Chapter 08: Styling trong Next.js

A. Mục tiêu

Sau bài học này, học viên sẽ:

- Hiểu và áp dụng được các phương pháp styling trong dự án Next.js.
- Sử dụng thành thạo **CSS Modules** để tạo style cục bộ cho component.
- Tích hợp và sử dụng **SASS/SCSS** để viết CSS nâng cao với các tính năng như variables, nesting, mixins.
- Cài đặt và sử dụng **Tailwind CSS**, một framework utility-first để phát triển giao diện nhanh chóng và nhất quán.
- So sánh ưu, nhược điểm của các phương pháp styling và lựa chọn phương pháp phù hợp cho từng tình huống.

B. Nội dung lý thuyết

1. Global CSS

Global CSS là cách áp dụng các style chung cho toàn bộ ứng dụng. Trong Next.js, các file CSS toàn cục chỉ được phép import trong file `pages/_app.js`. Điều này giúp đảm bảo rằng các style toàn cục được áp dụng một cách thống nhất và tránh xung đột.

Cách sử dụng:

- Tạo file `styles/globals.css` (hoặc `.scss` nếu dùng SASS).
- Import file trong `pages/_app.js`:

```
import '../styles/globals.css';
```

- Dùng Global CSS để:
  - Định nghĩa các **CSS variables** (ví dụ: `--primary-color`, `--background-color`).
  - Áp dụng các **style reset** (như normalize.css).
  - Thiết lập **typography** hoặc **theme variables** chung cho toàn ứng dụng.

Lưu ý:

- Lạm dụng Global CSS có thể dẫn đến **xung đột class name** hoặc khó bảo trì.
- Chỉ nên sử dụng cho các style áp dụng trên toàn ứng dụng (ví dụ: font mặc định, màu nền, margin/padding cơ bản).

Ví dụ:

```css
/* styles/globals.css */
:root {
  --primary-color: #0070f3;
  --background-color: #f4f4f4;
}

body {
  background-color: var(--background-color);
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
}
```

---

2. CSS Modules

CSS Modules là giải pháp tích hợp sẵn trong Next.js để viết CSS cục bộ (component-scoped). Mỗi file CSS Modules sẽ chỉ áp dụng style cho component tương ứng, tránh xung đột với các component khác.

Cách sử dụng:

- Tạo file CSS với định dạng `[ComponentName].module.css` (ví dụ: `Button.module.css`).
- Import file CSS vào component:

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

- Sử dụng class trong JSX:

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

- Next.js tự động tạo **class name duy nhất** (ví dụ: `Button_primaryButton__aB3d1`) để đảm bảo style không bị rò rỉ ra ngoài.

Ưu điểm:

- **Tính đóng gói**: Style chỉ áp dụng cho component cụ thể.
- **Tránh xung đột**: Không lo trùng tên class với các component khác.
- **Dễ bảo trì**: Mỗi component có file CSS riêng, dễ dàng quản lý.

Ví dụ:

```
/* Button.module.css */
.button {
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

.primary {
  background-color: var(--primary-color);
```

```
  color: white;
}
```

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

export default function Button({ variant }) {
  return (
    <button className={`${styles.button} ${variant === 'primary' ? styles.pr
      Click Me
    </button>
  );
}
```

---

3. SASS/SCSS Pre-processor

Next.js hỗ trợ **SASS/SCSS** ngay từ đầu, cho phép sử dụng các tính năng mạnh mẽ như **variables**, **nesting**, **mixins**, và **modules**. Để sử dụng, bạn cần cài đặt package `sass`:

```
npm install sass
```

Cách sử dụng:

- Đổi tên file `.css` thành `.scss` (ví dụ: `globals.css` → `globals.scss`, `Button.module.css` → `Button.module.scss`).
- Sử dụng các tính năng của SASS như:
  - **Variables**: `$primary-color: #0070f3;`
  - **Nesting**: Gộp các style liên quan trong cùng một khối.
  - **Mixins**: Tái sử dụng các đoạn style.

Ví dụ:

```scss
/* Button.module.scss */
$primary-color: #0070f3;

.button {
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  cursor: pointer;

  &.primary {
    background-color: $primary-color;
    color: white;

    &:hover {
      background-color: darken($primary-color, 10%);
    }
  }
}
```

```jsx
// Button.js
import styles from './Button.module.scss';

export default function Button({ variant }) {
  return (
    <button className={`${styles.button} ${variant === 'primary' ? styles.pr
      Click Me
    </button>
  );
}
```

Ưu điểm:

- Viết CSS ngắn gọn, dễ đọc hơn với nesting và variables.
- Tái sử dụng code qua mixins và modules.

- Tích hợp tốt với cả Global CSS và CSS Modules.

Nhược điểm:

- Cần cài đặt thêm `sass`.
- Có thể làm tăng thời gian build nếu dự án lớn.

---

4. Tailwind CSS (Utility-First Framework)

Tailwind CSS là một framework CSS theo triết lý **utility-first**, cho phép bạn style giao diện trực tiếp trong JSX bằng cách sử dụng các class tiện ích có sẵn.

Triết lý:

- Thay vì viết file CSS riêng, bạn áp dụng style thông qua các class như `p-6`, `bg-white`, `rounded-xl`.
- Ví dụ:

```
<div className="p-6 max-w-sm mx-auto bg-white rounded-xl shadow-lg flex
  <div className="text-xl font-medium text-black">Hello, Tailwind!</div>
</div>
```

Ưu điểm:

- **Tốc độ phát triển nhanh**: Không cần viết CSS từ đầu.
- **Tính nhất quán**: Các class được định nghĩa sẵn, đảm bảo giao diện đồng bộ.
- **File CSS nhỏ**: Tailwind chỉ giữ lại các class được sử dụng nhờ **PurgeCSS**.
- **Không cần đặt tên class**: Loại bỏ vấn đề nghĩ tên class (BEM, SMACSS...).

Nhược điểm:

- JSX có thể trở nên dài dòng nếu sử dụng nhiều class.

- Cần học cú pháp các class của Tailwind.
- Không phù hợp với các dự án cần style tùy chỉnh phức tạp.

Cài đặt Tailwind CSS trong Next.js:

1. Cài đặt các package:

```
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

2. Cấu hình file `tailwind.config.js` :

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    './pages/**/*.{js,ts,jsx,tsx}',
    './components/**/*.{js,ts,jsx,tsx}',
  ],
  theme |: {
    extend: {},
  },
  plugins: [],
};
```

3. Thêm Tailwind vào `styles/globals.css` :

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

4. Khởi động dự án Next.js:

```
npm run dev
```

Ví dụ sử dụng:

```
// UserProfileCard.js
export default function UserProfileCard() {
  return (
    <div className="max-w-sm mx-auto bg-white rounded-xl shadow-lg p-6">
      <img
        className="w-24 h-24 rounded-full mx-auto"
        src="/avatar.jpg"
        alt="Avatar"
      />
      <h2 className="text-xl font-bold text-center mt-4">John Doe</h2>
      <p className="text-gray-600 text-center">
        Frontend Developer with a passion for React and Next.js
      </p>
    </div>
  );
}
```

C. Bài tập thực hành

Bài 1: Global CSS

1. Tạo file `styles/globals.css`.
2. Định nghĩa các CSS variables:

```
:root {
  --primary-color: #0070f3;
```

```
  --background-color: #f4f4f4;
  --text-color: #333;
}

body {
  background-color: var(--background-color);
  color: var(--text-color);
  font-family: 'Inter', sans-serif;
  margin: 0;
  padding: 0;
}
```

3. Import file `globals.css` vào `pages/_app.js`:

```
import '../styles/globals.css';

export default function MyApp({ Component, pageProps }) {
  return <Component {...pageProps} />;
}
```

4. Kiểm tra xem style có áp dụng cho toàn ứng dụng không.

Bài 2: CSS Modules

1. Tạo component `Button.js`:

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

export default function Button({ variant, children }) {
  return (
    <button
      className={`${styles.button} ${variant === 'primary' ? styles.prim
    >
      {children}
    </button>
```

```
  );
}
```

2. Tạo file `Button.module.css`:

```
.button {
  padding: 12px 24px;
  border: none;
  border-radius: 6px;
  cursor: pointer;
  font-size: 16px;
}

.primary {
  background-color: var(--primary-color);
  color: white;
}
```

3. Sử dụng component trong `pages/index.js`:

```
import Button from '../components/Button';

export default function Home() {
  return (
    <div>
      <Button variant="primary">Primary Button</Button>
      <Button>Default Button</Button>
    </div>
  );
}
```

Bài 3: SASS

1. Cài đặt `sass`:

```
npm install sass
```

2. Đổi tên `styles/globals.css` thành `styles/globals.scss`:

```
$primary-color: #0070f3;
$background-color: #f4f4f4;
$text-color: #333;

:root {
  --primary-color: #{$primary-color};
  --background-color: #{$background-color};
  --text-color: #{$text-color};
}

body {
  background-color: $background-color;
  color: $text-color;
  font-family: 'Inter', sans-serif;
  margin: 0;
  padding: 0;
}
```

3. Đổi tên `Button.module.css` thành `Button.module.scss`:

```
$primary-color: #0070f3;

.button {
  padding: 12px 24px;
  border: none;
  border-radius: 6px;
  cursor: pointer;
  font-size: 16px;

  &.primary {
    background-color: $primary-color;
    color: white;
```

```
    &:hover {
      background-color: darken($primary-color, 10%);
    }
  }
}
```

4. Cập nhật import trong `Button.js`:

```
import styles from './Button.module.scss';
```

Bài 4: Cài đặt Tailwind CSS

1. Cài đặt các package:

```
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

2. Cấu hình `tailwind.config.js` như đã hướng dẫn ở phần lý thuyết.
3. Thêm Tailwind vào `styles/globals.css`:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

4. Kiểm tra xem Tailwind có hoạt động bằng cách thêm class `text-blue-500` vào một thẻ `<p>` trong `pages/index.js`.

Bài 5: Sử dụng Tailwind CSS

1. Tạo component `UserProfileCard.js`:

```jsx
export default function UserProfileCard() {
  return (
    <div className="max-w-sm mx-auto bg-white rounded-xl shadow-lg p-6 f
      <img
        className="w-24 h-24 rounded-full"
        src="/avatar.jpg"
        alt="Avatar"
      />
      <h2 className="text-xl font-bold mt-4">John Doe</h2>
      <p className="text-gray-600 text-center mt-2">
        Frontend Developer with a passion for React and Next.js
      </p>
      <button className="mt-4 bg-blue-500 text-white px-4 py-2 rounded-l
        Follow
      </button>
    </div>
  );
}
```

2. Sử dụng component trong `pages/index.js`:

```jsx
import UserProfileCard from '../components/UserProfileCard';

export default function Home() {
  return (
    <div className="min-h-screen flex items-center justify-center">
      <UserProfileCard />
    </div>
  );
}
```

3. Kiểm tra giao diện và đảm bảo không cần viết CSS riêng.

D. Bổ sung: So sánh các phương pháp styling

Phương pháp	Ưu điểm	Nhược điểm
Global CSS	Dễ áp dụng cho toàn ứng dụng, phù hợp với style chung (typography, reset).	Dễ gây xung đột class name, khó bảo trì nếu lạm dụng.
CSS Modules	Tính đóng gói cao, tránh xung đột, dễ quản lý.	Cần tạo file CSS riêng cho mỗi component, có thể tăng số lượng file.
SASS/SCSS	Hỗ trợ variables, nesting, mixins, code dễ đọc và tái sử dụng.	Cần cài đặt thêm `sass`, tăng thời gian build.
Tailwind CSS	Phát triển nhanh, nhất quán, file CSS nhỏ, không cần đặt tên class.	JSX dài dòng, cần học cú pháp class, khó tùy chỉnh style phức tạp.

E. Tài liệu tham khảo

- Next.js Documentation - CSS Support
- Tailwind CSS Documentation
- SASS Official Website

F. Bài tập nâng cao (tùy chọn)

1. **Kết hợp Tailwind và CSS Modules**: Tạo một component sử dụng cả Tailwind CSS (cho layout) và CSS Modules (cho style tùy chỉnh). Ví dụ: Dùng Tailwind cho padding, margin, flexbox; dùng CSS Modules cho hover effects hoặc animations.

2. **Tùy chỉnh Tailwind**: Thêm màu sắc tùy chỉnh vào `tailwind.config.js` (ví dụ: `primary: #0070f3`) và sử dụng trong component.
3. **Responsive Design**: Sử dụng Tailwind để tạo một layout responsive cho `UserProfileCard` (ví dụ: thay đổi kích thước avatar hoặc font dựa trên kích thước màn hình).