**THIẾT KẾ WEB**

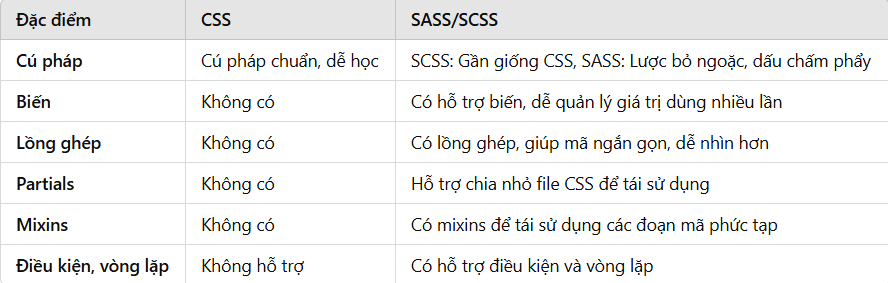
**CSS *&* SASS/SCSS**

**Nhóm: Syntax Error Squad**

* **Thành viên:**
* **Nguyễn Hồ Quốc An**
* **Nguyễn Phạm Khôi Minh**
* **Trần Anh Minh**
* **Trần Thanh Tú**
* **Tổng quan về chủ đề của nhóm:**

CSS và SASS / SCSS là những công cụ quan trọng trong phát triển giao diện web, giúp tạo và quản lý kiểu dáng, bố cục của trang web. Vì thấy được những lợi ích và tầm quan trọng của CSS và SASS / SCSS, nhóm Syntax Error Squad đã chọn đề tài này để tìm hiểu, nghiên cứu để có thể mang lại nhiều hiểu biết nhất về những công cụ này.

1. **Sự khác biệt giữa CSS và SASS/SCSS:**



1. **Vai trò chính của CSS và sự phát triển của các ngôn ngữ tiền xử lý như SASS/SCSS:**

CSS (Cascading Style Sheets) đóng vai trò quan trọng trong việc tạo giao diện cho các trang web, giúp xác định cách hiển thị của các thành phần HTML và mang đến trải nghiệm trực quan cho người dùng. Dưới đây là một số vai trò chính của CSS và sự phát triển của các ngôn ngữ tiền xử lý như SASS/SCSS:

### Vai trò thiết kế giao diện của CSS:

* **Tạo bố cục:** CSS cho phép tạo bố cục linh hoạt cho website, từ việc căn chỉnh, phân chia các khối, đến điều chỉnh khoảng cách giữa các thành phần, giúp trang web trở nên trực quan và dễ sử dụng.
* **Tối ưu hóa trải nghiệm người dùng:** CSS hỗ trợ các kỹ thuật như hiệu ứng động, animation, transition giúp tăng cường trải nghiệm tương tác của người dùng với website.
* **Responsive Design:** CSS cho phép xây dựng các trang web đáp ứng (responsive), tự động điều chỉnh kích thước và bố cục để phù hợp với các thiết bị khác nhau (máy tính, điện thoại, máy tính bảng).
* **Tách biệt giữa nội dung và hình thức:** CSS giúp tách biệt phần nội dung HTML và phong cách hiển thị, cho phép dễ dàng thay đổi giao diện mà không ảnh hưởng đến cấu trúc nội dung.
* ****Tăng khả năng bảo trì:**** CSS giúp trang web dễ dàng bảo trì hơn, vì CSS có thể tách biệt các định dạng styles thiết kế bố cục ra khỏi phần nội dung (HTML) của trang web bằng việc tạo ra một file CSS riêng biệt. Điều này giúp các nhà phát triển website dễ dàng cập nhật thay đổi giao diện mà không ảnh hưởng đến nội dung.
* **Tương thích với các trình duyệt:** CSS giúp tạo ra trang web tương thích với các trình duyệt khác nhau, đảm bảo rằng trang web hiển thị đồng nhất trên nhiều nền tảng.

### ****Sự phát triển của các ngôn ngữ tiền xử lý CSS (SASS/SCSS):****

* **Tăng tính cấu trúc và khả năng tái sử dụng mã**: SASS/SCSS cung cấp các tính năng như biến (variables), vòng lặp (loops), và hàm (functions), giúp mã CSS có tính cấu trúc hơn, dễ tái sử dụng và dễ bảo trì hơn so với CSS thuần.
* **Sử dụng nested syntax**: Việc lồng (nesting) các selector giúp mã CSS trở nên gọn gàng và dễ hiểu hơn, đặc biệt là khi làm việc với các bố cục phức tạp.
* **Modular CSS**: SASS/SCSS hỗ trợ import file, cho phép chia nhỏ mã CSS thành nhiều file riêng biệt và import khi cần thiết, dễ dàng quản lý và tái sử dụng mã trong các dự án lớn.
* **Mixins và hàm**: Mixins cho phép tái sử dụng các đoạn mã CSS phức tạp bằng cách truyền các tham số, giúp giảm thiểu sự trùng lặp mã và tiết kiệm thời gian viết mã.
* **Quản lý tốt hơn các tệp CSS lớn**: Trong các dự án lớn, việc viết mã CSS thuần sẽ dễ dẫn đến tình trạng mã phức tạp và khó bảo trì. Các ngôn ngữ tiền xử lý giúp tổ chức lại mã CSS một cách hiệu quả hơn, dễ dàng bảo trì và nâng cấp.

### ****Ứng dụng và xu hướng sử dụng SASS/SCSS:****

* Ngày càng nhiều các framework CSS (như Bootstrap, Foundation) đã hỗ trợ SASS/SCSS, giúp các lập trình viên dễ dàng tùy chỉnh và mở rộng giao diện theo yêu cầu.
* Xu hướng phát triển CSS hiện đại hướng đến tối ưu hóa và tự động hóa, do đó việc sử dụng SASS/SCSS ngày càng trở nên phổ biến trong các dự án web chuyên nghiệp.

**I/ CSS là gì?**

CSS (Cascading Style Sheets) là một dạng ngôn ngữ được sử dụng để điều chỉnh, định dạng lại các phần tử được tạo ra bởi ngôn ngữ đánh dấu siêu dữ liệu HTML (viết tắt của cụm từ HyperText Markup Language) trên trang web. CSS giúp thay đổi các yếu tố về mặt hiển thị như màu sắc, phông chữ, kích thước, bố cục, hiệu ứng, khoảng cách giữa các phần tử, và vị trí trên trang, làm cho các trang web trở nên trực quan và dễ nhìn hơn. CSS phù hợp với các dự án nhỏ, giúp tạo và thay đổi kiểu dáng nhanh chóng mà không cần cài đặt thêm công cụ.



**A. Các thành phần cơ bản của CSS:**

1. Selector (Bộ chọn)

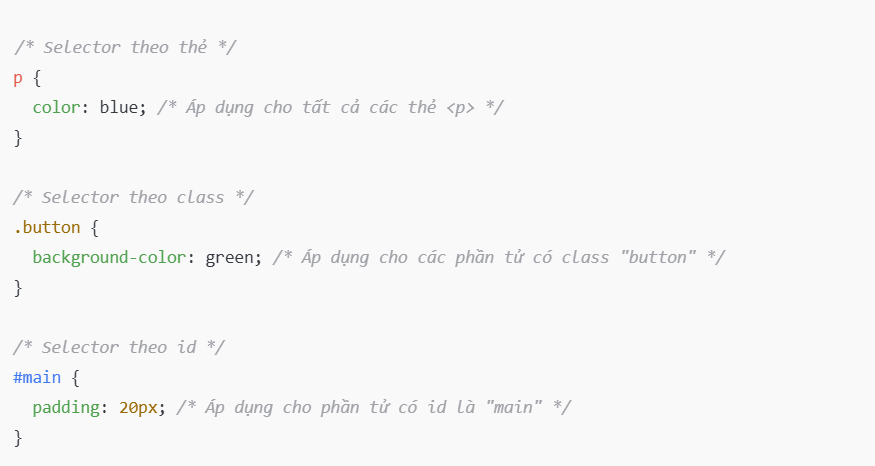
- Selector dùng để chọn các phần tử HTML mà ta muốn áp dụng các kiểu CSS.

- Có nhiều loại selector khác nhau:

* Selector cơ bản: Chọn các phần tử HTML theo thẻ, class, id.
* Selector phức hợp: Chọn các phần tử dựa vào cấu trúc, quan hệ và đặc điểm của chúng.

- Cụ thể hơn ta có những bộ chọn như:

* Selector theo tên phần tử: Chọn tất cả các phần tử có cùng tên.
* Selector theo ID: Chọn phần tử có ID cụ thể.
* Selector theo class: Chọn tất cả các phần tử có class cụ thể.
* Selector theo mối quan hệ: Chọn các phần tử dựa trên mối quan hệ của chúng với các phần tử khác.

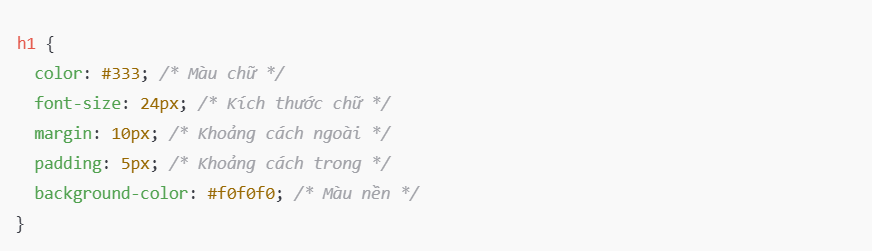


1. Property (Thuộc tính)

- Thuộc tính dùng để xác định các kiểu định dạng sẽ được áp dụng cho phần tử HTML được chọn.

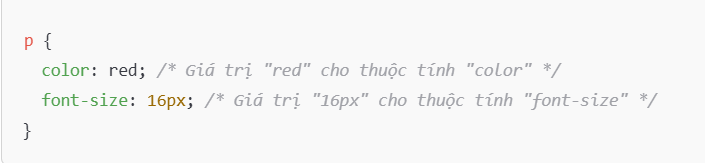
- Một số thuộc tính phổ biến:

* color: Màu sắc của văn bản.
* size: Kích thước chữ, hình ảnh.
* margin: Khoảng cách bên ngoài của phần tử.
* padding: Khoảng cách bên trong của phần tử.
* background-color: Màu nền của phần tử.
* vị trí : Xác định vị trí của các phần tử HTML trên trang web.
* font: Thay đổi font chữ của văn bản.



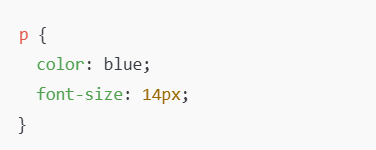
1. Value (Giá trị)

* Values là giá trị của các thuộc tính CSS. Giá trị của các thuộc tính CSS cũng có nhiều loại khác nhau như:
* Giá trị Number (số): Giá trị số nguyên, số thập phân, v.v.
* Giá trị String (chuỗi): Chuỗi ký tự.
* Giá trị Color (màu sắc): Tên màu, mã hex, giá trị RGB, v.v.



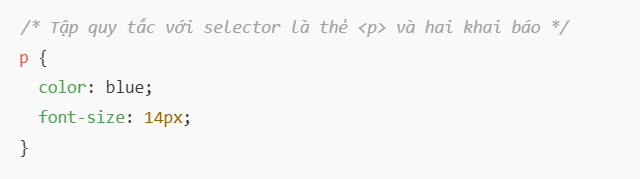
1. Declaration (Khai báo)

- Mỗi khai báo bao gồm một thuộc tính và giá trị của nó, được đặt giữa dấu {} và kết thúc bằng dấu “;”.



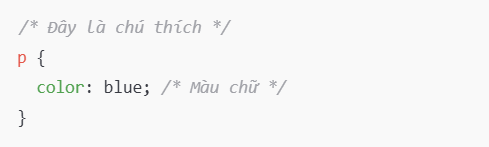
1. Rule Set (Tập quy tắc)

- Tập quy tắc bao gồm selector và các khai báo của nó. Đây là khối mã chính trong CSS.



1. Comment (Chú thích)

- Dùng để ghi chú, giải thích mã CSS. Các chú thích không ảnh hưởng đến mã CSS.



**=>** Các thành phần trên kết hợp với nhau tạo ra mã CSS hoàn chỉnh, giúp định dạng và điều chỉnh cách hiển thị của các phần tử trên trang web.

1. **Các đơn vị trong CSS:**

CSS hỗ trợ nhiều đơn vị để xác định kích thước, khoảng cách cho các phần tử trang web. Một số đơn vị phổ biến:

**- Đơn vị tuyệt đối:**

* Px: pixel (điểm ảnh)
* In: inch (inch)
* Cm: centimeter (centimet)
* Mm: millimeter (milimet)

**- Đơn vị tương đối:**

* %: Đơn vị phần trăm
* Em: Đơn vị em được tính toán dựa trên kích thước của phần tử cha hoặc phần tử gần nhất có thuộc tính font-size. Giá trị 1em tương đương với kích thước font chứa nó.
* Rem: Đơn vị rem được tính toán dựa trên kích thước của phần tử gốc (root element), thường là phần tử <html>. Giá trị 1rem tương đương với kích thước font chứa nó trong phần tử gốc.
* Vw/vh: Viewpoint width/Viewpoint height: Tương ứng chiều rộng/cao của cửa sổ trình duyệt hay của màn hình hiển thị.

1. **CSS và HTML :**

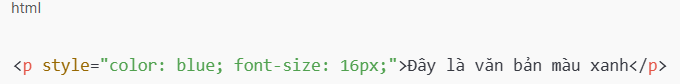
CSS và HTML kết hợp với nhau để tạo giao diện người dùng bằng cách áp dụng các quy tắc và định dạng từ CSS lên các phần tử trong tài liệu HTML. HTML cung cấp cấu trúc và nội dung của trang web, trong khi CSS đảm nhận việc thiết kế, làm cho giao diện trở nên sinh động và thu hút người dùng hơn.

**- Cách HTML và CSS kết hợp với nhau:**

CSS có ba loại chính: inline CSS, internal CSS, và external CSS. Mỗi loại có các ưu, nhược điểm riêng, và thường được lựa chọn dựa trên quy mô và mục đích của dự án.

1. **CSS Inline (Nội tuyến)**

CSS được thêm trực tiếp vào phần tử HTML thông qua thuộc tính style.



**Ưu điểm:**

- Áp dụng nhanh: Inline CSS thường được sử dụng cho các thay đổi nhỏ và nhanh chóng trên một phần tử cụ thể, không cần phải tạo file CSS riêng.

- Không cần tải thêm file: Vì mã CSS được tích hợp trong HTML, nên không cần phải tải file CSS từ máy chủ, làm cho thời gian tải trang nhanh hơn khi có ít mã CSS.

**Nhược điểm:**

**-** Khó bảo trì và quản lý: Inline CSS có thể gây khó khăn khi cập nhật hoặc sửa đổi giao diện, nhất là với trang web lớn, vì mã CSS bị phân tán và trộn lẫn trong HTML.

- Thiếu khả năng tái sử dụng: Không thể tái sử dụng đoạn mã cho các phần tử khác, dẫn đến việc lặp lại mã và làm tăng kích thước file HTML.

- Thiếu tổ chức: Khi sử dụng inline CSS, khó có thể áp dụng các quy tắc thiết kế nhất quán trên toàn bộ trang web.

1. **CSS Internal (Nội bộ)**

CSS được đặt trong thẻ <style> bên trong phần <head> của tài liệu HTML.



**Ưu điểm:**

- Tương đối dễ quản lý: Mã CSS được đặt trong cùng một file HTML, nên có thể dễ dàng quản lý các kiểu cho từng trang riêng lẻ.

- Phù hợp với trang web nhỏ: Với những trang web nhỏ hoặc trang đơn lẻ (single page), internal CSS là lựa chọn phù hợp, không cần phải tạo file CSS riêng.

**Nhược điểm:**

- Không tái sử dụng giữa các trang: Internal CSS chỉ áp dụng cho một trang HTML cụ thể, không thể sử dụng lại cho các trang khác, gây khó khăn khi muốn thay đổi giao diện đồng bộ.

- Tăng kích thước tài liệu HTML: Khi có nhiều mã CSS, phần <style> sẽ làm tăng kích thước file HTML, ảnh hưởng đến tốc độ tải trang.

1. **CSS External (Ngoại vi)**

CSS được viết trong một file .css riêng biệt và liên kết với tài liệu HTML qua thẻ <link> trong phần <head>. Đây là cách phổ biến nhất vì giúp mã dễ quản lý, bảo trì và tái sử dụng.



**styles.css**



**Ưu điểm:**

- Khả năng tái sử dụng cao: Có thể sử dụng cùng một file CSS cho nhiều trang web khác nhau, giúp giảm thiểu công sức và thời gian khi cập nhật giao diện.

- Giúp tách biệt mã HTML và CSS: External CSS giúp tách biệt giữa nội dung và giao diện, giữ cho mã HTML gọn gàng và dễ quản lý.

- Tối ưu hóa tốc độ tải trang: Khi người dùng truy cập nhiều trang trên cùng một trang web, file CSS sẽ được tải một lần và lưu trữ trong bộ nhớ cache của trình duyệt, giúp tăng tốc độ tải các trang tiếp theo.

**Nhược điểm:**

- Thêm yêu cầu tải file: Với mỗi trang tải, trình duyệt cần phải tải file CSS riêng biệt từ máy chủ, điều này có thể làm tăng thời gian tải trang ban đầu nếu không sử dụng bộ nhớ cache.

- Không phù hợp với trang web nhỏ hoặc thay đổi tạm thời: Với các thay đổi tạm thời hoặc trang web nhỏ, external CSS có thể gây phức tạp hơn so với inline hoặc internal CSS.

* Vậy thông qua việc lựa chọn loại CSS phù hợp với mục đích và quy mô của dự án, ta có thể tối ưu hóa giao diện cũng như hiệu suất trang web của mình:
* Inline CSS: Phù hợp cho các thay đổi nhanh chóng, nhỏ lẻ.
* Internal CSS: Tốt cho trang đơn lẻ hoặc các thay đổi giao diện riêng lẻ của một trang.
* External CSS: Phù hợp với các trang web lớn, cần tính nhất quán và dễ quản lý trong dài hạn.

**- Cách CSS áp dụng định dạng lên HTML:**

Khi trang web được tải, trình duyệt đọc file HTML để dựng cấu trúc của trang và sau đó áp dụng các quy tắc CSS. Các quy tắc CSS nhắm đến các phần tử HTML dựa trên các bộ chọn như thẻ, class, id, và áp dụng các thuộc tính để điều chỉnh giao diện như màu sắc, font chữ, bố cục, khoảng cách, và nhiều thuộc tính khác.

- Ví dụ, nếu HTML có các phần tử như sau:



Và CSS được liên kết như sau:



Trình duyệt sẽ định dạng các phần tử HTML theo quy tắc CSS:

* container sẽ có chiều rộng 80% và căn giữa trang.
* title có kích thước chữ 24px, màu chữ là #333.
* description có kích thước chữ 16px, màu chữ là #666.

1. **CSS Preprocessor là gì?**

CSS Preprocessors là ngôn ngữ tiền xử lý CSS. Là một ngôn ngữ kịch bản mở rộng của CSS và được biên dịch thành cú pháp CSS giúp ta viết CSS nhanh hơn và có cấu trúc rõ ràng hơn. CSS Preprocessor có thể giúp ta tiết kiệm thời gian viết CSS với các tính năng mạnh mẽ như biến, mixin, lồng (nesting), vòng lặp, và kế thừa, giúp dễ dàng bảo trì và phát triển CSS.

**Vì sao nên sử dụng CSS Preprocessor?**

* Tái sử dụng mã: Giúp giảm mã lặp lại với biến và mixin, dễ duy trì và chỉnh sửa.
* Cấu trúc rõ ràng: Với khả năng lồng, ta có thể tổ chức mã CSS theo cấu trúc HTML, giúp dễ đọc hơn.
* Khả năng mở rộng: Dễ dàng thêm tính năng mới mà không phải thay đổi toàn bộ mã.rì và phát triển CSS.

**- Các tính năng chính của CSS Preprocessor:**

1. **Variables (Biến)**

Lưu trữ các giá trị tái sử dụng như màu sắc, font, hoặc kích thước.



1. **Nesting (Lồng)**

Cho phép lồng các quy tắc CSS theo cấu trúc của HTML.



1. **Mixin**

Định nghĩa các khối thuộc tính có thể tái sử dụng và gọi lại khi cần, có thể nhận tham số để tùy biến.



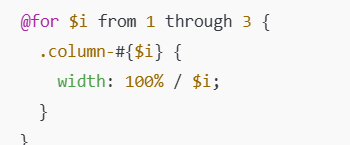
1. **Inheritance (Kế thừa)**

Cho phép phần tử kế thừa thuộc tính của phần tử khác bằng @extend.



1. **Control Directives & Expressions**

Hỗ trợ vòng lặp, điều kiện và các phép tính giúp mã CSS mạnh mẽ hơn.



**- Cách hoạt động của CSS Preprocessor:**

+ CSS Preprocessor sẽ biên dịch mã nguồn (file .scss hoặc .less) thành CSS chuẩn trước khi gửi đến trình duyệt. Trình duyệt không đọc trực tiếp mã từ các file SASS hoặc LESS mà thay vào đó, CSS Preprocessor tạo ra file CSS cuối cùng chứa các quy tắc CSS chuẩn.

+ CSS Preprocessor rất hữu ích trong các dự án lớn nhờ khả năng tổ chức mã tốt, tiết kiệm thời gian và công sức khi xây dựng và duy trì giao diện trang web.

II/ SASS, SCSS là gì?

Bất cứ một lập trình viên nào đều phải từng làm việc liên quan đến CSS. Tuy nhiên làm việc với CSS thuần một thời gian dài ta sẽ thấy nó rất nhàm chán. ta có thể viết CSS một cách chuyên nghiệp hơn, nhanh và rõ ràng mạch lạc hơn bằng SASS/SCSS.



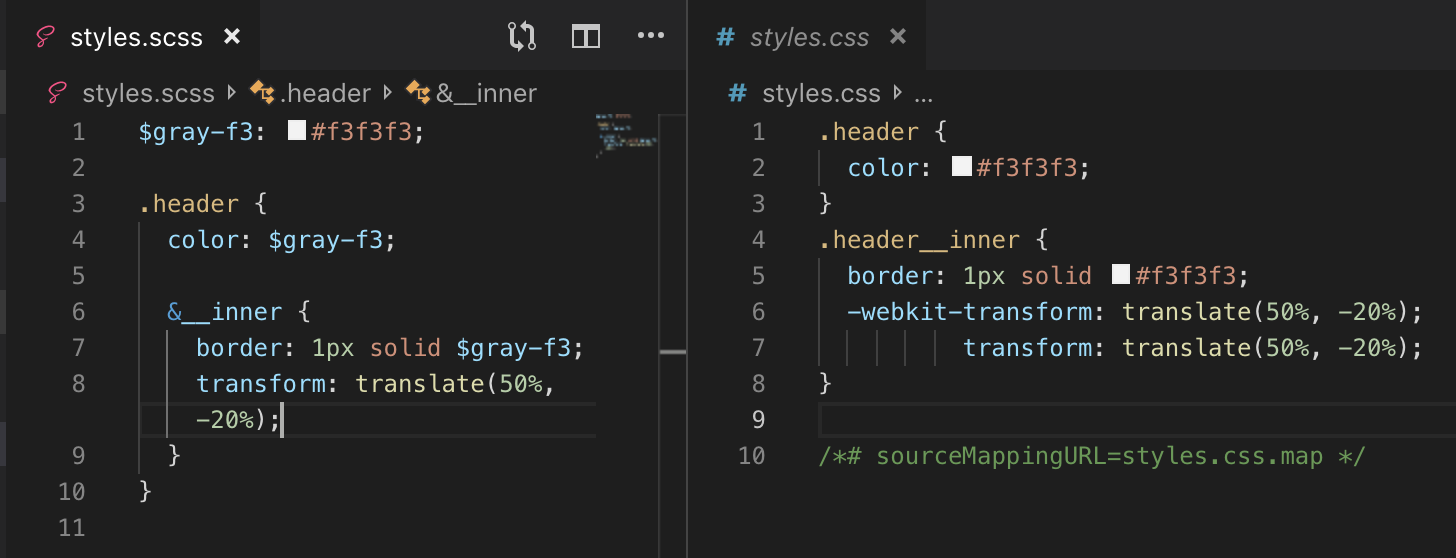
- SASS/SCSS là một chương trình tiền xử lý CSS (*CSS preprocessor),* gọi như vậy là vì chúng cung cấp các tính năng giúp viết mã CSS trước khi được biên dịch thành mã CSS thuần. SASS/SCSS giúp ta viết CSS theo cách của một ngôn ngữ lập trình, có cấu trúc rõ ràng, rành mạch, dễ phát triển và bảo trì code hơn. Ngoài ra nó có rất nhiều các thư viện hỗ trợ kèm theo giúp ta viết code CSS một cách dễ dàng vào đơn giản hơn. Có rất nhiều loại CSS Preprocessor trong đó bao gồm SASS, Stylus hay LESS. SASS/SCSS phù hợp với các dự án lớn, nơi cần quản lý phức tạp và tái sử dụng mã. SASS/SCSS giúp giảm thời gian chỉnh sửa và bảo trì mã, giúp tổ chức mã một cách rõ ràng và dễ quản lý hơn nhờ vào các tính năng mở rộng của chúng.

**- SASS**và **SCSS**về bản chất vấn đề là giống nhau, chỉ khác nhau ở cách viết

Sass là chữ viết tắt của Syntactically Awesome Style Sheets, chương trình tiền xử lý bằng ngôn ngữ kịch bản (Preprocessor Scripting Language ), sẽ được biên dịch thành CSS. Nghĩa là, mình sẽ làm style bằng SASS, rồi SASS sẽ render việc mình làm thành file CSS.

**- SASS** bản thân có hai kiểu viết khác nhau, một kiểu như là HAML, Pug – Sử dụng indent (cách thụt đầu dòng) để phân tách các khối code , sử dụng xuống dòng để phân biệt rules , có phần mở rộng là **.sass.**

**- SCSS** sử dụng cú pháp giống với Ruby (vì đơn giản nó được thiết kế bởi các lập trình viên Ruby)**.**Có phần mở rộng là **.scss**, SCSS ra đời sau SASS và có cú pháp viết tương tự như cách viết CSS. Cú pháp này được tạo ra nhằm thu hẹp khoảng cách giữa SASS và CSS bằng cách mang lại một thứ gì đó thân thiện với CSS.



– Bên trái: Được viết bằng SCSS

– Bên phải: là code CSS được biên dịch từ SCSS

1. Các tính năng cơ bản của SASS:

**1. Biến (Variables)**

SASS cho phép sử dụng biến để lưu trữ các giá trị như màu sắc, kích thước font, khoảng cách, v.v., giúp dễ dàng tái sử dụng và quản lý chúng ở nhiều nơi trong mã CSS.



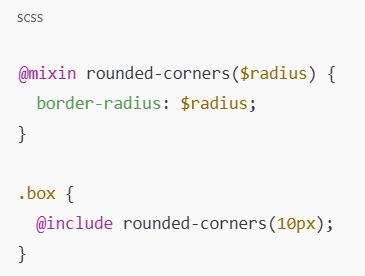
**2. Lồng nhau (Nesting)**

SASS hỗ trợ cú pháp lồng nhau, giúp viết mã CSS có cấu trúc dễ nhìn, giống như cấu trúc HTML, giúp quản lý mã trong các thành phần phức tạp dễ dàng hơn.



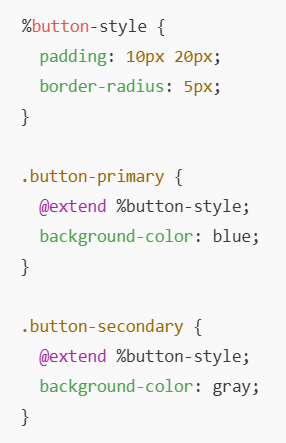
**3. Mixin**

Mixin là các khối mã CSS có thể tái sử dụng và có thể truyền tham số vào. Điều này giúp tránh việc lặp lại mã khi cần dùng các thuộc tính CSS giống nhau ở nhiều nơi.



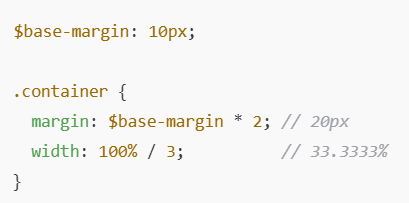
**4. Kế thừa (Inheritance / Extend)**

SASS cho phép kế thừa các kiểu từ một selector khác bằng cách sử dụng @extend, giúp tránh việc viết lại mã khi các lớp có cùng kiểu dáng.



**5. Toán tử (Operators)**

SASS hỗ trợ các phép toán như cộng, trừ, nhân, chia, giúp tính toán giá trị thuộc tính CSS dễ dàng, đặc biệt hữu ích khi làm việc với các đơn vị đo lường và kích thước.d, giúp tránh việc viết lại mã khi các lớp có cùng kiểu dáng.



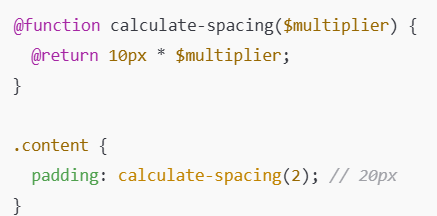
**6. Phân chia file (Partial và Import)**

SASS cho phép chia mã CSS thành nhiều file nhỏ (gọi là partials) và nhập chúng vào một file chính, giúp tổ chức mã rõ ràng hơn trong các dự án lớn.

* Partial: Đặt tên file bắt đầu với dấu gạch dưới \_ (ví dụ: \_header.scss). Điều này giúp SASS nhận diện file là partial và không biên dịch riêng lẻ.
* Import: Sử dụng @import để nhập các file partial vào một file chính.

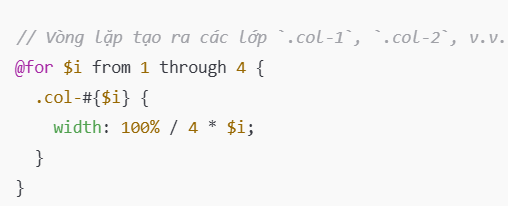
**7. Hàm (Functions)**

SASS hỗ trợ viết hàm riêng để thực hiện các tính toán phức tạp và trả về giá trị cho các thuộc tính CSS. Các hàm tích hợp sẵn như lighten(), darken() giúp dễ dàng thao tác với màu sắc.



**8. Kiểm tra điều kiện (Control Directives and Loops)**

SASS hỗ trợ các cấu trúc điều kiện và vòng lặp (@if, @for, @each, @while), giúp tạo ra các quy tắc CSS phức tạp và linh hoạt.

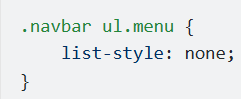


* **Quy tắc xếp chồng (Nested Rule) của SASS:**

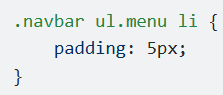
Quy tắc xếp chồng được sử dụng rất nhiều khi vào 1 project có viết css bằng SASS. Ví dụ ta có một đoạn HTML:



Giả sử ta chỉ muốn CSS cho thẻ ul với class menu, với CSS thuần ta gõ:



Nếu tiếp tục muốn CSS cho thẻ li trong thẻ ul (có class là menu) thì:



Nếu tiếp tục muốn CSS cho thẻ a trong thẻ li... thì cứ phải viết lặp đi lặp lại tên tag (hoặc class, hoặc id) cha của thẻ muốn CSS

Nhưng với Nested Rule của SASS, ta có thể làm điều trên một cách đơn giản hơn. Cú pháp như sau:



Và sau khi được đoạn SASS trên được compile ra CSS thuần sẽ như sau:



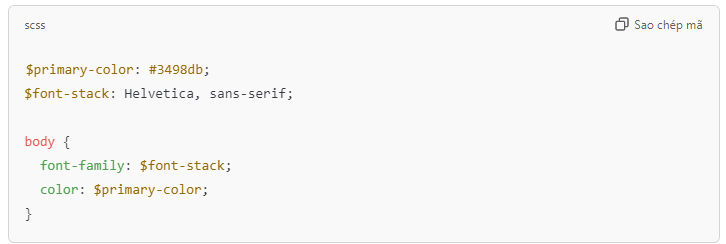
=> Những tính năng cơ bản của SASS giúp CSS trở nên mạnh mẽ hơn, tăng khả năng tái sử dụng và dễ bảo trì mã. Điều này giúp các nhà phát triển quản lý giao diện dễ dàng hơn và tăng tốc độ phát triển trong các dự án lớn.

1. Các tính năng cơ bản của SCSS:

SCSS có các tính năng tương tự SASS nhưng với cú pháp gần giống CSS truyền thống hơn, giúp các nhà phát triển CSS dễ dàng tiếp cận và áp dụng các tính năng mở rộng.

**1. Biến (Variables)**

SCSS cho phép tạo biến để lưu các giá trị như màu sắc, font, kích thước, giúp dễ dàng tái sử dụng và thay đổi.



**2. Lồng nhau (Nesting)**

SCSS hỗ trợ lồng các selector, giúp mã ngắn gọn và dễ đọc hơn khi làm việc với các phần tử có cấu trúc phân cấp.



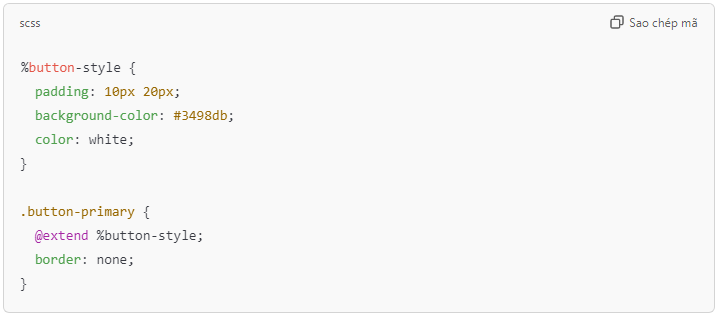
**3. Mixin**

Mixin cho phép tạo các khối mã có thể tái sử dụng. Điều này rất hữu ích khi ta cần một nhóm thuộc tính CSS dùng chung nhiều nơi.



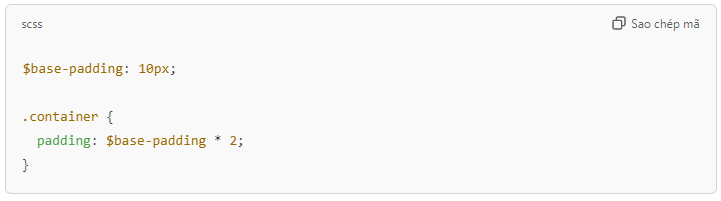
**4. Kế thừa (Inheritance / Extend)**

SCSS cho phép sử dụng @extend để kế thừa các thuộc tính từ selector khác, giúp mã gọn gàng hơn và hạn chế trùng lặp.



**5. Toán tử (Operators)**

SCSS hỗ trợ các toán tử (như +, -, \*, /, %) để thực hiện tính toán trên các giá trị, đặc biệt là kích thước và màu sắc.



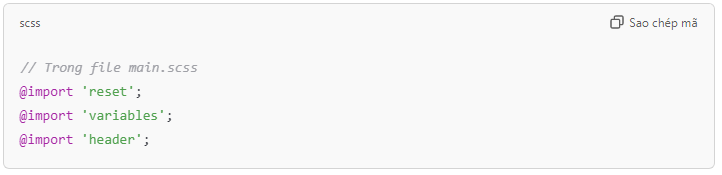
**6. Hàm màu sắc (Color Functions)**

SCSS cung cấp các hàm xử lý màu sắc để tạo màu sắc mới từ màu có sẵn, như lighten, darken, adjust-hue, rgba, v.v.



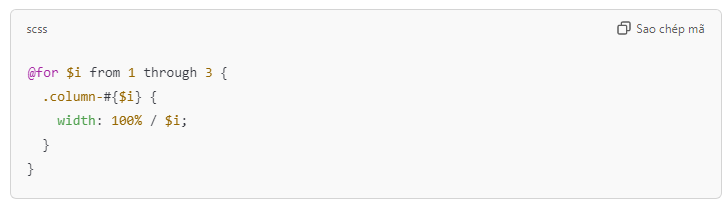
**7. Import**

SCSS hỗ trợ import các file SCSS khác, giúp tổ chức mã và chia nhỏ các phần thành nhiều file dễ quản lý.



**8. Vòng lặp (Loop)**

SCSS cung cấp các cấu trúc lặp như @for, @each, và @while để tự động tạo các nhóm thuộc tính hoặc selectors.



**9. Điều kiện (Conditionals)**

SCSS hỗ trợ các câu điều kiện @if, @else if, và @else, giúp tùy chỉnh CSS theo điều kiện cụ thể.



1. Thông tin thêm:
2. **Nguồn gốc và hoàn cảnh ra đời của SASS:**

Vào những năm 2000, CSS đã trở thành ngôn ngữ chuẩn để tạo phong cách cho các trang web. Tuy nhiên, khi các trang web ngày càng phức tạp hơn, việc quản lý các tệp CSS lớn trở thành một thách thức. CSS thuần thiếu các tính năng như biến, hàm, lồng nhau, và không hỗ trợ modular hóa, dẫn đến mã CSS rất dễ bị lặp lại, khó bảo trì và sửa lỗi. Nhằm giải quyết các vấn đề trên, Hampton Catlin đã phát triển SASS như một ngôn ngữ mở rộng cho CSS, với mục đích bổ sung các tính năng giúp CSS mạnh mẽ và hiệu quả hơn. SASS cho phép các nhà phát triển sử dụng biến, cấu trúc lồng nhau và các tính năng logic, tạo ra một cách tiếp cận mới trong việc xây dựng giao diện web. Natalie Weizenbaum và Chris Eppstein đã tham gia vào dự án và giúp mở rộng các tính năng của SASS. Họ bổ sung mixin, inheritance, và các tính năng quan trọng khác, giúp SASS trở nên phổ biến và mạnh mẽ hơn.

1. **Sự ra đời của SCSS:**

Khái niệm SCSS: Vào năm 2010, SCSS (Sassy CSS) được giới thiệu như một cú pháp khác của SASS, với cú pháp tương tự CSS truyền thống hơn. SCSS giữ nguyên cấu trúc của CSS (sử dụng {}, ;, và hầu hết các quy tắc chuẩn), đồng thời tích hợp tất cả tính năng nâng cao của SASS. Điều này giúp các nhà phát triển CSS truyền thống dễ dàng học và áp dụng SCSS hơn.

SCSS được phát triển để giúp SASS dễ tiếp cận hơn cho những người quen thuộc với CSS thuần. Ban đầu, cú pháp SASS không sử dụng {} và ;, khác biệt lớn so với CSS, có thể khiến các nhà phát triển khó chuyển đổi. SCSS xuất hiện như một phiên bản nâng cấp và thân thiện hơn, giúp việc chuyển đổi sang SASS trở nên mượt mà và dễ dàng hơn cho các lập trình viên.

1. **Điểm giống nhau và khác nhau về tính năng của CSS Preprocessor và tính năng của SCSS:**

- SCSS là một loại CSS Preprocessor, vì vậy nó kế thừa các tính năng chính của CSS Preprocessor và thêm vào một số tính năng độc đáo.

**- Những điểm giống nhau cơ bản giữa tín năng của CSS Preprocessor và của SCSS:**

**1. Variables:**

Cho phép sử dụng biến để lưu trữ các giá trị như màu sắc, font, kích thước,... giúp mã CSS dễ chỉnh sửa và quản lý.

**2. Nesting:**

Cho phép tổ chức các quy tắc CSS bằng cách lồng chúng vào nhau theo cấu trúc của HTML, giúp mã CSS dễ đọc hơn.

**3. Mixin:**

Cho phép định nghĩa một nhóm thuộc tính có thể tái sử dụng nhiều lần.

4. Inheritance:

Cho phép các phần tử kế thừa thuộc tính từ các phần tử khác.

**5. Loops / Conditionals:**

Cho phép sử dụng các vòng lặp và điều kiện như @for, @if, @each để tự động tạo ra các quy tắc CSS, giúp tiết kiệm thời gian khi viết CSS phức tạp.

**- Những điểm khác nhau giữa CSS Preprocessor và SCSS:**

1. **Cú pháp:**

|  |  |
| --- | --- |
| CSS Preprocessor | SCSS |
| Các preprocessors khác nhau có cú pháp riêng, ví dụ như SASS sử dụng thụt lề và không có dấu {}, ;, trong khi LESS sử dụng cú pháp gần giống CSS hơn. | Có cú pháp tương thích với CSS tiêu chuẩn, tức là sử dụng {}, ; và dễ học cho những người đã quen với CSS. |

1. **Khả năng tương thích với CSS:**

SCSS được thiết kế để tương thích hoàn toàn với CSS tiêu chuẩn, vì vậy ta có thể sao chép mã CSS vào file SCSS mà không cần sửa đổi. Một số CSS Preprocessors khác, như LESS, có thể không hoàn toàn tương thích với cú pháp CSS.

1. **Tính năng riêng biệt của từng Preprocessor:**

* SCSS có nhiều tính năng hơn hầu hết các CSS Preprocessor khác, như module system (hệ thống module), hỗ trợ import có thể mở rộng, và một cộng đồng lớn giúp phát triển và cập nhật thường xuyên.
* LESS hoặc Stylus có một số tính năng và cú pháp khác biệt nhưng ít linh hoạt hơn so với SCSS.

1. **Hệ sinh thái và công cụ hỗ trợ:**

SCSS là phần mở rộng của SASS, có cú pháp CSS hoàn chỉnh và được sử dụng rộng rãi trong các dự án lớn nhờ cộng đồng và tài liệu phong phú.

III/ Tổng kết

**CSS (Cascading Style Sheets):**

Dễ học, dễ sử dụng và là nền tảng của mọi trình duyệt. Hầu hết các lập trình viên đều có kiến thức về CSS. CSS nguyên bản có thể giảm thiểu độ phức tạp và tăng tốc độ tải trang nếu được viết gọn gàng. Với lịch sử lâu dài, CSS có tài liệu phong phú và hỗ trợ từ cộng đồng. Tuy nhiên CSS không hỗ trợ tổ chức mã hiệu quả, gây khó khăn khi phát triển dự án lớn. Các file CSS lớn dễ lặp lại, khó tìm lỗi và khó bảo trì. CSS thiếu các tính năng nâng cao như biến số, vòng lặp và điều kiện, khiến mã phức tạp hơn.

**SASS/SCSS (Syntactically Awesome Style Sheets):**

SASS cho phép sử dụng biến, hàm, mixins và vòng lặp, giúp mã ngắn gọn, dễ đọc và dễ bảo trì hơn. SASS cho phép phân chia mã thành nhiều file nhỏ, giúp quản lý dễ dàng và rõ ràng hơn. Với các tính năng mạnh mẽ, SASS giúp giảm thời gian viết mã và tăng năng suất. SASS cũng có những nhược điểm như phải được biên dịch thành CSS trước khi sử dụng, làm tăng thêm bước trong quy trình làm việc. SASS phức tạp hơn CSS, đòi hỏi lập trình viên học và làm quen với cú pháp mới. Cần sử dụng thêm công cụ biên dịch (như Node.js, webpack), khiến việc triển khai và phát triển phức tạp hơn.