Mục tiêu:

1. Hiểu được SASS và SCSS là gì và cú pháp sử dụng SCSS?
2. Áp dụng thành thạo các tính năng của SCSS như lồng nhau, biến, mixin, function, extends, import, operator, loop, conditional, v.v.
3. Sử dụng SCSS trong xây dựng responsive website.
4. Sử dụng SCSS kết hợp quy tắc đặt tên BEM.

Khóa học này tích hợp trình biên dịch Sass online. Bạn có thể chỉnh sửa trực tiếp vào các ví dụ nhé.

SASS và SCSS là gì?

Khái niệm

SASS là một ngôn ngữ mở rộng của CSS, được phát triển bởi các lập trình viên Ruby để giải quyết các hạn chế của CSS truyền thống.

SASS cho phép sử dụng các tính năng như biến, lồng nhau (nesting), mixins, và các quy tắc logic điều kiện, giúp mã CSS trở nên linh hoạt và dễ quản lý hơn.

Tuy nhiên SASS cũng là khá mới và dễ gây nhầm lẫn với người đã quen code CSS truyền thống vì nó không có `{}`, `;` và dùng `tab` để lồng nhau.

Cho đến phiên bản 3.0 được phát hành vào tháng 5/2010. Nó được giới thiệu thêm một cú pháp mới có tên là **SCSS (Sassy CSS)** với cú pháp tương tự như CSS truyền thống, giúp cho người mới học dễ tiếp cận hơn.

Cú pháp này nhằm thu hẹp sự khác biệt giữa CSS và SCSS, giúp cho việc chuyển đổi từ CSS sang SCSS dễ dàng hơn.

Cho đến nay hầu hết người dùng SASS đều sử dụng SCSS.

Những lợi ích của SCSS

SCSS có nhiều lợi ích rõ ràng khi sử dụng như:

- Cú pháp lồng nhau giúp bố cục mã nguồn rõ ràng, dễ đọc và dễ bảo trì, tiết kiệm code khi phải viết những selector dài.
- Biến giúp tái sử dụng giá trị nhiều lần mà không cần phải viết lại.
- `Mixin` giúp tái sử dụng các đoạn mã CSS mà không cần phải viết lại.
- `Function` giúp tính toán giá trị CSS mà không cần phải viết lại.
- `Extends` giúp kế thừa các thuộc tính CSS từ một selector khác.
- `Import` giúp chia nhỏ mã nguồn CSS thành các file nhỏ hơn, dễ quản lý hơn.
- `Operator` giúp thực hiện các phép toán trên giá trị CSS mà không cần đến `calc()` nhưng trong CSS.
- Việc sử dụng SCSS giúp tăng hiệu suất làm việc, giảm thời gian code và giảm lỗi.

Cú pháp sử dụng SCSS

Cú pháp sử dụng SCSS tương tự như CSS truyền thống, các tính năng sẽ được chia sẻ chi tiết ở phần sau.

Đuôi mở rộng của file SCSS là `.scss`.

SCSS cũng không thể được code inline trong file HTML hoặc đặt ở cặp thẻ `style`, mà cần phải được biên dịch thành CSS trước khi sử dụng.

```
.navbar {
  ul.menu {
    list-style: none;
    li {
      padding: 5px;
      a {
        text-decoration: none;
      }
    }
  }
}
```

Biên dịch và sử dụng SCSS

Hiện nay có rất nhiều trình biên dịch hỗ trợ SCSS như. Tiện lợi nhất khi bạn sử dụng Visual Studio Code là sử dụng extension `Live Sass Compiler`.

Ngoài ra, bạn cũng có thể sử dụng `node-sass` hoặc phần mềm kola để biên dịch file SCSS.

Sau khi biên dịch SCSS thành CSS, bạn có thể sử dụng file CSS đó trong file HTML như bình thường.

Các tính năng của SCSS

Lồng nhau (Nesting selector)

Khi bạn xây dựng các selector lồng nhau trong HTML, bạn cũng có thể lồng nhau các selector trong SCSS. Điều này giúp bạn tiết kiệm việc viết lại các selector cha.

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
    li {
      display: inline-block;
      a {
        display: block;
        padding: 6px 12px;
        text-decoration: none;
      }
    }
  }
}
```

Tuy nhiên, việc để quá nhiều các selector lồng nhau trong SCSS cũng là điều không được khuyến khích. Tối đa chỉ nên lồng nhau 3 cấp, nếu nhiều hơn sẽ làm cho mã nguồn trở nên khó đọc và bảo trì.

Biến

Việc sử dụng biến trong SCSS rất dễ dàng, bạn không cần ghi trong bất cứ một bộ chọn nào nếu như biến đó là biến toàn cục.

```
$primary-color: #3498db;
$font-size: 16px;

body {
  font-size: $font-size;
  color: $primary-color;
}
```

Sử dụng $ để khai báo biến và gọi biến trong SCSS cũng giúp tiết kiệm thời gian và giảm lỗi khi cần thay đổi giá trị của biến.

Mixin

Mixin là cách để định nghĩa một nhóm các thuộc tính CSS có thể tái sử dụng ở nhiều nơi khác nhau. Bạn có thể truyền tham số vào mixin để điều chỉnh giá trị.

```
@mixin flex-center {
  display: flex;
  justify-content: center;
  align-items: center;
}

.container {
  @include flex-center;
  height: 100vh;
}
```

Trong ví dụ này, mixin flex-center giúp tái sử dụng các thuộc tính liên quan đến căn chỉnh flexbox.

Hàm (Function)

SCSS có các hàm tích hợp sẵn và cho phép bạn định nghĩa các hàm riêng để xử lý giá trị.

```
@function calculate-rem($px-value) {
  @return $px-value / 16px * 1rem;
}

.container {
```

```
  font-size: calculate-rem(32px); // Kết quả: 2rem
}
```

Kế thừa (Inheritance)

SCSS hỗ trợ kế thừa, cho phép một selector kế thừa các thuộc tính từ một selector khác bằng cách sử dụng từ khóa @extend.

```
%button-style {
  padding: 10px 20px;
  border-radius: 5px;
  font-size: $font-size;
}

.button {
  @extend %button-style;
  background-color: $primary-color;
}
```

Import

SCSS cho phép bạn chia nhỏ mã nguồn CSS thành các file nhỏ hơn và import chúng vào một file chính. Với những file SCSS không dùng trực tiếp, bạn có thể đặt dấu _ ở đầu tên file để biểu thị rằng đó là file partial.

```
@import 'filename';
```

- Bạn không cần thêm phần mở rộng .scss khi nhập tệp.
- SCSS sẽ tự động tìm kiếm và nhập tệp với tên filename.scss.
- Nếu bạn nhập một tệp bắt đầu bằng dấu gạch dưới (ví dụ: _filename.scss), SCSS sẽ coi đó là một partial (phần tệp), và tệp này sẽ không được biên dịch thành CSS riêng mà chỉ được kết hợp vào tệp chính.

Ví dụ:

```
@import 'base';
@import 'components';
@import 'layout';
@import 'pages';
@import 'utils';
```

Ở phía trên là nội dung một file index.scss, file này sẽ import các file partial khác nhau.

Khuyến nghị: Sử dụng @use và @forward

@use: Đây là một cách mới để import một module vào một file khác. Nó giúp tránh tình trạng import trùng lặp và giúp quản lý tốt hơn.

Ví dụ:

```
// _variables.scss
$primary-color: #3498db;
$font-size: 16px;
```

```
// main.scss
@use 'variables';

body {
  font-size: variables.$font-size;
  color: variables.$primary-color;
}
```

Tại file main.scss, bạn sử dụng @use để import file _variables.scss vào file main.scss. Sau đó, bạn có thể sử dụng biến $font-size và $primary-color từ file _variables.scss.

@forward: Được sử dụng để chuyển tiếp các biến, mixin, function từ một module sang một module khác.

Ví dụ:

```
// _variables.scss
$primary-color: #3498db;
$font-size: 16px;
```

```
// _shared.scss
@forward 'variables';
```

```
// main.scss
@use 'shared';

body {
  font-size: shared.$font-size;
  color: shared.$primary-color;
}
```

Nhờ cú pháp `import` mà các file SCSS thường được hệ thống theo cấu trúc thư mục nhất định, giúp cho việc quản lý mã nguồn dễ dàng hơn.

Cấu trúc thư mục scss

```
styles/
|
├── base/
|   ├── _reset.scss
|   ├── _typography.scss
|   └── _base.scss
|
├── components/
|   ├── _buttons.scss
|   ├── _carousel.scss
|   └── _modal.scss
|
├── layout/
|   ├── _header.scss
|   ├── _footer.scss
|   └── _grid.scss
|
├── pages/
|   ├── _home.scss
|   └── _contact.scss
|
├── utils/
|   ├── _variables.scss
|   ├── _mixins.scss
|   └── _functions.scss
|
└── main.scss
```

7-1 Pattern

Mẫu 7-1 (7-1 pattern) là một kiến trúc tổ chức tệp tin thường được sử dụng trong các dự án SCSS lớn để duy trì sự gọn gàng, dễ quản lý và mở rộng mã nguồn. Cái tên "7-1" xuất phát từ cách chia cấu trúc thành 7 thư mục chính và 1 tệp SCSS đầu vào chính.

Cấu trúc 7-1 gồm 7 thư mục con, mỗi thư mục có nhiệm vụ tổ chức một nhóm cụ thể các quy tắc SCSS, và 1 tệp SCSS chính để nhập tất cả các tệp con từ các thư mục này.

```
scss/
|
|- abstracts/
|   |- _variables.scss        # Biến
|   |- _functions.scss        # Hàm
|   |- _mixins.scss           # Mixin
|
|- base/
|   |- _reset.scss            # Reset/Normalize CSS
|   |- _typography.scss       # Quy tắc cơ bản cho typography
```

```
|
|- components/
|   |- _buttons.scss          # Các thành phần như button, card
|   |- _nav.scss              # Thanh điều hướng
|
|- layout/
|   |- _header.scss           # Định nghĩa header
|   |- _footer.scss           # Định nghĩa footer
|   |- _grid.scss             # Grid layout
|
|- pages/
|   |- _home.scss             # CSS cho trang chủ
|   |- _about.scss            # CSS cho trang about
|
|- themes/
|   |- _default.scss          # Giao diện mặc định
|   |- _dark.scss             # Giao diện dark mode
|
|- vendors/
|   |- _bootstrap.scss        # CSS từ thư viện bên thứ ba (ví dụ:
Bootstrap)
|
`- main.scss                # Tệp chính để import tất cả các tệp SCSS
khác
```

Trong đó:

- **abstracts/**: Chứa các tệp SCSS chứa biến, hàm, mixin, không tạo ra CSS đầu ra.
- **base/**: Chứa các tệp SCSS chứa các quy tắc CSS cơ bản như reset, typography.
- **components/**: Chứa các tệp SCSS chứa các quy tắc CSS cho các thành phần giao diện như button, card, nav.
- **layout/**: Chứa các tệp SCSS chứa các quy tắc CSS cho layout như header, footer, grid.
- **pages/**: Chứa các tệp SCSS chứa các quy tắc CSS cho từng trang cụ thể.
- **themes/**: Chứa các tệp SCSS chứa các quy tắc CSS cho các chủ đề giao diện khác nhau.
- **vendors/**: Chứa các tệp SCSS chứa các quy tắc CSS từ các thư viện bên thứ ba như Bootstrap.
- **main.scss**: Tệp chính để import tất cả các tệp SCSS khác.

Trong thực tế nên linh hoạt trong việc tổ chức cấu trúc thư mục tùy thuộc vào dự án cụ thể, không nhất thiết cần phải tuân thủ một cách cứng nhắc theo 7-1 pattern khi gặp những dự án nhỏ.

Operator (toán tử)

SCSS cho phép thực hiện các phép toán với các giá trị số, màu sắc, v.v. Điều này rất hữu ích khi tính toán kích thước, khoảng cách, hoặc làm sáng/tối màu sắc.

```scss
$base-font-size: 16px;
$base-margin: 10px;
```

```
body {
  font-size: $base-font-size;
  margin: $base-margin * 2;
}

.button {
  width: 100px + 20px;
  background-color: lighten($primary-color, 10%);
}
```

Ở đây, SCSS thực hiện phép toán với kích thước và làm sáng màu $primary-color lên 10%.

Loop and Conditional

SCSS cung cấp các điều kiện (như @if, @else, @for, @each) giúp tạo ra các quy tắc CSS linh hoạt hơn.

```
@mixin responsive($device) {
  @if $device == 'mobile' {
    @media (max-width: 600px) {
      @content;
    }
  } @else if $device == 'tablet' {
    @media (max-width: 768px) {
      @content;
    }
  }
}

.container {
  @include responsive('mobile') {
    background-color: red;
  }

  @include responsive('tablet') {
    background-color: blue;
  }
}
```

Responsive website với SCSS

1. Việc lồng nhau các selector giúp bạn tạo ra các quy tắc CSS cho từng thiết bị một cách dễ dàng.

```
.container {
  width: 100%;

  @media (min-width: 768px) {
    width: 80%;
  }
```

```
  @media (min-width: 1024px) {
    width: 60%;
  }
}
```

2. Sử dụng biến cho các điểm ngắt:

```
$breakpoint-mobile: 600px;
$breakpoint-tablet: 768px;
$breakpoint-desktop: 1024px;

.container {
  width: 100%;

  @media (min-width: $breakpoint-tablet) {
    width: 80%;
  }

  @media (min-width: $breakpoint-desktop) {
    width: 60%;
  }
}
```

Biến $breakpoint-mobile, $breakpoint-tablet, $breakpoint-desktop giúp quản lý điểm ngắt dễ dàng hơn. Khi cần thay đổi kích thước điểm ngắt, chỉ cần cập nhật giá trị của các biến.

3. Mixin cho responsive:

Bạn có thể tạo mixin để tái sử dụng các đoạn mã media queries cho các kích thước màn hình khác nhau. Điều này làm cho mã nguồn dễ bảo trì và gọn gàng hơn.

```
// Định nghĩa mixin cho các điểm ngắt
@mixin respond-to($breakpoint) {
  @if $breakpoint == mobile {
    @media (max-width: 600px) {
      @content;
    }
  } @else if $breakpoint == tablet {
    @media (max-width: 768px) {
      @content;
    }
  } @else if $breakpoint == desktop {
    @media (max-width: 1024px) {
      @content;
    }
  }
}
```

```
// Sử dụng mixin
.container {
  width: 100%;

  @include respond-to(tablet) {
    width: 80%;
  }

  @include respond-to(desktop) {
    width: 60%;
  }
}
```

Mixin `respond-to` giúp bạn không phải viết lại media queries cho mỗi kích thước màn hình, và khi cần thay đổi kích thước, chỉ cần chỉnh lại trong phần định nghĩa mixin.

4. Font chữ responsive:

Với SCSS, bạn có thể làm typography (kiểu chữ) trở nên responsive bằng cách sử dụng phép toán với đơn vị rem hoặc vw (viewport width).

```
$base-font-size: 16px;

body {
  font-size: $base-font-size;

  @media (min-width: 768px) {
    font-size: $base-font-size + 2px;
  }

  @media (min-width: 1024px) {
    font-size: $base-font-size + 4px;
  }
}
```

Ở đây, kích thước font sẽ tăng dần khi màn hình lớn hơn. Bạn có thể sử dụng đơn vị rem hoặc vw để thay đổi font chữ dựa trên kích thước viewport.

Còn rất nhiều cách khác tuỳ vào kinh nghiệm người xây dựng giao diện có nhiều kinh nghiệm đến đâu khi sử dung SCSS. Chúc các bạn sớm thành thạo và tích luỹ cho mình những tips, tricks riêng.

Kết hợp quy tắc đặt tên BEM với SCSS

SCSS có tính năng lồng nhau (nesting) rất phù hợp khi sử dụng cùng với BEM. Bạn có thể sử dụng các quy tắc lồng nhau trong SCSS để tổ chức các lớp theo cách dễ đọc, tuân theo quy tắc đặt tên của BEM.

```
// Block: .menu
.menu {
```

```
    display: flex;
    list-style: none;

    // Element: .menu__item
    &__item {
      margin-right: 10px;
    }

    // Element: .menu__link
    &__link {
      text-decoration: none;
      color: #333;

      // Modifier: .menu__link--active
      &--active {
        color: blue;
      }
    }
  }
```

hoặc:

```
$primary-color: #3498db;
$secondary-color: #2ecc71;

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

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

  &--secondary {
    background-color: $secondary-color;
    color: white;
  }

  &__icon {
    margin-right: 5px;
  }
}
```

Kiến thức cần nhớ

1. SASS là một ngôn ngữ mở rộng của CSS, giúp viết CSS một cách linh hoạt và dễ quản lý hơn.

2. SCSS là một cú pháp mới của SASS, giống với CSS truyền thống, giúp dễ tiếp cận hơn với người mới học.
3. SCSS hỗ trợ nhiều tính năng mạnh mẽ như biến, lồng nhau, mixin, function, extends, import, operator, loop, conditional, v.v.
4. Kết hợp SCSS trong xử lý responsive và quy tắc đặt tên BEM giúp mã nguồn trở nên dễ đọc và bảo trì hơn.