[Tìm hiểu về HTML, CSS 9](#_Toc193579670)

[HTML và CSS là gì? 9](#_Toc193579671)

[Làm quen với Dev tools 9](#_Toc193579672)

[Cấu trúc của 1 file HTML 9](#_Toc193579673)

[Comments trong HTML 9](#_Toc193579674)

[Các thẻ HTML thông dụng 10](#_Toc193579675)

[Attribute trong HTML là gì? 10](#_Toc193579676)

[Sử dụng CSS trong HTML 10](#_Toc193579677)

[ID và Class (CSS selector) 10](#_Toc193579678)

[CSS selectors cơ bản 10](#_Toc193579679)

[1. Giới thiệu 10](#_Toc193579680)

[2. Sử dụng CSS selector ID 11](#_Toc193579681)

[3. Sử dụng CSS selector class 11](#_Toc193579682)

[Độ ưu tiên trong CSS (Priority) 14](#_Toc193579683)

[1. Internal. External ? 14](#_Toc193579684)

[2. Mức độ ưu tiên của Selectors (Specificity) 14](#_Toc193579685)

[3. Thứ tự trong Cascade (Luồng CSS) 15](#_Toc193579686)

[4. !important (Mạnh nhất) 15](#_Toc193579687)

[5. Tổng kết về độ ưu tiên 16](#_Toc193579688)

[6. Universal Selector (\*) 16](#_Toc193579689)

[7. Inheritance (Kế thừa thuộc tính trong CSS) 17](#_Toc193579690)

[a. Thuộc tính nào được kế thừa? 17](#_Toc193579691)

[b. Điều khiển Kế thừa với inherit, initial, unset 17](#_Toc193579692)

[Sử dụng biến trong CSS 18](#_Toc193579693)

[1. Cách Khai Báo Biến CSS 18](#_Toc193579694)

[2. Cách Sử Dụng Biến CSS 18](#_Toc193579695)

[3. Biến Cục Bộ vs. Biến Toàn Cục 19](#_Toc193579696)

[4. Biến với Giá Trị Mặc Định 19](#_Toc193579697)

[5. Thay Đổi Biến bằng JavaScript 20](#_Toc193579698)

[6. Ứng Dụng Thực Tế 20](#_Toc193579699)

[Tổng Kết 20](#_Toc193579700)

[Các đơn vị trong CSS (CSS Units) 21](#_Toc193579701)

[1. Đơn Vị Tuyệt Đối (Absolute Units) 21](#_Toc193579702)

[2. Đơn Vị Tương Đối (Relative Units) 21](#_Toc193579703)

[a. Đơn vị liên quan đến font-size 22](#_Toc193579704)

[b. Đơn vị liên quan đến kích thước viewport 22](#_Toc193579705)

[c. Đơn vị liên quan đến flexbox và grid 23](#_Toc193579706)

[d. Khi nào nên dùng đơn vị nào? 23](#_Toc193579707)

[Các Hàm Thường Dùng Trong CSS 23](#_Toc193579708)

[1. Hàm Toán Học 24](#_Toc193579709)

[a. calc() - Tính toán động trong CSS 24](#_Toc193579710)

[b. min(), max() - Giới hạn kích thước linh hoạt 24](#_Toc193579711)

[2. Hàm Xử Lý Màu Sắc 24](#_Toc193579712)

[a. rgb(), rgba() - Định nghĩa màu bằng giá trị RGB 24](#_Toc193579713)

[b. hsl(), hsla() - Màu sắc theo Hue, Saturation, Lightness 25](#_Toc193579714)

[c. var() - Sử dụng biến CSS 25](#_Toc193579715)

[3. Hàm Xử Lý Background & Hình Ảnh 26](#_Toc193579716)

[a. linear-gradient() - Tạo hiệu ứng chuyển màu tuyến tính 26](#_Toc193579717)

[b. radial-gradient() - Tạo hiệu ứng chuyển màu hình tròn 26](#_Toc193579718)

[c. conic-gradient() - Hiệu ứng chuyển màu hình nón 26](#_Toc193579719)

[4. Hàm Xử Lý Nội Dung & Hiệu Ứng 27](#_Toc193579720)

[a. attr() - Lấy giá trị từ thuộc tính HTML 27](#_Toc193579721)

[b. url() - Chèn hình ảnh từ URL 27](#_Toc193579722)

[5. Hàm Xử Lý Ký Tự & Văn Bản 27](#_Toc193579723)

[c. content() - Chèn nội dung động trong ::before và ::after 27](#_Toc193579724)

[🎯 Tổng Kết 28](#_Toc193579725)

[Pseudo Classes trong CSS 28](#_Toc193579726)

[1️. Pseudo-classes cơ bản 28](#_Toc193579727)

[a. :hover - Khi người dùng di chuột vào phần tử 28](#_Toc193579728)

[b. :focus - Khi phần tử được focus (chọn) 29](#_Toc193579729)

[c. :active - Khi phần tử đang được nhấn 29](#_Toc193579730)

[d. :visited - Khi liên kết đã được nhấn 29](#_Toc193579731)

[2️. Pseudo-classes chọn phần tử dựa trên vị trí 30](#_Toc193579732)

[a. :first-child - Chọn phần tử con đầu tiên 30](#_Toc193579733)

[b. :last-child - Chọn phần tử con cuối cùng 30](#_Toc193579734)

[c. :nth-child(n) - Chọn phần tử con thứ n 30](#_Toc193579735)

[d. :nth-child(odd/even) - Chọn phần tử con chẵn/lẻ 30](#_Toc193579736)

[e. :nth-of-type(n) - Chọn phần tử thứ n theo kiểu loại 31](#_Toc193579737)

[f. :first-of-type & :last-of-type - Chọn phần tử đầu/cuối theo kiểu loại 31](#_Toc193579738)

[3️. Pseudo-classes trạng thái đặc biệt 31](#_Toc193579739)

[a. :checked - Chọn phần tử checkbox hoặc radio được chọn 31](#_Toc193579740)

[b. :disabled - Chọn phần tử bị vô hiệu hóa 32](#_Toc193579741)

[c. :not(selector) - Chọn tất cả ngoại trừ phần tử chỉ định 32](#_Toc193579742)

[🎯 Tổng kết 32](#_Toc193579743)

[Pseudo-elements trong CSS 33](#_Toc193579744)

[1. *::before* - Thêm nội dung trước phần tử 33](#_Toc193579745)

[2. :*:after* - Thêm nội dung sau phần tử 33](#_Toc193579746)

[3. *::first-letter* - Chọn chữ cái đầu tiên của đoạn văn 34](#_Toc193579747)

[4. *::first-line* - Chọn dòng đầu tiên của đoạn văn 34](#_Toc193579748)

[5. ::selection - Tạo kiểu cho nội dung khi bôi đen 34](#_Toc193579749)

[6. *::placeholder* - Tạo kiểu cho placeholder trong input 34](#_Toc193579750)

[📝 Tổng hợp các Pseudo-elements phổ biến 35](#_Toc193579751)

[Thuộc tính padding trong CSS (đệm) 35](#_Toc193579752)

[1. Cách sử dụng padding 35](#_Toc193579753)

[a. giá trị – Áp dụng cho cả 4 cạnh 35](#_Toc193579754)

[b. giá trị – Áp dụng cho (Trên/Dưới) và (Trái/Phải) 36](#_Toc193579755)

[c. giá trị – (Trên) (Trái/Phải) (Dưới) 36](#_Toc193579756)

[d. giá trị – (Trên) (Phải) (Dưới) (Trái) 36](#_Toc193579757)

[2. padding theo từng cạnh riêng lẻ 36](#_Toc193579758)

[3. padding với đơn vị khác nhau 36](#_Toc193579759)

[4. box-sizing và padding 37](#_Toc193579760)

[🎯 Tổng kết 37](#_Toc193579761)

[Thuộc tính border trong CSS (đường viền) 37](#_Toc193579762)

[1. Cấu trúc của border 37](#_Toc193579763)

[2. Các kiểu đường viền (border-style) 38](#_Toc193579764)

[3. Đặt đường viền cho từng cạnh riêng lẻ 38](#_Toc193579765)

[4. Bo tròn góc với border-radius 39](#_Toc193579766)

[5. border và box-sizing 39](#_Toc193579767)

[🎯 Tổng kết 40](#_Toc193579768)

[Thuộc tính margin trong CSS (khoảng cách lề) 40](#_Toc193579769)

[1. Cấu trúc của margin 40](#_Toc193579770)

[2. Đặt margin riêng lẻ cho từng cạnh 40](#_Toc193579771)

[3. Cách viết rút gọn (margin shorthand) 41](#_Toc193579772)

[4. Sử dụng auto để căn giữa 41](#_Toc193579773)

[5. margin và box-sizing 42](#_Toc193579774)

[6. Hiện tượng margin collapse (sụp lề) 42](#_Toc193579775)

[🎯 Tổng kết 42](#_Toc193579776)

[Thuộc tính box-sizing trong CSS 43](#_Toc193579777)

[1. Cách hoạt động của box-sizing 43](#_Toc193579778)

[2. Các giá trị của box-sizing 43](#_Toc193579779)

[a. content-box (mặc định) 43](#_Toc193579780)

[b. border-box (hay dùng nhất) 43](#_Toc193579781)

[3. So sánh content-box vs border-box 44](#_Toc193579782)

[4. Áp dụng box-sizing cho toàn trang 44](#_Toc193579783)

[🎯 Tổng kết 45](#_Toc193579784)

[Thuộc tính background-image trong CSS 45](#_Toc193579785)

[1. Giới thiệu 45](#_Toc193579786)

[2. Cách sử dụng cơ bản 45](#_Toc193579787)

[3. Sử dụng nhiều hình ảnh nền 46](#_Toc193579788)

[4. Các thuộc tính liên quan 46](#_Toc193579789)

[a. background-repeat (Lặp lại hình nền) 46](#_Toc193579790)

[b. background-position (Vị trí hình nền) 46](#_Toc193579791)

[c. background-size (Kích thước hình nền) 47](#_Toc193579792)

[d. background-attachment (Cố định hình nền khi cuộn) 47](#_Toc193579793)

[e. background (Viết gộp) 48](#_Toc193579794)

[5. Tổng kết 48](#_Toc193579795)

[Thuộc tính background-origin trong CSS 48](#_Toc193579796)

[1. Giới thiệu 48](#_Toc193579797)

[2. Các giá trị của background-origin 49](#_Toc193579798)

[3. Ví dụ minh họa 49](#_Toc193579799)

[a. Ví dụ 1: So sánh 3 giá trị của background-origin 49](#_Toc193579800)

[b. Ví dụ 2: Kết hợp với background-clip 50](#_Toc193579801)

[4. Khi nào nên sử dụng background-origin? 50](#_Toc193579802)

[5. Tổng kết 50](#_Toc193579803)

[Thuộc tính background-position trong CSS 51](#_Toc193579804)

[1. Giới thiệu 51](#_Toc193579805)

[2. Cú pháp của background-position 51](#_Toc193579806)

[3. Các giá trị của background-position 51](#_Toc193579807)

[4. Ví dụ minh họa 52](#_Toc193579808)

[a. Ví dụ 1: Căn hình nền vào các góc 52](#_Toc193579809)

[b. Ví dụ 2: Dùng phần trăm (%) để định vị 52](#_Toc193579810)

[c. Ví dụ 3: Dùng đơn vị pixel (px) 53](#_Toc193579811)

[5. Khi nào nên dùng background-position? 53](#_Toc193579812)

[6. Tổng kết 53](#_Toc193579813)

[Cú pháp "shorthand" cho background trong CSS 54](#_Toc193579814)

[1. Giới thiệu 54](#_Toc193579815)

[2. Cú pháp đầy đủ của background 54](#_Toc193579816)

[3. Ví dụ minh họa 54](#_Toc193579817)

[a. Ví dụ 1: Viết đầy đủ từng thuộc tính 54](#_Toc193579818)

[b.Ví dụ 2: Viết gọn bằng background shorthand 54](#_Toc193579819)

[c. Ví dụ 3: Chỉ dùng màu nền 55](#_Toc193579820)

[d. Ví dụ 4: Chỉ dùng hình nền với vị trí và lặp lại 55](#_Toc193579821)

[e. Ví dụ 5: Định vị và kích thước hình nền 55](#_Toc193579822)

[5. Tổng kết 55](#_Toc193579823)

[Các giá trị màu sắc trong CSS 56](#_Toc193579824)

[1️. Tên màu (Color Names) 56](#_Toc193579825)

[2️. Mã màu HEX (Hexadecimal Colors) 56](#_Toc193579826)

[3️. Màu RGB (Red, Green, Blue) 57](#_Toc193579827)

[4️. Màu HSL (Hue, Saturation, Lightness) 57](#_Toc193579828)

[5️. Màu hệ thống (System Colors) 58](#_Toc193579829)

[6️. CurrentColor (Màu kế thừa) 58](#_Toc193579830)

[**📌 Tổng kết: Các cách biểu diễn màu sắc** 58](#_Toc193579831)

[Thuộc tính position: relative; trong CSS 59](#_Toc193579832)

[1️. position: relative; là gì? 59](#_Toc193579833)

[2️. Cách hoạt động của position: relative; 59](#_Toc193579834)

[3️. Cú pháp & Ví dụ 59](#_Toc193579835)

[4️. So sánh relative với các position khác 60](#_Toc193579836)

[5️. Ứng dụng relative thực tế 60](#_Toc193579837)

[a. Căn chỉnh một phần tử so với vị trí ban đầu của nó: 60](#_Toc193579838)

[b. Làm cha chứa cho absolute (Tạo tooltip, dropdown, popup, v.v.): 61](#_Toc193579839)

[6️. Kết luận 61](#_Toc193579840)

[CSS position: absolute 61](#_Toc193579841)

[1️. position: absolute; là gì? 61](#_Toc193579842)

[2️. Cách hoạt động 62](#_Toc193579843)

[3️. Ví dụ 62](#_Toc193579844)

[a. Ví dụ 1: absolute không có cha relative (định vị theo body) 62](#_Toc193579845)

[b. Ví dụ 2: absolute bên trong relative 62](#_Toc193579846)

[4️. So sánh với các position khác 63](#_Toc193579847)

[5️. Ứng dụng absolute thực tế 63](#_Toc193579848)

[6️. Kết luận 64](#_Toc193579849)

[CSS position: fixed; 64](#_Toc193579850)

[1️. position: fixed; là gì? 64](#_Toc193579851)

[2️. Cách hoạt động 65](#_Toc193579852)

[3️. Ví dụ 65](#_Toc193579853)

[a. Ví dụ 1: Thanh điều hướng cố định trên cùng 65](#_Toc193579854)

[b. Ví dụ 2: Nút "Lên đầu trang" cố định góc phải dưới 65](#_Toc193579855)

[4️. So sánh với các position khác 66](#_Toc193579856)

[5️. Ứng dụng fixed thực tế 66](#_Toc193579857)

[CSS position: sticky; 66](#_Toc193579858)

[1️. position: sticky; là gì? 66](#_Toc193579859)

[2️. Cách hoạt động 67](#_Toc193579860)

[3️. Ví dụ 67](#_Toc193579861)

[a. Ví dụ 1: Thanh tiêu đề "dính" khi cuộn xuống 67](#_Toc193579862)

[b. Ví dụ 2: Cột điều hướng bên trái "dính" khi cuộn 67](#_Toc193579863)

[4️. So sánh với relative, absolute, fixed 68](#_Toc193579864)

[5️. Ứng dụng sticky thực tế 68](#_Toc193579865)

[Giới thiệu về Flexbox trong CSS 68](#_Toc193579866)

[1. Cấu trúc cơ bản của Flexbox 68](#_Toc193579867)

[**Ví dụ cơ bản:** 69](#_Toc193579868)

[2. Các thuộc tính của Flexbox 70](#_Toc193579869)

[a. flex-direction (Xác định hướng của trục chính) 70](#_Toc193579870)

[b. justify-content (Căn chỉnh theo trục chính) 70](#_Toc193579871)

[c. align-items (Căn chỉnh theo trục phụ) 71](#_Toc193579872)

[3. Thuộc tính của Flex Items 71](#_Toc193579873)

[a. flex-grow (Mở rộng phần tử) 72](#_Toc193579874)

[b. flex-shrink (Thu nhỏ phần tử) 72](#_Toc193579875)

[c. flex-basis (Kích thước ban đầu) 72](#_Toc193579876)

[d. flex (Viết tắt của flex-grow, flex-shrink, flex-basis) 73](#_Toc193579877)

[e. align-self (Căn chỉnh riêng từng item) 73](#_Toc193579878)

[4. Ví dụ thực tế - Thanh điều hướng với Flexbox 73](#_Toc193579879)

[5. Bổ sung flex-wrap trong CSS 74](#_Toc193579880)

[🚀 Tóm tắt nhanh về Flexbox 75](#_Toc193579881)

[6. Học Flexbox qua ví dụ 75](#_Toc193579882)

[a. Code mẫu sử dụng trong video 75](#_Toc193579883)

[b. Chơi game 75](#_Toc193579884)

# Tìm hiểu về HTML, CSS

## HTML và CSS là gì?

* **HTML** là chữ viết tắt của **Hypertext Markup Language**. Nó giúp người dùng tạo và cấu trúc các thành phần trong trang web hoặc ứng ụng
* **CSS** là ngôn ngữ tạo phong cách cho trang web – **Cascading Style Sheet** language. Nó dùng để tạo phong cách và định kiểu cho những yếu tố được viết dưới dạng ngôn ngữ đánh dấu, như là HTML.

# Làm quen với Dev tools

# Cấu trúc của 1 file HTML

<!DOCTYPE html>

<html lang="en">

    <!-- Phần đầu -->

    <head>

        <meta charset="UTF-8">

        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <title>This is TITLE</title>

    </head>

    <!-- Phần thân -->

    <body>

        Hello World!

    </body>

</html>

# Comments trong HTML

* Dùng làm chú thích, giải thích code, vô hiệu hóa dòng code.
* Ví dụ:

<!-- Phần đầu -->

<!-- Phần thân -->

# Các thẻ HTML thông dụng

        <!--

        1. h1 - h6 (heading - tiêu đề)

        2. p (paragraph - đoạn văn)

        3. img (image - hình ảnh)

        4. a (anchor - mỏ neo) => Dùng để gắn link

        5. ul/ol, li (unordered list, list item - dạng danh sách)

        6. table (bảng) - thead - tbody - th - tr - td

        7. input

        8. button (nut)

        9. div (tạo một khối bao quanh phần tử khác)

        -->

# Attribute trong HTML là gì?

<h1 title="Đây là tiêu đề">Attribute?</h1>

* Title là một attribute (thuộc tính) của thẻ html nằm trong thẻ mở.

# Sử dụng CSS trong HTML

1. Internal (bên trong)
2. External (bên ngoài)
3. Inline (tại dòng, sử dụng attribute style)

# ID và Class (CSS selector)

* Nếu là class (.class)
* Nếu là ID (#id)

# CSS selectors cơ bản

## 1. Giới thiệu

Trong CSS, CSS selectors là các cách chúng ta sử dụng để chọn ra các phần tử (elements) mà chúng ta muốn "style" cho chúng. Có 2 CSS selectors thân thuộc với chúng ta nhất là id và class.

## 2. Sử dụng CSS selector ID

<h2 id="primary-heading">Tiêu đề 1</h2>

<h2 id="secondary-heading">Tiêu đề 2</h2>

...

<style>

    #primary-heading {

        color: red;

    }

</style>

Kết quả, thẻ có id="primary-heading" thành màu đỏ.

A white background with black text

AI-generated content may be incorrect.

**Trong cùng 1 page HTML, mỗi ID cần là duy nhất. Đây là nguyên tắc của việc sử dụng ID, nếu vi phạm sẽ gặp cảnh báo trong tab Console của dev tool, và Javascript sẽ không lấy ra được cả 2 phần tử có dùng ID (Javascript các bạn sẽ học ở**[**khóa sau**](https://fullstack.edu.vn/courses/javascript-co-ban)**).**

## 3. Sử dụng CSS selector class

**Ví dụ 1: Lựa chọn qua 1 class**

<p class="paragraph">Đoạn văn 1</p>

<p class="paragraph">Đoạn văn 2</p>

<p class="other-paragraph">Đoạn văn 3</p>

<style>

    .paragraph {

        color: green;

    }

</style>

Kết quả, tất cả các class paragraph đều có màu xanh lục.

A white rectangular object with a black border

AI-generated content may be incorrect.

**Ví dụ 2: Lựa chọn qua nhiều class**

Một thẻ có thể có nhiều class, mỗi class cách nhau bằng 1 khoảng trắng (dấu space).

<p class="p1-normal">Đoạn văn 1</p>

<p class="p1-normal">Đoạn văn 2</p>

<p class="p1-normal p2-red">Đoạn văn 3</p>

<p class="p1-normal p2-red">Đoạn văn 4</p>

<style>

    .p1-normal {

        color: green;

    }

    /\* Lưu ý, các tên class viết liền nhau, không có khoảng trắng \*/

    .p1-normal.p2-red {

        color: red;

    }

</style>

Kết quả, tất cả các thẻ đồng thời có cả class p1-normal và p2-red đều có màu đỏ.

A white rectangular object with a black border

AI-generated content may be incorrect.

**Tương tự như vậy, nếu thẻ có nhiều class hơn bạn chỉ cần nối thêm tên class vào khi CSS, ví dụ: .class1.class2.class3. Trong trường hợp này, thuộc tính CSS nằm trong selector được nối bởi nhiều class hơn (chỉ ra chi tiết hơn) sẽ được ưu tiên hơn.**

**Ví dụ 3: CSS cho các thẻ con**

<p class="children-1">Đoạn văn 1 (không parent)</p>

<p class="children-2">Đoạn văn 2 (không parent)</p>

<div class="parent">

    <p>Đoạn văn 1 (parent 1)</p>

    <p class="children-1">Đoạn văn 1 (con của .parent)</p>

    <p class="children-2">Đoạn văn 2 (con của .parent)</p>

</div>

<style>

    /\* Chọn tất cả thẻ "p" là con của thẻ có class "parent" \*/

    .parent p {

        color: green;

    }

    /\* Chọn tất cả thẻ có class "children-1" là con của thẻ có class "parent" \*/

    .parent .children-1 {

        color: red;

    }

</style>

A white rectangular object with black lines

AI-generated content may be incorrect.

**Chúng ta có thể tùy ý kết hợp các CSS selectors đã học để lựa chọn ra đúng thẻ chúng ta muốn CSS. Ở ví dụ trên, để lựa chọn thẻ con nằm trong 1 thẻ khác ta sẽ sử dụng selector có cú pháp selector-cha selector-con (lưu ý, giữa 2 selectors có một khoảng trắng - dấu cách).**

**Bảng tra cứu CSS selectors cơ bản**

Trong bài này mình chỉ đưa ra những CSS selectors cơ bản và thường xuyên được sử dụng nhất.

| **Selector** | **Ví dụ** | **Mô tả** |
| --- | --- | --- |
| .class | .intro | Chọn tất cả các thẻ có class="intro" |
| .class1.class2 | .name1.name2 | Chọn tất cả các thẻ có cả name1 và name2 được đặt trong thuộc tính class của nó |
| .class1 .class2 | .name1 .name2 | Chọn tất cả các thẻ có class="name2" là con của một phần tử có class="name1" |
| \* | \* | Chọn tất cả các thẻ |
| element | h2 | Chọn tất cả các thẻ h2 |
| element.class | div.box | Chọn tất cả thẻ div có class="box" |
| element, element | div, h2 | Chọn tất cả thẻ div và h2 |
| element element | div p | Chọn tất cả thẻ p trong thẻ div |
| element > element | div > p | Chọn tất cả thẻ p là con trực tiếp của thẻ div |
| element + element | div + p | Chọn thẻ p đứng liền kề sau thẻ div |
| element ~ element | div ~ p | Chọn tất cả thẻ p đứng sau thẻ div |

Tham khảo thêm tại: <https://www.w3schools.com/cssref/css_selectors.asp>

**CSS selectors là công cụ vô cùng mạnh mẽ. Toàn bộ các selectors trong CSS đều có thể linh động kết hợp với nhau khi sử dụng giúp lập trình viên có thể chọn được bất cứ phần tử HTML nào trong website.**

# Độ ưu tiên trong CSS (Priority)

## 1. Internal. External ?

* Ông nào gọi trước ông đó được ưu tiên

## 2. Mức độ ưu tiên của Selectors (Specificity)

Mỗi selector có một mức độ ưu tiên (specificity) theo công thức:

**Inline styles > ID selectors > Class, Attribute & Pseudo-class selectors > Element & Pseudo-element selectors**

* **Inline styles** (style="") → **priority cao nhất** (1000 điểm)
* **ID selectors** (#id) → 100 điểm
* **Class, Attribute, Pseudo-class selectors** (.class, [attr], :hover) → 10 điểm
* **Element & Pseudo-element selectors** (div, h1, ::before) → 1 điểm

Ví dụ:

/\* 1 điểm \*/

div { color: red; }

/\* 10 điểm \*/

.myClass { color: blue; }

/\* 100 điểm \*/

#myId { color: green; }

/\* 1000 điểm (cao nhất) \*/

<button style="color: orange;">Click Me</button>

## 3. Thứ tự trong Cascade (Luồng CSS)

Nếu hai quy tắc có cùng specificity, quy tắc nào xuất hiện **sau** sẽ được áp dụng.

Ví dụ:

p { color: blue; }

p { color: red; } /\* Quy tắc này sẽ thắng vì xuất hiện sau \*/

**✅** Kết quả: <p> sẽ có màu đỏ.

## 4. !important (Mạnh nhất)

*!important* sẽ **đánh bại tất cả các quy tắc khác**, kể cả specificity hay cascade.

Ví dụ:

p { color: red !important; }

p { color: blue; }

✅ Kết quả: <p> sẽ có màu đỏ vì !important thắng.

📌 **Lưu ý**: Tránh lạm dụng !important vì nó làm mất tính linh hoạt của CSS.

## 5. Tổng kết về độ ưu tiên

| **Loại** | **Độ ưu tiên** |
| --- | --- |
| !important | Cao nhất 🔥 |
| Inline styles (style="") | 1000 |
| ID selectors (#id) | 100 |
| Class, Attribute, Pseudo-class (.class, [attr], :hover) | 10 |
| Element, Pseudo-element (div, h1, ::before) | 1 |

📌 **Tóm lại**:

* Dùng ID selectors cẩn thận, nên ưu tiên **class** thay vì ID.
* Không lạm dụng !important, chỉ dùng khi thực sự cần thiết.
* Nếu có xung đột, kiểm tra **specificity**, **thứ tự CSS** và **!important**.

## 6. Universal Selector (\*)

* Universal selector (\*) chọn tất cả các phần tử trong tài liệu HTML (trừ một số trường hợp đặc biệt như ::before và ::after nếu không được đặt content).
* Chủ yếu được dùng để thiết lập reset hoặc áp dụng một quy tắc chung.

Ví dụ:

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

  }

📌 **Tác dụng**: Loại bỏ margin, padding mặc định của trình duyệt cho tất cả các phần tử.

Ứng dụng nâng cao:

\* {

    font-family: 'Arial', sans-serif;

  }

**📌 Tất cả các phần tử trên trang sẽ dùng font Arial**, trừ những phần tử bị ghi đè bởi quy tắc khác có độ ưu tiên cao hơn

## 7. Inheritance (Kế thừa thuộc tính trong CSS)

### a. Thuộc tính nào được kế thừa?

Không phải tất cả các thuộc tính đều được kế thừa. CSS chia thành **2 loại chính**:

✅ **Các thuộc tính liên quan đến văn bản** (có kế thừa mặc định):

* color
* font
* font-family
* visibility
* letter-spacing
* word-spacing
* cursor
* direction  
  📌 **Các thuộc tính này tự động kế thừa từ phần tử cha**.

❌ **Các thuộc tính không được kế thừa mặc định**:

* margin, padding, border, width, height
* background, box-shadow
* position, display, flex, grid  
  📌 **Lý do**: Nếu chúng được kế thừa mặc định, bố cục trang sẽ bị phá vỡ.

### b. Điều khiển Kế thừa với inherit, initial, unset

 **inherit** → Kế thừa từ phần tử cha (dù mặc định có kế thừa hay không).

 **initial** → Đặt về giá trị mặc định của trình duyệt.

 **unset** → Nếu thuộc tính có kế thừa, nó sẽ kế thừa; nếu không, nó đặt về giá trị mặc định.

Ví dụ:

p {

    color: red; /\* Áp dụng cho tất cả <p> \*/

  }

  span {

    color: inherit; /\* Lấy màu của phần tử cha \*/

  }

  div {

    background: inherit; /\* Sẽ không kế thừa vì background không có kế thừa mặc định \*/

  }

# Sử dụng biến trong CSS

* Biến trong CSS (CSS Variables) giúp tái sử dụng giá trị trong stylesheet một cách linh hoạt, giúp **dễ dàng thay đổi giao diện** mà không cần chỉnh sửa nhiều lần.

## 1. Cách Khai Báo Biến CSS

* Biến CSS được khai báo bằng cú pháp ***--tên-biến*** và sử dụng bằng ***var(--tên-biến)***.

Ví dụ đơn giản:

:root {

    --primary-color: #3498db;

    --secondary-color: #2ecc71;

    --font-size: 16px;

  }

📌 **:root** đại diện cho **đối tượng gốc** của tài liệu (thường là <html>), giúp biến có phạm vi toàn cục.

## 2. Cách Sử Dụng Biến CSS

Sau khi khai báo, có thể sử dụng biến trong bất kỳ thuộc tính CSS nào:

button {

    background-color: var(--primary-color);

    color: white;

    font-size: var(--font-size);

    padding: 10px 20px;

    border-radius: 5px;

    border: none;

  }

  button:hover {

    background-color: var(--secondary-color);

  }

📌 Khi thay đổi giá trị biến tại **:root,** toàn bộ trang sẽ cập nhật ngay lập tức!

## 3. Biến Cục Bộ vs. Biến Toàn Cục

* **Biến Toàn Cục**: Được khai báo trong :root, có thể dùng ở mọi nơi.
* **Biến Cục Bộ**: Chỉ có hiệu lực trong một selector cụ thể.

Ví dụ về biến cục bộ:

.card {

    --card-bg: #f4f4f4;

    background-color: var(--card-bg);

    padding: 20px;

    border-radius: 10px;

}

.card.dark {

    --card-bg: #333;

    color: white;

}

📌 Ở **.card.dark,** biến **--card-bg** sẽ ghi đè biến mặc định của **.card!**

## 4. Biến với Giá Trị Mặc Định

Nếu biến không tồn tại, có thể đặt giá trị mặc định bằng cách truyền tham số thứ hai cho **var().**

h1 {

    color: var(--heading-color, black); /\* Nếu --heading-color không tồn tại, dùng màu đen \*/

  }

## 5. Thay Đổi Biến bằng JavaScript

Bạn có thể thay đổi giá trị biến CSS **trực tiếp từ JavaScript** để tạo hiệu ứng động.

Ví dụ thay đổi màu nền khi nhấn nút:

    document.documentElement.style.setProperty('--primary-color', '#e74c3c');

📌 Điều này giúp **tạo chế độ Dark Mode** hoặc **thay đổi giao diện** theo người dùng.

## 6. Ứng Dụng Thực Tế

* Dark Mode với CSS Variables:

:root {

    --bg-color: white;

    --text-color: black;

  }

.dark-mode {

    --bg-color: black;

    --text-color: white;

}

body {

    background-color: var(--bg-color);

    color: var(--text-color);

}

* Chuyển chế độ tối bằng JavaScript:

    document.body.classList.toggle('dark-mode');

📌 **Khi nhấn nút**, tất cả màu nền và chữ sẽ thay đổi ngay lập tức!

## Tổng Kết

✅ **Dễ bảo trì** – Thay đổi một nơi, áp dụng toàn bộ.  
✅ **Tối ưu hiệu suất** – Giảm lặp lại mã CSS.  
✅ **Linh hoạt** – Dễ dàng chỉnh sửa bằng JavaScript.  
✅ **Hỗ trợ tất cả trình duyệt hiện đại** (IE11 cần prefix var(--name)).

# Các đơn vị trong CSS (CSS Units)

CSS cung cấp nhiều loại đơn vị đo lường khác nhau, chia thành **2 nhóm chính**:

* **Đơn vị tuyệt đối (Absolute units)** – Có kích thước cố định.
* **Đơn vị tương đối (Relative units)** – Phụ thuộc vào phần tử khác hoặc viewport.

## 1. Đơn Vị Tuyệt Đối (Absolute Units)

Những đơn vị này có kích thước **cố định**, không bị thay đổi bởi các yếu tố bên ngoài.

| **Đơn vị** | **Ý nghĩa** | **Giá trị tham khảo** |
| --- | --- | --- |
| px | Pixels (điểm ảnh) | 1px = 1 điểm ảnh |
| cm | Centimet | 1cm = 37.8px |
| mm | Millimet | 1mm = 3.78px |
| in | Inches (inch) | 1in = 96px |
| pt | Points (điểm in ấn) | 1pt = 1/72 inch |
| pc | Picas (pica = 12pt) | 1pc = 16px |

Ví dụ sử dụng **px**:

.box {

    width: 200px;

    height: 100px;

    background-color: lightblue;

  }

📌 px là đơn vị phổ biến nhất vì **tính nhất quán** trên các thiết bị.

## 2. Đơn Vị Tương Đối (Relative Units)

Những đơn vị này thay đổi **dựa trên phần tử cha, viewport hoặc font-size**.

### a. Đơn vị liên quan đến font-size

| **Đơn vị** | **Ý nghĩa** | **Giá trị tham khảo** |
| --- | --- | --- |
| em | Tỉ lệ so với font-size của phần tử cha | 1em = 100% giá trị cha |
| rem | Tỉ lệ so với font-size của html | 1rem = 100% giá trị của html |
| % | Phần trăm của giá trị phần tử cha | 100% = giá trị của phần tử cha |

Ví dụ **em** và **rem:**

html {

    font-size: 16px;

}

p {

    font-size: 1.5em; /\* 1.5 \* 16px = 24px \*/

}

h1 {

    font-size: 2rem; /\* 2 \* 16px = 32px \*/

}

📌 **rem** thường được dùng thay vì **em** để giữ kích thước ổn định trên toàn trang.

### b. Đơn vị liên quan đến kích thước viewport

| **Đơn vị** | **Ý nghĩa** | **Giá trị tham khảo** |
| --- | --- | --- |
| vw | % chiều rộng của viewport | 1vw = 1% viewport width |
| vh | % chiều cao của viewport | 1vh = 1% viewport height |
| vmin | Giá trị nhỏ hơn giữa vw và vh | — |
| vmax | Giá trị lớn hơn giữa vw và vh | — |

Ví dụ **vw** và **vh**:

.fullscreen {

    width: 100vw;  /\* Chiếm toàn bộ chiều rộng màn hình \*/

    height: 100vh; /\* Chiếm toàn bộ chiều cao màn hình \*/

    background-color: black;

    color: white;

  }

📌 Rất hữu ích khi làm **trang web full-screen** hoặc **định kích thước** linh hoạt.

### c. Đơn vị liên quan đến flexbox và grid

| **Đơn vị** | **Ý nghĩa** | **Ứng dụng** |
| --- | --- | --- |
| fr | Fraction (phần tỷ lệ trong CSS Grid) | Chia cột, hàng Grid |
| auto | Kích thước tự động dựa trên nội dung | Layout linh hoạt |
| min-content | Kích thước nhỏ nhất có thể để không bị vỡ nội dung | Chia cột tự động |
| max-content | Kích thước lớn nhất mà nội dung có thể mở rộng | Giữ nội dung nguyên vẹn |

Ví dụ **fr** trong CSS Grid:

.container {

    display: grid;

    grid-template-columns: 1fr 2fr; /\* Cột đầu 1 phần, cột sau 2 phần \*/

  }

📌 **fr** giúp chia **tỷ lệ linh hoạt**, không cần xác định **px**.

### d. Khi nào nên dùng đơn vị nào?

✅ **px**: Khi cần kích thước **cố định** (icon, button).  
✅ **em/rem**: Khi cần kích thước **linh hoạt theo font** (text, spacing).  
✅ **%**: Khi cần **tỷ lệ phần trăm so với cha** (chiều rộng, lề).  
✅ **vw/vh**: Khi cần **chiếm toàn bộ màn hình** (hero section, fullscreen modal).  
✅ **fr**: Khi dùng **CSS Grid** để chia tỷ lệ.

# Các Hàm Thường Dùng Trong CSS

CSS cung cấp nhiều **hàm (functions)** giúp bạn **tính toán, thay đổi màu sắc, xử lý hình ảnh** và nhiều hiệu ứng khác. Dưới đây là những hàm phổ biến nhất trong CSS.

## 1. Hàm Toán Học

### a. calc() - Tính toán động trong CSS

Dùng để thực hiện **phép toán số học** (+, -, \*, /) giữa các đơn vị khác nhau.

Ví dụ:

.container {

    width: calc(100% - 50px); /\* Chiều rộng là 100% trừ đi 50px \*/

    height: calc(50vh + 20px); /\* Chiều cao là 50% viewport cộng 20px \*/

  }

📌 **Ứng dụng**: Khi bạn cần **tính toán kích thước** linh hoạt thay vì giá trị cố định.

### b. min(), max() - Giới hạn kích thước linh hoạt

* **min(a, b):** Lấy giá trị **nhỏ nhất** giữa hai giá trị.
* **max(a, b):** Lấy giá trị **lớn nhất** giữa hai giá trị.
* **clamp(min, preferred, max):** Giới hạn trong khoảng **từ min đến max**.

Ví dụ:

.box {

    width: min(50vw, 300px); /\* Chọn giá trị nhỏ nhất giữa 50% viewport và 300px \*/

    font-size: max(16px, 2vw); /\* Chọn giá trị lớn nhất giữa 16px và 2% viewport \*/

    height: clamp(200px, 50vh, 500px); /\* Giới hạn từ 200px đến 500px \*/

  }

📌 **Ứng dụng**: Đảm bảo kích thước luôn **tự động điều chỉnh** mà không vượt quá mức giới hạn.

## 2. Hàm Xử Lý Màu Sắc

### a. rgb(), rgba() - Định nghĩa màu bằng giá trị RGB

Ví dụ:

.text {

    color: rgb(255, 0, 0);   /\* Màu đỏ \*/

    color: rgba(0, 0, 255, 0.5); /\* Màu xanh với độ trong suốt 50% \*/

  }

### b. hsl(), hsla() - Màu sắc theo Hue, Saturation, Lightness

Ví dụ:

.text {

    color: hsl(240, 100%, 50%); /\* Màu xanh dương \*/

    color: hsla(120, 100%, 50%, 0.7); /\* Màu xanh lá với độ trong suốt 70% \*/

  }

📌 **Ứng dụng**: **hsl()** giúp dễ **tinh chỉnh độ sáng/tối của màu sắc**.

### c. var() - Sử dụng biến CSS

Dùng để **tái sử dụng giá trị** đã khai báo trong **:root**.

Ví dụ:

:root {

    --main-color: #3498db;

    --text-size: 18px;

}

.button {

background: var(--main-color);

font-size: var(--text-size);

}

📌 **Ứng dụng**: Dễ dàng **thay đổi toàn bộ màu sắc** của trang chỉ bằng cách chỉnh sửa biến.

## 3. Hàm Xử Lý Background & Hình Ảnh

### a. linear-gradient() - Tạo hiệu ứng chuyển màu tuyến tính

Ví dụ:

.gradient {

    background: linear-gradient(to right, red, yellow);

  }

Kết quả:

A red and yellow gradient

AI-generated content may be incorrect.

📌 **Ứng dụng**: Làm **nút bấm, hiệu ứng nền** đẹp mắt.

### b. radial-gradient() - Tạo hiệu ứng chuyển màu hình tròn

Ví dụ:

.gradient {

    background: radial-gradient(circle, red, yellow);

}

Kết quả:

A red and yellow circle

AI-generated content may be incorrect.

📌 **Ứng dụng**: Hiệu ứng nền **dạng ánh sáng** hoặc **hiệu ứng 3D**.

### c. conic-gradient() - Hiệu ứng chuyển màu hình nón

Ví dụ:

.gradient {

    background: conic-gradient(from 0deg, red, yellow, green);

  }

Kết quả:

A red green and yellow gradient

AI-generated content may be incorrect.

📌 **Ứng dụng**: Tạo **biểu đồ tròn**, hiệu ứng **xoay vòng**.

## 4. Hàm Xử Lý Nội Dung & Hiệu Ứng

### a. attr() - Lấy giá trị từ thuộc tính HTML

Ví dụ:

.tooltip::after {

    content: attr(data-tooltip); /\* Lấy nội dung từ thuộc tính data-tooltip \*/

}

📌 **Ứng dụng**: **Tooltip động**, hiển thị nội dung từ HTML.

### b. url() - Chèn hình ảnh từ URL

Ví dụ:

.logo {

background-image: url('logo.png');

}

📌 **Ứng dụng**: Chèn **background động, hình ảnh từ API**.

## 5. Hàm Xử Lý Ký Tự & Văn Bản

### c. content() - Chèn nội dung động trong ::before và ::after

Ví dụ:

button::before {

    content: "🔥 ";

  }

📌 **Ứng dụng**: Thêm **biểu tượng** hoặc **văn bản bổ sung**.

## 🎯 Tổng Kết

| **Hàm** | **Công dụng** |
| --- | --- |
| calc() | Tính toán kích thước linh hoạt |
| min(), max(), clamp() | Giới hạn kích thước động |
| rgb(), hsl() | Chỉnh màu sắc |
| var() | Sử dụng biến CSS |
| linear-gradient(), radial-gradient() | Hiệu ứng chuyển màu |
| url() | Chèn ảnh từ URL |
| attr() | Lấy giá trị từ HTML |
| content() | Chèn nội dung động |

# Pseudo Classes trong CSS

**Pseudo-classes** là các **từ khóa đặc biệt** trong CSS giúp bạn **chọn các phần tử dựa trên trạng thái, vị trí hoặc hành động** mà không cần phải thêm class hoặc ID vào HTML.

## 1️. Pseudo-classes cơ bản

### a. :hover - Khi người dùng di chuột vào phần tử

Ví dụ:

button:hover {

    background-color: #ff5733;

    color: white;

}

📌 **Ứng dụng:** Tạo **hiệu ứng hover** cho nút bấm, liên kết, hình ảnh.

### b. :focus - Khi phần tử được focus (chọn)

Ví dụ:

input:focus {

    border: 2px solid blue;

    outline: none;

}

📌 **Ứng dụng:** Làm nổi bật **ô nhập liệu** khi người dùng nhấn vào.

### c. :active - Khi phần tử đang được nhấn

Ví dụ:

button:active {

    transform: scale(0.95);

    background-color: #c0392b;

}

📌 **Ứng dụng:** Hiệu ứng **bấm xuống** khi click vào nút.

### d. :visited - Khi liên kết đã được nhấn

Ví dụ:

a:visited {

    color: purple;

}

📌 **Ứng dụng:** Đổi màu liên kết sau khi **người dùng đã truy cập**.

## 2️. Pseudo-classes chọn phần tử dựa trên vị trí

### a. :first-child - Chọn phần tử con đầu tiên

Ví dụ:

p:first-child {

    color: red;

}

📌 **Ứng dụng:** Tô màu cho **đoạn văn đầu tiên** trong một nhóm.

### b. :last-child - Chọn phần tử con cuối cùng

Ví dụ:

li:last-child {

    font-weight: bold;

}

📌 **Ứng dụng:** Làm nổi bật **phần tử cuối** trong danh sách.

### c. :nth-child(n) - Chọn phần tử con thứ n

Ví dụ:

li:nth-child(2) {

    color: blue;

}

📌 **Ứng dụng:** Tô màu phần tử thứ **2** trong danh sách.

### d. :nth-child(odd/even) - Chọn phần tử con chẵn/lẻ

Ví dụ:

tr:nth-child(even) {

    background-color: #f2f2f2;

}

📌 **Ứng dụng:** **Tô màu xen kẽ** cho bảng dữ liệu.

### e. :nth-of-type(n) - Chọn phần tử thứ n theo kiểu loại

Ví dụ:

p:nth-of-type(3) {

    color: green;

}

📌 **Ứng dụng:** Chọn **đoạn văn thứ 3** trong trang.

### f. :first-of-type & :last-of-type - Chọn phần tử đầu/cuối theo kiểu loại

Ví dụ:

h2:first-of-type {

    color: red;

}

📌 **Ứng dụng:** Định dạng **tiêu đề đầu tiên** trong một nhóm.

## 3️. Pseudo-classes trạng thái đặc biệt

### a. :checked - Chọn phần tử checkbox hoặc radio được chọn

Ví dụ:

input[type="checkbox"]:checked {

    outline: 2px solid green;

}

📌 **Ứng dụng:** Làm nổi bật checkbox **đã chọn**.

### b. :disabled - Chọn phần tử bị vô hiệu hóa

Ví dụ:

button:disabled {

    background: gray;

    cursor: not-allowed;

}

📌 **Ứng dụng:** Định dạng **nút không thể bấm**.

### c. :not(selector) - Chọn tất cả ngoại trừ phần tử chỉ định

Ví dụ:

p:not(.special) {

    color: gray;

}

📌 **Ứng dụng:** Áp dụng CSS **cho mọi phần tử trừ một số phần tử cụ thể**.

## 🎯 Tổng kết

| **Pseudo-class** | **Mô tả** |
| --- | --- |
| :hover | Khi di chuột vào phần tử |
| :focus | Khi phần tử được chọn (focus) |
| :active | Khi phần tử đang được bấm |
| :visited | Khi liên kết đã được truy cập |
| :first-child | Phần tử đầu tiên của cha |
| :last-child | Phần tử cuối cùng của cha |
| :nth-child(n) | Phần tử thứ n của cha |
| :nth-child(odd/even) | Phần tử lẻ/chẵn của cha |
| :nth-of-type(n) | Phần tử thứ n của loại cụ thể |
| :checked | Checkbox hoặc radio được chọn |
| :disabled | Phần tử bị vô hiệu hóa |
| :not(selector) | Chọn tất cả ngoại trừ phần tử chỉ định |

# Pseudo-elements trong CSS

Pseudo-elements trong CSS giúp bạn **tạo kiểu cho một phần cụ thể của một phần tử** mà không cần thêm HTML mới. Chúng bắt đầu với :: thay vì : (mặc dù một số trình duyệt vẫn hỗ trợ :).

Dưới đây là một số pseudo-elements phổ biến và cách sử dụng chúng! 🚀

## 1. *::before* - Thêm nội dung trước phần tử

* Chèn nội dung **trước phần tử** mà không cần chỉnh sửa HTML.
* **Bắt buộc** dùng thuộc tính content.

Ví dụ:

h1::before {

    content: "🔥 ";

    color: red;

    font-size: 24px;

  }

📌 **Ứng dụng:** Thêm **biểu tượng hoặc ký tự đặc biệt** trước tiêu đề.

## 2. :*:after* - Thêm nội dung sau phần tử

* Chèn nội dung **sau phần tử**.

Ví dụ:

h1::after {

    content: " 🚀";

    color: blue;

  }

📌 **Ứng dụng:** Thêm **biểu tượng, dấu chấm, chữ ký tự động...**.

## 3. *::first-letter* - Chọn chữ cái đầu tiên của đoạn văn

* Tạo hiệu ứng **chữ cái lớn đầu đoạn văn** (*drop cap*).

Ví dụ:

p::first-letter {

    font-size: 2rem;

    font-weight: bold;

    color: crimson;

  }

📌 **Ứng dụng:** Tạo **hiệu ứng truyện cổ tích**, bài báo,...

## 4. *::first-line* - Chọn dòng đầu tiên của đoạn văn

* Tạo kiểu **chỉ cho dòng đầu tiên** trong một đoạn văn.

Ví dụ:

p::first-line {

    font-weight: bold;

    color: darkblue;

  }

📌 **Ứng dụng:** Làm nổi bật **dòng đầu tiên** trong nội dung bài viết.

## 5. ::selection - Tạo kiểu cho nội dung khi bôi đen

* Tạo kiểu cho văn bản khi **người dùng chọn (highlight)**.

Ví dụ:

p::selection {

    background: yellow;

    color: black;

  }

📌 **Ứng dụng:** Làm **trải nghiệm đọc thú vị hơn**.

## 6. *::placeholder* - Tạo kiểu cho placeholder trong input

* Dùng để **tạo kiểu cho văn bản gợi ý** trong các ô nhập liệu.

Ví dụ:

input::placeholder {

    color: gray;

    font-style: italic;

  }

📌 **Ứng dụng:** Cải thiện **trải nghiệm người dùng** với các gợi ý trong form.

## 📝 Tổng hợp các Pseudo-elements phổ biến

| **Pseudo-element** | **Chức năng** |
| --- | --- |
| ::before | Thêm nội dung trước phần tử |
| ::after | Thêm nội dung sau phần tử |
| ::first-letter | Tạo kiểu cho chữ cái đầu tiên của đoạn văn |
| ::first-line | Tạo kiểu cho dòng đầu tiên của đoạn văn |
| ::selection | Tạo kiểu cho văn bản được bôi đen |
| ::placeholder | Tạo kiểu cho văn bản gợi ý trong input |

# Thuộc tính padding trong CSS (đệm)

padding là **khoảng cách giữa nội dung của một phần tử và viền (border) của nó**. Nó giúp tạo khoảng trống bên trong phần tử, làm cho nội dung không bị dính sát vào viền.

## 1. Cách sử dụng padding

Bạn có thể đặt padding theo nhiều cách khác nhau:

### a. giá trị – Áp dụng cho cả 4 cạnh

div {

    padding: 20px;  /\* Trên = Dưới = Trái = Phải = 20px \*/

  }

### b. giá trị – Áp dụng cho (Trên/Dưới) và (Trái/Phải)

div {

    padding: 10px 20px;  /\* Trên & Dưới = 10px, Trái & Phải = 20px \*/

  }

### c. giá trị – (Trên) (Trái/Phải) (Dưới)

div {

    padding: 10px 15px 20px;  /\* Trên = 10px, Trái & Phải = 15px, Dưới = 20px \*/

  }

### d. giá trị – (Trên) (Phải) (Dưới) (Trái)

div {

    padding: 5px 10px 15px 20px;

    /\* Trên = 5px, Phải = 10px, Dưới = 15px, Trái = 20px \*/

  }

## 2. padding theo từng cạnh riêng lẻ

Bạn cũng có thể đặt từng cạnh riêng biệt:

div {

    padding-top: 10px;

    padding-right: 15px;

    padding-bottom: 20px;

    padding-left: 25px;

  }

## 3. padding với đơn vị khác nhau

Ngoài **px**, bạn có thể sử dụng nhiều đơn vị khác nhau:

| **Đơn vị** | **Ý nghĩa** |
| --- | --- |
| px | Pixel (cố định) |
| % | Phần trăm của chiều rộng phần tử cha |
| em | Dựa theo kích thước font của phần tử |
| rem | Dựa theo kích thước font của html (gốc) |
| vw | Phần trăm chiều rộng của viewport |
| vh | Phần trăm chiều cao của viewport |

Ví dụ dùng % :

div {

    width: 50%;

    padding: 10%;

  }

📌 **Khi dùng %, padding sẽ được tính theo width của phần tử cha**.

## 4. box-sizing và padding

Mặc định, padding làm **tăng kích thước tổng thể** của phần tử. Nếu bạn muốn kích thước **không thay đổi**, hãy dùng box-sizing: border-box;.

Ví dụ:

div {

    width: 200px;

    padding: 20px;

    box-sizing: border-box;  /\* Kích thước không bị thay đổi \*/

  }

## 🎯 Tổng kết

* padding tạo **khoảng cách bên trong phần tử**.
* Có thể thiết lập padding cho từng cạnh hoặc dùng giá trị rút gọn.
* Hỗ trợ nhiều **đơn vị kích thước** khác nhau (px, %, em, rem...).
* Dùng box-sizing: border-box; để giữ nguyên kích thước phần tử.

# Thuộc tính border trong CSS (đường viền)

border là thuộc tính **tạo đường viền xung quanh phần tử**. Nó giúp làm nổi bật nội dung hoặc tạo khung cho các thành phần trong giao diện.

## 1. Cấu trúc của border

***border*** có **3 thành phần chính**:

***border:*** [độ dày] [kiểu] [màu sắc];

Ví dụ cơ bản:

div {

    border: 2px solid red;

  }

📌 **Giải thích:**

* 2px – Độ dày của đường viền
* solid – Kiểu đường viền (liền)
* red – Màu sắc của đường viền

## 2. Các kiểu đường viền (border-style)

Bạn có thể thay đổi kiểu đường viền bằng thuộc tính ***border-style***:

| **Giá trị** | **Ý nghĩa** |
| --- | --- |
| solid | Đường liền |
| dashed | Đường gạch đứt |
| dotted | Đường chấm chấm |
| double | Đường đôi |
| groove | Hiệu ứng 3D khắc chìm |
| ridge | Hiệu ứng 3D nổi lên |
| inset | Hiệu ứng chìm vào trong |
| outset | Hiệu ứng nổi ra ngoài |
| none | Không có viền |
| hidden | Ẩn viền (giống none, nhưng có thể ảnh hưởng layout bảng) |

Ví dụ với nhiều kiểu đường viền:

div {

    border-style: dashed;

  }

## 3. Đặt đường viền cho từng cạnh riêng lẻ

Bạn có thể đặt viền **chỉ cho từng cạnh**:

div {

    border-top: 3px solid blue;

    border-right: 5px dashed green;

    border-bottom: 2px dotted red;

    border-left: 4px double black;

  }

Hoặc sử dụng các thuộc tính rút gọn:

border-width: 5px;       /\* Độ dày viền \*/

border-style: solid;      /\* Kiểu viền \*/

border-color: blue;       /\* Màu viền \*/

## 4. Bo tròn góc với border-radius

Bạn có thể làm viền tròn hoặc bo góc bằng border-radius:

✅ **Bo góc nhẹ:**

div {

    border: 2px solid black;

    border-radius: 10px;  /\* Bo góc 10px \*/

  }

✅ **Làm hình tròn:**

div {

    width: 100px;

    height: 100px;

    border: 2px solid black;

    border-radius: 50%;

  }

## 5. border và box-sizing

Mặc định, border **làm tăng kích thước tổng thể** của phần tử. Để tránh điều này, dùng box-sizing: border-box;.

Ví dụ:

div {

    width: 200px;

    padding: 10px;

    border: 5px solid black;

    box-sizing: border-box; /\* Đảm bảo kích thước không bị tăng thêm \*/

  }

## 🎯 Tổng kết

* **border** tạo **đường viền xung quanh phần tử**.
* Gồm 3 thành phần: **độ dày**, **kiểu**, **màu sắc**.
* Có thể đặt viền cho **từng cạnh riêng biệt** **(border-top, border-right, ...**).
* **border-radius** giúp **bo góc** hoặc tạo hình tròn.
* Dùng **box-sizing: border-box**; để giữ nguyên kích thước phần tử.

# Thuộc tính margin trong CSS (khoảng cách lề)

**margin** là thuộc tính **tạo khoảng cách lề bên ngoài phần tử**, giúp kiểm soát khoảng cách giữa các phần tử trong bố cục trang web.

## 1. Cấu trúc của margin

Bạn có thể sử dụng margin theo nhiều cách khác nhau:

**margin: [trên] [phải] [dưới] [trái];**

Ví dụ cơ bản:

div {

    margin: 20px;  /\* Áp dụng 20px cho tất cả các cạnh \*/

  }

## 2. Đặt margin riêng lẻ cho từng cạnh

Bạn có thể đặt **khoảng cách lề cho từng cạnh riêng biệt**:

div {

    margin-top: 10px;    /\* Lề trên \*/

    margin-right: 15px;  /\* Lề phải \*/

    margin-bottom: 20px; /\* Lề dưới \*/

    margin-left: 25px;   /\* Lề trái \*/

  }

📌 **Lưu ý:** Nếu chỉ đặt margin cho một phía, các phía khác **không bị ảnh hưởng**.

## 3. Cách viết rút gọn (margin shorthand)

Bạn có thể sử dụng margin với **1, 2, 3 hoặc 4 giá trị**:

| **Cách viết** | **Tương đương** |
| --- | --- |
| margin: 10px; | margin-top: 10px; margin-right: 10px; margin-bottom: 10px; margin-left: 10px; |
| margin: 10px 20px; | margin-top: 10px; margin-bottom: 10px; margin-right: 20px; margin-left: 20px; |
| margin: 10px 20px 30px; | margin-top: 10px; margin-right: 20px; margin-left: 20px; margin-bottom: 30px; |
| margin: 10px 20px 30px 40px; | margin-top: 10px; margin-right: 20px; margin-bottom: 30px; margin-left: 40px; |

Ví dụ:

div {

    margin: 10px 15px 20px 25px;

  }

**Khi dùng 4 giá trị:**

* **Giá trị 1** → top
* **Giá trị 2** → right
* **Giá trị 3** → bottom
* **Giá trị 4** → left

## 4. Sử dụng auto để căn giữa

**margin: auto**; thường dùng để **căn giữa phần tử** khi có **chiều rộng xác định** (width).

Ví dụ căn giữa div:

div {

    width: 300px;

    margin: 0 auto; /\* Căn giữa theo chiều ngang \*/

  }

📌 **Lưu ý:** margin: auto; **chỉ hoạt động khi có width**.

## 5. margin và box-sizing

* Mặc định, **margin** **không ảnh hưởng đến kích thước của phần tử**.
* Dùng **box-sizing: border-box**; để **giữ nguyên kích thước phần tử**, nhưng **không làm thay đổi margin**.

## 6. Hiện tượng margin collapse (sụp lề)

Khi hai phần tử có **margin** chạm nhau (theo chiều dọc), **giá trị lớn hơn sẽ được sử dụng thay vì cộng dồn**.

Ví dụ **về margin collapse:**

.div1 {

    margin-bottom: 30px;

}

.div2 {

    margin-top: 20px;

}

👉 *Khoảng cách thực tế giữa .div1 và .div2 là 30px, không phải 50px.*

🚀 **Cách khắc phục:** Dùng **padding** hoặc **overflow: hidden**; để tránh **margin collapse.**

## 🎯 Tổng kết

* **margin** tạo **khoảng cách bên ngoài phần tử**.
* Có thể đặt **cho từng cạnh riêng lẻ** (**margin-top, margin-right,** ...).
* **margin: auto**; giúp **căn giữa phần tử** theo chiều ngang.
* **margin collapse** làm **gộp khoảng cách lề theo chiều dọc**.

# ****Thuộc tính**** box-sizing ****trong CSS****

**box-sizing** giúp kiểm soát cách tính toán kích thước của phần tử, đặc biệt là khi sử dụng **width** và **height.**

## ****1. Cách hoạt động của**** box-sizing

Mặc định, khi bạn đặt width và height cho một phần tử, kích thước **chỉ áp dụng cho phần nội dung (content)**, còn padding và border sẽ làm tổng kích thước tăng lên.

* + Nếu không dùng **box-sizing**, kích thước thực tế của phần tử có thể **lớn hơn mong đợi**.

## ****2. Các giá trị của**** box-sizing

### ****a.**** content-box ****(mặc định)****

* width và height **chỉ tính phần nội dung**.
* padding và border **bị cộng thêm vào tổng kích thước thực tế**.

**Ví dụ**:

.box {

    width: 200px;

    height: 100px;

    padding: 20px;

    border: 10px solid black;

    box-sizing: content-box;

  }

📌 **Kích thước thực tế của phần tử**:  
👉 **Rộng** = 200px (nội dung) + 20px \* 2 (padding) + 10px \* 2 (border) = 260px  
👉 **Cao** = 100px (nội dung) + 20px \* 2 (padding) + 10px \* 2 (border) = 160px

### ****b.**** border-box ****(hay dùng nhất)****

* width và height **bao gồm cả padding và border**.
* Phần nội dung sẽ tự động **thu nhỏ lại** để giữ nguyên kích thước tổng.

**Ví dụ**:

.box {

    width: 200px;

    height: 100px;

    padding: 20px;

    border: 10px solid black;

    box-sizing: border-box;

  }

📌 **Kích thước thực tế của phần tử**:  
👉 **Rộng** = 200px (không thay đổi)  
👉 **Cao** = 100px (không thay đổi)

💡 **Lưu ý:** Với **border-box**, **phần nội dung sẽ bị thu nhỏ lại** để padding và border ***không làm tăng kích thước tổng.***

## ****3. So sánh**** content-box ****vs**** border-box

| **Thuộc tính** | **content-box (Mặc định)** | **border-box** |
| --- | --- | --- |
| **Width / Height** | Chỉ tính phần **nội dung** | Tính cả padding & border |
| **Padding** | **Cộng thêm vào tổng kích thước** | **Không làm thay đổi tổng kích thước** |
| **Border** | **Cộng thêm vào tổng kích thước** | **Không làm thay đổi tổng kích thước** |
| **Dùng khi nào?** | Khi cần giữ kích thước nội dung cố định | Khi muốn phần tử không vượt quá kích thước mong muốn |

## ****4. Áp dụng**** box-sizing ****cho toàn trang****

Bạn có thể đặt **box-sizing: border-box**; cho toàn bộ trang để tránh các vấn đề về kích thước:

\* {

    box-sizing: border-box;

  }

✅ **Lợi ích của cách này:**

* **Dễ kiểm soát kích thước phần tử**.
* **Tránh lỗi layout khi thêm padding hoặc border**.

## ****🎯 Tổng kết****

* box-sizing **quyết định cách tính kích thước của phần tử**.
* content-box (mặc định) **chỉ tính phần nội dung**, còn padding & border sẽ làm tăng kích thước thực tế.
* border-box **tính cả padding & border vào tổng kích thước**, giúp phần tử không vượt quá kích thước mong muốn.
* Dùng \* { box-sizing: border-box; } để tránh lỗi về kích thước trên toàn trang.

# Thuộc tính background-image trong CSS

## 1. Giới thiệu

Thuộc tính **background-image** được sử dụng để đặt hình ảnh làm nền cho một phần tử HTML.

## 2. Cách sử dụng cơ bản

**Cú pháp:**

**selector {**

**background-image: url("duong-dan-hinh-anh.jpg");**

**}**

Ví dụ:

.box {

    width: 300px;

    height: 200px;

    background-image: url("https://example.com/image.jpg");

  }

📌 Hình ảnh từ "https://example.com/image.jpg" sẽ làm nền cho phần tử .box.

## 3. Sử dụng nhiều hình ảnh nền

Bạn có thể sử dụng **nhiều ảnh nền**, ảnh trên cùng sẽ che ảnh dưới:

.box {

    background-image: url("top-layer.png"), url("bottom-layer.png");

  }

📌 **Hình ảnh top-layer.png** sẽ hiển thị trên bottom-layer.png.

## 4. Các thuộc tính liên quan

### a. background-repeat (Lặp lại hình nền)

Mặc định, hình nền sẽ tự động lặp lại. Bạn có thể thay đổi nó:

.box {

    background-image: url("image.jpg");

    background-repeat: no-repeat;

  }

📌 Các giá trị:

* **repeat** (mặc định) – Lặp lại theo cả **chiều ngang & dọc**.
* **repeat-x** – Lặp lại theo **chiều ngang**.
* **repeat-y** – Lặp lại theo **chiều dọc**.
* **no-repeat** – **Không lặp lại**.

### b. background-position (Vị trí hình nền)

Xác định vị trí hiển thị của hình nền:

.box {

    background-image: url("image.jpg");

    background-position: center center;

  }

📌 Các giá trị phổ biến:

* **left top, left center, left bottom**
* **center top, center center, center bottom**
* **right top, right center, right bottom**
* Hoặc dùng **tọa độ cụ thể**: 50px 20px

### c. background-size (Kích thước hình nền)

Điều chỉnh kích thước hình nền:

.box {

    background-image: url("image.jpg");

    background-size: cover;

  }

📌 Các giá trị:

* auto (mặc định) – Giữ nguyên kích thước gốc.
* cover – **Phủ kín phần tử** (có thể cắt bớt hình ảnh).
* contain – **Hiển thị toàn bộ ảnh**, có thể để lại khoảng trống.
* Hoặc **kích thước cụ thể**: 100px 200px, 50% 50%.

### d. background-attachment (Cố định hình nền khi cuộn)

.box {

    background-image: url("image.jpg");

    background-attachment: fixed;

  }

📌 Các giá trị:

* scroll (mặc định) – Hình nền **cuộn theo trang**.
* fixed – Hình nền **cố định khi cuộn**.
* local – Hình nền cuộn theo nội dung phần tử.

### e. background (Viết gộp)

Thay vì viết từng dòng, bạn có thể gộp lại:

.box {

    background: url("image.jpg") no-repeat center/cover fixed;

  }

📌 Thứ tự các giá trị:

1. background-image
2. background-repeat
3. background-position
4. background-size
5. background-attachment

## 5. Tổng kết

* background-image giúp đặt hình ảnh làm nền.
* Kết hợp với các thuộc tính khác để kiểm soát hiển thị:
  + background-repeat: Điều chỉnh lặp lại.
  + background-position: Điều chỉnh vị trí.
  + background-size: Điều chỉnh kích thước.
  + background-attachment: Cố định hoặc cuộn theo trang.
* Dùng **viết gộp background** để tối ưu code.

# Thuộc tính background-origin trong CSS

## 1. Giới thiệu

Thuộc tính **background-origin** xác định **vị trí bắt đầu hiển thị của hình nền** bên trong phần tử. Nó quy định **hình nền sẽ bắt đầu từ đâu** trong hộp CSS (**border-box**, **padding-box**, hoặc **content-box**).

## 2. Các giá trị của background-origin

**Cú pháp:**

**selector {**

**background-origin: value;**

**}**

📌 **Các giá trị hợp lệ:**

| **Giá trị** | **Mô tả** |
| --- | --- |
| border-box | Hình nền bắt đầu từ **đường viền** (bao gồm border). |
| padding-box | Hình nền bắt đầu từ **phần padding** (không bao gồm border). *(Mặc định)* |
| content-box | Hình nền bắt đầu từ **bên trong nội dung** (không bao gồm border & padding). |

## 3. Ví dụ minh họa

### a. Ví dụ 1: So sánh 3 giá trị của background-origin

.box {

    width: 300px;

    height: 200px;

    border: 10px solid black;

    padding: 20px;

    background-image: url("https://via.placeholder.com/300");

    background-repeat: no-repeat;

    background-size: cover;

  }

.border-box {

    background-origin: border-box;

  }

.padding-box {

    background-origin: padding-box;

  }

.content-box {

    background-origin: content-box;

  }

✅ **Giải thích:**

* border-box: Hình nền trải dài từ viền ngoài.
* padding-box: Hình nền chỉ bắt đầu từ phần padding.
* content-box: Hình nền chỉ nằm trong phần nội dung.

### b. Ví dụ 2: Kết hợp với background-clip

Bạn có thể kết hợp **background-origin** với **background-clip** để kiểm soát phạm vi hiển thị hình nền.

.box {

    width: 300px;

    height: 200px;

    border: 10px solid red;

    padding: 20px;

    background-image: url("https://via.placeholder.com/300");

    background-repeat: no-repeat;

    background-size: cover;

    background-origin: content-box;

    background-clip: content-box;

  }

📌 **Trong ví dụ này:**

* Hình nền chỉ bắt đầu từ content-box (bên trong nội dung).
* background-clip: content-box đảm bảo hình ảnh chỉ hiển thị trong nội dung, không lan ra padding hay border.

## 4. Khi nào nên sử dụng background-origin?

* Khi bạn muốn hình nền **bắt đầu từ viền**, padding, hoặc nội dung.
* Khi cần **tinh chỉnh hình nền** cho các phần tử có **border hoặc padding lớn**.
* Khi kết hợp với background-clip để tạo hiệu ứng **cắt hình nền**.

## 5. Tổng kết

| **Giá trị** | **Ảnh nền bắt đầu từ** |
| --- | --- |
| **border-box** | **Viền ngoài (border)** |
| **padding-box** | **Phần padding (Mặc định)** |
| **content-box** | **Bên trong nội dung** |

📌 **background-origin** rất hữu ích khi thiết kế **nút bấm**, **thẻ bài**, hoặc **các phần tử có border & padding lớn**.

# Thuộc tính background-position trong CSS

## 1. Giới thiệu

Thuộc tính background-position xác định **vị trí đặt hình nền** bên trong một phần tử HTML. Nó cho phép bạn điều chỉnh hình nền theo tọa độ **ngang (x)** và **dọc (y)**.

## 2. Cú pháp của background-position

**selector {**

**background-position: x y;**

**}**

* x: Vị trí theo **chiều ngang** (trái → phải).
* y: Vị trí theo **chiều dọc** (trên → dưới).
* Nếu chỉ có **một giá trị**, giá trị còn lại sẽ mặc định là center.

## 3. Các giá trị của background-position

| **Giá trị** | **Mô tả** |
| --- | --- |
| left top | Góc trên bên trái |
| left center | Giữa bên trái |
| left bottom | Góc dưới bên trái |
| right top | Góc trên bên phải |
| right center | Giữa bên phải |
| right bottom | Góc dưới bên phải |
| center top | Giữa trên |
| center center (hoặc chỉ center) | Ở chính giữa |
| center bottom | Giữa dưới |
| x% y% | Dùng phần trăm (%) để căn chỉnh theo kích thước của phần tử |
| x px y px | Sử dụng pixel (px) để định vị chính xác hình nền |

## 4. Ví dụ minh họa

### a. Ví dụ 1: Căn hình nền vào các góc

.box {

    width: 300px;

    height: 200px;

    border: 2px solid black;

    background-image: url("https://via.placeholder.com/300");

    background-size: cover;

  }

  .top-left {

    background-position: left top;

  }

  .center {

    background-position: center;

  }

 .bottom-right {

    background-position: right bottom;

  }

📌 **Giải thích:**

* **left top**: Hình nền nằm ở góc trên bên trái.
* **center**: Hình nền nằm ở chính giữa.
* **right bottom**: Hình nền nằm ở góc dưới bên phải.

### b. Ví dụ 2: Dùng phần trăm (%) để định vị

.box {

    width: 300px;

    height: 200px;

    border: 2px solid black;

    background-image: url("https://via.placeholder.com/300");

    background-size: cover;

    background-position: 30% 70%;

  }

📌 **Giải thích:**

* **30% 70%** có nghĩa là **đặt điểm 30% theo chiều ngang và 70% theo chiều dọc**.

### c. Ví dụ 3: Dùng đơn vị pixel (px)

.box {

    width: 300px;

    height: 200px;

    border: 2px solid black;

    background-image: url("https://via.placeholder.com/300");

    background-size: cover;

    background-position: 50px 100px;

  }

📌 **Giải thích:**

* 50px 100px có nghĩa là **dịch hình nền 50px từ trái và 100px từ trên xuống**.

## 5. Khi nào nên dùng background-position?

✅ Khi muốn **định vị hình nền** theo vị trí mong muốn.  
✅ Khi kết hợp với **background-size: cover**; để hiển thị hình nền chuẩn đẹp.  
✅ Khi tạo **hiệu ứng thị giác**, như căn chỉnh hình nền để phù hợp với bố cục.

## 6. Tổng kết

| **Giá trị** | **Ý nghĩa** |
| --- | --- |
| left top | Hình nền bắt đầu từ góc trái trên |
| center center | Hình nền căn giữa |
| right bottom | Hình nền ở góc phải dưới |
| % % | Vị trí dựa trên phần trăm |
| px px | Vị trí theo pixel |

📌 **Gợi ý:** Bạn có thể kết hợp với **background-size, background-repeat** để tối ưu hiển thị.

# Cú pháp "shorthand" cho background trong CSS

## 1. Giới thiệu

**background** là một **thuộc tính shorthand** (viết tắt) trong CSS, giúp bạn khai báo nhiều thuộc tính liên quan đến nền trong **một dòng duy nhất** thay vì viết từng thuộc tính riêng lẻ.

## 2. Cú pháp đầy đủ của background

**selector {**

**background: color image position/size repeat origin clip attachment;**

**}**

📌 Các thuộc tính có thể gộp vào **background**:

1. **background-color** → Màu nền
2. **background-image** → Hình nền
3. **background-position** → Vị trí hình nền
4. **background-size** → Kích thước hình nền (viết sau dấu /)
5. **background-repeat** → Kiểu lặp hình nền
6. **background-origin** → Gốc vẽ hình nền
7. **background-clip** → Phạm vi cắt hình nền
8. **background-attachment** → Hình nền cố định hoặc cuộn theo trang

## 3. Ví dụ minh họa

### a. Ví dụ 1: Viết đầy đủ từng thuộc tính

.box {

    background-color: lightblue;

    background-image: url("https://via.placeholder.com/300");

    background-position: center;

    background-size: cover;

    background-repeat: no-repeat;

    background-attachment: fixed;

  }

### b.Ví dụ 2: Viết gọn bằng background shorthand

.box {

    background: lightblue url("https://via.placeholder.com/300") center/cover no-repeat fixed;

  }

📌 **Giải thích:**

* **lightblue** → Màu nền
* **url**("...") → Hình nền
* **center** → Căn giữa
* **/cover** → Hình nền phủ toàn bộ
* **no-repeat** → Không lặp lại
* **fixed** → Cố định khi cuộn

### c. Ví dụ 3: Chỉ dùng màu nền

.box {

    background: #ffcc00;

  }

### d. Ví dụ 4: Chỉ dùng hình nền với vị trí và lặp lại

.box {

    background: url("https://via.placeholder.com/300") left top repeat;

  }

### e. Ví dụ 5: Định vị và kích thước hình nền

.box {

    background: url("https://via.placeholder.com/300") 50% 50% / contain no-repeat;

  }

📌 **Giải thích:**

* **50% 50%** → Căn giữa hình nền
* **/contain** → Hình nền vừa khớp trong phần tử mà không bị cắt

## 5. Tổng kết

| **Cách viết** | **Ví dụ** |
| --- | --- |
| Chỉ màu nền | background: red; |
| Chỉ hình nền | background: url('image.jpg'); |
| Màu + hình nền | background: lightblue url('image.jpg'); |
| Vị trí + kích thước | background: url('image.jpg') center/cover; |
| Full thuộc tính | background: #000 url('image.jpg') center/contain no-repeat fixed; |

📌 **Mẹo:** Dùng **shorthand** giúp code **ngắn gọn hơn**, nhưng nếu có nhiều giá trị, bạn nên viết riêng để **dễ đọc và bảo trì**. 🚀

# Các giá trị màu sắc trong CSS

Trong CSS, màu sắc có thể được biểu diễn bằng nhiều cách khác nhau. Dưới đây là các kiểu giá trị màu phổ biến trong CSS:

## 1️. Tên màu (Color Names)

📌 CSS hỗ trợ một số tên màu chuẩn mà bạn có thể sử dụng trực tiếp.

Ví dụ:

color: red;

background-color: blue;

border: 2px solid green;

✅ **Một số màu phổ biến:**

* black, white, red, blue, green, yellow, purple, orange, pink, gray, brown, v.v.

👉 **Danh sách đầy đủ**: [Danh sách màu CSS](https://developer.mozilla.org/en-US/docs/Web/CSS/color_value)

## 2️. Mã màu HEX (Hexadecimal Colors)

📌 Mã **HEX** được biểu diễn bằng dấu # theo sau bởi 3 hoặc 6 ký tự (số hoặc chữ cái từ 0-9 và A-F).

Ví dụ:

color: #ff0000; /\* Màu đỏ \*/

background-color: #00ff00; /\* Màu xanh lá \*/

border: 2px solid #0000ff; /\* Màu xanh dương \*/

✅ **Rút gọn HEX (3 ký tự):**

color: #f00; /\* #ff0000 \*/

    color: #0f0; /\* #00ff00 \*/

    color: #00f; /\* #0000ff \*/

📌 Khi rút gọn, mỗi ký tự sẽ được nhân đôi **(#f00 → #ff0000**).

## 3️. Màu RGB (Red, Green, Blue)

📌 Giá trị RGB xác định mức độ màu **đỏ (Red), xanh lá (Green), xanh dương (Blue)** từ 0 đến 255.

Ví dụ:

    color: rgb(255, 0, 0); /\* Màu đỏ \*/

background-color: rgb(0, 255, 0); /\* Màu xanh lá \*/

border: 2px solid rgb(0, 0, 255); /\* Màu xanh dương \*/

✅ **Định dạng RGB với độ trong suốt (rgba)**  
Bạn có thể sử dụng **rgba(r, g, b, a)**, trong đó **a** là giá trị **alpha (độ trong suốt)** từ 0 (hoàn toàn trong suốt) đến 1 (không trong suốt).

Ví dụ:

    color: rgba(255, 0, 0, 0.5); /\* Màu đỏ với 50% trong suốt \*/

    background-color: rgba(0, 0, 255, 0.2); /\* Màu xanh dương với 20% trong suốt \*/

## 4️. Màu HSL (Hue, Saturation, Lightness)

📌 HSL đại diện cho màu theo 3 thành phần:

* **Hue** (tông màu) (0 - 360 độ, ví dụ: 0 = đỏ, 120 = xanh lá, 240 = xanh dương)
* **Saturation** (độ bão hòa) (%), 100% là màu đậm nhất, 0% là màu xám
* **Lightness** (độ sáng) (%), 0% là đen, 50% là màu chính, 100% là trắng

Ví dụ:

    color: hsl(0, 100%, 50%); /\* Màu đỏ \*/

    background-color: hsl(120, 100%, 50%); /\* Màu xanh lá \*/

    border: 2px solid hsl(240, 100%, 50%); /\* Màu xanh dương \*/

✅ **HSL với độ trong suốt (hsla)** Tương tự như **rgba, hsla(h, s, l, a)** thêm một giá trị alpha (0 - 1).

Ví dụ:

    color: hsla(0, 100%, 50%, 0.5); /\* Màu đỏ 50% trong suốt \*/

## 5️. Màu hệ thống (System Colors)

📌 Các màu này thay đổi dựa trên giao diện hệ thống của người dùng.

Ví dụ:

    color: ButtonText; /\* Màu chữ của nút \*/

    background-color: Canvas; /\* Màu nền hệ thống \*/

    border: 1px solid Highlight; /\* Màu vùng được chọn \*/

## 6️. CurrentColor (Màu kế thừa)

📌 Giá trị **currentColor** giúp kế thừa màu chữ của phần tử cha.

Ví dụ:

.box {

    color: red;

    border: 2px solid currentColor; /\* Biên sẽ có cùng màu với chữ \*/

  }

**📌 Tổng kết: Các cách biểu diễn màu sắc**

| **Cách** | **Ví dụ** | **Ghi chú** |
| --- | --- | --- |
| ***Tên màu*** | red, blue, green | Dễ nhớ nhưng không linh hoạt |
| ***HEX*** | #ff0000, #00ff00, #0000ff | Phổ biến nhất, có thể rút gọn |
| ***RGB*** | rgb(255, 0, 0), rgb(0, 255, 0) | Trực quan, dễ điều chỉnh |
| ***RGBA*** | rgba(255, 0, 0, 0.5) | Có thêm độ trong suốt |
| ***HSL*** | hsl(0, 100%, 50%) | Dễ điều chỉnh độ sáng & bão hòa |
| ***HSLA*** | hsla(0, 100%, 50%, 0.5) | Có thêm độ trong suốt |
| ***System Colors*** | ButtonText, Canvas | Phụ thuộc vào hệ điều hành |
| ***currentColor*** | border: 2px solid currentColor; | Kế thừa màu chữ hiện tại |

🔥 **Lời khuyên:**

* **HEX** là lựa chọn phổ biến nhất cho thiết kế web.
* **RGB/RGBA** hữu ích khi cần màu sắc chính xác và trong suốt.
* **HSL/HSLA** giúp dễ dàng điều chỉnh độ sáng và độ bão hòa.
* **Tên màu CSS** dễ nhớ nhưng bị giới hạn.

# Thuộc tính position: relative; trong CSS

## 1️. position: relative; là gì?

📌 Khi một phần tử có **position: relative;,** nó sẽ **giữ nguyên vị trí ban đầu** trong tài liệu, nhưng có thể **dịch chuyển** bằng các thuộc tính **top, right, bottom, left.**

## 2️. Cách hoạt động của position: relative;

* Không ảnh hưởng đến các phần tử khác.
* Phần tử vẫn chiếm không gian **ban đầu**, nhưng có thể dịch chuyển **tương đối với chính nó**.
* Các phần tử con **absolute** bên trong nó sẽ được **định vị tương đối với phần tử này**.

## 3️. Cú pháp & Ví dụ

🔹 **Cơ bản:**

.box {

    position: relative;

    top: 20px;    /\* Dịch xuống 20px \*/

    left: 30px;   /\* Dịch sang phải 30px \*/

    background-color: lightblue;

    width: 200px;

    height: 100px;

}

🔹 **HTML:**

<div class="box">Tôi được dịch chuyển!</div>

✅ **Kết quả:** Phần tử di chuyển **xuống 20px và sang phải 30px**, nhưng khoảng trống ban đầu vẫn giữ nguyên.

## 4️. So sánh relative với các position khác

| **Giá trị** | **Mô tả** |
| --- | --- |
| **static (mặc định)** | Phần tử nằm đúng theo luồng tài liệu, không bị di chuyển. |
| **relative** | Phần tử di chuyển dựa trên vị trí ban đầu mà không ảnh hưởng đến bố cục chung. |
| **absolute** | Phần tử được loại khỏi luồng tài liệu, định vị theo **relative** của cha gần nhất. |
| **fixed** | Phần tử cố định trên màn hình, không bị cuộn. |
| **sticky** | Phần tử hoạt động như relative, nhưng sẽ dính khi cuộn đến một vị trí nhất định. |

## 5️. Ứng dụng relative thực tế

### a. Căn chỉnh một phần tử so với vị trí ban đầu của nó:

.button {

    position: relative;

    left: 10px;

    top: 5px;

  }

### b. Làm cha chứa cho absolute (Tạo tooltip, dropdown, popup, v.v.):

.container {

    position: relative;

    width: 200px;

    height: 100px;

    background-color: lightgray;

  }

  .child {

    position: absolute;

    top: 0;

    left: 0;

    background-color: red;

    width: 50px;

    height: 50px;

  }

📌 **Giải thích:**

* **.container** có **position: relative**;, giúp **.child** (với position: absolute;) **định vị dựa trên nó** thay vì **body**.

## 6️. Kết luận

✅ **position: relative;** hữu ích khi:  
*✔️ Muốn di chuyển phần tử mà* ***không làm ảnh hưởng bố cục xung quanh****.  
✔️ Dùng làm* ***phần tử cha cho absolute*** *để quản lý vị trí tốt hơn.  
✔️ Tạo hiệu ứng nhỏ như* ***nút bấm, tooltip, dropdown****.*

# CSS position: absolute

## 1️. position: absolute; là gì?

📌 **position: absolute;** giúp phần tử **thoát khỏi luồng tài liệu bình thường** và được định vị dựa trên **phần tử cha gần nhất có position: relative | absolute | fixed.**

* Nếu không có phần tử cha nào có **position**, nó sẽ **định vị theo <html> (hoặc <body>)**.

## 2️. Cách hoạt động

✅ **Không chiếm không gian trong bố cục**.  
✅ **Có thể di chuyển bằng top, right, bottom, left.**  
✅ **Định vị dựa trên phần tử cha có position: relative; gần nhất**.

## 3️. Ví dụ

### a. Ví dụ 1: absolute không có cha relative (định vị theo body)

css

.box {

    position: absolute;

    top: 50px;

    left: 100px;

    width: 200px;

    height: 100px;

    background-color: lightblue;

  }

html

<div class="box">Tôi được định vị theo `body`</div>

✅ **Kết quả:** **.box** **nằm cách top 50px, left 100px so với body**.

### b. Ví dụ 2: absolute bên trong relative

css

.container {

    position: relative;

    width: 300px;

    height: 200px;

    background-color: lightgray;

  }

  .child {

    position: absolute;

    top: 10px;

    right: 20px;

    width: 100px;

    height: 50px;

    background-color: red;

  }

html

<div class="container">

  <div class="child">Tôi được định vị theo `.container`</div>

</div>

✅ **Kết quả:** **.child** **nằm trong .container và cách top 10px, right 20px**.

## 4️. So sánh với các position khác

| **Giá trị** | **Mô tả** |
| --- | --- |
| **static (mặc định)** | Không thể dùng top, left, right, bottom. |
| **relative** | Dịch chuyển dựa theo vị trí ban đầu. |
| **absolute** | Thoát khỏi luồng, định vị theo phần tử cha relative. |
| **fixed** | Giống absolute nhưng cố định trên màn hình. |
| **sticky** | Giữ nguyên vị trí đến khi cuộn tới một điểm nào đó. |

## 5️. Ứng dụng absolute thực tế

✔️ **Tạo badge thông báo góc trên của nút**

css

.button {

  position: relative;

  padding: 10px 20px;

  background-color: blue;

  color: white;

  display: inline-block;

}

.badge {

  position: absolute;

  top: -5px;

  right: -5px;

  background-color: red;

  color: white;

  padding: 5px;

  border-radius: 50%;

  font-size: 12px;

}

html

<div class="button">

    Nút bấm

    <div class="badge">3</div>

  </div>

✅ **Kết quả:** Số 3 hiển thị góc trên phải của nút.

## 6️. Kết luận

📌 **absolute** **giúp định vị linh hoạt mà không ảnh hưởng bố cục xung quanh**.  
📌 **Nhớ đặt relative cho phần tử cha để kiểm soát vị trí!** 🔥

Top of Form

Bottom of Form

# CSS position: fixed;

## 1️. position: fixed; là gì?

📌 **position: fixed;** giúp phần tử **thoát khỏi luồng bình thường** và được cố định **so với cửa sổ trình duyệt (viewport)**.  
📌 **Không bị ảnh hưởng khi cuộn trang**.

## 2️. Cách hoạt động

✅ **Không chiếm không gian trong bố cục**.  
✅ **Cố định tại một vị trí và không bị cuộn theo trang**.  
✅ **Sử dụng top, right, bottom, left để định vị**.

## 3️. Ví dụ

### a. Ví dụ 1: Thanh điều hướng cố định trên cùng

css

.navbar {

    position: fixed;

    top: 0;

    left: 0;

    width: 100%;

    background-color: black;

    color: white;

    padding: 10px 20px;

  }

html

<div class="navbar">Tôi là thanh điều hướng cố định</div>

✅ **Kết quả:** Thanh điều hướng luôn nằm trên cùng khi cuộn trang.

### b. Ví dụ 2: Nút "Lên đầu trang" cố định góc phải dưới

css

  .back-to-top {

    position: fixed;

    bottom: 20px;

    right: 20px;

    background-color: red;

    color: white;

    padding: 10px;

    border-radius: 5px;

  }

html

<button class="back-to-top">▲ Lên đầu</button>

✅ **Kết quả:** Nút luôn nằm ở góc phải dưới ngay cả khi cuộn.

## 4️. So sánh với các position khác

| **Giá trị** | **Mô tả** |
| --- | --- |
| static (mặc định) | Không thể dùng top, left, right, bottom. |
| relative | Dịch chuyển dựa theo vị trí ban đầu. |
| absolute | Định vị dựa trên phần tử cha relative. |
| fixed | Cố định trên màn hình, không bị cuộn. |
| sticky | Giữ nguyên vị trí đến khi cuộn tới một điểm nào đó. |

## 5️. Ứng dụng fixed thực tế

✔️ **Tạo header cố định**  
✔️ **Nút gọi điện cố định góc dưới**  
✔️ **Tạo quảng cáo dính màn hình**

📌 **Lưu ý:** **position: fixed;** có thể gây lỗi khi kết hợp với **transform** hoặc **overflow: hidden;.**

🚀 **Kết luận:** **fixed rất hữu ích để giữ các phần tử quan trọng luôn hiển thị!** 🚀

# CSS position: sticky;

## 1️. position: sticky; là gì?

📌 **position: sticky;** kết hợp giữa **relative và fixed**:  
✔ **Ban đầu hoạt động như relative** (theo vị trí ban đầu).  
✔ Khi cuộn trang đến một **mốc cố định**, phần tử **dính lại** như **fixed**.

## 2️. Cách hoạt động

✅ **Cần ít nhất một giá trị top, bottom, left, right** để biết khi nào "dính".  
✅ **Không rời khỏi cha có overflow: hidden; hoặc overflow: auto;**.  
✅ **Vẫn chiếm không gian trong bố cục**.

## 3️. Ví dụ

### a. Ví dụ 1: Thanh tiêu đề "dính" khi cuộn xuống

css

  .header {

    margin-top: 50px;

    position: sticky;

    top: 30px; /\* Dính lại khi chạm mép trên \*/

    background-color: yellow;

    padding: 10px;

    font-size: 20px;

  }

html

<div class="header">Tôi là header dính</div>

<p>Cuộn xuống để thấy hiệu ứng...</p>

✅ **Kết quả:** Thanh tiêu đề sẽ **giữ nguyên trên cùng** khi cuộn đến.

### b. Ví dụ 2: Cột điều hướng bên trái "dính" khi cuộn

css

  .sidebar {

    position: sticky;

    top: 20px; /\* Dính khi cuộn đến 20px từ trên \*/

    width: 200px;

    height: 300px;

    background-color: lightblue;

  }

html

<div class="sidebar">Menu dính khi cuộn</div>

✅ **Kết quả:** Cột menu **cố định khi cuộn đến một mức nhất định**.

## 4️. So sánh với relative, absolute, fixed

| **Giá trị** | **Mô tả** |
| --- | --- |
| relative | Dịch chuyển theo vị trí gốc của nó. |
| absolute | Dịch chuyển theo phần tử cha có relative. |
| fixed | Cố định trên màn hình, không bị cuộn. |
| sticky | Dính khi cuộn đến điểm xác định. |

## 5️. Ứng dụng sticky thực tế

✔ **Thanh điều hướng luôn hiển thị khi cuộn**  
✔ **Tiêu đề cột trong bảng "dính" khi cuộn**  
✔ **Nút quay lại đầu trang dính khi cần thiết**

📌 **Lưu ý**: **position: sticky;** không hoạt động nếu cha của nó có **overflow: hidden;.**

🚀 **Kết luận:** **Rất hữu ích để giữ các phần tử quan trọng hiển thị mà không chiếm không gian như fixed!** 🚀Top of Form

# Giới thiệu về Flexbox trong CSS

**📌 Flexbox** (**Flexible Box Layout**) là một kỹ thuật trong CSS giúp bố cục trang web trở nên linh hoạt, dễ dàng sắp xếp, căn chỉnh và phân phối không gian giữa các phần tử trong container mà không cần dùng đến float hoặc positioning.

## 1. Cấu trúc cơ bản của Flexbox

Flexbox bao gồm **2 thành phần chính**:

* **Flex Container** (Phần tử cha)
* **Flex Items** (Các phần tử con)

A diagram of a diagram

AI-generated content may be incorrect.

**Ví dụ cơ bản:**

css

.container {

    display: flex;  /\* Kích hoạt Flexbox \*/

    border: 1px solid black;

    justify-content: space-between; /\* Các phần tử con sẽ được căn cách nhau một cách đều \*/

}

Html

<div class="container">

        <div class="item">1</div>

        <div class="item">2</div>

        <div class="item">3</div>

    </div>

## 2. Các thuộc tính của Flexbox

**Thuộc tính của Flex Container**

| **Thuộc tính** | **Ý nghĩa** |
| --- | --- |
| display: flex; | Kích hoạt Flexbox |
| flex-direction | Quyết định hướng sắp xếp của items |
| justify-content | Căn chỉnh items theo trục chính (horizontal/vertical) |
| align-items | Căn chỉnh items theo trục phụ |
| align-content | Căn chỉnh khi có nhiều dòng Flex |

### a. flex-direction (Xác định hướng của trục chính)

| **Giá trị** | **Mô tả** |
| --- | --- |
| row (default) | Các phần tử nằm ngang từ trái sang phải |
| row-reverse | Các phần tử nằm ngang từ phải sang trái |
| column | Các phần tử xếp dọc từ trên xuống dưới |
| column-reverse | Các phần tử xếp dọc từ dưới lên trên |

**Ví dụ:**

css

.container {

    display: flex;  /\* Kích hoạt Flexbox \*/

    border: 1px solid black;

    justify-content: space-between; /\* Các phần tử con sẽ được căn cách nhau một cách đều \*/

    flex-direction: row; /\* Hoặc column, row-reverse, column-reverse \*/

}

### b. justify-content (Căn chỉnh theo trục chính)

| **Giá trị** | **Mô tả** |
| --- | --- |
| flex-start | Căn về bên trái (mặc định) |
| flex-end | Căn về bên phải |
| center | Căn giữa |
| space-between | Các phần tử có khoảng cách đều, phần tử đầu và cuối sát lề |
| space-around | Các phần tử có khoảng cách đều nhau (bao gồm khoảng cách hai bên) |
| space-evenly | Các phần tử cách đều nhau (bao gồm cả khoảng cách với lề) |

**Ví dụ:**

css

.container {

    display: flex;  /\* Kích hoạt Flexbox \*/

    border: 1px solid black;

    justify-content: space-between; /\* Các phần tử con sẽ được căn cách nhau một cách đều \*/

}

### c. align-items (Căn chỉnh theo trục phụ)

| **Giá trị** | **Mô tả** |
| --- | --- |
| stretch (default) | Kéo dài phần tử theo trục phụ |
| flex-start | Căn lên trên |
| flex-end | Căn xuống dưới |
| center | Căn giữa theo trục phụ |
| baseline | Căn theo dòng chữ gốc (text baseline) |

**Ví dụ:**

css

.container {

    display: flex;  /\* Kích hoạt Flexbox \*/

    border: 1px solid black;

    height: 500px;

    justify-content: space-between; /\* Các phần tử con sẽ được căn cách nhau một cách đều \*/

    align-items: center; /\* Căn giữa theo trục phụ \*/

}

## 3. Thuộc tính của Flex Items

| **Thuộc tính** | **Ý nghĩa** |
| --- | --- |
| flex-grow | Xác định độ mở rộng của phần tử |
| flex-shrink | Xác định độ thu nhỏ của phần tử |
| flex-basis | Xác định kích thước ban đầu |
| flex | Viết tắt của 3 thuộc tính trên |
| align-self | Căn chỉnh từng item riêng biệt |

### a. flex-grow (Mở rộng phần tử)

* Xác định mức độ mở rộng của một phần tử so với phần còn lại.

**Ví dụ:**

Css

.item:nth-child(1) {

    flex-grow: 1; /\* Phần tử này chiếm nhiều không gian hơn \*/

}

.item:nth-child(2) {

    flex-grow: 2; /\* Gấp đôi phần tử 1 \*/

}

### ***b***. flex-shrink (Thu nhỏ phần tử)

* Xác định mức độ thu nhỏ của một phần tử khi không đủ không gian.

**Ví dụ:**

Css

.item {

    flex-shrink: 1; /\* Mặc định là 1 \*/

}

.item.large {

    flex-shrink: 0; /\* Không cho phép thu nhỏ \*/

}

### c. flex-basis (Kích thước ban đầu)

* Xác định kích thước mặc định của phần tử trước khi áp dụng **flex-grow** hoặc **flex-shrink**.

**Ví dụ:**

Css

.item {

    flex-basis: 100px; /\* Mỗi phần tử sẽ có kích thước ban đầu là 100px \*/

}

### d. flex (Viết tắt của flex-grow, flex-shrink, flex-basis)

**Cú pháp:**

css

.item {

    flex: 1 1 auto; /\* flex-grow | flex-shrink | flex-basis \*/

  }

**Ví dụ:**

css

.item {

    flex: 1; /\* Tương đương với flex: 1 1 0% \*/

}

### e. align-self (Căn chỉnh riêng từng item)

* Giúp một phần tử có thể có cách căn chỉnh khác với các phần tử còn lại.

**Ví dụ:**

css

.item:nth-child(2) {

    align-self: flex-end; /\* Chỉ phần tử thứ 2 nằm dưới cùng \*/

  }

## 4. Ví dụ thực tế - Thanh điều hướng với Flexbox

html

<nav class="navbar">

        <div class="logo">Logo</div>

        <ul class="menu">

          <li><a href="#">Home</a></li>

          <li><a href="#">About</a></li>

          <li><a href="#">Services</a></li>

          <li><a href="#">Contact</a></li>

        </ul>

</nav>

css

.navbar {

    display: flex;

    justify-content: space-between;

    align-items: center;

    background: #333;

    padding: 10px 20px;

    color: white;

  }

  .menu {

    display: flex;

    list-style: none;

  }

  .menu li {

    margin: 0 10px;

  }

## 5. Bổ sung flex-wrap trong CSS

* **flex-wrap** kiểm soát việc các phần tử con có xuống dòng hay không khi không đủ không gian.

**Giá trị chính:**

* **nowrap** (mặc định) → Các phần tử nằm trên một dòng, có thể bị thu nhỏ.
* **wrap** → Các phần tử tự động xuống dòng khi không đủ chỗ.
* **wrap-reverse** → Xuống dòng nhưng theo hướng ngược lại.

📌 **Ứng dụng:** Giúp bố cục linh hoạt hơn, đặc biệt trong thiết kế **responsive**. 🚀

**Khi nào sử dụng flex-wrap?**

✅ Khi bạn muốn **đảm bảo phần tử không bị thu nhỏ** quá mức trên màn hình nhỏ.  
✅ Khi bạn **thiết kế giao diện responsive** và muốn các phần tử tự động xuống dòng.  
✅ Khi bạn muốn **bố cục linh hoạt**, tránh việc các phần tử bị co dồn vào nhau.

## 🚀 Tóm tắt nhanh về Flexbox

| **Thuộc tính** | **Ý nghĩa** |
| --- | --- |
| display: flex; | Kích hoạt Flexbox |
| flex-direction | Chọn hướng sắp xếp (row, column, ...) |
| justify-content | Căn chỉnh theo trục chính (center, space-between, ...) |
| align-items | Căn chỉnh theo trục phụ (center, flex-start, ...) |
| align-content | Căn chỉnh nhiều dòng flex |
| flex-grow | Mở rộng phần tử theo tỉ lệ |
| flex-shrink | Thu nhỏ phần tử khi không đủ không gian |
| flex-basis | Kích thước ban đầu |
| flex | Viết tắt của 3 thuộc tính trên |
| align-self | Căn chỉnh riêng từng item |
| flex-wrap | Các phần tử tự động xuống dòng khi không đủ chỗ |

## 6. Học Flexbox qua ví dụ

### a. Code mẫu sử dụng trong video

* **Chưa có flexbox:** <https://codepen.io/ng-ngc-sn-the-bashful/pen/JjdWEaq>
* **Có flexbox:** <https://codepen.io/ng-ngc-sn-the-bashful/pen/RwPbzKr>
* **Trang web tham khảo**: [https://codepen.io/enxaneta/full/adLPwv/](https://codepen.io/enxaneta/full/adLPwv/" \t "_blank)

### b. Chơi game

Nếu muốn rèn luyện thêm và nắm chắc hơn về flexbox thì các bạn có thể tham khảo 3 games sau nhé:

1. <https://codingfantasy.com/games/flexboxadventure/play>
2. <https://flexboxfroggy.com/>
3. <http://www.flexboxdefense.com/>

Bottom of Form

# BEM là gì?

**BEM** (Block - Element - Modifier) là một phương pháp đặt tên class trong CSS giúp mã nguồn dễ đọc, dễ bảo trì và mở rộng.

## Cấu trúc BEM:

1. **Block (Khối)** – Phần tử độc lập có ý nghĩa riêng.  
   📌 Ví dụ: .button, .card, .menu
2. **Element (Phần tử)** – Thành phần con của Block, không thể tồn tại độc lập.  
   📌 Ví dụ: .menu\_\_item, .card\_\_title, .button\_\_icon
3. **Modifier (Biến thể)** – Dùng để thay đổi trạng thái hoặc kiểu dáng của Block/Element.  
   📌 Ví dụ: .button--primary, .menu\_\_item—active

## Ví dụ BEM trong CSS:

css

/\* Block \*/

.button {

    padding: 10px 20px;

    background-color: blue;

}

/\* Element \*/

.button\_\_icon {

    margin-right: 5px;

}

/\* Modifier \*/

.button--primary {

    background-color: red;

}

html

<button class="button button--primary">

    <span class="button\_\_icon">🔍</span> Search

</button>

Lợi ích của BEM:  
✅ Dễ đọc, dễ bảo trì  
✅ Tránh xung đột CSS  
✅ Hỗ trợ tái sử dụng code 🚀

# Đặt tên class khi Block lồng nhau trong BEM

Trong BEM, khi một **Block** chứa một **Block** khác, ta nên giữ chúng **độc lập** thay vì sử dụng dấu \_\_ (vì \_\_ chỉ dành cho **Element**).

## Cách đặt tên đúng:

Mỗi **Block** vẫn giữ nguyên quy tắc đặt tên riêng, tránh việc tạo class quá dài hoặc phụ thuộc vào Block cha.

### Ví dụ 1: Menu chứa Button

html

<nav class="menu">

<button class="button button--primary">Login</button>

</nav>

Css

.menu {

background-color: #f5f5f5;

padding: 10px;

}

.button {

padding: 8px 16px;

border: none;

}

.button--primary {

background-color: blue;

color: white;

}

👉 **Cách này đúng**, vì .button là một Block riêng, không phải Element của .menu.

### Ví dụ 2: Card chứa Header và Content

html

<div class="card">

<div class="card\_\_header">

<h2 class="card\_\_title">Title</h2>

</div>

<div class="card\_\_content">

<p>Lorem ipsum dolor sit amet...</p>

</div>

</div>

👉 **Cách này đúng**, vì .card\_\_header và .card\_\_content là **Element** của .card, nhưng nếu bên trong có một Block khác (ví dụ .button), thì giữ nó độc lập.

## Lưu ý sai lầm cần tránh ❌

html

<div class="menu">

<button class="menu\_\_button menu\_\_button--primary">Login</button>

</div>

❌ **Sai**: .menu\_\_button không phải Element của .menu, nó nên là một Block riêng.

✅ **Đúng**:

html

<div class="menu">

<button class="button button--primary">Login</button>

</div>

## Tổng kết

✔ **Block độc lập**, không bị phụ thuộc vào Block cha.  
✔ **Element chỉ thuộc về Block**, không thể là một Block khác.  
✔ **Modifier chỉ thay đổi trạng thái của Block hoặc Element.** 🚀