Contents

[Flexbox 2](#_Toc162127880)

[Position 2](#_Toc162127881)

[Transition 3](#_Toc162127882)

# Flexbox

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| container | | | item | | |
| Flex-direction  (dọc, ngang) | row | Xếp ngang | Align-self  (ngang) | Auto | Giữa |
| Row-reverse |  | Flex-star | Trên |
| Column | Xếp dọc | Flex-end | Dưới |
| Column-reverse |  | Center | Giữa |
| Flex-wrap  (xuống hàng) | Wrap | Tràn thì xuống dòng | Baseline | Trên biên |
| No-wrap | Khôgn xuống dòng | Stretch | 100% |
| Wrap-reverse | Xuống dòng và đảo ngược | Flex-grow  (chiếm) | Flex-grow:1 |  |
| Align-item  (ngang) | flex-star | Dồn lên trên theo trục ngang | Flex-grow:2 | Chiếm gấp 2 ptu 1 |
| Flex-end | Dồn xuống dưới theo chiều ngang | Flex-shrink |  | Ngược với row |
| Center | Căn giữa theo chiều ngang | Order  (thứ tự) | order:1 | Đứng trước hay đứng sau |
| Baseline | Nằm trên trục ngang |  | Order:2 |  |
| Stretch | Chiếm hết chiều dọc |  |  |  |
| Justyfi-content  (dọc) | Flex-star | Dàn sang trái trục dọc |  |  |  |
| Flex-end | Dàn phải trục dọc |  |  |  |
| Space-between | Dàn đều không trừa biên |  |  |  |
| Center |  |  |  |  |
| space-around | Dàn đều trừa biên |  |  |  |
| Align-content  (ngang) | Flex-star | Dồn theo trục ngang |  |  |  |
| Flex-end |  |  |  |  |
| Space-between |  |  |  |  |
| Center |  |  |  |  |
| space-around |  |  |  |  |
| Stretch |  |  |  |  |

# Position

1. Static (mặc định):

- `position: static;`: Đây là giá trị mặc định, các phần tử được hiển thị theo thứ tự trong HTML và không bị ảnh hưởng bởi các thuộc tính `top`, `bottom`, `left`, `right`.

2. Relative (tương đối):

- `position: relative;`: Các phần tử với vị trí tương đối sẽ được dịch chuyển dựa trên vị trí ban đầu của chúng, và các thuộc tính `top`, `bottom`, `left`, `right` sẽ làm thay đổi vị trí so với vị trí ban đầu.

3. Absolute (tuyệt đối):

- `position: absolute;`: Các phần tử với vị trí tuyệt đối sẽ được dịch chuyển dựa trên vị trí của phần tử cha đầu tiên có `position` không phải `static`, nếu không có thì nó sẽ dựa trên phần tử body. Các thuộc tính `top`, `bottom`, `left`, `right` sẽ làm thay đổi vị trí so với phần tử cha.

4. Fixed (cố định):

- `position: fixed;`: Các phần tử với vị trí cố định sẽ được dịch chuyển dựa trên cửa sổ trình duyệt. Các thuộc tính `top`, `bottom`, `left`, `right` sẽ làm thay đổi vị trí so với cửa sổ trình duyệt.

5. Sticky (dính):

- `position: sticky;`: Các phần tử với vị trí dính sẽ hoạt động như tương đối cho đến khi chúng gặp một ranh giới, sau đó nó sẽ hoạt động như tuyệt đối.

Ranh giới trong trường hợp này là một ngưỡng được thiết lập bởi một phần tử cha có thuộc tính **overflow**, **overflow-x**, hoặc **overflow-y** không phải là **visible**. Khi phần tử với **position: sticky;** vượt qua ranh giới này, nó sẽ bắt đầu hoạt động như một phần tử với **position: fixed;**, tức là nó sẽ giữ vị trí của mình đối với cửa sổ trình duyệt.

# Transition

Trong CSS, thuộc tính `transition` cho phép bạn thay đổi trơn tru các thuộc tính của một phần tử khi các sự kiện như hover, focus, active, hoặc khi thay đổi trạng thái của phần tử. Điều này tạo ra hiệu ứng chuyển đổi mượt mà giữa các trạng thái khác nhau của phần tử.

Cú pháp cơ bản của `transition` như sau:



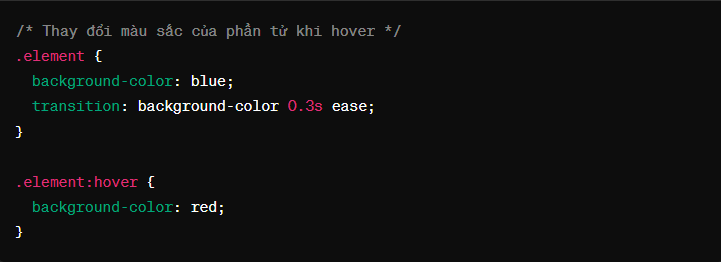
- `property`: Thuộc tính CSS mà bạn muốn áp dụng chuyển đổi.

- `duration`: Thời gian mà chuyển đổi sẽ diễn ra, có thể là giây hoặc mili giây.

- `timing-function`: Hàm thời gian xác định cách chuyển đổi thay đổi qua thời gian. Các giá trị thông dụng bao gồm `ease`, `linear`, `ease-in`, `ease-out`, `ease-in-out`, và `cubic-bezier(n,n,n,n)` (với các giá trị từ 0 đến 1).

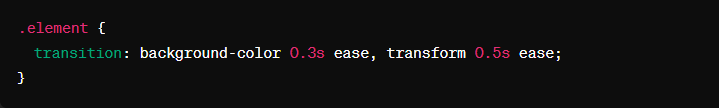
- `delay`: Thời gian trì hoãn trước khi chuyển đổi bắt đầu.

Ví dụ đơn giản:



Trong ví dụ trên, khi bạn di chuột qua phần tử có lớp `.element`, màu nền sẽ chuyển từ màu xanh sang màu đỏ trong vòng 0.3 giây với hiệu ứng chuyển đổi mềm mại (`ease`).

Ngoài ra, bạn có thể áp dụng chuyển đổi cho nhiều thuộc tính cùng một lúc:



Trong ví dụ này, màu nền và thuộc tính biến đổi (`transform`) của phần tử sẽ được chuyển đổi một cách mượt mà khi có sự thay đổi.

# Animation

Thuộc tính `animation` cho phép bạn tạo ra các hiệu ứng hoạt ảnh động cho các phần tử trên trang web.

Cú pháp:

animation: name duration timing-function delay iteration-count direction fill-mode play-state;

Các thuộc tính:

- `name`: Tên của keyframe animation.

- `duration`: Thời gian một vòng hoạt động của hoạt ảnh (ví dụ: `2s` hoặc `200ms`).

- `timing-function`: Hàm thời gian (ví dụ: `ease`, `linear`, `ease-in`, `ease-out`, `ease-in-out`).

- `delay`: Thời gian trễ trước khi hoạt động bắt đầu.

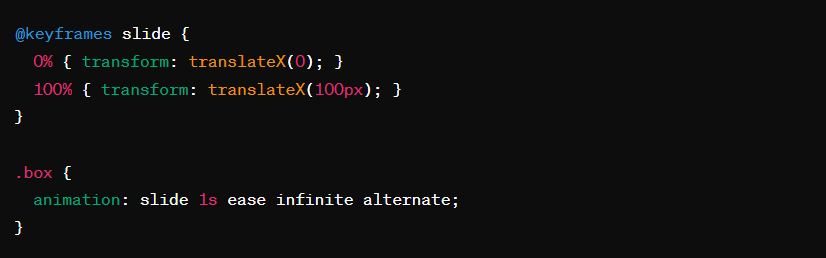
- `iteration-count`: Số lần mà hoạt ảnh lặp lại (ví dụ: `infinite`).

- `direction`: Hướng hoạt động của hoạt ảnh (ví dụ: `normal`, `alternate`).

- `fill-mode`: Xác định trạng thái của phần tử trước và sau khi hoạt động (ví dụ: `forwards`, `backwards`).

- `play-state`: Xác định trạng thái hoạt động của hoạt ảnh (ví dụ: `running`, `paused`).

Ví dụ:



Trong ví dụ này, `.box` sẽ di chuyển sang phải 100px và sau đó di chuyển về vị trí ban đầu trong 1 giây. Hiệu ứng này sẽ lặp lại vô hạn.

# Transform

Thuộc tính `transform` cho phép bạn thực hiện các biến đổi như di chuyển, xoay, co giãn cho phần tử.

Cú pháp:



Các hàm:

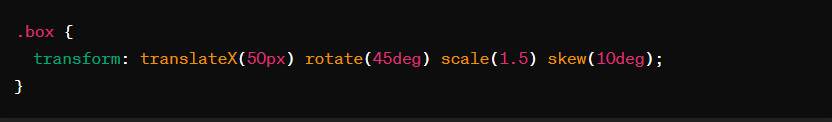
- `translate()`: Di chuyển phần tử.

- `rotate()`: Xoay phần tử.

- `scale()`: Co giãn hoặc thu nhỏ phần tử.

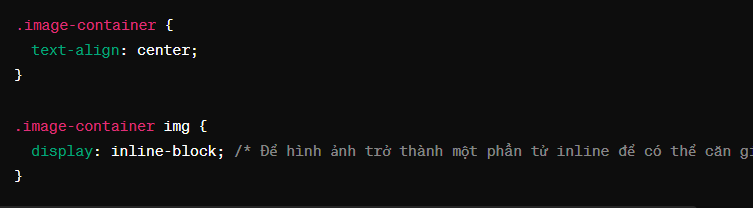
- `skew()`: Nghiêng phần tử.

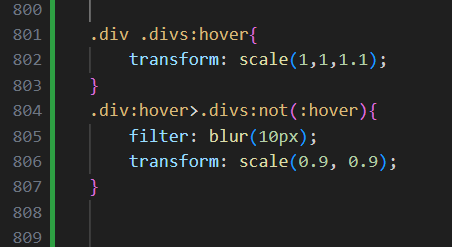
Ví dụ:



Trong ví dụ này, `.box` sẽ được di chuyển sang phải 50px, xoay 45 độ, co giãn lên 1.5 lần kích thước ban đầu, và nghiêng 10 độ.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

1. **Bố cục và Kích thước**:
   * **display**: Điều chỉnh cách phần tử được hiển thị (block, inline, inline-block, flex, grid).
   * **position**: Điều chỉnh vị trí của các phần tử trên trang (relative, absolute, fixed, sticky).
   * **width**, **height**: Xác định kích thước của phần tử.
   * **margin**, **padding**: Điều chỉnh khoảng cách giữa các phần tử và nội dung bên trong chúng.
2. **Màu sắc và Nền**:
   * **color**: Xác định màu chữ.
   * **background-color**: Đặt màu nền cho phần tử.
   * **background-image**: Sử dụng hình ảnh làm nền cho phần tử.
   * object-fit: cover; /\* Thay đổi kích thước của hình ảnh để nó lấp đầy hoàn toàn khung chứa \*/
   * background-size: cover;
   * **Căn giữa ngang (center horizontally)**
   * 
3. **Kiểu chữ và Định dạng văn bản**:
   * **font-family**: Chọn font chữ.
   * **font-size**: Xác định kích thước của chữ.
   * **font-weight**: Xác định độ đậm nhạt của chữ.
   * **text-align**: Căn chỉnh văn bản trong phần tử.
   * **line-height**: Điều chỉnh độ cao của dòng chữ.
4. **Hiệu ứng và Chuyển động**:
   * **transition**: Tạo hiệu ứng chuyển đổi mượt mà khi có sự thay đổi trạng thái của phần tử.
   * **animation**: Tạo các hoạt ảnh động cho các phần tử.
   * **transform**: Thực hiện các biến đổi (như di chuyển, xoay, co giãn) cho phần tử.
5. **Responsive Design**:
   * **@media**: Định nghĩa các điều kiện responsive cho các kích thước màn hình khác nhau.
   * **flexbox**: Sử dụng để tạo bố cục linh hoạt và dễ dàng điều chỉnh trên các kích thước màn hình khác nhau.
   * **grid**: Cho phép bạn tạo lưới bố cục linh hoạt và phức tạp hơn.
6. **Tranh trích và Giao diện người dùng**:
   * **box-shadow**: Tạo bóng đổ cho các phần tử.
   * **border-radius**: Làm tròn góc của các phần tử.
   * **hover**, **focus**, **active**: Tạo hiệu ứng khi người dùng tương tác với các phần tử.
7. **Thiết kế Phẳng (Flat Design)**:
   * **color**: Sử dụng màu sắc tươi sáng, rõ ràng và ít gradient.
   * **typography**: Sử dụng font chữ rõ ràng, dễ đọc và ít phức tạp.
   * **simplicity**: Giữ giao diện đơn giản và trực quan.



Đoạn code CSS bạn cung cấp là để tạo hiệu ứng khi bạn hover vào một phần tử và tác động đến các phần tử con của nó.

1. `.div .divs:hover { transform: scale(1,1,1.1); }`: Khi bạn di chuột qua phần tử có lớp `divs` và nằm trong một phần tử có lớp `div`, nó sẽ thực hiện transform với thuộc tính scale. Thuộc tính scale trong trường hợp này được sử dụng để phóng to phần tử một cách tuyến tính trên cả hai trục x và y, và giữ nguyên trạng thái ban đầu trên trục z (do chỉ có hai giá trị được chỉ định là 1, không thay đổi trên trục z). Cụ thể, thuộc tính `transform: scale(1,1,1.1);` đang phóng to phần tử `.divs` lên 1.1 lần trên trục z khi bạn hover vào nó.

2. `.div:hover > .divs:not(:hover) { filter: blur(10px); transform: scale(0.9, 0.9); }`: Khi bạn hover vào phần tử có lớp `div`, nó sẽ tác động lên các phần tử con của nó có lớp `divs`, nhưng không phải phần tử nào cũng đang được hover. Trong trường hợp này, nó sử dụng `:not(:hover)` để xác định phần tử `.divs` không được hover. Nó áp dụng một filter là blur với giá trị 10px, làm mờ phần tử `.divs` không được hover. Đồng thời, nó cũng thực hiện một transform với thuộc tính scale, phóng nhỏ phần tử `.divs` xuống 0.9 lần trên cả hai trục x và y.