# Tổng quan về Front-end

## Cấu trúc Website

## HTML, CSS

#### Các thẻ cơ bản của HTML

H1 – H6: Tên tiêu đề của trang website. Thường dùng 1 thẻ h1 duy nhất cho tiêu đề của trang đó

P: Thẻ dùng để viết đoạn văn

Img: Hiển thị ảnh

A: Chèn liên kết vào website

Ul, li: Hiển thị dạng danh sách

Table: Tạo bảng có các thẻ thead và tbody. Trong tbody gồm có tr và td

Input: From nhập liệu

Button: Tạo ra các nút

Div: Tạo ra các khối

#### Các thẻ cơ bản của CSS

#### SASS

#### Quy tắc đặt tên BEM

# 3. CSS

## ID và Class

- ID: Tên id chỉ có 1 tên duy nhất trong trang web

- Class: Cỏ thể đặt trùng tên

## Mức độ ưu tiên

- Nếu 2 thẻ được gọi giống nhau thì thẻ được gọi sau ưu tiên hơn thẻ được gọi trước

- Mức độ ưu tiên với các name của thẻ html trong css:

+ inline (style trực tiếp vào thẻ đó): 1000 điểm

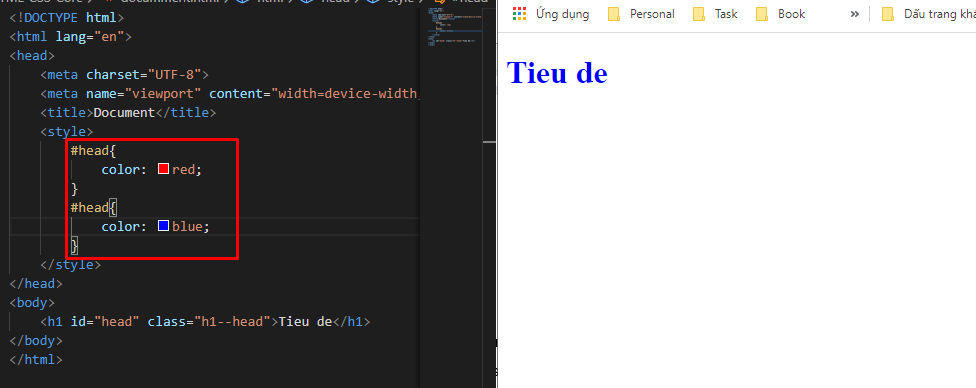
+ id: 100 điểm

+ class: 10 điểm

+ tagname: 1 điểm

- Với điểm càng cao thì mức độ ưu tiên càng nhiều

- Giả sử 2 thẻ có chung 1 id và được khai báo đồng thời thì thông thường thẻ nào được thông báo sau sẽ được style cho element đó.



Nhưng khi ta thêm class vào 1 element nào đó thì nó sẽ ưu tiên hơn element kia. Theo cách tính điểm mức độ ưu tiên ở trên thì element có lass sẽ có 110 điểm. element còn lại có 100 điểm. Và dĩ nhiên element có 110 điểm sẽ được ưu tiên cao hơn.



Ở đoạn code này máy tính sẽ hiểu là 1 element có chứa id là head và class là h1—head

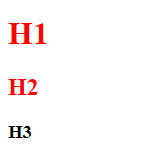
- \*{} - 0 điểm: Thường được đặt chung để reset css

- !important – 10000 điểm: Ưu tiên đặc biệt. Ghi đè được tất cả các trường hợp.

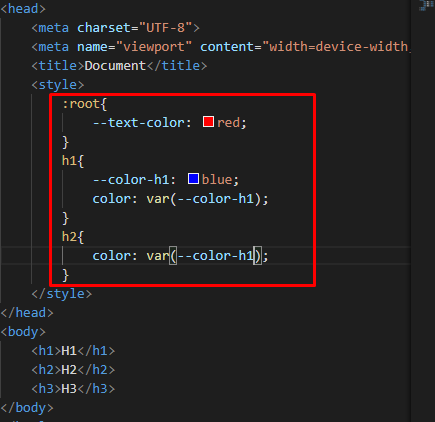
## 3.3. Đặt biến cho CSS

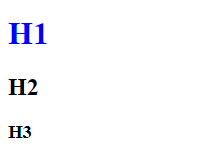
1. Biến toàn cục





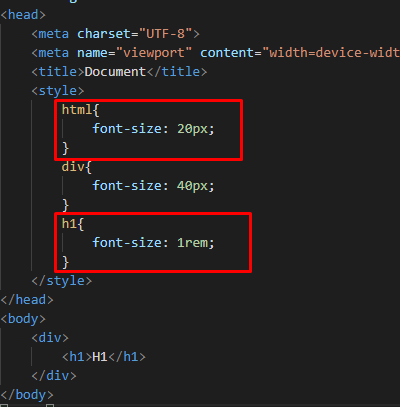
1. Biến cục bộ

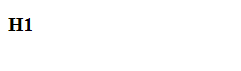




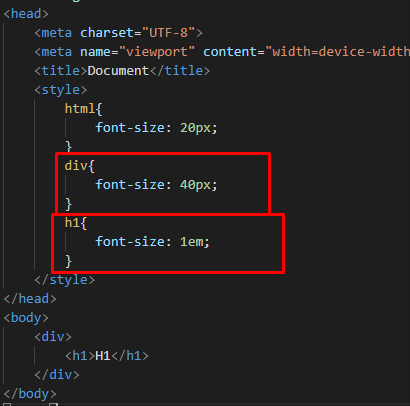
## 3.4. Đơn vị trong thẻ

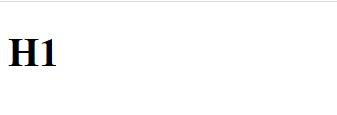
- rem : giá trị của rem phụ thuộc vào giá trị của thuộc tính tương ứng được khai báo trong html



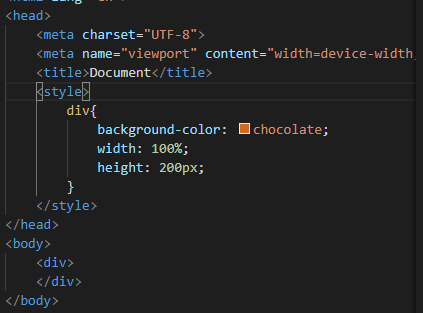


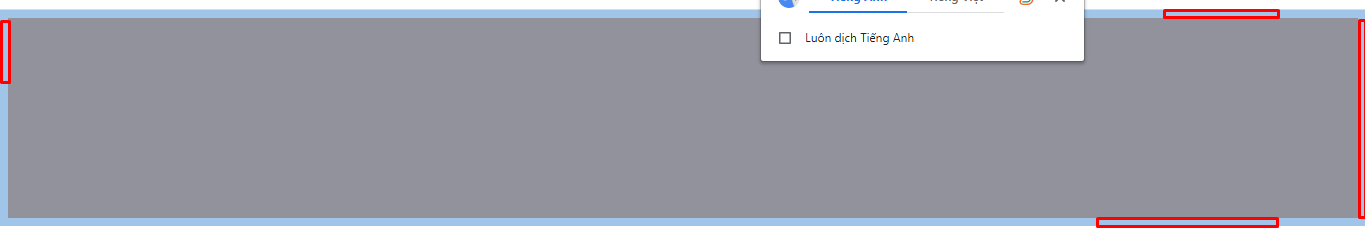
- em : giá trị của em phụ thuộc vào giá trị của thuộc tính tương ứng được khai báo trong thẻ gần nhất chứa nó





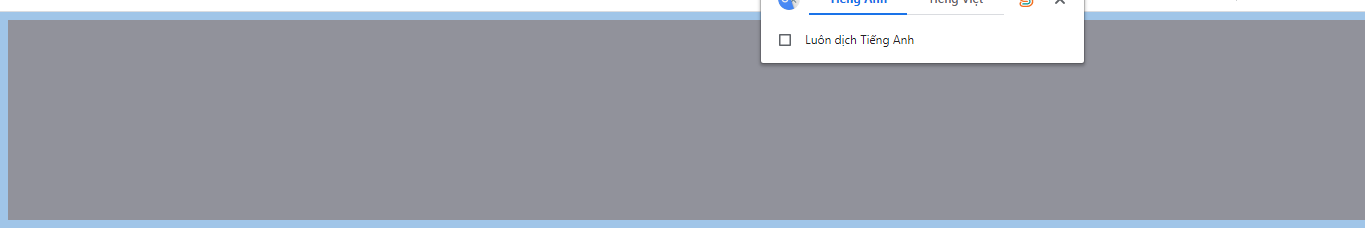
- vw : Kích thước chiều ngang bằng với số phần trăm kích thước khung nhìn





Với cách khai báo % như trên ta thấy nó sẽ bị phụ thuộc vào các thẻ phía trên. Nếu nó còn khoảng trắng thì thẻ div cũng sẽ có khoảng trắng theo



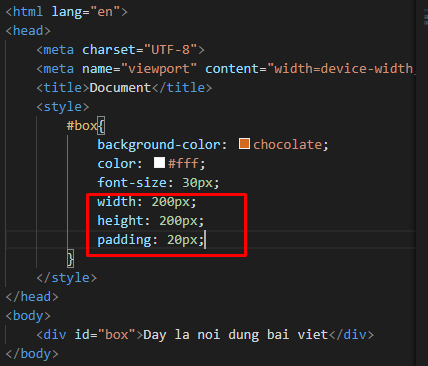


Với cách khai báo vw = 100 thì ta thấy phần bên phải không còn khoảng trắng nữa.

- vh : Kích thước chiều dọc bằng với số phần trăm khích thước khung nhìn

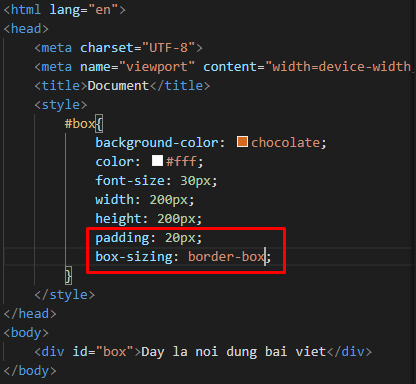
## 3.5. Thuộc tính CSS cần biết

## 3.5.1. Box-sizing





Khi ta dùng padding là 20px thì width và height sẽ bị tăng lên 240px. Để làm cho chữ cách đều khối mà khối vẫn giữ được kích thước ta sẽ sử dụng thuộc tinh box-sizing:





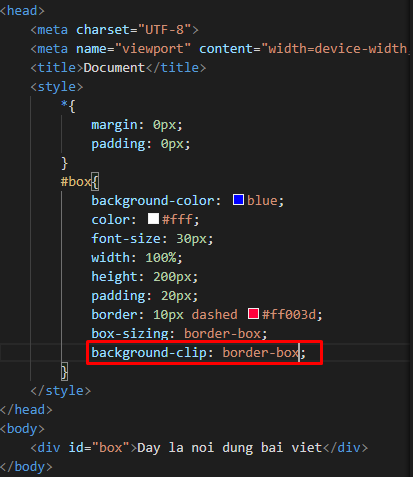
Nhưng khi ta khai báo padding quá lớn thì kích thước cũng sẽ bị tăng lên.

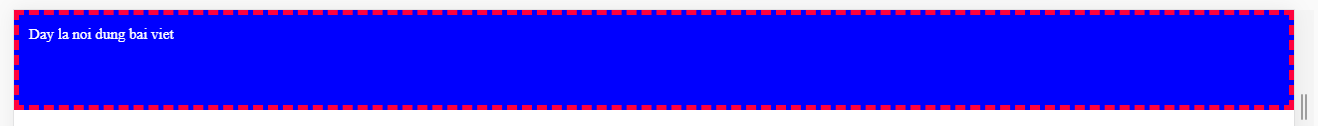
## 3.5.2 Background

#### 3.5.2.1 Background-clip

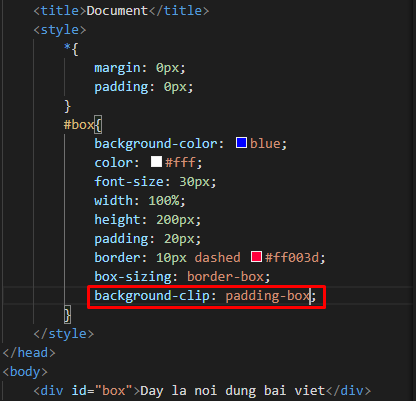
- Quyết định background sẽ đổ từ phần nào

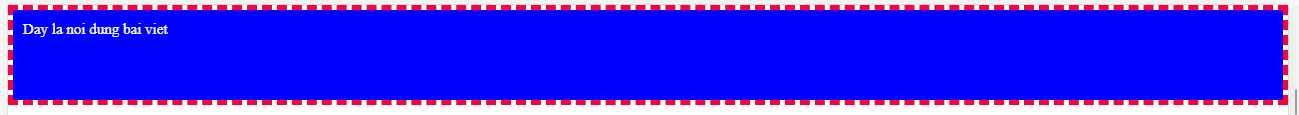
- border-box: background sẽ đổ từ phần border



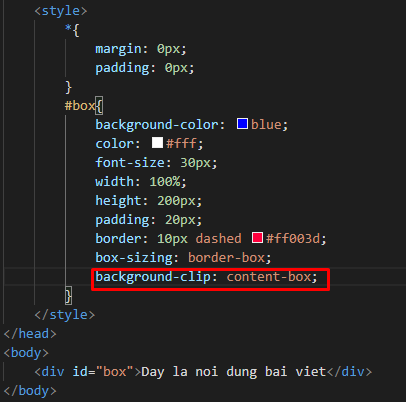


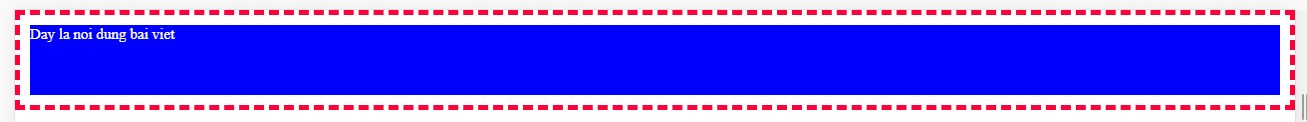
- padding-box: background sẽ đổ từ phần padding





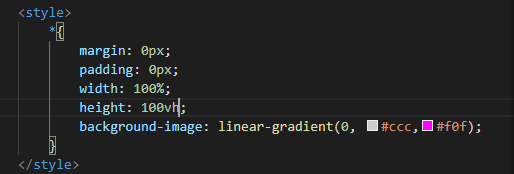
- content-box: background sẽ đổ từ phần content





#### 3.5.2.2 Background-image

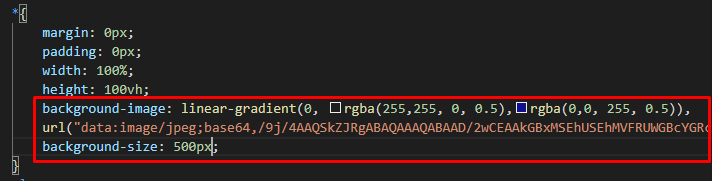
- linear-gradient(): Hòa trộn 2 màu với nhau





Với 0 có thể là 180 độ hoặc bất cứ giá trị nào.

- Kết hợp hình ảnh với linear-gradient():



Mỗi ảnh sẽ cách nhau 1 dấu phẩy



#### 3.5.2.3. Background-size

- contain : Lấy toàn bộ bức ảnh và khi chúng ta co trình duyệt lại thì nó sẽ co theo mà vẫn giữ được kích thước của bức ảnh. Vấn đề ở đây là contain sẽ tạo ra các khoảng trắng nằm ngang nếu kích thước trình duyệt vượt quá kích thước ảnh

- cover : Lấy chiều ngang của bức ảnh làm chiều mặc định và co giãn bức ảnh theo chiều ngang. Làm cho chiều dọc của bức ảnh có thể bị cắt.

#### 3.5.2.4. Background-origin

- Giống như thuộc tính background-clip nhưng áp dụng cho ảnh

#### 3.5.2.5. Background-position

- Set vị trí của ảnh

## 3.5.3. Position

- Thiết lập vị trí cho element

#### 3.5.3.1. Relative

- Thiết lập vị trí tương đối của element. Nó không phụ thuộc vào element nào cả mà nó lấy chính vị trí của nó đang đứng làm gốc tọa độ. Vì thế nó phụ thuộc vào chính nó.

- Chúng ta có thể thay đổi vị trí của những element có thuộc tính position: relative bằng các thuộc tính left, right, top, buttom

#### 3.5.3.2. Absolute

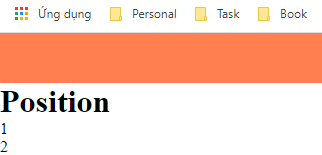
- Thiết lập vị trí tuyệt đối cho element. Nó phụ thuộc vào element cha gần nhất có thuộc tính relative.

- Chúng ta có thể thay đổi vị trí trong giới hạn element cha của nó bằng các thuộc tính top, left, buttom, right. Để cho element con nó thể nổi lên trên thẻ element cha thì ta sử dụng thuộc tính z-index

- Những element có thuộc tính absolute sẽ tìm những element cha có chứa thuộc tính position gần nhất để làm thẻ cha cho nó. Cho dù thẻ cha gần nhất của nó có là adsolute thì nó vẫn nhận thẻ đó là thẻ cha

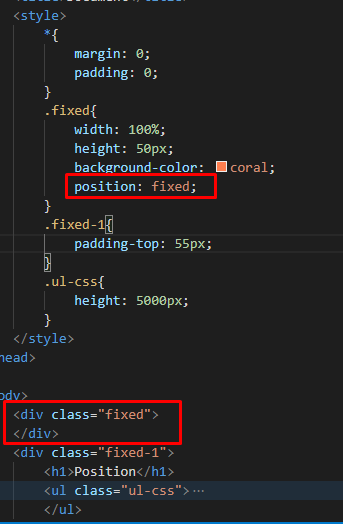
#### 3.5.3.3. Fixed

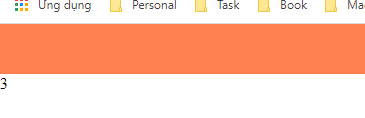
- Áp dụng khi muốn cho một đối tượng phụ thuộc vào cửa sổ trình duyệt





- Khi chưa áp dụng element navbar sẽ chạy theo trình duyệt. Và khi chúng ta cuộn trang thì thanh navbar sẽ cuộn theo





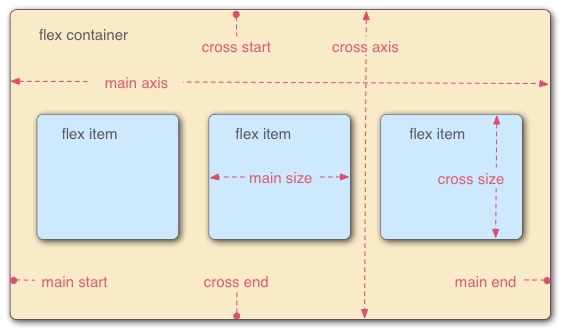
- Khi áp dụng fixed ta thấy thanh navbar luôn dính ở vị trí trên đầu của trình duyệt mặc dù ta có cuộn trang

#### 3.5.3.4. Sticky

- Giống như fixed thì sticky cũng bám dính thi trình duyệt kéo đến nhưng nó chỉ áp dụng cho một số trình duyệt

- Khi sử dụng fixed thì nó sẽ giữ khoảng cách khi trình duyệt cuộn còn với sticky nó sẽ đứng yên khi trình duyệt cuộn nhưng khi cuộn đến nơi nó sẽ dính luôn trên trình duyệt

## 3.5.4. Flexbox



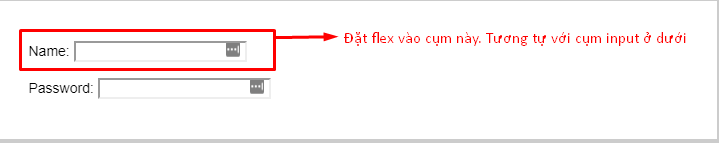
#### 3.5.4.1. Display: flex

- Khi đặt thuộc tính display: flex thì mặc định thuộc tính flex-direction: row sẽ được áp dụng cho element. Cho nên các element con sẽ cùng nằm trên một hàng

- Để style cho các element form được như hình bên dưới



Ta nên đặt thuộc tính flex vào từng thẻ input.



#### 3.5.4.2. Flex-direction

- Flex-direction sẽ nhận 2 kiểu là row và column

+ row: Các element con sẽ nằm ngang hàng nhau

+ column: Các element con sẽ nằm dọc

+ row-reverse: Đảo các element theo hàng ngang

+ column-reverse: Đảo các element theo hàng dọc

#### 3.5.4.3. Justify-content

- Justify-content sẽ nhận các kiểu là flex-start, flex-end, center, space-between và space-around

+ flex-start: Các element sẽ dồn về phía bên trái

+ flex-end: Các element sẽ dồn về phái bên phải

+ center: Các element sẽ ra giữa

+ space-between: Các element sẽ cách xa nhau và cách đều nhau

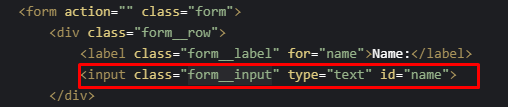
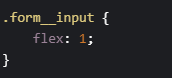
+ space-around: Các elenent sẽ cách xa nhau và sẽ cách hai cạnh của element cha một khoảng bằng một nửa khoảng cách giữa 2 element

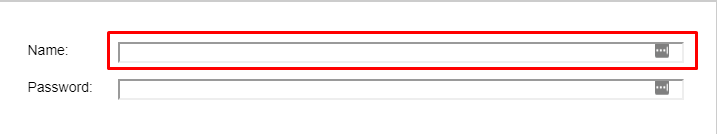
+ space-evenly: Các elenent sẽ cách xa nhau và sẽ cách hai cạnh của element cha một khoảng bằng khoảng cách giữa 2 element

#### 3.5.4.4. flex: 1

- Nếu có nhiều element con cùng nằm trên một hàng ngang ta có thể đặt thuộc tính flex với giá trị là các số thứ tự.

- Nếu chỉ có 1 element nằm trên hàng ngang thì nếu ta đặt flex: 1 nghĩa là element đó sẽ chiếm toàn bộ hàng ngang



#### 3.5.4.5. flex-basis

- Xét kích thước main size. Nếu có 3 khối ta đặt flex-basis: 30% thì cả 3 khối đề có độ rộng bằng nhau là 30% và tổng là 90%. Như vậy còn thừa ra 10%

#### 3.5.4.6. flex-wrap

- wrap: khi cho thuộc tính này vào main size thì các element con sẽ bị nhảy xuống dưới nếu chúng sát lại gần nhau.

- no-wrap: Sẽ không áp dụng flex-wrap

- wrap-reverse: Các element con sẽ nhảy lên trên nếu trình duyệt co lại

#### 3.5.4.7. Căn giữa cho text

- Ta đặt thuộc tính display: flex vào element cha. Và đặt trong element con là thuộc tính margin: auto

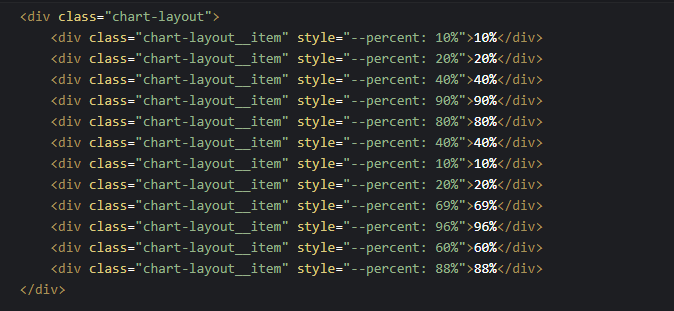
#### 3.5.4.8. align-self

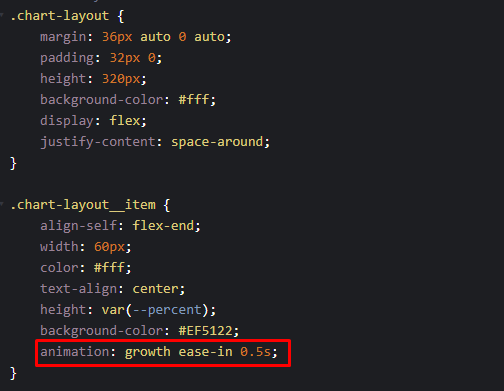
- Đảo chiều theo hướng dọc của element con. Với:

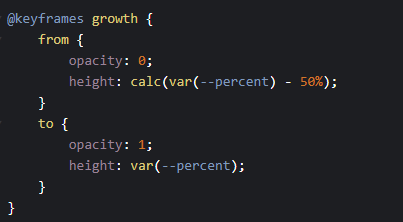
- flex-end: Từ dưới lên trên

- flex-start: Từ trên xuống dưới. Giá trị mặc định.

#### 3.5.4.9. Sử dụng animation







#### 3.5.4.10. order

- Thuộc tính order với giá trị bằng 1, 2, 3,… Với order bằng giá trị nhỏ hơn thì nó sẽ đứng ở giá trị ở trên hoặc bên trái.

## 3.6. Hàm thường dùng

- rgb(): Hàm set màu

- calc(): Hàm tính toán [ calc(100px+20px) ];

Attr(): Hàm gọi thuộc tính của các phần tử

## 3.7. Lớp giả

- :root : Tham chiếu đến phần tử gốc

- :hover : Thay đổi element khi di chuột vào đối tượng

- :active : Thay đổi element khi click chuột vào đối tượng

- :first-child : Lấy element con đầu tiên

- :last-child : Lấy element cuối cùng

- ::before : Tạo một lớp giả và luôn đứng trước các element cùng cấp

- ::after : Tạo một lớp giả và luôn đứng sau các element cùng cấp

- ::first-letter : Tạo lớp giả có thể style cho chữ cái đầu tiên

- ::first-line : Tạo lớp giả có thể style cho dòng đầu tiên

- ::selection : Tạo lớp giả để style cho phần tử đang bôi chuột lên text