**CSS**

Dùng để tạo kiểu hiển thị cho các phần tử trong html như màu, chữ, hoạt ảnh,...

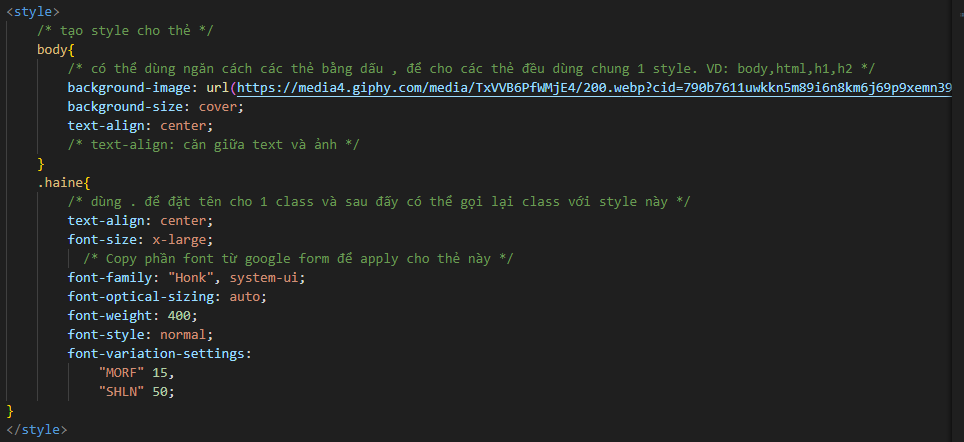
# Có 3 cách sử dụng CSS

## Inline

Lồng ngay trong dòng của các phần tử trong dòng đó, thêm đặc tính style.



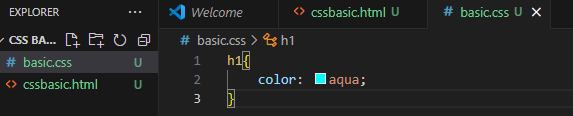
## Internal

Dùng ở ngay đầu trước khi code, viết trong cặp thẻ style và quy định cho tất cả các thẻ có cùng tên.

## External

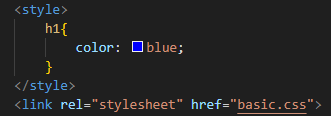
Viết ở 1 file ngoài có đuôi .css, thực hiện kết nối với file html bằng thẻ “link” (đặt ở đầu hoặc trong cặp thẻ <head></head đều được> và với 1 file CSS có thể sử dụng cho nhiều file html.





Độ ưu tiên: Inline sẽ có độ ưu tiên cao nhất, sau đó sẽ theo thứ tự xuất hiện.

VD:

 Nếu link external xuất hiện sau cặp thẻ style, sẽ ưu tiên cặp thẻ style được kích hoạt.

# Bộ chọn CSS (Selecter)

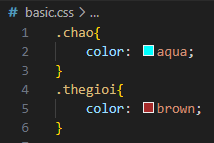
Bộ chọn giúp ta có thể chọn và thao tác chỉnh sửa các phần tử theo ý muốn. Có 3 bộ chọn:

## Chọn theo class

Định nghĩa 1 class bằng phần tử class



Trong CSS, gọi class muốn thao tác bằng dấu “.” sau đó là tên class



## Chọn theo id

Cách sử dụng gần như tương tự theo class, nhưng thay dấu “.” bằng dấu “#”



Trong CSS, gọi id muốn thao tác bằng “#” sau đó là tên id

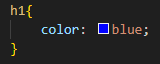


Tuy nhiên id không thông dụng bằng class, id phục vụ chủ yếu cho các trường hợp đặc biệt, id mang tính duy nhất chứ ko chỉ để nhóm class phục vụ cho css. VD:



Với ví dụ trên, khi id được gán vào và cặp thẻ label được sử dụng cho id đó thì sẽ giúp ta ấn vào không chỉ trong phần text mà ấn cả vào **tên người dùng:** cũng sẽ trỏ về phần trong text. Chung quy sẽ giúp cải thiện cho thao tác người dùng vì id là để đại diện cho phần trong text đó.

## Chọn theo tên thẻ

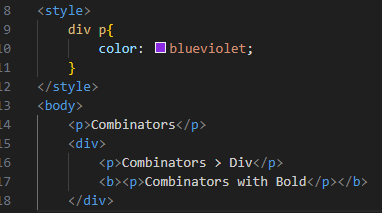
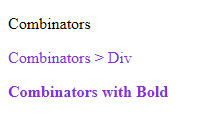
 phân biệt bằng cách ko có định dạng dấu “.” hoặc dấu “#”. Cách sử dụng tương tự.

# CSS Combinators (Bộ chọn kết hợp)

Trình kết hợp CSS là các ký tự hoặc cặp ký tự được sử dụng để chọn các phần tử HTML dựa trên mối quan hệ vị trí giữa chúng trong cây phần tử. Dưới đây là một số trình kết hợp CSS phổ biến:

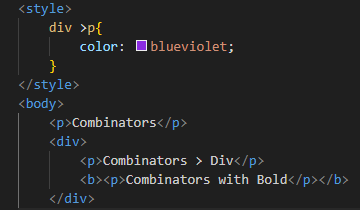
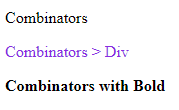
## Trình kết hợp dấu cách (Space combinator):

* + Được biểu diễn bằng dấu cách (khoảng trắng).
  + Sử dụng để chọn các phần tử con của một phần tử khác.
  + Ví dụ: **div p** sẽ chọn tất cả các phần tử **<p>** là con của một phần tử **<div>**.



Với việc kết hợp như trên, bộ chọn CSS sẽ áp dụng cho các phần tử thẻ p và thẻ p phải trong thẻ div, BẤT KỂ thẻ p có là thẻ con của thẻ khác (ở đây là thẻ b) trong thẻ cha div

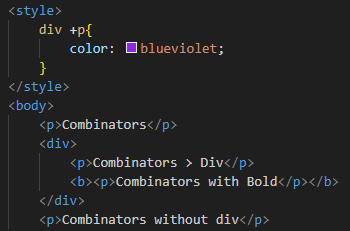
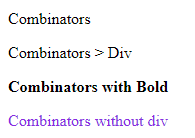
## Trình kết hợp dấu **>** (Child combinator):

* + Được biểu diễn bằng dấu **>**.
  + Sử dụng để chọn các phần tử con trực tiếp của một phần tử khác.
  + Ví dụ: **div >p** sẽ chọn tất cả các phần tử **<p>** là con trực tiếp của một phần tử **<div>**.
  + 

Với style >p, bộ chọn CSS sẽ chỉ áp dụng với các thẻ p là tập con của thẻ div, nhưng nếu thẻ p nằm trong 1 thẻ khác (ở đây là thẻ b) và cả 2 thẻ đều nằm trong thẻ div thì thẻ p nằm trong thẻ b sẽ ko đc áp dụng.

## Trình kết hợp dấu **+** (Adjacent sibling combinator):

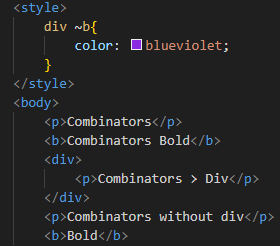
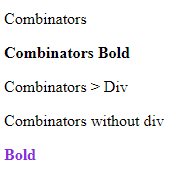
* + Được biểu diễn bằng dấu **+**.
  + Sử dụng để chọn phần tử anh em ngay sau một phần tử khác.
  + Ví dụ: **div +p** sẽ chọn phần tử **<p>** ngay sau một phần tử **<div>**.



Với style +p, bộ chọn CSS sẽ áp dụng cho thẻ p ngay sau thẻ /div, nếu giữa thẻ /div và thẻ p là 1 thẻ khác thì trình kết hợp này sẽ bị vô hiệu

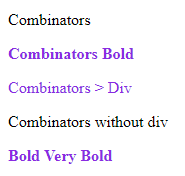
## Trình kết hợp dấu **~** (General sibling combinator):

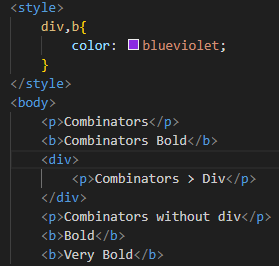
* + Được biểu diễn bằng dấu **~**.
  + Sử dụng để chọn tất cả các phần tử anh em cùng cấp (trong cùng một cấp cha) của một phần tử khác.
  + Ví dụ: **div ~p** sẽ chọn tất cả các phần tử **<p>** là anh em của một phần tử **<div>** trong cùng một cấp.



Với Style ~b, bộ chọn CSS sẽ áp dụng cho tất cả thẻ b bắt đầu từ sau thẻ /div

## Trình kết hợp dấu “,”





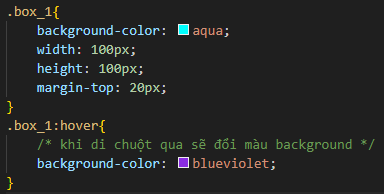
Với style ,b bộ chọn CSS sẽ áp dụng cho tất cả các thẻ b và div, ko phân biệt vị trí

# Effect cơ bản

## Các effect với đường link

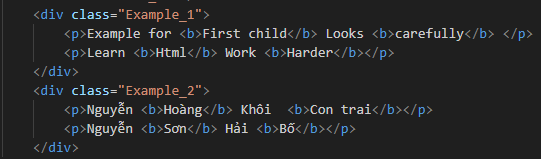
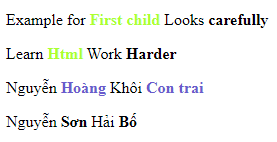


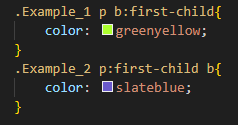
## Các effect với thẻ



## First Child

"first child" (con đầu tiên) là một pseudo-class (lớp giả) được sử dụng để chọn phần tử con đầu tiên của một phần tử cha. Khi sử dụng "first-child", bạn có thể áp dụng các quy tắc CSS cho phần tử con đầu tiên trong danh sách các phần tử con của phần tử cha đó.



Với Example\_1, phần tử b đầu tiên của mỗi phần tử p sẽ được đổi màu xanh lá, trong khi các phần tử b tiếp theo của mỗi phần tử p sẽ không được bôi màu. 2 chữ “First child” và “Html” chính là first child của mỗi phần tử p.

Với Example\_2, phần tử p(Nguyễn Hoàng Khôi Con trai) ở dòng đầu là first child, và chỉ đổi màu với các phần tử b(Hoàng Con trai) của first child đó

## Pseudo Elements

"Pseudo-element" trong CSS là một cách để chọn và tạo ra các phần tử ảo trong HTML, và sau đó áp dụng các quy tắc CSS cho các phần tử này. Pseudo-elements bắt đầu bằng hai dấu hai chấm **::**, theo sau là tên của pseudo-element.

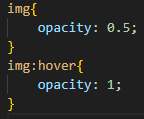
Các pseudo-element phổ biến nhất trong CSS bao gồm:

1. **::before**: Tạo ra một phần tử ảo được chèn vào trước nội dung của phần tử được chọn.
2. **::after**: Tạo ra một phần tử ảo được chèn vào sau nội dung của phần tử được chọn.
3. **::first-line**: Chọn và tạo ra một phần tử ảo đại diện cho dòng đầu tiên của một phần tử văn bản.
4. **::first-letter**: Chọn và tạo ra một phần tử ảo đại diện cho chữ cái đầu tiên của một phần tử văn bản.

## Ocapacity/Transparent

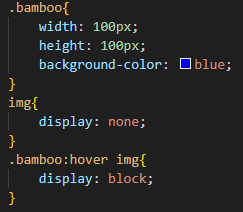
Dùng để ẩn/ hiện text hoặc hình ảnh, có thể kết hợp với effect hover hoặc nhiều effect khác áp dụng trong thực tế

### Ocapacity



để điều chỉnh hình ảnh đậm nhạt sau khi di chuột qua

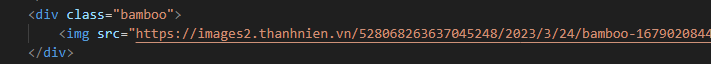
### Display, visibility



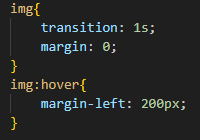
Để xác định 1 khối vuông màu xanh, sau khi di chuột qua khối vuông đó sẽ hiển thị ra hình ảnh

Lưu ý: Với Display bắt buộc phải xác định kích thước cho block trong đó vì thuộc tính display:none sẽ xóa cả ô trống của ảnh

.bamboo:hover img (để sử dụng chức năng hover cho thẻ hoặc class nằm trong thẻ cha (ở đây thẻ cha là .bamboo)



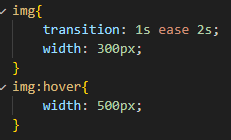
## Transition



Dùng để tạo hiệu ứng chuyển đổi bắt mắt, có thể áp dụng với

nhiều thuộc tính như width, height, color, opacity, margin,…

Để thực hiện được, phải có 2 giá trị thuộc tính giống nhau có thay đổi ở 2 CSS, ví dụ width tăng lên, đổi màu, ocapicity mờ đi



Với cấu tạo 1 transition, vị trí đầu tiên là **khoảng thời gian để hiệu ứng diễn ra,** thứ 2 là **hàm thời gian**, thứ 3 là **khoảng thời gian delay** để hiệu ứng bắt đầu hoạt động



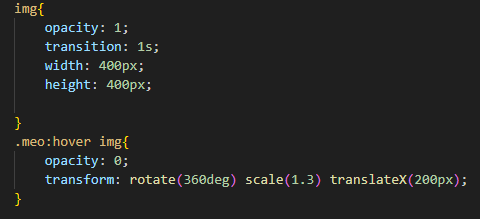
Có thể điều chỉnh từng thuộc tính bị ảnh hưởng trong 1 dòng lệnh, ngăn cách nhau bởi dấu “,”

## Transform

Dùng để thay đổi vị trí, hình dạng, kích thước của 1 vật thể.

Có thể dùng kết hợp như sau:

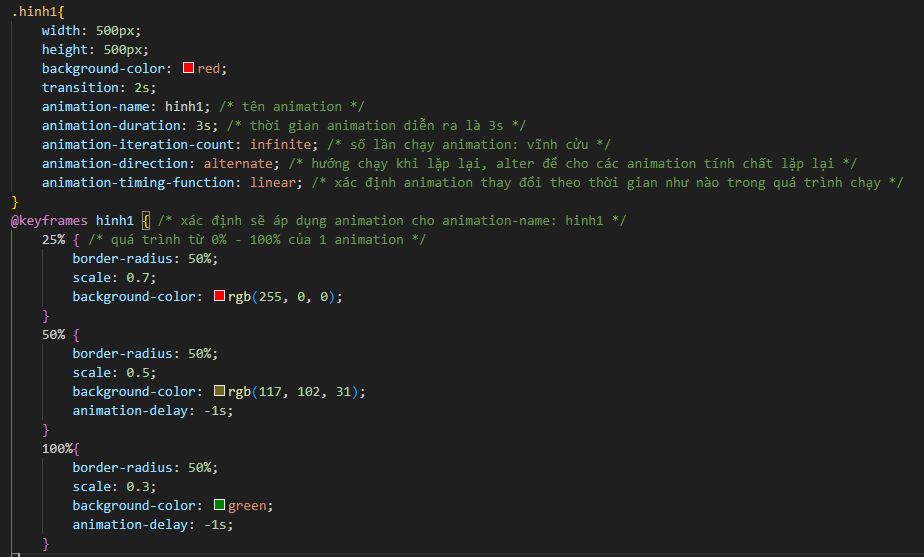
1. Translate (dịch chuyển):
   * translateX(value **pixels**): Dịch chuyển phần tử theo trục X.
   * translateY(value): Dịch chuyển phần tử theo trục Y.
   * translate(valueX, valueY): Dịch chuyển phần tử theo cả hai trục X và Y.
2. Rotate (xoay):
   * rotate(angle **degree**): Xoay phần tử theo góc được chỉ định. Góc được tính bằng độ và có thể là giá trị âm.
3. Scale (tỉ lệ):
   * scaleX(value **times(1.5)**): Thay đổi kích thước của phần tử theo trục X.
   * scaleY(value): Thay đổi kích thước của phần tử theo trục Y.
   * scale(valueX, valueY): Thay đổi kích thước của phần tử theo cả hai trục X và Y.
4. Skew (nghiêng):
   * skewX(angle **degree**): Nghiêng phần tử theo trục X.
   * skewY(angle): Nghiêng phần tử theo trục Y.
   * skew(angleX, angleY): Nghiêng phần tử theo cả hai trục X và Y.



Lưu ý:

Với những hoạt ảnh 3D, cần thêm thuộc tính perspective(px) ở thẻ/ classs cha để có thể sử dụng được.

## Animation

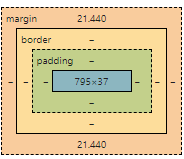
là một cách để tạo ra các hiệu ứng chuyển động và thay đổi trạng thái của các phần tử trên trang web. Animation cho phép bạn định nghĩa các keyframes để mô tả các trạng thái khác nhau của phần tử trong quá trình thời gian và áp dụng chúng cho một hoặc nhiều thuộc tính CSS.

Trong đó:

* **animation-name**: Tên của keyframes được sử dụng cho animation.
* **animation-duration**: Thời gian mà animation diễn ra, có thể là giây hoặc mili giây.
* **animation-timing-function**: Hàm thời gian (timing function) xác định cách animation thay đổi theo thời gian.
* **animation-delay**: Thời gian chờ trước khi animation bắt đầu.
* **animation-iteration-count**: Số lần lặp lại của animation, hoặc giá trị "infinite" để lặp vô hạn.
* **animation-direction**: Hướng của animation khi lặp lại.
* **animation-fill-mode**: Xác định trạng thái của phần tử trước và sau animation.
* **animation-play-state**: Xác định trạng thái phát hoặc tạm dừng của animation.

# Box model

Là hình dạng sẵn có của các thẻ html, mỗi 1 thẻ html đều có 1 box model.



Mỗi 1 box model đều có đầy đủ các yếu tố:

## Margin

khoảng cách từ viên đến các thẻ khác

Có thể tùy chỉnh margin qua các thẻ thuộc tính:

1. **margin-top**: Xác định khoảng cách giữa phần tử và phần tử phía trên.
2. **margin-right**: Xác định khoảng cách giữa phần tử và phần tử phía bên phải.
3. **margin-bottom**: Xác định khoảng cách giữa phần tử và phần tử phía dưới.
4. **margin-left**: Xác định khoảng cách giữa phần tử và phần tử phía bên trái.
5. **margin :** Xác định khoảng cách của cả 4 cạnh.

Có thể căn chỉnh bằng giá trị auto. Khi đó, margin sẽ được căn giữa cả 4 chiều so với phần tử cha

Hoặc sử dụng margin: 20px 10px để căn chỉnh theo cặp trên dưới, trái phải

## Border

Khoảng cách từ padding đến margin

Có thể tùy chỉnh border qua các thẻ thuộc tính:

1. **border-width**: Xác định độ dày của border.
2. **border-style**: Xác định kiểu của border, như solid, dashed, dotted, và nhiều hơn nữa.
3. **border-color**: Xác định màu sắc của border.
4. **border-radius**: Xác định độ cong của border, tạo ra góc bo tròn.
5. **border**: Thuộc tính rút gọn để thiết lập border-width, border-style và border-color cùng một lúc.

Các chức năng top, bot, left, right sử dụng như margin.

## Padding

Khoảng cách từ nội dung đến viền

Thẻ padding có cách điều chỉnh y như margin, thay tên thuộc tính thành padding-top/bottom/left/right

## Phần nội dung (content)

nội dung, ví dụ như hình ảnh, văn bản,…

Trong CSS, không có một thuộc tính cụ thể nào được sử dụng để tùy chỉnh nội dung (content) của một phần tử HTML. Nội dung của một phần tử HTML thường được quy định bằng các thẻ HTML như **<div>**, **<p>**, **<span>**, **<h1>** đối với văn bản, hoặc **<img>**, **<video>**, **<audio>** đối với các phần tử đa phương tiện như hình ảnh, video, và âm thanh.

Tuy nhiên, bạn có thể sử dụng các thuộc tính CSS khác để tùy chỉnh vị trí, kích thước và hiển thị của nội dung bên trong một phần tử. Dưới đây là một số thuộc tính CSS thường được sử dụng để tùy chỉnh nội dung:

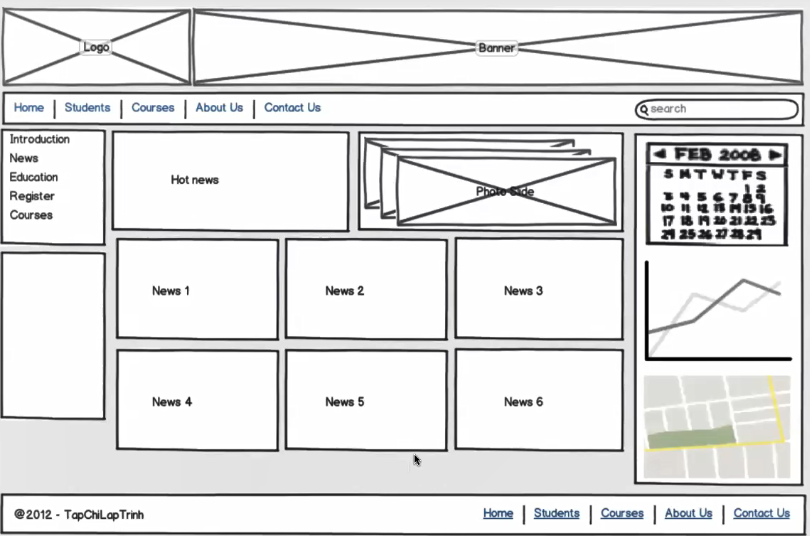
1. **font-size**: Xác định kích thước của văn bản.
2. **font-family**: Xác định font chữ được sử dụng cho văn bản.
3. **color**: Xác định màu sắc của văn bản.
4. **text-align**: Xác định căn chỉnh của văn bản (left, center, right).
5. **line-height**: Xác định độ cao của dòng văn bản.
6. **background-color**: Xác định màu nền của phần tử.

Với các phần tử đa phương tiện như hình ảnh, video và âm thanh, bạn có thể sử dụng các thuộc tính như **width**, **height**, và **object-fit** để tùy chỉnh kích thước và hiển thị của chúng.

# CSS Layout

Là giao diện cơ bản của một trang web, bao gồm có:

Header, Navtive, Main, Footer



Để chia được các bố cục, ta tận dụng các thuộc tính trong CSS như layout hoặc flexbox để chia cho đẹp

Để hình dung rõ hơn về các CSS layout, đọc thêm tại [đây](CSS/CSS%20effect/Facebook/fb.html)

## Float

Trong HTML và CSS, thuộc tính **float** được sử dụng để di chuyển một phần tử (thường là phần tử block) sang trái hoặc phải của phần tử cha của nó, và cho phép các phần tử khác trôi lấp xung quanh nó.

Dưới đây là cách sử dụng thuộc tính **float**:

<div style="float: left;">Phần tử được di chuyển sang trái</div>

hoặc

<div style="float: right;">Phần tử được di chuyển sang phải</div>

Khi bạn áp dụng thuộc tính **float** cho một phần tử, phần tử đó sẽ không chiếm toàn bộ không gian của phần tử cha mà nó đang nằm trong. Thay vào đó, các phần tử khác sẽ trôi lấp xung quanh phần tử đó.

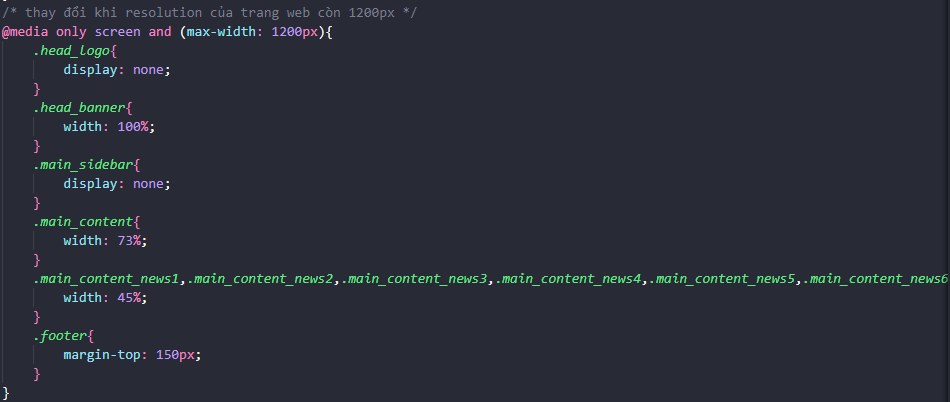
Lưu ý rằng việc sử dụng **float** có thể gây ra một số vấn đề liên quan đến bố cục của trang web, đặc biệt là khi không quản lý được các phần tử chồng lên nhau. Để xử lý các vấn đề này, thường được khuyến khích sử dụng các kỹ thuật bố cục hiện đại như Flexbox hoặc Grid Layout.

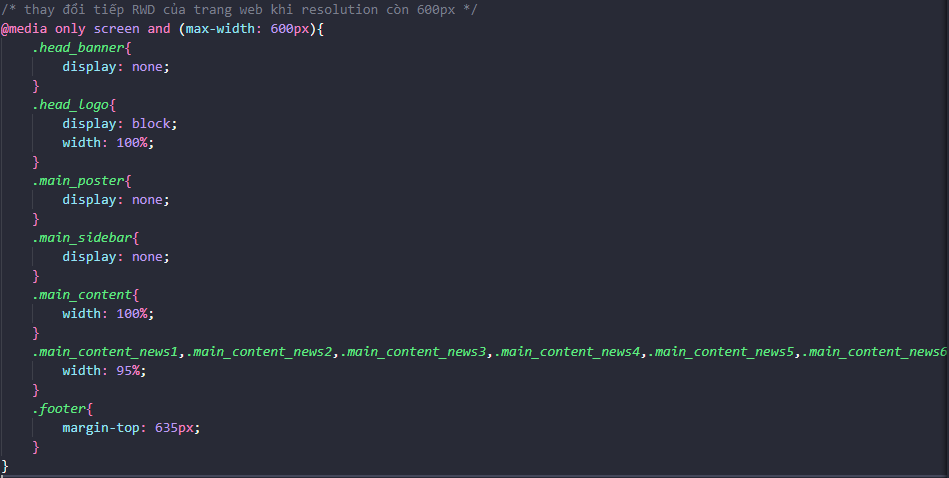
# RWD (Responsive Web Design)

Thiết kế web đáp ứng: Đây là một phương pháp thiết kế và phát triển trang web nhằm cung cấp trải nghiệm người dùng tốt trên nhiều thiết bị và kích thước màn hình khác nhau, từ máy tính để bàn đến điện thoại di động.

Các kỹ thuật về RWD:

## Media queries

Sử dụng media queries để thay đổi kiểu hiển thị của trang web dựa trên các đặc điểm của thiết bị, như kích thước màn hình, chiều rộng, hoặc loại thiết bị. Điều này cho phép bạn điều chỉnh kích thước, bố cục, font chữ và các thuộc tính khác của trang web để phù hợp với các điều kiện hiển thị khác nhau.



## Grid column (col-)

là một kỹ thuật thiết kế trang web linh hoạt dựa trên việc sử dụng đơn vị đo lường linh hoạt như phần trăm thay vì các đơn vị cố định như pixel để xác định kích thước của các phần tử trên trang web.

Cụ thể, trong fluid grids, các cột và hàng của lưới được thiết kế để tự động điều chỉnh kích thước của chúng dựa trên kích thước của trình duyệt hoặc thiết bị mà trang web đang hiển thị trên đó. Điều này cho phép trang web phản ánh tỉ lệ khung hình của màn hình và hiển thị một cách linh hoạt trên nhiều thiết bị và kích thước màn hình khác nhau.

/\* xây dựng grid với 12 cột, mỗi cột có động rộng là 100% / 12 = 8.33% \*/

/\* cột col-1 là 8.33%, col-2 là 8.33% + 8.33% = 16.66%, mỗi cột sẽ + thêm 8.33% cho đến 100% \*/

*.col-1*{width: 8.33%;}

*.col-2*{width: 16.66%;}

*.col-3*{width: 25%;}

*.col-4*{width: 33.33%;}

*.col-5*{width: 41.66%;}

*.col-6*{width: 50%;}

*.col-7*{width: 58.33%;}

*.col-8*{width: 66.66%;}

*.col-9*{width: 75%;}

*.col-10*{width: 83.33%;}

*.col-11*{width: 91.66%;}

*.col-12*{width: 100%;}

/\* định nghĩa CSS cho tất cả class có chứa chuỗi "col-" \*/

[*class* \*="col-"] {

    float: left;

    padding: 15px;

    border: 1px solid black;

    box-sizing: border-box;

    text-align: center;

<!-- thêm các “col-(số thứ tự để chọn kích thước muốn hiển thị đã cho trước)“ vào trước các tên class chính để phần tử này sẽ ăn 3 lớp CSS -->

<body>

    <div *class*="head">

        <div *class*="col-3 head\_logo">Logo</div>

        <div *class*="col-9 head\_banner">Banner</div>

    </div>

    <div *class*="col-12 nav">Nav</div>

    <div *class*="col-12 main">

        <div *class*="col-2 main\_sidebar">

            <div *class*="col-12 sidebar\_1">sidebar\_1</div>

            <div *class*="col-12 sidebar\_2">sidebar\_2</div>

        </div>

## Display Grid

CSS Grid Layout là một hệ thống bố cục hai chiều mạnh mẽ được xây dựng trong CSS. Nó cho phép bạn tạo các bố cục phức tạp bằng cách sử dụng các hàng và cột. Dưới đây là một hướng dẫn cơ bản về cách sử dụng CSS Grid.

### 1. Khởi Tạo Grid Container

Để bắt đầu sử dụng Grid, bạn cần chỉ định một phần tử là grid container bằng cách sử dụng thuộc tính **display: grid**.

*.main* {

    display: grid;

    /\* tạo hiển thị dạng lưới \*/

}

### 2. Định Nghĩa Cột và Hàng

Bạn có thể định nghĩa số lượng cột và hàng trong grid container bằng cách sử dụng các thuộc tính **grid-template-columns** và **grid-template-rows**.

    grid-template-columns: auto auto auto;

    /\* chiều ngang của các phần tử, số lượng cột tương ứng với số đơn vị \*/

    grid-template-rows: 100px 100px 100px;

    /\* chiều cao của các phần tử trong các hàng, số lượng hàng tương ứng với số đơn vị\*/

Bạn cũng có thể sử dụng các giá trị linh hoạt như phần trăm hoặc **fr** (fractional unit):

*.main* {

display: grid;

grid-template-columns: 1fr 2fr 1fr; /\* 3 cột, tỷ lệ 1:2:1 \*/

grid-template-rows: 200px auto; /\* Hàng đầu tiên 200px, hàng thứ hai tự động \*/

}

### 3. Đặt Các Phần Tử Vào Ô

Mỗi phần tử con bên trong grid container sẽ tự động được đặt vào các ô của grid. Bạn có thể sử dụng các thuộc tính như **grid-column** và **grid-row** để chỉ định vị trí cụ thể.

*.sub1* {

    border: 1px solid black;

    grid-column: 1/3; /\* Bắt đầu từ cột 1 và kết thúc trước cột 3 \*/

    grid-row: 1 / 2; /\* Bắt đầu từ hàng 1 và kết thúc trước hàng 2 \*/

}

### 4. Khoảng Cách Giữa Các Ô

Sử dụng grid-gap hoặc gap để tạo khoảng cách giữa các ô.

*.main* {

    gap: 5px;

    /\* khoảng cách giữa các phần tử là 5px \*/

    row-gap: 5px;

    /\* khoảng cách các hàng giữa các phần tử \*/

    column-gap: 10px;

    /\* khoảng cách các cột giữa các phần tử \*/

}

### 5. Điều chỉnh vị trí phần tử

*.main* {

    justify-content: space-between;

    /\* các phần tử cách đều nhau \*/

    justify-content: center;

    /\* đưa các phần tử về giữa theo hàng ngang\*/

    align-content: center;

    /\* đưa các phần tử về giữa theo hàng dọc \*/