SASS

CYBERSOFT.EDU.VN

Mục tiêu

1. SASS là gì?
2. Lợi ích của việc dùng SASS
3. Cách tổ chức project
4. Thực hành thông qua layout

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 dễ 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 tối ưu css hơn. Nhưng **kết quả cuối cùng** sau khi build file sass => ta được file **css**.

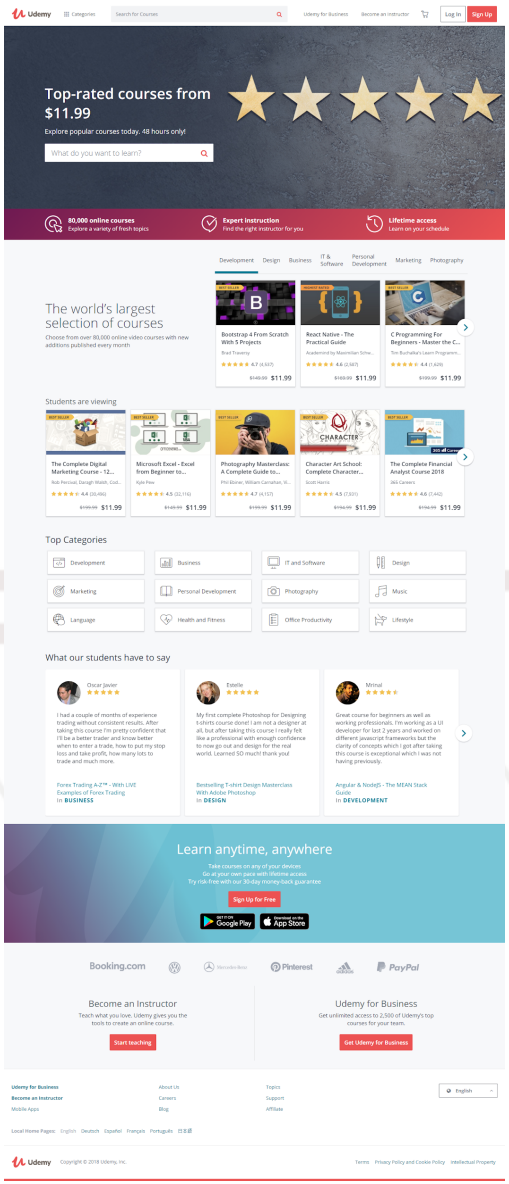
Lợi ích

- Sử dụng các biến như các ngôn ngữ lập trình, dễ chỉnh sửa. Ví dụ như cài đặt các biến chứa màu sắc để sử dụng, khi chỉnh sửa màu chỉ cần chỉnh sửa giá trị biến này và biên dịch lại.
- Tiết kiệm thời gian
- Các đoạn CSS giống nhau (Code Block) có thể được gom nhóm và quản lý, tái sử dụng
- Hỗ trợ cách biểu diễn màu, thuộc tính cho tất cả trình duyệt
- Xây dựng các Mixin(hàm function có thể sử dụng ở tất cả mọi nơi nhưng phải include) có thể truyền tham số tương tự như hàm trong ngôn ngữ lập trình.

Layout dự án

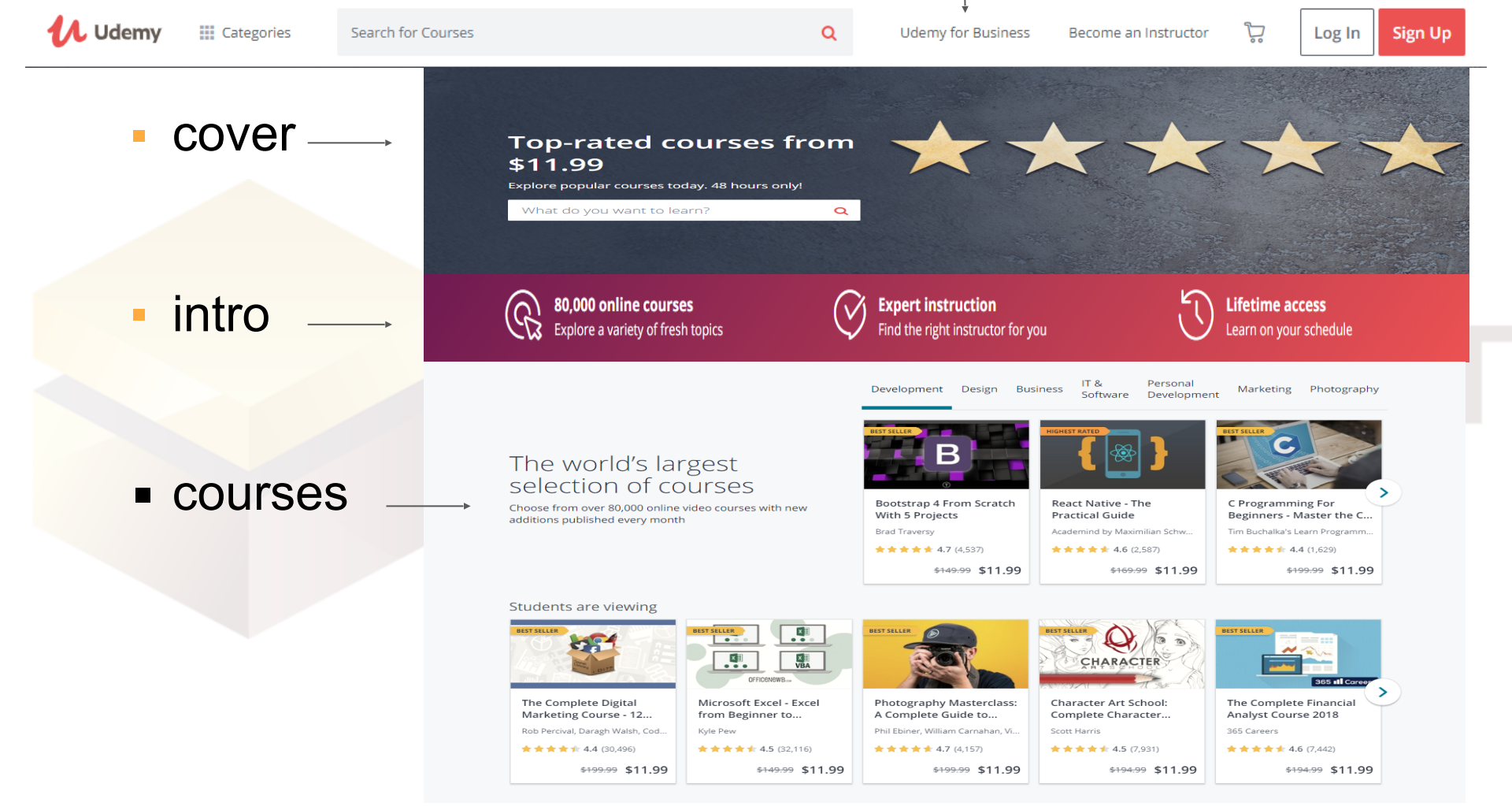
Các bước thực hiện:

1. Phân tích layout (các phần trong trang web hoặc bộ website).
2. Tổ chức thư mục trong SASS
3. Tìm, lấy các biến cần xài trong layout
4. Tiến hành dàn layout sử dụng mixin, extend, placeholder, function ...



Layout dự án

- header
- cover
- intro
- courses



CYBERSOFT
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

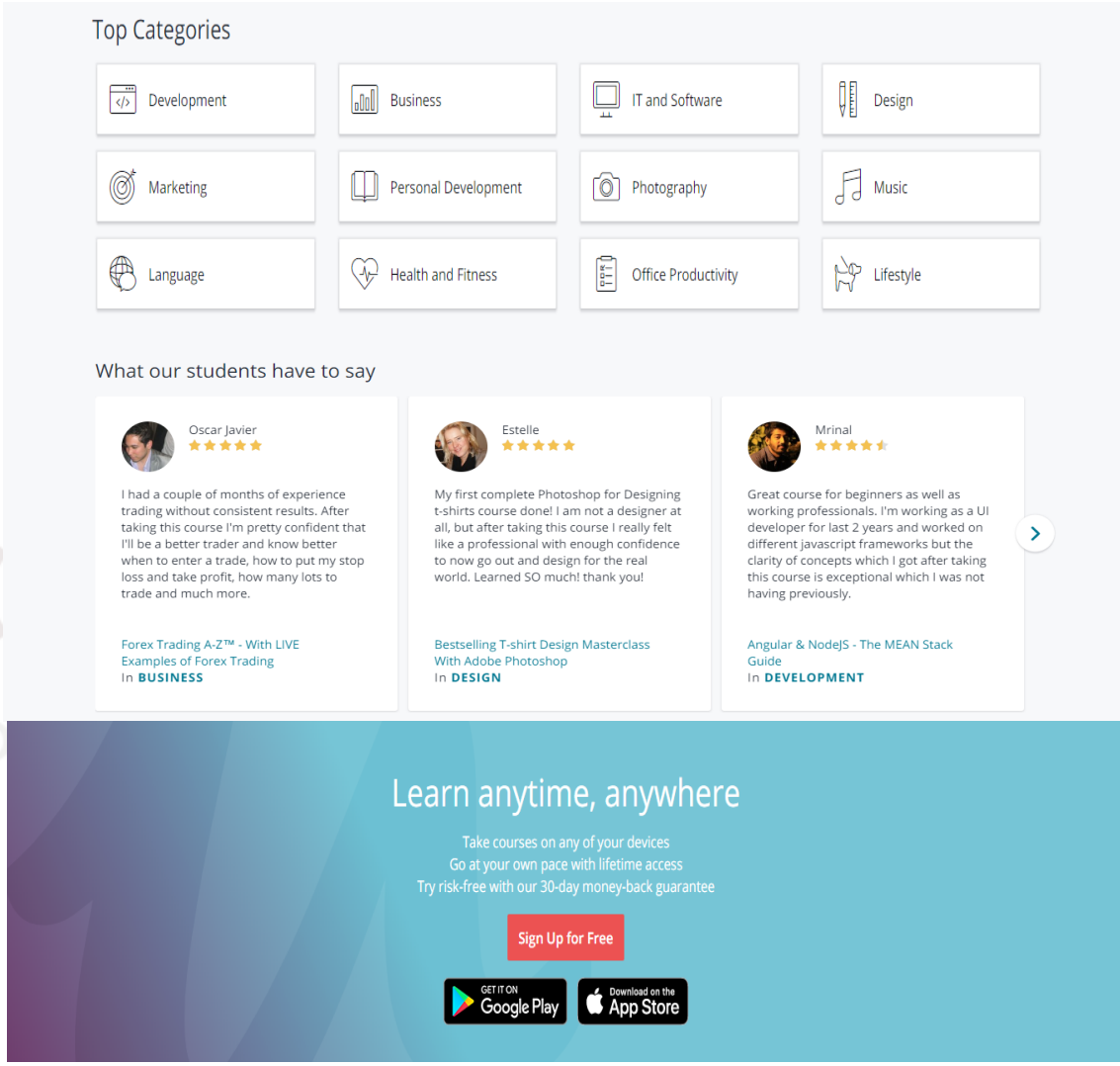
LASS

MYCODER

timviecit.com

Layout dự án

- categories →
- student →
- banner →



CYBERSOFT
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

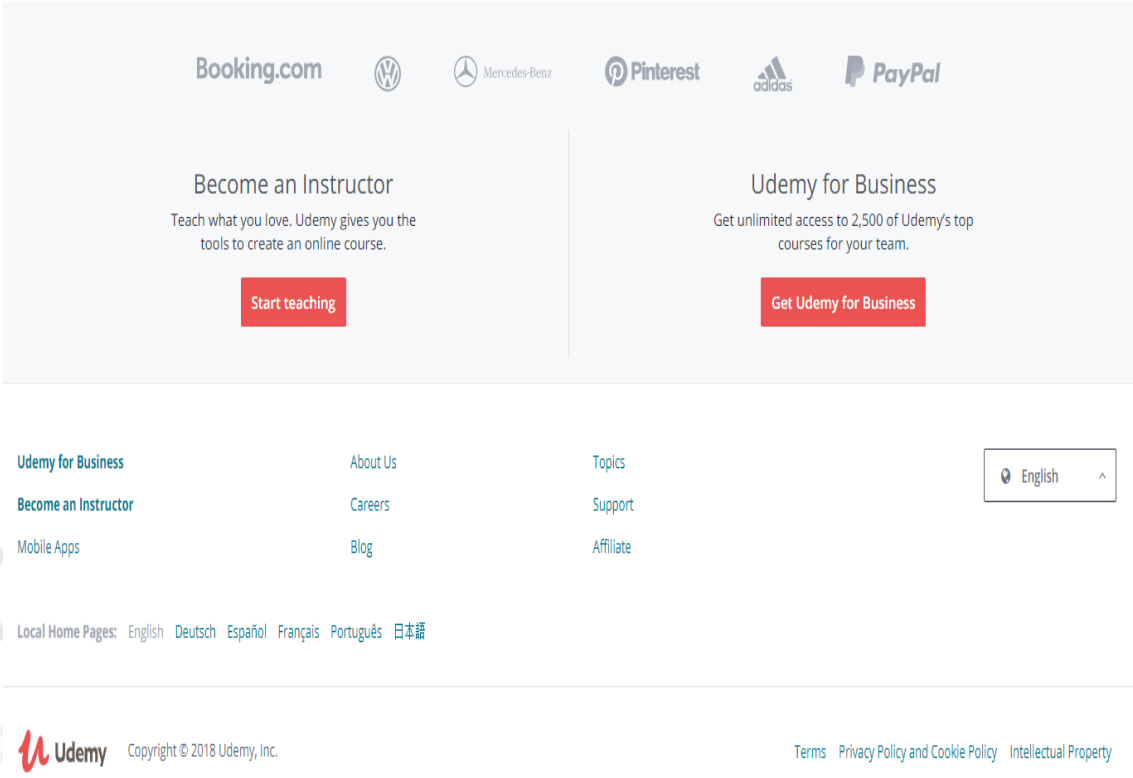
LASS

MYCODER

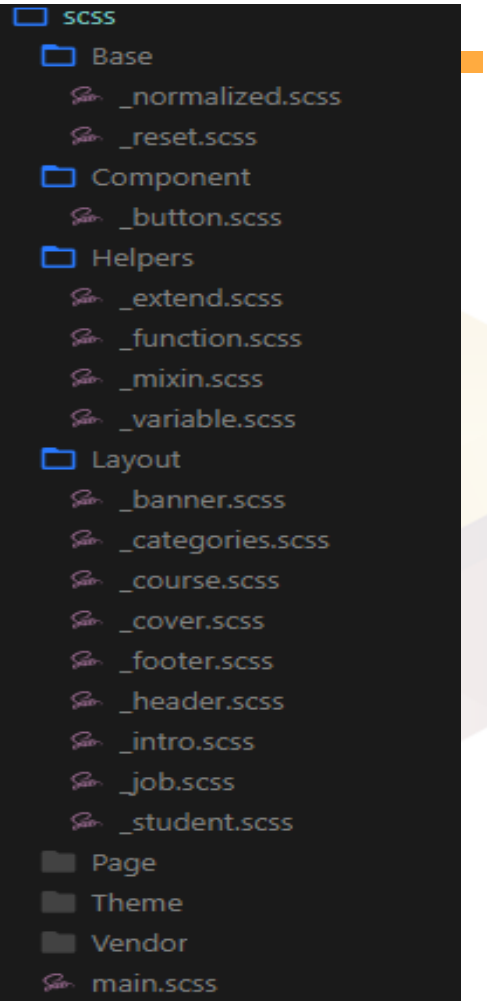
timviecit.com

Layout dự án

- job
- footer

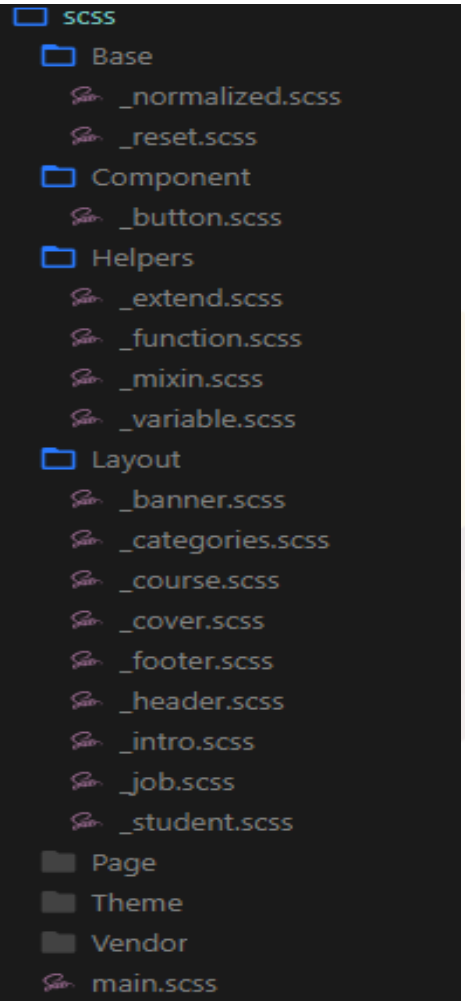


Cách tổ chức dự án



- Ngoài những thư mực css, img, js như bình thường, chúng ta sẽ có thêm thư mục SASS chứa những folder con trong đó
  - Folder “Base” -- chứa những file chỉnh sửa lại mặc định của trình duyệt
  - Folder “Helpers” -- chứa những file biến, mixin, extend, function …
  - Folder “Component” -- chứa những file phần tử nhỏ của trang web (button, dropdown, card …)

Cách tổ chức dự án



- Folder “Layout” -- chứa những thành phần chung của trang web trong bộ website chúng ta xây dựng
- Folder “Page” -- chứa những thành phần riêng của trang web khác nhau. Ví dụ: trang home thì có Introduction, trang liên hệ thì có phần contact …
- Folder “Theme” -- chứa những theme chủ đạo khác nhau của trang web. Ví dụ như trang web đổi theo mùa (đông thì sẽ có màu trắng, tuyết bay …)
- Folder “Vendors” -- chứa những thư viện mà chúng ta sử dụng

Cách tổ chức dự án

main.scss

```scss
@import "Base/_reset.scss";

@import "Helpers/_variable.scss";
@import "Helpers/_mixin.scss";
@import "Helpers/_extend.scss";
@import "Helpers/_function.scss";

@import "Layout/_header.scss";
@import "Layout/_cover.scss";
@import "Layout/_intro.scss";
@import "Layout/_course.scss";
@import "Layout/_categories.scss";
@import "Layout/_student.scss";
@import "Layout/_banner.scss";
@import "Layout/_job.scss";
@import "Layout/_footer.scss";

@import "Component/_button.scss";
```

- File cuối cùng là file main. Quy tắc: file main sẽ không được chứa bất kì một dòng code nào, chỉ chứa @import
- Các file còn lại 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.
- Và import phải theo thứ tự nhất định → "Base" → "Vendors" → "Helpers" → "Layout" → "Component" → "Page" → "Theme"

Cách tổ chức biến

ex.html _variable.scss ×

```
$color-black : black;
$color-white: white;
$color-grey: #999;
$color-blue: #00adef;
$color-dark: #202020;
$color-grey-dark: #1d1d1d;
$color-black-light: rgba(0,0,0,0.5);
$color-black-dark: rgba(0,0,0,0.8);

$pd-0: 5px;
$pd-1: 10px;
$pd-2: 20px;
$pd-3:30px;
$pd-4:4%;
$pd-5:7%;

$mg-1:10px;
$mg-2: 20px;
$mg-3:30px;
$mg-4:40px;
$mg-5:50px;
$mg-6:200px;

$fs-0:13px;
$fs-1:15px;
$fs-2: 17px;
$fs-3:20px;
$fs-4:30px;
$fs-5:40px;

$fw-bold:bold;
$fw-normal:normal;
```

- Biến sẽ được đặt theo syntax sau:
  - $variable_name = value;
- Quy tắc đặt tên biến:
  - Phân ra từng loại biến: biến màu, padding, margin, font-size …
  - Đặt theo chỉ số hoặc tên màu tương ứng → dễ thêm, hoặc dễ chỉnh sửa theo ý mình.
  - Không phải khi nào cũng cần tạo biến
    - Giá trị đó phải được sử dụng ít nhất 2 lần
    - Những giá trị có thể được update (biến màu)

Các khái niệm cần nắm

- **Sử dụng biến (Variable)**
- **Gom nhóm thành phần HTML**
- **Parent Selector**
- **Mixin**
- **Extend**
- **Cấu trúc vòng lặp for**
- **Responsive với SASS**
- **...**

Gom nhóm các selector để định nghĩa

- Ở phần HTML, ta sẽ thấy các thẻ được để 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.

```
<nav class="navbar navbar-expand-md navbar-light myNavBar">
    <div class="col-md-6">
        <div class="row myNavBar__left">
            <a class="navbar-brand" href="#">···
            </a>
            <div class="categories">···
            </div>
            <form action="" class="form_search">···
            </form>
        </div>
    </div>
```

Thẻ nav cha nằm ngoài, các thẻ div tương tự nằm bên trong thẻ nav

Gom nhóm các selector để định nghĩa

```scss
.myNavBar{
    padding: $pd-2;
    .myNavBar__left{
        flex-wrap: wrap;
        .navbar-brand{
            img{
                width: 110px;
            }
            flex-grow: 1;
        }
        .categories{
            @include text($color-black-light, $fs-1, fw-normal);
            flex-grow: 1;
            padding-top: $pd-2 - 5;
            i{
                color: $color-gray-light;
            }
        }
```

Thẻ nav có class .myNavBar sẽ được để ngoài, và ở trong đó chúng ta có thể viết nhưng thẻ con lồng trong theo cấu trúc tương tự như bên HTML

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.

Parent selector

```scss
.overlay{
    display: none;
    position: absolute;
    background-color: $color-white-0;
    padding: $pd-2;
    border: 1px solid $color-gray-2;
    width: 350px;
    top: 56px;
    left : -100%;
    &:before{
        content:"";
        width: 15px;
        height: 15px;
        position: absolute;
        top: -9px;
        right: 10%;
        background-color: $color-white-0;
        transform: rotate(45deg);
        -webkit-transform: rotate(45deg);
        -moz-transform: rotate(45deg);
        -ms-transform: rotate(45deg);
        -o-transform: rotate(45deg);
        border-top: 1px solid $color-gray-2;
        border-left: 1px solid $color-gray-2;
        z-index: 2;
    }
}
```

Parent selector “&” đại diện cho overlay đứng đằng trước nó

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

- 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 có mang tham số (định nghĩa)

```scss
@mixin text($color, $fs, $fw){
    color: $color;
    font-size: $fs;
    font-weight: $fw;
}

@mixin button($fs, $fw, $color, $bg-color, $border, $pd-y, $pd-x, $bg-hover){
    font-weight: $fw;
    font-size: $fs;
    color: $color;
    background-color: $bg-color;
    border: 1px solid $border;
    padding: $pd-y $pd-x;
    &:hover{
        background-color: $bg-hover;
    }
}
```

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

Mixin khi được sử dụng

```
.nav-link{
    @include text($color-black-light, $fs-1, $fw-normal);
    margin-top: $mg-0 +3;
}
```

```
.button__login{
    @include button($fs-0,$fw-bold-0, $color-gray-4,
     $color-white-0, $color-black-light, $pd-1+1,
     $pd-1+2, $color-gray-1);
}
.button__signup{
    @include button($fs-0,$fw-bold-0, $color-white-0,
    $color-red-0, $color-red-0, $pd-1+1,
    $pd-1+2, $color-red-1);
}
```

* 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

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.

Extend

```
.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?

Mixin

- SASS

```scss
@mixin border-radius($radius){
  border-radius: $radius;
  -o-border-radius: $radius;
  -ms-border-radius: $radius;
  -moz-border-radius: $radius;
  -webkit-border-radius: $radius;
}

button{
  @include border-radius(10px)
}

a{
  @include border-radius(10px)
}
```

```css
button {
  border-radius: 10px;
  -o-border-radius: 10px;
  -ms-border-radius: 10px;
  -moz-border-radius: 10px;
  -webkit-border-radius: 10px; }

a {
  border-radius: 10px;
  -o-border-radius: 10px;
  -ms-border-radius: 10px;
  -moz-border-radius: 10px;
  -webkit-border-radius: 10px; }
```

Extend

- SASS

```scss
.border--radius{
  border-radius: 5px;
  -o-border-radius: 5px;
  -ms-border-radius: 5px;
  -moz-border-radius: 5px;
  -webkit-border-radius: 5px;
}

button{
  @extend .border--radius;
}

a{
  @extend .border--radius;
}
```

```scss
.border--radius, button, a {
  border-radius: 5px;
  -o-border-radius: 5px;
  -ms-border-radius: 5px;
  -moz-border-radius: 5px;
  -webkit-border-radius: 5px; }
```

Responsive Web with SASS

- @for
- @each
- @while
- @if
- @warn
- @error
- ....
- Chúng ta sẽ lồng những directives vào khi chúng ta làm responsive

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 cho nó

Mixin Respond

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

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

Mixin Respond

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

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 Respond

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

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.
“Điểm breakpoint đó không có ở trong biến mảng breakpoints”

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 khi chúng ta responsive.
- ...

@for -- Vòng lặp

- Để tạo động class chúng ta có thể vòng for để duyệt tạo ra một cách nhanh nhất

```
@for $number from 1 through 6{
  .delay-#{$number}{
    animation-delay: 0.4s * $number!important;
  }
}
```

Tạo ra một chuỗi các class .delay bằng cách dùng vòng for
Tương tự như đối với class col-1 tới col-12 trong BS4. Cũng dùng vòng for để lặp ra.
Khi sử dụng thì thêm class vào html

Thank You!

CYBERSOFT
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

YLASS

MYCODER

timviecit.com