**Tự học lập trình**

git remote add origin git@github.com:trandat2509/Savoy-web.git

**Làm việc với git**

1. Cái đặt git
2. Xác định danh tính:

Set your username:  
git config --global user.name "FIRST\_NAME LAST\_NAME"

Set your email address:  
git config --global user.email "MY\_NAME@example.com"

1. Kéo code từ github về thông qua git

Truy cập vao địa chỉ github: click vào code -> sao chép mã ở phần HTTPS

Tạo một thư mục mới để lưu vào -> chuột phải -> git bash here

Ở phần git bash here nhập: git clone “paste phần mới copy từ github về”

Mở file ở thư mục mới tạo đó và làm việc với nó

1. Có 2 cách để quản lý source với github:

* Tạo 1 project ở github sau đó kéo về và bắt đầu code (thường là làm việc theo team và porject đã được leader tạo sẵn).
* Khi tạo một project mới ở github để clone thì làm theo các bước clone code về như ở trên và sau đó làm việc ở file project mới clone về đó.
* Khi làm việc xong muốn đẩy phần mình mới làm lên thì làm theo các bước sau:
* Mở git bash here ở folder mình làm việc
* Nhập: git status để kiểm tra trạng thái (xem thư có bao nhiêu file đưa lên, lúc này các file có màu đỏ)
* Nhập: git add . (thêm tất cả các file)
* Quay lại nhập: git status (lúc này các file đã chuyển thành màu xanh)
* Nhập: git commit -m “(nhập bất kỳ cái gì bạn muốn vào đây)”
* Nhập: git push origin main -> lúc này nó sẽ hiện lên 1 cửa sổ có tên là: Connect to Github -> Sign In with your browser -> chuyển đến một tab mới ở trình duyệt của bạn -> nhấp chuột vào **authorize** để xác nhận danh tính của mình trên github -> nhập password gihub của bạn. Đồng thời lúc này trong git bash nó hiện các thông tin bạn đã đẩy code lên.
* Hoàn thành công việc.
* Tóm tắt lại các bước trên

1. Tạo một gihub repository (folder) – nơi chứa code hoặc nếu muốn đưa code lên project có sẵn thì không cần tạo.
2. Clone project này: git clone link\_your\_repo…
3. Viết code
4. Mở git bash
5. Dùng các câu lệnh:

* git status (kiểm tra trạng thái file, có rất nhiều file trong folder của bạn, nếu hôm nay bạn chỉ chỉnh sửa một file thì nó chỉ hiện một file, bạn chỉnh sửa bao nhiêu file thì nó hiện bấy nhiêu file).
* Git add . (thêm file)
* Git commit -m “your message” (tạo nội dung commit).
* Git push origin main.

1. Hoàn thanh công việc đẩy code lên.

* Code sẵn ở máy tính và sau đó đẩy lên github
* Tạo một project bạn muốn đẩy code lên github

1. Copy dòng lệnh: git remote add origin …. Hiện lên khi bạn tạo xong project.
2. Mở file mà bạn muốn đưa lên github (lúc này trong file đó không có mục .git) -> mở git bash
3. Gõ câu lệnh: git init
4. Gõ: git status để kiểm tra trạng thái
5. Gõ: git add .
6. Gõ: git status
7. Gõ: git commit – m “…”
8. Git remote add origin <paste địa chỉ vừa copy ở bước 1 vào> (bước quan trong nhất)
9. Gõ: git push origin master (lúc này code đã được đẩy lên github).
10. Hoàn thành công việc.

Hai cách trên hoàn toàn khác nhau

1. Sự khác nhau giữa dùng https và ssh

* https và ssh sẽ giúp chúng ta xác minh danh tính

https: cần phải xác thực thông qua usernam và password

ssh: chỉ cần cấu hình, mỗi lần đẩy code thì không cần phải xác thực là ai nữa bởi vì điều đó đã được github thực hiện

* tạo key(cấu hình) cho ssh:

1. vào setting trong github
2. vào phần SSH and GPG keys

* tạo key
* mở git bash và paste dòng lệnh này vào:
* search keyword: create ssh github
* ssh-keygen -t ed25519 -C "trandat25092004@gmail.com"
* enter tất cả
* check xem id ở đâu: cmd -> whoami
* C:\Users\DELL\.ssh
* Chuột phải vào file màu chọn: edit with notepad++
* Copy đoạn text
* Quay lại github vào ssh và chọn new ssh key
* Copy đoạn text vào phần key
* Add ssh key
* Hoàn thành.

1. Mô hình hoạt động của git

* Git add <file1> <file2>, … <fileN>
* Git commit: chú thích cho nội dung công việc chúng ta làm
* -m: viết tắt của từ massage
* Git push: giúp chúng ta đẩy code từ máy tính chúng ta lên github, gitlab, …

1. Làm việc nhóm với git

* Branch: nhánh
* Git pull = git fetch + git merge: kéo về (kéo code của đồng đội khi làm việc teamwork về để chạy). git pull origin main
* Git merge: khi làm việc trên các nhánh riêng lẻ thì phải cần đến merge code
* Git checkout -b name-branch: tạo ra một nhánh
* Cách cho new-branch tạo vào branch-main:
* Click vào **compare & pull request.**: tạo một yêu cầu để gộp vào nhánh main -> create -> merge request -> confirm merge -> hoàn thành.
* Git branch: kiểm tra hiện tại đang có bao nhiêu nhánh (dấu \* thể hiện mình đang ở branch nào)
* Coflict: VD có 1 file nhưng nhiều người sửa nếu đẩy code lên thì sẽ có lỗi -> git pull

1. Thành thạo git với VS Code

* TH 1: Add code lần đầu và đưa thẳng lên git

git init

git add README.md

git commit -m "first commit"

git branch -M main

git@github.com:trandat2509/html.git (đường dẫn lúc tạo project)

git push -u origin main

# Bài 1: Làm quen với html

Cấu trúc tệp HTML, tìm hiêu DOCTYPE, thẻ HTML, thẻ HEAD trong lập trình web.

1. <!DOCTYPE html> là một khai báo DOCTYPE trong html, đây là mã khai báo cho trình duyệt biết rằng bạn đang sử dụng HTML5 – phiên bản mới nhất html hiện tại.
2. Cặp thẻ <html></html>:

* Được sử dụng để định nghĩa phần thân của tài liệu HTML
* Thẻ html có thể kèm thuộc tính “lang” để chỉ ngôn ngữ được viết của website
* Trong một file html chỉ có duy nhất một cặp thẻ <html></html>.

1. Cặp thẻ <head></head>:

* Thẻ head là một phần tử HTML được sử dụng để chứa các thông tin về trang web mà không được hiển thị trực tiếp trên trình duyệt của người dùng
* Các thông tin có trong thẻ head như tiêu đề trang, mô tả trang, từ khóa, thông tin tác giả, các tập tin CSS và JavaScript được liên kết với trang và nhiều thông tin khác.
* Các thông tin trong thẻ head rất quan trọng cho việc tối ưu hóa công cụ tìm kiếm (SEO), vì chúng giúp các công cụ tìm kiếm hiểu được nội dung và ý nghĩa của trang web.

1. Cặp thẻ <body></body>:

* Thẻ body là một phần tử HTML được sử dụng để chứa nội dung chính của trang web, bao gồm các phần tử như văn bản, hình ảnh, đoạn mã JavaScript, các phần tử định dạng v.v…
* Các phần tử bên trong thẻ body được hiển thị trực tiếp trên trình duyệt của người dùng và chúng tạo thành nội dung chính của trang web.
* Thẻ <h> được sử dụng để định nghĩa tiêu đề của trang web
* Thẻ <p> paragraph được sử dụng để chứa các đoạn văn bản
* Thẻ <b> bold - in đậm (dùng strong thay thế thể hiện nội dung quan trọng)
* Thẻ <i> italic – in nghiêng (dùng em-emphasis để nhấn mạnh nội dung)
* Thẻ <u> undeline – gạch chân
* Thẻ <a> anchor – đính kèm đường dẫn
* Thẻ <img> image – hình ảnh
* Thẻ <ul> undered-list danh sách không có thứ tự
* Thẻ <ol> ordered-list danh sách có thứ tự

1. Cặp thẻ <p></p>:

* Để chứa các đoạn văn bản

1. Thẻ <br> - break: phá vỡ hoặc ngắt dòng
2. Thẻ <hr> - horizontal rule: tạo một đường kẻ ngang ngăn cách

Emmet – viết tắt:

* ul>li:
* <ul>
* <li></li>
* </ul>
* ol>li\*5{item}:
* <ol>
* <li>item</li>
* <li>item</li>
* <li>item</li>
* <li>item</li>
* <li>item</li>
* </ol>

1. Thẻ <a></a>: anchor – gắn kèm liên kết, email, phone

* Liên kết số mail: mailto
* Chuyển hướng đến ứng dụng để gửi email
* Một tham số có thể truyền trước cho mailto
* Cc: danh sách email nhận đc bản sao của thư
* Bcc: danh sách email ẩn nhận được bản sao của thư
* Subject: chủ đề của thư
* Body: Nội dung của mail
* ?: dấu phân cách giữa phần mailto và tham số
* &: dấu phân cách tham số
* Liên kết số điện thoại: tel
* Điều hướng đến ứng dụng có thể gọi điện thoại
* Liên kết đường dẫn: Thuộc tính “target” trong thẻ a được sử dụng để chỉ định cách mở liên kết được nhúng trong thẻ đó.
* Một số VD:
* <a href="">Click vào đây</a>
* <a href="#">Click vào đây</a>

Nếu để trống thì khi click vào sẽ tải lại trang

Nếu có dấu # thì sẽ quay lại đầu trang

<a href="https://tek4.vn/" target="">Click vào đây</a>

<a href="https://tek4.vn/" target="\_self">Click vào đây</a>

Khi click vào sẽ mở trang web tại trang hiện tại

<a href="https://tek4.vn/" target="\_blank">Click vào đây</a>

Khi click vào sẽ mở trang web ở tab mới.

1. Thẻ <img> - image: là thẻ tự đóng (hiển thị hình ảnh lên web site)

* Thuộc tính *src* được sử dụng để chỉ định đướng dẫn tới tập tin hình ảnh
* “alt” alternative – thay thế: để
* Hiển thị text khi link ảnh lỗi
* Hỗ trợ đọc màn hình cho người khiếm thị
* Công cụ tìm kiếm thu thập dữ liệu trên trang

1. Thẻ pre – preformatted: dùng để dữ nguyên định dạng văn bản trong trình soạn thảo code. VD: khoảng trắng dư thừa, tab thụt vào đầu dòng.
2. Thẻ <code></code>: là một thẻ có ngữ nghĩa

* Sử dụng khi có một đoạn code, trình duyệt sẽ hiểu đây là một đoạn code
* Thể hiện một khối code độc lập (cần chuyển các ký tự đặc biệt sang HTML Entities)

1. HTML table

* Sử dụng căp thử <table></table> để tạo bảng bên trong có các:
* <tr>: đại diện cho một hàng trong bảng
* <td>: đại diện cho một ô dữ liệu trong bảng
* <th>: định nghĩa một ô tiêu đề trong bảng
* <thead> - table header: định nghĩa phần đầu của bảng, thường chứa các ô tiêu đề (“th”) của các cột trong bảng.
* <tbody> - table body: định nghĩa phần thân của bảng. Thường chứa các thẻ <tr> và các ô dữ liệu <td> tương ứng.
* <tfoot> - table footer: định nghĩa phần cuối của bảng. Thường chứa các ô chú thích, tổng kết hoặc thông tin cuối cùng của bảng
* <colspan>: chỉ định phần tử chiếm bao nhiêu cột.
* <rowspan>: chỉ định phần tử chiếm bao nhiêu hàng

# Bài 2: CSS cơ bản

## Tổng quan CSS

* CSS là ngôn ngữ được sử dụng để tìm và định dạng lại các phần tử được tạo ra bởi ngôn ngữ đánh dấu HTML.
* Nếu html đóng vai trò định dạng các phần tử trên website như việc tạo ra các đoạn văn bản, các tiêu đề, … thì CSS có thể giúp chúng ta thêm style vào các phần tử HMLT đó như đổi bố cục, màu sắc trang, đổi màu chữ, font chữ,..

## Vị trí đặt CSS:

* Inline (CSS nội tuyến): là cách viết mã ngay bên trong phần tử cơ bản HTML

<tag style = “css”>

* Internal (CSS nội bộ)

<style>css</style>

* External (CSS bên ngoài)

<link href = “style.css”>

* Độ ưu tiên: Thằng nào viết sau thằng đó mạnh hơn

## Các thuộc tính cơ bản của CSS

* Opacity: Quy định độ trong suốt của phần tử
  + 0.0 -> 1.0
  + 1% -> 100%
* Display:
* None: Ẩn phần tử khỏi giao diện (xóa cả không gian nó đang chiếm)
* Visibility: Ẩn hoặc hiện element (không nhìn thấy trên giao diện, vẫn chiếm không gian, không tương tác được.)
* Hidden: ẩn
* Visible: hiện

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Opcity: 0 | Display: none | Visibility: hidden |
| Ẩn khỏi giao diện | Có | Có | Có |
| Chiếm không gian | Có | Không | Có |
| Có thể tương tác | Có | Không | Không |
| Ẩn phần tử con | Có | Có | Có |
| Có Tính kế thừa | Không | Không | Có |
| Hiển thị phần tử con | Không | Không | Có |
| Hỗ trợ animations | Có | Không | Có |

### Trường hợp sử dụng opacity, display và visibility

## Favicon

* Các bước lấy favicon của một trang web:
* Ctrl + u: tìm đến nguồn trang
* Ctrl + f: tìm kiếm từ khóa icon
* Tải ảnh về và thay vào lin href trong HTML
* Các bước để chỉnh kích thước favicon ở các thiết bị khác nhau:
* Truy cập trang web: make fovicon online
* Vào đường dẫn sau: <https://www.favicon-generator.org/>
* Kích vào chọn tệp và đưa hình ảnh fovicon vào -> chọn vào creat favicon
* Sau khi tạo xong kích vào dowload để tải bộ file về -> giải nén
* Copy toàn bộ link được hiển thị khi làm bước 3 sau đó đưa vào HTML.
* Bôi đen toàn bộ href sau đó nhấn tổ hợp ctrl + h để thay đổi đường link.
* Còn một hình ảnh không tuân theo href thì chúng ta phải tự sử nó riêng.
* Là icon của trang web bạn tạo
* \* Hộp thoại thay thế: Ctrl + H: Thay thế tất cả những gì bạn bôi đen

## CSS Prefix

* Khái quát: CSS có rất nhiều thuộc tính và không phải thuộc tính nào cũng hỗ trợ tất cả các trình duyệt.
* Khi gặp các thuộc tính không hỗ trợ all trình duyệt thì phải cần thêm nhiều yếu tố prefix để được hỗ trợ.
* Để kiểm tra prefix trình duyệt là gì:
* Tìm kiếm trang web: **vendor prefix css:**
  + - * **Internet Explorer:** -ms-
      * **Chrome:** -webkit-
      * **Safari:** -webkit-
      * **Firefox:**-moz-
      * **iOS:** -webkit-
      * **Android:** -webkit-
      * **Opera:** -o-
      * **Microsoft Edge:** -ms-
* Autoprefix: copy css -> trang web sẽ tự động thêm prefix cho những thuộc tính cần thiết.
* Làm sao để biết thuộc tính mới gặp lần đầu tiên có cần prefix để hỗ trợ:
* Caniuse.com: Trang web hỗ trợ kiểm tra thuộc tính css có cần sử dụng thêm prefix để hỗ trợ đa trình duyệt.

## CSS Selector cơ bản

* 1. Tag name:
* Bộ chọn các tên thẻ mà ta dùng:
* VD: h1 -> h6
  1. ID: Khi chỉ muốn áp dụng CSS cho một phần tử chỉ định nào đó -> dùng id.
     1. ID 1:
* ID là duy nhất, theo quy chuẩn không cho phép đặt trùng ID đã có
* ID, Class không được đặt tên bắt đầu bằng số, nếu có 2 ký tự trở lên thì đặt dấu – hoặc \_ để nối.
* Gọi tên ID bằng cách viết dấu ‘#’ đầu
* Check code hợp lệ [tại đây](https://validator.w3.org/).
  1. Class:
* Do không thể sử dụng lại tên id -> trong trường hợp muốn khai báo lại thì dùng class.
* Một thẻ có thể có 2 hoặc nhiều class:

<h3 *class*="film film-2">Trường Nguyệt Tẫn Minh 2</h3>

## Combinator selector: Bộ chọn kết hợp

* Bộ chọn CSS:

1. Bộ kết hợp CSS:

* Bộ chọn con cháu (dấu cách)
* Bộ chọn con (>)
* Bộ chọn anh chị em liền kề (+)
* Bộ chọn anh chị em chung (~)

1. Lớp giả CSS:

* Tạo kiểu cho một phần tử khi người dùng di chuột qua nó
* Tạo kiểu khác nhau cho các liên kết đã truy cập và chưa truy cập
* Tạo kiểu cho một phần tử khi nó được lấy nét
* Cú pháp của các lớp giả:

selector:pseudo-class {  
  property: value;  
}

1. Phần tử giả CSS:

* Tạo kiểu cho chữ cái hoặc dòng đầu tiên của một phần tử
* Chèn nội dung vào trước hoặc sau nội dung của một phần tử
* Các thuộc tính sau áp dụng cho ::first-line phần tử giả:
* thuộc tính phông chữ
* đặc tính màu sắc
* thuộc tính nền
* khoảng cách từ
* khoảng cách chữ
* trang trí văn bản
* căn chỉnh theo chiều dọc
* chuyển đổi văn bản
* chiều cao giữa các dòng
* thông thoáng

1. Bộ chọn CSS attr: Bộ [attribute]chọn được sử dụng để chọn các phần tử có thuộc tính được chỉ định.
   1. Combinator selector-p1

* \* Selector: chọn tất cả element (tất cả các thẻ)
* Có thể dùng body cũng được
  1. Conbinator selector-p2
* **element,element**
* VD: div, p: nghĩa là chọn tất cả các thẻ div và thẻ p
* **.class.class**
* VD: .name1.name2: nghĩa là chọn tất cả các thẻ có đủ cả class name1 và name2.
  1. Combinator selctor-p3
* **Space**: sử dụng khoảng trắng để ngăn cách các phần tử giữa bộ chọn kết hợp
* VD: div p: nghĩa là chọn tất cả các thẻ p nằm trong thẻ div
* **Element>element**:
* VD: div>p: nghĩa là chọn tất cả các thẻ p là con trực tiếp của thẻ div
* <div>
* <p>Đoạn văn 1</p>

Thẻ p trên chính là con trực tiếp của thẻ div

<div>

      <p>Đoạn văn 1</p>

      <ul>

        <p>Không có cài gì trong tay</p>

      </ul>

Còn thẻ p nằm trong thẻ ul không phải là con trực tiếp của thẻ div.

* 1. Combinator selector 4
* **Element+element**: chọn một phần tử đầu tiên nằm ngay sau nó (dấu + không áp dụng cho thẻ cha con)
* VD: div+p: nghĩa là chọn thẻ p đầu tiên nằm sau phần tử div.
* **Element~element**: chọn tất cả các thẻ được tác động nằm sau nó
* VD: div~p: nghĩa là chọn tất cả các thẻ p nằm sau thẻ div

## Attribute selector: Bộ chọn thuộc tính

* 1. Attribute part 1

<!-- cú pháp: [attribute]

VD: [href]

ý nghĩa: chọn tất cả các thẻ có thuộc tính (property): href -->

<!-- cú pháp: [attribute="value"],

VD: target="\_blank",

ý nghĩa: chọn tất cả các thẻ có thuộc tính: [target="\_blank"] -->

* 1. Attribute part 2

1. [attribute~=“value”]

<!-- cú pháp: [attribute~="value"],

VD: [title~="public"],

ý nghĩa: chọn tất các thẻ có thuộc tính: "title" có chứa từ "public" trong giá trị -->

1. [attribute\*=value]

<!-- cú pháp: [attribute\*=value],

VD: [title\*="troi"],

ý nghĩa: chọn tất cả các thẻ có thuộc tính: "title"có chứa từ "troi" trong chuỗi giá trị -->

<!-- VD: troi12, troidatsinhsoi,... -->

1. [attribute|=value]

<!-- cú pháp: [attribute|=value],

VD: [title|=em],

ý nghĩa: chọn tất cả các thẻ có thuộc tính: "title" có từ duy nhất bắt đầu bằng chữ "em" -->

<!-- lưu ý: từ đứng đầu tiên chỉ có thể đứng đơn lẻ (một mình nó) hoặc các từ phía sau phải nối nhau bằng dấu "-" -->

1. [attribute^=value]

<!-- cú pháp: [attribute^=value],

VD: [title^=pic],

ý nghĩa: chọn tất cả các thẻ có thuộc tính: "title" có từ bắt đầu bằng chữ "pic" (không quan tâm đằng sau nó như thế nào) -->

1. [attribute$=value]

<!-- cú pháp: [attribute$=value],

VD: [title$=pic],

ý nghĩa: chọn tất cả các thẻ có thuộc tính: "title" mà kết thúc bằng từ "pic" -->

## Pseudo – classes: Lớp giả CSS

Pserdo class: Một lớp giả được sử dụng để định nghĩa một trạng thái đặc biệt của một phần tử

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Pseudo - Classes | VD | Ý nghĩa |
| 1 | **:hover** | **p :hover** | **Chọn các thẻ <p> khi duyệt qua** |
| 2 | **:active** | **a :active** | **Chọn các thẻ <a> khi nhấn giữ chuột** |
| 3 | **:link** | **a : link** | **Chọn các thẻ <a> có liên kết chưa click vào** |
| 4 | **:visited** | **a :visited** | **Chọn các thẻ <a> có liên kết đã click truy cập** |
| 5 | **:empty** | **div :empty** | **Chọn các thẻ <div> không có nội dung** |
| 6 | **:target** | **#home :target** | **Chọn thẻ có id= “home” khi trên URL có hash tương ứng.**  **VD: https://domain.com/#home** |

## Pseudo classes – child selector: thẻ con

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Child selector | VD | Ý nghĩa |
| 7 | **:first child** | **p:first child** | **Chọn các thẻ <p> là thẻ con đầu tiên** |
| 8 | **:last child** | **p:last child** | **Chọn các thẻ <p> là thẻ con cuối cùng** |
| 9 | **:nth-child(n)** | **p:nth-child(2)** | **Chọn các thẻ <p> là thẻ con từ thứ 2 trên xuống** |
| 10 | **:nth-last-child(n)** | **p:nth-last-child(2)** | **Chọn các thẻ <p> là thẻ con từ thứ 2 cuối trở lên** |
| 11 | **:only-child** | **p:only-child** | **Chọn các thẻ <p> là thẻ con duy nhất** |

Trong thẻ :nth-child(n) và thẻ :nth-last-child(n) có thể dùng các từ sau để chọn vị trí:

* odd: nhảy theo (n) lẻ
* even: nhảy theo (n) chẵn
* 2n+1: nhảy theo step mong muốn

## Pseudo classes – type slector

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Type slector | VD | Ý nghĩa |  |
| 12 | **:first of type** | **P:first-of-type** | **Chọn các thẻ <p> xuất hiện đầu tiên trong thẻ cha** |  |
| 3 | **:last-of-type** | **P:last-of-type** | **Chọn các thẻ <p> xuất hiện cuối cùng trong thẻ cha** |  |
| 4 | **:nth-of-type(n)** | **P:nth-of-type(n)** | **Chọn các thẻ <p> xuất hiện thứ n trong thẻ cha** |  |
| 5 | **:nth-last-of-type(n)** | **P:nth-last-of-type(n)** | **Chọn các thẻ <p> xuất hiện thứ n trong thẻ cha, tính từ cuối lên** |  |
| 6 | **:only-of-type** | **P:only-of-type** | **Chọn các thẻ <p> xuất hiện duy nhất trong thẻ cha** |  |

Thẻ :nth-of-type(n) và thẻ :nth-last-of-type(n) cũng có thể dùng các từ như bên child selector để chọn vị trí:

* Odd: nhảy theo (n) lẻ
* Even: nhảy theo (n) chẵn
* 2n+1: nhảy theo step mong muốn

## Pseudo classes - :not(selector) & Quy tắc thứ tự ưu tiên

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Type selector | VD | Ý nghĩa |
| 17 | **:not(selector)** | **.top:not(p)** | **Chọn các thẻ có class=“top” loại trừ thẻ <p>.** |

Selector có thẻ chọn bất kỳ bộ chọn nào ta đã học trong 16 loại ở trên.

1. Thứ tự ưu tiên CSS Selector: mức độ ưu tiên (mạnh -> yếu) áp dụng khi có nhiều thuộc tính chồng chéo nhau cùng tác động lên một phần tử.
2. ***!important***
3. ***Inline style (viết code trong HTML)***
4. ***ID(#) selector***
5. ***Class(.) hoặc Pseudo-class :selector***
6. ***Element selector(h, p, li, div, …)***
7. ***Universal selector (\*) bộ chọn chung***

Bảng điểm ưu tiên

|  |  |  |
| --- | --- | --- |
| Selector | Specificity | Calculation |
| P | 1 |  |
| p.test | 11 |  |
| P#demo | 101 |  |
| <p style=“color: pink”;> | 1000 |  |
| #demo | 100 |  |
| .test | 10 |  |
| .test1.test2 | 20 |  |
| \* | 0 |  |

**Điểm giá trị:**

**\* : 0đ**

**Selector : 1đ**

**Class : 10đ**

**ID : 100đ**

**Style inline : 1000đ**

**!important : tối thượng**

Cách tính điểm: nếu như 2 “specificity” đi với nhau thì điểm giá trị của chúng được cộng lại với nhau

VD: p.demo -> 1+10=11 điểm giá trị

## Play game CSS Selector

<http://game.tuhoc.cc>

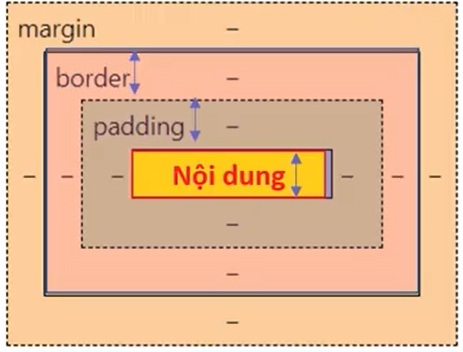
## Box bodel

**Cài đặt: pesticide for chrome**

### Box model - mô hình hộp CSS

Box model: Trình duyệt luôn coi mỗi phần tử là một box – hình chữ nhật

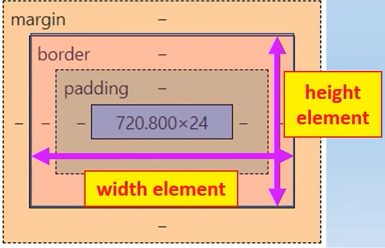
* + Nội dung (widht, height) -> hình ảnh, text, video,…
  + Paddding -> khoảng đệm nội dung và viền (không chứa nội dung)
  + Border -> viền của phần tử (viền nét liền, đứt, …)
  + Margin -> khoảng cách với các phần tử liền kề khác



Kích thước cao \* rộng 1 phần tử

***= content + padding + border***

(Không bao gồm margin)

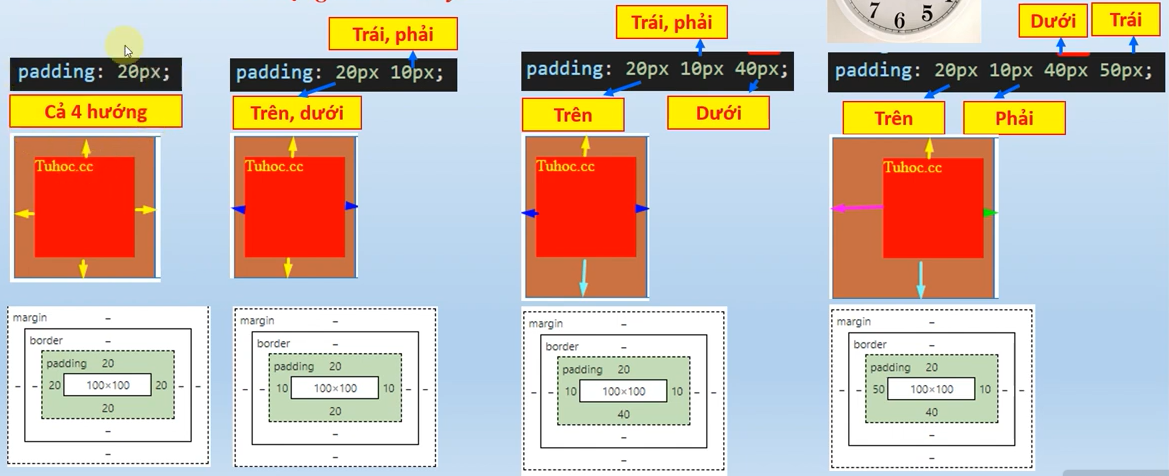


### Box model – Padding

Padding: Được sử dụng để xác định khoảng cách giữa các nội dung bên trong phần tử.

Padding – property: các thuộc tính padding cơ bản

* Padding: 20px; cả 4 hướng
* Padding-top: 20px; phía trên
* Padding-bottom: 20px; phía dưới
* Padding-left: 20px; bên trái
* Padding-right: 20px; bên phải



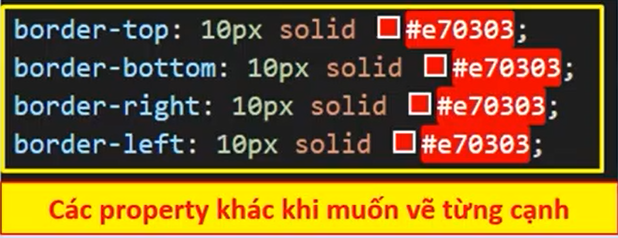
Có thể sử dụng padding như hình trên

### Box model – Border

* + Border: viền
* Cú pháp viết tắt: **Syntax: <line-width> || <line-style> || <color>**

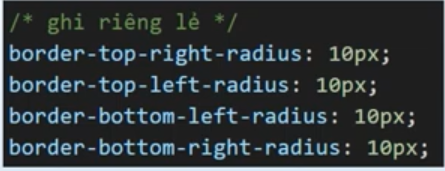
Độ rộng Kiểu vền Màu viền

* Các loại viền:
* Soilid: đường nét liền
* Dashed: đường nét đứt
* Dotted: viền là các chấm dot
* Double: viền là hai đường song song liền nhau
* Groove: viền 3D
* Ridge: viền 3D
* Inset: viền 3D
* Outset: viền 3D
* Trường hợp muốn viền ở cách cạnh khác nhau:



* + Border: cách viết đầy đủ
* Border-width: độ dày của viền
* Border-color: màu viền
* Border-style: kiểu viền
  + Border-radius: Bo góc
* Border-radius: 30px; bo góc cả 4 hướng
* Border-radius: 30px 50px; bo góc trên,dưới trái,phải
* Border-radius: 30px 20px 40px; bo góc trên trái,phải dưới
* Border-radius: 10px 20px 30px 40px; bo góc trên phải dưới trái

Có thể viết cụ thể như thế này:



### Box model – Margin

Margin: Sử dụng để xác định khoảng cách giữa các phần tử trên trang, nó không can thiếp đến nội dụng giữa các phần tử.

Margin giống với padding ở chuyển đổi số.

Nó cũng có thể chỉnh từng cạnh trên, dưới, trái, phải.

* + Dùng margin căn giữa trong CSS:
* **Margin: 0 auto; sau đó dùng max-width: ;**

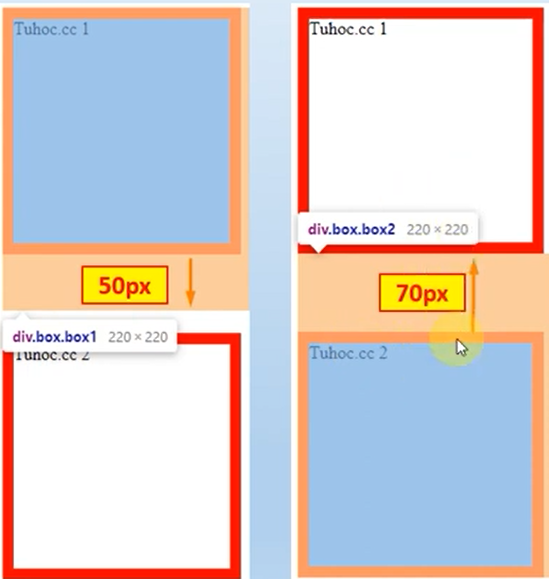
#### Margin – vận dụng margin âm (tạo hình ảnh chồng nhau)

Margin cố định của thẻ body trong css là 8px. **margin: 8px;**

Để ảnh chồng lên nhau thì sửu dụng **margin-left: ;**

#### Margin collapse & cách khắc phục văn bản

**TH 1:** Margin collapsse: Là hiện tượng chồng lặp thuộc tính margin khi hai phần tử liền kề cùng thiết lặp margin với phần tử còn lại.



Hình ảnh trên là ví dụ của margin collapse: trong trường hợp trên thì css sẽ lấy trường hợp có kích thước lớn hơn.

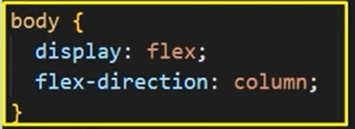
* + **Lưu ý:** margin collapse chỉ tác dụng theo chiều dọc còn chiều ngang thì sẽ cộng hai giá trị lại với nhau.

VD: Nhứ ở trường hợp trên nếu tính theo chiều ngang thì margin của hai phần tử cộng lại sẽ là 120px.

* Nói qua một chút về flexbox: display mặc định của thẻ div là block (display: block;).

Cách khắc phục:

* Chỉ tác động margin một chiều (chỉ tác động lên một phần tử, phần tử còn lại thì giữ nguyên).
* Nếu trong trường hợp bắt buộc thì có thể sử dụng cách sau để xử lý:



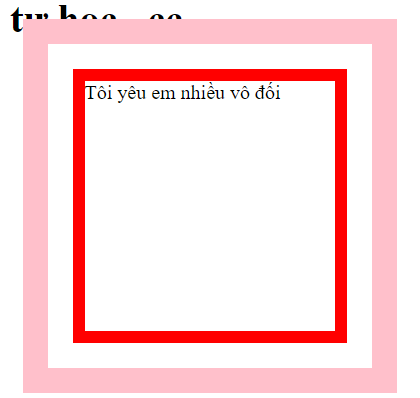
**TH 2:** Xảy ra với phần tử có quan hệ cha con

Khi xét margin cùng chiều cho thẻ cha và thẻ con thì. Thì sẽ bị collapse (kết hợp). Kích thước nào lớn hơn sẽ quyết định.

### Box model – Outline

Outline: Là đường viền bao quanh mép ngoài cùng của đối tượng

* Không làm tăng thêm kích thước của phần tử
* Đơn giản là nó chỉ là một phần bao quanh bên ngoài và chiếm cả phần margin.
* Nó có thuộc tính giống với **boder** nhưng nó chỉ tính từ phần boder đi ra ngoài và có thể chồng lên cả margin.
* Không được áp dụng vẽ theo nhiều chiều như border.
* Có thêm thuộc tính: **outline-offset:** Nó là khoảng cách được tính từ kích thước phần tử đi ran ngoài (nó là phần trắng mũi tên chỉ).



* Nếu nó lớn và chiếm sang đến các phần tử khác thì nó sẽ che luôn phần tử đó(nó che **chữ tự học . cc**)

### Box model – Boxsizing

* + Box sizing: **content-box** và **border-box**

Thực tế dự án ta sử dụng border-box thường xuyên hơn, để kích thước khớp với thiết kế, tránh phải tính toán lại phần nội dung.

Content-box: khi nhập giá trị của width và height thì giá trị đó chính là kích thước của content (có nghĩa là khi thêm các giá trị như padding, margin,… vào trong phần tử thì nó sẽ phóng to ra và ảnh hưởng đến các phần tử khác).

Border-box: khi nhập các giá trị width và height thì giá trị đó chính là kích thước của cả phần tử (có nghĩa là khi thêm các giá trị như padding, margin,… vào trong phần tử thì kích thước của phần tử sẽ giống như giá trị đã nhập vào còn phần content bên trong sẽ nhỏ đi để cố định kích thước của phần tử).

Trạng thái bình thường của box-sizing khi chưa nhập gì là: content-box

## Default CSS value

Default CSS value: Trình duyệt sẽ tự thêm vào các thuộc tính mặc định để hiển thị các thẻ cơ bản chúng ta đã học.

## CSS INHERITANCE – kế thừa trong CSS

Inheritance – kế thừa: Các phần tử con nhận được các giá trị thuộc tính kế thừa từ phần tử cha. Các thuộc tính kế thừa chủ yếu liên quan đến văn bản.

Những thuộc tính được kế thừa:

* Font-family
* Font-size
* Font-weight
* Font-style
* Color
* Text-align
* Line-height
* Letter-spacing
* …

## Reset CSS

### 17.1 Tại sao phải reset CSS

Default CSS value: Trình duyệt sẽ tự thêm vào các thuộc tính mặc định để hiển thị các thẻ HTML, tuy nhiên các trình duyệt khác nhau có thể các thuộc tính được thêm vào sẽ khác nhau.

Các thẻ có CSS mặc định làm khó khăn trong việc kiểm soát kích thước và tính toán kích thước so với thiết kế.

Tìm kiếm từ khóa: browser default styles, tìm kiếm một thẻ bất kỳ (xem các thuộc tính mặc định mà các trình duyệt web có sẵn).

### 17.2 Cách reset CSS

* Lưu ý: Đặt file reset.css ở vị trí trên cùng trước khi sử dụng các CSS khác để tránh file reset.css sẽ ghi đè do thứ tự ưu tiên về đặt vị trí.

Cách reset CSS:

* Tìm kiếm từ khóa reset CSS
* Vào trang web bất kỳ.
* Copy toàn bộ file liên quan đến css
* Tạo một file CSS mới: reset.css
* Dán toàn bộ nội dung vừa copy vào trong file reset.css
* Link html đến file reset.css
* Nhớ lưu ý phía trên.

## CSS Layout-display

Các thẻ HTML đều có thuộc tính display: block, inline, inline-block

Các thẻ HTML đều có trạng thái display mặc định, 2 giá trị cơ bản của display là: block và inline.

### 18.1 Block-element

Các thẻ ở trạng thái mặc định luôn chiếm 100% chiều rộng của thẻ cha bất kể nội dung dài hay ngắn. VD: div, p, h1, h1, …

### 18.2 Inline-element

Hộp có kích thước phụ thuộc vào nội dung bên trong nó (box chứa nó chỉ vừa với nội dung).

* Chú ý: Một số thuộc tính không áp dụng với inline-element
* Padding-top, paddinh-bottom
* Margin-top, margin-bottom
* Width, height, min-width, min-height

Sở dĩ inline không áp dụng được một số thuộc tính trên vì nó sinh ra để nằm trên cùng một hàng tránh phá vỡ cấu trúc của hàng đó.

### 18.3 Inline-Block

Là sự kết hợp của inline và block

## CSS units – đơn vị trong CSS

Đơn vị tuyệt đối:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | unit | VD | Ý nghĩa |
| 1 | **px** | **100px** | **pixels** |

Đơn vị tương đối:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | unit | VD | Ý nghĩa |
| 2 | **%** | **12%** | **Kích thước bằng 12% so với phần tử cha** |
| 3 | **vw(viewport-width)** | **50vw** | **50% chiều rộng khung nhìn(viewport – phần hiển thị trên browser)** |
| 4 | **vh(viewport-height)** | **20vh** | **20% chiều cao khung nhìn(viewport – phần hiển thị trên browser)** |
| 5 | **em** | **2em** | **Kích thước gấp hai lần phông chữ hiện tại (có kế thừa)** |
| 6 | **rem** | **3rem** | **Kích thước gập 3 lần font chữ root element (không kế thừa)** |

Cách dịch phần không thể dịch sang tiếng anh trên trang web:

* Ấn phím f12 hoặc chuột phải sau đó vào phần kiểm tra
* Tìm đến nơi mà không dịch được sang tiếng anh
* Xóa chữ “notransalate”
* Sau đó quay trở lại dịch lại lần nữa.

### 19.1 Pixels – px

Pixel – Picture Element – điểm ảnh là đơn vị cơ bản để đo kích thước một hình ảnh kỹ thuật số.

Khi dùng đơn vị pixel thì trong trường hợp thay đổi nào nó vẫn giữ nguyên giá trị như vậy

### 19.2 Đợn vị %

Khi dùng đơn vị % thì giá trị được lấy đó sẽ có kích thước bằng số % đó của phần tử cha.

VD: thẻ body đang có chiều cao height: 500px; thì khi lấy 50% chiều cao cho phần tử con thì phần tử con sẽ có chiều cao là height: 250px;

* Lưu ý: Nếu thẻ cha không có kích thước thì sẽ lấy kích thước mặc định của trình duyệt. VD font-size mặc định của trình duyệt thường là 16px.

### 19.3 VW-VH

### 19.4 Em

Vì có tính kế thừa nên đơn vị em có các trường hợp sau:

**TH 1:** Thẻ cha và thẻ con không có font-size -> lấy kích thước từ font-size của root (html).

**TH 2:** Thẻ cha có font-size tuyệt đối -> thẻ con được kế thừa từ thẻ cha

**TH 3:** Cả thẻ cha và thẻ con đều đặt font-size -> lấy fon-size hiện tại của chính nó.

**TH 4:** Thẻ cha có font-size tương đối

* Thẻ cha sẽ lấy kích thước mặc định từ trình duyệt
* Thẻ con tiếp tục kế thừa từ thẻ cha
  + Tóm lại, độ ưu tiên của đơn vị **em** giảm dần theo thứ tự đưới đây:
* Font-size chính của phần tử đó
* Phụ thuộc vào font-size của thẻ cha
* Phụ thuộc vào font-size của thẻ root (html)
  + Do có tính kế thừa nên khi dùng cần chú ý, các thẻ con sẽ xếp tầng và kết thừa từ thẻ cha.

### 19.5 Rem

Rem: Có kích thước chỉ phụ thuộc vào font-size thẻ html (root element).

* + Sử dụng rem trong trường hợp muốn thiết kế một khối cô định các kích thước co giãn theo tỉ lệ
  + Tính toán quy đổi **pixel** sang **rem** đơn giản:
* Thông thường 1rem = 16px
* Cần quy đổi tính tỉ lệ 1rem = 10px
* 10/16 \* 100 = 62.5%
* Cần giảm font-size của HTML xuống 62.5%
* Các bước để chỉnh kích thước theo ý muốn:
* Chỉnh thẻ html ghi đè lên thuộc tính mặc định cho font-size xuống 62.5%. Lúc này font-size mặc định của html còn 10px chứ không mặc định là 16px như bình thường.
* Các kích cở còn lại muốn tăng hay giảm thì chỉnh thành đơn vị rem. Lấy đơn vị **px/10.rem** sẽ ra

## CSS Colors

Làm việc với màu sắc

1. Sử dụng RGB(red, green, blue)

* Có đến **16.7** **triệu màu**.
* Có thể bỏ dấu phẩy và ngăn cách nhau bằng dấu cách

1. Sử dụng RGBA(red, green, blue, alpha)

* Alpha chạy từ 0 đến 1 giá trị càng nhỏ thì độ trong suốt càng mờ.
* Giá trị alpha nằm cuối cùng trong bốn số.
* Nếu các giá trị màu mà ngăn cách nhau bằng dấu cách thì pahir suer dụng dấu / đẻ ngăn giá trị alpha
* VD: RGB(255 0 0 **1**) sẽ có màu đỏ đậm nhất

1. Hệ màu hex (hệ thập lục phân)

* Bắt đầu bằng dấu #
* Hai ký tự đầu: red.
* Hai ký tự giữa: green.
* Hai ký tự cuối: blue.
* Giá trị mỗi kênh màu chạy từ 00 -> FF (rgb: 0 -> 255).
* 2 ký tự giống nhau trong mã có thể viết rút gọn thành 1
* Một vài hệ đếm cơ bản:
* Hệ đếm cơ số 10: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
* Hệ nhị phân: 0, 1.
* Hệ lục phân: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f.

Sử dụng hệ màu hex rất ngắn gọn, có thể tiết kiệm dung lượng bộ nhớ.

* Có thể dùng alpha trong hệ thập lục phân: thêm 2 ký tự vào cuối, nó có giá trị chạy từ 00 -> ff.

Sử dụng alpha thì thường laayss màu bằng RGBA sẽ tiện hơn.

1. Color name: tên màu

* Viết tên màu trực tiếp
* Ít dùng vì nó không thể hiện được hết sắc độ

1. Currentcolor

* Sử dụng màu hiện tại mà color đang có
* Vì color có tính kế thừa nên nó có thể kế thừa màu của thẻ cha, nếu muốn sử dụng lại màu của thẻ cha thì có thể sử dụng màu của currentcolor.
* VS Code có thể chỉnh màu theo ý muốn

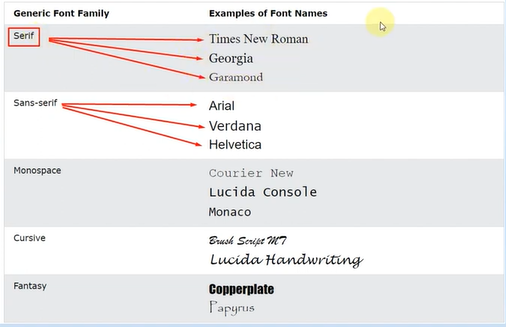
## CSS Fonts

### 21.1 Font name & font family

Các kiểu font chữ chung phổ biến sử dụng trong lập trình web

|  |  |  |
| --- | --- | --- |
| STT | Kiểu font | Minh họa |
| 1 | Font không có gai ở đầu chữ cái **Sans serif** |  |
| 2 | Font có gai ở dầu chữ cái **Serif** |  |
| 3 | Font có cùng độ rộng |  |

**Font name: Tên của họ nhóm font**

****

Search từ khóa **“google font”**

### 21.2 System font CSS – đưa font vào dự án

Sử dụng font chữ của hệ thống – system fonts: Sử dụng font chữ có sẵn trên hệ điều hành máy khách hàng (Mục đích: Tối ưu hóa truy cập, đẩy nhanh tốc độ tải trang).

Mỗi font chữ của hệ điều hành khác nhau thì sử dụng font chữ khác nhau, cái nào viết trước thì cái đó được ưu tiên hơn (mã code viết trong html và css).

Tìm kiêm từ khóa **System font stack**.

Có rất nhiều font chữ đẹp. tuy nhiên không phải font chữ nào cũng có sẵn trên máy khách -> cần cài đặt font chữ nhất quán phòng cả trường hợp máy khách không có font chữ này.

* Nhúng font chữ, sử dụng trên web sử dụng trên trang wbe để bất cứ máy nào cũng hiển thị giống nhau, kể cả trường hợp máy khách không cài đặt font này.

### 22.3 CSS Google Fonts

Tìm kiếm từ khóa **gg font** sau đó tìm kiếm font chữ muốn đặt cho trang web.

Các tiêu đề thường dùng là font chữ có chân.

## Typogrephy in CSS – kiểu chữ

### 22.1 Font-weight

Thuộc tính của font-weight: độ dày của chữ (100 – 900)

### 22.2 Line-height

Thuộc tính của line-height: chiều cao dòng

Khai báo theo số và phần trăm %:

VD: **line-height: 1.5;** chiều cao dòng bằng 1.5 lần font-size

**line-height: 150%** giống như trên.

Có thể dùng line-height để căn giữa, nhưng không khuyến khích dùng line-height để căn giữa.

### 22.3 Letter-spacing

Thuộc tính letter-spacing: tăng giảm khoảng cách giữa các ký tự.

### 22.4 Word-spacing

Thuộc tính word-spacing: tăng giảm khoảng cách giữa các từ.

### 22.5 Text-align

Thuộc tính text-align: căn text.

Các giá trị của text-align:

* Left: căn trái
* Right: căn phải
* Center: căn giữa
* Justify: căn đều ra hai bên

Thuộc tính text-align chỉ căn chỉnh nội dung bên trong box chứa nó. Nhưng trong trường hợp là **display: inline** có box chỉ bao quanh phần nội dung nên nếu dùng text-align thì nó sẽ không thể thực hiện (**display: inline-block** nó cũng không tác động).

### 22.6 Text-dicoration

Các thuộc tính text-dicoration:

* None: bỏ gạch chân
* Underline: thêm gạch chân
* Line-through: thêm gạch ngang chữ
* Overline: gạch trên đầu

Có thể định kiểu của dấu gạch, màu sắc, kích thước (giống boder).

Cũng có thể kết hợp 3 loại dưới cùng với nhau (underline, line-through, overline).

### 22.7 Text-indent

Thuộc tính text-indent: thụt lề

Text-indent: giá trị(px, …);

Chỉ thụt lề của dòng đầu tiền của đoạn văn, các dòng sau giữ nguyên.

### 22.8 Text-transform

Các thuộc tính của text-transform:

* Capitalize: viết hoa chữ cái đầu cúa các từ
* Lowercase: viết thường toàn bộ text
* Uppercase: viết hoa tất cả text

### 22.9 White-space

Thuộc tính white-space: quy định về khoảng trắng khi code.

Ngoại trừ thẻ <pre>, giữ nguyên dạng code:

* Trong code có nhiều khoảng trắng => 1 khoảng trắng
* Nhiều dấu xuống dòng => 1 khoảng trắng
* Nội dung quá dài, chạm tới lề => xuống dòng
* White-space: normal; trạng thái mặc định
* White-space: nowrap; không xuống dòng (kể cả chạm tới lề)
* White-space: pre; giữ nguyên định dạng code, giống thẻ pre.
* White-space: pre-line; giữ định dạng theo line (enter)
* Tất cả khoảng trắng thừa mỗi dòng trong nội dung văn bản sẽ bi loại bỏ.
* Dòng mới (xuống dòng) được giữ nguyên.
* While-space: pre-wrap;
* Khoảng trắng thừa trên mỗi dòng và khoảng trắng thừa khi xuống dòng được giữ nguyên.
* Nhưng khi gặp lề thì sẽ bị cách xuống.

### 22.10 Word-break

Thuộc tính word-break

Khi đoạn text hoặc link dài, người dùng nhập liền nhau thì có thể gây vỡ giao diện vì text không bị xuống dòng.

* + - Có thể dùng thuộc tính word-break: break-all để bẻ xuống dòng.

### 22.11 Text-shadow

Thuộc tính text-shadow: bóng đổ text

Text-shadow: 0px 0px 0px color;

Bốn đối số trên lần lượt là:

* Horizontal: trục ngang (mặc định là đổ bóng qua phải, nếu giá trị âm thì ngược lại)
* Vertical: trục đứng (mặc định là đổ bóng xuống dưới, giá trị âm thì ngược lại)
* Blur: độ nhòe
* Color: màu đổ bóng

### 22.12 Overflow

Thuộc tính overflow: giới hạn nội dung nằm trong box khi kích thước của box nhở hơn độ dài của text.

Các thuộc tính của overflow:



**Text-overflow: ellipsis;** : tạo dấu … khi nội dung văn bản dài bị giới hạn trong một khoảng nhất định và bị ẩn đi nhưng muốn người khác biết vẫn còn đoạn văn bản nữa. (giống phần xem thêm)

### 22.13 Overflow – Scrollbar

Thông thường có một số trang web sẽ có các bài viết nhỏ bên trong và sẽ có các thanh cuộn. Và khi cuộn hết thanh cuộn đó thì nó sẽ cuộn tiếp thanh cuộn chính của trang web đó, vì thê nên chúng ta cần sử dụng **“overscroll-behavior: contain;”**

* Overscroll-behavior: là một thuộc tính CSS sử dụng để điều chỉnh cách trình duyệt xử lý sự kiện “overscrolling”, tức là khi người dùng cuộn trang đến khi không còn nội dung để cuộn nữa.
* Custom scrollbar: tuy biến giúp thanh cuốn đẹp hơn.
* *::-webkit-scrollbar* {
* */\* chiều rộng\*/*
* width: 10px;
* }
* /\* Track \*/: nền của thanh cuộn
* *::-webkit-scrollbar-track* {
* background: #f1f1f1;
* }
* /\* Handle \*/: thanh cuộn
* *::-webkit-scrollbar-thumb* {
* background: #888;
* }
* /\* Handle on hover \*/: hành ddonjogj khi chỉ vào thanh cuộn
* *::-webkit-scrollbar-thumb:hover* {
* background: #555;
* }
* :: gọi là perseudo-element

## Font Awesome

Nhúng font icon vào html bằng cách tìm kiếm từ khóa **awesome font cdn**

## Background

### 24.1 Background-color

Các giá trị cơ bản: chiều dài, chiều rộng, màu, …

Nền sẽ đổ vào toàn bộ các phần tính vào kích thước phần tử (border, padding, content).

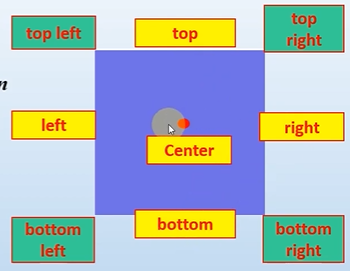
### 24.2 Background-image

Background-image: khi chèn ảnh làm background ở trạng thái mặc định, nếu ảnh nhỏ thì hình ảnh sẽ lặp vô hạn, chiếm hết kích thước cảu phần tử.

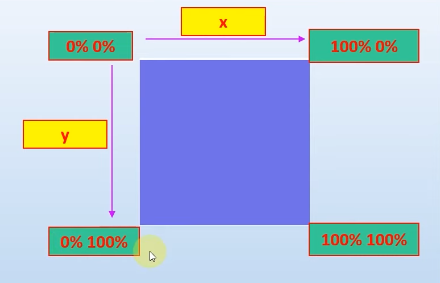
* Thuộc tính background-repeat:
* Mặc định nó là giá trị repeat: nó sẽ lặp lại cả trúc x và y
* No-repeat: không lặp lại
* Repeat-x: lặp lại trên trục x
* Repeat-y: lặp lại trên trục y

Sử dụng kết hợp nhiều ảnh nền: ảnh nào xếp trước sẽ nằm trên, ảnh nào xếp sau sẽ nằm dưới. (có thể dùng cái này để dự phòng trường hợp ảnh bị lỗi thì cá ảnh khác thay thế vào.)

* Thuộc tính background-size: điều chỉnh kích thước ảnh nền
* Background-size: contain; -> ảnh vừa khung hình, giữ nguyên tỉ lệ ảnh, không bị mất ảnh, nếu còn tỉ lệ ảnh sẽ bị lặp lại.
* Background-size: cover; -> ảnh vừa khớp với khung hình, giữ nguyên tỉ lệ, không bị bóp méo, có thể mất một phần hình ảnh, không bị lặp ảnh.
* Có thể điều chỉnh kích thước ảnh bằng đơn vị đo. Bg-size: 100% (100% chiều rộng khung hình) 100% (100% chiều cao khung hình);. Điều chỉnh kích thước bằng đơn vị có thể ảnh sẽ bị bóp méo.
* Thuộc tính bakcground-position: vị trí bắt đầu đặt ảnh nền (áp dụng khi ảnh lớn hơn kích thước của hộp chứa nó và muốn lấy một vị trí ảnh cần trong bức ảnh đó).



* Sử dụng đơn vị % để chọn vị tri bắt đầu của ảnh
* Background-position: %(tính theo chiều ngang) %(tính theo chiều dọc);



* Sử dụng đơn vị px để chọn vị trí bắt đầu
* Thuộc tính linear-gradient: dải màu nền chuyển sắc
* Cú pháp: linear-gradient: hướng chuyển màu, color1 %, color2 %,…
* %: là % không áp dụng chuyển sắc cho từng màu -> có nghĩa là từ hướng bắt đầu chuyển màu đế vị trí % đó sẽ giữ nguyên màu mặc định không chuyển màu
* Cách nhập hướng chuyển màu:
* Mặc định là từ trên xuống dưới
* To left: từ phải sang trái
* To right: từ trái qua phải
* To bottom: trừ trên xuống dưới
* To top: từ dưới lên trên
* To top left: hướng lên trên bên trái
* To top right hướng lên trên bên phải
* Có thể chuyển hướng màu theo độ bằng cách nhập giá trị vào phần hướng chuyển màu (độ.deg)

VD background: linear-gradient(

    145deg,

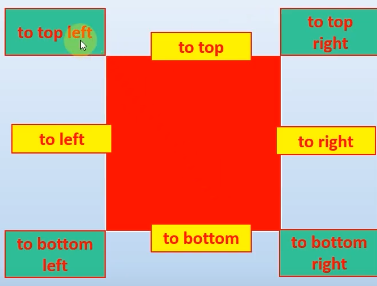
    rgba(21, 63, 226, 1) 30%,

    rgba(23, 138, 218, 1) 65%,

    rgba(8, 211, 223, 1) 100%

  );

* Có thể sử dụng công cụ hỗ trợ [gradient.io](https://cssgradient.io/)



### 24.3 Background-attachment

Thông thường trong các đoạn văn bản sẽ có ảnh đi kèm và khi chúng ta di chuyển thanh cuốn thì ảnh cũng sẽ di chuyển theo, nên chúng ta sẽ sử dụng bg-attachment để điều chỉnh anh theo ý muốn của mình.

Bg-attachment: scroll; khi di chuyển thanh cuộn thì bức ảnh sẽ di chuyển theo.

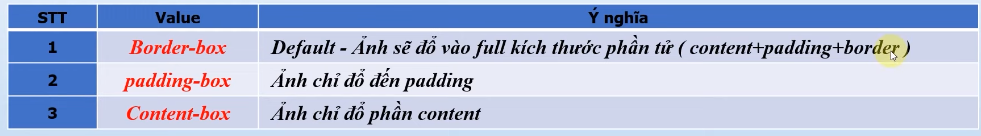
Bg-attachment: fixed; khi cuộn thanh cuốn đi thì ảnh luôn đứng yên.

Bg-attachment: local; khi di chuyển thanh cuốn ảnh sẽ dính liền với nội dung. và di chuyển theo nó.

### 24.4 Background-clip

Thông thường khi ta thêm một hình nền thì hình nề đó sẽ chiếm diện tích của phần tử đó và tính từ boder đi vào. Sử dụng bg-clip để quy định vùng đổ nền cho nó.

Bg-clip: quy định vùng đổ nền



Sử dụng bg-clip để đổ nền chữ: background-clip: text;

* Color: transparent; để màu chữ thành trong suốt

Cách kiểm tra trình duyệt có hỗ trợ thuộc tính hay không:

@supports(background-clip: text) or (*-webkit-background-clip*: text){

// các thuộc tính sẽ được thực hiện nếu trình duyệt có hỗ trợ hai chức năng trên

}

* Làm việc trong dấu ngoặc nhọn: nếu trình duyệt hỗ trợ thì sẽ hiển thị tất cả những gì làm trong dấu ngoặc nhọn, nếu trình duyệt không hỗ trợ thì sẽ không xảy ra việc gì cả.

### 24.5 Background-origin

Thông thường khi sử dụng bg-clip để quy định vùng đổ nền thì quy định ở vùng nào nó sẽ đổ nền ở vùng đó nhưng gốc của bức ảnh tính từ phần boder đi vào và phần không được quy định vùng đổ sẽ bị che mất, nên ta sẽ dùng bg-origin để quy định gốc ảnh cho nó.

Bg-origin: xác định gốc của ảnh nền nằm ở chỗ nào.



### 24.6 Background – shorthand - cú pháp viết tắt

Search từ khóa css background property

Cú pháp: background: bg-color bg-image position/bg-size bg-repeat bg-origin bg-clip bg-attachment initial|inherit;

### 24.7 Backdrop-filter – làm mờ nền ảnh

Backdrop-filter được sử dụng để làm hiệu ứng đồ họa cho khu vực phía sau của một phần tử (cụ thể ở đây là hình n)

Cú pháp: backdrop-filter: none | filter | initial | inherit;

Các thuộc tính của backdrop-filter:

* Blur(value); làm mờ nền với giá trị value
* Brightness(value); điều chỉnh độ sáng của nền
* Contrast(); điều chỉnh độ tương phản của nền
* Grayscale(%); chuyển nền thành ảnh đen trắng
* Invert(); đảo ngược màu nền
* Hue-rotate(deg); xoay màu sắc của nền
* Opacity(); điều chỉnh độ trong suốt của nền
* Saturate(giá trị là một số thập phân); điều chỉnh độ bão hòa của nền
* Sepia(); chuyển màu nền thành ảnh sepia

Search từ khóa backdrop-filter để kiểm tra các thuộc tính

Bg-filter cần prefix vì nó không hỗ trợ 100% các trình duyệt

Có thể sử dụng @support để kiểm tra trình duyệt đó có hỗ trợ hay không

## Clip-path

Search từ khóa css clip path generator

Clip-path:

* Tạo một vùng cắt, nội dung trong vùng cắt sẽ được hiển thị
* Nội dung ngoài vùng cắt sẽ ẩn đi



* Circle(value); Giá trị truyền vào các thuộc tính sẽ là bán kính của hình.
* Cách tính bán kính hình tròn:

/\*

Tính toán bán kính của hình tròn:

radius = (50% \* max(width, height))

Xác định giá trị của max(width, height)

từ kích thước đã cho. Trong trường hợp này,

.box có width: 200px và height: 500px,

vì vậy max(width, height) = 500px.

Tiến hành tính toán bán kính của hình tròn:

radius = (50% \* 500px) = 250px

Vậy, trong trường hợp này,

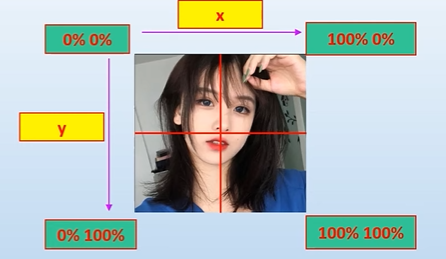
giá trị 50% tương ứng với 250px.

\*/

* Nếu muốn dịch tâm đường tròn theo tọa độ x, y thì sử dụng thuộc tính at, tâm mặc định nằm ở vị trí mà chúng ta đặt bán kính cho nó.

VD: clip-path: circle(40%); trong trường hợp này tâm nó sẽ nằm ở 40% trên trục x và 40% trên trục y

clip-path: circle(40% at 60%(trục x) 60%(trục y)); khi sử dụng at thì tâm sẽ chuyển về vị trí mà chúng ta đặt cho trục x và trục y



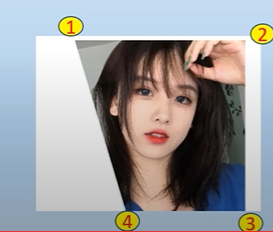
* Inset(1 2 3 4);
* Các giá trị tuyền vào lên lượt là trên, phải, dưới, trái. Nếu nhập vào các giá trị ở đó thì hình sẽ cắt lần lượt theo từng giá trị để tạo ra hình chữ nhật.
* Có thể nhập lần lượt các giá trị giống như border đã từng học.

VD: clip-path: inset(40px); trong trường hợp này sẽ cắt đều cả 4 hướng.

* Thêm bo góc khai báo sau key word round
* Các giá trị bo góc nhập vào tương tự như trên

VD: clip-path: inset(40px 10px 30px 20px round 50%); trong trường hợp này sẽ bo đều cả 4 góc.

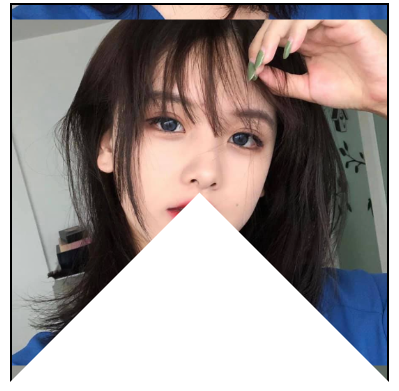
* Polygon();
* Cú pháp chung: cặp x, y là các tọa độ các góc theo x, y
* Clip-path: polygon(xy, xy, xy, xy);
* Các vị trí cắt lần lượt là:



* Sử dụng [CSS clip-path genarator](https://www.cssportal.com/css-clip-path-generator/) để lấy hình mẫu

VD: clip-path: polygon(0 0, 100% 0, 100% 100%, 50% 50%, 0 100%);

Trong VD trên sẽ được hình như thế này:



## Box-shadow

Box-shadow: tạo hiệu ứng đổ bóng box của phần tử.

Sử dụng [Box-shadow css genarator](https://www.cssportal.com/css3-box-shadow-generator/#google_vignette)  hoặc [csssan](https://getcssscan.com/css-box-shadow-examples) để thử nghiệm.

Cú pháp: box-shadow: offset x   offset y    blur   spread   opacity   color;

Các giá trị nhập vào lần lượt là:

* Offset x: đổ bóng về hướng nào theo trục x
* Offset y: đổ bóng về hướng nào theo trục y
* Blur: độ mờ của bóng
* Spread: độ lớn của bóng
* Color: màu cảu bóng

VD: box-shadow: 60*px* -50*px* 10*px* 20*px*  #CECECE;

Kết quả của trường hợp trên là:



Trường hợp trên là đổ bóng ra phía sau, nếu muốn bóng nằm trong box thì sử dụng key word inset nằm đầu tiên trong thuộc tính box-shadow.

VD: box-shadow: inset 60*px* -50*px* 10*px* 20*px* #cecece;

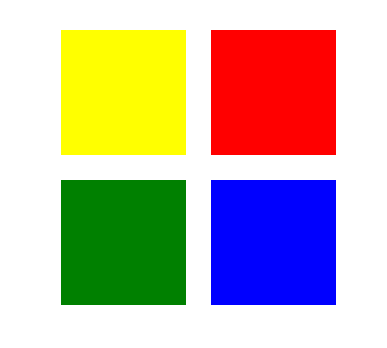
Kết quả của trường hợp này là:



Đổ nhiều bóng cùng lúc:

VD: box-shadow: 120*px* 0 red, 0 120*px* green, 120*px* 120*px* blue;

Kết quả:



## Pointer-events

Pointer-events trong CSS được sử dụng để kiểm soát hoạt động của chuột trên các phần tử HTML.

Sử dụng pointer-events: none; cho phần tử ảnh thì sẽ không thể truy cập được địa chỉ ảnh.

Dùng để ẩn không cho click vào các thẻ a.

## Img – làm việc với hình ảnh

Lazy-load:

* Sử dụng để tối ưu trang web, trong trường hợp nội dung web dài, việc load toàn bộ hình ảnh ngay từ khi truy cập là không cần thiết.
  + - Lazy load: giúp hỗ trợ chỉ tải ảnh khi người dụng cuộn gần đến phần nội dung hình ảnh.
    - Khi sử dụng lazy-load: ảnh chưa load khi người dùng chưa di chuyển đến đoạn văn bản chứa hình ảnh.

## Pseudo-element - tìm hiểu về phần tử giả

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Pseudo-element | VD | Ý nghĩa |
| 1 | ***::first-line*** | ***P::first-line*** | Chọn dòng đầu tiên các thẻ <p> |
| 2 | ***::first-letter*** | ***P::first-letter*** | Chọn chữ các đầu tiên các thẻ <p> |
| 3 | ***::selection*** | ***P::selection*** | Chọn phần tử vùng đang được bôi đen các thẻ <p> |
| 4 | ***::marker*** | ***Li::marker*** | Chọn các điểm đánh dấu của các phần tử <li> |
| 5 | ***::before*** | ***P::before*** | Thêm phần tử vào đằng trước thẻ <p> |
| 6 | ***::after*** | ***P::after*** | Thêm phần tử vào đăng sau thẻ <p> |

Lưu ý:

* First-line không áp dụng thuộc tính được với thẻ inline. Chỉ áp dụng được với thẻ block và inline-block.

## CSS Counter

Css counter: tự động gắn số thứ tự cho các thẻ tiêu đề

## Css calc() function

Hàm cala():

* Là một hàm trong CSS cho phép thực hiện các phép tính đơn giản như + - \* /.
* Có thể sử dụng với tất cả các thuộc tính CSS về kích thước như width, height, margin, padding, top, left, right, background-position, …

## CSS variables

Khi một giá trị CSS dùng đi dùng lại nhiều lần, ta có thể gán biến cho nó để tái sử dụng.

Quy tắc đặt tên biến: --tên\_biến

Quy tắc gọi biến: var(tên\_biến)

* Lưu ý:
* Khai báo root hoặc html sẽ áp dụng cho toàn trang.
* Khai báo trong phần tử nào thì sẽ áp dụng cho phần tử đó và các cấp thấp hơn của nó (con của nó).
* Có thể bị ghi đè theo thứ tự ưu tiên.
* Giá trị backup của biến. Thêm một giá trị sau biến được gọi phòng trường hợp biến được gọi không thể thực hiện.

## CSS position

Thuộc tính position: cho phép điều chỉnh vị trí của các phần tử trên trang web.

### 33.1 Position static

Là giá trị mặc định của position

* + - Các phần tử HTML mà chúng ta biết có vị trí cố định trên trang khi chưa can thiệp bằng CSS.

### 33.2 Position relative

Định vị so với vị trí ban đầu của nó một khoảng x:

* Left: cách lề trái ban đầu một khoảng x;
* Right: …
* Top: cách phía trên ban đầu một khoảng x;
* Bottom: …

Tuy nhiên, không có bất kỳ không gian mới nào được tạo ra để chứa phần tử được chỉ định theo kiểu này, vì vậy các phần tử khác trong trang web vẫn có thể bị che khuất bởi phần tử được chỉ định theo kiểu relative.

### 33.2 Position absolute

Phần tử thực sự được đưa ra khỏi luồng ban đầu.

* Nếu không có tổ tiên có **position: relative** thì phần tử được định vị theo top left corner webpage (góc trái trên cùng của webpage).
* Phần tử được định vị theo tổ tiên gần nhất đặt positon là relative.

### 33.3 Position fixed

Fixed -> fix cứng phần tử theo viewport

### 33.4 Z - Index

Dùng để xác định phần tử nào cao hơn phần tử nào

Phần tử thấp hơn sẽ bị che khuất

* Khi các phần tử có cùng kiểu vị trí (position) được định nghĩa là relative, absolute hoặc fixed.
* Các phần tử static không bị ảnh hưởng bởi z-index.
* Ở trạng thái mặc định, chưa z-index thì phần tử nào viết sau sẽ nằm trên.

### 33.5 Position sticky

Định vị phần tử dính lại theo thẻ cha gần nhất có thanh cuộn scroll bar

## Page layout & component layout

Page layout còn gọi là bố cục - dàn trang cơ bản là cách ta xếp đặt khối bố cục con trên webpage.

Việc xếp đặt giúp cho trang web nhìn đẹp, thu hút người xem và truyền tải nội dung một cách có trật tự và logic.

Các component layout là các bố cục con, nơi mà: text, images, video, … và các nội dung khác được xếp đặt.

* Có 3 cách phổ biến để xây dụng page layout với CSS
* Float layout
* Flexbox
* CSS grid

1. Khái quát về float layout

Float layout có thể giúp dàn trang theo chiều ngang hoặc chữ bào quanh ảnh.

1. Cách sử dụng float layout

ở trạng thái mặc định, các phần tử:

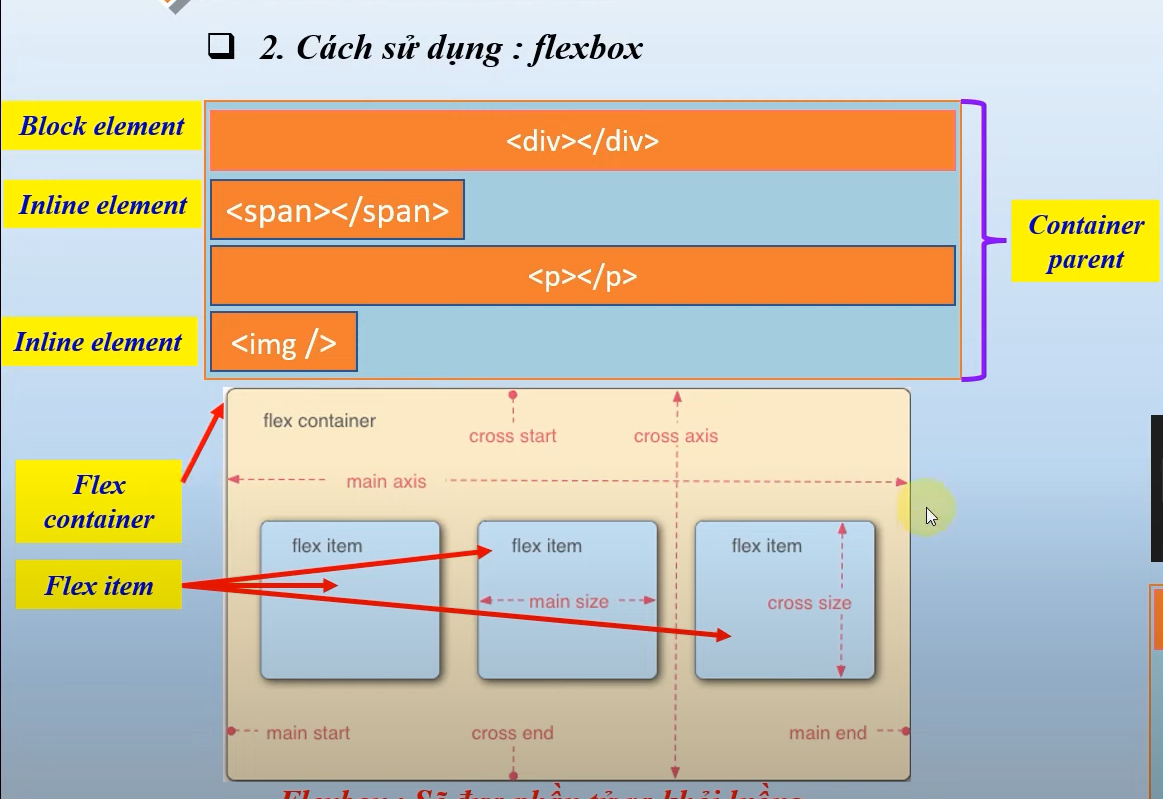
* Inline: nằm trên cùng một hàng
* Block: sẽ chiếm hết chiều ngang

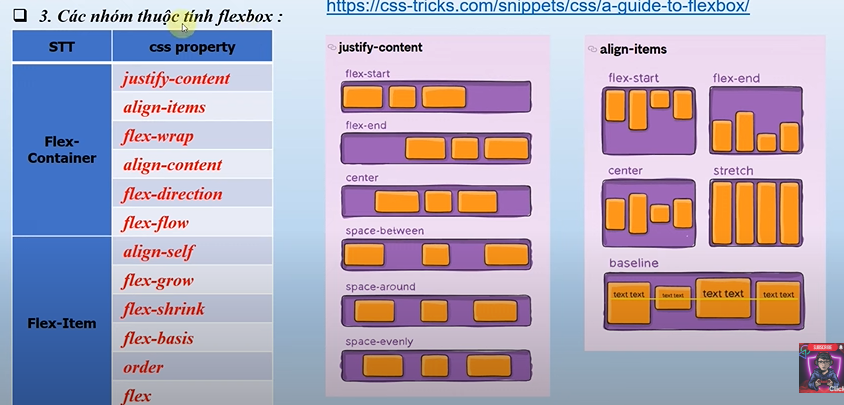
Các phần tử sẽ sắp xếp theo chiều dọc từ trên xuống dưới.

* Khi sử dụng thuộc tính float, thì ảnh sẽ bị trôi khỏi luồn bình thường, các phần tử còn lại sẽ bao quanh hình đó.
* Trong trường hợp muốn phần tử không bị chèn ép để bao quanh hình ta cần sử dụng:
* Clear: left; clear giá trị float bên trái
* Clear: right; clear giá trị float bên phải
* Clear: both; clear giá trị float cả hai bên.

## Flex-box

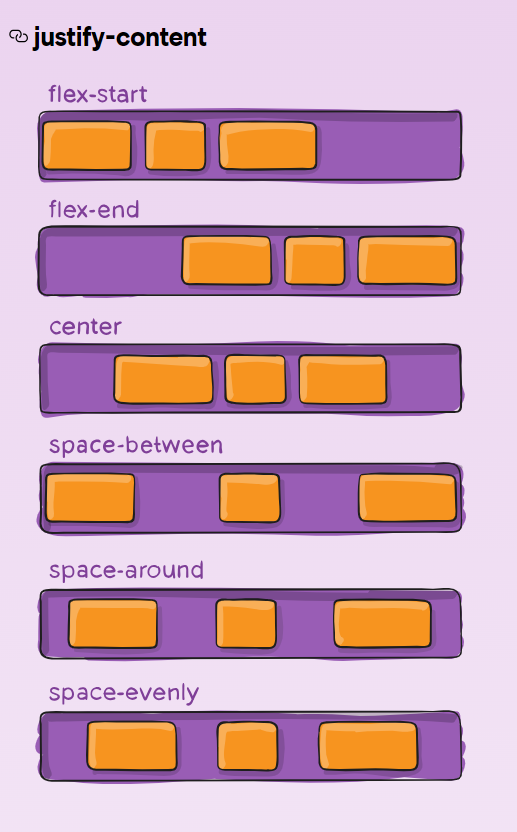
Cách sử dụng flexbox





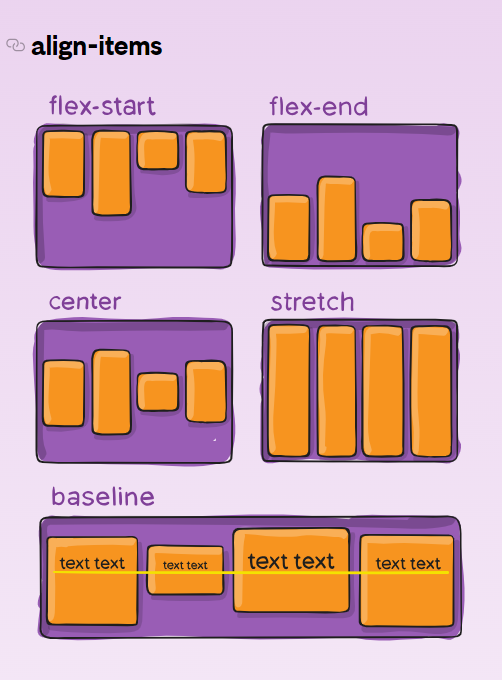
### 35.2 Justify-content - container

Theoc mặc định, trục main axis sẽ nằm ngàng còn trục cross axis sẽ nằm dọc. Justify content sẽ căn các phần tử theo trục main axis.



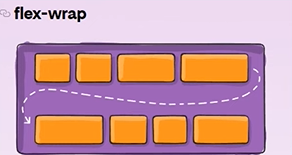
### 35.3 Align-items - container

Align-items có giá trị mặc định là strech: kéo giãn các phần tử bằng container theo trúc cross

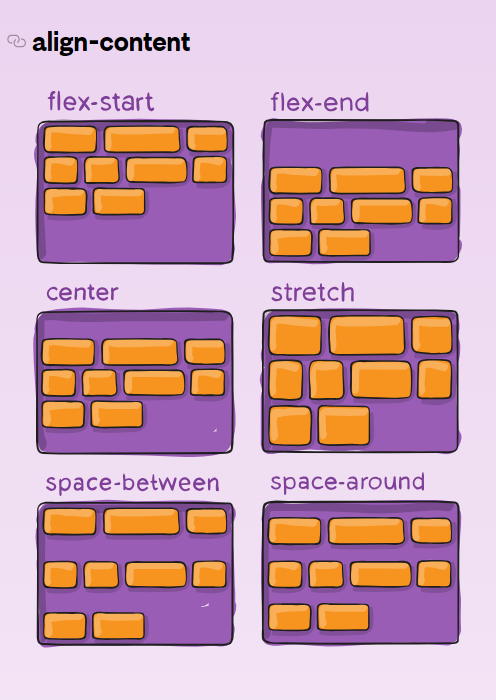


### 35.4 Flex-wrap – align-content – container

Flex-wrap sử dụng để xuống dòng các phần tử khi trên một dòng chiếm quá nhiều phần tử.



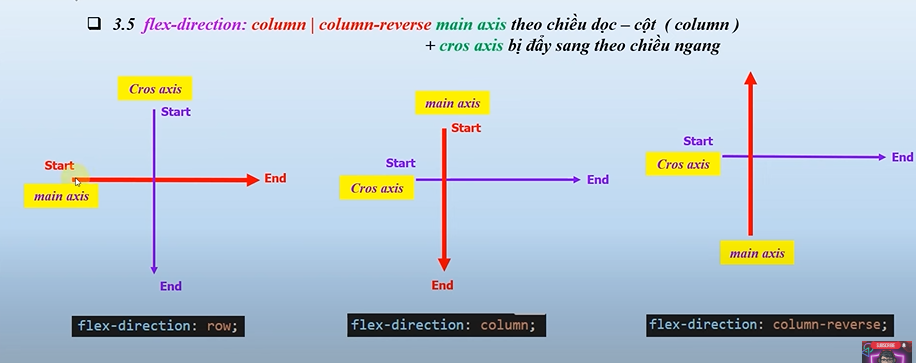
Align-content sử dụng kèm với flex-wrap, dùng để căn chỉnh nhiều hàng theo cros axis.



### 35.5 Flex-direction – flex-flow – container

Flex-direction dùng để xoay hướng trục main axis, cros axis không đổi.

Nếu sử dụng thuộc tính flex-direction: colum thì khi đó trục main axis sẽ xoay dọc còn trục cros axis sẽ xoay ngang.



Flex-flow: flex-wrap | flex-direction cú pháp viết tắt

### 35.6 Học flexbox qua game

### 35.6 Align-seft – flex-grow – Flex-item

Align-seft: flex-start | flex-end | center | baseline | strech;

Flex-grow: chia tỉ lệ chiếm theo trọng số



### 35.7 Flex-shrink – flex-basic – Flex-item

Flex-shrink: xác định khả năng co lại của item nếu cần