**CSS (14) - Mô hình cái hộp (1)**

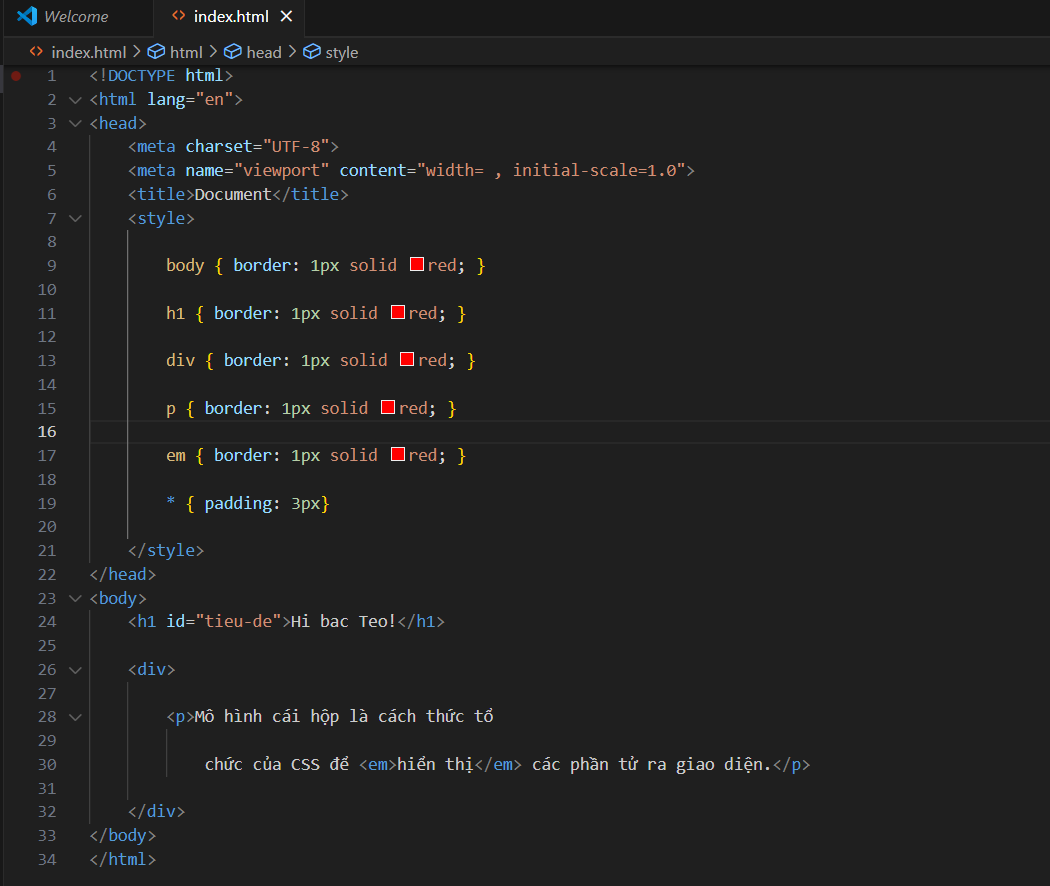
**1 Định dạng bố cục**

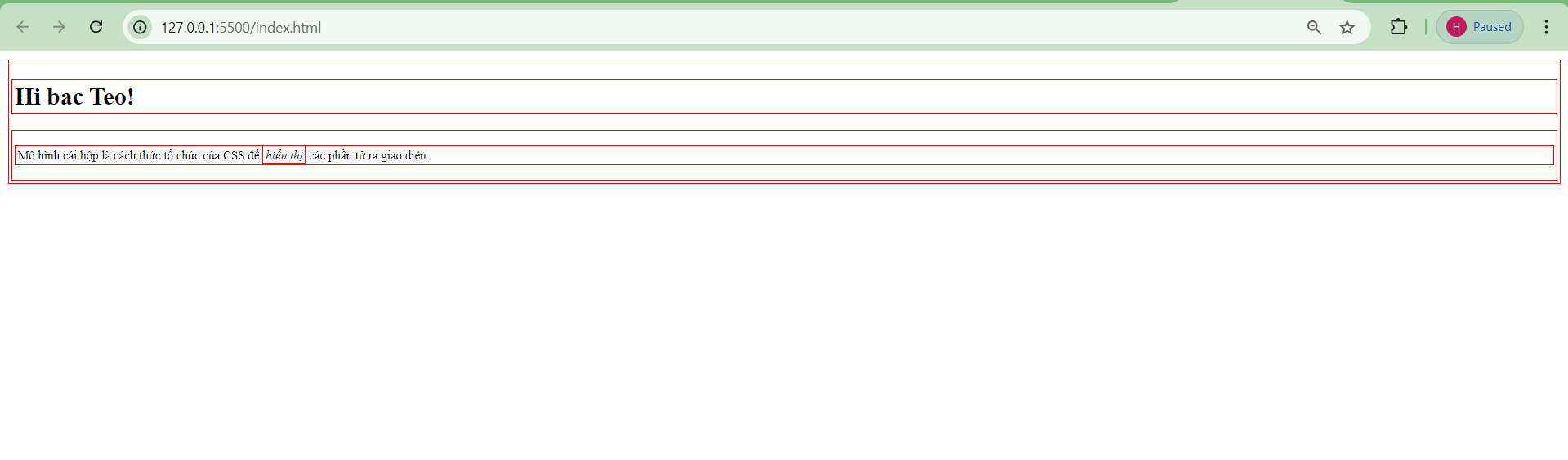
**1.1 Mô hình cái hộp**

**1.1.1 Định nghĩa**

Như trong phần Tổng quan về CSS đã đề cập. Trong việc định dạng hiển thị, chúng ta sẽ phải đặt các phần tử HTML ở một ví trí cụ thể trên giao diện, với chiều rộng, chiều dài xác định; hoặc đổ màu nền. Để tiện lợi cho việc này, CSS đưa ra khái niệm mô hình cái hộp (box model).

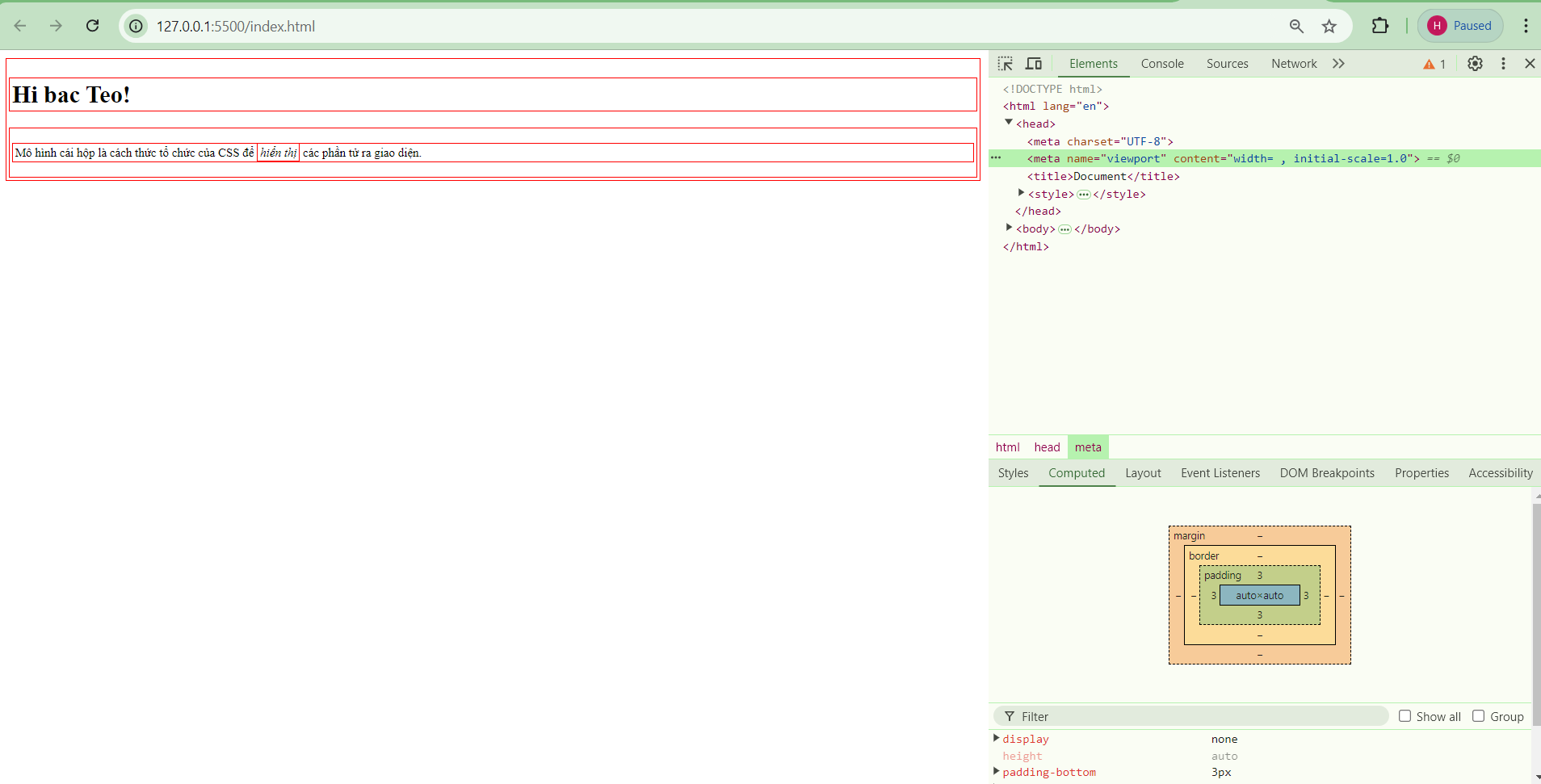
Ví dụ, đoạn mã sau minh họa hình ảnh của box model, gồm 5 cái hộp, với cấu trúc lồng nhau.

Kết quả:



Qua quan sát, bạn sẽ thấy các hộp của phần tử kiểu block có chiều rộng chiếm hết màn hình, trong khi các hộp của phần tử kiểu inline có kích thước vừa đủ để chứa nội dung bên trong.

Chúng ta cũng có thể sử dụng Developer tools để quan sát box model một cách trực quan. Như hình sau:



**1.1.2 Các thành phần của một hộp**

Một phần tử HTML, dù là kiểu block hay inline đều có cái hộp của riêng nó, hình chữ nhật, gồm nhiều thành phần. Xem hình minh họa,

(Lưu ý: để dễ hiểu, chúng ta sẽ dịch các thành phần của cái hộp sang tiếng Việt, Tuy nhiên, khi làm việc hoặc khi đã hiểu rồi thì nên sử dụng từ tiếng Anh cho ngắn gọn, dễ trao đổi với mọi người.)

– Vùng nội dung (content area): là vùng trong cùng của hộp, chứa nội dung của phần tử. Vùng màu trắng chứa văn bản ở hình trên

– Cạnh trong (inner edge): là cạnh của Vùng nội dung, nó là đường phân cách giữa Vùng nội dung và Vùng đệm trong (padding area). Đường này không được hiển thị trên trang web

– Vùng đệm trong (padding area, dãn biên): là khoảng không nằm giữa Vùng nội dung và Đường viền. Vùng đệm trong là tùy chọn (có thể có hoặc không). Ở hình trên, Vùng đệm trong có màu cam vàng

– Đường viền (border): là đường bao quanh phần “nhìn thấy trên giao diện” của phần tử và Vùng đệm trong (nếu có thiết lập). Đường viền là tùy chọn. Đường màu đen ở hình trên

– Vùng đệm ngoài (margin area, căn lề): là khoảng không thêm vào phía ngoài của Đường viền. Ở hình trên, Vùng đệm ngoài có màu xanh lợt. Tuy nhiên, thực tế, Vùng đệm ngoài luôn trong suốt, do vậy phần nhìn thấy sẽ là màu nền của phần tử cha

– Cạnh ngoài (outer edge): là cạnh của Vùng đệm ngoài, đây chính là cạnh giới hạn toàn bộ không gian của một phần tử trên trang web. Ở hình trên, Cạnh ngoài là đường nét đứt, thực tế, đường này không được hiển thị trên trang web. Như vậy, kích thước một phần tử sẽ bao gồm kích thước của Vùng nội dung, Vùng đệm trong, Đường viền, và Vùng đệm ngoài.

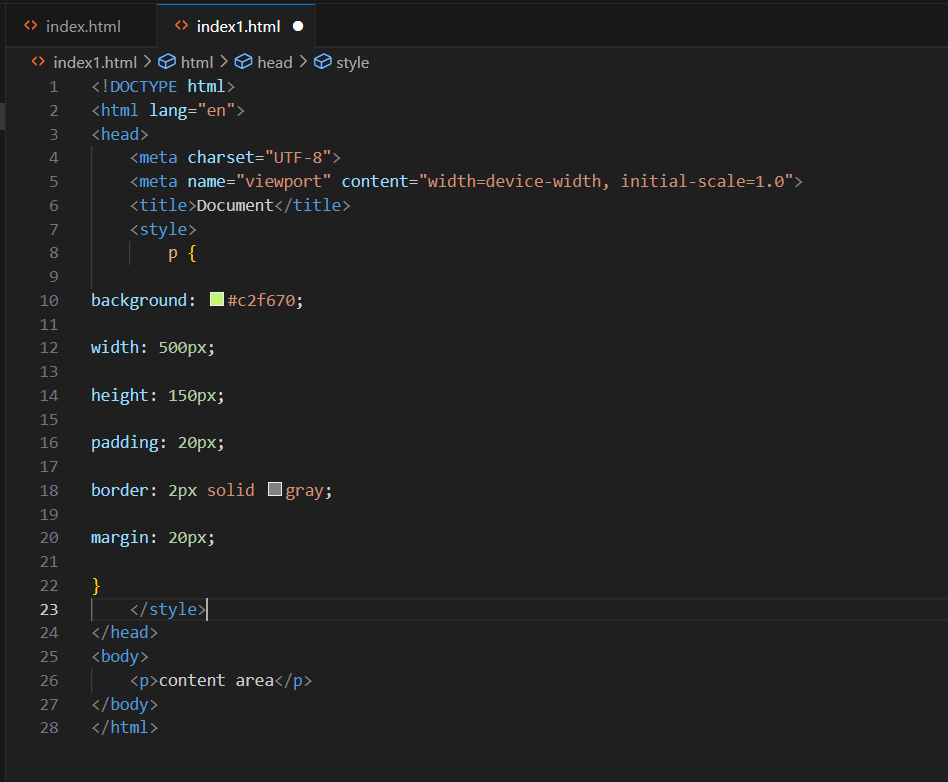
**1.1.3 Xác định kích thước hộp**

Mặc định, kích thước (chiều rộng và chiều cao) của phần tử kiểu block được trình duyệt tự động xác lập (auto). Nó sẽ bằng chiều rộng của cửa sổ trình duyệt, hoặc của phần tử chứa nó; chiều cao sẽ vừa đủ để chứa hết nội dung. Tuy nhiên, có thể sử dụng hai thuộc tính width và height để thiết lập chiều rộng và chiều cao tùy ý.

**Thiết lập kích thước kiểu content-box**

Mặc định, khi không có khai báo thuộc tính box-sizing, thì giá trị width và height sẽ được tính cho Vùng nội dung, hay nói cách khác là kiểu tính kích thước content-box. Tuy nhiên, lập trình viên cũng có thể thiết lập một cách tường minh bằng đoạn mã box-sizing: content-box.

Ví dụ, đoạn mã sau sẽ thiết lập kích thước cho phần tử p gồm các giá trị: rộng 500px, cao 150px, vùng đệm trong (cả bốn phía) 20px, đường viền (cả bốn cạnh) 2px, vùng đệm ngoài (cả bốn phía) 20px. Giả sử kiểu kích thước là content-box, nên chiều rộng và chiều cao sẽ là của vùng nội dung.



Kết quả:

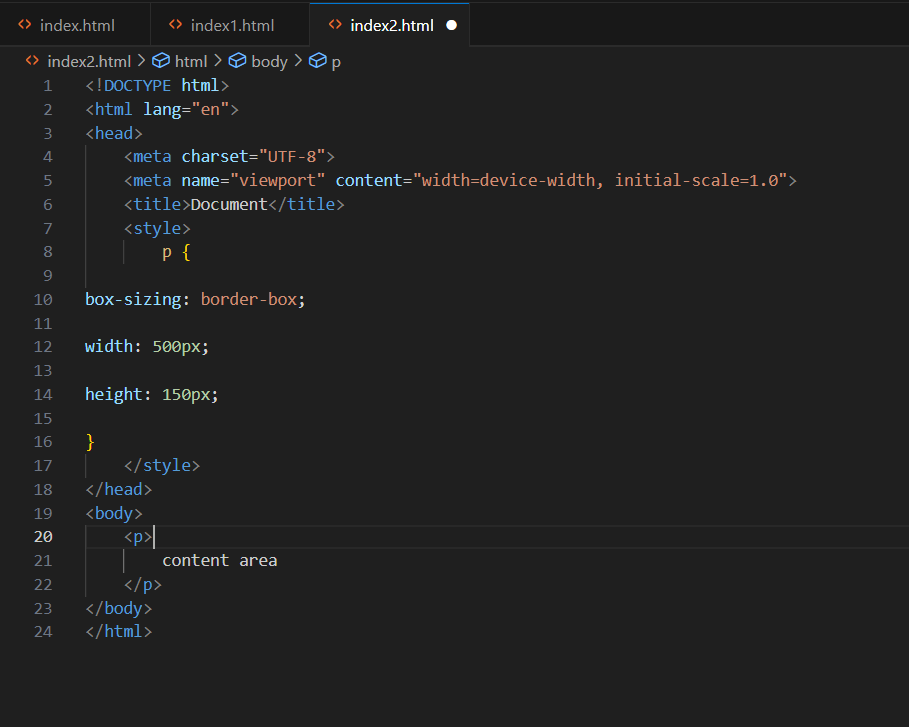


**Thiết lập kích thước kiểu border-box**

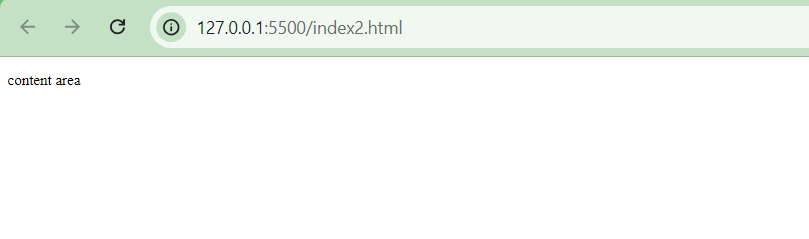
Đây không phải là chế độ tính kích thước mặc định, nên cần khai báo tường minh, box-sizing: border-box.

Giá trị width và height sẽ bao gồm kích thước của vùng nội dung, vùng đệm trong và đường viền.

Ví dụ, quan sát kết quả khi thực hiện thiết lập width cho phần tử p là 500px, và height là 150px,



Kết quả:



Xác định chiều cao

Thông thường, chiều cao của phần tử ít được thiết lập, mà nó hay để ở chế độ mặc định. Nghĩa là, chiều cao của phần tử sẽ tùy thuộc vào lượng nội dung, kích thước của phông chữ, các thiết lập của người dùng và các yếu tố khác. Với phần tử chứa văn bản, nếu có thiết lập chiều cao, thì cần để ý trường hợp nội dung vượt quá sức chứa của phần tử.

Trường hợp nội dung văn bản vượt quá sức chứa của phần tử (do thiết lập chiều cao), CSS có một số tùy chọn để hiển thị phần nội dung bị tràn.

Xử lý khi bị tràn nội dung

Khi bị tràn nội dung, có thể sử dụng thuộc tính overflow để chọn cách xử lý.

Thuộc tính overflow:

– Giá trị: visible | hidden | scroll | auto

– Mặc định: visible

– Áp dụng: các phần tử kiểu block và kiểu inline không chứa văn bản

– Kế thừa: không

– visible: chế độ mặc định, nội dung bị tràn vẫn được hiển thị, nhưng nằm ngoài vùng giới hạn của phần tử

– hidden: nội dung bị tràn sẽ bị ẩn đi

– scroll: phần tử có thêm thanh cuộn, giúp đọc được các nội dung bị tràn. Tuy nhiên, thanh cuộn này luôn tồn tại, dù nội dung không bị tràn.

– auto: trình duyệt tự xử lý khi có nội dung bị tràn, thường là sẽ có thanh cuộn. Tuy nhiên, khi nội dung không bị tràn thì thanh cuộn này sẽ tự động mất đi.

**1.1.4 Vùng đệm trong**

Như đã biết, vùng đệm trong là khoảng không nằm giữa vùng nội dung (content area) và đường viền (border). Nếu đường viền không được thiết lập thì vùng đệm này cũng có thể xem như một “đường viền” cho vùng nội dung.

Với mỗi phần tử, khi thiết lập màu nền và đường viền thì cũng nên thiết lập luôn vùng đệm trong, điều này giúp cho vùng nội dung được gọn gàng, cân đối, tránh tình trạng văn bản nằm quá sát với đường viền.

Có thể thêm vùng đệm trong cho cả hai loại phần tử block và inline.

Có thể thêm vùng đệm trong cho từng cạnh, hoặc cả bốn cạnh một lần.

Để thêm vùng đệm trong cho từng cạnh, sử dụng các thuộc tính sau,

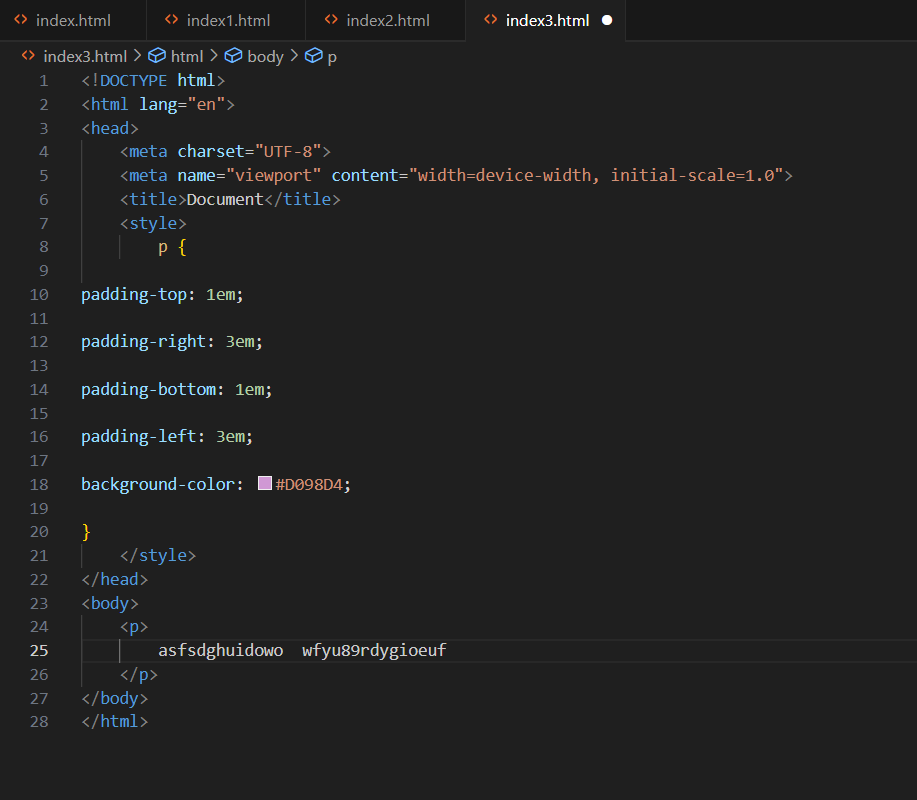
padding-top, padding-right, padding-bottom, padding-left:

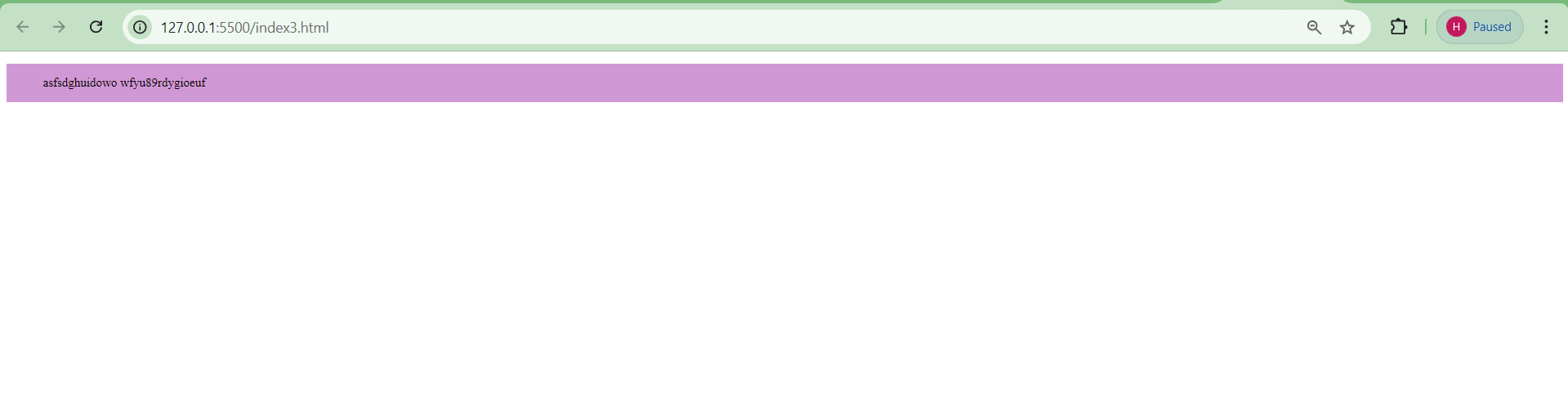
– Giá trị: đơn vị đo chiều dài | %

– Mặc định: 0

– Áp dụng: mọi phần tử

– Kế thừa: không

Kết quả:



**Thuộc tính padding**

Ngoài cách thiết lập padding cho từng cạnh, có thể sử dụng thuộc tính padding để thiết lập một lần cho bốn cạnh.

Thuộc tính padding,

– Giá trị: đơn vị đo chiều dài | %

– Mặc định: 0

– Áp dụng: mọi phần tử

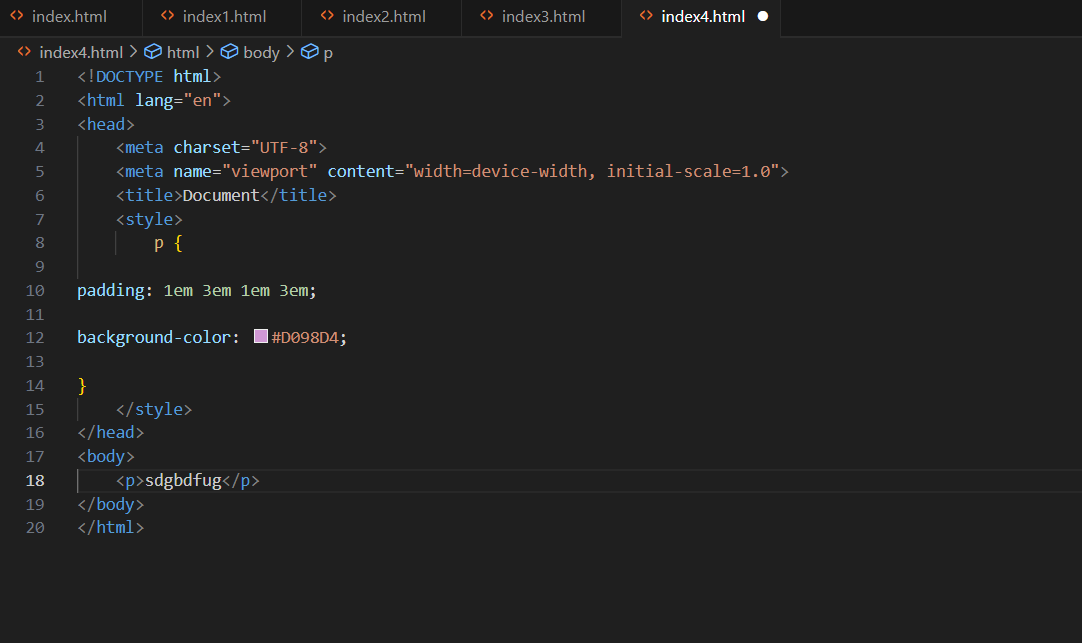
– Kế thừa: không

Thuộc tính padding có thể chấp nhận bốn, ba, hai, hoặc một giá trị.

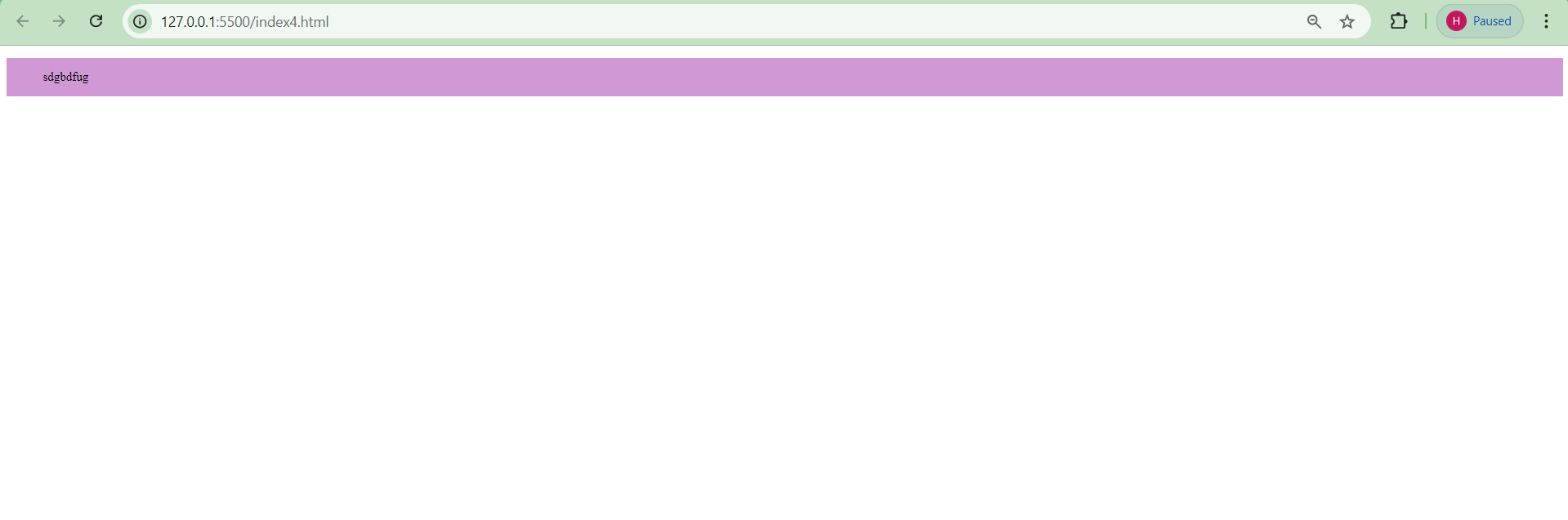
Khi padding có bốn giá trị, các giá trị này sẽ được áp dụng lần lượt cho các cạnh tương ứng, theo chiều kim đồng hồ, bắt đầu là cạnh trên (top), sau đó đến cạnh phải (right), cạnh dưới (bottom), cuối cùng là cạnh trái (left). Để dễ nhớ, tiếng Anh có từ TRouBLe, mỗi chữ viết hoa sẽ ứng với một cạnh.

Cú pháp,

padding: top right bottom left;



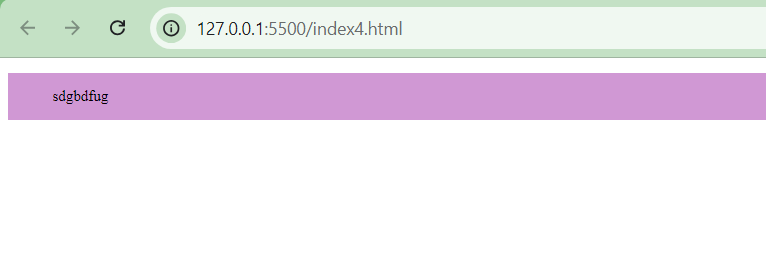
Kết quả:



Nếu giá trị padding của cạnh trái và phải bằng nhau, thì có thể rút gọn thuộc tính padding còn lại ba giá trị. Khi đó, giá trị của cạnh phải (giá trị thứ hai trong chuỗi giá trị) cũng được sử dụng làm giá trị cho cạnh trái. Thực tế là, khi biên dịch mã CSS, trình duyệt thấy thiếu giá trị của cạnh trái nên nó mặc định lấy giá trị của cạnh phải để thay vào.

Cú pháp,

padding: top right/left bottom;

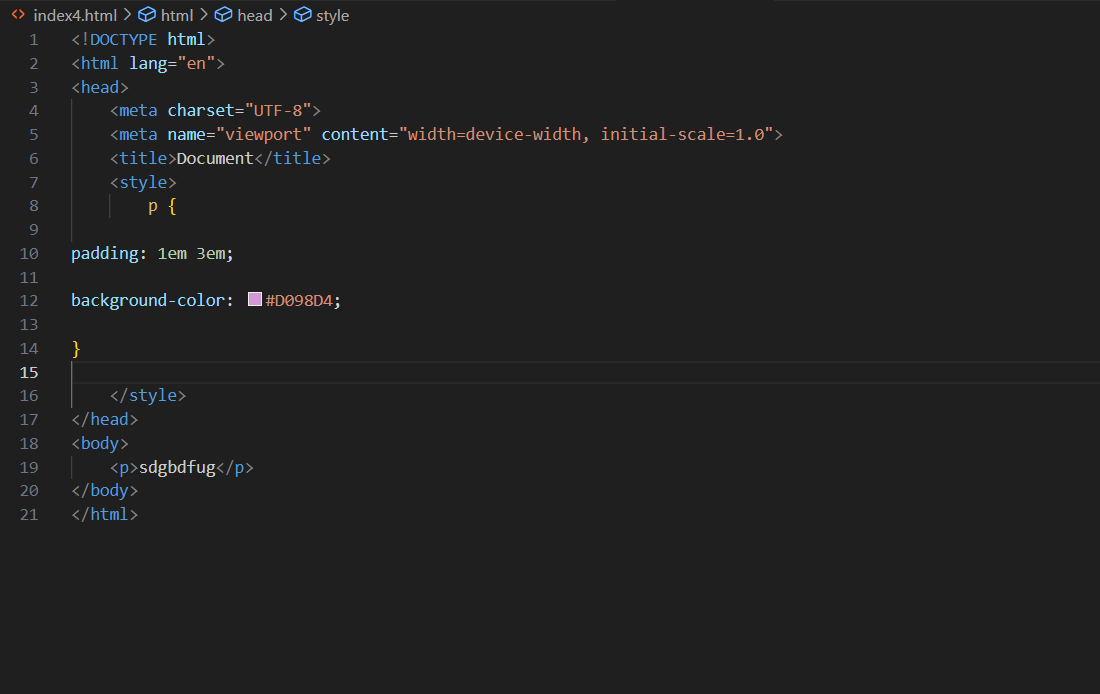


Tiếp tục, nếu chỉ cung cấp hai giá trị cho thuộc tính padding, thì giá trị đầu tiên sẽ là của cạnh trên và dưới, giá trị thứ hai sẽ là của cạnh trái và phải.

Cú pháp,

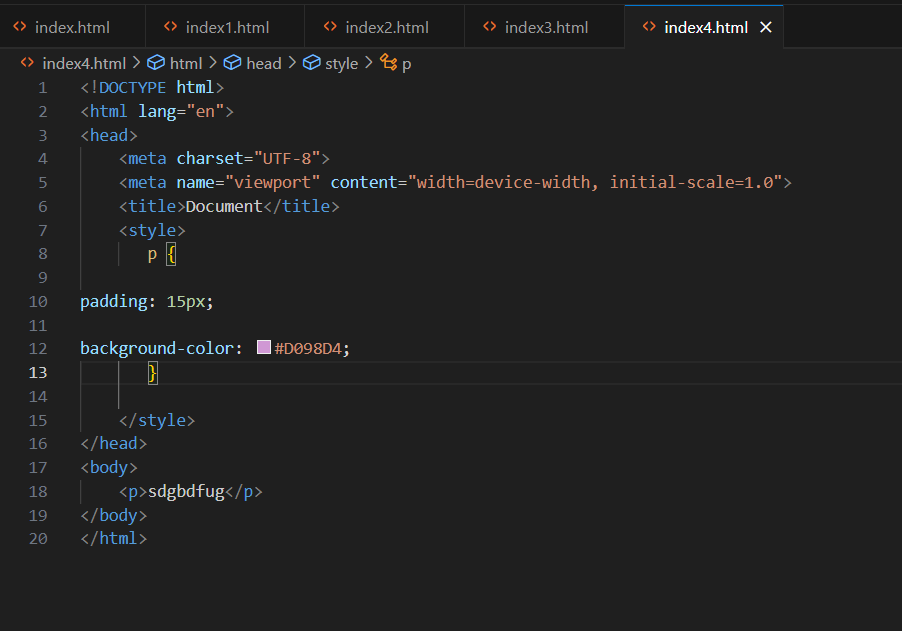
padding: top/bottom right/left;

Ví dụ, chỉ dẫn CSS ở phía trên có thể viết gọn lại như sau,



Cuối cùng, nếu chỉ cung cấp một giá trị cho thuộc tính padding, thì giá trị này sẽ là của cả bốn cạnh.

Ví dụ, chỉ dẫn CSS sau sẽ thiết lập padding cho cả bốn cạnh là 15px,



**1.1.5 Đường viền**

Đường viền là đường bao quanh phần tử (gồm vùng nội dung và vùng đệm trong). Có tám kiểu đường viền, Ngoài ra, có thể định dạng độ dày và màu sắc tùy ý. Có thể thiết lập đường viền xung quanh phần tử, hoặc chỉ thiết lập cho một số cạnh cụ thể. CSS cũng cung cấp các thuộc tính giúp tạo các góc bo tròn (rounding of the corners) và tạo đường viền bằng hình ảnh.

**Các kiểu đường viền**

Ví dụ sau sẽ thiết lập bốn kiểu đường viền khác nhau cho bốn cạnh,

p {

border-top-style: solid;

border-right-style: dashed;

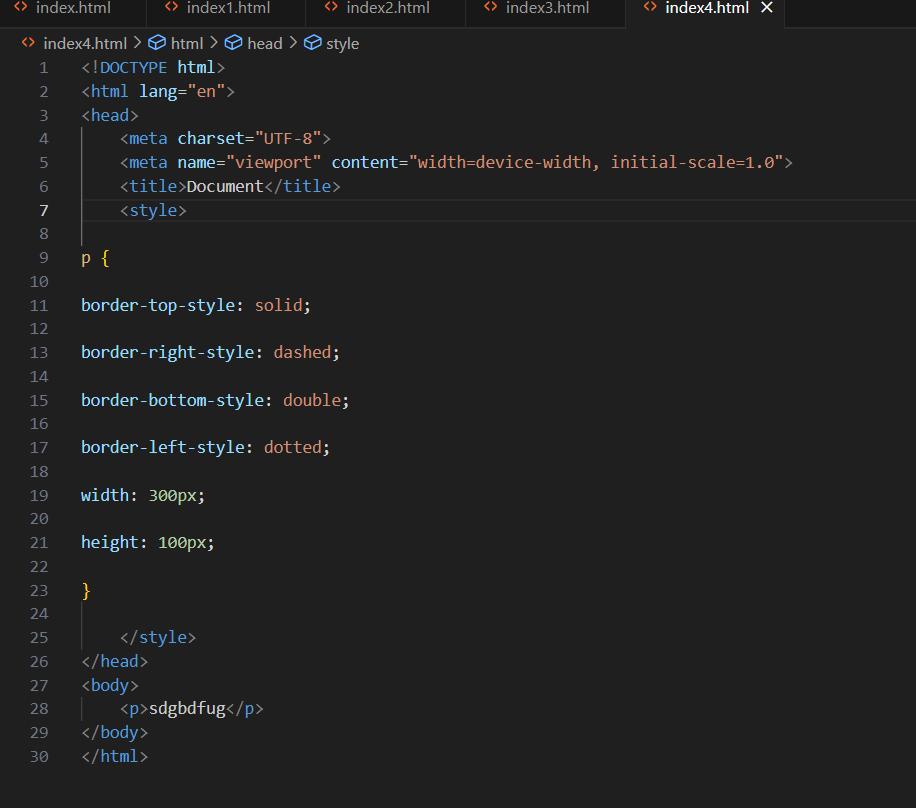
border-bottom-style: double;

border-left-style: dotted;

width: 300px;

height: 100px;

}



Kết quả:



Cũng có thể thiết lập kiểu đường viền cho cả bốn cạnh một lần bằng thuộc tính: border-style,

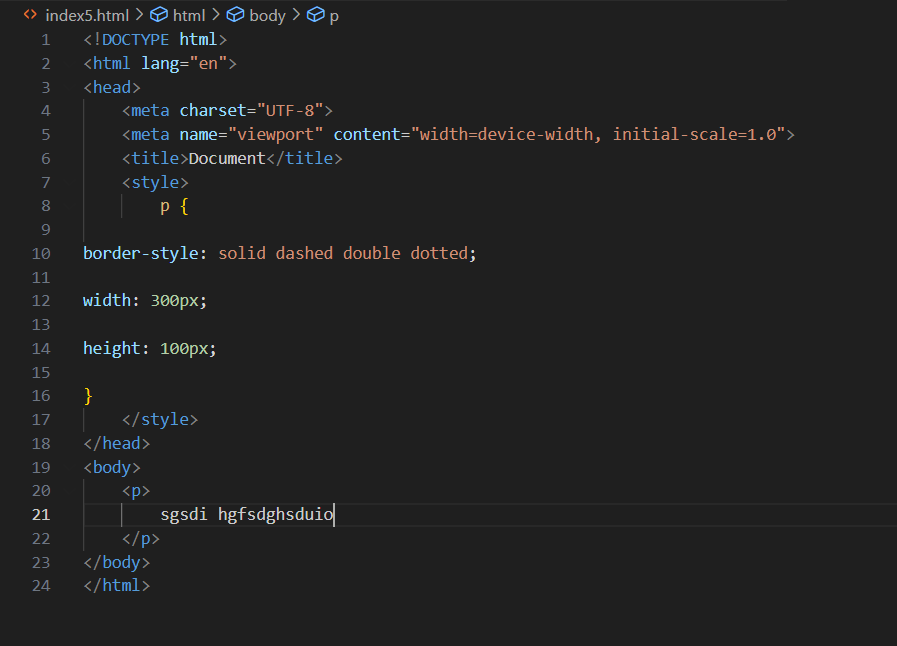
– Giá trị: none | dotted | dashed | solid | double | groove | ridge | inset | outset | hidden

– Mặc định: none

– Áp dụng: mọi phần tử

– Kế thừa: không

Thuộc tính border-style thiết lập kiểu đường viền cho bốn cạnh theo chiều kim đồng hồ (clockwise). Giống như thuộc tính padding đã tìm hiểu, thứ tự các cạnh sẽ được áp dụng sẽ là trên, phải, dưới, trái (để dễ nhớ, sử dụng từ TRouBLe rồi phiên ra thứ tự). Cần cung cấp bốn giá trị cho bốn cạnh này, tuy nhiên, nếu cạnh trên/dưới và phải/trái có kiểu giống nhau thì chỉ cần cung cấp hai giá trị, nếu cả bốn cạnh có kiểu giống nhau thì chỉ cần cung cấp một giá trị.



Kết quả:



**Độ dày của đường viền**

Có thể thiết lập độ dày cho từng cạnh đường viền bằng các thuộc tính: border-top-width, border-right-width, border-bottom-width, border-left-width,

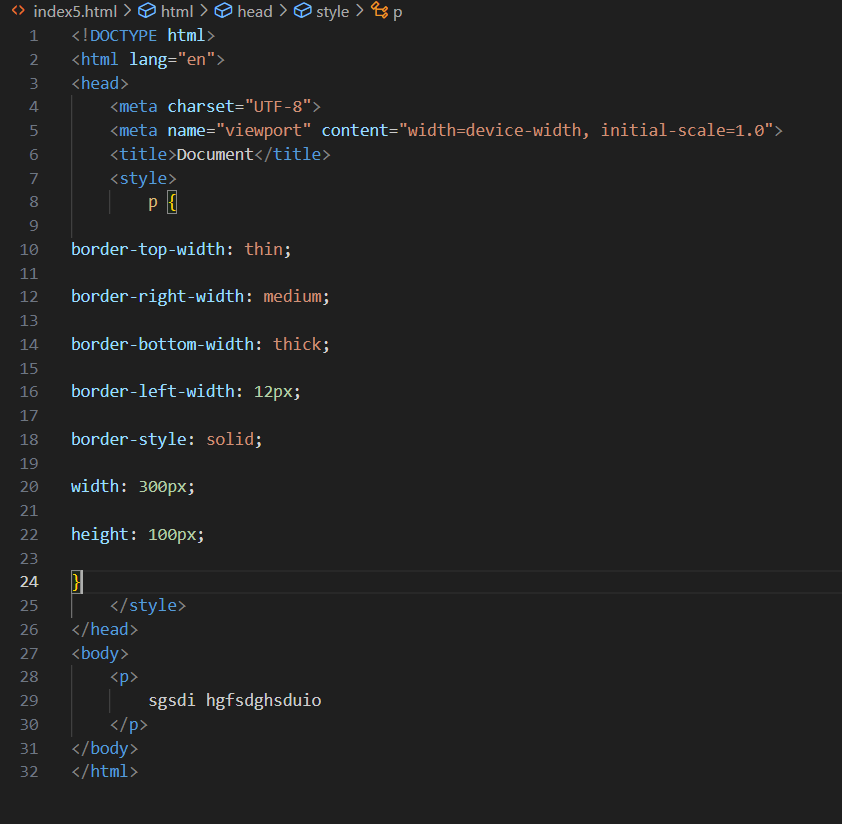
– Giá trị: đơn vị đo độ dày | thin | medium | thick

– Mặc định: medium

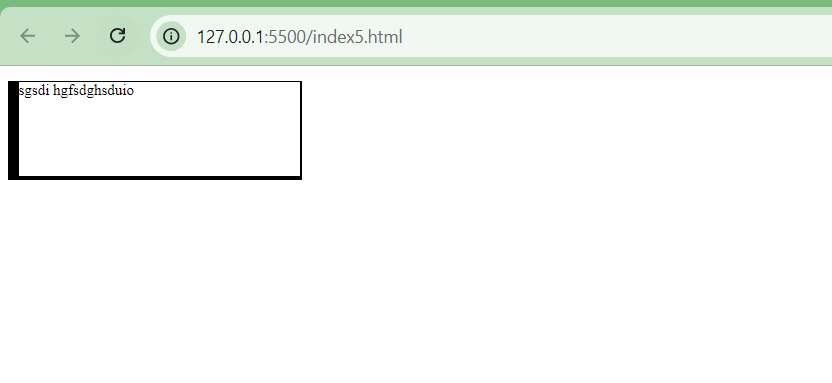
– Áp dụng: mọi phần tử

– Kế thừa: không

Ví dụ sau sẽ thiết lập độ dày cho từng cạnh,



Kết quả:



Đơn vị đo độ dày thường là px hoặc em. Tuy nhiên, cũng có thể sử dụng các từ khóa: thin (mỏng), medium (vừa, trung bình) hoặc thick (dày).

Cũng có thể thiết lập độ dày cho cả bốn cạnh một lần bằng thuộc tính: border-width,

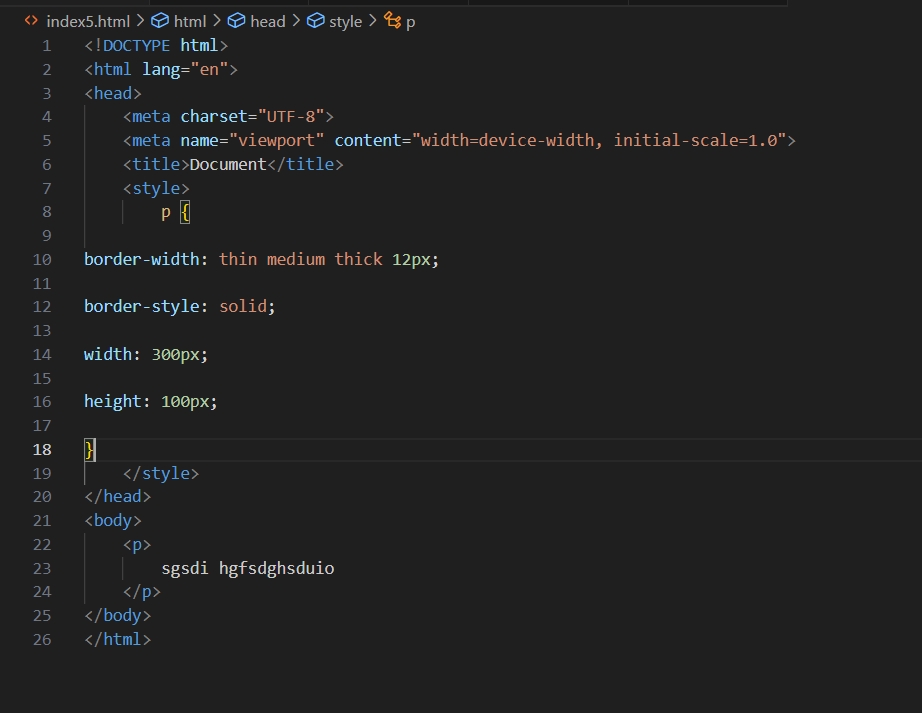
– Giá trị: đơn vị đo độ dày | thin | medium | thick

– Mặc định: medium

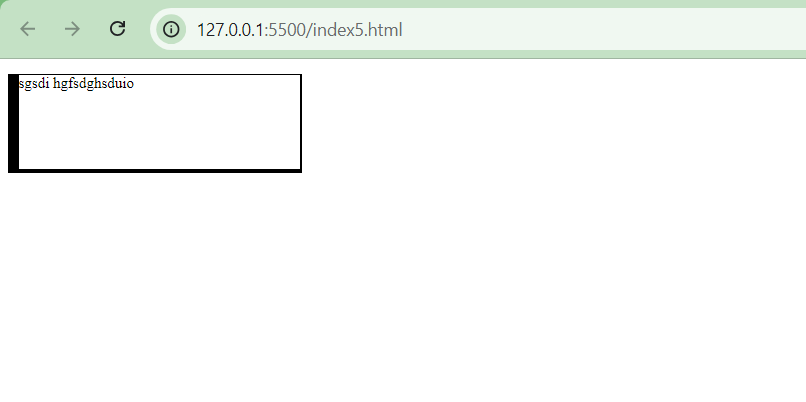
– Áp dụng: mọi phần tử

– Kế thừa: không

Thuộc tính border-width thiết lập độ dày cho bốn cạnh theo chiều kim đồng hồ (clockwise). Giống như thuộc tính border-style, thứ tự các cạnh sẽ được áp dụng sẽ là trên, phải, dưới, trái (để dễ nhớ, sử dụng từ TRouBLe rồi phiên ra thứ tự). Cần cung cấp bốn giá trị cho bốn cạnh này, tuy nhiên, nếu cạnh trên/dưới và phải/trái có độ dày bằng nhau thì chỉ cần cung cấp hai giá trị, nếu cả bốn cạnh có độ dày bằng nhau thì chỉ cần cung cấp một giá trị.



Kết quả:



**Màu của đường viền**

Để thiết lập màu cho đường viền, cũng có thể thiết lập cho từng cạnh, hoặc cho tất cả các cạnh một lần. Khi thiết lập màu cho đường viền, nó sẽ đè lên màu đường viền trước đó (được thiết lập bằng thuộc tính color).

Có thể thiết lập màu cho từng cạnh bằng các thuộc tính: border-top-color, border-right-color, border-bottom-color, border-left-color,

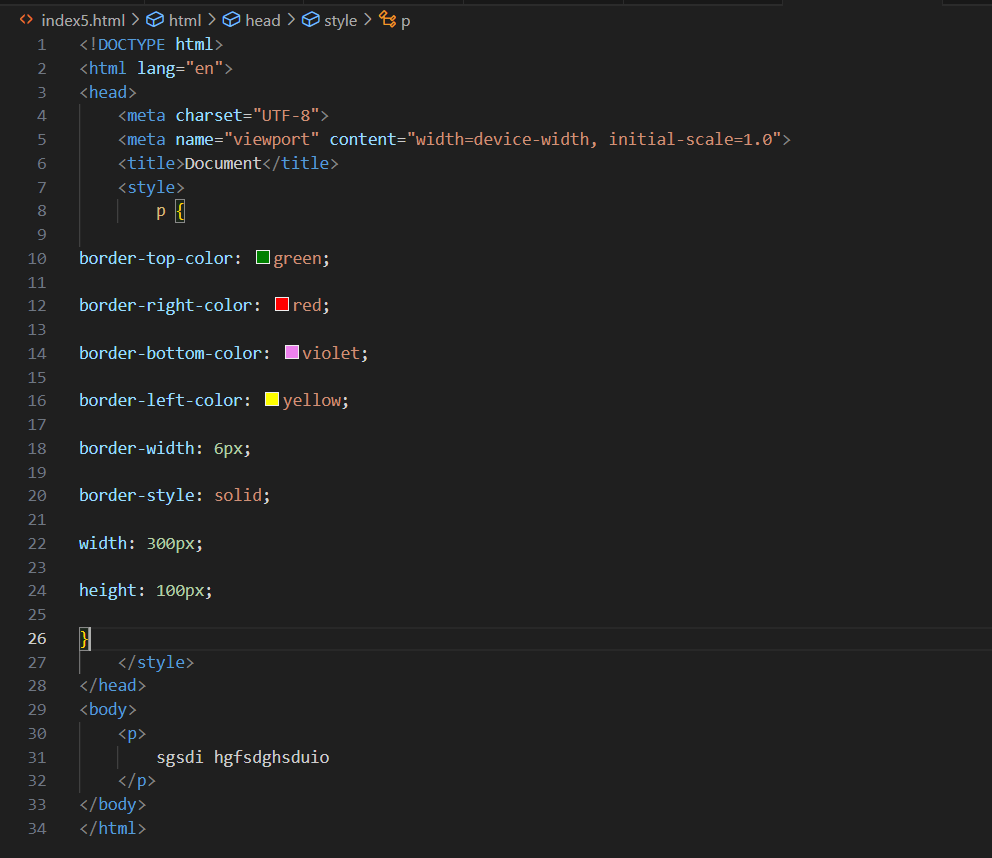
– Giá trị: tên màu hoặc giá trị RGB/HSL | transparent

– Mặc định: màu do thuộc tính color thiết lập

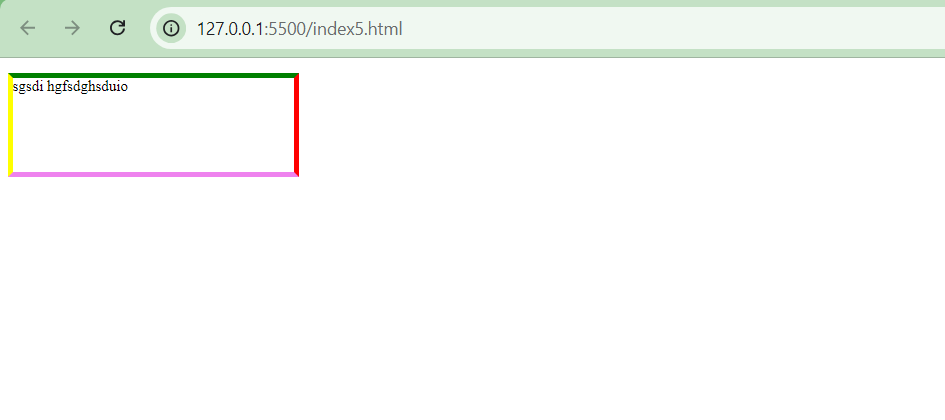
– Áp dụng: mọi phần tử

– Kế thừa: không

Ví dụ sau sẽ thiết lập màu cho từng cạnh,



Kết quả:



Cũng có thể thiết lập màu cho cả bốn cạnh một lần bằng thuộc tính border-color,

– Giá trị: tên màu hoặc giá trị RGB/HSL | transparent

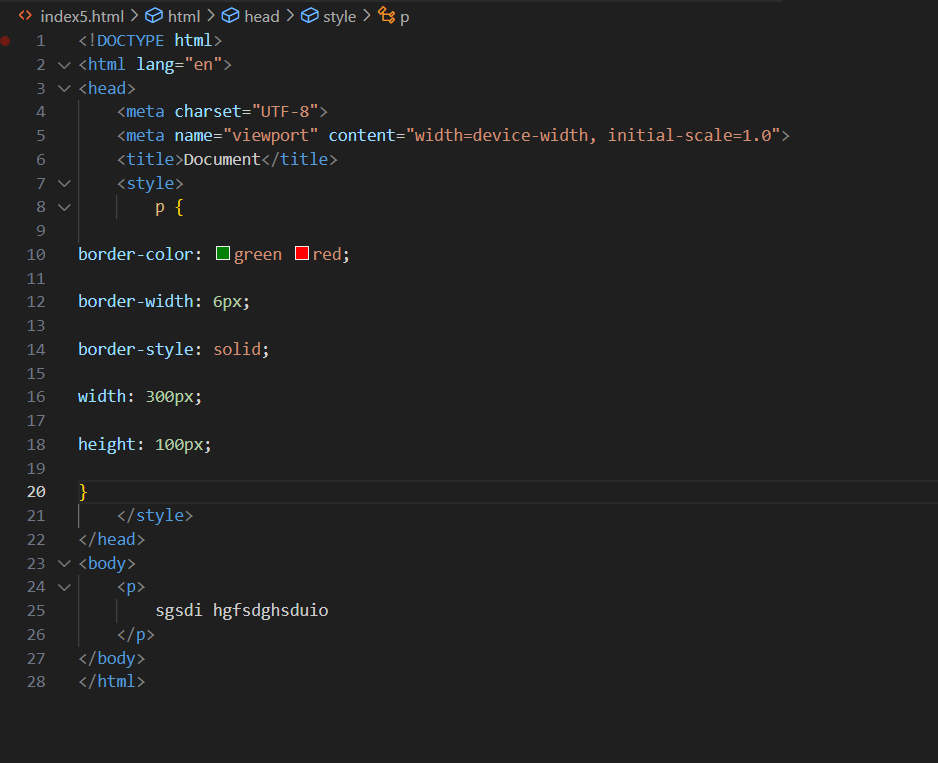
– Mặc định: màu do thuộc tính color thiết lập

– Áp dụng: mọi phần tử

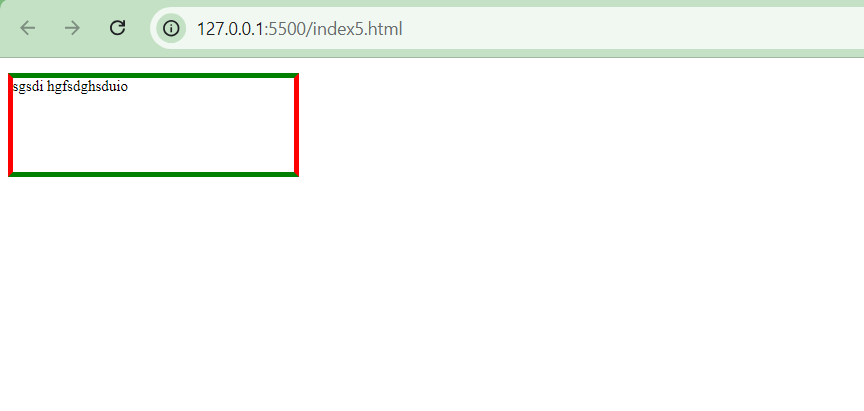
– Kế thừa: không

Thuộc tính border-color thiết lập màu cho bốn cạnh theo chiều kim đồng hồ (clockwise). Giống như thuộc tính border-width, thứ tự các cạnh sẽ được áp dụng sẽ là trên, phải, dưới, trái (để dễ nhớ, sử dụng từ TRouBLe rồi phiên ra thứ tự). Cần cung cấp bốn giá trị cho bốn cạnh này, tuy nhiên, nếu cạnh trên/dưới và phải/trái có màu giống nhau thì chỉ cần cung cấp hai giá trị, nếu cả bốn cạnh có màu giống nhau thì chỉ cần cung cấp một giá trị.

Ví dụ, cạnh trên/dưới có màu xanh, cạnh phải/trái có màu đỏ,



Kết quả:



**Kết hợp kiểu, độ dày và màu**

Để viết mã nhanh hơn, CSS cho phép định dạng cả ba giá trị gồm: kiểu, độ dày và màu một lần cho từng cạnh, hoặc một lần cho cả bốn cạnh.

Để định dạng cho từng cạnh, sử dụng các thuộc tính: border-top, border-right, border-bottom và border-left.

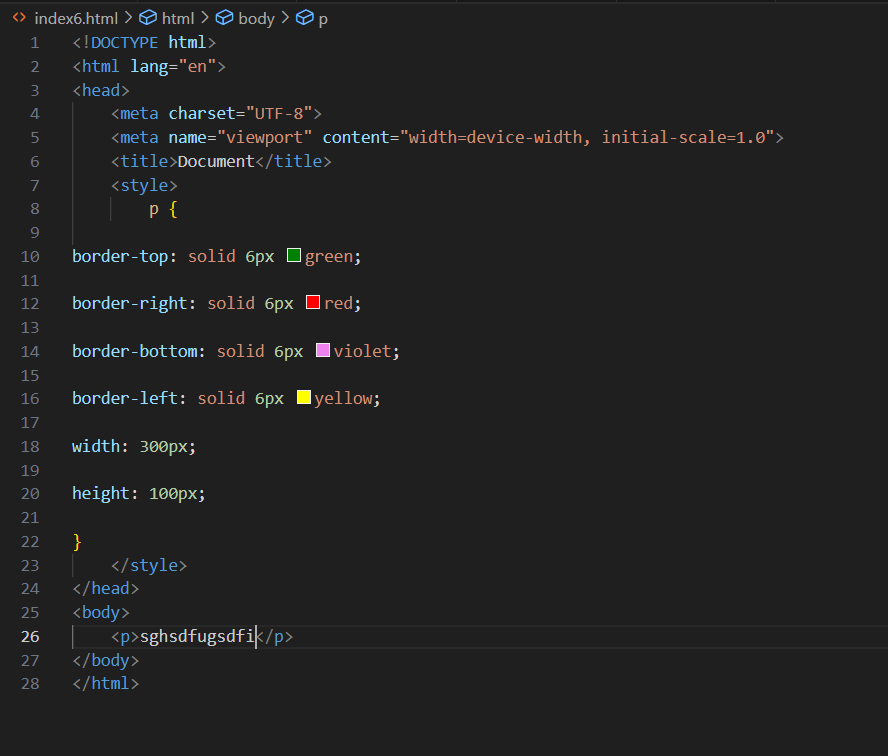
– Giá trị: border-style border-width border-color

– Mặc định: giá trị mặc định của mỗi thuộc tính

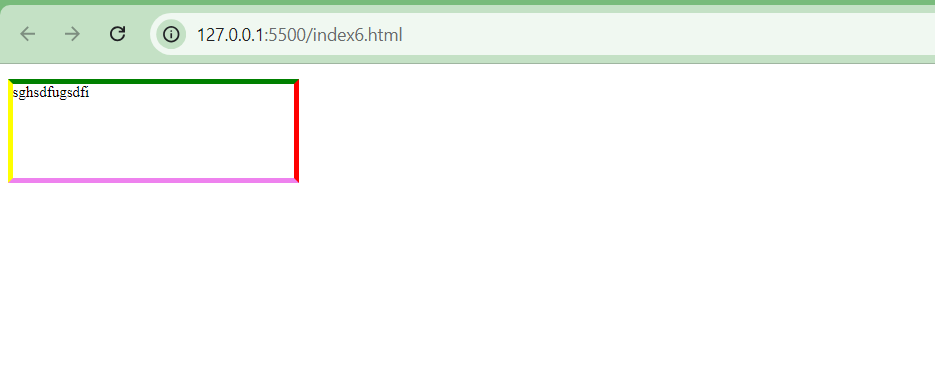
– Áp dụng: mọi phần tử

– Kế thừa: không

Ví dụ,



Kết quả:



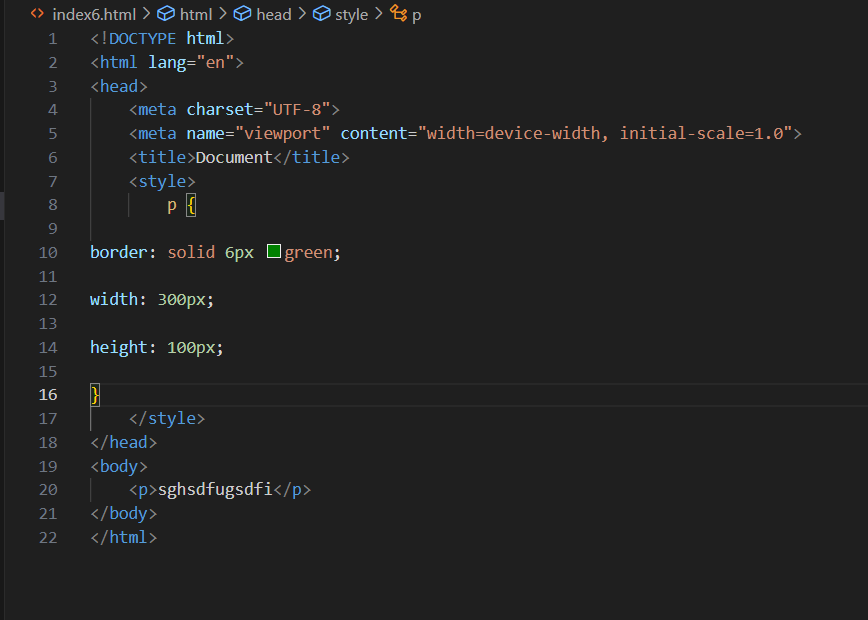
Để định dạng một lần cho cả bốn cạnh, sử dụng thuộc tính border, các giá trị của thuộc tính border sẽ luôn luôn áp dụng trên cả bốn cạnh,

– Giá trị: border-style border-width border-color

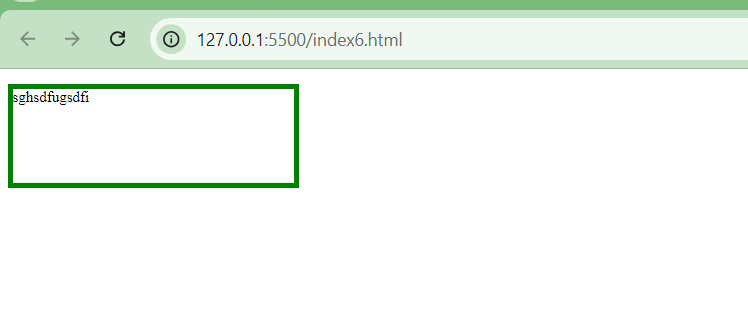
– Mặc định: giá trị mặc định của mỗi thuộc tính

– Áp dụng: mọi phần tử

– Kế thừa: không



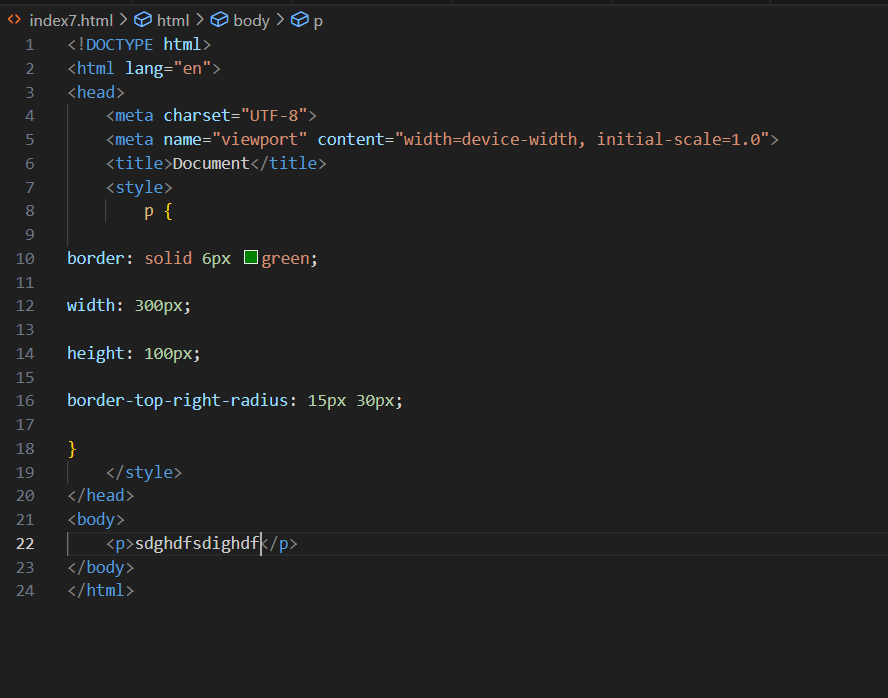
Kết quả:



**Tạo góc bo elipse**

Để tạo góc bo elipse, cần cung cấp hai giá trị là bo theo chiều ngang (horizontal) và bo theo chiều dọc (vertical).

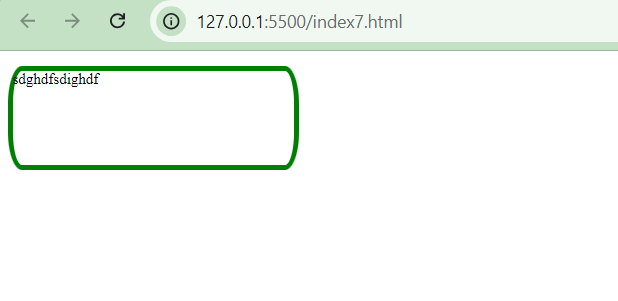
Ví dụ,



Kết quả:

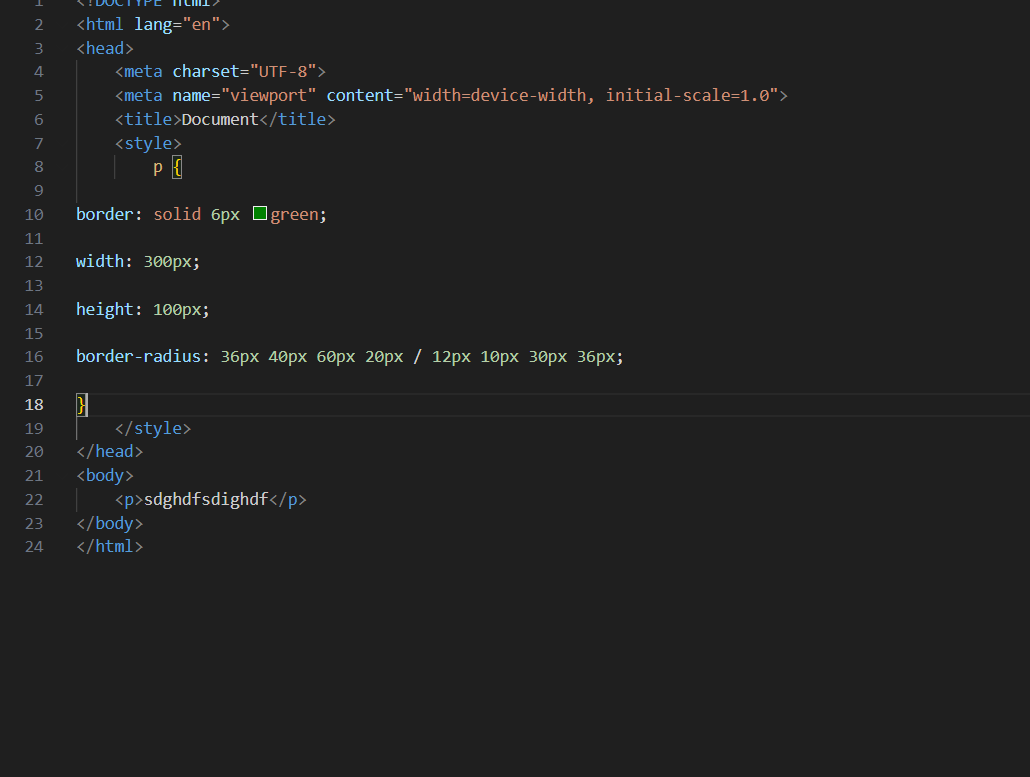


Nếu sử dụng thuộc tính border-radius thì giá trị của chiều ngang và chiều dọc sẽ ngăn cách nhau bằng dấu xuyệt (/).

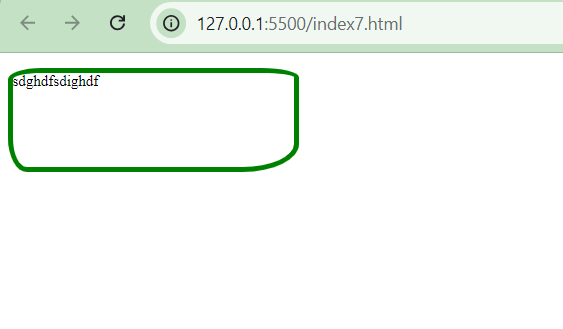
Ví dụ, 

Tuy nhiên, để xác định chiều ngang và chiều dọc cho từng góc, các giá trị chiều ngang sẽ viết liền nhau, các giá trị chiều dọc sẽ viết liền nhau, ngăn cách nhau bằng dấu xuyệt.

Ví dụ,



Kết quả:



**Tạo đường viền bằng hình ảnh**

Để tạo đường viền và bo góc bằng hình ảnh, sử dụng thuộc tính border-image.

– Giá trị: border-image-source border-image-slice border-image-width border-image-outset border-image-repeat

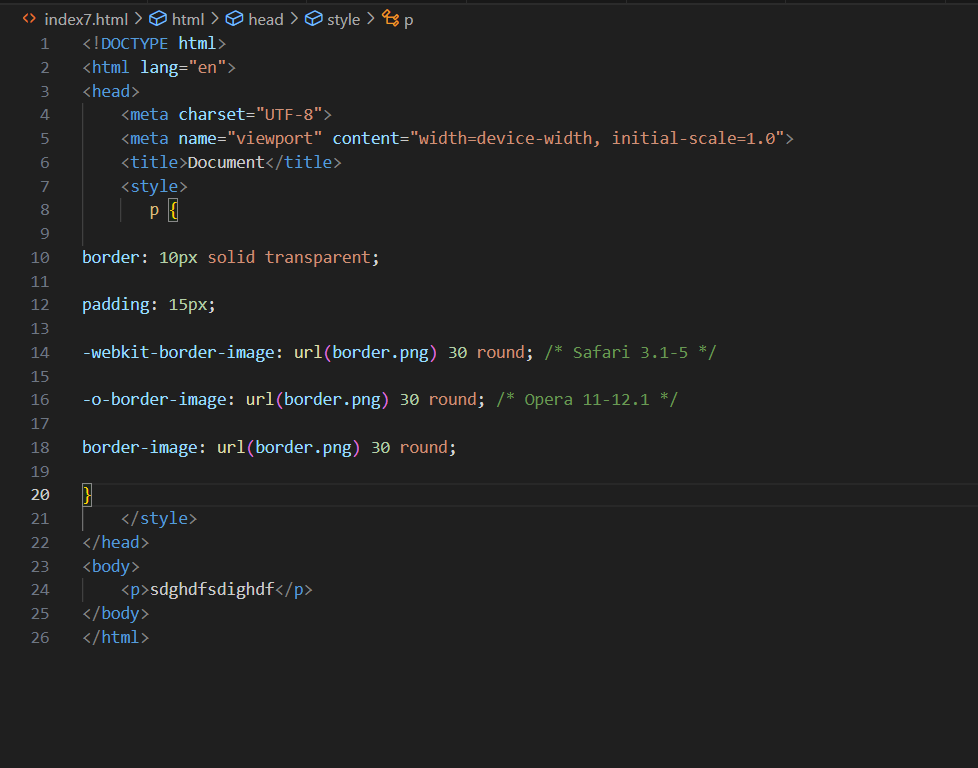
– Mặc định: giá trị mặc định của mỗi thuộc tính

– Áp dụng: mọi phần tử ngoại trừ phần tử table có border-collapse là collapse

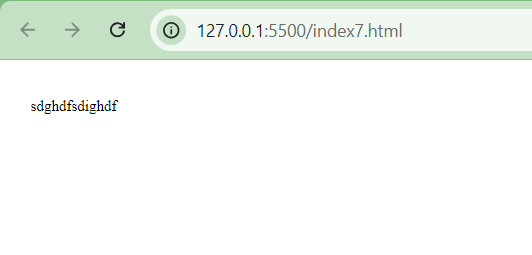
– Kế thừa: không

Để dễ hiểu, quan sát hình minh họa dưới đây.

Ví dụ, tạo đường viền bằng hình ảnh, có thêm tiếp đầu ngữ -webkit và -o để đoạn mã làm việc tốt trên cả hai trình duyệt Safari và Opera tương ứng,

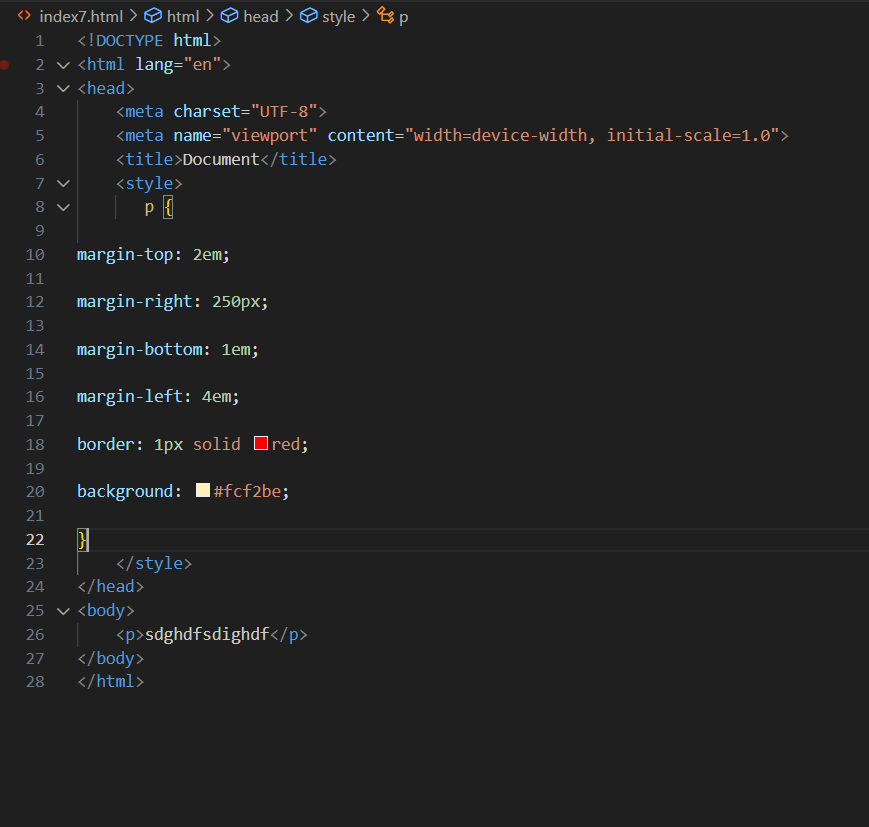


Kết quả:

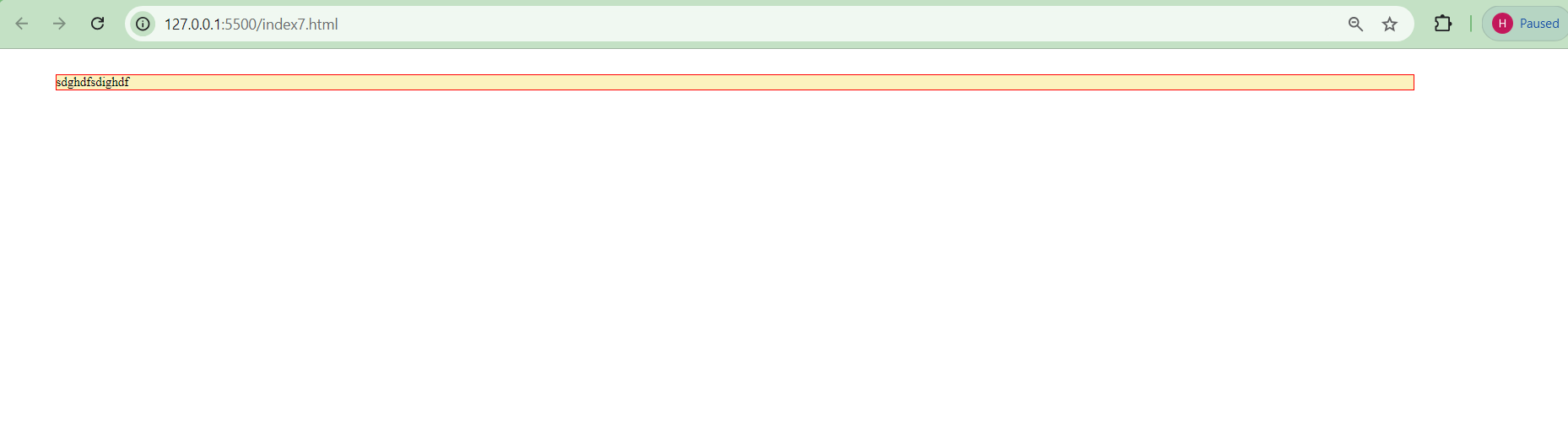


**1.1.6 Đường viền**

Thành phần cuối cùng của mô hình cái hộp là “vùng đệm ngoài” (margin), đây là vùng tùy chọn, là khoảng-không được thêm vào phía ngoài của đường viền (border).

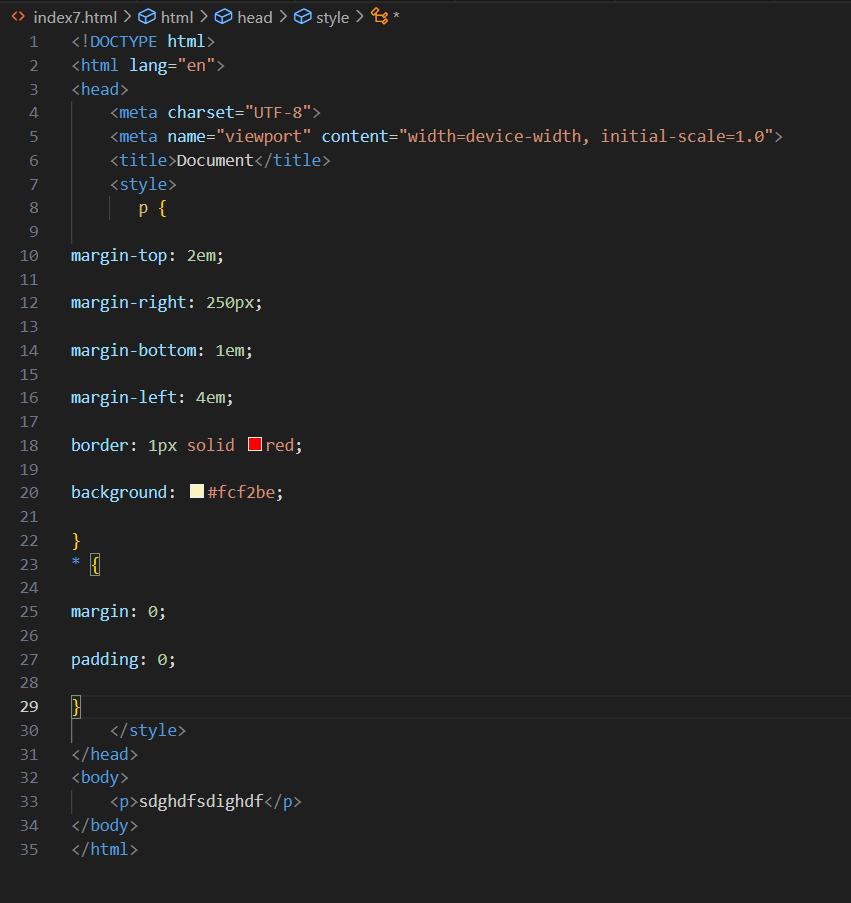


Kết quả:

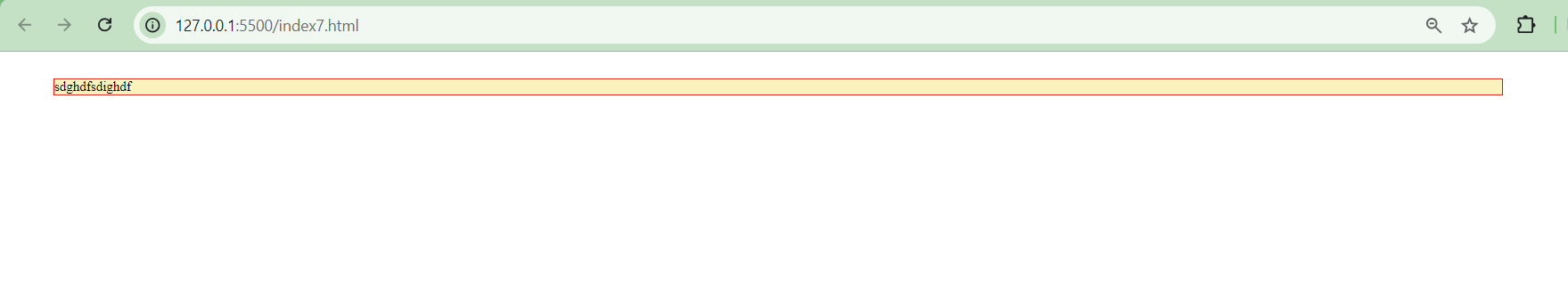


Đơn vị để thiết lập margin có thể là em, px hoặc % (phần trăm chiều rộng của phần tử cha).

Lưu ý, khi thiết lập margin là %, nếu chiều rộng của phần tử cha thay đổi, sẽ dẫn tới margin bị thay đổi theo, điều này làm cho việc kiểm soát bố cục phức tạp.

Từ khóa auto là chế độ thiết lập vùng đệm ngoài mặc định của trình duyệt. Với chế độ này, nhiều khi làm cho người thiết kế web không thể kiểm soát chính xác được bố cục, giải pháp là đặt lại (reset) giá trị margin và padding của mọi phần tử về 0. 

Kết quả:

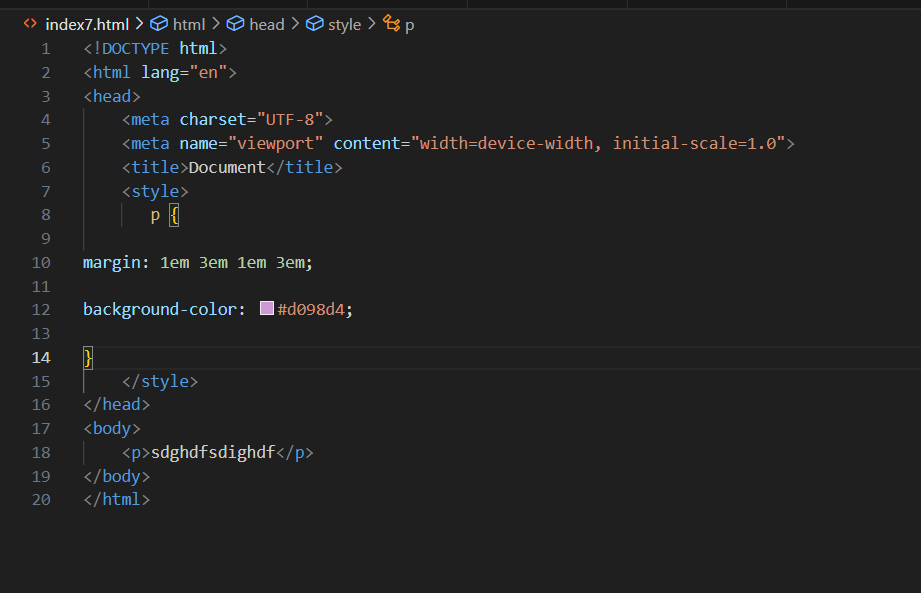


**Thuộc tính margin có thể chấp nhận bốn, ba, hai, hoặc một giá trị.**

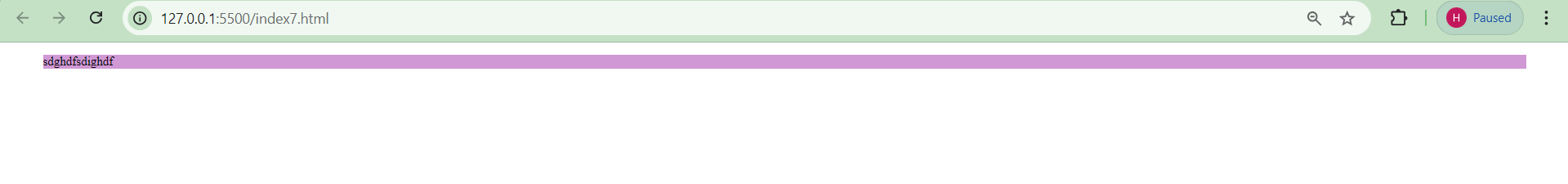
Khi margin có bốn giá trị, các giá trị này sẽ được áp dụng lần lượt cho các cạnh tương ứng, theo chiều kim đồng hồ, bắt đầu là cạnh trên (top), sau đó đến cạnh phải (right), cạnh dưới (bottom), cuối cùng là cạnh trái (left). Để dễ nhớ, tiếng Anh có từ TRouBLe, mỗi chữ viết hoa sẽ ứng với một cạnh.

Cú pháp,

margin: top right bottom left;



Kết quả:



Nếu giá trị margin của cạnh trái và phải bằng nhau, thì có thể rút gọn thuộc tính margin còn lại ba giá trị. Khi đó, giá trị của cạnh phải (giá trị thứ hai trong chuỗi giá trị) cũng được sử dụng làm giá trị cho cạnh trái. Thực tế là, khi biên dịch mã CSS, trình duyệt thấy thiếu giá trị của cạnh trái nên nó mặc định lấy giá trị của cạnh phải để thay vào.

Cú pháp,

margin: top right/left bottom;

Ví dụ, chỉ dẫn CSS ở phía trên có thể viết gọn lại như sau,

Cuối cùng, nếu chỉ cung cấp một giá trị cho thuộc tính margin, thì giá trị này sẽ là của cả bốn cạnh.

Ví dụ, chỉ dẫn CSS sau sẽ thiết lập margin cho cả bốn cạnh là 15px,

p {

margin: 15px;

background-color: #d098d4;

}