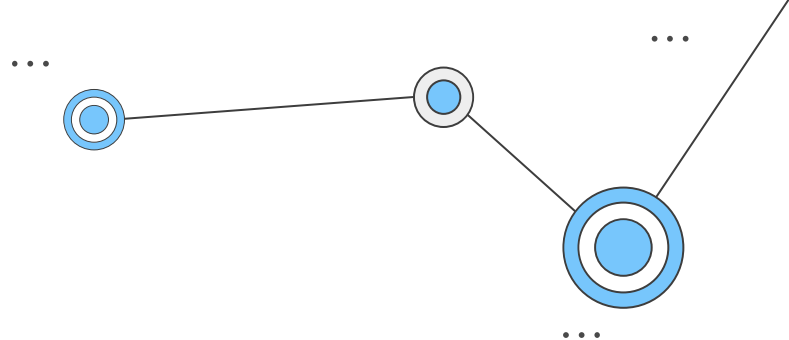
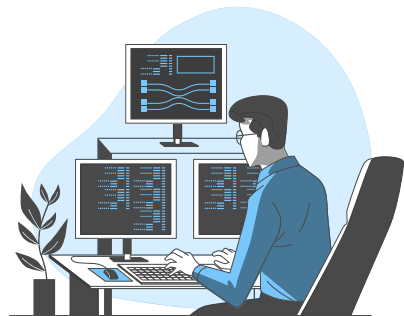


CSS



Khóa học Frontend

Bài 04: Học CSS3 (Tiết 1)



Nội dung

01

Khái niệm, cú pháp

...

02

Ba cách chèn CSS

...

03

Color (Màu sắc)

...

04

Background (Nền)

...

05

Text

...

06

Fonts

...

07

Icons

...

08

Opacity (Độ mờ)

...

09

9. Các loại Selectors
(Bộ chọn)

...

10

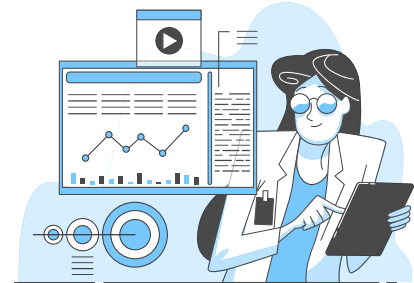
Simple selectors
(Bộ chọn đơn giản)

...

11

Combinator selectors
(Bộ chọn tổ hợp)

...



01

Khái niệm, cú pháp

01. Khái niệm, cú pháp

- **Khái niệm**

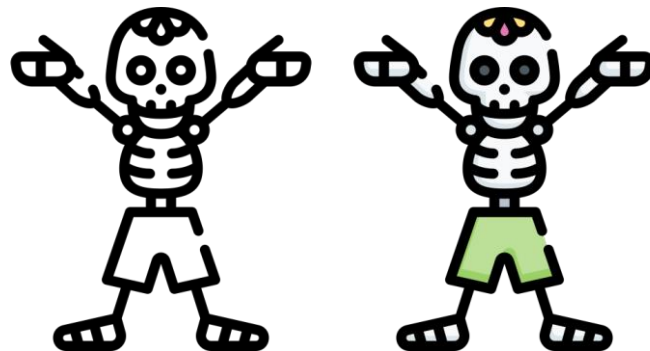
- **CSS**: viết tắt của **C**ascading **S**tyle **S**heets.
- Là ngôn ngữ được dùng để định dạng kiểu hiển thị cho các phần tử HTML.

- **Cú pháp**

```
selector {  
  property: value;  
}
```

- **Trong đó:**

- **selector**: Được gọi là bộ chọn.
- **property**: Được gọi là thuộc tính.
- **value**: Được gọi là giá trị của property.



HTML

CSS

02

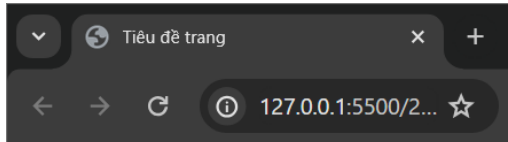
Ba cách chèn CSS

02. Ba cách chèn CSS

- **Inline CSS (Nội tuyến)**

- Thêm thuộc tính **style** vào trong thẻ mở HTML.
- Ví dụ:

```
<h2 style="color: red;">Nội dung thẻ h2</h2>  
<h3 style="color: green;">Nội dung thẻ h3</h3>
```



Nội dung thẻ h2

Nội dung thẻ h3

02. Ba cách chèn CSS

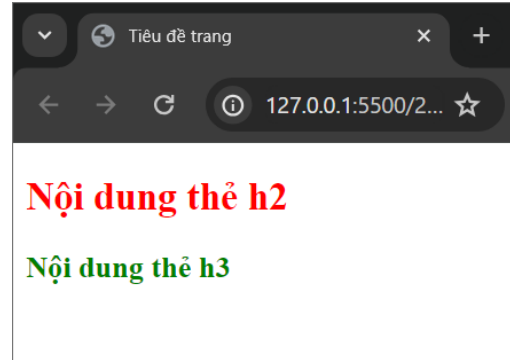
- **Internal CSS (Nội bộ)**

- Thêm thẻ **<style></style>** và bất cứ đâu trong file HTML.
- Viết CSS vào trong thẻ **<style>** đó.
- Ví dụ:

```
<style>
  h2 {
    color: red;
  }

  h3 {
    color: green;
  }
</style>

<h2>Nội dung thẻ h2</h2>
<h3>Nội dung thẻ h3</h3>
```

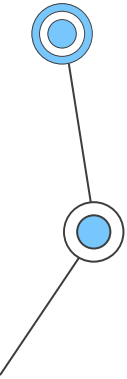
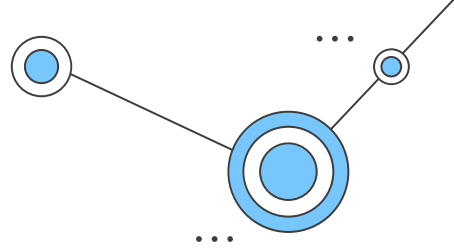


02. Ba cách chèn CSS

- **External CSS (Bên ngoài):**

- Tạo 1 file CSS ở bên ngoài.
- Chèn file CSS vào HTML:

```
<head>  
  <title>Tiêu đề trang</title>  
  <link rel="stylesheet" href="link-file-css" />  
</head>
```





03

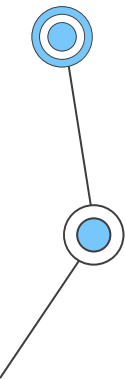
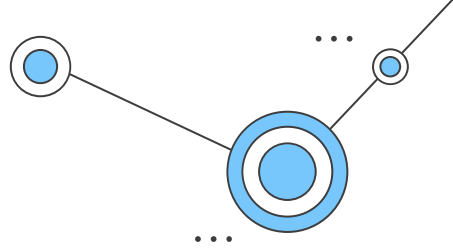
Color (Màu sắc)



03. Color (Màu sắc)

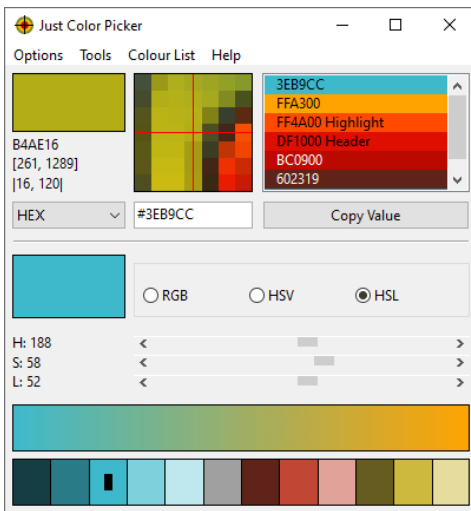
- Có 2 cách lấy màu sắc phổ biến:
 - Màu đặt tên sẵn: white, black, red, green, blue, yellow, orange,...
 - Màu dạng mã **HEX** (*Phổ biến nhất*)
- Màu dạng mã **HEX**
 - Cú pháp: **#RRGGBB**
 - Trong đó
 - RR: Red (đỏ)
 - GG: Green (xanh lục)
 - BB: Blue (xanh dương)
- Ví dụ:

```
h2 {  
  color: #FF0000;  
}
```



03. Color (Màu sắc)

- Cài đặt phần mềm **Just Color Picker**
 - Dùng để lấy mã màu nhanh.
 - Link cài đặt: <https://annystudio.com/software/colorpicker/>
 - Cách dùng:
 - Bước 1: Di chuột vào vị trí màu muốn lấy
 - Bước 2: Bấm **Alt + X** để lấy mã màu





04

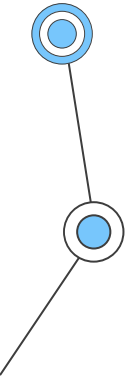
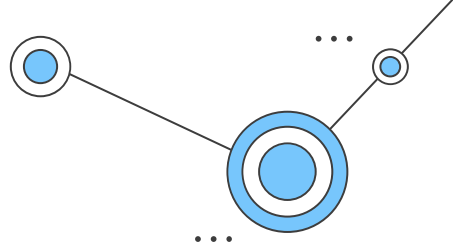
Background (Nền)



04. Background (Nền)

- Background (nền) dùng để thêm nền vào cho một phần tử.
- Thuộc tính **background-color**
 - Thêm nền là **màu sắc**.
 - Cú pháp:

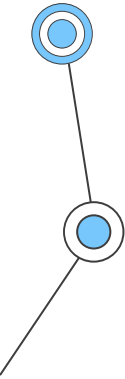
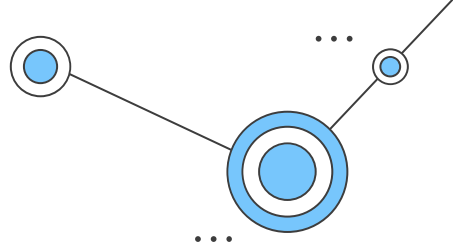
```
selector {  
  background-color: mã màu;  
}
```



04. Background (Nền)

- Background (nền) dùng để thêm nền vào cho một phần tử.
- Thuộc tính **background-image**
 - Thêm nền là **hình ảnh**.
 - Cú pháp:

```
selector {  
  background-image: url("duong-dan-anh");  
}
```



04. Background (Nền)

- Background (nền) dùng để thêm nền vào cho một phần tử.

- Thuộc tính **background-size**

- Để thiết lập **kích thước** của hình nền.
- Cú pháp:

```
selector {  
  background-size: giá trị;  
}
```

Có 2 giá trị:

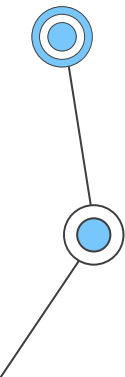
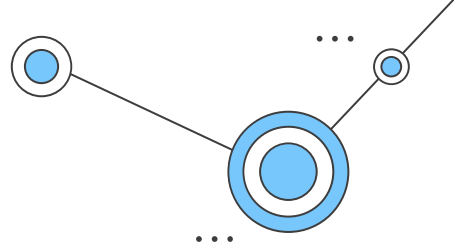
- + **contain**: hình nền vừa với khung, có thừa khoảng trắng
- + **cover**: cắt ảnh để cho vừa khối (bỏ đi phần thừa của ảnh)

nếu để no-repeat thì ảnh sẽ bị khoảng trắng

có thể đặt giá trị ví dụ
background-size: 300px 200px;



không có background repeat thì ảnh sẽ bị lặp lại

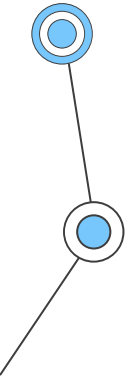
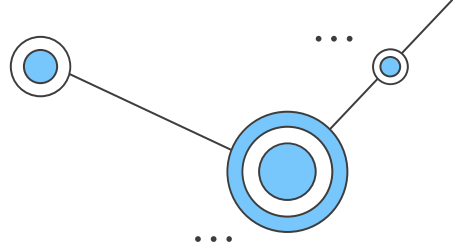


04. Background (Nền)

- Background (nền) dùng để thêm nền vào cho một phần tử.
- Thuộc tính **background-repeat**
 - Để thiết lập **hình nền có lặp lại hay không**.
 - Cú pháp:

```
selector {  
  background-repeat: giá trị;  
}
```

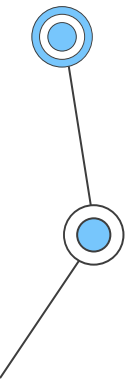
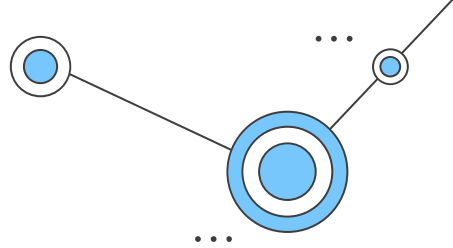
giá trị để không lặp lại ảnh là
+ no-repeat




04. Background (Nền)

- Background (nền) dùng để thêm nền vào cho một phần tử.
- Thuộc tính **background-position**
 - Để thiết lập **vị trí** của hình nền so với phần tử.
 - Cú pháp:

```
selector {  
  background-position: giá trị;  
}
```





05

Text

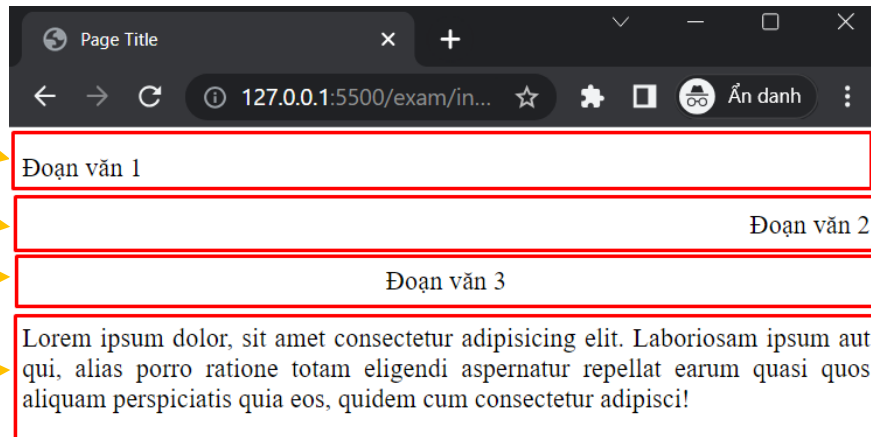


05. Text

- Thuộc tính **text-align**
 - Để **căn lề** cho văn bản.
 - Cú pháp:

```
selector {  
  text-align: giá trị;  
}
```

```
p {  
  text-align: left;  
}  
p {  
  text-align: right;  
}  
p {  
  text-align: center;  
}  
p {  
  text-align: justify;  
}
```



05. Text

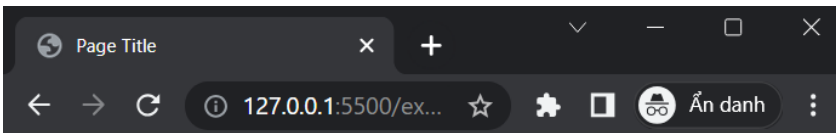
- Thuộc tính **text-transform**
 - Để định dạng **viết hoa, viết thường** cho văn bản.
 - Cú pháp:

```
selector {  
  text-transform: giá trị;  
}
```

```
p {  
  text-transform: uppercase;  
}
```

```
p {  
  text-transform: capitalize;  
}
```

```
p {  
  text-transform: lowercase;  
}
```



ĐOẠN VĂN 1

Đoạn Văn 2

đoạn văn 3



06

Fonts

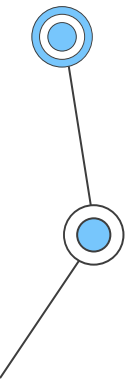
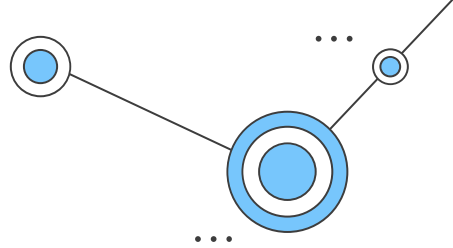


06. Fonts

- Thuộc tính **font-family**
 - Để chỉ định **loại font chữ** được sử dụng cho văn bản.
 - Cú pháp:

```
selector {  
  font-family: giá trị;  
}
```

- Ví dụ: <https://fonts.google.com>



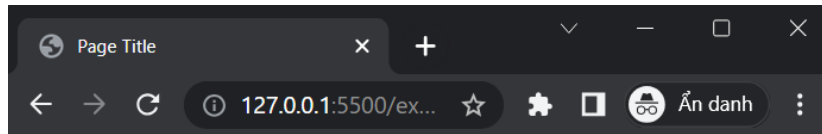
06. Fonts

- Thuộc tính **font-style**
 - Để chỉ định văn bản **có in nghiêng hay không**.
 - Cú pháp:

```
selector {  
  font-style: giá trị;  
}
```

```
p {  
  font-style: normal;  
}
```

```
p {  
  font-style: italic;  
}
```



Đoạn văn 1

Đoạn văn 2

06. Fonts

- Thuộc tính **font-weight**

- Để chỉ định độ dày của văn bản.
- Các giá trị: 100, 200, 300, 400, 500, 600, 700, 800, 900.
- Cú pháp:

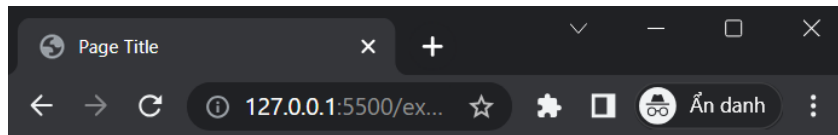
```
selector {  
  font-weight: giá trị;  
}
```

```
p {  
  font-weight: 400;  
}  
p {  
  font-weight: 600;  
}  
p {  
  font-weight: 900;  
}
```

Đoạn văn 1

Đoạn văn 2

Đoạn văn 3

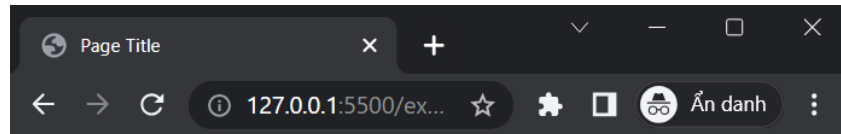


06. Fonts

- Thuộc tính **font-size**
 - Để chỉ định **kích cỡ** của văn bản.
 - Cú pháp:

```
selector {  
  font-size: giá trị;  
}
```

```
p {  
  font-size: 12px;  
}  
p {  
  font-size: 16px;  
}  
p {  
  font-size: 20px;  
}
```



Đoạn văn 1

Đoạn văn 2

Đoạn văn 3

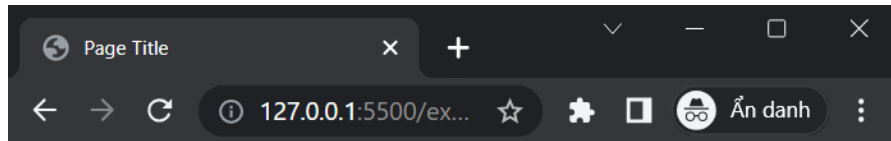
07


Icons


07. Icons

- Link trang chủ Font Awesome: <https://fontawesome.com/search?o=r&m=free>
- Link web để lấy mã nhúng: <https://cdnjs.com/libraries/font-awesome>

```
<p><i class="fa-brands fa-facebook"></i> Facebook</p>  
<p><i class="fa-brands fa-twitter"></i> Twitter</p>  
<p><i class="fa-brands fa-instagram"></i> Instagram</p>
```



 Facebook

 Twitter

 Instagram



08

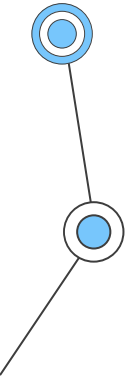
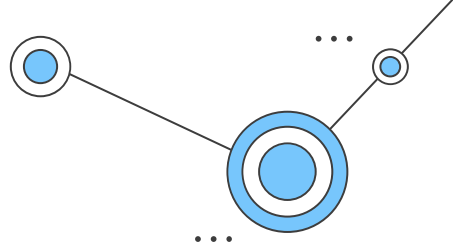
Opacity (Độ mờ)



08. Opacity (Độ mờ)

- Để chỉ định **độ mờ** của một phần tử.
- Giá trị từ 0 đến 1.
- Giá trị càng nhỏ thì phần tử càng mờ.
- Cú pháp:

```
selector {  
  opacity: giá trị;  
}
```

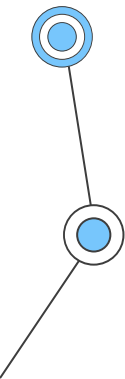
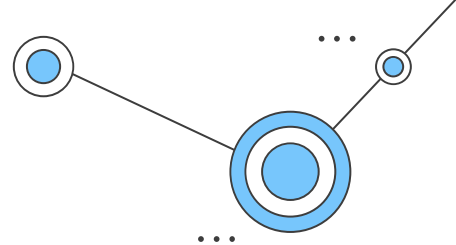


09

Các loại Selectors (Bộ chọn)

09. Các loại Selectors (Bộ chọn)

- Bộ chọn dùng để chọn đúng các phần tử HTML đang cần thêm CSS.
- Bộ chọn được chia thành 5 nhóm:
 - Simple selectors (Bộ chọn đơn giản)
 - Combinator selectors (Bộ chọn tổ hợp)
 - Pseudo-elements selectors (Bộ chọn phần tử giả)
 - Pseudo-class selectors (Bộ chọn lớp giả)
 - Attribute selectors (Bộ chọn thuộc tính)



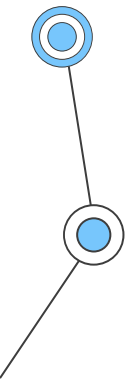
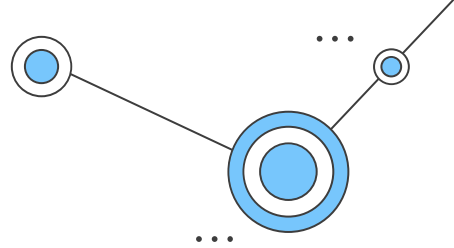
10

**Simple selectors
(Bộ chọn đơn giản)**

10. Simple selectors (Bộ chọn đơn giản)

- **Bộ chọn phần tử** (element selector)
 - Chọn các phần tử HTML dựa trên **tên phần tử**.
 - Ví dụ:

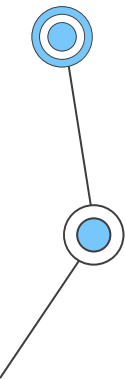
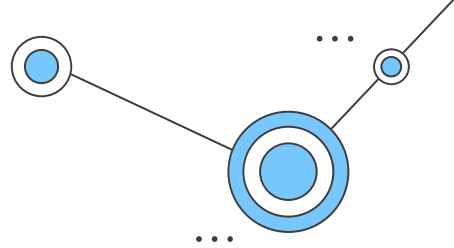
```
p {  
  text-align: center;  
  color: red;  
}
```



10. Simple selectors (Bộ chọn đơn giản)

- **Bộ chọn class** (class selector)
 - Chọn các phần tử HTML dựa trên **thuộc tính class**.
 - Ví dụ:

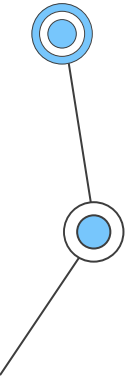
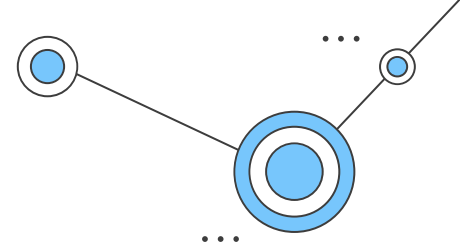
```
.title {  
  text-align: center;  
  color: red;  
}
```



10. Simple selectors (Bộ chọn đơn giản)

- **Bộ chọn nhóm** (grouping selector)
 - Chọn các phần tử HTML dựa trên **một nhóm**.
 - Ví dụ:

```
h1, h2, p {  
  text-align: center;  
  color: red;  
}
```

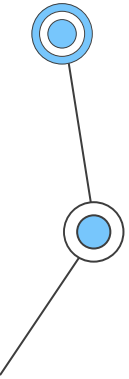
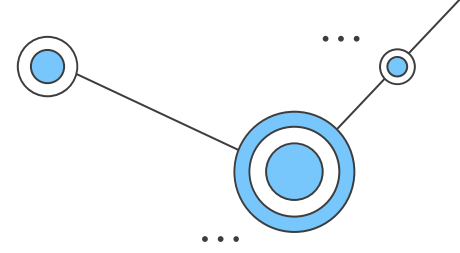
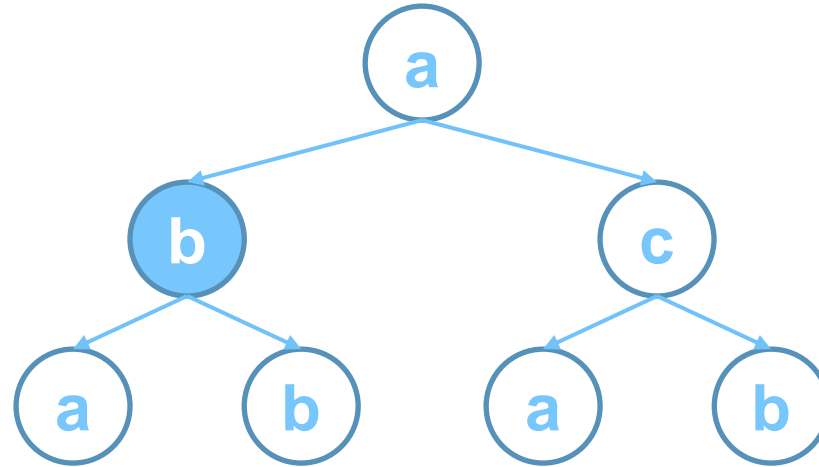


11

Combinator selectors (Bộ chọn tổ hợp)

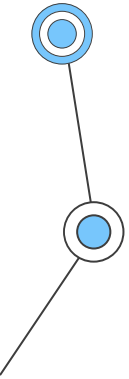
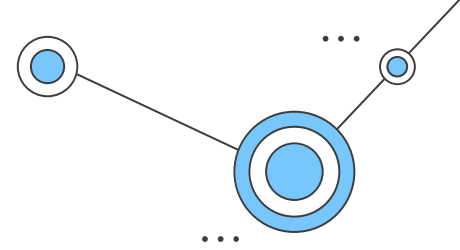
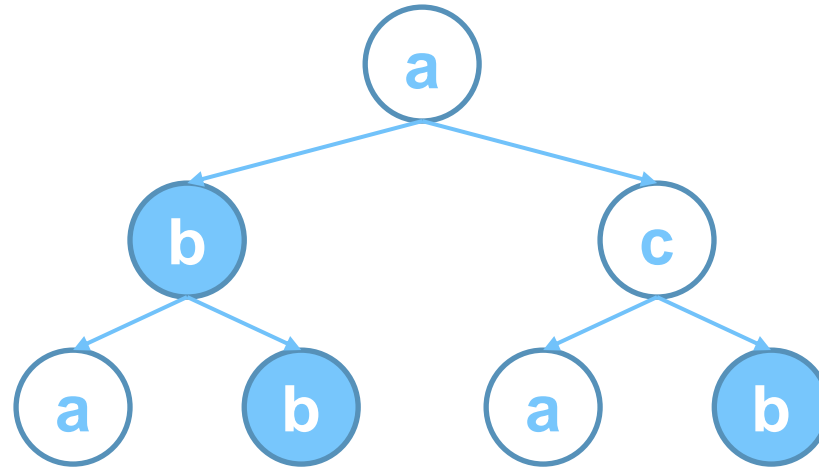
11. Combinator selectors (Bộ chọn tổ hợp)

- **Bộ chọn con** (Child selectors)
 - Chọn tất cả phần tử **b** là cấp con đầu tiên (cấp 1) của phần tử **a**.
 - Cú pháp: **a > b { }**

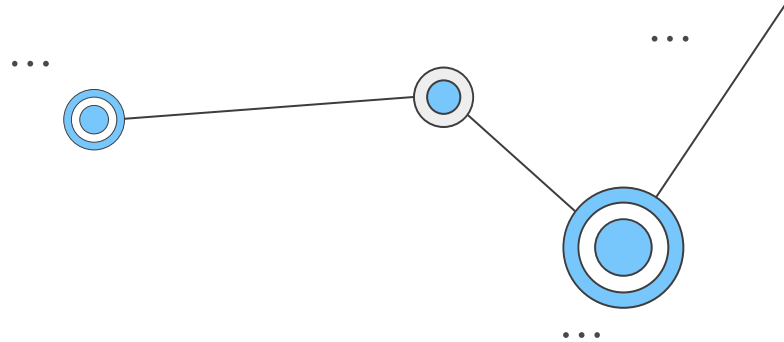


11. Combinator selectors (Bộ chọn tổ hợp)

- **Bộ chọn hậu duệ** (Descendant selectors)
 - Chọn tất cả phần tử **b** bên trong phần tử **a**.
 - Cú pháp: **a b { }**

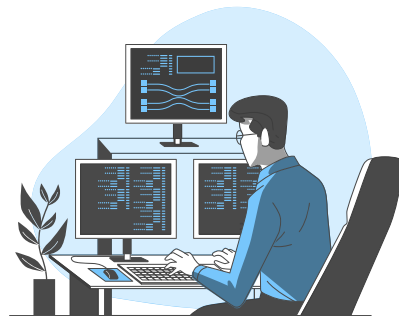


CSS



Khóa học Frontend

Bài 05: Học CSS3 (Tiết 2)



Nội dung

01

Pseudo-elements selectors
(Bộ chọn phần tử giả)

02

Pseudo-class selectors
(Bộ chọn lớp giả)

03

Attribute selectors
(Bộ chọn thuộc tính)

04

Specificity và !important

05

Box Model, Content,
Padding, Border, Margin

06

Thuộc tính width, height

07

Box Sizing (Kích thước hộp)

08

Đơn vị chiều dài

09

Overflow (Tràn ra)

10

Display (Hiển thị)



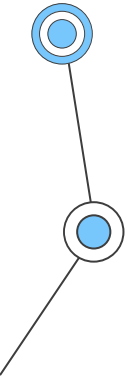
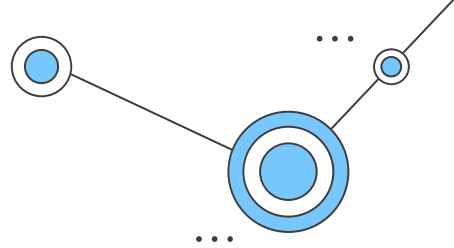
01

Pseudo-elements selectors (Bộ chọn phần tử giả)

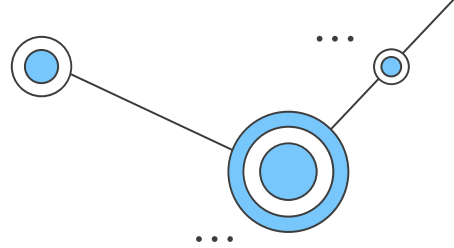
01. Pseudo-elements selectors (Bộ chọn phần tử giả)

- Được sử dụng để tạo ra một phần tử giả và CSS cho phần tử giả đó mà không cần tạo ra một phần tử thật.
- Cú pháp:

```
selector::pseudo-element {  
  property: value;  
}
```

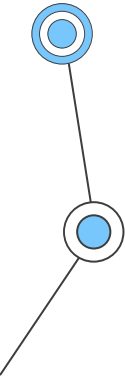


01. Pseudo-elements selectors (Bộ chọn phần tử giả)



- Một số Pseudo-elements phổ biến:
 - **::before**
 - Tạo một phần tử mới ở bên trong phần tử cha, và đứng ở vị trí đầu tiên.
 - Cú pháp:

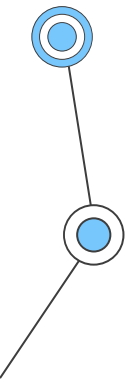
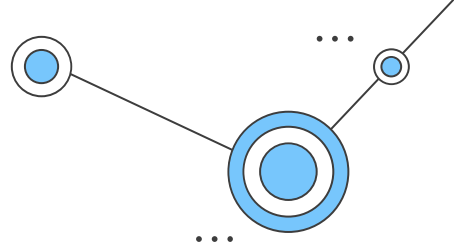
```
selector::before {  
  property: value;  
}
```



01. Pseudo-elements selectors (Bộ chọn phần tử giả)

- Một số Pseudo-elements phổ biến:
 - **::after**
 - Tạo một phần tử mới ở bên trong phần tử cha, và đứng ở vị trí cuối cùng.
 - Cú pháp:

```
selector::after {  
  property: value;  
}
```



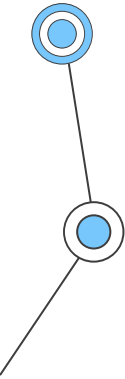
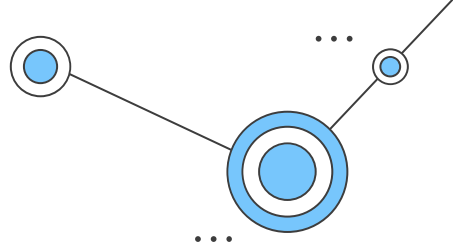
02

Pseudo-class selectors (Bộ chọn lớp giả)

02. Pseudo-class selectors (Bộ chọn lớp giả)

- Dùng để xác định trạng thái đặc biệt của một phần tử.
- Cú pháp:

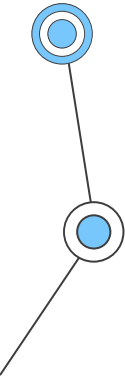
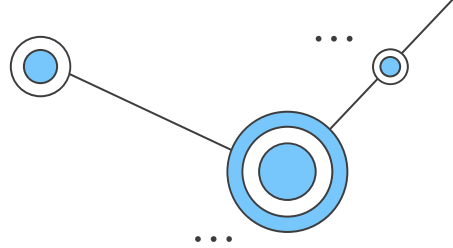
```
selector:pseudo-class {  
  property: value;  
}
```



02. Pseudo-class selectors (Bộ chọn lớp giả)

- Một số Pseudo-class phổ biến:
 - **:hover**
 - Chọn phần tử khi người dùng di chuột qua nó.
 - Cú pháp:

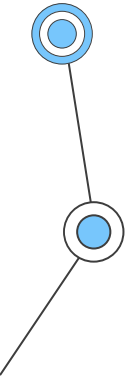
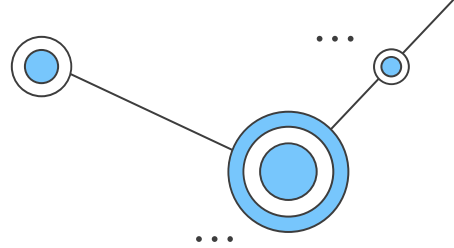
```
selector: hover {  
  property: value;  
}
```



02. Pseudo-class selectors (Bộ chọn lớp giả)

- Một số Pseudo-class phổ biến:
 - **:first-child**
 - Chọn phần tử con đầu tiên ở bên trong phần tử cha.
 - Cú pháp:

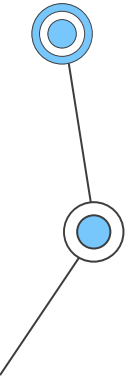
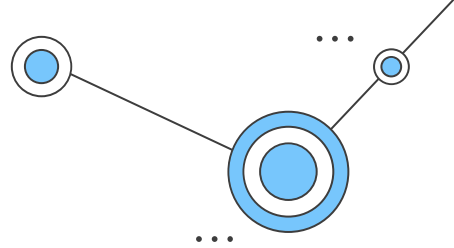
```
selector:first-child {  
  property: value;  
}
```



02. Pseudo-class selectors (Bộ chọn lớp giả)

- Một số Pseudo-class phổ biến:
 - **:last-child**
 - Chọn phần tử con cuối cùng ở bên trong phần tử cha.
 - Cú pháp:

```
selector:last-child {  
  property: value;  
}
```

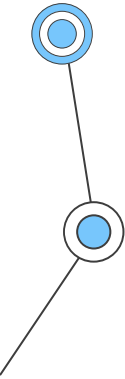
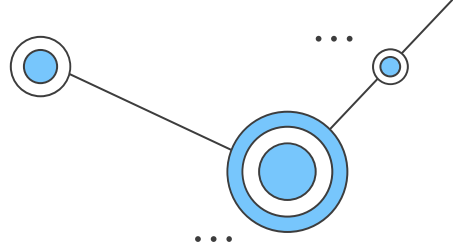


02. Pseudo-class selectors (Bộ chọn lớp giả)

- Một số Pseudo-class phổ biến:
 - **:nth-child(an + b)**
 - Chọn phần tử con có vị trí cụ thể ở bên trong phần tử cha.
 - Cú pháp:

```
selector:nth-child(an + b) {  
  property: value;  
}
```

- Trong đó:
 - a, b là các số biết trước
 - n là số nguyên dương tăng dần 0, 1, 2, ...

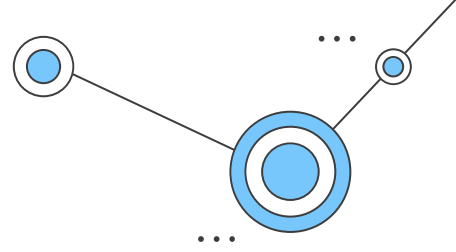


03

Attribute selectors (Bộ chọn thuộc tính)

03. Attribute selectors (Bộ chọn thuộc tính)

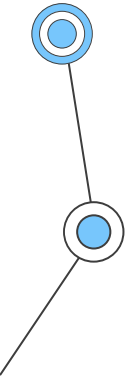
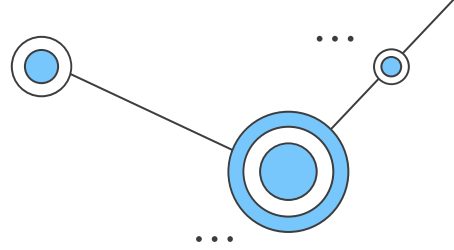
- Dùng để chọn các phần tử HTML dựa trên thuộc tính của phần tử đó.



03. Attribute selectors (Bộ chọn thuộc tính)

- Một số cách chọn phổ biến:
 - **[attribute]**
 - Chọn tất cả các phần tử có thuộc tính **attribute**.
 - Cú pháp:

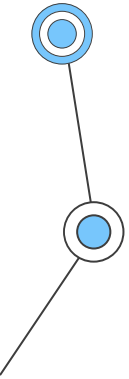
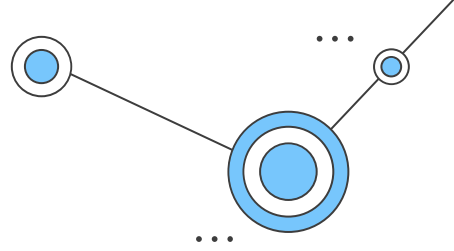
```
selector:[attribute] {  
  property: value;  
}
```



03. Attribute selectors (Bộ chọn thuộc tính)

- Một số cách chọn phổ biến:
 - **[attribute="value"]**
 - Chọn tất cả các phần tử có thuộc tính **attribute** có giá trị bằng **value**.
 - Cú pháp:

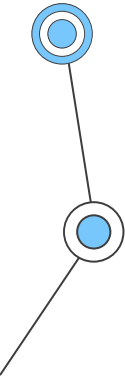
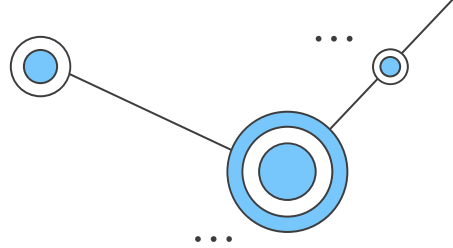
```
selector:[attribute="value"] {  
  property: value;  
}
```



03. Attribute selectors (Bộ chọn thuộc tính)

- Một số cách chọn phổ biến:
 - **[attribute*="value"]**
 - Chọn tất cả các phần tử có thuộc tính **attribute** chứa giá trị **value**.
 - Cú pháp:

```
selector:[attribute*="value"] {  
  property: value;  
}
```





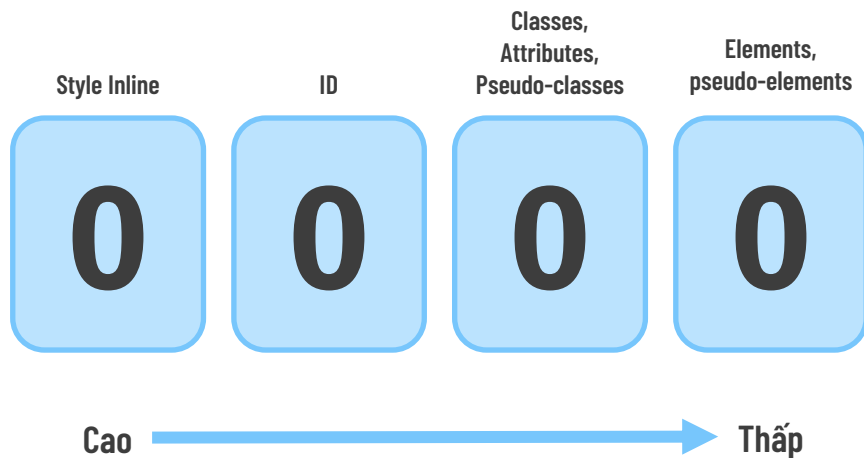
04

**Specificity
và !important**

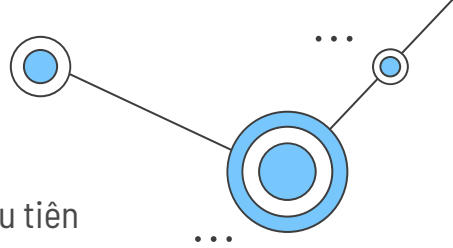


4.1. Specificity (Tính đặc hiệu - Độ ưu tiên)

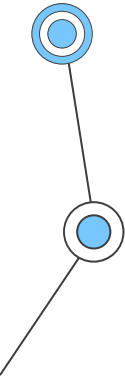
- **Độ ưu tiên** trong CSS là cách mà trình duyệt quyết định sẽ **ưu tiên thuộc tính CSS** nào với một phần tử **khi có nhiều bộ chọn CSS cùng trỏ đến phần tử HTML**.



4.2. Cú pháp !important



- Khi một thuộc tính trong CSS được gắn thêm cú pháp **!important** thì sẽ có mức độ ưu tiên cao nhất.
- **Nếu có nhiều** thuộc tính cùng có **!important** thì lại **quay về bài toán tính độ ưu tiên**.
- **Hạn chế dùng !important** vì khó bảo trì sau này.

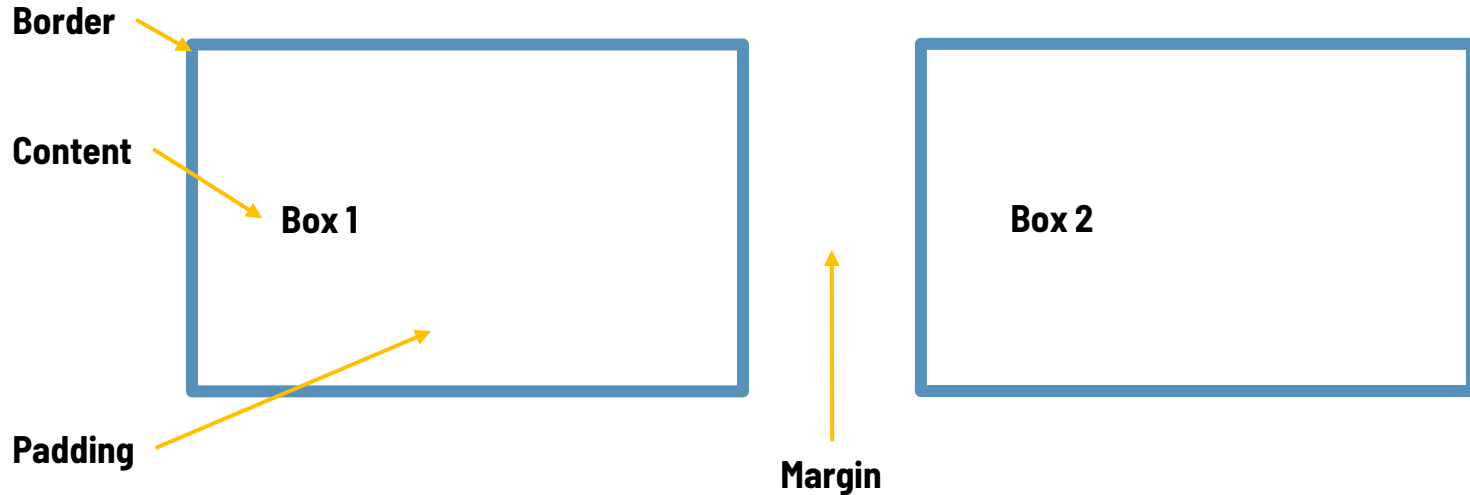


05

Box Model, Content, Padding, Border, Margin

5.1. Box Model (Mô hình hộp)

- Một phần tử trong HTML có thể được coi là một cái **hộp**.
- Hộp này bao gồm có 4 thành phần:
 - Content (Nội dung) - Là phần nội dung bên trong hộp.
 - Border (Đường viền): Đường viền của hộp.
 - Padding (Phần đệm) - Là khoảng cách giữa nội dung và đường viền.
 - Margin (Lề) - Là khoảng cách giữa các hộp.



5.2. Borders (Đường viền)

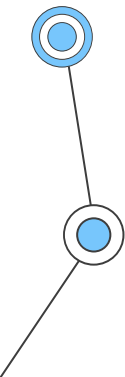
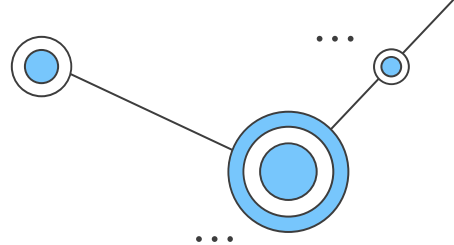
- Là đường viền của hộp.
- Cú pháp:

```
selector {  
  border: [border-width] [border-style] [border-color];  
}
```

- Trong đó:
 - **border-width**: Độ dày của đường viền.
 - **border-style**: Kiểu của đường viền.
 - **dotted**: Đường viền chấm
 - **dashed**: Đường viền nét đứt
 - **solid**: Đường viền liền
 - **border-color**: Màu của đường viền.

```
div {  
  border: 5px solid black;  
}
```

Box 1



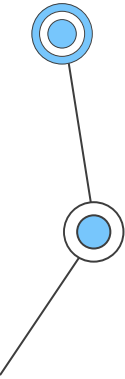
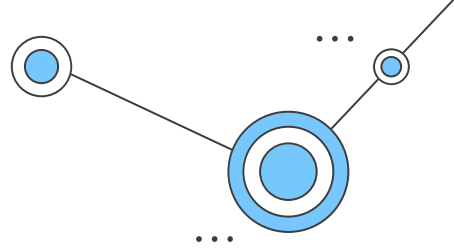
5.2. Borders (Đường viền)

- Cú pháp **bo góc** đường viền:

```
selector {  
  border-radius: giá trị;  
}
```

```
div {  
  border: 5px solid black;  
  border-radius: 8px;  
}
```

Box 1

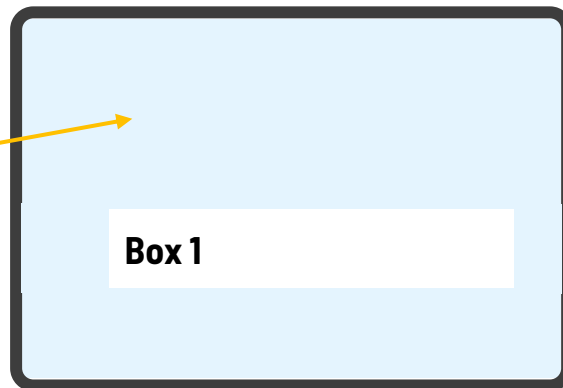


5.3. Padding (Phần đệm)

- Là khoảng cách giữa nội dung và đường viền.
- Cú pháp:

```
selector {  
  padding: top right bottom left;  
}
```

```
div {  
  border: 5px solid black;  
  border-radius: 5px;  
  padding: 100px 20px 50px 50px;  
}
```

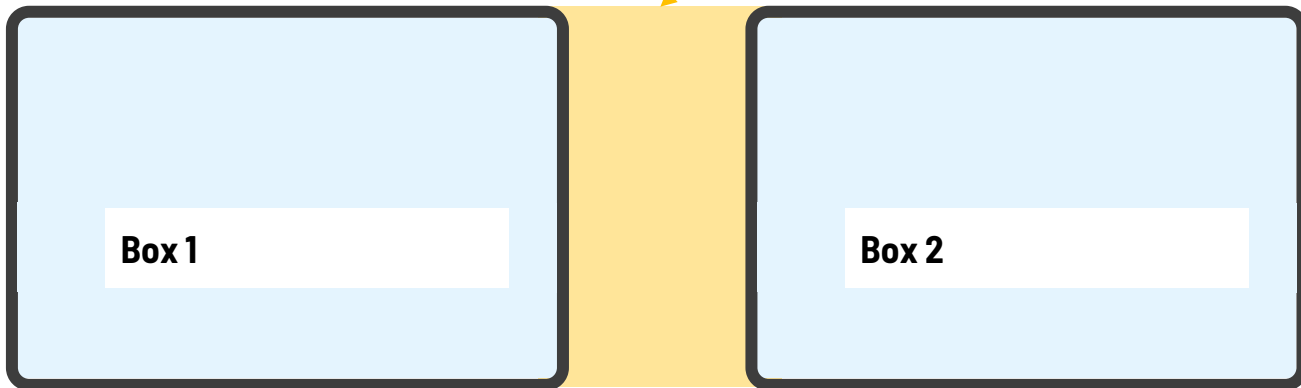


5.4. Margin (Lề)

- Là khoảng cách giữa các hộp.
- Cú pháp

```
selector {  
  margin: top right bottom left;  
}
```

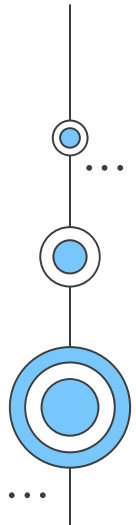

```
div {  
  border: 5px solid black;  
  border-radius: 5px;  
  padding: 15px 20px 10px 5px;  
  margin: 0px 80px 0px 0px;  
}
```





06

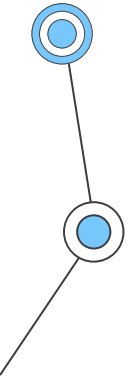
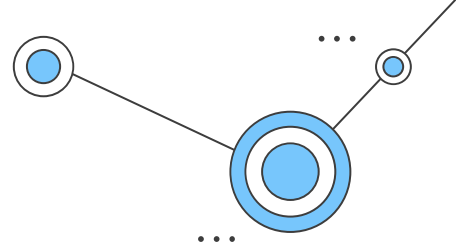
**Thuộc tính
width, height**



06. Thuộc tính width, height

- Dùng để thiết lập chiều rộng (width) và chiều cao (height) cho một phần tử.
- Cú pháp:

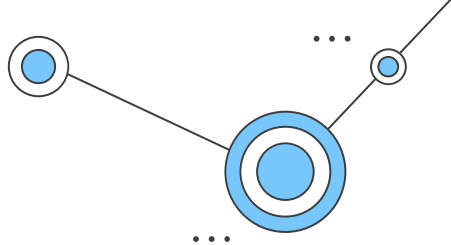
```
selector {  
  width: giá trị;  
  height: giá trị;  
}
```



07

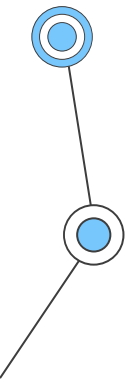
Box Sizing (Kích thước hộp)

07. Box Sizing (Kích thước hộp)



- Thuộc tính **box-sizing** được sử dụng để xác định **cách trình duyệt tính toán kích thước của một phần tử**.
- Có **hai giá trị** chính cho box-sizing:
 - **content-box**: Giá trị mặc định. Trình duyệt tính toán kích thước của phần tử dựa trên nội dung (content) bên trong, không bao gồm padding và border.
 - **border-box**: Trình duyệt tính toán kích thước của phần tử bao gồm cả nội dung, padding và border.
- Cú pháp:

```
selector {  
  box-sizing: giá trị;  
}
```





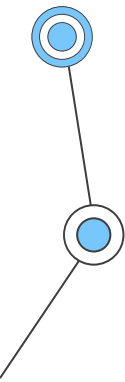
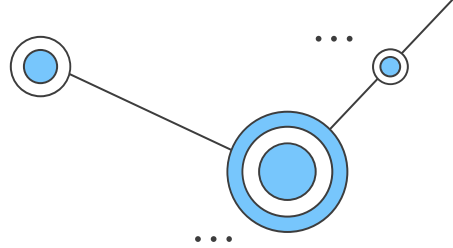
08

Đơn vị chiều dài



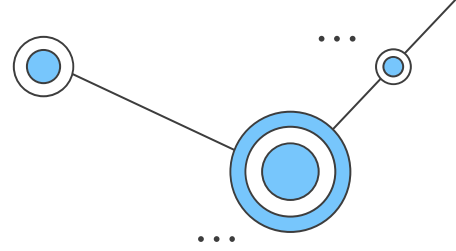
8.1. Đơn vị px

- Một pixel tương đương với một điểm ảnh trên màn hình.
- Là đơn vị đo lường tuyệt đối.



8.2. Đơn vị %

- Giá trị tương đối so với kích thước của phần tử cha.
- Là đơn vị đo lường tương đối.

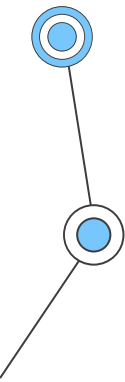
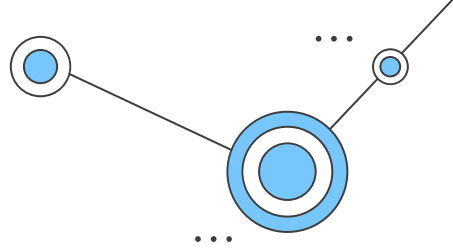


8.3. Đơn vị em

- **1em** tương đương với font-size của phần tử cha có định nghĩa font-size.
- Là đơn vị đo lường tương đối.

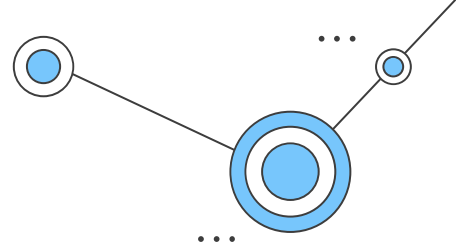
ví dụ thẻ cha có font-size: 12px vậy thì thẻ con dùng đơn vị 1em sẽ tương đương 12px của thẻ cha

còn nếu thẻ cha không sử dụng thì nó sẽ tìm ra ngoài nữa cho đến thẻ `<html></html>`



8.4. Đơn vị rem


- **1rem** tương đương với font-size của phần tử gốc (Chính là thẻ <html>).
- Là đơn vị đo lường tương đối.





09

Overflow (Tràn ra)



09. Overflow (Tràn ra)

- Thuộc tính **overflow** được sử dụng để xác định cách xử lý nội dung khi vượt quá kích thước của phần tử.
- Cú pháp:

```
selector {  
  overflow: giá trị;  
}
```

- Có các giá trị sau cho thuộc tính **overflow**:
 - **visible**: Hiển thị nội dung vượt quá kích thước của phần tử.
 - **hidden**: Ẩn nội dung vượt quá kích thước của phần tử.
 - **scroll**: Hiển thị thanh cuộn để di chuyển qua lại nội dung vượt quá kích thước của phần tử.
 - **auto**: Tự động hiển thị thanh cuộn nếu nội dung vượt quá kích thước của phần tử.

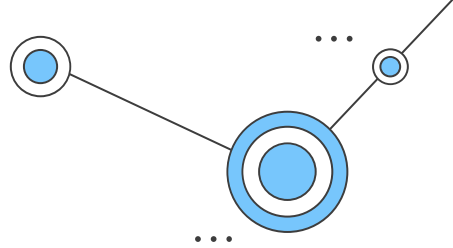


10

Display (Hiển thị)



10.1. Thuộc tính display



- **display: inline;**

- Không đặt được chiều rộng và chiều cao.
- Không thêm được margin và padding cho bên trên và bên dưới.

Phần tử 1

Phần tử 2

- **display: block;**

- Chiều rộng kéo dài từ trái sang phải, luôn bắt đầu ở một hàng mới.
- Cho phép đặt chiều rộng và chiều cao.
- Cho phép đặt margin, padding cho bên trên và bên dưới.

Phần tử 1

Phần tử 2

- **display: inline-block;**

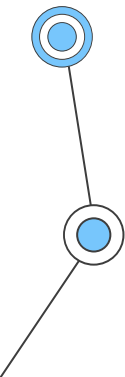
- Chiều rộng mặc định bằng chiều rộng của phần tử.
- Không bắt đầu ở một hàng mới.
- Cho phép đặt chiều rộng và chiều cao.
- Cho phép đặt margin, padding cho bên trên và bên dưới.

Phần tử 1

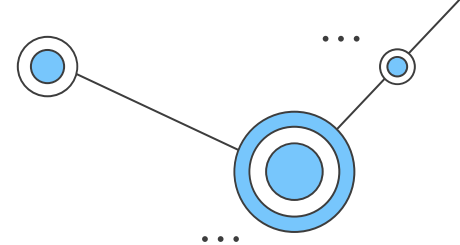
Phần tử 2

- **display: none;**

- Ẩn phần tử hoàn toàn



10.2. Thuộc tính visibility



- **visibility: visible;**

- Mặc định. Phần tử luôn được hiển thị.

Phần tử 1

Phần tử 2

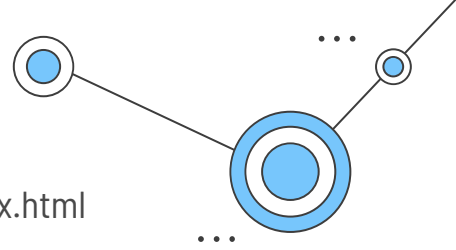
- **visibility: hidden;**

- Ẩn nhưng vẫn chiếm diện tích của phần tử đó.

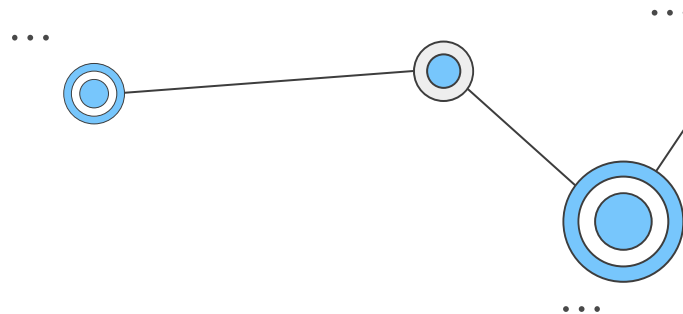


Bài tập

- **Link bài tập:** <https://course-front-end-24.vercel.app/lessons/lesson-4-5/index.html>

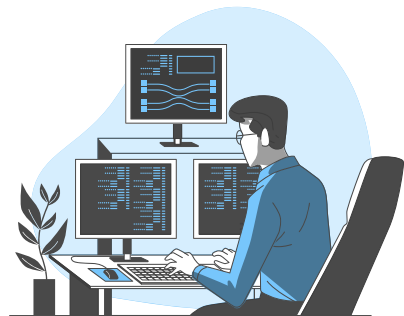


CSS



Khóa học Frontend

Bài 06: Học CSS3 (Tiết 3)



Nội dung

01

Position (Vị trí)

...

02

z-index

...

03

Variables (Biến)

...

04

Flexbox

...



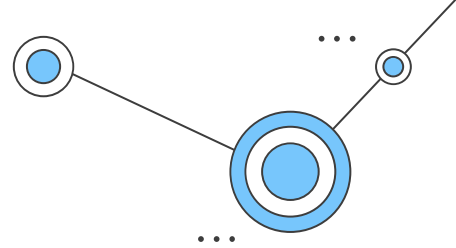


01

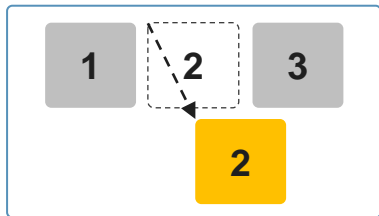
Position (Vị trí)



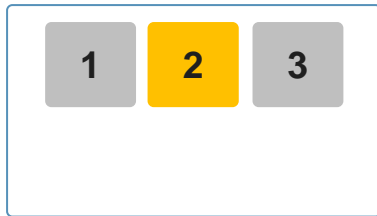
01. Position (Vị trí)



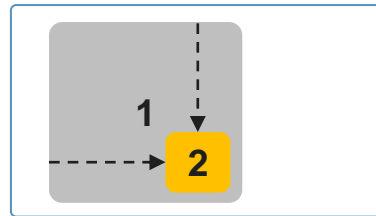
- Thuộc tính **position** được sử dụng để xác định vị trí của một phần tử trên trang web.
- Có **5 giá trị**:



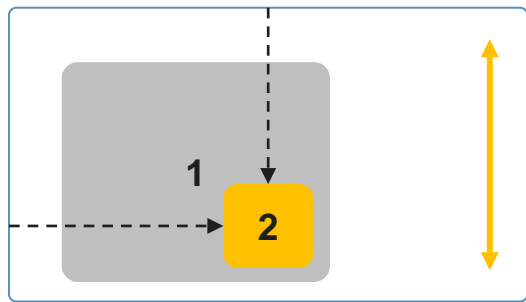
relative (tương đối)



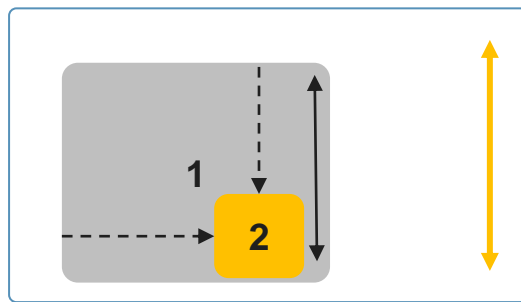
static (tĩnh)



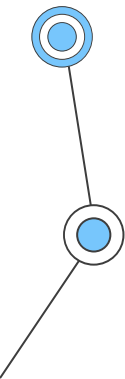
absolute (tuyệt đối)



fixed (cố định)

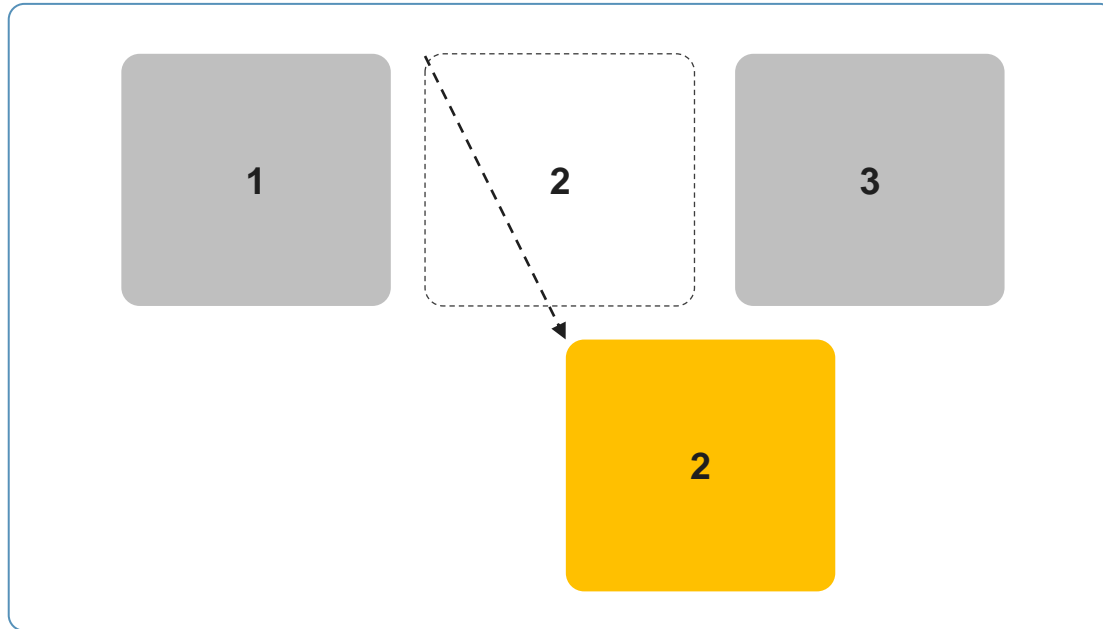


sticky (tương đối + tuyệt đối)

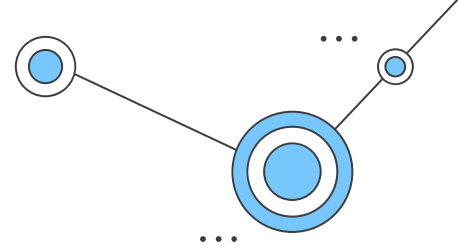


01. Position (Vị trí)

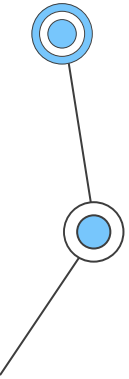
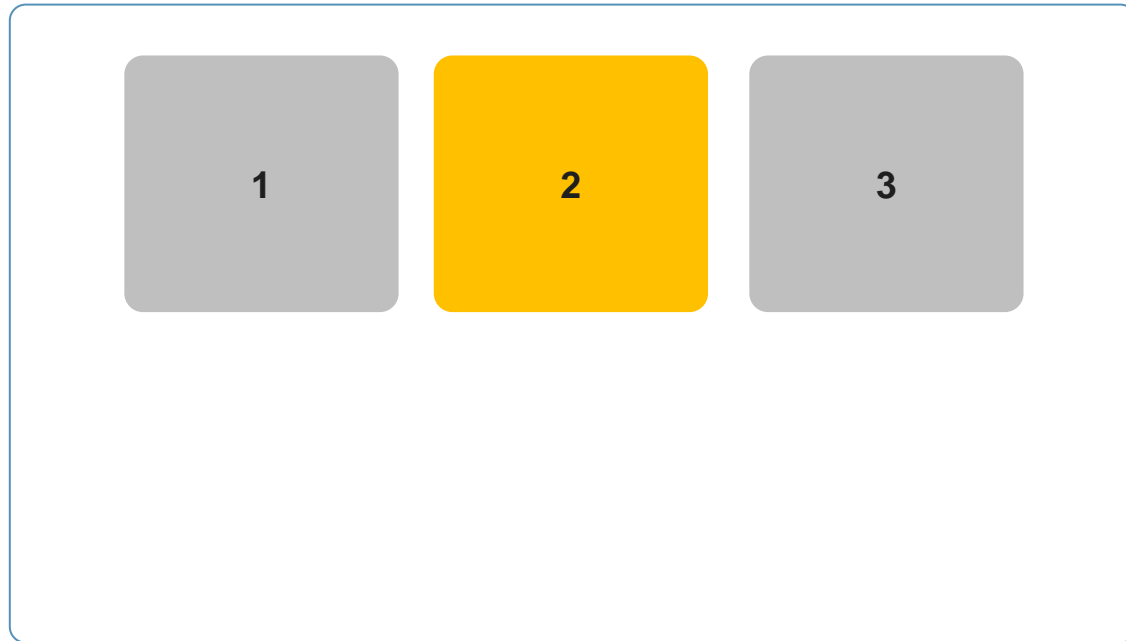
- **relative** (tương đối)
 - Vị trí tương đối so với vị trí mặc định của chính nó.
 - Sử dụng được các thuộc tính top, right, bottom, left.



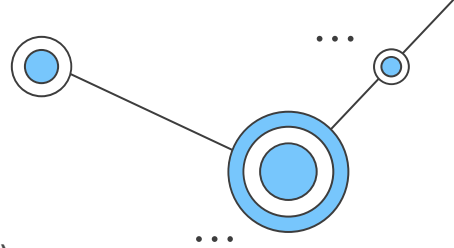
01. Position (Vị trí)



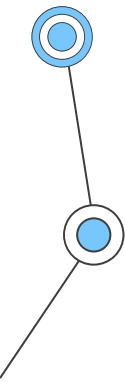
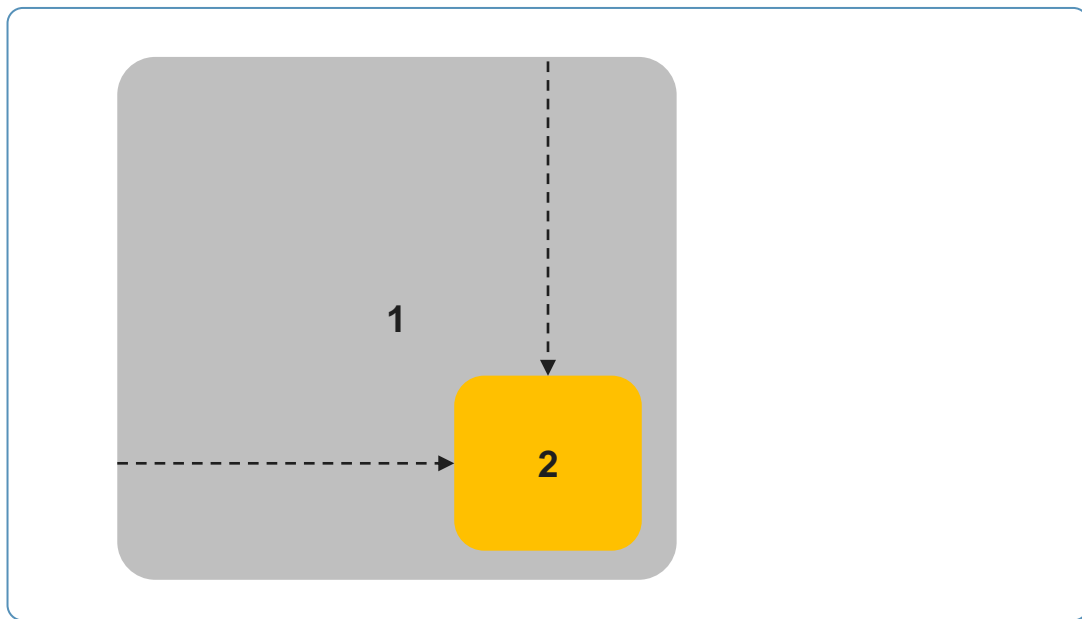
- **static** (tĩnh)
 - Vị trí luôn cố định so với vị trí mặc định của chính nó.
 - Không sử dụng được các thuộc tính top, bottom, right, left.



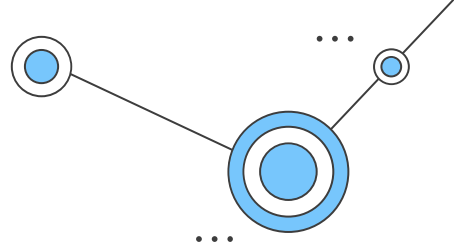
01. Position (Vị trí)



- **absolute** (tuyệt đối)
 - Vị trí tuyệt đối so với phần tử bao ngoài (phần tử cha - có position khác static).
 - Sử dụng được các thuộc tính top, right, bottom, left.

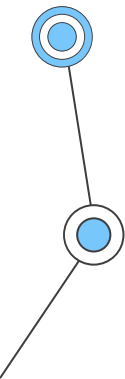
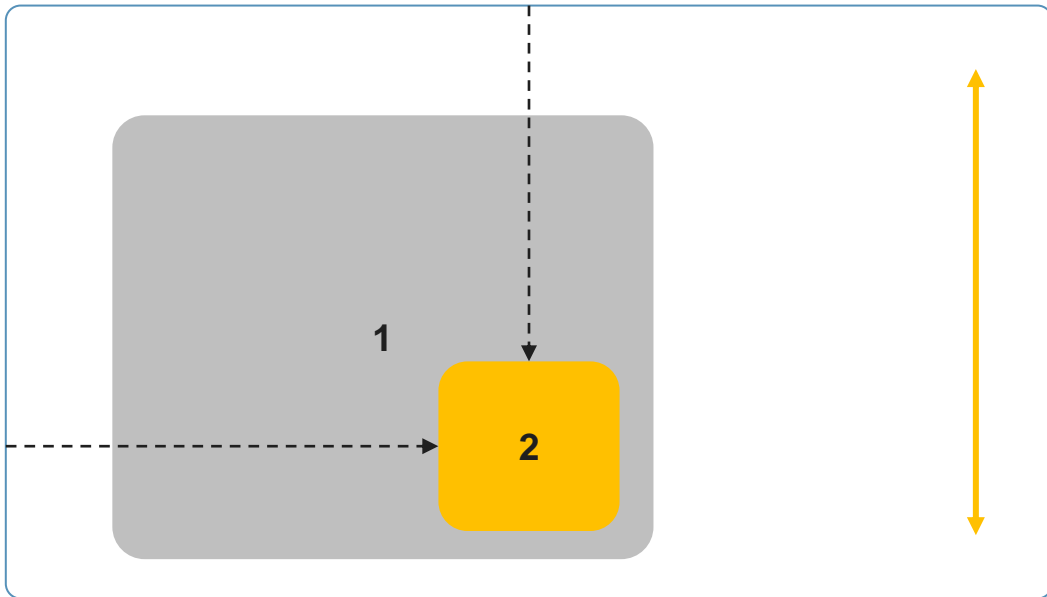


01. Position (Vị trí)

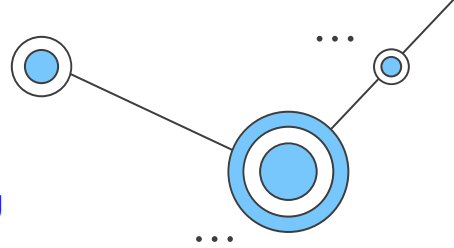


- **fixed** (cố định)
 - Phần tử được đặt vị trí tương đối với cửa sổ trình duyệt.
 - Dù có cuộn trang, phần tử sẽ vẫn giữ vị trí ban đầu của nó.
 - Sử dụng được các thuộc tính top, right, bottom, left.

Đối với Fixed thì sẽ không bị chiếm diện tích của trang



01. Position (Vị trí)



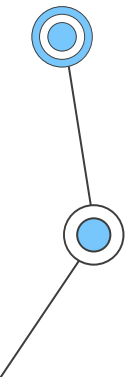
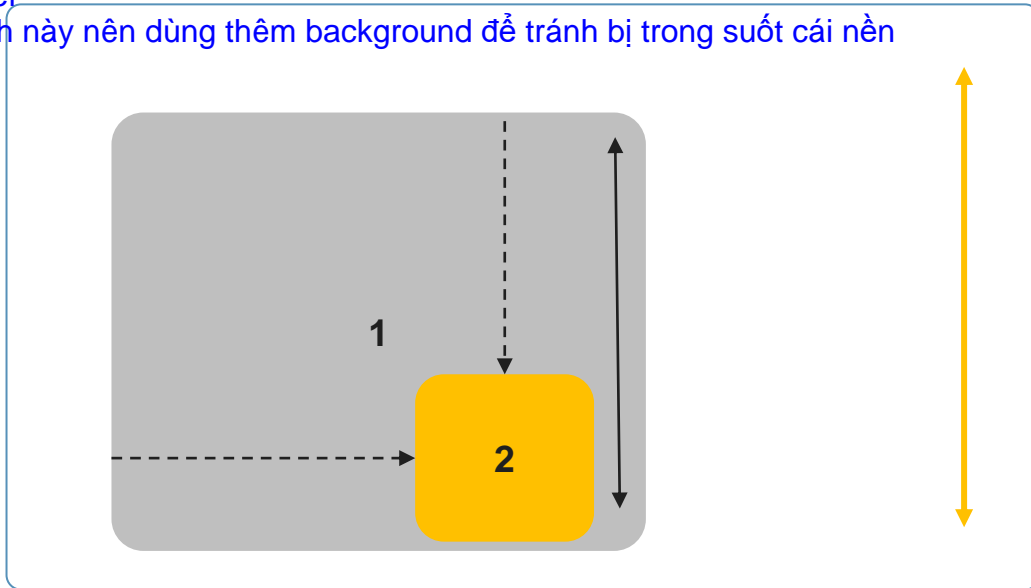
- **sticky** (tương đối + tuyệt đối)

Có chiếm diện tích của trang

- Là sự kết hợp của 2 kiểu **relative** và **fixed**.
- Khi **chưa scroll đến** phần tử đó thì phần tử sẽ **hiển thị kiểu relative**.
- Khi **scroll đến** phần tử đó thì phần tử sẽ **hiển thị kiểu fixed**.

Thường sẽ dùng cho header

Lưu ý ở box dùng thuộc tính này nên dùng thêm background để tránh bị trong suốt cái nền khi bị scroll đến





02

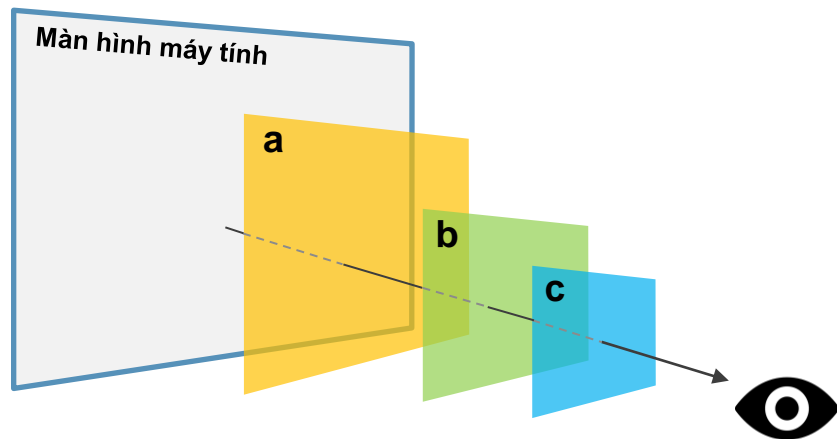
z-index



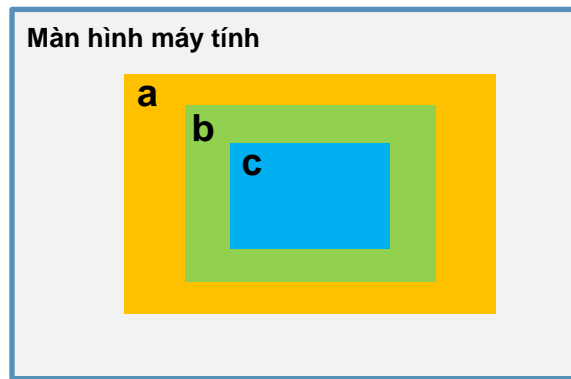
02. z-index

- Thuộc tính **z-index** được sử dụng để xác định thứ tự xếp chồng của các phần tử trùng vị trí.
- Giá trị mặc định là 0.
- z-index càng cao thì phần tử đó càng nằm trên.
- **Chú ý:** **z-index chỉ có tác dụng** khi thuộc tính **position khác static**.

không để position thì z-index còn nhỏ hơn z-index = 0 nữa nhé



Hình 3D



Hình 2D

03

Variables (Biến)

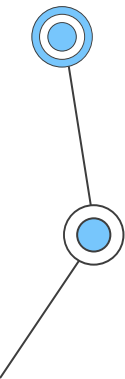
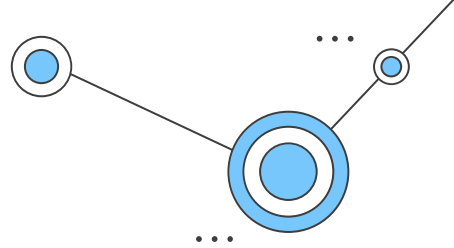
03. Variables (Biến)

- Đặt tên biến để **tái sử dụng** được một **giá trị** ở nhiều nơi.
- Cú pháp khai báo:

```
:root {  
  --ten-bien-1: giá trị 1;  
  --ten-bien-2: giá trị 2;  
  ...  
}
```

- Cú pháp sử dụng:

```
selector {  
  thuộc-tính-a: var(--ten-bien-1);  
  thuộc-tính-b: var(--ten-bien-2);  
}
```





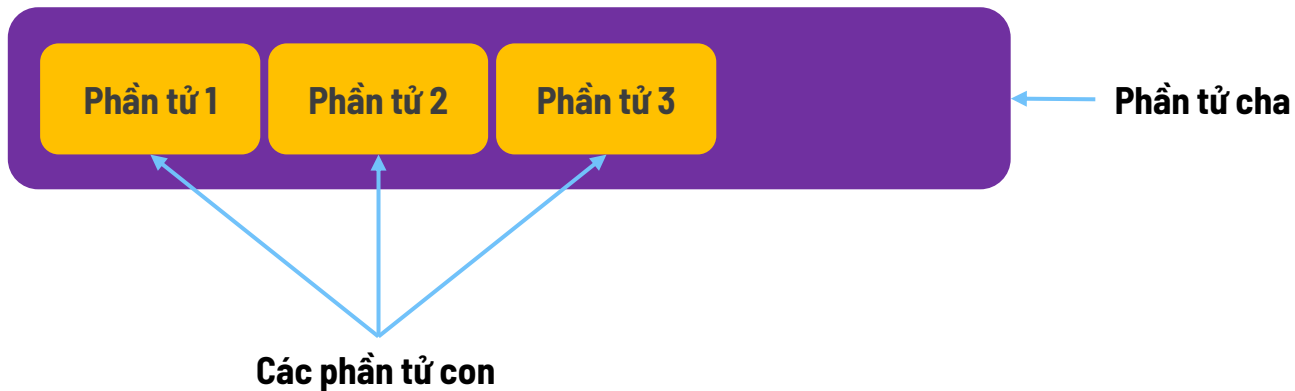
04

Flexbox



4.1. Flexbox là gì?

- **Flexbox** trong CSS là một công cụ mạnh mẽ dùng để **xây dựng và điều chỉnh bố cục linh hoạt** của các phần tử trên trang web.
- Flexbox có 2 thành phần là:
 - **Phần tử cha** (dùng để bọc bên ngoài)
 - **Các phần tử con**





4.2. Display Flex

- **Display Flex** giúp sắp xếp các phần tử một cách dễ dàng, linh hoạt hơn trước.
- Cú pháp:

```
.phan-tu-cha {  
  display: giá trị;  
}
```

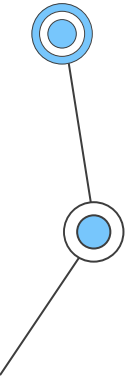
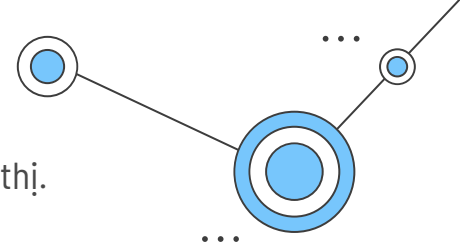
- Các giá trị:

Giá trị	Mô tả	Minh họa
flex	Giúp các phần tử con hiển thị linh hoạt. Phần tử cha có chiều rộng là 100%.	
inline-flex	Giúp các phần tử con hiển thị linh hoạt. Phần tử cha có chiều rộng bằng chiều rộng của các phần tử con bên trong cộng lại.	

4.3. Flex Direction

- Thuộc tính **flex-direction** dùng để **xác định hướng của các phần tử con** được hiển thị.
- Cú pháp:

```
.phan-tu-cha {  
  display: giá trị;  
  flex-direction: giá trị;  
}
```



4.3. Flex Direction

còn gọi là trục chính

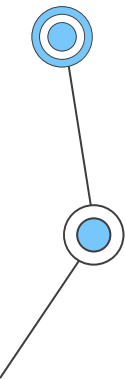
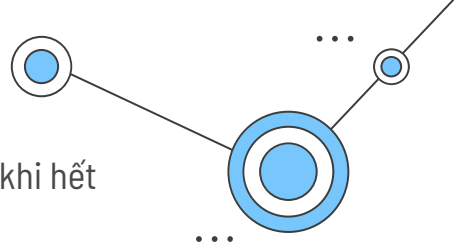
- Các giá trị:

Giá trị	Mô tả	Minh họa
trục chính chạy từ trái sang phải row	Giá trị mặc định. Các phần tử con được sắp xếp theo hướng từ trái sang phải .	
row-reverse	Các phần tử con được sắp xếp theo hướng từ phải sang trái .	
trục chính chạy từ trên xuống dưới column	Các phần tử con được sắp xếp theo hướng từ trên xuống dưới .	
column-reverse	Các phần tử con được sắp xếp theo hướng từ dưới lên trên .	

4.4. Flex Wrap

- Thuộc tính **flex-wrap** dùng để xác định liệu các phần tử con có tự động xuống hàng khi hết chiều ngang hay không.
- Cú pháp:

```
.phan-tu-cha {  
  display: giá trị;  
  flex-wrap: giá trị;  
}
```



4.4. Flex Wrap

- Các giá trị:

Giá trị	Mô tả	Minh họa
nowrap	Giá trị mặc định. Các phần tử con luôn nằm trên một hàng .	
wrap	Các phần tử con tự động xuống hàng khi hết chiều ngang.	
wrap-reverse	Tương tự như wrap . Các phần tử con tự động thêm hàng mới lên trên .	

nếu flex-direction: column thì nó nhảy sang phải

4.5. Justify Content

- Thuộc tính **justify-content** dùng để **điều chỉnh vị trí của các phần tử con theo hướng trục chính** (hướng của flex-direction).
- Cú pháp:

```
.phan-tu-cha {  
  display: giá trị;  
  flex-direction: giá trị;  
  justify-content: giá trị;  
}
```

justify-content: space-evenly với
flex-direction: column



4.5. Justify Content

mặc định với
flex-direction: row-reverse;

• Các giá trị:

mặc định với flex-direction: row;

Căn các phần tử theo
hướng trục chính

Giá trị	Mô tả	
flex-start	Giá trị mặc định. Các phần tử con được căn về phía đầu trục chính .	
flex-end	Các phần tử con được căn về phía cuối trục chính .	
center	Các phần tử con được căn giữa trục chính .	
space-between	Các phần tử con có khoảng cách đều nhau . Phần tử con đầu tiên và cuối cùng sẽ sát với lề của phần tử cha .	
space-around	Các phần tử con có khoảng cách đều nhau . Khoảng cách giữa phần tử với lề = 1/2 khoảng cách giữa phần tử với phần tử .	
space-evenly	Các phần tử con có khoảng cách đều nhau . Khoảng cách giữa phần tử với lề = khoảng cách giữa phần tử với phần tử .	

vì hướng
của trục
chính nằm
ngang nên
nó mới
ở giữa

Các phần
tử
con có
khoảng
cách bằng
nhau

4.6. Align Items

- Thuộc tính **align-items** dùng để **điều chỉnh vị trí của các phần tử con theo hướng trục phụ** (vuông góc với hướng của flex-direction).
- Dùng để điều chỉnh cho **một dòng**.
- Cú pháp:

```
.phan-tu-cha {  
  display: giá trị;  
  flex-direction: giá trị;  
  align-items: giá trị;  
}
```

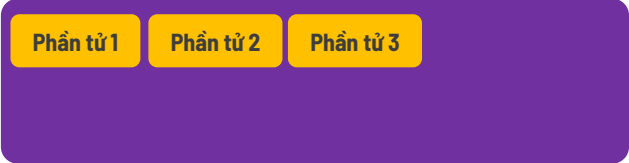

vuông góc với trục chính

nếu có nhiều dòng thì chiều cao mỗi dòng bằng nhau nhé !

4.6. Align Items

thêm chiều cao phần tử cha
thì phần tử con sẽ cao theo.
xem cái lưu ý thêm

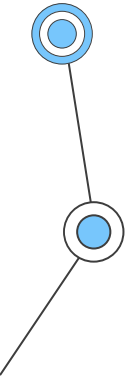
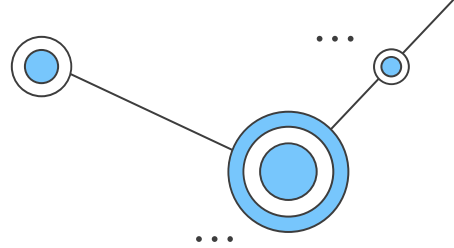
- Các giá trị:

Giá trị	Mô tả	Minh họa
stretch	Giá trị mặc định. Các phần tử con tự động tăng chiều cao để lấp đầy chiều cao của phần tử cha. <i>Lưu ý: Nếu các phần tử con có thuộc tính height thì sẽ ưu tiên hơn.</i>	
flex-start	Các phần tử con được căn về phía đầu trục phụ .	
flex-end	Các phần tử con được căn về phía cuối trục phụ .	
center	Các phần tử con được căn giữa trục phụ .	

4.7. Align Content

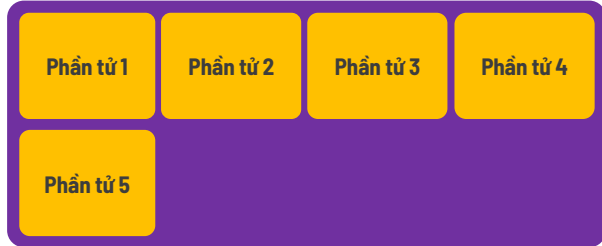
- Tương tự **align-items** nhưng áp dụng cho **nhiều dòng**.
- Cú pháp:

```
.phan-tu-cha {  
  display: giá trị;  
  flex-direction: giá trị;  
  align-content: giá trị;  
}
```





4.7. Align Content

- Các giá trị:

Giá trị	Mô tả	Minh họa
stretch	<p>Giá trị mặc định.</p> <p>Các phần tử con tự động tăng chiều cao để lấp đầy chiều cao của phần tử cha.</p> <p><i>Lưu ý: Nếu các phần tử con có thuộc tính height thì sẽ ưu tiên hơn.</i></p>	
flex-start	<p>Các phần tử con được căn về phía đầu trực phụ.</p>	

4.7. Align Content

- Các giá trị:

Giá trị	Mô tả	Minh họa
flex-end	Các phần tử con được căn về phía cuối trục phụ .	
center	Các phần tử con được căn giữa trục phụ .	

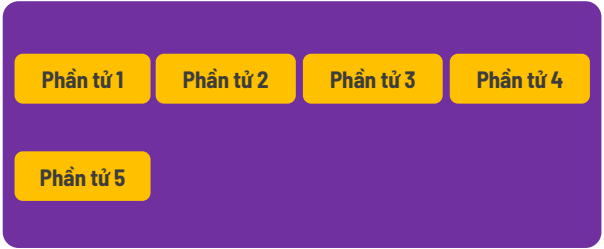
4.7. Align Content

- Các giá trị:

Giá trị	Mô tả	Minh họa
space-between	Các phần tử con có khoảng cách đều nhau . Phần tử con đầu tiên và cuối cùng sẽ sát với lề của phần tử cha.	
space-around	Các phần tử con có khoảng cách đều nhau . Khoảng cách giữa phần tử với lề = 1/2 khoảng cách giữa phần tử với phần tử.	

4.7. Align Content

- Các giá trị:

Giá trị	Mô tả	Minh họa
space-evenly	Các phần tử con có khoảng cách đều nhau . Khoảng cách giữa phần tử với lề = khoảng cách giữa phần tử với phần tử .	

4.8. Align Self

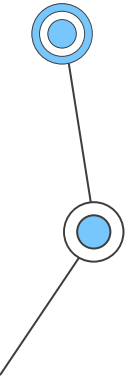
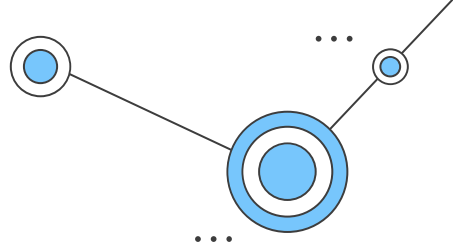
- Tương tự **align-items** nhưng áp dụng cho **một phần tử con** cụ thể.
- Thuộc tính align-self có thể được sử dụng để ghi đè giá trị của **align-items**.
- Cú pháp:

```
.phan-tu-con-cu-the {  
  align-self: giá trị;  
}
```

ví dụ

```
.child:nth-child(2) {  
  align-self: center;  
}
```

trực phụ



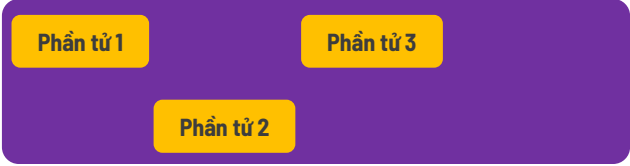
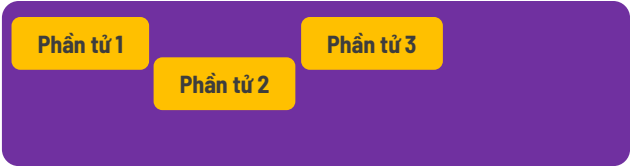
4.8. Align Self

- Các giá trị:

Giá trị	Mô tả	Minh họa
auto	Giá trị mặc định. Phần tử con cụ thể được kế thừa giá trị của align-items .	
stretch	Phần tử con cụ thể tự động tăng chiều cao để lấp đầy chiều cao của phần tử cha. <i>Lưu ý: Nếu có thuộc tính height thì sẽ ưu tiên hơn.</i>	
flex-start	Phần tử con cụ thể được căn về phía đầu trục phụ .	

4.8. Align Self

- Các giá trị:

Giá trị	Mô tả	Minh họa
flex-end	Phần tử con cụ thể được căn về phía cuối trục phụ .	
center	Phần tử con cụ thể được căn giữa trục phụ .	

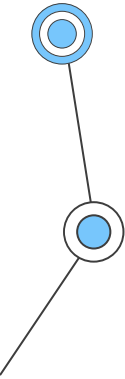
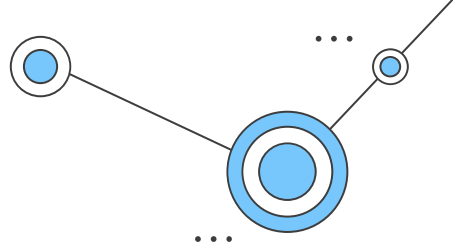
4.9. Order

- Thuộc tính **order** dùng để xác định **thứ tự xuất hiện** của một **phần tử con**.
- Cú pháp:

```
.phan-tu-con-cu-the {  
  order: giá trị;  
}
```

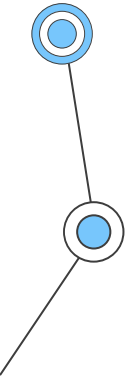
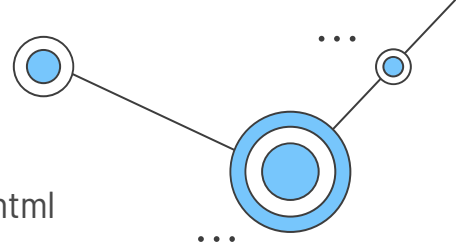
- Giá trị mặc định = 0.  phần tử con ngầm định có
- Giá trị của **order** là một **số nguyên ≥ 0** .

order càng nhỏ càng xuất hiện trước

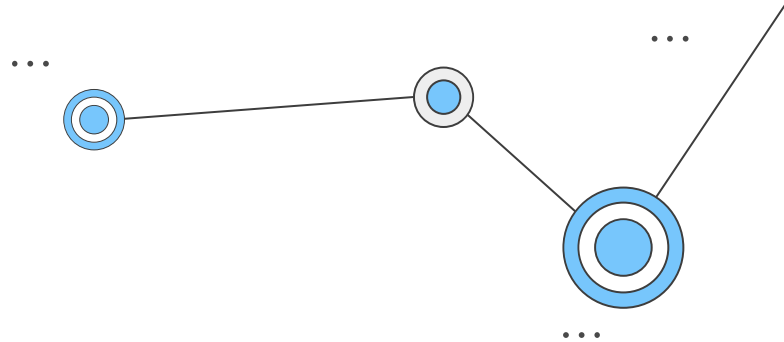


Bài tập

- **Link bài tập:** <https://course-front-end-24.vercel.app/lessons/lesson-6/index.html>

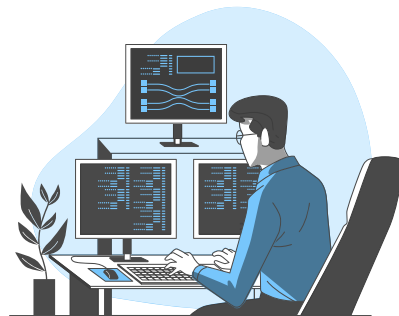


CSS



Khóa học Frontend

Bài 07: Học CSS3 (Tiết 4)



Nội dung

01

Dựng layout mẫu theo
bản thiết kế

...

02

Media Queries

...

03

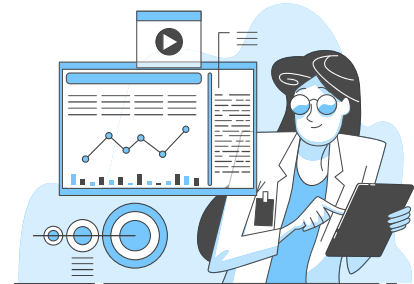
Responsive giao diện mẫu

...

04

Giới thiệu Project 1

...

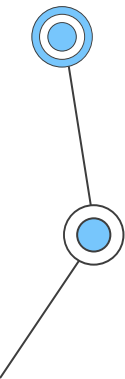
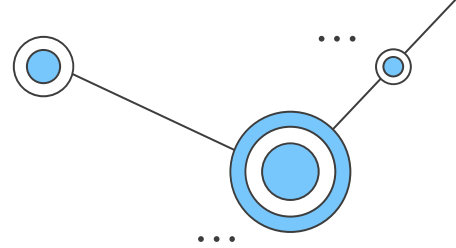


01

Dựng layout mẫu
theo bản thiết kế

01. Dựng layout mẫu theo bản thiết kế

- **Link Figma:** *Gửi trong buổi học.*
- Các bước xây dựng layout: *Hướng dẫn trong buổi học.*





02

Media Queries



02. Media Queries

- **Responsive giao diện** là lập trình ra giao diện website **tương thích** với **nhiều thiết bị** có kích thước màn hình khác nhau.
- Để responsive được giao diện cần **sử dụng Media Queries** trong CSS.
- Cú pháp:

```
@media (max-width: giá trị) {  
    /* Code CSS */  
}
```

ví dụ 1200px thì từ 1200px trở xuống thì sẽ áp dụng đoạn css trong này

mỗi khối code thì responsive ngay bên dưới khối đó

gọi là màn hình có kích thước tối đa là 1200px chẳng hạn :))

```
@media (max-width: 1230px) {  
    .container {  
        width: 720px;  
    }  
}  
/* End .container */
```

khoảng dư cho nó đẹp

viết đúng thứ tự như này thì mới đúng, vì các thẻ h1 cùng độ ưu tiên nên nó sẽ lấy chỗ css cuối cùng

```
h1 {  
    color: green;  
}  
  
@media (max-width: 1230px) {  
    h1 {  
        color: blue;  
    }  
}  
  
@media (max-width: 750px) {  
    h1 {  
        color: red;  
    }  
}
```

```
@media (max-width: 1230px) {  
    .header {  
        padding: 15px 0;  
        position: sticky;  
        top: 0;  
        left: 0;  
        background-color: #fff;  
    }  
}
```

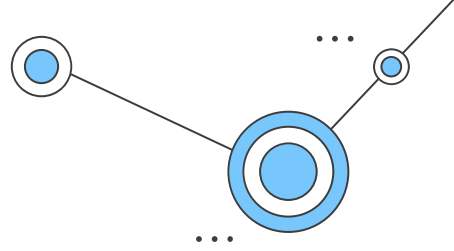
đoạn này lặp lại thì bỏ đi không cần code lại

03

Responsive giao diện mẫu

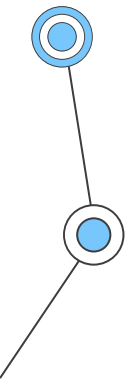
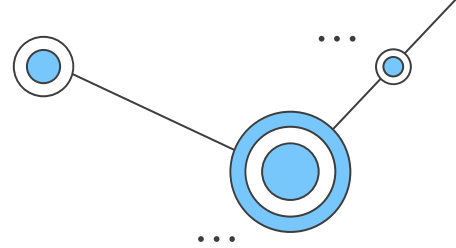
3.1. Responsive cho màn hình máy tính bảng

- *Hướng dẫn trong buổi học.*



3.2. Responsive cho màn hình điện thoại

- *Hướng dẫn trong buổi học.*



04

Giới thiệu Project 1

04. Giới thiệu Project 1

- Link Figma của Project 1: *Gửi trong buổi học.*

