Bài: Áp dụng SASS vào project website landing page

Xem bài học trên website để ủng hộ Kteam: Áp dụng SASS vào project website landing page

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!

Dẫn nhập

Ở bài học trước chúng ta đã cùng nhau SETUP MÔI TRƯỜNG ĐỂ COMPILE SASS SANG CSS

Từ đó, trong bài học này chúng ta đã có thể bắt tay **Áp dụng SASS vào project website landing page** mà chúng ta đang thực hiện.

Nội dung

Để tiếp thu tốt bài học này, các bạn cần:

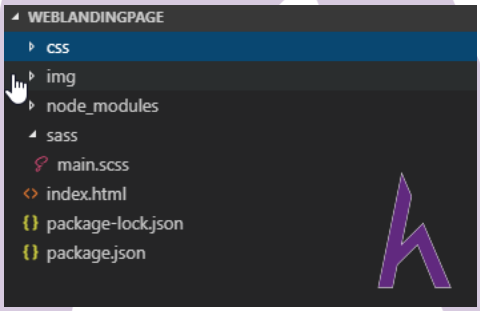
- Học qua bài SETUP MÔI TRƯỜNG ĐỂ COMPILE SASS SANG CSS
- Có kiến thức CSS cơ bản.

Ở trong bài học này chúng ta sẽ cùng nhau:

- Tìm hiểu cách compile Sass sang CSS
- Auto Compile Sass khi lưu code.
- Convert code Css sang Sass

Tìm hiểu cách compile Sass sang CSS

Quay trở lại **command prompt** với **directory.** Trước tiên thì chúng ta sẽ tạo 1 folder **sass** (nơi chứa code sass của chúng ta). Tạo 1 file main.**scss** chứa code sẽ được chúng ta compile sang css



Sau đó chuyển toàn bộ code trong file **style.css** sang **main.scss**

Ở **main.scss** ta sử dụng variables trong sass các biến cho 3 màu chính mà chúng ta sẽ sử dụng trong project

:

```
$color-primary: #5a73fc;
$color-primary-light:#8e9efc;
$color-primary-dark:#1c3eff;
```

Và thay thế giá trị các màu này bằng tên biến vừa đặt (ở **property** của phần **header**)

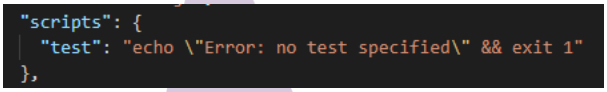
Code Sass

:

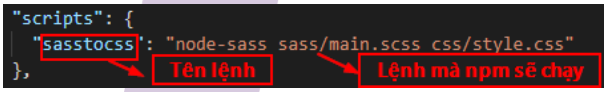
```
.header{
    background-image:
    linear-gradient(to right bottom, rgba($color-primary-light, .8),rgba($color-primary-dark, .8)),
    url('../img/background1.jpg');
    background-size: cover;
    background-

    height: 95vh;
}
```

Vậy điều chúng ta quan bây giờ là làm thế nào để **compile file main.scss** sang **css**

Quay trở lại file **package.json**, chúng ta sẽ thay đổi phần "**test**" "**scripts**" thành lệnh compile sass sang css

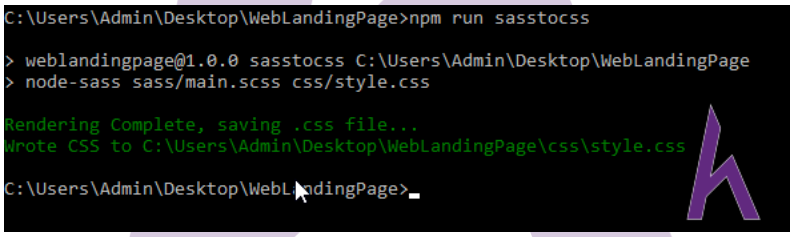


Chuyển sang lệnh compile sass



Ở đây phần tên lệnh là do bạn từ đặt, mình đặt là **sasstocss** để thể hiện là việc compile sass sang css. Phần lệnh chạy gồm **node-sass** (tên package) **sass/main.scss** (địa chỉ file scss) **css/style.css** (địa chỉ để chứa code css sau khi đã compile)

Để chạy compile, chúng ta lại quay lại **command prompt**, sử dụng lệnh **npm run <tên lệnh>**, trong trường hợp này là **npm run sasstocss**



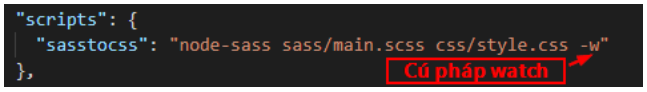
Bây giờ bạn mở file **style.css** ra sẽ thấy code **css** như ban đầu, không còn có các tên biến như trong file **main.scss** nữa. File **index.html** của chúng ta vẫn sẽ sử dụng code ở file **style.css** chứ **không sử dụng được code sass trong main.scss**

Vậy là chúng ta đã biết cách compile sass sang css sử dụng package **node-sass**.

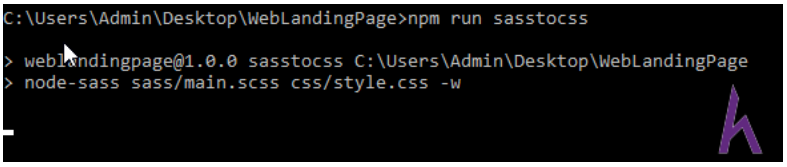
Auto Compile Sass khi lưu code

Như các bạn thấy là sau mỗi lần chúng ta chỉnh sửa code ở **main.scss** thì lại phải chạy lại lệnh **npm run sasstocss**, rất mất thời gian.

Để tránh việc này lặp đi lặp lại, chúng ta sẽ sửa câu lệnh **npm** đi 1 chút



Việc thêm **-w** ở cuối dòng lệnh (viết tắt của watch) là câu lệnh sẽ giúp npm theo dõi, cứ mỗi khi file **main.scss** thay đổi (sau khi bạn save bằng tổ hợp Ctrl +S) thì sẽ tự động compile sang css, vô cùng thuận tiện. Từ đây sau khi chạy lệnh **npm run sasstocss** 1 lần ban đầu và bạn chỉ còn phải tập trung code ở file sass là được.



Convert code CSS sang Sass

Việc convert code CSS ở bài học này sẽ sử dụng VARIABLES VÀ NESTING TRONG SASS.

Sử dụng variables với color

Ta sẽ chuyển tất cả color sử dụng trong project thành variables:

Màu font chữ mặc định: **#777777**

:

```
$color-grey: #777777;
```

Màu font chữ heading: **#fff**

:

```
$color-white: #fff;
```

Màu shadow: **black, rgba(0,0,0,.2)** chuyển thành **rgba($color-black,.2)**

:

```
$color-black: #000;
```

Và việc của chúng ta sẽ thay tất cả các giá trị màu đã code bằng các biến vừa tạo và chạy lệnh bên dưới trong command line

> **npm run sasstocss**

Việc tiếp theo chúng ta sẽ sử dụng nesting trong Sass

Chắc bạn vẫn còn nhớ cách mà Sass sử dụng Nesting trong bài TÌM HIỂU VARIABLES VÀ NESTING TRONG SASS.. Ở đây chúng ta sẽ áp dụng nó để cấu trúc lại cấu trúc của code Sass và thấy được lợi ích của việc sử dụng **BEM**.

Nguyên tắc sẽ là những **class** có chung **B (Block)** thì gom chung lại trong 1 khai báo block, và element trong block đó chung element thì lại tiếp tục gom vào 1 khai báo elemet.

:

```
.<block-name>{
    &__<element-name-1>{
       &--<modified-name>{
       }
    }

    &__<element-name-2>{
       &--<modified-name>{
       }
    }
}
```

Trước hết là phần **.header**:

:

```
.header{
    background-image:
    linear-gradient(to right bottom, rgba($color-primary-light, .8),rgba($color-primary-dark, .8)),
    url('../img/background1.jpg');
    background-size: cover;
    background-

    height: 95vh;
    -webkit-clip-path: ellipse(60% 60% at 50% 33%);
    clip-path: ellipse(60% 60% at 50% 33%);

    &__text-box{

        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
        text-align: center;
    }

    &__logo{
        height: 7rem;
    }

    &__logo-box{

        top: 3rem;
        left: 3rem;
    }
}
```

Tiếp đến là **.heading-primary**

:

```
.heading-primary {
    color: $color-white;
    text-transform: uppercase;
    margin-bottom: 6rem;

    &--main {
        display: block;
        font-size: 6.2rem;
        font-weight: 400;
        letter-spacing: 1.5rem;
        animation: moveInLeft 1s ease-out;
    }

    &--sub {
        display: block;
        font-size: 2rem;
        font-weight: 700;
        letter-spacing: 1.8rem;
        animation: moveInRight 1s ease-out
    }
}
```

> **Chú ý: pseudo-class** vẫn có thể sử dụng tương tự nhé, kí tự & đơn giản chỉ thay thế cho phần text được khai báo bên ngoài.

Ví dụ:

:

```
block{
    &_test{
     color: red;
    }
}
```

Tương đương với

:

```
block_test {
  color: red; }
```

Mặc dụ khai báo trên là sai do **block_test** không phải là tên của 1 element trong html

Phần tiếp cuối cùng là **.btn**:

:

```
.btn {

    &:link,
    &:visited {
        text-transform: uppercase;
        text-decoration: none;
        padding: 1.5rem 4rem;
        display: inline-block;
        border-radius: 10rem;
        font-size: 1.6rem;

        transition: all .2s;
    }

    &::after {
        content: "";
        display: inline-block;
        height: 100%;
        width: 100%;

        top: 0;
        left: 0;
        z-index: -1;
        border-radius: 10rem;
        transition: all .4s;
    }

    &:hover {
        transform: translateY(-.3rem);
        box-shadow: 0 .5rem 2rem rgba($color-black, .2);

        &::after {
            transform: scaleX(1.4) scaleY(1.6);
            opacity: 0;
        }

    }

    &:active {
        outline: none;
        transform: translateY(-.1rem);
        box-shadow: 0 .5rem 1rem rgba($color-black, .2);
    }

    &--white {
        color: $color-grey;
        background-color: $color-white;

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

    &--animated {
        animation: moveInBottom .5s ease-out .75s;
        animation-fill-mode: backwards;
    }
}
```

Kết luận

Ở bài này chúng ta đã hiểu cách sử dụng Node-sass compile SASS sang CSS và ứng dụng vào trong project của chúng ta.

Ở trong bài học sau chung ta sẽ cùng quay trở lại với project WebLandingPage để CHIA FILE MAIN.SCSS THÀNH CÁC COMPONENT.

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận của mình để giúp phát triển bài viết tốt hơn. Đừng quên: "**Luyện tập – Thử thách – Không ngại khó**".