

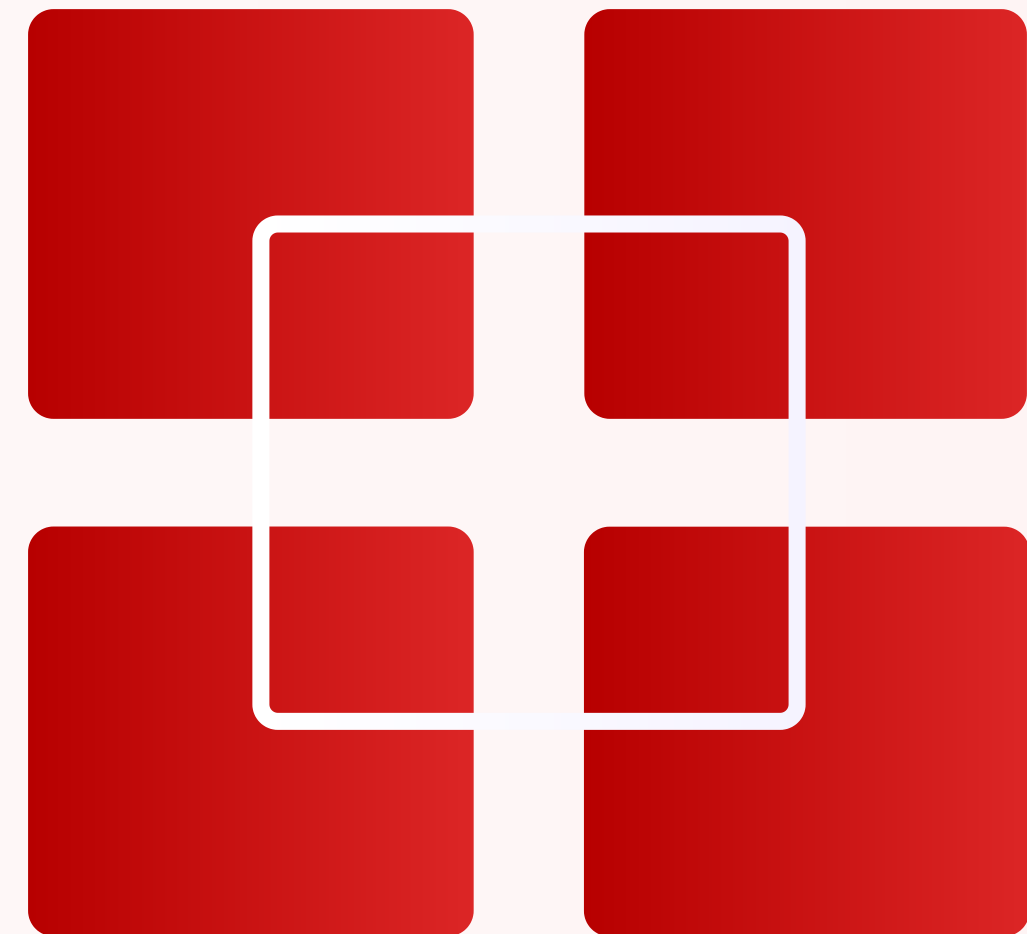


# HTML & CSS advance

Day 2: Using HTML & CSS

# Nội dung buổi học

- ✨ **Layout trong CSS**
- 📄 **FlexBox, Grid**
- 🛠️ **Position and Z-index** - position, z-index
- 🧑‍🎓 **CSS Animations** - Transition, Transform, css3 animation
- 📄 **Using framework of CSS** - bootstrap
- 🧑💻 **Bài tập sử dụng html và css** - css thuần, framework



## Layout trong CSS

Layout có thể hiểu là cách mà chúng ta bố trí các thành phần chính trên một trang web. Thẻ `<div>` thường được dùng để phân chia các thành phần chính của trang web kết hợp với thuộc tính định dạng CSS.

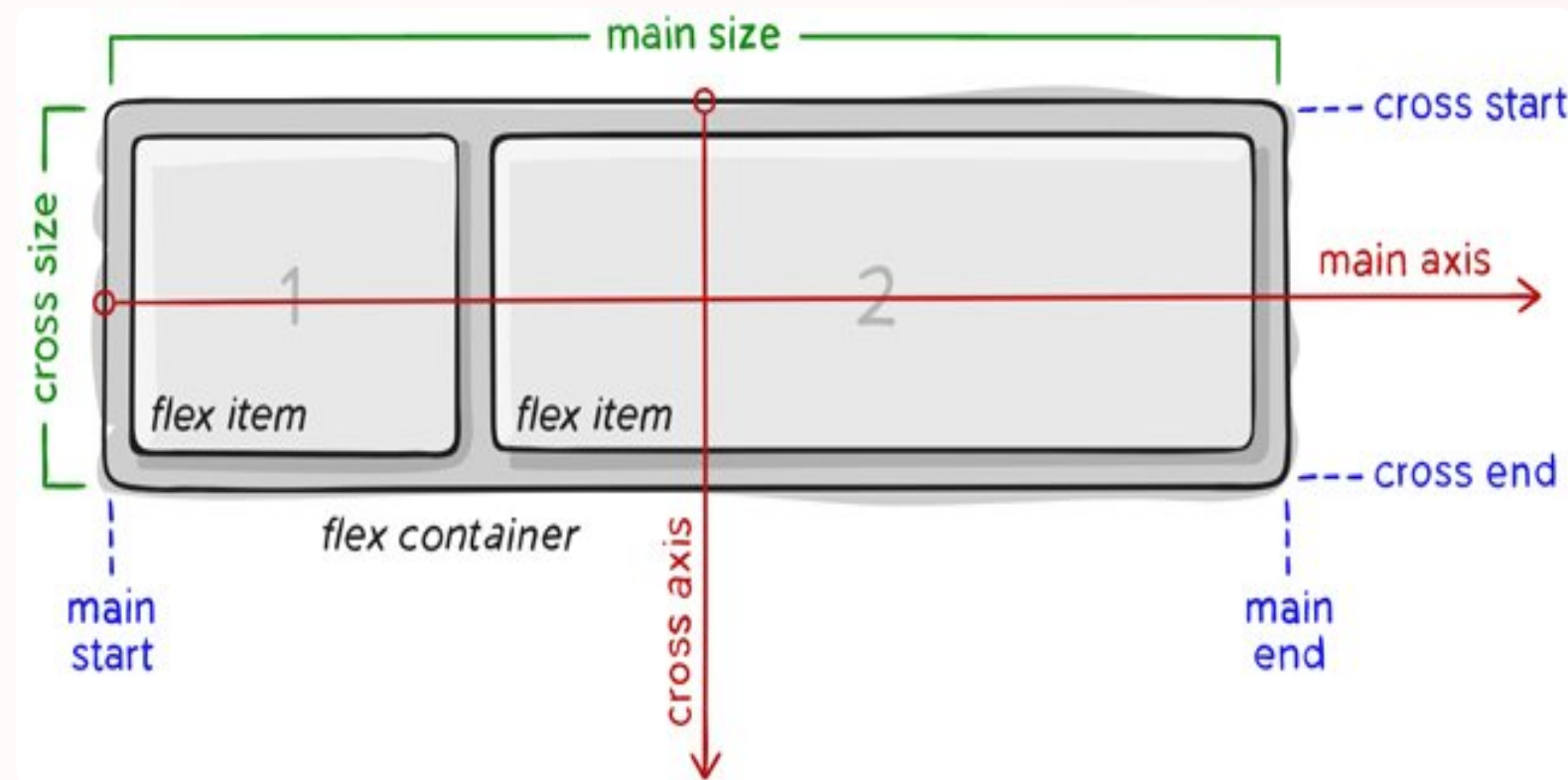


## LAYOUT

- **Header** thường nằm ở đầu trang web (hoặc ngay bên dưới menu điều hướng trên cùng). Phần này thường chứa logo hoặc tên website hay một vài khẩu hiệu của trang web.
- **Thanh điều hướng - Navigation Bar** hay còn gọi là thanh menu, được dùng để điều hướng các mục chính trên website.
- **Nội dung** Layout trong section này thường phụ thuộc vào đối tượng người dùng
- **Footer** được đặt ở cuối trang, thường chứa thông tin như bản quyền, thông tin liên lạc...

## FLEXBOX

- Flexbox là một kiểu dàn trang (layout mode) mà nó sẽ tự cân đối kích thước của các phần tử bên trong để hiển thị trên mọi thiết bị
- Bố cục Flex được thiết lập từ một khung lớn (parent container) đóng vai trò là khung linh hoạt (flex container) và các thẻ con ngay trong nó (immediate children) đóng vai trò các mục nhỏ linh hoạt (flex item). sơ đồ cấu trúc Flexbox:

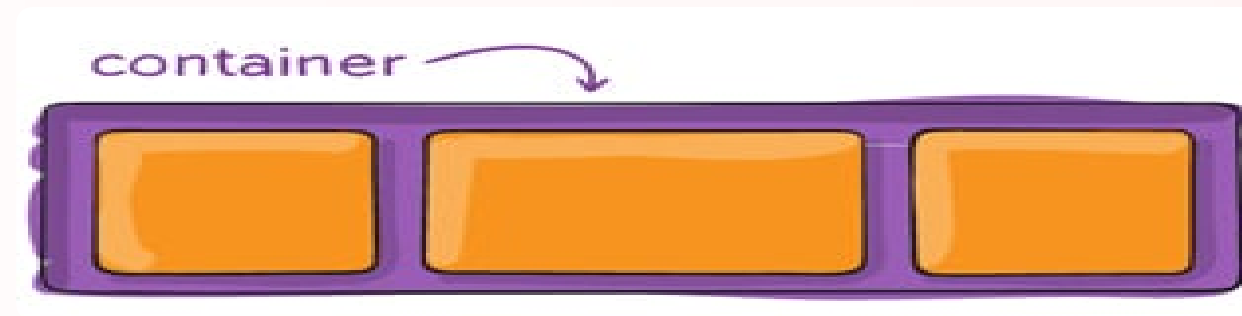


Tài liệu tham khảo: <https://yoksel.github.io/flex-cheatsheet/>

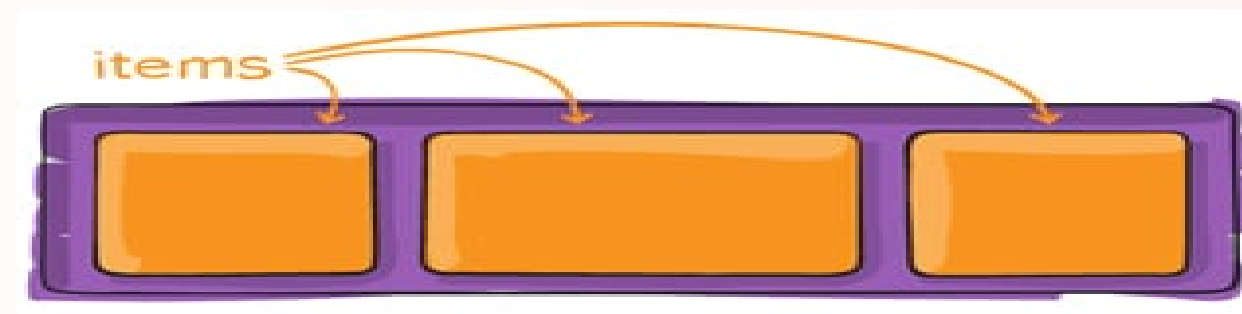
## FlexBox

Thành phần quan trọng của Flexbox gồm:

- **container:** là thành phần lớn bao quanh các phần tử bên trong, các item bên trong sẽ hiển thị dựa trên thiết lập của container này.



- **item:** là phần tử con của container, bạn có thể thiết lập nó sẽ sử dụng bao nhiêu cột trong một container, hoặc thiết lập thứ tự hiển thị của nó.



**Note:** Các item sẽ được bố trí theo trục main axis (bắt đầu từ main-start, kết thúc ở main-end) hoặc theo trục cross axis (bắt đầu từ cross-start, kết thúc ở cross-end).

## CÁCH SỬ DỤNG

- **display**: Để sử dụng Bố cục Flexbox bạn chỉ cần đặt giá trị cho thuộc tính display trên khung lớn (parent container).

```
.flex-container {  
  display: -webkit-flex; /* Safari */  
  display: flex;  
}
```

Hoặc bạn muốn nó hiển thị như một phần tử inline....

```
.flex-container {  
  display: -webkit-inline-flex; /* Safari */  
  display: inline-flex;  
}
```

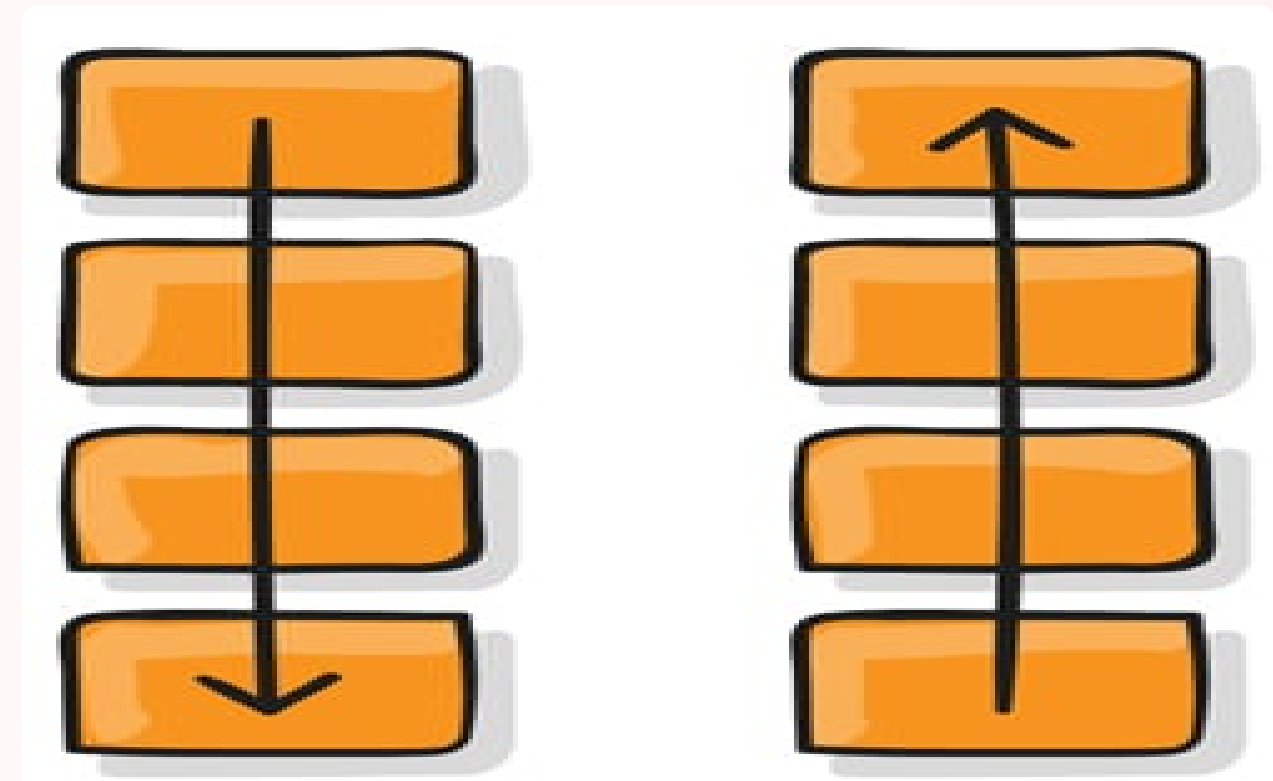
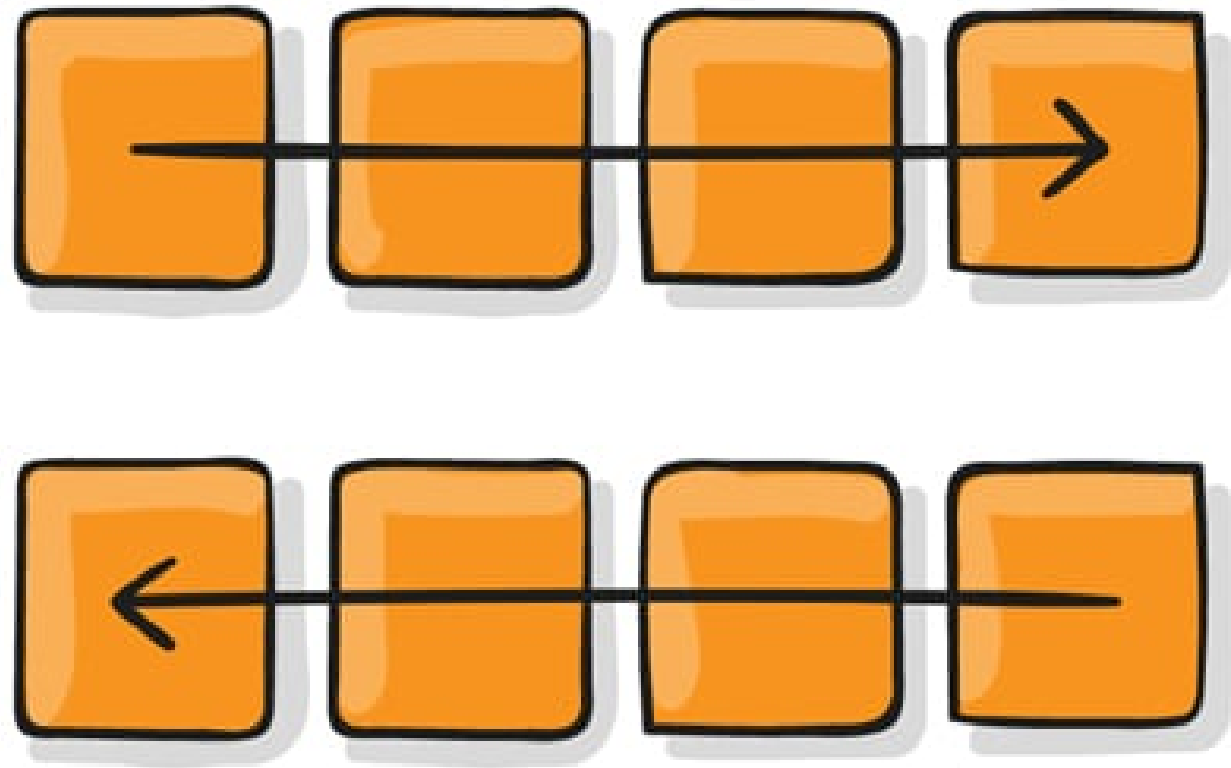
NOTE: Bạn chỉ cần đặt thuộc tính trên vào khung lớn là các thẻ con sẽ lập tức trở thành các mục linh hoạt.

## CÁC THUỘC TÍNH CỦA FLEXBOX

### flex-direction

Thuộc tính flex-direction xác định hướng của main-axis để container sắp xếp các item.

```
.container {  
  flex-direction: row | row-reverse | column | column-reverse;  
}
```

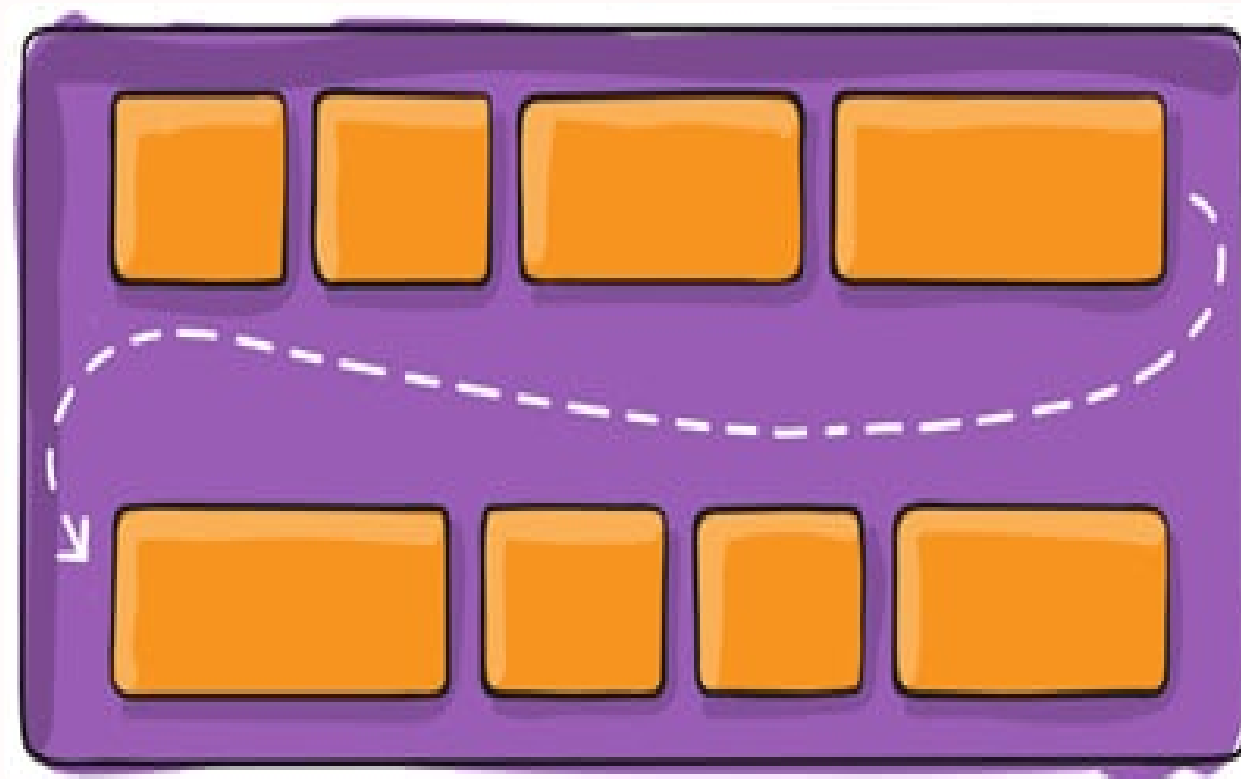




## flex-wrap

Tạo một bố cục với các mục xếp thành nhiều hàng.

```
.container{  
  flex-wrap: nowrap | wrap | wrap-reverse;  
}
```



## flex-flow

Sử dụng để gộp chung hai thuộc tính flex-direction và flex-wrap.

```
`flex-flow: <'flex-direction'> || <'flex-wrap'>`
```

```
.flex-container {  
  display: flex;  
  -webkit-flex-flow: <flex-direction> || <flex-wrap>; /* Safari */  
  flex-flow:      <flex-direction> || <flex-wrap>;  
}
```

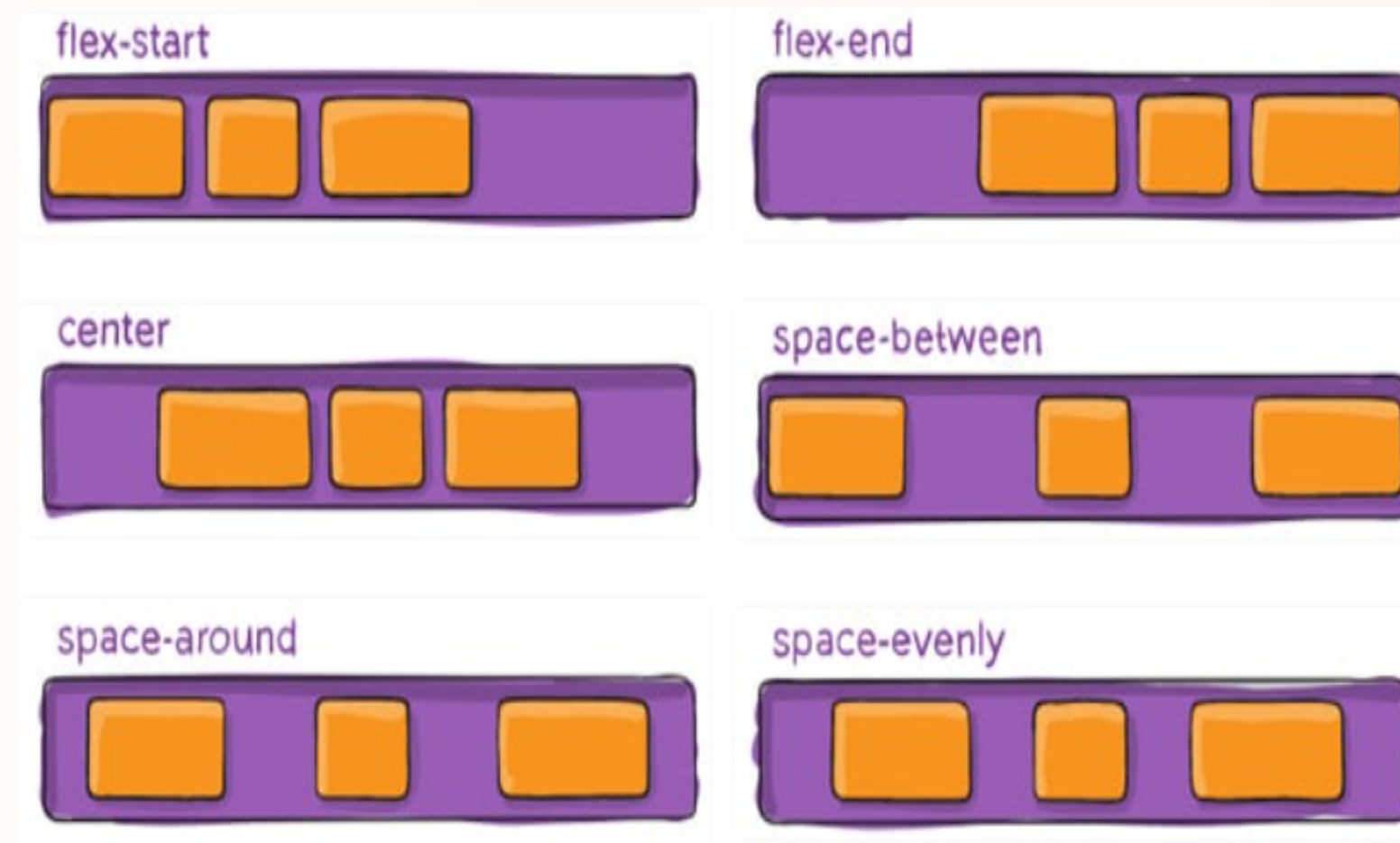
Giá trị mặc định: row nowrap

## justify-content

```
.container {  
  display: flex;  
  justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly;  
}
```

- **flex-start**: giá trị mặc định, item sẽ bắt đầu từ lề chính main-start của container.
- **flex-end**: item sẽ bắt đầu từ lề chính main-end của container (khác với row-reverse là đổi hướng hiển thị).
- **center**: item sẽ nằm giữa container.
- **space-between**: các item sẽ có khoảng cách giữa các phần tử bằng nhau do container sẽ tự động căn khoảng cách, item đầu tiên sát lề chứa điểm main-start, item cuối cùng sát lề chứa điểm main-end.
- **space-around**: tương tự space-between, nhưng khác ở chỗ là mỗi item có khoảng cách 2 bên cạnh và những khoảng cách này bằng nhau.
- **space-evenly**: các item được phân phối sao cho khoảng cách giữa hai item bất kỳ, giữa item và các lề là bằng nhau.

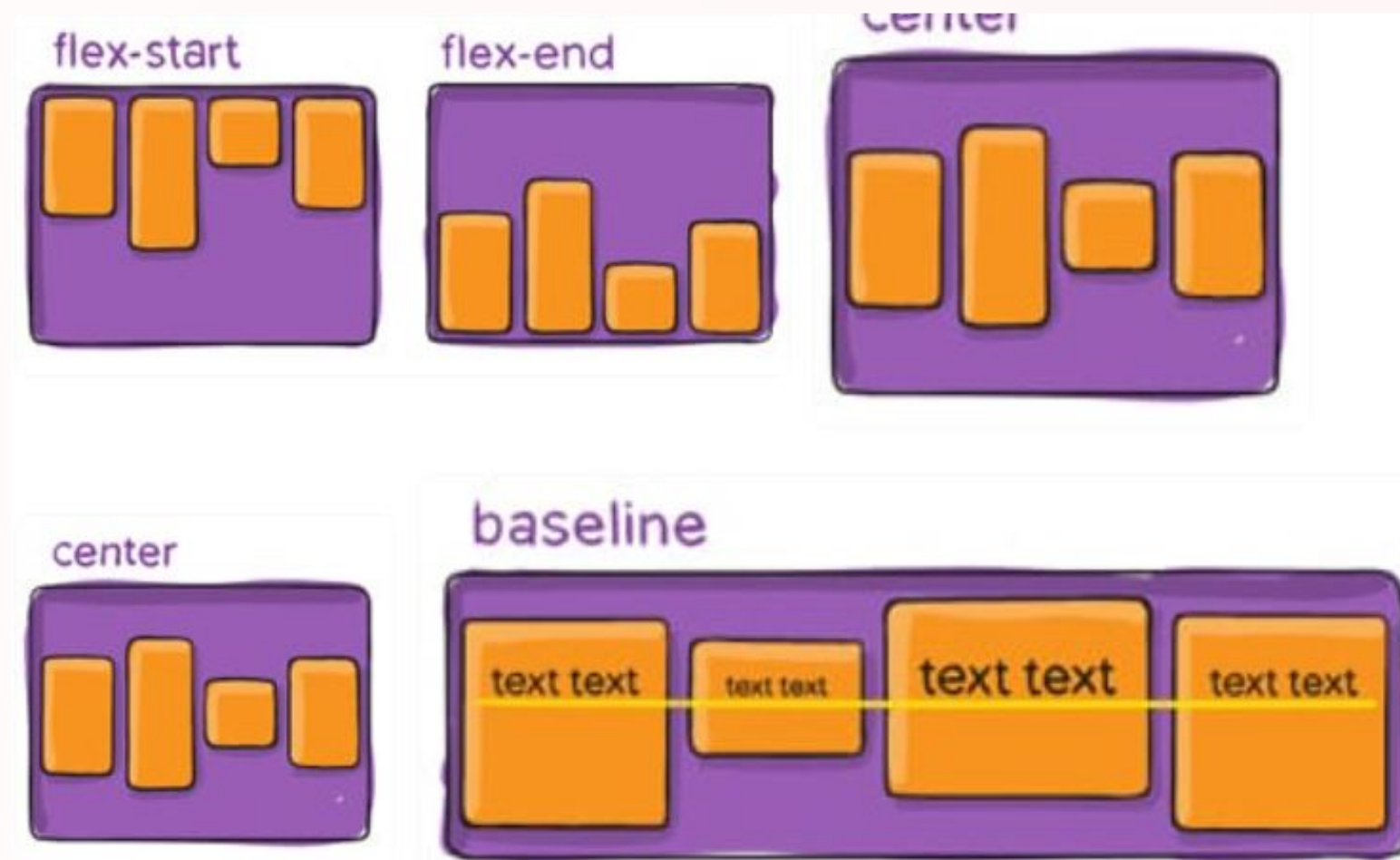
## justify-content



## align-items

Thuộc tính align-items sử dụng để điều chỉnh vị trí bắt đầu và căn chỉnh các item bên trong container theo dọc theo trục cross axis, chiều ngang hoặc chiều dọc tùy thuộc vào flex-direction.

```
.container {  
  align-items: stretch | flex-start | flex-end | center | baseline;  
}
```



## align-items

---

- stretch: giá trị mặc định, các phần tử sẽ được kéo dài để lấp đầy container chứa nó, nhưng sẽ ưu tiên giá trị height/width nếu có, khi đó item sẽ không cao full mà chỉ lấy giá trị height/width mà bạn set
- flex-start: item sẽ bắt đầu từ lề cross-start của container.
- flex-end: item sẽ bắt đầu từ lề cross-end của container. Trường hợp mặc định với cross axis đứng dọc, flex-direction: row thì các item sẽ dồn xuống dưới.
- center: item sẽ căn giữa theo chiều của cross axis.
- baseline: item được căn chỉnh theo đường cơ sở của chúng.

## Các thuộc tính của Flex Item

- order: các item sẽ hiển thị theo thứ tự xuất hiện trong HTML, nhưng với thuộc tính order, bạn có thể sắp xếp lại vị trí sắp xếp của các item.
- flex-grow: cho phép các phần tử giãn theo độ rộng của container.
- flex-shrink: ngược lại với thuộc tính flex-grow ở trên, nó không giãn ra mà lại co lại khi chúng ta thay đổi độ rộng của container.
- flex-basis: Thuộc tính flex-basis sử dụng để xác định độ dài ban đầu của một item.
- align-self: Thuộc tính align-self có tác dụng tương tự như align-items của phần container nhưng sử dụng riêng cho từng item, bạn có thể dùng nó để đặt lại vị trí cho một số item mà align-items đã quy định.

## GRID

Hệ thống Grid Layout của CSS cung cấp một hệ thống bố cục dạng lưới, với cột và hàng (Khá giống với col và row của Bootstrap) mà không còn cần sử dụng đến float hay position.

```
.container {
  display: grid;
}
```

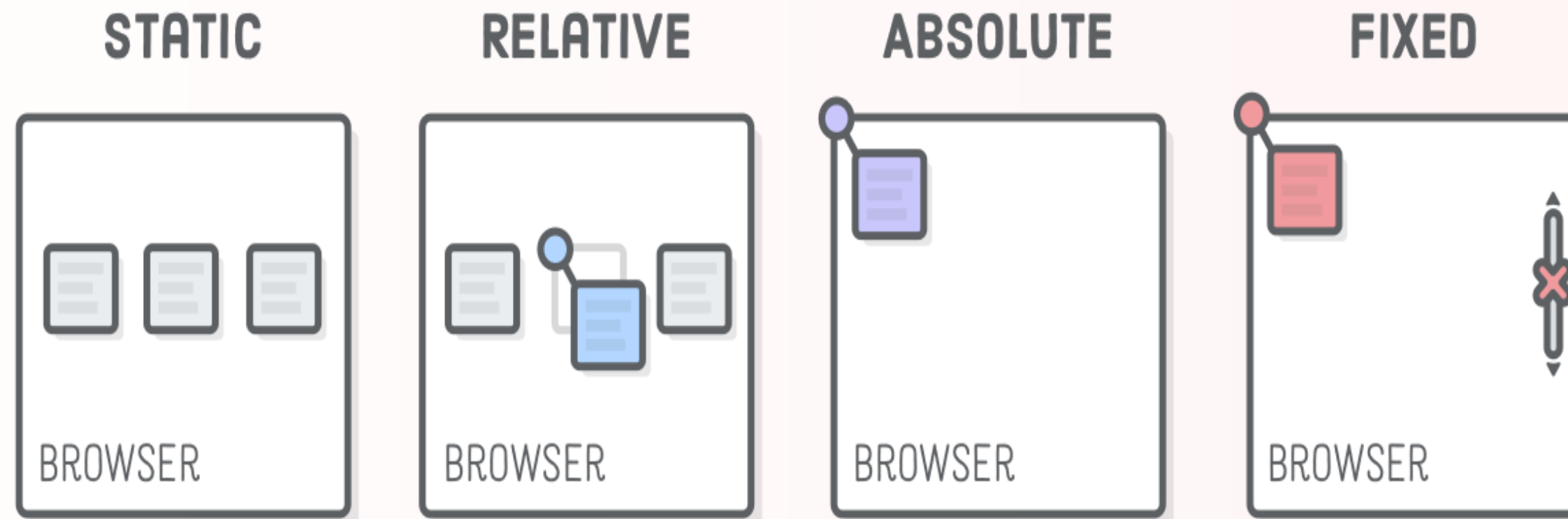
Khi dùng display: grid;, chúng ta sẽ được một container chứa grid dưới dạng block

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49		9.3				
			61		10.2				
	15		62	10.1	10.3				4
11	16	57	63	11	11.2	all	62	11.4	6.2
	17	58	64	TP					
		59	65						
		60	66						



## POSITION AND Z-INDEX

Thuộc tính position trong CSS dùng để xác định vị trí hiển thị cho thẻ HTML và thường được dùng để xây dựng CSS cho menu đa cấp, tooltip hoặc một số chức năng khác.



## Thuộc tính Position Static

Thuộc tính `position: static` được xem là giá trị mặc định (default) của position.

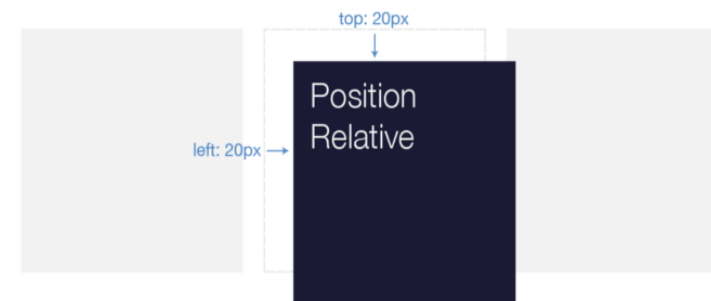


Đây là các giá trị mà dù bạn có khai báo chúng hay không khai báo thì các phần tử (element) sẽ tự được sắp xếp vị trí một cách như bình thường trên trang web.

## Thuộc tính Position Relative

`position: relative` trong CSS giúp Định vị trí tuyệt đối cho các thành phần, sử dụng kèm theo với các thuộc tính căn chỉnh tọa độ của thành phần

```
selector {  
  position: relative;  
}
```



**Top:** là cách trên, **Bottom:** là cách dưới, **Left:** là cách trái, **Right:** là cách phải.

**Note:** các giá trị của các thẻ này là chính là đơn vị đo như px.

## Thuộc tính Position Absolute

`position: absolute` trong CSS có tác dụng giúp định vị trí tuyệt đối cho thành phần theo thành phần bao ngoài, hoặc ít nhất là theo cửa sổ trình duyệt.

```
.absolute {  
  position: absolute;  
  left: 260px;  
  top: 290px;  
}
```

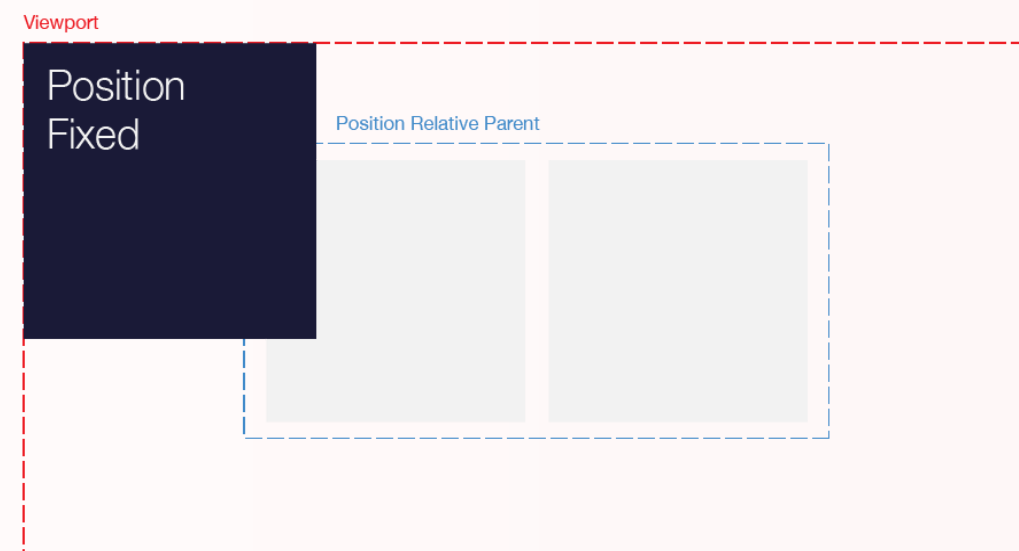
Position Relative Parent



## Thuộc tính Position Fixed

- o dùng để định vị một thành phần so với window hiển thị của các trình duyệt

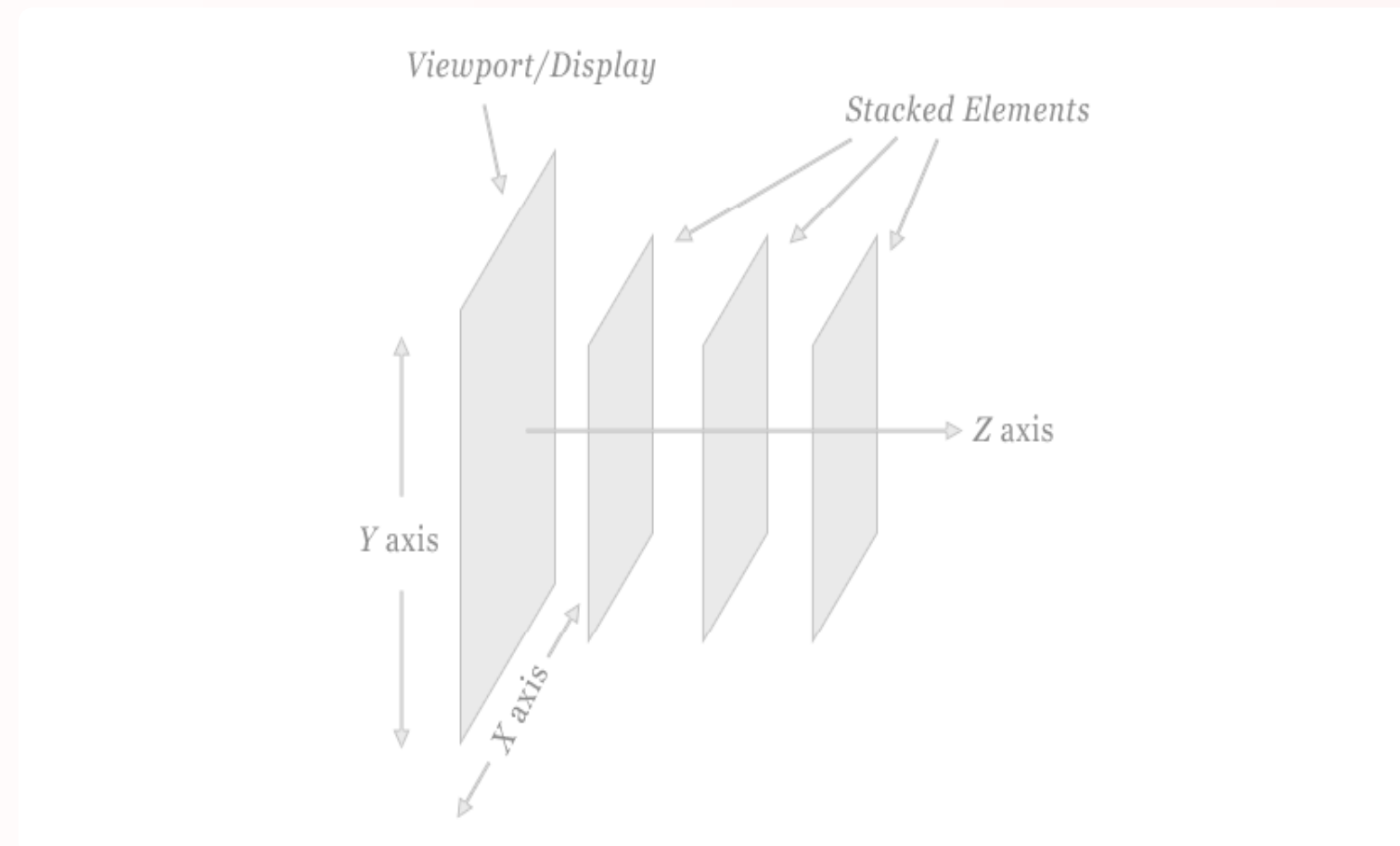
```
.fixed {  
  position: fixed;  
}
```



- o Giá trị này hoàn toàn không phụ thuộc vào phần tử cha, khi nào scroll trình duyệt là nó hoạt động thôi


## Z-index

- z-index trong css có cách thức hoạt động là mỗi element trên trang web được hiển thị ngang và dọc theo 2 trục x và y, hiển thị thứ tự chồng lấn theo trục z.
- z-index càng cao thì element đó đứng trước và hiện lên trên.




## EXAMPLE

### Hello, we have brought together the best quality services, offers, projects for you!




#### Outstanding design

Designed to be flexible according to all your needs. Create your site with all module position.



#### Responsive Layout

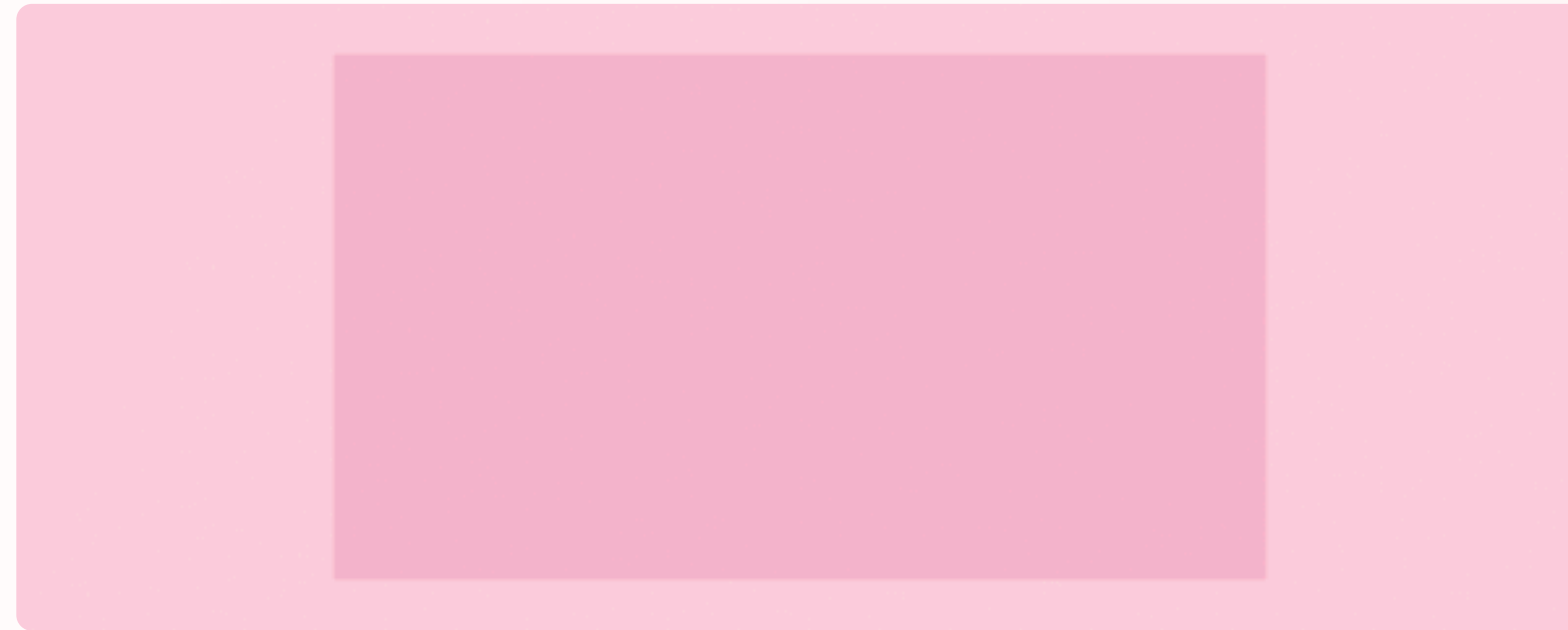
Genius template can be easily viewed on any mobile device. Smoothly responsive.



#### Easy to use

The modules we have prepared are simple to use. No code information is needed.

## CSS TRANSITION



CSS transitions cung cấp một cách để điều khiển tốc độ của hiệu ứng khi thay đổi các thuộc tính của CSS. Thay vì, các thay đổi thuộc tính tạo ra ảnh hưởng ngay lập tức, bạn có thể làm cho các thay đổi này diễn ra trong một khoảng thời gian.

2 thuộc tính cơ bản nhất để visualize quá trình chuyển đổi

- transition-duration
- transition-property



## TRANSITION OPTIONS

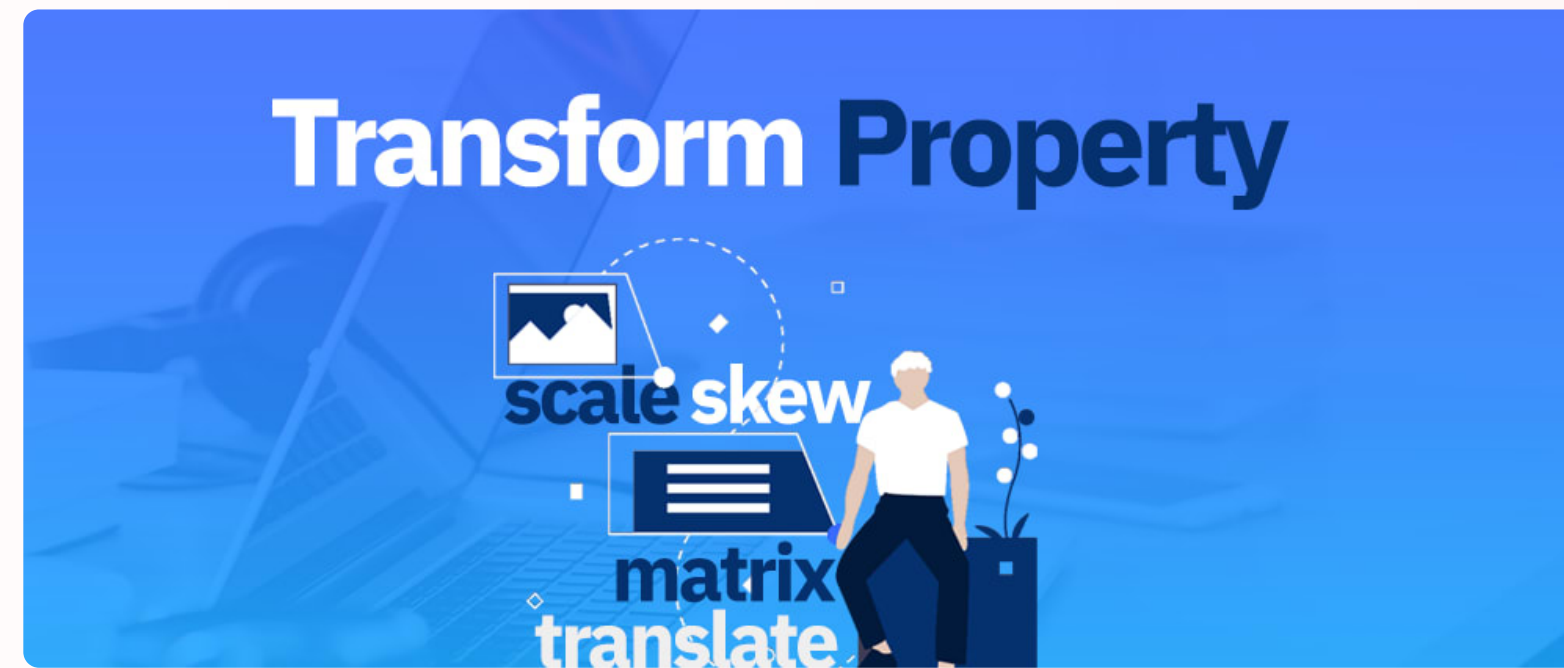
### transition-timing-function:

- ease: thay đổi bắt đầu chậm, sau đó tăng dần vận tốc, rồi chậm lại và dừng.
- linear: vận tốc không thay đổi trong cả quá trình đổi trạng thái.
- ease-in: Vận tốc tăng chậm từ 0, tăng dần đến khi cán đích.
- ease-out: Vận tốc lúc đầu lớn, sau đó chậm lại trước khi cán đích.
- ease-in-out: Bắt đầu chuyển động chậm rãi, rồi tăng tốc ở giai đoạn giữa, sau đó lại chậm lại.
- cubic-bezier: Xác định giá trị cho hiệu ứng của quá trình chuyển đổi theo từng giai đoạn, giá trị xác định chỉ có thể từ 0 tới 1.

**transition-delay:** Đây là thuộc tính quyết định độ trễ của hành động (đơn vị: m hoặc ms)

```
div {  
  transition: property duration timing-function delay;  
}
```

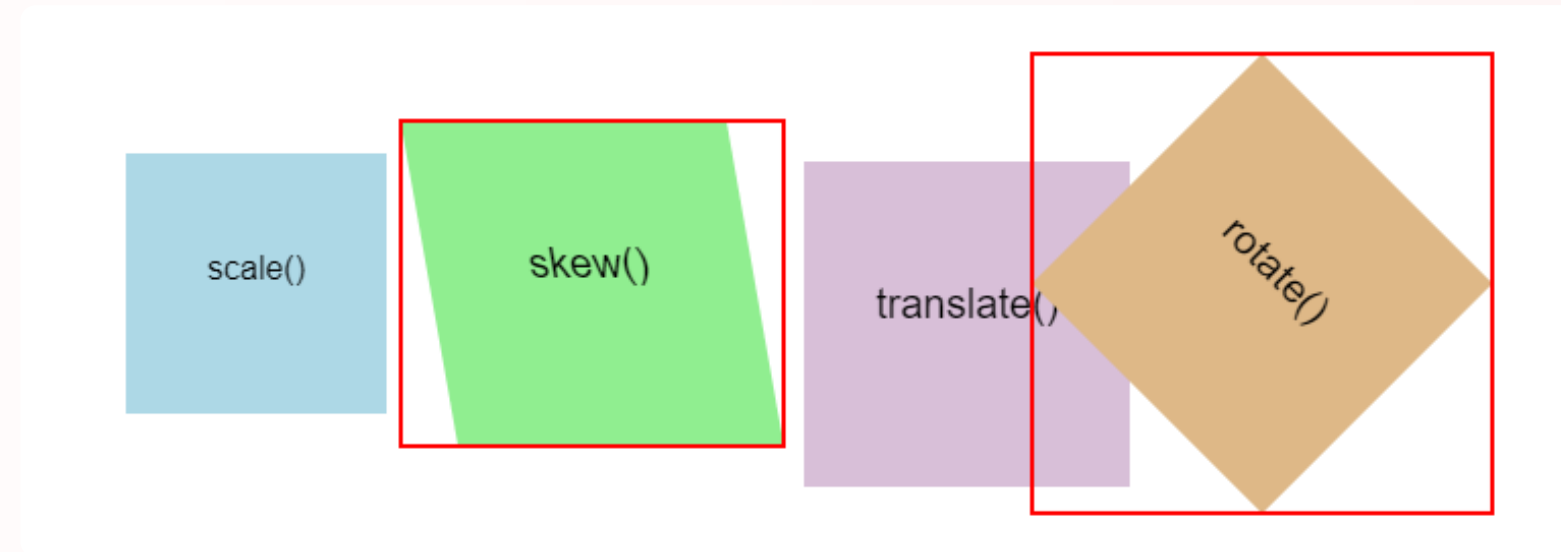
## TRANSFORMS



Thuộc tính transforms là thuộc tính được sử dụng để xác định các chuyển đổi 2 chiều, 3 chiều, có thể là chuyển đổi xoay, thay đổi tỉ lệ, hình dạng, di chuyển,... Transformation là cách gọi chung của hiệu ứng thay đổi hình dạng, kích thước và vị trí của phần tử.

Các dạng transforms: **2D transforms** và **3D transforms**.

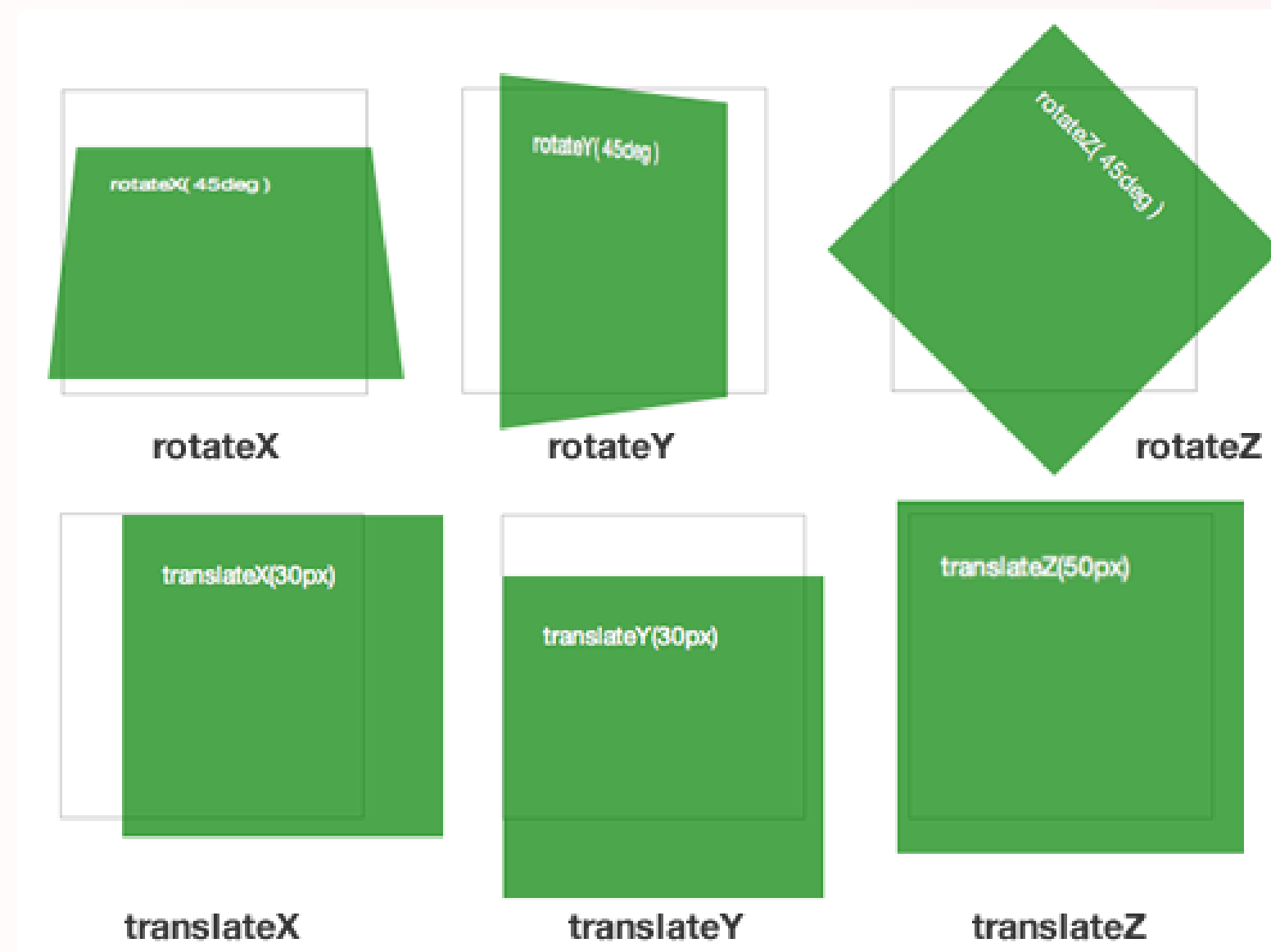
## 2D TRANSFORMS



- Scale: Cho phép thay đổi kích thước đối tượng
- Translate: Điều khiển sẽ thay đổi vị trí của đối tượng
- Rotate: Xoay đối tượng theo góc bất kỳ
- Skew: kéo dãn dài tượng theo trục x hoặc y
- Transform origin: Transform origin xác định tâm điểm mà thuộc tính transform áp dụng lên.

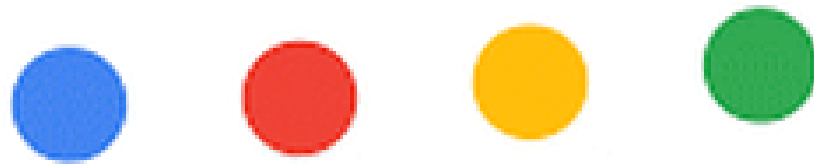
## 3D TRANSFORM

3D Transform là những thuộc tính dùng để xử lý hiệu ứng di chuyển 3D cho các phần tử theo 3 trục: x, y và z. Các giá trị thường dùng với 3D transform là: rotateX() rotateY() rotateZ()



## KEYFRAMES & ANIMATION

- CSS animation là công nghệ được giới thiệu trong phiên bản CSS3. Nó cho phép chúng ta tạo hiệu ứng chuyển động mà không phải sử dụng Javascript hay Flash



## Keyframes

Keyframes là yếu tố cấu thành nên CSS animations. Nó định nghĩa hiệu ứng sẽ trông ra sao tại mỗi thời điểm trong dải thời gian của hiệu ứng. Bao gồm

- Tên của hiệu ứng
- Các mốc thời gian của hiệu ứng
- Thuộc tính CSS

### Cú Pháp

```
@keyframes Name {  
  // code  
}
```

- Name: là tên của chuyển động.
- Code: là các đoạn code cho tiến trình chuyển động.

## Keyframes

```
@keyframes pacman {  
  from {  
    //something here  
  }  
  to {  
    // something else  
  }  
}
```

- from: được tính là 0%
- to: 100%

Keyframes xác định quá trình thay đổi tại các thời điểm mong muốn của hiệu ứng. Giữa các thời điểm, đối tượng sẽ thay đổi trạng thái dần dần từ thời điểm trước đến thời điểm sau.

## Animation

Thuộc tính animation có 2 nhiệm vụ:

- Gán hành động được định nghĩa ở @keyframes cho đối tượng mà bạn muốn thực hiện hành động.
- Định nghĩa cách diễn ra hành động.
  - **animation-name**
  - **animation-duration**

```
img {  
  animation-name: pacman;  
  animation-duration: 3s;  
}
```

hoặc rút gọn lại: `animation: pacman 3s;`



## ANIMATION OPTIONS

Các thuộc tính con của animation:

- **animation-name:** pacman;
- **animation-duration:** 3s;
- **animation-delay:** 2s;
- **animation-timing-function:** linear; ``ease` (default value), linear, ease-in, ease-out, ease-in-out, initial, inherit, cubic-bezier``
- **animation-iteration-count:** 1;
- **animation-direction:** normal; ``(normal, reverse, alternate, alternate-reverse)``
- **animation-play-state:** running; ``(running, pause)``
- **animation-fill-mode:** forwards; ``(normal , backwards, forwards, both)``

## Short syntax & Đa hiệu ứng

Có thể viết ngắn gọn các thuộc tính con:

**animation: [animation-name] [animation-duration] [animation-timing-function] [animation-delay]**

**[animation-iteration-count] [animation-direction] [animation-fill-mode] [animation-play-state];**

Để thêm cùng lúc nhiều hiệu ứng đã được định nghĩa keyframes, thì chỉ cần ngăn cách các define của mỗi hiệu ứng bởi dấu phẩy như sau:

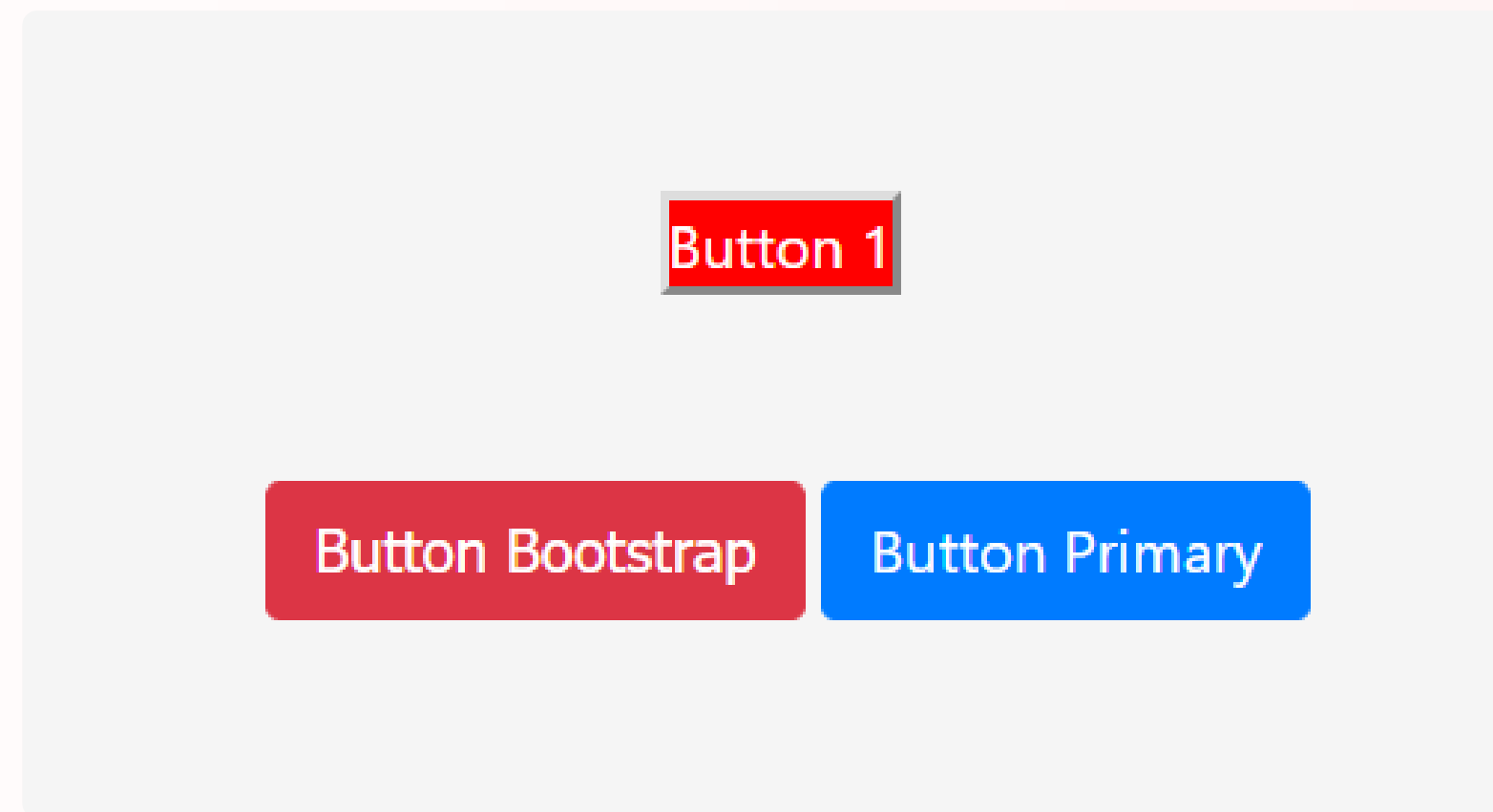
```
.div {  
  animation: pacman 3s, disapear 4s;  
}
```

## BOOTSTRAP LÀ GÌ?

- Một framework của css và javascript. Hay nói cách khác, nó là những đoạn code viết sẵn.
- Định nghĩa sẵn các class css và các hàm trong javascript. Chúng ta chỉ việc sử dụng các class này để tùy chỉnh thêm

Ưu điểm của bootstrap

- Giảm thiểu bớt việc viết code, đóng vai trò như bộ khung nền, giúp phát triển web nhanh hơn.



## TÍCH HỢP CSS VÀO WEBSITE

- Tải về và dùng trực tiếp local
  - Tải file source của Bootstrap tại [getbootstrap.com](https://getbootstrap.com)
  - Thêm thẻ link, chỉ đường dẫn đến file bootstrap.css hoặc bootstrap.min.css

```
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
```

- Thêm thẻ script vào cuối thẻ body.

```
<script type="text/javascript" src="bootstrap/js/bootstrap.js"></script>
```

- Có thể sử dụng CDN (Content Delivery Network) thêm các dòng sau vào trước thẻ `</head>` của website:

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" crossorigin="anonymous">
```

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" crossorigin="anonymous"></script>
```

## RESPONSIVE META TAG

```
`<meta name="viewport" content="width=device-width, initial-scale=1">`
```

Để đảm bảo hiển thị phù hợp và phóng to chạm cho tất cả các thiết bị, thêm thẻ meta chế độ xem đáp ứng vào `<head>`.

## MÃ CODE ĐƠN GIẢN

```
<body>
  <div class="container">
    <div class="row">
      // example bootstrap...
    </div>
  </div>
</body>
```

## BÀI TẬP THỰC HÀNH

Thiết kế design html cho trang web:

<http://mauweb.monamedia.net/anphuoc/>