



Mục Tiêu

01 Tại sao sử dụng SASS ?

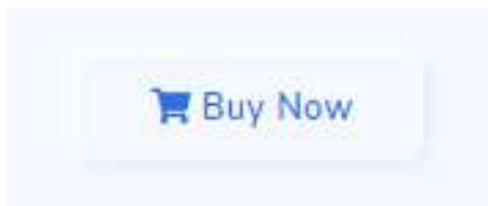
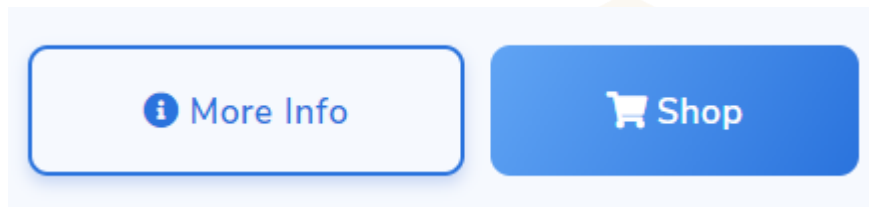
02 SASS là gì ?

03 Thực hành SASS thông qua layout

Tại sao sử dụng SASS ?

Tình huống 1: Trong website có nhiều kiểu button khác nhau và được đặt ở nhiều vị trí. Các kiểu button này có thuộc tính css giống nhau nhưng chỉ khác giá trị (màu, font-size, border...).

Nếu khách hàng muốn đổi toàn bộ button thì chúng ta làm thế nào? Chúng ta sẽ vào code và tìm từng button để thay đổi?

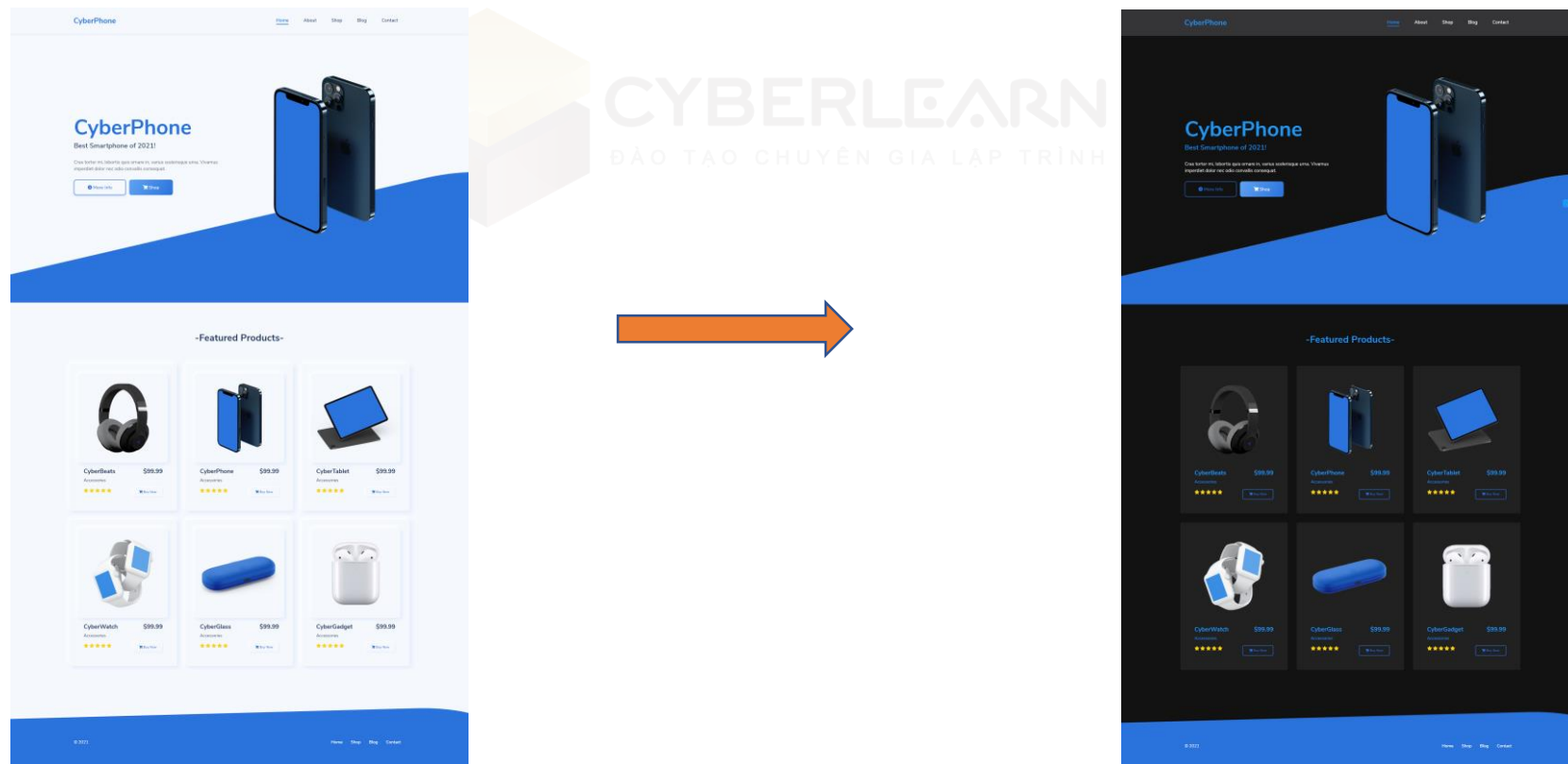


```
background: transparent;  
padding: 13.5px 44px;  
border: 2px solid #2a73dd;  
border-radius: 8px;
```

Tại sao sử dụng SASS ?

Tình huống 2: Trang web của khách hàng đang sử dụng kiểu light theme (màu sáng), họ muốn bổ sung thêm kiểu dark theme (màu tối) để người dùng có nhiều sự lựa chọn.

1. **Vậy chúng ta làm cách nào để xây dựng thêm theme với CSS? Có phải chúng ta sẽ copy code của light theme rồi tìm và thay đổi các thuộc tính màu?**
2. **Giả sử khách hàng muốn thêm nhiều theme hơn (màu hồng, theme dành cho các dịp lễ...). Vậy chúng ta làm sao để tạo theme nhanh chóng và quản lý tốt hơn?**



Tại sao sử dụng SASS ?

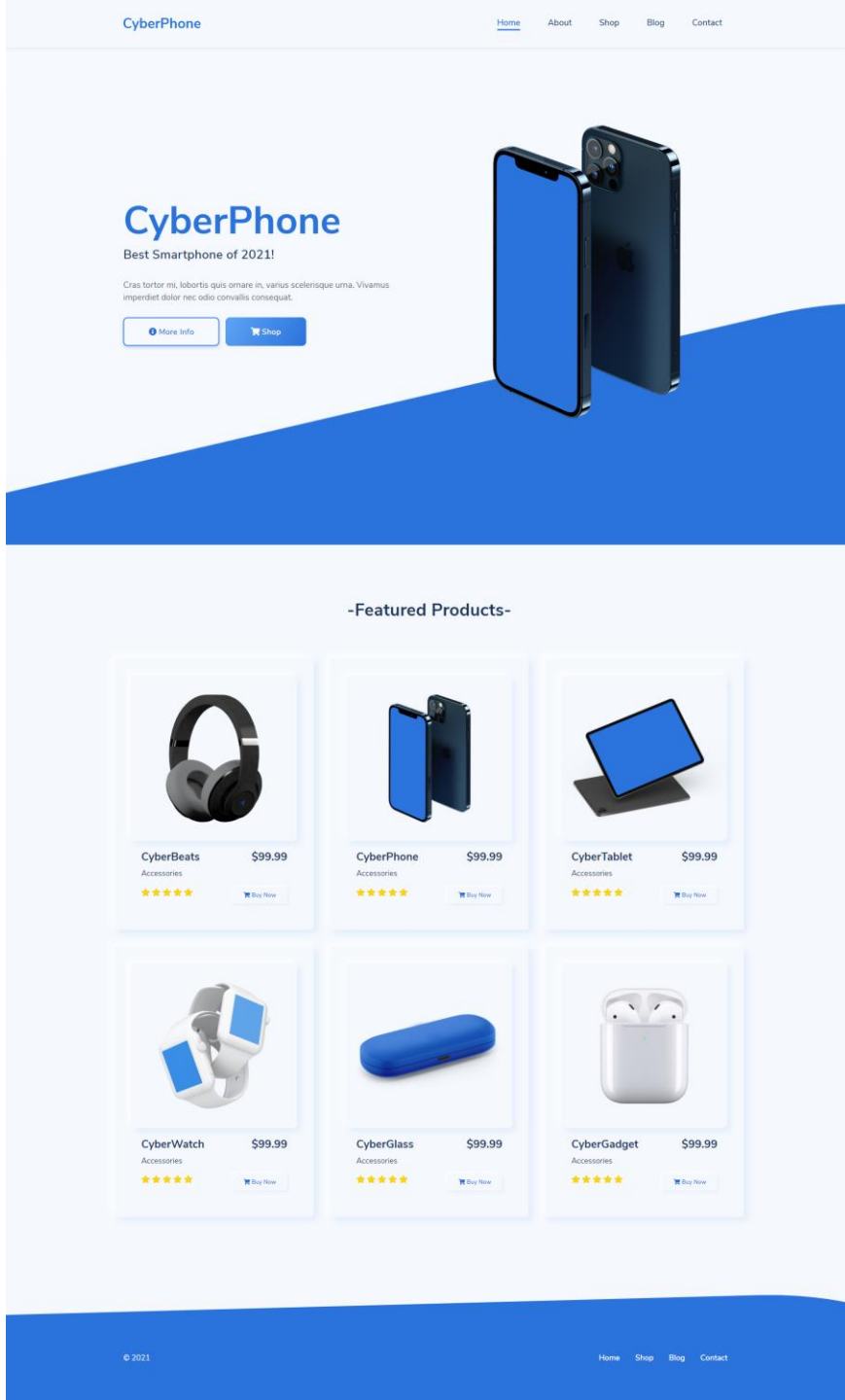
Giải pháp:

- SASS cho phép khai báo các giá trị css thành các biến giúp cho việc chỉnh sửa dễ dàng và tiết kiệm thời gian.
- Chúng ta có thể xây dựng các Mixin (tương tự như hàm) để sử dụng ở mọi nơi và có thể truyền tham số như hàm
- SASS giúp quản lý code dễ dàng hơn nhờ cấu trúc lại code CSS. Chia code thành từng file riêng dựa vào chức năng và nội dung.
- Các đoạn CSS giống nhau (Code Block) có thể được gom nhóm để quản lý, tái sử dụng tránh bị trùng lặp và dư thừa code.

SASS là gì?

- Là CSS processor: công cụ để tạo ra các tập tin css nhanh hơn thông qua một ngôn ngữ khác (scss, less, ...).
- SASS có hai định dạng file là:
 - *.**sass** (viết gần giống ruby vì nó được phát triển bởi các lập trình viên ruby, cách viết phải tuân thủ các nguyên tắc thụt đầu dòng).
 - *.**scss** (phiên bản mới của sass bắt đầu từ version 3.0 cách viết gần với css hơn để lập trình hơn).
 - Tuy cách viết khác nhau nhưng cách định nghĩa các control hay function vẫn mang ý nghĩa tương tự

Tóm lại sass là một bộ công cụ giúp chúng ta định nghĩa và tổ chức css theo phong cách lập trình hơn, hệ thống hơn và tối ưu css hơn. Nhưng kết quả cuối cùng sau khi build file sass là ta được file css.



Layout dự án

01 Tổ chức thư mục

02 Tổ chức biến

03 Tạo cấu trúc trang

04 Biên dịch scss sang css

05 Chia bố cục trang web

06 Tiến hành dàn layout sử dụng mixin,
extend ...

[Link hình ảnh của layout](#)

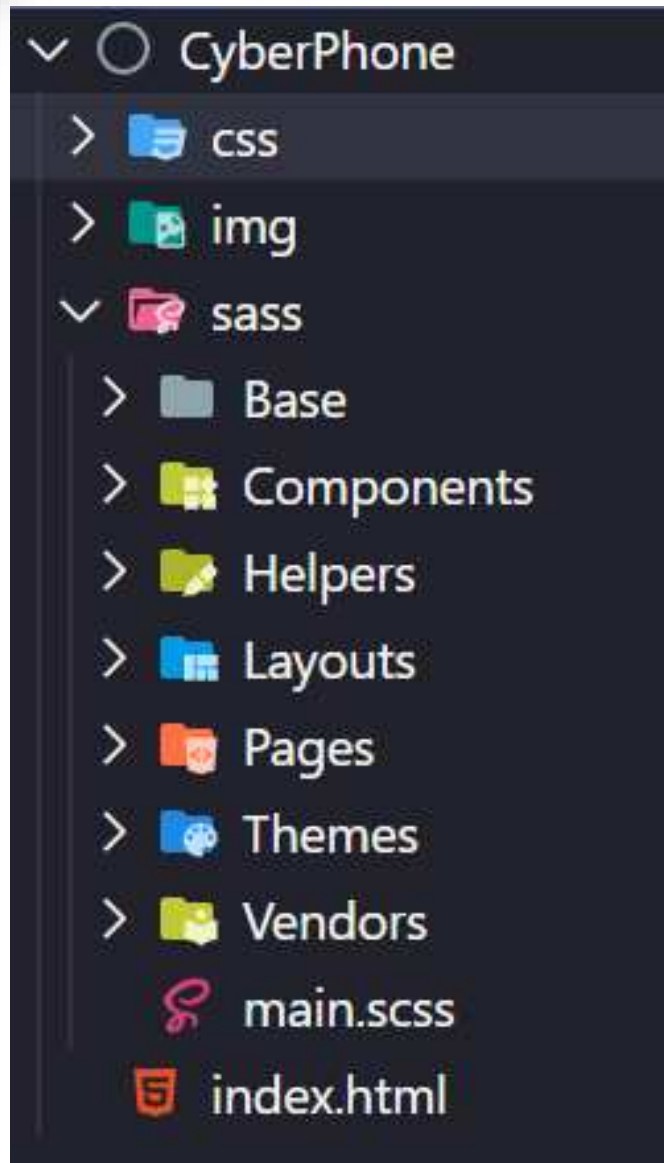


01

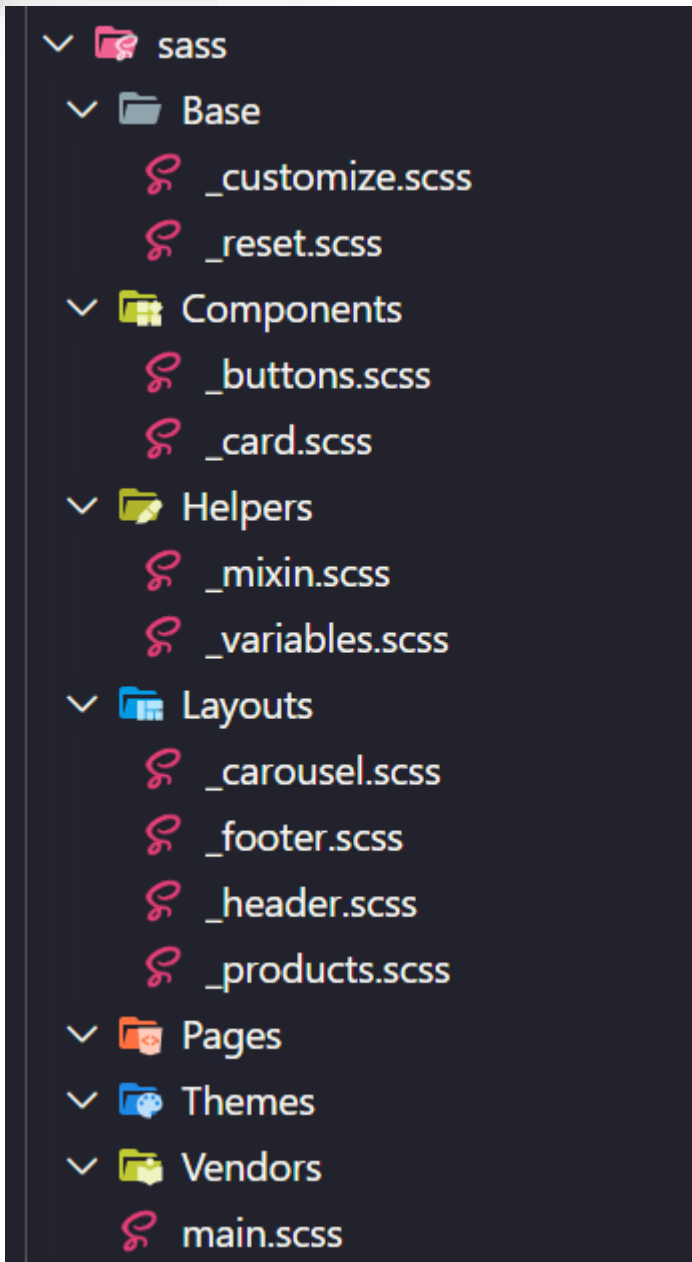


CYBERLEARN
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

Tổ chức thư mục



1. Tạo folder chính (CyberPhone) cho dự án
2. Tạo các file và folder con:
 - **Folder SASS:** chứa file main.scss và 7 folder con (quy tắc 7-1)



7 folders trong scss

- **Base:** chứa những file chỉnh sửa lại mặc định của trình duyệt hoặc của thư viện
- **Helpers:** chứa những file variables, mixin, extend, function ...
- **Components:** chứa những file phần tử nhỏ của trang web (button, dropdown,..)
- **Layouts:** chứa những thành phần lớn dùng chung của trang web.(header, footer, sidebar..)
- **Pages:** chứa style riêng của những page khác nhau. (home, contact, about...)
- **Themes:** chứa những theme chủ đạo khác nhau của trang web
- **Vendors:** chứa file scss của những thư viện mà chúng ta sử dụng

```

//Helpers
@import "./Helpers/variables";
@import "./Helpers/mixin";

//Vendors

//Base
@import "./Base/reset";

//Components
@import "./Components/buttons";
@import "./Components/card";

//Layouts
@import "./Layouts/header";
@import "./Layouts/carousel";
@import "./Layouts/products";
@import "./Layouts/footer";

//Pages

//Theme

```

3. Thêm đường dẫn của các file scss con vào file main.scss bằng @import
- Các file scss có dấu “_” đằng trước. Để biểu thị cho việc khi chúng ta complie ra file css thì các file đó sẽ không complie trực tiếp ra mà sẽ complie thông qua file main.scss.
 - Các file phải import theo thứ tự nhất định:
 - Helpers
 - Vendors
 - Base
 - Components
 - Layouts
 - Pages
 - Themes

02



CYBERLEARN
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

Tổ chức biến

Tổ chức biến?

- Biến sẽ được đặt theo syntax sau:
`$variable_name: value;`
- Quy tắc đặt tên biến:
 - Đặt theo chỉ số hoặc tên màu tương ứng → dễ thêm, hoặc dễ chỉnh sửa theo ý mình.
`$color-black: black; //màu đen`
`$fs-0: 15px; //font-size 15px`
- Xác định những giá trị css thỏa các điều kiện bên dưới để tạo thành biến:
 - Giá trị đó phải được sử dụng ít nhất 2 lần
 - Những giá trị có thể thường xuyên thay đổi trong tương lai
 - Phân ra từng loại biến: color, font-size ,padding, margin, border-radius, line-height, font-weight, break points ...

```
//color
$color-dark-1: #656774;
$color-white-1: #f5f9fe;
$color-white-2: #e2ebfc;
$color-white-3: #fff;
$color-blue-1: #1d365e;
$color-blue-2: #2a73dd;
$color-blue-3: #c9d8f4;
$color-yellow-1: #f9d412;
```

Variable color

```
//font size
$fs-1: 14px;
$fs-2: 1.1rem;
$fs-3: 1.8rem;
$fs-4: 70px;
$fs-5: 1.6rem;
$fs-6: 2.25rem;
$fs-7: 20px;
$fs-8: 44px;
```

Variable font size



03



CYBERLEARN
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

Biên dịch scss sang css

Các công cụ giúp biên dịch

- **Koala:** là phần mềm miễn phí, dễ sử dụng
- **gulp.js:** là thư viện mã lệnh, cần phải biết javascript mới có thể sử dụng
- **Live Sass Compiler:** extension hỗ trợ của VS code
- **JavaScript Framework:** các framework như Angular và Reactjs được tích hợp sẵn công cụ giúp biên dịch sass

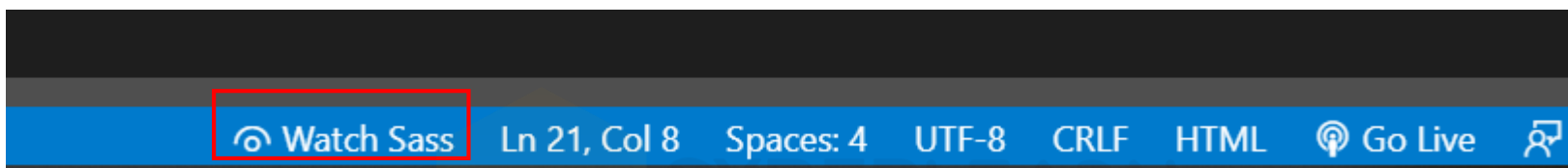
Cài đặt và sử dụng Live Sass Compiler

- Search **Live Sass Compiler** và install vào VS code
- Xét đường dẫn tới folder css của dự án:
 - Mở file **setting.json** trong VS code (Settings -> Extensions-> Live Sass Compile Config)
 - Thêm đoạn code bên dưới vào file setting.json
 - Thêm đường dẫn tương đối tới folder css (chứa file output từ main.scss) vào **savePath**

```
"liveSassCompile.settings.formats":[  
  // This is Default.  
  {  
    "format": "expanded",  
    "extensionName": ".css",  
    "savePath": "~/../css/"  
  },  
]
```

Cài đặt và sử dụng Live Sass Compiler

- Biên dịch file main.scss sang css:
 - Mở main.scss và click **Watch Sass** từ Statusbar



- Sau khi biên dịch, trong folder css sẽ xuất hiện 2 file:
 - _ main.css: file chứa code css của cả dự án.
 - _ main.css.map: liên kết giữa file scss và css, giúp developer dễ dàng xem và debug code sass trên developer tool của các trình duyệt.
- Import file main.css vào html

```
....<!-- Main CSS -->
....<link rel="stylesheet" href="._css/main.css">
```



04



CYBERLEARN
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

Chia bố cục trang web

```
<header> ...  
</header>  
<section class="phoneCarousel"> ...  
</section>  
<section class="products container section"> ...  
</section>  
<footer class="footerPhone"> ...  
</footer>
```

Layout có 4 phần chính:

- 2 phần chính luôn xuất hiện ở các website là header và footer
- Các phần chính ở giữa chúng ta chia theo nội dung của từng phần. Các phần này tạo bằng thẻ section và đặt tên class theo nội dung của phần đó, để dễ quản lý và thêm css.



05



CYBERLEARN
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

Các kiến thức cần nắm

Gom nhóm các selector để định nghĩa

- Ở phần HTML, ta sẽ thấy các thẻ được lồng vào nhau, và có cấu trúc phân cấp.
- Thì bên SASS cũng định nghĩa cấu trúc lồng nhau như bên HTML cho bên scss.

```
<header>
  <nav class="phoneNav container navbar navbar-expand"
    <a class="navbar-brand" href="#">CyberPhone<
    <button class="navbar-toggler" type="button"
    </button>

    <div class="collapse navbar-collapse" id="ph
    </div>
  </nav>
</header>
```

```
header {
  background-color: $color-white-1;
  box-shadow: 0 2px 4px $color-white-2;
  position: fixed;
  width: 100%;
  z-index: 10;
  .navbar-brand {
    @include styleText($fs-3, $fw-3, $color-blue-2);
  }
  .navbar-toggler-icon {
    font-size: $fs-5;
    line-height: 1.5em;
    color: $color-blue-1;
  }
  #phoneNavbar { ...
}
}
```

Parent selector

- Khi chúng ta dùng lại chính selector để làm những sự kiện như hover, focus. Hoặc thậm chí có thể nối thêm tên vào thì chúng ta sẽ dùng parent selector.
- Parent selector sẽ đại diện cho cha trước nó một bậc.

```
#phoneNavbar {  
  .nav-link {  
    @include styleText($fs-2, $fw-2, $color-blue-1);  
    padding: 25px;  
    transition: 0.2s;  
    &::after { ← Parent selector "&"  
      content: "";      đại diện cho nav-link  
      width: 100%;      đứng đằng trước nó  
      height: 2px;  
      display: block;  
      background-color: $color-blue-2;  
      transform: scaleX(0);  
      transition: 0.2s;  
    }  
    &:hover {  
      @include styleText($fs-2, $fw-2, $color-blue-1);  
      &::after {  
        transform: scaleX(1);  
      }  
    }  
  }  
}
```

Mixin

- Mixin là một tính năng của SASS. Điểm chính của mixin chính là khả năng tái sử dụng và là một “DRY” (Don’t Repeat Yourself) Component
- Trong mixin chúng ta sẽ bỏ vào đó những thuộc tính hay được sử dụng để định nghĩa một thành phần html nào đó.

```
@mixin styleText($fs,$fw,$color) {  
    font-size:$fs ;  
    font-weight:$fw ;  
    color:$color ;  
}  
  
@mixin styleButton($bg,$pd-y,$pd-x,$bd, $radius,$ls) {  
    background: $bg;  
    padding: $pd-y $pd-x;  
    border: $bd;  
    border-radius: $radius;  
    letter-spacing: $ls;  
}
```

Ở đây mình định nghĩa ra 2 mixin, 1 mixin text, 1 mixin button. Và có truyền những tham số tương ứng vào.

Mixin

- Chúng ta sẽ di chuyển đến selector chúng ta muốn định nghĩa.
- Sử dụng `@include` để gọi mixin ra.
- Một điểm lưu ý là khi truyền giá trị vào thì phải đúng thứ tự như khi chúng ta định nghĩa

```
html,body{
  font-family: 'Nunito Sans', sans-serif;
  @include styleText($fs-1,$fw-1,$color-dark-1);
  background-color: $color-white-1;
}
```

Extend

- Extend tương tự như mixin nhưng khác mixin ở chỗ không thể thay đổi giá trị của thuộc tính như trong mixin
- Extend kế thừa tất cả những thuộc tính của class đã được định nghĩa sẵn.

```
.background_cover{  
  background-size: cover;  
  background-position: center center;  
  background-repeat: no-repeat;  
  height: 500px;  
}
```

```
.cover{  
  background-image: url("../img/cover.jpg");  
  @extend .background_cover;  
}
```

- background_cover là class chúng ta tự định nghĩa ra.
- Khi xài chúng ta gọi @extend ở trong element chúng ta muốn định dạng cho nó.

Mixin vs Extend

Có 3 khía cạnh để chúng ta so sánh:

- Cách mà mixin và extend sinh ra
- Cách sử dụng của mixin và extend. Khi nào nên sử dụng?
- Ưu, nhược điểm của từng loại là gì?





06



CYBERLEARN
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

Responsive Web with SASS

Biến Mảng

- Khi làm responsive chúng ta sẽ có nhiều loại màn hình khác nhau (xem lại bootstrap)
- Thì khi đó chúng ta sẽ quản lý tất cả thể loại màn hình đó thông qua một mảng biến

```
//break points  
$breakpoints:(  
  'extra-large': 1200px,  
  'large' : 992px,  
  'medium' : 768px,  
  'small' : 576px,  
)
```

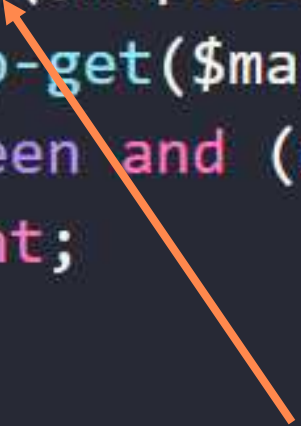
Thay vì chúng ta gán trực tiếp giá trị cho biến, thì bây giờ nó sẽ là một mảng theo từng cặp key:value và được cách nhau bởi dấu “,”

Biến Mảng

- Để sử dụng được biến mảng đó, ta sẽ thông qua hàm map-get(\$map, \$key) → thì sẽ tự động trả về giá trị value
- Ví dụ: map-get(\$breakpoints, 'medium') → sẽ tự trả về giá trị 768px
- Để lấy được tất cả giá trị khác nhau, chúng ta sẽ tự định nghĩa luôn thành một mixin

Mixin Responsive

```
@mixin responsive($screen) {  
  @if(map-has-key($map:$breakpoints , $key:$screen )){  
    $value: map-get($map:$breakpoints , $key:$screen );  
    @media screen and (max-width:$value){  
      @content;  
    }  
  }  
  @else{  
    @warn "`{$screen}` isn't in breakpoint";  
  }  
}
```



Hàm map-has-key cho chúng ta biết được là trong biến mảng “breakpoints” có key mà chúng ta truyền vào hay không, và nó sẽ trả về giá trị true hoặc false

Chúng ta sẽ dùng @if để khi nào điều kiện đó đúng thì nó sẽ thực hiện việc trả value cho chúng ta

Mixin Responsive

```
@mixin responsive($screen) {  
  @if(map-has-key($map:$breakpoints , $key:$screen )){  
    $value: map-get($map:$breakpoints , $key:$screen );  
    @media screen and (max-width:$value){  
      @content;  
    }  
  }  
  @else{  
    @warn "`{$screen}` isn't in breakpoint";  
  }  
}
```

Sau khi lấy được giá trị trả về, chúng ta sẽ dùng media query để responsive cho trang web
@content để chỉ nội dung mà chúng ta responsive cho trang web mình

Mixin Responsive

```
@mixin responsive($screen) {  
  @if(map-has-key($map:$breakpoints , $key:$screen )){  
    $value: map-get($map:$breakpoints , $key:$screen );  
    @media screen and (max-width:$value){  
      @content;  
    }  
  }  
  @else{  
    @warn "`{$screen}` isn't in breakpoint";  
  }  
}
```

Tương tự, nếu điều kiện không đúng thì nó sẽ đi vào @else → @warn sẽ xuất hiện khi điều kiện của chúng ta không đúng.

Tóm lại

- @if, @else được sử dụng khi chúng ta có những điều kiện kèm theo nó.
- @warn để xuất hiện cảnh báo khi chúng ta viết code xảy ra lỗi
- @content sẽ nằm trong @media để chứa nội dung css khi chúng ta responsive.



07



CYBERSEARN
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

Những tính năng khác trong SASS

@for và @while

```
// For
$number: 1;

@for $number from 1 to 13 {
  ....col-#{$number}{
  ....width: 100% * $number/12;
  ....}
}

@for $number from 1 through 12 {
  ....col-#{$number}{
  ....width: 100% * $number/12;
  ....}
}
```

Để tạo động class chúng ta có thể dùng vòng lặp for, while giúp tạo ra class một cách nhanh nhất
Ví dụ: Tạo ra một chuỗi các class col-1 tới col-12 trong BS4.

@for và @while

```
// while
$number: 1;
@while $number < 13 {
  .col-#{ $number } {
    width: 100% * $number / 12;
  }
  $number: $number + 1;
}
```

Để tạo động class chúng ta có thể dùng vòng lặp for, while giúp tạo ra class một cách nhanh nhất
Ví dụ: Tạo ra một chuỗi các class col-1 tới col-12 trong BS4.

@each

```
$sizes: 40px, 50px, 80px;

@each $size in $sizes {
  .icon-#{$size} {
    font-size: $size;
    height: $size;
    width: $size;
  }
}

$icons: ("book": "\f2b9", "card": "\f2bb");

@each $name, $code in $icons {
  .icon-#{$name}:before {
    content: $code;
  }
}
```

- @each giúp duyệt các giá trị trong biến mảng, dùng tạo class nhanh. Ví dụ: Tạo font icon riêng dựa trên font awesome

Placeholder Selectors

```
%toolbelt {  
  box-sizing: border-box;  
  border-top: 1px rgba(0, 0, 0, .12) solid;  
  padding: 16px 0;  
  width: 100%;  
  
  &:hover { border: 2px rgba(0, 0, 0, .5) solid; }  
}  
  
.action-buttons {  
  @extend %toolbelt;  
  color: #4285f4;  
}  
  
.reset-buttons {  
  @extend %toolbelt;  
  color: #cddc39;  
}
```

- Placeholder được khai báo **%tên**
- Placeholder thường được dùng để khai báo các thuộc tính và giá trị css được sử dụng nhiều chỗ trong website
- SCSS của placeholder sẽ không được biên dịch sang css. Chỉ có những class kế thừa placeholder qua extend sẽ chứa code css của placeholder.

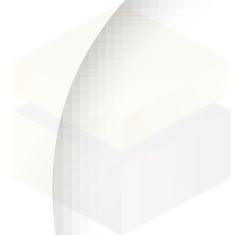
@function

```
@function pow($base, $exponent) {  
  $result: 1;  
  @for $result from 1 through $exponent {  
    $result: $result * $base;  
  }  
  @return $result;  
}  
  
.sidebar {  
  float: left;  
  margin-left: pow(4, 3) * 1px;  
}
```

- @function thường được dùng khi cần tạo giá trị động vì function sẽ return một kết quả mới



```
.sidebar {  
  float: left;  
  margin-left: 64px;  
}
```

CYBERLEARN
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

Thank You