành cú pháp Sass (thụt lề).
4. **Less**: Tạo một menu điều hướng sử dụng Less, với lồng và biến.
5. **Stylus**: Tạo một button với hiệu ứng hover sử dụng Stylus, áp dụng hàm `darken`.
6. **CSS-in-JS**: Tạo một component React với Styled-Components, định dạng một card với hiệu ứng hover.
7. **CSS Modules**: Tạo một card sử dụng CSS Modules (giả lập hoặc dùng Webpack), đảm bảo scope cục bộ.